Errata 01 for
MIPI I3C Basic Specification
(Improved Inter Integrated Circuit)
Specification Version 1.2
Specification Dated 16 December 2024
Specification MIPI Board Adopted 17 April 2025

Errata 01 Dated 13 March 2026
Errata MIPI Board Approved 14 April 2026

* IMPORTANT NOTE TO IMPLEMENTERS *

- The issues listed in this Errata document will be corrected in the next edition of this MIPI Specification.
- Implementations should observe all Corrections listed here.
- The location of each Correction is also marked in the attached copy of the MIPI Specification. To reduce the risk of incorrect implementations, we suggest you consider discarding any previous copies of this MIPI Specification not so marked.
- This MIPI Specification as modified by the corrections listed in this Errata document is also a MIPI Specification, as the MIPI Bylaws defines the term.
- **MIPI member companies' rights and obligations apply to the modified MIPI Specification as defined in the MIPI Membership Agreement and MIPI Bylaws.**

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

<table>
<tr><th>Item</th><th>Spec Page Number</th><th>PDF Page Number</th><th>Correction</th></tr>
<tr><td>1</td><td>viii – x</td><td>30 – 32</td><td>Editorial or Technical: Editorial
Location: In Figures table of contents, entries for Figures 104 through 195 (inclusive)
Correction: Replace consecutive Figure numbers 104 through 195 (inclusive) with consecutive numbers 103 through 193 (inclusive).
Reason: To correct a document production error. (The figure titles are correct, only the figure numbers are wrong.)
Technical Impact: None</td></tr>
<tr><td>2</td><td>xiv</td><td>36</td><td>Editorial or Technical: Editorial
Location: Release History table, rows v1.0, v1.1, and v1.1.1
Corrections:
[1] In row v1.0, column Date, replace “31-Dec-2018” with “08-Oct-2018”
[2] Delete row v1.1 (there was no v1.1 release of I3C basic)
[3] In row v1.1.1, column Date, replace “11-Jun-2021” with “21-Jul-2021”.
Reason: To correct a document drafting error.
Technical Impact: None</td></tr>
<tr><td>3</td><td>3</td><td>39</td><td>Editorial or Technical: Editorial
Location: Section 1, Table 1, after row “Annex F”
Correction:
Add the two table rows shown below.
Reason: To correct a document drafting omission.
Technical Impact: None</td></tr>
<tr><td colspan="4"><table>
<tr><th>Sections</th><th>Topics</th></tr>
<tr><td>Annex G</td><td>MIPI I3C Basic Specification
Supplemental Patent Licensing Terms</td></tr>
<tr><td>Annex H</td><td>I3C Basic Development Companies</td></tr>
</table></td></tr>
<tr><td>4</td><td>17</td><td>53</td><td>Editorial or Technical: Editorial
Location: Section 2.4.1, Line 476
Correction:
Change: “Adoped 21 July 2021”
To: “Adopted 23 July 2021”
Reason: To correct a typo and the adoption date.
Technical Impact: None</td></tr>
<tr><td>5</td><td>47</td><td>83</td><td>Editorial or Technical: Editorial
Location: Section 4.3.2.2.2, Line 1037
Correction:
Change: “I3I3C Basic C Bus”
To: “I3C Basic Bus”
Reason: To correct two typos.
Technical Impact: None</td></tr>
</table>

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

6	51	87	**Editorial or Technical:** Editorial **Location:** Section 4.3.2.2.5, Table 10, row 110 1111 / 111 0101 **Correction:** **Change:** “7vh75” **To:** “7’h75” **Reason:** To correct a typo. **Technical Impact:** None
7	74	110	**Editorial or Technical:** Technical **Location:** ***Section 4.3.5.3***, Lines 1897–1900 **Correction:** **Change:** A passive Hot-Joining Target will first wait for an SDR Frame addressed to the Broadcast Address (7'h7E/W) ending in STOP, and then wait for the next START, which must be issued by either the Active Controller or any other Target that can request a START. This ensures that the Target is on an I3C Bus. **To:** A passive Hot-Joining Target will first wait for an SDR Frame addressed to the Broadcast Address (7'h7E/W) to ensure it is on an I3C Bus, and then either: **Strict Mode:** Wait for a STOP (i.e., the end of the first SDR Frame), and then wait for the next START (which must be issued by either the Active Controller or any other Target that can request a START), and then issue the Hot-Join Address (i.e., the reserved Target Address, 7'h02) into the Arbitrable Address Header to emit its Hot-Join Request, in the same manner as a standard Hot-Joining Target. or **Relaxed Mode:** Respond to the next ENTDAA CCC sent by the Active Controller, as soon as it knows that it is indeed on an I3C Bus. **Reason:** To relax the requirements for a passive Hot-Joining Target that can detect whether it is on an I3C Bus and then respond to the first use of the ENTDAA CCC that it sees (i.e., after a recognizable 7’h7E/W pattern) without needing to use the Hot-Join Request. This reduces delay for certain types of I3C Targets that are used in systems where both the Target(s) and the Controller are powered on at the same time. **Technical Impact:** Medium
8	74	110	**Editorial or Technical:** Technical **Location:** ***Section 4.3.5.3***, Lines 1913–1915 **Correction:** Delete lines 1913–1915 **Reason:** Because these lines apply only to Strict Mode (see Errata 01 Item 7). **Technical Impact:** See Errata 01 Item 7.

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

9	74	110	**Editorial or Technical:** Technical **Location:** ***Section 4.3.5.3***, Lines 1923–1924. **Correction:** **Change:** *…(such as ENEC/DISEC), except for the ENTDAA CCC since it would not be eligible to participate in Dynamic Address Assignment before it emits the Hot-Join Request.* **To:** *…(such as ENEC/DISEC). If such a Target follows Strict Mode, then it will not process the ENTDAA CCC, since it would not be eligible to participate in Dynamic Address Assignment until it emits the Hot-Join Request. However, if such a Target follows Relaxed Mode, then it will process the ENTDAA CCC, since it would become eligible as soon as it determines that it is on an I3C Bus. If a passive Hot-Joining Target does not see the ENTDAA, SETDASA or SETAASA CCCs and is not assigned a Dynamic Address, then it must still emit the Hot-Join Request in order to inform the Active Controller that it has joined the I3C Bus and is waiting to receive a Dynamic Address.* **Reason:** To clarify the relaxed requirements for a passive Hot-Joining Target, in addition to the corrections from Errata 01 Item 7. **Technical Impact:** Medium
10	153	189	**Editorial or Technical:** Technical **Location:** ***Section 4.3.8.1.8***, Line 3231 **Correction:** **Change:** …the Target acting as Controller may… **To:** …the Controller acting as a Target may… **Reason:** To correct an authoring error. The uncorrected text is inconsistent with other text in the same section describing the type of device and its current role on the I3C Bus. Initially, the Controller acts as Target when it initiates the Error Type DBR procedure, before it determines that there is no Active Controller on the I3C Bus. **Technical Impact:** Low
11	156	192	**Editorial or Technical:** Editorial **Location:** ***Section 4.3.8.2.4***, Lines 3312 and 3315 **Correction:** **Change:** Bullet points #5 and #6 **To:** Bullet points #1 and #2 **Reason:** To correct a typo. These bullet points are the start of a new procedure, not a continuation of any previous procedure. **Technical Impact:** None

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

12	161	197	**Editorial or Technical:** Editorial **Location:** ***Section 4.3.9.1***, Line 3514 **Correction:** Remove extra closing parenthesis [“)”] at the end of the sentence. **Reason:** To correct a typo. **Technical Impact:** None
13	246	282	**Editorial or Technical:** Technical **Location:** ***Section 6.2.3.2***, ***Figure 105*** **Correction:** Replace Figure 105 with the figure on the following page. **Reason:** To correct an authoring error. The uncorrected Figure 105 shows Bit 0 in the payload of the HDR-DDR Command Word as a Reserved bit, whereas it is actually a Parity Adjustment bit per the Table 82 definition. This inconsistency could potentially mislead implementers to always treat Bit 0 as reserved (i.e., value 1'b0), resulting in unexpected parity errors which would only be found during interoperability testing. ***Note:*** *It is safe for I3C Controllers that comply with v1.2 of the I3C Basic Specification, as amended by this Errata 01, to use the correct Parity Adjustment bit value, since Table 82 does have the correct definition, and the uncorrected Figure 105 does show this bit as Reserved (i.e., not a fixed value of 1'b0).* **Technical Impact:** Medium

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

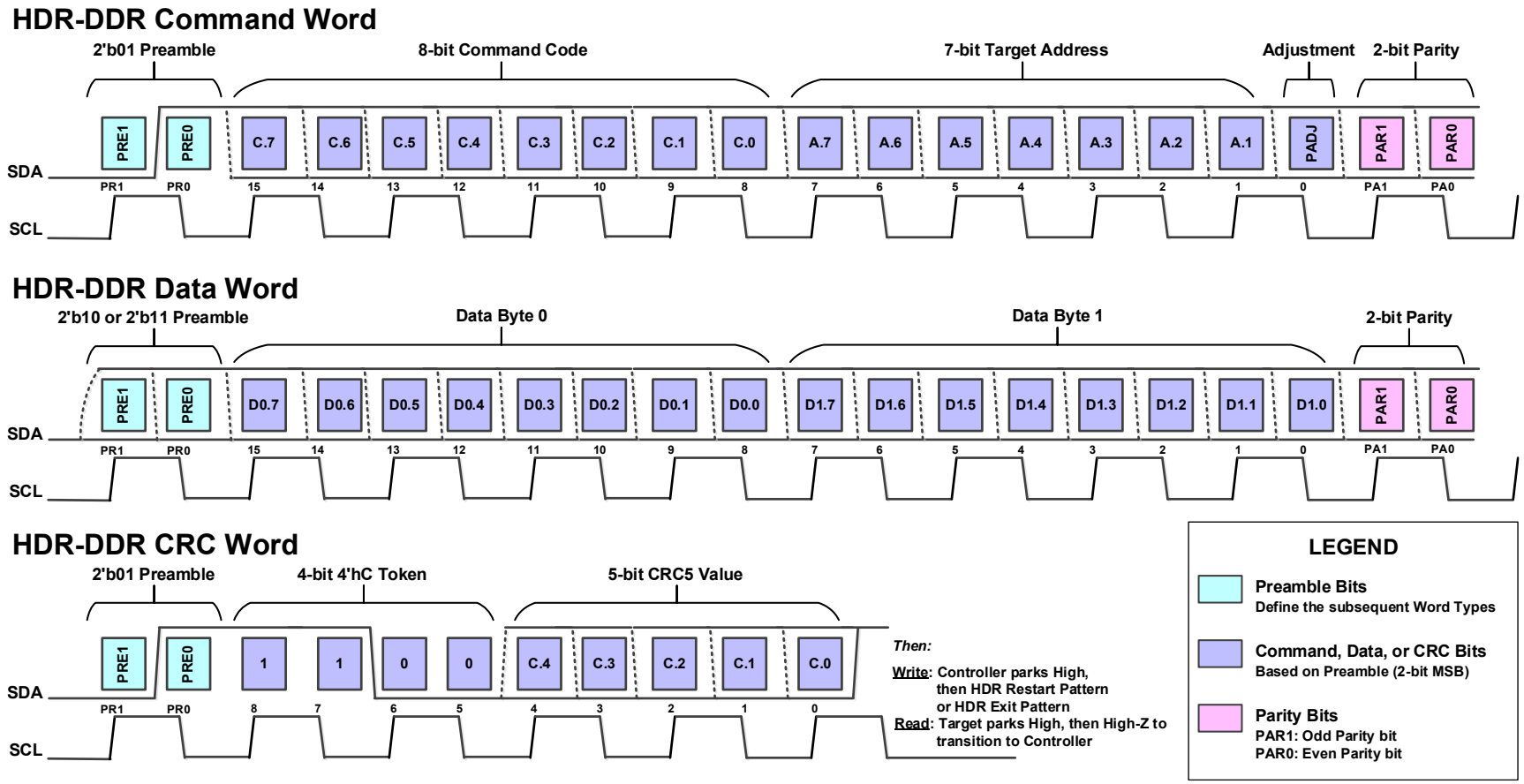


Figure 105 HDR-DDR Word Formats

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

Item	Spec Page Number	PDF Page Number	Correction
14	250	286	**Editorial or Technical:** Technical **Location:** ***Section 6.2.3.2***, ***Table 81***, row Command, column Payload (16 bits) **Correction:** **Change:** Reserved (1 bit) **To:** Parity Adjustment (1 bit) / See ***Table 82*** **Reason:** To correct an authoring error: the uncorrected Table 82 defines Bit 0 in the payload of the HDR-DDR Command Word as a reserved bit, but this is incorrect, i.e., is inconsistent with the normative definitions of the HDR-DDR Command Word. Per Table 82 (and other places), Bit 0 is actually a Parity Adjustment bit. The uncorrected text could potentially mislead implementers to always treat Bit 0 as reserved (i.e., value 1'b0), resulting in unexpected parity errors that would only be found during interoperability testing. **Technical Impact:** Medium
15	251	287	**Editorial or Technical:** Technical **Location:** ***Section 6.2.3.2***, ***Figure 107*** **Correction:** Replace Figure 107 with the figure on the following page. **Reason:** To correct an editing error during v1.2 drafting: an updated version of Figure 108 was mistakenly used to replace the previous version of Figure 107 too. **Technical Impact:** Medium

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

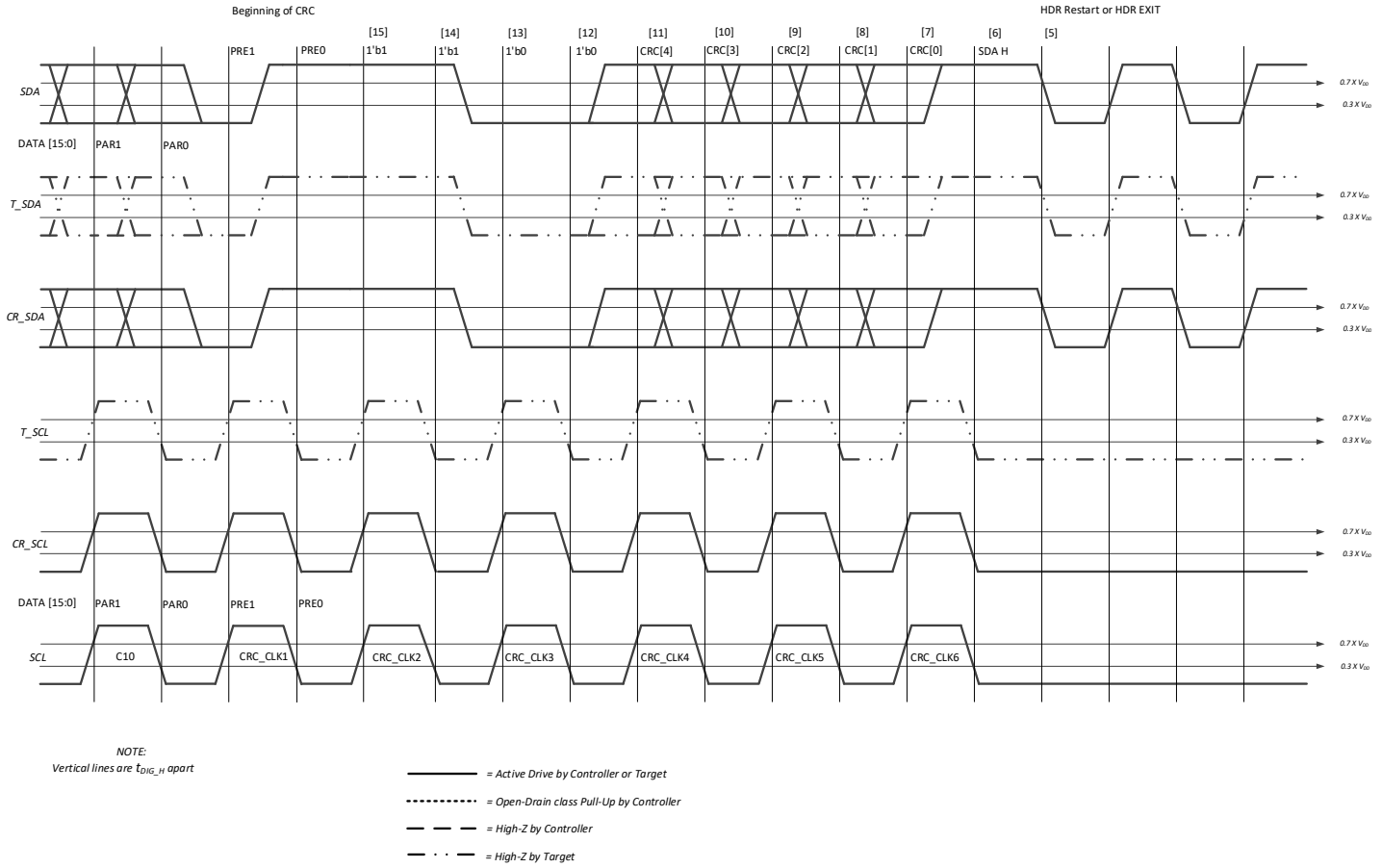


Figure 107 HDR-DDR CRC and HDR Restart Pattern or HDR Exit Pattern in Write Transaction

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

Item	Spec Page Number	PDF Page Number	Correction
16	262	298	**Editorial or Technical:** Technical **Location:** ***Section 6.2.3.3.1.5***, Line 4817 **Correction:** **Change:** Upon receiving the HDR-DDR CRC Word the Target shall… **To:** After transmitting the HDR-DDR CRC Word, the Target shall… **Reason:** To correct an authoring error. During Direct Read CCCs, the Target sends the CRC Word at the end of the data transfer (i.e., it does not receive it). **Technical Impact:** Low

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

<table>
<tr>
<td>17</td>
<td>266</td>
<td>302</td>
<td>
Editorial or Technical: Technical
Location: Section 6.2.3.4, Lines 4900–4903
Correction:
Change last part of Note:
…However, all of the flow control procedures defined in this section and its sub-sections are mandatory for Targets that support HDR-DDR Mode and also comply with version 1.1 or later of the I3C Specification, as well as version 1.1.1 or later of the I3C Basic Specification.
To new body paragraph and new Table 83A:
Table 83A shows which flow control elements are required, conditionally required, and optional for Targets that support HDR-DDR Mode and also comply with either (A) v1.1 or later of the I3C Specification, or (B) v1.1.1 or later of the I3C Basic Specification.
Table 83A Flow Control Element Requirements for HDR-DDR Mode (I3C Basic v1.1.1+)
<table>
<tr><th>Flow Control Element</th><th>Required / Conditional / Optional</th><th>Notes</th></tr>
<tr><td>Support for ENDXFER CCC</td><td>Required</td><td>Defining Bytes 0xF7 and 0xAA.</td></tr>
<tr><td>Write ACK/NACK</td><td>Required</td><td>Must be enabled with ENDXFER CCC before use.</td></tr>
<tr><td>Read transfer early termination</td><td>Required</td><td>Always allowed, no enabling necessary.</td></tr>
<tr><td>Write transfer early termination</td><td>Optional</td><td>Target shall indicate support in Bit[6] of GETCAP2 byte (HDR-DDR Write Abort).</td></tr>
<tr><td>Receive CRC Word after write transfer early termination</td><td>Conditional</td><td>Required if write transfer early termination is supported.
Must be enabled with ENDXFER CCC before use.</td></tr>
<tr><td>Send CRC Word after read transfer early termination</td><td>Optional</td><td>Target shall indicate support in Bit[7] of GETCAP2 byte (HDR-DDR Abort CRC).
Must be enabled with ENDXFER CCC before use.</td></tr>
</table>
Reason: To correct an authoring error during v1.2 drafting. The description of which HDR-DDR flow control methods are mandatory for Targets to support is inconsistent technically with normative behavior defined elsewhere in Section 6.2.3.4 and Section 5.4.
Technical Impact: High
</td>
</tr>
</table>

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

18	266	302	**Editorial or Technical:** Technical **Location:** ***Section 6.2.3.4***, Lines 4912–4913 **Correction:** **Change:** However, if this occurs, then the Target is always obligated to send the CRC Word after the last Data Word. **To:** However, if this occurs, then per ***Section 6.2.3.4.5*** the Target will not send the CRC Word after the last Data Word. **Reason:** To correct an authoring error during v1.2 drafting. The description of Target behavior when an HDR-DDR read transfer is terminated is incorrect (since the Target is required to not send the CRC Block in this case) and inconsistent with normative behavior defined in ***Section 6.2.3.4.5***. **Technical Impact:** High
19	277	313	**Editorial or Technical:** Editorial **Location:** ***Section 6.2.3.4.4***, Lines 5082–5085 **Correction:** **Change:** If the Controller determines that the set-up was successful, then it shall then send the ENDXFER CCC with sub-command 0xAA. If the Direct version of ENDXFER is used, then RnW is set to 1'b0 (WRITE) for the directly addressed Devices and the data is set to 0xAA (i.e., the sub-command value is reused for the databyte value, to facilitate error detection). **To:** If the Controller determines that the set-up was successful, then it shall then send the ENDXFER CCC with sub-command 0xAA. This may be sent as either a Broadcast CCC (i.e., to all Targets) or a Direct CCC (i.e., to specific Targets or Groups). If sent as a Direct CCC, then the Controller shall set RnW to 1'b0 (WRITE). Regardless of the CCC form, the Controller shall send a data byte of 0xAA (i.e., the sub-command value is reused for the data byte value, to facilitate error detection). **Reason:** To improve clarity. The revised language clarifies the use of the ENDXFER CCC to apply early transfer termination set-up: either Broadcast or Direct forms of ENDXFER CCC with Defining Byte 0xAA can be used to apply the set-up, there is no need to limit to just the Direct form. **Technical Impact:** Medium

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

Item	Spec Page Number	PDF Page Number	Correction
20	278	314	**Editorial or Technical:** Editorial **Location:** ***Section 6.2.3.4.5***, Line 5100–5101 **Correction:** **Change:** *…whereas the Controller's decision to end the Read Transfer is effectively a Preamble Value of 2'b10 followed by an HDR Restart Pattern or HDR Exit Pattern.* **To:** *…whereas the Controller's decision to end the Read Transfer is effectively a Preamble Value of 2'b10 that tells the Target to stop sending Data Words. Subsequently, the Controller will conditionally start the CRC Word if it previously enabled this with the ENDXFER CCC, and the Target will provide the CRC-5 checksum per* ***Section 6.2.3.4.5.2****; alternately, the Controller will simply send the HDR Restart Pattern or HDR Exit Pattern per* ***Section 6.2.3.4.5.1****.* **Reason:** To improve clarity. The revised language clarifies that the CRC Word is conditionally sent based on ENDXFER configuration. The uncorrected text is based on the I3C v1.0 definition (i.e., not in I3C Basic v1.0) which does not include the possibility that the CRC Word can be sent after Read Transfer termination if previously configured by the Controller. See the equivalent note in ***Section 6.2.3.4.6***, lines 5221–5223. **Technical Impact:** Medium
21	278	314	**Editorial or Technical:** Editorial **Location:** ***Section 6.2.3.4.5***, Line 5102 **Correction:** **Change:** See ***Figure 115***, ***Figure 116***, and ***Figure 117***. **To:** See ***Figure 119***, ***Figure 120***, and ***Figure 121***. **Reason:** To correct editing errors during v1.2 drafting by fixing three incorrect cross-references to indicate the intended figures. **Technical Impact:** None
22	284	320	**Editorial or Technical:** Editorial **Location:** ***Section 6.2.3.4.6***, Line 5220 **Correction:** **Change:** *The procedures for ending the Read Transfer without the CRC Word…* **To:** *The procedures for ending the Write Transfer without the CRC Word …* **Reason:** To correct an editing error: this section defines procedures for Write Transfers, not Read Transfers. **Technical Impact:** Low

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

23	297	333	**Editorial or Technical:** Editorial **Location:** ***Section 6.3***, ***Table 85*** *title* **Correction:** **Change:** **Optional Feature 007 Status** **To:** **Optional Feature 002 Status** **Reason:** To correct an editing error. **Technical Impact:** None
24	299	335	**Editorial or Technical:** Editorial **Location:** ***Section 6.4***, ***Table 86,*** row "Optional Feature specification was last updated in I3C Spec Version" **Correction:** **Change:** I3C Basic v1.2 **To:** I3C v1.2 **Reason:** To correct an editing error. **Technical Impact:** None
25	305	341	**Editorial or Technical:** Technical **Location:** ***Section 6.4.3.2***, Lines 5545–5546 and ***Table 91*** **Correction:** **Insert at the end of Line 5546:** Additionally, Bit[5] and Bit[4] are swapped in the **Transition_Control** byte within the HDR-BT Data Block. **In Table 91:** Insert "(See Note 3)" below "Bit4" and "Bit5". **After Table 91:** Insert additional table note: *3. In order to prevent parity errors, Bit4 and Bit5 are swapped in the* ***Transition_Control*** *byte within the HDR-BT Data Block. Bit4 will be sent on SDA[1] instead of SDA[0], and Bit5 will be sent on SDA[0] instead of SDA[1]. These bits are not swapped for any other uses of* ***Transition****-type bytes.* **Reason:** To correct a technical error for HDR-BT Dual Lane, for situations where the I3C Controller terminates a Read transfer. Leaving these bits un-swapped could result in parity errors if the I3C Controller terminates the Read transfer when the I3C Target is attempting to use the Data Block Delay mechanism. **Technical Impact:** High
26	358	398	**Editorial or Technical:** Editorial **Location:** ***Section 6.4.3.3.3***, after Line 5813 **Correction:** Insert sub-bullet: • The transmitter shall use the natural bit ordering when it includes Transition bytes in the checksum, i.e., it shall **not** use the bit-swizzled value when calculating the CRC in either Dual Lane or Quad Lane (see ***Section 6.4.3.2***). **Reason:** To improve clarity. Need to state that the Transition bytes are treated as Data bytes for purposes of CRC checksum calculation. **Technical Impact:** Low

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

Item	Spec Page Number	PDF Page Number	Correction
27	333	369	**Editorial or Technical:** Editorial **Location:** ***Section 6.4.3.6***, ***Figure 137*** **Correction:** Replace Figure 137 with the figure below. **Reason:** To correct an authoring error. The uncorrected Figure 137 shows the wrong handoff point for an HDR-BT Read transfer for Single Lane, where the Target starts driving SCL (if configured to do so). The handoff point should be before the falling edge of C25 (i.e., the first clock cycle of the Transition byte), not after the falling edge. Although normative text correctly defines the intended handoff behavior, the uncorrected Figure 137 could potentially mislead implementers. **Technical Impact:** Medium.

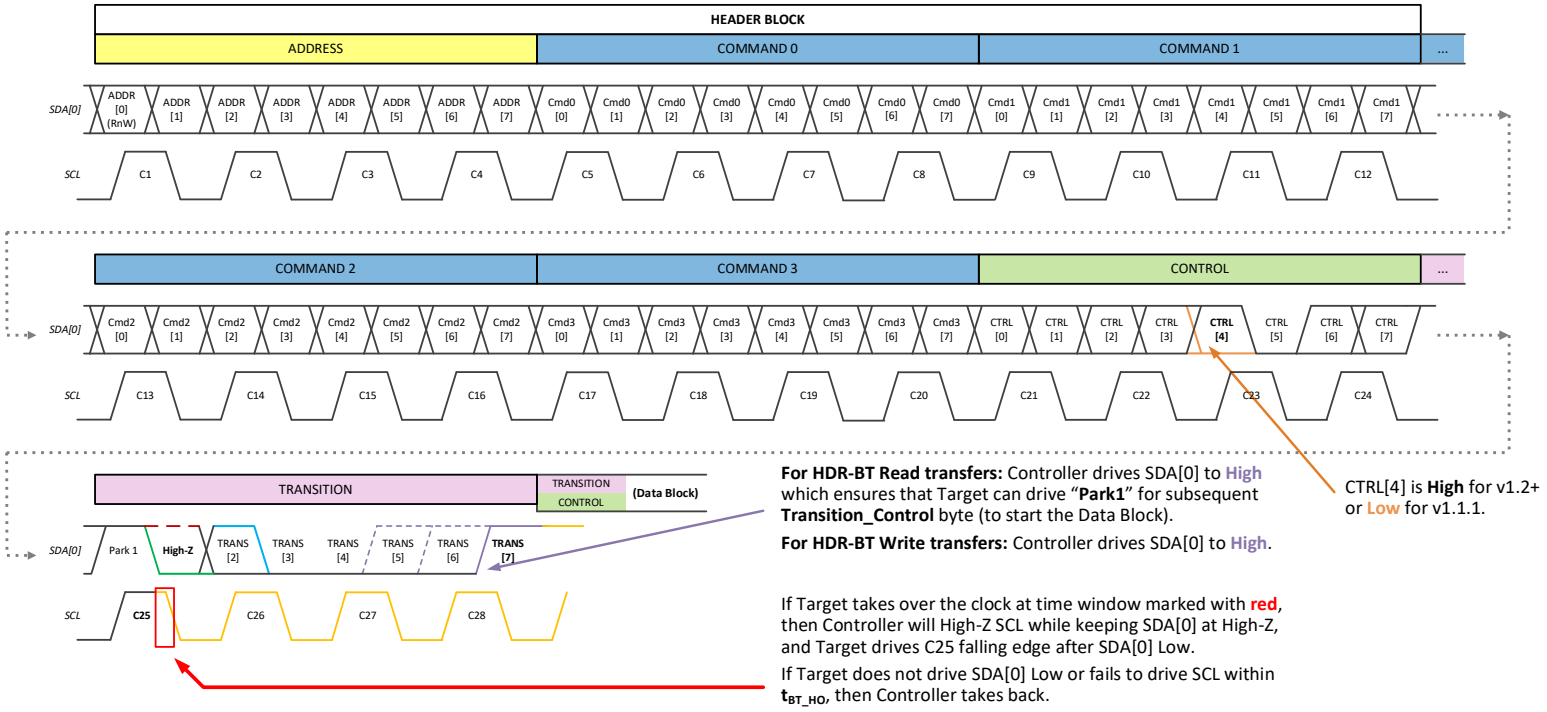


Figure 137 Header Block for Single Lane

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

Item	Spec Page Number	PDF Page Number	Correction
28	334	370	**Editorial or Technical:** Editorial **Location:** ***Section 6.4.3.6***, ***Figure 138*** **Correction:** Replace Figure 138 with the figure below. **Justification:** Authoring error. **Reason:** To correct an authoring error. The uncorrected Figure 138 shows the wrong handoff point for an HDR-BT Read transfer for Dual Lane, where the Target starts driving SCL (if configured to do so). The handoff point should be before the falling edge of C13 (i.e., the first clock cycle of the Transition byte), not after the falling edge. Although normative text correctly defines the intended handoff behavior, the uncorrected Figure 138 could potentially mislead implementers. **Technical Impact:** Medium.

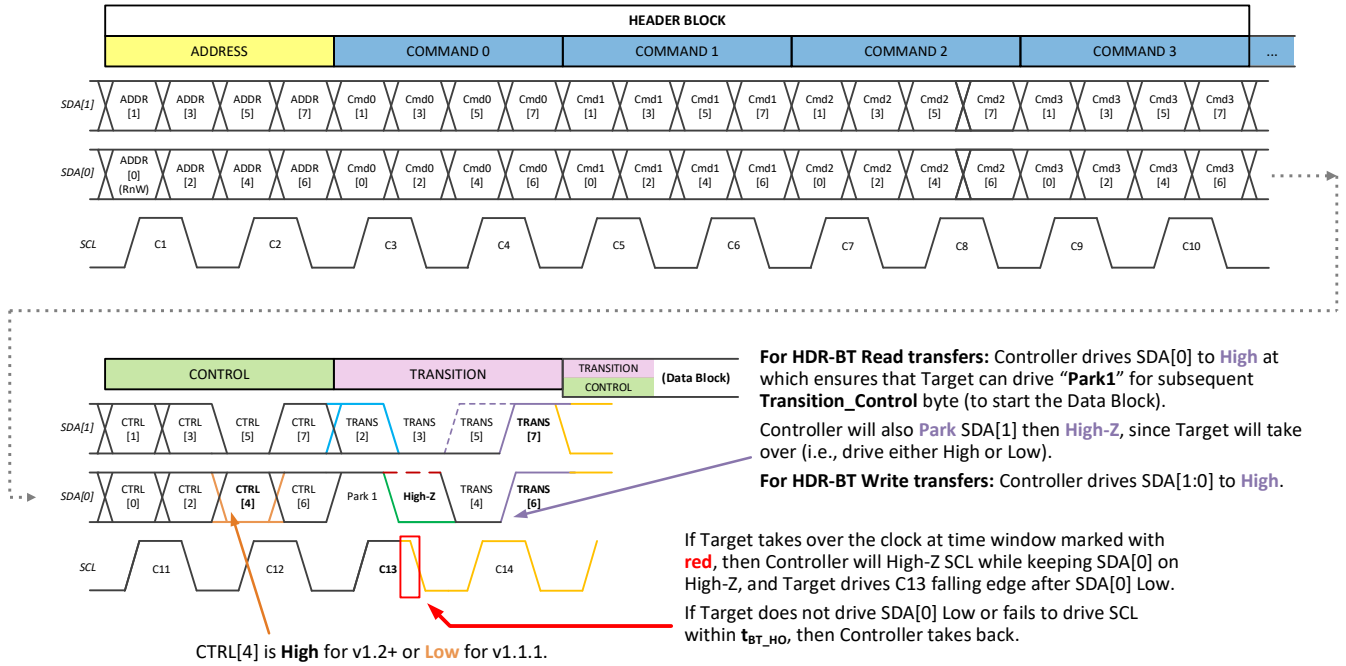


Figure 138 Header Block for Dual Lane

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

Item	Spec Page Number	PDF Page Number	Correction
29	335	371	**Editorial or Technical:** Editorial **Location:** ***Section 6.4.3.6***, ***Figure 139*** **Correction:** Replace Figure 139 with the figure below. **Reason:** To correct an authoring error. The uncorrected Figure 139 shows the wrong handoff point for an HDR-BT Read transfer for Quad Lane, where the Target starts driving SCL (if configured to do so). The handoff point should be before the falling edge of C7 (i.e., the first clock cycle of the Transition byte), not after the falling edge. Although normative text correctly defines the intended handoff behavior, the uncorrected Figure 139 could potentially mislead implementers. **Technical Impact:** Medium.

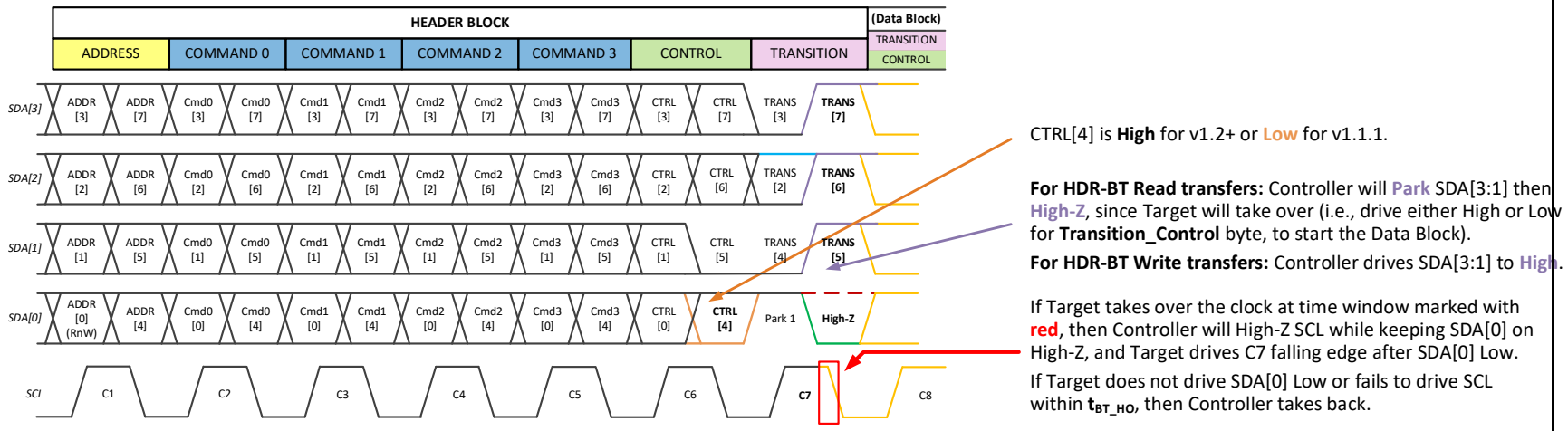
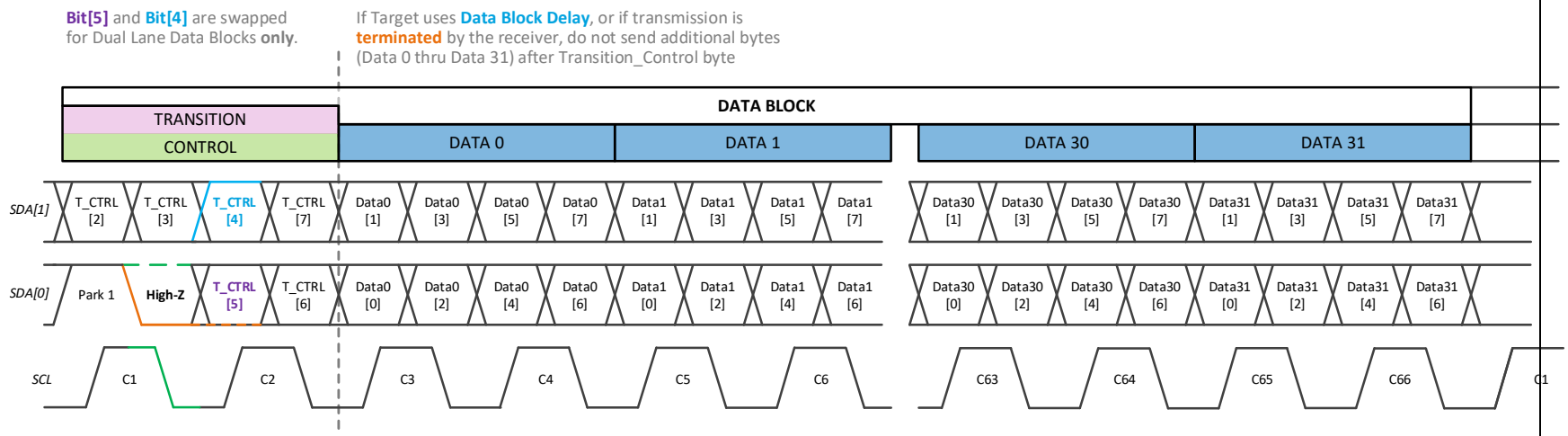


Figure 139 Header Block for Quad Lane

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

Item	Spec Page Number	PDF Page Number	Correction
30	337	373	**Editorial or Technical:** Editorial **Location:** ***Section 6.4.3.6***, ***Figure 141*** **Correction:** Replace Figure 141 with the figure below. **Reason:** To correct a technical error. The uncorrected Figure 141 does not show the updated HDR-BT Dual Lane bit packing in the **Transition_Control** byte. See Table 91 as corrected by Errata 01 Item 25. **Technical Impact:** Medium.



In Transition_Control byte, transmitter High-Zs SDA[0] for **second half-clock** of C1. If receiver drives SDA[0] to **Low** (**shown in orange**), this terminates the transmission after the Transition_Control byte. Transmitter then emits a CRC Block **as the next byte** (after Transition_Control). Transmitter **must** then keep SDA[0] at **Low** for T_CTRL[4].

If Data Block Delay is permitted: Target as transmitter may drive Transition_Control Bit[4] **High** (**shown in blue**) to indicate delayed data. Target may repeat for maximum of t_{BT_DBD} Delay bytes. (Does not apply for Last Data Block before CRC Block.)

Termination overrides **Data Block Delay**

Figure 141 Data Block for Dual Lane

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

Item	Spec Page Number	PDF Page Number	Correction
31	357	393	**Editorial or Technical:** Editorial **Location:** ***Section 6.6***, ***Table 96***, row “Optional Feature ID” **Correction:** **Change:** F002 **To:** F005 **Reason:** To correct an editing error. **Technical Impact:** None
32	357	393	**Editorial or Technical:** Editorial **Location:** ***Section 6.6***, Line 6799 **Correction:** **Change:** “Asynchrnous” **To:** “Asynchronous” **Reason:** To correct a typo. **Technical Impact:** None
33	375	411	**Editorial or Technical:** Editorial **Location:** ***Section 6.7.3.1***, after Line 7122 **Correction:** Insert: ***Note:*** *Each ML-capable Target Device will respond to the Direct GET MLANE CCC by sending data on SDA[0] as well as all Additional Data Lanes that it supports. The Controller should monitor all Additional Data Lanes during the Direct GET MLANE CCC (per* ***Section 6.7.3.6****) to ensure that each Target Device has the expected capabilities and connectivity.* **Reason:** To improve clarity. This is how the Controller can test whether an ML-capable Target is connected correctly before using the MLANE CCC to configure the Target for ML transfers. For example, if an ML-capable Target reported that it is capable of ML transfers in QUAD LANE configurations (i.e., in its response to the MLANE Direct GET CCC with Defining Byte 0xFF), but the Target did **not** drive the correct number of Additional Data Lanes in its response to other Defining Bytes for specific I3C Modes (i.e., no data sent on SDA[3:1]), then the Controller will know that this Target is **not** connected properly and it should **not** configure this Target to use ML transfers with QUAD LANE. If an ML-capable Target is not connected to the expected Additional Data Lanes, then subsequent ML transfers would likely fail. **Technical Impact:** Low

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

Item	Spec Page Number	PDF Page Number	Correction
34	376	412	**Editorial or Technical:** Editorial **Location:** ***Section 6.7.3.1***, Lines 7143–7144 **Correction:** **Change:** …(per ***Figure 97***)… **To:** …(per ***Figure 161***)… **Reason:** To correct an authoring error: the wrong figure was cross-referenced. The intention was to cross-reference Figure 161 which shows the use of the MLANE Direct GET CCC to verify connectivity for additional data lanes. Figure 97 is an unrelated figure in a different section. **Technical Impact:** Medium
35	403	439	**Editorial or Technical:** Technical **Location:** ***Section 6.7.3.6***, Line 7896, after "…0x23 only" but before the period in that sentence and "***Figure 161*** shows…" **Correction:** Insert: and is required even if the Target has **not** yet been configured to use ML transfers in any supported I3C Modes. The Target shall send the same DATA/T values shall be sent on all Additional Lanes that it supports. **Reason:** To clarify that the Target must drive data on all Additional Data Lanes, even before it is configured to use ML transfers. This is a necessary step for the Controller to test the Target's connectivity before configuring the Target to use ML transfers. **Technical Impact:** Low

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

<table>
<tr><th>Item</th><th>Spec Page Number</th><th>PDF Page Number</th><th>Correction</th></tr>
<tr><td>36</td><td>407</td><td>421</td><td>Editorial or Technical: Technical
Location: Section 6.7.3.6, Table 108, column “Description”, row “0x9C”
Correction:
After the end of the second paragraph (“…or NACK its Target Address.”), insert:
If the Target does support separate Group ML configurations but none are currently configured, then it may return either a 1-byte message of 0x0; or a 2-byte message of the number of possible configurations followed by 0x0.
Also replace the entire Example paragraph and subsequent lines in this cell with the text below.
Reason: To clarify expectations for ML-capable Targets that support Group ML configurations. Additionally, the provided example refers to HDR-DDR-ML which is not included in the I3C Basic Specification.
Technical Impact: Low</td></tr>
<tr><td colspan="4">Example: For a Target that supports up to 4 Group Addresses, supports ML only in HDR-BT Mode, and that is currently assigned to 3 Groups, the Target returns the byte 0x04 to indicate 4 separate Group ML configurations that are possible, followed by a 3-byte tuple for each configuration entry that is valid (i.e., a Group that is currently configured for ML transfers). Any invalid (i.e., non-configured) entries shall be returned after all valid entries, and shall be represented by a 3-byte tuple of all ZEROs. For valid entries, the first byte of each tuple is the Group Address, the second byte is the HDR-BT value (0x23), and the third byte is the ML Frame format’s data byte. This Target is assigned to Group Addresses 7'h71, 7'h72, and 7'h74.

The addressed Target therefore transfers the following thirteen bytes:
0x04, The number of Group ML configurations across all I3C Modes
0x71, 0x23, 0x00, Group Address 7’h71, HDR-BT mode 0x00, 1-Lane default
0x72, 0x23, 0x01, Group Address 7’h72, HDR-BT mode 0x01, 2-Lane ML with Coding 0
0x74, 0x23, 0x1B, Group Address 7’h74, HDR-BT mode 0x1B, 4-Lane ML with Coding 3
0x00, 0x00, 0x00 Not configured</td></tr>
<tr><td>37</td><td>407</td><td>443</td><td>Editorial or Technical: Technical
Location: Section 6.7.3.6, Table 108, column “Additional Data Bytes”, row “0x9C”, after last bullet
Correction: Insert:
• Controller may terminate the transfer after receiving a Group ML configuration tuple (i.e., after the first byte) that starts with 0x0 (i.e., the first invalid or non-configured entry).
Reason: To clarify expectations for Controllers that query Targets for Group ML configurations.
Technical Impact: Low</td></tr>
<tr><td>38</td><td>413</td><td>449</td><td>Editorial or Technical: Editorial
Location: Section 6.9, Table 110, row “Optional Feature ID”
Correction: In the first row (Optional Feature ID), replace “F005” with “F008” to match the feature ID defined for this section.
Change: F005
To: F008
Reason: To correct an editing error.
Technical Impact: None</td></tr>
</table>

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

Item	Spec Page Number	PDF Page Number	Correction
39	450	486	**Editorial or Technical:** Technical **Location:** ***Annex E***, *Figure 188* **Correction:** Replace Figure 188 with the figure below. **Reason:** To correct an authoring error. It is necessary to remove the flow steps for Extended Address (i.e., 10-bit) transfers from the FSM that describes Legacy I²C Controller behavior. There was a technical error in the existing figure relating to transfers to an Extended Address. Additionally, since I3C Controllers do not support Extended Address mode and would not use these flow steps, the I3C WG decided to simplify this FSM. **Technical Impact:** Low

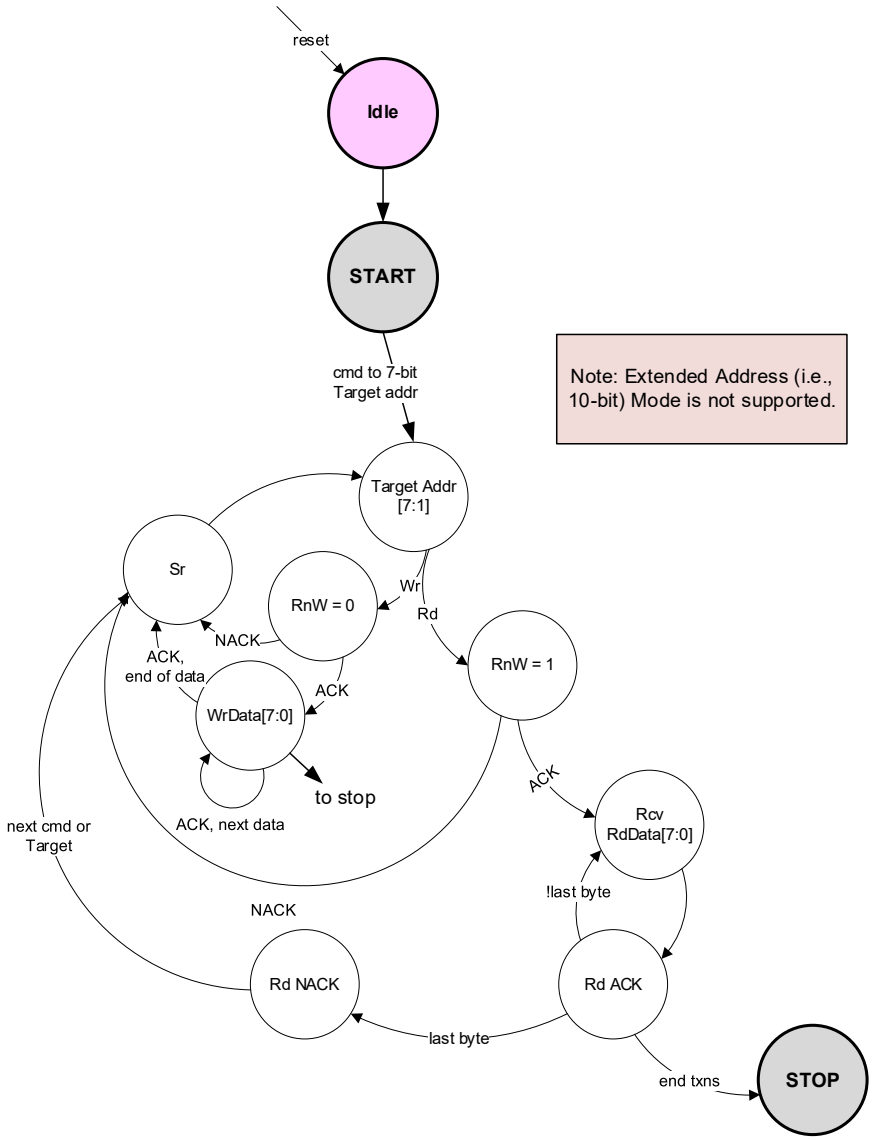


Figure 278 I²C Legacy Controller FSM

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2026 MIPI Alliance, Inc.
Public Release Edition

Specification for I3C Basic™

Improved Inter Integrated Circuit

Version 1.2

16 December 2024

MIPI Board Adopted Date: 17 April 2025

Public Release Edition

CHANGED IN THIS VERSION

	New Features	Technical Updates	Editorial Updates
Required Elements	None	None	None
Optional Features	None	None	None

This document is subject to further editorial and technical development as work continues in the MIPI Alliance I3C Basic Ad Hoc Working Group.

Copyright © 2016–2025 MIPI Alliance, Inc.

NOTICE OF DISCLAIMER

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®. The material contained herein is provided on an "AS IS" basis and 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 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.

All materials contained herein are 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, tradenames, and other intellectual property are the exclusive property of MIPI Alliance and cannot be used without its express prior written permission.

ALSO, THERE IS NO WARRANTY OF CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THIS MATERIAL OR THE CONTENTS OF THIS DOCUMENT. IN NO EVENT WILL ANY AUTHOR OR DEVELOPER OF THIS MATERIAL OR THE CONTENTS OF THIS DOCUMENT 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, SPECIFICATION OR DOCUMENT RELATING TO THIS MATERIAL, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Without limiting the generality of this Disclaimer stated above, the user of the contents of this Document is further notified that MIPI: (a) does not evaluate, test or verify the accuracy, soundness or credibility of the contents of this Document; (b) does not monitor or enforce compliance with the contents of this Document; and (c) does not certify, test, or in any manner investigate products or services or any claims of compliance with the contents of this Document. The use or implementation of the contents of this Document 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 Document or otherwise.

Questions pertaining to this document, or the terms or conditions of its provision, should be addressed to: secretary@mipi.org.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Contents

Figures ..... **vi**

Tables ..... **xi**

Release History ..... **xiv**

Preface to the I3C Basic Specification **1**

1 What is MIPI I3C Basic? 1
2 Motivation 1
3 IPR Status 1
4 Relationship to MIPI I3C v1.x Specifications 2
4.1 I3C v1.1.1 Functions Not Included in I3C Basic 2
4.2 Organization of This Specification 2
4.3 I3C Basic v1.2 as an Update of I3C Basic v1.1.1 2
5 How I3C Basic Devices Work with I3C v1.x Devices 2

1 Structure of This Specification (Informative) **3**

1.1 Managing Optional Element Combinations 4

2 Introduction **5**

2.1 Scope and Purpose (Informative) 6
2.2 I3C Key Features (Informative) 6
2.3 Terminology 10
2.3.1 Use of Special Terms 10
2.3.2 Definitions 10
2.3.3 Abbreviations 15
2.3.4 Acronyms 15
2.4 References 17
2.4.1 Informative References 17
2.4.2 Normative References 17

3 Technical Overview (Informative) **19**

3.1 I3C Fundamental Principles 20
3.2 I3C Controller and Target Devices 23
3.2.1 I3C Controller Device 24
3.2.2 I3C Target Device 26

4 Required Elements **31**

4.1 Scope and Purpose (Informative) 31
4.2 Technical Overview (Informative) 31
4.3 Technical Specification 33
4.3.1 Bus Configuration 33
4.3.2 Bus Communication 41
4.3.3 Bus Conditions 59
4.3.4 Bus Initialization and Dynamic Address Assignment Mode 62
4.3.5 Hot-Join Mechanism 70
4.3.6 In-Band Interrupt 75
4.3.7 Common Command Codes (CCC) 82
4.3.8 Error Detection and Recovery Methods for SDR 150
4.3.9 Target Reset 160
4.3.10 HDR Pattern Tolerance 167
4.3.11 Electrical Specifications 171

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

5 Assigned Values 197
5.1 DCR Byte (Device Configuration Register) 198
5.2 SETBUSCON Context Byte 199
5.3 Defining Byte for Core CCCs 200
5.4 GETCAPS Capability Bits 203
5.4.1 GETCAPS Format 2 206
5.4.2 Read Fixed Test Pattern (GETCAPS Format 2, Defining Byte TESTPAT 0x5A) 208
5.4.3 Get Controller Feature Capabilities (GETCAPS Format 2, Defining Byte CRCAPS 0x91) 209
5.4.4 Get Debug Feature Capabilities (GETCAPS Format 2, Defining Byte 0xD7 DBGCAPS) 211
5.4.5 Get Virtual Target Capabilities (GETCAPS Format 2, Defining Byte 0x93 VTCAPS) 212
5.5 In-Band Interrupt MDB (Mandatory Data Byte) 215

6 Optional Features 219
6.1 HDR Modes 221
6.1.1 Technical Overview (Informative) 221
6.1.2 HDR Restart Pattern 222
6.1.3 CCC Framing in HDR Modes 225
6.1.4 Transition to HDR Mode Transfer 240
6.2 F001: HDR-DDR Mode 243
6.2.1 Scope and Purpose (Informative) 243
6.2.2 Technical Overview (Informative) 243
6.2.3 Technical Specification 245
6.3 F002: HDR Ternary Modes 297
6.4 F003: HDR-BT Mode 299
6.4.1 Scope and Purpose (Informative) 299
6.4.2 Technical Overview (Informative) 299
6.4.3 Technical Specification 301
6.5 F004: Secondary Controller 345
6.5.1 Scope and Purpose (Informative) 345
6.5.2 Technical Overview (Informative) 345
6.5.3 Technical Specification 346
6.6 F005: Timing Control 357
6.6.1 Scope and Purpose (Informative) 357
6.6.2 Technical Overview (Informative) 357
6.6.3 Technical Specification 359
6.7 F006: Multi-Lane 373
6.7.1 Scope and Purpose (Informative) 373
6.7.2 Technical Overview (Informative) 373
6.7.3 Technical Specification 375
6.8 F007: Monitoring Device 411
6.9 F008: D2D Tunneling 413
6.10 F009: Controller Clock Stalling 415
6.10.1 Scope and Purpose (Informative) 415
6.10.2 Technical Overview (Informative) 415
6.10.3 Technical Specification 416
6.11 F010: Alternative Electrical Specifications 421
6.11.1 Scope and Purpose (Informative) 421
6.11.2 Technical Specification 422

Annex A Changes to Required Elements 427
Changes to Required Elements in I3C Basic Version 1.2 427

Annex B I3C Implementation Summary Templates 429
B.1 Full Format 429
B.2 Concise Format 430

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Annex C I3C Basic Communication Format Details 431
C.1 I3C Basic CCC Transfers 431
C.2 I3C Basic Private Write and Read Transfers 432
C.3 Legacy I^2C Write and Read Transfers on the I3C Basic Bus 434
C.4 Dynamic Address and Enter HDR 436
Annex D SDR Mode Error Type Origins 439
D.1 Error Types in I3C Basic CCC Transfers 439
D.2 Error Types in I3C Basic Private Read and Write Transfers 440
D.3 Error Types in Dynamic Address Arbitration 442
Annex E I3C Basic Controller FSMs 443
Annex F Typical I3C Basic Protocol Communications 451
F.1 Typical SDR Private Read 451
F.2 Typical Direct CCC in SDR Mode 452
F.3 Typical Broadcast CCC in SDR Mode 453
F.4 Typical HDR-DDR Read 454
F.5 Typical HDR-BT Read 455
Annex G MIPI I3C Basic Specification Supplemental Patent Licensing Terms 457
Attachment A 459
Annex H I3C Basic Development Companies 461
Participants 463

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Figures

Figure 1 I3C System Diagram 5
Figure 2 I3C vs. I²C FM+ Data Blocks Bit Rates in Mbps (12.5 MHz Clock) 7
Figure 3 I3C vs. I²C FM+ 1 kB Effective Data Transfer Times in ms 8
Figure 4 I3C vs. I²C FM+ Energy per 1 kB Effective Data Block, in µJ 8
Figure 5 I3C Multi-Lane Effective Bit Rates, in Mbps 9
Figure 6 Example of Data Traffic on the I3C Bus 20
Figure 7 I3C Communication Flow 21
Figure 8 I3C Controller Device Block Diagram 24
Figure 9 I3C Target Device Block Diagram 26
Figure 10 I3C Basic Bus with I²C Devices and I3C Basic Devices 33
Figure 11 I3C Basic Minimal Bus with I3C Basic Target Devices 34
Figure 12 Address Header Comparison 42
Figure 13 Address Arbitration During Header 48
Figure 14 Transition from Address ACK to Mandatory Byte During IBI 54
Figure 15 Transition from IBI Address NACK to Repeated START or STOP 55
Figure 16 Bus Condition Timing 60
Figure 17 Dynamic Address Assignment Transaction 66
Figure 18 IBI Sequence with Mandatory Data Byte 76
Figure 19 CCC Broadcast General Frame Format 84
Figure 20 CCC Direct General Frame Format 84
Figure 21 ENEC/DISEC Format 1: Direct 97
Figure 22 ENEC/DISEC Format 2: Broadcast 97
Figure 23 ENTASx Format 1: Direct 99
Figure 24 ENTASx Format 2: Broadcast 99
Figure 25 RSTDAA Format 101
Figure 26 ENTDAA Format 101
Figure 27 Direct SETMWL/GETMWL Format 102
Figure 28 Broadcast SETMWL Format 102
Figure 29 Direct SETMRL/GETMRL Format 103
Figure 30 Broadcast SETMRL Format 103
Figure 31 DEFTGTS Format 104
Figure 32 ENTTM Format 106
Figure 33 SETDASA Format 1: Primary 108
Figure 34 SETDASA Format 2: Point-to-Point 109
Figure 35 SETNEWDA Format 110
Figure 36 GETPID Format 111
Figure 37 GETBCR Format 111
Figure 38 GETDCR Format 112
Figure 39 GETSTATUS Format 1 113

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Figure 40 GETSTATUS Format 2 115
Figure 41 GETSTATUS Format 2 with Defining Byte PRECR 118
Figure 42 GETACCCR Format 1: Accepted 119
Figure 43 GETACCCR Format 2: Not Accepted 120
Figure 44 GETACCCR Format 3: Incorrect Cancel 120
Figure 45 SETBRGTGT Format 121
Figure 46 GETMXDS Format 1: Without Turnaround 123
Figure 47 GETMXDS Format 2: With Turnaround 123
Figure 48 GETMXDS Format 3: With Defining Byte 126
Figure 49 GETMXDS Format 3 With Defining Byte CRHDLY 128
Figure 50 GETCAPS CCC Format Overview 130
Figure 51 SETROUTE Format 131
Figure 52 Example Routing Device 132
Figure 53 SETAASA Format 133
Figure 54 ENDXFER Format 1: Broadcast 134
Figure 55 ENDXFER Format 2: Direct 134
Figure 56 RSTACT Format 1: Broadcast 136
Figure 57 RSTACT Format 2: Direct Write 137
Figure 58 RSTACT Format 3: Direct Read 137
Figure 59 SETGRPA Format 141
Figure 60 RSTGRPA Format 1: Direct 142
Figure 61 Resetting a Group Address with the RSTGRPA Direct CCC 143
Figure 62 RSTGRPA Format 2: Broadcast 143
Figure 63 DEFGRPA Format 144
Figure 64 SETBUSCON Format 145
Figure 65 Example Waveform for Error Type TE0 151
Figure 66 Target Reset Operations Flow 162
Figure 67 Target Reset Pattern: General Case 163
Figure 68 Target Reset Pattern: From Idle 163
Figure 69 HDR Exit Pattern Followed by STOP 168
Figure 70 HDR Exit Pattern After IDLE Followed by STOP 168
Figure 71 Example HDR Exit Pattern Detector (Schematic) 169
Figure 72 Metastable Changes on SCL and SDA Do Not Break the Exit Pattern Detector 169
Figure 73 Components of Clock-to-Data Turnaround Delay (t_{SCO}) 180
Figure 74 I3C Basic Legacy Mode Timing 181
Figure 75 t_{DIG_H} and t_{DIG_L} 181
Figure 76 I3C Basic Write Address ACK 182
Figure 77 I3C Basic Write Address NACK 183
Figure 78 I3C Basic START Timing 184
Figure 79 I3C Basic STOP Timing 184
Figure 80 I3C Basic Controller Out Timing 185
Figure 81 I3C Basic SDR Target Out Timing 185

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Figure 82 Controller SDR Timing 186
Figure 83 Controller DDR Timing 186
Figure 84 T-Bit When Target Ends Read and Controller Generates STOP 187
Figure 85 T-Bit When Target Ends Read and Controller Generates Repeated START 188
Figure 86 T-Bit When Target and Controller Agree to Continue Read Message 189
Figure 87 T-Bit When Controller Ends Read with Repeated START and STOP 190
Figure 88 T-Bit When Controller Ends Read via Repeated START and Further Transfer 191
Figure 89 Controller to Controller Bus Handoff 192
Figure 90 I^2C Spike Filter Behavior 193
Figure 91 GETCAPS Format 1 203
Figure 92 GETCAPS Format 2 206
Figure 93 GETCAPS Format 2 with Defining Byte TESTPAT 208
Figure 94 GETCAPS Format 2 with Defining Byte CRCAPS 209
Figure 95 GETCAPS Format 2 with Defining Byte DBGCAPS 211
Figure 96 GETCAPS Format 2 with Defining Byte VTCAPS 212
Figure 97 HDR Restart with Next Edge 222
Figure 98 Combined HDR Restart and Exit Pattern Detector (Schematic) 223
Figure 99 Begin Broadcast CCC Per HDR Mode 230
Figure 100 Begin Direct CCC Per HDR Mode 231
Figure 101 End of CCC Procedures Per HDR Mode 233
Figure 102 Entering HDR Mode After ENTHDRx CCC (Dedicated Edge Clock) 241
See Errata 01, Item 1
Figure 104 Next HDR Mode Transfer After HDR Restart Pattern (Dedicated Edge Clock) 242
Figure 106 Typical HDR-DDR Mode Frame 244
Figure 107 HDR-DDR Word Formats 246
Figure 108 HDR-DDR Preamble Bits State Diagram 248
Figure 109 HDR-DDR CRC and HDR Restart Pattern or HDR Exit Pattern in Write Transaction 251
Figure 110 HDR-DDR CRC and HDR Restart Pattern or HDR Exit Pattern in Read Transaction 253
Figure 111 HDR-DDR Command Code After ENTHDR0 255
Figure 112 HDR-DDR Command Code After HDR Restart Pattern 255
Figure 113 HDR-DDR Broadcast CCC Format 257
Figure 114 HDR-DDR Direct CCC Format 258
Figure 115 HDR-DDR CCC End Procedures 265
Figure 116 Target Accepts DDR WRITE Command from Controller 268
Figure 117 Target Does Not Respond to DDR WRITE Command from Controller 270
Figure 118 Target Accepts DDR READ Command from Controller 272
Figure 119 Target Does Not Respond to DDR READ Command from Controller 274
Figure 120 Controller HDR-DDR ENDXFER Steps Flow 276
Figure 121 Controller Ends DDR Read and Provides Either HDR Restart Pattern or HDR Exit Pattern 279
Figure 122 Controller Ends DDR Read and Target Provides CRC 281
Figure 123 Controller Continues DDR Read from Target 283
Figure 124 Target Requests DDR Write Termination and Controller Provides HDR Restart Pattern or HDR Exit Pattern 285

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

See Errata 01, Item 1

Figure 125 Target Requests DDR Write Termination and Controller Provides CRC 287
Figure 126 Target Continues the DDR Write From Controller 289
Figure 127 SCL Stalling After PRE0 Bit on Write ACK 290
Figure 128 SCL Stalling After PRE0 Bit on Write Abort 291
Figure 129 SCL Stalling in PRE1 Bit of DDR Write 292
Figure 130 SCL Stalling on PRE1 Bit of DDR Read Transfer 293
Figure 131 SCL Stalling on HDR Restart Pattern / HDR Exit Pattern After CRC 294
Figure 132 SCL Stalling After Write/Read Abort 294
Figure 133 HDR-BT Header Block After ENTHDR3 301
Figure 134 HDR-BT Header Block After HDR Restart Pattern 302
Figure 135 Typical HDR-BT Mode Frame 303
Figure 136 Begin Broadcast CCC in HDR-BT 320
Figure 137 Begin Direct CCC in HDR-BT 321
Figure 138 HDR-BT CCC End Procedure Options 331
Figure 139 Header Block for Single Lane 333
Figure 140 Header Block for Dual Lane 334
Figure 141 Header Block for Quad Lane 335
Figure 142 Data Block for Single Lane 336
Figure 143 Data Block for Dual Lane 337
Figure 144 Data Block for Quad Lane 338
Figure 145 CRC Block for Single Lane 339
Figure 146 CRC Block for Dual Lane 339
Figure 147 CRC Block for Quad Lane 340
Figure 148 HDR-BT Flow Control for Write Transfers 342
Figure 149 HDR-BT Flow Control for Read Transfers 343
Figure 150 Pre-Handoff Steps Flow 347
Figure 151 Graphical Representation of Async Mode 0 Timestamp Interpolation 361
Figure 152 Example of Asynchronous Mode 0 Timestamp Data Transfer 365
Figure 153 SETXTIME Format 1: Broadcast 368
Figure 154 SETXTIME Format 2: Direct 368
Figure 155 GETXTIME Format 370
Figure 156 Typical Generic ML I3C Frame (HDR) 384
Figure 157 Typical ML I3C Frame Based on HDR-BT Mode (Coding 0) 391
Figure 158 Typical ML I3C Frame Based on HDR-BT Mode (Coding 3) 392
Figure 159 Typical ML I3C Frame Based on HDR-BT Mode (Coding 7) 393
Figure 160 Valid State Transitions for HDR-BT Data Transfer Codings 397
Figure 161 MLANE Format 1: Broadcast 403
Figure 162 MLANE Format 2: Direct 403
Figure 163 MLANE Direct GET Version of SET/GET CCC (SDR Mode Only) 404
Figure 164 Controller Clock Stalling in ACK Phase 417
Figure 165 Controller Clock Stalling in Write Parity Bit 417
Figure 166 Controller Clock Stalling in T-Bit Before Next Read Data 418

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

See Errata 01, Item 1

Figure 167 Controller Clock Stalling in T-Bit Before STOP 418
Figure 168 Controller Clock Stalling in Low T-Bit Before Repeated START 419
Figure 169 Controller Clock Stalling in High T-Bit Before Repeated START 419
Figure 170 Controller Clock Stalling in High T-Bit Before Repeated START and STOP 420
Figure 171 Controller Clock Stalling in Dynamic Address First Bit 420
Figure 172 I3C Basic CCC Transfers 431
Figure 173 I3C Basic Private Write and Read Transfers with 7'h7E Address 432
Figure 174 I3C Basic Private Write and Read Transfers without 7'h7E Address 433
Figure 175 Legacy I^{2}C Write and Read Transfers in I3C Basic Bus with 7'h7E Address 434
Figure 176 Legacy I^{2}C Write and Read Transfers in I3C Basic Bus without 7'h7E Address 435
Figure 177 Dynamic Address Allocation ENTDAA CCC Bus Modal 436
Figure 178 Enter HDR Mode CCC Bus Modal 437
Figure 179 Error Type Origins: I3C Basic CCC Transfers 439
Figure 180 Error Type Origins: I3C Basic Private Write & Read Transfers with 7'h7E Address 440
Figure 181 Error Type Origins: I3C Basic Private Write & Read Transfers without 7'h7E Address 441
Figure 182 Error Type Origins: Dynamic Address Arbitration 442
Figure 183 I3C Basic Primary Controller FSM 443
Figure 184 In-Band Interrupt (Target Interrupt) Request FSM 444
Figure 185 Dynamic Address Assignment FSM 445
Figure 186 Hot-Join FSM 446
Figure 187 Controller Role Request FSM 447
Figure 188 Controller Regaining Bus Ownership FSM 448
Figure 189 HDR-DDR Mode FSM 449
Figure 190 I^{2}C Legacy Controller FSM 450
Figure 191 Example Communication Using I3C Basic SDR Mode with Private Read 451
Figure 192 Example Communication Using I3C Basic SDR Mode with Direct CCC 452
Figure 193 Example Communication Using I3C Basic SDR Mode with Broadcast CCC 453
Figure 194 Example Communication Using HDR-DDR Protocol 454
Figure 195 Example Communication Using HDR-BT Protocol 455

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Tables

Table 1 Structure of This Specification 3
Table 2 Roles for I3C Compatible Devices 23
Table 3 Required Elements Status 31
Table 4 I3C Basic Devices Roles vs Responsibilities 35
Table 5 I^{2}C Features Allowed in I3C Basic Targets 36
Table 6 Legacy I^{2}C-Only Target Categories and Characteristics 37
Table 7 Bus Characteristics Register (BCR) 39
Table 8 I3C Device Characteristics Register (DCR) 40
Table 9 Legacy I^{2}C Virtual Register (LVR) 40
Table 10 I3C Basic Target Address Restrictions 51
Table 11 Available Options for Bus Operating Parameters, Per I3C Bus Configuration 57
Table 12 Activity States 61
Table 13 Mandatory Data Byte Field Format 77
Table 14 Mandatory Data Byte Value Ranges 78
Table 15 CCC Frame Field Definitions 85
Table 16 I3C Basic Common Command Codes 90
Table 17 Enable/Disable Target Events Command Codes (ENEC/DISEC) 97
Table 18 Enable Target Events Command Byte Format 97
Table 19 Disable Target Events Command Byte Format 97
Table 20 Enter Activity State 0–3 Command Codes (ENTAS0–ENTAS3) 99
Table 21 Enter Activity State CCCs (ENTASx) 100
Table 22 Set/Get Max Write Length Command Codes (SETMWL/GETMWL) 102
Table 23 Set/Get Max Read Length Command Codes (SETMRL/GETMRL) 103
Table 24 DEFTGTS Data Bytes for Target or Group 105
Table 25 ENTTM Test Mode Byte Values 106
Table 26 Enter HDR Mode CCCs (ENTHDRx) 107
Table 27 GETSTATUS MSB-LSB Format 1 114
Table 28 GETSTATUS Format 2 Defining Byte Values 116
Table 29 Status Bytes for GETSTATUS Format 2 with Defining Byte PRECR 118
Table 30 maxWr Byte Format 124
Table 31 maxRd Byte Format 124
Table 32 maxRdTurn Format 125
Table 33 GETMXDS Defining Byte Values 126
Table 34 CRHDLY1 Byte Format 129
Table 35 Data Transfer Ending Procedure Control Command Codes (ENDXFER) 134
Table 36 ENDXFER Defining Byte Values 135
Table 37 Target Reset Action Command Codes (RSTACT) 136
Table 38 RSTACT Defining Byte Values 138
Table 39 RSTACT Format 3: Direct Read CCC Byte Read Data Format 140

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Table 40 Reset Group Address Command Codes (RSTGRPA) 142
Table 41 SETBUSCON Context Values 146
Table 42 Direct CCC Support for Group Addresses 148
Table 43 SDR Target Error Types 150
Table 44 SDR Controller Error Types 155
Table 45 Scope of Target Reset Actions per Reset Type 164
Table 46 I3C Basic I/O Stage Characteristics Common to Push-Pull Mode and Open Drain Mode 172
Table 47 Legacy I^2C Device Requirements When Operating on I3C Basic 174
Table 48 I3C Basic Timing Requirements When Communicating With I^2C Legacy Devices 175
Table 49 I3C Basic Open Drain Timing Parameters 176
Table 50 I3C Basic Push-Pull Timing Parameters for SDR, ML, HDR-DDR, and HDR-BT Modes 178
Table 51 Timing and Drive for Start of New Frame: No Contention on A6 194
Table 52 Timing and Drive for Start of New Frame: With Contention on A6 194
Table 53 Timing and Drive for Continuation of Frame Using Repeated START 194
Table 54 DCR Byte Values Status 198
Table 55 I3C Basic Device Characteristics Register (DCR) 198
Table 56 SETBUSCON Context Byte Values Status 199
Table 57 SETBUSCON Context Values 199
Table 58 Defining Byte Values Status 200
Table 59 CCCs with Defining Bytes 200
Table 60 GETCAPS Capability Bit Values Status 203
Table 61 GETCAP1 Byte Format 204
Table 62 GETCAP2 Byte Format 204
Table 63 GETCAP3 Byte Format 205
Table 64 GETCAP4 Byte Format 206
Table 65 GETCAPS Format 2 Defining Byte Values 207
Table 66 TESTPAT Message Format 208
Table 67 CRCAP1 Byte Format 210
Table 68 CRCAP2 Byte Format 210
Table 69 DBGCAPS Message Format 211
Table 70 VTCAP1 Byte Format 213
Table 71 VTCAP2 Byte Format 214
Table 72 In-Band Interrupt MDB Values Status 215
Table 73 Mandatory Data Byte Field Format 215
Table 74 Mandatory Data Byte Value Ranges 216
Table 75 CCCs Permitted in HDR Modes, with Notes and Limitations 226
Table 76 CCCs Not Permitted in HDR Modes 227
Table 77 HDR CCC Details Per HDR Mode 234
Table 78 Optional Feature 001 Status 243
Table 79 HDR-DDR Preamble Values 247
Table 80 HDR-DDR Word Format: Command, Data, Reserved 249
Table 81 HDR-DDR Word Formats 250

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Table 82 HDR-DDR Command Word Format 254
Table 83 Read and Write Command Spaces for HDR-DDR Mode 254
Table 84 ENDXFER CCC Additional Data Byte for Sub-Command 0xF7 (HDR-DDR Protocol) 277
Table 85 Optional Feature 007 Status 297
Table 86 Optional Feature 003 Status 299
Table 87 Data Byte Bit Packing: Single Lane 304
Table 88 Data Byte Bit Packing: Dual Lane 304
Table 89 Data Byte Bit Packing: Quad Lane 304
Table 90 Transition Byte Bit Packing: Single Lane 305
Table 91 Transition Byte Bit Packing: Dual Lane with Bit2 Swizzle 305
Table 92 Transition Byte Bit Packing: Quad Lane with Bit4 Swizzle 305
Table 93 Example CRC Checksum Calculations 315
Table 94 Bus Efficiency 319
Table 95 Optional Feature 004 Status 345
Table 96 Optional Feature 005 Status 357
Table 97 Asynchronous Timing Control Modes 364
Table 98 Set Exchange Timing Information Command Codes (SETXTIME) 368
Table 99 SETXTIME Sub-Command Byte Values 369
Table 100 GETXTIME Supported Modes Byte Fields 370
Table 101 GETXTIME State Byte Fields 371
Table 102 Optional Feature 006 Status 373
Table 103 ML Lane Configurations 376
Table 104 All ML Frame Formats 386
Table 105 HDR-BT Frame Formats in Multi-Lane 389
Table 106 HDR-BT Frame Format Details and Coding Interoperability for CCCs in Multi-Lane 394
Table 107 Multi-Lane Data Transfer Control Command Codes (MLANE) 403
Table 108 MLANE Defining Bytes and Data Bytes 405
Table 109 Optional Feature 007 Status 411
Table 110 Optional Feature 008 Status 413
Table 111 Optional Feature 009 Status 415
Table 112 Controller Clock Stall Times 416
Table 113 Optional Feature 010 Status 421
Table 114 I3C Low Voltage / High Capacitive Load I/O Stage Characteristics for Push-Pull Mode and Open Drain Mode 423

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Release History

See Errata 01, Item 2

Date	Version	Description
31-Dec-2016	v1.0	Initial Board Adopted release.
11-Dec-2019	v1.1	Board Adopted release.
11-Jun-2021	v1.1.1	Board Adopted release.
17-Apr-2025	v1.2	Board Adopted release. First release using new specification framework.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

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 ***[MIPI08]***, 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 Allianceôs 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 G***. 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 ñRAND-Zò terms: that is, under royalty free (the Z references zero royalty) and otherwise reasonable and non-discriminatory terms.

As described in ***Annex G***, 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 G*** are listed in ***Annex H***. ***Annex H*** 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.

Copyright É 2016ï 2025 MIPI Alliance, Inc.

Public Release Edition

4 Relationship to MIPI I3C v1.x Specifications

The MIPI I3C Basic Specification is a subset of the MIPI I3C Specification. In the I3C Basic Specification, the terms ñI3C,ò ñI3C Device,ò and ñI3C Busò should be interpreted as referring to both I3C and I3C Basic.

In addition, I3C Basic:

- Addresses all issues addressed by all published Errata for previous versions of the I3C Specification
- Includes numerous fixes, clarifications, and editorial improvements planned for inclusion in the MIPI I3C Specification. In general, these are not new feature additions, but represent the latest understanding and definition of I3C features and capabilities.

4.1 I3C v1.1.1 Functions Not Included in I3C Basic

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

- Timing Control (***Section 6.6***), namely the portions addressing:
  - Synchronous Time Control
  - Asynchronous Time Control (Modes 1ï 3)
- Monitoring Device Early Termination Capability (***Section 6.8***)
- Device to Device(s) Tunneling (***Section 6.9***)
- HDR Ternary Modes (HDR-TSP and HDR-TSL) (***Section 6.3***)
- Multi-Lane support for SDR Mode, HDR-DDR Mode, and HDR-TSP Mode (***Section 6.7***)

4.2 Organization of This Specification

The structure of this Specification, i.e., the naming and numbering of Section headings, is consistent with the I3C v1.2 Specification. In order to preserve alignment with the I3C v1.2 Specificationôs structure, numerous references to I3C v1.2 functions and capabilities that were removed for I3C Basic v1.2 are retained. In many cases the associated section headings are retained with a note that the content is only available in the full I3C v1.2 Specification.

4.3 I3C Basic v1.2 as an Update of I3C Basic v1.1.1

I3C Basic v1.2 is an updated, clarified version of I3C Basic v1.1.1, and is also a strict subset of the full I3C v1.2 Specification. As such, I3C Basic v1.2 includes numerous clarifications and corrections that were also applied to the full I3C v1.2 Specification. Note that I3C Basic v1.2 does not include any new features as compared with I3C Basic v1.1.1.

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

MIPI I3C Basic Devices, whether Primary Controller, Secondary Controller, or Target, 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.

Copyright É 2016ï 2025 MIPI Alliance, Inc.
Public Release Edition

1 Structure of This Specification (Informative)

This specification contains the full specification for MIPI I3C Basic, including all required and optional elements.

The specification has the following structure:

Table 1 Structure of This Specification

Sections	Topics
Preface to the I3C Basic Specification **1 Structure of This Specification (Informative)** **2 Introduction**	General introductory material
3 Technical Overview (Informative)	Presents global technical overview of I3C Basic technology, including all required and optional elements.
4 Required Elements	Specifies all essential core elements of an I3C Basic Device, including the I3C SDR Protocol and all electrical and timing specifications. ***Annex A*** summarizes changes in this version.
5 Assigned Values	Lists all MIPI Alliance assigned values for I3C Basic. This includes values for: • DCR Bytes • SETBUSCON context Bytes • CCC Bytes • IBI MDB Capability Bits • and other items
6 Optional Features	Specifies all I3C Device Optional Features defined by MIPI Alliance. Every defined Optional Feature is specified independently in a separate subsection.
Annex A	Summary of ***Section 4*** changes in this version.
Annex B	Syntax for I3C Basic Implementation Summaries.
Annex C	Details for I3C Basic Communication format
Annex D	SDR Mode Error Type Origins
Annex E	I3C Basic Controller state machines
Annex F	Typical I3C Basic Protocol communications

See Errata 01, Item 3

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

1.1 Managing Optional Element Combinations

Because the number of possible combinations of I3Côs optional elements is large, this specification includes measures intended to help implementers manage the resulting complexity:

- A table on the specification cover summarizes which separable blocks (i.e.: ***Section 4 Required Elements*** and each Optional Feature) have been updated in this specification version.
- Throughout the specification, each separable block has its own independent version number and change history.
- To help implementers determine whether a given I3C specification version requires revision of a given I3C product, blocks that have not changed in this specification version are clearly indicated.
- Because some of the optional elements are mutually incompatible, each Optional Feature indicates any other Optional Features with which it is technically incompatible.
- ***Annex B*** presents a concise syntax which is recommended for expressing which optional elements an I3C product does and does not implement. This syntax is intended for use in product data sheets and other technical marketing, to simplify comparison between products and matching application requirements to I3C parts.

Copyright É 2016ï 2025 MIPI Alliance, Inc.
Public Release Edition

2 Introduction

I²C was introduced many years ago to provide a low cost method for connecting a processor to a set of devices to support the needs of consumer electronics; various flavors of I²C have continued to be widely used in many industries where its low cost ability to connect a set of devices (AKA Multi-Drop) has served the needs reasonably well, although often with the need for side-band pins, and often with interoperability challenges.

In mobile wireless devices the I²C tradition continued but changing requirements had made I²C more and more problematic. With the increasing use of sensors, this started creating fragmentation in terms of choice of bus, including I²C, SPI, and UART to address bandwidth and other issues. Further, none of those buses addressed the pin count problems made far worse with more complex end-devices, including the need for various per-device GPIOs in parallel to the bus to cover interrupt (Target Device notifying Controller), sleep and reset signals, and so on. On top of that, the increasing number of devices with increasing amounts of data to send/receive has put pressure on bus bandwidth and latency. Adding more buses adds more pins, more traces, and more power.

The MIPI I3C interface has been developed to address these sensor integration concerns, as well as those historically encountered while using I²C, SPI, and UART in any product, by providing a fast, low cost, low power and managed, two-wire digital interface. MIPI worked to address a set of key issues across: performance, power, extraneous pins (e.g., Targets interrupting and reset control), interoperability, bus management, and error-handling/stuck-bus protections; it chose to do all of this while maintaining backwards compatibility with legacy I²C, to aid in transition and evolution.

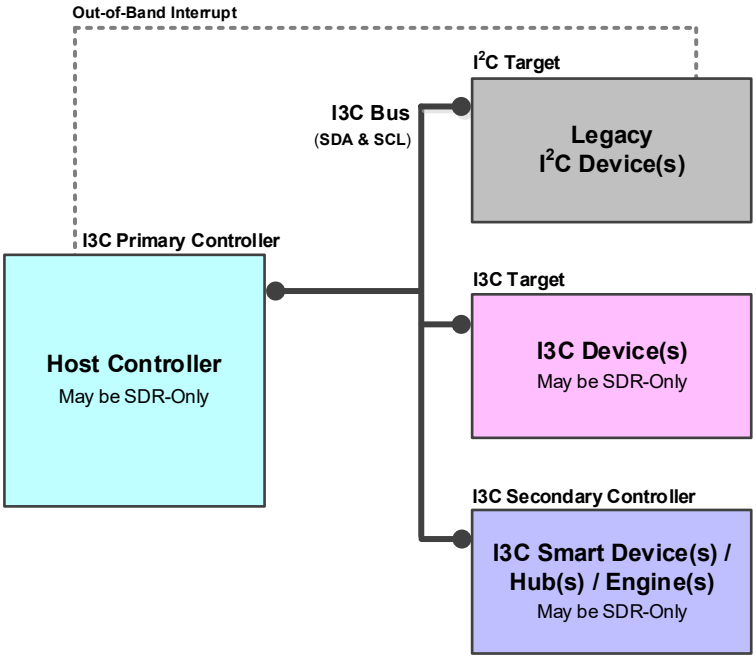


Figure 1 I3C System Diagram

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

2.1 Scope and Purpose (Informative)

The following topics are in scope for this Specification:

- I3C interface protocols and commands
- Electrical specifications, such as timing and voltage levels
- Support for specific classes of Devices (e.g., sensors)

The following topics are out of scope for this Specification:

- Mechanical, system, and implementation details within an I3C Device
- ESD (Electrostatic Discharge) structures
- System power management
- Use case specific data or format definitions besides Bus management command codes

The I3C interface is intended to improve upon the features of the I^2C interface ***[NXP01]***, preserving backward compatibility. This Specification defines a standard Multi-Drop interface between Host processors and peripheral Devices (e.g., sensors).

Implementing the I3C Specification greatly increases the flexibility system designers have to supplant incumbent interfaces (i.e., I^2C, SPI, UART) and support a wide array of Devices of increasing complexity and diversity in their system (e.g., sensor subsystem) and at as low a cost as possible.

2.2 I3C Key Features (Informative)

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

Therefore, the I3C interface features:

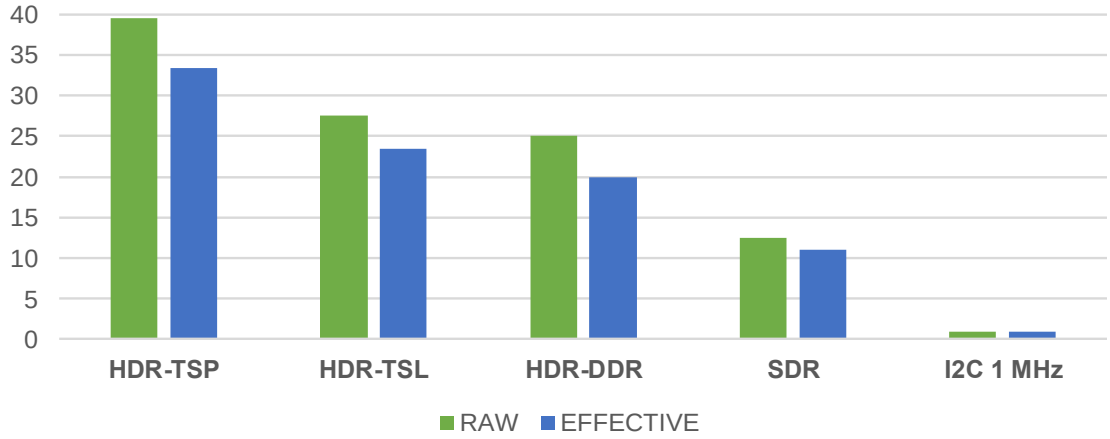
- Two-wire serial interface up to 12.5 MHz using Push-Pull
- Legacy I^2C Device co-existence on the same Bus (with some limitations)
- Dynamic Addressing while supporting Static Addressing for Legacy I^2C Devices
- Legacy I^2C messaging
- I^2C-like Single Data Rate messaging (SDR)
- Optional High Data Rate messaging Modes (HDR):
  - HDR-DDR for Double Data Rate
  - **NOT INCLUDED IN I3C BASIC:** HDR-TSL for Ternary Symbol Legacy-inclusive-Bus (I^2C Devices allowed)
  - **NOT INCLUDED IN I3C BASIC:**HDR-TSP for Ternary Symbol for Pure Bus (no I^2C Devices allowed)
  - HDR-BT for Bulk Transport
- Multi-Drop a multitude of physically and virtually attached Devices
- Multi-Controller capability
- In-Band Interrupt support
- Hot-Join support
- **PARTLY INCLUDED IN I3C BASIC:** Synchronous Timing Support and Asynchronous Time Stamping
- Grouped Addressing
- Target Reset without additional wires
- **NOT INCLUDED IN I3C BASIC:**Monitoring Device Early Termination
- **NOT INCLUDED IN I3C BASIC:**Device to Device(s) Tunneling
- **PARTLY INCLUDED IN I3C BASIC:**Multi-Lane Data Transfer

Copyright É 2016ï 2025 MIPI Alliance, Inc.
Public Release Edition

The I3C interface provides improved efficiencies in Bus power while providing significant speed improvements over I^2C. ***Figure 2*** through ***Figure 5*** show different Bus Mode options that provide tradeoffs for performance/power vs. target Device complexity. The following conditions were assumed for all calculations:

- Devices are in close vicinity:
  - Targets clustered close to Controller
  - No delay on the wires
- Total leakage on each wire: 4 µA
- Driver's internal resistance: 90 Ω
- Total Bus capacitance: 50 pF
- Bus main clock: 12.5 MHz
- V_{DD} = 1.8 V
- Pull-Up resistor on Open-Drain: 2833 Ω
- Equal probability for 1 and 0 on data transmission

Figure 2 shows the Raw and Effective single-Lane bit rates of the I3C Modes compared to I^2C FM+ (1 MHz).



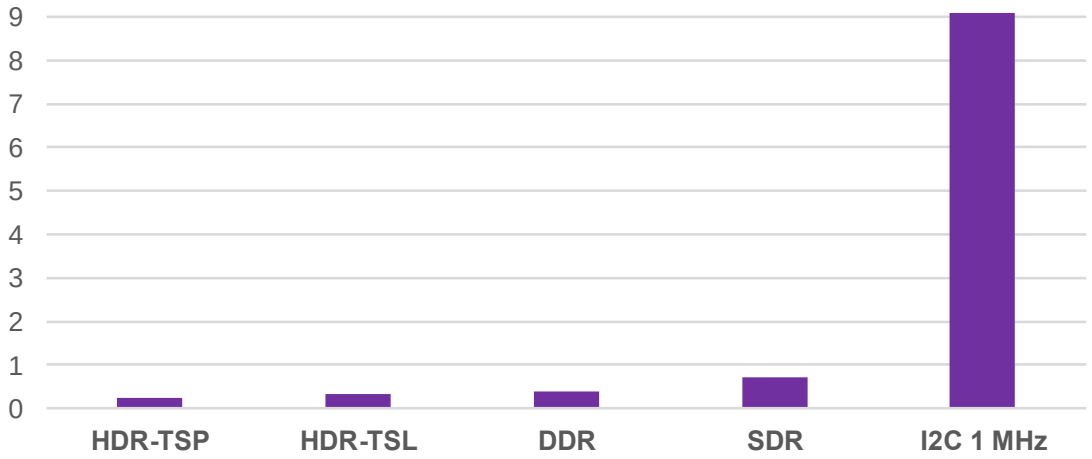
Note:

HDR-TSP Mode and HDR-TSL Mode are not included in I3C Basic.
To gain access to these HDR Modes, please contact MIPI Alliance.

Figure 2 I3C vs. I^2C FM+ Data Blocks Bit Rates in Mbps (12.5 MHz Clock)

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Figure 3 shows the effective single lane transfer time of 1 kB block of data for the I3C Modes compared to I²C FM+ (1 MHz).

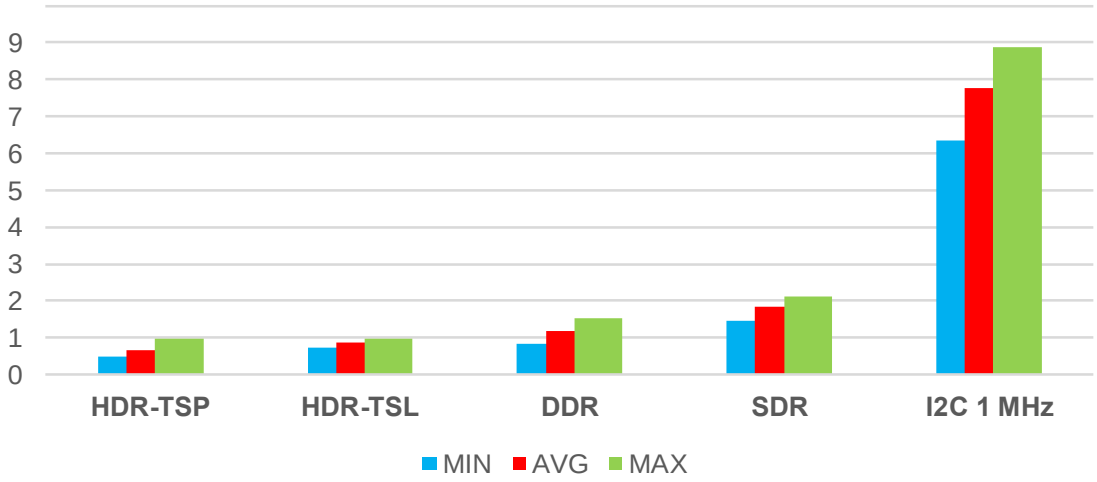


Note:

HDR-TSP Mode and HDR-TSL Mode are not included in I3C Basic.
To gain access to these HDR Modes, please contact MIPI Alliance.

Figure 3 I3C vs. I²C FM+ 1 kB Effective Data Transfer Times in ms

Figure 4 shows the amount of energy (µJ) consumed for transferring an effective 1 kB block of data on single-Lane versions of I3C Modes compared to I²C FM+ (1 MHz).



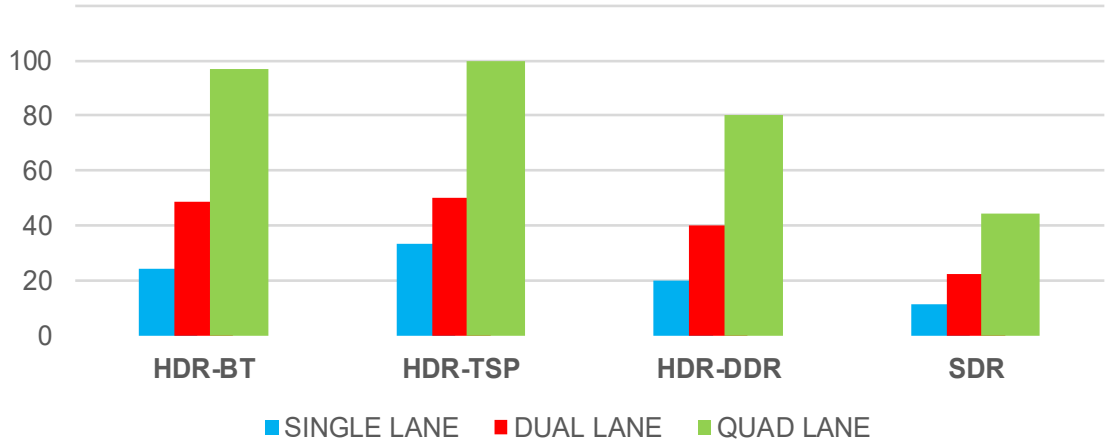
Note:

HDR-TSP Mode and HDR-TSL Mode are not included in I3C Basic.
To gain access to these HDR Modes, please contact MIPI Alliance.

Figure 4 I3C vs. I²C FM+ Energy per 1 kB Effective Data Block, in µJ

Copyright © 2016–2025 MIPI Alliance, Inc.
All rights reserved.
Confidential

179 ***Figure 5*** shows the effective bitrates (Mbps) of the different I3C Modes over Multi-Lane.



180

181 ***Note:***

182 *Multi-Lane support for HDR-TSP Mode and SDR Mode are not included in I3C Basic.*

183 *To gain access to these capabilities, please contact MIPI Alliance.*

Figure 5 I3C Multi-Lane Effective Bit Rates, in Mbps

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

2.3 Terminology

2.3.1 Use of Special Terms

The MIPI Alliance has adopted Section 13.1 of the *IEEE Standards Style Manual*, which dictates use of the words ósha llô, óshouldô, ómayô, and ócanôin the development of documentation, as follows:

> The word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the Specification and from which no deviation is permitted (*shall* equals *is required to*).
>
> The use of the word *must* is deprecated and shall not be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.
>
> The use of the word *will* is deprecated and shall not be used when stating mandatory requirements; *will* is only used in statements of fact.
>
> The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).
>
> The word *may* is used to indicate a course of action permissible within the limits of the Specification (*may* equals *is permitted to*).
>
> The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

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

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

2.3.2 Definitions

1-Wire: A communication bus protocol.

ACK: Short for ócknowledgeô. See also **NACK**.

Active Controller: The I3C Device that presently has control (i.e., the Controller Role) of the I3C Bus.

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

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

Arbitrable: Subject to decision by Arbitration.

Arbitration: If two Devices start transmission at the same time, then Arbitration is required to determine Bus control. Arbitration could also be required during a Target transmission if a Controller addresses multiple Targets.

Bit Banged: When GPIOs are manually changed by software to follow a protocol. For example, software could operate in I^2C Controller mode quite easily by writing the SCL pin and reading/writing the SDA pin in between.

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

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

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

Bus Available Condition: A period during which the Bus Free Condition, after a STOP and before a START, is sustained continuously for a duration of at least t_{AVAL} (see ***Table 49*** and ***Section 4.3.3.2.2***).

Bus Free Condition: A period after a STOP and before a START with a duration of at least t_{CAS} (see ***Table 48*** and ***Table 49***) for a Pure Bus, and at least t_{BUF} (see ***Table 48*** and ***Section 4.3.3.2***) for a Mixed Bus.

Copyright É 2016ï 2025 MIPI Alliance, Inc.
Public Release Edition

Bus Idle Condition: A period during which the Bus Free Condition, after a STOP and before a START, is sustained continuously for a duration of at least t_{IDLE} (see ***Table 49*** and ***Section 4.3.3.2.3***).

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

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

Clock to Data Turnaround Time: The time duration between reception of an SCL edge and the start of driving an SDA change. See t_{SCO} in ***Table 50***.

Coding: See Data Transfer Coding.

Controller: An I3C Device that is capable of controlling the I3C Bus.

Note:

In previous versions of the I3C Basic Specification, a Controller Device was called a "Master Device". This version of the I3C Basic Specification has deprecated the term "Master", and now uses the updated normative term "Controller." Please note that the technical definition of such a Device, and its Role on an I3C Bus, are unchanged.

*Unless specified otherwise, a Controller as a Device (or its Role) is a generic term between I3C Basic and I3C. Such a Device may comply with either this I3C Basic Specification or a compatible version of the full I3C Specification **[MIPI08]** in cases where this distinction is not technically pertinent.*

Controller Role: State in which the device assumes control of the Bus. See Active Controller.

Controller Role Request: A method whereby a Controller-capable Device a Secondary Controller) requests to become the next Active Controller of the Bus, using a form of interrupt request (i.e., In-Band Interrupt).

CRC-5: Cyclic Redundancy Check with fifth-order polynomial length:

$$\text{CRC-5} = X^5 + X^2 + X^0$$

CRC-16: Cyclic Redundancy Check with sixteenth-order polynomial length, as defined by ANSI and used in USB (also known as CRC-16-IBM):

$$\text{CRC-16} = X^{16} + X^{15} + X^2 + 1$$

CRC-32: Cyclic Redundancy Check with thirty-second-order polynomial length:

$$\text{CRC-32} = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

Data Transfer Coding: A specific contract for extending an I3C Mode to use additional Data Lanes (i.e., Multi-Lane). The contract may include changes to the framing, flow, and data bit/byte packing of the structured protocol elements. In some sections of this Specification related to Multi-Lane, a Data Transfer Coding may also be referred to as simply a "Coding".

Device: A Controller or Target.

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

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

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Flow Element: A structured protocol element, such as a Word or a Block, used as portion of the framing of a given flow to indicate various phases of data transmission to define the Frame, as well as other necessary conditions such as Bus Turnaround.

Frame: A Frame begins with a START, followed by the Address of the Target(s), Data, and finally a STOP.

High: Defines a signal level that is a logical ó1ô.

High Data Rate (HDR): High Data Rate Modes that achieve higher speed than SDR Mode.

High-Keeper: A weak Pull-Up type Device used to keep lines on High level when their drivers are on High-Z condition.

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

Hot-Join: Targets that join the Bus after it is already started, whether because they were not powered previously or because they were physically inserted into the Bus; the Hot-Join mechanism allows the Target to notify the Controller that it is ready to get a Dynamic Address.

Hot-Join Request: A method whereby a Target joins the Bus and requests a Dynamic Address from the Controller, using a special Interrupt request with a reserved I3C Address.

Hot-Joining Device: A Device that is currently attempting to join the Bus (i.e., as a Target or Secondary Controller) using the Hot-Join Request.

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

I²C Device: A Controller or Target that meets the requirements of the I²C Specification ***[NXP01]***.

I3C Basic Device: A Controller or Target that meets the requirements of this I3C Basic Specification (not the full I3C Specification ***[MIPI08]***). That is, an I3C Basic Device supports only the I3C Basic core capabilities plus any optional features and capabilities defined in this I3C Basic Specification; it does not support any optional features defined only in the full I3C Specification.

Note:

An I3C Basic Device is defined to be interoperable with I3C with respect to the core capabilities that I3C and I3C Basic share, and with respect to I3C Basic's optional features and capabilities. See the Preface for an overview of I3C v1.2 functions not included in I3C Basic and how I3C Basic Devices work with I3C v1.x Devices; see also other sections in this Specification that define the specific features.

I3C Controller: See Controller.

I3C Device: A Controller or Target that meets the requirements of either this I3C Basic Specification, or of a version of the full I3C Specification that is interoperable with I3C Basic (e.g., I3C v1.2 ***[MIPI08]***).

Note:

In this I3C Basic Specification, the term I3C Device typically describes compatibility with I3C in a generic sense. An I3C Device does not always imply an I3C Basic Device, since an I3C Device can support either I3C or I3C Basic where such distinctions are not technically pertinent. However, in cases where an I3C Bus contains both I3C Basic Devices and other I3C Devices that comply with the full I3C Specification and that also use features or capabilities that are not included in I3C Basic, there might be restrictions on which features or capabilities can be used with such an I3C Basic Device, and certain compatibility concerns should be understood and addressed. See the Preface for an overview.

I3C Master: Deprecated term, see Controller.

I3C Slave: Deprecated term, see Target.

I3C Target: See Target.

Legacy I²C: I3C maintains the industry standard architecture of I²C and supports existing I²C Target Devices. I3C does not support I²C Bus Controllers.

Copyright É 2016ï 2025 MIPI Alliance, Inc.
Public Release Edition

Note:

A Legacy I²C "Target" Device is defined to have the same meaning as an I²C "Slave" Device, as defined in the current version of the I²C Specification ***[NXP01]****.*

Low: Defines a signal level that is a logical '0'.

Master: Deprecated term, see Controller.

Message: A packetized communication between Devices.

Minimal Bus: An I3C Bus with one Controller Device (potentially with reduced functionality) and one active Target Device with a fixed and reserved Target Address value of 7'h01.

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

Mixed Bus: An I3C Bus topology with both I²C and I3C Devices present on the I3C Bus. May be either a Mixed Fast Bus or a Mixed Slow/Limited Bus.

Mixed Fast Bus: I3C Bus topology with both I²C and I3C Devices present on the I3C Bus, where the I²C Devices have a true I²C 50 ns Spike Filter on the SCL line.

Mixed Slow/Limited Bus: I3C Bus topology with both I²C and I3C Devices present on the I3C Bus, where the I²C Devices do not have a true I²C 50 ns Spike Filter on the SCL line.

Mode: Distinguishes different data transfer methods used in I3C and I3C Basic including Legacy I²C Mode, Single Data Rate Mode (SDR), High Data Rate Mode (HDR), Dual Data Rate Mode (HDR-DDR), Ternary Symbol Legacy Mode (HDR-TSL), Ternary Symbol for Pure Bus Mode (HDR-TSP), and HDR Bulk Transport Mode (HDR-BT). Note that I3C Basic does not include support for the HDR Ternary Modes that are defined in the full I3C Specification: Ternary Symbol Legacy Mode (HDR-TSL) and Ternary Symbol for Pure Bus Mode (HDR-TSP).

Multi-Controller: Multiple Bus Controllers present on the Bus. Used when multiple nodes on the Bus need to initiate a transfer.

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

Multi-Lane: A method of using more physical wires (i.e., additional Data Lanes along with the SDA Lane) to achieve higher data transfer rates in various I3C Modes.

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

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

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

Overflow: A state in which a Device or process is being fed with data at a faster speed than the data is being consumed.

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

Peripheral: The portion of the logic of a Device that implements the I3C Bus protocol, usually including any FIFOs or other buffers, and an interface to the rest of the system, such as an internal bus to a processor.

Preamble: The bits preceding the data Words in HDR-DDR.

Primary Controller: The Controller-capable I3C Device that initializes the I3C Bus and performs configuration of all Target Devices. It acts as the authority for the Bus in its initial state, and becomes the first Active Controller once the Bus is configured.

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

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

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

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

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

Read Segment: A series of Flow Elements that comprise a Read transaction addressing a Target, within the framing of a given HDR Mode. Usually part of a Direct SET CCC flow.

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

Route: Path through a Device connecting two different I3C Buses. A given Device may contain one or more Routes.

Routing Device: Device on the I3C Bus that allows conversations between two or more different I3C Buses through integrated logic on the given Device. The Routing Device (as distinct from a Bridge Device) buffers/queues the transactions, causing the Device to be non-transparent.

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

Secondary Controller: A Controller-capable I3C Device that initially acts as a Target, but can also accept the Controller Role from any Active Controller (including the Primary Controller). Once a Secondary Controller accepts the Controller Role it becomes the new Active Controller, and may then pass the Controller Role along: either back to the previous Active Controller, or on to any other Controller-capable Device present on the I3C Bus.

Segment: A series of Flow Elements that comprise either a Read or Write transaction addressing a Target or Group, within the framing of a given HDR Mode. Usually part of a Direct CCC flow.

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

Slave: Deprecated term, see Target.

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

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

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

START Request: A method for a Target to force the Controller to issue a START on an idled I3C Bus.

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

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

Swizzle: To swap or interleave bits.

T-Bit: Alternative use of the 9th bit.

Target: A Target Device can only respond to either Common or individual commands from a Controller.

Note:

*In previous versions of the I3C Basic Specification (as well as some previous versions of the full I3C Specification), a Target Device was called a "Slave" Device. This version of the I3C Basic Specification has deprecated the term "Slave", and now uses the updated normative term "Target." Please note that the technical definition of such a Device, and its Role on an I3C Bus, are unchanged. Unless specified otherwise, a Target as a Device (or its Role) is a generic term between I3C Basic and I3C. Such a Device may comply with either this I3C Basic Specification or a compatible version of the full I3C Specification **[MIPI08]** in cases where this distinction is not technically pertinent.*

Copyright É 2016ï 2025 MIPI Alliance, Inc.
Public Release Edition

Target Reset: The Target Reset mechanism allows the Controller to request a reset of specific Target Devices, including: reset of the I3C Peripheral, reset of the whole Device, and wake from deepest sleep. Target Reset uses a specialized pattern of Bus activity (specifically, an extension of the HDR Exit Pattern) that cannot occur in any other way in the I3C protocol.

Termination Marker: A Flow Element or other unique HDR Pattern that signifies the end of a specific Flow Element, a Segment, or the data message sent by the Controller as part of a Broadcast CCC flow.

Ternary Mode: These Modes are defined in the full I3C Specification, and I3C Basic does not include support for them: HDR-TSP (Ternary Symbol for Pure Bus) Mode and HDR-TSL (Ternary Symbol Legacy) Mode.

Timestamping: The act of determining and assigning the time at which an event occurred.

Timing Control: Methods of exchanging and controlling system timing information, with the intent to synchronize and/or Timestamp I3C Bus Devices and events. See ***Section 6.6***.

Underrun: A state in which a Device or process is being fed with data at a slower speed than the data is being consumed.

Virtual Target: A physical Device that represents multiple I3C Targets, often implemented as multiple Target Devices sharing a common set of I3C pads. Additionally, there can be multiple Virtual Targets inside a single Device or a Bridge/Hub that manages the Bus connection. See also ***Table 7***, ***Section 4.3.2.1.2***, and ***Section 4.3.7.3.19***.

Virtual Target Detect Operation: A method for determining which Virtual Targets share the same Peripheral logic within a particular I3C Device.

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

Write Segment: A series of Flow Elements that comprise a Write transaction addressing a Target or Group, within the framing of a given HDR Mode. Usually part of a Direct SET CCC flow.

2.3.3 Abbreviations

ACK	Acknowledge
e.g.	For example (Latin: exempli gratia)
HDR-TSx	HDR-TSP and/or HDR-TSL
High-Z	An output driver that is set to high impedance mode (cannot source or sink current)
i.e.	That is (Latin: id est)
NACK	Not Acknowledge
P	STOP
PHY	Physical Layer
S	START
Sr	Repeated START
T	Transition Bit

2.3.4 Acronyms

AXI	Advanced eXtensible Interface
BCR	Bus Characteristics Register
CCC	Common Command Code
CRC	Cyclic Redundancy Check

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

CRR	Controller Role Request
D2DT	Device to Device(s) Tunneling
DCR	Device Characteristics Register
DDR	Double Data Rate
ESD	Electro Static Discharge
FSM	Finite State Machine
HDR	High Data Rate
HDR-DDR	HDR Double Data Rate Mode
HDR-TSL	Ternary Symbol Legacy
HDR-TSP	Ternary Symbol for Pure Bus (no I^2C Devices)
IBI	In-Band Interrupt
IMU	Inertial Measurement Unit
LCR	Legacy Characteristics Register
LIN	Local Interconnect Network
LSB	Least Significant Byte
LSb	Least Significant Bit
Mbps	Megabits per second
MHz	Mega Hertz
MID	MIPI Manufacturer Identification ***[MIPI01]***
ML	Multi-Lane
MSB	Most Significant Byte
MSb	Most Significant Bit
NVMEM	Non-Volatile Memory
OCP	Open Core Protocol
OD	Open Drain
PUR	Pull-Up Resistor
SCL	Serial Clock
SDA	Serial Data
SDR	Single Data Rate
SPI	Serial Peripheral Interface
SRFF	State Retention Flip-Flop
UART	Universal Asynchronous Receiver Transmitter

Copyright É 2016ï 2025 MIPI Alliance, Inc.
Public Release Edition

2.4 References

2.4.1 Informative References

[MIPI03] *MIPI Alliance I3C Application Note: General Topics*, App Note version 1.1, MIPI Alliance, Inc., 27 April 2022 (approved 27 July 2022).

[MIPI08] MIPI Alliance, Inc., "MIPI I3C & MIPI I3C Basic" section of public website, https://www.mipi.org/specifications/i3c-sensor-specification.

[MIPI09] *MIPI Alliance Specification for I3C Basic*, *diff file between version 1.1.1 and version 1.2*, MIPI Alliance, Inc., 22 March 2025.

[MIPI10] *MIPI Alliance I3C Application Note: Virtual Devices and Virtual Targets*, App Note version 1.0, MIPI Alliance, Inc., 30 August 2021 (approved 4 September 2021).

[MIPI11] *MIPI Alliance Specification for I3C Basic*, version 1.1.1, MIPI Alliance, Inc., 9 June 2021 (Adoped 21 July 2021) **See Errata 01, Item 4**

[NXP01] UM10204, *I²C-bus specification and user manual*, Rev. 6, NXP Semiconductors N.V., 4 April 2014.

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

2.4.2 Normative References

[MCTP01] *MCTP I3C Transport Binding Specification*, version 1.0 or newer, DMTF, https://dmtf.org.

[MIPI01] MIPI Alliance, Inc., "MIPI Alliance Manufacturer ID Page", https://mid.mipi.org.

[MIPI02] MIPI Alliance, Inc., "Current I3C Device Characteristic Register (DCR) Assignments", https://www.mipi.org/MIPI_I3C_device_characteristics_register.

[MIPI04] *MIPI Alliance Specification for Debug Over I3C*, version 1.1, MIPI Alliance, Inc., 1 February 2024 (Board Adopted 26 May 2024).

[MIPI05] MIPI Alliance, Inc., "I3C Mandatory Data Byte (MDB) Assignments", https://www.mipi.org/MIPI_I3C_mandatory_data_byte_values_public.

[MIPI07] MIPI Alliance, Inc., "I3C SETBUSCON Table", https://www.mipi.org/MIPI_I3C_bus_context_byte_values_public.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright É 2016ï 2025 MIPI Alliance, Inc.
Public Release Edition

3 Technical Overview (Informative)

This Section generally describes the I3C Bus, the I3C interface, and I3C Controller and Target Devices.

I3C is a two-wire bidirectional serial Bus, optimized for multiple Target Devices and controlled by only one I3C Controller Device at a time. I3C is backward compatible with many Legacy I^2C Devices, however I3C Devices support significantly higher speeds, new communication Modes, new Device Roles, and an ability to change Device Roles over time (i.e., the initial Controller can pass the Controller Role to another I3C Device that supports the Secondary Controller feature).

I3C includes:

- Support for many Legacy I^2C Target Devices and Messages
- **I3C Single Data Rate (SDR) Mode:** New I3C enhanced version of the I^2C protocol supporting private Messages, and adding two kinds of standard built-in Messages:
  - **Broadcast Messages,** which are sent to all I3C Targets on the Bus
  - **Direct Messages,** which are addressed to specific Targets
- **I3C High Data Rate (HDR) Modes:** Additional optional Modes that add significant functionality:
  - **Dual Data Rate (HDR-DDR) Mode:** Uses same signaling as SDR Mode, but it transfers data on both clock edges to achieve about twice the speed of SDR. Not significantly different from the I^2C protocol
  - **Bulk Transport (HDR-BT) Mode:** Gives the highest possible throughput using a Clock-and-Data transmission model leveraging dual data rate over single-, dual-, or quad-lane configurations

Readers interested in a detailed, low-level introduction to the operation of the I3C Bus for each of the defined I3C Modes are encouraged to study and compare the example waveforms shown in informative ***Annex F***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3.1 I3C Fundamental Principles

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

The two wires are designated SDA and SCL:

- SDA (Serial Data) is a bidirectional data pin
- SCL (Serial Clock) can be either a clock pin or a Bi-directional data pin while in certain HDR Modes

An I3C Bus supports the mixing of various Message types:

1. I^2C-like SDR Messages, with SCL clock speeds up to 12.5 MHz
2. Broadcast and Direct Common Command Code (CCC) Messages that allow the Controller to communicate to all or one of the Targets on the I3C Bus, respectively
3. HDR Mode Messages, which achieve higher data rates than SDR Mode per equivalent clock cycle
4. I^2C Messages to Legacy I^2C Targets
5. Target-initiated START Request to the Controller, for example to send an In-Band Interrupt or to request the Controller Role

An example traffic pattern on the I3C Bus is shown in ***Figure 6***.

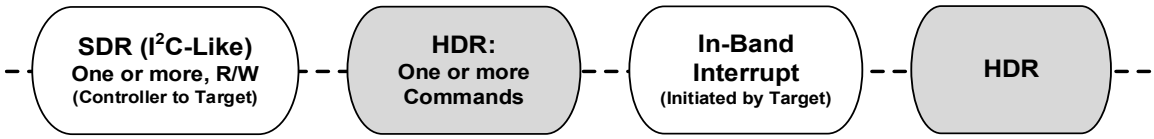


Figure 6 Example of Data Traffic on the I3C Bus

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Figure 7 illustrates how I3C communication is initiated:

- All I3C communication occurs within a Frame. The Frame begins with a START, followed by one or more transfers, and a STOP.
- For the HDR Modes:
  - First the dedicated Broadcast I3C Address (7'h7E) is issued to all Targets on the I3C Bus.
  - Then one of the Enter HDR CCCs is issued, indicating that the Controller is entering an HDR Mode. Each HDR Mode has its own Enter HDR CCC.
  - This is followed by one or more HDR transfers.
  - HDR Mode is ended by using the HDR Exit Pattern protocol.

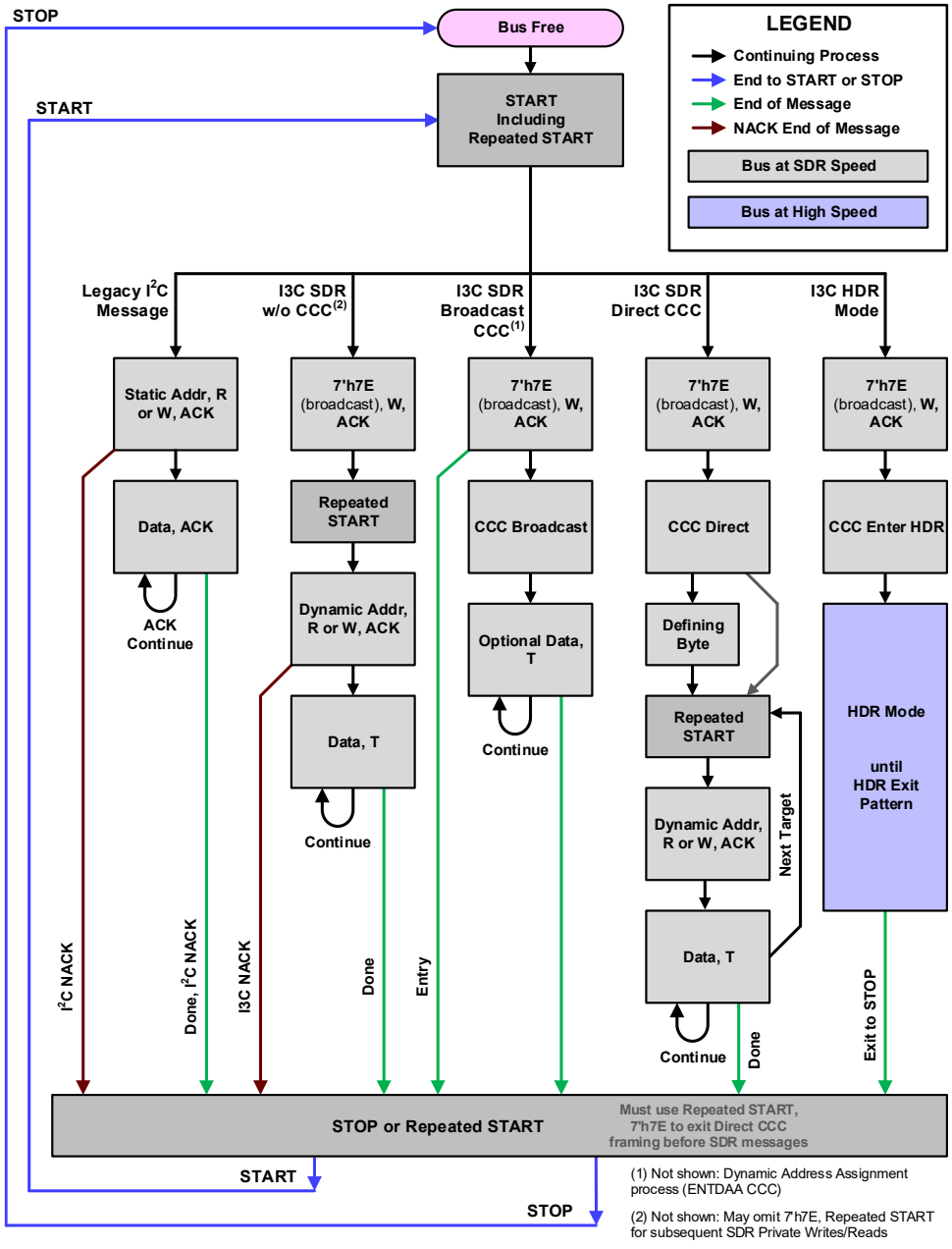


Figure 7 I3C Communication Flow

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

I3C is based on a Frame encapsulation approach. A Frame includes a Data Payload. The transfer protocol for the Data Payload is either SDR or HDR. Frames are bordered by I^2C-like Bus management.

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

The Header following a START allows for Bus Arbitration. The Controller uses the Header to address Target Device(s). I3C Target Devices(s) may use the Header Arbitration for multiple purposes: for In-Band Interrupt, for Hot-Join, and for Secondary Controller functionality.

ENTHDRx Common Command Codes (CCCs) are used to enter the High Data Rate (HDR) Modes. It is important to understand that I3C Bus activity for the HDR Message does not follow the Legacy I^2C format. Supported HDR protocols use the SDA line and the SCL line to transfer commands, data, and other control information, using alternate formats specified in ***Section 6.1***.

I3C allows only one Controller to have control of the I3C Bus at a time. Mechanisms for handoff of the Controller Role from one Device to another Device are provided.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3.2 I3C Controller and Target Devices

A given I3C Bus always has one Controller and one or more Targets. This Section generally describes I3C Controller Devices and I3C Target Devices.

A given I3C Device can be designed to function either solely as an I3C Controller, solely as an I3C Target, or with both I3C Controller and I3C Target capabilities.

An I3C Device with both I3C Controller and I3C Target capabilities cannot function as both Controller and Target at the same time, instead it must be configured either as an I3C Target Device or as an I3C Controller Device. Such an I3C Device can be initially configured (initialized) on an I3C Bus either as the Controller of that I3C Bus, or as a Target on that I3C Bus. However for that I3C Bus to function properly, only one of the multiple I3C Devices on the Bus can be initially configured (initialized) as an I3C Controller Device. That I3C Device will have the "Primary Controller" Device Role, and will be the first I3C Device on the Bus to serve as Active Controller; all other I3C Devices and Legacy I^{2}C Devices on the I3C Bus will be initially configured (initialized) as Targets.

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

I3C defines several Controller and Target Device Roles (see ***Table 2*** and ***Table 4***) to reflect the functional capabilities of a given I3C Controller or Target Device. A given I3C Device must support at least one Device Role, and can be designed to support multiple Device Roles. Every I3C Device exposes the Device Roles it supports via its Bus Characteristics Register (BCR, see ***Section 4.3.1.2.1***).

Table 2 Roles for I3C Compatible Devices

Device Type	Device Role	Description
I3C Controller[1]	I3C Primary Controller	Initially configures I3C Bus, has HDR support
	SDR-Only Primary Controller	Initially configures I3C Bus, no HDR support
	I3C Secondary Controller	Can control the Bus but initially functioning as Target
	SDR-Only Secondary Controller	Can control the Bus but initially functioning as Target, no HDR support
I3C Target[2]	I3C Target	Ordinary I3C Target, no Controller capability
	SDR-Only Target	Ordinary I3C Target, no Controller capability, no HDR support
	I^{2}C Target	No I3C Controller or I3C Target capabilities

Note:

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3.2.1 I3C Controller Device

An I3C Bus requires there to be exactly one I3C Device at a time functioning as an I3C Controller Device. In I3C terms, this I3C Controller Device is the Active Controller at that time. In typical applications, the Active Controller is the I3C Device on the Bus that sends the majority of the I3C Commands (CCC), addressing either all Targets (Broadcast CCCs) or specific individual Targets (Directed CCCs). The Active Controller is also the only Device on the I3C Bus allowed to send I^2C Messages.

In addition to sending I3C Commands and I^2C Messages, an I3C Controller Device also:

- Generates the Bus clock when in SDR Mode, HDR-DDR Mode, and HDR-BT Mode
  - I3C Targets may optionally generate the Bus clock in HDR-BT Mode, for Read transfers.
  - Note that there is no traditional clock in HDR Ternary Modes (which are not included in I3C Basic).
- Manages Pull-Up structures
- Manages Dynamic Address Assignment while acting as the Active Controller.
  - There are methods based on SETDASA, SETAASA, or ENTDAA (including Hot-Join events).
- Manages START Requests from I3C Target Devices on the Bus along with Address Arbitration requests:
  - For In-Band Interrupts
  - For Hot-Join events
  - To become Active Controller
- Supports I^2C Legacy Target Devices
- Supports I3C SDR Mode

In addition, an I3C Controller Device can optionally support any combination of I3C's defined HDR Modes.

Figure 8 is a block diagram of a typical generic I3C Controller Device.

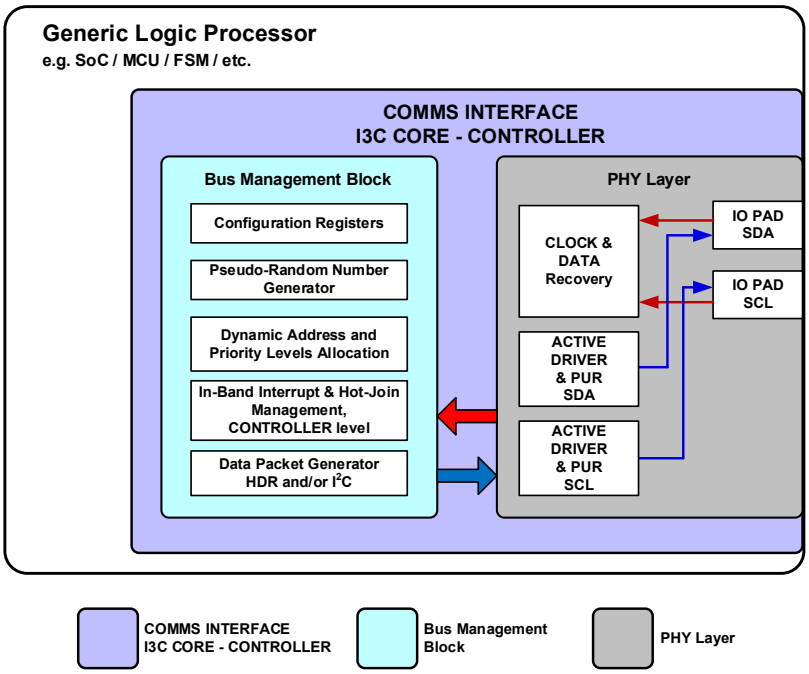


Figure 8 I3C Controller Device Block Diagram

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3.2.1.1 I3C Controller Device Roles

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

Primary Controller Device Roles:

- **Primary Controller:** The I3C Controller Device on the I3C Bus that initially configures the I3C Bus and serves as the first Active Controller. Only one I3C Device on a given I3C Bus can be the Primary Controller, i.e., this responsibility cannot be passed on to any other I3C Device on the I3C Bus. Supports both SDR Mode and at least one HDR Mode.
- **SDR-Only Primary Controller:** A Primary Controller that only supports I3C's SDR Mode, i.e., does not support any of the HDR Modes.

Secondary Controller Device Roles:

- **I3C Secondary Controller:** Any I3C Device on the I3C Bus, other than the Active Controller, with I3C Controller Capability. There can be multiple Secondary Controllers on an I3C Bus at the same time. By definition, a Secondary Controller functions as an I3C Target Device unless it becomes Active Controller. Supports both SDR Mode and at least one HDR Mode.
- **SDR-Only Secondary Controller:** A Secondary Controller 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:

*Active Controller is not formally defined as an I3C Device Role, and is not exposed in the I3C Device's Bus Characteristics Register (BCR, see **Section 4.3.1.2.1**).*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3.2.2 I3C Target Device

The maximum number of Devices that an I3C Bus supports will depend on trace length, capacitive load per Device, and the types of Devices (I^2C vs. I3C) present on the Bus, because these factors affect clock frequency requirements.

Note:

*Previous versions of the I3C Basic Specification listed a maximum number of 11 I3C Basic Target Devices. This limit was calculated based on typical electrical parameters (see **Section 4.3.1**).*

An I3C Target Device listens to the I3C Bus for relevant I3C Commands (CCCs) sent by the Active Controller and responds accordingly. This includes all Broadcast Commands (CCC), and any Directed Commands (CCC) addressed specifically to and supported by that I3C Target Device.

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

An I3C Target Device does not generate the Bus clock, and always follows the Bus clock generated by the Active Controller, except during:

- Read transfers in HDR-BT Mode (optional, at implementer discretion)

For addressing, an I3C Target Device supports at least one of the Dynamic Address Assignment methods: based on SETDASA, SETAASA, or ENTDAA.

In addition, an I3C Target Device can optionally:

- Request In-Band Interrupts
- Generate Hot-Join events
- Request to become Active Controller (If Target device is a Secondary Controller)
- Support any combination of I3C's defined HDR Modes

While functioning as a Target, an I3C Target Device functions in one of the Target Device Roles detailed in ***Section 3.2.2.1***.

Figure 9 is a block diagram of a typical generic I3C Target Device.

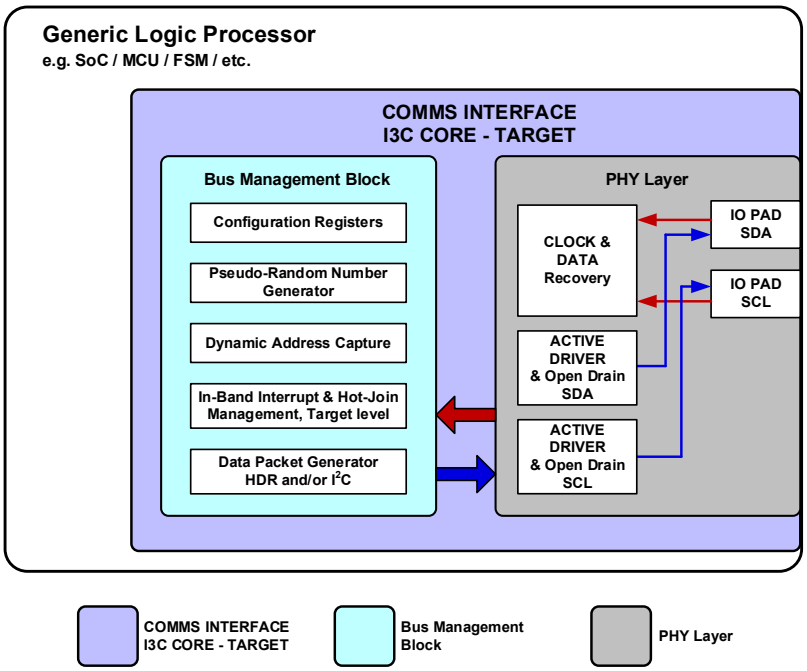


Figure 9 I3C Target Device Block Diagram

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3.2.2.1 I3C Target Device Roles

All I3C Target Devices support one of the two I3C Target Device Roles:

- **I3C Target:** An ordinary I3C Target Device without Controller capability. Supports both SDR Mode and at least one HDR Mode.
- **SDR-Only I3C Target:** An I3C Target without Controller capability that only supports I3C's SDR Mode (i.e., does not support any of the HDR Modes).

Note:

An additional Target Device Role is defined for I^2C Target, however this is not relevant for an I3C Basic Target Device.

See also ***Table 2*** and ***Table 4***.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

REQUIRED ELEMENTS

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4 Required Elements

Every I3C Basic Device shall implement all features and functionality specified in ***Section 4***.

All features and functionality specified in ***Section 4*** are required for I3C Basic.

Table 3 Required Elements Status

Required Elements last updated in I3C Specification Version	I3C v1.2
Required Elements is included in I3C Basic	Yes

Changes in this Version

- See ***Annex A***.

4.1 Scope and Purpose (Informative)

Every functioning I3C Basic Device must implement everything specified in ***Section 4***. Optional elements are out of scope for ***Section 4***.

4.2 Technical Overview (Informative)

This Section specifies the communication protocols for Single Data Rate (SDR) Mode and required tolerances for High Data Rate (HDR) Mode.

SDR Mode is the default Mode of the I3C Basic Bus. SDR Mode is used to enter other Modes, sub-Modes, and states; and for built-in features such as CCCs, IBIs, and transitions from I^2C to I3C Basic by assignment of a Dynamic Address.

I3C SDR Mode is significantly similar to the I^2C protocol ***[NXP01]*** in terms of procedures and conditions. As a result, I3C Basic Devices and many Legacy I^2C Target Devices (not Legacy I^2C Controller Devices) can coexist on the same I3C Basic Bus. However, SDR Mode also includes numerous new features not present in I^2C. The I3C Basic protocol is designed to allow I^2C traffic to be properly ignored by all I3C Basic Targets. I3C Basic traffic from an I3C Basic Controller to an I3C Basic Target will not be seen by most Legacy I^2C Target Devices because the I^2C Spike Filter is opaque to I3C Basic's higher clock speed.

HDR Mode is an optional Mode that allows for data speeds higher than SDR Mode. Although use of HDR modes is optional, handling of entry and exit to these modes is required for all I3C Basic Devices. This requirement allows for HDR capable devices and SDR devices to operate on the same Bus. After HDR Mode entry, an SDR-Mode-only device will ignore all messages (HDR Mode messages) until an HDR Exit is completed and the Bus returns to SDR Mode.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3 Technical Specification

This Section specifies all the core elements of an I3C Basic Device. It is important to note that the I3C Basic Bus is always initialized and configured in SDR Mode.

4.3.1 Bus Configuration

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

An example block diagram of I3C Basic interconnections is shown in ***Figure 10***. In this figure the color blue indicates Devices with a Controller Role, the color pink indicates Devices with an I3C Basic Target role, and the color purple indicates Devices with an I^{2}C Target role.

Note:

*I3C Secondary Controller Devices have the ability to function in either Controller or Target roles at different times, using the same Dynamic Address. In **Figure 10**, the I3C Primary Controller is the Active Controller of the Bus, so the I3C Secondary Controller acts as a Target (i.e., its Controller capability is not active).*

*I3C Basic Devices may also present multiple simultaneous roles. In **Figure 10**, a typical I3C Basic composite Device is shown that presents multiple Virtual Targets on the I3C Basic Bus, where each Virtual Target has a separate Dynamic Address. Such a composite I3C Basic Device uses shared Peripheral logic to handle transfers for its Virtual Targets, which are active simultaneously, per **Section 4.3.2.1.2**. Other types of composite I3C Basic Devices are possible, and might include support for other roles (i.e., Secondary Controller).*

I3C Basic Devices may also enable connections to other buses, either as a Bridge Device (i.e., to a different bus such as I^{2}C or SPI) or as a Routing Device (to another I3C Basic Bus). Such Devices might present one or more Virtual Targets on this I3C Basic Bus that may be used to communicate through the Bridge Device or Routing Device.

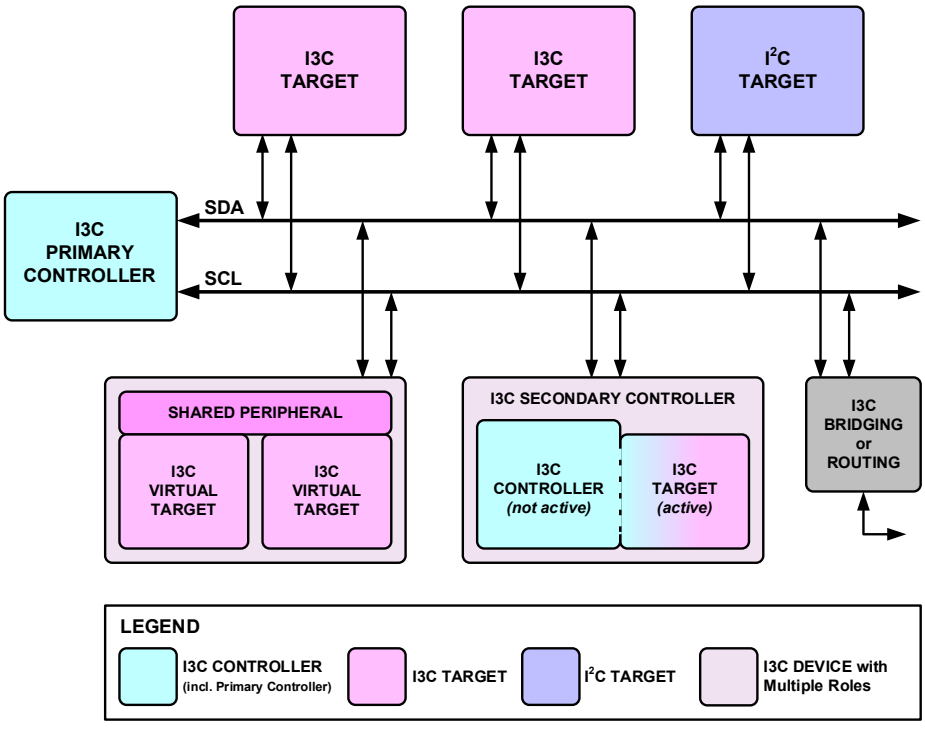


Figure 10 I3C Basic Bus with I^{2}C Devices and I3C Basic Devices

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Figure 11 shows another example block diagram, with an I3C Basic Minimal Bus use case for simple point-to-point communications. In this use case a single Controller assigns the same Dynamic Address to one or more I3C Basic Target Devices.

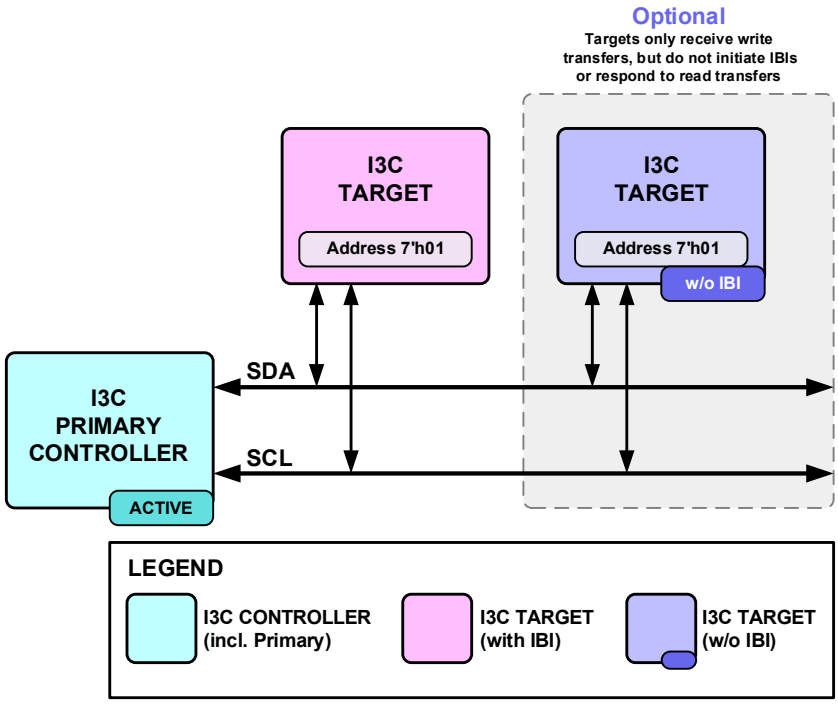


Figure 11 I3C Basic Minimal Bus with I3C Basic Target Devices

I3C Basic compatible Devices may have diverse features, as appropriate for their function within the I3C Basic Bus. Depending on the I3C Basic Bus's system design, it might not be necessary for all features of a given Device to be enabled for any particular Bus instantiation. However, the enabled features of every I3C Basic compatible Device shall be described in Characteristics Registers associated with the Device and its Roles, as described in ***Section 4.3.1.2***. The I3C Basic Primary Controller shall obtain the Characteristics of any Legacy I^2C Devices on the I3C Basic Bus before power-up (e.g., the fixed Address of each Legacy I^2C Device present on the Bus).

At every start-up from a powered-down state, the Primary Controller shall assign a unique Dynamic Address to every Device presenting an active role on the Bus, including itself. The Dynamic Address assignment procedure is described in ***Section 4.3.4***. Dynamic Addresses create a priority ranking of In-Band Interrupts. Any Secondary Controllers present on the I3C Basic Bus shall be made aware of the Dynamic Address assignments and Characteristic Registers associated with each I3C Basic compatible Device on the Bus via Common Command Codes as described in ***Section 4.3.7***.

4.3.1.1 I3C Basic Device Characteristics

The configuration of an I3C Basic Bus will depend upon the Characteristics of the I3C Basic Devices intended to be active on that I3C Basic Bus. Therefore, an active I3C Basic Device playing a given Role in a given I3C Basic Bus instantiation shall fulfill all responsibilities for that Role, as detailed in ***Table 4***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

716

Table 4 I3C Basic Devices Roles vs Responsibilities

Responsibilities / Features	Comments	Roles					
		Primary Controller	Secondary Controller	SDR-Only Primary Controller	SDR-Only Secondary Controller	Target	SDR-Only Target
Manages SDA Arbitration	For Address Arbitration, In-Band Interrupt, Hot-Join, Dynamic Address, as appropriate	Y	Y	Y	Y	N	N
Dynamic Address Assignment	Controller assigns Dynamic Address	Y	Optional[3]	Y	Optional[3]	N	N
Hot-Join Dynamic Address Assignment	Controller capable of Dynamic Address assignment after Hot-Join	Y	Optional[3]	Y	Optional[3]	N	N
Self Dynamic Address Assignment	Only Primary Controller can self-assign a Dynamic Address	Y	N	Y	N	N	N
Static I^2C Address[1]	–	N/A	Optional	N/A	Optional	Optional	Optional
Memory for Targets' Addresses and Characteristics	Retaining registers	Y	Y	Y	Y	N	N
HDR Target capable	Supports being accessed in at least one HDR Mode	Y	Y	N	N	Y	N
HDR Controller capable	Supports Bus control in at least one HDR Mode	Y	Y	N	N	N/A	N/A
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 4.3.4**.*

2) All Targets require an HDR Exit Pattern Detector, even Targets that are not HDR capable.

*3) A Secondary Controller may assign a Dynamic Address using the ENTDAA CCC while it is the Active Controller (i.e., when it holds the Controller Role, per **Section 6.5.3.3.3**). Some Secondary Controllers might have limited capabilities or reduced functionality while acting as the Active Controller of the I3C Bus. See **Section 6.5.3.3.4** and **Section 6.5.3.3.6**. However, only the Primary Controller may assign Dynamic Addresses using the SETDASA and/or SETAASA CCCs (i.e., during Bus Initialization), per **Section 4.3.4.2**.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

The I3C Basic protocol supports a subset of I²C Target features. For example, an I3C Basic Target can have a Static Address but also support Dynamic Addressing. A Device shall not have a 50 ns Spike Filter enabled when used in an I3C Basic Bus operating at full clock speed. These differences are summarized in ***Table 5***. When used in an I3C Basic system, I3C Basic Targets shall enable or disable appropriate I²C features as shown in ***Table 5***.

Table 5 I²C Features Allowed in I3C Basic Targets

I²C Feature When Used on an I3C Basic Bus	Required on I3C Basic	Desirable on I3C Basic	Not Used on I3C Basic	Not Allowed on I3C Basic	Note
Fm/Fm+ Speed	X	–	–	–	2, 3, 4
HS Speed	–	–	X	–	3
UFm Speed	–	–	X	–	3
Static I²C Address	–	X	–	–	–
50 ns Spike Filter	–	–	X	X (Shall disable)	3
Clock Stretch	–	–	–	X	–
20 mA Open Drain Driver	–	–	X	–	1, 3
Matches I²C AC Timing	–	–	X	–	2, 3
I²C Extended Address (10 bit)	–	–	X	–	3
I3C Reserved Addresses	–	–	–	X	5

Note:

1) *See **Table 46** and **Table 48***
2) *I3C Basic drive and timing requirements are different from I²C.*
3) *If an I3C Basic Target has I²C features intended for use on an I²C Bus, then they will not be used on an I3C Bus. As stated in **Section 4.3.2.1.1**, once the Target sees a 7'h7E, it will disable I²C features that are not used by I3C.*
4) *For a Mixed Bus, timing requirements when communicating with I²C-only Devices may depend on the maximum SCL clock frequency supported by such I²C-only Devices. See **Table 47** and **Table 48**.*
5) *I²C Targets are not allowed to use any of the addresses marked as reserved in the I3C Specification as a Static Address.*

Performance of an I3C Basic Bus is heavily dependent upon any I²C-only Devices that may be connected to that Bus. Consequently, all I²C-only Devices permitted on any instantiation of an I3C Bus must be compliant with one of the categories detailed in ***Table 6***. Furthermore (and as referenced in ***Table 47***), no I²C or I3C Basic Device present on an I3C Basic Bus shall have an I²C Static Address that matches any of the Addresses associated with Error Type TE0 (see ***Table 43***).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

728 **Table 6 Legacy I²C-Only Target Categories and Characteristics**

Index Specific	I²C-Only Devices Index 0	I²C-Only Devices Index 1	I²C-Only Devices Index 2
50 ns 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})

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.1.2 I3C Characteristics Registers

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

There are three Characteristics Register types:

- **Bus Characteristics Register** (BCR, see ***Section 4.3.1.2.1***)
- **Device Characteristics Register** (DCR, see ***Section 4.3.1.2.2***)
- **Legacy Virtual Register** (LVR, see ***Section 4.3.1.2.3***)

Every I3C compatible Device shall have associated Characteristics Registers, depending on the Device type:

- Every I3C Basic compliant Device (as defined in ***Table 4***) that holds one active role shall have one Bus Characteristics Register, and one Device Characteristics Register; these are associated with the Dynamic Address.
- I3C Basic compliant Devices that hold multiple active roles might use a single shared Bus Characteristics Register and Device Characteristics Register that applies to all such active roles; or one pair of such registers for each active role, associated with each role's Dynamic Address.
  - Composite I3C Basic Devices that present multiple Virtual Targets should usually have separate Bus Characteristics Registers and Device Characteristics Registers, i.e., one set for each Virtual Target, associated with its Dynamic Address.
- Every Legacy I^2C Device to be connected to an I3C Basic Bus shall have one associated Legacy Virtual Register. Since these are Legacy Devices, it is understood that this register will exist virtually, for example as part of the Device's driver.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.1.2.1 Bus Characteristics Register (BCR)

Each I3C Basic Device that is connected to the I3C Basic Bus shall have an associated read-only Bus Characteristics Register (BCR). This read-only register describes the I3C Basic compliant Device's role and capabilities for use in Dynamic Address assignment and Common Command Codes. The bits within the BCR shall conform to the Descriptions presented in in ***Table 7***.

Table 7 Bus Characteristics Register (BCR)

Bit	Name	Description	Notes
BCR[7]	**Device Role[1]**	**2'b00:** I3C Basic Target **2'b01:** I3C Basic Controller-capable **2'b10:** Reserved for future definition by MIPI Alliance I3C WG **2'b11:** Reserved for future definition by MIPI Alliance I3C WG	1
BCR[6]	**Device Role[0]**		
BCR[5]	**Advanced Capabilities**	**1:** Supports optional advanced capabilities. Use GETCAPS CCC (***Section 4.3.7.3.19***) to determine which ones. **0:** Does not support optional advanced capabilities	2
BCR[4]	**Virtual Target Support**	**0:** Is not a Virtual Target and does not expose other downstream Device(s) **1:** Is a Virtual Target, or exposes other downstream Device(s)	3
BCR[3]	**Offline Capable**	**0:** Device will always respond to I3C Basic Bus commands **1:** Device will not always respond to I3C Basic Bus commands	4
BCR[2]	**IBI Payload**	**0:** No data bytes follow the accepted IBI **1:** One data byte (MDB) shall follow the accepted IBI, and additional data bytes may follow; see also the Set/Get Maximum Read Length CCC (***Section 4.3.7.3.6***). Data byte continuation is indicated by T-Bit per ***Section 4.3.2.3.4***. See also ***Section 6.6*** on use of IBI Payloads for Timing Control.	–
BCR[1]	**IBI Request Capable**	**0:** Not Capable **1:** Capable	–
BCR[0]	**Max Data Speed Limitation**	**0:** No Limitation **1:** Limitation	5

Note:

1) The BCR Device Role bits for any I3C Basic Device capable of acting as I3C Basic Controller (either Primary Controller or Secondary Controller) are 2'b01.

2) In I3C v1.0, bit BCR[5] only indicated HDR support, and that the Controller should use the GETHDRCAPS CCC to determine what HDR Modes were supported. In I3C v1.1 and above, bit BCR[5] now indicates that the Controller should use the same CCC (now renamed GETCAPS) to determine what advanced features the I3C Device supports, including (but no longer limited to) HDR. Note that the I3C v1.0 method is fully interoperable with the I3C v1.1 and above method.

*3). In I3C Basic v1.0, bit BCR[4] only indicated Bridge Devices required to comply with the I3C Basic Specification. In I3C Basic v1.1 and above, bit BCR[4] now indicates that the Device is a Virtual Target or exposes other downstream Devices, and that the Controller should use the GETCAPS CCC with Defining Byte VTCAPS (**Section 4.3.7.3.19**) to determine what features and capabilities the I3C Device supports, including (but no longer limited to) Bridge Devices.*

*4) Offline Capable Devices retain the Dynamic Address, and are specified in **Section 2.3.2**.*

5) Controller shall use the GETMXDS CCC to interrogate the Target for specific limitation.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.1.2.2 Device Characteristics Register (DCR)

Each I3C Basic 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 Basic 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 8***.

MIPI Alliance maintains a public registry of approved and pending DCR value assignments ***[MIPI02]***.

Table 8 I3C Device Characteristics Register (DCR)

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

4.3.1.2.3 Legacy Virtual Register (LVR)

Each Legacy I^2C Device that can be connected to the I3C Basic Bus shall have an associated read-only Legacy Virtual Register (LVR) describing the Device's significant features. Since these are Legacy I^2C Devices, it is understood that this register will exist virtually, for example as part of the Device's driver. When Legacy I^2C Devices are present on an I3C Basic Bus, LVR data determines allowed Modes and maximum SCL clock frequency. The bits within the LVR shall conform to the descriptions in ***Table 9***.

All LVRs shall be established by the higher-level entity controlling the I3C Basic Bus, and shall be transferred to the I3C Basic Bus Primary Controller prior to Bus configuration. The LVR content for all I^2C Devices is always known by the Primary Controller. The Primary Controller shall transfer the LVR content for all I^2C Devices to Secondary Controllers by using the DEFTGTS CCC (see ***Section 4.3.7.3.7***).

Table 9 Legacy I^2C Virtual Register (LVR)

Bits	Name	Values
LVR[7:5]	**Legacy I^2C-Only [2:0]** per ***Table 6***	**3'b000:** Index 0: Spike Filter, Max SCL clock freq tolerant N/A **3'b001:** Index 1: No Spike Filter, Max SCL clock freq tolerant **3'b010:** Index 2: No Spike Filter, Not Max SCL clock freq tolerant
		3'b011 – 3'b111: Index 3 – 7: Reserved
LVR[4]	**I^2C Mode Indicator**	**0:** I^2C Fm+ **1:** I^2C Fm
LVR[3:0]	MIPI Alliance I3C WG Reserved	**0:** Reserved
		1 – 15: 15 available codes for describing the Device capabilities and function on the sensors' system.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.2 Bus Communication

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

- The I3C Basic START and STOP conditions (as shown in ***Figure 81*** and ***Figure 84***, respectively) are identical to the I²C START and STOP in their signaling, but they may vary from I²C in their timing. Compare ***Table 49*** vs. ***Table 48***.

  Note:

  In I3C Basic SDR (but not I3C HDR), a STOP or Repeated START is tolerated (i.e., may appear) any time that SCL is High while the Controller controls SDA or SDA is Open-Drain. This is unlike I²C, which wants STOP or Repeated START only after a NACK of an address, or after ACK/NACK of data. Although I3C Basic SDR also prefers STOP and Repeated START to be used only in those situations, and after an I3C Broadcast Address is ACKed, it does not disallow them in any other location. However, appearance of a STOP or Repeated START in the middle of an Address or in the middle of data is interpreted to cancel that Address or data.

- The I3C Basic Address Header is identical to the I²C Address Header in bit form and in signaling, but it may vary from I²C in its timing. See ***Section 4.3.2.2***, ***Table 49***, and ***Table 50***.
- The Data 9-bit Words use the same bit count as I²C, but differ in the ninth bit, as explained in ***Section 4.3.2.3***.
- In I3C, the SCL line is only driven by the Controller. Normally this drive is Push-Pull, but it can also be Open Drain.

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

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

An I3C Basic Message is an SDR Message if:

- The Address in the Address Header is 7'h7E (the I3C Broadcast Address). All I3C Targets shall match Address value 7'h7E. No I²C Target will match Address 7'h7E, because that value is reserved and unused in I²C.
- The Address in the Address Header matches the Target's Dynamic Address (as assigned by the I3C Controller per ***Section 4.3.4***). All I3C Basic Targets shall match their own Dynamic Address. (A Target may decide to NACK its own Address if needed.)

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

Note:

*Legacy I²C Targets will ignore any Message not addressed to them, and will await the next START or STOP. Per **Section 4.3.2.4**, Legacy I²C Targets may also not see some or all I3C Basic Messages and Modes due to the speed of SCL signaling.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

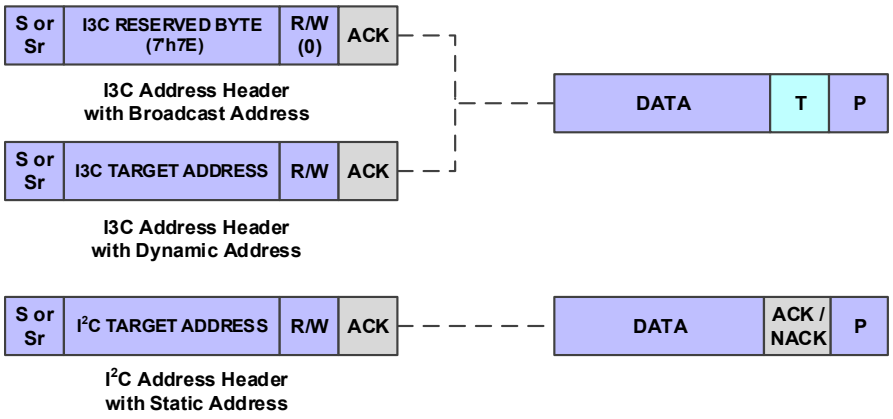


Figure 12 Address Header Comparison

4.3.2.1 Role of I3C Basic Target

An I3C Basic Target is a functional role that is embodied or presented by an I3C Basic Device. Such a Device may be a standalone physical Device that presents only one Target, or it may be a composite Device that presents multiple Virtual Targets using shared Peripheral logic. In either case, each Target (which may be a Virtual Target) presented on the I3C Basic Bus may be assigned its own unique Dynamic Address, and the Controller can address it independently for transfers.

Note:

*Many such I3C Basic Device implementations are possible that might embody or present one or more active I3C Basic Target roles. For composite I3C Basic Devices that present multiple Virtual Targets, the shared Peripheral logic might handle many of the underlying functions that would normally be handled by internal logic for a single, standalone I3C Basic Device that only presented a single Target on the I3C Basic Bus. For simplicity, the remainder of this Section treats both options equally, referring to "Target" in a generic sense. See **Section 4.3.2.1.2** for special exceptions.*

The I3C Basic Target does not have to know whether it is on a Legacy I²C Bus or an I3C Basic Bus. If it has an I²C Static Address, then it may participate as an I²C Device using that Address (per ***Section 4.3.2.1.1***) up until it is assigned a Dynamic Address. Once assigned a Dynamic Address, unless asked to Reset, it shall only operate as an I3C Basic Target.

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

Before Receiving a Dynamic Address

Targets that do recognize START and the 7'h7E Address may see any CCC, not just ENTDAA (see ***Section 4.3.7.3.4***), and shall behave as follows:

- The Target shall appropriately process all required Broadcast CCCs, including ENTDAA, RSTDAA (see ***Section 4.3.7.3.3***), ENEC (see ***Section 4.3.7.3.1***), and DISEC (see ***Section 4.3.7.3.1***).

  Examples of appropriate processing:

  - RSTDAA has no effect, since no Dynamic Address is assigned
  - For a Target Device intended to be used on I3C Buses that only use SETAASA and/or SETDASA to assign a Dynamic Address to it, ENTDAA support is optional.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

- If the Target would never issue a Controller Role Request, then the DISEC CCC for the Controller Role Request can be ignored.

- The Target shall recognize the CCCs ENTHDR0 through ENTHDR7 (see ***Section 4.3.7.3.9***), and then wait for the HDR Exit Pattern.
- The Target may choose to understand and process the SETDASA CCC (see ***Section 4.3.7.3.10***) when its Static Address matches.
- The Target may choose to understand and process the SETAASA CCC (see ***Section 4.3.7.3.21***).
- The Target shall disregard all Directed CCC commands, but shall properly recognize the ends of Directed CCCs (i.e., either Repeated START followed by 7'h7E, or STOP).
- When no Dynamic Address has been assigned yet, the Target may either support or ignore non-Required and Conditionally Required Broadcast CCCs.

  This is true even if the Target supports any of these CCCs after being assigned a Dynamic Address. For example, the Target may choose to only support Test Mode only when no Dynamic Address is assigned.
- The I3C Target shall ignore TE0 type errors related to incorrect Addresses only (see ***Table 43***).

Address Header Matching

The role of an I3C Target shall be as follows:

1. Following a START or a Repeated START, at any speed conforming to ***Section 4.3.1*** of this Specification, the I3C Target shall attempt to match the Address to the I3C Broadcast Address (7'h7E) or to its own Dynamic Address, once assigned.

   If the Target also supports Group Address capabilities (per ***Section 4.3.4.4***), then it shall also attempt to match the Address to any of its assigned Group Addresses.

   If any match is found, then the I3C Target shall treat that Message as I3C Basic SDR.
2. If the Message is addressed to the Target's Dynamic Address, then the Target may ACK or passively NACK the Address Header:
   A. If the Target ACKs the Address Header, then the Target shall process the Message as I3C Basic SDR, following all rules as outlined in this Section.
   B. If the Target NACKs the Address Header (does not drive the ACK bit Low), then the Target should disregard any bits that follow, up until the next Repeated START or STOP.
3. If the Message is addressed to any of the Target's assigned Group Addresses, with a Write (RnW bit is 0), then the Target may ACK or passively NACK the Address Header:
   A. If the Target ACKs the Address Header, then the Target shall process the Message, as described above (i.e., in the same manner as an I3C Basic SDR Write to its Dynamic Address).
4. If the Message is addressed to the I3C Broadcast Address (7'h7E), with a Write (RnW bit is 0), then the Target shall process that Message at least through the first byte of data (if any data is present in the Message):
   A. If there is a byte of data in a 7'h7E Broadcast Message, then the Message is a CCC Command, per ***Section 4.3.7***.
   B. The I3C Basic Target shall process all applicable Required CCC commands, per ***Section 4.3.7.3***. A command may be either Always Required, or only Conditionally Required, per ***Table 16***.

   For Required CCCs: The Target shall always remain ready to respond to certain Direct CCCs sent to its Dynamic Address, such as **Get Device Status** (GETSTATUS, see ***Section 4.3.7.3.15***) even if the Target (or its inner system) is processing previously sent Messages, and might not be able to process or ACK other CCCs or Messages. (See ***Section 4.3.8.2.5***)

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

C. If the CCC command changes the Mode of the I3C Basic Bus, then the I3C Target shall handle the new Mode in one of two ways.

Either:

i. If the new Mode is Dynamic Address Assignment Mode (see ***Section 4.3.4***), and required for all I3C Basic Targets, then the Target shall participate if it does not have a current Dynamic Address; otherwise, the Target shall await the STOP that indicates exit from Dynamic Address Assignment Mode.

Or:

ii. If the new Mode is HDR (High Data Rate) Mode, then the Target may either enter into HDR Mode if supported (per ***Section 6.1.4.1***), or else enable its HDR Exit Pattern detector (per ***Section 4.3.10.2*** and ***Section 4.3.10.2.1***) to await the exit from HDR Mode.

5. If the Message is not addressed to the I3C Broadcast Address (7'h7E) or to any of the Target's currently assigned Addresses (i.e., Dynamic Address or Group Address), then the I3C Basic Target shall await either a Repeated START or a STOP. The Target may record/monitor the bits as they pass (if desired), but the only obligation is to wait for either a Repeated START or a STOP:

Note:

*Repeated START and STOP are specified in **Section 2.3.2** and **Section 4.3.11.2**, and are similar to I²C. In both cases, I3C Basic timing might not be the same as with I²C.*

4.3.2.1.1 Role of I3C Basic Target Acting as an I²C Target with Static Address

As noted in other Sections, I3C Basic Targets are capable of acting as standard I²C Targets as long as they have an I²C Static Address.

There are two situations:

1. They are on a Legacy I²C Bus, and so all messaging is coming from a Legacy I²C Controller.
2. They are on an I3C Basic Bus, but the I3C Basic Controller is not assigning them a Dynamic Address, so they continue to operate as an I²C Target; Messages to/from its I²C Static Address use the I²C protocol.

For operation on an I²C Bus, the Target may use an I²C Fm/Fm+ Spike Filter of 50 ns (or more).

For operation on an I3C Basic Bus, the Target may have an I²C Fm/Fm+ Spike Filter of 50 ns (or more), which it shall disable once it sees a Message from an I3C Basic Controller, notably the first I3C Basic Address Header (i.e., a START followed by 7'h7E and a RnW bit of 0; see ***Section 4.3.2.2***). To support the disabling of the Spike Filter, the I3C Basic Controller shall emit at least the first such I3C Basic Address Header with FM/FM+ timing parameters (per ***Table 49***) such that the High and Low periods are long enough to be seen with the Spike Filter enabled (see ***Section 4.3.2.2.2***).

Note:

If the Target has such a Spike Filter enabled, and sees the Controller send such an I3C Basic Address Header with FM/FM+ timing parameters, then it shall disable the Spike Filter after the ACK (i.e., after acknowledging the I3C Broadcast Address).

An I3C Basic Target acting as an I²C Target will operate correctly on an I3C Basic Bus with the Spike Filter disabled, or without ever having a Spike Filter, because of two strict requirements on all conforming I3C Basic Targets:

1. An I3C Basic Target shall recognize all required Broadcast CCCs and act appropriately for each one, even if the Target has not yet received a Dynamic Address. This notably includes the ENTHDRn CCCs (see ***Section 4.3.7.2*** and ***Table 16***).
2. An I3C Basic Target shall recognize the HDR Exit Pattern (see ***Section 4.3.10.2***).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

As a result of these two requirements, it is safe for an I3C Basic Target acting as an I^2C Target to see the short SCL High periods (unlike a Legacy I^2C Device). As a result, it can operate correctly as an I^2C Target on an I3C Basic Bus without a Spike Filter (unlike Legacy I^2C Targets).

4.3.2.1.2 Composite Devices and Virtual Targets

As noted above, a Virtual Target is presented by an I3C Basic Device that also presents other such Virtual Targets on the I3C Basic Bus, and must manage the communications for all such Virtual Targets. In this manner, a Virtual Target can be considered a function of such a composite I3C Basic Device that holds multiple active Target roles simultaneously. The I3C Basic Device shall maintain separate configuration for each Virtual Target, including the assigned Dynamic Address.

Note:

For additional information on Virtual Targets, refer to the MIPI Alliance I3C Application Note: Virtual Devices and Virtual Targets ***[MIPI10]****.*

During Dynamic Address Assignment (see ***Section 4.3.4.2***) the shared Peripheral logic manages the interaction with the I3C Basic Controller and other I3C Basic Targets on the Bus, for the ENTDAA procedure. This includes managing Address Arbitration while providing the 48-bit Provisioned ID, and ensuring that all active Virtual Targets are able to receive Dynamic Addresses. The shared Peripheral logic also initiates the Hot-Join Request (per ***Section 4.3.5***) during Bus Initialization or whenever necessary.

Note:

A Device that presents multiple Virtual Targets might also support the SETDASA and/or SETAASA CCCs, instead of the ENTDAA CCC. If so, then each Virtual Target must have a unique Static Address.

Once Dynamic Addresses are assigned, the shared Peripheral logic typically handles many of the underlying communications with the I3C Basic Bus in SDR Mode, while each Virtual Target receives data for Private Write transfers addressed to its Dynamic Address, and also provides data to the shared Peripheral logic for responding to Private Read transfers and raising In-Band Interrupt requests.

The shared Peripheral logic also handles the signaling and framing for HDR Modes (if supported, per ***Section 4.3.10***) as well as participation in Broadcast and Direct CCC flows, both in SDR Mode (per ***Section 4.3.7.1***) as well as any HDR Modes in which CCCs might be supported (per ***Section 6.1.3***).

The shared Peripheral logic also handles the reset flows involving the RSTACT CCC with the Target Reset Pattern (see ***Section 4.3.9***), and generates appropriate internal reset messaging to Virtual Target functions based on the type of reset. In most cases, a Device that reports as Virtual Target capable should indicate this via Bit[4] of the BCR (per ***Section 4.3.1.2.1***) and also report additional capabilities using the GETCAPS CCC with Defining Byte VTCAPS (see ***Section 4.3.7.3.19***).

Depending on the implementation, certain actions or CCCs sent to one Virtual Target's Dynamic Address may cause side effects for other Virtual Targets presented by the same composite Device, or within the shared Peripheral logic itself. In some cases, certain features and capabilities that are reported by Virtual Target's Dynamic Address (e.g., by the GETCAPS CCC) shall actually apply equally to the entire Device (i.e., to all Virtual Targets and the shared Peripheral logic).

4.3.2.1.3 Targets Supporting Group Addresses

If a Target supports Group Addressing (per ***Section 4.3.4.4***), then it will have a single set of capabilities or other configurable parameters that pertain to either the Target, or, in special cases, the Device or its Peripheral logic.

- If such capabilities are accessible via a Direct GET CCC (e.g., the GETCAPS CCC, ***Section 4.3.7.3.19***) then these shall be assumed to apply to the Target and all its assigned Addresses, including any assigned Group Addresses, unless specified otherwise in the particular Direct CCC's definition.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

- If such configuration can be set via a Direct SET CCC (per ***Section 4.3.7.3***) then these shall also apply to the whole Target, with exceptions and limitations noted in ***Section 4.3.7.4***.

For such capabilities or other configurable parameters, the Target should accept (i.e., should ACK) such Direct SET CCCs that might be sent to any assigned Dynamic Addresses or assigned Group Addresses, unless the Direct CCC format is not supported for a Group Address, or a particular Direct CCC defines a different use or format that applies when sent to an assigned Group Address (e.g., the MLANE CCC, see ***Section 6.7.3.6***).

4.3.2.2 I3C Basic Address Header

The I3C Basic Address Header follows either a START, or a Repeated START. The format is the same as I^2C: 7 bits of Address, 1 bit of RnW, and 1 bit of ACK/NACK.

The Address Header following a START is an Arbitrable Address Header, as explained in ***Section 4.3.2.2.1***. This means the START and at least the first Address bit and ACK/NACK are issued on SDA using Open Drain Bus drive, similar to I^2C. However, some of the Arbitrable Address Header may be driven on SDA using Push-Pull and higher speed (see ***Section 4.3.2.2.2***).

The Address Header following a Repeated START is always driven on SDA using Push-Pull, with the exception of the ACK/NACK (see ***Section 4.3.2.2.4***).

Using the I3C Basic Arbitrable Address Header, I3C Basic Targets may transmit any of three requests to the I3C Basic Controller:

1. **An In-Band Interrupt**, per ***Section 4.3.6***. This is equivalent to toggling a wire to get the Controller's attention. The In-Band Interrupt request shall be made using the Target's Dynamic Address with a RnW bit of 1.
2. **A Controller Role Request**, per ***Section 6.5.3***. An I3C Device shall not make such a request unless it is marked as a Controller-capable Device in its BCR register, per ***Section 4.3.1.2.1***. The Controller Role Request may be made by a Secondary Controller wishing to gain the Controller Role from the Active Controller, or by the Primary Controller after it has relinquished the Controller Role and now wishes to regain it from the Active Controller. The Controller Role Request shall include the Device's Dynamic Address with a RnW bit of 0.
3. **A Hot-Join Request**, per ***Section 4.3.5***. An I3C Basic Target shall only make such a request when becoming available after the I3C Basic Bus is operational. The Hot-Join Request shall be made using the reserved Hot-Join Address (i.e., 7'h02).

The I3C Basic Targets shall make these requests to the I3C Basic Controller in only two Bus conditions:

1. A START (but not a Repeated START) is issued on the Bus following a Bus Free Condition, per ***Section 4.3.3***. The Target may transmit its Dynamic Address or the Hot-Join Address (7'h02) following the START, by adhering to the I3C Basic Address Arbitration rules per ***Section 4.3.2.2.1***.
2. The Bus is in a Bus Available Condition, per ***Section 4.3.3.2.2***, so the Target may issue a START by pulling the SDA Low.
   A. If the Target pulls SDA Low, then the Controller shall pull SCL Low within $\mathbf{t_{CAS}}$ (see ***Table 49***).
   B. The Controller shall also pull SDA Low (overlapping the Target pulling it Low).
   C. Once the Controller has pulled SCL Low, the Target shall control the SDA line in Open Drain mode (i.e., either pull Low, or release High).
   D. The Target may then issue its Address in the normal way (condition 1 above).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.2.2.1 I3C Basic Address Arbitration

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

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

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

   Note:

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

2. If the current bit to transmit is a 1, then the Device shall not drive SDA, but rather shall High-Z SDA on the falling edge of SCL.
   A. Additionally, the Device shall monitor the SDA on the rising edge of SCL to determine whether another Device has driven SDA Low.
   B. If another Device has driven the SDA Low, then the Device has ‘lost’ the Arbitration and shall not further participate in this Address Header. That is, the Device shall not transmit any more bits, but may wait for a future START (but not a Repeated START).

4.3.2.2.2 I3C Basic Address Arbitration Optimization

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

The Controller shall not apply the optimizations described in this Section for the first I3C Basic Address Header (i.e., START followed by 7'h7E and an RnW bit of 0) after Bus initialization, unless the Controller knows that no I3C Basic Target on the Bus has a Spike Filter. This permits any I3C Basic Target with an I^{2}C Spike Filter to detect that it is on an I3I3C Basic C Bus, and as a result disable the Spike Filter. **See Errata 01, Item 5**

As previously described in ***Section 4.3.2.2***, an I3C Basic Controller Device assigns 7-bit Dynamic Addresses with values in the range 7'h03 to 7'h7B. However, because the I3C Basic Controller treats the entire 9-bit Arbitrable Address Header as Open Drain, it has no way to detect whether an I3C Basic Target Device might be transmitting its own Address during some (or all) of the Address Header.

For I3C Secondary Controllers and I3C Basic Targets that can request In-Band Interrupts, the I3C Basic Controller is typically free to restrict assigned Dynamic Addresses to the lower half of the available range (7'h03 to 7'h3F), thus leaving the value of Address bit A6 (the first Address bit after the START) as 0 in all such assigned Dynamic Addresses. For other I3C Basic Targets that cannot request In-Band Interrupts, the I3C Basic Controller restricts assigned Dynamic Addresses to the upper half of the available range (7'h40 to 7'h7B), thus leaving the value of Address bit A6 as 1 in all such assigned Dynamic Addresses.

Note:

*This Dynamic Address Assignment strategy is not always possible, since some I3C Basic Targets might not support arbitrary Dynamic Address values in the available range (i.e., when an I3C Basic Target has an I^{2}C Static Address and also does not support the SETNEWDA CCC; see **Section 4.3.7.3.11**).*

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

- If the I3C Basic Controller is transmitting a 1 value (i.e., High-Z on SDA), as it would when transmitting 7'h7E, then it can monitor SDA on the rising edge of SCL. If SDA has the value 1 (i.e., if SDA is not being driven by any Target), then the I3C Basic Controller may optionally transmit the remainder of the Address Header (up until the ACK/NACK) in Push-Pull mode. See ***Figure 13***, upper waveform.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

- If the I3C Basic Controller is transmitting a 0 value (i.e., is driving SDA Low), or if SDA was driven Low by a Target, then the I3C Basic Controller shall transmit the remainder of the Address Header using Open Drain.
- If the I3C Basic Controller intends to transmit solely to other I3C Basic Devices (i.e., not to any I^2C Devices), then it may optionally maintain the SCL pulse width below 50 ns, with the result that any I^2C Devices present on the Bus will see only a 0 value. This shorter pulse width produces a higher data rate, because for Open Drain Arbitration only the Low time is extended. See ***Figure 13***, middle waveform.
- If the I3C Basic Controller intends to transmit to any I^2C Devices, then it must use the slower I^2C timing. See ***Figure 13***, lower waveform.

The upper waveform of ***Figure 13*** illustrates this optimization. When the I3C Basic Controller sees a value of 1 for Address bit A6, but previously made sure that all Dynamic Addresses assigned to such Targets have a 0 in bit A6, then the I3C Basic Controller knows that the line was not being driven by any such Target (i.e., for an In-Band Interrupt request or a Controller Role Request).

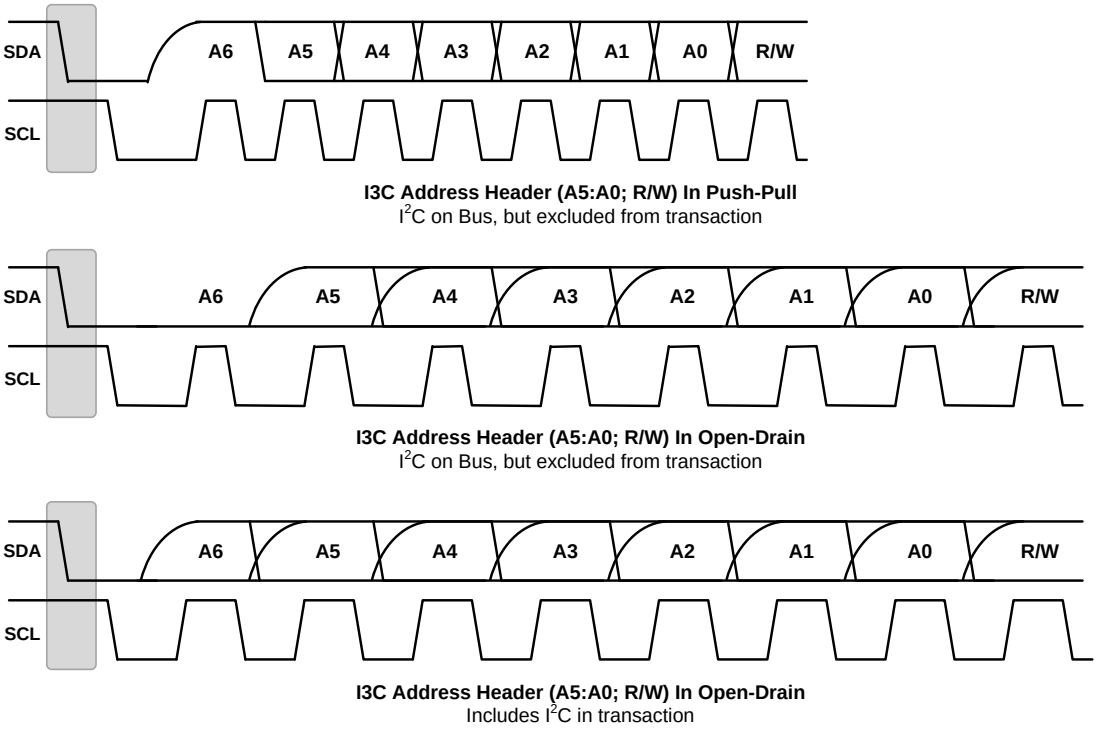


Figure 13 Address Arbitration During Header

Copyright É 2016ï 2025 MIPI Alliance, Inc.
Public Release Edition

4.3.2.2.3 Consequence of Controller Starting a Frame with I3C Basic Target Address

The I3C Basic Controller normally should start a Frame with 7'h7E (for all I3C Basic Messages) or an I^2C Static Address (when sending only to a Legacy I^2C Target).

In both cases the Address may be arbitrated, and so the Controller shall monitor to see whether an In-Band Interrupt request, a Controller Role Request (i.e., Secondary Controller requests to become the Active Controller), or a Hot-Join Request has been made.

- If not, then the Controller may proceed normally.
- If so, then the Controller may ACK or NACK that request and then proceed accordingly.

If the Controller chooses to start an I3C Basic Message with an I3C Basic Dynamic Address, then special provisions shall be made because that same I3C Basic Target may be initiating an IBI or a Controller Role Request. So, one of three things may happen:

1. The Addresses match, but the difference is caught on the RnW bit, and the Controller was initiating a Private Write (i.e., RnW = 0) while the Target was attempting to send an IBI request (i.e., RnW = 1).

   In this case, the Controller wins (i.e., an IBI with RnW = 1 loses), and the Target shall ACK or NACK the Private Write. The Target shall defer its IBI, and may retry at a later opportunity.

2. The Addresses match, but the difference is caught on the RnW bit, and the Controller was initiating a Private Read (i.e., RnW = 1) while the Target was attempting to send a Controller Role Request (i.e., RnW = 0).

   In this case, the Target wins (i.e., a Private Read with RnW = 1 loses), and the Controller must ACK or NACK the Controller Role Request. The Controller shall defer its Private Read, and may retry at a later opportunity.

3. The Addresses match and the RnW bits also match, and so neither Controller nor Target will ACK since both are expecting the other side to provide ACK. As a result, each side might think it had "won" arbitration, but neither side would continue, as each would subsequently see that the other did not provide ACK.
   A. If RnW = 0 (i.e., if the Controller was initiating a Private Write while the Target was attempting to send a Controller Role Request), then the Target shall defer its Controller Role Request, and may retry at a later opportunity.
   B. If RnW = 1 (i.e., if the Controller was initiating a Private Read while the Target was attempting to send an IBI request), then the Target shall defer its IBI request, and may retry at a later opportunity.
   C. For either value of RnW: Due to the NACK, the Controller shall defer the Private Write or Private Read, and should typically transmit the Target Address again after a Repeated START (i.e., the next one or any one prior to a STOP in the Frame). Since the Address Header following a Repeated START is not arbitrated, the Controller will always win (see ***Section 4.3.2.2.4***).

      This allows the Controller to determine whether the Target intended to provide the same RnW bit (i.e., which may have led to this NACK situation), or whether it may have simply chosen not to support this transaction. In this manner, the Controller can avoid a 'live-lock' situation.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.2.2.4 Address Header Following a Repeated START is Push-Pull

The Address transmitted by the I3C Basic Controller following a Repeated START shall not be arbitrated. That is, no I3C Basic Target shall attempt to transmit its own Dynamic Address nor the reserved Hot-Join Address (i.e., 7'h02) 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 I^2C Target. The ACK/NACK bit that follows the RnW bit is always Open Drain to allow the Target to ACK or passively NACK its Address.

4.3.2.2.5 I3C Basic Target Address Restrictions

The I3C Basic Target Address space is dependent on decisions by the Controller. That is, the Controller 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 10***.

- The I3C Basic Controller shall not use any of 7'h00, 7'h01, 7'h02, 7'h7E, 7'h7F. All are reserved for I3C.
- The I3C Basic Controller shall not use any of 7'h3E, 7'h5E, 7'h6E, 7'h76, 7'h7A, 7'h7C, 7'h7F. All are prohibited since I3C Basic Targets will interpret an I3C Basic Address Header with any of these Addresses as a Broadcast Address (7'h7E) with a single-bit error. See Error Type TE0, in particular the Note associated with ***Table 43*** (ñIn the ENTDAA mode, ñ7'h7E / Rò is excluded from the TE0 error definition.ò).
- The I3C Basic Controller may choose to not use 7'h03, marked in I^2C as reserved.
- The I3C Basic Controller shall not use 7'h04, 7'h05, 7'h06, 7'h07 if any Legacy I^2C Devices are present on the Bus that support I^2C ñHigh-Speed Modeò.
- The I3C Basic Controller shall not use 7'h7C or 7'h7D if any Legacy I^2C Devices are present on the Bus that support I^2C Device ID Mode.
- The I3C Basic Controller shall not use 7'h78, 7'h79, 7'h7A, 7'h7B if any Legacy I^2C Devices are present on the Bus that support I^2C Extended Address Mode and have an Extended Address or would be impacted by the Extended Address mechanism.

Copyright É 2016ï 2025 MIPI Alliance, Inc.
Public Release Edition

1139

Table 10 I3C Basic Target Address Restrictions

Target Dynamic Address		Restriction	Description
Binary	Hex		
000 0000	7'h00	Shall not use	I3C Basic Reserved
000 0001	7'h01	Shall not use	I3C Basic Reserved: For use with SETDASA CCC in special Point-to-Point Communication. See ***Section 4.3.7.3.10***.
000 0010	7'h02	Shall not use	I3C Basic Reserved: Hot-Join Address
000 0011	7'h03	Optional	Marked 'Reserved' by I^2C
000 0100	7'h04	Conditional	Available for use only if no Legacy I^2C Devices supporting I^2C "High-Speed Mode" are present on the Bus
000 0101	7'h05		
000 0110	7'h06		
000 0111	7'h07		
000 1000 011 1101	7'h08 – 7'h3D	Available for use	54 Addresses
011 1110	7'h3E	Shall not use	I3C Basic 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 Basic 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 Basic Reserved: Broadcast Address single bit error detect
110 1111 111 0101	7'h6F – 7vh75	Available for use	7 Addresses **See Errata 01, Item 6**
111 0110	7'h76	Shall not use	I3C Basic 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 I^2C Devices are present on the Bus that both a) support I^2C "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 Basic Reserved: Broadcast Address single bit error detect
111 1011	7'h7B	Conditional	Available for use only if no Legacy I^2C Devices are present on the Bus that both a) support I^2C "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 Basic Reserved: Broadcast Address single bit error detect (Also not available for use if any Legacy I^2C Devices supporting I^2C "Device ID Mode" are present on the Bus.)
111 1101	7'h7D	Conditional	Available for use only if no Legacy I^2C Devices supporting I^2C "Device ID Mode" are on the Bus
111 1110	7'h7E	Shall not use	I3C Basic Reserved: Broadcast Address single bit error detect (See Error Type TE0, ***Section 4.3.8.1.1***.)
111 1111	7'h7F	Shall not use	I3C Basic Reserved: Broadcast Address single bit error detect

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.2.3 I3C Basic SDR Data Words

In I3C Basic SDR, the Data Words match I²C only in the sense that they are both 9 bits long. I3C Basic SDR Data Words differ from I²C in three ways, as detailed in ***Sub-Sections 4.3.2.3.1***, ***4.3.2.3.3***, and ***4.3.2.3.4***.

In summary:

1. **Handoff from Address ACK to SDR Controller Write Data:** When performing an SDR Write, the handoff from the Target's Address Header ACK to the Controller's first Data bit is different in I3C Basic. I²C is Open Drain, so overlap from the ACK Low into the first bit is harmless. By contrast, I3C Basic is Push-Pull and so this handoff is specified (see ***Section 4.3.2.3.1***).
2. **Ninth Bit of SDR Controller Written Data as Parity:** In I²C, the ninth Data bit written by the Controller is an ACK by the Target. By contrast, in I3C Basic the ninth Data bit written by the Controller is the Parity of the preceding eight Data bits. Therefore, in I3C Basic the Target shall not drive the SDA line for Data written by the Controller in SDR. In SDR terms, the ninth bit of Write data is referred to as the T-Bit (for 'Transition') (see ***Section 4.3.2.3.3***).
3. **Ninth Bit of SDR Target Returned (Read) Data as End-of-Data:** In I²C, the ninth Data bit from Target to Controller is an ACK by the Controller. By contrast, in I3C Basic this bit allows the Target to end a Read, and allows the Controller to Abort a Read. In SDR terms, the ninth bit of Read data is referred to as the T-Bit (for 'Transition') (see ***Section 4.3.2.3.4***).

Note:

A Target should have an SDA Read detector that determines if the SCL clock has not changed for 150 µs or more, so that it can abandon the read by switching SDA to High-Z and waiting for Repeated START or STOP.

4.3.2.3.1 Transition from Address ACK to SDR Controller Write Data

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

- If 7'h7E, then it is the ACK of all I3C Basic Targets on the Bus.
- If a single Target Address, then it is the ACK (or NACK) of the addressed Target, or a NACK if no such Target is on the Bus.

When the Address Header results in an ACK, and the Message is SDR Write from Controller, the SDA line has to be turned from Open Drain to Push-Pull for the first data bit. To do that safely, I3C Basic SDR specifies how the handoff is to occur. This is summarized below and shown in ***Figure 76***.

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

   This shall be an Open Drain SCL Low period.
2. After the I3C Basic Target sees the rising edge of SCL, it releases the SDA line to High-Z.

   The I3C Basic Target shall release the SDA line using normal (Push-Pull) timing (release the SDA line as soon as it sees SCL rising).
3. After the rising edge of SCL, the I3C Basic Controller shall drive the SDA line Low.

   As a result, both Controller and Target will be driving the SDA line Low for a short overlap (which is safe).

   The SCL High period may be as short as the minimal t_{DIG_H}, per ***Section 4.3.11.2***.
4. On the falling edge of SCL the I3C Basic Controller shall begin driving data on the SDA line, using Push-Pull drive as shown in ***Figure 76***.

When the Address Header results in a NACK, the Controller may choose to **either**:

1. Continue the transaction, by generating a Repeated START

or

2. Relinquish the Bus, by generating a STOP as shown in ***Figure 77***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.2.3.2 Transition from Address ACK to Mandatory Byte during IBI

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

If the Controller detects one of the Target Addresses, and if the Controller chooses to receive the request, then the Controller will ACK the request per ***Section 4.3.6.2***.

- **ACK:** When the IBI Target Address Header results in an ACK, and the Target Device is capable of sending a Mandatory Byte, then the SDA line has to be turned from Open Drain to Push-Pull for the first data bit. To do that safely, I3C Basic SDR specifies how the handoff is to occur. This is summarized below and shown in ***Figure 14***.
  1. The I3C Basic Controller shall hold the SDA line Low during the ACK (i.e., while the SCL line is Low). This shall be an Open Drain SCL Low period.
  2. After the rising edge of the SCL line, the I3C Basic Target shall drive the SDA line Low as soon as possible. As a result, both Controller and Target will be driving the SDA line Low for a short overlap (which is safe).
  3. After (A) a minimum of the t_{SCO} time reported by the Target Device, (B) a predetermined safe time that could guarantee the takeover by all Targets, or (C) the Controller may optionally keep the SDA line Low until the next falling edge of the SCL line in order to be in full control of the SDA line and make sure it accommodates all t_{SCO} values from different Targets, the I3C Basic Controller may release the SDA line. The SCL High period may be as short as the minimal t_{DIG_H}, per ***Section 4.3.11.2***.
  4. On the falling edge of SCL, the I3C Basic Target shall begin driving data on the SDA line, using Push-Pull drive as shown in ***Figure 14***.
- **NACK:** When the Address Header results in a NACK, as shown in ***Figure 15***, the Controller may choose to either:
  - Continue the transaction, by generating a Repeated START, or
  - Relinquish the Bus by generating a STOP.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

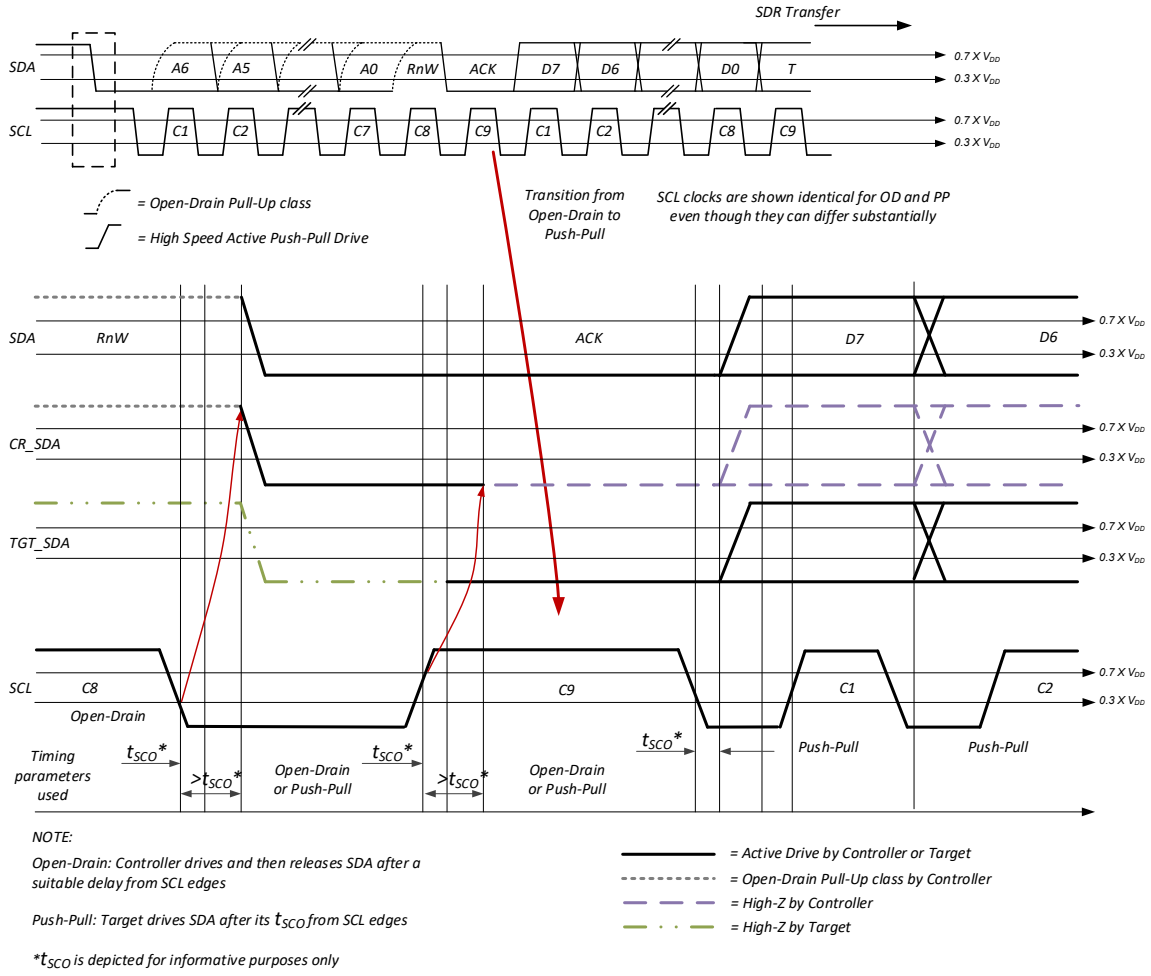
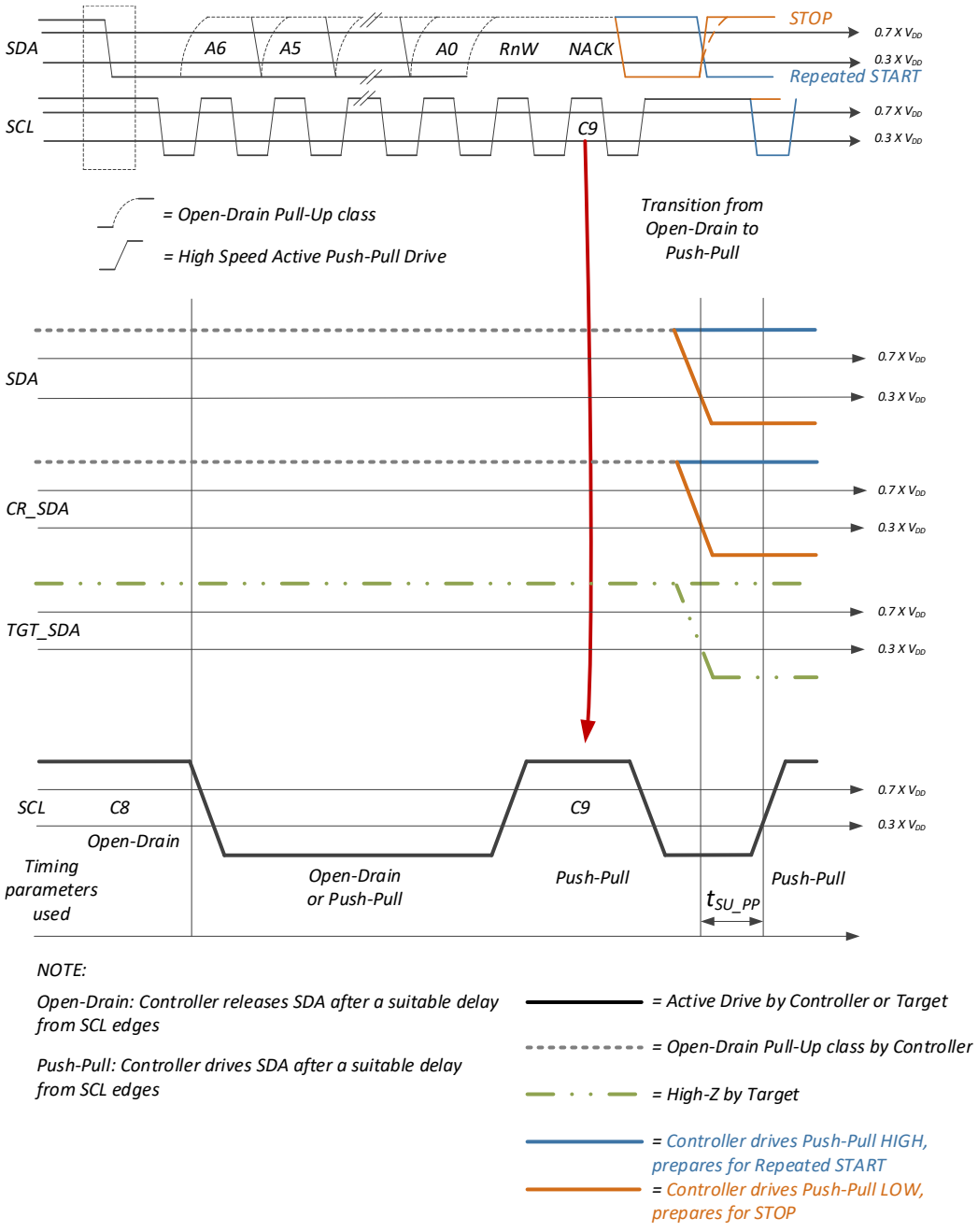


Figure 14 Transition from Address ACK to Mandatory Byte During IBI

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



1209

Figure 15 Transition from IBI Address NACK to Repeated START or STOP

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.2.3.3 Ninth Bit of SDR Controller Written Data as Parity

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

Examples

- If all eight data bits are 1's (0xFF), then the Parity bit value will be 1.
- If all eight data bits are 0's (0x00), then the Parity bit value will be 1.
- If an odd number of bits in the Data Word are 1's (e.g., 0xFE or 0x01), then the Parity bit value will be 0.

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

4.3.2.3.4 Ninth Bit of SDR Target Returned (Read) Data as End-of-Data

In I^2C, Read from Target has the issue that only the Controller ends the Read, so the Target has no ability to control the amount of data it returns. In I3C Basic SDR, by contrast, the Target controls the number of data Words it returns; but it also allows the I3C Basic Controller to abort the Read prematurely when necessary.

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

- The I3C Basic Target returns the ninth bit as 0 (SDA Low) to end the Message (i.e., when reaching the configured MRL value for read and IBI payload size for IBI payload as defined in ***Section 4.3.7.3.6***):
  - The Target shall set SDA Low on the falling edge of SCL.
  - On the following rising edge of SCL, the Target shall set SDA to High-Z.
  - The I3C Basic Controller shall drive SDA Low on the rising edge of SCL, thereby overlapping with the Target.
  - The I3C Basic Controller then shall issue either a STOP as shown in ***Figure 84***, or a Repeated START as shown in ***Figure 85*** (on the next clock, or one after, per the normal I^2C procedure for setting up SDA to issue a Repeated START).
- The I3C Basic Target returns the ninth bit as 1 (SDA High) to continue the Message (and permit the Controller to abort the Message):
  - The Target shall set SDA High on the falling edge of SCL.
  - On the following rising edge of SCL, the Target shall set SDA to High-Z, thereby Parking the Bus for the SCL High period:
    - If the I3C Basic Controller is able to continue the reply from the Target, then it shall do nothing. The weak Pull-Up resistor on SDA will keep SDA High during the SCL High period, as shown in ***Figure 86***.
    - If the I3C Basic Controller wants to abort the Message, then it shall drive SDA Low after the rising edge of SCL, thereby terminating the Message with a Repeated START. The I3C Basic Controller then takes control starting with the falling edge of SCL. The Controller shall ensure enough delay after SCL rising before driving SDA Low to ensure no contention. In order to achieve this delay, the Controller might have to extend the SCL High period.
      Since the transition of SDA Low when SCL is High is a Repeated START, the Controller may begin a new Address (see ***Figure 88***), or it may issue a STOP in the next cycle (see ***Figure 87***). However in a Mixed Bus the Controller should extend the SCL Low period, in order to ensure that any Spike Filters for Legacy I^2C Devices properly reset.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

- The Target shall monitor the SDA on the falling edge of SCL:
  - If SDA is High, then the Target shall continue with the next data value.
  - If SDA is Low (i.e., if there has been a Repeated START), then the Message has been aborted, and the Target shall not drive SDA after that.

4.3.2.4 Use of Clock Speed to Prevent Legacy I²C Devices from Seeing I3C Basic Traffic

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

1. **Pure Bus:** Only I3C Basic Devices are present on the Bus.
2. **Mixed Fast Bus:** Both I3C Basic Devices and Legacy I²C Devices are present on the Bus, such that the Legacy I²C Devices are restricted to ones that are generally permissible (i.e., Target-only, and no Target clock stretching), and that have a true I²C 50 ns Spike Filter on SCL. (I.e., I²C Devices that do not 'see' the SCL line as High when the High duration is less than 50 ns, across all temperatures and processes.)
3. **Mixed Slow/Limited Bus:** Both I3C Basic Devices and Legacy I²C Devices are present on the Bus, such that the Legacy I²C Devices are restricted to ones that are generally permissible (i.e., Target-only, and no Target clock stretching), but that do not have a true I²C 50 ns Spike Filter on SCL.

The Bus designer's choice of Bus Configuration affects what speed options are available for I3C Basic SDR, as well as what HDR Modes are possible at various clock speeds. ***Table 11*** shows the possible options for each Bus Configuration.

Table 11 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 / Limited Bus**
SDR Mode Speed	f_{SCL} (Min) to f_{SCL} (Max)	I²C Messages: Fm or Fm+, I3C Basic Messages: SCL High from $t_{DIG_H_MIXED}$ (Min) to $t_{DIG_H_MIXED}$ (Max) (see ***Section 4.3.2.4.1***)	Fm or Fm+ only[1]
HDR-DDR Mode	f_{SCL} (Min) to f_{SCL} (Max)	SCL High from $t_{DIG_H_MIXED}$ (Min) to $t_{DIG_H_MIXED}$ (Max) (see ***Section 4.3.2.4.1***)	No
HDR-TSP Mode	HDR-TSP Mode is not included in I3C Basic		No
HDR-TSL Mode	HDR-TSL Mode is not included in I3C Basic		No
HDR-BT Mode	f_{SCL} (Min) to f_{SCL} (Max)	SCL High from $t_{DIG_H_MIXED}$ (Min) to $t_{DIG_H_MIXED}$ (Max) (see ***Section 4.3.2.4.1***)	No

Note:

*1) May be faster if all Legacy I²C Devices are index 1, per **Table 6***

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

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

An I3C Pure Bus may simply change the clock speed to any frequency in the allowed speed range, as outlined in ***Section 4.3.1***. By contrast, a Mixed Fast Bus wanting to clock faster than the slowest Legacy I^2C Device needs to take advantage of the Spike Filter, i.e., shall ensure that the SCL High period is shorter than the Spike Filter, in order to prevent the Legacy I^2C Devices from seeing the SCL High period (see $t_{DIG_H_MIXED}$ in ***Table 50***). As a result, the Legacy I^2C Device 'sees' the SCL as staying Low the whole period.

However, a Mixed Fast Bus I3C Basic Controller can change the effective Bus frequency by varying the duty-cycle of SCL. This allows running the data rate slower, for example to accommodate Targets that need a lower rate as defined by GETMXDS CCC (see ***Section 4.3.7.3.18***). It may also be necessary to accommodate Legacy I^2C Devices with Spike Filters that need a longer Low period in order to function properly.

In this model the SCL High period shall never exceed $t_{DIG_H_MIXED}$, thus staying below the 50 ns required by the I^2C Spike Filter; however, the Low period is free to be any length permitted by I3C Basic's allowed clock frequency range. For example, depending on the clock generation capability of the I3C Basic Controller, the SCL Low period may be a multiple of the High period, or it may be any multiple of some higher frequency clock capable of providing a High period less than or equal to $t_{DIG_H_MIXED}$, but greater than the minimum SCL High period as defined in ***Table 48***.

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

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

Such adjustment of the clock Duty Cycle brings three benefits:

- It remains hidden from Legacy I^2C Devices, while still significantly increasing the SDR, HDR-DDR, and HDR-BT data rate.
- It ensures a longer Low period, so that Legacy I^2C Device Spike Filters continue to track SCL as Low.
- It is easy for a Controller to generate, and has no impact on I3C Basic Targets (which just react to clock edges).

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

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.3 Bus Conditions

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

4.3.3.1 Open Drain Pull-Up and High-Keeper

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

This active Pull-Up shall be implemented either:

1. As a passive resistance from V_{DD}, or
2. As a passive resistance from a current source, or
3. In any other method that both:
   A. Balances its current sourcing in order to ensure that SDA rises within t_{rDA} (see ***Table 48***), and
   B. Is not so strong as to prevent a Target with the minimum I_{OL} driver (see ***Table 46***) from driving SDA Low within t_{rDA} (see ***Table 48***).

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

The High-Keeper on the Bus shall be strong enough to prevent system leakage (i.e., the sum of the leakage of all Devices on the Bus) from pulling SDA, and sometimes SCL, Low. The High-Keeper on the Bus shall also be weak enough that a Target with the minimum I_{OL} driver (see ***Table 46***) is able to pull SDA, SCL, or both Low within the Minimum t_{DIG_L} period.

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

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

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

1. No Pull-Up (High-Z)
2. High-Keeper Pull-Up
3. Open Drain Pull-Up

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

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

Note:

The Controller may choose to not use the Open Drain class Pull-Up in cases of Open Drain mode where the SDA happens to be already High (i.e., is coming into the SCL Falling edge). In that case,

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

the Controller may choose to rely solely upon the High-Keeper, in order to use less power if and when a Target drives SDA Low.

4.3.3.2 Bus Condition Timing

After a STOP, three defined timing conditions are used to create permission for a Controller or a Target to take defined actions leading to a START: the Bus Free Condition, the Bus Available Condition, and the Bus Idle Condition. ***Figure 16*** illustrates timing for each condition, and each condition is detailed in a sub-section below.

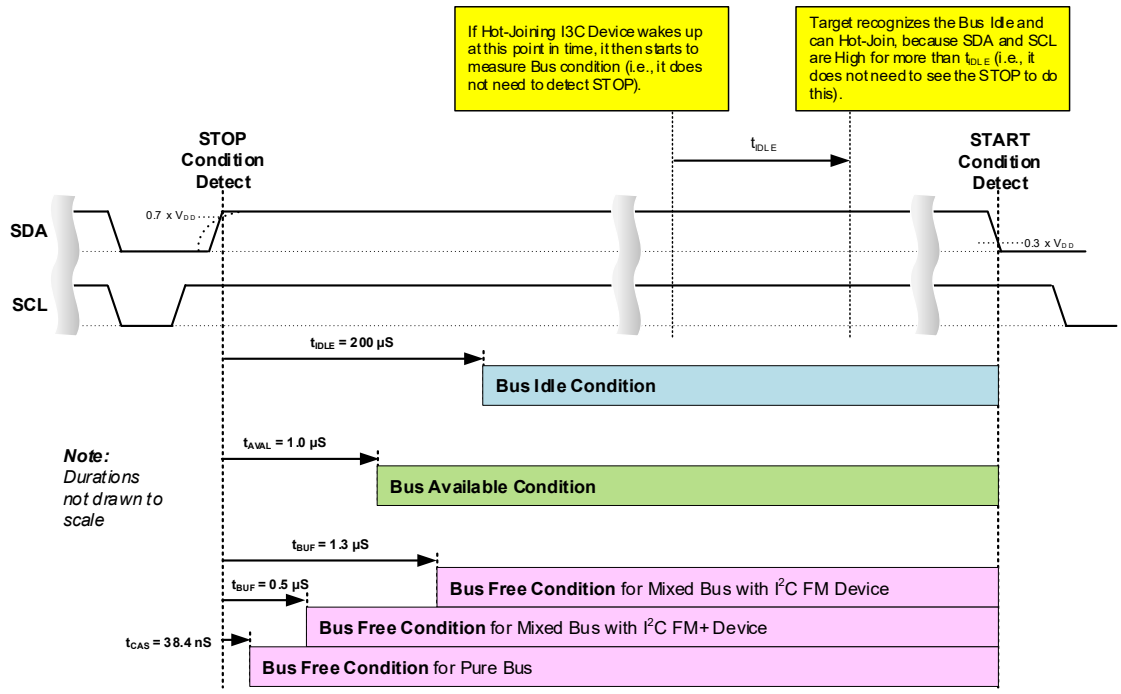


Figure 16 Bus Condition Timing

4.3.3.2.1 Bus Free Condition

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

- **For Pure Bus:** A duration of at least t_{CAS} (see ***Table 49***).
- **For Mixed Bus** (i.e., at least one Legacy I²C Device is present on the I3C Basic Bus)**:** A duration of at least t_{BUF} (see ***Table 48***). Note that the value of t_{BUF} depends upon whether any I²C FM Devices vs. I²C FM+ Devices are present on the Mixed Bus.

4.3.3.2.2 Bus Available Condition

The Bus Available Condition is defined as a period during which the Bus Free Condition is sustained continuously for a duration of at least t_{AVAL} (see ***Table 49***). A Target may only issue a START Request (e.g., for an In-Band Interrupt, or for a Controller Role Request) after a Bus Available Condition.

4.3.3.2.3 Bus Idle Condition

The I3C Basic Bus Idle Condition is defined in order to help ensure Bus stability during Hot-Join events, and is defined as a period during which the SDA and SCL lines both sustain High level for a duration of at least t_{IDLE} (see ***Table 49***).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.3.3 Activity States

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

Table 12 Activity States

Activity State	Activity Interval	CCC
0	1 µs	ENTAS0
1	100 µs	ENTAS1
2	2 ms	ENTAS2
3	50 ms	ENTAS3

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

The Controller uses CCC commands ENTAS0, ENTAS1, ENTAS2, and ENTAS3 (see ***Section 4.3.7.3.2***) to communicate the four expected Bus Activity states to Targets. Each ENTASx CCC has both a Broadcast version and a Directed (per-Target) version. The Controller may switch to a different Activity State in any way, and at any time, by issuing the appropriate ENTASx CCC command.

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

The Activity States mechanism is a basis for a general agreement between Controller and Target, i.e., that the Target may NACK any access occurring sooner than the general time factor. For example: If the Controller sends the ENTAS2 CCC, meaning the Controller is unlikely to initiate a request sooner than 2 ms from the time the CCC is sent, then a request arriving only 1 ms later could result in a NACK (although it will be ACKed if the request is repeated after the Target re-awakens). As a result, the Target shall wake up either upon any Bus activity, or upon matching 7'h7E and its own Dynamic Address.

The Activity State CCCs also adjust the maximum value for I3C Basic Bus timing parameter $\mathbf{t_{CAS}}$ (Clock after START; see ***Table 49***), the maximum amount of time that the Controller may take to generate the SCL clock (drive SCL Low) in response to the Target pulling SDA Low. Note that $\mathbf{t_{CAS}}$ is only a worst-case number; it does not indicate whether or not the Controller will ACK the In-Band Interrupt Request or Controller Role Request. Further, the selected Activity State does not necessarily indicate the time at which the Controller will read additional data from a Target in response to an In-Band Interrupt; that is a private contract between Controller and Target. A Target that does not support the ENTASx CCCs shall have a $\mathbf{t_{CAS}}$ maximum value of 50 ms (the ENTAS3 value).

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

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

4.3.4 Bus Initialization and Dynamic Address Assignment Mode

The Controller-capable Device with the job of initializing the I3C Basic Bus holds the special role of "Primary Controller". The Primary Controller acts as the authority for the initial configuration of the Bus and all Devices, including any Legacy I^2C Devices. Since it acts as the Bus's first Active Controller, during Bus initialization it also performs the Dynamic Address Assignment procedure as part of configuring all I3C Basic Devices that require a Dynamic Address.

Additionally, either the Primary Controller or any Secondary Controller that supports Hot-Join may also subsequently perform the Dynamic Address Assignment procedure while acting as Active Controller, i.e., when a Device needs to be reset or when a new Device is connected to an already-configured I3C Basic Bus and performs a Hot-Join Request.

As detailed in this Section, a Controller must be responsible for performing a Dynamic Address Assignment procedure, in order to provide a unique Dynamic Address to each I3C Basic Device (i.e., Targets or Secondary Controllers) that is connected to the Bus.

Note:

For the remainder of this Section, the term 'Controller' applies to any capable Controller, including the Primary Controller, performing the Dynamic Address Assignment procedure for any reason.

Once a Target or Secondary Controller receives a Dynamic Address, that Dynamic Address shall be used in all subsequent transactions on the I3C Basic Bus, until and unless the Controller changes the Dynamic Address (if applicable). The only way for the Controller to change the Dynamic Address is by using either the RSTDAA CCC command (see ***Section 4.3.7.3.3***) or the SETNEWDA CCC command (if applicable; see ***Section 4.3.7.3.11***). The Controller might choose to change the Dynamic Address due to re-prioritization, unless the Target does not support the SETNEWDA CCC.

The Controller controls the Dynamic Address Assignment process. This process includes an Address Arbitration procedure similar to I^2C's (see ***[NXP01]*** at ***Sections 3.1*** and ***3.1.8***). The I3C Basic Arbitration procedure differs from I^2C by using the values of the 48-bit Provisioned ID and the Device's I3C Basic Characteristic Registers (that is, BCR and DCR), concatenated. The Device on the I3C Basic Bus with the lowest concatenated value wins each Arbitration round in turn, and the Controller assigns a unique Dynamic Address to each winning Device.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.4.1 Device Requirements for Dynamic Address Assignment

This Section describes the requirements for Target Devices to be identified by the Controller for assignment of a Dynamic Address.

4.3.4.1.1 Target Device 48-bit Provisioned ID

A Device that supports the Broadcast Command Code **Enter Dynamic Address Assignment** (ENTDAA, see ***Section 4.3.4.2***) shall have a 48-bit Provisioned ID. The Controller shall use this 48-bit Provisioned ID, unless the Device has a Static Address and the Controller uses the Static Address.

The 48-bit Provisioned ID is composed of three parts:

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

   Note: The Most Significant Bit of the 16-bit MIPI Manufacturer ID is discarded, i.e., only the 15 Least Significant Bits are used.

2. **Bit[32]: Provisioned ID Type Selector** (One bit, 1'b1: Random Value, 1'b0: Vendor Fixed Value)
3. **Bits[31:0]:** 32 bits containing either a **Vendor Fixed Value** or a **Random Value**, depending on the value of Bit[32]:

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

   - **Bits[31:16]: Part ID:** The meaning of this 16-bit field is left to the Device vendor to define.
   - **Bits[15:12]: Instance ID:** The value in this 4-bit field should identify the individual Device, using a method selected by the system designer. For example: straps, fuses, non-volatile memory, or another appropriate method.
   - **Bits[11:0]:** The meaning of this 12-bit field is left for definition with additional meaning. For example: deeper Device Characteristics, which could optionally include Device Characteristic Register values.

   If the value of Bit[32] is 1'b1: Random Value, then:

   - **Bits[31:0]:** 32-bit value randomly generated by the Device. This value can be queried via General Test Mode, using the Command Code **Enter Test Mode (ENTTM)** (see ***Section 4.3.7.3.8***).

Note:

*Under Vendor Test Mode, the Device may provide a fully random or pseudo-random 32-bit value in Bits[31:0] of its Provisioned ID (see **Section 4.3.7.3.8**). Bits[47:33] shall not be randomized.*

4.3.4.1.2 Unique Identifiability Methods

In order to support the Dynamic Address Assignment procedure, each I3C Basic Device to be connected to an I3C Basic Bus shall be uniquely identifiable before starting the procedure. Two mechanisms are defined:

1. The Device shall have a 48-bit Provisioned ID (see ***Section 4.3.4.1.1***).
2. In addition, the Device may also optionally have a Static Address, in which case the Controller may use that Static Address (presuming it is known to the Controller). For example, an Address similar to what I^2C specifies ***[NXP01]***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.4.2 Bus Initialization Sequence with Dynamic Address Assignment

Bus Initialization and Dynamic Address Assignment shall be executed according to the following sequence. See also ***Figure 183***, Dynamic Address Assignment FSM.

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

In the procedure below, steps 1 and 2 shall be performed only by the Primary Controller, and only during Bus initialization. Subsequent steps may be performed by any capable Controller that is the Active Controller for subsequent Dynamic Address Assignment as needed.

1. The Primary Controller 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 Host, the following data:
   A. The number of I3C Basic compliant Devices that need to receive a Dynamic Address,
   B. The data for any I3C Basic Devices resident on the I3C Basic Bus that already have I^2C Static Addresses, and
   C. The data for any Legacy I^2C Devices resident on the I3C Basic Bus.
2. The Primary Controller should assign Dynamic Addresses to any I3C Basic Devices with a known Static Address, using the Command Code **Set Dynamic Address from Static Address** (SETDASA, see ***Section 4.3.7.3.10***), or by assigning all I3C Basic Devices their known I^2C Static Address using the Command Code **Set All Addresses to Static Address** (**SETAASA**) (see ***Section 4.3.7.3.21***).

   Under these pre-conditions, Targets supporting the SETAASA CCC and Targets only supporting the SETDASA CCC (see ***Section 4.3.7.3.10***) can be mixed on the I3C Basic Bus. Via SETDASA, the Controller will individually assign a Dynamic Address to each Target not supporting SETAASA.

   In a system that mixes I^2C-capable Devices and non-I^2C-capable Devices, the Controller should send the SETAASA CCC before sending the ENTDAA CCC (see ***Section 4.3.7.3.4***), in order to assign Dynamic Addresses to the I3C Basic-only Targets. In addition, any Target having a restricted Address (see ***Table 10***) shall not support SETAASA.

   All I3C Basic Target Devices shall support at least one of the aforementioned Dynamic Address assignment methods.
3. If the Active Controller knows it has already assigned all Targets their Dynamic Address(es) in the preceding Bus initialization steps, then it may skip the remaining steps in this procedure (i.e., is permitted to not use the ENTDAA CCC). Otherwise, the Active Controller shall send the Broadcast Command Code **Enter Dynamic Address Assignment** (ENTDAA, see ***Section 4.3.7.3.4***).

   Note:

   A Target Device that does not implement the ENTDAA Broadcast Command Code CANNOT BE RELIED ON TO BE ASSIGNED A DYNAMIC ADDRESS *on a generic I3C Basic Bus. DAA is an essential basic function of the generic I3C Basic Bus, and ENTDAA should not be omitted without careful consideration.*
4. The Active Controller shall send a Repeated START, and the I3C Basic Broadcast Address 7'h7E with RnW bit High (i.e., Read). Every I3C Basic Device on the I3C Basic Bus that supports Dynamic Address Assignment with ENTDAA and does not yet have an assigned Dynamic Address shall acknowledge the I3C Broadcast Address, except for Hot-Joining Devices that are not eligible to participate (per ***Section 4.3.5***).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

At least one I3C Basic Device on the I3C Basic Bus should acknowledge the I3C Broadcast Address in this step, if a Hot-Join Device had previously initiated a Hot-Join Request to signal that it needed a Dynamic Address and was eligible to participate in Dynamic Address Assignment with ENTDAA.

Note:

*Per **Section 4.3.5**, a Hot-Joining Device must first emit the Hot-Join Request at least once, before it may ACK the I3C Broadcast Address 7'h7E with RnW bit High and participate in a Dynamic Address Assignment procedure with ENTDAA. For example, a Hot-Joining Device that uses the standard Hot-Join method must wait to determine that the Bus is Idle (see **Section 4.3.3.2.3**), before it may request a START as part of a Hot-Join Request. If such a Device has not waited for the minimum time (i.e., Bus Idle Condition) to determine whether the Bus is Idle, then it shall not emit the Hot-Join Request, and hence it cannot participate in Dynamic Address Assignment with ENTDAA.*

*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 I^2C Specification **[NXP01]**.*

5. The Active Controller shall drive only the SCL line. The Active Controller 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 Basic Device that is eligible to participate and responds to the I3C Broadcast Address sent in Step 4 shall drive the SDA line with its own 48-bit Provisioned ID (using Big Endian bit order), until it loses Dynamic Address Arbitration.

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

   Note:

   *As mentioned above, a Hot-Joining Device shall only participate in the Dynamic Address Assignment procedure after emitting the Hot-Join Request (per **Section 4.3.5**).*
7. The Active Controller shall continue to drive the SCL line with the same clock, while still releasing the SDA line. The I3C Basic Device that did not yet lose the Arbitration shall then transfer its Bus Characteristics Register (BCR) and Device Characteristic Register(s) (DCR) per ***Section 4.3.1.2.2***, until it eventually loses Dynamic Address Arbitration. See also ***Section 4.3.4.3***.
8. The Device whose concatenated Provisioned 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 4.3.4.3**.*

   *Per **Section 6.10.3.4**, the Controller can stall SCL after receiving the last bit of the DCR, if it needs time to determine the correct Dynamic Address for the specific Device that has won this Arbitration round.*
9. The Active Controller shall transfer a 7-bit wide Dynamic Address for the winning Device, in Open Drain Mode. This Dynamic Address shall incorporate the priority level that the Active Controller assigns to the Device, per ***Section 4.3.6.2***. The Arbitration-winning Device shall acknowledge the assigned Dynamic Address. This Dynamic Address transfer procedure shall have the following steps:
   - The Active Controller shall drive the 7-bit Dynamic Address, followed by a parity bit (PAR), which is calculated as odd parity. Odd parity is the inverse of the XOR of the 7 bits. Therefore ~XOR(dynamic_address[7:1]) is placed in position 0.
   - If the parity is valid, then the Target shall acknowledge receipt of the Dynamic Address on the next SCL clock. If the parity is invalid, then the Target shall passively NACK on the next SCL.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

10. The Active Controller shall repeat this procedure, jumping back to step 4 until there is no ACK from any Device present on the I3C Basic Bus.
11. This Dynamic Address Assignment procedure shall be ended by the Active Controller issuing a STOP.

Figure 17 shows a typical Dynamic Address Assignment transaction using the flow that is entered with the ENTDAA CCC, including two iterations of steps 4–9 above. This figure shows how the Controller may send the ENTDAA CCC after either a START or a Repeated START.

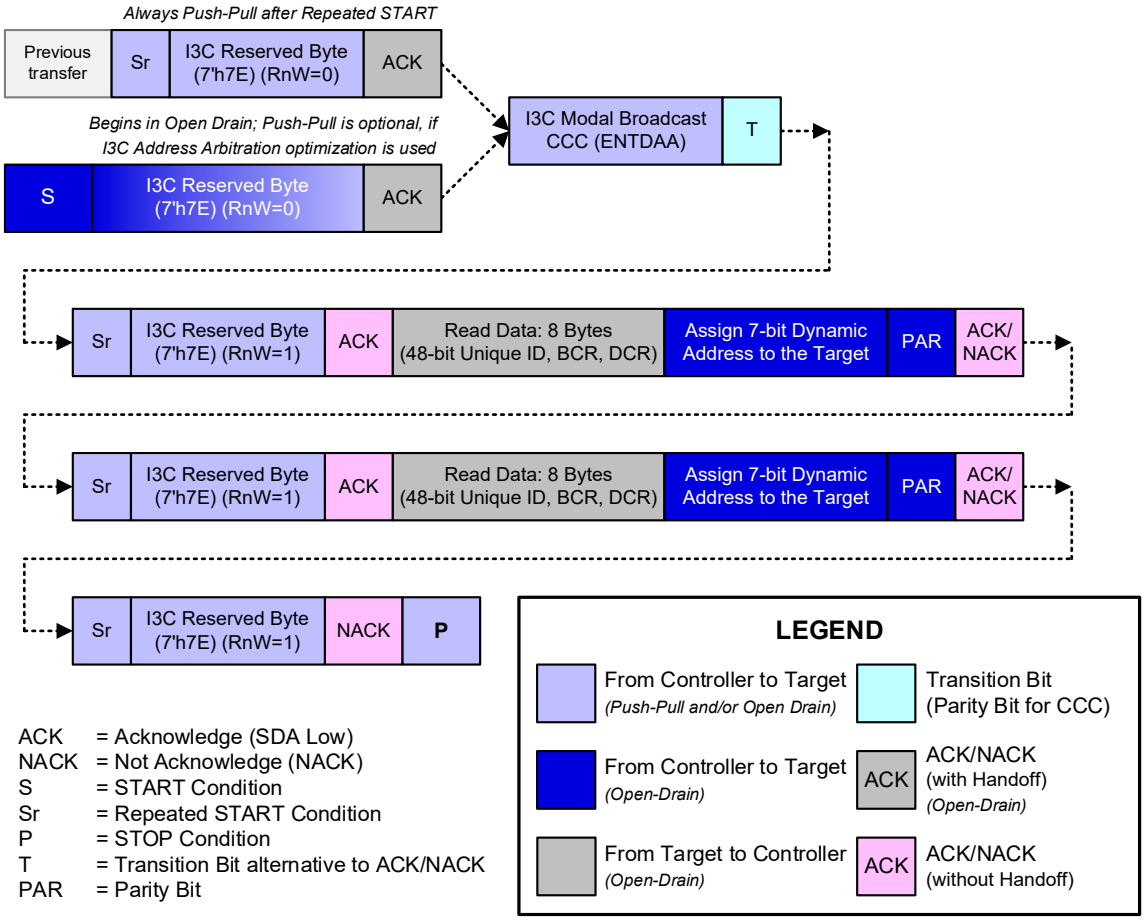


Figure 17 Dynamic Address Assignment Transaction

Regarding this Dynamic Address Assignment procedure, note that:

- The Active Controller may end the Dynamic Address Assignment procedure at any time, even if some of the pre-established I3C Basic Devices have not yet received their Dynamic Addresses.
- The Dynamic Address Assignment procedure can be started again any time the I3C Basic Bus is in Bus Free Condition, using the Command Code **Enter Dynamic Address Assignment (ENTDAA)** (see ***Section 4.3.7.3.4***).
- If a given Target does not acknowledge its assigned Dynamic Address, then the procedure requires the Active Controller to continue from step 4. The Target will then participate in the Address Arbitration using the same 48-bit Provisioned ID, and as a result the Target will win the Arbitration round. If the Target does not ACK the Dynamic Address a second time, then the Active Controller shall exit the Dynamic Address Assignment procedure and execute an error management procedure provided by the I3C Basic Bus designer.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

- Any Secondary Controllers wishing to receive the Dynamic Addresses assigned to all I3C Basic Devices present on the I3C Basic Bus can do one, or both, of the following:
  - Follow the Dynamic Address Allocation procedure. If a Secondary Controller has been connected to the Bus since the Bus was initialized by the Primary Controller, then it will have received all the same data for all other I3C Basic Devices that received a Dynamic Address.
  - Monitor the Bus and wait for the Active Controller to send the Command Code **Define List of Targets** (DEFTGTS, see ***Section 4.3.7.3.7***), then acquire the Addresses and Characteristics of all Devices on the I3C Basic Bus from the Broadcasted data.

After the Dynamic Address Assignment process is complete, the Active Controller knows which Devices on the I3C Basic Bus are Secondary Controllers. The Active Controller then addresses the Secondary Controllers with the Command Code **Define List of Targets** (DEFTGTS, see ***Section 4.3.7.3.7***) and transfers the data for all Devices on the I3C Basic Bus.

4.3.4.3 Provisioned ID Collision Detection and Correction

In the Dynamic Address Assignment procedure described in ***Section 4.3.4.2***, multiple I3C Basic-compliant Devices will receive the same Dynamic Address if they provide the Active Controller with both identical 48-bit Provisioned IDs, and identical Device Characteristic Register values. Although the probability of such a coincidence is quite small the potential does exist, and the I3C Basic Bus will not operate properly under such conditions (at least the Instance IDs must be different, see ***Section 4.3.4.1.1***). As a result the Primary Controller must test for this collision condition for any Devices present during Bus initialization, and resolve it if necessary, before the I3C Basic Devices present on the I3C Basic Bus can all be safely used together.

During Bus initialization the Primary Controller knows the number of Devices resident on the I3C Basic Bus requiring Dynamic Address Assignment (per procedure step 1.a. in ***Section 4.3.4.2***), which enables detection of collision errors. At completion of the Dynamic Address Assignment procedure the Primary Controller shall compare this expected number to the final count of Static Addresses and Dynamic Addresses that were actually assigned. If fewer Dynamic Addresses were assigned than expected, then multiple Devices must have received the same Dynamic Address, i.e., a collision has occurred.

If this collision condition is detected, then the Primary Controller shall resolve it using the following method:

- The Primary Controller resets the Dynamic Address Assignment process by using the Broadcast Command Code **Reset Dynamic Address Assignment (RSTDAA)** (per ***Section 4.3.7.3.3***), and then proceeding from step 3 of the procedure in ***Section 4.3.4.2*** from step 8, until the sooner of **either**:
  A. The expected number of I3C Basic Devices is discovered, **or**
  B. A set maximum number of attempts fail. If this limit is reached, then a system error message shall be sent to the Host. To avoid freezing the entire system, a limit of three (3) attempts is recommended.

A Secondary Controller might not be able to test for this collision without also knowing how many Devices are expected to require Dynamic Address Assignment. Additionally, for Devices that are connected to the Bus after it is configured, a Controller might not be able to detect a collision if the Devices both join at the same time and have identical Provisioned IDs.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.4.4 Group Address Assignment Procedure

As an optional feature, multiple I3C Basic Target Devices can share a single Group Address, allowing a Controller Device to send a given I3C Basic Message to all Target Devices in the Group at once rather than one at a time. In a Target that supports the Group Address capability, a Group Address is stored and exists alongside the Target's unique Dynamic Address; i.e., both Addresses are in effect simultaneously. A Target can optionally be designed to support multiple Groups; such Targets store one Group Address for each Group in which the Target is included, up to the maximum number of supported Groups; all of the Group Addresses are in effect simultaneously.

The Active Controller can write to a Group Address just as it would any I3C Basic Dynamic Address. However, the Active Controller shall not attempt to read data from I3C Basic Devices using a Group Address. Both rules apply for SDR and HDR Messages.

An I3C Basic Device with active Target role that has been assigned a Group Address shall not issue an In-Band Interrupt (IBI) using that Group Address.

The Active Controller can configure the Group Address function using the following CCC commands:

- **SETGRPA** (see ***Section 4.3.7.3.24***): Assigns a Group Address to one or more I3C Basic Targets, each selected via its Dynamic Address. (As a result, each such I3C Basic Device must be assigned a Dynamic Address first, before the Group Address is assigned via SETGRPA.)
- **GETCAPS** (see ***Section 4.3.7.3.19***): Gets the various capabilities of the I3C Basic Targets on the I3C Basic Bus, in particular whether the Group Address function is supported.
- **DEFTGTS** (see ***Section 4.3.7.3.7***): Informs any Secondary Controllers about what Addresses, including any Group Addresses, have been assigned to the I3C Basic Targets on the I3C Basic Bus.
- **DEFGRPA** (see ***Section 4.3.7.3.26***): Informs any Secondary Controllers that support the Group Address feature about what I3C Basic Targets on the I3C Basic Bus are included in each Group Address.
- **RSTGRPA** (see ***Section 4.3.7.3.25***): Removes one or more I3C Basic Targets from a Group by resetting their assigned Group Address, disbands a single Group, or removes all Groups and Group Addresses.

To use the Group Address feature, the Active Controller must support the assignment of Group Addresses. The Active Controller shall use the following procedure to assign one or more Group Addresses to each desired I3C Basic Device on the I3C Basic Bus:

1. After the Active Controller assigns Dynamic Addresses to the I3C Basic Devices on the I3C Basic Bus (via the ENTDAA and/or SETDASA CCC), the Active Controller shall determine the capabilities of the I3C Basic Devices on the I3C Basic Bus via the GETCAPS CCC.

   GETCAPS allows the Active Controller to determine, among other things, which I3C Basic Targets support the Group Address function.

   If an I3C Basic Device joins the I3C Basic Bus via Hot-Join, then the Active Controller shall determine that Device's capabilities via the GETCAPS CCC following completion of the Hot-Join (i.e., only after assignment of a Dynamic Address via the ENTDAA CCC).
2. If any of the I3C Basic Targets on the I3C Basic Bus support the Group Address feature, then the Active Controller shall assign the desired Group Address(es) to the desired I3C Basic Targets via the SETGRPA CCC.

   The act of assigning a Group Address to a Target includes that Target in the indicated Group (i.e., as a member).

   If the Active Controller wishes to have multiple Groups, then it would continue to issue the SETGRPA CCC for as many additional times as needed to complete configuration of the desired Groups. An I3C Basic Target capable of belonging to multiple Groups can be

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

included in as many Groups as the Target is capable of supporting, if the Active Controller so desires.

3. After the Active Controller assigns Group Addresses, the Active Controller shall use the DEFTGTS CCC to inform any Secondary Controllers about the Dynamic Addresses and Group Address(es) that were assigned.
4. If any Secondary Controllers support Group Addressing, then the Active Controller shall use the DEFGRPA CCC to inform them about the Group memberships that were assigned.

If the Active Controller detects the disconnection of any I3C Basic Devices from the I3C Basic Bus, then the Active Controller shall execute step 4 of the above procedure.

The Active Controller could use the RSTGRPA CCC to manage Group membership of Targets, and to modify the existing Groups, or reset all Groups and Group Addresses.

In addition to resetting the Target's Dynamic Address, the RSTDAA CCC shall also reset all of the Target's Group Addresses if the Target supports the Group Address function.

Note:

Controllers should use the RSTDAA CCC with caution, and re-assign any Group Addresses as needed after resetting all Dynamic Addresses. Additionally, any reset operation (such as a Target Reset) that causes a Target to lose or reset its assigned Dynamic Address, will also reset any assigned Group Addresses.

Group Addresses may also be used for WRITE transactions that use Multi-Lane formatting, if supported by all Targets assigned to a Group Address (see ***Section 6.7.3.1.1***).

If an I3C Basic Target supports the Group Address feature, and is presented by a composite Device that presents multiple Virtual Targets on the I3C Basic Bus using shared Peripheral logic (per ***Section 4.3.2.1.2***), then each Virtual Target shall report its Group Address capabilities using the GETCAPS CCC. Each such Virtual Target may optionally support multiple Groups, and the Device shall store all assigned Group Addresses for each Virtual Target. The Controller may assign any or all such Virtual Targets to as many Groups as each Virtual Target is capable of supporting, if the Active Controller so desires. The Device's shared Peripheral logic shall match incoming Private Write transfers addressed to any such Group Addresses, and deliver them internally to appropriate logic for each Virtual Target function that has been assigned to that Group Address.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

4.3.5 Hot-Join Mechanism

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

Note:

Hot-Join does not allow Targets to join the I3C Basic Bus before the I3C Basic Bus has been configured.

Hot-Join may be used for:

- I3C Basic Devices mounted on the same board, but de-powered until needed. Such Devices shall not violate electrical limits for Targets when de-powered (or while transitioning) as defined in ***Section 4.3.1***, including capacitive load.
- I3C Basic Devices mounted on a module/board that is physically inserted after the I3C Basic Bus has already been configured. This specification does not attempt to address how that physical insertion is handled, however such insertion shall not disrupt the SCL and SDA lines, including respecting all electrical limits defined in ***Section 4.3.1***.

Hot-Joining Devices (i.e., Hot-Joining Targets) may include any valid Device type that supports the I3C Basic Target role, such as a Secondary Controller. After a Hot-Join, the Active Controller shall use the **DEFTGTS** CCC as described in ***Section 4.3.7.3.7***. To ensure that any Secondary Controllers are aware of all of the available Targets, the Controller shall immediately notify the I3C Basic Bus if it discovers that any previously joined Targets are no longer present on the Bus (e.g., due either to non-response, or to mechanisms outside of the Bus).

Flow and Procedure

Since the Controller is not inherently aware when new Targets join the I3C Basic Bus, the Hot-Join mechanism provides the following procedure for informing the Controller about new Targets. Each Target shall determine its eligibility to emit the special Hot-Join Request on the I3C Basic Bus, per the method that it supports.

1. As each Target connects to the I3C Basic Bus, and when the Target determines that it is eligible to do so (per the Target Hot-Join Requirements below), the Target shall issue an In-Band Interrupt using the reserved Target Address (i.e., 7'h02) as an IBI, using W (write) after the START.

   This is a Hot-Join Request, and the Target shall issue this request at the appropriate time, based on whether it is using the standard Hot-Join method (i.e., based on the Bus Idle Condition) or a passive Hot-Join method (i.e., waiting for an SDR Frame ending with a STOP, per ***Section 4.3.5.3***).

   Note:

   Unlike a typical In-Band Interrupt request, a Hot-Join Request does not have a data payload or a Mandatory Data Byte.

   This requests a Dynamic Address Assignment process using the ENTDAA CCC (see ***Section 4.3.4.2***).
2. The Active Controller shall either ACK the request, to indicate that it has seen the Hot-Join and will assign a Dynamic Address at some point in the future; or NACK the request, to reject the Hot-Join and defer processing.
   A. If the Active Controller NACKs the request, then the Target attempting to Hot-Join shall try again, following the I3C Basic Target Interrupt Request procedure (see ***Section 4.3.6.2***).
   B. If the Active Controller ACKs the request, then the Target knows that the Hot-Join was accepted, and it shall wait for the Active Controller to initiate the Dynamic Address Assignment procedure with ENTDAA (see ***Section 4.3.4.2***). Once the Target has seen the ACK of the Hot-Join Request, it shall **not** send another Hot-Join Request.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

C. The Active Controller may choose to issue a Broadcast CCC to disable Hot-Join by setting the DISHJ bit in the Command Code **Disable Target Events Command** (DISEC, see ***Section 4.3.7.3.1***). The Active Controller may do so after either a NACK or an ACK of the Hot-Join Request, although this typically happens after a NACK. If another Target attempts to Hot-Join before the Controller is ready to assign Dynamic Addresses to the Targets, then the Controller might need to repeat this procedure.

D. Likewise, the Active Controller may choose to re-enable Hot-Join at a later time (i.e., after using the DISEC CCC with DISHJ bit set, to disable Hot-Join). To do this, the Active Controller may issue a Broadcast CCC to enable Hot-Join by setting the ENHJ bit in the Command Code **Enable Target Events Command** (ENEC, see ***Section 4.3.7.3.1***).

If any Hot-Joining Targets had previously received the DISEC CCC with DISHJ bit set, then such Targets may choose to re-send the Hot-Join Request after receiving the ENEC CCC with ENHJ bit set.

3. The Active Controller should eventually issue a Broadcast Command Code **Enter Dynamic Address Assignment** (ENTDAA, see ***Section 4.3.7.3.4***) to start the Dynamic Address Assignment process. Since only eligible Targets with no Dynamic Address shall participate, this allows the newly joined Target to receive its Dynamic Address.

   A. If the Active Controller has limited capabilities (e.g., is a reduced functionality Secondary Controller Device, per ***Section 6.5.3.3.4***), then it may choose to not process the Hot-Join Request, by using the NACK followed by the DISEC CCC (as listed above), and then it may pass the Controller Role to a more capable Controller (per ***Section 6.5.3.3.6***) at a later time.

   B. A Target that supports Hot-Join (either via the standard method, or the passive method per ***Section 4.3.5.3***) is eligible to participate in Dynamic Address Assignment, if and only if it has issued the Hot-Join Request (i.e., using the reserved Target Address, 7'h02).

   Note:

   Controller ACK in a Hot-Join Request does not imply that the Dynamic Address Assignment will necessarily start immediately. The Controller may wait for a potentially prolonged period before issuing the ENTDAA CCC.

   Note:

   Hot-Join is only compatible with ENTDAA. Targets that do not support ENTDAA shall not use Hot-Join.

   If an **Enter Dynamic Address Assignment (ENTDAA)** Command Code is issued as described in ***Section 4.3.7.3.4***, then the Target shall transmit its 48-bit Provisioned ID per the normal mechanism. Hot-Join Targets required to receive the same assigned Dynamic Address every time they join can use a fixed-value ID, per ***Section 4.3.4.1.1***.

Figure 184 in ***Annex E*** illustrates the Hot-Join flow for both standard and passive Hot-Joining Targets. This flow diagram is strictly informative and does not provide every possible combination of paths.

If a Target drops off the I3C Basic Bus due to being detached or depowered, then the Controller might not be aware of this. The Controller may determine this condition by repeated attempts to contact the Target using a safe Command Code (i.e., a CCC that the Target is required to always respond to), such as **Get Device Status** (GETSTATUS, see ***Section 4.3.7.3.15***). If the Controller determines that the Target is no longer part of the I3C Basic Bus, then it shall either recycle that Target's Dynamic Address, or else reserve that Target's Dynamic Address in case that Target later re-joins the Bus.

Note:

If the Controller's first attempt to emit the first I3C Basic Address Header (i.e., emitted slowly to allow for I3C Basic Targets to disable Spike Filters) is converted into a Hot-Join, then the Controller needs to ACK it, then emit STOP and attempt to emit the I3C Basic Address Header again, per ***Section 4.3.2.1.1****.*

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

A Hot-Joining Target may power-up at the same time as the Primary Controller (e.g., may be connected to the I3C Basic Bus when power is applied to the system). In that case, the Hot-Joining Target might pull SDA Low even before the I3C Basic Bus has been started, assuming that SCL and SDA are being pulled up. If a Primary Controller needs 1 ms or more in order to start acting on the I3C Basic Bus, then that Primary Controller shall tolerate the situation of SDA being held Low when it is ready to initialize the I3C Basic Bus, or it may hold SDA and/or SCL Low until ready, keeping the Hot-Joining Target from seeing the Bus Idle Condition.

Hot-Joining Targets should implement some method that allows the system designer to prevent the Device from attempting to Hot-Join when the Device is not being used as a Hot-Joining Target. For example: either NVMEM or a pin strap could be used when such a Device is soldered onto the board and powered with the rest of the I3C Basic Bus.

Target Hot-Join Requirements

A Hot-Joining Target (i.e., a Hot-Joining Device attempting to join with active Target role) should be 'single-minded' in nature, focusing exclusively on emitting the Hot-Join Request, until the Active Controller either ACKs the request (thereby indicating that it will eventually start the process of Dynamic Address Assignment) or NACKs the request (thereby indicating that the Target should continue providing Hot-Join Requests until the Active Controller either ACKs such a request, or sends the Broadcast DISEC CCC to disable Hot-Join, as mentioned above).

1. The first time the Hot-Joining Target connects to the I3C Basic Bus, the Target shall wait for the appropriate opportunity to send the Hot-Join Request, based on its eligibility:
   A. A Hot-Joining Target using the standard method shall become eligible after waiting for a period of at least Bus Idle Condition (t_{IDLE}, see ***Table 49***) and ensuring that the Bus remains Idle (i.e., that SDA and SCL are both High, per ***Section 4.3.3.2.3***).
   B. A Hot-Joining Target using a passive method shall follow the conditions of eligibility defined in ***Section 4.3.5.3***.
   C. If multiple Hot-Joining Targets become eligible at the same time, then they could all attempt to emit the Hot-Join Request together (i.e., at the same time). This could apply to Hot-Joining Targets using any combination of standard or passive Hot-Join methods.
2. The Hot-Joining Target shall, when eligible, send the Hot-Join Request on the I3C Basic Bus, following a START.
   A. The Hot-Join Request is an IBI from the reserved Target Address (i.e., 7'h02), with the RnW bit Low (i.e., Write).
   B. The Target shall send the Hot-Join Request, by either waiting for a START, or requesting a START by pulling SDA Low (for the standard method only). After the START, the Target shall drive the Address of 7'h02 / W into the Arbitrable Address Header.

   Note:

   To issue a START, a Hot-Joining Target using the standard method must pull SDA Low; the Target shall then wait for the Controller to drive SCL Low. This applies to the standard Hot-Joining method only; a passive Hot-Joining Target that does not have a timer must wait for another Device to drive a START Request, per **Section 4.3.5.3**.

   In the case where multiple Hot-Joining Targets successfully emit the Hot-Join Request at the same time, it would not be necessary for each of these eligible Targets to emit separate Hot-Join Requests in succession, with delays between each request. If a Hot-Join Request was successfully emitted, and if the Active Controller responds to that Hot-Join Request, then each such Target shall participate in a subsequent Dynamic Address Assignment procedure with ENTDAA that would be initiated by the Active Controller.

   C. The Target shall watch for the Active Controller to either ACK or NACK the Hot-Join Request.
3. The Hot-Joining Target shall conditionally continue sending the Hot-Join Request on each subsequent START (i.e., at the next appropriate opportunity) until and unless the Active Controller provides an ACK for the Hot-Join Request.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Once such an eligible Target sees the Active Controller ACK the Hot-Join Request (i.e., one that it initiates or that it is eligible to initiate, per step 1), then it shall stop sending further Hot-Join Requests.

4. Once the Hot-Joining Target sees the Active Controller ACK or NACK the Hot-Join Request, it shall follow all normative requirements for the Role of an I3C Basic Target per ***Section 4.3.2.1*** (i.e., one that has not yet received its Dynamic Address).

Note:

A Hot-Joining Target that uses the standard Hot-Join method (i.e., based on the Bus Idle Condition) shall not follow these normative requirements (i.e., as a Target that has not yet received its Dynamic Address) until and unless it emits the first Hot-Join Request, and then receives an ACK or NACK from the Active Controller. This is because the Bus Idle Condition is related to the minimum time necessary to determine that the Bus is free and not in any HDR Mode (i.e., no transfer is in progress).

A. This shall include receiving and appropriately processing all required Broadcast CCCs.

B. This may also include the Broadcast ENEC CCC with the ENHJ bit set, if the Target chooses to support re-sending Hot-Join Requests as a response to this Broadcast CCC, and if it previously received the Broadcast DISEC CCC with the DISHJ bit set. Note that this shall not affect whether the Target supports (or remains ready to respond to) the Broadcast ENTDAA CCC.

C. The Target shall also remember that it has seen an ACK/NACK response to at least one Hot-Join Request, and it shall remain ready to respond to the Broadcast ENTDAA CCC for subsequent Dynamic Address Assignment. Note that this may follow immediately after any of the following:

   i. An ACK or a NACK, followed by a Repeated START;

   ii. A STOP, followed by a delay (i.e., Bus Free Condition, or longer); or

   iii. A Repeated START that is followed by other valid Bus traffic, such as Private Reads, Private Writes, and/or other CCCs.

Note:

Once the Target has emitted the Hot-Join Request, the Target shall remain ready to respond to the Broadcast ENTDAA CCC, even if it sees a NACK to a Hot-Join Request, or a NACK followed by the Broadcast DISEC CCC with the DISHJ bit set.

If a Hot-Joining Target has not yet emitted the Hot-Join request, then it shall not be eligible to participate in Dynamic Address Assignment with the ENTDAA CCC.

A Hot-Join Request shall always follow a START, which means that this uses an Arbitrable Address Header. Hot-Joining Targets should attempt to arbitrate the reserved Target Address (i.e., 7'h02), per the conditions of eligibility to emit the Hot-Join Request (i.e., standard vs. passive Hot-Join method). Since this reserved Target Address has a very low value, it will almost always have higher priority over other I3C Basic Target Addresses, which means it is very likely to win Arbitration.

4.3.5.1 Failsafe Device Pads

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

The Device SCL and SDA pads shall be designed so as to avoid drawing current from the system SCL and SDA lines, both when they are being held High and when they are being held Low. Such leakage may be avoided by using special connectors, or other isolation methods (e.g., physical connection order in a plug). This excess current should be avoided whether due to diode effect, rail tie for ESD, or clamps. The leakage current variation between unpowered and powered shall not exceed the normal variation range of an active pad, I_i, as specified in ***Section 4.3.1***, ***Table 46***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.5.2 Legacy I²C Buses and Hot-Join

Hot-Join is not compatible with Legacy I²C. A Target shall not attempt a Hot-Join when on a Legacy I²C Bus. This may be accomplished, with a Hot-Join capable Target, in one of two ways:

1. The Target may use a pin or NVMEM to allow the system designer to turn the feature off when it is not appropriate for the system, as noted in ***Section 4.3.5***.
2. The Target may use a passive Hot-Join method per ***Section 4.3.5.3***. This prevents it trying to Hot-Join until it is sure it is on an I3C Basic Bus.

4.3.5.3 Passive Hot-Join

The passive Hot-Join method works the same basic way as outlined in ***Section 4.3.5***, except with a modification to the conditions of eligibility for a Hot-Joining Target.

A standard Hot-Joining Target must first wait for at least the Bus Idle Condition, and then it may pull SDA Low, or it may wait for a START. A passive Hot-Joining Target will first wait for an SDR Frame addressed to the Broadcast Address (7'h7E/W) ending in STOP, and then wait for the next START, which must be issued by either the Active Controller or any other Target that can request a START. This ensures that the Target is on an I3C Basic Bus. **See Errata 01, Item 7**

A passive Hot-Joining Target does not necessarily use or enable an internal timer that would otherwise be used to wait for the Bus Idle Condition (i.e., to emit the Hot-Join Request). If it does, then it must clearly recognize a START with the Broadcast Address (7'h7E) that another Device might issue (i.e., as opposed to a Repeated START).

In practice, this means that a passive Hot-Joining Target will first determine that it is indeed on an I3C Basic Bus in SDR Mode (per the conditions above), and then it will:

Either:

- Wait for at least Bus Free Condition, if the Active Controller issues the next START (i.e., for the next I3C Basic transaction in SDR Mode);

Or:

- Wait for at least Bus Available Condition, if another Target issues the next START (i.e., for an In-Band Interrupt Request or a Controller Role Request).

Once a passive Hot-Joining Target recognizes the next START, it shall then issue the Hot-Join Address (i.e., the reserved Target Address, 7'h02) into the Arbitrable Address Header to emit its Hot-Join Request, in the same manner as a standard Hot-Joining Target. **See Errata 01, Item 8**

Note:

Because a passive Hot-Join Target waits for a recognizable I3C Basic Address Header, on some Buses this could lead to a long delay before the Target is in fact Hot-Joined to the Bus.

A Hot-Joining Target that uses such a passive Hot-Join method should still follow all normative requirements for the Role of an I3C Basic Target in ***Section 4.3.2.1*** *(i.e., one that has not yet received its Dynamic Address), since it would have already seen an SDR Frame ending in STOP. In this case, such a Target should also process any Broadcast CCCs that it might receive before it emits the Hot-Join Request (such as ENEC/DISEC), except for the ENTDAA CCC since it would not be eligible to participate in Dynamic Address Assignment before it emits the Hot-Join Request.* **See Errata 01, Item 9**

If a Hot-Joining Target supports both the passive and standard Hot-Join methods, then it may use either method or both methods to emit the Hot-Join Request. For example, such a Target may opportunistically attempt to use a passive Hot-Join method if it clearly recognizes a START that another Device has issued; or it may wait for at least the Bus Idle Condition and then issue its own START, if the Bus remains Idle and no other Devices have issued a START during that period.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.6 In-Band Interrupt

This Section specifies I3C Basicôs priority-based In-Band Interrupt mechanism.

4.3.6.1 Priority Level

In I3C Basic, Priority Level controls the order in which In-Band Interrupt requests and Controller Role Requests from I3C Basic Targets are processed. The Priority Level of each Target (or Secondary Controller) in an I3C Basic Bus instantiation is encoded in its Address, with lower Addresses having higher Priority. That is, Targets with lower value Addresses and higher Priority Levels have their In-Band Interrupts and Controller Role Requests processed sooner than Targets with higher value Addresses and lower Priority Levels. This Priority Level ordering is a natural outcome of the I3C Basic Address Arbitration behavior specified at ***Section 4.3.2.2.1***, where Address bits with value 0 prevail over bits with value 1.

As a result, during each Dynamic Address Assignment operation (see ***Section 4.3.4***) the Active Controller should assign lower Addresses to Targets for higher Priority In-Band Interrupt requests.

4.3.6.2 I3C Basic Target Interrupt Request

In order to request an interrupt, an I3C Basic Target shall emit its Dynamic Address into the arbitrated Address header following a START (but not following a Repeated START). If no START is forthcoming within the Bus Available Condition, then the I3C Basic Target may issue a START by pulling the SDA line Low, and then wait for the Active Controller to pull the SCL Low.

If the Active Controller sets the SCL line Low, thus completing the START, and then provides clocks on the SCL line, then the Target shall drive the SDA line with its own Address, followed by an RnW bit with value 1'b1. If more than one Target has issued an IBI request after the same START, then the Active Controller shall process those IBIs in Priority Level order, per ***Section 4.3.6.1***. The Target(s) that lost the Arbitration may issue another IBI request, but shall not do so until after the Bus Available Condition.

At that point, the Active Controller shall perform one of the following three actions:

Either:

1. Accept the IBI by providing the ACK bit. The actions available to the Active Controller depend upon the value of the Targetôs BCR[2] bit (in the Targetôs BCR register):
   A. If the I3C Basic Targetôs BCR[2] bit is set to 1, then per ***Section 4.3.1.2.1*** the Active Controller shall read the Mandatory Data Byte that follows the accepted IBI request at any ôreadôclock speed allowable by the Target. This operation is similar to a ôreadôfrom the Target and all the related rules apply. Note that the Active Controller cannot avoid receiving the Mandatory Data Byte, since it is transmitted in Push-Pull mode.

   After sending the Mandatory Data Byte, the Target may optionally send (i.e., or not send) additional IBI data bytes, up to the IBI payload size limit per ***Section 4.3.7.3.6*** (the ***Set/Get Max Read Length*** CCC) indicating end of data by setting T bit to 0 per ***Section 4.3.2.3.4***. If the Target sends any such additional IBI data bytes, then the Active Controller may either accept or terminate them, depending upon (a) any private contract between Controller and Target regarding the meaning of these bytes, and (b) the Controllerôs current willingness and/or ability to take them. After the Target sends any such additional IBI data bytes, the Active Controller may issue either a STOP, or a Repeated START, per the normal I3C Basic protocol.

   If the Active Controller completes this transfer before all additional Data Bytes are read, then the Target shall not repeat the as-yet unserviced IBI request; the Active Controller has assessed the request, and will service it at a later time. Depending upon the character of the data, the Controller can either:

   i. Attempt to read either the full data (e.g., for short data packets), or

Copyright É 2016ï 2025 MIPI Alliance, Inc.

ii. Continue from the position at which the transfer was interrupted (e.g., for FIFO transfers), or

iii. Abort the IBI service (e.g., for Timing Control information).

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

One conceptual time diagram of this sequence is shown in ***Figure 18***.

Open Drain	Open Drain	Open Drain	Hand Off	Push-Pull	Drive High or Low, and then High-Z	Push-Pull
S	Target_addr_as_IBI/R	Controller_ACK	SCL High	Target_byte	T	Sr

Figure 18 IBI Sequence with Mandatory Data Byte

B. If the I3C Basic Target's BCR[2] bit is set to 0, then the Active Controller may take any other valid I3C Basic action to terminate the frame after providing the ACK bit; i.e., issue either a STOP or a Repeated START.

Or:

2. Refuse the IBI without disabling interrupts. To do this the Active Controller simply passively NACKs, thus denying the IBI. The Target will normally try again after the next Bus Available Condition, or on the next START.

Or:

3. Refuse the IBI and disable interrupts. To do this, the Active Controller NACKs to deny the IBI, then sends a Repeated START, and finally sets the DISINT bit in the Command Code **Disable Target Events Command (DISEC)** to the interrupting Target as described in ***Section 4.3.7.3.1***. (The Active Controller can set the ENINT bit in the Command Code **Enable Target Events Command (ENEC)** at a later time.)

Note:

If the Controller chooses to set the DISINT bit in the Command Code Disable Target Events Command (DISEC) to a Target, then it should clear any pending ACKed interrupts before doing so.

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

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.6.2.1 Mandatory Data Byte

The Mandatory Data Byte of the IBI payload is the data that follows the Dynamic Address when a Device sends an IBI request. The availability of a Mandatory Data Byte, together with the payload information that follows it, is indicated by bit BCR[2] in the Bus Characteristics Register (see ***Section 4.3.1.2.1***).

It is called a Mandatory Data Byte because the Target Device takes over the line when the Controller ACKs the IBI request and the Target continues to send data; the Controller cannot decline the reception of the first byte, and must wait for the T-Bit to terminate any subsequent data.

The Mandatory Data Byte provides the Controller additional information about the event that has happened, and is divided in two fields as shown in ***Table 13***:

- **Interrupt Group Identifier:** The three most significant bits: MDB[7:5]

  Values for the Interrupt Group Identifier are defined by MIPI Alliance I3C WG for different use cases.

- **Specific Interrupt Identifier:** The five least significant bits: MDB[4:0]

Example: D2DT is a MIPI Alliance I3C WG function. It uses the value 3'b001 for the Interrupt Group Identifier field, and the value 5'b10111 (5'h17) for the Specific Interrupt Identifier field.

MIPI Alliance maintains a web-based registry of defined MDB values ***[MIPI05]***.

Table 13 Mandatory Data Byte Field Format

Interrupt Group Identifier Field			Specific Interrupt Identifier Field				
MDB[7]	MDB[6]	MDB[5]	MDB[4]	MDB[3]	MDB[2]	MDB[1]	MDB[0]

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

2020

Table 14 Mandatory Data Byte Value Ranges

MDB Category	Interrupt Group Identifier	Specific Interrupt Identifier Value		Description
	MDB[7:5]	MDB[4:0]		
User Defined	3'b000	5'h00 – 5'h1F	Vendor Reserved	Defined by the vendor and reserved for vendor definition
I3C WG	3'b001	5'h00 – 5'h0C	MIPI Alliance I3C WG Reserved	Defined and interpreted based on some reserved values allocated by MIPI Alliance I3C WG. Indicates that the interrupt is related to a specific functionality, e.g., D2DT.
		5'h0D	Error with additional data bytes in IBI payload	
		5'h0E	Generic error, no additional data bytes in IBI payload	
		5'h0F – 5'h11	MIPI Alliance I3C WG Reserved	
		5'h12	Monitoring Devices Interrupt Enabling	
		5'h13 – 5'h16	MIPI Alliance I3C WG Reserved	
		5'h17	D2DT Identifier	
		5'h18 – 5'h1F	MIPI Alliance I3C WG Reserved	
MIPI Groups	3'b010	5'h00 – 5'h01	MIPI Alliance Camera WG Reserved	Allocated and reserved by MIPI Alliance Working Groups, and approved by MIPI Alliance I3C WG. Indicates that the interrupt is generated by a Device that wants to send a specific MIPI Alliance WG-related interrupt (e.g., Camera WG or Debug WG)
		5'h02 – 5'h1B	MIPI Alliance Reserved	
		5'h1C – 5'h1D	MIPI Alliance Debug WG Reserved	
		5'h1E – 5'h1F	MIPI Alliance Reserved	
Non MIPI Reserved	3'b011	5'h00 – 5'h1F	Non MIPI Reserved	Allocated for uses outside of MIPI Alliance
Timing Information	3'b100	5'h00 – 5'h1F	Vendor Reserved	May have any value defined by the vendor
Pending Read Notification	3'b101	5'h00 – 5'h0C	MIPI Alliance I3C WG Reserved	Indicates that the interrupt is generated by a Device that wants to send a specific MIPI Alliance WG-related or vendor-specific interrupt, and has a data message that will be returned on the next Private Read if ACKed (see ***Section 4.3.6.2.2***)
		5'h0D	MIPI Alliance Debug WG Reserved	
		5'h0E	MCTP Packet (for DMTF)	
		5'h0F	MIPI Alliance I3C WG Reserved	
		5'h10 – 5'h1F	Vendor Reserved	
Reserved	3'b110	—		Reserved
Reserved	3'b111	—		Reserved

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.6.2.2 Pending Read Notification

Target Devices that support In-Band Interrupt requests may also implement support for a particular Pending Read Notification contract, by which they may inform the Controller that there is data to be read for a specific purpose. Such a Target that expects that the Controller should also support this contract shall indicate its support by returning a value of 1'b1 in bit 6 of GETCAP3 byte (the third returned data byte, when the GETCAPS Format 1 CCC is sent (see ***Section 4.3.7.3.19***).

To signal a Pending Read Notification, the Target shall send an IBI with a Mandatory Data Byte value (see ***Section 4.3.6.2.1***) for which the first three bits (i.e., Interrupt Group Identifier) matches 3'b101. The remaining five bits in the Mandatory Data Byte value (i.e., the Specific Interrupt Identifier) may be specific for a particular type of Target, as assigned by MIPI Alliance I3C WG, or it may be any other value in the region reserved for a Vendor device.

If the Controller accepts the IBI and reads the Mandatory Data Byte from the Target, then it shall signal acceptance of the Controller's obligation to read the data, per this contract. The Target shall note that the MDB has been read, and treat the Pending Read Notification as active: i.e., it shall make the data available on the next Private Read request received from the Controller. However, if the Controller terminates the IBI before reading the MDB, or if the Controller NACKs the IBI, then it shall not signal acceptance, and the Target may try again at a later time.

The Controller should also read any additional data bytes as part of the IBI payload (see ***Section 4.3.6.2***), as the Target may add additional bytes specific to the type of Target or its function (per the Specific Interrupt Identifier) to provide context or further describe or classify the data that the Target will return on the next Private Read request. If present, the format and length of any additional bytes in the IBI payload shall depend on the Specific Interrupt Identifier.

Once the IBI and its MDB has been serviced and accepted, the Controller is obligated to initiate a subsequent Private Read request to the Target, to read the data that has been made available and ready for this Private Read request. The Controller may do this immediately after the IBI has been serviced, or it may do this at a later time. However, the Controller should initiate this Private Read request before taking any actions that would either reset the Target Device (thereby causing the data to be lost), or before any other queued commands or actions initiated by its Host can initiate a Private Read for another purpose (thereby reading the data and associating it with another transaction). The Controller may queue up several Private Read Notifications from several Targets and process them in any order.

The Private Read request may be done in SDR Mode, or in any supported HDR Mode using any HDR Read command value specific to that HDR Mode. The choice of whether to use an HDR Mode may be a private contract between the Controller and the Target, or it may be indicated in the Specific Interrupt Identifier or any additional data bytes in the IBI payload.

Note:

If the Target also supports any HDR Modes, then the Target may choose to support certain HDR Read command values for the active Pending Read Notification, by private contract or by assignment. The Target may also continue to support HDR Reads for other purposes, using other HDR Read command values, while it waits for the Controller to initiate the HDR Read for the active Pending Read Notification.

Once the Controller has initiated the Private Read request, it should read the data from the Target, queue it for processing or transfer to its Host, and retain the association of the data with the IBI notification, including the specific MDB value as well as any additional bytes in the IBI payload.

A Target may hold only one active Pending Read Notification at any time, while it is waiting for the Controller to read the data:

- While the Target is waiting for the Controller to initiate a Private Read request for its active Pending Read Notification, it shall not send an IBI with a Mandatory Data Byte value to indicate another Pending Read Notification until after the Controller has initiated the expected Private Read request to consume the data for the active Pending Read Notification.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

- However, the Target may send an IBI with another Mandatory Data Byte value (i.e., one that does not signal a Pending Read Notification) for other reasons, such as reporting error conditions or other types of interrupt events.
- If the Controller has serviced and accepted the IBI and its MDB, but has not initiated the expected Private Read request in a timely manner, then the Target may re-transmit the IBI with the same MDB (with additional data bytes, as described above) as a reminder.
  - The Controller need not initiate multiple Private Read requests (i.e., one for each additional IBI with MDB for a Pending Read Notification); the most recent IBI shall be the notification to consume the data for the active Pending Read Notification.
  - However, if the Controller does initiate additional Private Read requests due to additional IBIs as reminders, then the Target shall not accept such requests.
- The Target shall keep the data available for the Controller, to satisfy the expected Private Read request, unless it encounters an error (e.g., due to delayed read or other conditions).
  - If the Target encounters an error or is unable to return the data, then it shall not accept the Private Read request (i.e., the Target shall NACK the read), and should also report this as an error (i.e., an IBI with another MDB value).

By default, a Target should not attempt to send an IBI with a Mandatory Data Byte value to indicate a Pending Read Notification until it has been enabled to do so by the Controller.

Note:

*The method for configuring a Target to enable or disable IBIs with Pending Read Notifications is not defined by the I3C Basic Specification. This method might be a private contract between the Controller and the Target, or it might be defined for specific assigned MDB values in **Table 14** (i.e., with Interrupt Group Identifier value of 3'b101).*

4.3.6.3 I3C Secondary Controller Requests to be Active Controller

When the I3C Secondary Controller requests to be an Active Controller:

1. To perform the request, the I3C Secondary Controller shall wait for either a START (not a Repeated START), or a Bus Available Condition and issue its own START.
2. After a START, the I3C Secondary Controller shall issue its own Dynamic Address on the I3C Bus, followed by the RnW bit of 0 to request to become Active Controller.
3. If the I3C Secondary Controller wins the Arbitration by having the lower Dynamic Address, and if the Active Controller is currently willing to relinquish the Controller Role to the requester, then the Active Controller shall respond with ACK.

Figure 185 shows the Controller Role Request flow if the Secondary Controller wins the Arbitration. Any Controller-capable Device shall use this flow to request the Controller Role, including when a former Active Controller needs to regain the Controller Role after acting as a Target.

After acknowledging the Controller Role Request, the Active Controller should prepare for Handoff in a timely manner (see ***Section 6.5.3.1***). The Secondary Controller should also remain ready to accept the Controller Role (i.e., not transition into a lower-power mode or a “deep sleep” state) and respond to all relevant CCCs that the Active Controller may use, as part of the process of preparing for Handoff. ***Section 6.5.3.2*** specifies the process for passing the Controller Role.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.6.4 I3C Basic Active Controller Initiating a Transaction

In order to ensure that any other I3C Basic Device can initiate a Target Interrupt Request or a Controller Role Request, the I3C Basic Active Controller may choose to initiate new Frames with a START (not a Repeated START) followed by the I3C Basic Broadcast Address (7'h7E). This allows another Device to emit its Address into the arbitrated Address header, when the Bus would otherwise be busy and a Device might not be able to wait for the minimum Bus Available period to initiate its own START by pulling SDA Low.

If the other Device wins the Arbitration, then it may drive its IBI Request, and the Active Controller may either ACK or NACK the request. However if no other Device attempts to emit its Address into the arbitrated Address header, then the Active Controller may follow with a Repeated START, or any other valid Bus activity.

If the Active Controller instead drives the same Dynamic Address and RnW bit as a Target issuing an IBI or a Controller Role Request with its own Dynamic Address, then the Arbitration might not have a clear winner, and as a result a retry might be necessary in order to give the Target a chance to request an IBI or the Controller Role (see ***Section 4.3.2.2.3***).

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

4.3.7 Common Command Codes (CCC)

Common Command Codes (CCCs) are I3C Basic's standardized command set. In SDR Mode, Target support for a number of CCCs is Required, some CCCs are Conditionally required, and others are entirely Optional to support (see ***Table 16***). This Section specifies how SDR Mode CCCs are formatted and transmitted on the I3C Basic Bus, how each CCC functions, and which CCCs are Required, Conditional, and Optional for Targets to support in SDR Mode.

Note:

*In the HDR Modes, Targets are not required to support any of the CCC Commands listed in **Table 16**. Each manufacturer is free to choose to support any, all, or none of them in each supported HDR Mode.*

As detailed below, there are two main kinds of CCC:

- **Broadcast CCCs:**
  - Are addressed to all I3C Basic Target Devices on the I3C Basic Bus.
  - All Broadcast CCCs are Write CCCs.
- **Direct CCCs:**
  - Are typically directed to a single, specifically-addressed Target on the I3C Basic Bus, selected via its Dynamic Address. For some Direct Write CCCs a Group Address may also be used, per ***Section 4.3.7.4***.
  - If the Direct CCC definition requires a Target response, then only the single, specifically-addressed Target will reply.
  - There are three types of Direct CCCs: Read, Write, and Read/Write.

4.3.7.1 CCC Command Formats

In SDR Mode, the CCC Command always starts with the I3C Basic Broadcast Address, 7'h7E. That is, after a START or Repeated START, the Address field of a CCC Command shall always contain the value 7'h7E and the RnW bitfield shall always contain the value 1'b0 (Write).

All I3C Basic Targets shall recognize:

- The 7'h7E Broadcast Address,
- Their own Dynamic Address (once it has been assigned), and
- Optionally supported Group Addresses (once they have been assigned)

Note:

*The 7'h7E value of I3C's Broadcast Address is selected to avoid collision with any Legacy I^2C Devices that may potentially be present on the I3C Basic Bus: Per the I^2C specification (**Section 3.1.2** of **[NXP01]**), no I^2C Device will respond to the Address 7'h7E.*

Note:

*Formats for HDR Command Frames are different from the SDR Mode CCC Command format described in this Section, and are specified separately for the HDR Modes (see **Section 6.1.3**).*

CCC Commands are always available in SDR Mode and some CCC Commands are available in the HDR Modes, per ***Table 16***. Generally, CCCs that assign Dynamic Addresses, alter I3C Basic Bus Modes, or transfer the I3C Basic Bus Controller Role are not allowed within HDR Modes.

The I3C Basic Controller shall issue a CCC Command both before and after I3C Basic Dynamic Addresses are assigned.

There are four categories of CCC Command:

1. **Broadcast Write:** A Broadcast Write CCC is seen by all I3C Basic Targets. All Targets shall inspect every received Broadcast command, even if the Target then ignores the Broadcast command.

   Every Broadcast Write CCC Command ends with a Repeated START or a STOP.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Note:

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

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

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

Note:

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

3. **Direct Write:** A Direct Write CCC is directed to one or more specific I3C Basic Targets, selected by the Target Dynamic Address(es). This CCC type is only permitted to Write. Generally, only legacy I3C Basic v1.0 CCCs will be of this type.

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

4. **Direct Read:** A Direct Read CCC reads data from one or more specific I3C Basic Targets, one Target at a time, selected by the Target Dynamic Address(es). This CCC type is only permitted to Read. Generally, only legacy I3C Basic v1.0 CCCs will be of this type.

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

All CCC Commands share the same general Frame format, which is shown in ***Figure 19*** for Broadcast CCCs, and in ***Figure 20*** for Direct CCCs. Each CCC Command has a unique Command Code. The fields within this Frame format are detailed in ***Table 15***.

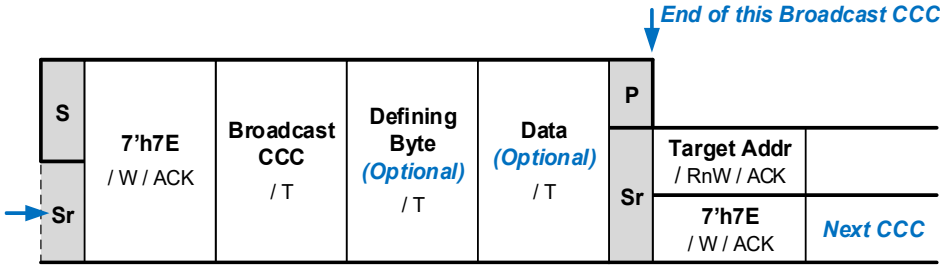


Figure 19 CCC Broadcast General Frame Format

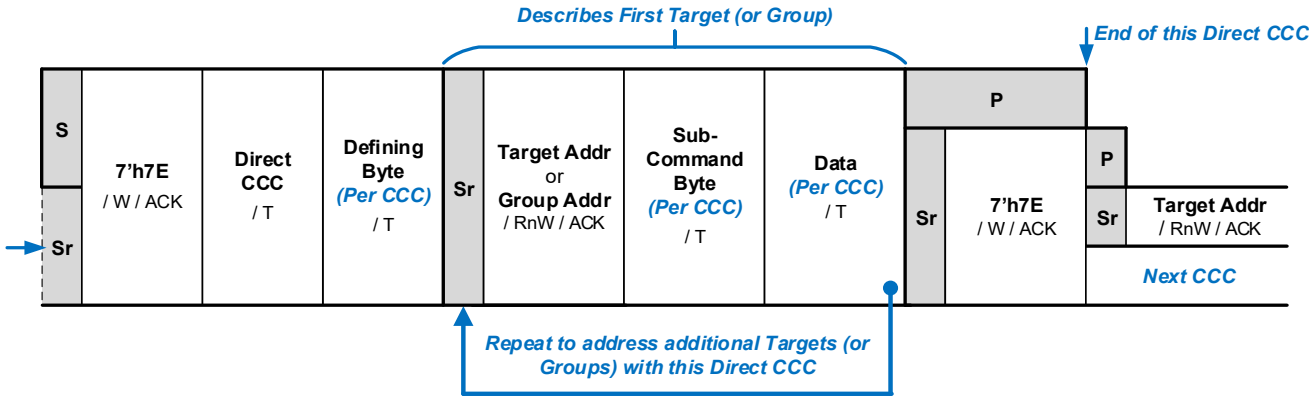


Figure 20 CCC Direct General Frame Format

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

2189 **Table 15 CCC Frame Field Definitions**

Field	Definition	
S or **Sr**	A CCC always begins with either START or Repeated START.	
I3C Basic **Address Header**	This field has three parts:	
	7'h7E	The CCC Frame starts with the global Broadcast Address, so that all I3C Basic Targets on the Bus will see the CCC Code that follows.
	W	The RnW Bit is set to value 1'b0, indicating that the Controller is writing a Message to the Targets. This Message always includes the CCC Code, and may optionally include further Data, depending upon the value of the CCC Code.
	ACK	Acknowledgement (SDA driven Low) by all I3C Basic Targets.
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 4.3.7.3***.	
Defining Byte / T *(Per CCC)*	This field is used with Broadcast CCC Messages, and for some Direct Read/Write CCC Messages. It is followed by a T-Bit. ***Section 4.3.7.3*** specifies, for each defined CCC Code, the use of a Defining Byte as a selector or Sub-Command, extending CCC capability or an action associated with the CCC. When the Defining Byte is used for a Direct CCC, it applies to all subsequent Direct CCC Messages (i.e., segments) until the Controller sends a new Direct CCC. For some Direct CCCs, a Defining Byte is optional; for others, it is always required.	
Sub-Command Byte / T *(Per CCC)*	This field is used with some Direct CCCs. It is followed by a T-Bit. In Direct CCC framing, this field always follows the Target Address (or Group Address) for a specific Direct Write/SET CCC Message (i.e., segment) and applies only to that Message. ***Section 4.3.7.3*** specifies, for each defined CCC Code, whether a Sub-Command Byte is required or optional. ***Note:*** *Sub-Command Bytes are not typically supported for Broadcast CCCs.*	
Data / T *(Per CCC)*	This field is used with Broadcast CCC Messages, and for some Direct Read/Write CCC Messages per Target (or, for some Write CCCs, per Group). It is followed by a T-Bit. ***Section 4.3.7.3*** specifies, for each defined CCC Code, how much Data (if any) appears here. ***Note:*** *For Direct Read/GET CCCs, the Target shall use the T-Bit in the same manner as a typical SDR Read transfer, per **Section 4.3.2.3.4**.*	
Sr / Broadcast Address or **P**	A CCC always ends with either a STOP or a Repeated START and Broadcast Address.	

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.2 Broadcast CCCs vs Direct CCCs

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

- Broadcast Commands are Command Codes 0x00 to 0x7F
- Direct Commands are Command Codes from 0x80 to 0xFE
- Command Code 0xFF is reserved.

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

4.3.7.2.1 End of a CCC Command

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

- A STOP after the Command or Data
- For a Broadcast Command, a Repeated START (for any Address value)
- For a Direct Command, a Repeated START followed by 7'h7E which may be
  - The start of a new CCC
  - Followed by a Repeated START
  - Followed by a STOP, as per ***Figure 20***

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

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

Although not a normal use, the Controller may terminate a Direct CCC Command without addressing any Target.

If the Controller invalidly terminates the data associated with a CCC prematurely, then the Target shall use best efforts to handle the termination and ascertain the proper course of action. That is, the Target may choose to disregard the event with no effect, or the Target may attempt to process the incomplete data.

4.3.7.2.2 Framing Model for Direct CCC Commands

In Direct CCC Commands the Frame is formatted to contain the following, see ***Figure 20*** for reference:

- The Command Code, then
  - One or more blocks of:
    - Repeated STARTs, then
    - The Address of the Target,
    - Followed by Data
- Finally, either a STOP or a Repeated START and 7'h7E.

A single Direct CCC Command may also optionally address more than one Target Device. A Group Address may also be used in the place of a Target Address for those Direct Write or Direct SET CCCs that can also be sent to an assigned Group.

With Direct Read/Write Direct CCC Commands, the Command Code may be followed by data, such as a Defining Byte. This will be explained in the details for each CCC. Further, each Target access may be a Write with zero or more data bytes following from the Controller, or it may be a Read with one or more data bytes following from the Target, if it has ACKed the Read. The CCC Command Definition for each Direct CCC (see ***Section 4.3.7.3***) will explain whether and how it uses the Write and Read.

Each addressed Target Device may either ACK or NACK the Direct CCC Command. For a Read request, the Target Device then returns the requested data in accordance with the SDR Read model, using the T-Bit to end the Read (see ***Section 4.3.2.3.4***).

Note:

Future I3C Basic CCC definitions could be extended so as to include additional data bytes beyond the ones specified in this version of the I3C Basic Specification. For this reason, all I3C Basic Controller and

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Target Devices shall ignore any additional, unrecognized data bytes (i.e., more data bytes than this Specification defines for the given CCC Command).

Target Handling of Unsupported Direct CCCs

If the Target detects an unsupported Direct CCC, then the Target shall generate NACK after its own matched (i.e., assigned) Address in the Direct CCC framing, and shall then wait for STOP or Repeated START.

A CCC is considered unsupported in the following cases:

- If the Target detects a Command Code it does not support
- If the Target detects a Defining Byte it does not support, for a Command Code that it does support
- If the Target receives a matching Address (i.e., Dynamic or Group) with Write when it expects (based on that CCC's definition) a Direct Read or Direct GET
- If the Target receives a matching Dynamic Address with Read when it expects (based on that CCC's definition) a Direct Write or Direct SET

Note:

*Handling an unsupported CCC is not necessarily associated with Error Type TE5 (see **Section 4.3.8.1.6**), which defines how a Target handles illegally formatted CCCs. The effect of a Target generating NACK for unsupported CCCs could seem identical to the I3C Basic Controller.*

Optional Defining Bytes

Starting with I3C v1.1 and I3C Basic v1.1.1, some Direct GET CCCs that earlier I3C Basic versions defined as not having Defining Bytes may now optionally use Defining Bytes. This mechanism allows Target behavior to be extended, or for the Target to indicate alternate status or capabilities, based on the Defining Byte value.

Note:

*This Section does not apply to Direct CCCs that show the Defining Byte field as Required in **Table 16** as well as the CCC's definition in this Specification.*

In these cases, an I3C Basic Target that supports Defining Bytes for such a CCC shall generally treat a Defining Byte value of zero as though the Defining Byte were not present; i.e., the same behavior as if the same CCC had been transmitted with no Defining Byte.

In order to properly distinguish a Direct CCC sent without a Defining Byte from one sent with a Defining Byte, each Target shall track the Command Code value as well as the Defining Byte value (if present), and then respond appropriately when it sees a Repeated START before its own matched Address. In effect, the Target can regard such a Direct CCC either as an 8-bit Command Code, or as a 16-bit Command Code that includes the optional Defining Byte.

If the Target supports a Direct CCC without a Defining Byte, but does not support that same Direct CCC when used with a particular Defining Byte value, then it shall NACK the Direct CCC Command after the Target Address. The Controller shall then assume that the Target does not support that particular Defining Byte for that Direct CCC; however, the Controller shall not infer the presence or absence of any particular Target feature or capability based solely upon the fact that the Target NACKed that particular combination of CCC and Defining Byte. The Controller should also not assume that the Target does not generally support the Direct CCC.

Note:

The Controller is responsible for determining whether the Target supports any given Defining Bytes for a given CCC. Depending on the particular CCC, this can usually be determined by checking other status bits or capabilities advertised by the Target, such as by reading a value returned from a Direct GET CCC without a Defining Byte.

The Controller must also determine whether the Target supports Defining Bytes at all (i.e., whether the Target complies with v1.1 or higher of the I3C Basic Specification). If the Controller attempts to send a Direct CCC with a Defining Byte to a Target that does not comply with I3C Basic v1.1 or higher, then the results may be undefined.

Copyright © 2016–2025 MIPI Alliance, Inc.

4.3.7.2.3 Retry Model for Direct GET CCC Commands

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

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

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

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

1. The Target shall NACK its Address.
2. The Controller shall then emit a Repeated START, followed by the Target Address. Note that this is a second transmission of the Target Address. This gives the Target extra time to prepare its response. As a further measure to give the Target even more time to prepare its response, the Controller may optionally also employ a clock delay (per ***Section 6.10***) after the Target's NACK and before the Controller's Repeated START.
3. The Target should respond to the retry attempt in step 2 with an ACK, and then transmit the results requested by the Direct GET CCC.

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

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3 CCC Command Definitions

Table 16 lists all defined I3C Basic Common Command Codes (CCCs). See also ***Table 75*** and ***Table 76***, which specify the HDR Modes in which each CCC can be used. Below ***Table 16***, a sub-section details each defined CCC.

In ***Table 16***:

- The **CCC is Required / Conditional / Optional** column indicates:
  - **R: The CCC is Required.** It shall be supported by all I3C Basic Controllers and by all I3C Basic Targets.
  - **C: The CCC is Conditionally required.** It shall be supported if the I3C Basic Controller or I3C Basic Target meets the "Required If:" condition shown in the Description column.
  - **O: The CCC is Optional.** It may be supported by an I3C Basic Controller or I3C Basic Target at the discretion of the manufacturer.
  - **N**: The CCC is **Not included** in I3C Basic. It shall not be supported by or used by any I3C Basic Devices.

Note:

*Other I3C Controller Devices that comply with the full I3C Specification should not expect to use CCCs indicated as **Not included** with an I3C Basic Target Device, and must properly mitigate any resulting interoperability issues.*

When a non-Required CCC is supported by an I3C Basic Device, it shall be implemented as defined in this Specification.

- The **CCC Framing** columns indicate, for CCC fields **Defining Byte** and **Sub-Command Byte**:
  - **R:** The field is **Required** for this CCC (i.e., the CCC's framing always includes the field).
  - **O:** The field is **Optional** for this CCC (i.e., the CCC's framing allows the field to be either provided, or not provided).
  - **N:** The field is **Not permitted** for this CCC (i.e., the CCC's framing never includes the field).

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

2326

Table 16 I3C Basic Common Command Codes

Command Code	CCC Type	CCC is Required / Conditional / Optional	CCC Framing		Command Name	Detailed in Section	Description
			Defining Byte	Sub-Command Byte			
0x00	Broadcast	R	N	N	**ENEC** Enable Events Command	***4.3.7.3.1***	Enable Target event driven interrupts Default State: On
0x01	Broadcast	R	N	N	**DISEC** Disable Events Command	***4.3.7.3.1***	Disable Target event driven interrupts Default State: Off
0x02	Broadcast	C[1]	N	N	**ENTAS0** Enter Activity State 0	***4.3.7.3.2***	Set Activity State 0 (normal operation). Default State: On **Required If:** Target supports any other ENTASx CCC(s) per ***Section 4.3.7.3.2***.
0x03	Broadcast	O[1]	N	N	**ENTAS1** Enter Activity State 1	***4.3.7.3.2***	Set Activity State 1 Default State: Off
0x04	Broadcast	O[1]	N	N	**ENTAS2** Enter Activity State 2	***4.3.7.3.2***	Set Activity State 2 Default State: Off
0x05	Broadcast	O[1]	N	N	**ENTAS3** Enter Activity State 3	***4.3.7.3.2***	Set Activity State 3 Default State: Off
0x06	Broadcast	R	N	N	**RSTDAA** Reset Dynamic Address Assignment	***4.3.7.3.3***	Forget current Dynamic Address and wait for new assignment
0x07	Broadcast	C[9]	N	N	**ENTDAA** Enter Dynamic Address Assignment	***4.3.7.3.4***	Controller has started the Dynamic Address Assignment procedure Per ***Section 4.3.4.2***, a Target that supports Dynamic Address Assignment shall participate unless it already has an Address assigned. **Required If:** Target does not have an assigned Static Address, or supports Hot-Join per ***Section 4.3.5***.
0x08	Broadcast	C	N	N	**DEFTGTS** Define List of Targets	***4.3.7.3.7***	Controller defines Dynamic Address, DCR Type, and Static Address (or 0) per Target **Required If:** There are any Secondary Controllers per ***Section 6.5***.
0x09	Broadcast	R[6]	N	N	**SETMWL** Set Max Write Length	***4.3.7.3.5***	Maximum write length in a single command
0x0A	Broadcast	R[7]	N	N	**SETMRL** Set Max Read Length	***4.3.7.3.6***	Maximum read length in a single command
0x0B	Broadcast	O	N	N	**ENTTM** Enter Test Mode	***4.3.7.3.8***	Controller has entered Test Mode
0x0C	Broadcast	O	N	N	**SETBUSCON** Set Bus Context	***4.3.7.3.27***	Controller specifies a higher-level protocol and/or I3C Basic specification version that the Bus will use

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Command Code	CCC Type	CCC is Required / Conditional / Optional	CCC Framing		Command Name	Detailed in Section	Description
			Defining Byte	Sub-Command Byte			
0x0D – 0x11	–	*N/A*	–	–	*MIPI Reserved*	–	*Reserved for future use by MIPI Alliance*
0x12	Broadcast	C	R	N	**ENDXFER** Data Transfer Ending Procedure Control	***4.3.7.3.22***	Framework for Controllers and Targets to exchange set-up parameters for ending data transfers in HDR Modes **Required If:** Target supports HDR-DDR Mode (see ***Section 6.2***), or HDR-Ternary Flow Control (Not included in I3C Basic), or Monitoring Devices Early Termination (Not included in I3C Basic).
0x13 – 0x1E	–	*N/A*	–	–	*MIPI Reserved*	–	*Reserved for future use by MIPI Alliance*
0x1F	N/A	C	–	–	**Reserved** For use in special CCC flows for HDR Modes only.	***6.1.3.1***	Dummy command code only; for use during CCC flows in HDR Modes, to signal the end of CCC modality and return to generic HDR transaction **Required If:** Target supports CCC flows in HDR Modes (e.g., HDR-DDR).
0x20	Broadcast	C[3]	N	N	**ENTHDR0** Enter HDR Mode 0	***4.3.7.3.9***	Controller has entered HDR – DDR Mode **Required If:** Target supports HDR Mode 0 per ***Section 4.3.10***.
0x21	Broadcast	N	N	N	**ENTHDR1** Enter HDR Mode 1	***4.3.7.3.9***	Controller has entered HDR – TSP Mode (Not supported in I3C Basic) This HDR Mode is not included in the I3C Basic Specification.
0x22	Broadcast	N	N	N	**ENTHDR2** Enter HDR Mode 2	***4.3.7.3.9***	Controller has entered HDR – TSL Mode (Not supported in I3C Basic) This HDR Mode is not included in the I3C Basic Specification.
0x23	Broadcast	C[3]	N	N	**ENTHDR3** Enter HDR Mode 3	***4.3.7.3.9***	Controller has entered HDR – BT Mode **Required If:** Target supports HDR Mode 3 per ***Section 4.3.10***.
0x24	Broadcast	N	N	N	**ENTHDR4** Enter HDR Mode 4	–	Controller has entered HDR – Future This HDR Mode is not included in the I3C Basic Specification, and is not yet defined in the full I3C Specification.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Command Code	CCC Type	CCC is Required / Conditional / Optional	CCC Framing		Command Name	Detailed in Section	Description
			Defining Byte	Sub-Command Byte			
0x25	Broadcast	N	N	N	**ENTHDR5** Enter HDR Mode 5	–	Controller has entered HDR – Future This HDR Mode is not included in the I3C Basic Specification, and is not yet defined in the full I3C Specification.
0x26	Broadcast	N	N	N	**ENTHDR6** Enter HDR Mode 6	–	Controller has entered HDR – Future This HDR Mode is not included in the I3C Basic Specification, and is not yet defined in the full I3C Specification.
0x27	Broadcast	N	N	N	**ENTHDR7** Enter HDR Mode 7	–	Controller has entered HDR – Future This HDR Mode is not included in the I3C Basic Specification, and is not yet defined in the full I3C Specification.
0x28	Broadcast	C	N	R	**SETXTIME** Exchange Timing Information	***6.6.3.3***	Framework for exchanging event timing information **Required If:** Target supports any Timing Control mode per ***Section 6.6***.
0x29	Broadcast	O	N	N	**SETAASA** Set All Addresses to Static Addresses	***4.3.7.3.21***	Controller tells every Target with a Static Address to use it as the Dynamic Address
0x2A	Broadcast	R	R	N	**RSTACT** Target Reset Action	***0***	Configure and query Target Reset action and timing
0x2B	Broadcast	C	N	N	**DEFGRPA** Define List of Group Address	***4.3.7.3.26***	Controller tells Secondary Controllers details about an indicated Group Address **Required If:** Device is a Secondary Controller and supports Group Addressing per ***Section 4.3.4.4***
0x2C	Broadcast	C	N	N	**RSTGRPA** Reset Group Address	***4.3.7.3.25***	Controller removes a Target from an indicated Group Address by resetting the assigned Group Address **Required If:** Target supports Group Addressing per ***Section 4.3.4.4***
0x2D	Broadcast	C	R	N	**MLANE** Multi-Lane Data Transfer Control	***6.7.3.6***	Control a Multi-Lane Data Transfer **Required If:** Target supports Multi-Lane per ***Section 6.7***
0x2E – 0x48	–	*N/A*	–	–	*MIPI I3C WG Reserved*	–	*Reserved for future use by MIPI Alliance I3C WG*
0x49 – 0x4C	*Broadcast*	*N/A*	–	–	*MIPI Camera WG Reserved – Broadcast CCCs*	–	*Reserved for future use by MIPI Alliance Camera Working Group*
0x4D – 0x57	*Broadcast*	*N/A*	–	–	*MIPI Reserved – Broadcast CCCs*	–	*Reserved for future use by other MIPI Working Groups*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Command Code	CCC Type	CCC is Required / Conditional / Optional	CCC Framing		Command Name	Detailed in Section	Description
			Defining Byte	Sub-Command Byte			
0x58 – 0x5B	*Broadcast*	*N/A*	–	–	*MIPI Debug WG Reserved – Broadcast CCCs*	–	*Reserved for use by MIPI Alliance Debug Working Group*
0x5C – 0x60	*Broadcast*	*N/A*	–	–	*MIPI RIO WG Reserved – Broadcast CCCs*	–	*Reserved for use by MIPI Alliance RIO Working Group*
0x61 – 0x7F	Broadcast	N/A	–	–	Vendor / Standards Extension – Broadcast CCCs	–	For Vendor or Standards[10] use
0x80	Direct	R	N	N	**ENEC** Enable Events Command	***4.3.7.3.1***	Enable Target event driven interrupts Default State: On
0x81	Direct	R	N	N	**DISEC** Disable Events Command	***4.3.7.3.1***	Disable Target event driven interrupts Default State: Off
0x82	Direct	C[1]	N	N	**ENTAS0** Enter Activity State 0	***4.3.7.3.2***	Set Activity Mode to State 0 (normal operation). Default State: On **Required If:** Target supports any other ENTASx CCC per ***Section 4.3.7.3.2***.
0x83	Direct	O[1]	N	N	**ENTAS1** Enter Activity State 1	***4.3.7.3.2***	Set Activity State 1 Default State: Off
0x84	Direct	O[1]	N	N	**ENTAS2** Enter Activity State 2	***4.3.7.3.2***	Set Activity State 2 Default State: Off
0x85	Direct	O[1]	N	N	**ENTAS3** Enter Activity State 3	***4.3.7.3.2***	Set Activity State 3 Default State: Off
0x86	Direct	R	N	N	***DEPRECATED CCC*:** **RSTDAA** Direct Reset Dynamic Address Assignment	***4.3.7.3.3***	v1.1 Controllers shall not use v1.0 Controllers should not use v1.1 Targets shall NACK
0x87	Direct Set	O	N	N	**SETDASA** Set Dynamic Address from Static Address	***4.3.7.3.10***	Controller assigns a Dynamic Address to a Target with a known Static Address
0x88	Direct Set	C[9]	N	N	**SETNEWDA** Set New Dynamic Address	***4.3.7.3.11***	Controller assigns a new Dynamic Address to any I3C Basic Target with an existing Dynamic Address **Required If:** ENTDAA is supported.
0x89	Direct Set	R[2]	N	N	**SETMWL** Set Max Write Length	***4.3.7.3.5***	Maximum write length in a single command
0x8A	Direct Set	R[2]	N	N	**SETMRL** Set Max Read Length	***4.3.7.3.6***	Maximum read length in a single command
0x8B	Direct Get	R[2]	N	N	**GETMWL** Get Max Write Length	***4.3.7.3.5***	Get Target's maximum possible write length
0x8C	Direct Get	R[2]	N	N	**GETMRL** Get Max Read Length	***4.3.7.3.6***	Get Target's maximum possible read length

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Command Code	CCC Type	CCC is Required / Conditional / Optional	CCC Framing		Command Name	Detailed in Section	Description
			Defining Byte	Sub-Command Byte			
0x8D	Direct Get	C	N	N	**GETPID** Get Provisioned ID	***4.3.7.3.12***	Get Target's Provisioned ID **Required If:** ENTDAA is supported.
0x8E	Direct Get	C	N	N	**GETBCR** Get Bus Characteristics Register	***4.3.7.3.13***	Get Target's Bus Characteristic Register (BCR) **Required If:** ENTDAA is supported.
0x8F	Direct Get	C	N	N	**GETDCR** Get Device Characteristics Register	***4.3.7.3.14***	Get a Device's Device Characteristics Register (DCR) **Required If:** ENTDAA is supported.
0x90	Direct Get	R	O	N	**GETSTATUS** Get Device Status	***4.3.7.3.15***	Get a Device's operating status
0x91	Direct Get	C	N	N	**GETACCCR** Get Accept Controller Role	***4.3.7.3.16***	Active Controller is passing the Bus Controller Role to a Secondary Controller and confirming its acceptance. **Required If:** There are Secondary Controllers per ***Section 6.5***.
0x92	Direct	C	R	N	**ENDXFER** Data Transfer Ending Procedure Control	***4.3.7.3.22***	Framework for Controllers and Targets to exchange set-up parameters for ending data transfers in supported HDR Modes **Required If:** Target supports HDR-DDR Mode (see ***Section 6.2***), or HDR-Ternary Flow Control (Not supported in I3C Basic), or Monitoring Devices Early Termination (Not included in I3C Basic).
0x93	Direct Set	O	N	N	**SETBRGTGT** Set Bridge Targets	***4.3.7.3.17***	Controller tells Bridge (to/from I^2C, SPI, UART, etc.) what endpoints it is talking to (by Dynamic Address and type/ID)
0x94	Direct Get	C[4]	O	N	**GETMXDS** Get Max Data Speed	***4.3.7.3.18***	Controller asks Target for its SDR Mode maximum Read and Write data speeds (& optionally maximum Read Turnaround time) **Required If:** Target has BCR Bit[0] set to 1'b1 per ***Section 4.3.1.2.1***.
0x95	Direct Get	C[5, 8]	O	N	**GETCAPS** *(formerly **GETHDRCAPS**)* Get Optional Feature Capabilities	***4.3.7.3.19***	Controller asks Target what optional capabilities it supports **Required If:** • An I3C Basic v1.0 Target supports HDR Modes per ***Section 4.3.10***. • The Target's I3C Basic version is v1.1 or later.
0x96	Direct	O	N	N	**SETROUTE** Set Route	***4.3.7.3.20***	Controller tells Routing Device what Route(s) to enable

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Command Code	CCC Type	CCC is Required / Conditional / Optional	CCC Framing		Command Name	Detailed in Section	Description
			Defining Byte	Sub-Command Byte			
0x97	Direct	N	R	O	D2DXFER Device to Device(s) Tunneling Control	***6.9***	This CCC is not included in the I3C Basic Specification.
0x98	Direct	C	N	R	**SETXTIME** Set Exchange Timing Information	***6.6.3.3***	Framework for exchanging event timing information **Required If:** Target supports any Timing Control Mode per ***Section 6.6***.
0x99	Direct	C	N	N	**GETXTIME** Get Exchange Timing Information	***6.6.3.4***	Framework for exchanging event timing information **Required If:** Target supports any Timing Control Mode per ***Section 6.6***.
0x9A	Direct	R	R	N	**RSTACT** Target Reset Action	***0***	Configure and query Target Reset action and timing
0x9B	Direct	C	N	N	**SETGRPA** Set Group Address	***4.3.7.3.24***	Controller assigns Group Addresses **Required If:** Target supports Group Addressing per ***Section 4.3.4.4***.
0x9C	Direct	C	N	N	**RSTGRPA** Reset Group Address	***4.3.7.3.25***	Controller removes a Target from an indicated Group Address by resetting the assigned Group Address **Required If:** Target supports Group Addressing per ***Section 4.3.4.4***.
0x9D	Direct	C	R	N	**MLANE** Multi-Lane Data Transfer Control	***6.7.3.6***	Control a Multi-Lane Data Transfer **Required If:** Target supports Multi-Lane per ***Section 6.7***.
0x9E – 0xBF	*Direct*	*N/A*	–	–	*MIPI I3C WG Reserved – Direct CCCs*	**–**	*Reserved for future use by MIPI Alliance I3C WG*
0xC0 – 0xC3	*Direct*	*N/A*	–	–	*MIPI Camera WG Reserved – Direct CCCs*	**–**	*Reserved for future use by MIPI Alliance Camera WG*
0xC4 – 0xD6	*Direct*	*N/A*	–	–	*MIPI Reserved – Direct CCCs*	**–**	*Reserved for future use by other MIPI Alliance WGs*
0xD7 – 0xDA	*Direct*	*N/A*	–	–	*MIPI Debug WG Reserved – Direct CCCs*	**–**	*Reserved for use by MIPI Alliance Debug WG*
0xDB – 0xDF	*Direct*	*N/A*	–	–	*MIPI RIO WG Reserved – Direct CCCs*	**–**	*Reserved for use by MIPI Alliance RIO WG*
0xE0 – 0xFE	Direct	N/A	–	–	Vendor / Standards Extension – Direct CCCs	**–**	For Vendor or Standards[10] use
0xFF	–	*N/A*	–	–	*MIPI I3C WG Reserved*	**–**	*Reserved for future use by MIPI Alliance I3C WG*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Command Code	CCC Type	CCC is Required / Conditional / Optional	CCC Framing		Command Name	Detailed in Section	Description
			Defining Byte	Sub-Command Byte			

Note:

1) Target Devices may self-power-manage based on this information.

2) Applies to Target Devices that support a variable limit on the maximum number of data bytes per Message, where that limit may exceed 16 bytes.

3) All I3C Basic compliant Targets shall recognize that the Bus is entering an HDR Mode, and detect the HDR Exit Pattern.

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

5) This CCC shall be supported by Target Devices that support any HDR Mode.

*6) See **Section 4.3.7.3.5 Set/Get Max Write Length (SETMWL/GETMWL)**.*

*7) See **Section 4.3.7.3.6 Set/Get Max Read Length (SETMRL/GETMRL)**.*

8) This CCC shall be supported by I3C Basic Devices that comply with I3C Basic version 1.1 and greater.

*9) A Target Device that will only ever be used on special-purpose I3C Basic Buses that do not require Enter DAA may choose to not implement support for the ENTDAA CCC. (Such special-purposes I3C Basic Buses use static addresses via SETAASA/SETDASA instead of ENTDAA). Such a Target may also choose not to implement support for the SETNEWDA CCC, per **Section 4.3.7.3.11**; in this case, the Target's Static Address effectively becomes its fixed Dynamic Address*

*10) Standards use of Vendor CCC's controlled by Bus Context. (See **Section 4.3.7.3.27** and public registry at https://www.mipi.org/MIPI_I3C_bus_context_byte_values_public.html **[MIPI07]**.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.1 Enable/Disable Target Events Command (ENEC/DISEC)

These four Direct (***Figure 21***) or Broadcast (***Figure 22***) CCCs allows the Controller to control when Target-initiated traffic is allowed (i.e., Enabled) vs. is not allowed (i.e., Disabled) on the I3C Basic Bus. This control governs a Target's attempts to request an Interrupt (ENINT/DISINT), to request the Controller Role (ENCR/DISCR), or to signify a Hot-Join event (ENHJ/DISHJ).

Table 17 Enable/Disable Target Events Command Codes (ENEC/DISEC)

Command	Support	Command Codes		Purpose
		Broadcast	Direct	
ENEC	Required	0x00	0x80	Enable Target Events
DISEC	Required	0x01	0x81	Disable Target Events

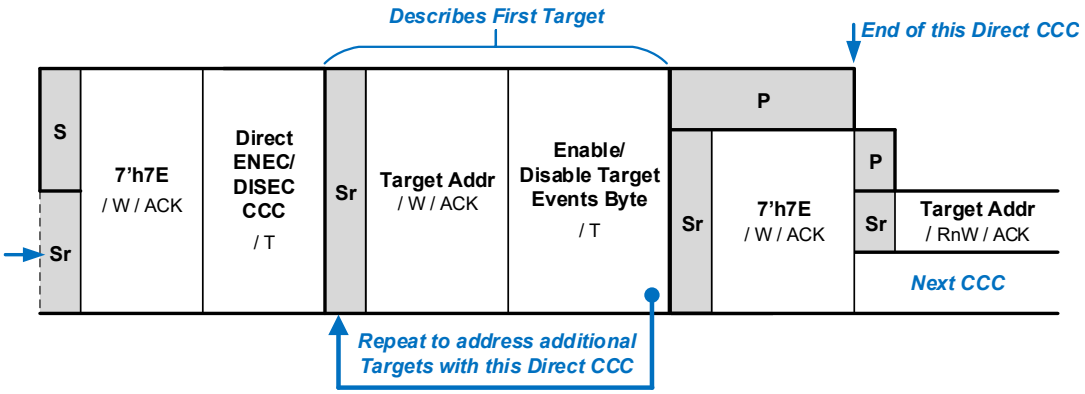


Figure 21 ENEC/DISEC Format 1: Direct

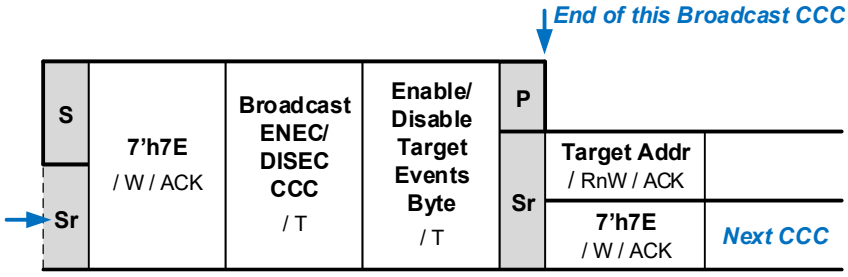


Figure 22 ENEC/DISEC Format 2: Broadcast

Table 18 and ***Table 19*** show the bits to set in the Target Events Byte to enable and disable, respectively, the four supported I3C Basic Target/Secondary Controller event types. Reserved bits are for future MIPI use.

Table 18 Enable Target Events Command Byte Format

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

Table 19 Disable Target Events Command Byte Format

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

The supported I3C Basic Target/Secondary Controller event types are:

- **Target Interrupt Requests:** Enable (ENINT) / Disable (DISINT)

  These bits allow the Controller to control when Target-initiated (i.e., In-Band) Interrupts are (ENINT) and are not (DISINT) allowed on the I3C Basic Bus.

  Note that this capability is enabled by default in Targets that support In-Band Interrupt capability.

- **Controller Role Requests:** Enable (ENCR) / Disable (DISCR)

  These bits allow the Active Controller to control when Controller Role Requests from Secondary Controllers are (ENCR) and are not (DISCR) allowed on the I3C Basic Bus.

  This capability shall be enabled by default in Secondary Controllers that support Controller Role Request capability.

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

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

  This capability shall be enabled by default in Targets that support Hot-Join capability.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

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

These four Direct (***Figure 23***) and four Broadcast (***Figure 24***) CCCs allow the Active Controller to inform one or all Target Devices that it will not be active on the I3C Basic Bus for an approximated amount of time, so that the Target Devices may use a lower power state during that period (see ***Section 4.3.3.3***). There is one Direct CCC and one Broadcast CCC per Activity State.

Note:

These CCCs do not place any requirements on Target Devices; they merely signal the Active Controller's intent to be inactive, i.e., a lower power state. However, all I3C Basic Target Devices shall maintain I3C Basic communications capabilities in all Activity States, regardless of the Active Controller's state.

Table 20 Enter Activity State 0–3 Command Codes (ENTAS0–ENTAS3)

Command	Support	Command Codes		Purpose
		Broadcast	Direct	
ENTAS0	Conditional	0x02	0x82	Enter Activity State 0
ENTAS1	Optional	0x03	0x83	Enter Activity State 1
ENTAS2	Optional	0x04	0x84	Enter Activity State 2
ENTAS3	Optional	0x05	0x85	Enter Activity State 3

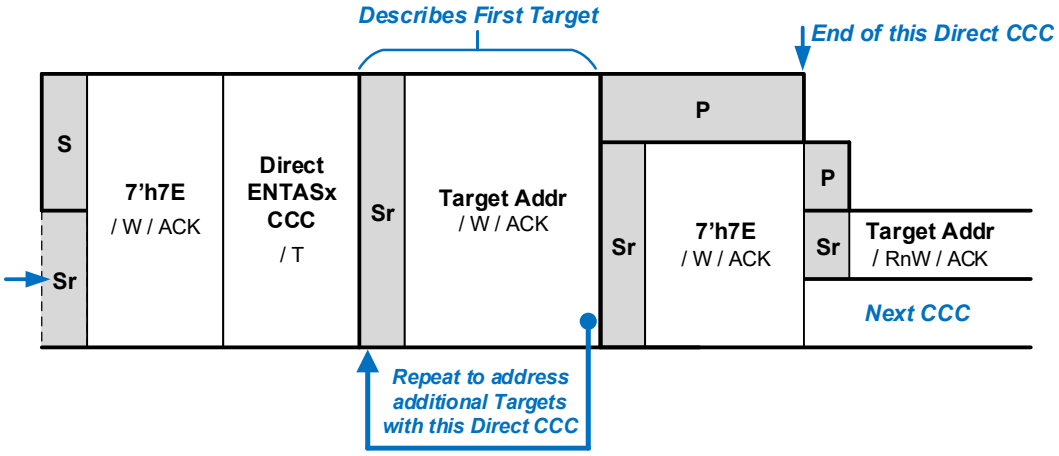


Figure 23 ENTASx Format 1: Direct

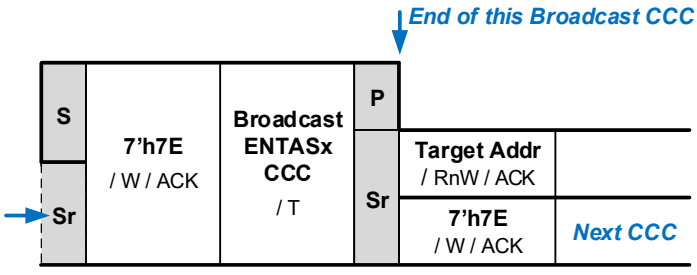


Figure 24 ENTASx Format 2: Broadcast

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

2365 ***Table 21*** gives the expected minimum Bus activity interval for each Activity State.

Table 21 Enter Activity State CCCs (ENTASx)

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.3 Reset Dynamic Address Assignment (RSTDAA)

This Broadcast CCC (***Figure 25***) indicates to all I3C Basic Devices that the Controller requires them to clear/reset their Controller-assigned Dynamic Address. If the I3C Basic Device supports the Group Address feature, then receipt of an RSTDAA CCC shall also reset (clear) all of its assigned Group Addresses. After clearing their Dynamic Address, the Devices shall be ready to participate in an I3C Basic procedure such as Dynamic Address Assignment (***Section 4.3.4***), SETDASA (***Section 4.3.7.3.10***), or SETAASA (***Section 4.3.7.3.21***), or is configured to communicate via I^2C (though without a Spike Filter).

The Command Code for the RSTDAA Broadcast CCC is 0x06. Support for this CCC is required.

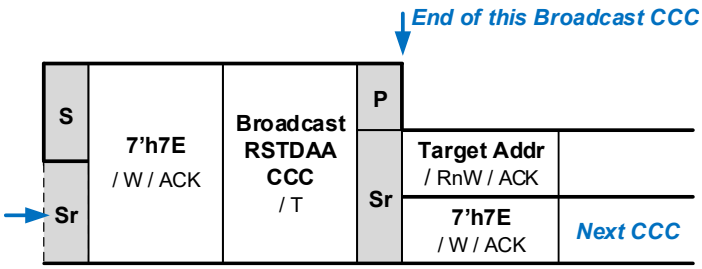


Figure 25 RSTDAA Format

4.3.7.3.4 Enter Dynamic Address Assignment (ENTDAA)

This Broadcast CCC (***Figure 26***) indicates to all I3C Basic Devices that the Controller requires them to enter the Dynamic Address Assignment procedure described in ***Section 4.3.4***. Target Devices that already have a Dynamic Address assigned shall not respond to this command.

The Command Code for the ENTDAA Broadcast CCC is 0x07. Support for this CCC is required, unless this I3C Basic Device has an I^2C Static Address.

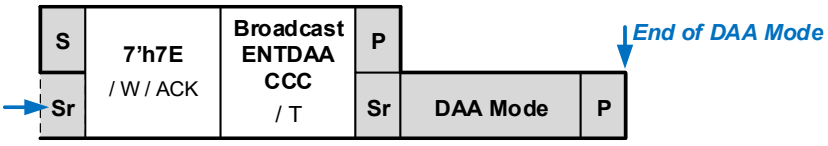


Figure 26 ENTDAA Format

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

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

These Direct and Broadcast CCCs (Direct Set or Get in ***Figure 27***, Broadcast Set in ***Figure 28***) allow the I3C Basic Controller to Set or Get a maximum data write length in bytes for one or all Target Devices. This Max Write Length does not affect data write lengths for Broadcast CCCs, since their data payloads are defined for each CCC. The Set/Get Max Write Length value is transmitted over two bytes, with the most significant byte (MSB) transmitted first. The Target should define the minimum possible value that can be set as Max Write Length.

Note:

If the Controller sends a MWL value (via SETMWL) that the Target does not support, then the Target should leave the MWL value unchanged, so that it would return that value on the next use of GETMWL.

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

Table 22 Set/Get Max Write Length Command Codes (SETMWL/GETMWL)

Command	Support	Command Codes		Purpose
		Broadcast	Direct	
SETMWL	See above	0x09	0x89	Set Max Write Length
GETMWL	See above	–	0x8B	Get Max Write Length

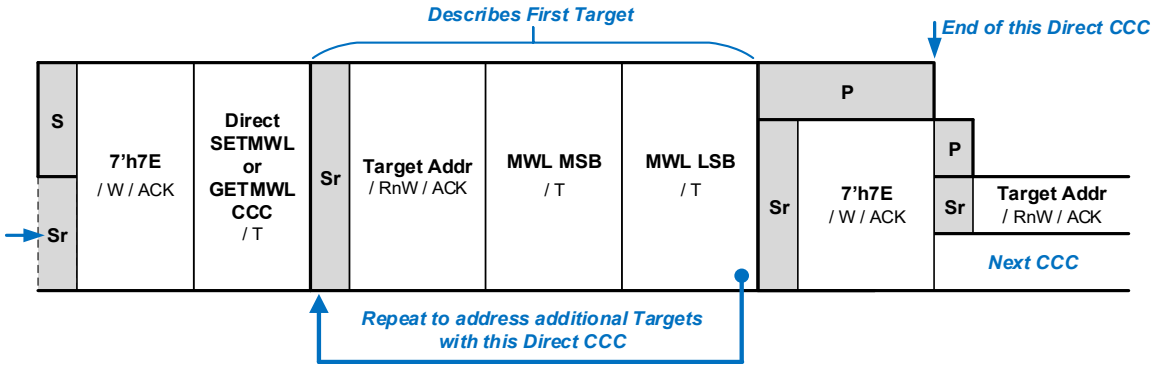


Figure 27 Direct SETMWL/GETMWL Format

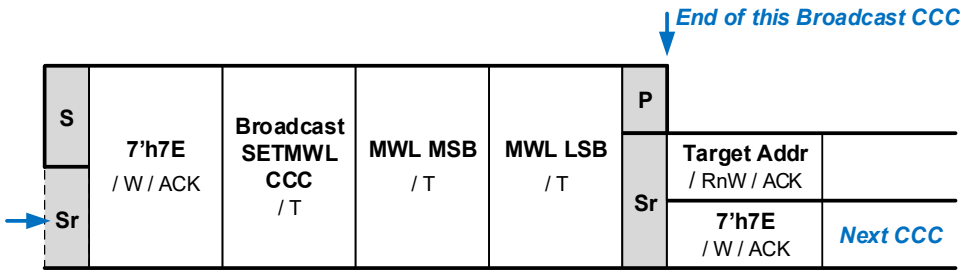


Figure 28 Broadcast SETMWL Format

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

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

These Direct and Broadcast CCCs (Direct Set or Get in ***Figure 29***, Broadcast Set in ***Figure 30***) allow the I3C Basic Controller to Set or Get a maximum data read length, and optionally a maximum IBI payload size.

The Set/Get Max Read Length value is transmitted over the first two bytes, with most significant byte (MSB) transmitted first. The Target should define the minimum possible value that can be set as the Max Read Length.

Note:

If the Controller sends a MRL value (via SETMRL) that the Target does not support, then the Target should leave the MRL value unchanged, so that it would return that value on the next use of GETMRL.

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 of 0 indicates an unlimited payload size. If Timing Control is used, then the minimum IBI payload size is either four bytes or five bytes, per the Timing Control method that is supported. If Timing Control (see ***Section 6.6***) is not used, then the minimum IBI payload size is one (one byte).

This CCC is optional for the Target, with two exceptions:

1. This CCC is required if both (a) any private Read Request Message(s) and/or any extended Read Request CCC(s) implemented by the Target support a variable limit on the maximum number of data bytes that the Target may return per Message, and (b) this limit is greater than 16 bytes.
2. This CCC is required if the Target both (a) supports an IBI Payload (as indicated with BCR bit 2), and (b) will transmit more than one byte of private payload (not counting Timing Control bytes, when Timing Control used).

Table 23 Set/Get Max Read Length Command Codes (SETMRL/GETMRL)

Command	Support	Command Codes		Purpose
		Broadcast	Direct	
SETMRL	See above	0x0A	0x8A	Set Max Read Length
GETMRL	See above	–	0x8C	Get Max Read Length

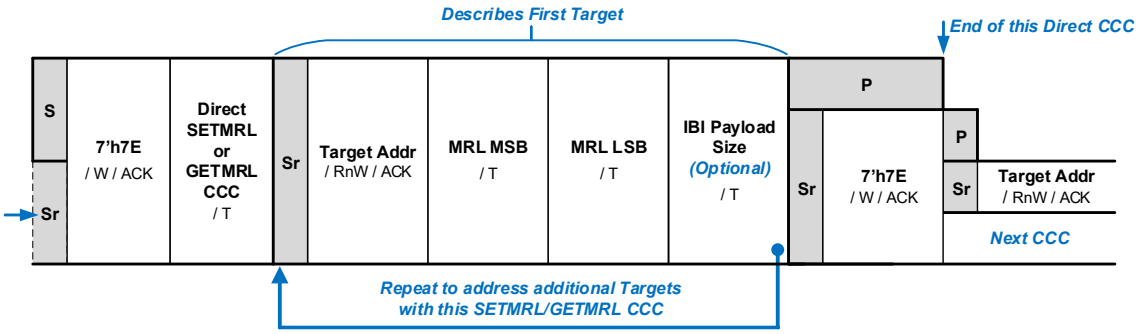


Figure 29 Direct SETMRL/GETMRL Format

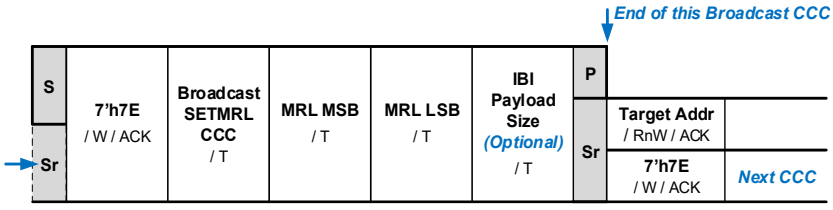


Figure 30 Broadcast SETMRL Format

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.7 Define List of Targets (DEFTGTS)

Note:

In previous versions of the I3C Basic Specification (as well as some versions of the full I3C Specification), this CCC was named "DEFSLVS". The usage and data format for this CCC are unchanged.

This Broadcast CCC (***Figure 31***) is only relevant to Secondary Controllers. This CCC tells Secondary Controller Devices which Targets (and Groups) are present on the I3C Basic Bus, via four consecutive Data Bytes per Target (or Group). First the Active Controller identifies itself by transmitting its own data in the first set of four data bytes, using the value 7'h7E as the Static Address. Then each additional Target or Group on the I3C Basic Bus is represented by a further set of four data bytes.

If the Active Controller supports the Group Address function, then the Active Controller shall send the Secondary Controllers any configured Group Address information, using the DEFTGTS CCC in the same manner used to transmit the Dynamic Address information. Each Group is treated as a virtual I3C Basic Device on the I3C Basic Bus, described by: Group Address, reserved DCR, BCR, and Static Address as defined below. The Group Address bytes may be transmitted within the same DEFTGTS CCC message as the Target Devices.

The Command Code for the DEFTGTS Broadcast CCC is 0x08. All Secondary Controllers are required to support this CCC.

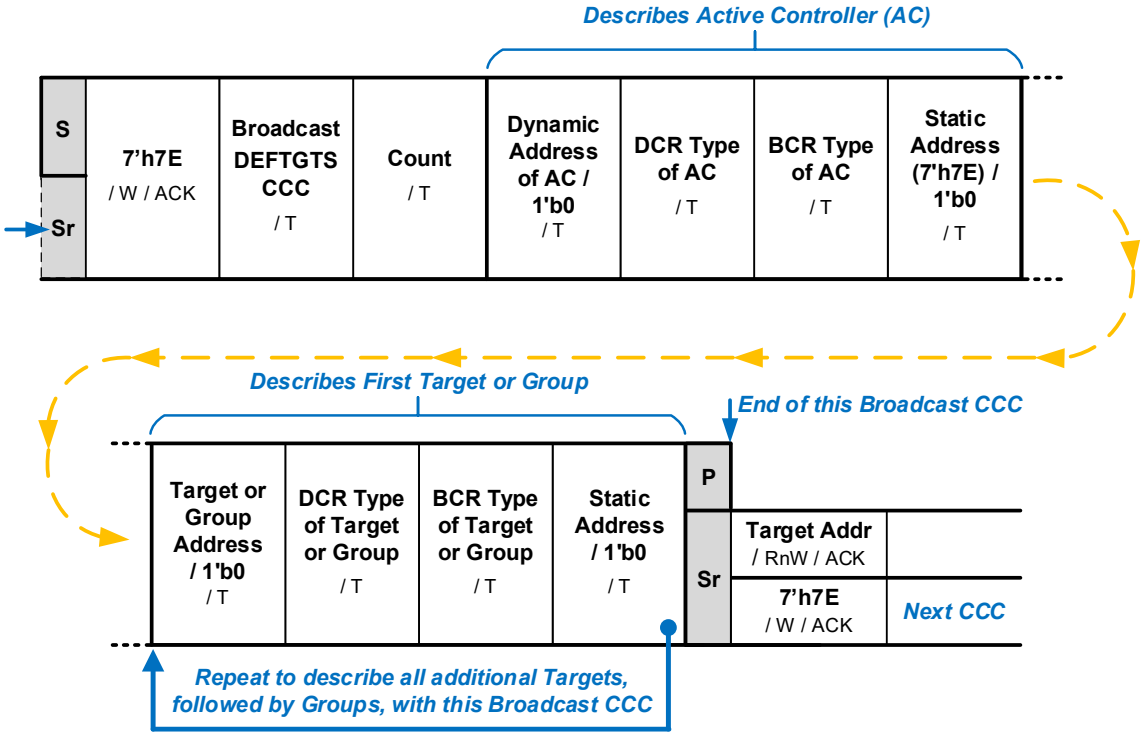


Figure 31 DEFTGTS Format

Count is the number of Targets and Groups present on the I3C Basic Bus. Each Target or Group is represented by a set of four data bytes, as specified in ***Table 24***.

Note:

*For each configured Group Address, the DEFGRPA CCC (see **Section 4.3.7.3.26**) is used to provide more information about the Group, such as the list of I3C Basic Targets that have been assigned to that Group Address.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Table 24 DEFTGTS Data Bytes for Target or Group

Field	Byte	For Target	For Group
Dynamic Address	0	The 7 most significant bits (Bits[7:1]) shall contain the current value of the Target's assigned 7-bit Dynamic Address. The least significant bit (Bit[0]) is filled with the value 1'b0. For a Legacy I^2C Device, this field shall contain the value 7'h00.	The Group Address
DCR Type	1	The Target's Device Characteristics Register value, or the value 0x00 if unknown. For a Legacy I^2C Device, this field shall contain the value of the Device's Legacy Virtual Register (LVR).	This field shall contain the value 8'hFF (all 1's), to indicate that a Group is being described. See ***Section 4.3.1.2.2*** for how to identify a Group value.
BCR Type	2	The Target's Bus Characteristics Register value, or 0x00 if unknown.	Reserved for Group information This field shall not be used as a BCR.
Static Address	3	The Target's original 7-bit static I^2C Address in the 7 most significant bits (Bits[7:1]), with the least significant bit (Bit[0]) filled with the value 1'b0. If no Static Address, then the value shall be 7'h00.	Reserved for Group information

The Active Controller shall not send the DEFTGTS CCC unless at least one Secondary Controller Device is present on the I3C Basic Bus. The Active Controller shall send the DEFTGTS CCC following every Hot-Join event (i.e., after each Dynamic Address Assignment), and as necessary, i.e., when any Dynamic Address or Group Address changes must be sent to Secondary Controller Devices.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.8 Enter Test Mode (ENTTM)

This Broadcast CCC (***Figure 32***) informs all I3C Basic Devices that the Controller is entering a specified Test Mode during manufacturing or Device test. The Enter Test Mode command Frame format includes a byte that specifies which Test Mode to enter. Supporting I3C Basic Devices shall enter the indicated Test Mode upon receipt of the Enter Test Mode CCC. ***Table 25*** lists the defined Test Mode byte values.

The Command Code for the ENTTM Broadcast CCC is 0x0B. Support for this CCC is optional.

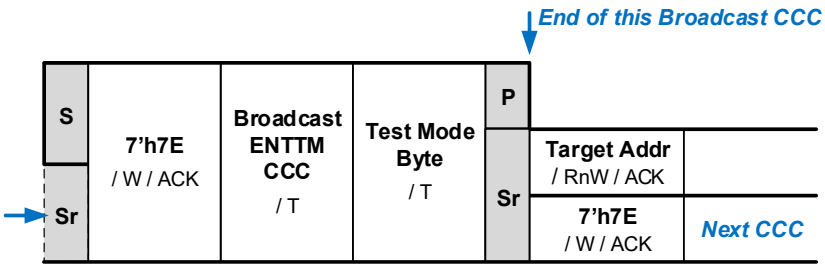


Figure 32 ENTTM Format

Table 25 ENTTM Test Mode Byte Values

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

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

These eight Broadcast CCCs inform all I3C Basic Devices that the Bus is being switched into the indicated HDR Mode. See the indicated Section of this Specification for further details on each HDR Mode.

Support for each CCC shall depend on whether the I3C Basic Device supports the associated HDR Mode (see ***Section 4.3.10***).

Note:

Of the eight possible HDR Modes, only HDR Mode 0 (HDR-DDR) and HDR Mode 3 (HDR-BT) are included in I3C Basic. To gain access to the capabilities of the other HDR Modes, please contact MIPI Alliance.

Table 26 Enter HDR Mode CCCs (ENTHDRx)

Command	Command Code *(All are Broadcast)*	Enter HDR Mode	HDR Mode Name	Supported in I3C Basic?	See Section
ENTHDR0	0x20	HDR Mode 0	HDR-DDR	Y	6.2
ENTHDR1	0x21	HDR Mode 1	HDR-TSP	N	6.3
ENTHDR2	0x22	HDR Mode 2	HDR-TSL	N	6.3
ENTHDR3	0x23	HDR Mode 3	HDR-BT	Y	6.4
ENTHDR4	0x24	HDR Mode 4	HDR Modes 4–7 are Reserved for future definition by MIPI Alliance		
ENTHDR5	0x25	HDR Mode 5			
ENTHDR6	0x26	HDR Mode 6			
ENTHDR7	0x27	HDR Mode 7			

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.10 Set Dynamic Address from Static Address (SETDASA)

This CCC (***Figure 33*** and ***Figure 34*** illustrate the two distinct command formats) allows the Controller to assign a Dynamic Address to one Target using the Target's Static Address. This is faster than the ENTDAA Dynamic Address Assignment procedure (see ***Section 4.3.7.3.4***). The SETDASA CCC should be used before the ENTDAA CCC is used; all Targets without assigned Dynamic Addresses will respond to the ENTDAA CCC.

Note:

*Per **Section 5.1.4**, only the SETNEWDA and RSTDAA CCCs will cause a Target that already has an assigned Dynamic Address to change or lose its address. Targets that support the SETDASA CCC shall not respond to this CCC if a Dynamic Address has already been assigned (i.e., by any of the CCCs that set or assign a Dynamic Address). If a Dynamic Address has already been assigned, then the Target shall NACK this CCC and ignore the command.*

Note:

The SETDASA Command Code is unusual in that it addresses the desired I3C Basic Target by its I^2C Static Address, rather than by its assigned Dynamic Address. As a result, this CCC can only be sent to a Target that has an I^2C Static Address.

*Per **Section 4.3.4.2**, SETDASA is intended as an optimization. An I3C Basic Target that supports SETDASA and does not support ENTDAA cannot be relied on to be assigned a Dynamic Address on a generic I3C Basic Bus, and also will not support Hot-Join (per **Section 4.3.5**). Use of SETDASA requires the I3C Basic Controller to have prior knowledge of the I3C Basic Target. Dynamic Address Assignment with ENTDAA is generally recommended for most use cases.*

The newly assigned Address is transmitted in the Dynamic Address Byte shown in ***Figure 33***, where the 7 most significant bits (Bits[7:1]) contain the 7-bit Dynamic Address, and the least significant bit (Bit[0]) is filled with the value 1'b0.

The Command Code for the SETDASA Direct CCC is 0x87. Support for this CCC is optional; however, the I3C Basic Target that supports this CCC must have an I^2C Static Address.

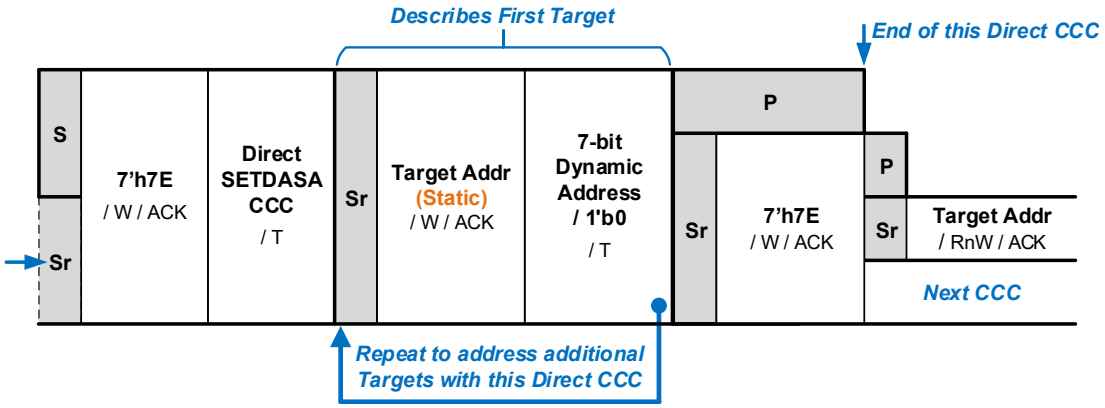


Figure 33 SETDASA Format 1: Primary

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.10.1 SETDASA Minimal Bus Point-to-Point Communication

The SETDASA CCC may also be specially used for simple point-to-point communication in I3C Basic Minimal Bus use cases (per ***Section 4.3.1*** and ***Figure 11***). An I3C Basic Minimal Bus is an I3C Basic Bus with one I3C Basic Controller Device (potentially with reduced functionality), and one I3C Basic Target Device. In this special usage of the SETDASA CCC, the Controller Device both uses the fixed value 7'h01 as the Static Address, and uses the fixed (and reserved) value of 7'h01 as the Dynamic Address (see ***Figure 34***). This special usage of the SETDASA CCC allows for simpler Controller Devices, and optionally simpler Target Devices intended for use specifically in Minimal Bus configurations.

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

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

- One active I3C Basic Target Device that is capable of supporting this special usage of the SETDASA CCC, or
- One or more I3C Basic Target Devices that act strictly as receivers, i.e., that do not use In-Band Interrupt and that will not be sent Read requests. This arrangement permits a single Controller Device to, in effect, Broadcast to the Target Devices.

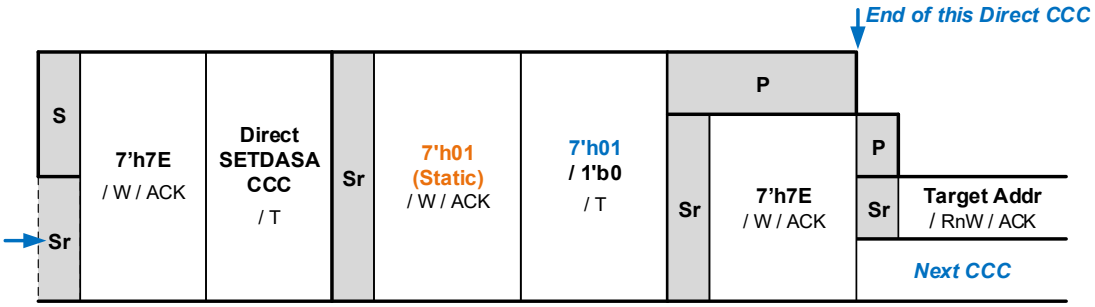


Figure 34 SETDASA Format 2: Point-to-Point

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.11 Set New Dynamic Address (SETNEWDA)

This Direct CCC (***Figure 35***) allows the I3C Basic Controller to assign a new Dynamic Address to one I3C Basic Target or Secondary Controller. In the Dynamic Address field, the 7 most significant bits (Bits[7:1]) contain the 7-bit Dynamic Address, and the least significant bit (Bit[0]) is filled with the value 1'b0.

If an I3C Basic Target (or Secondary Controller) supports this CCC and accepts the new 7-bit Dynamic Address from the Controller via this CCC, then this shall change its current Dynamic Address. After accepting the new Dynamic Address, the Target (or Secondary Controller) shall respond to subsequent transactions that are directly addressed to this new Dynamic Address (per ***Section 4.3.2.1***) and not to the previous Dynamic Address.

Note:

*This CCC may only be used to change the currently assigned Dynamic Address, which must be initially assigned via any supported method (per **Section 4.3.4.2**). If an I3C Basic Target (or Secondary Controller) has not received its Dynamic Address via the ENTDAA CCC (i.e., Dynamic Address Assignment) or via the SETAASA and/or SETDASA CCCs (i.e., from an I²C Static Address), then it cannot respond to the SETNEWDA CCC.*

*In version 1.0 of the I3C Basic Specification, all I3C Basic Targets were required to support this CCC. Starting with I3C Basic version 1.1.1, this CCC is now required only for Targets that also implement support for Dynamic Address Assignment via the ENTDAA CCC (see **Section 4.3.7.3.4**). This CCC is now **optional** for Targets that have I²C Static Addresses and do not support the ENTDAA CCC; therefore, such Targets must only receive Dynamic Addresses via the SETAASA and/or SETDASA CCCs (per **Section 4.3.4.2**). For most Targets with I²C Static Addresses and that do not support the ENTDAA CCC, support for the SETNEWDA CCC is recommended; however, if such a Target does not support the SETNEWDA CCC, then it shall not respond (i.e., shall NACK its current Dynamic Address).*

The Command Code for the SETNEWDA Direct CCC is 0x88. Support for this CCC is required, unless the I3C Basic Target has an I²C Static Address and does not support the ENTDAA CCC.

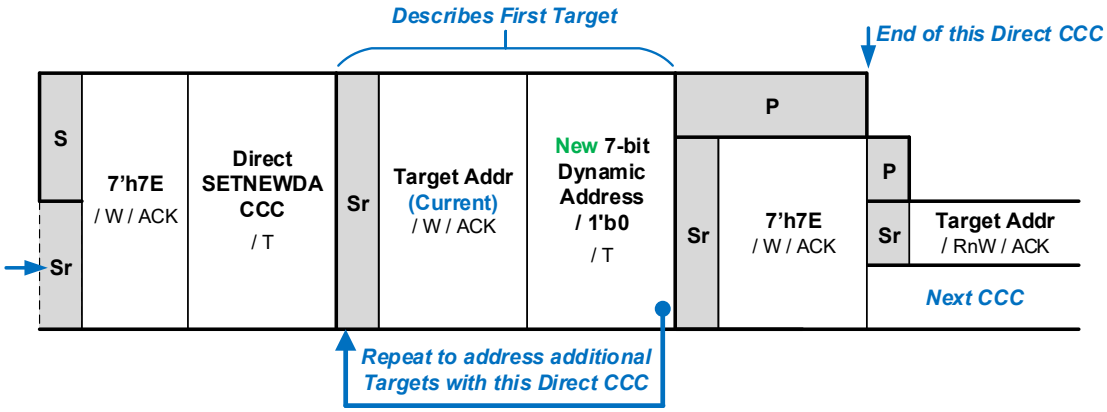


Figure 35 SETNEWDA Format

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.12 Get Provisioned ID (GETPID)

This Direct CCC (see ***Figure 17*** and ***Figure 36***) is a Get request for one I3C Basic Target Device to return its 48-bit Provisioned ID to the Controller, as described in ***Section 4.3.4***. Following transmission of the GETPID CCC, the 48-bit value is transmitted as 6 bytes, with MSB first.

The Command Code for the GETPID Direct CCC is 0x8D. If an I3C Basic Target supports Dynamic Address Assignment with the ENTDAA CCC, then it shall also support this CCC. If an I3C Basic Target does not support Dynamic Address Assignment with the ENTDAA CCC, then support for this CCC is optional.

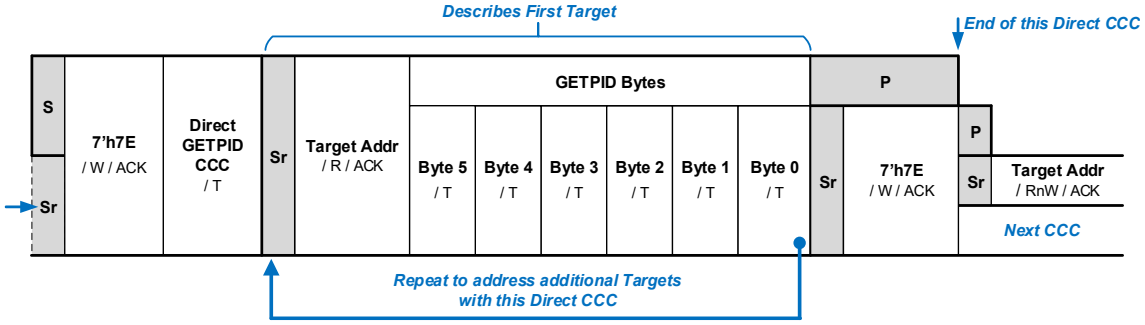


Figure 36 GETPID Format

4.3.7.3.13 Get Bus Characteristics Register (GETBCR)

This Direct CCC (***Figure 37***) is a Get request for one I3C Basic Target Device to return its Bus Characteristics Register (BCR) to the Controller, as described in ***Section 4.3.1.2.1***. The BCR value is transmitted in one byte, with the MSb transmitted first.

The Command Code for the GETBCR Direct CCC is 0x8E. If an I3C Basic Target supports Dynamic Address Assignment with the ENTDAA CCC, then it shall also support this CCC. If an I3C Basic Target does not support Dynamic Address Assignment with the ENTDAA CCC, then support for this CCC is optional.

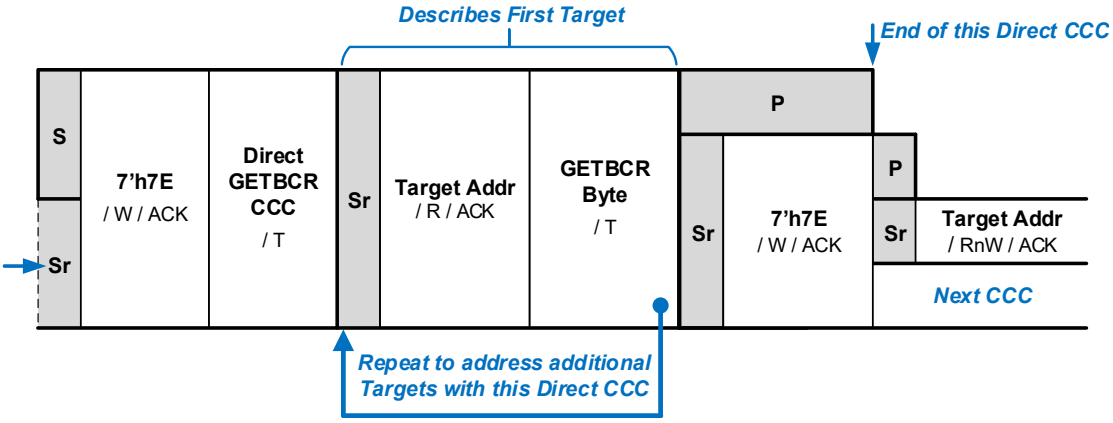


Figure 37 GETBCR Format

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.14 Get Device Characteristics Register (GETDCR)

This Direct CCC (***Figure 38***) is a Get request for one I3C Basic Target Device to return its Device Characteristics Register (DCR) to the Controller, as described in ***Section 4.3.1.2.2***. The DCR value is transmitted in one byte, with the MSb transmitted first.

The Command Code for the GETDCR Direct CCC is 0x8F. If an I3C Basic Target supports Dynamic Address Assignment with the ENTDAA CCC, then it shall also support this CCC. If an I3C Basic Target does not support Dynamic Address Assignment with the ENTDAA CCC, then support for this CCC is optional.

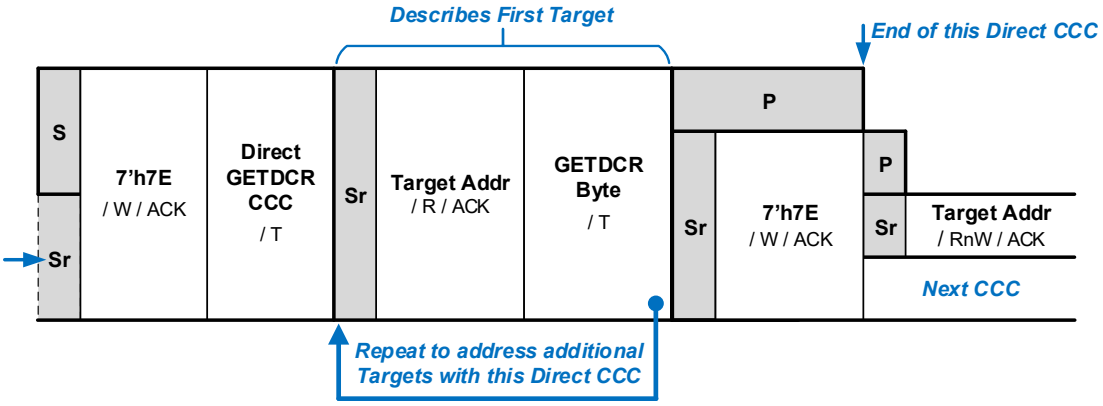


Figure 38 GETDCR Format

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.15 Get Device Status (GETSTATUS)

This Direct CCC is a Get request for one I3C Basic Target Device to return its current Status.

It has two formats:

- GETSTATUS Format 1 (***Figure 39***) returns the two-byte format detailed in ***Table 27***.
- The optional GETSTATUS Format 2 (***Figure 40***) adds a Defining Byte (***Table 28***) and returns a variable number of data bytes. The number of data bytes returned and their format depend upon the Defining Byte value. GETSTATUS Format 2, including its Defining Byte value PRECR, is detailed below.

The Command Code for the GETSTATUS Direct CCC is 0x90. Support for this CCC is required.

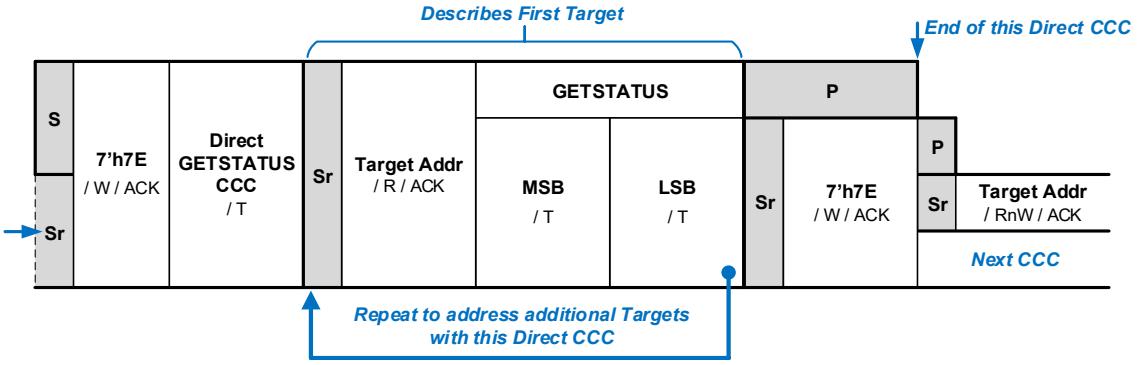


Figure 39 GETSTATUS Format 1

Table 27 GETSTATUS MSB-LSB Format 1

Byte	Bits	Field	Description
MSB	**15:8**	**Vendor Reserved**	Reserved for vendor-specific meaning.
LSB	**7:6**	**Activity Mode**	Contains the two-bit ID of the Target Device's current Activity Mode (i.e., its readiness to support data read of sensor or related information). A Controller-capable Device (i.e., Secondary Controller) must use this field to indicate when it is unable to participate in any of the steps to prepare for Handoff of the Controller Role (see ***Section 6.5.3.1***). If such a Device is currently unable to participate, then it shall return a value of 2'b11. Any other value shall indicate that the Device may be ready to participate. For all other Target Devices, the meanings of the four possible values are not defined by this Specification; instead, they depend upon a private contract between the Controller and each Target.
	5	**Protocol Error**	1'b1: The Target detected a protocol error since the last Status read. The Target might or might not be able to check for such errors. Note that this value self-clears upon every successful completion of a Controller read of the Target's Status.
	4	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG.
	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. This field indicates whether interrupts are pending (i.e., are enqueued and waiting to be sent) and does not depend on whether the Target was previously told to not send interrupts (i.e., if the Controller had sent the DISEC CCC with the 'DISINT' bit set). This value can still be read even when IBI is disabled for the Target with DISEC CCC.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.15.1 GETSTATUS Format 2

The optional GETSTATUS Format 2 (***Figure 40***) adds a Defining Byte (***Table 28***) and returns a variable number of data bytes. The number of data bytes returned and their format depend upon the Defining Byte value.

Support for the GETSTATUS Format 2 CCC is optional, typically depending upon the Device's role, capabilities, or other extended behavior. Before using GETSTATUS Format 2 for a given Target, a Controller shall determine whether that Target supports it. One way is to send the GETCAPS Format 1 CCC (no Defining Byte) and find that in Byte 3 (GETCAP3) of the data that the Target returns, bit 4 contains 1'b1 (see ***Table 63***)

GETSTATUS Format 2 conforms to the CCC Direct General Frame Format (***Figure 20***):

- **If the addressed Target Device does not support the given Defining Byte value for GETSTATUS Format 2:** Then the Target shall NACK its Target Address, thus terminating the Direct CCC.
- **If the addressed Target Device does support the given Defining Byte value for GETSTATUS Format 2:** Then the Target shall ACK its Target Address and return the requested data byte(s). The number of data bytes returned shall depend on the particular Defining Byte and its associated capabilities, as detailed below.
- **If the addressed Target Device is unable to immediately respond to the GETSTATUS Format 2 CCC with the given Defining Byte:** Then, per the Retry Model for Direct GET CCC Commands (***Section 4.3.7.2.3***), the Target shall NACK the first attempt, and then the Controller shall attempt a single retry. If the Controller receives a NACK after a retry, then the Controller shall assume that the Target does not support the GETSTATUS Format 2 CCC with the given Defining Byte, and hence cannot return data describing any capabilities corresponding to that Defining Byte value. In most cases the Controller can proceed as though it had read a value equivalent to all zeros as a default value. However, a NACK after a retry does not guarantee that the Target does not possess any capability or feature set associated with the given Defining Byte value.

A Target that supports the GETSTATUS Format 2 CCC (i.e., GETSTATUS with any optional Defining Bytes) shall return a value of 1'b1 in bit 4 of Byte 3 (GETCAP3), when the GETCAPS CCC is sent without a Defining Byte. A Controller that sees that bit set to 1'b1 will know that the Target is capable of supporting at least one Defining Byte with the GETSTATUS Format 2 CCC. The ACK/NACK method explained above allows the Target to indicate which Defining Byte values it actually supports.

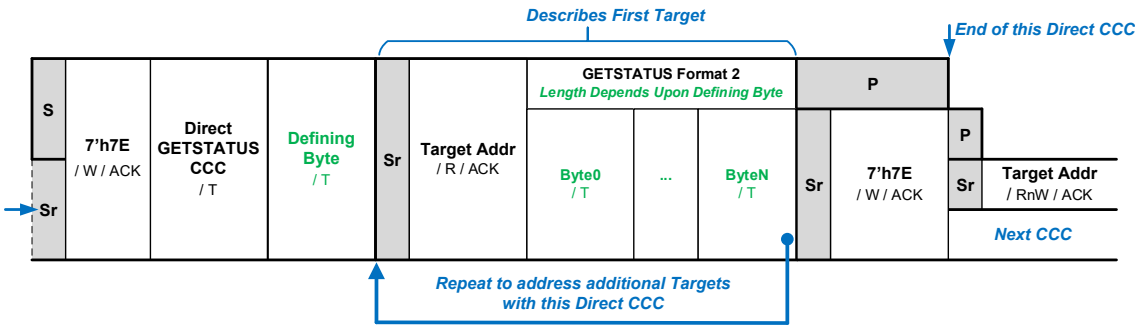


Figure 40 GETSTATUS Format 2

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

2582 **Table 28 GETSTATUS Format 2 Defining Byte Values**

Value	Encoding	Description	Data Bytes Returned
0x00	**TGTSTAT**	Returns Target Status (standard format). Equivalent to GETSTATUS Format 1 (without a Defining Byte).	2
0x01 – 0x90	Reserved	Reserved for future definition by MIPI Alliance I3C WG	–
0x91	**PRECR**	Returns alternate Status format describing a Controller-capable Device (i.e., Secondary Controller) per sub-section below. Recommended if an I3C Basic Device is both compliant with I3C Basic Specification v1.1 or higher, and advertises as Controller-capable (i.e., if the value of the BCR[7:6] bits is 2'b01).	2
0x92 – 0xBF	Reserved	Reserved for future definition by MIPI Alliance I3C WG	—
0xC0 – 0xD6	Reserved	Reserved for future definition by other MIPI Alliance WGs	—
0xD7	**NASTAT**	Returns current status of Debug logic in a device that supports MIPI Debug Over I3C.	—
0xD8	**SCMSTAT**	Returns the status of secure communications logic in a device that supports MIPI Debug Over I3C.	—
0xD9 – 0xDF	Reserved	Reserved for future definition by other MIPI Alliance WGs	—
0xE0 – 0xFE	**Vendor Extensions**	For Vendor use	—
0xFF	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG	—

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.15.2 Get Secondary Controller Status (GETSTATUS Format 2, Defining Byte PRECR)

Note:

This Section describes the GETSTATUS Format 2 CCC when the value of the Defining Byte is 0x91 (PRECR)

Support for the GETSTATUS Format 2 CCC with the PRECR Defining Byte is optional. All Controller-capable Devices should support the GETSTATUS Format 2 CCC with the PRECR Defining Byte while in Target mode, if they might enter such a deep sleep state, or might require significant time to recover and/or re-synchronize their internal state based on Broadcast data sent by the Active Controller.

This Direct CCC (***Figure 41***) allows the Active Controller to query a Controller-capable (i.e., Secondary Controller) I3C Basic Device, to read the current state of the Device (***Table 29***) in order to determine whether it entered a "deep sleep" state during which it might have missed any Broadcast DEFTGTS CCCs or DEFGRPA CCCs sent by the Active Controller, or is currently processing data from these Broadcast CCCs and is unable to safely accept the Controller Role until it has finished processing that data and updated its internal state.

It is anticipated that some classes of Controller-capable Devices operating as a Secondary Controller (i.e., in Target mode) may enter a deep sleep state, and hence would not only miss these Broadcast CCCs while in that state, but would also require a significant amount of time (from the perspective of the Bus and the Active Controller) to process these Broadcast CCCs sent by the Active Controller in an attempt to re-synchronize all Secondary Controller Devices, in preparation for a Controller Role handoff. If such a Device has entered a deep sleep state, then the Device shall indicate this state until it has received at least one Broadcast DEFTGTS CCC from the Active Controller. It is also expected that such a Device will remain active for a sufficient amount of time (i.e., after indicating its current Secondary Controller Status and processing the Broadcast CCC data received from the Active Controller) to ensure that it does not re-enter a deep sleep state which would invalidate any efforts to bring it back online and up to date.

When receiving the GETSTATUS Format 2 CCC with the PRECR Defining Byte, the Device returns its Status using the two-byte format detailed in ***Table 29***.

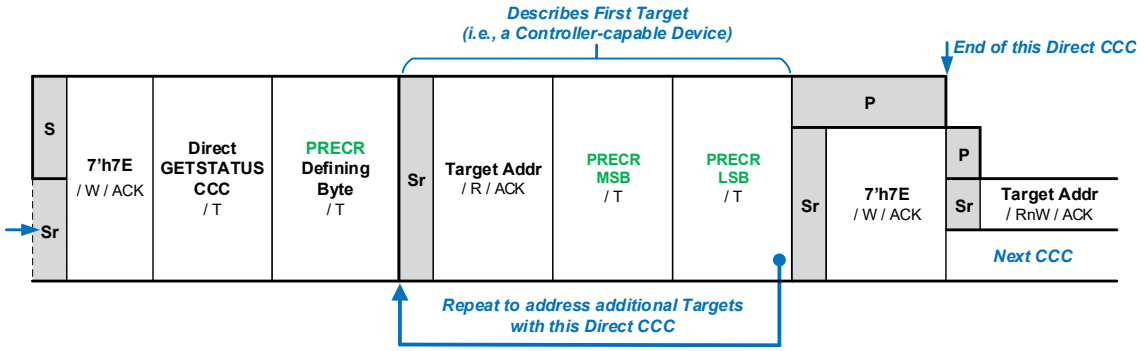
A Device that is not Controller-capable shall NACK its Target Address, to indicate that it does not support the GETSTATUS Format 2 CCC with the PRECR Defining Byte.

It is the responsibility of the Active Controller to determine whether a Controller-capable Device supports the GETSTATUS Format 2 CCC with the PRECR Defining Byte, and whether the deep sleep handling and re-synchronization is necessary before a Controller Role handoff.

Resynchronization: A Secondary Controller that may require re-synchronization due to a returning from a deep sleep state shall indicate this capability when the GETCAPS CCC is sent with a Defining Byte value of CRCAPS (see ***Section 4.3.7.3.19***). A Controller that sees that this capability is indicated knows that it then needs to send the GETSTATUS Format 2 CCC with Defining Byte value PRECR to read the current Secondary Controller Status from this Device, in order to determine its need to re-synchronize from a deep sleep state, or check to see whether it is processing the Broadcast CCC data, before expecting this Device to successfully ACK the GETACCCR CCC as part of the Controller Role handoff.

Additional Time: A Secondary Controller requiring additional time to process data from these Broadcast CCCs sent by the Active Controller shall indicate this capability when the GETCAPS CCC is sent with a Defining Byte value of CRCAPS (see ***Section 4.3.7.3.19***). If the Controller sees that capability is indicated, then it shall send this CCC with Defining Byte value PRECR to poll the Device, in order to frequently check its Status and determine when it has finished processing this data. This Secondary Controller shall indicate its current processing Status, by returning a value of 1'b1 in Bit[1] in the alternate Status two-byte format detailed below, until it has finished processing the data and can accept the Controller Role.

Copyright © 2016–2025 MIPI Alliance, Inc.



2626

Figure 41 GETSTATUS Format 2 with Defining Byte PRECR

2627

Table 29 Status Bytes for GETSTATUS Format 2 with Defining Byte PRECR

Byte	Bits	Field	Description
MSB	**15:8**	**Vendor Reserved**	Reserved for vendor-specific meaning.
LSB	**7:2**	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG.
	1	**Handoff Delay NACK**	This bit indicates whether this Device is currently processing any DEFTGTS CCC (and DEFGRPA CCC, if supported) Broadcasts that may have been sent by the Active Controller. It is expected that this Device might take significant time to process the data in such Broadcasts. **1'b1:** The Active Controller may wait to send GETACCCR CCC to this Device, or should at least be aware that any attempts to send GETACCCR CCC to this Device will be met with a NACK response, until this bit is read again later with a value of 1'b0, to indicate that the Device has finished processing the data in these Broadcasts and has updated its internal state. **1'b0:** This Device is not currently processing Broadcast data, or has finished processing this data; in either case, it can safely accept the Controller Role.
	0	**Deep Sleep Detected**	This bit indicates whether this Device has entered a “deep sleep” state, in which it might have missed any DEFTGTS CCC (and DEFGRPA CCC, if supported) Broadcasts sent by the Active Controller. Consequently, this Device’s internal state of known Target Devices (and known Group Addresses, if applicable) should be considered outdated. **1'b1:** The Active Controller is obligated to send a fresh Broadcast of DEFTGTS CCC (and DEFGRPA CCC, if applicable and supported) to update this Device’s internal state before sending GETACCCR CCC to this Device. **1'b0:** This Device has not entered a “deep sleep” state, or is not capable of doing so.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.16 Get Accept Controller Role (GETACCCR)

This Direct GET CCC (***Figure 42***) is used both to verify a Controller Role Request, if received earlier; and to allow the Active Controller to offer (i.e., to pass) the Controller Role to an I3C Secondary Controller, even if it was not previously requested. The GET is used to confirm acceptance; to do so, the Secondary Controller returns its 7-bit Dynamic Address as shown in ***Figure 42***. The value of the 7-bit Dynamic Address will be the same as the value of the Target Address. If the Secondary Controller does not accept, then it shall NACK the Get request as shown in ***Figure 43***.

A Secondary Controller requesting the Controller Role will gain the Controller Role only if all three of the following steps occur in the indicated order. The Active Controller may offer the Controller Role after preparing for handoff as detailed in ***Section 6.5.3.1***.

1. The Active Controller transmits a GETACCCR command to the Secondary Controller.
2. The Secondary Controller correctly replies with its 7-bit Dynamic Address. The value of the 7-bit Dynamic Address will be the same as the value of the Target Address.

   If the Secondary Controller instead responds with NACK (***Figure 43***), or with an incorrect 7-bit Address (***Figure 44***), then:

   - The Secondary Controller will not acquire the Controller Role
   - The GETACCCR command is canceled, and
   - The Active Controller can then either (a) Retry the CCC, or (b) Simply issue a Repeated START followed by 7'h7E to cancel (***Figure 44***).
3. The Active Controller issues a STOP and then begins the Controller to Controller Handoff Procedure per ***Section 6.5.3.2***.

   If the Active Controller instead issues a Repeated START, then:

   - The Secondary Controller will not gain the Controller Role
   - The GETACCCR command is canceled (***Figure 44***), and
   - The Active Controller can then either (a) Retry the CCC, or (b) End the transaction by providing a STOP.

The Command Code for the GETACCCR Direct CCC is 0x91. If a Controller-capable Device (e.g., a Primary Controller or Secondary Controller) intends to pass the Controller Role to another Controller-capable Device, then it shall support this CCC. All Secondary Controller Devices (i.e., those that initialize as Secondary Controllers) should support this CCC.

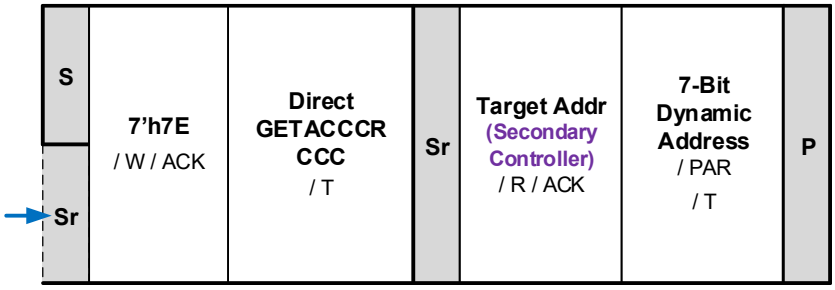
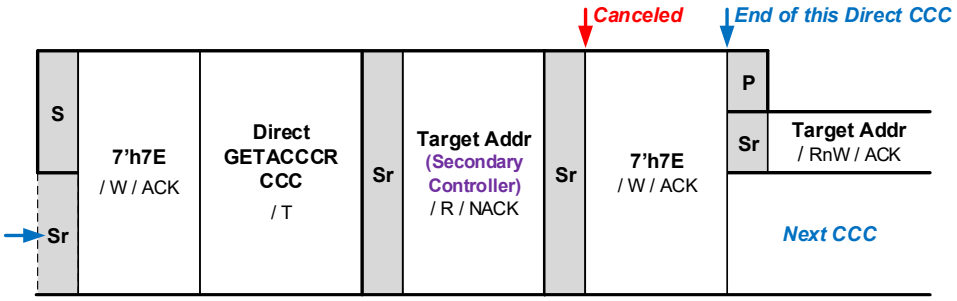


Figure 42 GETACCCR Format 1: Accepted

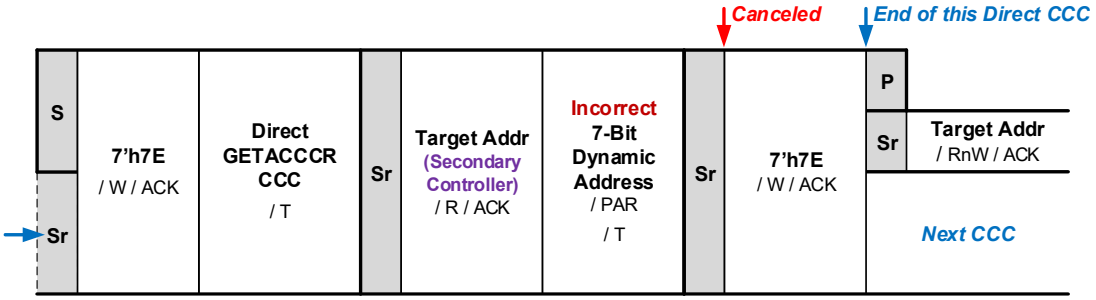
Note:

The value of the 7-Bit Dynamic Address will be the same as the value of the Target Address, and PAR is odd parity: ~XOR(Target Address[7:1]).



2661

Figure 43 GETACCCR Format 2: Not Accepted



2662

Figure 44 GETACCCR Format 3: Incorrect Cancel

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.17 Set Bridge Targets (SETBRGTGT)

This Direct CCC (***Figure 45***) is only relevant to Bridge Devices. An I3C Basic Controller shall only send this CCC to known Bridge Devices that accept it.

Note:

*For additional information on Bridge Devices, refer to the MIPI Alliance I3C Application Note: Virtual Devices and Virtual Targets **[MIPI10]**.*

The Command Code for the SETBRGTGT Direct CCC is 0x93. Support for this CCC is optional.

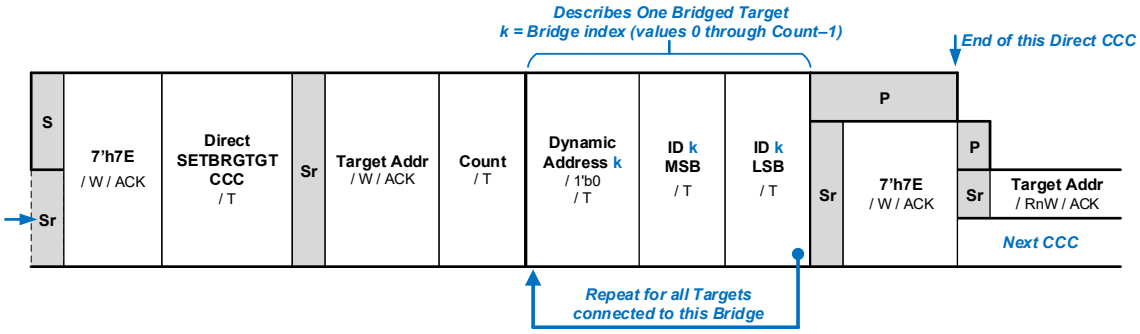


Figure 45 SETBRGTGT Format

Target Addr is the I3C Basic Dynamic Address of the Bridge Device.

Count is the number of bridged endpoints that will be exposed as Targets.

Each described bridged Target is represented by two fields:

- **Dynamic Address:** The 7 most significant bits (Bits[7:1]) contain the Target's assigned 7-bit Dynamic Address, and the least significant bit (Bit[0]) is filled with the value 1'b0. For Legacy I^2C Devices, the value of the Dynamic Address shall be 7'h00.
- **ID:** A 16-bit unambiguous identifier for the bridged Target (two bytes)

The meaning of the ID field is not defined by this Specification, and should be a contract between the Bridge and the Controller (or any other entity that supplies such information to the Controller). For example, the upper nibble of the MSB could indicate the communication type (e.g., I^2C, SPI, UART, LIN, etc.), and the remaining 12 bits (i.e., lower nibble of MSB plus full LSB) could be the port (as port number and Device selector).

Note:

1. If the Controller needs to change the Dynamic Address of a bridged Target that was configured using SETBRGTGT, then the change shall be performed using an updated SETBRGTGT, and not SETNEWDA.

2. The Controller shall not rely on each bridged Target having a unique BCR, DCR, or PID; so, use of GETBCR, GETDCR, and GETPID may return the same result for each.

3. The Controller shall not rely on each bridged Target handling directed CCCs as unique. So, for example, SETMRL to one bridged Target may affect all Targets accessed via the same bridge.

4. Use of GETMXDS is recommended for bridged Targets, such that each may return its own appropriate values based on protocol being bridged to.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.18 Get Max Data Speed (GETMXDS)

The Controller uses this Direct CCC (there are three formats: ***Figure 46***, ***Figure 47***, and ***Figure 48***) to determine the SDR Mode data speed limitations of one Target Device. The Controller is required to use this CCC only when Bit[0] of the addressed Target Device's Bus Control Register (BCR) is set to 1'b1 (see ***Table 7***). A Target Device is required to support this CCC only when Bit[0] of its Bus Control Register (BCR) is set to 1'b1 (see ***Table 7***).

Bit[0] of the Bus Characteristics Register shall be set in any Target Device with a Clock-to-Data turnaround time greater than 20 ns. Target Devices within the t_{SCO} delay range should not set the limitation bit (i.e., Bit[0]) unless other limitations exist. But Devices should support this CCC to allow better factoring of each number when computing the maximum effective frequency for reads, since the Controller/System-designer can be told the t_{SCO} parameter along with line length (propagation time), line capacitance, number of Targets, and stubs if present.

Bit[6] of the maxRd byte indicates whether the Controller may insert a STOP between the Write (index) and the Read. If the value of Bit[6] is 1'b1, then the Controller can use the extra time for other communications or end the communication and re-initiate it at the appropriate time. If the value of Bit[6] is 1'b0, then the Controller shall keep within one Frame. If the Maximum Read Turnaround time is more than a few micro-seconds, then the Target should permit STOP between the Write and Read.

Note:

*This CCC relates to bit data rates and Read Turnaround times, not data sizes. Data sizes are the subject of the CCCs Get Max Read Length (GETMRL, see **Section 4.3.7.3.6**) and Get Max Write Length (GETMWL, see **Section 4.3.7.3.5**).*

The Target Device shall return **either**:

- **GETMXDS Format 1:** Two data bytes containing the Target Device's Maximum Write Speed and Maximum Read Speed (see ***Figure 46***), **or**
- **GETMXDS Format 2:** Five data bytes containing the Target Device's Maximum Write Speed, Maximum Read Speed, and a three-byte Maximum Read Turnaround Time (see ***Figure 47***).

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

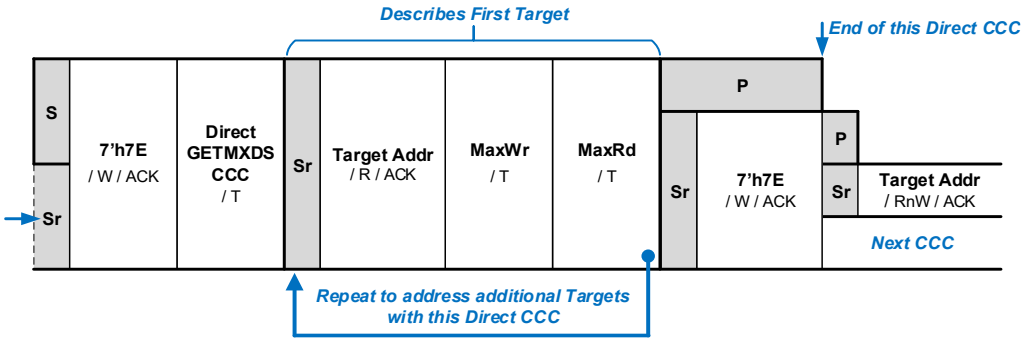
The Target's selection of Format 1 vs. Format 2 should be based upon whether Maximum Read Turnaround Time needs to be communicated to the Controller Device.

Interpretation of the returned fields maxWr, maxRd, and (for Format 2 only) maxRdTurn is shown in ***Table 30***, ***Table 31***, and ***Table 32*** respectively.

When reading the GETMXDS value from a Target, the SCL clock rate should be slowed (by frequency or duty cycle) to accommodate a possible slow Clock-to-Data turnaround (t_{SCO}). This is only needed if the BCR bit has indicated a limitation. If BCR is not known (e.g., if SETDASA or SETAASA was used), then either read GETBCR slowly, or simply read GETMXDS slowly no matter what. See also ***Section 5.2.5*** **BCR Use** in the *MIPI Alliance I3C Application Note: General Topics* ***[MIPI03]***.

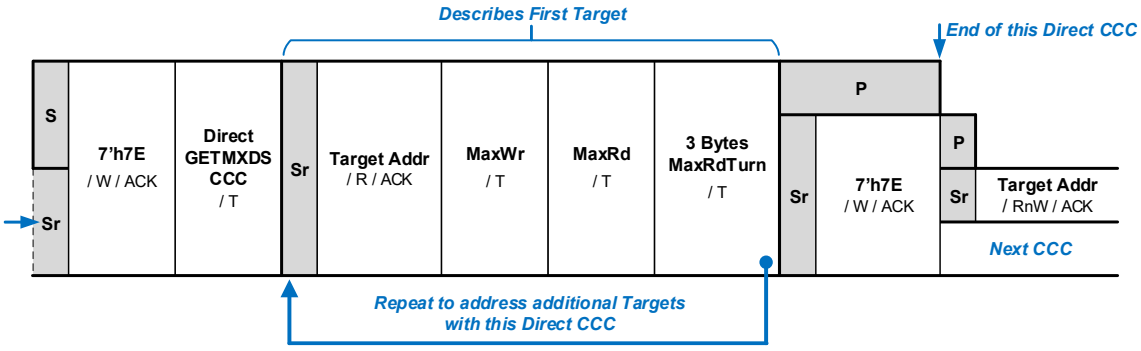
The Command Code for the GETMXDS Direct CCC is 0x94. Support for this CCC is required if Bit[0] of the I3C Basic Target's Bus Control Register (BCR) is set to 1'b1.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



2726

Figure 46 GETMXDS Format 1: Without Turnaround



2727

Figure 47 GETMXDS Format 2: With Turnaround

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

2728 **Table 30 maxWr Byte Format**

Bits	Field	Description
7:4	**Maximum Supported Read and Write Frequency**	**Values:** **0:** < 1 MHz **1:** 1 MHz **2:** 2 MHz **…** **10:** 10 MHz **11:** 11 MHz **12:** 12 MHz 13–15: Reserved for future use by MIPI Alliance
3	**Defining Byte Support**	Does this I3C Basic Device support an optional Defining Byte for this CCC? **0:** No **1:** Yes
2:0	**Maximum Sustained Data Rate for non-CCC Messages sent by Controller Device to Target Device**	**Values:** **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 MIPI Alliance

2729 **Table 31 maxRd Byte Format**

Bits	Field	Description
7	Reserved	Reserved for future use by MIPI Alliance
6	**Write-to-Read Permits Stop Between**	If maxRdTurn is not 0, then this field is used to tell the Controller whether the Target permits the Write-to-Read to be split by a STOP. **0:** STOP would cancel the Read **1:** The Target permits the Write-to-Read to be split by a STOP.
5:3	**Clock to Data Turnaround Time (t_{SCO})**	**Values:** **0:** ≤ 8 ns (default value) **1:** ≤ 12 ns **2:** ≤ 15 ns **3:** ≤ 20 ns 4–6: Reserved for future use by MIPI Alliance **7:** t_{SCO} is reported by private agreement
2:0	**Maximum Sustained Data Rate for non-CCC Messages sent by Target Device to Controller Device**	**Values:** **0:** f_{SCL} Max (default value) **1:** 8 MHz **2:** 6 MHz **3:** 4 MHz **4:** 2 MHz 5–6: Reserved for future use by MIPI Alliance **7:** Use bits 7:4 of maxWr Byte to identify accurate frequency

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Table 32 maxRdTurn Format

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.18.1 GETMXDS Format 3

Additionally, some I3C Basic Devices may support this CCC with an optional Defining Byte (***Figure 48***), to return other capabilities based on the value of the Defining Byte. This support may be required as part of the Device's role, capabilities, or other extended behavior. It is the Controller's responsibility to determine whether the Device supports this CCC with an optional Defining Byte. The Controller may determine whether the Target has such support, by first sending this CCC without a Defining Byte, and checking for a value of 1'b1 in bit 3 of Byte 1 (maxWr) in the data read from the Target (see ***Table 31***).

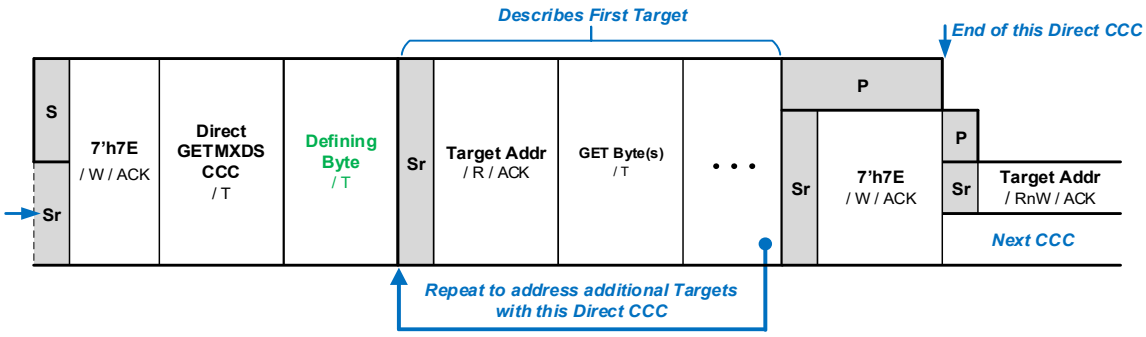


Figure 48 GETMXDS Format 3: With Defining Byte

To read the other Device capabilities accessed via a Defining Byte, the Controller shall send this CCC with a Defining Byte, according to the CCC Direct General Frame Format (***Figure 20***). If the Device identified by the Target Address does not support a particular Defining Byte value for this CCC, then it shall NACK its Target Address to terminate the Direct CCC. If the Device does support a particular Defining Byte, then it shall ACK its Target Address and return additional bytes of data. The length of the data returned shall depend on the particular Defining Byte and its associated capabilities (see ***Table 33*** for supported Defining Byte values; see subsections below for specifics per Defining Byte value).

Table 33 GETMXDS Defining Byte Values

Value	Encoding	Description
0x00	**WRRDTURN**	Describes standard (Target) Write/Read speed parameters, and optional Maximum Read Turnaround time. Equivalent to the GETMXDS CCC without a Defining Byte.
0x01 – 0x90	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG
0x91	**CRHDLY**	Describes delay parameters for a Controller-capable (i.e., Secondary Controller) Device, and its expected Activity State during the Controller handoff (see sub-section below). Recommended if an I3C Basic Device is compliant with version 1.1 or higher of the I3C Basic Specification, and if it advertises itself as Controller-capable (i.e., if BCR bits 7:6 contain the value 2'b01).
0x92 – 0xBF	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG
0xC0 – 0xDF	**Reserved**	Reserved for future definition by other MIPI Alliance WGs
0xE0 – 0xFE	**Vendor Extensions**	For Vendor use
0xFF	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG

According to the Retry Model for Direct GET CCC Commands, if a Device cannot provide an immediate response to this CCC with a particular Defining Byte, then it shall NACK the first attempt, and then the Controller shall attempt a single retry (see ***Section 4.3.7.2.3***). If the Controller receives a NACK after a retry, then the Controller shall assume that the Device does not support this CCC with this particular Defining Byte, and hence has no data

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

that would be returned for any capabilities relating to that Defining Byte value. In most cases, the Controller may proceed as though it read a value equivalent to all zeros as a default value. However, a NACK after a retry should not be interpreted as meaning that the Device does not possess a capability or feature set associated with that Defining Byte value.

A Target that supports this CCC with any optional Defining Bytes shall return a value of 1'b1 in bit 3 of Byte 1 (maxWr) when this CCC is sent without a Defining Byte. If the Controller sees that bit set to 1'b1, then it knows that the Target is capable of supporting at least one Defining Byte with this CCC. The ACK/NACK method explained above allows the Target to indicate which Defining Byte values it actually supports.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.18.2 Get Controller Handoff Delay Parameters (GETMXDS Format 3, Defining Byte CRHDLY)

This Direct CCC with Defining Byte value CRHDLY (***Figure 49***) allows the Active Controller to query a Controller-capable (i.e., Secondary Controller) I3C Basic Device, to read what optional Controller handoff delays or other timing parameters that Device will utilize. It is anticipated that a Device may advertise these parameters, to indicate whether it will enter an Activity State that requires the Active Controller to wait for an appropriate delay, before confirming its Controller Role or attempting the CE3 Error Recovery flow.

It is recommended that all Controller-capable Devices should support this CCC with this Defining Byte in Target mode. If such a Device does not follow the default delay settings, or if it indicates other non-default timing parameters, then the Device shall support this Defining Byte.

A Controller-capable Device may return one data byte, labeled CRHDLY1 per ***Figure 49***.

A Device that is not Controller-capable shall NACK its Target Address, to indicate that it does not support this CCC with this particular Defining Byte.

Fields within the CRHDLY1 Byte are detailed in ***Table 34***.

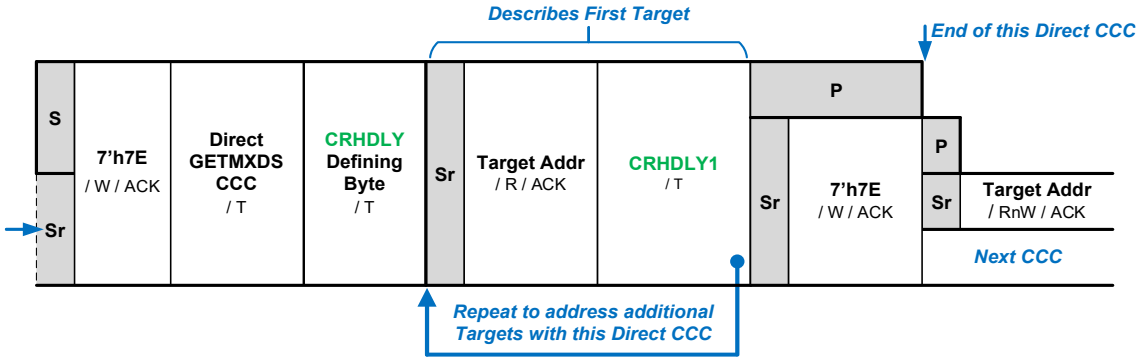


Figure 49 GETMXDS Format 3 With Defining Byte CRHDLY

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

2771

Table 34 CRHDLY1 Byte Format

Bits	Field	Description
7:3	Reserved	Reserved for future definition by MIPI Alliance I3C WG
2	**Set Bus Activity State**	This bit indicates whether the Active Controller should set the Bus to the Activity State indicated in bits 1:0 before handing the Controller Role off to this I3C Basic Device as part of the normal flow. **Values:** **0:** The Active Controller should not set the Bus to any Activity State before handoff to this Device. **1:** The Active Controller should set the Bus to the Activity State indicated by bits 1:0 before handoff to this Device. *For the Activity States and their delay intervals, see **Table 12**.*
1:0	**Controller Handoff Activity State**	This 2-bit integer indicates whether this I3C Device initially acts with a given Activity State, after becoming Active Controller on the Bus according to the Controller handoff flow. The indicated Activity State implies that the Device may have a delayed response to Bus activity, and so the former Controller should wait the specified delay time for the indicated Activity State before testing this Device, to confirm that it is controlling the Bus before initiating the CE3 error recovery flow. **Values:** **0:** Acts according to Activity State 0. **1:** Acts according to Activity State 1. **2:** Acts according to Activity State 2. **3:** Acts according to Activity State 3. *For the Activity States and their delay intervals, see **Table 12**.* *For the Error Type CE3 recovery flow, see **Section 4.3.8.2.4**.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.19 Get Optional Feature Capabilities (GETCAPS)

This Direct CCC (***Figure 50***) allows the Controller to query an I3C Basic Target to determine what optional I3C Basic features that Target supports. It replaces the GETHDRCAP CCC, which was defined in version 1.0 of the I3C Basic Specification.

This CCC has two formats:

- For GETCAPS Format 1, the I3C Basic Target may return the first 2, 3, or all 4 of the GETCAP1 through GETCAP4 Bytes as detailed in ***Section 5.4***.
- The optional GETCAPS Format 2 adds a Defining Byte and returns a variable number of data bytes. GETCAPS Format 2 is detailed further in ***Section 5.4***.

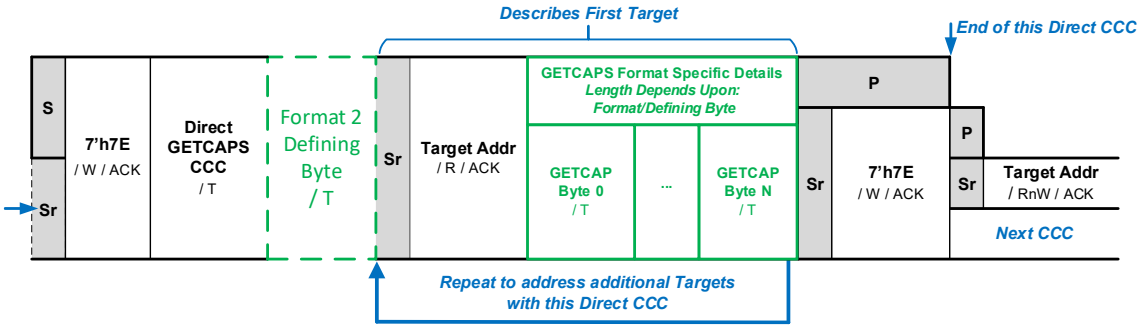


Figure 50 GETCAPS CCC Format Overview

Note:

It is possible for an I3C Basic Controller that supports I3C Basic version 1.1.1 or greater to be used with an I3C Basic Target Device that supports version 1.0 of the I3C Basic Specification. In such a case, the Target might only support the previous GETHDRCAP CCC format, and as a result might respond to the GETCAPS CCC by returning only one byte (i.e., the GETCAP1 Byte). Therefore, an I3C Basic v1.1.1 or greater Controller must recognize this as a valid configuration (even though the I3C Basic Target returns just a single byte and as a result has limited capabilities).

However, note that all I3C Basic Targets that support I3C Basic version 1.1.1 or greater are required to respond to the GETCAPS CCC by returning at least the first 2 bytes.

For the optional GETCAPS Format 2, the variable number of data bytes returned depends upon the Defining Byte value. A Controller shall not use the GETCAPS Format 2 CCC with any given Target unless it first determines that the Target supports Format 2 of the CCC. A Target that supports the GETCAPS Format 2 CCC shall return a value of 1'b1 in bit 4 of Byte 3 (GETCAP3), when the GETCAPS CCC is sent without a Defining Byte. If the Controller sees that bit set to 1'b1, then the Controller will know that the Target is capable of supporting at least one Defining Byte with the GETCAPS Format 2 CCC.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.20 Set Route (SETROUTE)

This Direct CCC (***Figure 51***) is used by an I3C Basic Controller to configure a Routing Device (Routing Devices are described below). Because this CCC is only relevant to Routing Devices, a Controller shall only send the SETROUTE CCC to a known Routing Device that accepts it.

Note:

*For additional information on Routing Devices, refer to the MIPI Alliance I3C Application Note: Virtual Devices and Virtual Targets **[MIPI10]**.*

The Command Code for the SETROUTE Direct CCC is 0x96. Support for this CCC is optional.

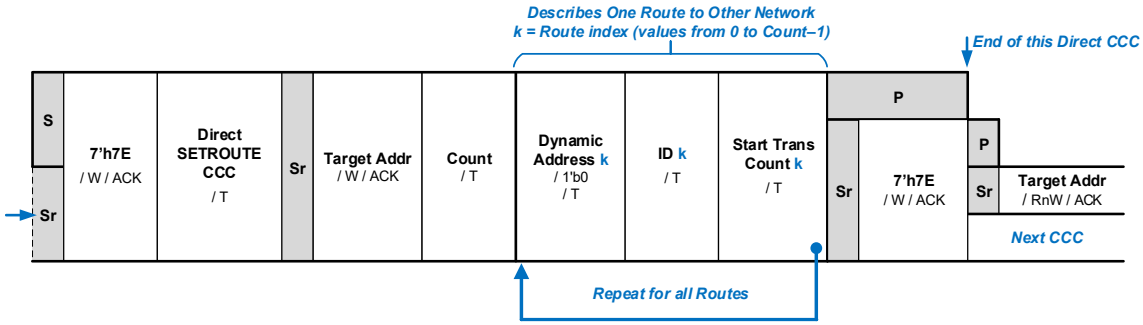


Figure 51 SETROUTE Format

Target Addr is the Address of the Routing Device: either the Target's original 7-bit static I^2C Address, or else the current value of the Target's assigned 7-bit Dynamic Address. This Target Address is used both for the Routing Device itself, and the other operations of the Device into which the routing behavior and logic is integrated.

Count is the number of configured Routes to be defined/assigned with this CCC (not the number of Routes the Routing Device provides).

Each Route is represented by three fields:

- **Dynamic Address:** The 7 most significant bits (Bits[7:1]) contain the current value of the Target's assigned 7-bit Dynamic Address, and the least significant bit (Bit[0]) is filled with the value 1'b0. This Address (not the Target Addr) is used when sending transactions to the associated I3C Basic Bus.
- **ID:** An 8-bit identifier, implementation-specific to the Routing Device, identifying the specific Route to the I3C Basic Bus endpoint provided by the Routing Device.
- **Start Trans Count:** An 8-bit value to which the Routing Device shall reset the transaction counter when initializing the Route.

The method by which the Controller is informed about the identifiers for the various Routes provided by the Routing Device is not defined by this Specification; the Controller shall have prior knowledge of the identifiers, through either discovery registers or a priori knowledge. Although the identifiers used by the Routing Device are arbitrary, they shall be unique per I3C Basic Bus endpoint.

Only one Controller using the SETROUTE CCC shall configure a set of Routes at any given time. Any Controller accessing the Routing Device on any of the I3C Basic Buses shall configure the Route(s). Only one Controller and its associated configured Routes shall be active at any given time. Each time a different Controller desires to configure its Route(s) through the Routing Device, it shall negotiate for ownership of the I3C Basic Bus first.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.20.1 Routing Devices

A Routing Device enables an I3C Basic Controller on a given I3C Basic Bus to communicate to one or more other I3C Basic Buses (see ***Figure 52***). That is, a Routing Device enables a connection (also known as a Route), or multiple distinct connections, to exist between the connected I3C Basic Buses. These I3C Basic Buses are independent, and may have different sets of Dynamic Addresses.

A Routing Device is distinguished from a Bridge Device, in that a Routing Device is not transparent; the transactions are relayed between the I3C Basic Buses. Since the Routing Device inevitably incurs delays on the transactions, it shall handle transactions through buffers/queues in order to meet the transmit/receive requirements of the I3C Basic protocol on each connected I3C Basic Bus. Transactions from the Controller on its I3C Basic Bus are queued by the Routing Device, and then sent to the other I3C Basic Buses configured by the Controller. Each Route should have its own separate queue, in order to handle differences in timing and clocking.

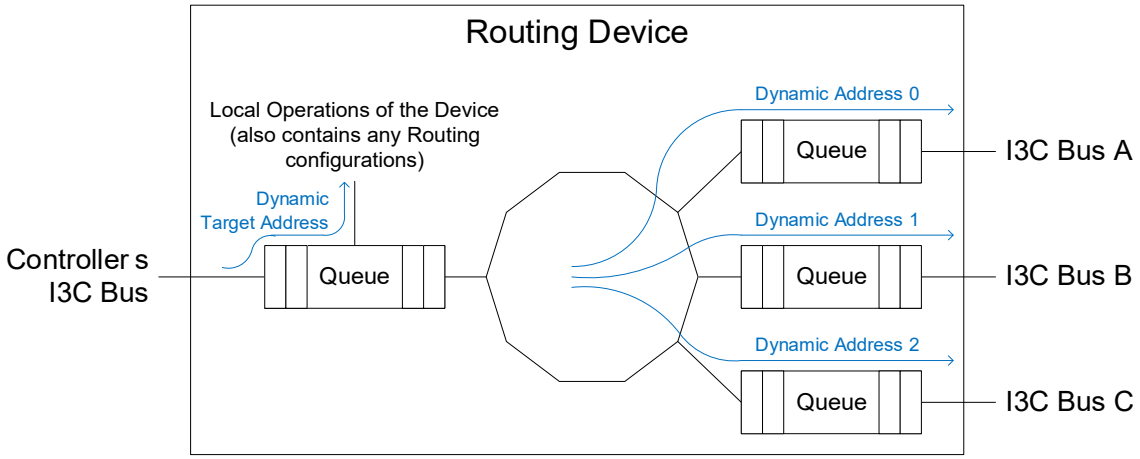


Figure 52 Example Routing Device

Routes through the Routing Device will work either in a single direction, or in multiple directions, depending on the capabilities of each I3C Basic Bus endpoint.

Examples:

- An I3C Basic Bus endpoint that is both a Target and a Secondary Controller supports Routes in both directions through that endpoint.
- An I3C Basic Bus endpoint that is only a Target, and not a Secondary Controller, only supports single-direction Routes (i.e., inward to the Routing Device).

Responses are queued and tagged with a transaction count, so that the original sender of the transaction can associate the responses with the original transaction. Each Route shall have its own counter. The initial value of the counter should be configurable, to allow the Controller to reset the counter to a previous value after a change in ownership.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.21 Set All Addresses to Static Address (SETAASA)

This CCC (***Figure 53***) allows the Controller to request that all connected Targets that have I^2C Static Addresses use their I^2C Static Address as their Dynamic Address. This is the fastest method to assign I3C Basic Dynamic Addresses to all I3C Basic Targets with I^2C Static Addresses (see ***Section 4.3.4.2***).

Note:

*Per **Section 5.1.4**, only the SETNEWDA and RSTDAA CCCs will cause a Target that already has an assigned Dynamic Address to change or lose its address. Targets that support the SETAASA CCC shall not respond to this CCC if a Dynamic Address has already been assigned (i.e., by any of the CCCs that set or assign a Dynamic Address). If a Dynamic Address has already been assigned, then the Target shall ignore this command.*

Note:

*Per **Section 4.3.4.2**, SETAASA is intended as an optimization. An I3C Basic Target that supports SETAASA and does not support ENTDAA cannot be relied on to be assigned a Dynamic Address on a generic I3C Basic Bus, and also will not support Hot-Join (per **Section 4.3.5**). Use of SETAASA requires the I3C Basic Controller to have prior knowledge of the I3C Basic Target, as well as all other I3C Basic Targets that support SETAASA. Dynamic Address Assignment with ENTDAA is recommended for most use cases.*

The Command Code for the SETAASA Broadcast CCC is 0x29. Support for this CCC is optional; however, any I3C Basic Target that supports this CCC shall have an I^2C Static Address.

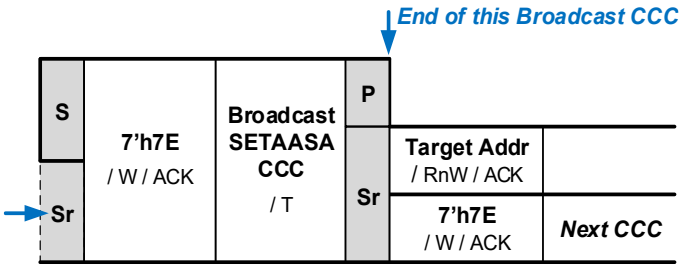


Figure 53 SETAASA Format

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.22 Data Transfer Ending Procedure Control (ENDXFER)

The ENDXFER CCC provides the framework for Controllers and Targets to exchange set-up parameters for ending data transfers in HDR Modes, and for Monitoring Devices to request early termination of running transactions. The ENDXFER CCC is available as both a Broadcast Command (***Figure 54***) and a Direct Command (***Figure 55***).

The ENDXFER CCC always includes the one-byte Defining Byte field, which acts as a sub-command identifier defining the specific function of the CCC. For some Sub-Commands (i.e., for some values of the Defining Byte field), additional Data Bytes follow. Interpretation of these additional Data Bytes depends upon the particular Defining Byte value, as detailed in ***Table 36***.

Table 35 Data Transfer Ending Procedure Control Command Codes (ENDXFER)

Command	Support	Command Codes		Purpose
		Broadcast	Direct	
ENDXFER	Conditional	0x12	0x92	Data Transfer Ending Procedure Control

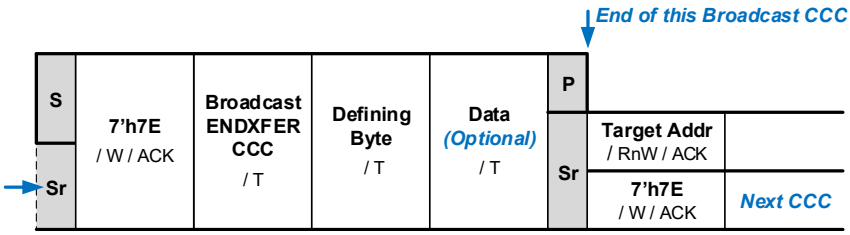


Figure 54 ENDXFER Format 1: Broadcast

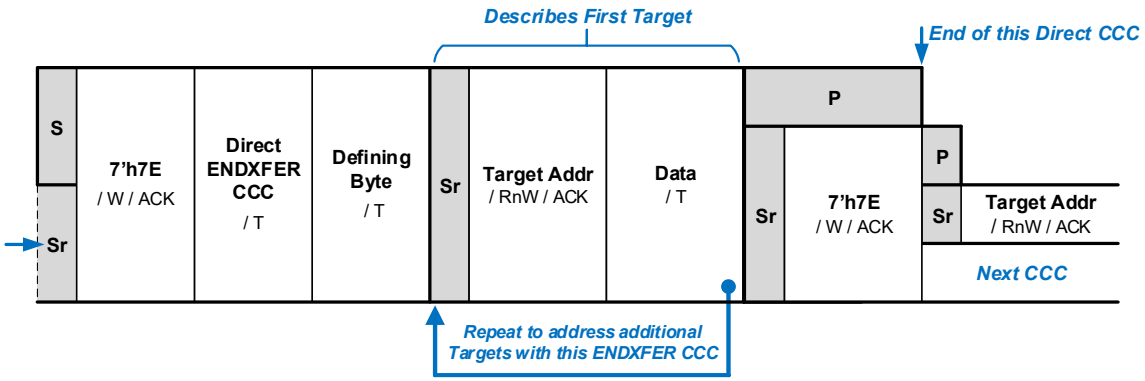


Figure 55 ENDXFER Format 2: Direct

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

2876

Table 36 ENDXFER Defining Byte Values

Defining Byte Value	Sub-Command Function	Description	Additional Data Bytes
0x7F	Sub-Commands for HDR-TSL and HDR-TSP protocols	These capabilities are not included in the I3C Basic Specification. To gain access to them, please contact MIPI Alliance.	One Byte
0x55			One Byte
0xF7	For HDR-DDR protocol: **SET/GET Data Transfer Early Termination Parameters**	SET and GET the parameters for the Data Transfer Early Termination procedure, for the HDR-DDR protocol (see ***Section 6.2.3.4***).	One Byte. See ***Section 6.2.3.4.4*** and ***Table 84*** for more details on the Defining Bytes for HDR-DDR protocol.
0xAA	For HDR-DDR protocol: **Confirm Set-up and Initiate Data Transfer** using Data Transfer Early Termination Procedure	Commands either all I3C Basic Devices on the Bus, or else only some directly addressed Devices on the Bus, to begin operating using the Data Transfer Early Termination procedure, as previously set up and confirmed using ENDXFER sub-command 0xF7.	One Byte. See ***Section 6.2.3.4.4*** and ***Table 84*** for more details on the Defining Bytes for HDR-DDR protocol.
0xFC	Sub-Command for Monitoring Device Early Termination Capability	These capabilities are not included in the I3C Basic Specification. To gain access to them, please contact MIPI Alliance.	One Byte
All Others	Reserved	Defining Byte values reserved and available for future definition (i.e., for new ENDXFER Sub-Commands)	Reserved for future definition

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.23 Target Reset Action (RSTACT)

This Broadcast (***Figure 56***), Direct Write (***Figure 57***), and Direct Read (***Figure 58***) CCC is used to configure the next Target Reset action, and may be used to retrieve a Target's reset recovery timing. The RSTACT CCC is used in conjunction with the Target Reset Pattern (***Section 4.3.9.3***), i.e., the reset action previously configured in a Target via the RSTACT CCC is triggered when the immediately following Target Reset Pattern is received.

The RSTACT CCC uses a Defining Byte (***Table 38***).

- For the Broadcast and Direct Write formats, the Defining Byte indicates which Target Reset action (including to take no action) is to be configured (values 0x00 through 0x7F).
  - Defining Bytes 0x00 and 0x01 are required: A Target shall support these operations, and shall ACK its Target Address if issued as a Direct CCC.
  - Support for other Defining Bytes (values 0x02 through 0x7F) is optional, and depends on other conditions or support for other capabilities. If a Target does not support such an operation, then it shall NACK its Target Address for such a Defining Byte, as well as any related Defining Bytes defined for the Direct Read format (values 0x82 through 0xFF) if issued as a Direct CCC.
- For the Direct Read format, the Defining Byte may also indicate the Controller's desire to read back the Target's reset recovering timing or other parameters for the operation (values 0x81 through 0xFF).
  - If the CCC is NACKed and the related reset operation is supported, then the Controller should assume the default reset return times of 1 ms to reset the Peripheral (i.e., the reset operation for Defining Byte 0x01) and 1 second to reset the whole Target Device (i.e., the reset operation for Defining Byte 0x02).

Table 37 Target Reset Action Command Codes (RSTACT)

Command	Support	Command Codes		Purpose
		Broadcast	Direct	
RSTACT	Required	0x2A	0x9A	Target Reset Action

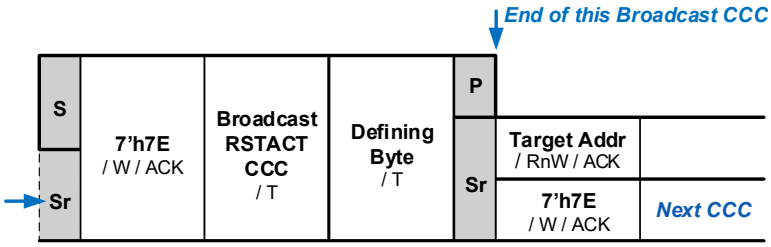
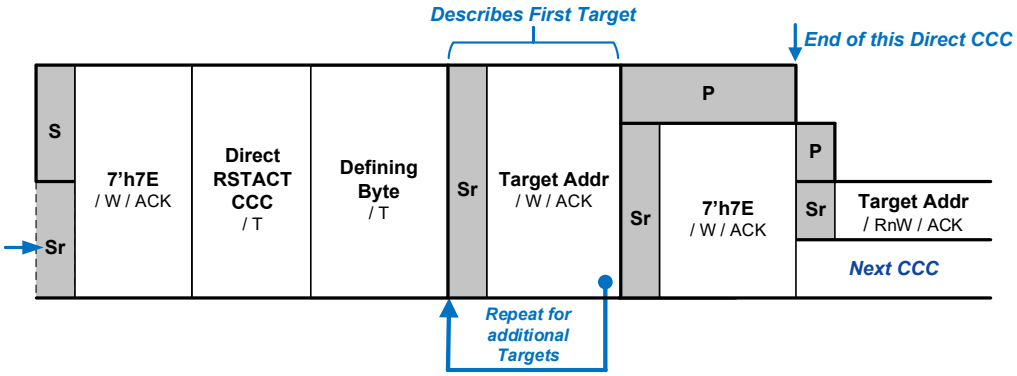


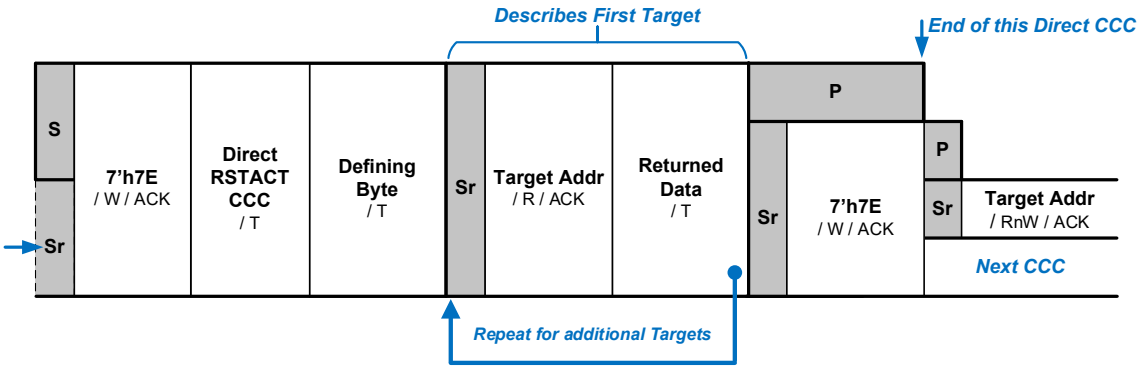
Figure 56 RSTACT Format 1: Broadcast

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



2898

Figure 57 RSTACT Format 2: Direct Write



2899

Figure 58 RSTACT Format 3: Direct Read

2900

Table 38 RSTACT Defining Byte Values

Defining Byte Value	Description	NACK Means Default Of	Notes	SET / GET
0x00	**No Reset on Target Reset Pattern**	–	**SET:** No data (Defining Byte is the value) **GET:** Returns currently set Defining Byte value (i.e., previously received in either a Direct SET or a Broadcast) or 0x80	SET / GET
0x01	**Reset the I3C Basic Peripheral Only** (Default)	–		SET / GET
0x02	**Reset the Whole Target**	Not supported		SET / GET
0x03	**Debug Network Adaptor Reset** (The Target shall not reset the I3C Basic Peripheral)	Debug Over I3C not supported		SET / GET
0x04 (Direct CCC only)	**Virtual Target Detect** • **SET:** The Target shall not prepare for reset, but shall set a flag in its shared Peripheral logic that can be read by all other linked Virtual Targets. This flag can be cleared with Defining Byte 0x00 (Direct or Broadcast) • **GET:** The Target shall return 0x01 if the flag is set, i.e., if a previous SET operation was directed to any linked Virtual Target in this Device (including itself), without an intervening use of Defining Byte 0x00. The Target shall return 0x00 if no SET was received, or if the flag was cleared by the RSTACT CCC with Defining Byte 0x00 (No Reset on the Target Reset Pattern)	Device is not Virtual Target capable		SET / GET
0x05 – 0x3F	Reserved by MIPI	–		–
0x40 – 0x7F	Reserved for Vendors and external standards	–		–
0x80	Reserved by MIPI	–	**SET:** Ignored (not used) **GET:** Returns the time, based upon the Defining Byte	–
0x81	**Return Time to Reset Peripheral**	1 ms		GET
0x82	**Return Time to Reset Whole Target**	1 s		GET
0x83	**Return Time for Debug Network Adaptor Reset**	100 ms		GET
0x84	**Return Virtual Target Indication** Return 0x01 if this Target is a Virtual Target and supports Defining Byte 0x04 (Virtual Target Detect). Otherwise return 0x00.	Target is not Virtual		GET
0x85 – 0xBF	Reserved for timing for MIPI reserved values	–		GET
0xC0 – 0xFF	Reserved for timing for Vendor and external standards reserved values	–		GET

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Note:

1. The Target shall clear its reset action (as configured via the RSTACT CCC) upon receipt of the next START (but not Repeated START).

*2. Reset operation Reset the Whole Target (Defining Byte 0x02) is optional, but support for this reset operation is recommended; see also **Section 4.3.9.4**. A Target Device that does not support this reset operation shall NACK its Target Address for this Defining Byte (if issued as a Direct CCC) or ignore the RSTACT CCC (if issued as a Broadcast CCC).*

4.3.7.3.23.1 Debug Network Adaptor Reset

Reset operation Debug Network Adaptor Reset (Defining Byte 0x03) is intended for Target Devices that support the MIPI Debug Over I3C specification ***[MIPI04]***. Support for this reset operation is strongly recommended for any Debug-capable I3C Basic Device that supports MIPI Debug Over I3C as a Debug-capable Target System (TS).

The Debug Network Adaptor Reset operation (Defining Byte 0x03) does **not** reset the I3C Basic Peripheral or its Dynamic Address; it only resets the Device's Debug Over I3C Network Adaptors and all attached Debug Function logic. If issued as a Broadcast CCC, then it shall only affect Targets that support a Debug Network Adaptor reset operation.

A Target Device that does not support the Debug Network Adaptor Reset operation:

- Shall NACK its Dynamic Address for this Defining Byte if issued as a Direct CCC
- Shall ignore the RSTACT CCC if issued as a Broadcast CCC.

4.3.7.3.23.2 Virtual Target Detect

Reset operation Virtual Target Detect (Defining Byte 0x04) is intended for composite Devices that support multiple Virtual Targets that share the same Peripheral logic (see ***Section 4.3.2.1.2***). Support for this Defining Byte is required for any such Device that reports Virtual Target capabilities; i.e.,is required if BCR bit[4] is set to 1'b1 and if the Device supports the GETCAPS CCC with Defining Byte VTCAPS to indicate multiple Virtual Targets using shared Peripheral logic (see ***Section 5.4.5***).

Additional requirements apply:

- This operation does **not** configure a Target or its shared Peripheral logic for any defined Target Reset action. I.e., it is only used for identification and association of any linked Virtual Targets.
- This Defining Byte is only supported with Direct CCCs, **not** Broadcast CCCs.
- The flag that resides in the shared Peripheral logic and which is set by Defining Byte 0x04 shall **not** be accessible or writeable from any application-specific logic connected to any Virtual Target. The state of this flag shall not be accessed by any Controller by any method other than using this CCC (RSTACT) with this Defining Byte(0x04), and this flag shall not be cleared by any Controller by any method other than using this CCC (RSTACT) with Defining Byte 0x00.

Additionally, if a shared Peripheral is implemented in a manner that passes incoming RSTACT CCC events as messages up to application-specific queues, or to buffers for any Virtual Targets, then the shared Peripheral shall **not** pass any RSTACT CCC events as messages if they contain Defining Byte 0x04 or 0x84 or otherwise change the flag. This is because these messages might notify a Virtual Target that the flag is being set or accessed by a Controller.

Note:

*For additional information on Virtual Targets, refer to the MIPI Alliance I3C Application Note: Virtual Devices and Virtual Targets **[MIPI10]**.*

A Target Device that does not support the Virtual Target Detect reset operation shall NACK its Dynamic Address for this Defining Byte (if issued as a Direct CCC).

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

4.3.7.3.23.3 Return Reset Times

If the Target supports the Direct Read format of the RSTACT CCC, and provides ACK to the RSTACT CCC with any Defining Byte value in the range from 0x81 through 0x83 (inclusive) to return the reset time, then the data byte value returned by the Target shall have the data format indicated in ***Table 39***.

Table 39 RSTACT Format 3: Direct Read CCC Byte Read Data Format

Bits	Description	Value
7	Indicates the time unit for the value in Bits[6:0]	1'b0: time unit of 10 ms 1'b1: time unit of 1 second
6:0	Amount of time in units	7'h00: 1 ms always (regardless of unit) 7'h01 – 7'h7F: 1 thru 127 units
	Exceptions:	8'hFF: Return time is either beyond the maximum value, or is indeterminate

Examples:

- Values 8'h00 and 8'h80 indicate 1 ms return time.
- Value 8'h01 indicates 10 ms return time.
- Value 8'h02 indicates 20 ms return time.
- Value 8'h7F indicates 1270 ms (i.e., 1.27 seconds) return time.
- Value 8'h81 indicates 1 second return time.
- Value 8'h82 indicates 2 seconds return time.
- Value 8'hFE indicates 126 seconds return time.
- Value 8'hFF indicates that the return time is either beyond the maximum value (i.e., reported by private contract), or is indeterminate.
  In this case, the Controller should either:
  - Wait for an appropriate event based on the reset type, which could be either a Hot-Join Request or some other notification (e.g., an In-Band Interrupt Request defined by the I3C Basic content protocol); or
  - Use some other polling mechanism (i.e., defined by private contract) to determine when the reset is complete.

Note:

In earlier versions of the I3C Basic Specification, the unit values were not clearly defined. Targets that comply with I3C Basic Specification v1.1.1 or earlier should return single-byte values assuming legacy units of 1 ms for Defining Byte 0x81; 1 second for Defining Byte 0x82; and 1 ms for Defining Byte 0x83.

If the Target resets quickly, i.e., if its return time is less than 1 ms, then the Target should either NACK its Dynamic Address (i.e., to indicate the default return time) or ACK its Dynamic Address and return a value of 1 ms (i.e., 8'h00 or 8'h80).

Before using the RSTACT Format 3 Direct Read CCC to read the return time, the Controller should also use the GETCAPS CCC to confirm that the Target complies with v1.2 or newer of the I3C Basic Specification. If so, then the Controller will know whether the Target uses the data format defined in **Table 39***. However, if this is not the case, then the Controller should use the legacy units to interpret the data byte value.*

If implementers choose to support any of the available RSTACT Defining Byte values (i.e., in the range of 0x40 thru 0x7F) to reset any other logic, functions or domains within the chip, then implementers should also support the corresponding RSTACT Defining Byte values (i.e., in the range of 0xC0 thru 0xFF) to indicate the expected return times for such reset operations, using the same data format defined in ***Table 39***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.24 Set Group Address (SETGRPA)

This Direct CCC (***Figure 59***) allows the Active Controller to assign a Group Address to an I3C Basic Target supporting the Group Address feature. The Target will then respond to both the configured Group Address and its Dynamic Address. If multiple I3C Basic Targets are configured with the same Group Address, then a single Message sent by the Active Controller to that Group Address will be received by all of those I3C Basic Targets simultaneously.

Note:

In this CCC, an I3C Basic Target to be assigned a Group Address is indicated by its Dynamic Address. As a result, this CCC cannot be used until the Target has been assigned a Dynamic Address.

If the Target supports multiple Group Addresses, then the same Dynamic Address can be used in multiple SETGRPA CCCs (up to the maximum number of Group Addresses supported by the I3C Basic Device), supplying a different Group Address in each instance. The Target will then respond to any of the configured Group Addresses, in addition to its Dynamic Address.

When a Target is assigned to the maximum number of groups that it is capable of supporting, the Target shall inform the Controller that this maximum capability has been reached by NACKing the Target Address for this Direct CCC.

The Group Address to be assigned is transmitted in the Group Address byte shown in ***Figure 59***, where the 7 most significant bits (Bits[7:1]) contain the 7-bit Group Address, and the least significant bit (Bit[0]) is filled with the value 1'b0.

The Command Code for the SETGRPA Direct CCC is 0x9B. Support for this CCC is required if the I3C Basic Target supports Group Addressing (per ***Section 4.3.4.4***).

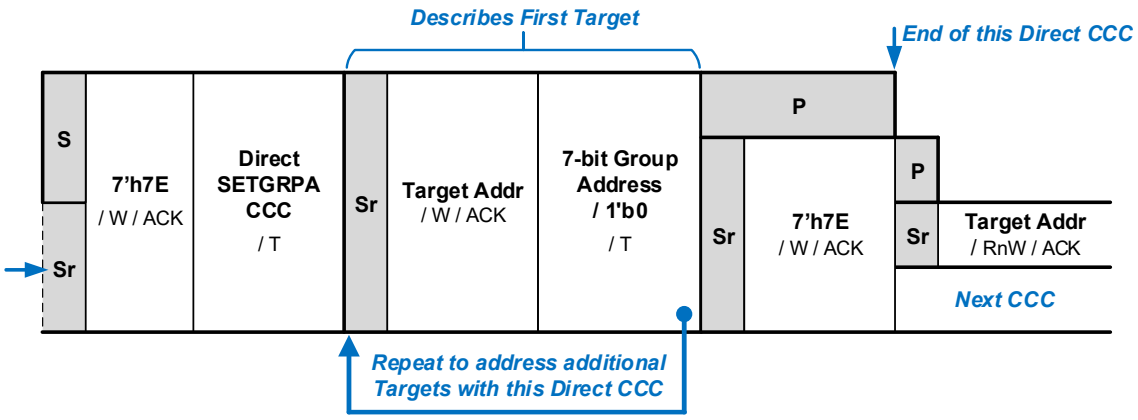


Figure 59 SETGRPA Format

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.25 Reset Group Address (RSTGRPA)

This CCC is available in Direct and Broadcast versions:

- **Direct** (***Figure 60***)**:** This CCC allows the Active Controller to remove one or more I3C Basic Target Devices from a Group by resetting their assigned Group Address. If a Group Address is used instead of a Target Address (***Figure 61***), then this CCC also allows the Active Controller to disband the indicated Group (i.e., the assigned Group Address is reset for all Devices in the Group, with the result that no Devices remain in the Group).
- **Broadcast** (***Figure 62***)**:** This CCC allows the Active Controller to disband all Groups, by resetting the assigned Group Address for all Devices with the result that no Devices remain in any Group.

Table 40 Reset Group Address Command Codes (RSTGRPA)

Command	Support	CCC Codes		Purpose
		Broadcast	Direct	
RSTGRPA	Conditional	0x2C	0x9C	Reset Group Address

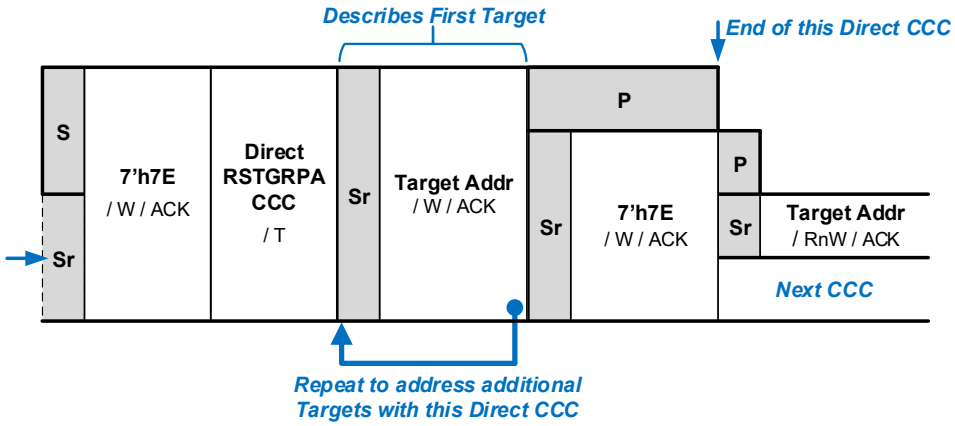


Figure 60 RSTGRPA Format 1: Direct

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

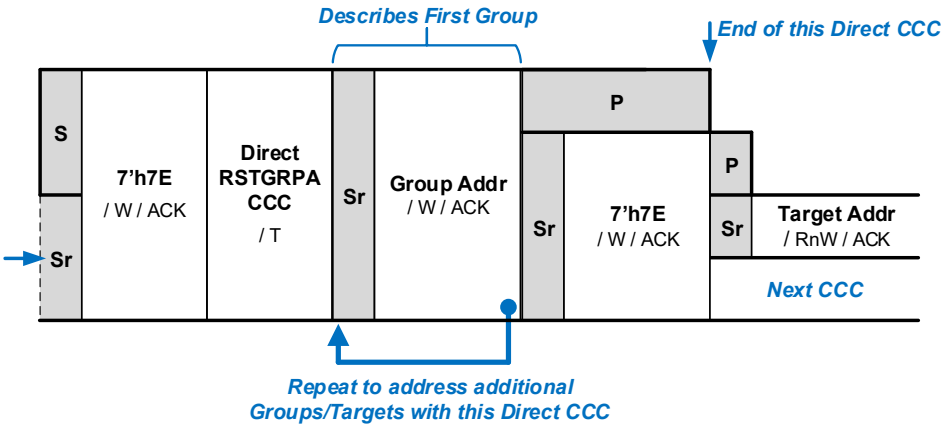


Figure 61 Resetting a Group Address with the RSTGRPA Direct CCC

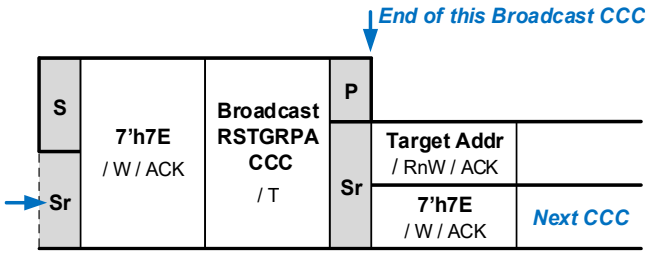


Figure 62 RSTGRPA Format 2: Broadcast

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.26 Define List of Group Addresses (DEFGRPA)

This Broadcast CCC (***Figure 63***) is only relevant to Secondary Controllers (as with the DEFTGTS CCC; see ***Section 4.3.7.3.7***). The DEFGRPA CCC tells the Secondary Controller more details about a particular Group, including the list of I3C Basic Targets that are members of the Group. If multiple Groups have been configured, then the Active Controller should issue one DEFGRPA CCC for each configured Group. Each use of the DEFGRPA CCC shall include the Group Address that uniquely identifies the Group, as well as the Dynamic Addresses of its I3C Basic Targets.

The Command Code for the DEFGRPA Broadcast CCC is 0x2B. Support for this CCC is required if an I3C Basic Secondary Controller supports Group Addressing (per ***Section 4.3.4.4***) as well as handoff and/or management of Group Addresses, as part of Controller Role Handoff.

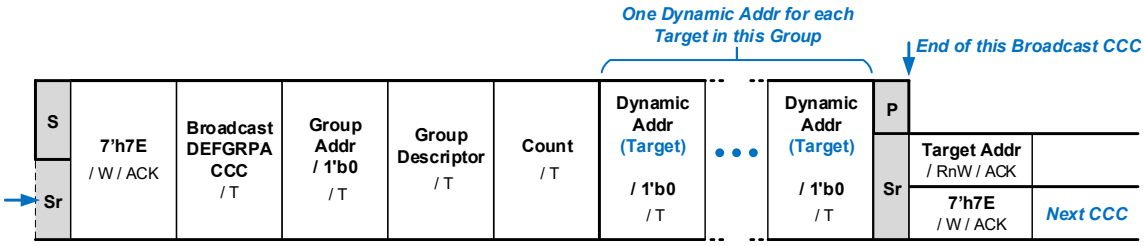


Figure 63 DEFGRPA Format

- **Group Descriptor:** Identifier for this Group, using the same format and the same MIPI-defined values as the DCR (see ***Section 4.3.1.2.2*** and MIPI-maintained web-based registry of defined values ***[MIPI02]***).

  Examples: 8'd0 for Generic v1.0 Device, 8'd33 for Touch, 8'd129 for Proximity, etc.
- **Count:** The number of the I3C Basic Targets that are members of this Group. This is the same as the number of Dynamic Addr bytes that follow.
- **Dynamic Address:** For each instance, the 7 most significant bits (Bits[7:1]) contain a member Target's current 7-bit Dynamic Address. The least significant bit (Bit[0]) has the value 1'b0.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.3.27 Set Bus Context (SETBUSCON)

This optional Broadcast CCC (***Figure 64***) allows the I3C Basic Controller to specify that a particular context (***Table 41***) is being used on the I3C Basic Bus, i.e., allows the Controller to set the Bus context. The context will usually be a higher-level protocol specification published by a standards-developing organization that relies upon MIPI I3C Basic for the communication, but it may also indicate which version of the MIPI I3C Basic Specification is being used.

This CCC allows Targets to activate any special functions needed to support the selected higher-level protocol on this I3C Basic Bus. This activation includes the definition of Vendor-specific or Standards-specific CCCs, and might include initiating any preparations required before starting the selected higher-level protocol, or potentially control of any other Device capabilities.

> **Example:** A given Target might need to reduce, expand, or modify its support for the given Bus, based upon the requirements of the selected higher-level protocol specification.

The SETBUSCON CCC is normally emitted only once during Bus initialization; however, the Controller might need to emit SETBUSCON again after a Hot-Join, a Target Reset, or any other situation that might cause a Target to lose its context.

Note:

> *In the SETBUSCON model, Targets only react to Bus context byte values that they recognize. For example, a Debug-capable Target (or Debug part of a Target) will react only to the Debug context byte value, whereas the normal application Target will react only to the context byte value representing the main context.*

The Command Code for the SETBUSCON Broadcast CCC is 0x0C. Support for this CCC is optional, and shall depend on the supported context(s).

Layered Protocol Contexts

On a given Bus, SETBUSCON CCC is typically used with only one context value. However in some situations it can be used more than once, with different context byte values, in order to support a layered protocol context.

Examples:

1. Emitting both the MIPI I3C Basic Specification revision context, and the context of a given higher-level protocol. The order would be: first emit the I3C Basic Specification revision context value, and then emit the higher-level protocol context value.
2. Adding an isolated, mixed-channel use, for example Debug. This allows a normal higher-level protocol to be used alongside Debug-specific Messages, in an intermixed manner. In such cases both contexts may be emitted.
3. Setting Bus context for JEDEC SPD changes rules for Dynamic Address Assignment, and defining a set of Vendor-specific / Standards-specific CCCs.

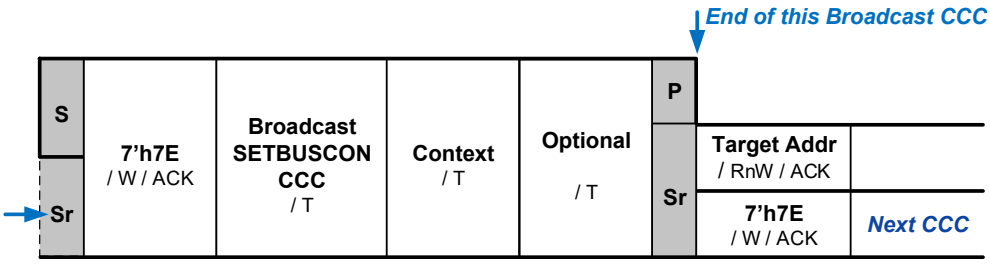


Figure 64 SETBUSCON Format

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3062 **Table 41 SETBUSCON Context Values**

Context Byte Value	Context Group	Context Description
0	None	Reserved, do not use
1 – 63	MIPI I3C Basic Specification v1.Y Minor Version	**Bits[7:6]:** 2'b00 **Bit[5]: I3C Basic Specification Editorial Revision** (within Minor Version) **1'b0**: Version 1.Y.0 **1'b1:** Version 1.Y.1 or greater **Bit[4]: I3C Specification Family** **1'b0:** MIPI I3C Specification **1'b1:** MIPI I3C Basic Specification ***Note:*** *The I3C Basic Specification should be regarded as a subset of the I3C Specification.* **Bits[3:0]: I3C Basic Specification Minor Version** (v1.Y) **4'b0000:** Illegal value, do not use (It would encode Version 1.0, but the SETBUSCON CCC was not available in I3C v1.0.*Z*) **4'b0001** – **4'b1111:** Version 1.1 – Version 1.15 **Examples:** Bit[5] Bit[4] Bits[3:0] **I3C v1.1.0:** 1'b0 || 1'b0 || 4'b0001 or 8'b00000001 **I3C v1.1.1:** 1'b1 || 1'b0 || 4'b0001 or 8'b00100001 **I3C v1.2.0:** 1'b0 || 1'b0 || 4'b0010 or 8'b00000010 **I3C Basic v1.1.1:** 1'b1 || 1'b1 || 4'b0001 or 8'b00110001 **I3C Basic v1.2.0:** 1'b0 || 1'b1 || 4'b0010 or 8'b00010010
64 – 127	Other MIPI Working Groups	Reserved for higher-level protocols defined by other MIPI Alliance specifications that use MIPI I3C Basic for communications
128 – 191	Other Standards Organizations	Reserved for higher-level protocols defined by other standards developing organizations. The values are assigned by MIPI Alliance I3C WG. See public registry at https://www.mipi.org/MIPI_I3C_bus_context_byte_values_public.html ***[MIPI07]***.
192 – 255	Vendor Custom	Available for private, per-vendor use (not tracked by MIPI Alliance)

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.7.4 Direct CCCs and Group Addresses

If a Target supports Group Address capabilities (see ***Section 4.3.4.4***) and has been assigned a Group Address, then it may receive and acknowledge certain Direct SET or Direct Write CCCs sent to an assigned Group Address, for Direct CCCs that are either defined in ***Table 16***, or otherwise reserved for other MIPI WGs or custom definition. The Controller may send a Direct CCC as a Write to a Group Address, such that all Targets in the Group may receive and acknowledge such Direct CCCs (if supported). Each such Target that supports such CCCs shall ACK the Direct CCC as a Write sent to an assigned Group Address, as indicated in this section, using the same manner for acknowledging a Direct CCC as a Write that would be sent to its assigned Dynamic Address (per ***Section 4.3.7.2.2***).

Note:

If no such Targets ACK a Direct SET or Direct Write CCC that is sent to an assigned Group Address, then the Controller shall detect this as a NACK (i.e., a failure to ACK the CCC).

Table 42 specifies which Direct CCCs are either recommended for use (with possible limitations) or not supported for use with a Group Address.

- A Target that supports any of these CCCs marked as ‘Not supported’ that might otherwise be supported (i.e., when sent to its assigned Dynamic Address) shall not acknowledge the same CCC when sent to a Group Address to which it is currently assigned.
- A Target that supports any of these CCCs marked as ‘May use as multicast’ may choose to acknowledge the CCC when sent to either its Dynamic Address or any assigned Group Address, with identical meaning. Such ‘multicast’ writes may be used for convenience of configuration.
  - The Controller should follow up with a Direct GET form of the same CCC (if applicable) to each Target’s Dynamic Address to confirm that the Direct SET to the Group Address was accepted.
  - If a Target does not support such a CCC when sent to a Group Address, then the Controller might not see its lack of acknowledgement.
- A Target that supports any of these CCCs marked as ‘Limited support’ or ‘Not recommended’ might support Group Addresses for certain formats or when used with certain Defining Bytes, for particular use cases only. In such cases, the Target might not support the CCC in the same manner as though it were sent to its Dynamic Address.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

3091 **Table 42 Direct CCC Support for Group Addresses**

Command Name	Status	Notes and Limitations
ENEC, DISEC	May use as multicast	Note that the event conditions that are controlled by these CCCs relate to Target Interrupt Requests, which cannot be sent from a Group Address.
ENTAS0, ENTAS1, ENTAS2, ENTAS3	May use as multicast	Activity State configuration generally applies to the whole Device. Note that many uses of these CCCs are to communicate expected latencies to expect in response to pulling SDA Low, which do not apply to an assigned Group Address, since a Target Interrupt Request cannot be generated from a Group Address.
SETMWL	May use as multicast	Maximum write length applies to the whole Device, not an individual Group Address. No standard method is defined to read the maximum write length for only a Group Address, as opposed to a Dynamic Address.
SETMRL	*Not recommended*	Multicast configuration not recommended. Maximum read length does not usually apply, since Controller cannot read from a Group Address.
SETDASA, SETNEWDA	*Not supported*	CCCs that assign or change a Dynamic Address cannot be sent to any Group Address.
SETBRGTGT	*Not supported*	Group Addressing is not defined for a Bridge Device.
SETROUTE	*Not supported*	Group Addressing is not defined for a Routing Device.
SETXTIME	May use as multicast	Timing Control settings apply to the Device as a whole. Note that any Timing Control settings sent to a Group Address should be used for multicast configuration as an alternative to the Broadcast SETXTIME CCC. Controllers should first confirm that all Targets in a Group support the same Timing Control Mode before using the SETXTIME CCC with a Group Address.
ENDXFER	May use as multicast	Controller should confirm that applied settings are accepted, using Direct GET CCC to each Target's Dynamic Address; see ***Section 6.2.3.4.4***.
D2DXFER	*Not supported*	Device to Device Transfers are addressed to a Receiver Address, not a Group Address. See ***Section 6.9***.
RSTACT	Limited support, for defined uses only.	Target Reset Actions are not defined for a Group Address. Specific Defining Bytes might be used with Group Addresses, for some special use cases. Recommend using Dynamic Addresses, unless specific situations require Group Addresses.
SETGRPA	*Not supported*	SETGRPA CCC may only be sent to a Target's Dynamic Address.
RSTGRPA	Limited support, for defined uses only.	In one form, removes all Targets from this Group Address; see ***Section 4.3.7.3.25***.
MLANE	Limited support, for defined uses only.	Sets the ML Frame format for subsequent transfers addressed to the Group Address for an I3C Mode. May only be sent if all Targets in the Group support separate ML configurations (per ***Section 6.7.3.1***).
MIPI WG Reserved	Not yet defined	Reserved for use by other MIPI Alliance Working Groups. Support for Group Addressing might be defined in another MIPI specification.
Vendor / Standards Extension	For Vendor or Standards use	Generally available for use (see ***Table 16***), but MIPI Alliance recommends careful consideration based on the intended use case. Contact MIPI Alliance I3C Working Group for guidance and specific recommendations.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Note:

*For any Direct CCCs listed in **Table 42** with either full or partial support for a Group Address, such CCC definitions in the respective sub-sections of **Section 4.3.7.3** that only have a Direct SET or Direct Write CCC form using "Target Addr" in the Direct CCC framing (i.e., after the Repeated START, or "Sr") should also be interpreted as supporting a valid Group Address with that particular form, where a Group Address may be used in place of the "Target Addr" in the framing (see **Section 4.3.2.1.3**).*

However, any CCCs that are listed as 'Limited support' or 'Not recommended' should generally not be interpreted as supporting a Group Address in place of the "Target Addr" (per notes and limitations). Additionally, any CCCs that are listed as 'Not supported' do not support a Group Address in place of the "Target Addr".

The Controller shall not send any Direct GET or Direct Read CCCs to a Group Address, and Targets shall not ACK any Direct CCCs that would attempt to read from an assigned Group Address.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

4.3.8 Error Detection and Recovery Methods for SDR

The error detection and recovery methods specified in this Section are provided in order to avoid fatal conditions when errors occur. A set of required methods is specified for I3C Basic Target Devices, and a separate set of required methods is specified for I3C Basic Controller Devices. Origins for all of these SDR Error Types are shown in ***Figure 177*** through ***Figure 180***.

4.3.8.1 SDR Error Detection and Recovery Methods for I3C Basic Target Devices

The Error Types summarized in ***Table 43*** shall be supported for all I3C Basic Target Devices. Each Error Type is further explained in a sub-section below the table.

Note:

In previous versions of the I3C Basic Specification, these Error Types for I3C Basic Target Devices were named "S0", "S1", etc. The purpose, detection methods, and recovery methods for these Error Types are unchanged.

Table 43 SDR Target Error Types

Error Type	Description	Error Detection Method	Error Recovery Method
TE0	**Invalid Broadcast Address/W** (7'h7E/W) or **Dynamic Address/RnW 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[1]	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 TE0/TE1 error situation.
TE1	**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 TE0/TE1 error situation.
TE2	**Write Data**	Parity Check, using T-Bit	Enable STOP or Repeated START detector and neglect other patterns.
TE3	**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 Provisioned ID.
TE4	**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
TE5	**Transaction after detecting CCC**	Detect illegally formatted CCC	Generate NACK (after Target Address), then enable STOP or Repeated START Detector and ignore all other patterns
TE6 (optional)	**Monitoring Error**	Target 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 (per Read type) and ignore all other patterns
DBR (optional)	**Dead Bus Recovery**	Controller acting as Target detects that the I3C Basic Bus is no longer being driven by a Controller	Controller acting as Target retakes control of the I3C Basic Bus, becoming Active Controller

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Note:

1. *In the ENTDAA mode, "7'h7E / R" is excluded from the TE0 error definition.*

4.3.8.1.1 Error Type TE0

If an error occurs during Broadcast Address/W or Dynamic Address/RnW after a Dynamic Address is assigned, then the Target will be unable to distinguish whether the transfer is a CCC transfer or a Private RnW transfer. For example, in the case of an ENTHDR CCC transfer the Target is not able to know that the I3C Basic Bus has changed to HDR Mode. A potentially fatal situation could ensue if this case is not detected and handled, because the Target might become confused by seeing many STARTs and STOPs as illustrated in ***Figure 65***, and might attempt to interpret the HDR transfer as though the I3C Basic Bus were still in SDR Mode.

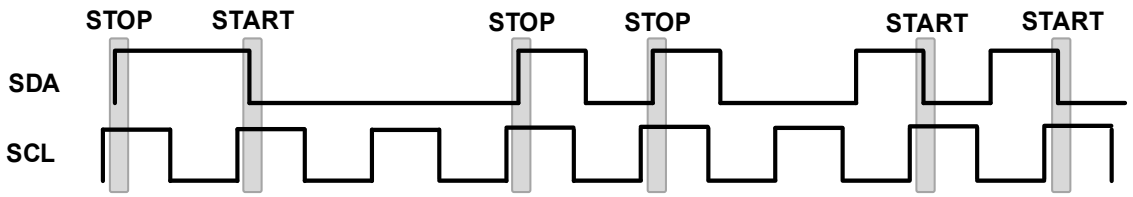


Figure 65 Example Waveform for Error Type TE0

In order to avoid this situation, the Controller shall not use any of the possible error case Addresses: 7'h7F, 7'h7C, 7'h7A, 7'h76, 7'h6E, 7'h5E and 7'h3E.

- The Controller shall not assign any of these Addresses to any Target as a Dynamic Address (or a Group Address, if supported) per the I3C Basic Target Address restrictions in ***Table 10*** (see ***Section 4.3.2.2.5***).
- The Controller shall not use any of these Addresses in an I3C Basic Address Header with the RnW bit of 1'b0, because the Target cannot distinguish such an I3C Basic Address Header from a write to the Broadcast Address (i.e., 7'h7E/W) that has a single bit error.

Except during a Dynamic Address Arbitration procedure, the Target shall consider receipt of any of these restricted Addresses (i.e., 7'h7F/W, 7'h7C/W, 7'h7A/W, 7'h76/W, 7'h6E/W, 7'h5E/W and 7'h3E/W), or the receipt of 7'h7E/R, as an error per the following conditions:

- The Target shall always use its Error Type TE0 detector after a Dynamic Address has been assigned, to monitor the I3C Basic Address Header for SDR Mode transfers.
  - This generally applies to any I3C Basic Address Header that follows either a START or a Repeated START (see ***Figure 177***, ***Figure 178***, and ***Figure 179***).
  - However, this shall not apply during the Dynamic Address Assignment procedure, if the Controller uses the ENTDAA CCC (per ***Section 4.3.4.2***). If this occurs, then the Target shall use its Error Type TE4 detector instead (per ***Section 4.3.8.1.5***) until the end of the Dynamic Address Assignment procedure (see ***Figure 180***).

If the Target's Error Type TE0 detector detects such a single bit error in an I3C Basic Address Header, then the Target shall ignore the rest of the signal until either:

- The Target detects the HDR Exit Pattern; or
- Optional: The Target detects that both SCL and SDA stay High for a period greater than 60 µs (see ***Section 4.3.8.1.9***).

Note:

The second method is optional, i.e., Target Devices are not required to support it. If this method is supported, then the Target shall enable its STOP or START detector to exit from the Error Type TE0 state. If supported, then this method may also be used for Error Type TE1.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.8.1.2 Error Type TE1

If the Target detects a parity error during a CCC code, then the Target will not be able to know that the I3C Basic Bus has changed to HDR Mode if the CCC is ENTHDR. This is similar to the situation in Error Type TE0.

In order to avoid this situation, if the Target detects a parity error during a CCC code, then the Target shall ignore the rest of the signal, until either:

- The Target detects the HDR Exit Pattern, or
- Optional: The Target detects that both SCL and SDA stay High for a period greater than 60 µs (see ***Section 4.3.8.1.9***).

Note:

The second method is optional, i.e., Target Devices are not required to support it. If this method is supported, then the Target shall enable its STOP or START detector to exit from the Error Type TE0 state. If supported, then this method may also be used for Error Type TE0.

4.3.8.1.3 Error Type TE2

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

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

1. Retain the CCC state until the Target detects the end of CCC command (i.e., when the Target is recovered by the Repeated START), or else
2. Disregard everything until the STOP.

4.3.8.1.4 Error Type TE3

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

4.3.8.1.5 Error Type TE4

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

4.3.8.1.6 Error Type TE5

If the Target detects an illegally formatted CCC, then the Target shall generate NACK (after the Dynamic Address) and then wait for STOP or Repeated START.

Examples of an illegally formatted CCC include:

- If the Target receives a matching Dynamic Address with Write during a Direct CCC that only has a Direct Read or Direct GET form (e.g., the GETBCR CCC)
- If the Target receives a matching Dynamic Address with Read during a Direct CCC that only has a Direct Write or Direct SET form (e.g., the SETGRPA CCC)

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

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

Note:

This Section does not provide all examples of illegally formatted CCCs.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Note:

*If the Target detects a Direct CCC that is formatted correctly, but that is not supported, then the Target shall handle such CCCs, or unsupported Defining Bytes, as described in **Section 4.3.7.2.2**.*

*In an HDR Mode CCC, the Target would wait for the CCC to end as described in **Section 6.1.3.1**, **Figure 101**.*

4.3.8.1.7 Error Type TE6 (Optional)

If an error occurs in the RnW Bit of the Address Header, then the Target might believe that it is responding to a Read, when what the Controller actually intends to initiate is a Write. If this happens, then the Write Data from the Controller might conflict with the Read Data from the Target.

The Target should always monitor the Data it transmits for Read transactions. If the Target does so, then if the monitored Data differs from the Data the Target intended to transmit, the Target shall consider this condition to be an error.

Note:

*The Controller should also monitor the Data it transmits. If this condition happens due to a Target's misinterpretation of the RnW bit during an intended Write transfer, then the Controller should also detect this condition as Error Type CE1 (see **Section 4.3.8.2.2**).*

*The Target shall not consider this condition to be an example of Error Type TE6 during the Arbitration round of a Dynamic Address Assignment procedure (i.e., when the Controller has sent the ENTDAA CCC) while the Target is attempting to drive its Provisioned ID, BCR and DCR (see **Section 4.3.4.2**.*

If the Target detects such an error during an intended Private Write transfer (i.e., when the Target believes it is responding to a Private Read transfer), then it shall stop the transmission, allow the Controller to finish, and then wait for STOP or Repeated START.

If the Target detects such an error during an intended Direct SET or Direct Write CCC (i.e., when the Target believes that it is responding to a Direct GET or Direct Read CCC), then the Target shall stop the transmission, allow the Controller to finish, and then either:

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

4.3.8.1.8 Error Type DBR (Optional)

This error allows a Controller-capable Device that is acting as a Target (i.e., a Secondary Controller) to regain control of a dead Bus (i.e., where the Active Controller has stopped working for whatever reason). This works both for a Primary Controller acting as Target (i.e., while a Device that initialized as a Secondary Controller is acting as Active Controller), and for a Secondary Controller acting as a Target.

The general model is that when the Controller acting as Target pulls SDA Low to request an IBI or CRR, it may measure the time before SCL goes Low (i.e., in response to this START Request). If SCL does not go Low in 50 ms (t_{CAS} maximum for Activity State 3), then the Controller acting as a Target may take action to take over control of the Bus and become the new Active Controller.

The steps are as follows:

1. The Controller acting as Target pulls SDA Low to initiate an IBI or CRR.
   A. It starts a timer.
2. If 50 ms has elapsed without SCL going Low, then the Controller acting as a Target assumes that the former Active Controller is not operational.
   A. If SCL is pulled Low before 50 ms, then all is well and Error Type DBR shall not apply.
   B. Otherwise, Error Type DBR applies and as a result the Target acting as Controller may start a transition to Controller mode.

See Errata 01, Item 10

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3. Once in Controller mode, the Device shall verify that SCL is still High, and if so, pull SCL Low to complete a START.
   A. If SCL is not High by the time the Device is a Controller, then the Device must switch back to Target mode, as Error Type DBR no longer applies. This might occur due to another Controller-capable Device (i.e., a Secondary Controller) having taken control of the Bus first.
4. After the START, the Controller may wish to check the status of the Targets.
   A. It may also initiate an ENTDAA sequence to see whether any Devices need a Dynamic Address to be assigned (see ***Section 4.3.4.2***).
   B. If available, the last DEFTGTS information that might have been sent by a previous Active Controller (i.e., as a Broadcast CCC) should be used for proper Dynamic Address assignment.

Note:

A Controller-capable Device acting as Target may choose to monitor the Bus at all times. In the event that SDA is pulled Low by any Target, the Device may then measure the time waiting for SCL to be pulled Low. If 50 ms elapses without SCL being pulled Low, then it may start the Dead Bus Recovery process above at step 3.

If the former Active Controller eventually resumes operation and senses activity on SCL and/or SDA, then it must assume that it has lost the Controller Role to the new Active Controller.

*Per **Section 4.3.9.4.1**, the Primary Controller shall use a similar procedure after receiving a Whole Target reset, to determine whether it should be the Active Controller of the Bus.*

4.3.8.1.9 Optional Recovery Method for Error Types TE0 and TE1

An I3C Basic Target can recover from an Error Type TE0 or Error Type TE1 situation not only by detecting the HDR Exit Pattern, but also by monitoring the SCL and SDA lines. If the Target detects that both lines stay at High level for a period exceeding 60 µs, then the I3C Basic Target can regard the Bus as operating in non-HDR mode. The I3C Basic Target can then recover from the TE0 or TE1 situation, and wait for a STOP or a START to resume normal operation.

Note:

Regarding timing, HDR's slowest clock rate is 10 kHz (100 µs total cycle). The period 60 µs is derived by assuming that HDR (i.e., HDR-DDR Mode) will always keep an approximately even duty cycle, especially at very slow clock rates (since the only reason to go so slow is for long lines and/or large capacitive load on Bus lines). 60 µs represents 60% of the duty cycle, and therefore is a safe duration to wait when seeing both lines High.

The Target can start measuring the period whenever it detects that both SCL and SDA are at High level. There is no need to start timing this period at any particular signal pattern (for example, at the STOP).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.8.2 SDR Error Detection and Recovery Methods for I3C Basic Controller Devices

Table 44 summarizes the defined Error Types for I3C Basic Controller Devices. All I3C Basic Controller Devices shall support Error Types CE0, CE2, and CE3, and should support Error Type CE1. Each Error Type is further explained in a sub-section below the table. Escalation and recovery are explained in ***Section 4.3.8.2.5*** and ***Section 4.3.8.2.6***.

Note:

In previous versions of the I3C Basic Specification, these Error Types for I3C Basic Controller / Controller Devices were named "M0", "M1", etc. The purpose, detection methods, and recovery methods for these Error Types are unchanged.

Table 44 SDR Controller Error Types

Error Type	Description	Error Detection Method	Error Recovery Method
CE0	**Transaction after sending CCC**	Detect illegally formatted CCC	Stop the transmission, then send STOP and retry the transmission.
CE1 (optional)	**Monitoring Error**	Controller 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.
CE2	**No response to Broadcast Address** (7'h7E)	Controller detects NACK after Broadcast Address (7'h7E) transmission	Upon detection of NACK, Controller transmits HDR Exit Pattern followed by STOP
CE3	**Failed Controller Handoff**	Active Controller detects New Controller does not drive Bus after handoff	Active Controller regains the Controller Role and drives Bus to assert its Controller Role

4.3.8.2.1 Error Type CE0

If the Controller detects an illegally formatted CCC, then the Controller shall stop the transmission, send STOP, and retry the transmission. An example of an illegally formatted CCC would be the Controller receiving just one byte from the Target in a GETMWL CCC code, since the Controller expects two bytes.

4.3.8.2.2 Error Type CE1 (Optional)

Error Type CE1 occurs when the I3C Basic Controller detects that the transmitted data differs from what the Controller intended to transmit. This might happen when the I3C Basic Controller or I3C Basic Targets misinterpret the RnW bit or the ACK/NACK during transactions (including IBI) defined by the I3C Basic specification. This Section describes only two kinds of occurrence conditions for Error Type CE1, and the recovery method for each kind. Note that Type CE1 errors are not an expected behavior.

If an error occurs in a RnW Bit in a Private Write transfer, then the Write Data from the Controller might conflict with the Read Data from the Target. For example, the Target could misinterpret a Private Write transfer as a Read transfer if there is an error in the RnW Bit, and as a result Write Data from the Controller would conflict with Read Data from the Target.

The Controller should always monitor the Data it transmits. If the Controller does so, then if the monitored Data differs from the Data the Controller intended to transmit (except for Data transferred during a Dynamic Address Arbitration procedure), the Controller shall consider that to be an error. If the Controller detects this error, then it shall stop the transmission, then send STOP and retry the transmission.

Copyright © 2016ï 2025 MIPI Alliance, Inc.
Public Release Edition

4.3.8.2.3 Error Type CE2

If the Controller does not receive an ACK of a transmitted Broadcast Address (7'h7E), then it shall transmit the HDR Exit Pattern followed by STOP in order to recover any Target after TE0, TE1, TE2, TE5, and TE6 errors.

4.3.8.2.4 Error Type CE3

After the Controller to Controller Handoff procedure (see ***Section 4.3.6.3***), the former Active Controller shall release SCL to High-Z and disable its Open Drain class Pull-Up on SDA, setting SDA to High-Z. However, the former Active Controller shall monitor the Bus to ensure that the new Controller has successfully asserted its Controller Role.

If the former Active Controller does not detect a START within a specified Handoff period, then it shall test the New Controller to determine whether it actually controls the Bus, and it shall regain control of the Bus if the New Controller has not asserted its Controller Role. The specified Handoff period shall be at least 100 µs, but may be longer if the new Controller indicates that it requires a longer Handoff period.

Before initiating the Controller to Controller Handoff procedure, the Active Controller shall first determine whether the Secondary Controller that is about to receive the Controller Role supports the GETMXDS CCC with Defining Byte value CRHDLY (see ***Section 4.3.7.3.18***). If the Secondary Controller needs a longer Handoff period to assert the Controller Role, then it shall do so by indicating an Activity State. The Active Controller shall wait for the time period associated with the indicated Activity State, or 100 µs (whichever is greater) before attempting to test the new Controller.

Once the Handoff period has elapsed, if the former Active Controller has not detected a START, then it should use the following steps to test the new Controller:

5. The former Active Controller shall pull SDA Low. This may be unnecessary if another I3C Basic Target also pulls SDA Low (since the Handoff period is longer than the Bus Available Condition), but the effect is the same. **See Errata 01, Item 11**
6. After SDA is pulled Low, the former Active Controller shall wait for another 100 µs, or the time indicated by the new Controller's indicated Activity State (as above, read with the GETMXDS CCC with Defining Byte value CRHDLY, if supported), whichever is greater.
   A. If the new Controller responds by pulling SCL Low within that period, then all is well: the new Controller has successfully asserted its Controller Role of the Bus. This is equivalent to a START, so this flow has a successful outcome. The former Active Controller shall release SDA and allow the new Controller to proceed.
   B. If the new Controller does not respond by pulling SCL Low within that period, then the former Active Controller shall pull SCL Low to regain the Bus Controller Role. This is equivalent to a START, but initiated in this case by the former Active Controller.

Note:
This outcome is an error condition on the part of the new Controller, which has not responded per its obligations. As a result, the new Controller shall return to being a Target, or remain as a Target.

In either case, the Controller that eventually 'wins' shall become the Active Controller, and shall drive a valid Bus action to assert its Bus Controller Role. Since the START has been driven, the next step is the I3C Basic Address Header (see ***Section 4.3.2.2***). The Active Controller shall attempt to drive its own Address (7'h7E). If another I3C Basic Target drives its own Address and wins Arbitration, then that Target may attempt to issue a Target Interrupt Request. However, if the Active Controller wins Address Arbitration but does not have any actions to take at that time, then it may follow with a STOP. Either is sufficient for the Active Controller to assert its Bus Controller Role.

If the former Active Controller regains the Bus Controller Role due to inactivity by the new Controller, then it is responsible for taking any necessary actions to determine the reason why the handoff failed. It may use

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

the GETSTATUS CCC or other commands to test for the presence of the new Controller, or to attempt to read its current status.

4.3.8.2.5 Controller Error Detection and Escalation Handling

If SDA is stuck Low, then see ***Section 4.3.8.2.6***. This Section addresses situations in which the Target is not responding.

If the Controller does not receive an ACK of a transmitted private Message to a Target, and if the following conditions are all true:

- Activity State is 0
- Either the Target Device has not indicated read-turnaround delays via GETMXDS, or the Target Device has indicated a GETMXDS period and a period longer than GETMXDS has elapsed
- The Target Device has not notified the Controller 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 Controller has the following escalation options to recover the system:

1. If the Controller is aware that the Target might sometimes need extra time, then the Controller can choose to try again after a short delay.
2. If that fails, then the Controller shall check whether the Target is responsive by issuing GETSTATUS to the Target. The idea here is that the Target 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 Controller shall use CE2 error handling (see ***Section 4.3.8.2.3***) to emit the sequence:

   S | 7'h7E (W) | ACK/NACK* | HDR Exit Pattern | STOP

   Note:

   In order to avoid any conflict with an incoming IBI from Target Devices on the Bus, the Controller should send the Broadcast address before the HDR-Exit Pattern. The I3C Basic Controller doesn't care whether the I3C Basic Targets' response is ACK vs. NACK.

   This ensures that the failure is not due to the Target thinking that the I3C Basic Bus is in HDR Mode.
4. If that fails, and if the Target is still not responsive, then the next step depends on the value of the BCR ñOffline Capableò bit (Bit[3]):
   A. If the Target is not Offline Capable, then:
      i. The Controller can try slowing the SCL clock rate, or try slowing its effective rate by using duty cycle.
      ii. If that fails, then the Target Reset mechanism can be used. A first attempt can be tried to recover via the Peripheral, and if that fails, then a second attempt will cause a chip reset. The Target Reset mechanism is explained in ***Section 4.3.9***.
      iii. If that fails, then any further escalation is outside the scope of the I3C Basic Specification.
   B. If the Target is Offline Capable, then:
      i. If the Target goes offline and is to be awoken via Target Reset, then the Target Reset approach should be used (see ***Section 4.3.9***).
      ii. If the Target is known (i.e., via a private contract) to have a long wake period, and also known to monitor the I3C Basic Bus during that wake period, then the Controller simply delays and tries later, after a delay based on the known or expected wake up time. Escalation and the GETSTATUS ensure that the Targetôs Dynamic Address was issued; that should serve as the wake trigger. Either the Target may issue an In-Band Interrupt at a later time to notify the Controller that it is ready, or else the Controller may initiate the retry.

Copyright É 2016ï 2025 MIPI Alliance, Inc.

iii. If Target is not known to have a long wake period and to always monitor the I3C Basic Bus, then the Controller marks the Target as offline. The Controller may then either retain the Target's Dynamic Address for re-use, or else discard it. The next action will be for the Target to issue a Hot-Join Request, in order to be re-joined to the I3C Basic Bus. In the Hot-Join operation the Controller will assign the Target a Dynamic Address; the new Address may be either the same Address that the Target used before being marked as offline, or a different Address.

4.3.8.2.6 Controller Stuck SDA Handling

If SDA is not stuck, but the Target is not responding, then see ***Section 4.3.8.2.5***. If the Controller has recovered from a crash or unexpected reset, then see ***Section 4.3.8.2.7***.

The steps to attempt a recovery from a stuck SDA are as follows:

1. If reading from an I^2C Device, and it is holding SDA Low, then:
   A. Pulse out nine (9) SCL clocks, to get it to see the NACK on 9th bit to get it to let go.
2. If reading from an I3C Basic Device in SDR Mode, and it is holding SDA Low, then:
   A. Pulse out one clock at a time (up to 8), as it is required to drive SDA High for the T-Bit (9^{th} bit).
   B. Watch for SDA going High, and abort the read by driving SDA Low when SCL is High.
3. If reading from an I3C Basic Device in SDR Mode, and SDA is being held Low, and the approach in step 2 did not work, then:
   A. Hold SCL level (High or Low) for 150 µs. The Target might support the 150 µs read-abort approach (per ***Section 4.3.2.3***), and if so, it will release SDA.
4. If reading from an I3C Basic Device in SDR Mode, and SDA is High, and the Device does not seem to abort when the Controller drove SDA Low on the T bit (i.e., when SCL was High), then:
   A. First try a 150 µs SCL hold (as explained in step 3).
   B. If that does not work, then do the same drive SDA Low for each SCL High, for up to 8 more times. If there is a framing misalignment, then this will ensure that the T-Bit is finally caught and the read aborted.

   Note:

   This risks contrary drive, so Controller could lower its drive strength to reduce current, if wanted.
5. If reading from an I3C Device in HDR-DDR Mode, and it goes wrong, then see ***Section 6.2.3.5***.

Note:

*The steps defined in this section do not constitute any form of Target reset procedure. There is no expectation that un-sticking the SDA line with these steps will induce a non-responding Target to reset any of its logic or other state. However, the I3C Basic Specification does already define an existing Target Reset mechanism; see **Section 4.3.9**.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.8.2.7 Controller Recovery After a Crash or Unexpected Reset

If the Controller is recovering from a crash or unexpected reset, and it does not know the context of the Bus, then the Controller must determine the state of the Bus. It is assumed that the Controller has tried to put the Bus into Bus Free condition (i.e., SCL High, SDA held High with a Pull-Up), and that enough time has elapsed for the Pull-Up to pull SDA High, if possible.

The status may be one of:

- **SDA is High and controlled by the Controller alone.** This is normal Bus Free Condition. In this case the Controller should emit an HDR Exit Pattern (to be safe), and then work to recover the Bus in the usual manner (e.g., RSTDAA, etc.).
- **SDA is High, but a Target is holding it High from a previous Read.** See below for detection.
- **SDA is Low.** This might be a Target still holding Low from a previous Read, or it might be a Target pulling SDA Low for an In-Band Interrupt, a Controller Role Request, or a Hot-Join.

If SDA is High, and the Controller needs to determine whether it is held High, then the Controller may use one of three ways to do so:

1. The Controller uses a weak Pull-Down resistor in its pad for SDA and turns off any active Pull-Up. If the line goes Low (after a reasonable time), then it is controlled by the Controller, and the Controller may proceed normally.
2. The Controller may drive SDA Low, risking contrary drive (e.g., 4 mA current), to determine whether the SDA is held High.
3. The Controller may lower the drive current of the SDA pad and then drive Low, to minimize contrary drive effects.

If the Controller determines SDA to be held High, then follow the instructions in ***Section 4.3.8.2.6***, step 4. This method can be used for both an SDR Read and an HDR-DDR Read, however an HDR-DDR Read will need 16 clocks (10 for data preamble, 16 in case last data into CRC).

If SDA is Low, then the Controller needs to determine why it is held Low. The Controller can use the following steps to do so:

1. The Controller drives SCL Low. If SDA is released (i.e., strong Pull-Up on SDA causes it to go High), then a Target was trying to perform an In-Band Interrupt, a Controller Role Request, or a Hot-Join. The Controller may pull SDA Low again and continue with a START, or emit a STOP (i.e., drive SCL High, then pull SDA High), and then proceed to restart the Bus by NACKing any In-Band Interrupt, Controller Role Request, or Hot-Join, and then using the RSTDAA CCC.
2. If SDA is not released, then the Controller should emit 19 or more SCL clocks. If SDA goes High at some point, then it should emit more SCL clocks to ensure 19 clocks were used with SDA High (in case it was in HDR-DDR Mode). If SDA is now High, then the Bus is not in HDR-DDR Read and also not in an I^2C Read. The Controller should test whether SDA is stuck High, as explained above (starting at “If SDA is High”). If SDA is not stuck High, then the Controller should issue an HDR Exit Pattern in case it had been in HDR-DDR Mode.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.9 Target Reset

This Section specifies the Target Reset mechanism.

The Target Reset mechanism:

- Allows the Controller to Reset one or more selected Targets, and avoid resetting any others
- Supports different levels of reset, ranging from resetting only the I3C Basic Peripheral within a Target, to resetting the whole Target Device
- Supports I3C Basic's error escalation mechanism (see ***Section 4.3.8***), including reset of an errant Target

The Target Reset mechanism is simple, relying on these principles:

- The uniqueness of the HDR Exit Pattern (i.e., cannot be confused with any other I3C Basic Bus traffic)
- Use of an in-band Target Reset Pattern to trigger the actual reset action. (In contrast to a reset/break hold as used in UART, one-wire [R], etc., which would require timing.)
- A defined default reset action that all Targets will use until and unless a different reset action is configured via the RSTACT CCC (Broadcast and Directed formats)
- An error recovery escalation mechanism, for use when needed (e.g., Error Types TE0 and TE1).

The Target Reset mechanism applies these principles to address three distinct reset cases:

1. Recovery from error/hang in the Target I3C Basic Peripheral
2. Recovery from more serious errors in the chip containing the I3C Basic Peripheral
3. Wake from deepest sleep, while minimizing the number of gates used in the always-on domain

4.3.9.1 Theory of Operation

The Target Reset mechanism uses a simple model (***Figure 66***). In normal operation the Controller directs the Targets in their actions, and the Targets take those actions. When the Target is in a deep enough sleep, or is broken, then the Target Reset performs an appropriate action. The Controller uses the RSTACT CCC (Broadcast and/or Directed formats, as needed) to configure which Targets are to be reset, the level(s) of reset to be used, and which Targets are not to be reset.

Controller: To trigger the Reset action, the Controller shall emit the following sequence:

- START
- Zero or more Message components, including RSTACT CCCs (Broadcast and/or Directed, as needed) each optionally ending in Repeated START
- At this point in the sequence the Controller shall not emit a STOP; the Controller shall instead keep the sequence within a single Frame (see Note below).
- The Target Reset Pattern, including the final Repeated START and STOP that trigger the actual reset action
- At this point in the sequence the Controller shall leave the Bus free for as long as needed (or desired) for the Targets to complete their reset cycles.
- START, with an I3C Basic Address Header (directed to either 7'h7E or any other Address)

Note:

It is mandatory to emit the RSTACT CCCs and the Target Reset Pattern together in a single Frame (i.e., before emitting a STOP) for several reasons:

- *To avoid the risk of the RSTACT getting disconnected from the Target Reset action, which would be dangerous. A properly operating Target will cancel an RSTACT CCC when it sees an SCL falling edge following a START (but not a Repeated START).*
- *Because neither I3C Basic nor I^2C define the Bus condition of SCL falling from Bus Free, and as a result some Devices might exhibit unknown or unexpected behavior.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

- *Because the Target Reset Pattern includes STOP, in order to preserve a clean Bus state for all Targets (i.e., both Targets that are being reset, and Targets that are not being reset).*
- *A Target could pull SDA Low after a STOP.*

Target: A Target shall react to receipt of the Target Reset Pattern based on its state at the time of reception:

- If the Target had been configured via the RSTACT CCC to take a particular reset action (including taking no action) and was able to process that RSTACT CCC (i.e., not broken), then the Target shall perform the reset action as configured.
- If the Target had been in a deepest-sleep state, then it may Wake up (e.g., power up, etc.). This Wake operation itself may either reset the Device or not, per ***Section 4.3.9.5***.
- If a Target had been unable to recognize a RSTACT CCC (i.e., was broken), or if no RSTACT CCC had been sent to the Target, then the Target shall either:
  - **If this is the first time that the Target has seen the Target Reset Pattern:** Then the Target shall reset the I3C Basic Peripheral. If a processor is in use, then it will notify the application via internal interrupt, so that it can reconfigure the I3C Basic Peripheral). **See Errata 01, Item 12**

  or

  - If the Target has previously reset the I3C Basic Peripheral and is seeing a new Target Reset Pattern with no intervening RSTACT or GETSTATUS CCC: Then the Target shall reset the whole chip, i.e., equivalent to an SRSTn pin being asserted (see ***Section 4.3.9.4***).

Note:

Any reset action (including inaction) configured via the RSTACT CCC shall be cleared by the next SCL falling edge following a START (but not by the next Repeated START).

*This specification is silent regarding which Target internal memory is retained vs. cleared in a Peripheral reset (where 'memory' could be either volatile or non-volatile, and could be implemented in any number of ways). See **Table 45** for more details. In particular, the Target's Dynamic Address may be either retained or cleared. If the Target clears its Dynamic Address, then it shall participate in an ENTDAA that follows the Target Reset Pattern; if the Target retains its Dynamic Address, then it shall not participate.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

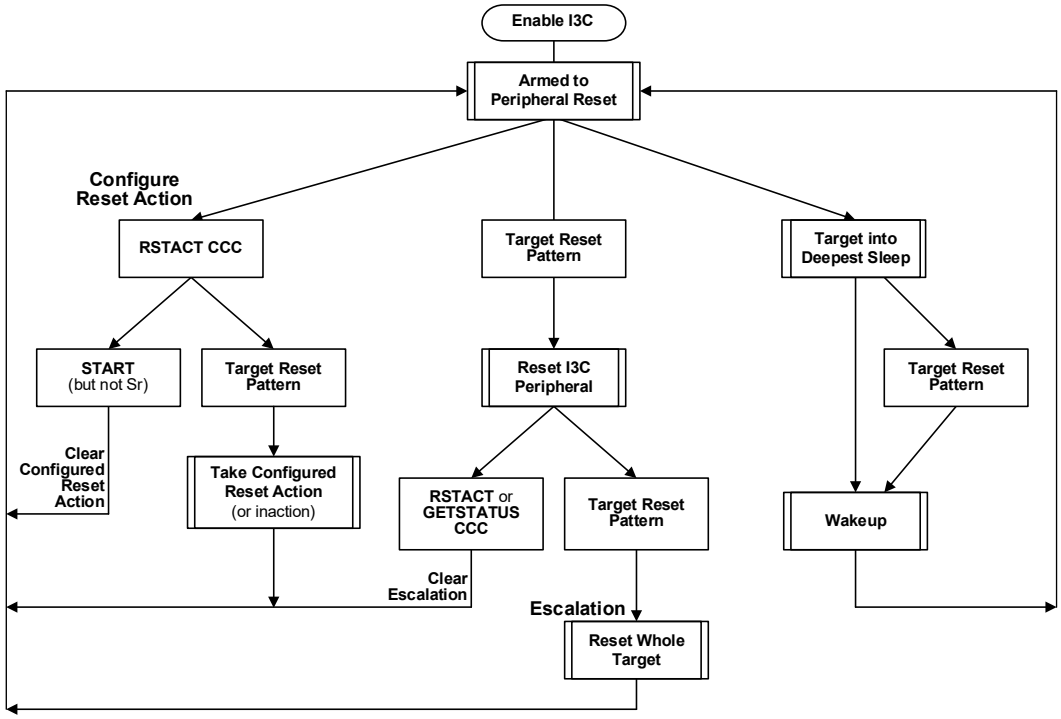


Figure 66 Target Reset Operations Flow

4.3.9.2 RSTACT CCC

The RSTACT CCC (fully detailed in ***Section 4.3.7.3.23***) has three formats with different uses:

- **Broadcast** to configure a Target Reset action for all Targets
- **Directed Write** to configure a Target Reset action for one or more specified Targets
- **Directed Read** to read the Reset recovery timing from each Target, including for reset of the Peripheral, Wakeup, and reset of the whole Device.

4.3.9.3 Target Reset Pattern

The Target Reset Pattern (***Figure 67***) is used to trigger the default or configured reset action. The Target Reset Pattern begins with fourteen SDA transitions while SCL is kept Low, and ends with a Repeated START followed by a STOP while SCL is kept high which triggers the actual Reset action. The Target Reset Pattern is easily distinguished both from the shorter HDR Exit Pattern on which it is based (see ***Section 4.3.10.2***), and from the HDR Restart Pattern.

Note:

- *Similar to the HDR Exit Pattern and the HDR Restart Pattern, the Controller shall observe the minimum timing while sending the Target Reset Pattern, as defined by t_{DIG_H} minimum value (see **Section 4.3.10.2** and **Section 6.1.2**). This shall apply to all transitions on both SDA and SCL.*
- *If the Target pulls SDA to Low first, before the Controller can pull SCL to Low to initiate the Target Reset Pattern, then the Controller is obligated to handle the situation by cancelling the Target Reset Pattern and driving transitions on SCL as part of the Arbitrable Address Header. Subsequently, the Controller may choose to send the DISEC CCC to disable Target requests before sending the next Target Reset Pattern.*
- *However, if the Controller pulls SCL to Low first, then the Target shall not pull SDA to Low, and must release SDA and wait until the Controller finishes the Target Reset Pattern.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

- *If the Controller is sending the Target Reset Pattern from the Idle state, then the Controller shall monitor the SDA line after pulling SCL Low to make sure that there is no SDA pull down currently initiated by the Target. This could take some time for the Controller to process in the case that an IBI is being initiated at the same time that the SCL line is pulled Low. The controller should add additional delay between the SCL Low and the first SDA transition in order to accommodate propagation delays in the bus. In this scenario, the controller shall cancel the Target Reset Pattern and handle the IBI.*

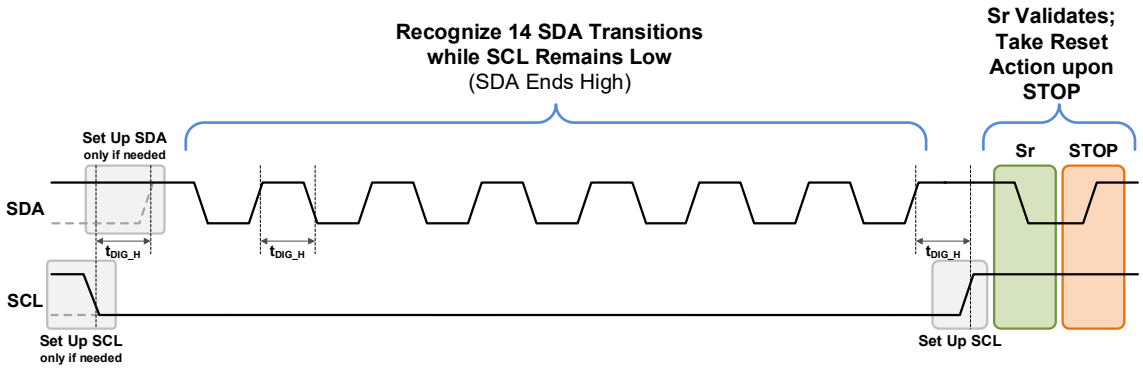


Figure 67 Target Reset Pattern: General Case

If the Target pulls SDA to Low to initiate IBI, Hot-Join, or CRR, then the Controller must cancel and service this request, and then should drive a STOP to end the SDR frame before sending the Target Reset Pattern. Once the I3C Basic Bus is idle, the Controller can send the Target Reset Pattern. The Controller can also drive the Target Reset Pattern at any time, to reset all Targets. ***Figure 68*** shows how the Target Reset Pattern is initiated when the I3C Basic Bus is idle.

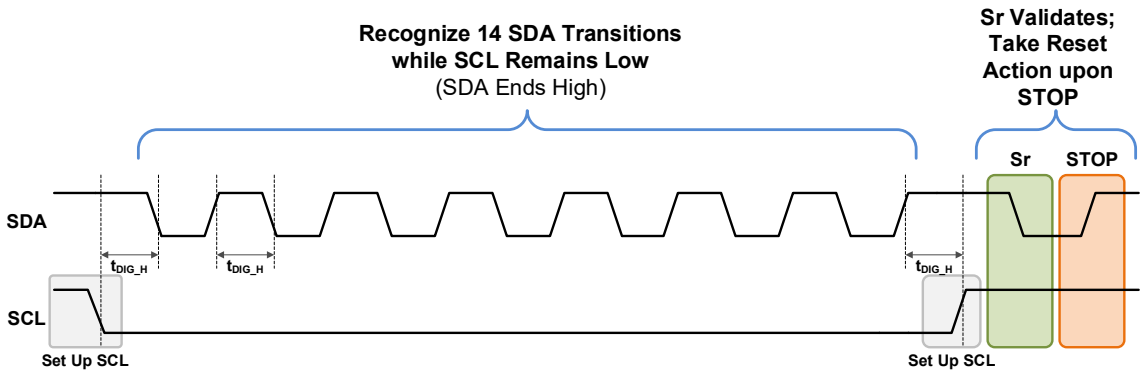


Figure 68 Target Reset Pattern: From Idle

Table 45 lists the scope of the various reset types that are triggered by the Target Reset Pattern, including required and optional behaviors.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Table 45 Scope of Target Reset Actions per Reset Type

RSTACT Defining Byte Value	Reset Type Description	Reset Address? (1)	Reset All Target Configuration? (2)	Reset Other Chip Logic?
0x00	**No Reset on Target Reset Pattern**	No	No	No
0x01	**Reset the I3C Basic Peripheral Only**	Optional	Optional	No
0x02	**Reset the Whole Target** (3)	Required	Required	Optional
0x03	**Debug Network Adaptor Reset**	No	No	Debug logic
0x04	**Virtual Target Detect**	No	No	No
0x05 – 0x3F	Reserved by MIPI	–	–	–
0x40 – 0x7F	Reserved for Vendors and external standards	–	–	–

Note:

In previous versions of the I3C Basic Specification, these requirements were not defined as clearly. I3C Basic Controllers should be aware that older Targets (i.e., those conforming to I3C Basic Specification v1.1.1 and earlier) might have different reset behaviors. However, all new implementations should use the requirements defined in the I3C Basic Specification v1.2 or later.

*Additional requirements may apply to Virtual Targets that are part of a composite Device (i.e., one that has multiple Dynamic Addresses), especially when the Broadcast RSTACT CCC is used. See also **Section 5.2.2.3** in the MIPI Alliance I3C Application Note: Virtual Devices and Virtual Targets **[MIPI10]**.*

Table Notes:

1) If the Target also supports Group Addressing, then resetting the Dynamic Address shall also reset any assigned Group Addresses.

2) This includes all I3C Basic Target configuration that is set by MIPI-defined CCCs, including ENEC/DISEC, ENTAS0-ENTAS3, SETMRL, SETMWL, ENTTM and ENDXFER. For an I3C Basic Peripheral reset, the implementer may determine how many of these are reset to defaults. For a Whole Target reset, all of these should be reset to defaults.

3) Targets are not required to support the Whole Target reset; it is optional. However, if a Target does support the Whole Target reset, then it shall reset all I3C Basic Target logic.

4.3.9.4 Whole Target Reset Behavior

Reset of the Whole Target can be caused either by escalation of default (per ***Figure 66***), or by a configured reset action via the RSTACT CCC with Defining Byte value 0x02 (per ***Table 38***). An I3C Basic Target that receives a Whole Target reset shall clear its entire I3C Basic Target configuration, as though it was fully reset back to its default configuration.

Additionally, implementers may choose to include a full/chip reset as part of the Whole Target reset. In general, a full/chip reset could include other logic within the chip/SoC (i.e., different domains or functions outside the I3C Basic Target logic) which would be reset at the same time as the I3C Basic Target logic.

This Specification does not strictly define behavior for a full/chip reset, however it is assumed to be similar to the behavior upon assertion of an SRST*n* pin:

- Since this is a warm reset (i.e., power was not cut), the chip's precise behavior as it comes back up is the responsibility of the implementation.
- In order to receive a Dynamic Address, the Target will need to participate in the Dynamic Address Assignment process (e.g., with the ENTDAA CCC), since all I3C Basic Target logic would be reset by the Whole Target reset.
- Any further steps after Dynamic Address Assignment are the responsibility of the driver controlling the Controller and Target. Such steps might be dictated by rules imposed by higher-

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

level standardized protocols (e.g., Camera, Touch, Debug, etc.), and/or by other *a priori* driver knowledge of the specific Target type.

Examples

- **For a RAM-Based Target:** Determining whether another firmware download is needed
- **For a Sensor:** Determining whether a full calibration vs. a push of stored calibration data is needed

However, implementers are not required to trigger a full/chip reset when performing a Whole Target reset. Implementers may choose to support any of the available RSTACT Defining Byte values (i.e., 0x40 thru 0x7F) that can optionally be used for resetting other logic, domains, or functions within the chip; and may also choose to use one of these available Defining Byte values as a dedicated full/chip reset (i.e., instead of Defining Byte value 0x02).

4.3.9.4.1 Primary Controller Behavior After Whole Target Reset

There is a special condition where the Primary Controller acting as a Target (i.e., a Secondary Controller) is reset by the Active Controller.

When the Primary Controller wakes up from the reset and it does not know why it was reset, it then needs to determine whether it was the Active Controller before the reset, or was acting as a Target (i.e., a Secondary Controller) and some other Controller-capable Device was the Active Controller.

In order to do this, the Primary Controller shall monitor the Bus for activity on SCL for t_{IDLE}. If the Bus is active, then the Device assumes it is not the Active Controller. If there is no activity, then the Device shall drive the SDA Low (if it wasn't already Low) and wait for 50 ms (i.e., t_{CAS} maximum for Activity State 3), and if an Active Controller that initiated the Reset does not respond by pulling down the SCL line, then the Primary Controller can assume that it is the Active Controller and start driving the SCL line. Then the Primary Controller shall broadcast the RSTDAA CCC and continue with the Bus initialization.

Note:

*This procedure is similar to the Dead Bus Recovery procedure for Error Type DBR (see **Section 4.3.8.1.8**).*

4.3.9.5 Wake from Target Reset Behavior

This Specification does not mandate any particular implementation or behavior for Wake from Target Reset (i.e., the arc starting with "Target into Deepest Sleep" in ***Figure 66***). Waking from deepest-sleep/standby might also involve a reset (in whole or part); however, this will usually depend upon how the deepest-sleep/standby is implemented.

Common types of Wake from Target Reset behavior include:

- **Wake Performs Reset:** The Target wakes from "power down" state with little or no restoration of saved information. In particular, the Target's Dynamic Address is not saved. As a result, this case behaves like a normal reset:
  - Following the reset operation, in order to receive its assigned DA the Target will need to participate in the Controller-initiated ENTDAA Dynamic Address Assignment process.
  - If Hot-Join is supported, then the Target should also issue the Hot-Join Request after the reset.
- **Wake with State Restoration:** Some amount of critical state information was previously saved somewhere outside the Target (e.g., always-on flops or the RTC domain), and is restored upon Target wake. This case works the same as the offline wake model. This usually means saving and restoring the Target's Dynamic Address, and can also involve additional state data.
  - If the Target's DA is restored in this manner, then the Controller will not need to take further steps to re-assign it (i.e., no need to then perform the ENTDAA CCC Dynamic Address Assignment procedure).
  - In this case, the Target is not required to issue a Hot-Join Request.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

- **Wake with State Preserved:** In this case, Target state information does not need to be explicitly restored because it is preserved in place during the reset-wake interval. This case behaves the same as a wake from light sleep. Data preservation might be achieved via state-retention flops (SRFF), or simply by not actually cutting power during the reset-wake interval.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.10 HDR Pattern Tolerance

This Section describes the required HDR communications for all I3C Basic Devices.

Although support for HDR Modes is optional in I3C, all I3C Basic Targets must understand the following, even if such Targets do not support a particular HDR Mode (or any HDR Modes):

- When the Controller enters an HDR Mode (i.e., by sending any ENTHDRx CCC); and
- When the Controller sends the HDR Exit Pattern (i.e., to exit the HDR Mode and return to SDR Mode).

This ensures device compatibility on the Bus.

4.3.10.1 HDR Entry

Each HDR Mode is separate from all other HDR Modes:

- I3C Basic Targets may choose to support any desired combination of the HDR Modes, including no HDR Modes. If the I3C Bus enters an HDR Mode that an I3C Basic Target does not support, then that Target can simply wait and watch for the common HDR Exit Pattern.
- I3C Basic Controllers may choose to support any desired combination of the HDR Modes, including no HDR Modes. However, manufacturers are generally encouraged to support all of the defined HDR Modes.
- I3C Basic Devices that support HDR-BT may choose to support DUAL Lane or QUAD Lane configurations, in addition to SINGLE Lane configuration.

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.

HDR Modes may support CCCs as indicated by ***Table 75***. Generally, CCCs that assign Dynamic Addresses, alter I3C Bus Modes, or transfer the I3C Bus Controller Role are not allowed within HDR Modes.

An HDR Mode period on the I3C Bus involves five steps:

1. The Controller Broadcasts an Enter HDR Mode CCC, indicating which particular HDR Mode to enter. (See the Command Codes **Enter HDR Mode 0** through **Enter HDR Mode 7** [ENTHDR0–ENTHDR7] in ***Section 4.3.7.3.9***).
2. The I3C Bus switches from SDR Mode to the requested HDR Mode (see ***Section 6.1.4***).
3. The Controller issues the first structured protocol element per the HDR Mode framing. Typically, this is either a Command or Header, followed by optional Data sent by the Controller or Target Device, etc.
4. An HDR Restart Pattern (see ***Section 6.1***) or HDR Exit Pattern is sent.
   - If an HDR Restart Pattern, then the Controller issues the first structured protocol element for the new HDR Mode transfer (i.e., another Command or Header, followed by optional Data, etc.; see ***Section 6.1.4.2***).

   The Controller may repeat this process to remain in the HDR Mode, or send an HDR Exit Pattern to exit the HDR Mode.
5. If the Controller sends an HDR Exit Pattern, then it is always followed by I3C STOP, which ends with the Bus Free Condition.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.10.2 HDR Exit 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 Basic Targets shall detect and respond to the HDR Exit Pattern, irrespective of whether the Target supports any particular HDR Mode. The HDR Exit Pattern Detector is specified in ***Section 4.3.10.2.1***.

The HDR Exit Pattern is defined thus:

- SDA starts High, SCL starts Low
- SDA falls (from High to Low) 4 times, while SCL remains Low for the whole time
- Each transition on SDA and/or SCL is separated by a time interval of at least t_{DIG_H} (see ***Section 4.3.11.2***)

If necessary, then the HDR Exit Pattern shall be preceded by one additional edge on SCL and/or SDA to set each line up into the correct state (i.e., "Set Up SDA / Set Up SCL").

A normal I3C Basic STOP (i.e., SCL being High while SDA rises) always follows the HDR Exit Pattern, as shown in ***Figure 69***.

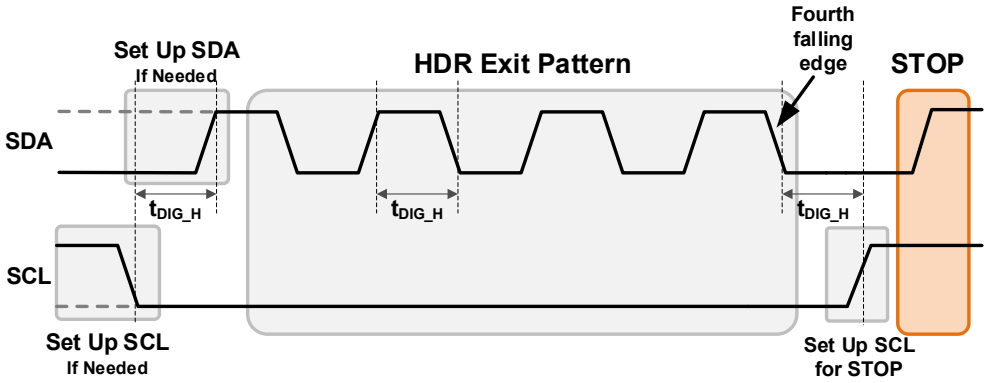


Figure 69 HDR Exit Pattern Followed by STOP

Note:

After the fourth SDA falling edge, the Controller shall drive SCL High in order to set up SCL to prepare for the I3C Basic STOP.

The Controller can also drive the HDR Exit Pattern at any time, specifically to recover the I3C Basic Bus from certain Target Error Types. ***Figure 70*** shows how the HDR Exit Pattern is initiated when the I3C Basic Bus is idle.

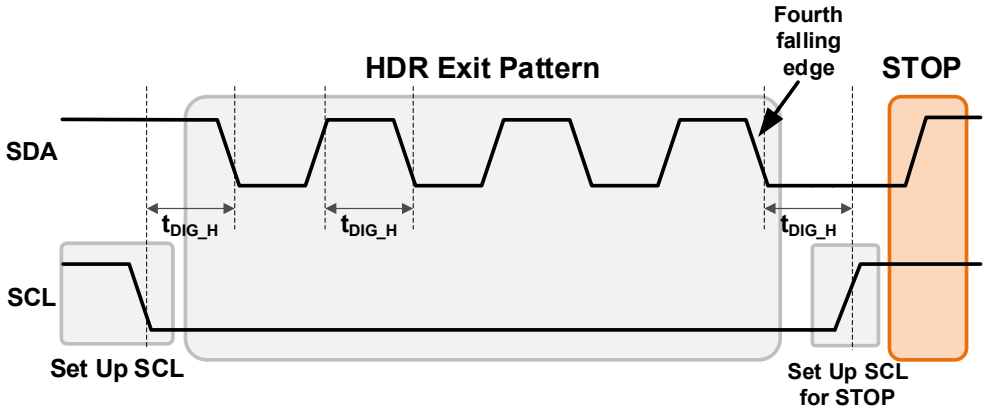


Figure 70 HDR Exit Pattern After IDLE Followed by STOP

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.10.2.1 HDR Exit Pattern Detector

All I3C Targets shall include an HDR Exit Pattern Detector.

The Target shall enable its HDR Exit Pattern Detector either:

- (A) When any HDR Mode is entered (i.e., when the ENTHDRx CCC is seen), or
- (B) When the Target enters any Target Error state that can be recovered by the HDR Exit Pattern (see ***Section 4.3.8.1***).

Once the Detector is enabled in these scenarios, the Target shall act on the HDR Exit Pattern once it is seen. The Detector may be implemented either in digital logic, or in software (Bit Banged).

Note:

The HDR Exit Pattern Detector does not strictly need to be disabled under other circumstances (i.e., no HDR Mode entered and not in a Target Error state) or once the HDR Exit Pattern has already been seen. However, the Target must ensure that the Detector does not initiate any special actions when an HDR Exit Pattern is not expected to be seen.

The remainder of this Section presents an example digital logic implementation in both schematic view and in RTL code.

The basic logic model is that SCL is held Low (0), and so SCL High (1) will reset the detector. The Detector also only uses falling edges of SDA, hence SDA is treated as a clock. An example Detector schematic is shown in ***Figure 71***.

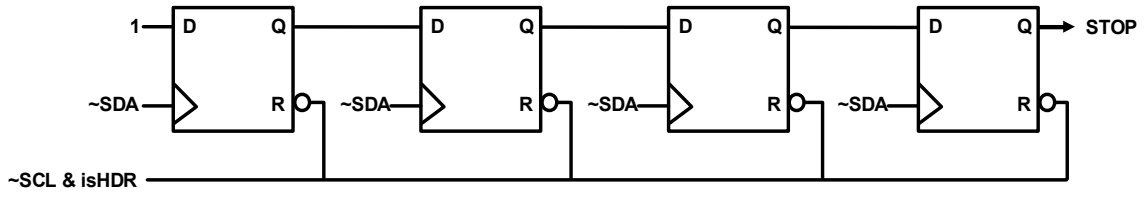


Figure 71 Example HDR Exit Pattern Detector (Schematic)

The Detector uses an inverted version of SDA as a clock (so positive-edge logic, but can use negative-edge logic), and is reset either when SCL is High (1), or when the block is not enabled (i.e., when the Target should not act on the HDR Exit Pattern, as defined above). The asynchronous nature of the reset makes this safe, even in the presence of HDR Ternary Modes which change SCL and SDA at the same time. This is shown in ***Figure 72***. Due to the nature of SCL vs. SDA changing at the same time with HDR Ternary Modes, the Bus Target could see SCL change after one SDA and before the next. If the Detector were only using clocking logic, then it would not see any change at all (SDA posedge would always see SCL Low in this example). Because the Detector uses an asynchronous reset on SCL, a change in SCL will impact the counter, even in case (b) above. Note that there will still be approximately 50 ns between SCL and SDA changes. So, as shown, if SCL rises at any time before the end of the HDR Exit Pattern (as defined above), then the Detector will be reset, and therefore will not mistakenly signal a false Exit.

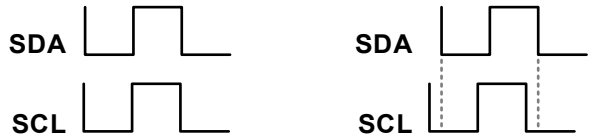


Figure 72 Metastable Changes on SCL and SDA Do Not Break the Exit Pattern Detector

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

An example RTL implementation of the ***Figure 71*** schematic is shown in ***Listing 1***.

Listing 1 Example RTL Code for HDR Exit Pattern Generator

```
wire          scl_rst_n;
wire          SDA_clk_n;      // ~SDA as clock
reg[3:0]     stp_cnt;        // HDR STOP counter (shift chain)

assign scl_rst_n = ~iSCL & iIsInHDR; // SCL=1 resets
// next uses glitch free XOR for SDA clock. Only inverts
// SDA as clock if not in scan. Could add a clock mux
// so uses one common clock in scan.
SAFE_CLK_XOR sda_neg_clk(iSDA, ~scan_mode, SDA_clk_n);

// counter for 3 and 4 rising SDA when SCL is High
// (else SCL resets). Whole thing held in reset if
// not in HDR Mode
always @ (posedge SDA_clk_n or negedge scl_rst_n)
  if (~scl_rst_n)
    stp_cnt <= 4'd0;          // SCL High or not HDR
  else
    stp_cnt <= {stp_cnt[2:0], 1'b1}; // shift chain counter
assign oHDR_STOP = stp_cnt[3]; // detects Exit (STOP)
```


Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.11 Electrical Specifications

4.3.11.1 DC I/O Characteristics

This Section describes the DC operating parameters of the I3C Basic 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 Basic:

- For typical use cases, the I3C Basic interface targets nominal operating voltages of 1.2 V, 1.8 V, and 3.3 V or less. The I3C Basic interface is not characterized for 5 V systems, but could be extended to support 5 V 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^2C. On I3C Basic, higher capacitance Busses are possible, but only at reduced speeds and with reduced features (e.g., if no I^2C Devices are supported).

The I3C Basic Specification also defines **Optional Feature F010: Alternative Electrical Specifications** (see ***Section 6.11.2.1***). This optional feature allows the I3C interface to be used with a nominal operating voltage of 1.0 V and a 100 pF capacitive loading for new usages such as Serial Presence Detect (SPD) in DDR5.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.11.1.1 Typical I3C Basic Use Cases

3754

Table 46 I3C Basic 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}	–	−0.1 * V_{DD}	–	0.3 * V_{DD}	V	8
High-Level Input Voltage	V_{IH}	–	0.7 * V_{DD}	–	1.1 * V_{DD}	V	8
Schmitt Trigger Inputs Hysteresis	V_{hys}	–	0.1 * V_{DD}	–	–	V	–
Output Low Level	V_{OL}	For V_{DD} < 1.4 V: I_{OL} = 2 mA	–	–	0.18	V	2
		For V_{DD} ≥ 1.4 V: 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 ≥ 1.8 V nominal	−10	–	10	µA	2
		−100 mV < V_i < V_{DD} + 100 mV for < 1.8 V 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 ≤ 5 pF	–	–	1.5	pF	5
		Difference between SDA and SCL capacitance C_i > 5 pF	–	–	3	pF	5
Push-Pull Only							
Output High Level	V_{OH}	For V_{DD} < 1.4 V: I_{OH} = −2 mA	V_{DD} – 0.18	–	–	V	2, 9
		For V_{DD} ≥ 1.4 V: I_{OH} = −3 mA	V_{DD} – 0.27	–	–	V	2, 9
Legacy Mode with Pull-Up							
Pull-Up for Open Drain	R_p	t_r = Max rise time C_b = Bus Capacitance V_{DD} = 1.2 V, 1.8 V, or 3.3 V	$\frac{V_{DD} - V_{OL}}{3\ mA}$ *for* $V_{DD} \geq 1.4\ V$		$\frac{t_r}{0.8473 * Cb}$	Ω	3, 4, 6, 7
		t_r = 120 ns C_b = 50 pF	$\frac{V_{DD} - V_{OL}}{2\ mA}$ *for* $V_{DD} < 1.4\ V$		2833		

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Note:

1) For typical I3C Basic use cases, this Specification considered only 1.2 V, 1.8 V, and 3.3 V, with associated tolerances. However, other voltage ranges are not prohibited; for example, see ***Section 6.11****. Any I3C Basic 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 that the Device is sinking current in this condition; see note 9.

3) The current of the Pull-Up must balance the time to pull SDA High against the capacitance of the line, with the current that a Target Device must sink in order to hold SDA at V_{OL}. Unless all Target Devices are known to have sufficiently strong drive (i.e., to pull SDA below V_{OL} in time), the Pull-Up should be sized between the minimum and maximum values.

$V(t1) = 0.3 * V_{DD} = V_{DD} (1 - e^{-t1/RC})$; 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 Basic 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 1.8 V and 3.3 V, the larger capacitance may be used.

6) Open Drain never occurs on SCL

7) A weak Pull-Up or High-Keeper is separately sized, whether passive or active; this needs to be applied for both Controller handoff (see ***Section 6.5.3.2****) and "Park1,High-Z" uses in HDR-BT Mode (see* ***Section 6.4.3.2****).*

8)The V_{IL} min and V_{IH} max allow for normal undershoot and overshoot on a Bus. Voltages that are lower than V_{IL} min and higher than V_{IH} max will be handled by protection circuits (i.e., ESD protection) as with any GPIO or standard pad Peripheral. The board designer should be made aware of any deviations outside V_{IL} min and V_{IH} max.

9) A Device shall be able to continuously sink the indicated current (indicated as a negative number) in this condition. For most pad types, the drive strength must be rated at twice the indicated current, since the rated drive strength is normally the peak current when the voltage difference is at its maximum. For V_{OL}, this is SDA at V_{DD}; for V_{OH}, this is SDA at ground.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4.3.11.1.2 Interoperability with Legacy I²C Targets

I3C Basic supports Legacy I²C Targets. An important feature of an I²C Target 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 Basic Bus, then the I3C Basic Bus may operate at maximum rated clock frequency. If any I²C Device does not have a Spike Filter, then the I3C Basic Bus speed is determined by the slowest Legacy I²C Device without a Spike Filter.

Note:

An I3C Basic Target that also supports Legacy I²C Target behaviors (i.e., that could also function on an I²C Bus) could also have a Spike Filter that is enabled by default, until and unless the I3C Basic Target learns that it is actually on an I3C Basic Bus. Refer to ***Section 4.3.2.1.1*** *for more details.*

Other requirements of an I²C Legacy Target are listed in ***Table 47***.

Table 47 Legacy I²C Device Requirements When Operating on I3C Basic

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 Target	–	–	–	X	–
I²C Extended Address (10 bit)	–	–	X	–	2
I3C Basic Reserved Address	–	–	–	X	–

Note:

1) *Lack of Spike Filter will severely degrade Bus performance and eliminate certain I3C Basic Bus features*
2) *"Not Used" means that the I3C Basic Controller will not make use of the I²C feature, however if the Target supports the feature, then it will not interfere with I3C Basic Bus operation.*

4.3.11.2 Timing Specification

A key feature of I3C Basic is to maximize the speed of data transfer and minimize the time required for low-power Devices to remain in non-sleep states. In SDR (Single Data Rate) Mode this is accomplished by keeping Bus capacitance low and clock speed high. For large packets of data an additional speed improvement is possible using HDR Modes, where data is transferred on every clock edge. I3C Basic supports Legacy I²C Fm and Fm+ Modes. ***Table 48*** gives reference timing requirements used in Legacy Mode.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3771

Table 48 I3C Basic 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 74*	600	–	260	–	ns	–
Hold Time for a (Repeated) START	t_{HD_STA}	*Figure 74*	600	–	260	–	ns	–
SCL Clock Low Period	t_{LOW}	*Figure 74* *Figure 75*	1300	–	500	–	ns	–
	t_{DIG_L}	*Figure 75*	$t_{LOW} + t_{rCL}$	–	$t_{LOW} + t_{rCL}$	–	ns	10
SCL Clock High Period	t_{HIGH}	*Figure 74* *Figure 75*	600	–	260	–	ns	–
	t_{DIG_H}	*Figure 75*	$t_{HIGH} - t_{rCL}$	–	$t_{HIGH} + t_{rCL}$	–	ns	10
Data Setup Time	t_{SU_DAT}	*Figure 74*	100	–	50	–	ns	–
Data Hold Time	t_{HD_DAT}	*Figure 74*	0	–	0	–	ns	–
SCL Signal Rise Time	t_{rCL}	*Figure 74*	20	300	–	120	ns	–
SCL Signal Fall Time	t_{fCL}	*Figure 74*	20 * (V_{DD} / 5.5 V)	300	20 * (V_{DD} / 5.5 V)	120	ns	–
SDA Signal Rise Time	t_{rDA} t_{rDA_OD}	*Figure 74* *Figure 78*	20	300	–	120	ns	–
SDA Signal Fall Time	t_{fDA}	*Figure 74*	20 * (V_{DD} / 5.5 V)	300	20 * (V_{DD} / 5.5 V)	120	ns	–
Setup Time for STOP	t_{SU_STO}	*Figure 74*	600	–	260	–	ns	–
Bus Free Time Between a STOP and a START	t_{BUF}	–	1.3	–	0.5	–	µs	–
Pulse Width of Spikes that the Spike Filter Must Suppress	t_{SPIKE}	*Figure 90*	0	50	0	50	ns	–

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3772 During an I3C Basic communication the drive on the SDA pin shall have the ability to dynamically switch
3773 between Push-Pull and Open Drain.

3774 **Table 49 I3C Basic Open Drain Timing Parameters**

Parameter	Symbol	Timing Diagram	I3C Basic Open Drain Mode		Units	Notes
			Min	Max		
Low Period of SCL Clock	t_{LOW_OD}	***Figure 78***	200	–	ns	1, 2
	$t_{DIG_OD_L}$	***Figure 78***	t_{LOW_ODmin} + t_{fDA_ODmin}	–	ns	–
High Period of SCL Clock (for First Broadcast Address)	t_{HIGH_INIT}	–	200	–	ns	9
High Period of SCL Clock (for Mixed Bus)	t_{HIGH}	***Figure 75***	–	41	ns	3, 4
	t_{DIG_H}	***Figure 75*** ***Figure 90***	–	t_{HIGH} + t_{CF}	ns	–
High Period of SCL Clock (for Pure Bus)	t_{HIGH}	***Figure 75***	24 ($t_{HIGHmin}$ from Push-Pull)	–	ns	–
	t_{DIG_H}	***Figure 75*** ***Figure 90***	32 (t_{DIG_Hmin} from Push-Pull)	–	ns	–
Fall Time of SDA Signal	t_{fDA_OD}	***Figure 78***	–	12	ns	–
SDA Data Setup Time During Open Drain Mode	t_{SU_OD}	***Figure 76*** ***Figure 78***	3	–	ns	–
Clock After START (S) Condition	t_{CAS}	***Figure 78***	38.4 nano	For ENTAS0: 1 µ	seconds	5, 6
				For ENTAS1: 100 µ		
				For ENTAS2: 2 milli		
				For ENTAS3: 50 milli		
Clock Before STOP (P) Condition	t_{CBP}	***Figure 79***	t_{CASmin} / 2	–	seconds	–
Active Controller to Secondary Controller Overlap time during handoff	$t_{CRHPOverlap}$	***Figure 89***	$t_{DIG_OD_Lmin}$	–	ns	8
Bus Available Condition	t_{AVAL}	–	1	–	µs	7
Bus Idle Condition	t_{IDLE}	–	200	–	µs	–
Time Internal Where New Controller Not Driving SDA Low	$t_{NEWCRLock}$	***Figure 89***	$t_{AVALmin}$	–	µs	–

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Note:

1) t_{LOW_OD} Min was specified to be enough time for a pull-up resistor to pull the SDA line High in a typical system.

2) The Controller 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 I^2C Devices, and/or in consideration of the interconnect (e.g., a short Bus)

*5) On a Legacy Bus where I^2C Devices need to see Start, the t_{CAS} Min value is further constrained (see **Section 4.3.3.3**)*

*6) Targets that do not support the optional ENTASx CCCs (see **Section 4.3.7.3.2**) shall use the t_{CAS} Max value shown for ENTAS3*

7) On a Mixed Bus with Fm Legacy I^2C Devices, t_{AVAL} is 300 ns shorter than the Fm Bus Free Condition time (t_{BUF})

8) This timing can be disregarded when external passive High-Keepers are applied on both SCL and SDA lines.

*9) The Controller uses this timing to send the first Broadcast Address, in order to disable the I^2C Spike Filter for applicable I3C Basic Target Devices (see **Section 4.3.2.2.2**).*

10) The reference points for t_{fCL}/t_{rCL} and t_{CF} are the same.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3775

Table 50 I3C Basic Push-Pull Timing Parameters for SDR, ML, HDR-DDR, and HDR-BT Modes

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 74***	24	–	–	ns	–
		t_{DIG_L}	***Figure 75***	32	–	–	ns	2, 4
SCL Clock High Period (for Mixed Bus)		t_{HIGH_MIXED}	***Figure 75***	24	–	–	ns	–
		$t_{DIG_H_MIXED}$	***Figure 75***	32	–	45	ns	2, 3
SCL Clock High Period (for Pure Bus)		t_{HIGH}	***Figure 74***	24	–	–	ns	–
		t_{DIG_H}	***Figure 75*** ***Figure 74***	32	–	–	ns	2
Clock In to Data Out for Target		t_{SCO}	***Figure 81***	–	–	20	ns	7, 8
SCL Clock Rise Time		t_{CR}	***Figure 75***	–	–	100e06 * 1 / f_{SCL} (capped at 60)	ns	5
SCL Clock Fall Time		t_{CF}	***Figure 75***	–	–	100e06 * 1 / f_{SCL} (capped at 60)	ns	5
SDA Signal Data Hold in Push-Pull Mode	**Controller**	t_{HD_PP}	***Figure 80***	t_{CR} + 3 and t_{CF} + 3	–	–	–	4, 6
	Target	t_{HD_PP}	***Figure 82*** ***Figure 83*** ***Figure 84*** ***Figure 85***	0	–	–	–	6
SDA Signal Data Setup in Push-Pull Mode		t_{SU_PP}	***Figure 80*** ***Figure 81***	3	–	N/A	ns	–
Clock After Repeated START (Sr) Condition		t_{CASr}	***Figure 87***	t_{CASmin} / 2	–	N/A	ns	9
Clock Before Repeated START (Sr) Condition		t_{CBSr}	***Figure 87***	t_{CASmin} / 2	–	N/A	ns	–
Capacitive Load per Bus Line (SDA/SCL)		C_b	–	–	–	50	pF	–
HDR-BT SCL Clock Frequency		t_{BT_FREQ}	–	0.1	12.5	12.9	MHz	–
HDR-BT Controller to Target Hand Off Delay		t_{BT_HO}	–	–	–	10	µS	–
HDR-BT Delay Bytes during Read (for Data Block Delay mechanism)		t_{BT_DBD}	–	–	–	1024	Bytes	10

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Note:

1) $F_{SCL} = 1 / (t_{DIG_L} + t_{DIG_H})$. Frequency is based on the t_{SCO}, propagation time, PAD delays, and bus capacitance, all of which should be considered for an effective frequency calculation. While the maximum sustained frequency is 12.5 MHz, a higher frequency up to 12.9 MHz could be seen due to random jitter or skew in the Controller. However, such a frequency variation should only occur in short bursts since a frequency higher than 12.5 MHz is not expected to be sustained for longer periods of time and is likely to cause communication errors on the I3C Basic Bus.

*2) t_{DIG_L} and t_{DIG_H} are the clock Low and High periods as seen at the receiver end of the I3C Basic Bus using V_{IL} and V_{IH} (see **Figure 74**). The t_{DIG_L}, t_{DIG_H}, and t_{DIG_MIXED} minimum values of 32 ns are provided to allow an extreme 40/60 duty cycle at typical clock frequency of 12.5 MHz.*

3) When communicating with an I3C Basic Device on a Mixed Bus, the $t_{DIG_H_MIXED}$ period must be constrained to make sure that I^2C Devices do not interpret I3C Basic signaling as valid I^2C signaling. The t_{HIGH}, t_{LOW}, and t_{HIGH_MIXED} minimum values of 24 ns are provided as a safe corner case, in order to allow enough time for processing under extreme conditions.

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 Controller mode vs. Target mode. Push-Pull Mode includes both SDR Mode and DDR Mode. In SDR Mode the Hold time parameter is referred to as t_{HD_SDR}, and in DDR Mode it is referred to as t_{HD_DDR}

7) Devices with more than 20 ns of t_{SCO} delay shall set the limitation bit in the BCR, and shall support the GETMXDS CCC to allow the Controller 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 Controller may be a Target in a Multi-Controller system, and thus shall also adhere to this requirement

9) Targets with speed limitations inform the Controller via the Bus Characteristics Register that the minimum might not be acceptable. As a result, if the given SCL High period is 50 ns or greater, then the Controller needs to accommodate for Legacy I^2C Devices that might see it.

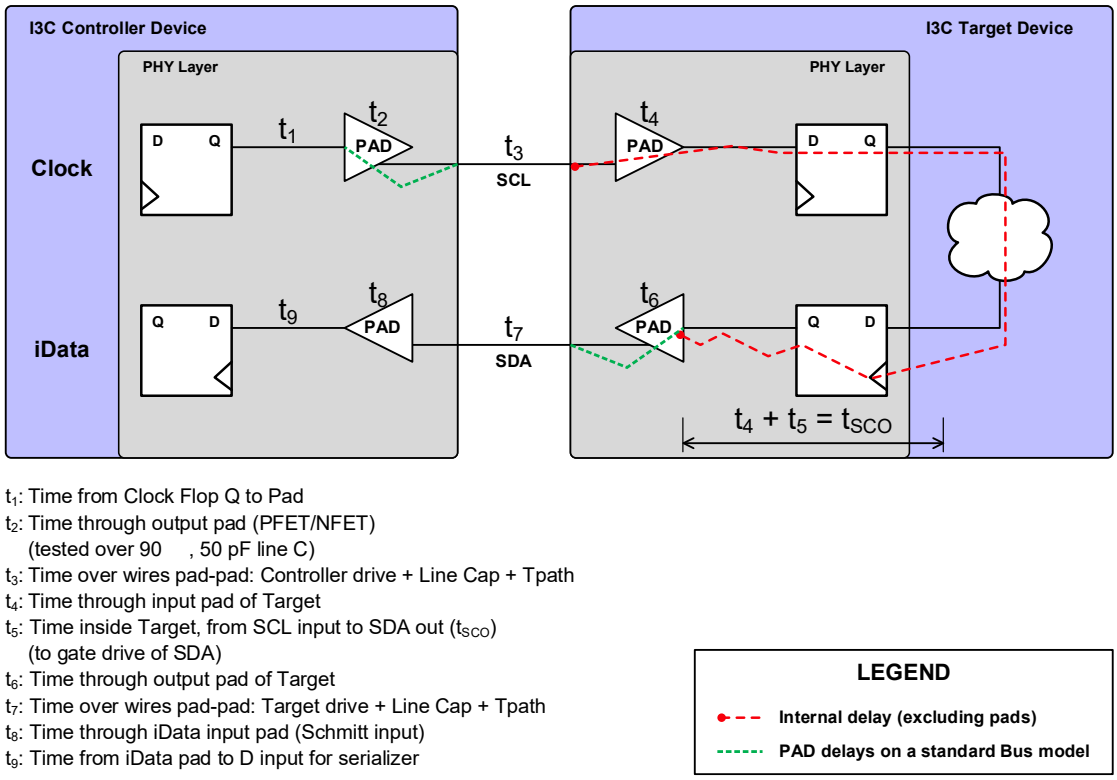
*10) This parameter represents the maximum number of Delay bytes that a Target may use during HDR-BT Read transfers, at each opportunity where the Target is unable to transmit or chooses to delay transmitting a complete Data Block. This is fundamentally different from Stalling the clock. In previous versions of this I3C Basic Specification, the HDR-BT Data Block Delay mechanism (i.e., the "Stall (delay)" mechanism) was defined with a symbol of t_{BT_STALL} for the maximum number of Delay bytes. This mechanism and its symbol have been renamed for clarity. See **Section 6.4.3.3.4**.*

Note:

It is not expected (and it would be unrealistic to expect) that all corner conditions will be present concurrently (i.e., that maximum frequency, worst duty cycle, slower edge, etc., would all occur at the same time).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

As shown in ***Figure 73***, the t_{SCO} delay is the measurement of the total internal delay from the SCL input to the start of SDA output (see also ***[MIPI03]***), excluding other factors like line capacitance and path delay that are factored out of the Device computation and put into the broader computation. The t_{SCO} delay is applicable to both Controller Devices and Target Devices, since Controller Devices may behave as Target Devices in a Multi-Controller system based on the system configuration.



t_1: Time from Clock Flop Q to Pad
t_2: Time through output pad (PFET/NFET)
(tested over 90 , 50 pF line C)
t_3: Time over wires pad-pad: Controller drive + Line Cap + Tpath
t_4: Time through input pad of Target
t_5: Time inside Target, from SCL input to SDA out (t_{SCO})
(to gate drive of SDA)
t_6: Time through output pad of Target
t_7: Time over wires pad-pad: Target drive + Line Cap + Tpath
t_8: Time through iData input pad (Schmitt input)
t_9: Time from iData pad to D input for serializer

Figure 73 Components of Clock-to-Data Turnaround Delay (t_{SCO})

Any Target Devices with a Clock-to-Data turnaround (t_{SCO}) delay greater than 20 ns shall set the Clock to Data Turnaround field of the maxRD Byte to 3'b111 and report this value to the Controller by private agreement.

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

The timing diagram in ***Figure 74*** depicts I3C Basic Legacy Mode for I^2C Devices. The timing parameters referenced in this timing diagram appear in ***Table 48***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

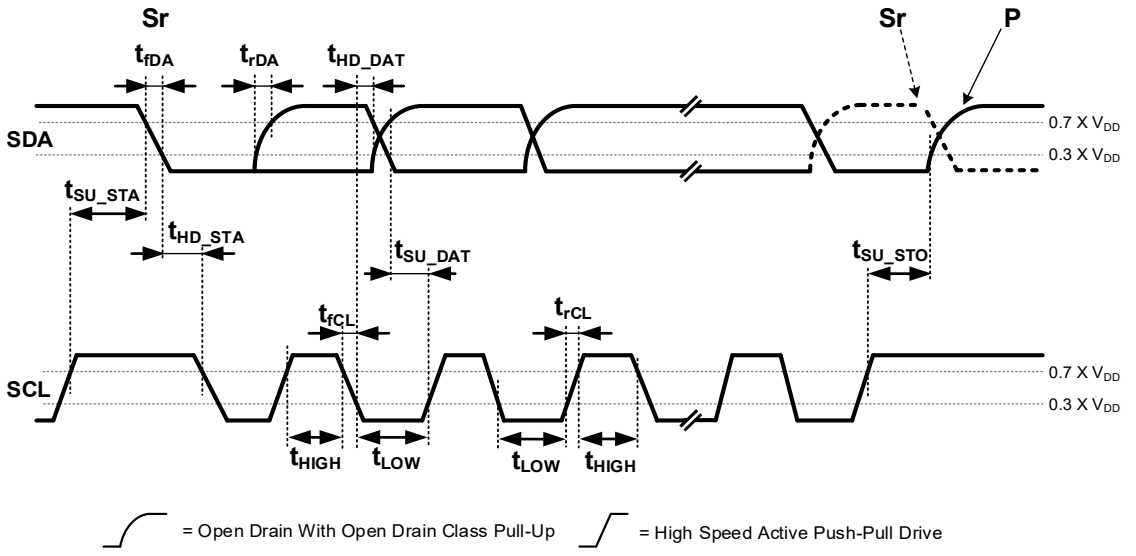


Figure 74 I3C Basic Legacy Mode Timing

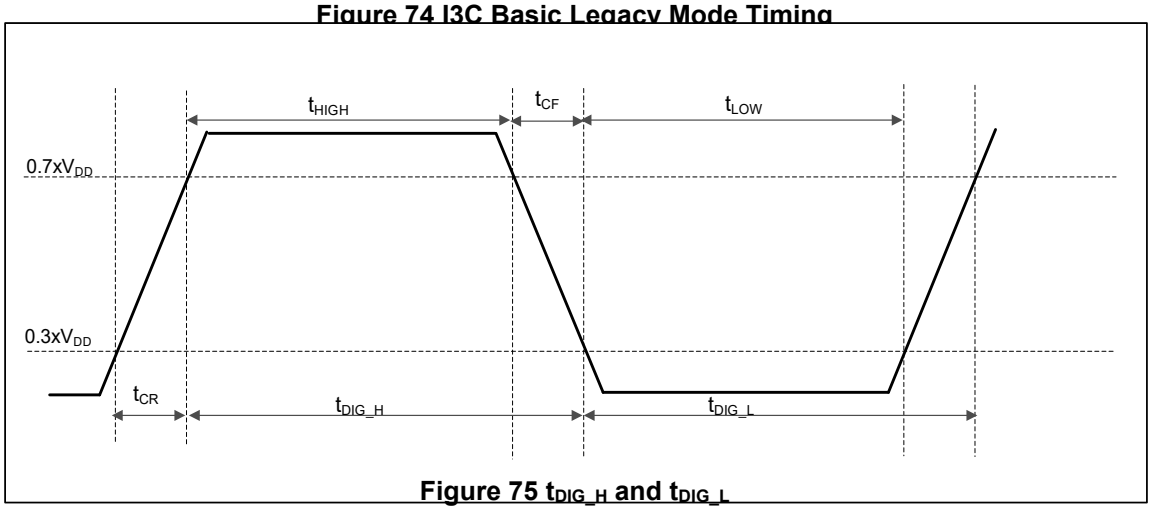
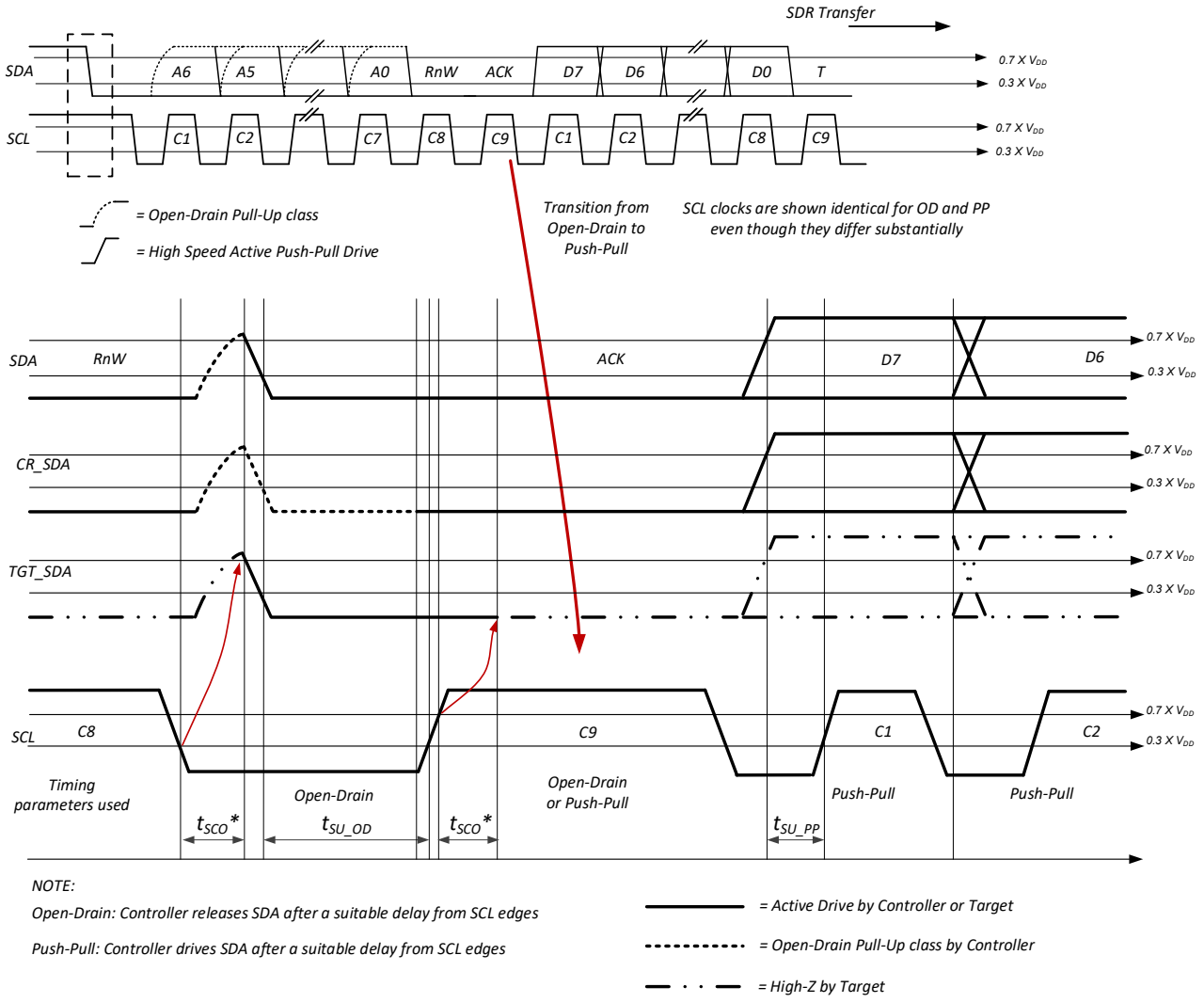


Figure 75 t_{DIG_H} and t_{DIG_L}

The start of a typical I3C Basic communication in SDR Mode is shown in ***Figure 76***. The initial communication looks very similar to I^2C, with additional commands that follow as described in ***Section 4***. The key difference is that higher clock speed is supported, up to 12.5 MHz. The higher clock speed allows Legacy I^2C Devices with 50 ns Spike Filters to ignore the communications. The Controller can then run in SDR Mode for I3C Basic Devices using full 12.5 MHz timing, though it will have to slow down in order to communicate with Legacy I^2C Devices. ***Figure 76*** shows the beginning of a transaction, where the Target acknowledges its Address.

Figure 77 shows the possible continuation of an I3C Basic SDR communication, in the case where the Target does not acknowledge its Address. The Controller may either STOP the communication, or else continue with a Repeated START.

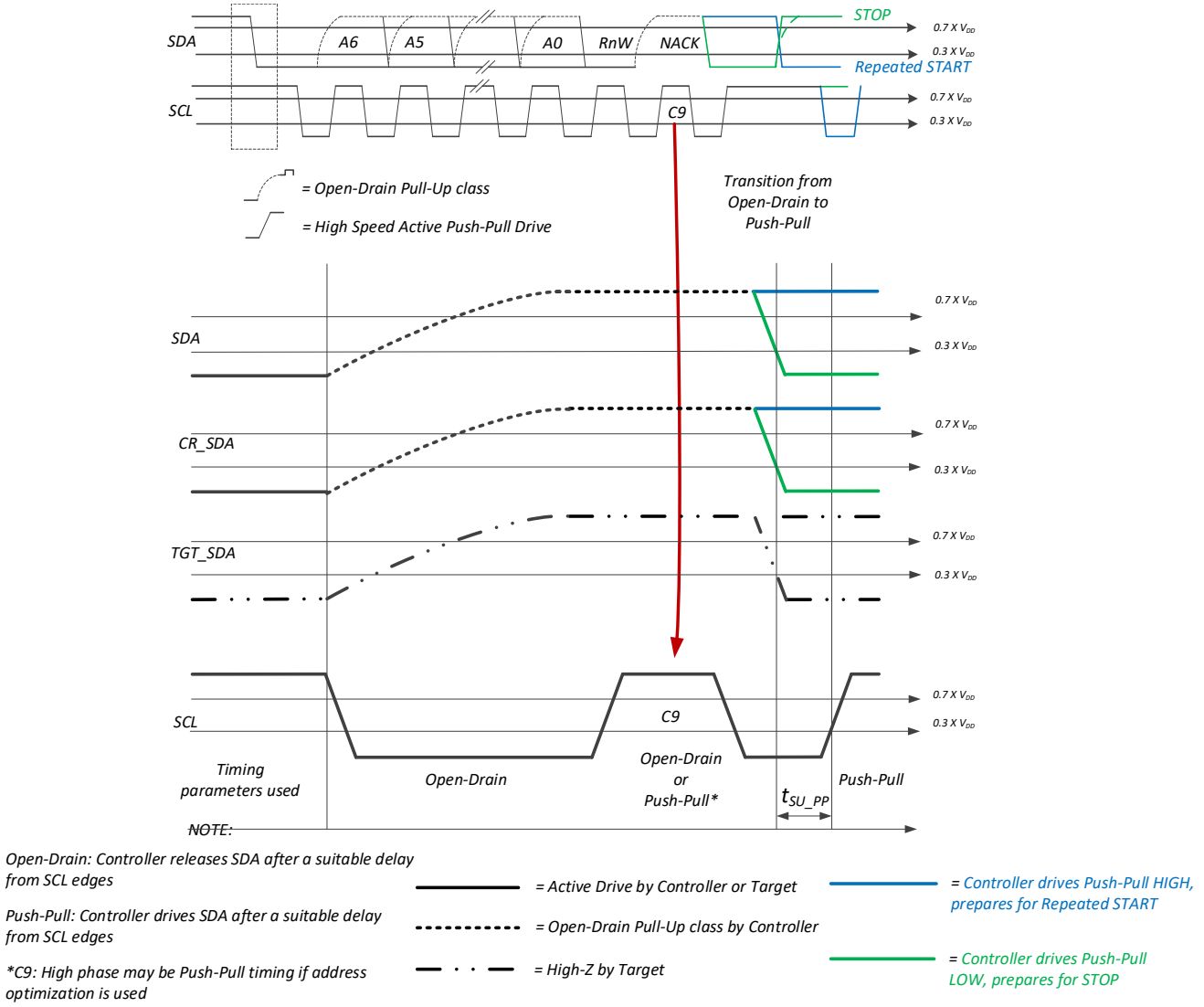
Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



3809

Figure 76 I3C Basic Write Address ACK

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

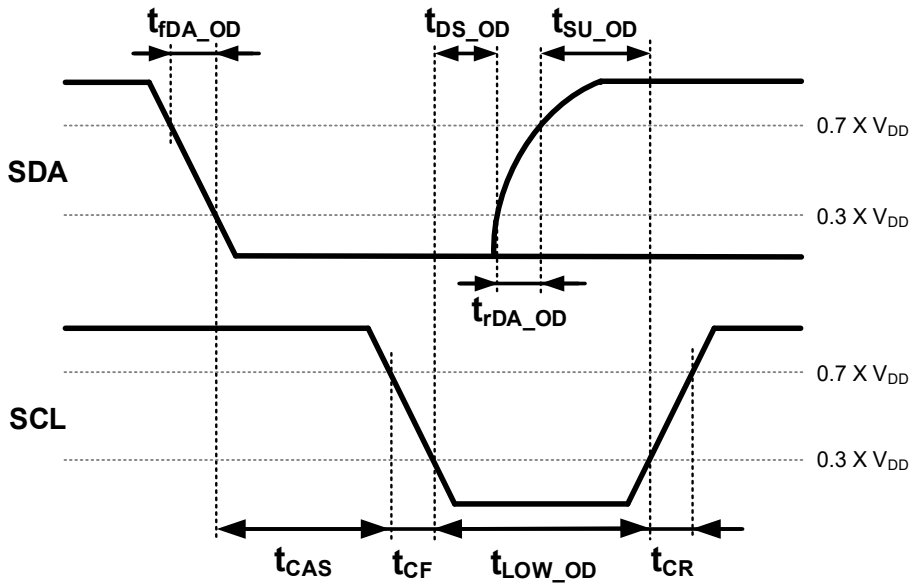


3810

Figure 77 I3C Basic Write Address NACK

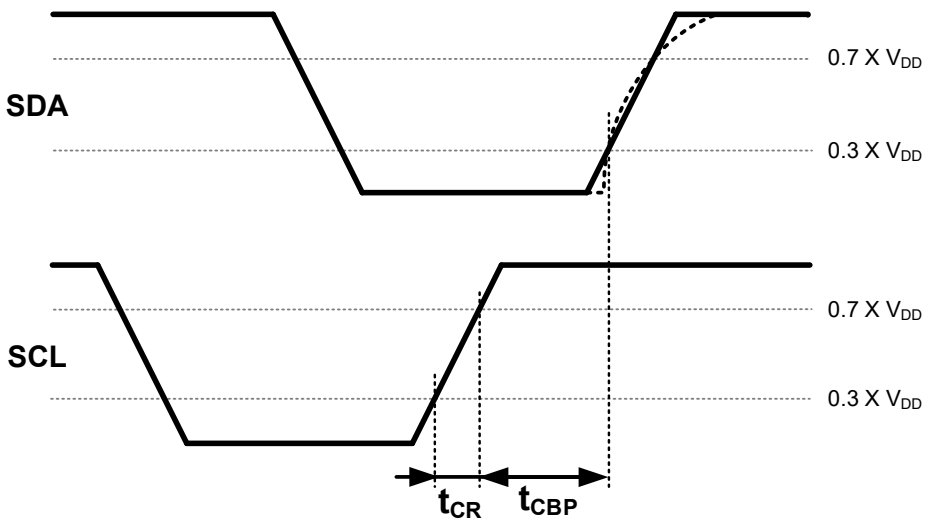
Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3811 ***Figure 78*** shows the timing parameters for an I3C Basic START (not Repeated START), and ***Figure 79***
3812 shows the timing parameters for an I3C Basic STOP.



3813

Figure 78 I3C Basic START Timing



3814

Figure 79 I3C Basic STOP Timing

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Figure 80 and ***Figure 81*** illustrate the timing parameters that are unique to the Controller Device and to the Target Device, as specified in ***Table 50***. The primary difference between the two is that a Controller always transmits the clock (SCL), whereas the Target is receiver-only on the SCL pin.

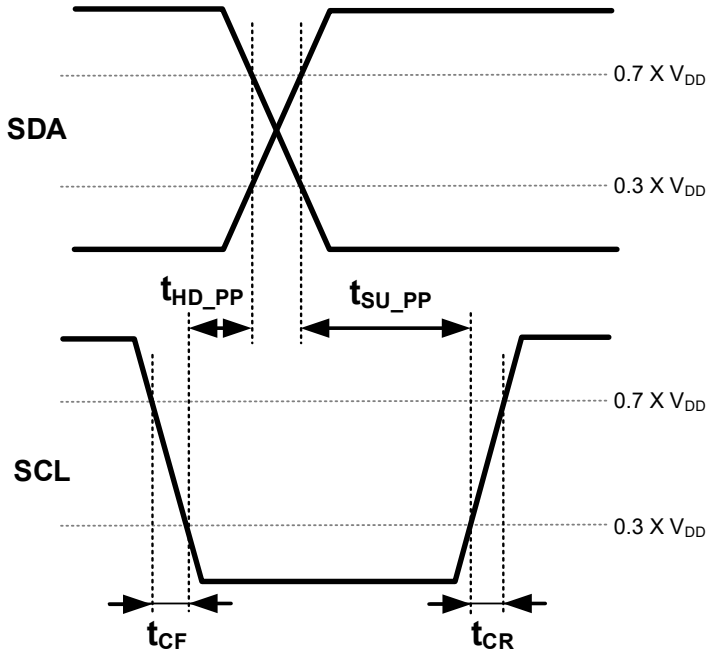
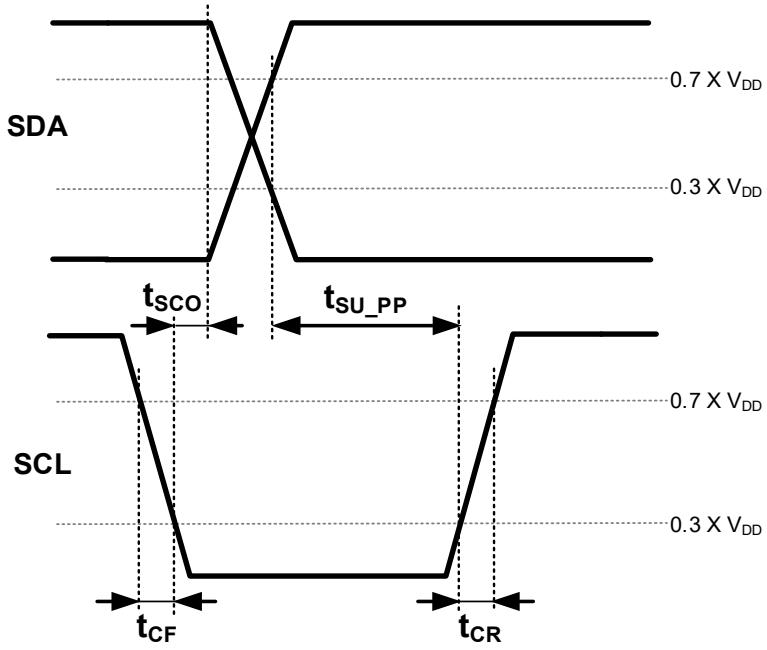


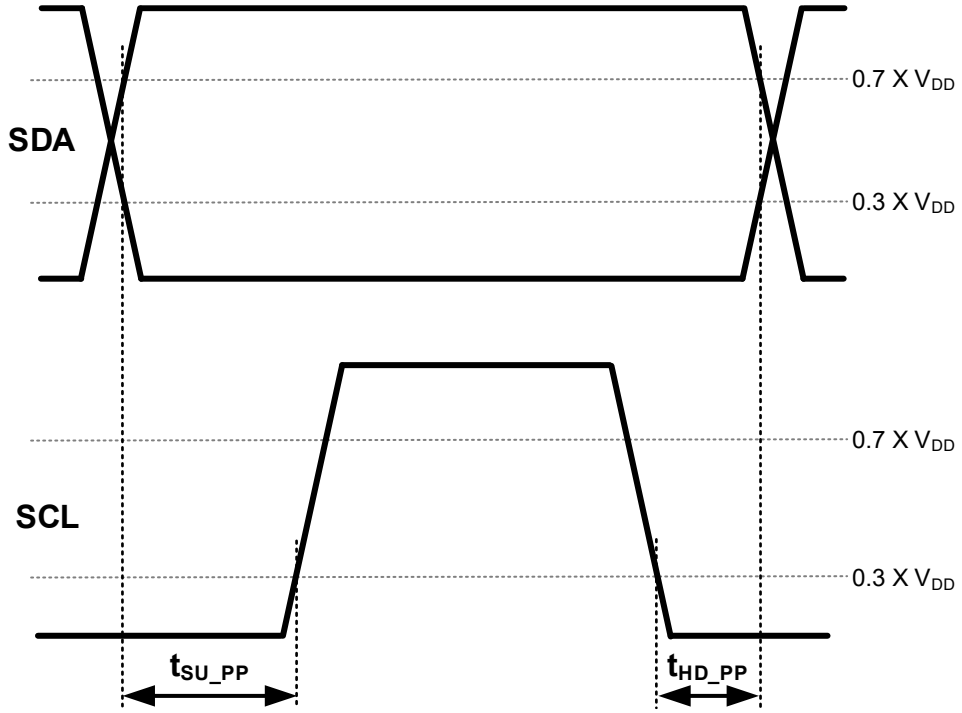
Figure 80 I3C Basic Controller Out Timing



** t_{SCO} is depicted for informative purposes only*

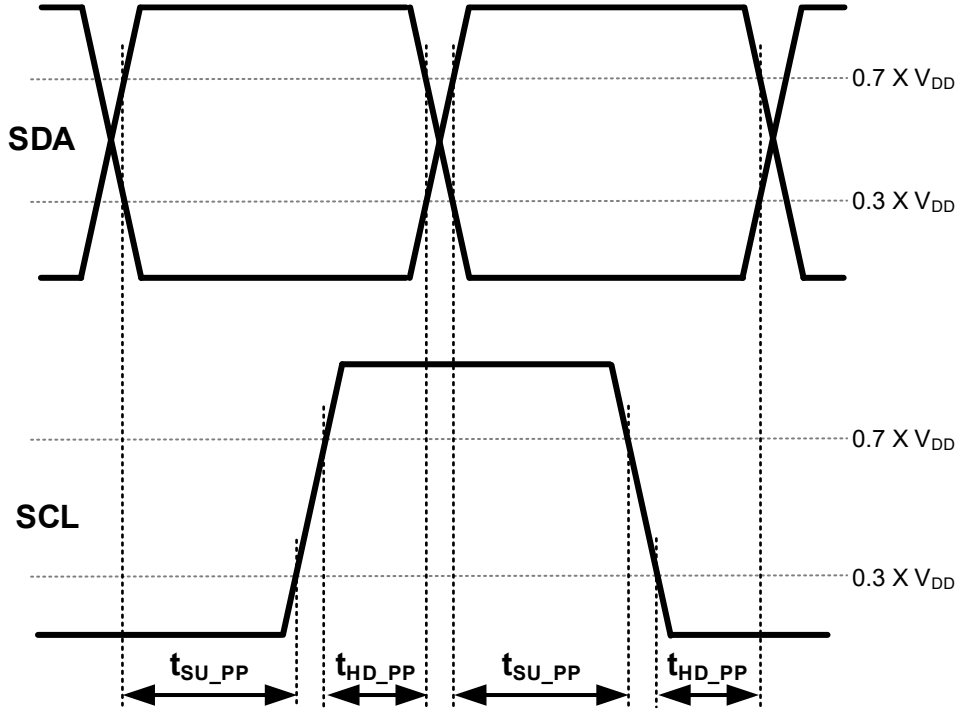
Figure 81 I3C Basic SDR Target Out Timing

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



3820

Figure 82 Controller SDR Timing



3821

Figure 83 Controller DDR Timing

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

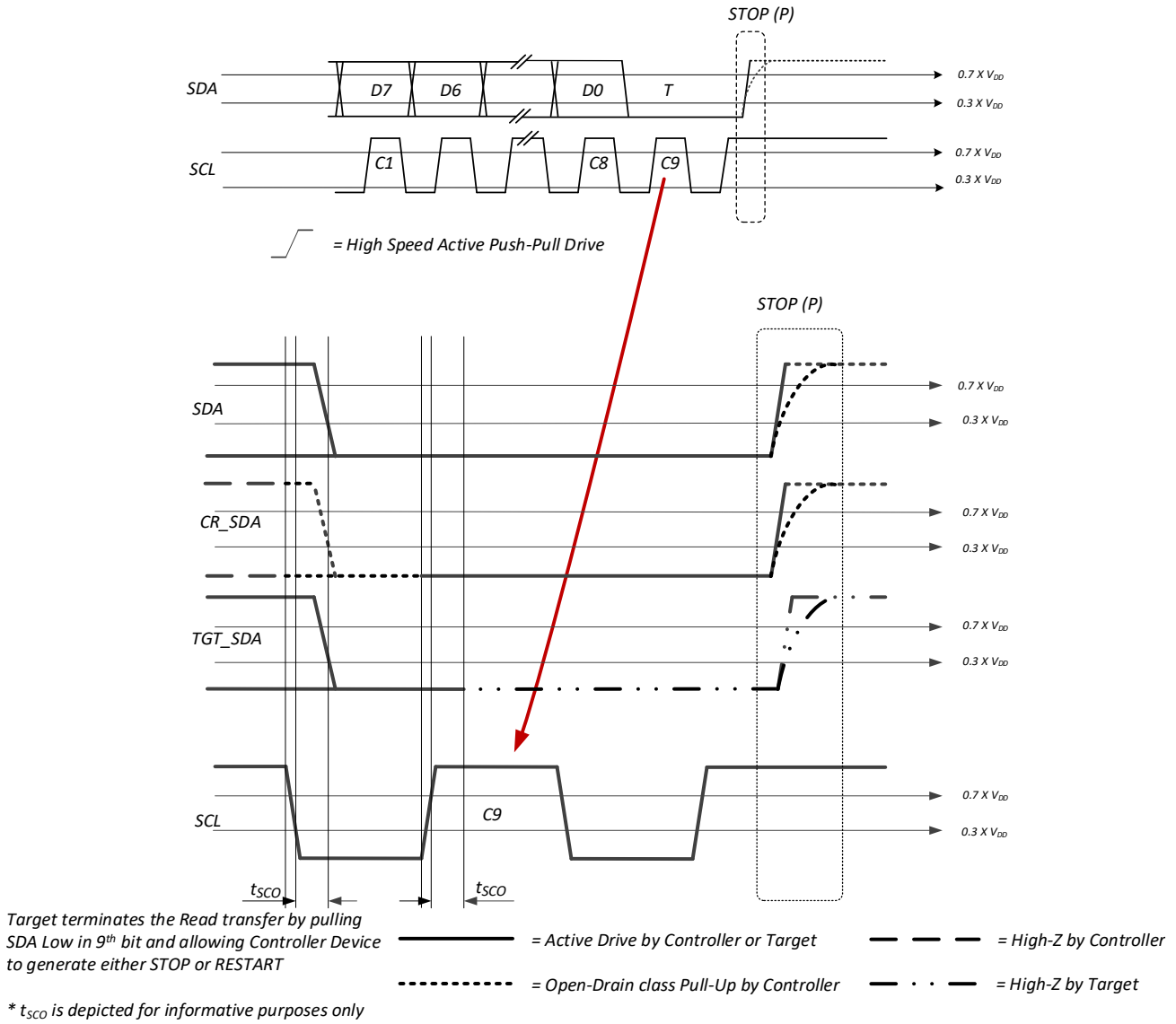
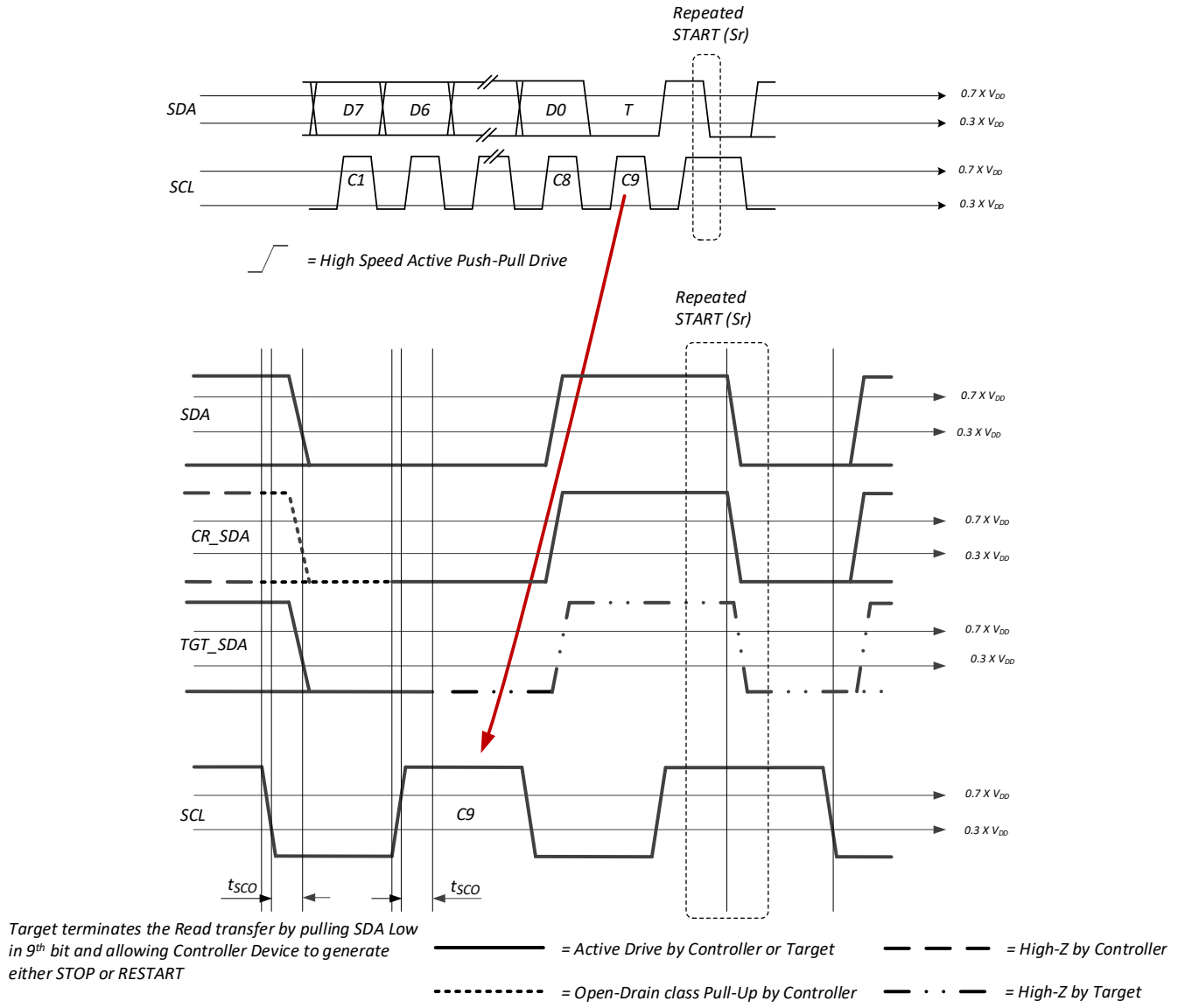


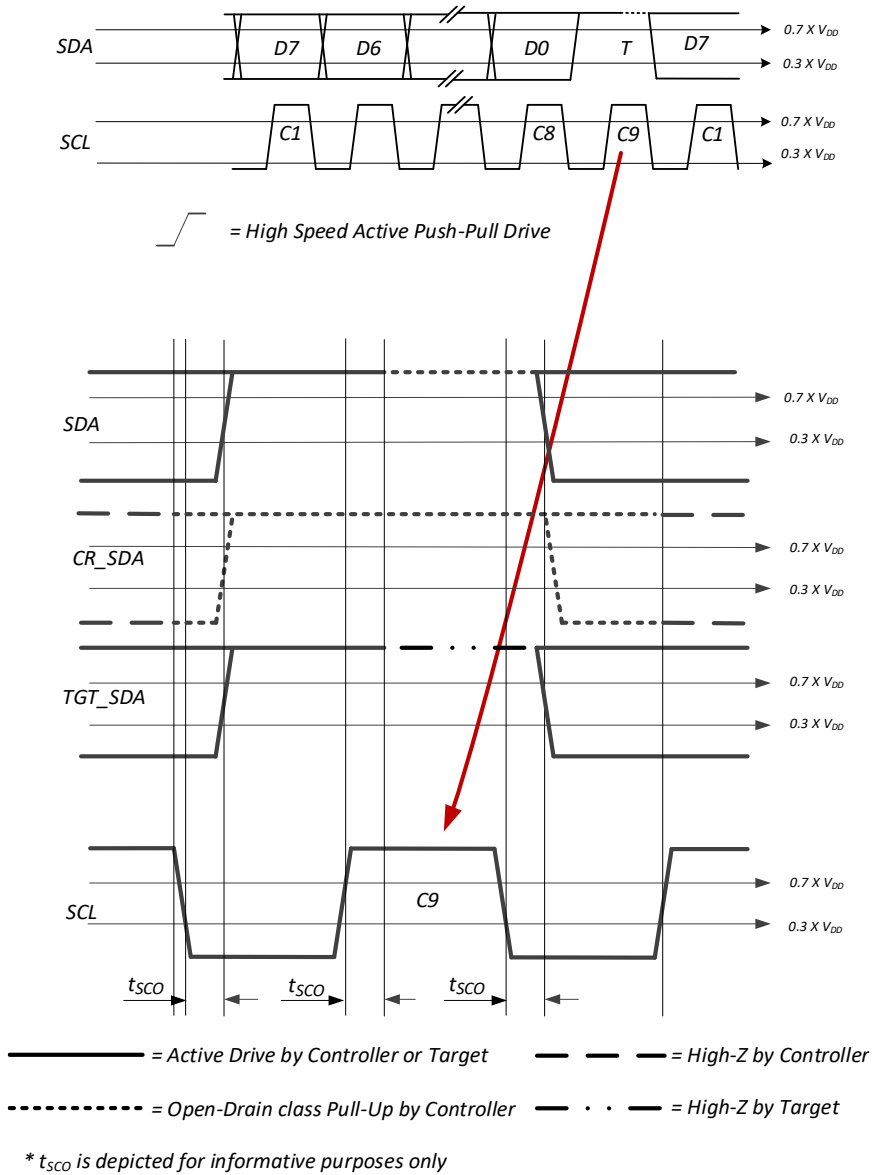
Figure 84 T-Bit When Target Ends Read and Controller Generates STOP



3823

Figure 85 T-Bit When Target Ends Read and Controller Generates Repeated START

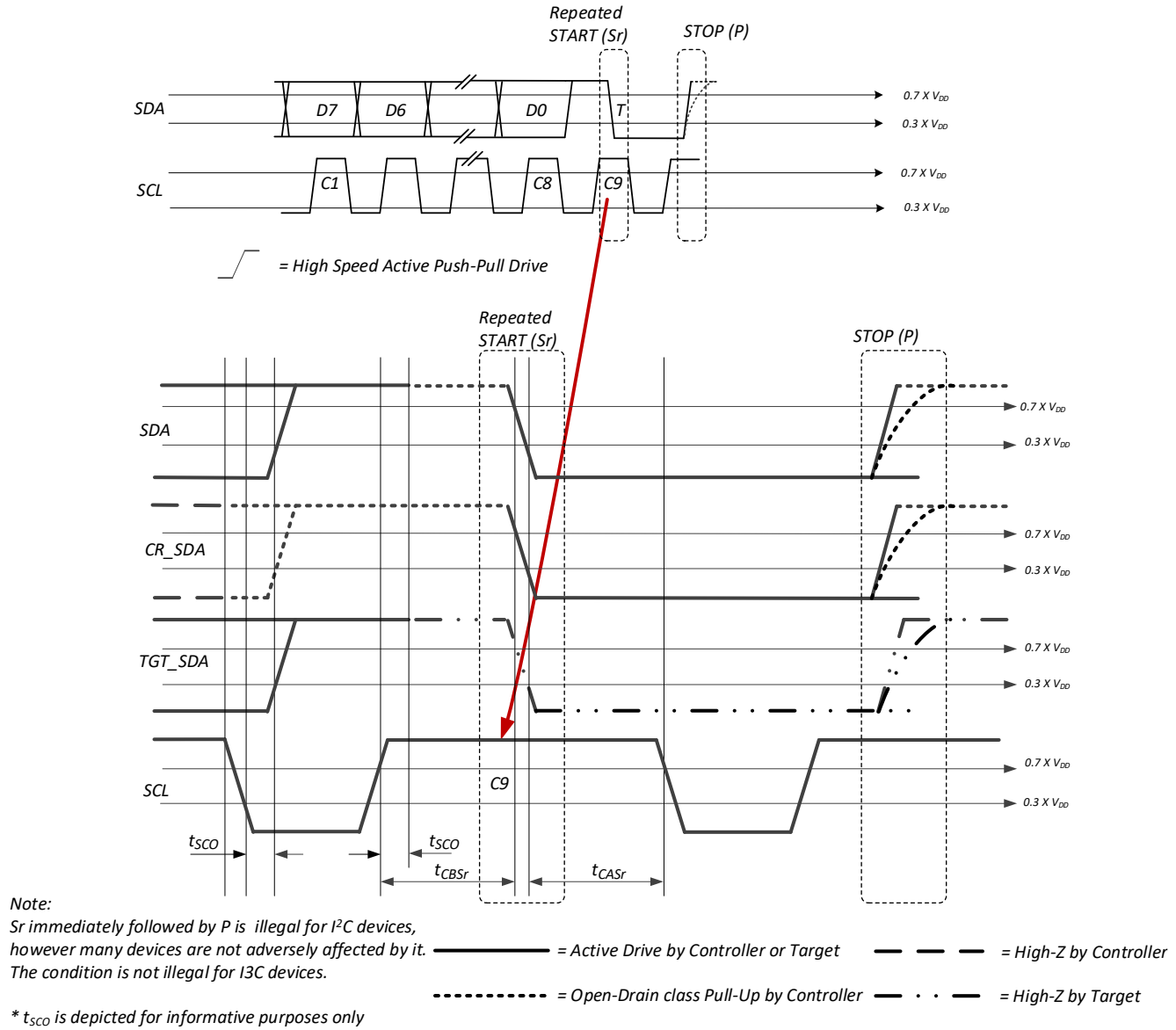
Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



3824

Figure 86 T-Bit When Target and Controller Agree to Continue Read Message

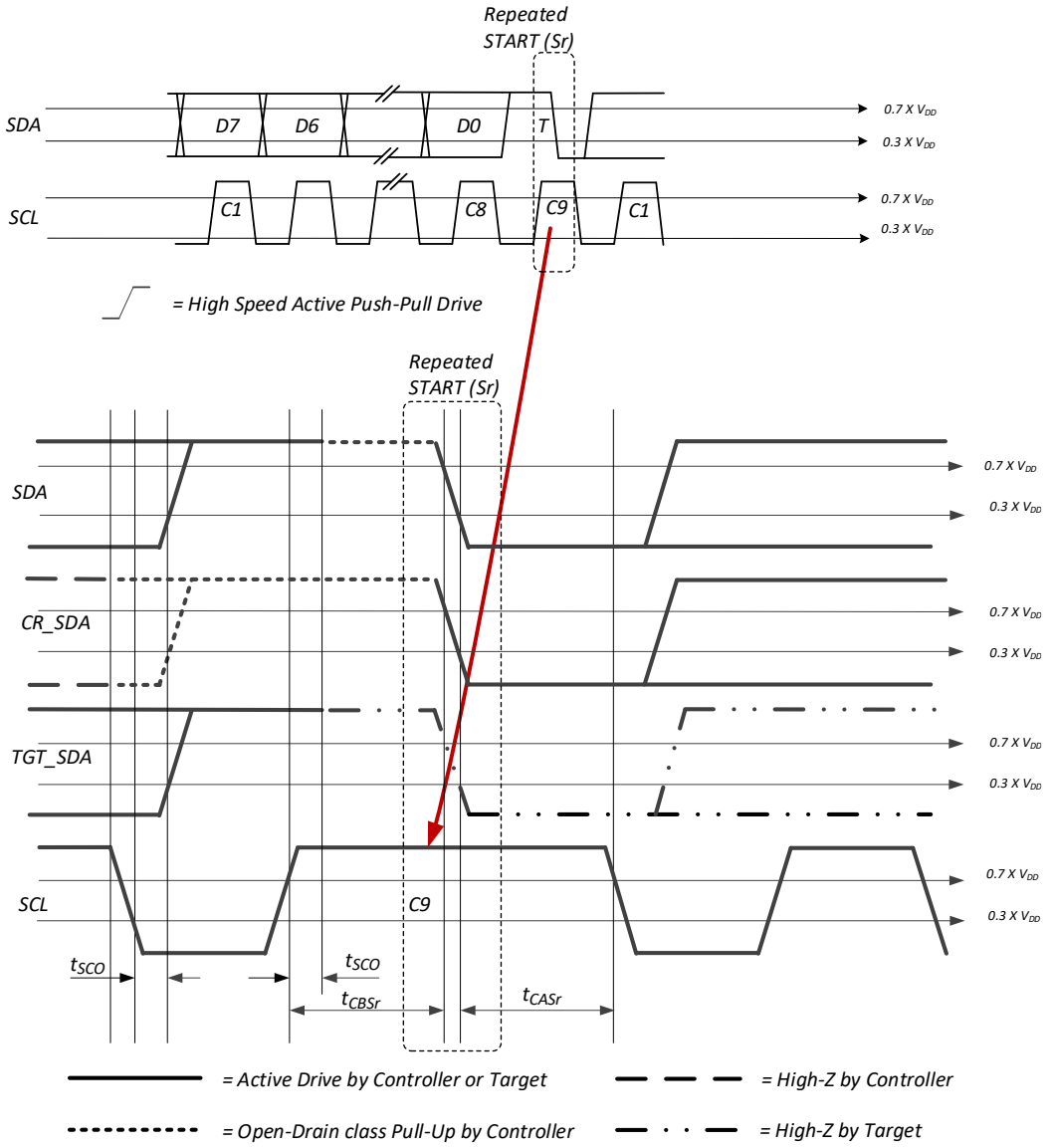
Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



3825

Figure 87 T-Bit When Controller Ends Read with Repeated START and STOP

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

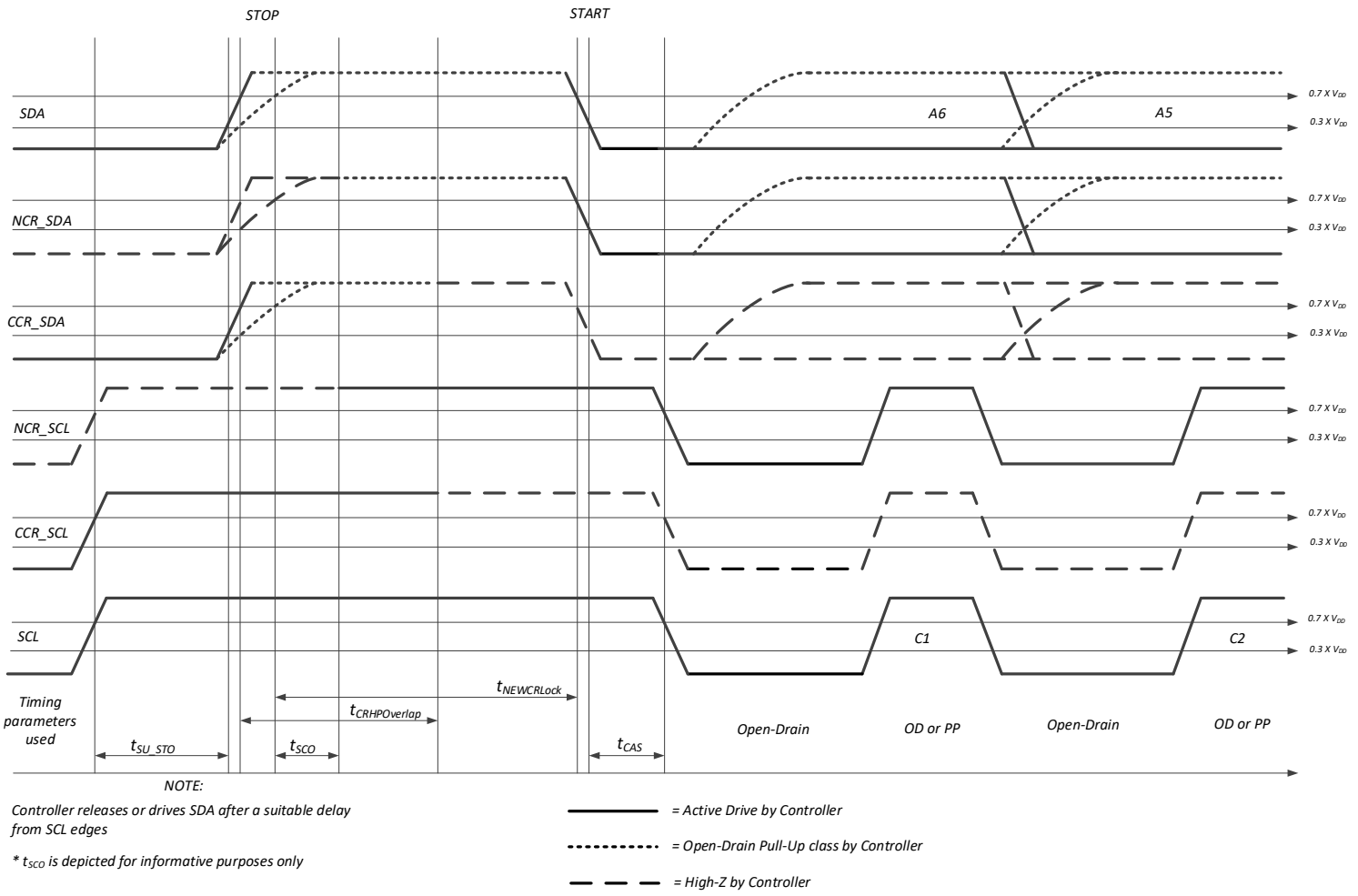


3826

* t_{SCO} is depicted for informative purposes only

Figure 88 T-Bit When Controller Ends Read via Repeated START and Further Transfer

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



3827

Figure 89 Controller to Controller Bus Handoff

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Figure 90 illustrates the I3C Basic timing parameters related to the Spike Filter present on Legacy I^2C Devices. This Spike Filter suppresses pulses shorter than 50 ns.

I3C Basic Mixed Bus clocking rates are chosen to prevent Legacy I^2C Devices from seeing I3C Basic Bus communications, by keeping at least the SCL High periods under 50 ns. To a Legacy I^2C Device, SCL will simply appear to remain Low, i.e., no clock pulses will be received. (Low periods are also generally less than 50 ns, though longer in some cases.)

Figure 90 details how when the slower Open Drain timing (t_{DIG_H}) is used both I3C Basic Devices (bottom trace) and Legacy I^2C Devices (middle trace) can see its longer periods, and therefore both can see the Bus traffic. But when I3C Basic's faster Push-Pull timing ($t_{DIG_H_MIXED}$) is used, the shorter periods will only reach the I3C Basic Devices and not the Legacy I^2C Devices, so only I3C Basic Devices can participate on the Bus. I3C Basic Push-Pull timing is specified in ***Table 50***.

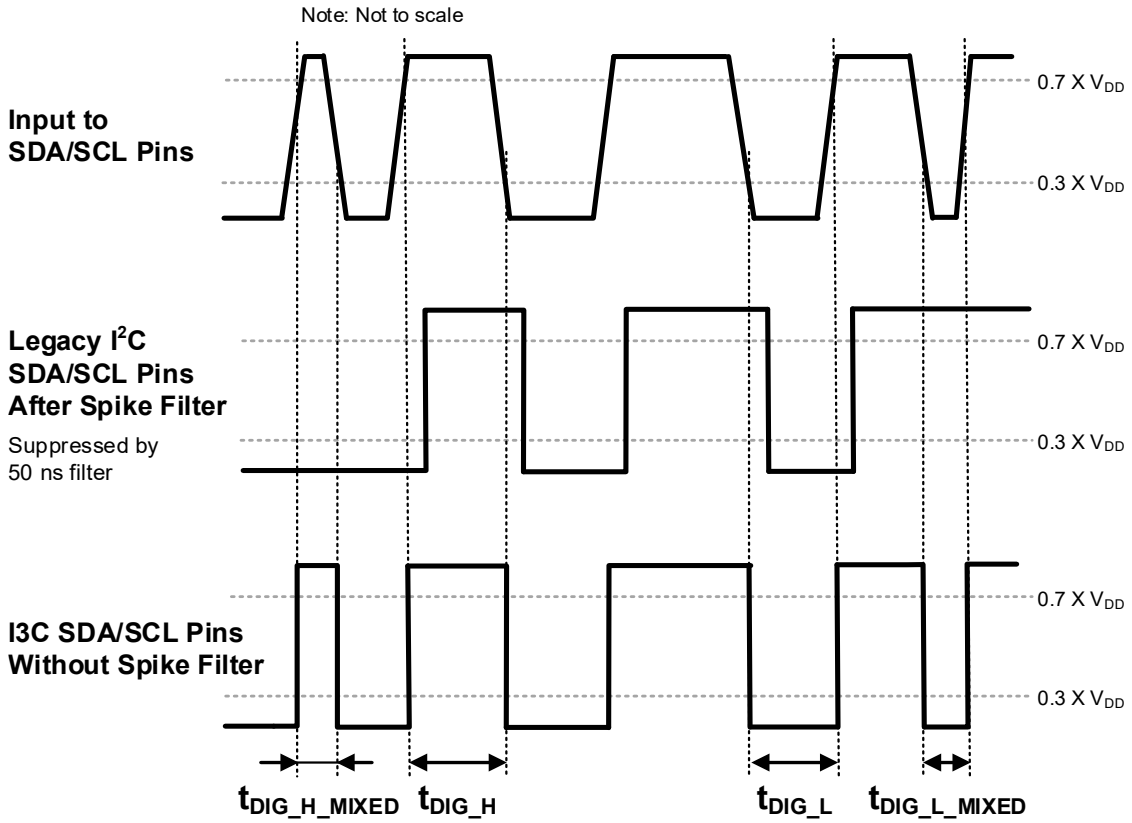


Figure 90 I^2C Spike Filter Behavior

3840 ***Table 51*** through ***Table 53*** detail how timing and drive strength are adjusted during transmission of an I3C
3841 Basic Message.

3842 **Table 51 Timing and Drive for Start of New Frame: No Contention on A6**

	S	Header				Data	P
	START	Addr[6] (Arbitrated)	Addr[5:0]	RnW	ACK	N Data + T-Bit	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
Note: *No contention on Addr[6], so the Controller may optionally use Push-Pull for Addr[5:0] and RnW, per* ***Section 4.3.2.2.2****.*							

3843 **Table 52 Timing and Drive for Start of New Frame: With Contention on A6**

	S	Header				Data	P
	START	Addr[6] (Arbitrated)	Addr[5:0] (Arbitrated)	RnW (Arbitrated)	ACK	N Data + T-Bit	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 Addr[6], so Arbitration continues through Addr[5:0] and RnW.*							

3844 **Table 53 Timing and Drive for Continuation of Frame Using Repeated START**

	S	Header / Data…	Sr	Header		Data	P
	START	…	Repeated START	Addr/RnW	ACK	N Data + T-Bit	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.*							

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

ASSIGNED VALUES

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

5 Assigned Values

This Section lists all MIPI Alliance assigned values for I3C Basic.

Defined Assigned Values

5.1 DCR Byte (Device Configuration Register)

5.2 SETBUSCON Context Byte

5.3 Defining Byte for Core CCCs

5.4 GETCAPS Capability Bits

5.5 In-Band Interrupt MDB (Mandatory Data Byte)

See also: The I3C section of the MIPI Alliance public website ***[MIPI08]***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

5.1 DCR Byte (Device Configuration Register)

Table 54 DCR Byte Values Status

Last updated in I3C Basic Specification Version	I3C Basic v1.2
Is included in I3C Basic	Yes

Each I3C Basic Device that is connected to the I3C Basic Bus shall have an associated read-only Device Characteristics Register (DCR). This read-only register describes the I3C Basic 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 8***.

MIPI Alliance maintains a public registry of approved and pending DCR value assignments ***[MIPI02]***.

Table 55 I3C Basic Device Characteristics Register (DCR)

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

5.2 SETBUSCON Context Byte

Table 56 SETBUSCON Context Byte Values Status

Last updated in I3C Basic Specification Version	I3C Basic v1.2
Is included in I3C Basic	Yes

Table 57 SETBUSCON Context Values

Context Byte Value	Context Group	Context Description
0	None	Reserved, do not use
1 – 63	MIPI I3C Basic Specification v1.Y Minor Version	**Bits[7:6]:** 2'b00 **Bit[5]: I3C Basic Specification Editorial Revision** (within Minor Version) **1'b0**: Version 1.Y.0 **1'b1:** Version 1.Y.1 or greater **Bit[4]: I3C Specification Family** **1'b0:** MIPI I3C Specification **1'b1:** MIPI I3C Basic Specification ***Note:*** *The I3C Basic Specification should be regard as a subset of the I3C Specification.* **Bits[3:0]: I3C Basic Specification Minor Version** (v1.Y) **4'b0000:** Illegal value, do not use *(It would encode Version 1.0, but the SETBUSCON CC was not available in I3C v1.0.Z)* **4'b0001** – **4'b1111:** Version 1.1 – Version 1.15 **Examples:** Bit[5] Bit[4] Bits[3:0] **I3C v1.1.0:** 1'b0 || 1'b0 || 4'b0001 or 8'b00000001 **I3C v1.1.1:** 1'b1 || 1'b0 || 4'b0001 or 8'b00100001 **I3C v1.2.0:** 1'b0 || 1'b0 || 4'b0010 or 8'b00000010 **I3C Basic v1.1.1:** 1'b1 || 1'b1 || 4'b0001 or 8'b00110001 **I3C Basic v1.2.0:** 1'b0 || 1'b1 || 4'b0010 or 8'b00010010
64 – 127	Other MIPI Working Groups	Reserved for higher-level protocols defined by other MIPI Alliance specifications that use MIPI I3C Basic for communications
128 – 191	Other Standards Organizations	Reserved for higher-level protocols defined by other standards developing organizations. The values are assigned by MIPI Alliance I3C WG and published via MIPI's public registry at https://www.mipi.org/MIPI_I3C_bus_context_byte_values_public.html ***[MIPI07]***. For details of each registered value shown below, see the registry. **128:** JEDEC **129:** MCTP **130:** ETSI **131 – 191:** Reserved for future assignment
192 – 255	Vendor Custom	Available for private, per-vendor use (not tracked by MIPI Alliance)

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

5.3 Defining Byte for Core CCCs

3862 **Table 58 Defining Byte Values Status**

Last updated in I3C Basic Specification Version	I3C Basic v1.2
Is included in I3C Basic	Yes

3863 **Table 59 CCCs with Defining Bytes**

CCC Command	Defining Bytes for this CCC		
	DB Value	DB Name	DB Description
ENTTM	**0x00**	**Exit Test Mode**	This value removes all I3C Basic Devices from Test Mode.
	0x01	**Vendor Test Mode**	This value indicates that I3C Basic Devices shall return a random 32-bit value in the Provisioned ID during the Dynamic Address Assignment procedure.
RSTACT	**0x00**	**–**	No Reset on Target Reset Pattern
	0x01	–	Reset the I3C Basic Peripheral Only
	0x02	–	Reset the Whole Target
	0x03	–	Debug Network Adaptor Reset
	0x04	–	Virtual Target Detect
	0x81	–	Return Time to Reset Peripheral
	0x82	–	Return Time to Reset Whole Target
	0x83	–	Return Time for Debug Network Adaptor Reset
	0x84	–	Return Virtual Target Indication

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

CCC Command	Defining Bytes for this CCC		
	DB Value	DB Name	DB Description
MLANE	**0x00**	**SDR-ML Frame Format**	Sets or returns the current ML configuration (i.e., the ML Frame format) for a given I3C Mode.
	0x20	**HDR-DDR-ML Frame Format**	Sets or returns the current ML configuration (i.e., the ML Frame format) for a given I3C Mode.
	0x21	**HDR-TSP-ML Frame Format**	Sets or returns the current ML configuration (i.e., the ML Frame format) for a given I3C Mode.
	0x23	**HDR-BT Frame Format**	Sets or returns the current ML configuration (i.e., the ML Frame format) for a given I3C Mode.
	0x9B	**Group ML Capabilities**	The directly addressed Target responds with one byte (see Additional Data Bytes column) indicating whether it supports separate Group ML configurations.
	0x9C	**Get Group ML Report**	The directly addressed Target responds with a report on the number of separate Group ML configurations (i.e., separate ML Frame formats for a Group Address) that it will support.
	0x7F	**Reset ML**	Targets return to Basic 2-wire I3C mode and a default Data Transfer Coding, for all supported ML Frame formats.
	0xFF	**Get ML Capabilities**	The directly addressed I3C Target responds with a 2-byte description of supported ML Frame formats, in every I3C Mode for which ML is supported.
GETSTATUS	**0x00**	**TGTSTAT**	Returns Target Status (standard format).
	0x91	**PRECR**	Returns alternate Status format describing a Controller-capable Device (i.e., Secondary Controller) per sub-section below.
	0xD7	**NASTAT**	Returns current status of Debug logic in a device that supports MIPI Debug Over I3C.
	0xD8	**SCMSTAT**	Returns the status of secure communications logic in a device that supports MIPI Debug Over I3C.
ENDXFER	0x7F	–	SET and GET parameters for the Data Transfer Early Termination procedure and Ternary Flow Control for the HDR-TSP and HDR-TSL protocols (which are not included in I3C Basic).
	0x55	–	Commands either all I3C Devices on the Bus, or else only some directly addressed Devices on the Bus, to begin operating using the Data Transfer Early Termination procedure and Ternary Flow Control (which are not included in I3C Basic), as previously set up and confirmed using ENDXFER sub-commands 0x7F.
	0xF7	–	SET and GET the parameters for the Data Transfer Early Termination procedure, for the HDR-DDR protocol.
	0xAA	–	Commands either all I3C Basic Devices on the Bus, or else only some directly addressed Devices on the Bus, to begin operating using the Data Transfer Early Termination procedure, as previously set up and confirmed using ENDXFER sub-command 0xF7.
	0xFC	–	(Not included in I3C Basic) SET and GET whether a Monitoring Device's Early Termination capability is enabled vs. disabled.
GETMXDS	**0x00**	–	Describes standard (Target) Write/Read speed parameters, and optional Maximum Read Turnaround time.
	0x91	–	Describes delay parameters for a Controller-capable (i.e., Secondary Controller) Device, and its expected Activity State during the Controller handoff.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

CCC Command	Defining Bytes for this CCC		
	DB Value	DB Name	DB Description
GETCAPS	**0x00**	**TGTCAPS**	Describes standard (Target) capabilities and features.
	0x5A	**TESTPAT**	Return a fixed 32-bit test pattern, to be used for signal integrity and speed testing.
	0x91	**CRCAPS**	Describes capabilities and features relating to a Controller-capable (i.e., a Secondary Controller) Device, and its behavior during the Controller handoff
	0x93	**VTCAPS**	Describes capabilities and features relating to a Virtual Target capable Device.
	0xD7	**DBGCAPS**	Describes capabilities and features relating to a Debug-capable Device conforming to the MIPI Debug Over I3C specification.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

5.4 GETCAPS Capability Bits

Table 60 GETCAPS Capability Bit Values Status

Last updated in I3C Basic Specification Version	I3C Basic v1.2
Is included in I3C Basic	Yes

This Direct CCC allows the Controller to query an I3C Basic Target to determine what optional I3C Basic features that Target supports. It replaces the GETHDRCAP CCC, which was defined in version 1.0 of the I3C Basic Specification.

It has two formats:

- For GETCAPS Format 1, the I3C Basic Target may return the first 2, 3, or all 4 of the GETCAP1 through GETCAP4 Bytes shown in ***Figure 91***. Fields within each GETCAPx Byte are detailed in ***Table 61*** through ***Table 64***.
- The optional GETCAPS Format 2 adds a Defining Byte and returns a variable number of data bytes. The number and format of data bytes returned depends upon the Defining Byte value. GETCAPS Format 2 is detailed below.

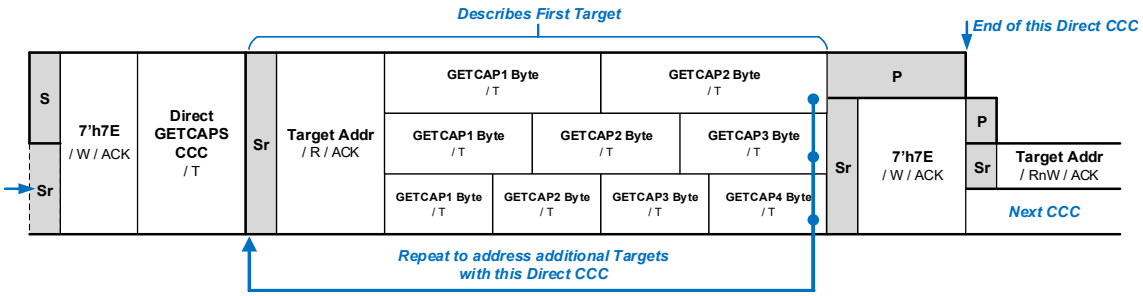


Figure 91 GETCAPS Format 1

Note:

An I3C Basic Controller that supports I3C Basic version 1.1 or greater might be used with an I3C Basic Target Device that supports version 1.0 of the I3C Basic Specification. In this case, such an I3C Basic Target might only support the previous GETHDRCAP CCC format, and as a result might only return one byte (i.e., only the GETCAP1 Byte) as a response to the GETCAPS CCC. Therefore, the I3C Basic Controller must recognize this as a valid configuration, even though the I3C Basic Target would return just a single byte and would have limited capabilities. However, all I3C Basic Targets that support I3C Basic version 1.1 or greater shall return at least the first 2 bytes as a response to the GETCAPS CCC.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3884

Table 61 GETCAP1 Byte Format

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
HDR Mode 7 ***Reserved***	HDR Mode 6 ***Reserved***	HDR Mode 5 ***Reserved***	HDR Mode 4 ***Reserved***	HDR Mode 3: HDR-BT	HDR Mode 2: HDR-TSL ***Not included in I3C Basic***	HDR Mode 1: HDR-TSP ***Not included in I3C Basic***	HDR Mode 0: HDR-DDR

3885

Table 62 GETCAP2 Byte Format

Bits	Field	Description
7	**HDR-DDR Abort CRC**	Is this I3C Target capable of emitting the CRC Word when a transaction in HDR-DDR Mode is aborted? **0:** No **1:** Yes
6	**HDR-DDR Write Abort**	Is this I3C Target capable of issuing the Write Abort in HDR-DDR Mode? **0:** No **1:** Yes
5:4	**Group Address Capabilities**	This 2-bit integer indicates the Group Address function capabilities of this I3C Basic Device. **Values:** **0:** Does not support Group Address function **1:** Can be assigned one Group Address **2:** Can be assigned two Group Addresses **3:** Can be assigned three or more Group Addresses
3:0	**I3C Basic 1.x Specification Version**	This 4-bit integer indicates the minor version number of the MIPI I3C Basic Specification with which this I3C Basic v1.x Device complies. That is, the ‘x’ in “I3C Basic v1.x”. **Examples:** If Bits[3:0] are set to 4'b0001 (decimal 1), then the Device complies with I3C Basic v1.1.1, etc. If Bits[3:0] are set to 4'b0010 (decimal 2), then the Device complies with I3C Basic v1.2 ***Note:*** *An I3C Device that complies with the full I3C Specification shall use the same encoding to indicate the version number of the full I3C Specification to which it complies.* ***Note:*** *Bits[3:0] set to 4’b0000 is an illegal value.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Table 63 GETCAP3 Byte Format

Bit	Field	Description
7	**Timing Control Feature Support**	Does this I3C Target support at least one Timing Control mode? (See Note below table.) **0:** No **1:** Yes
6	**IBI MDB Support for Pending Read Notification**	Does this I3C Basic Target expect that it can send In-Band Interrupt requests with a specific range of Mandatory Data Byte values to indicate a Pending Read Notification, which the Controller shall then follow with a Private Read request to fetch the data? (See ***Section 4.3.6.2.2***.) **0:** No **1:** Yes
5	**HDR-BT CRC-32 Support**	Does this I3C Target support CRC-32 data integrity verification in HDR Bulk Transport Mode? (See ***Section 6.4.3.3***.) **0:** No **1:** Yes
4	**Defining Byte Support in GETSTATUS**	Does this I3C Basic Target support an optional Defining Byte for the GETSTATUS CCC? **0:** No **1:** Yes
3	**Defining Byte Support in GETCAPS**	Does this I3C Basic Target support an optional Defining Byte for this CCC (GETCAPS)? **0:** No **1:** Yes
2	Device to Device Transfer (D2DXFER) IBI Capable	These capabilities are not included in the I3C Basic Specification. To gain access to them, please contact MIPI Alliance.
1	Device to Device Transfer (D2DXFER) Support	
0	**Multi-Lane (ML) Data Transfer Support**	Does this I3C Target support Multi-Lane (ML) Data Transfer? (See ***Section 6.7.3.6*** and ***Section 6.7***.) **0:** No **1:** Yes

Note:

In I3C Basic v1.2 and above: *Bit 7 indicates whether the Timing Control feature is supported. This allows the Controller to determine whether to expect an ACK response to the Direct SETXTIME and Direct GETXTIME CCCs.*

In I3C Basic v1.1.1 and below: *Bit 7 was Reserved. For interoperability with legacy I3C Basic Devices, this field should be cross-checked with GETCAP2[3:0] (I3C Basic 1.x Specification Version). For example, if GETCAP2[3:0] indicates that v1.1.1 is supported, then the status information returned in Bit GETCAP3[7] shall be disregarded.*

In order to obtain the full view of what Timing Control modes are supported, the Controller should follow up with the GETXTIME command:

For Targets supporting I3C v1.2 and above, this can be limited to Devices with GETCAP3[7] set to 1'b1.

For Targets supporting I3C v1.1.1 and below, all Devices must be queried with speculative GETXTIME CCC (i.e., the ACK vs. NACK response is used as a method to inform the Controller about whether the Timing Control feature is supported).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Table 64 GETCAP4 Byte Format

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
HDR Mode 7 ***Reserved***	HDR Mode 6 ***Reserved***	HDR Mode 5 ***Reserved***	HDR Mode 4 ***Reserved***	HDR Mode 3: HDR-BT CCCs Supported	HDR Mode 2: HDR-TSL CCCs Supported ***Not included in I3C Basic***	HDR Mode 1: HDR-TSP CCCs Supported ***Not included in I3C Basic***	HDR Mode 0: HDR-DDR CCCs Supported

5.4.1 GETCAPS Format 2

The optional GETCAPS Format 2 adds a Defining Byte and returns a variable number of data bytes. The number and format of data bytes returned depends upon the Defining Byte value. Before using GETCAPS Format 2 for a given Target, a Controller shall determine whether that Target supports it.

GETCAPS Format 2 conforms to the CCC Direct General Frame Format (***Figure 20***):

A Target that supports the GETCAPS Format 2 CCC shall return a value of 1'b1 in bit 4 of Byte 3 (GETCAP3), when the GETCAPS CCC is sent without a Defining Byte. A Controller that sees that bit set to 1'b1 will know that the Target is capable of supporting at least one Defining Byte with the GETCAPS Format 2 CCC.

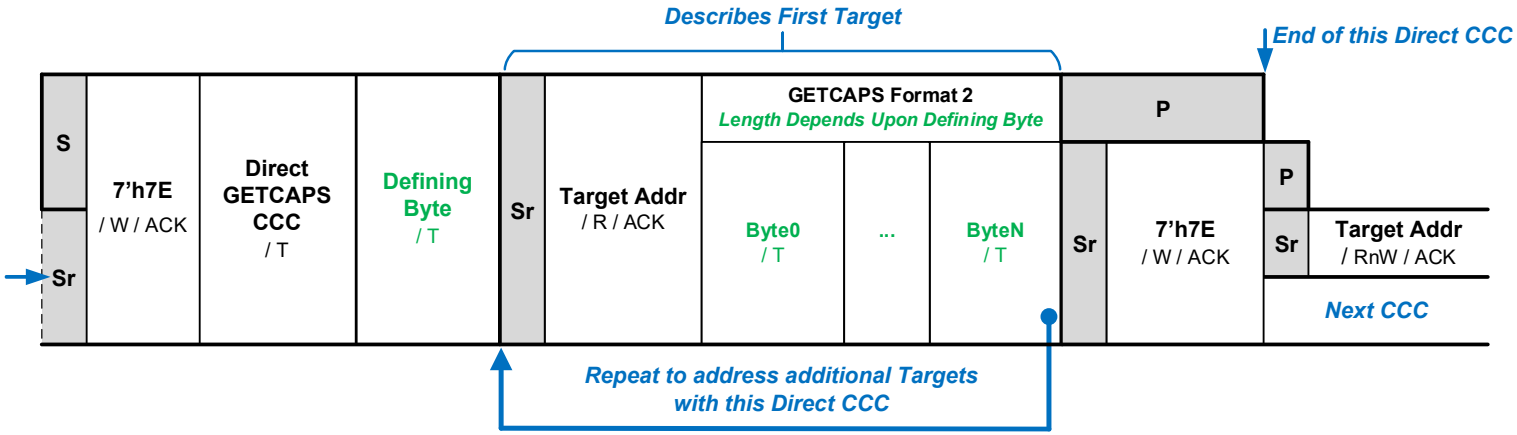


Figure 92 GETCAPS Format 2

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3904 **Table 65 GETCAPS Format 2 Defining Byte Values**

Value	Encoding	Description	Data Bytes Returned
0x00	**TGTCAPS**	Describes standard (Target) capabilities and features. Equivalent to GETCAPS Format 1 (same CCC but without a Defining Byte).	**1 – 4** (GETCAP1–GETCAP4)
0x01 – 0x59	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG	—
0x5A	**TESTPAT**	Return a fixed 32-bit test pattern, to be used for signal integrity and speed testing. Recommended if an I3C Basic Device supports the MIPI Debug Over I3C Specification as a Debug-capable Target System ('TS', see ***[MIPI04]***).	**4** (TESTPAT1–TESTPAT4)
0x5B – 0x90	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG	—
0x91	**CRCAPS**	Describes capabilities and features relating to a Controller-capable (i.e., a Secondary Controller) Device, and its behavior during the Controller handoff (see below). Recommended if an I3C Basic Device is both compliant with version 1.1.1 or higher of the I3C Basic Specification, and advertises as Controller-capable (i.e., the value of BCR[7:6] bits is 2'b01, see ***Section 4.3.1.2.1***).	**1 – 2** (CRCAP1–CRCAP2)
0x92	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG	—
0x93	**VTCAPS**	Describes capabilities and features relating to a Virtual Target capable Device. Required if an I3C Basic Device both is compliant with v1.1.1 or higher of the I3C Basic Specification, and presents multiple Virtual Targets (i.e., the value of BCR[4] bit is 1'b1, see ***Section 4.3.1.2.1***).	**1 – 2** (VTCAP1–VTCAP2)
0x94 – 0xBF	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG	—
0xC0 – 0xD6	**Reserved**	Reserved for future definition by other MIPI Alliance WGs	—
0xD7	**DBGCAPS**	Describes capabilities and features relating to a Debug-capable Device conforming to the MIPI Debug Over I3C specification ***[MIPI04]***. Recommended if an I3C Basic Device supports the MIPI Debug Over I3C Specification as a Debug-capable Target System ('TS', see ***[MIPI04]***).	**1 – N** (DBGCAP1–DBGCAPN)
0xD8 – 0xDF	**Reserved**	Reserved for future definition by other MIPI Alliance WGs	—
0xE0 – 0xFE	**Vendor Extensions**	For Vendor use	—
0xFF	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG	—

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

5.4.2 Read Fixed Test Pattern (GETCAPS Format 2, Defining Byte TESTPAT 0x5A)

This Direct CCC allows the Active Controller to request a Device to return a fixed 32-bit test pattern. This pattern may be used by a Controller to test signal integrity of the I3C Basic Bus or perform speed testing on the I3C Basic Bus.

All Debug-capable Devices should support this test pattern and return it when addressed by the GETCAPS Format 2 CCC with Defining Byte TESTPAT in Target mode. Any Target Devices that share the same Bus as a Debug-capable Device should also support this test pattern.

A Device that supports this Defining Byte shall return a fixed 32-bit test pattern, consisting of the value 0xA55AA55A. The Active Controller may terminate the read on each byte boundary. A Device that does not support this test pattern shall NACK its Target Address.

Additional Defining Bytes that return other test patterns may be defined at the discretion of the implementer, in order to perform more advanced signal integrity or speed testing procedures.

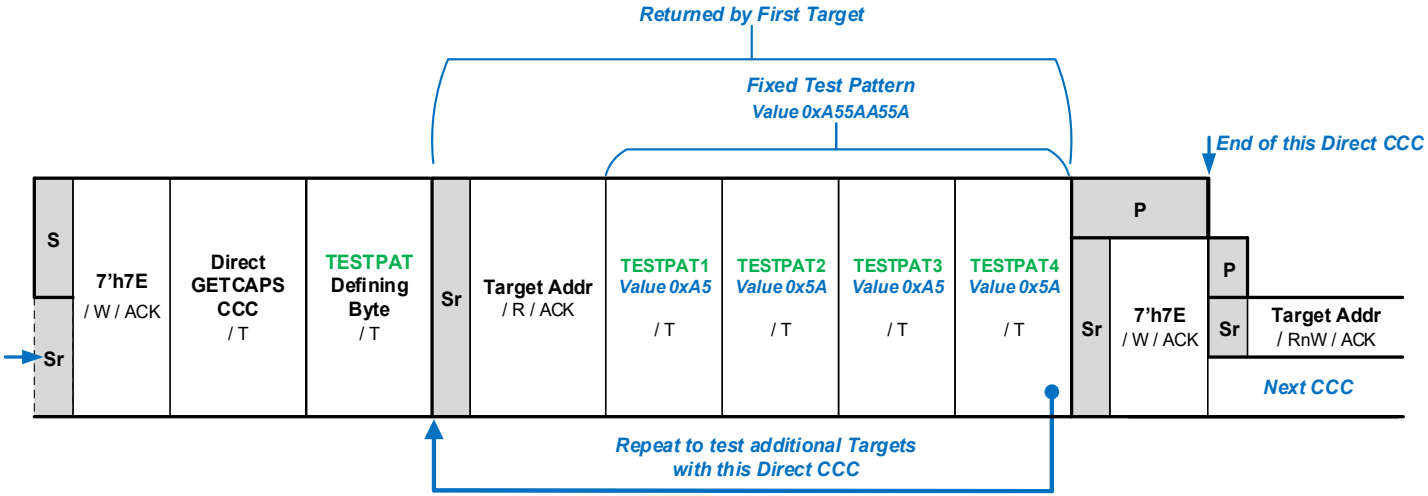


Figure 93 GETCAPS Format 2 with Defining Byte TESTPAT

Table 66 TESTPAT Message Format

Byte	Meaning	Description
TESTPAT1 – TESTPAT4	**Fixed Test Pattern**	Four bytes containing the fixed test pattern value 0xA55AA55A.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

5.4.3 Get Controller Feature Capabilities (GETCAPS Format 2, Defining Byte CRCAPS 0x91)

This Direct CCC allows the Active Controller to query a Controller-capable I3C Device (i.e., one that is currently acting as a Secondary Controller), to read what optional Controller features and capabilities that Device supports. The Active Controller can use this information to decide whether to ACK or NACK a Controller Role Request (CRR) from this Controller-capable Device, in the event that errors or undesirable situations arise from any mismatch of configuration or expectations.

All Controller-capable Devices should support the GETCAPS Format 2 CCC (with Defining Byte) in Target mode. A Device shall support the GETCAPS Format 2 CCC (with Defining Byte) if it does not follow the default capability settings or has limitations on its capabilities while operating as Active Controller.

Per ***Figure 94***, a Controller-capable Device shall return either the CRCAP1 data byte only or both CRCAP1 and CRCAP2, in that order. The Active Controller may terminate the read on each byte boundary. If only the one CRCAP1 data byte is returned, then the Active Controller may assume that the default values for the CRCAP2 data byte applyCRCAP2.

A Device that is not Controller-capable shall NACK its Target Address.

Fields within the CRCAP1 and CRCAP2 data bytes indicate various Controller features and capabilities, as detailed in ***Table 67*** and ***Table 68, respectively.***

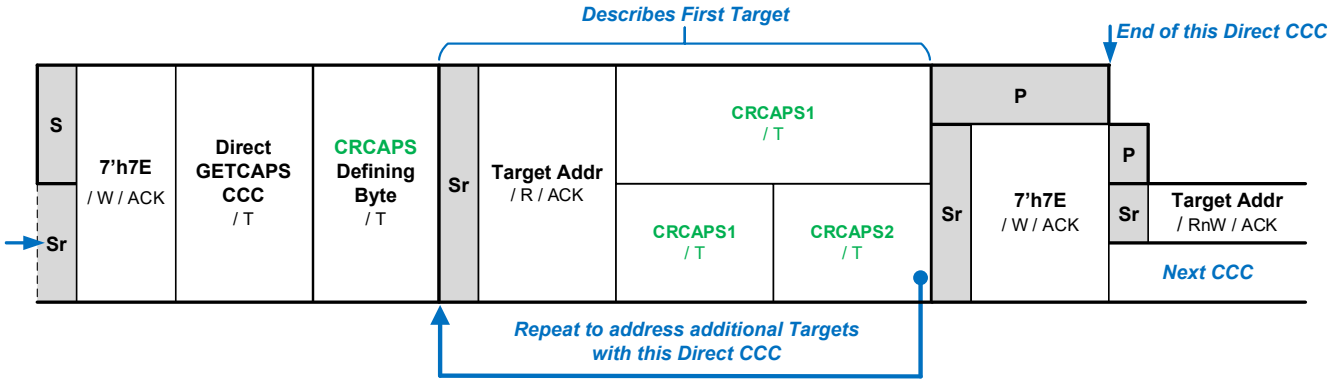


Figure 94 GETCAPS Format 2 with Defining Byte CRCAPS

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3926

Table 67 CRCAP1 Byte Format

Bits	Field	Description
7:3	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG
2	**Multi-Lane Support**	**1'b1:** This Device may use the MLANE CCC to change the ML configuration of other I3C Targets (***Section 6.7.3.6***). **1'b0:** This Device shall not use the MLANE CCC to change the ML configuration of other I3C Targets.
1	**Group Management Support**	**1'b1:** This Device may support Group Address handoff or management capabilities and shall use the DEFGRPA CCC to announce the current group membership data after making any changes. **1'b0:** This Device does not support Group Address capabilities. It shall not use the SETGRPA or RSTGRPA CCCs. It will ignore any DEFGRPA CCC Broadcasts while it is not Active Controller, and it shall not send DEFGRPA CCC Broadcasts before a Controller Role handoff to another Controller-capable Device.
0	**Hot-Join Support**	**1'b1:** This Device may support Hot-Join and will ACK an I3C Target Hot-Join Request while it is Active Controller. It shall support DAA and also send the DEFTGTS CCC to announce the presence of newly joined I3C Targets to other Controller-capable Devices. **1'b0:** This Device does not support Hot-Join, and will NACK any I3C Target Hot-Join Requests. It may also send DISEC CCC with the DISHJ bit set to disable Hot-Join Requests.

3927

Table 68 CRCAP2 Byte Format

Bits	Field	Description
7:4	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG
3	**Delayed Controller Handoff**	**1'b1:** This Device may need additional time to process Broadcast CCC data sent from the Active Controller, before it can successfully complete the Controller Role Handoff procedure and accept the Controller Role: **1'b0:** This Device does not need additional time to process Broadcast CCC data sent from the Active Controller.
2	**Deep Sleep Capable**	**1'b1:** This Device may enter a deep sleep state during which it may miss some Broadcast DEFTGTS and DEFGRPA CCCs sent by the Active Controller. see ***Section 4.3.7.3.15***). **1'b0:** This Device shall not enter a deep sleep state.
1	**Controller Pass-Back**	**1'b1:** This Device shall automatically pass the Controller Role back to the former Active Controller from which it received its Controller Role. It shall do so via the GETACCCR CCC, once it has finished performing any tasks that require it to request and gain the Controller Role. **1'b0:** This Device shall not automatically pass the Controller Role back to the former Active Controller.
0	**In-Band Interrupt Support**	**1'b1:** This Device may choose to accept an IBI request from any I3C Basic Target, when operating as Active Controller. It may selectively or globally enable interrupts upon receiving the Controller Role, and shall disable interrupts before passing the Controller Role on to another Controller-capable Device. **1'b0:** This Device shall not accept any IBI requests, and will respond with NACK.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

5.4.4 Get Debug Feature Capabilities (GETCAPS Format 2, Defining Byte 0xD7 DBGCAPS)

This Direct CCC allows the Active Controller to query a Debug-capable I3C Basic Device, to read the version of the MIPI Debug Over I3C specification supported as well as the optional Debug features and capabilities that Device supports. An Active Controller can use this information to determine which I3C Basic Devices on the Bus support the Debug features and capabilities defined in the MIPI Debug Over I3C Specification ***[MIPI04]***.

All Debug-capable Devices should support the GETCAPS Format 2 CCC (with Defining Byte) in Target mode.

Per ***Figure 95***, a Debug-capable Device shall return at least one data byte with additional data bytes depending on the Debug features and capabilities of the Device. The data bytes shall be returned in a defined order, depending on the MIPI Debug Over I3C version, per ***Table 69***. The meaning and formatting of the data bytes is fully described in ***Section 6*** of the MIPI Debug Over I3C Specification ***[MIPI04]***.

The Active Controller may terminate the read on each byte boundary. The Active Controller shall not make any assumptions about the presence of any Debug features or capabilities, beyond the contents of any data bytes received from the Device.

A Device that is not Debug-capable shall NACK its Target Address.

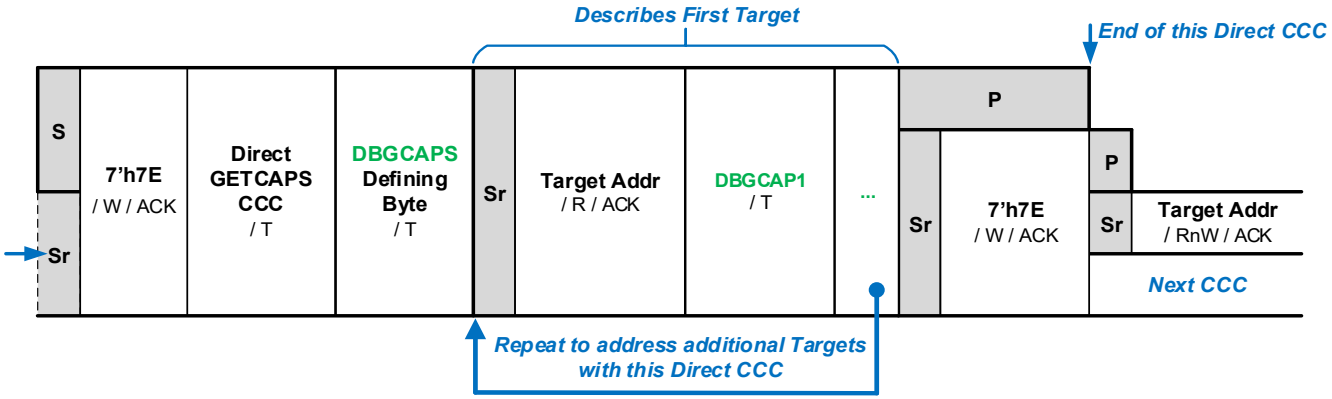


Figure 95 GETCAPS Format 2 with Defining Byte DBGCAPS

Table 69 DBGCAPS Message Format

Byte	Meaning	Description
DBGCAP1	**Debug Over I3C Version**	This byte indicates the version of the MIPI Debug Over I3C specification ***[MIPI04]*** supported by this Device. (Required)
DBGCAP2 – DBGCAP*N*	**Debug Features and Capabilities**	Number, meaning, and formatting of all subsequent bytes are defined in the MIPI Debug Over I3C specification ***[MIPI04]***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

5.4.5 Get Virtual Target Capabilities (GETCAPS Format 2, Defining Byte 0x93 VTCAPS)

This Direct CCC allows the Active Controller to query a Virtual Target capable I3C Basic Device, to read what optional features and capabilities that Device supports. The Active Controller can use this information to fully configure the Bus, and to enable advanced applications using Virtual Targets with multiple Dynamic Addresses in a single Device.

All Devices that report Virtual Target capabilities should support this CCC. For additional information on Virtual Target capabilities, refer to the MIPI Alliance I3C Application Note: Virtual Devices and Virtual Targets ***[MIPI10]***.

Per ***Figure 96***, a Virtual Target capable Device shall return either the VTCAP1 data byte only, or both VTCAP1 and VTCAP2, in that order. The Active Controller may terminate the read on each byte boundary. If only the one VTCAP1 data byte is returned, then the Active Controller may assume that the default values for the VTCAP2 data byte apply.

Fields within the VTCAP1 and VTCAP2 data bytes indicate various Virtual Target features and capabilities. ***Table 70*** covers the features and capabilities dealing with the Device's configuration. ***Table 71*** advertises how the Device handles any downstream Virtual Targets or other Devices, if those are supported.

A Device that is not Virtual Target capable shall NACK its Target Address.

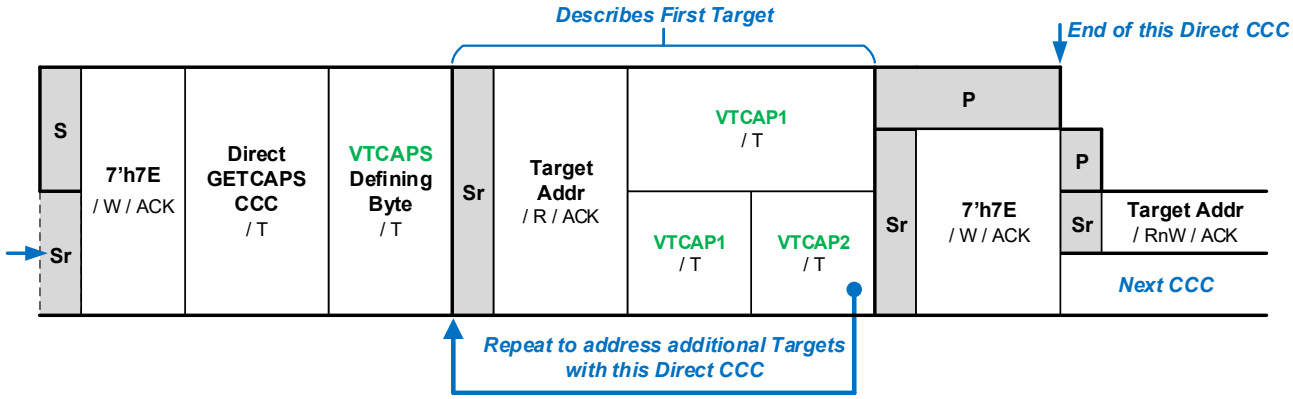


Figure 96 GETCAPS Format 2 with Defining Byte VTCAPS

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3952

Table 70 VTCAP1 Byte Format

Bits	Field	Description
7:6	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG
5	**Shared Peripheral Detect**	**1'b0:** Device does not use shared Peripheral logic. **1'b1:** Device uses shared Peripheral logic and supports the Virtual Target Detect operation as a Defining Byte for the RSTACT CCC (see ***Section 4.3.7.3.23***). Required for Virtual Target type 3'd5; Recommended for other Virtual Target types. See also ***Section 5.2.2*** in MIPI Alliance I3C Application Note: Virtual Devices and Virtual Targets [***MIPI10***].
4	**Side Effects**	**1'b0:** No impact on other Virtual Targets supported by this Device, if sending CCCs to reconfigure this Target's Dynamic Address. **1'b1:** Configuration CCCs sent to this Virtual Target's Dynamic Address may impact other Virtual Targets that are connected to or exposed by this Device.
3	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG
2:0	**Virtual Target Type**	**Values:** **3'd0:** This Device does not support or declare any Virtual Target capabilities. **3'd1:** This Device is a Bridge Device, and uses the SETBRGTGT CCC (***Section 4.3.7.3.17***) to configure its virtualized (bridged) Targets. **3'd2:** This Device is a Bridge Device, but does not require the use of the SETBRGTGT CCC; the Device will set up its own bridged Targets. **3'd3:** This Device is a Routing Device, and uses the SETROUTE CCC (***Section 4.3.7.3.20***) to configure its Routes to one or more I3C Basic Buses. **3'd4:** This Device is a mixed-function Debug Virtual Target, which uses a single Target Address to access both Debug functionality and application functionality. **3'd5:** This Device supports multiple Virtual Target Addresses with shared Peripheral logic. **3'd6:** This Device is a Hub that supports pass-through transactions to other Target Devices. 3'd7: Reserved

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Table 71 VTCAP2 Byte Format

Bits	Field	Description
7:5	**Reserved**	Reserved for future definition by MIPI Alliance I3C WG
4:3	**Bus Context and Conditions** *Valid for Virtual Target types 3'd1, 3'd2, and 3'd3 only.*	**2'd0:** Device manages all downstream Device configuration automatically. **2'd1:** Device will re-broadcast any context and event enable/disable commands to all downstream Devices when received from the Active Controller. Device can request Active Controller to send a fresh configuration. **2'd2:** Device will cache previously Broadcast context and event enable/disable configuration and use that to configure all new downstream Devices that join its downstream Bus segments. 2'd3: Reserved
2	**Address Remapping** *Valid for Virtual Target types 3'd1, 3'd2, and 3'd3 only.*	**1'b0:** Device will pass addresses of downstream Devices as they are assigned. **1'b1:** Device will transparently re-map addresses of downstream Devices to mask them from this Bus.
1:0	**Interrupt Requests** *Valid for Virtual Target types 3'd1, 3'd2, and 3'd3 only.*	**Values:** **2'd0:** Device will not coalesce or route interrupt requests from any downstream Devices. **2'd1:** Device must coalesce interrupt requests from downstream Devices, using this Target Address. **2'd2:** Device must route interrupt requests from downstream Devices using their Virtual Target Addresses. **2'd3:** Device may do either (i.e., coalesce and/or route) when forwarding interrupts from downstream Devices.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

5.5 In-Band Interrupt MDB (Mandatory Data Byte)

Table 72 In-Band Interrupt MDB Values Status

Last updated in I3C Basic Specification Version	I3C Basic v1.2
Is included in I3C Basic	Yes

The Mandatory Data Byte of the IBI payload is the data that follows the Dynamic Address when a Device sends an IBI request. The availability of a Mandatory Data Byte, together with the payload information that follows it, is indicated by bit BCR[2] in the Bus Characteristics Register (see ***Section 4.3.1.2.1***).

It is called a Mandatory Data Byte because the Target Device takes over the line when the Controller ACKs the IBI request and the Target continues to send data; the Controller cannot decline the reception of the first byte, and must wait for the T-Bit to terminate any subsequent data.

The Mandatory Data Byte provides the Controller additional information about the event that has happened, and is divided in two fields as shown in ***Table 13***:

- **Interrupt Group Identifier:** The three most significant bits: MDB[7:5]

  Values for the Interrupt Group Identifier are defined by MIPI Alliance I3C WG for different use cases.

- **Specific Interrupt Identifier:** The five least significant bits: MDB[4:0]

Example: D2DT is a MIPI Alliance I3C WG function. It uses the value 3'b001 for the Interrupt Group Identifier field, and the value 5'b10111 (5'h17) for the Specific Interrupt Identifier field.

MIPI Alliance maintains a web-based registry of defined MDB values ***[MIPI05]***.

Table 73 Mandatory Data Byte Field Format

Interrupt Group Identifier Field			Specific Interrupt Identifier Field				
MDB[7]	MDB[6]	MDB[5]	MDB[4]	MDB[3]	MDB[2]	MDB[1]	MDB[0]

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3971

Table 74 Mandatory Data Byte Value Ranges

MDB Category	Interrupt Group Identifier	Specific Interrupt Identifier Value		Description
	MDB[7:5]	MDB[4:0]		
User Defined	3'b000	5'h00 – 5'h1F	Vendor Reserved	Defined by the vendor and reserved for vendor definition
I3C WG	3'b001	5'h00 – 5'h0C	MIPI Alliance I3C WG Reserved	Defined and interpreted based on some reserved values allocated by MIPI Alliance I3C WG. Indicates that the interrupt is related to a specific functionality, e.g., D2DT.
		5'h0D	Error with additional data bytes in IBI payload	
		5'h0E	Generic error, no additional data bytes in IBI payload	
		5'h0F – 5'h11	MIPI Alliance I3C WG Reserved	
		5'h12	Monitoring Devices Interrupt Enabling	
		5'h13 – 5'h16	MIPI Alliance I3C WG Reserved	
		5'h17	D2DT Identifier	
		5'h18 – 5'h1F	MIPI Alliance I3C WG Reserved	
MIPI Groups	3'b010	5'h00 – 5'h01	MIPI Alliance Camera WG Reserved	Allocated and reserved by MIPI Alliance Working Groups, and approved by MIPI Alliance I3C WG. Indicates that the interrupt is generated by a Device that wants to send a specific MIPI Alliance WG-related interrupt (e.g., Camera WG or Debug WG)
		5'h02 – 5'h1B	MIPI Alliance Reserved	
		5'h1C – 5'h1D	MIPI Alliance Debug WG Reserved	
		5'h1E – 5'h1F	MIPI Alliance Reserved	
Non MIPI Reserved	3'b011	5'h00 – 5'h1F	Non MIPI Reserved	Allocated for uses outside of MIPI Alliance
Timing Information	3'b100	5'h00 – 5'h1F	Vendor Reserved	May have any value defined by the vendor
Pending Read Notification	3'b101	5'h00 – 5'h0C	MIPI Alliance I3C WG Reserved	Indicates that the interrupt is generated by a Device that wants to send a specific MIPI Alliance WG-related or vendor-specific interrupt, and has a data message that will be returned on the next Private Read if ACKed (see ***Section 4.3.6.2.2***)
		5'h0D	MIPI Alliance Debug WG Reserved	
		5'h0E	MCTP Packet (for DMTF)	
		5'h0F	MIPI Alliance I3C WG Reserved	
		5'h10 – 5'h1F	Vendor Reserved	
Reserved	3'b110	—		Reserved
Reserved	3'b111	—		Reserved

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

OPTIONAL FEATURES

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6 Optional Features

Section 6 specifies all I3C Device Optional Features (OFs) defined by MIPI Alliance. Every defined OF is specified independently in a separate sub-section of ***Section 6***, each with an independent version identifier and an independent change summary.

An I3C Device may optionally implement zero or more of the Optional Features (OFs) specified in ***Section 6***. An I3C Device could implement none of the OFs specified in ***Section 6***, yet still be a complete, functioning I3C Device.

Note:

Some OFs are not technically compatible with one or more other OFs. This is noted in the table at the start of each OF section.

Optional Normative: The OF specifications in ***Section 6*** are Optional Normative. This means that although the decision of whether or not to implement any given OF is left to the I3C Device implementer, if a given OF is implemented then it shall conform to the specification given in ***Section 6***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.1 HDR Modes

This Section describes the flows and framings that are common to optional HDR Modes.

6.1.1 Technical Overview (Informative)

These HDR Modes are designed to transfer more data at the same Bus frequency:

- **HDR Double Data Rate (HDR-DDR) Mode**: see ***Section 6.2***
- **HDR Bulk Transport (HDR-BT) Mode**: see ***Section 6.4***

Note:

*For information, the full I3C Specification also includes two additional HDR Modes using Ternary encoding (see **Section 6.3**) :*

• NOT INCLUDED IN I3C BASIC: HDR Ternary Symbol Pure-bus (HDR-TSP) Mode

• NOT INCLUDED IN I3C BASIC: HDR Ternary Symbol Legacy-inclusive-bus (HDR-TSL) Mode

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. The essential basic I3C specifications are found in **Section 4**.*

Details common to all HDR Modes are detailed in this Section and in ***Section 4.3.10***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.1.2 HDR Restart Pattern

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. All I3C Targets shall detect and respond to the HDR Restart Pattern while operating in any HDR Mode that the Target supports. The HDR Restart Pattern Detector is specified in ***Section 6.1.2***. Note that unlike the HDR Exit Pattern, the HDR Restart Pattern is only detected by I3C Targets that support the current HDR Mode.

The HDR Restart Pattern is based on a subset of the HDR Exit Pattern. It is defined thus:

- SDA starts High, SCL starts Low (same as HDR Exit Pattern)
- SDA toggles 4 times (fall, rise, fall, rise)
- The next edge is SCL rising. SDA may change with SCL rising, but SCL shall rise.
- Each transition on SDA and/or SCL is separated by a time interval of at least t_{DIG_H} (see ***Section 4.3.11.2***)

Figure 97 illustrates an HDR Restart Pattern (with edges to set SCL and SDA up into correct state) along with the necessary SCL ending edge.

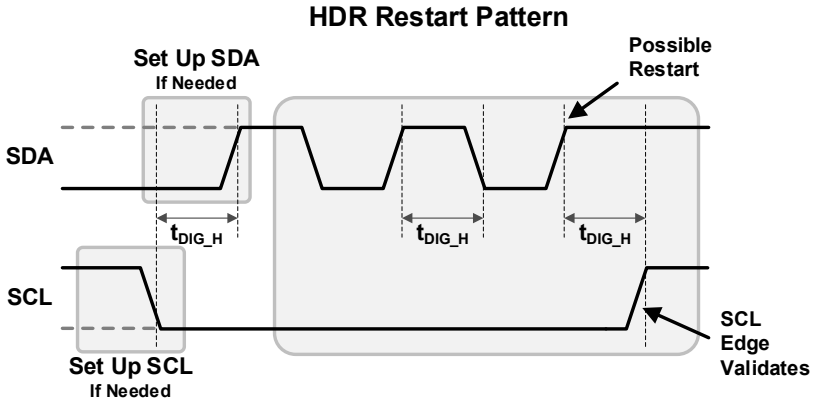


Figure 97 HDR Restart with Next Edge

Note:

After the necessary SCL ending edge, the I3C Bus remains in that HDR Mode, and the Controller typically follows by driving SCL Low to complete the SCL pulse, which signals the start of the next HDR Mode transfer. This begins the first structured protocol element (i.e., a Command or Header) for the HDR Mode framing, per ***Section 6.1.4.2****.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.1.2.1 HDR Restart and Exit Pattern Detector

Any I3C Target that supports at least one HDR Mode shall include HDR Restart Pattern detection. While this function is easily incorporated into the required HDR Exit Pattern Detector, or may be part of HDR Mode support, the particular design is not mandated by this Specification (i.e., is up to the manufacturer). The HDR Restart Pattern Detector may be implemented either in digital logic or in software (Bit Banged).

The remainder of this Section presents an example digital logic implementation in both schematic view and in RTL code, building upon the HDR Exit Pattern Detector presented in ***Section 4.3.10.2.1***.

The basic logic model is that SCL is held Low (0) and so SCL High (1) will reset the main detector. It also uses falling edges of SDA primarily, hence treats SDA as a clock. The HDR Restart Pattern is then detected only when two falling edges have been seen, and verifies the rising edge and then SCL change that is required for an HDR Restart.

The combined HDR Restart and Exit Pattern Detector schematic is shown in ***Figure 98***.

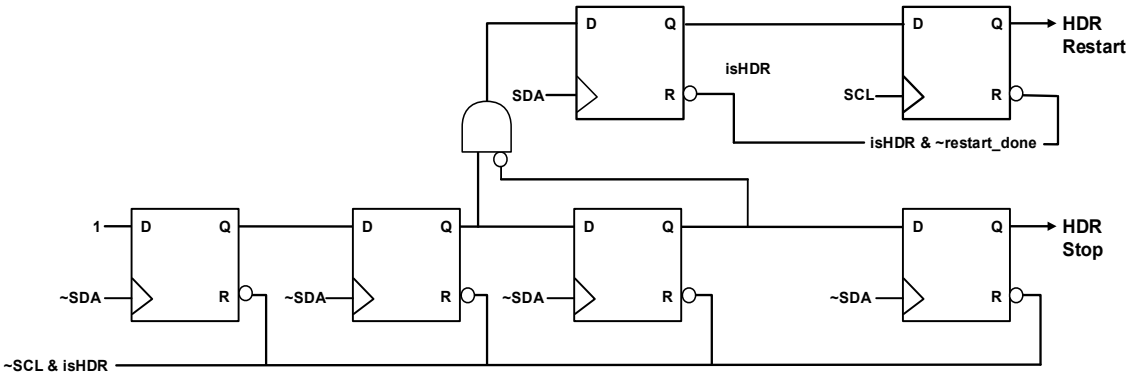


Figure 98 Combined HDR Restart and Exit Pattern Detector (Schematic)

This Detector design builds on the HDR Exit Pattern Detector. After exactly two SDA falling edges are seen, followed by a rising edge, the HDR Restart Pattern Detector checks for the rising edge of SCL. It does not matter whether SDA also falls at the same time; the rising SCL is the key. This works because even if SDA were falling (and therefore triggering the next flop in the HDR Exit Pattern Detector), the upper-left flop will still hold 1 because it has not yet seen a rising edge on SDA.

Note:

*Unlike the HDR Exit Pattern Detector, the HDR Restart Pattern Detector is only enabled when the Target is currently in a supported HDR Mode, and **not** when the Target has entered a Target Error state (i.e., one that can be recovered by the HDR Exit Pattern).*

An example RTL implementation of the ***Figure 98*** schematic is shown in ***Listing 2***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Listing 2 Example RTL Code for Combined HDR Pattern Detector: Exit and Reset

```
wire          scl_rst_n;
wire          restart_rst_n;
wire          SDA_clk_n;      // ~SDA as clock
reg[3:0]     stp_cnt;        // HDR STOP counter (shift chain)
reg           poss_restart;   // possible restart
reg           is_restart;

assign scl_rst_n     = ~iSCL & iIsInHDR; // SCL=1 resets
assign restart_rst_n = ~iRestartDone & iIsInHDR;

// next uses glitch free XOR for SDA clock. Only inverts
// SDA as clock if not in scan. Could add a clock mux
// so uses one common clock in scan.
SAFE_CLK_XOR sda_neg_clk(iSDA, ~scan_mode, SDA_clk_n);

// counter for 3 and 4 rising SDA when SCL is High
// (else SCL resets). Whole thing held in reset if
// not in HDR Mode
always @ (posedge SDA_clk_n or negedge scl_rst_n)
  if (~scl_rst_n)
    stp_cnt <= 4'd0;          // SCL High or not HDR
  else
    stp_cnt <= {stp_cnt[2:0], 1'b1}; // shift chain counter

assign oHDR_STOP = stp_cnt[3];// detects Exit (STOP)

// Possible Restart means exactly 2 falling edges
// of SDA and then rising edge of SDA. Actual
// Restart is from SCL then rising
always @ (posedge iSDA or negedge restart_rst_n)
  if (~restart_rst_n)
    poss_restart <= 1'b0;     // Restart ACK or not HDR
  else if (stp_cnt[1] & ~stp_cnt[2]) // after 2nd fall only
    poss_restart <= 1'b1;     // SDA rise after 2 falls
  else
    poss_restart <= 1'b0;     // else not possible restart

always @ (posedge SCL or negedge restart_rst_n)
  if (~restart_rst_n)
    is_restart <= 1'b0;       // Restart ACK or not HDR
  else if (poss_restart)
    is_restart <= 1'b1;       // SCL rises after SDA does
  else
    is_restart <= 1'b0;       // else not restart
```


Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.1.3 CCC Framing in HDR Modes

Special framings are used for transmitting Common Command Codes in HDR Modes. CCCs that are supported in any HDR Modes may be sent by an Active Controller, so long as it has entered the HDR Mode using the Enter HDR Mode CCC (see CCCs **Enter HDR Mode 0 through 7**, ***Section 4.3.7.3.9***). Both Broadcast and Direct CCCs are supported, with framing details specific to each HDR Mode. As noted in ***Table 75***, not all CCCs may be used in HDR Modes. Additionally, support for any CCCs in any particular HDR Mode for a Target is optional. Target Devices may implement support for some or all of the Command Codes that per ***Table 75*** may be used in HDR Modes, as well as other Command Codes reserved for use by a particular MIPI Alliance Working Group or set aside for Vendor Extensions. Generally, Target Devices may implement support in HDR Modes for a subset of all permitted Command Codes that they might support in SDR Mode.

The Controller shall know which Targets are active and support a particular HDR Mode, in order to determine whether a given Target will understand any Broadcast or Direct CCCs sent within an active HDR Mode. The Controller shall also determine which Targets support I3C v1.1 or later, and if so, which of them support a CCC within that HDR Mode.

- **For Targets that support I3C v1.1.1 or earlier:** The Controller should have *a priori* knowledge, otherwise it may determine this by attempting to receive an acknowledgement of a Direct CCC in a manner specific to the HDR Mode (see ***Section 6.1.3.3***).
- **For Targets that support I3C v1.2 or later:** The Controller should use the GETCAP4 byte in the Target's response to the GETCAPS Format 1 CCC to determine whether that Target supports CCCs in a given HDR Mode.

For any Targets that do not support a given CCC in that HDR Mode, the Controller may need to re-send that CCC in another supported HDR Mode, or in SDR Mode.

Table 75 summarizes which CCCs are permitted in HDR Modes, noting any per-HDR-Mode limitations and other advisory information.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4107 **Table 75 CCCs Permitted in HDR Modes, with Notes and Limitations**

CCC Type		Command Name	Notes and Limitations
Broadcast	Direct		
X	X	**ENEC** **DISEC**	Might not generally be useful in HDR Modes, as the Controller would need to exit HDR and return to SDR Mode for the Target to be able to raise any of the request types that are affected by these CCCs; see ***Section 6.1.3.4***.
X	X	**ENTAS0** **ENTAS1** **ENTAS2** **ENTAS3**	See ***Section 6.1.3.4***.
X	–	**DEFTGTS**	Not recommended for use in HDR Modes; generally only used for Secondary Controllers, to announce lists of known Targets, following changes to Target Dynamic Addresses or Hot-Join events, neither of which are permitted within HDR Modes
X	X	**SETMWL** **SETMRL**	See ***Section 6.1.3.4***.
–	X	**GETMWL** **GETMRL**	No limitations
X	–	**SETBUSCON**	Not recommended for use in HDR Modes
X	X	**ENDXFER**	Not recommended for use in HDR Modes
X	X	**SETXTIME**	No limitations
–	X	**GETXTIME**	No limitations
X	X	**RSTACT**	Not recommended for use in HDR Modes
–	X	**SETGRPA**	Target support for Group Addresses may vary; Target support for receiving HDR CCCs directed to Group Addresses may vary; see ***Section 6.1.3.4***.
X	–	**DEFGRPA**	Generally only used for Secondary Controllers, to announce lists of Targets assigned to a known Group Address after prior use of **SETGRPA** and **RSTGRPA** CCCs
X	X	**RSTGRPA**	See ***Section 6.1.3.4***.
X	X	**MLANE**	Broadcast or Direct SET versions of this CCC are generally not recommended for use in HDR Modes; Targets that support Direct GET versions of this CCC shall respond according to the HDR Mode framing for the active HDR Mode, instead of the “test mode” which returns the same data on any Additional Data Lanes (as defined in SDR Mode, see ***Figure 161***); see ***Section 6.1.3.4***. and ***Section 6.7.3.5***.
–	X	**GETBCR** **GETDCR**	Not recommended for use in HDR Modes; usually only used once per Target, during Bus Initialization
–	X	**GETSTATUS**	No limitations
–	X	**SETBRGTGT**	See ***Section 6.1.3.4***.
–	X	**GETMXDS**	No limitations
–	X	**GETCAPS**	No limitations
–	X	**SETROUTE**	See ***Section 6.1.3.4***.
–	X	**D2DXFER**	No limitations

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4108 ***Table 76*** summarizes which CCCs are not permitted in any HDR Mode, with the reason for each CCC.

4109 **Table 76 CCCs Not Permitted in HDR Modes**

CCC Type		Command Name	Reason
Broadcast	**Direct**		
X	X	**RSTDAA**	Target Dynamic Address changes are a fundamental configuration change, which should be done in SDR Mode only.
X	–	**ENTDAA**	Dynamic Address Assignment can only be performed in SDR Mode.
X	–	**ENTTM**	Test Mode can only be performed in SDR Mode. Test Mode is generally only used for manufacturing or Device test.
X	–	**ENTHDR0** through **ENTHDR7**	HDR Modes can only be entered from SDR Mode. There is no provision for switching directly to a different HDR Mode, it is necessary to exit to SDR Mode first.
X	–	**SETAASA**	Dynamic Address Assignment can be done in SDR Mode only. In addition, SETAASA (if supported) is typically used only once per Bus, during Bus Initialization which always takes place in SDR Mode.
–	X	**SETDASA**	Dynamic Address Assignment can be done in SDR Mode only. In addition, SETDASA (if supported) is typically used only once per Target, during Bus Initialization which always takes place in SDR Mode.
–	X	**SETNEWDA**	Target Dynamic Address changes are a fundamental configuration change, which should be done in SDR Mode only.
–	X	**GETPID**	Usually only associated with the **ENTDAA** CCC, or when a Secondary Controller that was not present during Bus Initialization becomes Active Controller (pursuant to a GETACCCR CCC) and requests information about a Target Device.
–	X	**GETACCCR**	The Controller Handoff procedure is only supported in SDR Mode.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.1.3.1 Elements, Framing and Modality

Within each HDR Mode, the framing for CCC flows differs from other HDR read/write transactions, providing a modality similar to the way CCCs are framed in SDR Mode.

This Specification defines a number of common Flow Elements (such as Word and Blocks) that together provide a generic framing model that can be used to describe the CCC flows for each individual HDR Mode.

The following Flow Elements are defined, where 'x' represents an HDR Mode:

- **HDR-x CCC Header Block**

  This Block has two variants:

  - Type Indicator
  - Type Selector

- **HDR-x Header Block**

  This Block signals the end of a CCC Frame for the particular HDR Mode. It is neither an Indicator nor a Selector.

- **HDR-x CCC Data Block**

  This Block has three variants:

  - When sent by the Controller as part of a Broadcast CCC flow
  - When sent by the Controller as part of a Direct Write or Direct SET CCC flow (i.e., in a Write Segment)
  - When sent by an addressed Target as part of a Direct Read or Direct GET CCC flow (i.e., in a Read Segment)

- **HDR-x CCC Termination Marker**

  This marker is sent only as needed, i.e., is optional per the particular HDR Mode and per the position in a type of CCC flow.

All of the above Flow Elements are common CCC Flow Elements, except the Write Segment and Read Segment variants of HDR-x CCC Data Block, and specific variants of the termination marker (optional). All Target Devices that support CCCs in an HDR Mode must be able to successfully receive and understand these common CCC Flow Elements, in order to track their place within the CCC flows, to know when to enter the modality (and to know when it has been ended), and to understand when they are being specifically addressed in a Direct CCC flow.

The beginning of a CCC in an HDR Mode is indicated by use of an HDR-x CCC Header Block of type Indicator. This is an HDR-x Header Block containing a special byte value (i.e., the Broadcast Address, 7'h7E) instead of a Target Address. This is different from the beginning of other HDR transactions.

The format of the Indicator HDR-x CCC Header Block is specified separately for each HDR Mode, but generally includes the Indicator, the Command Code, and the optional Defining Byte. If the HDR-x CCC Header Block includes a bitfield for the Defining Byte that is always present, then for CCCs that support an optional Defining Byte, a Defining Byte value of 8'h0 shall be equivalent to omitting the Defining Byte. This shall have the same effect as omitting the optional Defining Byte in SDR Mode for the same CCC.

For Broadcast CCCs, the Controller shall send the HDR-x CCC Header Block of type 'Indicator', containing the Command Code and optional Defining Byte, followed by one or more HDR-x CCC Data Blocks, containing the data bytes to be Broadcast to all supported Target Devices.

For Direct CCCs, the Controller shall send the HDR-x CCC Header Block of type 'Indicator', containing the Command Code and optional Defining Byte. The Controller shall then send the HDR Restart Pattern, and then start a new Segment by sending another HDR-x CCC Header Block, but of Selector type: it includes a Target Address and indicates whether the Direct CCC Segment is a Write Segment vs. a Read Segment.

- For a Direct Write or Direct SET CCC with data, the Controller generally sends one or more HDR-x CCC Data Blocks if the selected CCC's defined format so indicates. However, if the CCC's defined format includes no data to be sent to the Target (i.e., if a zero-byte 'Write' to the Target for

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

a given Command Code and optional Defining Byte is sufficient to instruct the Target to take some action), then the Controller would send the minimum number of HDR-x CCC Data Blocks for the HDR Mode's framing. In some cases, this might be zero Data Blocks, if the HDR Mode permits.

- If the Direct Write or Direct SET CCC requires a Sub-Command Byte, then the Controller shall send this Sub-Command Byte as the first byte in the Write Segment's data message, which is the first HDR-x CCC Data Block.

- For a Direct Read or Direct GET CCC, the Controller shall allow the addressed Target to control the Bus, and the Target may send one or more HDR-x CCC Data Blocks if the selected CCC's defined format so indicates. After the end of the Segment, the Controller shall resume control.

For either Broadcast or Direct CCCs, the Controller and the Targets shall observe all HDR Mode specific framing requirements:

- For zero-byte CCCs or very short CCCs, if a minimum number of HDR-x CCC Data Blocks must be sent for the HDR Mode's framing, then the sender shall provide dummy byte values of 0x00 in these Data Blocks, and the receiver shall correctly ignore these values, per the specific CCC definition.
- If the length of a CCC's data message is not evenly aligned to the size of the HDR-x CCC Data Block, then the sender shall pad the end of the data message with dummy byte values of 0x00, and the receiver shall correctly ignore any dummy byte values, per the specific CCC definition.

To end either a Broadcast CCC or a Direct CCC, the Controller shall perform any HDR-x CCC End Procedure that is valid for that HDR Mode.

As part of the data message that is either sent by the Controller (i.e., as part of a Broadcast CCC or a Write Segment for a Direct CCC), or sent by a Target (i.e., as part of a Read Segment for a Direct CCC), the sender shall denote the end of the data message by including an appropriately-constructed HDR-x CCC Termination Marker in a format specific for the particular HDR Mode and transaction type. The Controller shall also include a Termination Marker, at the end of the Selector type HDR-x CCC Header Block that begins the framing for a Direct CCC, before starting any Read or Write Segments. In both cases, this Termination Marker is required to allow the receiver to properly understand the end of the header block or data message and take the appropriate action to fit within the signaling for that HDR Mode.

Because a Termination Marker can also serve as the transition before the next common CCC Flow Element, it may include signaling acknowledging the preceding Flow Element, or signaling that includes Bus Turnaround or other forms of Handoff between the Controller and an addressed Target.

Depending on the specific HDR Mode and transaction type, this Termination Marker will take one of three possible forms:

1. **CRC Block**

   This Block includes valid CRC (checksum) data for the preceding HDR-x CCC Data Block(s) or HDR-x CCC Header Block. The receiver shall use the CRC data to verify that the data received from the sender is valid.

   - For HDR-DDR Mode: The CRC Block is the HDR-DDR CRC Word (see ***Section 6.2.3.3.1.3***).
   - For HDR-BT Mode: The CRC Block is the HDR-BT CRC Block (see ***Section 6.4.3.3.3***).

2. **Turnaround or Handoff Pattern**

   This form of Termination Marker is a specific pattern indicating Bus Turnaround or Handoff back to the Controller, if so indicated by the direction of transfer.

3. **No Termination Marker**

   Under particular conditions, no Termination Marker *per se* will be sent.

After this optional Termination Marker, the Controller shall reassert its control of the Bus (if necessary) and then send (or complete) the HDR Restart Pattern, or else continue on to perform any valid HDR-x CCC End Procedure that is appropriate for that framing.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Because the Termination Marker might take different forms, or might not exist on the I3C Bus at all depending on the particular HDR Mode or type of transaction, it is indicated generically in the following flow diagrams, using the label ‘TM’.

At several points in the CCC flows it might be necessary for the Target to acknowledge receipt of a given Flow Element. In the following flow diagrams, such an acknowledgement may also serve as a Termination Marker; the label ‘ACK’ indicates such cases.

Note:

The exact location of an acknowledgement may vary, depending on the particular HDR Mode and the flow control method used to indicate acknowledgement.

The framing for Direct CCCs also allows for multiple Target Addresses to be addressed, in a manner similar to the SDR Direct CCC framing model which addresses multiple Target Addresses as combinations of Read Segments (for Direct Read or Direct GET CCCs) and/or Write Segments (for Direct Write or Direct SET CCCs). The Controller may use this framing to address one or more Target Addresses by using multiple Segments, separated by the HDR Restart Pattern. When used in this manner the HDR Restart Pattern acts not as the end of the CCC (as it does with other, HDR traffic), but as the end of one Segment addressed to the indicated Target Address.

Note:

*In this framing the HDR Restart Pattern serves as an equivalent of SDR Mode’s Repeated START in Direct CCC framing (see **Section 4.3.7.2.2**).*

Target Devices shall store the Command Code and its optional Defining Byte (as the Controller sent in the Indicator type HDR-x CCC Header Block) and act on them as individually addressed in this framing. The Command Code and optional Defining Byte values shall be stored for later use, if a Target matches its Target Address in a subsequent Selector type HDR-x CCC Header Block, until the end of the current modality is indicated via any HDR-x CCC End Procedure valid for the particular HDR Mode.

The generic Broadcast CCC flow for the framings described above is shown in ***Figure 99***; the generic Direct CCC flow for the framings described above is shown in ***Figure 100***.

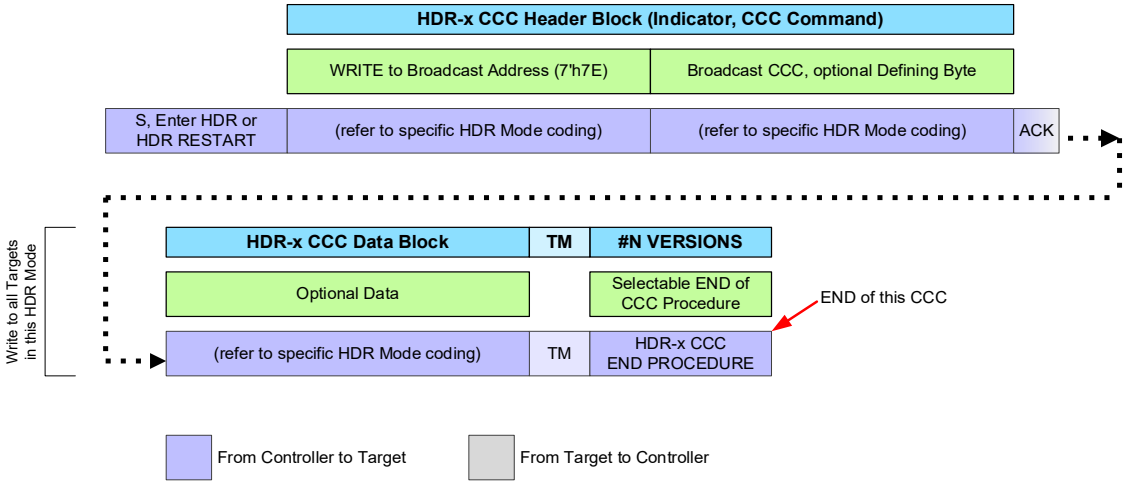


Figure 99 Begin Broadcast CCC Per HDR Mode

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

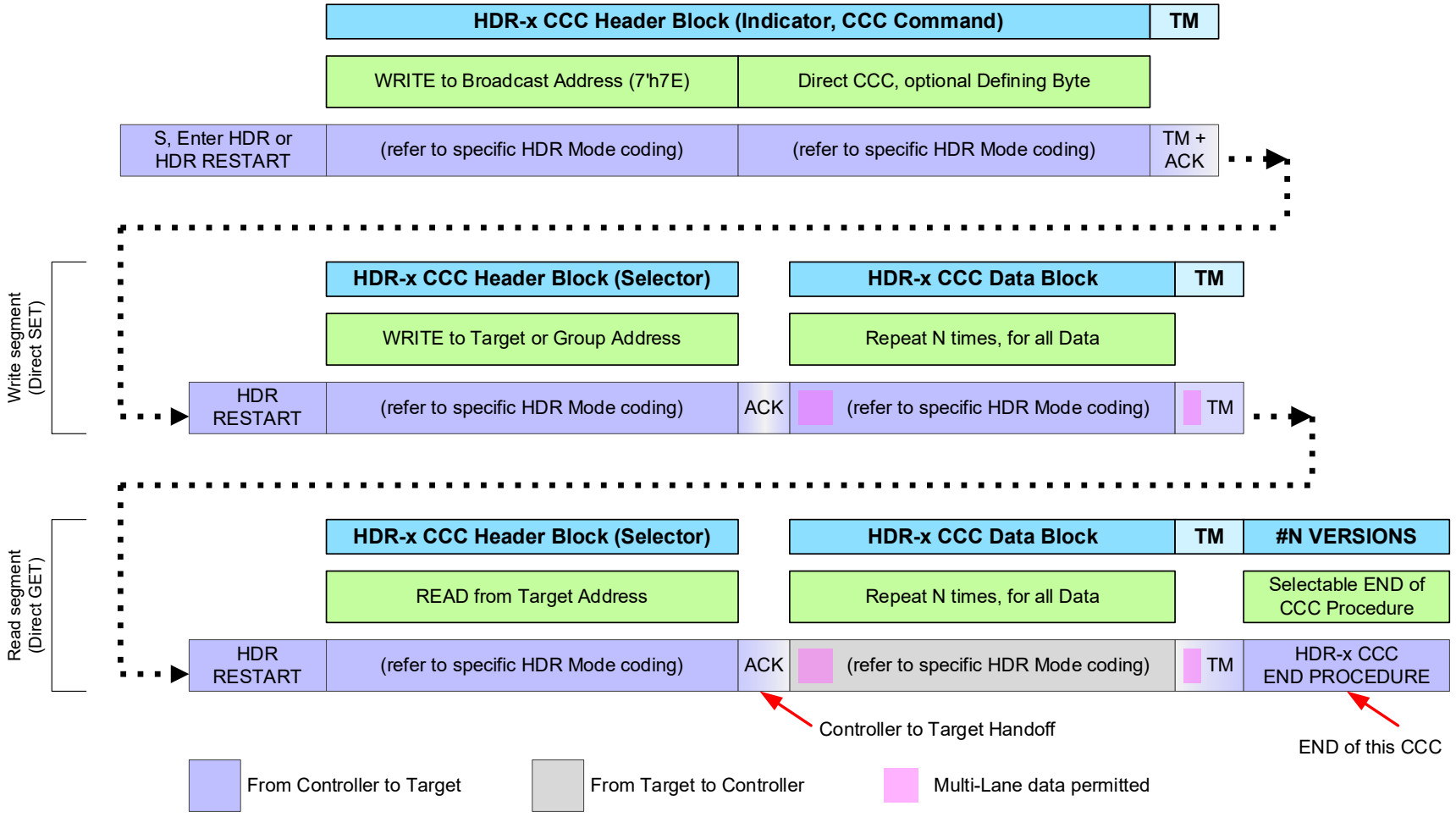


Figure 100 Begin Direct CCC Per HDR Mode

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

The ending of the CCC modality shall be indicated by any HDR-x CCC End Procedure valid for the particular HDR Mode. The HDR Restart Pattern alone shall not indicate the end of the CCC flow, as it would end a normal HDR read/write command flow. All Target Devices that support CCCs in the particular HDR Mode shall detect the beginning of the CCC and interpret the HDR Restart Pattern appropriately for the CCC framing until they receive an HDR-x CCC End Procedure valid for that HDR Mode. Each HDR Mode may define several HDR-x CCC End Procedures, depending on the HDR Mode's use cases and Frame formats.

Upon receiving a valid HDR-x CCC End Procedure, the Target shall either end the CCC modality and expect normal HDR traffic, or else immediately re-enter the CCC modality (depending on the type of procedure) by capturing a new Command Code and optional Defining Byte. Typically, a valid HDR-x CCC End Procedure starts with a common Flow Element, such as an Indicator type HDR-x CCC Header Block (or equivalent) to start a new CCC, or an HDR-x Header Block to end the CCC Frame. Such a Block shall be addressed to the Broadcast Address (7'h7E) and shall also indicate whether the modality is ended vs. re-entered for a new CCC. In this manner the Controller can use normal HDR traffic, Broadcast CCCs, and Direct CCCs in a continuous flow within a given HDR Mode, arranged in any sequential order, without needing to exit the HDR Mode to separate these commands or traffic flows.

Note:

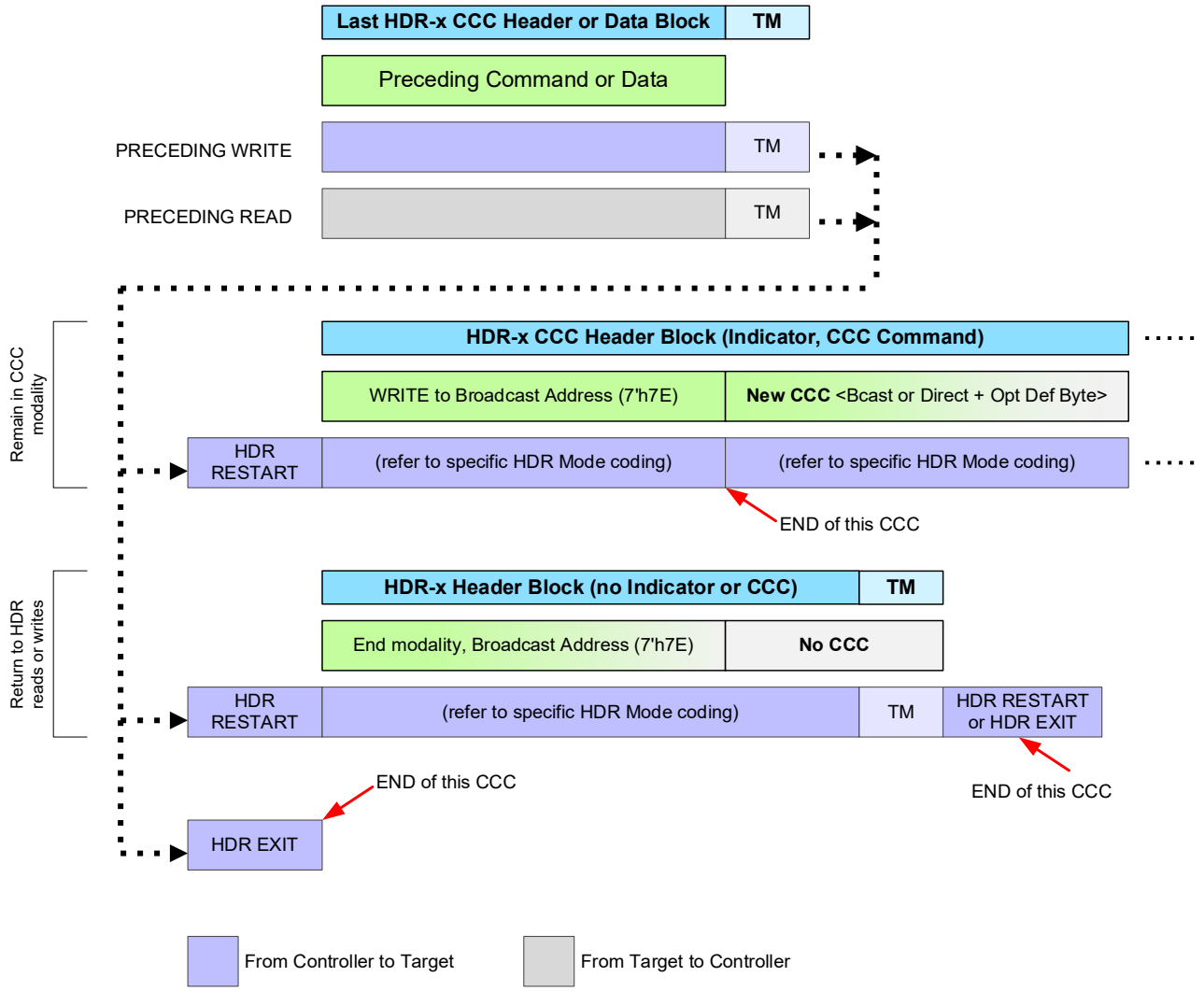
*A valid HDR-x CCC End Procedure for a particular HDR Mode that indicates the end of the CCC modality might use the dummy command code 0x1F (see **Table 16**) instead of a valid CCC, if the framing for that HDR Mode requires the Controller to use an additional Flow Element that would otherwise contain the CCC in order to comply with valid transfers in that HDR Mode. This dummy command code 0x1F is defined for this specific purpose. When using the dummy command code 0x1F, a dummy Defining Byte value 0x00 should also be used, if the additional Flow Element also has a field that would otherwise contain a valid Defining Byte.*

The following HDR-x CCC End Procedures are recommended, but this is not an exhaustive list of End Procedures for any given HDR Mode:

- HDR Restart Pattern followed by Indicator type HDR-x CCC Header Block (or equivalent), indicating that the CCC modality is still active, but providing a new CCC and optional Defining Byte. Flow then proceeds as a new Broadcast or Direct CCC (see ***Figure 100*** and ***Figure 101***).
- HDR Restart Pattern followed by HDR-x Header Block (or equivalent) indicating that the CCC modality is ended or no longer in 'continuation', followed by an optional Termination Marker to indicate the end of a transfer, followed by another HDR Restart Pattern (see ***Figure 101***). Flow then proceeds as standard HDR read/write transactions.
  - Any End Procedure that indicates that the CCC modality is ended must be distinct from the previous End Procedure, in order that all Target Devices may clearly recognize it and interpret subsequent HDR transactions as standard HDR read/write transactions. Typically, this requires its first Flow Element (i.e., the HDR-x Header Block) to be clearly recognized as neither an Indicator nor a Selector.
- An optional variant of the above, which ends with the HDR Exit Pattern instead of the HDR Restart Pattern. HDR Mode is then exited, and the Bus returns to SDR Mode.
- HDR Exit Pattern

The recommended HDR-x CCC End Procedures are shown in ***Figure 101***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



4272

Figure 101 End of CCC Procedures Per HDR Mode

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

In the figures above, the labels HDR-x CCC Header Block, HDR-x CCC Data Block, and Termination Marker (if supported) represent generic, higher-level elements of the flow for sending CCCs in all of the HDR Modes. In actual practice the specific sizes of the Blocks, their bitfield formats, and data encodings on the Bus vary depending on the particular HDR Mode being used. ***Table 77*** shows, for each HDR Mode that supports CCCs, where to find the CCC coding details for each Flow Element when using that HDR Mode.

Table 77 HDR CCC Details Per HDR Mode

HDR Mode	See Section	CCC Coding Details Section	Begin Broadcast CCC	Begin Direct CCC	CCC End Procedures
HDR-DDR	***6.2***	***6.2.3.3.1***	***Figure 111***	***Figure 112***	***Figure 113***
HDR-BT	***6.4***	***6.4.3.4***	***Figure 134***	***Figure 135***	***Figure 136***

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.1.3.2 Broadcast Address ACK

When addressing all Targets using Broadcast CCCs, or when starting the Direct CCC flow, the Controller shall send an Indicator type HDR-x CCC Header Block to signal that it is trying to write to all Targets. This Indicator block shall include the Broadcast Address (7'h7E) and will generally take the form of a Write command for that HDR Mode.

Each Target Device that supports CCC flows in that HDR Mode and receives an Indicator block shall use a valid HDR Flow Control method for that HDR Mode to acknowledge the Indicator block. This acknowledgement indicates that it has received the Indicator block, has acknowledged the beginning of the Broadcast CCC or Direct CCC (i.e., has entered the modality), and is ready to participate. The Target shall acknowledge even if it decides not to act on (or even if it does not support) the particular CCC or its optional Defining Byte.

Note:

Acknowledging the Indicator block is equivalent to SDR Mode CCC Framing, where all Targets must acknowledge the Broadcast Address at the beginning of a CCC.

For Direct CCC flows, acknowledgement of the Indicator block serves the important function of ensuring that the Target Device agrees to capture the Command Code and optional Defining Byte, and is ready to be addressed by any subsequent Selector blocks in this Direct CCC flow. By acknowledging the Indicator block, the Target signals that will capture these values into its internal state and use them to decide whether to respond to that particular Command Code and optional Defining Byte, if it matches its Target Address in a subsequent Selector block that begins a Read or Write Segment. In this case, the Target is considered to be 'primed' with the captured values, and capable of making a rapid ACK/NACK decision at the correct point in the flow.

If one such Target on the Bus is unable to respond and does not acknowledge the Indicator block, then this is equivalent to a NACK of the Broadcast Address in the Indicator block, for that HDR Mode. However, if other Targets on the Bus are present and are able to acknowledge the Indicator block, then one Target's failure to acknowledge the Indicator block must not block the CCC flow from continuing. This typically requires the HDR Mode to define a common flow control method that can be used by multiple Targets (e.g., passive NACK, active ACK) for a Write command. If this flow control method is not enabled by default, then the Controller must enable it for the Bus, as appropriate for the HDR Mode, before sending any CCCs in that HDR Mode.

The Controller shall use this acknowledgement to ensure that the CCC flow is being received by at least one Target in that HDR Mode. If the Controller does not receive acknowledgement from at least one Target, then it should initiate an error recovery procedure, which might include exiting HDR Mode and attempting to re-send the CCC in SDR Mode.

Note:

If a Bus has multiple Target Devices that support CCC flows in that HDR Mode, then they shall all acknowledge the Indicator block together.

In ***Figure 99*** and ***Figure 100***, the location for acknowledgement is shown after the Indicator type HDR-x CCC Header Block. The specific location and method for the Target to indicate its acknowledgment of the Indicator block is specific to the particular HDR Mode, but generally includes some active state change that is driven by the Target, and that can be measured by the Controller.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.1.3.3 Direct CCC ACK/NACK and Retry

When addressing specific Targets using Direct CCCs in HDR Modes, the Controller shall indicate whether it is trying to read from the Target (i.e., a Direct Read or Direct GET) or write to the Target (i.e., a Direct Write or Direct SET). The Target Device may use a valid HDR Flow Control method to indicate that it cannot respond to a read request, or decline a write request at any point, in a manner equivalent to SDR Mode CCC Framing where it would NACK its own Target Address for a Direct CCC (see ***Section 4.3.7.2.2***).

In ***Figure 100***, the acknowledgement is shown after the Selector type HDR-x CCC Header Block for a Read Segment and a Write Segment. The specific location and method for the Target to indicate its part in the acknowledgment of the Selector block and agreement to respond to the Direct CCC with optional Defining Byte is specific to the particular HDR Mode, but generally includes some active state change that is driven by the Target, and that can be measured by the Controller.

Note that ***Figure 100*** illustrates an example HDR Direct SET/GET CCC. However, this flow can also be applied to a Direct SET or a Direct GET, as the Read Segments and Write Segments are optional and can appear in any order as defined for the particular CCC. Some CCCs might not have any Write or Read Segments, and in some cases the HDR-x CCC Data Block might be omitted if the CCC doesn't define any data to write to or read from a Target. Multiple Segments may address the same Target, or different Targets.

If a Target is not able to respond to a Direct GET CCC immediately on the first attempt by the Controller, then it may use a method specific to the chosen HDR Mode to decline to respond, similar to a NACK of its own Target Address in SDR Mode. The Controller shall follow the same Direct GET CCC Retry model used in SDR Mode (see ***Section 4.3.7.2.3***), and the Target should generally respond on the second attempt.

A Target indicating a delayed response on the first attempt is not the same as a Target choosing to NACK its own Target Address to indicate that it does not support a given CCC and Defining Byte. As with SDR Mode, in this case the Controller would make a second attempt, and the Target would also decline to respond on the second attempt.

However, if a Target declines to respond to another type of Direct CCC (i.e., not a Direct GET CCC) on the first attempt, then it might not support the given CCC and Defining Byte, in which case the Retry model would not necessarily apply.

The Controller may also address Groups using Direct Write or Direct SET CCCs, if supported by the particular Direct CCC; other special conditions shall apply, per ***Section 6.1.3.5***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.1.3.4 Implications for Device and Bus Configuration

While it is not always recommended to use CCCs in HDR Modes that might change Bus configuration or Device configuration, this might be permitted if the Active Controller follows certain safety recommendations, and if the Target Devices obey a set of conventions per their optional support for these CCCs in HDR Modes.

- For Direct SET CCCs that change a Device's configuration, a Target receiving that CCC shall verify such configuration changes after successfully receiving all data sent by the Controller (as indicated by a Termination Marker or other command coding elements specific to that HDR Mode) and then apply the configuration change immediately, at the end of the Frame for that Target.
- For Broadcast CCCs that change Bus Configuration or Device Configuration for all valid Devices, all Targets receiving that CCC shall verify such configuration changes after successfully receiving all data sent by the Controller (as indicated by a Termination Marker or other command coding elements specific to that HDR Mode) and then apply the configuration change immediately.

It should also be clear that any Broadcast CCCs that announce updated Bus configuration data (e.g., DEFTGTS and DEFGRPA) while in HDR Mode will not be received by Target Devices that do not support that particular HDR Mode. In most circumstances, a Controller-capable Device should generally expect to use these CCCs in SDR Mode, unless it knows with certainty that all other Devices that need to receive this Bus configuration data (i.e., all Secondary Controllers) also support the same HDR Mode and support receiving these Broadcast CCCs in that HDR Mode. It is the Active Controller's responsibility to ensure that all Secondary Controllers support the same HDR Mode before using these Broadcast CCCs in HDR Mode.

Since custom CCCs can be defined by various MIPI Alliance WGs or Vendors for special purposes, some of these CCCs may also change additional aspects of Bus configuration or Device configuration, in order to enable as-yet-undefined use cases or features not yet covered by this Specification.

Note:

Groups that create their own CCCs (including MIPI Alliance WGs, outside standards groups, and Vendors) may choose to define custom CCCs in any supported HDR Mode, and so should be aware of the recommendations above when choosing whether to do so, and how to apply any configuration changes based on data received from the Active Controller in HDR Mode.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.1.3.5 Implications for Group Addressing

For I3C Target Devices that support Group Addresses, the Controller may choose to configure a Group Address and address multiple Targets in a Direct Write or Direct SET CCC form of any supported CCC within the HDR CCC flows, in a manner similar to Group Address support in Direct CCC flows within SDR Mode. In this manner, Group Addresses may be used in place of Dynamic (Target) Addresses in the common CCC Flow Elements (i.e., the Selector type HDR-x CCC Header Blocks for Direct CCC Write Segments), as long as the following conditions are met:

- The Selector Block must indicate a Direct CCC Write Segment, since per ***Section 4.3.4.4*** a Group Address shall only be used for Write transfers in HDR Modes.
- All Targets that have been assigned to that Group Address using the SETGRPA CCC (see ***Section 4.3.7.3.24***) must support CCC flows within that HDR Mode.
- All Targets in that Group shall ACK the Group Address when addressed by the Selector block for the Write Segment.
- If any Target in that Group is not able to respond to the Write Segment directed to the Group Address, then its failure to ACK the Group Address must not block the Write Segment from continuing, if the Controller detects that at least one other Target in that Group responds with an ACK.
  - This typically requires the HDR Mode to define a common flow control method that must be enabled by the Controller before sending any CCCs to a Group Address (see ***Section 6.1.3.2*** and ***Section 6.1.3.3***).

For Bus configurations where Group Addressing is used in conjunction with Multi-Lane, the Controller shall ensure that all Targets that have been assigned to the same Group Address are configured to use mutually interoperable Data Transfer Codings, and have been configured to use the same ML Frame format (i.e., the same Data Transfer Coding and number of additional Lanes) for transfers addressed to that Group Address. The Controller must verify this before attempting to send a Direct Write or Direct SET CCC to a Group Address in a Bus configured for Multi-Lane with HDR CCC flows.

- If not all Targets in a Group are configured to use the same ML Frame format in a particular HDR Mode, then the Controller must configure such Targets to use a common ML Frame format that they all support. In some cases, this might mean using a 1-Lane compatible ML Frame format to send a Direct Write or Direct SET CCC to a Group Address.

If a Target Device supports multiple current ML configurations for its assigned Group Addresses (per ***Section 6.7.3.1.1***), then:

- Within the CCC flows in HDR Modes, such a Target shall use the Address in the Selector Block to determine which ML configuration to use, as it prepares to send or receive data in that Direct CCC Segment. If the Address matches a currently assigned Group Address, then it shall use the correct ML configuration for that Group Address instead of the ML configuration for its Dynamic Address.
- After assigning such a Target Device to any Group Address, that Target shall have an independent ML configuration associated with that Group Address, and the Controller might need to re-configure the Target's ML configuration for CCCs directed to that Group Address (as a specific example of ML configuration requirements listed in ***Section 6.7.3.1.1***).
  - Such configuration changes shall not affect the ML configuration for the Target's Dynamic Address(es), and the Controller may continue sending Direct CCCs to a Target's Dynamic Address(es) using the previously configured ML Frame format for that Dynamic Address.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

If a Target Device does not support multiple current ML configurations for its assigned Group Addresses (per ***Section 6.7.3.1.1***), then:

- Within the CCC flows in HDR Modes, such a Target shall use the current ML configuration for its Dynamic Address (i.e., its single ML configuration for that HDR Mode) as it prepares to send or receive data in that Direct CCC Segment. If the Address in the Selector Block matches any currently assigned address (i.e., Dynamic Address, Broadcast Address or any Group Address), then it shall use its single ML configuration.
- After assigning that Target Device to any Group Address, the Controller might need to re-configure the ML Frame format for that Target, or for the other Targets in any other Group Addresses to which that Target Device is assigned, in order to ensure that all Target Devices that have been assigned to any particular Group Address are still configured to use the same ML Frame format after the change, since such a Target Device does not support multiple current ML configurations.

Note:

*The Target shall use the ML configuration (i.e., the currently configured ML Frame format) for its Dynamic Address in order to properly receive all common CCC Flow Elements as defined in **Section 6.1.3.1**). This includes any HDR-x transfers that are addressed to the Broadcast Address, which generally include the Indicator type HDR-x CCC Header Block as well as various HDR-x CCC End Procedures that are defined for that HDR Mode. Such common CCC Flow Elements must be sent by the Controller in a form that all Target Devices supporting that HDR Mode are configured to understand and properly receive, in a mutually interoperable manner; and each such Target Device shall respond according to such configuration, according to the current ML Frame format for that HDR Mode.*

*The requirements in this Section are a specific application of other normative requirements in this Specification (e.g., **Section 4.3.4.4**, **Section 6.7.3.6**, and **Section 6.7.3.1.1**). Such other sections also include other requirements, restrictions, and recommendations for Target Devices, as well as the Active Controller of the Bus. As the requirements in this Section are intended for a specific use case of HDR transfers as applied to CCC flows in HDR Modes, readers are warned not to interpret this Section as contradictory to any such requirements or restrictions in such other sections that govern all Multi-Lane configuration and all HDR transfers (including HDR transfers sent to Group Addresses).*

6.1.3.6 Bus Efficiency

It should be expected that remaining within a given HDR Mode is more efficient for specific, traffic-intensive use cases, and that using CCCs while in HDR Mode will help maintain this efficiency by avoiding the need for the Controller to exit HDR Mode in order to send a CCC (as with SDR Mode). It is also likely that CCC data can be sent more quickly, by taking advantage of the increased data rates of the HDR Modes and Multi-Lane functionality (if supported and configured).

However, depending on the Bus Configuration and its application and use cases, the Controller might occasionally need to exit HDR Mode, in order to provide a window for Target Devices to request In-Band Interrupts, the Controller Role, or Hot-Joins as may be required per the Bus Configuration and its application and use cases.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.1.4 Transition to HDR Mode Transfer

The Controller is responsible for starting the HDR Mode transfer, which consists of sending the first structured protocol element, per the HDR Mode framing. The Controller shall do this after entering the HDR Mode for the first transfer in that HDR Mode (see ***Section 6.1.4.1***), or after the HDR Restart Pattern, for subsequent transfers in that same HDR Mode (see ***Section 6.1.4.2***).

In both cases, the specific timing of when the Controller drives SDA in relation to SCL depends on the HDR Mode signaling. In this Section and its sub-sections, a possible example signaling method is provided:

- For HDR Modes that use SCL as an edge-triggering clock (i.e., like SDR Mode), the first bit starts on the next SCL rising edge, and the Controller shall drive SDA when SCL is Low (i.e., before the rising edge). This ensures that the Target samples SDA on the SCL rising edge.

Note:

The signaling method described in this Section is intended to be used as a template for all of the HDR Modes defined in the I3C basic Specification. Since other signaling methods are possible, this Section does not provide an exhaustive list of possible signaling methods that might be defined for future HDR Modes.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.1.4.1 Entering the HDR Mode After ENTHDRx CCC

To enter an HDR Mode, the Controller sends the Broadcast CCC for a particular HDR Mode, and then drives SDA after the completion of the T-Bit (i.e., on or after the falling edge of the SCL pulse for C9) to begin the first structured protocol element in that HDR Mode, according to the HDR Mode framing.

From that point, the Controller follows the common timing rules for starting a transfer in that HDR Mode per ***Section 6.1.4***. Note that the I3C Bus transitions from SDR Mode to that HDR Mode after the falling edge of the SCL pulse for C9.

Note:

*The Target is also required to enable its HDR Exit Pattern Detector after the Broadcast ENTHDRx CCC; see **Section 4.3.10.2.1**.*

Figure 102 illustrates the signaling for entering a generic HDR Mode (shown as "HDR-*x*") after the ENTHDRx CCC. In this example, this HDR Mode uses SCL as a dedicated, edge-triggering clock, so the Controller sets up SDA before the next SCL rising edge, to start sending the first structured protocol element (i.e., a Command or Header) for the first HDR Mode transfer.

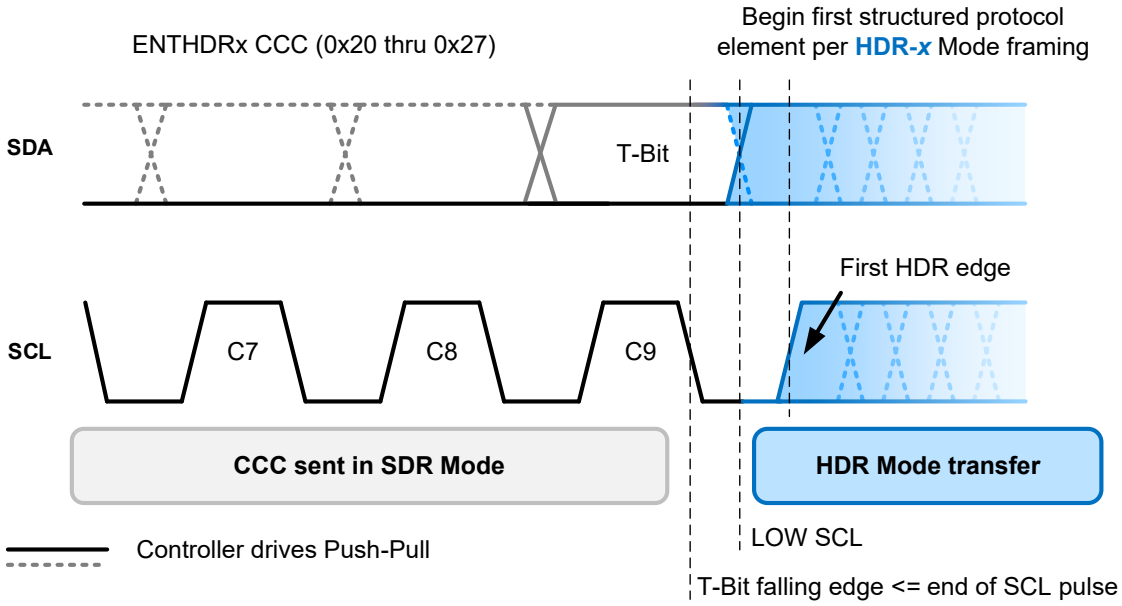


Figure 102 Entering HDR Mode After ENTHDRx CCC (Dedicated Edge Clock)

Note:

*In **Figure 102** the I3C Bus enters the HDR Mode after the T-Bit, since the ENTHDRx Broadcast CCCs do not support Defining Bytes.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.1.4.2 Next HDR Mode Transfer After HDR Restart Pattern

To remain in an HDR Mode after the completion of an HDR Mode transfer that ends with the HDR Restart Pattern (see ***Figure 97***), the Controller completes the SCL pulse by driving SCL Low, and then drives SDA to begin the first structured protocol element in that HDR Mode, according to the HDR Mode framing.

From that point, the Controller follows the common timing rules for starting a transfer in that HDR Mode per ***Section 6.1.4***. Note that the I3C Bus remains in the same HDR Mode through the HDR Restart Pattern.

Figure 103 illustrates the signaling for an HDR Restart Pattern followed by starting the next HDR Mode transfer, in a generic HDR Mode (shown as "HDR-*x*") that uses SCL as a dedicated, edge-triggering clock. This diagram might be a continuation of the signaling shown in ***Figure 102***: if so, then the I3C Bus remains in the same HDR Mode through the HDR Restart Pattern and the next HDR Mode transfer. After the completion of the SCL pulse that validates the HDR Restart Pattern, the Controller sets up SDA before the next SCL rising edge, to start sending the first structured protocol element for the next HDR Mode transfer (similar to ***Figure 102***).

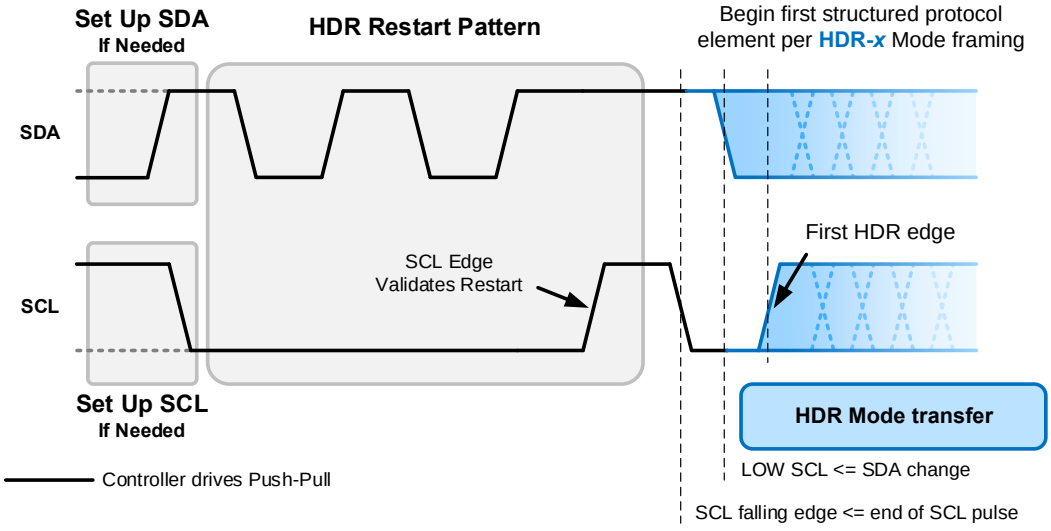


Figure 103 Next HDR Mode Transfer After HDR Restart Pattern (Dedicated Edge Clock)

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2 F001: HDR-DDR Mode

This Section defines I3C Optional Feature F001: HDR-DDR Mode.

Table 78 Optional Feature 001 Status

Optional Feature ID	F001
Optional Feature Name	HDR-DDR Mode
Optional Feature Version	1
Optional Feature specification was last updated in I3C Spec Version	I3C v1.2
Optional Feature is not compatible with other Optional Features	–
Optional Feature is included in I3C Basic	Yes

Changes in this Version

For detailed changes in this version, see the "diff" file between I3C v1.1.1 and I3C v1.2 in the MIPI Specification Repository ***[MIPI09]***.

6.2.1 Scope and Purpose (Informative)

The scope of ***Section 6.2*** is limited to fully specifying I3C Optional Feature F001: HDR-DDR Mode. An I3C Device that implements F001 is required to implement all **shall** statements in ***Section 6.2.3***, and can optionally implement any combination of **may** statements in ***Section 6.2.3***. In addition, MIPI Alliance recommends that manufacturers follow all **should** statements in ***Section 6.2.3***.

6.2.2 Technical Overview (Informative)

Like SDR Mode, HDR-DDR Mode uses SCL as a clock; however, unlike SDR Mode, Data and Commands change SDA on both SCL edges (when High and when Low), effectively doubling the data rate. By contrast, in SDR Mode SDA is changed only when SCL is Low. Since SDR Mode defines it as a START or a STOP for SDA to change while SCL remains High, HDR-DDR Mode is classified as an HDR Mode in order to prevent confusion.

HDR-DDR is an I3C HDR mode, which is entered in the standard way (see the CCC **ENTHDR0**, ***Section 4.3.7.3.9***) and exited via the HDR Exit Pattern. As with the other HDR modes, Messages while in the HDR-DDR mode are separated using the HDR Restart pattern.

Once in HDR-DDR Mode, the Controller issues a Command Word to start every transfer (unless NACKed). In HDR-DDR Mode, Commands shall be issued only by a Controller. Data Words may be issued by a Controller or by a Target, depending on the particular Command (i.e., Write or Read). At least one Data Word shall be issued, unless the Target does not accept (ACK) the Command per ***Section 6.2.3.4***.

Figure 104 illustrates a typical HDR-DDR Mode Frame with two HDR Commands and their associated data:

- I3C START or Repeated START
- I3C CCC to enter HDR-DDR Mode
- After entering HDR-DDR Mode, there is one HDR-DDR Command Word, followed by one or more HDR-DDR Data Words, and then an HDR-DDR CRC Word
- Then an HDR Restart Pattern
- Then another HDR-DDR Command Word, HDR-DDR Data Word, and HDR-DDR CRC Word
- Finally, HDR Mode is ended via the HDR Exit Pattern followed by I3C STOP

Note:

An HDR-DDR CRC Word might also be emitted after an abort, per **Section *6.2.3.4***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

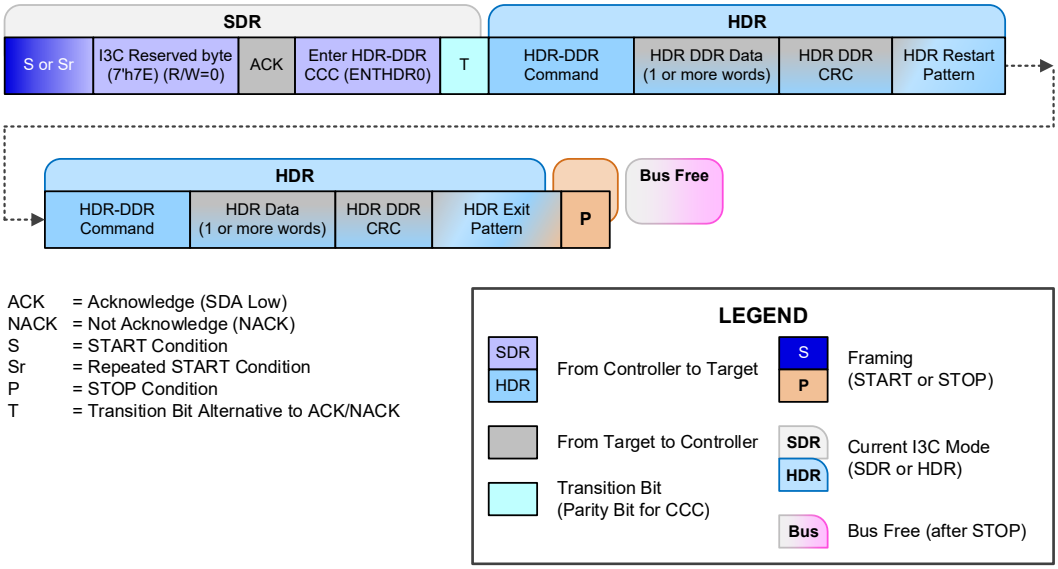


Figure 104 Typical HDR-DDR Mode Frame

Following the HDR-DDR CRC Word, the Controller provides either the HDR Restart Pattern or the HDR Exit Pattern.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3 Technical Specification

6.2.3.1 HDR-DDR Signaling

The HDR-DDR Protocol uses the same signaling as SDR, but operates with SDA changing after each SCL edge. In HDR-DDR Mode, just as in SDR Mode, only the I3C Bus Controller drives the SCL line. For Commands, the SDA line is driven by the Controller. For Data, the SDA line is driven either by the Target or by the Controller, depending on the Command direction. Bus Turnaround behavior is specified in ***Section 6.2.3.4***.

HDR-DDR Mode is entered using the ENTHDR0 CCC (see ***Section 4.3.7.3.9***). After the ENTHDR0 CCC and its ninth bit (the T-Bit), the SCL falling edge begins HDR-DDR Mode. The first HDR-DDR bit starts on the next SCL rising edge; that is, the Controller drives SDA when SCL is Low, and the Target samples SDA on the SCL rising edge. This allows the entry to HDR-DDR Mode, as shown in ***Figure 109***.

6.2.3.2 HDR-DDR Structured Protocol Elements

HDR-DDR moves data by Words. A Word generally contains 16 payload bits, as two bytes in a byte stream, and two parity bits. Four HDR-DDR Word Types are defined: Command Word, Data Word, CRC Word, and Reserved Word.

The HDR-DDR Protocol precedes each 18-bit Word with a 2-bit Preamble, for a total of 20 bits per Word.

The Preamble:

- Indicates the type of data that follows: either Command, Data, or CRC;
- Allows the Controller to terminate a Read, and to determine whether the Target is willing to respond to a Read;
- Allows the Target to NACK a Write before the first Data Word, if it has been configured to do so (see ***Section 6.2.3.4.4***); and
- Allows the Target to request a Write termination between Data Words, if it has been configured to do so (see ***Section 6.2.3.4.4***).

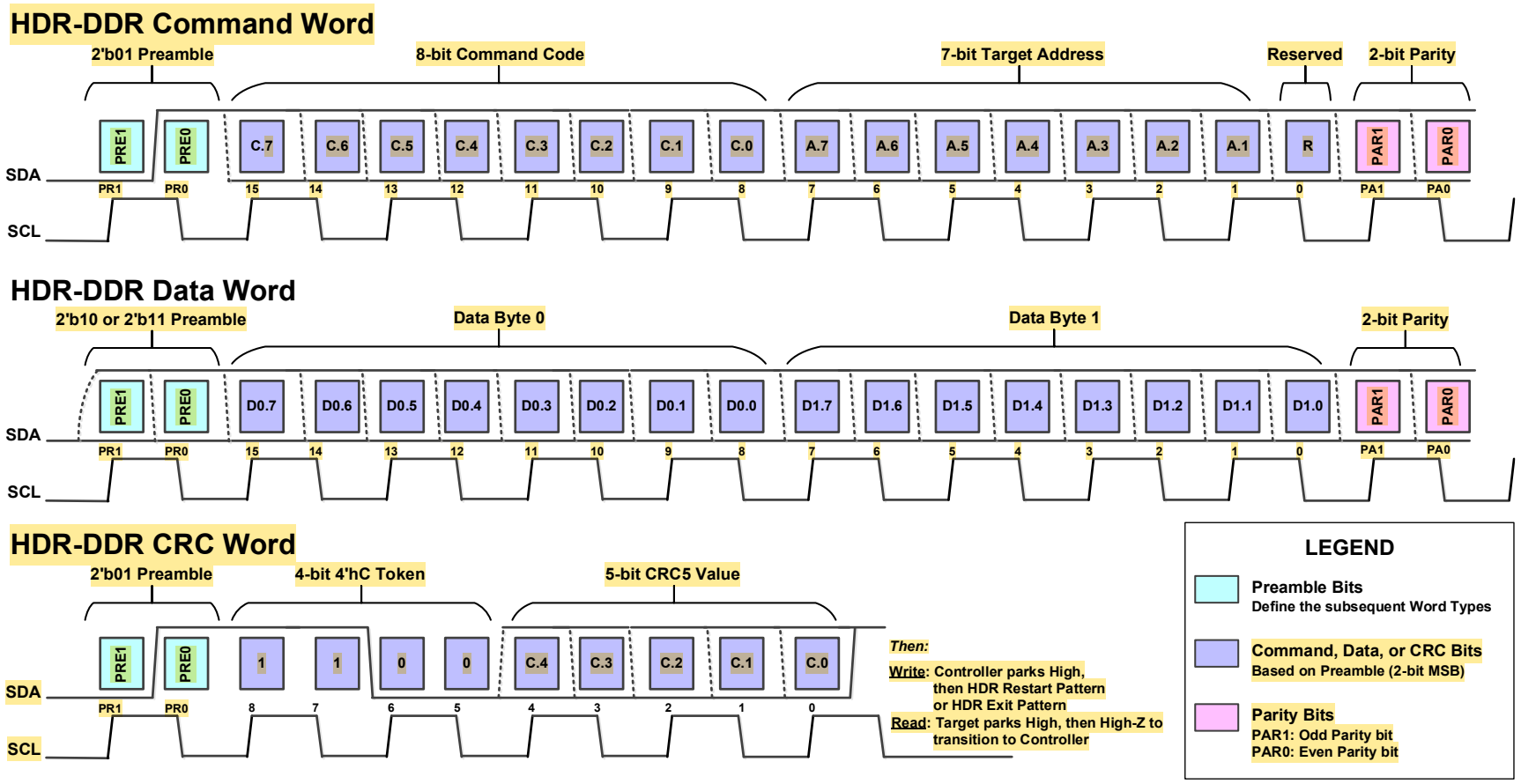
The related flow control procedures are described in ***Section 6.2.3.4***.

The main structuring of the HDR-DDR mode is broken into 3 top-level units:

Command Word (Address and Command Code)	**N * Data Words** Each word contains 2 bytes (for Single-Lane) or additional bytes (for Multi-Lane)	**CRC Word** (CRC-5)

Figure 105 shows the defined HDR-DDR structured protocol elements, where the Preamble bits determine the type of HDR-DDR Word.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



4558

Figure 105 HDR-DDR Word Formats

See Errata 01, Item 13

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Preamble bit interpretation depends on the context, as shown in ***Table 79***. The context controls the roles of Controller and Target, such that for Data, the Target or Controller drives the second Preamble bit after the other I3C Device has Parked High. The Preamble value of 2'b00 is reserved for special use, as indicated when such use is defined.

Table 79 HDR-DDR Preamble Values

Context	Preamble Value and Interpretation			
	2'b00	**2'b01**	**2'b10**	**2'b11**
After Enter HDR	Reserved for special use cases	Command Word follows	–	–
After Read CMD		–	Target ACK, Data follows	Target NACK, Aborted
After Read DATA		CRC Word follows	Controller Aborts, Target yields. *Controller drives second 0.*	Data follows. *Controller does not drive second bit.*
After Write CMD		–	Target ACK, Data follows	Target NACK, Aborted
After Write DATA		CRC Word follows	Target requests END Controller complies	Data follows

HDR-DDR Mode defines four Word Types (see ***Figure 105***):

- **Command Word:**
  - The Preamble before a Command Word shall have the value 2'b01.
  - The length of a Command Word shall be 18 bits (16 value bits, 2 parity bits).

    HDR-DDR Mode Command Codes allow up to 128 Write or Read Commands, respectively (see ***Section 6.2.3.3***).
- **Data Word:**
  - The Preamble before a Data Word shall contain the value 2'b10 or 2'b11 (see ***Table 79***).
  - The length of a normal Data Word shall be 18 bits (16 value bits as two bytes, 2 parity bits).
- **CRC Word:**
  - The Preamble before a CRC Word shall have the value 2'b01.
  - The length of a CRC Word shall be 10 clocked bits (4 bit 4'hC token, 5 bits of CRC-5 checksum, 1 bit of 1'b1 setup to prepare the Bus for the following HDR Exit Pattern or HDR Restart Pattern).
  - The value of the upper nibble (first 4 bits after the Preamble) of a CRC Word shall be 4'hC.
  - There is no parity checking for a CRC Word.
  - Following the falling edge of the last SCL pulse, the SDA shall be kept High (i.e., Open Drain class Pull-Up from Controller, while Target is in High-Z), for a time at least equal to t_{DIG_H}.
  - Following transmission of a CRC Word, an I3C Device shall set up SCL and SDA to allow for an HDR Restart Pattern or HDR Exit Pattern.

    The CRC-5 algorithm is specified in ***Section 6.2.3.6***.
- **Reserved Word (For special use only):**
  - The Preamble before a Reserved Word (i.e., not a CRC Word) shall have the value 2'b01.
  - The length of a Reserved Word shall be 18 bits (4-bit token 4'hD / 4'hE / 4'hF, 12 reserved bits, and 2 parity bits).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Note:

HDR-DDR Mode also allows for special use cases to be defined in the future, via Preamble value 2'b00.

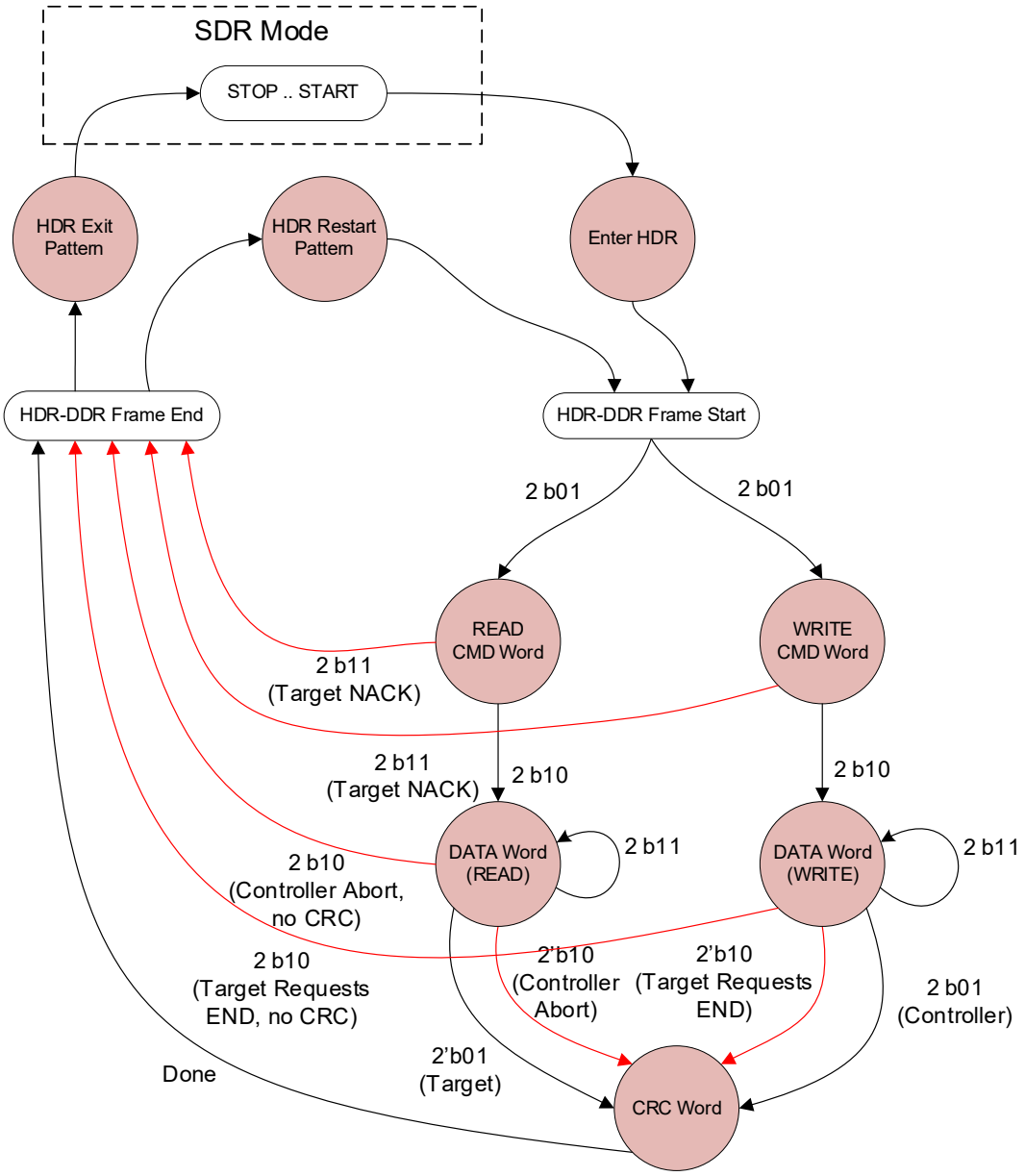


Figure 106 HDR-DDR Preamble Bits State Diagram

- A normal HDR-DDR Read Request shall be composed of a Command Word from the Controller, followed by one or more Data Words from the Target.
  - If the Controller requests the Read transfer to be terminated, then the Target shall conditionally send the CRC Word after termination, based on the configuration set by the ENDXFER CCC (see ***Section 6.2.3.4.5***).
  - If the Read transfer ends normally, then the Target shall send the CRC Word after the last Data Word.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

- A NACKed HDR-DDR Read Request shall be composed of a Command Word, followed by the Target not ACKing (driving SDA to Low) in the second Preamble bit, before what would be the first Data Word (see ***Section 6.2.3.4.2***).
- An HDR-DDR Write Request shall be composed of a Command Word, followed by one or more Data Words, all from the Controller.
  - If the Target requests the Write transfer to be terminated, then the Controller shall conditionally send the CRC Word after termination, based on the configuration set by the ENDXFER CCC (see ***Section 6.2.3.4.6***).
  - If the transfer ends normally, then the Controller shall send the CRC Word after the last Data Word.
- A NACKed HDR-DDR Write Request shall be composed of a Command Word, followed by the Target not ACKing (driving SDA to Low) in the second Preamble bit, before what would be the first Data Word (see ***Section 6.2.3.4.1***).

Each Command Word and each Data Word is 20 bits in length (i.e., 20 clock edges) including a two-bit Preamble. At 10 MHz the raw bit rate is 20 Mbps; at 12.5 MHz the raw bit rate is 25 Mbps. Because each Word includes two Preamble bits and two Parity bits, the real data rate (considering only the two bytes in the 16 payload bits per Word) is the raw rate times 16/20, or 16 Mbps at 10 MHz (20 Mbps at 12.5 MHz), which is a measure of the 16-bit data rate.

The common format used by HDR-DDR Command Words, Data Words, and Reserved Words is illustrated in ***Table 80***. The Command details are explained in ***Section 6.2.3.3***. The format used by HDR-DDR CRC Words is a subset of the Command Word and the Data Word (see ***Section 6.2.3.6***).

Table 80 HDR-DDR Word Format: Command, Data, Reserved

Preamble	**Payload**	**Parity** (not the CRC)	
2 bits	**16 bits**	**1 bit**	**1 bit**
2'b00: Not used **2'b01: Command or CRC** **2'b10, 2'b11: Data or Abort**	Command Code or Data Values	XOR of Odd index Payload bits	XOR of 1 and Even index Payload bits

The two parity bits are formed from the payload bits, using an XOR function:

- **PA1** = Parity of the odd index Payload bits:

$$D[15] \wedge D[13] \wedge D[11] \wedge D[9] \wedge D[7] \wedge D[5] \wedge D[3] \wedge D[1]$$

- **PA0** = Parity of the even index Payload bits and 1:

$$D[14] \wedge D[12] \wedge D[10] \wedge D[8] \wedge D[6] \wedge D[4] \wedge D[2] \wedge D[0] \wedge 1$$

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4627 ***Table 81*** summarizes the format of all four HDR-DDR Word Types.

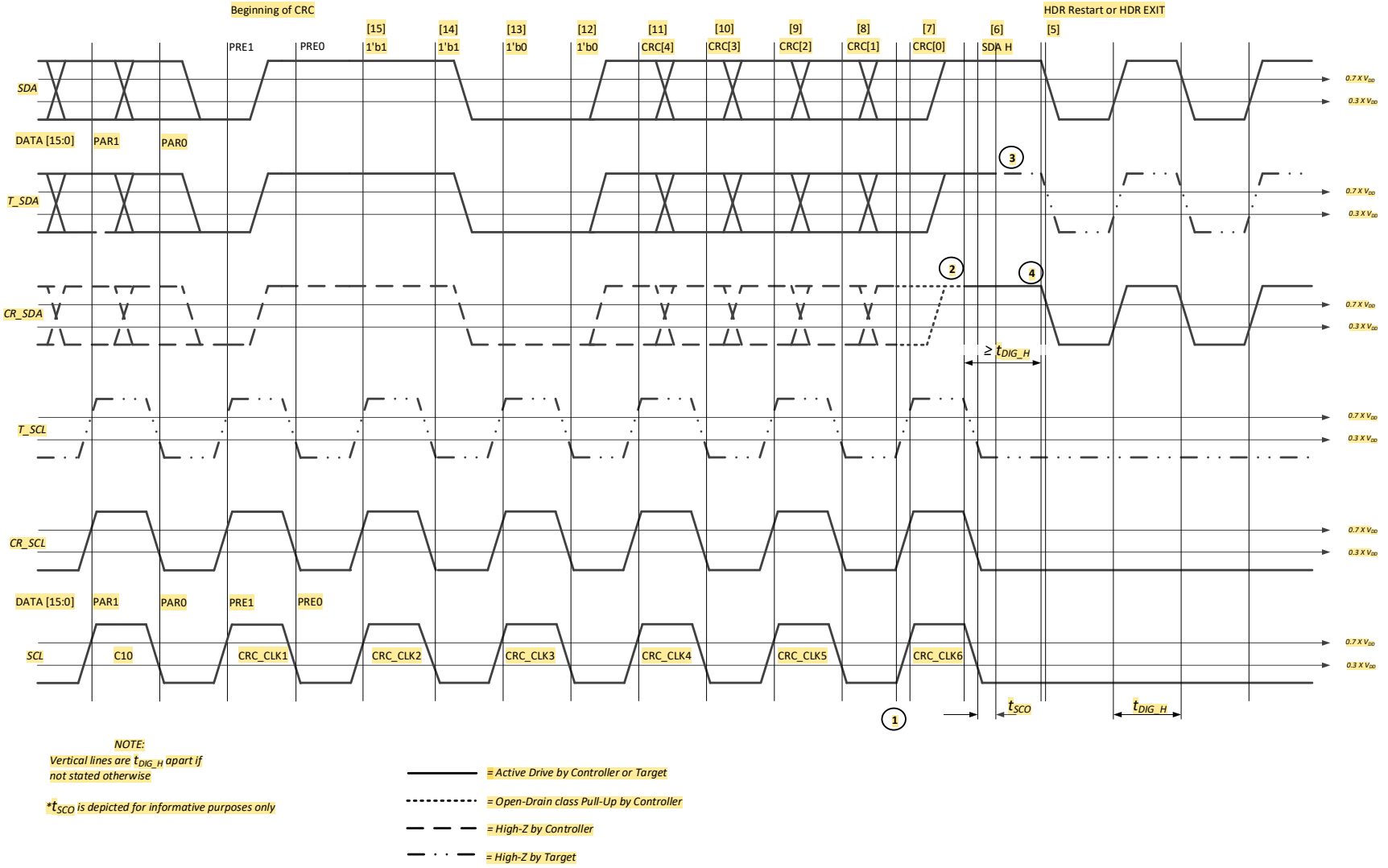
4628 **Table 81 HDR-DDR Word Formats**

See Errata 01, Item 14

Word Type	Preamble	Payload		Parity		Notes
	2 bits	16 bits		1b	1b	
Command	**2'b01**	15:8	Command Code (8 bits)	PA1	PA0	Command may follow only Enter HDR or HDR Restart. Command Codes: Write: 8'h00 to 8'h7F Read: 8'h80 to 8'hFF
		7:1	Target Address (7 bits)			
		0	Reserved (1 bit)			
Data	**2'b10** **2'b11**	15:8	First Data Byte (8 bits)	PA1	PA0	One or more Data Words shall follow Command, unless NACKed. Only CRC may follow Data.
		7:0	Second Data Byte (8 bits)			
CRC	**2'b01**	15:12	4'hC (token value) (4 bits)	Not Used		CRC value ends at bit 6, followed by either HDR Restart Pattern or HDR Exit Pattern. See signal diagrams in ***Figure 107*** and ***Figure 108***.
		11:7	CRC-5 checksum (5 bits)			
		6	1'b1 setup bit, followed by High level for a time of at least t_{DIG_H}, starting from the falling edge of the last SCL clock pulse			
		5:0	Reserved (6 bits), No clocks			
Reserved	**2'b01**	15:12	4'hD to 4'hF (4 bits)	PA1	PA0	Reserved for future use
		11:0	Reserved (12 bits)			
–	2'b00	Reserved for special use cases				

4629 On a WRITE transaction, the Controller controls the whole transfer; as a result, the transition from the CRC
4630 Word to either the HDR Restart Pattern or the HDR Exit Pattern is done with **only** the Controller driving the
4631 SDA, while the Target releases SDA (i.e., allows High-Z) (see ***Figure 107***).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



4632

Figure 107 HDR-DDR CRC and HDR Restart Pattern or HDR Exit Pattern in Write Transaction

See Errata 01, Item 15

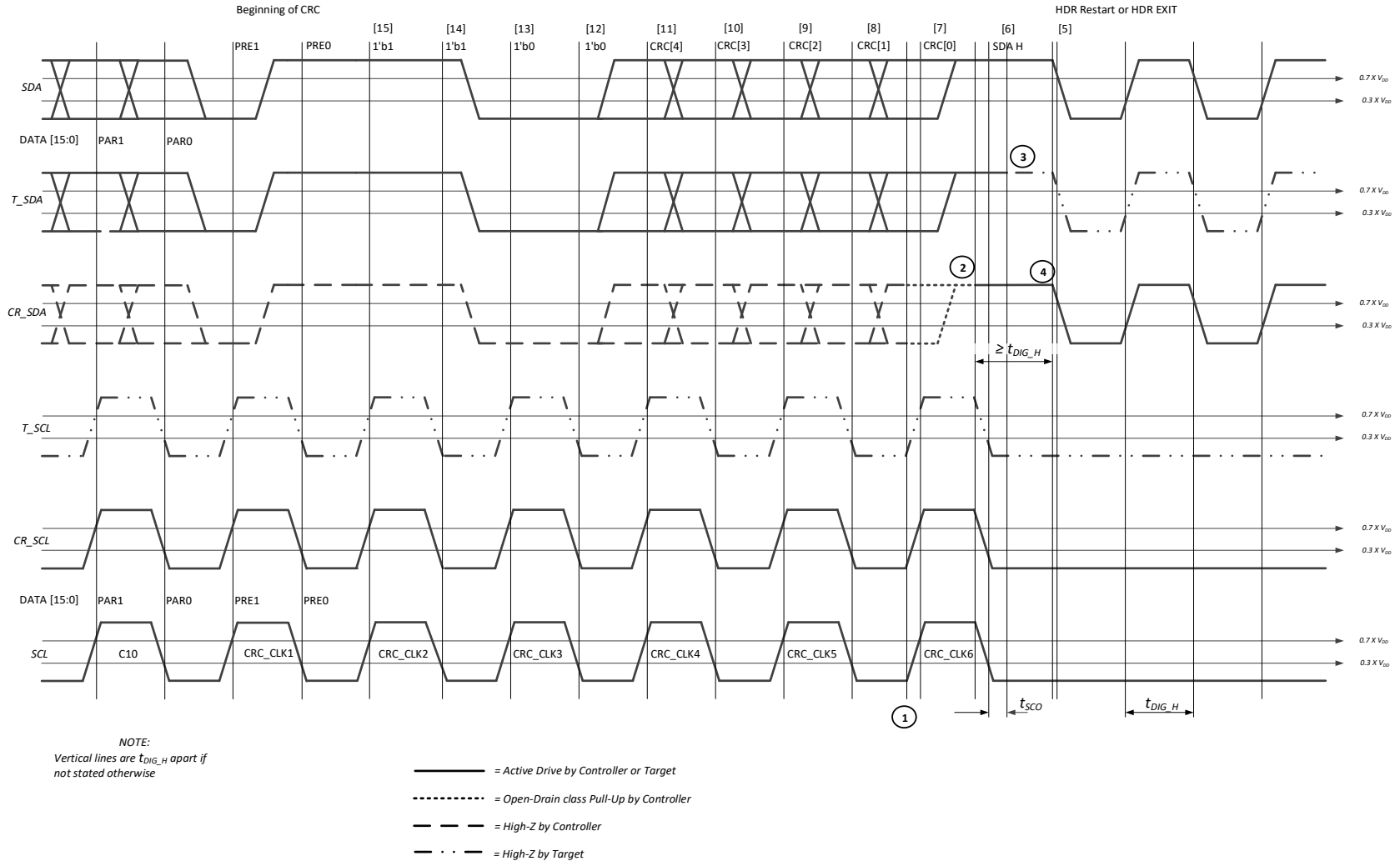
Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

On a Read transaction, the Target conditionally drives the CRC Word (as noted above) and the Controller takes over generation of the HDR Restart Pattern or HDR Exit Pattern. See ***Figure 108*** for the signal diagram for this SDA handoff.

This procedure has the following steps:

1. The Target actively transfers the data, while the Controller is in High-Z.
2. The Target actively drives the CRC Word, including pulling the setup bit (bit[6]) High.
3. After a time of t_{SCO} elapses after the falling edge of CRC_CLK6, the Target releases SDA on High-Z (***Figure 108***, point 3)
4. The Controller keeps its SDA output on High-Z until the start of rising edge of CRC_CLK6, when it enables the Open Drain class Pull-Up (***Figure 108***, point 1). The SDA line remains driven by the Target.
5. The Controller starts driving SDA High when it starts the falling edge of CRC_CLK6 (point 2 in the diagram). Since the Target was driving the setup bit (bit[6]) High, the two Devices are now driving in parallel.
6. After a time of at least t_{DIG_H} elapses after the falling edge of CRC_CLK6, the Controller may start the first falling edge of the SDA, commencing either the HDR Restart Pattern or the HDR Exit Patterns (***Figure 108***, point 4).
7. The result of this procedure is a safe handoff of the SDA line from the Target to the Controller.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



4651

Figure 108 HDR-DDR CRC and HDR Restart Pattern or HDR Exit Pattern in Read Transaction

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.3 HDR-DDR Command Coding

In HDR-DDR Mode a Command Word follows the initial ENTHDR0 CCC (and any HDR Restart Pattern). This Command Word uses the normal 18-bit model. ***Table 82*** illustrates the Command Word format for HDR-DDR Mode.

Table 82 HDR-DDR Command Word Format

Bits	Field	Size (bits)	Notes
15	**Read/Write**	1	1: Read (Target to Controller) 0: Write (Controller to Target)
14:8	**Command Code**	7	128 possible Write commands 128 possible Read commands
7:1	**Target Address**	7	Same Dynamic Address as used in I3C SDR Protocol
0	**Parity Adjustment**	1	Ensures that PA0 contains 1'b1, which allows for easier Bus Turnaround[1]

Note:

1) Because PA0 is the XOR of the even index Payload bits and 1, it is equal to:

XOR_outer(1, this bit, XOR_inner(2, 4, 6, 8, 10, 12, 14))

As a result, this bit should be set to the result of XOR_inner().

Regarding bits 15:8 together, the possible Command Code spaces for Read Commands and Write Commands in HDR-DDR Mode are illustrated in ***Table 83***.

Table 83 Read and Write Command Spaces for HDR-DDR Mode

Command Codes	Command Functions
0x00 – 0x7F	Available for any use for writes
0x80 – 0xFF	Available for any use for reads

HDR-DDR Command Words indicate the direction of data movement: either Write (Controller to Target), or Read (Target to Controller). After the Command Word, one or more Data Words are sent by the Controller (for a Write) or by the Target (for a Read, unless NACKed) until done, followed by the CRC Word (unless NACKed or aborted, per ENDXFER configuration). Finally, the Controller issues either an HDR Restart Pattern or an HDR Exit Pattern. The Controller may also terminate a Read prematurely, per ***Section 6.2.3.4.3***, however this is not a normal end for a Read, and should be used sparingly. The Target may request the termination of a Write if configured to do so, per ***Section 6.2.3.4.6***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Signal diagrams for starting the HDR-DDR Command Codes are shown in ***Figure 109*** and ***Figure 110***. ***Figure 109*** shows the HDR-DDR Command Code after ENTHDR0, and ***Figure 110*** shows the HDR-DDR Command Code after HDR Restart Pattern.

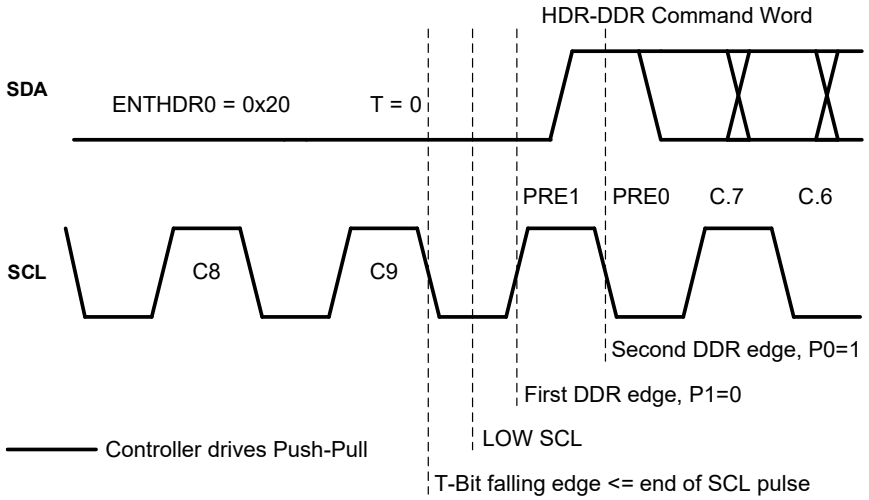


Figure 109 HDR-DDR Command Code After ENTHDR0

As ***Figure 109*** shows:

1. First the SCL pulse is ended by its falling edge (as is necessary for the Targets' logic design).
2. Then on the SCL Low that follows, the Controller positions the SDA (i.e., sets SDA either High or Low) as necessary to be read on the first SCL rising edge, using Push-Pull timing.

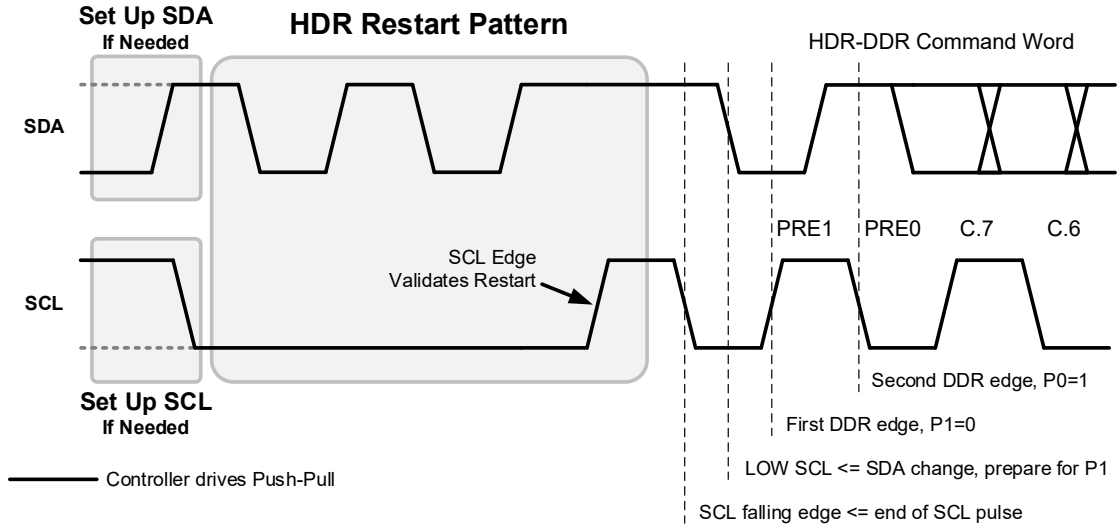


Figure 110 HDR-DDR Command Code After HDR Restart Pattern

As ***Figure 110*** shows:

1. After the SCL rising edge for validating the HDR Restart pattern, an SCL falling edge is provided. The result is an SCL pulse, identical to the SCL pulse of T-Bit (see ***Figure 109***).
2. Any possible changes on SDA are ignored until the SCL rising edge that follows (labeled "First DDR edge").

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

3. On the SCL Low following the SCL pulse falling edge, the Controller positions the SDA (i.e., sets SDA either High or Low) as necessary to be read on the first SCL rising edge, using Push-Pull timing.
4. The waveform that follows is identical to the same segment in ***Figure 109***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.3.1 CCC Transmission in HDR-DDR Mode

A special HDR-DDR syntax is used for transmitting Common Command Codes in the HDR-DDR protocol. The following formats are defined, as specific instantiations of the generic HDR CCC flows (see ***Section 6.1.3***). The Broadcast CCC flow is shown in ***Figure 111***, and the Direct CCC flow is shown in ***Figure 112***.

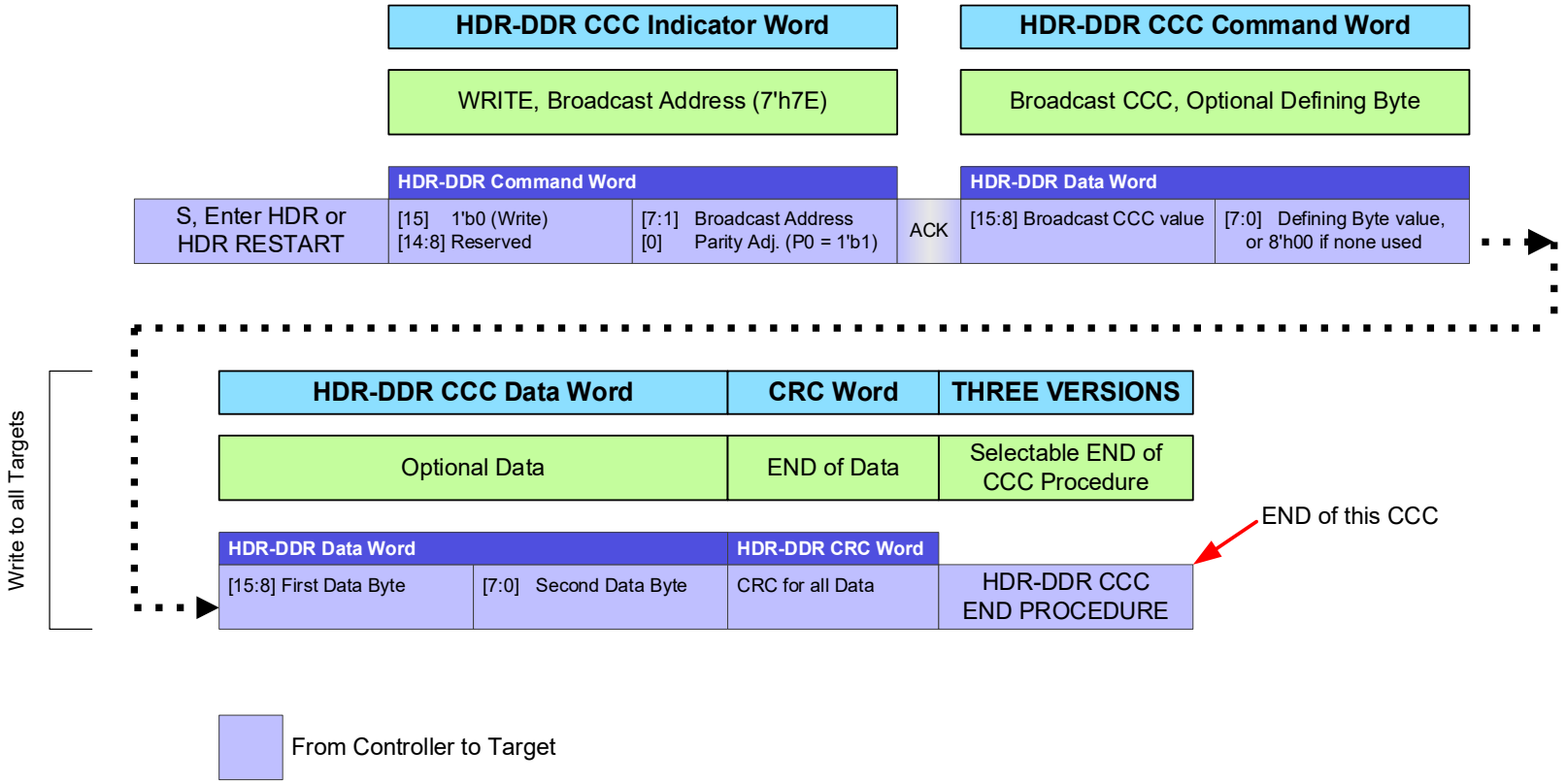


Figure 111 HDR-DDR Broadcast CCC Format

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

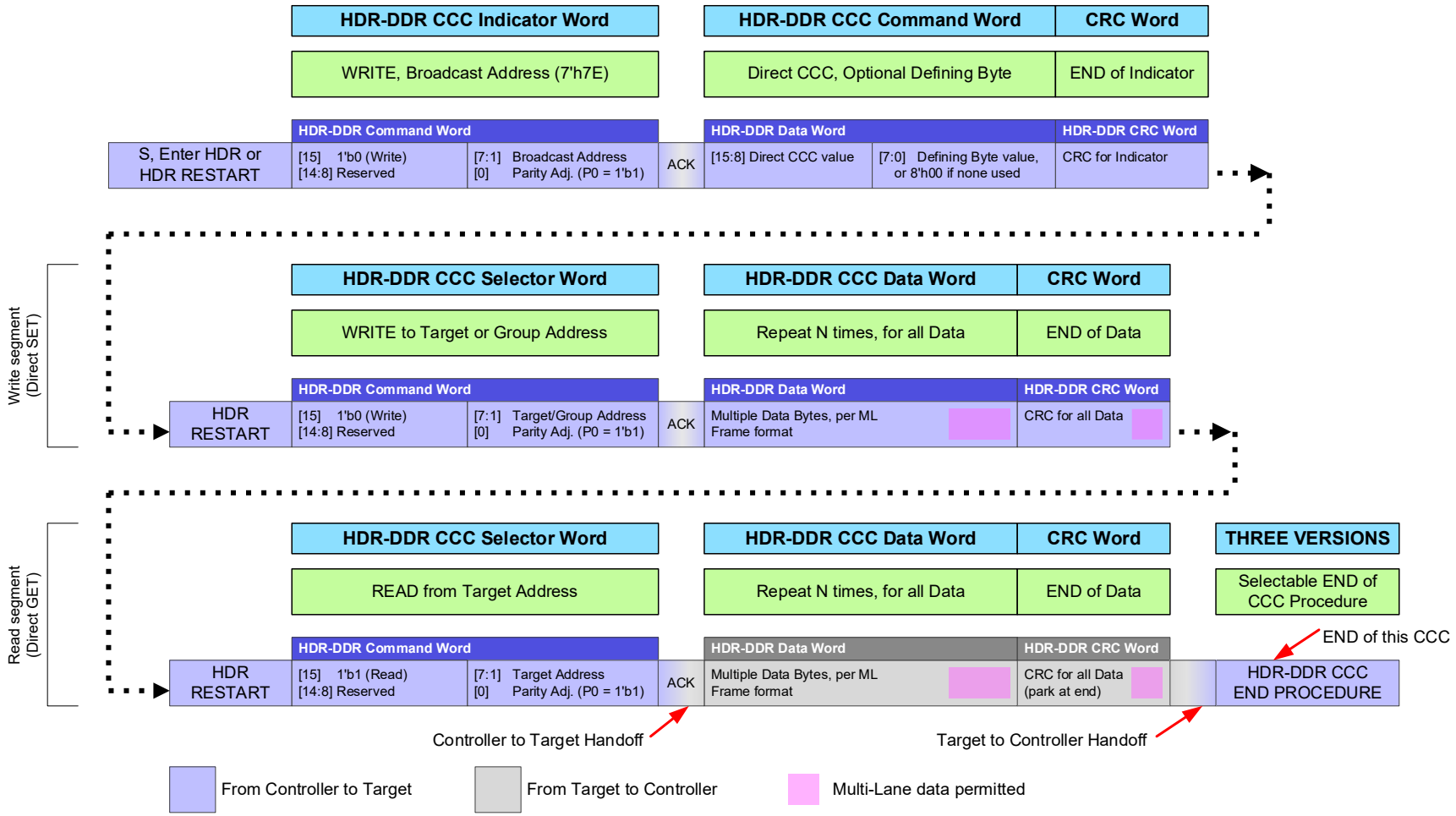


Figure 112 HDR-DDR Direct CCC Format

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.3.1.1 HDR-DDR CCC Header Block

The HDR-DDR CCC Header Block has several different formats that are used in the framing of CCC flows in HDR-DDR Mode. Two of these formats are commonly used: an 'Indicator' type, used at the start of every CCC, and a 'Selector' type, used at the start of a Write Segment or Read Segment for Direct CCCs. A generic Header Block (which is neither an 'Indicator' type nor a 'Selector' type) may also be used for other purposes, such as CCC End Procedures.

The HDR-DDR CCC Header Block always begins with an HDR-DDR Command Word, which shall take its structure (see ***Table 81*** and ***Table 82***). All Targets that support CCC flows in HDR-DDR Mode must acknowledge the HDR-DDR Header Block's Command Word if addressed, unless defined otherwise for a particular usage in CCC flows. If an acknowledgement is required, then Bit[0] shall contain a value chosen to ensure that the PA0 field (Parity bit 0, see ***Table 81***) contains the value 1'b1, per ***Section 6.2.3.4.1***.

Start of CCC: An HDR-DDR CCC Header Block that begins the framing for a Broadcast CCC or a Direct CCC shall consist of one HDR-DDR CCC Indicator Word, followed by one HDR-DDR CCC Command Word:

- The HDR-DDR CCC Indicator Word shall take the structure of the HDR-DDR Command Word Format (see ***Table 81*** and ***Table 82***):
  - Bit[15] shall contain 1'b0, indicating a WRITE command (Controller to Target).
  - Bits[14:8] are reserved, and shall be ignored by the Target Devices.
  - Bits[7:1] shall contain the Broadcast Address (7'h7E).
  - Acknowledgement is always required, so Bit[0] shall contain a value chosen to ensure that all Targets supporting CCC flows in HDR-DDR Mode will acknowledge this Indicator Word.
- The HDR-DDR CCC Command Word shall take the structure of the HDR-DDR Data Word Format (see ***Table 81***).
  - Bits[15:8] shall contain the Command Code, which may be a Broadcast CCC or a Direct CCC.
  - Bits[7:0] shall contain the optional Defining Byte for the Command Code. If the CCC does not support an optional Defining Byte, then this field shall contain the value 8'h00.

Start of Segment: An HDR-DDR CCC Header Block that begins a Read Segment or Write Segment in Direct CCC framing shall consist of one HDR-DDR CCC Selector Word.

- The HDR-DDR CCC Selector Word shall take the structure of the HDR-DDR Command Word Format:
  - Bit[15] distinguishes a Read Segment vs. a Write Segment:
    - For the start of a Write Segment (i.e., a Direct Write or Direct SET CCC) this field shall contain 1'b0 to indicate a Write (Controller to Target).
    - For the start of a Read Segment (i.e., a Direct Read or Direct GET CCC) this field shall contain 1'b1 to indicate a Read (Target to Controller).
  - Bits[14:8] are reserved, and shall be ignored by the Target Devices.
  - Bits[7:1] shall contain the Dynamic Address of the Target to be addressed for Write or Read in this Segment. For Write Segments, Bits[7:1] might also contain a valid Group Address (per ***Section 6.1.3.5***).
  - Acknowledgement is always required, so Bit[0] shall contain a value chosen to ensure that the addressed Target (or all Targets in the addressed Group) will acknowledge this Selector Word. This facilitates ACK/NACK for a Target (or Group) addressed by a Direct Write or Direct SET CCC (per ***Section 6.2.3.4.1***), and facilitates both ACK/NACK and Bus Turnaround for a Target addressed by a Direct Read or Direct GET CCC (per ***Section 6.2.3.4.2***).

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.2.3.3.1.2 HDR-DDR CCC Data Block

An HDR-DDR CCC Data Block may follow an HDR-DDR CCC Header Block, in the appropriate location for either a Broadcast CCC flow (see ***Figure 111***) or a Direct CCC flow (see ***Figure 112***).

- **Broadcast:** The HDR-DDR CCC Data Block in a Broadcast CCC flow shall take the structure of the HDR-DDR Data Word Format (see ***Table 81***) and shall contain 16 bits (2 bytes) of data in its payload.
  - Per ***Section 6.1.3***, all HDR-DDR CCC Data Blocks shall be transmitted using SDA[0] (i.e., 1-Lane mode) in order to ensure that all Target Devices that support HDR-DDR are always able to receive and understand the Broadcast CCC data, regardless of their currently configured ML Frame format, or the number of additional Lanes they might support.
- **Direct:** The HDR-DDR CCC Data Block in a Direct CCC flow shall have the same structure as that of a Broadcast CCC flow (i.e., an HDR-DDR Data Word using SDA[0] in 1-Lane mode).
  - If the total data length is an odd number of bytes, then the sender shall pad the Data Block with an extra byte with value 8'd0, and the receiver shall discard the extra padding byte.

Note:

Multi-Lane support for HDR-DDR Mode is not included in I3C Basic. The above configuration advisory is provided in order to maintain full compatibility with I3C Target Devices that might support Multi-Lane for HDR-DDR Mode, as defined in version 1.1 or higher of the full I3C Specification.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.3.1.3 HDR-DDR CRC Block

An HDR-DDR CRC Block serves as the Termination Marker, as described in the HDR-x generic CCC flows. To conform to the HDR-DDR Mode Framing model (see ***Figure 104***), this Termination Marker uses the same structure as the standard HDR-DDR CRC Word.

- For the end of Broadcast data sent in a Broadcast CCC flow, the HDR-DDR CRC Block shall be a single HDR-DDR CRC Word transmitted by the Controller in 1-Lane mode (see ***Table 81*** and ***Figure 107***), sent as a common CCC flow element to ensure that all Targets will receive it.
- For the end of the HDR-DDR CCC Command Word sent in a Direct CCC flow, the HDR-DDR CRC Block shall be a single HDR-DDR CRC Word transmitted by the Controller in 1-Lane mode as described above.
- For the end of a Read Segment or Write Segment in a Direct CCC flow, the HDR-DDR CRC Block shall be a single HDR-DDR CRC Word (i.e., the same as both flows above).

The HDR-DDR CRC Block shall be included with all common CCC Flow Elements, as it must be received and understood by all HDR-DDR Targets, including those that only support 1-Lane mode.

Note:

Both Broadcast and Direct CCC flows are special types of standard HDR-DDR transfers, so the HDR-DDR CRC Block is used, per standard HDR-DDR transfer requirements. As noted above (i.e., for the HDR-DDR CCC Data Block definition) Multi-Lane support for HDR-DDR Mode is not included in I3C Basic, so the HDR-DDR CRC Block must always use the same format (i.e., Data Transfer Coding) as the HDR-DDR Data Block that precedes it.

6.2.3.3.1.4 HDR-DDR CCC Indicator Word and Header ACK

Per ***Section 6.1.3.2***, a Target that supports CCCs in HDR-DDR Mode shall indicate acknowledgement of the receipt of the HDR-DDR CCC Indicator Word, as well as the HDR-DDR CCC Header Block's Command Word when used for other purposes, using the standard method of acknowledging an HDR-DDR write, per ***Section 6.2.3.4.1*** and ***Figure 114***. Such a Target shall detect matching Command Words in CCC flows (including Indicator Words) that are addressed to the Broadcast Address (i.e., 7'h7E) and have a correctly calculated PA0 field (i.e., Parity bit 0, see ***Table 82***) that contains the value 1'b1.

- This applies to the Indicator Word when used with both Broadcast CCC flows and Direct CCC flows, since the same HDR-DDR CCC Indicator Word format is used to signal the beginning of both flows.
- This also applies to the Command Word when used appropriately for other purposes in CCC flows, such as CCC End Procedures.

If the Controller does not receive acknowledgement from at least one Target, then it shall perform any valid HDR-DDR CCC End Procedure. Additionally, it may initiate an error recovery procedure, which might include exiting HDR-DDR Mode and attempting to re-send the CCC in SDR Mode.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.2.3.3.1.5 HDR-DDR Direct CCC Write Segment

An HDR-DDR CCC Write Segment for a Direct Write or Direct SET CCC shall start with an HDR-DDR CCC Selector Word to indicate the Address of the Target (or Group) to receive the data, followed by one or more HDR-DDR CCC Data Blocks of the appropriate structure, as required by the CCC definition.

After the Controller sends the HDR-DDR CCC Selector Word, the Target shall then either indicate acceptance of the Direct CCC by acknowledging the WRITE Command, or else reject the Direct CCC by not responding to the Controller. The Target shall use the cached Command Code and optional Defining Byte from the previously received HDR-DDR CCC Command Word, and shall decide whether to accept or reject the Direct CCC immediately after receiving the HDR-DDR CCC Selector Word and matching its own Dynamic Address (or Group Address, if assigned and applicable).

Note:

If the CCC definition does not require data to be written to the Target or Group, then the Controller shall send a single HDR-DDR CCC Data Block that contains padding bytes (i.e., values of 8'd0). The Target shall discard the extra padding bytes, based on the cached Command Code and optional Defining Byte that were sent previously, as part of the CCC Framing in HDR-DDR Mode.

The Target shall use the standard HDR-DDR Flow Control methods to either accept or reject an HDR-DDR WRITE Command, as defined in ***Section 6.2.3.4.1*** and as illustrated in ***Figure 114*** (accept) or ***Figure 115*** (reject).

If the Target rejects the Direct CCC, then the Controller shall either send the HDR Restart Pattern and attempt another Read or Write Segment for this CCC, or else perform any valid HDR-DDR CCC End Procedure. For Direct Write or Direct SET CCCs to a Group Address, the Controller only sees this rejection if no Targets accept the Direct CCC.

If the Target accepts the Direct CCC, then the Controller shall send one or more HDR-DDR CCC Data Blocks of the appropriate structure, if required by the CCC definition. Then the Controller shall send an HDR-DDR CRC Word to terminate the data and end the Write Segment.

At the end of a Write Segment, the Controller shall either send the HDR Restart Pattern and attempt another Read or Write Segment for this CCC, or else perform any valid HDR-DDR CCC End Procedure.

6.2.3.3.1.6 HDR-DDR Direct CCC Read Segment

An HDR-DDR CCC Read Segment for a Direct Read or Direct GET CCC shall start with an HDR-DDR CCC Selector Word to indicate the Address of the Target to provide data for the CCC, followed by a Bus Turnaround. In response, the indicated Target will either accept the Direct CCC by transmitting one or more HDR-DDR CCC Data Blocks of the appropriate structure, or else reject the Direct CCC by not responding to the Bus Turnaround condition.

The Target shall use the standard HDR-DDR Flow Control methods to either accept or reject an HDR-DDR READ Command, as defined in ***Section 6.2.3.4.2*** and as illustrated in ***Figure 116*** (accept) or ***Figure 117*** (reject).

If the Target accepts the Direct CCC, then the Controller shall yield control of the Bus to the Target so that it can control the Bus to send the Data Blocks. After sending the last Data Block, the Target shall send an HDR-DDR CRC Word, indicating the end of the read transfer. Upon receiving the HDR-DDR CRC Word the Target shall yield control of the Bus, and the Controller shall resume control. **See Errata 01, Item 16**

At the end of a Read Segment, the Controller shall either send the HDR Restart Pattern and attempt another Read or Write Segment for this CCC, or else perform any valid HDR-DDR CCC End Procedure.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.3.1.7 HDR-DDR CCC End Procedures

As illustrated in ***Figure 118***, the HDR-DDR CCC ends using one of three possible CCC End Procedures.

- **Option 1: END HDR-DDR CCC Followed by New HDR-DDR CCC**

  This End Procedure indicates the end of a Broadcast or Direct CCC in HDR-DDR Mode, and the beginning of a new Broadcast or Direct CCC in HDR-DDR Mode.

  This End Procedure begins with an HDR Restart Pattern, followed by an HDR-DDR CCC Header Block containing a new CCC and optional Defining Byte. This starts either a Broadcast CCC flow (***Figure 111***) or a Direct CCC flow (see ***Figure 112***) while remaining in the CCC modality.

  The Controller starts a new CCC by sending one HDR-DDR Command Word, acting as the HDR-DDR CCC Indicator Word and containing the Broadcast Address; followed by one HDR-DDR Data Word, acting as the HDR-DDR CCC Command Word and containing the Command Code and optional Defining Byte for the new CCC.

  Additional HDR-DDR Words shall follow, per the CCC flow type.

  The flow steps are:

  1. Controller sends HDR Restart Pattern.
  2. Controller sends HDR-DDR CCC Indicator Word, using the same format as described above.
  3. Targets acknowledge receipt of Indicator Word, as defined above.
  4. Controller sends HDR-DDR CCC Command Word, including the Command Code and optional Defining Byte.
  5. Flow continues as Broadcast CCC or Direct CCC, as shown above.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

- **Option 2: END HDR-DDR CCC Followed by Another HDR-DDR Generic Transaction (+ Optional HDR Exit)**

  This End Procedure indicates the end of CCCs in HDR-DDR, and either a return to standard HDR-DDR transactions or a subsequent exit from HDR-DDR Mode.

  This End Procedure begins with an HDR Restart Pattern, followed by an HDR-DDR Command Word which ends the CCC flow, but is neither an Indicator nor a Selector. This Command Word is a common flow element that is sent to all Targets and must be acknowledged (i.e., like the Indicator Word).

  This is followed by an HDR-DDR Data Word, a common flow element that is required for HDR-DDR WRITE transfers (per ***Section 6.2.3.2***). This Data Word contains the dummy command code value (i.e., 0x1F, see ***Table 16***) to indicate the end of the CCC modality in HDR-DDR Mode, per ***Section 6.1.3.1***.

  Note:

  This HDR-DDR Data Word is not an HDR-DDR CCC Command Word.

  This is followed by an HDR-DDR CRC Word as a common flow element. This CRC Word's checksum is computed from the contents of the preceding HDR-DDR Data Word.

  After this, the HDR Restart Pattern signals the end of CCC modality, and that the next HDR-DDR Command Word must be treated as a standard HDR-DDR transaction (not as a CCC flow). Alternatively, the HDR Exit Pattern signals the end of HDR-DDR Mode and a subsequent return to SDR Mode.

  In either case, the HDR-DDR Command Word that indicates a write to the Broadcast Address followed by an HDR-DDR Data Word that contains the dummy command code value (i.e., 0x1F) is distinct from any other valid CCC flow in HDR-DDR Mode. Target Devices shall interpret this flow as the ending of the CCC modality.

  The flow steps are:

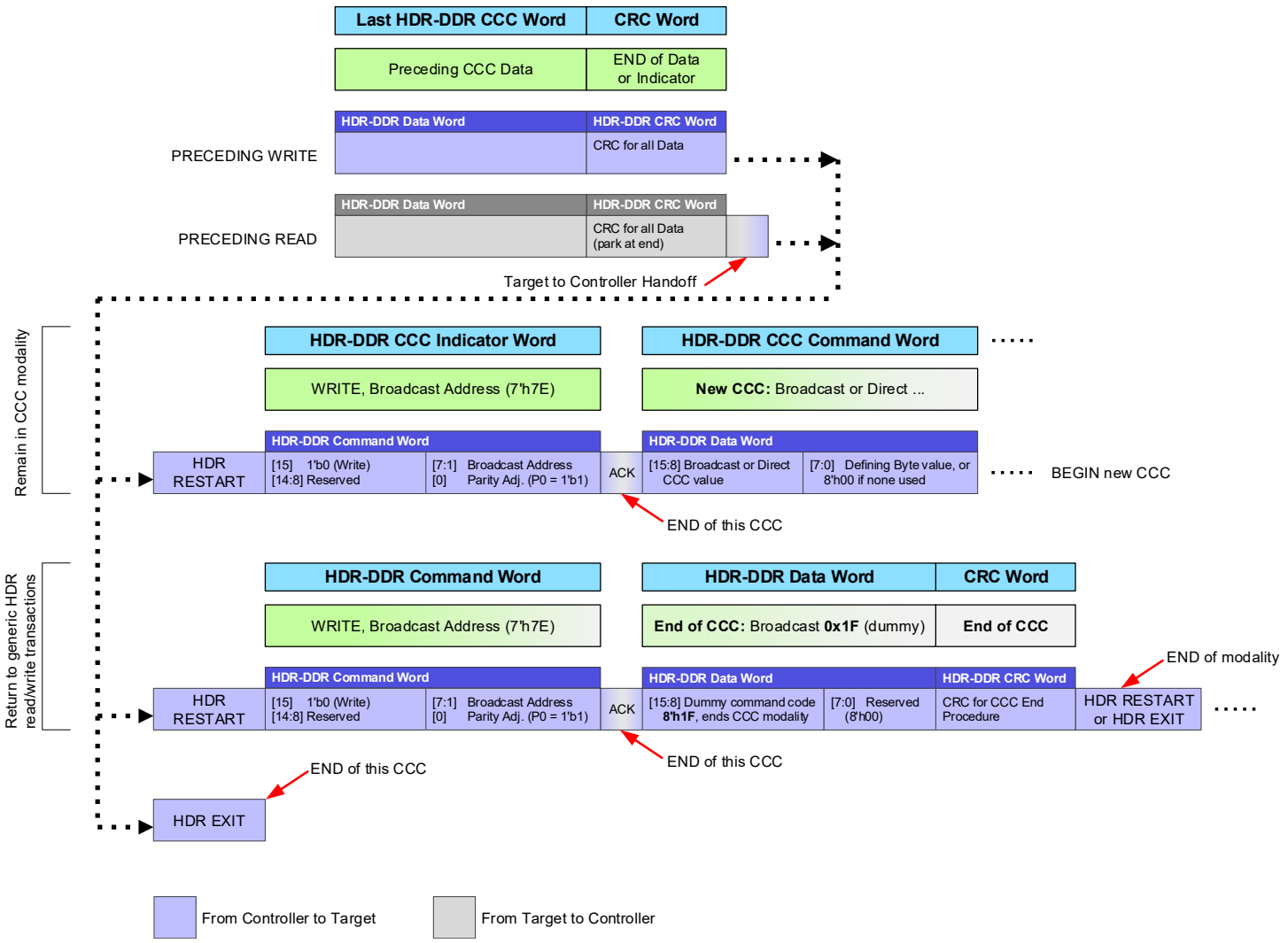
  1. Controller sends HDR Restart Pattern.
  2. Controller sends HDR-DDR Command Word (i.e., not Indicator or Selector), using the following values:
     - Bit[15] shall contain the value 1'b0, indicating a WRITE command (Controller to Target).
     - Bits[14:8] are reserved, and shall be ignored by the Target Devices.
     - Bits[7:1] shall contain the Broadcast Address (7'h7E).
     - Bit[0] shall follow the general rule for all Command Words in HDR-DDR CCC Header Blocks, to facilitate ACK/NACK by all Targets.
  3. Targets acknowledge receipt of the HDR-DDR Command Word, as above.
  4. Controller sends HDR-DDR Data Word, using the following values:
     - Bits[15:8] shall contain the dummy command code (0x1F) to end the CCC modality.
     - Bits[7:0] are reserved, and shall contain zeros.
  5. Controller sends HDR-DDR CRC Word.

     Targets detect this flow (i.e., Command Word, one Data Word with dummy Command Code, and CRC Word) and end CCC modality.
  6. Controller either sends HDR Restart Pattern, or HDR Exit Pattern.

- **Option 3: END HDR-DDR CCC and Return to SDR Mode**

  This is the simplest End Procedure. It indicates the end of CCCs in HDR-DDR Mode and an immediate exit from HDR-DDR Mode, when the Controller sends the HDR Exit Pattern.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



4884

Figure 113 HDR-DDR CCC End Procedures

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.4 HDR-DDR Flow Control Elements

The I3C HDR-DDR protocol provides several procedures that allow either the Controller or the Target to control the progress of data transfers. This Section describes these flow control elements.

HDR-DDR flow control is configured by the ENDXFER CCC which controls the specific parameters of the Data Transfer Early Termination procedure. ***Section 6.2.3.4.4*** covers the flow control configuration details.

Termination of a data transfer at the request of the Receiver is detailed in ***Section 6.2.3.4.5*** and ***Section 6.2.3.4.6***. The procedures allowing the Receiver to terminate a data transfer shall be used only after the system is appropriately configured. Once configured, the Controller and Target(s) shall use the appropriate procedures in these sections for both Read Commands and Write Commands, since the Receiver varies depending on the direction of the Command.

Note:

*The HDR-DDR flow control configuration defined in **Section 6.2.3.4.4** and flow control procedures defined in **Section 6.2.3.4.5** through **Section 6.2.3.4.6** are new for version 1.1 of the I3C Specification (i.e., they were not defined in version 1.0 of the I3C Specification). Since some older Targets that support HDR-DDR Mode might not support such procedures, it is the Controller's responsibility to determine which Targets support these flow control procedures, and also to enable HDR-DDR flow control appropriately (if supported). However, **all** of the flow control procedures defined in this section and its sub-sections are mandatory for Targets that support HDR-DDR Mode and also comply with version 1.1 or later of the I3C Specification, as well as version 1.1.1 or later of the I3C Basic Specification.* **See Errata 01, Item 17**

If HDR-DDR flow control is not enabled (i.e., the default state), then:

- The Target shall accept (i.e., shall absorb) all Data Words for a Write Command sent by the Controller.
  - This applies even if the Target does not support the Command Byte and will simply drop the data from the Controller.
  - If ACK/NACK is not enabled, then the Controller will have full control of all Preamble bits.
- The Target may NACK (i.e., may ignore) a Read Command from the Controller.
- The Controller may use the procedures defined in ***Section 6.2.3.4.5*** to terminate a Read Command. However if this occurs, then the Target is always obligated to send the CRC Word after the last Data Word. **See Errata 01, Item 18**
- The Target shall not terminate a Write Command.
- Additional details in ***Section 6.2.3.4.5*** through ***Section 6.2.3.4.6*** do not apply.

If HDR-DDR flow control is supported, and if it is subsequently enabled by the Controller using the ENDXFER CCC (per ***Section 6.2.3.4.4***), then:

- The Target may NACK a Write Command if it has been configured to do so (i.e., if Bit[4] = 1'b0 in the data byte for Sub-Command 0xF7).
  - In this case, the Target may use the procedure in ***Section 6.2.3.4.1.2 Ignoring the Write Command*** which will be interpreted as a NACK.
- **For Read Commands:**
  - The Controller and Target may use the procedures in ***Section 6.2.3.4.5*** to either end or continue the transfer of Data Words from the Target for the Read Command. If the Controller terminates the Read transfer, then the Target shall conditionally send the CRC Word after the last Data Word, per the configuration.
- **For Write Commands:** (i.e., if Bit[5] = 1'b0 in the data byte for Sub-Command 0xF7)
  - The Controller and Target may use the procedures in ***Section 6.2.3.4.6*** to either end or continue the transfer of Data Words from the Controller for the Write Command. If the Target terminates the Write transfer, then the Controller shall conditionally send the CRC Word after the last Data Word, per the configuration.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

- If the Target supports Write Abort, then it shall indicate this by returning Bit[6] = 1'b1 in the GETCAP2 byte, when responding to the GETCAPS CCC (see ***Section 5.4***).

6.2.3.4.1 Command to Write Data to Target

In response to a Write Command in HDR-DDR Mode, a Target shall either accept one or more Data Words, or else ignore the Write Command.

Note:

*Per **Table 80**, the Target's decision to accept or ignore the Write Command shall determine the Preamble Value of the first HDR-DDR Data Word. See **Figure 110** and* **Figure 111**.

6.2.3.4.1.1 Accepting the Write Command

To accept the DDR WRITE from the Controller, the Target actively drives SDA Low before the C1 falling edge (i.e., while SCL is High) for the first HDR-DDR Data Word.

Figure 114 illustrates Target acceptance (ACK) signaling for the Controller's DDR WRITE command, using numbered blue dots that correspond to the numbered steps below.

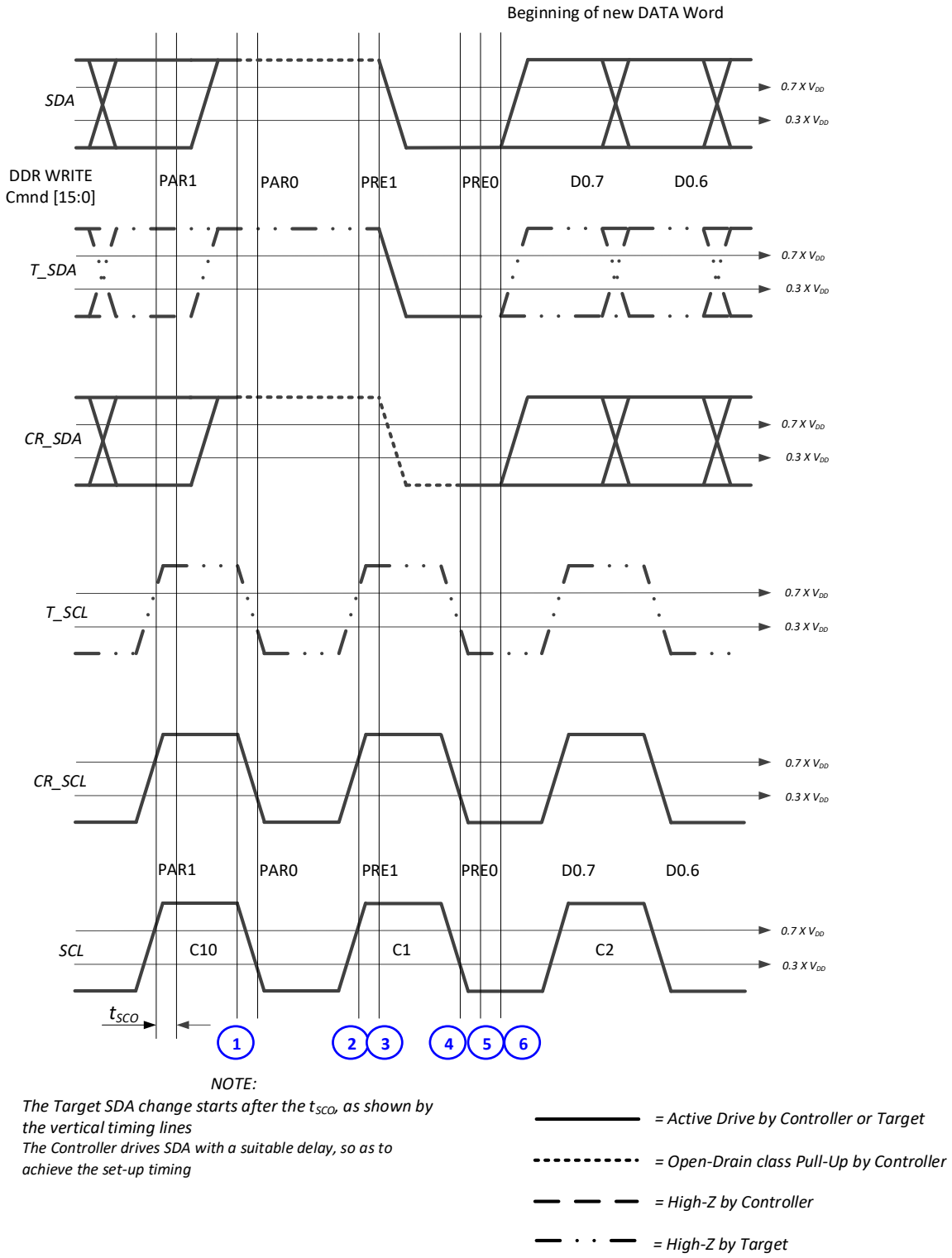
The significant steps are:

1. The Controller:
   A. Has selected bit 0 of the Command Word (the Parity Adjustment bit) to ensure that parity bit PA0 (the last bit in the Word) is High (1'b1)
   B. Disables the active drive of SDA
   C. Enables the Open Drain class Pull-Up structure on SDA, keeping SDA High
2. The first Preamble bit, PRE1, is found to be High (1'b1)
3. Once t_{SCO} elapses after the C1 rising edge, the Target actively drives SDA Low
4. The Controller then:
   A. Finds that the second Preamble bit, PRE0, is Low (1'b0), and that therefore the Target has accepted the DDR WRITE command, and
   B. Disables the Open Drain class Pull-Up, and
   C. Drives actively SDA Low, in parallel with the Target
5. Once t_{SCO} elapses after the C1 falling edge, the Target releases the SDA on High-Z
6. After a suitable delay, the Controller then starts actively driving SDA High or Low, as required by the value of the first data bit (D0.7)

The Controller continues transmitting the data, eventually ending the transfer with the CRC Word. The procedure for the Controller ending the data stream is specified in ***Section 6.2.3.4.2***.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition



4961

Figure 114 Target Accepts DDR WRITE Command from Controller

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.4.1.2 Ignoring the Write Command

To ignore the DDR WRITE from the Controller, the Target allows SDA to remain High through the C1 falling edge (i.e., while SCL goes from High to Low) for the first HDR-DDR Data Word. This is only allowed if the Controller previously used the ENDXFER CCC to enable ACK/NACK capability for the Write Command (see ***Section 6.2.3.4.4***).

Note:

Reasons for the lack of a response could include that the Target might not be present on the Bus, or that the Target might not want to accept the DDR WRITE command (e.g., the Command Byte is not supported Command Byte or because the Target is not ready).

Figure 115 illustrates signaling when the Target does not respond to the Controller's DDR WRITE command, using numbered blue dots that correspond to the numbered steps below.

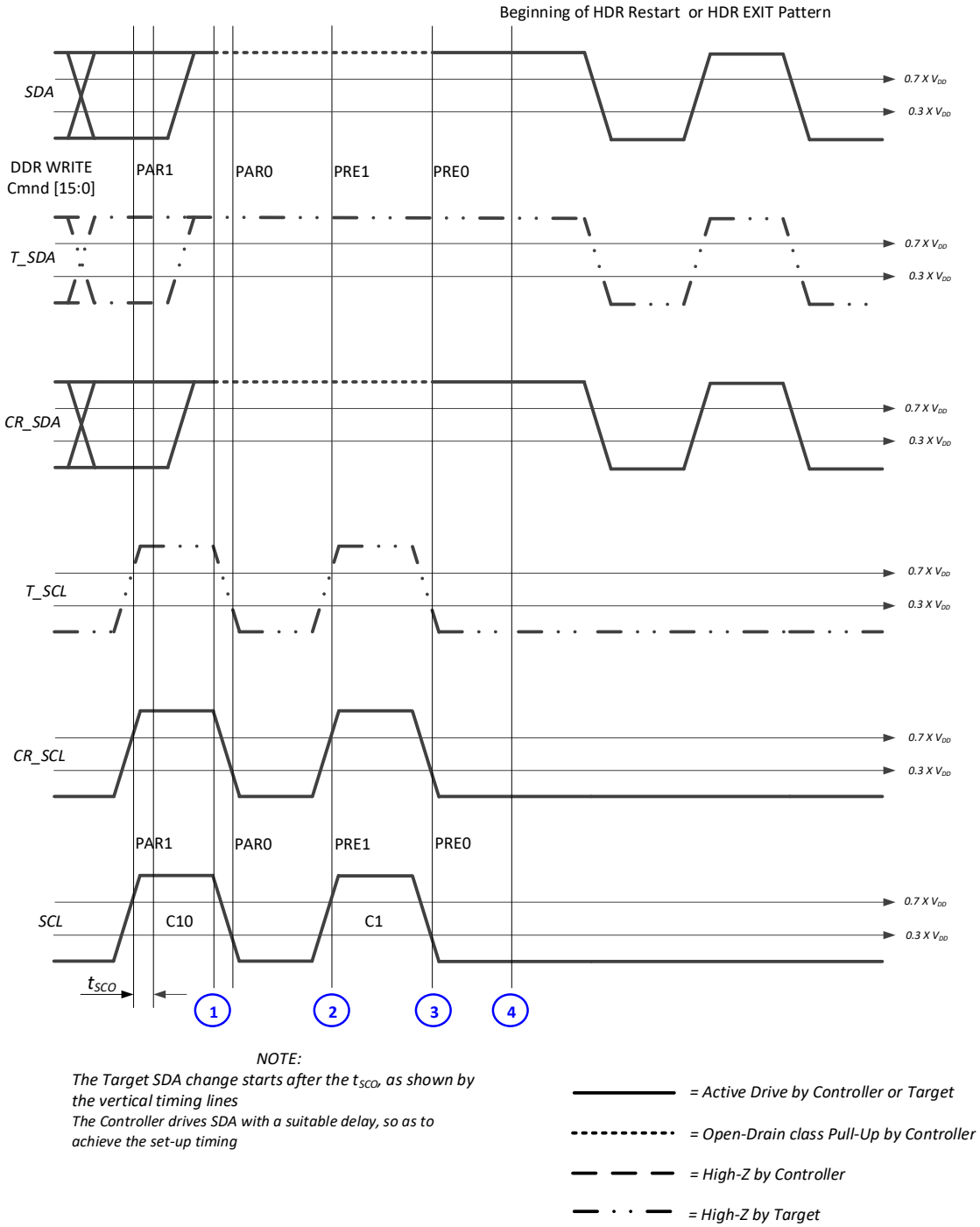
The significant steps are:

1. The Controller:
   A. Has selected bit 0 of the Command Word (the Parity Adjustment bit) to ensure that parity bit PA0 (last bit of the Word) has the value High (1'b1)
   B. Enables the Open Drain class Pull-Up structure on SDA, keeping SDA High
2. The Controller finds the first Preamble bit, PRE1, to be High (1'b1)
3. Because the Target does not control SDA, the second Preamble bit, PRE0, is also found to be High (1'b1)

   As a result, the Controller can consider the Target to be non-responsive to the DDR WRITE Command
4. The Controller concludes by starting either an HDR Restart, or an HDR Exit Pattern, respectively.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition



4984

Figure 115 Target Does Not Respond to DDR WRITE Command from Controller

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.4.2 Command to Read Data from Target

In response to a Read Command in HDR-DDR Mode, a Target shall either return one or more Data Words, or else ignore the Read Command. If the Command asks the Target to return one or more Data Words, then the I3C Bus Turnaround occurs on the Preamble value 2'b10 immediately preceding the Data.

Note:

*Per **Table 79**, the Target's decision to accept or ignore the Read Command shall determine the Preamble Value of the first HDR-DDR Data Word. See **Figure 116** and **Figure 117** below.*

6.2.3.4.2.1 Accepting the Read Command

To accept the DDR READ from the Controller, the Target actively drives SDA Low before the C1 falling edge (i.e., while SCL is High) for the first HDR-DDR Data Word. ***Figure 116*** illustrates Target acceptance (ACK) signaling for the Controller's DDR READ command, using numbered blue dots that correspond to the numbered steps below.

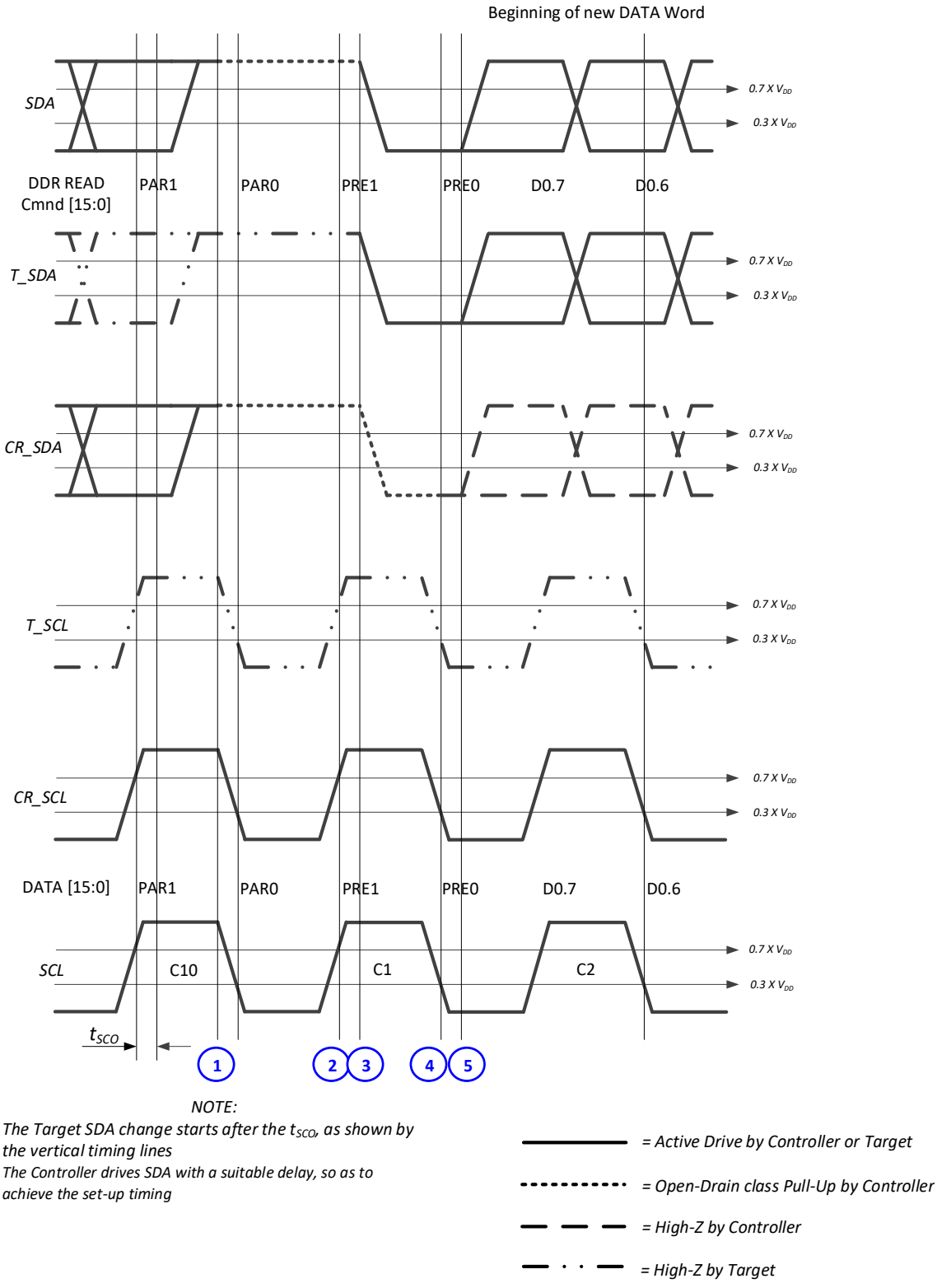
The significant steps are:

1. The Controller:
   A. Has selected bit 0 of the Command Word (the Parity Adjustment bit) to ensure that parity bit PA0 (the last bit in the Word) is High (1'b1)
   B. Disable the active drive of SDA
   C. Enables the Open Drain class Pull-Up structure on SDA, keeping SDA High
2. The first Preamble bit, PRE1, is found to be High (1'b1)
3. Once t_{SCO} elapses after the C1 rising edge, the Target actively drives SDA Low
4. The Controller then:
   A. Finds that the second Preamble bit, PRE0, is Low (1'b0), and that therefore the Target has accepted the DDR READ command, and
   B. Disables the Open Drain class Pull-Up and releases SDA on High-Z
5. Once t_{SCO} elapses after the C1 falling edge, the Target actively drives SDA either High or Low, as required by the value of the first data bit (D0.7)

The Target continues transmitting the data, eventually ending the transfer with the CRC Word. The procedure for the Target ending the data stream is specified in ***Section 6.2.3.4.3***.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition



5011

Figure 116 Target Accepts DDR READ Command from Controller

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.4.2.2 Ignoring the Read Command

To ignore the DDR READ from the Controller, the Target allows SDA to remain High through the C1 falling edge (i.e., while SCL goes from High to Low) when the Controller tries to drive the first HDR-DDR Data Word.

Note:

Reasons for the lack of a response could include that the Target might not be present on the Bus, or that the Target might not want to accept the DDR READ command.

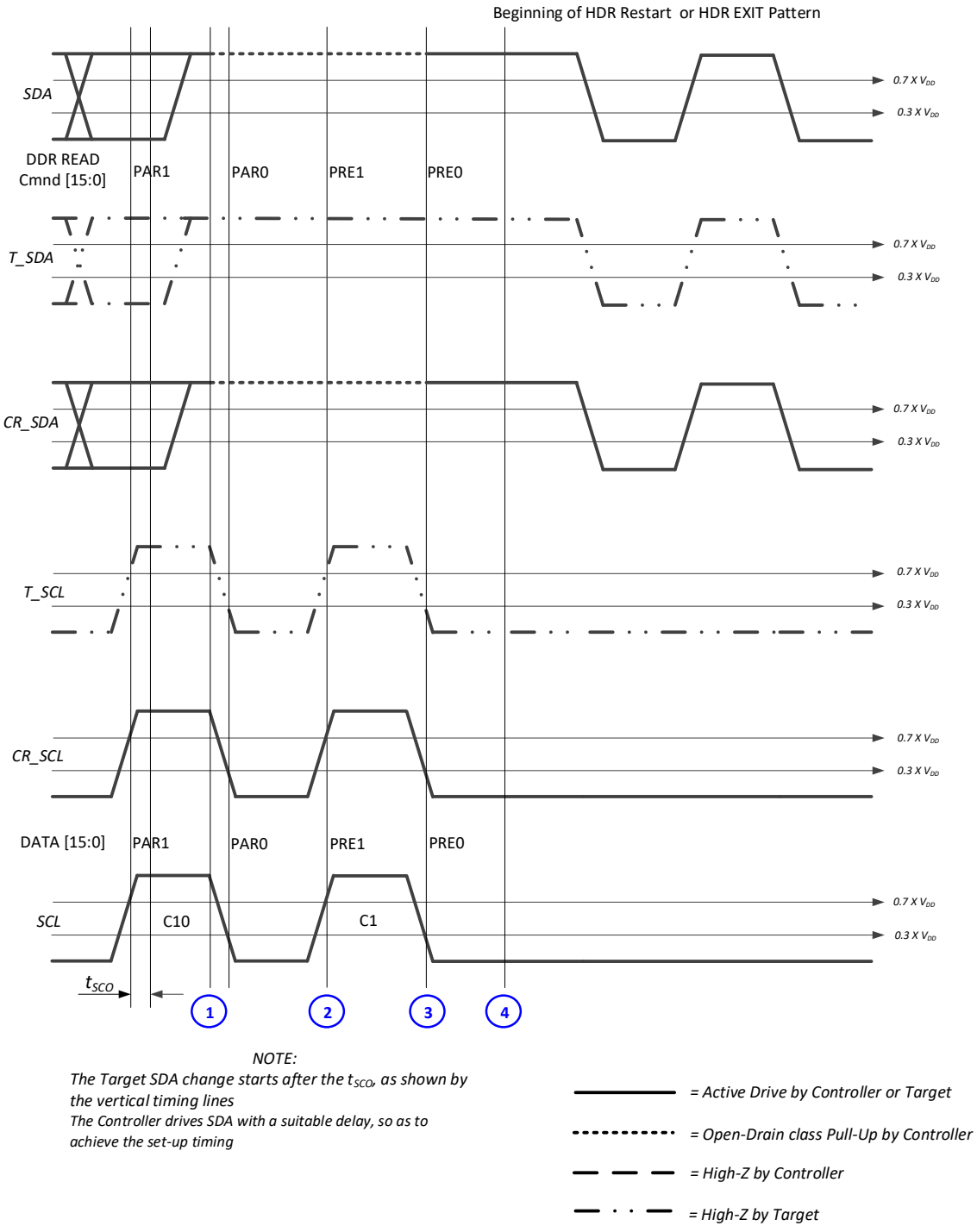
Figure 117 illustrates signaling when the Target does not respond to the Controller's DDR READ command, using numbered blue dots that correspond to the numbered steps below.

The significant steps are:

1. The Controller:
   A. Has selected bit 0 of the Command Word (the Parity Adjustment bit) to ensure that parity bit PA0 (last bit of the Word) has the value High (1'b1)
   B. Enables the Open Drain class Pull-Up structure on SDA, keeping SDA High
2. The Controller finds the first Preamble bit, PRE1, to be High (1'b1)
3. Because the Target does not control SDA, the second Preamble bit, PRE0, is also found to be High (1'b1)
   A. As a result, the Controller can consider the Target to be non-responsive to the DDR READ
4. The Controller concludes by starting either an HDR Restart, or an HDR Exit Pattern, respectively.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition



5031

Figure 117 Target Does Not Respond to DDR READ Command from Controller

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.4.3 End of a Complete Data Frame Transfer

The number of Data Words associated with a Command is in many cases a contract between the Controller and Target; that is, it depends on the particular Command Code. However, the HDR-DDR Protocol supports the transmission of variable length Data by the Controller (for a Write) by the Target (for a Read). Additionally, a Controller may terminate a Read (see ***Section 6.2.3.4.4***) or a Target can request a termination of a WRITE (see ***Section 6.2.3.4.5***).

For a Write (Controller to Target), the Target shall know that Data transmission is done when a CRC Word followed by the HDR Restart Pattern or HDR Exit Pattern is detected.

For a Read (Target to Controller), the Target shall issue a CRC Word upon conclusion of the data transmission, and then shall Park the SDA line High on the first bit after the CRC 5-bit value, followed by tri-stating the SDA line. The Controller shall see the CRC Word, and then expect to see the 1 (Parked). The Controller shall then make sure that the SCL line is Low, and then transmit an HDR Restart Pattern or HDR Exit Pattern, after the Target has High-Z'ed the SDA line.

6.2.3.4.4 HDR-DDR Early Transfer Termination Set-Up

The Controller usually ends a Write Message with a CRC Word, and then continues with either an HDR Restart Pattern or an HDR Exit Pattern. For HDR-DDR Mode, the normal model is that the Target ends a Read with a CRC Word, and then hands control of the I3C Bus back to the Controller. Both these cases are described in ***Section 6.2.3.4.3.***

However, HDR-DDR Mode also allows the Controller to terminate a Read early if necessary, for example if there is an unexpected need to regain the Bus. HDR-DDR Mode also allows the Target to request early termination of a Write data transfer if necessary, for example if the Target's internal storage overflows.

Early termination signaling is accomplished using the two Preamble bits that the Controller or Target transmit before every Data Word. As ***Table 81*** illustrates, the first Preamble bit (PRE1) is always High (1'b1) and is read on the rising edge of SCL. The second Preamble bit (PRE0) is read on the falling edge of the same SCL pulse. If PRE0 is Low (1'b0), then this indicates termination of the data transfer; if PRE0 is High (1'b1), then the data transfer continues.

There are two possibilities for the sender to end the transmission: either providing the corresponding CRC Word, or ending the data string without it. Since the CRC is computed on the complete Message, there are implementations where CRC could not be calculated in time. In order to accommodate both types of design, the HDR-DDR early Message termination offers the option of either providing the CRC Word, or not providing it.

To avoid any possible collision between active drivers in the Target and Controller, the choice of whether to provide the CRC is controlled by a set-up procedure which is acknowledged by the Devices involved in the particular HDR-DDR data transfer.

The ENDXFER CCC (***Section 4.3.7.3.22***) controls the set-up and invocation of the Data Transfer Early Termination Procedure for the HDR-DDR protocol. The related ENDXFER CCC's Defining Byte acts as a sub-command that determines the follow-up actions.

Copyright © 2016–2025 MIPI Alliance, Inc.

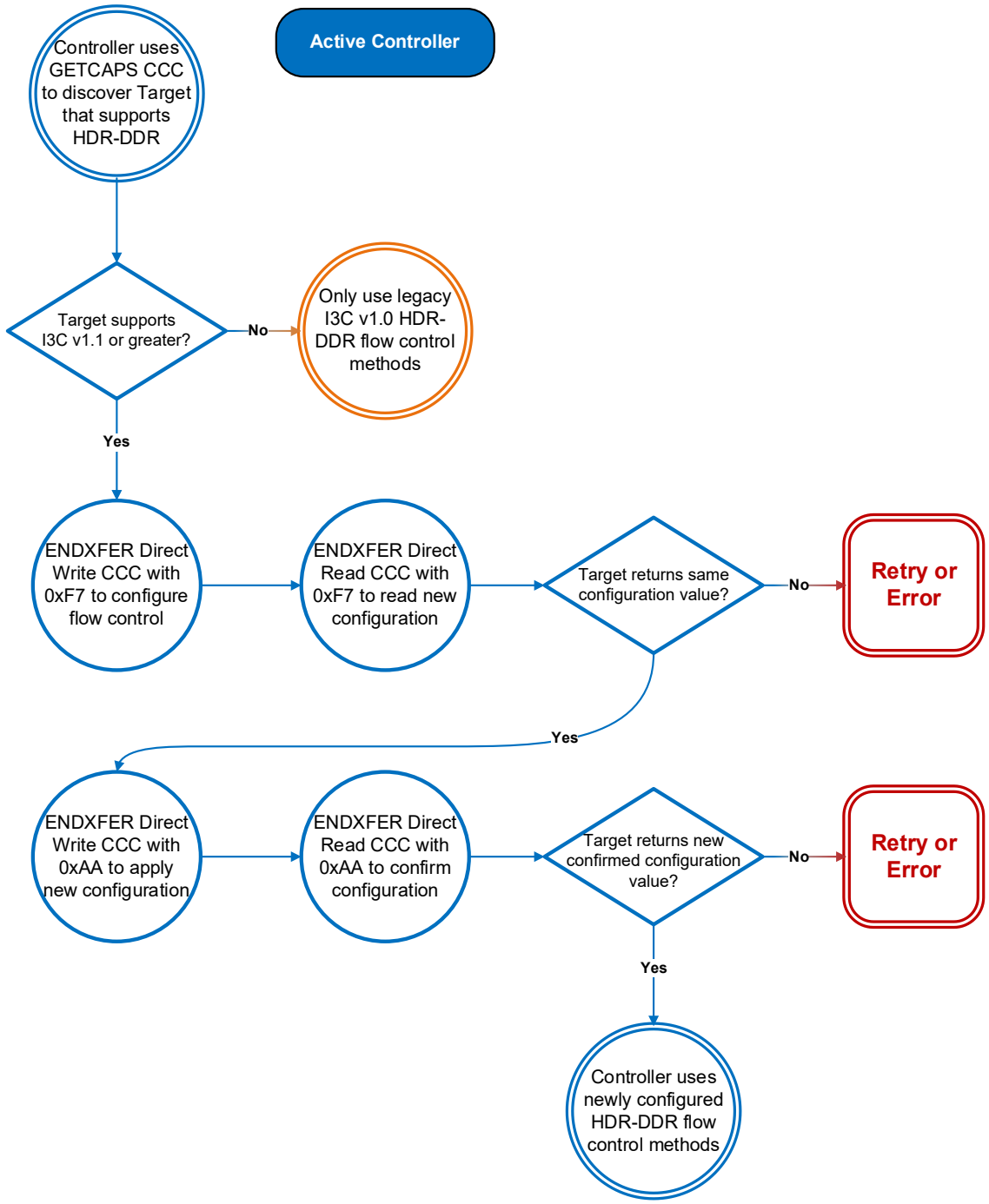


Figure 118 Controller HDR-DDR ENDXFER Steps Flow

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

ENDXFER's Broadcast or Direct sub-command 0xF7 shall be used during Data Transfer Protocol configuration. The sub-command byte 0xF7 is followed by one additional data byte (see ***Table 84***) containing the parameters for the HDR-DDR Data Transfer Early Termination procedure.

Table 84 ENDXFER CCC Additional Data Byte for Sub-Command 0xF7 (HDR-DDR Protocol)

Bits	Description	Values
[7:6]	CRC Word Indicator	**2'b11:** No CRC Word follows Early Termination request **2'b01**: CRC Word follows Early Termination request (Checksum uses CRC-5 algorithm, per ***Section 6.2.3.6***) **Other:** Reserved for future definition by MIPI Alliance
[5]	Enable WRITE Early Termination Request	**1'b0:** Enable **1'b1:** Disable
[4]	Enable ACK/NACK Capability for WRITE Command	**1'b0:** Enable **1'b1:** Disable
[3:0]	Reserved for future use.	–

The Controller starts the set-up procedure by sending the ENDXFER CCC with sub-command 0xF7 and the additional data byte, indicating the desired set of parameters. If the Controller uses the Direct version of ENDXFER, then the RnW bit shall be 1'b0 (WRITE) for the directly addressed Devices.

Then the Controller sends the Direct ENDXFER CCC with sub-command 0xF7. It addresses each distinct recipient of the previous ENDXFER with the RnW bit set to 1'b1 (READ). Note that a READ command cannot be sent to a Group Address. Each addressed Device shall respond with its active additional data byte; if the value is identical to what was sent, then the Controller receives confirmation that the set-up is correct. If the value is different, then the Controller can repeat the ENDXFER CCC with the 0xF7 sub-command, using either the Broadcast version or the Direct version, with RnW set to 1'b0 (WRITE) for the directly addressed Devices.

If the Controller determines that the set-up was successful, then it shall then send the ENDXFER CCC with sub-command 0xAA. If the Direct version of ENDXFER is used, then RnW is set to 1'b0 (WRITE) for the directly addressed Devices and the data is set to 0xAA (i.e., the sub-command value is reused for the data byte value, to facilitate error detection). **See Errata 01, Item 19**

The Controller then checks for correct receipt of the command by using the Direct ENDXFER CCC with sub-command 0xAA, and with RnW set to 1'b1 (READ), for each directly addressed Device (not for a Group Address). Each addressed Device shall respond with a data byte containing the additional data byte pertaining to ENDXFER sub-command 0xF7 that is currently active in its system. If this value is correct, then the Controller shall begin communications using the agreed-upon Data Transfer Early Termination procedure. If the value is not correct, then the Controller shall either abort the Data Transfer Early Termination procedure, or else repeat the set-up process.

The Controller may start a new set-up at any time during the configuration process, or during the running stage, by sending an ENDXFER CCC with sub-command 0xF7 and a new additional data byte. The new set-up overwrites the old set-up.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.4.5 Controller Ends or Continues the Read Transfer

The Controller has the option to either end, or continue, a Read transfer from the Target. The early ending can be done either with or without the CRC Word, as described in this Section.

Note: **See Errata 01, Item 20**

*Per **Table 80**, the Controller's decision to continue the Read Transfer shall determine the Preamble Value of the next HDR-DDR Data Word, whereas the Controller's decision to end the Read Transfer is effectively a Preamble Value of 2'b10 followed by an HDR Restart Pattern or HDR Exit Pattern. See **Figure 115**, **Figure 116**, and **Figure 117**.* **See Errata 01, Item 21**

The procedures for ending the Read Transfer without the CRC Word and with the CRC Word are identical for the first 7 steps, as shown below. The Target shall determine whether to emit the CRC Word based on prior configuration using the ENDXFER CCC, per ***Section 6.2.3.4.4***.

6.2.3.4.5.1 Ending the Read Transfer without CRC Word

To end the DDR Read transfer, the Controller actively drives SDA Low before the C1 falling edge (i.e., while SCL is High), to signal the Target to stop returning data.

Figure 119 illustrates the ending of the Read transfer without the CRC Word, using numbered blue dots that correspond to the numbered steps below.

The significant steps are:

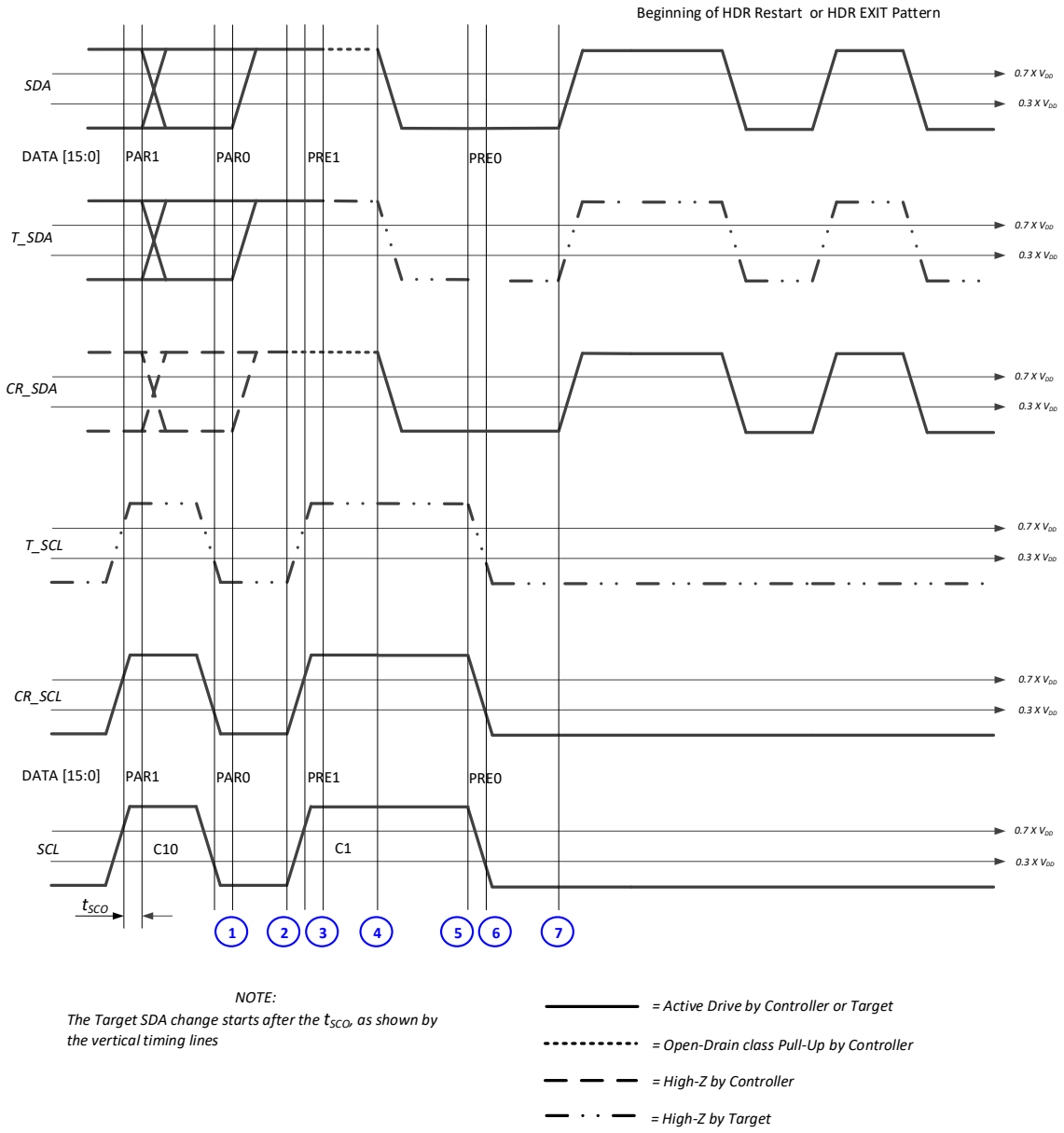
1. After the last parity bit (PAR0), the Target drives SDA High for the next Preamble bit (PRE1) which has the value 1'b1
2. The Controller then:
   A. On SCL, starts the Rising edge of clock pulse C1, and
   B. Enables the Open Drain class Pull-Up structure on SDA
3. Once t_{SCO} elapses after C1's rising edge, the Target then:
   A. Stops actively driving SDA, and
   B. Releases SDA on High-Z
4. After a time period longer than the Target's t_{SCO}, the Controller starts driving SDA Low.

   One of the simplest ways to determine the necessary delay is to use a half cycle of the SCL clock. It could be even the full cycle, which would preserve the phase on the driver's logic block. The resultant SCL pulse is longer than the 50 ns that is required for coexistence on the Bus with Legacy I^2C Devices. The signal waveform is a typical Repeated START; this is acceptable, since in the case of a Mixed Bus it is required to be followed by the HDR Exit Pattern and a STOP.
5. After another delay similar to the one described in step 4, the Controller starts driving SCL Low, starting C1's falling edge
6. The Target finds SDA being driven Low by the Controller, setting the second Preamble bit (PRE0) to 1'b0. Since this means the end of the Read transaction, the Target keeps SDA on High-Z.
7. After another delay similar to the one described in step 4, the Controller starts driving SDA High in order to prepare for either the HDR Restart Pattern or the HDR Exit Pattern.

   At this point, SCL is Low and SDA becomes High (actively driven by the Controller), while the Target has both lines on High-Z

At this point the Target-to-Controller Read data transfer has concluded.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



5135

Figure 119 Controller Ends DDR Read and Provides Either HDR Restart Pattern or HDR Exit Pattern

6.2.3.4.5.2 Ending the Read Transfer with CRC Word

To end the DDR Read transfer, the Controller actively drives SDA Low before the C1 falling edge (i.e., while SCL is High), to signal the Target to stop returning data and return the CRC Word.

Figure 120 shows the signal diagram for the case where the READ transfer ends with the Target providing the CRC Word, with the 4'hC token. The figure uses numbered blue and red dots that correspond to the numbered steps below.

The significant steps are:

1. After the last parity bit (PAR0), the Target drives SDA High for the next Preamble bit (PRE1), which must be 1'b1
2. The Controller:
   A. Starts the SCL Rising edge of C1 clock pulse, and
   B. Enables the Open Drain class pull-up structure on SDA
3. Once t_{SCO} elapses after the rising edge of C1, the Target then:
   A. Stops the active drive of SDA, and
   B. Releases SDA on High-Z
4. After a period of time longer than the Target's t_{SCO}, the Controller then starts driving SDA Low. One of the simplest ways to determine the necessary delay is to use a half cycle of the SCL clock. It could be even the full cycle, which would preserve the phase on the driver's logic block. The resultant SCL pulse is longer than the 50 ns required for coexistence with Legacy I^2C Devices. The signal waveform is a typical Repeated START; this is acceptable, since in the case of a Mixed Bus it is required to be followed by the HDR Exit Pattern and a STOP.
5. After another delay similar to that described in step 4, the Controller Starts driving SCL Low, starting the falling edge of C1
6. The Target then:
   A. Finds SDA being driven Low by the Controller, setting the second Preamble bit (PRE0) to 1'b0
   B. Since this condition means the end of the Read transaction, the Target preserves the SDA on High-Z
7. From this point on, the red bordered dots apply. After another delay similar to that descried in step 4, the Controller starts driving SDA High. This can be done even immediately after the falling edge of the C1, since it is under the Controller's control.
8. The Controller has provided the rising edge of the first SCL of the CRC Word, CLK_CRC1. Since SDA was High, the Target assesses the first bit of the CRC token as 1'b1.
9. At the falling edge of CLK_CRC1, the Target reads the second bit of the CRC token as 1'b1, since the Controller was still driving SDA High.
10. The Controller then drives SDA Low
11. At the rising edge of CLK_CRC2, the Target read the third bit of the CRC token as 1'b0, since the Controller was driving SDA Low.
12. After its t_{SCO} the Target starts driving SDA Low, in parallel with the Controller (there is no conflict, both lines are driven Low by the Devices).
13. The Controller then:
    A. Starts driving the falling edge of the CLK_CRC2 Low, and
    B. Releases SDA on High-Z
14. The Target (and the Controller) reads the fourth bit of the CRC token as 1'b0, since the Target was driving SDA Low
15. After a period of time longer than the Target's t_{SCO}, the Target starts driving SDA per the calculated CRC. The Target has had enough time to switch its output to the calculated CRC (almost two SCL clocks).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

At this point, the Read data transfer from the Target to the Controller has entered the CRC-based Ending Procedure.

Note:

*Up until dot 7, the steps are identical to **Figure 119**. The numbered blue dots (steps 1–7) are the same as ending the Read Transfer without the CRC Word, and the numbered red dots (steps 8–15) show the additional steps for ending the Read Transfer with the CRC Word.*

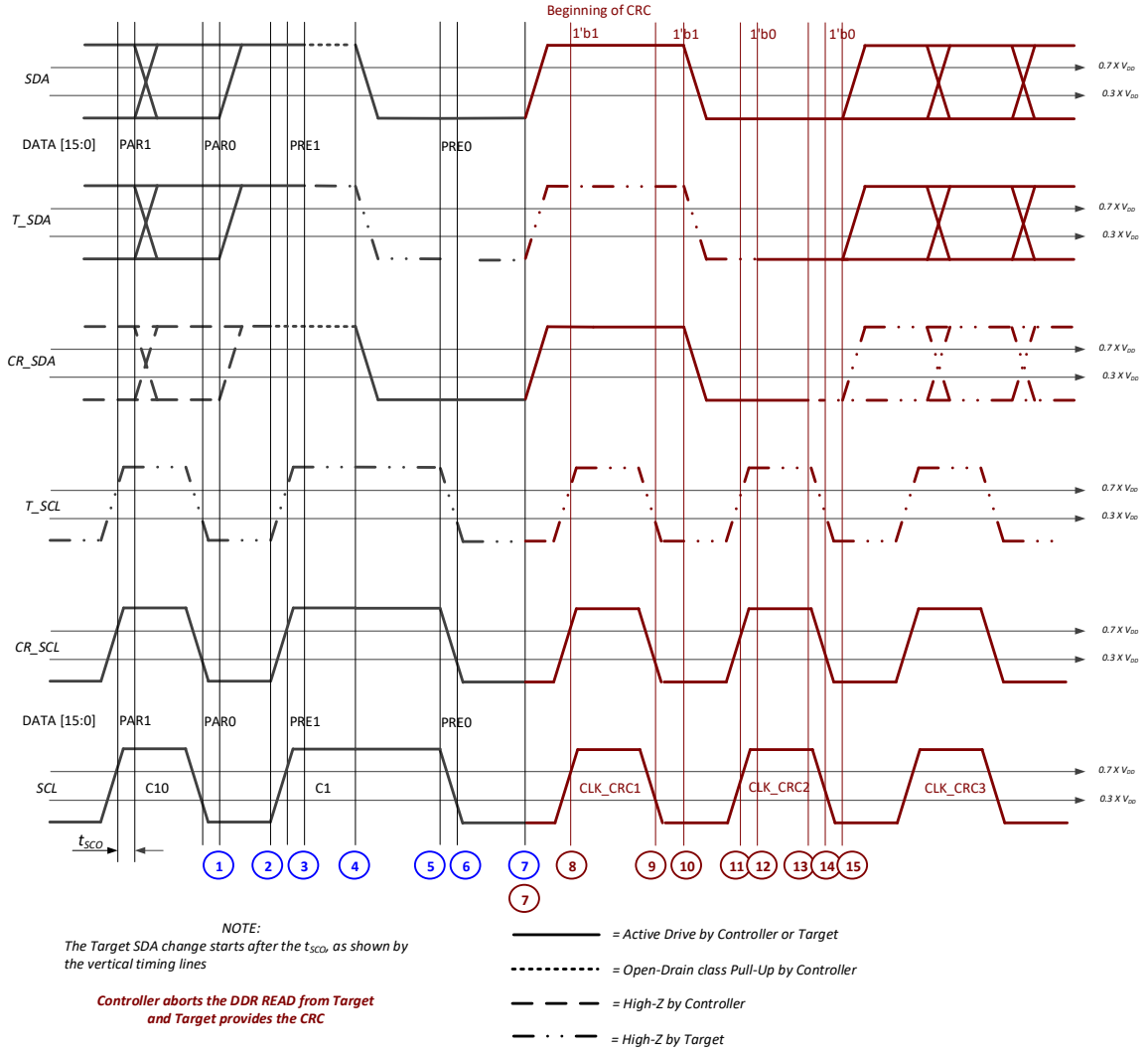


Figure 120 Controller Ends DDR Read and Target Provides CRC

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.4.5.3 Continuing the Read Transfer

To continue the DDR Read data transfer from the Target, the Controller allows SDA to remain High through the C1 falling edge (i.e., while SCL goes from High to Low) at the start of the next HDR-DDR Data Word.

Figure 121 illustrates continuation of the Read data transfer, using numbered blue dots that correspond to the numbered steps below.

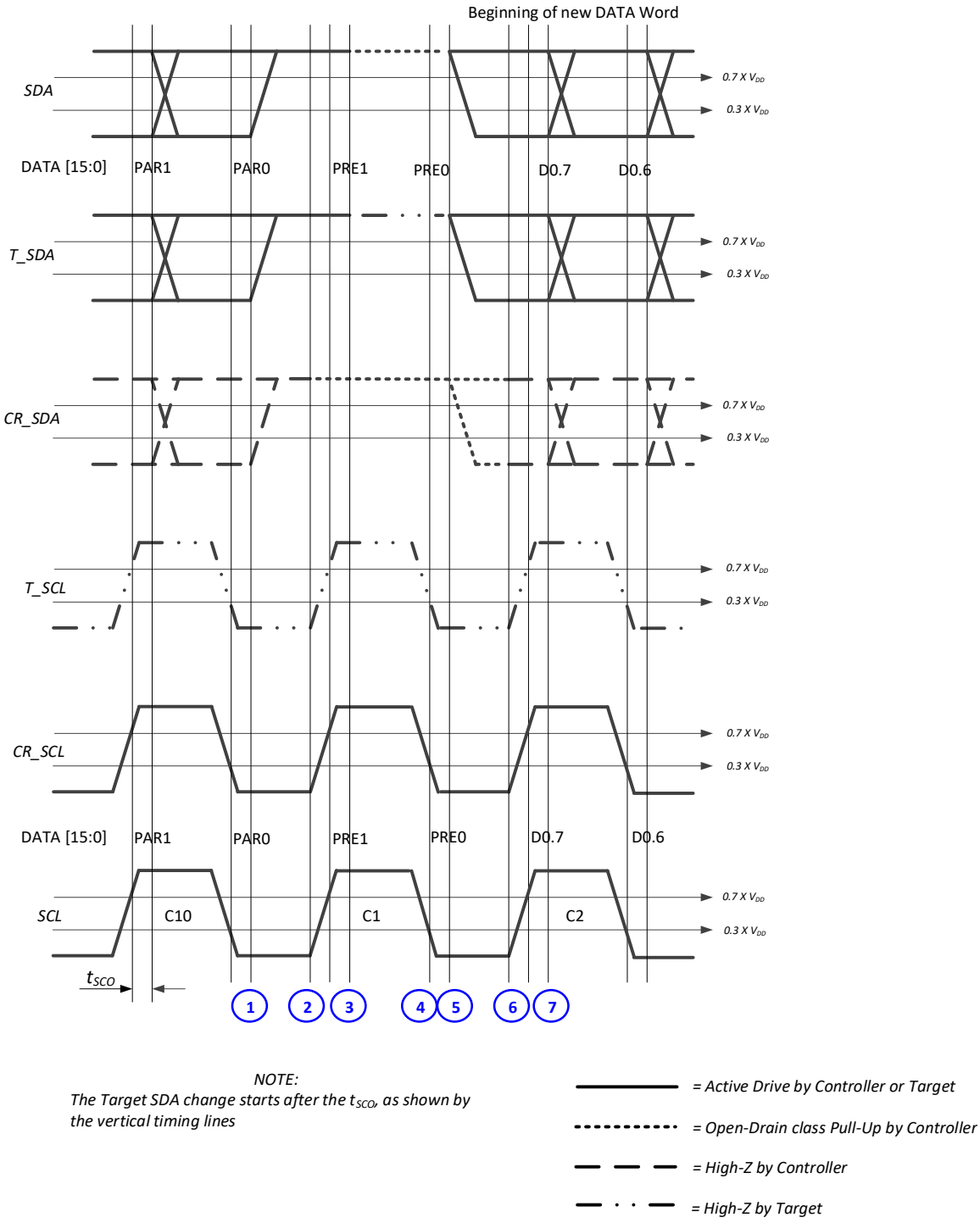
The significant steps are:

1. After the last parity bit (PAR0), the Target drives SDA High for the next Preamble bit (PRE1) which must have the value 1'b1
2. The Controller then:
   A. On SCL, starts the Rising edge of clock pulse C1, and
   B. Enables the Open Drain class Pull-Up structure on SDA
3. Once the Target's t_{SCO} elapses after C1's rising edge, the Target then:
   A. Stops actively driving SDA, and
   B. Releases SDA on High-Z
4. At clock pulse C2's falling edge, the Target finds that SDA is High
5. After the Target's t_{SCO} elapses, the Target then starts actively driving SDA for the payload's first data bit (D0.7)
6. After a suitable delay, the Controller then:
   A. Disables the Open Drain class Pull-Up structure on SDA, and
   B. Sets SDA on High-Z

   This delay shall be greater than or equal to Target's t_{SCO}. The start of C2's rising edge would be a safe moment. Depending upon the Controller's design, a shorter delay could be used.
7. After the Target's t_{SCO}, elapses, the Target starts driving SDA for the payload's second data bit (D0.6)

After step 7, the Target-to-Controller data transfer continues.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



5214

Figure 121 Controller Continues DDR Read from Target

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.4.6 Target Ends or Continues the Write Transfer

The Target has the option to either end, or continue, a Write transfer from the Controller. The early ending can be done either with or without the CRC Word, as detailed in this Section.

Note:

*Per **Table 79**, the Target's decision to continue or end the Write Transfer shall determine the Preamble Value of the next HDR-DDR Data Word. See **Figure 122**, **Figure 123**, and **Figure 124**.*

*The procedures for ending the Read Transfer without the CRC Word and with the CRC Word are identical for the first 6 steps, as shown below. The Controller shall determine whether to emit the CRC Word based on prior agreement with the Target(s) as configured with the ENDXFER CCC, per **Section 6.2.3.4.4**.* **See Errata 01, Item 22**

6.2.3.4.6.1 Ending the Write Transfer without CRC Word

To end the DDR Write transfer, the Target actively drives SDA Low before the C1 falling edge (i.e., while SCL is High), to request the Controller to end the transfer.

Figure 122 illustrates the ending of the Write transfer without the CRC Word, using numbered blue dots that correspond to the numbered steps below.

The significant steps are:

1. After the last parity bit (PAR0), the Controller drives SDA High for the next Preamble bit (PRE1) which must have the value 1'b1
2. The Controller then:
   A. On SCL, starts the rising edge of clock pulse C1, and
   B. Disables the active drive of SDA, and
   C. Enables the Open Drain class Pull-Up structure on SDA
3. After the Target's t_{SCO} elapses, the Target actively drives SDA Low
4. The Controller then:
   A. On SCL, starts the falling edge of clock pulse C1

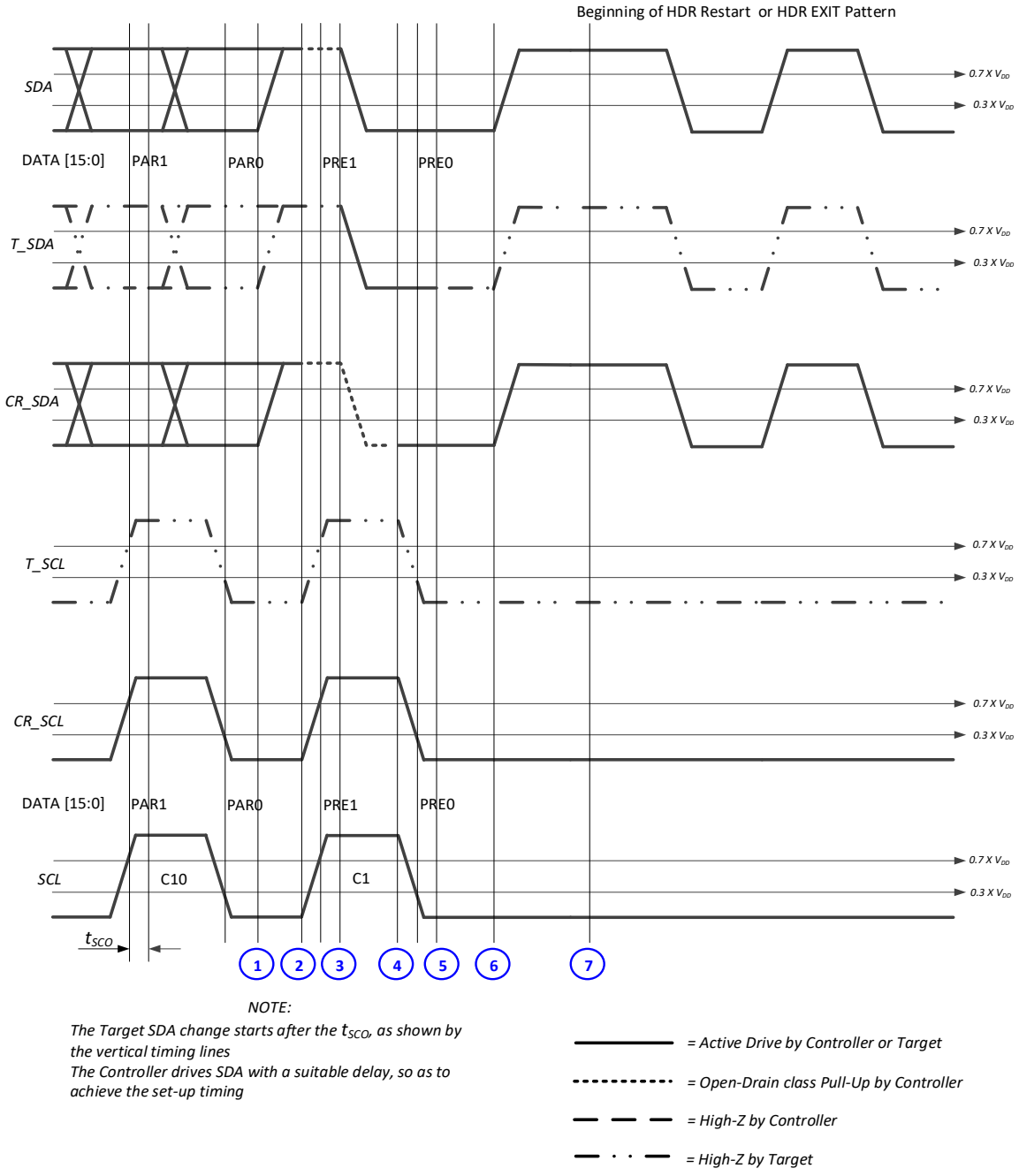
   The Controller knows that SDA was pulled Low by the Target; therefore, PRE0 is 1'b0
   B. The Controller then starts actively driving SDA Low

   At this point both the Controller and the Target are actively driving SDA Low, in parallel
5. Once the Target's t_{SCO} elapses, the Target releases SDA on High-Z
6. After a suitable delay, the Controller then starts actively driving SDA High

   One way to determine the necessary delay is to use one half of an SCL clock cycle.
7. At this point, following another delay similar to the one in step 6:
   A. SDA is High, actively driven by the Controller
   B. SCL is Low, actively driven by the Controller
   C. The Target has both SCL and SDA released on High-Z
   D. The Controller can start either an HDR Restart Pattern, or an HDR Exit Pattern

At this point, the Controller-to-Target data transfer has concluded.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



5250

Figure 122 Target Requests DDR Write Termination and Controller Provides HDR Restart Pattern or HDR Exit Pattern

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.4.6.2 Ending the Write Transfer with CRC Word

To end the DDR Write transfer, the Target actively drives SDA Low before the C1 falling edge (i.e., while SCL is High), to request the Controller to end the transfer.

Figure 123 shows the signal diagram for the case where the WRITE transfer ends with the Controller providing the CRC, with 4'hC token. The figure uses numbered blue and red dots that correspond to the numbered steps below.

The significant steps are:

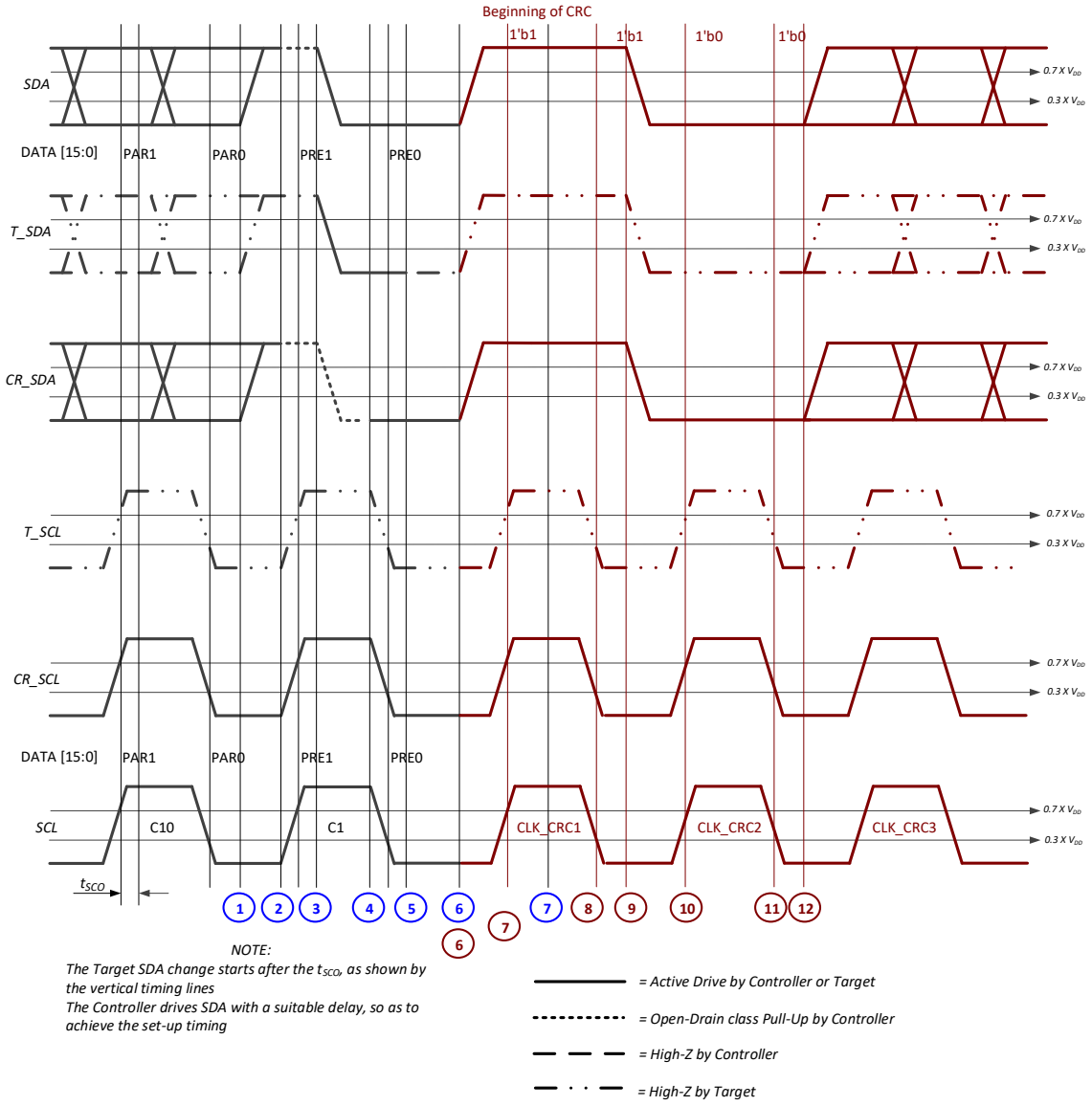
1. After the last parity bit (PAR0), the Controller drives SDA High for the next Preamble bit (PRE1) which must be 1'b1
2. The Controller then:
   A. Starts the rising edge of C1, and
   B. Disables the active drive of SDA, and
   C. Enables the Open Drain class pull-up structure on SDA
3. After the Target's t_{SCO} elapses, the Target actively drives SDA Low
4. The Controller tan:
   A. Starts the falling edge of C1, and
   B. Knows that, since the Target pulled SDA Low, the PRE0 bit has value 1'b0; and
   C. Starts actively driving SDA Low, in parallel with the Target
5. After the Target's t_{SCO} elapses, the Target releases SDA on High-Z
6. From this point on, the red bordered dots apply. After a suitable delay, the Controller starts actively driving SDA High.
7. The Controller has provided the rising edge of the first SCL of the CRC Word (CLK_CRC1). Since SDA was High, the Target reads the first bit of the CRC token as 1'b1.
8. At the falling edge of CLK_CRC1, the Target reads the second bit of the CRC token as 1'b1, since the Controller was still driving SDA High.
9. The Controller then drives SDA Low
10. At the rising edge of CLK_CRC2, the Target reads the third bit of the CRC token as 1'b0, since the Controller drove SDA Low.
11. At the falling edge of CLK_CRC2, the Target (and the Controller) reads the fourth bit of the CRC token as 1'b0, since the Target drove SDA Low.
12. The Controller then starts driving the SDA per the calculated CRC.

At this point, the WRITE data transfer from Controller to Target has entered the CRC-based Ending Procedure.

Note:

*Up until dot 6, the transactions are identical to **Figure 122**. The numbered blue dots (steps 1–6) are the same as ending the Write Transfer without the CRC Word, and the numbered red dots (steps 7–12) show the additional steps for ending the Write Transfer with the CRC Word.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



5287

Figure 123 Target Requests DDR Write Termination and Controller Provides CRC

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.4.6.3 Continuing the Write Transfer

To continue the DDR Write data transfer, the Target allows SDA to remain High through the C1 falling edge (i.e., while SCL goes from High to Low) at the start of the next HDR-DDR Data Word.

Figure 124 illustrates continuation of the Write data transfer, using numbered blue dots that correspond to the numbered steps below.

The significant steps are:

1. After the last parity bit (PAR0), the Controller drives SDA High for the next Preamble bit (PRE1) which must have the value 1'b1
2. The Controller then:
   A. On SCL, starts the rising edge of clock pulse C1, and
   B. Disables the active drive of SDA, and
   C. Enables the Open Drain class Pull-Up structure on SDA
3. The Controller then:
   A. On SCL, starts the falling edge of clock pulse C1

   The Controller knows that the Target left SDA High; therefore, PRE0 is 1'b1
   B. The Controller then starts actively driving SDA High
4. After a suitable delay, the Controller then starts actively driving SDA either High or Low, as required for the payload's first data bit (D0.7).

   One simple way to determine the delay is to use one half of an SCL clock cycle.
5. On SCL, the Controller then starts the rising edge of clock pulse C2.

   At this point, the Target has both SCL and SDA on High-Z
6. The Target then registers the value of the payload's first data bit (D0.7)
7. The Target then registers the value of the payload's second data bit (D0.6)

After step 7, the Controller-to-Target data transfer continues.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

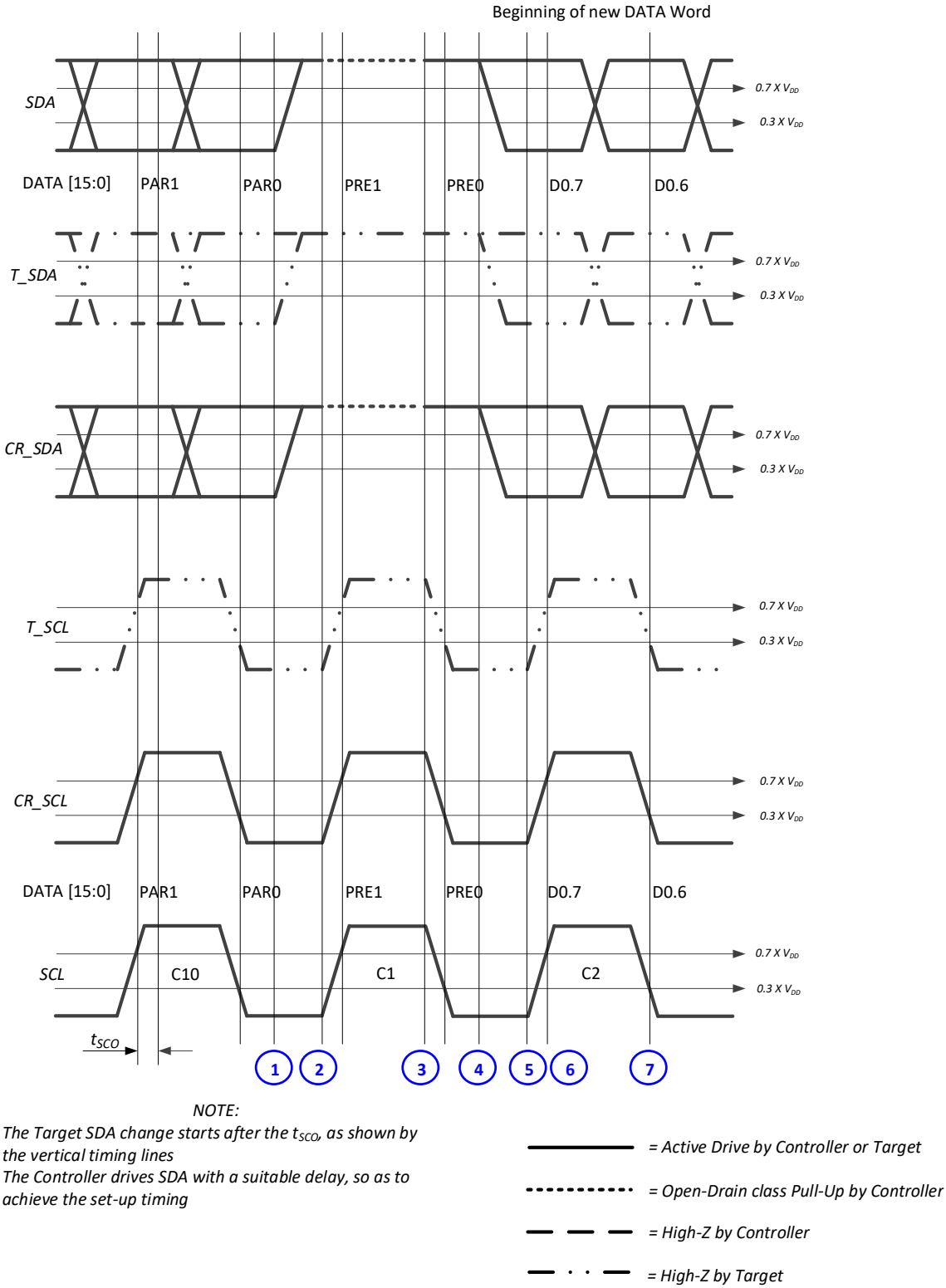


Figure 124 Target Continues the DDR Write From Controller

6.2.3.4.7 Controller Clock Stalling in HDR-DDR Mode

In HDR DDR Mode, the I3C Controller may stall the I3C Bus during the SCL Low period at specific transitory conditions.

Controller Clock stalling may be necessary and could be applied:

- Because the absolute or relative timing of a Message to a specific Target, or to all Targets, needs to be carefully controlled. Clock Stalling provides the Controller with fine-grained data timing control.
- Because the I3C Controller needs to internally synchronize data. This may be due to parts of the Controller's system waking up in response to data, to changing state, or to otherwise needing time during a transaction in cases of Overflow/Underrun.

As detailed in the following sub-sections, during HDR DDR communication the most common points at which the I3C Controller may stall (i.e., may hold SCL Low) are:

- First data bit following a DDR Write Command Acknowledge
- First token bit following a DDR write early termination by a Target
- Write data transfer, PRE1 of data word or CRC word
- Read data transfer, PRE1 of data word or CRC word
- HDR Restart Pattern or HDR Exit Pattern first SDA falling edge generation

6.2.3.4.7.1 First Data Bit Following a DDR Write Command Acknowledge

The DDR Write Acknowledge requires the Target Devices to send an ACK/NACK for a DDR command. The ACK is done during the SCL High period, which could require the Target Devices to analyze and prepare a response to the command. This ACK will last through the SCL High period, and the Target Devices will release the SDA line during SCL Low period. This could be dependent on the t_{sco} and setup time, which could require a longer time for a correct ACK/NACK analysis and data transfer. During this time, the Controller may stall the Clock to analyze the ACK/NACK response of the Target Devices. See ***Figure 125***.

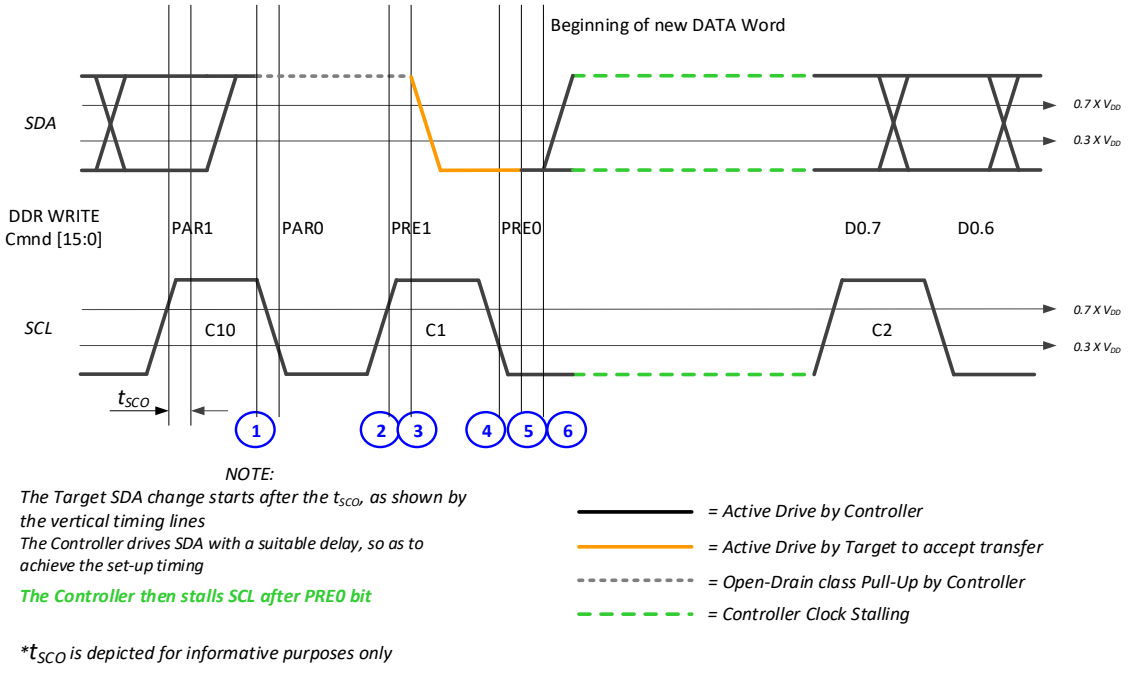


Figure 125 SCL Stalling After PRE0 Bit on Write ACK

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.4.7.2 First Token Bit Following a DDR Write Early Termination by a Target

During an early termination, the Target drives the SDA Low during the PRE0 to terminate a Controller write operation. This will require the Controller to abort sending data and send a CRC or proceed with an HDR Restart. This requires the Controller to take time to process the write termination request. The Controller may stall the Clock and proceed with the appropriate course of action based on the configuration that was set during the ENDXFER setup. See ***Figure 126***.

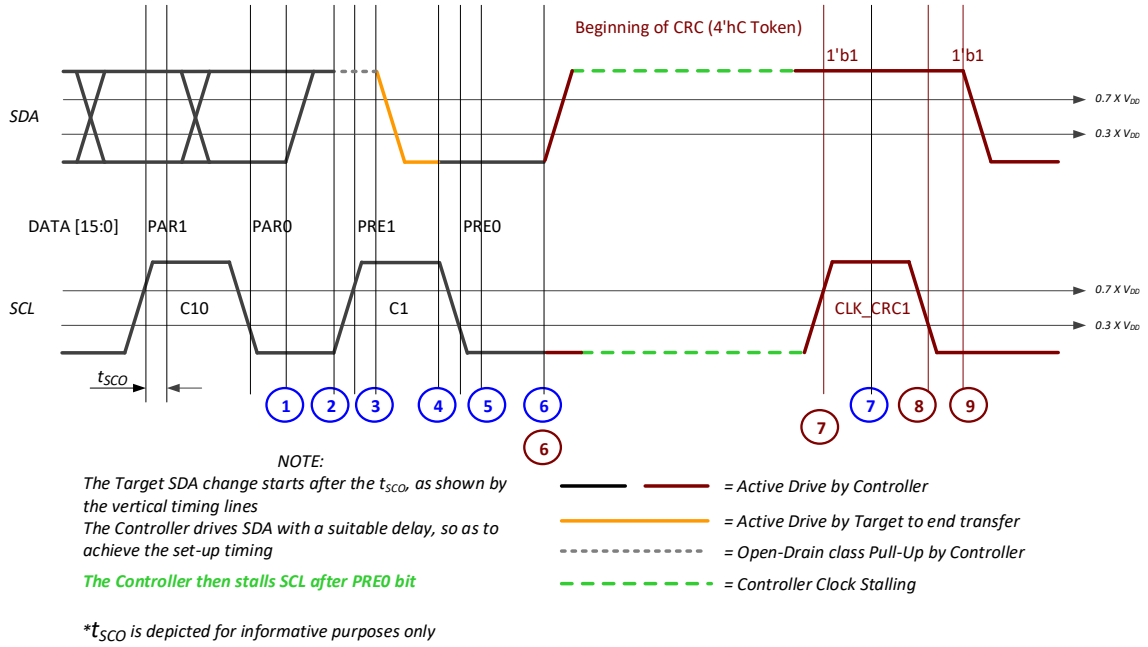


Figure 126 SCL Stalling After PRE0 Bit on Write Abort

6.2.3.4.7.3 Write Data Transfer, PRE1 of Data Word or CRC Word

The Controller may stall SCL between data bytes of an I3C HDR-DDR Write Data transfer, by Stalling the clock during the Low period of the PRE1 bit, in case of Underrun situations. See ***Figure 127***.

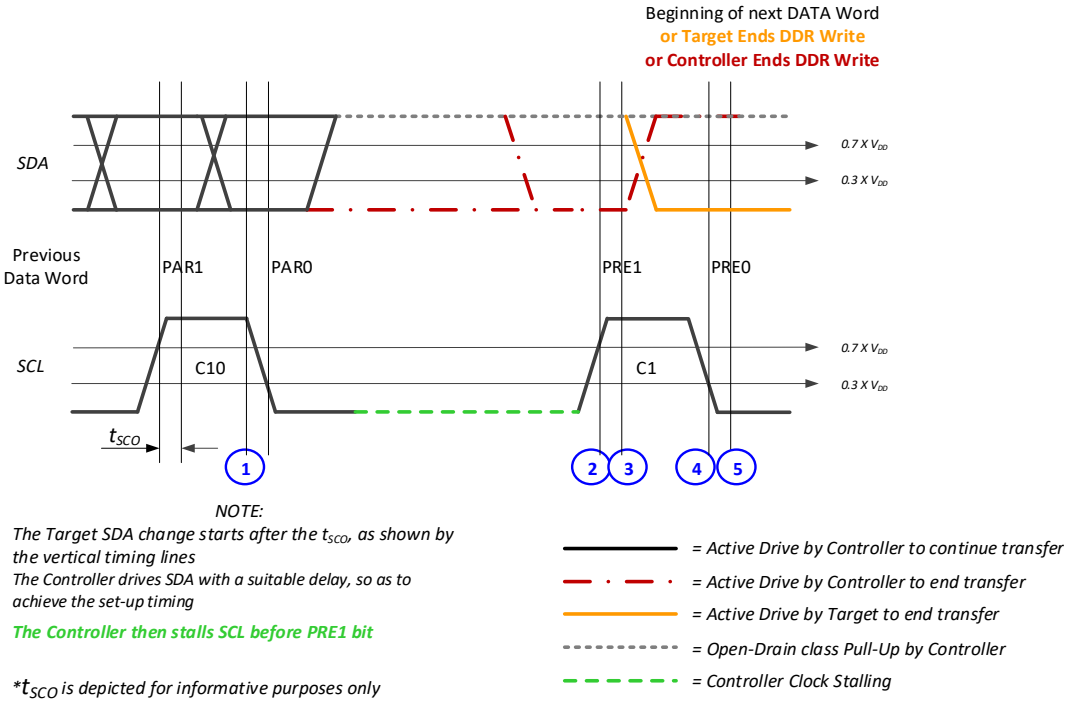


Figure 127 SCL Stalling in PRE1 Bit of DDR Write

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.4.7.4 Read Data Transfer, PRE1 of Data Word or CRC Word

The Controller may stall SCL between data bytes of an I3C HDR-DDR Read Data transfer, or before terminating, by Stalling the clock during the Low period of the PRE1 bit, in case of Overflow situations. See ***Figure 128***.

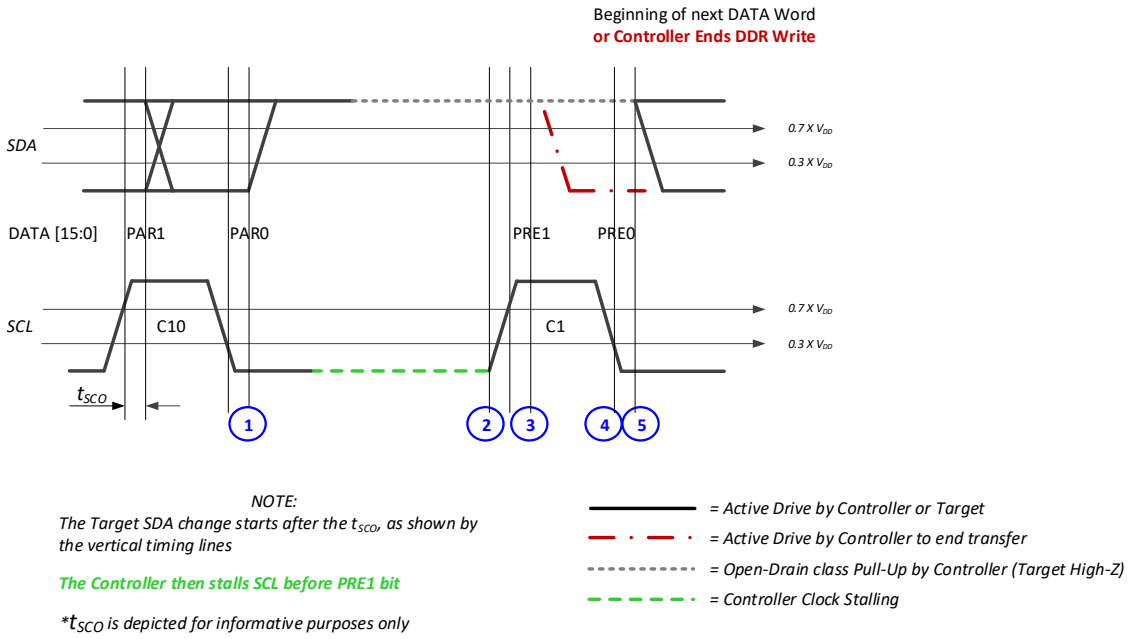


Figure 128 SCL Stalling on PRE1 Bit of DDR Read Transfer

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.4.7.5 HDR Restart Pattern or HDR Exit Pattern First SDA Falling Edge Generation

The Controller may stall SCL Low before the first SDA falling edge of an HDR Restart Pattern or HDR Exit Pattern after CRC data, or after an early termination. See ***Figure 129***.

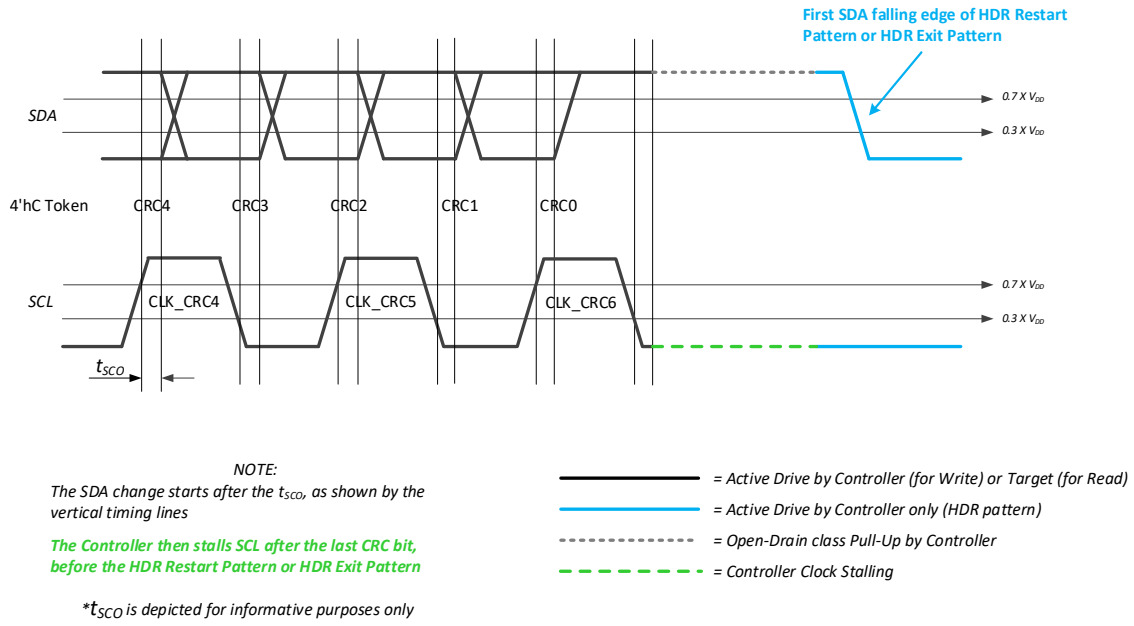


Figure 129 SCL Stalling on HDR Restart Pattern / HDR Exit Pattern After CRC

The Controller may stall SCL before generating the first SDA falling edge of an HDR Restart Pattern or HDR Exit Pattern following an early termination condition, in case of Underrun / Overflow situations linked to unavailability of command FIFO or response FIFO. See ***Figure 130***.

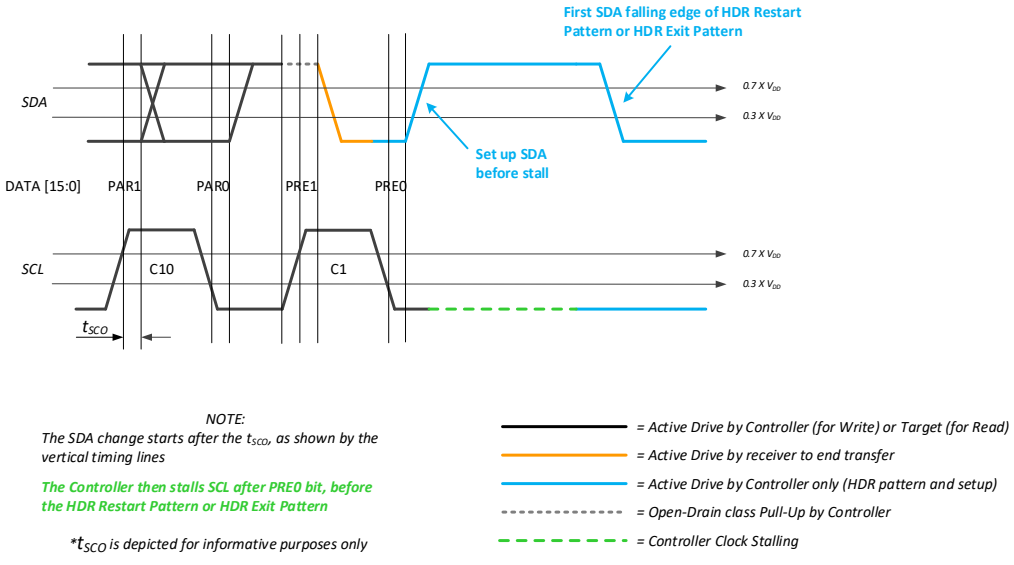


Figure 130 SCL Stalling After Write/Read Abort

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.2.3.5 HDR-DDR Error Detection

Four error types are defined for HDR-DDR Mode:

1. Framing – the Preamble 2-bits before Command and Data shall be valid values per ***Table 79***. This supports a positional error detection mechanism:
   - A Command Word shall always follow the Enter HDR CCC and HDR Restart Pattern, and shall never appear in any other position. It is an error condition for a Command Word to appear in any other position, or to be missing where expected.
   - A Data Word shall always follow either a Command Word or another Data Word, and shall never appear in any other position. It is an error condition for a Data Word to appear in any other position, or to be missing where expected.
   - A single CRC Word shall always follow the last Data Word for a Command, such that it ends the Message. As a result, a CRC Word shall always be followed by either the HDR Restart Pattern or the HDR Exit Pattern. It is error condition for a CRC Word to appear in any other position, or to be missing where expected.
   - The first nibble of a valid CRC shall contain the allowed value 4'hC. Any other value in the first nibble should be considered a framing error.
2. Parity encoding (for transmitters) parity checking (for receivers) shall be performed on all Command Words and Data Words. Mismatched parity is an error condition.
3. CRC-5 encoding (for transmitters) and CRC-5 checking (for receivers) shall be performed on all payload bits for Command Words and Data Words. Mismatched CRC is an Error.
4. NACK by the Target on a Read command is not a normal behavior (ACK is normal). The Controller may choose to treat a NACK on a Read command as a possible framing error, and take the actions detailed below.

Any error detected by the Target should result solely in waiting for HDR Exit Pattern or HDR Restart Pattern to be safe.

If the Controller detects an error in a Read, then it shall issue SCL clocks until 19 SCL clocks (38 bits) have been seen with SDA High. Because a continuous High run of this length cannot occur in a valid data transmission, this ensures that the Target has released the line. The Controller shall emit either an HDR Exit Pattern or an HDR Restart Pattern.

Note:

The Controller can choose to treat the NACK of a Read command as a possible line error, and therefore use the same approach in order to ensure that the Target is not driving the Bus.

If an error occurs in the RnW Bit during a Private Read Transfer, then the Write Data from the Controller might conflict with the Read Data from the Target. Both the Controller and the Target should always monitor the Data that they each transmit. If the Controller or Target performs such monitoring, and if the monitored Data differs from the Data that the Controller or Target intended to send (except for Data transferred during a Dynamic Address Arbitration procedure), then this shall be considered an error. After detecting this error, both the Controller and the Target should stop the transmission; if they do so, then the Target shall wait for an HDR Exit Pattern or HDR Restart Pattern. After the Controller sends the HDR Exit Pattern or HDR Restart Pattern, then it may retry the transmission.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.2.3.6 HDR-DDR CRC-5 Algorithm

The CRC-5 checksum is computed on a complete Message, including the Command Word and all Data Words.

- For a Command Word, the CRC-5 checksum is computed based on the value of the 16-bit payload, including:
  - The Read vs. Write bit
  - The Command Value
  - The Target Address, and
  - The lowest-order bit (Parity adjustment).
- For Data Words, the CRC-5 checksum is computed based on all Data 16-bit values transmitted for that Command.

The CRC-5 value is initialized to 0x1F. The CRC-5 polynomial is the same as used in ***[USB01]***:

$$CRC5 = X^5 + X^2 + X^0$$

The reason why CRC-5 is adequate is that most errors will be caught by the Parity bits, or by incorrect Preamble bits. The Parity bits will detect runs of one, two, three, or more errors in a row. Any Clock errors not detected by the Parity bits would generally be detected as framing errors. That is, the Preamble would not match the allowed patterns, including 2'b01 leading into the CRC. As a result, any error that could be missed by the Parity bits will result in a shift of the CRC polynomial, and therefore will be detected as a CRC error. There would have to be two separated errors in the same Word, with either both errors occurring in even-index bits or with both errors occurring in odd-index bits, to produce correct Parity bits. However these two errors would not correct the CRC, and as a result would still be detected. Even with two such error pairs in two different Words, the probability of producing the same CRC value is very low, even though the CRC-5 checksum has only 32 possible values. Since any higher error density would render the I3C Bus generally unusable, it can be concluded that the protection that CRC-5 provides is sufficient for the use cases addressed by this Specification.

CRC-5 is specified in ***[USB01]***. Two versions of the logic for computing the CRC-5 checksum are shown below:

- First, for one bit at a time,
- Second, for the more practical situation of two bits at a time (i.e., on SCL falling edge).

```
// CRC5 builds from each bit as it comes in, using the registered SDA_r.
// The next_CRC5 is registered on each cycle as CRC5[4:0].
// But note that it would have to be on each SCL edge, so two
// alternating buffers would be needed.
assign next_crc0 = SDA_r ^ CRC5[4];
assign next_CRC5[4:0] = {CRC5[3:2], next_crc0^CRC5[1],
                         CRC5[0],   next_crc0};

// CRC5 builds from 2 bits at a time (registered d_rise_r and live d_fall),
// on falling edge of SCL (by convention - could be on rising instead).
// The d_fall could be d_fall_r, but then order must match input order
// of the data bits.
// The CRC therefore processes the two bits by combining the 2 bit shifts.
// The next_CRC5 is registered on each cycle as CRC5[4:0]
assign next_CRC5 = {CRC5[2], d_rise_r^CRC5[4]^CRC5[1], d_fall^CRC5[3]^CRC5[0],
                    d_rise_r^CRC5[4], d_fall^CRC5[3]};
```


Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.3 F002: HDR Ternary Modes

This Section defines I3C Optional Feature F002: HDR Ternary Modes.

Table 85 Optional Feature 007 Status See Errata 01, Item 23

Optional Feature ID	F002
Optional Feature Name	HDR Ternary Modes
Optional Feature Version	1
Optional Feature specification was last updated in I3C Spec Version	I3C v1.2
Optional Feature is not compatible with other Optional Features	F008 D2D Tunneling
Optional Feature is included in I3C Basic	No

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4 F003: HDR-BT Mode

This Section defines I3C Optional Feature F003: HDR-BT Mode.

Table 86 Optional Feature 003 Status

Optional Feature ID	F003
Optional Feature Name	HDR-BT Mode
Optional Feature Version	1
Optional Feature specification was last updated in I3C Spec Version	I3C Basic v1.2 **See Errata 01, Item 24**
Optional Feature is not compatible with other Optional Features	F008 D2D Tunneling
Optional Feature is included in I3C Basic	Yes

Changes in this Version

For detailed changes in this version, see the "diff" file between I3C v1.1.1 and I3C v1.2 in the MIPI Specification Repository ***[MIPI09]***.

6.4.1 Scope and Purpose (Informative)

The scope of ***Section 6.4*** is limited to fully specifying I3C Optional Feature F003: HDR-BT Mode. An I3C Device that implements F003 is required to implement all **shall** statements in ***Section 6.4.3***, and can optionally implement any combination of **may** statements in ***Section 6.4.3***. In addition, MIPI Alliance recommends that manufacturers follow all **should** statements in ***Section 6.4.3***.

6.4.2 Technical Overview (Informative)

HDR-BT is provided to give the highest possible throughput using a Clock-and-Data transmission model. Based on the number of SDA Lanes (4 for Quad Lane configuration, 2 for Dual Lane, or 1 for Single Lane), HDR-BT provides 8x, 4x, and 2x the raw SDR rate, respectively, at the same clock frequency. Real data rates are about 8.7x, 4.36x, and 2.18x SDR real data rates, and 4.8x, 2.4x, and 1.2x HDR-DDR real data rates. In particular, at 12.5 MHz HDR-BT can move real data at 97 Mbps (12 Mbytes/second) over Quad Lane (SDA[3:0]), and ½ and ¼ that respectively for the Dual Lane and Single Lane configurations.

HDR-BT is an I3C HDR mode like the others. As a result, it is entered in the standard way (see the CCC **ENTHDR3**, ***Section 4.3.7.3.9***) and exited via the HDR Exit Pattern. As with the other HDR modes, Messages while in the HDR-BT mode are separated using the HDR Restart pattern.

HDR-BT also brings other key features in addition to speed:

- CRC-16 (and optionally CRC-32) for data integrity.

  Significantly, HDR-BT also has the receiver verify to the sender whether the transmitted CRC value for each Message Frame actually matched the CRC value calculated from the received data. This allows the transmitter to know with certainty whether it can release the buffer and proceed, or will have to re-send the data (whether immediately or scheduled for a later time).
- Allows the option for the Target to drive the SCL Clock during Read, when the Target supports it. This can be useful with long transmission lines, as there is no Clock-out to Data-in roundtrip time.
- Decouples Command and Control from Data, allowing division of labor between the Peripheral and system:
  - Fully supportive of classical I^2C and I3C protocol models, including write of Index or Command byte (or bytes) preceding the Data (for write) or the Read request (for read)

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

- Also appropriate for writing and reading file equivalents, where the command is the offset, using up to 32-bit offsets
- Uses a wide data block model to facilitate more natural processing for higher rate data.
  - In particular, because of the use of wide data blocks, HDR-BT is well suited to direct mapping to internal SRAM (e.g., 32-bit, 64-bit, etc.), wide internal buses (e.g., AXI, OCP, etc.), and Encryption/decryption cipher modes with inherent block sizes (e.g., 64-bit, 128-bit).
  - Although HDR-BT moves data in 32-byte data blocks, the last data block can "be ragged" and transmit fewer than 32 data bytes (i.e., 2, 4, 6, … to 32 bytes).
- HDR-BT is consistent no matter how many Data Lanes are used (1, 2, or 4 SDA Lanes).
  - Further, HDR-BT is intended to allow a fully-registered interface on both sides – no combinatorial decisions required

While HDR-BT is primarily aimed at Multi-Lane Buses, it can also be used in a Single-Lane context, providing about 1.2x the real data rate of HDR-DDR. This provides the option of using a single high-rate protocol regardless of the number of data Lanes needed. The rest of this Section will focus on Dual Lane and Quad Lane configurations, but HDR-BT protocol in Single Lane configuration works exactly the same – it just takes 2x or 4x more clocks to move the same amount of data.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3 Technical Specification

The transport form of HDR-BT is purely byte-oriented. Unlike other I3C forms, the protocol has no extra bits, just a pure byte stream, even for control and data bytes.

The wire form of HDR-BT is a DDR style, meaning that the 1, 2, or 4 SDA Lanes change on each clock edge and are read on the following edge, using a Setup time model (vs. Hold):

- For the Controller on Write, which Drives the clock and data, this means the SDA(s) change(s) only after the SCL has completely transitioned (as with normal I3C).
- For the Target on Read, when only driving the SDA(s) but not SCL, the SDA(s) change(s) only after the SCL change has been received/detected.
- For the Target on Read, when the Target is driving both clock and data, this means the SDA(s) change(s) only after the SCL has completely transitioned.

Signal diagrams for starting HDR-BT Mode transfers are shown in ***Figure 131*** and ***Figure 132***. ***Figure 131*** shows the HDR-BT Header Block (see ***Section 6.4.3.3.1***) after ENTHDR3, and ***Figure 132*** shows the HDR-BT Header Block after the HDR Restart Pattern.

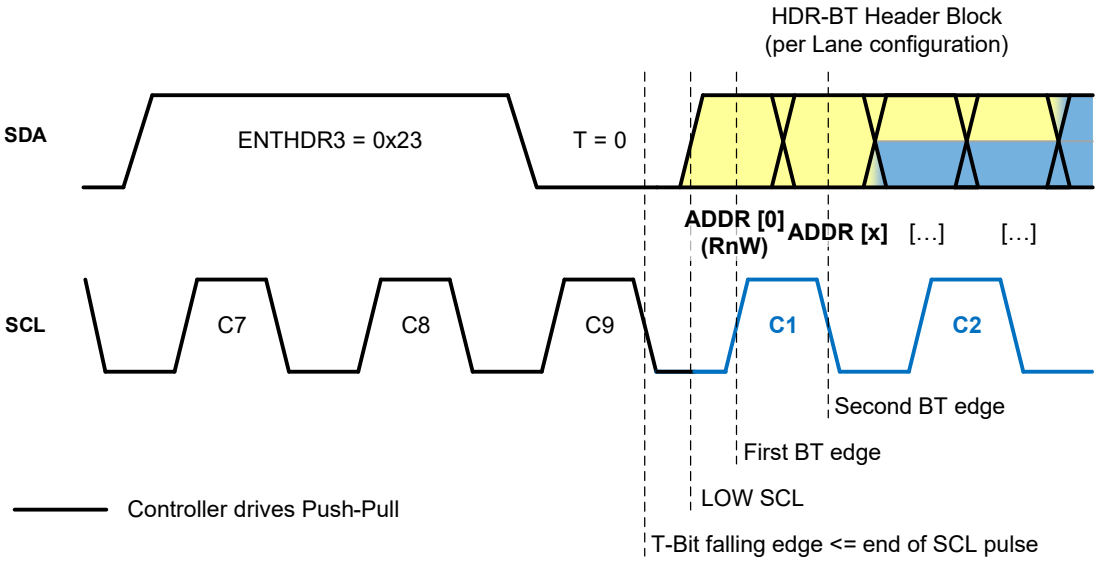


Figure 131 HDR-BT Header Block After ENTHDR3

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

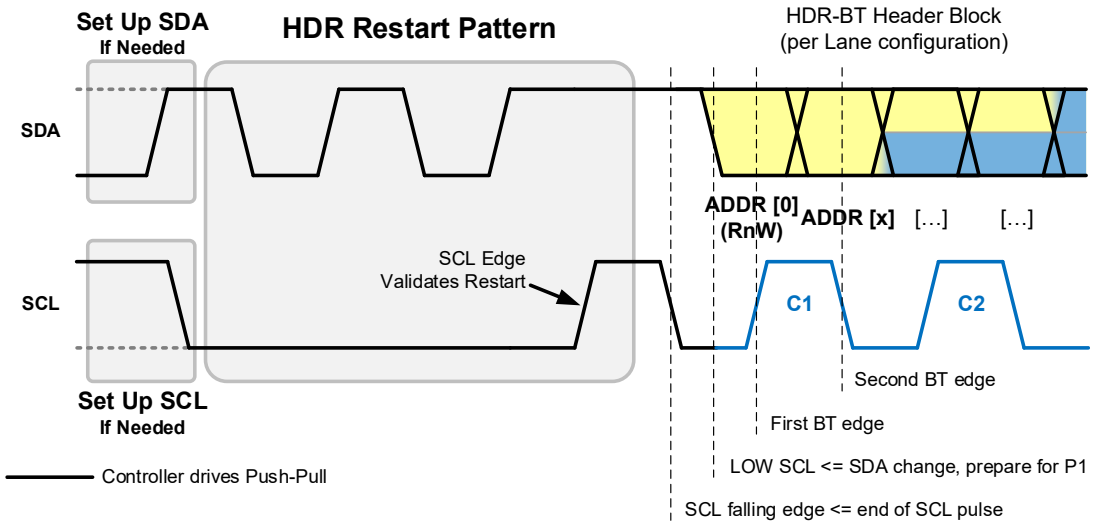


Figure 132 HDR-BT Header Block After HDR Restart Pattern

Note:

*Both **Figure 131** and **Figure 132** are derived from the edge-triggering examples shown in **Section 6.1.4**. In both cases, the Controller positions SDA[0] (i.e., SDA) after the falling SCL edge, and before the SCL rising edge labeled "C1". This bit shall be read by the addressed Target (or Group) as Bit[0] of the Address byte, which is the "RnW" bit for the transfer (per **Section 6.4.3.3**).*

*The HDR-BT Header Block is always sent according to the Lane configuration of the currently configured HDR-BT Data Transfer Coding for Multi-Lane (see **Section 6.7.3.5.3.1**) which is a global setting for the I3C Bus, using the defined Bit Packing for Data Bytes and Transition Bytes.*

Figure 133 illustrates a typical HDR-BT Mode Frame with two HDR-BT transfers and associated data:

- I3C START or Repeated START
- I3C CCC to Enter HDR-BT Mode
- After entering HDR-BT Mode, there is one HDR-BT transfer. This can take either of two forms:
  1. A write or read transfer with data, which starts with an HDR-BT Header Block, followed by one or more HDR-BT Data Blocks, and then an HDR-BT CRC Block; or
  2. A write or read transfer **without** data which only uses an HDR-BT Header Block (i.e., no HDR-BT Data Blocks and no HDR-BT CRC Block).
- Then an HDR Restart Pattern, followed by another HDR-BT transfer with data, having a similar flow.
- Finally, HDR-BT Mode is ended via the HDR Exit Pattern followed by I3C STOP.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

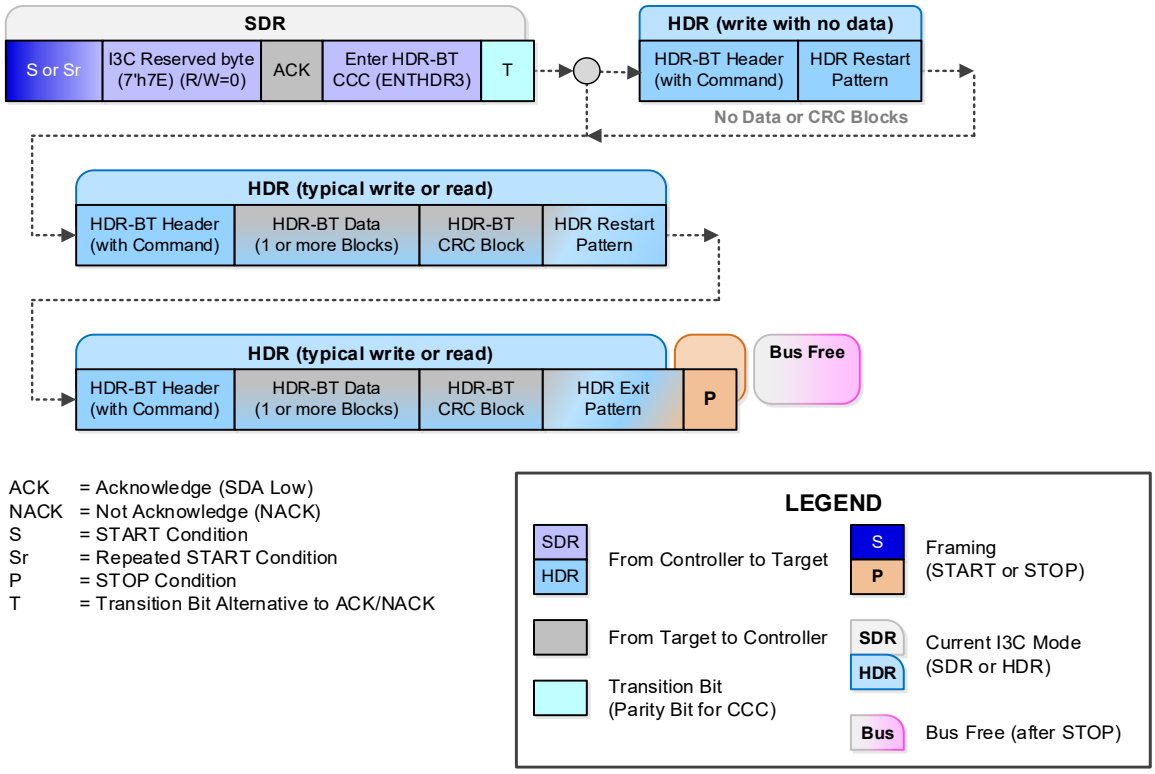


Figure 133 Typical HDR-BT Mode Frame

Note:

*Although **Figure 133** shows several examples of HDR-BT transfers in a particular order, in actual practice there is no need for these transfers to follow this order. See **Section 6.4.3.3** for more details on the HDR-BT structured protocol elements that comprise transfers in HDR-BT Mode. Additionally, the figure does not show CCCs that can be sent in HDR-BT Mode using the defined flows (see **Section 6.4.3.4**).*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.1 SDA Lane Bit Packing: Data Bytes (Commands, Data, CRC)

On the SDA[x] Lanes, HDR-BT always packs bytes of Data in the same way, based on number of Lanes as shown in ***Table 87*** through ***Table 89***.

Note that Transition control packing is different, per ***Section 6.4.3.2***.

Note:

Unlike SDR Mode, the bits are transmitted starting with LSb and progressing to MSb.

Table 87 Data Byte Bit Packing: Single Lane

SDA Lane	Clock 0		Clock 1		Clock 2		Clock 3	
SDA[0]	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7

Table 88 Data Byte Bit Packing: Dual Lane

SDA Lane	Clock 0		Clock 1	
SDA[0]	Bit0	Bit2	Bit4	Bit6
SDA[1]	Bit1	Bit3	Bit5	Bit7

Table 89 Data Byte Bit Packing: Quad Lane

SDA Lane	Clock 0	
SDA[0]	Bit0	Bit4
SDA[1]	Bit1	Bit5
SDA[2]	Bit2	Bit6
SDA[3]	Bit3	Bit7

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.2 SDA Lane Bit Packing: Transition Bytes

HDR-BT always packs bytes of Transition content, whether pure Transition or Transition and Control, in a consistent way. In particular, the SDA[0] Lane always follows the I3C "Park1,High-Z" convention for the first two half-clocks of the byte. As a result, SDA[0] is always 1 (Parked) for the rising SCL edge, and then treated as High-Z for the falling SCL edge, to enable flow control.

To keep things simple, the convention is to treat the "Park1" and "High-Z" as Bit[0] and Bit[1] of the byte, with the low-level logic handling the repositioning based on number of Lanes as shown in ***Table 90*** through ***Table 92***. That is, the byte received will have 2'b11 in Bits[1:0] when the High-Z was not driven Low, with the remaining bits representing the content.

When compared with the bit packing for data bytes, the transition bytes change as follows:

- **For Single Lane: No changes, since all Bits are sent in order.**
- **For Dual Lane:** Bit[2] is sent where Bit[1] would typically be sent for a data byte (i.e., the rising edge of SDA[1] for Clock 0).
- **For Quad Lane:** Bit[4] is sent where Bit[1] would typically be sent for a data byte (i.e., the rising edge of SDA[1] for Clock 0).

See Errata 01, Item 25

Table 90 Transition Byte Bit Packing: Single Lane

SDA Lane	Clock 0		Clock 1		Clock 2		Clock 3	
SDA[0]	Park1	High-Z	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7

Note:
*Compared to a data byte, Bits[7:2] are unchanged. See **Table 87** for comparison.*

Table 91 Transition Byte Bit Packing: Dual Lane with Bit2 Swizzle

SDA Lane	Clock 0		Clock 1	
SDA[0]	Park1	High-Z	Bit4	Bit6
SDA[1]	Bit2 *(See Note 1)*	Bit3	Bit5	Bit7

See Errata 01, Item 25

Note:
1. *Bit2 takes the typical place for Bit1 (i.e., for a data byte), since the High-Z bit on SDA[0] is always Bit1 for a transition byte.*
2. *Compared to a data byte, Bits[7:3] are unchanged. See **Table 88** for comparison.*

Table 92 Transition Byte Bit Packing: Quad Lane with Bit4 Swizzle

SDA Lane	Clock 0	
SDA[0]	Park1	High-Z
SDA[1]	Bit4 *(See Note 1)*	Bit5
SDA[2]	Bit2	Bit6
SDA[3]	Bit3	Bit7

Note:
1. *Bit4 takes the typical place for Bit1 (i.e., for a data byte), since the High-Z bit on SDA[0] is always Bit1 for a transition byte.*
2. *Compared to a data byte, Bits[3:2] and Bits[7:5] are unchanged. See **Table 88** for comparison.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.3 HDR-BT Mode Structured Protocol Elements

The main structuring of the HDR-BT mode is broken into 3 top-level units:

Header Block 7 bytes	**N * Data Blocks** Each block is 32 bytes data + 1 byte control	**CRC-16** or **CRC-32** 6 bytes
	Target may emit SCL if Read	

Where:

- The **Header** indicates the Target (or Group) and the Direction for the transfer, as well as 4 bytes optionally associated with command/index/offset and/or expected length.

  This follows the normal I^2C and I3C protocol of the first byte(s) on Write being the index/command, and Write preceding Read containing the index/command.

Note:

The actual interpretation is a contract between Controller and Target.

- **Data** is 0 or more 32-byte Data Blocks. The *processed* length of the last block may be ragged.

  Data Blocks use 33 bytes for each 32 bytes of real data, with one byte used for transition and control.

- **CRC** shall be either a CRC-16 or a CRC-32 verification at the end of the Message, occupying 6 bytes. The selection between CRC-16 and CRC-32 shall be made by the Controller, and shall be indicated in the Header.
  - If this is a write without data (e.g., a Broadcast CCC without a data payload), then the Controller shall not send any Data Blocks and shall not send the CRC Block.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.3.1 Header Block

The Header Block shall be 7 bytes in length, formed as 6 bytes + 1 Transition byte:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Address	**Cmd0**	**Cmd1**	**Cmd2**	**Cmd3**	**Control**	**Transition**

Where:

- **Address** shall be **either**:
  - The 7-bit I3C Dynamic Address of the Target to be addressed for this transfer in Bits[7:1], with Bit[0] = 1'b0 for a Write, or = 1'b1 for a Read; **or**
  - A valid 7-bit Group Address (if assigned and supported) in Bits[7:1], with Bit[0] set to 1'b0 (as this case is always a Write); **or**
  - The I3C Broadcast Address (i.e., 7'h7E) in Bits[7:1], which indicates framing for CCC transmission in HDR-BT Mode (per ***Section 6.4.3.4***).
- **Cmd0** to **Cmd3** may be used as the indicator to the Target (or Group) as to the meaning/context of the data:
  - If **Address** is a Dynamic Address or Group Address (i.e., is not 7'h7E), then this transfer is a generic Write or Read transfer.

    For a generic Write or Read transfer, fields **Cmd0** through **Cmd3** would typically be used as with standard I3C SDR protocols, for command/index/offset and length (actual or maximum), but the exact meaning is a contract between the Controller and the Target (or all Targets in an assigned Group).
  - If **Address** is 7'h7E (as defined above), then per ***Section 6.4.3.4*** this is a CCC.
    - If this is the first HDR-BT Header Block with Address 7'h7E, then **Cmd0** shall be the CCC Code, **Cmd1** shall be the optional Defining Byte, and the Controller starts the CCC modality (see ***Section 6.4.3.4.1***).
    - Subsequent HDR-BT Header Blocks shall comprise CCC flows, until the Controller exits the CCC modality by performing a valid HDR-BT CCC End Procedure per ***Section 6.4.3.4.7***.

      Note:

      *Per **Table 16**, only some CCCs may be used in HDR-BT Mode.*
- **Control** is a byte and shall be formatted thus:
  - **Bit[0]** = 1'b0 for a Write, or = 1'b1 for a Read. It shall match **Address** Bit[0].
  - **Bit[1]** = 1'b0 if CRC-16 is used, or = 1'b1 if CRC-32 is used.
    - The Controller shall not select CRC-32 if the addressed Target (or Group) does not support CRC-32.
    - The Target shall not accept (i.e., shall NACK) a Message indicating CRC-32 if it does not support CRC-32. (If **Address** indicates a Group Address, then all Targets assigned to that Group Address must support CRC-32.)
  - **Bit[2]:**
    - For a Read, Bit[2] = 1'b0 if the Controller transmits SCL, or = 1'b1 if the Target transmits SCL.

      The Controller shall not select “Target transmits SCL” unless the Target supports it. The Target shall not accept (i.e., shall NACK) a Message with Bit[2] set to 1'b1 if it does not actually support driving SCL.
    - For a Write, Bit[2] is Reserved and shall be set to 1'b0.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

- **Bit[3]** = 1'b0 for a normal Message, or = 1'b1 for a continuation of a Direct CCC (GET or SET).
- **Bit[4]** shall be set to 1'b1 for **all** HDR-BT Transfers, per the new HDR-BT framing.

Note:

In version 1.1.1 of the I3C Specification, Bit[4] was reserved (i.e., was always set to 1'b0) and this indicated the older framing for HDR-BT transfers. Starting with version 1.2 of the I3C Specification, Bit[4] set to 1'b1 now indicates the new HDR-BT framing.

The new HDR-BT framing can be summarized thus:

- *Previously, Broadcast CCCs used bytes **Cmd1** through **Cmd3** in the Indicator Block to hold the first data bytes of the CCC. Bit[4] set to 1'b1 now indicates that Broadcast CCCs will no longer use those bytes as data bytes (per **Section 6.4.3.4.1**). All data bytes for Broadcast CCCs will now be sent in HDR-BT Data Blocks (i.e., after the Indicator Block).*
- *Previously, the HDR-BT CRC Block was defined as required for all HDR-BT transfers. Bit[4] set to 1'b1 now indicates that the CRC Block is no longer required for, and will not be sent with, zero-byte transfers, which includes certain flow elements for CCCs sent in HDR-BT Mode (i.e., Header Blocks with no subsequent Data Blocks).*
- *Previously, the Controller was required to send one HDR-BT CCC Header Block to end the CCC modality for both Broadcast and Direct CCCs sent in HDR-BT Mode. Bit[4] set to 1'b1 now indicates that the Header Block is no longer required for, and will not be sent, when ending the CCC modality for Broadcast CCCs, since the CCC modality will end automatically with the HDR Restart Pattern.*

I3C Targets that interoperate with older I3C Controllers (i.e., those that only support v1.1.1 of the I3C Specification) might also need to support the older framing for HDR-BT transfers, where Bit[4] would always be set to 1'b0.

- **Bit[5]** is Reserved and shall be set to 1'b0.
- **Bits[7:6]** = Parity 0 and Parity 1 on the whole Header (up through this **Control** byte), using the even and odd positioned bits respectively. This follows a model similar to HDR-DDR (see ***Section 6.2.3.2***), using an XOR function:

Bit[6] = Parity of the even index bits, from Header Bytes 0 through 5, and 1'b1:

Address[0] ^ **Address**[2] ^ **Address**[4] ^ **Address**[6] ^
Cmd0[0] ^ **Cmd0**[2] ^ **Cmd0**[4] ^ **Cmd0**[6] ^
Cmd1[0] ^ **Cmd1**[2] ^ **Cmd1**[4] ^ **Cmd1**[6] ^
Cmd2[0] ^ **Cmd2**[2] ^ **Cmd2**[4] ^ **Cmd2**[6] ^
Cmd3[0] ^ **Cmd3**[2] ^ **Cmd3**[4] ^ **Cmd3**[6] ^
Control[0] ^ **Control**[2] ^ **Control**[4] ^ 1'b1

Bit[7] = Parity of the odd index bits, from Header Bytes 0 through 5:

Address[1] ^ **Address**[3] ^ **Address**[5] ^ **Address**[7] ^
Cmd0[1] ^ **Cmd0**[3] ^ **Cmd0**[5] ^ **Cmd0**[7] ^
Cmd1[1] ^ **Cmd1**[3] ^ **Cmd1**[5] ^ **Cmd1**[7] ^
Cmd2[1] ^ **Cmd2**[3] ^ **Cmd2**[5] ^ **Cmd2**[7] ^
Cmd3[1] ^ **Cmd3**[3] ^ **Cmd3**[5] ^ **Cmd3**[7] ^
Control[1] ^ **Control**[3] ^ **Control**[5]

Note:

*The Controller shall compute the Parity for all other bits in the Header Block, including any reserved fields; but not for any fields in the following **Transition** byte, which would not yet have been sent on the SDA Lane(s) by the time the **Control** byte was sent. If the received*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Parity bits do not match the computed Parity of the preceding field, then the Target should not acknowledge (i.e., should NACK) the HDR-BT Header Block.

- **Transition** occupies one byte, and shall use the SDA[0] Line for the first 2 half-clocks, and shall follow the I3C "Park1,High-Z" approach for Bus transition and acceptance (see ***Section 6.4.3.2***).
  - **Bits[1:0]:**
    - Bit[0] as SDA[0]'s first half-clock shall be Parked High by the Controller
    - Bit[1] as SDA[0]'s second half-clock shall be High-Z by Controller; the Target (or all Targets in the Group) shall either drive it Low in order to accept the Write or Read, or else leave it undriven to not accept (see ***Section 6.4.3.7***).

Note:

If there is a Read request, and if the Target sources the clock, then the Controller shall also release the clock during the second half-clock period if the Target is accepting (i.e., if SDA[0] is driven to zero), so that the Target drives the next edge.

The SCL handoff follows the convention that if the Target is accepting, then it shall drive/hold the SCL line High (i.e., the same as the Controller) during this half-clock, so that they overlap.

If the Target does not drive the clock within t_{BT_HO} (i.e., the HDR-BT handoff period) after accepting, then the Controller shall regain control.

  - **Bit[2]:**

For a Read:

- Bit[2] = 1'b0 indicates that the Target is permitted to use the Data Block Delay mechanism if it needs extra time to prepare data to return for the Read transfer. See ***Section 6.4.3.3.4*** for more details.

The Controller shall not use Bit[2] = 1'b0 if it already allowed, or previously configured, another Target to terminate a long transfer (per ***Section 6.4.3.3.4***).

- Bit[2] = 1'b1 indicates that the Target must not use Data Block Delay for this read transfer.

For a Write:

- Bit[2] is Reserved and shall be set to 1'b0

  - **Bits[4:3]** are Reserved and shall be set to 2'b00
  - **Bits[7:5]** are Reserved and cannot be used for future purposes. The Controller shall drive all bits to High on the last half-clock of the Header Block, according to the current Multi-Lane Coding:

For Write transfers:

- **For Single Lane:** Bit[7] on SDA[0] shall be driven High by the Controller. The Controller may optionally drive Bits[6:5] High.
- **For Dual Lane:** Bits[7:6] on SDA[1:0] shall be driven High by the Controller. The Controller may optionally drive Bit[5] High.
- **For Quad Lane:** Bits[7:5] on SDA[3:0] shall be driven High by the Controller.

For Read transfers:

- **For Single Lane:** Bit[7] on SDA[0] shall be driven High by the Controller. This allows the Target (i.e., the transmitter) to safely start the **Transition_Control** byte of the first Data Block. The Controller may optionally drive Bits[6:5] High.
- **For Dual-Lane:** Bit[6] on SDA[0] shall be driven High by the Controller, and Bit[7] on SDA[1] shall be Parked High (i.e., driven High and then High-Z) by the Controller. This allows the Target (i.e., the transmitter) to safely start the **Transition_Control** byte of the first Data Block. The Controller may optionally drive Bit[5] High.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

- **For Quad-Lane:** Bits[7:5] on SDA[3:1] shall be Parked High (i.e., shall be driven High and then High-Z) by the Controller. This allows the Target (i.e., the transmitter) to safely start the **Transition_Control** byte of the first Data Block.

Note:

*Previous versions of the I3C Specification defined Bits[7:5] differently, and treated them as reserved (3'b000) always. While these bits are still considered to be reserved and not usable for future purposes, implementers must now ensure that all SDA lines are put into a suitable state for the last half-clock of the Header Block, especially for HDR-BT Read Transfers. This makes it possible for the addressed Target to safely drive the necessary values at the start of the **Transition_Control** byte in the Data Block, i.e., on the next rising clock edge after the end of this **Transition** byte. In many cases it is simpler for the Controller to simply drive Bits[7:5] to High and then Park (i.e., High-Z) the SDA[x:0] lanes after the last half-clock.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.3.2 Data Blocks

Each Data block shall be formed as 33 bytes:

Transition_Control	Data 0 to 31

Where:

- **Transition_Control** occupies one byte, and shall use the SDA[0] line for the first two half-clocks, and follows the I3C "Park1,High-Z" approach for termination by the receiver (see ***Section 6.4.3.2***).
  - **Bits[1:0]:**
    - Bit[0] as SDA[0]'s first half-clock shall be Parked High by the transmitter of the data (i.e., by the Controller for a Write, or by the Target for a Read).
    - Bit[1] as SDA[0]'s second half-clock shall be High-Z'ed by the transmitter of the data (i.e., by the Controller for a Write, or by the Target for a Read).
      - The receiver may drive SDA[0] Low to terminate the transmission, otherwise the receiver leaves it undriven to allow the data to continue normally. If so, then the transmitter shall drive the next bit on SDA[0] in this **Transition_Control** byte to 1'b0. (This does not apply to Quad Lane, due to the Transition Byte framing).

    Note:

    *If terminated, then the transmitter will emit a CRC Block as the next byte (see **Section 6.4.3.7**). Termination takes priority over the Data Block Delay mechanism.*

  - **Bit[2]**:
    - Bit[2] shall be 1'b0 if this is not the Last Data Block, or (for Single Lane only) if the receiver has terminated the HDR-BT transfer (i.e., if SDA[0] was driven Low by the receiver for Bit[1]).
    - Bit[2] shall be 1'b1 if this is the intended Last Data Block to be sent by the transmitter (i.e., immediately before the CRC Block).
  - **Bit[3]** = Parity adjustment, to ensure odd parity of this **Transition_Control** byte bits [2] to [7]. This is not parity of the data. The data validity is handled by the CRC at the end (i.e., in the CRC Block; see ***Section 6.4.3.3.3***).
    - The transmitter shall write a value equal to XOR(1'b1, Bit[2], Bits[7:4]).
    - The receiver shall ensure that the overall parity is odd, i.e., 1'b0 == XOR(1'b1, Bits[7:2]).

    Note:

    *This parity adjustment mechanism is similar to the one used in the HDR-DDR Command Word (see **Section 6.2.3.3** for more details). If the **Transition_Control** parity does not match, or if the "Park1" is not 1'b1, then there is a framing error. The High-Z may be 1'b0 if the other side has terminated or if a separate Target was permitted to terminate and has done so. See **Section 6.4.3.3.4** for details.*

  - If this is the Last Data Block before the CRC Block, then:
    - **Bits[7:4]** = Byte-pairs in Data Block minus 1 (i.e., 0xF if all 32, or 0x0 for 2 bytes)
  - If this is not the Last Data Block, then:
    - **Bits[7:5]** are Reserved and shall be set to 3'd0.
    - **Bit[4]:**

      For a Read:

      - Bit[4] = 1'b0 indicates that the Target (as transmitter) is sending a valid Data Block.
      - Bit[4] = 1'b1 indicates that the Target (as transmitter) is using the Data Block Delay mechanism, to delay sending the next Data Block in this transfer. The Target may only use the Data Block Delay mechanism if the Controller

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

indicated that the Target was permitted to do so (i.e., if it sent the **Transition** byte with Bit[2] = 1'b0 in the Header Block for this Read transfer; see ***Section 6.4.3.3.4*** for more details).

For a Write: Bit[4] shall be set to 1'b0.

- **Data** shall be transmitted from Byte 0 to Byte 31.

Note:

For the Last Data Block, the receiver shall ignore padding Bytes past the ragged length. These extra bytes should be set to 8'd0, but may contain any value.

Note:

It is recommended for the Last Data Block length to match the receiver's natural data width, such as 32 bits or 64 bits to match internal memory or buses. Further, if the data is encrypted, then the encryption algorithm's blocking size should be considered (e.g., 128-bit for AES).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.3.3 CRC Block

The CRC Block is formed as 6 bytes:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Control	**CRC0**	**CRC1**	**CRC2** if CRC-32	**CRC3** if CRC-32	**Transition_ Verify**

Where:

- **Control** is a byte that indicates that this CRC Block is not another Data Block, since Bit[0] is 0 (Low) instead of 1 (i.e., Parked High).

  The CRC Block should be expected, since either the previous Data Block indicated it was the Last Data Block, or else the transfer was terminated (i.e., High-Z driven Low by the receiver).

  The **Control** byte also verifies the CRC type previously selected in the Header Block, and whether the previous Data Block was a Last Data Block, or was Terminated.

  Therefore, the **Control** byte shall contain:

  - **Bits[2:0]** = 3'b000 always.
  - **Bit[3]** = Parity adjustment, to ensure odd parity of this **Control** byte's Bits[7:2]:
    - The transmitter shall write a value equal to XOR(1'b1, Bit[2], Bits[7:4]).
    - •The receiver shall ensure that the overall parity is odd, i.e., 1'b0 == XOR(1'b1, Bits[7:2]).

    Note:

    *This parity adjustment mechanism is similar to the one used in the HDR-DDR Command Word (see **Section 6.2.3.3** for more details). If the parity does not match, or if Bit[0] is not = 1'b0, or if Bit[1] is not = 1'b0, then there is a framing error.*
  - **Bit[4]** = 1'b0 always
  - **Bit[5]** = Same value as the Header Block's **Control** byte, Bit[1]: 1'b0 for CRC-16, or 1'b1 for CRC-32.

    Note:

    If this value does not match the value from the Header bit, then there is an error.
  - **Bit[6]** = 1'b0 if the transaction ended normally, or = 1'b1 if the previous Data Block was terminated (i.e., if its High-Z was driven to 1'b0 by the receiver).

    Note:

    If the value of Bit[6] does not agree with what the receiver believes actually happened, then there is an error. That is, if the receiver did not see the last Data Block terminated (either by itself, or by a separate Device), and the transmitter says it was terminated, then there is an error. Likewise, if the receiver did not see the last Data Block marked as Last, but the transmitter says it ended normally, then there is an error.
  - **Bit[7]** = Reserved and shall be set to 1'b0.
- Bytes **CRC0** to **CRC3** shall contain the value of the specified CRC algorithm, as computed by the transmitter: either CRC-16, or CRC-32 (if supported).

  The transmitter shall calculate the checksum based on the valid contents of all previous Data Blocks that were sent before this CRC Block, using the following requirements:

  - For each Data Block except the Last Data Block, this includes the entire 33-byte contents of the Data Block (i.e., the value in the **Transition_Control** byte, as well as all 32 **Data** bytes).
  - For the Last Data Block, this includes the **Transition_Control** byte, as well as the number of valid **Data** bytes in the Last Data Block (i.e., as indicated by the number of byte-pairs, sent in

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Bits[7:4] of the **Transition_Control** byte). However, any padding bytes in the Last Data Block shall be ignored.

- In either case, all bits in the **Transition_Control** byte shall always be included, even if the receiver terminates the transfer during the **Transition_Control** byte (i.e., by driving SDA[0] to Low). **See Errata 01, Item 26**
  - For terminated transfers, the actual values driven on the SDA[0] Lane shall be used, and the transmitter shall include only the **Transition_Control** byte. Note that the transmitter will start emitting a CRC Block as the next byte after the **Transition_Control** byte, if the receiver terminates the transfer (see ***Section 5.2.4.3.2***).

    In this case, for the purposes of checksum calculation, the transmitter shall not include any subsequent data bytes that it intended to send, but would have been skipped due to a terminated transfer.

The following structured protocol elements shall not be included in the checksum calculation:

- The Header Block shall be ignored.
- If this is a Read transfer, and if the Target (as transmitter) was permitted to use the Data Byte Delay mechanism (see ***Section 5.2.4.3.4***), then the transmitter shall ignore any Delay bytes that it might send in place of valid Data Blocks. Only valid Data Blocks shall be used in the calculation of the checksum.
- The **Control** byte in the CRC Block shall be ignored.

The receiver shall also calculate the checksum based on the same requirements, as it receives each Data Block during the transfer.

If Bit[1] in the Header Block's Control byte is 1'b0, then the transmitter shall use CRC-16:

For CRC-16, the generator polynomial is:

$$X^{16} + X^{15} + X^2 + 1$$

Note:

For shift register implementations, this polynomial is expressed as 0x8005 in big-endian notation, or 0xA001 in little-endian notation.

- The initial value is all 1s (i.e., 0xFFFF).
- The natural bit order of data in HDR-BT Mode is little endian, so Bytes 1 and 2 (fields **CRC0** and **CRC1**) shall contain the calculated checksum:
  - Byte 1 (**CRC0**) shall contain bits[7:0] of the checksum
  - Byte 2 (**CRC1**) shall contain bits[15:8] of the checksum
- Bytes 3 and 4 (fields **CRC2** and **CRC3**) shall contain all 0s.

If Bit[1] in the Header Block's Control byte is 1'b1, then the transmitter shall use CRC-32:

For CRC-32, the generator polynomial is:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

Note:

For shift register implementations, this polynomial is expressed as 0x04C11DB7 in big-endian notation, or 0xEDB88320 in little-endian notation.

- The initial value is all 1s (i.e., 0xFFFFFFFF).
- The natural bit order of data in HDR-BT Mode is little-endian, so Bytes 1 through 4 (fields **CRC0** through **CRC3**) shall contain the calculated checksum, as follows:
  - Byte 1 (**CRC0**) shall contain bits[7:0] of the checksum
  - Byte 2 (**CRC1**) shall contain bits[15:8] of the checksum
  - Byte 3 (**CRC2**) shall contain bits[23:16] of the checksum
  - Byte 4 (**CRC3**) shall contain bits[31:24] of the checksum.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Example CRC Checksum Calculations

Table 93 presents example CRC checksum calculations as a reference. All values are presented using big-endian notation, and the CRC-16 and CRC-32 values are based on the initial value of all 1s. For each example it is assumed that the CRC checksum is calculated from a data message having a single byte.

Table 93 Example CRC Checksum Calculations

Data Byte	CRC-16	CRC-32
0x00	0x40BF	0x2DFD1072
0x91	0xEC7E	0xAAF5B3A0
0xF7	0xC6FE	0x0E2477CD

Note:

*In the examples above, the computed checksum after the single data byte could also be viewed as the state after processing the first byte of a longer message. The example for data byte 0xF7 is relevant, as this matches the expected value of the **Transition_Control** byte for a Last Data Block that contains 32 valid data bytes (see **Section 6.4.3.3.2**).*

- The **Transition_Verify** byte serves two purposes:
  - For a Read, it is used to hand the SDA[x] lines off, back to the Controller; similarly, if the Target is sourcing SCL, then **Transition_Verify** allows handing the SCL sourcing role back as well.
  - For a Read or Write, it allows the receiver (which could be either the Controller or the Target) to verify that the value provided in the CRC field (i.e., sent in fields **CRC0** through **CRC3**) matched the output of the selected CRC algorithm, and therefore that the Message was correctly received. The receiver shall use the same method for computing the CRC checksum as the transmitter, by using the valid contents of all received Data Blocks (as defined above).

    Transition_Verify shall use the SDA[0] line for the first two half-clocks, and shall follow the I3C "Park1,High-Z" approach for completion by the receiver (see ***Section 6.4.3.2***).
  - **Bits[1:0]:**
    - Bit[0] as SDA[0]'s first half-clock shall be Parked High by the transmitter of the data (i.e., by the Controller for a Write, or by the Target for a Read).
    - Bit[1] as SDA[0]'s second half-clock shall be High-Z'ed by the transmitter of the data (i.e., by the Controller for a Write, or by the Target for a Read).

      To confirm that the CRC field value matched the output of the selected CRC algorithm, the receiver shall drive SDA[0] Low. Alternatively, if the CRC did not match, then the receiver shall leave the SDA[0] line undriven (and therefore High) for Bit[1].

      Note:

      If the Target was driving the SCL during a Read, then it shall hand off the SCL on the second half-clock. The Controller shall drive/hold the SCL line High during this second half-clock, matching the Target, and then drive the next SCL clock edge.

      For error handling, if the Target stops driving the Clock for more than t_{BT_FREQ}, then the Controller will take over the clock. This would happen in the event that the Controller has lost framing due to an error.
  - **Bits[4:2]** = 3'b000 always.

    Note:

    In previous versions of this I3C Specification, Bits[4:2] were marked as reserved. However, due to the different bit packing schemes for HDR-BT transition bytes (which depend on the Multi-Lane configuration of the addressed Target or Group for a

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

transfer), these bits cannot be usable for any meaningful purpose by the receiver, and thus may never be used by the receiver to indicate any post-transfer status or subsequent actions.

- **Bits[7:5]:**
  - Reserved for future use by MIPI I3C WG. These bits shall be set to 3'd0.

For HDR-BT Read transfers, the Target shall release SDA[0] after the second half-clock (i.e., Bit[1]). Subsequent actions shall depend on the bit packing scheme, per ***Section 6.4.3.2***, as follows:

- **If the Target was configured to use a bit packing scheme with multiple clock cycles (i.e., Single Lane or Dual Lane):** If the Controller (as receiver) does not verify the transfer and leaves SDA[0] undriven for the second half-clock, then it shall pull SDA[0] Low after Bit[1] (i.e., the following half-clock), unless it loses framing or there is an error.
- **If the Target was configured to use a bit packing scheme with a single clock cycle (i.e., Quad Lane):** If the Controller (as receiver) does not verify the transfer and leaves SDA[0] undriven for the second half-clock, then SDA[0] shall remain High through the end of the **Transition_Verify** byte. After this, the CRC Block ends and the Controller typically drives either the HDR Restart Pattern or the HDR Exit Pattern.

For HDR-BT Read transfers that use additional Data Lanes, the Target shall drive these Data Lanes (e.g., SDA[1:3]) to Low for the first half-clock, and then High-Z these Lanes for the second half-clock (i.e., along with the High-Z of SDA[0]). If the Target was providing SCL clock, then this coincides with the handoff of SCL and SDA[0] to the Controller (as receiver). Unless the Controller encounters a framing issue and must use an error recovery procedure (i.e., without verifying the Read transfer), the Controller shall take ownership of these additional Data Lanes for the second half-clock, and shall drive them Low at the appropriate time to either verify or not verify the transfer.

For HDR-BT Write transfers, the Controller (as transmitter) shall drive SDA[0] to Low as required by the configured bit packing scheme, after providing the receiver (i.e., the addressed Target or Group) an opportunity to verify the transfer for the second half-clock. After this opportunity, the Controller shall control and drive SDA[0] as required by the bit packing scheme to ensure that Bits[4:2] are always 3'd0:

- **If the receiver was configured to use a bit packing scheme with multiple clock cycles (i.e., Single Lane or Dual Lane):** If the receiver does not verify the transfer and leaves SDA[0] undriven (i.e., High) for the second half-clock, then the Controller shall pull SDA[0] Low after Bit[1] (i.e., the following half-clock); and subsequently shall either hold SDA[0] Low through the end of the **Transition_Verify** byte, or optionally (i.e., by prior agreement and configuration) High-Z or drive to High SDA[0] as needed (i.e., for any such reserved Bits[7:5] that might be used) per the bit packing scheme. Following the last clock cycle, the Controller shall pull SDA[0] High before it drives either the HDR Restart Pattern or the HDR Exit Pattern.
- **If the receiver was configured to use a bit packing scheme with a single clock cycle (i.e., Quad Lane):** If the receiver does not verify the transfer and leaves SDA[0] undriven (i.e., High) for the second half-clock, then SDA[0] shall remain High through the end of the **Transition_Verify** byte. After this, the CRC Block ends and the Controller shall drive either the HDR Restart Pattern or the HDR Exit Pattern.

For HDR-BT Write transfers that use additional Data Lanes, the Controller shall drive these Data Lanes (e.g., SDA[1:3]) to Low (i.e., 0 values) for the first half-clock. Subsequently, the Controller shall either hold these Data Lanes Low for the remaining clock cycles of the **Transition_Verify** byte; or optionally (i.e., by prior agreement and configuration) High-Z or drive to High any relevant Data Lanes as needed (i.e., for any such reserved Bits[7:5] that might be used) for the receiver's bit packing scheme.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Note:

*The HDR-BT CRC Block diagrams in **Section 6.4.3.6** show the reserved Bits[7:5] as 3'd0.*

6.4.3.3.4 Data Block Delay Mechanism

The Data Block Delay mechanism is intended to allow Targets to get additional time when returning Read data. It is mainly intended for situations where the Controller provides the clock (i.e., controls SCL for a Read transfer). However, it may still be useful when the Target controls SCL for a Read transfer, because it allows the Controller to terminate the Read transfer instead of requiring the Controller to continue waiting for the Target to provide data (i.e., where the Controller would become stuck for up to the maximum number of delay attempts).

Operational Flow

At the start of each HDR-BT Read transfer, the Controller shall determine whether the Target is permitted to use the Data Block Delay mechanism, and indicate this in Bit[2] of the **Transition** byte of the Header Block: if Bit[2] has a value of 1, then the Target is not permitted to use the Data Block Delay mechanism. However, if Bit[2] has a value of 0, then the Target (as transmitter) is permitted to use the Data Block Delay mechanism (see ***Section 6.4.3.3.1***).

Note:

The remainder of this Section assumes that the Controller indicated that the Target was permitted to use the Data Block Delay mechanism during an HDR-BT Read transfer.

During the Read transfer, as the Target (as transmitter) prepares to send each Data Block except the Last Data Block (per ***Section 6.4.3.3.2***), it may choose either of two options:

Either:

- Send the next Data Block with a full 32 bytes of data,

  First, the Target sends the **Transition_Control** byte, with Bit[4] having a value of 0; then the Target sends the remaining 32 bytes of Data (i.e., Byte 0 to Byte 31). This is a complete Data Block.

Or:

- Delay sending the Data Block, and send a Delay byte in its place.

  The Target sends a single **Transition_Control** byte (per ***Section 5.2.4.3.2***), using the same "Park1,High-Z" mechanism on SDA[0], and the Controller (as receiver) leaves SDA[0] at High-Z (i.e., to continue the transfer). In Bit[4] the Target sends a value of 1, indicating that it is delaying the next Data Block and not providing a full Data Block at this time.

The Target (as transmitter) may send up to the maximum number of Delay bytes (i.e., up to t_{BT_DBD} at each opportunity) before it must either send the next real Data Block (i.e., with a full 32 bytes of data), or send a Last Data Block to end the HDR-BT Read transfer.

Usage Notes

The Controller (as receiver) shall examine the value of Bit[4] to determine whether the first byte is the **Transition_Control** byte (i.e., is the start of a valid Data Block), or is a Delay byte. Upon making this determination, the Controller may decide that it does not wish to continue waiting for delayed Data Blocks and instead wishes to end the transfer, and it may do so using the "Park1,High-Z" mechanism on SDA[0], at any of the Delay bytes (i.e., as it would with a typical Data Block).

Note:

*HDR-BT packs bytes of Transition content using different bit packing schemes, based on the Multi-Lane configuration of the Target that is addressed in the Header Block (per **Section 6.4.3.2**). Since these bit packing schemes have different locations for Bit[4] with respect to the timing and the "Park1,High-Z" mechanism, the knowledge that a received byte is a Delay byte (and not the first valid **Transition_Control** byte in a real Data Block) might be transmitted **after** the Controller's opportunity*

Copyright © 2016–2025 MIPI Alliance, Inc.

to drive SDA[0] Low to terminate the transmission. In QUAD Lane configuration, the Target sends Bit[4] early enough to give the Controller the opportunity to terminate the transmission. However, in DUAL Lane or SINGLE Lane configurations, the Target sends Bit[4] later, so Controller must react afterwards. As a result, the next opportunity for the Controller to use the "Park1,High-Z" mechanism (for DUAL Lane and SINGLE Lane configurations) is at the start of the next Delay byte, or the next real Data Block.

The Target may use the Data Block Delay mechanism repeatedly throughout the HDR-BT Read transfer, before the first Data Block or any subsequent Data Block, as long as Bit[2] of **Transition_Control** is 1'b0 (i.e., is not a Last Data Block) and the Controller has not terminated this transfer (i.e., pulls SDA[0] to Low as part of "Park1,High-Z" for this Data Block).

The Data Block Delay mechanism should not be used when the Controller has activated any other Targets wishing to have the right to terminate the transaction, as they might not be able to track this mechanism properly. That is, when the Controller is Reading from Target A, but has also activated Target B in order to be able to terminate the transaction, then it should indicate that Data Block Delay is not permitted.

If the Controller controls the SCL for a Read transfer, and if the Target is not permitted to use the Data Block Delay mechanism but still cannot provide Data Blocks to safely maintain the HDR-BT Read transfer (i.e., due to internal data readiness), then the Controller might need to slow the SCL (i.e., reduce the data rate) to better match the Target's capability. This either should be arranged by private contract between the Controller and Target, or should be discovered from the Target before the Controller starts the HDR-BT Read transfer.

Note:

The I3C Specification does not define a mechanism for discovering a Target's capability to provide Read data using HDR-BT, or whether it can sustain HDR-BT Read transfers at a given minimum transfer rate. It is assumed that this mechanism could be a private contract between the Controller and the Target, or could be discovered and controlled by other configuration methods.

Alternatively, if the Target is unable to provide the next Data Block that contains a full 32 bytes of data, then it may emit a Last Data Block and end the transaction early.

Note:

The I3C Specification does not define a mechanism for the Target (as transmitter) to explicitly inform the Controller (as receiver) that a Last Data Block was unexpected, in cases where the Target was unable to provide a Data Block that contained a full 32 bytes of data. This may happen if the Target was unable to provide a full 32 bytes of data at a sustained rate due to extended delay conditions (i.e., a prolonged stall from other application logic). The Target may encounter this situation, even when using the Data Block Delay mechanism, if it had already sent the maximum number of Delay bytes before the next Data Block. The Target may also encounter this situation if it was not permitted to use the Data Block Delay mechanism. In either case, the Controller must handle such a situation where the full data message for a Read transfer is shorter than expected, based on the specific I3C content protocol or a private contract between Controller and Target.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.3.5 Performance

The Bus efficiency can be seen in ***Table 94*** using 12.5 MHz; all numbers can be scaled to Freq:

Table 94 Bus Efficiency

Measurement	Raw Rate (Single / Dual / Quad)	Real Data Rate Calculation[1]	Single / Dual / Quad Real Data Rate (Rounded)
Data blocks	25 Mbps / 50 Mbps / 100 Mbps	32 / 33 * freq	24 Mbps / 48 Mbps / 96.97 Mbps
1 KByte Message	" / "	1024 / (1024+12) * 32 / 33 * freq	23.96 Mbps / 47.9 Mbps / 95.8 Mbps
10 KByte Message	" / "	10K / (10K+12) * 32 / 33 * freq	24 Mbps / 48 Mbps / 96.86 Mbps

Note:

*1. The equation for Messages (vs. just data blocks) factors in the header and CRC as overhead. However, to keep it simple, the expression is incorrect by 12*33/32 or 0.375*freq, so about 1 KHz worst case.*

Worst case and average delay when terminating a Message early at 12.5 MHz:

- Single Lane: Worst Case: 10.5 µs, Average: 5.28 µs. Plus 1.92 µs for CRC
- Dual Lane: Worst Case: 5.28 µs, Average: 2.64 µs. Plus 960 ns for CRC
- Quad Lane: Worst Case: 2.64 µs, Average 1.32 µs. Plus 480 ns for CRC

Note:

- *Parity is only lightly used in this protocol. It is mainly used to catch framing errors. The positive Verification at the end (after the CRC) ensures that the transmission was correct in terms of bits and framing.*
- *The Verify model allows the I3C Controller's Host system to operate a TCP/IP style windowing mechanism to avoid the processor having to check too often. This means that the Host can queue up blocks of, e.g., 1 K to 10 K bytes, and then pipeline back the verify successes/failures, allowing the Host to resubmit any that failed, freeing up ones that succeeded.*

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.4.3.4 CCC Transmission in HDR-BT Mode

A special HDR-BT syntax is used for transmitting Common Command Codes in the HDR-BT protocol. The following formats are defined, as specific instantiations of the generic HDR CCC flows (see ***Section 6.1.3***). The Broadcast CCC flow is shown in ***Figure 134***, and the Direct CCC flow is shown in ***Figure 135***.

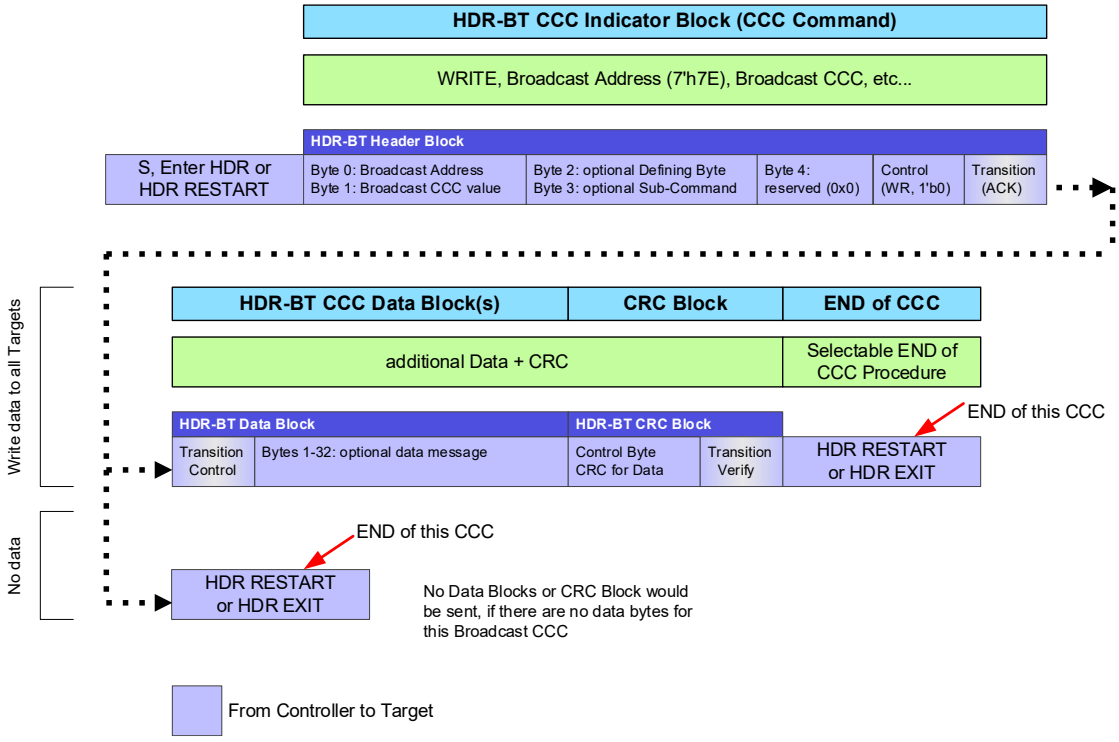
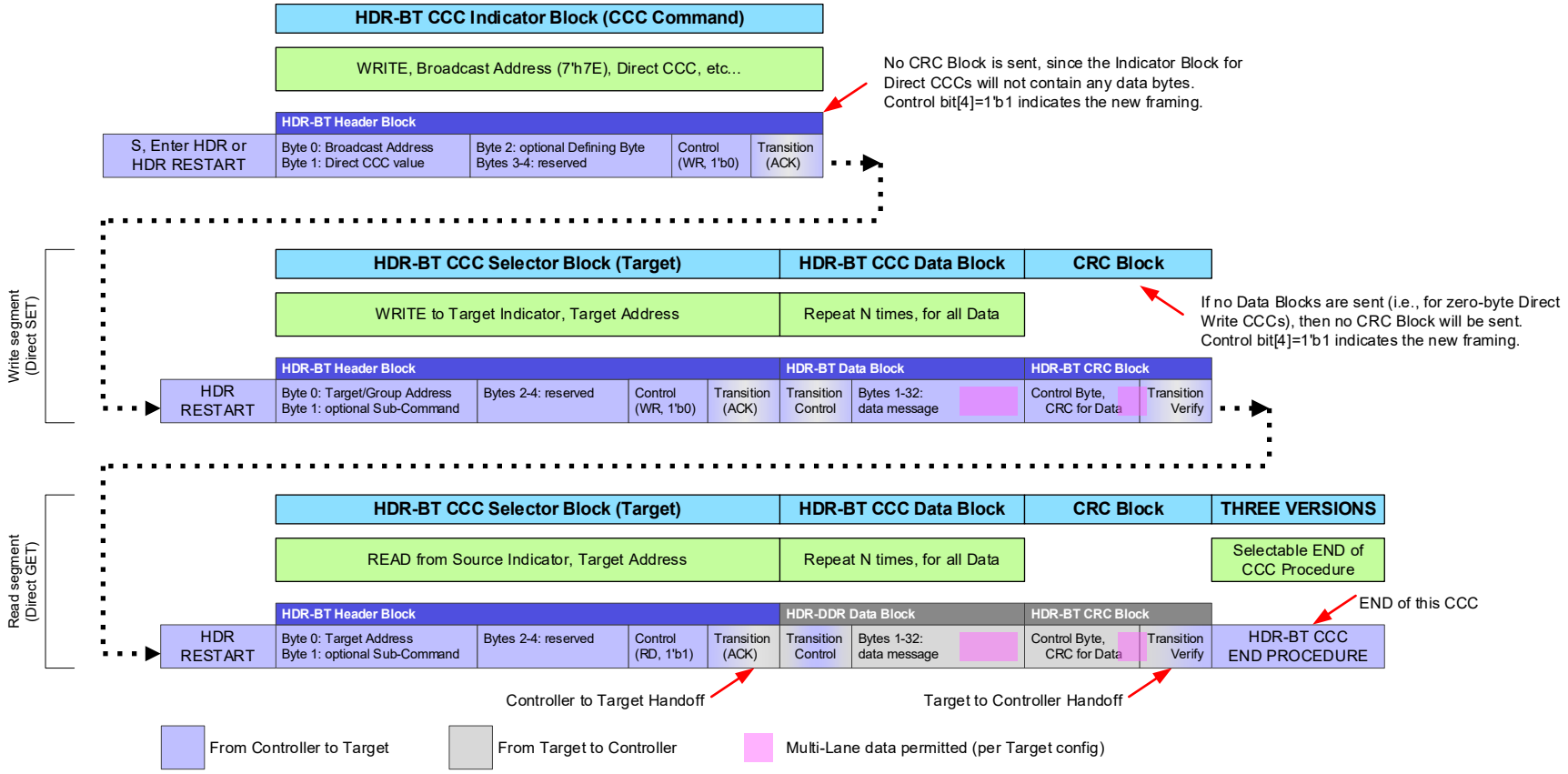


Figure 134 Begin Broadcast CCC in HDR-BT

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



6046

Figure 135 Begin Direct CCC in HDR-BT

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.4.1 HDR-BT CCC Header Block

Two versions of the HDR-BT Header Block are defined: HDR-BT CCC Indicator Block and HDR-BT CCC Selector Block.

6.4.3.4.1.1 HDR-BT CCC Indicator Block

An HDR-BT CCC Header Block that begins the framing for a Broadcast CCC or a Direct CCC shall be referred to as the HDR-BT CCC Indicator Block.

The HDR-BT CCC Indicator Block shall take the structure of the HDR-BT Header Block Format (see ***Section 6.4.3.3.1***). It is a common CCC Flow Element, and shall always be transmitted according to the bit packing format for the currently configured HDR-BT Coding (see ***Section 6.7.3.5.3.1***).

- Field **Address** (byte 0) shall contain the Broadcast Address (7'h7E).
- Field **Cmd0** (byte 1) shall contain the Command Code.
  - Per ***Section 4.3.7.2***, the upper bit of the Command Code is always 1'b0 for Broadcast CCCs or 1'b1 for Direct CCCs
- Field **Cmd1** field (byte 2) shall contain the optional Defining Byte for the Command Code.

  If the CCC does not support an optional Defining Byte, then the **Cmd1** field shall contain the value 8'h00.
- **For Broadcast CCCs Only:** Field **Cmd2** (byte 3) shall contain the optional Sub-Command Byte for the Command Code.

  If the Broadcast CCC does not support an optional Sub-Command Byte, or if this an Indicator Block for a Direct CCC, then field **Cmd2** shall contain the value 8'h00.
- **For All CCCs:** Field **Cmd3** (byte 4) is reserved and shall contain the value 8'h00.

  Note:

  *In previous versions of the I3C Specification, the Controller would send the first Broadcast CCC data bytes either in fields **Cmd1**/**Cmd2**/**Cmd3** if no Defining Byte, or in fields **Cmd2**/**Cmd3** if the Defining Byte was in **Cmd1**. This has changed in version 1.2 of the I3C Specification. For Broadcast CCCs, all data bytes shall now be sent in one or more HDR-BT CCC Data Blocks after this Indicator Block. The use of this new framing is indicated by a value of 1'b1 in Bit[4] of field **Control** (see below). However, Controllers might need to use the old framing (i.e., as defined in I3C v1.1.1 and earlier) when sending Broadcast CCCs to older I3C Targets that do not support the new framing.*
- Field **Control** (byte 5) shall be set to indicate the start of a CCC:
  - **Bit[0]** shall contain 1'b0 to indicate a Write.
  - **Bit[3]** shall contain 1'b0 to indicate the beginning of a CCC framing.
  - **Bit[4]** shall contain 1'b1 to indicate use of the new HDR-BT framing, as described above.
  - Other bitfields shall be set accordingly.

As noted earlier, in the new HDR-BT framing (indicated by **Bit[4]** being set to 1'b1) all of the following requirements apply to CCCs:

- **For Broadcast CCCs:** The Controller shall send all data bytes in one or more HDR-BT Data Blocks.
- HDR-BT CRC Blocks are no longer required for zero-byte transfers. This includes certain flow elements for CCCs, i.e., Header Blocks with no subsequent Data Blocks.
- The Controller shall not send the HDR-BT Header Block to end the CCC modality for Broadcast CCCs.

Targets will conditionally acknowledge receipt of the Indicator Block (i.e., during the **Transition** byte), per ***Section 6.4.3.4.4***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

After the Indicator Block:

- **For Broadcast CCCs Only:** If (and only if) this Broadcast CCC has data bytes, then the Controller shall conditionally send one or more Data Blocks followed by the CRC Block. Otherwise there are no data bytes and the Controller shall not send any Data Blocks and shall not send the CRC Block. The CCC modality shall automatically end with the HDR Restart Pattern.
- **For Direct CCCs Only:** The Indicator Block for Direct CCCs will not have any data bytes, so the Controller shall not send any Data Blocks and shall not send the CRC Block. The CCC modality shall continue through subsequent Read Segments and/or Write Segments, until the Controller uses a suitable End of CCC Procedure (per ***Section 6.4.3.4.7***).

6.4.3.4.1.2 HDR-BT CCC Selector Block

An HDR-BT CCC Header Block that begins a Read Segment or a Write Segment in Direct CCC framing shall be referred to as the HDR-BT CCC Selector Block.

The HDR-BT CCC Selector Block shall take the structure of the HDR-BT Header Block Format (see ***Section 6.4.3.3.1***). It is a common CCC Flow Element, and shall always be transmitted according to the bit packing format for the currently configured HDR-BT Coding (see ***Section 6.7.3.5.3.1***).

- Field **Address** (byte 0) shall contain the Address of the Target (or Group) to be addressed in this Segment of the Direct CCC.
- Field **Cmd0** (byte 1) shall contain the optional Sub-Command Byte for the Command Code.
  - If the Direct CCC does not support an optional Sub-Command Byte, then field **Cmd0** shall contain the value 8'h00.
- Fields **Cmd1**/**Cmd2**/**Cmd3** (bytes 2–4) shall be reserved, and shall each be set to 8'h00.
- Field **Control** (byte 5) shall be set to indicate the continuation of a Direct CCC:
  - **Bit[0]** shall contain 1'b0 to indicate a Write Segment (i.e., Direct Write or Direct SET), or 1'b1 to indicate a Read Segment (i.e., Direct Read or Direct GET). Note that Group Addresses may only be used with Write Segments.
  - **Bit[3]** shall contain 1'b1 to indicate the continuation of a Direct CCC framing.
  - **Bit[4]** shall contain 1'b1 to indicate use of the new HDR-BT framing as described above.
  - Other bitfields shall be set accordingly.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.4.3.4.2 HDR-BT CCC Data Block

An HDR-BT CCC Data Block may follow an HDR-BT CCC Header Block, in the appropriate place for either a Broadcast CCC flow or a Direct CCC flow (see ***Figure 134*** or ***Figure 135***).

6.4.3.4.2.1 Broadcast CCC Flow

The HDR-BT CCC Data Block in a Broadcast CCC flow shall take the structure of the HDR-BT Data Block Format (see ***Section 6.4.3.3.2***), and shall contain up to 32 bytes of data in its payload.

As described in ***Section 6.1.3***, HDR-BT CCC Data Blocks for Broadcast CCCs are common CCC Flow Elements and shall always be transmitted according to the bit packing format for the currently configured HDR-BT Coding (see ***Section 6.7.3.5.3.1***).

Note:

Since Broadcast CCCs in HDR-BT Mode are sent to all Targets, a Target ***should not*** *pull SDA[0] to Low during the "Park1,High-Z" of the* **Transition_Control** *byte of an HDR-BT CCC Data Block for a Broadcast CCC. Doing so would inappropriately terminate the Broadcast CCC data transfer to all Targets.*

6.4.3.4.2.2 Direct CCC Flow

The HDR-BT CCC Data Block in a Direct CCC flow (as part of a Write Segment or Read Segment) is a per-Target CCC Flow Element, and shall be transmitted according to the bit packing format of the currently configured Multi-Lane Frame format and Data Transfer Coding (see ***Section 6.7.3.5.3.1***) for the addressed Target (or Group).

This may use a different bit packing format, and may use more additional data Lanes than the preceding HDR-BT CCC Selector Block. If so, then the Controller and indicated Target (or Targets in a Group, if applicable) shall correctly switch from one bit-packing format to another, as part of the CCC flow.

- **For Direct CCC Write Segments With Data:** The Controller shall send the HDR-BT CCC Data Block(s) as well as the following HDR-BT CCC CRC Block. Following these, the Controller shall either send another Segment, or perform a suitable End of CCC Procedure (per ***Section 6.4.3.4.7***).
- **For Direct CCC Write Segments Without Data:** The Controller shall not send any HDR-BT CCC Data Blocks and shall not send an HDR-BT CCC CRC Block. Instead, it shall either send another Segment, or perform a suitable End of CCC Procedure (per ***Section 6.4.3.4.7***) after the Indicator Block.
- **For Direct CCC Read Segments:** If the addressed Target responds to the Selector Block with ACK, then this Target shall send the HDR-BT CCC Data Block(s) and the following HDR-BT CCC CRC Block. Following these, the Controller shall either send another Segment, or perform a suitable End of CCC Procedure (per ***Section 6.4.3.4.7***).

Note:

For a Direct CCC, no HDR-BT CCC Data Blocks or CRC Block will be sent with the Indicator Block.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.4.3 HDR-BT CCC CRC Block

An HDR-BT CCC CRC Block serves as the Termination Marker, as described in the HDR-x generic CCC flows. To conform to the HDR-BT Mode Framing model (see ***Section 6.4.3.3***), this Termination Marker uses the same structure as the standard HDR-BT CRC Block.

Note:

> ***For Write CCCs Only:*** *If the Controller does not send any HDR-BT CCC Data Blocks (i.e., for a zero-byte write), then the Controller shall not send the HDR-BT CRC Block. Instead, the Controller shall either (a) Perform the appropriate End of CCC procedure based on the type of CCC (Direct Write vs. Broadcast); or (b) For Direct CCCs only, send the HDR Restart Pattern followed by the next HDR-BT CCC Selector Block (i.e., to address the next Target with the same Direct CCC). In this case, the* ***Transition*** *byte in the preceding Indicator Block or Selector Block acts as the Termination Marker.*

6.4.3.4.3.1 Broadcast CCC Flow

For the end of Broadcast data in a Broadcast CCC flow, the Controller shall transmit the HDR-BT CRC Block (see ***Section 6.4.3.3.3***) at the end of the Broadcast data message (i.e., after the last HDR-BT CCC Data Block).

As described in ***Section 6.1.3***, HDR-BT CCC CRC Blocks for Broadcast CCCs are common CCC Flow Elements and shall always be transmitted according to the bit packing format for the currently configured HDR-BT Coding (see ***Section 6.7.3.5.3.1***). This bit packing format shall be the same as that used in the preceding HDR-BT CCC Data Block.

6.4.3.4.3.2 Direct CCC Flow

Indicator Block: The Controller **shall not** send the HDR-BT CCC CRC Data Block with the Indicator Block.

Segment: For the end of a Read or Write Segment sent in a Direct CCC flow, the HDR-BT CRC Block is a per-Target CCC Flow Element, and shall be transmitted according to the bit packing format of the currently configured Multi-Lane Frame format and Data Transfer Coding (see ***Section 6.7.3.5.3.1***) for the addressed Target or Group. This bit packing format shall be the same as that used in the preceding HDR-BT CCC Data Block.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.4.3.4.4 HDR-BT CCC Indicator Block ACK

Per ***Section 6.1.3.2***, a Target that supports CCCs in HDR-BT Mode shall indicate acknowledgement of the receipt of the HDR-BT CCC Indicator Block, using the standard method of acknowledging an HDR-BT write (i.e., by using Bits[1:0] of the Transition byte in the HDR-BT Header Block per ***Section 6.4.3.3.1***).

The following conditions apply:

- If the Parity bits in the Control byte do not match the computed Parity of the preceding fields in the HDR-BT Header Block, then the Target shall NACK (i.e., shall not acknowledge) the Indicator Block, since this is likely a transmission error.
- **For Older Targets:** If Bit[4] in the Control byte is set to 1'b1 but the Target does not understand the new HDR-BT framing, then the Target should NACK the Indicator Block.
- For all other situations, the Target shall ACK the Indicator Block.

All such Targets will acknowledge a properly formed Indicator Block for both Broadcast CCC flows and Direct CCC flows, since the same HDR-BT CCC Indicator Block format is used to signal the beginning of both flows.

If the Indicator Block's Command Code (byte 1) indicates a Direct CCC, then all Targets receiving that Indicator Block shall capture the new Command Code and optional Defining Byte (byte 2) and use these if they are addressed in a future Write or Read Segment in the Direct CCC framing.

Note:

*For Direct CCC flows, acknowledgement serves the important function of ensuring that a Target Device understands the CCC framing, has captured the Command Code and optional Defining Byte from the Indicator Block, and is ready to both (a) Match the Address on the next HDR-BT Header Block that is a Selector Block, and (b) Make a decision to respond to the Direct CCC if addressed, per **Section 6.1.3.2**.*

If the Controller does not receive acknowledgement from at least one Target, then it shall perform any valid HDR-BT CCC End Procedure. Additionally, the Controller may initiate an error recovery procedure, which might include exiting HDR-BT Mode and attempting to re-send the CCC in SDR Mode.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.4.5 HDR-BT CCC Write Segment

An HDR-BT CCC Write Segment for a Direct Write or Direct SET CCC shall start with an HDR-BT CCC Selector Block to indicate the Address of the Target (or Group) to receive the data, followed by one or more HDR-BT CCC Data Blocks of the appropriate structure and format, if required by the CCC definition. However, if the CCC definition does not require data to be written to the Target, then no additional Data Blocks and no CRC Block shall be sent.

After the Controller sends the HDR-BT CCC Selector Block, the Target shall either indicate acceptance of the Direct CCC by acknowledging the write, or else reject the Direct CCC by not responding to the Controller. The Target shall use the cached Command Code and optional Defining Byte from the previously received HDR-BT CCC Indicator Block, and shall decide whether to accept or reject the Direct CCC immediately after receiving the HDR-BT CCC Selector Block and matching its own Dynamic Address (or Group Address, if assigned and applicable).

The Target shall use the standard methods to either accept (i.e., ACK) or reject (i.e., NACK) an HDR-BT write, as defined in ***Section 6.4.3.3.1*** regarding the handling of Bits[1:0] of the HDR-BT Header Block's **Transition** byte field.

If the Target rejects the Direct CCC, then the Controller shall either send the HDR Restart Pattern and attempt another Read or Write Segment for this CCC, or else perform any valid HDR-BT CCC End Procedure. For Direct CCCs to a Group Address, the Controller only sees this rejection if no Targets accept the Direct CCC.

If the Target accepts the Direct CCC, then the Controller shall either:

- **For Direct Write CCCs With Data:** Send one or more HDR-BT CCC Data Blocks of the appropriate structure, if required by the CCC definition, and then send an HDR-BT CCC CRC Block to terminate the data and end the Segment.
- **For Zero-Byte Direct Write CCCs Only:** Send no HDR-BT CCC Data Blocks and no HDR-BT CCC CRC Block, since the Segment ends after the Selector Block.

At the end of a Write Segment, the Controller shall either send the HDR Restart Pattern and attempt another Read or Write segment for this CCC, or else end the CCC modality by performing any valid HDR-BT CCC End Procedure, per ***Section 6.4.3.4.7***.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.4.3.4.6 HDR-BT CCC Read Segment

An HDR-BT CCC Read Segment for a Direct Read or Direct GET CCC shall start with an HDR-BT CCC Selector Block to indicate the Address of the Target to provide data for the CCC, followed by a Bus Turnaround. In response, the indicated Target will either accept the Direct CCC by transmitting one or more HDR-BT CCC Data Blocks of the appropriate structure, or else reject the Direct CCC by not responding to the Bus Turnaround condition.

The Target shall use the standard methods to either accept (i.e., ACK) or reject (i.e., NACK) an HDR-BT read, as defined in ***Section 6.4.3.3.1*** regarding the handling of Bits[1:0] of the HDR-BT Header Block's **Transition** byte field.

If the Target accepts the Direct CCC, then the Controller shall yield control of the Bus to the Target, so that it can control the Bus to send the Data Blocks. After sending the Last Data Block, the Target shall send an HDR-BT CRC CCC Block, indicating the end of the read transfer. Upon sending the HDR-BT CCC CRC Block, the Target shall yield control of the Bus, and the Controller shall resume control.

At the end of a Read Segment, the Controller shall either send the HDR Restart Pattern and attempt another Read or Write Segment for this CCC, or else end the CCC modality by performing any valid HDR-BT CCC End Procedure per ***Section 6.4.3.4.7***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.4.7 HDR-BT CCC End Procedures

As illustrated in ***Figure 136***, the HDR-BT CCC ends using one of several possible CCC End Procedures.

6.4.3.4.7.1 Option 1: For Direct CCCs Only: End HDR-BT CCC Followed by New HDR-BT CCC

This End Procedure indicates the end of a Direct CCC in HDR-BT Mode, and the beginning of a new Broadcast or Direct CCC in HDR-BT Mode.

This End Procedure begins with an HDR Restart Pattern, followed by an HDR-BT CCC Indicator Block which provides a new CCC with optional Defining Byte. Since the Indicator Block's Control field (byte 5) signals the beginning of a CCC framing (i.e., Bit[3] is set to 1'b0) the Target shall interpret this as the end of a Direct CCC framing (if present).

Following the Indicator Block, additional HDR-BT Blocks shall follow, per the CCC flow type.

The flow steps are:

1. Controller sends HDR Restart Pattern.
2. Controller sends HDR-BT CCC Indicator Block, using the same format described above.

   The bit packing format of this Block shall be appropriate for the current Coding's common CCC flow elements (see ***Section 6.7.3.5.3.1***).
3. Targets acknowledge receipt of Indicator Block, as defined above.
4. Flow continues as Broadcast CCC or Direct CCC, as shown above.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.4.3.4.7.2 Option 2: For Direct CCCs Only: End HDR-BT CCC Followed by HDR-BT Generic Transaction (+ Optional HDR Exit)

This End Procedure indicates the end of a Direct CCC in HDR-BT Mode, and either a return to standard HDR-BT transactions, a subsequent Broadcast CCC in HDR-BT Mode, or a subsequent exit from HDR-BT Mode.

This End Procedure begins with an HDR Restart Pattern, followed by a special HDR-BT Header Block which ends the CCC flow, but is neither an Indicator nor a Selector. Since this Header Block is neither an Indicator nor a Selector, no CCC or optional Defining Byte is included. And since no data bytes are sent, this End Procedure has no HDR-BT Data Blocks and no HDR-BT CRC Block either.

After this, the HDR Restart Pattern signals the end of CCC modality, and that the next HDR-BT Header Block would be treated as a standard HDR-BT transaction (not as a CCC flow) unless it is addressed to 7'h7E (i.e., if the next message is a Broadcast CCC). Alternatively, the HDR Exit Pattern signals the end of HDR-BT Mode and a subsequent return to SDR Mode.

In either case, this HDR-BT Header Block is distinguished from an HDR-BT CCC Indicator Block because it indicates a ‘Read’ command from the Broadcast Address (7'h7E), which would otherwise be an invalid combination for HDR-BT transactions. Target Devices shall interpret this condition as the Termination Marker ending the CCC modality for the Direct CCC, since this condition is distinct from the start of a new Broadcast CCC or Direct CCC (per CCC End Procedure Option 1, above).

The flow steps are:

1. Controller sends HDR Restart Pattern.
2. Controller sends HDR-BT Header Block, using the following values:
   - The **Address** field (byte 0) shall contain the Broadcast Address (7'h7E / R).
   - The **Control** field (byte 5) shall be set to indicate the end of the CCC modality:
     - **Bit[0]** contains 1'b1 to indicate a Read.
     - **Bit[3]** contains 1'b0 to indicate that Direct CCC framing is not continued from a prior Selector Block (if applicable).
     - **Bit[4]** contains 1'b1 to indicate use of the new HDR-BT framing as described above.
     - Other bitfields shall be set accordingly.
   - The bit packing format of this Block shall be appropriate for the current Coding’s common CCC flow elements (see ***Section 6.7.3.5.3.1***).
3. Targets do not acknowledge receipt of the HDR-BT Header Block.
4. Targets detect this distinct pattern (i.e., Read from 7'h7E with no Data Block) and end CCC modality.
5. Controller sends either HDR Restart Pattern, or HDR Exit Pattern.
   - If HDR Exit Pattern, then Controller exits HDR-BT Mode and returns the Bus to SDR Mode.

6.4.3.4.7.3 Option 3: For Broadcast CCCs Only: Send HDR Restart Pattern

After a Broadcast CCC, the CCC modality ends automatically, and the Controller simply sends an HDR Restart Pattern and then continues with subsequent HDR-BT transactions.

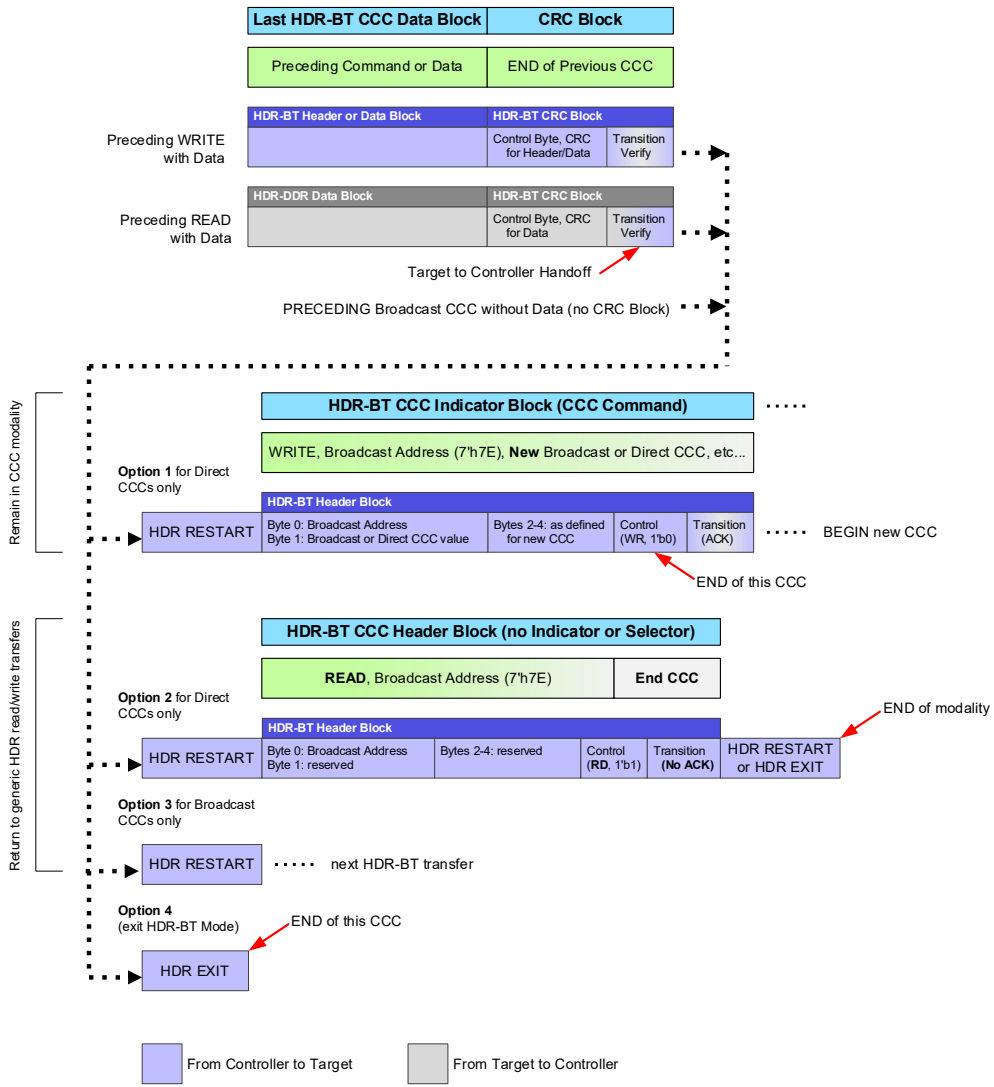
The single flow step is:

1. Controller sends HDR Restart Pattern.

6.4.3.4.7.4 Option 4: End HDR-BT CCC and Return to SDR Mode

This is the simplest End Procedure option. It indicates the end of CCCs in HDR-BT Mode and an immediate exit from HDR-BT Mode, when the Controller sends the HDR Exit Pattern.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



6292

Figure 136 HDR-BT CCC End Procedure Options

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.5 Options

Targets that support HDR-BT shall support CRC-16.

Targets that support HDR-BT may optionally also support CRC-32. When the **GETCAPS** Format 1 CCC is sent (see ***Section 4.3.7.3.19***), a Target that supports CRC-32 in HDR-BT Mode shall return a value of 1'b1 in bit 5 of the GETCAP3 byte (the third returned data byte). A Controller that sees the value 1'b1 in that bit will know that the Target is capable of supporting CRC-32 in HDR-BT Mode.

Targets that support HDR-BT shall support SCL from the Controller on Read. They may also optionally support emitting their own SCL on Read.

Note:

The I3C Specification does not define a mechanism for discovering a Target's capability to control SCL for Read transfers. The Controller may select whether the Target should control SCL or not control SCL, using Bit[2] of the **Control** *byte in the Header Block. Per* ***Section 6.4.3.3.1****, a Target that does not actually support driving SCL for Read transfers shall not accept (i.e., must provide NACK for) the Read transfer request, if it receives such a Header Block with Bit[2] set to 1 in the* **Control** *byte. In this case, the Controller should retry the same Read transfer with Bit[2] set to 0, to indicate that the Controller will control SCL for the Read transfer. If the Read transfer request's Header Block was otherwise valid, then the Target should accept (i.e., should provide ACK for) this Read transfer.*

Targets may choose to support the Data Block Delay mechanism for Read transfers. This mechanism allows the Target to delay sending a Data Block (when permitted by the Controller, as indicated in the Header Block) if it does not have enough data bytes to fill the Data Block (see ***Section 6.4.3.3.4***).

Note:

The I3C Specification does not define a mechanism for discovering whether the Target supports the Data Block Delay mechanism. If the Target does not support the Data Block Delay mechanism, then it shall ignore the value of Bit[2] in the ***Control*** *byte of the Header Block, when the Header Block indicates that this is a Read transfer. In such a case, the Target must either return full Data Blocks, except for the Last Data Block; or mitigate the situation, depending on whether it controls SCL for this Read transfer.*

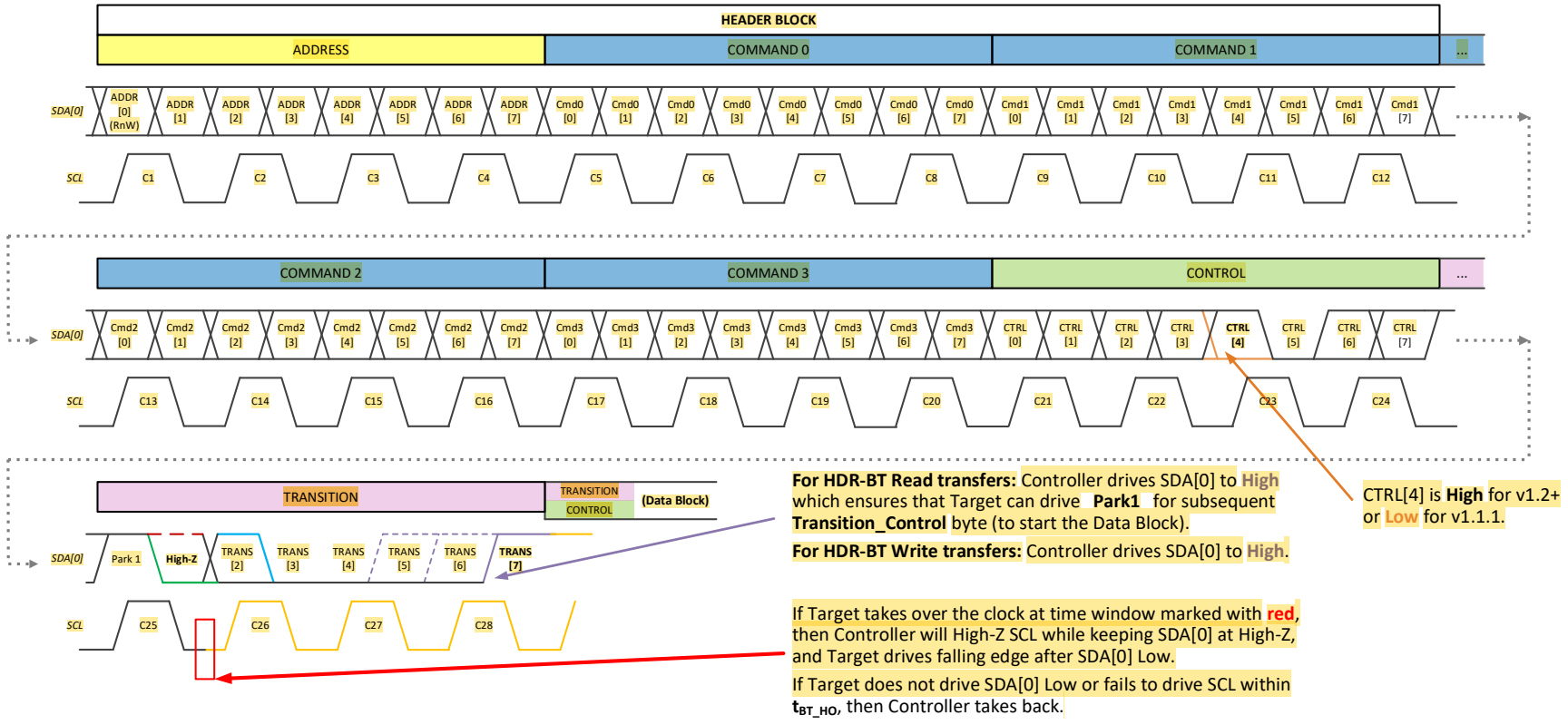
A) If the Target controls SCL for this Read transfer, then it must either reduce the data rate or end the transfer early, if it is unable to sustain the transfer of data bytes.

B) If the Target does not control SCL for this Read transfer, or if it does not have that capability, then it must end the transfer early, if it is unable to sustain the transfer of data bytes.

If the Target does support the Data Block Delay mechanism, and if it was permitted to use this mechanism for a Read transfer, then the Target may use either of these methods above that might be valid or permitted, and for which it has the capability; alternately, it may also use Delay bytes, per ***Section 6.4.3.3.4****.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.6 Diagrams

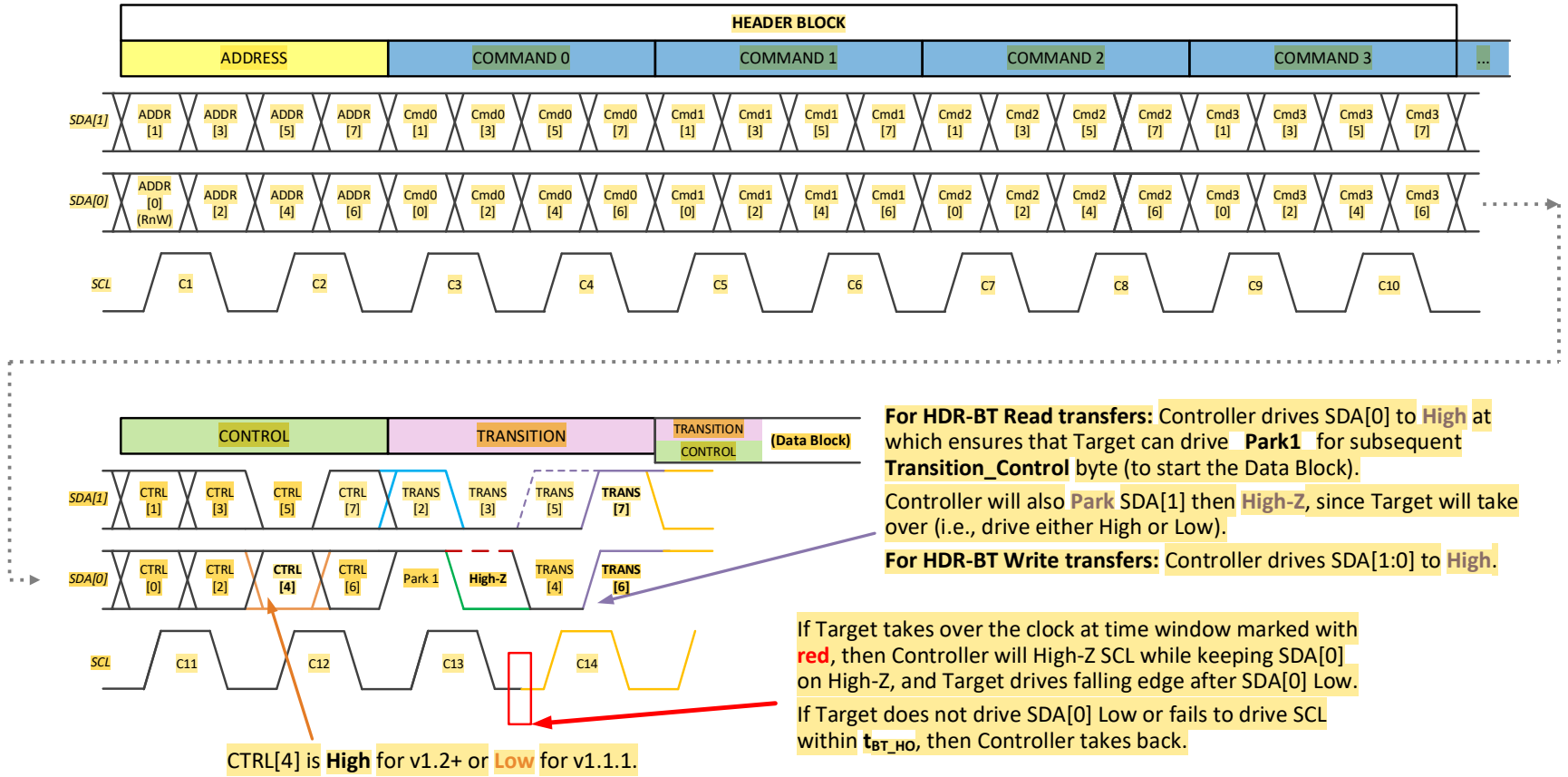


6328

Figure 137 Header Block for Single Lane

See Errata 01, Item 27

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

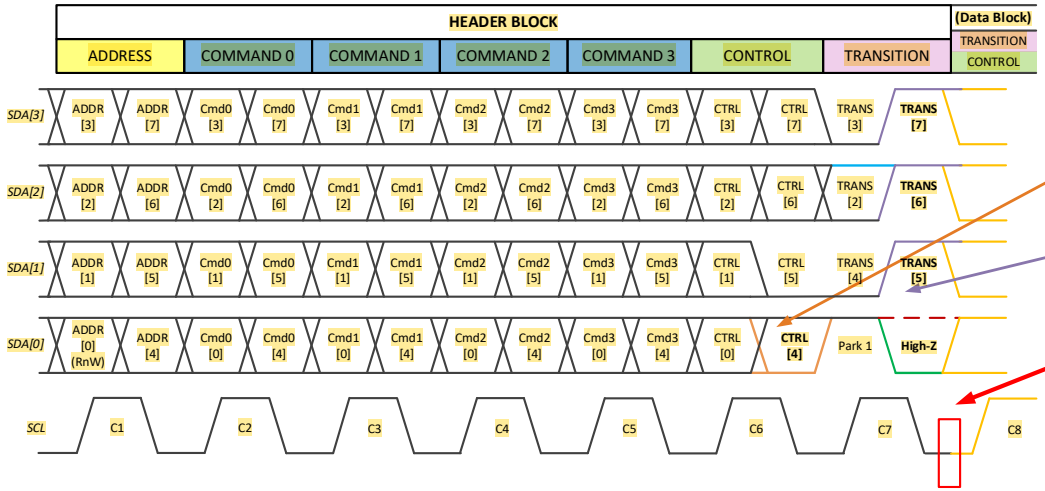


6329

Figure 138 Header Block for Dual Lane

See Errata 01, Item 28

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



CTRL[4] is **High** for v1.2+ or **Low** for v1.1.1.

For HDR-BT Read transfers: Controller will **Park** SDA[3:1] then **High-Z**, since Target will take over (i.e., drive either High or Low for **Transition_Control** byte, to start the Data Block).

For HDR-BT Write transfers: Controller drives SDA[3:1] to **High**.

If Target takes over the clock at time window marked with **red**, then Controller will High-Z SCL while keeping SDA[0] on High-Z, and Target drives falling edge after SDA[0] Low.

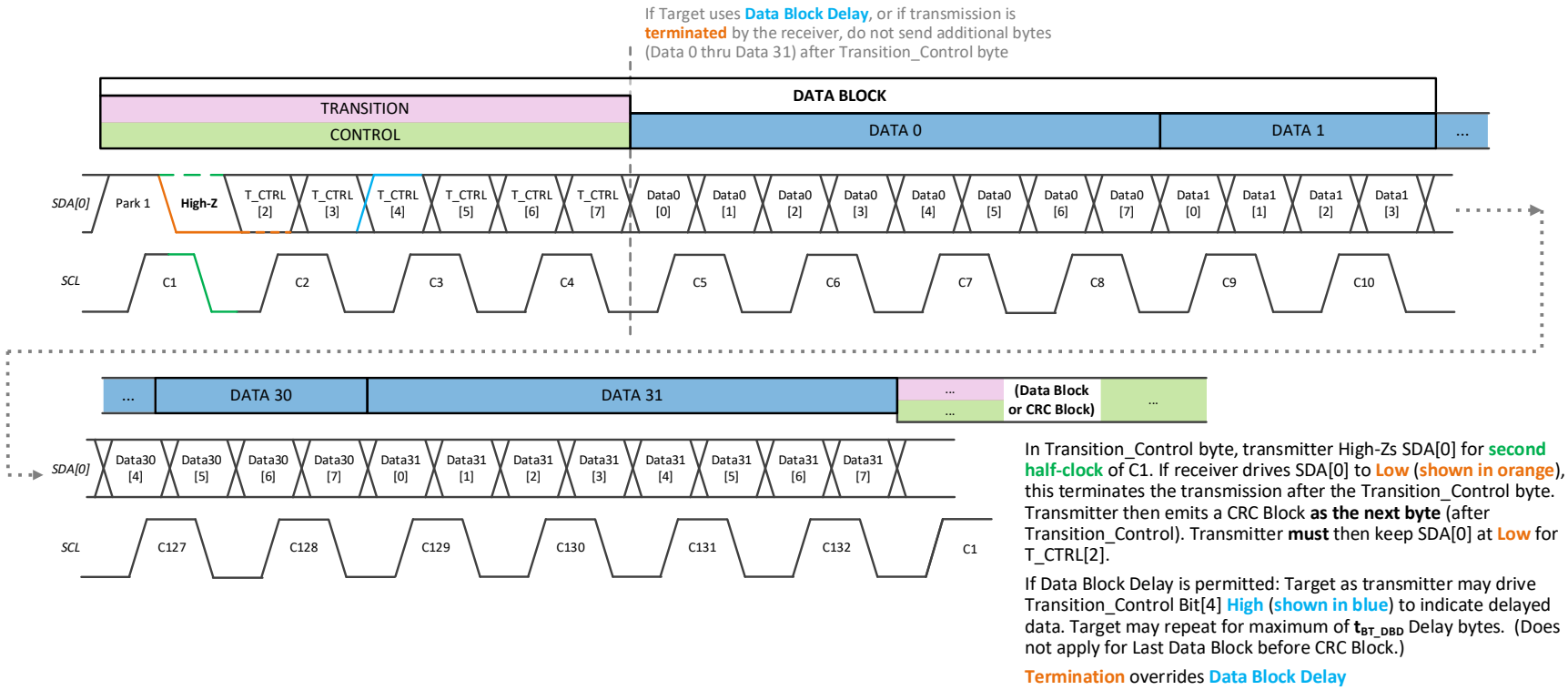
If Target does not drive SDA[0] Low or fails to drive SCL within t_{BT_HO}, then Controller takes back.

6330

Figure 139 Header Block for Quad Lane

See Errata 01, Item 29

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



In Transition_Control byte, transmitter High-Zs SDA[0] for **second half-clock** of C1. If receiver drives SDA[0] to **Low** (**shown in orange**), this terminates the transmission after the Transition_Control byte. Transmitter then emits a CRC Block **as the next byte** (after Transition_Control). Transmitter **must** then keep SDA[0] at **Low** for T_CTRL[2].

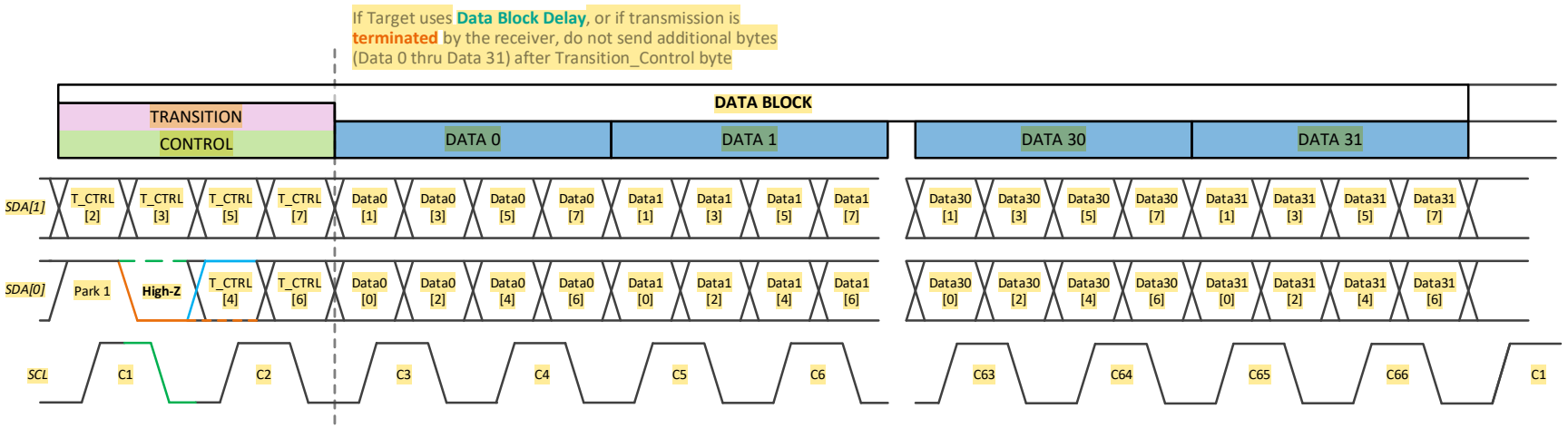
If Data Block Delay is permitted: Target as transmitter may drive Transition_Control Bit[4] **High** (**shown in blue**) to indicate delayed data. Target may repeat for maximum of t_{BT_DBD} Delay bytes. (Does not apply for Last Data Block before CRC Block.)

Termination overrides **Data Block Delay**

6331

Figure 140 Data Block for Single Lane

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



In Transition_Control byte, transmitter High-Zs SDA[0] for second half-clock of C1. If receiver drives SDA[0] to Low (shown in orange), this terminates the transmission after the Transition_Control byte. Transmitter then emits a CRC Block **as the next byte** (after Transition_Control). Transmitter **must** then keep SDA[0] at Low for T_CTRL[4].

If Data Block Delay is permitted: Target as transmitter may drive Transition_Control Bit[4] High (shown in blue) to indicate delayed data. Target may repeat for maximum of t_{BT_DBD} Delay bytes. (Does not apply for Last Data Block before CRC Block.)

Termination overrides Data Block Delay

6332

Figure 141 Data Block for Dual Lane

See Errata 01, Item 30

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

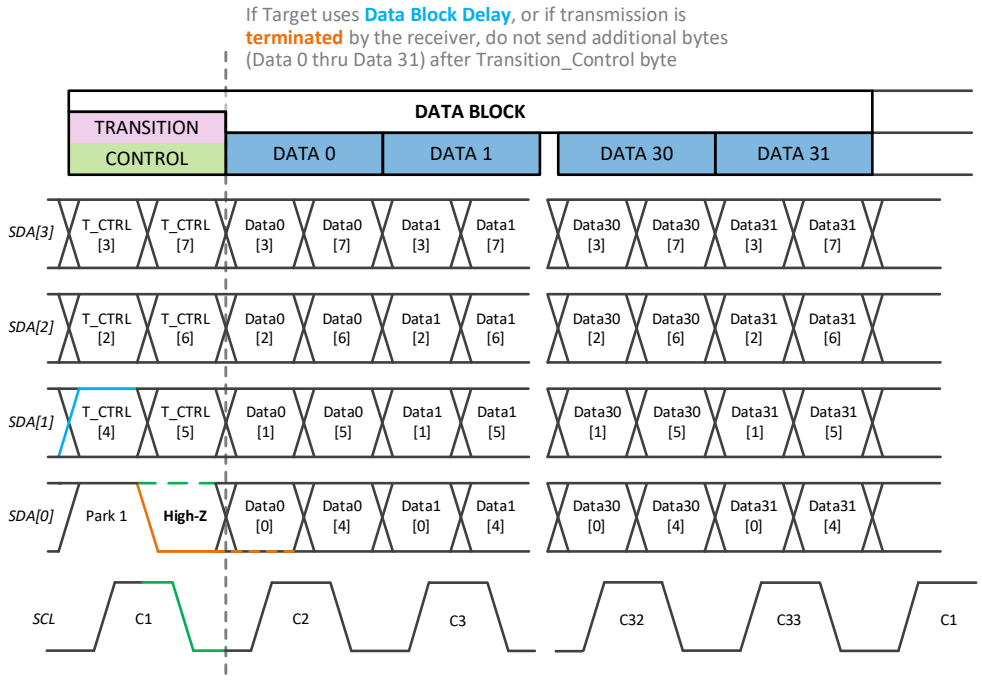


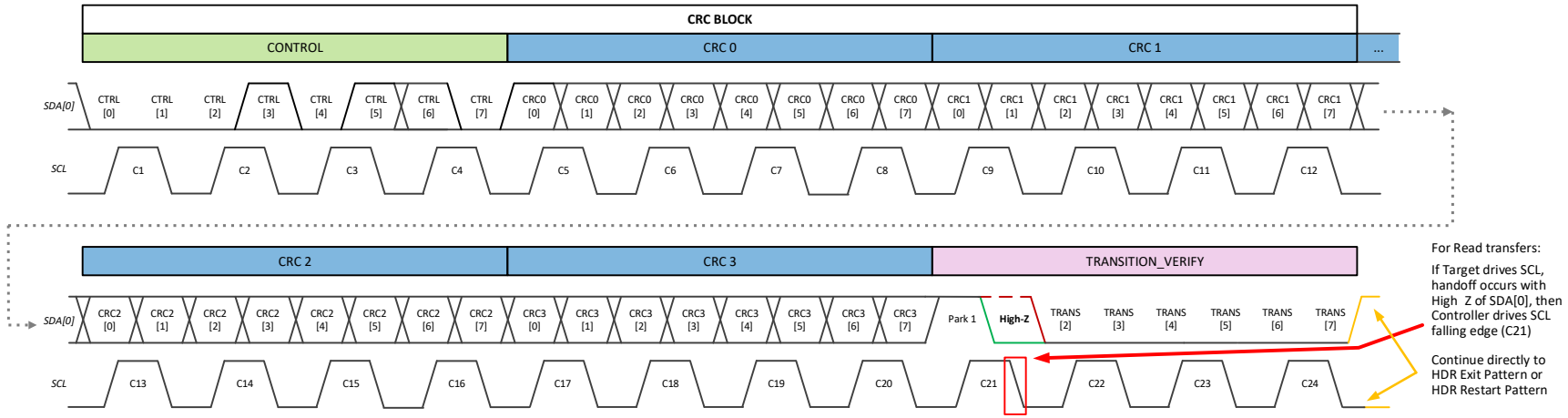
Figure 142 Data Block for Quad Lane

In Transition_Control byte, transmitter High-Zs SDA[0] for **second half-clock** of C1. If receiver drives SDA[0] to **Low** (**shown in orange**), this terminates the transmission after the Transition_Control byte. Transmitter then emits a CRC Block **as the next byte** (after Transition_Control). Transmitter then keeps SDA[0] at **Low**, since this is the beginning of the Control byte (in CRC Block).

If Data Block Delay is permitted: Target as transmitter may drive Transition_Control Bit[4] **High** (**shown in blue**) to indicate delayed data. Target may repeat for maximum of t_{BT_DBD} Delay bytes. (Does not apply for Last Data Block before CRC Block.)

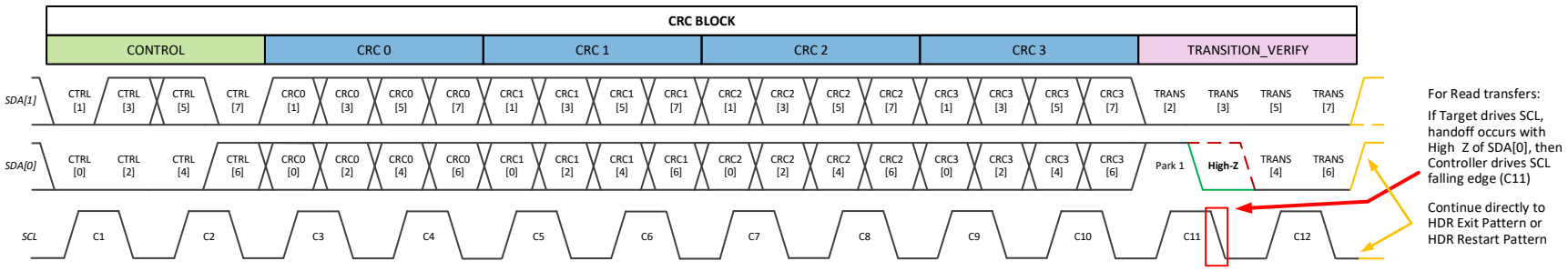
Termination overrides **Data Block Delay**

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



6334

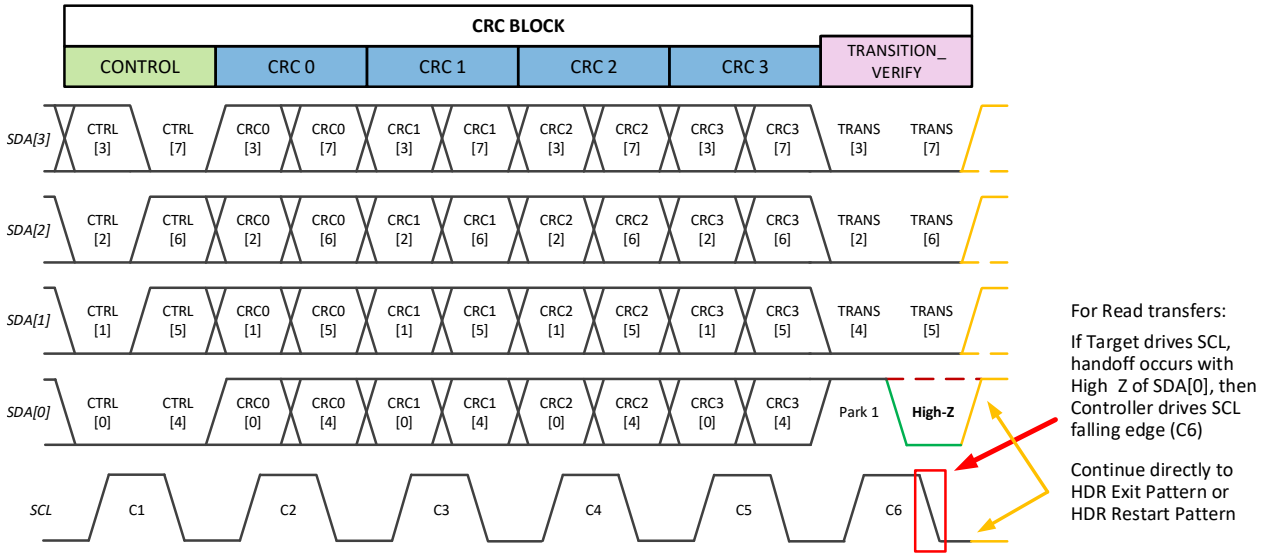
Figure 143 CRC Block for Single Lane



6335

Figure 144 CRC Block for Dual Lane

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



6336

Figure 145 CRC Block for Quad Lane

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.7 Transfer Flow Control

HDR-BT Mode provides several procedures that allow either the Controller or the Target to control the progress of data transfers. This Section describes these flow control elements.

Note:

In contrast to HDR-DDR Mode and HDR-Ternary Modes, in HDR-BT Mode the flow control elements are always available, i.e., the Controller does not need to use any special configuration procedures to configure Targets to either utilize, or properly react to, the flow control elements during transfers. This is in contrast to HDR-DDR Mode and the HDR-Ternary Modes, where flow control must be explicitly configured via the ENDXFER CCC.

6.4.3.7.1 Controller Clock Stalling

To avoid Underrun or Overflow situations, the Controller may stall the I3C Bus. Stalling occurs during the SCL Low period, under the conditions described below.

During the Header Block:

- The Controller may stall after the last falling edge of the Control byte in the HDR-BT Header Block, and hold SCL Low before the **Transition** byte. Subsequently, driving SCL to High for the first rising edge of the **Transition** byte will end the stall.

During a Write or Read Transfer:

- The Controller may stall after the last falling clock edge of the **Transition_Control** byte in the HDR-BT Data Block, and hold SCL Low before the first data byte. Subsequently, driving SCL to High for the first rising edge of the first data byte will end the stall.
  - However, if the receiver pulls SDA[0] to Low for Bit[1] to terminate the transfer, then the Controller should not stall the I3C Bus.

Note:

*Clock stalling in HDR-BT Mode is not allowed if the Target transmits SCL for a Read transfer. Instead of stalling, the Target should use the Data Block Delay mechanism (see **Section 6.4.3.3.4**).*

Clock stalling is not defined during the HDR-BT CRC Block, primarily because it should not be necessary to stall after the Last Data Block. (The transmitter should not need extra time to compute the CRC checksum for the transmitted data bytes, and the receiver should not need extra time to compute the CRC checksum for the received data bytes and compare it with the CRC checksum that it will receive in the CRC Block.)

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.4.3.7.2 Transfer Flow Control During Write Transfers

The Target can use the **Transition** byte, **Transition_Control** byte, and **Transition_Verify** byte to accept or reject a Write transfer, or to provide flow control at defined points during the Write transfer.

Figure 146 shows several examples of Write transfer flow control:

- A Target that does not accept the Write transfer;
- A Target that accepts the Write transfer and receives all Data Blocks;
- A Target that terminates the Write transfer before the Controller has sent all Data Blocks; and
- A Target that indicates a CRC mismatch (or other error) after it has received the Data Blocks.

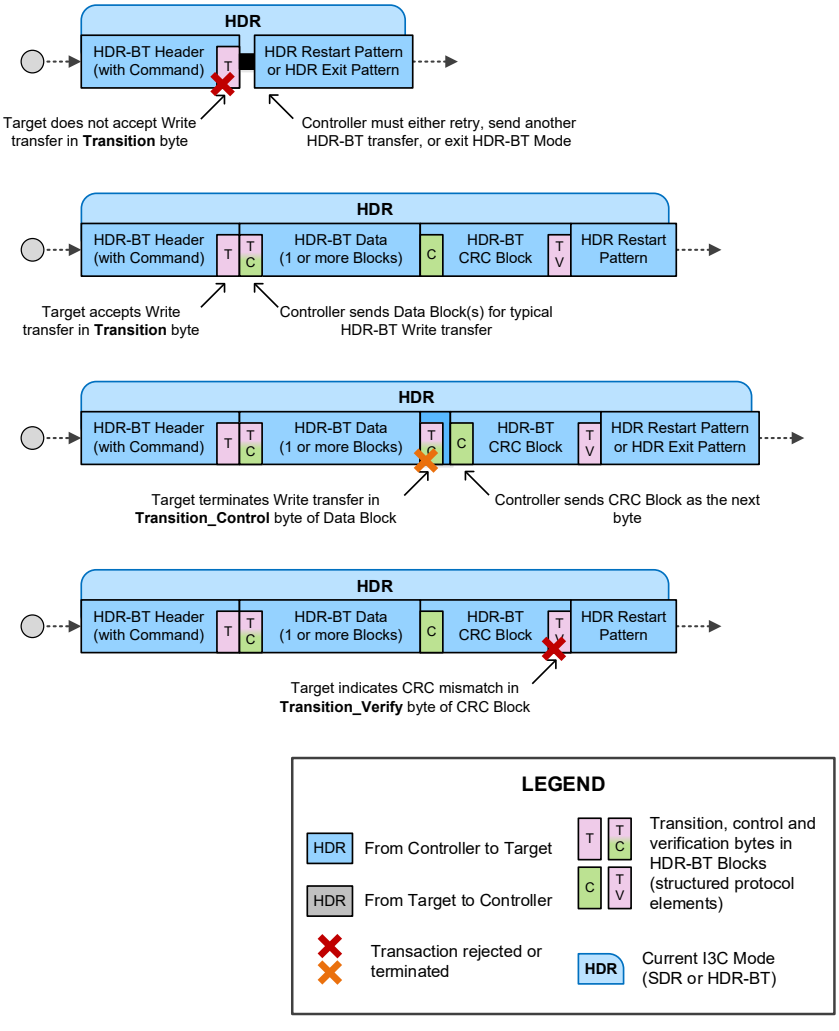


Figure 146 HDR-BT Flow Control for Write Transfers

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.4.3.7.3 Target Flow Control During Read Transfers

The Target can also use the **Transition** byte, **Transition_Control** byte, and **Transition_Verify** byte to accept or reject a Read transfer, or to provide flow control at defined points during the Read transfer. If the Data Block Delay mechanism is permitted (per ***Section 6.4.3.3.4***), then the Target may also use this mechanism to delay sending Data Blocks while it prepares data to send to the Controller.

Figure 147 shows several examples of Read transfer flow control:

- A Target that does not accept the Read transfer;
- A Target that accepts the Read transfer and sends all Data Blocks (i.e., without any delay or termination);
- A Target that handles the Controller terminating the Read transfer, and subsequently provides the CRC Block after the last Data Block that it sent; and
- A Target that uses the Data Block Delay mechanism before sending the first Data Block (i.e., assuming that was is allowed to do so by the Controller, as indicated by Bit[2] = 1'b0 in the **Transition** byte of this transfer's Header Block).

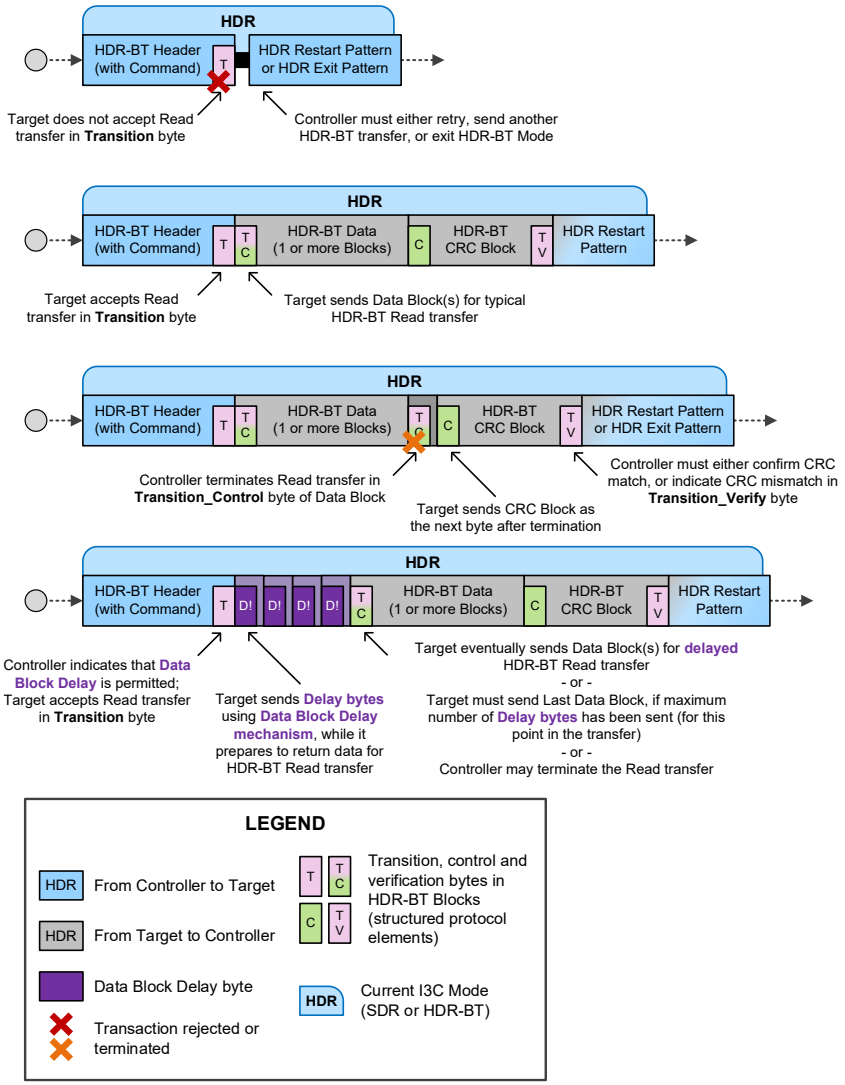


Figure 147 HDR-BT Flow Control for Read Transfers

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.5 F004: Secondary Controller

This Section defines I3C Optional Feature F004: Secondary Controller.

Table 95 Optional Feature 004 Status

Optional Feature ID	F004
Optional Feature Name	Secondary Controller
Optional Feature Version	1
Optional Feature specification was last updated in I3C Spec Version	I3C v1.2
Optional Feature is not compatible with other Optional Features	–
Optional Feature is included in I3C Basic	Yes

Changes in this Version

For detailed changes in this version, see the "diff" file between I3C v1.1.1 and I3C v1.2 in the MIPI Specification Repository ***[MIPI09]***.

6.5.1 Scope and Purpose (Informative)

The scope of ***Section 6.1*** is limited to fully specifying I3C Optional Feature F004: Secondary Controller. An I3C Device that implements F004 is required to implement all **shall** statements in ***Section 6.5.3***, and can optionally implement any combination of "may" statements in ***Section 6.5.3***. In addition, MIPI Alliance recommends that manufacturers follow all "should" statements in ***Section 6.5.3***.

6.5.2 Technical Overview (Informative)

It is possible for an I3C Bus to have multiple Devices on the Bus that are capable of being an Active Controller. These Devices, which initially act as a Target but can also accept the Active Controller Role from an Active Controller, are referred to as Secondary Controllers. The following sections specify how to implement a Secondary Controller, and define how the handoff of the Active Controller Role to a Secondary Controller is completed.

Secondary Controllers are not required to complete Bus Configuration, and they can be implemented for special feature handling, power savings, or other various purposes. The minimum requirements for a Secondary Controller are specified in ***Section 6.5.3.3.1*** and ***Section 6.5.3.3.2***.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.5.3 Technical Specification

An I3C Bus may have multiple Controller-capable Devices, with varying capabilities and purposes. All Controller-capable Devices shall support the required Bus Management procedures while acting as Active Controller. Although only one I3C Controller-capable Device can act as Active Controller of the I3C Bus at any given time, the role of Active Controller may be passed between Controller-capable Devices, using the Controller to Controller Handoff Procedure (see ***Section 6.5.3.2***). The state of a Controller-capable Device currently holding the role of Active Controller on the I3C Bus is referred to as the Controller Role.

At least one Controller-capable Device shall support the required Bus Configuration procedures for a Bus. A Secondary Controller may perform some of these Bus Configuration procedures, as documented in ***Section 6.5.3.3*** depending on the specific application for the I3C Bus and the intended use cases of the Secondary Controller. A Secondary Controller that cannot, or might not, perform all required Bus Configuration procedures should defer or pass the Controller Role to another suitable Controller-capable Device if it encounters a condition that it cannot or might not handle while acting as Active Controller.

Additionally, the Controller-capable Device that has the job of initializing the I3C Bus holds the special role of Primary Controller. A Primary Controller Device is, by definition, the first Controller-capable Device to be the Active Controller on the Bus. The Primary Controller acts as the authority for the initial configuration of the Bus and all Devices, including any Legacy I^2C Devices. Therefore, the Primary Controller must be capable of all required Bus Configuration procedures. However, once the Bus has been initialized and configured, the Primary Controller can also pass the Controller Role to any other Controller-capable Device currently acting as Secondary Controller.

Using the Handoff procedure, either multiple Controller-capable Devices can cooperatively pass the Controller Role back and forth, or else one Controller-capable Device can direct the flow of passing the Controller Role among other Controller-capable Devices according to the capabilities of any I3C Secondary Controllers on the Bus and the I3C Bus's particular application.

A Secondary Controller may request the Controller Role from the Active Controller using the Controller Role Request flow detailed in ***Section 4.3.6.3***. Alternately, the Active Controller may choose a particular Secondary Controller to receive the Controller Role, even if that Secondary Controller did not request the Controller Role. In either case, the names Active Controller and Secondary Controller largely reflect the Controller-capable Device's current role at any given point in time.

After acknowledging a Controller Role Request or choosing a Secondary Controller to receive the Controller Role, the Active Controller should:

1. Prepare the Bus and the indicated Secondary Controller for Handoff (see ***Section 6.5.3.1***);
2. Pass the Controller Role to the indicated Secondary Controller using the Controller to Controller Handoff Procedure (see ***Section 6.5.3.2***); and
3. Monitor the Bus to ensure that the indicated Secondary Controller asserts its Controller Role (see ***Section 4.3.8.2.4*** on Error Type CE3).

The process for passing the Controller Role is the same in either direction: it is always initiated by the Active Controller (i.e., the Controller-capable Device holding the Controller Role at that point in time). In most cases the immediately previous Active Controller can request to receive the Controller Role back again, or the indicated Secondary Controller can automatically pass the Controller Role back once it has finished its tasks. However, in some Bus implementations it might be advisable to designate one Controller-capable Device to direct and manage the flow of passing the Controller Role to/from any Secondary Controllers.

While acting as Active Controller, a Secondary Controller may choose not to handle any Bus Configuration procedures. In some circumstances, a Secondary Controller might be intended for a limited purpose or scope on a particular I3C Bus, and as a result might limit its activities while serving as Active Controller to some subset of what it is capable of performing; for example, by not supporting Bus Configuration procedures.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.5.3.1 Preparing for Handoff

It may be necessary for the Active Controller to issue one or more commands on the I3C Bus to prepare other I3C Target Devices for the Handoff and ensure an optimal transition of the Controller Role, irrespective of whether the handoff is due to a Secondary Controller requesting the Controller Role via a Controller Role Request, or is due to the Active Controller having simply chosen to pass the Controller Role to a particular Secondary Controller.

There are many possible variations of the steps that the Active Controller should take to prepare for a Handoff. ***Figure 148*** presents a generalized flow, and this Section provides a recommended procedure.

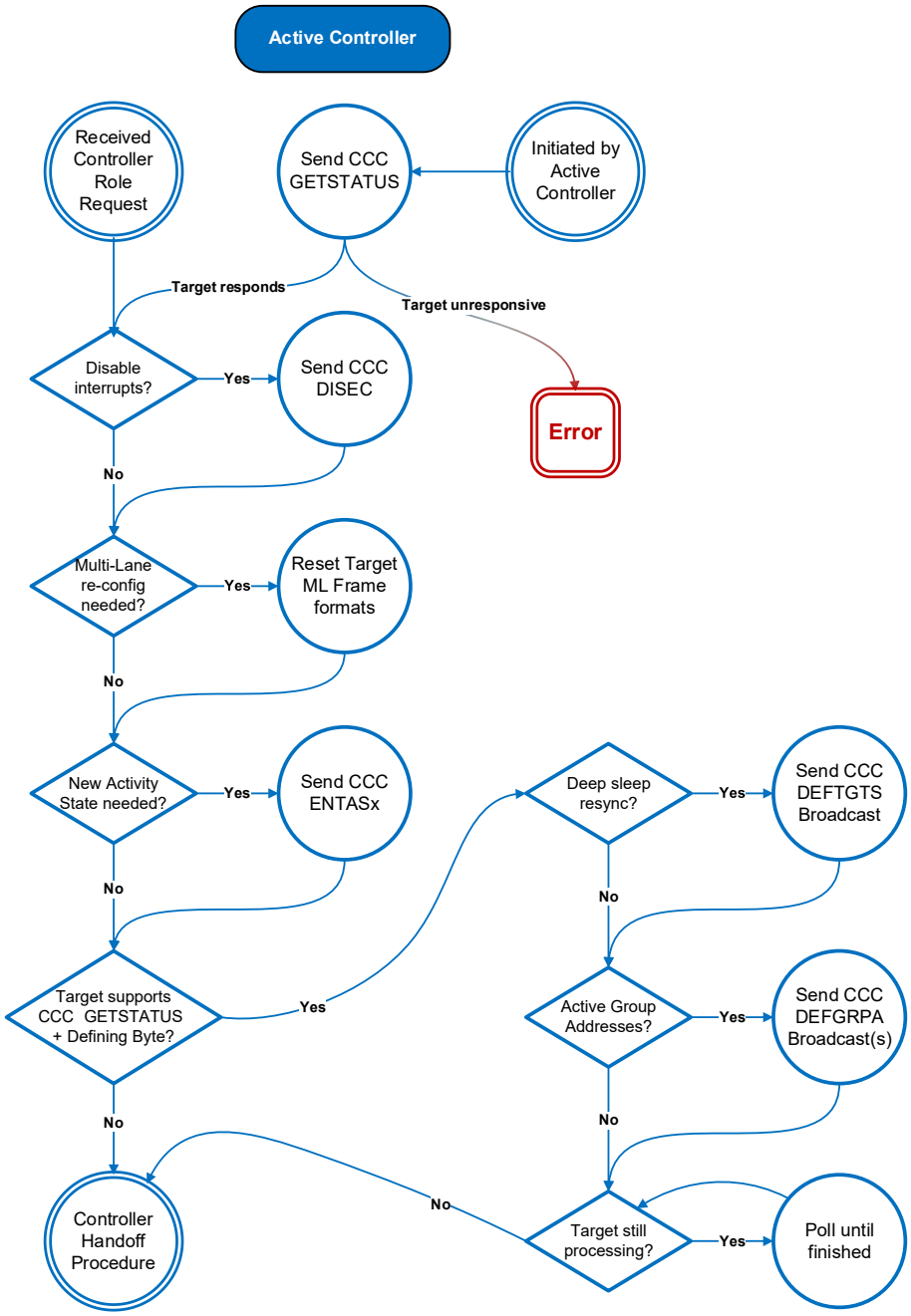


Figure 148 Pre-Handoff Steps Flow

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Recommended procedure:

1. If the Active Controller intends to pass the Controller Role to a selected Secondary Controller that did not send a Controller Role Request, then the Active Controller should verify that the selected Secondary Controller is active and ready to respond to additional commands (see ***Section 4.3.7.3.15***, the GETSTATUS CCC).

   Note:

   *If the Secondary Controller responds to the GETSTATUS CCC and returns a value of 2'b11 in bits 7:6 (i.e., LSB) of the GETSTATUS Format 1 CCC (see **Table 27**), then this indicates that it is not able to participate in any subsequent steps to prepare for Handoff. The Active Controller should stop the procedure at this point, and may initiate an error recovery procedure.*

2. If the Active Controller needs to disable Hot-Joins, Target Interrupt Requests, or other Bus events that could interfere with the Handoff, then it should send the appropriate Broadcast or Direct CCCs to disable those events before the Handoff (see ***Section 4.3.7.3.1***, the ENEC/DISEC CCCs).

   Note:

   Once the Handoff is complete, the new Active Controller should re-enable any events that are disabled in this step.

3. If the Active Controller is Multi-Lane capable, and does not have knowledge that the selected Secondary Controller supports and is configured to use the same Multi-Lane Modes and codings that any Multi-Lane capable Target Devices have been configured to use, then the Active Controller should re-configure the I3C Bus such that all Target Devices are configured to use an ML Frame format that is known to be supported by the selected Secondary Controller (see ***Section 6.7.3.6***, the MLANE CCC).

   Note:

   In some situations, ensuring maximum compatibility and preventing communication errors after Handoff might require resetting these Multi-Lane capable Target Devices to fundamental 2-Wire I3C Mode (i.e., 1-Lane).

4. If the Active Controller knows that the selected Secondary Controller must be put into a different Activity State before Handoff, then the Active Controller shall send the appropriate Broadcast or Direct CCCs to put the Bus (or selected Devices) into a different Activity State (see ***Section 4.3.7.3.18*** regarding the GETMXDS CCC with optional Defining Byte, and ***Section 4.3.7.3.2***, the ENTASx CCCs).
5. If the Active Controller determines that the indicated Secondary Controller has been in a "deep sleep" state and may need to be re-synchronized with the most current list of I3C Targets and Group Addresses, then the Active Controller should send the appropriate Broadcast CCCs (see ***Section 4.3.7.3.7***, the DEFTGTS CCC, and ***Section 4.3.7.3.26***, the DEFGRPA CCC). Afterwards, the Active Controller should poll the Secondary Controller to ensure that it has successfully processed this data and indicates that it is ready to accept the Controller Role (see ***Section 4.3.7.3.15*** regarding the GETSTATUS CCC with optional Defining Byte).

While steps 2–4 can generally be performed in any order, step 5 (or any other steps requiring re-synchronization of data) should be done after steps 1–4, and immediately before checking for the Secondary Controller's status in processing the Broadcast data (if this is supported and indicated).

Figure 148 presents a generalized flow of the steps the Active Controller should take to prepare for a Handoff.

Some of these steps require the Active Controller to determine some of the optional Secondary Controller features and capabilities advertised by the Secondary Controller. In most cases, these can be determined by reading the data returned by the Secondary Controller in response to various CCCs with a Defining Byte (see ***Section 4.3.7.3.18*** regarding the GETMXDS CCC with optional Defining Byte, and ***Section 4.3.7.3.19*** regarding the GETCAPS CCC with optional Defining Byte):

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

- If the Active Controller has not already gathered this information and needs to determine whether the Secondary Controller requires special handling based on these features and capabilities, then it shall do so before taking any related steps that would require this information.
- However, if the Active Controller has already gathered this information in advance, then it may rely on cached data, and does not need to gather it again.
- Once all the data has been gathered, the Active Controller must make its decisions relating to the pre-Handoff flow steps, based on the data returned from the Secondary Controller.

When the Active Controller has verified that the Bus is configured correctly, and the selected Secondary Controller indicates that it is ready to accept the Controller Role, the Active Controller shall initiate the Controller to Controller Handoff Procedure per ***Section 6.5.3.2*** (next).

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.5.3.2 Controller to Controller Handoff Procedure

After the Active Controller has prepared for Handoff, the Active Controller shall then issue a GETACCCR CCC (***Section 4.3.7.3.16***) followed by a STOP if the Handoff was successful, whereupon it shall release control of the SCL line, and therefore also releasing control of the I3C Bus (i.e., passing the Controller Role) to the selected Secondary Controller.

Following the STOP and Bus Available Condition, the selected Secondary Controller assumes the role of the Active Controller and takes control of the I3C Bus. The former Active Controller should monitor the I3C Bus to ensure that the new Active Controller asserts its Controller Role, according to the flow described in ***Section 4.3.8.2.4*** regarding Error Type CE3.

Figure 89 presents a simplified and generalized picture of the Controller to Controller Handoff procedure. While there are many possible variations to this procedure, the most significant states are:

1. At the end of data transfer from a successful GETACCCR CCC, the current Active Controller (indicated by '**CCR**' in the Figure) controls SCL and SDA, and the new Controller capable Device that will become the new Active Controller (indicated by '**NCR**' in the Figure, initially acting as a Secondary Controller) has SCL and SDA in High-Z state.
2. The **CCR** provides the rising edge of SCL, while keeping SDA Low
3. After t_{SU_STO} elapses, the **CCR** 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 **NCR** determines that SDA is High, and after both its Clock to Data Turnaround Time (t_{SCO}) and the time of flight elapse, it then takes two actions:
   A. The **NCR** actively drives SCL High, overlapping with the active drive of the **CCR**; and
   B. The **NCR** enables its Open Drain class Pull-Up, in parallel with the Open Drain class Pull-Up of the **CCR**

   Neither of these actions conflicts with the **CCR**.

   Note:

   *Although t_{SCO} generally applies to any Target Device, in this step it applies to the **NCR** because prior to the Bus transfer the **NCR** performs in the Target role (i.e., as a Secondary Controller unit until it receives the Controller Role).*

   *In the Figure, t_{SCO} is shown starting as if the **CCR** has used the Open Drain drive of SDA High. This illustrates the worst condition for this stage of the Handoff, showing the latest possible moment when the **NCR** starts overlapping the line controls with the **CCR**.*
5. After a time delay of $t_{CRHPOverlap}$, the Active Controller (i.e., the **CCR**):
   A. Releases SCL to High-Z; and
   B. Disables its Open Drain class Pull-Up on SDA and sets SDA to High-Z

   Note:

   *In the Figure, $t_{CRHPOverlap}$ is shown in the worst case: on the Open Drain drive of SDA, and where the **CCR** starts counting it from the lower side of the SDA rising edge. The **CCR** must control the lines until the **NCR** could safely take over, and that is certain after $t_{DIG_OD_L}$ Min. As a result, $t_{CRHPOverlap}$ is greater than or equal to $t_{DIG_OD_L}$ Min.*
6. After $t_{NEWCRLock}$ (i.e., the time interval during which the **NCR** shall not drive SDA Low), the **NCR** may actively drive SDA Low, producing a START.

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

   Note:

   In the Figure, $t_{NEWCRLock}$ is shown as starting at the SDA rising edge in Open Drain driving mode, similar to I3C's t_{AVAL} parameter and I^2C's t_{BUF} parameter.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

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

   Following this SCL falling edge, the new Active Controller shall actively drive SCL, while also driving SDA in Open Drain mode with the Arbitrable Address.

After the Handoff procedure, the new Active Controller (i.e., **NCR** in the steps above) should issue a START to assert its Controller Role within 100 µs, or the time interval indicated by its Activity State (as reported by the GETMXDS CCC with optional Defining Byte, per ***Section 4.3.7.3.18***), whichever is greater. If the new Active Controller has not issued a START, then it shall respond to SDA being pulled Low by pulling SCL Low; this is a START, and the new Active Controller may follow it with any valid Bus activity that is sufficient to assert the Controller Role.

However, if the new Active Controller does not pull SCL Low within 100 µs of SDA being pulled Low, then it might lose the Controller Role to the former Active Controller (i.e., **CCR** in the steps above). To avoid a hung Bus, the former Active Controller should detect whether the new Active Controller has taken over the Bus per the Error Type CE3 flow described in ***Section 4.3.8.2.4***.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.5.3.3 Secondary Controller Functions

A Secondary Controller Device initially acts as a Target when it joins the Bus (either at Bus Initialization, or via Hot-Join) and, like any other Target, shall respond to the standard (i.e., Required) CCCs.

A Secondary Controller shall indicate that it is a Controller-capable Device, by returning a value of 2'b01 in its BCR Device Role bits (see ***Table 7***) when queried by the Active Controller. The Active Controller shall use this value to determine whether any Secondary Controllers are on the Bus, so it knows whether it is obligated to send out other CCCs (such as DEFTGTS or DEFGRPA) that contain Broadcast data that the Secondary Controller needs to use when it becomes Active Controller.

Some Secondary Controllers may require special handling before becoming Active Controller, as listed in the steps to prepare for Handoff (see ***Section 6.5.3.1***). Therefore, such Devices must advertise certain capabilities or status to the Active Controller, by supporting one or more optional Defining Bytes for the following CCCs:

- GETSTATUS Format 2 CCC with the PRECR Defining Byte (see ***Section 4.3.7.3.15***)
- GETMXDS Format 3 CCC with the CRHDLY Defining Byte (see ***Section 4.3.7.3.18***)
- GETCAPS Format 2 CCC with the CRCAPS Defining Byte (see ***Section 4.3.7.3.19***)

A Secondary Controller may automatically enter lower-power modes or a “deep sleep” state based on inactivity, or via commands from other connected logic. Such a Secondary Controller should be capable of returning to an active state when it detects and responds to certain CCCs sent by the Active Controller, since the Active Controller may check to ensure that the Secondary Controller is online and ready to receive the Controller Role (see ***Section 4.3.8.2.5***) before starting the process of preparing for Handoff (see ***Section 6.5.3.1***). This is especially important for situations where the Active Controller is preparing to pass the Controller Role, but the Secondary Controller did not send a Controller Role Request.

For such situations, the Secondary Controller must return from its lower-power mode or “deep sleep” state, respond to CCCs sent by the Active Controller to prepare for Handoff, and remain ready to receive the Controller Role, unless it is constrained from doing so. If the Secondary Controller is constrained or unable to participate, then it may indicate this status by responding to the GETSTATUS Format 1 CCC (see ***Section 4.3.7.3.15***) and returning a value of 2'b11 in bits 7:6 (see ***Table 27***). However, if the Secondary Controller is able to participate and it supports the GETSTATUS Format 2 CCC with the PRECR Defining Byte, then it shall treat this CCC and Defining Byte as a definitive indication that the Active Controller intends to pass the Controller Role in the near future.

A Secondary Controller may initiate a Controller Role Request (see ***Section 4.3.6.3***) to the Active Controller. A Secondary Controller may also accept the Controller Role from any Active Controller (including the Primary Controller), at which point it becomes the new Active Controller.

Once granted control of the Bus by the previous Active Controller via the Controller to Controller Handoff procedure, the Secondary Controller maintains control until another Controller-capable Device is granted Bus control. After the Secondary Controller transitions to the being the Active Controller, it could encounter Bus management activities besides the data transfers that it initiates itself, including: In-Band Interrupts, Hot-Join Requests, and Controller Role Requests from other Controller-capable Devices such as another Secondary Controller or the previous Active Controller.

A Secondary Controller shall support the scenarios described in the following subsections ***6.5.3.3.1*** through ***6.5.3.3.9***, according to its capabilities.

Per ***Section 6.5.3.3.3***, the Secondary Controller may choose to perform some or all Bus Configuration procedures while acting as Active Controller. And per ***Section 6.5.3.3.4***, the Secondary Controller may also choose ***not*** to perform any Bus Configuration procedures while acting as Active Controller. In either case, if the Secondary Controller receives a request (e.g., a Hot-Join Request) that it elects not to handle, then it may choose to defer the request by sending a DISEC CCC (***Section 4.3.7.3.1***), and may also immediately pass the Controller Role along to a more capable Controller that can handling that request. Any Secondary Controller acting as Active Controller may still freely use Bus Management procedures, in addition to the ENEC/DISEC CCCs (***Section 4.3.7.3.1***), to enable or disable requests on the Bus.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.5.3.3.1 Hardware and Software Requirements

A Secondary Controller capable of acting as Active Controller shall have the following minimum hardware and software capabilities:

1. Memory for:
   A. All Bus Devices' capabilities and functions at system level
   B. Retaining the Dynamic Addresses of all I3C Bus Devices
   C. Retaining the data transfer protocol that the I3C Bus Devices are capable of
   D. Retaining the active Group Addresses, as well as the lists of all I3C Bus Devices assigned to these Group Addresses, if applicable and supported
   E. Retaining the currently-configured Multi-Lane Modes and codings of all I3C Bus Devices that support Multi-Lane, if applicable and supported
2. Link requirements for Targets that are intended to transmit In-Band Interrupt requests (IBI), e.g., the clock speed and the maximum length of data transfer
3. Data transfer protocol capability needed for communicating to all I3C Bus Devices

6.5.3.3.2 Minimum Bus Management Procedures

A Secondary Controller capable of acting as Active Controller shall perform the following minimum Bus Management procedures:

1. During Bus initialization, if a Secondary Controller is connected to the I3C Bus, then it shall monitor the Bus for the Primary Controller to send a DEFTGTS CCC. A Secondary Controller shall also monitor the Bus for any Active Controller to send a subsequent DEFTGTS CCC containing any updates as a result of Dynamic Address changes, or of subsequent Hot-Join Requests. Using the most recent Broadcast, the Secondary Controller shall acquire the Addresses and Characteristics of all Devices on the I3C Bus from the data in that DEFTGTS CCC (see ***Section 4.3.7.3.7***).

   A Secondary Controller may also monitor the Bus during Dynamic Address assignment (see ***Section 4.3.4.2***).
2. Manage the In-Band Interrupt procedure. The Secondary Controller needs to know the meaning of the interrupt from the Target. If the Secondary Controller accepts the IBI, then the Secondary Controller shall service the IBI appropriately (see ***Section 6.5.3.3.5***).
3. Procedure for requesting transfer of Bus control to another Controller-capable Device, including the Primary Controller or any previous Active Controller. The Secondary Controller uses the GETACCCR CCC (Get Accept Controller Role, ***Section 4.3.7.3.16***). The Secondary Controller should also handle Controller Role Requests from another Controller-capable Device.

In addition, while acting as Active Controller a Secondary Controller may also utilize other CCCs to change the state of the Bus, or the state of any Devices, such as entering an Activity State, or disabling Target Events such as Interrupts or Controller Role Requests. If a Secondary Controller does so, then before passing the Controller Role to another Controller-capable Device the Secondary Controller should restore the state of the Bus, or the state of any Devices, to a prior state, or to a state known to be compatible with the prior state.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.5.3.3.3 Bus Configuration Procedures

A Secondary Controller capable of changing the Bus Configuration while acting as Active Controller may perform any of the following Bus Configuration procedures:

1. Perform the Dynamic Address Assignment procedure for Hot-Join Devices:
   A. In order to be able to assign the correct priority level to the Hot-Join Device, the Secondary Controller needs to know all Bus Devices' functionality at the system level.
   B. The Secondary Controller may also perform the Dynamic Address Assignment procedure while acting as Active Controller, without a Hot-Join event. It may also reset a Device's Dynamic Address, or directly assign a Dynamic Address to a Device for any other reason.
   C. If the Secondary Controller changes the Bus Configuration for the above reasons while acting as Active Controller, then it shall issue the DEFTGTS CCC (Define List of Targets, ***Section 4.3.7.3.7***) to ensure that other Controller-capable Devices have the same knowledge.
2. Manage Group Address assignments for other I3C Devices that support Group Addressing, if applicable and supported.

   To support Group Addressing, a Secondary Controller uses the following CCCs:
   - SETGRPA (Set Group Address), ***Section 4.3.7.3.24***
   - RSTGRPA (Reset Group Address), ***Section 4.3.7.3.25***
   - DEFGRPA (Define List of Group Addresses), ***Section 4.3.7.3.26***

   The Secondary Controller shall also monitor the Bus for an Active Controller to send DEFGRPA CCCs containing Group Address data, comprising lists of the I3C Target Devices assigned to each known Group Address, if any Group Addresses are known and reported in a DEFTGTS CCC.

   If the Secondary Controller changes any Group Address assignments while acting as Active Controller, then it shall send the appropriate Group Address data using the DEFTGTS (***Section 4.3.7.3.7***) and DEFGRPA (***Section 4.3.7.3.26***) CCCs to ensure that other Controller-capable Devices have the same knowledge.
3. Configure other Multi-Lane capable I3C Devices to use any supported Multi-Lane Modes and codings while acting as Active Controller, if applicable and supported. The Secondary Controller uses the MLANE CCC for this (Multi-Lane Data Transfer Control, ***Section 6.7.3.6***).

A Secondary Controller that can perform any of these Bus Configuration procedures shall advertise its support for these features and capabilities by providing the appropriate bit values in the CRCAP1 byte that is returned when an Active Controller sends the Direct GETCAPS Format 2 CCC with the optional CRCAPS Defining Byte (Secondary Controller Capabilities, see ***Section 4.3.7.3.19***).

In some cases, a Secondary Controller might not need to assign or change any Dynamic Addresses for any Devices, nor assign nor change Group Addresses for any Devices, nor change the Multi-Lane configurations of any Devices, while acting as Active Controller. The decision on whether to use these Bus Configuration procedures could depend upon the Secondary Controller's available features and capabilities, upon instructions or settings received from another Controller (including the Primary Controller), or upon the specific use cases required for that Secondary Controller within the Bus.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.5.3.3.4 Reduced Functionality Secondary Controllers

Secondary Controllers are not required to implement or perform any Bus Configuration procedures. The Secondary Controller may defer or transfer the Bus control to another Controller-capable Device when needed; see ***Section 6.5.3.3.1*** and ***Section 6.5.3.3.2***.

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

1. Memory sufficient for retaining Device Addresses and Characteristics of any Targets it supports
2. Memory sufficient for retaining the Device Address and Characteristics of the Controller to which it will return Bus control

6.5.3.3.5 In-Band Interrupt Handling

A Secondary Controller may either accept an In-Band-Interrupt request from a Target, or refuse it.

Since the Target emits its Address in the Arbitrated Address header, the Secondary Controller may base the decision to accept or reject the IBI based on the Address (per ***Section 4.3.6***) and/or the Secondary Controller's intended use case. If a Secondary Controller elects to accept an IBI from a particular Target, then it shall service that IBI request normally per ***Section 5.1.6.2***.

Some Secondary Controllers may elect to simply refuse IBIs from all Addresses, whereas other Secondary Controllers may base the decision on the Address in each IBI request. If a Secondary Controller elects to refuse an IBI from a particular Target for any reason, then it should issue the DISEC CCC to the interrupting Target, with the DISINT bit set (Disable Target Events Command, ***Section 4.3.7.3.1***), and then defer IBI handling to another capable Controller.

A Secondary Controller that refuses IBIs from all Target Addresses should also issue a Broadcast DISEC CCC with the DISINT bit set, to disable interrupts from all Targets. To avoid Target issues that might arise from an IBI not being accepted and serviced in a timely manner, a Secondary Controller that refuses an IBI should also plan to pass the Controller Role to another capable Controller within a reasonably short time after refusing that IBI.

6.5.3.3.6 Hot-Join Management

A Secondary Controller may either support Hot-Join Requests, or not support them.

If a Secondary Controller supports Hot-Join Requests, then it may either manage the Hot-Join Request while acting as Active Controller per ***Section 4.3.5***, or else defer management to a capable Controller by issuing a Broadcast DISEC CCC to disable Hot-Join Requests by setting the DISHJ bit (Disable Target Events Command, ***Section 4.3.7.3.1***).

If a Secondary Controller does not support Hot-Join Requests, then it shall deny any Hot-Join Requests and defer management to another capable Controller.

6.5.3.3.7 Group Address Management

A Secondary Controller may either support Group Address commands, or not support them.

If a Secondary Controller supports Group Address commands, then it can participate in the transfer of Group Address data and manage the Group Addresses of I3C Target Devices as described in ***Section 4.3.4.4***. If the Secondary Controller determines that it needs to perform this function while acting as Active Controller, then it may use the Group Address commands to set or reset Group Addresses for any I3C Targets that support Group Addressing. The Secondary Controller may also participate in the transfer of Group Address data to another Controller-capable Device using the DEFGRPA CCC (Define List of Group Addresses, ***Section 4.3.7.3.26***).

If a Secondary Controller does not support Group Addressing, then it shall not send these CCCs and cannot participate in the transfer of Group Address data, either before or after Handoff.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

If the Controller Role passes through a Secondary Controller that does not support Group Address management, and that does not use the DEFGRPA CCC to pass on any Group Address data received from the previous Active Controller to a future Controller, then it shall be the responsibility of either the previous Active Controller, or of a future Controller, to resolve any issues relating to this Secondary Controller's lack of support for Group Addressing.

6.5.3.3.8 Multi-Lane Management

A Secondary Controller may either support Multi-Lane Data Transfer, or not support it.

If while acting as Active Controller a Secondary Controller that supports Multi-Lane Data Transfer determines that it is necessary to configure other Multi-Lane capable I3C Devices as described in ***Section 6.7***, then it may do so. In this situation, the Secondary Controller shall rely upon the Active Controller to have configured the I3C Bus, and all Multi-Lane capable I3C Devices, to use a compatible Multi-Lane Frame format before the Secondary Controller accepts the Controller Role.

However if a Secondary Controller does not support Multi-Lane Data Transfer, then the Active Controller shall configure the I3C Bus and all Multi-Lane capable I3C Devices to use fundamental 2-Wire I3C mode (i.e., 1-Lane) before passing the Controller Role to that Secondary Controller.

6.5.3.3.9 Interoperability Among Controllers Based on Different I3C Versions

If multiple Controller-capable Devices on a given I3C Bus support different versions of the MIPI I3C Specification, then certain features or capabilities added in later I3C versions will not be supported (nor even known to exist) by Devices based on earlier I3C versions, raising potential interoperability issues.

In particular, some features and capabilities added in I3C version 1.1 may require special handling during Controller Role Handoff and the recommended Handoff preparation flow. These potential issues could impact the new Active Controller after Handoff, or any subsequent communications between a Controller and a Target.

To avoid leaving the Bus in a non-functioning state, the system designer must give the following situations careful consideration:

- **Use cases that depend upon features or capabilities introduced in I3C v1.1**, such as Multi-Lane Data Transfer, Group Addressing, or flow control in HDR Modes.

  A Controller based on an earlier version of I3C might not support these features or capabilities, and could potentially be negatively impacted if such features or capabilities were to be enabled.

- **Use cases that depend upon Target-initiated requests specified in a more recent I3C Specification version**, such as Device to Device Tunneling.

  A Controller based on an earlier version of I3C might not be able to understand and/or handle such requests.

- **Use cases that pass the Controller Role between Controllers based on different I3C Specification versions**.

  If a given Bus implementation calls for Controller Role Handoffs between multiple Controllers, and not all of those Controllers are based on the same version of the I3C Specification, then the system designer is responsible for ensuring smooth Controller Role Handoff between these Devices, without interoperability issues or loss of communications. In particular, some Controllers might require or expect special handling to prepare for a successful Handoff (see ***Section 6.5.3.1***).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.6 F005: Timing Control

This Section defines I3C Optional Feature F005: Timing Control.

Table 96 Optional Feature 005 Status

Optional Feature ID	F002 See Errata 01, Item 31
Optional Feature Name	Timing Control
Optional Feature Version	1
Optional Feature specification was last updated in I3C Spec Version	I3C v1.2
Optional Feature is not compatible with other Optional Features	–
Optional Feature is included in I3C Basic	Yes (partial)

Changes in this Version

For detailed changes in this version, see the "diff" file between I3C v1.1.1 and I3C v1.2 in the MIPI Specification Repository ***[MIPI09]***.

6.6.1 Scope and Purpose (Informative)

The scope of ***Section 6.6*** is limited to fully specifying I3C Optional Feature F005: Timing Control. An I3C Device that implements F005 is required to implement all **shall** statements in ***Section 6.6.3***, and can optionally implement any combination of **may** statements in ***Section 6.6.3***. In addition, MIPI Alliance recommends that manufacturers follow all **should** statements in ***Section 6.6.3***.

6.6.2 Technical Overview (Informative)

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

The I3C Timing Control framework provides:

- Flexible implementation with significant capability enhancements
- Means for synchronizing the timing references/clocks, and subsequently the events, of Target Devices on the I3C Bus to the timing reference/clock of the Controller

  This can result in reduced energy consumption at the system level.
- Transmission of timing information, with minimal complexity for the Target Devices
- Controllable timing accuracy, suitable for the intended use cases

I3C defines two forms of main systems and events for Timing Control: Synchronous mode and Asynchronous mode. These modes and their versions can be used independently and concurrently on a given I3C Bus, per the application-specific requirements, with only one restriction: a Target can enable only one Asynchronous mode variant at a time. If a Target is configured to one of the Asynchronous modes, and the Controller sends an unsupported Asynchrnous mode, then the Target shall remain in the previously configured mode. If Targets support the new mode, then they shall disable the previous mode and switch to the new mode. See Errata 01, Item 32

- **Synchronous Time Control** (see ***Section 6.6.3.1***): NOT INCLUDED IN I3C BASIC.
- **Asynchronous Time Control** (see ***Section 6.6.3.2***): Targets timestamp the moment at which they acquire sampled data, permitting the Controller to time-correlate samples received from multiple, different sensors in an easier and more accurate manner. Timestamping ensures that the Controller always has the time at which sensor data acquisition occurred (subject to the timing granularity

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

supported), even if there are latencies in moving the data from the Target to the Controller. Four different Asynchronous Time Control versions ('Modes') provide different methods for measuring time, and/or calibrating timing, with increasing accuracy.

Note:

I3C Basic only includes Asynchronous Time Control (Mode 0). The full I3C Specification includes additional Asynchronous Time Control Modes that provide different methods for measuring time and/or calibrating timing, with increasing accuracy.

6.6.2.1 General Principles

The exchange of timing information is based on agreements between the Controller and Targets.

The elements of this agreement are:

1. Two I3C Common Command Codes (CCCs):
   - Set Exchange Timing Information (SETXTIME), see ***Section 6.6.3.3***, and
   - Get Exchange Timing Information (GETXTIME), see ***Section 6.6.3.4***.

   These CCCs:
   - Identify Bus events and markers (i.e., specific SDA edge and/or SCL edge),
   - Transfer timing or control information (e.g., 'abort' command),
   - Specify which timing control procedure is in effect, and
   - Query the Target Device for Exchange Timing support details, such as which Timing Control Mode(s) are supported, the current Timing Control state, frequency, inaccuracy, etc.
2. One defined and Mode-specific marker is sent by the Controller, and used to synchronize the Targets
3. One data collection event is sent by the Target. It includes the time information previously requested by the Controller.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.6.3 Technical Specification

6.6.3.1 Synchronous Systems and Events

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

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

In a typical system, different Targets will sample their data at different, uncorrelated times. This is true even if all Targets are set to the same sampling frequency, because Target-to-Target oscillator accuracy differences will cause drift over time. In I3C's Synchronous Time Control mechanism, the Controller periodically emits a synchronization pulse, called a SYNC Tick. This method permits all Target data sampling to occur very close together in time, so that Targets can prepare and activate their individual sampling mechanisms, even if there are variances across their clocks.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.6.3.2 Asynchronous Systems and Events

In the full I3C Specification this section defines four variations (called Modes, summarized in ***Table 97***) of a method for increasing the precision of the acquired time-of-occurrence for an event occurring in an I3C Target connected to an I3C Controller. The overall method is called Asynchronous Timing Control. In this version of the I3C Basic Specification, only Async Mode 0 is supported.

The actual event itself might not be periodic, and the Controller and Target Devices need not use the same time base clock (source, frequency, or accuracy). The procedure can help address the large inaccuracies/offsets to the event's acquired time-of-occurrence that IBI launch/acknowledge latencies can introduce.

In I3C Asynchronous Timing Control, a Target Device timestamps events occurring within that Target, and then notifies the Controller about it by generating an In-Band Interrupt (IBI). The Target's IBI might not reach the Controller immediately, either because the I3C Bus was busy, or because the Target lost IBI Arbitration, or because the Controller did not immediately accept the IBI. The Asynchronous mechanism allows the I3C Controller to convert the Target's timestamp value to its own internal time scale, in terms of both timing resolution and timing accuracy.

Terminology

- **T_C1:** Target counter from Event occurrence to a Controller-initiated end-of-count (EOT_C1) point depending on Mode (e.g., IBI Acknowledged point in Mode 0 & Mode 1).
- **T_C2:** Target counter from EOT_C1 to a second Controller-initiated end-of-count, EOT_C2 (e.g., T-Bit of IBI Mandatory byte in Modes 0 & 1).

Figure 149 illustrates the I3C Asynchronous Timing Control model. In this model, the I3C Controller Device maintains its own notion of time ("I3C Controller Ref Counter" in the Figure) using whatever timing source and units it needs; for example, a real-time clock, or a simple 32-bit rollover timer/counter. This notion of time allows the Controller to determine the relative timing among multiple events generated by different Targets. These relative time differences can then be used to correlate the Target-generated events, for example to support sensor fusion, to plot sensor events vs. absolute time, or for other purposes.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

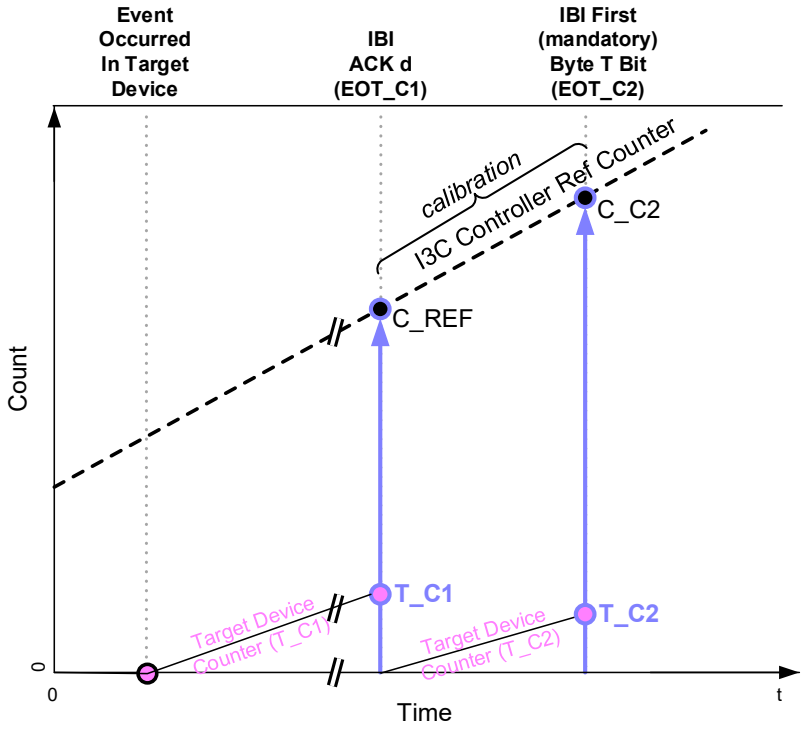


Figure 149 Graphical Representation of Async Mode 0 Timestamp Interpolation

I3C Target Devices report the occurrence of sensor events to the I3C Controller using IBI, but because the IBI can be delayed before reaching the Controller, the Controller's notion of time alone is not sufficient to accurately gauge event timing. For this reason, the Target maintains its own counter ("Target Device Counter (SC1)" in ***Figure 149***), driven by its own clock. At the time when the Target is able to generate the IBI to report the occurrence of a sensor event to the I3C Controller, the number of Target clocks elapsed since the event occurred ('T_C1' in the Figure) and the calibration period ('T_C2') shall be sent as IBI payload of the same IBI.

The Target Device counts can be converted into Controller time via a Controller-provided method whereby the Target can measure the time between two points ('EOT_C1' and 'EOT_C2' in the Figure) as T_C1, in terms of number of its clock cycles whose approximate clock frequency is known to the Controller through the GETXTIME CCC, see ***Section 6.6.3.4***). The Target then counts the time (T_C2) from EOT_C1 to EOT_C2, and then communicates both T_C1 and T_C2 to the Controller as IBI payload. This allows the Controller to correlate T_C1 and T_C2 with its own internal counter values C_REF and C_C2, respectively. The Controller knows the center clock frequency of the Target Device (again via the GETXTIME CCC), and can compare the Target-provided count with its expected value, given the known length of time reported by T_C2. The delta between the expected count and the actual count can be used as guidance to improve the accuracy of the T_C2 value, derived from the GETXTIME CCC, and thereby the accuracy of the event's calculated time-of-occurrence.

Note that the higher the frequency of the Target clock vs. the effective SCL clock frequency during the SC1-to-T_C2 measurement, the more accurate the error delta will be. This means that the Controller could gain more accuracy in the scale factor by extending the T_C1-to-T_C2 period.

The Target shall report the T_C1 value as a 2-byte (16-bit) value, transmitting the Least Significant Byte first, and then the Most Significant Byte. The Target shall report the T_C2 value as a 1-byte value. If either count overflows (which can happen with a busy Bus with a large transaction), then the Target shall report values of all 1'b1 bits: 8'hFF for T_C2, and 16'hFFFF for T_C1.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

A Target may choose to not count T_C2 value if its counter's clock source is insufficiently accurate, however in this case it shall report a value of 8'h00 for T_C2 in the IBI payload. A Controller receiving a T_C2 value of 8'h00 shall rely solely on the frequency value reported by GETXTIME for scaling. Targets having very stable clocks, especially when fast enough, could help the Controller achieve higher time-stamping accuracy by supporting and reporting the T_C2 count.

In all four Asynchronous Modes, the key points are:

1. All I3C Devices supporting any of the Asynchronous Modes shall drive a Mandatory Byte after Acknowledgement of their IBI. As a result, the BCR[2] bit for these I3C Devices shall be set (i.e., shall have the value 1'b1).

In the Mandatory Byte:

- Bits[7:5] shall be set to 3'b100, indicating that a timestamp follows.
- Bits[4:0] shall be set by the Target and read by the Controller. These bits are typically used to convey some aspect of the event, and are interpreted according to a private contract between the two Devices.

2. Two mutually identifiable Events on the I3C Bus between I3C Controller and I3C Target Devices:

- **End of T_C1 (EOT_C1):** The T_C1 latch point and the start of the Calibration Window to count T_C2.

  The actual event used depends on which of the four Asynchronous Modes is being used. It could be either the IBI ACK point of the current IBI transaction, or an SCL edge, or some other event on the I3C Bus.

- **End of T_C2 (EOT_C2):** The end of the Calibration Window.

  This event depends on which of the four Asynchronous Modes is being used. It could be either the T-Bit following the IBI's Mandatory Byte for the current IBI transaction, or an SCL edge, or some other event on the I3C Bus.

3. Additional common Events (**aBE**) are defined for other Asynchronous Modes that are not supported in I3C Basic.
4. After the Mandatory Byte, the Target shall send its timestamp values:

- **T_C1:** The 16-bit timing counter from the original sensor-generated event
- **T_C2:** An 8-bit reference/calibration count
- **aBE_TICKs:** An optional 1-byte value indicating the number of aBEs that the Target has detected, from the sensor event to the IBI-ACK. These are not defined for Asynchronous Mode 0.

The values of both T_C1 and T_C2 are in terms of the Target's time scale. The Controller will convert these values into its own time scale, taking into consideration the Target's counter clock frequency as reported by the GETXTIME CCC.

In cases where the aBE event is used, the Controller then can calculate when the event occurred by taking the timestamp of the aBE event indicated by the Target (by count) and subtracting the time in T_C1 (translated to Controller time resolution).

In order to support a wide range of applications, several degrees of accuracy are required for such timing related information. As a result, I3C provides four different Asynchronous Modes which are summarized in ***Table 97*** and detailed in the following subsections. Each of these methods addresses a different level of accuracy, and presents a distinct complexity tradeoff option for the Controller and/or Target. The SETXTIME CCC (see ***Section 6.6.3.3***), is used to enter all four Asynchronous Modes, using the Sub-Command Byte field to select the desired Mode.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Note:

The full I3C Specification defines additional Timing Control modes that present additional choices, including different levels of accuracy and distinct complexity tradeoff options for the Controller and/or Target. I3C Controllers that support the full I3C Specification may optionally use the same SETXTIME CCC to enter these other Timing Control modes, and I3C Targets that support the full I3C Specification might optionally implement support for such Timing Control modes. However, such Timing Control modes shall not be supported by or enabled by any I3C Devices that support the I3C Basic Specification.

These other Timing Control modes are not included in I3C Basic. To gain access to these other Timing Control modes, please contact MIPI Alliance.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

Table 97 Asynchronous Timing Control Modes

	Async Mode 0
	Asynchronous Basic Mode
See Section	**6.6.3.2.1**
Description	Simple for I3C Controller and I3C Target Allows all I^2C out-of-band interrupt usages to be replaced with I3C In-Band Interrupt
Required	Mandatory for Controller Optional for Target
SETXTIME Sub-Command to Enter Mode Sub-Command Byte Value	Enter Async Mode 0 8'hDF
Clock Used for Target's Timer Counter	Local clock in Target
Sensor Sample	T0
EOT_C1	In-Band Interrupt ACK Bit
EOT_C2	T-Bit following the Mandatory Byte after ACK of In-Band Interrupt
aBE	N/A

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.6.3.2.1 Async Mode 0: Asynchronous Basic Mode

All I3C Controller Devices capable of Asynchronous Timing Control procedures shall support Async Mode 0, Basic Asynchronous Timestamping. This Mode expects that a reasonably accurate and stable clock source is available in the Target to drive the timer counter. Note that in this Mode, the two SCL edges both occur after the Target and the Controller agree that the transaction was valid.

To select Async Mode 0, the Controller sends the SETXTIME CCC (see ***Section 6.6.3.3***) with the Sub-Command Enter Async Mode 0 (0xDF).

Terminology

- **EOT_C1:** First SCL positive-going edge after the IBI ACK for the Target Device
- **EOT_C2:** First SCL positive-going edge after the T-Bit of the IBI's Mandatory Byte
- **aBE:** Not used

Figure 149 graphically depicts Async Mode 0 timestamp data transfer from the Target to the Controller, allowing the Controller to calculate the time at which sensor data sampling has taken place. It is presumed that the Target is able to transfer the data within a time interval that the Target's timing counter can meaningfully measure (i.e., that the clock driving the timing counter preserves a sufficiently stable frequency to increment it in a substantially linear manner with respect to time).

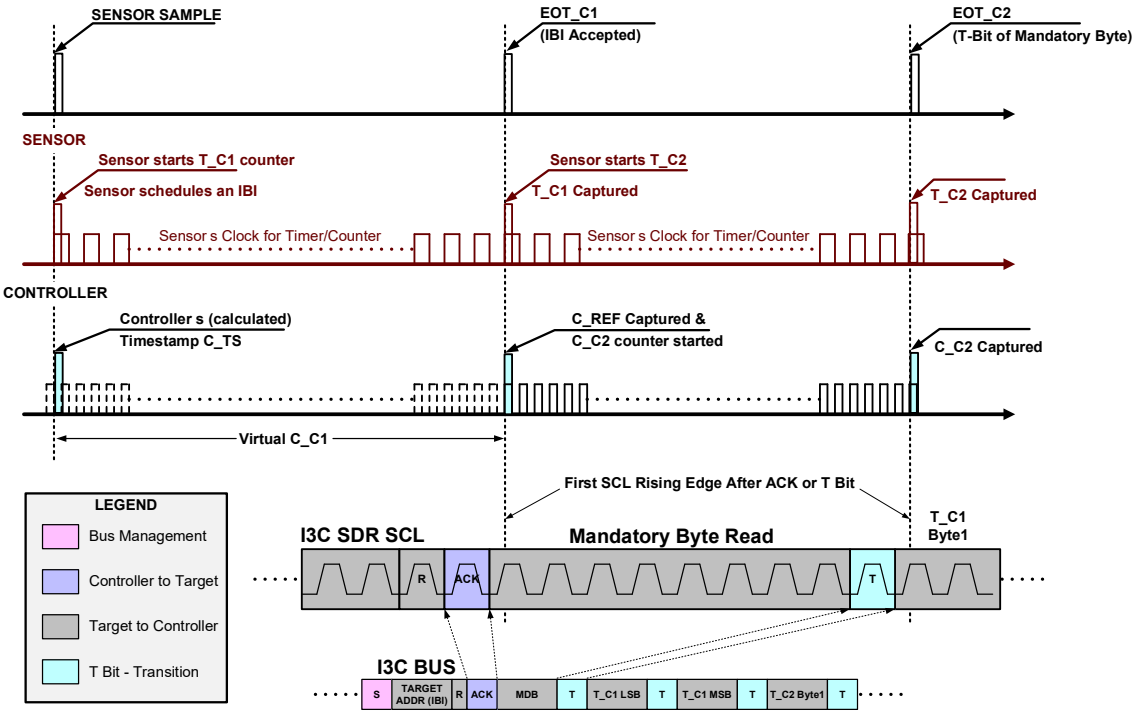


Figure 150 Example of Asynchronous Mode 0 Timestamp Data Transfer

Figure 150 illustrates the basic sequence of operations during a simple asynchronous time information transfer:

1. The top black baseline shows the events visible outside of the Target and the Controller
2. The middle (red) baseline shows the activity inside the Target's timing counters
3. The bottom black baseline presents the virtual and real activity inside the Controller's timing counter

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

In more detail, ***Figure 150*** illustrates that:

4. At sensor sampling/event time, the Target shall start its internal timing counter, and initiate the IBI request.
5. If the IBI request is accepted, then:
   A. Immediately:
   - The Target shall record its internal timing counter, T_C1
   - The Controller shall record its internal timing counter, C_REF

   B. At the T-Bit of the IBI payload's mandatory byte:
   - The Target shall record its internal timing counter, T_C2
   - The Controller shall record its internal timing counter, C_C2

   C. The Target shall transfer the recorded T_C1 & T_C2 values in the prescribed order.
6. If the IBI request is not accepted, then:
   - The Target shall continue to increment its own counter for T_C1
   - The Target shall wait for the next opportunity for the IBI request to be issued & accepted
   - When the IBI request is eventually accepted, the Target shall proceed as described in step 2 above.
7. Evaluating the timing counter data:

If the numbers are non-zero, then the Controller calculates the real time interval based on its internal timing counter, using the formula:

$$C_TS = C_REF – C_C2 * T_C1/T_C2$$

- If T_C1 is zero, then no time information is assigned to the event
- If T_C2 is zero, then the scaling shall be done based on the Target Device counter's clock frequency as can be read with GETXTIME CCC.

Note that the above method of recovering the target event's timestamp is only suitable for periods during which the Target clock is appropriately stable. The calibration time available is the duration elapsed between EOT_C1 and EOT_C2, which in this mode is 9-bit durations on the I3C Bus for the Mandatory Byte. To enable better calibration for Targets using low-frequency clocks, the Controller may optionally use clock Stalling during the Mandatory Byte transfer, thus lengthening the calibration window; however, clock Stalling has the disadvantage that IBIs will consume more I3C Bus time.

If better calibration is required and clock Stalling is not an option, then other, more refined procedures or other Async Modes can be used.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.6.3.2.2 Async Mode 1: Asynchronous Advanced Mode

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

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

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

6.6.3.2.4 Async Mode 3: Async High-Precision Triggerable Mode

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.6.3.3 Set Exchange Timing Information (SETXTIME)

The SETXTIME CCC provides the framework for Controller(s) and Target(s) to exchange event timing information for purposes such as synchronizing controls, collecting or reconstructing timestamps, and specifying the timing data procedure. The SETXTIME CCC is available both as a Broadcast Command (***Figure 151***) and as a Direct Command (***Figure 152***).

The SETXTIME CCC always includes the one-byte Sub-Command field, which acts as an identifier that defines the specific function of the CCC. For some Sub-Command values, additional Data Bytes follow. Interpretation of these additional Data Bytes depends upon the particular Sub-Command value, as detailed in ***Table 99***.

Table 98 Set Exchange Timing Information Command Codes (SETXTIME)

Command	Support	Command Codes		Purpose
		Broadcast	Direct	
SETXTIME	Conditional	0x28	0x98	Set Exchange Timing Information

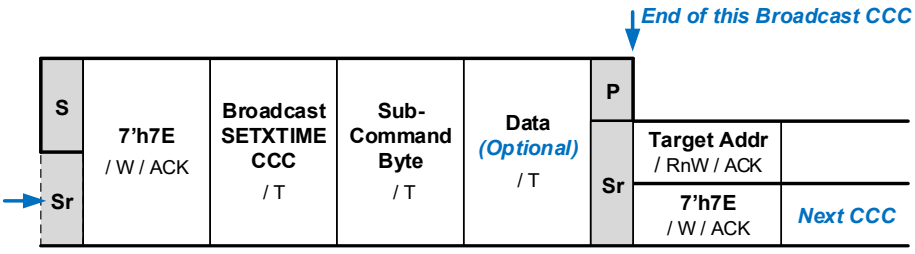


Figure 151 SETXTIME Format 1: Broadcast

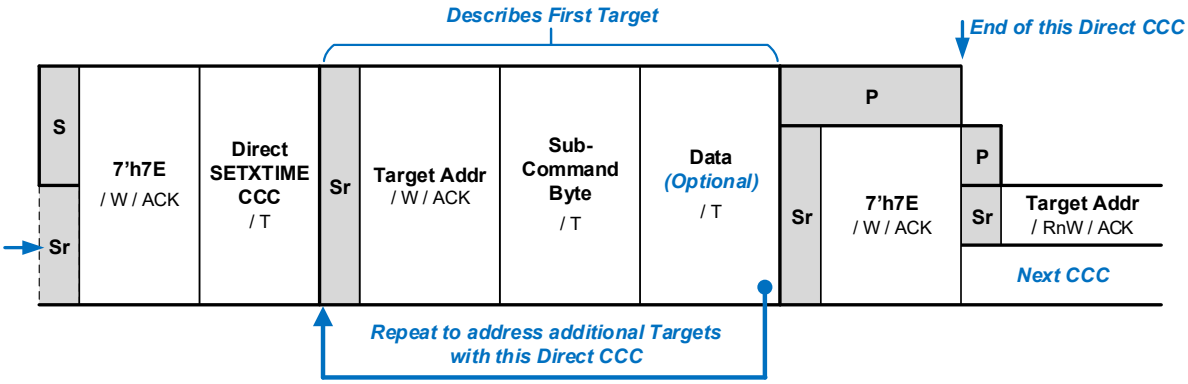


Figure 152 SETXTIME Format 2: Direct

Note:

In previous versions of the I3C Specification, the Sub-Command Byte was named the "Defining Byte". In version 1.1 the field name has been changed to avoid confusion, as the term "Defining Byte" is now reserved for use solely in Direct CCCs (see ***Section 4.3.7.1*** *and* **Section 4.3.7.2.2***). Only the field name has been changed, the usage and data format remain exactly the same as before.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

7020 **Table 99 SETXTIME Sub-Command Byte Values**

Sub-Command Value	Sub-Command Function	Description	Additional Data Bytes
0x7F	**Sync Tick 'ST'**	These capabilities are not included in the I3C Basic Specification. To gain access to these capabilities, please contact MIPI Alliance.	None
0xBF	**Delay Time 'DT'**		One byte
0xDF	**Enter Async Mode 0**	Commands all I3C Targets and Controllers on the Bus to operate in Async Basic Mode. See ***Section 6.6.3.2.1***. Anchor points: In-Band Interrupt ACK and T-Bit after the first In-Band Interrupt Payload byte. Upon receiving this Sub-Command, all I3C Devices shall reset all internal counts. All I3C Controllers that support Timing Control shall support Async Mode 0.	None
0xEF	**Enter Async Mode 1**	These capabilities are not included in the I3C Basic Specification. To gain access to these capabilities, please contact MIPI Alliance.	None
0xF7	**Enter Async Mode 2**		None
0xFB	**Enter Async Mode 3**		None
0xFD	**Async Trigger** (for Async Mode 3)		None
0x3F	**TPH**		One or more bytes
0x9F	**TU**		One byte
0x8F	**ODR**		One byte
0xFF	**Exit**	Commands all I3C Targets to exit from the enabled timing modes, or to turn timing control off entirely.	None
All Others	**Reserved**	Available Sub-Command values for future definition of new SETXTIME CCC Sub-Commands	(For future definition)

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.6.3.4 Get Exchange Timing Support Information (GETXTIME)

This Direct CCC (***Figure 153***) provides the framework for the Controller to query the Exchange Timing capabilities supported by the I3C Targets. The GETXTIME CCC causes the addressed Target to return four data bytes containing key information on supported Timing Control modes, current state, and internal oscillator/clock frequency and inaccuracy.

The Command Code for the GETXTIME Direct CCC is 0x99. Support for this CCC is required if an I3C Target supports any Timing Control modes.

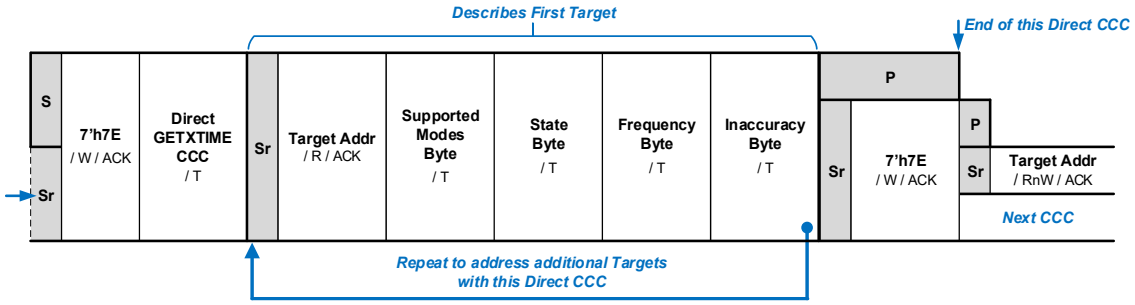


Figure 153 GETXTIME Format

The queried I3C Target returns four data bytes, with the following interpretations:

- **Supported Modes Byte:** Bit mask indicating which Timing Control Mode(s) the Target supports (see ***Table 100***). If a bit is set (has value 1'b1), then that Target supports the corresponding Timing Control Mode.

Table 100 GETXTIME Supported Modes Byte Fields

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	Reserved	Reserved	Supports Async Mode 3 ***Not included in I3C Basic***	Supports Async Mode 2 ***Not included in I3C Basic***	Supports Async Mode 1 ***Not included in I3C Basic***	Supports Async Mode 0	Supports Sync Mode ***Not included in I3C Basic***

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

- **State Byte:** Bit mask indicating which Timing Control Mode (if any) is currently enabled for the Target, and whether any counter overflows have occurred since the most recent previous check (see ***Table 101***). If a Timing Control Mode bit is set (has value 1'b1), then that Target has currently enabled the corresponding Timing Control Mode. If the Overflow bit is set (has value 1'b1), then that Target experienced a counter overflow since the most recent previous check.

Table 101 GETXTIME State Byte Fields

Bit 7	Bit 6	Bit 5	Timing Control Mode Bits				
			Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Overflow	Reserved	Reserved	Async Mode 3 Enabled ***Not included in I3C Basic***	Async Mode 2 Enabled ***Not included in I3C Basic***	Async Mode 1 Enabled ***Not included in I3C Basic***	Async Mode 0 Enabled	Sync Mode Enabled ***Not included in I3C Basic***

- **Frequency Byte:** This byte represents the frequency of the Target's internal oscillator, in increments of 0.5 MHz (i.e., 500 KHz), up to 127.5 MHz. The value 0 is an exception, and indicates an internal oscillator frequency of approximately 32 KHz.

  Example: A value of 8'd20 represents an oscillator frequency of 10.0 MHz.

- **Inaccuracy Byte:** This byte represents the maximum variation of the Target's internal oscillator in $1/10^{th}$ percent (0.1%) increments, up to 25.5%.

  Example: A value of 8'd25 represents a maximum frequency variation of 2.5%.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.7 F006: Multi-Lane

This Section defines I3C Optional Feature F006: Multi-Lane.

Table 102 Optional Feature 006 Status

Optional Feature ID	F006
Optional Feature Name	Multi-Lane
Optional Feature Version	1
Optional Feature specification was last updated in I3C Spec Version	I3C v1.2
Optional Feature is not compatible with other Optional Features	–
Optional Feature is included in I3C Basic	Yes (partial)

Changes in this Version

- For detailed changes in this version, see the "diff" file between I3C v1.1.1 and I3C v1.2 in the MIPI Specification Repository ***[MIPI09]***.

6.7.1 Scope and Purpose (Informative)

The scope of ***Section 6.7*** is limited to fully specifying I3C Optional Feature F006: Multi-Lane. An I3C Device that implements F006 is required to implement all **shall** statements in ***Section 6.7.3***, and can optionally implement any combination of **may** statements in ***Section 6.7.3***. In addition, MIPI Alliance recommends that manufacturers follow all **should** statements in ***Section 6.7.3***.

6.7.2 Technical Overview (Informative)

Although I3C's HDR Modes achieve high data rates using two physical wires and advanced data transfer protocols, some applications would benefit from further increases in data throughput. Multi-Lane Data Transfer (ML) is one approach for achieving such increases, by employing any available additional physical wires to transfer the data payload faster. In ML the normal I3C SDA line is referred to as SDA[0], and the additional wires are called SDA[1], SDA[2], and SDA[3].

ML's advantages include reducing energy consumption and increased flexibility in adding features (e.g., procedures to improve data integrity).

> **Example:** An application with a Camera module using Quad Lanes, an IMU using Dual Lanes, and a simple sensor (such as a temperature sensor) using only I3C SCL and SDA[0] wires, would benefit from ML capability.

ML data transfer is available for data transfers between ML-capable I3C Devices that have ML functionality enabled.

Depending upon the Bus design, a given ML-capable Device could reside on the same 2-wire I3C Bus as other non-ML-capable Devices (and/or other ML-capable Devices not currently in an ML-enabled state). Because ML performs all I3C Bus management functions (i.e., START, EXIT, Enter HDR, CCC, Flow Control, etc.) in the ordinary manner using only the SCL and SDA[0] lines, they remain valid and unchanged for all I3C Devices. As a result, ML transfers on an I3C Bus do not adversely affect any non-ML Devices resident on that Bus.

ML data transfer is available for supported I3C Modes (e.g., HDR-BT), using a per-Mode ML Frame Format that specifies how SCL, SDA (which for ML purposes is referred to as SDA[0]), and the additional data wire(s) are used for ML data transfers in that Mode (see ***Section 6.7.3.3***). During each ML Frame, ML-

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

capable (and ML-enabled) Devices transfer data using SCL, SDA[0], and the additional data wires supported by that ML Frame format, while any non-ML Devices use only SCL and SDA[0] as usual.

For each supported I3C Mode, each ML Frame format shall define the sequence and timing of the structured protocol elements (i.e., Words or Blocks) that are used for control and data in ML transfers:

- For some ML Frame formats, the sequence and timing are similar or identical to I3C's normal (i.e., non-ML) Frame format, adding only the traffic on the additional wires, namely:
  - SDA[1] for DUAL Lane transfers, and
  - SDA[1], SDA[2], and SDA[3] for QUAD Lane transfers.
- For other ML Frame formats, the sequence and timing might differ from the equivalent (i.e., non-ML) Frame format.

During ML data transfers, activity on the additional data wire(s) (i.e., SDA[1], SDA[2], and SDA[3]) would not be visible to any non-ML Devices that might be resident on that Bus. Additionally, any ML-capable Devices that support the same I3C Mode but might not be connected to the same additional data wires, or that have not been enabled and configured to utilize these data wires, might not see or properly understand any ML transfers that use more data wires. For such cases, this might lead to failure of interoperability between Devices supporting the same I3C Mode. The specific nature of interoperability failures might depend on the currently configured Data Transfer Coding of such Devices, as well as the ML Frame format used for a particular ML transfer. To prevent such situations, Controllers shall ensure that all Devices supporting that same I3C Mode are configured to use a compatible (i.e., interoperable) Data Transfer Coding that enables interoperability on the I3C Bus. The recommended Data Transfer Coding(s) to use for such situations requiring interoperability shall be defined for each I3C Mode that supports Multi-Lane.

Note:

Multi-Lane (ML) Frame Formats for other I3C Modes (e.g. SDR, HDR-DDR and HDR-TSP) are not included in I3C Basic. Placeholders for these I3C Modes appear under ***Section 6.7.3.3****.*

Section 6.7.3.3 specifies the Data Transfer Codings for the supported ML Frame formats, for all I3C Modes.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.7.3 Technical Specification

Set-up of an ML data transfer on an ML-capable I3C Bus has two main parts:

1. Configure and enable the I3C Devices capable of ML transfers using the MLANE CCC (***Section 6.7.3.1***)
2. The ML Frame format (***Section 6.7.3.3***)

ML-capable Devices shall default to SINGLE Lane configuration (i.e., basic 2-wire I3C), unless the system designer selects a different default configuration.

6.7.3.1 ML Device Configuration

Controllers shall use the MLANE CCC (see ***Section 6.7.3.6***) to set up and control ML functionality for ML-capable Devices. The MLANE CCC sets or gets, for a given I3C Mode (***Table 108***), the number of Additional Data Lanes (***Table 103***) and the Data Transfer Coding number (see ***Section 6.7.3.3***) to use while communicating to one or more ML-capable Target Devices using that I3C Mode.

Controllers shall configure and enable ML-capable Devices via the MLANE CCC prior to starting ML data transfers, for each I3C Mode for which ML data transfers are to be used. An ML-capable Device may be configured to use any supported number of Additional Data Lanes for any supported I3C Mode.

The Controller shall configure each ML-capable Device with knowledge of its capabilities, and with awareness of the other Devices that support the same I3C Mode(s) and would also be used with ML data transfers, ensuring that such configuration is compatible with the configuration and ML capabilities of such other Devices.

It is the Controller's responsibility to successfully manage the configuration of all Target Devices on the Bus when ML data transfers are used. This depends on capability, connectivity, coordination, and configuration. To build a successful Multi-Lane configuration for all Devices on the I3C Bus, the Controller shall first determine which Target Devices are ML-capable, and then test each such Target Device's connectivity to the Additional Data Lanes (i.e., by using the Direct GET version of the MLANE CCC). For each I3C Mode in which Multi-Lane Transfers will be used, the Controller shall then determine which ML Frame formats are supported by a ML-capable Target Device, and coordinate this with similar data collected from all other ML-capable Target Devices that support that I3C Mode. Finally, once it determines the correct settings that use interoperable Data Transfer Codings, the Controller shall appropriately configure the I3C Bus and all Target Devices, using the MLANE CCC to set the ML Frame Format that shall be used in each I3C Mode.

See Errata 01, Item 33

An ML-capable Target Device shall have only one ML configuration setting (i.e., number of Additional Data Lanes and Data Transfer Coding number) per I3C Mode at a time, which is associated with its Dynamic Address, unless such a Device has multiple Dynamic Addresses per ***Section 6.7.3.1.2***. The Controller shall choose the appropriate Defining Byte as the Sub-Command to configure an ML-capable Target Device for ML transfers in that I3C Mode.

Note:

*The appropriate ML configuration setting (i.e., ML Frame format) for a particular I3C Mode might depend on the other Devices on the I3C Bus that also support that same I3C Mode (per **Section 6.7**). Controllers shall ensure interoperability by choosing an ML configuration setting that is known to be interoperable with other such Devices (i.e., a compatible Data Transfer Coding) that also support that I3C Mode. A recommended procedure for Multi-Lane configuration should include using the MLANE CCC with the Sub-Command for Get ML Capabilities (i.e., Defining Byte 0xFF) to discover the supported capabilities (i.e., the possible ML configurations) of all ML-capable Devices, before using the MLANE CCC with other Sub-Commands to configure any ML-capable Devices for ML transfers for specific I3C Modes.*

*Controllers should use the GETCAPS Format 1 CCC (see **Section 4.3.7.3.19** and **Table 63**) to determine whether a Target Device is ML-capable, before using the MLANE CCC to configure and enable it for ML transfers. Additionally, for ML transfers in any HDR Mode, Controllers should also ensure that the HDR Mode is supported by such a Target Device, by using the GETCAPS Format 1 CCC (see **Table 61**). Controllers must also ensure that each Target Device that advertises ML*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

*transfer capability is indeed connected to the appropriate number of additional data lanes (per **Figure 97**), before configuring such a Target Device to use a ML Frame format requiring such data lanes.*

See Errata 01, Item 34

*Special additional configuration requirements apply for ML-capable Devices that support Group Addressing (see **Section 6.7.3.1.1**), ML-capable Devices that present Virtual Targets with multiple Dynamic Addresses (see **Section 6.7.3.1.2**), and ML-capable Devices that do both of the above.*

Once received by the Target, the ML configuration for the Target's Dynamic Address shall remain in effect until a later MLANE CCC is correctly received, or until a RESET sequence that includes ML configuration is performed by the Controller.

The default Lane configuration for ML-capable Devices shall be SINGLE Lane (3'b000). As a result, the default value of the MLANE Data Byte is 0x00.

Table 103 ML Lane Configurations

Number of Additional Data Wires	Lane Configuration	Description	Multi-Lane?	Data Wires	Total Wires	Wires Supported				
						SCL	SDA[0]	SDA[1]	SDA[2]	SDA[3]
0	SINGLE	Ordinary 2-Wire I3C	N	1	2	✓	✓	–	–	–
1	DUAL	Multi-Lane Data Transfer	Y	2	3	✓	✓	✓	–	–
3	QUAD		Y	4	5	✓	✓	✓	✓	✓
2, 4 through 7	Reserved for future definition by MIPI Alliance									

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.7.3.1.1 ML Devices with Group Addresses

An ML-capable Target Device may also support Group Address capabilities (see ***Section 4.3.4.4***) as indicated by bits[5:4] of the GETCAPS CCC's GETCAP2 byte (see ***Section 4.3.7.3.19*** and ***Table 62***), and a Controller may assign such a Target to a Group Address and then subsequently send transactions to that Group Address or to the Target's Dynamic Address using Multi-Lane formatting. The Controller needs to know the ML capabilities of each Target Device in a Group, and configure them appropriately, such that all transactions sent to that Group Address are properly understood and received by all Target Devices that have been assigned to that Group Address.

Note:

*Per **Section 4.3.4.4**, a Target Device that supports Group Address capabilities shall receive transfers addressed to its Dynamic Address as well as any assigned Group Addresses; all such addresses are in effect simultaneously. If such a Target also supports any HDR Modes, then the assigned Group Addresses are in effect for HDR Write transfers as well as SDR Write transfers, and the Controller may send transfers to the Group Address using any I3C Mode that this Target (and other Targets assigned to the same Group Address) might support.*

Before the Controller assigns a Group Address to an ML-capable Target Device on an I3C Bus that supports Multi-Lane, it also needs to know the ML configuration of the other Target Devices that are assigned to that Group Address, and take steps to ensure that all such Target Devices are configured to use the same number of Additional Data Lanes and the same Data Transfer Coding, when it sends transfers to that Group Address. The Controller also needs to know whether the Target only supports a single ML configuration (i.e., variant 2'b00) that applies to its Dynamic Address as well as any currently-assigned Group Addresses, or whether the Target supports separate ML configurations (i.e., a different number of Additional Data Lanes or an alternate Data Transfer Coding) for each assigned Group Address (i.e., variant 2'b01).

The decision to support multiple ML configurations within a Target Device (i.e., the variant that is reported by the MLANE Direct GET CCC with the Sub-Command for Group ML Capabilities) is left up to the implementer; however, support for separate ML configurations is strongly recommended for ML-capable Target Devices that will be used in mixed-lane Group configurations (i.e., for I3C Buses where Target Devices assigned to a Group Address do not all support the same number of Additional Data Lanes).

Note:

*Per **Section 6.7.3.1.2**, a composite Device that has shared Peripheral logic and supports a separate Dynamic Address for each Virtual Target that is presented on the Bus shall support a separate ML configuration for each Dynamic Address. If such a composite Device also supports Group Address capabilities, then it shall also support separate ML configurations for each currently assigned Group Address (i.e., variant 2'b01) per this section, such that each Dynamic Address may be assigned to Group Addresses in a similar manner (i.e., act according to the requirements for variant 2'b01 as detailed below).*

If an I3C Bus contains ML-capable Targets of both variants (i.e., variant 2'b00 and variant 2'b01), then the Controller shall not assign the same Group Address to Target Devices of both variants, due to the increased complexity of configuring Multi-Lane transfers for such a Group Address, as well as the possibility of misconfiguration among such Target Devices.

Variant 2'b01

If an ML-capable Target that supports Group Address capabilities also supports separate ML configurations (i.e., variant 2'b01), then it shall maintain a configuration for each currently assigned Group Address, once assigned by the Controller using the SETGRPA CCC (see ***Section 4.3.7.3.24***). A Target's initial ML configuration for a newly-assigned Group Address shall be set to the common ML configuration (i.e., ML Frame format) for the I3C Mode and the configuration of the Device, per ***Section 6.7.3.1***:

- For HDR-BT Mode, the initial common ML configuration shall be the same as the default ML configuration (as above). However, the common ML configuration may subsequently be changed, if the Controller configures the Device to use an ML Frame format that is based on any alternate Data Transfer Coding (per ***Section 6.7.3.5.3***).

Copyright © 2016–2025 MIPI Alliance, Inc.

- For Group Address assignments that were present before configuring such a Device to use an alternate Data Transfer Coding, the Device shall change the ML configuration to use the new common ML Frame format for this alternate Data Transfer Coding. In some cases, this might result in an ML configuration for an assigned Group Address changing to reflect the common ML Frame format, which may include a different number of additional data lanes than what was originally configured for that Group Address.
- For Group Address assignments that were configured with the SETGRPA CCC after such a Device was configured to use an alternate Data Transfer Coding, the Device shall use the new common ML configuration for this alternate Data Transfer Coding.
- In both cases, the data returned in response to the MLANE Direct GET CCC with the Sub-Command for Get Group Report (i.e., Defining Byte 0x9C) shall always reflect the correct status of each Group Address assignment.

- Additional requirements may apply for ML-capable Devices that also present multiple Virtual Targets (per ***Section 6.7.3.1.2***).

The Controller may subsequently configure all such Targets that support separate ML configurations and are assigned to this Group Address to use a different ML Frame format for WRITE transactions directed to the Group, by sending the MLANE Direct SET CCC to the Group Address. The Controller shall only use the Group Address with the MLANE CCC if and only if all such Targets (i.e., members assigned to that Group Address) support separate ML configurations, and only if all such Targets support the indicated ML Frame format. It is the responsibility of the Controller to make this determination before assigning ML-capable Targets to a Group Address, and before sending the MLANE Direct SET CCC to a Group Address with a particular Data Byte value with the intent to configure the Group (i.e., to set the ML Frame format for all Targets that are members assigned to that Group Address).

The Controller should determine whether an ML-capable Target does support separate ML configurations by using the MLANE Direct GET CCC (see ***Section 6.7.3.6***) with the Sub-Command for Group ML Capabilities (i.e., Defining Byte 0x9B). A Target that supports separate ML configurations shall indicate this support (see ***Table 108***) as well as the number of Group Addresses to which it is currently assigned.

To verify that an ML configuration change sent to the Group Address was successfully applied, the Controller should use the MLANE Direct GET CCC with the Sub-Command for Get Group ML Report (i.e., Defining Byte 0x9C) for each Target assigned to that Group (i.e., by sending the MLANE Direct GET CCC with Defining Byte 0x9C to each Target's Dynamic Address) to read its current ML configurations for all currently assigned Group Addresses.

The Controller may also reset all current ML configurations for a Target (including any configured Group Address configurations) by sending the MLANE CCC with the Sub-Command for Reset ML (i.e., Defining Byte 0x7F). This shall reset all such Targets assigned to the Group Address to a known common ML Frame format.

If a Target supports multiple current ML configurations for all its assigned Group Addresses, then each ML configuration may be set to any of the supported ML Frame formats, as returned by the MLANE Direct GET CCC with the Sub-Command for Get ML Capabilities (i.e., Defining Byte 0xFF), as long as they are all configured to be mutually interoperable within the same I3C Mode.

Note:

Special restrictions apply for certain ML Frame formats that are not interoperable with the current ML Frame format that is configured for the Dynamic Address to which the Group Address is assigned. Such restrictions apply for I3C Modes (such as HDR-BT) that define alternate Data Transfer Codings that are not interoperable with the default Data Transfer Coding. In no case shall a Device allow any assigned Group Address to be configured to use an ML Frame format that is based on a Data Transfer Coding that is not interoperable with the Data Transfer Coding of the Dynamic Address's currently configured ML Frame format.

The Controller may use the MLANE CCC to configure the current ML Frame format for transfers addressed to the Group Address (i.e., by sending the MLANE Direct SET CCC to the Group Address) to set this

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

configuration, which will not change the current ML configuration for the Target's Dynamic Address of any such Target within that Group. Such a Target will keep track of its multiple current configurations for its assigned Addresses, and will use the Address received for each transaction to match and select the correct ML configuration based on the Address.

Note:

If such an ML-capable Target supports separate ML configurations for each assigned address (i.e., multiple ML configurations for its assigned Group Addresses), then it must do so for each I3C Mode in which it supports Multi-Lane transfers. An ML-capable Target shall not implement partial support for separate ML configurations for only certain I3C Modes (i.e., variant 2'b01) and use a single ML configuration (i.e., variant 2'b00) for other I3C Modes. The variant is a global option that applies to the Target Device.

*Per **Section 6.7.3.6**, such an ML-capable Target shall also support the MLANE CCC as a Broadcast SET CCC in addition to a Direct SET CCC addressed to an assigned Group Address, for supported I3C Modes. In such a case, the MLANE Broadcast SET CCC shall change the ML configuration for all currently assigned Group Addresses as well as the assigned Dynamic Address, and may also set (or reset) the common ML configuration for newly-assigned Group Addresses, if the I3C Mode allows that to be changed (i.e., for alternate Data Transfer Codings, as noted above).*

Variant 2'b00

However, if an ML-capable Target that supports Group Address capabilities does not support separate ML configurations (i.e., variant 2'b00), then the ML configuration for the Target's Dynamic Address shall also be applied to all Group Addresses to which the Target is currently assigned. For such Targets, the Controller might need to re-configure other Targets in all Group Addresses to which this Target is assigned, such that all Targets use the same ML Frame format. Alternately, the Controller might need to frequently reconfigure the ML Frame format for this Target, when switching from directed transactions using the Target's Dynamic Address to those using an assigned Group Address (or vice versa). It is the Controller's responsibility to determine whether this needs to happen (and how often), which will depend on the ML capabilities of any other Targets that are currently assigned to the same Group Address (or any other assigned Group Addresses, if applicable).

In some cases, ML configurations using Additional Data Lanes simply might not be possible for ML-capable Targets that only support a single ML configuration (i.e., variant 2'b00), depending on the ML capabilities and current ML configuration of other Targets that might be assigned to the same Group Address.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.7.3.1.2 ML Devices with Multiple Dynamic Addresses

If an ML-capable Target Device is a Virtual Target with its BCR bit[4] set to 1 (see ***Table 7***), and also shares its Peripheral logic with other Virtual Targets within the same composite Device (see ***Section 4.3.2.1.2*** and ***Section 4.3.7.3.23***, i.e. the RSTACT CCC with Defining Byte 0x04 in ***Table 38***), then the Device shall support separate ML configurations for each Dynamic Address that such a Device presents to the I3C Bus.

Note:

*Before assigning separate ML configurations to each Virtual Target, the Controller should use the GETCAPS Format 2 CCC with Defining Byte VTCAPS (see **Section 4.3.7.3.19**) to determine which Devices present Virtual Targets using shared Peripheral logic; and also use the RSTACT CCC with Defining Bytes 0x04 and 0x84 (see **Section 4.3.7.3.23**) to correctly identify which Dynamic Addresses are associated together within the same composite Device having shared Peripheral logic. With this knowledge, the Controller can prepare for any interactions that might result from sending the MLANE Direct SET CCC to one Dynamic Address, if such interactions or side effects are defined for a particular I3C Mode.*

For such composite Devices, using the MLANE Direct SET CCC to read or change the current ML configuration for one Virtual Target's Dynamic Address shall only act on that Virtual Target, and shall not affect any other Virtual Targets within the same Device, unless the Controller sends a MLANE Direct SET CCC for a particular I3C Mode to configure one Dynamic Address to use an Data Transfer Coding that is not interoperable with the default Data Transfer Coding or current Data Transfer Coding of the ML-capable Device. Such exceptions are specifically defined for such alternate Data Transfer Codings in sub-sections of ***Section 6.7.3.3***, for the I3C Modes with interoperability advisories. Note that such alternate Data Transfer Codings also carry special requirements that apply to the entire ML-capable Device with respect to its support for ML transfers in that I3C Mode.

In all other cases, the Controller may send an MLANE Direct SET CCC to configure one Dynamic Address with a valid ML configuration, and the Device shall apply that valid ML configuration to only the Virtual Target using that assigned Dynamic Address (i.e., with no effect on other Virtual Targets using other assigned Dynamic Addresses).

Note:

*An ML-capable Target Device that presents multiple Virtual Targets might also support Group Addressing (see **Section 6.7.3.1.1**) in addition to multiple Dynamic Addresses. If so, then the conditions in that Section shall also apply, and each assigned Dynamic Address may be assigned to any available Group Address.*

*Per **Section 6.7.3.6**, such an ML-capable Target shall also support the MLANE CCC as a Broadcast SET CCC in addition to a Direct SET CCC addressed to an assigned Group Address, for supported I3C Modes. In such a case, this shall attempt to change the ML configuration for all currently assigned Group Addresses as well as the assigned Dynamic Address, and may also set (or reset) the common ML configuration for newly-assigned Group Addresses. For HDR Modes, this may also change the common Data Transfer Coding that is used for ML transfers that are addressed to the Broadcast Address (e.g., for CCC flows in HDR Modes).*

All Virtual Targets within the same Device should return the same output when using the MLANE Direct GET CCC with Defining Byte 0xFF (see ***Table 108***) to read the supported ML Frame formats. In cases where a Device has been set to use an alternate Data Transfer Coding (i.e., one that is not interoperable with the default Data Transfer Coding, as above) then the Device may filter the output of the MLANE Direct GET CCC, and only return the possible ML Frame formats that may be chosen within the currently configured Data Transfer Coding.

If the Controller uses the MLANE Direct SET CCC with Defining Byte 0x7F, then the Sub-Command for Reset ML shall only apply to the addressed Virtual Target (i.e., the Dynamic Address to which it was sent), and shall not affect any other Virtual Targets within the same Device. In all cases, the Sub-Command for Reset ML as a Direct SET CCC shall not cause the Device to cause (or attempt to apply) an inconsistent set of ML configurations that might have a mix of Data Transfer Codings for different Virtual Targets that are not interoperable; for such special cases, the defined behavior of a Sub-Command for Reset ML as a Direct

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

SET CCC shall be specifically defined in the appropriate sub-sections of ***Section 6.7.3.3*** that might define alternate Data Transfer Codings for a particular I3C Mode (as above). If not so defined, then the Sub-Command for Reset ML as a Direct SET CCC shall have the effect of returning one Virtual Target's Dynamic Address to the default Data Transfer Coding with the default Lane Configuration.

If the Controller uses the MLANE Broadcast CCC with the Sub-Command for Reset ML (i.e., Defining Byte 0x7F), then this action shall apply to all Virtual Targets (i.e., all assigned Dynamic Addresses within the Device) and return them all to the default Data Transfer Coding with the default Lane Configuration.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.7.3.2 The ML Frame

Before starting any ML data transfers, the Controller shall configure the resident ML-capable Devices using the appropriate version of the MLANE CCC (see ***Section 6.7.3.6***). Controllers should use MLANE's Direct GET Sub-Command for the relevant I3C Modes, to check that the current ML configuration of each ML-capable Device is correct for all I3C Modes in which Multi-Lane transfers will be used. A Device's ML configuration parameters (i.e., Data Transfer Coding number and number of Additional Data Lanes) per I3C Mode remain in effect until a new MLANE SET CCC for that I3C Mode is correctly received, or until the Controller uses the MLANE CCC with the Sub-Command for Reset ML (i.e., Defining Byte 0x7F) to reset a single Device using its Dynamic Address (i.e., via Direct CCC) or to reset all Devices on the Bus (i.e., via Broadcast CCC).

Since all I3C Bus management is done using only the SCL and SDA[0] lines, an I3C ML Frame starts as specified by the ordinary framing rules for the selected I3C Mode. That is, the START, Address Arbitration, and entry into an HDR Mode (if so chosen) are all done in the same manner as for I3C SINGLE Lane configuration. During the ML data transfer, and within the Frame, ML-capable (and ML-enabled) Devices shall automatically communicate using the configured number of additional data Lanes and according to the configured Data Transfer Coding. EXIT from the I3C Frame follows the ordinary I3C rules specified for the selected I3C Mode.

For Multi-Lane transfers using any HDR Mode, the following common structured protocol elements are defined for ML transfers while in that HDR Mode:

- **HDR-x Header Element**, which typically indicates the Dynamic Address (or Group Address, if supported) to which the transfer is directed
  - In HDR-BT Mode, this is the HDR-BT Header Block (see ***Section 6.4.3.3.1***).
- **HDR-x Data Element**, which typically contains data bytes using Additional Data Lane(s)
  - In HDR-BT Mode, this is the HDR-BT Data Block (see ***Section 6.4.3.3.2***).
- **HDR-x Termination Element**, which indicates the end of the transfer
  - In HDR-BT Mode, this is the HDR-BT CRC Block (see ***Section 6.4.3.3.3***).

Figure 154 illustrates a typical ML Frame in a generic HDR Mode, shown as "HDR-x". Although the example shows a period with no Address Arbitration, the same basic principles also apply for Direct addressing and Address Arbitration.

1. **START of Frame, I3C Reserved Byte with RnW = 1'b0, Enter HDRx**

   This section takes place traditionally, using SDR Mode over SCL and SDA[0]

2. **HDR-x Header Element**

   This structured protocol element is sent by the Controller, and must be received and understood by all Devices that support this HDR Mode. As such, the Controller shall ensure that any data sent on the additional data wire(s) is either "optional" in nature, for Devices that are either not ML-capable or not configured to listen to such wires; or that all such Devices are ML-capable, and configured for a Data Transfer Coding that allows them to receive and understand the data sent on such wires.

3. **HDR-x Data Element(s) comprising a data Transfer**

   This structured protocol element is sent using the ML-enabled Lanes and the chosen Data Transfer Coding for this HDR Mode. It might be sent once or in a repeated sequence, per the HDR Mode.

   - For a Write transfer, the Controller sends these structured protocol elements, and the addressed Device (or all Devices that are members of the addressed Group) indicated by the HDR-x Header Element receives them, having been configured to understand the Data Transfer Coding and the same number of additional data wires.
   - For a Read transfer, the addressed Device sends these structured protocol elements, and the Controller receives them according to the known Data Transfer Coding for the Device and this specific transfer.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

4. HDR-x Termination Element

This structured protocol element is optionally sent per the rules of the HDR Mode, typically by the same Device that also sends the HDR-x Data Element(s) as part of the data transfer.

Additional specific activity per HDR Mode (such as Bus Turnaround) might follow; this is not shown in this example.

5. HDR Restart Pattern

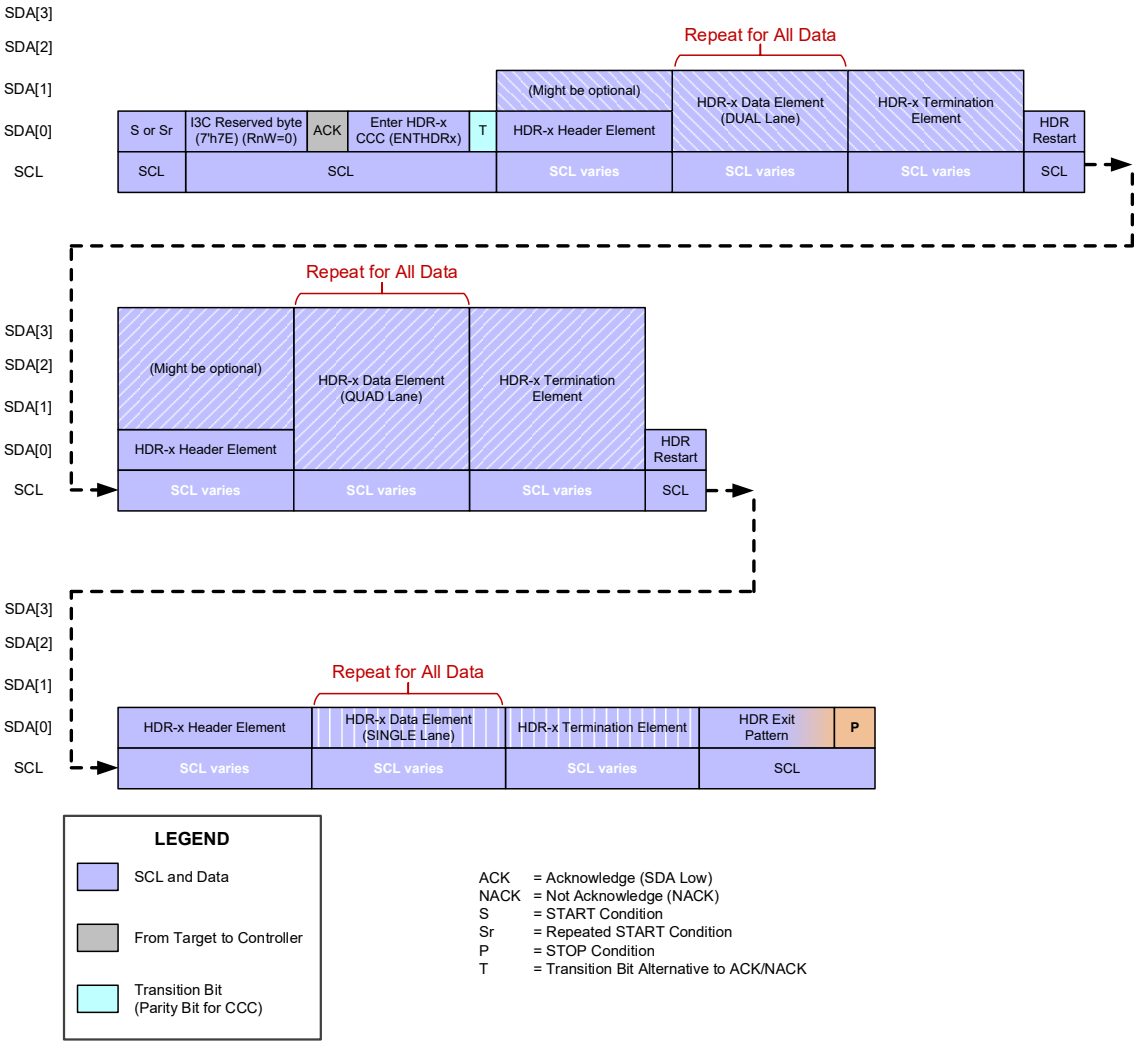
Restarting the ML frame uses the usual HDR Restart Pattern on SCL and SDA[0], and then all Devices supporting that HDR Mode start monitoring to receive the next the HDR-x Header Element (jump to step 2 above).

6. End of ML Frame

Ending the ML frame uses the usual HDR Exit Pattern on SCL and SDA[0], and then all Devices start monitoring the Bus Lanes as the Bus returns to SDR Mode.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition



7398

Figure 154 Typical Generic ML I3C Frame (HDR)

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

The ***Figure 154*** example shows three sample ML transfers: one for QUAD-lane, one for DUAL-lane, and one for SINGLE-lane. This ML framing model represents how any HDR Mode might support ML transfers for any Lane Configuration (see ***Table 103***) using the same basic ML framing model. This model enables generic HDR read/write transfers using some or all additional data wires, as well as CCC flows (i.e., Broadcast CCCs and Direct CCCs) per ***Section 6.1.3.1***.

For each HDR Mode that supports Multi-Lane, this Specification shall define Data Transfer Codings for that HDR Mode, such that the specific HDR-x Header Elements sent by the Controller shall be understood and received by all Devices. However, on an I3C Bus with multiple Devices supporting that HDR Mode, the Controller shall have the responsibility of ensuring that all such Devices are capable of receiving (and configured to understand) the HDR-x Header Elements. In some cases, multiple such Devices might use different Lane Configurations. This might require the use of a compatible Data Transfer Coding, specific to each HDR Mode.

- For the ***Figure 154*** example, the use of multiple transfers with different Lane Configurations would require a Coding that is compatible with all Devices on the Bus. Any data sent on the additional data wire(s) for the “HDR-x Data Element” and optional “HDR-x Termination Element” must only be “optional” in nature, for such Devices that are either not ML-capable or not configured to listen to all such wires. For this reason, the HDR-x Header Elements only show SCL and SDA[0] with required data, whereas SDA[1] through SDA[3] are shown as “Might be optional”.
- For other cases where all such Devices are configured to use some (or all) of the additional data wires, the Controller might use an “alternate mode” Data Transfer Coding that uses additional data wires for the HDR-x Header Elements, since it knows that all such Devices can receive and understand the HDR-x Header Element in order to know when they are being addressed. If an HDR Mode defines such “alternate mode” Data Transfer Codings for these use cases, then their use and configuration shall be clearly specified, along with guidelines for interoperability.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.7.3.3 ML Frame Formats

Table 104 indicates which Section of this specification describes the ML Frame format used for each I3C Mode that supports Multi-Lane. For each supported I3C Mode, the 1-Lane (Default) Mode is also detailed in the referenced section.

Table 104 All ML Frame Formats

I3C Mode	ML Data Transfer	MLANE CCC Fields		Coding	Additional Data Lanes	Section	For More Details See
		Defining Byte	Data Byte				
HDR-BT	HDR-BT **DUAL**	0x23	0x01	0	1	***6.7.3.5.3.2***	***Table 105***
		0x23	0x19	3	1		
	HDR-BT **QUAD**	0x23	0x03	0	3	***6.7.3.5.3.3***	
		0x23	0x1B	3	3		
		0x23	0x3B	7	3		

Note:

In addition to the above, the full I3C Specification includes definitions for more ML Frame formats:

SDR Based ML Frame Formats:

SDR-ML DUAL Coding

SDR-ML QUAD Coding

HDR-DDR Based ML Frame Formats:

HDR-DDR-ML DUAL Coding

HDR-DDR-ML QUAD Coding

HDR-TSP Based ML Frame Formats:

HDR-TSP-ML DUAL Coding

HDR-TSP-ML QUAD Coding

To gain access to these capabilities, please contact MIPI Alliance.

6.7.3.4 (Unused Heading)

Note:

This heading number is not used in the I3C Basic Specification. It has been retained to preserve section number synchronization with the full I3C Specification.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.7.3.5 HDR Mode for Multi-Lane

For I3C Buses that support Multi-Lane (ML) functionality (see ***Section 6.7***), the Controller may configure any ML-compatible Target Devices to use additional data Lanes to increase data transfer rates, in a supported HDR Mode that supports multiple ML Frame formats using these additional data Lanes.

Note:

For I3C Basic, HDR-BT is the only HDR Mode that supports Multi-Lane (ML) functionality.

However, if some Target Devices that support this HDR Mode do not support the same ML Frame formats (i.e., if there is varying support for the number of additional data Lanes, or for alternate Data Transfer Codings with different ML Frame formatting), then compatibility issues can arise as these Target Devices would not be able to correctly receive and understand all the common CCC Flow Elements the Controller transmits. The same issues can arise if any of the Target Devices are configured to use ML Frame formats that are not mutually interoperable with the rest of the Target Devices, even if all Target Devices support such ML Frame formats.

If a Controller chooses to use any ML Frame formats that use additional data Lanes and change the formatting of common CCC Flow Elements, then the Controller must ensure that all Target Devices can support (and are configured to use) mutually interoperable ML Frame formats before attempting to drive any CCC flows using such an ML Frame format.

In general, all supported HDR Modes include a 1-Lane compatibility mode that is supported by all Multi-Lane capable Target Devices in their default configuration for that HDR Mode, as well as all Target Devices that only connect to one Lane. To ensure maximum compatibility, a Controller should generally use 1-Lane compatible ML Frame formats for the common CCC Flow Elements, so that all Target Devices can properly receive and understand the CCC flows. This includes all HDR-x CCC Header Blocks, and any HDR-x CCC Data Blocks that are part of Broadcast CCC flows.

For such a Bus, if a Controller has previously configured some Target Devices to use an ML Frame format that uses additional data Lanes or different formatting, then the Controller must use a “compatibility mode” ML Frame format that uses the 1-Lane compatible Frame formatting to transmit all common CCC Flow Elements, and such Target Devices shall use the 1-Lane compatible Frame formatting to receive the common CCC Flow Elements. However, different formatting that uses additional data Lanes shall be used for per-Target transactions involving such Target Devices (if previously configured to do so) for the HDR-x CCC Data Blocks within the Read Segments or Write Segments of Direct CCC flows.

Alternately, a Controller may configure all Target Devices to use an “alternate mode” Coding that uses additional data Lanes or different formatting, in order to increase the performance or efficiency of HDR CCC flows. Such an “alternate mode” Coding might intentionally be incompatible with the “compatibility mode” Coding for this HDR Mode, but it should not be used unless all Target Devices support it. The Controller must ensure that all Target Devices are configured to use a mutually interoperable “alternate mode” Coding, and the rules for the “alternate mode” Coding shall apply.

The specifics of Multi-Lane support for each HDR Mode are listed in the Framing details (see ***Table 77***). Each Section describing a specific HDR Mode lists the Flow Elements that may utilize Multi-Lane Frame formatting, while maintaining compatibility with all other Target Devices that might not be configured to use ML Frame formats. The ML Frame formats that are mutually interoperable are also defined in this specification.

Since the Controller has the responsibility for configuring Multi-Lane for the Bus and all compatible Target Devices as well as initiating the CCC flow, it shall conform to these requirements, and shall ensure that all configuration is successfully completed. Likewise, the Controller must ensure that any core CCC Flow Elements that must be received and understood by all Target Devices supporting that HDR Mode are sent in an appropriate ML Frame format, as defined in the ML Frame formats that have been configured for all Target Devices, and that all Target Devices are configured to be mutually interoperable with respect to these core CCC Flow Elements. Likewise, all Target Devices shall also correctly receive and understand the Indicator and Selector blocks of the CCC flow in that ML Frame format, even if they have been otherwise configured

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

to use an ML Frame format that uses additional Lanes or a different ML Frame format for standard (i.e., non-CCC) HDR transactions.

Use of the MLANE CCC within an HDR Mode should be managed carefully, since the Direct SET forms of this CCC provide the unique ability to change a Target Device's ML Frame format, and that directly impacts a Target's ability to receive and understand an HDR-x CCC Data Block sent by a Controller, and also impacts a Target's ability to respond on the data Lanes in a format that the Controller expects it to utilize for a given ML Frame format. As such, any misunderstanding or mismatch of expectations between a Controller and a Target may result in communication errors, which would require the Controller to catch any resulting errors and initiate a recovery procedure to restore communications.

In some situations, this may require temporarily resetting a Target's ML Frame format using the MLANE CCC (see ***Section 6.7.3.6***) with Defining Byte value 0x7F (see ***Table 108***). If the Target fails to respond to this CCC with Defining Byte value 0x7F while in HDR Mode, then the Controller should exit HDR Mode.

To avoid these issues, it is strongly recommended to not use the Direct SET forms of this CCC within any HDR Mode, unless the Controller is able to ensure that all possible communication errors can be reliably detected and resolved.

6.7.3.5.1 (Unused Heading)

Note:

This heading number is not used in the I3C Basic Specification. It has been retained to preserve section number synchronization with the full I3C Specification.

6.7.3.5.2 (Unused Heading)

Note:

This heading number is not used in the I3C Basic Specification. It has been retained to preserve section number synchronization with the full I3C Specification.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.7.3.5.3 HDR-BT Based ML Frame Formats

For all HDR-BT Based ML Frame formats, the MLANE Defining Byte value shall be 0x23 (i.e., the same numerical value as the ENTHDR3 CCC).

Within Defining Byte 0x23 there could be several Data Transfer Coding schemes specifying the data payload format and optional additional information being transferred. Additionally, several "alternate mode" Data Transfer Codings have been defined, which allow for enhanced performance by using additional data Lanes for transferring HDR-BT Header Blocks, as well as the common CCC flow elements in the HDR-BT CCC flows (see ***Section 6.7.3.5.3.1*** and ***Table 105***).

The default Data Transfer Coding is HDR-BT Coding 0 (compatible mode). Other Data Transfer Codings could be added, as suitable for specific applications or performance requirements.

The currently defined ML Frame formats for HDR-BT Based ML Frames are shown in ***Table 105***.

Table 105 HDR-BT Frame Formats in Multi-Lane

I3C Mode	ML Data Transfer	MLANE CCC Fields		Coding	Additional Data Lanes	Formatting Notes	Section
		Defining Byte	Data Byte				
HDR-BT	HDR-BT (1-Lane)	0x23	0x00	0	0	All HDR-BT Blocks use 1-Lane formatting (Compatible Mode) *Note:* *This is the initial default ML configuration for all HDR-BT Target Devices.*	***6.7.3.5.3.1***
	HDR-BT **DUAL**	0x23	0x01	0	1	Header Blocks and common CCC flow elements use 1-Lane formatting (Compatible Mode)	***6.7.3.5.3.2***
		0x23	0x19	3	1	Header Blocks and common CCC flow elements use 2-Lane formatting ("Alternate Mode")	
	HDR-BT **QUAD**	0x23	0x03	0	3	Header Blocks and common CCC flow elements use 1-Lane formatting (Compatible Mode)	***6.7.3.5.3.3***
		0x23	0x1B	3	3	Header Blocks and common CCC flow elements use 2-Lane formatting ("Alternate Mode")	
		0x23	0x3B	7	3	Header Blocks and common CCC flow elements use 4-Lane formatting ("Alternate Mode")	

Note:

The "Alternate Mode" Codings for HDR-BT Mode shall not be used unless all HDR-BT Target Devices on the Bus are ML-capable and have been configured to use interoperable Codings.

*Configuring an HDR-BT capable Target Device to use an "Alternate Mode" Coding shall change how the Device interprets the HDR-BT Header Block, which affects its ability to match an assigned Address for subsequent HDR-BT transfers. This includes Devices that support Group Addressing per **Section 6.7.3.1.1**, as well as Devices that support multiple Dynamic Addresses as Virtual Targets per **Section 6.7.3.1.2**.*

Selecting an "Alternate Mode" Coding requires the Controller to use a different formatting for all HDR-BT Header Blocks, as well as all common CCC flow elements in the HDR-BT CCC flows. If any

Copyright © 2016–2025 MIPI Alliance, Inc.

*HDR-BT Target Device is unable to understand that formatting due to an incompatible configuration, or due to being non-ML-capable, then it will not properly receive and understand many of the HDR-BT Blocks, and this might cause communication errors. See **Section 6.7.3.5.3.1** for more details on which HDR-BT Multi-Lane Frame formats are interoperable for "Alternate Mode".*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Figure 155 illustrates a common ML Frame for HDR-BT, using Coding 0. This Coding supports HDR-BT Target Devices that use any number of additional data Lanes (including no additional data Lanes).

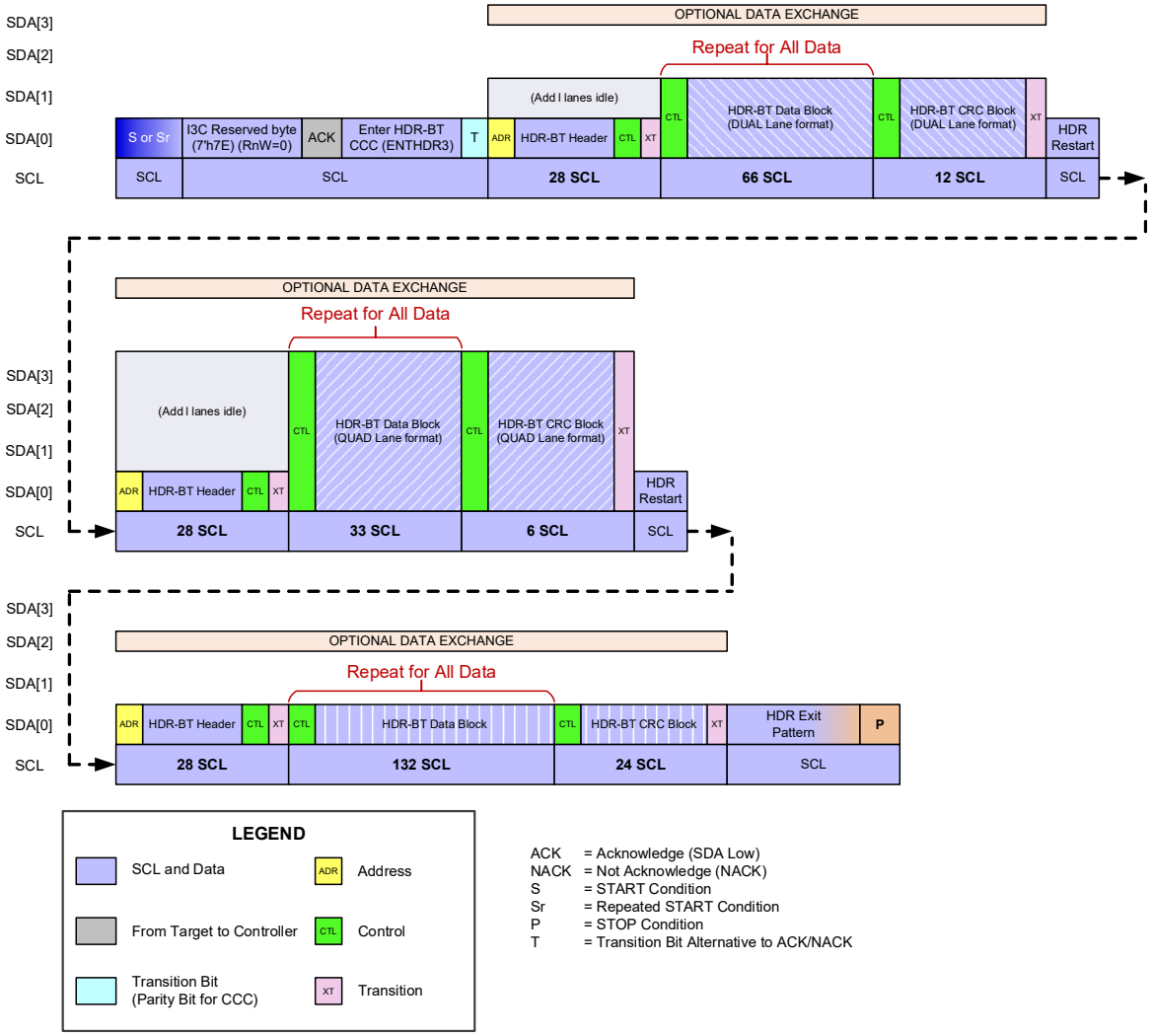
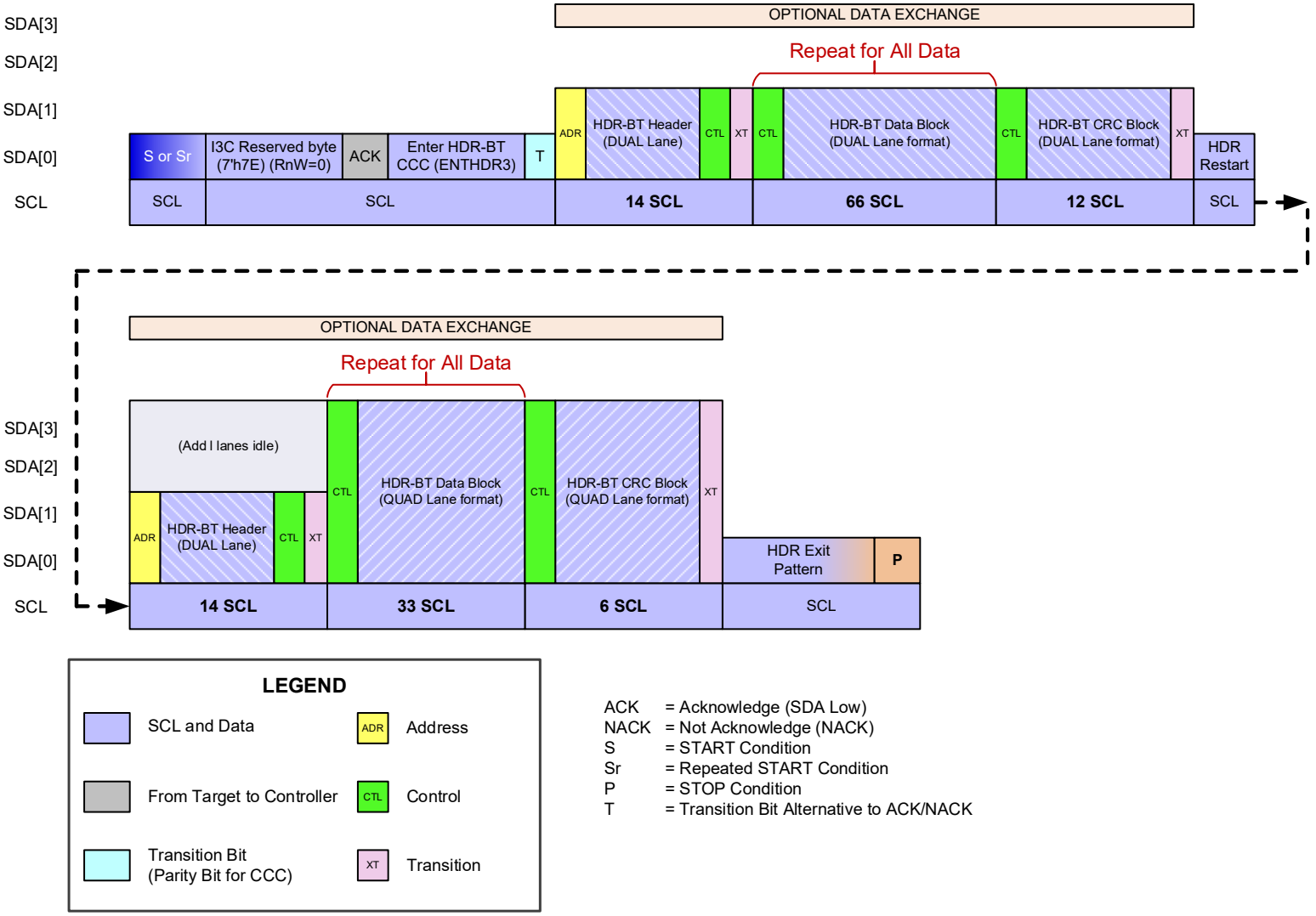


Figure 155 Typical ML I3C Frame Based on HDR-BT Mode (Coding 0)

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

7540 ***Figure 156*** illustrates a common ML Frame for HDR-BT, using Coding 3. When using this Coding, no 1-Lane HDR-BT Target Devices are supported.

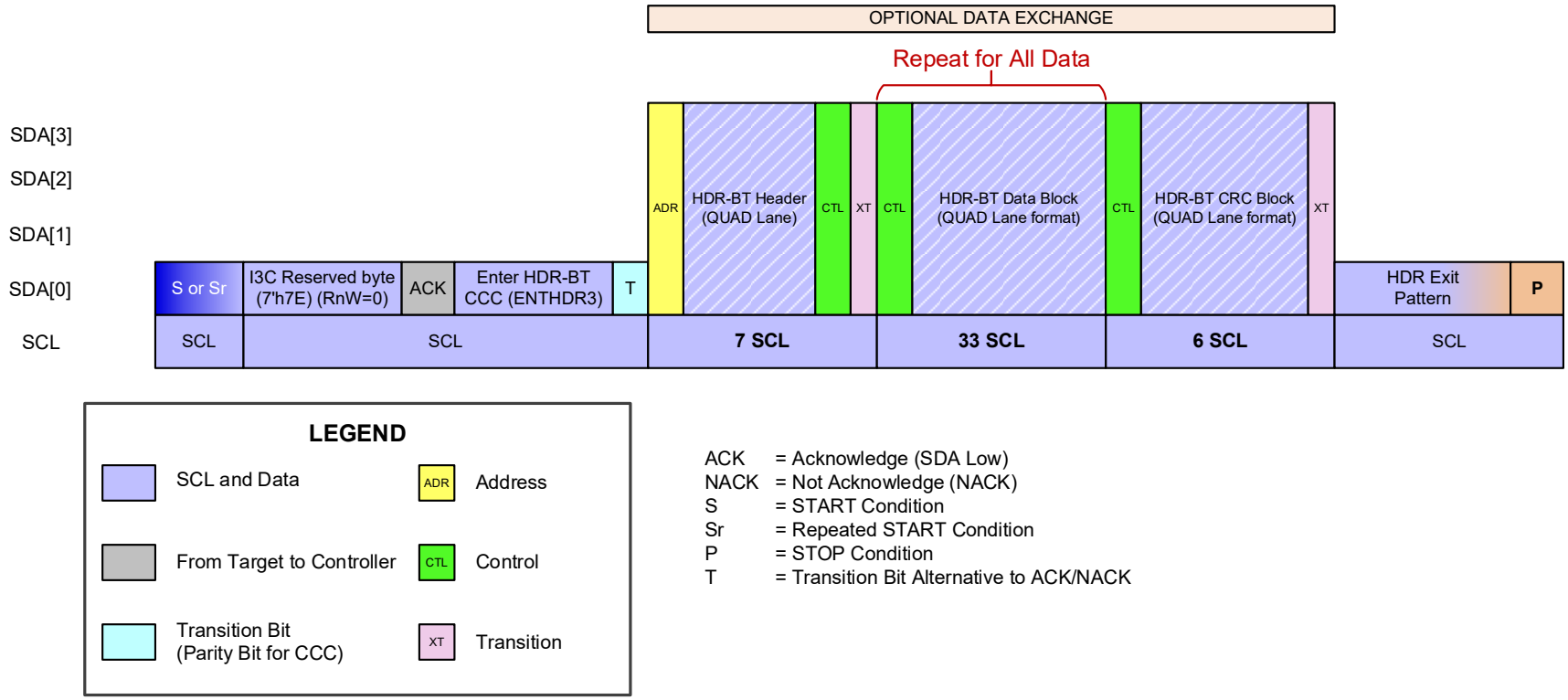


7541

Figure 156 Typical ML I3C Frame Based on HDR-BT Mode (Coding 3)

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

7542 ***Figure 157*** illustrates a common ML Frame for HDR-BT, using Coding 7. When using this Coding, no 1-Lane or 2-Lane ML HDR-BT Target Devices are
7543 supported.



7544

Figure 157 Typical ML I3C Frame Based on HDR-BT Mode (Coding 7)

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.7.3.5.3.1 HDR-BT Codings and Interoperability for CCCs

HDR-BT defines several “Alternate Mode” ML Frame formats that may optionally be used for increased performance efficiency. However, not all of these are mutually interoperable. ***Table 106*** lists the supported Codings and defines how they affect the formatting of various CCC flow elements that comprise CCC flows in HDR-BT Mode.

Table 106 HDR-BT Frame Format Details and Coding Interoperability for CCCs in Multi-Lane

Coding and Meaning	HDR-BT Header Block Format	HDR-BT Data Blocks and CRC Block Format	Common ML Frame Format (i.e., Default Data Byte)	Common CCC Flow Elements	Per-Target CCC Flow Elements	Notes
0 (Compatible Mode)	SINGLE Lane (see ***Figure 137***)	1-Lane: ***Table 87*** and ***Table 90*** 2-Lane ML: ***Table 88*** and ***Table 91*** 4-Lane ML: ***Table 89*** and ***Table 92***	0x00 (SINGLE Lane)	Use 1-Lane HDR-BT Frame formatting	May use 1-Lane, 2-Lane ML, or 4-Lane ML (as configured per Target’s ML Frame format)	The only Coding that is compatible with 1-Lane (non-ML) HDR-BT Targets
3 (“Alternate Mode”)	DUAL Lane (see ***Figure 138***)	1-Lane: Not Supported 2-Lane ML: ***Table 88*** and ***Table 91*** 4-Lane ML: ***Table 89*** and ***Table 92***	0x19 (DUAL Lane)	Use 2-Lane ML HDR-BT Frame formatting	May use 2-Lane ML or 4-Lane ML (as configured per Target’s ML Frame format)	Requires all HDR-BT Targets to support at least 1 additional data Lane
7 (“Alternate Mode”)	QUAD Lane (see ***Figure 138***)	1-Lane: Not Supported 2-Lane ML: Not Supported 4-Lane ML: ***Table 89*** and ***Table 92***	0x3B (QUAD Lane)	Use 4-Lane ML HDR-BT Frame formatting	Use 4-Lane ML only	Requires all HDR-BT Targets to support all 3 additional data Lanes

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

All supported ML Frame formats shown in ***Table 105*** shall indicate which of these Data Transfer Codings they support, which in a mixed ML-configuration Bus determines their level of interoperability with other ML Frame formats.

As noted in ***Section 6.7.3.5.3***, configuring the I3C Bus (i.e., all HDR-BT capable Targets) to use an "Alternate Mode" Data Transfer Coding requires all such Targets to be configured to use the "Alternate Mode" Coding. This configuration may be done individually for each Target, by sending the MLANE Direct SET CCC with Defining Byte 0x23 for each HDR-BT capable Target; or it may be done simultaneously for all Targets, by sending the MLANE Broadcast CCC with Defining Byte 0x23.

However, the Controller shall not attempt to configure any ML-capable HDR-BT Targets to use any "Alternate Mode" Coding, unless it first ensures that all other HDR-BT Targets are capable of interoperating with this Coding and may be configured to use a mutually interoperable ML Frame format that is based on the same Coding.

When a Target that supports any HDR-BT "Alternate Mode" Codings first receives either an MLANE Direct SET CCC with Defining Byte 0x23 sent to an assigned Dynamic Address, or an MLANE Broadcast CCC with Defining Byte 0x23 that is sent to the I3C Bus; and either CCC has a message with a Data Byte that selects an ML Frame format based on such an "Alternate Mode" Coding which is not interoperable with Coding 0, this shall change the HDR-BT configuration for the Device, and affect how the Device and its Peripheral logic must interpret HDR-BT Header Blocks for all subsequent HDR-BT transfers. For HDR-BT Mode, this configuration is defined as a 'sticky' state, and shall also determine how subsequent MLANE CCCs are either accepted or rejected.

- Such a change shall become 'sticky' if accepted, and shall affect the Peripheral logic that receives HDR-BT transfers within the Device:
  - If the Device receives the MLANE Direct SET CCC with Defining Byte 0x23, then such a change shall only be accepted if the Device was previously configured to use Coding 0 (i.e., not in the 'sticky' state).
  - If the Device receives the MLANE Broadcast CCC with Defining Byte 0x23, then such a change shall always be accepted.
- While in a 'sticky' state, the Device shall not accept a configuration change via the MLANE Direct SET CCC with Defining Byte 0x23 to use any other ML Frame formats that are not interoperable with that "Alternate Mode" Coding.
- However, the Device shall continue to accept configuration changes via the MLANE Broadcast CCC with Defining Byte 0x23 to use any other ML Frame formats that are valid and supported. For such MLANE Broadcast CCCs, 'sticky' state checking is bypassed.

For HDR-BT Targets that support multiple assigned Addresses, HDR-BT defines several sets of conditions as valid actions (per ***Section 6.7.3.1***) that apply to the use of the MLANE CCC with "Alternate Mode" Codings:

- Entering the 'sticky' state (i.e., switching from Coding 0 to any "Alternate Mode" Coding) affects all assigned Addresses within the Device (i.e., the shared Peripheral logic that receives HDR-BT transfers and matches assigned Addresses within HDR-BT Header Blocks). The Device shall not accept any subsequent MLANE Direct SET CCCs that would attempt to re-configure one assigned Address to use a Data Transfer Coding that is incompatible (i.e., not interoperable) with the first "Alternate Mode" Coding that was used to put the Device into the 'sticky' state.
- If the Device supports separate ML configurations for Group Addresses (per ***Section 6.7.3.1.1***), then:
  - On entering the 'sticky' state, the Device shall change the ML configuration for all currently assigned Group Addresses to the common ML Frame format for that "Alternate Mode" Coding, per ***Table 106***. The Device shall also apply this same common ML Frame format as the initial ML configuration for any subsequently assigned Group Addresses.
  - Such a configuration change shall also limit the available ML Frame formats that the Device might accept upon subsequent use of the MLANE Direct SET CCC with Defining Byte 0x23.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

This shall prevent any assigned Group Address from being re-configured to use a Coding that is not interoperable with the Dynamic Address (or one of the Dynamic Addresses, if multiple Dynamic Addresses are supported).

- Per ***Section 6.7.3.1.1***, an MLANE Direct SET CCC sent to an assigned Group Address shall not cause the Device to enter or leave the 'sticky' state, nor shall the Device change the ML configuration for any assigned Dynamic Address as a result. The Device shall only accept an MLANE Direct SET CCC with an ML Frame format sent to an assigned Group Address, if that ML Frame format is compatible (i.e., interoperable) with the current ML configuration of the associated Dynamic Address.

- If the Device supports multiple assigned Dynamic Addresses (i.e., a composite Device that presents multiple Virtual Targets, per ***Section 6.7.3.1.2***), then:
  - The first such MLANE Direct SET CCC sent to any assigned Dynamic Address with an ML Frame format based on an "Alternate Mode" Coding shall also change the ML configuration for the Device's other assigned Dynamic Addresses, to use the common ML Frame format for that "Alternate Mode" Coding.

    In this case, the Controller may (and should) send subsequent MLANE Direct SET CCCs to the Device's other assigned Dynamic Addresses, with compatible ML Frame formats (or the same ML Frame format).
  - Similarly, the MLANE Broadcast CCC shall cause a simultaneous ML configuration change to all assigned Dynamic Addresses, as though the Device had received (and accepted) a series of MLANE Direct SET CCCs that were sent to each of its assigned Dynamic Addresses.
  - Note that the provided ML Frame format (i.e., the Data Byte after the Defining Byte) for each MLANE SET CCC (either Direct or Broadcast) must be valid and supported for the Device, if such configuration changes are to take effect, per ***Section 6.7.3.6***.
  - Subsequent MLANE Direct SET CCCs sent to any assigned Dynamic Address shall only be accepted if the provided ML Frame format is compatible with that same "Alternate Mode" Coding (i.e., uses the same Coding, or one that is expressly defined to be interoperable). This shall prevent any assigned Dynamic Address from being subsequently re-configured to use a Coding that is not interoperable with other Dynamic Addresses.
- If the Device supports both Group Addresses and multiple Dynamic Addresses (i.e., the union of both), then both sets of conditions listed above shall apply.

Once the Controller has configured the I3C Bus and all known HDR-BT capable Target Devices to use an "Alternate Mode" Coding, it must avoid such situations that might arise, in cases where a new Target Device might join or re-join the Bus (i.e., a Hot-Join Request, or a return from a deep sleep state with loss of configuration) and the Controller has not yet determined whether the Target Device supports HDR-BT Mode (and, if so, whether it also supports and can be configured to use an interoperable Coding). For some situations, the Controller must immediately stop all HDR-BT transfers if it detects an unknown Target Device, until it can fully determine such a Target Device's capabilities and current configuration.

An HDR-BT Target that has been configured to use Coding 0 with additional data Lanes (i.e., DUAL or QUAD transfers in Coding 0) may be switched back to 1-Lane compatibility mode (in Coding 0) via the MLANE CCC, using Defining Byte 0x23 and Data Byte 0x00, via Broadcast CCC or Direct CCC. This result can also be achieved by sending the MLANE CCC with the Sub-Command for Reset ML (i.e., Defining Byte 0x7F) in its Broadcast CCC or Direct SET CCC formats. However, an HDR-BT Target that has been configured to use an "Alternate Mode" Coding may only be configured back to 1-Lane compatibility mode (i.e., back to Coding 0 and leaving the 'sticky' state) by sending the MLANE Broadcast CCC (i.e., not a Direct SET CCC) with Defining Byte 0x23 and Data Byte 0x00, or by sending the MLANE Broadcast CCC with the Sub-Command for Reset ML.

Note:

When an HDR-BT Target that has been configured to use an "Alternate Mode" Coding leaves the 'sticky' state or accepts a new ML Frame format based on an "Alternate Mode" Coding (i.e., by

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

receiving the MLANE Broadcast CCC with either sub-command, as defined above), the ML configurations for all assigned Addresses shall be simultaneously configured back to either 1-Lane compatibility mode (i.e., with the Sub-Command for Reset ML), or to the new ML Frame format (i.e., with Defining Byte 0x23).

Figure 158 illustrates the valid state transitions for HDR-BT Targets that support any "Alternate Mode" Codings, when used with the MLANE SET CCC with Defining Byte 0x23.

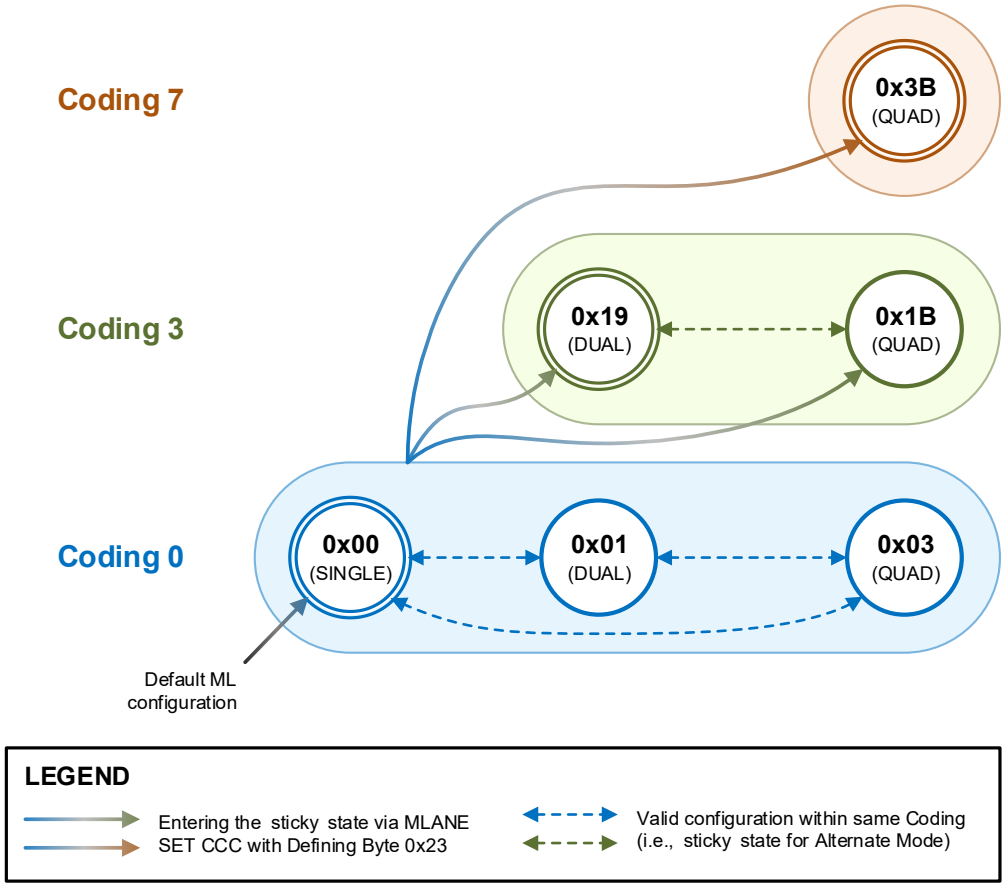


Figure 158 Valid State Transitions for HDR-BT Data Transfer Codings

Note:

*In **Figure 158**, the solid lines show the valid state transitions for entering the 'sticky' state from Coding 0, via the MLANE Direct SET CCC or the MLANE Broadcast CCC. The dashed lines show the valid state transitions within a currently configured Coding, via the MLANE Direct SET CCC with Defining Byte 0x23. Such state transitions apply to Devices that support a single Address (i.e., only a Dynamic Address) as well as Devices that support multiple Addresses (i.e., any combination of Virtual Targets with multiple Dynamic Addresses, or currently assigned Group Addresses). The circles with double lines show the common ML configurations for each Coding, which shall be used for each newly assigned Group Address (if supported). This diagram does not show the valid state transitions for leaving the 'sticky' state or changing to a different 'Alternate Mode' Coding via the MLANE Broadcast CCC, although such valid state transitions are always possible via the MLANE Broadcast CCC with Defining Byte 0x23.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

In a Bus that mixes 1-Lane Target Devices (or ML-capable Target Devices in their default 1-Lane configuration) with 2-Lane ML and/or 4-Lane ML Target Devices that are all configured for Coding 0, any Target Devices that are configured for SINGLE Lane Coding 0 shall ignore any HDR-BT Data Blocks and HDR-BT CRC Blocks that are part of a transaction (either generic, or Direct CCC flow) addressed to another Target Device that is configured for DUAL Lane Coding 0 or QUAD Lane Coding 0. Because these DUAL Lane or QUAD Lane formatted Blocks might not have the same format and length that a Target Device in its default configuration can understand, such a Target Device may instead ignore these HDR-BT Blocks and wait for the HDR Restart Pattern, after which it will be able to receive and understand a subsequent HDR-BT Header Block and correctly parse its **Address** field.

CCC transactions in HDR-BT that use Multi-Lane shall conform to the special requirements of the CCC framing and modality for HDR-BT (see ***Section 6.4.3.4***), as a specific instantiation of the generic CCC framing and modality for all HDR Modes (see ***Section 6.1.3***). Such HDR-BT-capable Targets shall monitor the Bus for HDR-BT transactions that are either HDR-BT generic write/read transfers, or common CCC flow elements, according to the currently configured Data Transfer Coding of the Device:

- **If such a Target only has a single Dynamic Address:** Then the ML configuration (i.e., currently configured ML Frame format) for the Target's Dynamic Address shall apply to match either the assigned Dynamic Address or the Broadcast Address (i.e., 7'h7E) for common CCC flow elements, according to the Data Transfer Coding for this ML configuration.
- **If such a Target has multiple Dynamic Addresses** (i.e., a composite Device that presents Virtual Targets, per ***Section 6.7.3.1.2***)**:** Then the ML configuration (i.e., currently configured ML Frame format) for all such Dynamic Addresses must be interoperable, so that the Target can match the Broadcast Address (i.e., 7'h7E) with the specified ML Frame formatting of the Data Transfer Coding, for common CCC flow elements (per ***Table 106***).

Coding 0

The default ML Frame format for HDR-BT Devices is Coding 0 (Compatible Mode) which requires no additional data Lanes.

Coding 0 is intended for Buses with HDR-BT Targets of mixed support for Multi-Lane's additional data Lanes. It is interoperable with all 1-Lane, 2-Lane ML, and 4-Lane ML HDR-BT Targets that support Coding 0.

In Coding 0:

- All HDR-BT Header Blocks shall be transmitted using SDA[0] (i.e., 1-Lane mode) in order to ensure that all Targets that support HDR-BT are always able to receive and understand the format of the Header Block. This includes the specific variants of the Header Block used for the common CCC flow elements in the HDR-BT CCC flows. The bit packing shall follow the formats shown in ***Table 87*** for Data Bytes and ***Table 90*** for the Transition Byte.
- For generic HDR-BT transactions (i.e., not part of the HDR-BT CCC flows), all subsequent HDR-BT Data Blocks and the HDR-BT CRC Block shall be transmitted in a bit packing format appropriate for the Target's currently configured ML Frame format (see ***Table 106***). If the Target is not ML-capable, or has not been configured to use a ML Frame format that uses additional data Lanes, then this bit packing format shall be the standard 1-Lane HDR-BT format (i.e., SDA[0] only).
- For all HDR-BT CCC flows, all common CCC flow elements shall be transmitted using SDA[0] (i.e., 1-Lane mode), so that all HDR-BT Targets can participate in the CCC flows regardless of their currently configured ML Frame format or the number of additional data Lanes they support. In this manner, all ML Frame formats that support Coding 0 are mutually interoperable for the common CCC flow elements.
- Following the HDR-BT CCC Selector Block in a Direct CCC flow, all subsequent HDR-BT Data Blocks and the HDR-BT CRC Block that are addressed to a Target currently configured for Coding 0 shall be transmitted in a bit packing format appropriate for the Target's currently configured ML Frame format (see ***Table 106***).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Coding 3

An optional Coding for HDR-BT Devices is Coding 3 (“Alternate Mode”), which requires at least one additional data Lane.

Coding 3 is intended for Buses with HDR-BT Targets that support at least one additional data Lane for Multi-Lane. It is interoperable with all other “Alternate Mode” Codings marked as Coding 3 in ***Table 105***, and expressly not interoperable with any 1-Lane HDR-BT Targets, or any 2-Lane ML HDR-BT Targets that do not support Coding 3. When addressing HDR-BT Targets that support three additional data Lanes, it provides additional transfer speed for these per-Target transactions, while still allowing interoperability with any other HDR-BT Targets that only support one additional data Lane and have been configured for Coding 3.

In Coding 3:

- All HDR-BT Header Blocks shall be transmitted using SDA[0] and SDA[1], using the DUAL Lane bit packing formats shown in ***Table 88*** for Data Bytes and ***Table 91*** for the Transition Byte. See ***Figure 138*** for the HDR-BT Header Block format, ***Figure 141*** for the HDR-BT Data Block format, and ***Figure 145*** for the HDR-BT CRC Block format.
- For generic HDR-BT transactions (i.e., not part of the HDR-BT CCC flows), all subsequent HDR-BT Data Blocks and the HDR-BT CRC Block shall be transmitted in a bit packing format appropriate for the Target’s currently configured ML Frame format, using either DUAL Lane or QUAD Lane formats (see ***Table 106***).
- For all HDR-BT CCC flows, all common CCC flow elements shall be transmitted using SDA[0] and SDA[1], using the same DUAL Lane bit packing formats. All HDR-BT Targets that are currently configured for Coding 3 can participate in the CCC flows regardless of the number of additional Lanes (i.e., Lanes beyond SDA[1]) that they are configured to use. In this manner, all ML Frame formats that support Coding 3 are mutually interoperable for the common CCC flow elements.
- Following the HDR-BT CCC Selector Block in a Direct CCC flow, all subsequent HDR-BT Data Blocks and the HDR-BT CRC Block addressed to a Target currently configured for Coding 3 shall be transmitted in a bit packing format appropriate for the Target’s currently configured ML Frame format, using either DUAL Lane or QUAD Lane formats (see ***Table 106***).

This “Alternate Mode” is not compatible with other ML-capable HDR-BT Targets not configured to use an “Alternate Mode” Coding marked as Coding 3, or other HDR-BT Targets that are not ML-capable. Such Targets will not correctly receive and understand any HDR-BT Header Blocks transmitted in this bit packing format, and as a result they will not participate in any HDR-BT transactions, and could even cause communication errors.

Note:

In Coding 3, the Controller shall not use the SINGLE Lane formats for any HDR-BT structured protocol elements, such as the HDR-BT Header Block, Data Block or CRC Block.

Once an HDR-BT capable Target that supports Coding 3 has been configured to use an ML Frame format that uses Coding 3 or any other “Alternate Mode” Coding that is interoperable, the entire Device enters a ‘sticky’ state and shall not accept any configuration changes via the MLANE CCC that might use any other Coding not interoperable with Coding 3, except for certain MLANE sub-commands sent as a Broadcast CCC. If the MLANE Direct SET CCC is used, then such a Device may only enter the ‘sticky’ state from Coding 0.

A Controller shall not enable any ML-capable HDR-BT Targets to use any “Alternate Mode” Coding marked as Coding 3, unless it first ensures that all other HDR-BT Targets are capable of interoperating with Coding 3. A Controller shall not initiate any HDR-BT transactions using DUAL Lane or QUAD Lane bit packing formats for all HDR-BT Blocks, unless it has previously configured all HDR-BT Targets on the Bus to use an “Alternate Mode” Coding marked as Coding 3. Once a Controller has configured all HDR-BT Targets on the Bus to use Coding 3, it must drive all HDR-BT transactions using the correct bit packing format for each HDR-BT Block, according to the conditions above.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

Coding 7

An optional Coding for HDR-BT Devices is Coding 7 (“Alternate Mode”), which requires three additional data Lanes.

Coding 7 is intended for Buses with HDR-BT Targets that support three additional data Lanes for Multi-Lane. It is not interoperable with any other Codings in ***Table 106***, and is expressly not interoperable with any 1-Lane or 2-Lane ML HDR-BT Targets, nor with any 4-Lane ML HDR-BT Targets that do not support Coding 7.

In Coding 7:

- All HDR-BT Blocks shall be transmitted using SDA[0], SDA[1], SDA[2], and SDA[3], using the QUAD Lane bit packing formats shown in ***Table 89*** for Data Bytes and ***Table 92*** for the Transition Byte. See ***Figure 138*** for the HDR-BT Header Block format, ***Figure 141*** for the HDR-BT Data Block format, and ***Figure 145*** for the HDR-BT CRC Block format.
- For generic HDR-BT transactions and for all HDR-BT CCC flows, all HDR-BT Data Blocks and HDR-BT CRC Blocks shall also be transmitted in the same QUAD Lane bit packing format.

This “Alternate Mode” is not compatible with other ML-capable HDR-BT Targets that are not configured to use an “Alternate Mode” Coding marked as Coding 7, or other HDR-BT Targets that are not ML-capable. Such Targets will not correctly receive and understand any HDR-BT Header Blocks transmitted in this bit packing format, and as a result they will not participate in any HDR-BT transactions, and could even cause communication errors.

Note:

In Coding 7, the Controller shall not use the SINGLE Lane or DUAL Lane formats for any HDR-BT structured protocol elements, such as the HDR-BT Header Block, Data Block or CRC Block.

Once an HDR-BT capable Target that supports Coding 7 has been configured to use an ML Frame format that uses Coding 7 or any other “Alternate Mode” Coding that is interoperable, the entire Device enters a ‘sticky’ state and shall not accept any configuration changes via the MLANE CCC that might use any other Coding not interoperable with Coding 7, except for certain MLANE sub-commands sent as a Broadcast CCC. If the MLANE Direct SET CCC is used, then such a Device may only enter the ‘sticky’ state from Coding 0.

A Controller shall not enable any ML-capable HDR-BT Targets to use any “Alternate Mode” Coding marked as Coding 7, unless it first ensures that all other HDR-BT Targets are capable of interoperating with Coding 7. A Controller shall not initiate any HDR-BT transactions using QUAD Lane bit packing formats for all HDR-BT Blocks, unless it has previously configured all HDR-BT Targets on the Bus to use an “Alternate Mode” Coding marked as Coding 7. Once a Controller has configured all HDR-BT Targets on the Bus to use Coding 7, it must drive all HDR-BT transactions using QUAD Lane bit packing formats for all HDR-BT Blocks.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.7.3.5.3.2 HDR-BT DUAL Coding

The default Coding for HDR-BT DUAL is Coding 0 (compatible mode); its additional Data Byte used by the MLANE CCC is 0x01 (i.e., {5'd0, 3'd1}).

In this ML Frame format, all HDR-BT Blocks that are part of generic HDR-BT transactions (except for the HDR-BT Header Blocks), or that comprise the per-Target CCC flow elements, shall be transmitted using SDA[0] and SDA[1], using the DUAL Lane bit packing formats shown in ***Table 88*** for Data Bytes and ***Table 91*** for the Transition Byte. See ***Figure 141*** for the HDR-BT Data Block format, and ***Figure 145*** for the HDR-BT CRC Block format. For additional details, see also Coding 0 in ***Section 6.7.3.5.3.1***.

Any Targets that are not configured for DUAL Lane Coding 0 might not be able to receive and understand any HDR-BT Data Blocks and HDR-BT CRC Blocks that are not part of a transaction (either generic, or Direct CCC flow) addressed to their Target Address. Because these Blocks might not have the same format and length as their currently configured ML Frame format, such a Target may instead ignore these HDR-BT Blocks and wait for the HDR Restart Pattern, after which it will be able to receive and understand a subsequent HDR-BT Header Block and correctly parse its **Address** field.

“Alternate Mode” Codings for DUAL Lane: Coding 3

In some Bus configurations, it might be desirable to utilize one additional data Lane for HDR-BT Blocks, for increased performance and Bus efficiency. This requires all Targets on the Bus that support HDR-BT to be configured to support ML Frame formats that are interoperable (i.e., that expect to use the same number of additional data Lanes in the same locations in HDR-BT transactions).

One “Alternate Mode” Coding for HDR-BT DUAL is Coding 3; its additional Data Byte used by the MLANE CCC is 0x19 (i.e., {5'd3, 3'd1}).

In this ML Frame format, HDR-BT Blocks that are part of generic HDR-BT transactions (except for the HDR-BT Header Blocks), or that comprise the per-Target CCC flow elements, shall be transmitted using SDA[0] and SDA[1], using the DUAL Lane bit packing formats listed above. For additional details, see also Coding 3 in ***Section 6.7.3.5.3.1***.

As previously stated, the HDR-BT Header Blocks and all HDR-BT Blocks comprising the core CCC flow elements shall be transmitted in DUAL Lane bit packing format.

In a Bus that mixes 2-Lane ML and 4-Lane ML Targets, all of which are configured for Coding 3, any Targets configured for DUAL Lane Coding 3 shall ignore any HDR-BT Data Blocks and HDR-BT CRC Blocks that are part of a transaction (either generic, or Direct CCC flow) addressed to another Target configured for QUAD Lane Coding 3.

Any Targets not configured for DUAL Lane Coding 3 might not be able to receive and understand any HDR-BT Data Blocks and HDR-BT CRC Blocks that are not part of a transaction (either generic, or Direct CCC flow) addressed to their Target Address. Because these Blocks might not have the same format and length as their currently configured ML Frame format, such a Target may instead ignore these HDR-BT Blocks and wait for the HDR Restart Pattern, after which it will be able to receive and understand a subsequent HDR-BT Header Block and correctly parse its **Address** field.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.7.3.5.3.3 HDR-BT QUAD Coding

The default Coding for HDR-BT QUAD is Coding 0 (Compatible Mode); its additional Data Byte used by the MLANE CCC is 0x03 (i.e., {5'd0, 3'd3}).

In this ML Frame format, all HDR-BT Blocks that are part of generic HDR-BT transactions (except for the HDR-BT Header Blocks), or that comprise the per-Target CCC flow elements, shall be transmitted using SDA[0], SDA[1], SDA[2], and SDA[3], using the QUAD Lane bit packing formats shown in ***Table 89*** for Data Bytes and ***Table 92*** for the Transition Byte. See ***Figure 141*** for the HDR-BT Data Block format, and ***Figure 145*** for the HDR-BT CRC Block format. For additional details, see also Coding 0 in ***Section 6.7.3.5.3.1***.

Any Targets not configured for QUAD Coding 0 might not be able to receive and understand any HDR-BT Data Blocks and HDR-BT CRC Blocks that are not part of a transaction (either generic, or Direct CCC flow) addressed to their Target Address. Because these Blocks might not have the same format and length as their currently configured ML Frame format, such a Target may instead ignore these HDR-BT Blocks and wait for the HDR Restart Pattern, after which it will be able to receive and understand a subsequent HDR-BT Header Block and correctly parse its **Address** field.

“Alternate Mode” Codings for QUAD Lane

In some Bus configurations, it might be desirable to utilize the three additional data Lanes for HDR-BT Blocks, for increased performance and Bus efficiency. This requires all Targets on the Bus that support HDR-BT to be configured to support ML Frame formats that are interoperable (i.e., that expect to use the same number of additional data Lanes in the same locations in HDR-BT transactions).

Coding 3

One “Alternate Mode” Coding for HDR-BT QUAD is Coding 3; its additional Data Byte used by the MLANE CCC is 0x1B (i.e., {5'd3, 3'd3}).

In this ML Frame format, HDR-BT Blocks that are part of generic HDR-BT transactions (except for the HDR-BT Header Blocks), or that comprise the per-Target CCC flow elements, shall be transmitted using SDA[0], SDA[1], SDA[2], and SDA[3], using the QUAD Lane bit packing formats listed above. See also Coding 3 in ***Section 6.7.3.5.3.1***.

As previously stated, the HDR-BT Header Blocks and all HDR-BT Blocks comprising the core CCC flow elements shall be transmitted in DUAL Lane bit packing format.

In a Bus that mixes 2-Lane ML and 4-Lane ML Targets, all of which are configured for Coding 3, any Targets configured for QUAD Lane Coding 3 shall ignore any HDR-BT Data Blocks and HDR-BT CRC Blocks that are part of a transaction (either generic, or Direct CCC flow) addressed to another Target configured for DUAL Lane Coding 3.

Any Targets not configured for QUAD Lane Coding 3 might not be able to receive and understand any HDR-BT Data Blocks and HDR-BT CRC Blocks that are not part of a transaction (either generic, or Direct CCC flow) addressed to their Target Address. Because these Blocks might not have the same format and length as their currently configured ML Frame format, such a Target may instead ignore these HDR-BT Blocks and wait for the HDR Restart Pattern, after which it will be able to receive and understand a subsequent HDR-BT Header Block and correctly parse its **Address** field.

Coding 7

Another “Alternate Mode” Coding for HDR-BT QUAD is Coding 7; its additional Data Byte used by the MLANE CCC is 0x3B (i.e., {5'd7, 3'd3}).

In this ML Frame format, all HDR-BT Blocks shall be transmitted using SDA[0], SDA[1], SDA[2], and SDA[3], using the QUAD Lane bit packing formats listed above. See also Coding 7 in ***Section 6.7.3.5.3.1***.

As QUAD Lane Coding 7 is not interoperable with any other Codings, nor with any other HDR-BT Targets supporting fewer than 3 additional data Lanes, it shall not be used in any Bus configuration in which it could cause communication errors.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.7.3.6 Multi-Lane Data Transfer Control (MLANE)

The MLANE CCC is available in Broadcast (***Figure 159***) and Direct (***Figure 160***) versions. It is used to set up and control ML functionality for ML-capable Target Devices (see ***Section 6.7***) by setting or getting, for a given I3C Mode (***Table 108***), the number of Additional Data Lanes (***Table 103***) and Data Transfer Coding number (see ***Section 6.7.3.3***) to use while communicating to the Target using that I3C Mode.

This CCC always includes the one-byte Defining Byte field, which acts as a GET, SET, or SET/GET sub-command. Many of the defined sub-commands that have Broadcast CCC or Direct SET CCC versions also require one or more additional Data Bytes, as detailed in ***Table 108***.

Note:

For SDR Mode Only: *The GET version of the SET/GET MLANE CCC requires the addressed Target to provide the DATA/T values on the Additional Data Lanes (i.e., on SDA[1] for DUAL Lane configurations, and on SDA[1], SDA[2], and SDA[3] for QUAD Lane configurations). The Controller can use this to test whether the Target is connected to the Additional Data Lanes. Since I3C Basic only supports Multi-Lane transfers for HDR-BT Mode, this shall apply for Defining Byte 0x23 only.* ***Figure 161*** *shows the CCC GET format for QUAD Lane configurations; the version for DUAL Lane configurations does not use SDA[2] or SDA[3].*

See Errata 01, Item 35

For HDR Modes: *See MLANE Command Name in* ***Table 75*** *for details on the format of the Target's response to the Format 2 (Direct GET) MLANE CCC while in HDR Modes.*

Table 107 Multi-Lane Data Transfer Control Command Codes (MLANE)

Command	Support	Command Codes		Purpose
		Broadcast	Direct	
MLANE	Conditional	0x2D	0x9D	Multi-Lane Data Transfer Control

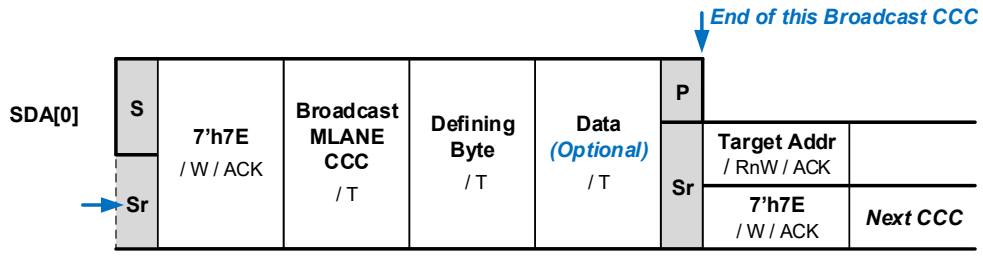


Figure 159 MLANE Format 1: Broadcast

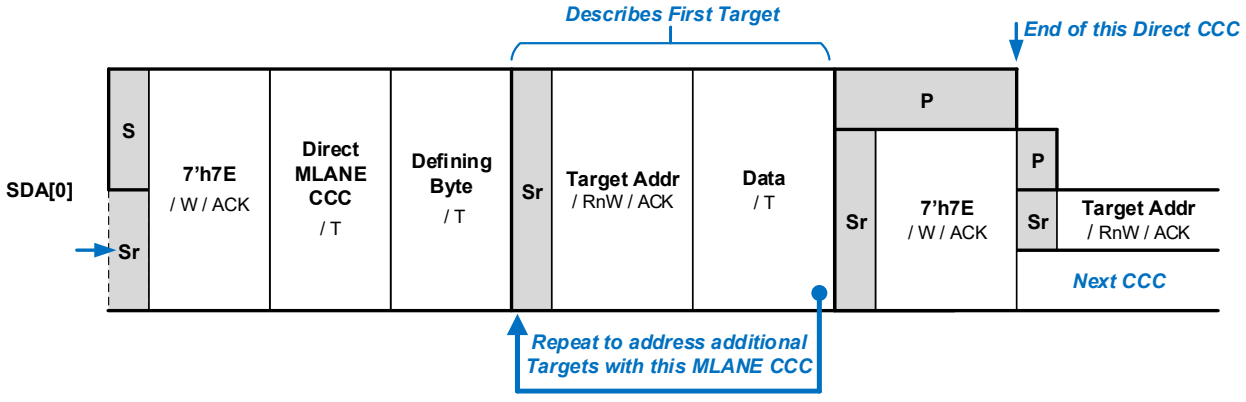


Figure 160 MLANE Format 2: Direct

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

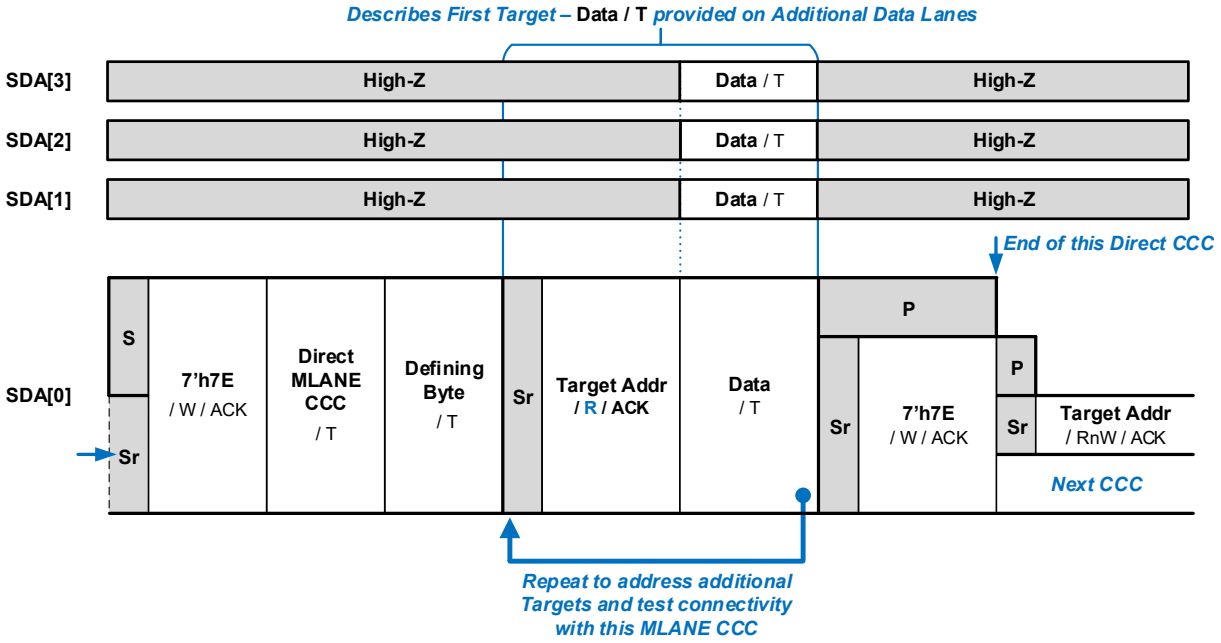


Figure 161 MLANE Direct GET Version of SET/GET CCC (SDR Mode Only)

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

7905

Table 108 MLANE Defining Bytes and Data Bytes

Defining Byte Value	Sub-Command Function	Description	Additional Data Bytes
0xFF	**Get ML Capabilities** Direct Read Get only	The directly addressed I3C Target responds with a 2-byte description of supported ML Frame formats, in every I3C Mode for which ML is supported: **Example:** For a Target capable of ML on HDR-DDR (Sub-Command 0x20), with both Coding 1 and Coding 2 on Quad Lanes, two byte pairs are needed. The first byte of both pairs is the HDR-DDR value (0x20). The second byte of the first pair is {5'd1, 3'd3} or 0x0B, and the second byte of the second pair is {5'd2, 3'd3} or 0x13. The addressed Target therefore transfers the four bytes: 0x20, 0x0B, 0x20, 0x13	One byte pair per every I3C Mode for which the Target supports ML functionality. Within each byte pair: • **First Byte:** Identifier for the I3C Mode supported (same values used in the Sub-Command Function column at left): **0x00:** SDR (Not supported in I3C Basic) **0x20:** HDR-DDR (Not supported in I3C Basic) **0x21:** HDR-TSP (Not supported in I3C Basic) **0x23:** HDR-BT • **Second Byte:** 3-bit number of Additional Data Lanes and 5-bit Data Transfer Coding number, formatted as described below
0x7F	**Reset ML** Broadcast or Direct Set	**Broadcast SET:** All I3C Targets return to Basic 2-wire I3C mode and a default Data Transfer Coding, for all supported ML Frame formats. **Direct SET:** The directly addressed Target returns to Basic 2-wire I3C mode and a default Data Transfer Coding, for all supported ML Frame formats. If the Target also supports separate Group ML configurations (see 0x9B and 0x9C), then these shall also be reset to the same default state.	No additional data bytes for this sub-command. Target(s) now return to a default state, where they can be addressed in Basic 2-wire I3C mode. Useful before a Controller Handoff to an I3C Secondary Controller that is not connected to additional Lanes.
0x00	**SDR-ML Frame Format** Set/Get *(Not supported in I3C Basic)*	Multi-Lane support for SDR Mode is not included in I3C Basic. To gain access to this capability, please contact MIPI Alliance.	
0x20	**HDR-DDR-ML Frame Format** Set/Get *(Not supported in I3C Basic)*	Multi-Lane support for HDR-DDR Mode is not included in I3C Basic. To gain access to this capability, please contact MIPI Alliance.	
0x21	**HDR-TSP-ML Frame Format** Set/Get *(Not supported in I3C Basic)*	Multi-Lane support for HDR-TSP Mode is not included in I3C Basic. To gain access to this capability, please contact MIPI Alliance.	

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Defining Byte Value	Sub-Command Function	Description	Additional Data Bytes
0x23	**HDR-BT Frame Format** Set/Get	Sets or returns the current ML configuration (i.e., the ML Frame format) for a given I3C Mode. The Defining Byte value indicates the I3C Mode and its available ML Frame formats to which the sub-command shall apply: • **0x23:** HDR-BT ML Frame formats: see ***Section 6.7.3.5.3*** **Direct SET to an assigned Dynamic Address**: The Target uses the provided ML Frame format variation with the indicated number of Additional Data Lanes and Data Transfer Coding version. **Direct SET to an assigned Group Address** (if supported, per ***Section 6.7.3.1.1***): The Target uses the provided ML Frame format variation (as above) for all subsequent transfers addressing that Group Address. Changes shall not apply to the ML configuration for Dynamic Address(es). **Broadcast SET:** If the Target supports the given I3C Mode, then it shall use the provided ML Frame format, for all assigned Dynamic Addresses and Group Addresses (if applicable). See ***Section 6.7.3.1*** and sub-sections for full configuration requirements. **Direct GET to an assigned Dynamic Address**: The Target responds with one byte (see right column) indicating which variation of the ML Frame format is selected for ML operations involving that Dynamic Address. If the Target also supports separate Group ML configurations (per ***Section 6.7.3.1.1***) or multiple Dynamic Addresses (per ***Section 6.7.3.1.2***), then the data byte shall apply to transactions involving that Dynamic Address (i.e., not any other assigned Addresses).	One byte as the ML Frame format, concatenating: • **Bits[2:0]:** Number of Additional Data Lanes: **3'b000:** Basic 2-wire I3C: SINGLE Lane config **3'b001:** One additional Lane: DUAL Lane config **3'b011:** Three additional Lanes: QUAD config Other values not used • **Bits[7:3]:** Data Transfer Coding number to use (see ***Section 6.7.3.3***) For Broadcast or Direct SET: The Target shall only accept a valid and supported ML Frame format for a given I3C Mode, as indicated by Defining Byte 0xFF. See sub-sections below. If the Target also supports Group Address capabilities (per ***Section 4.3.4.4***), then it shall also support Defining Byte 0x9B (i.e., the sub-command for **Group ML Capabilities**). If the Target also supports separate Group ML configurations (per ***Section 6.7.3.1.1***), then it shall also support Defining Byte 0x9C (i.e., the sub-command for **Get Group ML Report**). For additional details and requirements, see ***Section 6.7.3.1*** and sub-sections. Specific exceptions may apply per I3C Mode, per ***Section 6.7.3.3*** and sub-sections.
0x9B	**Group ML Capabilities (optional)** Direct Read Get only	**Direct GET to an assigned Dynamic Address:** The directly addressed Target responds with one byte (see Additional Data Bytes column) indicating whether it supports separate Group ML configurations, and if so, in what way. Required if the Target supports Group Addresses in any I3C Modes using ML. If the Target does not support Group Addresses in any I3C Modes using ML, then it shall NACK its Target Address. See ***Section 6.7.3.1.1*** for additional requirements.	One byte, concatenating: • **Bits [1:0]:** Whether separate Group ML configurations are supported: **2'b00:** Not Supported (all configured Group Addresses use the same ML Frame format as the Target Address) **2'b01:** Supported; each newly configured Group Address has its own configuration, set to a common frame format, when the Target is assigned to a Group Address Other values not used • **Bits [5:2]:** The number of Group Addresses to which the Target is currently assigned (value from 4'd0 to 4'd15) • Bits [7:6]: Reserved

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Defining Byte Value	Sub-Command Function	Description	Additional Data Bytes
0x9C	**Get Group ML Report (optional)** Direct Read Get only	**Direct GET to an assigned Dynamic Address:** The directly addressed Target responds with a report on the number of separate Group ML configurations (i.e., separate ML Frame formats for a Group Address) that it will support, followed by a 3-byte tuple describing the currently configured ML Frame format for each supported I3C Mode for each Group Address to which the Target is assigned. Required if the Target supports Group Addresses in any I3C Modes using ML. If the Target does not support separate Group ML configurations, then it may return a 1-byte message of 0x00, or NACK its Target Address. **See Errata 01, Item 36** **Example:** For a Target that supports up to 4 Group Addresses, supports ML only in HDR-DDR Mode, and that is currently assigned to 3 Groups, the Target returns the byte 0x04 to indicate 4 separate Group ML configurations, followed by a 3-byte tuple for each. The first byte of each tuple is the Group Address, the second byte is the HDR-DDR value (0x20), and the third byte is the ML Frame format's data byte. This Target is assigned to Group Addresses 7'h71, 7'h72, and 7'h74. The addressed Target therefore transfers the following thirteen bytes: 0x04, The number of Group ML configurations across all I3C Modes 0x71, 0x20, 0x00, Group Address 7'h71, HDR-DDR mode 0x00, 1-Lane default 0x72, 0x20, 0x09, Group Address 7'h72, HDR-DDR mode 0x09, 2-Lane ML with Coding 1 0x74, 0x20, 0x13, Group Address 7'h74, HDR-DDR mode 0x13, 4-Lane ML with Coding 2 0x00, 0x00, 0x00 Not configured	One byte for the number of separate Group ML configurations that can be configured, followed by a 3-byte tuple for each described Group ML configuration. Within each byte tuple: • **First Byte:** The Group Address (if assigned), or 0x00 (if not configured, which means the Second Byte and Third Byte are not valid) • **Second Byte:** If configured, the identifier for the I3C Mode supported (see 0xFF above) • **Third Byte:** If configured, the 3-bit number of Additional Data Lanes and 5-bit Data Transfer Coding number, formatted as described above For all valid Group ML configuration tuples, the first two bytes shall be unique. • All valid (configured) Group ML configurations shall be returned before any invalid (non-configured). A Target may terminate the transfer after at least one Group ML configuration tuple consisting only of all ZEROs. **See Errata 01, Item 37**
All Others	Reserved	Available Defining Byte values. Reserved for future definition of new MLANE CCC Sub-Commands.	–

Note:

*An I3C Controller that supports any version of the I3C Basic Specification might be on the same I3C Bus with an I3C Target that supports the full I3C Specification, and it might receive a report using the MLANE Direct GET CCC with the Sub-Command for Get ML Capabilities (i.e., Defining Byte 0xFF) or other Sub-Commands that include ML Frame formats for I3C Modes that the I3C Controller does not support. In such a case, the I3C Controller shall ignore any ML Frame formats that pertain to such I3C Modes (i.e., those marked as "Not included in I3C Basic" in **Table 108**), and shall not use the corresponding Sub-Commands for such I3C Modes, as the I3C Controller would not have the capability to support Multi-Lane Data Transfers for such I3C Modes.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.7.3.6.1 Direct SET Sub-Commands Sent to a Dynamic Address

The Controller may send the MLANE Direct SET CCC with the Sub-Commands to set the ML Frame format for supported I3C Modes (i.e., Defining Byte 0x23) when addressed to a Target Device's Dynamic Address. The Target shall receive and acknowledge the CCC, to write a supported ML Frame format with the Additional Data Byte for that I3C Mode, and the Target shall store this Data Byte in its ML configuration (i.e., configured ML Frame format) for the corresponding I3C Mode. Unless defined otherwise for a particular I3C Mode, the Target shall return this same Data Byte from its stored ML configuration upon receiving the MLANE Direct GET CCC with the same sub-command, addressed to the same Dynamic Address. The Target shall not reset or change this stored ML configuration, unless it properly receives the Sub-Command for Reset ML, or receives any other valid action that might be defined for specific I3C Modes (i.e., as defined in sub-sections of ***Section 6.7.3.3***). The stored ML configuration for each supported I3C Mode shall be stored and used for subsequent transfers involving this Dynamic Address (and the Broadcast Address, if the Target supports CCC flows in HDR Modes per ***Section 6.1.3***) unless it is subsequently reset, or changed by a SET Sub-Command (or another valid action). See ***Section 6.7.3.1*** for more details on ML-capable Device configuration.

If the Device presents multiple Virtual Targets and uses shared Peripheral logic to manage transactions addressed to multiple Dynamic Addresses (per ***Section 4.3.2.1.2***), then the ML configurations (i.e., configured ML Frame formats) for each assigned Dynamic Address shall be configured and stored independently. All conditions above shall apply for each Dynamic Address, and the Device shall honor the SET/GET semantics for the MLANE Direct CCC with Sub-Commands addressed to any Dynamic Addresses. Each CCC with Sub-Command to set a stored ML configuration shall only change the ML configuration for the particular Dynamic Address (per ***Section 6.7.3.1*** and ***Section 6.7.3.1.2***) unless other requirements apply for specific I3C Modes (i.e., as defined in sub-sections of ***Section 6.7.3.3***) that have valid actions that would create exceptions to the requirements in ***Section 6.7.3.1.2*** for independent ML configurations per each Dynamic Address.

Note:

The Controller should determine whether such a Device has multiple Dynamic Addresses and presents multiple Virtual Targets using shared Peripheral logic, before attempting to set an ML configuration with the MLANE Direct SET CCC to any Dynamic Address exposed by such a Device.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.7.3.6.2 Direct SET Sub-Commands Sent to a Group Address

If the Target Device supports Group Address capabilities (per ***Section 4.3.4.4***) and also supports separate Group ML configurations (i.e., variant 2'b01 per ***Section 6.7.3.1.1***) then the Target shall also receive and acknowledge the MLANE Direct SET CCC with Sub-Commands for supported I3C Modes (i.e., Defining Byte 0x23) when addressed to an assigned Group Address. If the Target accepts the CCC with such a Sub-Command, then it shall only set its stored ML configuration (i.e., configured ML Frame format) for transactions involving that Group Address, but shall not change the stored ML configuration for any other assigned addresses. For HDR Modes, if the Target accepts the CCC with such a Sub-Command, then the stored ML configuration (if valid) shall be used for all subsequent transfers addressed to that Group Address (i.e., generic HDR Write commands); if the Target also supports CCC flows in that HDR Mode (per ***Section 6.1.3***), then this ML configuration shall also be used for Direct SET CCCs that are addressed to that Group Address in that particular HDR Mode (per ***Section 6.1.3.5***). See ***Section 6.7.3.1.1*** for more details on Group ML configuration. The Target shall not acknowledge the CCC with such a Sub-Command when addressed to any Group Address to which it has not been assigned.

If the Target supports Group Address capabilities (per ***Section 4.3.4.4***) and does not support separate Group ML configurations (i.e., variant 2'b00, per ***Section 6.7.3.1.1***), then the Target shall not acknowledge the MLANE Direct SET CCC with Sub-Commands for supported I3C Modes (i.e., Defining Byte 0x23) when addressed to an assigned Group Address. For such a Target, the stored ML configuration (i.e., configured ML Frame format) for transfers involving this Dynamic Address shall also be used for subsequent transfers involving any assigned Group Addresses. See ***Section 6.7.3.1.1*** for more details.

Note:

If the Target does not support Group Address capabilities, then the Target shall only acknowledge the MLANE Direct SET CCC with Sub-Commands for its supported I3C Modes (i.e., Defining Byte 0x23) when addressed to its assigned Dynamic Address.

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

6.7.3.6.3 Broadcast SET Sub-Commands

The Controller may also send the MLANE CCC with the Sub-Commands to set the ML Frame format for supported I3C Modes (i.e., Defining Byte 0x23) as a Broadcast CCC, and all Target Devices on the I3C Bus shall acknowledge the CCC (per ***Table 15***). However, only Targets that support the MLANE CCC and also support the particular I3C Mode indicated by the Defining Byte may attempt to apply the ML configuration that follows the Defining Byte in the Broadcast SET CCC message. Each such Target shall compare the ML configuration to see whether it is valid (i.e., supported) for that I3C Mode: if so, then it shall store this Data Byte in its ML configuration (i.e., configured ML Frame format) for the corresponding I3C Mode, as applied to its assigned Dynamic Address (or all assigned Dynamic Addresses, if it is a Device that presents multiple Virtual Targets, per ***Section 4.3.2.1.2*** and ***Section 6.7.3.1.2***) as well as all currently assigned Group Addresses (if supported, per ***Section 6.7.3.1.1***). See ***Section 6.7.3.1*** for more details on ML-capable Device configuration.

If the provided ML configuration is not valid (i.e., not supported) for that I3C Mode, then the Target shall not apply the Data Byte and shall not make any configuration changes based on the Broadcast SET CCC message. Additionally, if the Target does not support the particular I3C Mode indicated by the Defining Byte, then it shall not act on the rest of the Broadcast SET CCC message. However, in both cases the Controller would not know that the Target did not make any changes to its ML configuration as a result of this Broadcast SET CCC, so the Controller should follow up by sending the MLANE Direct GET CCC with the same sub-command, addressed to each Dynamic Address that is known to support that particular I3C Mode.

Note:

If some I3C Targets on the Bus do not support certain I3C Modes, or if some Targets do not support certain ML Frame formats for a given I3C Mode, then the MLANE Broadcast CCC with a given Defining Byte and Data Byte would not always configure all such Targets to use the same ML Frame format for the same I3C Mode. It is the responsibility of the I3C Controller to determine which ML configuration changes (including those sent using the MLANE CCC as a Broadcast SET CCC) have been accepted by all Targets, and whether any such Targets might not have acted on the Broadcast SET CCC message. For situations where the Controller discovers that some Targets did not accept or apply such an ML configuration change, the Controller must resolve the situation by ensuring that all Targets are configured to use ML Frame formats that are mutually interoperable for each I3C Mode, or using an error recovery procedure which might include the Sub-Command for Reset ML (i.e., Defining Byte 0x7F).

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.8 F007: Monitoring Device

This Section defines I3C Optional Feature F007: Monitoring Device.

Table 109 Optional Feature 007 Status

Optional Feature ID	F007
Optional Feature Name	Monitoring Device
Optional Feature Version	1
Optional Feature specification was last updated in I3C Spec Version	I3C v1.2
Optional Feature is not compatible with other Optional Features	–
Optional Feature is included in I3C Basic	No

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.9 F008: D2D Tunneling

This Section defines I3C Optional Feature F008: D2D Tunneling.

Table 110 Optional Feature 008 Status

Optional Feature ID	F005 **See Errata 01, Item 38**
Optional Feature Name	D2D Tunneling
Optional Feature Version	1
Optional Feature specification was last updated in I3C Spec Version	I3C v1.2
Optional Feature is not compatible with other Optional Features	–
Optional Feature is included in I3C Basic	No

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.10 F009: Controller Clock Stalling

This Section defines I3C Optional Feature F009: Controller Clock Stalling.

Table 111 Optional Feature 009 Status

Optional Feature ID	F009
Optional Feature Name	Controller Clock Stalling
Optional Feature Version	1
Optional Feature specification was last updated in I3C Spec Version	I3C v1.2
Optional Feature is not compatible with other Optional Features	–
Optional Feature is included in I3C Basic	Yes

Changes in this Version

For detailed changes in this version, see the "diff" file between I3C v1.1.1 and I3C v1.2 in the MIPI Specification Repository ***[MIPI09]***.

6.10.1 Scope and Purpose (Informative)

The scope of ***Section 6.10*** is limited to fully specifying I3C Optional Feature F009: Controller Clock Stalling. An I3C Device that implements F009 is required to implement all **shall** statements in ***Section 6.10.3***, and can optionally implement any combination of **may** statements in ***Section 6.10.3***. In addition, MIPI Alliance recommends that manufacturers follow all **should** statements in ***Section 6.10.3***.

6.10.2 Technical Overview (Informative)

In SDR Mode the I3C Controller is permitted to Stall the I3C Bus during the SCL Low period, but only under the specific, transitory conditions described in this Section.

Stalling may be necessary for either of two reasons:

1. The absolute or relative timing of a Message to a specific Target, or to all Targets, needs to be carefully controlled. Clock Stalling provides the Controller with fine-grained data timing control.
2. The I3C Controller needs to internally synchronize data. This may be due to parts of the Controller's system waking up in response to data, to changing state, or to otherwise needing time during a transaction.

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.10.3 Technical Specification

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

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

1. I3C/I^{2}C Transfer, ACK/NACK Phase
2. Write Data Transfer, Parity Bit
3. I3C Read Transfer, Transition Bit
4. Dynamic Address Assignment, First Bit of Assigned Address

The maximum Stall time (SCL Low period) shall be 15 ms; note, this is an absolute maximum. The Controller should use the shortest Stall duration possible given current circumstances, per ***Table 112***.

Table 112 Controller Clock Stall Times

Condition	Section	Maximum Stall Time
I3C/I^{2}C Transfer, ACK/NACK Phase	***6.10.3.1***	100 µs
Write Data Transfer, Parity Bit	***6.10.3.2***	100 µs
I3C Read Transfer, Transition Bit	***6.10.3.3***	100 µs
Dynamic Address Assignment, First Bit of Assigned Address	***6.10.3.4***	15 ms

In all cases, after Stalling the SCL Low period, the continuation of the clock (i.e., the first SCL rising edge after the extended Low period) shall follow the normal rules for I3C SDR (for I3C SDR Messages), or for I^{2}C (for I^{2}C Messages) including rise time, change in SDA (if any) before SCL rise, next bit, etc. For example, the I3C Controller might choose to terminate the Frame after this Stalled bit with a STOP; but in this case the Controller is required to observe the requirements for STOP timing, as appropriate.

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

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.10.3.1 I3C/I²C Transfer, ACK/NACK Phase

Controller Clock Stalling during the ACK/NACK phase of the I3C/I²C transfer (see ***Figure 162***) indicates one of the following:

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

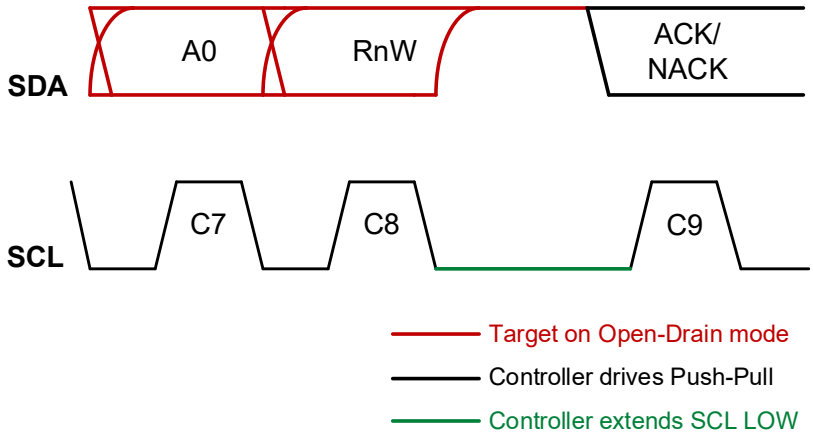


Figure 162 Controller Clock Stalling in ACK Phase

6.10.3.2 Write Data Transfer, Parity Bit

The Controller can Stall SCL between data bytes of an I3C Write Data transfer, by Stalling the clock during the Low period of the Parity Bit, in case of Underrun situations. See ***Figure 163***.

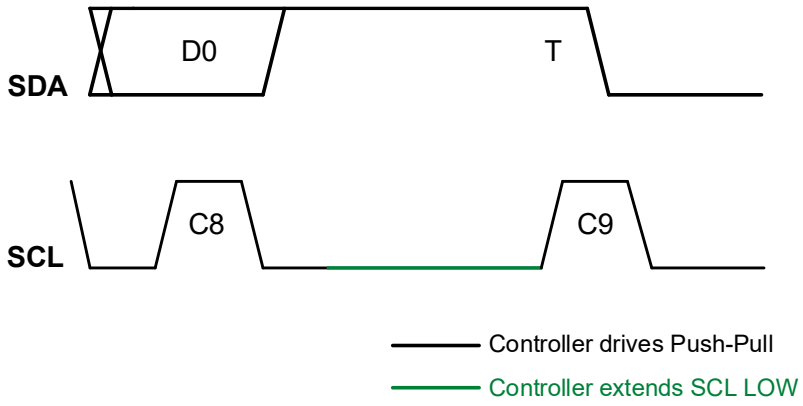


Figure 163 Controller Clock Stalling in Write Parity Bit

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.10.3.3 I3C Read Transfer, Transition Bit

The Controller can Stall SCL between data bytes of an I3C Read Data transfer, or before terminating, by Stalling the clock during the Low period of the Transition Bit, in case of Overflow situations. See ***Figure 164*** through ***Figure 168***.

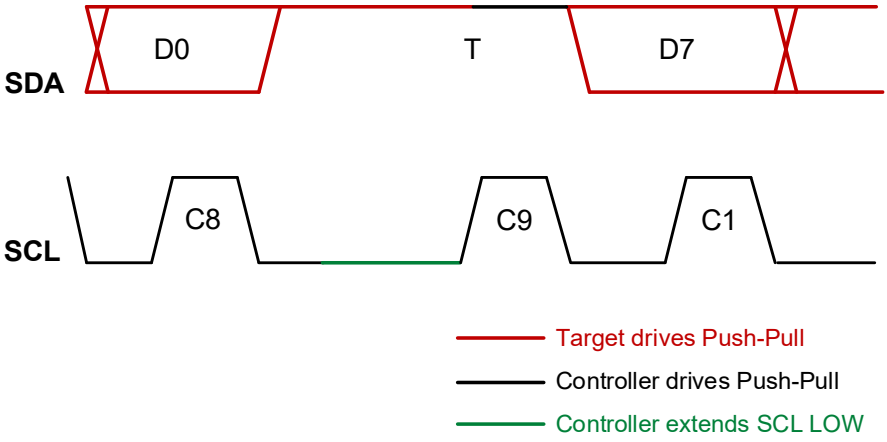


Figure 164 Controller Clock Stalling in T-Bit Before Next Read Data

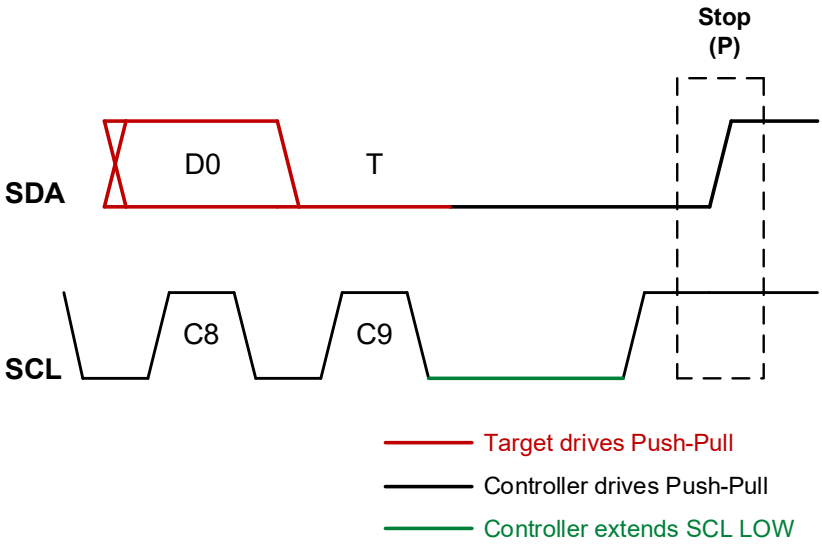
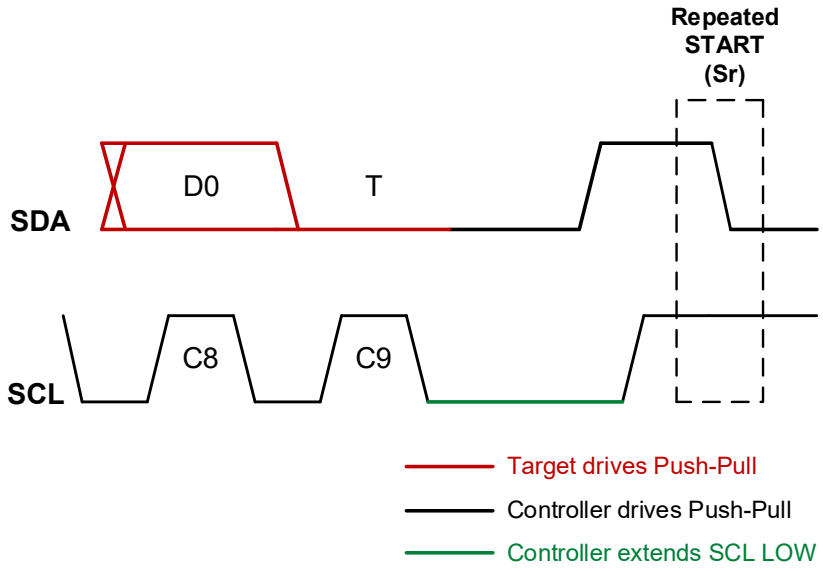


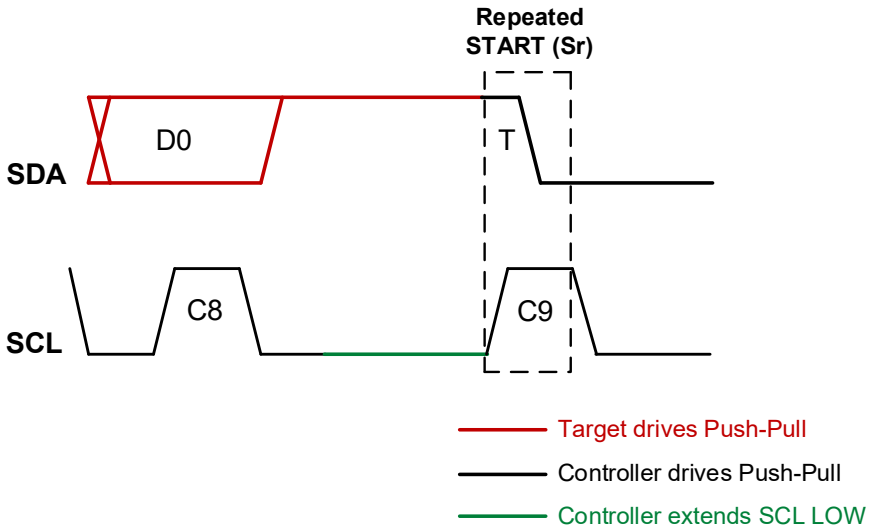
Figure 165 Controller Clock Stalling in T-Bit Before STOP

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



8061

Figure 166 Controller Clock Stalling in Low T-Bit Before Repeated START



8062

Figure 167 Controller Clock Stalling in High T-Bit Before Repeated START

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

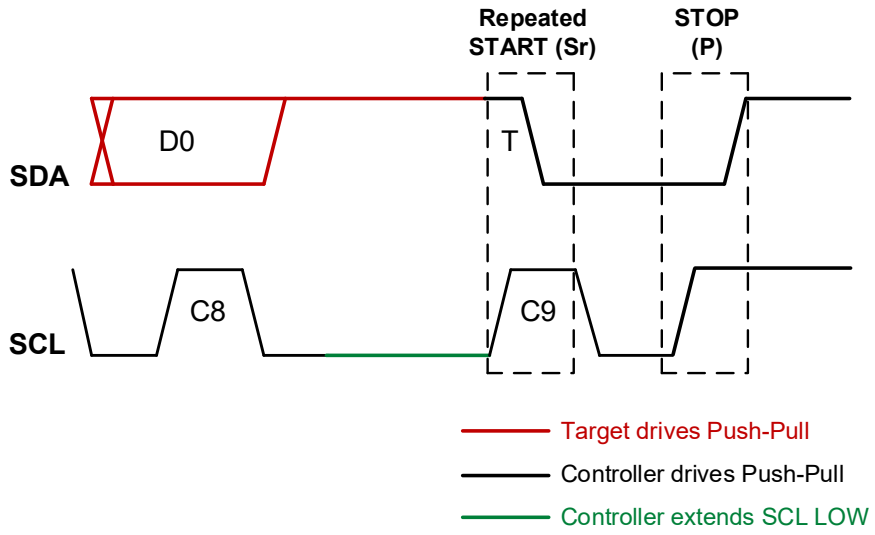


Figure 168 Controller Clock Stalling in High T-Bit Before Repeated START and STOP

6.10.3.4 Dynamic Address Assignment, First Bit of Assigned Address

The Controller can Stall SCL during the Low period of the first bit of the Assigned Address phase of the Enter Dynamic Address Assignment CCC command (ENTDAA, see ***Section 4.3.7.3.4***), for example to gain time to assign the Dynamic Address to the Device based on the Target's Bus Characteristics Register BCR (see ***Section 4.3.1.2.1***) and Device Characteristics Register DCR (see ***Section 4.3.1.2.2***) bytes. See ***Figure 169***.

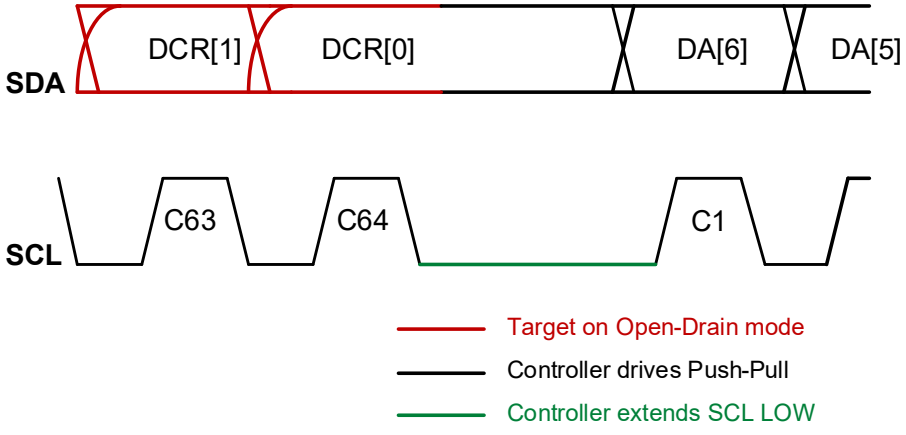


Figure 169 Controller Clock Stalling in Dynamic Address First Bit

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.11 F010: Alternative Electrical Specifications

This Section defines I3C Optional Feature F010: Alternative Electrical Specifications.

Table 113 Optional Feature 010 Status

Optional Feature ID	F010
Optional Feature Name	Alternative Electrical Specifications
Optional Feature Version	1
Optional Feature specification was last updated in I3C Spec Version	I3C v1.2
Optional Feature is not compatible with other Optional Features	–
Optional Feature is included in I3C Basic	Yes

Changes in this Version

This feature is new for I3C v1.2.

6.11.1 Scope and Purpose (Informative)

The scope of ***Section 6.11*** is limited to fully specifying I3C Optional Feature F010: Alternative Electrical Specifications.

An I3C Device that implements F010 may choose any of the available sub-sections within ***Section 6.11.2*** as a alternative definitions of the electrical specifications, rather then strictly following the standard electrical specifications defined in **Required Elements** ***Section 4.3.11***.

For each of the chosen sub-sections that apply, an I3C Device that uses alternative electrical specifications is required to implement all **shall** statements in ***that sub-section***, and can optionally implement any combination of **may** statements in ***that sub-section***. In addition, MIPI Alliance recommends that manufacturers follow all **should** statements in ***that sub-section***.

Note:

*As a result, it is possible that **shall**, **should**, and **may** statements for this Optional Feature will override some of the requirements given in the standard electrical specifications defined in **Required Elements Section 4.3.11**.*

The purpose of this Optional Feature is to provide flexibility for implementers addressing special use cases where the standard electrical specifications defined in ***Section 4.3.11*** would be infeasible or unsuitable. This allows the I3C Specification to be used as a base specification, despite differences in the native electrical specifications as defined by other standards bodies that govern those use cases.

Unless specified otherwise, the alternative electrical specifications in this Optional Feature allow the same I3C protocol to be used in special use cases where the standard electrical specification does not work for a particular use case, but the upper protocol layers would still be used.

In some cases, vendors and other standards bodies might define extensions to, or deviations from, the standard I3C content protocol. This may include custom CCCs as well as other methods of I3C Target discovery and configuration. The I3C Specification does not preclude this, however implementers are cautioned that such extensions or deviations might made an I3C Device usable only for the intended use case, i.e., not usable for general use cases.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

6.11.2 Technical Specification

Alternative electrical specifications may optionally be used for special I3C Bus use cases.

Note:

*The MIPI I3C WG strongly recommends that I3C Device implementers use the standard electrical specifications (see **Section 4.3.11**) for I3C Devices that are meant for general usage across a broad set of use cases, unless there is a compelling reason why the standard electrical specifications would not be suitable for a specific use case.*

6.11.2.1 Low Voltage / High Capacitive Load I3C Use Cases

For special Buses with lower voltage and higher capacitive loads (i.e., greater than 50 pF), the following characteristics should be used. In some scenarios, Buses with higher capacitive loads might not be able to achieve the maximum clock speed of 12.5 MHz.

Note:

*The characteristics defined in **Table 114** were originally included only in v1.1.1 and earlier of the MIPI I3C Basic Specification **[MIPI11]**. These characteristics are intended for special use cases only, i.e., the SPD bus that is used with JEDEC-compliant DDR5 memory modules.*

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

8112

Table 114 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	V_{DD}	–	0.95	1.0	1.05	V	–
Low-Level Input Voltage	V_{IL}	–	−0.3	–	0.3	V	–
High-Level Input Voltage	V_{IH}	–	0.7	–	1.05	V	1
Schmitt Trigger Inputs Hysteresis	V_{hys}	–	0.1 * V_{DD}	–	0.4 * V_{DD}	V	–
Output Low Level	V_{OL}	I_{OL} = 4 mA	–	–	0.3	V	2
Input Current (per Input-Only I/O Pin)	I_i	-100 mV < Vi < V_{DD}	−5	–	5	µA	2
Capacitance (per I/O Pin)	C_i	–	–	–	5	pF	–
Capacitance Mismatch Between Pins	ΔC	–	–	–	1.5	pF	–
Push-Pull Only							
Output High Level	V_{OH}	I_{OH} = -3 mA	0.75	–	–	V	2
Legacy Mode with Pull-Up							
Pull-Up for Open Drain	R_p	t_r = Max rise time (120 ns) C_b = Bus Capacitance (100 pF) V_{DD} = 1 V	$\frac{V_{DD} - V_{OL}}{4\ mA}$		$\frac{t_r}{0.8473\ *\ Cb}$	Ω	3, 4, 6, 7, 8

Note:

While these characteristics have been broadly used for JEDEC-compliant SPD buses, MIPI Alliance has not fully validated these characteristics for use in general-purpose I3C Buses.

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) For higher capacitance Buses, Pull-Up for Open Drain should be switched off during I3C Push-Pull operation. This may be implemented as: A Pull-Up internally; a current source internally; an external Pull-Up resistor driven by a pin; or any combination of the above. The effective resistance comprising the Open Drain Pull-Up and High-Keeper Pull-Up shall be sized sufficiently to allow the Target to pull SDA Low within t_{DIG_L} for adjusted clock speeds, and to allow SDA to rise within t_{rDA}, per **Section 4.3.3.1**.*

6) Open-Drain never occurs on SCL

7) A weak pull-up needs to be applied as well for Controller handoff

8) R_p Min should be decided based on the V_{IL} specification

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

ANNEXES

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Annex A Changes to Required Elements

This Annex summarizes the changes made to ***Section 4 Required Elements*** in version 1.2 of this Specification.

For detailed changes in this version, see the "diff" file between I3C Basic v1.1.1 and I3C Basic v1.2 in the MIPI Specification Repository ***[MIPI09]***.

Changes to Required Elements in I3C Basic Version 1.2

- **Document reorganization to move all assigned values to *Section 5 Assigned Values*:**
  - SETBUSCON CCC context bytes moved to ***Section 5.2 SETBUSCON Context Byte***
  - GETCAPS capability bytes (for both Format 1 and Format 2) moved to ***Section 5.4 GETCAPS Capability Bits***
  - IBI MDB values moved to ***Section 5.5 In-Band Interrupt MDB (Mandatory Data Byte)***
- **Document reorganization to move every optional feature to an independent subsection of *Section 6 Optional Features*:**
  - HDR Modes (common definitions) moved to ***Section 6.1***
  - HDR-DDR Mode moved to ***Section 6.2***
  - HDR-Ternary Modes moved to ***Section 6.3*** (details not included for I3C Basic)
  - HDR-BT Mode moved to ***Section 6.4***
  - Secondary Controller requirements and definitions moved to ***Section 6.5***
  - Timing Control feature moved to ***Section 6.6***:
    - This includes the definitions of the SETXTIME and GETXTIME CCCs
  - Multi-Lane Data Transfer feature moved to ***Section 6.7***:
    - This includes the definition of the MLANE CCC
  - Monitoring Device feature moved to ***Section 6.8***
  - ENDXFER CCC details moved:
    - Moved to ***Section 6.2*** for HDR-DDR Mode
    - Not included for HDR-Ternary Modes
    - To ***Section 6.8*** for Monitoring Devices
  - Device-to-Device Tunneling feature moved to ***Section 6.9***
  - Controller Clock Stalling requirements and definitions moved to ***Section 6.10***
- **All items that were previously addressed in Errata 01 for I3C Basic v1.1.1, including:**
  - Removed restrictions on values used in SETMWL and GETMWL CCCs
  - Requirements for an I3C Basic Target that initially acted as an I²C Target with a Spike Filter enabled
  - Updated default read value for RSTACT Direct Read CCC; new value of 0x80 means that no reset action has been configured
- **Clarifications on timing parameter t_{SCO}** in several diagrams
- **Updates for Targets sending IBI payloads:**
  - When sending IBI data payload, a Target is now required to observe the configured MRL (Maximum Read Length) values
  - Clarified that a Target is required to use the T-Bit (i.e., T=0) to end the IBI data payload after sending the last byte
- **When a Target's Dynamic Address is reset, all assigned Group Addresses are also reset**
- **Clarifications on the use of passive Hot-Join**
- **Changed ENTDAA CCC from Required to Conditional**

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

- Also clarified details on when a Target is required to support the ENTDAA CCC

- **Clarifications on MRL and MWL configuration:**
  - If a Target does not support a new MRL/MWL value sent via the SETMRL/SETMWL CCCs, then it ignores the new value and returns the previously configured MRL/MWL value with the next GETMRL/GETMWL CCC
- **Clarifications on the use of the SETDASA and SETAASA CCCs**, including:
  - Once a Dynamic Address is assigned to a Target with a Static Address (using SETDASA or SETAASA), the Target will not respond to the ENTDAA CCC, since it already has an assigned Dynamic Address
  - Dynamic Address can be changed or reset with subsequent use of RSTDAA or SETNEWDA CCCs
- **Clarifications on GETSTATUS indication of pending interrupt status**
- **Moved Clock-to-Data Turnaround Delay details** to ***Section 4.3.11.2 Timing Specification***
- **Clarifications on the use of Target Reset**, including:
  - Added timing details for the Target Reset Pattern
  - Added clarifications on when the Target Reset Pattern can be sent, as well as how the Controller handles a possible conflict between sending the Target Reset Pattern and receiving an IBI from a Target
  - For the configured reset action to be taken, the RSTACT CCC(s) and the Target Reset Pattern must now appear in the same SDR Frame. I.e., there can be no intervening STOP condition.
  - Added new table that shows the Required/Optional resets within the I3C Basic Target for various defined reset types
  - Whole-Target reset (i.e., the RSTACT CCC with Defining Byte 0x02) is not required
  - Whole-Target reset is not required to include a “full/chip” reset (however implementers may include this if desired)
- **New definitions for byte field format and implied time units for the RSTACT Direct Read CCC** when a Target returns the expected reset time (for Defining Bytes 0x81 thru 0x83):
  - Implementers can optionally use the same format and time units for other RSTACT reset types
  - Added clarifications of how Controllers should interpret the reset time values returned by older I3C Basic Targets (i.e., those conforming to v1.1.1 and earlier of the I3C Basic Specification), since the time units were not defined in those versions of the spec
- **Several clarifications on Error Reset and Error Recovery methods**
- **Added timing details to diagrams for the HDR Exit Pattern and the HDR Reset Pattern:**
  - Note that the definition of the HDR Reset Pattern was also moved into ***Section 6.1 HDR Modes***, since a Target is only required to support the pattern if it supports at least one of the HDR Modes

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Annex B I3C Implementation Summary Templates

I3C implementers should use the formats presented in this Annex to concisely express what I3C features their products do and do not implement. Use of these formats simplifies both identification of candidate I3C products that meet a system designer's requirements, and comparison between different I3C products.

Two formats are provided: Full Format and Concise Format. Both formats are appropriate for use in product data sheets and other technical marketing materials.

B.1 Full Format

Product Name/Number: _____

Supports I3C Bus Operating Voltage(s): □ 1.5 V □ 1.1 V

Implements I3C Optional Feature(s):

□ F001 <Name of this optional feature>

□ F002 < Name of this optional feature >

□ F003 < Name of this optional feature >

Example

Product Name/Number: MyCompanyName A3445-88GGH

Supports I3C Bus Operating Voltage(s): ✓ 1.5 V □ 1.1 V

Implements I3C Optional Feature(s):

□ F001 <Name of this optional feature>

✓ F002 < Name of this optional feature >

□ F003 < Name of this optional feature >

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

B.2 Concise Format

Product: _____

I3C Bus Operating Voltage(s): _____

Optional Feature(s): _____

Examples

Product: MyCompanyName A3445-88GGH

1.5 V

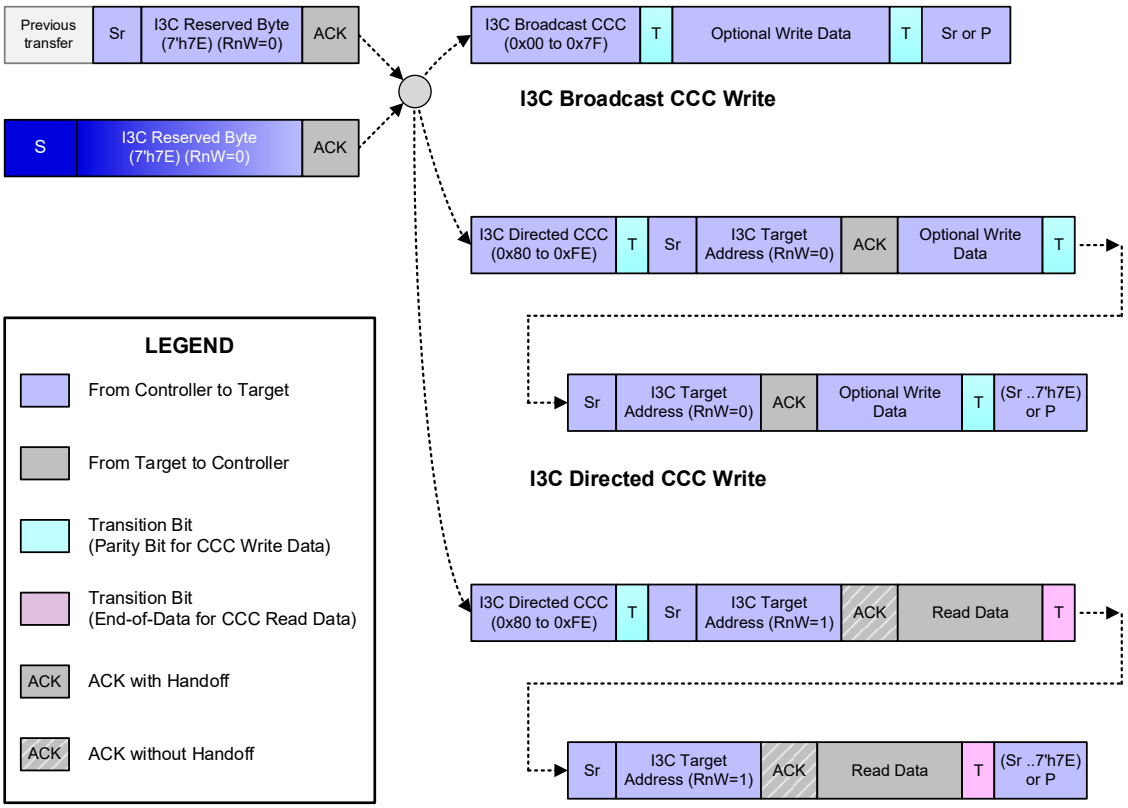
Optional Features: F003, F006, F009

MyCompanyName A3445-88GGH: I3C 1.5 V; Optional Features F003, F006, F009

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Annex C I3C Basic Communication Format Details

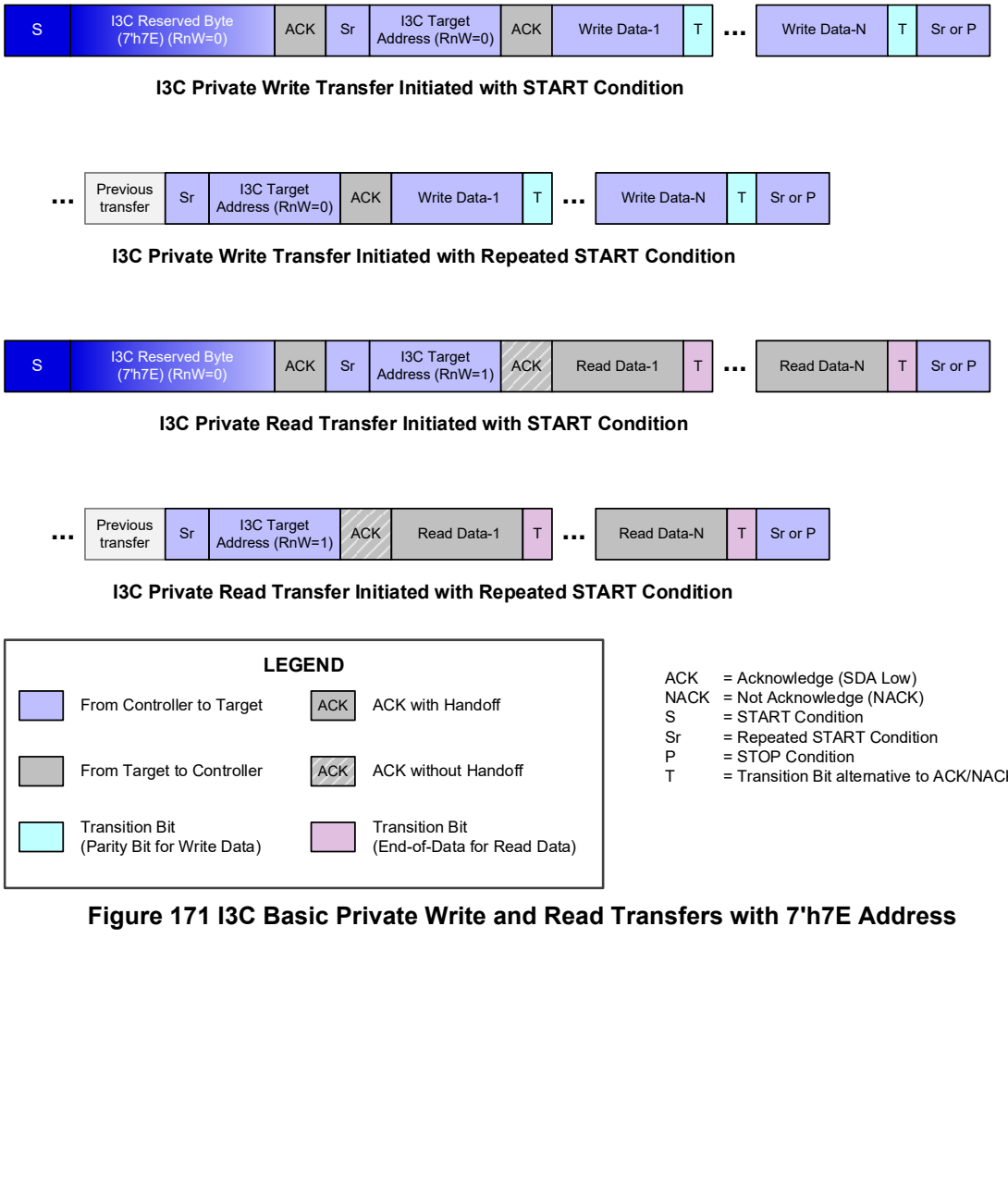
C.1 I3C Basic CCC Transfers



8217

Figure 170 I3C Basic CCC Transfers

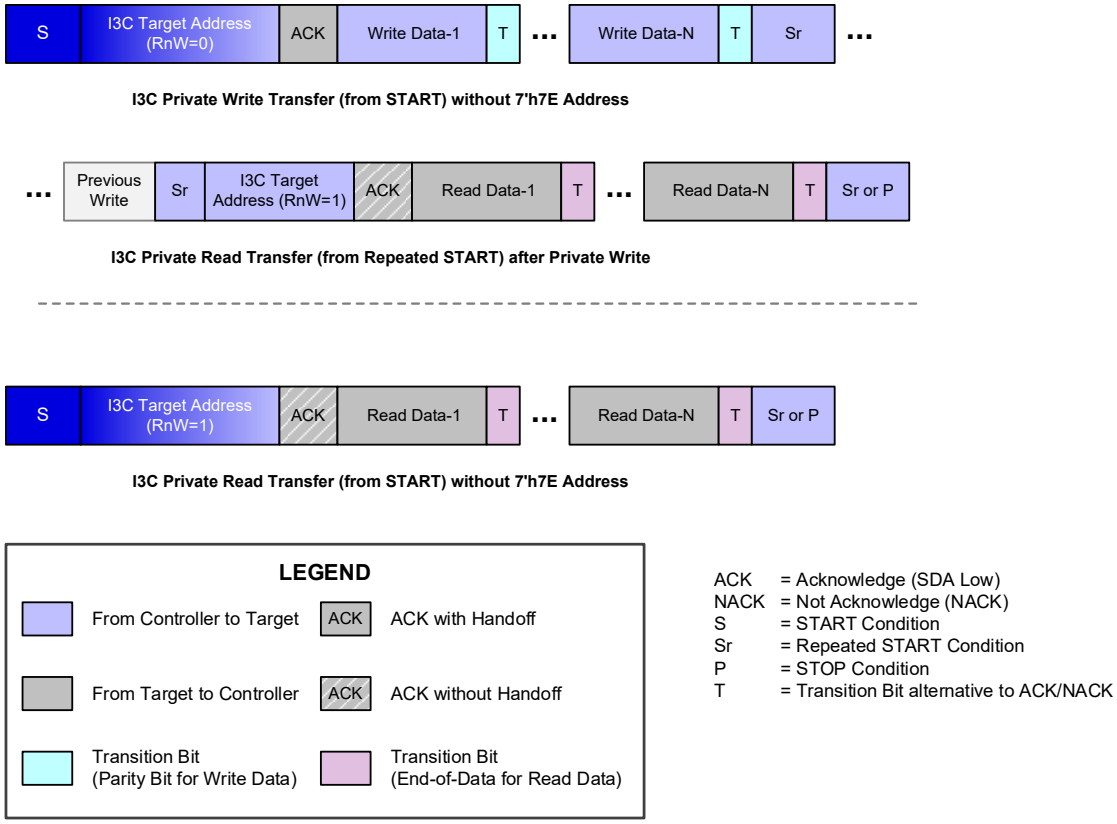
C.2 I3C Basic Private Write and Read Transfers



8218

Figure 171 I3C Basic Private Write and Read Transfers with 7'h7E Address

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



8219

Figure 172 I3C Basic Private Write and Read Transfers without 7'h7E Address

C.3 Legacy I²C Write and Read Transfers on the I3C Basic Bus



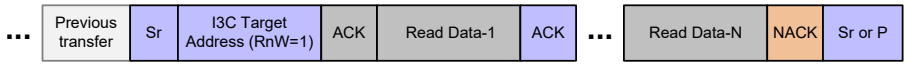
Legacy I²C Write Transfer Initiated with START Condition



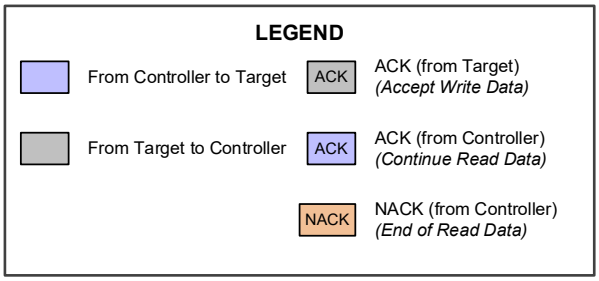
Legacy I²C Write Transfer Initiated with Repeated START Condition



Legacy I²C Read Transfer Initiated with START Condition



Legacy I²C Read Transfer Initiated with Repeated START Condition

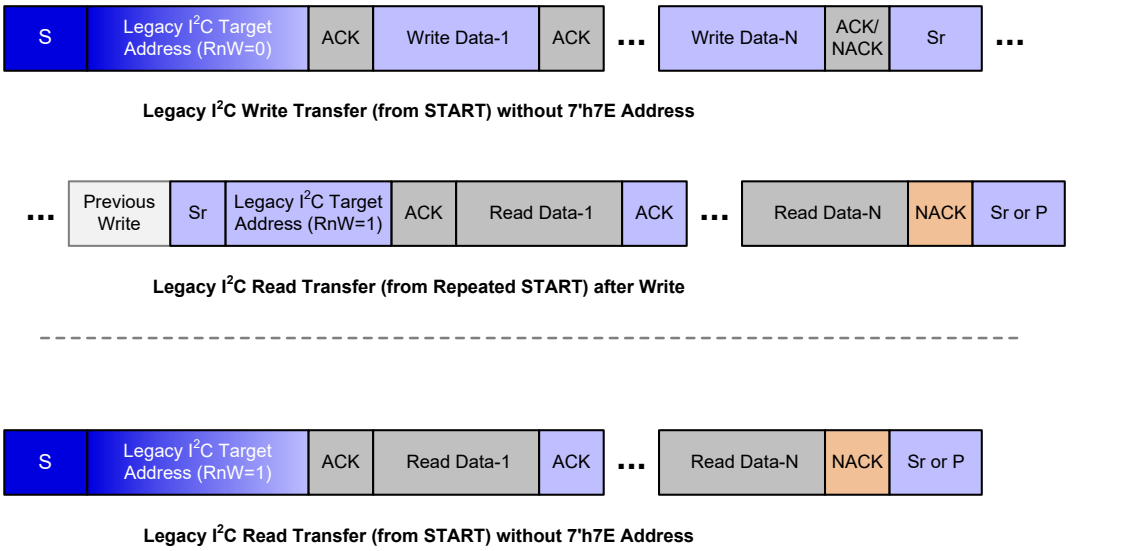


ACK = Acknowledge (SDA Low)
NACK = Not Acknowledge (NACK)
S = START Condition
Sr = Repeated START Condition
P = STOP Condition

8220

Figure 173 Legacy I²C Write and Read Transfers in I3C Basic Bus with 7'h7E Address

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

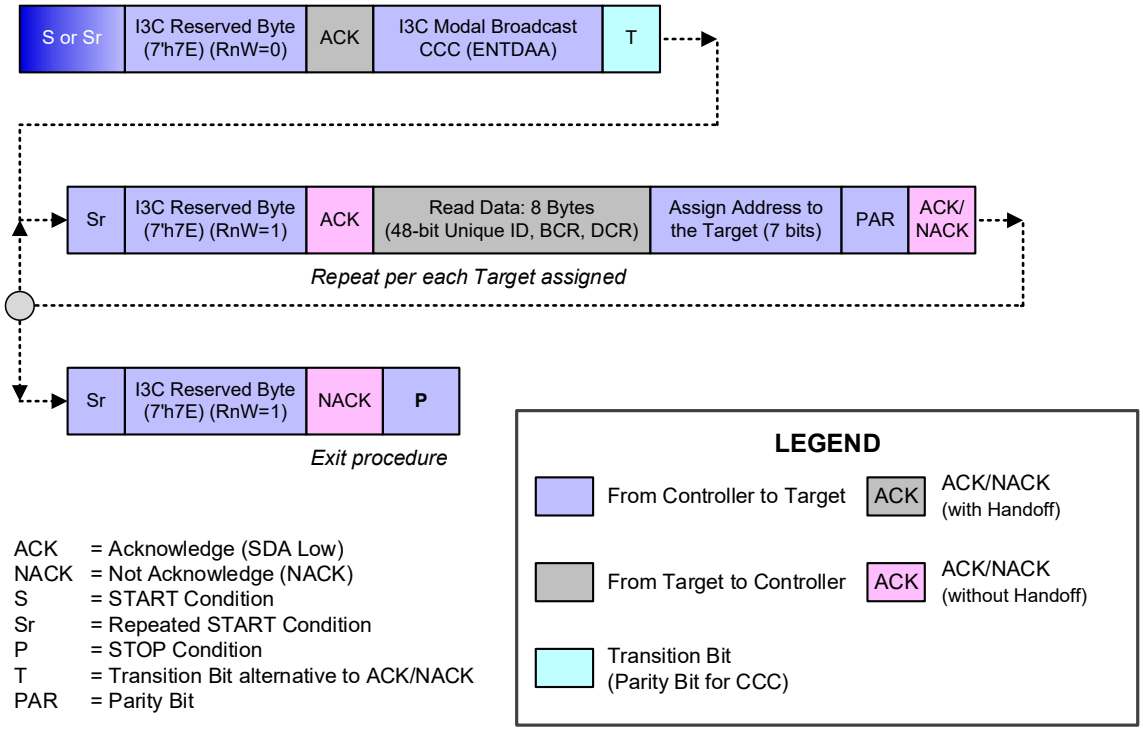


8221

Figure 174 Legacy I²C Write and Read Transfers in I3C Basic Bus without 7'h7E Address

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

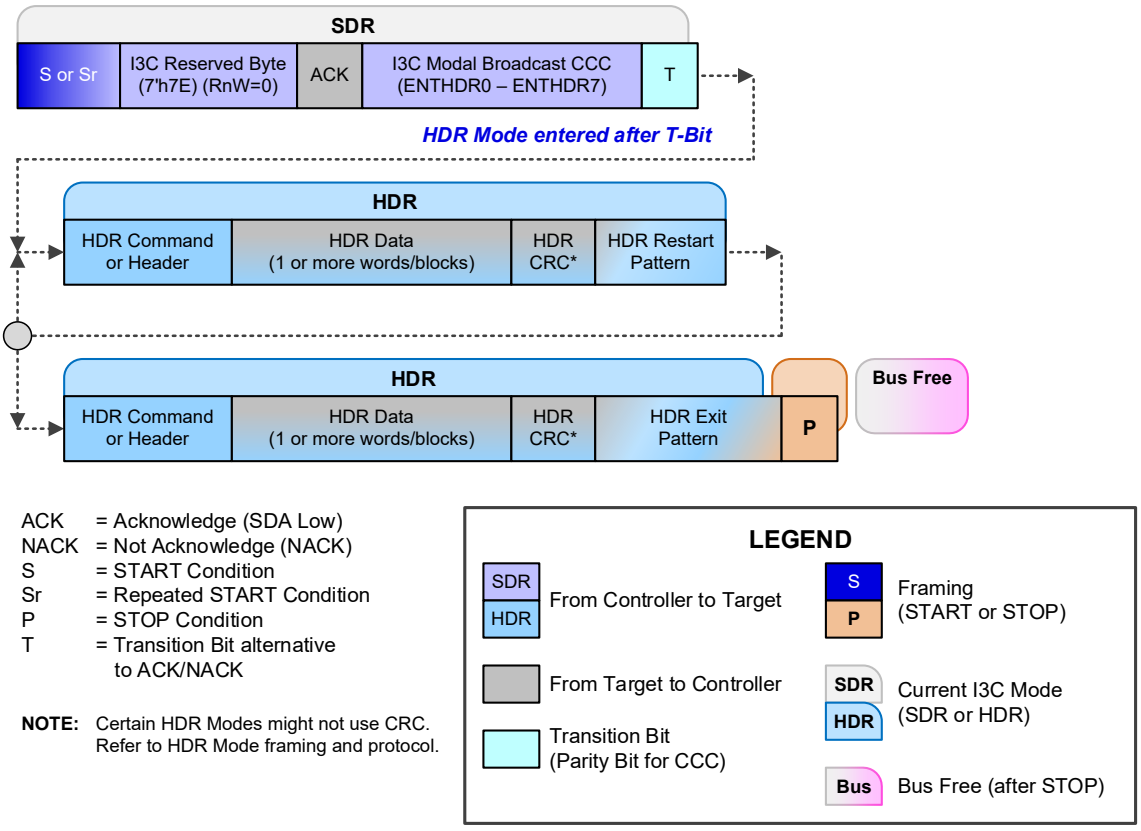
C.4 Dynamic Address and Enter HDR



8222

Figure 175 Dynamic Address Allocation ENTDAA CCC Bus Modal

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



8223

Figure 176 Enter HDR Mode CCC Bus Modal

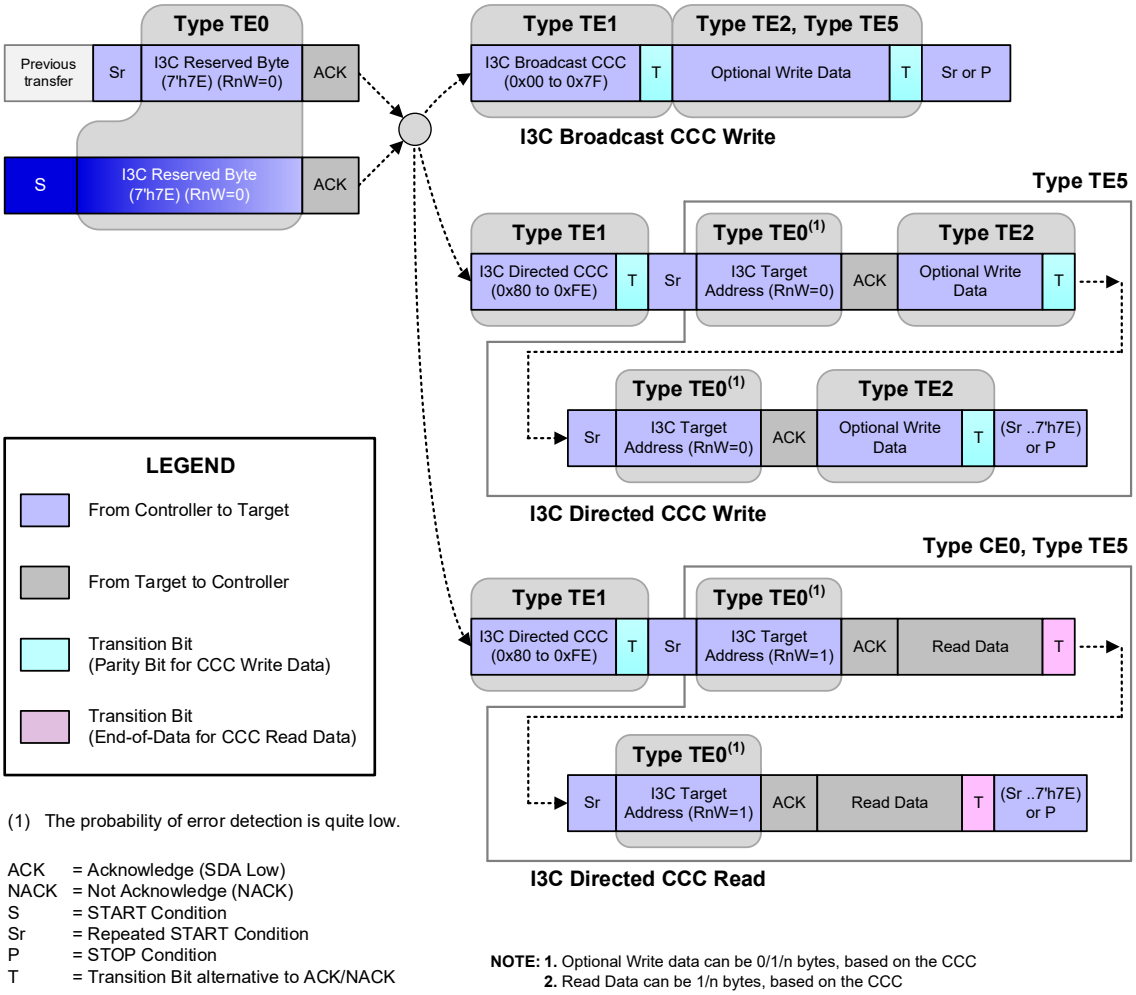
Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Annex D SDR Mode Error Type Origins

D.1 Error Types in I3C Basic CCC Transfers



(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

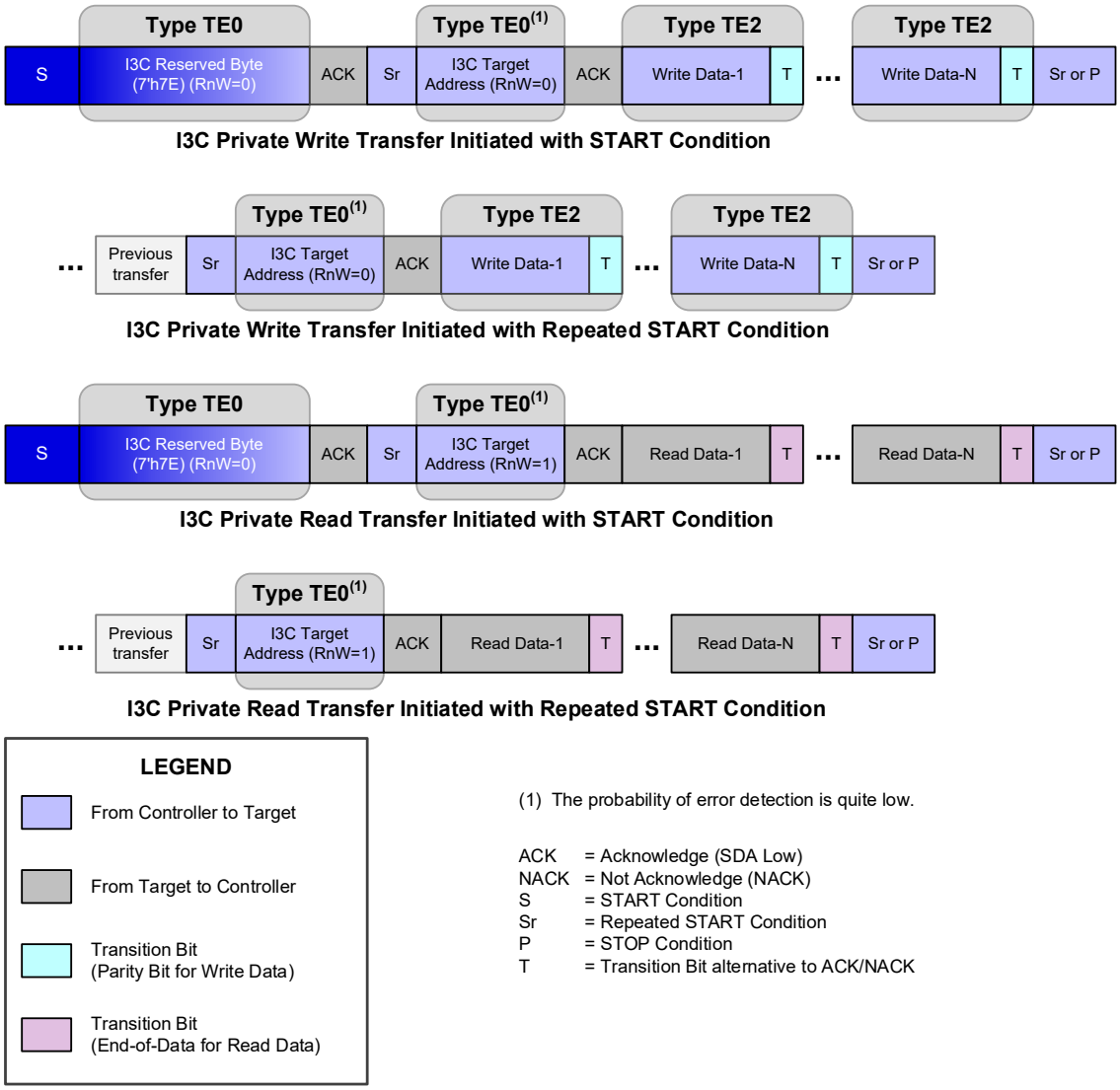
NOTE: 1. Optional Write data can be 0/1/n bytes, based on the CCC
2. Read Data can be 1/n bytes, based on the CCC

8224

Figure 177 Error Type Origins: I3C Basic CCC Transfers

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

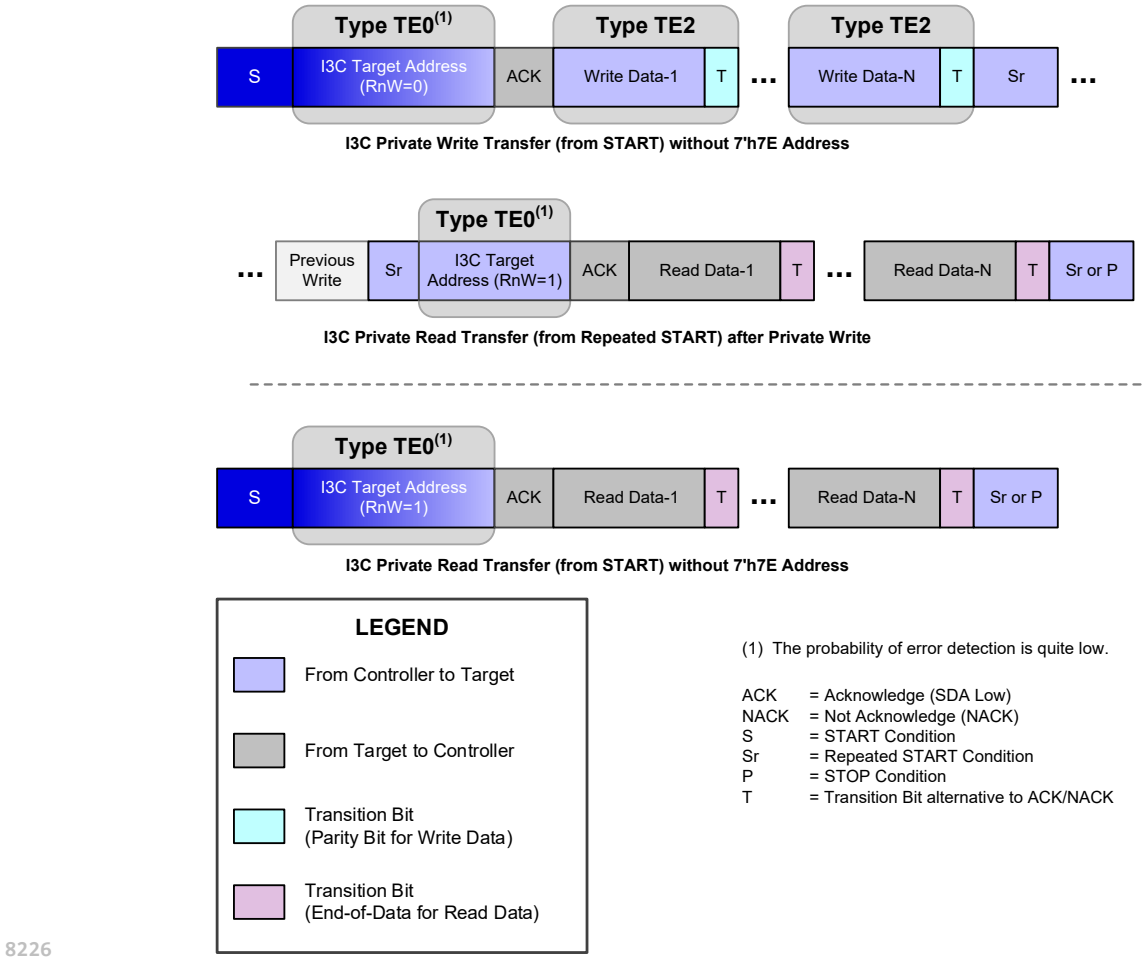
D.2 Error Types in I3C Basic Private Read and Write Transfers



8225

Figure 178 Error Type Origins: I3C Basic Private Write & Read Transfers with 7'h7E Address

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

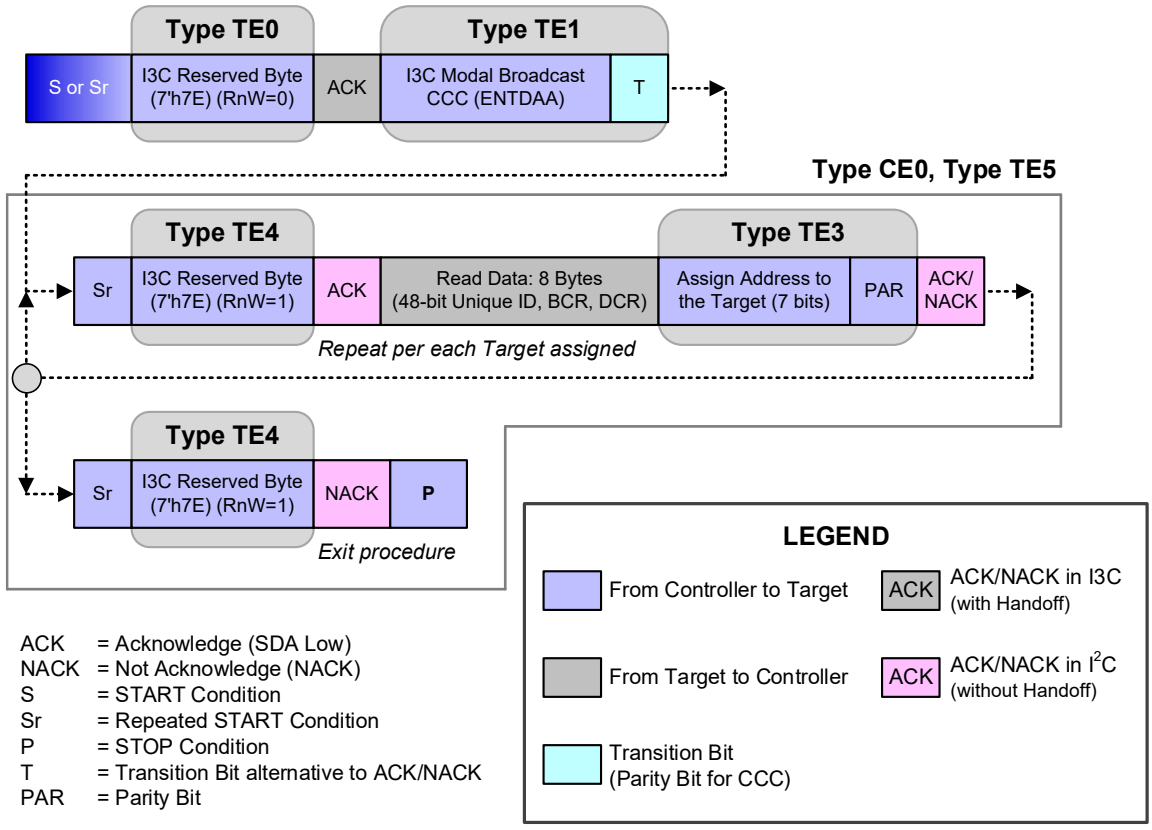


8226

Figure 179 Error Type Origins: I3C Basic Private Write & Read Transfers without 7'h7E Address

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

D.3 Error Types in Dynamic Address Arbitration



8227

Figure 180 Error Type Origins: Dynamic Address Arbitration

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Annex E I3C Basic Controller FSMs

Figure 181 shows the key transmission Modes and their entry, resume, and exit sequences. It includes the SDR transmission states, and differentiates these from other capabilities and Modes. It captures the states at which Arbitration happens as well.

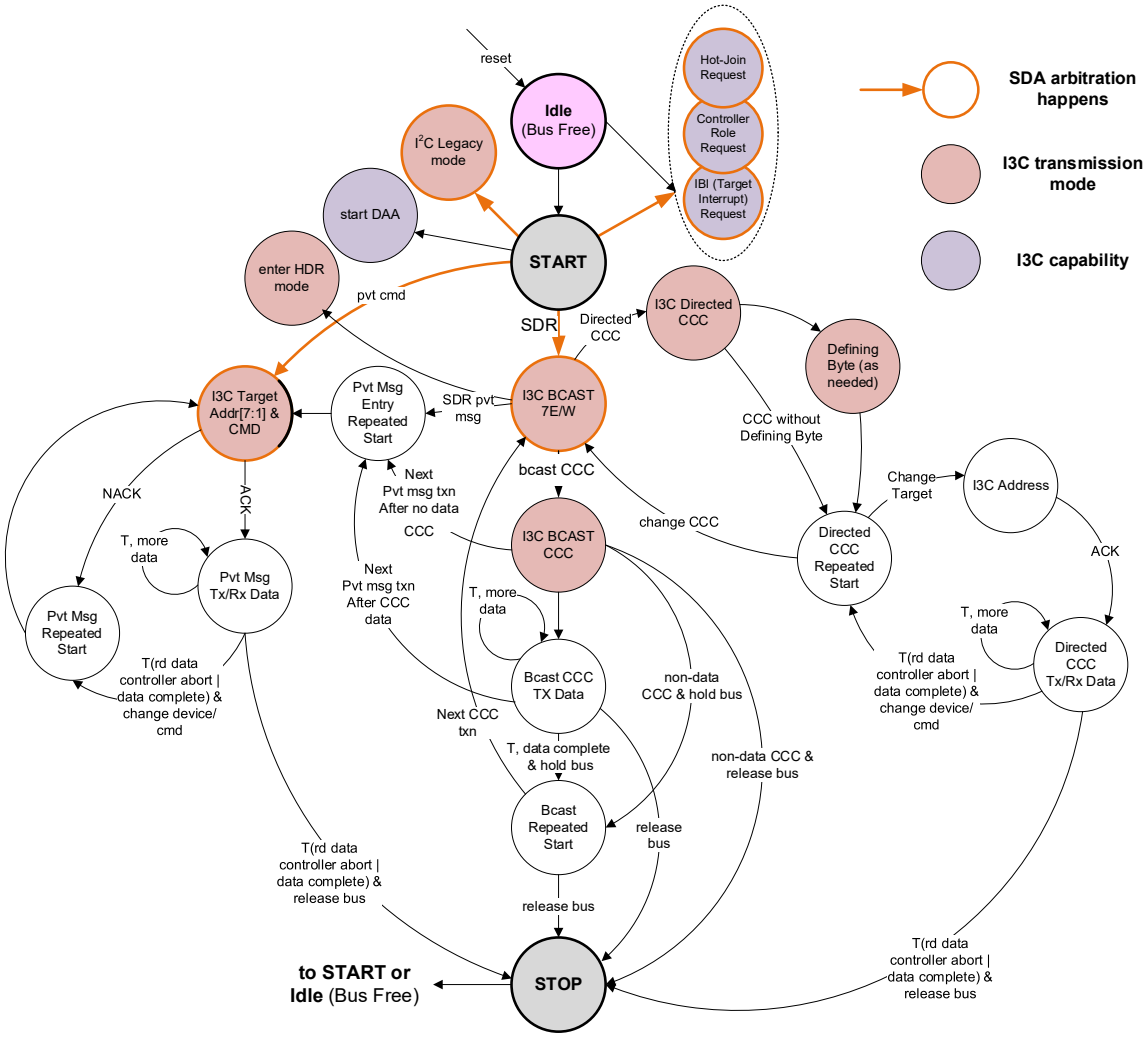
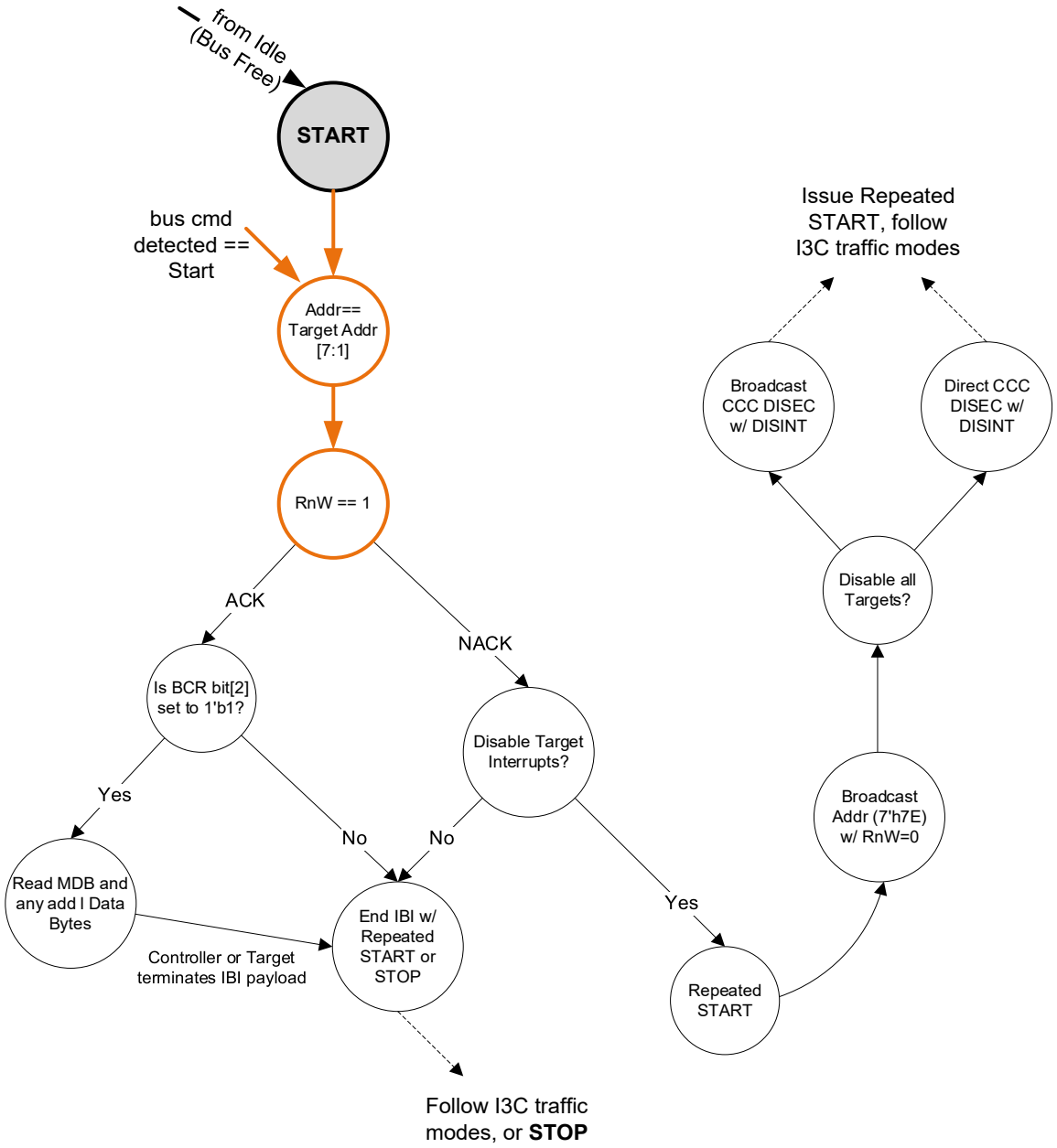


Figure 181 I3C Basic Primary Controller FSM

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

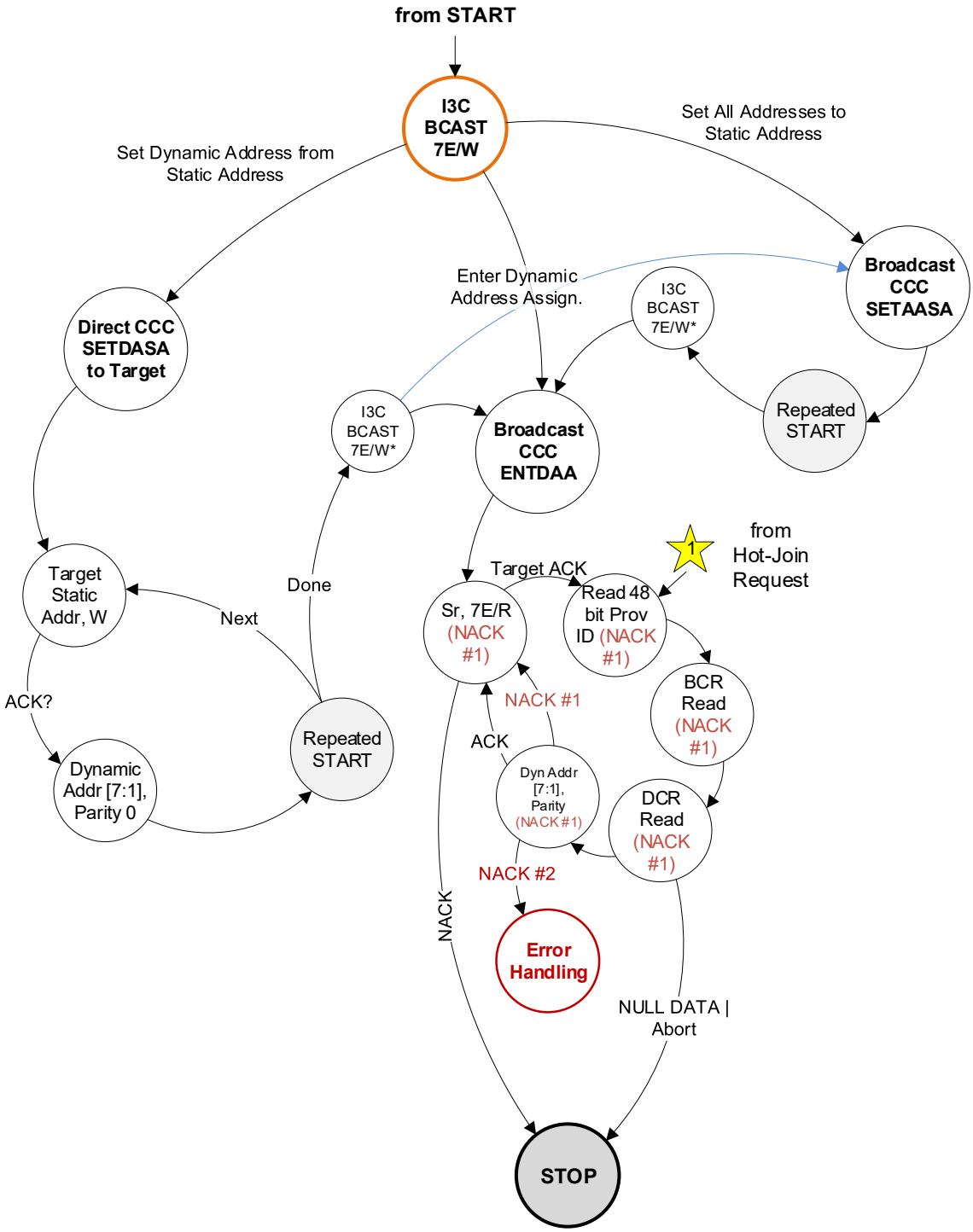
8232 ***Figure 182*** through ***Figure 186*** represent I3C Basic features including In-Band Interrupts, Secondary
8233 Controller, Dynamic Address Assignment, and Hot-Join.



8234

Figure 182 In-Band Interrupt (Target Interrupt) Request FSM

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



8235

Figure 183 Dynamic Address Assignment FSM

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

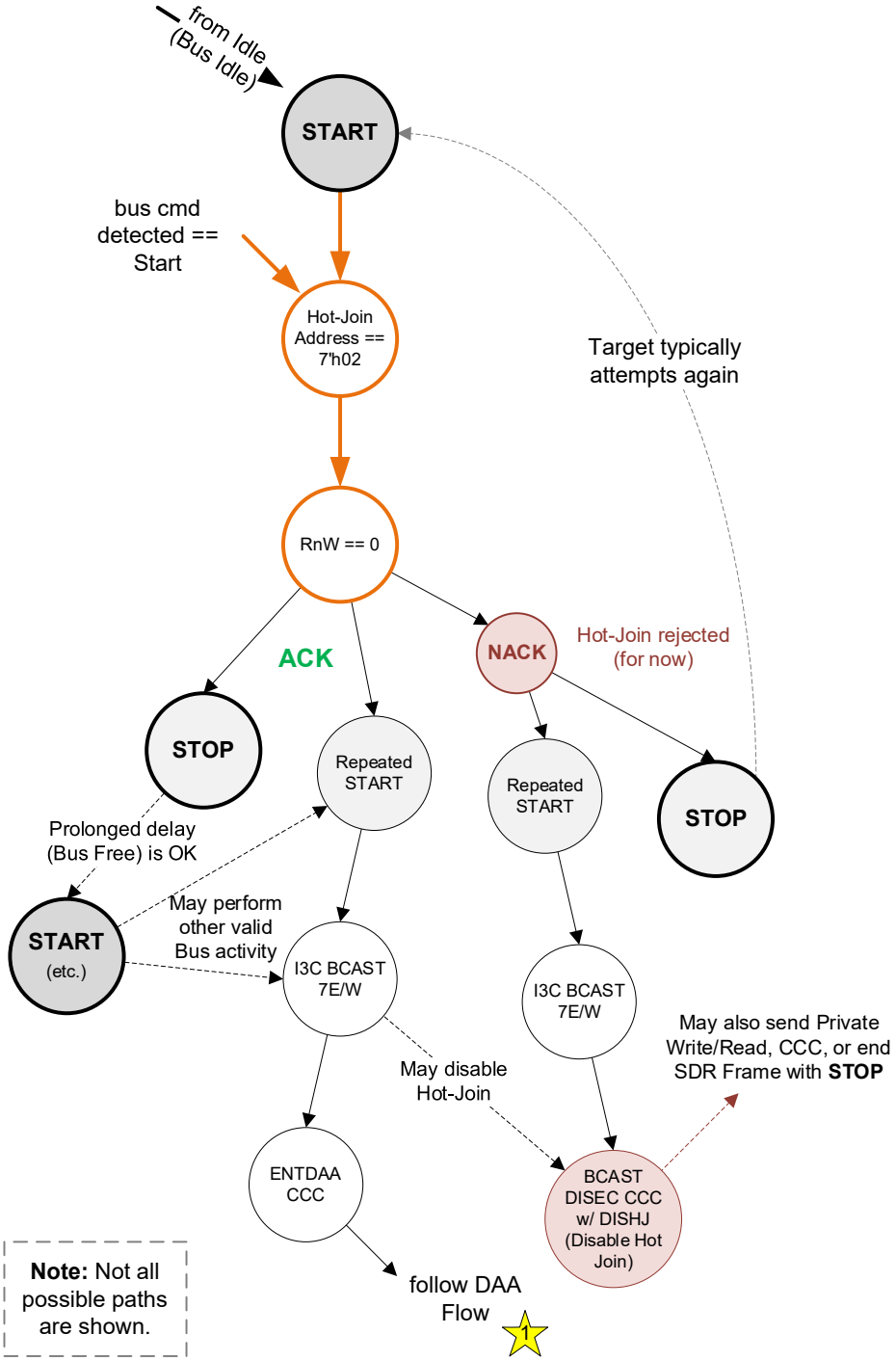
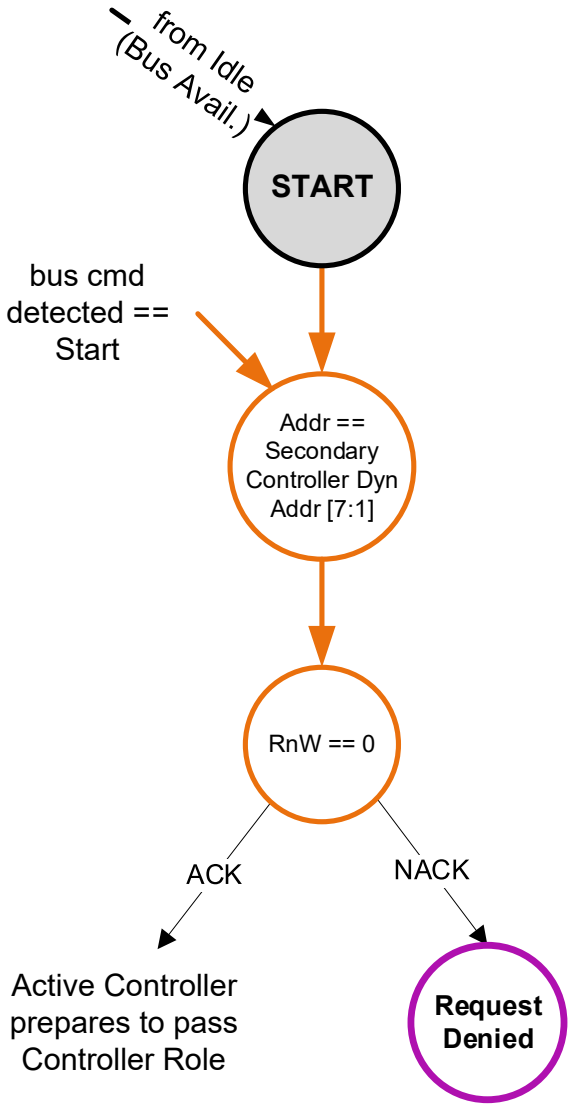


Figure 184 Hot-Join FSM

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition



8237

Figure 185 Controller Role Request FSM

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

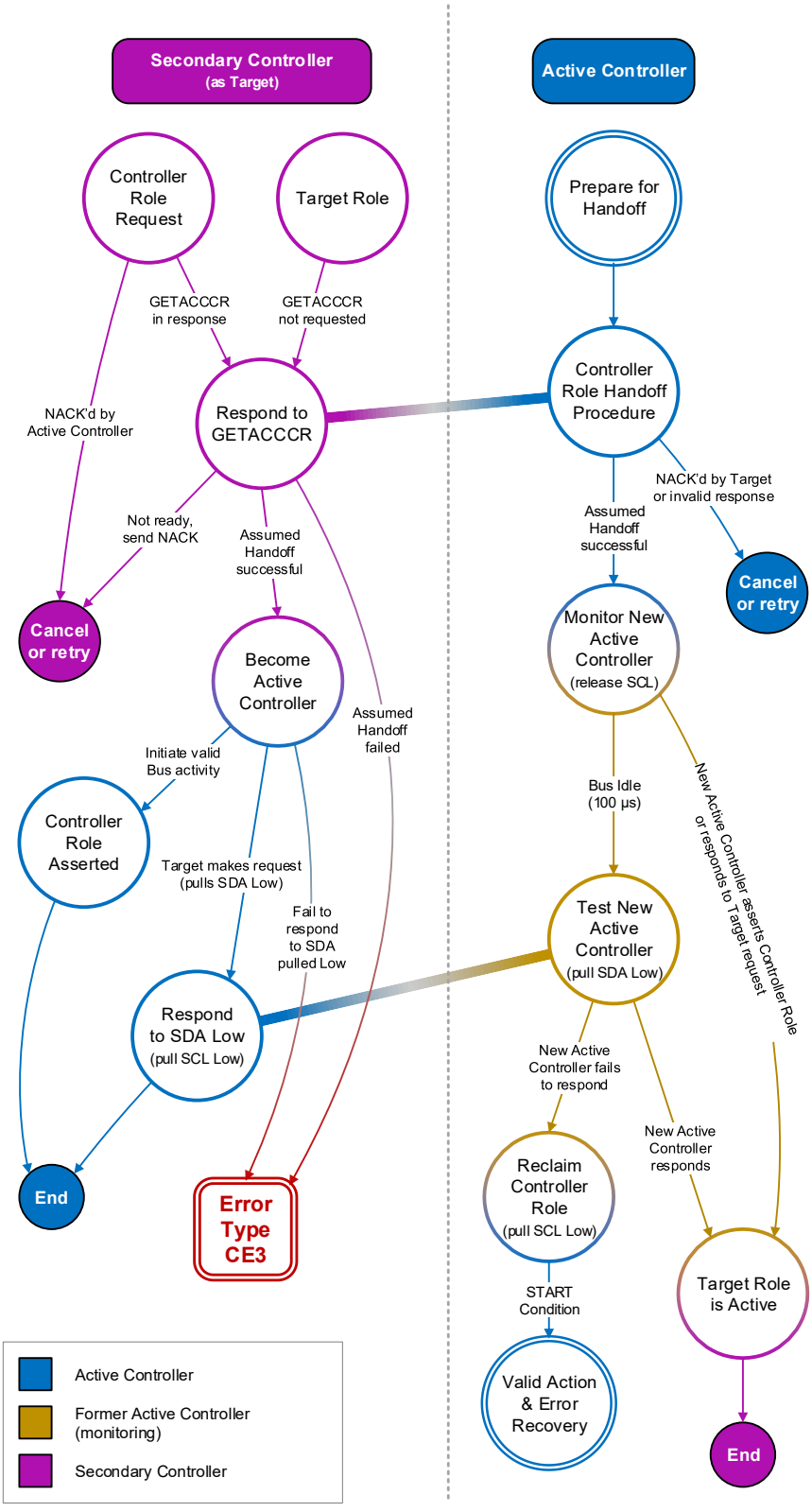


Figure 186 Controller Regaining Bus Ownership FSM

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

The FSMs in ***Figure 187*** represent the Controller states when in HDR transmission Modes, including Entry, Restart, and Exit sequence for HDR-DDR Mode.

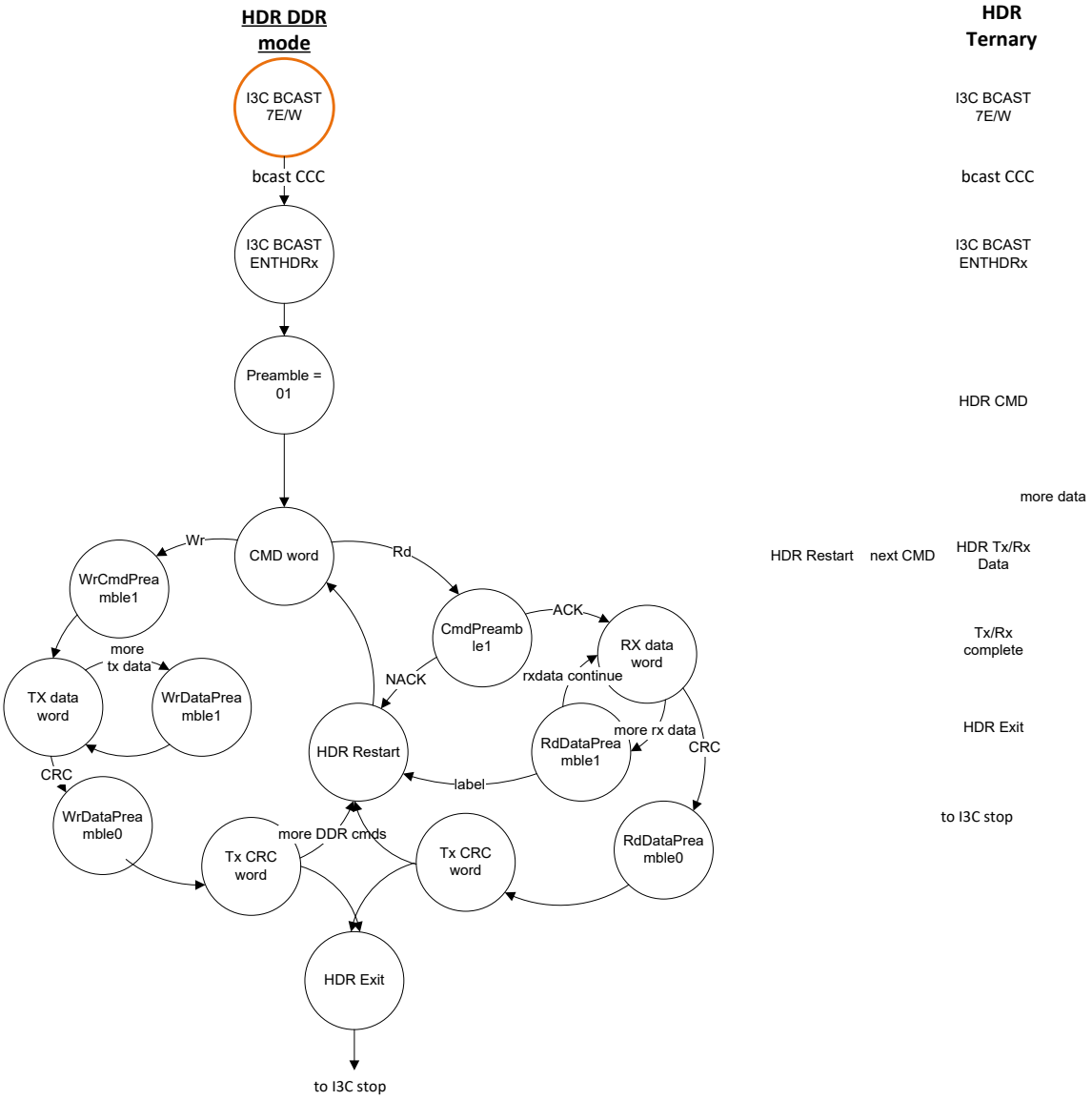
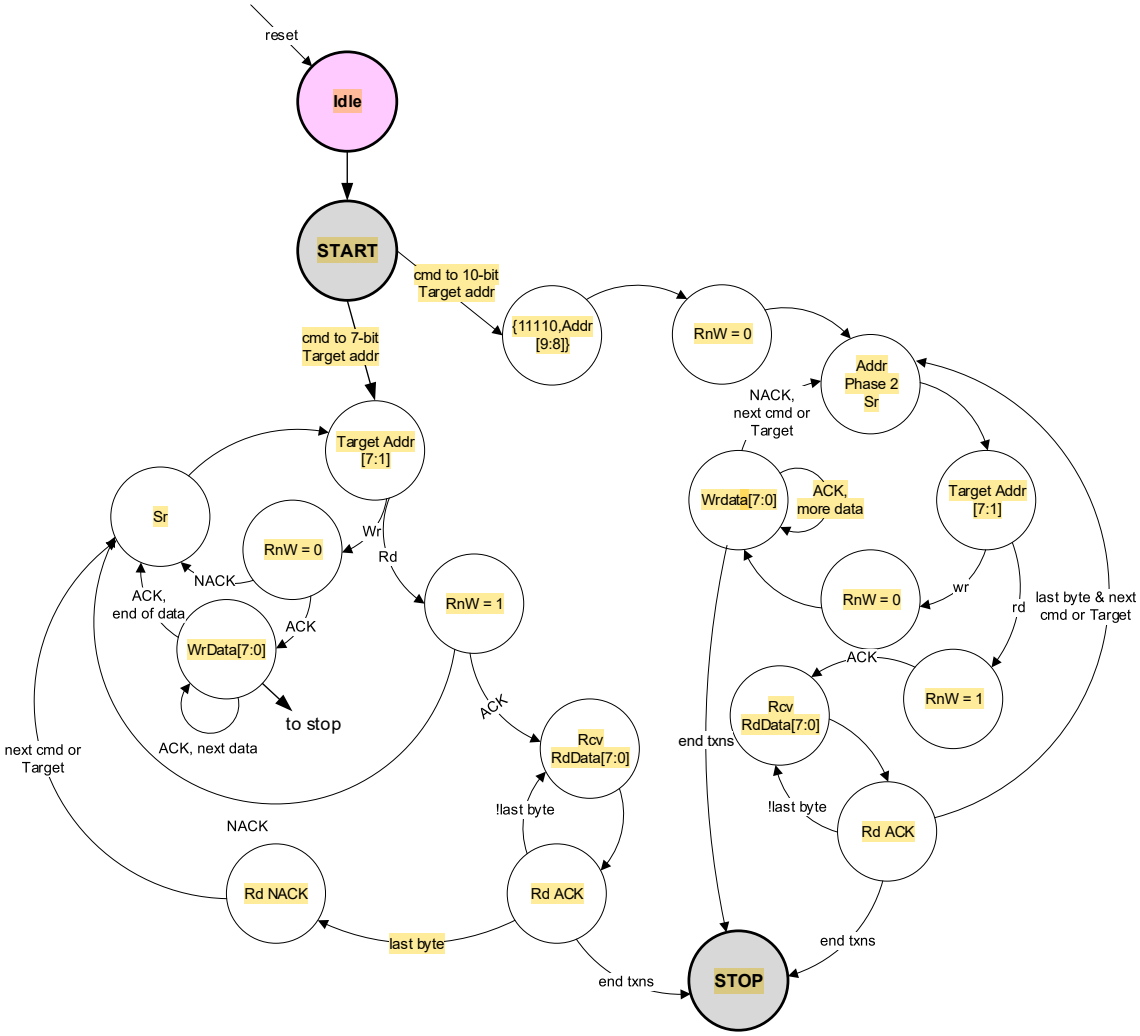


Figure 187 HDR-DDR Mode FSM

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

8242 ***Figure 188*** is for reference only.



8243

Figure 188 I²C Legacy Controller FSM

See Errata 01, Item 39

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Annex F Typical I3C Basic Protocol Communications

Figure 189 through ***Figure 193*** illustrate a typical communication for each of the supported I3C Basic Protocols. While these figures do not exhaustively illustrate all possible I3C Basic communications, they do serve as useful introductions to the signaling and transmission formatting used in each I3C Basic Protocol.

F.1 Typical SDR Private Read

Figure 189 illustrates example communication using I3C Single Data Rate (SDR) Mode. It shows the Controller reading a byte of data from the Target at Address 0x2B in SDR Mode.

From the Bus Free Condition, the Controller 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 Controller turns on a Pull-Up resistor and goes to Open Drain. All Targets ACK by pulling the SDA line Low (in the Figure, pink fill means the Target is in control of the SDA line at this time). The Controller then issues a Repeated START, then the Address of the Target (0x2B) it wants to read followed by RnW (1 for Read). The Controller then turns on a Pull-Up resistor and goes to Open Drain, allowing the Target to acknowledge by pulling the SDA line Low. At this point, the Controller continues to toggle the SCL line and release the SDA line, allowing the Target to drive SDA to send one byte of data (0x4A) followed by 'T'. T = 1 informs the Controller that there is additional data, whereas T = 0 signals the end. Here there is additional data, so the Target drives SDA High until SCL goes High, at which time it releases SDA. The Controller has the option of holding SDA High with a weak Pull-Up, which signals to the Target that the Controller allows another byte to be transmitted, or to pull SDA Low (while SCL is High – hence a Repeated START), which would signal to the Target that the Controller has terminated the Read and is taking over.

SDR Mode is backwards compatible with Legacy I^{2}C Devices, because the High time of an SCL pulse is always less than 50 ns and therefore SCL will always appear to be Low because of the I^{2}C 50 ns Spike Filter.

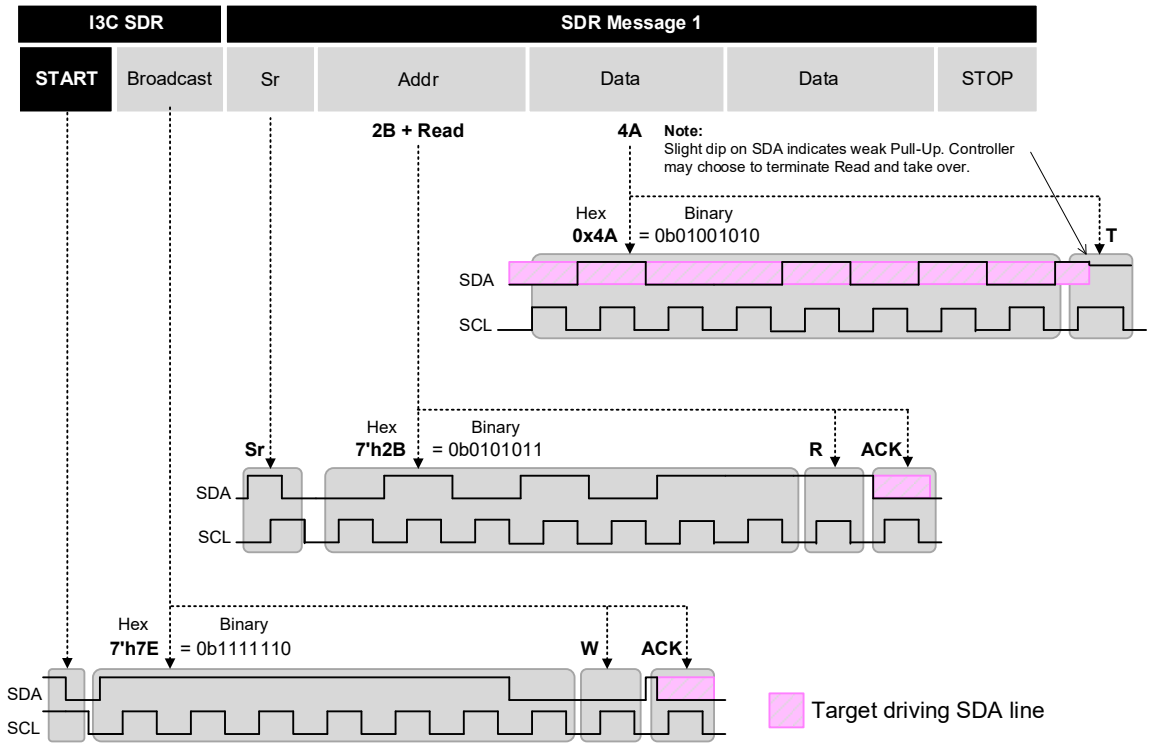


Figure 189 Example Communication Using I3C Basic SDR Mode with Private Read

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

F.2 Typical Direct CCC in SDR Mode

Figure 190 shows the Controller issuing a Direct CCC to a single Target. This particular command (GETPID) reads the Provisioned ID of a Target.

From the Bus Free Condition, the Controller 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 Controller turns on a Pull-Up resistor and goes to Open Drain. All Targets ACK by pulling SDA Low (in the Figure, pink fill means the Targets are in control of SDA at this time). The Controller then issues the Direct Common Command Code for GETPID (0x8D) followed by parity bit 'T' (odd parity = 1 for 0x8D) then the 7-bit Dynamic Address of the Target (chosen arbitrarily here to be 0x2B) followed by a RnW bit (1 for Read). Then the Controller turns on a Pull-Up resistor and goes to Open Drain, allowing the Target at Address 0x2B to ACK by pulling SDA Low, which tells the Controller that the Target Acknowledges the command and will comply. (Alternatively, the Target may NACK by not pulling SDA Low, which would inform the Controller that the Target will not comply – in this case, that an error occurred.) Following the ACK the Target outputs its 48-bit PID one byte at a time, and then the Controller issues a Repeated START (this part of the waveform sequence is not shown in the Figure).

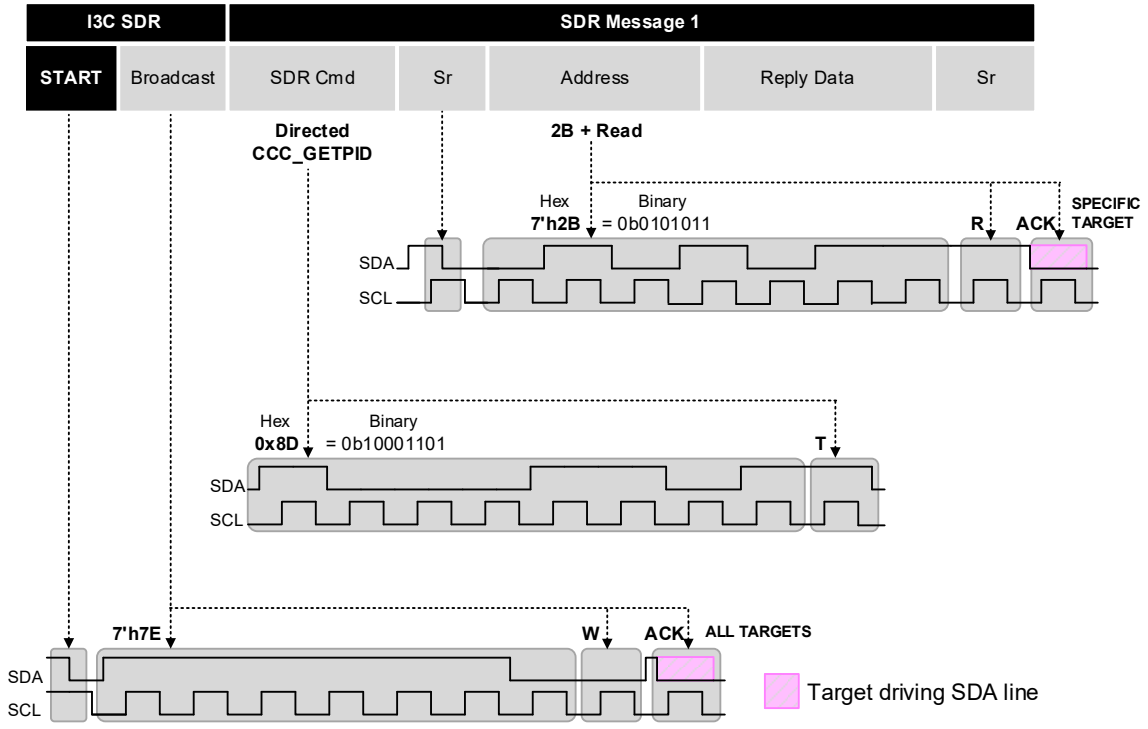


Figure 190 Example Communication Using I3C Basic SDR Mode with Direct CCC

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

F.3 Typical Broadcast CCC in SDR Mode

Figure 191 illustrates example SDR communication with a Broadcast CCC. The command used in this example sets the Maximum Read Length of all Targets to 43 bytes (0x002B).

From the Bus Free Condition, the Controller 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 Controller turns on a Pull-Up resistor and goes to Open Drain. All Targets ACK by pulling SDA Low (in the Figure, pink fill means the Targets are in control of SDA at this time). The Controller then issues the Broadcast Common Command Code for SETMRL (0x0A) followed by parity bit 'T' (odd parity = 1 for 0x0A), and then 2 data bytes (MSB first, and MSb first within each byte) to define the maximum number of bytes that can be read from a Target in a single read operation. Each data byte is followed by a 'T' bit (parity bit – odd parity). After this the Controller issues a Repeated START.

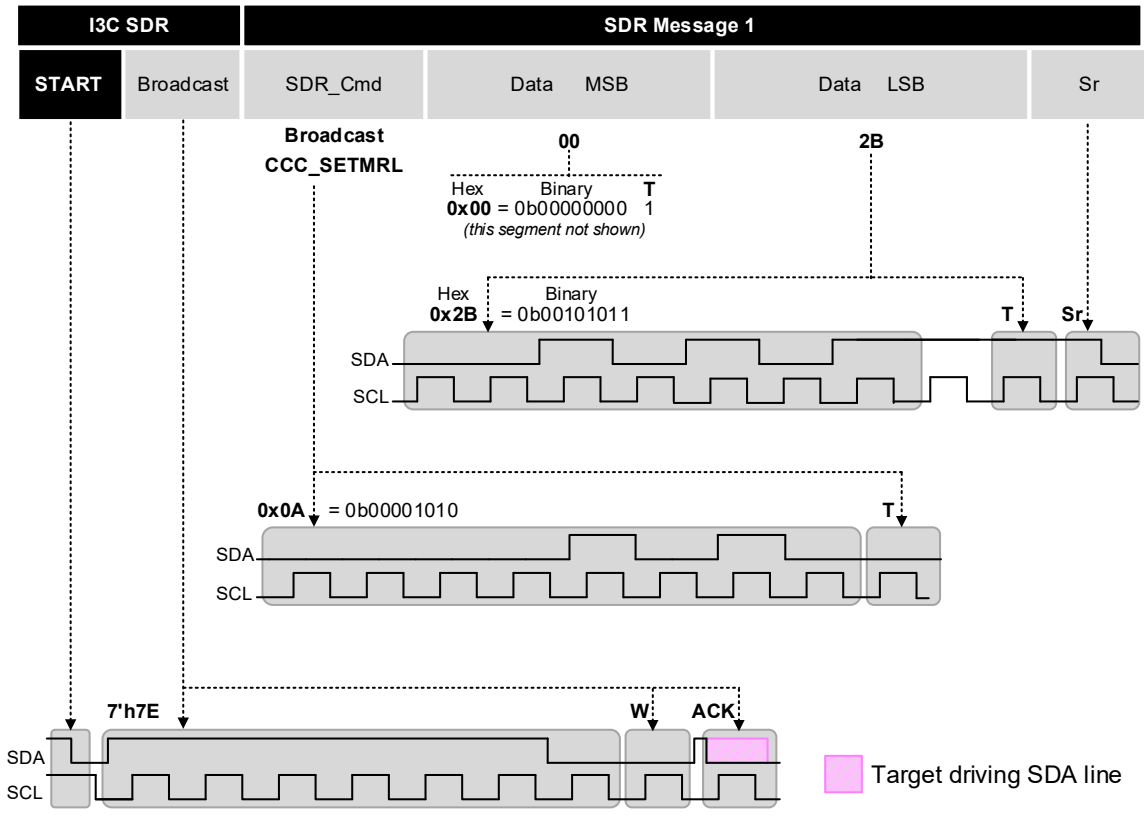


Figure 191 Example Communication Using I3C Basic SDR Mode with Broadcast CCC

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

F.4 Typical HDR-DDR Read

Figure 192 illustrates use of the HDR-DDR (Double Data Rate) Mode. It shows how the Controller can change the Mode from SDR (Single Data Rate) Mode to HDR-DDR Mode, and a sample of the HDR-DDR data format.

From the Bus Free Condition, the Controller issues a START by driving the SDA line Low while keeping the SCL line High. The Controller then issues the Broadcast Address (7'h7E) followed by RnW (0 for Write). Then the Controller turns on a Pull-Up resistor and goes to Open Drain. All Targets 'ACK' by pulling SDA Low (in the Figure, pink fill means the Targets are in control of SDA at this time). The Controller then issues the Broadcast Common Command Code for ENTHDR0 (0x20), followed by parity bit 'T' (odd parity = 0 for 0x20). At this point, the Bus is in HDR-DDR Mode. In the HDR-DDR protocol, the SDA line is sampled on every SCL edge (both Low-to-High and High-to-Low transitions of SCL). The HDR-DDR Word consists of a 2-bit Preamble, followed by two bytes of data, followed by two parity bits. The waveform for the 5-bit CRC and following traffic is not shown in the Figure.

HDR-DDR Mode is backwards-compatible with Legacy I^{2}C Devices, because the High time of an SCL pulse is always less than 50 ns and as a result SCL will always appear to be Low because of the I^{2}C 50 ns Spike Filter.

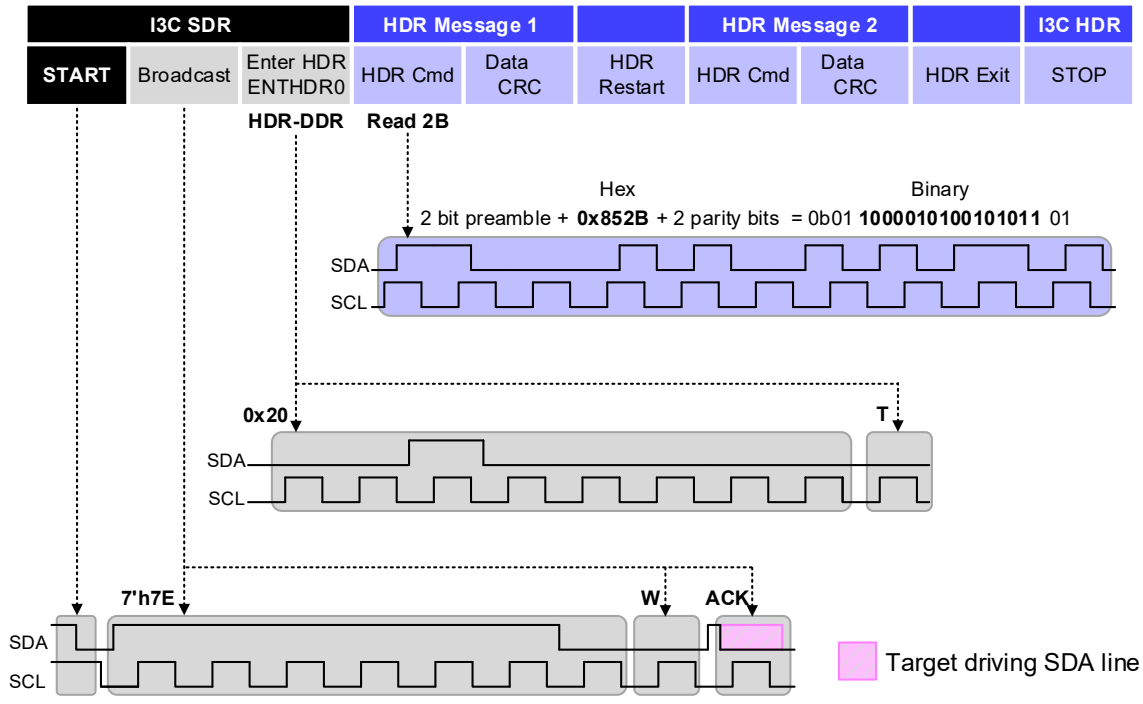


Figure 192 Example Communication Using HDR-DDR Protocol

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

F.5 Typical HDR-BT Read

Figure 193 illustrates use of the HDR-BT (Bulk Transport) Mode. It shows how the Controller can change the Mode from SDR (Single Data Rate) Mode to HDR-BT Mode, and a sample of the HDR-BT data format using 1-lane configuration (i.e., Coding 0). Note that other ML lane configurations and Codings are possible.

From the Bus Free Condition, the Controller issues a START by driving the SDA line Low while keeping the SCL line High. The Controller then issues the Broadcast Address (7'h7E) followed by RnW (0 for Write). Then the Controller turns on a Pull-Up resistor and goes to Open Drain. All Targets 'ACK' by pulling SDA Low (in the Figure, pink fill means the Targets are in control of SDA at this time). The Controller then issues the Broadcast Common Command Code for ENTHDR3 (0x23), followed by parity bit 'T' (odd parity = 0 for 0x23). At this point, the Bus is in HDR-BT Mode. In the HDR-BT protocol, the SDA line is sampled on every SCL edge (both Low-to-High and High-to-Low transitions of SCL). Note that this example shows a Read, so the Controller sets up the SDA line appropriately such that the first bit (ADDR[0]) is sampled correctly for a Read transfer (i.e., RnW = 1'b1).

In this example, the Controller starts the transfer by sending the HDR-BT Header Block in 1-lane form, and the Controller drives the SDA line to send the Address and RnW bit, to indicate the addressed Target (0x2B) as well as the direction of transfer; the Cmd0 byte (0x85) and additional Cmd1–Cmd3 bytes, to indicate which data to read; and the additional Control byte for transfer settings. The Controller then uses the Transition byte to set up SDA for the Target to accept, and the Target drives SDA Low to indicate acceptance. The Target then drives the SDA line (and optionally the SCL line, if supported) for one or more HDR-BT Data Blocks and the HDR-BT CRC Block that follows, with up to 32 bytes per Data Block. (The waveforms for the Data Blocks and CRC Block are not shown in the Figure; see ***Section 6.4.3.6*** for reference.)

HDR-BT Mode is backwards compatible with Legacy I^{2}C Devices, because the High time of an SCL pulse is always less than 50 ns and as a result SCL will always appear to be Low because of the I^{2}C 50 ns Spike Filter.

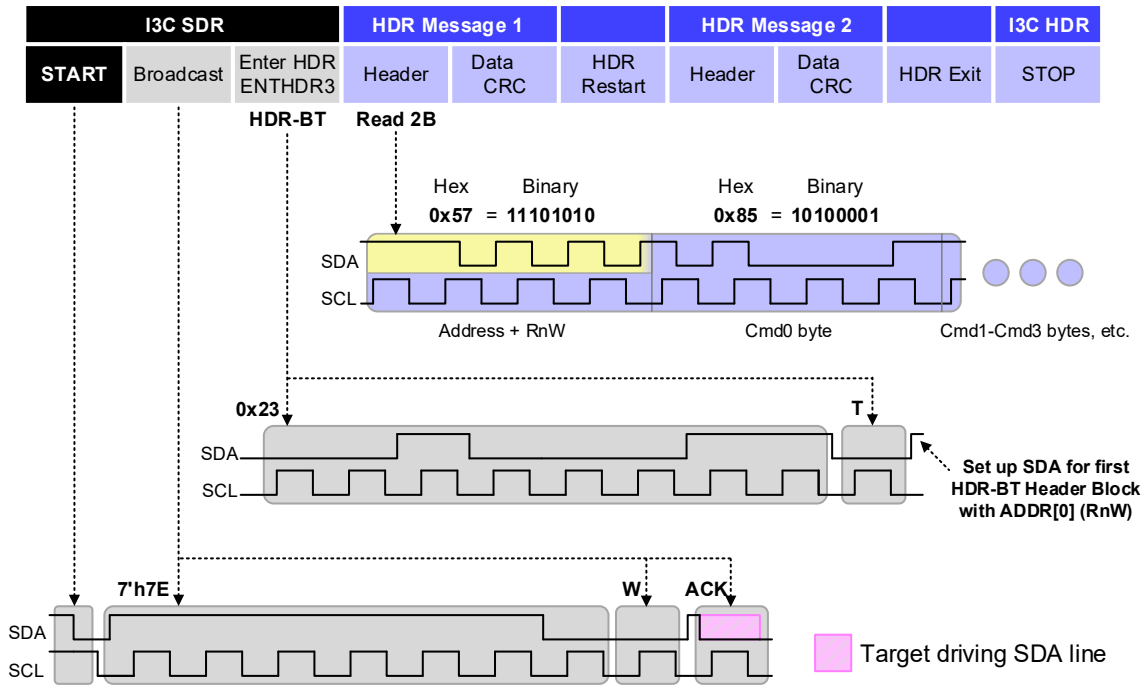


Figure 193 Example Communication Using HDR-BT Protocol

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Annex G MIPI I3C Basic Specification Supplemental Patent Licensing Terms

This agreement (the "Agreement") between MIPI Alliance Inc. ("MIPI") and each MIPI member or other party that has manifest agreement to these terms (each a "Licensor" and collectively the "Licensors") is effective as of the date the I3C Basic Specification (defined below) is first approved by the MIPI Board (the "Effective Date"). Capitalized terms used in this Agreement that are not expressly defined here have the meaning identified in the MIPI Membership Agreement or MIPI Bylaws, as applicable. For convenience, key definitions are reproduced in Attachment A. For the avoidance of doubt: (a) in connection with this Agreement, any reference to a "MIPI Specification" means the MIPI I3C Basic Specification as described in Section 2, and (b) any rights or obligations created under this Agreement are independent of any rights or obligations created under the MIPI Membership Agreement, Bylaws or any other agreements, and nothing in this Agreement is intended to alter rights or obligations established elsewhere.

1. Background. Typically, MIPI specifications are implemented only by MIPI members. MIPI members make certain intellectual property licensing commitments to other members under the MIPI Membership Agreement, with different rules applying to "Mobile Terminals" and "Accessories," in contrast to other types of implementations. The MIPI Board intends to make the MIPI I3C Basic Specification available for implementation by parties who are non-members of MIPI, however. Further, both MIPI members and non-members may use this specification inside and outside of Mobile Terminals or Accessories. The Licensors contributed to the development of the MIPI I3C Basic Specification, and desire to see it widely used. MIPI and the Licensors believe that making licenses available (as set forth in this Agreement) to both member and non-member implementers, for all types of implementations, will facilitate widespread adoption of the MIPI I3C Basic Specification, to the benefit of MIPI, the Licensors, and the broader community that MIPI serves.

2. MIPI I3C Basic Specification. "MIPI I3C Basic Specification" means the specification titled "I3C Basic 1.0" as approved by the MIPI Board, and all subsequent versions of such specification approved by the MIPI Board after the Effective Date. Any party implementing the MIPI I3C Basic Specification, whether or not a MIPI member, is an "Implementer." For the avoidance of doubt: the MIPI I3C Basic Specification is distinct from the MIPI I3C Specification version 1.0 approved by the MIPI Board on Dec. 31, 2016. The terms of this Agreement apply exclusively to the MIPI I3C Basic Specification.

3. License commitment.

a. RAND-Z license obligation. For the MIPI I3C Basic Specification only, Licensor hereby agrees to grant, and to cause its Affiliates to grant, to any requesting Implementer a worldwide, non-exclusive, non-sublicensable license under the Necessary Claims of Licensor or its Affiliates, with zero royalties or other compensation, under terms and conditions that are reasonable and nondiscriminatory, to make, have made, use, import, offer to sell, lease, sell, promote and otherwise distribute Compliant Portions. Licensor shall not be obligated to license any part or function of a product in which a Compliant Portion is incorporated that is not itself a Compliant Portion.

b. Reciprocity; defensive suspension. Licensor shall not be obligated to license any Implementer if that Implementer does not agree to make patent licenses available under any Necessary Claims of that Implementer and its Affiliates to Licensors and all other Implementers under terms substantially identical to the terms described in this Agreement. Further, a Licensor may suspend any license granted under this Agreement to any Implementer if that Implementer or its Affiliate initiates against any party litigation that alleges infringement of a Necessary Claim of Implementer or its Affiliate in connection with the MIPI I3C Basic Specification. Additionally, subject to Section 3.1(f) of the MIPI Membership Agreement as between MIPI Members, a Licensor may terminate, ab initio, any license granted pursuant to this Agreement to any Implementer that initiates litigation against the Licensor alleging infringement of any patent claim of the Implementer or its Affiliate. For the purposes of this Agreement, a party that files a suit which is defensive based on a patent infringement claim or suit by another party will not be deemed to have initiated litigation.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

c. Circumvention or transfer. Licensor agrees that it has not transferred and will not transfer any patent having Necessary Claims solely for the purpose of circumventing the obligations described in this Agreement. In addition, Licensor agrees that any transfers by Licensor to a third party of a patent having relevant Necessary Claims, whether or not recognized as Necessary Claims at the time of transfer, shall be subject to (i) the terms and conditions of this Agreement, and (ii) the agreement that the third party shall grant licenses under said Necessary Claims to Implementers pursuant to the terms of this Agreement in like manner and to the same extent as the third party would be required to do if it had executed this Agreement. A transfer of ownership in a business entity which owns or has the right to license a patent having Necessary Claims shall be considered a transfer of such patent.

4. Termination of obligation to license future versions. Each Licensor may terminate its participation in this Agreement at any time by providing written notice to the MIPI Managing Director; termination will be effective 30 days after such notice is actually received, subject to the survival points below. Promptly upon receipt of such notice, the MIPI Managing Director will alert the MIPI Board and will use reasonable efforts to notify all Licensors of such termination. After termination, the agreement to grant a license as provided in Section 3 shall survive in full force and effect only: (a) for versions of the MIPI I3C Basic Specification which the Board had approved before the effective date of termination; (b) for Necessary Claims relating to any version of the MIPI I3C Basic Specification approved after the effective date of termination that are used in a substantially similar manner and to a substantially similar extent with a substantially similar result as the Necessary Claims that were used in a prior version for which the Licensor is obligated to grant licenses under this Agreement; and (c) for those Necessary Claims that directly result from inclusion of Licensor-provided material in the draft version of the specification that existed immediately prior to Licensor's withdrawal. Termination of this Agreement by one Licensor does not impact the Agreement among MIPI or the other Licensors, not does it not impact Licensor's MIPI Membership Agreement, which will remain in full force and effect.

5. Third party beneficiaries. All Implementers, whether or not they are MIPI members, are intended third party beneficiaries of this Agreement.

6. Counterparts; additional Licensors. This Agreement may be signed in any number of counterparts. If additional parties manifest agreement to terms substantially identical to this Agreement, those parties will be deemed Licensors under this Agreement.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

8405 ## 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).

8406

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).

8407

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.

8408

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.

8409

"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.

8410

"Board" means the Board of Directors of MIPI.

8411

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Annex H I3C Basic Development Companies

Advanced Micro Devices, Inc.
Analogix Semiconductor, Inc.
Binho LLC
Cadence Design Systems, Inc.
Google LLC
HiSilicon Technologies Co. Ltd.
Intel Corporation
Introspect Technology
KIOXIA Corporation
LX Semicon Co., Ltd.
Marvell Asia Pte, Ltd.
MediaTek Inc.
Micron Technology, Inc.
Microsoft Corporation
Mixel, Inc.
NVIDIA
OmniVision Technologies, Inc.
Prodigy Technovations Pvt. Ltd.
Realtek Semiconductor Corp.
Robert Bosch GmbH
Siemens Industry Software, Inc.
SK Hynix
SmartDV Technologies India Private Limited
Soliton Technologies Private Limited
STMicroelectronics
Synaptics
Synopsys, Inc.
Texas Instruments Incorporated
Western Digital Technologies Inc.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Participants

The list below includes those persons who participated in the Working Group that developed this Specification and who consented to appear on this list.

Eugen Becker, Robert Bosch GmbH
David Bennett, Google LLC
Rajesh Bhaskar, Micron Technology, Inc.
Chris Browy, Siemens Industry Software, Inc.
Enrico Carrieri, Intel Corporation
Brijesh Chandrala, Prodigy Technovations Pvt. Ltd.
Geraud Cheenne, STMicroelectronics
B. Jack Chen, Realtek Semiconductor Corp.
Eddie Chen, Siemens Industry Software, Inc.
Huimin Chen, Intel Corporation
Dragos Davidescu, STMicroelectronics
Siqing Du, HiSilicon Technologies Co. Ltd.
Praveen Kumar Durairaj, SmartDV Technologies India Private Limited
Mahmoud El-Banna, Mixel, Inc.
Heng Fan, OmniVision Technologies, Inc.
Todd Farrell, Microsoft Corporation
Kenneth Foust, Intel Corporation
John Geldman, KIOXIA Corporation
Jonathan Georgino, Binho LLC
Steve Glaser, NVIDIA
Rajasekar Gopalsamy, Intel Corporation
Arunkumar Govindharaj, SmartDV Technologies India Private Limited
Chris Grigg, MIPI Alliance Inc. (Team)
Mohamed Hafed, Introspect Technology
Hyosun Joo, SK Hynix
Janusz Jurski, Intel Corporation
Raghu Krishnamurthy, NVIDIA
Lalit Kumar, Cadence Design Systems, Inc.
David Landsman, Western Digital Technologies Inc.
Seungmee Lee, LX Semicon Co., Ltd.
Tung-Sheng Lin, MediaTek Inc.
Johnson Liu, Synaptics
Leilk Liu, MediaTek Inc.
Myron Loewen, SK Hynix
Chetan Maheshwari, Intel Corporation
Jose Mantovani, Google LLC
Tim McKee, Intel Corporation
Rama Krishna Meda, Synopsys, Inc.
Saravana Kumar Muthusamy, Soliton Technologies Private Limited
Pratap Neelasheety, Synopsys, Inc.
Dina Osama, Mixel, Inc.
David Papazian, STMicroelectronics
Michele Scarlatella, MIPI Alliance Inc. (Team)
Matthew Schnoor, Intel Corporation
Anthony Seely, Texas Instruments Incorporated
Goutham Seshadri Kumar, Marvell Asia Pte, Ltd.
Jason Sher, Siemens Industry Software, Inc.
Pradeep Singh, Intel Corporation
Ming So, Advanced Micro Devices, Inc.
Dong Hyun Song, SK Hynix
Licinio Sousa, Synopsys, Inc.
Amit Srivastava, Intel Corporation
Przemyslaw Sroka, Cadence Design Systems, Inc.
Greg Stewart, Analogix Semiconductor, Inc.
Paul Suhler, KIOXIA Corporation
Eyuel Zewdu Teferi, STMicroelectronics
Jack Trump, HiSilicon Technologies Co. Ltd.
Suresh Venkatachalam, Synopsys, Inc.
Baskaran Venkatesan, SmartDV Technologies India Private Limited
Liu Yan, HiSilicon Technologies Co. Ltd.
Martin Cavallo, Binho LLC
Mo Akbari, Texas Instruments Incorporated
Prashant Rupapara, Samsung Electronics, Co.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Previous contributors to v1.1.1:

Amit Ashara, Texas Instruments Incorporated
Rajesh Bhaskar, Intel Corporation
Daniel Bogard, Cirrus Logic
Anamitra Chakrabarti, Synopsys, Inc.
Brijesh Chandrala, Prodigy Technovations Pvt. Ltd.
Geraud Cheenne, STMicroelectronics
Kenneth Foust, Intel Corporation
Chris Grigg, MIPI Alliance (Team)
Janusz Jurski, Intel Corporation
Paul Kimelman, NXP Semiconductors
Yu Liu, Robert Bosch GmbH
Davide Magnoni, STMicroelectronics
Tim McKee, Intel Corporation
Takashi Miyamoto, Sony Corporation
Aruni Nelson, Intel Corporation
Gajendra Patro, Tektronix, Inc.
Radu Pitigoi-Aron, Qualcomm Incorporated
Michele Scarlatella, STMicroelectronics
Matthew Schnoor, Intel Corporation
Satwant Singh, Lattice Semiconductor Corp.
Amit Srivastava, Intel Corporation
Vasudev Uttharahalli, Prodigy Technovations Pvt. Ltd.
Suresh Venkatachalam, Synopsys, Inc.
James Wang, InvenSense, Inc.
Cheng Yu, Robert Bosch GmbH
Tadaaki Yuba, Sony Corporation
Eyuel Zewdu Teferi, STMicroelectronics

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition

Previous contributors to v1.0:

Yossi Amon, Qualcomm Incorporated
Miguel Barasch, Qualcomm Incorporated
Rajesh Bhaskar, Intel Corporation
Geraud Cheenne, STMicroelectronics
Sriraman Dakshinamurthy, InvenSense, Inc.
Kenneth Foust, Intel Corporation
Chris Grigg, MIPI Alliance (Team)
Anubhav Gupta, Intersil Corporation
Hsiehchang Ho, MediaTek Inc.
Doug Hoffman, Qualcomm Incorporated
Toshihisa Hyakuda, Sony Corporation
Paul Kimelman, NXP Semiconductors
Michael Laisne, Dialog Semiconductor
Alessandro Mecchia, STMicroelectronics
Pratap Neelasheety, Synopsys, Inc.
Alex Passi, Cadence Design Systems, Inc.
Venkata Suresh Perumalla, NVIDIA
Radu Pitigoi-Aron, Qualcomm Incorporated
Duane Quiet, Intel Corporation
James Rippie, MIPI Alliance (Team)
Jim Ross, Skyworks Solutions, Inc.
Volker Rzehak, Texas Instruments Incorporated
Brad Sharpe-Geisler, Lattice Semiconductor Corp.
Satwant Singh, Lattice Semiconductor Corp.
Dong Hyun Song, SK Hynix
Amit Srivastava, Intel Corporation
Przemyslaw Sroka, Cadence Design Systems, Inc.
Tatsuya Sugioka, Sony Corporation
Hiroo Takahashi, Sony Corporation
Asmah Truky, Intel Corporation
Suresh Venkatachalam, Synopsys, Inc.
Girish Venkatraman, Intel Corporation
Niel Warren, Knowles Electronics
Rick Wietfeldt, Qualcomm Incorporated
Tim White, IDT

Copyright © 2016–2025 MIPI Alliance, Inc.

Public Release Edition

This page intentionally left blank.

Copyright © 2016–2025 MIPI Alliance, Inc.
Public Release Edition