System Integrators Application Note for MIPI I3C® v1.0 and I3C Basic℠ v1.0

App Note Version 1.8
8 December 2018

MIPI Board Approved 8 December 2018

Public Release Edition

NOTICE OF DISCLAIMER

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

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

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

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

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

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

Special Note Concerning I3C and I3C Basic

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

Contents

Figures

Release History

Date	Version	Description
2018-12-08	v1.0	Initial Board approved release.

This page intentionally left blank.

1 Introduction

The MIPI I3C Bus interface ***[MIPI03]*** is an evolutionary specification that improves upon the legacy I²C standard. It is designed to reduce the number of physical pins used in sensor system integration, and supports low-power, high-speed digital communication typically associated with UART and SPI interfaces.

I3C's main features include:

- In-Band Interrupts
- Dynamic Addressing
- Multi-Master and Multi-drop capabilities
- Hot-Join support
- Backward compatibility with I²C

The I3C interface is expected to play a fundamental role in streamlining sensor integration in smartphones, wearables, and Internet-of-Things (IoT) devices. This Application Note is intended to help users understand how the I3C interface works.

1.1 Scope

This System Integrator Application Note is intended to guide three different groups:

- Those developing MIPI I3C Masters and Slaves to understand how their parts will fit into different types of systems, and the considerations for functionality and features.
- System Designers who have to wire together different MIPI I3C Devices, and potentially I²C Devices, and who need to know considerations of trace layout and connections, voltage regulation for the Devices, any strapping or other ID factors, etc.
- MIPI I3C Master software considerations in the one or more Masters in the system. This includes users of both standardized Host Controller APIs and MCU/DSP firmware.

This Application Note has several parts, each focusing on a different area and covering both required considerations and optional ones, based on which features and topology are used in a given system. This approach makes it easier for any of the targeted groups to focus on what matters to them based on what configurations they will be working with.

This App Note is intended to be used together with the I3C Specification ***[MIPI03]***. Each Application Note section corresponds to one or more Specification sections, primarily focusing on Specification ***Section 6, I3C Electrical Specifications***. The App Note sections amplify the Specification with additional context (e.g., analysis data to back up recommended use models) and details (e.g., total Bus capacitance vs. per-Device capacitance allowances) that would not be appropriate in a protocol specification.

2 Terminology

See also ***Section 2*** in the MIPI I3C Specification ***[MIPI03]***.

2.1 Definitions

System Designer: Engineer designing a system that includes an I3C Bus.

2.2 Abbreviations

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

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

2.3 Acronyms

I3C MIPI Improved Inter Integrated Circuit interface or its Specification document ***[MIPI03]***

3 References

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

[MIPI02] *MIPI Alliance Specification for I3C Basic℠ (Improved Inter Integrated Circuit – Basic)*, version 1.0, MIPI Alliance, Inc., 19 July 2018 (Adopted 8 October 2018).

[MIPI03] Either ***[MIPI01]*** or ***[MIPI02]***.

[MIPI04] MIPI Alliance, Inc., “Current I3C Device Characteristic Register (DCR) Assignments”, https://www.mipi.org/MIPI_I3C_device_characteristics_register, last accessed 8 December 2018.

4 Overview

This Application Note describes two broad categories of information:

- Details of a few typical and supported topologies, including different trace distances. This covers the allowances and challenges presented by each topology.
- Details of added functionality and allowances, including Hot-Join and Hot-insertion, Timing control (time-stamping) considerations, power states of Devices and the implications for the Bus, etc.

4.1.1 Supported Topologies

It is envisioned that the different groups developing aspects of such I3C-based systems will want to consult the relevant App Note sections that will help avoid mistakes, and provide guidance not found in the I3C Specification ***[MIPI03]***.

Analysis data is provided to help understand the considerations that impact placement and mixed Devices uses, and can be extrapolated to other configurations. This analysis data could be used, for example, when choosing what types of Devices to mix on one Bus with a given topology (vs. split into two Buses, etc.), when deciding how best to manage relative Device locations (floor-planning), and when gauging system reliability.

4.1.2 Using Added Functionality

The I3C Specification ***[MIPI03]*** includes optional extended functionality and the flexibility to allow for different use cases. Some of these use cases have implications for the System Designer and for software on the Master.

The I3C Specification is focused on how each feature works from a protocol perspective; by contrast, this Application Note provides additional information on how that feature can be incorporated into a system. This additional information guides both assessments of whether it will be possible to accomplish what is wanted, as well as how to accomplish it.

Examples:

- Understanding how the Hot-Join feature can be used both in systems that physically connect a separate module, and in systems that simply power-off portions of the Bus.
- Understanding how different timing control strategies can be used, and what must be looked for in terms of the various Devices in the system that have to be compatible.

5 System Integration Guidelines

This section discusses key guidelines that system integrators will want to follow in order to implement an optimized I3C Bus in both Pure Bus (I3C Devices only) and Mixed Bus (I3C and I^2C Devices) configurations. These guidelines focus on:

- I3C Device Characteristics
- Legacy I^2C Device Characteristics
- Dynamic Address Assignment (DAA)
- Bus Topologies
- Physical/Electrical/Testing Considerations (CTS)
- Bus High Keeper
- Bridge Devices

5.1 I3C Device Characteristics

The configuration of a given 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 will fulfill all responsibilities for that Role, as detailed in ***Figure 1*** (***Table 3*** from the I3C Specification ***[MIPI03]***).

The defined Roles are:

- Main Master
- Secondary Master
- SDR Only Main Master
- SDR Only Secondary Master
- Slave
- SDR Only Slave

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 Slave capable	Supports being accessed in at least one HDR Mode	Y	Y	N	N	Y	N
HDR Master capable	Supports Mastering in at least one HDR Mode	Y	Y	N	N	N/A	N/A
HDR Exit Pattern Generation capable[2]	Able to generate the HDR Exit Pattern on the Bus for error recovery	Y	Y	Y	Y	N	N
HDR Tolerant	Recognizes HDR Exit Pattern	Y	Y	Y	Y	Y	Y

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

Figure 1 I3C Device Roles vs Responsibilities

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 will not have a 50 ns filter enabled when used in an I3C Bus operating at full clock speed. These differences are summarized in ***Figure 2*** (***Table 4*** in the I3C Specification). When used in an I3C system, I3C Slaves enable or disable appropriate I²C features as shown in ***Figure 2***.

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 71** and **Table 73***
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.*

Figure 2 I²C Features Allowed in I3C Slaves

5.1.1 Clock-to-Data Turnaround Time (t_{SCO})

This section discusses the impact of Clock-to-Data turnaround (t_{SCO}) on a single-ended, single-clock Bus such as I3C when operating the interface over long traces (> 0.5 m). ***Figure 3*** (***Figure 23*** in the I3C Basic Specification ***[MIPI02]***) illustrates this scenario.

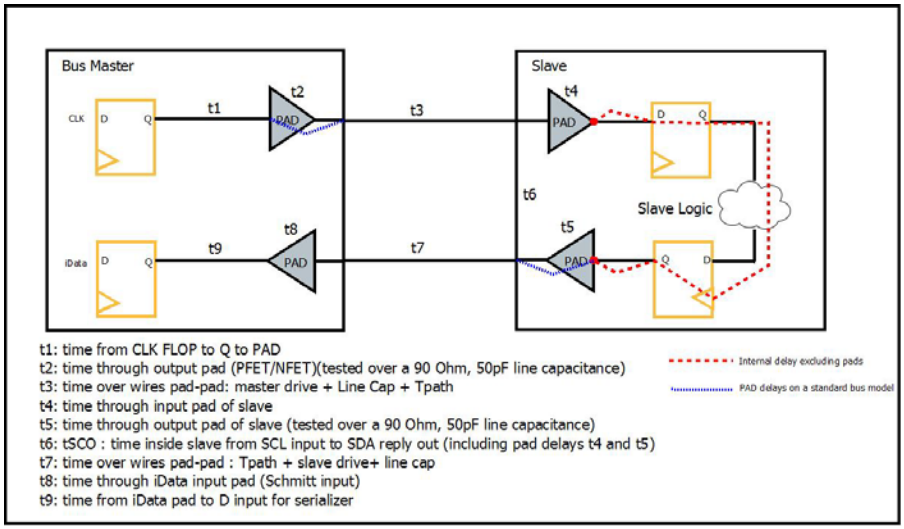

Figure 3 Factors Contributing to Latency

5.1.2 Pad Capacitance

The pad capacitance adds to the capacitance of the whole Bus, and must be considered when computing the effective Bus frequency or other Bus parameters. A Device's pad capacitance also contributes directly to the skew of the signals that the Device drives on the Bus, to the Device internal delay, and to the t_{SCO}. For this reason, t_{SCO} will be characterized on a standard 50 pf, 90 Ω and 4 mA driver.

5.1.3 Pad Drive Strength

The pad drive strength is another factor contributing to signal skew, and to the Device's ability to drive the pad within the required rise and fall time. The minimum recommended Bus drive strength is 4 mA drive. Greater drive strength can be implemented, as long as the Bus reflections and the power requirements of the system design are not affected. Most Bus timing parameters in the I3C Specification were determined using 4 mA drive strength as the measurement reference. When choosing greater drive strength, care must be taken to avoid overshoot.

5.1.4 BCR Use

Each I3C Device that is connected to the I3C Bus has an associated read-only Bus Characteristics Register (BCR). Per ***Figure 4*** (***Table 6*** from the I3C Specification), the BCR describes the I3C-compliant Device's Role and its capabilities as regards Dynamic Address assignment and Common Command Codes (CCCs).

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 Sensor WG 2'b11 – Reserved for future definition by MIPI Sensor WG
BCR [6]	**Device Role [0]**	
BCR [5]	**SDR Only / SDR and HDR Capable**	0 – SDR only 1 – HDR Capable
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 5.1.2.3.3***
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 this MIPI Specification for I3C.*
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.*

Figure 4 Bus Characteristics Register (BCR)

The Max Data Speed Limitation bit (BCR bit [0]) is set when the Device has one or more of the following limitations:

- t_{SCO} is larger than 12 ns
- Overall internal Device delays including Pads is larger than 12 ns
- The Device requires a longer time to prepare the requested data
- The Device is not designed to operate at the maximum operating frequency (12.5 MHz)

This Max Data Speed Limitation bit notifies the Master that the Device has some limitations, and the Master will have to send a GETMXDS command at a lower speed that is acceptable by all Devices to determine the particular limitation(s), and then take the appropriate measure(s) necessary to accommodate the Device.

5.2 Legacy I²C Device Characteristics

Performance of an I3C Bus is heavily dependent upon any I²C-only Devices that might 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 ***Figure 5*** (***Table 5*** in the I3C Specification ***[MIPI03]***).

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})*

Figure 5 Legacy I²C-Only Slave Categories and Characteristics

Furthermore, per ***Figure 6*** (***Table 72*** in the I3C Specification ***[MIPI01]***, and ***Table 56*** in the I3C Basic Specification ***[MIPI02]***), no I²C or I3C Device present on an I3C Bus will have a fixed I²C Address that matches any of the Addresses associated with Error Type S0 per ***Figure 7*** (***Table 59*** in the I3C Specification ***[MIPI01]***, and ***Table 52*** in the I3C Basic Specification ***[MIPI02]***).

Feature	Required	Desirable	Not Used	Not Allowed	Notes
Fm Speed	X	–	–	–	–
Fm+ Speed	–	X	–	–	–
HS and UFm	–	–	X	–	2
Static I²C Address	X	–	–	–	–
50ns Spike Filter	–	X	–	–	1
Clock Stretching by Slave	–	–	–	X	–
I²C Extended Address (10 bit)	–	–	X	–	2
I3C Reserved Address	–	–	–	X	–

Note:
1) *Lack of Spike Filter will severely degrade Bus performance and eliminate certain I3C Bus features*
2) *"Not Used" means that the I3C Master will not make use of the I²C feature, however if the Slave supports the feature, then it will not interfere with I3C Bus operation.*

Figure 6 Legacy I²C Device Requirements When Operating on I3C

Error Type	Description	Error Detection Method	Error Recovery Method
S0	**Invalid Broadcast Address/W** (7'h7E/W) or **Dynamic Address/RnW after DA assignment**	Detect any of the following: 7'h3E / W 7'h5E / W 7'h6E / W 7'h76 / W 7'h7A / W 7'h7C / W 7'h7F / W 7'h7E / R	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 Provisioned 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'h7E / R" shall be excluded from the S0 error definition.*			

Figure 7 SDR Slave Error Types

5.2.1 Pad Capacitance

As previously stated, the pad capacitance adds to the whole Bus capacitance and needs to be considered when computing the effective Bus frequency or other Bus parameters. The pad capacitance of a Device also contributes directly to the skew of the signals that the Device drives on the Bus, the Device internal delay, and the t_{SCO}. For this reason, it is important that Device vendors accurately specify the maximum pad capacitance such that System Designers can optimize I3C Bus performance for the total capacitance seen on the SDA and SCL lines.

5.2.2 Spike Filter

For an I²C Device on an I3C Bus, it is mandatory to have a 50 ns spike filter on both the SCL pad and the SDA pad. This is done because important I3C-specific communications are done at a higher frequency (12.5 MHz), and I²C Devices without such spike filters might be liable to misinterpret these communication frames, or to misbehave on the Bus. Having the spike filter on both pads allows the I²C Devices to filter the I3C communications and still remain responsive to lower-speed communications intended for the I²C Devices (or all Devices) on the Bus.

5.2.2.1 Detecting Presence of 50 ns Spike Filter

If it is not known whether a given I²C Device has a 50 ns spike filter or not, then this can be tested by sending a private read/write communication with the broadcast address transmitted first.

This pattern is shown in ***Figure 8***. The signaling marked in grey indicates Open Drain communication at lower frequency. Communication marked in blue indicate a high-speed communication where the clock High width (t_{DIG_High}) is less than 50 ns, or an I3C Mixed Bus timing is maintained.

START	7-bit Broadcast Address	RnW=0	ACK from DUT	
Repeated START	7-bit I²C Address	RnW=0	NACK from DUT	STOP

Figure 8 Spike Filter Detection Pattern

Based on the response for the I²C address, the Master can determine whether the Device has a 50 ns spike filter or not:

- If the Device responds with ACK, then the Device does not have a 50 ns spike filter, since it can detect the high-speed communications on the Bus.
- If the Master receives NACK, then the Device does have a 50 ns spike filter, since it did not properly process the high-speed communication on the Bus.

Note that the I²C Device does not respond to the broadcast address, and the Bus might be have a combination of I²C and I3C Devices.

This is also applicable for I²C/I3C capable Devices which are supposed to respond to I²C commands directed to them using their I²C static address. If no Dynamic Address is assigned to these Devices, then the Devices will have the 50 ns spike filter active and behave as I²C Devices. If the Master assigns a Dynamic Address, or upon the detection of the broadcast address, such Devices could disable their spike filters and act as I3C Devices.

5.2.3 SDA Drive Strength

As previously discussed, the pad drive strength is another factor that contributes to the signal skew and to a Device's ability to drive the pad within the required rise and fall time. Notably, I²C (Fm/Fm+) and I3C can have different specified drive strengths, thus care should be taken to minimize any impact to Bus performance due to reflections, overshoot, etc. The minimum recommended Bus drive strength is 4 mA drive. A greater drive strength can be implemented, as long as the Bus reflections and the power requirements of the system design are not affected. Most Bus timing parameters in the I3C Specification were developed using the 4 mA drive strength as the measurement reference. When choosing greater drive strengths, care must be taken to avoid overshoot.

5.2.4 I²C Clock Stretch is Not Allowed in I3C

Ideally, the I3C Bus Master will drive the clock (SCL) in a Push-Pull manner to optimize the Bus for speed, efficiency, and loading. Given this, and the fact that I²C drives the SCL line in an Open Drain manner, I3C does not allow I²C clock stretching (i.e., where the I²C Slave has permission to hold the SCL line Low in order to delay the Bus).

5.2.5 Legacy Virtual Register Use

Each Legacy I²C Device that can be connected to the I3C Bus has an associated read-only Legacy Virtual Register (LVR) describing the Device's significant features. Since these are Legacy I²C Devices, it is understood that this register will exist not in on-Device hardware, but rather virtually (for example, as part of the Device's driver). When Legacy I²C Devices are present on an I3C Bus, LVR data determines allowed Modes and maximum SCL clock frequency. LVR bits conform to ***Figure 9*** (***Table 8*** in the I3C Specification).

Bit	Name	Description
LVR [7]	**Legacy I²C 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 I²C only [1]** (Indexed in ***Table 5***)	
LVR [5]	**Legacy I²C only [0]** (Indexed in ***Table 5***)	
LVR [4]	**I²C Mode Indicator**	0 – I2C Fm+ 1 – I2C Fm
LVR [3]	MIPI Sensor WG Reserved	15 available codes for describing the Device capabilities and function on the sensors' system.
LVR [2]	MIPI Sensor WG Reserved	
LVR [1]	MIPI Sensor WG Reserved	
LVR [0]	MIPI Sensor WG Reserved	

Figure 9 Legacy I²C Virtual Register (LVR)

All LVRs will be established by the higher-level entity that controls the I3C Bus, and are transferred to the I3C Bus Main Master prior to Bus configuration. The LVR content for all I²C Devices is always known by the Main Master. The LVR information can be transferred to any Secondary Masters present on the I3C Bus with the ***Define List of Slaves (DEFSLVS)*** CCC (see ***Section 5.1.9.3.7*** in the I3C Specification ***[MIPI03]***).

5.3 Dynamic Address Assignment (DAA)

A component connected to an I3C Bus can only behave fully as an I3C Device after being assigned (and receiving) a Dynamic Address (DA). The Dynamic Address is one of the most advantageous features of I3C Bus since:

- The DA allows any I3C Device on the Bus to be addressed with only 7 bits, avoiding the need for any traditional extended address space
- The DA value includes the Device's priority rank, which determines the winner of the In-Band Interrupt (IBI) request arbitration and hence the servicing order for IBI requests.

The initial assignment of DAs is the duty of the Main Master: as soon as the I3C Bus is powered-up, the Main Master assigns one DA to each of the connected Bus's clients. The Main Master has a unique position among other possible Masters that might be connected to the Bus: it is informed of the number and characteristics of all connected components to be enabled as I3C Devices. Priority rank is established *a priori*, as appropriate for the particular application controlling that I3C Bus.

Any Secondary Masters connected to the Bus need to be informed of the correspondence between the assigned DAs and their associated Devices. There are two ways to communicate this information:

- The Secondary Masters monitor the DA Assignment (DAA) procedure, or
- The Main Master uses the ***Define List of Slaves (DEFSLVS)*** CCC (***Section 5.1.9.3.7*** in the I3C Specification ***[MIPI03]***)

During normal operation after initial Bus configuration, the then Current Master can change a Device's DA. In addition, the then Current Master will assign a DA to each Device that is newly attached to the Bus via Hot-Join. (More details are provided below.)

The selection of the DA is vital for the correct functioning of the I3C Bus. The DA address is 7 bits wide, for a space of 128 possible values, but the number of physical Devices resident on a I3C Bus instantiation is limited by the maximum allowed capacitance at any IO pin (i.e., 50 pF). Although the I3C Specification states that an I3C Bus can only support up to eleven (11) I3C Slave Devices, the current consensus is that more than eleven DA could be used (in addition to the number of Devices physically present on the Bus, this count would also include all Virtual Devices and/or Group Addresses in use).

There are several constraints when selecting a DA, as detailed in the I3C Specification at ***Section 5.1.2.2.5 I3C Slave Address Restrictions***. A couple of noticeable reasons are the I3C-reserved addresses, and the addresses that are within a one-bit error of the I3C Broadcast Address (i.e., 7'h7E).

A good practice when selecting a DA is to allow spaces (i.e., unused addresses) in between the assigned addresses, in order to:

- Allow for further changes in the IBI processing priority ranking
- Provide locations for any expected Hot-Join Devices, i.e. pre-interleaved at the desired spot in the IBI priority ranking order

As an option, the DA selection could use efficient allocations as described in the I3C Specification at ***Section 5.1.2.2.2*** (***I3C Address Arbitration Optimization***). Notice the shortened duration of the address arbitration, resulting in an increased data rate.

Thus, the controlling application layer establishes a table of Dynamic Addresses on the Main Master, based on the desired priority ranking of the target Devices.

The Main Master can start assigning DAs to the corresponding Devices once all the Devices are ready. The I3C Bus management layer is responsible for ensuring such readiness.

The DA Assignment (DAA) procedure is described in the I3C Specification at ***Section 5.1.4.2*** (***Bus Initialization Sequence with Dynamic Address Assignment***).

It is optional, but recommended, to speed up the DAA procedure by first assigning DAs to all Devices with a known Static I^2C Address via the CCC ***Set Dynamic Address from Static Address (SETDASA)*** (***Section 5.1.9.3.10*** of the I3C Specification).

For the remaining I3C Device targets, the Main Master shall execute the DAA procedure. (Note that DAA is an I3C Mode, and hence the entry and exit rules are specific.) The Main Master starts the DAA procedure by using the CCC ***Enter Dynamic Address Assignment (ENTDAA)*** (***Section 5.1.9.3.4*** of the I3C Specification). The DAA procedure is described in detail at ***Section 5.1.4.2 Bus Initialization Sequence with Dynamic Address Assignment***.

For the Open Drain portions of the datagram, the timing parameters shall be suitable for the slowest I^2C target Device(s) as such Device(s) must be allowed to participate in the DAA procedure.

The DAA procedure ends with a STOP – a robust signaling condition easily identifiable by all Devices connected to the I3C Bus.

All I3C-capable Devices that were not yet assigned an I3C DA will respond to the ENTDAA CCC. A Hot-Join Device can (and will) only respond to ENTDAA after it requests an IBI using the reserved address 7'h02, 1'b0, and that IBI has been accepted.

The Main Master can end the DAA procedure at any time. Any Devices that haven't yet received an assigned I3C DA shall participate in the next DAA procedure (i.e., initiated some time later via an ENTDAA CCC).

DA Collisions: If any Devices use the "Random Value" variant of the 48-bit Provisional ID (per I3C Specification ***Section 5.1.4.1.1***), then there is a slight possibility that two different Devices on the I3C Bus could have the same BCR, DDR, and Provisional ID values, resulting in an address collision. A collision could also arise from an unfortunate signal integrity problem. A collision is detected when the final number of I3C DAs assigned is be less than the number of I3C-compliant Devices known to require an I3C DA. If a collision is detected, the Main Master shall use the CCC ***Reset Dynamic Address Assignment (RSTDAA)*** (I3C Specification ***Section 5.1.9.3.3***), which causes all I3C Devices to reset their I3C Dynamic Addresses, and then initiate a new DAA procedure. If a collision reoccurs more than a given number of times (three is recommended), then the Main Master shall inform the Application Layer that the I3C Bus is not functional.

After the I3C Bus has been successfully configured, the Current Master can dynamically change assigned DAs as desired for optimal operation of the particular running application, using the CCC ***Set New Dynamic Address (SETNEWDA)*** (***Section 5.1.9.3.11*** of the I3C Specification). If any Secondary Masters are connected to the I3C Bus, then the Current Master must inform them of any DA changes by using the ***Define List of Slaves (DEFSLVS)*** CCC (***Section 5.1.9.3.7*** in the I3C Specification).

The Current Master should assign a DA to Hot-Join Devices. However, not all Secondary Masters are capable of assigning DAs; if any such Secondary Masters are acting as the Current Master, then they shall return Bus Mastership to the Main Master (or to some other sufficiently capable Master that they know of), which will provide the Hot-Join Device with its requested DA. The Current Master will then inform the rest of the Masters via the ***Define List of Slaves (DEFSLVS)*** CCC (***Section 5.1.9.3.7*** in the I3C Specification).

As emphasized in the I3C Specification, having an I3C DA assigned is what distinguishes a Device performing as a I3C Device from a Device remaining in its initial power-up configuration. This distinction is particularly important when the I3C-capable Device is also an I^2C Device. I^2C Devices have their 50 ns IO spike filters enabled and active until the I3C DA is assigned, and this must be taken into account by the I3C Bus controlling layer for both the DAA procedure and the SETDASA CCC. Further, consideration must be given to the fact that the I^2C Devices will enable their 50 ns spike filters after the RSTDAA CCC.

Note that Hot-Join Devices do not have the 50 ns spike filter, because the I^2C specification does not support Hot-Join functionality.

5.4 Bus Topologies

The I3C Specification ***[MIPI03]*** defines the three Bus topologies discussed in this section: Point-to-Point, Daisy Chain, and Star-on-Stick.

- **Point-to-Point** (***Figure 10***) Bus topology has one Master connected to one Slave.

  The two Devices are connected by a board trace of Length LT.

  The medium could be board + cable, board trace only, or any other medium.

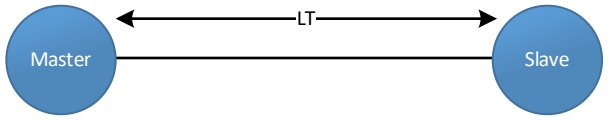

Figure 10 Point-to-Point Bus Topology

- **Daisy Chain** Bus topology (***Figure 11***) primarily means a multi-drop Bus, where a single Master is connected in series to two or more Slaves A and B (etc.).

  The distance of the medium between the Master and Slave A is L1, and the distance from Slave A to Slave B is L2; thus, Slave B's distance LT from the Master is L1 + L2.

  Figure 11 also shows each Slave connected through a stub with length St, and after the stub another medium of length L3. As a result, Slave A's distance from the Master is L1 + St + L3, and Slave B's distance from the Master is L1 + L2 + St + L3. Stub length plays an important role in signal integrity of the I3C Bus signals (SDA and SCL).

  The medium could be board trace only, or board and cables.

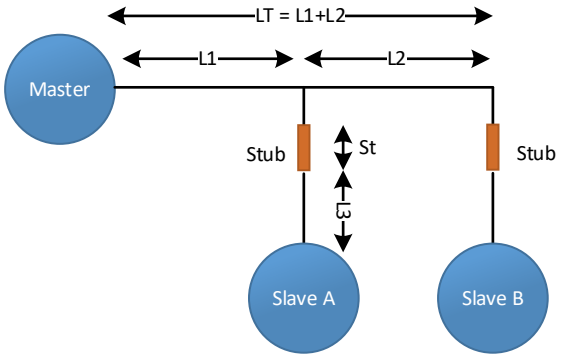

Figure 11 Daisy Chain Bus Topology

- **Star-on-Stick** Bus topologies (***Figure 12***) are for Multi-drop Busses where all of the Slaves are far apart from the Master, but close to each other (i.e., L1 is much greater than L2).

  The distance from the Master to each Device is L1 + st + L2, and L1 is much greater than L2.

  Although ***Figure 12*** shows equal distances, there could be per-Slave distance variations. The System Designer must make sure that the distance for each Slave is as equal as possible. In Star-on-Stick topologies, stubs also play an important role in signal integrity of the I3C Bus signals (SDA and SCL).

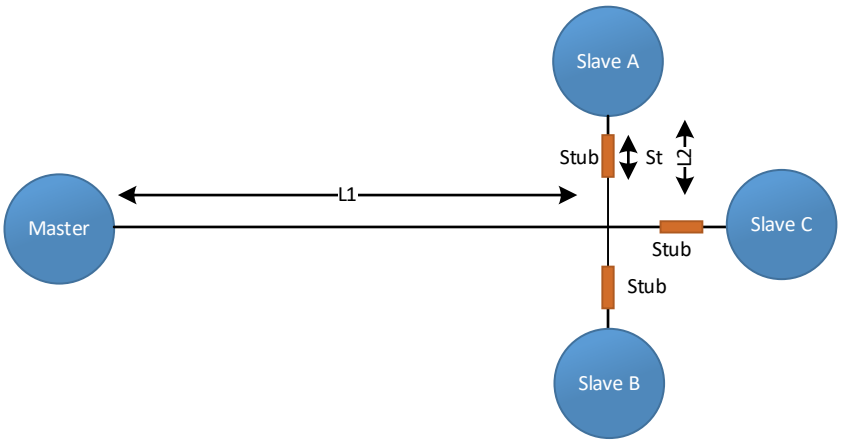

Figure 12 Star-on-Stick Bus Topology

5.4.1 Trace/Medium

In the Bus topologies shown in ***Figure 10*** through ***Figure 12***, the medium connected to the Devices could be FR4 board trace, or board trace + cable, or trace + via + trace + cable, etc. The System Designer must make sure that the length of the trace or medium is chosen to meet the maximum Bus capacitance supported by the I3C Specification.

Example: A Point-to-Point Bus topology targeting 20 inches will result in 44 pF of board trace capacitance, where FR4 is 2.2 pF/inch.

Note:

The case of FR4 trace + Cable will result in different medium capacitance, and the calculation will not be direct as compared to single FR4 trace, because the discontinuity in the signal board trace also impacts the capacitance.

5.4.1.1 Length

The length of board trace that can be supported depends upon various factors, key among them being:

- **Bus Topologies:** Bus topologies impact the length supported for SDR/DDR mode and for HDR mode. Point-to-Point topologies will permit longer board traces or medium, assuming fixed capacitance from load.
- **Device Capacitance:** Device capacitance impacts Bus performance and maximum length for the targeted medium. The number of Devices present on the Bus will also increase the Bus capacitance, thus reducing medium length.
- **Reflections:** Signal integrity will affect the targeted medium length. The System Designer must be sure to avoid reflection or discontinuity, which will directly reduce targeted Bus length.

- **Medium:** Selection of a lossy medium will reduce targeted board trace length.

5.4.1.2 Material

The I3C Specification doesn't constrain selection of the trace or medium material. The medium could be either lossy, or less lossy. The degree of loss will directly impact overall medium length. The System Designer must take care to select the right medium within the constraints of targeted board space.

5.4.1.3 Design/Matching

The SDA and SCL lines should be matched both in terms of silicon design and in terms of board design. A mismatch will result in skew and limit the timing budget.

An impedance Characteristic mismatch between the board the buffer will result in reflections, directly impacting and reducing the timing window for read/write operations.

For writes, skew will be a key parameter that must be controlled for SDR/DDR Mode. For HDR-TSP/TSL Mode, where both lines are driving data, Read/Write skew will limit the timing window.

The System Designer must make sure to match the impedance Characteristic, avoid reflection through damping resistors, and avoid mismatches as far as possible.

5.4.1.4 GND Management

Although this is not stated in the I3C Specification, as a good practice to ensure reliable operation of single-ended interfaces, Ground differentials between Master and Slave should be held to be within +/-50 mV.

5.4.1.5 Stubs

Stubs play an important role in keeping signal integrity well-defined. Stubs will result in discontinuity, which will in turn produce reflection and timing loss. The greater the stub length, the greater the reflection. It is recommended to limit stub lengths to a maximum of 1.5 inch.

5.4.2 Mixed Bus Considerations

I3C supports legacy I^2C Devices using Fast-mode (400 KHz) and FastMode+ (1 MHz) with the 50 ns spike filter, but no other I^2C modes, nor I^2C Devices lacking the spike filter, nor I^2C Devices that stretch the clock.

5.4.3 Point to Point Optimizations

I3C covers most of SPI's speed range, but is not intended for the highest speed grades which really only work well with a point-to-point interface (e.g., as in SPI Flash).

5.4.4 Hot-Join/Failsafe/Offline Capability

The I3C Bus protocol supports a mechanism for Slaves to join the I3C Bus after it has already been configured. The I3C Specification defines the conditions under which a Slave can do that (e.g., a Slave must wait for a Bus Idle condition).

Slaves that join the Bus after it is already started, whether because they weren't previously powered or because they were physically inserted into the Bus. The Hot-Join mechanism allows the Slave to notify the Master that the new Slave is present and ready to receive a Dynamic Address.

The Bus Idle condition is defined to help ensure Bus stability during Hot-Join events. This condition is defined as a period during which the Bus Available condition is sustained continuously for a duration of at least t_{IDLE}. As Hot-Join Devices do not know the Bus condition, they must wait until the Bus is Idle (meaning that the SCL and SDA lines are both High for the t_{IDLE} period, ~1 ms).

5.4.4.1 Hot Plug

The I3C Specification does not attempt to define physical connectors for Hot Plug. Instead, it provides basic guidance and some rules. It otherwise only focuses on the logical effect of a Hot-Join onto the I3C Bus. The most critical rule is that the Hot Plug or Unplug cannot disturb the SCL or SDA in a way perceptible to the Slaves and Masters on the I3C Bus. This generally implies and informs three guidelines for the Hot Plug connector design:

1. Ground will be connected first, to stabilize the ground plane.
   - This also requires consideration of impedance issues, to avoid any ground plane disturbance on the mother/base board.
   - The typical approach is to make the ground connector ‘finger’ longer than the other fingers, so that ground is always connected first. It can also be handled via a connector jacket that connects first.
2. You may want Power to be connected before IOs.
   - See the Failsafe section below for some of the considerations.
   - The power for the I3C Devices on the plug-in module should be reasonably voltage matched.
   - Inrush management can be needed to avoid an unacceptable voltage drop to the I3C Bus components. Generally this is a ramp rate if a PoL Regulator is used on the plug-in module, or adequate decoupling capacitance on the base board near the connector.
   - It’s useful to consider a ferrite bead or other approaches to control any high-frequency noise that could transferred onto the base board, or that could cause noise amplification on the newly plugged-in board.
3. The IOs for SCL and SDA must not cause undue current draw during the connection, but otherwise have a few special restrictions:
   - Besides Failsafe (if power is not pre-applied and long enough to avoid this concern), it is critical to consider any pre-power-up anti-float measures that the Device has (e.g., weak Pull-Up or Pull-Down resistances in the pads), and also general pad design in terms of effective capacitance, especially in this newly powered-up (or not yet powered-up) state.
   - The I3C Bus may not be able to tolerate large shifts in capacitance, or ringing that such a connection could potentially cause. Therefore, care is required in the design of the connection and/or the connector-related characteristics of the Bus topology.
   - The System Designer should consider the new topology created with the connection and without, including the effective stub from the plug-in module, as well as any inductance from the connector itself.
   - The IOs are in Hi-Z until the Hot-Join mechanism drives SDA.

5.4.4.2 Failsafe

As explained in ***Section 5.1.5.1*** of the I3C Specification, any IO on the I3C Bus while unpowered must use Failsafe pads, i.e., no parasitic current draw from the SCL or SDA lines. Parasitic draw can be due to the current trying to power the V_{DD} rail in the Device through the PMOS (where it acts like a diode), excess ground leakage through the NMOS, or through the ESD structure (when it uses an active/powered mechanism vs. fixed voltage such as snapback). The input (Schmitt) can also cause leakage beyond the allowed leakage and capacitance, depending on the design. It is also important to look at such parasitic leakage over the temperature range at which the Bus will be operating: some Devices can perform well at room temperature, but have excessive current at higher temperatures, including due to self-heating (e.g., the Device may have been unplugged and re-plugged before cooling down).

Each Device manufacturer should document what types of pads they are using; if not, the System Designer cannot assume that the pads are Failsafe. If not Failsafe, then separate methods can be used to isolate the Device from the Bus when using Offline, and power should be pre-connected when using Hot-Plug. For Offline, this can be isolation transistors on SCL and SDA, as long as they are capable of managing I3C switching speeds (i.e., 12.5 MHz) without excess delay (i.e., under 1 ns lag). It's also possible to disconnect from ground and power when offline, as long as reconnection can be done without violating the above considerations.

5.4.4.3 Offline

Offline is defined in I3C as a form of Hot-Join, although Devices with retention can use a faster re-join. The basic model is a Device that can be powered down and powered back up dynamically while the I3C Bus remains active.

The offline Device might be powered up when the I3C Bus first is started (and so de-powered later), or it might be unpowered when the I3C Bus is first started, then powered up later.

The offline Device then must be Failsafe on the SCL and SDA lines when unpowered, and cannot disturb the I3C Bus when powering up or losing power. The definition of disturbance is covered in ***Section 5.1.5*** of the I3C Specification (Hot-Join).

Offline Devices will rejoin the Bus either using a Hot-Join request, or if they still retain their Dynamic Address from earlier in the same session, they can use an IBI, or simply wait for a message. In the case of a retention scheme (e.g., standby with power supplied only to a special retention memory and pin set), it is important to distinguish whether the I3C Bus is initially coming up, vs. is still in the same session, because the saved Dynamic Address is only meaningful for the one session. If in doubt, Hot-Join is safer.

In both cases of Hot-Join and Offline, the Device must first monitor the Bus to wait for a t_{IDLE} period of inactivity. This assures it that the Bus is free, and therefore that it is safe to emit the Hot-Join IBI.

In general, it is advisable for Devices which will Hot-Join the Bus to be configurable to either Hot-Join or not, rather than always Hot-Joining. That is, if the Device will be powered up along with the I3C Bus (i.e., when the Bus first starts), then as a general rule that Device should not try to Hot-Join. As a result, the Device might need to be configured (i.e., via some mechanism such as NVMEM, pin-strap, etc.) to either Hot-Join or not.

5.5 Physical/Electrical/Testing Considerations (I3C CTS)

In order to improve interoperability of products implementing the I3C interface, the MIPI Alliance Sensor Working Group has developed an I3C Conformance Test Suite (CTS).

The I3C CTS contains tests designed to determine whether a product conforms to a subset of the requirements defined in the I3C Specification v1.0. In order to meet time-to-market requirements imposed by the rapid adoption of I3C in the marketplace, the first version of the CTS is intentionally limited and focuses on:

- SDR-only Devices without optional I3C capabilities
- All Master and Slave Error Detection and Recovery methods
- Basic HDR Enter/tolerance/Restart/Exit

The Sensor WG considers the I3C CTS a living document, and plans to continue expanding CTS scope through future revisions, eventually encompassing all required and optional features of the I3C Specification.

The CTS is organized as one section of tests for a Master Device under test (DUT), and a separate section of tests for a Slave DUT. Within each section, the tests appear in the same order as the I3C Specification Sections containing the related requirement(s), to make it easier to find the relevant Specification details.

5.6 Bus High Keeper

The I3C Bus always needs a weak High pull on the SDA line, for times when there is no active drive and no strong Pull-Up resistor (or equivalent).

Per the I3C Specification, this SDA weak High pull could be provided in either of two ways:

1. The I3C Master will be responsible for this, and it provides the Bus High Keeper effect in some form or fashion.

   Note:

   If more than one Master will be used on the Bus (i.e., if there are any Secondary Masters), then each one will also need to be able to provide the High Keeper while it is the Current Master. If any Secondary Master can't do this, then the second method below would need to be used.

2. When not supported by the Master (or not supported by all Masters), the Bus must be wired with weak passive Pull-Up resistors on the SDA line. These could be 50 KΩ, 100 KΩ, or higher-value resistors. The exact value, and whether more than one resistor is used in parallel, depends on the Bus in terms of leakage sources (e.g., Slaves and Masters over the Bus topology) and noise induction.

   Note:

   If the Master will not be using its strong Pull-Up on the SDA line during Bus Free condition (idle), then a stronger passive Pull-Up resistor should be used to prevent noise from causing false START requests.

The SCL might also need a weak High pull. If the Master (or Masters) will not be providing a strong or weak Pull-Up on SCL when the Bus is in the Bus Free condition (idled), including if the Master is in deep-sleep state, then a passive weak Pull-Up should be provided on the SCL line. Likewise, if HDR-TSP or HDR-TSL is to be used, then if the Master will not be providing a weak pull High on SCL during the handoff, then a weak passive Pull-Up must be wired onto SCL. The exact value, and whether more than one resistor is used in parallel, depends on the Bus in terms of leakage sources (likely far less than for SDA, since SCL is passive for most Slaves) and the incidence of noise on SCL. It is important that the passive Pull-Up is strong enough to hold SCL High during a long-idled Bus state.

Example: If the total leakage current is determined to be 10 µA from the I3C Devices on the SDA, and we are using a 3.3V Bus, then the minimum resistance needed is:

$$3.3\ V / 0.00001\ A = 330\ K\Omega.$$

Now, we have to allow for noise current, which will be lowered by the capacitance of the line; so, for example, we can decide we have a 20 µA displacement due to the calculated capacitance of our line flattening a spike of ~2 V ground coupled. So, we sum those to equivalent leakages (we do not care when noise is in the opposite direction of static leakage normally, but we do care when it is in the same direction), and arrive at 110 KΩ. So, we choose to use the more normal 100 KΩ Pull-Up.

Further, for a long trace line we will place more than one resistor in parallel along the trace, to ensure stability regardless of the source and placement of the leakage or noise.

Note that the resistor's only job is to keep the V above V_{IH} (and really V_{IH}-V_{hys}) after it is already parked at High (i.e., was driven High). As a result, a weaker Pull-Up could be used in cases of more noise, as long as the V remains above V_{IH}. The essential goal is to keep V above V_{IH} **for the I3C Devices on the Bus**; a V dip along an empty stretch of trace would only matter if it were to impact I3C Devices in other areas.

5.7 Bridge Devices

The MIPI I3C specification covers inter-bus bridging support, both passively and actively. In all cases, the bridged targets (i.e., the Devices that are being bridged) are virtual I3C Slaves, each with their own I3C Dynamic Address, that transact with the I3C Bus through the Bridge Device.

There are two general ways an I3C Bridge Device can get the bridged targets configured with their I3C Dynamic Addresses:

1. The Master Software knows what is physically attached to the Bridge Device, using information provided by the system designer (i.e., by table or otherwise), and uses the SETBRGTGT CCC to inform the Bridge Device about each endpoint/Slave and its assigned Dynamic Address.
2. The Bridge Device itself knows what is attached, and uses the ENTDAA CCC to represent each one separately to the Master, so that each bridged Device is assigned a unique Dynamic Address.

Note:

- *The Bridge Device will have a Dynamic Address in the first case, and can optionally have its own Dynamic Address in the second case.*
- *The Bridge could also be configured using some private contract between the Master and the Bridge Slave, and the model would likely be similar to the effect of the SETBRGTGT CCC: the Master allocates a set of Dynamic Addresses. It could also be that the Master blindly programs the Bridge (from higher level software) and then uses the ENTDAA CCC again to learn each target, whether by Master decision or as the result of a Hot Join.*
- *Bridging the other way, i.e., from some other bus into an I3C Bus, would require a full Master or Secondary Master; as a result, that would work the same as with any I3C Master or Secondary Master.*

5.7.1 Use of SETBRGTGT (Set Bridge Targets)

The steps to use the SETBRGTGT CCC are as follows:

1. The system designer implants knowledge about the endpoint Slaves in the Master firmware, whether an HCI-I3C Master or a more direct I3C Master.
   A. This can be a table, such as is used to capture knowledge about other I3C (and any legacy I^2C) Slaves, or it can be handled in other ways.
   B. This would include the level of detail to indicate protocol, which pin-channels, etc., as suitable for the ID[15:0] field of the SETBRGTGT CCC.
2. The I3C Bridge Device participates in the normal I3C DAA mechanism, whether at Bus initialization or as the result of a Hot-Join.
   A. This means that the I3C Bridge Device can have a static address and the SETDASA CCC is used, or it can only respond to the ENTDAA CCC.
   B. The I3C Bridge Device should have the Bridge DCR value, and its BCR should be that of a normal Slave.
   C. The I3C Bridge Device should have a normal manufacturer ID and 48-bit PID. If more than one physical Bridge Device can be used on the same I3C Bus, then it should use the instance ID to separate them.
   D. The resulting Dynamic Address (DA) is then used only for communication between the Master and the Bridge Device itself. It could never be used, or it could be used for such purposes as configuration, error handling, etc.
3. After the SETDASA and ENTDAA process, the Master will then use the SETBRGTGT CCC to configure the Bridge Device.
   A. The Bridge Device will assign a unique Dynamic Address for each endpoint which is being bridged. It will use the ID[15:0] field to provide info to the Bridge Device on the attached Device, as needed.

B. The bridged Devices will likely be treated as having the BCR ‘limitations’ bit set. This is so that the Master can interrogate the bridged targets for their maximum data transfer limits, read turnaround time delays, and any other consideration(s).

C. The bridged Devices can generate an IBI via the Bridge Device, and the Master can use the GETBCR CCC to read how they handle the IBI, unless this is already known from the system designer.

4. Each bridged Device will be treated as normal I3C Slave, though perhaps with some limitations necessitated by being bridged:

A. Slow slaves like I^2C will obviously need more time for write-read turnaround. This can be returned via the GETMXDS CCC.

B. Some Slaves and/or the Bridge will have limits on maximum write data which can be settable using the SET/GETMAX CCCs.

C. The Bridge Device can NACK writes to a Slave if its buffers are full.

D. The Bridge Device will likely limit advanced and optional features such as HDR, specialized CCCs, and other uses that may not translate well.

5.7.2 Use of ENTDAA to Provide the Bridged Targets

The steps for a Bridge Device which has the knowledge about what it bridges is quite different than the SETBRGTGT CCC mechanism. In this case, the Master does not have to know anything about the Bridge Device in advance.

This mechanism works as follows:

1. The Bridge Device has some way to know what is connected to it, with some level of specificity.

A. The Bridge Device might only know the bus format (e.g. I^2C, SPI, UART, etc.) for each slave, or it might know what type of device it is and other info.

B. The Bridge Device might know this information from pin straps, from pin testing at reset, from programmed fuses or NVMEM, or from some other means.

Note:

The variant of the Master informing the Bridge Device via a private message works the same, and so is not discussed separately here. The only notable aspect is that the Bridge Device itself must have an address for the Master to do this (whether using I^2C or by assigning it an I3C Dynamic Address and then informing it).

2. During ENTDAA mode, whether for bus initialization or for Hot-Join, the Bridge Device represents each bridged Slave as a separate and unique device.

A. The Manufacturer ID, PID, BCR, and DCR can be mostly reused, different completely, different only in Instance ID only, etc. In all cases, each Slave that is bridged must be unique by ID or DCR.

B. The Master will assign each bridged Slave a unique DA.

C. The Bridge Device itself can also represent itself with a unique ID+DCR if it needs or wants; this could also have been done in a previous round, if the Master programs it by private contract.

D. In general, the BCR will be more about the Bridge Device and bridging:

i. Each Slave should indicate a ‘limitation’. The limitation bit will ensure the Master makes requests such as GETMXDS, GET max data sizes, etc.

ii. The HDR bit should be set if HDR is supported by the Bridge itself (since an I3C issue)

iii. The IBI bit should be set if the Bridge will use IBI for that Slave.

3. Each bridged Device will be treated as a normal I3C Slave, though perhaps with some limitations necessitated by being bridged:

A. Slow Slaves like I^2C will obviously need more time for write-read turnaround. This can be returned via the GETMXDS CCC.
B. Some Slaves and/or the Bridge will have limits on maximum write data which can be settable using the SET/GETMAX CCCs.
C. The Bridge Device could NACK writes to a Slave if its buffers are full.
D. The Bridge Device will likely limit advanced and optional features such as HDR, specialized CCCs, and other uses that may not translate well.

5.7.3 Bridged Devices and IBI

The Bridge Device can use an IBI when a Bridged Slave Device pulls a GPIO of the Bridge Device to notify that it has new data (or in the case of UART, that it has sent new data). The IBI could also be used for errors, such as overrun. The Master and Slave must have some agreement on its use.

The Bridge Device could choose to support Time Control, such as Asynchronous Time Control, which would allow the Bridge Device to record the time at which the GPIO was pulled (or UART sent data) and provide that to the Master.

The Bridge Device could first read data from the Slave on GPIO assert (before the IBI is generated), or it could simply wait for the Master to request the data.