**# Title:**

Correct example in \_DMA method definition

**# Status:**

Draft

**# Document:**

ACPI Specification Version 6.4 (Future Errata)

**# License:**

SPDX-License-Identifier: CC-BY-4.0

**# Submitter:**

* Samer El-Haj-Mahmoud, ARM
* TianoCore Community (<https://www.tianocore.org>)

**# Summary of the change**

Correct the resource type example of \_DMA method

**# Benefits of the change**

The resources described with the \_DMA method are defined as:

<https://uefi.org/specs/ACPI/6.4/06_Device_Configuration/Device_Configuration.html#dma-direct-memory-access>

“ It specifies the ranges the bus controller (bridge) decodes on the child-side of its interface. (This is analogous to the \_CRS object, which describes the resources that the bus controller decodes on the parent-side of its interface.) Any ranges described in the resources of a \_DMA object can be used by child devices for DMA or bus master transactions.”

The QWordMemory macro defined the resource type as:

<https://uefi.org/specs/ACPI/6.4/19_ASL_Reference/ACPI_Source_Language_Reference.html#qwordmemory-qword-memory-resource-descriptor-macro>

“ResourceUsage specifies whether the Memory range is consumed by this device (ResourceConsumer) or passed on to child devices (ResourceProducer). If nothing is specified, then ResourceConsumer is assumed.”

However, the \_DMA example in section 6.2.4 uses a “ResourceConsumer”, when it should really be “ResourceProducer” according to these definitions: It describes , the child devices view of the address range, so the "translation" added is the CPU's view of the same range.

This is consistent with the Linux code implementation of \_DMA, which checks for ResourceProducer:

<https://elixir.bootlin.com/linux/v5.12-rc7/source/drivers/acpi/resource.c#L225>

**# Impact of the change**

Platforms using \_DMA may need to make sure the resource is described correctly as ResourceProducer. Currently, RPi4 UEFI code in edk2-platforms has this issue, and there is a patch to correct it, pending ACPI spec example clarification (in this ECR), see <https://edk2.groups.io/g/devel/topic/81935645#73858>

**# Detailed description of the change [normative updates]**

* Insertions in **green**
* Removals in **~~red~~**

**6.2.4. \_DMA (Direct Memory Access)**

…

\_DMA Example ASL:

Device(BUS0)

{

//

// The \_DMA method returns a resource template describing the

// addresses that are decoded on the child side of this

// bridge. The contained resource descriptors thus indicate

// the address ranges that bus masters living below this

// bridge can use to send accesses through the bridge toward a

// destination elsewhere in the system (e.g. main memory).

//

// In our case, any bus master addresses need to fall between

// 0 and 0x80000000 and will have 0x200000000 added as they

// cross the bridge. Furthermore, any child-side accesses

// falling into the range claimed in our \_CRS will be

// interpreted as a peer-to-peer traffic and will not be

// forwarded upstream by the bridge.

//

// Our upstream address decoder will only claim one range from

// 0x20000000 to 0x5fffffff in the \_CRS. Therefore \_DMA

// should return two QWORDMemory descriptors, one describing

// the range below and one describing the range above this

// "peer-to-peer" address range.

//

Method(\_DMA, ResourceTemplate()

{

QWORDMemory(

~~ResourceConsumer,~~

ResoureceProducer,

PosDecode, // \_DEC

MinFixed, // \_MIF

MaxFixed, // \_MAF

Prefetchable, // \_MEM

ReadWrite, // \_RW

0, // \_GRA

0, // \_MIN

0x1fffffff, // \_MAX

0x200000000, // \_TRA

0x20000000, // \_LEN

,

,

,

)

QWORDMemory(

~~ResourceConsumer,~~

ResoureceProducer,

PosDecode, // \_DEC

MinFixed, // \_MIF

MaxFixed, // \_MAF

Prefetchable, // \_MEM

ReadWrite, // \_RW

0, // \_GRA

0x60000000, // \_MIN

0x7fffffff, // \_MAX

0x200000000, // \_TRA

0x20000000, // \_LEN

,

,

,

)

})

}