
# Modular-Peripheral Sideband Tunneling Interface (M-PESTI) Base Specification

Part of the

DC-MHS (Data Center Modular Hardware System) Revision 1.0 family

Version 1.0 Release Candidate 2

September 23, 2022

## M-PESTI Revision 1.0 Primary Editors/Contributors:

**Dell, Inc:** Jeff Kennedy, Tim Lambert, Shawn Dube, Charlie Ziegler

**Intel Corporation:** Javier Lasa, Eduardo Estrada, Clifford Dubay, Aurelio Rodriguez Echevarria, Brian Aspnes

**Microsoft Corporation:** Priya Raghu, Priscilla Lam

**Advanced Micro Devices, Inc:** Greg Sellman

# Table of Contents

# 1. License

## 1.1 Open Web Foundation (OWF) CLA

Contributions to this Specification are made under the terms and conditions set forth in Open Web Foundation Modified Contributor License Agreement ("OWF CLA 1.0") ("Contribution License") by:

- Advanced Micro Devices, Inc
- Dell, Inc.
- Google LLC
- Hewlett Packard Enterprise Company
- Intel Corporation
- Meta Platforms, Inc.
- Microsoft Corporation

Usage of this Specification is governed by the terms and conditions set forth in **Open Web Foundation Modified Final Specification Agreement ("OWFa 1.0") ("Specification License").**

You can review the applicable OWFa1.0 Specification License(s) referenced above by the contributors to this Specification on the OCP website at http://www.opencompute.org/participate/legal-documents/. For actual executed copies of either agreement, please contact OCP directly.

**Notes**:

1. The above license does not apply to the Appendix or Appendices. The information in the Appendix or Appendices is for reference only and non-normative in nature.

## 1.2 Acknowledgements

The Contributors of this Specification would like to acknowledge the following companies for their feedback:

With the hope of making this specification useful for the entire OCP community, we acknowledge and appreciate the contributions, review, and feedback of various individuals and companies that participated in DC-MHS.

# 2. Version Table

| Date | Version # | Description |
|---|---|---|
| 4/22/22 | v0.84 | Initial public release |
| 6/22/22 | v0.85 | -Relaxed tAPTAR timing to align to M-CRPS requirements.<br>-Revised Source to Destination detection algorithm to remove 'misses.'<br>-Added Fanout support chapter |
| 8/11/22 | v0.90 | Added Fanout dead lock handling guidance<br>General cleanup<br>Removed Future Work section<br>Added contributor name |
| 9/19/22 | V1.0 Release Candidate | Added security section<br>Clarified width field refers to connector width and not AIC<br>Clarified fanout mode status and control |
| 9/23/22 | V1.0 Release Candidate2 | Added new intro section<br>Removed Supplemental B. It was M-XIO supported configs unrelated to M-PESTI base spec<br>Numerous minor rewordings based on feedback for clarifications. |

# 3. Introduction and Scope

This Modular Peripheral Sideband Tunnelling Interface (M-PESTI) specification includes base requirements for electrical and protocol compatibility between components of a DC-MHS platform. The M-PESTI protocol overloads a common ***PRSNT#*** signal with additional capabilities beyond simple presence/absence of a peripheral. Many of the companion DC-MHS specifications require implementation of M-PESTI on I/O and power connectors at the Host Processor Module (HPM). However, it is not a requirement for any peripheral to implement the M-PESTI target capability. A peripheral that asserts a static low (present) on any ****PRES_PESTI_N*** signal is supported and DC-MHS compliant. This document specifies the electrical and protocol requirements for bi-directional communication between a M-PESTI initiator and target on top of that same presence signal.

## Benefits of M-PESTI Protocol

M-PESTI is a simple half-duplex protocol that can be implemented in a low-cost Microcontroller Unit (MCU) or Complex Programmable Logic Device (CPLD.) Nearly all low-cost MCUs implement a Universal Asynchronous Receiver-Transmitter (UART) as a communication peripheral. UARTs have a long history as a serial communication interface within and between compute systems. Although not required to be implemented in a PLD, its simplicity enables a logic device on a HPM to detect presence of all M-PESTI peripherals and collect physical attributes of those devices without any firmware or high-level processor dependence. Presence detection and collection of attributes describes the M-PESTI ***discovery*** phase of the protocol. Discovery can occur at initial power on of the system while the Baseboard Management Controller (BMC) is still booting.

A general-purpose modular system may have a variety of PCIe CEM risers and storage subsystem configurations. The multitude of configurations results in a large matrix of source to destination cabled couplings that may include independent data fabric and power cables. In a modular system, a single I/O connector on the HPM may have a dozen or more possible destinations. Traditionally, the number of configurations has been limited and it has been feasible to hard code the PCIe root port interface types and bifurcations for each configuration detected. It has been increasingly difficult for system firmware (e.g., BIOS) to uniquely identify one of many configurations and initialize the system accordingly. With many HPM I/O connectors near one another, it is virtually impossible to impose specific/limited cable source to destination couplings based on physical cable length.

M-PESTI discovery of attributes, including data fabric routing to one or more end points (destination) on a riser from a particular root port (source,) enable system FW components to configure and manage the system without requiring a-priori knowledge of peripherals and configurations to support them.

Example1: A storage backplane that directly describes the data fabric routing and bifurcations to multiple storage device slots can be supported by a BIOS that has not been previously introduced to that subsystem. If those subsystems attributes are discoverable, the BIOS is not

required to implement a table that matches a backplane identifier to a fixed set of configuration settings.

Example2: A cabled riser with multiple PCIe CEM slots may include support for one or more of those slots to be powered in the system standby or soft off state as defined by the Advanced Configuration and Power Interface (ACPI) standard. Data Processing Unit (DPU,) Infrastructure Processing Unit (IPU) and Smart Network Interface Card (SmartNIC) are examples of AIC devices that may have independent compute resources and operating systems that do not require a traditional host processor to be on to function. Because not every PCIe CEM Add-In-Card (AIC) supports being powered in the system standby state, it becomes imperative to identify which power cable path is coupled to that device so that a power enable signal can be asserted by system management FW along that path when desired.

M-PESTI bi-directional communication protocol has a command and response structure.
In addition to discovery of peripherals and their cabled source to destination couplings, the M-PESTI protocol supports the exchange of data. Data exchange of virtual wires as well as control requests from an initiator and status responses from a target is possible during the ***active*** phase of the protocol. The active phase is identified by the successful completion of the discovery phase. Data exchange between initiator and target is not a requirement of the M-PESTI active phase.

## Document Scope

This document defines the base technical specification for the DC-MHS Peripheral Sideband Tunneling Interface (M-PESTI). Any supplier seeking OCP recognition for a hardware product based on this spec must be 100% compliant with any and all features or requirements described in this specification.

Main objectives of this specification:

- Establish a standard method for discovery of subsystem, self-describing attributes, and status (e.g., versus a priori knowledge, hard coding firmware and BIOS for fixed or limited configurations). Examples include a vendor/module class, physical bus connectivity descriptions, add-in card presence and precise source to destination cable coupling determination.
- To make common and minimize the number of physical sideband signals between baseboards and various system interconnects, while extending potential applications/subsystem types via software-defined, real-time, multiplexed virtual wires. Exploit transistors and programmable hardware/firmware over wasted, static, near static or custom/form factor specific and non-scalable, physical connectors and cable pins/wires. The benefits include pay-as-you go models, greater density, higher quality, and message over only signal integrity.

M-PESTI Is:

- A generic and extensible 1-wire, bidirectional circuit, and protocol for applications such as cabled high speed I/O interposers, managed power distribution, cooling subsystems and control panels.

- An enabler to maximize hardware leverage/re-use via a “plug-n-code” model (versus plug-n-play) for new systems, configurations, and applications. Examples include vendor-defined virtual wires, and vendor/class code-based discovery.
- A protocol that includes independent commands for static payloads during discovery, dynamic virtual wires and broadcast virtual wires that have error detection via byte parity and message checksum.
- A protocol overlayed on a physical presence signal thereby avoiding a signal tax.
- Frames are based on standard UARTs to support ubiquitous MCU targets.
- Could optionally replace the need for a physical FRU SEEPROM in some applications

M-PESTI Is not:

- Replacing non-real-time interfaces such as SMBUS/I3C.
- Intended to provide large static payloads to optimize hardware cost.
- Ultra-fast or differential signaling
- Teaching of M-PESTI target FW update, attestation, or encryption methods.
- Explicitly defined for extension into standard end form factors.
- Does not explicitly support peripheral card hot plug
- Leveraging an existing standard (proprietary 1-wire or standard multi-wire).
- Specific guaranteed latency when higher-level traffic patterns are used, such as a repeated discovery phase in between active phase and virtual wire exchanges.
- Point-to-point with optional MUXed fanout support (via opcode snooping agent)

## 3.1 Items not in Scope of Specification

- The method and location for storage and retrieval of payload data by consumers
- The method for asserting the source detection stimulus to each M-PESTI wire.
- The method and location for storage and retrieval of source to destination information by consumers
- Error handling and reporting of unassigned/missed sources due to a required cable not being coupled from source to destination.

## 3.2 M-PESTI Application Examples

In typical applications, M-PESTI may be used for an interposer/riser/paddle card to self-describe its HW capabilities and tunnel virtual sideband wires between local logic and end form factor(s) with the base system. In the below diagrams M-XIO is a name for high speed I/O interconnect.

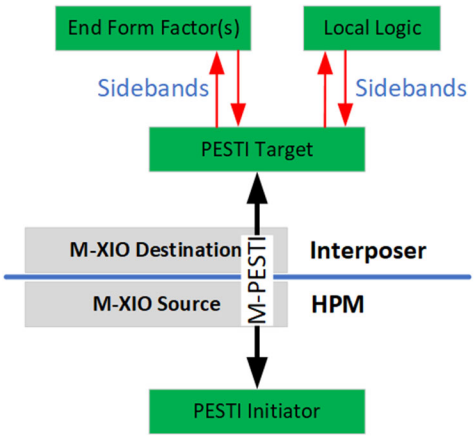



Figure 1. Example M-PESTI Application

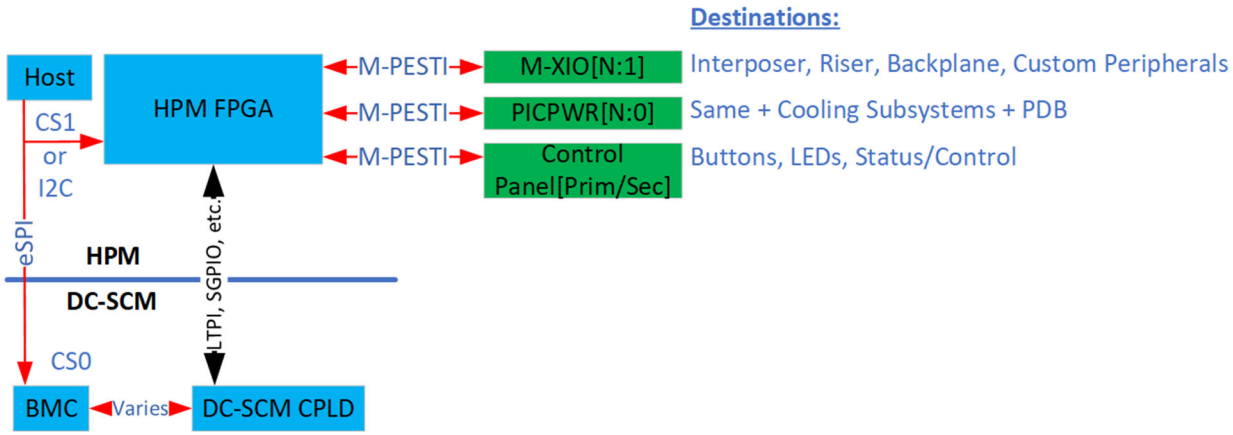



Figure 2. Example M-PESTI applications in the DC-MHS context

In this context, DC-SCM means Datacenter Secure Control Module. HPM is a Host Processor Module. eSPI is an enhanced SPI interface. M-XIO is a modular extensible I/O interconnect (i.e., cabled PCIe). PICPWR is a platform infrastructure connectivity power connector. PDB is a power distribution board.

## 3.3 Typical OCP Sections Not Applicable

This is a Base specification, requiring other DC-MHS specifications to fully define a design. The following typical Sections of an OCP specification are not included because they are not applicable to this specification.

Rack Compatibility
Physical Spec
Thermal Design
Rear Side Power, I/O, Expansion
Mechanical
Onboard Power System
Environmental Regulations/Requirements

Prescribed Materials
Software Support
System Firmware
Hardware Management

# 4. M-PESTI Electrical Overview:

**Physical layer overview:**
- +3.3V LVCMOS signaling
- Open-Drain drivers
- Circuit provides voltage back feed detection
- Optional local voltage bleed off circuit

**Physical Circuit:**

## 4.1 Example M-PESTI Circuit Topology (1 x8 source to 1 x8 destination):

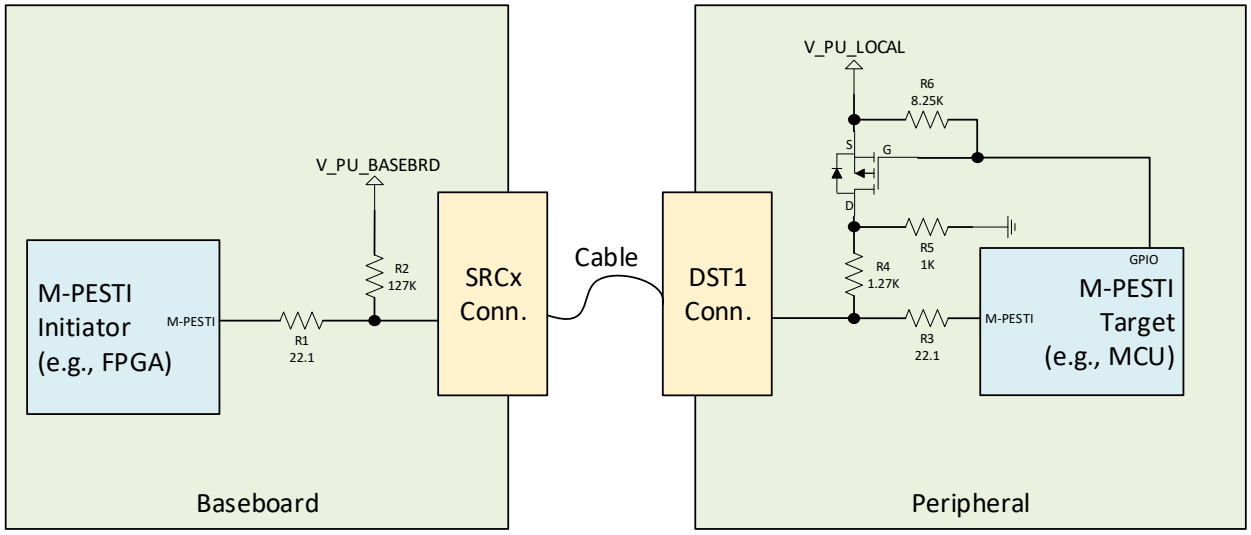



Figure 3. M-PESTI Electrical circuit

## 4.2 Electrical Component Selection Factors:

R1 & R3 are recommended series termination resistor values. They may need to be adjusted to the driver and transmission line characteristics.

R2 must be selected with the following in mind
- Select the maximum value to minimize the current sourced into an unpowered target
- Logic high=1 must be guaranteed to meet the baseboard initiator logic Vin High Minimum when cable/M-PESTI target is not attached/present

R4 & R5 must be selected with the following in mind
- R4 &R5 (pull-down) and R2 (pull-up) must guarantee a logic low=0 at the baseboard initiator logic for voltage back feed detection
- R4 value provides the rise time to a logic high at the MCU/FPGA for an open-drain interface.
- R5 provides a path to GND for the current sourced by R2 on the source board to drain or bleed voltage accumulation at an unpowered target.

P-FET (or equivalent) is enabled (drive gate low) by the M-PESTI target when the cable is detected to be “fully seated” and the device is ready to respond to a discovery request. Selecting a target GPIO that defaults to an input (high-Z) will meet the initial condition requirement. The P-FET enablement results in a rising edge (BREAK release) to be observed at the initiator. BREAK is defined in the ***M-PESTI Protocol*** Section. The method for determining “fully seated” is beyond the scope of this specification as it varies based on connector, latching schemes and cable assemblies.

## 4.3 Example Multi-interconnect M-PESTI topology

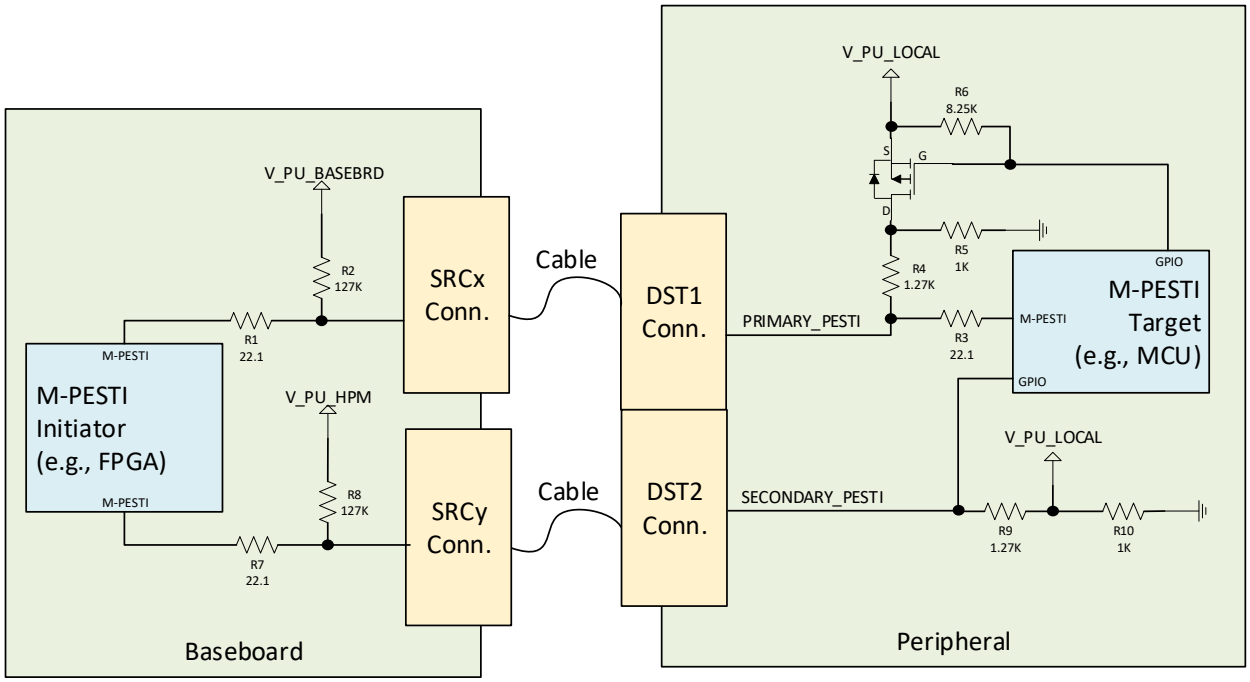



Figure 4. Multi-Interconnect M-PESTI Circuit

The SECONDARY_PESTI wire is not used for frame-based communication but is used to discover source to destination couplings and thus does not require the P-FET or series term elements. The pull-up resistor is required to avoid a floating input at the M-PESTI target when the secondary cable is missing, and the pull-up/pull-down enables voltage back feed detection by system management firmware.

If the cables are swizzled, then the M-PESTI Initiator connected to the secondary high speed data path enables system management firmware to detect a misconfiguration (see ***Source to Destination Detection Phase*** section.) If only the secondary destination(s) is coupled to a source, then the circuit aids system management firmware to determine if something is coupled but unsure what, thereby indicating a misconfiguration.

# 5. M-PESTI Protocol

## 5.1 M-PESTI Protocol Phases

- Discovery Phase
  - The presence and discovery of the attached peripheral must occur prior to the active phase. During discovery, a payload of static attributes is captured from a M-PESTI target (i.e., MCU) to the M-PESTI initiator (i.e., baseboard FPGA).
- Active Phase
  - The active phase is characterized by a repetitive exchange of virtual wires between the initiator (Baseboard FPGA) and the target (MCU, CPLD or other).

## 5.2 Overview of Framing and Error detection

- Simple, multi-byte read/write byte-level commands.
- 250,000 BAUD +/- 3% (Chosen as max supportable BAUD across diverse channels and low-cost MCU and PLD families)
- 8-O-1 (8 data bits, Odd Parity, 1 stop bit)
- Discovery Payload checksum (CRC-8). RX error detection via parity of frames & checksum of payloads. Inbound error detection responses defined.
- System Command / Target Response protocol. No async interrupt.
- Broadcast support for cases where a single initiator UART is shared with a group of targets AND very low latency virtual wires are required.
- Optional Source & Destination connector instance coupling determination method

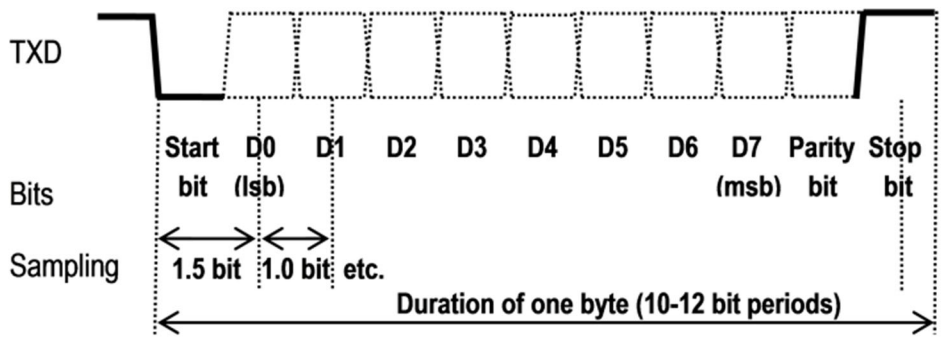



Figure 5. M-PESTI Frame

| Property | M-PESTI |
|---|---|
| Extended Peripheral Data & Power Path(s) | Initially targeted use cases: High Speed I/O connections, internal power distribution management, control panels, cooling subsystems. |
| Peripheral Card (Target) TX Protocol | Frame Based |
| Planar Logic (Initiator) TX Protocol | Frame Based |
| Initial Discovery | M-PESTI target issues a BREAK pulse (Rising edge observed at Initiator) |
| Voltage Back-feed Circuit | Yes |
| Target Pin Function | UART RX/TX + GPI (Initiator Abort) |
| Initiator/Target TX Driver Type | Open-Drain |
| Payload Data Types | Discovery (Static) & Active (Dynamic) protocol phases |
| Control/Status Support | Yes |
| 8-bit Checksum (Discovery Payload Only*) | CRC-8 |
| Checksum Validator | Initiator and Target |

*Active phase CRC-8 is excluded in M-PESTI 1.0. From physical testing, a multi-bit error has never been observed during the active phase. As we speed up the interface (> rev 1.0), a PEC byte will become more important. Additionally, START=0. PARITY=ODD and STOP=1 are 3 of the 11 bits per frame that are predictable. STOP=0 is detected as a framing error.

Table 1 M-PESTI Protocol Overview

## 5.3 UART BREAK Definition

A UART BREAK event is defined as the wire being driven/held low for a time greater than the entire frame length. Example: At 250 KBAUD, the M-PESTI frame length is nominally 4 us * 11-bit positions which is 44 us. FPGA logic may detect a BREAK condition by starting a timer at the falling edge of the signal. A valid BREAK assertion does not require a falling edge to be detected. Some UART receivers do not include a BREAK_DETECT detect status register. Depending upon the UART implementation in the MCU or other device, the BREAK event may appear as an abnormal frame with a data value = 00h that has both a Parity Error and a Framing Error (Stop bit = 0.)

Target Implementation Options

| Target Implementation Options | Simple Presence | Discovery Only | Discovery+ | Discovery & Active Phase |
|---|---|---|---|---|
| Discovery Phase | No | Yes | Yes | Yes |
| Source/Destination Detection | No | No | Yes | Yes |
| Virtual Wire Exchange | No | No | No | Yes |
| FW Target Device Required | No | Yes | Yes | Yes |

## 5.4 Discovery Phase Payload Options:

The discovery payload is typically consumed by baseboard logic, system firmware and BIOS. The payload contents can be streamlined to only contain the minimum required fields to describe the format and size of the payload itself along with peripheral characteristics (static attributes) to consumers.

Payload contents include description of supported virtual wires, and OEM defined fixed attributes like payload version, vendor ID, module class, module Unique ID and checksum. Some bits may be able to change but another Discovery Phase is required to capture them (e.g., source/destination coupling determination method utilizing the stipulated stimulus/response scheme). Note: If M-PESTI is extended into other industry form factors, then the discovery and active phase bit definitions need to be applied to the respective device classes.

| M-PESTI Discovery Payload Content Options | None | Minimum | Maximum |
|---|---|---|---|
| Presence Detection Mechanism | Simple Presence | BREAK Release | BREAK Release |
| Discovery Payload Content | None | Header Only | Complete |
| Virtual-Wire Support | No | Yes | Yes |
| Self-describing Physical Routing | No | No | Yes |
| Source/Destination Detection | No | No | Yes |

A) **Simple Presence:**
- M-PESTI wire is held static low to indicate simple presence
- No device identity or precise source/destination instance coupling identification needed.
- System firmware reads FRU SEEPROM, or hard codes config.
- Use when limited programmable resources exist or attributes not needed (purpose-built system).

B) **Minimum:**
- System firmware uses vendor class, module class and/or unique ID to look up in a BMC store's extended attributes library, Source/destination couplings. e.g., Source Conn#2 coupled to Riser#1 conn#1.
- Locally readable attributes (e.g., MCU firmware version) or locally read inputs (e.g., non-hot plug slot presence).

C) **Maximum:**

- Plug-N-Play where peripheral is fully self-describing and baseboard BMC+BIOS needs no prior knowledge of the peripheral.
- May be preferred when BIOS-BMC interactions are limited.
- Example attributes: Physical PCIe, I2C and Power routing/MUXing/switching topologies, physical or thermal characteristics of the module or elements therein. Although transferring a full FRU SEEPROM image is possible, the header should be limited to critical items needed before 2-wire reads are possible.

### 5.4.1 Optional Active Phase:

- To optimize latency, dynamic virtual wires should be latency appropriate real-time signals and not static attributes that could be carried in the discovery phase payload or a FRU SEEPROM. The dynamic phase includes simple send and receive of **all** virtual wires at once (by M-PESTI initiator and target).

System side logic may source or sink status & controls to multiple domains (such as BMC, BIOS or HW controls).

## 5.5 Example: System FPGA Control and Status Registers

Prior to describing discovery and active phase state transitions, it is useful to define some fundamental M-PESTI target status fields that are implemented within the initiator for system management FW consumption. Shortened names are included that are used to reference these fields in subsequent sections of this specification. See *Target Reset and Fault Handling* for additional information

DISCOVERY_STATUS[1:0] (RO) [AKA: DSTAT]
00b : No BREAK detected, and M-PESTI wire is static HIGH=1
01b : M-PESTI wire is static LOW=0 (simple presence)
10b : Discovery payload has been received with a good checksum
11b : BREAK release detected, but payload has not been successfully received

DISCOVERY_PAYLOAD_ENABLE (R/W) [AKA: DPEN]
0b : Initiator will not send payload request command
1b : Initiator will send payload request command, with retries, until DISCOVERY_STATUS[1:0] = 10b

ACTIVE_PHASE_ENABLE (R/W) [AKA: APEN]
0b : Initiator will not enter command/response phase for that M-PESTI instance
1b : Initiator will enter command/response phase if DISCOVERY_STATUS[1:0] = 10b

ACTIVE_PHASE_ERROR (R/W1C) [AKA: APERR]
0b : Response RX error (i.e., parity, framing) or timeout has NOT occurred.
1b : Response RX timeout or byte receive error has occurred since last cleared. Sticky bit. System management FW must write a ‘1’ to this bit to clear the error once it has been acknowledged.

## 5.6 M-PESTI Discovery

Discovery of a M-PESTI target that is capable of frame-based communication begins with presence detection. A rising edge (UART BREAK release) observed at the initiator indicates that a M-PESTI target is present and available for frame-based discovery. G3 (AC Off) and S5 (System Standby or Soft Off) in the diagrams below refer to system power states as defined in the ACPI specification.

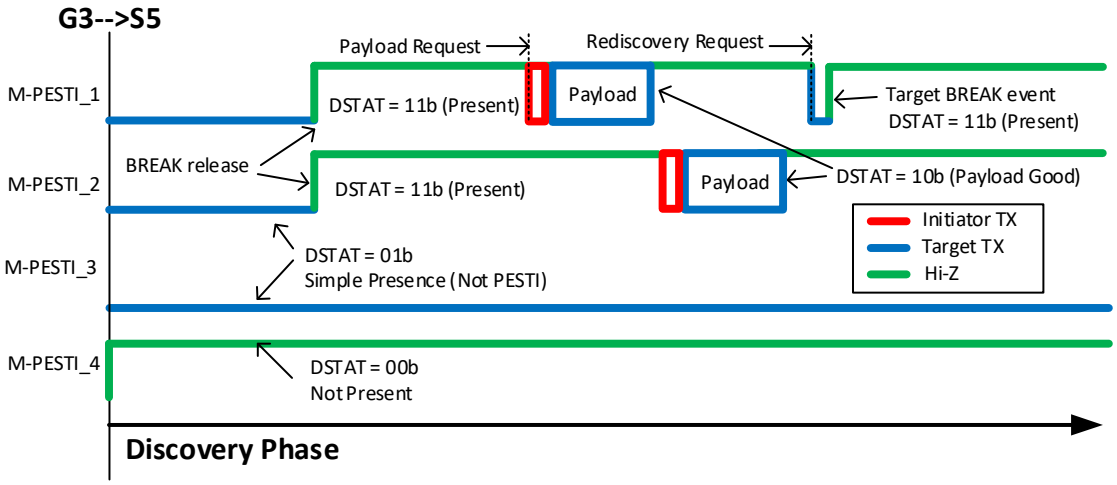



Figure 6. Discovery Phase

### 5.6.1 Target Presence Detection Rules

- Initiator shall not attempt communication while M-PESTI is held low by the target
- Minimum target discovery BREAK low assertion width required to guarantee detection = 50 us
    - If minimum assertion width is met, BREAK shall always be detected by the initiator.
    - This infers the use of parallel circuits for each M-PESTI wire regardless of whether a single initiator UART is shared among multiple targets.
- The target must not release BREAK until:
    - Aux power is good
    - M-PESTI Peripheral is fully seated/mated
    - M-PESTI target is ready to respond to the payload request command.
- Any time a BREAK condition is detected, communication with that device will be halted until the condition is released.
- A device may request a re-start of the discovery process by asserting and releasing BREAK at any time.

NOTE: The discovery mechanism infers that V_PU_BASEBOARD on the planar and V_PU_LOCAL at the module are enabled simultaneously and that the initiator presence detection circuit does not come out of reset until V_PU_LOCAL at the module is “Good.”

This prevents the power up ramp on the module to appear as a break condition at the initiator logic. System dependent delay within the M-PESTI initiator logic guarantees this timing is met. See Figure 4 for the circuit diagram.

### 5.6.2 M-PESTI Discovery Status (DSTAT) Supported Transitions

| Case | DSTAT Current State | DSTAT Future State | State Input |
|---|---|---|---|
| 1 | XXb = Any State | 00b = Absent | Power on Reset (POR) |
| 2 | Absent | 01b = Present/Simple | BREAK asserted following POR de-assertion |
| 3 | Present/Simple | 11b = Present / M-PESTI | BREAK release by the target |
| 4 | Present/Simple | 11b = Present / M-PESTI | Simple presence or M-PESTI device removal prior to target BREAK release while system power is on. |
| 5 | Present/M-PESTI | 10b = Payload Good | Payload received with good checksum |
| 6 | Payload Good | 11b = Present / M-PESTI | BREAK assertion and release by target if not in active phase. Payload Good is protected/locked during the active phase against target resets. This transition also occurs if DPEN is toggled from 0 to 1 while the device is still in the discovery phase (APEN=0.) |
| 7 | Absent | 01b = Present/Simple | Target asserts all 1WIREs low that are not used for communication at the transition to the active phase |

Note: It is not recommended for a user to disconnect a cable while system input power is enabled/connected (Case #4.) If the cable becomes disconnected prior to a payload being received, system management FW can recognize that discovery did not complete successfully. If a cable becomes disconnected during the active phase, ASTAT would indicate that the target is unresponsive.

M-PESTI wires that are not used for frame-based communication fall into two categories.

- Case 2 (table above) when simple cable presence is sufficient and source to destination discovery is not required.
    - BREAK asserted and not released since power on reset de-assertion
- Case 7 (table above) at the completion of source to destination coupling discovery for that wire.
    - Transition to the active phase will cause the target to assert all wires not used for communication to low=0 (simple presence.)

### 5.6.3 Voltage Back feed Detection

Note that when the power path is different from the data path (e.g., separate cables), it is possible that the data cable is present, but the power is missing. This is a detectable condition. A general algorithm for a possible voltage back feed condition is the following

- A M-PESTI signal is LOW=0 (DSTAT = 01b) at the baseboard initiator logic

- System management firmware did not account for that M-PESTI instance during the source to destination stimulus & response discovery phase
- The wire with DSTAT=01b is not expected or allowed to use “simple presence.”

### 5.6.4 M-PESTI Protocol Phase Diagram (Express)

Figure 7 below depicts an autonomous transition from the discovery phase to the active phase without any source/destination coupling detection. With APEN=1 and the discovery payload received successfully; the default exchange of virtual wires occurs without any firmware intervention.

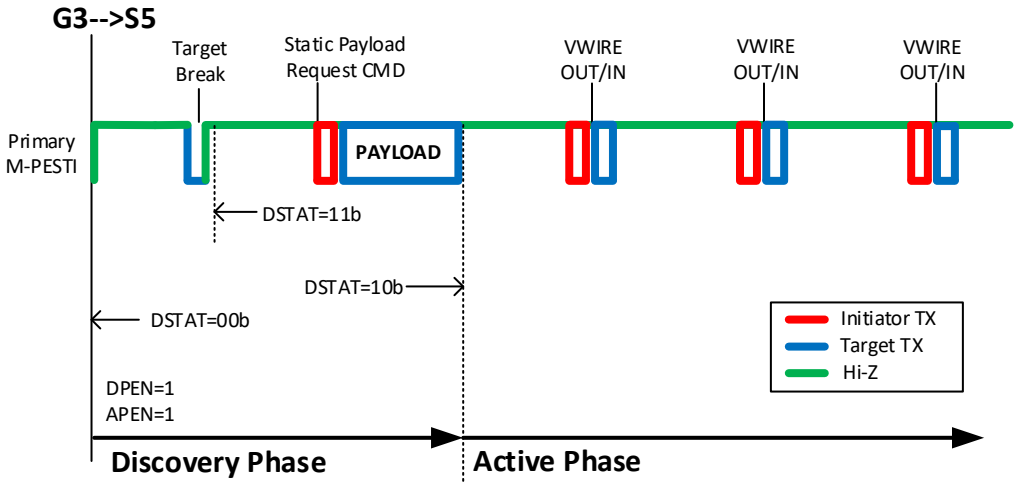



Figure 7. Discovery and Active Phase

#### *5.6.4.1 Example (Express) Discovery Process Flow (Figure 7 above)*

1. Initial Conditions:
    - DISCOVERY_STATUS[1:0] = 00b : Not present (reset value)
    - DISC_PAYLOAD_ENABLE = 1
    - ACTIVE_PHASE_ENABLE = 1
2. Following the transition from G3 to S5 power state, once M-PESTI target devices have initialized, all M-PESTI communication signals will be asserted low and released (low to high) to indicate presence of a M-PESTI target to the initiator.
    - DISCOVERY_STATUS will transition from 00b (not present) to 01b (simple presence during the BREAK event, then to 11b : M-PESTI target present with available payload upon request once the BREAK event is released.
    - Initiator sends payload request command to the target
    - Target responds with the static payload
    - DISCOVERY_STATUS = 10b : Payload received with a good checksum and is available for consumption by system management firmware
3. Initiator enters hardware controlled virtual WIRE exchange with the target
    - Hardware and firmware may control and query status of virtual wires via vendor defined methods

### *5.6.4.2 Discovery Command and Response Format*


Figure 8. Discovery Command and Response Format

tDPTAR: Discovery Phase Turnaround time between initiator completing transmission of the command and the payload response beginning to be received.

#### *5.6.4.3 Discovery Payload Rules*

- Turnaround time minimum is 100 ns.
- Target must complete the discovery payload response within the payload RX timeout of 1 sec.
- Target shall not transition to the active phase until a successful discovery payload is received.
    - “Successful” = Within the RX timeout period with no byte parity or framing errors and a verified checksum.
- Initiator continuously attempts to retrieve a discovery payload from a M-PESTI target that is present unless DISC_PAYLOAD_ENABLE = 0.
    - Example: Round robin servicing by a single initiator to multiple targets:
      If the discovery payload is not successfully received after an Initial attempt plus two retries per target, the next target in the round-robin rotation will be serviced. When servicing returns to the target, the discovery request command is sent again as a set (initial + two retries) until successful.
- Initially, the target must release (tri-state) all “additional” source wire GPIOs indicating simple presence following power on or reset
- Target must assert all M-PESTI wire sources that are not used for frame-based communication after the FIRST response to a virtual wire exchange in the active phase
- Target must release all M-PESTI wire sources that are not used for frame-based control after a discovery payload request if that payload request has occurred during the active phase.
    - Infers module is in the discovery phase until the next virtual wire command occurs.

#### *5.6.4.4 Payload Format (Riser/Interposer)*

| Number of Bytes | Description |
|---|---|
| 12 | Header Information |
| 5 | Destination 1: Data Fabric and SMBus Physical Routing Description |
| 5 | … |
| 5 | Destination N: Data Fabric and SMBus Physical Routing Description |
| 2 | Destination Wire Descriptors |
| Varies | Misc. Vendor Specific Region |
| Varies | *Padding (0 – 7 Bytes) |
| 1 | Checksum |

Minimum Required Payload Regions

*Payload size must be padded to be a multiple of 8 bytes including the checksum byte. Other/future device classes may have different descriptor groups between the header and the vendor specific region.

### *5.6.4.5 HEADER (with example data for a 2 CEM slot Riser/Interposer):*

| Byte/Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 00h | ***PAYLOAD_VERSION[7:0] = 00h*** | | | | | | | |
| 01h | ***DEVICE_CLASS[7:0] = 00h (CEM Riser)*** | | | | | | | |
| 02h | ***STATIC_PAYLOAD_SIZE[7:0]= 04h (32 Bytes)*** | | | | | | | |
| 03h | NUM_VIRTUAL_WIRE_OUTPUT BYTES[3:0] = 0001b (1 Out Byte) | | | | NUM_VIRTUAL_WIRE_INPUT BYTES[3:0] = 0001b (1 In Byte) | | | |
| 04h | DEVICE_ID[15:8] = 00h | | | | | | | |
| 05h | DEVICE_ID[7:0] = 27h | | | | | | | |
| 06h | VENDOR_ID[15:8] = 80h | | | | | | | |
| 07h | VENDOR_ID[7:0] = 86h | | | | | | | |
| 08h | DEVICE_VERSION[7:0] = A0h | | | | | | | |
| 09h | AFU = FFh | | | | | | | |
| 0Ah | NUM_DST_WIRES[3:0]= 0100b (4 CBL_PRES signals) | | | | AFU = 0b | NUM_PICPWR_DST_WIRES[2:0] = 001b (1 PWR) | | |
| 0Bh | AFU = 0b | AFU = 0b | AFU = 0b | NUM_EP_DESCRIPTORS[3:0] = 00010b (Two Slots) | | | | |
| 0Ch | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b |

Minimum Required Fields

See ***Appendix A*** which includes definitions and a two-slot riser payload example.

### 5.6.5 Source to Destination Detection Phase (Optional)

System management firmware can provide a stimulus to a M-PESTI target by asserting the corresponding M-PESTI instance LOW=0. Once the stimulus is asserted, an updated static payload can be queried for a corresponding response to that stimulus by that target. This stimulus and response method can be used to govern source to destination couplings and provide end-to-end data fabric physical mapping from a root port to the destination through any source connection.

#### *5.6.5.1 M-PESTI Protocol Phase Diagram (With Optional Source Detection)*

The example below depicts a target destination with two source wires through two x8 or a single x16 destination connector. The first row of the wave diagram below shows the primary M-PESTI used for frame-based communication. The primary source wire is associated with a specific set of data fabric lanes that is described to be physically routed to a destination in the initial discovery payload. This results in the additional source wires requiring a stimulus response mechanism to discover the physical routing of those data fabric lanes to the destination. It is preferred to move this function up the stack to system firmware to simplify the target device hardware and firmware (e.g., avoids multiple UARTs).
The second wave diagram below depicts that a stimulus to a wire not associated with that target did not result in a corresponding response. Because it was a stimulus/response "MISS", a third and successful stimulus and response was required to discover the physical cable source.

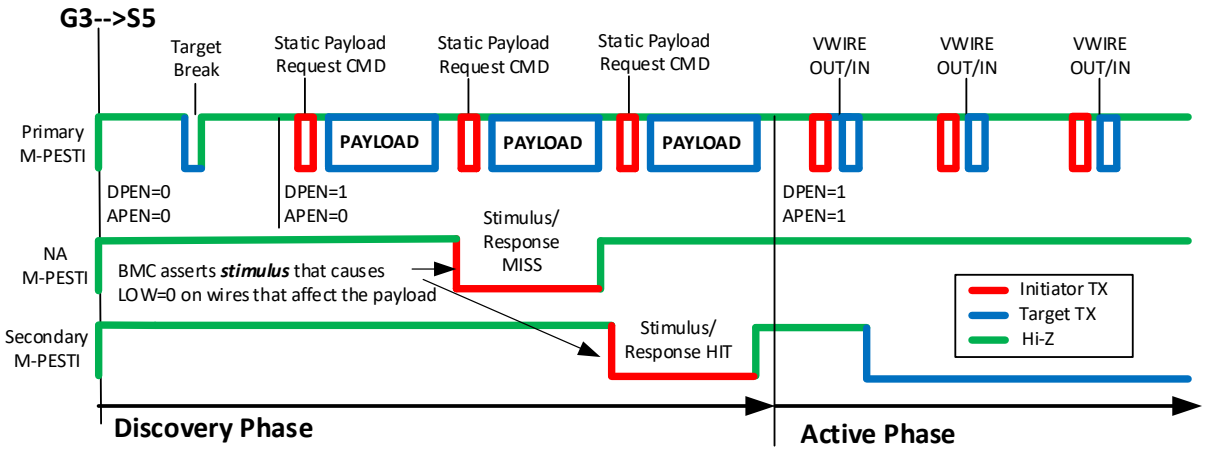



Figure 9. Discovery Command and Response Format with Optional Source Detection

#### *5.6.5.2 Example of Source Discovery Process Flow*

Initial Conditions:

- G3 to S5 power state transition
- DSTAT[1:0] = 00b : Not present
- DPEN = 1
- APEN = 0

Shortly after the transition from G3 to S5 power state, all M-PESTI targets will de-assert BREAK to indicate presence to the system.

- DSTAT = 11b : M-PESTI target present with a payload that is available upon request.

With DPEN=1, the platform FPGA will retrieve initial discovery payloads from each M-PESTI target while the BMC may be booting.

Once booted, the BMC can perform source to destination discovery for any target that includes multiple source couplings in the discovery payload (NUM_DST_WIRES, NUM_PICPWR_DST_WIRES)

1. Set APEN=1. This will cause the initiator to send the v-wire exchange command. In response, the target will drive all secondary wires low (simple presence) prior to responding with platform v-wire inputs. The BMC can observe that simple presence/break is active on all secondary wires attached to that target. DSTAT = 01b (Present) that were previously DSTAT=00b (Empty)
2. Set APEN=0 to prevent target transition to the active phase during destination discovery iterations.
3. BMC writes DPEN=0, then DPEN=1
    - The rising edge of DPEN causes the initiator to send payload request command. As a result, the target tri-states all secondary wires because it is in the discovery phase and responds to the payload request after it samples the input state of those wires.
4. The BMC enables a low=0 stimulus to an individual secondary M-PESTI wire and clears any previous stimulus that was asserted.
5. BMC repeats steps 3,4 until all source to destination couplings are identified
6. BMC sets APEN = 1
7. Initiator and target enter HW controlled V-WIRE exchange
    - Firmware sends commands and reads the responses via the OEM command register interface.
8. Target asserts simple presence on all secondary M-PESTI wires which can be ignored during source to destination discovery for additional targets.

## 5.7 Target Reset and Fault Handling

If the target resets during the active phase, it would be observed as temporary unresponsiveness at the initiator. This would be reflected in the ACTIVE_PHASE_ERROR (APERR) register described above if any RX timeout or parity error occurred. Following target reset, a BREAK assertion and release would be observed by the initiator. During the active phase, the discovery status must be protected (locked) at a value of 10b=Payload Good. Locking the discovery status and payload data is required so that the host can safely consume the contents at any warm or cold reset. Since the discovery has already occurred, the initiator would resume sending the virtual wire exchange command and the target would not observe a discovery request command. Once a device has transitioned to the active phase, the only method to unlock the discovery payload and status is by system management firmware clearing, then setting DPEN.

If the target resets during the discovery phase prior to the entry to the active phase (e.g., APEN=0), the DISCOVERY_STATUS value would revert to 11b (Payload not received.) The initiator would autonomously send the payload request command if DISC_PAYLOAD_ENABLE = 1. The target device would not be transitioned to the active phase until DISCOVERY_STATUS=10b. If discovery had not previously completed or DPEN=0, the device would not be discovered and transition to the active phase.

## 5.8 Active Phase: Dynamic Virtual Wires

Once the target transitions to the active phase, the initiator autonomously exchanges virtual wires with each target device. The HW controlled virtual wire inputs and outputs are target device class specific. The total (HW controlled + firmware controlled) number of bytes in and out are advertised within the static discovery payload.

### 5.8.1 Virtual Wire Exchange Example (1 Byte Out/In)

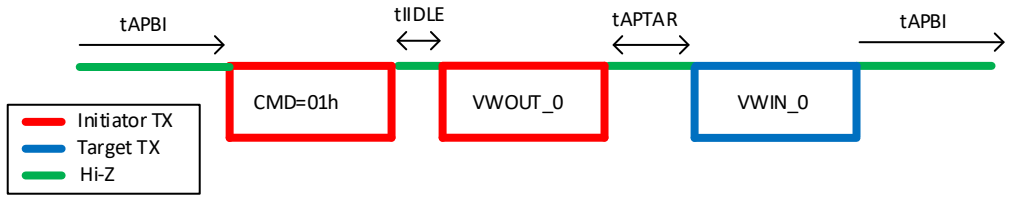



Figure 10. Single Byte Virtual Wire Exchange

### 5.8.2 Virtual Wire Exchange Example (N Byte Out/M Byte N)

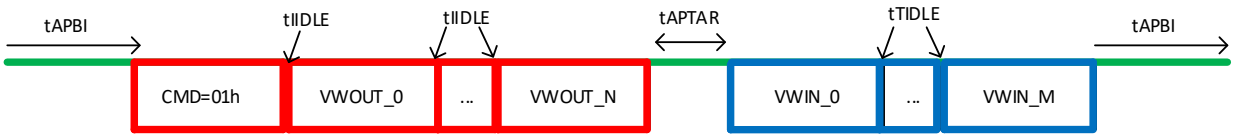



Figure 11. Multi-byte Virtual Wire Exchange

### 5.8.3 Active Phase Rules

- Minimum Initiator M-PESTI idle period between RX of the previous target response & TX of the next target command is tAPBI
- A target must wait tAPTAR minimum before beginning to transmit the response.
- A target must complete a response to the initiator within the RX timeout of tAPRTO.
- The maximum period between commands sent from an initiator to that same target is not bound by this specification.
- If the target device does not support virtual wires (active phase is not required) in either direction:
    - Target shall ignore the virtual wire out value and respond with a virtual wire input value=00h
    - Target shall drive any/all secondary M-PESTI wires low after receiving CMD=01h.
    - The BMC may clear APEN to disable the active phase for a device once the secondary M-PESTI wires are driven low by the target.
        - Even though the primary M-PESTI wire is static high=idle, the discovery payload and status would be locked until the next power cycle of the target or until DPEN is cleared, then set by system management FW.
- Number of out bytes and in bytes can be asymmetrical as suited for the application.

The method to route virtual wires to/from internal system logic or other entities and the M-PESTI wire is outside the scope of this document. The usage of the virtual wires (internal commands/policies or connections to local physical signals) by the target is also outside the scope of this document.

### 5.8.4 Virtual Wire (HW) Definitions by Device Class

#### *5.8.4.1 EXAMPLE DEVICE_CLASS = 00h (CEM Interposer/Riser)*

**VWOUT_0 (System Output Virtual Wires)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| AFU | AFU | AFU | AFU | AFU | AFU | S0_RUN | PWR_BRK |

***PWR_BRK*** : Active high (1=Assert, 0=De-assert) virtual wire

***S0_RUN*** : Active high (1=True, 0=False) virtual wire indicating system power state is ACPI S0_RUN.

***AFU*** : Available for Future Use

**VWIN_0 (System Input Virtual Wires)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| AFU | AFU | AFU | AFU | AFU | AFU | AFU | WAKE |

***WAKE*** : Active high (1=Assert, 0=De-assert) virtual wire that indicates the target device is requesting entry to the ACPI S0_RUN state.

***AFU*** : Available for Future Use

All other device classes are vendor specific until a point where such standardization occurs

## 5.9 Broadcast Commands

To reduce the amount of baseboard logic required, it may be desirable to utilize a single initiator UART that is shared (multiplexed) among multiple targets. This has a limitation that commands and responses cannot be in progress to multiple targets simultaneously. If initiator sharing is implemented at the baseboard logic, each target that shares a common initiator must be serviced in a round-robin rotation.

Following is a simplified FPGA logic diagram of a M-PESTI target group that shares a single M-PESTI initiator. It depicts the parallel break detectors required for each M-PESTI wire and logic that acts as an internal MUX. The internal MUX can be directed to select/focus on a single M-PESTI wire at a time or enable the UART TX traffic to be forwarded to every target within the group

### 5.9.1 FPGA Logic Diagram

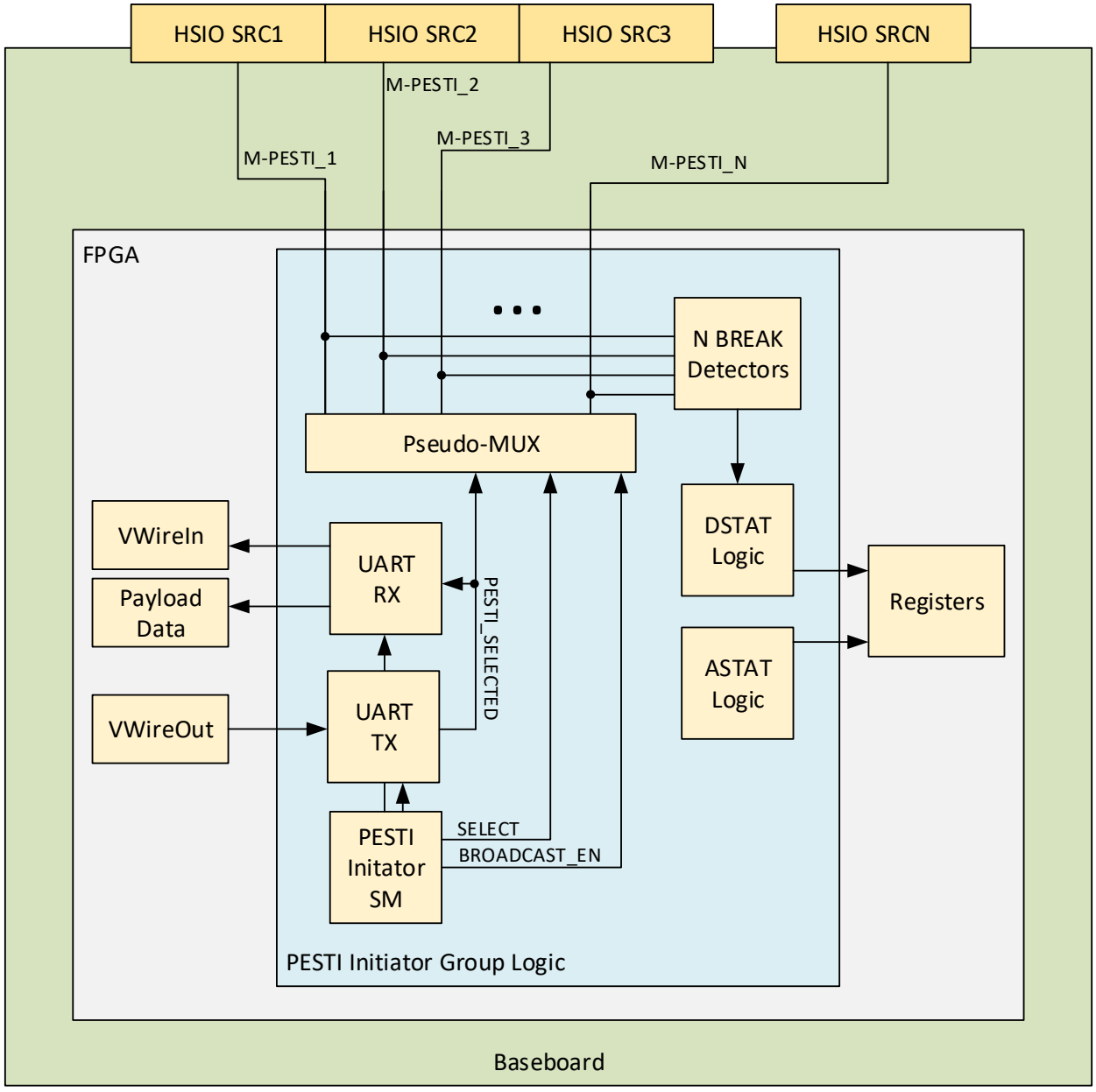



Figure 12. Example PESTI Initiator Logic

PWR_BRK is an example low latency output virtual wire required to be communicated to all applicable M-PESTI targets within ~400 us (system specific). To meet this requirement while allowing implementation flexibility to share an initiator among multiple targets, a special broadcast command is required.

This following broadcast command is required to be supported by applicable targets.  It is required to be implemented in the baseboard logic if an initiator is shared with multiple targets.

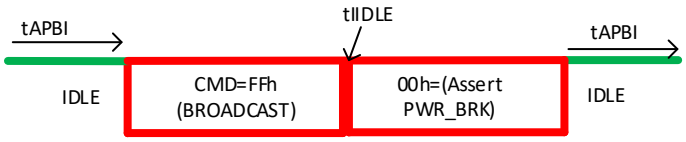



Figure 13. Example Broadcast Command

There is no response to this command, so that it can optionally be repeated to ensure that it was received by the target

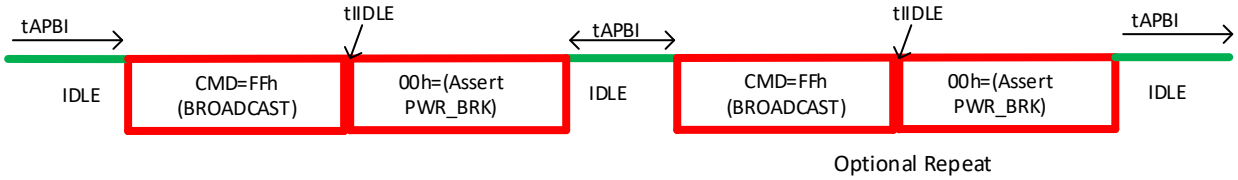



Figure 13. Example Broadcast Command Retransmission

A benefit is that PWR_BRK would be set in the next virtual wire exchange to each applicable target if the triggering system event remains active. Round robin servicing naturally staggers PWR_BRK de-assertion (which is a common system preference to avoid excessive inrush current from high power loads). Additional delay between PWR_BRK de-assertion to multiple targets is implementation specific and controlled via system firmware via baseboard logic that feeds into the appropriate M-PESTI channel.

### 5.9.2 Example of Round Robin VWIRE exchange and Broadcast

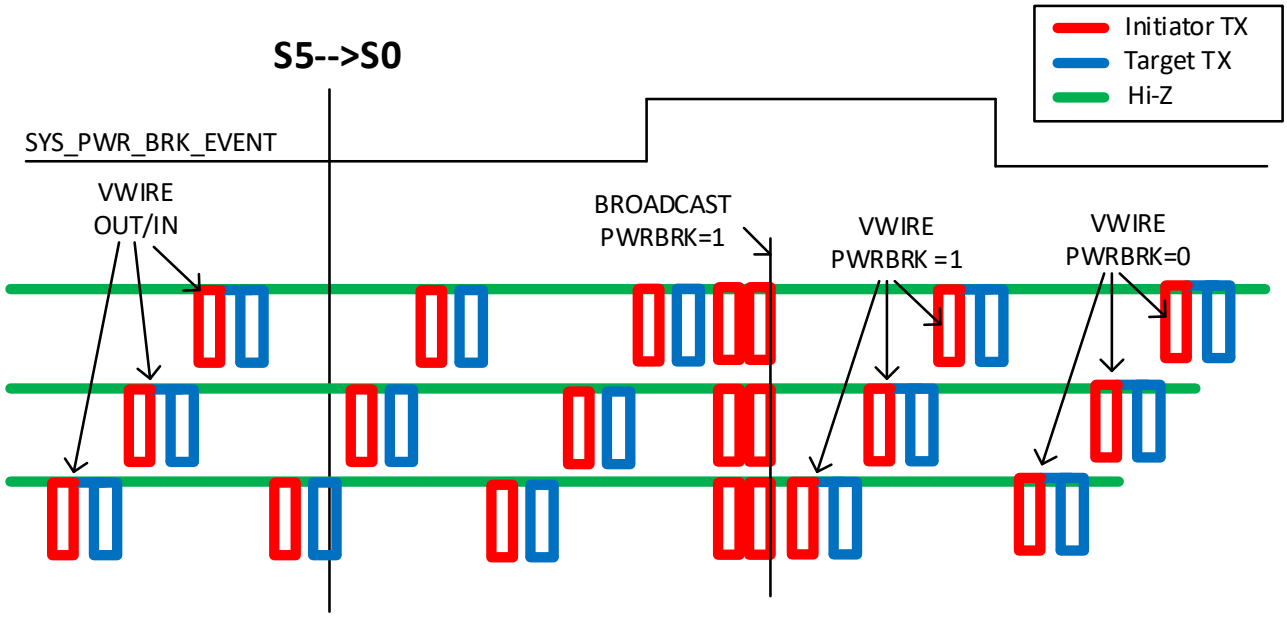



Figure 14. Example Round Robin and Broadcast Transitions

SYS_PWR_BRK_EVENT (Active high) in the diagram above represents the logical combination of any hardware or firmware triggers that request assertion of PWRBRK# (Active-low, Emergency Power Reduction State Request) as defined in the PCI Express Base Specification.

## 5.10 Initiator Abort Mechanism

In some examples low latency commands needs to be sent and thus an abort mechanism is required. In these examples, PWRBRK is the example low latency command for illustration purposes.
To limit a PWRBRK command insertion latency, the initiator may abort any communication exchange by asserting the M-PESTI wire low. There are two general cases for the initiator abort sequence.

1. Case 1 : While the initiator is transmitting or about to transmit.
2. Case 2 : While the target is transmitting or about to transmit.

*Case 1* : The abort sequence while the initiator is transmitting is synchronous to the START of a frame. The initiator abort assertion (tABREAK) will begin at the same instant that the falling edge of START would begin. The assertion width is guaranteed to encompass the START bit, 8 data bits, parity bit and STOP bit. There is additional margin of the assertion width beyond the STOP bit to allow the target to capture the BREAK as an invalid frame with both a parity (even) and framing (STOP=0) error. The worst-case insertion latency occurs when the system PWRBRK event occurs just after the beginning of a normal START condition. In this case, the tABREAK assertion does not begin until after the STOP bit of the previous frame.

**CASE 1 : PWRBRK Insertion Latency During Initiator TX Phase**

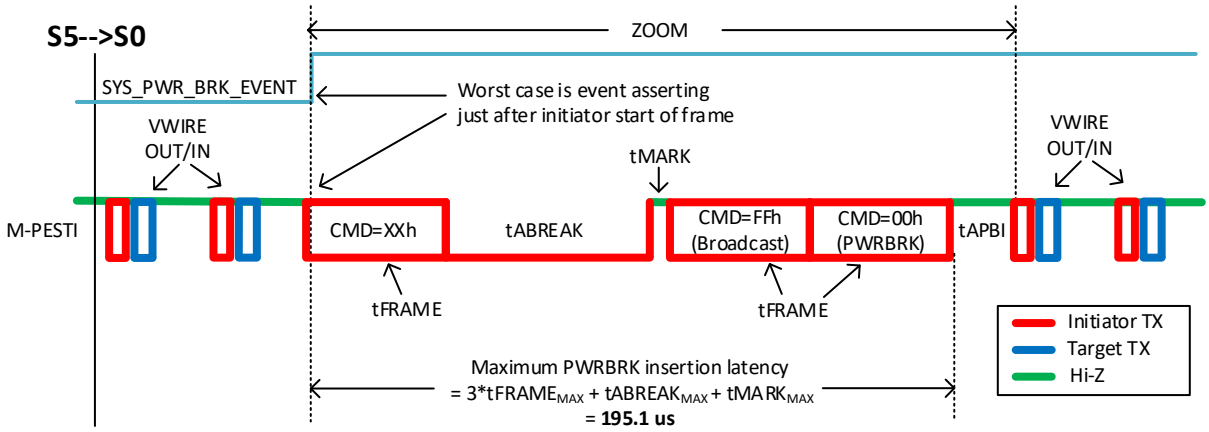



Figure 15. PWRBRK Insertion Latency During Initiator TX Phase

*Case 2* : The abort sequence while the target is transmitting is ***asynchronous*** to the START of a frame. The initiator abort assertion (tABREAK) will occur at any point within the target transmitted frame. The assertion width is guaranteed to encompass the START bit, 8 data bits, parity bit, STOP bit, tTAR and tTIDLE. Because there is sufficient margin, the TARGET is only required to sample for the abort just prior to the START of any frame. The worst-case insertion latency occurs when the system PWRBRK event occurs just after the beginning of a normal START condition. In this case, the abort is not recognized by the target until just before the following START of a response frame.

**CASE 2 : PWRBRK Insertion Latency During Target TX Phase**

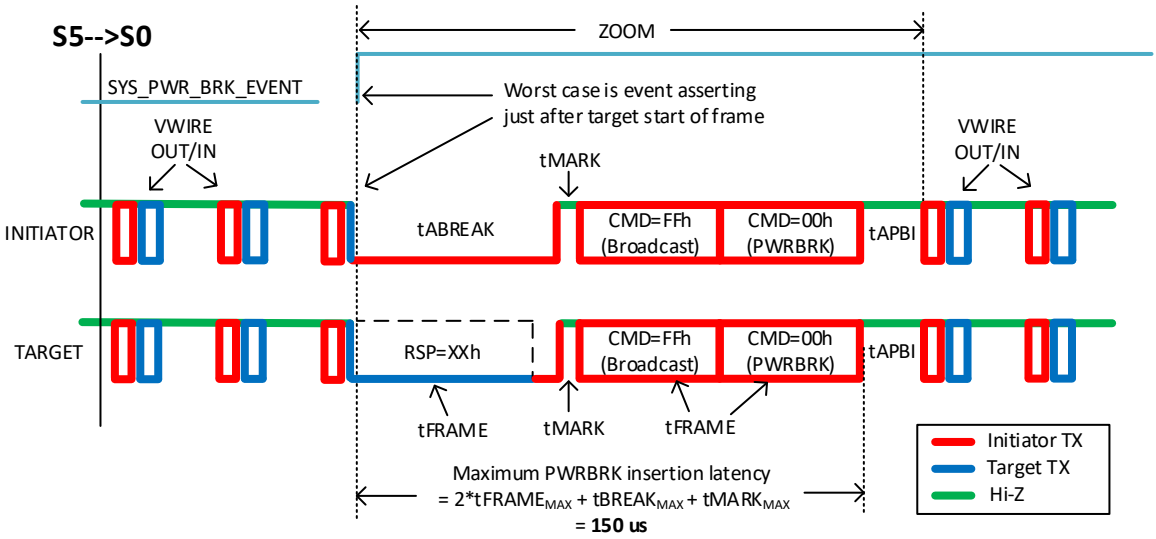



Figure 16. PWRBRK Insertion Latency During Target TX Phase

# 6. M-PESTI Fan Out

Applications exist where the ability to fan out a M-PESTI bus to multiple targets exist. One such example is a motherboard or Host Processor Module (HPM) connection to a Power Distribution Board (PDB) with N number of target subsystems such as backplanes or risers. Since the number N is not a priori known by the motherboard, and pre-plumbing for a maximum quantity requires additional interconnects, M-PESTI fan out support helps scale the ability to support M-PESTI type features.

This specification version includes the scope of fanout from between 2 to 8 downstream busses. Although out of scope, nesting of multiple tiers of fan out within a single hierarchy is possible with additional CMD codes and circuitry. In all fanout cases, it is left to the designer to understand the latency effects of fanout width (and depth). Two methods are shown where the MUX method is for typical fanout needs and the Switch method is targeted for applications that require the ability to broadcast commands simultaneously with all targets.

Two fanout methods of operation include the MUX and Switch methods (see circuits below).

- **MUX method**: Typical 1-to-many fanout. Broadcast commands are ineffective.
- **Switch method**: For applications requiring broadcast commands to all attached targets.

Two modes of operation for a fanout controller include Target Mode & Snoop Mode

- **Target Mode (Default)**:
    - Controller acts as a target supporting discovery and active phases
    - Only the fanout controller may be attached to the initiator bus
        - MUX Method: No channels are selected
        - Switch Method: Fanout controller on Ch0 is always enabled; all others default disabled
    - Shall support >=1 status command for the initiator reading information from the fanout controlling PESTI target such as:
        - The current MUX/switch settings
        - If an issue was observed such as a closed switch bus hang watchdog timeout
        - Live status of downstream subsegments (when the MUX is not focused or switch closed on a particular subsegment)
- **Snoop Mode**: Fanout controllers must:
    - Enter snoop mode any time any other target(s) are attached to a bus
    - May process broadcast commands if application relevant
    - Listen/Process only special fanout control commands (MUX Select or Switch channel enable), thereby ignoring discovery and target mode active phase commands intended for other targets
      Example MUX Select commands may include:
        - Go to Target Mode and de-select all subsegments
        - Select specific subsegment(s) as connected, which may comprise of 1, all or select groups.

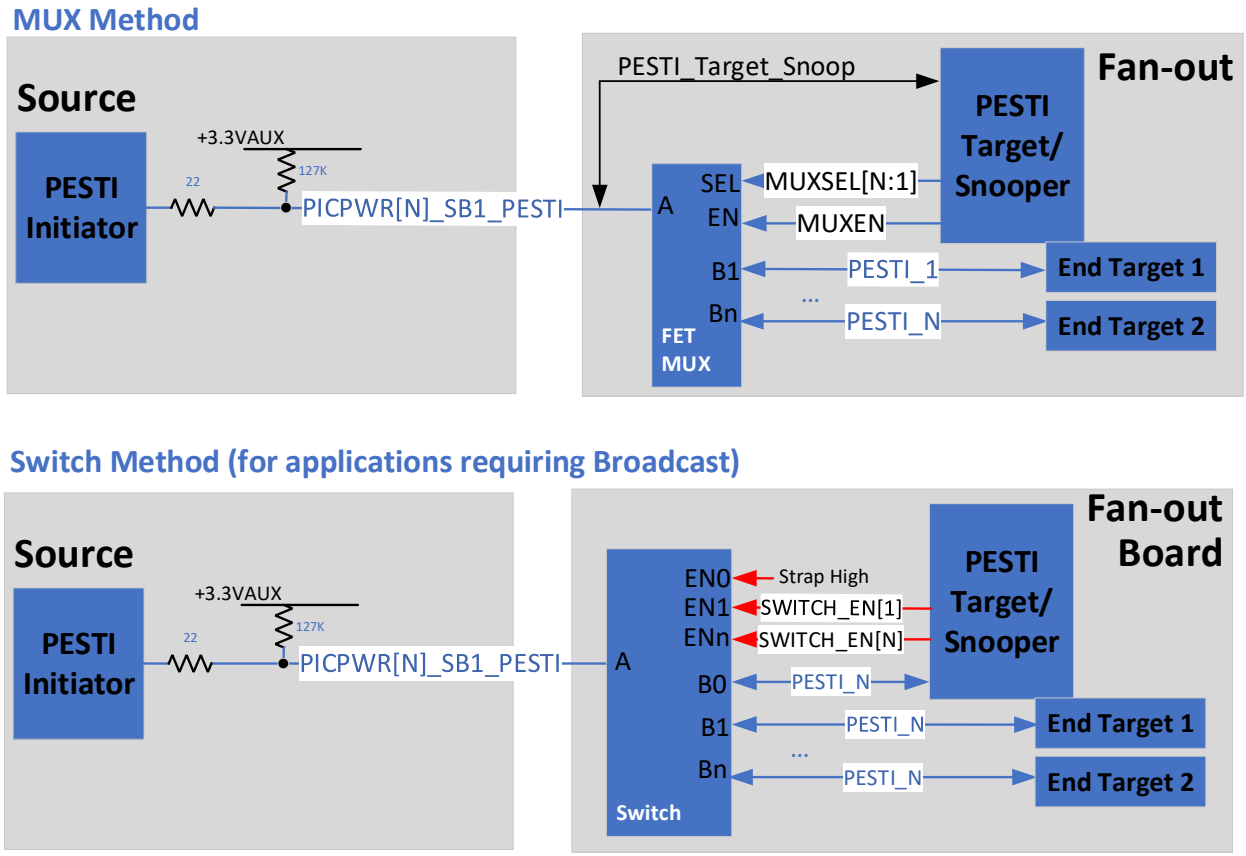



Figure 17. Example M-PESTI FANOUT Methods

Designer note: Buffer(s) may be necessary if the Target/Snooper and the FET MUX or Switch have significant layout stubs. The designer may choose a switch with active buffers versus a FET switch to avoid reflection effects.

Since any downstream interface may have a strap (such as a strong presence pulldown resistor) or be in a stuck state driving high or low, it is imperative to not create a deadlock condition where the Fanout controller cannot observe new commands to change the MUX or switch settings. Therefore, it is recommended to have the fanout controller implement a watchdog timer of at least 20ms after closing 1 or more channels to see any signaling activity. If no such activity (rising or falling edges), open the just selected channel(s) and set a status to inform the PESTI initiator why the channel(s) automatically opened.

An alternative option is for the fanout controller to always be able to observe and report upstream the state of any downstream subchannel so that the PESTI initiator can choose to not close a suspect channel.

# 7. Electrical Specifications

## 7.1 DC Specifications

| Symbol | Parameter | Min | Max | Units | Comments |
|---|---|---|---|---|---|
| VDD | Bus Voltage | 3.135 | 3.465 | V | 3.3 +/-5% |
| VIH | HIGH level input voltage | 2.0 | | V | 3.3V LVCMOS |
| VIL | LOW level input voltage | | 0.8 | V | 3.3V LVCMOS |

## 7.2 AC Specifications

| Symbol | Parameter | Min | Max | Units | Comments |
|---|---|---|---|---|---|
| tBAUD | BAUD rate | 242500 | 257500 | kHz | 250000 +/- 3% |
| tFRAME | Start + 8b Data + Parity + Stop | 42.7 | 45.3 | us | 1/tBAUD * 11 bits |
| tF | Fall Time | - | 120 | ns | Same as 1MHz SMBus ($V_{IH,MIN}$ + 0.15 V) to ($V_{IL,MAX}$ - 0.15 V) |
| tR | Rise Time | - | 120 | ns | Same as 1MHz SMBus ($V_{IL,MAX}$ - 0.15 V) to ($V_{IH,MIN}$ + 0.15 V) |
| tSPIKE | Noise Spike suppression time | 0 | 50 | ns | Same as 400kHz SMBus. Noise suppression is recommended, but not required |
| tABREAK | Initiator Abort BREAK assertion time | 50 | 55 | us | Initiator abort BREAK assertion falling edge is synchronous to a normal START of frame when the initiator is transmitting. It is asynchronous to START when the target is transmitting. MAX spec. reduces worst case PWRBRK insertion latency |
| tDBREAK | Target Discovery BREAK assertion time | 50 | - | us | M-PESTI target BREAK can be persistent. |
| tIIDLE | Initiator End of STOP to START | 0 | - | ns | STOP is typically sampled at the midpoint of the bit, so there is approximately 2 us of time to the next START. |
| tTIDLE | Target End of STOP to START | - | 1000 | ns | This is required for the target to sample for initiator abort prior to START of target TX. |
| tMARK | End of BREAK to START time | 3.88 | 4.12 | us | MARK time is 1 BAUD period between end of initiator abort BREAK and START of new command |
| tDPTAR | Discovery Phase Turnaround Time | 100 | - | ns | MAX not specified. It is bound by payload size and tDPRTO. |
| tAPTAR | Active Phase Turnaround Time | 0.1 | 20 | us | Between Target RX and Target TX of a response. MAX reduces time to sample initiator BREAK/Abort signal just before START. Target MCU should not have trouble meeting minimum time required to allow the initiator to prepare for RX following TX. |
| tDPRTO | Discovery payload receive timeout | - | 250 | ms | Allows for 150 ms tDPTAR + 2048 byte payload size. |
| tAPRTO | Active Phase receive timeout | - | 500 | us | Includes margin beyond $tAPTAR_{MAX}$ + 8*$tFRAME_{MAX}$ + 7*$tTIDLE_{MAX}$ = 389 us |
| tAPBI | Active phase bus IDLE time | 10 | - | us | Between initiator RX from target and initiator TX to same target |

# 8. Security Considerations

Although M-PESTI is a basic, low level messaging and virtual wire tunnel, the following threats are identified with possible mitigations.
It is believed that necessary mitigations can be implemented on top of the base specification or if necessary may be included in future revisions of this base specification.
Revision 1.0 does not explicitly teach the mitigations due to needing further analysis on the often application specific impacts to latency, complexity and implementation costs.

Threats

1) Physical implant / signal re-routing: Assurance, where applicable, that the PESTI target is on the same HW (or same target) as other management interfaces. SPDM types of security capabilities are the likely current art in which to address this treat.
2) Physical implant, man-in-the-middle snooping or alteration of payloads or virtual wires in flight. Potentially mitigated with encrypted payloads most likely using SPDM defined methods.

# Supplemental Material

## Supplemental Material A: 2 x16 CEM Slot Riser Payload Example

### *Payload Format*

| Number of Bytes | Description |
|---|---|
| 12 | Header Information |
| 5 | End Point 1: Data Fabric, Power and SMBus Physical Routing Description |
| 5 | … |
| 5 | End Point N: Data Fabric, Power and SMBus Physical Routing Description |
| 2 | Destination Wire Descriptors |
| Varies | Misc. Vendor Specific Region |
| Varies | *Padding (0 – 7 Bytes) |
| 1 | Checksum |

Minimum Required Payload Regions *Payload size must be a multiple of 8 including checksum

### *HEADER Definition*

| Byte/Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 00h | ***PAYLOAD_VERSION[7:0]*** | | | | | | | |
| 01h | ***DEVICE_CLASS[7:0]*** | | | | | | | |
| 02h | ***STATIC_PAYLOAD_SIZE[7:0]*** | | | | | | | |
| 03h | NUM_VIRTUAL_WIRE_OUTPUT BYTES[3:0] | | | | NUM_VIRTUAL_WIRE_INPUT BYTES[3:0] | | | |
| 04h | DEVICE_ID[15:8] | | | | | | | |
| 05h | DEVICE_ID[7:0] | | | | | | | |
| 06h | VENDOR_ID[15:8] | | | | | | | |
| 07h | VENDOR_ID[7:0] | | | | | | | |
| 08h | DEVICE_VERSION[7:0] | | | | | | | |
| 09h | AFU = FFh | | | | | | | |
| 0Ah | NUM_DST_WIRES[3:0] | | | | AFU | NUM_PICPWR_DST_WIRES[2:0] | | |
| 0Bh | AFU | AFU | AFU | NUM_EP_DESCRIPTORS[4:0] | | | | |
| 0Ch | AFU | AFU | AFU | AFU | AFU | AFU | AFU | AFU |

Minimum Required

**PAYLOAD_VERSION[7:0]** :
Indicates the format version of the payload. Future revisions may relocate, remove or add additional bit-fields beyond the first four bytes. Any alteration of the definitions to bytes 0-3 must be avoided for backward compatibility to bit fields that are directly consumed by FPGA logic.

**DEVICE_CLASS[7:0]** : Indicates the device class of the M-PESTI target peripheral
00h : Riser/Interposer
All others reserved.

**STATIC_PAYLOAD_SIZE[7:0]** : Indicates the total static payload size. The value of this bit field represents the SIZE/8.
Example: STATIC_PAYLOAD_SIZE = 04h indicates the size as 4 * 8 = 32 Bytes

**NUM_VIRTUAL_WIRE_OUTPUT_BYTES[3:0]** :
Indicates the number of output bytes (from target to initiator) supported for future system management firmware interaction.
**NUM_VIRTUAL_WIRE_INPUT_BYTES[3:0]** :
Indicates the number of output bytes (from initiator to target) supported for future system management firmware interaction.
***NOTE***: The number of HW controlled Virtual wire bytes in and out of a target device and their meaning is defined to be DEVICE_CLASS specific

**VENDOR_ID** leverages PCIe VENDOR_ID table
The formats of **DEVICE_ID** and **DEVICE_VERSION** are vendor specific.

**NUM_DST_WIRES[3:0]** :
Indicates the total number of M-PESTI destinations (required per x8 data fabric segment) that are connected to this device.
Examples:

- A single x8 target requires only one (primary) M-PESTI wire
- A four-slot riser with x16 routed to each slot requires 8 source wires

**NUM_PICPWR_DST_WIRES[2:0]** : Indicates the total number of PICPWR M-PESTI sources supported

**NUM_EP_DESCRIPTORS[4:0]** : Indicates the number of End Point descriptor groups. 5-bits accommodate more than 16 end points to be described with 1-indexing.
Examples:

- A two-slot riser would populate NUM_EP_DESCRIPTORS = 00010b (two)
- A device that does not require any physical routing description would populate NUM_EP_DESCRIPTORS = 00000b (None)

*Endpoint Descriptor Bit fields*

| BYTE/BIT | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | SMB_UP_CH[2:0] | | | SMB_MUX_DCH[2:0] | | | SMB_MUX_PRES | EP_PRES |
| 1 | EP_TYPE[2:0] | | | PICPWR_DST_INDEX[2:0] | | | AFU | HOT_PLUG |
| 2 | EP_LANE_WIDTH[2:0] | | | INDIRECT_DISC_ORDER[3:0] | | | | INDIRECT |
| 4 | AFU | DST_INDEX_A[2:0] | | EP_LANE_OFFSET_A[3:0] | | | | |
| 5 | AFU | DST_INDEX_B[2:0] | | EP_LANE_OFFSET_B[3:0] | | | | |

**SMB_MUX_PRES :**

1 : indicates that a SMBus MUX exists between the connector and the destination

0 : there is no MUX present, and the physical routing is described by SMB_UP_CH

**SMB_MUX_DCH[2:0] :** When SMB_MUX_PRES=1, this field indicates which downstream channel (up to 8) of the MUX is physically routed to this end point.

**SMB_UP_CH[2:0] :** Indicates the physical connector index that sources SMBus to the end point (SMB_MUX_PRES=0) or to the upstream channel of a MUX (SMB_MUX_PRES=1)

000b = D1A (Destination Connector1, I2C_A)
001b = D1B (Destination Connector1, I2C_B)
010b = D2A
011b = D2B
100b = D3A
101b = D3B
110b = D4A
111b = D4B (Destination Connector4, I2C_B)

**AFU** : Available for Future Use

**EP_TYPE[2:0]** : Indicates the end point type that is being described to be attached to this lane group. Note: This bit field definitions for EP_TYPE may be specific to a DEVICE_CLASS
Examples (CEM Riser/Interposer):

000b = PCIe CEM Slot
001b = Embedded (Device Down) End Point
010b = Upstream port of a PCIe Switch
Others = AFU

Examples (Storage Class)

000b = NVMe Slot
001b = SAS/SATA Slot
001b = Universal Slot

**PICPWR_DST_INDEX[2:0]** :
Indicates which PICPWR destination connector provides power to this EP (End Point)

**HOT_PLUG** : Applicable to NVME direct attach storage or similar EP. Physical presence of the device is communicated via an out-of-Band mechanism, data fabric in-band method or dynamic virtual wire.

**EP_PRES** :

1 = Device is present

static=1 for embedded devices

downstream cable presence for cabled sources downstream of a switch

0 = Device such as an Add-In Card or downstream cable is not present

Not applicable to a hot plug EP.

**EP_LANE_WIDTH[2:0]** : //Note this is the connector width and not the slot installed add-in card.

000b = x1

001b = x2

010b = x4

011b = x8

100b = x16 (infers two sources required, all others only require one source)

All others reserved

Note: When EP_TYPE describes a PCIe CEM slot, the EP_LANE_WIDTH attribute describes the number of lanes routed to the slot. It does not describe the width of the Add In Card (AIC) that may be inserted into the slot.

**INDIRECT :**

0 = EP's data fabric physical routing is direct from the destination connector

1 = EP is downstream of a PCIe switch

**INDIRECT_DISC_ORDER[3:0]** :

Applicable when INDIRECT = 1 indicating that this destination is downstream of a PCIe switch. The discovery order index requires depth-first traversal during enumeration of the switch's downstream ports.

0000b = First

0001b = 2nd

…

111b = 15th

**DST_INDEX_x[2:0]** :

Indicates which destination connector(s) source data fabric lanes connect to this end point.

000b = Destination 1A

001b = Destination 1B

…

110b = Destination 4A

111b = Destination 4B

**EP_LANE_OFFSET_x[3:0] :** This field indicates the starting lane offset of the destination connector that the end point consumes.

EP_LANE_OFFSET_A[3:0] indicates which destination connector segment and lane are physically routed to lane 0 of the end point.
X8 Example: Natural order routing would indicate an offset of 0 from the High Speed I/O destination to lane 0 of the EP. Reverse order routing would indicate an offset of 7 from the destination to lane 0 of the EP.
EP_LANE_OFFSET_B[3:0] is only applicable to a EP with EP_LANE_WIDTH[2:0] = 100b (x16). All other end points only require physical routing description from a single DST_INDEX.

*Source Wire Descriptors and Stimulus Response*

| BYTE/BIT | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | P_D4 | P_D3 | P_D2 | P_D1 | COMM_SRC_TYPE | COMM_SRC_INDEX[2:0] | | |
| 1 | M_D4B | M_D4A | M_D3B | M_D3A | M_D2B | M_D2A | M_D1B | M_D1A |

**COMM_SRC_INDEX[2:0]** :
Indicates which source index is being used for frame-based communication. The physical routing of this wire is not able to be discovered through the stimulus response method. The corresponding destination index will always appear as '1b' in the discovery payload.

**COMM_SRC_TYPE :** 0b = High Speed I/O (aka M-XIO)
1b = Power (aka PICPWR)

**M_D*XY*** :
X = Destination connector index
Y = Destination connector sub-Index (1 per x8)
These fields will change state during the source discovery stimulus phase so that system management firmware can map the data fabric and I2C physical routing from the end point back to the planar.

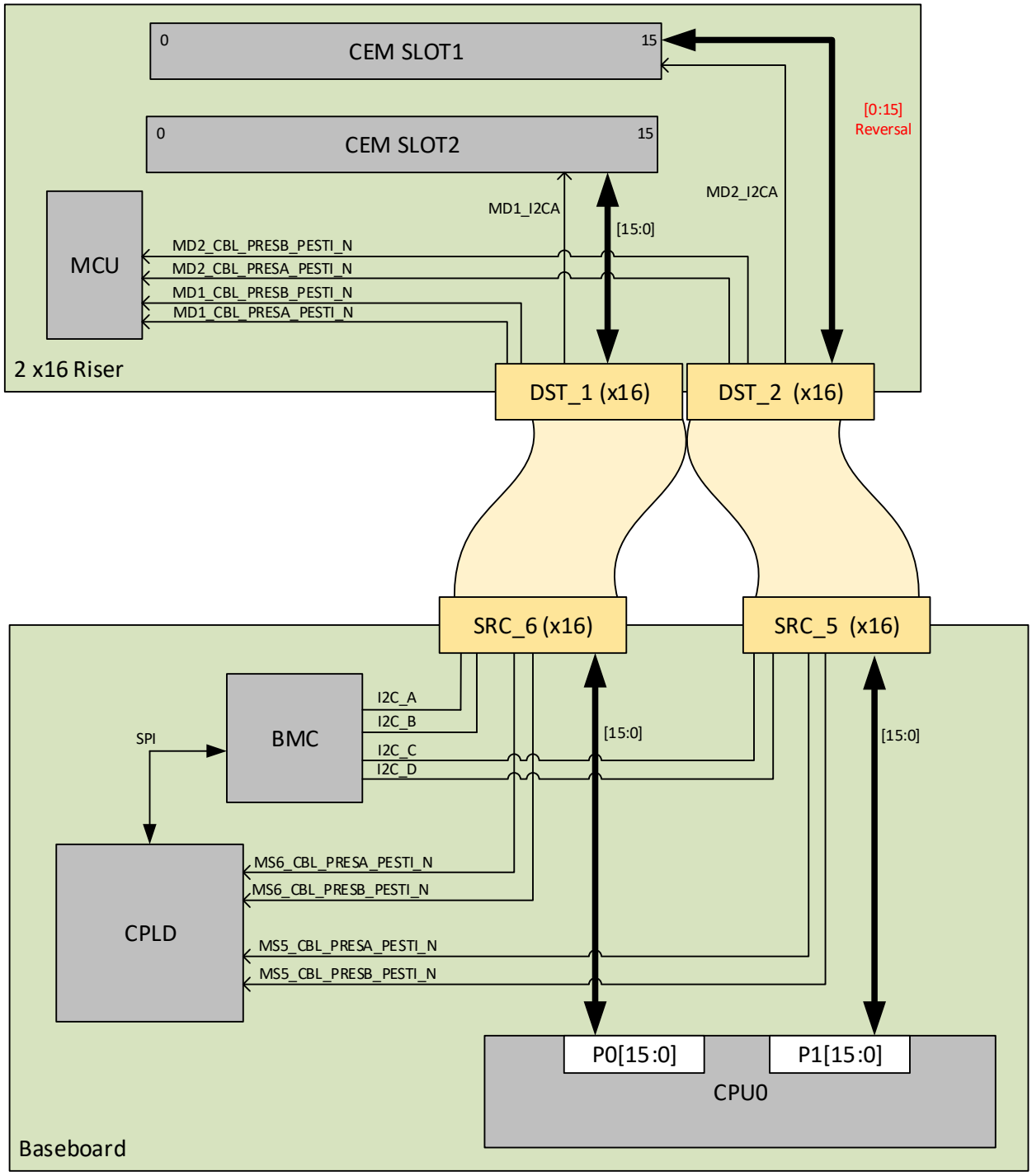
**P_D*X*** :
X = PICPWR destination index
These fields will change state during the source discovery stimulus phase so that system management firmware can map the power path to this end point. This is useful for optional enablement to a DPU EP in the ACPI S5 power state.

*Checksum*

| BYTE/BIT | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | CHECKSUM[7:0] | | | | | | | |

**CHECKSUM[7:0]** : CRC-8 checksum with polynomial = 0x07, Seed = 0x00

Example 2-slot Riser Block Diagram



The above example depicts an M-PESTI riser with two x16 CEM slots attached to two x16 cabled High Speed source connectors on the baseboard. All four of the M-PESTI wires are connected between the riser destination connectors and the target MCU so that it natively supports all valid source connections to the baseboard. Valid source connection topologies include one x16 or two x8 sources per x16 destination connector.

Payload Example (Initial Discovery Payload)

| Byte/Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Value |
|---|---|---|---|---|---|---|---|---|---|
| 00h | PAYLOAD_VERSION[7:0] = 00h | | | | | | | | 00h |
| 01h | DEVICE_CLASS[7:0] = 00h (CEM Riser) | | | | | | | | 00h |
| 02h | STATIC_PAYLOAD_SIZE[7:0]= 04h (32 Bytes) | | | | | | | | 04h |
| 03h | NUM_VIRTUAL_WIRE_OUTPUT BYTES[3:0] = 0000b | | | | NUM_VIRTUAL_WIRE_INPUT BYTES[3:0] = 0000b | | | | 00h |
| 04h | DEVICE_ID[15:8] = 00h | | | | | | | | 00h |
| 05h | DEVICE_ID[7:0] = 27h | | | | | | | | 27h |
| 06h | VENDOR_ID[15:8] = 80h | | | | | | | | 80h |
| 07h | VENDOR_ID[7:0] = 86h | | | | | | | | 86h |
| 08h | DEVICE_VERSION[7:0] = A0h | | | | | | | | A0h |
| 09h | NUM_DST_WIRES[3:0] = 0100b (4 CBL_PRES signals) | | | | AFU = 0b | NUM_PICPWR_DST_WIRES[2:0] = 001b | | | 41h |
| 0Ah | AFU = 0b | AFU = 0b | AFU = 0b | NUM_DST_DESCRIPTORS[4:0] = 00010b (Two Slots) | | | | | 02h |
| 0Bh | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | 00h |
| 0Ch | 010b (I2C Source is D2A Connector) | | | 000b (NA, No Mux) | | | 0b ( No Mux) | 1b (Present) | 41h |
| 0Dh | 000b (CEM slot) | | | 001b (P_D1 power) | | | 0b (AFU) | 0b (not HP) | 04h |
| 0Eh | 100b (x16 EP) | | | 000b (NA, Direct) | | | | 0 (Direct) | 80h |
| 0Fh | 0b (AFU) | 011b (D2A Destination) | | 0111b (Lane 7 Dest. to EP Lane 0, reversed) | | | | | 37h |
| 10h | 0b (AFU) | 010b (D2B Destination) | | 0111b (Lane 7 Dest. to EP Lane 8, reversed) | | | | | 27h |
| 11h | 000b (D1A Connector) | | | 000b (NA, No Mux) | | | 0b ( No Mux) | 0b (Absent) | 00h |
| 12h | 000b (CEM slot) | | | 001b (P_D1 power) | | | 0b (AFU) | 0b (not HP) | 04h |
| 13h | 100b (x16 EP) | | | 000b (NA, Direct) | | | | 0 (Direct) | 80h |
| 14h | 0b (AFU) | 000b (D1A Destination) | | 0000b (Lane 0 Dest. to EP Lane 0, natural order) | | | | | 00h |
| 15h | 0b (AFU) | 001b (D1B Destination) | | 0000b (Lane 0 Dest. to EP Lane 8, natural order) | | | | | 10h |
| 16h | 0b (NA) | 0b (NA) | 0b (NA) | 1b (valid) | 0b = Data Comm | 000b (D1A Connector) | | | 10h |
| 17h | 0b (NA) | 0b (NA) | 0b (NA) | 0b (NA) | 1b (valid) | 1b (valid) | 1b (valid) | 1b (valid) | 0Fh |
| 18h | 00h (Padding) | | | | | | | | 00h |
| 19h | 00h (Padding) | | | | | | | | 00h |
| 1Ah | 00h (Padding) | | | | | | | | 00h |
| 1Bh | 00h (Padding) | | | | | | | | 00h |
| 1Ch | 00h (Padding) | | | | | | | | 00h |
| 1Dh | 00h (Padding) | | | | | | | | 00h |
| 1Eh | 00h (Padding) | | | | | | | | 00h |
| 1Fh | 31h (Checksum) | | | | | | | | 31h |

## Supplemental Material B: Estimated latency for HW owned virtual wires

Assumptions:

- 242500 BAUD (250000 -3%); 8-O-1
- Worst case 8 device group with round robin servicing to each. This is if the source device wishes to reduce logic count by implementing for example 1 UART servicing 8X M-PESTI instances. Such an implementation would need a parallel BREAK detect circuit to not miss signaling from a not currently in use M-PESTI instance.

### Nominal Latency

*Dedicated Initiator Only:*

Single VWIRE byte in each direction

tAPBI2$_{MIN}$ + [2* tFRAME$_{MAX}$ TX (90.6 us)] + tAPTAR$_{TYP}$ (1.0 us) + [1 * tFRAME$_{MAX}$ RX (45.3 us)] = ***145.9 us***

Note: tAPBI$_{MAX}$ is not specified and is system dependent

*8 targets per Initiator:*

Single VWIRE byte in each direction

[[2* tFRAME$_{MAX}$ TX (90.6 us)] + tAPTAR$_{TYP}$ (1.0 us) + [1 * tFRAME$_{MAX}$ RX (45.3 us)]]

= 136.9 us * 8 devices

= **1167.2 us**

### Worst Case Latency

*Dedicated Initiator Only:*

For a dedicated initiator per target, the active phase RX timeout (tAPRTO) will add to the latency between each successive attempt to send the command and receive a valid response. The RX timeout period is ***much*** greater than (tAPTAR + 1 * tFRAME$_{MAX}$ RX) and would affect latency the most.

A single RX timeout would increase the nominal latency to:

2* [2* tFRAME$_{MAX}$ TX (90.6 us)] + tAPRTO = 681 us

*8 targets per Initiator:*

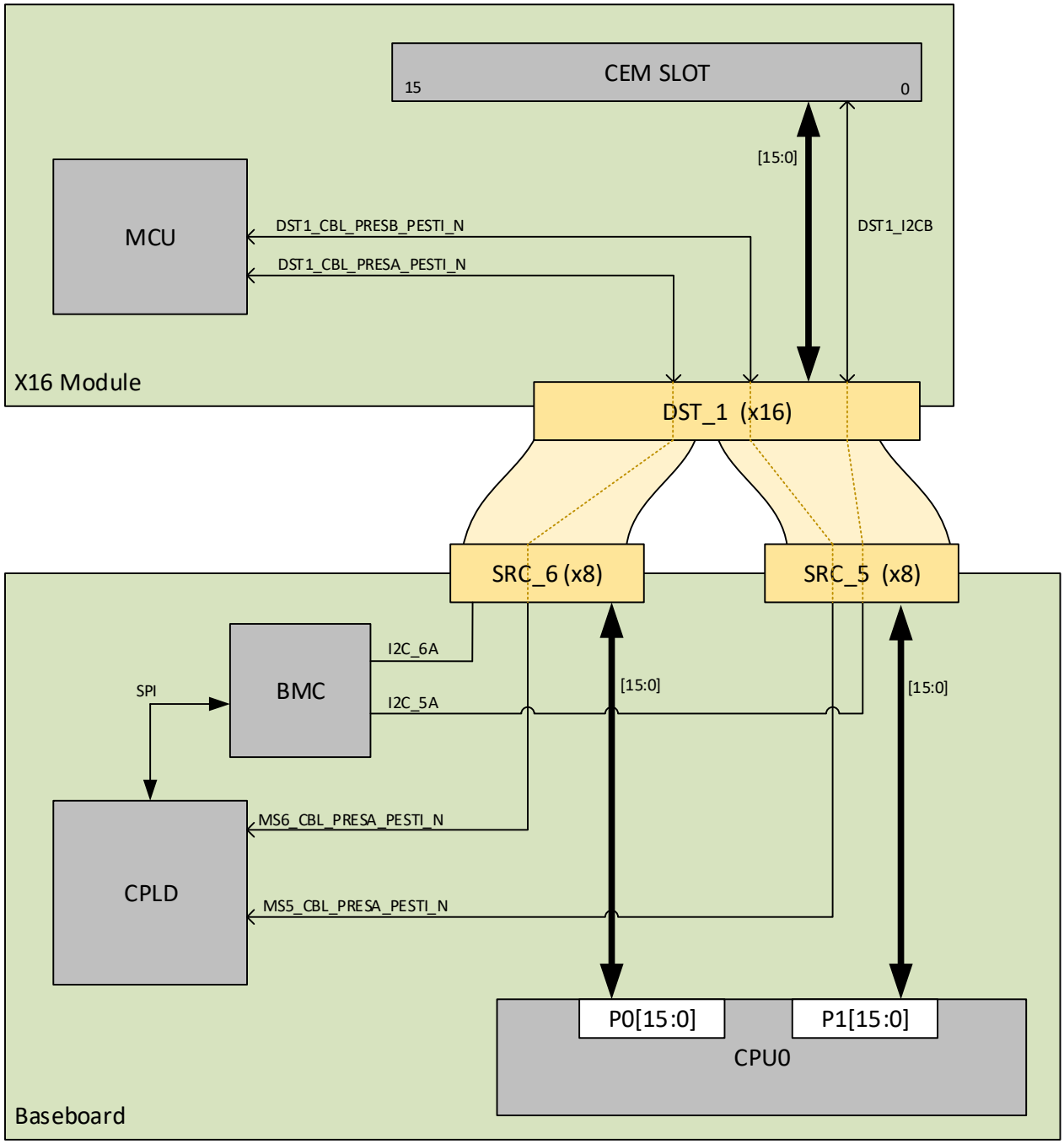
With 3 attempts (initial + 2 retries) per M-PESTI, the max latency with a single target not able to respond successfully adds significant latency to the nominal value of **1167.2 us**

**=** 1167.2 us - tAPTAR + tAPRO + 2* [135.9 us + tAPRTO] **=** 2.802 ms

Note: Firmware disablement (APEN=0) of one or more misbehaving M-PESTI wires would remediate the latency increase for the other devices within the group.

## Supplemental Material C: Cable Topology Case Studies

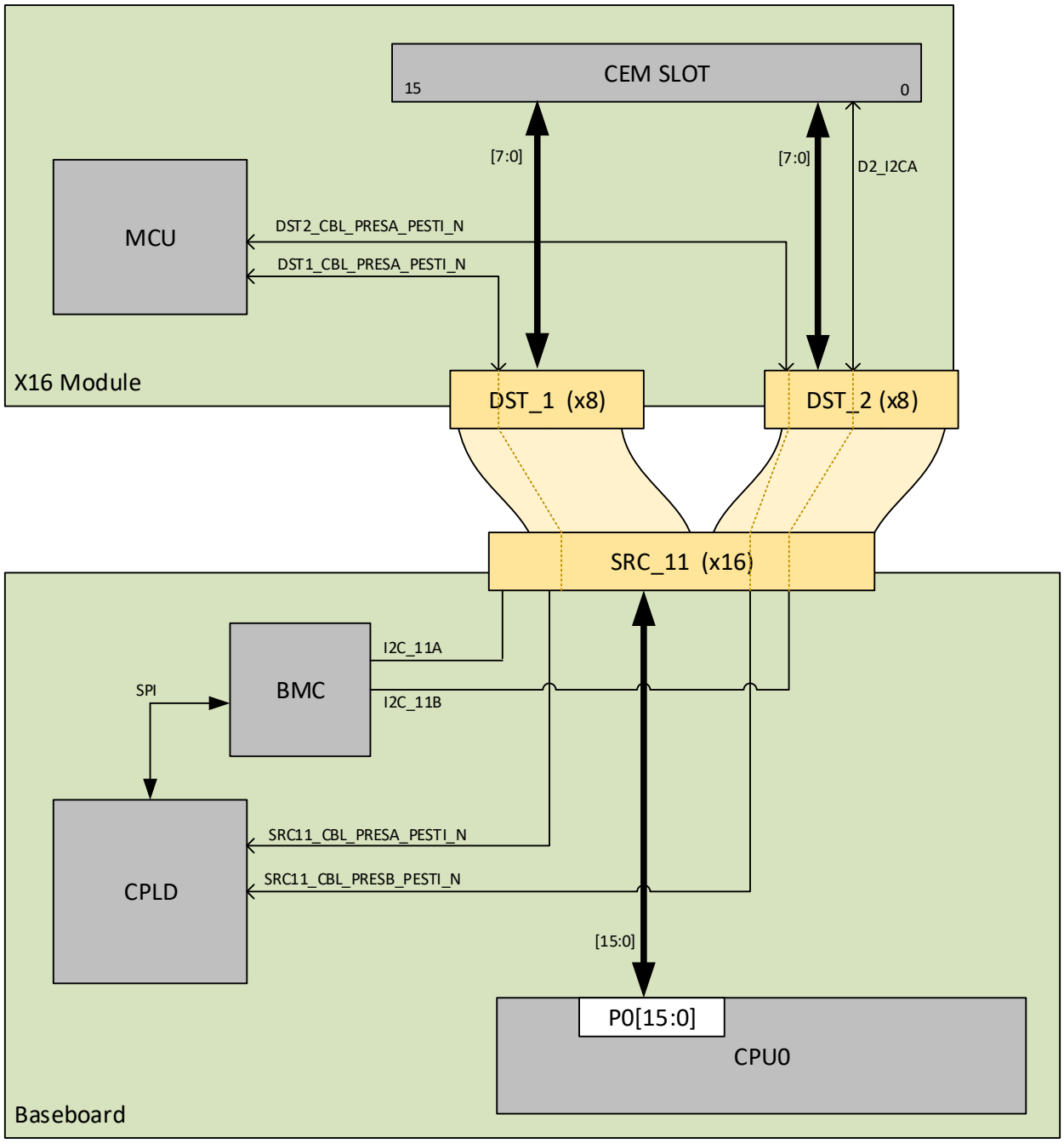
### Case 1: 2x8 Source to 1x16 End Point Block Diagram



## Case 1: 2x8 Source to 1x16 Destination to 1x16 End Point Payload

| Byte/Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Value |
|---|---|---|---|---|---|---|---|---|---|
| 00h | PAYLOAD_VERSION[7:0] = 00h | | | | | | | | 00h |
| 01h | DEVICE_CLASS[7:0] = 00h (CEM Riser) | | | | | | | | 00h |
| 02h | STATIC_PAYLOAD_SIZE[7:0]= 03h (24 Bytes) | | | | | | | | 03h |
| 03h | NUM_VIRTUAL_WIRE_OUTPUT BYTES[3:0] = 0000b | | | | NUM_VIRTUAL_WIRE_INPUT BYTES[3:0] = 0000b | | | | 00h |
| 04h | DEVICE_ID[15:8] = 00h | | | | | | | | 00h |
| 05h | DEVICE_ID[7:0] = 27h | | | | | | | | 27h |
| 06h | VENDOR_ID[15:8] = 80h | | | | | | | | 80h |
| 07h | VENDOR_ID[7:0] = 86h | | | | | | | | 86h |
| 08h | DEVICE_VERSION[7:0] = A0h | | | | | | | | A0h |
| 09h | NUM_DST_WIRES[3:0] = 0010b (2 CBL_PRES signals) | | | | AFU = 0b | NUM_PICPWR_DST_WIRES[2:0] = 000b | | | 20h |
| 0Ah | AFU = 0b | AFU = 0b | AFU = 0b | NUM_DST_DESCRIPTORS[4:0] = 00001b (One Slot) | | | | | 02h |
| 0Bh | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | 00h |
| 0Ch | 010b (I2C Source is D2A Connector) | | | 000b (NA, No Mux) | | | 0b ( No Mux) | 1b (Present) | 41h |
| 0Dh | 000b (CEM slot) | | | 000b (No discoverable power) | | | 0b (AFU) | 0b (not HP) | 00h |
| 0Eh | 100b (x16 EP) | | | 000b (NA, Direct) | | | | 0 (Direct) | 80h |
| 0Fh | 0b (AFU) | 000b (D1A Destination) | | | 0000b (Lane 0 Dest. to EP Lane 0, reversed) | | | | 00h |
| 10h | 0b (AFU) | 001b (D1B Destination) | | | 0000b (Lane 0 Dest. to EP Lane 8, reversed) | | | | 10h |
| 11h | 0b (NA) | 0b (NA) | 0b (NA) | 0b (NA) | 0b = Data Comm | 000b (D1A Connector) | | | 00h |
| 12h | 0b (NA) | 0b (NA) | 0b (NA) | 0b (NA) | 0b (NA) | 0b (NA) | 1b (valid) | 1b (valid) | 03h |
| 13h | 00h (Padding) | | | | | | | | 00h |
| 14h | 00h (Padding) | | | | | | | | 00h |
| 15h | 00h (Padding) | | | | | | | | 00h |
| 16h | 00h (Padding) | | | | | | | | 00h |
| 17h | 39h (Checksum) | | | | | | | | 39h |

## Case 2: 1x16 Source to 2x8 Destination to 1x16 End Point



## Case 2: 1x16 Source to 2x8 Destination to 1x16 End Point

| Byte/Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Value |
|---|---|---|---|---|---|---|---|---|---|
| 00h | PAYLOAD_VERSION[7:0] = 00h | | | | | | | | 00h |
| 01h | DEVICE_CLASS[7:0] = 00h (CEM Riser) | | | | | | | | 00h |
| 02h | STATIC_PAYLOAD_SIZE[7:0]= 03h (24 Bytes) | | | | | | | | 03h |
| 03h | NUM_VIRTUAL_WIRE_OUTPUT BYTES[3:0] = 0000b | | | | NUM_VIRTUAL_WIRE_INPUT BYTES[3:0] = 0000b | | | | 00h |
| 04h | DEVICE_ID[15:8] = 00h | | | | | | | | 00h |
| 05h | DEVICE_ID[7:0] = 27h | | | | | | | | 27h |
| 06h | VENDOR_ID[15:8] = 80h | | | | | | | | 80h |
| 07h | VENDOR_ID[7:0] = 86h | | | | | | | | 86h |
| 08h | DEVICE_VERSION[7:0] = A0h | | | | | | | | A0h |
| 09h | NUM_DST_WIRES[3:0] = 0010b (2 CBL_PRES signals) | | | | AFU = 0b | NUM_PICPWR_DST_WIRES[2:0] = 000b | | | 20h |
| 0Ah | AFU = 0b | AFU = 0b | AFU = 0b | NUM_DST_DESCRIPTORS[4:0] = 00001b (One Slot) | | | | | 02h |
| 0Bh | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | AFU = 0b | 00h |
| 0Ch | 010b (I2C Source is D2A Connector) | | | 000b (NA, No Mux) | | | 0b ( No Mux) | 1b (Present) | 41h |
| 0Dh | 000b (CEM slot) | | | 000b (No discoverable power) | | | 0b (AFU) | 0b (not HP) | 00h |
| 0Eh | 100b (x16 EP) | | | 000b (NA, Direct) | | | | 0 (Direct) | 80h |
| 0Fh | 0b (AFU) | 000b (D1A Destination) | | | 0000b (Lane 0 Dest. to EP Lane 0, reversed) | | | | 00h |
| 10h | 0b (AFU) | 010b (D2A Destination) | | | 0000b (Lane 0 Dest. to EP Lane 8, reversed) | | | | 20h |
| 11h | 0b (NA) | 0b (NA) | 0b (NA) | 0b (NA) | 0b = Data Comm | 000b (D1A Connector) | | | 00h |
| 12h | 0b (NA) | 0b (NA) | 0b (NA) | 0b (NA) | 0b (NA) | 1b (valid) | 0b (NA) | 1b (valid) | 05h |
| 13h | 00h (Padding) | | | | | | | | 00h |
| 14h | 00h (Padding) | | | | | | | | 00h |
| 15h | 00h (Padding) | | | | | | | | 00h |
| 16h | 00h (Padding) | | | | | | | | 00h |
| 17h | 66h (Checksum) | | | | | | | | 66h |

Note: The only bytes that change relative to Case 1 are payload offsets 0x10 and 0x12 to account for two PRESA wires from 2x8 destination connectors versus PRESA and PRESB wires connecting through a 1x16 destination connector.