Specification for
I3C Basic[SM]

Improved Inter Integrated Circuit

Version 1.0 – 19 July 2018

MIPI Board Adopted 8 October 2018

* NOTE TO IMPLEMENTERS *

This document is a MIPI Specification. MIPI member companies' rights and obligations apply to this MIPI Specification as defined in the MIPI Membership Agreement and MIPI Bylaws.

Non-members' rights and obligations are described in the *Terms and Conditions for Download and Implementation of the MIPI I3C Basic Specification v1.0* found on the MIPI publicly available web page where this specification is made available.

This page intentionally left blank.

Specification for I3C Basic[SM]

Improved Inter Integrated Circuit – Basic

Version 1.0

19 July 2018

MIPI Board Adopted 8 October 2018

NOTICE OF DISCLAIMER

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

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

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

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

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

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

Contents

Figures

Tables

Release History

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

This page intentionally left blank.

Preface to the I3C Basic Specification

1. What is MIPI I3C Basic?

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

2. Motivation

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

3. IPR Status

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

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

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

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

4. Relationship to MIPI I3C v1.x Specifications

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

4.1 I3C v1.0 Functions Not Included in I3C Basic

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

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

4.2 I3C v1.1 Functions Included in I3C Basic

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

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

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

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

1 Introduction

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

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 situation is becoming 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.

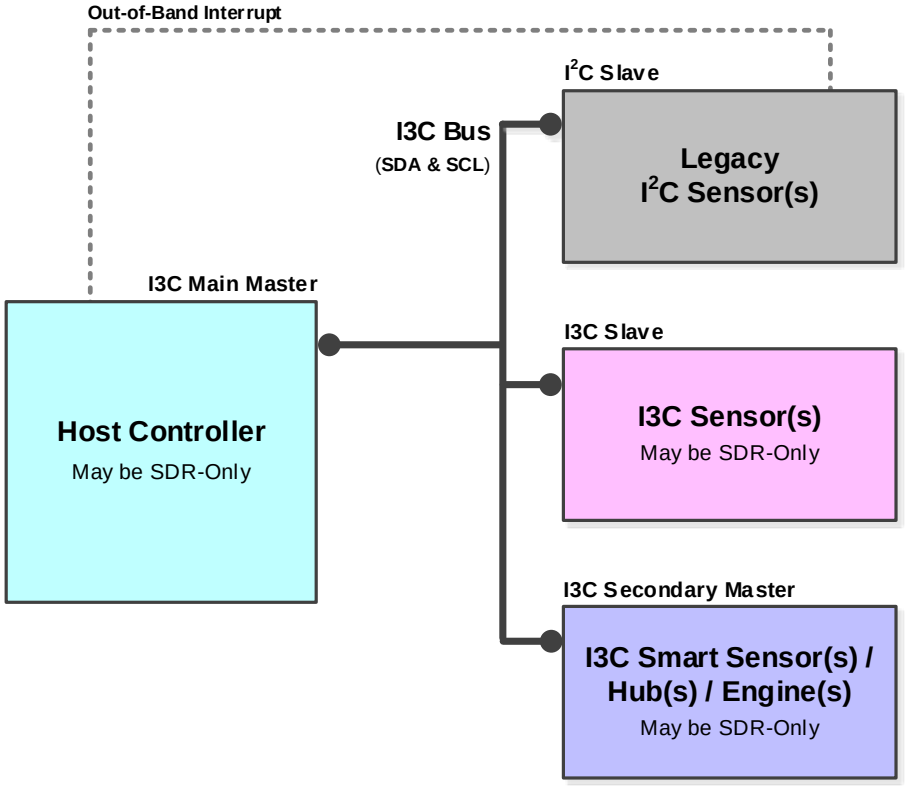


Figure 1 I3C System Diagram

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

Table 1 Sensor Classes Addressed by I3C

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

1.1 Scope

The following topics are in scope for this Specification:

- I3C interface protocols and commands leveraged for I3C Basic
- Electrical specifications, such as timing and voltage levels
- Support for specific classes of sensors and other Devices

The following topics are out of scope for this Specification:

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

1.2 I3C Purpose

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

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

1.3 I3C Key Features

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

Therefore, the I3C interface features:

- Two wire serial interface up to 12.5 MHz using Push-Pull
- Legacy I^2C Device co-existence on the same Bus (with some limitations)
- Dynamic Addressing while supporting Static Addressing for Legacy I^2C Devices
- Legacy I^2C messaging
- I^2C-like Single Data Rate messaging (SDR)
- **NOT SUPPORTED IN I3C BASIC:** Optional High Data Rate messaging Modes (HDR)
- Multi-Drop capability
- Multi-Master capability
- In-Band Interrupt support
- Hot-Join support
- **NOT SUPPORTED IN I3C BASIC:** Synchronous Timing Support and Asynchronous Time Stamping

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

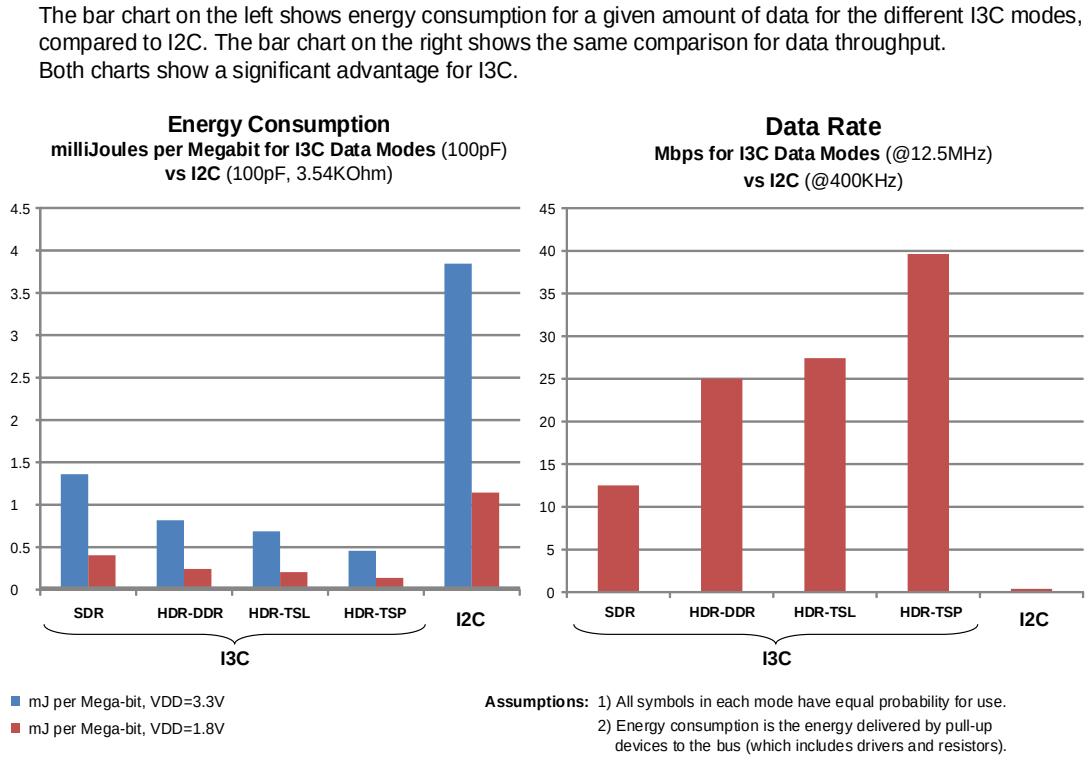
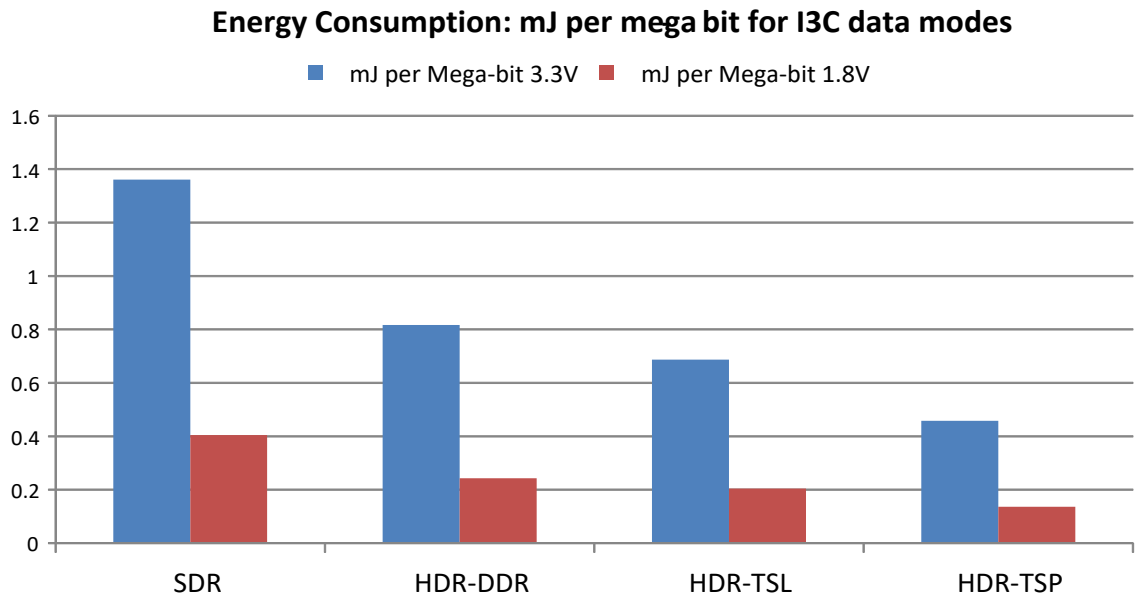


Figure 2 Energy Consumption and Raw Data Rate: I3C vs. I^2C

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

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



Assumptions: 1) All symbols in each mode have equal probability for use.
2) Energy consumption is the energy delivered by pull up devices to the bus (which includes drivers and resist

109

110 **Figure 3 Energy Consumption Comparison for I3C Data Modes**

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.

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

2.2 Definitions

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

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

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

Arbitrable: Subject to decision by Arbitration.

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

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

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

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

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

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

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

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

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

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

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

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

Device: A Master or Slave.

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

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

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

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

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

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

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

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

Hot-Join: Slaves 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 Slave to notify the Master that it is ready to get a Dynamic Address.

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

I^2C Device: A Master or Slave that meets the requirements of the I^2C Specification ***[NXP01]***.

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

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

I3C Slave: See Slave.

Legacy I^2C: I3C maintains the industry standard architecture of I^2C and supports existing I^2C Slave Devices. I3C does not support I^2C Bus Masters.

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

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

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

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

Message: A packetized communication between Devices.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 Master 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^2C ***[NXP01]***.

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

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

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

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

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

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

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

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

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

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

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

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

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

2.3 Abbreviations

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

2.4 Acronyms

BCR	Bus Characteristics Register
BER	Bit Error Rate
BMB	Bus Management Block
CCC	Common Command Code
CRC	Cyclic Redundancy Check
DAR	Dynamic Address Request
DCR	Device Characteristics Register
DDR	Double Data Rate
ESD	Electro Static Discharge
FSM	Finite State Machine
HDR	High Data Rate
HDR-DDR	HDR Double Data Rate Mode
IBI	In-Band Interrupt
ISTO	Industry Standards and Technology Organization
LCR	Legacy Characteristics Register
LSb	Least Significant Bit
Mbps	Megabits per second
MHz	Mega Hertz
MID	MIPI Manufacturer Identification ***[MIPI01]***
MSb	Most Significant Bit
NVMEM	Non-Volatile Memory

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

3 References

3.1 Normative References

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

3.2 Informative References

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

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

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

4 Technical Overview (Informative)

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

I3C is a two-wire bidirectional serial Bus, optimized for multiple sensor Slave Devices and controlled by only one I3C Master Device at a time. I3C is backward compatible with many Legacy I^{2}C Devices, but I3C Devices also support significantly higher speeds, new communication Modes, and new Device roles, including an ability to change Device Roles over time (i.e., the initial Master can cooperatively pass the Master role to another I3C Device on the Bus, if the second I3C Device supports that feature).

I3C Basic includes:

- Support for many Legacy I^{2}C Slave Devices and messages
- **I3C Single Data Rate (SDR) Mode:** New I3C enhanced version of the I^{2}C protocol supporting private messages, and adding two kinds of standard built-in messages:
  - **Broadcast messages**, which are sent to all I3C Slaves on the Bus
  - **Direct messages**, which are addressed to specific Slaves

4.1 I3C Fundamental Principles

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

The two wires are designated SDA and SCL:

- SDA (Serial Data) is a bidirectional data pin
- SCL (Serial Clock) is a clock pin

An I3C Bus supports the mixing of various Message types:

1. I^{2}C-like SDR Messages, with SCL clock speeds up to 12.5MHz
2. Broadcast and Direct Common Command Code (CCC) Messages that allow the Master to communicate to all or one of the Slaves on the I3C Bus, respectively
3. **NOT SUPPORTED IN I3C BASIC:** HDR Mode Messages, which achieve higher data rates per equivalent clock cycle
4. I^{2}C Messages to Legacy I^{2}C Slaves
5. Slave-initiated START Request to the Master, for example to send an In-Band Interrupt or to request the Master role

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

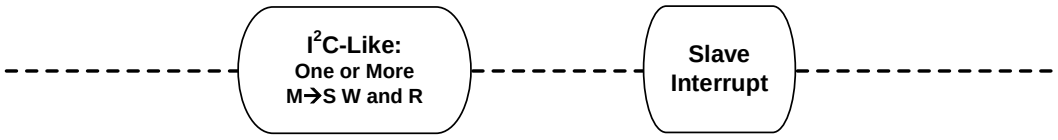


Figure 4 Example of Data Traffic on the I3C Bus

Figure 5 illustrates how I3C communication is initiated:

- All I3C communication occurs within a Frame. The Frame begins with a START, followed by one or more transfers, and a STOP.
- For the HDR Modes that are not supported in I3C Basic, but are tolerated by I3C Basic Devices:
  - First the dedicated Broadcast I3C Address (7'h7E) is issued to all Slaves on the I3C Bus.
  - Then one of the EnterHDR CCCs is issued, indicating that the Master is entering an HDR Mode. Each HDR Mode has its own EnterHDR CCC.
  - This is followed by one or more HDR transfers.
  - HDR Mode is ended by using the HDR Exit Pattern protocol.

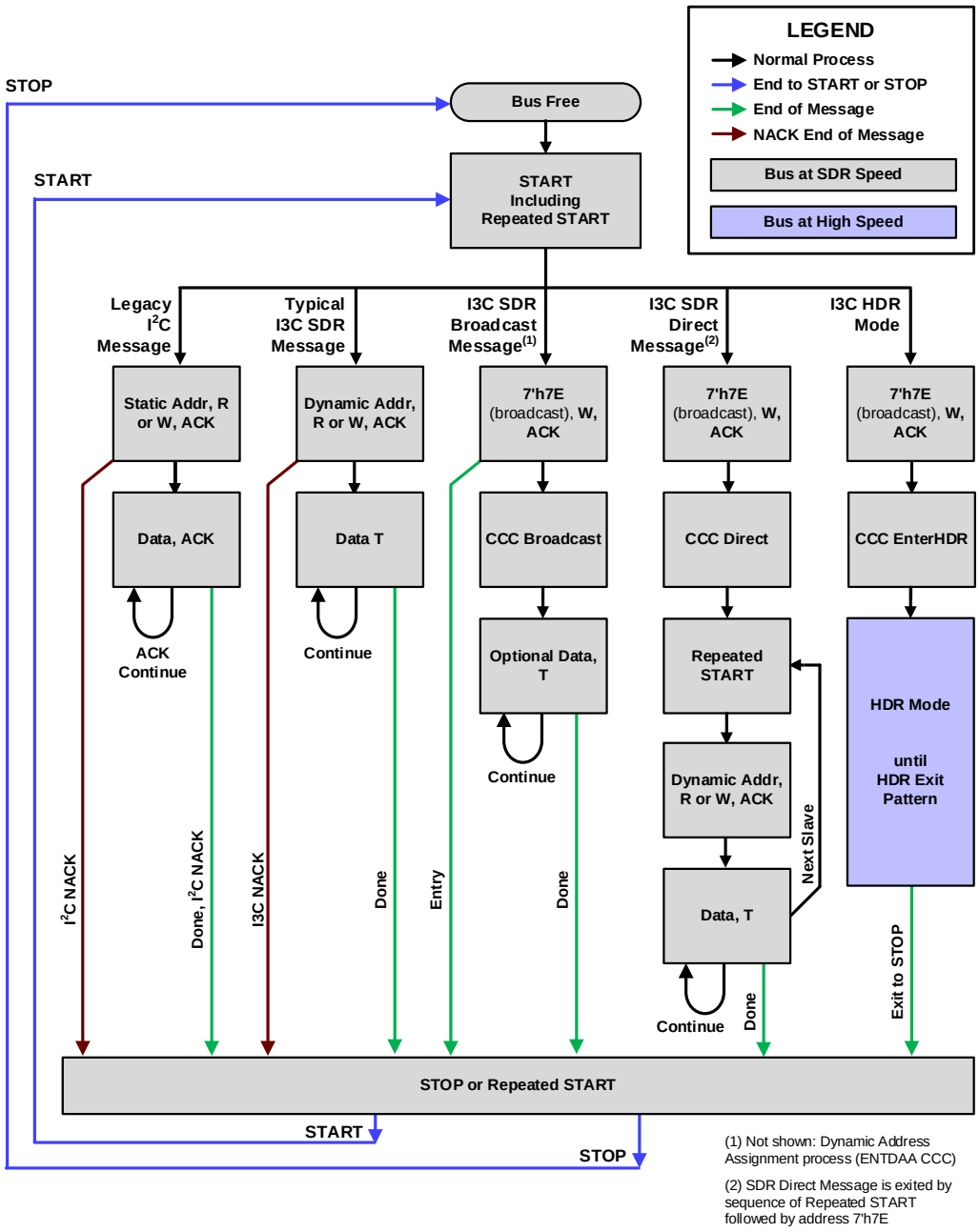


Figure 5 I3C Communication Flow

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

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

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

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

4.2 I3C Master and Slave Devices

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

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

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

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

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

Table 2 Roles for I3C Compatible Devices

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

Note:

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

4.2.1 I3C Master Device

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

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

- Generates the Bus clock
- Manages Pull-Up structures
- Manages the Dynamic Address Assignment procedure (including Hot-Join events) while acting as the Main Master
- Manages START Requests from I3C Slave Devices on the Bus along with Address Arbitration requests:
  - Generate In-Band Interrupts
  - For Hot-Join events
  - To become Current Master
- Supports I^2C Legacy Slave Devices
- Supports I3C SDR Mode

Figure 6 is a block diagram of a typical generic I3C Master Device.

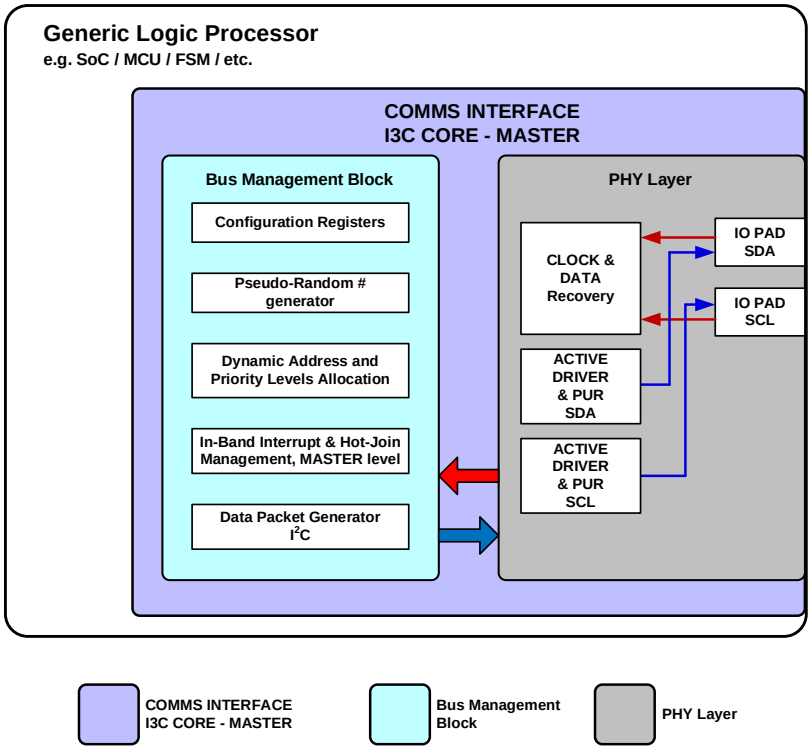


Figure 6 I3C Master Device Block Diagram

4.2.1.1 I3C Master Device Roles

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

Main Master Device Roles:

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

Secondary Master Device Roles:

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

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

Note:

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

4.2.2 I3C Slave Device

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

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

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

In addition, an I3C Slave Device can optionally:

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

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

Figure 7 is a block diagram of a typical generic I3C Slave Device.

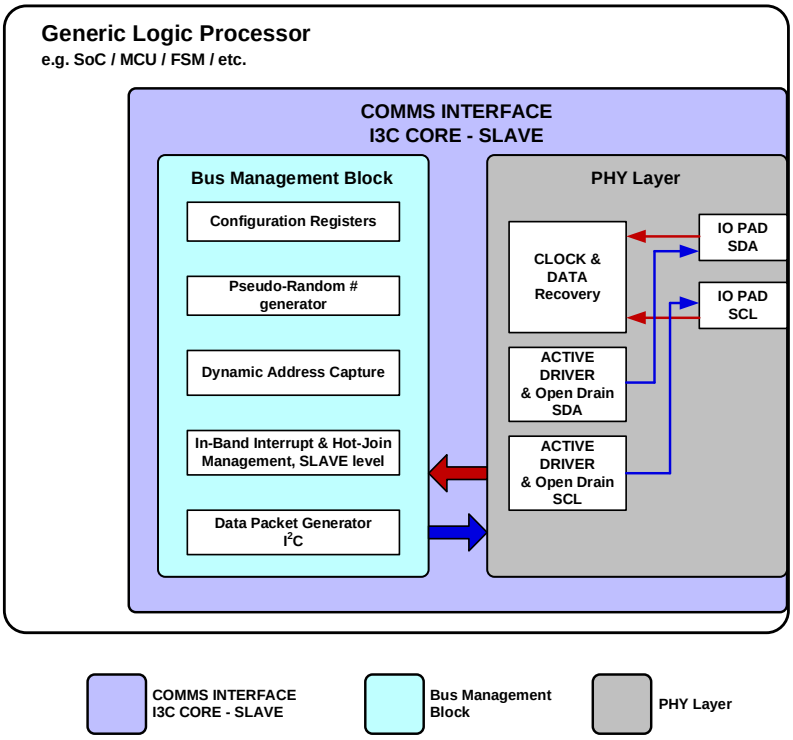


Figure 7 I3C Slave Device Block Diagram

4.2.2.1 I3C Slave Device Roles

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

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

Note:

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

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

This page intentionally left blank.

5 I3C Protocol

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

- **Single Data Rate (SDR) Mode:** See ***Section 5.1***
- **NOT SUPPORTED IN I3C BASIC: High Data Rate (HDR) Modes:** See ***Section 5.2***
  - **HDR Ternary Symbol Pure-bus (HDR-TSP) Mode**
  - **HDR Ternary Symbol Legacy-inclusive-bus (HDR-TSL) Mode**
  - **HDR Double Data Rate (HDR-DDR) Mode**

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

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

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

5.1 Single Data Rate (SDR) Mode

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

SDR Mode is the default Mode of the I3C Bus, and is primarily used for private messaging from the Current Master Device to Slave Devices. SDR Mode is also used to enter other Modes, sub-Modes, and states (as described in ***Section 5.1*** and ***Section 5.2***); and for built-in features such as Common Commands (CCCs), In-Band Interrupts, and transition from I^2C to I3C by assignment of a Dynamic Address.

I3C SDR Mode is significantly similar to the I^2C protocol ***[NXP01]*** in terms of procedures and conditions, and as a result I3C Devices and many Legacy I^2C Slave Devices (but not I^2C Master Devices) can coexist on the same I3C Bus. However SDR Mode also includes numerous new features not present in I^2C. For the procedures and conditions that I3C shares with I^2C, SDR Mode closely follows the definitions in the I^2C Specification. I^2C traffic from an I3C Master to an I^2C Slave will be properly ignored by all I3C Slaves, because the I3C protocol is designed to allow I^2C traffic. I3C traffic from an I3C Master to an I3C Slave will not be seen by most Legacy I^2C Slave Devices, because the I^2C Spike Filter is opaque to I3C's higher clock speed.

5.1.1 Bus Configuration

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

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

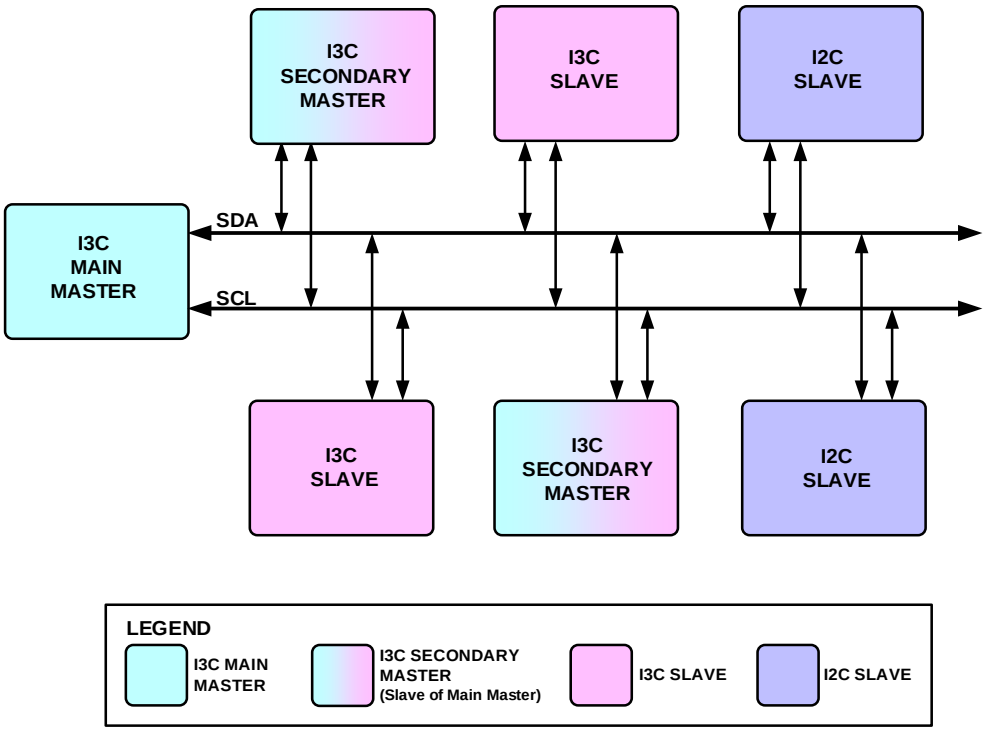


Figure 8 I3C Bus with I^{2}C Devices and I3C Devices

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

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

5.1.1.1 I3C Device Characteristics

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

Table 3 I3C Devices Roles vs Responsibilities

Responsibilities / Features	Comments	Roles					
		Main Master	**Secondary Master**	**SDR Only Main Master**	**SDR Only Secondary Master**	**Slave**	**SDR Only Slave**
Manages SDA Arbitration	For Address Arbitration, In-Band Interrupt, Hot-Join, Dynamic Address, as appropriate	Y	Y	Y	Y	N	N
Dynamic Address Assignment	Master assigns Dynamic Address	Y	N	Y	N	N	N
Hot-Join Dynamic Address Assignment	Master capable of assignment Dynamic Address after Hot-join	Y	Optional	Y	Optional	N	N
Self Dynamic Address Assignment	Only Main Master can self-assign a Dynamic Address	Y	N	Y	N	N	N
Static I²C Address[1]	–	N/A	Optional	N/A	Optional	Optional	Optional
Memory for Slaves' Addresses and Characteristics	Retaining registers	Y	Y	Y	Y	N	N
HDR Exit Pattern Generation capable[2]	Able to generate the HDR Exit Pattern on the Bus for error recovery	Y	Y	Y	Y	N	N
HDR Tolerant	Recognizes HDR Exit Pattern	Y	Y	Y	Y	Y	Y

Note:
*1) A Static Address may be used to more quickly assign a Dynamic Address. See **Section 5.1.4**.*
2) All Slaves require an HDR Exit Pattern Detector, even Slaves that are not HDR capable

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

Table 4 I²C Features Allowed in I3C Slaves

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

Note:
1) *See **Table 54** and **Table 57***
2) *I3C drive and timing requirements are different from I²C*
3) *If an I3C Slave has I²C features intended for use on an I²C Bus, then they will not be used on an I3C Bus. As stated in **Section 5.1.1.1**, once the Slave sees a 7'h7E, it will disable I²C features that are not used by I3C.*

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

Table 5 Legacy I²C-Only Slave Categories and Characteristics

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

Note:
1) *Allows tolerance of HDR Modes and SDR at SCL High periods of $t_{DIG\_H\_MIXED}$ or less*
2) *Allows compliance up to maximum SDR SCL clock frequency (f_{SCL})*

5.1.1.2 I3C Characteristics Registers

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

There are three Characteristics Register types:

- **Bus Characteristics Register** (BCR, see ***Section 5.1.1.2.1***)
- **Device Characteristics Register** (DCR, see ***Section 5.1.1.2.2***)
- **Legacy Virtual Register** (LVR, see ***Section 5.1.1.2.3***)

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

- Every I3C compliant Device (as defined in ***Table 3***) shall have one Bus Characteristics Register, and one Device Characteristics Register.
- Every Legacy I^2C Device to be connected to an I3C Bus shall have one associated Legacy Virtual Register. Since these are Legacy Devices, it is understood that this register will exist virtually, for example as part of the Device's driver.

5.1.1.2.1 Bus Characteristics Register (BCR)

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

Table 6 Bus Characteristics Register (BCR)

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

Note:

1) *For an I3C Device acting as I3C Main Master, the BCR Device Role bits will contain the value 2'b01.*
2) *Bridge Devices are required to comply with the MIPI Specification for I3C v 1.0* ***[MIPI02]****.*
3) *Offline Capable Devices retain the Dynamic Address, and are specified in* ***Section 2.2****.*
4) *Master shall use the GETMXDS CCC to interrogate the Slave for specific limitation.*

5.1.1.2.2 Device Characteristics Register (DCR)

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

Table 7 I3C Device Characteristics Register (DCR)

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

5.1.1.2.3 Legacy Virtual Register (LVR)

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

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

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

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

5.1.2 Bus Communication

The primary protocol and Mode of I3C is SDR (Single Data Rate) Mode. The SDR protocol is based on the I^2C standard protocol ***[NXP01]***, with a few notable variations:

- The I3C START and STOP (as shown in ***Figure 37*** and ***Figure 39***, respectively) are identical to the I^2C START and STOP in their signaling, but they may vary from I^2C in their timing. Compare ***Table 58*** vs. ***Table 57***.
- The I3C Address Header is identical to the I^2C Address Header in bit form and in signaling, but it may vary from I^2C in its timing. See ***Section 5.1.2.2***, ***Table 58***, and ***Table 59***.
- The Data 9-bit Words use the same bit count as I^2C, but differ in the ninth bit, as explained in ***Section 5.1.2.3***.
- The I3C Data is normally sent using Push-Pull signaling, whereas I^2C uses Open Drain signaling. There are exceptions, including use in DAA (see ***Section 5.1.4.2***) and an allowance for the Master and Slave to agree to let the Slave to return Read operations using Open Drain. The use of Push-Pull impacts transitions between Master and Slave. See ***Section 5.1.2.3*** and its sub-sections.
- In I3C, the SCL line is only driven by the Master. Normally this drive is Push-Pull, but it can also be Open Drain.

Because the bit count is the same for Address Header and Data Word, the I3C Slave only needs to know whether a Message is an I3C Message (vs. is an I^2C Message) if the Message is addressed to that Slave (either directly or by Broadcast).

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

An I3C Message is an SDR Message if:

- The Address in the Address Header is 7’h7E (the I3C Broadcast Address). All I3C Slaves shall match Address value 7’h7E. No I^2C Slave will match Address 7’h7E, because that value is reserved and unused in I^2C.
- The Address in the Address Header matches the Slave’s Dynamic Address (as assigned by the I3C Master per ***Section 5.1.4***). All I3C Slaves shall match their own Dynamic Address. (It is permitted to then NACK the Header if needed.)

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

Note:

*Legacy I^2C Slaves will ignore any Message not addressed to them, and will await the next START or STOP. Per **Section 5.1.2.4**, Legacy I^2C Slaves may also not see some or all I3C Messages and Modes due to the speed of SCL signaling.*

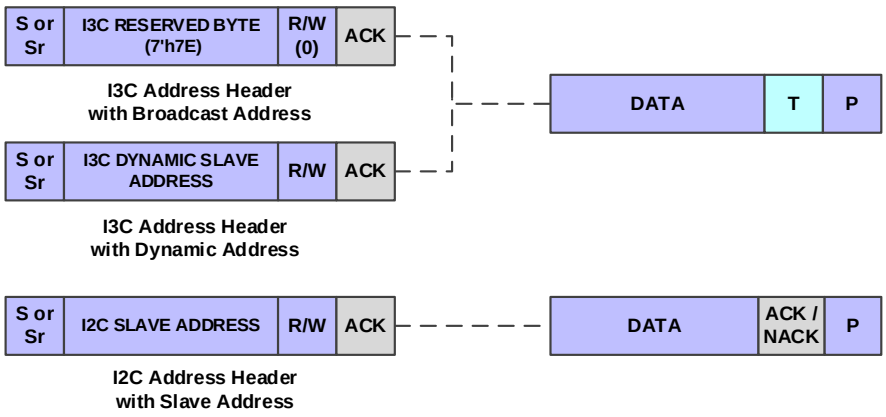


Figure 9 Address Header Comparison

5.1.2.1 Role of I3C Slave

The I3C Slave does not have to know whether it is on a Legacy I^{2}C Bus or an I3C Bus. If it has a Legacy I^{2}C Static Address, then it may participate using that Address up until (and if) it is assigned a Dynamic Address. Once assigned a Dynamic Address, unless asked to Reset, it shall only operate as an I3C Slave.

An I3C Capable Slave may act as an I^{2}C Device before it gets its Dynamic Address (DA) assigned. However, the Slave shall ACK the START with address 7'h7E. (The only exception would be if the Slave is choosing to remain an I^{2}C-only Device on a given bus or use, in which case it would leave its 50ns Spike Filter enabled.)

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

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

  Examples of appropriate processing:
  - RSTDAAA has no effect, since no Dynamic Address is assigned
  - If the Slave would never issue a Master Request, then DISEC for Master Request can be ignored.
- The Slave shall recognize the CCCs ENTHDR0 through ENTHDR7 (see ***Section 5.1.9.3.9***), and then wait for the HDR Exit Pattern.
- The Slave may choose to understand and process the SETDASA CCC (see ***Section 5.1.9.3.10***) when its Static Address matches.
- The Slave may choose to understand and process the SETAASA CCC (see ***Section 5.1.9.3.22***).
- The Slave shall disregard all Directed CCC commands, but shall properly recognize the ends of Directed CCCs (either repeated START followed by 7'h7E, or STOP).
- When no Dynamic Address has been assigned yet, the Slave may either support or ignore non-Required and Conditionally Required Broadcast CCCs.

  This is true even if the Slave supports any of these CCCs after being assigned a Dynamic Address. For example, the Slave may choose to only support Test Mode only when no Dynamic Address is assigned.
- The I3C Slave shall ignore S0 type errors related to incorrect addresses only (see ***Section 5.1.10.1.1***).

The role of an I3C Slave shall be as follows:

1. Following a START or a Repeated START, at any speed conforming to ***Section 6*** of this Specification, the I3C Slave shall attempt to match the Address to the I3C Broadcast Address (7'h7E) or to its own Dynamic Address, once assigned. If a match is found, then the I3C Slave shall treat that Message as I3C SDR.
2. If the Message is addressed to the Slave's Dynamic Address, then the Slave may ACK or passively NACK the Address Header:
   a. If the Slave ACKs the Address Header, then the Slave shall process the Message as I3C SDR, following all rules as outlined in this Section.
   b. If the Slave NACKs the Address Header (does not drive the ACK bit Low), then the Slave may disregard any bits that follow, up until the next Repeated START or STOP.
3. If the Message is addressed to the I3C Broadcast Address (7'h7E), with a Write (RnW bit is 0), then the Slave shall process that Message at least through the first byte of data (if any data is present in the Message):
   a. If there is a byte of data in a 7'h7E Broadcast Message, then the Message is a CCC (Common Command Code) Command, per ***Section 5.1.9***.
   b. The I3C Slave shall process all applicable Required CCC commands, per ***Section 5.1.9.3***. A command may be either Always Required, or only Contextually Required, per ***Table 9***.
   c. If the CCC command changes the Mode of the I3C Bus, then the I3C Slave shall handle the new Mode in one of two ways.

      Either:

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

      Or:

      ii. If the new Mode is HDR (High Data Rate) Mode, then the Slave may either enter into HDR Mode if it supports that specific HDR Mode, or else enable its HDR Exit Pattern detector (per ***Section 5.2.1*** and ***Section 5.2.1.3***) to await the exit from HDR Mode.
4. If the Message is not addressed to the I3C Broadcast Address (7'h7E) or to the Slave's Dynamic Address, then the I3C Slave shall await either a Repeated START or a STOP. The Slave may record/monitor the bits as they pass (if desired), but the only obligation is to wait for either a Repeated START or a STOP:
   - A Repeated START is defined by the SDA line changing from High to Low while the SCL line is High, per ***Section 6***.
   - A STOP is indicated by the SDA line changing from Low to High while the SCL line is High, per ***Section 6***.

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

5.1.2.2 I3C Address Header

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

The Address Header following a START is an Arbitrable Address Header, as explained in ***Section 5.1.2.2.1***. This means the START and at least the first Address bit and ACK/NACK are issued on SDA using Open Drain Bus drive, similar to I^2C. However, some of the Arbitrable Address Header may be driven on SDA using Push-Pull and higher speed (see ***Section 5.1.2.2.2***).

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

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

1. An In-Band Interrupt, per ***Section 5.1.6***. This is equivalent to toggling a wire to get the Master's attention. The In-Band Interrupt request shall be made using the Slave's Dynamic Address with a RnW bit of 1.
2. A Secondary Master request, per ***Section 5.1.7***. An I3C Slave shall not make such a request unless it is marked as a Secondary Master in its BCR register, per ***Section 5.1.1.2.1***. The Secondary Master request shall be made using the Slave's Dynamic Address with a RnW bit of 0.
3. A Hot-Join request, per ***Section 5.1.5***. An I3C Slave shall only make such a request when becoming available after the I3C Bus is operational. The Hot-Join request shall be made using the special Hot-Join Address of 7'h02.

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

1. A START (but not a Repeated START) is issued on the Bus following a Bus Available Condition, per ***Section 5.1.3***. The Slave may transmit its Dynamic Address or the Hot-Join Address (7'h02) following the START, by adhering to the I3C Address Arbitration rules per ***Section 5.1.2.2.1***.
2. The Bus is in a Bus Available Condition, per ***Section 5.1.3***, so the Slave may issue a START by pulling the SDA Low.
    a. If the Slave pulls the SDA Low, then the Master shall pull SCL Low within a best-efforts period of time, where that time is not explicitly defined.
    b. The Master shall also pull SDA Low (overlapping the Slave pulling it Low).
    c. Once the Master has pulled the SCL Low, the Slave shall control the SDA line in Open Drain mode (i.e., either pull Low, or release High).
    d. The Slave may then issue its Address in the normal way (condition 1 above).

5.1.2.2.1 I3C Address Arbitration

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

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

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

   Note:

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

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

5.1.2.2.2 I3C Address Arbitration Optimization

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

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

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

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

- If the I3C Master is transmitting a 1 value (i.e. High-Z on SDA), as it would when transmitting 7’h7E, then it can monitor SDA on the rising edge of SCL. If SDA has the value 1 (i.e. if SDA is not being driven by any Slave), then the I3C Master may optionally transmit the remainder of the Address Header (up until the ACK/NACK) in Push-Pull mode. See ***Figure 10***, upper waveform.
- If the I3C Master is transmitting a 0 value (i.e. is driving SDA Low), or if SDA was driven Low by a Slave, then the I3C Master shall transmit the remainder of the Address Header using Open Drain.
- If the I3C Master intends to transmit solely to other I3C Devices (i.e. not to any I^2C Devices), then it may optionally maintain the SCL pulse width below 50ns, with the result that any I^2C Devices present on the Bus will see only a 0 value. This shorter pulse width produces a higher data rate, because for Open Drain Arbitration only the Low time is extended. See ***Figure 10***, middle waveform.
- If the I3C Master intends to transmit to any I^2C Devices, then it must use the slower I^2C timing. See ***Figure 10***, lower waveform.

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

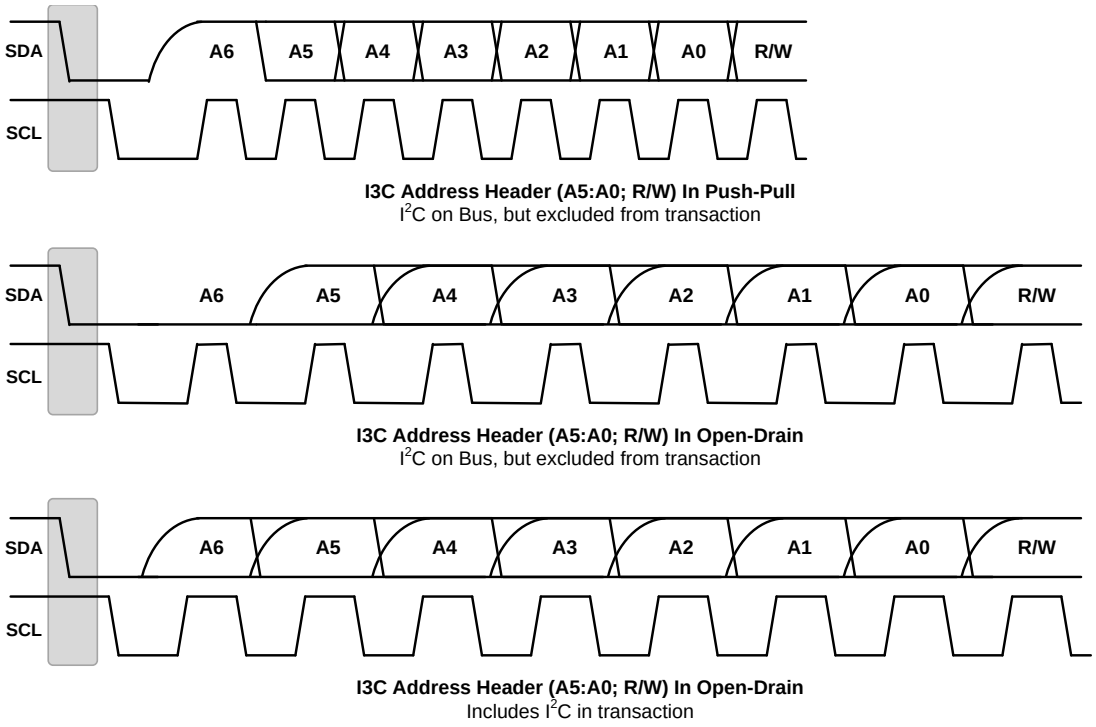


Figure 10 Address Arbitration During Header

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

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

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

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

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

1. The Addresses match, but the difference is caught on the RnW bit, and the Master was Writing (RnW=0); so the Master wins (IBI with RnW=1 loses), and proceeds normally.
2. The Addresses match, but the difference is caught on the RnW bit, and the Master was Reading (RnW=1); so the Master loses, and must ACK or NACK the Mastership request (RnW=0).
3. The Addresses match as do the RnW and so neither Master nor Slave shall ACK since both are expecting the other side to.
   a. This is a problem since the Master cannot tell whether the NACK was due to this condition, or simply because the Slave may have chosen to NACK it.
   b. The Master shall transmit the Slave Address again after a Repeated START (the next one or any one prior to a STOP in the Frame). This allows it to determine which condition occurred and to avoid a deadlock.

5.1.2.2.4 Address Header Following a Repeated START is Push-Pull

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

As a result, the Address Header (i.e., the 7 bits of Address plus the RnW bit) shall be transmitted on SDA using Push-Pull mode when the Message is not to a Legacy I^2C Slave. The ACK/NACK bit that follows the RnW bit is always Open Drain to allow the Slave to ACK or passively NACK its Address.

5.1.2.2.5 I3C Slave Address Restrictions

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

- The I3C Master shall not use any of 7'h00, 7'h01, 7'h02, 7'h7E, 7'h7F. All are reserved for I3C.
- The I3C Master shall not use any of 7'h3E, 7'h5E, 7'h6E, 7'h76, 7'h7A, 7'h7C, 7'h7F. All are prohibited for detecting an error in the Broadcast Address (7'h7E).
- The I3C Master may choose to not use 7'h03, marked in I^2C as reserved.
- The I3C Master shall not use 7'h04, 7'h05, 7'h06, 7'h07 if any Legacy I^2C Devices are present on the Bus that support I^2C "High-Speed Mode".
- The I3C Master shall not use 7'h7C or 7'h7D if any Legacy I^2C Devices are present on the Bus that support I^2C Device ID Mode.
- The I3C Master shall not use 7'h78, 7'h79, 7'h7A, 7'h7B if any Legacy I^2C Devices are present on the Bus that support I^2C Extended Address Mode and have an Extended Address or would be impacted by the Extended Address mechanism.

727 **Table 9 I3C Slave Address Restrictions**

Slave Dynamic Address		Restriction	Description
Binary	**Hex**		
000 0000	**7'h00**	**Shall not use**	I3C Reserved
000 0001	**7'h01**	**Shall not use**	I3C Reserved: For use with SETDASA CCC in special Point-to-Point Communication. See ***Section 5.1.9.3.10***.
000 0010	**7'h02**	**Shall not use**	I3C Reserved: Hot-Join Address
000 0011	**7'h03**	**Optional**	Marked 'Reserved' by I^2C
000 0100	**7'h04**	**Conditional**	Available for use only if no Legacy I^2C Devices supporting I^2C "High-Speed Mode" are present on the Bus
000 0101	**7'h05**		
000 0011	**7'h06**		
000 0011	**7'h07**		
000 1000 011 1101	**7'h08 – 7'h3D**	**Available for use**	54 Addresses
011 1110	**7'h3E**	**Shall not use**	I3C Reserved: Broadcast Address single bit error detect
011 1111 101 1101	**7'h3F – 7'h5D**	**Available for use**	31 Addresses
101 1110	**7'h5E**	**Shall not use**	I3C Reserved: Broadcast Address single bit error detect
101 1111 110 1101	**7'h5F – 7'h6D**	**Available for use**	15 Addresses
110 1110	**7'h6E**	**Shall not use**	I3C Reserved: Broadcast Address single bit error detect
110 1111 111 0101	**7'h6F – 7'h75**	**Available for use**	7 Addresses
111 0110	**7'h76**	**Shall not use**	I3C Reserved: Broadcast Address single bit error detect
111 0111	**7'h77**	**Available for use**	1 Address
111 1000	**7'h78**	**Conditional**	Available for use only if no Legacy I^2C Devices are present on the Bus that both a) support I^2C "Extended Address Mode", and b) either have an Extended Address, or would be impacted by the Extended Address mechanism
111 1001	**7'h79**		
111 1010	**7'h7A**	**Shall not use**	I3C Reserved: Broadcast Address single bit error detect
111 1011	**7'h7B**	**Conditional**	Available for use only if no Legacy I^2C Devices are present on the Bus that both a) support I^2C "Extended Address Mode", and b) either have an Extended Address, or would be impacted by the Extended Address mechanism
111 1100	**7'h7C**	**Shall not use**	I3C Reserved: Broadcast Address single bit error detect (Also not available for use if any Legacy I^2C Devices supporting I^2C "Device ID Mode" are present on the Bus.)
111 1101	**7'h7D**	**Conditional**	Available for use only if no Legacy I^2C Devices supporting I^2C "Device ID Mode" are on the Bus
111 1110	**7'h7E**	**Shall not use**	I3C Reserved: Broadcast Address
111 1111	**7'h7F**	**Shall not use**	I3C Reserved: Broadcast Address single bit error detect

5.1.2.3 I3C SDR Data Words

In I3C SDR, the Data Words match I^2C only in the sense that they are both 9 bits long. I3C SDR Data Words differ from I^2C in three ways, as detailed in ***Sub-Sections 5.1.2.3.1***, ***5.1.2.3.2***, and ***5.1.2.3.4***.

In summary:

1. **Handoff from Address ACK to SDR Master Write Data:** When performing an SDR Write, the handoff from the Slave's Address Header ACK to the Master's first Data bit is different in I3C. I^2C is Open Drain, so overlap from the ACK Low into the first bit is harmless. By contrast, I3C is Push-Pull and so this handoff is specified (see ***Section 5.1.2.3.1***).
2. **Ninth Bit of SDR Master Written Data as Parity:** In I^2C, the ninth Data bit written by the Master is an ACK by the Slave. By contrast, in I3C the ninth Data bit written by the Master is the Parity of the preceding eight Data bits. Therefore, in I3C the Slave shall not drive the SDA line for Data written by the Master in SDR. In SDR terms, the ninth bit of Write data is referred to as the T-Bit (for 'Transition') (see ***Section 5.1.2.3.2***).
3. **Ninth Bit of SDR Slave Returned (Read) Data as End-of-Data:** In I^2C, the ninth Data bit from Slave to Master is an ACK by the Master. By contrast, in I3C this bit allows the Slave to end a Read, and allows the Master to Abort a Read. In SDR terms, the ninth bit of Read data is referred to as the T-Bit (for 'Transition') (see ***Section 5.1.2.3.4***).

5.1.2.3.1 Transition from Address ACK to SDR Master Write Data

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

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

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

1. The I3C Slave shall hold the SDA line Low during the ACK (while SCL is Low).
   - This is an Open Drain SCL Low period.
2. After the I3C Slave sees the rising edge of SCL, it releases the SDA line to High-Z.
   - The I3C Slave shall release the SDA line using normal (Push-Pull) timing (release the SDA line as soon as it sees SCL rising).
3. After the rising edge of SCL, the I3C Master shall drive the SDA line Low.
   - As a result, both Master and Slave will be driving the SDA line Low for a short overlap (which is safe).
   - The SCL High period may be as short as the minimal t_{DIG_H}, per ***Section 6.2***.
4. On the falling edge of SCL the I3C Master shall begin driving data on the SDA line, using Push-Pull drive as shown in ***Figure 32***.

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

1. Continue the transaction, by generating a Repeated START

or

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

5.1.2.3.2 Transition from Address ACK to Mandatory Byte during IBI

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

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

- **ACK:** When the IBI Slave Address Header results in an ACK, and the Slave Device is capable of sending a Mandatory Byte, then the SDA line has to be turned from Open Drain to Push-Pull for the first data bit. To do that safely, I3C SDR specifies how the handoff is to occur. This is summarized below.
  1. The I3C Master shall hold the SDA line Low during the ACK (i.e., while the SCL line is Low). This is an Open Drain SCL Low period.
  2. After the rising edge of the SCL line, the I3C Slave shall drive the SDA line Low as soon as possible. As a result, both Master and Slave will be driving the SDA line Low for a short overlap (which is safe).
  3. After (A) a minimum of the t_{SCO} time reported by the Slave Device, or (B) a predetermined safe time that could guarantee the takeover by all Slaves, the I3C Master shall release the SDA line. The SCL High period may be as short as the minimal t_{DIG_H}, per ***Section 6.2***.
  4. On the falling edge of SCL, the I3C Slave shall begin driving data on the SDA line, using Push-Pull drive.
- **NACK:** When the Address Header results in a NACK, the Master may choose to either:
  - Continue the transaction, by generating a Repeated START, or
  - Relinquish the Bus by generating a STOP.

5.1.2.3.3 Ninth Bit of SDR Master Written Data as Parity

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

Examples:

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

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

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

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

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

- The I3C Slave returns the ninth bit as 0 (SDA Low) to end the Message:
  - The Slave shall set SDA Low on the falling edge of SCL.
  - On the following rising edge of SCL, the Slave shall set SDA to High-Z.
  - The I3C Master shall drive SDA Low on the rising edge of SCL, thereby overlapping with the Slave.
  - The I3C Master then shall issue either a STOP as shown in ***Figure 39***, or a Repeated START as shown in ***Figure 40*** (on the next clock, or one after, per the normal I^2C procedure for setting up SDA to issue a Repeated START).
- The I3C Slave returns the ninth bit as 1 (SDA High) to continue the Message (and permit the Master to abort the Message):
  - The Slave shall set SDA High on the falling edge of SCL.
  - On the following rising edge of SCL, the Slave shall set SDA to High-Z, thereby Parking the Bus for the SCL High period:
    - If the I3C Master is able to continue the reply from the Slave, then it shall do nothing. The weak Pull-Up resistor on SDA will keep SDA High during the SCL High period, as shown in ***Figure 41***.
    - If the I3C Master wants to abort the Message, then it shall drive SDA Low after the rising edge of SCL, thereby terminating the Message with a Repeated START. The I3C Master then takes control starting with the falling edge of SCL. The Master shall ensure enough delay after SCL rising before driving SDA Low to ensure no contention. In order to achieve this delay, the Master might have to extend the SCL High period.
      Since the transition of SDA Low when SCL is High is a Repeated START, the Master may begin a new Address (see ***Figure 43***), or it may issue a STOP in the next cycle (see ***Figure 42***). However in a Mixed Bus the Master should extend the SCL Low period, in order to ensure that any Spike Filters for Legacy I^2C Devices properly reset.
  - The Slave shall monitor the SDA on the falling edge of SCL:
    - If SDA is High, then the Slave shall continue with the next data value.
    - If SDA is Low (i.e., if there has been a Repeated START), then the Message has been aborted, and the Slave shall not drive SDA after that.

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

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

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

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

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

Bus Operating Parameter	Available Options Per Bus Configuration		
	Pure Bus	**Mixed Fast Bus**	**Mixed Slow and Limited Bus**
SDR Mode Speed	f_{SCL} (Min) to f_{SCL} (Max)	I²C messages: Fm or Fm+, I3C messages: SCL High from $t_{DIG\_H\_MIXED}$ (Min) to $t_{DIG\_H\_MIXED}$ (Max) with SCL low up to t_{DIG_L} (Max) (see ***Section 5.1.2.4.1***)	Fm or Fm+ only[1]
Note: *1) May be faster if all Legacy I²C Devices are index 1, per **Table 5***			

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

A pure I3C Bus may simply change the clock speed to any frequency in the allowed speed range, as outlined in ***Section 6***. By contrast, a Mixed Fast Bus wanting to clock faster than the slowest Legacy I^2C Device needs to take advantage of the Spike Filter, i.e, shall ensure that the SCL High period is shorter than the Spike Filter, in order to prevent the Legacy I^2C Devices from seeing the SCL High period (see $t_{DIG\_H\_MIXED}$ in ***Table 59***). As a result, the Legacy I^2C Device 'sees' the SCL as staying Low the whole period.

However, a Mixed Fast Bus I3C Master can change the effective Bus frequency by varying the duty-cycle of SCL. This allows running the data rate slower, for example to accommodate Slaves which need a lower rate as defined by GETMXDS CCC (see ***Section 5.1.9.3.18***). It may also be necessary to accommodate Legacy I^2C Devices with Spike Filters that need a longer Low period in order to function properly.

In this model the SCL High period shall never exceed $t_{DIG\_H\_MIXED}$, thus staying below the 50ns required by the I^2C Spike Filter; however, the Low period is free to be any length permitted by I3C's allowed clock frequency range. For example, depending on the clock generation capability of the I3C Master, the SCL Low period may be a multiple of the High period, or it may be any multiple of some higher frequency clock capable of providing a High period less than or equal to $t_{DIG\_H\_MIXED}$, but greater than the minimum SCL High period as defined in ***Table 57***.

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

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

Such adjustment of the clock Duty Cycle brings three benefits:

- It remains hidden from Legacy I2C Devices, while still significantly increasing the SDR data rate.
- It ensures a longer Low period, so that Legacy I^2C Device Spike Filters continue to track SCL as Low.
- It is easy for a Master to generate, and has no impact on I3C Slaves (which just react to clock edges).

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

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

5.1.2.5 Master Clock Stalling

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

Stalling may be necessary for either of two reasons:

1. The absolute or relative timing of a Message to a specific Slave, or to all Slaves, needs to be carefully controlled. Clock Stalling provides the Master with fine-grained data timing control.
2. The I3C Master needs to internally synchronize data. This may be due to parts of the Master's system waking up in response to data, to changing state, or to otherwise needing time during a transaction.

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

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

1. I3C/I^{2}C Transfer, ACK/NACK Phase
2. Write Data Transfer, Parity Bit
3. I3C Read Transfer, Transition Bit
4. Dynamic Address Assignment, First Bit of Assigned Address

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

Table 11 Master Clock Stall Times

Condition	Section	Maximum Stall Time
I3C/I^{2}C Transfer, ACK/NACK Phase	***5.1.2.5.1***	100 µs
Write Data Transfer, Parity Bit	***5.1.2.5.2***	100 µs
I3C Read Transfer, Transition Bit	***5.1.2.5.3***	100 µs
Dynamic Address Assignment, First Bit of Assigned Address	***5.1.2.5.4***	15 ms

In all cases, after Stalling the SCL Low period, the continuation of the clock (i.e., the first SCL rising edge after the extended Low period) shall follow the normal rules for I3C SDR (for I3C SDR Messages), or for I^{2}C (for I^{2}C Messages) including rise time, change in SDA (if any) before SCL rise, next bit, etc. For example, the I3C Master might choose to terminate the Frame after this Stalled bit with a STOP; but in this case the Master is required to observe the requirements for STOP timing, as appropriate.

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

5.1.2.5.1 I3C/I²C Transfer, ACK/NACK Phase

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

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

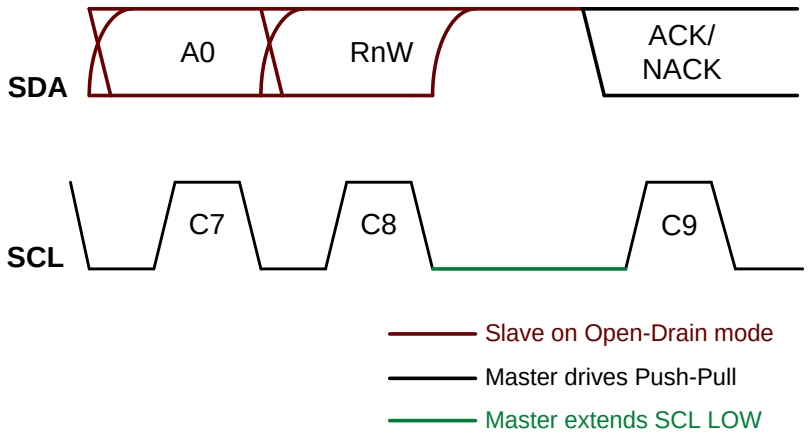


Figure 11 Master Clock Stalling in ACK Phase

5.1.2.5.2 Write Data Transfer, Parity Bit

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

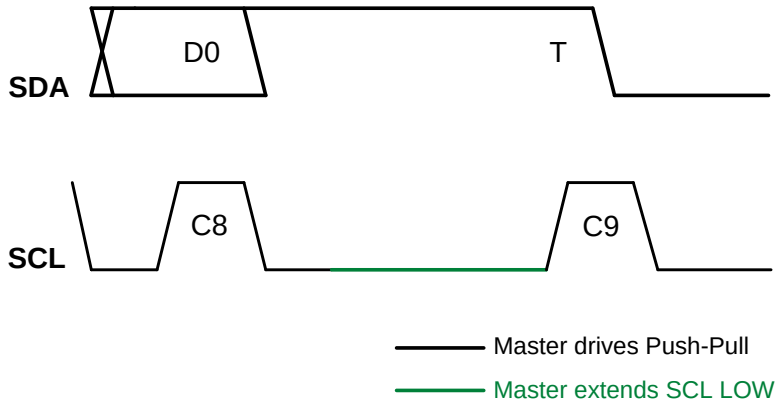


Figure 12 Master Clock Stalling in Write Parity Bit

5.1.2.5.3 I3C Read Transfer, Transition Bit

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

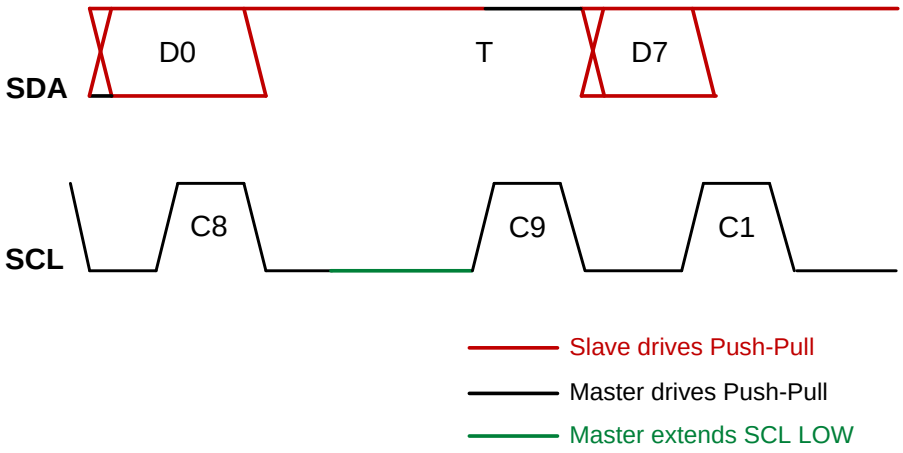


Figure 13 Master Clock Stalling in T-Bit Before Next Read Data

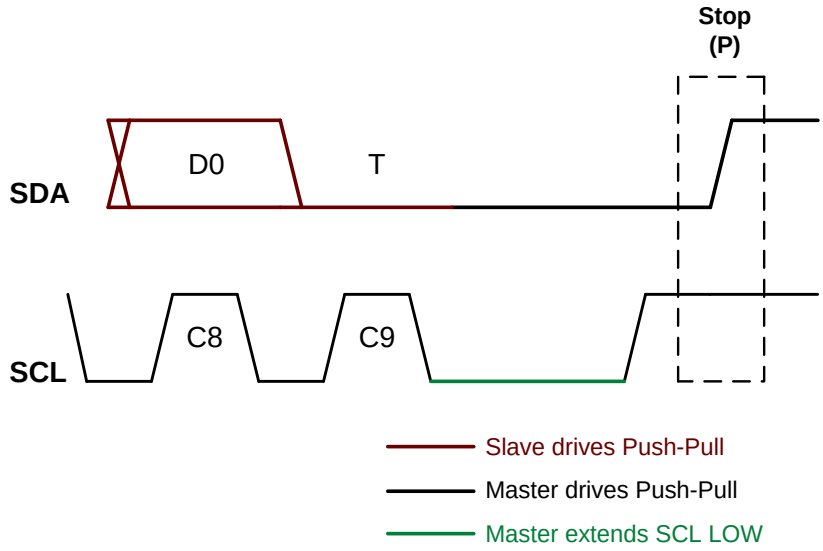


Figure 14 Master Clock Stalling in T-Bit Before STOP

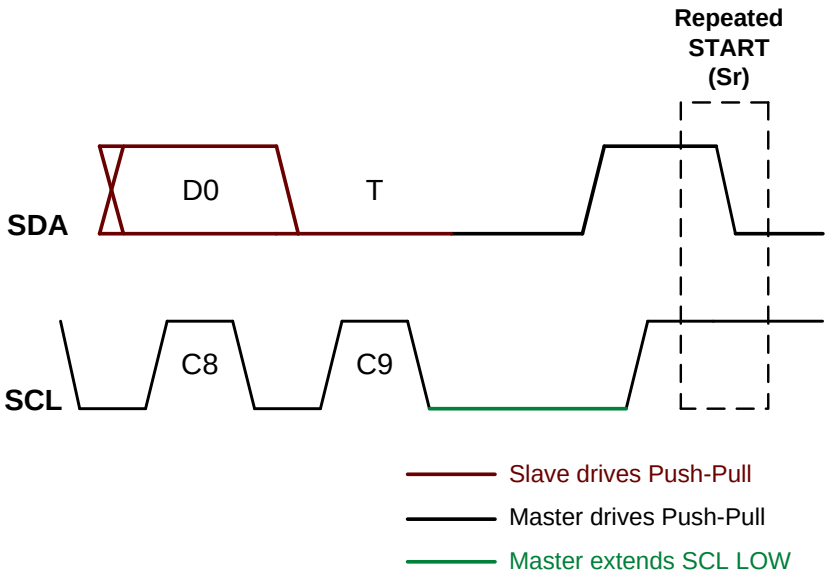


Figure 15 Master Clock Stalling in Low T-Bit Before Repeated Start

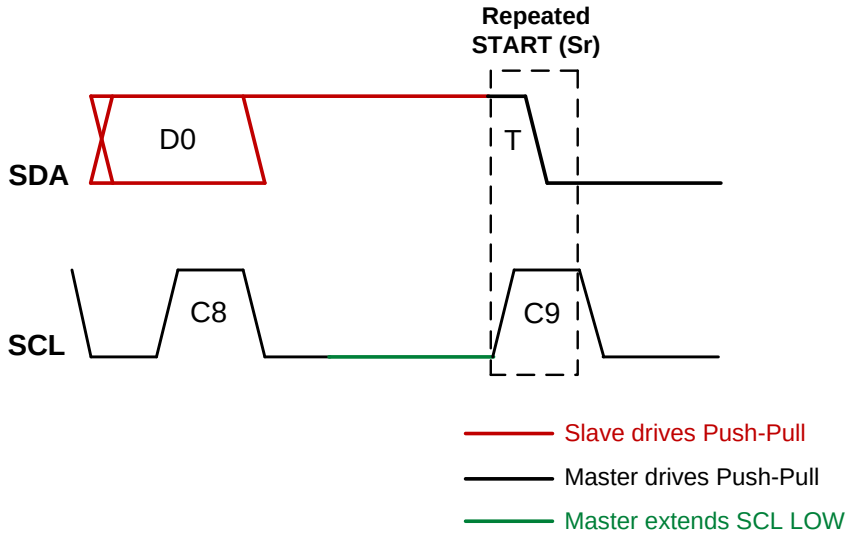


Figure 16 Master Clock Stalling in High T-Bit Before Repeated START

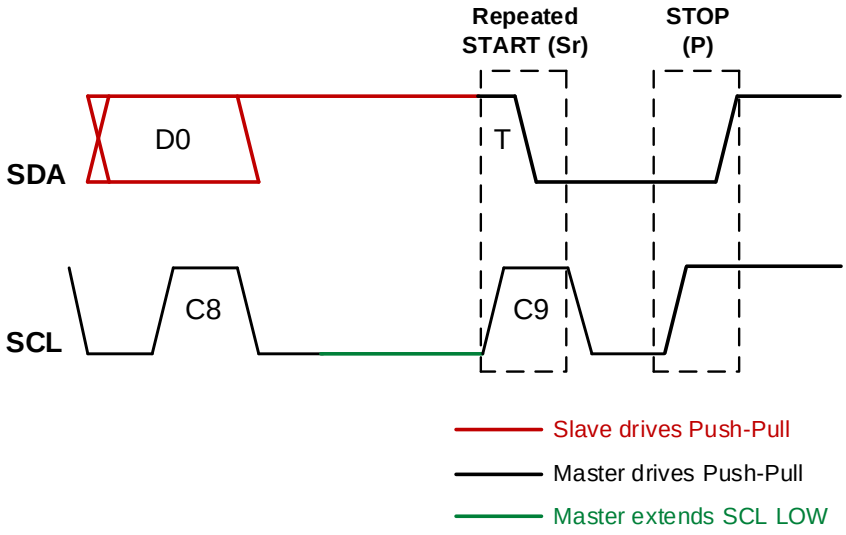


Figure 17 Master Clock Stalling in High T-Bit Before Repeated START and STOP

5.1.2.5.4 Dynamic Address Assignment, First Bit of Assigned Address

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

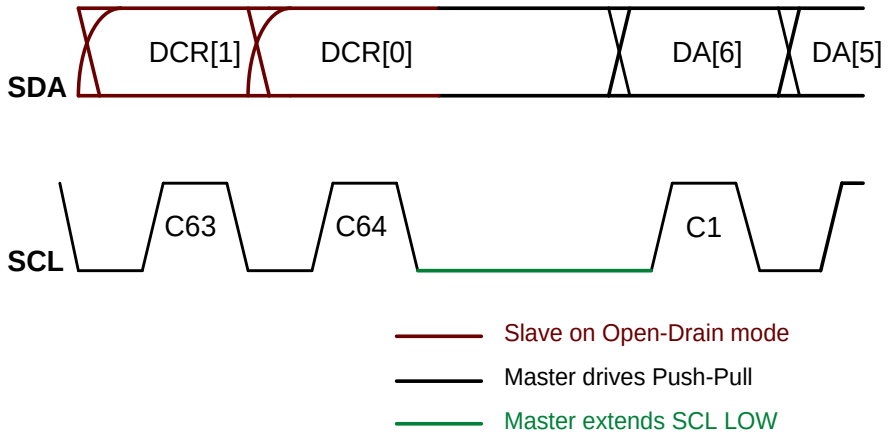


Figure 18 Master Clock Stalling in Dynamic Address First Bit

5.1.3 Bus Conditions

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

5.1.3.1 Open Drain Pull-Up and High-Keeper

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

This active Pull-Up shall be implemented either:

1. As a passive resistance from V_{DD}, or
2. As a passive resistance from a current source, or
3. In any other method that both:
   a. Balances its current sourcing in order to ensure that SDA rises within t_{rDA} (see ***Table 57***), and
   b. Is not so strong as to prevent a Slave with the minimum I_{OL} driver (see ***Table 54***) from driving SDA Low within t_{rDA} (see ***Table 58***).

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

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

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

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

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

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

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

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

Note:

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

5.1.3.2 Bus Free Condition

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

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

5.1.3.3 Bus Available Condition

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

5.1.3.4 Bus Idle Condition

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

Note:

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

5.1.3.5 Activity States

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

Table 12 Activity States

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

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

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

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

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

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

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

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

5.1.4 Bus Initialization and Dynamic Address Assignment Mode

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

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

1. Upon any initialization of the I3C Bus, and
2. When the Device is connected to an already configured I3C Bus.

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

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

5.1.4.1 Device Requirements for Dynamic Address Assignment

5.1.4.1.1 Unique Identifiability

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

Either:

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

   For example, an Address similar to what I^2C specifies ***[NXP01]***.

Or:

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

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

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

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

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

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

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

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

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

Note:

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

5.1.4.2 Bus Initialization Sequence with Dynamic Address Assignment

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

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

1. The Main Master shall begin in an appropriately configured state, and shall have, either in its own non-volatile memory or as a result of having received it from the Application Host, the following data:
   a. The number of I3C compliant Devices that need to receive a Dynamic Address,
   b. The data for any I3C Devices resident on the I3C Bus that already have Static Addresses, and
   c. The data for any Legacy I^2C Devices resident on the I3C Bus.
2. The Main Master shall assign Dynamic Addresses to any I3C Devices with a known Static Address, using the Command Code **Set Dynamic Address from Static Address** (**SETDASA**) (see ***Section 5.1.9.3.10***), or by assigning all I3C Devices their known I^2C Static Address using the Command Code **Set All Addresses to Static Address** (**SETAASA**) (see ***Section 5.1.9.3.22***)

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

   In a system that mixes I^2C-capable Devices and non-I^2C-capable Devices, the Master shall send the SETAASA CCC before sending the ENTDAA CCC (see ***Section 5.1.9.3.4***), in order to assign Dynamic Addresses to the I3C-only Slaves. In addition, any Slave having a restricted Address (see ***Table 9***) shall not support SETAASA.
3. The Main Master shall send the Broadcast Command Code **Enter Dynamic Address Assignment (ENTDAA)** (see ***Section 5.1.9.3.4***). There is no data associated with this Command Code.
4. The Main Master shall send a Repeated START, and the I3C Broadcast Address 7'h7E with RnW bit High (i.e. Read). Every I3C Device on the I3C Bus that does not yet have an assigned Dynamic Address, and that is not a Hot-Join Device, shall acknowledge the I3C Broadcast Address. (I.e., Hot-Join Devices that do not yet have an assigned Dynamic Address shall not acknowledge the I3C Broadcast Address in this step.)

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

   Note:

   *This use of the I3C Broadcast Address (7'h7E) with RnW bit High (i.e. Read) is specific to Dynamic Address Assignment Mode. It is also acceptable per the I^2C Specification **[NXP01]**.*
5. The Main Master shall drive only the SCL line. The Main Master shall release the SDA line to a High-Z state, allowing SDA to go to High level via the Bus Pull-Up resistor.
6. Every I3C Device that responds to the I3C Broadcast Address sent in Step 4 shall drive the SDA line with its own 48-bit Provisional ID (using Big Endian bit order), until it loses Dynamic Address Arbitration.
   a. The 48-bit Provisional ID shall be transferred continuously, starting with the most significant bit (bit [47]), with no delimitation or ACK/NACK pulse.
   b. The Hot-Join Devices shall participate in the Dynamic Address Assignment procedure only after requesting it via an In-Band Interrupt using the Reserved Slave Address of 7'b0000_010 and RnW bit Low (i.e. Write).
7. The Main Master shall continue to drive the SCL line with the same clock, while still releasing the SDA line. The I3C Device that did not yet lose the Arbitration shall then transfer its Bus Characteristics Register (BCR) and Device Characteristic Register(s) (DCR) per ***Section 5.1.1.2.2***, until it eventually loses Dynamic Address Arbitration. See also ***Section 5.1.4.3***.
8. The Device whose concatenated Provisional ID, BCR, and DCR has the lowest value will win the Arbitration round, due to the nature of Arbitration.

Note:

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

9. The Main Master shall transfer a 7-bit wide Dynamic Address for the winning Device, in Open Drain Mode. This Dynamic Address shall incorporate the priority level that the Main Master assigns to the Device, per ***Section 5.1.6.2***. The Arbitration-winning Device shall acknowledge the assigned Dynamic Address. This Dynamic Address transfer procedure shall have the following steps:
   - The Main Master shall drive the 7-bit Dynamic Address, followed by a parity bit (PAR), which is calculated as odd parity. Odd parity is the inverse of the XOR of the 7 bits. Therefore ~XOR(dynamic_address [7:1]) is placed in position 0.
   - If the parity is valid, then the Slave shall acknowledge receipt of the Dynamic Address on the next SCL clock. If the parity is invalid, then the Slave shall passively NACK on the next SCL.

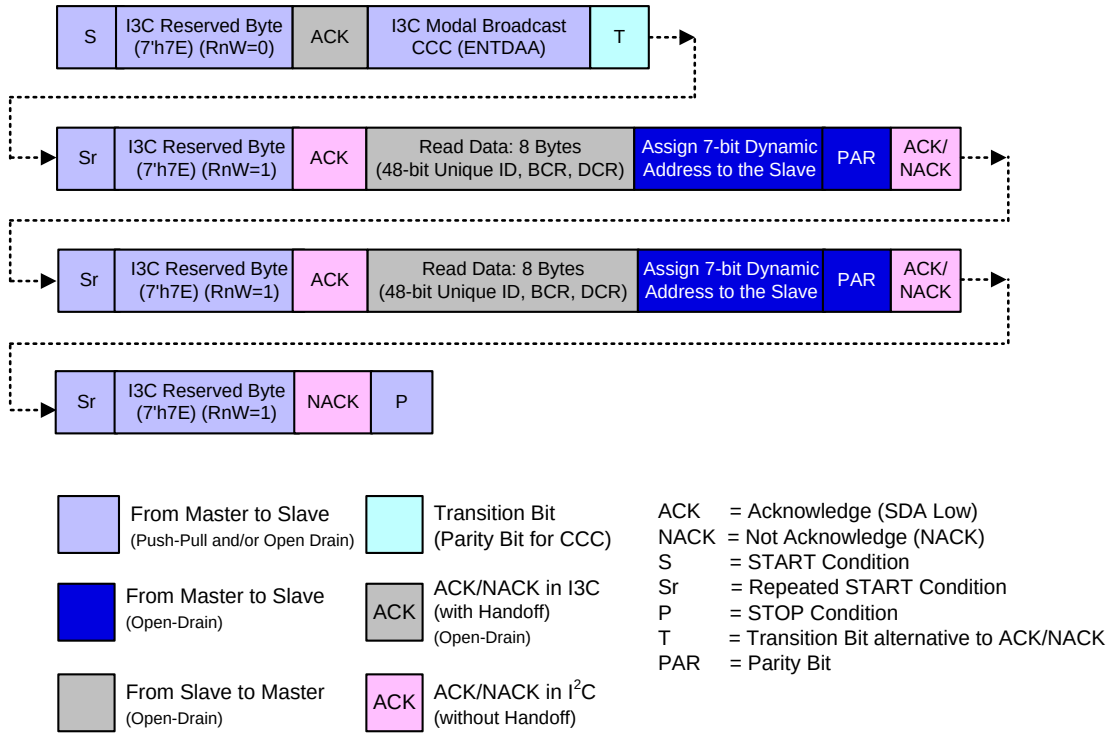


Figure 19 Dynamic Address Assignment Transaction

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

Regarding this Dynamic Address Assignment procedure, note that:

- The Main Master is able to end the Dynamic Address Assignment procedure at any time, even if some of the pre-established I3C Devices have not yet received their Dynamic Addresses.
- The Dynamic Address Assignment procedure can be started again any time the I3C Bus is in Bus Available Condition, using the Command Code **Enter Dynamic Address Assignment (ENTDAA)** (see ***Section 5.1.9.3.4***).
- If a given Slave does not acknowledge its assigned Dynamic Address, then the procedure requires the Main Master to continue from step 4. The Slave will then participate in the Address Arbitration using the same 48-bit Provisional ID, and as a result the Slave will win the Arbitration round. If the Slave does not ACK the Dynamic Address a second time, then the Main Master shall exit the Dynamic Address Assignment procedure and execute an error management procedure provided by the I3C Bus designer.

- Any Secondary Masters wishing to receive the Dynamic Addresses assigned to all Devices present on the I3C Bus can do one, or both, of the following:
  - Follow the Dynamic Address Allocation procedure. A Secondary Master that follows the same procedure will have received all the same data.
  - Use the Command Code **Define List of Slaves (DEFSLVS)** (see ***Section 5.1.9.3.7***) to acquire (eventually selectively) the Addresses and Characteristics of all Devices on the I3C Bus from the Main Master.

After the Dynamic Address Assignment process is complete, the Main Master knows which Device on the I3C Bus is the Secondary Master. The Main Master then addresses the Secondary Masters with the Command Code **Define List of Slaves (DEFSLVS)** (see ***Section 5.1.9.3.7***) and transfers the data for any and all Legacy I^2C Devices on the I3C Bus. In order to keep the I3C Bus under control, this could all be done within the same transaction (i.e. the Main Master continues the communication by issuing a Repeated START each time, with no intervening STOP).

5.1.4.3 Provisional ID Collision Detection and Correction

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

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

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

- The Main Master directs a Reset Dynamic Address request to the Dynamic Address most likely to have been duplicated, using the Command Code **Reset Dynamic Address Assignment (RSTDAA)** (see ***Section 5.1.9.3.3***).
- Alternatively, the Main Master resets the Dynamic Address Assignment process by using the Broadcast Command Code **Reset Dynamic Address Assignment (RSTDAA)** (per ***Section 5.1.9.3.3***), and then proceeding from step 3 of the procedure in ***Section 5.1.4.2*** from step 8, until the sooner of either:
  a. The expected number of I3C Devices is discovered, or
  b. A set maximum number of attempts fail. If this limit is reached, then a system error message shall be sent to the Host. To avoid freezing the entire system, a limit of three (3) attempts is recommended.

5.1.5 Hot-Join Mechanism

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

Note:

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

Hot-Join may be used for:

- I3C Devices mounted on the same board, but de-powered until needed. Such Devices shall not violate electrical limits for Slaves when depowered (or while transitioning) as defined in ***Section 6***, including capacitive load.
- I3C Devices mounted on a module/board that is physically inserted after the I3C Bus has already been configured. This specification does not attempt to address how that physical insertion is handled, however such insertion shall not disrupt the SCL and SDA lines, including respecting all electrical limits defined in ***Section 6***.

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

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

1. The Slave shall wait for at least t_{IDLE} of Bus Available Condition (see ***Table 58***).
2. After the Slave determines that the Bus is Idle it may either issue a START, or else wait for a START. (To issue a START, pull the SDA line Low and hold it Low until the SCL line goes Low.)

Note:

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

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

3. The Slave issues the Reserved Slave Address of 7'b0000_010 as an IBI, using W (write) after the START. This requests a Dynamic Address Assignment process.
4. The Current Master shall perform one of 3 actions:
   a. NACK the request. The Slave shall try again at the next START.
   b. ACK the request, but issue a Broadcast CCC to disable Hot-Join by setting the DISHJ bit in the Command Code **Enable/Disable Slave Events Command (ENEC/DISEC)** (see ***Section 5.1.9.3.1***). If another Device attempts to Hot-Join before the Master is ready to assign Dynamic Addresses to the Slaves, then the Master might need to repeat this procedure.
   c. ACK the request and then issue a Broadcast Command Code **Enter Dynamic Address Assignment (ENTDAA)** (see ***Section 5.1.9.3.4***) to start the Dynamic Address Assignment process. Since only Slaves with no Dynamic Address shall participate, this allows the newly joined Slave to receive its Dynamic Address.
5. If an **Enter Dynamic Address Assignment (ENTDAA)** Command Code is issued as described in ***Section 5.1.9.3.4***, then the Slave shall transmit its 48-bit Address as per the normal mechanism.

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

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

5.1.5.1 Failsafe Device Pads

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

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

5.1.6 In-Band Interrupt

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

5.1.6.1 Priority Level

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

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

5.1.6.2 I3C Slave Interrupt Request

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

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

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

1. Accept the IBI by providing the ACK bit. The actions available to the Current Master depend upon the value of the Slave's BCR [2] bit (in the Slave's BCR register):
    a. If the I3C Slave's BCR [2] bit is set to 1, then, per ***Section 5.1.1.2.1***, the Current Master shall read the Mandatory Data Byte that follows the accepted IBI request at any "read" clock speed allowable by the Slave. This operation is similar to a "read" from the Slave and all the related rules apply. Note that the Current Master cannot avoid receiving the Mandatory Data Byte, since it is transmitted in Push-Pull mode.

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

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

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

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

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

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

Figure 20 IBI Sequence with Mandatory Data Byte

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

2. Refuse the IBI without disabling interrupts. To do this, the Current Master simply passively NACKs to deny the IBI. The Slave shall try again after the Bus Available Condition.
3. Refuse the IBI and disable interrupts. To do this, the Current Master NACKs to deny the IBI, then sends a Repeated START, and finally sets the DISINT bit in the Command Code **Disable Slave Events Command (DISEC)** to the interrupting Slave as described in ***Section 5.1.9.3.1***. (The Current Master can set the ENINT bit in the Command Code **Enable Slave Events Command (DISEC)** at a later time.)

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

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

5.1.6.3 I3C Secondary Master Requests to be Current Master

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

1. To perform the request, the I3C Secondary Master shall wait for either a START (not a Repeated START), or a Bus Available Condition and issue its own START.
2. After a START, the I3C Secondary Master shall issue its own Dynamic Address on the I3C Bus, followed by the RnW bit of 0 to request to become Current Master.
3. If the I3C Secondary Master wins the Arbitration by having the lower Dynamic Address, and if the Current Master is currently willing to relinquish Mastership to the requester, then the Current Master shall respond with ACK.
4. After the ACK, the Current Master may issue one or more commands, and shall then issue a GETACCMST CCC (see ***Section 5.1.9.3.16***) followed by a STOP, whereupon it releases control of the SCL line, and therefore also releases Master control of the I3C Bus.
5. Following the STOP and Bus Available Condition, the I3C Secondary Master assumes the role of the Current Master and takes control of the I3C Bus.

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

1. At the end of data transfer, the Current Master (indicated by 'CM' in the Figure) controls SCL and SDA, and the New Master ('NM' in the Figure) has SCL and SDA in High-Z state.
2. The Current Master provides the rising edge of SCL, while keeping SDA Low

3. After t_{SU_STO} elapses, the Current Master drives SDA High using either an active drive, or the Open-Drain class Pull-Up. However, once SDA is High the Open Drain class Pull-Up shall be used.
4. After the New Master determines that SDA is High, and after both its Clock to Data Turnaround time and the time of flight elapse, it then takes two actions:
   a. The New Master Actively drives SCL High, overlapping with the Current Master's active drive; and
   b. The New Master enables its Open-Drain class Pull-Up, in parallel with the Current Master's Open-Drain class Pull-Up

   Neither of these actions conflicts with the Current Master.

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

   In the Figure, the t_{SCO} is shown starting as if the CM has used the Open-Drain drive of SDA High; this depicts the worst condition for this stage of the handoff, showing the latest possible moment when the NM starts overlapping the line controls with the CM.
5. After a time delay of $t_{MMoverlap}$, the Current Master:
   a. Releases SCL to High-Z; and
   b. Disables its Open-Drain class Pull-Up on SDA, and sets SDA to High-Z

   In the Figure, $t_{MMoverlap}$ is shown in the worst case: on the Open-Drain drive of SDA, and where the CM starts counting it from the lower side of the SDA rising edge. The CM must control the lines until the NM could safely take over, and that is certain after $t_{DIG\_OD\_L}$ **Min**. As a result, $t_{MMoverlap}$ shall be greater than or equal to $t_{DIG\_OD\_L}$ **Min**.
6. After t_{MMlock} (the time interval during which the New Master shall not drive SDA Low), the New Master could actively drive SDA Low, producing a START condition.

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

   In the Figure, t_{MMlock} is shown as referred to the SDA rising edge in Open-Drain driving mode, similar to I3C's t_{AVAL} and I^2C's t_{BUF}.
7. After t_{CAS} expires, the New Master may drive SCL Low, producing the first SCL falling edge.

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

Note:

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

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

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

5.1.6.4 I3C Main Master Initiating a Transaction

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

Note:

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

1) *Wait more than the minimum "Bus Available" period before starting a new communication.*
2) *Transmit the reserved I3C Broadcast Address, followed by a Repeated START and normal Messages. (The reserved I3C Broadcast Address has lower priority than any interrupts.)*

5.1.7 Secondary Master Functions

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

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

5.1.7.1 Hardware and Software Requirements

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

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

5.1.7.2 Bus Management Procedures

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

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

   The Secondary Master needs to know the meaning of the interrupt from the Slave, and needs to service the interrupt appropriately.
4. Procedure for requesting transfer of Bus control to another Master, including the Main Master

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

5.1.7.3 Reduced Functionality Secondary Masters

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

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

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

5.1.7.4 In-Band Interrupt Handling

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

5.1.7.5 Hot-Join Management

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

5.1.8 Timing Control

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

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

The I3C Timing Control framework provides:

- Flexible implementation with significant capability enhancements
- Means for synchronizing the timing references/clocks, and subsequently the events, of Slave Devices on the I3C Bus to the timing reference/clock of the Master

  This can result in reduced energy consumption at the system level.
- Transmission of timing information, with minimal complexity for the Slave Devices
- Controllable timing accuracy, suitable for the target use cases

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

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

5.1.8.1 General Principles

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

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

The elements of this agreement are:

1. Two I3C Common Command Codes:
   - Set Exchange Timing Information (SETXTIME), see ***Section 5.1.9.3.20***, and
   - Get Exchange Timing Information (GETXTIME), see ***Section 5.1.9.3.21***.

   These CCCs:
   - Identify Bus events and markers (i.e., specific SDA edge and/or SCL edge),
   - Transfer timing or control information (e.g., 'abort' command),
   - Specify which timing control procedure is in effect, and
   - Query the Slave Device for Exchange Timing support details, such as which Timing Control Mode(s) are supported, the current Timing Control state, frequency, inaccuracy, etc.
2. One defined and Mode-specific marker is sent by the Master, and used to synchronize the Slaves
3. One data collection event is sent by the Slave. It includes the time information previously requested by the Master.

5.1.8.2 Synchronous Systems and Events

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

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

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

5.1.8.3 Asynchronous Systems and Events

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

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

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

Table 13 Asynchronous Timing Control Modes

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

5.1.8.3.1 Async Mode 0: Asynchronous Basic Mode

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

5.1.8.3.2 Async Mode 1: Asynchronous Advanced Mode

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

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

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

5.1.8.3.4 Async Mode 3: Async High-Precision Triggereable Mode

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

5.1.9 Common Command Codes (CCC)

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

5.1.9.1 CCC Command Format

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

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

Note:

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

There are four categories of CCC Command:

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

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

Note:

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

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

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

   Note:

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

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

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

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

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

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

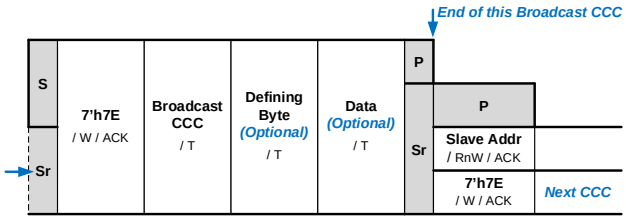


Figure 21 CCC Broadcast General Frame Format

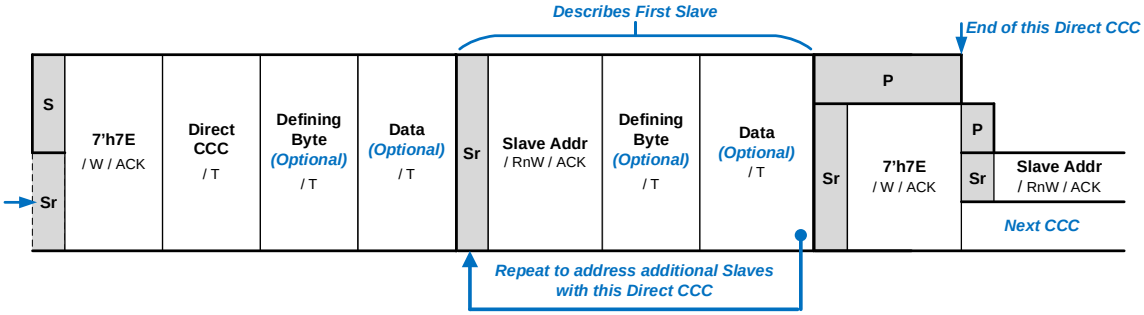


Figure 22 CCC Direct General Frame Format

Table 14 CCC Frame Field Definitions

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

5.1.9.2 Broadcast CCCs vs Direct CCCs

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

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

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

5.1.9.2.1 End of a CCC Command

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

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

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

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

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

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

5.1.9.2.2 Framing Model for Direct CCC Commands

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

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

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

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

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

5.1.9.2.3 Retry Model for Direct GET CCC Commands

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

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

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

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

1. The Slave shall NACK its Address.
2. The Master shall then emit a Repeated START, followed by the Slave Address. Note that this is a second transmission of the Slave Address. This gives the Slave extra time to prepare its response. As a further measure to give the Slave even more time to prepare its response, the Master may optionally also employ a clock delay (as per ***Section 5.1.2.5***) after the Slave's NACK and before the Master's Repeated START.
3. The Slave should respond to the retry attempt in step 2 with an ACK, and then transmit the results requested by the Direct GET CCC.

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

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

5.1.9.3 CCC Command Definitions

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

Table 15 I3C Common Command Codes

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

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

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

Command Code	CCC Type	Required	Command Name	Default	Section	Brief Description
0x92	–	–	*MIPI Reserved*	–	–	*Reserved for future use by MIPI*
0x93	Direct Set	N	**SETBRGTGT** Set Bridge Targets	–	***5.1.9.3.17***	Master tells Bridge (to/from I^2C, SPI, UART, etc.) what endpoints it is talking to (by Dynamic Address and type/ID).
0x94	Direct Get	N[4]	**GETMXDS** Get Max Data Speed	–	***5.1.9.3.18***	Master asks Slave for its SDR Mode max. Read and Write data speeds (& optionally max. Read Turnaround time)
0x95	Direct Get	N[5]	**GETHDRCAP** Get HDR Capability	–	***5.1.9.3.19***	This CCC is not included in the I3C Basic Specification. To gain access to this capability please contact MIPI Alliance.
0x96 – 0x97	–	–	*MIPI Reserved*	–	–	*Reserved for future use by MIPI*
0x98	Direct	N	**SETXTIME** Set Exchange Timing Information	–	***5.1.9.3.20***	This CCC is not included in the I3C Basic Specification. To gain access to this capability please contact MIPI Alliance.
0x99	Direct	N	**GETXTIME** Get Exchange Timing Information	–	***5.1.9.3.21***	This CCC is not included in the I3C Basic Specification. To gain access to this capability please contact MIPI Alliance.
0x9A – 0xBF	*Direct*	–	*MIPI Reserved – Direct CCCs*	–	–	*Reserved for future use by MIPI*
0xC0 – 0xD6	*Direct*	–	*MIPI Reserved for other WG's – Direct CCCs*	–	–	*Reserved for future use by MIPI*
0xD7 – 0xDA	*Direct*	–	*MIPI Debug WG Reserved – Direct CCCs*	–	–	*Reserved for use by MIPI Debug WG*
0xDB – 0xDF	*Direct*	–	*MIPI RIO WG Reserved – Direct CCCs*	–	–	*Reserved for use by MIPI RIO WG*
0xE0 – 0xFE	Direct	–	Vendor Extension – Direct CCCs	–	–	For Vendor use
0xFF	–	–	*MIPI Reserved*	–	–	*Reserved for future use by MIPI*

Note:

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

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

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

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

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

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

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

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

Table 16 ENEC/DISEC Format 1: Direct

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

Table 17 ENEC/DISEC Format 2: Broadcast

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

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

Table 18 Enable Slave Events Command Byte Format

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

Table 19 Disable Slave Events Command Byte Format

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

The supported I3C Slave/Secondary Master event types are:

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

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

- **Mastership Requests:** Enable (ENMR) / Disable (DISMR)

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

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

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

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

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

Table 20 ENTASx Format 1: Direct

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

Table 21 ENTASx Format 2: Broadcast

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

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

Table 22 Enter Activity State CCCs (ENTASx)

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

5.1.9.3.3 Reset Dynamic Address Assignment (RSTDAA)

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

Table 23 RSTDAA Format 1: Direct

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

Table 24 RSTDAA Format 2: Broadcast

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

5.1.9.3.4 Enter Dynamic Address Assignment (ENTDAA)

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

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

Table 25 ENTDAA Format

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

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

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

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

Table 26 Direct SETMWL/GETMWL Format

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

Table 27 Broadcast SETMWL Format

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

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

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

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

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

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

1. This CCC is required if both (a) any private Read Request Message(s) and/or any extended Read Request CCC(s) implemented by the Slave support a variable limit on the maximum number of data bytes that the Slave may return per Message, and (b) this limit is greater than 16 bytes.
2. This CCC is required if the Slave both (a) supports an IBI Payload (as indicated with BCR bit 1), and (b) will transmit more than one byte of private payload.

Table 28 Direct SETMRL/GETMRL Format

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

Table 29 Broadcast SETMRL Format

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

5.1.9.3.7 Define List of Slaves (DEFSLVS)

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

Table 30 DEFSLVS Format

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

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

1. **Dynamic Address:** The 7 most significant bits (Bits [7:1]) contain the current value of the Slave's Main Master-assigned 7-bit Dynamic Address, and the least significant bit (Bit [0]) is filled with the value 1'b0. For Legacy I^2C Devices, the value of the Dynamic Address shall be 7'h00.
2. **DCR Type:** The Slave's Device Characteristics Register value, or 0x00 if unknown. For Legacy I^2C Devices the value of the DCR Type shall be the Device's LVR value.
3. **BCR Type:** The Slave's Bus Characteristics Register value, or 0x00 if unknown.
4. **Static Address:** The Slave's original 7-bit static I^2C Address in the 7 most significant bits (Bits [7:1]) with the least significant bit (Bit [0]) filled with the value 1'b0. If no Static Address, then the value shall be 7'h00.

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

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

5.1.9.3.8 Enter Test Mode (ENTTM)

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

Table 31 ENTTM Format

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

Table 32 ENTTM Test Mode Byte Values

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

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

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

Table 33 Enter HDR Mode CCCs (ENTHDRx)

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

5.1.9.3.10 Set Dynamic Address from Static Address (SETDASA)

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

Note:

The SETDASA Command Code is unusual in that it addresses the desired I3C Device by its I^2C Static Address, rather than by its Dynamic Address. As a result, this CCC can only be sent to a Slave that has an I^2C Static Address.

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

Table 34 SETDASA Format 1: Primary

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

SETDASA Minimal Bus Point-to-Point Communication

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

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

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

- One active I3C Slave Device that is capable of supporting this special usage of the SETDASA CCC, or
- One or more I3C Slave Devices that act strictly as receivers, i.e., that do not use In-Band Interrupt and that will not be sent Read requests. This arrangement permits a single Master Device to, in effect, Broadcast to the Slave Devices.

Table 35 SETDASA Format 2: Point-to-Point

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

5.1.9.3.11 Set New Dynamic Address (SETNEWDA)

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

Table 36 SETNEWDA Format

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

5.1.9.3.12 Get Provisional ID (GETPID)

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

Table 37 GETPID Format

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

5.1.9.3.13 Get Bus Characteristics Register (GETBCR)

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

Table 38 GETBCR Format

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

5.1.9.3.14 Get Device Characteristics Register (GETDCR)

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

Table 39 GETDCR Format

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

5.1.9.3.15 Get Device Status (GETSTATUS)

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

Table 40 GETSTATUS Format

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

Table 41 GETSTATUS MSb-LSb Format

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

5.1.9.3.16 Get Accept Mastership (GETACCMST)

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

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

1. The Current Master ACKs the Mastership request
2. The Current Master transmits a GETACCMST command to the Secondary Master
3. The Secondary Master correctly replies with its 7-bit Dynamic Address. The value of the 7-bit Dynamic Address will be the same as the value of the Slave Address.

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

   - The Secondary Master will not acquire Mastership
   - The GETACCMST command is canceled, and
   - The Current Master can then either (a) Retry the CCC, or (b) End the transaction by providing a STOP.
4. The Current Master issues a STOP

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

   - The Secondary Master will not gain Mastership
   - The GETACCMST command is canceled (***Table 44***), and
   - The Current Master can then either (a) Retry the CCC, or (b) End the transaction by providing a STOP.

Table 42 GETACCMST Format 1: Accepted

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

Note:

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

Table 43 GETACCMST Format 2: Not Accepted

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

Table 44 GETACCMST Format 3: Incorrect Cancel

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

5.1.9.3.17 Set Bridge Targets (SETBRGTGT)

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

Table 45 SETBRGTGT Format

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

Slave Addr is either the Slave's original 7-bit static I^2C Address, or else 7'h00 if the Slave had no such static I^2C Address.

Count is the number of Bridged "Slaves".

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

- **Dynamic Address:** The 7 most significant bits (Bits [7:1]) contain the current value of the Slave's Main Master-assigned 7-bit Dynamic Address, and the least significant bit (Bit [0]) is filled with the value 1'b0. For Legacy I^2C Devices, the value of the Dynamic Address shall be 7'h00.
- **ID:** A 16-bit unambiguous identifier for the Bridged Device (two bytes)

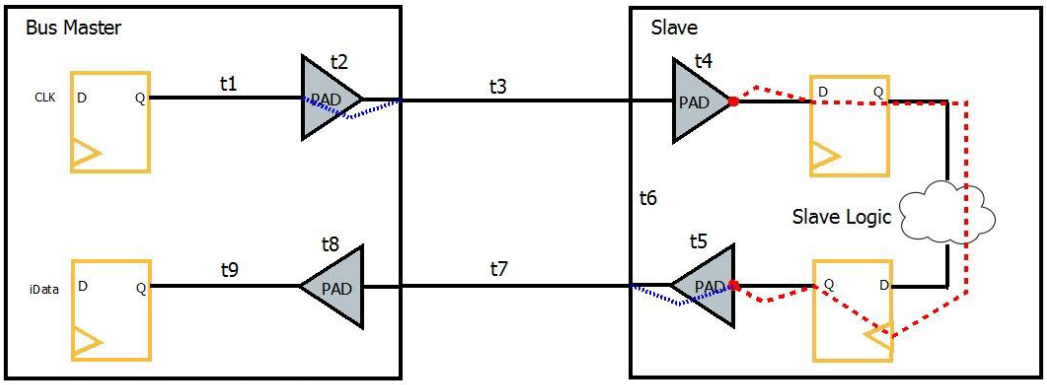
The meaning of the ID field is not defined by this Specification, and should be a contract between the Bridge and the Master (or any other entity that supplies such information to the Master). For example, the upper nibble could indicate the communication type (e.g. I^2C, SPI, UART, LIN, etc.), and the lower 12 bits could be the port (as port number and Device selector).

5.1.9.3.18 Get Max Data Speed (GETMXDS)

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

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

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



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

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

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

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

Note:

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

The Slave Device shall return either:

- **GETMXDS Format 1:** Two data bytes containing the Slave Device's Maximum Write Speed and Maximum Read Speed (see ***Table 46***), or
- **GETMXDS Format 2:** Five data bytes containing the Slave Device's Maximum Write Speed, Maximum Read Speed, and a three-byte Maximum Read Turnaround Time (see ***Table 47***).

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

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

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

Table 46 GETMXDS Format 1: Without Turnaround

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

Table 47 GETMXDS Format 2: With Turnaround

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

Table 48 maxWr Byte Format

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

Table 49 maxRd Byte Format

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

Table 50 maxRdTurn Format

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

5.1.9.3.19 Get HDR Capability (GETHDRCAP)

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

5.1.9.3.20 Set Exchange Timing Information (SETXTIME)

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

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

5.1.9.3.21 Get Exchange Timing Support Information (GETXTIME)

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

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

5.1.9.3.22 Set All Addresses to Static Address (SETAASA)

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

Table 51 SETAASA Format

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

5.1.10 Error Detection and Recovery Methods for SDR

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

5.1.10.1 SDR Error Detection and Recovery Methods for I3C Slave Devices

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

Table 52 SDR Slave Error Types

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

Note:

1. *In the ENTDAA mode, "7'hE / R" shall be excluded from the S0 error definition.*

5.1.10.1.1 Error Type S0

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

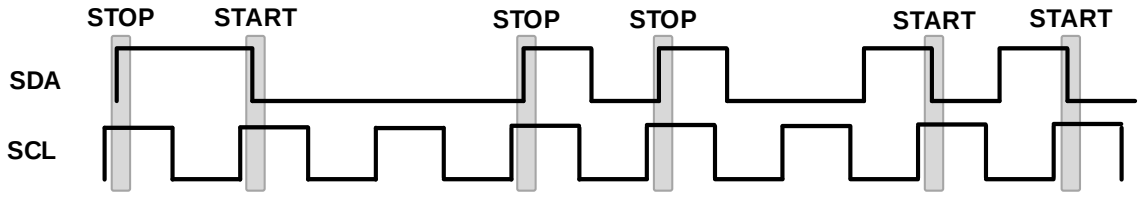


Figure 24 Example Waveform for Error Type S0

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

5.1.10.1.2 Error Type S1

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

5.1.10.1.3 Error Type S2

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

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

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

5.1.10.1.4 Error Type S3

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

5.1.10.1.5 Error Type S4

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

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

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

5.1.10.1.6 Error Type S5

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

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

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

5.1.10.1.7 Error Type S6 (Optional)

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

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

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

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

5.1.10.1.8 Optional Recovery Method for Error Types S0 and S1

An I3C Slave can recover from an Error Type S0 or Error Type S1 situation not only by detecting the HDR EXIT pattern, but also by monitoring the SCL and SDA lines. If the Slave detects that both lines stay at High level for a period exceeding 60 µs, then the I3C Slave can regard the bus as operating in non-HDR mode. The I3C Slave can then recover from the S0 or S1 situation, and wait for a STOP or a START condition to resume normal operation.

Note:

Regarding timing, HDR's slowest clock rate is 10 kHz (100 µs total cycle). The period 60 µs is derived by assuming that HDR (like DDR and TSP) will always keep an approximately even duty cycle, especially at very slow clock rates (since the only reason to go so slow is for long lines and/or large capacitive load on bus lines). 60 µs represents 60% of the duty cycle and therefore is a safe duration to wait when seeing both lines High.

The Slave can start measuring the period whenever it detects that both SCL and SDA are at High level. There is no need to start timing this period at any particular signal pattern (for example, at the STOP condition).

5.1.10.2 SDR Error Detection and Recovery Methods for I3C Master Devices

The two Error Types summarized in ***Table 53*** shall be supported for all I3C Master Devices. Each Error Type is further explained below the table.

Table 53 SDR Master Error Types

Error Type	Description	Error Detection Method	Error Recovery Method
M0	**Transaction after sending CCC**	Detect illegally formatted CCC	Stop the transmission, then send STOP and retry the transmission.
M1 (optional)	**Monitoring Error**	Master detects (through monitoring) transmitted Data different from what it intended to transmit (Does not apply during Dynamic Address Arbitration)	Stop the transmission, then send STOP and retry the transmission.
M2	**No response to Broadcast Address** (7'h7E)	Master detects NACK after Broadcast Address (7'h7E) transmission	Upon detection of NACK, Master transmits HDR Exit Pattern followed by STOP

5.1.10.2.1 Error Type M0

If the Master detects an illegally formatted CCC, then the Master shall stop the transmission, send STOP, and retry the transmission. An example of an illegally formatted CCC would be the Master receiving just one byte from the Slave in a GETMWL CCC code, since the Master expects two bytes.

5.1.10.2.2 Error Type M1 (Optional)

Note:

Although this section describes methods to mitigate or handle the Type M1 error, this error is not an expected behavior.

Error Type M1 occurs when the I3C Master detects that the transmitted data differs from what the Master intended to transmit. This might happen when the I3C Master or I3C Slaves misinterpret the RnW bit or the ACK/NACK during transactions (including IBI) defined by the I3C specification. This Section describes only two kinds of occurrence conditions for Error Type M1, and the recovery method for each kind. Note that Type M1 errors are not an expected behavior.

If an error occurs in a RnW Bit in a Private Write transfer, then the Write Data from the Master might conflict with the Read Data from the Slave. For example, the Slave could misinterpret a Private Write transfer as a Read transfer if there is an error in the RnW Bit, and as a result Write Data from the Master would conflict with Read Data from the Slave.

The Master should always monitor the Data it transmits. If the Master does so, then if the monitored Data differs from the Data the Master intended to transmit (except for Data transferred during a Dynamic Address Arbitration procedure), the Master shall consider that to be an error. If the Master detects this error, it shall stop the transmission, then send STOP and retry the transmission.

If the Slave does not reply, then the Master shall transmit a pattern including HDR EXIT and STOP. This requirement applies regardless of whether the I3C Bus is in SDR Mode or in HDR Mode at the time.

Example:

The Master should try the following sequence of steps when attempting to recover the Slave:

1. The Master transmits the Private Read several times.
   - If the Slave responds (including ACK), then the Slave temporarily did not have the Data needed for a response.
   - If there is no response from the Slave, then proceed to the next step.
2. The Master transmits a pattern including HDR EXIT and STOP.
   - If the Slave responds (including ACK), then the Slave either detected an illegal format, or detected an error.
   - If there is no response from the Slave, then no Slave with that Address is present on the I3C Bus.

5.1.10.2.3 Error Type M2

If the Master does not receive an ACK of a transmitted Broadcast Address (7’h7E), then it shall transmit the HDR Exit Pattern followed by STOP in order to recover any Slave after S0, S1, S2, S5, and S6 errors.

5.1.10.2.4 Master Error Detection and Escalation Handling

If the Master does not receive an ACK of a transmitted private Message to a Slave, and if the following conditions are all true:

- Activity State is 0 (and has been)
- Either the Slave Device has not indicated read-turnaround delays via GETMXDS, or the Slave Device has indicated a GETMXDS period and a period longer than GETMXDS has elapsed
- The Slave Device has not notified the Master that it will be going into a lower Activity State via a private contract In-Band Interrupt (and either GETSTATUS or a private activity state status),

then the Master has the following escalation options to recover the system:

1. If the Master is aware that the Slave might sometimes need extra time, then the Master can choose to try again after a short delay.
2. If that fails, then the Master shall check whether the Slave is responsive by issuing GETSTATUS to the Slave. The idea here is that the Slave might be forced to NACK a private Message due to its inner system being unready, whereas GETSTATUS is a lightweight request that does not necessarily involve the inner system.
3. If that fails, then the Master shall use M2 error handling (see ***Section 5.1.10.2.3***) to emit the sequence:

   S | 7’h7E (W) | ACK/NACK* | HDR Exit Pattern | STOP

   Note:

   ** The I3C Master doesn’t care whether the I3C Slaves’ response is ACK vs. NACK.*
4. If that fails, and if the Slave is still not responsive, then the next step depends on the value of the BCR “Offline-Capable” bit (bit [3]):
   a. If the Slave is not Offline-Capable, then:
      i. The Master can try slowing the SCL clock rate, or try slowing its effective rate by using duty cycle.
      ii. If that fails, then any further escalation is outside the scope of the I3C Specification.
   b. If the Slave is Offline-capable, then:
      i. If the Slave is known (i.e., via a private contract) to have a long wake period, and also known to monitor the I3C Bus during that wake period, then the Master simply delays and tries later, after a delay based on the known or expected wake up time. Escalation and the GETSTATUS ensure that the Slave’s Dynamic Address was issued; that should serve as the wake trigger. Either the Slave may issue an In-Band Interrupt at a later time to notify the Master that it is ready, or else the Master may initiate the retry.
      ii. If Slave is not known to have a long wake period and to always monitor the I3C Bus, then the Master marks the Slave as offline. The Master may then either retain the Slave’s Dynamic Address for re-use, or else discard it.

      The next action will be for the Slave to issue a Hot-Join request, in order to be re-joined to the I3C Bus. In the Hot-Join operation the Master will assign the Slave a Dynamic Address; the new Address may be either the same Address that the Slave used before being marked as offline, or a different Address.

This page intentionally left blank.

5.2 High Data Rate (HDR) Modes

I3C Basic does not support the HDR Modes specified in the I3C Specification version 1.x ***[MIPI02]***, but per ***Table 3***, I3C Basic Devices must detect the HDR Enter, HDR Exit, and HDR Restart patterns in order to be compatible with I3C v1.x Devices. I3C v1.x's HDR Modes are designed to transfer more data at the same Bus frequency.

Note:

It is important to note that the I3C Bus is always initialized and configured in SDR Mode, never in any of the HDR Modes.

I3C v1.x's HDR Modes have Bus-wide effect. That is, the whole I3C Bus can be put into a given HDR Mode, and once entered that HDR Mode shall remain in effect until the end of that transaction.

An HDR Mode period on the I3C Bus involves five steps:

1. The I3C v1.x Master Broadcasts an Enter HDR Mode CCC, indicating which particular HDR Mode to enter. (See the Command Codes **Enter HDR Mode 1** through **Enter HDR Mode 8 [ENTHDR1–ENTHDR8]** in ***Section 5.1.9.3.9***).
2. The I3C v1.x Devices on the Bus switch to the requested HDR Mode.
3. The I3C v1.x Master issues an HDR Command, followed by HDR data sent by the I3C v1.x Master or I3C v1.x Slave Device.
4. An HDR Restart Pattern or HDR Exit Pattern (see ***Section 5.2.1***) is sent.
   - If an HDR Restart Pattern, then a new HDR Command is sent.
5. An I3C STOP, which ends with the Bus Free Condition.

5.2.1 HDR Exit Pattern and HDR Restart Pattern

Once an HDR Mode is entered, the HDR Exit Pattern is used to leave it, always exiting back to SDR Mode. The same HDR Exit Pattern is used to exit all HDR Protocols; that Pattern does not appear in any HDR Protocol's normal data or command flow. All I3C Slaves shall detect and respond to the HDR Exit Pattern, irrespective of whether the Slave supports any particular HDR Mode. The HDR Exit Pattern Detector is specified in ***Section 5.2.1.3***.

As an alternative to the HDR Exit Pattern, the HDR Restart Pattern is also available. The HDR Restart Pattern allows multiple Messages to be sent while in HDR Mode, without forcing intervening exits to SDR Mode. That is: While the I3C Bus is in a given HDR Mode, an HDR Command can be sent to or from a Slave, and then the HDR Restart Pattern can be used to immediately send another HDR Command to or from the Slave (or a different Slave), without needing to exit the current HDR Mode between the HDR Commands. All I3C Slaves shall detect and respond to the HDR Restart Pattern while operating in any HDR Mode that the Slave supports. The HDR Restart Pattern Detector is specified in ***Section 5.2.1.4***. Note that unlike the HDR Exit Pattern, the HDR Restart Pattern is only detected by I3C Slaves that support the current HDR Mode.

5.2.1.1 HDR Exit Pattern

The HDR Exit Pattern is defined thus:

- SDA starts High, SCL starts Low
- SDA falls (from High to Low) 4 times, while SCL remains Low (for the whole time)
- Each SDA transition is separated by a time interval of at least t_{DIG_H} (see ***Section 6.2***)
- At the end of the HDR Exit Pattern, first SCL rises and then SDA rises. This is a normal I3C STOP.

Figure 25 illustrates an HDR Exit Pattern in the upper diagram (with edges to set SCL and SDA up into correct state), and an HDR Exit Pattern with appended I3C STOP in the lower diagram (SCL being High while SDA rises).

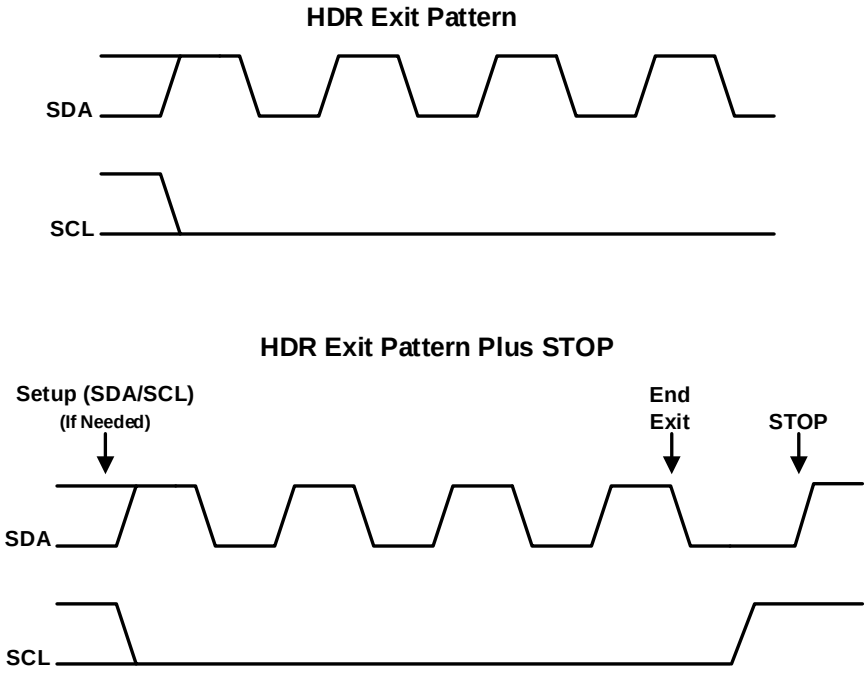


Figure 25 HDR Exit Pattern and Exit Plus STOP

5.2.1.2 HDR Restart Pattern

The HDR Restart Pattern is based on a subset of the HDR Exit Pattern. It is defined thus:

- SDA starts High, SCL starts Low (same as HDR Exit Pattern)
- SDA toggles 4 times (fall, rise, fall, rise)
- The next edge is SCL rising. SDA may change with SCL rising, but SCL shall rise.

Figure 26 illustrates an HDR Restart Pattern (with edges to set SCL and SDA up into correct state) along with the necessary SCL ending edge.

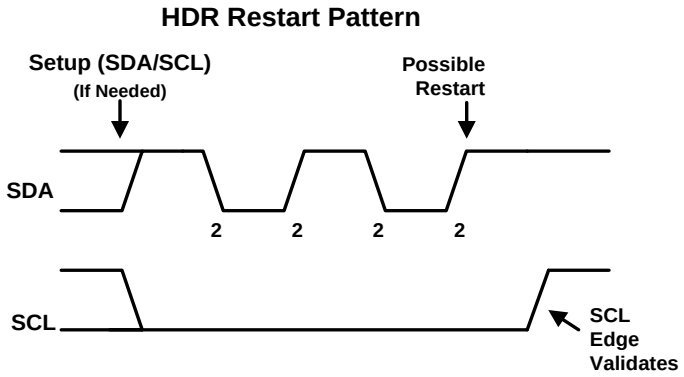


Figure 26 HDR Restart with Next Edge

5.2.1.3 HDR Exit Pattern Detector

All I3C Slaves shall include an HDR Exit Pattern Detector. The HDR Exit Pattern Detector shall be enabled only after an HDR Mode is entered, and shall be disabled after an HDR Exit pattern is detected. The HDR Exit Pattern Detector may be implemented either in digital logic, or in software (bit banged).

The remainder of this Section presents an example digital logic implementation in both schematic view and in RTL code.

The basic logic model is that SCL is held Low (0), and so SCL High (1) shall reset the detector. It also only uses falling edges of SDA, hence treats SDA as a clock. The HDR Exit Pattern Detector schematic is shown in ***Figure 27***.

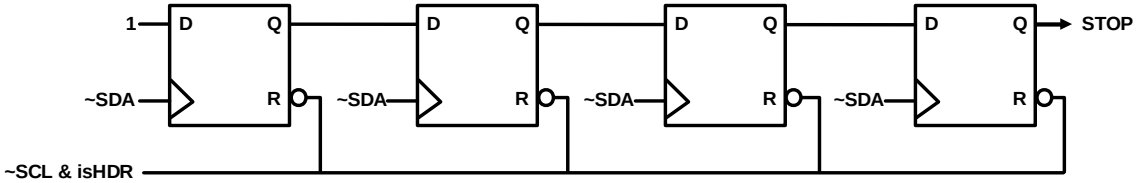


Figure 27 Example HDR Exit Pattern Detector (Schematic)

The Detector uses an inverted version of SDA as a clock (so positive edge logic, but can use negative edge logic), and is reset when SCL is High (1), or when the block is not in HDR Mode. The asynchronous nature of the reset makes this safe. This is shown in ***Figure 28***. Due to the nature of SCL vs. SDA changing at the same time with HDR Modes, the Bus Slave may see SCL change after on SDA and before the next. If the HDR Exit Pattern Detector were only using clocking logic, then it would not see any change at all (SDA posedge would always see SCL Low in this example). Because the Detector uses an asynchronous reset on SCL, a change in SCL will impact the counter, even in case (b) above. Note that SCL and SDA will still be approximately 50ns between changes. So, as shown, if SCL rises at any time, then the HDR Exit Pattern Detector shall be reset, and therefore will not mistakenly signal a false Exit.

Figure 28 Metastable Changes on SCL and SDA Do Not Break the Exit Pattern Detector

An example RTL implementation of the above schematic is shown in ***Listing 1***.

Listing 1 Example RTL Code for HDR Exit Pattern Generator

```
wire          scl_rst_n;
wire          SDA_clk_n;      // ~SDA as clock
reg[3:0]     stp_cnt;        // HDR STOP counter (shift chain)

assign scl_rst_n = ~iSCL & iIsInHDR; // SCL=1 resets
// next uses glitch free XOR for SDA clock. Only inverts
// SDA as clock if not in scan. Could add a clock mux
// so uses one common clock in scan.
SAFE_CLK_XOR sda_neg_clk(iSDA, ~scan_mode, SDA_clk_n);

// counter for 3 and 4 rising SDA when SCL is High
// (else SCL resets). Whole thing held in reset if
// not in HDR Mode
always @ (posedge SDA_clk_n or negedge scl_rst_n)
  if (~scl_rst_n)
    stp_cnt <= 4'd0;          // SCL High or not HDR
  else
    stp_cnt <= {stp_cnt[2:0], 1'b1}; // shift chain counter

assign oHDR_STOP = stp_cnt[3]; // detects Exit (STOP)
```

5.2.1.4 HDR Restart and Exit Pattern Detector

Any I3C Slave that supports at least one HDR Mode shall include HDR Restart Pattern detection. While this function is easily incorporated into the required HDR Exit Pattern Detector, or may be part of HDR Mode support, the particular design is not mandated by this Specification (i.e. is up to the manufacturer). The HDR Restart Pattern Detector may be implemented either in digital logic or in software (bit banged).

The remainder of this Section presents an example digital logic implementation in both schematic view and in RTL code, building upon the HDR Exit Pattern Detector presented in ***Section 5.2.1.3***.

The basic logic model is that SCL is held Low (0) and so SCL High (1) will reset the main detector. It also uses falling edges of SDA primarily, hence treats SDA as a clock. The HDR Restart is then detected only when two falling edges have been seen, and verifies the rising edge and then SCL change that is required for an HDR Restart.

The combined HDR Restart and Exit Pattern Detector schematic is shown in ***Figure 29***.

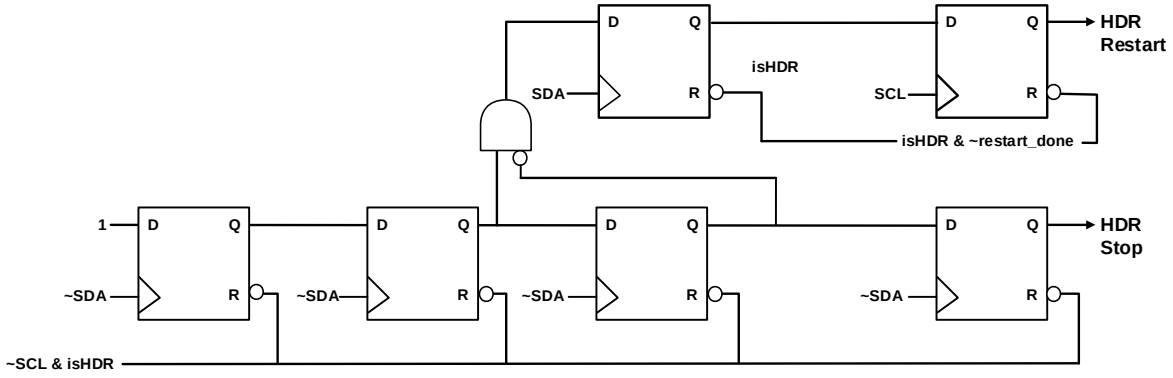


Figure 29 Combined HDR Restart and Exit Pattern Detector (Schematic)

This Detector design builds on the HDR Exit Pattern Detector. After exactly two SDA falling edges are seen, followed by a rising edge, the HDR Restart Pattern Detector checks for the rising edge of SCL. It does not matter whether SDA also falls at the same time; the rising SCL is the key. This works because even if SDA

were falling (and therefore triggering the next flop in the Exit Pattern Detector), the upper-left flop will still hold 1 because it has not yet seen a rising edge on SDA.

An example RTL implementation of the above schematic is shown in ***Listing 2***.

Listing 2 Example RTL Code for Combined HDR Pattern Detector: Exit and Reset

```
wire          scl_rst_n;
wire          restart_rst_n;
wire          SDA_clk_n;      // ~SDA as clock
reg[3:0]     stp_cnt;        // HDR STOP counter (shift chain)
reg           poss_restart;   // possible restart
reg           is_restart;

assign scl_rst_n     = ~iSCL & iIsInHDR; // SCL=1 resets
assign restart_rst_n = ~iRestartDone & iIsInHDR;

// next uses glitch free XOR for SDA clock. Only inverts
// SDA as clock if not in scan. Could add a clock mux
// so uses one common clock in scan.
SAFE_CLK_XOR sda_neg_clk(iSDA, ~scan_mode, SDA_clk_n);

// counter for 3 and 4 rising SDA when SCL is High
// (else SCL resets). Whole thing held in reset if
// not in HDR Mode
always @ (posedge SDA_clk_n or negedge scl_rst_n)
  if (~scl_rst_n)
    stp_cnt <= 4'd0;          // SCL High or not HDR
  else
    stp_cnt <= {stp_cnt[2:0], 1'b1}; // shift chain counter

assign oHDR_STOP = stp_cnt[3];// detects Exit (STOP)

// Possible Restart means exactly 2 falling edges
// of SDA and then rising edge of SDA. Actual
// Restart is from SCL then rising
always @ (posedge iSDA or negedge restart_rst_n)
  if (~restart_rst_n)
    poss_restart <= 1'b0;     // Restart ACK or not HDR
  else if (stp_cnt[1] & ~stp_cnt[2]) // after 2nd fall only
    poss_restart <= 1'b1;     // SDA rise after 2 falls
  else
    poss_restart <= 1'b0;     // else not possible restart

always @ (posedge SCL or negedge restart_rst_n)
  if (~restart_rst_n)
    is_restart <= 1'b0;       // Restart ACK or not HDR
  else if (poss_restart)
    is_restart <= 1'b1;       // SCL rises after SDA does
  else
    is_restart <= 1'b0;       // else not restart
```

5.2.1.5 Compatibility of HDR Pattern Detection and Ternary Modes

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

5.2.2 HDR Double Data Rate Mode (HDR-DDR)

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

Like SDR Mode, HDR-DDR Mode uses SCL as a clock; however unlike SDR, Data and Commands change SDA on both SCL edges (when High and when Low), effectively doubling the data rate. By contrast, in SDR Mode SDA is changed only when SCL is Low. Since SDR Mode defines it as a START or a STOP for SDA to change while SCL remains High, HDR-DDR Mode is classified as an HDR Mode in order to prevent confusion.

5.2.2.1 HDR-DDR Overview

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

5.2.2.2 HDR-DDR Command Coding

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

5.2.2.3 HDR-DDR Bus Turnaround

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

5.2.2.3.1 Command to Read Data from Slave

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

5.2.2.3.2 End of a Read Command Message

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

5.2.2.3.3 Master Termination of a Read Command Message

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

5.2.2.4 HDR-DDR Error Detection

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

5.2.2.5 HDR-DDR CRC5 Algorithm

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

5.2.3 HDR Ternary Modes (HDR-TSP and HDR-TSL)

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

I3C defines two HDR Modes that use Ternary Coding:

- HDR-TSP: Ternary Symbol for Pure Bus (no I^2C Devices)
- HDR-TSL: Ternary Symbol Legacy-inclusive-Bus

These HDR Ternary Modes are entered in the standard way (see ***Section 5.1.9.3.9***), followed by a Command and then zero or more Data Words. In HDR Ternary Modes, Commands shall be issued only by a Master. Data Words may be issued by a Master or by a Slave, depending on the particular Command (Write or Read).

5.2.3.1 HDR Ternary Signaling and Coding Protocol

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

5.2.3.1.1 Ternary Signaling

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

5.2.3.1.2 Ternary Coding Protocol

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

5.2.3.2 HDR Ternary Command Coding

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

6 I3C Electrical Specifications

6.1 DC I/O Characteristics

This Section describes the DC operating parameters of the I3C interface in two modes: Push-Pull Mode and Open Drain Mode. In Push-Pull Mode, the SDA pin drives at higher speeds with a totem-pole driver.

Two important parameters should be noted for I3C:

- The I3C interface targets nominal operating voltages of 1.2V, 1.8V, and 3.3V or less. The I3C interface is not characterized for 5V systems, but could be extended to support 5V if sufficient driver strength, reduced diameter, and/or reduced speed is used in the system.
- For peak speeds the capacitance loading allowed is reduced to 50 pF, which is much lower than Legacy I^2C. With I3C higher capacitance busses are possible, but only at reduced speeds and with reduced features (e.g. no I^2C Devices are supported).

Optionally, the I3C Basic interface also targets a nominal operating voltage of 1.0V and a 100 pF capacitive loading for new usages, such as Serial Presence Detect (SPD) in DDR5.

2236

Table 54 I3C I/O Stage Characteristics Common to Push-Pull Mode and Open Drain Mode

Parameter	Symbol	Conditions	Min	Typ	Max	Unit	Notes
Operating Voltage	V_{DD}	–	1.10	1.20	1.30	V	1
			1.65	1.80	1.95		
			2.97	3.30	3.63		
Low-Level Input Voltage	V_{IL}	–	-0.3	–	0.3 * V_{DD}	V	
High-Level Input Voltage	V_{IH}	–	0.7 * V_{DD}	–	V_{DD} + 0.3	V	
Schmitt Trigger Inputs Hysteresis	V_{hys}	–	0.1 * V_{DD}	–	–	V	
Output Low Level	V_{OL}	For V_{DD} < 1.4V: I_{OL} = 2 mA	–	–	0.18	V	2
		For V_{DD} ≥ 1.4V: I_{OL} = 3 mA	–	–	0.27	V	2
Input Current (per Input-Only I/O Pin)	I_i	-100 mV < V_i < V_{DD} + 100 mV for ≥ 1.8V nominal	-10	–	10	µA	2
		-100 mV < V_i < V_{DD} + 100 mV for < 1.8V nominal	-5	–	5	µA	2
Capacitance (per I/O Pin)	C_i	For < 1.8V nominal	–	–	5	pF	5
		For ≥ 1.8V nominal	–	–	10	pF	5
Capacitance Mismatch Between Pins	**ΔC**	Difference between SDA and SCL capacitance C_i ≤ 5pF	–	–	1.5	pF	5
		Difference between SDA and SCL capacitance C_i > 5pF	–	–	3	pF	5
Push-Pull Only							
Output High Level	V_{OH}	For V_{DD} < 1.4V: I_{OH} = -2 mA	V_{DD} – 0.18	–	–	V	2
		For V_{DD} ≥ 1.4V: I_{OH} = -3 mA	V_{DD} – 0.27	–	–	V	2
Legacy Mode with Pull-Up							
Pull-Up for Open Drain	R_p	t_r = Max rise time C_b = Bus Capacitance V_{DD} = 1.2V, 1.8V, or 3.3V	$\frac{V_{DD} - V_{OL}}{3\ mA}$ for V_{DD} ≥ 1.4V	$\frac{t_r}{0.8473\ *\ Cb}$		Ω	3, 4, 6, 7
		t_r = 120 ns C_b = 50 pF	$\frac{V_{DD} - V_{OL}}{2\ mA}$ for V_{DD} < 1.4V	2833			

Note:

1) This Specification considered only 1.2V, 1.8V, and 3.3V, with associated tolerances. Other voltage ranges are not prohibited. Any I3C Bus implementation shall ensure the correct operation of the Devices resident on the Bus, notably the inter-compatibility of their voltage ranges.

2) Negative sign for currents indicates current direction.

3) $V(t1) = 0.3 * V_{DD} = V_{DD} (1 - e^{-t1/RC})$; then $t1 = 0.3566749 * RC$
$V(t2) = 0.7 * V_{DD} = V_{DD} (1 - e^{-t2/RC})$; then $t2 = 1.2039729 * RC$
$T = t2 - t1 = 0.8473 * RC$

4) Pull-Up for Open Drain shall be switched off during I3C Push-Pull operation. May be implemented as: A Pull-Up internally; A current source internally; or an external Pull-Up resistor driven by a pin.

5) For Devices that support both 1.8V and 3.3V, the larger capacitance may be used.

6) Open-Drain never occurs on SCL

7) A weak pull-up needs to be applied as well for Master handoff

2237 **Table 55 I3C Low Voltage / High Capacitive Load I/O Stage Characteristics for Push-Pull Mode and Open Drain Mode**

Parameter	Symbol	Conditions	Min	Typ	Max	Unit	Notes
Operating Voltage	VDD	–	1.0	1.1	1.2	V	–
Low-Level Input Voltage	VIL	–	-0.2	–	0.3 * VDD	V	–
High-Level Input Voltage	VIH	–	0.7 * VDD	–	VDD	V	1
Schmitt Trigger Inputs Hysteresis	Vhys	–	0.1 * VDD	–	0.4 * VDD	V	–
Output Low Level	VOL	IOL = 4 mA	–	–	0.25 * VDD	V	2
Input Current (per Input-Only I/O Pin)	Ii	-100 mV < Vi < VDD	-10	–	10	µA	2
Capacitance (per I/O Pin)	Ci	-	–	–	5	pF	–
Capacitance Mismatch Between Pins	ΔC	-	–	–	1.5	pF	–
Push-Pull Only							
Output High Level	VOH	IOH = -4 mA	0.75 * VDD	–	–	V	2
Legacy Mode with Pull-Up							
Pull-Up for Open Drain	Rp	tr = Max rise time (100 ns) Cb = Bus Capacitance (100 pf) VDD = 1V	$\frac{V_{DD} - V_{OL}}{4\ mA}$		$\frac{t_r}{0.8473\ *\ Cb}$	Ω	3, 4, 6, 7, 8

Note:

1) V_{DD} with respect to Typical V_{DD}

2) Negative sign for currents indicates current direction.

3) $V(t1) = 0.3 * V_{DD} = V_{DD} (1 - e^{-t1/RC})$; then $t1 = 0.3566749 * RC$
$V(t2) = 0.7 * V_{DD} = V_{DD} (1 - e^{-t2/RC})$; then $t2 = 1.2039729 * RC$
$T = t2 - t1 = 0.8473 * RC$

4) Pull-Up for Open Drain shall be switched off during I3C Push-Pull operation. May be implemented as: A Pull-Up internally; A current source internally; or an external Pull-Up resistor driven by a pin.

6) Open-Drain never occurs on SCL

7) A weak pull-up needs to be applied as well for Master handoff

8) R_p Min should be decided based on the V_{IL} specification

2238

I3C supports Legacy I^2C Slaves. An important feature of an I^2C Slave is the 50 ns Spike Filter on the SDA and SCL pads. If a Spike Filter is implemented on all I^2C Devices present on the I3C Bus, then the I3C Bus may operate at maximum rated clock frequency. If any I^2C Device does not have a Spike Filter, then the I3C Bus speed is determined by the slowest Legacy I^2C Device without a Spike Filter. Other requirements of an I^2C Legacy Slave are listed in ***Table 56***.

Table 56 Legacy I^2C Device Requirements When Operating on I3C

Feature	Required	Desirable	Not Used	Not Allowed	Notes
Fm Speed	X	–	–	–	–
Fm+ Speed	–	X	–	–	–
HS and UFm	–	–	X	–	2
Static I^2C Address	X	–	–	–	–
50 ns Spike Filter	–	X	–	–	1
Clock Stretching by Slave	–	–	–	X	–
I^2C Extended Address (10 bit)	–	–	X	–	2
I3C Reserved Address	–	–	–	X	–

Note:

1) *Lack of Spike Filter will severely degrade Bus performance and eliminate certain I3C Bus features*
2) *"Not Used" means that the I3C Master will not make use of the I^2C feature, however if the Slave supports the feature, then it will not interfere with I3C Bus operation.*

6.2 Timing Specification

A key feature of I3C is to maximize the speed of data transfer and minimize the time required for low-power Devices to remain in Active Mode. In Single Data Rate Mode this is accomplished by keeping Bus capacitance low and clock speed high. For large packets of data an additional speed improvement is possible using HDR Modes, where data is transferred on every clock edge. I3C supports Legacy I^2C Fm and Fm+ Modes. ***Table 57*** gives reference timing requirements used in Legacy Mode.

2250

Table 57 I3C Timing Requirements When Communicating With I²C Legacy Devices

Parameter	Symbol	Timing Diagram	Legacy Mode 400kHz / Fm		Legacy Mode 1MHz / Fm+		Units	Notes
			Min	Max	Min	Max		
SCL Clock Frequency	f_{SCL}	–	0	0.4	0	1.0	MHz	–
Setup Time for a Repeated START	t_{SU_STA}	***Figure 30***	600	–	260	–	ns	–
Hold Time for a (Repeated) START	t_{HD_STA}	***Figure 30***	600	–	260	–	ns	–
SCL Clock Low Period	t_{LOW}	***Figure 30***	1300	–	500	–	ns	–
	t_{DIG_L}	***Figure 30*** ***Figure 45***	$t_{LOW} + t_{rCL}$	–	$t_{LOW} + t_{rCL}$	–	ns	–
SCL Clock High Period	t_{HIGH}	***Figure 30***	600	–	260	–	ns	–
	t_{DIG_H}	***Figure 30*** ***Figure 45***	$t_{HIGH} + t_{rCL}$	–	$t_{HIGH} + t_{rCL}$	–	ns	–
Data Setup Time	t_{SU_DAT}	***Figure 30***	100	–	50	–	ns	–
Data Hold Time	t_{HD_DAT}	***Figure 30***	–	–	–	–	ns	–
SCL Signal Rise Time	t_{rCL}	***Figure 30***	20	300	–	120	ns	–
SCL Signal Fall Time	t_{fCL}	***Figure 30***	20 * (V_{DD} / 5.5V)	300	20 * (V_{DD} / 5.5V)	120	ns	–
SDA Signal Rise Time	t_{rDA}	***Figure 30***	20	300	–	120	ns	–
SDA Signal Fall Time	t_{fDA}	***Figure 30***	20 * (V_{DD} / 5.5V)	300	20 * (V_{DD} / 5.5V)	120	ns	–
Setup Time for STOP	t_{SU_STO}	***Figure 30***	600	–	260	–	ns	–
Bus Free Time Between a STOP Condition and a START Condition	t_{BUF}	**–**	1.3	–	0.5	–	µs	–
Pulse Width of Spikes that the Spike Filter Must Suppress	t_{SPIKE}	***Figure 45***	0	50	0	50	ns	–

2251 During an I3C communication the drive on the SDA pin shall have the ability to dynamically switch between
2252 Push-Pull and Open Drain.

2253 **Table 58 I3C Open Drain Timing Parameters**

Parameter	Symbol	Timing Diagram	I3C Open Drain Mode		Units	Notes
			Min	Max		
Low Period of SCL Clock	t_{LOW_OD}	***Figure 34***	200	–	ns	1, 2
	$t_{DIG\_OD\_L}$	***Figure 34***	$t_{LOW\_ODmin} + t_{fDA\_ODmin}$	–	ns	–
High Period of SCL Clock	t_{HIGH}	***Figure 31***	–	41	ns	3, 4
	t_{DIG_H}	***Figure 31*** ***Figure 45***	–	$t_{HIGH} + t_{CF}$	ns	–
Fall Time of SDA Signal	t_{fDA_OD}	***Figure 34***	t_{CF}	12	ns	–
SDA Data Setup Time During Open Drain Mode	t_{SU_OD}	***Figure 32*** ***Figure 34***	3	–	ns	1
Clock After START (S) Condition	t_{CAS}	***Figure 34***	38.4 nano	For ENTAS0: 1 µ	seconds	5, 6
				For ENTAS1: 100 µ		
				For ENTAS2: 2 milli		
				For ENTAS3: 50 milli		
Clock Before STOP (P) Condition	t_{CBP}	***Figure 35***	t_{CASmin} / 2	–	seconds	–
Current Master to Secondary Master Overlap time during handoff	$t_{MMOverlap}$	***Figure 44***	$t_{DIG\_OD\_Lmin}$	–	ns	–
Bus Available Condition	t_{AVAL}	–	1	–	µs	7
Bus Idle Condition	t_{IDLE}	–	200	–	µs	–
Time Internal Where New Master Not Driving SDA Low	t_{MMLock}	***Figure 44***	$t_{AVALmin}$	–	us	–

Note:

1) This is approximately equal to $t_{LOWmin} + t_{DS\_ODmin} + t_{rDA\_ODtyp} + t_{SU\_ODmin}$

2) The Master may use a shorter Low period if it knows that this is safe, i.e., that SDA is already above V_{IH}

3) Based on t_{SPIKE}, rise and fall times, and interconnect

4) This maximum High period may be exceeded when the signals can be safely seen by Legacy I^2C Devices, and/or in consideration of the interconnect (e.g., a short Bus)

*5) On a Legacy Bus where I^2C Devices need to see Start, the t_{CAS} Min value is further constrained (see **Section 5.1.3.5**)*

*6) Slaves that do not support the optional ENTASx CCCs (see **Section 5.1.9.3.2**) shall use the t_{CAS} Max value shown for ENTAS3*

7) On a Mixed Bus with Fm Legacy I^2C Devices, t_{AVAL} is 300 ns shorter than the Fm Bus Free Condition time (t_{BUF})

2254

Table 59 I3C Push-Pull Timing Parameters for SDR Mode

Parameter		Symbol	Timing Diagram	Min	Typ	Max	Units	Notes
SCL Clock Frequency		f_{SCL}	–	0.01	12.5	12.9	MHz	1
SCL Clock Low Period		t_{LOW}	***Figure 30***	24	–	–	ns	–
		t_{DIG_L}	***Figure 31***	32	–	–	ns	2, 4
SCL Clock High Period for Mixed Bus		t_{HIGH_MIXED}	***Figure 31***	24	–	–	ns	–
		$t_{DIG\_H\_MIXED}$	***Figure 31***	32	–	45	ns	2, 3
SCL Clock High Period		t_{HIGH}	***Figure 30***	24	–	–	ns	–
		t_{DIG_H}	***Figure 31*** ***Figure 30***	32	–	–	ns	2
Clock in to Data Out for Slave		t_{SCO}	***Figure 37***	–	–	12	ns	7, 8
SCL Clock Rise Time		t_{CR}	***Figure 30***	–	–	150e06 * 1 / f_{SCL} (capped at 60)	ns	5
SCL Clock Fall Time		t_{CF}	***Figure 30***	–	–	150e06 * 1 / f_{SCL} (capped at 60)	ns	5
SDA Signal Data Hold in Push-Pull Mode	**Master**	t_{HD_PP}	***Figure 36***	t_{CR} + 3 and t_{CF} + 3	–	–	–	4, 6
	Slave	t_{HD_PP}	***Figure 38***	0	–	–	–	6
SDA Signal Data Setup in Push-Pull Mode		t_{SU_PP}	***Figure 36*** ***Figure 37***	3	–	N/A	ns	–
Clock After Repeated START (Sr)		t_{CASr}	***Figure 42***	t_{CASmin}	–	N/A	ns	–
Clock Before Repeated START (Sr)		t_{CBSr}	***Figure 42***	t_{CASmin} / 2	–	N/A	ns	–
Capacitive Load per Bus Line (SDA/SCL)		C_b	–	–	–	50	pF	–

Note:

1) $F_{SCL} = 1 / (t_{DIG\_L} + t_{DIG\_H})$

*2) t_{DIG_L} and t_{DIG_H} are the clock Low and High periods as seen at the receiver end of the I3C Bus using V_{IL} and V_{IH} (see **Figure 30**)*

3) When communicating with an I3C Device on a mixed Bus, the $t_{DIG\_H\_MIXED}$ period must be constrained in order to make sure that I^2C Devices do not interpret I3C signaling as valid I^2C signaling.

4) As both edges are used, the hold time needs to be satisfied for the respective edges; i.e., t_{CF} + 3 for falling edge clocks, and t_{CR} + 3 for rising edge clocks.

5) The clock maximum rise/fall time is capped at 60 ns. For lower frequency rise and fall the maximum value is limited at 60 ns, and is not dependent upon the clock frequency.

6) t_{HD_PP} is a Hold time parameter for Push-Pull Mode that has a different value for Master mode vs. Slave mode. In SDR Mode the Hold time parameter is referred to as t_{HD_SDR}.

7) Devices with more than 12ns of t_{SCO} delay shall set the limitation bit in the BCR, and shall support the GETMXDS CCC to allow the Master to read this value and adjust computations accordingly. For purposes of system design and test conformance, this parameter should be considered together with pad delay, bus capacitance, propagation delay, and clock triggering points.

8) Pad delay based on 90 Ω / 4 mA driver and 50 pF load. Note that Master may be a Slave in a multi-Master system, and thus shall also adhere to this requirement

The timing diagram in ***Figure 30*** depicts I3C Legacy Mode for I²C Devices. The timing parameters referenced in this diagram appear in ***Table 57***.

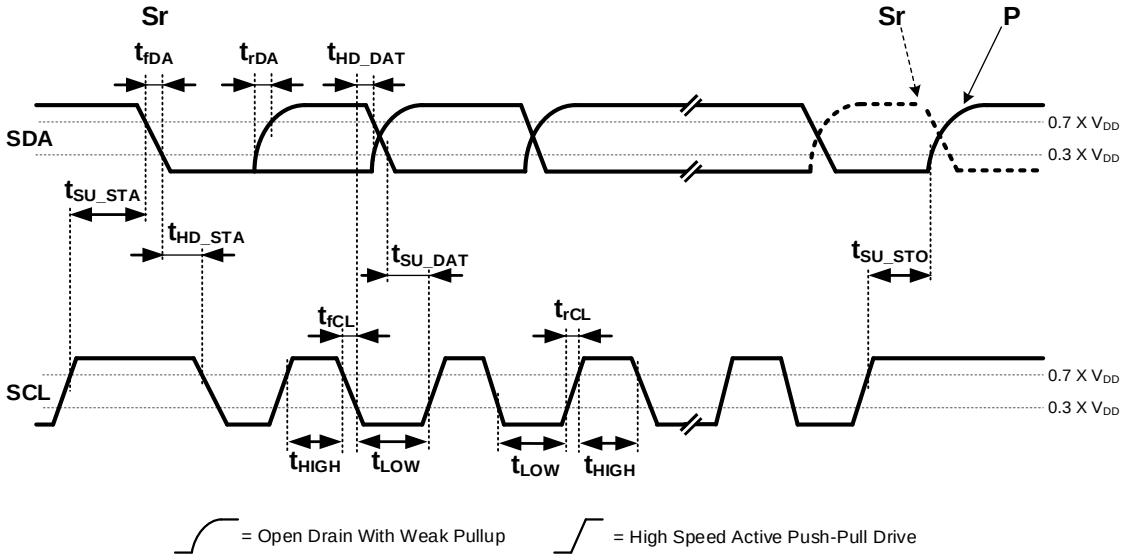


Figure 30 I3C Legacy Mode Timing

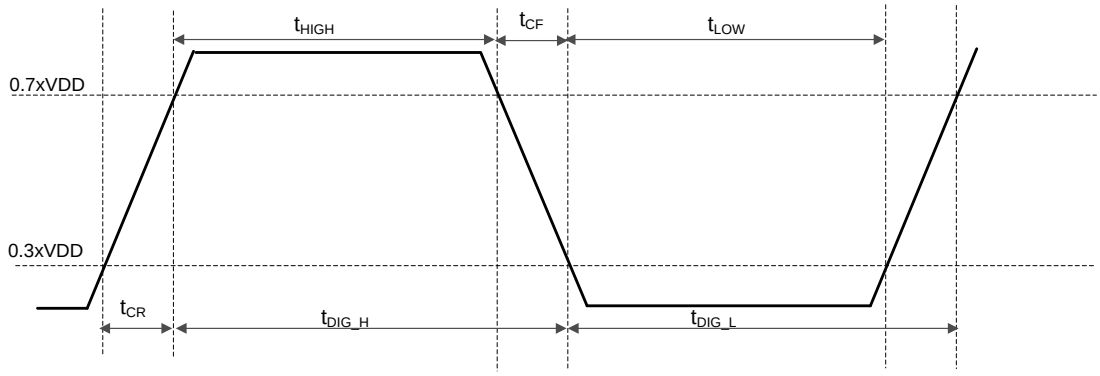


Figure 31 t_{DIG_H} and t_{DIG_L}

The start of a typical I3C communication in SDR Mode is shown in ***Figure 32***. The initial communication looks very similar to I²C, with additional commands that follow as described in ***Section 4***. The key difference is that higher clock speed is supported, up to 12.5 MHz. The higher clock speed allows Legacy I²C Devices with 50 ns Spike Filters to ignore the communications. The Master can then run in SDR Mode for I3C Devices using full 12.5 MHz timing, though it will have to slow down in order to communicate with Legacy I²C Devices. ***Figure 32*** shows the beginning of a transaction, where the Slave acknowledges its address.

Figure 33 shows the possible continuation of an I3C SDR communication, in the case where the Slave does not acknowledge its address. The Master may either STOP the communication, or else continue with a Repeated START.

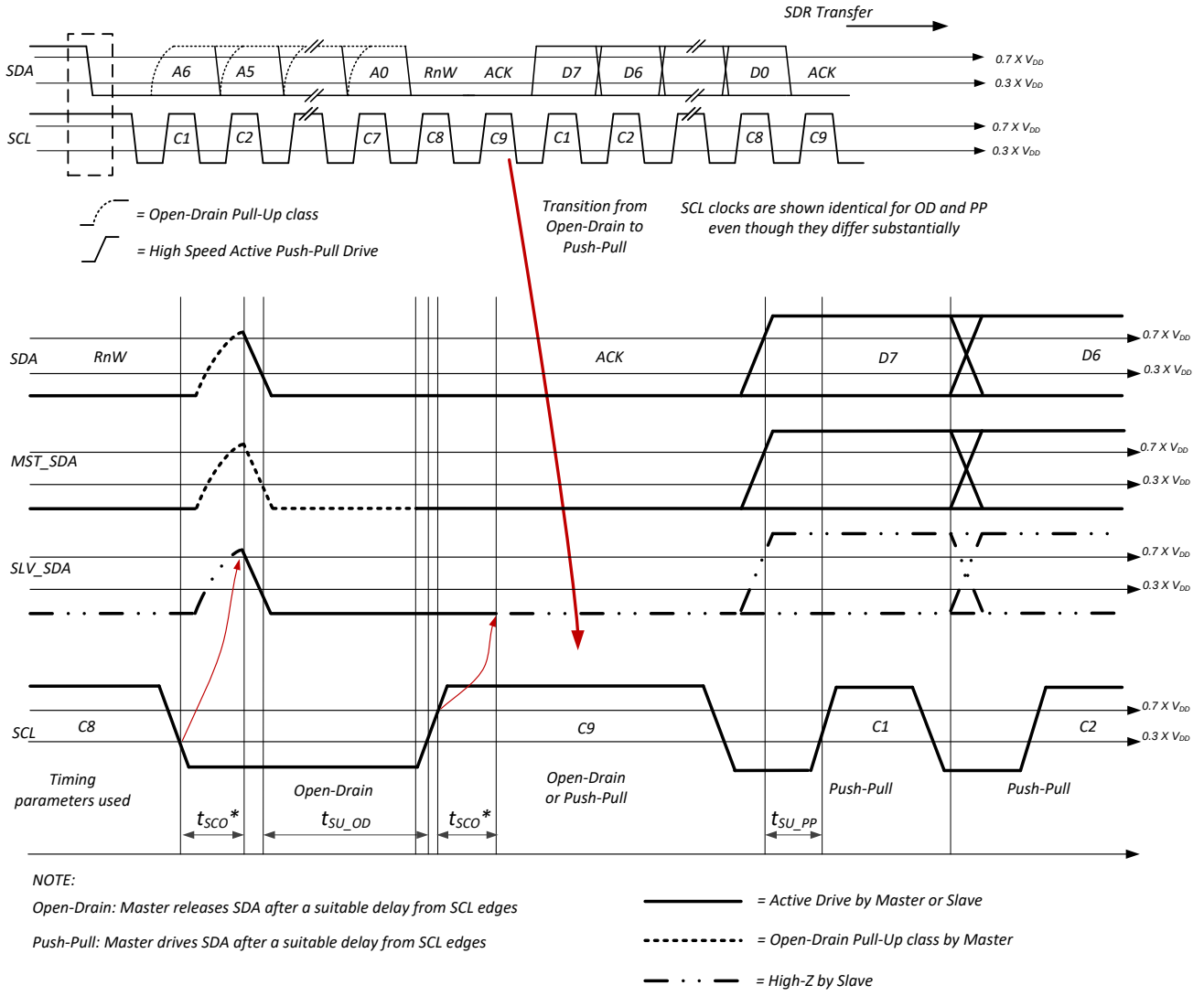


Figure 32 I3C Data Transfer – ACK by Slave

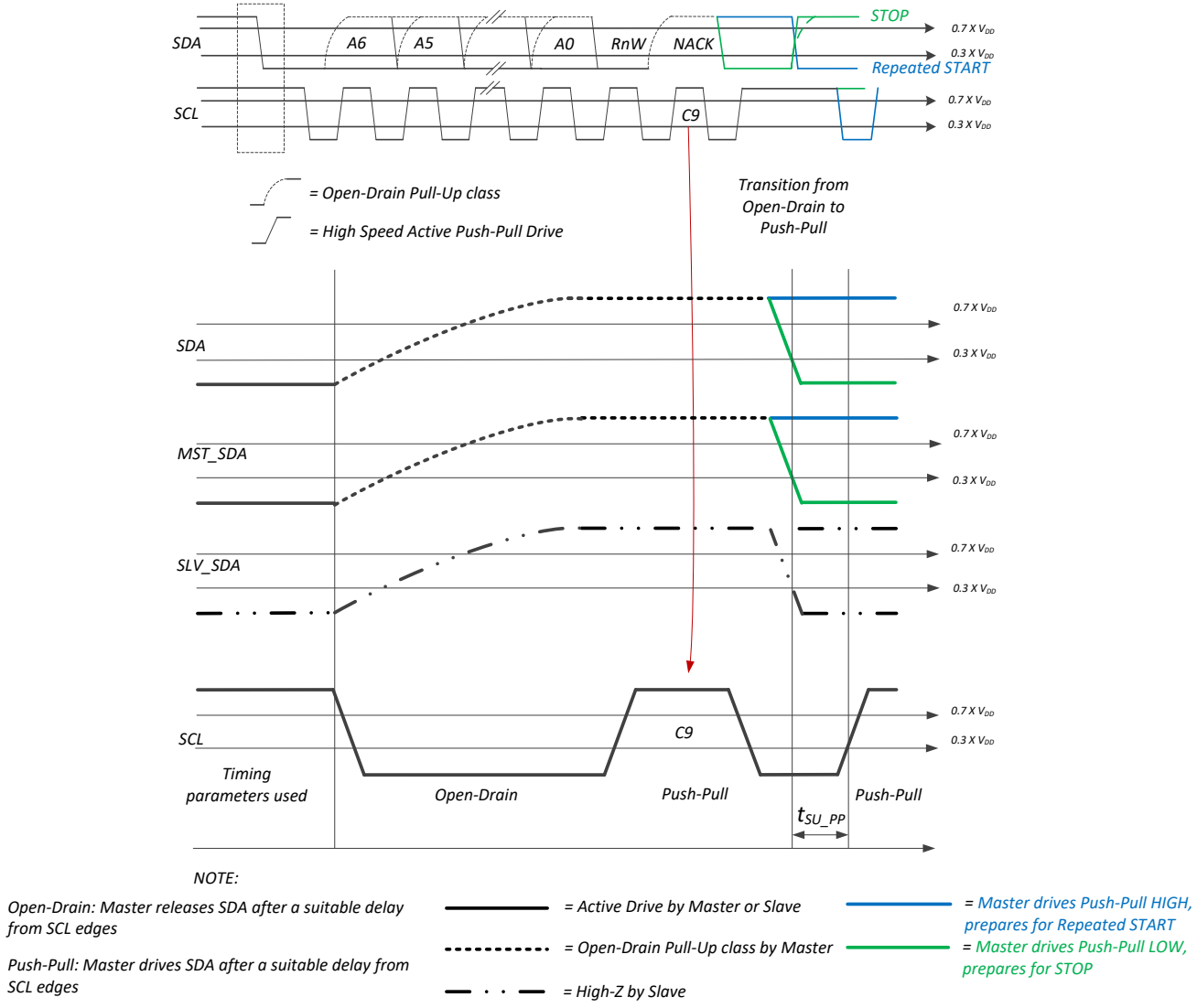


Figure 33 I3C Data Transfer – NACK by Slave

Figure 34 shows the timing parameters for an I3C START Condition, and ***Figure 35*** shows the timing parameters for an I3C STOP Condition. Notice for both the START and the STOP, the SDA pin is in Open Drain mode, as indicated by t_{DV}'s slow rise time.

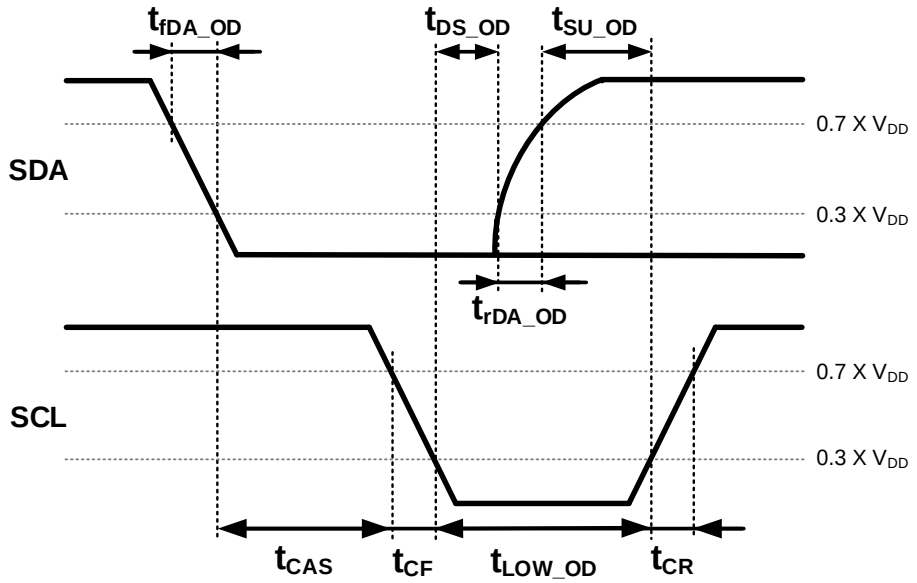


Figure 34 I3C START Condition Timing

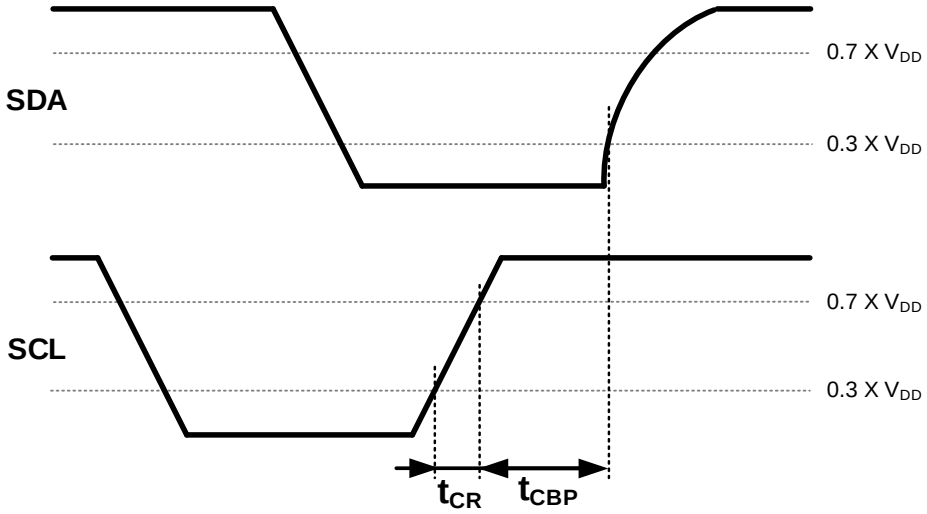


Figure 35 I3C STOP Condition Timing

Figure 36 and ***Figure 37*** illustrate the timing parameters that are unique to the Master Device and to the Slave Device, as specified in ***Table 59***. The primary difference between the two is that a Master always transmits the clock (SCL), whereas the Slave is receiver-only on the SCL pin.

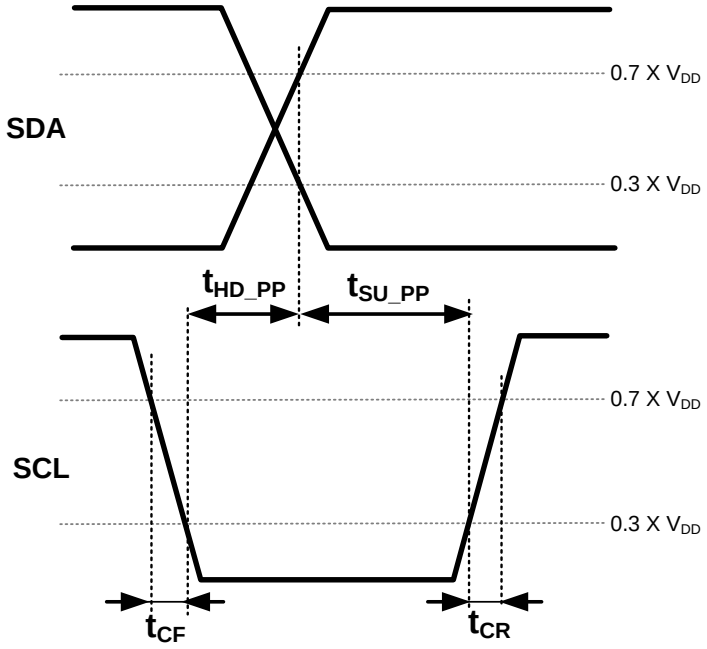


Figure 36 I3C Master Out Timing

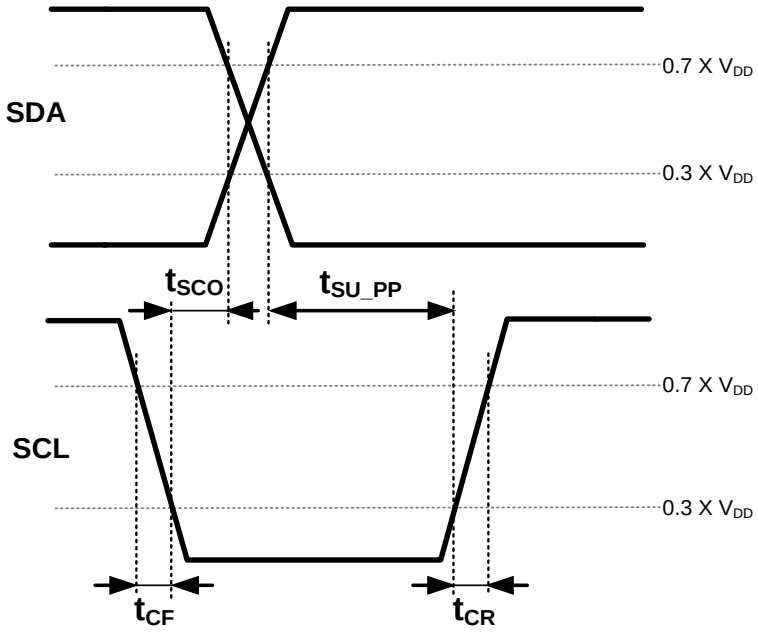


Figure 37 I3C Slave Out Timing

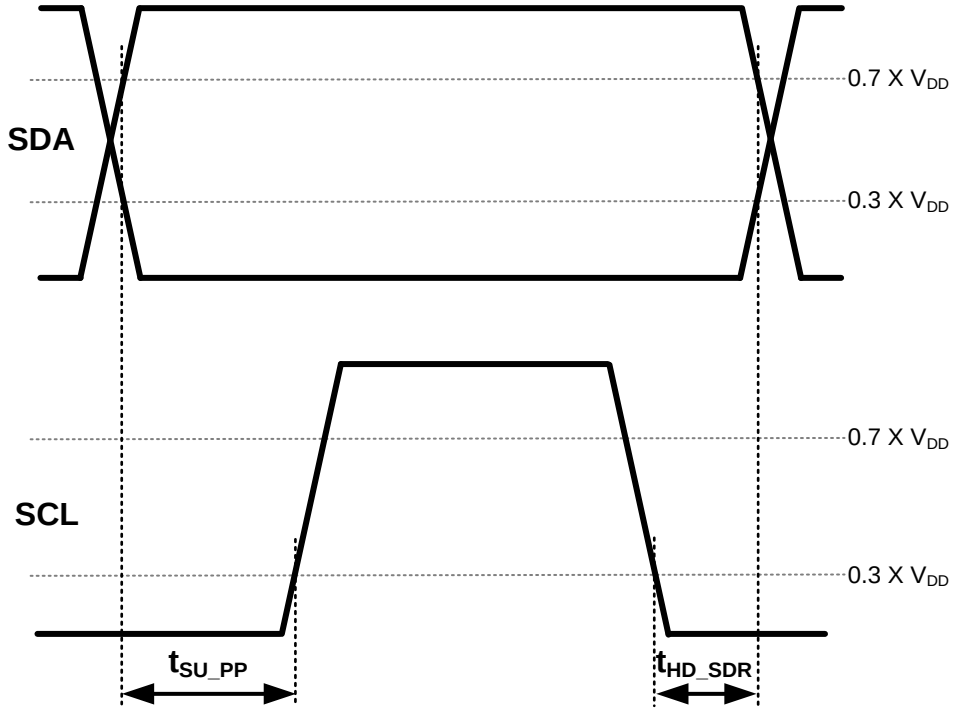


Figure 38 Master SDR Timing

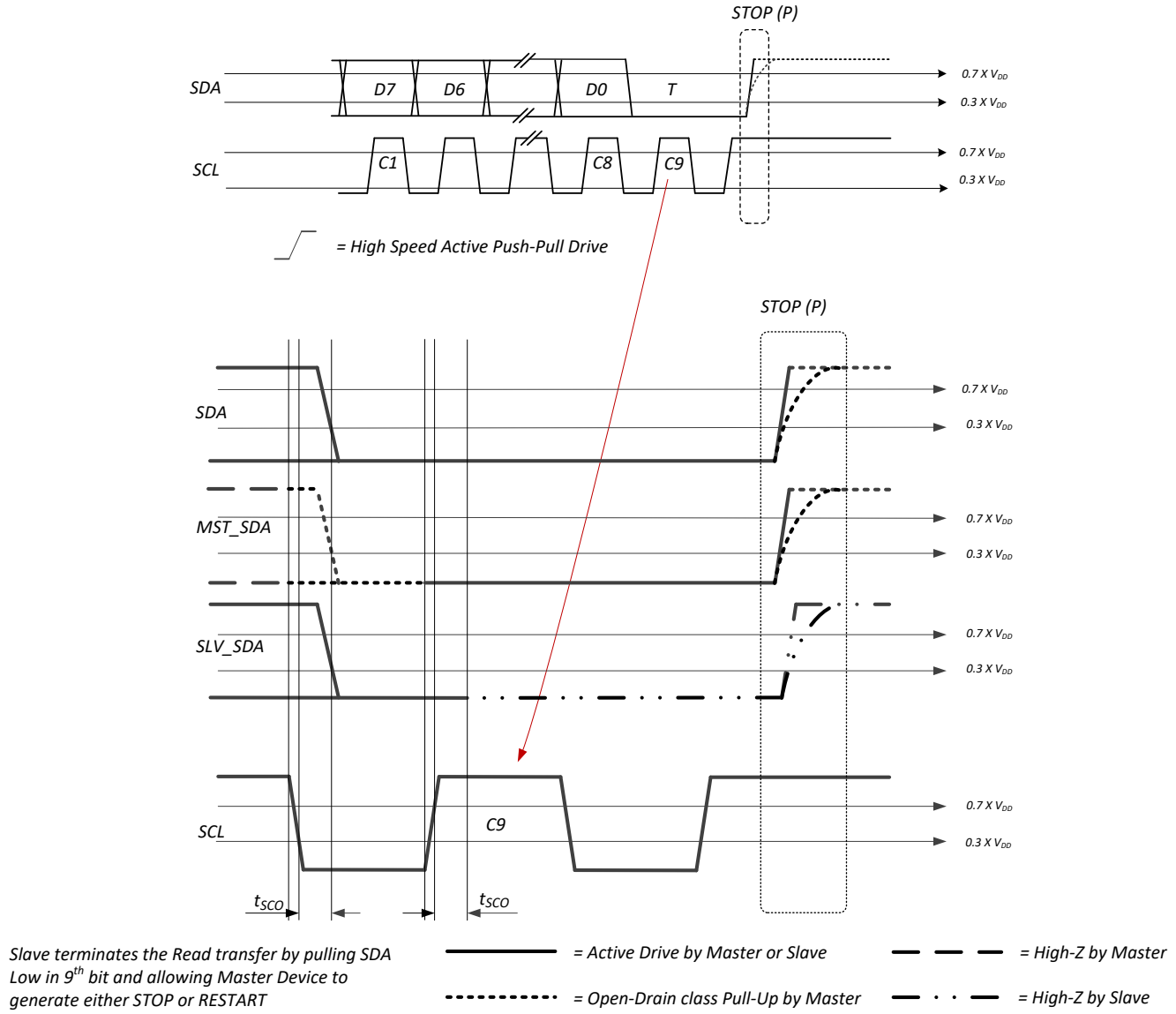


Figure 39 T-Bit When Slave Ends Read and Master Generates STOP

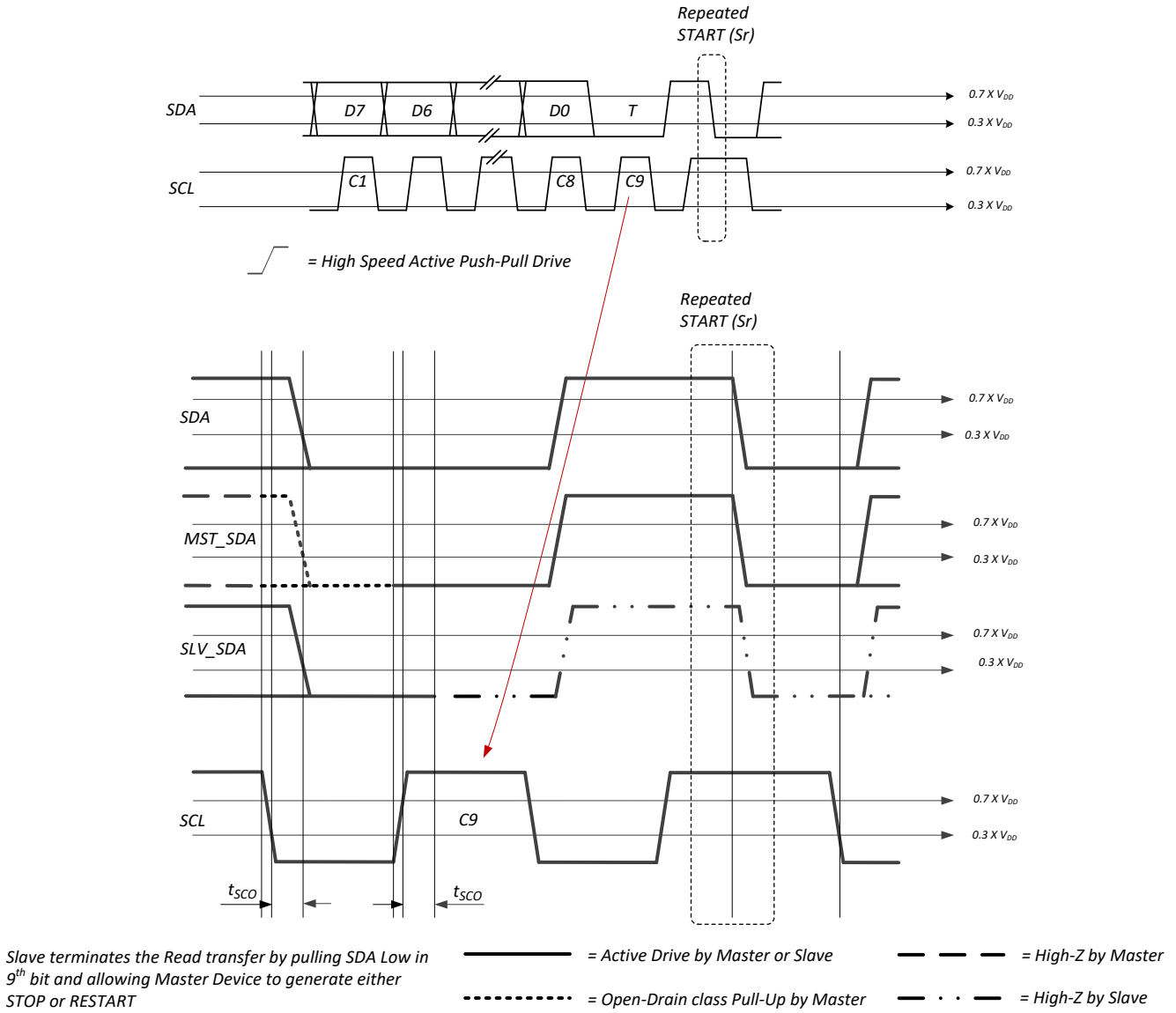


Figure 40 T-Bit When Slave Ends Read and Master Generates Repeated START

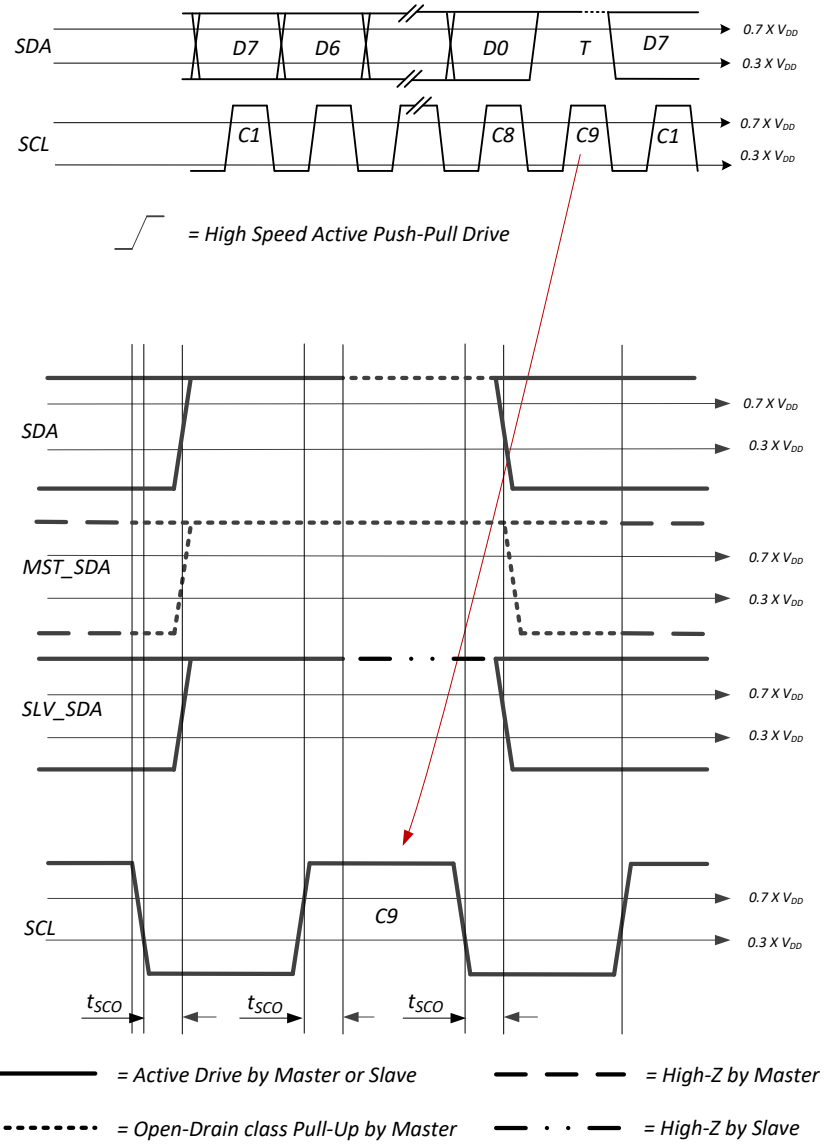


Figure 41 T-Bit When Slave and Master Agree to Continue Read Message

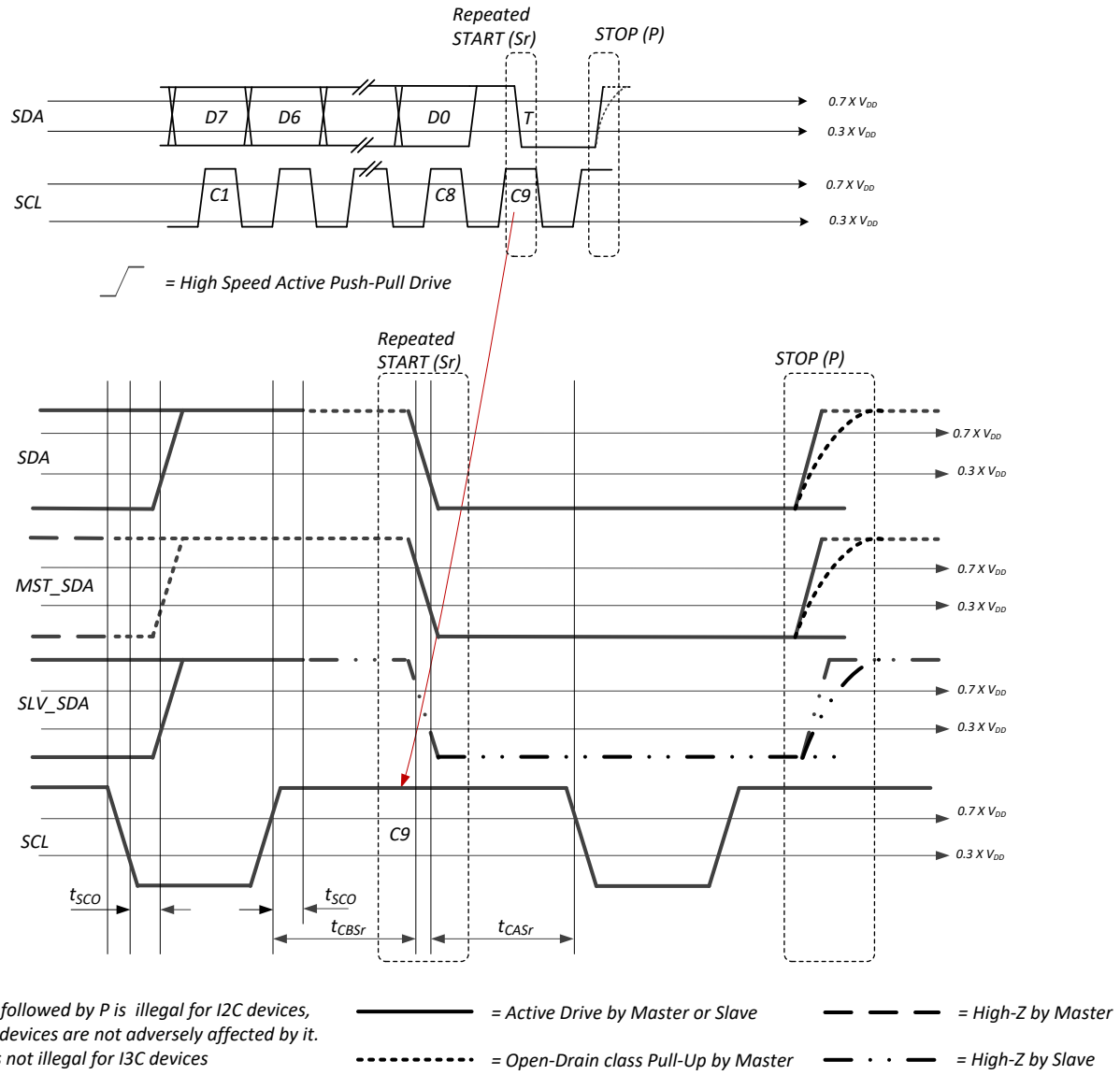


Figure 42 T-Bit When Master Ends Read with Repeated START and STOP

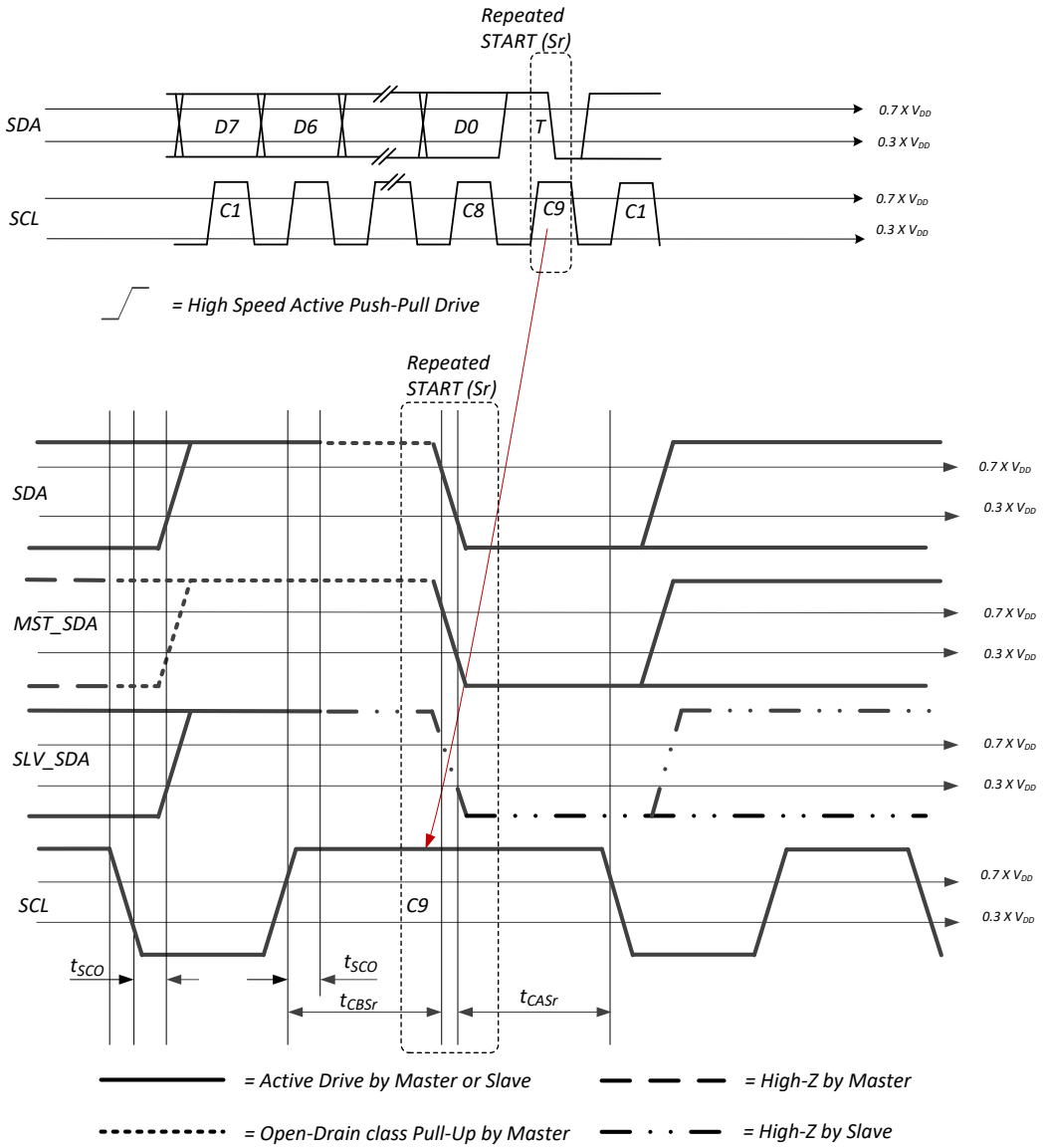


Figure 43 T-Bit When Master Ends Read via Repeated START and Further Transfer

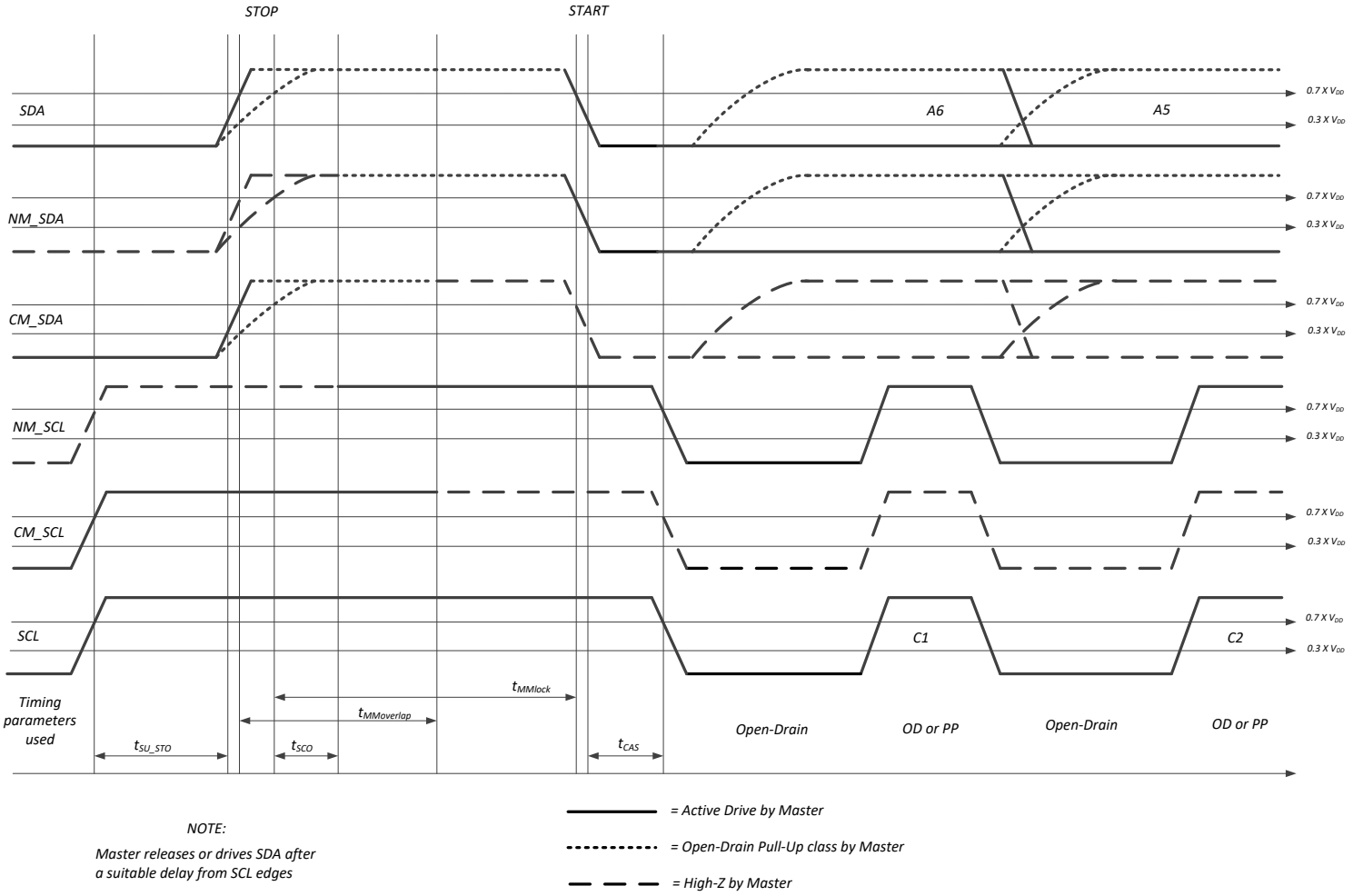


Figure 44 Master to Master Bus Handoff

Figure 45 shows the timing parameters related to the Spike Filter on a Legacy I^{2}C Device. This timing shall be met, in order to ensure that Legacy I^{2}C Devices properly ignore I3C High Speed Mode. It is recommended that both the SCL pin and the SDA pin have a Spike Filter. The timing is specified in ***Table 59***.

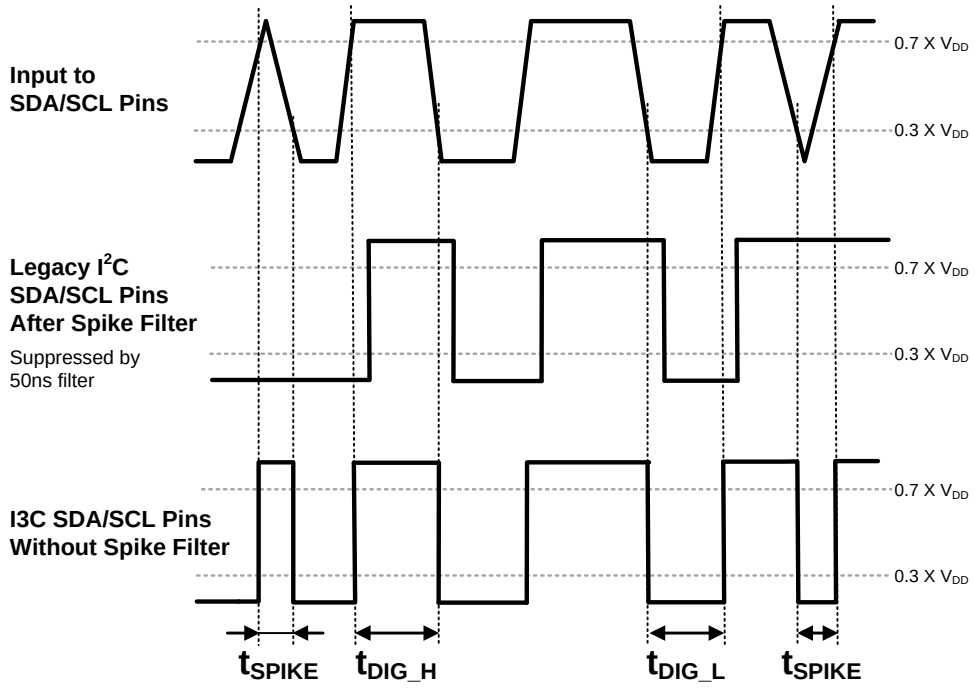


Figure 45 I^{2}C Spike Filter Behavior

2304 ***Table 60*** through ***Table 62*** detail how timing and drive strength are adjusted during transmission of an I3C
2305 Message.

2306 **Table 60 Timing and Drive for Start of New Frame: No Contention on A7**

	S	Header				Data	P
	START	ArbBit	Addr [5:0]	RnW	ACK	N Data + Parity	STOP
SDA Mode	Open Drain	Open Drain	Push-Pull	Push-Pull	Open Drain	Push-Pull	Push-Pull
Clocks	1	1	6	1	1	9 * N	1

2307 **Table 61 Timing and Drive for Start of New Frame: With Contention on A7**

	S	Header				Data	P
	START	ArbBit	Addr [5:0]	RnW	ACK	N Data + Parity	STOP
SDA Mode	Open Drain	Open Drain	Open Drain	Open Drain	Open Drain	Push-Pull	Push-Pull
Clocks	1	1	6	1	1	9 * N	1
Note: *Contention on A6, so Arbitrated over A5 to A0 and RnW.*							

2308 **Table 62 Timing and Drive for Continuation of Frame Using Repeated START**

	S	Header / Data…	Sr	Header		Data	P
	START	…	START	Addr/RW	ACK	N Data + Parity	STOP
SDA Mode	Open Drain	…	Push-Pull	Push-Pull	Open Drain	Push-Pull	Push-Pull
Clocks	1	…	1	8	1	9 * N	1
Note: *There may be any number of Repeated STARTs before the STOP.*							

This page intentionally left blank.

Annex A I3C Communication Format Details

A.1 I3C CCC Transfers

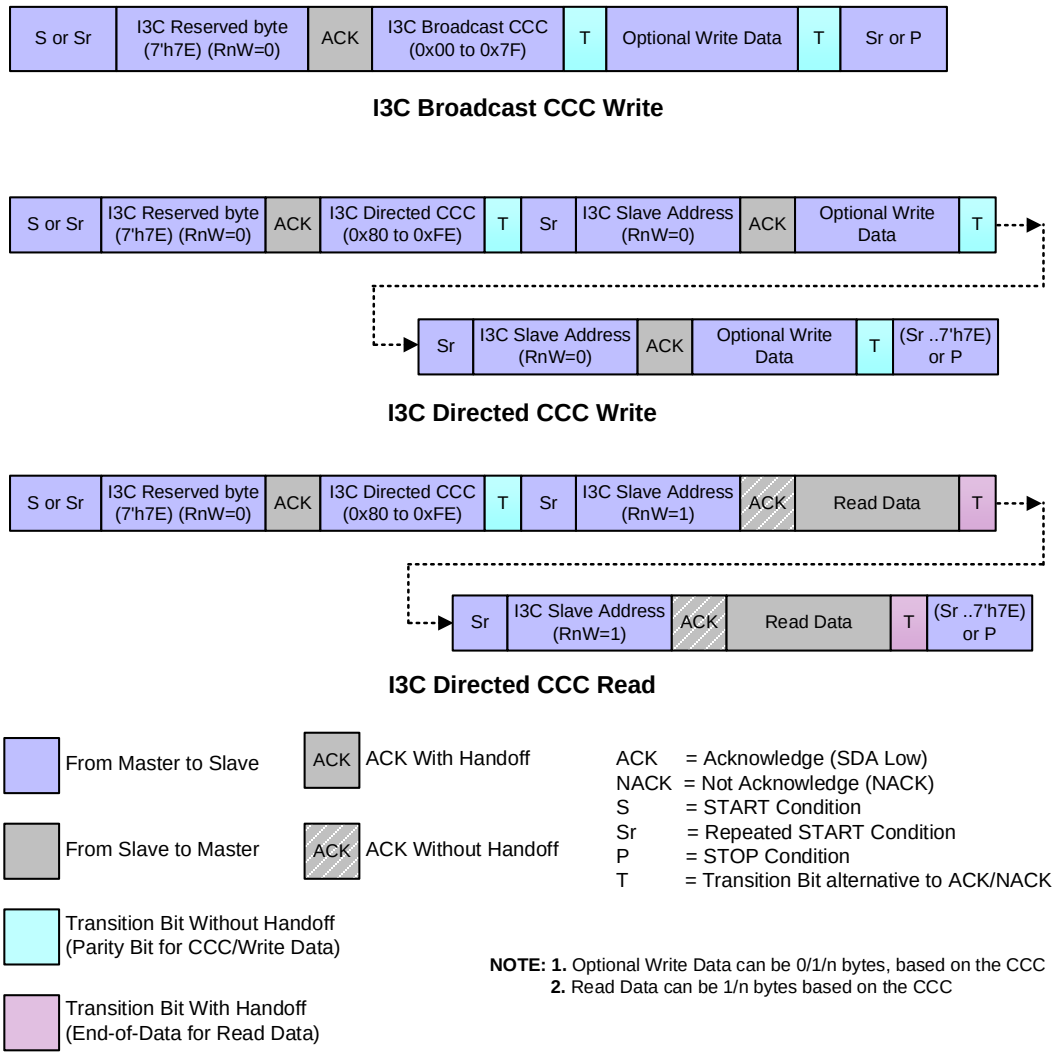


Figure 46 I3C CCC Transfers

A.2 I3C Private Write and Read Transfers

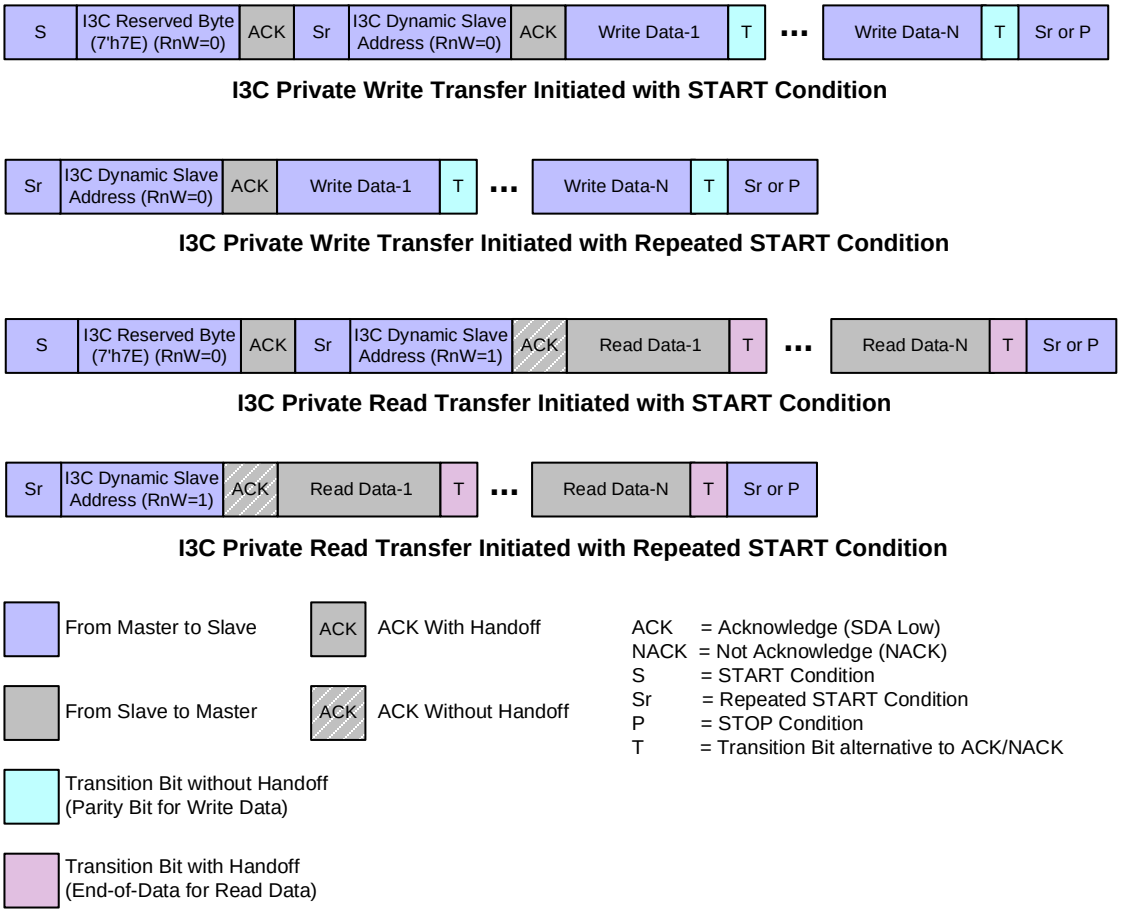


Figure 47 I3C Private Write and Read Transfers with 7'h7E Address

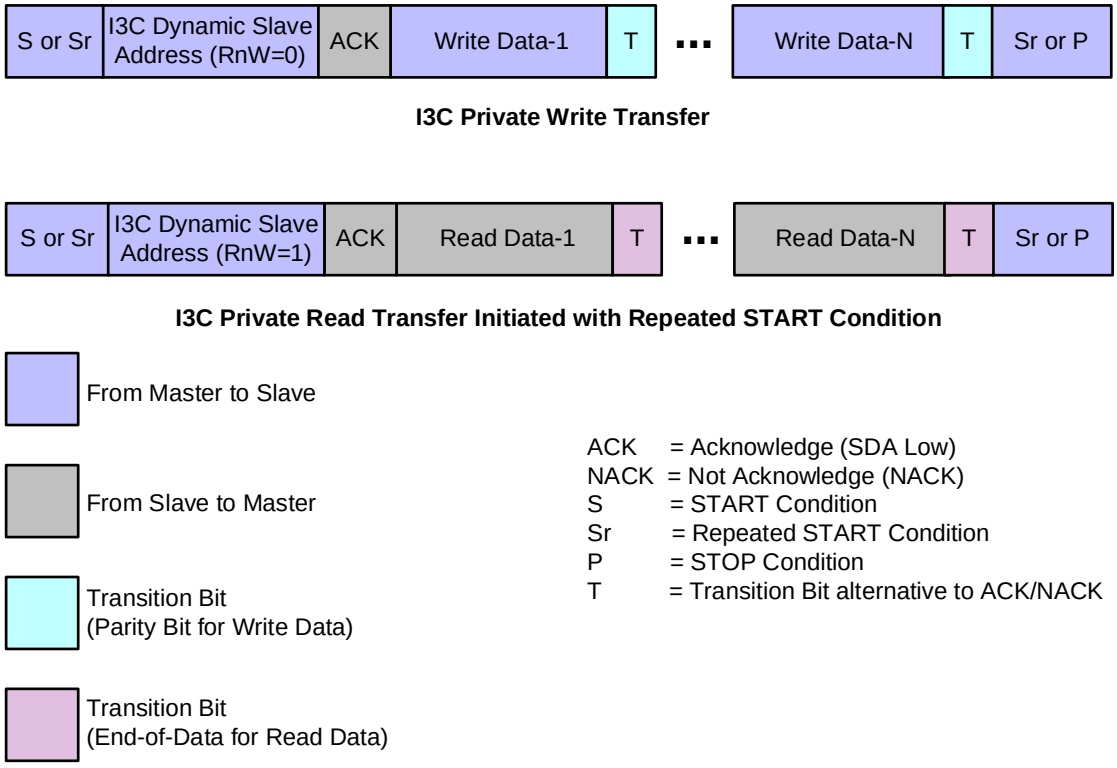


Figure 48 I3C Private Write and Read Transfers without 7'h7E Address

A.3 Legacy I²C Write and Read Transfers on the I3C Bus

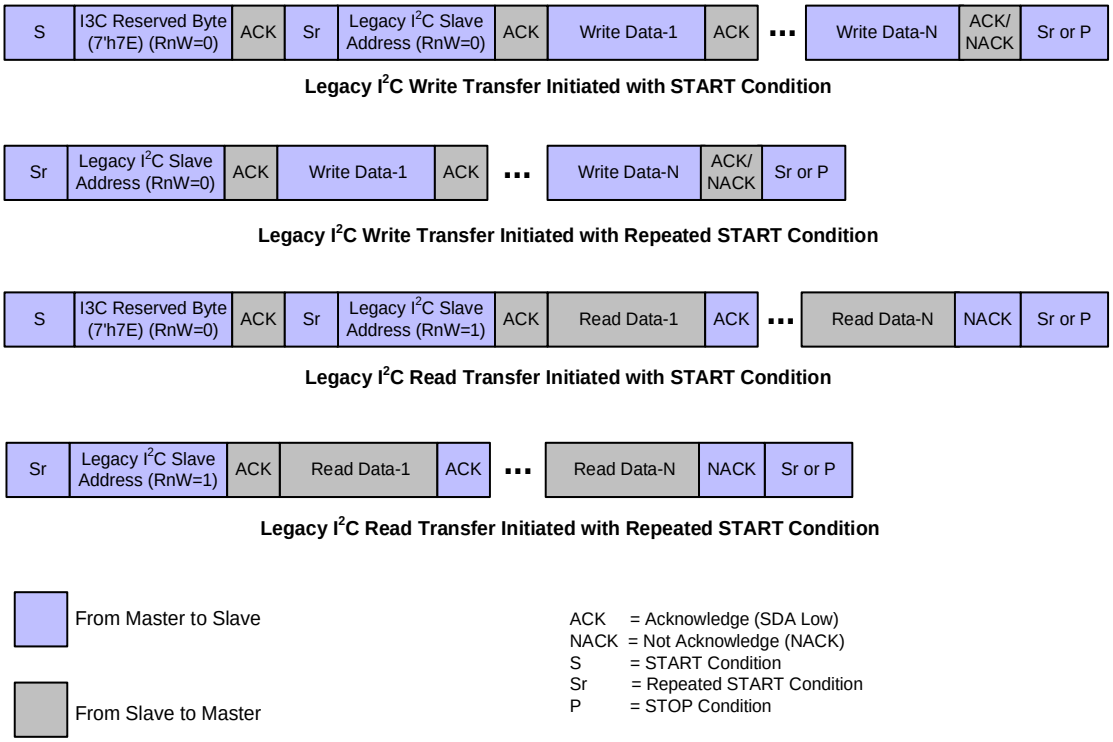


Figure 49 Legacy I²C Write and Read Transfers in I3C Bus with 7'h7E Address

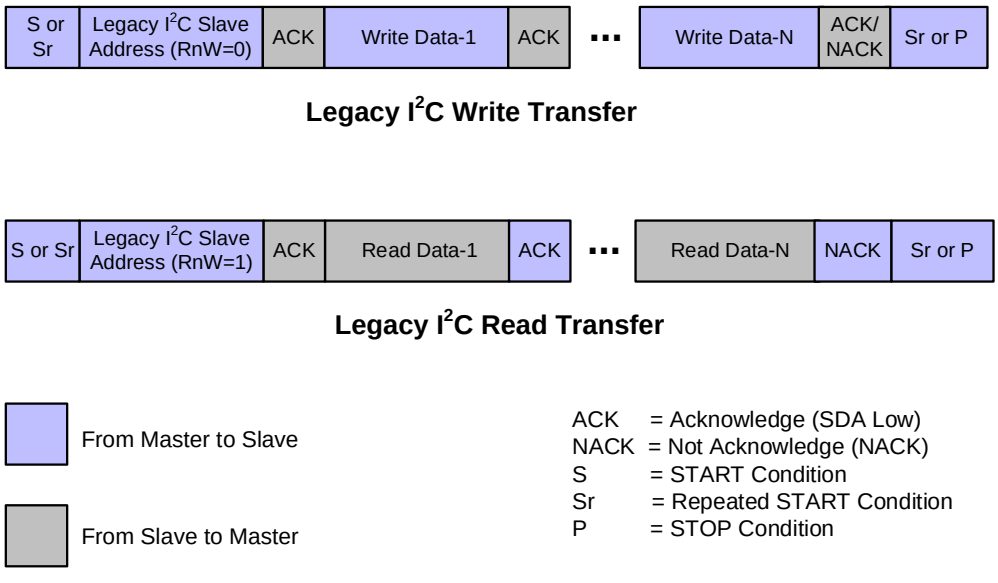


Figure 50 Legacy I²C Write and Read Transfers in I3C Bus without 7'h7E Address

A.4 Dynamic Address and Enter HDR

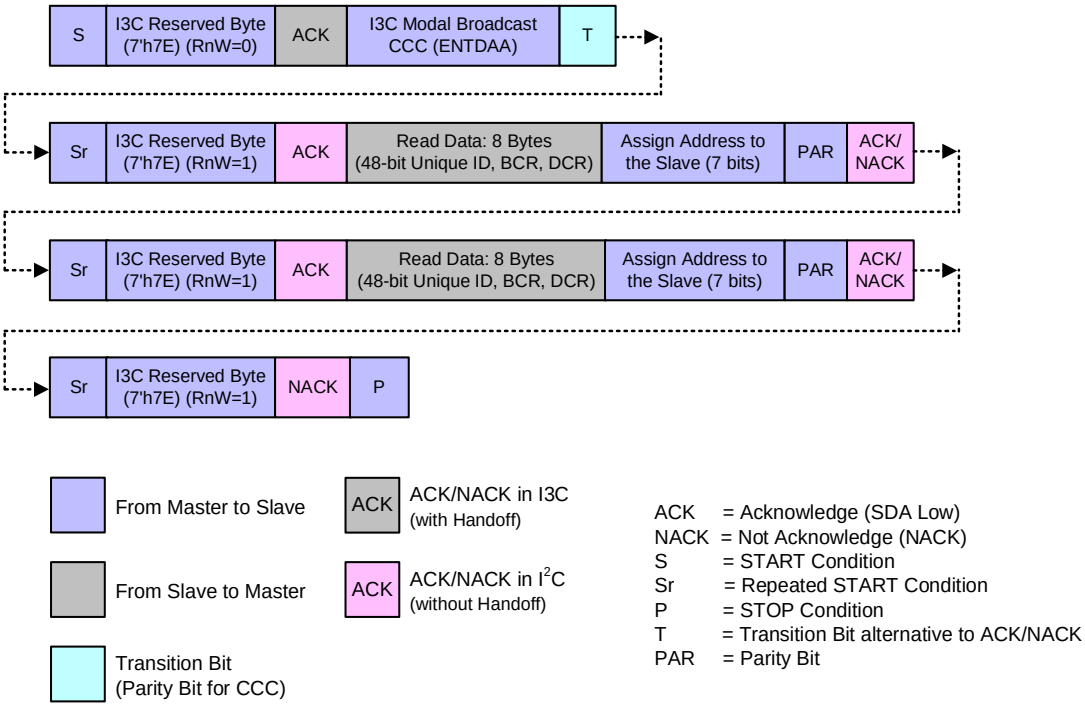


Figure 51 Dynamic Address Allocation ENTDAA CCC Bus Modal

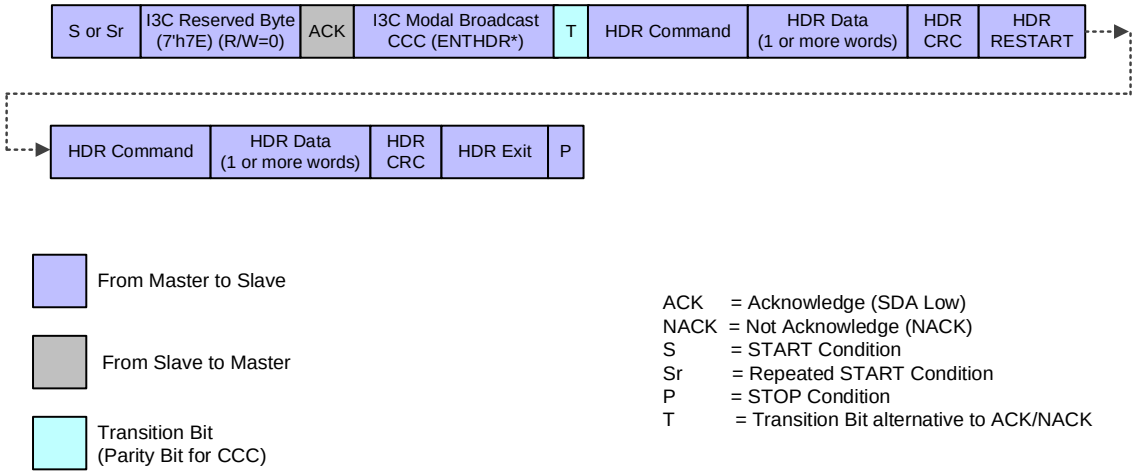


Figure 52 Enter HDR Mode CCC Bus Modal

This page intentionally left blank.

Annex B SDR Mode Error Type Origins

B.1 Error Types in I3C CCC Transfers

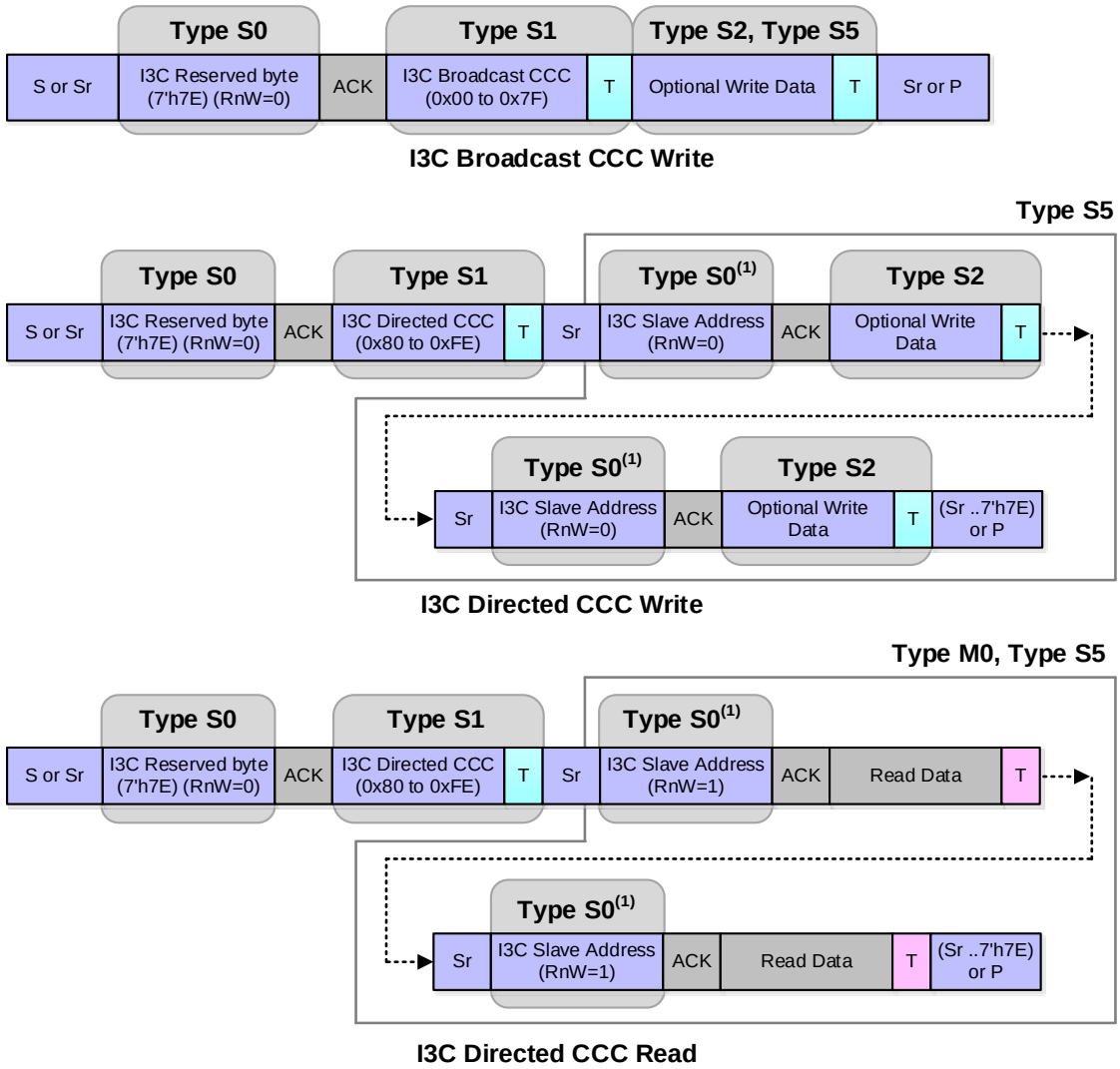


Figure 53 Error Type Origins: I3C CCC Transfers

B.2 Error Types in I3C Private Read and Write Transfers

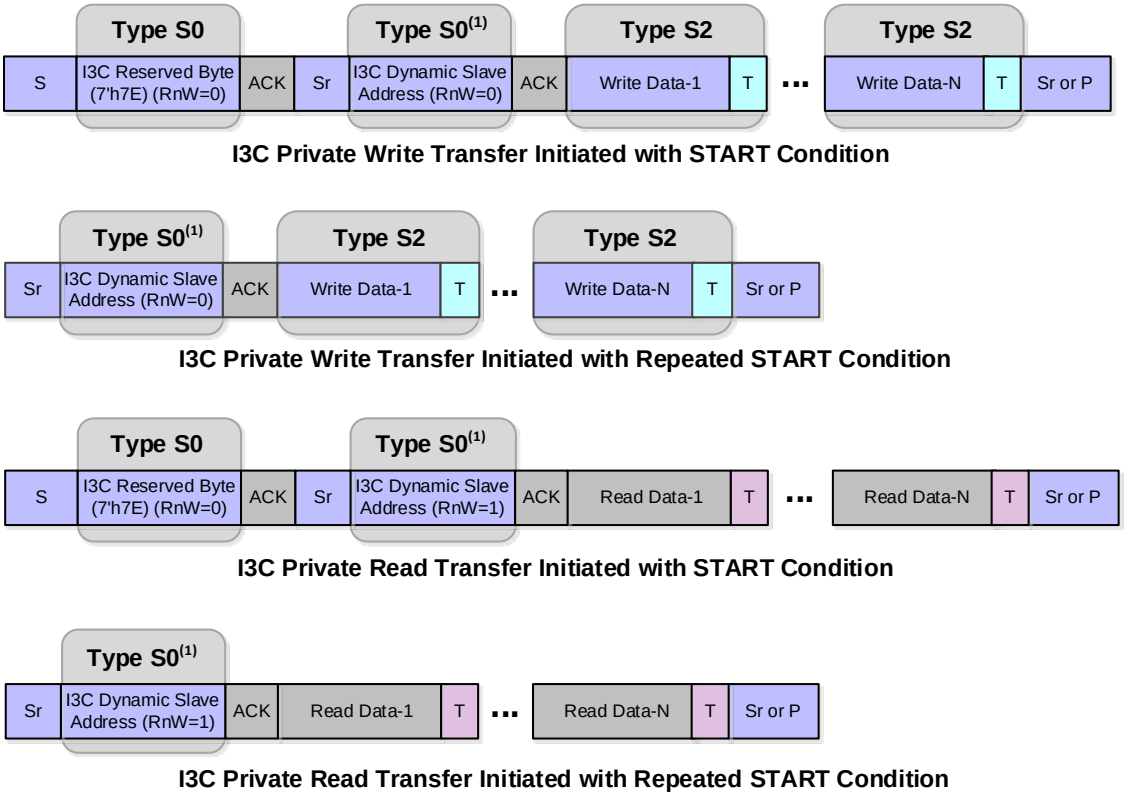


Figure 54 Error Type Origins: I3C CCC Private Write & Read Transfers with 7'h7E Address

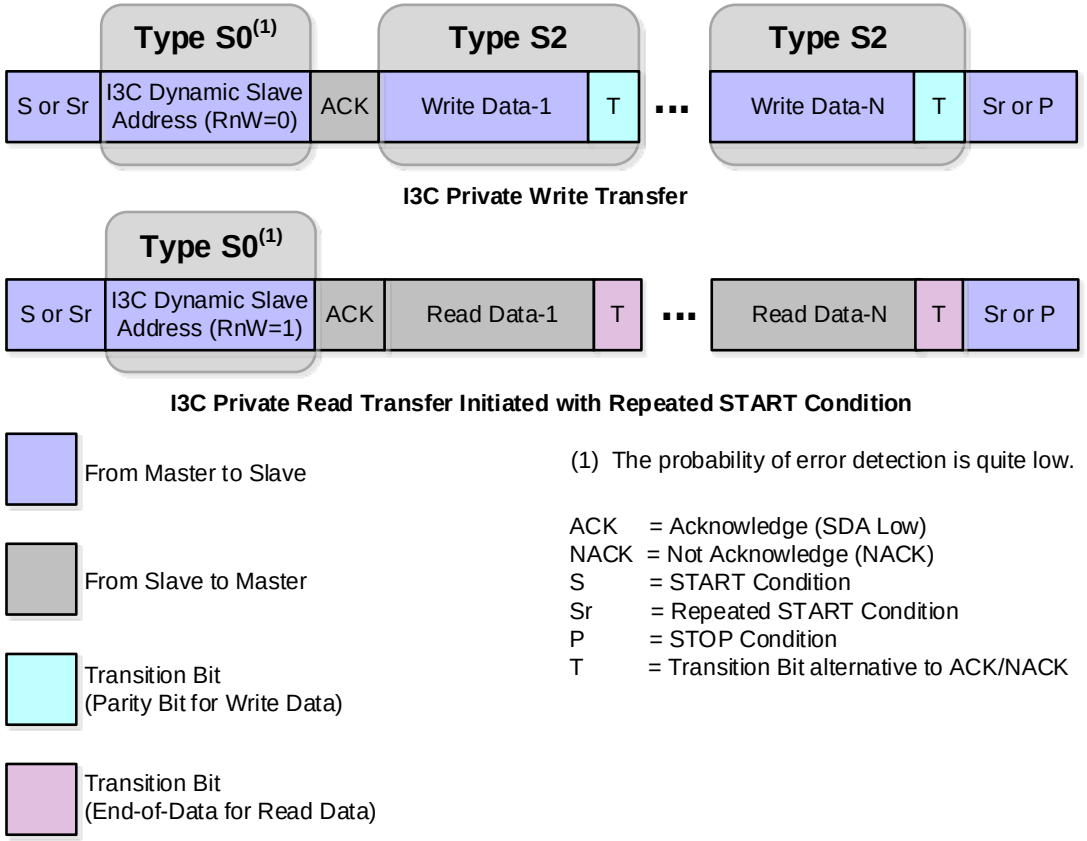


Figure 55 Error Type Origins: I3C Private Read Transfers without 7'h7E Address

B.3 Error Types in Dynamic Address Arbitration

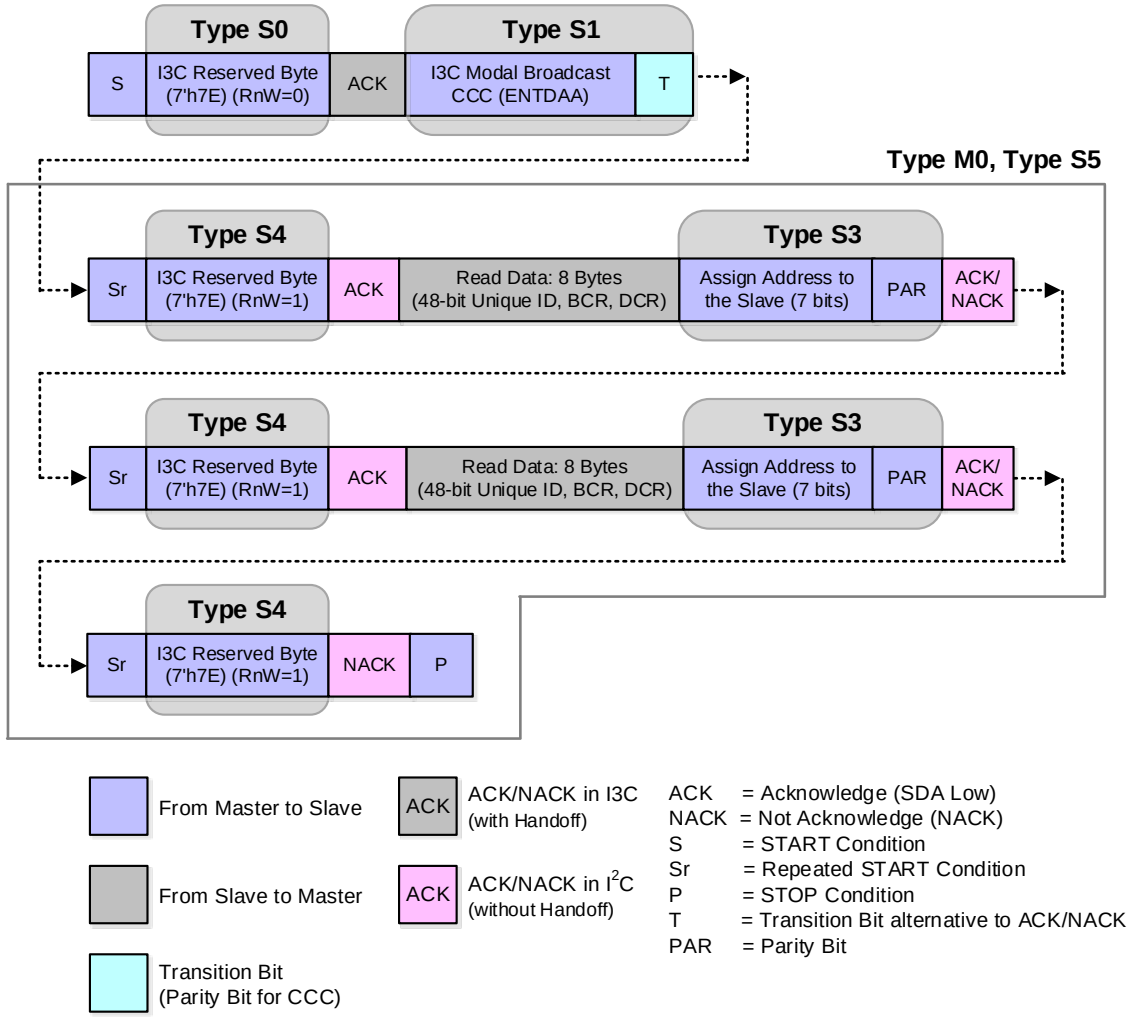


Figure 56 Error Type Origins: Dynamic Address Arbitration

Annex C I3C Master FSMs

Figure 57 shows the key transmission Modes and their entry, resume and exit sequences. It includes the SDR transmission states, and differentiates these from other capabilities and Modes. It captures the states at which Arbitration happens as well.

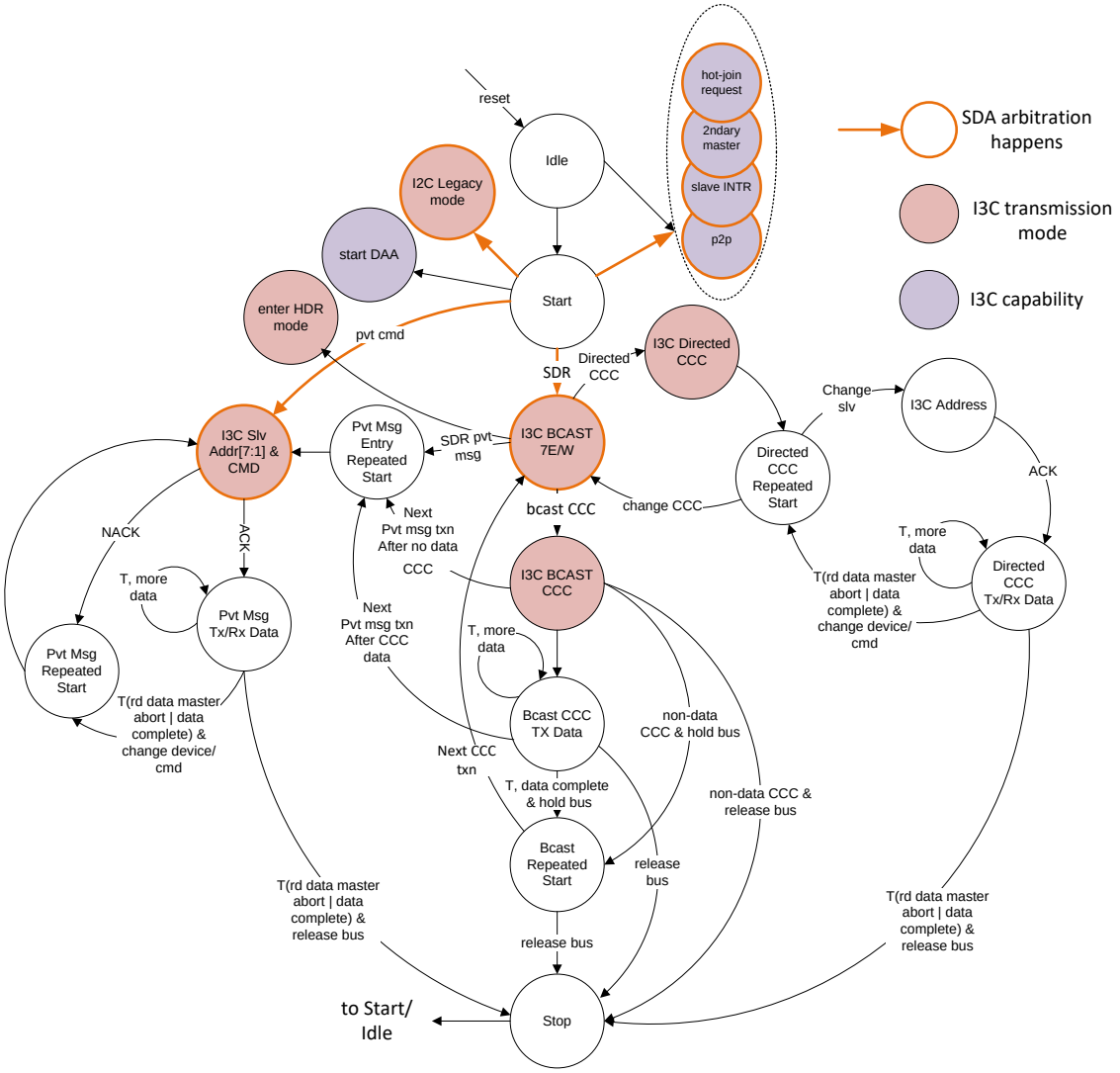


Figure 57 I3C Main Master FSM

Figure 58 through ***Figure 62*** represent I3C features including In-Band Interrupts, Secondary Master, Dynamic Address Assignment, and Hot-Join.

Slave Interrupt request

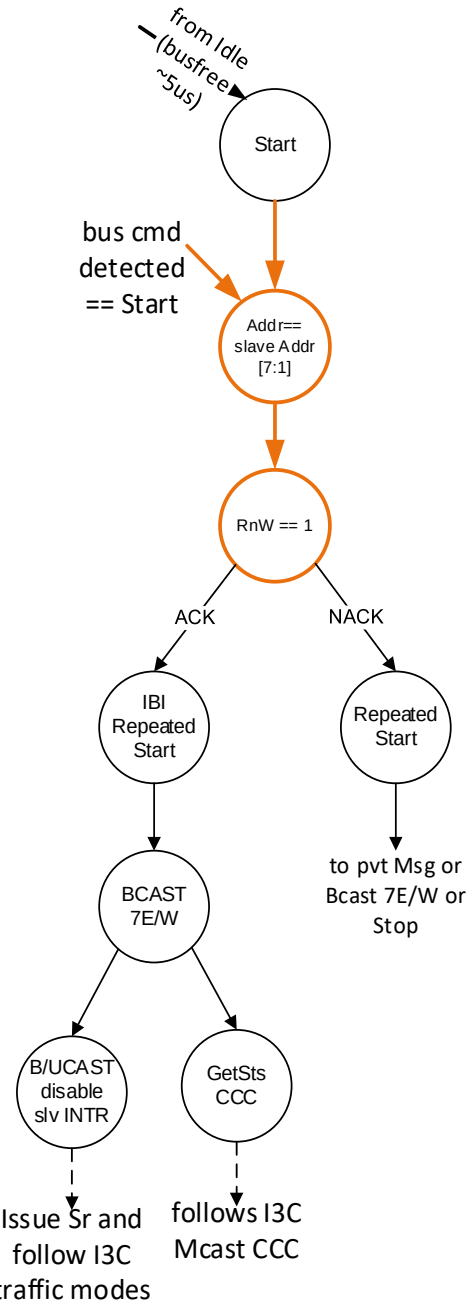


Figure 58 Slave Interrupt Request FSM

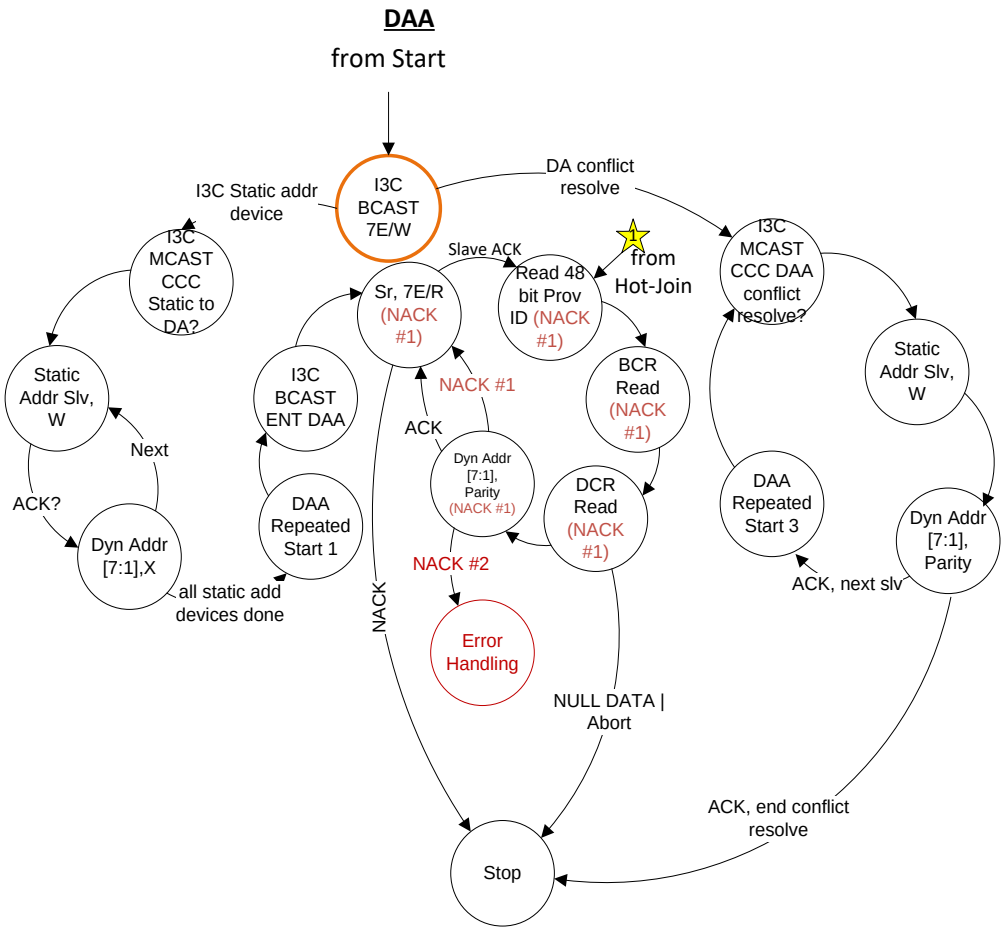


Figure 59 Dynamic Address Assignment FSM

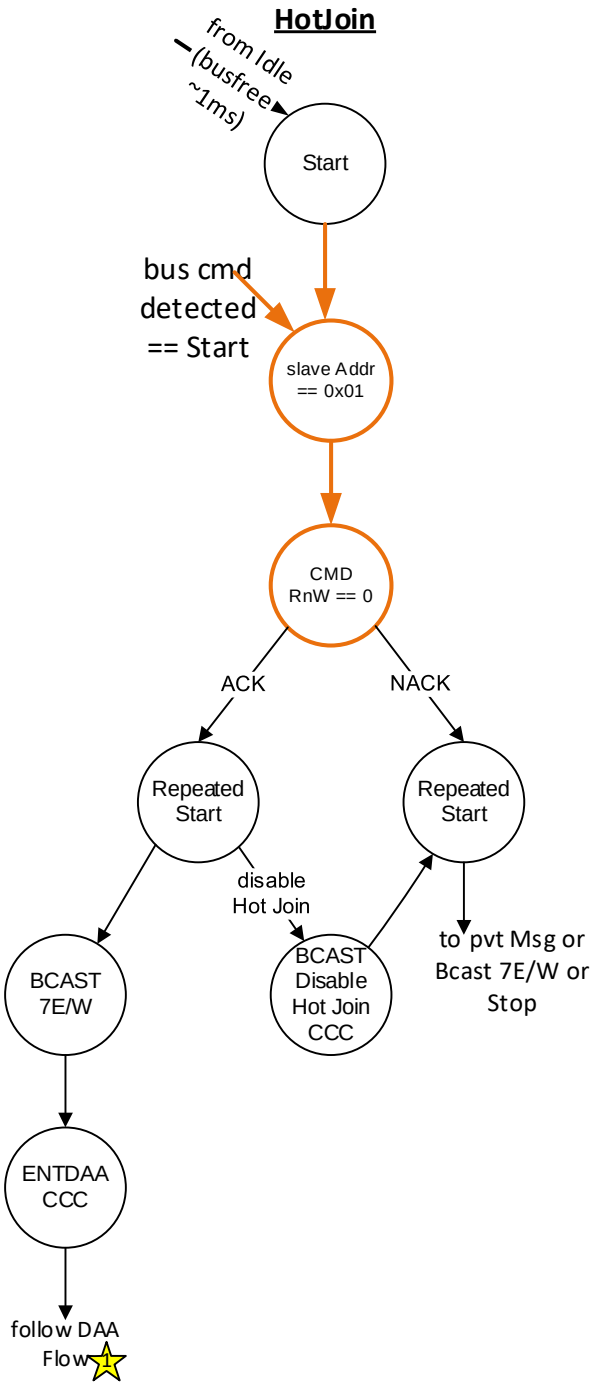


Figure 60 Hot-Join FSM

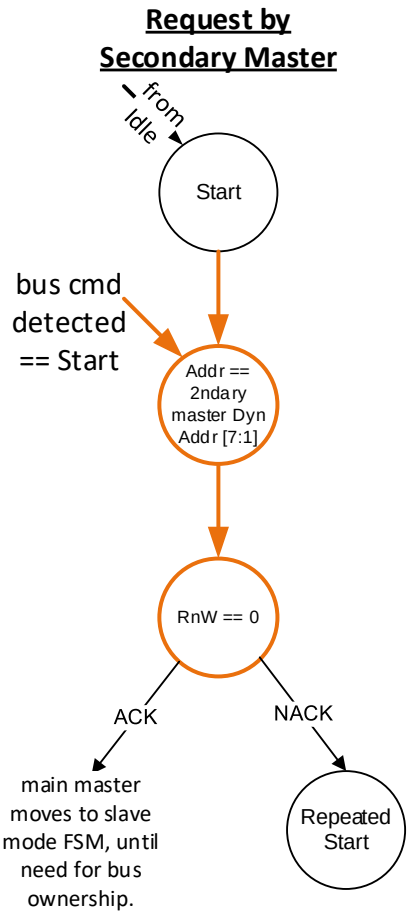


Figure 61 Secondary Master Request FSM

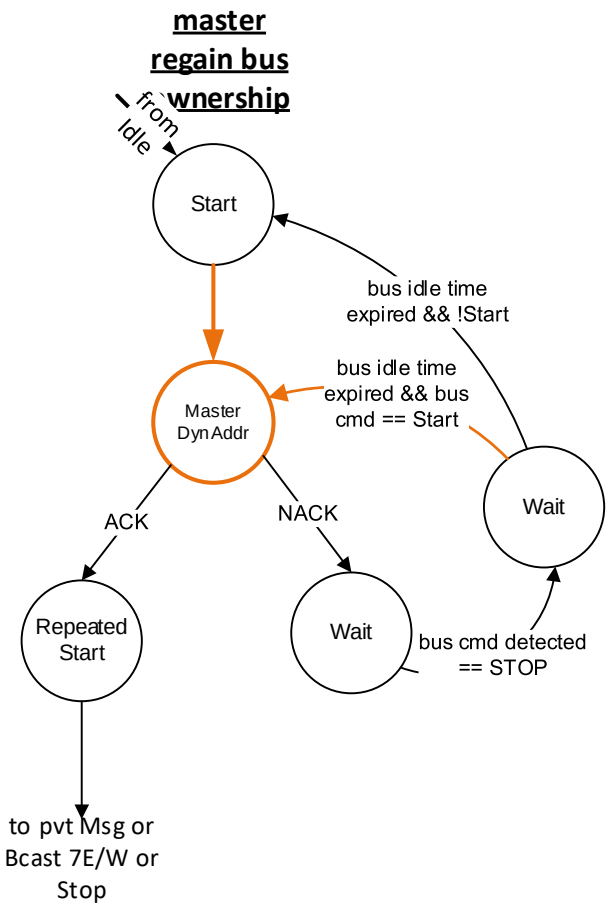
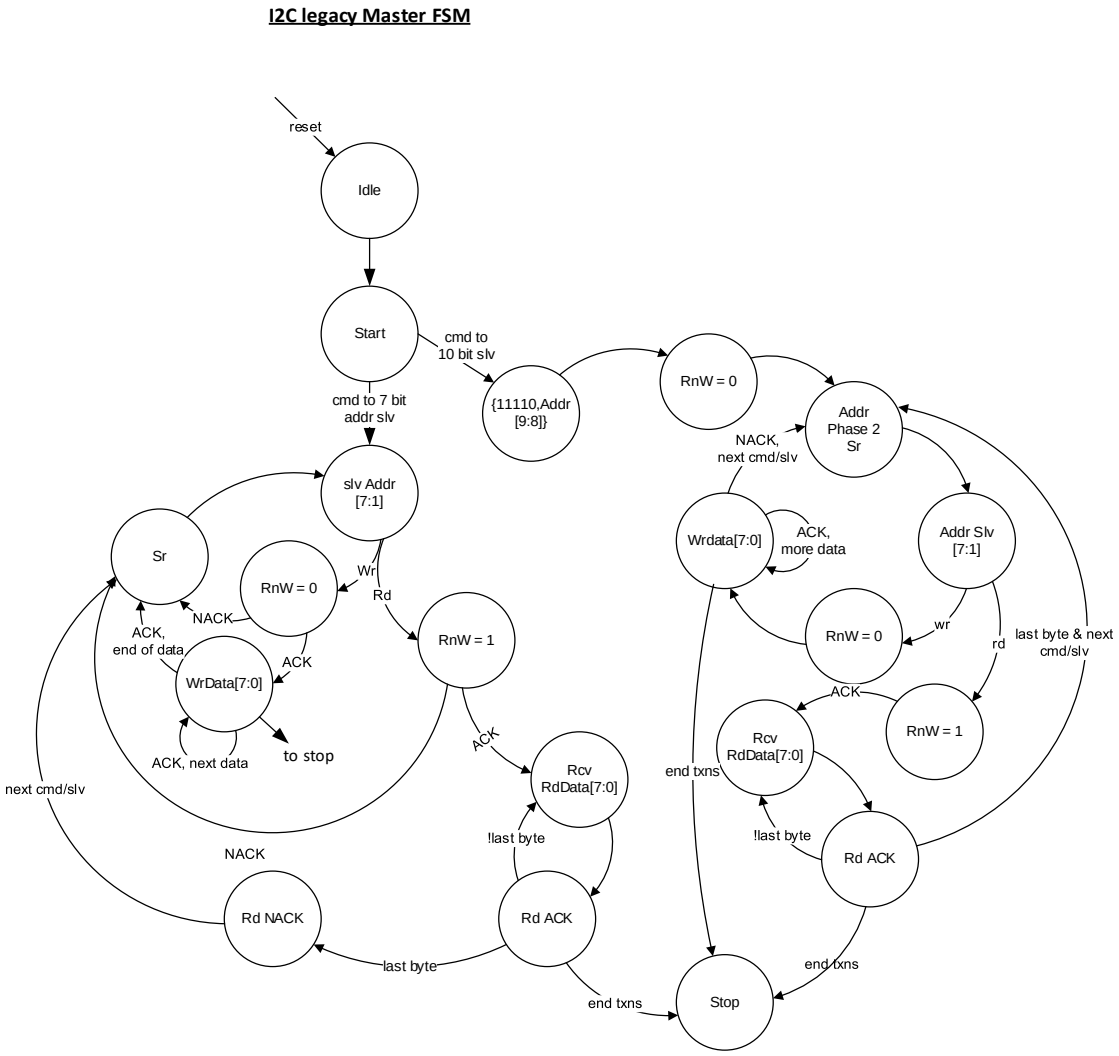


Figure 62 Master Regaining Bus Ownership FSM

2349 ***Figure 63*** is for reference only.



2350

2351 **Figure 63 I²C Legacy Master FSM**

This page intentionally left blank.

Annex D Typical I3C Basic Protocol Communications

Figure 64 through ***Figure 66*** each illustrate a typical communication for the I3C SDR Protocol. While these diagrams do not exhaustively illustrate all possible I3C SDR communications, they do serve as useful introductions to the signaling and transmission formatting used in the I3C SDR Protocol.

Figure 64 illustrates example communication using I3C Single Data Rate (SDR) coding. It shows the Master reading a byte of data from the Slave at Address 0x2B in SDR Mode.

From the Bus Free Condition, the Master issues a START by driving the SDA line Low while keeping the SCL line High. It then issues the Broadcast Address (7'h7E) followed by RnW (0 for Write). Then the Master turns on a pull-up resistor and goes to Open Drain. All Slaves ACK by pulling the SDA line Low (in the Figure, pink fill means the Slave is in control of the SDA line at this time). The Master then issues a Repeated START, then the Address of the Slave (0x2B) it wants to read followed by RnW (1 for Read). The Master then turns on a pull-up resistor and goes to Open Drain, allowing the Slave to acknowledge by pulling the SDA line Low. At this point, the Master continues to toggle the SCL line and release the SDA line, allowing the Slave to drive SDA to send one byte of data (0x4A) followed by 'T'. T=1 informs the Master that there is additional data, whereas T=0 signals the end. Here there is additional data, so the Slave drives SDA High until SCL goes High, at which time it releases SDA. The Master has the option of holding SDA High with a weak pull-up, which signals to the Slave that the Master allows another byte to be transmitted, or to pull SDA Low (while SCL is High – hence a Repeated START), which would signal to the Slave that the Master has terminated the Read and is taking over.

SDR Mode is backwards compatible with Legacy I^2C Devices, because the High time of an SCL pulse is always less than 50ns and therefore SCL will always appear to be Low because of the I^2C 50ns Spike Filter.

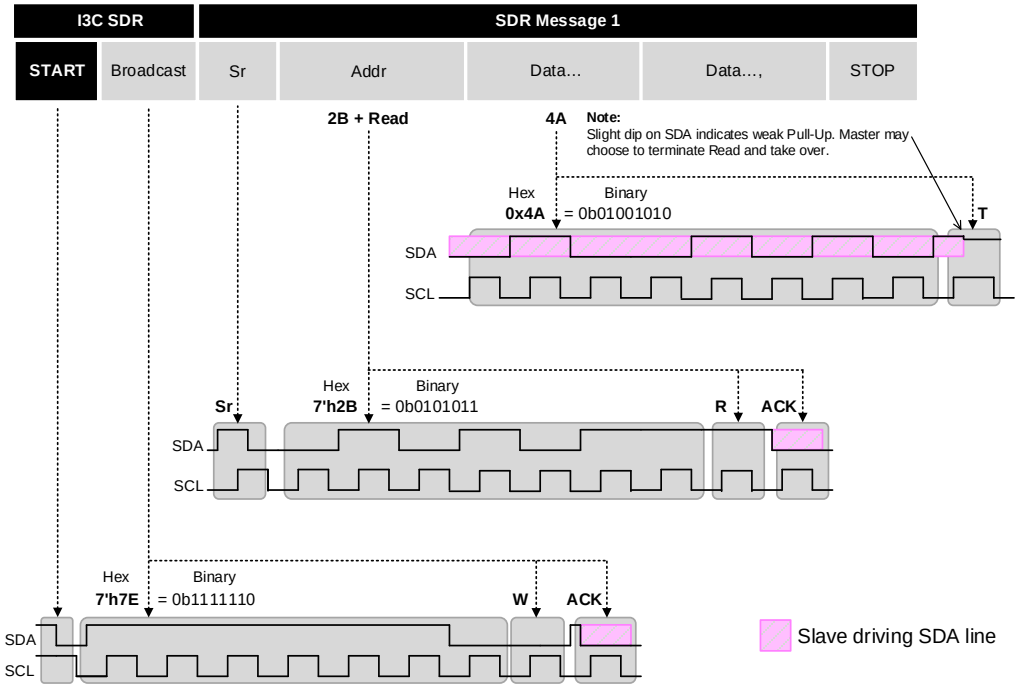


Figure 64 Example Communication Using I3C Coding SDR

Figure 65 shows the Master issuing a CCC Direct Command to a single Slave. This particular command (GETPID) reads the Provisional ID of a Slave.

From the Bus Free Condition, the Master issues a START by driving the SDA line Low while keeping the SCL line High. It then issues the Broadcast Address (7'h7E) followed by RnW (0 for Write). Then the Master turns on a pull-up resistor and goes to Open Drain. All Slaves ACK by pulling SDA Low (in the Figure, pink fill means the Slaves are in control of SDA at this time). The Master then issues the Direct Common Command Code for GETPID (0x8C) followed by parity bit 'T' (odd parity = 0 for 0x8C) then the 7-bit Dynamic Address of the Slave (chosen arbitrarily here to be 0x2B) followed by a RnW bit (1 for Read). Then the Master turns on a pull-up resistor and goes to Open Drain, allowing the Slave at Address 0x2B to ACK by pulling SDA Low, which tells the Master that the Slave Acknowledges the command and will comply. (Alternatively, the Slave may NACK by not pulling SDA Low, which would inform the Master that the Slave will not comply – in this case, that an error occurred.) Following the ACK the Slave outputs its 48-bit PID one byte at a time, and then the Master issues a Repeated START (this part of the waveform sequence is not shown in the Figure).

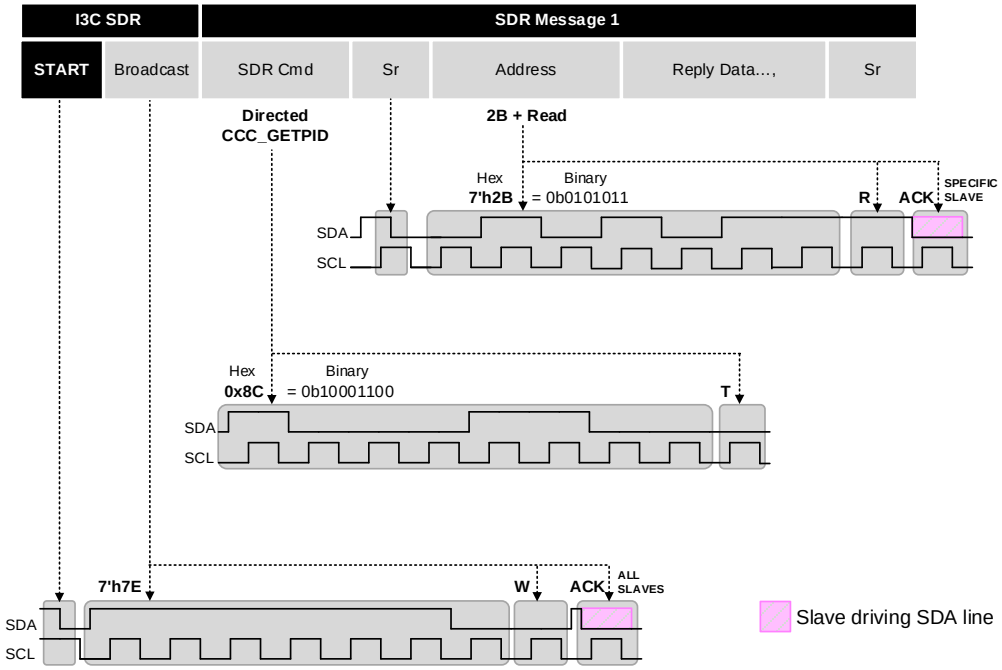
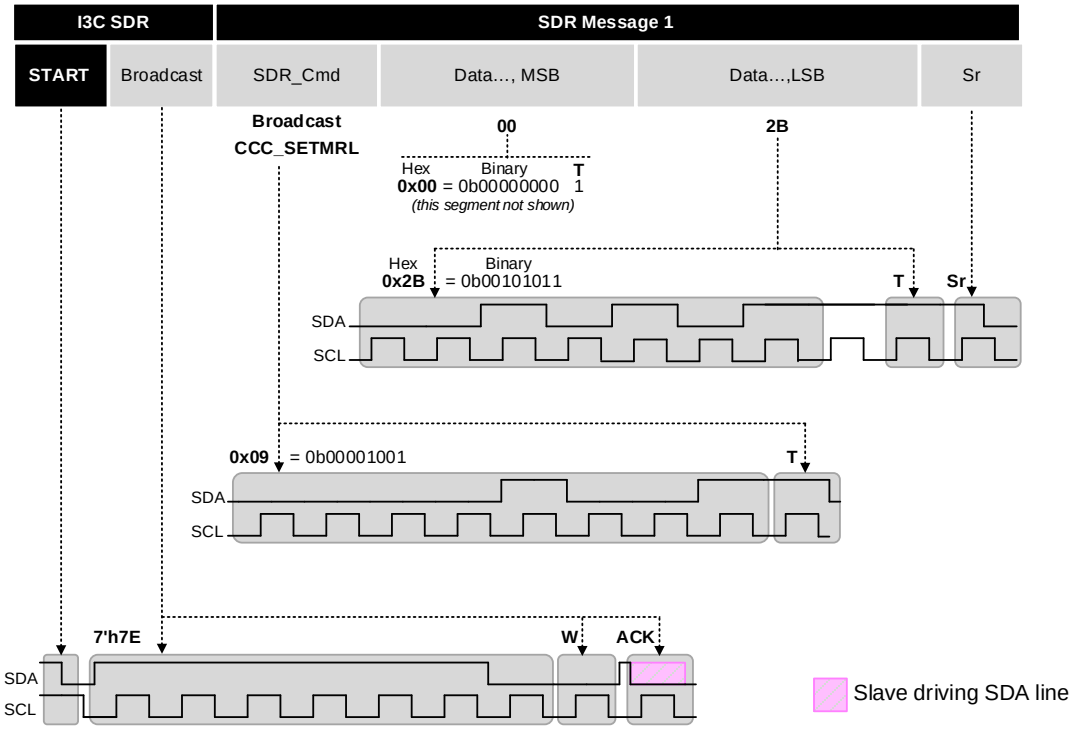


Figure 65 Example Communication Using I3C Coding SDR with CCC Direct Addressing

Figure 66 illustrates example SDR communication with a CCC Broadcast command. The command used in this example sets the Maximum Read Length of all Slaves to 43 bytes (0x002B).

From the Bus Free Condition, the Master issues a START by driving the SDA line Low while keeping the SCL line High. It then issues the Broadcast Address (7'h7E) followed by RnW (0 for Write). Then the Master turns on a pull-up resistor and goes to Open Drain. All Slaves ACK by pulling SDA Low (in the Figure, pink fill means the Slaves are in control of SDA at this time). The Master then issues the Broadcast Common Command Code for SETMRL (0x09) followed by parity bit 'T' (odd parity = 1 for 0x09), and then 2 data bytes (MSb first) to define the maximum number of bytes which can be read from a Slave in a single read operation. Each data byte is followed by a 'T' bit (parity bit – odd parity). After this the Master issues a Repeated START.



2356

2357 **Figure 66 Example Communication Using I3C Coding SDR with CCC Broadcast**

This page intentionally left blank.

Annex E MIPI I3C Basic Specification Supplemental Patent Licensing Terms

This agreement (the "Agreement") between MIPI Alliance Inc. ("MIPI") and each MIPI member or other party that has manifest agreement to these terms (each a "Licensor" and collectively the "Licensors") is effective as of the date the I3C Basic Specification (defined below) is first approved by the MIPI Board (the "Effective Date"). Capitalized terms used in this Agreement that are not expressly defined here have the meaning identified in the MIPI Membership Agreement or MIPI Bylaws, as applicable. For convenience, key definitions are reproduced in Attachment A. For the avoidance of doubt: (a) in connection with this Agreement, any reference to a "MIPI Specification" means the MIPI I3C Basic Specification as described in Section 2, and (b) any rights or obligations created under this Agreement are independent of any rights or obligations created under the MIPI Membership Agreement, Bylaws or any other agreements, and nothing in this Agreement is intended to alter rights or obligations established elsewhere.

1. Background. Typically, MIPI specifications are implemented only by MIPI members. MIPI members make certain intellectual property licensing commitments to other members under the MIPI Membership Agreement, with different rules applying to "Mobile Terminals" and "Accessories," in contrast to other types of implementations. The MIPI Board intends to make the MIPI I3C Basic Specification available for implementation by parties who are non-members of MIPI, however. Further, both MIPI members and non-members may use this specification inside and outside of Mobile Terminals or Accessories. The Licensors contributed to the development of the MIPI I3C Basic Specification, and desire to see it widely used. MIPI and the Licensors believe that making licenses available (as set forth in this Agreement) to both member and non-member implementers, for all types of implementations, will facilitate widespread adoption of the MIPI I3C Basic Specification, to the benefit of MIPI, the Licensors, and the broader community that MIPI serves.

2. MIPI I3C Basic Specification. "MIPI I3C Basic Specification" means the specification titled "I3C Basic 1.0" as approved by the MIPI Board, and all subsequent versions of such specification approved by the MIPI Board after the Effective Date. Any party implementing the MIPI I3C Basic Specification, whether or not a MIPI member, is an "Implementer." For the avoidance of doubt: the MIPI I3C Basic Specification is distinct from the MIPI I3C Specification version 1.0 approved by the MIPI Board on Dec. 31, 2016. The terms of this Agreement apply exclusively to the MIPI I3C Basic Specification.

3. License commitment.

a. RAND-Z license obligation. For the MIPI I3C Basic Specification only, Licensor hereby agrees to grant, and to cause its Affiliates to grant, to any requesting Implementer a worldwide, non-exclusive, non-sublicensable license under the Necessary Claims of Licensor or its Affiliates, with zero royalties or other compensation, under terms and conditions that are reasonable and nondiscriminatory, to make, have made, use, import, offer to sell, lease, sell, promote and otherwise distribute Compliant Portions. Licensor shall not be obligated to license any part or function of a product in which a Compliant Portion is incorporated that is not itself a Compliant Portion.

b. Reciprocity; defensive suspension. Licensor shall not be obligated to license any Implementer if that Implementer does not agree to make patent licenses available under any Necessary Claims of that Implementer and its Affiliates to Licensors and all other Implementers under terms substantially identical to the terms described in this Agreement. Further, a Licensor may suspend any license granted under this Agreement to any Implementer if that Implementer or its Affiliate initiates against any party litigation that alleges infringement of a Necessary Claim of Implementer or its Affiliate in connection with the MIPI I3C Basic Specification. Additionally, subject to Section 3.1(f) of the MIPI Membership Agreement as between MIPI Members, a Licensor may terminate, ab initio, any license granted pursuant to this Agreement to any Implementer that initiates litigation against the Licensor alleging infringement of any patent claim of the Implementer or its Affiliate. For the purposes of this Agreement, a party that files a suit which is

defensive based on a patent infringement claim or suit by another party will not be deemed to have initiated litigation.

c. Circumvention or transfer. Licensor agrees that it has not transferred and will not transfer any patent having Necessary Claims solely for the purpose of circumventing the obligations described in this Agreement. In addition, Licensor agrees that any transfers by Licensor to a third party of a patent having relevant Necessary Claims, whether or not recognized as Necessary Claims at the time of transfer, shall be subject to (i) the terms and conditions of this Agreement, and (ii) the agreement that the third party shall grant licenses under said Necessary Claims to Implementers pursuant to the terms of this Agreement in like manner and to the same extent as the third party would be required to do if it had executed this Agreement. A transfer of ownership in a business entity which owns or has the right to license a patent having Necessary Claims shall be considered a transfer of such patent.

4. Termination of obligation to license future versions. Each Licensor may terminate its participation in this Agreement at any time by providing written notice to the MIPI Managing Director; termination will be effective 30 days after such notice is actually received, subject to the survival points below. Promptly upon receipt of such notice, the MIPI Managing Director will alert the MIPI Board and will use reasonable efforts to notify all Licensors of such termination. After termination, the agreement to grant a license as provided in Section 3 shall survive in full force and effect only: (a) for versions of the MIPI I3C Basic Specification which the Board had approved before the effective date of termination; (b) for Necessary Claims relating to any version of the MIPI I3C Basic Specification approved after the effective date of termination that are used in a substantially similar manner and to a substantially similar extent with a substantially similar result as the Necessary Claims that were used in a prior version for which the Licensor is obligated to grant licenses under this Agreement; and (c) for those Necessary Claims that directly result from inclusion of Licensor-provided material in the draft version of the specification that existed immediately prior to Licensor's withdrawal. Termination of this Agreement by one Licensor does not impact the Agreement among MIPI or the other Licensors, not does it not impact Licensor's MIPI Membership Agreement, which will remain in full force and effect.

5. Third party beneficiaries. All Implementers, whether or not they are MIPI members, are intended third party beneficiaries of this Agreement.

6. Counterparts; additional Licensors. This Agreement may be signed in any number of counterparts. If additional parties manifest agreement to terms substantially identical to this Agreement, those parties will be deemed Licensors under this Agreement.

Attachment A

1.3. **"Compliant Portions"** means only those specific portions of products (hardware, software or combinations thereof) that: (i) both implement and are compliant with the relevant portions of the MIPI Specification, (ii) are qualified pursuant to the MIPI qualification process (if available), (iii) meet the requirements set forth in any compliance requirements set forth by the Corporation, applied to all Members on a nondiscriminatory basis, and (iv) are within the bounds of the Scope of IPR (defined below).

1.5 **"Interface"** means the protocols, signaling characteristics, commands, clocking signals, register models, application program interfaces and data structures to the extent they enable interoperation, interconnection or communication between integrated circuits (even if located on the same die).

1.7. **"Necessary Claims"** mean those claims of all patents and patent applications, other than design patents and design registrations, throughout the world which (i) a Member or its Affiliates has the right, at any time during the term of this Agreement, to grant licenses of the nature granted or agreed to be granted herein without such grant resulting in payment of royalties or other consideration to third parties (except for payments to Affiliates or to employees within the scope of their employment); (ii) are within the Scope of IPR; and (iii) are necessarily infringed by an implementation of a MIPI Specification, wherein such infringement could not have been avoided by another commercially reasonable non-infringing implementation of such MIPI Specification. Necessary Claims do not include (i) any claims other than those set forth above even if contained in the same patent as Necessary Claims, or (ii) any claims that read on any implementation of the Specification to the extent that such implementation is not within the bounds of the MIPI Specification.

1.8 **"Scope of IPR"** means Interfaces, solely to the extent disclosed with particularity in a MIPI Specification, where the purpose and sole licensed (under this agreement) use of such disclosure is to define, implement, and utilize an interface that enables interoperation, interconnection or communication in accordance with a MIPI Specification. Notwithstanding the foregoing, the Scope of IPR shall not include (i) any enabling technologies that may be necessary to make or use any product or portion thereof that complies with a MIPI Specification, but are not themselves expressly set forth in a MIPI Specification; (ii) semiconductor manufacturing technology, DSP architecture, processor architecture/microarchitecture, wireless communication technology, compiler technology, integrated circuit packaging technology, security technology, internal architectures of integrated circuits, applications which run on integrated circuits, audio coding technology, video coding technology or basic operating system technology; (iii) SDO Standards, whether in whole or significant part, not developed by or for the Corporation, but referred to or incorporated in a MIPI Specification, or (iv) any portions of any product and any combination except for that portion or portions which are required solely in order to achieve an interface that is compliant with a MIPI specification; (v) any methods or processes practiced, in whole or in part, over an Interface that are not expressly set forth in a MIPI Specification.

"Affiliate," means any corporation, partnership, or other entity that, directly or indirectly, owns, is owned by, or is under common ownership with, such Member hereto, for so long as such ownership exists. For the purposes of the foregoing, "own," "owned," or "ownership" shall mean ownership of more than fifty (50%) of the stock or other equity interests entitled to vote for the election of directors or an equivalent governing body of an entity that is directly or indirectly controlled by, under common control with or that controls the subject party.

"Board" means the Board of Directors of MIPI.

This page intentionally left blank.

Annex F I3C Basic Development Companies

Analog Devices, Inc.

Analogix Semiconductor, Inc.

Avery Design Systems, Inc.

Cadence Design Systems, Inc.

Intel Corporation

Introspect Test Technology Inc.

InvenSense, Inc.

L&T Technology Services

Lattice Semiconductor Corp.

MediaTek Inc.

NXP Semiconductors

Qualcomm Incorporated (provided notice of termination effective August 13, 2018)

Robert Bosch GmbH

SmartDV Technologies India Private Limited

STMicroelectronics

Synaptics

Synopsys, Inc.

Toshiba Memory Corporation

Valens Semiconductor

This page intentionally left blank.

Participants

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

Eugen Becker, Robert Bosch GmbH

Rajesh Bhaskar, Intel Corporation

Enrico Carrieri, Intel Corporation

Geraud Cheenne, STMicroelectronics

Ladvine D Almeida, Synopsys, Inc.

Kenneth Foust, Intel Corporation

Chris Grigg, MIPI Alliance

Paul Kimelman, NXP Semiconductors

Abinaya Kubendran, L&T Technology Services

Peter Lefkin, MIPI Alliance

Satwant Singh, Lattice Semiconductor Corp.

Przemyslaw Sroka, Cadence Design Systems, Inc.

Greg Stewart, Analogix Semiconductor, Inc.

Eyuel Zewdu Teferi, STMicroelectronics

Suresh Venkatachalam, Synopsys, Inc.

James Wang, InvenSense, Inc.

Miles Williams, Introspect Test Technology Inc.

This page intentionally left blank.