
TWENTIETH ANNUAL
TestConX™
March 3 - 6, 2019
Hilton Phoenix / Mesa Hotel
Mesa, Arizona
Archive

COPYRIGHT NOTICE

www.testconx.org

MIPI I3C®: A New Bare-Metal Interface for Debug and Test

Enrico Carrieri
Intel Corporation
Debug WG Chair, MIPI Alliance

TestConX™
Mesa, Arizona • March 3 - 6, 2019

Topics

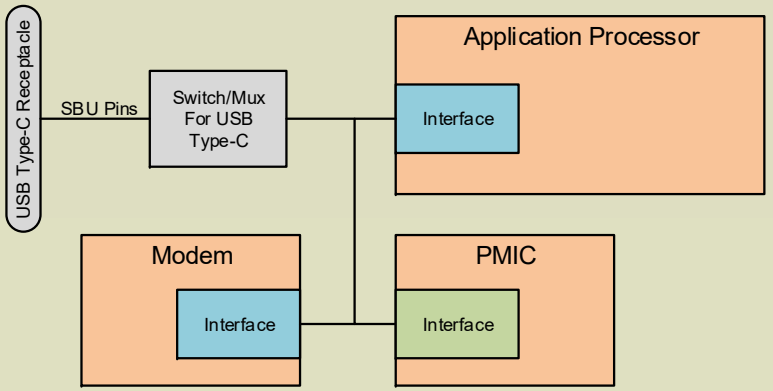
- Problem Statement
- Goals
- Solution: MIPI I3C
- Debug/Test Extensions to MIPI I3C
- On-going Development
- Summary / Conclusions

Problem Statement

Current debug and test interface solutions (e.g., JTAG/cJTAG, I²C, UART) are falling short in meeting newer requirements/use cases.

For example:

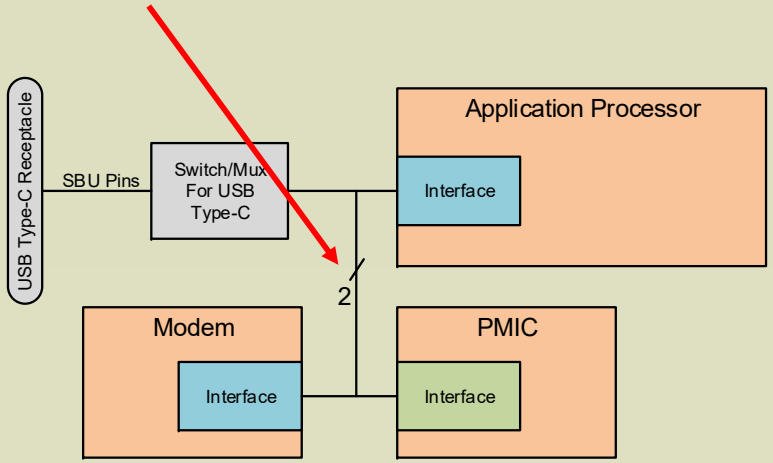
- Want to debug/test all components (application processor, PMIC, & modem).
- Want to attach debug and test system (DTS) to external pins yet still use the application processor and modem as DTS.
- Must work if application processor is powered down (e.g., low power state).



Goals

Have a scalable, multi-mastering debug and test interface that can connect power managed components on a platform. It shall…

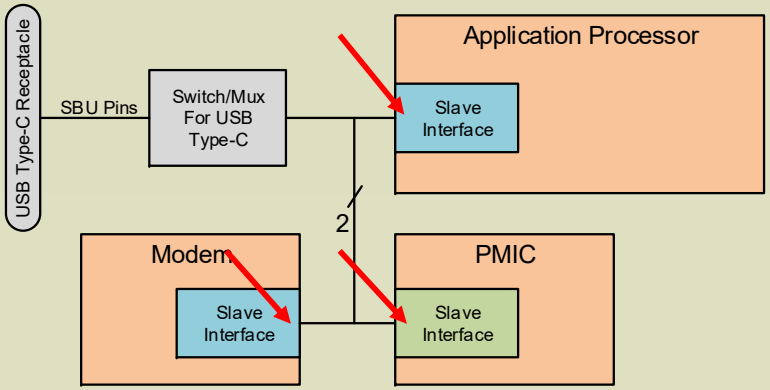
1. Require a minimal set of pins (i.e., 2 wires) including those needed for interrupts.



Goals (#2)

Have a scalable, multi-mastering debug and test interface that can connect power managed components on a platform. It shall…

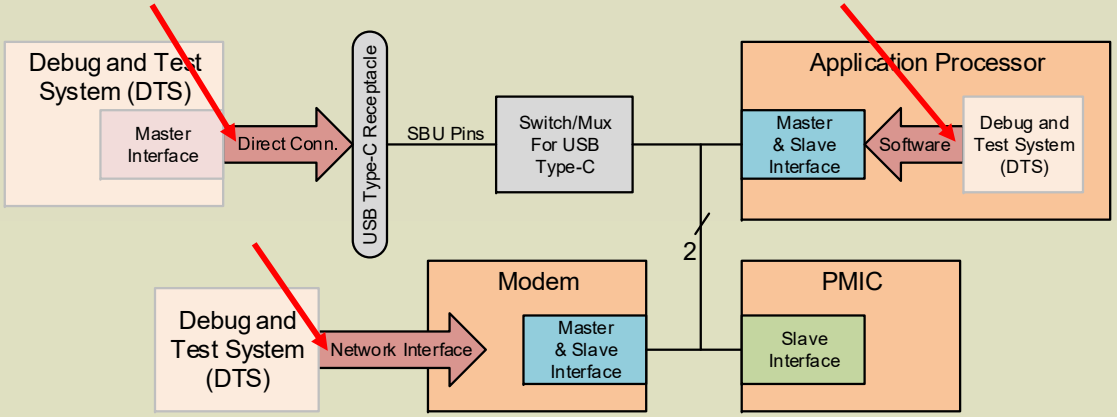
2. Allow multi-component connectivity.



Goals (#3)

Have a scalable, multi-mastering debug and test interface that can connect power managed components on a platform. It shall…

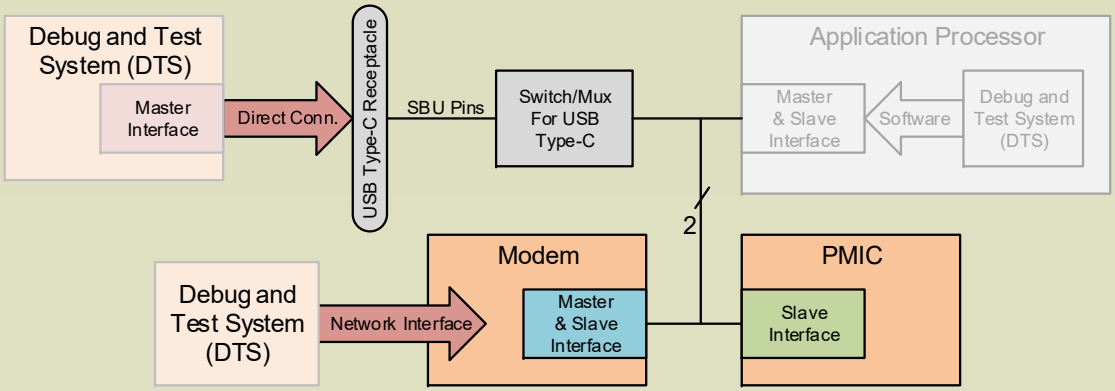
3. Support multiple entry/mastering points to allow flexibility across the platform's lifecycle.



Goals (#4)

Have a scalable, multi-mastering debug and test interface that can connect power managed components on a platform. It shall…

4. Allow components in the platform to power down (and not break the network) and rejoin the network upon powering up.



Foundation: MIPI Alliance I3C®

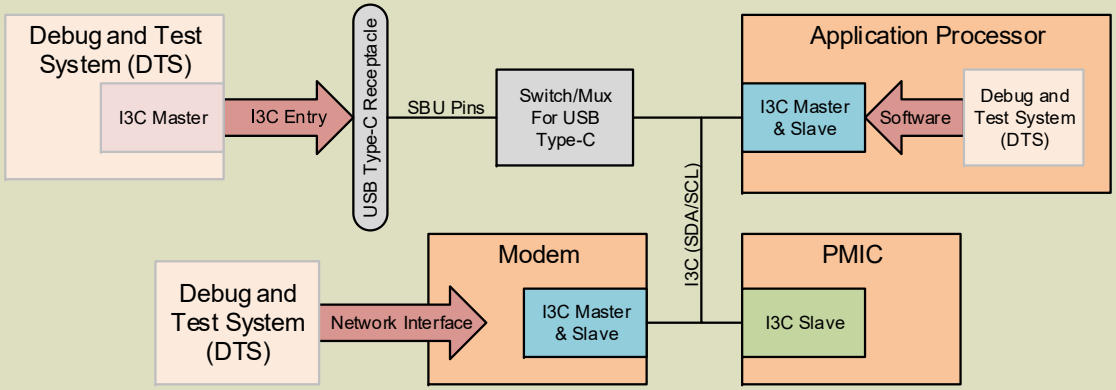
The MIPI Improved Inter Integrated Circuit (I3C) interface has key features that address debug and test requirements.

- **Multi-drop, two wire** interface supporting **multiple masters** and the ability for devices to **hot-join**. Supports clocks up to **12.5MHz** and **high data rate** modes.
- Uses a **master-slave** communication/control method.
  - Supports multiple masters and uses a handoff method to change mastership of the bus.
- Supports **in-band interrupts (IBI)** from the slave to the master.
  - For debug, can be used to indicate changes in states and when trace buffering thresholds have been reached.

- The solution is based on the MIPI I3C Basic℠ v1.0 Specification and the upcoming MIPI I3C® v1.1 Specification.

Solution: MIPI Debug for I3C

The MIPI I3C specification alone helps to achieve our goals…



… but there is more that can be done to help debug/test!

MIPI Debug for I3C Specification

How Debug and Test Uses/Extends I3C

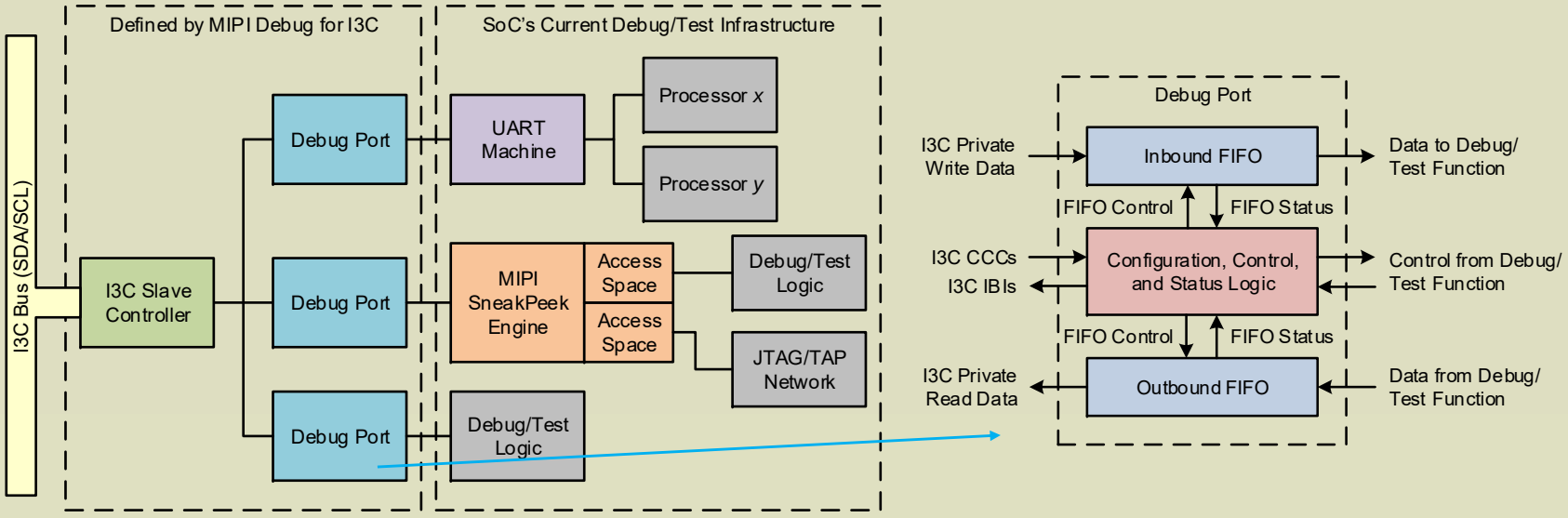
Using the MIPI I3C Specification as the foundation, the solution extends the capabilities specifically for debug and test functionality.

- The MIPI Debug for I3C Specification describes these “extensions” for using the I3C bus interface for debug and test.
- All the features of I3C including 2 wires, hot-join, IBI, multi-drop, broadcast messaging, and multi-mastering are supported.
- It uses existing common command codes (CCC) as well as defines specific CCCs for debug/test configuration, function selection, and action/event triggers and interrupts.
- It also standardizes the data exchange mechanisms on predefined port based communication.

MIPI Debug for I3C Features

- Allows dedicated debug/test or shared bus topologies.
- Handles debug communication over defined byte-oriented streaming interface ports that can support different protocols.
- Allows the target system (TS) to expose multiple debug/test interfaces/ports from a single physical connection.
- Allows the debug and test system (DTS) to send broadcast or directed action requests. (e.g., halt, reset)
- Allow the TS to send event indications via IBIs. (e.g., triggers, requests)

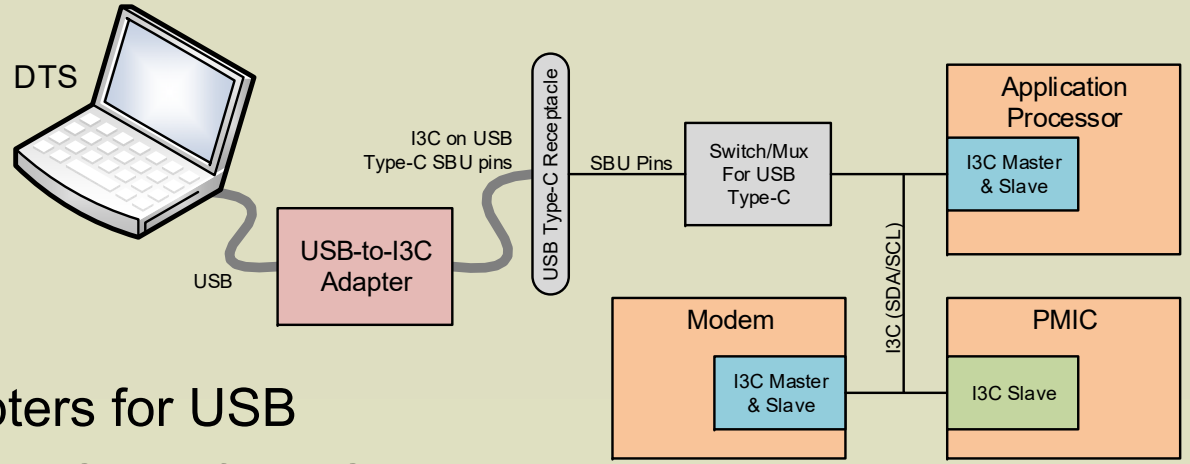
Conceptual Target System Diagram



- The MIPI SneakPeek Engine is defined by the MIPI SneakPeek Protocol Specification.

Besides Hardware Specifications…

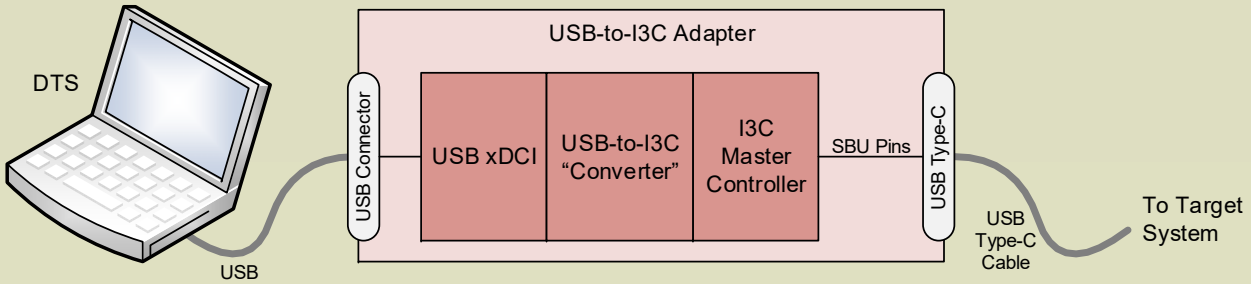
…work is being done to help define tooling and software.



- DTS Adapters for USB
- USB Device Class for I3C
- Software Flows including those for multiple-mastering

DTS Development

- Common (both debug/test and not) USB Device Class for I3C Mastering from the DTS.
- Integrating debug/test extensions into the MIPI I3C Host Controller Interface (HCI[SM]) specification.



On-going Development

- The MIPI Debug WG is targeting Q3 2019 completion for the MIPI Debug for I3C v1.0 Specification.
- USB I3C Device Class Specification in the USB-IF is planned for late 2019 completion.
- The MIPI Software WG is targeting mid 2019 completion of the I3C HCI Specification with the necessary features for debug test.
- The MIPI Debug WG is planning on a software flows whitepaper sometime in 2019.

Summary / Conclusions

- MIPI I3C provides a **scalable, multi-mastering debug/test interface** that can **connect power managed components** on a platform.
  - It meets newer requirements/use cases that legacy interfaces (e.g., JTAG/cJTAG, I2C, UART) do not.
- The MIPI Debug for I3C Specification **extends the I3C bus interface** for debug and test.
  - It includes the features of I3C including 2 wires, hot-join, IBI, multi-drop, broadcast messaging, and multi-mastering. It adds debug/test-specific CCCs and standardizes the data exchange mechanisms.
- Besides the hardware specifications, work is being done on the **DTS tooling and software**.
  - Defining a USB I3C Device Class, adding debug/test extensions to the MIPI I3C HCI specification, and defining software flows.

Call to Action

There is still an opportunity to get involved with the development!

- All are encouraged to get involved in the on-going development.
  - Architects, implementers (IP, component, system), tool vendors, software developers,…
- Increased involvement ensures:
  - Better, wider spread adoption.
  - Larger eco-system for the solution.
  - Lower barriers to implement.

For more information…

- MIPI I3C Web Site:
  https://mipi.org/specifications/i3c-sensor-specification
- MIPI Debug for I3C Web Site:
  https://mipi.org/specifications/debug-i3c
- MIPI Alliance Web Site:
  http://mipi.org
- MIPI Debug WG Public Page:
  https://www.mipi.org/specifications/debug
- MIPI Architecture Overview for Debug:
  https://www.mipi.org/sites/default/files/mipi_Architecture-Overview-for-Debug_v1-2.pdf
- MIPI Debug on Wikipedia:
  http://en.wikipedia.org/wiki/MIPI_Debug_Architecture