Frequently Asked Questions (FAQ) for MIPI I3C® v1.1.1 & MIPI I3C Basic℠ v1.1.1

FAQ Version 1.1

22 August 2022

MIPI Board Approved 22 August 2022

Public Release Edition

This is an informative document, not a MIPI Specification.

Various rights and obligations that apply solely to MIPI Specifications (as defined in the MIPI Membership Agreement and MIPI Bylaws) including, but not limited to, patent license rights and obligations, do not apply to this document.

This document is subject to further editorial and technical development.

NOTICE OF DISCLAIMER

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

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

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

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

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

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

Special Note Concerning MIPI I3C and MIPI I3C Basic

As described in the I3C Basic specification, certain parties have agreed to grant additional rights to I3C Basic implementers, beyond those rights granted under the MIPI Membership Agreement or MIPI Bylaws. Contribution to or other participation in the development of this FAQ document does not create any implication that a party has agreed to grant any additional rights in connection with I3C Basic. Consistent with the statements above, nothing in or about this FAQ document alters any party's rights or obligations associated with I3C or I3C Basic.

Contents

Release History

Date	Version	Description
04-Sep-2021	FAQ v1.0	Initial Board Approved release.
22-Aug-2022	FAQ v1.1	Board Approved release.

This page intentionally left blank.

1 Introduction

This FAQ has been developed to introduce the MIPI I3C ***[MIPI01][MIPI11][MIPI13]*** and I3C Basic ***[MIPI09][MIPI14]*** specifications to developers and users. The I3C WG has compiled these frequently asked questions (FAQs) to assist Member implementation activity. Some areas also include clarification when an area of the specification was ambiguous, and this FAQ will show the intended resolution of the ambiguity.

For I3C v1.1.1 ***[MIPI13]*** and I3C Basic v1.1.1 ***[MIPI14]***, new FAQs have been added, and the existing FAQs have been updated as needed. Some FAQs show historical information for context, i.e., from older versions of I3C or I3C Basic.

Note:

> ***For the full MIPI I3C Specification, the most current version is I3C v1.1.1.*** *FAQ entries reflect all updates, both technical and editorial (i.e., the changes from I3C v1.0 or v1.1 to v1.1.1).*
>
> ***For the MIPI I3C Basic Specification, the most current version is I3C Basic v1.1.1.*** *FAQ entries reflect all updates, both technical and editorial (i.e., the changes from I3C Basic v1.0 to v1.1.1). Note that* ***there is no I3C Basic v1.0****; this version number was skipped.*

Throughout this FAQ document, unless otherwise noted, the terms 'MIPI I3C' and 'I3C' refer to both MIPI I3C ***[MIPI01][MIPI11][MIPI13]*** and MIPI I3C Basic ***[MIPI09][MIPI14]***, unless specified otherwise.

Note:

> *Unless otherwise noted, a reference to a numbered section of the MIPI I3C specification applies to both the I3C Specification and the I3C Basic Specification (i.e., section numbers are aligned across the two specifications).*
>
> *None of the answers in this FAQ are intended to overwrite or overrule the information in either the I3C Specification* ***[MIPI01][MIPI11][MIPI13]*** *or the I3C Basic Specification* ***[MIPI09][MIPI14]****.*

The FAQ questions are organized into Sections by topic, and grouped into two higher-level categories: general questions about MIPI I3C and the ecosystem; and detailed technical questions about material in the I3C specifications.

Section	Title	Focus
2.1	**Introduction to MIPI I3C**	I've heard about I3C. Where can I read a bit more about it?
2.2	**Migration from Legacy I²C or Other Buses**	What are some of the key reasons to upgrade to I3C from other Buses?
2.3	**I3C Versions and Releases**	What versions of I3C or I3C Basic have been released, and what has changed?
2.4	**Up and Coming**	Questions related to the next revision (and/or Errata) of the I3C specification.
2.5	**Naming and Terminology**	Questions related to recent changes to I3C terminology
2.6	**Implementation: Ecosystem**	Questions related to design kits, IP, test, and other parts of the enablement ecosystem.
2.7	**Implementation: As a System Designer**	Questions asked by early system designers.
2.8	**Implementation: As a Software Developer**	Questions asked by early software developers.
2.9	**Interoperability Workshops**	Questions asked by early Interoperability Workshop participants.
2.10	**Conformance Testing**	Questions related to testing device conformance to the I3C specification.
2.11	**Legal and Intellectual Property Related Questions**	Questions related to legal and IPR aspects of the I3C and I3C Basic specifications and implementations.
2.12	**New Capabilities in I3C**	What new capabilities does I3C bring for key use cases?
2.13	**Limits and Performance**	What limits or restrictions should be observed on I3C Buses?
2.14	**Minimum Required Features**	What must all I3C Controllers and Targets support?
2.15	**Backwards Compatibility with I²C**	How can I3C Devices interoperate with Legacy I²C Buses?
2.16	**Address Assignment**	What do I3C Devices need to understand regarding Dynamic Addresses?
2.17	**In-Band Interrupt and Hot-Join**	How can I3C Targets raise Interrupts on an I3C Bus or join an I3C Bus?
2.18	**Common Command Codes (CCCs)**	What are Common Command Codes, how can they be used, and what details need to be understood by system integrators?
2.19	**High Data Rate (HDR) Modes**	How can HDR Modes be used to improve data transfer speeds, and what implementation challenges should be addressed?
2.20	**I3C Advanced Capabilities**	How can I3C's other optional capabilities extend an I3C Bus to meet challenging new requirements for special use cases?
2.21	**Electricals and Signaling**	I've started to read the I3C specification. Tell me a more about the electrical and signaling, and how to design a system for I3C Devices.
2.22	**Bus Conditions and States**	What requirements do I3C Devices need to understand for Bus activity and states?
2.23	**Resets and Error Handling**	How should I3C Controllers and Targets detect errors, and how to recover from errors with defined reset methods?
2.24	**Timing Parameters**	I need more guidance on the specific timing parameters in the I3C specification.

2 Frequently Asked Questions

This FAQ is organized into topics by general area and I3C features/capabilities:

General Questions

Detailed Technical Questions

General Questions

2.1 Introduction to MIPI I3C

Q1.1 What is MIPI I3C and I3C Basic?

MIPI I3C is a serial communication interface specification that improves upon the features, performance, and power use of I²C, while maintaining backward compatibility for most devices.

MIPI I3C Basic is technically identical to MIPI I3C, except with a reduced feature set and RAND-Z licensing (see ***Section 2.11***).

Q1.2 What does the I3C acronym mean?

The official name is *MIPI Alliance Improved Inter Integrated Circuit*.

Q1.3 Why is MIPI I3C being introduced?

The main purpose of MIPI I3C is threefold:

1. To standardize sensor communication,
2. To reduce the number of physical pins used in sensor system integration, and
3. To support low power, high speed, and other critical features that are currently covered by I²C and SPI.

MIPI I3C's purpose is now widening to cover many types of devices currently using I²C/SMbus, SPI, and UART.

Q1.4 What are the main features of MIPI I3C?

MIPI I3C carries the advantages of I²C in simplicity, low pin count, easy board design, and multi-drop (vs. point-to-point), but provides the higher data rates, simpler pads, and lower power of SPI. I3C then adds higher throughput for a given frequency, In-Band Interrupts (from Target to Controller), Dynamic Addressing, advanced power management, and Hot-Join.

Q1.5 For which applications or use cases is I3C intended to be used?

I3C was initially intended for mobile applications as a single interface that can be used for all digitally interfaced sensors. However, it is now intended for all mid-speed embedded and deeply-embedded applications across sensors, actuators, power regulators, MCUs, FPGAs, etc. The interface is also useful for other applications, as it offers high-speed data transfer at very low power levels while allowing multi-drop, which is highly desirable for any embedded system.

Q1.6 How can the MIPI I3C specifications be obtained?

- **MIPI I3C Specification:** MIPI Alliance members have access and rights to the ***MIPI I3C Specification*** through their MIPI membership and member website. The latest adopted version is MIPI I3C v1.1.1 ***[MIPI13]***.
- **MIPI I3C Basic Specification:** MIPI Alliance made the ***MIPI I3C Basic v1.0 Specification [MIPI09]*** publicly available for download in December 2018. The latest adopted version is MIPI I3C Basic v1.1.1 ***[MIPI14]***. MIPI Alliance members have access and rights to the I3C Basic specification through their MIPI membership and member website.
  - **Non-members may download** a copyright-only version of the I3C Basic specification by visiting the MIPI I3C Basic page on the MIPI Alliance website: https://www.mipi.org/specifications/i3c-sensor-specification.

2.2 Migration from Legacy I²C or Other Buses

Q2.1 Why replace I²C with I3C?

While I²C has seen wide adoption over the years, it lacks some critical features – especially as mobile and mobile-influenced systems continue to integrate more and more sensors and other components. I²C limitations worth mentioning include: 7-bit fixed address (no virtual addressing), no in-band interrupt (requires additional wires/pins), limited data rate, and the ability of Targets to stretch the clock (thus potentially hanging up the system, etc.). I3C aims both to fix these limitations, and to add other enhancements.

Q2.2 Does I3C use less power than I²C?

The power consumption per bit-transfer in all I3C modes is more efficient than I²C, due to the use of push-pull (vs. Open-Drain) and strong Pull-Up signaling.

Further, I3C can save considerable device power through higher data rates (because the device can be put back to sleep sooner), built-in configuration and control (without intruding on the main communication protocols), In-Band Interrupt (IBI) as a low-cost wake mechanism, and the ability for Targets to shut down all internal clocks while still operating correctly on the I3C Bus.

Q2.3 How is I3C different from I²C?

I3C offers dynamic address assignment, Target-initiated communication, and significantly higher communication speeds than I²C.

Q2.4 Why replace SPI (Serial Peripheral Interface) with I3C?

SPI requires four wires and has many different implementations because there is no clearly defined standard. In addition SPI requires one additional chip select (or enable) wire for each additional device on the bus, which quickly becomes cost-prohibitive in terms of number of pins and wires, and power. I3C aims to fix that, as it uses only two wires and is well defined.

I3C covers most of the speed range of SPI, but is not intended for the highest speed grades that really only work well with a point-to-point interface, such as for SPI Flash.

2.3 I3C Versions and Releases

Q3.1 What is new in I3C v1.1?

MIPI I3C v1.1 is an advancement of the MIPI I3C Specification that includes not only clarifying edits to make for a more easily-implemented interface, but also new optional features that make I3C even more attractive to a broader set of use cases and industries.

The new features include:

- HDR-BT (Bulk Transport) Mode
- Device to Device(s) Tunneling
- Grouped Addressing
- Multi-Lane for Speed
- Target Reset

Additionally, many clarifying edits have been made to the specification, and the GETHDRCAPS CCC has been replaced with the GETCAPS CCC (which is required for I3C v1.1 Devices).

Q3.2 What are the required features in I3C v1.1 vs. I3C 1.0?

Almost all new features of I3C v1.1 are optional.

However, it should be noted that:

- The formerly optional CCC GETHDRCAPS has been changed to GETCAPS, and its support is required for I3C v1.1 Devices, so as to indicate it is a v1.1 (or later) Device.
- Additionally, it is necessary to support the new RSTACT CCC; as a result, support for a minimal Target Reset is required.

MIPI recommends that Targets consider support of improvement features (such as Flow control for HDR, and Group Addressing), as well as features they likely could use for their application.

Q3.3 Are there any I3C v1.0 features that are not supported in I3C v1.1 and beyond?

The Directed form of the RSTDAA CCC is not supported in I3C v1.1 and beyond.

Q3.4 What is new in MIPI I3C v1.1.1?

MIPI I3C v1.1.1 is an editorial update of MIPI I3C v1.1 that contains no new features or capabilities, but resolves issues, clarifies, and improves on the I3C v1.1 specification.

MIPI I3C v1.1.1 includes:

- Fixes for inconsistencies and other issues that were technical errors in MIPI I3C v1.1 (including all issues resolved by Errata 01)
- Clarifications to the requirements and procedures used for Dynamic Address Assignment, Hot-Join Requests and In-Band Interrupts
- Clarifications to Target Error Types TE0, TE5, and TE6
- Improved clarity and fixes for issues in the sections that define CCC flows in HDR Modes (generic as well as HDR Mode–specific)
- Additional clarifications and explanations in some sections that did not fully explain new features and capabilities introduced with I3C v1.1
- Better explanations of I3C Devices that support advanced features, such as composite I3C Devices that present multiple I3C Target roles (i.e., Virtual Targets)
- Additional clarifications concerning the intersection of several of these new features and capabilities when implemented together (i.e., when used in concert in the same I3C Bus with I3C Devices that choose to support several or all such features or capabilities concurrently)
- Limited allowances for flexibility in how certain new features and capabilities could be applied, or which of these new features and capabilities might be regarded as required, optional, or conditionally required per use case
- Improved descriptions for Multi-Lane Device configuration, including clarified requirements for I3C Devices that present multiple Dynamic Addresses and/or support Group Addressing
- Additional diagrams for HDR-BT Mode, including the 1-Lane forms of the structured protocol elements (i.e., Blocks) that are used in HDR-BT transfers, as well as an example of an HDR-BT transfer
- Improved descriptions of error detection and recovery methods, such as Error Type TE0 and Error Type DBR (Dead Bus Recovery)
- Changes to the Multi-Lane signaling of the Header Byte in SDR Mode

Q3.5 Is I3C v1.1.1 compatible/interoperable with I3C v1.1?

Yes. I3C v1.1.1 is a purely editorial update that aims to resolve inconsistencies and improve clarity compared to I3C v1.1; no new features or capabilities are added. I3C v1.1.1 addresses all I3C v1.1 issues that Errata 01 did. Implementers are strongly advised to confirm that any I3C v1.1–compliant Devices adhere to the fixes published in Errata 01.

In particular, see ***Q4.4*** for fixed issues regarding HDR-DDR Mode inconsistencies pertaining to zero-HDR-DDR-Data-Word flows, which are no longer allowed. This affects all HDR-DDR transactions, including CCC flows in HDR-DDR Mode.

Q3.6 What is new in I3C Basic v1.1.1?

I3C Basic v1.1.1 is a fundamental update to I3C Basic v1.0, and adds many of the new optional capabilities from I3C v1.1. I3C Basic v1.1.1 also includes clarifications and fixes for issues that were addressed in I3C v1.1.1 (see ***Q3.4***). Devices that comply with I3C Basic v1.1.1 should be mutually interoperable with I3C v1.1.1, and vice versa.

I3C Basic v1.1.1 is a subset of I3C v1.1.1, and adds the following features and capabilities:

- HDR-DDR Mode
- HDR-BT (Bulk Transport) Mode
- Grouped Addressing
- Multi-Lane for Speed (for HDR-BT Mode only)
- Target Reset with RSTACT CCC (previously defined in a separate addendum for I3C Basic v1.0)
- Timing Control (Async Mode 0 only)
- CCCs in HDR Modes
- GETCAPS CCC
- SETBUSCON CCC
- SETROUTE CCC
- Virtual Target support

I3C Basic v1.1.1 also includes numerous clarifications and improved definitions, including Secondary Controllers, Hot-Join, Dynamic Address Assignment, Target Error Types and FSM diagrams.

2.4 Up and Coming

Q4.1 What future MIPI specifications will be leveraging I3C?

Many other MIPI Alliance Working Groups are in the process of leveraging the I3C specification. As of the writing of this FAQ, the list includes:

- **Camera WG:** Camera Control Interface (CCI) chapter of the ***MIPI Specification for Camera Serial Interface 2 (CSI-2), v4.0 [MIPI06]*** (In development)
- **Debug WG:** ***MIPI Specification for Debug for I3C, v1.1 [MIPI15]*** (In development)

Q4.2 Are there any impending fixes or errata for MIPI I3C v1.0 or I3C Basic v1.0 that should be applied now?

Note:

*With the release of I3C v1.1 this question has been deprecated; it is retained here for reference. See **Q3.1** for what's new in I3C v1.1.*

All fixes or Errata for I3C v1.0 have also been applied to I3C v1.1. However, I3C v1.1 has since been superseded by MIPI I3C v1.1.1, which is the newest recommended version of the I3C specification.

All fixes for Errata for I3C Basic v1.0 have been incorporated into I3C Basic v1.1.1, which is the newest recommended version of the I3C Basic specification.

Based on learning from early implementations, I3C Interoperability Workshops, queries from adopters, and reviews by the I3C WG, this FAQ represents clarifications, improvements that can be implemented by I3C v1.0 Devices or I3C Basic v1.0 Devices, and other guidance for implementers.

Q4.3 Are any revisions to MIPI I3C v1.0 expected?

No, there are no pending updates at this time. However, MIPI Alliance strongly recommends that all implementers move to MIPI I3C v1.1.1 as the newest recommended version of the I3C specification.

Q4.4 Are there any impending fixes or errata for MIPI I3C v1.1 that should be applied now?

Yes, several fixes to MIPI I3C v1.1 have been identified, and MIPI has published these fixes as Errata 01.

These fixes fall into four categories:

1. **Resolving inconsistencies in HDR-DDR Mode**

   I3C v1.0 and v1.1 did not consistently describe whether an HDR-DDR Data Word was always required for HDR-DDR transactions. Additionally, I3C v1.1 introduced CCC flows in HDR-DDR Mode, which assumed that flows with zero HDR-DDR Data Words were both possible and valid, and it did not always provide appropriate acknowledgement for Target Devices.

   Errata 01 resolves these inconsistencies by always requiring at least one HDR-DDR Data Word for all HDR-DDR transactions. Furthermore, Errata 01 clarifies the requirements for CCC flows in HDR-DDR Mode, and re-defines the special use of the structured protocol elements for these CCC flows to always include at least one HDR-DDR Data Word for appropriate acknowledgement.

2. **Clarifying Open-Drain timing parameters**

   The timing parameters for I3C v1.1 did not show appropriate definitions for a 'Pure Bus' configuration, and also did not provide clarity for sending the first I3C Address Header with the Broadcast Address (i.e., 7'h7E) in order to disable the I^2C Spike Filter (for certain I3C Devices).

   Errata 01 clarifies these timing parameters and fixes other related incorrect timing parameter definitions.

3. **Updating obsolete FSM Diagrams**

   Several FSM diagrams in Annex C that were initially created during early stages of I3C v1.0 specification development were obsolete. For example, the FSM diagram for Dynamic Address Assignment did not reflect the correct Dynamic Address Assignment procedure, and did not include newer CCCs (such as the SETAASA CCC) that were added after I3C v1.0. Several other issues and inconsistencies were also found in other FSM diagrams in Annex C.

   Errata 01 updates these FSM diagrams to reflect the correct procedures (per the normative text in the I3C v1.1 specification) and to remove other obsolete terminology.

4. **Typographical fix regarding HDR-TSP and Multi-Lane support**

   Errata 01 corrects a minor typographical error, and resolves an inconsistency regarding which HDR Modes support Multi-Lane transfers in I3C v1.1.

Additionally, these issues, plus numerous other inconsistencies, clarifications, and fixes have been addressed in MIPI I3C v1.1.1 ***[MIPI13]***. MIPI Alliance strongly recommends that all implementers move to MIPI I3C v1.1.1 as the newest recommended version of the I3C specification.

Q4.5 Are any revisions to I3C v1.1 expected?

MIPI I3C v1.1.1 addresses all issues found in I3C v1.1, including those that were published as Errata 01. MIPI Alliance strongly recommends that all implementers move to MIPI I3C v1.1.1 as the newest recommended version of the I3C specification.

Q4.6 Are any revisions to I3C v1.1.1 expected?

Not at this time. However, the MIPI I3C WG meets regularly and is considering proposals to revise and extend I3C. As part of maintaining the I3C specification, the MIPI I3C WG seeks to improve the I3C specification in areas where it would benefit from clarification or additional explanation. Please direct any comments or suggestions to MIPI Alliance.

Q4.7 Are there any impending fixes or errata for I3C v1.1.1 and I3C Basic v1.1.1 that should be applied now?

Yes, several fixes to the MIPI I3C v1.1.1 and I3C Basic v1.1.1 specifications have been identified. MIPI has published some of these in Errata 01 for I3C v1.1.1, and is currently working to publish others.

These fixes fall into four categories:

1. **Clarifying requirements for Passive Hot-Join**

   A key technical detail was omitted in I3C v1.1.1 and I3C Basic v1.1.1: a passive Hot-Joining Target needs to see an SDR Frame that is addressed to the Broadcast Address (i.e., 7'h7E / W) in order to determine that it is indeed on an I3C Bus (see ***Q17.7***). Additionally, once it sees this SDR Frame, a passive Hot-Joining Target may either (a) pull SDA Low to raise its own Hot-Join Request, or (b) wait for another I3C Device to pull SDA Low and then arbitrate the Hot-Join Address (i.e., 7'h02) into the Arbitrable Address Header.

   Note:

   The I3C v1.1 specification did note the correct requirement for the Broadcast Address.

2. **Clarifying requirements for RSTACT Direct CCC read values**

   In I3C v1.1.1 and I3C Basic v1.1.1, the RSTACT Direct CCC with valid Defining Bytes for different Target Reset operations (i.e., Defining Bytes 0x00 through 0x03) should return the last configured Defining Byte accepted by the Target (unless no RSTACT CCC had previously been used, in which case the defined read value was 0x00). However, implementers discovered that the default read value of 0x00 was not distinguishable from the state when Defining Byte 0x00 had been previously written by the RSTACT CCC (i.e., either Direct Write, or Broadcast use of Defining Byte 0x00 to take no reset action on the Target Reset Pattern, per ***Table 53***). This meant that value 0x00 was not mapped to a single clear state of an unconfigured Target, and this caused ambiguity.

Additionally, since the default action for an I3C Target to take after a Target Reset Pattern with no preceding RSTACT CCC(s) is a reset of the I3C Peripheral, the default read value of 0x00 did not accurately reflect an unconfigured state or the default action that would be taken. The new default read value for the RSTACT Direct CCC with valid Defining Bytes has been changed to 0x80, to distinguish clearly from both (a) the unconfigured state (i.e., armed to reset the I3C Peripheral), and (b) the state where the Target has accepted a previous RSTACT CCC with Defining Byte 0x00 (i.e., take no reset action).

3. **Clarifying requirements for Max Write Length (SETMWL/GETMWL) and Max Read Length (SETMRL/GETMRL)**

   I3C v1.0 specified that the minimum value for SETMWL (Max Write Length) was 8 bytes. I3C v1.1 and v1.1.1 changed this minimum value to 16 bytes. However, this caused confusion for implementers. The minimum values for SETMWL and SETMWL are no longer defined, and implementers may now choose which values for Max Write Length and Max Read Length to support. (See also ***Q18.18***.)

4. **Clarifying minimum timing parameters for the HDR Restart Pattern and Target Reset Pattern**

   I3C v1.1 and v1.1.1 did not explicitly define the minimum timing parameters that the Controller must use when sending either the HDR Restart Pattern or the Target Reset Pattern. Since both of these patterns are based on the HDR Exit Pattern, it was implied that the minimum timing parameters for the HDR Exit Pattern also applied to these other patterns. This is now explicitly defined for all such patterns, and the Controller shall observe the minimum timing as defined by parameter t_{DIG_H} for all transitions. (See also ***Q24.6***.)

MIPI Alliance plans to address these issues by including these clarifications in the next update to the I3C and I3C Basic specifications.

Note:

MIPI Alliance strongly recommends that all implementers use the latest version of the I3C specifications with the most current published Errata.

Q4.8 What new features, if any, are coming to MIPI I3C?

There are no new approved features, however the MIPI I3C WG is considering the following:

- Automotive-focused capabilities
- Security over I3C
- Improved reliability
- Speed increases
- New Multi-Lane uses
- Long Reach
- New HDR Modes
- Refining existing features

2.5 Naming and Terminology

Q5.1 What is an I3C "Controller" Device, and why was the I3C "Master" Device renamed?

As part of a terminology replacement effort across MIPI Alliance, starting with I3C v1.1.1 and I3C Basic v1.1.1 the terms Master and Slave have been deprecated. An I3C v1.0/v1.1 Master Device is now called a Controller. There is no change to the technical definition of such an I3C Device or its role on an I3C Bus. The term Controller is a better, more accurate description of the Device's role on an I3C Bus.

Due to this change, the names of various CCCs and other, related terms have also changed starting with v1.1.1, including:

Deprecated Prior Term *I3C and I3C Basic before v1.1.1*	**Replacement Term** *I3C and I3C Basic v1.1.1 and Later*
Master	Controller
Current Master	Active Controller
Secondary Master	Secondary Controller
Main Master	Primary Controller
New Master (relating to Handoff)	New Active Controller
Master-capable Device	Controller-capable Device
Mastership, Mastering the Bus, etc.	Controller Role, Control of the Bus, etc.
Mastership Request	Controller Role Request
GETACCMST CCC	GETACCCR CCC
Error Types M0 through M3	Error Types CE0 through CE3

See also ***Q5.2***.

Q5.2 What is an I3C "Target" Device, and why was the I3C "Slave" Device renamed?

As part of a terminology replacement effort across MIPI Alliance, starting with I3C v1.1.1 and I3C Basic v1.1.1 the terms Master and Slave have been deprecated. An I3C v1.0/v1.1 Slave Device is now called a Target. There is no change to the technical definition of such an I3C Device or its role on an I3C Bus.

The term Target is a better, more accurate description of the Device's role on an I3C Bus. In particular, the previous term did not describe I3C transfers, which are typically sent by the I3C Controller to individual I3C Devices or to all I3C Devices. The replacement term Target better describes how individual transfers are addressed (i.e., are "targeted") to particular I3C Devices.

Due to this change, the names of various CCCs and other, related terms have also changed starting with v1.1.1, including:

Deprecated Prior Term *I3C and I3C Basic before v1.1.1*	**Replacement Term** *I3C and I3C Basic v1.1.1 and Later*
Slave	Target
Slave Reset Pattern	Target Reset Pattern
DEFSLVS CCC	DEFTGTS CCC
Error Types S0 through S6	Error Types TE0 through TE6

See also ***Q5.1***.

2.6 Implementation: Ecosystem

Q6.1 Who is defining the MIPI I3C Specifications?

The I3C specification is defined by the MIPI Alliance I3C Working Group (originally named the Sensor Working Group) which was formed in 2013. I3C Basic is defined by the MIPI Alliance I3C Basic Ad-Hoc Working Group which was formed in 2018.

Q6.2 Is anyone currently using I3C?

Yes, a number of companies have released products that feature integrated I3C Controller and I3C Target support. Other companies offer IP blocks and associated verification software for adding I3C Bus support into various integrated circuit designs. Some companies also offer protocol analyzers and verification hardware to help analyze I3C Bus traffic for testing and development.

Since this document cannot provide a comprehensive list of such products, those who are interested in learning more about products that support or enable I3C should contact MIPI Alliance.

Q6.3 What is the availability of development hardware for I3C prototyping, including FPGAs?

Several vendors have provided FPGA based design kits, including some low-cost FPGAs that might be good enough for smaller production runs.

Q6.4 What is the I3C IP core availability in the market?

Some vendors have started to offer Target and/or Controller IP cores for integration into ASIC devices and FPGAs, including a free-of-cost Target IP available for prototyping and integration.

2.7 Implementation: As a System Designer

Q7.1 What is the maximum capacitance load allowed on the I3C Bus?

The I3C specification lists the maximum per-Device capacitance on SCL and SDA, but the goal is that most or all Devices will be well below that. As with any Bus, capacitance alone is not sufficient to determine maximum frequency on the I3C Bus. It is important to consider maximum propagation length, effect of stubs, internal clock-to-data (t_{SCO}) of the Targets, as well as capacitive load.

Q7.2 What is the maximum wire length for I3C communication?

The maximum wire length would be a function of speed, as all the reflections and Bus turnaround must complete within one cycle. Larger distances can be achieved at the lower speeds than at the higher ones. For example, at 1 meter (between Controller and Target), the maximum effective speed is around 6 MHz for read, to allow for clock propagation time to Target and SDA return time to Controller.

Q7.3 Can I²C repeaters be used for I3C?

Not directly, for a couple of reasons:

1. The I3C Bus works with push-pull modes (in addition to the open drain for some transfers), and
2. Much higher speeds. Most such devices are quite limited in speed, because of the lag effect of changing states on SCL and SDA due to both series-resistance and assumptions about Open-Drain.

Long wire approaches are being evaluated for a future version of the I3C specification.

Q7.4 Will the I²C devices respond to I3C commands?

No. The I3C CCCs are always preceded by the I3C Broadcast Address, 7'h7E. Since the I²C specification reserves address 7'h7E, no Legacy I²C Target will match the I3C Broadcast Address, and thus no Legacy I²C Target would respond to the I3C commands. Likewise, the Dynamic Addresses assigned to I3C Devices would not overlap the I²C static addresses, so no I²C device would respond to any I3C address – even if it could see it.

Q7.5 How are communication conflicts resolved on the I3C Bus?

I3C Targets are only allowed to drive the Bus under certain situations. Besides during a read, and when ACKing their own address, I3C Targets may also drive after a START (but not Repeated START). After a START, the I3C Bus reverts back to Open-Drain Pull-Up resistor mode; thus, the Target that drives a low value (i.e., logic 0) would win.

Q7.6 Can I3C Devices cause the communication Bus to hang?

Unlike I²C, there is no natural way to hang the I3C Bus. In I²C, clock stretching (where the Target holds the clock low, stopping it from operating) often causes serious problems with no fix: there's simply no way to get the Target's attention if it has hung the Bus. By contrast, in I3C only the Controller drives the clock, and so the Target performs all actions on SDA relative to that clock, thereby eliminating the normal causes of such hangs.

Further, since I3C is designed to ensure that I3C Targets can operate their back-end I3C peripheral off the SCL clock (vs. oversampling), any problems elsewhere in the Target won't translate into Bus hangs.

If a system implementer is highly concerned about a Target accidently locking itself, then a separate hard-reset line could be used. Alternatively, the I3C v1.1.1 and I3C Basic v1.1.1 specifications add a new feature called Target Reset for resetting non-responsive I3C Targets: if an I3C Controller emits the Target Reset Pattern (a defined unique Bus pattern that does not otherwise occur during regular communication), then the Devices on the Bus will treat it just like a hardwired reset line.

This question has been updated for I3C v1.1.1 and I3C Basic v1.1.1.

Q7.7 Will all I3C Devices be compatible with all CCCs?

No. Some CCCs are mandatory, whereas others are optional or conditionally supported, and a given I3C Device will either support them or not, depending upon the Device's capabilities. See also ***Q14.2*** and ***Q18.7***.

2.8 Implementation: As a Software Developer

Q8.1 Are there any companion MIPI I3C Specifications that enable software development?

Yes. The following MIPI specifications are expected to help with software development:

- ***MIPI Specification for I3C Host Controller Interface (I3C HCI), v1.0 [MIPI02]***
  and
  MIPI Specification for I3C Host Controller Interface (I3C HCI), v1.1 [MIPI12]

  Creates a standard definition that allows a single OS driver (also known as 'in-box driver') to support I3C hardware from several vendors, while also allowing vendor-specific extensions or improvements. The target audience of the HCI specification is application processor host controllers; in particular, developers of host controller (i.e., I3C Primary Controller) hardware, and developers of I3C host controller software.
- ***MIPI Specification for Discovery and Configuration (DisCo), v1.0 [MIPI03]***

  Describes a standardized device discovery and configuration mechanism for interfaces based on MIPI specifications, which can simplify component design and system integration. Also oriented to application processors.
- ***MIPI DisCo Specification for I3C, v1.0 [MIPI04]***

  Allows operating system software to use ACPI (Advanced Configuration and Power Interface) structures to discover and configure the I3C host controller and attached I3C Devices in ACPI-compliant systems. Also oriented to application processors.

In addition to these MIPI specifications, a supporting document is also available. The ***I3C Application Note: General Topics, App Note v1.1 [MIPI05]*** has been developed to help ASIC hardware developers, system designers, and others working in the more deeply embedded I3C Devices. The I3C WG plans to develop additional Application Notes that will cover new features and capabilities included in I3C v1.1.1 and I3C Basic v1.1.1.

Q8.2 Are there software libraries available for I3C?

Yes. Core I3C infrastructure has been added to the Linux Kernel as part of the I3C subsystem. The I3C subsystem also includes drivers for several I3C Controller devices and IP core implementations, including MIPI I3C HCI–compliant Host Controllers (see ***[MIPI12]***).

The current list of Linux Kernel Patches for the I3C subsystem can be accessed via ***[LINX01]***.

2.9 Interoperability Workshops

Q9.1 What is a MIPI I3C Interoperability Workshop?

It is a MIPI Alliance sponsored event where different vendors bring their I3C implementations and check interoperation with other vendors.

Q9.2 What is the output from a MIPI I3C Interoperability Workshop?

There are three major outputs from a MIPI I3C Interoperability Workshop:

- Participating vendors can get detailed information about how well their I3C implementations interoperate with other vendors' implementation. Vendors can also compare their results with one another.
- MIPI Alliance can generate an overall picture of the industry state-of-the-I3C-implementaion. For example, how many vendors have implemented I3C, and how many implementations pass or fail against one another.
- The MIPI I3C Working Group gets better understanding about any major issues with the I3C specification. The WG can then leverage that learning by adding to this FAQ, other supporting documents (such as Application Notes, per Q8.1), and possible future revision of MIPI I3C specifications.

Q9.3 Are MIPI I3C Interoperability Workshops an ongoing activity?

MIPI arranges a particular I3C Interoperability Workshop event in response to requests from its membership. They have typically been co-located with regularly scheduled MIPI Member Meetings.

Q9.4 Who can attend or participate in a MIPI I3C Interoperability Workshop?

In general, any MIPI Alliance members who have I3C hardware ready to interop can participate.

Q9.5 What HW/SW is typically needed to participate in a MIPI I3C Interoperability Workshop?

While this could change in future, the minimal requirements to date have been the availability of a board with an I3C Device that can connect to other Devices via the three wires SDA, SCL, and GND. It's also useful to have software (e.g., running on a laptop connected to the board and I3C Device) to interactively view transmitted and received Bus communications, but this might not be required for Targets.

Currently there are solutions working at 3.3V and 1.8V.

Q9.6 Are there any I3C Interoperability Workshops planned for I3C v1.1.1 or I3C Basic v1.1.1?

MIPI Alliance has been hosting I3C Interoperability Workshops in conjunction with MIPI Member Meetings, typically two to three times per year. At this time this FAQ was last updated (August 2022), in-person MIPI Member Meetings have been suspended due to the pandemic. However, MIPI Alliance expects to schedule I3C Interoperability Workshops once in-person MIPI Member Meetings resume.

2.10 Conformance Testing

Q10.1 What is a MIPI Conformance Test Suite (CTS)?

A MIPI WG develops a Conformance Test Suite (CTS) document in order to improve the interoperability of products that implement a given MIPI interface specification. The CTS defines a set of conformance or interoperability tests whereby a product can be tested against other implementations of the same specification.

Q10.2 Is there a MIPI CTS for I3C?

Yes, MIPI Alliance has released a CTS for I3C v1.1.1 and I3C Basic v1.1.1 ***[MIPI08]***.

Q10.3 What is the scope of tests for the I3C CTS?

The CTS tests are designed to determine whether a given product conforms to a subset of the common I3C requirements defined in both I3C v1.1.1 and I3C Basic v1.1.1 (i.e., the requirements that are common to both specifications, since I3C Basic is a subset of I3C). The scope of this version of the CTS is intentionally limited, in order to meet time-to-market requirements imposed by the rapid adoption of I3C in the marketplace, focusing on:

1. SDR-only Devices without optional I3C capabilities,
2. All Controller and Target Error Detection and Recovery methods, and
3. Basic HDR Enter/tolerance/Restart/Exit procedures. However, specific HDR Modes are not covered by this version of the CTS.

Considering the CTS a living document, the I3C WG plans to continue expanding the scope of the CTS through future revisions or subordinate CTS documents for specific features or capabilities (e.g., HDR Modes). The growing set of CTS documents should eventually encompass a broad array of all required and optional features of both the I3C specification and the I3C Basic specification.

The CTS tests are organized as Controller DUT tests (Device Under Test) and Target DUT tests. Tests for each are presented in the order in which they appear in the I3C specification, to simplify identification of pertinent detail between the two documents.

Q10.4 Does the I3C Interoperability Workshop follow the I3C CTS?

Interoperability Workshops will ultimately follow the tests identified in the I3C CTS, as and when such events can be arranged by MIPI Alliance.

Q10.5 What details are provided for each I3C CTS test case?

Each test in the I3C CTS contains:

- A clear purpose
- References
- Resource requirements
- Tracked last technical modification
- Discussion
- All test case detail (i.e., Setup, Procedure, Results, and Problems).

DC/AC parametric requirements are embedded in each test (not split out into a separate PHY-related CTS or subsection).

2.11 Legal and Intellectual Property Related Questions

Q11.1 Is MIPI I3C Basic royalty free?

The parties that directly developed the MIPI I3C Basic specification have agreed to license all implementers on royalty free terms, as further described in the I3C Basic specification document ***[MIPI09][MIPI14]***. Further, all implementers of the I3C Basic specification must commit to grant a reciprocal royalty free license to all other implementers if they wish to benefit from these royalty free license commitments. And, of course, MIPI itself does not charge royalties in connection with its specifications. MIPI's intent is to create a robust royalty free environment for all implementers of I3C Basic.

No set of IPR terms can comprehensively address all potential risks, however. The terms apply only to those parties that agree to them, for example, and the scope of application is limited to what is described in the terms. Implementers must ultimately make their own risk assessment.

Q11.2 What license terms apply to MIPI I3C v1.x?

MIPI's regular IPR terms apply to the full MIPI I3C specification. MIPI's terms require that members make licenses available only to other members, as described in the MIPI Membership Agreement and MIPI Bylaws. To benefit from the license commitments, a party must be a MIPI member.

For features that are included in MIPI I3C Basic, a MIPI member can implement under the regular IPR terms, or can opt to implement the feature under the I3C Basic framework. If a member opts in to the I3C Basic framework, then they must grant the reciprocal licenses required under that framework. MIPI Alliance members are not required to participate in the I3C Basic license framework, however. Features of I3C 1.x that are not included in I3C Basic are subject only to MIPI's regular IPR terms.

Prior to the release of I3C Basic, MIPI had made certain versions of the full MIPI I3C specification available for public review, under "copyright only" terms – that is, MIPI published the specification, but noted that no rights to implement the specification were granted under any party's patent rights. MIPI no longer publishes the full I3C specification publicly. A non-member is not granted any right to implement the full MIPI I3C 1.x specification, either by MIPI or any MIPI member.

I3C Basic is available to non-members, as described in ***Q1.6***.

Detailed Technical Questions

2.12 New Capabilities in I3C

Q12.1 Can I3C Targets initiate communication (i.e., interrupt the Controller)?

Yes, I3C Targets can initiate communication using In-Band Interrupt requests. Communication conflicts are solved by Target Address Arbitration.

Q12.2 How can Controllers and Targets communicate on the I3C Bus?

The basic byte-based messaging schemes used in I²C and SPI map easily onto I3C. Additionally, a set of Common Command Codes (CCCs) has been defined for standard operations like enabling and disabling events, managing I3C-specific features (e.g., Dynamic Addressing, Timing Control), and other functions. CCCs are either Broadcasted (i.e., sent to all Devices on the I3C Bus), or else Directed to a particular Device on the I3C Bus (i.e., by Address).

CCCs do not interfere with, and do not consume any of the message space of, normal Controller-to-Target communications. That is, I3C provides a separate namespace for CCCs (see the specification at ***Section 5.1.9.3***).

Q12.3 What are CCCs (Common Command Codes) and why are they used?

CCCs are the commands that an I3C Controller uses to communicate to some or all of the Targets on the I3C Bus. The CCCs are sent to the I3C Broadcast address (which is 7'h7E) so as not to interfere with normal messages sent to a Target. In other words, CCCs are separated from the standard "content protocol" used by normal messages, such as Private Write and Read transfers (in SDR Mode).

The CCCs are used for standard operations like enabling/disabling events, managing I3C-specific features, and other Bus operations. CCCs can be either Broadcasted (i.e., sent to all Devices on the I3C Bus), or else Directed to specific Devices on the I3C Bus (i.e., by Address). All CCC command number values are allocated by MIPI Alliance, and some values are reserved for specific purposes including MIPI Alliance enhancements and other extensions (see ***Q18.4***).

Q12.4 How are the following similar and/or different: In-Band Interrupt, Hot-Join, and Controller Role Request (IBI / HJ / CRR)?

All three are special, in-band methods that allow the Target to notify the Controller of a new request or state, without having to wait for the Controller to query or poll the Target(s). The term 'in-band' refers to doing this via the I3C Bus wires/pins themselves, rather than using methods requiring extra wires/pins.

- **In-Band Interrupt (IBI):** A Target uses an IBI Request to notify the Controller of a new state or event. If the Target so indicates in the BCR, then an IBI may include one or more following data bytes. A Target can only use IBI if it has indicated the intent to do in its BCR.

  If the Target indicates it will send data with an IBI, then it is required to always send at least 1 byte, called the Mandatory Data Byte (MDB). Starting with I3C v1.1, the MDB is coded following certain rules. Any data after the MDB is a contract between Controller and Target.
- **Hot-Join (HJ):** A Hot-Join Request is used only by an I3C Target that hasn't yet been assigned a Dynamic Address and is attached or awakened on the I3C Bus after the Primary Controller has initialized it. The Hot-Join Request uses a fixed address which is reserved for this purpose only. The Controller will recognize this fixed address and then initiate a new Dynamic Address Assignment procedure. However, a Target cannot use the Hot-Join Request before verifying that the I3C Bus is in SDR Mode.
- **Controller Role Request (CRR):** A Secondary Controller (including the Primary Controller, once it has given up the Controller Role) sends a CRR when it wants to become Controller of the I3C Bus. If the Active Controller accepts the CRR, then it will issue a GETACCCR CCC to pass the Controller Role to the requesting Secondary Controller. It is also possible for the Active Controller

to initiate handoff on its own without any Target initiating an CRR, for example when a Secondary Controller wants to return control.

2.13 Limits and Performance

Q13.1 What is the maximum number of I3C Devices per Bus?

In I3C v1.1 and I3C Basic v1.0 the maximum number of I3C Target Devices was limited to 11. However, this limit was calculated based on typical electrical parameters, whereas different system designs might present other limitations or challenges. Also, the actual number of Targets presented on the Bus could be higher if some of the I3C Devices enable Bridging (see ***Q20.3***) or present Virtual Targets (see ***Q20.2***) with unique Dynamic Addresses.

In I3C v1.1.1 and I3C Basic v1.1.1 the maximum number of I3C Target Devices is no longer stated as a fixed number. Instead, system designers should determine a limit that satisfies all I3C electrical requirements (per ***Section 6***), based on the particular system's unique layout and the unique selection of I3C Devices to be used on the Bus.

Q13.2 Can there be more than one I3C Target inside a chip?

Yes, multi-Target I3C chips are possible. I3C v1.1 also defines Virtual Target capabilities; see ***Q20.2*** for examples of Virtual Targets.

Q13.3 What is the bit rate for I3C?

I3C has several Modes, each with one or more associated bit rates.

The base raw bitrate for SDR Mode is 12.5 Mbps, with 11 Mbps real data rate at 12.5 MHz clock frequency. This is the only Mode supported in both I3C v1.0 and I3C Basic v1.0.

The maximum raw bitrate is 33.3 Mbps at 12.5 MHz, with real data rate of 30 Mbps. This is achieved via HDR Modes that are currently only available in I3C v1.x (i.e., not available in I3C Basic v1.0).

Most traffic will use the 10–11 Mbps rate, while large messages can use one of the optional higher data rate (HDR) Modes or optional Multi-Lane transfers, which are available in I3C v1.1+ or I3C Basic v1.1.1.

Note:

This question was updated for I3C v1.1.1. The I3C Controller should use the GETCAPS CCC (formerly named GETHDRCAPS) to determine which optional HDR Modes are supported for a given I3C Target. For details, see the specification at ***Section 5.1.9.3.19****.*

Q13.4 Is it possible to have multiple Controllers on the same I3C Bus?

Yes, I3C allows for multiple Controllers on the same Bus. However, only one Controller can have control of the Bus (i.e., can possess the Controller Role) at any given time.

An I3C Bus has one Primary Controller that initially configures the Bus and acts as the initial Active Controller. Optionally, the Bus can also have one or more Secondary Controller Devices; these initially act as Targets, but any one of them can send the Active Controller a Controller Role Request (CRR) to ask to take over the role of Active Controller. Once the Active Controller agrees to a CRR and transfers Bus control (i.e., transfers the Controller Role) to a requesting Secondary Controller Device, the requesting Device then becomes the new Active Controller.

Note:

The previous Active Controller (including the Primary Controller) can attempt to regain Bus control by performing this same CRR process. Once the previous Active Controller passes the Controller Role to another Controller-capable Device, it typically acts as a Target (i.e., with limited scope) until it receives the Controller Role again. The terms Active Controller and Secondary Controller reflect the current role of the Device at any given time, not the Device's initial configuration or capabilities. See the specification at ***Section 5.1.7*** *for more details.*

If the Active Controller crashes or becomes unresponsive, then other Controller-capable Devices may use the optional Error Type DBR procedure to test the Bus and regain control if necessary. For details, see the specification at ***Section 5.1.10.1.8***.

Note:

This question was updated for I3C v1.1.1 and I3C Basic v1.1.1.

Q13.5 Can a Target indicate any speed limit that it might have?

All I3C Targets must be tolerant of the 12.5 MHz maximum frequency, and all Targets must be able to manage those speeds for CCCs. However, Targets may limit the maximum effective data rate for private messages – either write, read, or both.

Q13.6 Is there a maximum limit to I3C Bus payload length?

By default, there is no limit to the maximum message length. However, to reduce Bus availability latency across multiple Targets, the I3C Bus allows Controller and Target to negotiate for maximum message lengths. Further, the Controller can terminate a Read, which makes it possible to regain control of the Bus while in a long message.

2.14 Minimum Required Features

Q14.1 Which features are required for a Device to be a compliant I3C Controller?

A compliant I3C Device that can fulfill the Controller Role is required to:

- Assign a unique Dynamic Address to any I3C Targets on the I3C Bus, using any combination of the ENTDAA, SETDASA, and SETAASA CCCs that is appropriate for such I3C Targets.
  - Specification ***Section 5.1.4.2*** defines the full requirements of the Dynamic Address Assignment procedure.
  - The specific CCCs and known Static Addresses (if any) must be a prior configuration, i.e., already known to the system designer.
  - Note that the SETAASA CCC was not defined in MIPI I3C v1.0, it was added in v1.1.
- Manage its Pull-Up structures, including the Open Drain class Pull-Up and High-Keeper Pull-Up for both SDA and SCL. Specification ***Section 5.1.3.1*** defines the full requirements for these Pull-Up structures; see also ***Q21.3*** and ***Q21.4***.
- Manage START requests and Address Header arbitration in Open Drain mode.
- Recover I3C Target Devices using the Error Recovery Escalation Model (per ***Section 5.1.10***).
- Support all of the CCC commands that are mandatory for Controllers, including ENEC, DISEC, ENTDAA, SETDASA, RSTDAA, GETCAPS, RSTACT, GETPID, GETBCR, GETDCR, and GETSTATUS.

Note:

The requirements above apply to the I3C Device that is the Primary Controller of its I3C Bus (i.e., the first Active Controller). An I3C Device that is a Secondary Controller during Bus initialization (or one that subsequently joins after Bus initialization) does not need to meet all of these requirements. See specification ***Section 5.1.7*** *for specific requirements for I3C Buses with multiple Controller-capable Devices, including reduced-function Secondary Controllers.*

Q14.2 Which features are required for a Device to be a compliant I3C Target?

A compliant I3C Device that can fulfill the role of Target is required to:

- Accept an assigned Dynamic Address from the Active Controller, using any or all of the supported methods (i.e., ENTDAA, SETDASA, and/or SETAASA; see ***Q18.7*** and specification ***Section 5.1.4.2***). Note that some of these methods require an I^2C Static Address.
  - If the ENTDAA method is supported, then the Device must have a MIPI Provisional ID and a MIPI-compliant DCR.
- Detect when it is addressed by its assigned Dynamic Address in SDR Mode, and respond to any I3C Private Read or Private Write transfers (as appropriate for the I3C content protocol).
- Respond to the mandatory CCCs for Targets, including ENEC, DISEC, RSTDAA, GETCAPS, GETSTATUS, and RSTACT
  - Note that other CCCs might also be conditionally required, such as GETPID, GETBCR, and (if ENTDAA is supported) GETDCR.
- Detect when the Active Controller enters any HDR Mode, using the ENTHDR0 – ENTHDR7 CCCs, and either:
  - **If the Target supports that HDR Mode:** Monitor Bus activity according to the HDR Mode's signaling and coding, and respond appropriately to any HDR Read or HDR Write transfers that are addressed to the Device's Dynamic Address (or HDR Write transfers that are addressed to an assigned Group Address, if applicable).

  Or:
  - **If the Target does not support that HDR Mode:** Ignore all activity on the Bus until the Target sees the HDR Exit Pattern (per specification ***Section 5.2*** and ***Section 5.2.1***).

- Implement Error detection and recovery methods for an I3C Target, including Error Types TE0 through TE5 per specification ***Section 5.1.10***.

2.15 Backwards Compatibility with I²C

Q15.1 Is I3C backward compatible with I²C?

Yes, most Legacy I²C Target devices can be operated on an I3C Bus, provided they have a 50 ns spike (glitch) filter and do not attempt to stall the clock. Such use will not degrade the speed of communications to I3C Targets; it will require decreased speed only when communicating with the I²C Targets.

I3C supports Legacy I²C Target devices using Fast-mode (Fm, 400 KHz) and FastMode+ (Fm+, 1 MHz) with the 50 ns spike filter, but not the other I²C modes, and not I²C devices lacking the spike filter, or I²C devices that stretch the clock.

Q15.2 Can I3C Devices operate on a Legacy I²C Bus?

I3C Target Devices that have a Static Address can operate as I²C Targets on an I²C bus; optionally, they can also have a 50 ns spike filter.

Additional requirements also apply; see also ***Q15.4*** and the specification at ***Section 5.1.1.1***.

Q15.3 Can I3C and I²C co-exist on the same bus?

Yes, both I3C and I²C can share the same bus, with some limitations:

- I3C does not support Legacy I²C Controller Devices, because they cannot share the Bus with I3C Devices.
- I3C does support many Legacy I²C Target Devices, on the condition that they meet certain guidelines; see also ***Q15.1*** regarding backward compatibility.

Note:

This question has been updated for I3C v1.1.1 and I3C Basic v1.1.1.

Q15.4 How does an I3C Target behave with an I²C Controller vs. with an I3C Controller?

An I3C Target that supports both Legacy I²C and I3C buses typically initializes in I²C mode, as it does not yet possess a Dynamic Address, and also might not know what type of bus is used. However, the Target should be ready to receive a Dynamic Address from an I3C Controller. If the Target also has an I²C Static Address, then it may operate on a Legacy I²C bus using that Static Address. An I3C Target may also support an I²C 50 ns Spike Filter for I²C Fm and Fm+ modes, and it may support other I²C features that are not supported by I3C (such as Device ID). However, per specification ***Section 5.1.1.1***, these features may only be used on a Legacy I²C bus, never on an I3C Bus.

For I3C Buses with such I3C Targets, the I3C Controller must emit the first I3C Address Header with the Broadcast Address (7'h7E) at a rate that is slow enough to be seen through an I²C Spike Filter. This allows such an I3C Target to disable its Spike Filter once it sees the first I3C Address Header with the Broadcast Address (see ***Q24.1*** and the specification at ***Section 5.1.2.1.1***).

If the I3C Target does not have an I²C Static Address, then it will simply wait for the first I3C Address Header from an I3C Controller (i.e., an I3C Address Header containing the Broadcast Address). Such a Target would be of no value on a Legacy I²C bus, since I²C relies on each Target having a Static Address.

Note:

If such an I3C Target supports any Legacy I²C features that are not allowed on an I3C Bus (e.g., clock stretching), then the implementer must ensure that these features are never used, unless the Target knows with certainty that it is on a Legacy I²C bus and not on an I3C Bus.

*An I3C Target must not use clock stretching on an I3C Bus, since clock stretching is not allowed (see the specification at **Section 5.1.1.1**) and the SCL line is typically managed by the I3C Controller, and is driven in Push-Pull mode.*

2.16 Address Assignment

Q16.1 Are all I3C Targets required to support Dynamic Address Assignment with the ENTDAA CCC?

No. If an I3C Target will only be used on I3C Buses that rely on the SETAASA CCC (a Broadcast CCC that auto-sets the Target's Dynamic Address from its I^2C Static Address) and/or the SETDASA CCC (a Direct CCC where the Controller sets the Target's Dynamic Address using a Direct CCC that references the Target's I^2C Static Address), then that Target will never be asked to use the ENTDAA CCC. In such a case, the I3C Target could participate on the I3C Bus despite not implementing ENTDAA CCC support.

An I3C Target may choose to implement support for either or both of the SETAASA and SETDASA CCCs.

Since both of these CCCs rely on the Controller knowing that Target's I^2C Static Address (and perhaps the Static Addresses of all other Targets as well), these methods can save time for certain use cases, although they do impose some implementation requirements on the system designer.

Nonetheless, MIPI Alliance strongly recommends supporting the ENTDAA CCC, otherwise the Device will only ever be usable on that narrow subset of I3C Buses.

Q16.2 How can an I3C Target lose its I3C Dynamic Address, and how does it become an I^2C Target again?

Normally, once an I3C Target is assigned an I3C Dynamic Address, it will be retained until the Target is de-powered. However, an I3C Target will lose its I3C Dynamic Address as a result of the RSTDAA Broadcast CCC, since this resets all I3C Targets back to their initial state. After RSTDAA, I3C Targets that can also operate on a Legacy I^2C Bus (per ***Q15.2***) would behave as I^2C Targets, per the specification at ***Section 5.1.2.1.1***.

Note:

The RSTDAA Broadcast CCC is used to reset all Dynamic Addresses, typically before assigning new Dynamic Addresses to I3C Targets, or to return I3C Targets to their initial state.

An I3C Target could also lose its Dynamic Address under other circumstances, for example:

- The Device is reset by an out-of-band method, such as a pin-reset
- The Device is reset by a full Target Reset (starting with I3C v1.1) which can be invoked two different ways:
  - RSTACT CCC with Defining Byte 0x02, followed by Target Reset Pattern (per the specification at ***Section 5.1.11.2***)
  - Two consecutive Target Reset Patterns (per the specification at ***Section 5.1.11.1***)
- The Device goes into deepest-sleep (i.e., power down)

In the case of an I3C Device losing its Dynamic Address in non-standard ways, the Hot-Join mechanism allows the Target to notify the Controller of the event and receive a new Dynamic Address. In cases where the Controller has deliberately caused the Target to lose its Dynamic Address (e.g., by sending the RSTDAA CCC, or by using a Target Reset), the I3C Controller will start a new Dynamic Address Assignment process using the ENTDAA, SETDASA, or SETAASA CCCs.

Note:

*See also **Q16.8** for CCCs that can change a currently assigned Dynamic Address.*

*See also **Q20.1** for Offline Mode for I3C Targets, where the Devices retain their Dynamic Addresses through a power-down or deepest-sleep cycle.*

Q16.3 What is a Provisioned ID, and why is it needed?

During Bus initialization, the I3C Controller assigns a 7-bit Dynamic Address to each Device on the I3C Bus. For this to happen, each Target device must have a 48-bit Provisioned ID (that is, each Target is provisioned with its ID). The Provisioned ID has multiple fields, including MIPI Manufacturer ID and a vendor-defined part number.

Note:

The I3C Target may also have a Static Address. If the Controller knows this Static Address, then the Dynamic Address can be assigned faster.

Q16.4 How do the first 32 bits of the Provisioned ID (PID) work? Are they random or fixed?

The first part of the PID contains a unique Manufacturer ID. Companies need not be MIPI Alliance members to be assigned a unique Manufacturer ID.

The second part of the PID normally contains a part number (which is normally divided up into general and specific part info for that vendor), as well as possibly an instance number which allows for multiple instances of the same device on the same I3C Bus. The instance ID is usually fed from a pin-strap, fuse(s), or non-volatile memory (NVM).

A random number may be used for the part number, although normally only for test mode, as set by the Controller using the ENTTM (Enter Test Mode) CCC. When a Device that supports random values enters the test mode, the PID[31:0] bits are randomized. When the Controller exits the test mode, the Devices reset bits PID[31:0] to their default value.

Note:

The use of a random number in the PID allows for many instances of the same Device to be attached to a gang programmer/tester, relying on the random number to uniquely give each a Dynamic Address. However, the random number should not be used for typical I3C applications where I3C Devices must be uniquely identified, especially by higher-level software that runs on the Application Host that is driving the I3C Controller.

Q16.5 What if the Controller detects a collision during Dynamic Address Assignment with the ENTDAA CCC?

With most configurations this is not possible, because each Device will have its own Manufacturer ID and a unique part number; as a result, no collisions are possible. But if more than one instance of the same Device (product) is used on a given I3C Bus, then each such instance must have a separate instance ID; otherwise there would be a collision. Likewise, if any Device is using a random number for its part number (i.e., in the PID), then multiple instances from that manufacturer could collide (i.e., could have the same random value that time).

If the Controller knows the number of Devices on the I3C Bus, then it can detect this condition: the number of Dynamic Addresses assigned would be less than the expected number of Devices. If that is detected, then the I3C Controller can take steps to resolve such collisions, for example by resetting all Dynamic Addresses with the RSTDAA CCC and restarting the process, or by declaring a system error after a set maximum number (e.g., 3) of such attempts fail.

Q16.6 What CCCs must an I3C Target support before a Dynamic Address is assigned?

All I3C Targets must be able to process Broadcast CCCs at any time, whether or not they have been assigned a Dynamic Address. I3C v1.1.1 and I3C Basic v1.1.1 clarify the I3C Target requirements (see the specifications at ***Section 5.1.2.1***) and also clarify which CCCs are required to be supported (see ***Q18.7*** and the specifications at ***Section 5.1.9.3***).

Example: An I3C Device may act as an I²C device before it receives its assigned Dynamic Address. However, the Device is still expected to ACK the START with the Broadcast Address (7'h7E). The only exception would be if the Device were to choose to remain an I²C-only device; in this case, the Device would leave any 50 ns spike filter enabled (see the specifications at ***Section 5.1.2.1***).

Note:

Devices that do recognize START with the Broadcast Address could see any CCC (not just ENTDAA, SETDASA, or SETAASA). When determining what effect each CCC will have, these Device may take into account whether or not the Device has received an assigned Dynamic Address. For example, if the Device has not yet received its assigned Dynamic Address, then receipt of the RSTDAA CCC should probably have no effect.

Q16.7 What implicit state or configuration is required for an I3C Device that supports Group Addressing?

Typically, an I3C Device that supports Group Addressing can be assigned to one or more Group Addresses, but has the same I3C Target configuration and state as any other I3C Target (i.e., its role does not change). In effect, this Device gains the ability to receive I3C transfers that are addressed (i.e., targeted) to the Group Addresses to which it might be currently assigned, but the same configuration or state applies to the entire Target, equally for its assigned Dynamic Address as well as any and all assigned Group Addresses.

For some configuration changes, the I3C Controller may use certain Direct CCCs to configure an I3C Target, either individually (i.e., by sending the Direct Write or Direct SET CCC to its Dynamic Address) or in a multicast manner (i.e., by sending the Direct SET CCC to the assigned Group Address). When used as a multicast, the Direct CCC is received by all I3C Targets that are assigned to that Group, and all such I3C Targets apply the same configuration change (if that CCC is supported), exactly as though each I3C Target had received the same Direct CCC addressed to its Dynamic Address. No other internal state is required, and no difference in behavior is expected, when such Direct CCCs are sent to a commonly-assigned Group Address vs. each individual Dynamic Address (see the specification at ***Section 5.1.9.4***).

For other configuration changes, specifically for Multi-Lane configuration changes (if the I3C Target supports Multi-Lane transfers and separate Group Address configurations for Multi-Lane transfers, as defined in specification ***Section 5.3.1.1.1***), a Direct CCC sent to a Group Address is a special configuration operation, and the I3C Target must store this configuration differently than it would for the equivalent Direct CCC sent to its Dynamic Address. The MLANE CCC is a special case requiring different handling, where the Group Address must be treated specially (i.e., differently than the MLANE CCC sent to a Dynamic Address).

Other Direct CCCs are defined in a different way, and the I3C Target must treat Group Addresses specially. Certain Direct CCCs should never be sent to a Group Address (see the specification at ***Section 5.1.2.1.3*** and ***Section 5.1.9.4***.)

Note:

> ***Section 5.1.4.4*** *in v1.1 of the I3C specification has a technical inaccuracy when it states that the SETGRPA CCC "assigns and unassigns a Group Address" to I3C Devices. In fact, the SETGRPA CCC only assigns a Group Address, it does not unassign a Group Address. This misstatement has been corrected in v1.1.1 of the I3C specification.*

Q16.8 Can any CCCs change or override an I3C Target's assigned Dynamic Address?

Per ***Section 5.1.4***, a currently assigned Dynamic Address can only be changed if the Controller sends the SETNEWDA Direct CCC (if supported) or the RSTDAA Broadcast CCC. No other CCCs will change a currently assigned Dynamic Address. This applies regardless of how the Dynamic Address was originally assigned (i.e., either via ENTDAA, SETDASA, or SETAASA).

If the Target also supports the optional SETDASA or SETAASA CCCs, then these only assign the Dynamic Address if it was not already assigned:

1. The Target will only respond to (i.e., will only ACK) and act on the SETDASA CCC sent to its Static Address if the Target did not already have an assigned Dynamic Address at that time. If a Dynamic Address was already assigned, then the SETDASA CCC has no effect.
2. The Target will only act on the SETAASA Broadcast CCC sent to the I3C Bus if it did not already have an assigned Dynamic Address at that time. However, the Target must still provide ACK (as with any Broadcast CCC) but the SETAASA CCC will have no effect.

Additionally, per ***Section 5.1.9.3.4***, a Target that already has an assigned Dynamic Address will not respond to the ENTDAA CCC.

Q16.9 Are I3C Targets required to provide ACK for Broadcast CCCs that are not supported?

Yes. If an I3C Target is powered and knows that it is on an I3C Bus, then it must provide ACK to the Broadcast Address in the SDR Frame (i.e., after a START or Repeated START). This also applies to Broadcast CCCs sent in SDR Mode: an I3C Target must provide ACK to the Broadcast Address before it receives the Command Code for the Broadcast CCC, even if the Target does not support that particular CCC (see ***Q14.2***, ***Q16.6***, and ***Q17.4***).

Note:

*This also applies to the ENTHDR0 – ENTHDR7 Broadcast CCCs that put the I3C Bus into an HDR Mode. If the Target does not support that particular HDR Mode, then it must still provide ACK to such a Broadcast CCC in the SDR Frame, then after seeing the ENTHDRx CCC that it does not support, it must ignore all activity on the Bus until it sees the HDR Exit Pattern (per specification **Section 5.2** and **Section 5.2.1**). Additionally, if the Target did not support such an HDR Mode, it would be unable to recognize the use of the Broadcast Address in subsequent HDR commands (e.g., for CCCs sent in HDR Modes).*

*The only exception to this requirement is if this is a Hot-Joining Target that is not yet safely on the I3C Bus (see **Q17.4**). Such a Target must wait to either see the Bus Idle condition (i.e., for the standard Hot-Join method) or otherwise determine that it is on an I3C Bus by watching for special activity (i.e., for a passive Hot-Join method; see **Q17.7**); before this point, it would not have emitted its Hot-Join Request. In such a case, the Hot-Joining Target would not be able to recognize valid I3C Bus activity.*

2.17 In-Band Interrupt and Hot-Join

Q17.1 What changed with In-Band Interrupts (IBIs) in I3C v1.1.1?

While the definition of In-Band Interrupts has not changed in I3C v1.1.1, some of the details of the Pending Read Notification contract (see specification ***Section 5.1.6.2.2***) have been clarified. An I3C Target Device may only queue one active Pending Read Notification (i.e., a single message that will be read with either an SDR Private Read, or an HDR Generic Read) at a time; and it may now send another IBI if the I3C Controller has not followed up to read the enqueued data for the Pending Read Notification:

- This IBI may be a reminder, using the same Mandatory Data Byte value that was sent earlier (i.e., it cannot be another MDB value that is also used for Pending Read Notifications).
  - If so, then it is forbidden to use it to indicate that additional Pending Read Notifications are waiting (i.e., that they are active and are enqueued after this notification), and the Controller is not obligated to keep count of these IBIs.
- Alternatively, this IBI may also indicate an error code or some other condition (i.e., due to delayed read).
  - However, the I3C Target must still keep the data available to read, if it can.

Q17.2 How can an I3C Controller support Pending Read Notifications?

I3C Controller Devices that comply with the MIPI I3C Host Controller Interface specification (i.e., I3C HCI v1.0 ***[MIPI02]*** or v1.1 ***[MIPI12]***) can easily support Pending Read Notifications. MIPI I3C HCI already defines a standard feature known as "Auto-Command" that conforms to the Pending Read Notification contract and enables (in SDR Mode) automatic initiation of Private Reads or (in supported HDR Modes) Generic Reads in hardware, without software intervention, based on matching MDB values in IBIs received from I3C Target Devices.

Implementers of other I3C Controller Devices could choose to offer support for Pending Read Notifications as a feature in hardware and/or firmware, with varying degrees of configurability. In situations where hardware or firmware support is not feasible, an I3C Controller Device and its Host could also support this in software, provided that the Host's software upholds all the expectations in the Pending Read Notification contract (see specification ***Section 5.1.6.2.2***). Note that this may require the ability to pause or cancel any previously enqueued Read transfers if the I3C Controller receives such an IBI with matching MDB value to signal a Pending Read Notification, as this would oblige the I3C Controller or its Host to initiate a Read (i.e., SDR Private Read or HDR Generic Read) that is expected for this particular IBI.

Q17.3 What is Hot-Join?

The I3C Bus protocol supports a mechanism for Targets to join the I3C Bus after the Bus is already configured. This mechanism is called Hot-Join. The I3C specification defines the conditions under which a Target can do this, e.g., a Target must wait for a Bus Idle condition.

Q17.4 Is an I3C Target required to receive and process the Broadcast ENEC, DISEC, and other Bus-state CCCs before sending a Hot-Join Request, or before being assigned a Dynamic Address?

Yes. An I3C Target is expected to monitor Broadcast CCCs; the special exception is for Hot-Join Targets, as explained below.

Under most circumstances the Target should process all supported Broadcast CCCs, however the Target is specifically required to process supported Broadcast CCCs that affect Bus state. This includes the ENTHDRn CCCs (which turn the HDR Exit Detector on), as well as the ENEC and DISEC CCCs for whatever events the Target supports (e.g., IBI).

Note:

As a practical matter, the I3C Primary Controller will not generally use any CCCs other than Dynamic Address assignment while bringing up the I3C Bus. However, it is allowed to use Bus state CCCs as needed.

For a Hot-Join Target (i.e., a Target that needs to emit a Hot-Join Request to receive a Dynamic Address), this rule only applies once the Target is safely on the I3C Bus and eligible to emit a Hot-Join Request. Such a Target might also join the Bus without knowing the current state, so in order to know that a Broadcast CCC is being sent it needs to see a period of inactivity followed by a valid START with the Broadcast Address. I3C v1.1.1 and I3C Basic v1.1.1 clarify these rules for Hot-Join eligibility; see also ***Q17.10***. In such cases,a Target that has not yet seen a START and has also not become eligible to emit the Hot-Join Request might need to wait for a Bus Available Condition (per specification ***Section 5.1.3.2.2***) before it can see a START; or a Bus Idle Condition (***Section 5.1.3.2.3***) before it would be eligible to pull SDA Low to initiate a START to emit its own Hot-Join Request (i.e., using the standard method).

So, as a general rule, the Target will emit the Hot-Join Request before the I3C Controller is able to emit any Broadcast CCCs. However, after that the Target may see one or more Broadcast CCCs prior to being assigned a Dynamic Address.

Q17.5 Is an I3C Target required to wait the full 1 ms before it can send a Hot-Join Request?

Note:

This question does not apply to I3C Basic v1.0 and I3C v1.1.

In I3C v1.0, an I3C Target was required to wait 1 ms (i.e., the t_{IDLE} minimum value) before it could send a Hot-Join Request. Newer versions of the I3C specification define a new t_{IDLE} minimum value of 200 µs, since that is sufficient for all valid uses. I3C v1.0 devices may choose to support that smaller delay now. I3C Basic v1.0 and I3C v1.0 already support a t_{IDLE} minimum value of 200 µs.

Note:

The new t_{IDLE} minimum value of 200 µs is safe, as SCL at High in HDR Mode cannot exceed 100 µs (i.e., assuming a typical duty cycle) since the minimum I3C Bus frequency being 10 KHz.

Q17.6 Can I3C Hot-Join Target Devices be used on a Legacy I²C bus?

Only if they have a way to turn the Hot-Join feature off, or if passive Hot-Join is supported. (See ***Q17.7***)

Hot-Join Requests are not compatible with Legacy I²C controllers, so Hot-Join would have to be disabled for the Target to be used on a Legacy I²C bus. The disabling of the Hot-Join feature should be done via some feature that is not part of the I3C protocol (i.e., not via the DISEC CCC), since an I²C controller does not support the I3C protocol.

Q17.7 Can an I3C Target support Hot-Join when used on an I3C Bus, and still function correctly on a Legacy I²C Bus?

Yes, but only if it supports the passive Hot-Join method defined in specification ***Section 5.1.5.3***. If the Target does not support passive Hot-Join, then see ***Q17.6***.

Before initiating the Hot-Join Request, a Target that supports passive Hot-Join must first ensure that it is actually on an I3C Bus. This can be done by waiting for a recognizable end of an SDR Frame that ends with a STOP. The key difference is that in order to determine that this is actually an I3C Bus, the Target must first see an SDR Frame addressed to the Broadcast Address (7'h7E / W). Following this, the Target could either pull SDA Low to drive a START condition, or wait to see a START that another I3C Device initiates, before emitting the Hot-Join Request (i.e., arbitrating the special Hot-Join Address 7'h02 into the Arbitrable Address Header following the START).

Note:

A passive Hot-Joining Target needs to see an SDR Frame that is addressed to the Broadcast Address in order to determine that the Bus is in SDR Mode. Without this knowledge, such a Target might see Bus activity in HDR Modes and misinterpret it as STARTs and STOPs. Alternatively, a passive Hot-Joining Target could wait for the HDR Exit Pattern because it clearly indicates a return to SDR Mode.

The detection of an SDR Frame that is addressed to the Broadcast Address is critical because a passive Hot-Joining Target might not engage its timer or oscillator (i.e., to check for Bus Idle or Bus Available condition) until it determines that it is on an I3C Bus and that the Bus is in SDR Mode. Without this detection, such a Target will not know whether it is safe to emit the Hot-Join Request.

If the Target ensures that it is indeed on an I3C Bus in this manner, then it must observe the standard behaviors of an I3C Target that has not yet received a Dynamic Address, as defined in specification ***Section 5.1.2.1***. The Target must acknowledge the Broadcast Address (7'h7E), which means that it is required to understand and process all required CCCs, including ENEC and DISEC. Such a Target is not eligible to respond to the ENTDAA CCC before it has emitted the Hot-Join Request at least once.

A Target that supports both standard and passive Hot-Join methods is free to either A) Initiate a START, wait for the appropriate time (i.e., Bus Idle condition), emit the Hot-Join Request, and then pull SDA Low like a standard Hot-Joining Device; or B) Wait for a START that another I3C Device emits (i.e., after waiting for Bus Idle condition).

Q17.8 Can multiple I3C Targets use the same reserved Hot-Join Address, or can multiple Hot-Joining I3C Targets raise a Hot-Join Request at the same time?

Yes, the reserved Hot-Join Address (7'h02) is safe even if multiple I3C Targets all simultaneously attempt to emit a Hot-Join Request. If multiple I3C Targets do join the I3C Bus and all become eligible to emit the Hot-Join Request (per specification ***Section 5.1.5***) at the same time, then they would be expected (and allowed) to all emit the Hot-Join Request at the same time. Since a Hot-Join Request is a special form of the In-Band Interrupt Request with no data payload (i.e., no Mandatory Data Byte), multiple I3C Targets may all emit this request at the same time, provided that they are all eligible to do so.

Example: As one I3C Target pulls SDA Low to initiate the START Request and drive the 7'h02 Hot-Join Address into the Arbitrable Address Header, and the other eligible I3C Targets see this activity while they are eligible, they would also emit the same Hot-Join Address and behave accordingly. This could be useful if one such I3C Target supported the standard Hot-Join method and waited for the Bus Available condition, while another such I3C Target only supported a passive Hot-Join method but was waiting for another I3C Device to initiate a START Request.

Upon receiving the Hot-Join Request and responding with ACK, the Controller sends the ENTDAA CCC (per specification ***Section 5.1.4.2***) to signal its intent to start the Dynamic Address Assignment procedure and assign Dynamic Addresses to all I3C Targets that have not yet received one. Since the Dynamic Address Assignment procedure inherently supports detection of multiple I3C Targets and uses Arbitration to select one I3C Target at a time (i.e., for each iteration of assignment), the Controller should continue the Dynamic Address Assignment procedure so it can catch all eligible I3C Targets that emitted the Hot-Join Request together (as well as any that might have previously emitted the Hot-Join Request).

See ***Q17.10*** for more information about Hot-Join Requests and the specification clarifications that are new for I3C v1.1.1.

Q17.9 In a Hot-Join, when should the DISEC CCC be sent? After ACK, or after NACK?

After the NACK is preferred, but after the ACK is also acceptable.

The Hot-Join mechanism allows the Controller to first NACK, and then send the DISEC CCC with the DISHJ bit set to disable subsequent Hot-Join Requests. If the Controller ACKs the Hot-Join Request, then that is interpreted as a promise that the Controller will eventually send the ENTDAA CCC to assign a Dynamic Address to the Target(s) that emitted the Hot-Join Request.

If the Controller were to send a subsequent DISEC CCC with the DISHJ bit set, then that would cancel this promise, but it would also leave the Target(s) with no Dynamic Address.

Note:

If any additional Targets joined the Bus after the DISEC CCC was sent, then they would not have seen the DISEC CCC, and as a result would likely send their own Hot-Join Requests.

See ***Q17.10*** for additional clarifications and updates in I3C v1.1.1 and I3C Basic v1.1.1.

Q17.10 What has changed regarding Hot-Join in I3C v1.1.1?

In the I3C v1.0, I3C Basic v1.0, and I3C v1.1 specifications, the requirements for Hot-Join were not clearly defined and the ***Annex C*** Hot-Join FSM diagram did not illustrate all of the possible intended flows.

The I3C WG received implementer feedback on these points, and in I3C v1.1.1 clarified the normative requirements for a Hot-Joining Device. The WG also added a new definition of the Hot-Join procedure which is now defined separately from the Target requirements. In addition, the ***Annex C*** Hot-Join FSM diagram now informatively illustrates a typical flow, rather than being presented as a representation of all possible Hot-Join Request flows.

To summarize the Hot-Join changes:

- A Hot-Joining Device that has not yet raised the Hot-Join Request (i.e., an In-Band Interrupt to the Reserved I3C Address of 7'h02 with Write) does not follow the standard behaviors of an I3C Target that has not yet received a Dynamic Address (as defined in specification ***Section 5.1.2.1***), with the following exceptions:
  - If the Target supports a passive Hot-Join method (specification ***Section 5.1.5.3***) and will be using that method instead of the standard Hot-Join Request (which usually requires the I3C Bus to go idle long enough to satisfy the Bus Idle Condition), then it must verify that the bus it is on is on an I3C Bus by watching for an SDR Frame. ***Q17.7*** provides more clarity on the requirements for Targets that support passive Hot-Join.
- The Controller may choose to either ACK or NACK the Hot-Join Request, and may then send the ENTDAA CCC afterwards:
  - This may happen either immediately (i.e., after the next Repeated START), or later (i.e., after a prolonged period). The ENTDAA CCC might not be in the same SDR Frame, since the Controller may choose to send this after a STOP, followed by a delay (i.e., Bus Free Condition or longer) and then a START. Optionally, the Controller may initiate other transfers and/or CCCs between the ACK/NACK and the ENTDAA CCC.
  - In short, any valid actions are allowed between the ACK/NACK and ENTDAA CCC, and the Target should still respond to the ENTDAA CCC appropriately. If the Target knows that it needs a Dynamic Address and it has already raised the Hot-Join Request, then it should be ready to respond to the ENTDAA CCC when the Controller starts the Dynamic Address Assignment procedure.
  - If the Controller provided an ACK to the Hot-Join Request, then any Targets that raised the Hot-Join Request must remember that an ACK was provided, cease raising Hot-Join Requests, and remain ready to participate in a subsequent ENTDAA CCC (which the Controller should send at a later time). The Target should not send a subsequent Hot-Join Request if the Controller is slow to start the ENTDAA procedure.
  - Although providing an ACK for a Hot-Join Request is a typical flow, providing a NACK is also acceptable. The Target should remain ready to respond to the ENTDAA CCC in either case, even if the Target receives a NACK or is told by the Controller to stop sending Hot-Join Requests (i.e., using the DISEC CCC with the DISHJ bit set).
  - Note that the Controller may choose to send the DISEC CCC with the DISHJ bit set, after either providing an ACK or a NACK to a Hot-Join Request (see ***Q17.9***).
- Any Targets that have already raised a Hot-Join Request are required to also respond to other required CCCs, per specification ***Section 5.1.2.1*** which defines the standard behaviors for an I3C Target that has not yet received a Dynamic Address:
  - Such Targets must acknowledge the Broadcast Address (7'h7E). This means they are required to understand and process all required CCCs, including ENEC and DISEC. (Targets that support passive Hot-Join methods are already required to do this, once they successfully determine that they are indeed on an I3C Bus; see ***Q17.7***).

- After either providing ACK or NACK in response to the Hot-Join Request, the Controller may send (and such Targets are required to properly receive) the DISEC CCC with the DISHJ bit set. The Controller doing so is required to prevent any Targets that raised a Hot-Join Request and received a NACK from subsequently attempting to raise a Hot-Join Request.
- Subsequently, the Controller may send (and such Targets are required to properly receive) the ENEC CCC with the ENHJ bit set, to effectively re-enable Hot-Join Requests on the I3C Bus. Any eligible Targets that receive this ENEC CCC and that previously received the DISEC CCC with the DISHJ bit set may choose to re-send the Hot-Join Request if these Targets both (1) are capable of doing so, and (2) had originally emitted a Hot-Join Request, and (3) have not yet received a Dynamic Address.
- If the Active Controller is a Secondary Controller Device that is not capable of processing Hot-Join or Dynamic Address Assignment, or one that wishes to defer processing to a more capable Controller (i.e., to the Primary Controller), then it may choose to disable Hot-Join on the I3C Bus by sending the DISEC CCC with the DISHJ bit set. It may then pass the Controller Role to any other Controller-capable Device, including but not limited to the Primary Controller.

2.18 Common Command Codes (CCCs)

Q18.1 What are the differences between I3C v1.0 and I3C v1.1 in how CCCs are defined?

CCCs in I3C v1.1 are defined and marked differently than in I3C v1.0, in two important ways:

1. **Conditionally Required CCCs:** In the I3C v1.1 specification, ***Table 16*** in ***Section 5.1.9.3*** (i.e., the main table that defines the I3C Common Command Codes) now marks every CCC as either Required ('R'), Conditional ('C'), or Optional ('O'). Required and Optional retain the same meanings they had in I3C v1.0, but in I3C v1.1 some CCCs have been changed to Conditional.

   For Conditional CCCs the Description column includes a note indicating the condition(s) under which the CCC is required ("Required If:"). Typical conditions would be the use of a particular feature, or support for another particular CCC. If the conditions for a given Conditional CCC are not met, then there is no requirement to support that CCC, and support for it can be regarded as Optional.

   Example: The ENTAS0 CCC is marked as Conditional and only "Required If" any of the other ENTASx CCCs (i.e., ENTAS1, ENTAS2, and/or ENTAS3, all of which are Optional) are supported. Of course, an implementer may choose to support the ENTAS0 CCC even if none of these other Optional ENTASx CCCs are supported. But if at least one of the other ENTASx CCCs are supported, then support for the ENTAS0 CCC is required for that configuration.
2. **CCCs in HDR Modes:** At the discretion of the implementer, CCCs may also be supported in any supported HDR Modes. Specification ***Section 5.2.1.2*** defines the general concepts of CCC framing while in HDR Modes, and specifies key requirements and details regarding their use within an HDR Mode. In general, the use of CCCs within any HDR Mode provides the option to send certain CCCs efficiently while remaining in HDR Mode (i.e., without the extra overhead of exiting HDR Mode and then returning to SDR Mode). The specific CCC framing details for each supported HDR Mode are listed in ***Table 53*** and defined in the sections specifying each HDR Mode.

   Table 51 defines which CCCs may be supported and permitted for use in any HDR Mode, both for Broadcast CCC and Direct CCC flows. Additionally, ***Table 52*** defines which CCCs are prohibited in any HDR Mode and explains why. Since support for CCCs in HDR Modes is optional, and since an implementer might choose to support a subset of CCCs from ***Table 16*** (i.e., those which are also permitted for use, as per ***Table 51*** and ***Table 52***), it is important for the I3C Controller to know which CCCs are supported in HDR Modes. This might include CCCs set aside for Vendor / Standard Extension use, or ones reserved for another MIPI Alliance WG. It is also required that any CCC that is supported in any HDR Mode is supported in SDR Mode too.

Note:

This last point means that if an I3C Device does not acknowledge a given CCC in a given HDR Mode, then the I3C Controller can determine whether that CCC is only unsupported in that HDR Mode (vs. is not supported at all) by retrying that same CCC in SDR Mode.

Q18.2 Does the mandated "single-retry model" apply to all Directed Read CCCs?

At ***Section 5.1.9.2.3*** the I3C v1.1 specification states: “I3C mandates a single-retry model for Direct GET CCC Commands.” Based on this statement, adopters have a general notion that the Controller is required to retry once for the Directed GET CCCs if the Target NACKs the first try. This may not be applicable for other “read” Directed CCCs (such as RSTACT, MLANE, vendor read, etc.) that are not defined for dedicated Read CCCs (i.e., CCCs that have names of the form GETxxx).

Q18.3 What has changed in CCC use or coding in I3C v1.1 or v1.1.1?

The coding and framing details for CCCs have been changed or extended, in three important areas:

1. **Direct Write/Read Commands:** In I3C v1.0, Direct CCCs were either Direct Read Commands (Direct GET CCC) or Direct Write Commands (Direct SET CCC). I3C v1.1 adds an additional Direct Write/Read Command (Direct SET/GET CCC) form: a Direct Write immediately followed by a Direct Read with the same Command Code. This form is used for alternately writing to, and then reading data from, one or more specific I3C Target Devices. The Direct Write/Read Command uses the Direct CCC framing model per specification ***Section 5.1.9.2.2***.

   Example: An I3C Controller might use the MLANE CCC (***Section 5.1.9.3.30***) to configure an I3C Target Device to use a specific Multi-Lane configuration using a Direct SET CCC, followed by a Direct GET CCC to read back the active Multi-Lane configuration and confirm that it was selected for operation. Using both forms of Direct SET CCC and Direct GET CCC within the same SDR frame is considered a Direct SET/GET CCC.

2. **Defining Byte for Directed CCCs:** In I3C v1.1, some Directed CCCs add support for a Defining Byte:

   A. In CCC framing for SDR Mode, the coding of the initial CCC is:

   S, 7’h7E, CCC_byte, Defining_Byte, Sr, Target_Addr, ...

   For more framing details for SDR Mode, see ***Table 15*** in ***Section 5.1.9.1***.

   B. In CCC framing for HDR Modes, the Defining Byte is sent in the HDR-x CCC Header Block of type Indicator. This framing element is defined differently for each HDR Mode; for more framing details, see ***Table 53*** in ***Section 5.2.1.2.1***.

   Depending on the particular CCC, this Defining Byte can be used in several possible ways:

   - It might define a register/code to set, either with or without additional per-Target info
   - It might select a particular register/code to read, from among several available for that CCC
   - It might indicate one of several completely different sub-commands, each of which might be either required or optional, as needed for the particular CCC definition and use case.

   Example: When used with the MLANE CCC, a Defining Byte selects a sub-command. One sub-command might be used to read the available Multi-Lane capabilities for an I3C Target Device, as a Direct GET CCC; another sub-command might be used to either configure an I3C Target Device to use a specific Multi-Lane configuration, or read back the same active Multi-Lane configuration, as either a Direct GET CCC or a Direct SET CCC. For this use case, each Defining Byte has a different purpose and a different, specific data format.

   Example: A Defining Byte for the GETCAPS CCC (see ***Section 5.1.9.3.19***) selects among several different capabilities areas, to read from an I3C Target Device as a Direct GET CCC. For this use case, the Defining Byte may also indicate different formats for the data message returned by the I3C Target Device.

   For some new Direct CCCs defined in I3C v1.1, a Defining Byte is required.

3. **New Optional Defining Bytes:** In I3C v1.1, some CCCs that in I3C v1.0 had no Defining Byte now do have optional Defining Bytes to extend their functionality and support other new behaviors. If the I3C Controller sends the CCC *without* the Defining Byte then the original functionality or behavior occurs, but if the CCC is sent *with* the Defining Byte then the optional extended functionality or new behavior is accessed (see ***Section 5.1.9.2.2***).

   Many Direct GET CCCs that allow optional Defining Bytes also have a method for indicating whether any additional Defining Bytes are supported. This support would be indicated in the data message returned by the Direct version of a particular CCC (which might be the same CCC) when sent without a Defining Byte. This allows an I3C Controller to query an I3C Target Device to determine whether this capability is supported. The particular CCCs that allow optional Defining Bytes are defined in version 1.1 of the I3C specification at ***Section 5.1.9.3***. Additionally, for I3C Target Devices that support such Direct CCCs and also support any optional Defining Bytes, a Defining Byte value of 0x00 generally results in the same behavior as when no Defining Byte is sent.

   Note:

   While Defining Byte support for such Direct GET CCCs is generally optional, certain Defining Byte values are required or strongly recommended for certain use cases, or when used in conjunction with other features or capabilities. Such Direct CCCs list these conditions or recommendations, in addition to any changes in the format of the data message that might be returned when a Defining Byte is used.

   Example: In I3C v1.1, an I3C Controller can use the GETSTATUS CCC (see ***Section 5.1.9.3.15***) to determine the operational status of an I3C Target Device. All I3C Targets support the use of GETSTATUS without a Defining Byte. But if an I3C Target also supports the GETSTATUS CCC with any optional Defining Bytes, then the I3C Controller may also do either of the following things:

   - Send the Direct GET GETSTATUS CCC with a Defining Byte of 0x00, to return the same data message as if no Defining Byte were used;
   - Send the Direct GET GETSTATUS CCC with any other Defining Byte value listed in ***Table 24***. Any I3C Target that supports that particular Defining Byte is required to ACK the CCC and return an appropriate data message for that Defining Byte (i.e., not the standard Target Status). For example, if a Defining Byte value of 0x91 (“PRECR”) is supported, then using this Defining Byte would request the Target to return alternate status information related to the Target’s Secondary Controller capabilities.

Q18.4 What are Vendor / Standard Extension CCCs, who can use them, and how are they differentiated among different uses?

In general, the ranges of command codes byte values that are defined for Vendor or Standards use in specification ***Table 16*** in ***Section 5.1.9.3*** (i.e., the main table that defines the I3C Common Command Codes) can be used by any implementer or any standards developing organization to define custom CCCs that extend I3C to accommodate new use cases. However, the definition of custom CCCs should be considered carefully because the practical issues of implementation might lead to situations where interoperability could be affected across multiple I3C Target Devices that interpret the same command code differently.

For example, different implementers or standards groups might define a custom CCC using the same command code value (i.e., CCC byte value) with (for a Direct GET CCC) different interpretations of the message format that their custom Target Device should return, or (for a Direct SET CCC) different expectations about how their custom Target Device should act; or different expectations about which Defining Bytes might be supported for this CCC, for their custom Target Device. For these conflicting definitions, interoperability issues would certainly arise if the Controller used this custom CCC on an I3C Bus that used custom Target Devices from multiple implementers, or custom Target Devices that conformed to different standards published by several such standards groups. The Controller would not necessarily have knowledge of this situation until it detected protocol issues or encountered other errors.

To help alleviate this situation, I3C v1.1 and I3C Basic v1.1.1 add a new SETBUSCON CCC (see specification ***Section 5.1.9.3.31***) that allows the Controller to set the Bus context. This CCC informs Targets about which version of I3C is used on the Bus, and whether it is a standards-based Bus and/or a vendor-private Bus. This information allows the Target to coordinate its interpretation of extended CCCs, as well as other uses of the Bus, to match the actual current Bus context. Although the SETBUSCON feature is fully optional, it provides a powerful way to align standards-group-based uses of I3C with coordinated private uses. To foster proper coordination, SETBUSCON Context Byte values are published on the MIPI Alliance public website ***[MIPI10]***, and may be assigned by request.

MIPI Alliance strongly recommends that standards groups utilize the SETBUSCON CCC to prevent such interoperability issues, by requesting an assigned Context Byte for their particular usage, which might include a specific interpretation of any command codes that are defined for vendor or Standards use. Additionally, such custom Target Device implementations should not enable this custom interpretation by default, until they receive the SETBUSCON CCC with an assigned Context Byte from the I3C Controller.

MIPI Alliance offers a similar recommendation to implementers seeking to define custom CCCs for private use in a custom Target Device, by using similar logic in such a Target implementation to not enable this custom interpretation by default, until receiving the SETBUSCON CCC with a suitable Context Byte from the I3C Controller to explicitly enable a private interpretation.

Q18.5 How should custom CCCs be used as part of a content protocol based on I3C?

For most use cases, custom CCCs (including the command codes that are defined for vendor or Standards use, see ***Q18.4***) should generally be used as configuration or control commands, sent from an I3C Controller to one or more I3C Target Devices. Using custom CCCs for larger data transfers is not recommended.

When considering the practical concerns of implementing support for custom CCCs in an I3C Target, it is important to note that CCCs in I3C are generally used as shorter messages that are separated from the standard content protocol (i.e., Write and Read transfers in SDR Mode, or any HDR Modes) and are not intended to interfere with normal messages sent to a Target (see ***Q12.3***).

Additionally, since the Command Code and Defining Byte for CCCs are sent to the I3C Broadcast Address 7’h7E and rely on a special ‘modality’ that must be observed by all Targets, handling for custom CCCs might need to be implemented differently within the Target.

With these considerations in mind, implementers should consider using custom CCCs for special purposes, taking advantage of the CCC flows and their separation from the content protocol, to affect changes to the flow or meaning of transfers that are used in their content protocol. By contrast, transfers within their content protocol (such as Write or Read transfers in SDR Mode, or any HDR Modes) should be used for

data-intensive transactions. In most cases, custom CCCs should enable new mechanisms for configuring or controlling a Target Device, while transfers in their content protocol should be used for data transfers between a Controller and a Target.

The following examples are provided as guidance for custom CCC definitions:

- Configuration commands that switch between various supported formats of index or selection that might be used in a Write command, or the first phase of a two-phase Write+Read command.
- Configuration commands that send a directed mode change to particular Targets, or broadcast mode changes to all Targets that support a particular capability or feature (if applicable).
  - Note that custom Broadcast CCCs (i.e., for vendor or Standards use) should be used with caution, as these are received by all Targets on the I3C Bus and various Targets might handle such CCCs differently (i.e., by not using a common interpretation; see ***Q18.4*** for guidance).
- Control commands that switch between multiple endpoints within a Target, using a multiplex model that chooses how and where a standard Write transfer might be directed and processed, or where a Target might source the data message for a Read transfer.
  - In this case, the custom CCC acts as a command to the multiplexor logic.
  - However, such a multiplex model means that only one endpoint at a time can be selected for active use, and this might present a limitation for the use case, and might restrict the modality for using such a Target.
- Special messages sent only to the Target hardware (i.e., the Peripheral logic) whereas the data-intensive transfers in the standard content protocol might be implemented via software or other agents that connect to the Peripheral logic, and would not need to see such special messages.
  - In this case, the Peripheral logic would be expected to offer a quick response due to CCC framing, so a shorter CCC message would offer rapid response due to the expectations based on capabilities in Peripheral logic, versus waiting for software or other agents to respond, which might lead to delays.
  - By contrast, an implementation that used Peripheral logic with software to respond to Direct GET CCCs might not be capable of successfully responding to the first such attempt, and would rely on ideal timing conditions to successfully respond to subsequent attempts.
  - Note that this might only apply to implementations that rely on software, versus implementations that handle custom CCCs natively (i.e., entirely within hardware).
- Commands that change modes to initialize new functions or enable a new modality, which might change the interpretation of standard transfers for the content protocol (e.g., firmware download, key exchange).
  - For example, a custom CCC might act as a method for entering or exiting the modality in which the standard transfers are applied to a new purpose (such as transferring new firmware contents or key data). Upon exiting the modality, the new function of the Target would be applied based on the data transferred during the modality, and the standard content protocol would be used for subsequent operations with the new function.

For other use cases that do not generally follow these guidelines, or that rely on larger message lengths for data-intensive transactions, a content protocol that primarily uses standard transfer commands (i.e., not CCCs) is strongly recommended. Using custom CCCs for data-intensive transactions on the I3C Bus might cause implementation issues for various Target Devices that are based on Peripheral logic and use “software” to handle the response for custom Direct GET CCCs. Additionally, using custom CCCs might introduce integration issues or other platform power concerns that might only appear on I3C Buses with other Target Devices which would not have been fully optimized to ignore such custom CCCs, or with other Target Devices that relied on different interpretations of custom CCCs and might not understand a particular use of SETBUSCON for the use case (as defined in ***Q18.4***). For such situations, it might not be possible to predict these issues in advance until full system integration testing is performed.

Note:

*Higher-level use cases could use a mix of Write/Read transfers for the content protocol, together with custom CCCs for targeted configuration and control functions. Such a protocol would take advantage of the strengths of CCCs, as well as the ways in which CCCs are well-suited for effective changes to control, modes, or operational parameters of an I3C Target Device. For some specific aspects custom CCCs might be recommended, whereas other use cases involving multiple simultaneous endpoints might be better served with Virtual Target capabilities (see **Q20.2**).*

Implementers seeking more specific guidance or recommendations should contact the I3C WG within MIPI Alliance for assistance.

Q18.6 What is the new Command Code value 0x1F for CCCs, and how should it be used?

In I3C v1.1.1 and I3C Basic v1.1.1, a new dummy command code value 0x1F is defined for special use only in CCC flows for HDR Modes that require special structured protocol elements (i.e., Words or Blocks) to conform to that HDR Mode's coding. This 0x1F dummy command code has no meaning as a standard CCC. But in such special flows, it takes the place of a valid CCC and is used in a valid End-of-CCC Procedure to signal the end of CCC modality and return to that HDR Mode's standard generic HDR Write and Read transfers. Dummy command code 0x1F is currently used solely in HDR-DDR Mode, where every HDR-DDR Write must include at least one HDR-DDR Data Word.

Note:

*In I3C v1.1, the equivalent End-of-CCC Procedure was incorrectly defined. Errata 01 for I3C v1.1 addresses this with an updated definition using this dummy command code. Implementers should use only the updated definition in Errata 01 (or newer), or in I3C v1.1.1 (or newer). See **Q4.4**.*

Dummy command code 0x1F may not be used in SDR Mode, nor in any HDR Modes other than HDR-DDR Mode. Dummy command code 0x1F has no meaning as a standard CCC.

Q18.7 Which Dynamic Address Assignment CCCs is a Device required to support?

In I3C v1.0 and I3C Basic v1.0, support for certain CCCs relating to Dynamic Address Assignment (DAA) was defined as always required. The SETDASA CCC was originally intended to be a quicker method of assigning the initial Dynamic Address (i.e., from a fixed Static Address). In I3C Basic v1.0, the SETAASA CCC was also added in response to a request to more quickly assign all Dynamic Addresses from such fixed Static Addresses for I3C Target Devices that support these CCCs and that also have fixed Static Addresses. In both cases, support for the ENTDAA and SETNEWDA CCCs was defined as always required, as the SETDASA CCC was intended as a time-saving alternative for I3C Target Devices that also supported the ENTDAA CCC.

In I3C v1.1, the normative text for Dynamic Address Assignment in specification ***Section 5.1.4.2*** was revised to allow the Controller to end the assignment procedure early without using the ENTDAA CCC, when it knew that all such I3C Targets already had Dynamic Addresses assigned from Static Addresses (i.e., via the SETDASA and/or SETAASA CCCs). Additionally, the SETDASA and SETAASA CCCs were defined as fully-supported options, whereas the ENTDAA CCC was defined as required except under special conditions (i.e., if a fixed Static Address was used). The SETNEWDA CCC was changed to 'conditionally required' status. Unfortunately, the I3C v1.1 specification text incompletely defined when this CCC should be supported, and continued to rely on assumptions about the use of this CCC (in particular, assumptions about the Controller's ability to change a Target's Dynamic Address) which conflicted with the CCC's new definition as being conditionally required depending upon the use case.

The I3C v1.1.1 specification resolves these conflicts and inconsistencies. The definition of the SETNEWDA CCC has been revised to clarify how it affects an I3C Target Device's assigned Dynamic Address, and when the CCC may be used.

Q18.8 Why was the RSTDAA Directed CCC deprecated, and why is it being removed?

The RSTDAA CCC originally (i.e., In I3C v1.0 and I3C Basic v1.0) had a Direct CCC form which could be used to reset a Dynamic Address for a single I3C Target. The Direct CCC form of RSTDAA was deprecated In I3C v1.1 and I3C Basic v1.1.1 because it was realized that the RSTDAA CCC should not be directed to just one Target.

Note:

A Controller that needs to reassign one Target's address can use the SETNEWDA CCC, if the Target supports that CCC.

Q18.9 How is the GETMXDS CCC (maximum data speed) updated in I3C v1.1?

The standard GETMXDS CCC itself was not changed in I3C v1.1, though the definition of t_{SCO} was clarified. However, the GETMXDS CCC now supports a Defining Byte value that allows the return of other forms of information. With no Defining Byte (or with a Defining Byte with value 0), the GETMXDS CCC behaves exactly the same as the original I3C v1.0 GETMXDS CCC.

Q18.10 For Secondary Controller Devices, which format of the GETMXDS Direct CCC is used with the MSTHDLY Defining Byte?

In I3C v1.1, specification ***Section 5.1.7.3*** erroneously stated that this is the GETMXDS Format 2 CCC. In fact, it is Format 3, as stated in other sections. I3C v1.1.1 corrects this error and renames the Defining Byte to CRHDLY (i.e., Controller Role Handoff Delay).

Q18.11 What is the new GETACCCR CCC, and how is it different from the GETACCMST CCC?

In I3C v1.1.1 and I3C Basic v1.1.1, the GETACCMST CCC has been renamed to GETACCCR (Get Accept Controller Role) because the term 'Master' has been deprecated per ***Q5.1***. Note that this is only a naming change; there is no technical change. In all other respects, GETACCCR is identical to the former GETACCMST, and is used in exactly the same manner.

Q18.12 What is the new DEFTGTS CCC, and how is it different from the DEFSLVS CCC?

In I3C v1.1.1 and I3C Basic v1.1.1, the DEFSLVS CCC has been renamed to DEFTGTS (Define list of Targets) because the term 'Slave' has been deprecated per ***Q5.2***. Note that this is only a naming change; there is no technical change. In all other respects, DEFTGTS is identical to the former DEFSLVS, and is used in exactly the same manner.

Q18.13 Why has the figure for the GETCAPS CCC changed?

The I3C v1.1.1 specification has updated the format of the GETCAP Format 1 (Direct) CCC figure in ***Section 5.1.9.3.19*** to make it clearer that I3C Devices that comply with I3C version 1.1 or greater are required to return at least 2 bytes for this CCC. In earlier I3C specification versions, this figure showed four possible options, one of which allowed a single data byte (i.e., GETCAP1 alone). While this was valid for an I3C Device that complied with version 1.0 and supported the GETHDRCAP CCC (the precursor to the GETCAPS CCC), it was not helpful for new I3C Device implementers. The updated figure in v1.1.1 now shows the supported options for I3C v1.1 and higher Devices.

Q18.14 Why have some of the Defining Byte names changed for the GETCAPS, GETSTATUS, and GETMXDS CCCs?

Starting with I3C and I3C Basic v1.1.1, a number of terms used in earlier versions, including the names of some Defining Bytes, have been deprecated as detailed in ***Q5.1*** and ***Q5.2***. Note that these are only naming changes; there are no technical changes. In all other respects, these Defining Bytes are identical to the ones in earlier I3C specification versions, and are used in exactly the same manner.

Q18.15 Where is the Defining Byte for the SETXTIME CCC?

In I3C v1.1 and v1.0, the SETXTIME Broadcast and Direct CCC had a Defining Byte. However, the new Direct CCC framing model introduced in I3C v1.1 used Defining Bytes differently after the Direct CCC. This was done to allow the I3C Controller to send the Defining Byte after the Command Code and before the first Repeated START; doing so gives an individual I3C Target the opportunity to ACK or NACK its Dynamic Address, based on its cached values of the Command Code and Defining Byte (see ***Q18.3***). The SETXTIME CCC's definition of a Defining Byte was not aligned with this new standard framing model, which was a source of confusion.

Since the SETXTIME CCC used this byte as a Sub-Command, the I3C v1.1.1 specification now clarifies this by renaming the SETXTIME CCC Defining Byte: it is now called the Sub-Command Byte, since in the Direct CCC format the byte is sent after the Dynamic Address.

Q18.16 What has changed with the ENDXFER CCC for HDR-TSP and HDR-TSL Modes?

Note:I

This question does not apply to I3C Basic.

In I3C v1.1.1, Defining Byte values 0x7F and 0x55 now describe the process of configuring and enabling Data Transfer Early Termination in addition to the previously described HDR Ternary Flow Control. In the I3C v1.1 specification this was defined as ACK/NACK only for WRITE Commands, whereas it actually applies to all HDR-Ternary Commands. The new definition of HDR Ternary Flow Control clarifies the full set of features that these Defining Bytes (as Sub-Commands) configure.

Additionally, specification ***Section 5.2.3.3*** now provides more context for these capabilities, provides a better explanation of the default state of HDR Ternary Flow Control, and defines which of the procedures defined in its sub-sections apply when HDR Ternary Flow Control is enabled vs. is disabled.

Q18.17 What has changed with the MLANE CCC and Multi-Lane Device configuration?

In I3C v1.1.1 the definition of the MLANE CCC in specification ***Section 5.1.9.3.30*** and details of Multi-Lane Device configuration in ***Section 5.3.1.1*** have been revised and re-written to improve clarity and resolve inconsistencies:

- The MLANE CCC definition was unclear about how Group Addresses can be used with the CCC's Direct SET form. It was also unclear whether all such ML-capable I3C Devices are required to support separate ML configurations for each assigned Group Address. This has now been clarified, and the full behavior of the MLANE CCC is now described in detail.
- Multi-Lane configuration of I3C Devices that support multiple Addresses concurrently (i.e., Group Addresses and/or multiple Dynamic Addresses as Virtual Targets) has been clarified and expanded to cover possible cases of feature intersection. I3C v1.1.1 now describes how I3C Devices handle complex configurations, including the default configuration of newly-assigned Group Addresses based on the I3C Mode and the chosen Data Transfer Coding for Multi-Lane.
- Figures in ***Section 5.3*** showing sample ML Frame formats in HDR Modes included a spurious Repeated START, after the Enter HDR CCC and before Multi-Lane transfers begin in the HDR Mode. The v1.1.1 specification corrects these diagrams; they now conform to the ***Section 5.2.1.3*** text.
- In addition, the I3C WG discovered complexities regarding I3C Devices that support HDR-BT Mode and its Alternate Mode Data Transfer Codings (see specification ***Section 5.3.2.4.1***) with multiple Addresses. These nuanced issues were discovered after I3C v1.1's release and publication, and their full impact was only realized after deep review and investigation by implementers.
- In the v1.1.1 specification, the definitions of HDR-BT ML Data Transfer Codings address additional inconsistencies and typographical errors.

All of these I3C v1.1.1 changes are intended to clarify Multi-Lane, in order to help resolve potential implementation issues and interoperability concerns that might have resulted in differing or incorrect interpretations, including potential assumptions that there are unstated (i.e., implicitly defined) requirements.

Q18.18 How should implementers determine the minimum values for Set Max Write Length (SETMWL) and Set Max Read Length (SETMRL)?

Implementers may choose to support any minimum value for these parameters, based on the Target's use case and the supported I3C content protocol. This overrides any restrictions in previous versions of the I3C Specifications. Implementers may also require that a Target supports any value within the valid range; alternately, Implementers may also allow a Target to reject certain unsupported values within an otherwise valid range (i.e., not applying the change and returning the previous value unchanged on the next use of GETMWL or GETMRL). This allows for I3C content protocols that support byte streaming (i.e., access to a simple buffer or FIFO) as well as structured data messages with mandatory minimum lengths and other overall length requirements (i.e., incremental counts of structures with fixed or variable size).

- In version 1.0 of the I3C Specification, the minimum value for SETMWL was 8 bytes, and the minimum value for SETMRL was 16 bytes.
- In version 1.1 and later of the I3C Specification, the minimum value for SETMWL was changed to 16 bytes.
- Additionally, the GETMRL was previously defined to return a value of 0 for a value that was less than 16 bytes.

All such restrictions (i.e., those listed above) are now removed.

Q18.19 Do the configured Max Write Length and Max Read Length values apply to HDR Modes?

In general, yes. The intent for the SETMWL and SETMRL CCCs was to set the maximum transfer lengths in bytes of meaningful data, without regard to the specific I3C Mode or the Multi-Lane data transfer coding (if applicable). However, many HDR Modes require data bytes to be sent in structured protocol elements (such as Words or Blocks) that hold multiple bytes, where partial data elements are not allowed. Some HDR Modes may require the sender to insert padding bytes when there are not enough meaningful bytes to fill the last data element in a transfer; others might allow the sender to truncate the last data element in the transfer, subject to HDR Mode specific conditions.

If the Target supports SETMWL and/or SETMRL and also allows the Controller to configure a maximum transfer length that is not an integer multiple of the data element size (i.e., the total number of data bytes in the Word/Block), then the Target must interpret the configured maximum transfer length for the meaningful data, and account for any padding bytes beyond the last meaningful data byte.

For example:

- **In HDR-DDR Mode:** Each Data Word holds 2 bytes. If the configured maximum transfer length is an odd number of bytes (i.e., is not an integer multiple of 2 bytes), then:
  - If the transfer length from the sender is the maximum, then the sender must add a padding byte in the last Data Word, and then compute the CRC in the CRC Word based on the entire transfer (including the last padding byte).
    - The receiver will process all Data Words and ignore the padding byte, since the number of meaningful data bytes cannot be larger than the maximum transfer length.
  - Similarly, if the transfer length from the sender is less than the maximum, then the sender would conditionally add a padding byte in the last Data Word, if the transfer has an odd number of meaningful bytes. The CRC in the CRC Word will be computed for the entire transfer (including a possible last padding byte).
    - The receiver will process all Data Words and determine which bytes are valid based on the I3C content protocol; in other words, the receiver must determine whether any padding bytes are present.

- **In HDR-BT Mode:** Each Data Block holds 32 data bytes, but the Last Data Block may hold fewer data bytes. If the configured maximum transfer length is not an integer multiple of 32 bytes, then:
  - The sender will indicate how many valid data bytes are in the Last Data Block, using the **Transition_Control** byte, and will provide padding bytes as needed. The sender will use a sufficient number of Data Blocks to hold the valid data bytes as well as the required padding.
  - The CRC in the CRC Block will be computed based on the valid data bytes in the transfer, but not the padding bytes in the Last Data Block.
  - The receiver will process all Data Blocks that are received, and will explicitly know which bytes in the Last Data Block are valid.

Note:

The appropriate value for padding bytes may vary, per each I3C Mode and particular Multi-Lane data transfer coding (if applicable).

Implementers that support maximum transfer lengths should account for these implications, for transfer lengths that are not an integer multiple of HDR Mode data elements. An I3C Device should be able to handle incoming HDR Mode transfers and determine whether any padding bytes are present, based on the HDR Mode framing, the particular I3C content protocol and the meaning of write/read transfers to/from an I3C Target. However, this may require that the I3C Device is able to receive a sufficient number of data elements (i.e., Words/Blocks) even if these data elements could potentially hold a larger total number of bytes (i.e., due to the required padding bytes).

Note:

*Per **Section 5.1.9.2.2**, I3C Controller and Target Devices shall ignore any additional, unrecognized data bytes in CCC data payloads. Each standard CCC has its own defined data payload format.*

Q18.20 How does the Target respond to the GETSTATUS CCC if the Controller sends ENEC/DISEC CCCs to enable or disable In-Band Interrupts?

If the Target supports In-Band Interrupts, then the GETSTATUS Format 1 response should still indicate whether any interrupts are pending (i.e., are waiting to be sent on the Bus as IBI Requests).

The Controller can use the ENEC and DISEC CCCs to enable or disable IBI Requests (per specification ***Section 5.1.9.3.1***), however this only affects the Target's ability to send a pending interrupt via an IBI Request. If the Controller uses the DISEC CCC with the DISINT bit set, then the Target should still indicate whether it has any pending interrupts in Bits 3:0 of the GETSTATUS Format 1 response, regardless of whether IBI Requests are enabled or disabled.

Q18.21 In the DEFTGTS CCC data bytes, where are the padding bits for the Controller's Addresses?

The data bytes for Addresses, the Controller's Dynamic Address, and the Controller's Static Address (7'h7E) are in Bits[7:1], with Bit[0] used a padding bit with value 1'b0. This is the same format used for the subsequent data bytes that describe the Target and Group Addresses, as well as for other CCCs (such as SETDASA and SETNEWDA).

2.19 High Data Rate (HDR) Modes

Q19.1 Does Figure 44 HDR-DDR Format apply for Command, Data, and CRC? Or only for Data?

Note:

This question does not apply to I3C Basic v1.0.

Only for Data. The Data bytes are sent the same way as in SDR Mode. ***Figure 1*** shows how Data is transferred from memory to the I3C Data Line.

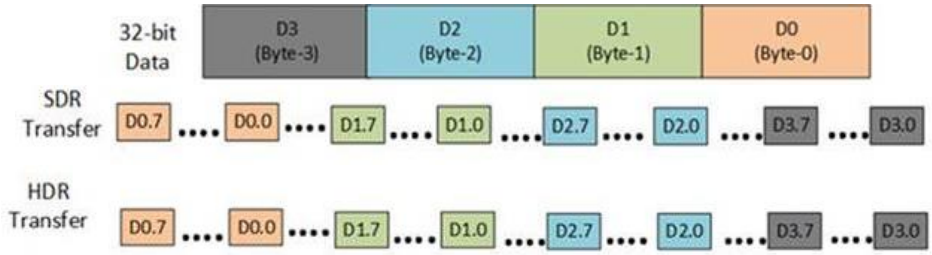

Figure 1 Data DWORDs sent as SDR Data Bytes and HDR-DDR Data Words

The Command and CRC Byte are each transferred as a packet instead:

Command (MSb to LSb):

- Preamble (2 bits)
- Command (8 bits)
- Target address (7 bits)
- Reserved bit (1 bit)

CRC (MSb to LSb):

- Preamble (2 bits)
- Token (4 bits)
- CRC Byte (5 bits)
- Setup bits (2 bits)

Note:

I3C Basic v1.1.1 and I3C v1.1+ already clarify the HDR-DDR protocol and coding details.

Q19.2 Does the Controller provide an additional CLK after the Terminate Condition and before the Restart/Exit Pattern, as shown in Figure 52 Controller Terminates Read?

Note:

This question does not apply to I3C Basic v1.0.

No, there is an error in the I3C v1.0 specification's ***Figure 52***. The beginning of the Restart/Exit Pattern should show SCL Low and SDA changing. This error was corrected in the I3C v1.1 specification and subsequent versions.

Q19.3 During HDR-DDR Mode CRC 5 transmission, how many clocks should the Target expect to receive?

Note:

This question does not apply to I3C Basic v1.0.

The CRC transmission ends at bit 6 (counting down from bit 15), but bit 5 allows High-Z.

Q19.4 For HDR-DDR Mode transfers, how should the Controller manage the Pull-Ups at the end of the HDR-DDR Command Word?

Note:

This question does not apply to I3C Basic v1.0.

Per ***Q21.3*** and ***Q21.4***, the Controller manages its Pull-Ups to allow for ACK/NACK as well as Bus Turnaround. The Controller sends the HDR-DDR Command Word with SDA in Active drive (i.e., Push-Pull mode) for the Command Word until the last Parity bit (i.e., PA0), which is initially driven High. Before the falling edge of C10 cycle, the Controller prepares for the Target to respond by disabling Push-Pull mode and engaging an appropriate Pull-Up to keep SDA at High. After the rising edge of the next C1 cycle, the addressed Target has the opportunity to either:

- Accept the DDR Write or Read by pulling SDA to Low, or
- Ignore the DDR Write or Read by leaving SDA at High (see also ***Q19.5***)

Note:

This scheme is similar to SDR Mode's ACK/NACK scheme for the Address Header, where SDA is "Parked" at High and the Target can pull SDA to Low to acknowledge the transaction.

The Controller should use the appropriate Pull-Up to enable Bus Turnaround or acceptance by the Target. For most use cases, the Open-Drain class Pull-Up is recommended; however, the High-Keeper could also be used, based on system design factors per specification ***Section 5.1.3.1***. In either case, the Target must be able to pull SDA to Low before the falling edge of the C1 cycle.

Note that the next C1 cycle is the start of the first HDR-DDR Data Word (if the Target accepts the transfer) and the PRE1 bit (i.e., the rising edge of the C1 cycle) is always 1'b1 per ***Section 5.2.2***. Using this scheme, the Target's response determines the preamble bits:

- Bits 2'b10 indicate a Target ACK (i.e., accepting the DDR Write or Read) which starts the first HDR-DDR Data Word; or
- Bits 2'b11 indicate a Target NACK (i.e., ignoring the DDR Write or Read). The Controller then drives the HDR Restart Pattern or HDR Exit Pattern.

Q19.5 In HDR-DDR Mode, how do Controllers and Targets enable flow control for NACK?

Note:

This question does not apply to I3C Basic v1.0.

In I3C v1.0, HDR-DDR Mode did not define a flow control mechanism for a Target to actively deny (i.e., to NACK) a DDR Write Command. This meant that Targets had to ignore a DDR Write Command that they did not support (i.e., if the provided value of the HDR Command Byte was not supported), and the Controller would send the Data Words regardless. In such a situation, the Controller would proceed with the DDR Write and the Target would ignore all the Data Words (i.e., would take no action).

In I3C v1.1 and later, HDR-DDR Mode added a mechanism for a Controller to configure a Target to either ignore the DDR Write Command, or else use an ACK/NACK mechanism similar to SDR Mode (see ***Q19.4***). All Targets that comply with I3C v1.1 or newer are required to support this ACK/NACK mechanism, as well as the method of configuration from the Controller. Configuration is accomplished by using the ENDXFER CCC (see ***Section 5.1.9.3.25***).

The flow control diagram in ***Section 5.2.2*** (i.e., ***Figure 128 HDR-DDR Preamble Bits State Diagram***) shows how the suitably configured Target can provide NACK for a DDR Write Command, by providing a 1'b1 during the second preamble bit (i.e., before the first Data Word). In effect, Preamble value 2'b11 means that the Target has provided a NACK to the Controller; if the Controller sees this, then it must drive either the HDR Restart Pattern or the HDR Exit Pattern. By contrast, Preamble value 2'b10 means that the Target has pulled SDA Low and accepted the DDR Write Command (i.e., has provided ACK). ***Section 5.2.2.3.1*** defines the ACK and NACK procedures in greater detail. The Controller must also ensure that the Target's response is received: this typically requires the Controller to use a suitable delay in order to achieve proper set-up timing on the SDA line.

Note:

If the Controller does not configure the Target to use the ACK/NACK capability for DDR Write Commands, then (A) the Target ignores the Data Words for any DDR Write Commands that it does not support; and (B) the Controller always drives Preamble value 2'b10 for the first Data Word, since it knows that the Target will ignore any DDR Write Commands that it does not support.

Q19.6 What has changed regarding HDR Modes in I3C v1.1.1?

Note:

This question does not apply to I3C Basic.

In I3C v1.1 and earlier, the flow for transitioning from ENTHDRx CCCs into HDR Modes was not fully defined. I3C v1.1.1 fully defines the flow for transitioning from such CCCs into the first transfer in an HDR Mode, as well as the corresponding flow transitions from the HDR Restart Pattern to the next HDR transfer in the same HDR Mode.

Q19.7 What has changed regarding HDR Ternary Modes in I3C v1.1.1?

Note:

This question does not apply to I3C Basic.

In addition to the changes regarding ENDXFER and HDR Ternary Flow Control (see ***Q18.16***), I3C v1.1.1 now includes:

- Clarifications on the use of Group Addresses for Direct CCC Read/Write segments in HDR Ternary Modes
- Configuration advisories regarding use of common HDR Ternary Flow Control with CCCs, especially when sending Direct CCCs to Group Addresses.
- Clarification to the Option 2 End of CCC Procedure, which is now similar to starting a new CCC: all I3C Targets must acknowledge the Write Command to the Broadcast Address before the Controller sends an HDR Restart Pattern to end the CCC modality and return to generic HDR Ternary Read/Write commands.

Q19.8 What has changed regarding HDR-BT Mode in I3C v1.1.1?

In addition to the Multi-Lane changes for HDR-BT Mode (see ***Q18.17***), specification ***Section 5.2.4*** now adds:

- Definitions on how to compute the Parity bits in the HDR-BT Header Block
- Definitions and examples for the CRC-16 and CRC-32 polynomials as used for HDR-BT CRC Block checksums
- Clarifications on how the HDR-BT CRC checksum values are calculated based on the data for each HDR-BT transfer
- Correction of an error in the figure showing Direct CCC flows in HDR-BT Mode
- Clarifications on the use of Group Addresses for Direct CCC Read/Write segments in HDR-BT Mode
- Correction of a specification error concerning the use of HDR-BT CRC Blocks in CCC Flows in HDR-BT Mode
- Clarifications to the **Transition_Verify** byte in the HDR-BT CRC Block; Bits[4:2] are now redefined as always zero
- Added new figures in specification ***Section 5.2.4.6*** showing the Single-Lane format for all HDR-BT Mode structured protocol elements
- Corrections to the Dual-Lane and Quad-Lane figures for the HDR-BT Mode structured protocol elements, to resolve inconsistencies with the normative text and to show proper SDA Lane bit packing (i.e., the **Transition**, **Transition_Control**, and **Transition_Verify** bytes)
- Detailed explanation of the Data Block Delay mechanism (formerly called the Stall [delay] mechanism), which allows an I3C Target to delay sending the next HDR-BT Data Block until it has collected enough data bytes (see ***Q19.10***)

- Better explanations of HDR-BT flow control details, including a new ***Section 5.2.4.7*** with example HDR-BT transfers with different actions at the flow control points (i.e., the various **Transition** bytes)

Q19.9 Are there any issues with the HDR-BT diagrams?

Yes. In the I3C v1.1 specification, ***Figure 164 CRC Block for Dual Lane and Quad Lane*** (i.e., the HDR-BT CRC Block) presented the Dual Lane encoding of the CRC Block inaccurately, specifically in the ***Transition_Verify*** byte:

- For HDR-BT Read transfers where the Target provides the clock, the figure incorrectly indicated the "handoff" point (where the Controller resumes driving SCL) as the C12 falling edge, i.e., the very end of the **Transition_Verify** byte for Dual Lane. The text of specification ***Section 5.2.4.2***. and ***Section 5.2.4.3.3***, by contrast, correctly defines the "handoff" point for Dual-Lane as the second half-clock of the **Transition_Verify** byte, i.e., the C11 falling edge.
- ***Figure 164*** also incorrectly showed that the SDA[0] Lane could optionally be High after the "Park1, High-Z" on the C11\ clock cycle (i.e., Bits 4 and 6). The specification text, by contrast, correctly defines Bits[7:2] of the **Transition_Verify** byte as reserved, and requires them to be set to 0. In I3C v1.1.1, the specification text now defines Bits[4:2] as always 0, with Bits[7:5] still reserved for future definition by the MIPI I3C WG.

In both points above the v1.1 specification normative text was correct, and the Dual Lane HDR-BT CRC Block in ***Figure 164*** was incorrect. The "handoff" point is indeed at the C11 falling edge, not the C12 falling edge. The I3C v1.1.1 specification clarifies these details and corrects the figure. Additionally, SDA[0] must be driven Low for Bit 4, even if the receiver does not drive SDA[0] Low and inform the transmitter that the CRC did not match after the "Park1, High-Z" (i.e., for the C11 falling edge).

Note:

*For a Read transfer where the Target was providing clock, the "handoff" is required to occur before the transmitter (i.e., the Target providing Read data) completes transmission of the **Transition_Verify** byte. In this case, the Target must then follow the Controller's clock, since the Controller would drive SCL after the falling edge of C11.*

Figure 2 shows a corrected version of the relevant details for the Dual Lane HDR-BT CRC Block, focusing on the **Transition_Verify** byte. The figure includes a portion of ***Figure 175*** from the I3C v1.1.1 specification (see ***Section 5.2.4.6***), highlighting the changes and clarifications.

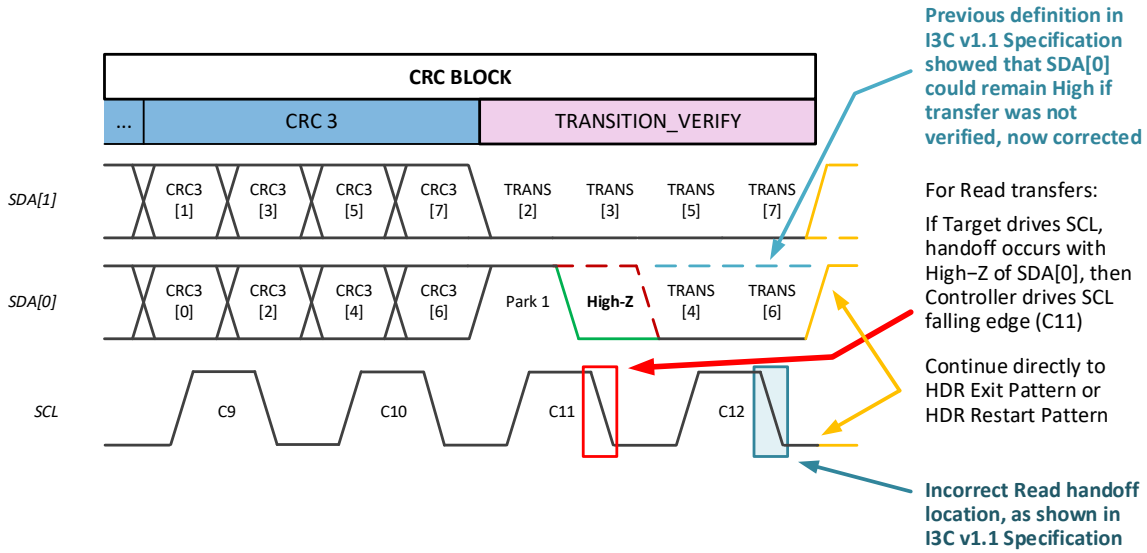

Figure 2 Corrected Figure 164, CRC Block for Dual Lane

Note:

*The I3C v1.1.1 specification provides clarifications and improves the descriptions of the requirements for handling the **Transition_Verify** byte at the end of the CRC Block, for all defined Multi-Lane configurations, including cases where the receiver fails to acknowledge the transfer. In general, the Controller must control SDA[0] and any other SDA[1:3] data Lanes (per the Lane configuration) due to the updated definition of the bit fields in the **Transition_Verify** byte.*

Q19.10 What is the HDR-BT Data Block Delay mechanism?

This mechanism allows an I3C Target to delay sending a complete HDR-BT Data Block during an HDR-BT Read transfer, for situations where the I3C Target or its inner system was unable to fully prepare enough data bytes to fill a Data Block as part of the Read transfer in HDR-BT Mode.

In I3C v1.1, the HDR-BT Data Block Delay mechanism was called the “Stall (delay) mechanism” and was not fully defined. Additionally, the name “Stall (delay)” was too easily confused with other I3C defined behaviors in which SCL clock stalling is permitted (i.e., stalling by the I3C Controller is allowed, but never stalling by an I3C Target). The confusion occurred because this HDR-BT mechanism is fundamentally different from SDR Mode SCL clock stalling, and I3C forbids SCL clock stalling in HDR-BT Mode

The I3C v1.1.1 specification addresses this confusion by renaming the mechanism to more accurately reflect its definition, and by adding clearer, more complete normative details. In addition, the name of parameter “t_{BT_STALL}” was changed to “t_{BT_DBD}”.

During an HDR-BT Read transfer, the Data Block Delay mechanism allows the I3C Target to transmit either:

- A valid Data Block, which is not intended to be the last such Data Block, if the Target has a full 32 bytes of data to transmit; or
- A single Delay byte, indicating that the Target is not yet ready to transmit a Data Block and that the I3C Controller should therefore continue the Read if it wishes to receive more data.

This Delay byte differs from a valid **Transition_Control** byte (i.e., the first byte of an HDR-BT Data Block). The I3C Target may defer sending a Data Block (i.e., by sending a Delay byte up to a maximum of 1024 times) at any point of a Read transfer, before it must terminate the Read transfer. The I3C Controller also has the opportunity either to continue waiting (i.e., receiving more Delay bytes until the I3C Target has enough data), or to terminate the Read transfer at its discretion.

This mechanism does not allow an I3C Target to delay sending either the CRC Block or the Last Data Block (e.g., one that might have “ragged” data).

The I3C Controller determines whether the I3C Target may use the Data Byte Delay mechanism. The Controller indicates this in the Control byte of the HDR-BT Header Block that starts each HDR-BT Read transfer.

- If the I3C Target supports this mechanism and chooses to use it for this transfer, then the Target may use the mechanism if needed.
- If the I3C Target does not support this mechanism, or if the I3C Controller has indicated that the Data Byte Delay is not permitted, then the I3C Target must not use this mechanism, in case it encountered a situation where it was unable to fully prepare enough bytes to fill a Data Block. In such a situation, the I3C Target might be forced to end the HDR-BT Read transfer early (i.e., by sending incomplete data).

This mechanism is primarily useful when the I3C Controller controls the SCL clock during the HDR-BT Read transfer, as the I3C Target would otherwise not have a method for indicating that the transfer should be slowed to match the actual rate of Data Blocks that it can reasonably produce (i.e., from its inner system that might provide the data bytes).

2.20 I3C Advanced Capabilities

Q20.1 What is Offline, and what does it mean to be Offline Capable?

The Offline capability allows a Target to become inactive on the I3C Bus at some times, but then return to normal activity later. The Offline capability is now indicated in the Target's Bus Control Register (BCR). Offline-capable Targets indicate this capability with BCR Bit[3] set to 1'b1.

An Offline-capable Target can become inactive on the I3C Bus, but some portion of the Target will still monitor the Bus either for Target Reset, or for the use of the Target's Dynamic Address. The monitoring portion of the Target retains the Target's Dynamic Address, even though the Target might be mostly powered-off or in deepest sleep.

While offline, such Targets can be awakened (i.e., can be re-activated) by a Target Reset or by the use of their Dynamic Address. They will take some time to become active on the Bus again, such as the RSTACT CCC recovery from Full Reset time. While they are offline, and while awakening, these Targets will not be responsive to the Controller, nor will they record CCCs, nor will they necessarily retain state (e.g., the ENEC/DISEC CCCs); as a result, the Controller will have to wait until such Targets become active, and then might also need to configure them again. This is all by private contract (agreement).

See specification ***Section 5.1.10.2.5*** for details, both in I3C v1.0 (which does not include Target Reset) and in I3C v1.1 (which does include Target Reset).

Note:

Offline-capable Targets are required to preserve their Dynamic Addresses as long as they are powered. A Target that can become fully inactive on the I3C Bus and lose its Dynamic Address will subsequently Hot-Join to get a new Dynamic Address; such a Target is not considered to be an Offline-capable Target, and it must set BCR Bit[3] to 1'b0.

Q20.2 What is a Virtual Target?

Generally, a Virtual Target is a function of a single physical Device that represents multiple I3C Targets on the I3C Bus, such that the I3C Controller can address each of those Targets independently.

In the simplest form, a Virtual Target could be one of a set of several Target Devices, all integrated into the same physical package and all sharing a common set of pins connecting them to an I3C Bus.

In a more advanced form, a Virtual Target could act as one of several virtualized functions presented by a highly-integrated Device that stores a different Dynamic Address for each function. Depending on the implementation, such virtualized functions might share configuration information, and might return the same values for some CCCs.

Examples could include Bridging or Routing Devices, as well as other types of Devices that expose multiple functions and use shared Peripheral logic. I3C v1.1 defines several new capabilities and features for such Virtual Targets.

For details, see the I3C Specification at ***Section 5.1.9.3.19***, and the ***I3C Application Note: General Topics [MIPI05]*** at ***Section 5.7***.

Q20.3 Does the I3C Bus support Bridges?

Yes. Bridge Devices enable an I3C Bus to be bridged to other protocols, such as SPI, UART, etc. The SETBRGTGT (Set Bridge Target) CCC is defined to enable Bridge Devices, where the Controller either knows in advance that certain Devices are bridges, or can discover a Bridge Device during Bus initialization.

Q20.4 How does the Set Bridge Targets (SETBRGTGT) CCC differ between I3C v1.0 and I3C v1.1+?

In I3C v1.1, use of this CCC is expanded to support Bridging Devices that have Dynamic Addresses, which makes multiple Bridging Devices on the same I3C Bus possible. The context of the SETBRGTGT CCC is also redefined: a Bridging Device is now one of several types of Devices that expose or present Virtual Targets. Bit[4] of the Target's Bus Configuration Register (BCR) now indicates Virtual Target capabilities.

For bridged Targets that are enabled by a Bridging Device, I3C v1.1 clarifies the use of other CCCs (such as SETMRL) that address a bridged Target. The I3C v1.1 specification also clarifies that to change the Dynamic Address of a bridged Target, the SETBRGTGT CCC (not the SETNEWDA CCC) must be used.

For details, see the I3C Specification at ***Sections 5.1.1.2.1***, ***5.1.9.3.17***, and ***5.1.9.3.19***.

Q20.5 Does the I3C Bus enable Routing?

Yes. I3C v1.1+ and I3C Basic v1.1.1 define the requirements, expectations, and configuration for Routing Devices.

Routing Devices enable the creation of multiple Routes across I3C Buses. A Routing Device enables more advanced Bus topologies, and requires buffers or queues to handle transactions across each Route.

A Routing Device contains a Control Function which is presented on the I3C Bus as a Virtual Target. The Controller configures the Routing Device by sending the SETROUTE CCC to its Control Function. Routes to other I3C Buses are treated as downstream targets, each of which generally has a Target Function which is also presented as a Virtual Target with its own Dynamic Address. Transactions are sent to the Route's Target Function via its Dynamic Address, and the Routing Device manages the communications on the downstream I3C Bus.

For details, see the I3C Specification at ***Section 5.1.9.3.20***.

Q20.6 Why does I3C allow more than one Controller on the I3C Bus? What can a Secondary Controller do that the Primary Controller can't?

The system designer decides whether their system needs more than one Controller on the Bus. To provide that flexibility MIPI I3C allows this, but also does not require it. Controller Role Handoff is a well-defined and controlled mechanism in I3C; if used, it can be relied on.

For most use cases, a Secondary Controller is not a required component for an I3C Bus. If your Primary Controller has all the capabilities and features that you need, and if your use of the I3C Bus wouldn't benefit from having multiple Controller-capable Devices, then you might not need a Secondary Controller.

Examples where Secondary Controllers are useful:

1. A Debug controller, if present, would be a Secondary Controller
2. A sensor hub or other offload device can be used to continue operating during periods when the Host processor is in deep sleep (i.e., to save power)
3. The system can switch between standalone use (such as IoT devices) and connected uses. For example, perhaps a USB-to-I3C cable is attached to take over temporarily.

Note that Devices such as MCUs will usually be able to operate as fully-fledged Targets, as fully-fledged Primary Controllers, and as Secondary Controllers (i.e., Devices that come up as Targets but can become the Active Controller later), depending solely on the particular needs of the given system. They can simply be configured for the Bus they are on by the firmware.

Note:

*This FAQ entry has been updated for I3C v1.1.1 and I3C Basic v1.1.1. In this FAQ entry, the terms "Primary Controller" and "Secondary Controller" refer to an I3C Device's initial configuration and capabilities. In other sections of this FAQ and the I3C specification, the term "Secondary Controller" might instead reflect a Controller-capable Device's current role, i.e., as and when it is not currently the "Active Controller" of the Bus. See **Q13.4** for additional details.*

Q20.7 Is any time-stamping capability defined for the I3C Bus?

Note:

- *This question does not apply to I3C Basic v1.0.*
- *I3C Basic v1.1.1 only supports Timing Control with Async Mode 0.*

Yes. The I3C Bus supports an optional Timing Control mechanism which has multiple timing modes. One timing mode is synchronous (from the synchronized timing reference) and four modes are asynchronous (Target provides timestamp data). All I3C Controllers are expected to support at least Async Mode 0.

- **Synchronous:** The Controller emits a periodic time sync that allows Targets to set their sampling time relative to this sync. This may be used in conjunction with one of the Asynchronous modes.
- **Asynchronous:** The Targets apply their own timestamps to the data at the time they acquire samples, permitting the Controller to time-correlate samples received from multiple different Targets or sensors.

  There are four types (timing modes) of asynchronous time controls:

  - **Async Mode 0:** Basic timing mode that assumes that a Target has access to a reasonably accurate and stable clock source to drive the time stamping – at least accurate for the duration of the time it has to measure (i.e., from event to IBI). A set of counters, in conjunction with IBI, are used to communicate time stamping information to the Controller.
  - **Async Mode 1:** Advanced timing mode extends the Basic mode by using some mutually identifiable Bus events, like I3C START.
  - **Async Mode 2:** High-precision timing mode that uses SCL falling edges (for SDR and HDR-DDR modes) as a common timing reference for Controller and Target. A burst oscillator is used to interpolate the time between a detected event and next SCL falling edge. For HDR-TSL and HDR-TSP modes, this timing mode uses both SDA transitions and SCL transitions as timing references.
  - **Async Mode 3:** Highest-precision triggerable timing mode that supports precise time triggering and measurement across multiple transducers applications like beam forming.

Q20.8 Can Synchronous and Asynchronous Timing Control both be enabled at the same time?

Note:

This question does not apply to I3C Basic v1.0.

Yes. This is allowed such that the ODR (Output Data Rate) rate controls the In-Band Interrupt (IBI) rate, and the Async Mode timestamp on the IBI indicates how long ago the sample was collected.

Q20.9 Is there a way to turn off Timing Control?

Note:

This question does not apply to I3C Basic v1.0.

Yes, the Controller can turn off Timing Control by sending the SETXTIME CCC with the value 0xFF in the Sub-Command byte.

Q20.10 What has changed regarding Multi-Lane for SDR Mode?

Note:

This question does not apply to I3C Basic v1.0.

In I3C v1.1, the Data Transfer Codings for Multi-Lane in SDR Mode (SDR-ML) define the Header Byte to contain supplementary information to be sent on SDA[1] (i.e., the first Additional Data Lane) when Multi-Lane is enabled for SDR Mode. This supplementary information includes the inverse of the Address, since the Header Byte was defined to include the I3C Address Header and was intended to be received and understood by existing I3C Targets that did not necessarily support SDR-ML.

Upon review, the I3C WG realized that this method would cause implementation challenges during an Arbitrable Address Header (i.e., after a START) if the I3C Controller were required to echo any changes it saw on SDA[0] (i.e., to track changes that an I3C Target might drive during Address Arbitration) when emitting the inverse on SDA[1]. The I3C v1.1.1 specification address these issues by changing the definition of the Header Byte for SDR-ML to only require the I3C Controller to send this supplementary information on SDA[1] for the Header Byte (i.e., an Address Header) after a Repeated START. When sending the Header Byte after a START, the I3C Controller is now required to keep SDA[1] and any other Data Lanes in a High-Z state, rather than driving this supplementary information.

In SDR Mode, CCCs are always sent in 1-Lane mode, allowing all I3C Targets to track the Command Code and Defining Byte (if any) in the CCC Framing. This rule places limitations on the use of SDR-ML:

1. If SDR-ML is used, then the Targets should not rely on supplementary information on SDA[1] for the Header Byte (i.e., the Address Header); the supplemental information should be treated as optional, because 1-Lane Targets (i.e., Targets that might not support SDR-ML) must track CCC framing and flow elements but can only see SDA[0].
2. Transfers after a Repeated START that comprise Broadcast CCCs (i.e., transfers addressed to 7’h7E) must also be sent only in SDR 1-Lane mode. As a result, SDR-ML cannot be used for Broadcast CCCs in SDR Mode.
3. Transfers after a Repeated START that comprise CCC flow elements (e.g., Direct CCC segments addressed to a specific Target or Group) must only be sent in SDR 1-Lane mode. As a result, SDR-ML also cannot be used for Direct CCCs in SDR Mode.

Conditions #2 and #3 above are especially relevant because both Broadcast CCCs and Direct CCCs may be mixed in among Private Write/Read transfers in continuous SDR Mode framing (i.e., without intervening STOP Conditions). In such cases the Controller should not send the supplementary information during the Address Header, and Targets supporting SDR-ML are required not to depend on it, because they will know that the Controller is sending a CCC. Furthermore, the Controller must not use SDR-ML data byte encoding for CCCs (both Broadcast and Direct) because some Targets on the I3C Bus might not understand the encoding.

2.21 Electricals and Signaling

Q21.1 How many signal lines does I3C have?

I3C has two mandatory signal lines: Data (SDA) and Clock (SCL).

I3C v1.1+ and I3C Basic v1.1.1 also support optional Multi-Lane transfers, which use additional Data lines for supported I3C Modes. In Multi-Lane, the SDA line is called SDA[0] and the additional data lines are called SDA[1], SDA[2], etc.

Q21.2 Does I3C require Pull-Up resistors on the bus like I²C?

Not necessarily. I3C Controllers manage an active (i.e., dynamic) Pull-Up resistance on SDA, which they can enable and disable as the Bus transitions between Open Drain and Push-Pull mode. This might be a board-level resistor that is switchable (i.e., that can be engaged/disengaged as needed, controlled by an output pin from the Controller), or internal to the Controller, or any combination of the two.

Note:

If an I3C Bus has multiple Controller-capable Devices (i.e., one Active Controller and one or more Secondary Controllers), then the Active Controller manages the Pull-Up on SDA.

Q21.3 When is the Pull-Up resistor enabled?

In order to achieve higher data rates, much of the activity on the I3C Bus occurs in Push-Pull mode (i.e., with the Open Drain Pull-Up resistor disabled).

However for some Bus management activities, and for backwards compatibility with I^2C, Pull-Up-resistor-based Open Drain mode is enabled. Examples include:

- Arbitration during Dynamic Address Assignment
- Address Header following a START, which is arbitrable and can become an In-Band Interrupt Request
- ACK/NACK during the 9^{th} bit of an Address Header.

With very few exceptions, the I3C Controller is responsible for providing an Open Drain class Pull-Up resistor when the Bus is in the Open Drain mode. See specification ***Section 5.1.3.1***.

Note:

The Open Drain class Pull-Up for SDA could either be provided internally by the Controller, or it could be an external device that is engaged or disengaged as needed, i.e., switchable between these states and controlled by a Controller output pin.

By contrast, SCL should always be driven by the Active Controller (i.e., Push-Pull) and no Open Drain class Pull-Up is needed.

Q21.4 Is a High-Keeper needed for the I3C Bus?

A High-Keeper is used for Controller-to-Target and Target-to-Controller Bus handoff, as well as optionally when the Bus is idle (see ***Q22.1***). The High-Keeper may be a passive weak Pull-Up resistor on the Bus, or an active weak Pull-Up (or equivalent) in the Controller. The High-Keeper is only required to be strong enough to prevent system-leakage from pulling the Bus Low. At the same time, the High-Keeper for the SDA line must be weak enough that a Target with the minimum I_{OL} driver can pull the SDA line Low within the $\mathbf{t_{rDA}}$ minimum period. The system designer is responsible for balancing these factors.

Note:

*A High-Keeper is required for both SDA and SCL. External High-Keepers might be required if the Active Controller's High-Keeper is not adequate per specification **Section 5.1.3.1**. Additionally, if an I3C Bus has multiple Controller-capable Devices (i.e., one Primary Controller and one or more Secondary Controllers), then the Active Controller is responsible for managing the High-Keeper, and Controllers using the Controller Role Handoff Procedure are required to engage or disengage their High-Keepers at the defined times during this procedure (per **Section 5.1.7.2**).*

2.22 Bus Conditions and States

Q22.1 What are some of the I3C Bus conditions when the Bus is considered inactive?

In addition to Open-Drain, Pull-Up, and High-Keeper, the I3C Bus has three distinct conditions under which the Bus is considered inactive: Bus Free, Bus Available, and Bus Idle.

- **Bus Free** condition is defined as a period occurring after a STOP and before a START and for a given duration (e.g., t_{CAS} and t_{BUF} timing).
- **Bus Available** condition is defined as a Bus Free condition with duration of at least t_{AVAL}. A Target may only issue a START request (e.g., for In-Band Interrupt or Controller Handoff) after a Bus Available condition.
- **Bus Idle** condition is defined to help ensure Bus stability during Hot-Join events. This condition is defined as a period during which the Bus Available condition is sustained continuously for a duration of at least t_{IDLE}.

Q22.2 When an I3C Device wishes to send an In-Band Interrupt (IBI) Request, does it need to see a STOP before a Bus Idle?

For normal active I3C Targets, yes. They should only send an In-Band Interrupt (IBI) Request when they have seen a STOP and the t_{AVAL} time has elapsed (about 1 µs), and in response to a START (but not a Repeated START).

For Hot-Joining Devices using the standard Hot-Join method, they do not necessarily know the Bus condition, so they wait until the Bus is Idle (i.e., until SCL and SDA are both high for a duration of at least t_{IDLE}).

Q22.3 When can an I3C Target issue an In-Band Interrupt (IBI) Request?

An I3C Target can issue the IBI in the following two ways:

- Following a START (but not a Repeated START)
- If no START is forthcoming within the Bus Available condition, then an I3C Target can issue a START request by pulling the SDA line Low. The I3C Controller would then complete the START condition by pulling the SCL clock line Low and taking over the SDA line.

Q22.4 What are the I3C Bus Activity States?

Bus Activity States provide a mechanism for the Controller to inform the Targets about the expected upcoming levels of activity or inactivity on the Bus, in order to help Targets better manage their internal states (e.g., to save power).

There are four Bus Activity States, each with an expected activity interval:

- **Activity State 0:** Normal activity
- **Activity State 1:** Expect quiet for at least 100 µs
- **Activity State 2:** Expect quiet for at least 2 ms
- **Activity State 3:** Expect quiet for at least 50 ms

2.23 Resets and Error Handling

Q23.1 Are there any test modes in the I3C Bus?

Yes. The Directed and Broadcast ENTTM CCCs (see specification ***Section 5.1.9.3.8***) allow the Controller to enter and exit the test modes. Support for the ENTTM CCC is optional for I3C Targets.

The I3C v1.1+ and I3C Basic v1.1.1 specifications also define an optional Defining Byte for the GETCAPS CCC: the Read Fixed Test Pattern command (see specification ***Section 5.1.9.3.19***). This provides simple Bus testing that can be supported by I3C Controllers and Targets.

Q23.2 Are there any error detection and recovery methods in I3C?

Yes, the I3C Bus has elaborate error detection and recovery methods. Seven Target error types (TE0 through TE6) and four Controller error types (CE0 through CE3) are defined for SDR Mode, along with suggested recovery methods. In addition, a similar set of errors is defined for each HDR Mode.

Note:

This question has been updated for I3C v1.1+ and I3C Basic v1.1.1. These versions add new Error Types CE3 and DBR.

Q23.3 What happens if the Controller crashes during a Read?

An I3C Target may optionally choose to time out if it detects more than 100 µs without an SCL edge (see specification ***Section 5.1.2.3***). If that happens, the Target can abandon the Read and release SDA to avoid a Bus hang when the Controller restarts.

The optional timeout of 100 µs with no SCL activity is a recommended minimum. It does not have to be precise, but since I3C's minimum frequency is 10 KHz, anything longer than 100 µs means that the Controller has failed to complete the Read, so the Target should High-Z the SDA line and abandon the Read.

Note:

If the Target is in Legacy I^2C mode, then it would not normally abandon the read, since there is no minimum frequency, and because 9 clocks by a Controller will be enough to abort the Read (since the 9th bit of data is ACK/NACK for a Read in Legacy I^2C Mode).

Q23.4 Is there any way to exit from an Error of Type TE0 or TE1, other than waiting for an Exit Pattern?

Yes. An I3C Target may optionally watch for 60 µs with no SCL or SDA changes to make sure that the Bus is not in HDR Mode (and therefore must be in SDR Mode). After that, it is appropriate to wait for START (assumed to be Repeated START) or STOP.

Q23.5 Can a Controller issue a STOP condition regardless of whether or not a Target has issued an acknowledgment indicating a completed transaction?

The STOP can be issued anywhere the Target is not driving the SDA during SCL High. It may not be appropriate to do so in terms of completion of a message. But ACK and completed transaction do not belong together in I3C.

Q23.6 What errors are reported on the GETSTATUS Protocol Error bit?

The Protocol Error Report bit on the GETSTATUS CCC is intended to report errors that have no other way of being detected unless the Device reports them. It is intended to cover parity error, CRC error, and anything else that means that a message from the Controller was lost and the Controller has no way of knowing it.

The Protocol Error Report on the GETSTATUS CCC would not cover the case of TE5 errors, as they are more related to an unsupported CCC or Defining Byte (which is not a protocol error *per se*). It is also not intended for situations when the Target NACKs, because the Controller will know that an error occurred when the Target NACKs and recovers it. Errors such as Error Types TE0, TE3, TE4, and TE5 are detected by the Controller when the Target sends a NACK response, and are recovered by using the appropriate recovery methods; as a result, they need not be reported via the GETSTATUS CCC.

Q23.7 What errors does Target Error Type TE5 cover?

Error Type TE5 covers illegally formatted CCCs that an I3C Target might see on the I3C Bus.

Due to confusion around the use of the words “illegally formatted” in the I3C v1.0 specification, I3C v1.1 more precisely defined what errors are considered to be instances of Error Type TE5. This section covers only four cases of errors, as follows.

- The first two cases listed in I3C v1.1 are Unsupported Command Codes and Unsupported Defining Bytes (i.e., for a supported Command Code), both of which are ignored by a Target if not supported.

  Note that these are not strictly “illegally formatted” CCCs *per se* according to v1.1.1’s clarified definition of Error Type TE5, since the CCC format itself might be correct (if the Target happened to support that CCC). Nonetheless, a Target must still NACK any Command Code or Defining Byte that it does not support, so that the Controller will see the NACK and know that the Target is unable to respond to the CCC. In cases where these commands have a special exit condition, the Target should also still wait for the applicable exit condition.
- The two cases addressed by Error Type TE5 are when the Controller sends the wrong RnW Bit for a Direct CCC command. That is, the Controller sends a Dynamic Address and Read bit for a Direct Write or Direct SET CCC command, or vice versa. In these cases, the Target must NACK its Address, thus notifying the Controller that an error has occurred. The Controller will then use the Retry and Escalation models.

Additional Defining Bytes, or Additional Data on CCC Payloads, are not error conditions. Targets should ignore any additional unrecognized data bytes, per the specification at ***Section 5.1.9.2.2***.

Note:

*In previous versions of I3C and I3C Basic, Error Type TE5 was named Error Type S5; see **Q5.2** for name change details.*

Q23.8 Are Devices required to wait for a Repeated START or STOP, or both, to recover from Error Types TE2–TE5?

Error Types TE2 through TE5 should be recovered by STOP.

However:

- **Error Types TE2, TE3, and TE5:** Vendors may optionally elect to have Devices recover from these Error Types upon seeing a Repeated START, per the transaction type.
  - For these Error Types that support optional Repeated START recovery, STOP is the second step of escalation after Repeated START recovery.
  - If Error Types TE2 and TE5 are seen during a Direct CCC, then the Target must retain the CCC state until the Target detects the end of the CCC (i.e., the end of the modality). Since the Controller uses the Direct CCC framing model, the CCC modality will continue until the Controller sends either STOP, or Repeated START with 7’h7E (see specification ***Section 5.1.9.1***).
    - If the Controller simply uses a Repeated START (without 7’h7E) and stays in the Direct CCC framing model, then the Target must remember that the CCC modality is still active, and must treat the subsequent transfer as a Direct CCC, not a Private Read/Write.
    - Note that Broadcast CCCs do not require Repeated START with 7’h7E to end the CCC modality: these end with a simple Repeated START (i.e., no 7’h7E is needed).
    - If I3C Devices are not sure whether Repeated START recovery is appropriate for a situation, then it is safer to ignore subsequent CCCs in the SDR Frame and wait for STOP.
- **Error Type TE4:** Requires STOP recovery (the Repeated START option does not apply).

Q23.9 What has changed regarding Target Error Types in I3C v1.1.1?

Error Type TE0 (formerly named Error Type S0; see specification ***Section 5.1.10.1.1***) now more clearly defines the I3C Target Address restrictions for an I3C Controller, and explains the single-bit error conditions that might occur if the restricted Addresses were used.

Error Type TE5 (formerly named Error Type S5; see ***Section 5.1.10.1.6***) now only provides examples of "illegally formatted" CCCs that include an incorrect RnW bit for a Direct CCC. Error Type TE5 no longer lists unsupported CCCs as an error case (see ***Q23.7***).

> Note that the Target requirements for handling an unsupported Command Code or unsupported Defining Byte are now defined in specification ***Section 5.1.9.2.2***. Although these cases are not strictly covered by Error Type TE5, a Controller might interpret a Target's NACK response similarly and handle it with the same recovery procedure.

Error Type TE6 (formerly named Error Type S6; see ***Section 5.1.10.1.7***) now clearly defines the difference between the Target's response to a CCC that it perceives as a Read (i.e., a Direct GET or a Direct Read) when there is an error in the RnW bit. This better explains the way that the Target should handle the error situation, i.e., when the Controller actually intended to send a Write (i.e., a Direct SET or a Direct Write).

Q23.10 When does the RSTACT CCC state clear in an I3C Target?

The RSTACT state will be cleared back to default upon either:

1. Detection of a Target Reset Pattern. This will enact the RSTACT action, and then clear the state.
2. Detection of a completed START (but not repeated START). The RSTACT may be cleared either on that falling edge, or on the rising edge that follows.

Q23.11 What is the minimal Target Reset support required in I3C v1.1 or v1.1.1?

I3C Targets that comply with I3C v1.1+ or I3C v1.1.1 must support the Target Reset Pattern, and at least the Peripheral Reset action (i.e., RSTACT CCC with Defining Byte 0x01).

Although MIPI Alliance strongly recommends true support of Target Reset, such that a Controller can fully reset a Target chip when needed, it is not required. Full/Chip Reset (see specification ***Section 5.1.11.4***) allows replacement of a dedicated pin that would normally be used to reset a Device.

Note:

> *If supported, a Full/Chip Reset causes a typical I3C Target to return to its power-on configuration, which means re-enabling its I^2C Spike Filter if it has one. If so, then the I3C Controller must tell the I3C Target to turn off its Spike Filter again (see **Q24.1**).*

The minimal reset is a reset of the I3C peripheral, at least to a level that will allow a 'stuck' I3C peripheral to start working again. How much of a reset that requires is up to the Target vendor; for example, it could be handled by an internal interrupt which would allow firmware/software in the Target to handle the reset.

Q23.12 When does a Target escalate Target Reset to Full/Chip Reset?

If the Target does not receive a RSTACT CCC before seeing the Target Reset Pattern, then it uses the default action of Peripheral Reset: it resets just the I3C block. If the Target again receives no RSTACT CCC before seeing a Target Reset, it then escalates to a Full/Chip Reset. It therefore retains the state after any default Peripheral Reset, so it can then activate the escalation. This state is cleared if the Target sees either an RSTACT CCC, or a GETSTATUS CCC addressed to it.

Note:

> *The purpose of this mechanism is to fix a broken system or setup. Because the Target Reset Pattern Detector logic is normally separated both from the I3C block and from other parts of the main system, it will continue to work even if the rest of the chip becomes broken (e.g., clocks stopped, Bus locked up, etc.). This means that not seeing a RSTACT CCC would be a symptom of a large problem, so the Target escalates in order to recover from such a broken condition. The Target first performs a Peripheral Reset, in case the fault condition exists only in the I3C block itself.*

Q23.13 How is Target escalation affected when the RSTACT CCC is received?

The RSTACT CCC only determines how the Target will interpret the next Target Reset Pattern that it receives, the RSTACT CCC does not alter the handling of any currently in-progress Target escalation.

For this reason:

- A Device that supports the RSTACT CCC should always prioritize RSTACT configuration over any currently occurring Target escalation.
- Any Target escalation operation that might be in progress when the RSTACT CCC is received should be cleared.

2.24 Timing Parameters

Q24.1 Are there any special timing requirements for sending the first START with the Broadcast Address?

Yes. The I3C Controller must emit the first START with the Broadcast Address (7'h7E) at Open-Drain speeds (i.e., usually I^2C Fm+ timings) so that I3C Targets with I^2C Spike Filters (per ***Q15.4***) will be able to see the I3C Broadcast Address, and then as a result turn off the Spike Filter (per the specification at ***Section 5.1.2.1.1***).

Note:

If another I3C Target arbitrates its own Address (or the special reserved address for a Hot-Join Request) into this Address Header and thereby wins arbitration, then the I3C Controller must repeat this special Address Header.

*If such an I3C Target supports Full/Chip Reset using the Target Reset Pattern (which might be preceded by the RSTACT CCC; see specification **Section 5.1.11.4**), then it will most likely re-enable its I^2C Spike Filter after such a Full/Chip Reset. In this case, the I3C Controller must emit the START condition with the Broadcast Address (7'h7E) at Open-Drain speeds again, so that the I3C Target can turn off the Spike Filter.*

However, if the Controller knows that none of the I3C Targets has a Spike Filter, then it may omit this. Similarly, if the Controller knows that all of the I3C Targets have accurate Spike Filters, then it may use the I3C 2.5 MHz Open-Drain speed (i.e., 200 ns Low, 200 ns High).

Q24.2 What is the I3C Open-Drain t_{High} Max? Table 10 shows it as 41 ns, but a Note says it may be longer

In the I3C v1.1 specification, ***Table 110 I3C Open Draining Timing Parameters*** shows the t_{High} symbol maximum value as 41 ns, and Note 4 states "t_{High} may be exceeded when the signals can be safely seen by I^2C Targets". This means that a 41 ns t_{High} will ensure that no Legacy I^2C Target will see the SCL changes, and so no Legacy I^2C Target will interpret the START followed by the Address phase nor the ACK/NACK. If there is no harm with a Legacy I^2C Target on the I3C Bus seeing the clocks, then the Controller may use a much longer t_{High}. For example, Legacy I^2C Targets will see the STOP and then a START. It is acceptable for them to see the Address that follows (arbitrated or not), since none of those Addresses will match their own Address. However, as the t_{Low} may well be 200 ns, this may present problems for any Legacy I^2C Targets not fast enough to correctly handle a 200 ns Low period.

In the I3C v1.1.1 specification, the parameters are defined in the equivalent ***Table 122***. In the I3C Basic v1.1.1 specification, the parameters are defined in the equivalent ***Table 86***.

Q24.3 How should t_{SCO} timing be interpreted?

This is extensively covered in the I3C v1.1+ and I3C Basic v1.1.1 specifications at ***Section 5.1.9.3.18***.

Q24.4 If a Device has a t_{SCO} value greater than 12 ns, does that mean it doesn't qualify as an I3C Device?

Note:

This question does not apply to I3C Basic v1.0.

No. The t_{SCO} (Clock-to-Data Turnaround delay time) parameter is information provided by Target Devices so that system designers can properly compute the maximum effective frequency for reads on the Bus. The t_{SCO} number is meant to be used together with the line capacitance (trace length) and number of Targets and stubs (if present).

However, I3C Targets with t_{SCO} delay greater than 12 ns must do all of the following:

- Set the Limitation bit in the Bus Characteristics Register (BCR) to 1'b1
- Set the Clock-to-Data Turnaround field of the maxRD Byte to 3'b111
- Communicate the t_{SCO} value to the Controller by private agreement (i.e., product datasheet)

Note:

I3C Basic v1.0 and I3C v1.1+ already clarify these aspects of communications in I3C.

Q24.5 How do t_{CBSr} and t_{CASr} timing differ between I3C v1.1 and I3C v1.0?

In I3C v1.0, parameter t_{CASr}'s minimum value was t_{CASmin}. In I3C v1.1, this was reduced to $t_{CASmin/2}$. Since satisfying this reduced duration might be challenging for some Targets, the Controller is required to provide suitable timing, i.e., is required to accommodate the slowest Devices on the Bus.

In version 1.1 of the I3C specification, a new Note clarifying this point was added to ***Table 111 I3C Push-Pull Timing Parameters for SDR, ML, HDR-DDR, and HDR-BT Modes*** (in ***Section 6.2***):

9) Targets with speed limitations inform the Controller via the Bus Characteristics Register (BCR) that the minimum may 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 I²C Devices that might see it.

In the I3C v1.1.1 specification, this Note appears in the equivalent ***Table 123***. In the I3C Basic v1.1.1 specification, it appears in the equivalent ***Table 87***.

Q24.6 Are there any special timing parameters for sending the HDR Exit Pattern, HDR Restart Pattern or Target Reset Pattern?

Yes. The I3C Controller must always observe the minimum timing requirements for all of these defined patterns. ***Section 6.2*** defines the minimum value of parameter $\mathbf{t_{DIG\_H}}$ for such transitions, and this is mentioned explicitly in ***Section 5.2.1.1.1*** (i.e., for the HDR Exit Pattern).

Since the HDR Restart Pattern and the Target Reset Pattern are both based on the HDR Exit Pattern, the same timing parameters apply to all patterns. Note that the Controller may send all such patterns at slower clock speeds if needed, as long as each transition on SDA and/or SCL observes the minimum value parameter $\mathbf{t_{DIG\_H}}$.

3 Terminology

See also ***Section 2*** in the MIPI I3C Specification ***[MIPI01][MIPI09][MIPI11]***.

3.1 Definitions

Bus Available: I3C Bus condition in which a Device is able to initiate a transaction on the Bus.

Bus Free: I3C Bus condition after a STOP and before a START with a duration of at least t_{CAS}.

Bus Idle: An extended duration of the Bus Free condition, during which Devices may attempt to Hot-Join the I3C Bus.

Controller: The I3C Bus Device that is controlling the Bus. (I3C and I3C Basic versions prior to v1.1.1 used the deprecated term Master.)

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

Hot-Join: Targets that join the I3C 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.

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.

Master: Deprecated term used in I3C and I3C Basic versions prior to v1.1.1. See Controller.

Primary Controller: Controller-capable Device that has initial control of the I3C Bus. Formerly called “Main Master”.

Slave: Deprecated term used in I3C and I3C Basic versions prior to v1.1.1. See Target.

Target: An I3C Target Device can only respond to either Common or individual commands from a Controller. (I3C and I3C Basic versions prior to v1.1.1 used the deprecated term Slave.)

3.2 Abbreviations

ACK	Short for “acknowledge” (an I3C Bus operation)
DisCo	Discovery and Configuration (family of MIPI Alliance interface specifications)
e.g.	For example (Latin: exempli gratia)
i.e.	That is (Latin: id est)

3.3 Acronyms

See also the acronyms defined in the MIPI I3C Specification ***[MIPI01][MIPI09][MIPI11]***.

CCC	Common Command Code (an I3C common command or its unique code number)
CTS	Conformance Test Suite
DAA	Dynamic Address Assignment (an I3C Bus operation)
FAQ	Frequently Asked Questions
HCI	Host Controller Interface (a MIPI Alliance interface specification ***[MIPI02][MIPI12]***)
HDR	High Data Rate (a set of I3C Bus Modes)
HDR-BT	HDR Bulk Transfer (an I3C Bus Mode)
HDR-DDR	HDR Double Data Rate (an I3C Bus Mode)
HDR-TSL	HDR Ternary Symbol Legacy (an I3C Bus Mode)
HDR-TSP	HDR Ternary Symbol for Pure Bus (an I3C Bus Mode)
I3C	Improved Inter Integrated Circuit (a MIPI Alliance interface specification ***[MIPI01][MIPI09][MIPI11]***)
IBI	In-Band Interrupt (an I3C Bus feature)
ML	Multi-Lane (an I3C Bus feature, and set of Data Transfer Codings for I3C Bus Modes)
ODR	Output Data Rate
SCL	Serial Clock (an I3C Bus line)
SDA	Serial Data (an I3C Bus line)
SDR	Single Data Rate (an I3C Bus Mode)
SPI	Serial Peripheral Interface (an interface specification)

4 References

[MIPI01] *MIPI Alliance Specification for I3C® (Improved Inter Integrated Circuit)*, version 1.0, MIPI Alliance, Inc., 23 December 2016 (Adopted 31 December 2016).

[MIPI02] *MIPI Alliance Specification for I3C Host Controller Interface (I3C HCI℠)*, version 1.0, MIPI Alliance, Inc., 29 September 2017 (Adopted 4 April 2018).

[MIPI03] *MIPI Alliance Specification for Discovery and Configuration (DisCo℠)*, version 1.0, MIPI Alliance, Inc., 1 July 2016 (Adopted 28 December 2016).

[MIPI04] *MIPI Alliance DisCo℠ Specification for I3C℠*, version 1.0, MIPI Alliance, Inc., 25 January 2019 (Adopted 18 June 2019).

[MIPI05] *MIPI Alliance I3C Application Note: General Topics*, App Note version 1.1, MIPI Alliance, Inc., In press.

[MIPI06] *MIPI Alliance Specification for Camera Serial Interface 2 (CSI-2℠)*, version 4.0, MIPI Alliance, Inc., 26 September 2021 (Adopted 8 December 2021).

[MIPI07] *MIPI Alliance Specification for Debug for I3C℠*, version 1.0, MIPI Alliance, Inc., 21 April 2020 (Adopted 4 September 2020).

[MIPI08] *MIPI Alliance Conformance Test Suite (CTS) for I3C℠ v1.1.1 and I3C Basic v1.1.1*, CTS version 1.0, MIPI Alliance, Inc., 4 August 2021 (approved 5 August 2021).

[MIPI09] *MIPI Alliance Specification for I3C Basic℠ (Improved Inter Integrated Circuit)*, version 1.0, MIPI Alliance, Inc., 19 July 2018 (Adopted 8 October 2018).

[MIPI10] MIPI Alliance, Inc., "I3C SETBUSCON Table", <https://www.mipi.org/MIPI_I3C_bus_context_byte_values_public.html>, last accessed 22 August 2022.

[MIPI11] *MIPI Alliance Specification for I3C® (Improved Inter Integrated Circuit)*, version 1.1, MIPI Alliance, Inc., 27 November 2019 (Adopted 11 December 2019).

[MIPI12] *MIPI Alliance Specification for I3C Host Controller Interface (I3C HCI℠)*, version 1.1, MIPI Alliance, Inc., 20 May 2021 (Adopted 20 May 2021).

[MIPI13] *MIPI Alliance Specification for I3C® (Improved Inter Integrated Circuit)*, version 1.1.1, MIPI Alliance, Inc., 11 June 2021 (Adopted 8 June 2021).

[MIPI14] *MIPI Alliance Specification for I3C Basic℠ (Improved Inter Integrated Circuit)*, version 1.1.1, MIPI Alliance, Inc., 9 June 2021 (Adopted 21 July2021).

Note:

Version number v1.1 was not used for I3C Basic.

[MIPI15] *MIPI Alliance Specification for Debug for I3C℠*, version 1.1, MIPI Alliance, Inc., In press.

[LINX01] Linux Kernel Patches for I3C subsystem, <https://patchwork.kernel.org/project/linux-i3c/list/>, last accessed 22 August 2022.