HVC Calling Conventions

System Software on ARM®

Document number: ARM DEN 0020

HVC Calling Conventions
System Software on ARM®

Release information

The *Change history* table lists the changes made to this document.

Table 1 Change history

Date	Issue	Confidentiality	Change
26 January 2012	A	Confidential	First release

Proprietary notice

Confidentiality status

ARM web address

http://www.arm.com

Table of Contents

1 Introduction 4
1.1 Scope 4
1.2 Additional reading 4
2 Overview 5
3 Condensed HVC calling convention 6
3.1 HVC immediate value, `<imm16>` 6
3.2 Call Parameters 6
3.3 Return Result 6
3.4 Persistent register state over HVC call 7
3.5 Hyper-call 7
3.6 Hyper-call function number ranges 8
3.7 Hyper-call I/D discovery mechanism 9
4 Standard Platform Hyper-calls 11
5 ARM CPU/System hyper-calls 12

1 Introduction

This document describes a set of software conventions for Hyper-calls to be used with the ARM virtualization extensions.

A Hyper-call is made by the use of the HVC instruction.

1.1 Scope

This document is applicable to all 32-bit ARM systems with the Virtualization Extensions.

1.2 Additional reading

This section lists publications by ARM and by third parties.

The following documents contain information relevant to this document:

- *ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition* (ARM DDI 0406)
- *Virtualization Extensions Architecture Specification* (ARM GENC008353)
- *RFC 4122 – A Universally Unique Identifier (UUID) URN Namespace*
- *Procedure Call Standard for the ARM Architecture* (ARM IHI 0042)

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

2 Overview

The ARM virtualization extensions provide a new domain for software configuration.

This document covers a number of areas for standardization.

- HVC calling convention
- Hyper-call function numbering
- Hyper-call ID mechanism
- Standard Platform Hyper-calls

3 Condensed HVC calling convention

A simple synchronous HVC calling convention is presented here to enable basic system calls and the identification of services offered by Hypervisor layer software.

This calling convention is designed to be compatible with AAPCS, see:

- Procedure Call Standard for the ARM Architecture. ARM IHI 0042D

Complex hypervisors are anticipated to provide richer HVC calling conventions that are a superset of this one. They are expected to marshal complex arguments in conformance to the AAPCS and provide asynchronous capabilities.

3.1 HVC immediate value, `<imm16>`

The 16-bit immediate value, `<imm16>`, encoded in HVC instruction is ignored by this call convention.

Function numbers are passed by value in register R0.

By default the constant value of 0x0000 should be encoded into each HVC instruction.

Making use of runtime discovered functions is simplified, by passing the value of the function number in a register, and calling a common HVC instruction. Conversely a calling convention that did use the immediate value would require the dynamic creation of HVC instructions at runtime.

3.2 Call Parameters

Parameters to the HVC are passed in the first four registers. The condensed calling convention only uses fundamental data types of 4-byte or smaller or 32-bit pointers to data within the callers address space.

Table 2: Register parameters

Register	Meaning
`R0`	Hyper-call function number, see Table 4
`R1`	1st parameter: 4, 2 or 1 byte fundamental data type or a 32-bit pointer.
`R2`	2nd parameter: 4, 2 or 1 byte fundamental data type or a 32-bit pointer.
`R3`	3rd parameter: 4, 2 or 1 byte fundamental data type or a 32-bit pointer.

3.3 Return Result

The return results are passed in registers R0-R3, depending upon the size of the return data type.

Table 3: Return results

Register	Value	Meaning

R0	0x00000000 to 0xFFFFFFFE	Hyper-call return value, 32-bit integer. See each calling function for return result definition.
R0	0xFFFFFFFF	Error: Invalid call number or request.
R1-R3		Call specific return data.

3.4 Persistent register state over HVC call

Registers R0-R3 are not required to be preserved over the HVC call, and may contain return results.

Registers R4-R11 and SP (R13) must be preserved over the HVC call.

All other registers R12,R14 and R15 may be modified.

3.5 Hyper-call

Simple assembler functions with C interfaces can be used as wrappers for the HVC, as this calling convention interworks with the AAPCS.

Function call number and input parameters are passed in R0-R4, and any return values from the HVC will remain R0-R3 as control is passed back up to the caller.

```
/* Example 32bit HVC call with three integer parameters, and 32-bit
return value*/

_asm int HVC_call_3(int fn, int p1, int p2, int p3)

{

       HVC #0

       BX  lr

}
```

The same code in GCC format:

```
int __attribute__((naked)) HVC_call_3(int fn, int p1, int p2, int p3)
{
       asm ("HVC #0;"
       "BX   lr");

}
```

3.6 Hyper-call function number ranges

The hyper-call function number range is divided into sub ranges for the varying service owners.

Obviously not all of these will be present in every device.

Table 4: Hyper-call function number

HVC Function Number Range	Owner	Notes
`0x0000 0000 –` `0x7FFF FFFF`	Hypervisor	Rich hypervisor interface
`0x8000 0000 –` `0x81FF FFFF`	Standard Platform Hyper-calls	Vendor neutral hyper-calls * Hypervisor Installation
`0x8200 0000 –` `0x83FF FFFF`	RESERVED	For future use
`0x8400 0000 –` `0x85FF FFFF`	SIP	SoC specific calls
`0x8600 0000 –` `0x87FF FFFF`	ODM	ODM calls
`0x8800 0000 –` `0x89FF FFFF`	OEM	OEM calls
`0x8A00 0000 –` `0x8FFF FFFF`	RESERVED	For future use
`0x9000 0000 –` `0x91FF FFFF`	ARM	ARM CPU/System hyper-calls * ID mechanism * ARM CPU Specific calls * ARM System IP Specific calls * ARM Errata
`0x9200 0000 –` `0xFFFF FFFF`	RESERVED	For future use

The lowest value in each range section is referred to as the `RangeBase` and the highest as `RangeLimit`.

3.7 Hyper-call I/D discovery mechanism

Hyper-call numbers for identification and discovery are defined at the top of each region.

Table 5: Hyper-call I/D numbers

HVC Function Call	HVC Handler Meaning	Return value
`RangeBase + 0x01FF FF00`	Hyper-call Count	32-bit value. Maximum number of hyper-calls implemented by this handler. Excluding the 16 numbers in the ID mechanism call space.
`RangeBase + 0x01FF FF01`	UUID	128-bit return value. Hyper-call interface UUID
`RangeBase + 0x01FF FF02`	Revision Number	Concatenated 64-bit return value. R0 = Major Revision Number R1 = Minor Revision Number

`RangeBase + 0x01FF FF03`
`to`
`RangeBase + 0x01FF FFFF`
	Reserved for future expansion	

3.7.1 Invalid Return Value

Out of range and not implemented HVC calls will return the values of 0xFFFF FFFF in registers R0-R3.

3.7.2 Hyper-call range

Hyper-calls are expected to be defined sequentially within this region.

For backwards compatibility hyper-call holes that are no longer active functions will return the Invalid HVC value.

3.7.3 Hyper-call count

This call returns the maximum hyper-call function number offset from the base of this region.

Counting up from the base hyper-call number, `RangeBase`.

3.7.4 UUID

This 128-bit value shall identify the owner of this hyper-call region.

Creation of UUID are defined by RFC 4122.

UUIDs with 0xFFFFFFFF as the highest 32 bits should be avoided as they clash with the Invalid Return Value.

3.7.5 Revision Number

Revision information shall be defined by two 32-bit values, for major and minor revision of this hyper-call interface.

Major version number changes may result in incompatible HVC calls.

Minor revision changes are expected to be runtime compatible.

The major version number will be in the most significant bytes of the 64-bit returned value.

4 Standard Platform Hyper-calls

Proposed common hyper-calls

Table 6: Common hyper-calls

HVC Function Number Range	Name	Operation
`0x8000 0000`	Install Hypervisor	Install new Hypervisor that is not signed or validated. New Hypervisor image must be resident in Contiguous IPA address space. R1 = Image address (IPA) R2 = Size in Bytes R3 = Image Entry Point
–	-	-
`0x81FF FF00`	Hyper-call Count	0x0000 0001
`0x81FF FF01`	UUID	128-bit return MAGIC VALUE (TBD)
`0x81FF FF02`	Revision Number	Major = 0x0000 0001 Minor = 0x0000 0001

Standard platform calls that are not Hyper-calls:

- SMC interface for the power control of a processor.
  For the controlled shutdown of a processor core, and programming of its wakeup reasons.

5 ARM CPU/System hyper-calls

Proposed common hyper-calls

Table 7: System hyper-calls

HVC Function Number Range	Name	Operation
0x9000 0000	HVC_SWITCHER_CLUSTER_SWITCH	Switch payload software execution between the two clusters in a big.LITTLE system
…	…	…
0x9000 0100	HVC_VIRT_MPIDR_READ	Read the Physical MPIDR register
…	…	…
0x9000 1000	HVC_PMU_PMCR_READ	Read PMU.PMCR register
0x9000 1001	HVC_PMU_PMCR_WRITE	Write PMU.PMCR register
0x9000 1002	HVC_PMU_PMSELR_READ	Read PMU.PMSELR register
0x9000 1003	HVC_PMU_PMSELR_WRITE	Write PMU.PMSELR register
0x9000 1004	HVC_PMU_PMXEVTYPER_READ	Read PMU.PMXEVTYPER register
0x9000 1005	HVC_PMU_PMXEVTYPER_WRITE	Write PMU.PMXEVTYPER register
0x9000 1006	HVC_PMU_PMCNTENSET_READ	Read PMU.PMCNTENSET register
0x9000 1007	HVC_PMU_PMCNTENSET_WRITE	Write PMU.PMCNTENSET register
0x9000 1008	HVC_PMU_PMCNTENCLR_READ	Read PMU.PMCNTENCLR register
0x9000 1009	HVC_PMU_PMCNTENCLR_WRITE	Write PMU.PMCNTENCLR register
0x9000 100A	HVC_PMU_PMCCNTR_READ	Read PMU.PMCCNTR register
0x9000 100B	HVC_PMU_PMCCNTR_WRITE	Write PMU.PMCCNTR register
0x9000 100C	HVC_PMU_PMOVSR_READ	Read PMU.PMOVSR register
0x9000 100D	HVC_PMU_PMOVSR_WRITE	Write PMU.PMOVSR register
0x9000 100E	HVC_PMU_PMXEVCNTR_READ	Read PMU.PMXEVCNTR register
0x9000 100F	HVC_PMU_PMXEVCNTR_WRITE	Write PMU.PMXEVCNTR register
0x9000 1010	HVC_PMU_PMINTENSET_READ	Read PMU.PMINTENSET register
0x9000 1011	HVC_PMU_PMINTENSET_WRITE	Write PMU.PMINTENSET register

0x9000 1012	HVC_PMU_PMINTENCLR_READ	Read PMU.PMINTENCLR register
0x9000 1013	HVC_PMU_PMINTENCLR_WRITE	Write PMU.PMINTENCLR register
0x9000 1100	HVC_PMU_SWITCH	Triggers PMU Save/Restore/Migration mode changes
0x9000 1200	HVC_PMU_GET_COUNTERS_SIZE	Returns the maximum memory size handled by HVC_PMU_SYNC_PMU_COUNTERS
0x9000 1201	HVC_PMU_SYNC_PMU_COUNTERS	Synchronizes all PMU counters status.
–	-	-
0x91FF FF00	Hyper-call Count	0x0000 1202
0x91FF FF01	UUID	128bit return MAGIC VALUE (TBD)
0x91FF FF02	Revision Number	Major = 0x0000 0001 Minor = 0x0000 0001

ARM CPU/System calls that are not Hyper-calls:

- SMC interfaces for ARM CPU Errata fixes.