I3C Application Note: Virtual Devices and Virtual Targets

For MIPI I3C® v1.1+ and I3C Basic^{SM} v1.1.1+

App Note Version 1.0
30 August 2021

MIPI Board Approved 4 September 2021 – Approved for Public Release
Public Release Edition

Contents

Figures

Tables

Release History

Date	Version	Description
04-Sep-2021	v1.0	Initial Board approved release.

This page intentionally left blank.

1 Introduction

The MIPI I3C Bus interface ***[MIPI01]*** 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-Controller 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. The I3C interface is also extensible to newer and more advanced used cases that go beyond the original scope of sensor integrations. This Application Note is intended to help users understand how the I3C Bus can be extended to support device virtualization using various types of composite I3C Devices, connections to other buses (including I3C Bus segments) and other advanced integrations of I3C Target functionality.

1.1 Scope

This Application Note on Virtual Devices and Virtual Targets is intended to guide three different groups:

- Those developing MIPI I3C Target Devices with advanced capabilities, to understand how these Devices can expose advanced Virtual Device capabilities and implement them while remaining compatible with MIPI I3C v1.1.
- System Designers who have to design systems that integrate such advanced MIPI I3C Devices, who need to know the implementation concerns with these advanced capabilities as well as the configuration aspects that need to be considered by a MIPI I3C Controller and its connected Host System.
- MIPI I3C Controller software considerations in the one or more Controller-capable Devices 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 advanced features 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 Application Note is also a complement to the ***I3C Application Note: General Topics [MIPI04]***, which covers many fundamental topics of integrating MIPI I3C Devices in a system, as well as other I3C Bus considerations that apply to all I3C Devices (i.e., electrical details and topology).

This Application Note is intended to be used together with the I3C Specification ***[MIPI01]***. Each Application Note section corresponds to one or more Specification sections, primarily focusing on Specification ***Section 6***, ***I3C Electrical Specifications***. The Application Note amplifies the Specification with additional context (e.g., analysis data to back up recommended use models) and details (e.g., specific usage notes for more advanced use cases) that would not be appropriate in a protocol specification.

2 Terminology

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

2.1 Definitions

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

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

Controller: An I3C Device (or Role embodied by such a Device) that is capable of controlling the I3C Bus.

Note:

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

Downstream: An I3C Bus or I3C Bus Segment that is subordinate to, or at a lower level of hierarchy compared to, another I3C Bus (or Bus Segment). A Downstream Bus Segment typically receives I3C transfer commands from its Upstream I3C Bus Controller (i.e., the Controller in charge of, or at a higher level of hierarchy), and these I3C transfer commands are passed or translated via an agent Device (such as a Bridge Device, Routing Device, or Hub Device) that spans both I3C Bus Segments.

Hub Device: An I3C Device that provides isolation between Upstream and Downstream I3C Bus Segments having different electrical parameters, while still allowing transactions to pass between segments.

I3C Bus. As defined in the MIPI I3C Specification.

I3C Bus Segment: An I3C Bus that exists within a hierarchy of several other I3C Buses, where a Bridge Device, Routing Device, or Hub Device acts as an agent that relays transfer commands and responses from one such I3C Bus to another Bus.

Manager Function: A Virtual Target presented by a Bridge Device, Routing Device or Hub Device that allows the I3C Controller to manage and control the Device itself, via Private Write/Read transfers and/or CCCs.

Master: Deprecated term, see Controller.

Offline Capable: As defined in the MIPI I3C Specification.

Route: As defined in the MIPI I3C Specification.

Routing Device: As defined in the MIPI I3C Specification. Device on the I3C Bus that allows conversations between two or more different I3C Buses (i.e., Bus Segments) through integrated logic on the given Device. The Routing Device (as distinct from a Bridge Device) buffers/queues the transactions using store-and-forward architecture, causing the Device to be non-transparent.

Routing Target Function: A Virtual Target presented by a Routing Device which exposes a Route and accepts Private Read or Private Write transactions to/from the Downstream I3C Bus Segment for that Route through the Routing Device, as well as CCCs to control the parameters for that Route. See also "Route" in the MIPI I3C Specification.

Shared Peripheral: Common logic of an I3C Device that handles low-level I3C Bus protocol, usually including FIFOs or other buffers for Virtual Targets, and an interface to the rest of the system that enables multiple applications, where each can present an individual Virtual Target. See "Peripheral" in the MIPI I3C Specification.

Slave: Deprecated term, see Target.

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

Target: An I3C Device (or a Role embodied by such a Device) that can only respond to either Common or individual commands from a Controller. Some Targets can initiate In-Band Interrupt requests on the I3C Bus.

Note:

> *In previous versions of the I3C Specification, a Target Device was called a "Slave" Device. This Application Note uses the updated normative term "Target." Please note that the technical definition of such a Device, and its Role on an I3C Bus, are unchanged.*

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

Upstream: An I3C Bus or I3C Bus Segment that is in charge of, or at a higher level of hierarchy compared to, another I3C Bus (or Bus Segment). An Upstream Bus Segment has an I3C Bus Controller (i.e., Primary Controller) that directs the operations on one or more subordinate Downstream Bus Segments, connected via an agent Device (such as a Bridge Device, Routing Device, or Hub Device) that spans both I3C Bus Segments.

Virtual Function: A virtualized entity or endpoint that exists within an I3C Device and which can receive I3C transfer commands that imply actions or special transactions that affect other I3C Devices, such as those on a Downstream I3C Bus Segment. Virtual functions are typically associated with Virtual Targets presented to the Upstream I3C Bus Segment, and often have associated functions that suit the purpose and capabilities exposed by such an I3C Device.

Virtual Target: As defined in the MIPI I3C Specification. May act as, or be presented as, a uniquely addressable I3C Target on the I3C Bus.

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

2.2 Abbreviations

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

2.3 Acronyms

BCR	Bus Characteristics Register
CCC	Common Command Code
DCR	Device Characteristics Register
I3C	MIPI Improved Inter Integrated Circuit interface or its Specification document ***[MIPI01]***
PID	Provisional ID (i.e., 48-bit unique identifier for I3C Devices)
SoC	System-on-Chip
VT	Virtual Target

3 References

[MIPI01]	*MIPI Alliance Specification for I3C® (Improved Inter Integrated Circuit)*, version 1.1.1, MIPI Alliance, Inc., 11 June 2021 (MIPI Board Adopted 11 June 2021).
[MIPI02]	*MIPI Alliance Specification for I3C® (Improved Inter Integrated Circuit)*, version 1.1 incorporating Errata 01, MIPI Alliance, Inc., 27 December 2019 (MIPI Board Adopted 11 December 2019; Errata 01 approved 24 June 2020).
[MIPI03]	MIPI Alliance, Inc., "Current I3C Device Characteristic Register (DCR) Assignments", <https://www.mipi.org/MIPI_I3C_device_characteristics_register>, last accessed 4 September 2021.
[MIPI04]	*MIPI Alliance I3C Application Note: General Topics*, App Note version 1.0, MIPI Alliance, Inc., In Press.
[MIPI05]	*MIPI Alliance Specification for Debug for I3C*^{SM}, version 1.0, MIPI Alliance, Inc., 21 April 2020 (MIPI Board Adopted 4 September 2020).

4 Overview

This Application Note describes several broad categories of information for Virtual Devices and Virtual Targets:

- Details of a few typical and supported capabilities, including different advanced capabilities. This covers the allowances and challenges presented by each such set of capabilities.
- Details of added functionality for such capabilities, including the set of which features are typically integrated together (i.e., used in concert) to build a typical, coherent use case.
- Integration aspects that are relevant for system designers and hardware/software integrators who need to make use of these capabilities, in order to enable a particular use case or present such capabilities to another Host using I3C.

4.1 Supported Capabilities

Readers developing aspects of such I3C-based systems should consult the relevant sections of this Application Note, which are intended to help avoid mistakes and to provide guidance not found in the I3C Specification ***[MIPI01]***.

Analysis data is provided to help understand the use cases and impacts of choosing to implement certain advanced capabilities relating Virtual Target behavior in a MIPI I3C Device. This analysis data could be used, for example, when choosing what features and capabilities to integrate into advanced I3C Devices that expose Virtual Targets on an I3C Bus, or into I3C Devices that bridge, present, or otherwise expose other downstream entities to an I3C Bus via standard I3C Read/Write transfers and/or CCCs using a given I3C content protocol.

It is anticipated that the nature of such I3C Devices, their use cases for a particular application, and the specific I3C content protocol chosen to access them might vary substantially. Implementers could make different choices for a particular implementation, due to the nature of the functionality that is exposed, or the aspects of downstream entities (i.e., devices on another bus, fabric or other interconnect) that are presented or virtualized as Virtual Targets. However, the core concepts of Virtual Target capabilities remain largely the same across the different classes of I3C Devices that expose or present Virtual Target capabilities or other similar I3C Device virtualization techniques that are described in this Application Note.

4.2 Using Added Functionality

The I3C Specification ***[MIPI01]*** 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 Controller.

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:

- Offering electrical guidance on simple Devices that integrate multiple I3C Targets (i.e., with no advanced capabilities or features)
- Defining the various types of buffers, queues, or other structures recommended or required in Shared Peripheral Logic for a composite I3C Device that presents multiple I3C Targets on the Bus, and allows an I3C Controller to interact with the I3C Targets using an I3C content protocol
- Understanding how the Target Reset Pattern and RSTACT CCC can be used to selectively reset certain functions or "endpoints" within an advanced I3C Device, and defining which Target Reset actions affect which internal entities or logic in such an I3C Device that presents multiple Virtual Targets
- Providing guidance on how Group Addresses and/or Multi-Lane configuration affects an I3C Device that supports and presents multiple I3C Targets on the Bus
- Enabling I3C Bridge Devices or I3C Routing Devices
- Enabling I3C Hub Devices that have advanced capabilities and include at least one Virtual Target that provides management of the Hub isolation circuitry

5 Virtual Target Concepts

This section discusses key guidelines that system integrators will want to follow, in order to implement various types of I3C Devices that embody the concept of Virtual Devices.

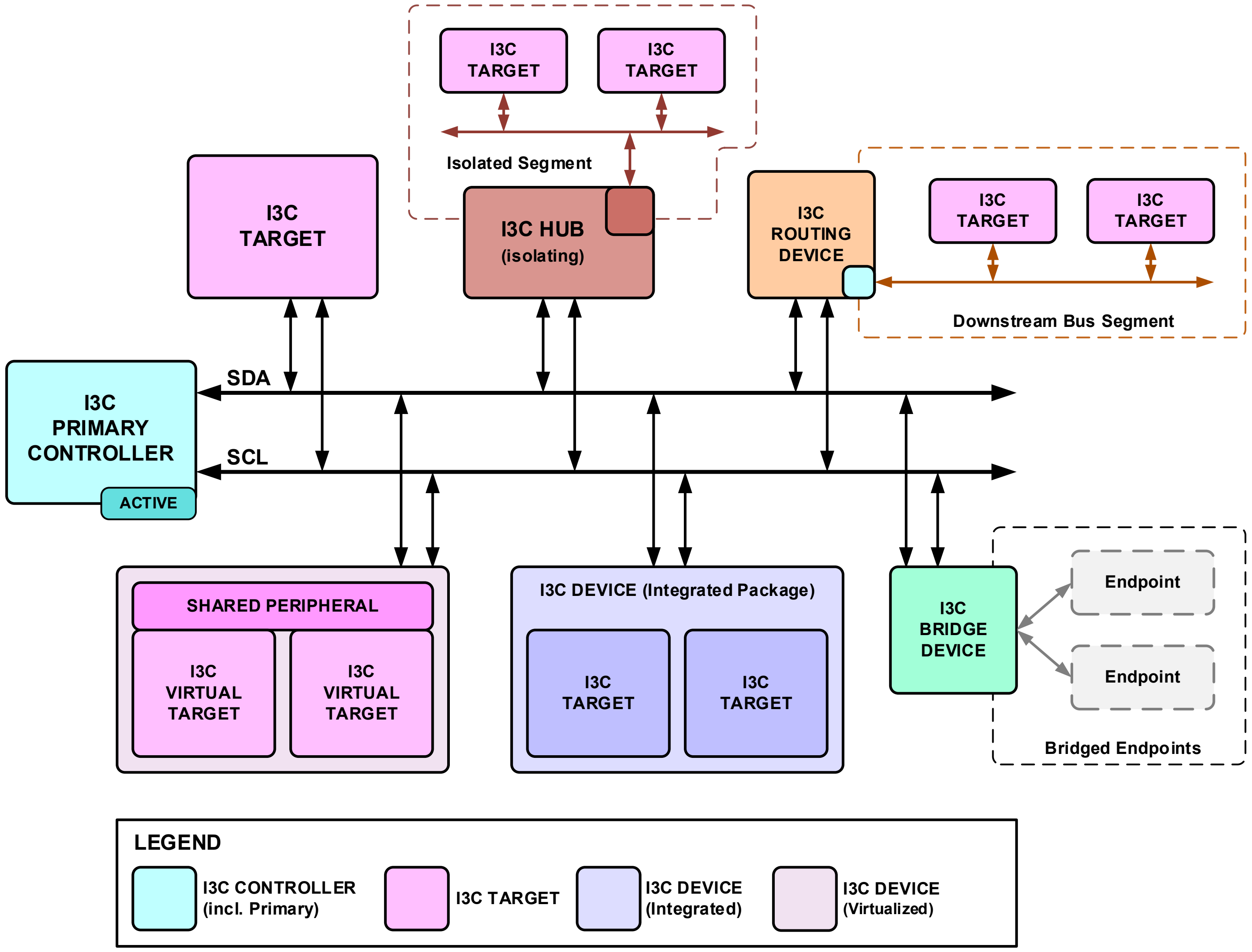
Table 1 Section 5 Subjects

Subject	Section
Simple Integrated I3C Devices with Virtual Targets	***5.1***
Composite I3C Devices presenting Virtual Targets	***5.2***
I3C Bridge Devices	***6.2***
I3C Routing Devices	***6.3***
I3C Hub Devices with advanced isolation capabilities	***6.4***

The configuration of a given I3C Bus will depend upon the characteristics of the I3C Devices intended to be active on that I3C Bus, as well as the use cases that drive the need for particular I3C Devices that expose or present Virtual Targets. For I3C Devices that present Virtual Targets on an I3C Bus, many implementation choices are possible, and this Application Note will interpret the requirements of Virtual Targets per the MIPI I3C Specification. This Application Note will also show the most common options as well as recommendations that can be applied to different situations.

Figure 1 shows an example I3C Bus that contains a variety of I3C Devices that embody or present Virtual Targets, of the following types:

- **Simple integrated I3C Devices** with multiple Virtual Targets in a single unit such as a die, stack, or package (shown in the color purple)
- **Composite I3C Devices** using Shared Peripheral logic to present multiple Virtual Targets (shown in the color pink)
- **I3C Bridge Devices** that expose bridged Targets (i.e., to a different bus such as I^2C or SPI) where each bridged endpoint is presented as an individual Virtual Target
- **I3C Routing Devices** that enable transactions to/from other I3C Buses, with each Downstream Bus segment exposed as a separate Virtual Target to which the I3C Controller can send Private Write/Read transactions
- **I3C Hub Devices** that provide electrical isolation for some I3C Targets on a Downstream I3C Bus Segment, but that do not change the fundamental nature of the I3C Bus protocol that the I3C Controller uses for transactions to/from these I3C Targets



203

Figure 1 I3C Bus with I3C Devices and Virtual Targets

5.1 Simple Integrated I3C Devices

Virtual Targets inside an I3C Device can be a simple integration of multiple I3C Target Devices that exist within the same physical unit or component, such as a replicated die, a die/wafer stack, or an integrated package. This integration relies on different methods that can combine multiple such units together, for the purpose of taking existing I3C Device implementations and bundling them together. Such steps could be taken to save space, to reduce physical integration costs, or to meet other platform design considerations without requiring a full re-design of existing I3C Device logic.

The specific nature of the integration method does not matter for this Application Note, but the core concepts below apply for any of the methods:

- Each I3C Target Device might otherwise exist in a separate instantiation, if it were manufactured and packaged separately (i.e., in a unit that included this I3C Device alone without other such I3C Devices).
- Each I3C Target Device could be separable from the other I3C Devices, and would neither expose nor rely on any other, more advanced Virtual Target capabilities (i.e., those described in other sections of this Application Note).

For many I3C Devices that present such Virtual Targets, the simple integration model is well suited to reusing existing I3C Device logic that can be packaged into a new form factor by a component designer.

5.1.1 Internal Topologies

For use cases that rely on existing I3C Device logic, a simple integration could consist of various types of I3C Devices that act as I3C Targets within the same unit as an integrated (i.e., combined) Device. In this manner, each I3C Target is a Virtual Target because it resides within the same physical Device, but does not necessarily possess any additional advanced functionality, nor advertise any additional Virtual Target capabilities on an I3C Bus.

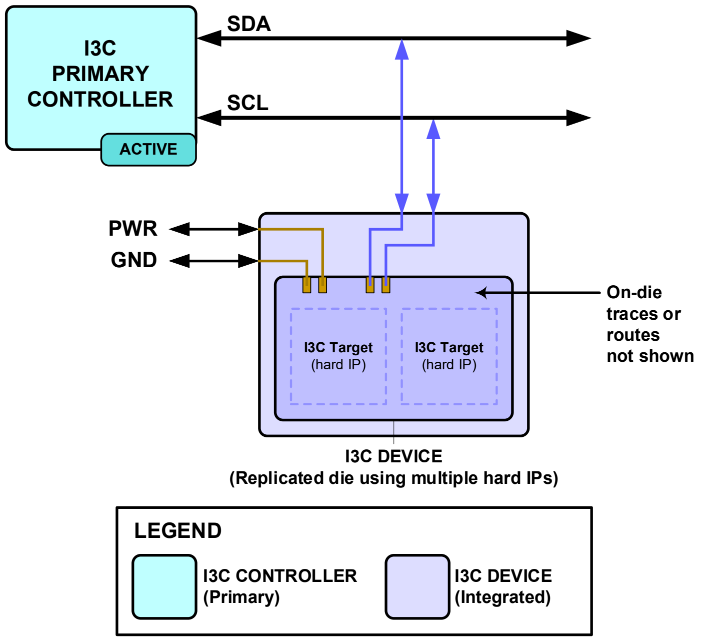
There could be various topologies for such an integration. As examples, several models for topologies are provided:

- An I3C Bus that includes one such integrated I3C Device with multiple Virtual Targets could be similar to a "Star-on-Stick" Bus topology, as described in ***Section 5.4*** of the ***I3C System Integrator's Application Note [MIPI04]***.
- An I3C Bus that includes one such integrated I3C Device with multiple Virtual Targets could be similar to a "Daisy-Chain" Bus topology, as described in ***Section 5.4*** of the ***I3C System Integrator's Application Note [MIPI04]***.

Since all such Virtual Targets reside within the same physical unit, the internal distances (i.e., L2 and/or L3) can be expected to be considerably shorter than external distance L1 (i.e., the distance from the I3C Controller to the integrated I3C Device), and the "stub" distances to each individual Virtual Targets are directly controlled by the component designer.

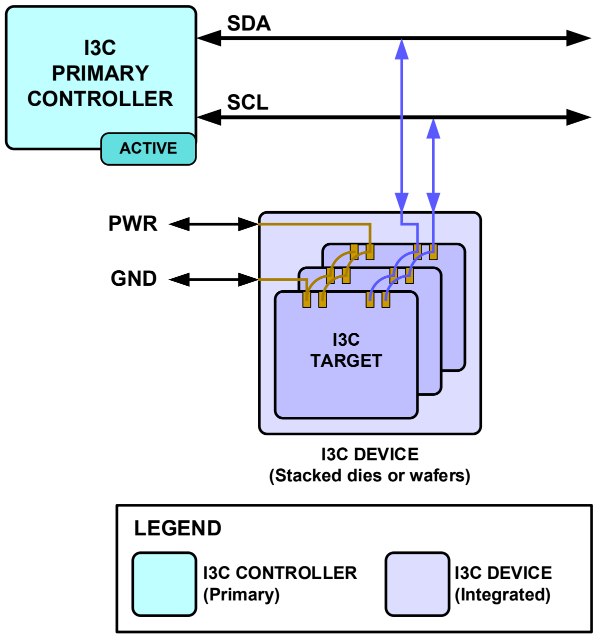
- For integrations that consist of multiple existing silicon layouts (i.e., "hard IPs") that are printed or replicated onto the same combined die, the internal distances and stub properties are directly based on the trace lengths between the pads in the combined die. However, the on-die traces or additional routes might not necessarily be replicated, and in some cases might need to be specially designed and integrated; whereas the "hard IPs" for the I3C Targets will likely be simple to replicate (see ***Figure 2***).
- For integrations that consist of multiple wafers or dies that are printed or layered into a vertical stack, the internal distances and stub properties are a function of the die stacking parameters, which include the spacing distance as well as the type of die interconnect methods (e.g., wire bonding, see ***Figure 3***).
- For integrations that consist of multiple dies within a package that are joined with various die-to-die interconnects that might take the form of bridges, interposers, or simple wires, the internal distances and stub properties are a function of the overall distance across such interconnects, which could be considerably longer than the other two examples above, and depend on the sizes of the dies and the locations of the pads (see ***Figure 4*** and ***Figure 5***).

Figure 2 through ***Figure 5*** illustrate examples of I3C Devices using each of the integration methods mentioned above and in Table 2.



256

Figure 2 Integrated I3C Device with Targets as Hard IPs on a Die



257

Figure 3 Integrated I3C Device with Targets as Stacked Dies or Wafers

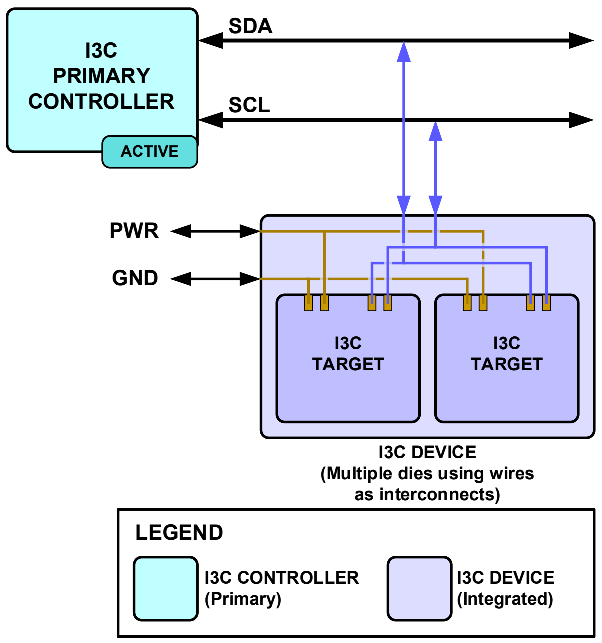


Figure 4 Integrated I3C Device with Targets Connected by Wire Interconnects

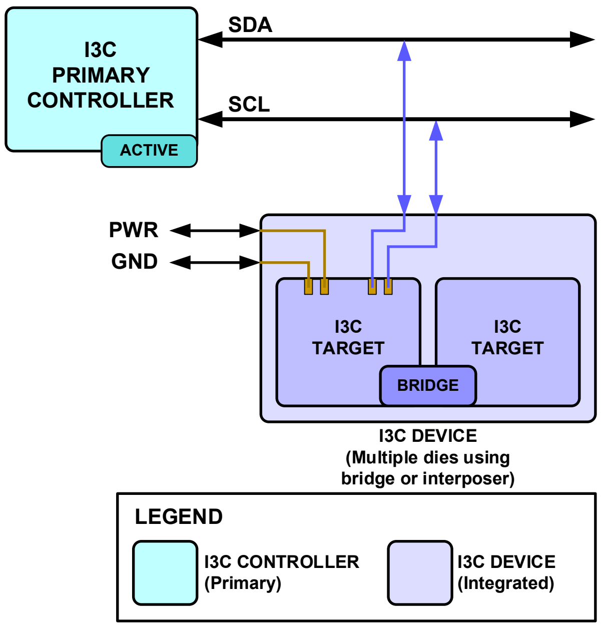


Figure 5 Integrated I3C Device with Targets Connected by Bridge or Interposer

Each of these integration methods offers various advantages, but also poses different challenges as shown in ***Table 2***.

Table 2 Comparison of I3C Device Integration Methods for Virtual Targets

Integration Method	Advantages	Challenges
Hard IPs in Same Dies ***Figure 2***	• Allows reuse of existing verified hardened IPs • On-die traces or routes provide more control over signal integrity	• Could require silicon re-verification based on foundry process • Some hard IPs might not be compatible on same foundry process • Always requires new metal layers for on-die traces or routes between dies and pads
Stacked Dies or Wafers ***Figure 3***	• Works well for horizontally-constrained applications, or other situations where existing dies/wafers have been proven	• Die/wafer thickness impacts vertical stack growth • Possible interactions between individual dies/wafers in Z-axis • Requires planning for die/wafer attachment
Multiple Dies using Wire Interconnects ***Figure 4***	• Simplest integration method for existing I3C Targets • Can unite dies from diverse foundry processes or varying functions	• Differing die sizes could prevent space-efficient integrations (in X-axis or Y-axis) • Wire lengths and stub properties could be longer than with other methods
Multiple Dies using Bridge or Interposer ***Figure 5***	• Well-suited for larger integrations that also integrate other functionality (i.e., other than I3C Targets) and already need package and substrate design	• Some dies might require pads that are suited for bridge or interposer connections, based on package and substrate choices

Note:

If an I3C Bus includes multiple I3C Devices that approach the example topologies or embody the above die design methodologies for integrated Virtual Targets, then the overall topology becomes more complex, i.e., a combination of either "Star-on-Stick" or "Daisy-Chain" for the I3C Devices containing such Virtual Targets. In such cases, the overall I3C Bus topology could be more complex to model. However, the effects of such combined topologies are not expected to be significant, since the internal distances can be expected to be shorter than other methods of system design (i.e., boards or modules) and the "stub" properties can be understood and constrained by the component designer.

5.1.2 Number of Virtual Targets in a Unit

Version 1.0 and version 1.1 of the I3C Specification specified that an I3C Bus could support up to 11 I3C Target Devices, based on typical electrical parameters including trace length and capacitive load for each separate Target Device. However, version 1.1.1 of the I3C Specification removes this limit (see ***[MIPI01]*** at ***Section 4.2.2***) and does not specify a maximum limit of I3C Target Devices. Component designers can calculate and publish the appropriate electrical parameters when designing integrated I3C Devices with multiple Virtual Targets, and System Designers should use these parameters to guide the overall system design of the I3C Bus.

5.1.3 Virtual Target Power Management

If multiple Virtual Targets reside within the same unit, and if each Target could theoretically have been packaged into its own separately I3C Device, then the integrated unit could also have per-Virtual Target power management capabilities, with varying levels of granularity as needed.

For example, the following optional capabilities could be implemented, at the discretion of the component designer:

- Separate power pins for each individual Virtual Target
- Global power pins providing power for all Virtual Targets
- Shared power pins providing power for some Virtual Targets, with other Virtual Targets powered by other pins (i.e., either separate, shared, or global)
- Global/shared power pins to the unit, with power management logic that provides direct power control for one or more individual Virtual Targets, based on out-of-band direct pin inputs (i.e., signals sent from another control unit or Application Host)
- Global/shared power pins to the unit, with power management logic that provides automatic power control for one or more individual Virtual Targets, based on system conditions or Application Host configuration
- Global/shared power pins to the unit, with one Virtual Target providing in-band mechanisms to change the power states for the other Virtual Targets in the same unit, using I3C transactions to control power management logic (i.e., Private Writes/Reads or CCCs)

5.1.4 Bus Configuration Registers

The I3C Specification states that each Virtual Target shall have a Bus Configuration Register (BCR) that accurately describes its own configuration and capabilities (see ***[MIPI01]*** at ***Section 5.1.1.2***).

Since a simple integrated I3C Device contains existing Target logic that has been instantiated with a new integration, each Virtual Target does not necessarily need to report any advanced Virtual Target capabilities using BCR Bit[4]. As a result, each of the Virtual Targets inside an integrated I3C Device will typically set BCR Bit[4] to **1'b0**, unless:

- An individual Virtual Target is a Bridge Device (per version 1.0 of the I3C specification) and supports the SETBRGTGT CCC (see ***[MIPI01]*** at ***Section 5.1.9.3.17***).
- An individual Virtual Target supports advanced Virtual Target capabilities (per version 1.1+ of the I3C Specification), including but not limited to Bridging, Routing, or exposing other Downstream Targets (i.e., advertising as a VT-capable Device, per ***[MIPI01]*** at ***Section 5.1.9.3.19***).

Note:

*The meaning and interpretation of BCR Bit[4] changed between version 1.0 and version 1.1+ of the I3C Specification. Starting with version 1.1+, BCR Bit[4] indicates support for one of several types of Virtual Target capabilities; see **[MIPI01]** at **Section 5.1.1.2.1** for more details. The I3C Controller can determine whether a Virtual Target supports I3C version 1.1+ by reading BCR Bit[5] and by using the GETCAPS CCC, per **[MIPI01]** at **Section 5.1.9.3.19**.*

5.1.5 Dynamic Address Assignment and Provisioned ID Management

If a simple integrated I3C Device contains Virtual Targets that also support Dynamic Address Assignment with ENTDAA, then the component designer must ensure that each Target has been assigned a unique 48-bit Provisioned ID. However, it could become a challenge to ensure that the Provisioned IDs are truly unique across:

- All Virtual Targets within the integrated I3C Device, and
- All I3C Devices in the system (i.e., the entire I3C Bus).

The I3C Specification defines the requirements (see ***[MIPI01]*** at ***Section 5.1.4.1***) for I3C Targets, including the need for a unique 48-bit Provisioned ID for each I3C Target on the I3C Bus. However, the Specification does not specify how to accomplish this, especially for Virtual Targets. This Application Note suggests several methods for ensuring uniqueness, both within the integrated I3C Device and across the I3C Bus that contains such an I3C Device (i.e., one or more instances).

For the 48-bit Provisioned ID values, the following fields are recommended for use by the Virtual Targets within an integrated I3C Device:

- **Bits[31:16]:** Part ID
- **Bits[15:12]:** Instance ID (preferred for this use)
- **Bits[11:0]:** Available for definition by the I3C Target implementer

Below are five strategies that could be used, either separately or together, to set the various portions of the 48-bit Provisioned ID for the Virtual Targets within an integrated I3C Device.

5.1.5.1 Strategy 1

This strategy typically works well for diverse Virtual Targets, or for Virtual Targets that might not all originate from the same MIPI Manufacturer.

Strategy: Set(s) of external pins that provide direct input for some or all of the recommended fields of the 48-bit Provisioned ID, to be directly pulled High or Low by the platform, as either 1’b1 or 1’b0 values.

- Common set of direct pins that apply to all Virtual Targets
- Shared set of direct pins that apply to some Virtual Targets
- Individual set of direct pins that only apply to one Virtual Target

5.1.5.2 Strategy 2

This strategy typically works well for Virtual Targets with similar characteristics, or that are identical in nature and capabilities.

Strategy: Set(s) of external pins that control internal logic within the I3C Device, as indirect configuration inputs recalling “presets” or various stored combinations of some or all of the recommended fields of the 48-bit Provisioned ID. In this case, the internal logic would cause the affected Virtual Target’s field bits to be set to 1’b1 or 1’b0. However, some portions could be unaffected or unchanged.

- Common configuration inputs that affect fields for all Virtual Targets
- Shared configuration inputs that affect fields for some Virtual Targets
- Individual configuration inputs that only affect fields for one Virtual Target

5.1.5.3 Strategy 3

This strategy could be suitable for I3C Devices that typically need additional configuration before initializing the Virtual Targets.

Strategy: Internal logic within the integrated I3C Device to directly set some or all of the recommended fields of the 48-bit Provisioned ID, based on stored configuration (i.e., PROM) or some other configuration received via bus or other channel.

- Can configure some or all Virtual Targets, based on stored or received configuration.

5.1.5.4 Strategy 4

Strategy: Fixed configuration of some of the recommended fields of the 48-bit Provisioned ID, set by internal circuitry.

> ***Note:***
> *This strategy is of course not appropriate for all of such fields in the 48-bit Provisioned ID; obviously, such a decision would restrict an I3C Bus to only one of these I3C Devices that hard-coded its Virtual Targets to have fixed Provisioned IDs and could not be changed by any external means.*

5.1.5.5 Strategy 5

Any combination of strategies 1–4 above.

5.2 Composite I3C Devices

Virtual Targets within an I3C Device can also exist as separate functions or entities that are presented or exposed by a composite I3C Device, using Shared Peripheral logic. The Shared Peripheral logic presents each Virtual Target with its own Dynamic Address, and manages the low-level I3C transactions using shared FIFOs or buffers and a common set of pads (i.e., per-Device GPIOs for SDA and SCL).

While such an approach requires re-design of the logic to implement I3C Target functionality, one key advantage of using a composite I3C Device with Shared Peripheral logic is that the electrical parameters can be more carefully controlled within the composite I3C Device, rather than requiring the serial clock and data signals to be routed within a die, chip, or package as with a Simple Integrated I3C Device (see ***Section 5.1.1***).

5.2.1 Architectural Overview

A composite I3C Device that presents Virtual Targets on the I3C Bus needs to act, work, and function as though it contained multiple I3C Targets, and in all respects other than being a single I3C Device (i.e., with electrical parameters of a single connection to the SCL and SDA lines) such functions must be indistinguishable from the multiple separate but equivalent I3C Target Devices that would otherwise be needed to provide the same functionality.

Note:

*The sole exception to this requirement is that a composite I3C Device that advertises Virtual Target capabilities must support the Virtual Target Detect operation via the RSTACT CCC, as defined in **Section 5.2.2.4** and in the I3C Specification **[MIPI01]** at **Section 5.1.9.3.26**. By contrast, the equivalent separate I3C Target Devices (i.e., non-Virtual Targets) would not support the Virtual Target Detect operation.*

Figure 6 shows an example of a composite I3C Device that contains Shared Peripheral logic (see ***Section 5.2.2***) and presents multiple I3C Virtual Targets on an I3C Bus, in a manner that would otherwise not be seen as such by the I3C Controller.

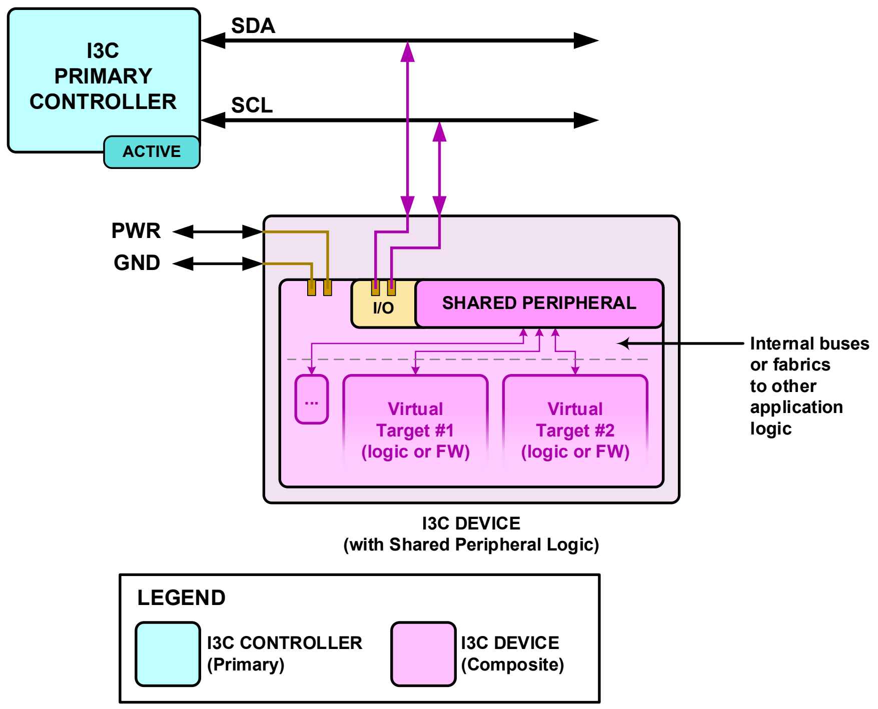

I3C PRIMARY CONTROLLER
ACTIVE
SDA
SCL
PWR
GND
I/O
SHARED PERIPHERAL
Internal buses or fabrics to other application logic
...
Virtual Target #1 (logic or FW)
Virtual Target #2 (logic or FW)
I3C DEVICE (with Shared Peripheral Logic)
LEGEND
I3C CONTROLLER (Primary)
I3C DEVICE (Composite)

Figure 6 Composite I3C Device with Virtual Targets and Shared Peripheral Logic

5.2.2 Shared Peripheral Logic

From an electrical perspective, it is recommended that a composite I3C Device that presents multiple Virtual Targets be implemented with a single set of pins connecting to pads within or adjacent to the Shared Peripheral logic. This would empower the Shared Peripheral logic to respond to transactions on the I3C Bus. The Shared Peripheral logic could contain most of the lower-level components normally associated with a typical generic I3C Target Device (as shown in ***Figure 9*** in version 1.1 of the I3C Specification ***[MIPI02]***) and would also have additional internal state and memory to support the presentation of multiple functions. Each function would enable the Shared Peripheral logic to present a Virtual Target that could respond to CCCs, Private Writes, and Private Reads. In many cases, the Shared Peripheral logic could handle some of the basic CCCs regarding Bus Configuration and Dynamic Address Assignment.

The ***Figure 6*** example shows Shared Peripheral logic within a die or an SoC that also has an internal bus or fabric containing application logic associated with one or more Virtual Targets. This application logic might be entirely implemented in internal logic (i.e., all hardware), or it might also accept configuration in the form of firmware and/or software. Once configured, the application logic would be capable of responding to Private Write transactions sent from the I3C Controller, and preparing data to return via I3C Private Read transactions (in the event that the I3C Controller initiates such a transaction). The Shared Peripheral logic stores the Dynamic Addresses and optional Group Addresses assigned to each of the known Virtual Targets, and uses its internal FIFOs or buffers to manage data for Private Write, Private Read, or CCC transaction requests.

5.2.2.1 Hot-Join and Dynamic Address Assignment

Since the Shared Peripheral logic handles the low-level I3C Bus protocol communications, it must also match the Address Header for any assigned Addresses (i.e., Dynamic Addresses or Group Addresses) and either direct those transactions to the appropriate logic within the composite I3C Device, or else handle the transactions with its own FIFOs or buffers. As a result, it needs to know all of the assigned Addresses for all of the Virtual Targets, and it therefore makes sense for the Shared Peripheral Logic to handle Dynamic Address Assignment for all Virtual Targets within the I3C Device.

If the Shared Peripheral logic does so, then it manages the process of joining the I3C Bus (i.e., sending the Hot-Join Request, per ***[MIPI01]*** at ***Section 5.1.5***) and participating in Dynamic Address Assignment for every Virtual Target it presents or exposes to the I3C Controller.

The Shared Peripheral logic can be designed to present all Virtual Targets as Hot-Joining the I3C Bus with a single Hot-Join Request, with the expectation that they can all receive Dynamic Addresses from the Controller in a single phase (if possible). To the I3C Controller, this would appear as though all Virtual Targets received power and joined the I3C Bus at the same time. Alternately, the Shared Peripheral logic can be designed to emit multiple Hot-Join Requests in various phases, presenting one or more new Virtual Targets on the I3C Bus per phase (i.e., as though some Virtual Targets received power and joined the I3C Bus at a later time than others).

5.2.2.2 BCR Values and GETCAPS CCC Support

Depending on the specifics of the implementation of such a composite I3C Device, some I3C commands sent to one such Virtual Target might have side effects on other Virtual Targets using this Shared Peripheral logic (such as Target Resets). Therefore, it is often important for the I3C Controller to identify such Virtual Targets as having Shared Peripheral logic, and to understand which operations will have impact on other Virtual Targets within the composite I3C Device.

For all Virtual Targets within such a composite I3C Device, the I3C Specification states that each Virtual Target shall have a Bus Configuration Register (BCR) that accurately describes its own configuration and capabilities (see ***[MIPI01]*** at ***Section 5.1.1.2***); and shall also support the GETCAPS Format 2 CCC with Defining Byte VTCAPS (0x93), and return a message of at least one byte (i.e., the VTCAP1 byte, see ***[MIPI01]*** at ***Section 5.1.2.1.2*** and ***Section 5.1.9.3.19***). As such, the following conditions apply:

- In this first byte (VTCAP1), Bits[2:0] have a value of **3'd5** to inform the Controller that this Virtual Target is presented by Shared Peripheral logic.
    - Bit[4] will be set appropriately, to indicate whether certain configuration CCCs (e.g., SETNEWDA, SETMRL) sent to one Virtual Target have side effects on other Virtual Targets connected to the same Shared Peripheral logic (see ***[MIPI01]*** at ***Section 5.1.9.3.19***). If Bit[4] is set to **1'b1**, then the implementer must provide this information in the documentation for such a composite I3C Device, regarding which specific configuration CCCs have these side effects..
    - Bit[5] will be set to **1'b1** to indicate that this Device also supports the Virtual Target Detect operation using the RSTACT CCC, as specified in the I3C Specification (see ***[MIPI01]*** at ***Section 5.1.9.3.26***). If BCR Bit[4] has a value of **1'b1**, then support for this operation is required for all Virtual Target Devices using Shared Peripheral logic within such a composite I3C Device.
- If this Virtual Target returns additional bytes in the message for this CCC, then the format of these bytes (e.g., VTCAP2) has not yet been specified for this Virtual Target type.

It is best for all Virtual Targets within the same composite I3C Device to return the same contents for the GETCAPS Format 2 CCC with Defining Byte VTCAPS.

Additionally, the Shared Peripheral logic might choose to report some or all Virtual Targets as Offline Capable, by setting BCR Bit[3] to 1'b1. This might be useful when simulating power management states, for use cases when a particular Virtual Target needs to block incoming messages and the Shared Peripheral logic chooses to NACK incoming I3C transactions based on the matched Address.

5.2.2.3 RSTACT CCC Support

System Designers working with multiple Virtual Targets with individual Dynamic Addresses using Shared Peripheral logic (or similar designs) are advised to carefully consider the following implications for the RSTACT CCC.

When using the RSTACT CCC as a Direct SET CCC:

- Using the RSTACT CCC with Defining Byte 0x01 to reset a Virtual Target's I3C Peripheral logic might interrupt access to that Virtual Target (i.e., anything exposed via a specific Dynamic Address). Depending on the implementation of the Shared Peripheral, it might preserve enough of its internal state and retain its Dynamic Address for that Virtual Target, despite this interruption; or it might fully reset its internal state for that Virtual Target, after which it will need to participate in Dynamic Address Assignment again. However, it is unlikely that any other Virtual Targets connected to the same Shared Peripheral logic will be affected.
- Using the RSTACT CCC with Defining Byte 0x02 to reset the whole Target (i.e., the whole chip) could reset the entire Device, including all Virtual Targets presented by the shared Peripheral logic.
- If one Virtual Target enables Debug functionality and conforms to the MIPI Debug for I3C Specification ***[MIPI05]***, then it is unlikely that using the RSTACT CCC with Defining Byte 0x03 to reset that Virtual Target's Debug Network Adaptors will interrupt any other Virtual Target functionality or access on the Shared Peripheral logic. Additionally, if the Debug functionality is internally connected to other application logic within the Device (which might be exposed by another application-specific Virtual Target), then that application logic will not be reset or otherwise impacted by a Debug Network Adaptor Reset to the Debug-capable Virtual Target.

When using the RSTACT CCC as a Broadcast CCC:

- The RSTACT CCC with various Defining Bytes will cause an equivalent reset operation to be applied to all Virtual Targets as well as the Shared Peripheral Logic. The specific nature of the reset operation will depend on the Defining Byte. In some cases this might cause all Dynamic Addresses for all Virtual Targets to be reset, after which all Virtual Targets would need to participate in Dynamic Address Assignment again.

Table 3 shows the application of the various Defining Byte values for the RSTACT CCC that can be used with either the Direct SET form or the Broadcast form, and how the implementation of a Virtual Target capable Device should interpret these Defining Byte values. It is derived from ***Table 53*** in the I3C Specification ***[MIPI01]***.

Table 3 Application of RSTACT Defining Byte Values for Virtual Target Capable Devices

Defining Byte Value	Description (for standard I3C Target Device)	Direct SET CCC to a Virtual Target	Broadcast CCC
0x00	**No Reset on Target Reset Pattern**	**No change**	**No change**
0x01	**Reset the I3C Peripheral Only** (Default)	**Reset portions of the Shared Peripheral logic associated with the indicated Virtual Target** Might include a reset of this Dynamic Address	**Reset the entire Shared Peripheral for all Virtual Targets presented by the same Device** Might include a reset of all Dynamic Addresses for all Virtual Targets
0x02	**Reset the Whole Target**	**Reset all logic associated the indicated Virtual Target** Should not impact other Virtual Targets presented by this same Device	**Reset the entire Device** Should impact all Virtual Targets presented by this Device
0x03	**Debug Network Adaptor Reset** The Target shall not reset the I3C Peripheral (per I3C Specification at ***Section 5.1.9.3.26***)	**Reset all Debug Network Adaptors for the indicated Virtual Target** Does not reset any portion of the Shared Peripheral logic	**Reset all Debug Network Adaptors for all Virtual Targets which are Debug-capable** Does not reset any portion of the Shared Peripheral logic, or have impact on any other Virtual Targets presented by the same Device which are not Debug-capable
0x04	**Virtual Target Detect** Documented in this Application Note at ***Section 5.2.2.4***	**Set flag to detect other Virtual Targets presented by the same Device** Does not initiate any reset actions	*No effect, since this Defining Byte is only supported with Direct CCCs*
0x05–0x3F	Reserved by MIPI	*N/A*	*N/A*
0x40–0x7F	Reserved for Vendors and external standards	*N/A*	*N/A*
0x80–0xFF	Not applicable to Direct SET CCC or Broadcast CCC	*N/A*	*N/A*

5.2.2.4 Virtual Target Detect Operation

Although a Device implementer might choose to denote the association between multiple Virtual Targets sharing the same Peripheral logic in a custom manner, a standard method provides for far easier detection and association. Hence, in order to correctly associate all Virtual Targets sharing the same Peripheral logic, such a Device will support the Virtual Target Detect operation described in the I3C Specification ***[MIPI01]*** at ***Section 5.1.9.3.19***. To use this Virtual Target Detect operation, the I3C Controller sends the RSTACT CCC with Defining Bytes 0x04 and 0x84 (see ***Section 5.1.9.3.26***) with any Dynamic Addresses that are configured to any Virtual Targets presented by the composite I3C Devices.

The Virtual Target Detect operation utilizes a single "flag" bit in the Shared Peripheral logic. The Controller can set this flag by using the RSTACT CCC with Defining Byte 0x04 directed to any Dynamic Address assigned to any Virtual Target (with Direct SET CCC) or confirm that the flag was set on the same Virtual Target's Dynamic Address (with Direct GET CCC). Once set, the flag can be tested on any other Virtual Target's Dynamic Address by using the RSTACT CCC with Defining Byte 0x04 (with Direct GET CCC). For Virtual Targets whose shared Peripheral's flag was set, each of them will return a value of 0x01 to indicate that the flag in their shared Peripheral has been previously set.

The Controller can clear the flag in the shared Peripheral by sending the RSTACT CCC with Defining Byte 0x00, either as a Broadcast CCC to all Devices, or as a Direct CCC to all Devices including all Virtual Target capable Devices which support the Virtual Target Detect operation.

Note:

*To minimize the security risks of side-channel attacks, the Shared Peripheral logic can internally handle all Direct CCC accesses to RSTACT with Defining Byte 0x04, without notifying any application or debug logic in any Target-specific implementation which connects to the Shared Peripheral logic. Additionally, the state of this shared flag must not be accessible to any Virtual Target's application logic behind its Shared Peripheral logic; as a result, any messages on the Bus pertaining to setting, confirming, or clearing this shared flag must also not be passed to the Virtual Target's application logic connected to this Shared Peripheral logic. See **[MIPI01]** at **Section 5.1.9.3.26**.*

Since one or more Virtual Target capable Devices might independently connect to an I3C Bus, the following procedure can be used to correctly identify and associate all Virtual Targets with Shared Peripheral logic, using only their Dynamic Addresses and no other prior knowledge.

Detection Procedure

1. Send a Broadcast RSTACT CCC with Defining Byte 0x00, which will clear the detect flags in all Virtual Target capable Devices with Shared Peripheral logic. It will also disable the default Target Reset option (which is acceptable during the Virtual Target detection procedure).
2. Assemble a temporary list L containing all Targets where BCR Bit[4] was previously found to have a value of **1'b1**. If any Targets' BCRs have not yet been read, now is the time to read them and determine the value of Bit[4].
3. Loop over all Targets in list L.

   For each Target T:

   A. Send a Direct GET RSTACT CCC with Defining Byte 0x84 to Target T, to detect whether this Target also supports Defining Byte 0x04.
      - If this request is NACK'd by Target T, then remove Target T from list L and continue.
      - If this request is ACK'd by Target T, but the response is 0x00, then remove Target T from list L and continue.
4. Loop again over all Targets in list L.

   For each Target T:

   A. Send a Direct SET RSTACT CCC with Defining Byte 0x04 to Target T, to set the detect flag in one such I3C Device that is Virtual Target capable, so that it appears in all Virtual Targets sharing its Peripheral logic.
      i. If this request is NACK'd by the Target T, then remove Target T from list L, and go back to step 4.
      ii. If this request is ACK'd by the Target T, then send a Direct GET RSTACT CCC with Defining Byte 0x04 to the same Target T, and confirm that it returns a value of 0x01.
         - If it does not return 0x01, then the flag was not set. Retry from step 4.A. If this fails again, then remove Target T from list L, and go back to step 4.
         - If it does return 0x01, then the flag was set.
      iii. Loop over all other Targets in list L (not including Target T).

         For each Target P:

         1. Send a Direct GET RSTACT CCC with Defining Byte 0x04 to Target P, to read the detect flag in its Peripheral logic.
            - If this request is NACK'd by Target P, then continue to the next device.
            - If this request is ACK'd by Target P and the value is 0x01, then Target P and Target T must share the same Peripheral logic. Create an association between them (i.e., mark Target P as a counterpart to Target T), remove Target P from list L, and continue to the next device.
            - If this request is ACK'd by Target P but the value is 0x00, then Target P and Target T must not share the same Peripheral logic. Continue to the next device.
      iv. At the end of the loop, record all found associations with Target T.
      v. Send another Broadcast RSTACT CCC with Defining Byte 0x00, to clear the detect flags in all VS-capable Devices with Peripheral logic.
      vi. Remove Target T from list L.
5. Any Targets remaining in list L should be handled as exceptions, i.e., as Virtual Targets with no corresponding counterparts.

By following this procedure, the Host system which connects to the Controller can associate all Virtual Targets with Shared Peripheral logic, as linked to their counterpart Virtual Targets. As soon as any Target is found to have its detect flag set (i.e., by the Direct GET CCC returning a known value), its association with another Virtual Target (i.e., the Target that had its detect flag set by the Direct SET CCC) is recorded, and it is removed from the temporary List (to shorten processing time).

5.2.3 Enabling Virtual Target Transactions

If the Shared Peripheral logic is not able to fulfill a particular transaction request immediately, but is configured to accept such a transaction request on behalf of a Virtual Target, then the Shared Peripheral Logic can steer each transaction request to the appropriate Virtual Target.

This Virtual Target might exist in one of several forms:

- Separate internal application logic, per Virtual Function;
- A unique virtual "endpoint" per Virtual Function, that is presented based on internal configuration of configurable application logic or other programmable elements; or
- Configuration and/or data received from other external buses, fabrics, or execution units within the die, SoC, or system (i.e., platform).

The Shared Peripheral Logic steers the transactions based on the matched Address in the Address Header (i.e., the Dynamic Address or Group Address assigned to a Virtual Target), and transactions are handled appropriately based on transaction type:

- **For Write-type transfers,** the Shared Peripheral will accept (i.e., will ACK) the Write and then receive data bytes on behalf of the Virtual Target; these data bytes will be queued in a FIFO or buffer, and transferred to the application logic when appropriate.
- **For Read-type transfers,** the Shared Peripheral could either:
  - Pre-cache response data from the application logic, for one or more expected Read-type transfers having certain parameters, and then be ready to ACK a Read-type transfer request that matches those same parameters; or
  - Initially reject (i.e., NACK) any Read-type transfer request for which it does not have data ready to respond, and then initiate an internal request to the application logic to provide response data for a subsequent Read-type transfer having the same parameters, with the expectation that the I3C Controller would retry the transfer request, in anticipation of a successful result (i.e., an ACK on the retry)

5.2.3.1 Special Considerations for HDR Modes and Multi-Lane Transfers

HDR Modes

If a composite I3C Device with Shared Peripheral logic also supports any of the optional HDR Modes (per ***[MIPI01]*** at ***Section 5.2***), then these could be mapped into additional higher-level transactions as HDR WRITE or HDR READ commands, which can be steered to the specific logic within the composite Device for a given Virtual Target, based on the matched Address in the HDR structured protocol element (i.e., Command Word or Header Block).

Note:

> *HDR Modes also support additional parameters and metadata per transaction request, such as HDR Command Bytes (per HDR Mode). As a result, the internal configuration of the Shared Peripheral logic must appropriately handle valid transaction requests with different HDR Command Byte values and other parameters, and must only ACK such a transaction if it can respond to such a supported transaction for which it is ready to send or receive data.*
>
> *If the Shared Peripheral logic receives a transfer with an HDR Command Byte or other parameters that are not supported, then the Shared Peripheral logic must NACK the transaction request.*

Multi-Lane Transfers

If a composite I3C Device with Shared Peripheral logic also supports Multi-Lane transfers, then it will use the correct Multi-Lane configuration (i.e., for supported I3C Modes) based on the Dynamic Address or Group Address for each transfer in that I3C Mode.

- **In HDR-DDR and HDR-TSP Modes:** The current Multi-Lane configuration does not affect the way that the Shared Peripheral logic must parse the Address field of every transaction, since all Data Transfer Codings use SDA[0] only to transmit the Address for the HDR Command Word for Multi-Lane transfers (i.e., the same as for single-Lane transfers in the same HDR Mode).
- **In HDR-BT Mode:** The current Multi-Lane configuration affects the way that the Shared Peripheral logic must parse the Address field of every transaction, per the I3C Specification (see ***[MIPI01]*** at ***Section 5.3.2.4***). This is a consequence of selecting a Data Transfer Coding that uses the additional Data Lanes (i.e., SDA[1–3]) since HDR-BT Mode defines a different bit packing format for these alternate Data Transfer Codings, and this changes how the Shared Peripheral logic must parse the Address field of the HDR-BT Header Block.
  - If such alternate Data Transfer Codings are used, then the I3C Controller and all HDR-BT capable I3C Devices must agree to use the same interpretation and bit packing format for HDR-BT Header Blocksin order to correctly parse the Address field of every transaction in HDR-BT Mode. If this agreement is not achieved, then communication errors are likely to happen.
  - If an HDR-BT capable composite I3C Device cannot support (or is not configured to support) such an alternate Data Transfer Coding, or if there is a mismatch between the Data Transfer Coding that the I3C Controller uses and the Data Transfer Coding that such a composite I3C Device is configured to expect, then there will be communication errors on the I3C Bus. For example, the composite I3C Device will likely misunderstand the data that the I3C Controller drives on SDA[0] and any additional Data Lanes (i.e., SDA[1–3]) during the HDR-BT Header Block. This will likely lead to incorrectly matched Addresses or mistaken ACKs at the wrong time, for Address values that were interpreted incorrectly (i.e., differently than the I3C Controller intended to drive) across any of the Data Lanes.

6 Connecting to Other Buses via I3C Virtual Targets

6.1 Common Concepts

Bridge Devices (***Section 6.2***) and Routing Devices (***Section 6.3***) share many similarities: they present one or more Virtual Target devices to an I3C Bus, and they handle transactions to and from these Downstream targets using assigned Dynamic Addresses.

For Bridge Devices, the Downstream targets are not on an I3C Bus, but the Bridge Device presents its bridged endpoints to the Upstream I3C Bu as individual Virtual Targets. Each bridged endpoint has its own Dynamic Address and can be addressed directly via Private Write/Read transfers.

- The Downstream targets might connect via some other multi-drop bus, such as I^{2}C or SMbus, and might actually be individual devices on that bus, for which there would be a 1-to-1 mapping of Dynamic Addresses.
- Alternately, the Downstream targets might connect via another interface, such as SPI or UART, or might use another signaling method implemented using an interface (e.g., GPIOs) in the Bridge Device. For such a case, there might not always be multiple distinct targets on that other interface, but their presence might be simulated or virtualized by the Bridge Device, based on the nature of the communication protocol.
- For more details about Bridge Devices, see ***Section 6.2***.

For Routing Devices, the Downstream targets represent other I3C Bus Segments, and the Routing Device typically acts as a Controller for those segments.

- The Routing Device acts as a router that forwards Private Write transfers from the Upstream I3C Bus Segment, to send the data payload to I3C Devices on the Downstream I3C Bus Segment. The Routing device also performs the reverse for any corresponding data that will be read from I3C Devices on the Downstream segment, typically taking the form of a Private Read transfer on the Upstream segment.
- A Routing Device can route transactions for one or more Downstream I3C Bus Segments, and handles communications across the I3C Bus Segments. In order to determine which of the I3C Devices on the Downstream segment to address, the I3C content protocol for the Routing Device relies on some method of encapsulating or framing the Private Write and Private Read transfers. This typically takes a structured format that includes the identifier or I3C Address for specific I3C Devices on that Downstream segment.
- However, the I3C Targets on the Downstream segments are not directly presented to the Upstream segment (i.e., a different Dynamic Address is presented), and the Routing Device must store and forward each transaction. The Routing Device can also use an In-Band Interrupt (IBI) Notification on the Upstream segment to notify the I3C Controller that data is ready to be consumed for a Private Read transaction from the Downstream segment.
- For more details about Routing Devices, see ***Section 6.3***.

Additionally, I3C also enables new use cases for specialized Hub Devices. Hub Devices provide some measure of electrical isolation between the Upstream I3C Bus Segment and the other I3C Target Devices on a Downstream I3C Bus Segment.

- Such a Hub Device might also present one or more Virtual Targets, if it allows for advanced management of the circuitry that provides isolation, or if it manages the flow of transactions on one or more Downstream segments. The Virtual Target might accept commands using CCCs and/or Private Write/Read transfers to configure the Hub Device.
- For more details about Hub Devices that use Virtual Targets, see ***Section 6.4***.

6.1.1 Manager Function

Bridge Devices, Routing Devices, and specialized Hub Devices will provide at least one Virtual Target as a Manager Function. This Manager Function allows the I3C Controller on the Upstream Bus segment to configure the Device, to expose its capabilities, and optionally to configure its connections to Downstream targets, bridged endpoints, or other types of Bus segments, depending on the Device type.

Common capabilities of the Manager Function for such Devices will typically include:

- Support for Dynamic Address Assignment with ENTDAA, using a unique 48-bit Provisioned ID (i.e., as with any other typical I3C Target on the Upstream I3C Bus Segment)
- Bus Configuration Register (BCR) and Device Configuration Register (DCR) values that expose Virtual Target capabilities, because this Device presents or exposes other Downstream Virtual Targets (depending on its type)
- Presentation of the Manager Function as its own Virtual Function for configuration or control, in addition to any other Virtual Targets based on functions that the Device might present (per its type).
- Peripheral logic that can accept and handle I3C transfers, including Private Write/Read transfers and CCCs, to handle configuration and management via this Dynamic Address
  - If the Device supports I3C version 1.1 or greater, then this includes the GETCAPS Format 2 CCC with Defining Byte VTCAPS (see ***[MIPI01]*** at ***Section 5.1.9.3.19***)
  - This might also include other required CCCs, such as RSTACT
- Additional advanced logic that enables specific capabilities, and presents other Virtual Targets per the use case (i.e., Bridging, Routing, or Hub)

6.1.2 Configuration of Downstream Virtual Targets

Bridge Devices and Routing Devices might automatically configure their Downstream targets or endpoints. However, Hub Devices as described in this Application Note are not required to automatically configure their Downstream I3C Targets.

Note:

Configuring such Targets or endpoints does not always mean they must be automatically exposed or presented on the Upstream I3C Bus Segment. For example, some Bridge Devices might configure their bridged endpoints with fixed configuration or directly-applied settings, but still rely on the I3C Controller to use the SETBRGTGT CCC to enable such bridged endpoints to appear on the I3C Bus as unique Virtual Targets with unique Dynamic Addresses. However, most internal configuration tasks are usually best left under the direct control of the Bridge Device itself.

For Bridge Devices and Routing Devices that automatically configure any Downstream targets, the Manager Function could indicate this capability in the VTCAP2 byte (as returned from GETCAPS Format 2 CCC with Defining Byte VTCAPS). If the Manager Function does not support a two-byte message for this CCC, then the I3C Controller can assume that automatic configuration is the default.

- For Bridge Devices, each bridged endpoint is a separate Virtual Target. It is essential that the method for configuring a Downstream target be either built-in, configured directly to the Bridge Device, or otherwise derived from the configuration sent by the Controller of the Upstream I3C Bus Segment (i.e., the same Bus Configuration CCCs). Additional configuration steps might be necessary, based on the specific Downstream bus or interface, as well as the Downstream Target or endpoint that is being presented as a Virtual Device.

  This can be remapped into a format which is suitable for the type of Downstream interface, to the extent that such CCCs can be mapped into the set of configuration commands which are possible. However, this might not always be applicable for all such CCCs.

  - In some cases, the Bridge Device might need to perform automatic configuration of some modes or request types, or might need to cache certain types of downstream segment events that have not been configured, or that do not have a clean analogue to I3C configuration CCCs. The

Bridge Device typically defines and implements a method for mapping these events or conditions into Private Read/Write transfers or In-Band Interrupts, but this is left to the discretion of the implementer.

- For Routing Devices, each Downstream target (i.e., an I3C Target Device connected to a Downstream I3C Bus Segment) must be configured on that I3C Bus Segment.

  For most applications, this configuration will be derived from any Bus Configuration CCCs (e.g., ENEC, DISEC, or SETBUSCON) that the Controller might have previously broadcast to all Devices on the Upstream I3C Bus Segment, or that the Controller might have sent directly to this Routing Device. For special use cases, the Controller might send special configuration to the Routing Device, or the Routing Device might have internal configuration for its Downstream Bus configuration.

  - The Routing Device could cache this configuration that might have been received from the Controller; if so, it would replay it on any Downstream segments as needed, in order to configure new Downstream I3C Target Devices that appear (or that Hot-Join, if Hot-Join is supported). However, certain use cases involving the SETBUSCON CCC could rely on repeated Broadcasts with different context bytes and optional data (as defined per the context byte) that must be emitted in the correct order. (Refer to the SETBUSCON CCC, see ***[MIPI01]*** at ***Section 5.1.9.3.31***, "Layered Protocol Contexts".) Such a Routing Device must be present on the Bus to receive these Broadcasts, and upon receiving them it must cache them in the correct order, based on the original intent as received from the Controller of the Upstream I3C Bus Segment.
  - Alternately, the Routing Device could simply pass along some or all of these Bus Configuration CCCs to downstream I3C Target Devices, as soon as they are sent by the Upstream I3C Bus Segment's Controller. In that case, the Routing Device can request the I3C Controller to re-send all active and current configuration (i.e., what is most current) without relying on an internal cache. However, the Routing Device might need to filter or block certain types of requests from downstream Devices as they "join" the Bus, until the requested configuration has been received and re-sent.

Routing Devices that do not automatically configure any Downstream targets and that require the I3C Controller on the Upstream I3C Bus Segment to configure every Downstream I3C Target Device are also possible. For such Bus configurations, the Routing Device must transparently pass all Bus Configuration CCCs to each new Downstream I3C Target Device once the I3C Controller is made aware of its presence on the I3C Bus. The I3C Controller must also be prepared to repeatedly send these Bus Configuration CCCs when a new Downstream target (presented by the Routing Device as a Virtual Target) is observed to join the I3C Bus.

Bridge Devices that do not automatically configure any Downstream bridged endpoints as Virtual Targets are possible, but this option is not always recommended because the Bridge Device must rely on the I3C Controller (i.e., in the Upstream I3C Bus Segment) to have perfect knowledge of the configuration of all such bridged endpoints.

6.2 Bridge Devices

The MIPI I3C Specification ***[MIPI01]*** covers inter-bus bridging support, both passively and actively. In all cases, the bridged Targets (i.e., the Devices that are being bridged) are presented as Virtual Target devices, each with its own I3C Dynamic Address. The bridged Targets can transact with the I3C Bus through the Bridge Device.

6.2.1 Architectural Overview

In many aspects, a Bridge Device is similar in concept to a composite I3C Device (see ***Section 5.2.1***) that uses Peripheral logic to present its Virtual Targets and interacts with its Upstream I3C Bus through its additional capabilities (i.e., capabilities over and above those required of a simple I3C Target).

Figure 7 shows an example of an I3C Bridge Device that presents multiple I3C Virtual Targets on an I3C Bus, where each Virtual Target is a representation or abstraction of one or more bridged endpoints on an external Bus. In this example, the Bridge Device also has a Bridge Manager function as the primary Virtual Target, used by the I3C Controller (i.e., on its Upstream I3C Bus Segment) to configure and manage the Bridge Device and its presentation of the bridged Targets, the associated bridged Endpoints, and optionally any controllable parameters associated with its external I/O to its Downstream bus or interface.

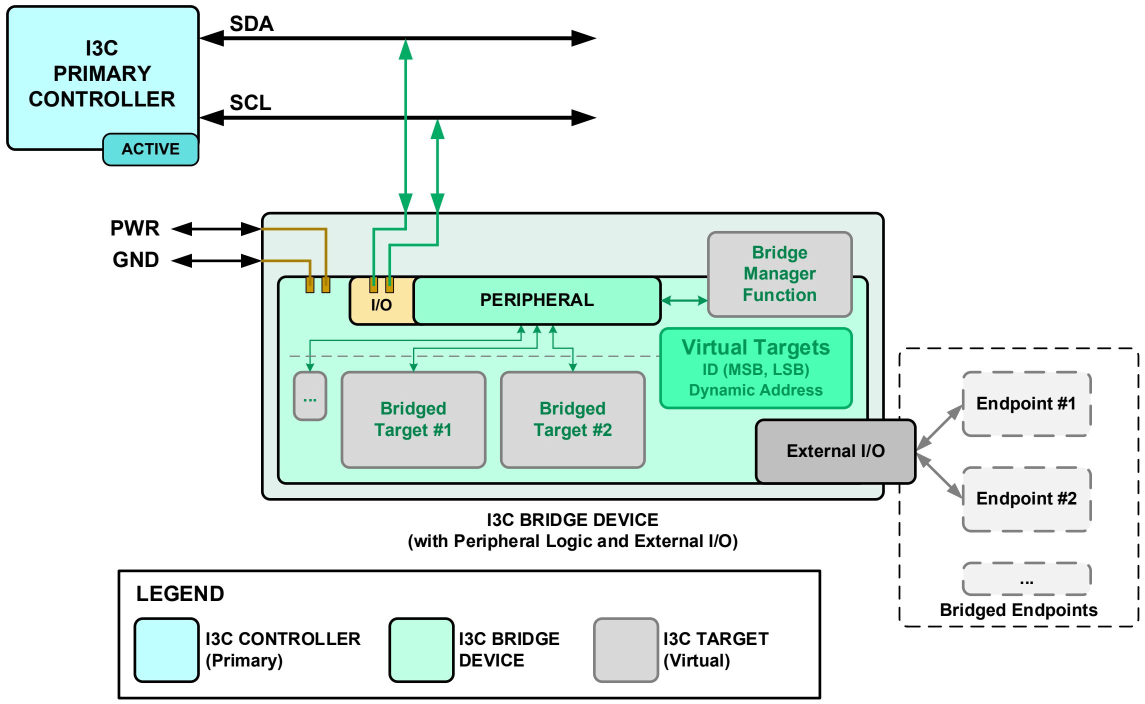


Figure 7 I3C Bridge Device with Virtual Targets and Bridged Endpoints

6.2.2 Bridge Manager Function

Bridge Devices present at least one Bridge Manager Function as a Virtual Target. A Bridge Manager Function exposes the Device's presence on the I3C Bus and describes its capabilities and optional features (see ***Section 6.1.1***).

Note:

> *Previous versions of this Application Note used the term "Control Function" to describe the Virtual Target that provides management and control of the Bridge Device. This term has been replaced with "Manager Function" to avoid confusion with the new normative term "Controller", see **Section 2.1**.*

If such Devices are fully compliant with version 1.1+ of the I3C Specification ***[MIPI01][MIPI02]***, then they must report this capability in the Bus Configuration Register (BCR) for the Manager Function's Virtual Target. Specifically, Bit[4] of the BCR for the Manager Function must indicate that it is a Virtual Target (see ***[MIPI01]*** at ***Section 5.1.1.2.1***). Such Targets also typically support the GETCAPS Format 2 CCC with Defining Byte VTCAPS (see ***[MIPI01]*** at ***Section 5.1.9.3.19***) to describe more details regarding the capabilities and features of Bridge Devices.

- Manager Functions for Bridge Devices will support the GETCAPS Format 2 CCC with Defining Byte VTCAPS, and return a message of at least one byte, but preferably two bytes (i.e., both the VTCAP1 byte and the VTCAP2 byte).
  - In the first byte (VTCAP1), Bits[2:0] will have a value of either **3'd1** or **3'd2**, depending on whether the I3C Controller must use the SETBRGTGT CCC to configure the bridged targets as Virtual Targets (see ***Section 6.2.3.1***), vs. the Bridge Device will configure its bridged targets automatically (see ***Section 6.2.3.2***).
  - In the second byte (VTCAP2), the following fields will be set to reflect a Bridge Device:
    - Bits[1:0] will have either a value of **2'd2** if the Bridge Device can send an IBI for each bridged target with its Target Address, or a value of **2'd0** if the Bridge Device does not support IBIs from any bridged targets.

    Note:

    > *A value of **2'd1** is not recommended for most Bridge Device implementations.*

    - Bit[2] must have a value of **1'b1**. (A value of **1'b0** does not make sense for Bridge Devices.)
    - Bits[4:3] will generally have a value of **2'd0**, unless the Bridge Device requires more specific configuration of its bridged targets with the help of Broadcast CCCs (such as SETBUSCON, ENEC, or DISEC) and cannot accomplish the task without repeated reminders from the I3C Controller.

6.2.3 Initialization and Configuration

There are two general ways an I3C Bridge Device can get the bridged Virtual Targets configured with their I3C Dynamic Addresses, and presented using its bridged endpoints:

1. The Host's software or firmware 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 and its assigned Dynamic Address.
   A. For Bridge Devices that comply with version 1.1+ of the I3C Specification, such a Bridge Device exposes a Manager Function which advertises Virtual Target capabilities in its BCR (see ***[MIPI01]*** at ***Section 5.1.1.2.1***) and indicates a Virtual Target Type **3'd1** in its VTCAP1 data byte (see ***[MIPI01]*** at ***Section 5.1.9.3.19***).
   B. To configure the bridged Downstream Devices, the Controller sends the SETBRGTGT CCC to the Manager Function (see ***Section 6.2.3.1***).
2. The Bridge Device itself knows what is attached, and uses the ENTDAA CCC to represent each one separately to the Controller, so that each bridged Target is assigned a unique Dynamic Address.
   A. For Bridge Devices that comply with version 1.1+ of the I3C specification, such a Bridge Device exposes a Manager Function which advertises Virtual Target capabilities in its BCR (see ***[MIPI01]*** at ***Section 5.1.1.2.1***) and indicates a Virtual Target Type **3'd2** in its VTCAP1 data byte (see ***[MIPI01]*** at ***Section 5.1.9.3.19***).
   B. The Controller does not need to explicitly configure the bridged Downstream Devices using the SETBRGTGT CCC.

Note:

- *The Manager Function for 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 Controller and the Bridge Device, and the model would likely be similar to the effect of the SETBRGTGT CCC: the Controller allocates a set of Dynamic Addresses. It could also be the case that the Controller programs the Bridge Device without searching for its initial bridged Targets (i.e., based on configuration received from the Host's higher-level software) and then uses the ENTDAA CCC again to learn each target, whether by Controller decision or as the result of a subsequent Hot-Join Request.*
- *Bridging in the other direction (i.e., from some other bus into an I3C Bus) would require a full I3C Controller implementation (i.e., a Secondary Controller), and would also require the I3C Active Controller to pass the Controller Role to the Bridge Device for some period of time. As a result, that would work the same as with any I3C Controller-capable Device.*

6.2.3.1 Use of SETBRGTGT (Set Bridge Targets)

An I3C Bridge Device might require use of the SETBRGTGT CCC (see I3C Specification, ***Section 5.1.9.3.17***) to configure the bridged Downstream Targets so they can be used on the I3C Bus.

The steps to use the SETBRGTGT CCC are as follows:

1. The System Designer implants knowledge about the bridged endpoint's Virtual Targets in the Primary Controller's firmware; this might be either an I3C HCI compliant Bus Controller, or an I3C Bus Controller with a more direct Host interface.
   A. This can be a table, such as is used to capture knowledge about other I3C (and any legacy I^2C) Targets, 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 Manager Function of the I3C Bridge Device participates in the normal I3C Dynamic Address Assignment procedure, whether at Bus initialization or as the result of a Hot-Join Request.
   A. This means that the Manager Function might have a static address and either SETDASA or SETAASA CCC would be used, or it might only respond to the ENTDAA CCC.
      - If Hot-Join is supported, then ENTDAA support is also required and the Manager Function must also have an assigned 48-bit Provisioned ID (PID) and support the GETPID CCC (see ***[MIPI01]*** at ***Section 5.1.9.3.12***).
   B. The Manager Function's DCR value indicates its capabilities as a Virtual Target for a Bridge Device, and its BCR value is that of a typical I3C Target.
      - For Bridge Devices that comply with version 1.1 of the I3C specification, the Manager Function's BCR must have Bit[4] set to indicate Virtual Target capabilities (see ***[MIPI01]*** at ***Section 5.1.1.2.1***), because the Bridge Device exposes other Downstream Devices.
   C. If Dynamic Address Assignment with ENTDAA is supported, then the Manager Function will have a normal MIPI Manufacturer ID and 48-bit PID (per ***[MIPI01]*** at ***Section 5.1.4.1***).
      - If more than one physical Bridge Device of the same manufacturer and type (i.e., part ID) can be used on the same I3C Bus, then these must be unique on the I3C Bus (as is the case for all I3C Targets), and the instance ID portion of the 48-bit PID can be used to differentiate them.
      - The Manager Function must also support the GETPID CCC (see ***[MIPI01]*** at ***Section 5.1.9.3.12***).
   D. The Manager Function's assigned Dynamic Address is then used only for communication between the Controller and the Manager Function itself.
      - For some use cases, the Controller might never use this Dynamic Address after initial configuration of the bridged Targets. For other use cases, this Dynamic Address might subsequently be used for such purposes as re-configuration, error handling, etc.

3. After assigning a Dynamic Address to the Manager Function, the Controller will then use the SETBRGTGT CCC with the Manager Function to configure the other Virtual Targets for the Bridge Device’s bridged endpoints:
    A. The Bridge Device will assign a unique Dynamic Address for each target or endpoint that is being bridged. It will use the ID[15:0] field of its 48-bit Provisioned ID to provide info for each new Virtual Target for a bridged endpoint, as needed.
        - Since the Controller and Bridge device already know about the bridged endpoints, and have shared understanding of the Dynamic Addresses that the Virtual Targets will use, this Dynamic Address assignment process does not rely on the ENTDAA CCC.
    B. The Bridge Device will likely set the BCR ‘limitations’ bit for these Virtual Targets. This allows the Controller to interrogate these Virtual Targets to determine their maximum data transfer limits, their read turnaround time delays, and to discover any other consideration(s).
    C. The Bridge Device can generate an IBI that originates from such a Virtual Target’s Dynamic Address using the Peripheral logic. The Controller can use the GETBCR CCC during configuration, to read how such a Bridge Device handles the IBI, unless this is already known from the system designer.
    D. For Bridge Devices that comply with version 1.1 of the I3C specification, the BCR for each Virtual Target of a bridged endpoint will have Bit[4] set because the Virtual Target is a Downstream Device that is being presented by the Bridge Device’s Peripheral logic.

6.2.3.2 Use of ENTDAA to Discover the Bridged Targets

Unlike the steps for configuring the bridged endpoints of a Bridge Device that is already known to the Controller (i.e., one that knows about what it presents on the I3C Bus), the steps for configuring the bridged endpoints of a Bridge Device that is detected during I3C Bus enumeration is quite different than the SETBRGTGT CCC mechanism (see ***Section 6.2.3.1***). In this case, the Controller does not need to know anything about the Bridge Device or its bridged endpoints 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.

   The Bridge Device might only know the bus format (e.g., I^{2}C, SPI, UART, etc.) for each Target, or it might know what type of device each Target is as well as other data about the specific endpoints. The Bridge Device might know this configuration information from pin straps, from pin testing at reset, from programmed fuses or NVMEM, or from some other means.

   Note:

   > *It is possible for the Controller to configure the Bridge Device in-band, via Private Write transfers. The end result is the same: the Bridge Device receives its configuration regarding its bridged endpoints. This works similarly to the other specific methods, and so is not discussed separately here. The only notable aspect is that for the Controller to do this, the Bridge Device itself must have an assigned Dynamic Address, and the Controller must determine that the Bridge Device requires configuration data, before sending the configuration data via I3C Private Write transfers. The Bridge Manager Function is responsible for receiving and applying this configuration data, before presenting the bridged endpoints as new Virtual Targets.*

2. During Dynamic Address Assignment with ENTDAA, the Bridge Device presents each bridged Target as a separate and unique Virtual Target.

   This might happen during Bus Initialization, or it might happen later (i.e., after Bus Initialization using a Hot-Join Request).

   - The Peripheral logic might reuse some or all of the Bridge Manager Function’s Manufacturer ID, PID, BCR, and DCR. Alternately, the Peripheral logic might present these as entirely new (or mostly new) values for each new Virtual Function.

     In either case, each new Virtual Function for a bridged endpoint must still appear to be unique, for the Provisional ID and DCR values, across all other Targets within this Bridge Device and the entire I3C Bus.

   - The Controller will assign each such Virtual Target a unique Dynamic Address.
   - The Bridge Device’s Manager Function can also represent itself with a unique Provisional ID and DCR if it needs or wants; this could also have been done in a previous round of assignment, if the Controller programs it by private contract.
   - In general, the BCR values will describe the Bridge Device and bridging capabilities, per ***[MIPI01]*** at ***Section 5.1.1.2.1***.

     For example:

     - Each Virtual Target for a bridged endpoint will indicate a ‘limitation’ using BCR Bit[0]. This will ensure that the Controller makes requests such as GETMXDS, GETMWL and GETMRL, etc.
     - If a Virtual Target supports IBIs, then it will report this capability using BCR Bit[1]. If IBI data payloads are also supported, then this shall be reported using BCR Bit[2].

   - Each Virtual Target will also report support for other optional capabilities using the GETCAPS CCC, including support for HDR Mode transfers (see***[MIPI01]*** at ***Section 5.1.9.3.19***).

6.2.4 Bridged Endpoints Exposed as Virtual Targets

Each Downstream bridged endpoint exposed by the Bridge Device is also a *de facto* Virtual Target which can be treated as a normal I3C Target, though perhaps with some limitations required by bridging:

1. Virtual Targets for slower targets (such as I^{2}C Devices) or endpoints on a slower external Downstream interface will obviously need more time for write-read turnaround.

   This delay or turnaround time can typically be returned via the GETMXDS CCC.
2. Some Virtual Targets and/or the Bridge Device will have maximum write data limits. These can be adjusted using the SETMWL and GETMWL CCCs, if supported.
3. The Bridge Device could NACK writes to a Downstream target if its buffers are full.
4. The Bridge Device could likely limit its support for advanced and optional features such as transfers in HDR Modes, specialized CCCs, and other uses that might not translate well to bridging.
5. For Bridge Devices that comply with version 1.1 of the I3C specification, the Virtual Targets presented by the Bridge Device need not support the GETCAPS Format 2 CCC with Defining Byte VTCAPS.

   However, for some use cases, it might be advisable to include this support and present these Virtual Targets as having Shared Peripheral logic (i.e., Virtual Target Type **3'd5**). This could be useful for Bridge Devices that indicate a Virtual Target Type **3'd2** for the Manager Function, since the Controller might not otherwise have knowledge of which Virtual Targets are associated with this Bridge Device.

The method for determining which bridged endpoints on the Downstream bus or interface to expose is left to the designer of the Bridge Device, and this could depend on the type of bus or interface that is being bridged. In general there could be a 1-to-1 mapping for each Downstream target device to a particular Dynamic Address, but the specifics will depend on the use case.

Note:

For certain use cases, the Virtual Target for a Downstream bridged endpoint might in fact be a virtualized function, or it might map to other resources that do not directly correspond with a unique device on the Downstream bus or interface, so the Dynamic Address could represent a notional entity or a composite endpoint.

6.2.5 Bridged Devices and IBI

The Bridge Device can use an IBI when a bridged endpoint sends a signal, such as pulling a GPIO of the Bridge Device, to notify that it has new data (or, in the case of UART, that it has already sent new data). The IBI could also be used for errors, such as a buffer overrun. The Controller and Target must have some agreement on its use.

The Bridge Device could choose to support I3C Time Control, such as Asynchronous Time Control, which would allow the Bridge Device to record the time when the GPIO was pulled (or when data was written to the UART) in order to provide that timestamp data to the Controller.

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

If IBIs are used, then the Bridge Device might opt to pre-emptively read the new data from a bridged endpoint, store the data in an internal buffer, and then signal the Upstream Controller with the IBI. The Bridge Device might also choose to support Pending Read Notifications (see ***[MIPI01]*** at ***Section 5.1.6.2.2***) when signaling the presence of newly-buffered data from a bridged endpoint.

6.3 Routing Devices

The MIPI I3C Specification ***[MIPI01]*** also defines simple CCC support for I3C routing, to enable Routing Devices that can store and forward Private Write/Read transactions from a main I3C Bus Segment to I3C Target Devices on one or more Downstream I3C Bus Segments. All such segments have independent parameters and different sets of Dynamic Addresses, but still enable communication via the Routing Device.

6.3.1 Architectural Overview

A Routing Device shares some key features and capabilities of a Composite I3C Device (see ***Section 5.2.1***) that uses Peripheral logic to present its Virtual Targets, and that interacts with its Upstream I3C Bus through its additional capabilities (i.e., capabilities over and above those required of a Simple I3C Target).

Figure 8 shows an example of an I3C Routing Device that presents multiple I3C Virtual Targets on an I3C Bus, where each Virtual Target is a Routing Target Function that provides a Read/Write interface to its corresponding Downstream I3C Bus Segment. Each Routing Target Function has an assigned Dynamic Address that serves as the access point for interfacing with a set of Transfer Queues. The Transfer Queues are accessed by a dedicated I3C Bus Controller for the Downstream I3C Bus Segment, which drives the Private Write/Read transactions to the indicated I3C Target Devices on its segment. In this example, the I3C Routing Device also has a Routing Manager function (per ***Section 6.1.1***) as the primary Virtual Target, used by the I3C Controller on its Upstream I3C Bus Segment to configure and manage the Routing Device and its presentation of the Downstream I3C Bus Segments.

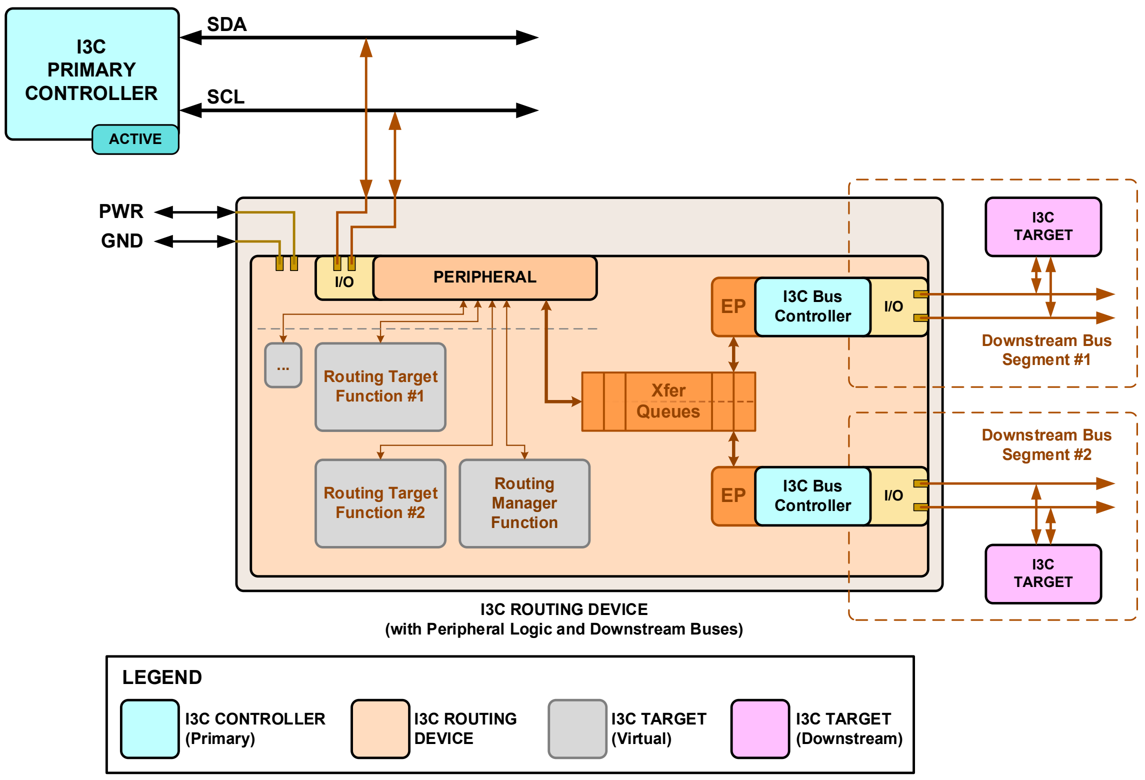


Figure 8 I3C Routing Device with Downstream I3C Buses as Virtual Targets

6.3.2 Routing Manager Function

In general, Routing Devices present at least one Routing Manager Function as a Virtual Target. A Routing Manager Function exposes the Device's presence on the I3C Bus (i.e., the Upstream segment) and describes its capabilities and optional features (see ***Section 6.1.1***).

If such Devices are fully compliant with version 1.1+ of the I3C Specification ***[MIPI01][MIPI02]***, then they must report this capability in the Bus Configuration Register. Specifically, the Manager Function's BCR Bit[4] must indicate that that it is a Virtual Target (see ***[MIPI01]*** at ***Section 5.1.1.2.1***). Such Targets will also support the GETCAPS CCC Format 2 CCC with Defining Byte VTCAPS (see ***Section 5.1.9.3.19***) to describe more details regarding the Routing Device's capabilities and features.

- Manager Functions for Routing Devices will support the GETCAPS Format 2 CCC with Defining Byte VTCAPS and return a message of at least one byte, but preferably two bytes (i.e., both the VTCAP1 byte and the VTCAP2 byte).
  - In the first byte (VTCAP1), Bits[2:0] will have a value of **3'd3** to inform the I3C Controller that it must use the SETROUTE CCC to configure the Routing Device and its Routes to Downstream I3C Bus Segments.
    - It is recommended that the Manager Function have a different value for its Device Characteristics Register (DCR) than the DCR of its Routing Target Function(s).
    - Bit[5] will be set to **1'b1** to indicate that this Device also supports the Virtual Target Detect operation using the RSTACT CCC, as specified in the I3C Specification at ***Section 5.1.9.3.26***. Support for this operation is recommended for Manager Functions of all Routing Devices.
  - In the second byte (VTCAP2), the field values will reflect the Routing Device's implementation and its configuration of Downstream I3C Bus Segments.
    - Bit[2] will have a value of **1'b1**, unless the Routing Device only has a single Downstream I3C Bus Segment and allows its Downstream Target Devices to have the same Dynamic Addresses as the ones the I3C Controller on the Upstream I3C Bus Segment assigned. Special restrictions might apply for such a use case, and therefore Bit[2] would have a value of **1'b0**.
    - Bits[4:3] can be set at the implementer's discretion, and will reflect the Routing Device's capabilities including whether it is capable of caching context and event enable/disable configuration (such as SETBUSCON, ENEC, or DISEC) that the I3C Controller on the Upstream I3C Bus Segment previously Broadcast.

6.3.3 Downstream Bus Target Presentation

A Downstream I3C Bus Segment exposed by a Routing Device will present itself as a Virtual Target Device, with BCR Bit[4] indicating that it is also a Virtual Target.

For Routing Devices that support one or more Routes to Downstream I3C Bus Segments, each such Route is a target, is presented as a Virtual Target Device, and is known as a Routing Target Function.

- Target Functions for Routing Devices will support the GETCAPS Format 2 CCC with Defining Byte VTCAPS (see ***[MIPI01]*** at ***Section 5.1.9.3.19***) and return a message of at least one byte, but preferably two bytes (i.e., both the VTCAP1 byte and the VTCAP2 byte).
  - In the first byte (VTCAP1), Bits[2:0] will have a value of **3'd3** to inform the I3C Controller that the Routing Target Function represents a Route to a Downstream I3C Bus Segment, and that it must be configured using its Manager Function (see ***Section 6.3.2***).
    - It is recommended that the Routing Target Function DCR value be different from the Manager Function DCR value.
    - Bit[5] will be set to **1'b1** to indicate that this Device also supports the Virtual Target Detect operation using the RSTACT CCC, as specified in the I3C Specification at ***Section 5.1.9.3.26***. Support for this operation is recommended for Target Functions of all Routing Devices.
  - In the second byte (VTCAP2), the fields will have values applicable to the Downstream I3C Bus Segment being presented:
    - Bits[1:0] would have a value of **2'd1**, unless special exceptions apply, as per the use case.
    - Bit[2] will have a value of **1'b1** to indicate that this Target Function will hide the addresses of its I3C Target Device(s) on this Downstream I3C Bus Segment, and will transparently remap all commands and private writes/reads based on the message contents.
    - Bits[4:3] will return a value of **2'd0**. It is expected that all Target Functions exposed by a given Routing Device would share the same settings for Bus Context and Conditions (which are reported via the Manager Function).

Routing Devices might also allow the Downstream Bus Segments, or the I3C Targets on such segments, to be indirectly configured by the I3C Controller on the Upstream Bus segment via each Routing Target Function, per ***Section 6.1.2***. However, the Bus Controller of the Downstream Bus segment is responsible for assigning individual Dynamic Addresses to each of the I3C Targets on that segment. If the Routing Device serves this function, then it will typically configure each Downstream Bus segment and serves as the segment's Active Controller (as well as the Primary Controller for initialization).

6.3.3.1 Encapsulated Transactions to Downstream Bus

Typical Routing Devices do not directly map the I3C Targets on their Downstream Bus segment(s) to the Upstream Bus segment, meaning that the Primary Controller on the Upstream Bus segment must communicate via the Routing Target Function in order to drive I3C transactions on the Downstream Bus segment. This requires some encapsulation of the I3C transactions that are sent over the Upstream Bus segment, as part of the Routing Target Function's I3C content protocol.

Once a Routing Device has been configured and a Route to a Downstream Bus segment has been established, the Routing Target Function receives Private Writes from its Upstream Bus (i.e., from the Primary Controller). The Routing Device then examines the encapsulated write data per the I3C content protocol, enqueues the I3C transfer commands based on the data structures that it received, and then its internal Bus Controller executes the I3C transfer commands on the Downstream Bus segment. Similarly, the Routing Device collects the results of each I3C transfer command, enqueues them for delivery to the Upstream Bus segment, and then notifies the Primary Controller that its requested transactions have been processed (i.e., executed either to successful completion, failure, or termination for other reasons) per the I3C content protocol of the Routing Target Function.

The Routing Device could also relay any In-Band Interrupts (IBIs) sent by I3C Targets on the Downstream Bus segment to the Upstream Bus segment, so that the Primary Controller could be aware of them. This could happen in one of two different ways:

- The Primary Controller could periodically poll the Routing Device's Routing Target Function to see whether any transfer command responses or IBI Notifications are pending (i.e., via a Private Read). If any IBI Notifications are pending, then the Routing Target Function provides ACK and then returns a description of transfer command responses and/or IBI Notifications (i.e., per the I3C content protocol).
- The Routing Device could raise an IBI Request on the Upstream Bus segment containing a short notification that some transfer command responses or IBI Notifications are pending and must be read (e.g., using the I3C Pending Read Notification contract, per ***[MIPI01]*** at ***Section 5.1.6.2.2***). The subsequent read contains a description of transfer command responses and/or IBI Notifications (i.e., per the I3C content protocol).

Notification of such responses and/or IBIs could be encapsulated as part of the I3C content protocol for a Routing Target Function. For IBI Notifications, this encapsulation would include status, such as whether the Bus Controller provided ACK or NACK to each IBI Request from a Downstream I3C Target, identified by its Dynamic Address, as well as whether the IBI Request was ACK'd. If the IBI Request was ACK'd, then the encapsulation would also include any IBI data payload (if applicable) as well any subsequent read that would be performed automatically by the Bus Controller (i.e., for an IBI that signaled a Pending Read Notification, per ***[MIPI01]*** at ***Section 5.1.6.2.2***).

Note:

The specific I3C content protocol that a Routing Target Function supports, for encapsulated I3C transfer commands and responses as well as IBI notifications, is not defined in this Application Note, nor is it defined in the I3C Specification.

6.3.3.2 Downstream Target Mapping Option

For Routing Devices that only support one Route to a Downstream I3C Bus Segment, an implementer could decide to present some or all of the I3C Target Devices on the Downstream I3C Bus Segment as a Virtual Target Device that can be directly addressed (i.e., via store-and-forward) on the Upstream Bus segment. For some use cases, this removes the need to encapsulate certain I3C transactions involving the Downstream I3C Target.

If this option is chosen, then each such Target Device would have its own Dynamic Address that is visible on the Upstream Bus segment (i.e., assigned directly by the Primary Controller), for which the Routing Device's Peripheral logic must act as a proxy or intermediary.

Each mapped Downstream Target could either be configured automatically by the Routing Device (i.e., as and when it is discovered by the Bus Controller), or via other configuration commands that could be sent from the Primary Controller. As each mapped Downstream Target is configured (i.e., virtually exposed) to the Upstream Bus segment, the Routing Device simulates a Hot-Join Request for a new Virtual Target and allows the Primary Controller to assign a Dynamic Address that is used for that particular mapping. The Routing Device then associates that new Dynamic Address with the mapped Downstream Target for future communications (e.g., Private Writes, Private Reads, CCCs).

Note:

> *The Routing Device would need to intercept certain CCCs such as SETNEWDA from being sent to the mapped Downstream Target, as the Dynamic Address assigned by the Primary Controller only applies to communications with the Virtual Target (i.e., as presented to the Upstream Bus segment).*

As the Primary Controller sends I3C transactions to the Virtual Target, the Routing Device's Peripheral logic would respond on behalf of the Downstream Target, in a manner similar to a composite I3C Device (per ***Section 5.2***) per the transaction type:

- **For Write transactions:** The Routing Device would typically ACK the write on behalf of the Downstream I3C Target, then cache the data for later relaying to the Downstream I3C Bus.
- **For Read transactions:** The Routing Device must only ACK if it has data ready to respond (i.e., from a pre-emptive read that it could have issued earlier, or some other notification from the Downstream Target); otherwise, it must NACK the read.
- **For Direct CCCs:** The Routing Device typically responds to Direct CCCs sent to the Downstream Target. It could optionally pre-fetch certain responses (i.e., for Direct GET or Direct Read) based on known status or a private contract, per the Downstream Target type. This use model does not work well for all Direct CCCs.

Figure 9 shows an example of an I3C Routing Device (derived from ***Figure 8***) that also maps two I3C Targets on its Downstream I3C Bus segment as separate Virtual Targets, presented via its Peripheral logic. The Routing Device could still accept I3C Bus transactions directed to the entire Bus via its Routing Target Function (i.e., either Broadcast CCCs or transactions directed to any other I3C Targets that might not be mapped to the Upstream Bus segment).

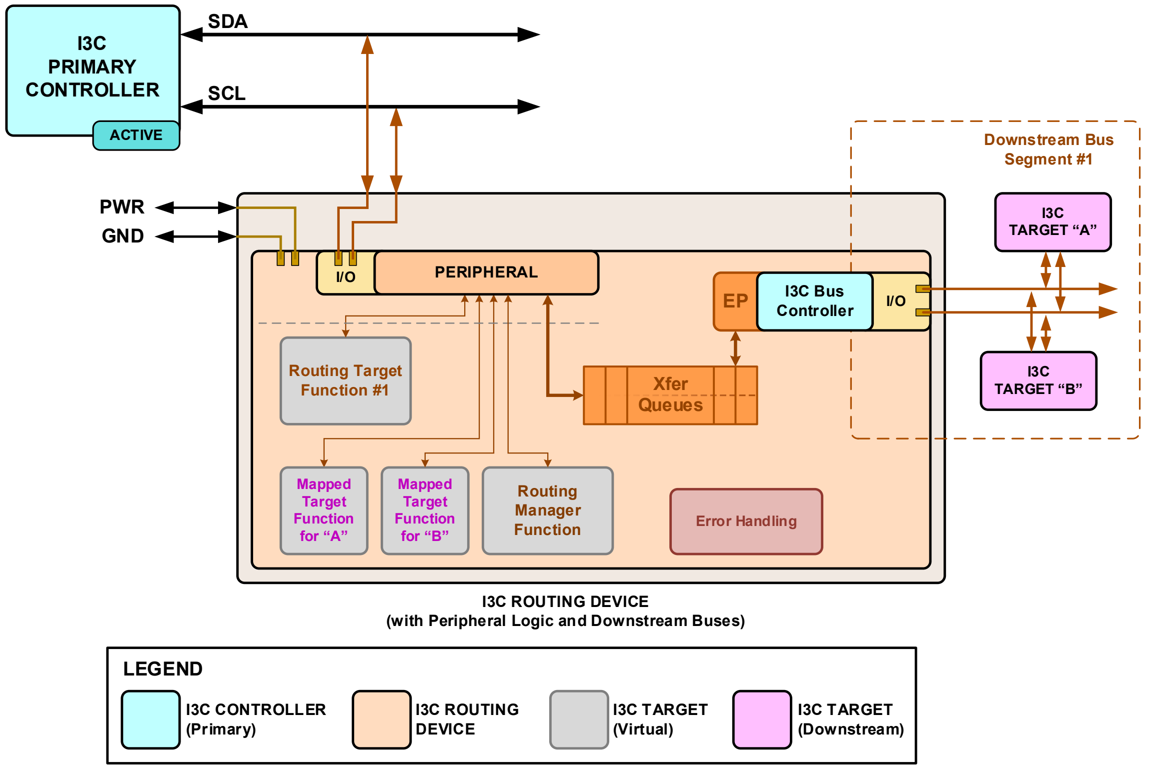


Figure 9 I3C Routing Device with Mapped Downstream Targets

Note:

*The decision to map Downstream Targets to the Upstream I3C Bus segment is left up to the implementer, as it requires an additional level of routing complexity that might not be appropriate for all implementations or all use cases. If this option is chosen, then the Routing Device would present a Virtual Target for each such mapped Downstream Target as though it had "Shared Peripheral Logic" (i.e., had Virtual Target type **3'd5**). Additionally, in this scenario, an implementer must still configure the Routing Device to present the Route as a Target Function for transactions directed to the Downstream Bus segment, or to any other such I3C Targets that do not map to Virtual Targets that can be directly addressed on the Upstream Bus segment.*

6.3.4 Support for Remote Downstream Buses

Typical I3C Routing Devices are implemented as single-component solutions. However, an advanced I3C Routing solution could also extend the I3C Routing solution into two distinct components, using a higher-speed link or network between them.

The two components are envisioned as follows:

- **An advanced I3C Routing Device** (i.e., the upper portion) that connects to the Upstream I3C Bus segment, presents Virtual Targets on this segment, manages the transactions to/from its Primary Controller, and connects to a higher-speed link; and
- **A remote Endpoint** (i.e., the lower portion) that also connects to that same link, owns the Downstream I3C Bus segment (i.e., contains the I3C Bus Controller for this segment), and drives the enqueued Private Write/Read transactions to the indicated I3C Target Devices on its segment.

Figure 10 shows an example of a two-component I3C Routing solution featuring an I3C Routing Device (derived from ***Figure 8***) paired with a remote Endpoint, connected via a higher-speed link. The remote Endpoint controls its Downstream I3C Bus segment and receives transactions from the I3C Routing Device.

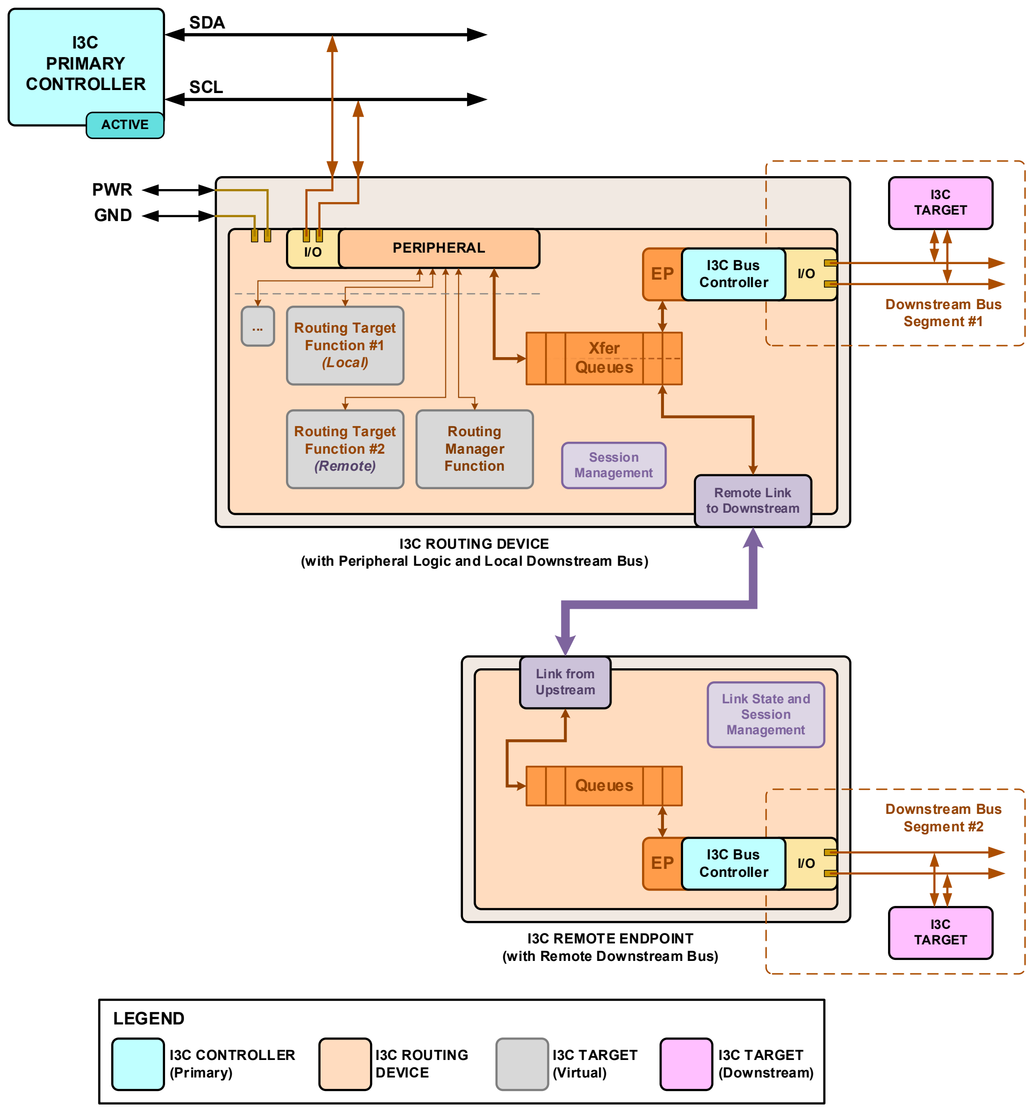

I3C PRIMARY CONTROLLER
ACTIVE
SDA
SCL
PWR
GND
I/O
PERIPHERAL
...
Routing Target Function #1 (Local)
Routing Target Function #2 (Remote)
Routing Manager Function
Xfer Queues
Session Management
Remote Link to Downstream
EP
I3C Bus Controller
I/O
I3C TARGET
Downstream Bus Segment #1
I3C ROUTING DEVICE (with Peripheral Logic and Local Downstream Bus)
Link from Upstream
Link State and Session Management
Queues
EP
I3C Bus Controller
I/O
Downstream Bus Segment #2
I3C TARGET
I3C REMOTE ENDPOINT (with Remote Downstream Bus)
LEGEND
I3C CONTROLLER (Primary)
I3C ROUTING DEVICE
I3C TARGET (Virtual)
I3C TARGET (Downstream)

Figure 10 I3C Routing Device with Remote Link to Downstream Bus

Note:

> *In the example above, the Virtual Target for Routing Target Function #1 is still associated with a **local** Downstream I3C Bus Segment, whereas the Virtual Target for Routing Target Function #2 is now associated with a **remote** Downstream I3C Bus Segment, provided by the routed Endpoint device. The key architectural difference from* **Figure 8** *is the insertion of a remote link in the Transfer Queues of the I3C Routing Device. In effect, the remote link extends the Transfer Queues across a longer physical distance, which could slightly increase the latency of I3C Private Write/Read transfers that are enqueued by the Primary Controller on the Upstream I3C Bus Segment. However, the overall flow remains similar to a single-component I3C Routing solution.*

It is beyond the scope of this Application Note to detail the complete set of requirements or implementation details pertaining to the higher-speed link between such an I3C Routing Device and any connected remote Endpoint devices that control the Downstream I3C Bus segments. However, implementers should consider some minimal guidelines and expectations for such a solution:

- The high-speed link or network between the Routing Device and the remote Endpoint must provide high reliability and robustness for transmitting messages or packets for enqueued I3C transfer commands, I3C IBI Notifications or other messages between these components.
  - Nonetheless, if the high-speed link or network is interrupted or if an established connection fails, then the I3C Routing Device must notify the Primary Controller, and not accept new enqueued I3C transfer commands directed to the Downstream I3C Bus segment.
- The high-speed link or network must also be bi-directional in nature, and have sufficiently high bandwidth and low latency, in order to mitigate the performance impacts on a remote I3C Routing solution (as compared with a single-component I3C Routing Device).
- The method for sending packets or messages over this link or network must allow both I3C transfer commands and IBI notifications (i.e., the encapsulated I3C content protocol for the I3C Routing Target Function) as well as link management commands and responses, which might be specific to the type of high-speed link or network.
- The remote Endpoint could either expose its own Transfer Queue size or provide some method of flow control to the Routing Device, in order to inform it of any limits that it might have.
- The I3C Bus Controller in the remote Endpoint must be sufficiently autonomous, in order to handle conditions (such as IBIs raised by Downstream I3C Targets) without requiring immediate response from the logic inside the Routing Device.
- The I3C content protocol for each Routing Target Function that is linked to a remote Downstream Bus segment could be the same as an I3C content protocol for a Routing Target Function of a single-component I3C Routing Device (see ***Section 6.3.3.1***) for consistency and interoperability.
- If the high-speed link or network supports a "one-to-many" connection scheme (i.e., where an I3C Routing Device could establish connections to multiple remote Endpoints) then the Routing Device could present each of its connected Endpoints as a unique Virtual Target (i.e., one for each Downstream Bus segment).
  - As each Endpoint is connected, the Peripheral Logic simulates a Hot-Join Request on the Upstream I3C Bus segment, presenting a new Virtual Target to the Primary Controller, where each receives a unique Dynamic Address.
  - The Routing Device then steers the encapsulated I3C transfer commands (i.e., Private Writes/Reads) to each remote Endpoint's Downstream I3C Bus segment, based on the Dynamic Address.
- The Routing Manager Function inside the Routing Device could offer additional session management capabilities to the Primary Controller of the Upstream I3C Bus segment. These might be specific to the type of high-speed link or network. Alternately, the session management capabilities of the Routing Device could be pre-configured, or configured via other means (i.e., out of band).

6.3.5 Support for Controller Role or Segment Secondary Controller

A Routing Device could also offer Secondary Controller capabilities on any of its Downstream I3C Bus Segments. For such a Routing Device, its I3C Bus Controller logic for the Downstream segment must have Secondary Controller capabilities, and could optionally initialize as a Secondary Controller, if so configured.

If such a Routing Device is capable of initializing as either the Primary Controller of the Downstream segment or as a Secondary Controller of it, then the Routing Device must be configured to provide a suitable Primary Controller (i.e., one with a separate I3C Device) on that Downstream segment, and the Routing Device must contain logic to manage the Controller Role Handoff procedure with respect to its Routing Target Function. Additionally, the Routing Device must be capable of handling the case in which it initializes as the Primary Controller of that Downstream segment, but then subsequently passes the Controller Role to another Controller-capable Device (i.e., one that initialized as a Secondary Controller and is capable of taking the Controller Role).

In either case, the Routing Device must know when it is not the Active Controller, since it cannot route Private Write/Read transfers from its Transfer Queues (i.e., to/from the Upstream segment) until it receives the Controller Role and can drive enqueued transfers on the Downstream segment. The Routing Target Function could implement support for sending an IBI to the Upstream segment to notify the Active Controller when it has handed the Controller Role off (i.e., is no longer the Active Controller of the Downstream segment).

Note:

> *If the Routing Device manages multiple Downstream segments and has the Secondary Controller capability for some or all of these segments, then it must separately manage the initial Role configuration and subsequent Controller Role transitions for each of these segments.*

Figure 11 shows a partial view of an example Routing Device (derived from a portion of ***Figure 8***) with one such Downstream I3C Bus Segment, where the I3C Bus Controller logic has Secondary Controller capability. For this Downstream segment the Routing Device can serve as both Active Controller and Secondary Controller, and can manage the transition between the Controller and Target roles on its I3C Bus Segment. The Transfer Queues that serve this I3C Bus Segment could (optionally) be bi-directional to support limited transactions in I3C Target mode. In this example, the Routing Device has been configured to act as a Secondary Controller for this I3C Bus Segment on initialization, since there is also a Primary Controller on the same segment (which is currently the Active Controller).

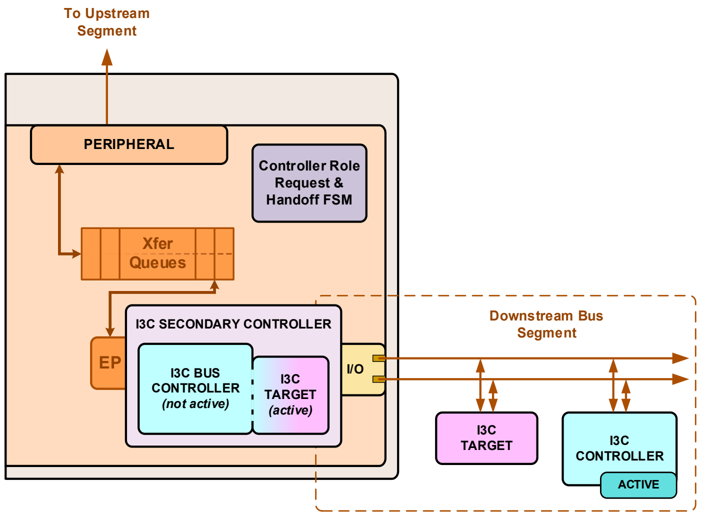


Figure 11 I3C Routing Device with Downstream Secondary Controller Capability

Note:

*The example shown in **Figure 11** does not imply that the Controller-capable Device on the Downstream Bus segment has any effective control or influence over the Upstream Bus segment. The Transfer Queues and other routing-specific logic inside the Routing Device will manage all communications across the Bus segments, and the Routing Device will present itself appropriately to each Bus segment (i.e., with its proper Role). In this example, the Upstream Bus segment is not visible to the Downstream Bus segment.*

6.4 Hub Devices

Specialized I3C Hub Devices can provide isolation between the separate electrical domains of an Upstream Bus segment and a Downstream Bus segment, enabling use cases where the two segments have electrical property differences (even incompatibilities) that necessitate such isolation.

For example, the I3C Target Devices in a Downstream Bus segment might operate at different electrical parameters, from the Upstream segment, including but not limited to:

- Different operating voltage requirements (i.e., parameter $\mathbf{V_{DD}}$, per ***[MIPI01]*** at ***Section 6.1***)
- Different current requirements (i.e., parameter $\mathbf{I_i}$, per ***[MIPI01]*** at ***Section 6.1***)

An I3C Hub Device can also provide buffering to alleviate the total loading of the I3C Bus, enabling applications in which the total number of I3C Target Devices would result in excessively high capacitance (i.e., parameters $\mathbf{C_i}$ and $\mathbf{C_b}$) if all the Devices were resident on the same Bus segment.

It is beyond the scope of this Application Note to detail the complete set of capabilities that could theoretically be provided by such an I3C Hub Device, but the core Hub Device concepts can be stated as:

- Isolation between the Upstream and Downstream Bus segments;
- Configurable I/O parameters for the Downstream Bus segment; and
- At least one Virtual Target, presented by the I3C Hub Device as a Hub Manager Function, that provides the I3C Controller on the Upstream segment control over those configurable parameters. (I.e., via CCCs or Private Write/Read transactions to the Hub Manager Function's Dynamic Address.)

Note:

In a strict sense, simple I3C Hub Devices that provide isolation between Upstream and Downstream I3C Bus Segments might not have a Virtual Target that acts as a Hub Manager Function, if the configuration of the Hub Device is accomplished by other means (i.e., not using in-band transactions on the Upstream Bus segment from the I3C Controller). However, this Application Note assumes that the advanced capabilities and configurability of such Hub Devices will be a desirable feature, and that the Virtual Target capabilities defined in this Application Note will provide a mechanism for providing this feature. In the remaining sections of this Application Note, the term 'Hub Device' is interpreted as an advanced I3C Hub Device that includes this capability, presents a Virtual Target as a Hub Manager Function, and allows configuration via this Virtual Target.

6.4.1 Architectural Overview

An I3C Hub Device is similar in concept to a Composite I3C Device (see ***Section 5.2.1***) that uses Peripheral logic to present at least one Virtual Target, and that interacts with its Upstream I3C Bus through its additional capabilities (i.e., capabilities over and above those required of a simple I3C Target). If the Hub Device also presents other Virtual Targets (i.e., ones not shown below, or not described in this Application Note), then the Peripheral Logic could have additional capabilities, however these are not required.

The Hub Device also contains Isolation Logic that enables I3C transactions to flow across the boundary between the electrically isolated segments, using appropriate isolation methods. This isolation could be accomplished by specialized circuits that track the states of the SDA and SCL lines, repeating or mirroring actions taken on one segment and replicating them to the other segment, per the I3C Specification (see ***Section 6.4.3***).

Figure 12 shows an example of a Hub Device that presents a single I3C Virtual Target on an I3C Bus, namely a Hub Manager Function. A Hub Manager Function is used by the Upstream Bus segment's I3C Controller to configure and manage the Hub Device and the electrical I/O parameters of its Downstream Bus segment, as well as to control the Isolation Logic's operating mode.

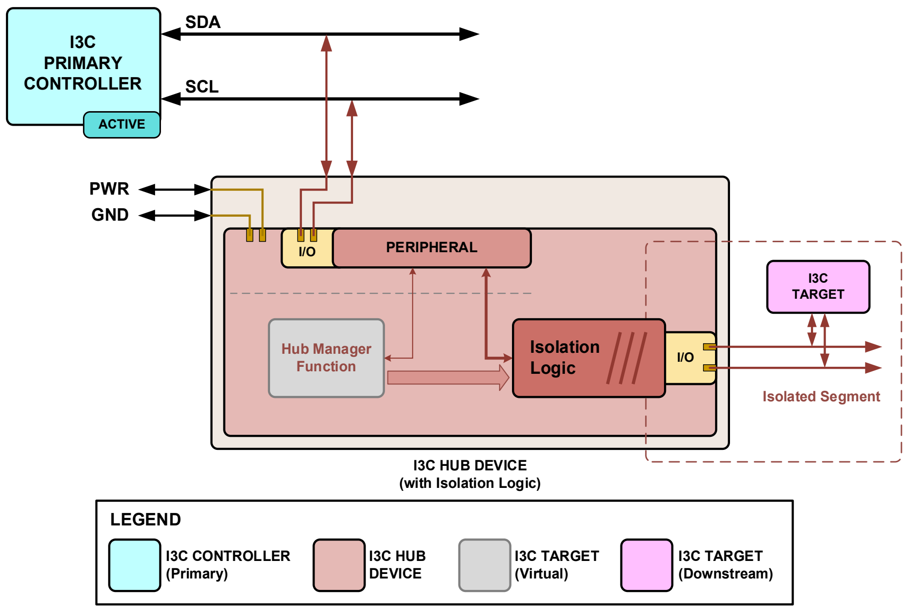


Figure 12 I3C Hub Device with Isolation Logic and Downstream Target Devices

The Peripheral logic in such a Hub Device maintains the Dynamic Address of the Virtual Targets, including the Hub Manager Function (and potentially other optional Functions), and also responds to I3C transactions that address such Virtual Targets. However, the Peripheral logic allows I3C transactions that address other Target Addresses (i.e., those not assigned to any internal Virtual Targets) to pass to the Isolation Logic where they can be tracked and echoed to the Downstream Bus segment.

6.4.2 Hub Manager Function

Hub Devices present at least one Hub Manager Function as a Virtual Target. A Hub Manager Function exposes the Device's presence on the I3C Bus and describes its capabilities and optional features (see ***Section 6.1.1***).

If such Devices are fully compliant with version 1.1 of the I3C Specification ***[MIPI02]***, then they must report this capability in the Bus Configuration Register (BCR) for the Manager Function's Virtual Target. Specifically, the Manager Function's BCR Bit[4] must indicate that that it is a Virtual Target (see ***[MIPI01]*** at ***Section 5.1.1.2.1***). Such Targets will also support the GETCAPS CCC Format 2 CCC with Defining Byte VTCAPS (see ***[MIPI01]*** at ***Section 5.1.9.3.19***) to describe more details regarding the Hub Device's capabilities and features.

- Manager Functions for Hub Devices will support the GETCAPS Format 2 CCC with Defining Byte VTCAPS and return a message of at least one byte (i.e., the VTCAP1 byte).
  - In the first byte (VTCAP1), Bits[2:0] will have the value **3'd6** to indicate that it is a Hub Device that supports pass-through transactions.
  - The second byte (VTCAP2) does not currently define any bit fields relating to Hub Device capabilities or features.

The Hub Manager Function typically also describes its capabilities and optional features, including:

- The type of Isolation Logic, including the electrical parameters and I/O pad controls that the I3C Controller can access or change;
- Any additional internal routing delays that the Isolation Logic might add (i.e., for each transaction that goes through the Isolation Logic to the I/O pads and addresses an I3C Target Device on the Downstream segment);
- The Isolation Logic's initial mode (i.e., whether it is engaged vs. disengaged) when the Hub Device is powered on and initialized by the I3C Controller;
- The method of engaging the Isolation Logic and bringing any I3C Target Devices on its Downstream Bus segment online (i.e., with a virtual connection to the Upstream Bus segment); and
- Any additional advanced capabilities that might be supported, including error handling capabilities.

Note:

The specific I3C content protocol that a Hub Manager Function supports is not defined in this Application Note, nor is it defined in the I3C Specification.

6.4.3 Isolation Logic and Transactions for Downstream Devices

The Isolation Logic plays a special role in tracking the actions that are initiated by I3C Devices on the Upstream Bus segment and repeating or mirroring such actions, so that they might be repeated (or replicated) to the Downstream Bus segment. Similarly, the Isolation Logic must track actions that the I3C Devices on the Downstream Bus segment might initiate, and must track these in order to repeat them to the Upstream Bus segment.

Notably, the Isolation Logic must contain a Leader/Follower tracking circuit that identifies electrical state changes on one Bus segment and replicates them to the other segment with minimal delay. This tracking circuit must determine which segment is the “leader” and which is the “follower” at any given moment, according to the I3C Specification’s defined behaviors for I3C Devices and how they respond to changing states. This enables the Isolation Logic to act as a nearly invisible pass-through for I3C transfers, in contrast to an I3C Bridge Device (which converts I3C transfers to/from actions on another bus or interface), or an I3C Routing Device (which uses internal queues to store and forward I3C transfers across its Bus segments).

The typical use case for a Hub Device assumes that the Active Controller resides on the Upstream Bus segment, and enables the following types of transfers and other procedures across its Isolation Logic:

- I3C Devices on the Downstream Bus segment can see and participate in transactions initiated on the Upstream segment, i.e., by the Active Controller, including:
  - I3C Broadcast CCCs that are sent to all Targets, where the I3C Devices on the Downstream Bus segment must ACK the Broadcast Address;
  - I3C Direct CCCs that use framing and require specific Targets to provide ACK/NACK, where the Targets on the Downstream Bus segment must ACK or NACK their Target Address at the appropriate time, and optionally respond with data (i.e., for Direct Read/GET CCCs);
  - I3C modal flows (e.g., Dynamic Address Assignment with ENTDAA), where the I3C Devices on the Downstream Bus segment must participate in the flows (i.e., arbitrate and eventually receive a Dynamic Address);
  - Private Write transactions in SDR Mode that are directed to Targets that are on the Downstream Bus segment and will ACK the Write;
  - Private Write transactions in SDR Mode that are directed to Group Addresses having at least one Target assigned to that Group on the Downstream segment that will ACK the write;
  - Private Read transactions in SDR Mode that are directed to Targets that are on the Downstream Bus segment, and for which an ACK with response to the Read will be returned;
  - Entering HDR Modes, using the HDR Restart Pattern to frame different transfers in the same HDR Mode, and driving HDR Generic Write and Read transfers in supported HDR Modes directed to Targets (and optionally Groups, as above) on the Downstream Bus segment, per the signaling protocol of that HDR Mode;
  - Exiting HDR Modes using the HDR Exit Pattern;
  - Error recovery procedures (per ***[MIPI01]*** at ***Section 5.1.10***) that are meant for all I3C Devices on both Upstream and Downstream segments, or (optionally) only on the Downstream segment;
  - Reset flows that include the Target Reset Pattern (per ***[MIPI01]*** at ***Section 5.1.11***) that are meant for all I3C Devices on both Upstream and Downstream segments, or (optionally) only on the Downstream segment;

- I3C Devices on the Downstream segment can detect defined Bus Conditions (including Bus Idle and Bus Available, per specification ***Section 5.1.3.2***) and know when they can drive a START request to initiate various types of In-Band Interrupt requests, including:
  - In-Band Interrupt Requests from a Target;
  - Hot-Join Requests; and
  - Controller Role Requests (although the typical use case for such a Hub Device does not support an I3C Controller-capable Device on its Downstream Bus segment);
- I3C Devices on the Downstream segment can monitor the Address Header after a START condition and optionally arbitrate their assigned Dynamic Address into the Arbitrable Address Header, to drive an In-Band Interrupt Request (as listed above).

Note:

> ***The use cases listed above are not exhaustive****, but are intended to provide an overview of the kinds of situations that the Isolation Logic would expect to encounter and handle, with the goal of tracking changes in state on the SDA and SCL lines that connect to both the Upstream and Downstream segments. The Isolation Logic needs some level of awareness of the I3C electrical states (i.e., how and when to transition SDA between Open Drain vs. Push-Pull), such that it accurately replicates to one segment the actions driven by one or more I3C Devices connected to the other segment. The Isolation Logic designer must assume that the behavior of the I3C Devices on each Segment conforms to the I3C Specification.*

The Isolation Logic also works with the Peripheral logic to ensure that when transactions from the Upstream Bus segment are addressed to internal Virtual Targets (such as the Hub Manager Function; see ***Section 6.4.2***), the Isolation Logic gives priority to the Upstream Bus segment and always treats the Downstream Bus segment as the "follower".

Hub Devices with more advanced functionality could provide additional control over the Isolation Logic's operating mode, and could support optional requests sent to the Hub Manager Function via a content protocol. Such optional requests could include:

- Control over when attempts to raise In-Band Interrupt requests from I3C Targets on the Downstream Bus segment would be allowed vs. not allowed. This could be achieved by:
  - Intercepting and overriding the Dynamic Address of such I3C Targets (i.e., by determining when a START condition or IBI Request has been initiated by a Downstream Target, and then trying to drive a lower Address value into the Arbitrable Address Header); or
  - Providing NACK when the Dynamic Address of a known I3C Target on the Downstream Bus segment is detected after a START, when the Downstream Target initiated the IBI Request (i.e., by temporarily disconnecting the Leader/Follower tracking circuit, driving NACK with a clock cycle, etc.);
- Control over the Isolation Logic's ability to pass HDR Modes to the Downstream Bus segment, for situations where the additional propagation delay or other electrical parameters do not support transfers in some or all HDR Modes, for the specific I3C Devices on the Downstream Bus segment;
- Control over when the Downstream Bus segment is "parked" and kept in a Bus Idle state (i.e., when SCL and SDA lines for the segment are pulled to High and stay High) and the Isolation Logic would not pass any transactions from the Upstream Bus segment;
- Control over whether the Isolation Logic is allowed to pass the HDR Exit Pattern and/or I3C Target Reset Pattern (either with or without preceding RSTACT CCCs) to the Downstream Bus segment, when operating in SDR Mode.

Hub Devices with more advanced functionality could also support special error recovery commands sent to the Hub Manager Function via a content protocol. For these special error recovery commands, the Hub Manager Function could temporarily command the Isolation Logic to disconnect its Leader/Follower tracking circuit, and then initiate certain error recovery procedures that only affect the Downstream Bus segment. Such procedures could include:

- Sending START, 7'h7E/W, HDR Exit Pattern followed by STOP to recover from various Target Errors (per ***[MIPI01]*** at ***Section 5.1.10.2.3*** and ***Section 5.1.10.2.5***) for situations when the Isolation Logic detects that the Leader/Follower tracking circuit definitively determined that at least one I3C Device on the Upstream Bus segment did provide ACK to the Broadcast Address, but no I3C Devices on the Downstream Bus segment provided ACK to the Broadcast Address.
  - For such cases, the Isolation Logic must determine whether such an ACK was seen on earlier occasions from any I3C Devices on the Downstream Bus segment.
  - If the Isolation Logic also has the ability to proactively monitor for such situations (i.e., loss of Broadcast ACK on the Downstream Bus segment), then the Peripheral logic and Hub Manager Function could also generate an In-Band Interrupt Request to the Active Controller with a notification of that condition.
- Stuck SDA Handling (per ***[MIPI01]*** at ***Section 5.1.10.2.6***) for situations when the Isolation Logic detects an I3C Device holding SDA High or Low on the Downstream Bus segment, and automatically disconnects its Leader/Follower tracking circuit accordingly (i.e., to protect the Upstream Bus segment).
  - For situations where the Isolation Logic must automatically disconnect its Leader/Follower tracking circuit to protect the Upstream Bus segment, the Peripheral logic and Hub Manager function could generate an In-Band Interrupt Request to the Active Controller with a notification of that condition.
- Target Reset flows using the Target Reset Pattern, with zero or more preceding RSTACT CCCs in a sequence (per ***[MIPI01]*** at ***Section 5.1.11***) sent to I3C Target(s) on the Downstream Bus segment only, but not visible to any I3C Devices on the Upstream Bus segment.
- Direct reset pin control or power management of one or more I3C Devices on the Downstream Bus segment, using special reset and power control logic that could force such I3C Devices to be reset or power-cycled in extreme cases of failure or non-response.
  - This would also require additional GPIO outputs (for reset pin control) or power relay circuits (for power management) that could be controlled via the content protocol.

6.4.4 Support for Open Drain Pull-Up Handling

For some system designs, the Downstream Bus segment might require a special Pull-Up value due to the number of Target Devices that must be isolated from the Upstream Bus segment, and the overall design of the Downstream Bus (see I3C electrical parameter **R_P**, per ***[MIPI01]*** at ***Section 6.1***). In such situations, the I3C Hub Device must provide the appropriate Pull-Up for Open Drain mode for its Downstream Bus segment, per the I3C Specification. Parameter **R_P** for the Downstream Bus segment needs to be chosen carefully because the Hub Device does not store and forward I3C transfers (as an I3C Routing Device would do). This means that all Private Write/Read transfers initiated by the Active Controller on the Upstream Bus segment must necessarily happen at the same data transfer rate on the Downstream Bus segment.

The Hub Device must either detect when the Active Controller on the Upstream Bus segment is using its own Pull-Up; or the Hub Device could utilize an external Pull-Up control pin provided by the Active Controller for this purpose (if available). The I3C Specification does allow an I3C Controller Device to provide an external pin for Open Drain Pull-Up (see ***[MIPI01]*** at ***Section 5.1.3.1*** and ***Section 6.1***); this pin can the Controller's sole source of Pull-Up, or it can be in addition to an internal Open Drain class Pull-Up structure (e.g., an internal Pull-Up resistor or current source which is only engaged or enabled for Open Drain mode).

Figure 13 shows a partial view of an example Hub Device (derived from a portion of ***Figure 12***) with a direct pin connection to the Active Controller of the Upstream Bus segment (i.e., the Primary Controller). The Hub Device monitors this direct pin connection, and selectively enables or disables its own Open Drain class Pull-Up for its Downstream Bus segment in response to the signal driven by the Active Controller (as well as other conditions detected by the Leader/Follower tracking circuit). In this example, the Open Drain class Pull-Up for the Downstream Bus segment can either be internal to the Hub Device, or external (i.e., a Pull-Up resistor controlled by an output pin in the I/O pads facing the Downstream Bus segment). Such a design allows the Hub Device to be tuned to the specific needs of the Downstream Bus segment and its Target Devices.

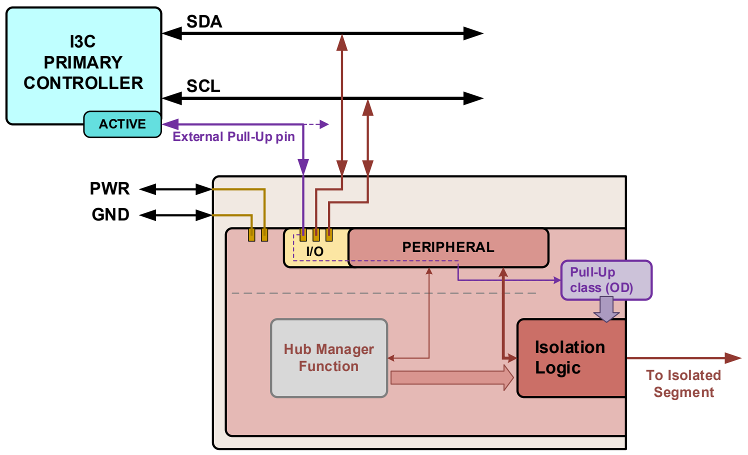


Figure 13 I3C Hub Device using External Pin for Open Drain Pull-Up

Note:

> *Per the I3C Specification, each Bus segment also requires a High-Keeper Pull-Up (see* ***[MIPI01]*** *at* ***Section 5.1.3.1****), however these High-Keeper Pull-Ups are not shown in* ***Figure 13****. If the Hub Device also provides the High-Keeper Pull-Up, then it can be either internal or external. The Hub Device must also disengage both Pull-Ups (i.e., for High-Z) as necessary when it detects certain types of Handoff transitions.*
>
> *An I3C Controller Device that provides an external Pull-Up for Open Drain will typically use this pin to control an external Pull-Up resistor for the Upstream Bus segment. This Pull-Up resistor is also not shown in* ***Figure 13****.*