ARM® Architecture Standard Configurations
System Software on ARM® Processors

Document number: ARM DEN0016B

ARM®

ARM Architecture Standard Configurations
System Software on ARM Processors

Release information

The Release history table lists the releases of this document.

Table 1 Release history

Date	Issue	Confidentiality	Change
19 Jul 2011	A	Confidential	First release
19 Aug 2012	B	Non-confidential	Second release

Proprietary notice

ARM web address

http://www.arm.com

Table of Contents

1 Introduction

The ARM architecture contains a significant number of features that are described as OPTIONAL or IMPLEMENTATION DEFINED. Feedback from platform operating system vendors indicates that this variability is a substantial problem for developers of system code, because operating systems must cater for a wide variety of different system configurations, leading to increases in development cost and reducing portability and quality.

This document examines the high degree of configurability of the ARM architecture, and proposes an approach that can be used to avoid this configurability being a problem for the operating system vendors.

1.1 Additional reading

This section lists publications by ARM and by third parties.

See Infocenter, http://infocenter.arm.com, for access to ARM documentation.

1.1.1 ARM publications

The following documents contain information relevant to this document:

- *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition* (ARM DDI 0406)
- *ARM Generic Interrupt Controller Architecture Specification* (ARM IHI 0048).

2 Background

ARM processors are used in a wide variety of system-on-chip (SoC) products in diverse markets. The constraints on products in these markets are inevitably very different, and it is impossible to produce a single product that meets of the needs of all these markets.

The ARMv7 architecture profiles – Application (A), Real-time (R), and Microcontroller (M) – help segment the solutions produced by ARM and describe the characteristics of particular target markets. The differences between products implementing the different profiles are typically substantial because of the very different functional requirements of their target market segments.

However, even within an architectural profile, the wide range of uses of a product means there are frequent requests for features to be removed to save silicon area. This is particularly the case for products targeted at cost sensitive markets, where the cost of customizing the software to allow for the lack of a feature is small compared to the overall cost-saving of removing features.

In other markets, such as those that require an open platform where the software to be run is very complex, the savings from removing a feature from the hardware are outweighed by the cost of software development to support the different variants. In addition, software development is often performed by third parties, and uncertainty about whether new features are widely deployed can act as a substantial brake to the adoption of those features.

The ARM Application profile must balance these two competing business pressures. It offers a wide range of features, such as Floating-point, Advanced SIMD, and TrustZone technology, to tackle an increasing range of problems, while allowing features to be removed from implementations that do not need them and where the silicon area, and cost savings involved are compelling.

2.1 Feedback from the Platform Operating System vendors

Platform Operating System vendors make extensive use of the ARM Application profile, across multiple SoCs that contain processors designed by ARM and its Architectural licensees. Operating System vendors have stated that the variability of features delivered by differing implementations of the ARM Application profile results in significant cost, and associated quality risk, to their software development. They would like to see unnecessary variability constrained.

2.2 Standard Configurations

To address this problem, this document defines a set of *Standard Configurations* of the ARM Application profile. Operating System vendors and other system software vendors can require that an implementation must be consistent with one of these Standard Configurations to receive their full support. ARM processor implementations support these Standard Configurations, and ARM encourages their use by implementation licensees targeting the markets where such standardization is advantageous.

ARM does not mandate the use of these Standard Configurations, because there are customers and market segments where the benefit of standardization is less important, and where the original motivation for flexibility applies.

2.2.1 Principles behind the Standard Configurations

As ARM introduces major new features, additional Standard Configurations will be defined. However, the number of defined Standard Configurations will be substantially smaller than

the number of options available in the architecture, and new configurations will have a defined backwards-compatibility relationship.

An implementation is consistent with a Standard Configuration if it implements all of the functionality of that Standard Configuration at performance levels appropriate for the target uses of that Standard Configuration. This means that all functionality of a Standard Configuration can be exploited by software without unexpectedly poor performance.

Note: This is intended to avoid approaches such as software emulation of functionality that is critical to the performance of software that uses the Standard Configuration. It is not intended to act as a restriction of legitimate exploration of the power, performance or area tradeoffs that characterize different products, nor to restrict the use of trapping within a system that supports Virtualization.

An implementation that is consistent with a Standard Configuration might include additional features that are not included in the definition of that Standard Configuration. However, software written for a Standard Configuration must run, unaltered, on an implementation that includes such additional functionality.

Note: The UNDEFINED instruction behavior of an implementation that includes such additional features is not required to be identical to that of an implementation that implements only features defined in the Standard Configuration. It is assumed that software consistent with a Standard Configuration will not use UNDEFINED instruction encodings that might be defined in an extended system, or in a later Standard Configuration.

2.2.2 Software use of Standard Configurations

Software running on a system that includes an ARM processor inevitably includes code that is specific to the functionality of that system. Typically, such code is partitioned from the rest of the system software by Firmware, Hardware Abstraction Layers, Board Support Packages, Drivers, or similar constructs. This document refers to such constructs as *Hardware Specific Software*.

ARM recognizes that no exercise to standardize the configurations of the ARM processor or the platform can ever remove the need for such code. However, the provision of Standard Configurations of the ARM processors should reduce the range of functionality that has to be handled using Hardware Specific Software. Over time, as the scope and adoption of Standard Configurations increases, it is envisaged that some parts of the system that are currently handled using Hardware Specific Software will be handed by system software that is consistent with a particular Standard Configuration.

This document uses the term *software consistent with a Standard Configuration* to indicate pieces of software that are not Hardware Specific Software, and are designed to be fully portable between different implementations that are consistent with that Standard Configuration.

2.3 Status of Debug Components within Standard Configurations

Debug functionality provided by the ARM architecture has the following two distinct views:

- The programmers' model used by system software, such as the debug component within an operating system that programs the debug resources. This document takes this view to be synonymous with Monitor debug-mode, and refers to it as the *software view*.
- The view that is presented to an external debug agent. This document takes this view to be synonymous with Halting debug-mode.

In principle, these two views can be used simultaneously, provided appropriate steps are taken over the allocation of resources, such as breakpoints or watchpoints. However, this is rarely done outside initial system bring-up, particularly for systems targeted by the Standard Configurations. In other words, the most common uses of debug resources do not overlap. That is, the debug components might be:

- used by an external debug agent, or
- used by system software running on the target device.

The definition of the Standard Configurations only standardizes the software view, and only defines what debug components must be accessible for software debug use when no external debug agent is attached. The provision or otherwise of Halting debug functionality is outside the scope of the Standard Configurations.

Similarly, ETM and PTM functionality is considered to be outside the scope of the Standard Configurations.

2.3.1 Performance Monitors and Standard Configurations

Performance monitoring hardware must be present in a Standard Configuration, and must provide a minimum set of performance monitoring capabilities. However, many of the events counted by Performance Monitors depend on the micro-architecture of an implementation, so any software that uses or interprets Performance Monitor data is very likely to be Hardware Specific Software.

In addition, more complex implementations are likely to have more complex features to monitor and, therefore, are likely to implement more performance monitors than are implemented on a simpler, more area-sensitive implementation.

For this reason, the minimum level of functionality within a Standard Configuration does not imply that this is the only functionality that should be supported by an implementation of that configuration. Rather, it indicates the maximum level of functionality that can be used by software that is consistent with the Standard Configuration.

3 Standard Configurations

At present, four Standard Configurations are envisaged. The following sections describe these configurations.

3.1 Level 0 Standard Configuration

This follows the ARMv7-A architecture. The following features must be present:

- VFPv3-D32 with
  - hardware implementation of both single-precision and double-precision,
  - hardware support for all unusual cases (NANs, Denormals, and so on)
- Advanced SIMD Extension v1
- Security Extensions
- 16-bit and 32-bit Thumb instructions
- VMSAv7 covering an address range of at least 4GB.
- Performance Monitors with the following features:
  - A minimum of two event counters
  - At least the following events:
    - 0x00, Software increment
    - 0x03, Level 1 data cache refill
    - 0x04, Level 1 data cache access
    - 0x10, Mispredicted or not predicted branch speculatively executed branch
    - 0x12, Predictable Branch speculatively executed
  - An interrupt routed to the IRQ interrupt

Note: See section 3.4.1

Instruction cache handling:

- The Instruction Cache is permitted to be the basic Instruction Cache (compatible with an ASID-Tagged VIVT implementation). Some software environments will require the IVIPT Extensions. These remain outside the scope of the Standard Configurations. Any requirement for the IVIPT Extensions is deprecated.

Many features are *not* included in the Level 0 Standard Configuration. Features can be omitted for several reasons:

- If a feature is deprecated.
- If a feature may be present but should not be used or its use should be limited to a particular subset.
- If it is IMPLEMENTATION DEFINED whether a feature is present or not.
- If a feature is OPTIONAL and can appear in an implementation but cannot be used by software consistent with the Standard Configuration.
- If a feature can be included in another Standard Configuration.

The following subsections give more information about omitted features.

3.1.1 Features not included in any Standard Configuration

The following features are not included in any Standard Configuration:

- The Fast Context Switch Extension. Software consistent with the Level 0 Standard Configuration must not set the FCSEIDR to a non-zero value. An implementation that is consistent with the Level 0 Standard Configuration might treat this register as RAZ/WI.
- VFP Short Vector operation. Software consistent with the Level 0 Standard Configuration must not set the FPSCR Len or Stride fields to non-zero values. An implementation that is consistent with the Level 0 Standard Configuration might treat all floating-point instructions affected by the Short Vector operation as UNDEFINED if the FPSCR Len or Stride fields are non-zero, consistent with their being handled in software.
- The SWP and SWPB instructions. Software consistent with the Level 0 Standard Configuration must not use the SWP or SWPB instructions. An implementation that is consistent with the Level 0 Standard Configuration might treat these instructions as UNDEFINED.

Note: An Operating System consistent with Level 0 Standard Configuration cannot guarantee that SWP and SWPB instructions are never used. The Operating System must provide a software emulation of the SWP and SWPB instructions as part of the Undefined Instruction exception handler.

- Hardware management of the Access Flag. Software consistent with the Level 0 Standard Configuration must not set the SCTRL.HA to 1. An implementation that is consistent with the Level 0 Standard Configuration might treat this bit as RAZ/WI.
- The NSACR.RFR bit. Software consistent with the Level 0 Standard Configuration must not set the NSACR.RFR bit to 1. An implementation that is consistent with the Level 0 Standard Configuration might treat this bit as RAZ/WI.
- Direct Vector handling of Interrupts. Software consistent with the Level 0 Standard Configuration must not set the SCTLR.VE bit to 1. An implementation that is consistent with the Level 0 Standard Configuration might treat this bit as RAZ/WI. The PMSA is not part of the ARMv7-A architecture, and software consistent with the Level 0 Standard Configuration must not assume the existence of the PMSA.
- ThumbEE. Software consistent with the Level 0 Standard Configuration must not use any ThumbEE instructions, including the ENTERX and LEAVEX instructions. An implementation that is consistent with the Level 0 Standard Configuration might not provide these instructions.
- VFP Traps. Software consistent with the Level 0 Standard Configuration must not set the FPSCR IDE, IXE, UFE, OFE, DZE or IOE bits to 1. An implementation that is consistent with the Level 0 Standard Configuration might treat these bits as RAZ/WI.
- Data cache maintenance by Set/Way. Software consistent with the Level 0 Standard Configuration must not use the cache maintenance operations by Set/Way (DCISW, DCCSW, and DCCISW) for coherency management with non-coherent observers. An implementation that is consistent with the Level 0 Standard Configuration (or with a subsequent configuration) might include

features, such as multiprocessor handling or the use of virtualization, that make these instructions poorly suited to coherency management.

Any implementation consistent with a Level 0 Standard Configuration might require Hardware Specific Software to provide maintenance and synchronization of caches that lie between the processor and the point of coherency, but such a requirement is deprecated.

3.1.2 Optional and IMPLEMENTATION DEFINED features not included in any level of the standard configurations

These features might be present in an implementation but must not be included in any Standard Configuration listed in this document:

- Any dependence on a particular implementation of Jazelle DBX, either a trivial implementation or a more advanced implementation. Software that is consistent with a Standard Configuration must not rely on any particular behavior associated with the Jazelle extension.
- Tightly-coupled memory, DMA or lockdown support. Software that is consistent with the Level 0 Standard Configuration must not rely on the presence of any IMPLEMENTATION DEFINED tightly-coupled memory, DMA or lockdown features provided by an implementation. Other implementations might not provide those features.
- CPACR.D32DIS, NSACR.D32DIS. Software consistent with the Level 0 Standard Configuration must not set these bits to 1. An implementation that is consistent with the Level 0 Standard Configuration might treat these bits as RAZ/WI.
- The IMPLEMENTATION DEFINED remapping of memory attributes when the MMU is disabled. Software consistent with the Level 0 Standard Configuration must not rely on any IMPLEMENTATION DEFINED behaviors caused by the remapping of memory attributes provided by an implementation. Other implementations might not provide those features.
- Parity or ECC detection within the caches. The provision of Parity or ECC error detection within caches is not required as part of the Level 0 Standard Configuration. Any specific software support for handling errors is outside the scope of the Level 0 Standard Configuration. Software should provide either a fatal error in the event of encountering these errors, or provide a software-loadable mechanism that allows the installation of Hardware Specific Software that can provide the diagnosis of such errors.
- The Auxiliary Control Register, Auxiliary Data Fault Status Register, Auxiliary Instruction Fault Status Register, and Auxiliary ID Register. Software consistent with the Level 0 Standard Configuration must not set any values in these registers, and must not interpret any particular values read from those registers. Implementations are not required to provide any consistency in the meaning or effect of bits of fields within these registers.
- Any IMPLEMENTATION DEFINED values in the DFSR or IFSR. The provision of implementation-specific fault mechanisms is not required as part of the Level 0 Standard Configuration. Any specific software support for handling faults is outside the scope of the Level 0 Standard Configuration. Software should provide either a fatal error in the event of encountering these errors, or provide a software-loadable mechanism that allows the installation of Hardware Specific Software that can provide the diagnosis of such errors.
- Any coprocessors not defined in the *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition*. Software consistent with the Level 0 Standard

Configuration must not rely on any such undefined coprocessors, and must not set any of CPACR bits [27:24,19:0] to 1, or set any of NSACR bits [13:12, 9:0] to 1. An implementation that is consistent with the Level 0 Standard Configuration might treat those bits as RAZ/WI and might not implement any of coprocessors {cp0-cp9, cp12, cp13}.

- Any IMPLEMENTATION DEFINED functionality accessed through CP15 with CRn==15. Software consistent with the Level 0 Standard Configuration must not rely on any implementation-specific functionality accessed through CP15 with CRn==15. Implementations are not required to provide any consistency in the meaning or effect of registers in this space.
- Implementation-specific performance monitoring functionality. Software consistent with the Level 0 Standard Configuration must not use the implementation-specific performance monitoring events or any IMPLEMENTATION DEFINED performance monitoring functionality. Different implementations consistent with the Level 0 Standard Configuration might not provide the same implementation-specific performance monitoring events or functionality
- A Low interrupt latency configuration. Software consistent with the Level 0 Standard Configuration must not set the SCTLR.FI bit to 1. An implementation that is consistent with the Level 0 Standard Configuration might treat this bit as RAZ/WI.
- Any use of the 0b110 TEX Remap encoding. Software consistent with the Level 0 Standard Configuration that sets SCTR.TRE to 1 must not use any translation table entries with {TEX[0]==1, C==1, B==0}. The meaning of this encoding is IMPLEMENTATION DEFINED, and therefore different implementations might use this encoding in different ways.
- SCTLR.RR. Software consistent with the Level 0 Standard Configuration must not program the SCTLR.RR bit to 1. An implementation that is consistent with the Level 0 Standard Configuration might treat this bit as RAZ/WI.

Note: Software consistent with the Level 0 Standard Configuration must not rely on any particular cache replacement algorithm.

- SCR.nET. Software consistent with the Level 0 Standard Configuration must not program the SCR.nET bit to 1. An implementation that is consistent with the Level 0 Standard Configuration might treat this bit as RAZ/WI.
- NMFI. A system that is consistent with the Level 0 Standard Configuration must not cause the SCTLR.NMFI bit to be set to 1. An implementation that is consistent with the Level 0 Standard Configuration might not provide any mechanism by which the SCTLR.NMFI bit can be set to 1.
- **CP15SDISABLE**. Systems consistent with the Level 0 Standard Configuration must not assert the **CP15SDISABLE** input HIGH. An implementation that is consistent with the Level 0 Standard Configuration might not provide any mechanism by which the **CP15SDISABLE** input can be asserted HIGH.
- Supersections that provide access to a Physical Address range larger than 4GB. Software consistent with the Level 0 Standard Configuration must not use first level translation table descriptors that describe supersections with the fields [23:20] and [8:5] holding non-zero values. An implementation that is consistent with the Level 0 Standard Configuration might ignore any values programmed in those fields.

3.1.3 Optional and Implementation Defined Features not included in Level 0 but included in at least one higher level Standard Configuration

These features might be present in an implementation but must not be included in a Level 0 Standard Configuration. They are included in at least one higher level standard configuration defined in this document:

- The Half-precision Extension. Software consistent with the Level 0 Standard Configuration must not use any instructions described as part of the Half-precision Extension. An implementation that is consistent with the Level 0 Standard Configuration might treat these instructions as UNDEFINED.
- The `SDIV` and `UDIV` instructions. Software consistent with the Level 0 Standard Configuration must not use the `SDIV` or `UDIV` instructions. An implementation that is consistent with the Level 0 Standard Configuration might treat these instructions as UNDEFINED.
- CPACR.ASEDIS, NSACR.ASEDIS. Software consistent with the Level 0 Standard Configuration must not set these bits to 1. An implementation that is consistent with the Level 0 Standard Configuration might treat these bits as RAZ/WI.
- Distinction between Inner and Outer Shareable. Software consistent with the Level 0 Standard Configuration must not program any regions to be Inner Shareable, by setting the PRRR NOS fields to 1, or programming the TTBR0 or TTBR1 NOS fields to 1. An implementation that is consistent with the Level 0 Standard Configuration might treat these fields as RAZ/WI.
- Memory-mapped access to Debug Registers, and CP14 access to Debug Registers for which v7 Debug does not require support for CP14 accesses. Software consistent with the Level 0 Standard Configuration must not use the memory-mapped interface to the Debug Registers, and must not use the CP14 interface to access any Debug register unless the *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition* specifies that v7 Debug must provide CP14 access to that register. An implementation that is consistent with the Level 0 Standard Configuration might not provide memory mapped access to those registers, or might not provide CP14 access to these registers. Where software requires access to this functionality, that access is outside the scope of the Level 0 Standard Configuration, and must be provided in some implementation-specific manner.

3.1.4 Security Extensions and Software Views

The Level 0 Standard Configuration includes the Security Extensions, that provide hardware support for security. This introduces two distinct views of the hardware. These two views are:

- The *Single Machine* view. This is the functionality that can be relied upon by software running in a single security state.
- The *Hardware* view. This is the full functionality provided by the hardware implementation that is consistent with the Level 0 Standard Configuration.

The definition of these views means the Single Machine view provides a standard specification of the features an Operating System that is intended to run in a single security state can rely on, regardless of whether it runs in the Non-secure state or in the Secure state. Such a standard specification is useful because:

- Commonly, implementations that include the Security Extensions have an Operating System running in the Non-secure state, and have software provided by a different provider running in the Secure state.

- The Single Machine view enhances portability by providing a standard specification for functionality that Operating Systems can rely on, regardless of their target security state. This means two different Operating Systems, each corresponding to the Single Machine view, could be used on an implementation that is consistent with the Level 0 Standard Configuration. One of these Operating Systems would run in the Secure state, and the other in the Non-secure state.
- Where no security services are required, a single Operating System, consistent with the Single Machine view of the Level 0 Standard Configuration, can be run in the Secure state.

The software consistent with the Single Machine view of the Level 0 Standard Configuration can use all of the hardware functionality described in the Level 0 Standard Configuration with the following exceptions:

- The SCR, SDER and NSACR registers.
- The A12NSO* address translation operations.

Note: These address translation operations are part of the Virtualizer view described in section 3.4.1.

- `MSR` or `CPS` instructions that change into Monitor mode
- Any execution in Monitor mode

Software consistent with the Single Machine view of the Level 0 Standard Configuration can be designated as either *Secure* or *Non-secure*. The only difference between these two designations is whether the software is able to access the Secure physical address map, by the use of the NS bit within the translation tables:

- Software designated as Secure can access the Secure physical address map. The NS bit in the translation table descriptors identifies which address map is accessed.
- Software designated as Non-secure cannot access the Secure physical address map, regardless of the value of the NS bit in the translation table descriptors.

3.2 Level 1 Standard Configuration

The Level 1 Standard Configuration builds on the Level 0 Standard Configuration by adding the following features to those defined in Section 3.1

- The Multiprocessing extensions.
- CPACR.ASEDIS and NSACR.ASEDIS

Note*:* The CPACR.D32DIS and NSACR.NSD32DIS fields are not included in any Standard Configuration.

- The ARM Generic Interrupt Controller, as defined in the *ARM Generic Interrupt Controller Architecture Specification*, with:
  - Eight Non-secure Software Generated Interrupts
  - Eight Secure Software Generated Interrupts.

Note*:* The number of PPIs and SPIs are system dependent, and come within the scope of the Hardware Specific Software.

- The following additional debug features:
  - CP14 access to all registers that the *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition* specifies must be accessible, in a v7 Debug implementation, through at least one of the CP14 interface and the memory-mapped interface.
  - At least two synchronous watchpoints with the following features:
    - Support for the watchpoint address mask.
    - Support for a 4 bit Byte address select (BAS) field.
    - The watchpoints need not match on:
      - Failed Store-Exclusive instructions
      - Memory Hint instructions
      - Cache Maintenance operations by address.
  - At least two breakpoints
    - Support for the breakpoint address mask is not required, and software that is compliant with the Level 1 Standard Configuration must not attempt to use a nonzero breakpoint address mask value.

Note: The provision of debug functionality for use by system software implies that implementations should be incorporated into systems with Debug Enabled.

The following specific features are not included in the Level 1 Standard Configuration:

- CP14 access to ETM registers. Software consistent with the Level 1 Standard Configuration must not use the CP14 interface to access any ETM registers. An implementation that is consistent with the Level 1 Standard Configuration might not implement those registers.
- OS Save and Restore registers. Software consistent with the Level 1 Standard Configuration must not use the OS Save and Restore registers (DBGOSLAR, DBGOSLSR, and DBGOSSRR). An implementation that is consistent with the Level 1 Standard Configuration might not implement these registers.
- DBGPRCR.CORENPDRQ**.** Software consistent with the Level 1 Standard Configuration must not use the DBGPRCR.CORENPDRQ bit. An implementation that is consistent with the Level 1 Standard Configuration might not implement this bit.

Note: In Revision B of the *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition*, this bit was called the DBGnoPWRDWN bit.

The Level 1 Standard Configuration is the minimum that can be used for a Multiprocessing System.

3.3 Level 2 Standard Configuration

The Level 2 Standard Configuration builds on the Level 1 Standard Configuration by adding the following features to those defined in Section 3.2

- Half-precision Extensions for VFPv3.
- Two shareability domains.
- The ARM Generic Timer, implemented using CP15 register 14 encodings.

An implementation that is consistent with the Level 2 Standard Configuration must not rely on Hardware Specific Software to provide maintenance and synchronization of any cache that lies between a processor and the point of coherency.

3.4 Level 3 Standard Configuration

The Level 3 Standard Configuration builds on the Level 2 Standard Configuration by requiring the following features in addition to those defined in Section 3.3

- The VFPv4-D32 and Advanced SIMDv2 Extensions, including:
  - The Half-precision Extensions
  - Support for Fused Multiply Accumulate.
- The Large Physical Address Extension.

***Note**:* The actual size of physical address supported is outside the scope of the Standard Configurations.

- `UDIV` and `SDIV` instructions in both ARM and Thumb instruction sets.
- The Virtualization Extensions.
- Support for the handling of virtual interrupts in the ARM Generic Interrupt Controller implementation.
- The Performance Monitors Extension must implement PMUv2, and must support the following additional features:
  - Event 0x08, Instruction architecturally implemented.
  - Event 0x11, Cycle.
  - A minimum of four event counters.

3.4.1 Virtualization Support and Software Views

The Level 3 Standard Configuration includes hardware support for Virtualization, and this adds another view. This means the views are:

- The *Single Machine* view, which is backwards compatible with the Single Machine view of the Standard Configuration lower levels.
  - The Non-secure Single Machine view is the functionality presented to a Guest Operating system by a hypervisor.
- The *Virtualizer* view, which is the view that is presented to the hypervisor. This view only uses the Non-secure state, and uses the functionality added in Level 3 of the Standard Configuration that provides hardware support of Virtualization.
- The *Hardware* view, which is the full functionality provided by the hardware implementation that is consistent with the Level 3 Standard Configuration.

The separation of these views provides a standard specification of the features that can be relied upon by an Operating System that is intended to run in an environment that might, or might not, include a hypervisor, as well as a standard specification of the features that can be relied upon by a hypervisor.

Note*:* These views do not constitute different levels of the Standard Configurations.

Future levels of the Standard Configurations are likely to provide the same views, and compatibility requirements mean that a future level 4 Single Machine view will be

backwards compatible with the level 3 Single Machine view, but not with the level 3 Hardware view.

The following subsections give more information about these views, in a Level 3 Standard Configuration.

Single Machine View

The Single Machine view of the Level 3 Standard Configuration presents the functionality of the Non-secure PL1 and PL0 modes defined in the Level 3 Standard Configuration. It provides a minimum of two Performance Monitor event counters.

Software consistent with the Single Machine view of the Level 3 Standard Configuration must not make any use of Hyp mode, or of functionality that is controlled from Hyp mode, and must not rely on having more than two Performance Monitor event counters.

For software consistent with the Single Machine view of the Level 3 Standard Configuration all calls into Hyp mode using the HVC instruction must be provided by Hardware Specific Software that is aware of the presence, or absence, of a hypervisor that can handle those calls.

Note: The Standard Configurations do not specify what functionality is trapped to and managed by a hypervisor using the virtualization facilities provided by the Hardware or Virtualizer view. The extent to which such trapping is required is determined by the needs and functionality of a hypervisor. This trapping is permitted provided it meets the performance constraints described in section 2.2.1

Virtualizer View

The Virtualizer View of the Level 3 Standard Configuration presents all of the functionality of the Non-secure state defined in the Level 3 Standard Configuration, and makes no use of any of the Secure state functionality defined in the Level 3 Standard Configuration.

4 Glossary

The following table describes some of the terms used in this document.

Table 2 Glossary terms

Term	Description
FCSE	Fast Context Switch Extension
FSCEIDR	FCSE Process ID Register
NaN	Not a Number
FPSCR	Floating Point Status and Control Register
MMU	Memory Management Unit
ECC	Error Correcting Code