

# ACPI for Arm Components 1.2

## **Platform Design Document**

Non-confidential

The Arm logo, consisting of the lowercase letters 'arm' in a bold, sans-serif font.



## Contents

Release information	4
Arm Non-Confidential Document License (“License”)	5
<b>About this document</b>	<b>7</b>
Terms and abbreviations	7
References	7
Feedback	8
Inclusive terminology commitment	8
<b>1 Introduction</b>	<b>9</b>
<b>2 ACPI for Arm Components</b>	<b>11</b>
2.1 ACPI Identifiers	11
2.2 Reserved ACPI IDs for legacy Arm components	11
2.3 Reserved ACPI IDs for Arm components defined by Arm BSA	11
2.4 Reserved ACPI IDs for features based on Arm specifications	12
2.5 Reserved ACPI IDs for generic devices	12
2.5.1 Generic Diagnostic Dump and Reset Device Interface	12
2.6 Arm components requiring ACPI description	13
2.6.1 Arm DMC620 Memory Controller	13
2.6.2 Arm DynamIQ Shared Unit (DSU)	14
2.6.3 Arm CoreLink CMN Coherent Mesh Network Family	17
2.6.4 Arm CoreLink Network-on-chip Interconnect Family	25

Copyright © 2020, 2021, 2022, 2023, 2024, 2025 Arm Limited. All rights reserved.

## Release information

The Change History table lists the changes that are made to this document:

Date	Version	Confidentiality	Changes
Jul 2025	1.2 EAC1	Non-Confidential	<ul style="list-style-type: none"> <li>Clarify that the GSIV in AGDI is edge-triggered.</li> </ul>
Apr 2025	1.2	Non-Confidential	<ul style="list-style-type: none"> <li>Reword the “CMN Coherent Mesh Network Family” Section.</li> <li>Add support for the CMN-S3 product.</li> <li>Clarify that the CMN RAS vendor-defined structure is defined per CMN product revision.</li> <li>Added reference model for describing CMN error nodes in the ACPI AEST table.</li> <li>Allow platform to support both SDEI and GSIV signaling in AGDI.</li> </ul>
Nov 2021	1.1	Non-Confidential	<ul style="list-style-type: none"> <li>Added HID for the HD LCD available in some Arm development platforms such as the Juno.</li> <li>Added clarifying text on the Arm Generic UART and its relation to the PL011.</li> <li>Added support for CMN-700, CMN-650 and the CoreLink Networkon-chip Interconnect family.</li> <li>Added support for DSU-110.</li> <li>Added chapter to describe HIDs for Arm standards based features.</li> <li>Added reference to Arm BSA specification, and updated language to reflect nomenclature thereof.</li> <li>Updated HID definition for Arm CoreSight PMU Architecture PMU device to match definitions in DEN0117.</li> <li>Added the ACPI AGDI table. This table describes a Generic Diagnostic Dump and Reset device interface.</li> <li>Added clarifying text on the differences between CMN-600 and other CMN networks in terms of the way their register blocks are organized.</li> <li>Updated device object description for NI-xy0 series to support description of more than one PMU overflow interrupts.</li> </ul>
Jul 2020	1.0	Non-Confidential	<ul style="list-style-type: none"> <li>First external release.</li> </ul>

## Arm Non-Confidential Document License (“License”)

This License is a legal agreement between you and Arm Limited (“**Arm**”) for the use of Arm’s intellectual property (including, without limitation, any copyright) embodied in the document accompanying this License (“**Document**”). Arm licenses its intellectual property in the Document to you on condition that you agree to the terms of this License. By using or copying the Document you indicate that you agree to be bound by the terms of this License.

“**Subsidiary**” means any company the majority of whose voting shares is now or hereafter owned or controlled, directly or indirectly, by you. A company shall be a Subsidiary only for the period during which such control exists.

This Document is **NON-CONFIDENTIAL** and any use by you and your Subsidiaries (“Licensee”) is subject to the terms of this License between you and Arm.

Subject to the terms and conditions of this License, Arm hereby grants to Licensee under the intellectual property in the Document owned or controlled by Arm, a non-exclusive, non-transferable, non-sub-licensable, royalty-free, worldwide License to:

- (i) use and copy the Document for the purpose of designing and having designed products that comply with the Document;
- (ii) manufacture and have manufactured products which have been created under the License granted in (i) above; and
- (iii) sell, supply and distribute products which have been created under the License granted in (i) above.

**Licensee hereby agrees that the Licenses granted above shall not extend to any portion or function of a product that is not itself compliant with part of the Document.**

Except as expressly licensed above, Licensee acquires no right, title or interest in any Arm technology or any intellectual property embodied therein.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm’s view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

Reference by Arm to any third party’s products or services within this document is not an express or implied approval or endorsement of the use thereof.

THE DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. Arm may make changes to the Document at any time and without notice. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

NOTWITHSTANDING ANYTHING TO THE CONTRARY CONTAINED IN THIS LICENSE, TO THE FULLEST EXTENT PERMITTED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, IN CONTRACT, TORT OR OTHERWISE, IN CONNECTION WITH THE SUBJECT MATTER OF THIS LICENSE (INCLUDING WITHOUT LIMITATION) (I) LICENSEE’S USE OF THE DOCUMENT; AND (II) THE IMPLEMENTATION OF

THE DOCUMENT IN ANY PRODUCT CREATED BY LICENSEE UNDER THIS LICENSE). THE EXISTENCE OF MORE THAN ONE CLAIM OR SUIT WILL NOT ENLARGE OR EXTEND THE LIMIT. LICENSEE RELEASES ARM FROM ALL OBLIGATIONS, LIABILITY, CLAIMS OR DEMANDS IN EXCESS OF THIS LIMITATION.

This License shall remain in force until terminated by Licensee or by Arm. Without prejudice to any of its other rights, if Licensee is in breach of any of the terms and conditions of this License then Arm may terminate this License immediately upon giving written notice to Licensee. Licensee may terminate this License at any time. Upon termination of this License by Licensee or by Arm, Licensee shall stop using the Document and destroy all copies of the Document in its possession. Upon termination of this License, all terms shall survive except for the License grants.

Any breach of this License by a Subsidiary shall entitle Arm to terminate this License as if you were the party in breach. Any termination of this License shall be effective in respect of all Subsidiaries. Any rights granted to any Subsidiary hereunder shall automatically terminate upon such Subsidiary ceasing to be a Subsidiary.

The Document consists solely of commercial items. Licensee shall be responsible for ensuring that any use, duplication or disclosure of the Document complies fully with any relevant export laws and regulations to assure that the Document or any portion thereof is not exported, directly or indirectly, in violation of such export laws.

This License may be translated into other languages for convenience, and Licensee agrees that if there is any conflict between the English version of this License and any translation, the terms of the English version of this License shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. No license, express, implied or otherwise, is granted to Licensee under this License, to use the Arm trade marks in connection with the Document or any products based thereon. Visit Arm's website at <http://www.arm.com/company/policies/trademarks> for more information about Arm's trademarks.

The validity, construction and performance of this License shall be governed by English Law.

Copyright © 2020, 2021, 2022, 2023, 2024, 2025 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

Arm document reference: PRE-21585

version 5.0, March 2024

## About this document

### Terms and abbreviations

Term	Meaning
ACPI	Advanced Configuration and Power Interface
AEST	Arm Error Source Table
ASL	ACPI Source Language
CMN	Arm CoreLink Mesh Network
DSU	DynamicIQ Shared Unit
DTC	Debug and Trace Controller
GIC	Arm Generic Interrupt Controller
GSIV	Global System Interrupt Vector
HN	Home Node
PMU	Performance Monitoring Unit
RN	Requester Node
SDEI	Software Delegated Exception Interface
SPE	Statistical Profiling Extension
XP	Cross-point

### References

This section lists publications by Arm and by third parties.

See Arm Developer (<http://developer.arm.com>) for access to Arm documentation.

- [1] *Advanced Configuration and Power Interface Specification*. UEFI Forum, <https://uefi.org/specifications>.
- [2] *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile: ARM DDI 0487F.b (ID040120)*. Arm Limited.
- [3] *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4: Arm IHI 0069E (ID012119)*. Arm Limited.
- [4] *Arm® System Memory Management Unit Architecture Specification SMMU architecture versions 3.0, 3.1 and 3.2: Arm IHI 0070C.a*.
- [5] *Arm® CoreSight™ Architecture Specification v3.0: Arm IHI 0029E (ID022717)*. Arm Limited.
- [6] *Arm® Base System Architecture 1.0: DEN0094*. Arm Limited.
- [7] *Arm IO Remapping Table: DEN0049E*. Arm Limited.
- [8] *ACPI for CoreSight™ 1.1: DEN0067*. Arm Limited.
- [9] *Arm® Functional Fixed Hardware Specification Document number: Arm DEN 0048A*. Arm Limited, <https://uefi.org/acpi>.

- [10] *\_DSD (Device Specific Data) Implementation Guide v1.2*. UEFI Forum, <https://uefi.org/specifications>.
- [11] *ACPI for Arm v8-A Memory System Resource Partitioning and Monitoring: DEN0065*. Arm Limited.
- [12] *Arm Architecture Reference Manual Supplement Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A: ARM DDI 0598B.a*. Arm Limited.
- [13] *ACPI for the Armv8-A RAS Extensions 1.0: DEN0085*. Arm Limited.
- [14] *Arm® Reliability, Availability, and Serviceability (RAS) Specification: Arm DDI 0587C.b*. Arm Limited.
- [15] *ACPI for Arm® CoreSight™ Performance Monitoring Unit Architecture: DEN0117*. Arm Limited.
- [16] *Arm® CoreSight™ Performance Monitoring Unit Architecture Specification*. Arm Limited.
- [17] *Serial Port Console Redirection Table*. Microsoft Corporation, <https://uefi.org/acpi>.
- [18] *Debug Port Table 2*. Microsoft Corporation, <https://uefi.org/acpi>.
- [19] *Arm® DynamIQ™ Shared Unit, Revision r0p2, Technical Reference Manual: Arm 100453\_0002\_00\_en*. Arm Limited.
- [20] <https://developer.arm.com/ip-products/system-ip/corelink-interconnect/corelink-coherent-mesh-network-family>. Arm Limited.
- [21] *Arm Architecture Reference Manual for A-profile architecture: DDI0487*. Arm Limited.
- [22] *ACPI for Arm RASv8 Extension: DEN0085*. Arm Limited.
- [23] <https://developer.arm.com/ip-products/system-ip/corelink-interconnect/corelink-network-interconnect-family>. Arm Limited.
- [24] *Arm® Power State Coordination Interface Platform Design Document: DEN0022D*. Arm Limited.
- [25] *Software Delegated Exception Interface: DEN0054*. Arm Limited.

## Feedback

Arm welcomes feedback on its documentation.

If you have comments on the content of this manual, send an e-mail to [errata@arm.com](mailto:errata@arm.com). Give:

- The title (ACPI for Arm Components).
- The document ID and version (DEN0093 1.2).
- The page numbers to which your comments apply.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

### Inclusive terminology commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used terms that can be offensive. Arm strives to lead the industry and create change. We believe that this document contains no offensive terms. If you find offensive terms in this document, please contact [terms@arm.com](mailto:terms@arm.com).



# 1 Introduction

This document provides guidance for describing system components implemented or licensed by Arm, and their properties, in ACPI [1].

This specification does not cover ACPI description of the Arm Architecture [2] or related architectures, for example the Generic Interrupt Controller Architecture [3], the System Memory Management Unit Architecture [4], the CoreSight Architecture [5], or architected components that are described in the Base System Architecture [6]. The ACPI descriptions of these architectural components are covered in the following specifications instead:

**Table 3: Arm Architectures that are covered by ACPI Tables**

<i>Specification</i>	<i>Table or object</i>	<i>Arm Architecture Covered</i>
ACPI [1]	MADT	Arm Architecture [2]
ACPI [1]	MADT	GIC [3], SPE
I/O Remapping Table [7]	IORT	SMMU [4]
ACPI For CoreSight [8]	ACPI graph	CoreSight Architecture [5]
FFH for Arm [9]	FFH	Armv8 Activity Monitors Extension
ACPI _DSD Implementation Guide [10]	ACPI graph	CoreSight Architecture [5]
ACPI for MPAM [11]	MPAM	Armv8 MPAM Architecture [12]
ACPI for Arm RAS Extensions [13]	AEST	Armv8 RAS Extensions Architecture [14]
ACPI for Arm CoreSight PMU Architecture [15]	APMT	Armv8 CoreSight PMU Architecture [16]

The following table details coverage of specific Arm components in ACPI:

**Table 4: Arm Components that are covered by specific ACPI Tables**

<i>ACPI Tables</i>	<i>Arm Components Covered</i>
GTDT	Arm Generic Watchdog timer [6]
PPTT	Caches, cores, clusters
SPCR [17], DBG2 [18],	Arm Generic UART, PL011

This specification recommends that components that are not covered in standard or Arm-specific ACPI tables are described as ACPI device objects in ASL. Because a component might have multiple internal interfaces, Arm recommends that a separate ACPI device object is created to cover each of these interfaces to allow device drivers in OS to unequivocally bind to those specific interfaces. Interfaces like RAS Extensions and MPAM are not covered, and instead are described in appropriate tables, as indicated in Table 3 above.

The device objects should define the generic and specific properties of the component interface to aid software discovery from the OS, where:

- Generic properties include an ACPI namespace identifier for the component or its interface, the memory-mapped base address of the component or its interface, and the interrupts that the component can generate.
- Specific properties are component specific or vendor specific or both.

## 2 ACPI for Arm Components

### 2.1 ACPI Identifiers

ACPI Identifiers of Arm components follow the conventions that are described in [1]. The ACPI Hardware ID object, `_HID`, is used as the primary identifier. For Arm components, the format is `ARMH####`.

For some Arm components, existing standard ACPI or PNP identifiers may also be used as `_HID` values. In such cases, Arm recommends setting the `_CID` of the device object as `ARMH####` where relevant.

### 2.2 Reserved ACPI IDs for legacy Arm components

This section lists reserved ACPI IDs for components from Arm that are managed as a complete unit by a single device driver, or are required for cross-referencing from other Arm ACPI tables like the AEST [13] and the MPAM [12]. These components are listed in Table 5.

**Table 5: Reserved ACPI IDs for legacy Arm components**

Component	HID
Prime cell UART (PL011)	ARMH0011
Prime cell General Purpose I/O	ARMH0061
Arm HD LCD	ARMH0002

### 2.3 Reserved ACPI IDs for Arm components defined by Arm BSA

The following types of Arm components require a unique ACPI ID for identification:

- Specifically defined by the BSA specification [6]
- Not described in any standard ACPI table listed in Table 3 or Table 4
- Must be described in ACPI namespace by virtue of the above properties

**Table 6: Reserved ACPI IDs for BSA components**

BSA Component	HID	Notes
Arm Generic UART	ARMHB000	Some operating systems use the PL011 HID (see Table 5 above) to bind to the Arm Generic UART in the system. While this practice is flawed and not encouraged by Arm, Arm acknowledges that it must be permitted until formal support for the Arm Generic UART HID is made available in these operating systems. Arm strongly recommends use of the Arm Generic UART HID going forward.

A BSA component can be described in both ACPI namespace and the standard ACPI tables in a given system. For example, a system might have two instances of the Arm Generic UART: a primary UART that is used by the OS and a secondary, general-purpose, UART for application usage. In this example system, the primary

UART must be declared in the ACPI SPCR table. The secondary UART might be declared as a device object in ACPI namespace and assigned the HID that is defined in Table 6.

## 2.4 Reserved ACPI IDs for features based on Arm specifications

This section lists reserved ACPI IDs for components that are based on features described in Arm specifications. Such components are primarily described in Arm-specific ACPI tables, but might also have additional vendor-specific properties that can only be described by means of ACPI device objects in DSDT. The recommended practice is to link such device objects to the corresponding table node in the Arm ACPI table that describes the component. The standard HIDs for these device objects enable the OS to understand that they can be managed by standard drivers that conform to the Arm ACPI table based enumeration and configuration of the components.

**Table 7: HIDs for features described in Arm specifications**

Component	HID
MPAM Memory System Component (MSC)	ARMHAA5C
CoreSight PMU Architecture PMU	ARMHE001

## 2.5 Reserved ACPI IDs for generic devices

This section describes devices that are generic across Arm implementations and are not tied to an IP implementation.

### 2.5.1 Generic Diagnostic Dump and Reset Device Interface

Some use-cases, such as system management, require the ability to generate a non-maskable event to the OS to request the OS kernel to perform a diagnostic dump and reset the system. This section describes the ACPI representation for a Generic Diagnostic Dump and Reset Device for Arm systems that serves as an interface for initiating the non-maskable event on behalf of requesting agents.

The Generic Diagnostic Dump and Reset device is described using the ACPI table representation outlined in Section 2.5.1.1.

The interface supports both SDEI and native interrupt based generation of non-maskable events.

#### 2.5.1.1 ACPI Table Representation

**Table 8: ACPI AGDI Table**

Field	Byte length	Byte offset	Description
<b>Header</b>			Standard ACPI format for header.
Signature	4	0	'AGDI', Arm Generic Diagnostic Dump and Reset Interface table.
Length	4	4	Length of this table in bytes.
Revision	1	8	Must be 0 for version 1.0 of this specification.
Checksum	1	9	The entire table must sum to zero.

Field	Byte length	Byte offset	Description
OEM ID	6	10	OEM ID.
OEM Table ID	8	16	The table ID is the manufacture model ID.
OEM Revision	4	24	OEM revision of the AGDI table for the supplied OEM Table ID.
Creator ID	4	28	The vendor ID of the utility that created the table.
Creator Revision	4	32	The revision of the utility that created the table.
<b>Body</b>			
Flags node structures	1	36	Bits [1:0]: Signaling mode: <ul style="list-style-type: none"> <li>• 0: SDEI-based Signaling mode.</li> <li>• 1: Interrupt-based Signaling mode.</li> <li>• 2: Both SDEI and Interrupt-based signaling is supported. While an SDEI event handler is registered, the platform is allowed to not generate the wired interrupt.</li> </ul> All other values are reserved for future use by this specification.
Reserved	3	37	Reserved, must be zero.
SDEI Event number	4	40	SDEI Event number of the event generated by the device. This field is valid only if the signaling mode is set to 0 or 2.
GSIV	4	44	The GSIV of the interrupt that is generated by the device. This field is valid only if the signaling mode is set to 1 or 2. This interrupt is edge-triggered.

## 2.6 Arm components requiring ACPI description

This section describes Arm components that have at least one interface that is not covered by standard or Arm-specific ACPI tables, and that must be described in ACPI namespace.

### 2.6.1 Arm DMC620 Memory Controller

Table 9 describes interfaces within the DMC620 that require ACPI description to support software discovery.

#### 2.6.1.1 Interface identification

**Table 9: Arm DMC620 Memory Controller HID values**

Value	Description
ARMHD620	ACPI Hardware Identifier for the DMC620 PMU.

#### 2.6.1.2 The DMC620 PMU

The DMC620 PMU is assigned the HID value of ARMHD620, as specified in Table 9.

**Device configuration objects for the DMC620 PMU****Table 10: Configuration objects for the DMC620 PMU**

Object	Values	Type	Description
_CRS	Base address	QWordMemory	Base address of the PMU in the system address map
	GSIV	Interrupt	GSIV of the PMU overflow interrupt

**ASL reference code for the DMC620 PMU**

The DMC620 PMU register space is mapped at a 512-byte range that begins at offset 0x80000A00 in the system address space. In this reference code, it is assumed that the PMU overflow interrupt from the DMC620 is mapped to GSIV 312.

```
Device(MC00) { // PMU interface on the example DMC620 memory controller
               // instance in the system.

    Name(_HID, "ARMHD620")
    Name(_CID, "ARMHD620")
    Name(_UID, 0)
    Name(_CCA, 1)
    Name(_STR, Unicode("Socket0:MCU0"))
    Name(_STA, 0, NotSerialized) {
        Return(0x0f)
    }

    Name(_CRS, ResourceTemplate() {
        // Descriptor for 64-bit memory-mapped register space
        // of the DMC620
        QWordMemory(
            ResourceProducer, // ResourceUsage
            PosDecode,        // Decode
            MinFixed,         // Min range is fixed
            MaxFixed,         // Max range is fixed
            NonCacheable,    // Cacheable
            ReadWrite,        // ReadAndWrite
            0x0000000000000000, // Address Granularity - GRA
            0x0000100080000A00, // AddressMinimum - MIN
            0x0000100080000BFF, // AddressMaximum - MAX
            0x0000000000000000, // AddressTranslation - TRA
            0x0000000000000200, // RangeLength - LEN
            ,                  // ResourceSourceIndex
            ,                  // ResourceSource
            CFGS               // DescriptorName
        )

        // PMU overflow Interrupt, with GSIV = 312
        Interrupt(ResourceConsumer, Level, ActiveHigh, Exclusive) {312}
    })
}
```

**2.6.2 Arm DynamIQ Shared Unit (DSU)**

The Arm DynamIQ Shared Unit (DSU) provides circuitry, logic, interfaces, and an optional shared cache to support a DynamIQ cluster. The DSU is described in detail in [19]. Interfaces within the DSU that require

ACPI description to support software discovery are described here.

The DSU is shared by a set of cores organized as a cluster. The ACPI description of interfaces within the DSU is therefore expressed as a device object that is a child of the ACPI processor container object that describes the cluster.

The OS must parse the CPU topology in ACPI namespace to discover the DSU interface objects and to understand the associativity between those DSU instances and cores.

### 2.6.2.1 Interface Identification

**Table 11: Arm DSU HID values**

Value	Description
ARMH500	ACPI Hardware Identifier for the DSU PMU.
ARMH501	ACPI Hardware Identifier for the common DSU elements.
ARMH510	ACPI Hardware Identifier for the DSU 110 PMU.

### 2.6.2.2 Common DSU elements

Common DSU elements are collectively described by a single device object with a HID of ARMH501. This device object allows cross-referencing the DSU object from other ACPI objects and tables.

### 2.6.2.3 DSU PMU

The DSU provides a PMU for monitoring and recording miscellaneous events. DSU PMU registers are presented as System Registers to allow for native software discovery. Therefore, no ACPI description is required to locate them. When a PMU counter overflows, the PMU asserts an interrupt signal that can be routed to an interrupt controller such as the GIC.

The DSU PMU is assigned the HID value of ARMH500, as specified in Table 11.

### DSU PMU device configuration objects

**Table 12: Arm DSU PMU device configuration objects**

Object	Value	Type	Description
_CRS	GSIV	Interrupt	GSIV of the DSU PMU overflow interrupt

### ASL reference code

This reference code illustrates how the CPU topology that is associated with the DSU is described in ACPI namespace by including the DSU device objects in the CPU hierarchy description. The code showcases an example system that has two clusters, each with a single DSU. Each cluster includes two CPU cores that share the associated DSU.

The code also illustrates how the global DSU device object may be also included within the CPU topology description. The placement of the global DSU object allows generic references to the DSU from the OS.

```
Device (SYSM) { // System level states
    Name (_HID, "ACPI0010")
    Name (_UID, 0)
    Name (_LPI,
        Package() { ... }
    )
}
```

```

Device (CLU0) { // Cluster 0
    Name (_HID, "ACPI0010")
    Name (_UID, 1)
    Name (_LPI,
        Package() {...}
    )

    Device (DSU0) { // Common DSU Instance 0 associated with cluster 0
        Name (_HID, "ARMHD501")
        Name (_UID, 0)
    } // DSU descriptor ends here

    Device (DSP0) { // PMU interface on DSU 0 associated with cluster 0
        Name (_HID, "ARMHD500")
        Name (_UID, 0)
        Name(_CRS, ResourceTemplate () {
            // PMU overflow Interrupt, with GSIV = 302
            Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {302}
        })
    } // DSU0 PMU interface descriptor ends here

    Device (CPU0) { // Core0
        Name (_HID, "ACPI0007")
        Name (_UID, 0)
        Method (_LPI, 0, NotSerialized) {
            ...
        }
    } // CPU0 description ends here

    Device (CPU1) { // Core1
        Name (_HID, "ACPI0007")
        Name (_UID, 1)
        Method (_LPI, 0, NotSerialized) {
            ...
        }
    } // CPU1 description ends here
} // Cluster 0 descriptor ends here

Device (CLU1) { // Cluster 1
    Name (_HID, "ACPI0010")
    Name (_UID, 2)
    Method (_LPI, 0, NotSerialized) {
        ...
    }

    Device (DSU1) { // Common DSU Instance 1 associated with cluster 1
        Name (_HID, "ARMHD501")
        Name (_UID, 1)
    } // DSU descriptor ends here

    Device (DSP1) { // PMU interface on DSU 1 associated with cluster 1
        Name (_HID, "ARMHD500")
        Name (_UID, 1)
        Name(_CRS, ResourceTemplate () {
            // PMU overflow Interrupt, with GSIV = 402
            Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {402}
        })
    } // DSU1 PMU interface descriptor ends here
}

```



```

Device (CPU2) { // Core2
    Name (_HID, "ACPI0007")
    Name (_UID, 2)
    Method (_LPI, 0, NotSerialized) {
        ...
    }
} // CPU2 description ends here

Device (CPU3) { // Core3
    Name (_HID, "ACPI0007")
    Name (_UID, 3)
    Method (_LPI, 0, NotSerialized) {
        ...
    }
} // CPU3 description ends here
} // Cluster 1 descriptor ends here
} // End of system description

```

### 2.6.3 Arm CoreLink CMN Coherent Mesh Network Family

The Arm CoreLink CMN Coherent Mesh Network components are described in [20]. The CMN network consists of Cross-points (XPs), Home Nodes (HNs) and Request Nodes (RNs). A special Home Node, HN-D, houses the global configuration registers. In this network, PMU logic is integrated into Debug and Trace Logic Controllers (DTCs). HN-D also houses the primary DTC, which is labelled DTC0. Other DTCs, DTC1 to DTC3, are housed in special Home Nodes called HN-T. The CMN layout is illustrated in the following diagram:

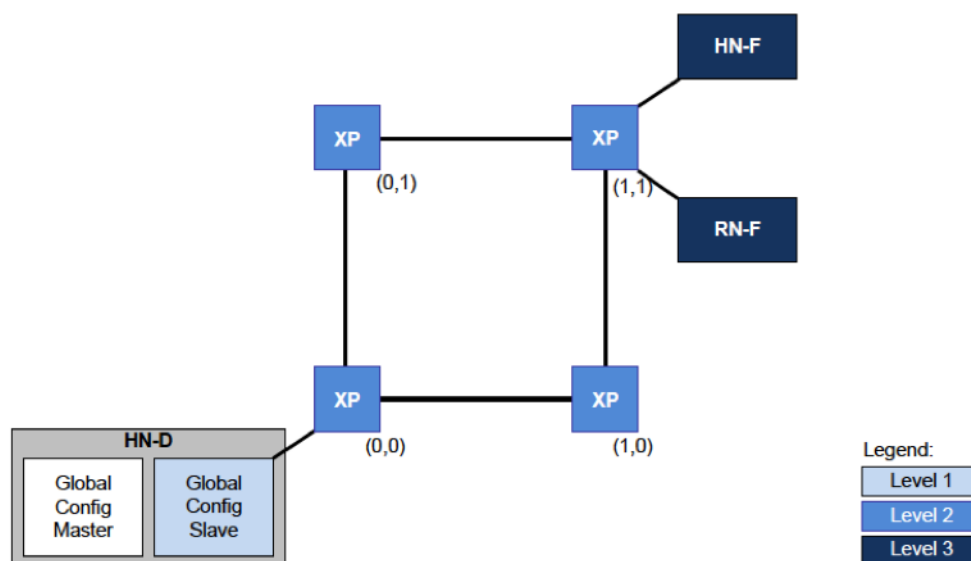
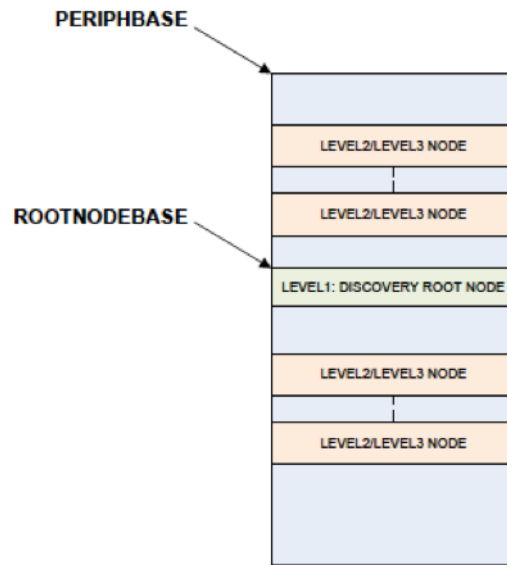


Figure 1: High-level layout of the CMN

All configuration registers that belong to the CMN network are mapped into System Address Map (SAM) at a pre-determined, 64MB aligned address range at offset PERIPHBASE. For the CMN-600, the configuration space of the root node of the network is mapped to an address range at offset ROOTNODEBASE. These offsets are illustrated in the following diagram:



**Figure 2: Memory-mapped configuration region of the CMN-600**

All registers are organized into one or more register blocks. Each register block is 16KB in size, and there is one register block for each logical block in the network. Each register block is called a node. The nodes are laid out in a tree hierarchy. For the CMN-600, the root of the tree hierarchy is the root node, as illustrated in Figure 2. For the other CMN series products, the register blocks start at PERIPHBASE itself.

#### **2.6.3.1 Interface Identification**

Notation: The `_HID` is composed of the following sub-identifier fields:

- The first four characters, “ARMH”, indicate that this is an Arm-specific hardware.
- The fifth character is set to ‘C’ to indicate that this is a product from the CoreLink interconnect family.
- The last three characters are based on the product name.

The `_HIDs` are defined, per CMN sub-system, in the following Sections:

- Section 2.6.3.2 : `_HID` for CMN PMU
- Section 2.6.3.3 : `_HID` for CMN RAS nodes

#### **2.6.3.2 CMN PMU**

CMN PMU logic is integrated into Debug and Trace Logic Controllers (DTCs). Thus, discovery of the PMU interface in a CMN network relies on the topology in which the DTCs are organized within the network.

The HN-D houses the primary DTC, which is labelled DTC0. Each DTC is associated with multiple DTMs. The parent DTC and its children DTMs form a DTC domain. The following diagram shows an example system with two DTC domains:

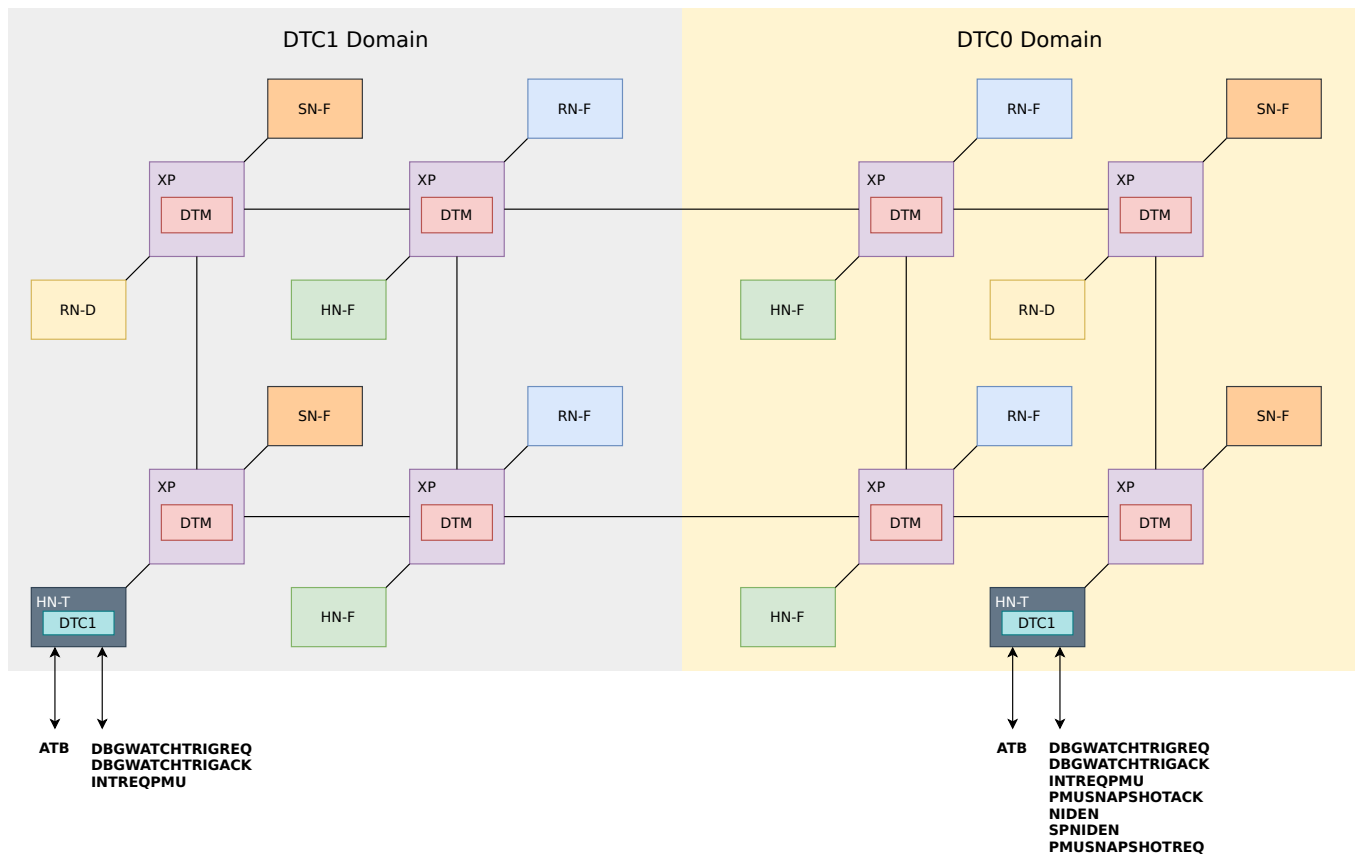


Figure 3: CMN-based topology with 2 DTCs

Each PMU asserts its own dedicated interrupt signal on overflow, called INTREQPMU. The interrupts from the PMUs are routed to the GIC, and appear as one or more GSIVs to software.

This means that, to support software discovery of CMN PMUs, the primary sources of required information are:

Table 13: Information sources for CMN PMU discovery

Information	Information source
Discovery of PMUs in CMN network. This is based on discovery of DTC $n$ for PMU $n$	PERIPHBASE
Discovery of PMUs in CMN-600 only. For the CMN-600, register blocks are organized in a tree hierarchy that begins at offset ROOTNODEBASE.	ROOTNODEBASE
Identity of PMU overflow interrupt for all PMU logic within the network, where PMU $n$ is housed in DTC $n$	INTREQPMU $n$ routing

#### **PMU overflow interrupt description ordering and Logical ID**

The CMN network specification allows for up to four DTCs in the system. DTC0 is the primary DTC, and must always exist. Additional DTCs may be installed on various cross-points, based on system design and topology.

The CMN provides logical numbering of DTCs to allow for their unique identification in hardware. The Logical ID is the only way for software to identify a unique DTC. However, the Logical IDs is not guaranteed to be the same as the DTC number. For example, DTC 1 is not guaranteed to obtain a Logical ID of 1.

The following is an example mapping for a CMN system with 4 DTCs:

**Table 14: DTC domain number to Logical ID mapping in an example system**

DTC Name	DTC Domain Number (Value)	DTC Logical ID (Notation)	DTC Logical ID (Value)	Description
DTC0	0	DTC[0]	0	DTC0 is always assigned Logical ID of 0.
DTC1	1	DTC[1]	$j$	DTC1 is assigned Logical ID $j$ , where $j \neq 0$ .
DTC2	2	DTC[2]	$k$	DTC2 is assigned Logical ID $k$ , where $j \neq 0$ & $k \neq j$ .
DTC3	3	DTC[3]	$m$	DTC1 is assigned Logical ID $m$ , where $m \neq 0$ & $m \neq j \neq k$ .

To address the lack of an explicit relation between the Logic ID and Domain Number as shown in Table 14, the PMU overflow interrupt descriptors are organized in increasing order of the hardware assigned Logical IDs of the related DTCs, respectively. This allows OS software to map interrupt descriptors to their parent DTCs. This means that INTDESC[1] corresponds to DTC0, INTDESC[2] corresponds to the DTC that is assigned the next smallest Logical ID, and so on.

#### **CMN PMU ACPI device identification**

The CMN PMU ACPI devices are identified by the \_HIDs represented in Table 15.

**Table 15: Arm CMN \_HID values**

Value	Description
ARMHC600	ACPI Hardware Identifier for the CMN-600 PMU
ARMHC650	ACPI Hardware Identifier for the CMN-650 PMU
ARMHC700	ACPI Hardware Identifier for the CMN-700 PMU
ARMHC003	ACPI Hardware Identifier for the CMN-S3 PMU

#### **Device configuration objects**

**Table 16: Arm CMN Coherent Mesh Network configuration objects**

Object	Values	Type	Description
_CRS	PERIPHBASE	QWordMemory	Base address of the memory-mapped region in the system address map where the CMN registers are mapped.
	ROOTNODEBASE	QWordMemory	Base address of the root node. This field is specific to the CMN-600 device object. This field is absent for products other than CMN-600.
	GSIV[ <i>n</i> ]	Interrupt	List of GSIVs of <i>n</i> distinct interrupts that can be generated by the <i>n</i> PMUs located in this instance of the CMN.

The list must always begin with PERIPHBASE, followed by ROOTNODEBASE if this is a CMN-600 device object, and finally the interrupt resource descriptors. Similarly, the interrupt resource descriptors for the PMUs, for example GSIV[*n*], must appear in numerically increasing order of the corresponding DTC[*n*]. So the first interrupt resource descriptor relates to DTC[0], the second interrupt resource descriptor relates to DTC[1] and so on. The *m*th interrupt resource descriptor in the list corresponds to DTC[*m*-1].

#### **ASL code example**

This code example showcases a CMN-600 based network with a dimension larger than 4x4 and that has two DTCs. Like the example illustrated in Figure 3 in Section 2.6.3.2, the first DTC is always DTC0 and is attached to HN-D. However, the second DTC could be assigned any Logical ID that will be non-zero. The term DTC[*n*] is used to represent the Logical ID of DTC*n*. In this example, PERIPHBASE of the CMN-600 is set as 0xAFE0000000, and the root node is at an offset of 0xC000 from PERIPHBASE.

```

Device(CMN6) { // CMN-600 device object for an X * Y mesh where X, Y > 4
    Name(_HID, "ARMHC600")
    Name(_CRS, ResourceTemplate() {
        // Descriptor for 256 MB of the CFG region at offset PERIPHBASE
        QWordMemory (
            ResourceConsumer, // bit 0 of general flags is 0
            PosDecode,
            MinFixed, // Range is fixed
            MaxFixed, // Range is Fixed
            NonCacheable,
            ReadWrite,
            0x00000000, // Granularity
            0xAFE0000000, // Min, set to PERIPHBASE
            0xAFEFFFFFFF, // Max
            0x00000000, // Translation
            0x0010000000, // Range Length = 256MB
            , // ResourceSourceIndex
            , // ResourceSource
            CFGR // DescriptorName
        )

        // Descriptor for the root node. This is a 16KB region at
        // offset ROOTNODEBASE. In this example, ROOTNODEBASE starts
        // at the 16KB aligned offset of PEIPHBASE + 0xC000
        QWordMemory (
            ResourceConsumer, // bit 0 of general flags is 0
            PosDecode,

```

```

        MinFixed,          // Range is fixed
        MaxFixed,          // Range is Fixed
        NonCacheable,
        ReadWrite,
        0x00000000,        // Granularity
        0xAFE000C000,      // Min, set to ROOTNODEBASE
        0xAFE000FFFF,      // Max
        0x00000000,        // Translation
        0x0000004000,      // Range Length = 16KB
        ,                  // ResourceSourceIndex
        ,                  // ResourceSource
        ROOT               // DescriptorName
    )

    // Interrupt on PMU0 overflow, attached to DTC[0], with GSIV = <
    ↪gsiv0>
    Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
    ↪gsiv0>}

    // Interrupt on PMU1 overflow, attached to DTC[1], with GSIV = <
    ↪gsiv1>
    Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
    ↪gsiv1>}

    })
}

```

The following ASL code example shows a CMN-700 based network that has a dimension less than 4x4 and that has four DTCs, DTC0-DTC3. The logical numbers of these DTCs are DTC[0] to DTC[3].

```

Device(CMN7) { // CMN-700 device object for an X * Y mesh where X, Y <= 4
    Name(_HID, "ARMHC700")
    Name(_CRS, ResourceTemplate () {
        // Descriptor for 256 MB of the CFG region at offset PERIPHBASE
        QWordMemory (
            ResourceConsumer, // bit 0 of general flags is 0
            PosDecode,
            MinFixed,          // Range is fixed
            MaxFixed,          // Range is Fixed
            NonCacheable,
            ReadWrite,
            0x00000000,        // Granularity
            0xAFE0000000,      // Min
            0xAFE0FFFFFFF,     // Max
            0x00000000,        // Translation
            0x0010000000,      // Range Length
            ,                  // ResourceSourceIndex
            ,                  // ResourceSource
            CFGR               // DescriptorName
        )

        // Interrupt on PMU0 overflow, attached to DTC[0], with GSIV = <
        ↪gsiv0>
        Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
        ↪gsiv0>}

        // Interrupt on PMU1 overflow, attached to DTC[1], with GSIV = <
        ↪gsiv1>
        Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
        ↪gsiv1>}
    })
}

```

```

// Interrupt on PMU2 overflow, attached to DTC[2], with GSIV = <
↪gsiv2>
Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
↪gsiv2>}

// Interrupt on PMU3 overflow, attached to DTC[3], with GSIV = <
↪gsiv3>
Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
↪gsiv3>}

})
}

```

### 2.6.3.3 RAS Description for CMN

The CMN products support RAS as defined by [21]. CMN devices support a set of error nodes, which must be described in the ACPI AEST table to support OS discovery, and kernel-first handling of errors occurring in these devices.

Devices that support RAS include HN-I, HN-F, CXHA/CCG, SBSX and XP. Each instance of these devices has a dedicated register block at a fixed offset from PERIPHBASE. The error node for that device is at a specific offset within the register block. The driver, handling RAS errors, must have knowledge of the offset of the error node within the register block. Note that the offset of the error nodes within a device register block varies per device and per CMN product revision.

Note: the PERIPHBASE is encoded in the `_CRS` object of the CMN ACPI DSDT device.

RAS interrupts from devices within the CMN are routed to a specific HN-D. This HN-D also holds the group-level error registers for the devices. A device can be wired to only one HN-D.

The error nodes must be described in the AEST as AEST nodes [22]. As an example, for a CMN implementation with  $M$  HN-Is and  $N$  XPs, there will be  $M+N$  AEST nodes. The AEST nodes are of type vendor-defined, with `_HID` set to the corresponding value defined in Table 17.

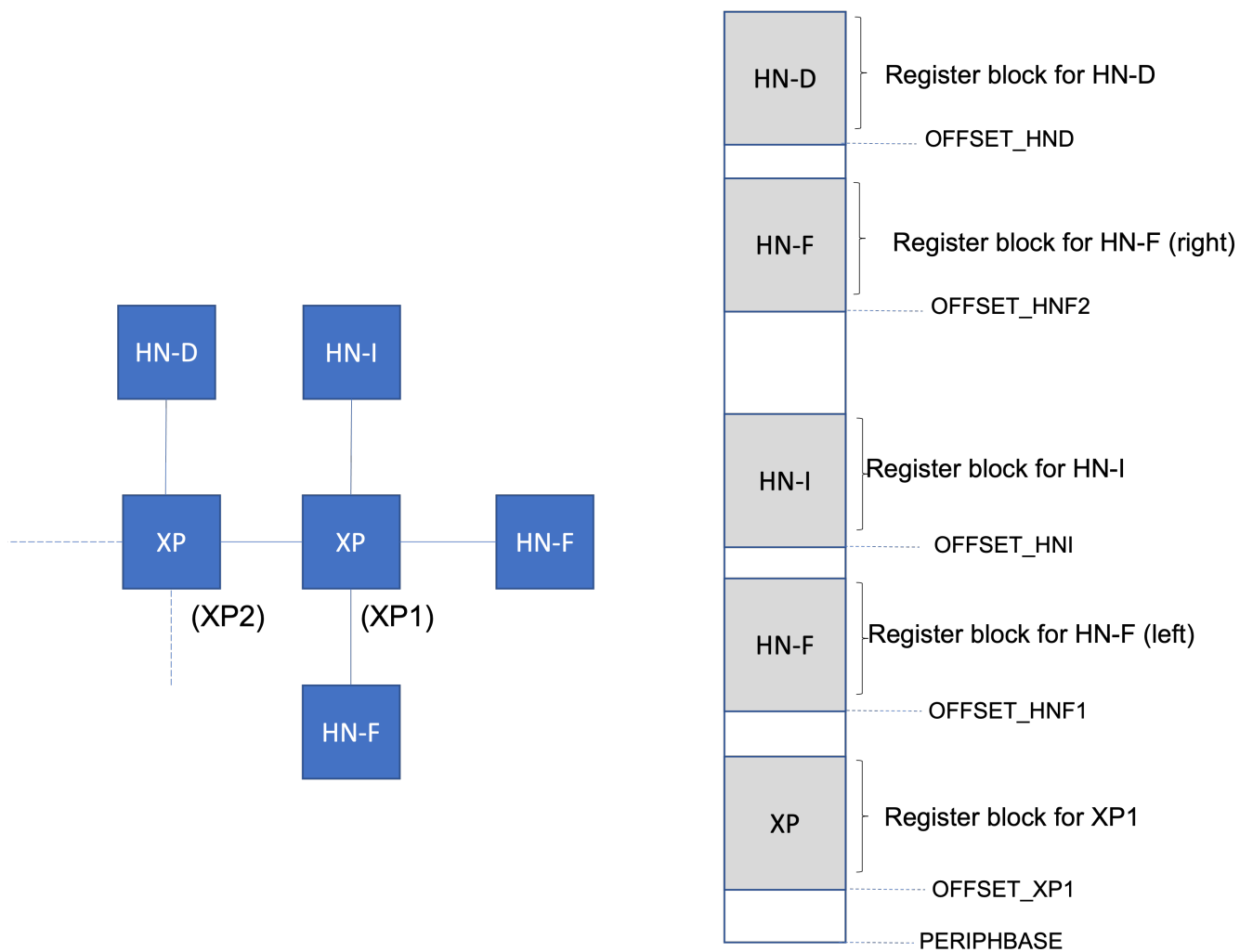
**Table 17: CMN product error node `_HID` list**

Value	Description
ARMHC601	ACPI Hardware Identifier for the CMN-600 error node
ARMHC651	ACPI Hardware Identifier for the CMN-650 error node
ARMHC701	ACPI Hardware Identifier for the CMN-700 error node
ARMHC103	ACPI Hardware Identifier for the CMN-S3 error node

The `_UID` is set to the CMN instance. The vendor-defined data field must be set as follows:

Vendor-defined data	Description
Bits[127:64]	Offset from PERIPHBASE where the register block for this device is present. Note: For the CMN-S3 product this information is redundant, and should be disregarded by the OS.
Bits[63:0]	Offset from PERIPHBASE where the register block for the HN-D is present.

An example CMN implementation is illustrated in Figure 4.



**Figure 4: Example CMN implementation with RAS-aware devices**

The ACPI AEST definition of this implementation is as illustrated in Figure 5.



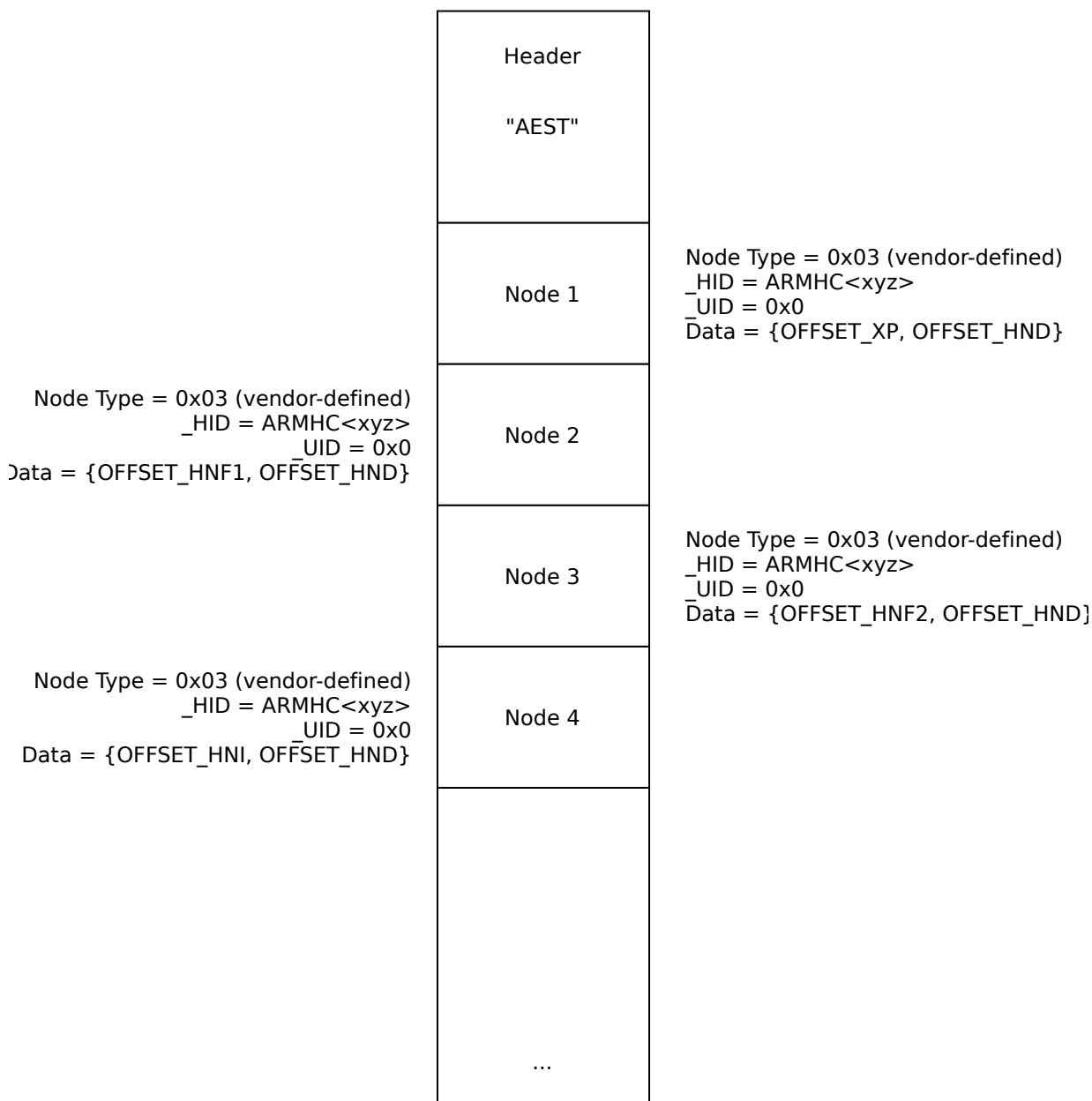


Figure 5: ACPI AEST Table representation for the example CMN implementation

## 2.6.4 Arm CoreLink Network-on-chip Interconnect Family

The Arm CoreLink NI Coherent Network-on-chip Interconnect family is described in [23]. Configuration registers of the NI components are mapped into a dedicated memory-mapped address range at offset PERIPHBASE. The PMU registers are then available at an offset from PERIPHBASE in a 4K register block called a node. Overflow interrupts from each of the eight counters are wired together to a summary overflow interrupt.

### 2.6.4.1 Interface Identification

Notation: The HID is composed of the following sub-identifier fields:

- The first four characters, "ARMH", indicate that this is an Arm-specific hardware.

- The fifth character is set to 'C' to indicate that this is a product from the CoreLink Network-on-chip interconnect family.
- The sixth character is set to 'B' to indicate that this is a product from the Arm CoreLink Network Interconnect family.
- The last two characters are set to the first two numbers of the product name.

**Table 19: Arm NI HID values**

Value	Description
ARMHCB70	ACPI Hardware Identifier for the NI-700 PMU

**Device configuration objects****Table 20: Arm NI configuration objects**

Object	Values	Type	Description
_CRS	PERIPHBASE	QWordMemory	Base address of the memory-mapped region in the system address map where the NI registers are mapped
	GSIV[n]	Interrupt	List of GSIVs for the overflow interrupts of the PMU instances in the NI. GSIV[i] is associated with the <i>i</i> th PMU block.

**ASL reference code example**

The following ASL code example shows an NI-700 interconnect being described in DSDT. The NI-700 registers are mapped to PERIPHBASE of 0x2000000000 in this example.

```

Device(CNIO) { // NI-700 device object
    Name(_HID, "ARMHCB70")
    Name(_CRS, ResourceTemplate () {
        // Descriptor for PERIPHBASE
        QWordMemory (
            ResourceConsumer, // bit 0 of general flags is 0
            PosDecode,
            MinFixed, // Range is fixed
            MaxFixed, // Range is Fixed
            NonCacheable,
            ReadWrite,
            0x00000000, // Granularity
            0x2000000000, // Min
            0x2000FFFFFFFF, // Max
            0x00000000, // Translation
            0x0010000000, // Range Length
            , // ResourceSourceIndex
            , // ResourceSource
            PERB // DescriptorName
        )

        // Interrupt on PMU 0 overflow, GSIV = <gsiv0>
        Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
            ↪gsiv0>}
    }
}

```

```
// ..  
  
// Interrupt on PMU 31 overflow, GSIV = <gsiv31>  
Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<  
    ↪gsiv31>}  
    })  
}
```

---