Specification for
I3C Transfer Command Response Interface
(MIPI I3C TCRI^{SM})

Version 1.0

24 May 2022

MIPI Board Adopted 7 September 2022

Public Release Edition

Further technical changes to this document are expected as work continues in the Software Working Group.

Contents

Figures

Tables

Release History

Date	Version	Description
07-Sep-2022	v1.0	Initial Board adopted release.

This page intentionally left blank.

1 Introduction

The proliferation of sensors in mobile wireless and mobile-influenced products has created significant design challenges. The number of sensors in platforms increases each year, and sensors are becoming crucial for enabling new use cases and for platform power management.

The standardization of interfaces and common software support allow for more reliable hardware and software. In platform design, the use of common components allows designers to focus efforts on actual sensor applications, rather than the interfaces.

If no consistent method for interfacing to I3C were available, then every platform vendor would be faced with designing and enabling their own interface to support I3C. In addition to the main interface other signals may be needed, such as dedicated interrupts, chip select signals, and enable and sleep signals. This increases the required number of Host GPIOs, and that in turn drives up system cost with more Host package pins and more PCB layers. As time passes and the number of sensors increases, this becomes increasingly difficult to support and manage.

The MIPI I3C interface has been developed to ease sensor system design architectures in mobile wireless products by providing a fast, low cost, low power, two-wire digital interface for sensors. I3C is compatible with existing Legacy I^{2}C Devices, with feature limitations. I3C defines the idea of Controller Devices and Target Devices (i.e., I3C Devices with the role of Controller and/or Target). I3C also allows for multiple Controller-capable Devices (i.e., one Primary Controller and optionally one or more Secondary Controllers) in the Bus topology.

This I3C TCRI Specification describes the Transfer Command/Response Interface exposed by hardware implementing I3C Controller Devices, i.e., the I3C Controller Interface for driving transfers on an I3C Bus with the MIPI I3C protocol ***[MIPI02]***.

1.1 Scope

This Specification contains sufficient detail to develop a compliant lower-level (i.e., peripheral) layer for any device that includes an I3C Controller, for use as part of an Application that can send Transfer Commands and receive Transfer Responses using the defined formats in this Specification. It includes an Architectural Overview, the Theory of Operation, and definitions of the data structures that define the I3C Controller's interface with its Application. However, this Specification alone does not provide sufficient detail to develop the entire Application, which may have additional capabilities beyond this Transfer Command/Response Interface.

The reader is assumed to be familiar with the I3C Specification ***[MIPI02]***. Concepts described in the I3C Specification will not be repeated here, except where necessary for proper understanding.

1.2 Purpose

The purpose of this Specification is to standardize the Transfer Command and Transfer Response interface for I3C Controller hardware, which can be integrated into any Application. This allows multiple I3C Applications to use the same data structures for interfacing with I3C Controller hardware.

The target audience of the document are developers of Active Controller (I3C Controller) hardware, and developers of I3C Applications.

2 Terminology

2.1 Use of Special Terms

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

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

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

2.2 Definitions

***a priori*:** Knowledge resulting from theoretical deduction, not from observation or experience (Latin).

ACK: Short for “acknowledge”. See also NACK.

Active Controller: The I3C Device that presently has Controller control of the I3C Bus.

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

Address: A set of bits designating a Device.

Application: A Host or controlling entity that implements this I3C Transfer Command/Response Specification and provides I3C Controller functionality.

Arbitrable: Subject to decision by Arbitration.

Arbitration: If two Devices start transmission at the same time, then Arbitration is required to determine Bus control. Arbitration could also be required during a Target transmission if a Controller addresses multiple Target Devices. Arbitration is required when a Controller sends an address and when a Target sends an In-Band Interrupt.

Atomic (Relating to an operation or transaction)**:** A sequence comprising a set of smaller operations, provided in a particular order, which must either be executed fully from start to finish without interruption; executed in part (if applicable to the sequence), stopping only at designated safe points of interruption, and not allowed to continue after any such interruption, due to unintended consequences that would induce behavior different from that which would be induced by continuous execution; or cancelled entirely after any such interruption. In cases of an interruption, any subsequent smaller operations that might not have been executed after the interruption (e.g., when some smaller operations were not yet received before the determination of the end of the sequence was known) must either be nullified or not executed.

Broadcast: Refers to an Address or command that is transmitted to multiple Target Devices.

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

Combo: A two phased transfer operation, typically consisting of the combination of two transfers as a single unit. Usually structured as either a Write + Write operation, or a Write + Read operation. The transfers for these phases might be separated by specific framing specific to the I3C Mode(s) used for each phase. Typically, the first phase is a shorter transfer which informs a Device to prepare for the second phase.

Command Descriptor: Structure used to define I3C Command, CCC or Transfer.

Common Command Codes (CCC): Globally supported commands to be transmitted either directly to a specific I3C Target Device or to all I3C Target Devices simultaneously.

Controller: A reference to the I3C Bus Device that is controlling the Bus.

Controller Logic: An internal component comprising dedicated logic and internal transfer mechanisms, integrated within the I3C Controller, that acts as the Controller of the I3C Bus in Active Controller mode. Typically contains internal FIFOs, clock logic, status, and other interfaces under the control of the I3C Controller to process Transfer Commands from the Command Queue, handle Interrupt Requests and perform other necessary tasks in the role of Active Controller.

Controller-role: Control of the I3C Bus, in an Active Controller role.

Device: A Controller or Target.

Device Address Table: Table that contains information on Target Devices addressable on the I3C Bus.

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

DWORD: A 32-bit data word.

Dynamic Address: A Device Address that is assigned or allocated during initialization of the I3C Bus, or subsequently as needed. Usually occurs after power up.

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

HCI: Host Controller Interface that implements this I3C Transfer Command/Response Interface and also meets the requirements of the I3C Host Controller Interface Specification ***[MIPI05]***.

HDR-DDR: HDR Double Data Rate

HDR-BT: HDR Bulk Transfer

High Data Rate (HDR): High Data Rate modes that achieve higher speed by transferring data on both clock edges.

High: A signal level of logical "1".

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

In-Band Interrupt: Interrupt from Target Device on I3C Bus without a separate pin connection.

I^{2}C Device: A Controller or Target that meets the requirements of the I^{2}C Specification ***[NXP01]***.

I3C Bus: The physical and logical implementation of the SCL and SDA lines according to I3C Specification ***[MIPI02]***.

I3C Device: A Controller or Target that meets the requirements of the I3C Specification ***[MIPI02]***.

I3C Bus Controller Logic: See Controller Logic.

I3C Target: See Target.

Legacy I^{2}C: I3C maintains the industry standard architecture of I^{2}C and supports existing I^{2}C Target Devices. I3C does not support I^{2}C Bus Controllers.

Low: A signal level of logical "0".

Message: A packetized communication between Devices.

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

Mode: I3C defined data transfer methods, namely: Legacy I^{2}C Mode; Single Data Rate Mode (SDR); and various High Data Rate (HDR) Modes including Dual Data Rate Mode (HDR-DDR), Bulk Transfer Mode (HDR-BT), Ternary Symbol Legacy Mode (HDR-TSL), and Ternary Symbol for Pure Bus Mode (HDR-TSP).

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

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

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

Primary Controller: Controller that has overall control of the I3C Bus, including control and handoff to Secondary Controllers.

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

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

Read-Type Transfer: An operation, typically having a single phase, in which a Device on the I3C Bus is addressed and then required to either acknowledge or refuse a read transfer, based on transfer parameters sent on the I3C Bus. If the Device acknowledges the transfer, then it shall immediately drive data bytes on the data line (i.e., SDA) which are received by the I3C Controller and made available for the Host to read.

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

Response Descriptor: Structure used to report Command or Transfer status

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

Secondary Controller: Controller-capable I3C Device that controls the I3C Bus only after receiving permission (i.e., after accepting Controller-role) from the Primary Controller. Control of the Bus might be temporary; if so, such a Device typically passes Controller-role back to either the Primary Controller, or to another Controller-capable Device.

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

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

START: START is the I3C Bus condition of a HIGH to LOW transition on the SDA line while the SCL line remains HIGH. In this Specification, a START is abbreviated as "S".

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

STOP: STOP is the I3C Bus condition of a LOW to HIGH transition on the SDA line while the SCL line remains HIGH. In this Specification, a STOP is abbreviated as "P".

Synchronization: Coordination of events to operate a system in unison.

Target: An I3C Device that can only respond to either Common or individual commands from a Controller. A Target Device cannot typically generate a clock.

Ternary Mode: A term used as reference to I3C HDR-TSP (Ternary Symbol for Pure Bus) or HDR-TSL (Ternary Symbol Legacy) Modes.

Version 1.1+ of the I3C Specification: See ***[MIPI06]***.

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

Write-Type Transfer: An operation, typically having a single phase, in which a Device on the I3C Bus is addressed and then required to either acknowledge or refuse a write transfer, based on transfer parameters sent on the I3C Bus. If the Device acknowledges the transfer, it shall then receive data bytes on the data line (i.e., SDA) which are driven by the I3C Controller.

2.3 Abbreviations

e.g.	For example (Latin: exempli gratia)
i.e.	That is (Latin: id est)
aka.	Also known as

2.4 Acronyms

ACK	Acknowledge
BCR	Bus Characteristics Register
CCC	Common Command Code
CRC	Cyclic Redundancy Check
CRR	Controller-role Request
DAT	Device Address Table
DCR	Device Characteristics Register
DDR	Double Data Rate
FSM	Finite State Machine
HCI	Host Controller Interface
HDR	High Data Rate
HDR-BT	HDR Bulk Transfer
HDR-DDR	HDR Double Data Rate
HJ	Hot-Join
IBI	In-Band Interrupt
MDB	Mandatory Data Byte
MHz	Mega Hertz
MID	MIPI Manufacturer ID ***[MIPI01]***
NACK	Not Acknowledge
P	STOP
S	START
SCL	Serial Clock
SDA	Serial Data
SDR	Single Data Rate
Sr	Repeated START
T	Transition Bit

TCRI	Transfer Command/Response Interface for I3C
TSL	Ternary Symbol Legacy
TSP	Ternary Symbol for Pure Bus (no I^{2}C Devices)

2.5 Color Coding

In some Figures in this Specification, color coding is used to indicate ownership. This includes ownership of state in FSMs and other modal states. Items driven or set by the Application are indicated with yellow, and items driven or set by the I3C Bus Controller are indicated with gray.

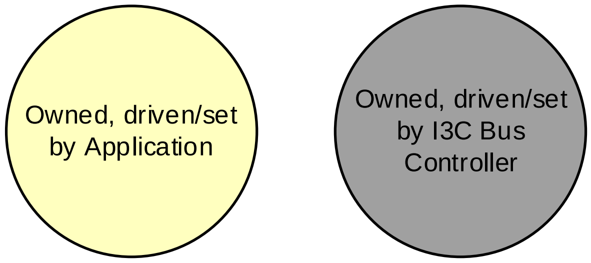
Owned, driven/set by Application
Owned, driven/set by I3C Bus Controller

Figure 1 Color Coding Scheme

3 References

[MIPI01]	MIPI Alliance, Inc., "MIPI Alliance Manufacturer ID Page", <http://mid.mipi.org>, last accessed 12 September 2022.
[MIPI02]	*Specification for Improved Inter Integrated Circuit (I3C®)*, Version 1.1.1 (including all released Errata), MIPI Alliance, Inc., 20 May 2021 (MIPI Board Adopted 20 May 2021).
[MIPI06]	Version 1.1+ of the MIPI I3C Specification.

Note:

In this Specification, the term "Version 1.1+ of the I3C Specification" refers to the most recently adopted MIPI I3C v1.1-based Specifications. At the time this I3C TCRI v1.0 Specification was adopted, this was the MIPI I3C v1.1.1 Specification (available to MIPI Alliance member companies only) and the MIPI I3C Basic v1.1.1 Specification (publicly available).

[MIPI03]	*I3C Application Note: General Topics (Applies to I3C v1.1+ and I3C Basic v1.1.1+)*, App Note version 1.1, MIPI Alliance, Inc., 27 April 2022 (MIPI Board approved 27 July 2022).
[MIPI04]	*Discovery and Configuration (DisCo^{SM}) Specification for I3C*, Version 1.0, MIPI Alliance, Inc., 23 January 2019 (MIPI Board Adopted 18 June 2019).
[MIPI05]	*MIPI Alliance Specification for I3C Host Controller Interface (I3C HCI^{SM})*, version 1.1 (including all released Errata), MIPI Alliance, Inc., 20 May 2021 (MIPI Board Adopted 20 May 2021).
[NXP01]	UM10204, *I^{2}C Bus Specification and User Manual*, Rev. 6, NXP Corporation N.V., 4 April 2014.
[PCISIG01]	PCI SIG, *PCI Code and ID Assignment Specification*, <https://pcisig.com/specifications>, Revision 1.9, 31 May 2017.
[USB01]	*Universal Serial Bus Specification*, <https://www.usb.org/document-library/usb-20-specification>, Revision 2.0 (including errata and ECNs through 18 December 2018), USB-IF, 27 June 27 2017.
[USB02]	*Universal Serial Bus 3.2 Specification*, <https://www.usb.org/document-library/usb-32-specification-released-september-22-2017-and-ecns>, Revision 1.0 (including errata and ECNs through 24 July 2018), USB-IF, 22 September 2017.
[USB03]	*Universal Serial Bus I3C Device Class Specification*, Revision 1.0, USB-IF, 7 January 2022.

4 Technical Overview

The MIPI I3C Specification ***[MIPI02]*** defines the behavior of the Devices on an I3C Bus. This ensures Device compatibility and interoperability. However, hardware manufacturers and platform integrators have the challenge of designing and integrating I3C Bus Controller Logic as part of a subsystem that can connect to a Host through a system bus.

To promote interoperability, MIPI has defined this I3C Transfer Command/Response Interface (I3C TCRI) to allow for easier adoption of I3C Controller capabilities into various Applications, in order to serve multiple ecosystems. This enables the development of common Application layer implementations, while also allowing for vendor-specific innovation. This Specification also defines the high-level architecture of such an I3C Controller subsystem, as shown in ***Figure 2***.

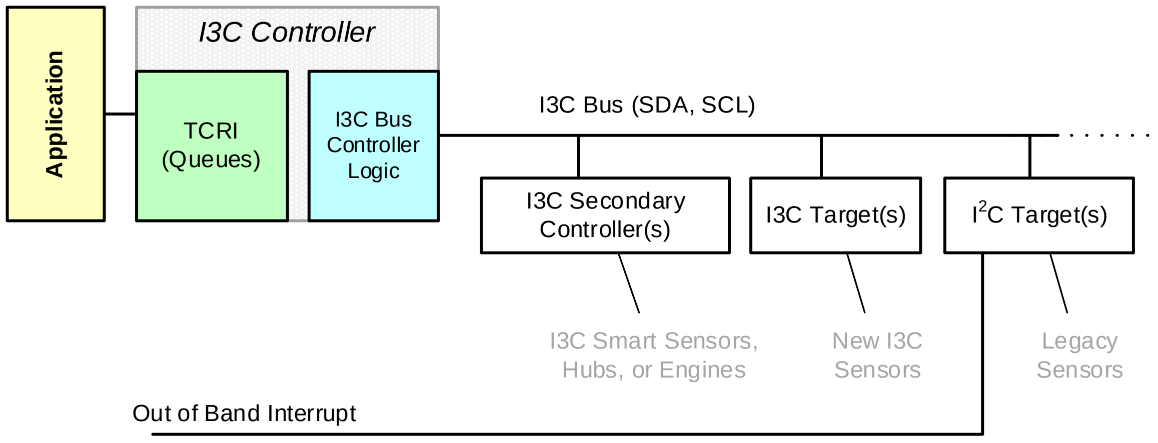

Figure 2 I3C System Overview

Note:

*Other capabilities of an instance of I3C Bus Controller Logic (i.e., capabilities not related to Transfer Commands and Responses) might require additional signals or interfaces that would be presented to the Application. Such aspects are not defined in this Specification. See **Section 4.1** and **Section 4.2** for additional context.*

4.1 Scope

This Specification defines the Application interface exposed by I3C Controller hardware, that allows for implementation of I3C Specification features in a standardized, predictable, and resource-efficient manner. This interface includes I3C Transfer Commands and I3C Transfer Responses, including the expected hardware response and behavior for specific actions. This Specification also defines the behavior and expectations required of an I3C Controller that processes single or multiple I3C Transfer Commands in an ordered sequence, and generates I3C Transfer Responses for such a sequence of I3C Transfer Commands that are either processed or not processed (i.e., due to interruption or transfer errors).

However, this Specification does not define the internal implementation of I3C Bus Controller Logic, nor does it define other interface details or additional capabilities that might be required for a particular Application or for optional extensions that might be chosen by the implementer. This Specification also does not define Application-specific hardware requirements beyond the scope of processing I3C Transfer Commands that are enqueued by the Application, and generating I3C Transfer Responses for such I3C Transfer Commands. A particular Application could extend this interface per the specific use case.

In particular, several aspects of I3C Controller behavior relating to I3C Transfer Commands and I3C Transfer Responses are **not** defined, such as:

- The specific method that a networked Application should use to establish an active session for subsequent communications with the I3C Controller
- The specific method that an Application should use to enqueue I3C Transfer Commands to the I3C Controller
- The channel or method that is used to send Write Data to the I3C Controller, for I3C Transfer Commands that are Write-type transfers (e.g., Private Writes, Broadcast CCCs, Direct Write, or Direct SET CCCs)
- The specific method that an Application should use to dequeue I3C Transfer Responses
- The channel or method that is used to retrieve Read Data from the I3C Controller, for I3C Transfer Responses that are generated for Read-type transfers (e.g., Private Reads, Direct Read, or Direct GET CCCs)
- The channel or method that the I3C Controller uses to notify the Application of I3C In-Band Interrupts, Hot-Join Requests, Controller Role Requests, or other I3C Bus events that are not listed here
- Means of configuring the I3C Controller or adjusting its specific parameters that govern timing, transfer rates, or methods of responding to I3C Bus events that are not defined in this Specification
- Methods of interrupting an I3C transfer that is currently in progress (if supported for that I3C Mode)
- Assignment of I3C Dynamic Addresses to any I3C Target Devices on the I3C Bus
- How to invoke any I3C-defined error recovery methods (if such a situation becomes necessary, per the I3C Specification)
- Additional functionality that might automatically respond to certain I3C Bus events, or particular transfers on the I3C Bus based on data received, transfer status or failure
- Additional functionality that might perform periodic actions without receiving directly-enqueued I3C Transfer Commands from the Application
- Multiplexing Command/Response processing among two or more Applications, or among multiple Command/Response streams (i.e., separate execution contexts) offered by the same Application
- Any internal architectural requirements or system-level details of the Application

This I3C TCRI Specification does not define system-specific or platform-specific setup for the Application. However, it does require certain capabilities that an Application must adhere to, in order to make use of the standard Transfer Command and Transfer Response flows defined in this Specification.

On its own, this I3C TCRI Specification does not provide a complete implementation of all I3C Controller capabilities that an Application might need. Readers are advised to read this I3C TCRI Specification in conjunction with an Application-specific document, such as the MIPI I3C Host Controller Interface Specification (I3C HCI, see ***[MIPI05]***) which includes additional definitions, normative behaviors, and expectations for a local Host Controller Interface.

4.2 I3C TCRI Purpose

The I3C Transfer Command/Response Interface is intended to standardize the Command/Response interface (which handles Host-initiated transactions such as reads, writes, and CCCs) for I3C Bus Controller logic within any Application, including local Host system peripherals, network bridge devices, and other integrations that use I3C Device functionality that uses the I3C Controller Role.

This Specification is intended to be used for several purposes, including:

- Defining the I3C Transfer Command/Response behavior used by hardware that is compliant with the MIPI I3C HCI Specification ***[MIPI05]***, when processing I3C Transfer Commands sent from a locally connected Host and sending I3C Transfer Responses for such I3C Transfer Commands that are processed to the Host; and
- Defining the I3C Transfer Command/Response behavior for an I3C network Bridge Device that exchanges structured network packets through an established session with a remote entity (i.e., another device on the I3C network), processes I3C Transfer Commands, and generates I3C Transfer Responses for such I3C Transfer Commands that are processed to the remote entity:
  - Suitable for MIPI A-PHY Bridging applications, where the remote entity is an A-PHY Device;
  - Suitable for I3C Routing Devices, where the remote entity is another I3C Controller on an “upstream” I3C Bus segment; or
  - Suitable for other network applications, where the remote entity is another networked device.

Naturally, per the I3C Specification, this I3C Controller Interface supports Transfer Commands that can be sent to either:

- I3C Devices (i.e., I3C Targets); and/or
- Legacy I^{2}C Devices, with compatibility as described in I3C Specification.

This I3C TCRI Specification does not limit the classes or capabilities delivered by any Devices enumerated on an I3C Bus, so long as such Devices conform to the I3C Specification.

Implementing the I3C TCRI Specification can reduce complexity for Application integrators developing solutions that make use of existing components. For example, vendors of standardized systems that integrate I3C Controller functionality can use the Transfer Command/Response Interface within their system designs as part of their Application.

4.3 I3C TCRI Key Features

The I3C TCRI Specification provides efficient means for Applications to interface to the features provided by the I3C Bus, and ensures power-efficient operation of the I3C Controller. Power efficiency is particularly important, as many I3C Device implementations typically target battery-powered environments.

In particular, the I3C TCRI supports the following I3C Specification features:

- Two-wire serial interface up to 12.5 MHz using Push-Pull with the following Data Rates supported:
  - I^{2}C compliant Data Rates:
    - I^{2}C Fast Mode (FM): 0 to 400 Kb/s
    - I^{2}C Fast Mode Plus (FM+): 0 to 1Mbps
  - Single Data Rate (SDR): I3C enhanced version of the I^{2}C protocol, running up to 12.5 MHz:
    - I3C Coding SDR with Directed and Broadcast Common Command Codes (CCC)
  - Optional High Data Rate (HDR) Modes: Additional I3C Modes, not available for the I^{2}C protocol Devices:
    - HDR-Dual Data Rate (HDR-DDR)
    - HDR-Ternary Symbol Legacy Mode (HDR-TSL)
    - HDR-Ternary Symbol for Pure Bus Mode (HDR-TSP)
- Supported Bus Roles:
  - I3C Primary Controller as the initial Active Controller
  - I3C Secondary Controller that can acquire the Controller Role and become Active Controller
- Legacy I^{2}C Target Device co-existence on the same Bus instance (with limitations as described in the I3C Specification ***[MIPI02]***)
- Legacy I^{2}C messaging
- Multi-Drop capability
- Common Command Code (CCC) Framing, namely:
  - CCCs (Direct and Broadcast) in SDR Mode only

4.4 I3C TCRI Fundamental Principles

The I3C TCRI is designed to support the full functionality defined in the I3C Bus Specification ***[MIPI02]***, as well as some addition features that are useful for modern platforms.

The fundamental principles for I3C TCRI are to:

- Provide a well-defined, comprehensive Application interface with full support for all transaction types in supported I3C Modes
- Be designed to support a variety of Applications, including local Host peripherals and remote network bridging solutions
- Provide Device operation flows, including power management aspects
- Use a Command/Response flow that defines various data structure formats (called Descriptors) for transaction flows:
  - This allows for simple transfers, but also enables more complex multi-part transactions
  - It also supports selected optional HDR Modes, and provides a path for future extensibility as the I3C Specification evolves
- Enable support for essential basic features of the I3C Specification version 1.1.1 ***[MIPI02]***

Note:

Additional features not defined in this I3C TCRI Specification are expected to be provided by the Application.

This Specification is written to cover the following capabilities:

- Transaction capability:
  - Capability to communicate to the same Device at multiple supported clock speeds, on a per-transaction basis
  - Capability to send multiple Transfer Commands that are processed and executed as I3C transactions in succession, within the same SDR Frame (i.e., without intervening STOP condition) or the same HDR Frame (i.e., without HDR Exit Pattern)
  - Capability to send Broadcast CCC and Direct CCC transfers using managed CCC transfer framing:
    - Includes support for all variants of Direct CCC transfers, such as Direct Write/SET, Direct Read/GET, and any valid combinations of the above
    - May address one or multiple Target Devices using Dynamic Addresses; may also send to Group Addresses (Direct Write/SET only)
  - Support I3C Pure Bus and Mixed Bus configurations

4.5 I3C TCRI Relationship to Other MIPI Specifications

This Specification describes a set of required behaviors and expectations for I3C Bus Controller Logic that functions as an I3C Controller Device, as defined by v1.1+ of the MIPI I3C Specification ***[MIPI06]***. Implementers are required to support the mandatory I3C Controller behaviors as well as transfers using SDR Mode. Implementers may also choose to support other optional features defined in the MIPI I3C Specification, such as transfers using specific HDR Modes.

A major purpose of this I3C TCRI Specification is to serve as the normative specification for Transfer Commands and Transfer Responses for MIPI's future I3C-related specifications, including future versions of the MIPI I3C HCI Specification ***[MIPI05]***. Sections of this I3C TCRI Specification were originally drafted for, and incorporated into, earlier versions of the I3C HCI Specification.

- Many of the normative behaviors defined in ***Section 6.2***, ***Section 6.3***, and ***Section 6.4*** were also defined in version 1.1 of the I3C HCI Specification.
- Format 1 of the Command Descriptor and Response Descriptor (see ***Section 7.1***) matches the formats that were defined in version 1.1 of the I3C HCI Specification, and provides a comprehensive interface for managing I3C transactions that use the essential (i.e., required) capabilities of the I3C Specification, as well as some optional modes and capabilities.
- By contrast, Format 2 of the Command Descriptor and Response Descriptor (see ***Section 7.2***) is an alternative to Format 1 that was originally developed for earlier drafts of the v1.1 I3C HCI Specification. Format 2 was not included in the adopted version of I3C HCI v1.1, but is now included here in this I3C TCRI Specification, to allow an implementer to choose the option that is best for the particular Application.

The I3C HCI Specification also defines additional aspects, features, and capabilities that are specific to a Host system bus connection as the specific type of Application. As such, the I3C HCI Specification defines additional data structures and behavioral flows for reporting I3C In-Band Interrupts (IBIs) and assigning Dynamic Addresses to I3C Target Devices, along with specific operating modes that are optimized for the defined Host system interface (i.e., the I3C Controller's specific system bus and the manner in which a Host interfaces with the Host Controller).

However, implementers of this I3C TCRI Specification are not limited by what the I3C HCI Specification defines, and this I3C Transfer Command and Transfer Response interface can be used for any Application.

5 Architectural Overview (informative)

5.1 Transfer Command/Response Interface Architecture

This Specification defines the I3C Transfer Command/Response Interface, a set of standard conventions and behaviors for an I3C Controller, including the semantics that an **Application** (i.e., a Host or controlling entity) uses when interfacing with I3C Bus Controller Logic, in order to enqueue I3C Transfer Commands and dequeue I3C Transfer Responses based on I3C Bus activity.

Figure 3 shows a high-level example of an I3C Controller that implements support for I3C Transfer Command and I3C Transfer Response processing from its Application. This figure does not show Application-specific functions that would be required for a complete implementation.

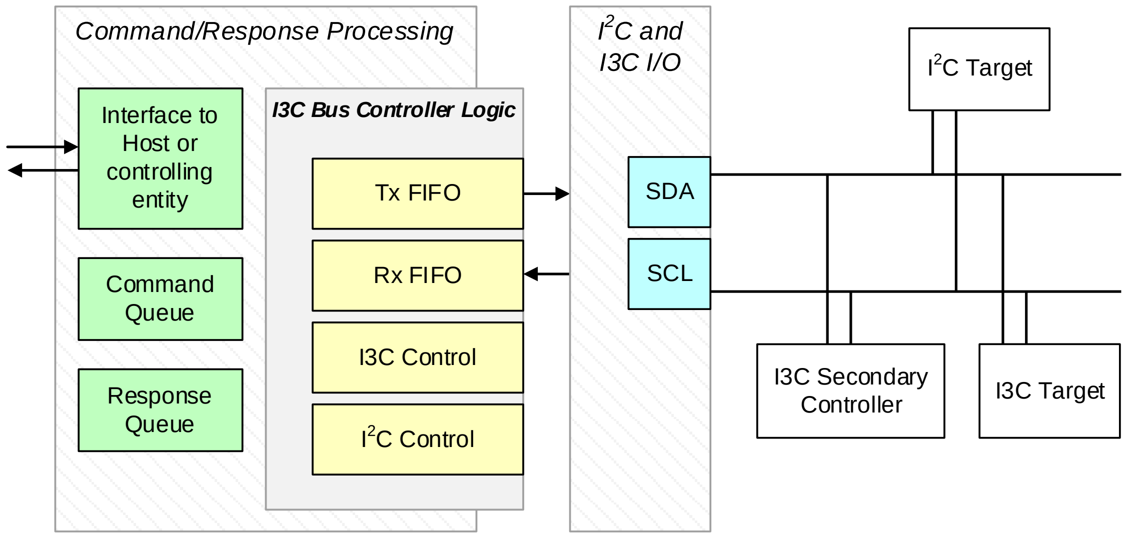

Figure 3 Example TCRI Integration with I3C Bus Controller Logic

In implementations like the one shown in ***Figure 3***, the I3C Bus Controller Logic box generally contains blocks for Tx FIFO, Rx FIFO, I3C Control, and I^{2}C Control. The I3C Bus Controller Logic is responsible for managing the I/O block, driving the enqueued transactions (as described by Command Descriptors), managing the transfer of data to/from the FIFOs, and returning status via Response Descriptors. The Application then uses these blocks in the appropriate manner, to enqueue I3C Transfer Commands with optional data, and to dequeue I3C Transfer Responses with optional data. The Application also uses specific methods to manage the I3C Bus Controller Logic and receive notifications of In-Band Interrupts as well as other I3C Bus events and conditions.

5.2 General Information

The Transfer Command Interface supports Active Controller mode, which enables a Command/Response interface that the Host uses to enqueue transfers and read response status, while the I3C Bus Controller Logic acts as the Active Controller of the I3C Bus.

Note:

> *An I3C Controller implementation may also support Standby Controller mode, if it can hold the role of I3C Secondary Controller on the I3C Bus. This I3C TCRI Specification does not define any requirements relating to I3C Secondary Controller capabilities or Standby Controller mode, but implementers may choose to add such capabilities, depending on the use case.*

The I3C Controller can support any number of I3C Devices and/or Legacy I^{2}C Devices on its single I3C Bus instance, as long as all such Devices have unique Dynamic Addresses (for I3C Devices) and non-conflicting Static Addresses (for Legacy I^{2}C Devices, and for I3C Devices when appropriate). The actual number of Devices that can be used with an implementation might depend on several other factors, including whether the supported Transfer Command format requires a DAT index, and whether the I3C Controller and its Application have sufficient internal memory to support simultaneous transfers to/from multiple Devices.

5.3 Target Device Support Model

5.3.1 I3C Devices

An implementation that supports this Transfer Command/Response Interface as well as the data structure definitions in this Specification is required to support all I3C compliant Devices, however such an implementation is not required to support all I3C Bus speeds, nor all optional I3C Modes.

5.3.2 I^{2}C Devices

Per the I3C Specification ***[MIPI02]***, many types of I^{2}C Target Devices can coexist on an I3C Bus, but I^{2}C Bus Controllers cannot coexist on an I3C Bus.

Interrupts: Legacy I^{2}C Devices use out-of-band mechanisms for interrupts. If the Application wishes to receive such interrupts, then it must implement a method for receiving these (i.e., out-of-band signal inputs or GPIO pins).

Read/Write: Read/writes for I^{2}C Devices are performed in a similar manner as for I3C Devices. However the array of options for scheduling traffic to an I^{2}C Device is necessarily more limited, in terms of speed and Mode.

6 Theory of Operation

The I3C Transfer Command/Response operational flow is defined as a contract and set of expectations for the Application to follow, as well as a set of behavioral requirements for a compliant I3C Bus Controller.

While some of these flows can be executed in parallel, they are treated separately here for purposes of describing I3C Controller operation. Any necessary synchronization between the flows is called out explicitly.

Note:

This section assumes that the I3C Bus Controller is operating as the Active Controller of its I3C Bus.

6.1 Device Management and I3C Addressing

While this Specification does not cover the specific management of I3C Targets or Legacy I^{2}C Targets on the I3C Bus, key requirements must be observed, as detailed in subsequent sections.

6.1.1 Device Attach, Enumeration, and Initialization

After initialization, the I3C Bus Controller shall enumerate all its I3C Devices, and assign unique Dynamic Addresses to each of them. If Legacy I^{2}C Target Devices are also supported, then the Static Addresses of such Legacy I^{2}C Target Devices must be provided by the Application. The method of I3C Address Assignment or configuration is not covered in this Specification.

6.1.2 Device Addressing

I3C Transfer Commands that are addressed to a specific Target Device must either include:

- A unique index for an entry in a special Device Address Table (i.e., **DAT entry**) that contains per-Target configuration, including the Target Address
  - If this is for an I3C Target, then this is typically the Target's assigned Dynamic Address
  - If this is for a legacy I^{2}C Target, then this is the Target's Static Address
- The Target's I3C Address (i.e., provided directly within the Command Descriptor structure)

The specific method used depends on the selected Command Descriptor structure format, and the specific Application might support one or both methods.

Note:

This I3C TCRI Specification does not define the structure of the I3C Controller's DAT entry, nor does it place any requirements on how many DAT entries are required to be implemented. However, if this Transfer Command/Response Interface is used within another overall specification for a particular Application, then that other specification could provide definitions for the structure of DAT entries, and could also place requirements on the number of DAT entries.

I3C Transfer Commands that are addressed to a valid Group Address can also be used in a Command Descriptor structure. In most cases this will be structured as a Write-type command that addresses the Group Address instead of a Target Address.

- Since a Group Address may be assigned to multiple I3C Targets, and since all such I3C Targets will participate in the ACK/NACK based on matching the Group Address in the Address Header, the I3C Bus Controller Logic will regard a Transfer Command as successfully executed if all such I3C Targets respond with ACK, or if at least one such I3C Target responds with an ACK.
- However, the I3C Bus Controller Logic has no way to determine how many of these I3C Targets responded with ACK (i.e., by the Write-type command that is addressed to the Group Address) since multiple ACKs are indistinguishable.
- If no I3C Targets provide ACK based on matching the Group Address in the Address Header, the I3C Bus Controller Logic shall regard the Transfer Command as a failure, and shall return an error code 0x5 (**NACK**) in field **ERR_STATUS** (see ***Section 6.4.1.5***).

I3C Transfer Commands that are addressed to the entire I3C Bus (e.g., Broadcast CCCs) shall indicate this, in a manner specific to the Command Descriptor structure and how the Application uses it. For example:

- If the Application uses DAT indexes in the Command Descriptor, then the Application might define that Transfer Commands for Broadcast CCCs should only include the Command Code, and the DAT entry index field's value is not needed (i.e., a value of 0x0 is acceptable).
- If the Application uses direct addressing, then the Application might define that Transfer Commands for Broadcast CCCs should be addressed to either 7'h00 or 7'h7E (i.e., the I3C Broadcast Address) in the Target Address field.

6.2 Transfer Command Handling

An I3C Controller allows its Application to enqueue transfers via its Command Queue. While acting as the Active Controller of the I3C Bus, these transfers are driven to the I3C Bus by the I3C Bus Controller Logic, and the response status for each transfer shall be read by the Application from the Response Queue, to determine whether the transfer succeeded or failed. This mode of operation is called Active Controller mode.

For each enqueued transfer, the I3C Bus Controller Logic uses the transfer parameters (such as any Command Code and optional Defining Byte for a CCC) encoded in the Command Descriptor structure. These parameters inform the I3C Bus Controller Logic which I3C/I^{2}C Target Device is the recipient of the transfer, or (for Broadcast CCC commands) inform the I3C Bus Controller Logic that a transfer is directed to all I3C Target Devices on the I3C Bus.

Transfers may be either Read-Type or Write-Type:

- **Read-Type:** A Read-Type Transfer Command may address any I3C Target Device that has been assigned a valid Dynamic Address.
  - This might take the form of a Private Read (in SDR Mode), or an HDR Generic Read (in any supported HDR Mode).
  - This might take the form of a Direct Read or Direct GET CCC (in SDR Mode).
  - Note that a Read-Type Transfer Command may not be sent to a Group Address.
- **Write-Type:** A Write-Type Transfer Command may address any I3C Target Device that has been assigned a valid Dynamic Address, or a valid Group Address (if Grouped Addressing is supported).
  - This might take the form of a Private Write (in SDR Mode), or an HDR Generic Write (in any supported HDR Mode).
  - This might take the form of a Direct Write or Direct SET CCC (in SDR Mode).
  - This might take the form of a Broadcast CCC, which addresses all I3C Devices on the Bus.
  - Certain Write-Type commands might be used to address I3C Target Device(s) that have not yet been assigned a Dynamic Address:
    - For example, a Write-Type Transfer Command taking the form of a Broadcast CCC (i.e., an Immediate Data Transfer Command) may be used to send the **SETAASA** CCC, which shall assign Dynamic Addresses to all I3C Target Devices that support this CCC.
    - The Application might also define special Address Assignment Command formats, but such formats are not defined in this I3C TCRI Specification.

Note:

The Transfer Command Interface shall block certain CCCs from being sent using Transfer Commands from the Application, as those CCCs would automatically be sent by the I3C Bus Controller Logic due to particular Transfer Command parameters.

*For example, the **ENTHDR0–ENTHDR7** CCCs are not directly accessible to the Application through a Transfer Command. These CCCs are sent automatically by the I3C Bus Controller Logic, for Transfer Commands that utilize an HDR Mode and also require the I3C Controller to enter that HDR Mode with the appropriate CCC, to start the framing and utilize the HDR coding (per **Section 6.2.5**).*

*Additionally, the **GETACCCR** CCC is not directly accessible to the Application through a Transfer Command, since this Specification does not define an interface that uses Standby Controller mode and does not provide a method for passing Controller-role to a Controller-capable Device (i.e., a Secondary Controller). However, this Specification can be used by Secondary Controller Devices that subsequently acquire the Controller Role and become the Active Controller.*

*Additionally, the **DISEC** CCC might also be sent automatically by the I3C Bus Controller Logic, in response to certain Interrupt Request types that are not allowed per the Application's configuration, or Application-specific per-Target configuration. However, it is allowed for this CCC to be sent via Transfer Commands from the Application.*

6.2.1 Target Device Requirements

In general, the Application shall not initiate a transfer request to an I3C Target Device until and unless the Target has been assigned a Dynamic Address. While the method of assigning a Dynamic Address is not defined by this Specification, the Application that uses this I3C Transfer Command/Response Interface might provide a side-band mechanism for this purpose, or it might rely on other Transfer Commands to perform this process (i.e., to enqueue CCCs).

Once a Target Device has received its Dynamic Address, the Application may initiate transfer requests for any support transfer types, including CCCs (per ***Section 6.3***) in SDR Mode or other optional HDR Modes, using a Command Descriptor and associated data. Each Command Descriptor shall indicate the Target Device to which the Transfer Command is directed.

Note:

*Per **Section 6.1**, if the I3C Controller supports a Command Descriptor format that requires indexed DAT entries, then the Application must first populate a DAT entry containing the Dynamic Address for a Device (or an assigned Group Address, if supported) before sending any transfer requests using the index for that DAT entry.*

6.2.2 Response Descriptor Generation

As the I3C Controller processes each Command Descriptor, it shall conditionally generate and provide Response Descriptors appropriately, as indicated in field **WROC** for each Command Descriptor, and each Response Descriptor shall provide status information for the related transfer. In general, there is a 1:1 mapping between Command Descriptors that the Application enqueues via the Command Queue, and corresponding Response Descriptors that the I3C Controller generates and provides via the Response Queue.

Note:

*The Application may also add special restrictions or change the behavior of field **WROC** for specific operating modes, such that field **WROC** could have a special meaning or could also be disregarded, depending on the Application requirements or the specific operating mode.*

The Application should use different (i.e., changing) values in field **TID** for Command Descriptors, to uniquely identify different Response Descriptors that are received and correlate them to the enqueued Command Descriptors.

6.2.3 Automatic Entry and Exit for HDR Modes

The Speed and Mode of the transfer is set via the **MODE** field in the Command Descriptor structure for a Transfer Command type (see ***Section 7***). This allows the Application to choose the I3C Mode for each transfer. The Command Descriptor may also define additional fields for I3C Modes that are specific to a particular Command Descriptor format.

For all transfers in HDR Modes, the I3C Controller shall automatically enter the chosen HDR Mode, using the **ENTHDR0–ENTHDR7** CCCs per version 1.1.1 of the I3C Specification ***[MIPI02]*** (see ***Section 5.1.9.3.9***) upon receiving the first transfer in such an HDR Mode. The I3C Controller shall automatically exit the HDR Mode at the appropriate time, using the HDR Exit Pattern. The I3C Controller shall determine when to exit the HDR Mode, if <u>any</u> of the following conditions are true:

- The value in the **MODE** field of the next enqueued Command Descriptor to be processed has a different value than the **MODE** field in the previous Command Descriptor; or
- The last Command Descriptor in the Command Queue has been processed, and either no more Command Descriptors remain to be processed; or
- The Application has instructed the I3C Controller to halt or abort further processing; or
- The I3C Controller has halted operation due to an unrecoverable error.

The I3C Controller shall either (a) automatically exit the HDR Mode after processing a Command Descriptor with a value of 1 in the **TOC** field, or (b) as an alternative, choose to remain in the HDR Mode for as long as possible, if subsequent Command Descriptors have been enqueued that will use the same HDR Mode.

The I3C Controller automatically drives the HDR Restart Pattern between transactions (i.e., separate Command Descriptors) in the same HDR Mode, as indicated by the **TOC** field containing the value 0.

Each transfer described by a single Transfer Command is generally a single data message between the framing elements for that HDR Mode. The I3C Bus Controller Logic shall:

1. Start the transfer by automatically entering into that HDR Mode (i.e., after using the appropriate **ENTHDR0–ENTHDR7** CCC, as mentioned above), or continuing after the HDR Restart Pattern (i.e., after a previous transfer in that same HDR Mode which continued the framing, using field **TOC**=0).
2. Send the appropriate structured protocol element (e.g., a Command Word or Header Block) to drive the Address, i.e., the Target Address, Group Address (if supported), or Broadcast Address; and other fields specific to the particular HDR Mode, for the Command Code, Command Bytes or other transfer parameters.
3. Send or receive the data bytes:
   A. For a Write Transfer, the I3C Bus Controller Logic shall send the data bytes using the appropriate structured protocol elements (e.g., one or more Data Words or Data Blocks), using Immediate Data Bytes if indicated by the Transfer Command, or an appropriate data buffer or queue with the length indicated by the Transfer Command.
   B. For a Read Transfer, the I3C Bus Controller Logic shall receive the data bytes from the indicated Target Device, and store them in the data buffer or queue.
   C. If the Command Descriptor format supports Combo transfers, and if the Transfer Command indicates that this is a Combo transfer with both phases in an HDR Mode, then the I3C Bus Controller Logic shall drive an HDR Restart Pattern between the first phase and second phase. The I3C Bus Controller Logic shall also re-send the appropriate structured protocol element (e.g., a Command Word or Header Block) to start the second phase of the transfer, after the HDR Restart Pattern, using the same Address as the first phase.
4. End the transfer, by either driving the HDR Restart Pattern to prepare for the next transfer in that same HDR Mode (i.e., when field **TOC**=0); or automatically exiting that HDR Mode and driving the HDR Exit Pattern (i.e., when field **TOC**=1, or when conditions require exiting that HDR Mode).

If the Command Descriptor format supports Combo transfers, and if the Transfer Command indicates that this is a Combo transfer with **only** the second phase in an HDR Mode, then the I3C Bus Controller Logic shall generate a Repeated START after the end of the first phase (i.e., in SDR Mode) and send the appropriate **ENTHDR0–ENTHDR7** CCC to enter the HDR Mode for the second phase, as defined above.

6.2.4 Sequences of Transfer Commands

The Application may enqueue individual Transfer Commands, where each Command Descriptor's field **TOC** has a value of 1'b1; or a consecutive sequence of Transfer Commands in a given order, comprising a transaction sequence:

- The first and subsequent Command Descriptors (except for last in the sequence) indicate the start and continuation of the sequence, with field **TOC** having a value of 1'b0; and
- The last Command Descriptor indicates the end of the sequence, with field **TOC** having a value of 1'b1.

Within such a sequence, field **TOC** roughly corresponds with the framing elements in the I3C Modes, as defined in ***Section 6.2.5***:

- **In SDR Mode:**
  - If field **TOC** has a value of 1'b1, then the Bus transaction for that Command Descriptor shall end with a STOP condition, in order to allow subsequent reception of In-Band Interrupts (IBIs).
  - If field **TOC** has a value of 1'b0, then the Bus transaction shall end with a Repeated START condition, and the next Command Descriptor shall be processed.
- **In HDR Modes:**
  - If field **TOC** has a value of 1'b1, then the Bus transaction for that Command Descriptor shall conditionally end with an HDR Exit Pattern (i.e., exiting the HDR Mode) per ***Section 6.2.5***.

- If field **TOC** has a value of 1’b0, then the Bus transaction shall end with an HDR Restart Pattern, and the next Command Descriptor shall be processed with continuous framing, per ***Section 6.2.5***.

Note that special additional conditions also apply for Command Descriptors that indicate CCCs, per ***Section 6.3***.

Whenever possible, the Application should attempt to enqueue all Command Descriptors comprising a sequence of Transfer Commands as a single continuous operation or uninterrupted stream of actions, in order to prevent a command sequence stall condition or timeout (as defined in ***Section 6.4.2***). If the Command Queue is unable to contain or accept all such Command Descriptors for this sequence, or if the sequence is longer than the maximum size of the Command Queue, then the Application must first enqueue some Command Descriptors to start processing the sequence, and then actively monitor the status of the Command Queue to ensure that it is able to enqueue additional Command Descriptors when possible, while preventing a command sequence stall condition or timeout.

The I3C Controller shall monitor the Command Queue and attempt to detect situations of possible command sequence stall conditions or timeouts, reporting them as warnings or errors per ***Section 6.13.2***. The I3C Controller shall also monitor the specific execution status of the operating mode, for other mode-specific conditions that might cause an interruption of transfer processing.

Whenever possible, the I3C Controller and its I3C Bus Controller Logic should attempt to mitigate or prevent such underflow situations that might become imminent as the Command Queue drains to approach an empty state, using the following methods as examples:

- Stalling the I3C Bus, using any methods that might be defined and enabled for the I3C Mode;
- Using any available I3C-Mode-specific methods to defer a transaction, or any upcoming actions that might be expected in the near future; or
- Reducing the clock speed at which transfers are driven on the Bus.

In other cases, the I3C Controller and its I3C Bus Controller Logic would not be able to prevent an underflow situation or other interruption of transfer processing, and would be required to cancel the sequence. In this case, the I3C Bus Controller Logic would be required to end the framing using either a STOP condition or HDR Exit Pattern, as appropriate for the I3C Mode. As a result, the I3C Bus Controller Logic would act as though the last executed Command Descriptor had been provided the value 1’b1 in field **TOC** (instead of its actual value of 1’b0).

If the I3C Controller is instructed or forced to cancel a transaction sequence for any reason (including a command sequence timeout) then it shall assert an error to the Application to provide notice that the sequence was terminated before receiving a Command Descriptor having field **TOC**=1. The specific warning and error conditions relating to a command sequence stall condition or timeout shall also be reported as errors to the Application.

Note:

*Since some I3C content protocols might not tolerate a forced STOP condition between two transfers that would otherwise need to be executed in a continuous sequence (i.e., with a Repeated START or HDR Restart Pattern), the Application shall determine whether enqueueing the next Command Descriptor after a cancelled sequence due to a timeout is a correct course of action, or whether the command sequence must be restarted from an earlier transfer. The Application shall determine whether to resume transfers after the STOP condition, or to restart from a prior Transfer Command, by enqueueing new Transfer Commands appropriately for the I3C content protocol after detecting the forced STOP condition. Alternatively, the Application may configure automatic halting if a command sequence timeout occurs (see **Section 6.4.2**).*

Refer to ***Section 6.2***, ***Section 7.1.2***, and ***Section 7.2.2*** for usage of Transfer Commands.

6.2.5 Speed and Mode Changes Within A Sequence

The Speed and Mode of the transfer is set via the **MODE** field in the Command Descriptor structure (see ***Section 7.1.2*** and ***Section 7.2.2***). The I3C Controller shall interpret the value in the **MODE** field in combination with the Transfer Command type.

If multiple Command Descriptors have been enqueued for processing, then the I3C Controller shall compare the values of the **MODE** field between two Command Descriptors, to see whether the transfer Mode and Speed change. If so, then the I3C Controller shall appropriately end the framing in that transaction's mode, which might end with a STOP condition or the HDR Exit Pattern, before starting the next Bus transaction according to the **MODE** field of the next Command Descriptor.

The following requirements shall be observed:

- For a transition from SDR Mode to any HDR Mode, the I3C Controller shall check the **TOC** field in the previous Command Descriptor, and use the appropriate method to end the framing:
  - If field **TOC** was set to 0 in the previous Command Descriptor, then the I3C Controller shall drive a Repeated START condition, before entering the indicated HDR Mode for the next Command Descriptor.
  - If field **TOC** was set to 1 in the previous Command Descriptor, then the I3C Controller shall end SDR Mode framing by driving a STOP condition, and then wait for a Bus Free Condition before driving the new transaction in the indicated HDR Mode for the next Command Descriptor.
- For a transition from any HDR Mode to any other I3C Mode (including another HDR Mode), the I3C Controller shall end the HDR Mode framing using the HDR Exit Pattern, which includes the STOP condition.
  - This applies even if the previous Command Descriptor's **TOC** field was set to 0. For such cases, the I3C Controller must necessarily exit the current HDR Mode by driving the HDR Exit Pattern, before starting the new transaction.
  - This applies for transitions from any HDR Mode to any other HDR Mode; and for transitions from any HDR Mode to SDR Mode (at any speed).
- For transitions between different values indicating changing data rates within SDR Mode, where the previous Command Descriptor's **TOC** field was set to 0:
  - The I3C Controller's standard behavior is to switch to the new data rate after driving the Repeated START condition that separates the two transfers.
  - A I3C Controller might also support optional alternate methods for handling such a transition. If such optional alternate methods are supported, then these must be disabled by default, and explicitly enabled by the Application, which will have a default configuration setting to use the standard behavior.
  - Alternate methods might include:
    - Terminate the transaction with an error, if the I3C Controller detects a different value of the **MODE** field indicating a different data rate within SDR Mode, when field **TOC** was set to 0 in the previous Command Descriptor. The error would be sent to the Host as an interrupt.
    - End the current transaction, drive a STOP condition, and then restart SDR framing with a new transaction in SDR Mode using the new data rate indicated in the next Command Descriptor. If the next Command Descriptor is a CCC, then the I3C Bus Controller Logic must start the CCC framing appropriately (per ***Section 6.3***). In effect, a change in the **MODE** field between two different SDR data rates would act as though the previous Command Descriptor's **TOC** field had been set to 1.

If field **TOC** is set to 1 for a Command Descriptor, then the Bus transaction shall end the framing in that I3C Mode.

- If such a Command Descriptor indicates SDR Mode at any speed, then the I3C Controller shall end with a STOP condition, in order to end the framing and also allow subsequent IBI reception.
- If such a Command Descriptor indicates any HDR Mode, then the I3C Controller shall generally exit the current HDR Mode by driving the HDR Exit Pattern, unless an implementer supports deferred or delayed exit as an optimization.

  Such an optimization must be available to the Application, that it might be explicitly enabled or disabled, where the Application will have a default configuration setting to use the standard behavior (i.e., the HDR Exit Pattern is driven if field **TOC** is set to 1).

If field **TOC** is set to 0, and if the values of the **MODE** field do not change between this Command Descriptor and the next Command Descriptor, then the Bus transaction shall end with either a Repeated START condition (for SDR Mode) or the HDR Restart Pattern (for HDR Modes), and then the next Command Descriptor shall be processed while remaining in the same speed and Mode.

Note:

> *For some Command Descriptors that indicate CCC transfers in SDR Mode, field **TOC** with a value of 0 might also end the CCC framing with the End of CCC Procedure for SDR Mode, depending on the Command Descriptor format and the transaction type of the next Command Descriptor (see **Section 6.3**).*

In some special cases, the I3C Controller might be forced to end a transaction with a STOP condition (for SDR Mode) or HDR Exit Pattern (for an HDR Mode), even though a Command Descriptor might have field **TOC** set to 0. Such situations should include the case where this Command Descriptor is the last Command Descriptor in the Command Queue, and where stalling the I3C Bus clock might not be permitted. For such cases, in order to abide by the I3C Bus protocol, the I3C Controller would subsequently be required to drive a START condition once it began the next Transfer Command (e.g., if a subsequent Command Descriptor had been enqueued by the Application), if it had previously been forced to drive a STOP condition (or HDR Exit Pattern) after running out of Command Descriptors to process. Depending on the I3C Mode and the type of transfer indicated by the subsequent Command Descriptor, the I3C Controller might also need to automatically re-enter the HDR Mode (if applicable) or restart the CCC framing in SDR Mode (if applicable), in order to restore the I3C Bus to its prior state before the forced end of the prior activity.

Note:

> *Since an I3C content protocol might not tolerate a forced STOP condition between two transfers that would otherwise need to be executed in a continuous sequence (i.e., with a Repeated START or HDR Restart Pattern), the Application shall determine whether this is a correct course of action, or whether the command sequence must be restarted from an earlier transfer. The Application shall determine whether to resume after the STOP condition, or to restart from a prior Transfer Command, by enqueueing new Transfer Commands appropriately for the I3C content protocol after detecting the forced end of the prior activity.*

If the Application signals the I3C Controller to pause or abort operations, then the I3C Controller shall attempt to either complete or abort any in-progress transfers that the I3C Bus Controller Logic is currently driving. When necessary, the I3C Controller shall abort such transfers at the earliest opportunity, even if the associated Command Descriptor(s) have field **TOC** set to 0 (i.e., not the intended end of a sequence).

6.2.6 Support for I3C In-Band Interrupts

If I3C In-Band Interrupts (IBIs) are enabled on the I3C Bus, then the I3C Controller must also provide a mechanism for handling IBI Requests, Hot-Join Requests or Controller-Role Requests that might be raised by I3C Target Devices on the I3C Bus. These might occur with the first Transfer Command in a sequence (i.e., when the I3C Controller initiates a START condition to begin processing a Command Descriptor). If this is a sequence that is enqueued after a previous sequence ends, then the last Transfer Command in the previous sequence typically has field **TOC** set to 1, to drive the STOP condition (per ***Section 6.2.4***).

The Application should configure the I3C Controller to begin such sequences (or single transfers) with START followed by the I3C Broadcast Address (7’h7E) in order to provide opportunities for I3C Target Devices to raise such requests (i.e., if the Bus Available Condition is not seen).

Note:

*Starting I3C transfers with START followed by the I3C Broadcast Address (7’h7E) is recommended by the I3C Specification (see **[MIPI02] Annex A, Section A.2**). If the I3C Controller does not start Private Write or Private Read transfers with START followed by the I3C Broadcast Address, then any I3C Targets wishing to raise an In-Band Interrupt Request would need to either: (A) wait for a START, then arbitrate their Dynamic Address into the Address Header; or (B) wait for the appropriate Bus Available Condition to pull SDA Low to request a START, which requires the Application to temporarily pause sending Transfer Commands to create this opportunity.*

If the Application allows for this, then the I3C Controller should ACK the IBI Request, allow the I3C Target to send its IBI data payload, handle it appropriately (which might include initiating a subsequent Pending Read operation to fetch more data from the I3C Target, if appropriate for the Mandatory Data Byte), and then drive a Repeated START followed by the intended Transfer Command. Such IBI handling should be largely transparent from the I3C Transfer Command/Response Interface, and the I3C Controller should also notify the Application that the IBI Request was received and processed accordingly (i.e., via an Application-specific interface).

Note:

The interface for reporting In-Band Interrupt notifications to the Application is not defined in this Specification.

*When using START followed by the Broadcast Address, if no I3C Targets take this opportunity to raise an In-Band Interrupt Request, and if the Broadcast Address wins the Address Arbitration in the Address Header, then the I3C Controller shall wait for the ACK of the Broadcast Address, and then drive the intended Private Write or Private Read transfer after a Repeated START. However, if this transfer is a CCC, then the START / 7’h7E is part of the CCC framing (i.e., entering the modality, per **Section 6.3**).*

6.2.7 Support for I3C 'Short' Read-Type Transfers

For I3C Read-type transfers, a successful Response Descriptor (i.e., field **ERR_STATUS** having a value of 0x0) might indicate a 'short' Read-Type transfer, where the I3C Target Device returns fewer bytes than expected, compared to the value of field **DATA_LENGTH** in the corresponding Transfer Command. This might not necessarily be an error, as some I3C content protocols might depend on variable-length Private Read transfers.

A Transfer Command for an I3C Read-type might indicate that a 'short' Read-Type transfer is permitted; or it might indicate that the I3C Read-Type transfer will only be successful if the full number of requested bytes is received from the I3C Target Device, and that processing should be halted if the I3C Target Device does not return the requested number of bytes in the Transfer Command.

- **If the Transfer Command indicates that a 'short' Read-Type transfer is allowed, and if a Read-Type transfer is determined to be 'short':**

  Then the I3C Controller shall generate a Response Descriptor, as usual:

  - In this case, the 'short' Read-Type transfer shall be handled as a Read-Type transfer of the full expected length, where an I3C Target returned all requested data bytes; and shall be regarded as successful unless other errors had occurred.
  - In other words, if a 'short' Read-Type transfer generated no other errors, then the I3C Controller shall generate a Response Descriptor with error code 0x0 (**SUCCESS**), and shall continue processing any subsequent Transfer Commands that might be enqueued. It is the responsibility of the Application to ensure that the length of the returned data is appropriate for the I3C content protocol, in the context of any preceding activity (e.g., any Write-Type transfers or CCCs sent before such a Read-Type transfer) before determining whether a 'short' Read-Type transfer is actually a successful result.

- **If the Transfer Command indicates that a 'short' Read-Type transfer is not permitted, and if a Read-Type transfer is determined to be 'short':**

  Then the I3C Controller shall treat this as an error:

  - In this case, the I3C Controller shall end the framing after the Read-Type transfer (i.e., regardless of the value of field **TOC** in the Transfer Command), and shall generate a Response Descriptor with error code 0x7 (**I3C_SHORT_READ_ERR**) in field **ERR_STATUS** (per ***Section 6.4.1.7***). The I3C Controller shall then halt operations, and shall assert an error to the Application.

  Specific Transfer Command types that allow a 'short' Read-Type transfer to be treated as an error include the following:

  - For Format 1 (Legacy Format, Indexed with DAT): Regular Data Transfer Command (see ***Section 7.1.2.2***)
  - For Format 2 (Legacy Format, Direct Addressed): Regular Data Transfer Command (see ***Section 7.2.2.2***)

Note:

If the Transfer Command type does not specify otherwise, or if a Transfer Command type does not define a field that provides control over whether a 'short' Read-Type transfer might be permitted or not permitted, then the I3C Controller shall always treat a 'short' Read-Type transfer as though it were permitted (i.e., successful unless other errors occur).

Since the I3C Controller always generates Response Descriptors for Read-Type transfers that are processed, field **DATA_LENGTH** for the Response Descriptor of a 'short' Read-Type transfer shall contain the actual number of data bytes returned from the I3C Target Device.

6.3 Managed CCC Transfer Framing Model

Per ***Figure 4*** (reproducing ***Figure 41*** from the I3C v1.1.1 Specification ***[MIPI02])***, the framing model for Direct CCC commands makes it possible to address more than one Target with a sequence of commands, as described in the I3C Specification at ***Section 5.1.9.2.2 [MIPI02]***.

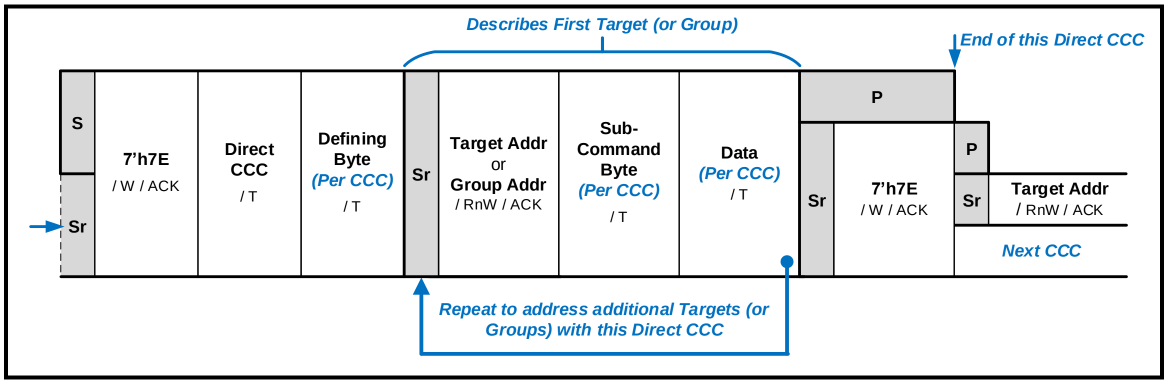
Describes First Target (or Group)
End of this Direct CCC
S
Sr
7'h7E / W / ACK
Direct CCC / T
Defining Byte (Per CCC) / T
Sr
Target Addr or Group Addr / RnW / ACK
Sub-Command Byte (Per CCC) / T
Data (Per CCC) / T
P
Sr
7'h7E / W / ACK
P
Sr
Target Addr / RnW / ACK
Next CCC
Repeat to address additional Targets (or Groups) with this Direct CCC

Figure 4 Direct CCC Framing Model

Per ***Figure 5*** (reproducing ***Figure 40*** from the I3C v1.1.1 Specification ***[MIPI02]***), Broadcast CCCs also have a similar framing model.

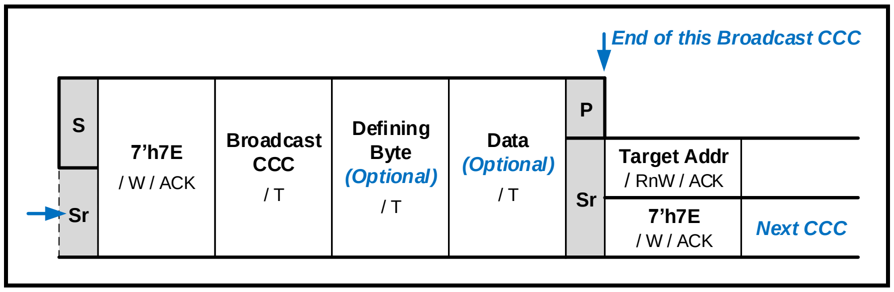

Figure 5 Broadcast CCC Framing Model

Both of these models are allowed within SDR Mode framing. As a result, a single large transfer may contain any combination of Direct CCC segments, Broadcast CCC messages, and Private Read/Write transfers.

The I3C Controller shall support a managed CCC transfer framing model, using Command Descriptors to send one or more CCCs using standard CCC framing in SDR Mode, and optionally in supported HDR Modes. Each Command Descriptor that indicates a CCC transfer (i.e., a Transfer Command that is either a Direct CCC segment or a Broadcast CCC message) shall be sent according to the defined CCC framing in the indicated I3C Mode.

Note:

*Support for Managed CCC framing in optional HDR Modes shall depend on the Command Descriptor format, and the Application may choose to support HDR Modes. See **Section 6.3.5** for more details.*

For example, if the Application enqueues a single Command Descriptor with field **TOC** set to 1'b1, then the I3C Bus Controller Logic shall automatically drive any necessary framing elements that are appropriate for the CCC transfer:

- **For a Direct CCC:** The I3C Bus Controller Logic shall start the framing automatically, by driving a START or Repeated START condition. The I3C Bus Controller Logic shall then send the Broadcast Address (i.e., 7'h7E), the Command Code for the Direct CCC, the optional Defining Byte (if indicated), and a Repeated START, before sending the Dynamic Address of the indicated Device from the Command Descriptor.
  - If the indicated Device ACKs its Dynamic Address along with the RnW bit:
    - **For Direct Write or Direct SET CCCs:** The I3C Bus Controller Logic shall send the data for this Direct CCC segment. The data might be sent as Immediate Data Bytes contained within the Command Descriptor, or it might be sent from the Tx Data Buffer.
    - **For Direct Read or Direct GET CCCs:** The I3C Bus Controller Logic shall allow the indicated Device to return data, which shall be stored into the Rx Data Buffer.
  - If the indicated Device NACKs its Dynamic Address along with the RnW bit:
    - The I3C Controller shall attempt to retry the Direct CCC at least once following the first NACK, per the following conditions:
      - If the indicated Device's DAT entry indicates no retries for NACKed Transfer Commands, then the I3C Controller shall always attempt one retry for a NACKed Direct CCC, per the I3C Specification's mandatory single-retry model (see the I3C Specification at ***Section 5.1.9.2.3 [MIPI02]***).
      - However, if the indicated Device's DAT entry indicates any additional retries, then the I3C Controller shall retry according to that field's setting.
    - If the indicated Device NACKs its Dynamic Address repeatedly, beyond the retry count determined by the above conditions, then this shall be reported as an error per ***Section 6.3.3***.
- **For a Broadcast CCC:** The I3C Bus Controller Logic shall start the framing automatically, by driving a START or Repeated START condition. The I3C Bus Controller Logic shall then send the Broadcast Address, the Command Code for the Broadcast CCC, the optional Defining Byte (if indicated), and the data message indicated by (or contained within) the Command Descriptor.

Since this transfer uses a single Command Descriptor, the I3C Bus Controller Logic shall drive a STOP condition at the end of the transfer.

The Application may enqueue such a Command Descriptor indicating a CCC with any Command Code value, for any CCC (Broadcast or Direct) that is not blocked by the I3C Controller, according to the list of blocked CCCs (per ***Section 6.2***).

Similarly, the Application can control whether a Transfer Command containing a CCC will use a Defining Byte or not. The specific method shall vary, depending on the specific Transfer Command type and the field values in the Transfer Command (see ***Section 7***).

The I3C Controller shall neither expect nor require the Application to enqueue any preceding Command Descriptors that might attempt to manually drive the CCC framing (i.e., by sending the Broadcast Address or other bytes) and do not also contain all necessary values for the Direct CCC segment (i.e., the indicated Device and the data) or the Broadcast CCC message (i.e., the data to send to the I3C Bus). The Application should not attempt to enqueue any Command Descriptors that would attempt to manually drive such CCC framing (or second-guess the I3C Bus Controller Logic by directing transfers to the I3C Broadcast Address for each component of CCC framing).

6.3.1 Direct CCC Framing with Multiple Segments in a Transfer

To perform such a transfer comprising multiple Direct CCC segments using the same CCC and optional Defining Byte, the Application shall enqueue multiple consecutive Command Descriptor entries in the Command Queue. For multiple segments of the same Direct CCC, the Command Descriptors shall have the following properties:

- The Command Attribute (**CMD_ATTR**) field shall be set to an appropriate value for a Transfer Command, specific to the Transfer Command type (see ***Section 7.1.2*** and ***Section 7.2.2***)
- The Terminate on Completion (**TOC**) field shall be set to 1'b0 for all such Command Descriptors, except the last
- The **TOC** field shall be set to 1'b1, only for the last Command Descriptor
- Specific other fields shall indicate a CCC-type transfer (i.e., segment), depending upon the Transfer Command type:
  - For an Immediate Data Transfer Command, see ***Section 7.1.2.1.1*** and ***Section 7.2.2.1.1***
  - For a Regular Data Transfer Command, see ***Section 7.1.2.2.1*** and ***Section 7.2.2.2.1***
  - Note that Direct CCCs are in the range 0x80–0xFF (see the I3C Specification ***[MIPI02]*** at ***Section 5.1.9.3***)
- In addition, other fields shall contain the same Command Code and optional Defining Byte values in all such Command Descriptors, per the Transfer Command type.

CCCs with Defining Bytes may be used in sequences of CCCs as part of Direct CCC command framing. The I3C Controller shall drive CCC framing appropriately, and shall determine when to restart the CCC framing upon detecting any transitions between different Defining Byte values within the same CCC, or transitions between using a Defining Byte or not using a Defining Byte (in either direction).

The first Command Descriptor structure for such a transfer shall include all values needed to send the first Direct CCC segment, i.e., to send the desired Command with optional Defining Byte to the first Target Device. The subsequent Command Descriptor structures within the same transfer shall each include all values needed to send a subsequent Direct CCC segment, and each shall have all field values matching the first Command Descriptor structure, **except** for the following:

- If the Command Descriptor format uses indexes to a DAT, then field **DEV_INDEX** shall provide the index to a valid DAT entry, containing the specific Target's Dynamic Address (or a valid Group Address, if appropriate) that is addressed by a subsequent Direct CCC segment.
- If the Command Descriptor format directly uses the Target addresses, then field **DEV_ADDRESS** shall provide the specific Target's Dynamic Address (or a valid Group Address, if appropriate) that is addressed by a subsequent Direct CCC segment.
- Specific fields shall indicate that the Command Descriptor is a Direct CCC segment, as shown above
- Fields shall indicate the data to read from the addressed Target Address, for a Direct Read or Direct GET CCC segment; or the data to write to the addressed Target Address (or Group Address), for a Direct Write or Direct SET CCC segment:
  - For Direct Write or Direct SET CCC segments, the Command Descriptor might contain Immediate Data Bytes, which might also require a different value in field **CMD_ATTR** or other fields, per the Transfer Command type; the I3C Controller shall otherwise treat such Command Descriptors equivalently (i.e., Immediate versus Regular) in order to verify the intent of writing data and continuing the CCC framing;
- Field **TOC** shall be set to 1'b1 only in the last Command Descriptor; otherwise, field **TOC** shall be set to 1'b0 for other subsequent Command Descriptors (i.e., not the last in the sequence, per ***Section 6.2.4***).

The I3C Controller shall process such a sequence of multiple consecutive Command Descriptor entries having these properties, and shall drive the appropriate actions on the I3C Bus. Since all such Command Descriptors have the same CCC and optional Defining Byte values, the I3C Bus Controller Logic shall only start the framing once, by sending the Broadcast Address (i.e., 7'h7E), the CCC, the optional Defining Byte (if indicated), and a Repeated START, before sending the Dynamic Address of the first indicated Device from the first Command Descriptor. The I3C Bus Controller Logic shall also not be required to re-send these elements (i.e., restarting the CCC framing) between Direct CCC segments, since the CCC and optional Defining Byte are the same for all such Command Descriptors.

However, in situations where the I3C Controller detects that the CCC or optional Defining Byte values do change from one Command Descriptor to the next, the I3C Bus Controller Logic shall restart the CCC framing appropriately, by sending the Broadcast Address, the new CCC, the new Defining Byte and then another Repeated START, per Direct CCC framing in SDR Mode.

Note:

*The managed CCC transfer framing model of one Command Descriptor per Direct CCC segment is structured such that, were each single Command Descriptor in such a sequence to be enqueued separately by the Application (i.e., not consecutively as part of a sequence) with field **TOC** set to 1 for each Command Descriptor, the I3C Bus Controller Logic would still have all values needed to send each Direct CCC segment on its own (i.e., without any other Command Descriptors to indicate any CCC framing elements in SDR Mode) as a separate transaction on the I3C Bus. Note that such a change would mean that the I3C Bus Controller Logic must necessarily start the Direct CCC framing (i.e., by starting the SDR frame and sending the Broadcast Address, CCC and optional Defining Byte) before each Direct CCC segment, and it must also drive a STOP condition at the end of each Direct CCC segment. Determining whether this would be a valid and correct use of such CCCs in a sequence for any particular I3C content protocol is beyond the scope of this I3C TCRI Specification.*

6.3.1.1 Operating Modes, Flow, and Requirements

For Direct Write or Direct SET CCC segments with short payloads: The Application may optionally use Immediate Data Transfer Commands under certain circumstances, according to the Transfer Command definition and the length of the Direct SET CCC payload (e.g., see ***Section 7.1.2.1*** and ***Section 7.2.2.1***).

For Direct Write or Direct SET CCC segments with longer payloads: The Application shall use Regular Data Transfer Commands (e.g., see ***Section 7.1.2.2*** and ***Section 7.2.2.2***).

For all Direct Read or Direct GET CCC segments: The Application shall use Regular Data Transfer Commands (e.g., see ***Section 7.1.2.2*** and ***Section 7.2.2.2***).

All such Command Descriptors should be enqueued to the Command Queue in sequential order. The Application should generally prepare and enqueue these Command Descriptors in one operation (i.e., without allowing the Command Queue to become empty in the middle of processing).

Processing Flow and Requirements

Figure 6 shows a representation of the high-level logical flow that the I3C Controller shall use when processing multiple consecutive Command Descriptors in such a multi-segment Direct CCC transfer, using Direct CCC framing in SDR Mode. This flow does not depend on any specific Transfer Command type (per ***Section 6.2***) and shows how the I3C Bus Controller Logic determines when it must restart the CCC framing. This flow allows the I3C Bus Controller Logic to optimize for efficient I3C Bus transfers by not re-sending CCCs or Defining Bytes unless necessary.

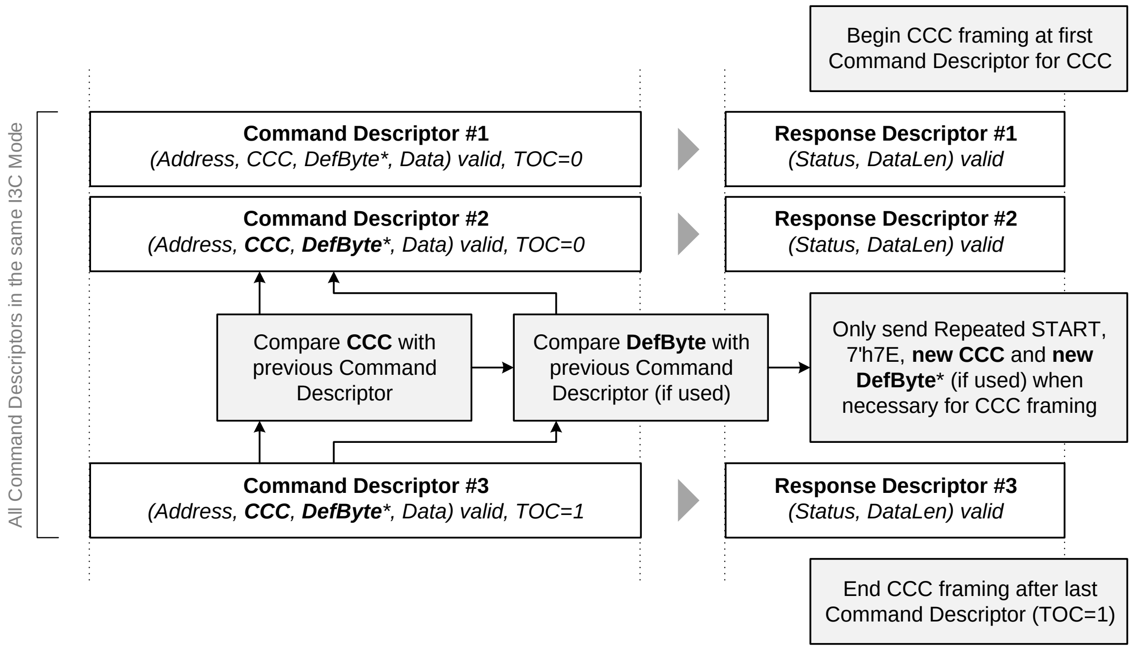
Begin CCC framing at first Command Descriptor for CCC
All Command Descriptors in the same I3C Mode
Command Descriptor #1
(Address, CCC, DefByte*, Data) valid, TOC=0
Response Descriptor #1
(Status, DataLen) valid
Command Descriptor #2
(Address, CCC, DefByte*, Data) valid, TOC=0
Response Descriptor #2
(Status, DataLen) valid
Compare CCC with previous Command Descriptor
Compare DefByte with previous Command Descriptor (if used)
Only send Repeated START, 7'h7E, new CCC and new DefByte* (if used) when necessary for CCC framing
Command Descriptor #3
(Address, CCC, DefByte*, Data) valid, TOC=1
Response Descriptor #3
(Status, DataLen) valid
End CCC framing after last Command Descriptor (TOC=1)

Figure 6 Support for Direct CCC Commands Framing Model

In the high-level logical flow illustrated above, the flow shows how a transfer comprising three consecutive Command Descriptors for Direct CCC GET or SET segments (either with or without Defining Bytes) shall be processed in an optimal manner, as the I3C Controller compares the relevant field values for two adjacent Command Descriptors.

While processing Direct CCC framing, the following conditions and requirements are defined:

- The I3C Controller shall drive a START or Repeated START condition for the first such Command Descriptor, followed by the Broadcast Address (i.e., 7'h7E), then the Command Code, followed by the optional Defining Byte (if indicated in the Command Descriptor).
- If a Command Descriptor's field **TOC** is 0, then the I3C Controller shall attempt to stay in the CCC framing for SDR Mode after that segment. However, if a Command Descriptor's field **TOC** is 1, then the I3C Controller may exit the CCC framing for SDR Mode, by driving a STOP condition after that segment.
- The I3C Controller shall compare the fields containing the CCC and optional Defining Byte values, and shall only restart the CCC framing (i.e., drive a Repeated START, the Broadcast Address, the CCC and optional Defining Byte) if required for the next CCC segment, if any of the following cases are detected:
  - If the previous Command Descriptor indicated no Defining Byte, but the next Command Descriptor did indicate a Defining Byte was present;
  - If the previous Command Descriptor indicated that a Defining Byte was present, but the next Command Descriptor indicated no Defining Byte;
  - If the previous Command Descriptor and next Command Descriptor had the same CCC value, and both did indicate different Defining Byte values; or

- If the previous Command Descriptor and next Command Descriptor had different CCC values.

- If none of the above cases were detected, then the I3C Controller shall not restart the CCC framing, and shall only drive a Repeated START, followed by the Dynamic Address of the Target Address (or Group Address, for Direct SET CCCs that might support such addressing) in order to start the next segment.

These conditions and requirements for restarting the Direct CCC framing based on Command Descriptor fields are aligned with the End of CCC Command method specified in the I3C Specification at ***Section 5.1.9.2.1 [MIPI02]***.

6.3.1.2 Example Flows

Figure 7 shows an example of three consecutive Command Descriptors for Direct CCC segments, each having the same CCC value and different Defining Byte values. In this example, the I3C Controller detects the changing Defining Byte fields in the second and third Command Descriptors, so its I3C Bus Controller Logic must restart the CCC framing in order to send the and new Defining Byte value before each such segment, even though the CCC field does not change. Since restarting the CCC framing means that the new Defining Byte value is written to the I3C Bus (along with the CCC value), it remains in force for the next Direct CCC segment addressing a Target Device.

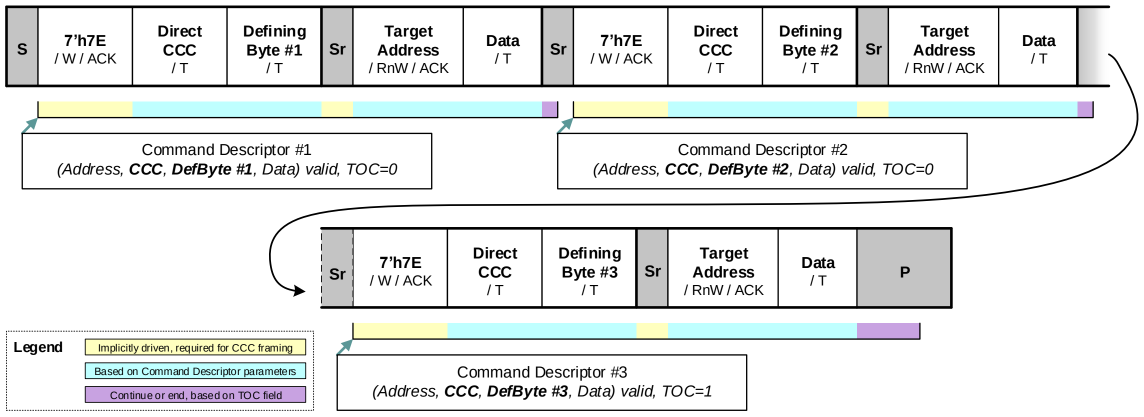
S
7'h7E / W / ACK
Direct CCC / T
Defining Byte #1 / T
Sr
Target Address / RnW / ACK
Data / T
Sr
7'h7E / W / ACK
Direct CCC / T
Defining Byte #2 / T
Sr
Target Address / RnW / ACK
Data / T
Command Descriptor #1 (Address, CCC, DefByte #1, Data) valid, TOC=0
Command Descriptor #2 (Address, CCC, DefByte #2, Data) valid, TOC=0
Sr
7'h7E / W / ACK
Direct CCC / T
Defining Byte #3 / T
Sr
Target Address / RnW / ACK
Data / T
P
Command Descriptor #3 (Address, CCC, DefByte #3, Data) valid, TOC=1
Legend
Implicitly driven, required for CCC framing
Based on Command Descriptor parameters
Continue or end, based on TOC field

Figure 7 Direct CCC Commands Framing Model: Example 1

Figure 8 shows an example of three consecutive Command Descriptors for Direct CCC segments, each having different CCC values. In this example, the I3C Controller detects the changing CCC fields in the second and third Command Descriptors, so its I3C Bus Controller Logic must restart the CCC framing in order to send the CCC value and new Defining Byte value before each such segment. For both such transitions, restarting the CCC framing means that the new CCC value and its Defining Byte value are written to the I3C Bus, such that they remain in force for the next Direct CCC segment addressing a Target Device.

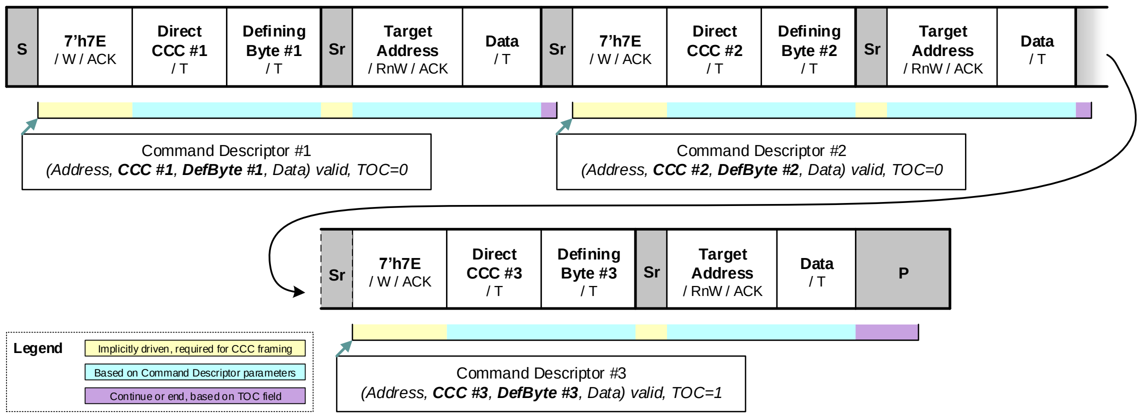
S
7'h7E / W / ACK
Direct CCC #1 / T
Defining Byte #1 / T
Sr
Target Address / RnW / ACK
Data / T
Sr
7'h7E / W / ACK
Direct CCC #2 / T
Defining Byte #2 / T
Sr
Target Address / RnW / ACK
Data / T
Command Descriptor #1 (Address, CCC #1, DefByte #1, Data) valid, TOC=0
Command Descriptor #2 (Address, CCC #2, DefByte #2, Data) valid, TOC=0
Sr
7'h7E / W / ACK
Direct CCC #3 / T
Defining Byte #3 / T
Sr
Target Address / RnW / ACK
Data / T
P
Command Descriptor #3 (Address, CCC #3, DefByte #3, Data) valid, TOC=1
Legend
Implicitly driven, required for CCC framing
Based on Command Descriptor parameters
Continue or end, based on TOC field

Figure 8 Direct CCC Commands Framing Model: Example 2

Figure 9 shows an example of three consecutive Command Descriptors for Direct CCC segments, where the first has no Defining Byte value, while the second and third each have different Defining Byte values for the same CCC. In this example, even though the I3C Controller detects that the CCC fields are unchanged for all three Command Descriptors, the transition from a CCC without a Defining Byte value to a CCC with a particular Defining Byte value (i.e., between the first and second Command Descriptors) requires the I3C Bus Controller Logic to restart the CCC framing, in order to re-send the CCC value with the Defining Byte value before the second Direct CCC segment. Similarly, the transition between different Defining Byte values (i.e., between the second and third Command Descriptors) requires the I3C Bus Controller Logic to restart the CCC framing again, in order to send the CCC value and the new Defining Byte value before the third Direct CCC segment. For both such transitions, restarting the CCC framing means writing the CCC value and the new Defining Byte value to the I3C Bus, such that they remain in force for the next Direct CCC segment addressing a Target Device.

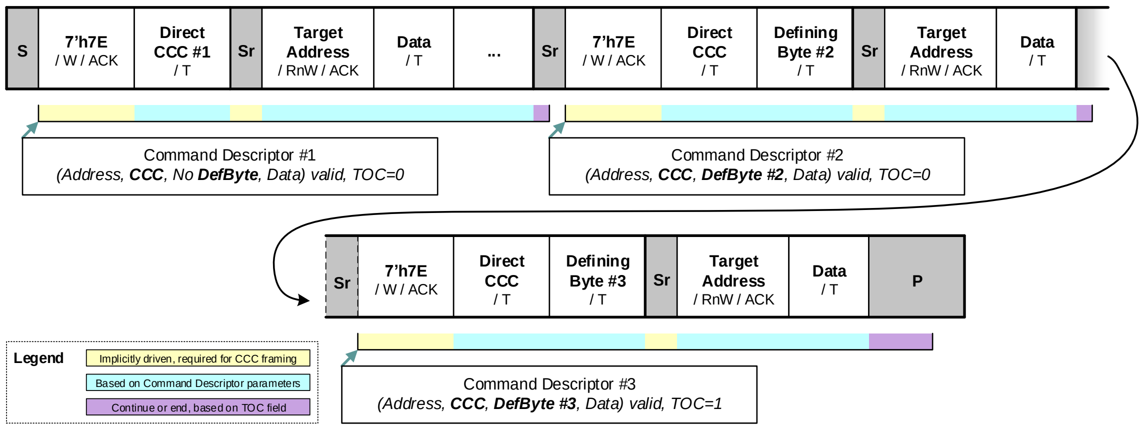
S
7'h7E / W / ACK
Direct CCC #1 / T
Sr
Target Address / RnW / ACK
Data / T
...
Sr
7'h7E / W / ACK
Direct CCC / T
Defining Byte #2 / T
Sr
Target Address / RnW / ACK
Data / T
Command Descriptor #1 (Address, CCC, No DefByte, Data) valid, TOC=0
Command Descriptor #2 (Address, CCC, DefByte #2, Data) valid, TOC=0
Sr
7'h7E / W / ACK
Direct CCC / T
Defining Byte #3 / T
Sr
Target Address / RnW / ACK
Data / T
P
Command Descriptor #3 (Address, CCC, DefByte #3, Data) valid, TOC=1
Legend
Implicitly driven, required for CCC framing
Based on Command Descriptor parameters
Continue or end, based on TOC field

Figure 9 Direct CCC Commands Framing Model: Example 3

Note:

The I3C Controller would also detect the reverse of the situation shown in the previous example, where one Command Descriptor indicated a CCC with a particular Defining Byte value, and the next Command Descriptor indicated the same CCC without a Defining Byte. Such a situation would require the I3C Bus Controller Logic to restart the CCC framing, in order to re-send the CCC value (without a Defining Byte) before sending the Direct CCC segment associated with the next Command Descriptor.

Figure 10 shows an example of three consecutive Command Descriptors for Direct CCC segments, each having the same CCC and Defining Byte values. In this example, the I3C Controller drives the CCC and Defining Byte value, as part of starting the CCC framing on behalf of the first Command Descriptor. It then subsequently detects no changes to the CCC fields or the Defining Byte fields, for either the second or third Command Descriptors. Since fields do not change, the I3C Bus Controller Logic is not required to restart the CCC framing for each Direct CCC segment. The CCC and Defining Byte that were sent at the start of CCC framing (i.e., on behalf of the first Command Descriptor) remain in force, as all such Target Devices that support CCCs shall have been required to track the CCC and Defining Byte value, per CCC framing in SDR Mode.

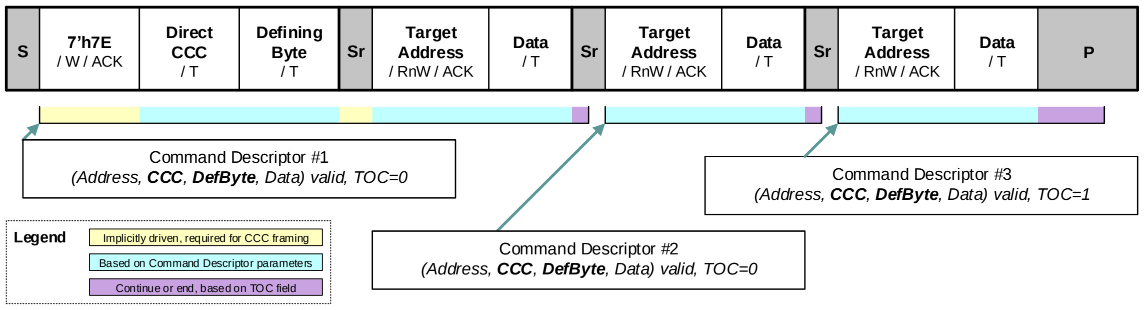
S
7'h7E / W / ACK
Direct CCC / T
Defining Byte / T
Sr
Target Address / RnW / ACK
Data / T
Sr
Target Address / RnW / ACK
Data / T
Sr
Target Address / RnW / ACK
Data / T
P
Command Descriptor #1 (Address, CCC, DefByte, Data) valid, TOC=0
Command Descriptor #2 (Address, CCC, DefByte, Data) valid, TOC=0
Command Descriptor #3 (Address, CCC, DefByte, Data) valid, TOC=1
Legend
Implicitly driven, required for CCC framing
Based on Command Descriptor parameters
Continue or end, based on TOC field

Figure 10 Direct CCC Commands Framing Model: Example 4

6.3.2 Mixing Direct and Broadcast CCCs

The Application may also use the **TOC** field to indicate continuous framing with multiple consecutive Command Descriptors, in order to compose larger transfers in CCC framing which might include Broadcast CCCs in addition to Direct CCCs, in any valid sequence allowed for CCCs in SDR Mode. The I3C Controller shall inspect the CCC related field values for such Command Descriptors, and handle the transitions appropriately (i.e., from a Broadcast CCC message to a Direct CCC segment, and from a Direct CCC segment to a Broadcast CCC message) according to the flows for CCC framing in SDR Mode.

Each Command Descriptor shall either indicate a Target Address (or Group Address, if allowed) for Direct CCCs, or use a Broadcast CCC value with an appropriate field value to address all Devices on the I3C Bus.

To send a Broadcast CCC, the Command Descriptor structure shall have the following properties:

- The Command Attribute (**CMD_ATTR**) field shall be set to an appropriate value for a Transfer Command, specific to the Transfer Command type (see ***Section 7.1.2*** and ***Section 7.2.2***)
- The **DEV_INDEX** field shall be set to zero, per ***Section 6.2***
- Specific other fields shall indicate a CCC-type transfer (i.e., segment), per the Transfer Command type:
  - For an Immediate Data Transfer Command, see ***Section 7.1.2.1.1*** and ***Section 7.2.2.1.1***
  - For a Regular Data Transfer Command, see ***Section 7.1.2.2.1*** and ***Section 7.2.2.2.1***
  - Note that Broadcast CCCs are in the range 0x00–0x7F, per the I3C Specification at ***Section 5.1.9.3 [MIPI02]***.

The I3C Controller shall interpret the fields in each Command Descriptor according to this model, and drive all necessary framing elements, including restarting CCC framing as needed, without any special Command Descriptors or other actions from the Application to handle the lower-level I3C framing elements for CCCs in SDR Mode.

6.3.3 Error Handling for CCC Flows

Each segment or phase of CCC flows (i.e., transfers in a continuous sequence) shall be processed and reported individually, per the setting of field **WROC** in each Command Descriptor that comprises the flow(see ***Section 6.2.2***). However, Applications should generally set field **WROC** to 1'b1 to receive Response Descriptors for each segment or phase.

For Command Descriptors describing Direct CCC segments, a Target might choose to NACK the Direct CCC sent to its Dynamic Address. If this occurs, and if the Direct CCC has been retried according to the NACK retry count, then the Response Descriptor shall indicate this result using a value of 0x5 (**NACK**) in field **ERR_STATUS**. Note that a Target Device would typically NACK its Dynamic Address, for any CCCs (or any CCCs with a particular Defining Byte value) that it does not support.

For Direct CCC segments that address a Group (i.e., a Direct Write or Direct SET CCC), all Targets that are assigned to the Group Address and that support the CCC may choose to respond to the Direct CCC segment and ACK the Group Address; however, if none of these Targets choose to respond, then the I3C Bus Controller Logic shall treat this as a NACK of the Group Address, and shall retry the Direct CCC according to the NACK retry count for the Group (i.e., as with a Direct CCC that is sent to a Dynamic Address).

For Command Descriptors describing Broadcast CCC messages, if no Targets ACK the Broadcast Address, then this is a CCC framing error, and the Response Descriptor shall indicate this result using a value of 0x4 (**ADDR_HEADER**) in field **ERR_STATUS**. This might also occur for Direct CCCs, if a Command Descriptor requires the I3C Controller to re-send the Broadcast Address (per the conditions above).

If any Command Descriptor in the middle of such a sequence of multiple consecutive Command Descriptors results in an error, then the I3C Controller shall stop processing the sequence early, end the CCC framing and drive a STOP condition, and allow the Application to run an error recovery procedure.

Note:

After stopping the sequence due to an error, the I3C Controller shall leave the I3C Bus in a valid state (i.e., STOP condition) and wait for the Application to handle the error. This might require the Application to determine which error occurred and initiate an error recovery procedure (i.e., by driving an HDR Exit Pattern).

6.3.4 Mixing CCCs and Private Read/Write Transfers

The Application may also use field **TOC** to indicate continuous framing with multiple consecutive Command Descriptors, to compose larger transfers in CCC framing which might include CCCs in addition to Private Read/Write transfers, in any valid sequence allowed by SDR Mode framing. The I3C Controller shall inspect the field values for such Command Descriptors, and handle the transitions appropriately (i.e., from a CCC message/segment to a Private Read/Write transfer, and from a Private Read/Write transfer to a CCC message/segment) according to the flows for starting and ending CCC framing in SDR Mode.

In this manner, if a Command Descriptor's field **TOC** is 0, the I3C Controller shall drive a Repeated START at the end of the transfer or message/segment, and shall also drive other necessary framing elements, depending on the transfer type and related field values for this Command Descriptor as well as the next Command Descriptor.

- **Transition from Private Read/Write to CCC:** If this Command Descriptor indicated a Private Read/Write transfer, and the next Command Descriptor indicates a CCC, the I3C Controller shall start the CCC framing automatically, by driving a Repeated START condition, followed by the Broadcast Address (i.e., 7'h7E), then the Command Code in the next Command Descriptor.
  - If the next Command Descriptor indicates that a Defining Byte is present, the I3C Controller shall send the Defining Byte.
  - If the next Command Descriptor indicates a Direct CCC, then the I3C Controller shall drive another Repeated START condition to continue in CCC framing, followed by the Dynamic Address of the Target Address (or Group Address, if applicable) indicated in the next Command Descriptor, per CCC framing.
- **Transition from CCC to Private Read/Write:** If this Command Descriptor indicated a CCC, and the next Command Descriptor indicates a Private Read/Write transfer, the I3C Controller shall end the CCC framing automatically, in an appropriate manner for the End of CCC Command (as defined in the I3C Specification at ***Section 5.1.9.2.1 [MIPI02]***).
  - If this Command Descriptor was a Direct CCC, then the I3C Controller shall drive a Repeated START, followed by the Broadcast Address (i.e., 7'h7E), followed by another Repeated START to exit the Direct CCC framing in SDR Mode.
  - If this Command Descriptor was a Broadcast CCC, then the I3C Controller shall drive a Repeated START, since the I3C Bus does not remain in CCC framing after a Broadcast CCC.

Note:

Additional requirements shall also apply, per the I3C Controller's supported Command Descriptor format (see **Section 7***).*

6.3.5 Support for HDR Modes

This Managed CCC Framing Model may also be used to send CCCs in HDR Modes, if the I3C Controller supports transfers in at least one HDR Mode, and if the I3C Controller supports a Command Descriptor format that enables CCCs in HDR Modes.

Note:

> *If the I3C Controller does not support transfers in at least one HDR Mode or if it does not support a Command Descriptor format that enables CCCs in HDR Modes, then it shall only support CCC framing in SDR Mode. In this case, if the value of field **MODE** changes between two consecutive Command Descriptors in a sequence (i.e., if the first such Command Descriptor indicates SDR Mode and the value of field **TOC** is 1'b0) and the second Command Descriptor transitions from SDR Mode to a supported HDR Mode, then the I3C Bus Controller Logic shall automatically end the CCC framing in SDR Mode, and enter the indicated HDR Mode for generic HDR Write or HDR Read transfers (as per **Section 6.2.5**).*

An I3C Controller that fully supports Managed CCC Framing in one or more HDR Modes shall automatically enter the appropriate HDR Mode to send CCCs in that HDR Mode, when the Application enqueues a Managed CCC Transfer Command that indicates HDR Mode, using field **MODE**. This enables transition between I3C Modes, i.e., between SDR Mode and an HDR Mode. However use of such Transfer Commands does not support transitions between two different HDR Modes, or from HDR Mode to SDR Mode.

The Application may also use multiple consecutive Transfer Commands in any HDR Mode, in order to drive sequences of Broadcast CCCs, Direct CCCs, or any combination of both. This works similarly to the model for SDR Mode (see ***Section 6.3.2***): if a Transfer Command's field **TOC** is 0, then the I3C Controller shall drive the HDR Restart Pattern at the end of the message/segment. However, in any HDR Mode the Application must use the appropriate End of CCC Procedures, by enqueueing the special Transfer Command to end the CCC framing in that HDR Mode.

The I3C Controller shall also drive other necessary framing elements, depending on the transfer type and related field values for the supported Command Descriptor format as well as the next enqueued Command Descriptor.

- **Transition from Generic HDR-x Read/Write to HDR-x CCC:** If this Command Descriptor indicated a Generic HDR-*x* Read/Write transfer (i.e., not a Managed CCC Transfer Command type), and the next Command Descriptor is a Managed CCC Transfer Command, then the I3C Controller shall start the CCC framing automatically by driving the HDR Restart Pattern followed by the appropriate HDR-*x* Header Block of type 'Indicator' containing the Command Code and optional Defining Byte in the next Command Descriptor.
  - The I3C Controller shall detect this based on the changing values of field **CMD_ATTR**.
  - If the next Command Descriptor indicates that a Defining Byte is present, then the I3C Controller shall send the Defining Byte.
  - If the next Command Descriptor indicates a Direct CCC in HDR Mode, then the I3C Controller shall drive another HDR Restart Pattern to continue in CCC framing, followed by the appropriate HDR-*x* Header Block of type 'Selector' containing the Target Address (or Group Address, if applicable) indicated in the next Command Descriptor, per CCC framing.
- **Transition from HDR-x CCC to Generic HDR-*x* Read/Write:** If this Command Descriptor is a Managed CCC Transfer Command, and the next Command Descriptor indicates a Generic HDR-*x* Read/Write transfer (i.e., not a Managed CCC Transfer Command type), then the I3C Controller shall ensure that the CCC framing was ended appropriately.
  - **For all CCCs in HDR Modes:** The Application shall first end the framing by sending a special Managed CCC Transfer Command to end the CCC framing.
    - If the Application has not previously ended the framing (i.e., if this Command Descriptor is a CCC message/segment in an HDR Mode), then this is an error and the I3C Controller shall return an error code in the Response Descriptor for the next Command Descriptor. The Application shall accommodate this returned status in its error handling flow, using the error codes defined in

Section 6.4.1. The specific error code shall be 0xC (**Transfer Type Specific: CCC_FRAMING_NOT_ENDED**) in field **ERR_STATUS**.

- If the Application has ended the framing (i.e., if this Command Descriptor is a special Managed CCC Transfer Command to end the framing in that HDR Mode), then the I3C Controller and I3C Bus Controller Logic shall end the CCC framing automatically in an appropriate manner for the HDR-*x* End of CCC Procedure (as defined in the I3C Specification at ***Section 5.2.1.2 [MIPI02]***). The I3C Controller shall then proceed to the Generic HDR-*x* Read/Write transfer indicated by the next Command Descriptor.

For all the requirements defined for the flow above, the Application shall not change the value of field **MODE** in all such Command Descriptors, in order to ensure that the I3C Controller does indeed remain in the same HDR Mode. Note that the term "HDR-*x*" serves as a placeholder for a specific HDR Mode, and that the generic CCC flows in HDR Modes (as defined in the I3C Specification at ***Section 5.2.1.2 [MIPI02]***) serve as representative examples of the CCC flow requirements and definitions for specific HDR Modes.

For Managed CCC flows, several differences apply between SDR Mode and all HDR Modes:

- Broadcast CCCs in SDR Mode do not require a Managed CCC Transfer Command to end the CCC framing and return to Private Read/Write transfers. However, Broadcast CCCs in HDR Modes always require a special Managed CCC Transfer Command to end the CCC framing and return to Generic Read/Write transfers in that same HDR Mode.
- The specific End of CCC Procedure for a given HDR Mode may vary, given the circumstances. However, the Managed CCC Framing Model abstracts these differences and provides a special form of Managed CCC Transfer Command to use, regardless of the HDR Mode.

6.4 Error Handling

The general approach to error handling in the I3C Transfer Command/Response Interface is as follows:

- Any detectable error on a transaction (e.g., a NACK) will result in an error code in the Response Descriptor structure. Reception of the Response triggers an interrupt to the Application.

  In addition, the I3C Controller shall halt processing to allow the Application to handle the error.
  - The specific error code shall be returned in field **ERR_STATUS** of the Response Director, as defined in ***Section 7.1.3*** (for Format 1) and ***Section 7.2.3*** (for Format 2). Subsequent sections contain additional details on the defined error codes in field **ERR_STATUS**.
  - The Application is responsible for deciding how to handle the error. This includes invoking error recovery procedures, resuming processing from the next Transfer Command that was enqueued, retrying the Transfer Command that caused the error, or clearing (i.e., emptying) the Command Queue in order to receive new Transfer Commands from the Application.
- For Write requests, the Application should issue a Transfer Command with the **GETSTATUS** Direct CCC to determine the Device's overall error status.
- For In-Band Interrupt requests, the Application will report these in an Application-defined manner.

The I3C Controller shall not autonomously issue the **GETSTATUS** Direct CCC, nor issues any other command, to determine Device state/health. The Application shall issue **GETSTATUS** Direct CCC when required, and shall subsequently take appropriate measures to recover the Bus or the I3C Controller.

Upon any non-recoverable internal error, the I3C Controller shall stop processing of Transfer Commands and immediately report the error to the Application.

Note:

Specific Applications may choose to define additional error handling schemes on receiving a detectable error in a Response Descriptor, or via other means.

6.4.1 Error Status Codes in Response Descriptor

The error codes listed in ***Table 1*** and detailed in the following sub-sections are defined in ***Table 11*** for error status indication of a Command Transfer in Active Controller mode:

Table 1 Error Status Codes in Response Descriptor

Error Code	ERR_STATUS Value	Detailed in Sub-Section
CRC	0x1	***6.4.1.1***
PARITY	0x2	***6.4.1.2***
FRAME	0x3	***6.4.1.3***
ADDR_HEADER	0x4	***6.4.1.4***
NACK	0x5	***6.4.1.5***
OVL	0x6	***6.4.1.6***
I3C_SHORT_READ_ERR	0x7	***6.4.1.7***
HC_ABORTED	0x8	***6.4.1.8***
I2C_WR_DATA_NACK or **BUS_ABORTED**	0x9	***6.4.1.9***
NOT_SUPPORTED	0xA	***6.4.1.10***
ABORTED_WITH_CRC	0xB	***6.4.1.11***
Transfer Type Specific	0xC – 0xF	***6.4.1.12***

6.4.1.1 CRC

For field **ERR_STATUS** having a value of 0x1 (**CRC**):

- **In HDR-DDR Mode:**
  - For Generic Read transfers, the I3C Bus Controller Logic shall detect a CRC error by inspecting the CRC5 checksum provided by the addressed Target Device in the HDR-DDR CRC Word. If the provided checksum does not match the computed checksum for the received payload data, then this is a CRC error (per version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.2.2.4***).
- **In HDR-BT Mode:**
  - For Generic Read transfers, the I3C Bus Controller Logic shall detect a CRC error by inspecting the CRC checksum (either CRC-16 or CRC-32 according to the HDR-BT Header Block) provided by the addressed Target Device in the HDR-BT CRC Block. If the provided checksum does not match the computed checksum for the received payload data, then this is a CRC error (per version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.2.4.3.3***).
  - For Generic Write transfers, the I3C Bus Controller Logic shall detect cases where a Target Device indicates a CRC mismatch or a failure to verify the Write transfer, by watching SDA[0] during the **Transition_Verify** byte of the HDR-BT CRC Block. If the Target Device does not drive SDA[0] Low, then this indicates a CRC error or other failure to verify the Write transfer, as detected by the Target Device (per version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.2.4.3.3***).
    - If the Write transfer was allowed to complete normally, then the I3C Controller shall report this as a verification error due to CRC mismatch.
    - If the Write transfer was aborted early, then the I3C Controller shall report this as an Aborted with CRC Error (see ***Section 6.4.1.11***).

6.4.1.2 PARITY

For field **ERR_STATUS** having a value of 0x2 (**PARITY**):

- **In HDR-DDR Mode:**
  - For Generic Read transfers, the I3C Bus Controller Logic shall detect a Parity error by inspecting the Parity bits (i.e., PA1, PA0) for all HDR-DDR Words provided by the Target Device. If the provided Parity bits are not correct, then this is a Parity error per version 1.1.1 of the I3C Specification ***[MIPI06]*** at ***Section 5.2.2.4***.
- **In HDR-TSP/TSL Mode:**
  - For Generic Read transfers, the I3C Bus Controller Logic shall detect a Parity error by inspecting the Parity bits from the decoded HDR-Ternary Data Word. If the provided Parity bits are not correct, then this is a Parity error (per version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.2.3.4***).
- **In HDR-BT Mode:**
  - For Generic Read transfers, the I3C Bus Controller Logic shall detect a Parity error by inspecting Bit[3] in the **Transition_Control** byte provided by the addressed Target Device in an HDR-BT Data Block. If the provided parity bit does not match the computed parity bit for the other bits in the **Transition_Control** byte, then this is a Parity error (per version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.2.4.3.2***).
  - For Generic Read transfers, the I3C Bus Controller Logic shall detect a Parity error by inspecting Bit[3] in the **Control** byte provided by the addressed Target Device in the HDR-BT CRC Block. If the provided parity bit does not match the computed parity bit for the other bits in the **Control** byte, then this is a Parity error (per version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.2.4.3.3***).

6.4.1.3 FRAME

For field **ERR_STATUS** having a value of 0x3 (**FRAME**):

- **In HDR-DDR Mode:**
  - For Generic Read transfers, the I3C Bus Controller Logic shall detect a Framing error by inspecting the 2 Preamble bits (i.e., PRE1, PRE0) at the start of every HDR-DDR Word provided by the Target Device. If the Preamble bits are incorrect, or if the format and length of the HDR-DDR Word is malformed based on the indicated Preamble bits, then this is a Framing error (per version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.2.2.4***).
  - The I3C Bus Controller Logic shall also detect a Framing error if the provided HDR-DDR Words appear in an incorrect sequence for HDR-DDR framing, or if a CRC Word contains a first nibble having a value other than 4'hC, or if a Reserved Word is used.
- **In HDR-TSP/TSL Mode:**
  - For Generic Read transfers, the I3C Bus Controller Logic shall detect a Framing error if it receives certain types of illegal Bus activity driven by the Target Device. For example, if the I3C Bus Controller Logic receives two instances of Symbol 2 within the same HDR-Ternary Data Word, then this is a Framing error (per version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.2.3.4***).
- **In HDR-BT Mode:**
  - For Generic Read transfers, the I3C Bus Controller Logic shall detect a Framing error by inspecting the **Transition_Control** byte provided by the addressed Target Device in an HDR-BT Data Block. If the Park1 on SDA[0] is not 1, then this is a Framing error (per version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.2.4.3.2***).
  - For Generic Read transfers, the I3C Bus Controller Logic shall detect a Framing error by inspecting the **Control** byte provided by the addressed Target Device in the HDR-BT CRC Block. If any of the following conditions are detected, then this is a Framing error (per version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.2.4.3.3***):
    - If Provided Bit[0] is not 0;
    - If Provided Bit[1] is not 0;
    - If Provided Bit[5] does not have the same value that was sent in the Control Byte for the HDR-BT Header Block for this transfer; or
    - If Provided Bit[6] does not match the expected value (for the transaction expected to be terminated early, compared with the last HDR-BT Data Block marked as Last).

6.4.1.4 ADDR_HEADER

For field **ERR_STATUS** having a value of 0x4 (**ADDR_HEADER**):

- **In SDR Mode:**
  - If no Target Devices acknowledge the Broadcast Address (i.e., 7'h7E) driven automatically by the I3C Bus Controller Logic at the start of CCC framing (per ***Section 6.3***), then this is an Address Header error (see version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.1.9.1***).
  - If no Target Devices acknowledge the Broadcast Address before the start of a private Transfer Command, driven automatically by the I3C Bus Controller Logic after every START, then this is an Address Header error. The Application may configure the I3C Controller to use START / 7'h7E before every private Transfer Command, which allows for IBI reception (see ***Section 6.2.6***).
  - If no Target Devices acknowledge the Broadcast Address at any other time for SDR framing where it is appropriate for the I3C Bus Controller Logic to automatically drive the Broadcast Address for any reason, then this is an Address Header error.

    Note:

    In SDR Mode, any other I3C Target Device could attempt to drive its Dynamic Address onto the I3C Bus during the Arbitrable Address Header after a START (but not a Repeated START). In cases where another I3C Target Device's Dynamic Address wins the Address Arbitration (i.e., during an Interrupt Request or when sending the Broadcast Address), this shall not be an Address Header error. The I3C Controller shall process this as an incoming interrupt request, *and it shall either defer processing of the Transfer Command until after the STOP (and subsequent Bus Free Condition), or after the Repeated START following the Interrupt Request. Refer to the I3C Specification **[MIPI02]** at **Section 5.1.2.2**.*

- **In any HDR Modes:**
  - For generic transfers in any HDR Mode, if no Target Devices that support this specific HDR Mode actively acknowledge the HDR write command that is addressed to the Broadcast Address (i.e., 7'h7E), then this is an Address Header error. Note that an HDR write command addressed to the Broadcast Address is only used for CCCs in HDR Modes, per ***[MIPI02]*** at ***Section 5.2.2.2.1***.

6.4.1.5 NACK

For field **ERR_STATUS** having a value of 0x5 (**NACK**):

- **During Dynamic Address Assignment with ENTDAA:**
  - If any Target Device that responds to the **ENTDAA** procedure (i.e., that is initiated by an Application-specific Address Assignment Command) does not accept the Dynamic Address that it has been assigned, then this is a NACK error.
- **In SDR Mode:**
  - If an addressed Target Device does not acknowledge its Dynamic Address for a Private Write/Read transfer, or for a Direct CCC transfer (per ***Section 6.3***), and the maximum number of retries has been exceeded, then this is a NACK error.
  - Similar conditions apply to Private Write transfers and Direct SET CCC flows addressing a Group Address, if none of the Target Devices assigned to that Group Address acknowledge the transfer.
- **In any HDR Modes:**
  - If an addressed Target Device that supports this specific HDR Mode does not actively acknowledge its Dynamic Address in the appropriate Block or Word for that HDR Mode, for a Generic HDR Read/Write transfer or a Direct CCC flow in that HDR Mode, then this is a NACK error.
  - Similar conditions apply to Generic HDR Write transfers and Direct SET CCC flows addressing a Group Address, if none of the Target Devices assigned to that Group Address acknowledge the appropriate Block or Word in that HDR Mode.

6.4.1.6 OVL

For field **ERR_STATUS** having a value of 0x6 (**OVL**):

- This error code is reserved for Application-specific errors that deal with data underflow conditions (i.e., for Write-type transfers), or data overflow conditions (i.e., for Read-type transfers).
  - While this Specification does not define a data transfer mechanism for an Application, the implementer must define a data TX/RX for Write-type or Read-type transfers. If the I3C Controller runs out of Application-provided TX data during a Write-type transfer, or if the Application does not consume RX data during a Read-type transfer, the I3C Controller shall terminate the transfer and report this error.
- This error code may also be used for Application-specific errors that deal with special sequences of Transfer Commands.
  - This may include special framing of Transfer Commands (i.e., Atomic transactions) that might be framed by Application-defined Internal Control Commands for special purposes. Some of these uses could include reporting the failure of such an Atomic transaction if the I3C Controller experiences an unexpected Command Sequence Underflow error (i.e., if the Application is unable to provide all such Transfer Commands in time to prevent a stall or timeout on the I3C Bus).

6.4.1.7 I3C_SHORT_READ_ERR

For field **ERR_STATUS** having a value of 0x7 (**I3C_SHORT_READ_ERR**):

- **For I3C Read-Type transfers in any I3C Mode:**
  - The I3C Controller shall report this error code for I3C Read-Type transfers that did not return the requested number of data bytes (i.e., ‘short’ Read-Type transfers, per ***Section 6.2.7***) when the Transfer Command indicated that a ‘short’ Read-Type transfer was not permitted. The I3C Controller shall also halt operations and treat this as a transfer error.

Note:

*For Transfer Commands in I3C Mode, a ‘short’ Read-Type transfer, where the I3C Target Device returns fewer bytes than expected, does not necessarily indicate an I3C Bus error. However, since the Transfer Command indicated that a ‘short’ Read-Type transfer was not allowed (per **Section 7.1.2.2** and **Section 7.2.2.2**), the I3C Controller must treat this as an error and halt operations so that the Application can resolve the situation.*

6.4.1.8 HC_ABORTED

For field **ERR_STATUS** having a value of 0x8 (**HC_ABORTED**):

- The I3C Controller shall report this error code for a transaction that was aborted due to an abort operation that was sent by the Application during a transfer. The method for aborting a transaction is defined by the Application.

6.4.1.9 I2C_WR_DATA_NACK or BUS_ABORTED

For field **ERR_STATUS** having a value of 0x9, the interpretation of this error code depends on whether this is an I^{2}C transfer or an I3C transfer:

- **In SDR Mode for I^{2}C transfer speeds:**
  - The I3C Controller shall report error code **I2C_WR_DATA_NACK** for I^{2}C write data transactions, where the I^{2}C Target Device does not acknowledge the transfer.
- **For I3C transfers in SDR Mode or HDR Modes (except HDR-BT Mode):**
  - The I3C Controller shall report error code **BUS_ABORTED** for:
    - A transaction that was aborted due to Early Termination (which might have been caused by a Monitoring Device on the I3C Bus); or
    - A transaction that ended early, due to a Target Device not completing the data phase of the transfer. This is not the same as a I3C Controller initiated transfer abort (i.e., termination) that would be indicated by field **ERR_STATUS** having a value of 0x8 (**HC_ABORTED**).
- **For I3C transfers in HDR-BT Mode:**
  - For Generic Read transfers, the I3C Bus Controller Logic shall detect a transaction terminated early by inspecting the **Control** byte provided by the addressed Target Device in the HDR-BT CRC Block. If any of the following conditions are detected, then this is a Framing error (per version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.2.4.3.3***):
    - If Provided Bit[6] does not match the expected value for the transaction that was expected to be continued, where the previous HDR-BT Data Block was not marked as Last.
    - Any other method of aborting a Read transaction before expected completion.
  - For Generic Write transfers, the I3C Bus Controller Logic shall detect cases where a Target Device wishes to terminate the Write transaction early, and drives SDA[0] Low during the **Transition_Control** byte (i.e., the first byte) of the HDR-BT Data Block. If this happens, then the I3C Controller shall terminate the transmission and shall send an HDR-BT CRC Block (per version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.2.4.3.2***).
    - The I3C Controller shall report this as an aborted transaction with error code **BUS_ABORTED**, unless the Target Device also reports a CRC mismatch (see ***Section 6.4.1.1***). In that case, the I3C Controller shall report this as a combined error, i.e., Aborted with CRC (i.e., error code **ABORTED_WITH_CRC**, see ***Section 6.4.1.11***).

Note:

> *The I3C Controller shall report other values in field **ERR_STATUS** for Combo Transfer Commands if such Transfer Commands are aborted on the Bus, during phases other than the first phase (see **Section 6.4.1.11**, **Section 7.1.2.3**, and **Section 7.2.2.3**).*

6.4.1.10 NOT_SUPPORTED

For field **ERR_STATUS** having a value of 0xA (**NOT_SUPPORTED**):

- **In any mode:**
  - The I3C Controller shall report this error code for any Command Descriptor with an illegal or invalid combination of field values.

6.4.1.11 ABORTED_WITH_CRC

For field **ERR_STATUS** having a value of 0xB (**ABORTED_WITH_CRC**):

- **In HDR-BT Mode:**
  - For Generic Write transfers, the I3C Bus Controller Logic shall detect cases where a Target Device wishes to terminate the Write transaction early, and then subsequently indicates that the CRC that it received in the following HDR-BT CRC Block does not match the computed checksum for the received payload data (per ***Section 6.4.1.1***). In the special case where both of these events happen in the same Write transaction, the I3C Controller shall report this as an Aborted with CRC Error.

6.4.1.12 Transfer Type Specific

For field **ERR_STATUS** having a value of 0xC – 0xF (**Transfer Type Specific**):

These error codes are used for special variants of Transfer Commands, or in other operating modes.

- **While processing a Combo Transfer Command:**
  - The I3C Controller shall report error codes 0xC or 0xD if the I3C Bus Controller Logic encounters an error or unexpected condition during the second phase of a Combo transfer, as defined in ***Section 7.1.2.3*** and ***Section 7.2.2.3***. The specific error code depends on the type of error that the I3C Bus Controller Logic detects:
    - Error code 0xC (**Transfer Type Specific: COMBO_NACK_2ND**) is equivalent to **NACK** for the second phase.
    - Error code 0xD (**Transfer Type Specific: COMBO_BUS_ABORTED_2ND**) is equivalent to **BUS_ABORTED** for the second phase.
  - For Combo transfers, error codes 0x5 (**NACK**) and 0x9 (**BUS_ABORTED**) shall apply only to the first phase of the transfer (i.e., the initial Write operation).
- **While processing a sequence of Command Descriptors for an Atomic transaction:**
  - The I3C Controller shall report error codes 0xC or 0xD for an Application-defined special Internal Control Command that indicates the end of such a sequence, if the Atomic transaction failed due to early cancellation by the Application, or the Command Queue had an underflow condition before the end of the sequence (i.e., if the Application could not enqueue all such Command Descriptors in a timely manner, to maintain continuous framing and avoid a timeout). However, this I3C TCRI Specification does not define any specific Internal Control Commands for such purposes.
  - If the Command Queue underflow condition occurs, then any individual Transfer Commands that were rejected (i.e., not driven on the I3C Bus due to underflow) shall report error code 0x8 (**HC_ABORTED**).
  - For other transfer errors, the usual error codes shall apply to the individual Transfer Commands within the sequence.
- **While processing a sequence of Managed CCC Transfer Commands:**
  - The I3C Controller shall report error code 0xC (**CCC_FRAMING_NOT_ENDED**) if the next Command Descriptor is not a Managed CCC Transfer Command type, and the CCC framing has not been ended appropriately as defined in ***Section 6.3.4*** for SDR Mode and ***Section 6.3.5*** for HDR Modes. This error is only reported if the I3C Controller supports Managed CCC Transfer Commands.

6.4.2 Errors Due to Command Sequence Stall or Timeout

In some situations, the I3C Controller might fully drain the Command Queue while processing a command sequence where one or more Command Descriptors were enqueued into the Command Queue, with the last Command Descriptor being a Transfer Command having field **TOC**=0. As the I3C Bus Controller Logic started processing these enqueued Transfer Commands, the Application might not enqueue another suitable Command Descriptor having field **TOC**=1 in a timely manner, and the I3C Bus Controller Logic was forced to terminate the sequence.

- In SDR Mode, the I3C Bus Controller Logic might stall the Bus at certain points in SDR framing, according to the maximum allowed stall times defined for SCL Low (see the I3C Specification ***[MIPI02]*** at ***Section 5.1.2.5***).
- In HDR Modes, the I3C Bus Controller Logic might use a stall method that is specific to that HDR Mode, if supported and previously enabled for use.
  - If the stall method was not previously enabled, or if no stall method is supported, then the I3C Controller must terminate the transaction with the HDR Exit Pattern.

If a stall condition is permitted, then the I3C Bus Controller Logic might recover from a stall if the Application enqueues a new Command Descriptor before the maximum stall time, and if the Command Descriptor is a valid Transfer Command. However, if the stall condition continues to the maximum allowed time, then the I3C Bus Controller Logic must terminate the transaction with a STOP condition (for SDR Mode) or the HDR Exit Pattern (for HDR Modes).

If the I3C Bus Controller Logic detects such a condition that it has temporarily mitigated by stalling the clock, then the I3C Controller shall report this as a warning event, to the Application. However, if the condition is not resolved, and the stall turns into a timeout, then the I3C Controller shall report this as a possible error to the Application. Both such notifications should be high-priority interrupts, although the specifics of such an interface are not defined in this Specification.

The Application may subsequently resume sending valid Command Descriptors after the timeout, and the I3C Controller shall attempt to restart a new transaction in the indicated I3C Mode, starting with the next Transfer Command in the sequence, unless any of the following conditions are true:

- The Response Descriptor for the previous Transfer Command (i.e., the last one processed before the stall turned into a timeout) indicated a transfer error in field **ERR_STATUS**;
- The I3C Controller was processing Transfer Commands in a special Atomic transaction that had a timeout condition, and would reject any subsequent Transfer Commands after this timeout, until the Application sends a special command to end this transaction mode;
- The I3C Controller was configured by the Application to halt operations on any Command Sequence Timeout; or
- Any other transfer error or operational error was encountered.

If the I3C Controller is configured to halt on any Command Sequence Timeout, then it must halt processing and wait for the Application to resolve the condition using an error recovery procedure. This is useful when the Application sends sequences of Command Descriptors that are Transfer Commands which must be run in sequence using continuous framing in order to be successful, i.e., for a particular I3C content protocol that does not tolerate a disruption (i.e., forced end of the framing) if certain successive transfers are to be successful.

> For example, in cases where an unresolved stall becomes a timeout that causes a STOP condition or HDR Exit Pattern, this would disrupt the operation of the addressed Target(s), or have other side effects on the operation of addressed Target(s) that require subsequent Transfer Commands to be continuously executed with any preceding Transfer Commands. If so configured, then the I3C Controller shall not execute any subsequent Transfer Commands after the forced STOP condition or HDR Exit Pattern.

However, if the I3C Controller is not configured to halt on any Command Sequence Timeout, then it may generally resume processing without waiting for the Application to resolve the condition. This is useful when the Application sends standard Transfer Commands that may be executed separately or together, with the use of continuous framing as a performance optimization.

In this case, an unresolved stall that becomes a timeout shall still be reported as an error, but the I3C Controller shall resume operation upon receiving the next enqueued Command Descriptor, unless the I3C Controller was processing a special sequence of Transfer Commands that did not allow automatic resume of operations, such as an Atomic transaction.

Note:

The Application may also define special Atomic transactions that always require the I3C Controller to halt on any Command Sequence Timeout condition.

This page intentionally left blank.

7 Transfer Command/Response Structures

This Section describes the standard data structures used for the I3C Transfer Command/Response Interface. These are presented as data structures.

Two formats of structures are defined:

Table 2 Command/Response Formats

Format	Description	Size (DWORDs) (Command / Response)	Defined in Section
1	Legacy Format, Indexed with DAT	2 / 1	***7.1***
2	Legacy Format, Direct Addressed	2 / 1	***7.2***

Key differences between these Command/Response formats include:

- Whether the Command Descriptor for I3C transfers requires an index to a particular DAT entry for a specific I3C Target (i.e., Format 1), or whether it directly contains the address of the I3C Target for the transaction (i.e., Format 2).
  - **For Format 1:** The DAT index is provided in a field that is 5 bits in size, allowing for up to 32 unique DAT entries in the I3C Application that can be used in Transfer Commands.
  - **For Format 2:** The Target device is provided in a field that is 7 bits in size, and the Application directly provides the Target's Dynamic Address for transactions.
- Whether the Target type (i.e., I3C vs I^{2}C) must be defined in the DAT entry, or whether it is provided in a unique field (i.e., a single bit).
  - **For Format 1:** The Application's DAT must have a bit field in each DAT entry, set for each Target that has an entry and that can be addressed with a Transfer Command. The I3C Bus Controller logic uses the bit field in the indicated DAT entry.
  - **For Format 2:** The Application's DAT (if it has one) does not require such a bit field in each DAT entry, but the Transfer Command includes a field used by the I3C Bus Controller logic.

Note:

If the I3C Controller supports Format 2, then the I3C Application may (and should) still use a DAT to configure per-Target responses to particular I3C Bus events and conditions. When using Format 2, Legacy I^{2}C Devices do not typically require individual DAT entries. However, the scope of the functionality that can be controlled by a particular Application's DAT is not defined in this TCRI Specification.

In all other respects, Format 1 and Format 2 provide equivalent functionality and the Application may choose either one, depending on the use case requirements.

7.1 Format 1: Legacy Format, Indexed

The size of this Command Descriptor format is 2 DWORDs. This Command Descriptor format matches the format that was defined in version 1.1 of the I3C HCI Specification ***[MIPI05]***, which enables selected new features that were added in version 1.1+ of the I3C Specification ***[MIPI06]***.

The Command Descriptor is defined in ***Section 7.1.2*** and supports several types of Transfer Commands, as well as the Internal Control type command, as indicated by the **CMD_ATTR** field. Transfers are limited to 64 KB in size. For all Transfer Commands, the Command Descriptor includes a 5-bit **DEV_INDEX** field containing an index into the DAT table:

- **For private transfers and Direct CCCs:** The I3C Controller shall fetch the Dynamic Address from the DAT entry indicated by the **DEV_INDEX** field. The Application shall provide a valid index to a DAT entry that has been populated with a valid Dynamic Address of a Device on the I3C Bus (or an assigned Group Address for Write-type transfers, if supported).
- **For Broadcast CCCs**: The I3C Controller shall not use the **DEV_INDEX** field. The Application should set **DEV_INDEX** to 5’h00.

Note:

*This Command Descriptor format does not support transfers using some of the new features enabled by version 1.1+ of the I3C Specification **[MIPI06]**, such as HDR-BT (Bulk Transfer) Mode, Multi-Lane transfers in any supported I3C Modes, CCC flows in HDR Modes, or several types of Combo transfers that are necessary for specific use cases such as Device to Device Tunneling. A future version of this I3C TCRI Specification is expected to enable such features.*

7.1.1 Common Aspects of Transfer Commands

7.1.1.1 I3C Modes and Data Rates

An I3C Controller implementer may choose which I3C Modes to support. The following I3C Modes are provided as guidance.

Table 3 Supported I3C Transfer Modes

MODE Value	I3C Transfer Mode	Support
0x0 – 0x4	I3C SDR Mode	Required
0x5	I3C HDR-Ternary Modes	Optional
0x6	I3C HDR-DDR Mode	Optional
0x7	Reserved	–

An I3C Controller implementer may choose differing interpretations for the specific data rates for values of the **MODE** field, which are used by the various Transfer Command types that the I3C Controller supports.

Note:

*In this Command Descriptor format, the **MODE** field encodes both the transfer mode and the data rate.*

Depending on the implementation, the specific data rates for the available the options for I3C SDR Mode transfers might depend on the specific clock logic used within the I3C Controller, or other clock logic used within the System. The following guidelines are provided, based on the maximum values.

Table 4 Maximum Values for I3C SDR Data Transfer Speeds

MODE Field Value	Listed Speed	Maximum Sustainable Data Rate
0x0	I3C SDR0	12.5 MHz, Standard SDR Speed, f_{SCL} Max
0x1	I3C SDR1	8 MHz
0x2	I3C SDR2	6 MHz
0x3	I3C SDR3	4 MHz
0x4	I3C SDR4	2 MHz

Specific user-defined data rates for the available options for I^{2}C Mode transfers may also be implemented in the I3C Controller.

Table 5 Maximum Values for I^{2}C Data Transfer Speeds

MODE Field Value	Listed Speed	Maximum Sustainable Data Rate
0x0	I^{2}C FM	400 KHz, I^{2}C Fast Mode Speed, f_{SCL} Max
0x1	I^{2}C FM+	1 MHz, I^{2}C Fast Mode Plus Speed, f_{SCL} Max
0x2	I^{2}C UDR1	User Defined Data Rate 1
0x3	I^{2}C UDR2	User Defined Data Rate 2
0x4	I^{2}C UDR3	User Defined Data Rate 3

An I3C Controller implementer might choose to provide specific controls over the data rates used by the I3C Bus Controller Logic, for various values supported by the **MODE** field in the various Transfer Command types supported by the I3C Controller (see ***Section 7.1.2***). If such parameters need to be controlled by the Application, then the implementer should define an interface to access these parameters. Note that such parameters for I3C SDR Mode timing might also include separate fields to affect the different transfer speeds for the Open-Drain vs. Push-Pull phases of I3C transactions in SDR Mode, including:

- I3C Address Arbitration phase after a START condition
- I3C Address Header after a Repeated START condition
- Specific transfer rates for CCCs used during Bus Initialization and Dynamic Address Assignment:
  - **SETDASA** and **SETAASA** CCCs
  - **ENTDAA** CCC, including the various phases of the procedure with Dynamic Address Arbitration (per the I3C Specification ***[MIPI02]*** at ***Section 5.1.4.2***)

For the various data rates, an implementer should also determine whether the I3C Controller and its I3C Bus Controller Logic provide additional control over the clock speed (for I3C Pure Bus), or the duty cycle to limit the effective minimum data rate (for I3C Mixed Buses) for specific Applications where the system requires that the data rate stay above a given minimum transfer rate (per the I3C Specification ***[MIPI02]*** at ***Section 5.1.2.4.1***). If such control is needed, then the implementer should provide this control to the Application via an interface.

An implementer should also define whether the I3C Controller supports Controller Clock Stalling (per the I3C Specification ***[MIPI02]*** at ***Section 5.1.2.5***), and if so, whether such parameters that control Controller Clock Stalling need to be controlled by the Application. If so, then the implementer should provide this control to the Application via an interface.

7.1.1.2 Managed CCC Framing for Transfers

I3C Controllers that support this Command Descriptor format shall automatically support Managed CCC Framing for all Transfer Commands.

- **Transition from Private Read/Write to CCC:** If this Command Descriptor indicated a Private Read/Write transfer, and the next Command Descriptor indicates a CCC, the I3C Controller shall start the CCC framing automatically, as defined in ***Section 6.3.4***.
    - The I3C Controller shall detect whether the next Command Descriptor is a CCC based on the value of fields **MODE**, **CP** and **CMD**.
- **Transition from CCC to Private Read/Write:** If this Command Descriptor indicated a CCC, and the next Command Descriptor indicates a Private Read/Write transfer, the I3C Controller shall end the CCC framing automatically, in an appropriate manner for the End of CCC Command (as defined in the I3C Specification at ***Section 5.1.9.2.1 [MIPI02]***).
    - If this Command Descriptor was a Direct CCC, then the I3C Controller shall drive a Repeated START, followed by the Broadcast Address (i.e., 7'h7E), followed by another Repeated START to exit the Direct CCC framing in SDR Mode.

For all Transfer Commands that are CCCs, the Application should ensure that any DAT entries that are provided in the **DEV_INDEX** fields for such a sequence do indicate that the Target devices are I3C Devices.

Note:

The specific method of indicating whether a DAT entry describes an I3C Device versus a Legacy I^{2}C Device shall be defined by the Application. Legacy I^{2}C Target Devices do not support CCCs. Sending CCCs to Legacy I^{2}C Target Devices is not recommended.

7.1.2 Command Descriptor

The write-only Command Descriptor structure defines a transaction, including its parameters, and is sent by the Application to schedule a command to a Target Device on the I3C Bus while the I3C Controller is operating in Active Controller mode.

The Command Descriptor is 64 bits (2 DWORDs) in length, and supports a number of common transfer types (see ***Section 6.2***).

All I3C Transfer Commands can be grouped into the supported Command Types shown in ***Table 6***. This Specification defines a Command Descriptor structure for each listed Command Type, at the indicated Section. ***Table 6*** also shows the value of the **CMD_ATTR** field, for each listed Command Type.

Table 6 Supported Command Types for Command Descriptor, Format 1

Code	Command Type	Support	CMD_ATTR	Section
I	Immediate Data Transfer Command	Required	0x1	***7.1.2.1***
R	Regular Transfer Command	Required	0x0	***7.1.2.2***
C	Combo Transfer Command	Optional	0x3	***7.1.2.3***
M	Internal Control Command	N/A	0x7	N/A
A	Address Assignment Command	N/A	0x2	N/A

Figure 11 provides a high-level overview of the Command Types supported by the Command Descriptor for all supported I3C Commands, showing one row per Command Type. For the defined Command Types, field **DEV_INDEX** holds an index to a specific DAT entry for all transfer operations.

- Field **DEV_INDEX** is defined for all transfer Commands that use DAT entries for the Dynamic Address of the indicated Target Device. If Group Addresses are supported, then a DAT entry shall also be used for each such Group Address.

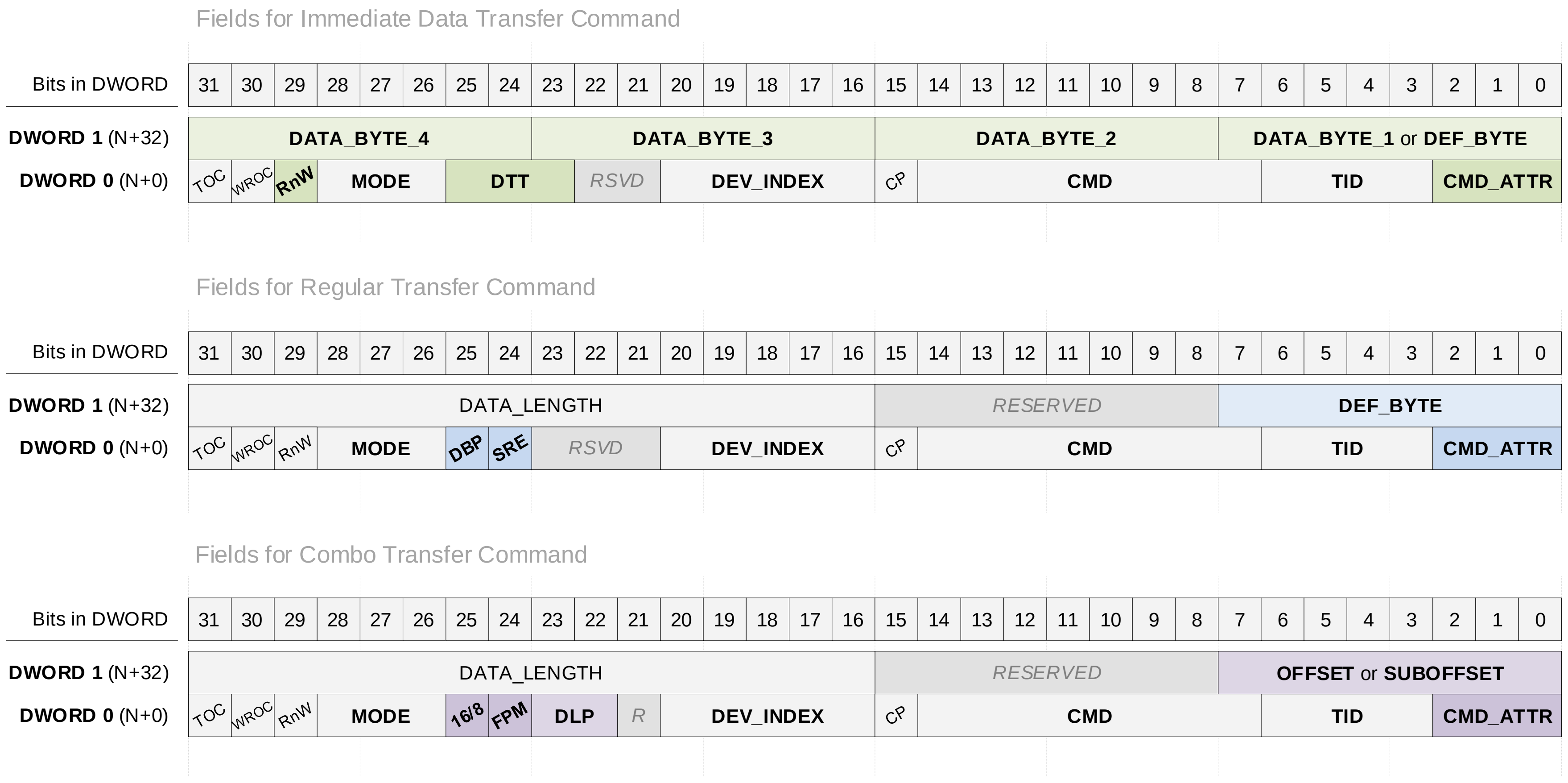

Figure 11 Overview of Supported Command Types for Command Descriptor, Format 1

Note:

*The Transfer Command formats for Format 1 are based on the formats that are defined in version 1.1 of the I3C HCI Specification **[MIPI05]**. Field **WROC** was formerly defined as field **ROC** in the I3C HCI Specification, but the meaning and intent are the same.*

7.1.2.1 Immediate Data Transfer Command

This section defines the Command Descriptor structure for Immediate Data Transfer commands.

This structure directly contains data bytes to be transferred, and as a result is only useful for shorter Write-Type transfers, or CCCs that write data (i.e., write segments for Direct CCCs or Broadcast CCC messages; see ***Section 7.1.2.1.1***). This structure shall not be used for Read-Type transfers (i.e., to receive data from a Device).

Note:

Immediate transfers are not CCC-specific, they can describe any transfer. The design intent is to provide the Application with a method for sending short, immediate transfers in order to reduce the number of transactions that would otherwise have to be made on internal busses.

Table 7 Immediate Data Transfer Command Structure

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
8 [63:56]	**DATA_BYTE_4**	W	0x0	**Immediate Data Transfer Data Byte 4** Direct argument
8 [55:48]	**DATA_BYTE_3**	W	0x0	**Immediate Data Transfer Data Byte 3** Direct argument
8 [47:40]	**DATA_BYTE_2**	W	0x0	**Immediate Data Transfer Data Byte 2** Direct argument
8 [39:32]	**DATA_BYTE_1 / DEF_BYTE**	W	0x0	**Immediate Data Transfer Data Byte 1** or **DefByte** Direct argument, optionally treated as the Defining Byte (and thus placed before the Device Address).
1 [31]	**TOC**	W	0x0	**Immediate Data Transfer Terminate on Completion** Controls what Bus condition is issued after completion of the data transfer. **Values:** • **1'b0: RESTART**: Issue Repeated START (Sr) at end of data transfer • **1'b1: STOP**: Issue Stop (P) at end of data transfer
1 [30]	**WROC**	W	0x0	**Immediate Data Transfer Response on Completion** Controls whether Response Status is required after successful completion of the data transfer. ***Note:*** *The Application may change the meaning of this field, or whether this field is used.* **Values:** • **1'b0: NOT_REQUIRED**: Response Status is not required • **1'b1: REQUIRED**: Response Status is required

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
1 [29]	**RNW**	W	0x0	**Immediate Data Transfer Direction (RnW)** Identifies direction of the transfer. This field shall always be set to 1'b0, because Immediate transfers are valid for Write transactions only. **Values:** • **1'b0: WRITE**: Write transfer • **1'b1:** Reserved, do not use
3 [28:26]	**MODE**	W	0x0	**Immediate Data Transfer Mode and Speed** Sets the Mode and speed for the I3C or I^{2}C transfer. Interpretation of this field depends on whether the Device is in I3C Mode vs. I^{2}C Mode (per the DAT Table entry indexed by field **DEV_INDEX**). **Values for I3C Mode:** • **0x0: I3C SDR0** Standard SDR Speed, f_{SCL} Max (up to 12.5 Mhz) • **0x1–0x4: I3C SDR1–SDR4** Reduced data rates (see ***Section 7.1.1***) • **0x5: I3C HDR-TSx** HDR-Ternary Mode • **0x6: I3C HDR-DDR** HDR Double Data Rate Mode • **0x7:** Reserved **Values for I^{2}C Mode:** • **0x0–0x4:** I^{2}C at supported data rates (see ***Section 7.1.1***) • **0x5–0x7:** Reserved
3 [25:23]	**DTT**	W	0x0	**Immediate Data Transfer Type and Byte Count** Number of valid data bytes to use in this Immediate Data Transfer Descriptor. Values 5-7 indicate that the first Data Byte shall be treated as the Defining Byte, for CCCs with a Defining Byte. ***Note:*** *This field should be set to a non-zero value, except for valid transfers with no data bytes that only require an ACK of a Dynamic Address (such as any Direct CCCs that do not have any payload defined). Broadcast CCCs with no subsequent bytes (such as **SETAASA**) are also valid examples of zero-byte payloads.* **Values:** • **0:** No payload • **1–4:** N bytes are valid • **5:** Defining Byte + 0 • **6:** Defining Byte + 1 • **7:** Defining Byte + 2
2 [22:21]	RESERVED	–	–	–

Size [Bits]	Field Name	Memory Access	Reset Value	Description
5 [20:16]	**DEV_INDEX**	W	0x0	**Immediate Data Transfer Device Index** Indicates the DAT table index for the Target Device being addressed with the transfer. The DAT entry indicated by this field must contain a valid Dynamic Address, for Read/Write transfers and for Direct CCCs. This field is ignored for Broadcast CCCs.
1 [15]	**CP**	W	0x0	**Immediate Data Transfer Command Present** Indicates whether field **CMD** is valid for a CCC or HDR Transfer. **Values:** • **1'b0: TRANSFER**: This structure describes an SDR transfer, so the **CMD** field is not valid. • **1'b1: CCC_HDR**: This structure describes a CCC or HDR transfer, so field **CMD** is valid.
8 [14:7]	**CMD**	W	0x0	**Immediate Data Transfer CCC / HDR Command Code Value** Specifies the I3C Command code. The interpretation of this field depends on field **CP**. **For CCC:** 8 bits (i.e., the Command Code, see ***Section 6.3***) **For HDR:** 7 bits (i.e., bits 13:7 are used for the lower 7 bits of the HDR-DDR or HDR-TSx Command Code, and the upper bit is determined by bit 29; bit 14 is ignored).
4 [6:3]	**TID**	W	0x0	**Immediate Data Transfer Transaction ID** Used as an identification tag for this command. This field shall be populated by the Application, and the same value shall be reflected in the Response Descriptor.
3 [2:0]	**CMD_ATTR**	W	0x0	**Immediate Data Transfer Command Attribute** Command Type, defining the format of the other fields. **Values:** • **0x1: IMMED_DATA_XFER**: Immediate Data Transfer • All other values are defined for other Command Types, or reserved for future use.

7.1.2.1.1 Usage for CCCs with Managed CCC Framing

An Immediate Data Transfer command that is used for a CCC with Managed CCC Framing (per ***Section 6.3***) shall have field **CP** set to 1’b1, and field **MODE** set to any valid value that indicates a transfer in SDR Mode.

The Application shall provide the I3C Common Command Code in field **CMD**. If the Transfer Command indicates that the CCC is sent with a Defining Byte, then the Application shall also set field **DTT** appropriately, per ***Table 8***. For Direct Write or Direct SET CCCs, field **DTT** shall also indicate the length of the data payload that is sent to the addressed I3C Target Address or Group Address, as part of Direct CCC framing (per ***Section 6.3***).

- If the CCC is sent with a Defining Byte, then bits 39:22 (i.e., field **DEF_BYTE**) indicates the Defining Byte value, and subsequent fields (i.e., fields **DATA_BYTE_2** and **DATA_BYTE_3**) may be used for optional first and second bytes (respectively) of the data payload.
- If the CCC is sent without a Defining byte, then bits 39:22 (i.e., field **DATA_BYTE_1**) may be used for the first byte of the optional data payload; and subsequent fields (i.e., **DATA_BYTE_2**, **DATA_BYTE_3** and **DATA_BYTE_4**) may be used for optional second, third and fourth bytes (respectively) of the data payload.

This Transfer Command type may also be used for a Broadcast CCC. In this case, the Application shall set field **DEV_INDEX** to zero, since the I3C Controller ignores this field for Broadcast CCCs.

Table 8 Immediate Data Transfer Command Usage for CCCs and Defining Bytes

Field DTT Value	CCC Sent with Defining Byte?	First Byte is in Field	Second Byte is in Field	Third Byte is in Field	Fourth Byte is in Field
0	No	–	–	–	–
1		DATA_BYTE_1	–	–	–
2		DATA_BYTE_1	DATA_BYTE_2	–	–
3		DATA_BYTE_1	DATA_BYTE_2	DATA_BYTE_3	–
4		DATA_BYTE_1	DATA_BYTE_2	DATA_BYTE_3	DATA_BYTE_4
5	Yes	–	–	–	–
6		DATA_BYTE_2	–	–	–
7		DATA_BYTE_2	DATA_BYTE_3	–	–

Note:

*An Immediate Data Transfer command that indicates a CCC with a Defining Byte does not support more than two data bytes for the data payload. In order to send a CCC with a Defining Byte and more than two data bytes for the data payload, the Application must use the Regular Data Transfer command (see **Section 7.1.2.2.1**).*

7.1.2.2 Regular Data Transfer Command

This section defines the Command Descriptor structure for Regular Data Transfer commands. This Transfer Command type may be used for any Private Read or Private Write, as well as many CCCs (per ***Section 7.1.2.2.1***).

The Command Descriptor structure for Regular Data Transfer commands indicates that the transfer should use a data buffer or queue, based on the operating mode of the Application.

Table 9 Regular Data Transfer Command Structure

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
16 [63:48]	**DATA_LENGTH**	W	0x0	**Data Transfer** **Data Length** Indicates the number of bytes to be transferred. ***Note:*** *This field should be set to a non-zero value. For valid transfers with no data bytes that only require an ACK of a Dynamic Address (such as any Direct CCCs that do not have payload defined), Immediate Data Transfer Commands should typically be used instead.*
8 [47:40]	RESERVED	–	–	–
8 [39:32]	**DEF_BYTE**	W	0x0	**Data Transfer** **Defining Byte for Present CCC** Valid if field **DBP** contains 1’b1
1 [31]	**TOC**	W	0x0	**Data Transfer** **Terminate on Completion** Controls what Bus condition will be issued after completion of the transfer, per ***Section 6.2.5***. **Values:** • **1’b0: RESTART**: Issue Repeated START (Sr) at end of transfer • **1’b1: STOP**: Issue Stop (P) at end of transfer
1 [30]	**WROC**	W	0x0	**Data Transfer** **Response on Completion** Controls whether Response Status is required after successful completion of the data transfer. ***Note:*** *The Application may change the meaning of this field, or whether this field is used.* **Values:** • **1’b0: NOT_REQUIRED**: Response Status is not required • **1’b1: REQUIRED**: Response Status is required

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
1 [29]	**RNW**	W	0x0	**Data Transfer Direction (RnW)** Identifies the direction of this transfer. **Values:** • **1'b0: WRITE**: Write transfer • **1'b1: READ**: Read transfer
3 [28:26]	**MODE**	W	0x0	**Data Transfer Speed and Mode** Sets the Mode and speed for the I3C or I^{2}C transfer. Interpretation of this field depends on whether the Device is in I3C Mode vs. I^{2}C Mode (per the DAT Table entry indexed by field **DEV_INDEX**). **Values for I3C Mode:** • **0x0: I3C SDR0** Standard SDR Speed, f_{SCL} Max (up to 12.5 MHz) • **0x1–0x4: I3C SDR1–SDR4** Reduced data rates (see ***Section 7.1.1***) • **0x5: I3C HDR-TSx** HDR-Ternary Mode • **0x6: I3C HDR-DDR** HDR Double Data Rate Mode • **0x7:** Reserved **Values for I^{2}C Mode:** • **0x0–0x4:** I^{2}C at supported data rates (see ***Section 7.1.1***) • **0x5–0x7:** Reserved
1 [25]	**DBP**	W	0x0	**Data Transfer Defining Byte for CCC Present** If this field contains 1'b1, then field **DEF_BYTE** contains the Defining Byte value.
1 [24]	**SHORT_READ_ERR**	W	0x0	**Data Transfer Short Read Is Error** Controls whether a 'short' Read-type transfer is permitted or treated as an error. ***Note:*** *This field is valid for I3C Read-type transfers only. For I3C Write-type transfers or I^{2}C transfers, the Application shall always set this field to 1'b0.* **Values (for I3C Read transfers):** • **1'b0: ALLOW_SHORT_READ**: All successful Read transfers are permitted, for lengths up to and including field **DATA_LENGTH**. • **1'b1: SHORT_READ_IS_ERROR**: Only allowed if field **RNW** is set to 1'b1. A 'short' Read transfer is not permitted, and will be treated as a transfer error. If the indicated Target Device does not return the number of bytes requested via field **DATA_LENGTH**, then this Read transfer shall be treated as an error and the I3C Controller shall halt after the end of the Read-type transfer (see ***Section 6.2.1***).

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
3 [23:21]	RESERVED	–	–	–
5 [20:16]	**DEV_INDEX**	W	0x0	**Data Transfer Device Index** Contains the DAT table index for the Target Device being addressed with the transfer. The DAT entry indicated by this field must contain a valid Dynamic Address, for Read/Write transfers and for Direct CCCs. This field is ignored for Broadcast CCCs.
1 [15]	**CP**	W	0x0	**Data Transfer Command Present** Indicates whether field **CMD** is valid for a CCC or HDR Transfer. **Values:** • **1'b0: TRANSFER**: This structure describes an SDR transfer, so the CMD field is not valid. • **1'b1: CCC_HDR**: This structure describes a CCC or HDR transfer, so field **CMD** is valid.
8 [14:7]	**CMD**	W	0x0	**Data Transfer CCC / HDR Command Code Value** Specifies the I3C Command code. The interpretation of this field depends on field **CP**. • **For CCC:** 8 bits (i.e., the Command Code; see ***Section 6.3***) • **For HDR:** 7 bits (i.e., bits 13:7 are used for the lower 7 bits of the HDR-DDR or HDR-TSx Command Code, and the upper bit is determined by bit 29; bit 14 is ignored).
4 [6:3]	**TID**	W	0x0	**Data Transfer Transaction ID** Identification tag for this command.
3 [2:0]	**CMD_ATTR**	W	0x0	**Data Transfer Command Attribute** Command Type, defining the format of the other fields. **Values:** • **0x0: XFER**: Regular Transfer • All other values are defined for other Command Types, or reserved for future use.

7.1.2.2.1 Usage for CCCs with Managed CCC Framing

A Regular Data Transfer command that is used for a CCC with Managed CCC Framing (per ***Section 6.3***) shall have field **CP** set to 1'b1, and field **MODE** set to any valid value that indicates a transfer in SDR Mode.

The Application shall provide the I3C Common Command Code in field **CMD**. If the Transfer Command indicates that the CCC is sent with a Defining Byte, then the Application shall also set field **DBP** to 1'b1 and provide the Defining Byte value in field **DEF_BYTE**. However, if the CCC is sent without a Defining Byte, then the Application shall set field **DBP** to 1'b0, and the value of field **DEF_BYTE** shall be ignored.

This Transfer Command type may be used for Broadcast CCC messages, or for any type of Direct CCC segments (i.e., a single segment per Transfer Command):

- For a Broadcast CCC message or a single Write segment of a Direct CCC (i.e., Direct Write or Direct SET), the Application shall set field **RNW** to indicate a Write-Type transfer, and shall provide the write data to the appropriate Queue/Buffer, per the operating mode.
- For a single Read segment of a Direct CCC (i.e., Direct Read or Direct GET), the Application shall set field **RNW** to indicate a Read-Type transfer, and the data returned from the indicated I3C Target Device shall be read from the appropriate Queue/Buffer, per the operating mode.
- Additionally, for a Broadcast CCC message, the Application shall set field **DEV_INDEX** to zero, since the I3C Controller ignores this field for Broadcast CCCs.

7.1.2.3 Combo Transfer (Write + Write/Read) Command

This section defines the Command Descriptor structure for Combo Transfer commands.

Each Combo Transfer Command Descriptor describes a combined Write + Read/Write operation. These two phases of the Combo Transfer shall be separated by an appropriate condition that restarts the framing:

- **For both phases in SDR Mode:** Repeated START.
- **For the first phase in SDR Mode and the second phase in an HDR Mode:** Repeated START, followed by the CCC to enter the appropriate HDR Mode (i.e., using the **ENTHDR0–ENTHDR7** CCCs, as defined in version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.1.9.3.9***).
- **For both phases in HDR Mode:** The HDR Restart Pattern.

The Combo Transfer Command Descriptor varies in format, depending on the particular transfer operation being requested on the I3C Bus:

- **SDR:**

  S + ADR + ACK + Offset + Sr + ADR + Data + P
- **HDR (16_BIT_SUBOFFSET == 1, Length first):**

  S + WrCmd + ADR + Length (16 bits) + Offset (16 bits) + Sr + CMD + ADR + Data + P
- **HDR (16_BIT_SUBOFFSET == 1, Length second):**

  S + WrCmd + ADR + Offset (16 bits) + Length (16 bits) + Sr + CMD + ADR + Data + P
- **HDR (16_BIT_SUBOFFSET == 1, no Length):**

  S + WrCmd + ADR + Offset (16 bits) + Sr + CMD + ADR + Data + P
- **HDR (16_BIT_SUBOFFSET == 0, Length first):**

  S + WrCmd + ADR + Length (8 bits) and Offset (8 bits) + Sr + CMD + ADR + Data + P
- **HDR (16_BIT_SUBOFFSET == 0, Length second):**

  S + WrCmd + ADR + Offset (8 bits) and Length (8 bits) + Sr + CMD + ADR + Data + P
- **HDR (16_BIT_SUBOFFSET == 0, no Length):**

  S + WrCmd + ADR + All 0s padding (8 bits) and Offset (8 bits) + Sr + CMD + ADR + Data + P

For Combo Transfers where the first phase is in an HDR Mode, the value of WrCMD is derived from the provided value of the 8-bit **CMD** field in the Command Descriptor. The I3C Controller shall mask off the high bit from the **CMD** field:

```
WrCmd = CMD & 0x7F
```

Note:

*Per the I3C Specification **[MIPI02]**, the command codes for Write are 0x00–0x7F, and for Read the command codes are 0x80–0xFF.*

If the **FIRST_PHASE_MODE** field is set to 0, then the first write on the I3C Bus is executed in SDR Mode (i.e., SDR0), and the speed and Mode of the second phase are determined by the **MODE** field.

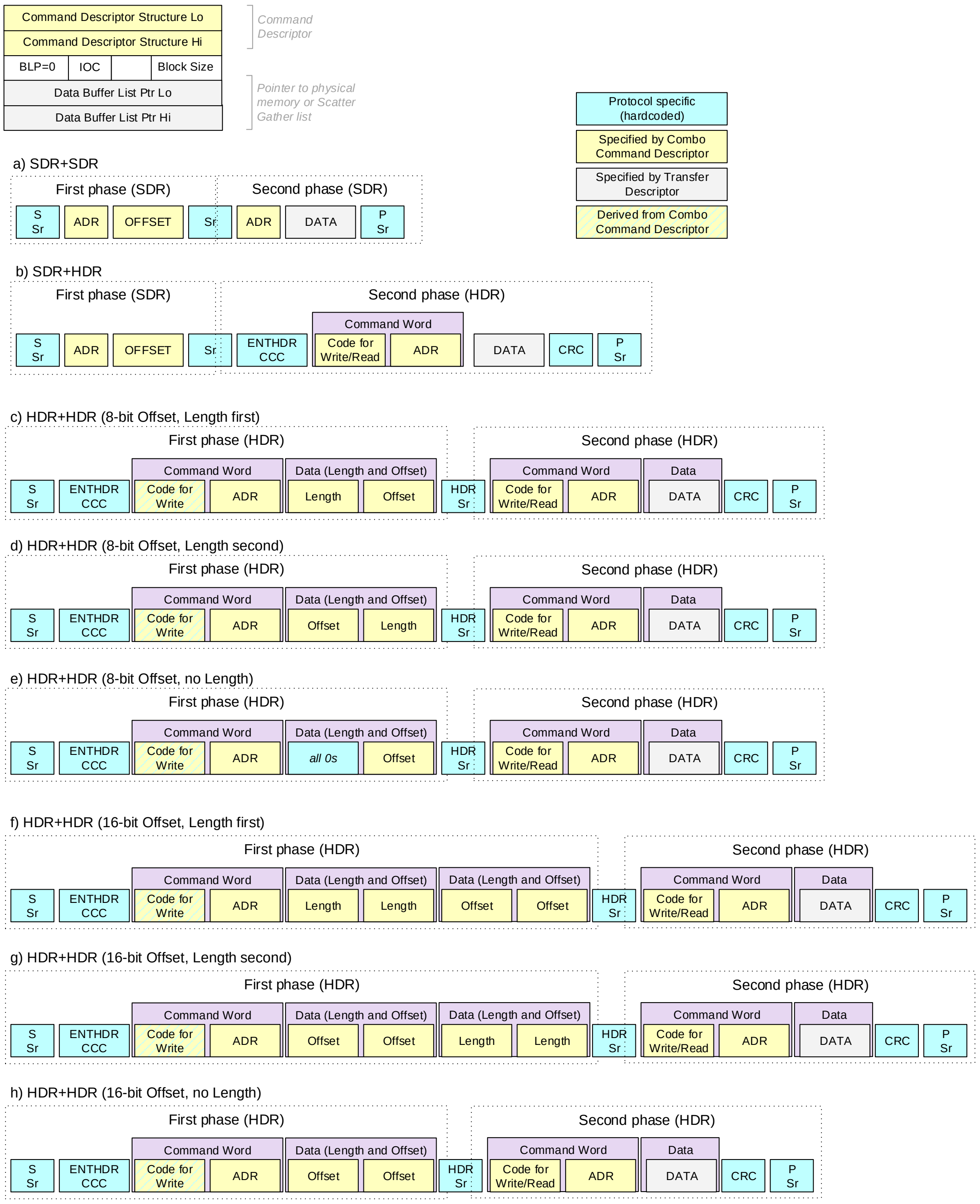
Command Descriptor Structure Lo
Command Descriptor Structure Hi
BLP=0
IOC
Block Size
Data Buffer List Ptr Lo
Data Buffer List Ptr Hi
Command Descriptor
Pointer to physical memory or Scatter Gather list
Protocol specific (hardcoded)
Specified by Combo Command Descriptor
Specified by Transfer Descriptor
Derived from Combo Command Descriptor
a) SDR+SDR
First phase (SDR)
Second phase (SDR)
S Sr
ADR
OFFSET
Sr
DATA
P Sr
b) SDR+HDR
Second phase (HDR)
ENTHDR CCC
Command Word
Code for Write/Read
CRC
c) HDR+HDR (8-bit Offset, Length first)
First phase (HDR)
Data (Length and Offset)
Code for Write
Length
Offset
HDR Sr
Data
d) HDR+HDR (8-bit Offset, Length second)
e) HDR+HDR (8-bit Offset, no Length)
all 0s
f) HDR+HDR (16-bit Offset, Length first)
g) HDR+HDR (16-bit Offset, Length second)
h) HDR+HDR (16-bit Offset, no Length)

Figure 12 I3C Bus Activity for Combo Transfer

In the case of a Combo Transfer Command failure due to a NACK or a transfer that was aborted early, the returned status in the Response Descriptor shall indicate the phase of the transaction in which the failure actually occurred. The Application shall accommodate this returned status in its error handling flow.

- For errors in the first phase of a Combo transfer:
  - If the indicated Target Device NACKs the Write, then the I3C Controller shall return an error code 0x5 (**NACK**) in field **ERR_STATUS**.
  - If the transfer is aborted early, then the I3C Controller shall return an error code 0x9 (**BUS_ABORTED**) in field **ERR_STATUS**.
- For errors in the second phase of a Combo transfer:
  - If the indicated Target Device NACKs the Write or Read as part of the second phase, then the I3C Controller shall return an error code 0xC (**Transfer Type Specific: COMBO_NACK_2ND**) in field **ERR_STATUS.**
  - If the second phase of the transfer is aborted early, then the I3C Controller shall return an error code 0xD (**Transfer Type Specific: COMBO_BUS_ABORTED_2ND**) in field **ERR_STATUS**.
  - These error codes are defined in ***Section 6.4.1***.

Table 10 Combo (Write + Write/Read) Transfer Command Structure

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
16 [63:48]	**DATA_LENGTH**	W	0x0	**Combo Transfer Data Length** Number of bytes to be transferred. ***Note:*** *This field must be set to non-zero value.*
16 [47:32]	**OFFSET / SUBOFFSET**	–	–	**Combo Transfer Offset / Sub-Offset** Offset of the target operation
1 [31]	**TOC**	W	0x0	**Combo Transfer Terminate on Completion** Controls what Bus condition is issued after completion of the transfer. **Values:** • **1'b0: RESTART**: Issue Repeated START (Sr) at end of transfer • **1'b1: STOP**: Issue Stop (P) at end of transfer
1 [30]	**WROC**	W	0x0	**Combo Transfer Response on Completion** Controls whether Response Status is required after successful completion of the Combo transfer. ***Note:*** *The Application may change the meaning of this field, or whether this field is used.* **Values:** • **1'b0: NOT_REQUIRED**: Response Status is not required • **1'b1: REQUIRED**: Response Status is required

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
1 [29]	**RNW**	W	0x0	**Combo Transfer Direction (RnW)** Sets transfer direction. This applies only to the second phase; the first phase is always a Write transfer. **Values:** • **1'b0: WRITE**: Second phase is a Write transfer • **1'b1: READ**: Second phase is a Read transfer. The Combo Transfer Command does not support 'short' Read transfers (see ***Section 6.2.1***).
3 [28:26]	**MODE**	W	0x0	**Combo Transfer Speed and Mode** Sets the Mode and speed for the I3C or I^{2}C transfer. Interpretation of this field depends on whether the Device is in I3C Mode vs. I^{2}C Mode (per the DAT Table entry indexed by field **DEV_INDEX**). **Values for I3C Mode:** • **0x0: I3C SDR0** Standard SDR Speed, f_{SCL} Max (up to 12.5 MHz) • **0x1–0x4: I3C SDR1–SDR4** Reduced data rates (see ***Section 7.1.1***) • **0x5: I3C HDR-TSx** HDR-Ternary Mode • **0x6: I3C HDR-DDR** HDR Double Data Rate Mode • **0x7:** Reserved **Values for I^{2}C Mode:** • **0x0–0x4:** I^{2}C at supported data rates (see ***Section 7.1.1***) • **0x5–0x7:** Reserved
1 [25]	**16_BIT_SUBOFFSET**	W	0x0	**Combo Transfer Sub Offset Size** Sets Sub-Offset length to 8 or 16 bits. **Values:** • **1'b0: 8_BIT_SUBOFFSET**: Sub-offset is 8 bits long. Value is encoded in Lower Byte of the **OFFSET / SUBOFFSET** field. • **1'b1: 16_BIT_SUBOFFSET**: Sub-offset is 16 bits long

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
1 [24]	**FIRST_PHASE_MODE**	W	0x0	**Combo Transfer First Phase Mode** Indicates whether the first phase of the Combo Transfer is executed in SDR Mode, vs. the Mode indicated by the **MODE** field. **Values:** • **1’b0: SDR**: First phase is executed in SDR Mode • **1’b1: MODE**: First phase is executed in the Mode indicated by the **MODE** field
2 [23:22]	**DATA_LENGTH_POSITION**	W	0x0	**Data Length Field Position** Indicates whether and where to put Data Length (**DATA_LENGTH**) in the first phase of the transfer. This field is only applicable if First phase of the transfer is executed in HDR Mode. Whether 8 bits vs. 16 bits of the Data Length field is used is indicated with field **16_BIT_SUBOFFSET**. In the case of 8 bits, it is encoded in Lower Byte of field **DATA_LENGTH**. **Values:** • **2’b0: NO**: Do not put Length field • **2’b1: FIRST**: Put Length as first field • **2’b2: SECOND**: Put Length as second field • **2’b3: RESERVED**: Don’t use
1 [21]	RESERVED	–	–	–
5 [20:16]	**DEV_INDEX**	W	0x0	**Combo Transfer Device Index** Contains the DAT table index for the Target Device being addressed with the transfer. The DAT entry indicated by this field must contain a valid Dynamic Address.
1 [15]	**CP**	W	0x0	**Combo Transfer Command Present** Indicates whether the **CMD** field is valid for a HDR Transfer. This field must be set to 1’b1 for a Combo transfer. **Values:** • **1’b0:** Invalid value for a Combo Transfer. • **1’b1:** If this is an HDR transfer, then field **CMD** is valid.

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
8 [14:7]	**CMD**	W	0x0	**Combo Transfer HDR Command Code Value** • **For HDR:** Contains the lower 7 bits of the HDR-DDR or HDR-TSx Command code (i.e., bits 13:7 are used, and bit 14 is ignored). The I3C Controller shall automatically use the correct upper bit for the second phase, as determined by bit 29. • **For SDR:** This field is reserved, and shall be set to zero.
4 [6:3]	**TID**	W	0x0	**Combo Transfer Transaction ID** Identification tag for the command.
3 [2:0]	**CMD_ATTR**	W	0x0	**Combo Transfer Command Attribute** Command Type, defining the format of the other fields. **Values:** • **0x3: WWR_COMBO_XFER**: Write + Write/Read Combo Transfer • All other values are defined for other Command Types, or reserved for future use.

7.1.2.4 Internal Control Command

Applications can also define specific Internal Control Commands that can be used for any purpose, including configuring the I3C Controller or other functional logic. Internal Control Commands can be enqueued on their own, or framed with other Transfer Commands as modifiers or qualifiers to how the I3C Controller or other functional logic processes transactions, initiates other actions or methods on the I3C Bus.

An Internal Control Command shares the same general format, with field **CMD_ATTR** set to 0x7. The remaining fields in the Command Descriptor are available for the Application to define for other purposes, and are not defined in this Specification.

7.1.2.5 Address Assignment Command

Applications can also define specific Address Assignment Commands that are used for the special task of Dynamic Address Assignment using any of its supported methods. Address Assignment Commands can be enqueued on their own, or in sequences. In most cases, Address Assignment Commands should not be intermixed with other Transfer Commands.

An Address Assignment Command shares the same general format, with field **CMD_ATTR** set to 0x2. The remaining fields in the Command Descriptor are available for the Application to define for other purposes, and are not defined in this Specification.

7.1.3 Response Descriptor

The Response Descriptor is a read-only structure describing the success or failure of a Transfer Command, and the amount of data transferred.

The Response Descriptor is 32 bits (i.e., 1 DWORD) in length.

Table 11 Response Descriptor Structure

Size [Bits]	Field Name	Memory Access	Reset Value	Description
4 [31:28]	**ERR_STATUS**	R	0x0	**Response Error Status** Indicates the Response status for the processed command (i.e., either success or the error type encountered). Error codes are described in ***Section 6.4.1***. **Values:** • **0x0: SUCCESS**: Transfer successful, no error. • **0x1: CRC**: CRC Error • **0x2: PARITY**: Parity Error • **0x3: FRAME**: Frame Error • **0x4: ADDR_HEADER**: Address Header Error • **0x5: NACK**: Address NACK'ed or Dynamic Address Assignment NACK'ed • **0x6: OVL**: Receive Overflow or Transfer Underflow Error • **0x7: I3C_SHORT_READ_ERR:** Target returned fewer data bytes than requested in field **DATA_LENGTH** of a Transfer Command that did not permit a 'short' read (per ***Section 6.2.1*** and ***Section 7.1.2.2***) • **0x8: HC_ABORTED**: Aborted (i.e., terminated) by I3C Controller due to internal error • **0x9:** Transfer terminated due to Bus action: • **For I^{2}C: I2C_WR_DATA_NACK**: Aborted due to NACK received during an I^{2}C Write Data transfer • **For I3C: BUS_ABORTED**: Aborted due to Early Termination, or Target not completing read or write of data phase of transfer • **0xA: NOT_SUPPORTED**: Command with specific parameters not supported by the I3C Controller implementation (e.g., specific Internal Control codes may not be supported) • **0xB: RESERVED** • **0xC–0xF: Transfer Type Specific Errors**, defined for specific transfer types
4 [27:24]	**TID**	R	0x0	**Command/Response Transaction ID** Identification tag for the command. This value shall match the value of field **TID**, for a previously enqueued Command Descriptor that was sent on the Bus. **Values:** • **0x0–0xF:** Valid Transaction IDs

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
8 [23:16]	RESERVED	–	–	–
16 [15:0]	**DATA_LENGTH**	R	0x0	**Data Length / Device Count** The meaning of this field depends on the context: **For Write Transfer:** Remaining data length (in bytes) **For Read Transfer:** Received data length (in bytes) **For Internal Control or Address Assignment:** Application may use for any purpose

7.2 Format 2: Legacy Format, Direct Addressed

The size of this Command Descriptor format is 2 DWORDs. This Command Descriptor format is derived from Format 1 (i.e., the format that was defined in version 1.1 of the I3C HCI Specification ***[MIPI05]***) and enables selected new features that were added in version 1.1+ of the I3C Specification ***[MIPI06]***.

The Command Descriptor is defined in ***Section 7.2.2*** and supports several types of Transfer Commands, as well as the Internal Control type command, as indicated by the **CMD_ATTR** field. Transfers are limited to 64 KB in size. For all Transfer Commands, the Command Descriptor includes a 7-bit **DEV_ADDRESS** field that directly addresses the Target:

- **For private transfers and Direct CCCs:** The I3C Controller directly uses the Target Address provided in the **DEV_ADDRESS** field. The Application shall provide a valid Dynamic Address of a Device on the I3C Bus (or an assigned Group Address for Write-type transfers, if supported).
- **For Broadcast CCCs**: The I3C Controller shall not use the **DEV_ADDRESS** field. The Application should set **DEV_ADDRESS** to 7'h00.

Note:

The Application may also choose to provide a DAT that allows for configuration of some or all Targets. However, the DAT entry is not used in this Command Descriptor format.

*This Command Descriptor format does not support transfers using some of the new features enabled by version 1.1+ of the I3C Specification **[MIPI06]**, such as HDR-BT (Bulk Transfer) Mode, Multi-Lane transfers in any supported I3C Modes, CCC flows in HDR Modes, or several types of Combo transfers that are necessary for specific use cases such as Device to Device Tunneling. A future version of this I3C TCRI Specification is expected to enable such features.*

7.2.1 Common Aspects of Transfer Commands

7.2.1.1 I3C Modes and Data Rates

An I3C Controller implementer may choose which I3C Modes to support. The following I3C Modes are provided as guidance.

Table 12 Supported I3C Transfer Modes

MODE Value	I3C Transfer Mode	Support
0x0 – 0x4	I3C SDR Mode	Required
0x5	I3C HDR-Ternary Modes	Optional
0x6	I3C HDR-DDR Mode	Optional
0x7	Reserved	–

An I3C Controller implementer may choose differing interpretations for the specific data rates for values of the **MODE** field, which are used by the various Transfer Command types that the I3C Controller supports.

Note:

*In this Command Descriptor format, the **MODE** field encodes both the transfer mode and the data rate, and the **I2C** field determines whether the indicated Target is an I3C Device or an I^{2}C Device.*

Depending on the implementation, the specific data rates for the available the options for I3C SDR Mode transfers might depend on the specific clock logic used within the I3C Controller, or other clock logic used within the System. The following guidelines are provided, based on the maximum values.

Table 13 Maximum Values for I3C SDR Data Transfer Speeds

MODE Field Value	Listed Speed	Maximum Sustainable Data Rate
0x0	I3C SDR0	12.5 MHz, Standard SDR Speed, f_{SCL} Max
0x1	I3C SDR1	8 MHz
0x2	I3C SDR2	6 MHz
0x3	I3C SDR3	4 MHz
0x4	I3C SDR4	2 MHz

Specific user-defined data rates for the available options for I^{2}C Mode transfers may also be implemented in the I3C Controller.

Table 14 Maximum Values for I^{2}C Data Transfer Speeds

MODE Field Value	Listed Speed	Maximum Sustainable Data Rate
0x0	I^{2}C FM	400 KHz, I^{2}C Fast Mode Speed, f_{SCL} Max
0x1	I^{2}C FM+	1 MHz, I^{2}C Fast Mode Plus Speed, f_{SCL} Max
0x2	I^{2}C UDR1	User Defined Data Rate 1
0x3	I^{2}C UDR2	User Defined Data Rate 2
0x4	I^{2}C UDR3	User Defined Data Rate 3

An I3C Controller implementer might choose to provide specific controls over the data rates used by the I3C Bus Controller Logic, for various values supported by the **MODE** field in the various Transfer Command types supported by the I3C Controller (see ***Section 7.2.2***). If such parameters need to be controlled by the Application, then the implementer should define an interface to access these parameters. Note that such parameters for I3C SDR Mode timing might also include separate fields to affect the different transfer speeds for the Open-Drain vs. Push-Pull phases of I3C transactions in SDR Mode, including:

- I3C Address Arbitration phase after a START condition
- I3C Address Header after a Repeated START condition
- Specific transfer rates for CCCs used during Bus Initialization and Dynamic Address Assignment:
  - **SETDASA** and **SETAASA** CCCs
  - **ENTDAA** CCC, including the various phases of the procedure with Dynamic Address Arbitration (per the I3C Specification ***[MIPI02]*** at ***Section 5.1.4.2***)

For the various data rates, an implementer should also determine whether the I3C Controller and its I3C Bus Controller Logic provide additional control over the clock speed (for I3C Pure Bus), or the duty cycle to limit the effective minimum data rate (for I3C Mixed Buses) for specific Applications where the system requires that the data rate stay above a given minimum transfer rate (per the I3C Specification ***[MIPI02]*** at ***Section 5.1.2.4.1***). If such control is needed, then the implementer should provide this control to the Application via an interface.

An implementer should also define whether the I3C Controller supports Controller Clock Stalling (per the I3C Specification ***[MIPI02]*** at ***Section 5.1.2.5***), and if so, whether such parameters that control Controller Clock Stalling need to be controlled by the Application. If so, then the implementer should provide this control to the Application via an interface.

7.2.1.2 Managed CCC Framing for Transfers

I3C Controllers that support this Command Descriptor format shall automatically support Managed CCC Framing for all Transfer Commands.

- **Transition from Private Read/Write to CCC:** If this Command Descriptor indicated a Private Read/Write transfer, and the next Command Descriptor indicates a CCC, the I3C Controller shall start the CCC framing automatically, as defined in ***Section 6.3.4***.
  - The I3C Controller shall detect whether the next Command Descriptor is a CCC based on the value of fields **MODE**, **CP**, and **CMD**.
- **Transition from CCC to Private Read/Write:** If this Command Descriptor indicated a CCC, and the next Command Descriptor indicates a Private Read/Write transfer, the I3C Controller shall end the CCC framing automatically, in an appropriate manner for the End of CCC Command (as defined in the I3C Specification at ***Section 5.1.9.2.1 [MIPI02]***).
  - If this Command Descriptor was a Direct CCC, then the I3C Controller shall drive a Repeated START, followed by the Broadcast Address (i.e., 7'h7E), followed by another Repeated START to exit the Direct CCC framing in SDR Mode.

For all Transfer Commands that are CCCs, the Application should ensure that the **DEV_ADDRESS** fields for such a sequence do indicate that the Target devices are I3C Devices (i.e., field **I2C** has a value of 1'b1).

Note:

Legacy I^2C Target Devices do not support CCCs. Sending CCCs to Legacy I^2C Target Devices is not recommended.

7.2.2 Command Descriptor

The write-only Command Descriptor structure defines a transaction, including its parameters, and is sent by the Application to schedule a command to a Target Device on the I3C Bus while the I3C Controller is operating in Active Controller mode.

The Command Descriptor is 64 bits (2 DWORDs) in length, and supports a number of common transfer types (see ***Section 6.2***).

All I3C Transfer Commands can be grouped into the supported Command Types shown in ***Table 15***. This Specification defines a Command Descriptor structure for each listed Command Type, at the indicated Section. ***Table 15*** also shows the value of the **CMD_ATTR** field, for each listed Command Type.

Table 15 Supported Command Types for Command Descriptor, Format 2

Code	Command Type	Support	CMD_ATTR	Section
I	Immediate Data Transfer Command	Required	0x1 (*)	***7.2.2.1***
R	Regular Transfer Command	Required	0x0 (*)	***7.2.2.2***
C	Combo Transfer Command	Optional	0x3 (*)	***7.2.2.3***
M	Internal Control Command	N/A	0x7	N/A
A	Address Assignment Command	N/A	0x2	N/A

Note:

*An Application might choose to support Format 2 in addition to Format 1, for specific operating contexts. If so, then the Application should map different values for the Direct addressed forms of Transfer Commands. See **Annex A, Section A.2** for more details.*

Figure 13 provides a high-level overview of the Command Types supported by the Command Descriptor for all supported I3C Commands, showing one row per Command Type. For the defined Command Types, field **DEV_ADDRESS** holds the Target's address for all transfer operations.

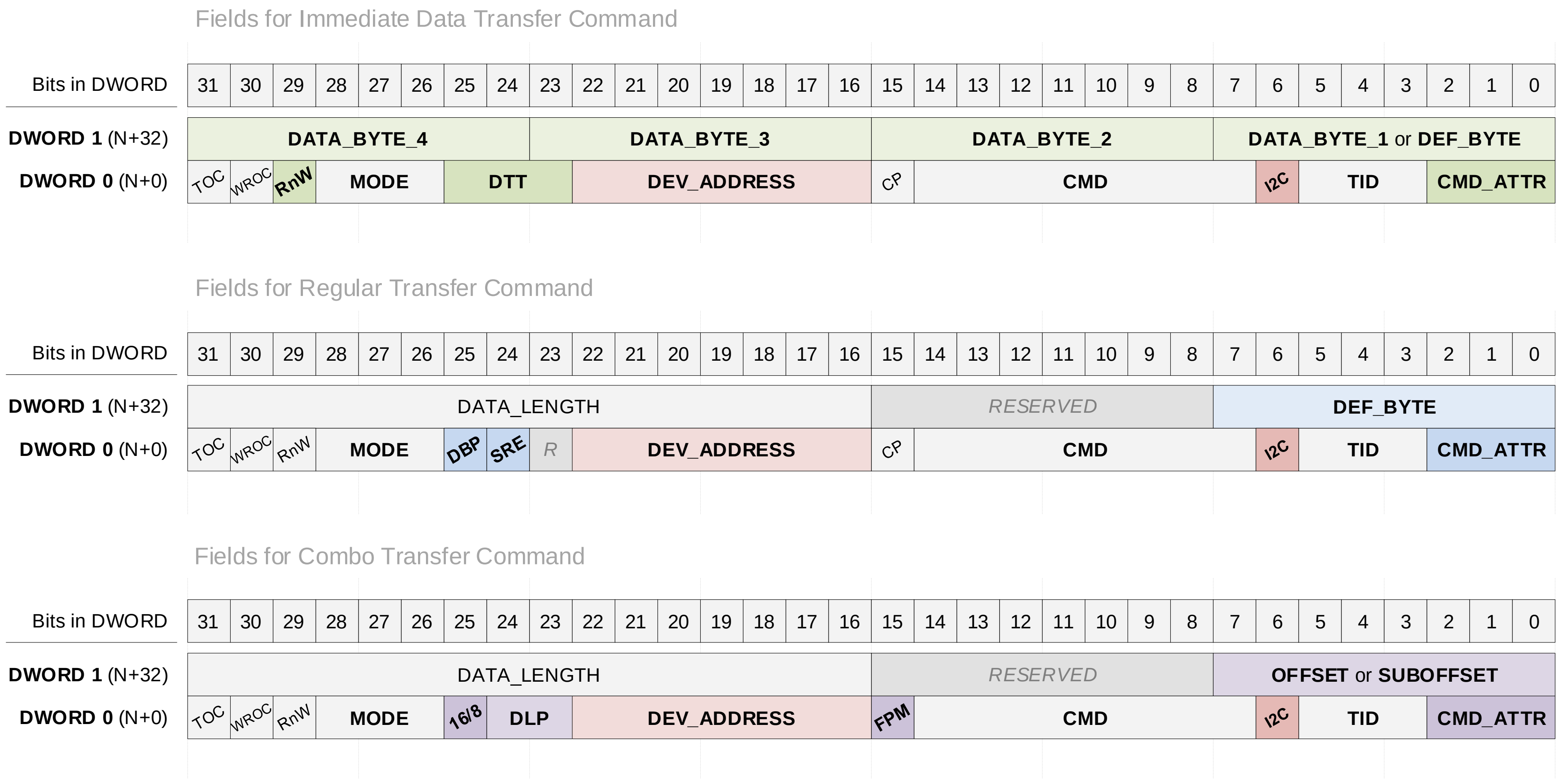

Figure 13 Overview of Supported Command Types for Command Descriptor, Format 2

Note:

*The Transfer Command formats for Format 2 are derived from the formats that are defined in version 1.1 of the I3C Host Controller Interface Specification **[MIPI05]**. These Transfer Commands are similar to Format 1, but some specific fields have been added or changed.*

7.2.2.1 Immediate Data Transfer Command

This section defines the Command Descriptor structure for Immediate Data Transfer commands.

This structure directly contains data bytes to be transferred, and as a result is only useful for shorter Write-Type transfers, or CCCs that write data (i.e., write segments for Direct CCCs or Broadcast CCC messages; see ***Section 7.2.2.1.1***). This structure shall not be used for Read-Type transfers (i.e., to receive data from a Device).

Note:

Immediate transfers are not CCC-specific, they can describe any transfer. The design intent is to provide the Application with a method for sending short, immediate transfers in order to reduce the number of transactions that would otherwise have to be made on internal busses.

Table 16 Immediate Data Transfer Command Structure

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
8 [63:56]	**DATA_BYTE_4**	W	0x0	**Immediate Data Transfer Data Byte 4** Direct argument
8 [55:48]	**DATA_BYTE_3**	W	0x0	**Immediate Data Transfer Data Byte 3** Direct argument
8 [47:40]	**DATA_BYTE_2**	W	0x0	**Immediate Data Transfer Data Byte 2** Direct argument
8 [39:32]	**DATA_BYTE_1 / DEF_BYTE**	W	0x0	**Immediate Data Transfer Data Byte 1** or **DefByte** Direct argument, optionally treated as the Defining Byte (and thus placed before the Device Address).
1 [31]	**TOC**	W	0x0	**Immediate Data Transfer Terminate on Completion** Controls what Bus condition is issued after completion of the data transfer. **Values:** • **1'b0: RESTART**: Issue Repeated START (Sr) at end of data transfer • **1'b1: STOP**: Issue Stop (P) at end of data transfer
1 [30]	**WROC**	W	0x0	**Immediate Data Transfer Response on Completion** Controls whether Response Status is required after successful completion of the data transfer. ***Note:*** *The Application may change the meaning of this field, or whether this field is used.* **Values:** • **1'b0: NOT_REQUIRED**: Response Status is not required • **1'b1: REQUIRED**: Response Status is required

Size [Bits]	Field Name	Memory Access	Reset Value	Description
1 [29]	**RNW**	W	0x0	**Immediate Data Transfer Direction (RnW)** Identifies direction of the transfer. This field shall always be set to 1'b0, because Immediate transfers are valid for Write transactions only. **Values:** • **1'b0: WRITE**: Write transfer • **1'b1:** Reserved, do not use
3 [28:26]	**MODE**	W	0x0	**Immediate Data Transfer Mode and Speed** Sets the Mode and speed for the I3C or I^2C transfer. Interpretation of this field depends on whether the Device is in I3C Mode vs. I^2C Mode (per the value in field **I2C**). **Values for I3C Mode:** • **0x0: I3C SDR0** Standard SDR Speed, f_{SCL} Max (up to 12.5 Mhz) • **0x1–0x4: I3C SDR1–SDR4** Reduced data rates (see ***Section 7.2.1***) • **0x5: I3C HDR-TSx** HDR-Ternary Mode • **0x6: I3C HDR-DDR** HDR Double Data Rate Mode • **0x7:** Reserved **Values for I^2C Mode:** • **0x0–0x4:** I^2C at supported data rates (see ***Section 7.2.1***) • **0x5–0x7:** Reserved
3 [25:23]	**DTT**	W	0x0	**Immediate Data Transfer Type and Byte Count** Number of valid data bytes to use in this Immediate Data Transfer Descriptor. Values 5-7 indicate that the first Data Byte shall be treated as the Defining Byte, for CCCs with a Defining Byte. ***Note:*** *This field should be set to a non-zero value, except for valid transfers with no data bytes that only require an ACK of a Dynamic Address (such as any Direct CCCs that do not have any payload defined). Broadcast CCCs with no subsequent bytes (such as **SETAASA**) are also valid examples of zero-byte payloads.* **Values:** • **0:** No payload • **1–4:** N bytes are valid • **5:** Defining Byte + 0 • **6:** Defining Byte + 1 • **7:** Defining Byte + 2

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
7 [22:16]	**DEV_ADDRESS**	W	0x0	**Immediate Data Transfer Device Address** Indicates the valid Dynamic Address, for Read/Write transfers and for Direct CCCs. This field is ignored for Broadcast CCCs.
1 [15]	**CP**	W	0x0	**Immediate Data Transfer Command Present** Indicates whether field **CMD** is valid for a CCC or HDR Transfer. **Values:** • **1'b0: TRANSFER**: This structure describes an SDR transfer, so the **CMD** field is not valid. • **1'b1: CCC_HDR**: This structure describes a CCC or HDR transfer, so field **CMD** is valid.
8 [14:7]	**CMD**	W	0x0	**Immediate Data Transfer CCC / HDR Command Code Value** Specifies the I3C Command code. The interpretation of this field depends on field **CP**. **For CCC:** 8 bits (i.e., the Command Code, see ***Section 6.3***) **For HDR:** 7 bits (i.e., bits 13:7 are used for the lower 7 bits of the HDR-DDR or HDR-TSx Command Code, and the upper bit is determined by bit 29; bit 14 is ignored).
1 [6]	**I2C**	W	0x0	**Immediate data Transfer Device Type** **Values:** • **1'b0**: **I3C**: I3C Device • **1'b1: I^{2}C**: Legacy I^{2}C Device
3 [5:3]	**TID**	W	0x0	**Immediate Data Transfer Transaction ID** Used as an identification tag for this command. This field shall be populated by the Application, and the same value shall be reflected in the Response Descriptor.
3 [2:0]	**CMD_ATTR**	W	0x0	**Immediate Data Transfer Command Attribute** Command Type, defining the format of the other fields. **Values:** • **0x1: IMMED_DATA_XFER**: Immediate Data Transfer • All other values are defined for other Command Types, or reserved for future use.

7.2.2.1.1 Usage for CCCs with Managed CCC Framing

An Immediate Data Transfer command that is used for a CCC with Managed CCC Framing (per ***Section 6.3***) shall have field **CP** set to 1'b1, and field **MODE** set to any valid value that indicates a transfer in SDR Mode. Field **I2C** should also be set to 1'b0.

The Application shall provide the I3C Common Command Code in field **CMD**. If the Transfer Command indicates that the CCC is sent with a Defining Byte, then the Application shall also set field **DTT** appropriately, per ***Table 8***. For Direct Write or Direct SET CCCs, field **DTT** shall also indicate the length of the data payload that is sent to the addressed I3C Target Address or Group Address, as part of Direct CCC framing (per ***Section 6.3***).

- If the CCC is sent with a Defining Byte, then bits 39:22 (i.e., field **DEF_BYTE**) indicates the Defining Byte value, and subsequent fields (i.e., fields **DATA_BYTE_2** and **DATA_BYTE_3**) may be used for optional first and second bytes (respectively) of the data payload.
- If the CCC is sent without a Defining byte, then bits 39:22 (i.e., field **DATA_BYTE_1**) may be used for the first byte of the optional data payload; and subsequent fields (i.e., **DATA_BYTE_2**, **DATA_BYTE_3** and **DATA_BYTE_4**) may be used for optional second, third and fourth bytes (respectively) of the data payload.

This Transfer Command may also be used for a Broadcast CCC. In this case, the Application shall set field **DEV_ADDRESS** to zero, since the I3C Controller ignores this field for Broadcast CCCs.

Table 17 Immediate Data Transfer Command Usage for CCCs and Defining Bytes

Field DTT Value	CCC Sent with Defining Byte?	First Byte is in Field	Second Byte is in Field	Third Byte is in Field	Fourth Byte is in Field
0	No	–	–	–	–
1		**DATA_BYTE_1**	–	–	–
2		**DATA_BYTE_1**	**DATA_BYTE_2**	–	–
3		**DATA_BYTE_1**	**DATA_BYTE_2**	**DATA_BYTE_3**	–
4		**DATA_BYTE_1**	**DATA_BYTE_2**	**DATA_BYTE_3**	**DATA_BYTE_4**
5	Yes	–	–	–	–
6		**DATA_BYTE_2**	–	–	–
7		**DATA_BYTE_2**	**DATA_BYTE_3**	–	–

Note:

*An Immediate Data Transfer command that indicates a CCC with a Defining Byte does not support more than two data bytes for the data payload. In order to send a CCC with a Defining Byte and more than two data bytes for the data payload, the Application must use the Regular Data Transfer command (see **Section 7.2.2.2.1**).*

7.2.2.2 Regular Data Transfer Command

This section defines the Command Descriptor structure for Regular Data Transfer commands. This Transfer Command type may be used for any Private Read or Private Write, as well as many CCCs (per ***Section 7.2.2.2.1***).

The Command Descriptor structure for Regular Data Transfer commands indicates that the transfer should use a data buffer or queue, based on the operating mode of the Application.

Table 18 Regular Data Transfer Command Structure

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
16 [63:48]	**DATA_LENGTH**	W	0x0	**Data Transfer** **Data Length** Indicates the number of bytes to be transferred. ***Note:*** *This field should be set to a non-zero value. For valid transfers with no data bytes that only require an ACK of a Dynamic Address (such as any Direct CCCs that do not have payload defined), Immediate Data Transfer Commands should typically be used instead.*
8 [47:40]	RESERVED	–	–	–
8 [39:32]	**DEF_BYTE**	W	0x0	**Data Transfer** **Defining Byte for Present CCC** Valid if field **DBP** contains 1'b1
1 [31]	**TOC**	W	0x0	**Data Transfer** **Terminate on Completion** Controls what Bus condition will be issued after completion of the transfer, per ***Section 6.2.5***. **Values:** • **1'b0: RESTART**: Issue Repeated START (Sr) at end of transfer • **1'b1: STOP**: Issue Stop (P) at end of transfer
1 [30]	**WROC**	W	0x0	**Data Transfer** **Response on Completion** Controls whether Response Status is required after successful completion of the data transfer. ***Note:*** *The Application may change the meaning of this field, or whether this field is used.* **Values:** • **1'b0: NOT_REQUIRED**: Response Status is not required • **1'b1: REQUIRED**: Response Status is required

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
1 [29]	**RNW**	W	0x0	**Data Transfer Direction (RnW)** Identifies the direction of this transfer. **Values:** • **1'b0: WRITE**: Write transfer • **1'b1: READ**: Read transfer
3 [28:26]	**MODE**	W	0x0	**Data Transfer Speed and Mode** Sets the Mode and speed for the I3C or I^{2}C transfer. Interpretation of this field depends on whether the Device is in I3C Mode vs. I^{2}C Mode (per the value in field **I2C**). **Values for I3C Mode:** • **0x0: I3C SDR0** Standard SDR Speed, f_{SCL} Max (up to 12.5 MHz) • **0x1–0x4: I3C SDR1–SDR4** Reduced data rates (see ***Section 7.2.1***) • **0x5: I3C HDR-TSx** HDR-Ternary Mode • **0x6: I3C HDR-DDR** HDR Double Data Rate Mode • **0x7:** Reserved **Values for I^{2}C Mode:** • **0x0–0x4:** I^{2}C at supported data rates (see ***Section 7.2.1***) • **0x5–0x7:** Reserved
1 [25]	**DBP**	W	0x0	**Data Transfer Defining Byte for CCC Present** If this field contains 1'b1, then field **DEF_BYTE** contains the Defining Byte value.
1 [24]	**SHORT_READ_ERR**	W	0x0	**Data Transfer Short Read Is Error** Controls whether a 'short' Read-type transfer is permitted or treated as an error. ***Note:*** *This field is valid for I3C Read-type transfers only. For I3C Write-type transfers or I^{2}C transfers, the Application shall always set this field to 1'b0.* **Values (for I3C Read transfers):** • **1'b0: ALLOW_SHORT_READ**: All successful Read transfers are permitted, for lengths up to and including field **DATA_LENGTH**. • **1'b1: SHORT_READ_IS_ERROR**: Only allowed if field **RNW** is set to 1'b1. A 'short' Read transfer is not permitted, and will be treated as a transfer error. If the indicated Target Device does not return the number of bytes requested via field **DATA_LENGTH**, then this Read transfer shall be treated as an error and the I3C Controller shall halt after the end of the Read-type transfer (see ***Section 6.2.1***).

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
1 [23]	RESERVED	–	–	–
7 [22:16]	**DEV_ADDRESS**	W	0x0	**Data Transfer Device Address** Indicates the valid Dynamic Address, for Read/Write transfers and for Direct CCCs. This field is ignored for Broadcast CCCs.
1 [15]	**CP**	W	0x0	**Data Transfer Command Present** Indicates whether field **CMD** is valid for a CCC or HDR Transfer. **Values:** • **1'b0: TRANSFER**: This structure describes an SDR transfer, so the CMD field is not valid. • **1'b1: CCC_HDR**: This structure describes a CCC or HDR transfer, so field **CMD** is valid.
8 [14:7]	**CMD**	W	0x0	**Data Transfer CCC / HDR Command Code Value** Specifies the I3C Command code. The interpretation of this field depends on field **CP**. • **For CCC:** 8 bits (i.e., the Command Code; see ***Section 6.3***) • **For HDR:** 7 bits (i.e., bits 13:7 are used for the lower 7 bits of the HDR-DDR or HDR-TSx Command Code, and the upper bit is determined by bit 29; bit 14 is ignored).
1 [6]	**I2C**	W	0x0	**Data Transfer Device Type** **Values:** • **1'b0: I3C**: I3C Device • **1'b1**: **I^{2}C**: Legacy I^{2}C Device
3 [5:3]	**TID**	W	0x0	**Data Transfer Transaction ID** Identification tag for this command.
3 [2:0]	**CMD_ATTR**	W	0x0	**Data Transfer Command Attribute** Command Type, defining the format of the other fields. **Values:** • **0x0: XFER**: Regular Transfer • All other values are defined for other Command Types, or reserved for future use.

7.2.2.2.1 Usage for CCCs with Managed CCC Framing

A Regular Data Transfer command that is used for a CCC with Managed CCC Framing (per ***Section 6.3***) shall have field **CP** set to 1'b1, and field **MODE** set to any valid value that indicates a transfer in SDR Mode.

The Application shall provide the I3C Common Command Code in field **CMD**. If the Transfer Command indicates that the CCC is sent with a Defining Byte, then the Application shall also set field **DBP** to 1'b1 and provide the Defining Byte value in field **DEF_BYTE**. However, if the CCC is sent without a Defining Byte, then the Application shall set field **DBP** to 1'b0, and the value of field **DEF_BYTE** shall be ignored.

This Transfer Command type may be used for Broadcast CCC messages, or for any type of Direct CCC segments (i.e., a single segment per Transfer Command):

- For a Broadcast CCC message or a single Write segment of a Direct CCC (i.e., Direct Write or Direct SET), the Application shall set field **RNW** to indicate a Write-Type transfer, and shall provide the write data to the appropriate Queue/Buffer, per the operating mode.
- For a single Read segment of a Direct CCC (i.e., Direct Read or Direct GET), the Application shall set field **RNW** to indicate a Read-Type transfer, and the data returned from the indicated I3C Target Device shall be read from the appropriate Queue/Buffer, per the operating mode.
- Additionally, for a Broadcast CCC message, the Application shall set field **DEV_ADDRESS** to zero, since the I3C Controller ignores this field for Broadcast CCCs.

7.2.2.3 Combo Transfer (Write + Write/Read) Command

This section defines the Command Descriptor structure for Combo Transfer commands.

Each Combo Transfer Command Descriptor describes a combined Write + Read/Write operation. These two phases of the Combo Transfer shall be separated by an appropriate condition that restarts the framing:

- **For both phases in SDR Mode:** Repeated START.
- **For the first phase in SDR Mode and the second phase in an HDR Mode:** Repeated START, followed by the CCC to enter the appropriate HDR Mode (i.e., using the **ENTHDR0–ENTHDR7** CCCs, as defined in version 1.1.1 of the I3C Specification ***[MIPI02]*** at ***Section 5.1.9.3.9***).
- **For both phases in HDR Mode:** The HDR Restart Pattern.

The Combo Transfer Command Descriptor varies in format, depending on the particular transfer operation being requested on the I3C Bus:

- **SDR:**

  S + ADR + ACK + Offset + Sr + ADR + Data + P
- **HDR (16_BIT_SUBOFFSET == 1, Length first):**

  S + WrCmd + ADR + Length (16 bits) + Offset (16 bits) + Sr + CMD + ADR + Data + P
- **HDR (16_BIT_SUBOFFSET == 1, Length second):**

  S + WrCmd + ADR + Offset (16 bits) + Length (16 bits) + Sr + CMD + ADR + Data + P
- **HDR (16_BIT_SUBOFFSET == 1, no Length):**

  S + WrCmd + ADR + Offset (16 bits) + Sr + CMD + ADR + Data + P
- **HDR (16_BIT_SUBOFFSET == 0, Length first):**

  S + WrCmd + ADR + Length (8 bits) and Offset (8 bits) + Sr + CMD + ADR + Data + P
- **HDR (16_BIT_SUBOFFSET == 0, Length second):**

  S + WrCmd + ADR + Offset (8 bits) and Length (8 bits) + Sr + CMD + ADR + Data + P
- **HDR (16_BIT_SUBOFFSET == 0, no Length):**

  S + WrCmd + ADR + All 0s padding (8 bits) and Offset (8 bits) + Sr + CMD + ADR + Data + P

For Combo Transfers where the first phase is in an HDR Mode, the value of WrCMD is derived from the provided value of the 8-bit **CMD** field in the Command Descriptor. The I3C Controller shall mask off the high bit from the **CMD** field:

```
WrCmd = CMD & 0x7F
```

Note:

Per the I3C Specification ***[MIPI02]****, the command codes for Write are 0x00–0x7F, and for Read the command codes are 0x80–0xFF.*

If the **FIRST_PHASE_MODE** field is set to 0, then the first write on the I3C Bus is executed in SDR Mode (i.e., SDR0), and the speed and Mode of the second phase are determined by the **MODE** field.

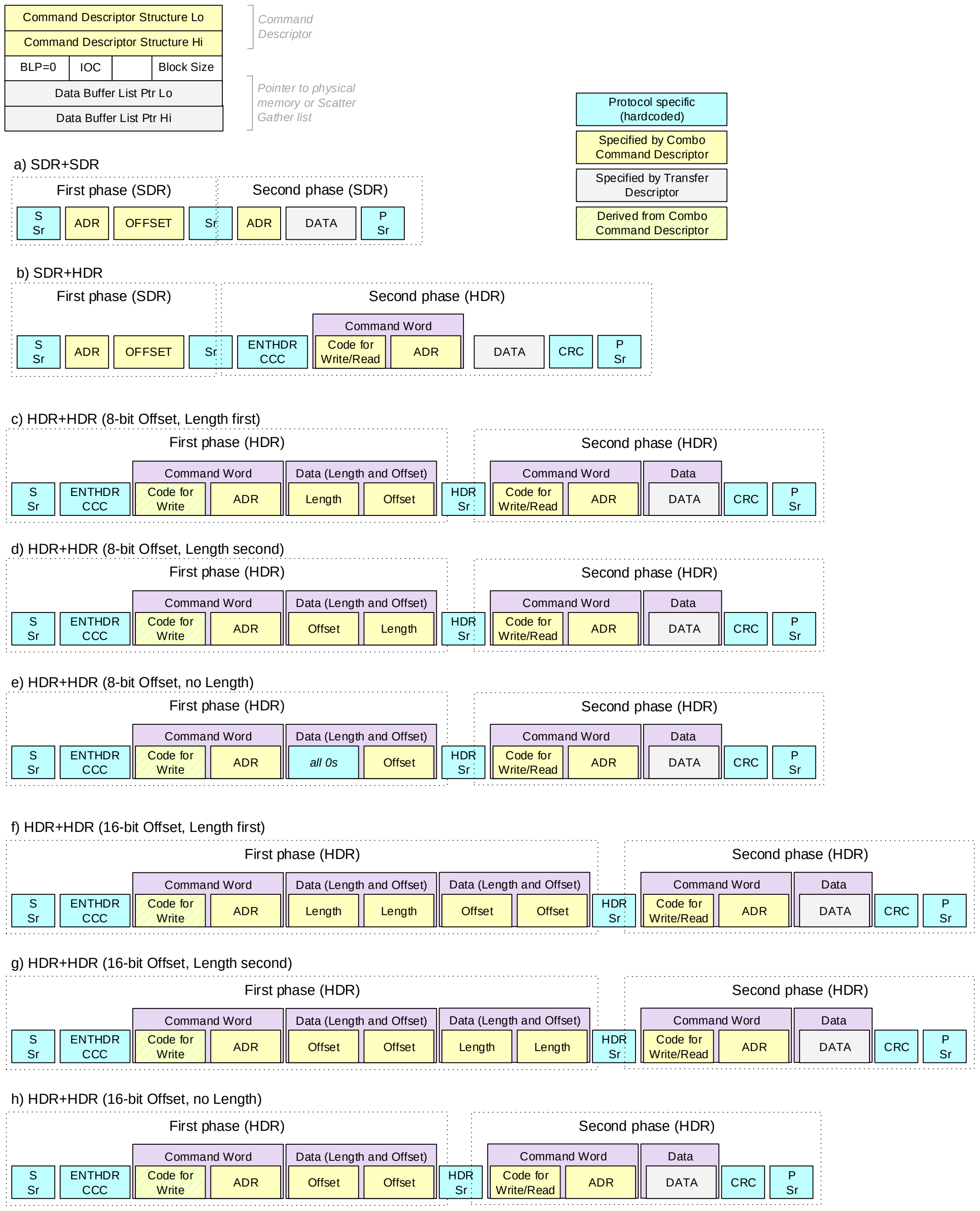
Command Descriptor Structure Lo
Command Descriptor Structure Hi
BLP=0
IOC
Block Size
Data Buffer List Ptr Lo
Data Buffer List Ptr Hi
Command Descriptor
Pointer to physical memory or Scatter Gather list
Protocol specific (hardcoded)
Specified by Combo Command Descriptor
Specified by Transfer Descriptor
Derived from Combo Command Descriptor
a) SDR+SDR
First phase (SDR)
Second phase (SDR)
S Sr
ADR
OFFSET
Sr
ADR
DATA
P Sr
b) SDR+HDR
Second phase (HDR)
ENTHDR CCC
Command Word
Code for Write/Read
CRC
c) HDR+HDR (8-bit Offset, Length first)
First phase (HDR)
Code for Write
Data (Length and Offset)
Length
Offset
HDR Sr
Data
d) HDR+HDR (8-bit Offset, Length second)
e) HDR+HDR (8-bit Offset, no Length)
all 0s
f) HDR+HDR (16-bit Offset, Length first)
g) HDR+HDR (16-bit Offset, Length second)
h) HDR+HDR (16-bit Offset, no Length)

Figure 14 I3C Bus Activity for Combo Transfer

In the case of a Combo Transfer Command failure due to a NACK or a transfer that was aborted early, the returned status in the Response Descriptor shall indicate the phase of the transaction in which the failure actually occurred. The Application shall accommodate this returned status in its error handling flow.

- For errors in the first phase of a Combo transfer:
  - If the indicated Target Device NACKs the Write, then the I3C Controller shall return an error code 0x5 (**NACK**) in field **ERR_STATUS**.
  - If the transfer is aborted early, then the I3C Controller shall return an error code 0x9 (**BUS_ABORTED**) in field **ERR_STATUS**.
- For errors in the second phase of a Combo transfer:
  - If the indicated Target Device NACKs the Write or Read as part of the second phase, then the I3C Controller shall return an error code 0xC (**Transfer Type Specific: COMBO_NACK_2ND**) in field **ERR_STATUS**.
  - If the second phase of the transfer is aborted early, then the I3C Controller shall return an error code 0xD (**Transfer Type Specific: COMBO_BUS_ABORTED_2ND**) in field **ERR_STATUS**.
  - These error codes are defined in ***Section 6.4.1***.

Table 19 Combo (Write + Write/Read) Transfer Command Structure

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
16 [63:48]	**DATA_LENGTH**	W	0x0	**Combo Transfer Data Length** Number of bytes to be transferred. ***Note:*** *This field must be set to non-zero value.*
16 [47:32]	**OFFSET / SUBOFFSET**	–	–	**Combo Transfer Offset / Sub-Offset** Offset of the target operation
1 [31]	**TOC**	W	0x0	**Combo Transfer Terminate on Completion** Controls what Bus condition is issued after completion of the transfer. **Values:** • **1'b0: RESTART**: Issue Repeated START (Sr) at end of transfer • **1'b1: STOP**: Issue Stop (P) at end of transfer
1 [30]	**WROC**	W	0x0	**Combo Transfer Response on Completion** Controls whether Response Status is required after successful completion of the combo transfer. ***Note:*** *The Application may change the meaning of this field, or whether this field is used.* **Values:** • **1'b0: NOT_REQUIRED**: Response Status is not required • **1'b1: REQUIRED**: Response Status is required

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
1 [29]	**RNW**	W	0x0	**Combo Transfer Direction (RnW)** Sets transfer direction. This applies only to the second phase; the first phase is always a Write transfer. **Values:** • **1'b0: WRITE**: Second phase is a Write transfer • **1'b1: READ**: Second phase is a Read transfer. The Combo Transfer Command does not support 'short' Read transfers (see ***Section 6.2.1***).
3 [28:26]	**MODE**	W	0x0	**Combo Transfer Speed and Mode** Sets the Mode and speed for the I3C or I^{2}C transfer. Interpretation of this field depends on whether the Device is in I3C Mode vs. I^{2}C Mode (per the value in field **I2C**). **Values for I3C Mode:** • **0x0: I3C SDR0** Standard SDR Speed, f_{SCL} Max (up to 12.5 MHz) • **0x1–0x4: I3C SDR1–SDR4** Reduced data rates (see ***Section 7.2.1***) • **0x5: I3C HDR-TSx** HDR-Ternary Mode • **0x6: I3C HDR-DDR** HDR Double Data Rate Mode • **0x7:** Reserved **Values for I^{2}C Mode:** • **0x0–0x4:** I^{2}C at supported data rates (see ***Section 7.2.1***) • **0x5–0x7:** Reserved
1 [25]	**16_BIT_SUBOFFSET**	W	0x0	**Combo Transfer Sub Offset Size** Sets Sub-Offset length to 8 or 16 bits. **Values:** • **1'b0: 8_BIT_SUBOFFSET**: Sub-offset is 8 bits long. Value is encoded in Lower Byte of the **OFFSET / SUBOFFSET** field. • **1'b1: 16_BIT_SUBOFFSET**: Sub-offset is 16 bits long

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
2 [24:23]	**DATA_LENGTH_POSITION**	W	0x0	**Data Length Field Position** Indicates whether and where to put Data Length (**DATA_LENGTH**) in the first phase of the transfer. This field is only applicable if First phase of the transfer is executed in HDR Mode. Whether 8 bits vs. 16 bits of the Data Length field is used is indicated with field **16_BIT_SUBOFFSET**. In the case of 8 bits, it is encoded in Lower Byte of field **DATA_LENGTH**. **Values:** • **2’b0: NO**: Do not put Length field • **2’b1: FIRST**: Put Length as first field • **2’b2: SECOND**: Put Length as second field • **2’b3: RESERVED**: Don’t use
7 [22:16]	**DEV_ADDRESS**	W	0x0	**Combo Transfer Device Address** Indicates the valid Dynamic Address, for Read/Write transfers and for Direct CCCs. This field is ignored for Broadcast CCCs.
1 [15]	**FIRST_PHASE_MODE**	W	0x0	**Combo Transfer First Phase Mode** Indicates whether the first phase of the Combo Transfer is executed in SDR Mode, vs. the Mode indicated by the **MODE** field. **Values:** • **1’b0: SDR**: First phase is executed in SDR Mode • **1’b1: MODE**: First phase is executed in the Mode indicated by the **MODE** field
8 [14:7]	**CMD**	W	0x0	**Combo Transfer HDR Command Code Value** • **For HDR:** Contains the lower 7 bits of the HDR-DDR or HDR-TSx Command code (i.e., bits 13:7 are used, and bit 14 is ignored). The I3C Controller shall automatically use the correct upper bit for the second phase, as determined by bit 29. • **For SDR:** This field is reserved, and shall be set to zero.
1 [6]	**I2C**	W	0x0	**Combo Transfer Device Type** **Values:** • **1’b0: I3C**: I3C Device • **1’b1: I^{2}C**: Legacy I^{2}C Device
3 [5:3]	**TID**	W	0x0	**Combo Transfer Transaction ID** Identification tag for the command.

Size [Bits]	Field Name	Memory Access	Reset Value	Description
3 [2:0]	**CMD_ATTR**	W	0x0	**Combo Transfer Command Attribute** Command Type, defining the format of the other fields. **Values:** • **0x3: WWR_COMBO_XFER**: Write + Write/Read Combo Transfer • All other values are defined for other Command Types, or reserved for future use.

Note:

*Compared with Format 1, Format 2 does not have an equivalent for field **CP**, since this field is not required for Combo Transfers.*

7.2.2.4 Internal Control Command

Applications can also define specific Internal Control Commands that can be used for any purpose, including configuring the I3C Controller or other functional logic. Internal Control Commands can be enqueued on their own, or framed with other Transfer Commands as modifiers or qualifiers to how the I3C Controller or other functional logic processes transactions, initiates other actions or methods on the I3C Bus.

An Internal Control Command shares the same general format, with field **CMD_ATTR** set to 0x7. The remaining fields in the Command descriptor are available for the Application to define for other purposes and are not defined in this Specification.

7.2.2.5 Address Assignment Command

Applications can also define specific Address Assignment Commands that are used for the special task of Dynamic Address Assignment using any of its supported methods. Address Assignment Commands can be enqueued on their own, or in sequences. In most cases, Address Assignment Commands should not be intermixed with other Transfer Commands.

An Address Assignment Command shares the same general format, with field **CMD_ATTR** set to 0x2. The remaining fields in the Command Descriptor are available for the Application to define for other purposes and are not defined in this Specification.

7.2.3 Response Descriptor

The Response Descriptor is a read-only structure describing the success or failure of a Transfer Command, and the amount of data transferred.

The Response Descriptor is 32 bits (i.e., 1 DWORD) in length.

Table 20 Response Descriptor Structure

Size [Bits]	Field Name	Memory Access	Reset Value	Description
4 [31:28]	**ERR_STATUS**	R	0x0	**Response Error Status** Indicates the Response status for the processed command (i.e., either success or the error type encountered). Error codes are described in ***Section 6.4.1***. **Values:** • **0x0: SUCCESS**: Transfer successful, no error. • **0x1: CRC**: CRC Error • **0x2: PARITY**: Parity Error • **0x3: FRAME**: Frame Error • **0x4: ADDR_HEADER**: Address Header Error • **0x5: NACK**: Address NACK'ed or Dynamic Address Assignment NACK'ed • **0x6: OVL**: Receive Overflow or Transfer Underflow Error • **0x7: I3C_SHORT_READ_ERR:** Target returned fewer data bytes than requested in field **DATA_LENGTH** of a Transfer Command that did not permit a 'short' read (per ***Section 6.2.1*** and ***Section 7.2.2.2***) • **0x8: HC_ABORTED**: Aborted (i.e., terminated) by I3C Controller due to internal error • **0x9:** Transfer terminated due to Bus action: ◦ **For I^{2}C: I2C_WR_DATA_NACK**: Aborted due to NACK received during an I^{2}C Write Data transfer ◦ **For I3C: BUS_ABORTED**: Aborted due to Early Termination, or Target not completing read or write of data phase of transfer • **0xA: NOT_SUPPORTED**: Command with specific parameters not supported by the I3C Controller implementation (e.g., specific Internal Control codes may not be supported) • **0xB: RESERVED** • **0xC–0xF: Transfer Type Specific Errors**, defined for specific transfer types
4 [27:24]	**TID**	R	0x0	**Command/Response Transaction ID** Identification tag for the command. This value shall match the value of field **TID**, for a previously enqueued Command Descriptor that was sent on the Bus. **Values:** • **0x0–0xF:** Valid Transaction IDs

Size [Bits]	**Field Name**	**Memory Access**	**Reset Value**	**Description**
8 [23:16]	RESERVED	–	–	–
16 [15:0]	**DATA_LENGTH**	R	0x0	**Data Length / Device Count** The meaning of this field depends on the context: **For Write Transfer:** Remaining data length (in bytes) **For Read Transfer:** Received data length (in bytes) **For Internal Control or Address Assignment:** Application may use for any purpose

Annex A Implementation Guidance

A.1 Application Capability Reporting

I3C Applications that conform to this I3C TCRI Specification should report the following capabilities and details, as part of exposing the functionality of the I3C Controller and its Transfer Command/Response Interface:

- **Which Command Descriptor format is supported** (per ***Section 7***), including:
  - The size in of the Command Descriptor and Response Descriptor (in DWORDs)
  - Whether it supports certain Transfer Command types in the Command Descriptor format that are defined as optional
  - Which I3C Modes (i.e., SDR or HDR) are supported, and whether there might be limitations on any I3C Transfer Command options that apply to certain modes
  - Whether it defines any specific Command Descriptor forms for Address Assignment Commands or Internal Control Commands
- **Specific methods for enqueueing Command Descriptors and dequeuing Response Descriptors**, including:
  - How to send TX Data (i.e., write into the data buffer for Transfer Commands that are Write-type transactions) and receive RX Data (i.e., read from data buffer for Transfer Commands that are Read-type transactions)
  - Whether the Application has a single operating context (i.e., entity that accesses the Command and Response queues) or whether it supports multiple operating contexts
- **How I3C In-Band Interrupts (IBIs) are received and reported to the Application**, including:
  - Whether the Application also supports special handling of special IBIs, including Pending Read Notifications (per ***[MIPI02] Section 5.1.6.2.2***) that might be processed autonomously
- **Whether other internal aspects of the I3C Controller are configurable**, either at initialization or during typical operation, including:
  - Internal registers, parameters or configuration fields that can be accessed and changed by the Application, or its upper layers (i.e., controlling software or other agents in a stack)
  - Whether the Application has optional methods for handling transitions between I3C Modes and transfer rates, or alternate handling of the **TOC** field (i.e., end of sequence) as mentioned in ***Section 6.2.5***
  - Whether the I3C Controller starts new transfers to I3C Devices with an initial Broadcast Address (i.e., START, 7’h7E, Repeated START) to ensure that I3C Targets have the opportunity to raise IBI Requests

The exact mechanism for reporting these capabilities and other details is not defined in this I3C TCRI Specification. However, Applications may use a variety of methods, including registers, descriptors (i.e., defined data structures) returned as part of detecting the I3C Controller and establishing a session, and other reporting or signaling interfaces.

A.2 Support for DAT-based and Direct Transfer Commands (Format 1 and Format 2)

An Application may choose to support both options (i.e., DAT-based and Direct Commands), meaning that it chooses to implement both Format 1 (see ***Section 7.1***) and Format 2 (***Section 7.2***) of the Command Descriptor and Response Descriptor. This might be useful for Applications that have limited DAT entries, or Applications that have multiple operating contexts (i.e., where the DAT is used for one context but not others).

In such Applications, these two Command and Response Descriptor formats can be interoperable, with some modifications. To aid in the decoding of Command Descriptor field **CMD_ATTR**, the following table provides guidance on field values to uniquely identify both DAT-based and Direct Transfer Commands.

Table 21 Guidance on CMD_ATTR Values for Command Descriptor Formats 1 and 2

Code	Command Type	Form	CMD_ATTR	Section
I	Immediate Data Transfer Command (Required)	DAT-based	0x1	***7.1.2.1***
		Direct	0x5 (*)	***7.2.2.1***
R	Regular Transfer Command (Required)	DAT-based	0x0	***7.1.2.2***
		Direct	0x4 (*)	***7.2.2.2***
C	Combo Transfer Command (Optional)	DAT-based	0x3	***7.1.2.3***
		Direct	0x6 (*)	***7.2.2.3***
M	Internal Control Command	N/A	0x7	N/A
A	Address Assignment Command	N/A	0x2	N/A

Note:

*In **Table 21** above, the Direct Transfer Command formats use different values for field* **CMD_ATTR** *than the values defined in **Table 15**. This allows the I3C Controller to distinguish whether the enqueued Transfer Command is referring to a DAT entry by its index, or whether it directly contains the Target's address. The Application could choose to allow both sets of Transfer Command types to be enqueued at any time, or the Application might choose to reserve certain Transfer Command types for specific operating contexts (i.e., the primary set of queues vs. other autonomous logic within the Application).*

Participants

The list below includes those persons who participated in the Working Group that developed this Specification and who consented to appear on this list.

Guruprasad Ardhanari, Intel Corporation
Mohammad Asad Javed, Intel Corporation
Rajesh Bhaskar, Intel Corporation
Anamitra Chakrabarti, Synopsys, Inc.
Rob Gough, Intel Corporation
Chris Grigg, MIPI Alliance (Team)
Takayuki Hirama, Sony Group Corporation
Paul Kimelman, NXP Semiconductors
Makoto Nariya, Sony Group Corporation
Pratap Neelashetty, Synopsys, Inc.
Laura Nixon, MIPI Alliance (Team)
Tomasz Pielaszkiewicz, Intel Corporation

2018

Radu Pitigoi-Aron, Qualcomm
Nicolas Pitre, BayLibre
Guruprasad Ramachandra, Synopsys, Inc.
Rob Santoro, MIPI Alliance (Team)
Matthew Schnoor, Intel Corporation
Katherine Valenti, MIPI Alliance (Team)
Suresh Venkatachalam, Synopsys, Inc.
Kasper Wszolek, Intel Corporation
Qijie Yang, Intel Corporation
Tadaaki Yuba, Sony Group Corporation
Fred Zhou, Intel Corporation

This page intentionally left blank.