A Deep Dive into MIPI Debug for I3C℠
Webinar Agenda

- Overview of MIPI Debug for I3C
- Network Adaptors
- Debug Common Command Codes (CCCs)
- Session Management and Debug Resets
- I3C In-Band Interrupts for Debug Events
- Debug Ecosystem
- Conclusions
A Deep Dive into
MIPI Debug for I3C℠

Matthew Schnoor
Intel Corporation
MIPI Debug Working Group
About MIPI Debug for I3C

Enables new debug interface standard, where existing interface solutions (e.g., JTAG/cJTAG, I2C, UART) are falling short

- Enables debug of multiple components on the board
- Debug and test system (DTS) can attach 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)
MIPI Debug for I3C Key Features

- Uses core capabilities of MIPI I3C® or MIPI I3C Basic℠
- Allows dedicated 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

- Application(s)
- Sources

Driver

- Network Adaptor
  - Debug for I3C Specifications

- Infrastructure

I3C Transport & Network Layer

- I3C Link Layer

Network Adaptor

- Inbound Data Flow Control
- Outbound Data Flow Control

Debug Function

- FIFO Action
- Version

Network Adaptor Configuration, Control, and Status

- Status
- Enable
- Session Management State Machine

I3C Transport Layer

- Debug Actions
- Global IBI Logic

I3C Network Layer (i.e., I3C Target)

- 13C Write Transfer
- DBGACTION CCC
- DBGOPCODE CCC
- IBI
- I3C Read Transfer

I3C Bus

Debug for I3C Functional Diagram

- Inbound Pipe
- Outbound Pipe
- Inbound FIFO
- Outbound FIFO

DTS

TS
Conceptual Target System Diagram

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

Specific to the debug function(s) on the given SoC. TS may have 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
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 Private 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

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Direction</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>SneakPeek Protocol (SPP)</td>
<td>Inbound &amp; Outbound</td>
<td>Used for debug communication to a SneakPeek Command Engine. Bi-directional transfer of blocks of bytes (SPTBs) formatted to TinySPP</td>
</tr>
<tr>
<td>Trace Wrapper Protocol (TWP)</td>
<td>Outbound</td>
<td>Used for trace data output from trace infrastructure. Uni-directional transfer of a stream of bytes formatted to TWP</td>
</tr>
<tr>
<td>System Trace Protocol (STP)</td>
<td>Outbound</td>
<td>Used for trace data output from a System Trace Macrocell. Uni-directional transfer of a stream of bytes formatted to STP</td>
</tr>
<tr>
<td>Simplified Address-Mapped Protocol (SAM)</td>
<td>Inbound &amp; Outbound</td>
<td>Used for communication with simple address-mapped access to TS resources</td>
</tr>
<tr>
<td>UART</td>
<td>Inbound &amp; Outbound</td>
<td>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</td>
</tr>
<tr>
<td>Implementation-Defined</td>
<td>Inbound &amp; Outbound</td>
<td>Implementation-defined function not described in this specification</td>
</tr>
</tbody>
</table>
Example Implementation of Network Adaptors

- SneakPeek Engine
  - SneakPeek Network Adaptor Index 0
- STM
  - STP Network Adaptor Index 2
- SAM Debug Engine
  - SAM Network Adaptor Index 4
- TWP Formatter
  - TWP Network Adaptor Index 5
- UART
  - UART Network Adaptor Index 14

- I3C Network & Transport Layer (i.e., I3C Target)
  - Debug-Specific Handling

- I3C Bus

- Debug Actions
  - scanf()
  - printf()
Network Adaptor Details: SPP

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Direction</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>SneakPeek Protocol (SPP)</td>
<td>Inbound &amp; Outbound</td>
<td>Used for debug communication to a SneakPeek Command Engine. Bi-directional transfer of blocks of bytes (SPTBs) formatted to TinySPP</td>
</tr>
</tbody>
</table>

SPP provides a richer address-mapped mechanism for read+write transactions, to replace dedicated interfaces (such as JTAG).

Each **SneakPeek Transfer Block** (SPTB) is a single I3C Private Write or Private Read transfer.

Reference: [https://www.mipi.org/specifications/spp](https://www.mipi.org/specifications/spp)
Network Adaptor Details: SPP

SPP write: SPTB (Command Packets)
DTS → TS

SPTB #1 (with Command Packets)

SPP Payload from DTS to TS

Network Adaptor Details: SPP

SPP write: SPTB (Command Packets)
DTS → TS
Network Adaptor Details: SPP

**SPP read: SPTB**
*(Response Packets)*

DTS ← TS

---

**SPTB #1 with Response Packets**

---

**SPP Payload from TS to DTS**
## Network Adaptor Details: TWP

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Direction</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Wrapper Protocol (TWP)</td>
<td>Outbound</td>
<td>Used for trace data output from trace infrastructure. Unidirectional transfer of a stream of bytes formatted to TWP</td>
</tr>
</tbody>
</table>

TWP enables trace data output from multiple trace sources.

TWP Frames can be spread across one or more I3C Private Read transfers.

Reference: [https://www.mipi.org/specifications/twp](https://www.mipi.org/specifications/twp)
Network Adaptor Details: TWP

TWP read Frame(s):
DTS ← TS
Network Adaptor Details: STP

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Direction</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Trace Protocol (STP)</td>
<td>Outbound</td>
<td>Used for trace data output from a System Trace Macrocell. Uni-directional transfer of a stream of bytes formatted to STP</td>
</tr>
</tbody>
</table>

STP enables trace data output from application-specific trace protocols

STP Packets can be spread across one or more I3C Private Read transfers

Reference: [https://www.mipi.org/specifications/stp](https://www.mipi.org/specifications/stp)
Network Adaptor Details: STP

STP read Packet(s):
DTS ← TS

STP Packet n
STP Packet n + 1
STP Packet n + 2 (Synchronization)
STP Packet n + 3
STP Packet n + 4
Network Adaptor Details: SAM

SAM provides a simple address-mapped mechanism for read+write transactions, intended for register or memory access.

SAM commands can be GET, SET, READ or WRITE.

• SET + WRITE commands use one I3C Private Write.

• GET + READ commands use “Write-then-Read” semantics: one I3C Private Write followed by one I3C Private Read.
Network Adaptor Details: SAM

SAM SET or WRITE command example:
DTS → TS
Network Adaptor Details: SAM

SAM GET or READ command example:
DTS ↔ TS

Write then Read

GET or READ Commands

SAM Payload from DTS to TS

SAM Payload from TS to DTS
Debug Common Command Codes (CCC)

Defined CCCs specifically for use by the debug logic:

- **DBGOPCODE** – Direct CCC (0xD7)
  - Request a particular operation of a given Network Adaptor that is part of the TS.
  - Can be Write only, or **Write-then-Read**, or both (per opcode)

- **DBGACTION** – Direct (0xD8) or Broadcast (0x58)
  - Initiate one or more particular debug actions on a single TS (Direct) or all TS instances (Broadcast) on the I3C bus
Debug Common Command Codes (CCC)

- **DBGOPCODE** – Direct CCC (0xD7) opcodes:

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Format</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPABILITIES</td>
<td>Read only (Write-then-Read)</td>
<td>Read Debug for I3C version and capabilities</td>
</tr>
<tr>
<td>CFG</td>
<td>Write / Read</td>
<td>Configure polling vs. interrupts</td>
</tr>
<tr>
<td>START_NA</td>
<td>Write / Read</td>
<td>Network Adaptor start session (and status)</td>
</tr>
<tr>
<td>STOP_NA</td>
<td>Write / Read</td>
<td>Network Adaptor stop session (and status)</td>
</tr>
<tr>
<td>FIFO_THRESHOLD</td>
<td>Write / Read</td>
<td>Network Adaptor FIFO threshold settings</td>
</tr>
<tr>
<td>FIFO_ACTION</td>
<td>Write / Read</td>
<td>Network Adaptor FIFO other actions</td>
</tr>
<tr>
<td>SELECT</td>
<td>Write only</td>
<td>Select Network Adaptor</td>
</tr>
<tr>
<td>REPORT_ERROR</td>
<td>Write only</td>
<td>Inform Network Adaptor of received error</td>
</tr>
</tbody>
</table>
Debug Common Command Codes (CCC)

- **DBGACTION** – Direct (0xD8) or Broadcast (0x58)
  - Build a set of actions, in continuous CCC framing

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>DBGRST</td>
<td>Debug Reset</td>
</tr>
<tr>
<td>0x01</td>
<td>STARTSET</td>
<td>Start a set of actions</td>
</tr>
<tr>
<td>0x02</td>
<td>EXECSET</td>
<td>Execute the set of actions</td>
</tr>
<tr>
<td>0x03 – 0x7F</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td>0x80 – 0xFF</td>
<td>IMPDEF</td>
<td>Implementation defined</td>
</tr>
</tbody>
</table>
Network Adaptor Session Management

Concept of Session Management

- Each Network Adaptor is managed by its internal state machine
- DTS can start/stop transactions by using DBGOPCODE CCC

- **Specific meanings of TS state transitions will depend on the Network Adaptor type.**
Debug Resets

• How to reset Debug logic in TS?
  - DBGACTION CCC with ‘DBGRST’ value 0x00

  or

  - RSTACT CCC (w/ Defining Byte 0x03) followed by I3C Target Reset Pattern
Debug Resets and Layers

**Mechanism:**
- DBG ACTION CCC DBGRST Action (I3C v1.0 or later)
- Target Reset with RSTACT Defining Byte of 0x03 (I3C v1.1 or later)

**Mechanism:**
- Target Reset of Whole Chip (I3C v1.1 or later)

**Mechanism:**
- Target Reset of I3C Peripheral (I3C v1.1 or later)
Debug In-Band Interrupts

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

- **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
### Debug In-Band Interrupts (example)

<table>
<thead>
<tr>
<th>Target (TS) Address / R (i.e. IBI from TS) / ACK (ActvCtrl)</th>
<th>Debug IBI DBGSTATUS Cause / T=1</th>
<th>Network Adaptor Index and Sub-Cause / T</th>
<th>Additional Data[0..N] (Optional) / T</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x5C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b010 5'b11100</td>
<td></td>
</tr>
<tr>
<td>0x5D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x5D</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3'b010 5'b11101</td>
<td></td>
</tr>
</tbody>
</table>

### Additional Data
- [0..N] (Optional)

### Network Adaptor Index and Sub-Cause
- / T

### Target Addr
- / RnW / ACK

### Additional Data
- / T

### Target Addr
- 7'h7E / W / ACK

---

**End of this IBI readback**
Debug In-Band Interrupts (example)

Data sent with I3C Pending Read Notification contract, initiated by I3C Controller after IBI

Recommended… (SDR example shown)
Debug Ecosystem Components

• I3C Target interface extensions
  – Implementing support for Debug CCCs, IBIs and connection to debug functions via Network Adaptors

• I3C Controller interface to DTS Host
  – MIPI I3C HCl℠ available for a standard Host Controller
  – USB-IF Device Class standard is in development...

• Debug connectors that include I3C serial pins
  – Not yet defined, stay tuned...
Conclusions

• MIPI I3C® provides a scalable, multi-mastering debug 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:
  – Builds on the core I3C capabilities, including two wires, hot-join, IBI, multi-drop, broadcast messaging and multi-mastering
  – Adds debug/test-specific CCCs and standardizes the data exchange mechanisms
For more information...

- MIPI Alliance website: [http://mipi.org](http://mipi.org)
- MIPI Debug WG page: [https://www.mipi.org/specifications/debug](https://www.mipi.org/specifications/debug)
- MIPI Architecture Overview for Debug whitepaper: [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)
- MIPI I3C page: [https://mipi.org/specifications/i3c-sensor-specification](https://mipi.org/specifications/i3c-sensor-specification)
- MIPI Debug for I3C page: [https://mipi.org/specifications/debug-i3c](https://mipi.org/specifications/debug-i3c)
Questions and Answers
Thank you!
# Network Adaptors: Key Differences

<table>
<thead>
<tr>
<th>Capability</th>
<th>SneakPeak (TinySPP)</th>
<th>SAM</th>
<th>UART</th>
</tr>
</thead>
<tbody>
<tr>
<td>Separate debug functions per Adaptor</td>
<td>Up to 8</td>
<td>Up to 16</td>
<td>1</td>
</tr>
<tr>
<td>Function discovery and identifiers</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Function handling</td>
<td>Multiple methods</td>
<td>Address/Data only</td>
<td>N/A</td>
</tr>
<tr>
<td>Addressable space</td>
<td>64-bit</td>
<td>32-bit</td>
<td>N/A</td>
</tr>
<tr>
<td>Commands provided</td>
<td>Rd, Wr, Wr+Rd, Loops, Triggers</td>
<td>Rd, Wr</td>
<td>N/A</td>
</tr>
<tr>
<td>Largest transfer size (bytes)</td>
<td>$2^{7} - 1$</td>
<td>$2^{16} - 1$</td>
<td>N/A</td>
</tr>
<tr>
<td>Address pointers (per function)</td>
<td>Yes</td>
<td>Optional</td>
<td>N/A</td>
</tr>
<tr>
<td>Many more advanced capabilities</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Debug Ecosystem

• DTS connections via USB
  – Developing new USB-IF Device Class for I3C Controller
  – Work is in progress...