

A network diagram with a teal background filled with various mobile-related icons like smartphones, Wi-Fi signals, and globes. A network of white lines connects several colored nodes (white, orange, red, purple, orange, white) across the top and left sides of the slide.

## MIPI Debug for I3C<sup>SM</sup>: The Next Generation Debug Interface

**Enrico Carrieri**

Intel Corporation &  
MIPI Debug Working  
Group Chair

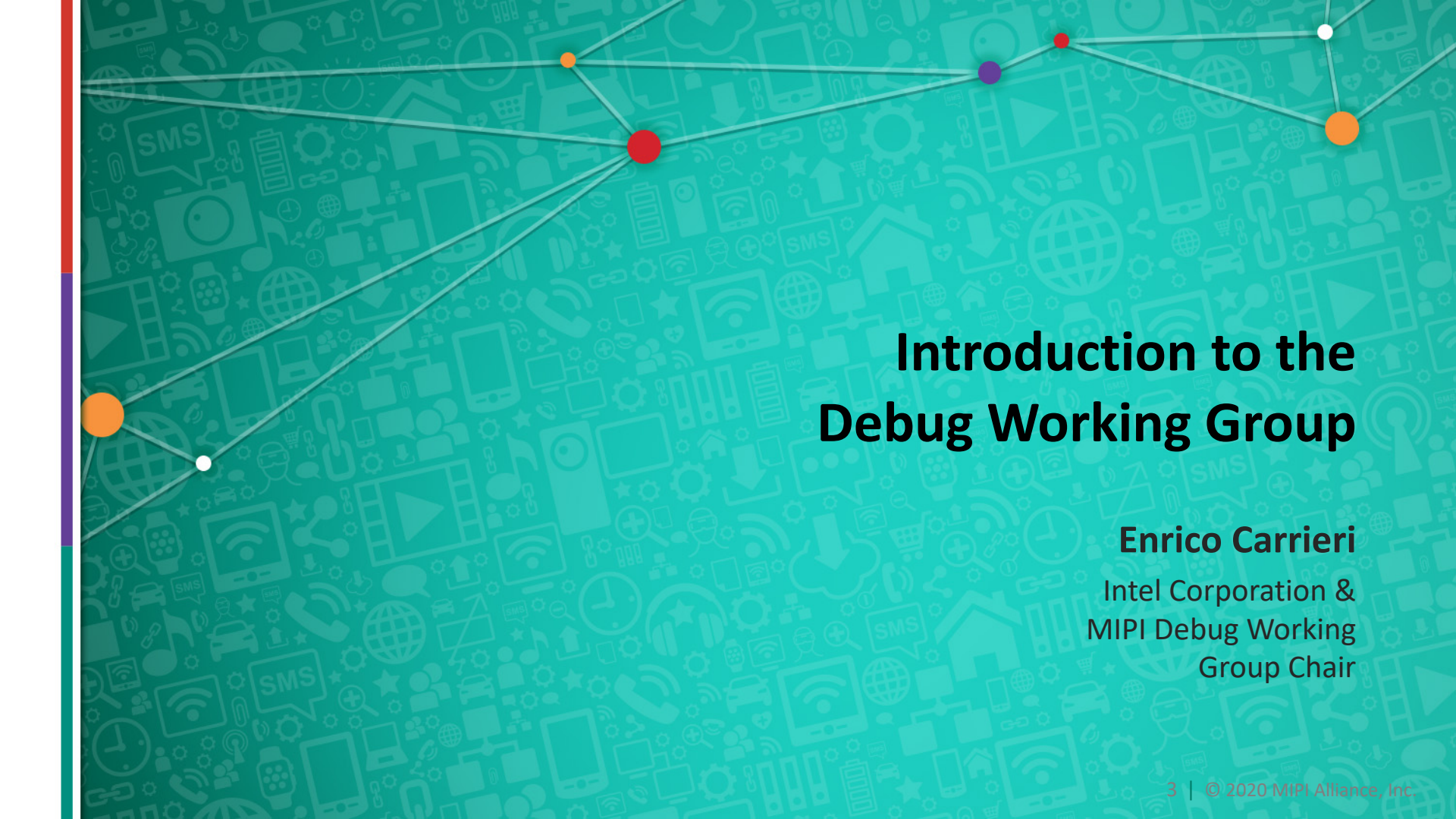
**Matthew Schnoor**

Intel Corporation

# Presentation Agenda

- **Introduction to the Debug Working Group**
- **Problem Statement and Goals**
- **Basic Principles of Debug for I3C**
- **Supported Network Adaptors**
- **Summary and Conclusions**
- **Questions and Answers**





# Introduction to the Debug Working Group

**Enrico Carrieri**  
Intel Corporation &  
MIPI Debug Working  
Group Chair

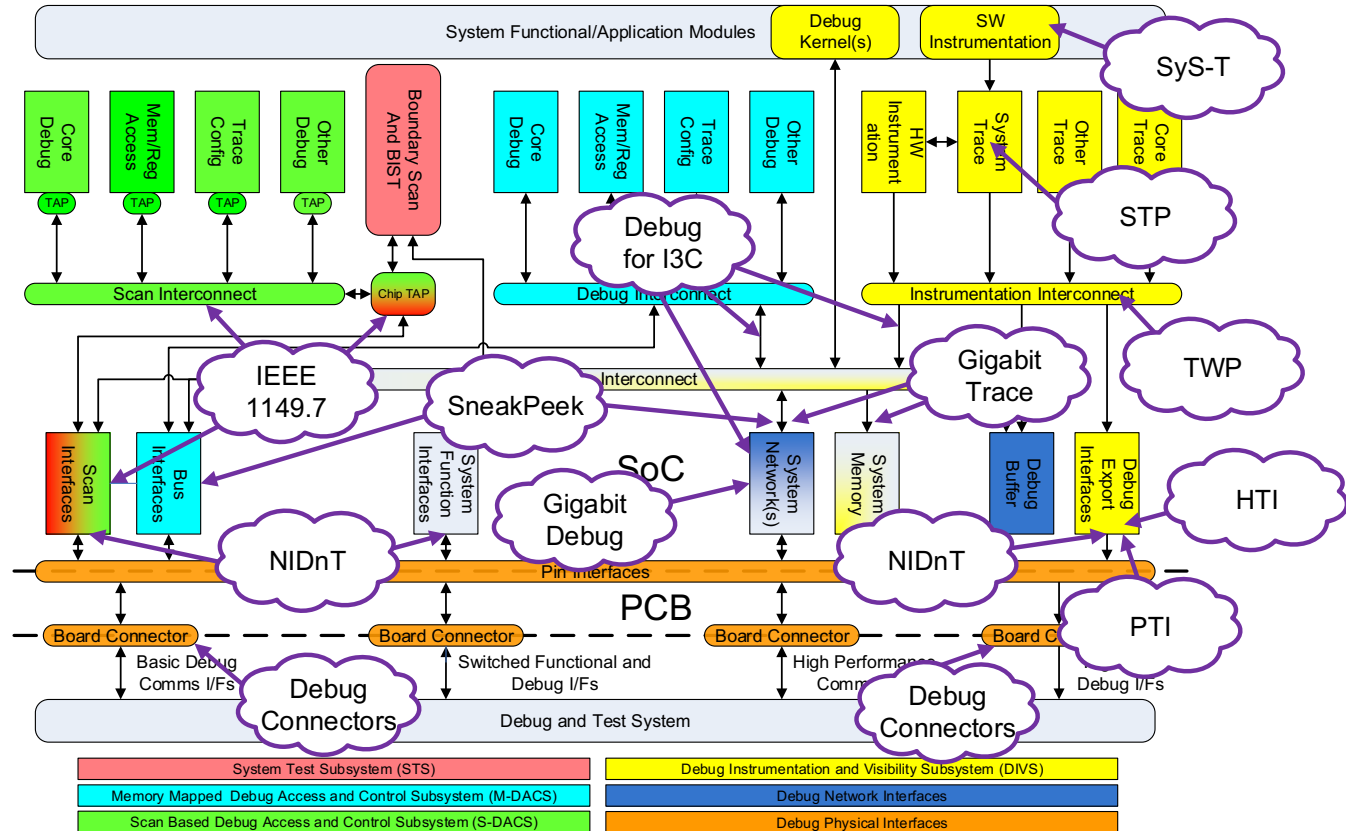
# MIPI Debug Working Group

The focus is to unify/define:

- Protocols that support debug/trace
  - With a particular focus on highly integrated, fielded systems
  - Software layers to support and/or implement these protocols
- Configuration/control mechanisms required directly by debug/trace protocols
- Reuse of functional interfaces and protocols for debug/trace
- Mating connections and pin assignments
- Electrical characteristics

# MIPI Debug Specifications

- Over 10 specifications adopted or in progress
- Plus whitepapers and other documentation
- All MIPI Debug and Trace specifications are available for download





The background is a teal color with a dense pattern of small, light-colored icons representing various digital and communication concepts like Wi-Fi, SMS, a globe, a smartphone, a play button, and a gear. Overlaid on this is a network diagram consisting of several nodes (colored orange, red, purple, and white) connected by thin white lines. The nodes are arranged in a roughly triangular pattern across the top and left sides of the slide.

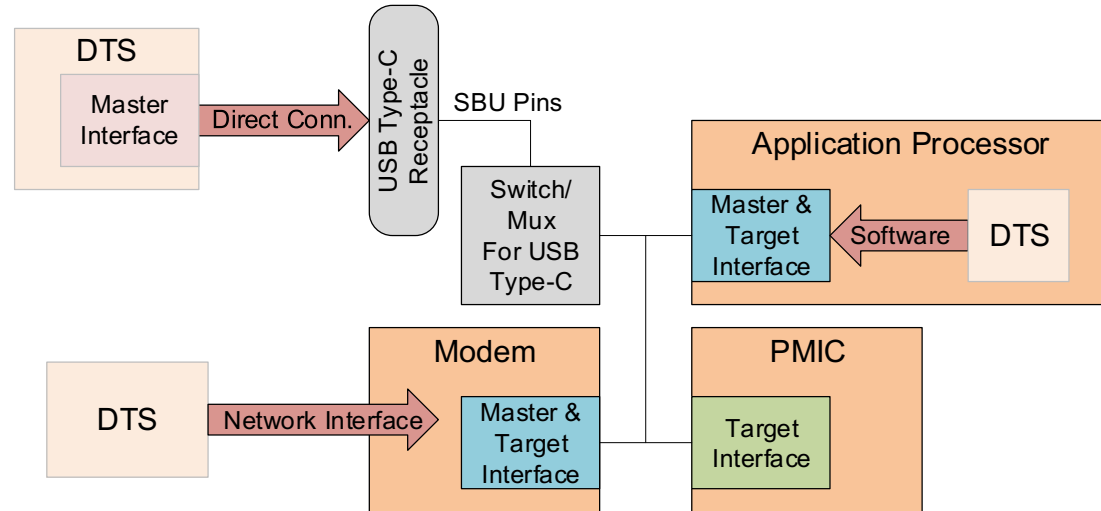
# Problem Statement and Goals

# Problem Statement

Current debug interface solutions (e.g., JTAG/cJTAG, I<sup>2</sup>C, UART) are falling short in meeting new requirements/use cases

For example:

- Want to debug all components on the board
- 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)



# Goal

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

- Require a minimal set of pins (i.e., two wires), including those needed for interrupts
- Allow multi-component connectivity
- Support multiple entry/mastering points to allow flexibility across the platform's lifecycle
- Allow components in the platform to power down (and not break the network) and rejoin the network upon powering up





# Basic Principles of Debug for I3C

# Foundation: MIPI I3C<sup>®</sup>

## 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 **12.5 MHz** and **high data rate modes**
- Uses a “master to target” communication/control method
- Supports multiple masters and uses a handoff method to change mastership of the bus
- Supports **In-Band Interrupts (IBIs)** from the target to the master
- For debug, can be used to indicate changes in states and when trace buffering thresholds have been reached



The Debug for I3C solution can be based on either the MIPI I3C Basic<sup>SM</sup> v1.0 (publicly available) or the MIPI I3C v1.0/1.1 specification.

# How Debug Uses/Extends MIPI I3C

## Using the MIPI I3C specification as the foundation, the solution extends the capabilities specifically for debug functionality

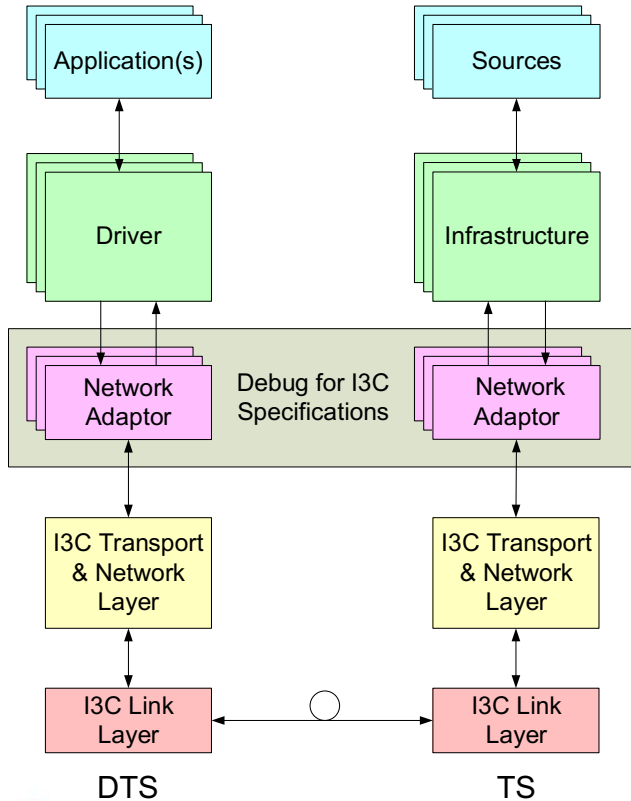
- The MIPI Debug for I3C specification describes “extensions” for using the I3C bus interface for debug
- All the features of I3C including two wires, hot-join, IBI, multi-drop, broadcast messaging, and multi-mastering are supported
- Uses existing Common Command Codes (CCCs), as well as defines specific CCCs for debug configuration, function selection, and action/event triggers and interrupts
- Uses IBIs with specific Mandatory Data Byte (MDB) values indicating debug interrupts
- Standardizes the data exchange mechanisms on predefined Network Adaptor (i.e., port) based communication

# MIPI Debug for I3C Features

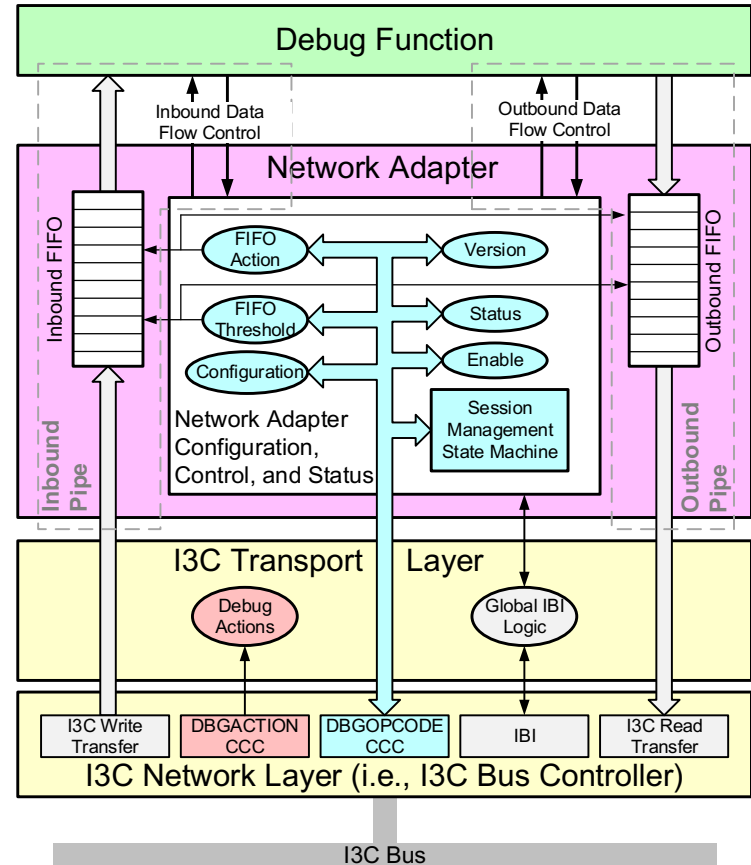
- Allows dedicated debug or shared bus topologies
- Handles debug communication over defined byte-oriented streaming interface ports that can support different protocols
- Allows the Target System (TS) to expose multiple debug interfaces/ports (referred to as Network Adaptors) from a single physical 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 System Diagram

## MIPI Debug WG Functional Layering Diagram



## Debug for I3C Functional Diagram





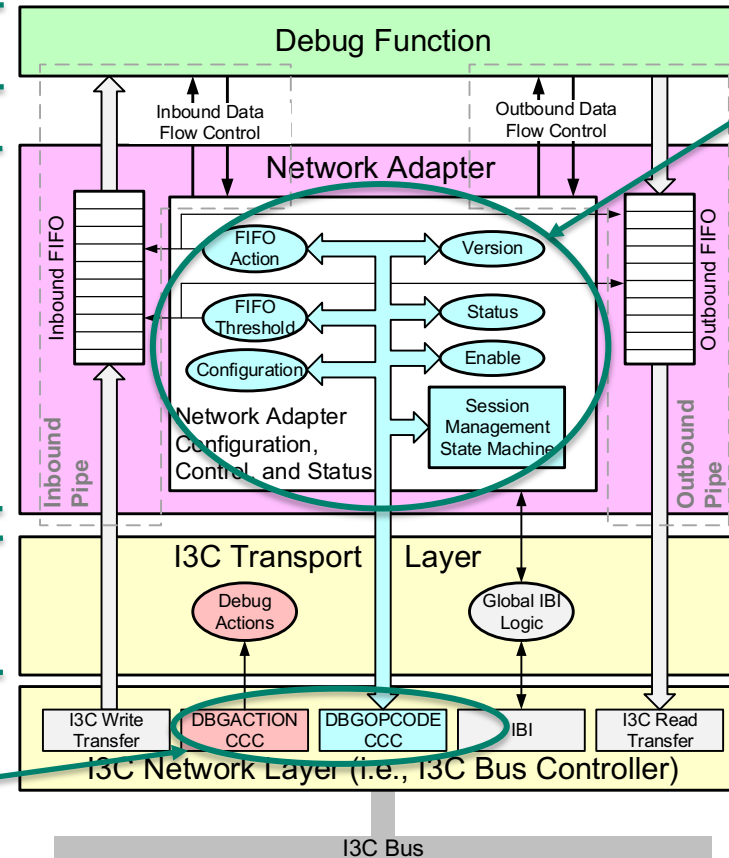
# Conceptual Target System Diagram

e.g., Internal TAP network,  
Debug Registers, Trace  
Infrastructure

Specific to the debug  
function(s) on the given  
SoC. There may be several  
Network Adaptors

Debug-specific  
Transport Layer Logic  
(i.e., beyond base I3C)

Debug “Extensions”



Common Logic per  
Network Adaptor  
“Extensions”

Defined by the Debug for  
I3C specification

# Debug Common Command Codes (CCCs)

## Defined CCCs specifically for use by the debug logic:

- **DBGOPCODE** – Direct CCC (0xD7) used to request a particular operation of a given Network Adaptor that is part of the TS
  - Read the debug capabilities of the TS, configure the Network Adaptors, control the debug session, and control the FIFOs
- **DBGACTION** – Direct (0xD8) or Broadcast (0x58) CCC used to initiate a particular debug action(s) on a single (direct) or all (broadcast) given TS instances on the I3C bus
  - e.g., issue a debug reset, issue a halt, start trace

# Debug In-Band Interrupts

## Defined three mandatory data byte (MDB) identifiers specifically for debug:

- **DBGSTATUS** – (MDB = 0x5C) Used to indicate a change in status
  - e.g., FIFO threshold met, session stopped, processor halted
- **DBGERROR** – (MDB = 0x5D) Used to indicate an error condition
  - e.g., I3C transport layer error or an error in the Network Adaptor
- **DBGDATAREADY** – (MDB = 0xAD) Used to indicate data is ready in a given outbound FIFO
  - A specific Pending Read Notification for debug data

The background is a teal color with a dense pattern of small, light-colored icons representing various digital and network-related concepts such as Wi-Fi, SMS, mobile phones, and network nodes. Overlaid on this is a network diagram consisting of several nodes (colored orange, red, purple, and white) connected by thin white lines. The nodes are arranged in a roughly horizontal line across the top and middle of the page, with some lines extending downwards and outwards.

# Supported Network Adaptors

# Network Adaptor

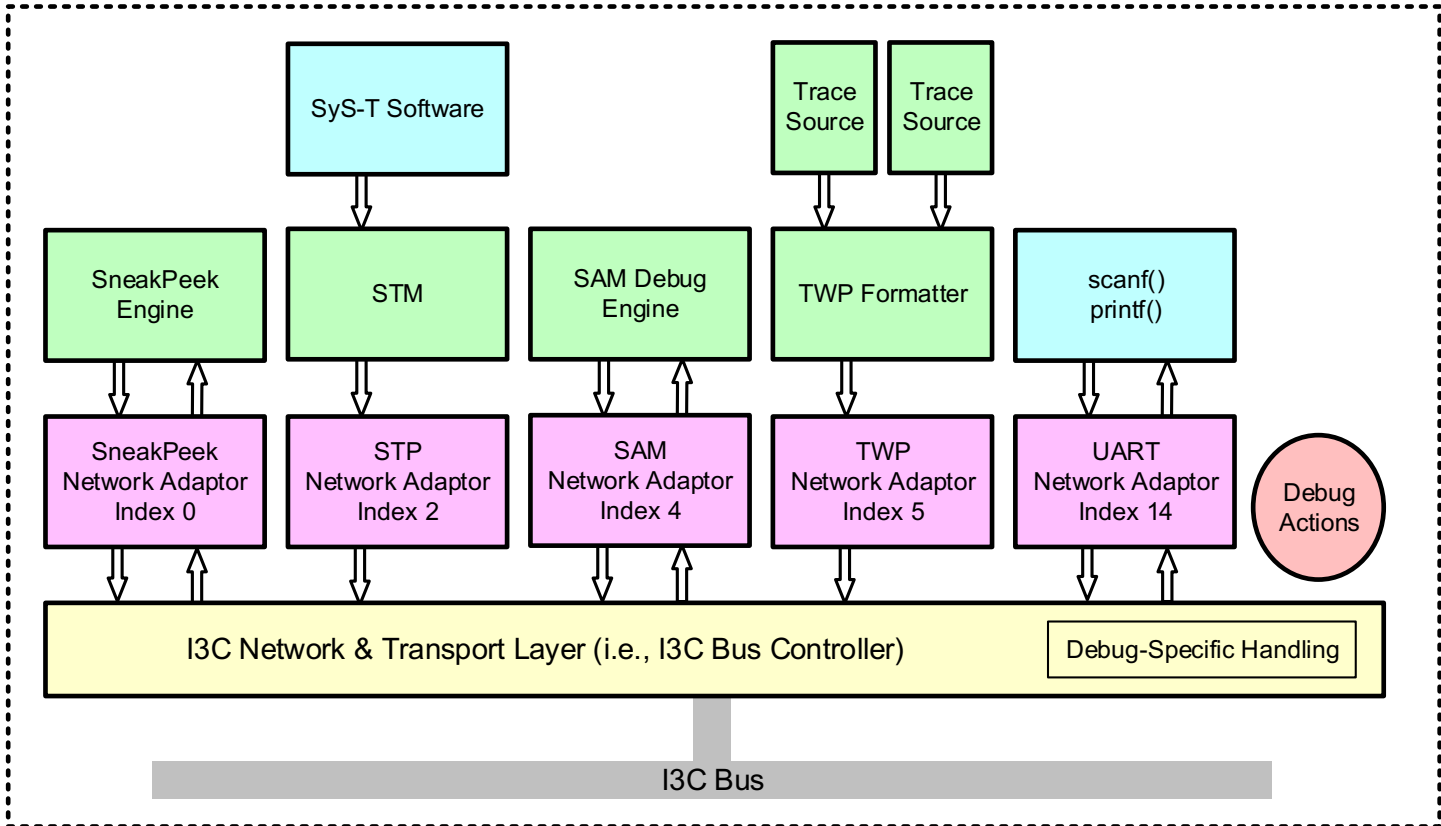
- A mechanism used for communicating with debug functions within the TS
- Contains either an inbound data pipe, an outbound data pipe, or both
  - Accessed via I3C private write and read transfers
- Mapped to a single instance of a debug function with a particular protocol
  - Maximum 16 instances
  - Each network adaptor does not have to use a unique protocol



# Supported Network Adaptor Protocols

Protocol	Direction	Details
SneakPeek Protocol (SPP)	Inbound & Outbound	Used for debug communication to a SneakPeek Command Engine. Bi-directional transfer of blocks of bytes (SPTBs) formatted to TinySPP
Trace Wrapper Protocol (TWP)	Outbound	Used for trace data output from trace infrastructure. Uni-directional transfer of a stream of bytes formatted to TWP
System Trace Protocol (STP)	Outbound	Used for trace data output from a System Trace Macrocell. Uni-directional transfer of a stream of bytes formatted to STP
Simplified Address-Mapped Protocol (SAM)	Inbound & Outbound	Used for communication with simple address-mapped access to TS resources
UART	Inbound & Outbound	Used for communication with character-oriented software agents such as <code>scanf()</code> / <code>printf()</code> , or a GDB monitor. Bi-directional transfer of bytes
Implementation-Defined	Inbound & Outbound	Implementation-defined function not described in this specification

# Example Implementation of Network Adaptors



The background is a teal color with a dense pattern of small, light-colored icons representing various digital and communication concepts, such as Wi-Fi signals, SMS messages, mobile phones, and social media symbols. Overlaid on this background is a network diagram consisting of several nodes (colored circles) connected by thin white lines. The nodes are located at various points: one orange node on the left edge, one white node below it, one red node in the upper-middle, one purple node to its right, one orange node on the right edge, and one white node at the top right. The text 'Summary and Conclusions' is centered in the middle of the slide.

# Summary and Conclusions

# Summary and Conclusions

- MIPI I3C provides a **scalable, multi-mastering debug interface** that can **connect power-managed components** on a platform:
  - Meets newer requirements/use cases that legacy interfaces (e.g., JTAG/cJTAG, I<sup>2</sup>C, UART) do not
- The MIPI Debug for I3C specification **extends the I3C bus interface** for debug:
  - Includes the features of I3C including two wires, hot-join, in-band interrupts, multi-drop, broadcast messaging and multi-mastering. It adds debug/test-specific CCCs and standardizes the data exchange mechanisms

# For More Information

MIPI Alliance:	<a href="http://www.mipi.org">http://www.mipi.org</a>
MIPI Debug Working Group Public Page:	<a href="https://www.mipi.org/specifications/debug">https://www.mipi.org/specifications/debug</a>
MIPI Architecture Overview for Debug:	<a href="https://www.mipi.org/sites/default/files/mipi_Architecture-Overview-for-Debug_v1-2.pdf">https://www.mipi.org/sites/default/files/mipi_Architecture-Overview-for-Debug_v1-2.pdf</a>
MIPI I3C Web Page:	<a href="https://www.mipi.org/specifications/i3c-sensor-specification">https://www.mipi.org/specifications/i3c-sensor-specification</a>
MIPI Debug for I3C Web Page:	<a href="https://www.mipi.org/specifications/debug-i3c">https://www.mipi.org/specifications/debug-i3c</a>



The background is a teal color with a dense pattern of small, light-colored icons representing various digital and communication concepts such as Wi-Fi, SMS, mobile phones, and social media. Overlaid on this is a network diagram consisting of several nodes (colored circles in orange, red, purple, and white) connected by thin white lines. The nodes are arranged in a roughly horizontal line across the top and middle of the page, with some lines extending downwards.

# Questions & Answers