

# IO Remapping Table

## System Software on ARM<sup>®</sup> Platforms

Document number: ARM DEN 0049B

Copyright ARM Limited or its affiliates 2015



## IO Remapping Table System Software on ARM

Copyright © 2015 ARM Limited or its affiliates. All rights reserved.

### Release information

The Change History table lists the changes made to this document.

**Table 1 Change history**

Date	Issue	Confidentiality	Change
17 April 2015	A	Non-Confidential	First release.
5 October 2015	B	Non-Confidential	Added SMMUv3

### Non-Confidential Proprietary Notice

This document is protected by copyright and the practice or implementation of the information herein may be protected by one or more patents or pending applications. No part of this document may be reproduced in any form by any means without the express prior written permission of ARM. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document.**

This document is **Non-Confidential** but any disclosure by you is subject to you providing the recipient the conditions set out in this notice and procuring the acceptance by the recipient of the conditions set out in this notice.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any patents.

This document is provided “as is”. ARM makes no representations or warranties, either express or implied, included but not limited to, warranties of merchantability, fitness for a particular purpose, or non-infringement, that the content of this document is suitable for any particular purpose or that any practice or implementation of the contents of the document will not infringe any third party patents, copyrights, trade secrets, or other rights. Further, ARM makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of such third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT LOSS, LOST REVENUE, LOST PROFITS OR DATA, SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO ANY FURNISHING, PRACTICING, MODIFYING OR ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Words and logos marked with ® or TM are registered trademarks or trademarks, respectively, of ARM Limited. Other brands and names mentioned herein may be the trademarks of their respective owners. Unless otherwise stated in the terms of the Agreement, you will not use or permit others to use any trademark of ARM Limited.

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

In this document, where the term ARM is used to refer to the company it means “ARM or any of its subsidiaries as appropriate”.

Copyright © 2015, ARM Limited or its affiliates. All rights reserved.

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

# Contents

<b>1</b>	<b>ABOUT THIS DOCUMENT</b>	<b>4</b>
1.1	References	4
1.2	Terms and abbreviations	4
1.3	Feedback	4
1.3.1	Feedback on this manual	4
<b>2</b>	<b>INTRODUCTION</b>	<b>6</b>
<b>3</b>	<b>IO REMAPPING TABLE</b>	<b>7</b>
3.1.1	IORT node types	9
	<b>APPENDIX A RATIONALE AND EXAMPLES</b>	<b>18</b>
	<b>APPENDIX B OS USAGE OF MEMORY ATTRIBUTES</b>	<b>24</b>

# 1 About this Document

This document provides a proposal for an ACPI representation of IO topology to be used by ARM-based systems.

## 1.1 References

This document refers to the following documents.

Reference	Document Number	Title
[SBSA]	ARM DEN 0029	<i>Server Base System Architecture</i>
[ACPI6.0]	ACPI v6.0	<i>Advanced Configuration and Power Interface Specification</i>
[GIC]	ARM IHI 0069	<i>ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0</i>
[SMMUv2]	ARM IHI 0062	<i>ARM® System Memory Management Unit Architecture Specification, version 2.0</i>
[PCIFW]		<i>PCI Firmware Specification, Revision 3.1</i>

## 1.2 Terms and abbreviations

This document uses the following terms and abbreviations.

Term	Meaning
RID	Requestor ID for a PCI express device.
BDF	Bus Device Function. Equivalent to a RID.
DeviceID	Identifier for a device exposed to the GICv3/4 Interrupt Translation Service. See [GIC] for more details.
ITS	GIC Interrupt Translation Service. See [GIC] for more details.
StreamID	A StreamID uniquely identifies to an SMMU a stream of transactions that can originate from one or more devices but are associated with the same context. See [SMMUv2] for more details.
SBSA	Server Base System Architecture.
IO Coherent	A device is IO Coherent with the processor caches if its transactions snoop the processor caches for cacheable regions of memory. The processor does not snoop the device cache.

## 1.3 Feedback

ARM welcomes feedback on its documentation.

### 1.3.1 Feedback on this manual

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.
- The document and version number, ARM DEN 0049B.

- The page numbers to which your comments apply.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

## 2 Introduction

This document describes the *Input Output Remapping Table* (IORT), which represents the IO topology of an ARM-based system for use with the *Advanced Configuration and Power Interface* (ACPI). The IORT describes how various components are connected together, and how those components that need identification reserve values in the appropriate identification space.

In particular, the IORT:

- Provides an ACPI description for IO topology, SMMUs, and GIC ITSs.
- Identifies which components are behind which SMMU.
- Identifies which components are behind an ITS or group of ITSs.
- Describes the IO relationships of PCIe root complexes and relates this description to the MCFG table [PCIFW] and the ACPI namespace.
- Describes the IO relationships between devices represented in the ACPI namespace.
- Represents the following ID mapping relationships:
  - From BDF requestor ID, for a PCIe device, to a StreamID for an SMMU, and then to a DeviceID for an ITS.
  - From BDF requestor ID to a DeviceID, for a device that is not connected to SMMU but which can generate MSIs.
  - Individual endpoint StreamIDs.
  - Individual endpoint DeviceIDs.
- Does not support arbitrarily complex ID mappings. Only simple offset-based mappings are supported.

The next section describes the IORT in detail. Appendix A explains how the IORT is used to describe a system and provides examples. Finally, Appendix B describes how the memory attributes that are described for a device can be used to ascertain when cache management is required.

### 3 IO Remapping Table

Table 2 shows the structure of the IORT. Apart from the basic header, the table contains a number of IORT Nodes. Each node represents a component, which can be an SMMU, an ITS Group, a root complex, or a component that is described in the namespace.

**Table 2 The IORT**

Field	Byte Length	Byte Offset	Description
Header			Standard ACPI format for header. See section 5.2 of [ACPI6.0] for further details.
Signature	4	0	'IORT'. IO Remapping Table.
Length	4	4	Length, in bytes, of the entire IORT.
Revision	1	8	0.
Checksum	1	9	The entire table must sum to zero.
OEMID	6	10	OEM ID.
OEM Table ID	8	16	For the IORT, the table ID is the manufacture model ID.
OEM Revision	4	24	OEM revision of the IORT for the supplied OEM Table ID.
Creator ID	4	28	The vendor ID of the utility that created the table. For tables containing Definition Blocks, this is the ID for the ASL Compiler.
Creator Revision	4	32	The revision of the utility that created the table. For tables containing Definition Blocks, this is the revision for the ASL Compiler.
Body			
Number of IORT Nodes	4	36	The number of nodes in the IORT Node Array.
Offset to Array of IORT Nodes	4	40	The offset from the start of the table to the first node in the array of IORT nodes.
Reserved	4	44	Reserved, must be zero.
Optional padding	---	--	
Array of IORT Nodes	---	--	Array of IORT Nodes.

Nodes describe component *identifiers* (IDs) and the mapping between the input space of those IDs and the output space. For example, a root complex node, sitting behind an SMMU, describes how the input RID space of the root complex maps onto the output StreamID space that is directed to the SMMU. Not every node needs IDs. Some nodes, such as those that represent ITS groups, only consume IDs. ITSs consume DeviceIDs. Table 3 shows the format of IORT Nodes.

**Table 3 Node Format**

Field	Byte Length	Byte Offset	Description
Generic data:			
Type	1	0	Possible values and their meanings are: <ul style="list-style-type: none"> <li>• 0: ITS Group.</li> <li>• 1: Named component.</li> <li>• 2: Root complex.</li> <li>• 3: SMMUv1 or SMMUv2.</li> <li>• 4: SMMUv3.</li> <li>• 5-255: Reserved.</li> </ul>
Length	2	1	Length of node in bytes.
Revision	1	3	Revision of the IORT node.
Reserved	4	4	Must be zero.
Number of ID mappings	4	8	Number of ID entries in the ID Array.
Reference to ID Array	4	12	Offset from the start of the IORT node to the start of its Array of ID mappings. This field has a value of 0 in the case of an ITS that has no IDs.
Data that varies for node type:			
Data specific to a Node	X	16	
ID Section:			
Array of ID mappings	20xN	16+X	ID mapping Array, where N is the number of ID mappings.

ID mappings represent the formula by which an ID from a source is converted to an ID in a destination. For example, for a root complex behind an SMMU, the RID originating from that root complex must be converted to a StreamID in the destination SMMU. With IORT, ID mappings are declared in the source node. Each mapping describes the destination of the IDs, also known as the output, as well as the numerical relationship that must hold between input IDs and output IDs. Each entry in the array of ID mappings has the following format:

**Table 4 ID mapping format**

Field	Byte Length	Byte Offset	Description
Input base	4	0	The lowest value in the input range.
Number of IDs	4	4	The number of IDs in the range minus one.
Output base	4	8	The lowest Value in output range.
Output Reference	4	12	A reference to the output IORT Node. This field contains the address offset of the IORT Node relative to the start of the IORT. For example, if this ID mapping is for a root complex outputting to an SMMU, the value of this field is the difference between the start of the SMMU IORT node and the start of the IORT.
Flags	4	16	See Table 5.



Table 5 shows the format of the ID flags.

**Table 5 ID flags format**

Field	Bit Length	Bit Offset	Description
Single mapping	1	0	Single mapping. Apply the output base regardless of the input IDs. This flag is only valid when the mapping is contained inside a named component or root complex node.
Reserved	31	1	Reserved, must be zero.

The following sections describe IORT nodes for SMMUs, ITS groups, named components, and root complexes.

### 3.1.1 IORT node types

The following sections describe each type of IORT node.

#### 3.1.1.1 SMMUv1 or SMMUv2 node

This section describes the format of the IORT node for SMMUv1 or SMMUv2.

**Table 6 Node format for SMMUv1 or SMMUv2**

Field	Byte Length	Byte Offset	Description
Type	1	0	This field has a value of 3 for SMMUv1 or SMMUv2.
Length	2	1	The length of the node.
Revision	1	3	0.
Reserved	4	4	Must be zero.
Number of ID mappings	4	8	The number of ID mappings.
Reference to ID Array	4	12	Offset from the start of the IORT node to the start of its Array of ID mappings.
SMMUv1/2 specific data.			
Base address	8	16	The SMMU base address.
Span	8	24	The length of the memory range that is covered by SMMU memory-mapped IO.
Model	4	32	Possible values are: <ul style="list-style-type: none"> <li>• 0: Generic SMMUv1.</li> <li>• 1: Generic SMMUv2.</li> <li>• 2: ARM® Corelink™ MMU-400.</li> <li>• 3: ARM® Corelink™ MMU-500.</li> <li>• All other values are reserved.</li> </ul>
Flags	4	36	The SMMU flags. See Table 7.

Reference to Global Interrupt Array	4	40	The offset from the start of this IORT node to the start of its global interrupt array section.
Number of context interrupts	4	44	The number of context interrupts.
Reference to Context Interrupt Array	4	48	The offset from the start of this IORT node to the start of its context interrupt array section.
Number of PMU Interrupts	4	52	The number of PMU Interrupts.
Reference to PMU Interrupt Array	4	56	The offset from the start of this IORT node to the start of its PMU interrupt array section.
Global Interrupt Array section			
SMMU_NSglrpt	4	60	The GSIV of the SMMU_NSglrpt interrupt.
SMMU_NSglrpt interrupt	4	64	The SMMU_NSglrpt interrupt flags. See Table 8.
SMMU_NSgCfgrpt	4	68	The GSIV of the SMMU_NSgCfgrpt interrupt. This field has a value of 0 if not implemented.
SMMU_NSgCfgrptinterrupt	4	72	The SMMU_NSgCfgrpt interrupt flags. See Table 8.
Context Interrupts Array section			
Context Interrupts Array	8xN	76	<p>Each interrupt is described by two 4-byte fields:</p> <ul style="list-style-type: none"> <li>Bytes 0:3: GSIV of interrupt.</li> <li>Bytes 4:7: Interrupt flags as described in Table 8.</li> </ul> <p>For SMMUv2 implementations, there must be exactly one interrupt per context bank. In the case of a single, combined interrupt, it must be listed multiple times.</p> <p>Interrupts are indexed by a context bank number.</p>
PMU Interrupt Array section			
PMU Interrupt Array	8xN	--	<p>Each interrupt is described by two 4-byte fields:</p> <ul style="list-style-type: none"> <li>Bytes 0:3 : GSIV of interrupt.</li> <li>Bytes 4:7: Interrupt flags as described in Table 8.</li> </ul> <p>Interrupts must be ordered by PMU group. That is, for every implemented PMU group an Interrupt entry must be provided. The order of the entries reflects the order of the PMU groups.</p>
IDs for SMMUv1/2 section			
Array of ID mappings	20xN	--	ID Array. N is the Number of ID mappings.

Table 7 describes the SMMUv1 and SMMUv2 table flags.

**Table 7 SMMUv1 and SMMUv2 flags**

Field	Bit Length	Bit Offset	Description
DVM Supported	1	0	1: The SMMU supports <i>Distributed Virtual Memory</i> (DVM) messages and therefore supports broadcast TLB maintenance operations. 0: The SMMU does not support DVM.
Coherent Page Table Walk	1	1	1: The page table walk done by the SMMU is coherent with CPU caches. 0: The page table walk done by the SMMU is not coherent with CPU caches.
Reserved	30	2	Reserved, must be zero.

The SMMU table interrupts have the following flags:

**Table 8 Interrupt flags**

Field	Bit offset	Number of bits	Description
Interrupt Flags	0	1	1: The interrupt is edge-triggered. 0: The interrupt is level-triggered.
Reserved	1	31	Must be zero.

ID mappings that are defined in an SMMU IORT node can only have an ITS Group node as an output reference, or no IDs in the case of a system that does not have ITS units. All other object types are explicitly forbidden. By implication, ID mappings of an SMMU cannot have an SMMU as an output. In other words, nesting of SMMUs is not allowed.

### 3.1.1.2 SMMUv3

Table 9 shows the format of an SMMUv3 IORT node.

**Table 9 SMMUv3 Format**

Field	Byte Length	Byte Offset	Description
Type	1	0	SMMUv3
Length	2	1	Length of node
Revision	1	3	0
Reserved	4	4	Must be zero
Number of ID mappings	4	8	
Reference to ID Array	4	12	Offset from the start of an IORT node to the start of the following ID array section

SMMUv3 specific data			
Base address	8	16	Base address of SMMU
Flags	4	24	See Table 10
Reserved	4	28	
VATOS address	8	32	Optional, 0 if not supported
Model	4	40	0: Generic SMMU-v3
Event	4	44	GSIV of the Event interrupt if SPI based, 0 if not
PRI	4	48	GSIV of the PRI interrupt if SPI based, 0 if not
GERR	4	52	GSIV of the GERR interrupt if GSIV based, 0 if not
Sync	4	56	GSIV of the Sync interrupt if GSIV based, 0 if not
IDs for SMMUv3			
Array of ID mappings	20xN	60	ID Array

ID mappings of an SMMU cannot have an SMMU as an output. In other words, nesting of SMMUs is not allowed. Table 10 shows the SMMUv3 flags. When using wired interrupts, the SMMU architecture requires them to be edge sensitive, and therefore no interrupts flags are described in the SMMUv3 IORT node.

**Table 10 SMMUv3 flags**

Field	Bit offset	Number of bits	Description
COHACC Override	0	1	Overrides the value in SMMU_IDR0.COHACC
HTTU Override	1	2	Overrides the value in SMMU_IDR0.HTTU
Reserved	3	29	Must be zero

### 3.1.1.3 ITS group node

ITS group nodes describe which ITS units are in the system. A node allows grouping of more than one ITS, but all ITSs in the group must share a common understanding of DeviceID values. That is, a given DeviceID must represent the same device for all ITS units in the group.

ITS group nodes have no ID mappings. Table 11 shows the format of ITS groups.

**Table 11 ITS Group Format**

Field	Byte Length	Byte Offset	Description
Type	1	0	This field has a value of 0.
Length	2	1	The length of the node in bytes.
Revision	1	3	0.
Reserved	4	4	Must be zero.
Number of ID mappings	4	8	This field has a value of 0. ITS groups do not have IDs.

Reference to ID Array	4	12	This field has a value of 0. There is no ID array.
ITS-specific data			
Number of ITSs	4	16	The number of ITSs.
GIC ITS Identifier Array	4xN	20	The array of ITS identifiers. These IDs must match the value used in the <i>Multiple APIC Description Table</i> (MADT) GIC ITS structure for each relevant ITS unit. See [ACPI6.0].

### 3.1.1.4 Named component node

Named component nodes are used to describe devices that are also included in the *Differentiated System Description Table* (DSDT). See [ACPI6.0].

These nodes can have one or more ID mappings, and the mappings can use SMMUs or ITS groups as output references. All other output node types are forbidden as output references.

For devices described in the DSDT, Table 12 describes the IORT node format.

**Table 12 Named component node format**

Field	Byte Length	Byte Offset	Description
Type	1	0	For a named component, this field has a value of 1.
Length	2	1	The length of the node.
Revision	1	3	IORT revision number, currently 0.
Reserved	4	4	Must be zero.
Number of ID mappings	4	8	The number of ID mappings.
Reference to ID Array	4	12	Offset from the start of the IORT node to the start of its Array of ID mappings.
Named component-specific data:			
Node flags	4	16	Reserved, must be zero.
Memory access properties	8	20	These properties are described in Table 13.
Device memory address size limit	1	28	The number of address bits, starting from the least significant bit that can be generated by a device when it accesses memory.
Device object name	--	29	The ASCII Null terminated string with the full path to the entry in the namespace for this object.
Padding	--	--	Padding is to 32-bit word-aligned.
IDs for Named component:			
Array of ID mappings	20xN	--	ID Array. N is the Number of ID mappings.

Table 13 describes device node memory access properties.

Table 13 Memory access properties

Field	Byte Length	Byte Offset	Description
CCA: Cache Coherent Attribute	4	0	<p>This value must match the value returned by the _CCA object that is defined in the DSDT for the device represented by this node. The attribute can take the following values:</p> <p>0x1: The device is fully coherent. No cache maintenance* is required for memory that is shared with the device that is mapped on CPUs as <i>Inner Write-Back (IWB)</i>, <i>Outer Write-back (OWB)</i>, and <i>Inner shareable (ISH)</i>. In addition, during system initialization at cold boot, or after wakeup from low-power state, if the cache coherency requires an SMMU override or some specific device configuration, the platform firmware has to ensure that this has been done. Therefore the semantics represented by a value of 0x1 are always correct at the time of hand-off from firmware to OS.</p> <p>0x0: The device is not coherent. Therefore:</p> <ul style="list-style-type: none"> <li>Cache maintenance is required for memory that is shared with the device that is mapped on CPUs as IWB-OWB-ISH.</li> <li>No cache maintenance is required for memory that is shared with the device that is mapped on CPUs as device or Non-cacheable.</li> </ul> <p>All other values are reserved.</p>
AH: Allocation Hints	1	4	<p>This field can be ignored without loss of correctness.</p> <p>Allocation hints have the following format:</p> <ul style="list-style-type: none"> <li>Bits[7:4] are ignored by the OS and must be zero.</li> <li>Bit 3: Allocation Hints Override (AHO). <ul style="list-style-type: none"> <li>0: If the bit is clear, use the incoming Read Allocate (RA), Write Allocate (WA), and Transient (TR) hints.</li> <li>1: If the bit is set, override allocation hints based on the values in bits RA, WA, and TR.</li> </ul> </li> <li>Bit 2: Read Allocate (RA). <ul style="list-style-type: none"> <li>0: Clear read allocation hint if AHO is set to 1.</li> <li>1: Set read allocation hint if AHO is set to 1.</li> </ul> </li> <li>Bit 1: Write Allocate (WA). <ul style="list-style-type: none"> <li>0: Clear write allocation hint if AHO is set to 1.</li> <li>1: Set write allocation hint if AHO is set to 1.</li> </ul> </li> <li>Bit 0: Transient (TR). <ul style="list-style-type: none"> <li>0: Clear transient hint if AHO is set to 1.</li> <li>1: Set transient hint if AHO is set to 1.</li> </ul> </li> </ul>

Reserved	2	5	Reserved, must be zero.
MAF: Memory Access Flags	1	7	See Table 14.

\* Note: Caching operations described in this document apply to the CPU caches and any other caches in the system where device memory accesses can hit.

Table 14 describes the memory access properties flags field format.

**Table 14 Memory Access Flags**

Field	Bit Length	Bit Offset	Description
CPM: Coherent Path to Memory	1	0	<p>0x1: If set, this indicates that the device has path to memory that allows coherency with the CPU cache hierarchy. This means that if the CPU maps the memory as IWB, OWB, ISH, and the device is outputting the same attributes, or being overridden through an SMMU to provide the same attributes, no cache maintenance is required.</p> <p>0x0: The device does not have a path to memory that is coherent with the CPU cache hierarchy. Therefore:</p> <ul style="list-style-type: none"> <li>Cache maintenance is required for memory that is shared with the device that is mapped on the CPU as IWB-OWB-ISH.</li> <li>No cache maintenance is required for memory that is shared with the device that is mapped on the CPU as device or Non-cacheable.</li> </ul> <p>Note that if CCA is 0x1, CPM must also be 0x1. Conversely, If CPM is 0x0 then CCA must be 0x0. However, it possible for the system to boot with CCA set to 0x0 and CPM set to 0x1. See Table 15 and 0for further details.</p>
DACS: Device attributes are Cacheable and Inner-Shareable	1	1	<p>The device outputs IWB-OWB-ISH attributes:</p> <p>0x1: The device outputs IWB-OWB-ISH attributes.</p> <p>0x0: The device does not output IWB-OWB-ISH attributes.</p>
Reserved	6	2	Reserved must be zero.

Not every combination of memory attribute values is valid or useful. Table 15 lists the valid combinations and their uses:

**Table 15 Valid Memory Attributes**

CCA	CPM	DACS	AH	Comment
1	1	1	Can be applied	The device outputs the correct attributes that enable cache coherency (IWB-OWB-ISH). If the shared memory is mapped on the CPU with the same attributes, no cache management is necessary, and the device

				<p>exploits cache coherency.</p> <p>If the device is behind an SMMU, the OS might override the allocation hint attributes using the values that are supplied in the AH. However, in this case the OS must maintain the coherency guarantee indicated by the CCA value of 1. Therefore, it must not change the cacheability and shareability attributes provided by the device (IWB-OWB-ISH).</p>
1	1	0	Can be applied	<p>The device does not natively provide IWB-OWB-ISH attributes, and cache coherency is provided by an override through an SMMU.</p> <p>The value of 1 for CCA shows that boot firmware has configured the SMMU to ensure cache coherency.</p> <p>The OS might apply the allocation hints supplied in AH when overriding the device attributes.</p> <p>A device with this combination of flag values must be behind an SMMU. Therefore, it must have an ID in its array of ID mappings that has an SMMU IORT node as its output reference.</p>
1	0	-	N/A	Illegal. If CPM is 0, CCA cannot be 1.
0	1	0	Can be applied	<p>The device does not natively provide IWB-OWB-ISH attributes, but cache coherency can be provided by an override through an SMMU.</p> <p>The OS can provide cache coherency by using an SMMU to override the device attributes to IWB-OWB-ISH. In this case, the OS might apply the allocation hints supplied in AH when overriding the device attributes.</p> <p>A device with this combination of flag values must be behind an SMMU. Therefore, it must have an ID in its array of ID mappings that has an SMMU IORT node as its output reference.</p>
0	1	1	Can be applied	Illegal. If the device has a coherent path to memory, and natively outputs IWB-OWB-ISH attributes then CCA must be set to 1.
0	0	*	N/A	<p>The device is not coherent and cannot be made coherent.</p> <p>If the CPU maps memory that is shared with the device as IWB-OWB-ISH, cache maintenance is required. If the CPU maps the memory as Non-cacheable, then no cache maintenance is required.</p>

Appendix B describes how an OS can use this information and when caching operations are required.

### 3.1.1.5 PCI root complex node



The root complex node is used to describe PCI root complexes. The format is described in Table 16. In addition, the following rules and assumptions apply:

- ID mappings can use SMMUs or ITS groups as output references.
- It is assumed that PCI segment numbers have a one-to-one mapping with root complexes. Each segment number can represent only one root complex.

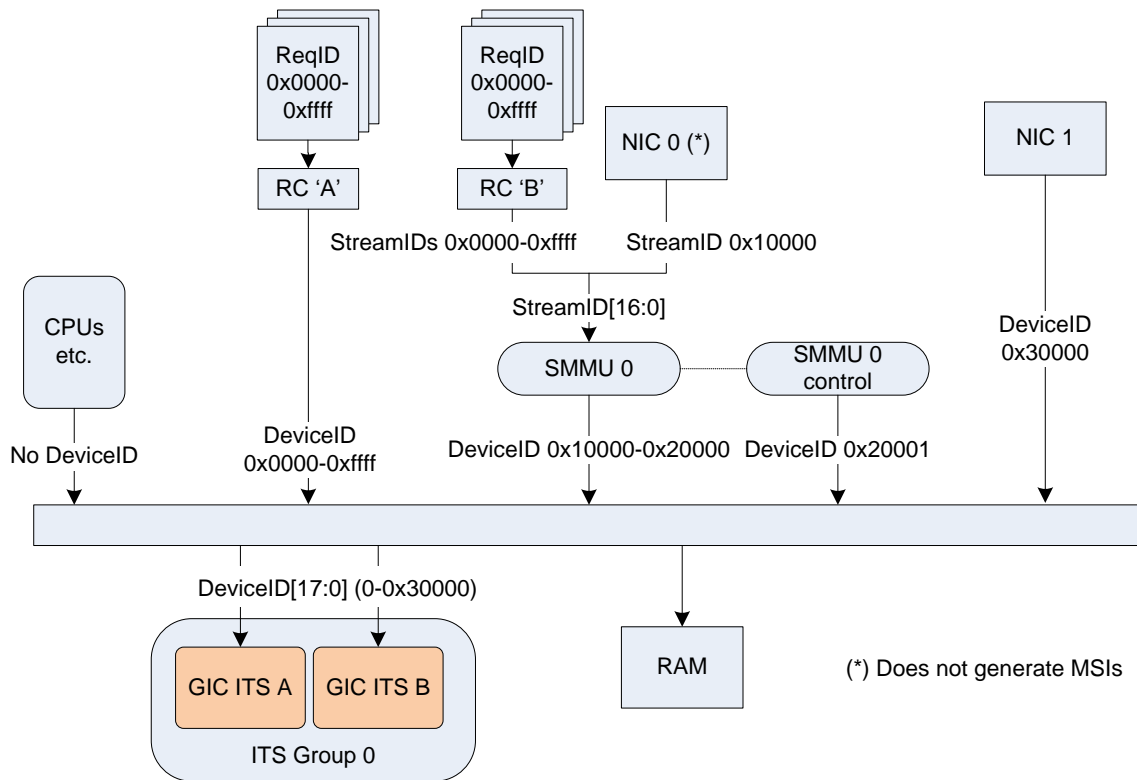
**Table 16 Root Complex Node**

Field	Byte Length	Byte Offset	Description
Type	1	0	For the root complex type, this field has a value of 2.
Length	2	1	The length of the node.
Revision	1	3	IORT revision number, currently 0.
Reserved	4	4	Must be zero.
Number of ID mappings	4	8	The number of ID mappings.
Reference to ID Array	4	12	Offset from the start of the IORT node to the start of its Array of ID mappings.
Root complex specific data.			
Memory access properties	8	16	These properties are described in Table 13. Note that SBSA [SBSA] requires root complexes to support IO coherency and to be in the same shareability domain as the CPUs.
ATS Attribute	4	24	0x1: The root complex supports ATS. 0x0: The root complex does not support ATS.
PCI Segment number	4	28	The PCI segment number, as in MCFG and as returned by _SEG in the namespace.
IDs for root complex			
Array of ID mappings	20xN	32	Array of IDs. N is the Number of ID mappings.

It is expected that all accesses from PCIe devices behind the root complex are cache coherent and in the same inner shareability domain as the CPUs controlled by the OS that is parsing these tables.

## Appendix A Rationale and Examples

Figure 1 depicts an example of a system with a relatively complex topology.



**Figure 1 Example system**

Figure 1 depicts a set of components that can communicate with a group of GIC ITs and devices behind an SMMU. Figure 2 provides a different view of the example system. It depicts the IDs for each component across the various ID spaces, RID, StreamID, and DeviceID

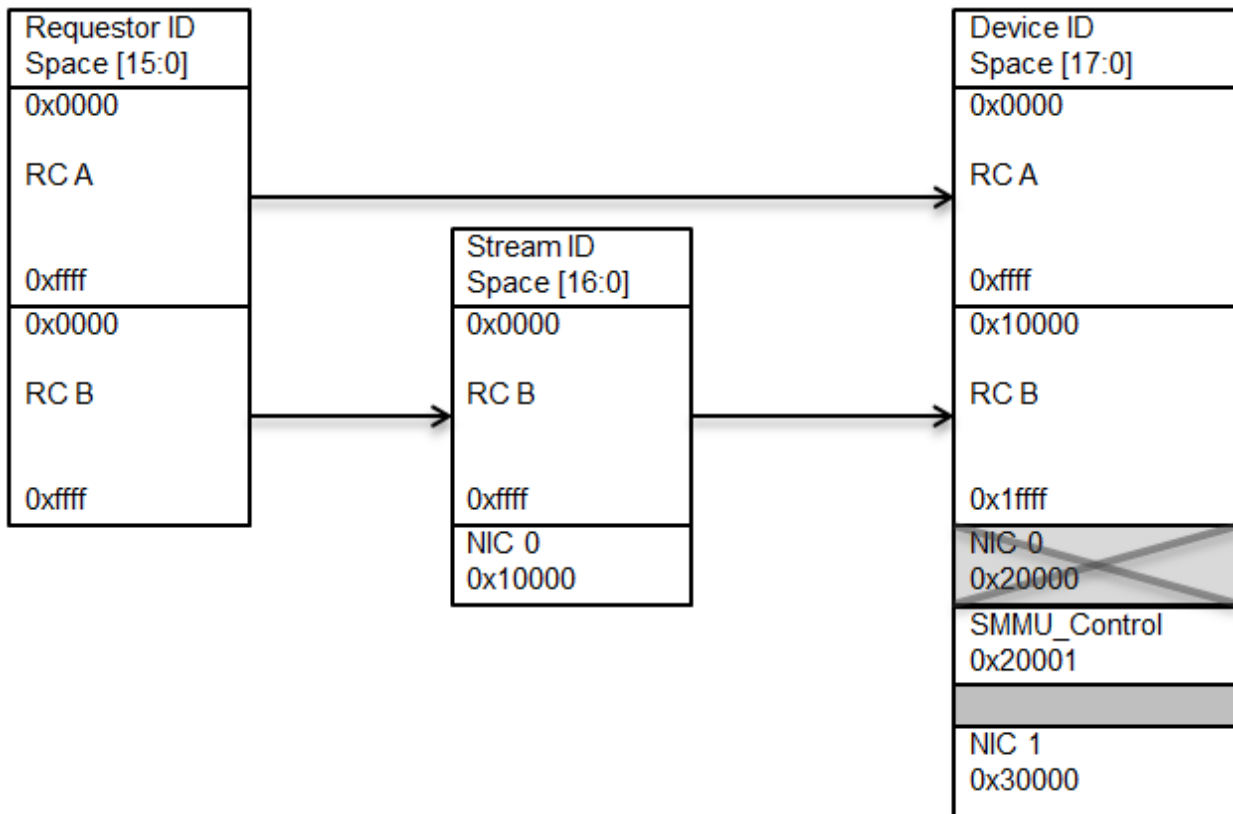


Figure 2 ID mappings

The first obvious feature of the system is that there are two overall types of ID that can be described for a given component:

- Some component, such as root complexes, require a range of IDs.
- Some components, for example NIC1 or NIC0, require only a single ID.

Two kinds of ID definitions emerge, a range and an endpoint.

Figure 2 also shows how IDs in one space map to another. For example, the RID space of RC B, which spans 0x0->0xffff, maps to StreamID space 0x0000->0xffff in SMMU 0, which in turn maps to DeviceID space 0x10000->0x1ffff in the ITS group. This can be viewed as an input ID to output ID mappings, for example RC B has an input ID in the RID space and an output ID in the StreamID space of SMMU 0. However, this rule does not apply to every device, because some devices, such as the endpoints NIC 0 and NIC 1 in the example, only have an output ID. Finally, not every device is capable of generating MSIs, so some devices, such as NIC 0 in the example system, can only have a StreamID mapping. The mappings in Figure 2 can be described by the following pseudocode:

```
RC A
    // doesn't use SMMU 0 so just outputs DeviceIDs to ITS GROUP 0
    // Input ID    --> Output reference: Output ID
    0x0000-0xffff --> ITS GROUP 0 : 0x0000->0xffff

RC B
    // Input ID    --> Output reference: Output ID
    0x0000-0xffff --> SMMU 0      : 0x0000->0xffff

NIC 0
    // endpoint device doesn't have an explicit input ID
```

```

// In this example the NIC 0 does not generate MSI so
// has no DeviceID
// Input ID --> Output reference: Output ID
N/A --> SMMU 0 : 0x10000
SMMU 0
// Note that range of StreamIDs that map to DeviceIDs excludes
// the NIC 0 DeviceID as it does not generate MSIs
// Input ID --> Output reference: Output ID
0x0000-0xffff --> ITS GROUP 0 : 0x10000->0x1ffff
// SMMU 0 Control interrupt is MSI based
// Input ID --> Output reference: Output ID
N/A --> ITS GROUP 0 : 0x200001
NIC 1
// end point device doesn't have an explicit input ID
// Input ID --> Output reference: Output ID
N/A --> ITS GROUP 0 : 0x30000

```

In the example above, each component declares any IDs it owns. For each ID, it provides detail of how that ID maps to the ID of another component. The pseudocode describes both the topological arrangement of the system and the ID relationships. For IDs that cover a range, the IDs described allow computing input to output mappings by using the following simple offsetting:

$$\text{OutputID} = \text{ID} - \text{Input Base} + \text{Output base}$$

For example, RC B has an ID range that can be described as follows:

0x0000-0xffff --> SMMU 0 : 0x0000->0xffff

That is, RID range 0x0000-0xffff maps to the StreamID range of 0x0000-0xffff belonging to SMMU 0. Therefore, for example, RID 0x0003, when emitted from RC B, maps to a StreamID as follows:

$$\text{StreamID } 0x0003 = \text{RID } 0x0003 - \text{RID base } 0x0000 + \text{StreamID base } 0x0000$$

In the example, the DeviceID mapping follows from the ID mapping for SMMU 0, which has the following format:

0x0000-0xffff --> ITS GROUP 0 : 0x10000->0x1ffff

The StreamID range 0x0000 to 0xffff for SMMU 0 maps to the DeviceID range of 0x10000 to 0x1ffff for ITS GROUP 0. Therefore, the StreamID of 0x0003 maps as follows:

$$\text{DeviceID } 0x10003 = \text{StreamID } 0x0003 - \text{StreamID base } 0x0000 + \text{DeviceID base } 0x10000$$

Endpoint IDs are described directly. For example, NIC 1 has a DeviceID of 0x30000. Effectively, these IDs can be treated as range mappings, where the range contains only one ID.

Note that the resolution of the ID space for a PCIe root complex, an SMMU, or an ITS might not always increase monotonically, as you move further out in the system, that is, from PCIe RID (16 bits) to SMMU (IMPLEMENTATION DEFINED) or ITS (IMPLEMENTATION DEFINED). For example, an SMMU might implement an 8-bit StreamID space. The hardware can be arranged so that not every bit of the RID is emitted to the SMMU. For example, the following relationship might hold:

- StreamID bits[5:0] = RID bits[5:0].
- StreamID bits[7:6] = RID bits[9:8].

This type of relationship can still be described in pseudocode by describing the individual mappings that arise:

```
RC X
// Input ID  --> Output reference: Output ID
0x0000-0x003F --> SMMU Y      : 0x00->0x3F
// 0x0040-0x00FF // Invalid range
0x0100-0x013F --> SMMU Y      : 0x40->0x7F
// 0x0140-0x01FF // Invalid range
0x0200-0x023F --> SMMU Y      : 0x80->0xBF
// 0x0240-0x02FF // Invalid range
0x0300-0x033F --> SMMU Y      : 0xC0->0xFF
// 0x0340-0xFFFF // Invalid range
```

All of the relationships described in this example can be described using the ID mapping arrays provided by the IORT nodes. See Table 3 and Table 4 for further detail. The following sections provide some examples.

### Representing components that generate MSIs and are connected to an SMMU

In the example system, root complex B is connected to SMMU 0 which in turn connects to ITS GROUP 0. The ID relationships for a PCIe device behind root complex B are as follows:

- StreamID for SMMU 0 = RID.
- DeviceID for ITS GROUP 0 = StreamID + 0x10000.

The resulting IORT Node for root complex B has the following single ID entry:

Field	Description
Length	0xffff
Flags	0x0
Input Base	0x0
Output base	0x0
Output Reference	Offset from the start of the IORT to the start of the SMMU 0 node.

The ID table maps the whole 16-bit RID range, bit by bit, to a 16-bit StreamID range. The node for SMMU 0 has the following ID entry:

Field	Description
Length	0xffff

Flags	0x0
Input Base	0x0
Output base	0x10000
Output Reference	Offset from start of the IORT to the start of the ITS GROUP 0 node.

The ID table maps the 16-bit StreamID range to a DeviceID range, with an offset of 0x10000.

### Representing components that generate MSIs but are not connected to an SMMU

In the example, root complex A, RC A, can generate MSIs but is not connected to an SMMU. For RC A the IORT Node ID array has the following single entry:

Field	Description
Length	0xffff
Flags	0x0
Input Base	0x0
Output base	0x0
Output Reference	Offset from the start of the IORT to the start of an ITS GROUP 0.

This example maps one-to-one between a RID and a DeviceID, that is, DeviceID = RID.

NIC 1 also follows this pattern, and has the following ID mapping in its IORT node:

Field	Description
Length	0x1
Flags	0x0
Input Base	0x0
Output base	0x30000
Output Reference	Offset from the start of the IORT to the start of an ITS GROUP 0.

This mapping results in a DeviceID of 0x30000 for NIC 1.

### Representing Components that do not generate MSIs but are connected to an SMMU

There are essentially two options in this category of components:

1. Mappings are provided that generate a DeviceID for the device, but as the device driver does not use MSIs, the DeviceID remains unused.
2. Mappings are provided that make it impossible to reserve a DeviceID for the device.

Option 2 is described below and applies to NIC 0 in the example system. This device represents its StreamID relationship to SMMU 0 with the following ID entry:

Field	Description
Length	0x1
Flags	0x0
Input Base	0x0
Output base	0x10000
Output Reference	Offset from the start of the IORT to the start of the SMMU 0 node.

StreamID 0x10000 is reserved in SMMU 0 for NIC 0. For SMMU 0, the following ID mapping is declared for StreamID to DeviceID conversion:

Field	Description
Length	0xffff
Flags	0x0
Input Base	0x0
Output base	0x10000
Output Reference	Offset from the start of the IORT to the start of an ITS GROUP 0.

The omission of a StreamID to DeviceID mapping for StreamID 0x10000, generated by NIC 0, implies that NIC 0 does not generate MSIs.

## Appendix B OS Usage of Memory Attributes

Figure 3 illustrates how an IORT-aware OS can interpret the memory attribute properties of a device, when the OS maps device shared memory on the CPU as IWB-OWB-ISH.

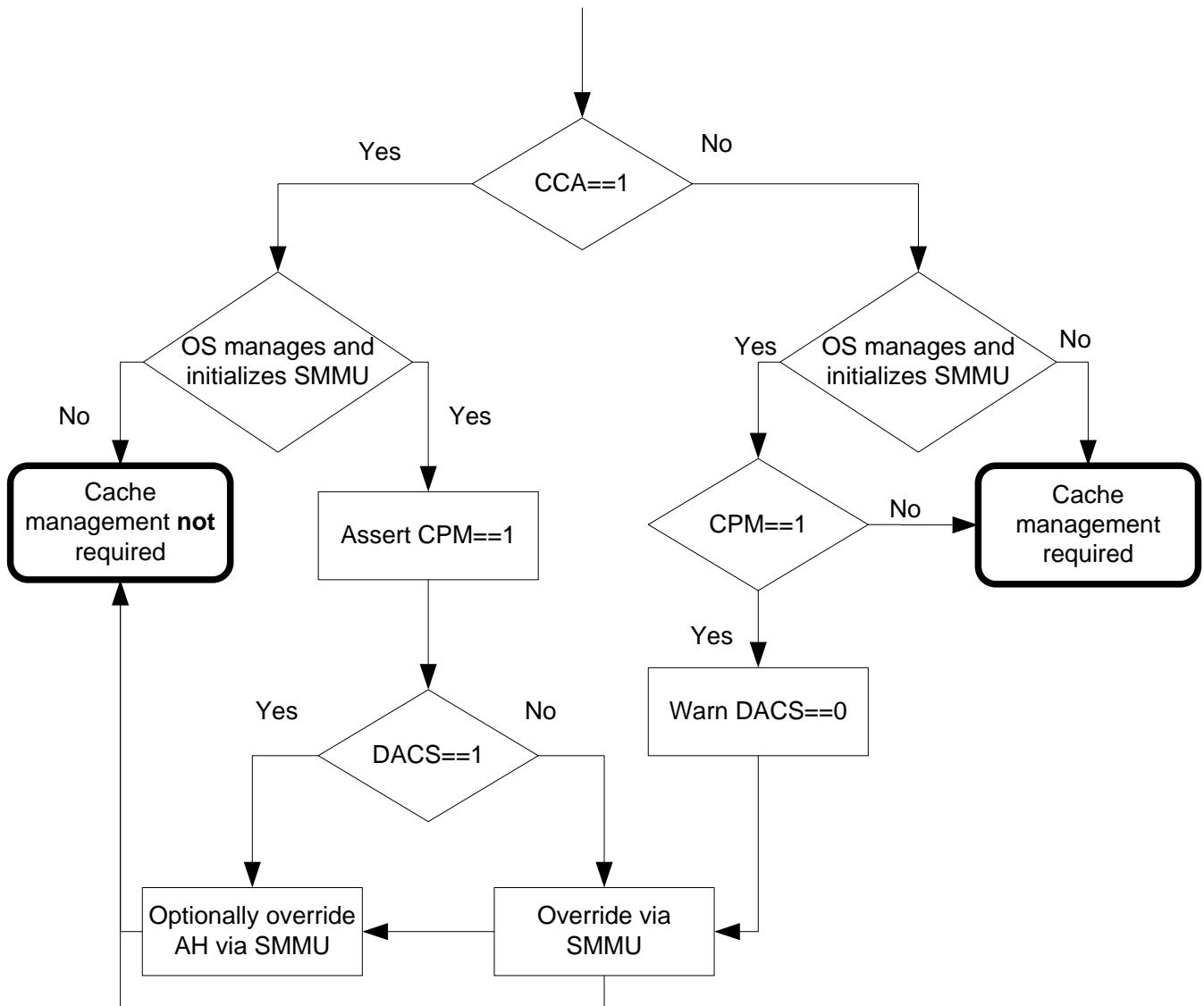


Figure 3 OS decision tree for cache management assuming CPU maps memory as IWB-OWB-ISH

Table 17 describes cache management requirements in more detail, taking into account the values of the flags, the memory mapping type used on the CPU, and the override memory type used by the SMMU, when applicable. In the table, NC represents a Non-cacheable type (Normal non-cacheable or Device).



**Table 17 Cache management requirements**

CCA	CPM	DACS	SMMU override for memory mapping	CPU memory mapping	Cache Maintenance required*
1	1	1	None	IWB-OWB-ISH	No
1	1	1	NC	NC	No
1	1	1	NC	IWB-OWB-ISH	Yes
0	1	0	IWB-OWB-ISH	IWB-OWB-ISH	No
0	1	0	None	NC	No
0	1	0	None	IWB-OWB-ISH	Yes
0	0	0	N/A	IWB-OWB-ISH	Yes
0	0	0	N/A	NC	No

\* Note: The caching operations that are described in this document apply to the CPU caches and any other caches in the system where device memory accesses can hit.