Arm® PCI Configuration Space Access Firmware Interface 1.0BET1

Platform Design Document

Non-confidential

Notice

This document is a Beta version of a specification undergoing review by Arm partners. It is provided to give advanced information only.

Document number: DEN0115A

Contents

Release information

Date	Version	Changes
2021/May/03	1.0BET1	• Clarified that this is a standard firmware interface that is an alternative to ECAM, not just a workaround. • Added implementation note on returning an error when accessing > 4KB window.
2020/Dec/14	1.0BET0	• Initial version of the specification.

Arm Non-Confidential Document Licence (“Licence”)

This Licence 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 Licence (“**Document**”). Arm licenses its intellectual property in the Document to you on condition that you agree to the terms of this Licence. By using or copying the Document you indicate that you agree to be bound by the terms of this Licence.

“**Subsidiary**” means any company the majority of whose voting shares is now or hereafter owner 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 Licence between you and Arm.

Subject to the terms and conditions of this Licence, 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 licence 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 licence granted in (i) above; and

(iii) sell, supply and distribute products which have been created under the licence granted in (i) above.

Licensee hereby agrees that the licences 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 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.

NOTWITHSTANING ANYTHING TO THE CONTRARY CONTAINED IN THIS LICENCE, TO THE FULLEST EXTENT PETMITTED 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 LICENCE (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 LICENCE). 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 Licence 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 Licence then Arm may terminate this Licence immediately upon giving written notice to Licensee. Licensee may terminate this Licence at any time. Upon termination of this Licence 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 Licence, all terms shall survive except for the licence grants.

Any breach of this Licence by a Subsidiary shall entitle Arm to terminate this Licence as if you were the party in breach. Any termination of this Licence 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.

LES-PRE-21585 version 4.0

About this document

Terms and abbreviations

Term	Meaning
CPU	A hardware implementation of the Arm architecture
ECAM	PCIe Enhanced Configuration Access Mechanism. See [1]
EL	Exception Level
FW	Firmware
HVC	Hypervisor Call, an Arm assembler instruction that causes an exception that is taken synchronously into EL2
OS	Operating System
PE	Processing element. An abstract machine defined in the Arm architecture, see [2]
RW1C	Write-one-to-clear PCI registers. See [1]
SMC	Secure Monitor Call. An Arm assembler instruction that causes an exception that is taken synchronously into EL3
SoC	System on Chip

References

This section lists publications by Arm and by third parties.

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

[1] *PCI Express Base Specification Revision 5.0, version 1.0*. PCI-SIG.

[2] *Arm® Architecture Reference Manual for Armv8-A architecture profile*. (ARM DDI 0487 E.a) Arm Ltd.

[3] *SMC Calling Convention System Software on Arm® Platforms*. (ARM DEN 0028 C) Arm Ltd.

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. Give:

- The title (PCI Configuration Space Access Firmware Interface).
- The document ID and version (DEN0115A 1.0BET1).
- The page numbers to which your comments apply.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

1 Introduction

This document defines a standard firmware interface for a caller, such as an OS or a hypervisor, to access the PCI configuration space.

This interface can be used as an alternative to the Enhanced Configuration Access Mechanism (ECAM) hardware mechanism, which is defined in the PCIe Specification [1].

The interface described in Section 2 enables a caller to:

- Access PCI configuration space reads/writes
- Discover the implemented PCI segment groups, and bus ranges for each segment

1.1 Calls defined per ABI version

Table 3 relates the ABI version to the defined calls and their requirement status.

Table 3: ABI required functions per version

Call name	Mandatory from	Optional from
PCI_VERSION	v1.0	–
PCI_FEATURES	v1.0	–
PCI_READ	v1.0	–
PCI_WRITE	v1.0	–
PCI_GET_SEG_INFO	v1.0	–

1.2 Calling convention

This ABI complies with the SMCCCv1.1 [3] calling convention. The ABI can only be present in a system that is compliant with SMCCCv1.1 or higher.

All specified functions take 32-bit parameters and return 32-bit values. The ABI is designed so that all functions are callable from an AArch32 or AArch64 client context.

In systems that implement EL3, Arm recommends the use of the SMC conduit to call the functions that are defined in this specification. If EL3 is not present, but EL2 is present, then the HVC conduit must be used.

1.3 ABI discovery

The SMCCC mandates the SMCCC implementation to return NOT_SUPPORTED if the called function is not implemented [3].

The presence of the PCI Configuration Space Access ABI must be discovered by calling PCI_VERSION. A PCI Configuration Space Access ABI implementation is present if and only if a call to PCI_VERSION returns a non-negative value in W0.

The PCI_FEATURES function must be present in any implementation of this ABI. The PCI_FEATURES function is implemented if a call to PCI_VERSION returns a non-negative value in W0.

The presence of the remaining functions in the PCI Configuration Space Access ABI is determined through calls to PCI_FEATURES passing the FID of the call as the argument in W1 (pci_func_id). See section 2.2 for information on PCI_FEATURES.

Mandatory functions are guaranteed to be present if the PCI Configuration Space Access ABI version is greater or equal than the version of the ABI the particular function was mandated on. See section 1.1 for information on ABI versions and mandatory functions.

For platforms that support the ACPI FW interface, the FW must not publish the ‘MCFG’ ACPI table when this ABI is intended to be used for PCI configuration space access. This is necessary to allow the operating systems and hypervisors to discover this ABI and use it instead of the PCIe ECAM interface described in ‘MCFG’.

2 Interface

2.1 PCI_VERSION

The function returns the implemented version of the PCI Configuration Space Access ABI. The version is composed of two revision fields, major and minor.

2.1.1 Function definition

Function ID (W0)	0x8400_0130		
Parameters			
	W1–W7	Reserved (MBZ)	
Returns			
uint32	Success		
		W0[30:16]	Major revision
		W0[15:0]	Minor revision
		W1 – W3	Reserved (MBZ)

Table 4: PCI_VERSION function definition

2.1.2 Usage

The function returns a 15-bit major revision and a 16-bit minor revision as an aggregate 31-bit value in R0/W0. The 15 bits W0[30:16] contain the major revision, and the least significant 16 bits (W0[15:0]) contain the minor revision. A minor revision increment cannot break backward compatibility with older minor revisions within the same major revision. A major revision can introduce changes which break compatibility with prior major revisions. The caller can use the return value as a discovery mechanism for ABI functions that Section 1.1 lists as mandatory.

2.1.3 Caller responsibilities

The caller has the following responsibilities:

- The caller must ensure that SMCCC_VERSION reports a SMCCC version greater or equal than 1.1 [3] before calling PCI_VERSION.

2.1.4 Implementation responsibilities

The Implementation has the following responsibilities:

- The implementation must guarantee that all the mandatory functions are implemented for the version that it reports, as specified in Section 1.1.

2.2 PCI_FEATURES

The caller discovers PCI Configuration Space Access ABI functions implemented in the FW.

2.2.1 Function definition

Function ID (W0)	0x8400_0131			
Parameters				
	W1	pci_func_id		
	W2–W7	Reserved (MBZ)		
Returns				
	Success (W0 $\geqslant$ 0)			
			SUCCESS	Function is implemented.
			> 0	Function is implemented and has specific capabilities, see function definition.
	Error (W0 $<$ 0)			
			NOT_SUPPORTED	Function with FID=pci_func_id is not implemented

Table 5: PCI_FEATURES function definition

2.2.2 Usage

The caller can determine if functions defined in the PCI Configuration Space Access ABI are present in the ABI implementation. The caller can determine function specific features (signaled by a positive return status in W0). The function specific features must be described in the function definition.

2.2.3 Caller responsibilities

The caller has the following responsibilities:

- The caller must ensure the PCI Configuration Space Access ABI is present before calling this function.

2.2.4 Implementations responsibilities

The function implementation has the following responsibilities:

- The implementation must return NOT_SUPPORTED if pci_func_id is a value not defined in the PCI Configuration Space Access ABI.

2.3 PCI_READ

The caller reads a given register address from the PCI configuration space of a given PCI device.

2.3.1 Function definition

Function ID (W0)	0x8400_0132		
Parameters			
	W1	PCI device address	
		Bits [31:16] : PCI segment group number	
		Bits [15:8] : PCI bus number	
		Bits [7:3] : PCI device number	
		Bits [2:0] : PCI function number	
	W2	Register offset	
	W3	Access size for read operations	
		1 : 1 Byte	
		2 : 2 Bytes	
		4 : 4 Bytes	
		All other values reserved	
	W4 – W7	Reserved (MBZ)	
Returns			
	Success (W0 = 0)		
		W0	MBZ
		W1	Data Read from the PCI device configuration space register
	Error (W0 < 0)		
		W0	NOT_SUPPORTED INVALID_PARAMETER
		W1	Reserved (MBZ)

Table 6: PCI_READ function definition

2.3.2 Usage

The call returns the data read from the configuration space of the PCI device register identified by the input Parameters.

2.3.3 Caller responsibilities

The caller has the following responsibilities:

- The caller must ensure that this function is implemented before issuing a call. This function is discoverable by calling PCI_FEATURES with pci_func_id set to 0x8400_0132.

2.3.4 Implementation responsibilities

The Implementation has the following responsibilities:

- The implementation must return INVALID_PARAMETER if any of the W4–W7 registers differs from zero, or if the W3 register contains a value other than 1, 2, or 4.
- The implementation must return INVALID_PARAMETER if the addressed register offset and access size is greater than 4KB.
- The implementation must ensure that concurrent calls to any of the functions in the PCI Configuration Space Access ABI are multi-processor safe.
- On some platforms, the SoC may only support 32-bit PCI configuration space reads. On such platforms, calls to this function with access size of 1 or 2 bytes may result in inadvertently corrupting adjacent fields. This could happen, for example, if the adjacent fields have configuration space access read side effects, as defined in [1]. It is the implementation responsibility to be aware of these situations and guard against them if possible.

2.4 PCI_WRITE

The caller writes a value to a given register address in the PCI configuration space of a given PCI device.

2.4.1 Function definition

Function ID (W0)	0x8400_0133	
Parameters		
	W1	PCI device address
		Bits [31:16] : PCI segment group number
		Bits [15:8] : PCI bus number
		Bits [7:3] : PCI device number
		Bits [2:0] : PCI function number
	W2	Register offset
	W3	Access size for write operation
		1 : 1 Byte
		2 : 2 Bytes
		4 : 4 Bytes
		All other values reserved
	W4	Data to write to the PCI device
		configuration space register
	W5 – W7	Reserved (MBZ)
Returns		
	Success (W0 = 0)	
		W0 MBZ
	Error (W0 < 0)	
		W0 NOT_SUPPORTED
		INVALID_PARAMETER

Table 7: PCI_WRITE function definition

2.4.2 Usage

The call writes the requested input data to the configuration space of the PCI device register identified by the input Parameters.

2.4.3 Caller responsibilities

The caller has the following responsibilities:

- The caller must ensure that this function is implemented before issuing a call. This function is discoverable by calling PCI_FEATURES with pci_func_id set to 0x8400_0133.

2.4.4 Implementation responsibilities

The Implementation has the following responsibilities:

- The implementation must return INVALID_PARAMETER if any of the W5–W7 registers differs from zero, or if the W3 register contains a value other than 1, 2, or 4.
- The implementation must return INVALID_PARAMETER if the addressed register offset and access size is greater than 4KB.
- The implementation must complete the PCI configuration space write operation before returning to the caller.
- The implementation must ensure that concurrent calls to any of the functions in the PCI Configuration Space Access ABI are multi-processor safe.
- On some platforms, the SoC may only support 32-bit PCI configuration space writes. On such platforms, calls to this function with access size of 1 or 2 bytes may require the implementation of this function to perform a PCI configuration read, following by the write. This could result in inadvertently corrupting adjacent RW1C fields. It is the implementation responsibility to be aware of these situations and guard against them if possible.

2.5 PCI_GET_SEG_INFO

The caller gets information on the available PCI segment groups in the platform, and the PCI bus ranges for each segment.

2.5.1 Function definition

Function ID (W0)	0x8400_0134		
Parameters			
	W1		
		Bits [31:16] Reserved (MBZ)	
		Bits [15:0] pci_seg : PCI segment group number	
	W2 – W7	Reserved (MBZ)	
Returns			
	Success (W0 = 0)		
		W0	MBZ
		W1	Bits [31:16] : Reserved (MBZ)
			Bits [15:8] : Ending PCI Bus number
			Bits [7:0] : Starting PCI Bus number
		W2	pci_next_seg : Next PCI segment group number,
			or zero if pci_seg is the last segment.
	Error (W0 < 0)		
		W0	NOT_SUPPORTED
			INVALID_PARAMETER
			NOT_IMPLEMENTED
		W1	Reserved (MBZ)
		W2	Reserved (MBZ)

Table 8: PCI_GET_SEG_INFO function definition

2.5.2 Usage

The call checks if a requested PCI segment group in pci_seg is implemented, and returns the PCI bus range for that segment. It also returns the next implemented pci_next_seg, if there are any, or zero if this is the last segment. This allows the caller to discover all implemented PCI segments and their PCI bus ranges.

2.5.3 Caller responsibilities

The caller has the following responsibilities:

- The caller must ensure that this function is implemented before issuing a call. This function is discoverable by calling PCI_FEATURES with pci_func_id set to 0x8400_0134.
- The caller may use this function to iterate through all the supported PCI segments in the platform. This can be done by calling this function with the value of zero in the pci_seg parameter, then using the value returned in pci_next_seg as the pci_seg input for subsequent calls to this function, until a value of zero is returned in pci_next_seg.

2.5.4 Implementation responsibilities

The Implementation has the following responsibilities:

- The implementation must return INVALID_PARAMETER if any of the W3–W7 registers differs from zero, or if the W1[31:16] differ from zero.
- The implementation must ensure that concurrent calls to any of the functions in the PCI Configuration Space Access ABI are multi-processor safe.
- The implementation must implement PCI segment group zero.
- The implementation must return NOT_IMPLEMENTED if the segment specified in register W1[15:0] is not implemented by the platform.
- For platforms that support the ACPI FW interface, the implementation must ensure that the PCI segment group numbers used in this ABI correspond to the values used in the ACPI name space for the applicable host bridge device.

2.6 Return codes

The following status return codes are defined for the PCI Configuration Space Access ABI calls.

Name	Value
SUCCESS	0
NOT_SUPPORTED	-1
INVALID_PARAMETER	-2
NOT_IMPLEMENTED	-3