Arm® Errata Management Firmware Interface 1.0EAC1

Platform Design Document

Non-confidential

Document number: DEN0100

Contents

Release information

Date	Version	Changes
2022/Oct/07	1.0EAC1	• Minor text clarifications • Add usage Appendix
2021/Sep/06	1.0EAC0	• First document release

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.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-21585 version 4.0

About this document

Terms and abbreviations

Term	Meaning
CPU	A hardware implementation of the Arm architecture.
EL	Exception Level
Erratum	The description of a hardware feature that deviates from the hardware designer intent, and which is perceivable in some form by the software that is executing upon the platform.
HVC	Hypervisor Call, an Arm assembler instruction that causes an exception that is taken synchronously into EL2.
IP	Intellectual Property
OS	Operating System.
SMC	Secure Monitor Call. An Arm assembler instruction that causes an exception that is taken synchronously into EL3.
SoC	System on Chip
Workaround	A set of steps that software must implement in order to mitigate a specific erratum. Some workarounds can be entirely implemented by a single EL, others require actions by the calling and a higher EL. This is called a split responsibility workaround.

References

This section lists publications by Arm and by third parties.

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

[1] *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 (Errata Management Firmware Interface).
- The document ID and version (DEN0100 1.0EAC1).
- 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 firmware interface for an OS or hypervisor to discover details about CPU errata.

Errata describe hardware features which deviate from the design intent. An erratum can have an associated workaround, implementable in software, to mitigate the erratum. Some workarounds are implementable at different ELs. Also, some workarounds may require actions to be taken at multiple ELs to fully mitigate the erratum. An OS must be able to discover the errata that it must deploy mitigations for.

The interface described in Section 2 enables an OS to:

- Discover the errata known to higher ELs and that have been fixed in hardware or mitigated at a higher EL.
- Discover the errata which require mitigation by the OS.

Any IP that is present in the SoC can potentially have defects and consequently errata. The version of the interface that is described in this document solely handles CPU errata. The interface assumes that firmware only implements a single workaround for each erratum.

1.1 Calls defined per ABI version

The following table relates the ABI version to the defined calls and their requirement status.

Call name	Mandatory from	Optional from
EM_VERSION	v1.0	–
EM_FEATURES	v1.0	–
EM_CPU_ERRATUM_FEATURES	v1.0	–

1.2 CPU IP erratum

A CPU IP erratum is identified by the CPU_erratum_ID identifier. The CPU_erratum_ID is a core IP vendor specified 32 bit value that unambiguously identifies the erratum on a particular core. A disclosed erratum must specify the CPU_erratum_ID and the core that it relates to.

The core IP vendor must provide the following documentation for every erratum:

- MIDR[63:4] of the affected core and list of known affected core revisions.
  - The core revision is defined by the MIDR[3:0] field and can be augmented on a core basis by other relevant ID registers.
- CPU erratum identifier (CPU_erratum_ID).
- Description or a pointer to a document describing the workaround to be implemented.
- Exception levels where the workaround can be implemented.

1.3 Calling convention

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

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.

1.4 ABI discovery

The SMCCC mandates the SMCCC implementation to return NOT_SUPPORTED if the called function is not implemented [1].

The presence of the Errata ABI must be discovered by calling EM_VERSION. An Errata ABI implementation is present if and only if a call to EM_VERSION returns a non-negative value in W0.

The EM_FEATURES function must be present in any Errata ABI implementation. The EM_FEATURES function is implemented if a call to EM_VERSION returns a non-negative value in W0.

The presence of the remaining functions in the Errata ABI is determined through calls to EM_FEATURES passing the FID of the call as the argument in W1 (em_func_id). See section 2.2 for information on EM_FEATURES.

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

1.5 Errata status

The errata management interface allows the firmware to report the following statuses on specific CPU errata:

- Unknown: The firmware does not recognise the erratum identifier (<CPU_erratum_ID>) for the calling CPU, or the erratum is **not** mitigated by a higher EL and the erratum cannot be mitigated by the calling EL.
- Not affected: The erratum was fixed in hardware. This core revision is not affected by the erratum.
- Mitigated at a higher EL: The erratum is fully mitigated at a higher EL.
- Affected: The erratum is not fully mitigated by a higher exception level.

Note: For a split responsibility workaround:

- If a higher EL implements its half of the workaround then the erratum status is Affected.
- If a higher EL does **not** implement its half of the workaround then the erratum status is Unknown.

1.5.1 Errata status applicability to Exception Levels

On real platforms it is plausible that EL3 firmware could have fresher information on relevant errata when compared to a hypervisor at EL2. EL3 should distinguish between EL1 and EL2 callers and reply to an EL1 caller accordingly – even if indirectly via EL2.

The function EM_CPU_ERRATUM_FEATURES (Section 2.3) is defined relative to the calling EL.

The status returned by EM_CPU_ERRATUM_FEATURES refers to the calling EL or lower:

- If EL2 is the calling EL - the return of EM_CPU_ERRATUM_FEATURES refers to {EL2, EL1, EL0}.
- If EL1 is the calling EL - the return of EM_CPU_ERRATUM_FEATURES refers to {EL1, EL0}.

1.5.2 Errata status result predictability

For a particular CPU, any two calls to EM_CPU_ERRATUM_FEATURES that are made in the interval from system boot until system power off, from the same EL and with the same CPU_erratum_ID argument, must return the same status.

Note: Some erratum may be induced by factors that are external to a particular CPU implementation. On some platforms, CPUs that are otherwise identical can have different affected statuses. The return of EM_CPU_ERRATUM_FEATURES is only valid for the calling CPU.

2 Interface

2.1 EM_VERSION

The function returns the implemented version of the Errata ABI. The version is composed of two revision fields, major and minor.

2.1.1 Function definition

Function ID (W0)	0x8400_00F0		
Parameters			
	W1–W7	Reserved (MBZ)	
Returns			
int32	(Success $\geqslant$ 0)		
		W0[30:16]	Major revision
		W0[15:0]	Minor revision
		W1 – W3	Reserved (MBZ)

Table 4: EM_VERSION function definition

2.1.1.1 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 previous major revisions. The caller can use the return value as a discovery mechanism for ABI functions that Section 1.1 lists as mandatory.

2.1.1.2 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 [1] before calling EM_VERSION.

2.1.1.3 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 EM_FEATURES

The caller can use the function EM_FEATURES to discover the Errata ABI functions that are implemented in the firmware.

2.2.1 Function definition

Function ID (W0)	0x8400_00F1		
Parameters			
	W1	em_func_id	
	W2–W7	Reserved (MBZ)	
Returns			
int32	Success (W0 $\geqslant$ 0)		
		0	Function is implemented.
		> 0	Function is implemented and has specific capabilities, see function definition.
	Error (W0 $<$ 0)		
		NOT_SUPPORTED	Function with FID=em_func_id is not implemented

Table 5: EM_FEATURES function definition

2.2.2 Usage

The caller can determine if functions that are defined in the Errata ABI are present in the ABI implementation. The caller can determine function specific features, which are 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 Errata ABI is present before calling this function.

2.2.4 Implementation responsibilities

The function implementation has the following responsibilities:

- The implementation must return NOT_SUPPORTED if em_func_id is a value not defined in the Errata ABI.

2.3 EM_CPU_ERRATUM_FEATURES

The caller obtains the features of a given CPU erratum. These features describe whether software at the calling EL or lower can be affected by an erratum. See Section 1.5.1 for more information.

2.3.1 Function definition

Function ID (W0)	0x8400_00F2			
Parameters				
	W1	CPU_erratum_ID		
	W2	forward_flag (MBZ when called from EL1)		
	W3–W7	Reserved (MBZ)		
Returns				
int32	Success ($W0 \geqslant 0$)			
			W0	HIGHER_EL_MITIGATION - Erratum is fully mitigated at a higher EL.
				NOT_AFFECTED - Erratum has been fixed in hardware.
				AFFECTED - Erratum is not fully mitigated by a higher EL.
	Error ($W0 < 0$)			
			W0	INVALID_PARAMETERS
				UNKNOWN_ERRATUM

Table 6: EM_CPU_ERRATUM_FEATURES function definition

2.3.1.1 Usage

The call returns the features of the erratum, identified by CPU_erratum_ID, on the calling core and related to the calling or lower ELs. See Section 1.5.1 for more information. When the call is made at EL2, the argument forward_flag can be used to emulate a call made from EL1. When forward_flag $\neq 0$ the implementation returns the status as if the call had been made from EL1.

2.3.1.2 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 EM_FEATURES with em_func_id set to 0x8400_00F2.
- A caller at EL1 must ensure forward_flag=0.

2.3.1.3 Implementation responsibilities

The Implementation has the following responsibilities:

- The firmware must implement at most one workaround per erratum.
- If the call originates in EL2 and forward_flag $\neq 0$ then the implementation must return the status as if the call had been made from EL1.
- The implementation must return:
  - INVALID_PARAMETERS if any of the W3–W7 registers differs from zero or if the call originates at EL1 and forward_flag $\neq 0$.
  - UNKNOWN_ERRATUM if the erratum with CPU_erratum_ID:

    - is not recognised by the implementation for the current CPU;
    - is *not* mitigated at a higher EL and the erratum cannot be mitigated by the calling EL or lower;
    - is split responsibility and the top half of the workaround is not implemented at any higher EL.
  - NOT_AFFECTED if the erratum has been fixed in hardware.
  - HIGHER_EL_MITIGATION if the erratum is fully mitigated at a higher EL.
  - AFFECTED if the calling EL or lower is responsible for mitigating the erratum with CPU_erratum_ID in the calling core.
- The status returned by a given EL must only reflect the information that is directly managed by this EL. For example, EL3 must not derive the status returned to an EL1 caller by obtaining information from registers controlled by EL2.

2.4 Return codes

The following status return codes are defined for Errata Management ABI calls.

Name	Value
HIGHER_EL_MITIGATION	3
NOT_AFFECTED	2
AFFECTED	1
SUCCESS	0
NOT_SUPPORTED	-1
INVALID_PARAMETERS	-2
UNKNOWN_ERRATUM	-3

Part I

Errata ABI usage

This Appendix describes the expected usage of the Errata ABI by an OS.

OS-side per-CPU erratum workaround detection

Once the OS has established that the Errata ABI is present and that EM_CPU_ERRATUM_FEATURES is implemented by firmware, the OS can call EM_CPU_ERRATUM_FEATURES. The return of EM_CPU_ERRATUM_FEATURES is valid for the calling CPU only, the call must be performed on each CPU that the OS knows can be affected by a particular erratum.

The following routine can be called on each CPU for the OS to determine if it must deploy the workaround for cpu_erratum_id on the calling CPU.

Listing: Determining if the local workaround for the erratum, identified by cpu_erratum_id, is required on the calling CPU.

```
/*
 * The routine can only be called once the OS established
 * that firmware implements EM_CPU_ERRATUM_FEATURES.
 *
 * A particular erratum may have different identifiers for variations of a CPU
 * IP where the workaround is the same.
 * An OS is expected to traverse a list of potential CPU erratum identifiers,
    ↪for the calling CPU,
 * in order to determine if the OS must deploy the local workaround for the
    ↪calling CPU.
 */
bool need_cpu_erratum_local_wa(u32 cpu_erratum_id_list[], int
    ↪num_erratum_entries)
{
    int forward_flag = 0;

    for (int idx = 0; idx < num_erratum_entries; idx++) {

        u32 cpu_erratum_id = cpu_erratum_id_list[idx];
        int ret = smccc_call(EM_CPU_ERRATUM_FEATURES, cpu_erratum_id,
            ↪forward_flag);

        switch (ret) {
          case EM_HIGHER_EL_MITIGATION:
          case EM_NOT_AFFECTED:
            return false;

          case EM_UNKNOWN_ERRATUM:
            // Firmware does not recognise the workaround with id
            // cpu_workaround_id on this CPU.
            // Continue traversing the ids in the cpu_erratum_id_list.
            continue;

          case EM_AFFECTED:
            // The CPU is affected by the erratum, the OS should
            // deploy a workaround.
            return true;
        }
    }
    // If the execution flow reached this point, then firmware returned
        ↪EM_UNKNOWN_ERRATUM
    // for all the elements in cpu_erratum_id_list.
    // The OS should adopt an OS-specific policy on whether to deploy the local
        ↪ workaround
    // or not.
    return true;
}
```