Chapter 3
System Control Coprocessor

This chapter describes the purpose of the system control coprocessor, its structure, operation, and how to use it. It contains the following sections:

3.1 About the system control coprocessor

The section gives an overall view of the system control coprocessor. For detail of the registers in the system control coprocessor, see *System control processor registers* on page 3-13.

The purpose of the system control coprocessor, CP15, is to control and provide status information for the functions implemented in the ARM1176JZF-S processor. The main functions of the system control coprocessor are:

- overall system control and configuration
- cache configuration and management
- *Tightly-Coupled Memory* (TCM) configuration and management
- *Memory Management Unit* (MMU) configuration and management
- DMA control
- system performance monitoring.

The system control coprocessor does not exist in a distinct physical block of logic.

3.1.1 System control coprocessor functional groups

The system control coprocessor appears as a set of 32-bit registers that you can write to and read from. Some of the registers permit more than one type of operation. The functional groups for the registers are:

- *System control and configuration* on page 3-5
- *MMU control and configuration* on page 3-6
- *Cache control and configuration* on page 3-7
- *TCM control and configuration* on page 3-8
- *Cache Master Valid Registers* on page 3-8
- *DMA control* on page 3-9
- *System performance monitor* on page 3-10
- *System validation* on page 3-10.

The system control coprocessor controls the TrustZone operation of the processor:

- some of the registers are only accessible in the Secure world
- some of the registers are banked for Secure and Non-secure worlds
- some of the registers are common to both worlds.

Note

When Secure Monitor mode is active the core is in the Secure world. The processor treats all accesses as Secure and the system control coprocessor behaves as if it operates in the Secure world regardless of the value of the NS bit, see *c1, Secure Configuration Register* on page 3-52. In Secure Monitor mode, the NS bit defines the copies of the banked registers in the system control coprocessor that the processor can access:

NS = 0 Access to Secure world CP15 registers

NS = 1 Access to Non-secure world CP15 registers.

Registers that are only accessible in the Secure world are always accessible in Secure Monitor mode, regardless of the value of the NS bit.

Table 3-1 on page 3-3 lists the overall functionality for the system control coprocessor as it relates to its registers.

Table 3-2 on page 3-14 lists the registers in the system control processor in register order and gives their reset values.

Table 3-1 System control coprocessor register functions

Function	Register/operation	Reference to description
System control and configuration	Control	*c1, Control Register* on page 3-44
	Auxiliary control	*c1, Auxiliary Control Register* on page 3-48
	Secure Configuration	*c1, Secure Configuration Register* on page 3-52
	Secure Debug Enable	*c1, Secure Debug Enable Register* on page 3-54
	Non-Secure Access Control	*c1, Non-Secure Access Control Register* on page 3-55
	Coprocessor Access Control	*c1, Coprocessor Access Control Register* on page 3-51
	Secure or Non-secure Vector Base Address	*c12, Secure or Non-secure Vector Base Address Register* on page 3-121
	Monitor Vector Base Address	*c12, Monitor Vector Base Address Register* on page 3-122
	ID code[a]	*c0, Main ID Register* on page 3-20
	Feature ID, CPUID scheme	*c0, CPUID registers* on page 3-26
MMU control and configuration	TLB Type	*c0, TLB Type Register* on page 3-25
	Translation Table Base 0	*c2, Translation Table Base Register 0* on page 3-57
	Translation Table Base 1	*c2, Translation Table Base Register 1* on page 3-59
	Translation Table Base Control	*c2, Translation Table Base Control Register* on page 3-60
	Domain Access Control	*c3, Domain Access Control Register* on page 3-63
	Data Fault Status	*c5, Data Fault Status Register* on page 3-64
	Instruction Fault Status	*c5, Instruction Fault Status Register* on page 3-66
	Fault Address	*c6, Fault Address Register* on page 3-68
	Instruction Fault Address	*c6, Instruction Fault Address Register* on page 3-69
	Watchpoint Fault Address	*c6, Watchpoint Fault Address Register* on page 3-69
	TLB Operations	*c8, TLB Operations Register* on page 3-86
	TLB Lockdown	*c10, TLB Lockdown Register* on page 3-100
	Memory Region Remap	*c10, Memory region remap registers* on page 3-101
	Peripheral Port Memory Remap	*c15, Peripheral Port Memory Remap Register* on page 3-130
	Context ID	*c13, Context ID Register* on page 3-128
	FCSE PID	*c13, FCSE PID Register* on page 3-126
	Thread And Process ID	*c13, Thread and process ID registers* on page 3-129
	TLB Lockdown Access	*c15, TLB lockdown access registers* on page 3-149

Table 3-1 System control coprocessor register functions (continued)

Function	Register/operation	Reference to description
Cache control and configuration	Cache Type	*c0, Cache Type Register* on page 3-21
	Cache Operations	*c7, Cache operations* on page 3-69
	Data Cache Lockdown	*c9, Data and instruction cache lockdown registers* on page 3-87
	Instruction Cache Lockdown	*c9, Data and instruction cache lockdown registers* on page 3-87
	Cache Behavior Override	*c9, Cache Behavior Override Register* on page 3-97
TCM control and configuration	TCM Status	*c0, TCM Status Register* on page 3-24
	Data TCM Region	*c9, Data TCM Region Register* on page 3-89
	Instruction TCM Region	*c9, Instruction TCM Region Register* on page 3-91
	Data TCM Non-secure Access Control	*c9, Data TCM Non-secure Control Access Register* on page 3-93
	Instruction TCM Non-secure Access Control	*c9, Instruction TCM Non-secure Control Access Register* on page 3-94
	TCM Selection	*c9, TCM Selection Register* on page 3-96
Cache Master Valid	Instruction Cache Master Valid	*c15, Instruction Cache Master Valid Register* on page 3-147
	Data Cache Master Valid	*c15, Data Cache Master Valid Register* on page 3-148
DMA control	DMA Identification and Status	*c11, DMA identification and status registers* on page 3-106
	DMA User Accessibility	*c11, DMA User Accessibility Register* on page 3-107
	DMA Channel Number	*c11, DMA Channel Number Register* on page 3-109
	DMA enable	*c11, DMA enable registers* on page 3-110
	DMA Control	*c11, DMA Control Register* on page 3-112
	DMA Internal Start Address	*c11, DMA Internal Start Address Register* on page 3-114
	DMA External Start Address	*c11, DMA External Start Address Register* on page 3-115
	DMA Internal End Address	*c11, DMA Internal End Address Register* on page 3-116
	DMA Channel Status	*c11, DMA Channel Status Register* on page 3-117
	DMA Context ID	*c11, DMA Context ID Register* on page 3-120
System performance monitor	Performance Monitor Control	*c15, Performance Monitor Control Register* on page 3-133
	Cycle Counter	*c15, Cycle Counter Register* on page 3-137
	Count Register 0	*c15, Count Register 0* on page 3-138
	Count Register 1	*c15, Count Register 1* on page 3-139

Table 3-1 System control coprocessor register functions (continued)

Function	Register/operation	Reference to description
System validation	Secure User and Non-secure Access Validation Control	*c15, Secure User and Non-secure Access Validation Control Register* on page 3-132
	System Validation Counter	*c15, System Validation Counter Register* on page 3-140
	System Validation Operations	*c15, System Validation Operations Register* on page 3-142
	System Validation Cache Size Mask	*c15, System Validation Cache Size Mask Register* on page 3-145

a. Returns device ID code.

3.1.2 System control and configuration

The purpose of the system control and configuration registers is to provide overall management of:

- TrustZone behavior
- memory functionality
- interrupt behavior
- exception handling
- program flow prediction
- coprocessor access rights for CP0-CP13.

The system control and configuration registers also provide the processor ID.

The system control and configuration registers consist of three 32-bit read only registers and eight 32-bit read/write registers. Figure 3-1 shows the arrangement of registers in this functional group.

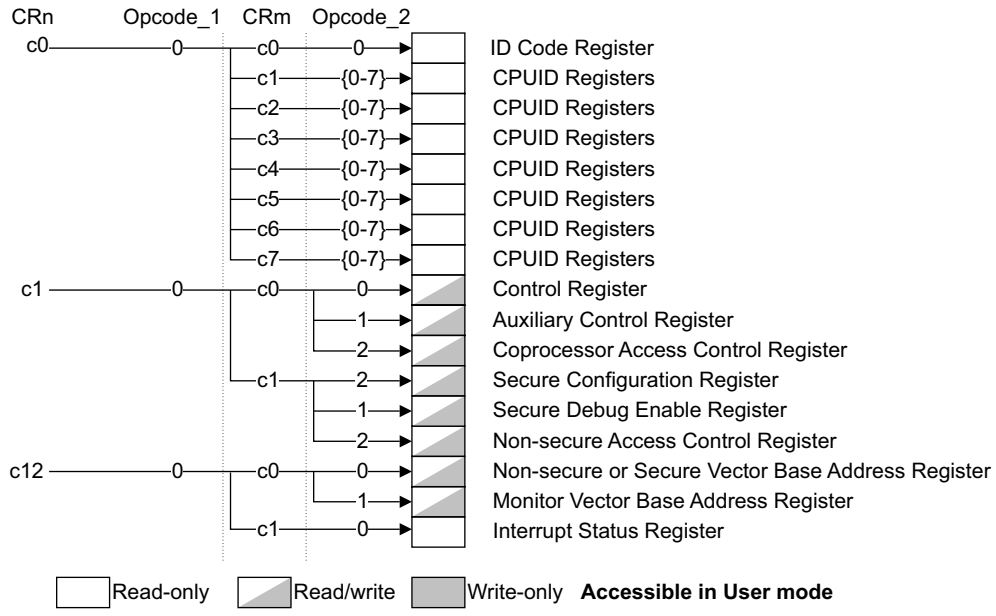

Figure 3-1 System control and configuration registers

To use the system control and configuration registers you read or write individual registers that make up the group, see *Use of the system control coprocessor* on page 3-12.

Some of the functionality depends on how you set external signals at reset.

System control and configuration behaves in three ways:

- as a set of flags or enables for specific functionality
- as a set of numbers, values that indicate system functionality
- as a set of addresses for processes in memory.

3.1.3 MMU control and configuration

The purpose of the MMU control and configuration registers is to:

- allocate physical address locations from the *Virtual Addresses* (VAs) that the processor generates.
- control program access to memory.
- designate areas of memory as either:
  - noncacheable
  - unbufferable
  - noncacheable and unbufferable.
- detect MMU faults and external aborts
- hold thread and process IDs
- provide direct access to the TLB lockdown entries.

The MMU control and configuration registers consist of one 32-bit read-only register, one 32-bit write-only register, and 22 32-bit read/write registers. Figure 3-2 on page 3-7 shows the arrangement of registers in this functional group.

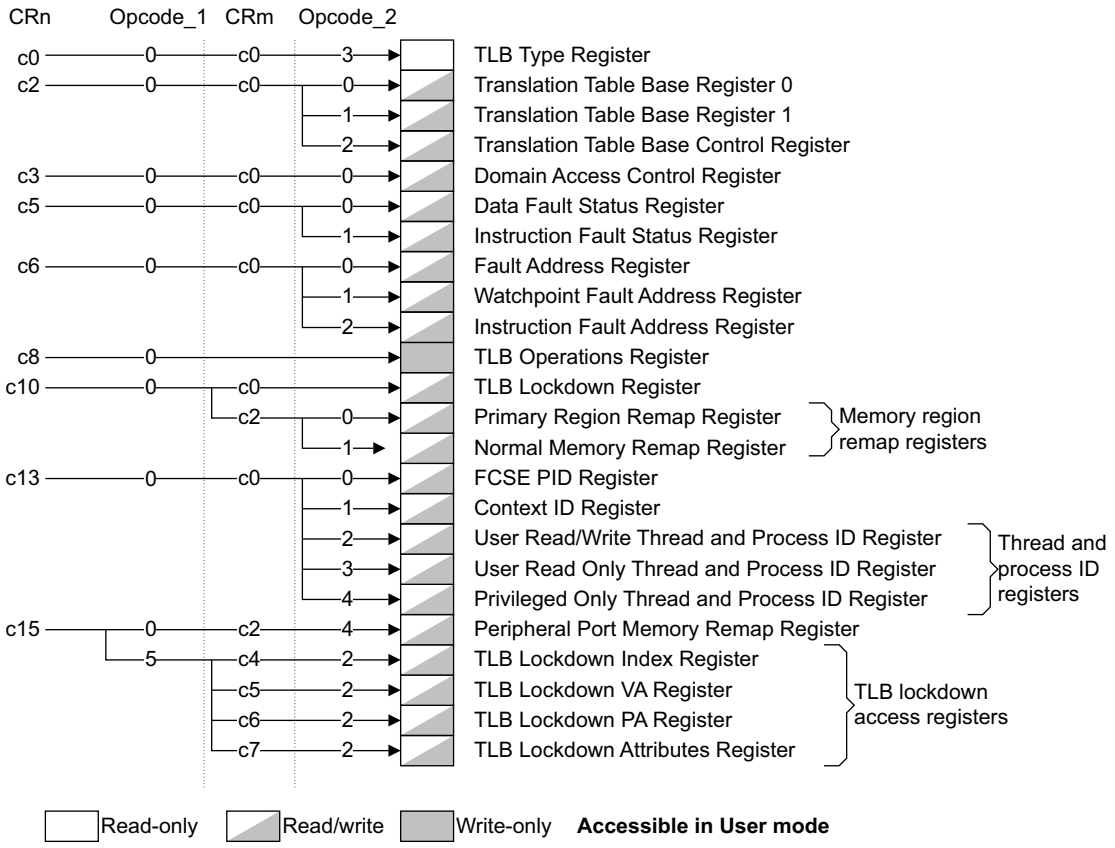

Figure 3-2 MMU control and configuration registers

To use the MMU control and configuration registers you read or write individual registers that make up the group, see *Use of the system control coprocessor* on page 3-12.

MMU control and configuration behaves in three ways:

- as a set of numbers, values that describe aspects of the MMU or indicate its current state
- as a set of addresses for tables in memory
- as a set of operations that act on the MMU.

3.1.4 Cache control and configuration

The purpose of the cache control and configuration registers is to:

- provide information on the size and architecture of the instruction and data caches
- control instruction and data cache lockdown
- control cache maintenance operations that include clean and invalidate caches, drain and flush buffers, and address translation
- override cache behavior during debug or interruptible cache operations.

The cache control and configuration registers consist of one 32-bit read only register and four 32-bit read/write registers. Figure 3-3 on page 3-8 shows the arrangement of the registers in this functional group.

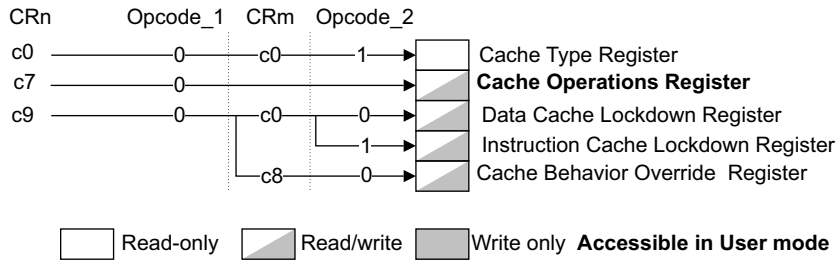

Figure 3-3 Cache control and configuration registers

To use the system control and configuration registers you read or write individual registers that make up the group, see *Use of the system control coprocessor* on page 3-12.

Cache control and configuration registers behave as:

- a set of numbers, values that describe aspects of the caches
- a set of bits that enable specific cache functionality
- a set of operations that act on the caches.

3.1.5 TCM control and configuration

The purpose of the TCM control and configuration registers is to:

- inform the processor about the status of the TCM regions
- define TCM regions.

The TCM control and configuration registers consist of one 32-bit read-only register and five 32-bit read/write registers. Figure 3-4 shows the arrangement of registers.

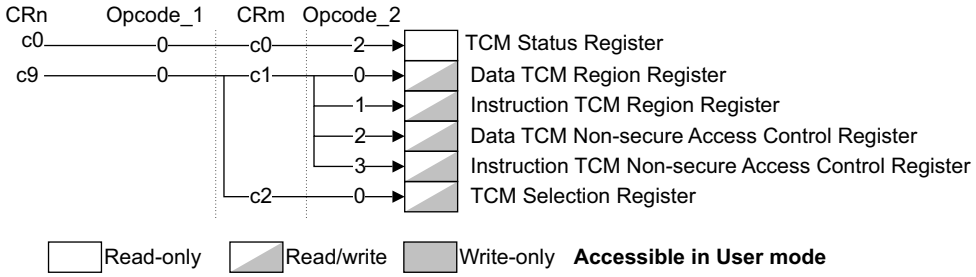

Figure 3-4 TCM control and configuration registers

To use the system control and configuration registers you read or write individual registers that make up the group, see *Use of the system control coprocessor* on page 3-12.

TCM control and configuration behaves in three ways:

- as a set of numbers, values that describe aspects of the TCMs
- as a set of bits that enable specific TCM functionality
- as a set of addresses that define the memory locations of data stored in the TCMs.

3.1.6 Cache Master Valid Registers

The purpose of the Cache Master Valid Registers is to hold the state of the Master Valid bits of the instruction and data caches.

The cache debug registers consist of two 32-bit read/write registers. Figure 3-5 on page 3-9 shows the arrangement of registers in this functional group.

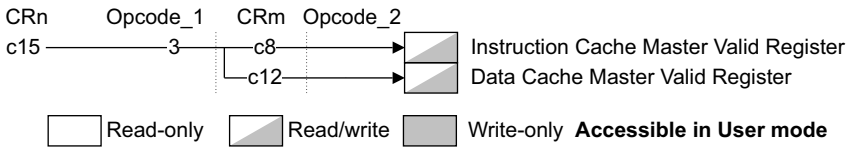

Figure 3-5 Cache Master Valid Registers

To use the Cache Master Valid Registers you read or write the individual registers that make up the group, see *Use of the system control coprocessor* on page 3-12.

The Cache Master Valid Registers behave as a set of bits that define the cache contents as valid or invalid. The number of bits is a function of the cache size.

3.1.7 DMA control

The purpose of the DMA control registers is to:

- enable software to control DMA
- transfer large blocks of data between the TCM and an external memory
- determine accessibility
- select DMA channel.

The Enable, Control, Internal Start Address, External Start Address, Internal End Address, Channel Status, and Context ID Registers are multiple registers with one register of each for each channel that is implemented.

The DMA control registers consist of five 32-bit read-only registers, three 32-bit write-only registers and seven 32-bit read/write registers. Figure 3-6 shows the arrangement of registers.

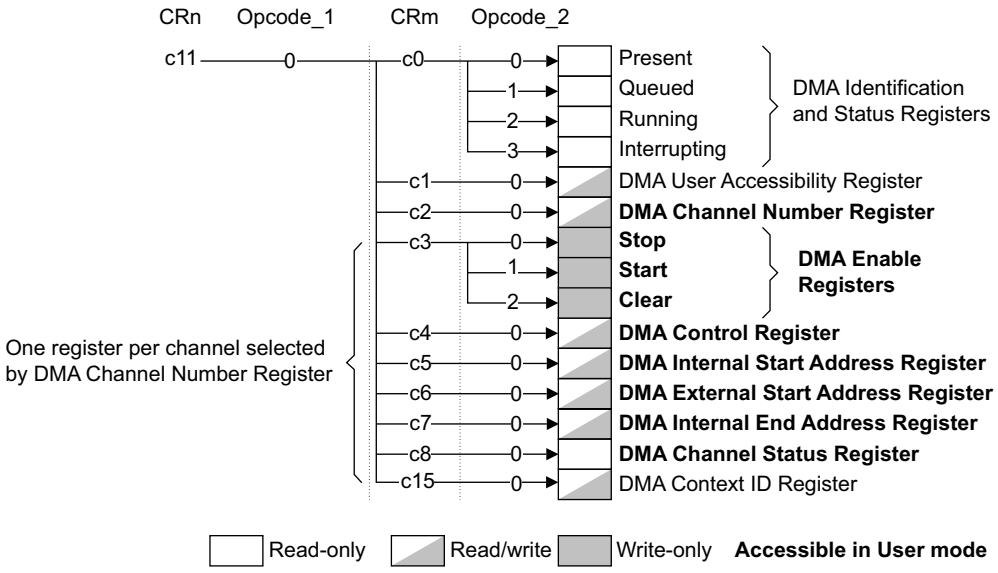

Figure 3-6 DMA control and configuration registers

To use the DMA control and configuration registers you read or write the individual registers that make up the group, see *Use of the system control coprocessor* on page 3-12.

Code can execute several DMA operations while in User mode if these operations are enabled by the DMA User Accessibility Register.

If DMA control registers attempt to execute a privileged operation in User mode the processor takes an Undefined instruction trap.

The DMA control registers operation specifies the block of data for transfer, the location of where the transfer is to, and the direction of the DMA. For more details on the operation see *DMA* on page 7-10.

DMA control behaves in four ways:

- as a set of numbers, values that describe aspects of the DMA channels or indicate their current state
- as a set of bits that enable specific DMA functionality
- as a set of addresses that define the memory locations of data for transfer
- as a set of operations that act on the DMA channels.

3.1.8 System performance monitor

The purpose of the performance monitor registers is to:

- control the monitoring operation
- count events.

The system performance monitor consist of four 32-bit read/write registers. Figure 3-7 shows the arrangement of registers in this functional group.

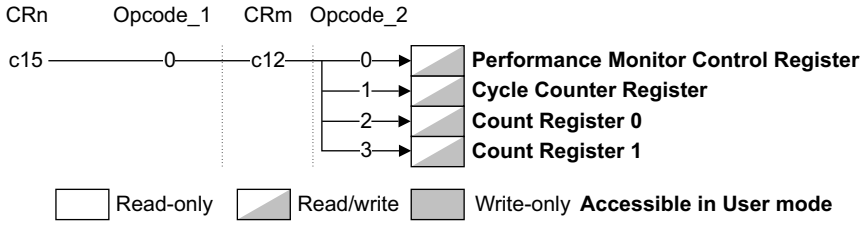

Figure 3-7 System performance monitor registers

To use the system performance monitor registers you read or write individual registers that make up the group, see *Use of the system control coprocessor* on page 3-12.

Note

The counters are only enabled when the **SPNIDEN** input and the SUNIDEN bit, see *c1, Secure Debug Enable Register* on page 3-54, are appropriately set. When the core is in a mode where non-invasive debug is not permitted, events are not counted but the cycle count register, CCNT, continues to count.

You can not use the system performance monitor registers at the same time as the system validation registers, because both sets of registers use the same physical counters. You must disable one set of registers before you start to use the other set. See *System validation*.

System performance monitoring counts system events, such as cache misses, TLB misses, pipeline stalls, and other related features to enable system developers to profile the performance of their systems. It can generate interrupts when the number of events reaches a given value.

3.1.9 System validation

The system validation registers extend the use of the system performance monitor registers to provide some functions for validation and must not be used for other purposes. The system validation registers schedule and clear:

- resets

- interrupts
- fast interrupts
- external debug requests.

The system validation registers consist of four 32-bit read/write registers. Figure 3-8 shows the arrangement of registers.

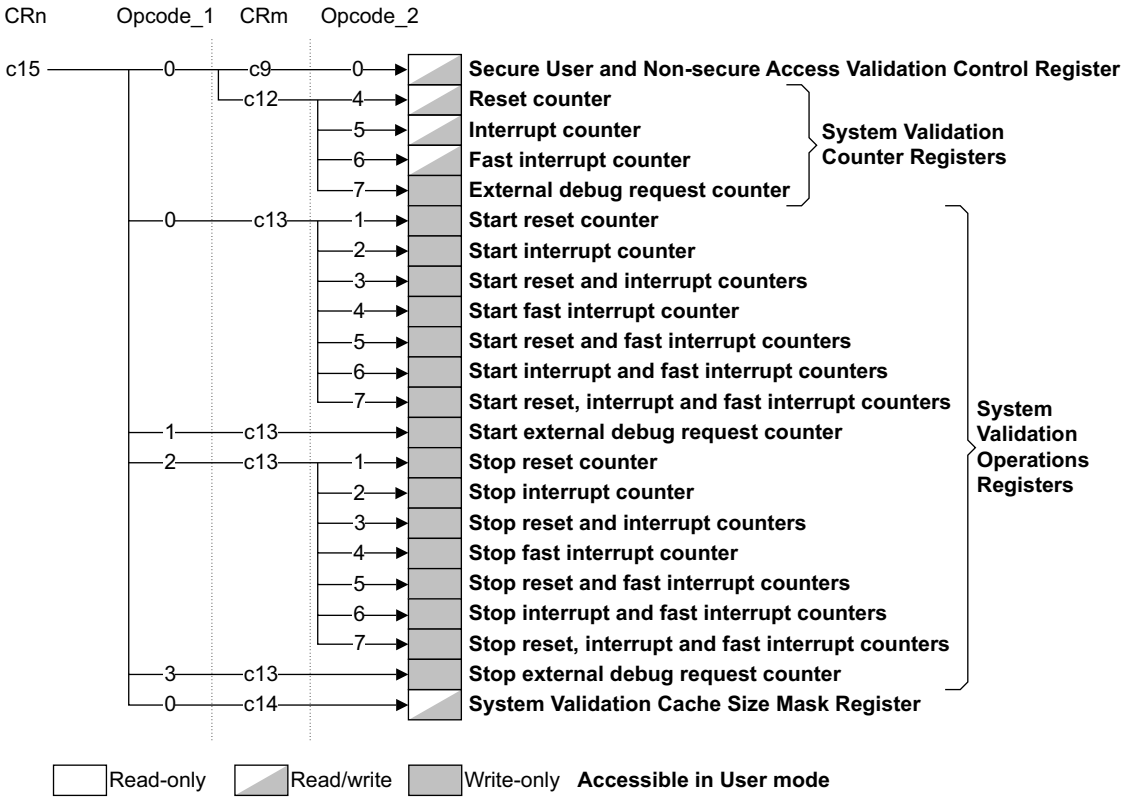

Figure 3-8 System validation registers

The System Validation Counter Register and System Validation Operations Register reuse the Cycle Counter Register, Count Register 0, and Count Register 1, see *System performance monitor* on page 3-10, to schedule resets, interrupts and fast interrupts respectively. External debug requests are scheduled using an additional 6 bit counter that is not used by the System performance monitor registers.

Each of the four counters counts upwards, and when the counter overflows, the corresponding event occurs. To the core, the events are indistinguishable from ordinary external events. The System Validation Registers provide functions for loading the counter registers with the required number of clock cycles before the event occurs, and starting, stopping and clearing the counters, to return them to their System performance monitor functionality.

The System Validation Registers are usually only accessible from Secure privileged modes, but a Secure User and Non-secure Access Validation Control Register is provided to permit access to the System Validation Registers from User modes and Non-secure modes.

The System Validation Cache Size Mask Register masks the physical size of the caches and TCMs to make their size appear different to the processor. You can use this in validation by simulation, but you must not use it in a manufactured device because it can corrupt correct operation of the processor.

To use the system validation registers you read or write individual registers that make up the group, see *Use of the system control coprocessor*.

You cannot use the System Validation Registers at the same time as the System Performance Monitor Registers, because both sets of registers use the same physical counters. You must disable one set of registers before starting to use the other set. See *System performance monitor* on page 3-10.

System validation behaves in three ways:

- as a set of bits that enable specific system validation functionality
- as a set of operations that schedule and clear system validation events
- as a set of numbers, values that describe aspects of the caches and TCMs for system validation.

3.1.10 Use of the system control coprocessor

This section describes the general method for use of the system control coprocessor.

You can access system control coprocessor CP15 registers with MRC and MCR instructions.

```
MCR{cond} P15,<Opcode_1>,<Rd>,<CRn>,<CRm>,<Opcode_2>
MRC{cond} P15,<Opcode_1>,<Rd>,<CRn>,<CRm>,<Opcode_2>
```

Figure 3-9 shows the instruction bit pattern of MRC and MCR instructions.

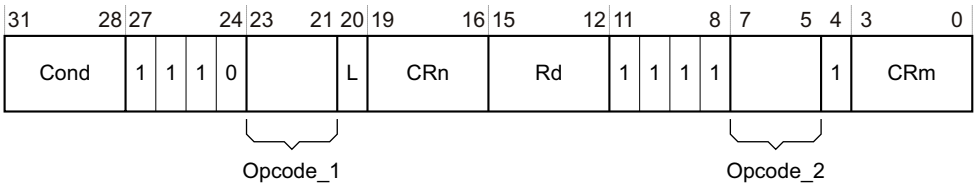

Figure 3-9 CP15 MRC and MCR bit pattern

The CRn field of MRC and MCR instructions specifies the coprocessor register to access. The CRm field and Opcode_2 fields specify a particular operation when addressing registers. The L bit distinguishes between an MRC (L=1) and an MCR (L=0).

Instructions CDP, LDC, and STC, together with unprivileged MRC and MCR instructions to privileged-only CP15 registers, and Non-secure accesses to Secure registers, cause the processor to take the Undefined instruction trap.

Note

Attempting to read from a nonreadable register, or to write to a nonwriteable register causes Undefined exceptions.

The Opcode_1, Opcode_2, and CRm fields Should Be Zero in all instructions that access CP15, except when the values specified are used to select required operations. Using other values results in Undefined exceptions.

In all cases, reading from or writing any data values to any CP15 registers, including those fields specified as *Unpredictable* (UNP), *Should Be One* (SBO), or *Should Be Zero* (SBZ), does not cause any physical damage to the chip.

3.2 System control processor registers

This section gives details of all the registers in the system control coprocessor. The section presents a summary of the registers and detailed descriptions in register order of CRn, Opcode_1, CRm, Opcode_2.

You can access CP15 registers with MRC and MCR instructions:

```
MCR{cond} P15,<Opcode_1>,<Rd>,<CRn>,<CRm>,<Opcode_2>
MRC{cond} P15,<Opcode_1>,<Rd>,<CRn>,<CRm>,<Opcode_2>
```

3.2.1 Register allocation

Table 3-2 on page 3-14 lists the allocation and reset values of the registers of the system control coprocessor where:

- CRn is the register number within CP15
- Op1 is the Opcode_1 value for the register
- CRm is the operational register
- Op2 is the Opcode_2 value for the register.
- Type applies to the Secure, S, or the Non-secure, NS, world and is:
  - B, registers banked in Secure and Non-secure worlds. If the registers are not banked then they are common to both worlds or only accessible in one world.
  - NA, no access
  - **RO**, read-only access
  - RO, read-only access in privileged modes only
  - **R/W**, read/write access
  - R/W, read/write access in privileged modes only
  - **WO**, write-only access
  - WO, write-only access in privileged modes only
  - X, access depends on another register or external signal.

Table 3-2 Summary of CP15 registers and operations

CRn	Op1	CRm	Op2	Register or operation	S type	NS type	Reset value	Page
c0	0	c0	0	Main ID	RO	RO	0x41xFB76x[a]	page 3-20
			1	Cache Type	RO	RO	0x10152152[b]	page 3-21
			2	TCM Status	RO	RO	0x00020002[c]	page 3-24
			3	TLB Type	RO	RO	0x00000800	page 3-25
		c1	0	Processor Feature 0	RO	RO	0x00000111	page 3-26
			1	Processor Feature 1	RO	RO	0x00000011	page 3-27
			2	Debug Feature 0	RO	RO	0x00000033	page 3-29
			3	Auxiliary Feature 0	RO	RO	0x00000000	page 3-30
			4	Memory Model Feature 0	RO	RO	0x01130003	page 3-31
			5	Memory Model Feature 1	RO	RO	0x10030302	page 3-32
			6	Memory Model Feature 2	RO	RO	0x01222100	page 3-33
			7	Memory Model Feature 3	RO	RO	0x00000000	page 3-35
		c2	0	Instruction Set Feature Attribute 0	RO	RO	0x00140011	page 3-36
			1	Instruction Set Feature Attribute 1	RO	RO	0x12002111	page 3-37
			2	Instruction Set Feature Attribute 2	RO	RO	0x11231121	page 3-39
			3	Instruction Set Feature Attribute 3	RO	RO	0x01102131	page 3-40
			4	Instruction Set Feature Attribute 4	RO	RO	0x00001141	page 3-42
			5	Instruction Set Feature Attribute 5	RO	RO	0x00000000	page 3-43
			6-7	Reserved	-	-	-	-
		c3-c7	-	Reserved	-	-	-	-
c1	0	c0	0	Control	R/W, B[d], X	R/W	0x00050078[e]	page 3-44
			1	Auxiliary Control	R/W	RO	0x00000007	page 3-48
			2	Coprocessor Access Control	R/W	R/W	0x00000000	page 3-51
		c1	0	Secure Configuration	R/W	NA	0x00000000	page 3-52
			1	Secure Debug Enable	R/W	NA	0x00000000	page 3-54
			2	Non-Secure Access Control	R/W	RO	0x00000000	page 3-55

Table 3-2 Summary of CP15 registers and operations (continued)

CRn	Op1	CRm	Op2	Register or operation	S type	NS type	Reset value	Page
c2	0	c0	0	Translation Table Base 0	R/W, B, X	R/W	0x00000000	page 3-57
			1	Translation Table Base 1	R/W, B	R/W	0x00000000	page 3-59
			2	Translation Table Base Control	R/W, B, X	R/W	0x00000000	page 3-60
c3	0	c0	0	Domain Access Control	R/W, B, X	R/W	0x00000000	page 3-63
c4				Not used				
c5	0	c0	0	Data Fault Status	R/W, B	R/W	0x00000000	page 3-64
			1	Instruction Fault Status	R/W, B	R/W	0x00000000	page 3-66
c6	0	c0	0	Fault Address	R/W, B	R/W	0x00000000	page 3-68
			1	Watchpoint Fault Address	R/W	NA	0x00000000	page 3-69
			2	Instruction Fault Address	R/W, B	R/W	0x00000000	page 3-69
c7	0	c0	4	Wait For Interrupt	WO	WO	-	page 3-85
		c4	0	PA	R/W, B	R/W	0x00000000	page 3-80
		c5	0	Invalidate Entire Instruction Cache	WO	WO, X	-	page 3-71
			1	Invalidate Instruction Cache Line by MVA	WO	WO	-	page 3-71
			2	Invalidate Instruction Cache Line by Index	WO	WO	-	page 3-71
			4	Flush Prefetch Buffer	**WO**	**WO**	-	page 3-79
			6	Flush Entire Branch Target Cache	WO	WO	-	page 3-79
			7	Flush Branch Target Cache Entry by MVA	WO	WO	-	page 3-79
		c6	0	Invalidate Entire Data Cache	WO	NA	-	page 3-71
			1	Invalidate Data Cache Line by MVA	WO	WO	-	page 3-71
			2	Invalidate Data Cache Line by Index	WO	WO	-	page 3-71
		c7	0	Invalidate Both Caches	WO	NA	-	page 3-71
		c8	0-3	VA to PA translation in the current world	WO	WO	-	page 3-82
			4-7	VA to PA translation in the other world	WO	NA	-	page 3-83

Table 3-2 Summary of CP15 registers and operations (continued)

CRn	Op1	CRm	Op2	Register or operation	S type	NS type	Reset value	Page
c7	0	c10	0	Clean Entire Data Cache	WO, X	WO, X	-	page 3-71
			1	Clean Data Cache Line by MVA	WO	WO	-	page 3-71
			2	Clean Data Cache Line by Index	WO	WO	-	page 3-71
			4	Data Synchronization Barrier	**WO**	**WO**	-	page 3-83
			5	Data Memory Barrier	**WO**	**WO**	-	page 3-84
			6	Cache Dirty Status	RO, B	RO	0x00000000	page 3-78
		c13	1	Prefetch Instruction Cache Line	WO	WO	-	page 3-71
		c14	0	Clean and Invalidate Entire Data Cache	WO, X	WO, X	-	page 3-71
			1	Clean and Invalidate Data Cache Line by MVA	WO	WO	-	page 3-71
			2	Clean and Invalidate Data Cache Line by Index	WO	WO	-	page 3-71
c8	0	c5	0	Invalidate Instruction TLB unlocked entries	WO, B	WO	-	page 3-86
			1	Invalidate Instruction TLB entry by MVA	WO, B	WO	-	page 3-86
			2	Invalidate Instruction TLB entry on ASID match	WO, B	WO	-	page 3-86
c8	0	c6	0	Invalidate Data TLB unlocked entries	WO, B	WO	-	page 3-86
			1	Invalidate Data TLB entry by MVA	WO, B	WO	-	page 3-86
			2	Invalidate Data TLB entry on ASID match	WO, B	WO	-	page 3-86
		c7	0	Invalidate unified TLB unlocked entries	WO, B	WO	-	page 3-86
			1	Invalidate unified TLB entry by MVA	WO, B	WO	-	page 3-86
			2	Invalidate unified TLB entry on ASID match	WO, B	WO	-	page 3-86

Table 3-2 Summary of CP15 registers and operations (continued)

CRn	Op1	CRm	Op2	Register or operation	S type	NS type	Reset value	Page
c9	0	c0	0	Data Cache Lockdown	R/W	R/W, X	0xFFFFFFF0	page 3-87
			1	Instruction Cache Lockdown	R/W	R/W, X	0xFFFFFFF0	page 3-87
		c1	0	Data TCM Region	R/W, X	R/W, X	0x00000014[f]	page 3-89
			1	Instruction TCM Region	R/W, X	R/W, X	0x00000014[g]	page 3-91
			2	Data TCM Non-secure Control Access	R/W, X	NA	0x00000000	page 3-93
			3	Instruction TCM Non-secure Control Access	R/W, X	NA	0x00000000	page 3-94
		c2	0	TCM Selection	R/W, B	R/W	0x00000000	page 3-96
		c8	0	Cache Behavior Override	R/W[h]	R/W	0x00000000	page 3-97
c10	0	c0	0	TLB Lockdown	R/W, X	R/W, X	0x00000000	page 3-100
		c2	0	Primary Region Memory Remap Register	R/W, B, X	R/W	0x00098AA4	page 3-101
			1	Normal Memory Region Remap Register	R/W, B, X	R/W	0x44E048E0	page 3-101
c11	0	c0	0-3	DMA identification and status	RO	RO, X	0x0000000B[i]	page 3-106
		c1	0	DMA User Accessibility	R/W	R/W, X	0x00000000	page 3-107
		c2	0	DMA Channel Number	**R/W**, X	**R/W**, X	0x00000000	page 3-109
		c3	0-2	DMA enable	**WO**, X	**WO**, X	-	page 3-110
		c4	0	DMA Control	**R/W**, X	**R/W**, X	0x08000000	page 3-112
		c5	0	DMA Internal Start Address	**R/W**, X	**R/W**, X	-	page 3-114
		c6	0	DMA External Start Address	**R/W**, X	**R/W**, X	-	page 3-115
		c7	0	DMA Internal End Address	**R/W**, X	**R/W**, X	-	page 3-116
		c8	0	DMA Channel Status	**RO**, X	**RO**, X	0x00000000	page 3-117
		c15	0	DMA Context ID	R/W	R/W, X	-	page 3-120
c12	0	c0	0	Secure or Non-secure Vector Base Address	R/W, B, X	R/W	0x00000000	page 3-121
			1	Monitor Vector Base Address	R/W, X	NA	0x00000000	page 3-122
		c1	0	Interrupt Status	RO	RO	0x00000000[j]	page 3-123

Table 3-2 Summary of CP15 registers and operations (continued)

CRn	Op1	CRm	Op2	Register or operation	S type	NS type	Reset value	Page
c13	0	c0	0	FCSE PID	R/W, B, X	R/W	0x00000000	page 3-126
			1	Context ID	R/W, B	R/W	0x00000000	page 3-128
			2	User Read/Write Thread and Process ID	**R/W**, B	**R/W**	0x00000000	page 3-129
			3	User Read-only Thread and Process ID	R/W, **RO**, B[k]	R/W, **RO**	0x00000000	page 3-129
			4	Privileged Only Thread and Process ID	R/W, B	R/W	0x00000000	page 3-129
c14				Not used				
c15	0	c2	4	Peripheral Port Memory Remap	R/W, B, X	R/W	0x00000000	page 3-130
		c9	0	Secure User and Non-secure Access Validation Control	R/W, X	NA	0x00000000	page 3-132
		c12	0	Performance Monitor Control	**R/W**, X	**R/W**, X	0x00000000	page 3-133
			1	Cycle Counter	**R/W**, X	**R/W**, X	0x00000000	page 3-137
			2	Count 0	**R/W**, X	**R/W**, X	0x00000000	page 3-138
			3	Count 1	**R/W**, X	**R/W**, X	0x00000000	page 3-139
			4-7	System Validation Counter	**R/W**, X	**R/W**, X	0x00000000	page 3-140
		c13	1-7	System Validation Operations	**R/W**, X	**R/W**, X	0x00000000	page 3-142
		c14	0	System Validation Cache Size Mask	**R/W**, X	**R/W**, X	0x00006655[l]	page 3-145
c15	1	c13	0-7	System Validation Operations	**R/W**, X	**R/W**, X	0x00000000	page 3-142
c15	2	c13	1-7	System Validation Operations	**R/W**, X	**R/W**, X	0x00000000	page 3-142
c15	3	c8	0-7	Instruction Cache Master Valid	R/W, X	NA	0x00000000	page 3-147
		c12	0-7	Data Cache Master Valid	R/W, X	NA	0x00000000	page 3-148
		c13	0-7	System Validation Operations	**R/W**, X	**R/W**, X	0x00000000	page 3-142
c15	4	c13	0-7	System Validation Operations	**R/W**, X	**R/W**, X	0x00000000	page 3-142
c15	5	c4	2	TLB Lockdown Index	R/W, X	NA	0x00000000	page 3-149
		c5	2	TLB Lockdown VA	R/W, X	NA	-	page 3-149
		c6	2	TLB Lockdown PA	R/W, X	NA	-	page 3-149
		c7	2	TLB Lockdown Attributes	R/W, X	NA	-	page 3-149
		c13	0-7	System Validation Operations	**R/W**, X	**R/W**, X	0x00000000	page 3-142
c15	6	c13	0-7	System Validation Operations	**R/W**, X	**R/W**, X	0x00000000	page 3-142
c15	7	c13	0-7	System Validation Operations	**R/W**, X	**R/W**, X	0x00000000	page 3-142

a. See *c0, Main ID Register* on page 3-20 for the values of bits [23:20] and bits [3:0].

b. Reset value depends on the cache size implemented. The value here is for 16KB instruction and data caches.
c. Reset value depends on the number of TCM banks implemented. The value here is for 2 data TCM and 2 instruction TCM banks.
d. Some bits in this register are banked and some Secure modify only.
e. Reset value depends on external signals.
f. Reset value depends on the TCM sizes implemented. The value here is for 16KB TCM banks.
g. Reset value depends on the TCM sizes implemented, and on the value of the **INITRAM** static configuration signal. The value here is for 16KB TCM banks, with **INITRAM** tied LOW.
h. Some bits in this register are common and some Secure modify only.
i. Reset value depends on the number of DMA channels implemented and the presence of TCMs.
j. Reset value depends on external signals.
k. This register is read/write in Privileged modes and read-only on User mode.
l. Reset value depends on the cache and TCM sizes implemented. The value here is for 2 banks of 16KB instruction and data TCMs and 16KB instruction and data caches.

Table 3-3 lists the operations available with MCRR operations:

```
MCRR{cond} P15,<Opcode_1>,<End Address>,<Start Address>,<CRm>
```

Table 3-3 Summary of CP15 MCRR operations

Op1	CRm	Register or operation	S type	NS type	Reset value	Page
0	c5	Invalidate instruction cache range	WO	WO	-	page 3-69
	c6	Invalidate data cache range	WO	WO	-	page 3-69
	c12	Clean data cache range	**WO**	**WO**	-	page 3-69
	c14	Clean and invalidate data cache range	WO	WO	-	page 3-69

3.2.2 c0, Main ID Register

The purpose of the Main ID Register is to return the device ID code that contains information about the processor.

The Main ID Register is:

- in CP15 c0
- a 32 bit read-only register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-10 shows the arrangement of bits in the register.

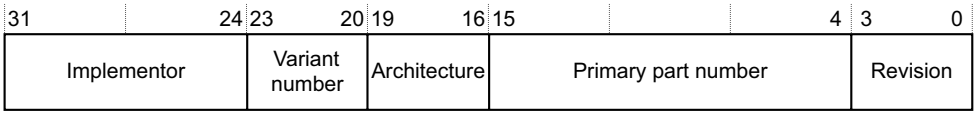

Figure 3-10 Main ID Register format

The contents of the Main ID Register depend on the specific implementation. Table 3-4 lists how the bit values correspond with the Main ID Register functions.

Table 3-4 Main ID Register bit functions

Bits	Field name	Function
[31:24]	Implementor	Indicates implementor, ARM Limited: `0x41`
[23:20]	Variant number	The major revision number *n* in the r*n* part of the r*n*p*n* revision status. `0x0`
[19:16]	Architecture	Indicates that the architecture is given in the CPUID registers: `0xF`
[15:4]	Primary part number	Indicates part number, ARM1176JZF-S: `0xB76`
[3:0]	Revision	The minor revision number *n* in the p*n* part of the r*n*p*n* revision status. For example: for release r0p0: `0x0` for release r0p7: `0x7`

Note

If an Opcode_2 value corresponding to an unimplemented or reserved ID register with CRm equal to c0 and Opcode_1 = 0 is encountered, the system control coprocessor returns the value of the main ID register.

Table 3-5 lists the results of attempted access for each mode.

Table 3-5 Results of access to the Main ID Register

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Main ID Register read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set to c0
- Opcode_2 set to 0.

For example:

```
MRC p15,0,<Rd>,c0,c0,0 ;Read Main ID Register
```

For more information on the processor features, see *c0, CPUID registers* on page 3-26.

3.2.3 c0, Cache Type Register

The purpose of the Cache Type Register is to provide information about the size and architecture of the cache for the operating system. This enables the operating system to establish how to clean the cache and how to lock it down. Inclusion of this register enables RTOS vendors to produce future-proof versions of their operating systems.

The Cache Type Register is:

- in CP15 c0
- a 32-bit read only register, common to Secure and Non-secure worlds
- accessible in privileged modes only.

All ARMv4T and later cached processors contain this register. Figure 3-11 shows the arrangement of bits in the Cache Type Register.

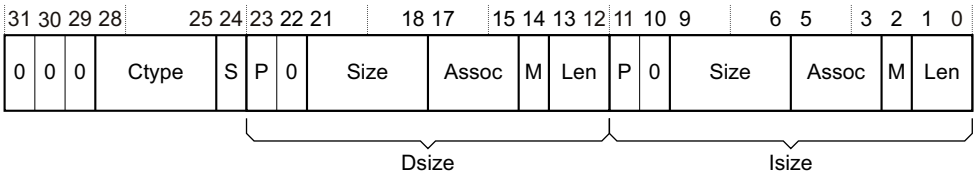

Figure 3-11 Cache Type Register format

Table 3-6 lists how the bit values correspond with the Cache Type Register functions.

Table 3-6 Cache Type Register bit functions

Bits	Field name	Function
[31:29]	-	0
[28:25]	Ctype	The Cache type and Separate bits provide information about the cache architecture. b1110, indicates that the ARM1176JZF-S processor supports: • write back cache • Format C cache lockdown • Register 7 cache cleaning operations.
[24]	S bit	S = 1, indicates that the processor has separate instruction and data caches and not a unified cache.

Table 3-6 Cache Type Register bit functions (continued)

Bits	Field name		Function
[23:12]	Dsize		Provides information about the size and construction of the Data cache. **Note** The ARM1176JZF-S processor does not support cache sizes of less than 4KB.
[23]		P bit	The P, Page, bit indicates restrictions on page allocation for bits [13:12] of the VA For ARM1176JZF-S processors, the P bit is set if the cache size is greater than 16KB. For more details see *Restrictions on page table mappings page coloring* on page 6-41. 0 = no restriction on page allocation. 1 = restriction applies to page allocation.
[22]		-	0
[21:18]		Size	The Size field indicates the cache size in conjunction with the M bit. b0000 = 0.5KB cache, not supported b0001 = 1KB cache, not supported b0010 = 2KB cache, not supported b0011 = 4KB cache b0100 = 8KB cache b0101 = 16KB cache b0110 = 32KB cache b0111 = 64KB cache b1000 = 128KB cache, not supported.
[17:15]		Assoc	b010, indicates that the ARM1176JZF-S processor has 4-way associativity. All other values for Assoc are reserved.
[14]		M bit	Indicates the cache size and cache associativity values in conjunction with the Size and Assoc fields. In the ARM1176JZF-S processor the M bit is set to 0, for the Data and Instruction Caches.
[13:12]		Len	b10, indicates that ARM1176JZF-S processor has a cache line length of 8 words, that is 32 bytes. All other values for Len are reserved.
[11:0]	Isize		Provides information about the size and construction of the Instruction cache.
[11]		P	The functions of the Isize bit fields are the same as the equivalent Dsize bit fields and the Isize values have the corresponding meanings.
[10]		-	
[9:6]		Size	
[5:3]		Assoc	
[2]		M	
[1:0]		Len	

Table 3-7 lists the results of attempted access for each mode.

Table 3-7 Results of access to the Cache Type Register

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Cache Type Register read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set to c0
- Opcode_2 set to 1.

For example:

```
MRC p15,0,<Rd>,c0,c0,1; returns cache details
```

Table 3-8, for example, lists the Cache Type Register values for an ARM1176JZF-S processor with:

- separate instruction and data caches
- cache size = 16KB
- associativity = 4-way
- line length = eight words
- caches use write-back, CP15 c7 for cache cleaning, and Format C for cache lockdown.

Table 3-8 Example Cache Type Register format

Bits	Field name		Value	Behavior
[31:29]	Reserved		b000	
[28:25]	Ctype		b1110	
[24]	S		b1	Harvard cache
[23]	Dsize	P	b0	
[22]		Reserved	b0	
[21:18]		Size	b0101	16KB
[17:15]		Assoc	b010	4-way
[14]		M	b0	
[13:12]		Len	b10	8 words per line, 32 bytes
[11]	Isize	P	b0	
[10]		Reserved	b0	
[9:6]		Size	b0101	16KB
[5:3]		Assoc	b010	4-way
[2]		M	b0	
[1:0]		Len	b10	8 words per line, 32 bytes

3.2.4 c0, TCM Status Register

The purpose of the TCM Status Register is to inform the system about the number of Instruction and Data TCMs available in the processor.

Table 3-9 lists the purposes of the individual bits in the TCM Status Register.

Note

In the ARM1176JZF-S processor there is a maximum of two Instruction TCMs and two Data TCMs.

The TCM Status Register is:

- in CP15 c0
- a 32-bit read-only register common to Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-12 shows the bit arrangement for the TCM Status Register.

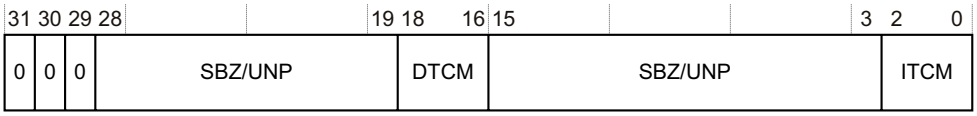

Figure 3-12 TCM Status Register format

Table 3-9 lists how the bit values correspond with the TCM Status Register functions.

Table 3-9 TCM Status Register bit functions

Bits	Field name	Function
[31:29]	-	Always b000.
[28:19]	-	UNP/SBZ
[18:16]	DTCM	Indicates the number of Data TCM banks implemented. b000 = 0 Data TCMs b001 = 1 Data TCM b010 = 2 Data TCMs All other values reserved
[15:3]	-	UNP/SBZ
[2:0]	ITCM	Indicates the number of Instruction TCM banks implemented. b000 = 0 Instruction TCMs b001 = 1 Instruction TCM b010 = 2 Instruction TCMs All other values reserved

Attempts to write the TCM Status Register or read it in User modes result in Undefined exceptions.

To use the TCM Status Register read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set to c0
- Opcode_2 set to 2.

For example:

```
MRC p15,0,<Rd>,c0,c0,2    ; returns TCM status register
```

3.2.5 c0, TLB Type Register

The purpose of the TLB Type Register is to return the number of lockable entries for the TLB.

The TLB has 64 entries organized as a unified two-way set associative TLB. In addition, it has eight lockable entries that the read-only TLB Type Register specifies.

The TLB Type Register is:

- in CP15 c0
- a 32-bit read only register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-13 shows the bit arrangement for the TLB Type Register.

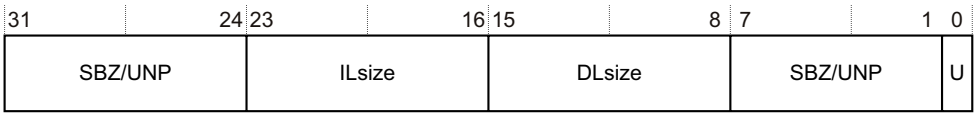

Figure 3-13 TLB Type Register format

Table 3-10 lists how the bit values correspond with the TLB Type Register functions.

Table 3-10 TLB Type Register bit functions

Bits	Field name	Function
[31:24]	-	UNP/SBZ
[23:16]	ILsize	Instruction lockable size specifies the number of instruction TLB lockable entries 0, indicates that the ARM1176JZF-S processor has a unified TLB
[15:8]	DLsize	Data lockable size specifies the number of unified or data TLB lockable entries `0x08`, indicates the ARM1176JZF-S processors has 8 unified TLB lockable entries
[7:1]	-	UNP/SBZ
[0]	U	Unified specifies if the TLB is unified, 0, or if there are separate instruction and data TLBs, 1. 0, indicates that the ARM1176JZF-S processor has a unified TLB

Table 3-11 lists the results of attempted access for each mode.

Table 3-11 Results of access to the TLB Type Register

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the TLB Type Register read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set to c0
- Opcode_2 set to 3.

For example:

```
MRC p15,0,<Rd>,c0,c0,3    ; returns TLB details
```

3.2.6 c0, CPUID registers

The section describes the CPUIID registers:

- *c0, Processor Feature Register 0*
- *c0, Processor Feature Register 1* on page 3-27
- *c0, Debug Feature Register 0* on page 3-29
- *c0, Auxiliary Feature Register 0* on page 3-30
- *c0, Memory Model Feature Register 0* on page 3-31
- *c0, Memory Model Feature Register 1* on page 3-32
- *c0, Memory Model Feature Register 2* on page 3-33
- *c0, Memory Model Feature Register 3* on page 3-35
- *c0, Instruction Set Attributes Register 0* on page 3-36
- *c0, Instruction Set Attributes Register 1* on page 3-37
- *c0, Instruction Set Attributes Register 2* on page 3-39
- *c0, Instruction Set Attributes Register 3* on page 3-40
- *c0, Instruction Set Attributes Register 4* on page 3-42
- *c0, Instruction Set Attributes Register 5* on page 3-43.

Note

The CPUID registers are sometimes described as the *Core Feature ID* registers.

c0, Processor Feature Register 0

The purpose of the Processor Feature Register 0 is to provide information about the execution state support and programmer's model for the processor.

Processor Feature Register 0 is:

- in CP15 c0
- a 32-bit read-only register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Table 3-12 lists how the bit values correspond with the Processor Feature Register 0 functions.

Figure 3-14 shows the bit arrangement for Processor Feature Register 0.

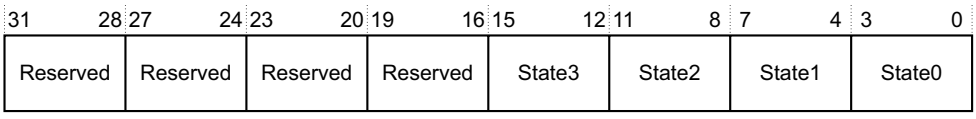

Figure 3-14 Processor Feature Register 0 format

Table 3-12 Processor Feature Register 0 bit functions

Bits	Field name	Function
[31:28]	-	Reserved. RAZ.
[27:24]	-	Reserved. RAZ.
[23:20]	-	Reserved. RAZ.

Table 3-12 Processor Feature Register 0 bit functions (continued)

Bits	Field name	Function
[19:16]	-	Reserved. RAZ.
[15:12]	State3	Indicates support for Thumb-2™ execution environment. 0x0, ARM1176JZF-S processors do not support Thumb-2.
[11:8]	State2	Indicates support for Java extension interface. 0x1, ARM1176JZF-S processors support Java.
[7:4]	State1	Indicates type of Thumb encoding that the processor supports. 0x1, ARM1176JZF-S processors support Thumb-1 but do not support Thumb-2.
[3:0]	State0	Indicates support for 32-bit ARM instruction set. 0x1, ARM1176JZF-S processors support 32-bit ARM instructions.

Table 3-13 lists the results of attempted access for each mode.

Table 3-13 Results of access to the Processor Feature Register 0

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Processor Feature Register 0 read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set to c1
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c0, c1, 0 ;Read Processor Feature Register 0
```

c0, Processor Feature Register 1

The purpose of the Processor Feature Register 1 is to provide information about the execution state support and programmer's model for the processor.

Processor Feature Register 1 is:

- in CP15 c0
- a 32-bit read-only register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-15 on page 3-28 shows the bit arrangement for Processor Feature Register 1.

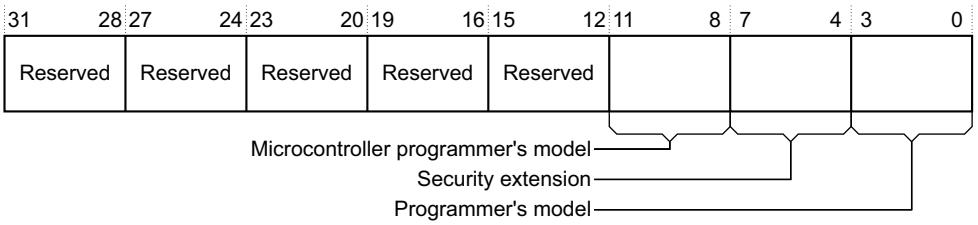

Figure 3-15 Processor Feature Register 1 format

Table 3-14 lists how the bit values correspond with the Processor Feature Register 1 functions.

Table 3-14 Processor Feature Register 1 bit functions

Bits	Field name	Function
[31:28]	-	Reserved. RAZ
[27:24]	-	Reserved. RAZ.
[23:20]	-	Reserved. RAZ.
[19:16]	-	Reserved. RAZ.
[15:12]	-	Reserved. RAZ.
[11:8]	Microcontroller programmer's model	Indicates support for the ARM microcontroller programmer's model. `0x0`, Not supported by ARM1176JZF-S processors.
[7:4]	Security extension	Indicates support for Security Extensions Architecture v1. `0x1`, ARM1176JZF-S processors support Security Extensions Architecture v1, TrustZone.
[3:0]	Programmer's model	Indicates support for standard ARMv4 programmer's model. `0x1`, ARM1176JZF-S processors support the ARMv4 model.

Table 3-15 lists the results of attempted access for each mode.

Table 3-15 Results of access to the Processor Feature Register 1

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Processor Feature Register 1 read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set to c1
- Opcode_2 set to 1.

For example:

```
MRC p15, 0, <Rd>, c0, c1, 1 ;Read Processor Feature Register 1
```

c0, Debug Feature Register 0

The purpose of the Debug Feature Register 0 is to provide information about the debug system for the processor.

Debug Feature Register 0 is:

- in CP15 c0
- a 32-bit read-only register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-16 shows the bit arrangement for Debug Feature Register 0.

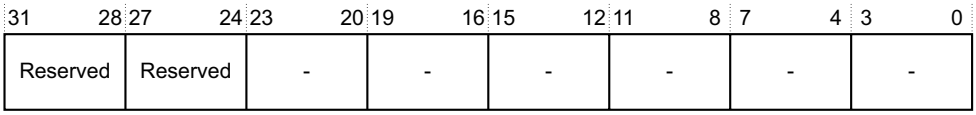

Figure 3-16 Debug Feature Register 0 format

Table 3-16 lists how the bit values correspond with the Debug Feature Register 0 functions.

Table 3-16 Debug Feature Register 0 bit functions

Bits	Field name	Function
[31:28]	-	Reserved. RAZ.
[27:24]	-	Reserved. RAZ.
[23:20]	-	Indicates the type of memory-mapped microcontroller debug model that the processor supports. `0x0`, ARM1176JZF-S processors do not support this debug model.
[19:16]	-	Indicates the type of memory-mapped Trace debug model that the processor supports. `0x0`, ARM1176JZF-S processors do not support this debug model.
[15:12]	-	Indicates the type of coprocessor-based Trace debug model that the processor supports. `0x0`, ARM1176JZF-S processors do not support this debug model.
[11:8]	-	Indicates the type of embedded processor debug model that the processor supports. `0x0`, ARM1176JZF-S processors do not support this debug model.
[7:4]	-	Indicates the type of Secure debug model that the processor supports. `0x3`, ARM1176JZF-S processors support the v6.1 Secure debug architecture based model.
[3:0]	-	Indicates the type of applications processor debug model that the processor supports. `0x3`, ARM1176JZF-S processors support the v6.1 debug model.

Table 3-17 lists the results of attempted access for each mode.

Table 3-17 Results of access to the Debug Feature Register 0

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Debug Feature Register 0 read CP15 with:

- Opcode_1 set to 0

- CRn set to c0
- CRm set to c1
- Opcode_2 set to 2.

For example:

```
MRC p15, 0, <Rd>, c0, c1, 2 ;Read Debug Feature Register 0
```

c0, Auxiliary Feature Register 0

The purpose of the Auxiliary Feature Register 0 is to provide additional information about the features of the processor.

The Auxiliary Feature Register 0 is:

- in CP15 c0
- a 32-bit read-only register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Table 3-18 lists how the bit values correspond with the Auxiliary Feature Register 0 functions.

Table 3-18 Auxiliary Feature Register 0 bit functions

Bits	Field name	Function
[31:16]	-	Reserved. RAZ.
[15:12]	-	Implementation Defined.
[11:8]	-	Implementation Defined.
[7:4]	-	Implementation Defined.
[3:0]	-	Implementation Defined.

The contents of the Auxiliary Feature Register 0 [31:16] are Reserved. The contents of the Auxiliary Feature Register 0 [15:0] are Implementation Defined. In the ARM1176JZF-S processor, the Auxiliary Feature Register 0 reads as `0x00000000`.

Table 3-19 lists the results of attempted access for each mode.

Table 3-19 Results of access to the Auxiliary Feature Register 0

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Auxiliary Feature Register 0 read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set to c1
- Opcode_2 set to 3.

For example:

```
MRC p15, 0, <Rd>, c0, c1, 3 ;Read Auxiliary Feature Register 0.
```

c0, Memory Model Feature Register 0

The purpose of the Memory Model Feature Register 0 is to provide information about the memory model, memory management, cache support, and TLB operations of the processor.

The Memory Model Feature Register 0 is:

- in CP15 c0
- a 32-bit read-only register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-17 shows the bit arrangement for Memory Model Feature Register 0.

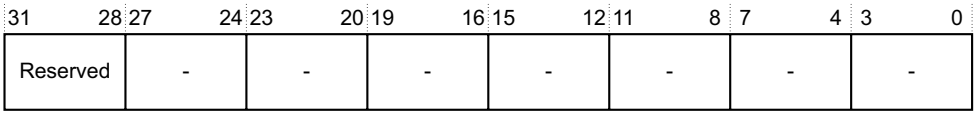

Figure 3-17 Memory Model Feature Register 0 format

Table 3-20 lists how the bit values correspond with the Memory Model Feature Register 0 functions.

Table 3-20 Memory Model Feature Register 0 bit functions

Bits	Field name	Function
[31:28]	-	Reserved. RAZ.
[27:24]	-	Indicates support for FCSE. `0x1`, ARM1176JZF-S processors support FCSE.
[23:20]	-	Indicates support for the ARMv6 Auxiliary Control Register. `0x1`, ARM1176JZF-S processors support the Auxiliary Control Register.
[19:16]	-	Indicates support for TCM and associated DMA. `0x3`, ARM1176JZF-S processors support ARMv6 TCM and DMA.
[15:12]	-	Indicates support for cache coherency with DMA agent, shared memory. `0x0`, ARM1176JZF-S processors do not support this model.
[11:8]	-	Indicates support for cache coherency support with CPU agent, shared memory. `0x0`, ARM1176JZF-S processors do not support this model.
[7:4]	-	Indicates support for *Protected Memory System Architecture* (PMSA). `0x0`, ARM1176JZF-S processors do not support PMSA
[3:0]	-	Indicates support for *Virtual Memory System Architecture* (VMSA). `0x3`, ARM1176JZF-S processors support: • VMSA v7 remapping and access flag.

Table 3-21 lists the results of attempted access for each mode.

Table 3-21 Results of access to the Memory Model Feature Register 0

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Memory Model Feature Register 0 read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set to c1
- Opcode_2 set to 4.

For example:

```
MRC p15, 0, <Rd>, c0, c1, 4 ;Read Memory Model Feature Register 0.
```

c0, Memory Model Feature Register 1

The purpose of the Memory Model Feature Register 1 is to provide information about the memory model, memory management, cache support, and TLB operations of the processor.

The Memory Model Feature Register 1 is:

- in CP15 c0
- a 32-bit read-only register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-18 shows the bit arrangement for Memory Model Feature Register 1.

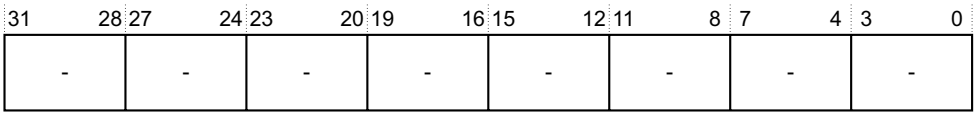

Figure 3-18 Memory Model Feature Register 1 format

Table 3-22 lists how the bit values correspond with the Memory Model Feature Register 1 functions.

Table 3-22 Memory Model Feature Register 1 bit functions

Bits	Field name	Function
[31:28]	-	Indicates support for branch target buffer. 0x1, ARM1176JZF-S processors require flushing of branch predictor on VA change.
[27:24]	-	Indicates support for test and clean operations on data cache, Harvard or unified architecture. 0x0, no support in ARM1176JZF-S processors.
[23:20]	-	Indicates support for level one cache, all maintenance operations, unified architecture. 0x0, no support in ARM1176JZF-S processors.
[19:16]	-	Indicates support for level one cache, all maintenance operations, Harvard architecture. 0x3, ARM1176JZF-S processors support: • invalidate instruction cache including branch prediction • invalidate data cache • invalidate instruction and data cache including branch prediction • clean data cache, recursive model using cache dirty status bit • clean and invalidate data cache, recursive model using cache dirty status bit.
[15:12]	-	Indicates support for level one cache line maintenance operations by Set/Way, unified architecture. 0x0, no support in ARM1176JZF-S processors.

Table 3-22 Memory Model Feature Register 1 bit functions (continued)

Bits	Field name	Function
[11:8]	-	Indicates support for level one cache line maintenance operations by Set/Way, Harvard architecture. 0x3, ARM1176JZF-S processors support: • clean data cache line by Set/Way • clean and invalidate data cache line by Set/Way • invalidate data cache line by Set/Way • invalidate instruction cache line by Set/Way.
[7:4]	-	Indicates support for level one cache line maintenance operations by MVA, unified architecture. 0, no support in ARM1176JZF-S processors.
[3:0]	-	Indicates support for level one cache line maintenance operations by MVA, Harvard architecture. 0x2, ARM1176JZF-S processors support: • clean data cache line by MVA • invalidate data cache line by MVA • invalidate instruction cache line by MVA • clean and invalidate data cache line by MVA • invalidation of branch target buffer by MVA.

Table 3-23 lists the results of attempted access for each mode.

Table 3-23 Results of access to the Memory Model Feature Register 1

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Memory Model Feature Register 1 read CP15 with:
- Opcode_1 set to 0
- CRn set to c0
- CRm set to c1
- Opcode_2 set to 5.

For example:

```
MRC p15, 0, <Rd>, c0, c1, 5 ;Read Memory Model Feature Register 1.
```

c0, Memory Model Feature Register 2

The purpose of the Memory Model Feature Register 2 is to provide information about the memory model, memory management, cache support, and TLB operations of the processor.

The Memory Model Feature Register 2 is:
- in CP15 c0
- a 32-bit read-only register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-19 on page 3-34 shows the bit arrangement for Memory Model Feature Register 2.

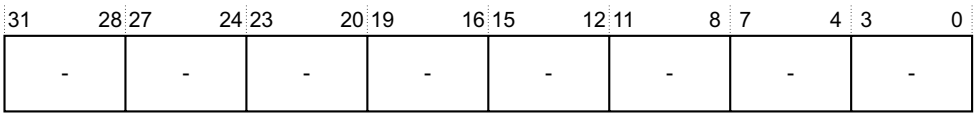

Figure 3-19 Memory Model Feature Register 2 format

Table 3-24 lists how the bit values correspond with the Memory Model Feature Register 2 functions.

Table 3-24 Memory Model Feature Register 2 bit functions

Bits	Field name	Function
[31:28]	-	Indicates support for a Hardware access flag. `0x0`, no support in ARM1176JZF-S processors.
[27:24]	-	Indicates support for Wait For Interrupt stalling. `0x1`, ARM1176JZF-S processors support Wait For Interrupt.
[23:20]	-	Indicates support for memory barrier operations. `0x2`, ARM1176JZF-S processors support: • Data Synchronization Barrier • Prefetch Flush • Data Memory Barrier.
[19:16]	-	Indicates support for TLB maintenance operations, unified architecture. `0x2`, ARM1176JZF-S processors support: • invalidate all entries • invalidate TLB entry by MVA • invalidate TLB entries by ASID match.
[15:12]	-	Indicates support for TLB maintenance operations, Harvard architecture. `0x2`, ARM1176JZF-S processors support: • invalidate instruction and data TLB, all entries • invalidate instruction TLB, all entries • invalidate data TLB, all entries • invalidate instruction TLB by MVA • invalidate data TLB by MVA • invalidate instruction and data TLB entries by ASID match • invalidate instruction TLB entries by ASID match • invalidate data TLB entries by ASID match.
[11:8]	-	Indicates support for cache maintenance range operations, Harvard architecture. `0x1`, ARM1176JZF-S processors support: • invalidate data cache range by VA • invalidate instruction cache range by VA • clean data cache range by VA • clean and invalidate data cache range by VA.
[7:4]	-	Indicates support for background prefetch cache range operations, Harvard architecture. `0x0`, no support in ARM1176JZF-S processors.
[3:0]	-	Indicates support for foreground prefetch cache range operations, Harvard architecture. `0x0`, no support in ARM1176JZF-S processors.

Table 3-25 lists the results of attempted access for each mode.

Table 3-25 Results of access to the Memory Model Feature Register 2

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Memory Model Feature Register 2 read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set to c1
- Opcode_2 set to 6.

For example:

```
MRC p15, 0, <Rd>, c0, c1, 6 ;Read Memory Model Feature Register 2.
```

c0, Memory Model Feature Register 3

The purpose of the Memory Model Feature Register 3 is to provide information about the memory model, memory management, cache support, and TLB operations of the processor.

The Memory Model Feature Register 3 is:

- in CP15 c0
- a 32-bit read-only register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-20 shows the bit arrangement for Memory Model Feature Register 3.

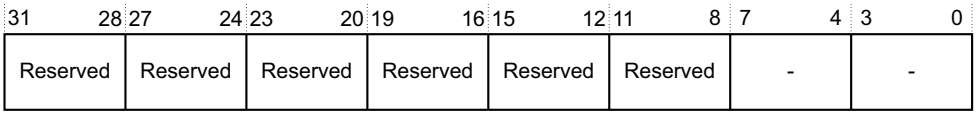

Figure 3-20 Memory Model Feature Register 3 format

Table 3-26 lists how the bit values correspond with the Memory Model Feature Register 3 functions.

Table 3-26 Memory Model Feature Register 3 bit functions

Bits	Field name	Function
[31:8]	-	Reserved. RAZ.
[7:4]	-	Support for hierarchical cache maintenance by MVA, all architectures `0x0`, no support in ARM1176JZF-S processors.
[3:0]	-	Support for hierarchical cache maintenance by Set/Way, all architectures. `0x0`, no support in ARM1176JZF-S processors.

Table 3-27 lists the results of attempted access for each mode.

Table 3-27 Results of access to the Memory Model Feature Register 3

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Memory Model Feature Register 3 read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set to c1
- Opcode_2 set to 7.

For example:

```
MRC p15, 0, <Rd>, c0, c1, 7 ;Read Memory Model Feature Register 3.
```

c0, Instruction Set Attributes Register 0

The purpose of the Instruction Set Attributes Register 0 is to provide information about the instruction set that the processor supports beyond the basic set.

The Instruction Set Attributes Register 0 is:

- in CP15 c0
- a 32-bit read-only register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-21 shows the bit arrangement for Instruction Set Attributes Register 0.

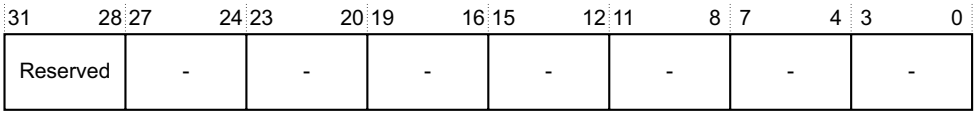

Figure 3-21 Instruction Set Attributes Register 0 format

Table 3-28 lists how the bit values correspond with the Instruction Set Attributes Register 0 functions.

Table 3-28 Instruction Set Attributes Register 0 bit functions

Bits	Field name	Function
[31:28]	-	Reserved. RAZ.
[27:24]	-	Indicates support for divide instructions. 0x0, no support in ARM1176JZF-S processors.
[23:20]	-	Indicates support for debug instructions. 0x1, ARM1176JZF-S processors support BKPT.

Table 3-28 Instruction Set Attributes Register 0 bit functions (continued)

Bits	Field name	Function
[19:16]	-	Indicates support for coprocessor instructions. 0x4, ARM1176JZF-S processors support: • CDP, LDC, MCR, MRC, STC • CDP2, LDC2, MCR2, MRC2, STC2 • MCRR, MRRC • MCRR2, MRRC2.
[15:12]	-	Indicates support for combined compare and branch instructions. 0x0, no support in ARM1176JZF-S processors.
[11:8]	-	Indicates support for bitfield instructions. 0x0, no support in ARM1176JZF-S processors.
[7:4]	-	Indicates support for bit counting instructions. 0x1, ARM1176JZF-S processors support CLZ.
[3:0]	-	Indicates support for atomic load and store instructions. 0x1, ARM1176JZF-S processors support SWP and SWPB.

Table 3-29 lists the results of attempted access for each mode.

Table 3-29 Results of access to the Instruction Set Attributes Register 0

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Instruction Set Attributes Register 0 read CP15 with:
- Opcode_1 set to 0
- CRn set to c0
- CRm set to c2
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c0, c2, 0 ;Read Instruction Set Attributes Register 0
```

c0, Instruction Set Attributes Register 1

The purpose of the Instruction Set Attributes Register 1 is to provide information about the instruction set that the processor supports beyond the basic set.

The Instruction Set Attributes Register 1 is:
- in CP15 c0
- a 32-bit read-only register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-22 on page 3-38 shows the bit arrangement for Instruction Set Attributes Register 1.

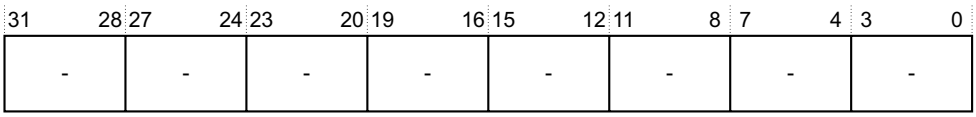

Figure 3-22 Instruction Set Attributes Register 1 format

Table 3-30 lists how the bit values correspond with the Instruction Set Attributes Register 1 functions.

Table 3-30 Instruction Set Attributes Register 1 bit functions

Bits	Field name	Function
[31:28]	-	Indicates support for Java instructions. 0x1, ARM1176JZF-S processors support BXJ and J bit in PSRs.
[27:24]	-	Indicates support for interworking instructions. 0x2, ARM1176JZF-S processors support: • BX, and T bit in PSRs • BLX, and PC loads have BX behavior.
[23:20]	-	Indicates support for immediate instructions. 0x0, no support in ARM1176JZF-S processors.
[19:16]	-	Indicates support for if then instructions. 0x0, no support in ARM1176JZF-S processors.
[15:12]	-	Indicates support for sign or zero extend instructions. 0x2, ARM1176JZF-S processors support: • SXTB, SXTB16, SXTH, UXTB, UXTB16, and UXTH • SXTAB, SXTAB16, SXTAH, UXTAB, UXTAB16, and UXTAH.
[11:8]	-	Indicates support for exception 2 instructions. 0x1, ARM1176JZF-S processors support SRS, RFE, and CPS.
[7:4]	-	Indicates support for exception 1 instructions. 0x1, ARM1176JZF-S processors support LDM(2), LDM(3) and STM(2).
[3:0]	-	Indicates support for endianness control instructions. 0x1, ARM1176JZF-S processors support SETEND and E bit in PSRs.

Table 3-31 lists the results of attempted access for each mode.

Table 3-31 Results of access to the Instruction Set Attributes Register 1

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Instruction Set Attributes Register 1 read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set to c2
- Opcode_2 set to 1.

For example:

```
MRC p15, 0, <Rd>, c0, c2, 1 ;Read Instruction Set Attributes Register 1
```

c0, Instruction Set Attributes Register 2

The purpose of the Instruction Set Attributes Register 2 is to provide information about the instruction set that the processor supports beyond the basic set.

The Instruction Set Attributes Register 2 is:

- in CP15 c0
- a 32-bit read-only register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-23 shows the bit arrangement for Instruction Set Attributes Register 2.

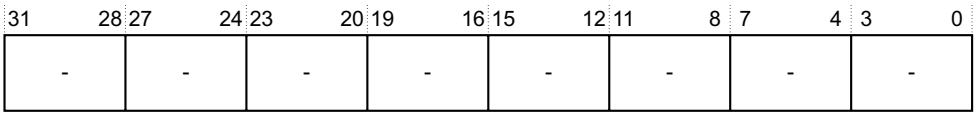

Figure 3-23 Instruction Set Attributes Register 2 format

Table 3-32 lists how the bit values correspond with the Instruction Set Attributes Register 2 functions.

Table 3-32 Instruction Set Attributes Register 2 bit functions

Bits	Field name	Function
[31:28]	-	Indicates support for reversal instructions. 0x1, ARM1176JZF-S processors support REV, REV16, and REVSH.
[27:24]	-	Indicates support for PSR instructions. 0x1, ARM1176JZF-S processors support MRS and MSR exception return instructions for data-processing.
[23:20]	-	Indicates support for advanced unsigned multiply instructions. 0x2, ARM1176JZF-S processors support: • UMULL and UMLAL • UMAAL.
[19:16]	-	Indicates support for advanced signed multiply instructions. 0x3, ARM1176JZF-S processors support: • SMULL and SMLAL • SMLABB, SMLABT, SMLALBB,SMLALBT, SMLALTB, SMLALTT, SMLATB, SMLATT, SMLAWB, SMLAWT, SMULBB, SMULBT, SMULTB, SMULTT, SMULWB, SMULWT, and Q flag in PSRs • SMLAD, SMLADX, SMLALD, SMLALDX, SMLSD, SMLSDX, SMLSLD, SMLSLDX, SMMLA, SMMLAR, SMMLS, SMMLSR, SMMUL, SMMULR, SMUAD, SMUADX, SMUSD, and SMUSDX.
[15:12]	-	Indicates support for multiply instructions. 0x1, ARM1176JZF-S processors support MLA.

Table 3-32 Instruction Set Attributes Register 2 bit functions (continued)

Bits	Field name	Function
[11:8]	-	Indicates support for multi-access interruptible instructions. 0x1, ARM1176JZF-S processors support restartable LDM and STM.
[7:4]	-	Indicates support for memory hint instructions. 0x2, ARM1176JZF-S processors support PLD.
[3:0]	-	Indicates support for load and store instructions. 0x1, ARM1176JZF-S processors support LDRD and STRD.

Table 3-33 lists the results of attempted access for each mode.

Table 3-33 Results of access to the Instruction Set Attributes Register 2

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Instruction Set Attributes Register 2 read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set to c2
- Opcode_2 set to 2.

For example:

```
MRC p15, 0, <Rd>, c0, c2, 2 ;Read Instruction Set Attributes Register 2
```

c0, Instruction Set Attributes Register 3

The purpose of the Instruction Set Attributes Register 3 is to provide information about the instruction set that the processor supports beyond the basic set.

The Instruction Set Attributes Register 3 is:

- in CP15 c0
- a 32-bit read-only registers common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-24 shows the bit arrangement for Instruction Set Attributes Register 3.

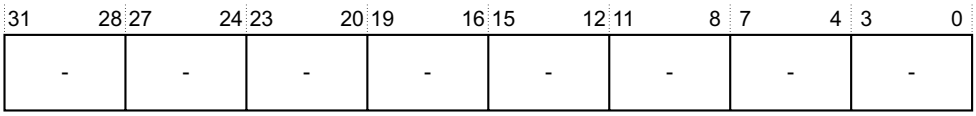

Figure 3-24 Instruction Set Attributes Register 3 format

Table 3-34 lists how the bit values correspond with the Instruction Set Attributes Register 3 functions.

Table 3-34 Instruction Set Attributes Register 3 bit functions

Bits	Field name	Function
[31:28]	-	Indicates support for Thumb-2 extensions. `0x0`, no support in ARM1176JZF-S processors.
[27:24]	-	Indicates support for true NOP instructions. `0x1`, ARM1176JZF-S processors support NOP and the capability for additional NOP compatible hints. ARM1176JZF-S processors do not support NOP16.
[23:20]	-	Indicates support for Thumb copy instructions. `0x1`, ARM1176JZF-S processors support Thumb MOV(3) low register ⇒ low register, and the CPY alias for Thumb MOV(3).
[19:16]	-	Indicates support for table branch instructions. `0x0`, no support in ARM1176JZF-S processors.
[15:12]	-	Indicates support for synchronization primitive instructions. `0x2`, ARM1176JZF-S processors support: • LDREX and STREX • LDREXB, LDREXH, LDREXD, STREXB, STREXH, STREXD, and CLREX
[11:8]	-	Indicates support for SVC instructions. `0x1`, ARM1176JZF-S processors support SVC.
[7:4]	-	Indicates support for *Single Instruction Multiple Data* (SIMD) instructions. `0x3`, ARM1176JZF-S processors support: PKHBT, PKHTB, QADD16, QADD8, QADDSUBX, QSUB16, QSUB8, QSUBADDX, SADD16, SADD8, SADDSUBX, SEL, SHADD16, SHADD8, SHADDSUBX, SHSUB16, SHSUB8, SHSUBADDX, SSAT, SSAT16, SSUB16, SSUB8, SSUBADDX, SXTAB16, SXTB16, UADD16, UADD8, UADDSUBX, UHADD16, UHADD8, UHADDSUBX, UHSUB16, UHSUB8, UHSUBADDX, UQADD16, UQADD8, UQADDSUBX, UQSUB16, UQSUB8, UQSUBADDX, USAD8, USADA8, USAT, USAT16, USUB16, USUB8, USUBADDX, UXTAB16, UXTB16, and the GE[3:0] bits in the PSRs.
[3:0]	-	Indicates support for saturate instructions. `0x1`, ARM1176JZF-S processors support QADD, QDADD, QDSUB, QSUB and Q flag in PSRs.

Table 3-35 lists the results of attempted access for each mode.

Table 3-35 Results of access to the Instruction Set Attributes Register 3

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Instruction Set Attributes Register 3 read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set to c2
- Opcode_2 set to 3.

For example:

```
MRC p15, 0, <Rd>, c0, c2, 3 ;Read Instruction Set Attributes Register 3
```

c0, Instruction Set Attributes Register 4

The purpose of the Instruction Set Attributes Register 4 is to provide information about the instruction set that the processor supports beyond the basic set.

The Instruction Set Attributes Register 4 is:

- in CP15 c0
- a 32-bit read-only register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-25 shows the bit arrangement for Instruction Set Attributes Register 4.

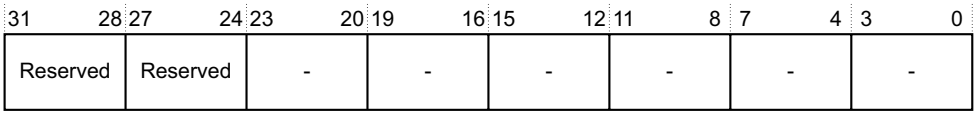

Figure 3-25 Instruction Set Attributes Register 4 format

Table 3-36 lists how the bit values correspond with the Instruction Set Attributes Register 4 functions.

Table 3-36 Instruction Set Attributes Register 4 bit functions

Bits	Field name	Function
[31:28]	-	Reserved. RAZ.
[27:24]	-	Reserved. RAZ.
[23:20]	-	Indicates fractional support for synchronization primitive instructions. 0x0, ARM1176JZF-S processors support all synchronization primitive instructions. See Table 3-34 on page 3-41.
[19:16]	-	Indicates support for barrier instructions. 0x0, None. ARM1176JZF-S processors support only the CP15 barrier operations.
[15:12]	-	Indicates support for SMC instructions. 0x1, ARM1176JZF-S processors support SMC.
[11:8]	-	Indicates support for writeback instructions. 0x1, ARM1176JZF-S processors support all defined writeback addressing modes.
[7:4]	-	Indicates support for with shift instructions. 0x4, ARM1176JZF-S processors support: • shifts of loads and stores over the range LSL 0-3 • constant shift options • register controlled shift options.
[3:0]	-	Indicates support for Unprivileged instructions. 0x1, ARM1176JZF-S processors support LDRBT, LDRT, STRBT, and STRT.

Table 3-37 lists the results of attempted access for each mode.

Table 3-37 Results of access to the Instruction Set Attributes Register 4

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Instruction Set Attributes Register 4 read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set to c2
- Opcode_2 set to 4.

For example:

```
MRC p15, 0, <Rd>, c0, c2, 4 ;Read Instruction Set Attributes Register 4
```

c0, Instruction Set Attributes Register 5

The purpose of the Instruction Set Attributes Register 5 is to provide additional information about the properties of the processor.

The Instruction Set Attributes Register 5 is:

- in CP15 c0
- a 32-bit read-only registers common to the Secure and Non-secure worlds
- accessible in privileged modes only.

The contents of the Instruction Set Attributes Register 5 are implementation defined. In the ARM1176JZF-S processor, Instruction Set Attributes Register 5 is read as `0x00000000`.

Table 3-38 lists the results of attempted access for each mode.

Table 3-38 Results of access to the Instruction Set Attributes Register 5

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

To use the Instruction Set Attributes Register 5 read CP15 with:

- Opcode_1 set to 0
- CRn set to c0
- CRm set toc2
- Opcode_2 set to 5.

For example:

```
MRC p15, 0, <Rd>, c0, c2, 5 ;Read Instruction Set Attribute Register 5.
```

3.2.7 c1, Control Register

This section contains information on:

- *Purpose of the Control Register*
- *Structure of the Control Register*
- *Operation of the Control Register* on page 3-45
- *Use of the Control Register* on page 3-47
- *Behavior of the Control Register* on page 3-48.

Purpose of the Control Register

The purpose of the Control Register is to provide control and configuration of:

- memory alignment, endianness, protection, and fault behavior
- MMU and cache enables and cache replacement strategy
- interrupts and the behavior of interrupt latency
- the location for exception vectors
- program flow prediction.

Table 3-39 on page 3-45 lists the purposes of the individual bits in the Control Register.

Structure of the Control Register

The Control Register is:

- in CP15 c1
- a 32 bit register, Table 3-39 on page 3-45 lists read and write access to individual bits for the Secure and Non-secure worlds
- accessible in privileged modes only
- partially banked, Table 3-39 on page 3-45 lists banked and Secure modify only bits.

Figure 3-26 shows the arrangement of bits in the register.

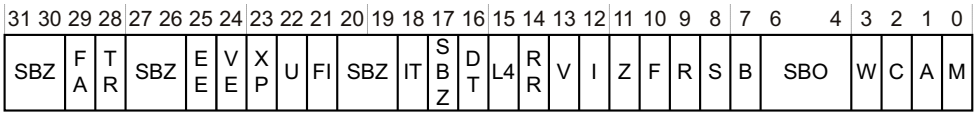

Figure 3-26 Control Register format

Operation of the Control Register

Table 3-39 lists how the bit values correspond with the Control Register functions.

Table 3-39 Control Register bit functions

Bits	Field name	Access	Function
[31:30]	-	-	This field is UNP when read. Write as the existing value.
[29]	FA	Banked	This bit controls the Force AP functionality in the MMU that generates Access Bit faults, see *Access permissions* on page 6-11 0 = Force AP is disabled, reset value. 1 = Force AP is enabled.
[28]	TR	Banked	This bit controls the TEX remap functionality in the MMU, see *Memory region attributes* on page 6-14. 0 = TEX remap disabled. Normal ARMv6 behavior, reset value 1 = TEX remap enabled. TEX[2:1] become page table bits for OS.
[27:26]	-	-	This field is UNP when read. Write as the existing value.
[25]	EE bit	Banked	Determines how the E bit in the CPSR bit is set on an exception. The reset value depends on external signals. 0 = CPSR E bit is set to 0 on an exception, reset value. 1 = CPSR E bit is set to 1 on an exception.
[24]	VE bit	Banked	Enables the VIC interface to determine interrupt vectors. See the description of the V bit, bit [13]. 0 = Interrupt vectors are fixed, reset value. 1 = Interrupt vectors are defined by the VIC interface.
[23]	XP bit	Banked	Enables the extended page tables to be configured for the hardware page translation mechanism. 0 = Subpage AP bits enabled, reset value. 1 = Subpage AP bits disabled.
[22]	U bit	Banked	Enables unaligned data access operations, including support for mixed little-endian and big-endian operation. The A bit has priority over the U bit. The reset value of the U bit depends on external signals. 0 = Unaligned data access support disabled, reset value. The processor treats unaligned loads as rotated aligned data accesses. 1 = Unaligned data access support enabled. The processor permits unaligned loads and stores and support for mixed endian data is enabled.
[21]	FI bit	Secure modify only	Configures low latency features for fast interrupts. This bit is overridden by the FIO bit, see *c1, Auxiliary Control Register* on page 3-48. 0 = All performance features enabled, reset value. 1 = Low interrupt latency configuration enabled. See *Low interrupt latency configuration* on page 2-40.
[20:19]	-	-	UNP/SBZ
[18]	IT bit	-	Deprecated. Global enable for instruction TCM. Function redundant in ARMv6. SBO
[17]	-	-	UNP/SBZ

Table 3-39 Control Register bit functions (continued)

Bits	Field name	Access	Function
[16]	DT bit	-	Deprecated. Global enable for data TCM. Function redundant in ARMv6. SBO
[15]	L4 bit	Secure modify only	Determines if the T bit is set for PC load instructions. For more details see the *ARM Architecture Reference Manual*. 0 = Loads to PC set the T bit, reset value. 1 = Loads to PC do not set the T bit, ARMv4 behavior.
[14]	RR bit	Secure modify only	Determines the replacement strategy for the cache. 0 = Normal replacement strategy by random replacement, reset value. 1 = Predictable replacement strategy by round-robin replacement.
[13]	V bit	Banked	Determines the location of exception vectors, see *c12, Secure or Non-secure Vector Base Address Register* on page 3-121 and *c12, Monitor Vector Base Address Register* on page 3-122. The reset value of the V bit depends on an external signal. 0 = Normal exception vectors selected, the Vector Base Address Registers determine the address range, reset value. 1 = High exception vectors selected, address range = `0xFFFF0000-0xFFFF001C`.
[12]	I bit	Banked	Enables level one instruction cache. 0 = Instruction Cache disabled, reset value. 1 = Instruction Cache enabled.
[11]	Z bit	Banked	Enables branch prediction. 0 = Program flow prediction disabled, reset value. 1 = Program flow prediction enabled.
[10]	F bit	-	Should Be Zero
[9]	R bit	Banked	Deprecated. Enables ROM protection. If you modify the R bit this does not affect the access permissions of entries already in the TLB. See *MMU software-accessible registers* on page 6-53. 0 = ROM protection disabled, reset value. 1 = ROM protection enabled.
[8]	S bit	Banked	Deprecated. Enables MMU protection. If you modify the S bit this does not affect the access permissions of entries already in TLB. 0 = MMU protection disabled, reset value. 1 = MMU protection enabled.
[7]	B bit	Secure modify only	Determines operation as little-endian or big-endian word invariant memory system and the names of the low four-byte addresses within a 32-bit word. The reset value of the B bit depends on the **BIGENDINIT** external signal. 0 = Little-endian memory system, reset value. 1 = Big-endian word-invariant memory system.
[6:4]	-	-	This field returns 1 when read. Should Be One.
[3]	W bit	-	Not implemented in the processor. Read As One Write Ignore.

Table 3-39 Control Register bit functions (continued)

Bits	Field name	Access	Function
[2]	C bit	Banked	Enables level one data cache. 0 = Data cache disabled, reset value. 1 = Data cache enabled.
[1]	A bit	Banked	Enables strict alignment of data to detect alignment faults in data accesses. The A bit setting takes priority over the U bit. 0 = Strict alignment fault checking disabled, reset value. 1 = Strict alignment fault checking enabled.
[0]	M bit	Banked	Enables the MMU. 0 = MMU disabled, reset value. 1 = MMU enabled.

Attempts to read or write the Control Register from Secure or Non-secure User modes results in an Undefined exception.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Attempts to write Secure modify only bit in Non-secure privileged modes are ignored.

Attempts to read Secure modify only bits return the Secure bit value. Table 3-40 lists the actions that result from attempted access for each mode.

Table 3-40 Results of access to the Control Register

Access type	Secure Privileged	Non-secure Privileged		User
		Read	Write	
Secure modify only	Secure bit	Secure bit	Ignored	Undefined exception
Banked	Secure bit	Non-secure bit	Non-secure bit	Undefined exception

Use of the Control Register

To use the Control Register it is recommended that you use a read modify write technique. To use the Control Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c1
- CRm set to c0
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c1, c0, 0    ; Read Control Register configuration data
MCR p15, 0, <Rd>, c1, c0, 0    ; Write Control Register configuration data
```

Normally, to set the V bit and the B, EE, and U bits you configure signals at reset.

The V bit depends on **VINITHI** at reset:

- **VINITHI** LOW sets V to 0
- **VINITHI** HIGH sets V to 1.

The B, EE, and U bits depend on how you set **BIGENDINIT** and **UBITINIT** at reset. Table 3-41 lists the values of the B, EE, and U bits that result for the reset values of these signals. See *Reset values of the U, B, and EE bits* on page 4-19.

Table 3-41 Resultant B bit, U bit, and EE bit values

UBITINIT	BIGENDINIT	EE	U	B
0	0	0	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	1	0

Behavior of the Control Register

These bits in the Control Register exhibit specific behavior:

A bit The A bit setting takes priority over the U bit. The Data Abort trap is taken if strict alignment is enabled and the data access is not aligned to the width of the accessed data item.

DT bit This bit is used in ARM946 and ARM966 processors to enable the Data TCM.

In ARMv6, the TCM blocks have individual enables that apply to each block. As a result, this bit is now redundant and Should Be One. See *c9, Data TCM Region Register* on page 3-89 for a description of the ARM1176JZF-S TCM enables.

IT bit This bit is used in ARM946 and ARM966 processors to enable the Instruction TCM.

In ARMv6, the TCM blocks have individual enables that apply to each block. As a result, this bit is now redundant and Should Be One. See *c9, Instruction TCM Region Register* on page 3-91 for a description of the ARM1176JZF-S TCM enables.

R bit Modifying the R bit does not affect the access permissions of entries already in the TLB. See *MMU software-accessible registers* on page 6-53.

S bit Modifying the S bit does not affect the access permissions of entries already in the TLB. See *MMU software-accessible registers* on page 6-53.

W bit The ARM1176JZF-S processor does not implement the write buffer enable because all memory writes take place through the Write Buffer.

3.2.8 c1, Auxiliary Control Register

The purpose of the Auxiliary Control Register is to control:

- program flow
- low interrupt latency
- cache cleaning
- MicroTLB cache strategy
- cache size restriction.

For more information on how the system control coprocessor operates with caches, see *Cache control and configuration* on page 3-7.

Table 3-42 lists the purposes of the individual bits in the Auxiliary Control Register.

The Auxiliary Control Register is:

- in CP15 c1
- a 32-bit:
  - read/write register in the Secure world
  - read only register in the Non-secure world
- accessible in privileged modes only.

Figure 3-27 shows the arrangement of bits in the register.

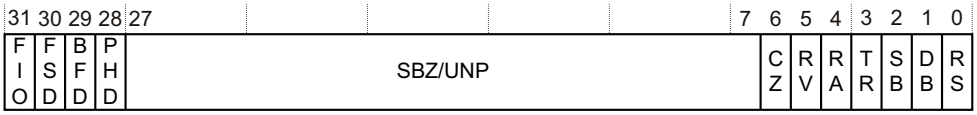

Figure 3-27 Auxiliary Control Register format

Table 3-42 lists how the bit values correspond with the Auxiliary Control Register functions.

Table 3-42 Auxiliary Control Register bit functions

Bits	Field name	Function
[31]	FIO	Provides additional level of control for low interrupt latency configuration. This bit overrides the FI bit, see FI bit in *c1, Control Register* on page 3-44: 0 = Normal operation for low interrupt latency configuration, reset value 1 = Low interrupt latency configuration overridden. This feature: • disables the fast interrupt response introduced by setting the FI bit • disables *Hit-Under-Miss* (HUM) functionality • abandons restartable external accesses so that all external aborts to loads are precise.
[30]	FSD	Provides additional level of control for speculative operations, see *c1, Control Register* on page 3-44. Force speculative operations force the PC to a new value because of static, speculative, branch prediction: 0 = Enable force speculative operations, reset value 1 = Disable force speculative operations.
[29]	BFD	Disables branch folding. This behavior also depends on the SB and DB bits, [2:1] in this register, and the Z bit, see *c1, Control Register* on page 3-44: 0 = Branch folding is enabled, when branch prediction is enabled, reset value 1 = Branch folding is disabled.
[28]	PHD	Disables instruction prefetch halting on unconditional, unpredictable instructions that later result in a prefetch buffer flush. This prefetch halting is a power saving technique: 0 = Prefetch halting is enabled, reset value 1 = Prefetch halting is disabled.
[27:7]	-	UNP/SBZ
[6]	CZ	Controls the restriction of cache size to 16KB. This enables the processor to run software that does not support ARMv6 page coloring. When set the CZ bit does not effect the Cache Type Register. See *Restrictions on page table mappings page coloring* on page 6-41 for more information: 0 = Normal ARMv6 cache behavior, reset value 1 = Cache size limited to 16KB.
[5]	RV	Disables block transfer cache operations: 0 = Block transfer cache operations enabled, reset value 1 = Block transfer cache operations disabled.

Table 3-42 Auxiliary Control Register bit functions (continued)

Bits	Field name	Function
[4]	RA	Disables clean entire data cache: 0 = Clean entire data cache enabled, reset value 1 = Clean entire data cache disabled.
[3]	TR	Enables MicroTLB random replacement strategy. This depends on the cache replacement strategy that the RR bit controls, see *c1, Control Register* on page 3-44. The MicroTLB strategy is only random when the cache strategy is random: 0 = MicroTLB replacement is Round Robin, reset value 1 = MicroTLB replacement is Random if cache replacement is also Random.
[2]	SB	Enables static branch prediction. This depends on program flow prediction that the Z bit enables, see *c1, Control Register* on page 3-44: 0 = Static branch prediction disabled 1 = Static branch prediction enabled, if the Z bit is set. The reset value is 1.
[1]	DB	Enables dynamic branch prediction. This depends on program flow prediction that the Z bit enables, see *c1, Control Register* on page 3-44: 0 = Dynamic branch prediction disabled 1 = Dynamic branch prediction enabled, if the Z bit is set. The reset value is 1.
[0]	RS	Enables the return stack. This depends on program flow prediction that the Z bit enables, see *c1, Control Register* on page 3-44: 0 = Return stack is disabled 1 = Return stack is enabled, if the Z bit is set. The reset value is 1.

Table 3-43 lists the results of attempted access for each mode.

Table 3-43 Results of access to the Auxiliary Control Register

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Data	Data	Undefined exception	Undefined exception

To use the Auxiliary Control Register you must use a read modify write technique. To access the Auxiliary Control Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c1
- CRm set to c0
- Opcode_2 set to 1.

For example:

```
MRC p15, 0, <Rd>, c1, c0, 1    ; Read Auxiliary Control Register
MCR p15, 0, <Rd>, c1, c0, 1    ; Write Auxiliary Control Register
```

3.2.9 c1, Coprocessor Access Control Register

The purpose of the Coprocessor Access Control Register is to set access rights for the coprocessors CP0 through CP13. This register has no effect on access to CP14, the debug control coprocessor, or CP15, the system control coprocessor. This register also provides a means for software to determine if any particular coprocessor, CP0-CP13, exists in the system.

The Coprocessor Access Control Register is:

- in CP15 c1
- a 32-bit read/write register common to Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-28 shows the arrangement of bits in the register.

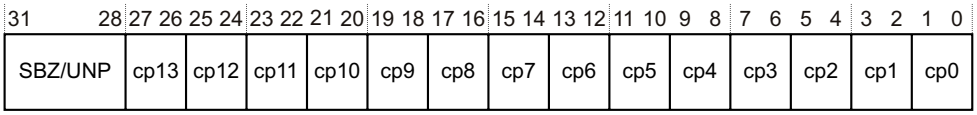

Figure 3-28 Coprocessor Access Control Register format

Table 3-44 lists how the bit values correspond with the Coprocessor Access Control Register functions.

Table 3-44 Coprocessor Access Control Register bit functions

Bits	Field name	Function
[31:28]	-	UNP/SBZ.
-	cp<n>[a]	Defines access permissions for each coprocessor. Access denied is the reset condition. Access denied is the behavior for non-existent coprocessors: b00 = Access denied, reset value. Attempted access generates an Undefined exception b01 = Privileged mode access only b10 = Reserved. b11 = Privileged and User mode access.

a. n is the coprocessor number between 0 and 13.

Access to coprocessors in the Non-secure world depends on the permissions set in the *c1, Non-Secure Access Control Register* on page 3-55.

Attempts to read or write the Coprocessor Access Control Register access bits depend on the corresponding bit for each coprocessor in *c1, Non-Secure Access Control Register* on page 3-55. Table 3-45 lists the results of attempted access to coprocessor access bits for each mode.

Table 3-45 Results of access to the Coprocessor Access Control Register

Corresponding bit in Non-Secure Access Control Register	Secure Privileged		Non-secure Privileged		User
	Read	Write	Read	Write	
0	Data	Data	b00	Ignored	Undefined exception
1	Data	Data	Data	Data	Undefined exception

To use the Coprocessor Access Control Register read or write CP15 with:

- Opcode_1 set to 0

- CRn set to c1
- CRm set to c0
- Opcode_2 set to 2.

For example:

```
MRC p15, 0, <Rd>, c1, c0, 2    ; Read Coprocessor Access Control Register
MCR p15, 0, <Rd>, c1, c0, 2    ; Write Coprocessor Access Control Register
```

You must perform an *Instruction Memory Barrier* (IMB) sequence immediately after an update of the Coprocessor Access Control Register, see *Memory Barriers* on page 5-8. You must not attempt to execute any instructions that are affected by the change of access rights between the IMB sequence and the register update.

To determine if any particular coprocessor exists in the system write the access bits for the coprocessor of interest with a value other than b00. If the coprocessor does not exist in the system the access rights remain set to b00.

3.2.10 c1, Secure Configuration Register

The purpose of the Secure Configuration Register is to define:

- the current world as Secure or Non-secure
- the world in which the core executes exceptions
- the ability to modify the A and I bits in the CPSR in the Non-secure world.

The Secure Configuration Register is:

- in CP15 c1
- a 32 bit read/write register
- accessible in Secure privileged modes only.

Figure 3-29 shows the arrangement of bits in the register.

Figure 3-29 Secure Configuration Register format

Table 3-46 lists how the bit values correspond with the Secure Configuration Register functions.

Table 3-46 Secure Configuration Register bit functions

Bits	Field name	Function
[31:7]	-	UNP/SBZ.
[6]	nET	The Early Termination bit is not implemented in ARM1176JZF-S processors. UNP/SBZ.
[5]	AW	Determines if the A bit in the CPSR can be modified when in the Non-secure world: 0 = Disable modification of the A bit in the CPSR in the Non-secure world, reset value 1 = Enable modification of the A bit in the CPSR in the Non-secure world.
[4]	FW	Determines if the F bit in the CPSR can be modified when in the Non-secure world: 0 = Disable modification of the F bit in the CPSR in the Non-secure world, reset value 1 = Enable modification of the F bit in the CPSR in the Non-secure world.

Table 3-46 Secure Configuration Register bit functions (continued)

Bits	Field name	Function
[3]	EA	Determines External Abort behavior for Secure and Non-secure worlds: 0 = Branch to abort mode on an External Abort exception, reset value 1 = Branch to Secure Monitor mode on an External Abort exception.
[2]	FIQ	Determines FIQ behavior for Secure and Non-secure worlds: 0 = Branch to FIQ mode on an FIQ exception, reset value 1 = Branch to Secure Monitor mode on an FIQ exception.
[1]	IRQ	Determines IRQ behavior for Secure and Non-secure worlds: 0 = Branch to IRQ mode on an IRQ exception, reset value 1 = Branch to Secure Monitor mode on an IRQ exception.
[0]	NS bit	Defines the world for the processor: 0 = Secure, reset value 1 = Non-secure.

Note

When the core runs in Secure Monitor mode the state is considered Secure regardless of the state of the NS bit. However, Monitor mode code can access nonsecure banked copies of registers if the NS bit is set to 1. See the *ARM Architecture Reference Manual* for information on the effect of the Security Extensions on the CP15 registers.

The permutations of the bits in the Secure Configuration Register have certain security implications. Table 3-47 lists the results for combinations of the FW and FIQ bits.

Table 3-47 Operation of the FW and FIQ bits

FW	FIQ	Function
1	0	FIQs handled locally.
0	1	FIQs can be configured to give deterministic Secure interrupts.
1	1	Non-secure world able to make denial of service attack, avoid use of this function.
0	0	Avoid because the core might enter an infinite loop for Non-secure FIQ.

Table 3-48 lists the results for combinations of the AW and EA bits.

Table 3-48 Operation of the AW and EA bits

AW	EA	Function
1	0	Aborts handled locally.
0	1	All external aborts trapped to Secure Monitor.
1	1	All external imprecise data aborts trapped to Secure Monitor but the Non-secure world can hide Secure aborts from the Secure Monitor, avoid use of this function.
0	0	Avoid because the core can unexpectedly enter an abort mode in the Non-secure world.

For more details on the use of Secure Monitor mode, see *The NS bit and Secure Monitor mode* on page 2-4.

To use the Secure Configuration Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c1
- CRm set to c1
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c1, c1, 0    ; Read Secure Configuration Register data
MCR p15, 0, <Rd>, c1, c1, 0    ; Write Secure Configuration Register data
```

An attempt to access the Secure Configuration Register from any state other than Secure privileged results in an Undefined exception.

3.2.11 c1, Secure Debug Enable Register

The purpose of the Secure Debug Enable Register is to provide control of permissions for debug in Secure User mode, see Chapter 13 *Debug*.

Table 3-49 lists the purposes of the individual bits in the Secure Debug Enable Register.

The Secure Debug Enable Register is:

- in CP15 c1
- a 32 bit register in the Secure world only
- accessible in Secure privileged modes only.

Figure 3-30 shows the arrangement of bits in the register.

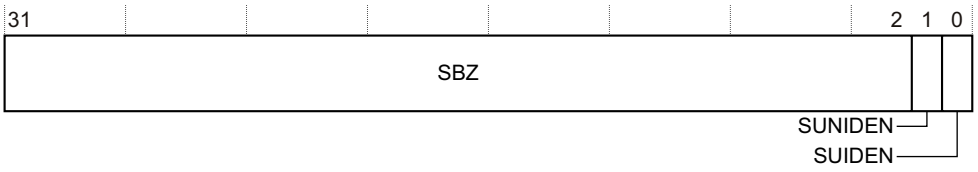

Figure 3-30 Secure Debug Enable Register format

Table 3-49 lists how the bit values correspond with the Secure Debug Enable Register functions.

Table 3-49 Secure Debug Enable Register bit functions

Bits	Field name	Function
[31:2]	-	This field is UNP when read. Write as the existing value.
[1]	SUNIDEN	Enables Secure User non-invasive debug: 0 = Non-invasive debug is not permitted in Secure User mode, reset value 1 = Non-invasive debug is permitted in Secure User mode.
[0]	SUIDEN	Enables Secure User invasive debug: 0 = Invasive debug is not permitted in Secure User mode, reset value 1 = Invasive debug is permitted in Secure User mode.

Table 3-50 lists the results of attempted access for each mode.

Table 3-50 Results of access to the Coprocessor Access Control Register

Secure Privileged		Non-secure Privileged	User
Read	**Write**		
Data	Data	Undefined exception	Undefined exception

To use the Secure Debug Enable Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c1
- CRm set to c1
- Opcode_2 set to 1.

For example:

```
MRC p15, 0, <Rd>, c1, c1, 1    ; Read Secure Debug Enable Register
MCR p15, 0, <Rd>, c1, c1, 1    ; Write Secure Debug Enable Register
```

3.2.12 c1, Non-Secure Access Control Register

The purpose of the Non-Secure Access Control Register is to define the Non-secure access permission for:

- coprocessors
- cache lockdown registers
- TLB lockdown registers
- internal DMA.

Note

This register has no effect on Non-secure access permissions for the debug control coprocessor, CP14, or the system control coprocessor, CP15.

The Non-Secure Access Control Register is:

- in CP15 c1
- a 32 bit register:
  - read/write in the Secure world
  - read only in the Non-secure world
- only accessible in privileged modes.

Figure 3-31 on page 3-56 shows the arrangement of bits in the register.

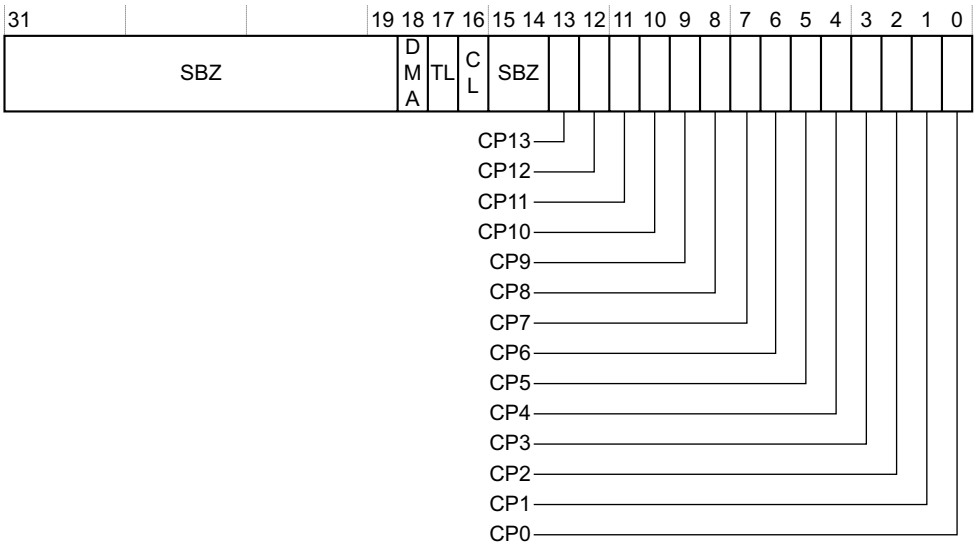

Figure 3-31 Non-Secure Access Control Register format

Table 3-51 lists how the bit values correspond with the Non-Secure Access Control Register functions.

Table 3-51 Non-Secure Access Control Register bit functions

Bits	Field name	Function
[31:19]	-	Reserved. UNP/SBZ.
[18]	DMA	Reserves the DMA channels and registers for the Secure world and determines the page tables, Secure or Non-secure, to use for DMA transfers. For details, see *DMA* on page 7-10: 0 = DMA reserved for the Secure world only and the Secure page tables are used for DMA transfers, reset value 1 = DMA can be used by the Non-secure world and the Non-secure page tables are used for DMA transfers.
[17]	TL	Prevents operations in the Non-secure world from locking page tables in TLB lockdown entries. The Invalidate Single Entry or Invalidate ASID match operations can match a TLB lockdown entry but an Invalidate All operation only applies to unlocked entries: 0 = Reserve TLB Lockdown registers for Secure operation only, reset value 1 = TLB Lockdown registers available for Secure and Non-secure operation.
[16]	CL	Prevents operations in the Non-secure world from changing cache lockdown entries: 0 = Reserve cache lockdown registers for Secure operation only, reset value 1 = Cache lockdown registers available for Secure and Non-secure operation.
[15:14]	-	Reserved. UNP/SBZ.
[13:0]	CPn[a]	Determines permission to access the given coprocessor in the Non-secure world: 0 = Secure access only, reset value 1 = Secure or Non-secure access.

a. n is the coprocessor number from 0 to 13.

To use the Non-Secure Access Control Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c1
- CRm set to c1
- Opcode_2 set to 2.

For example:

```
MRC p15, 0, <Rd>, c1, c1, 2    ; Read Non-Secure Access Control Register data
MCR p15, 0, <Rd>, c1, c1, 2    ; Write Non-Secure Access Control Register data
```

Table 3-52 lists the results of attempted access for each mode.

Table 3-52 Results of access to the Auxiliary Control Register

Secure Privileged		Non-secure Privileged		User
Read	**Write**	**Read**	**Write**	
Data	Data	Data	Undefined exception	Undefined exception

3.2.13 c2, Translation Table Base Register 0

The purpose of the Translation Table Base Register 0 is to hold the physical address of the first-level translation table.

You use Translation Table Base Register 0 for process-specific addresses, where each process maintains a separate first-level page table. On a context switch you must modify both Translation Table Base Register 0 and the Translation Table Base Control Register, if appropriate.

Table 3-53 on page 3-58 lists the purposes of the individual bits in the Translation Table Base Register 0.

The Translation Table Base Register 0 is:

- in CP15 c2
- a 32 bit read/write register banked for Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-32 shows the bit arrangement for the Translation Table Base Register 0.

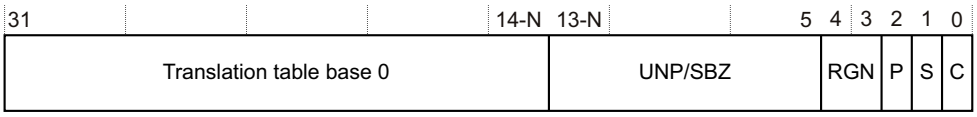

Figure 3-32 Translation Table Base Register 0 format

Table 3-53 lists how the bit values correspond with the Translation Table Base Register 0 functions.

Table 3-53 Translation Table Base Register 0 bit functions

Bits	Field name	Function
[31:14-N][a]	Translation table base 0	Holds the translation table base address, the physical address of the first level translation table. The reset value is 0.
[13-N:5][a]	-	UNP/SBZ.
[4:3]	RGN	Indicates the Outer cacheable attributes for page table walking: b00 = Outer Noncacheable, reset value b01 = Write-back, Write Allocate b10 = Write-through, No Allocate on Write b11 = Write-back, No Allocate on Write.
[2]	P	If the processor supports ECC, it indicates to the memory controller it is enabled or disabled. For ARM1176JZF-S processors this is 0: 0 = *Error-Correcting Code* (ECC) is disabled, reset value 1 = ECC is enabled.
[1]	S	Indicates the page table walk is to Non-Shared or to Shared memory: 0 = Non-Shared, reset value 1 = Shared.
[0]	C	Indicates the page table walk is Inner Cacheable or Inner Noncacheable: 0 = Inner noncacheable, reset value 1 = Inner cacheable.

a. For an explanation of N see *c2, Translation Table Base Control Register* on page 3-60.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Table 3-54 lists the results of attempted access for each mode.

Table 3-54 Results of access to the Translation Table Base Register 0

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception

A write to the Translation Table Base Register 0 updates the address of the first level translation table from the value in bits [31:7] of the written value, to account for the maximum value of 7 for N. The number of bits of this address that the processor uses, and therefore, the required alignment of the first level translation table, depends on the value of N, see *c2, Translation Table Base Control Register* on page 3-60.

A read from the Translation Table Base Register 0 returns the complete address of the first level translation table in bits [31:7] of the read value, regardless of the value of N.

To use the Translation Table Base Register 0 read or write CP15 c2 with:

- Opcode_1 set to 0
- CRn set to c2
- CRm set to c0

- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c2, c0, 0    ; Read Translation Table Base Register 0
MCR p15, 0, <Rd>, c2, c0, 0    ; Write Translation Table Base Register 0
```

Note

The ARM1176JZF-S processor cannot page table walk from level one cache. Therefore, if C is set to 1, to ensure coherency, you must either store page tables in Inner write-through memory or, if in Inner write-back, you must clean the appropriate cache entries after modification so that the mechanism for the hardware page table walks sees them.

3.2.14 c2, Translation Table Base Register 1

The purpose of the Translation Table Base Register 1 is to hold the physical address of the first-level table. The expected use of the Translation Table Base Register 1 is for OS and I/O addresses.

Table 3-55 lists the purposes of the individual bits in the Translation Table Base Register 1.

The Translation Table Base Register 1 is:

- in CP15 c2
- a 32 bit read/write register banked for Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-33 shows the bit arrangement for the Translation Table Base Register 1.

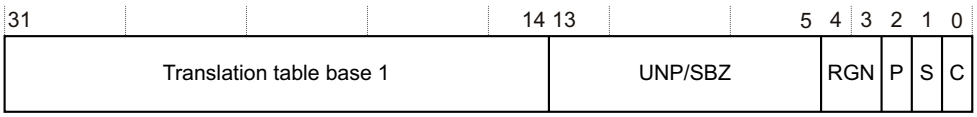

Figure 3-33 Translation Table Base Register 1 format

Table 3-55 lists how the bit values correspond with the Translation Table Base Register 1 functions.

Table 3-55 Translation Table Base Register 1 bit functions

Bits	Field name	Function
[31:14]	Translation table base 1	Holds the translation table base address, the physical address of the first level translation table. The reset value is 0.
[13:5]	-	UNP/SBZ.
[4:3]	RGN	Indicates the Outer cacheable attributes for page table walking: b00 = Outer Noncacheable, reset value b01 = Write-back, Write Allocate b10 = Write-through, No Allocate on Write b11 = Write-back, No Allocate on Write.

Table 3-55 Translation Table Base Register 1 bit functions (continued)

Bits	Field name	Function
[2]	P	If the processor supports ECC, it indicates to the memory controller it is enabled or disabled. For ARM1176JZF-S processors this is 0: 0 = *Error-Correcting Code* (ECC) is disabled, reset value 1 = ECC is enabled.
[1]	S	Indicates the page table walk is to Non-Shared or to Shared memory: 0 = Non-Shared, reset value 1 = Shared.
[0]	C	Indicates the page table walk is Inner Cacheable or Inner Non Cacheable: 0 = Inner Noncacheable, reset value 1 = Inner Cacheable.

Table 3-56 lists the results of attempted access for each mode.

Table 3-56 Results of access to the Translation Table Base Register 1

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception

A write to the Translation Table Base Register 1 updates the address of the first level translation table from the value in bits [31:14] of the written value. Bits [13:5] Should Be Zero. The Translation Table Base Register 1 must reside on a 16KB page boundary.

To use the Translation Table Base Register 1 read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c2
- CRm set to c0
- Opcode_2 set to 1.

For example:

```
MRC p15, 0, <Rd>, c2, c0, 1    ; Read Translation Table Base Register 1
MCR p15, 0, <Rd>, c2, c0, 1    ; Write Translation Table Base Register 1
```

Note

The ARM1176JZF-S processor cannot page table walk from level one cache. Therefore, if C is set to 1, to ensure coherency, you must either store page tables in Inner write-through memory or, if in Inner write-back, you must clean the appropriate cache entries after modification so that the mechanism for the hardware page table walks sees them.

3.2.15 c2, Translation Table Base Control Register

The purpose of the Translation Table Base Control Register is to determine if a page table miss for a specific VA uses, for its page table walk, either:

- Translation Table Base Register 0. The recommended use is for task-specific addresses
- Translation Table Base Register 1. The recommended use is for operating system and I/O addresses.

Table 3-57 lists the purposes of the individual bits in the Translation Table Base Control Register.

The Translation Table Base Control Register is:

- in CP15 c2
- a 32 bit read/write register banked for Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-34 shows the bit arrangement for the Translation Table Base Register 1.

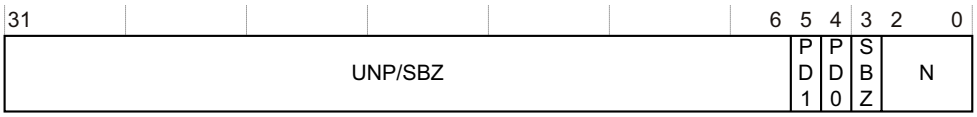

Figure 3-34 Translation Table Base Control Register format

Table 3-57 lists how the bit values correspond with the Translation Table Base Register 0 functions.

Table 3-57 Translation Table Base Control Register bit functions

Bits	Field name	Function
[31:6]	-	UNP/SBZ.
[5]	PD1	Specifies occurrence of a page table walk on a TLB miss when using Translation Table Base Register 1. When page table walk is disabled, a Section Fault occurs instead on a TLB miss: 0 = The processor performs a page table walk on a TLB miss, with Secure or Non-secure privilege appropriate to the current world. This is the reset value 1 = The processor does not perform a page table walk. If a TLB miss occurs with Translation Table Base Register 1 in use, the processor returns a Section Translation Fault.
[4]	PD0	Specifies occurrence of a page table walk on a TLB miss when using Translation Table Base Register 0. When page table walk is disabled, a Section Fault occurs instead on a TLB miss: 0 = The processor performs a page table walk on a TLB miss, with Secure or Non-secure privilege appropriate to the current world. This is the reset value 1 = The processor does not perform a page table walk. If a TLB miss occurs with Translation Table Base Register 0 in use, the processor returns a Section Translation Fault.
[3]	-	UNP/SBZ.
[2:0]	N	Specifies the boundary size of Translation Table Base Register 0: b000 = 16KB, reset value b001 = 8KB b010 = 4KB b011 = 2KB b100 = 1KB b101 = 512-byte b110 = 256-byte b111 = 128-byte.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Table 3-58 lists the results of attempted access for each mode.

Table 3-58 Results of access to the Translation Table Base Control Register

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception

To use the Translation Table Base Control Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c2
- CRm set to c0
- Opcode_2 set to 2.

For example:

```
MRC p15, 0, <Rd>, c2, c0, 2    ; Read Translation Table Base Control Register
MCR p15, 0, <Rd>, c2, c0, 2    ; Write Translation Table Base Control Register
```

A translation table base register is selected like this:

- If N is set to 0, always use Translation Table Base Register 0. This is the default case at reset. It is backwards compatible with ARMv5 and earlier processors.
- If N is set greater than 0, and bits [31:32-N] of the VA are all 0, use Translation Table Base Register 0, otherwise use Translation Table Base Register 1. N must be in the range 0-7.

Note

The ARM1176JZF-S processor cannot page table walk from level one cache. Therefore, if C is set to 1, to ensure coherency, you must either store page tables in Inner write-through memory or, if in Inner write-back, you must clean the appropriate cache entries after modification so that the mechanism for the hardware page table walks sees them.

3.2.16 c3, Domain Access Control Register

The purpose of the Domain Access Control Register is to hold the access permissions for a maximum of 16 domains.

Table 3-59 lists the purposes of the individual bits in the Domain Access Control Register.

The Domain Access Control Register is:

- in CP15 c3
- a 32-bit read/write register banked for Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-35 shows the bit arrangement of the Domain Access Control Register.

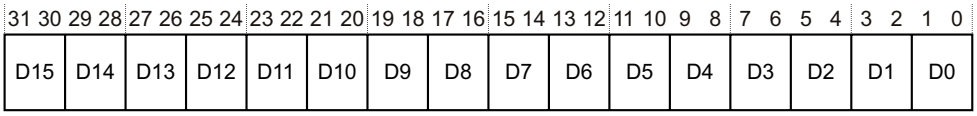

Figure 3-35 Domain Access Control Register format

Table 3-59 lists how the bit values correspond with the Domain Access Control Register functions.

Table 3-59 Domain Access Control Register bit functions

Bits	Field name	Function
-	D<n>[a]	The purpose of the fields D15-D0 in the register is to define the access permissions for each one of the 16 domains. These domains can be either sections, large pages or small pages of memory: b00 = No access, reset value. Any access generates a domain fault. b01 = Client. Accesses are checked against the access permission bits in the TLB entry. b10 = Reserved. Any access generates a domain fault. b11 = Manager. Accesses are not checked against the access permission bits in the TLB entry, so a permission fault cannot be generated.

a. n is the Domain number in the range between 0 and 15

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Table 3-60 lists the results of attempted access for each mode.

Table 3-60 Results of access to the Domain Access Control Register

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception

To use the Domain Access Control Register read or write CP15 c3 with:

- Opcode_1 set to 0
- CRn set to c3
- CRm set to c0
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c3, c0, 0    ; Read Domain Access Control Register
MCR p15, 0, <Rd>, c3, c0, 0    ; Write Domain Access Control Register
```

3.2.17 c5, Data Fault Status Register

The purpose of the Data Fault Status Register is to hold the source of the last data fault.

Table 3-61 lists the purposes of the individual bits in the Data Fault Status Register.

The Data Fault Status Register is:

- in CP15 c5
- a 32-bit read/write register banked for Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-36 shows the bit arrangement in the Data Fault Status Register.

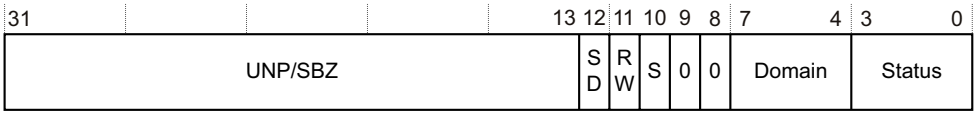

Figure 3-36 Data Fault Status Register format

Table 3-61 shows how the bit values correspond with the Data Fault Status Register functions.

Table 3-61 Data Fault Status Register bit functions

Bits	Field name	Function
[31:13]	-	UNP/SBZ.
[12]	SD	Indicates if an AXI Decode or Slave error caused an abort. This is only valid for external aborts. For all other aborts this Should Be Zero. See *Fault status and address* on page 6-34: 0 = AXI Decode error caused the abort, reset value 1 = AXI Slave error caused the abort.
[11]	RW	Indicates whether a read or write access caused an abort: 0 = Read access caused the abort, reset value 1 = Write access caused the abort.
[10]	S	Part of the Status field. See Bits [3:0] in this table. The reset value is 0.
[9:8]	-	Always read as 0. Writes ignored.

Table 3-61 Data Fault Status Register bit functions (continued)

Bits	Field name	Function
[7:4]	Domain	Indicates the domain from the 16 domains, D15-D0, is accessed when a data fault occurs. Takes values 0-15. The reset value is 0.
[3:0] with bit[10] = 0	Status	Indicates type of fault generated. See *Fault status and address* on page 6-34 for full details of Domain and FAR validity, and priorities: b0000 = no function, reset value b0001 = Alignment fault b0010 = Instruction debug event fault b0011 = Access Bit fault on Section b0100 = Instruction cache maintenance operation fault b0101 = Translation Section fault b0110 = Access Bit fault on Page b0111 = Translation Page fault b1000 = Precise external abort b1001 = Domain Section fault b1010 = no function b1011 = Domain Page fault b1100 = External abort on translation, first level b1101 = Permission Section fault b1110 = External abort on translation, second level b1111 = Permission Page fault.
[3:0] with bit[10] = 1	Status	Indicates type of fault generated. See *Fault status and address* on page 6-34 for full details of Domain and FAR validity, and priorities: b0000 = no function, reset value b0001 = no function b0010 = no function b0011 = no function b0100 = no function b0101 = no function b0110 = Imprecise external abort b0111 = no function b1000 = no function b1001 = no function b1010 = no function b1011 = no function b1100 = no function b1101 = no function b1110 = no function b1111 = no function.

Table 3-62 lists the results of attempted access for each mode.

Table 3-62 Results of access to the Data Fault Status Register

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception

Note

When the SCR EA bit is set, see *c1, Secure Configuration Register* on page 3-52, the processor writes to the Secure Data Fault Status Register on a Secure Monitor entry caused by an external abort.

To use the Data Fault Status Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c5
- CRm set to c0
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c5, c0, 0    ; Read Data Fault Status Register
MCR p15, 0, <Rd>, c5, c0, 0    ; Write Data Fault Status Register
```

3.2.18 c5, Instruction Fault Status Register

The purpose of the *Instruction Fault Status Register* (IFSR) is to hold the source of the last instruction fault.

Table 3-63 on page 3-67 lists the purposes of the individual bits in IFSR.

The Instruction Fault Status Register is:

- in CP15 c5
- a 32-bit read/write register banked for Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-37 shows the bit arrangement of the Instruction Fault Status Register.

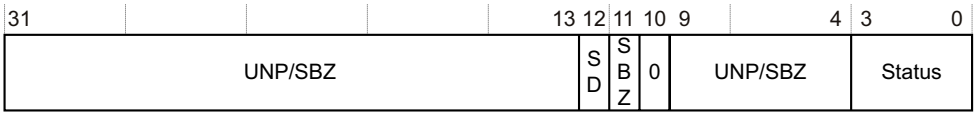

Figure 3-37 Instruction Fault Status Register format

Table 3-63 lists how the bit values correspond with the Instruction Fault Status Register functions.

Table 3-63 Instruction Fault Status Register bit functions

Bits	Field name	Function
[31:13]	-	UNP/SBZ.
[12]	SD	Indicates whether an AXI Decode or Slave error caused an abort. This bit is only valid for external aborts. For all other aborts this bit Should Be Zero. See *Fault status and address* on page 6-34: 0 = AXI Decode error caused the abort, reset value 1 = AXI Slave error caused the abort.
[11]	-	UNP/SBZ.
[10]	-	Part of the Status field, see bits [3:0] in this table. Always 0.
[9:4]	-	UNP/SBZ.
[3:0] with bit[10] = 0	Status	Indicates type of fault generated. See *Fault status and address* on page 6-34 for full details of Domain and FAR validity, and priorities: b0000 = no function, reset value b0001= Alignment fault b0010 = Instruction debug event fault b0011 = Access Bit fault on Section b0100 = no function b0101 = Translation Section fault b0110 = Access Bit fault on Page b0111 = Translation Page fault b1000 = Precise external abort b1001 = Domain Section fault b1010 = no function b1011 = Domain Page fault b1100 = External abort on translation, first level b1101 = Permission Section fault b1110 = External abort on translation, second level b1111 = Permission Page fault.

Table 3-64 lists the results of attempted access for each mode.

Table 3-64 Results of access to the Instruction Fault Status Register

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception

Note

When the SCR EA bit is set, see *c1, Secure Configuration Register* on page 3-52, the processor writes to the Secure Instruction Fault Status Register on a Secure Monitor entry caused by an external abort.

To use the IFSR read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c5
- CRm set to c0
- Opcode_2 set to 1.

For example:

```
MRC p15, 0, <Rd>, c5, c0, 1    ; Read Instruction Fault Status Register
MCR p15, 0, <Rd>, c5, c0, 1    ; Write Instruction Fault Status Register
```

3.2.19 c6, Fault Address Register

The purpose of the *Fault Address Register* (FAR) is to hold the *Modified Virtual Address* (MVA) of the fault when a precise abort occurs.

The FAR is:

- in CP15 c6
- a 32-bit read/write register banked for Secure and Non-secure worlds
- accessible in privileged modes only.

The Fault Address Register bits [31:0] contain the MVA that the precise abort occurred on. The reset value is 0.

Table 3-65 lists the results of attempted access for each mode.

Table 3-65 Results of access to the Fault Address Register

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception

To use the FAR read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c6
- CRm set to c0
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c6, c0, 0    ; Read Fault Address Register
MCR p15, 0, <Rd>, c6, c0, 0    ; Write Fault Address Register
```

A write to this register sets the FAR to the value of the data written. This is useful for a debugger to restore the value of the FAR.

The ARM1176JZF-S processor also updates the FAR on debug exception entry because of watchpoints, see *Effect of a debug event on CP15 registers* on page 13-34 for more details.

3.2.20 c6, Watchpoint Fault Address Register

Access to the Watchpoint Fault Address register through the system control coprocessor is deprecated, see *CP14 c6, Watchpoint Fault Address Register (WFAR)* on page 13-12.

3.2.21 c6, Instruction Fault Address Register

The purpose of the *Instruction Fault Address Register* (IFAR) is to hold the address of instructions that cause a prefetch abort.

The IFAR is:

- in CP15 c6
- a 32-bit read/write register banked for Secure and Non-secure worlds
- accessible in privileged modes only.

The Instruction Fault Address Register bits [31:0] contain the Instruction Fault MVA. The reset value is 0.

Table 3-66 lists the results of attempted access for each mode.

Table 3-66 Results of access to the Instruction Fault Address Register

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception

To use the IFAR read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c6
- CRm set to c0
- Opcode_2 set to 2.

For example:

```
MRC p15, 0, <Rd>, c6, c0, 2    ; Read Instruction Fault Address Register
MCR p15, 0, <Rd>, c6, c0, 2    ; Write Instruction Fault Address Register
```

A write to this register sets the IFAR to the value of the data written. This is useful for a debugger to restore the value of the IFAR.

3.2.22 c7, Cache operations

The purpose of c7 is to:

- control these operations:
  - clean and invalidate instruction and data caches, including range operations
  - prefetch instruction cache line
  - Flush Prefetch Buffer
  - flush branch target address cache
  - virtual to physical address translation.
- implement the *Data Synchronization Barrier* (DSB) operation
- implement the *Data Memory Barrier* (DMB) operation

- implement the *Wait For Interrupt* clock control function.

Note

Cache operations also depend on:

- the C, W, I and RR bits, see *c1, Control Register* on page 3-44.
- the RA and RV bits, see *c1, Auxiliary Control Register* on page 3-48.

The following cache operations globally flush the BTAC:

- Invalidate Entire Instruction Cache
- Invalidate Both Caches.

c7 consists of one 32-bit register that performs 28 functions. Figure 3-38 shows the arrangement of the 24 functions in this group that operate with the MCR and MRC instructions.

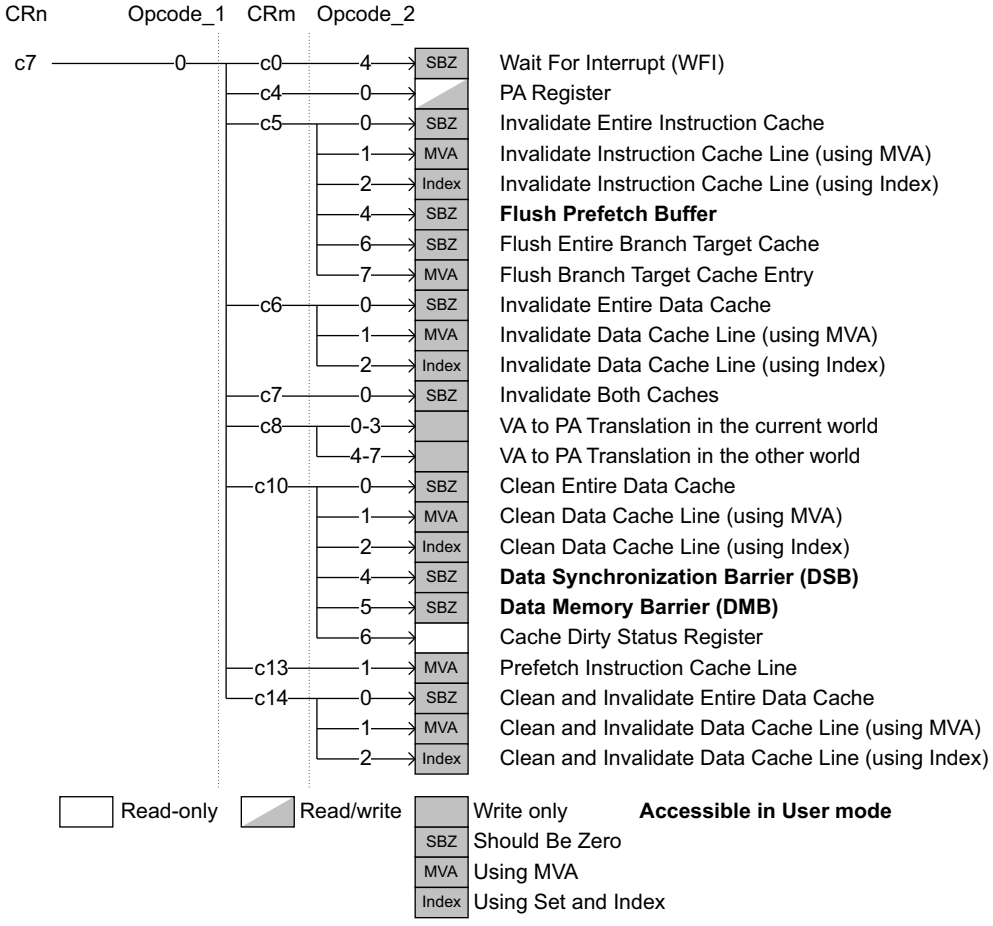

Figure 3-38 Cache operations

Figure 3-39 on page 3-71 shows the arrangement of the 4 functions in this group that operate with the MCRR instruction.

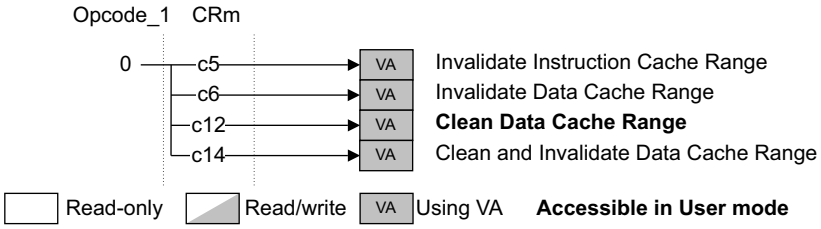

Figure 3-39 Cache operations with MCRR instructions

Note

- Writing c7 with a combination of CRm and Opcode_2 not listed in Figure 3-38 on page 3-70 or CRm not listed in Figure 3-39 results in an Undefined exception apart from the following operations, that are architecturally defined as unified cache operations and have no effect:
  - `MCR p15,0,<Rd>,c7,c7,{1-7}`
  - `MCR p15,0,<Rd>,c7,c11,{0-7}`
  - `MCR p15,0,<Rd>,c7,c15,{0-7}`.
- In the ARM1176JZF-S processor, reading from c7, except for reads from the Cache Dirty Status Register or PA Register, causes an Undefined instruction trap.
- Writes to the Cache Dirty Status Register cause an Undefined exception.
- If Opcode_1 = 0, these instructions are applied to a level one cache system. All other Opcode_1 values are reserved.
- All accesses to c7 can only be executed in a privileged mode of operation, except Data Synchronization Barrier, Flush Prefetch Buffer, Data Memory Barrier, and Clean Data Cache Range. These can be operated in User mode. Attempting to execute a privileged instruction in User mode results in the Undefined instruction trap being taken.

There are three ways to use c7:

- For the Cache Dirty Status Register, read c7 with the MRC instruction.
- For range operations use the MCRR instruction with the value of CRm to select the required operation.
- For all other operations use the MCR instruction to write to c7 with the combination of CRm and Opcode_2 to select the required operation.

  Depending on the operation you require set <Rd> for MCR instructions or <Rd> and <Rn> for MCRR instructions to:
  - *Virtual Address* (VA)
  - *Modified Virtual Address* (MVA)
  - Set and Index
  - Should Be Zero.

Invalidate, Clean, and Prefetch operations

The purposes of the invalidate, clean, and prefetch operations that c7 provides are to:

- Invalidate part or all of the Data or Instruction caches
- Clean part or all of the Data cache
- Clean and Invalidate part or all of the Data cache

- Prefetch code into the Instruction cache.

The terms used to describe the invalidate, clean, and prefetch operations are as defined in the *Caches and Write Buffers* chapter of the *ARM Architecture Reference Manual*.

For details of the behavior of c7 in the Secure and Non-secure worlds, see *TrustZone behavior* on page 3-77.

When it controls invalidate, clean, and prefetch operations c7 appears as a 32-bit write only register. There are four possible formats for the data that you write to the register that depend on the specific operation:

- Set and Index format
- MVA
- VA
- SBZ.

Set and Index format

Figure 3-40 shows the Set and Index format for invalidate and clean operations.

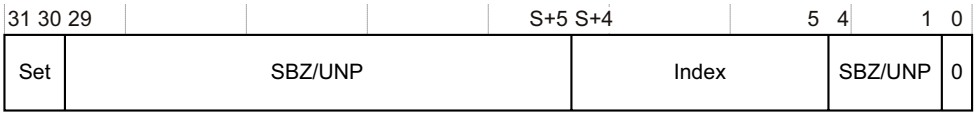

Figure 3-40 c7 format for Set and Index

Table 3-67 lists how the bit values correspond with the Cache Operation functions for Set and Index format operations.

Table 3-67 Functional bits of c7 for Set and Index

Bits	Field name	Function
[31:30]	Set	Selects the cache set to operate on, from the four cache sets. Value is the cache set number.
[29:S+5]	-	UNP/SBZ.
[S+4:5]	Index	Selects the cache line to operate on. Value is the index number.
[4:1]	-	SBZ.
[0]	0	For the ARM1176JZF-S, this Should Be Zero.

The value of S in Table 3-68 depends on the cache size. Table 3-68 lists the relationship of cache sizes and S.

Table 3-68 Cache size and S parameter dependency

Cache size	S
4KB	5
8KB	6
16KB	7
32KB	8
64KB	9

The value of S is given by:

$$S = \log_2\left(\frac{\text{cache size}}{\text{Associativity x line length in bytes}}\right)$$

See *c0, Cache Type Register* on page 3-21 for details of instruction and data cache size.

Note

If the data is stated to be Set and Index format, see Figure 3-40 on page 3-72, it identifies the cache line that the operation applies to by specifying the cache Set that it belongs to and what its Index is within the Set. The Set corresponds to the number of the cache way, and the Index number corresponds to the line number within a cache way.

MVA format

Figure 3-41 shows the MVA format for invalidate, clean, and prefetch operations.

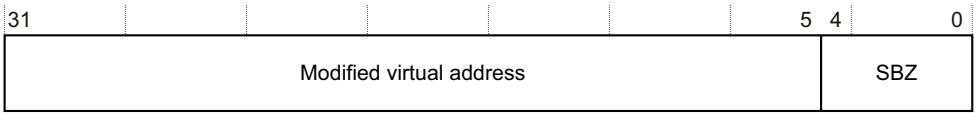

Figure 3-41 c7 format for MVA

Table 3-69 lists how the bit values correspond with the Cache Operation functions for MVA format operations.

Table 3-69 Functional bits of c7 for MVA

Bits	Field name	Function
[31:5]	MVA	Specifies address to invalidate, clean, or prefetch. Holds the MVA of the cache line.
[4:0]	-	Ignored. This means that the lower 5 bits of MVA are ignored and these bits are not used for the cache operations. Only the top bits are necessary to determine whether or not the cache line is present in the cache. Even if the MVA is not aligned to the cache line, the cache operation is performed by ignoring the lower 5 bits.

Note

- Invalidation and cleaning operations have no effect if they miss in the cache.
- If the corresponding entry is not in the TLB, these instructions can cause a TLB miss exception or hardware page table walk, depending on the miss handling mechanism.
- For the cache control operations, the MVAs that are passed to the cache are not translated by the FCSE extension.

VA format

Figure 3-42 shows the VA format for invalidate and clean operations. All VA format operations use the MCRR instruction.

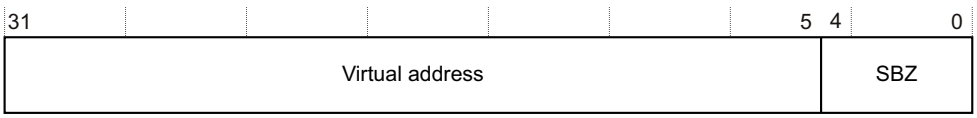

Figure 3-42 Format of c7 for VA

Table 3-70 lists how the bit values correspond with the Cache Operation functions for VA format operations.

Table 3-70 Functional bits of c7 for VA format

Bits	Field name	Function
[31:5]	Virtual address	Specifies the start or end address to invalidate or clean. Holds the true VA of the start or end of a memory block before any modification by FCSE.
[4:0]	-	SBZ.

You can perform invalidate, clean, and prefetch operations on:

- single cache lines
- entire caches
- address ranges in cache.

Note

- Clean, invalidate, and clean and invalidate operations apply regardless of the lock applied to entries.
- An explicit flush of the relevant lines in the branch target cache must be performed after invalidation of Instruction Cache lines or the results are Unpredictable. There is no impact on security. This is not required after an entire Instruction Cache invalidation because the entire branch target cache is flushed automatically.
- A small number of CP15 c7 operations can be executed by code while in User mode. Attempting to execute a privileged operation in User mode using CP15 c7 results in an Undefined instruction trap being taken.

To determine if the cache is dirty use the Cache Dirty Status Register, see *Cache Dirty Status Register* on page 3-78.

Entire cache

Table 3-71 lists the instructions and operations that you can use to clean and invalidate the entire cache.

Table 3-71 Cache operations for entire cache

Instruction	Data	Function
`MCR p15, 0, <Rd>, c7, c5, 0`	SBZ	Invalidate Entire Instruction Cache. Also flushes the branch target cache and globally flushes the BTAC.
`MCR p15, 0, <Rd>, c7, c6, 0`	SBZ	Invalidate Entire Data Cache.
`MCR p15, 0, <Rd>, c7, c7, 0`	SBZ	Invalidate Both Caches. Also flushes the branch target cache and globally flushes the BTAC.
`MCR p15, 0, <Rd>, c7, c10, 0`	SBZ	Clean Entire Data Cache.
`MCR p15, 0, <Rd>, c7, c14, 0`	SBZ	Clean and Invalidate Entire Data Cache.

Register c7 specifies operations for cleaning the entire Data Cache, and also for performing a clean and invalidate of the entire Data Cache. These are blocking operations that can be interrupted. If they are interrupted, the R14 value that is

captured on the interrupt is the address of the instruction that launched the cache clean operation + 4. This enables the standard return mechanism for interrupts to restart the operation.

If it is essential that the cache is clean, or clean and invalid, for a particular operation, the sequence of instructions for cleaning, or cleaning and invalidating, the cache for that operation must handle the arrival of an interrupt at any time when interrupts are not disabled. This is because interrupts can write to a previously clean cache. For this reason, the Cache Dirty Status Register indicates if the cache has been written to since the last clean of the cache was started, see *Cache Dirty Status Register* on page 3-78. You can interrogate the Cache Dirty Status Register to determine if the cache is clean, and if this is done while interrupts are disabled, the following operations can rely on having a clean cache. The following sequence shows this approach:

```
        ; interrupts are assumed to be enabled at this point
Loop1   MOV R1, #0
        MCR CP15, 0, R1, C7, C10, 0      ; Clean (or Clean & Invalidate) Cache
        MRS R2, CPSR
        CPSID iaf                        ; Disable interrupts
        MRC CP15, 0, R1, C7, C10, 6      ; Read Cache Dirty Status Register
        ANDS R1, R1, #1                  ; Check if it is clean
        BEQ UseClean
        MSR CPSR, R2                     ; Re-enable interrupts
        B Loop1                          ; - clean the cache again
UseClean Do_Clean_Operations             ; Perform whatever operation relies on
                                         ; the cache being clean/invalid.
                                         ; To reduce impact on interrupt
                                         ; latency, this sequence should be
                                         ; short
        MSR CPSR, R2                     ; Re-enable interrupts
```

The long cache clean operation is performed with interrupts enabled throughout this routine.

Single cache lines

There are two ways to perform invalidate or clean operations on cache lines:

- by use of Set and Index format
- by use of MVA format.

Table 3-72 lists the instructions and operations that you can use for single cache lines.

Table 3-72 Cache operations for single lines

Instruction	Data	Function
`MCR p15, 0, <Rd>, c7, c5, 1`	MVA	Invalidate Instruction Cache Line, using MVA
`MCR p15, 0, <Rd>, c7, c5, 2`	Set/Index	Invalidate Instruction Cache Line, using Index
`MCR p15, 0, <Rd>, c7, c6, 1`	MVA	Invalidate Data Cache Line, using MVA
`MCR p15, 0, <Rd>, c7, c6, 2`	Set/Index	Invalidate Data Cache Line, using Index
`MCR p15, 0, <Rd>, c7, c10, 1`	MVA	Clean Data Cache Line, using MVA
`MCR p15, 0, <Rd>, c7, c10, 2`	Set/Index	Clean Data Cache Line, using Index

Table 3-72 Cache operations for single lines (continued)

Instruction	Data	Function
MCR p15, 0, <Rd>, c7, c13, 1	MVA	Prefetch Instruction Cache Line
MCR p15, 0, <Rd>, c7, c14, 1	MVA	Clean and Invalidate Data Cache Line, using MVA
MCR p15, 0, <Rd>, c7, c14, 2	Set/Index	Clean and Invalidate Data Cache Line, using Index

Example 3-1 shows how to use Clean and Invalidate Data Cache Line with Set and Index to clean and invalidate one whole cache way, in this example, way 3. The example works with any cache size because it reads the cache size from the Cache Type Register.

Example 3-1 Clean and Invalidate Data Cache Line with Set and Index

```
                MRC     p15,0,R0,c0,c0,1        ; Read cache type reg
                AND     R0,R0,#0x1C0000         ; Extract D cache size
                MOV     R0,R0, LSR #18          ; Move to bottom bits
                ADD     R0,R0,#7                ; Get Index loop max
                MOV     r1,#3:SHL:30            ; Set up Set = 3
                MOV     R2,#0                   ; Set up Index counter
                MOV     R3,#1
                MOV     R3,R3, LSL R0           ; Set up Index loop max
index_loop
                ORR     R4,R2,r1                ; Set and Index format
                MCR     p15,0,R4,c7,c14,2       ; Clean&inval D cache line
                ADD     R2,R2,#1:SHL:5          ; Increment Index
                CMP     R2,R3                   ; Done all index values?
                BNE     index_loop              ; Loop until done
```

Address ranges

Table 3-73 lists the instructions and operations that you can use to clean and invalidate the address ranges in cache.

Table 3-73 Cache operations for address ranges

Instruction	Data	Function
MCRR p15,0,<End Address>,<Start Address>,c5	VA	Invalidate Instruction Cache Range
MCRR p15,0,<End Address>,<Start Address>,c6	VA	Invalidate Data Cache Range
MCRR p15,0,<End Address>,<Start Address>,c12	VA	Clean Data Cache Range[a]
MCRR p15,0,<End Address>,<Start Address>,c14	VA	Clean and Invalidate Data Cache Range

a. This operation is accessible in both User and privileged modes of operation. All other operations listed here are only accessible in privileged modes of operation.

The operations in Table 3-73 can only be performed using an MCRR or MCRR2 instruction, and all other operations to these registers are ignored.

The `End Address` and `Start Address` in Table 3-73 is the true VA before any modification by the *Fast Context Switch Extension* (FCSE). This address is translated by the FCSE logic. Each of the range operations operates between cache lines containing the `Start Address` and the `End Address`, inclusive of `Start Address` and `End Address`.

Because the least significant address bits are ignored, the transfer automatically adjusts to a line length multiple spanning the programmed addresses.

The `Start Address` is the first VA of the block transfer. It uses the VA bits [31:5]. The `End Address` is the VA where the block transfer stops. This address is at the start of the line containing the last address to be handled by the block transfer. It uses the VA bits [31:5].

If the Start Address is greater than the End Address the effect is architecturally Unpredictable. The ARM1176JZF-S processor does not perform cache operations in this case. All block transfers are interruptible. When Block transfers are interrupted, the R14 value that is captured is the address of the instruction that launched the block operation + 4. This enables the standard return mechanism for interrupts to restart the operation.

Exception behavior

The blocking block transfers cause a Data Abort on a translation fault if a valid page table entry cannot be fetched. The FAR indicates the address that caused the fault, and the DFSR indicates the reason for the fault.

TrustZone behavior

TrustZone affects cache operations as follows:

Secure world operations

In the Secure world cache operations can affect both Secure and Non-secure cache lines:

- Clean, invalidate, and clean and invalidate operations affect all cache lines regardless of their status as locked or unlocked.
- For clean, invalidate, and clean and invalidate operations with the Set and Index format, the selected cache line is affected regardless of the Secure tag.
- For MVA operations clean, invalidate, and clean and invalidate:
  - when the MVA is marked as Non-secure in the page table, only Non-secure entries are affected
  - when the MVA is marked as Secure in the page table, only Secure entries are affected.

Non-secure world operations

In the Non-secure world:

- Clean, invalidate, and clean and invalidate operations only affect Non-secure cache lines regardless of the method used.
- Any attempt to access Secure cache lines is ignored.
- Invalidate Entire Data Cache and Invalidate Both Caches operations cause an Undefined exception. This prevents invalidating lockdown entries that might be configured as Secure.
  - the Invalidate Both Caches operation globally flushes the BTAC.
- Invalidate Entire Instruction Cache operations:
  - cause an Undefined exception if lockdown entries are reserved for the Secure world
  - affect all Secure and Non-secure cache entries if the lockdown entries are not reserved for the Secure world
  - globally flush the BTAC.

Cache Dirty Status Register

The purpose of the Cache Dirty Status Register is to indicate when the Cache is dirty.

The Cache Dirty Status Register is:
- in CP15 c7
- a 32-bit read only register, banked for Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-43 shows the arrangement of bits in the Cache Dirty Status Register.

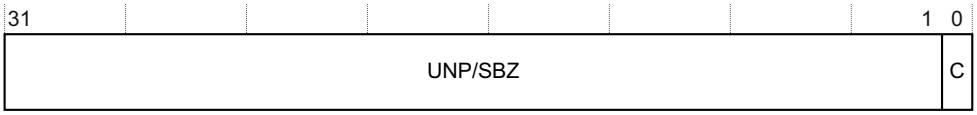

Figure 3-43 Cache Dirty Status Register format

Table 3-74 lists how the bit value corresponds with the Cache Dirty Status Register function.

Table 3-74 Cache Dirty Status Register bit functions

Bits	Field name	Function
[31:1]	-	UNP/SBZ.
[0]	C	The C bit indicates if the cache is dirty. 0 = indicates that no write has hit the cache since the last cache clean, clean and invalidate, or invalidate all operation, or reset, successfully left the cache clean. This is the reset value. 1 = indicates that the cache might contain dirty data.

The Cache Dirty Status Register behaves in this way with regard to the Secure and Non-secure cache:

- clean, invalidate, and clean and invalidate operations of the whole cache in the Non-secure world clear the Non-secure Cache Dirty Status Register
- clear, invalidate, and clean and invalidate operations of the whole cache in the Secure world clear both the Secure and Non-secure Cache Dirty Status Registers
- if the core is in the Non-secure world or targets Non-secure data from the Secure world, stores that write a dirty bit in the cache set both the Secure and the Non-secure Cache Dirty Status Register
- all stores that write a dirty bit in the cache set the Secure Cache Dirty Status Register.

All writes and User mode reads of the Cache Dirty Status Register cause an Undefined exception.

To use the Cache Dirty Status Register read CP15 with:
- Opcode_1 set to 0
- CRn set to c7
- CRm set to c10
- Opcode_2 set to 6.

For example:

```
MRC p15, 0, <Rd>, c7, c10, 6    ; Read Cache Dirty Status Register.
```

Flush operations

Table 3-75 lists the flush operations and instructions available through c7.

Table 3-75 Cache operations flush functions

Instruction	Data	Function
`MCR p15, 0, <Rd>, c7, c5, 4`	SBZ	Flush Prefetch Buffer[a].
`MCR p15, 0, <Rd>, c7, c5, 6`	SBZ	Flush Entire Branch Target Cache[b].
`MCR p15, 0, <Rd>, c7, c5, 7`	MVA[c]	Flush Branch Target Cache Entry with MVA.

a. These operations are accessible in both User and privileged modes of operation. All other operations are only accessible in privileged modes of operation.
b. This operation is accessible in both Privileged and User modes of operation when in Debug state.
c. The range of MVA bits used in this function is different to the range of bits used in other functions that have MVA data.

The Flush Branch Target Entry using MVA operation uses a different MVA format to that used by Clean and Invalidate operations. Figure 3-44 shows the MVA format for the Flush Branch Target Entry operation.

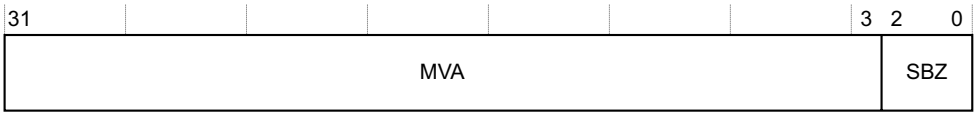

Figure 3-44 c7 format for Flush Branch Target Entry using MVA

Table 3-76 lists how the bit values correspond with the Flush Branch Target Entry using MVA functions.

Table 3-76 Flush Branch Target Entry using MVA bit functions

Bits	Field name	Function
[31:3]	MVA	Specifies address to flush. Holds the MVA of the Branch Target Cache line.
[2:0]	-	SBZ.

Note

The MVA does not have to be cache line aligned.

Flushing the prefetch buffer has the effect that all instructions occurring in program order after this instruction are fetched from the memory system after the execution of this instruction, including the level one cache or TCM. This operation is useful for ensuring the correct execution of self-modifying code. See *Explicit Memory Barriers* on page 6-25.

VA to PA translation operations

The purpose of the VA to PA translation operations is to provide a Secure means to determine address translation in the Secure and Non-secure worlds and for address translation between the Secure and Non-secure worlds. VA to PA translations operate through:

- *PA Register* on page 3-80

- *VA to PA translation in the current world* on page 3-82
- *VA to PA translation in the other world* on page 3-83.

PA Register

The purpose of the PA Register is to hold:

- the PA after a successful translation
- the source of the abort for an unsuccessful translation.

Table 3-77 lists the purpose of the bits of the PA Register for successful translations and Table 3-78 on page 3-81 lists the purpose of the bits of the PA Register for unsuccessful translations.

The PA Register is:

- in CP15 c7
- a 32 bit read/write register banked in Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-45 shows the format of the PA Register for successful translations.

Figure 3-45 PA Register format for successful translation

Figure 3-46 shows the format of the PA register for aborted translations.

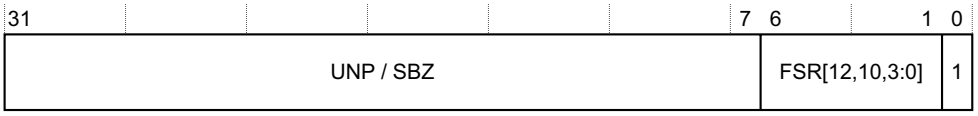

Figure 3-46 PA Register format for aborted translation

Table 3-77 lists the functional bits of the PA Register for successful translation.

Table 3-77 PA Register for successful translation bit functions

Bits	Field name	Function
[31:10]	PA	PA Translated physical address.
[9]	NS	Indicates the state of the NS Attribute bit in the page table: 0 = Secure memory 1 = Non-secure memory.
[8]	P	Not used in the ARM1176JZF-S processor. UNP/SBZ.
[7]	SH	Indicates shareable memory: 0 = Non-shared 1 = Shared.

Table 3-77 PA Register for successful translation bit functions (continued)

Bits	Field name	Function
[6:4]	INNER	Indicates the inner attributes from the page table: b000 = Noncacheable b001 = Strongly Ordered b010 = Reserved b011 = Device b100 = Reserved b101 = Reserved b110 = Inner Write-through, no allocate on write b111 = Inner Write-back, no allocate on write.
[3:2]	OUTER	Indicates the outer attributes from the page table: b00 = Noncacheable b01 = Write-back, allocate on write b10 = Write-through, no allocate on write b11 = Write-back, no allocate on write.
[1]	-	Reserved. UNP/SBZ.
[0]	-	Indicates that the translation succeeded: 0 = Translation successful.

Table 3-78 lists the functional bits of the PA Register for aborted translation.

Table 3-78 PA Register for unsuccessful translation bit functions

Bits	Field name	Function
[31:7]	-	UNP/SBZ.
[6:1]	FSR[12,10,3:0]	Holds the FSR bits for the aborted address, see *c5, Data Fault Status Register* on page 3-64 and *c5, Instruction Fault Status Register* on page 3-66. FSR bits [12], [10], and [3:0].
[0]	-	Indicates that the translation aborted: 1 = Translation aborted.

Attempts to access the PA Register in User mode results in an Undefined exception.

Note

The VA to PA translation can only generate an abort to the core if the operation failed because an external abort occurred on the possible page table request. In this case, the processor updates the Secure or Non-secure version of the PA register, depending on the Secure or Non-secure state of the core when the operation was issued. The processor also updates the Data Fault Status Register and the Fault Address Register:

- if the EA bit in the Secure Configuration Register is set, the Secure versions of the two registers are updated and the processor traps the abort into Secure Monitor mode
- if the EA bit in the Secure Configuration Register is not set, the processor updates the Secure or Non-secure versions of the two registers, depending on the Secure or Non-secure state of the core when the operation was issued.

For all other cases when the VA to PA operation fails, the processor only updates the PA register, Secure or Non-secure version, depending on the Secure or Non-secure state of the core when the operation was issued, with the Fault Status Register encoding and bit[0] set. The Data Fault Status Register and Fault Address Register remain unchanged and the processor does not send an abort to the core.

To use the PA Register read or write CP15 c7 with:

- Opcode_1 set to 0
- CRn set to c7
- CRm set to c4
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c7, c4, 0    ; Read PA Register
MCR p15, 0, <Rd>, c7, c4, 0    ; Write PA Register
```

VA to PA translation in the current world

The purpose of the VA to PA translation in the current world is to translate the address with the current virtual mapping for either Secure or Non-secure worlds.

The VA to PA translation in the current world operations use:

- CP15 c7
- four, 32-bit write-only operations common to the Secure and Non-secure worlds
- operations accessible in privileged modes only

The operations work for privileged or User access permissions and returns information in the PA Register for aborts, when the translation is unsuccessful, or page table information, when the translation succeeds.

Attempts to access the VA to PA translation operations in the current world in User mode result in an Undefined exception.

To use the VA to PA translation in the current world write CP15 c7 with:

- Opcode_1 set to 0
- CRn set to c7
- CRm set to c8
- Opcode_2 set to:
  - 0 for privileged read permission
  - 1 for privileged write permission
  - 2 for User read permission
  - 3 for User write permission.

General register <Rn> contains the VA for translation. The result returns in the PA Register, for example:

```
MCR p15,0,<Rn>,c7,c8,3        ;get VA = <Rn> and run VA-to-PA translation
                              ;with User write permission.
                              ;if the selected page table has the
                              ;User write permission, the PA is loaded
                              ;in PA register, otherwise abort information is
                              ;loaded in PA Register
MRC p15,0,<Rd>,c7,c4,0        ;read in <Rd> the PA value
```

Note

The VA that this operation uses is the true VA not the MVA.

VA to PA translation in the other world

The purpose of the VA to PA translation in the other world is to translate the address with the current virtual mapping in the Non-secure world while the core is in the Secure world.

The VA to PA translation in the other world operations use:

- CP15 c7
- four, 32-bit write-only operations in the Secure world only
- operations accessible in privileged modes only.

The operations work in the Secure world for Non-secure privileged or Non-secure User access permissions and returns information in the PA Register for aborts, when the translation is unsuccessful, or page table information, when the translation succeeds.

Attempts to access the VA to PA translation operations in the other world in any Non-secure or User mode result in an Undefined exception.

To use the VA to PA translation in the other world write CP15 c7 with:

- Opcode_1 set to 0
- CRn set to c7
- CRm set to c8
- Opcode_2 set to:
  - 4 for privileged read permission
  - 5 for privileged write permission
  - 6 for User read permission
  - 7 for User write permission.

General register <Rn> contains the VA for translation. The result returns in the PA Register, for example:

```
MCR p15,0,<Rn>,c7,c8,4          ;get VA = <Rn> and run Non-secure translation
                                ;with Non-secure privileged read permission.
                                ;if the selected page table has the
                                ;privileged read permission, the PA is loaded
                                ;in PA register, otherwise abort information is
                                ;loaded in PA Register
MRC p15,0,<Rd>,c7,c4,0          ;read in <Rd> the PA value
```

Data Synchronization Barrier operation

The purpose of the Data Synchronization Barrier operation is to ensure that all outstanding explicit memory transactions complete before any following instructions begin. This ensures that data in memory is up to date before the processor executes any more instructions.

Note

The Data Synchronization Barrier operation is synonymous with Drain Write Buffer and Data Write Barrier in earlier versions of the architecture.

The Data Synchronization Barrier operation is:

- in CP15 c7
- 32-bit write-only access, common to both Secure and Non-secure worlds

- accessible in both User and Privileged modes.

Table 3-79 lists the results of attempted access for each mode.

Table 3-79 Results of access to the Data Synchronization Barrier operation

Read	Write
Undefined exception	Data

To use the Data Memory Barrier operation write CP15 with <Rd> SBZ and:

- Opcode_1 set to 0
- CRn set to c7
- CRm set to c10
- Opcode_2 set to 4.

For example:

```
MCR p15,0,<Rd>,c7,c10,4    ; Data Synchronization Barrier operation.
```

For more details, see *Explicit Memory Barriers* on page 6-25.

Note

The W bit that usually enables the Write Buffer is not implemented in ARM1176JZF-S processors, see *c1, Control Register* on page 3-44.

This instruction acts as an explicit memory barrier. This instruction completes when all explicit memory transactions occurring in program order before this instruction are completed. No instructions occurring in program order after this instruction are executed until this instruction completes. Therefore, no explicit memory transactions occurring in program order after this instruction are started until this instruction completes. See *Explicit Memory Barriers* on page 6-25.

It can be used instead of Strongly Ordered memory when the timing of specific stores to the memory system has to be controlled. For example, when a store to an interrupt acknowledge location must be completed before interrupts are enabled.

The Data Synchronization Barrier operation can be performed in both privileged and User modes of operation.

Data Memory Barrier operation

The purpose of the Data Memory Barrier operation is to ensure that all outstanding explicit memory transactions complete before any following explicit memory transactions begin. This ensures that data in memory is up to date before any memory transaction that depends on it.

The Data Memory Barrier operation is:

- in CP15 c7
- a 32-bit write only operation, common to the Secure and Non-secure worlds
- accessible in User and Privileged mode.

Table 3-80 lists the results of attempted access for each mode.

Table 3-80 Results of access to the Data Memory Barrier operation

Read	Write
Undefined exception	Data

To use the Data Memory Barrier operation write CP15 with <Rd> SBZ and:

- Opcode_1 set to 0
- CRn set to c7
- CRm set to c10
- Opcode_2 set to 5.

For example:

```
MCR p15,0,<Rd>,c7,c10,5    ; Data Memory Barrier Operation.
```

For more details, see *Explicit Memory Barriers* on page 6-25.

Wait For Interrupt operation

The purpose of the Wait For Interrupt operation is to put the processor in to a low power state, see *Standby mode* on page 10-3.

The Wait For Interrupt operation is:

- in CP15 c7
- 32-bit write only access, common to Secure and Non-secure worlds
- accessible in privileged modes only.

Table 3-81 lists the results of attempted access for each mode.

Table 3-81 Results of access to the Wait For Interrupt operation

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Undefined exception	Wait For Interrupt	Undefined exception	Wait For Interrupt	Undefined exception

To use the Wait For Interrupt operation write CP15 with <Rd> SBZ and:

- Opcode_1 set to 0
- CRn set to c7
- CRm set to c0
- Opcode_2 set to 4.

For example:

```
MCR p15,0,<Rd>,c7,c0,4    ; Wait For Interrupt.
```

This puts the processor into a low-power state and stops it executing following instructions until an interrupt, an imprecise external abort, or a debug request occurs, regardless of whether the interrupts or external imprecise aborts are disabled by the masks in the CPSR. When an interrupt does occur, the MCR instruction completes. If interrupts are enabled, the IRQ or FIQ handler is entered as normal. The return link in R14_irq or R14_fiq contains the address of the MCR instruction plus 8, so that the normal instruction used for interrupt return (`SUBS PC,R14,#4`) returns to the instruction following the MCR.

3.2.23 c8, TLB Operations Register

The purpose of the TLB Operations Register is to either:

- invalidate all the unlocked entries in the TLB
- invalidate all TLB entries for an area of memory before the MMU remaps it
- invalidate all TLB entries that match an ASID value.

These operations can be performed on either:

- Instruction TLB
- Data TLB
- Unified TLB.

Note

The ARM1176JZF-S processor has a unified TLB. Any TLB operations specified for the Instruction or Data TLB perform the equivalent operation on the unified TLB.

The TLB Operations Register is:

- in CP15 c8
- a 32-bit write-only register banked for Secure and Non-secure world operations
- accessible in privileged modes only.

Table 3-82 lists the results of attempted access for each mode.

Table 3-82 Results of access to the TLB Operations Register

Secure Privileged		Non-secure Privileged		User
Read	**Write**	**Read**	**Write**	
Undefined exception	Secure data	Undefined exception	Non-secure data	Undefined exception

To access the TLB Operations Register write CP15 with:

- Opcode_1 set to 0
- CRn set to c8
- CRm set to:
  - c5, Instruction TLB
  - c6, Data TLB
  - c7, Unified TLB
- Opcode_2 set to:
  - 0, Invalidate TLB unlocked entries
  - 1, Invalidate TLB Entry by MVA
  - 2, Invalidate TLB Entry on ASID Match.

For example, to invalidate all the unlocked entries in the Instruction TLB:

```
MCR p15,0,<Rd>,c8, c5,0    ; Write TLB Operations Register
```

Functions that update the contents of the TLB occur in program order. Therefore, an explicit data access before the TLB function uses the old TLB contents, and an explicit data access after the TLB function uses the new TLB contents. For instruction accesses, TLB updates are guaranteed to have taken effect before the next pipeline flush. This includes Flush Prefetch Buffer operations and exception return sequences.

Invalidate TLB unlocked entries

Invalidate TLB unlocked entries invalidates all the unlocked entries in the TLB. This function causes a flush of the prefetch buffer. Therefore, all instructions that follow are fetched after the TLB invalidation.

Invalidate TLB Entry by MVA

You can use Invalidate TLB Entry by MVA to invalidate all TLB entries for an area of memory before you remap.

You must perform an Invalidate TLB Entry by MVA of an MVA in each area you want to remap, section, small page, or large page.

This function invalidates a TLB entry that matches the provided MVA and ASID, or a global TLB entry that matches the provided MVA.

This function invalidates a matching locked entry.

The Invalidate TLB Entry by MVA operation uses an MVA and ASID as an argument. Figure 3-47 shows the format of this.

Figure 3-47 TLB Operations Register MVA and ASID format

Invalidate TLB Entry on ASID Match

This is a single interruptible operation that invalidates all TLB entries that match the provided ASID value.

This function invalidates locked entries but does not invalidate entries marked as global.

In this processor this operation takes several cycles to complete and the instruction is interruptible. When interrupted the R14 state is set to indicate that the MCR instruction has not executed. Therefore, R14 points to the address of the MCR + 4. The interrupt routine then automatically restarts at the MCR instruction. If the processor interrupts and later restarts this operation, any entries fetched into the TLB by the interrupt that uses the provided ASID are invalidated by the restarted invalidation.

The Invalidate TLB Entry on ASID Match function requires an ASID as an argument. Figure 3-48 shows the format of this.

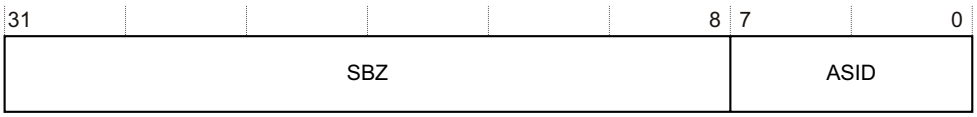

Figure 3-48 TLB Operations Register ASID format

3.2.24 c9, Data and instruction cache lockdown registers

The purpose of the data and instruction cache lockdown registers is to provide a means to lock down the caches and therefore provide some control over pollution that applications might cause. With these registers you can lock down each cache way independently.

There are two cache lockdown registers:

- one Data Cache Lockdown Register
- one Instruction Cache Lockdown Register.

The cache lockdown registers are:

- in CP15 c9
- two 32-bit read/write registers, common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-49 shows the bit arrangement of the cache lockdown registers.

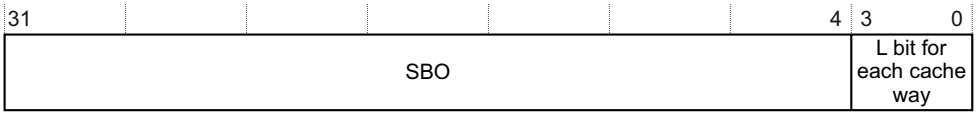

Figure 3-49 Instruction and data cache lockdown register formats

Table 3-83 lists how the bit values correspond with the cache lockdown registers functions.

Table 3-83 Instruction and data cache lockdown register bit functions

Bits	Field name	Function
[31:4]	SBO	UNP on reads, SBO on writes.
[3:0]	L bit for each cache way	Locks each cache way individually. The L bits for cache ways 3 to 0 are bits [3:0] respectively. On a line fill to the cache, data is allocated to unlocked cache ways as determined by the standard replacement algorithm. Data is not allocated to locked cache ways. If a cache way is not implemented, then the L bit for that way is hardwired to 1, and writes to that bit are ignored. 0 indicates that this cache way is not locked. Allocation to this cache way is determined by the standard replacement algorithm. This is the reset state. 1 indicates that this cache way is locked. No allocation is performed to this cache way.

The lockdown behavior depends on the CL bit, see *c1, Non-Secure Access Control Register* on page 3-55. If the CL bit is not set, the Lockdown entries are reserved for the Secure world. Table 3-84 lists the results of attempted access for each mode.

Table 3-84 Results of access to the Instruction and Data Cache Lockdown Register

CL bit value	Secure Privileged		Non-secure Privileged		User
	Read	Write	Read	Write	
0	Data	Data	Undefined exception	Undefined exception	Undefined exception
1	Data	Data	Data	Data	Undefined exception

The Data Cache Lockdown Register only supports the Format C method of lockdown. This method is a cache way based scheme that gives a traditional lockdown function to lock critical regions in the cache.

A locking bit for each cache way determines if the normal cache allocation mechanisms, Random or Round-Robin, can access that cache way. For details of the RR bit, that controls the selection of Random or Round-Robin cache policy, see *c1, Control Register* on page 3-44.

ARM1176JZF-S processors have an associativity of 4. With all ways locked, the ARM1176JZF-S processor behaves as if only ways 3 to 1 are locked and way 0 is unlocked.

To use the Instruction and Data Cache Lockdown Registers read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c9
- CRm set to c0
- Opcode_2 set to:
  - 0, for Data Cache
  - 1, for Instruction Cache.

For example:

```
MRC p15, 0, <Rd>, c9, c0, 0    ; Read Data Cache Lockdown Register
MCR p15, 0, <Rd>, c9, c0, 0    ; Write Data Cache Lockdown Register
MRC p15, 0, <Rd>, c9, c0, 1    ; Read Instruction Cache Lockdown Register
MCR p15, 0, <Rd>, c9, c0, 1    ; Write Instruction Cache Lockdown Register
```

The system must only change a cache lockdown register when it is certain that all outstanding accesses that might cause a cache line fill are complete. For this reason, the processor must perform a Data Synchronization Barrier operation before the cache lockdown register changes, see *Data Synchronization Barrier operation* on page 3-83.

The following procedure for lock down into a data or instruction cache way i, with N cache ways, using Format C, ensures that only the target cache way i is locked down.

This is the architecturally defined method for locking data or instructions into caches:

1. Ensure that no processor exceptions can occur during the execution of this procedure, by disabling interrupts. If this is not possible, all code and data or instructions used by any exception handlers that can be called must meet the conditions specified in step 2.
2. Ensure that all data or instructions used by the following code, apart from the data or instructions that are to be locked down, are either:
   - in an noncacheable area of memory, including the TCM
   - in an already locked cache way.
3. Ensure that the data or instructions to be locked down are in a Cacheable area of memory.
4. Ensure that the data or instructions to be locked down are not already in the cache, using cache Clean and/or Invalidate instructions as appropriate, see *c7, Cache operations* on page 3-69.
5. Enable allocation to the target cache way by writing to the Instruction or Data Cache Lockdown Register, with the CRm field set to 0, setting L equal to 0 for bit i and L equal to 1 for all other ways.
6. Ensure that the memory cache line is loaded into the cache by using an LDR instruction to load a word from the memory cache line, for each of the cache lines to be locked down in cache way i.

   To lock down an instruction cache use the c7 Prefetch Instruction Cache Line operation to fetch the memory cache line, see *Invalidate, Clean, and Prefetch operations* on page 3-71.
7. Write to the Instruction or Data Cache Lockdown Register, setting L to 1 for bit i and restore all the other bits to the values they had before this routine was started.

3.2.25 c9, Data TCM Region Register

The purpose of the Data TCM Region Register is to describe the physical base address and size of the Data TCM region and to provide a mechanism to enable it.

The Data TCM Region Register is:

- in CP15 c9
- a 32-bit read/write register common to Secure and Non-secure worlds
- accessible in privileged modes only.

If the processor is configured to have 2 Data TCMs, each TCM has a separate Data TCM Region Register. The TCM Selection Register determines the register in use.

Figure 3-50 shows the bit arrangement for the Data TCM Region Register.

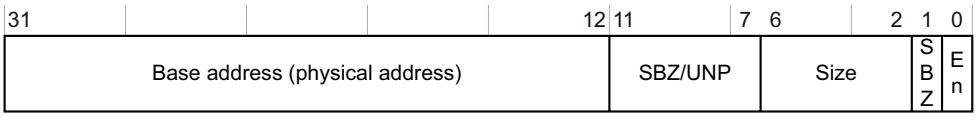

Figure 3-50 Data TCM Region Register format

Table 3-85 lists how the bit values correspond with the Data TCM Region Register functions.

Table 3-85 Data TCM Region Register bit functions

Bits	Field name	Function
[31:12]	Base address	Contains the physical base address of the TCM. The base address must be aligned to the size of the TCM. Any bits in the range [($\log_2$(RAMSize)-1):12] are ignored. The base address is 0 at Reset.
[11:7]	-	UNP/SBZ.
[6:2]	Size	Indicates the size of the TCM on reads[a]. All other values are reserved: b00000 = 0KB b00011 = 4KB b00100 = 8KB b00101 = 16KB b00110 = 32KB.
[1]	-	UNP/SBZ.
[0]	En	Indicates if the TCM is enabled. 0 = TCM disabled, reset value 1 = TCM enabled.

a. On writes this field is ignored. For more details see *Tightly-coupled memory* on page 7-7.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Note

When the NS access bit is 0 for Data TCM, see *c9, Data TCM Non-secure Control Access Register* on page 3-93, attempts to access the Data TCM Region Register from the Non-secure world cause an Undefined exception.

Table 3-86 lists the results of attempted access for each mode.

Table 3-86 Results of access to the Data TCM Region Register

NS access bit value	Secure Privileged		Non-secure Privileged		User
	Read	Write	Read	Write	
0	Data	Data	Undefined exception	Undefined exception	Undefined exception
1	Data	Data	Data	Data	Undefined exception

To use the Data TCM Region Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c9
- CRm set to c1
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c9, c1, 0    ; Read Data TCM Region Register
MCR p15, 0, <Rd>, c9, c1, 0    ; Write Data TCM Region Register
```

Attempting to change the Data TCM Region Register while a DMA operation is running has Unpredictable effects but there is no impact on security.

3.2.26 c9, Instruction TCM Region Register

The purpose of the Instruction TCM Region Register is to describe the physical base address and size of the Instruction TCM region and to provide a mechanism to enable it.

Table 3-87 on page 3-92 lists the purposes of the individuals bits of the Instruction TCM Region Register.

The Instruction TCM Region Register is:

- in CP15 c9
- a 32-bit read/write register common to Secure and Non-secure worlds
- accessible in privileged modes only.

If the processor is configured to have 2 Instruction TCMs, each TCM has a separate Instruction TCM Region Register. The TCM Selection Register determines the register in use.

Figure 3-51 shows the bit arrangement for the Instruction TCM Region Register.

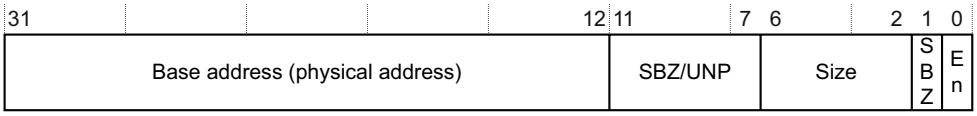

Figure 3-51 Instruction TCM Region Register format

Table 3-87 lists how the bit values correspond with the Instruction TCM Region Register functions.

Table 3-87 Instruction TCM Region Register bit functions

Bits	Field name	Function
[31:12]	Base address	Contains the physical base address of the TCM. The base address must be aligned to the size of the TCM. Any bits in the range [($\log_2$(RAMSize)-1):12] are ignored. The base address is 0 at Reset.
[11:7]	-	UNP/SBZ.
[6:2]	Size	Indicates the size of the TCM on reads[a]. All other values are reserved: b00000 = 0KB b00011 = 4KB b00100 = 8KB b00101 = 16KB b00110 = 32KB.
[1]	-	UNP/SBZ.
[0]	En	Indicates if the TCM is enabled: 0 = TCM disabled. 1 = TCM enabled. The reset value of this bit depends on the value of the **INITRAM** static configuration signal. If INITRAM is HIGH then this bit resets to 1. If INITRAM is LOW then this bit resets to 0. For more information see *Static configuration signals* on page A-4.

a. On writes this field is ignored. For more details see *Tightly-coupled memory* on page 7-7.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

The value of the En bit at Reset depends on the **INITRAM** signal:

- **INITRAM** LOW sets En to 0
- **INITRAM** HIGH sets En to 1.

When **INITRAM** is HIGH this enables the Instruction TCM directly from reset, with a Base address of `0x00000`. When the processor comes out of reset, it executes the instructions in the Instruction TCM instead of fetching instructions from external memory, except when the processor uses high vectors.

Note

When the NS access bit is 0 for Instruction TCM, see *c9, Instruction TCM Non-secure Control Access Register* on page 3-94, attempts to access the Instruction TCM Region Register from the Non-secure world cause an Undefined exception.

Table 3-88 lists the results of attempted access for each mode.

Table 3-88 Results of access to the Instruction TCM Region Register

NS access bit value	Secure Privileged		Non-secure Privileged		User
	Read	Write	Read	Write	
0	Data	Data	Undefined exception	Undefined exception	Undefined exception
1	Data	Data	Data	Data	Undefined exception

To use the Instruction TCM Region Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c9
- CRm set to c1
- Opcode_2 set to 1.

For example:

```
MRC p15, 0, <Rd>, c9, c1, 1    ; Read Instruction TCM Region Register
MCR p15, 0, <Rd>, c9, c1, 1    ; Write Instruction TCM Region Register
```

Attempts to change the Instruction TCM Region Register while a DMA operation is running has Unpredictable effects but there is no impact on security.

3.2.27 c9, Data TCM Non-secure Control Access Register

The purpose of the Data TCM Non-secure Access Register is to:

- set access permission to the Data TCM Region Register
- define data in the Data TCM as Secure or Non-secure.

The Data TCM Non-secure Control Access Register is:

- in CP15 c9
- a 32-bit read/write register in the Secure world only
- accessible in privileged modes only.

If the processor is configured to have 2 Data TCMs, each TCM has a separate Data TCM Non-secure Control Access Register. The TCM Selection Register determines the register in use.

Figure 3-52 shows the bit arrangement for the Data TCM Non-secure Control Access Register.

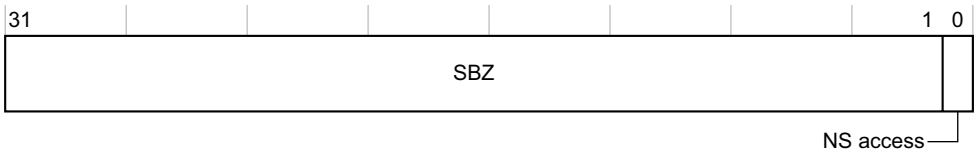

Figure 3-52 Data TCM Non-secure Control Access Register format

Table 3-89 lists how the bit values correspond with the register functions.

Table 3-89 Data TCM Non-secure Control Access Register bit functions

Bits	Field name	Function
[31:1]	-	UNP/SBZ.
[0]	NS access	Makes Data TCM invisible to the Non-secure world and makes TCM data Secure. 0 = Data TCM Region Register only accessible in the Secure world. Data TCM only visible in the Secure world and only when the NS Attribute in the page table is 0. The reset value is 0. 1 = Data TCM Region Register accessible in the Secure and Non-secure worlds. Data TCM is visible in the Non-secure world, and also in the Secure world if the NS Attribute in the page table is 1.

Table 3-90 lists the effect on TCM operations for different combinations of operating world and NS bits.

Table 3-90 Effects of NS items for data TCM operation

World	NS access	NS page table	Region visible	Control	Data
Secure	0	1	No	-	-
	1	0	No	-	-
	0	0	Yes	Secure privileged only	Secure only
	1	1	Yes	Secure and Non-secure privileged	Non-secure only
Non-secure	1	X	Yes	Secure and Non-secure privileged	Non-secure only
	0	X	No	-	-

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Attempts to access the Data TCM Non-secure Control Access Register in modes other than Secure privileged result in an Undefined exception.

To use the Data TCM Non-secure Control Access Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c9
- CRm set to c1
- Opcode_2 set to 2.

For example:

```
MRC p15,0,<Rd>,c9,c1,2    ; Read Data TCM Non-secure Control Access Register
MCR p15,0,<Rd>,c9,c1,2    ; Write Data TCM Non-secure Control Access Register
```

3.2.28 c9, Instruction TCM Non-secure Control Access Register

The purpose of the Instruction TCM Non-secure Control Access Register is to:

- set access permission to the Instruction TCM Region Register
- define instructions in the Instruction TCM as Secure or Non-secure.

The Instruction TCM Non-secure Control Access Register is:

- in CP15 c9
- a 32-bit read/write register in the Secure world only
- accessible in privileged modes only.

If the processor is configured to have 2 Instruction TCMs, each TCM has a separate Instruction TCM Non-secure Control Access Register. The TCM Selection Register determines the register in use.

Figure 3-53 shows the bit arrangement for the Instruction TCM Non-secure Control Access Register.

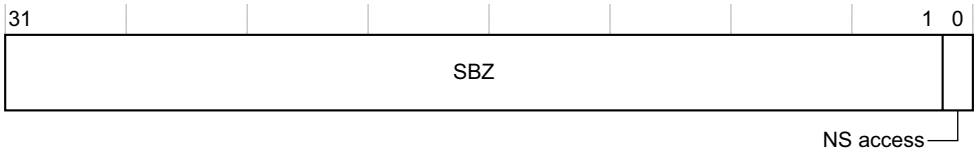

Figure 3-53 Instruction TCM Non-secure Control Access Register format

Table 3-91 lists how the bit values correspond with the register functions.

Table 3-91 Instruction TCM Non-secure Control Access Register bit functions

Bits	Field name	Function
[31:1]	-	UNP/SBZ.
[0]	NS access	Makes Instruction TCM invisible to the Non-secure world and makes TCM data Secure. 0 = Instruction TCM Region Register only accessible in the Secure world. Instruction TCM only visible in the Secure world and only when the NS Attribute in the page table is 0. The reset value is 0. 1 = Instruction TCM Region Register accessible in the Secure and Non-secure worlds. Instruction TCM is visible in the Non-secure world, and also in the Secure world if the NS Attribute in the page table is 1.

Table 3-92 lists the effect on TCM operations for different combinations of operating world, and NS bits.

Table 3-92 Effects of NS items for instruction TCM operation

World	NS access	NS page table	Region visible	Control	Data
Secure	0	1	No	-	-
	1	0	No	-	-
	0	0	Yes	Secure privileged only	Secure only
	1	1	Yes	Secure and Non-secure privileged	Non-secure only
Non-secure	1	X	Yes	Secure and Non-secure privileged	Non-secure only
	0	X	No	-	-

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Attempts to access the Instruction TCM Non-secure Control Access Register in modes other than Secure Privileged result in an Undefined exception.

To use the Instruction TCM Non-secure Control Access Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c9
- CRm set to c1
- Opcode_2 set to 3.

For example:

```
MRC p15,0,<Rd>,c9,c1,3    ;Read Instruction TCM Non-secure Control Access Register
MCR p15,0,<Rd>,c9,c1,3    ;Write Instruction TCM Non-secure Control Access Register
```

3.2.29 c9, TCM Selection Register

The purpose of the TCM Selection Register is to determine the bank of CP15 registers related to TCM configuration in use. These banks consist of:

- *c9, Data TCM Region Register* on page 3-89
- *c9, Instruction TCM Region Register* on page 3-91
- *c9, Data TCM Non-secure Control Access Register* on page 3-93
- *c9, Instruction TCM Non-secure Control Access Register* on page 3-94.

The TCM Selection Register is:

- in CP15 c9
- a 32-bit read/write register banked in the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-54 shows the bit arrangement for the TCM Selection Register.

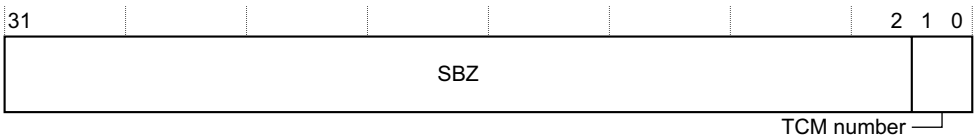

Figure 3-54 TCM Selection Register format

Table 3-93 lists how the bit values correspond with the TCM Selection Register functions.

Table 3-93 TCM Selection Register bit functions

Bits	Field name	Function
[31:2]	-	UNP/SBZ.
[1:0]	TCM number	Selects the bank of CP15 registers related to TCM configuration. Attempts to select a bank related to a TCM that does not exist are ignored: b00 = TCM 0, reset value. b01 = TCM 1. When there is only one TCM on both Instruction and Data sides, write access is ignored. b10 = Write access ignored. b11 = Write access ignored.

Accesses to the TCM Region Registers and TCM Non-secure Control Access Registers in the Secure world, access the bank of CP15 registers related to TCM configuration selected by the Secure TCM Selection Register. Accesses to the TCM Region Registers in the Non-secure world, access the bank of CP15 registers related to TCM configuration selected by the Non-secure TCM Selection Register.

Table 3-94 lists the results of attempted access for each mode.

Table 3-94 Results of access to the TCM Selection Register

Secure Privileged		Non-secure Privileged		User
Read	**Write**	**Read**	**Write**	
Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception

To use the TCM Selection Register read or write CP15 c9 with:

- Opcode_1 set to 0
- CRn set to c9
- CRm set to c2
- Opcode_2 set to 0.

For example:

```
MRC p15,0,<Rd>,c9,c2,0    ; Read TCM Selection register
MCR p15,0,<Rd>,c9,c2,0    ; Write TCM Selection register
```

3.2.30 c9, Cache Behavior Override Register

The purpose of the Cache Behavior Override Register is to control cache write through and line fill behavior for interruptible cache operations, or during debug. The register enables you to ensure that the contents of caches do not change, for example in debug.

The Cache Behavior Override Register is:

- in CP15 c9
- a 32 bit read/write register, Table 3-95 on page 3-98 lists the access for each bit in Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-55 shows the bit arrangement for the Cache Behavior Override Register.

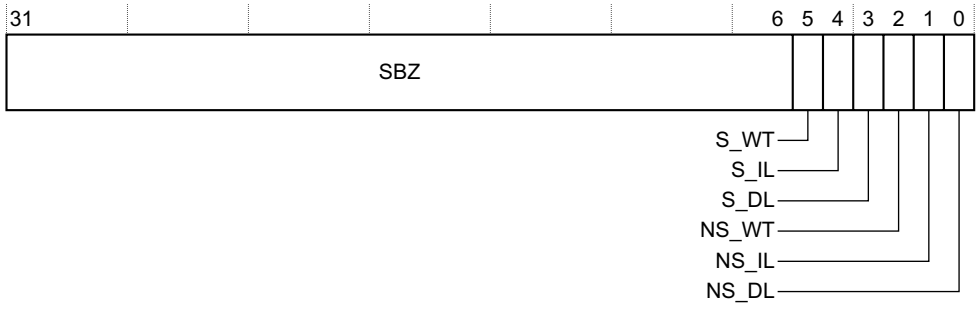

Figure 3-55 Cache Behavior Override Register format

Table 3-95 lists how the bit values correspond to the Cache Behavior Override Register.

Table 3-95 Cache Behavior Override Register bit functions

Bits	Field name	Access	Function
[31:6]	-	-	UNP/SBZ.
[5]	S_WT	Secure only	Defines write-through behavior for regions marked as Secure write-back: 0 = Do not force write-through, normal operation, reset value 1 = Force write-through.
[4]	S_IL	Secure only	Defines Instruction Cache linefill behavior for Secure regions: 0 = Instruction Cache linefill enabled, normal operation, reset value 1 = Instruction Cache linefill disabled.
[3]	S_DL	Secure only	Defines Data Cache linefill behavior for Secure regions: 0 = Data Cache linefill enabled, normal operation, reset value 1 = Data Cache linefill disabled.
[2]	NS_WT	Common	Defines write-through behavior for regions marked as Non-secure write-back: 0 = Do not force write-through, normal operation, reset value 1 = Force write-through.
[1]	NS_IL	Common	Defines Instruction Cache linefill behavior for Non-secure regions: 0 = Instruction Cache linefill enabled, normal operation, reset value 1 = Instruction Cache linefill disabled.
[0]	NS_DL	Common	Defines Data Cache linefill behavior for Non-secure regions: 0 = Data Cache linefill enabled, normal operation, reset value 1 = Data Cache linefill disabled.

Table 3-96 lists the actions that result from attempted access for each mode.

Table 3-96 Results of access to the Cache Behavior Override Register

Bits	Secure Privileged access	Non-secure Privileged access		User access
		Read	Write	
Secure only [5:3]	Data	Read As Zero	Ignored	Undefined exception
Common [2:0]	Data	Data	Data	Undefined exception

To use the Cache Behavior Override Register read or write CP15 with:

- Opcode_1 to 0
- CRn set to c9
- CRm set to c8
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c9, c8, 0    ; Read Cache Behavior Override Register
MCR p15, 0, <Rd>, c9, c8, 0    ; Write Cache Behavior Override Register
```

You might use the Cache Behavior Override Register during, for example, clean or clean and invalidate all operations in Non-secure world that might not prevent fast interrupts to the Secure world if the FW bit is clear, see *c1, Secure Configuration Register* on page 3-52. In this case, the Secure world can read or write the Non-secure locations in the cache, so potentially causing the

cache to contain valid or dirty Non-secure entries when the Non-secure clean or clean and invalidate all operation completes. To avoid this kind of problem, the Secure side must not allocate Non-secure entries into the cache and must treat all writes to Non-secure regions that hit in the cache as write-though.

Note

Three bits, nWT, nIL and nDL, are also defined for Debug state in CP14, see *CP14 c10, Debug State Cache Control Register* on page 13-23, and apply to all Secure and Non-secure regions. The CP14 register has precedence over the CP15 register when the core is in Debug state, and the CP15 register has precedence over the CP14 register in functional states.

For more information on cache debug, see Chapter 13 *Debug*.

3.2.31 c10, TLB Lockdown Register

The purpose of the TLB Lockdown Register is to control where hardware page table walks place the TLB entry in either:

- the set associative region of the TLB
- the lockdown region of the TLB, and if in the lockdown region, the entry to write.

Table 3-97 lists the purposes of the individual bits in the TLB Lockdown Register.

The TLB Lockdown Register is:

- in CP15 c10
- 32-bit read/write register common to Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-56 shows the bit arrangement of the TLB Lockdown Register.

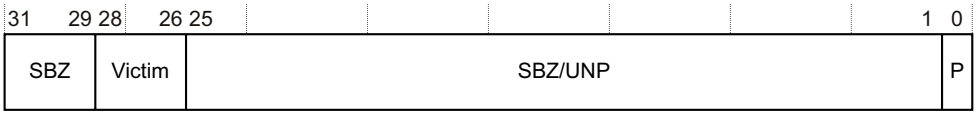

Figure 3-56 TLB Lockdown Register format

Table 3-97 lists how the bit values correspond with the TLB Lockdown Register functions.

Table 3-97 TLB Lockdown Register bit functions

Bits	Field name	Function
[31:29]	-	UNP/SBZ.
[28:26]	Victim	Specifies the entry in the lockdown region where a subsequent hardware page table walk can place a TLB entry. The reset value is 0. 0-7, defines the Lockdown region for the TLB entry.
[25:1]	-	UNP/SBZ.
[0]	P	Determines if subsequent hardware page table walks place a TLB entry in the lockdown region or in the set associative region of the TLB: 0 = Place the TLB entry in the set associative region of the TLB, reset value. 1 = Place the TLB entry in the lockdown region of the TLB as defined by the Victim bits [28:26].

The TLB lockdown behavior depends on the TL bit, see *c1, Non-Secure Access Control Register* on page 3-55. If the TL bit is not set, the Lockdown entries are reserved for the Secure world. Table 3-98 lists the results of attempted access for each mode.

Table 3-98 Results of access to the TLB Lockdown Register

TL bit value	Secure Privileged		Non-secure Privileged		User
	Read	Write	Read	Write	
0	Data	Data	Undefined exception	Undefined exception	Undefined exception
1	Data	Data	Data	Data	Undefined exception

The lockdown region of the TLB contains eight entries. *TLB organization* on page 6-4 describes the structure of the TLB.

The Invalidate TLB unlocked entries operation does not invalidate TLB entries in the lockdown region.

Invalidate TLB Entry by MVA and Invalidate TLB Entry on ASID Match operations invalidate any TLB entries that correspond to the MVA or ASID given in Rd, if they are in the lockdown region or if they are in the set-associative region of the TLB. See *c8, TLB Operations Register* on page 3-86 for a description of the TLB invalidate operations.

The victim automatically increments after any page table walk that results in a write puts an entry into the lockdown part of the TLB.

To use the TLB Lockdown Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c10
- CRm set to c0
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c10, c0, 0    ; Read TLB Lockdown Register
MCR p15, 0, <Rd>, c10, c0, 0    ; Write TLB Lockdown Register.
```

Example 3-2 is a code sequence that locks down an entry to the current victim.

Example 3-2 Lock down an entry to the current victim

```
ADR r1,LockAddr          ; set r1 to the value of the address to be locked down
MCR p15,0,r1,c8,c7,1     ; invalidate TLB single entry to ensure that
                         ; LockAddr is not already in the TLB
MRC p15,0,R0,c10,c0,0    ; read the lockdown register
ORR R0,R0,#1             ; set the preserve bit
MCR p15,0,R0,c10,c0,0    ; write to the lockdown register
LDR r1,[r1]              ; TLB misses, and entry is loaded
MRC p15,0,R0,c10,c0,0    ; read the lockdown register (victim
                         ; increments)
BIC R0,R0,#1             ; clear preserve bit
MCR p15,0,R0,c10,c0,0    ; write to the lockdown register
```

3.2.32 c10, Memory region remap registers

The purpose of the memory region remap registers is to remap memory region attributes encoded by the TEX[2:0], C, and B bits in the page tables that the Data side, Instruction side, and DMA use. For details of memory remap, see *Memory region attributes* on page 6-14.

The memory region remap registers are:

- in CP15 c10
- two 32-bit read/write registers banked for the Secure and Non-secure worlds:
  - — the Primary Region Remap Register
  - — the Normal Memory Remap Register.
- accessible in privileged modes only.

These registers apply to all memory accesses and this includes accesses from the Data side, Instruction side, and DMA. Table 3-99 on page 3-102 lists the purposes of the individual bits in the Primary Region Remap Register. Table 3-101 on page 3-103 lists the purposes of the individual bits in the Normal Memory Remap Register.

Note

The behavior of the memory region remap registers depends on the TEX remap bit, see *c1, Control Register* on page 3-44.

Figure 3-57 shows the arrangement of the bits in the Primary Region Remap Register.

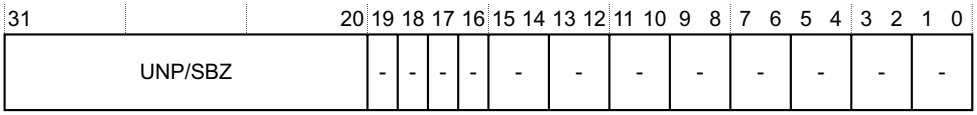

Figure 3-57 Primary Region Remap Register format

Table 3-99 lists the functional bits of the Primary Region Remap Register.

Table 3-99 Primary Region Remap Register bit functions

Bits	Field name	Function[a]
[31:20]	-	UNP/SBZ
[19]	-	Remaps shareable attribute when S=1 for Normal regions[b] 1 = reset value
[18]	-	Remaps shareable attribute when S=0 for Normal regions[b] 0 = reset value
[17]	-	Remaps shareable attribute when S=1 for Device regions[b] 0 = reset value
[16]	-	Remaps shareable attribute when S= 0 for Device regions[b] 1= reset value
[15:14]	-	Remaps {TEX[0],C,B} = b111 b10 = reset value
[13:12]	-	Remaps {TEX[0],C,B} = b110 b00 = reset value
[11:10]	-	Remaps {TEX[0],C,B} = b101 b10 = reset value
[9:8]	-	Remaps {TEX[0],C,B} = b100 b10 = reset value
[7:6]	-	Remaps {TEX[0],C,B} = b011 b10 = reset value
[5:4]	-	Remaps {TEX[0],C,B} = b010 b10 = reset value
[3:2]	-	Remaps {TEX[0],C,B} = b001 b01 = reset value
[1:0]	-	Remaps {TEX[0],C,B} = b000 b00 = reset value

a. The reset values ensure that no remapping occurs at reset

b. Shareable attributes can map for both shared and non-shared memory. If the Shared bit read from the TLB or page tables is 0, then the bit remaps to the Not Shared attributes in this register. If the Shared bit read from the TLB or page tables is 1, then the bit remaps to the Shared attributes of this register.

Table 3-100 lists the encoding of the remapping for the primary memory type.

Table 3-100 Encoding for the remapping of the primary memory type

Encoding	Memory type
b00	Strongly ordered
b01	Device
b10	Normal
b11	UNP, normal

Figure 3-58 shows the arrangement of the bits in the Normal Memory Remap Register.

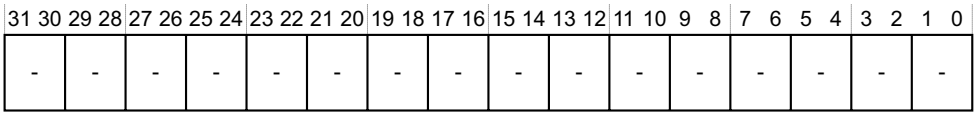

Figure 3-58 Normal Memory Remap Register format

Table 3-101 lists how the bit values correspond with the Normal Memory Remap Register functions.

Table 3-101 Normal Memory Remap Register bit functions

Bits	Field name	Function[a]
[31:30]	-	Remaps Outer attribute for {TEX[0],C,B} = b111 b01 = reset value
[29:28]	-	Remaps Outer attribute for {TEX[0],C,B} = b110 b00 = reset value
[27:26]	-	Remaps Outer attribute for {TEX[0],C,B} = b101 b01 = reset value
[25:24]	-	Remaps Outer attribute for {TEX[0],C,B} = b100 b00 = reset value
[23:22]	-	Remaps Outer attribute for {TEX[0],C,B} = b011 b11 = reset value
[21:20]	-	Remaps Outer attribute for {TEX[0],C,B} = b010 b10 = reset value
[19:18]	-	Remaps Outer attribute for {TEX[0],C,B} = b001 b00 = reset value
[17:16]	-	Remaps Outer attribute for {TEX[0],C,B} = b000 b00 = reset value
[15:14]	-	Remaps Inner attribute for {TEX[0],C,B} = b111 b01 = reset value

Table 3-101 Normal Memory Remap Register bit functions (continued)

Bits	Field name	Function[a]
[13:12]	-	Remaps Inner attribute for {TEX[0],C,B} = b110 b00 = reset value
[11:10]	-	Remaps Inner attribute for {TEX[0],C,B} = b101 b10 = reset value
[9:8]	-	Remaps Inner attribute for {TEX[0],C,B} = b100 b00 = reset value
[7:6]	-	Remaps Inner attribute for {TEX[0],C,B} = b011 b11 = reset value
[5:4]	-	Remaps Inner attribute for {TEX[0],C,B} = b010 b10 = reset value
[3:2]	-	Remaps Inner attribute for {TEX[0],C,B} = b001 b00 = reset value
[1:0]	-	Remaps Inner attribute for {TEX[0],C,B} = b000 b00 = reset value

a. The reset values ensure that no remapping occurs at reset.

Table 3-102 lists the encoding for the Inner or Outer cacheable attribute bit fields I0 to I7 and O0 to O7.

Table 3-102 Remap encoding for Inner or Outer cacheable attributes

Encoding	Cacheable attribute
b00	Noncacheable
b01	Write-back, allocate on write
b10	Write-through, no allocate on write
b11	Write-back, no allocate on write

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Table 3-103 lists the results of attempted access for each mode.

Table 3-103 Results of access to the memory region remap registers

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception

To use the memory region remap registers read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c10
- CRm set to c2

- Opcode_2 set to:
  - 0, Primary Region Remap Register
  - 1, Normal Memory Remap Register.

For example:

```
MRC p15, 0, <Rd>, c10, c2, 0    ;Read Primary Region Remap Register
MCR p15, 0, <Rd>, c10, c2, 0    ;Write Primary Region Remap Register
MRC p15, 0, <Rd>, c10, c2, 1    ;Read Normal Memory Remap Register
MCR p15, 0, <Rd>, c10, c2, 1    ;Write Normal Memory Remap Register
```

Memory remap occurs in two stages:

1. The processor uses the Primary Region Remap Register to remap the primary memory type, Normal, Device, or Strongly Ordered, and the shareable attribute.
2. For memory regions that the Primary Region Remap Register defines as Normal memory, the processor uses the Normal Memory Remap Register to remap the inner and outer cacheable attributes.

The behavior of the memory region remap registers depends on the TEX remap bit, see *c1, Control Register* on page 3-44. If the TEX remap bit is set, the entries in the memory region remap registers remap each possible value of the TEX[0], C and B bits in the page tables. You can therefore set your own definitions for these values. If the TEX remap bit is clear, the memory region remap registers are not used and no memory remapping takes place. For more information see *Memory region attributes* on page 6-14.

The memory region remap registers are expected to remain static during normal operation. When you write to the memory region remap registers, you must invalidate the TLB and perform an IMB operation before you can rely on the new written values. You must also stop the DMA if it is running or queued.

Note

You cannot remap the NS bit. This is for security reasons.

3.2.33 c11, DMA identification and status registers

The purpose of the DMA identification and status registers is to define:

- the DMA channels that are physically implemented on the particular device
- the current status of the DMA channels.

Processes that handle DMA can read this register to determine the physical resources implemented and their availability.

The DMA Identification and Status Register is:

- in CP15 c11
- four 32-bit read-only registers common to Secure and Non-secure worlds
- accessible only in privileged modes.

Figure 3-59 shows the format of DMA identification and status registers 0-3.

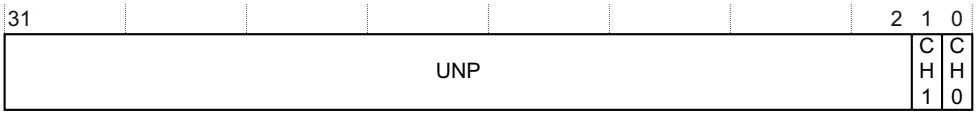

Figure 3-59 DMA identification and status registers format

Table 3-104 lists how the bit values correspond with the DMA identification and status registers.

Table 3-104 DMA identification and status register bit functions

Bits	Field name	Function
[31:2]	-	UNP/SBZ
[1]	CH1	Provides information on DMA Channel 1 functions: 0 = DMA Channel 1 function[a] disabled 1 = DMA Channel 1 function[a] enabled.
[0]	CH0	Provides information on DMA Channel 0 functions: 0 = DMA Channel 0 function[a] disabled 1 = DMA Channel 0 function[a] enabled.

a. See Table 3-105 for the function of the channel that Opcode_2 of the MRC instruction determines.

Table 3-105 lists the Opcode_2 values used to select the DMA channel function.

Table 3-105 DMA Identification and Status Register functions

Opcode_2	Function
0	Indicates channel present: 0 = the channel is not Present 1 = the channel is Present.
1	Indicates channel queued: 0 = the channel is not Queued 1 = the channel is Queued.

Table 3-105 DMA Identification and Status Register functions (continued)

Opcode_2	Function
2	Indicates channel running: 0 = the channel is not Running 1 = the channel is Running.
3	Indicates channel interrupting: 0 = the channel is not Interrupting 1 = the channel is Interrupting, through completion or an error.
4-7	Reserved. Results in an Undefined exception.

Access in the Non-secure world depends on the DMA bit, see *c1, Non-Secure Access Control Register* on page 3-55. The processor can only access these registers in Privileged modes. Table 3-106 lists the results of attempted access for each mode.

Table 3-106 Results of access to the DMA identification and status registers

DMA bit	Secure Privileged		Non-secure Privileged		User	
	Read	Write	Read	Write		
0	Data	Undefined exception	Undefined exception	Undefined exception	Undefined exception	
1	Data	Undefined exception	Data	Undefined exception	Undefined exception	

To access the DMA identification and status registers in a privileged mode read CP15 with:

- Opcode_1 set to 0
- CRn set to c11
- CRm set to c0
- Opcode_2 set to:
  - 0, Present
  - 1, Queued
  - 2, Running
  - 3, Interrupting.

For example:

```
MRC p15, 0, <Rd>, c11, c0, 0    ; Read DMA Identification and Status Register present
MRC p15, 0, <Rd>, c11, c0, 1    ; Read DMA Identification and Status Register queued
MRC p15, 0, <Rd>, c11, c0, 2    ; Read DMA Identification and Status Register running
MRC p15, 0, <Rd>, c11, c0, 3    ; Read DMA Identification and Status Register interrupting.
```

3.2.34 c11, DMA User Accessibility Register

The purpose of the DMA User Accessibility Register is to determine if a User mode process can access the registers for each channel.

The DMA User Accessibility Register is:

- in CP15 c11
- a 32-bit read/write register common to the Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-60 on page 3-108 shows the bit arrangement for the DMA User Accessibility Register.

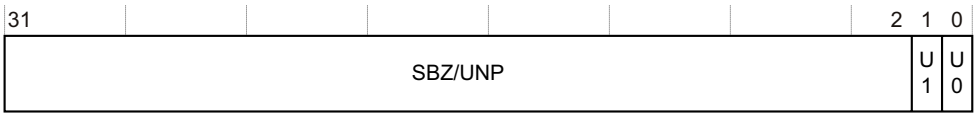

Figure 3-60 DMA User Accessibility Register format

Table 3-107 lists how the bit values correspond with the DMA User Accessibility Register.

Table 3-107 DMA User Accessibility Register bit functions

Bits	Field name	Function
[31:2]	-	UNP/SBZ.
[1]	U1	Indicates if a User mode process can access the registers for channel 1: 0 = User mode cannot access channel 1. User mode accesses cause an Undefined exception. This is the reset value. 1 = User mode can access channel 1.
[0]	U0	Indicates if a User mode process can access the registers for channel 0: 0 = User mode cannot access channel 0. User mode accesses cause an Undefined exception. This is the reset value. 1 = User mode can access channel 0.

Access in the Non-secure world depends on the DMA bit, see *c1, Non-Secure Access Control Register* on page 3-55. The processor can only access this register in Privileged modes. Table 3-108 lists the results of attempted access for each mode.

Table 3-108 Results of access to the DMA User Accessibility Register

DMA bit	Secure Privileged		Non-secure Privileged		User
	Read	Write	Read	Write	
0	Data	Data	Undefined exception	Undefined exception	Undefined exception
1	Data	Data	Data	Data	Undefined exception

To access the DMA User Accessibility Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c11
- CRm set to c1
- Opcode_2 set to0.

For example:

```
MRC p15, 0, <Rd>, c11, c1, 0    ; Read DMA User Accessibility Register
MCR p15, 0, <Rd>, c11, c1, 0    ; Write DMA User Accessibility Register
```

The registers that you can access in User mode when the U bit = 1 for the current channel are:

- *c11, DMA enable registers* on page 3-110
- *c11, DMA Control Register* on page 3-112
- *c11, DMA Internal Start Address Register* on page 3-114
- *c11, DMA External Start Address Register* on page 3-115
- *c11, DMA Internal End Address Register* on page 3-116
- *c11, DMA Channel Status Register* on page 3-117.

You can access the DMA channel Number Register, see *c11, DMA Channel Number Register*, in User mode when the U bit for any channel is 1.

The contents of these registers must be preserved on a task switch if the registers are User-accessible.

If the U bit for the currently selected channel is set to 0, and a User process attempts to access any of these registers the processor takes an Undefined instruction trap.

3.2.35 c11, DMA Channel Number Register

The purpose of the DMA Channel Number Register is to select a DMA channel.

Table 3-109 lists the purposes of the individual bits in the DMA Channel Number Register.

The DMA Channel Number Register is:
- in CP15 c11
- a 32-bit read/write register common to Secure and Non-secure worlds
- accessible in user and privileged modes.

Figure 3-61 shows the bit arrangement for the DMA Channel Number Register.

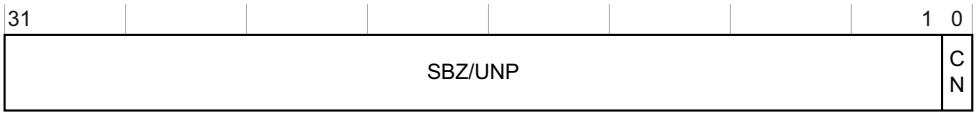

Figure 3-61 DMA Channel Number Register format

Table 3-109 lists how the bit values correspond with the DMA Channel Number Register.

Table 3-109 DMA Channel Number Register bit functions

Bits	Field name	Function
[31:1]	-	UNP/SBZ.
[0]	CN	Indicates DMA Channel selected: 0 = DMA Channel 0 selected, reset value 1 = DMA Channel 1 selected.

Access in the Non-secure world depends on the DMA bit, see *c1, Non-Secure Access Control Register* on page 3-55. The processor can access this register in User mode if the U bit, see *c11, DMA User Accessibility Register* on page 3-107, for any channel is set to 1. Table 3-110 lists the results of attempted access for each mode.

Table 3-110 Results of access to the DMA Channel Number Register

U1 and U0 bits	DMA bit	Secure Privileged Read or Write	Non-secure Privileged Read or Write	Secure User Read or Write	Non-secure User Read or Write
Both 0	0	Data	Undefined exception	Undefined exception	Undefined exception
	1	Data	Data	Undefined exception	Undefined exception
Either or both 1	0	Data	Undefined exception	Data	Undefined exception
	1	Data	Data	Data	Data

To access the DMA Channel Number Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c11
- CRm set to c2
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c11, c2, 0    ; Read DMA Channel Number Register
MCR p15, 0, <Rd>, c11, c2, 0    ; Write DMA Channel Number Register
```

3.2.36 c11, DMA enable registers

The purpose of the DMA enable registers is to start, stop or clear DMA transfers for each channel implemented.

The DMA enable registers are:

- in CP15 c11
- three 32-bit write only registers for each DMA channel common to Secure and Non-secure worlds
- accessible in user and privileged modes.

The commands that operate through the registers are:

Stop The DMA channel ceases to do memory accesses as soon as possible after the level one DMA issues the instruction. This acceleration approach cannot be used for DMA transactions to or from memory regions marked as Device. The DMA can issue a Stop command when the channel status is Running. The DMA channel can take several cycles to stop after the DMA issues a Stop instruction. The channel status remains at Running until the DMA channel stops. The channel status is set to Complete or Error at the point that all outstanding memory accesses complete. The Start Address Registers contain the addresses the DMA requires to restart the operation when the channel stops.

If the Stop command occurs when the channel status is Queued, the channel status changes to Idle. The Stop command has no effect if the channel status is not Running or Queued.

c11, DMA Channel Status Register on page 3-117 describes the DMA channel status.

Start The Start command causes the channel to start DMA transfers. If the other DMA channel is not in operation the channel status is set to Running on the execution of a Start command. If the other DMA channel is in operation the channel status is set to Queued.

A channel is in operation if either:

- its channel status is Queued
- its channel status is Running
- its channel status is Complete or Error, with either the Internal or External Address Error Status indicating an Error.

c11, DMA Channel Status Register on page 3-117 describes DMA channel status.

Clear The Clear command causes the channel status to change from Complete or Error to Idle. It also clears:

- all the Error bits for that DMA channel

- the interrupt that is set by the DMA channel as a result of an error or completion, see *c11, DMA Control Register* on page 3-112 for more details.

The Clear command does not change the contents of the Internal and External Start Address Registers. A Clear command has no effect when the channel status is Running or Queued.

Access in the Non-secure world depends on the DMA bit, see *c1, Non-Secure Access Control Register* on page 3-55. The processor can access these registers in User mode if the U bit, see *c11, DMA User Accessibility Register* on page 3-107, for the currently selected channel is set to 1. Table 3-111 lists the results of attempted access for each mode.

Table 3-111 Results of access to the DMA enable registers

U bit	DMA bit	Secure Privileged		Non-secure Privileged		Secure User		Non-secure User	
		Read	Write	Read	Write	Read	Write	Read	Write
0	0	Undefined exception	Data	Undefined exception	Undefined exception	Undefined exception	Undefined exception	Undefined exception	Undefined exception
	1	Undefined exception	Data	Undefined exception	Data	Undefined exception	Undefined exception	Undefined exception	Undefined exception
1	0	Undefined exception	Data	Undefined exception	Undefined exception	Undefined exception	Data	Undefined exception	Undefined exception
	1	Undefined exception	Data	Undefined exception	Data	Undefined exception	Data	Undefined exception	Data

To access a DMA Enable Register set the DMA Channel Number Register to the appropriate DMA channel and write CP15 with:

- Opcode_1 set to 3
- CRn set to c11
- CRm set to c3
- Opcode_2 set to:
  - 0, Stop
  - 1, Start
  - 2, Clear.

For example:

```
MCR p15, 0, <Rd>, c11, c3, 0    ; Stop DMA Enable Register
MCR p15, 0, <Rd>, c11, c3, 1    ; Start DMA Enable Register
MCR p15, 0, <Rd>, c11, c3, 2    ; Clear DMA Enable Register
```

Debug implications for the DMA

The level one DMA behaves as a separate engine from the processor core, and when started, works autonomously. When the level one DMA has channels with the status of Running or Queued, these channels continue to run, or start running, even if a debug mechanism stops the processor. This can cause the contents of the TCM to change while the processor stops in debug. To avoid this situation you must ensure the level one DMA issues a Stop command to stop Running or Queued channels when entering debug.

3.2.37 c11, DMA Control Register

The purpose of the DMA Control Register for each channel is to control the operations of that DMA channel. Table 3-112 lists the purposes of the individual bits in the DMA Control Register.

The DMA Control Register is:

- in CP15 c11
- one 32-bit read/write register for each DMA channel common to Secure and Non-secure worlds
- accessible in user and privileged modes.

Figure 3-62 shows the bit arrangement for the DMA Control Register.

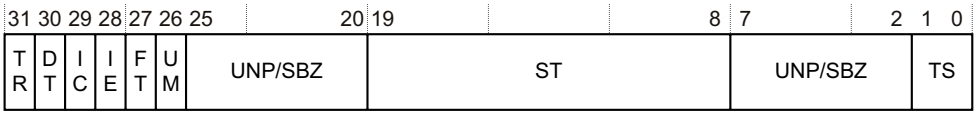

Figure 3-62 DMA Control Register format

Table 3-112 lists how the bit values correspond with the DMA Control Register.

Table 3-112 DMA Control Register bit functions

Bits	Field name	Function
[31]	TR	Indicates target TCM: 0 = Data TCM, reset value 1 = Instruction TCM.
[30]	DT	Indicates direction of transfer: 0 = Transfer from level two memory to the TCM, reset value 1 = Transfer from the TCM to the level two memory.
[29]	IC	Indicates whether the DMA channel must assert an interrupt on completion of the DMA transfer, or if the DMA is stopped by a Stop command, see *c11, DMA enable registers* on page 3-110. The interrupt is deasserted, from this source, if the processor performs a Clear operation on the channel that caused the interrupt. For more details see *c11, DMA enable registers* on page 3-110. The U bit[a] has no effect on whether an interrupt is generated on completion: 0 = No Interrupt on Completion, reset value 1 = Interrupt on Completion.
[28]	IE	Indicates that the DMA channel must assert an interrupt on an error. The interrupt is deasserted, from this source, when the channel is set to Idle with a Clear operation, see *c11, DMA enable registers* on page 3-110: 0 = No Interrupt on Error, if the U bit is 0, reset value 1 = Interrupt on Error, regardless of the U bit[a]. All DMA transactions on channels that have the U bit set to 1 Interrupt on Error regardless of the value written to this bit.
[27]	FT	Read As One, Write ignored In the ARM1176JZF-S this bit has no effect.

Table 3-112 DMA Control Register bit functions (continued)

Bits	Field name	Function
[26]	UM	Indicates that the permission checks are based on the DMA being in User or privileged mode. The UM bit is provided so that the User mode can be emulated by a privileged mode process. For a User mode process the setting of the UM bit is irrelevant and behaves as if set to 1: 0 = Transfer is a privileged transfer, reset value 1 = Transfer is a User mode transfer.
[25:20]	-	UNP/SBZ.
[19:8]	ST	Indicates the increment on the external address between each consecutive access of the DMA. A Stride of zero, reset value, indicates that the external address is not to be incremented. This is designed to facilitate the accessing of volatile locations such as a FIFO. The Stride is interpreted as a positive number, or zero. The internal address increment is not affected by the Stride, but is fixed at the transaction size. The stride value is in bytes. The value of the Stride must be aligned to the Transaction Size, otherwise this results in a bad parameter error, see *c11, DMA Channel Status Register* on page 3-117.
[7:2]	-	UNP/SBZ.
[1:0]	TS	Indicates the size of the transactions that the DMA channel performs. This is particularly important for Device or Strongly Ordered memory locations because it ensures that accesses to such memory occur at their programmed size: b00 = Byte, reset value b01 = Halfword b10 = Word b11 = Doubleword, 8 bytes.

a. See *c11, DMA User Accessibility Register* on page 3-107.

Access in the Non-secure world depends on the DMA bit, see *c1, Non-Secure Access Control Register* on page 3-55. The processor can access this register in User mode if the U bit, see *c11, DMA User Accessibility Register* on page 3-107, for the currently selected channel is set to 1. Table 3-113 lists the results of attempted access for each mode.

Table 3-113 Results of access to the DMA Control Register

U bit	DMA bit	Secure Privileged Read or Write	Non-secure Privileged Read or Write	Secure User Read or Write	Non-secure User Read or Write
0	0	Data	Undefined exception	Undefined exception	Undefined exception
	1	Data	Data	Undefined exception	Undefined exception
1	0	Data	Undefined exception	Data	Undefined exception
	1	Data	Data	Data	Data

To access the DMA Control Register set the DMA Channel Number Register to the appropriate DMA channel and read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c11
- CRm set to c4
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c11, c4, 0    ; Read DMA Control Register
MCR p15, 0, <Rd>, c11, c4, 0    ; Write DMA Control Register
```

While the channel has the status of Running or Queued, any attempt to write to the DMA Control Register results in architecturally Unpredictable behavior. For ARM1176JZF-S processors writes to the DMA Control Register have no effect when the DMA channel is running or queued.

3.2.38 c11, DMA Internal Start Address Register

The purpose of the DMA Internal Start Address Register for each channel is to define the first address in the TCM for that channel. That is, it defines the first address that data transfers go to or from.

The DMA Internal Start Address Register is:

- in CP15 c11
- one 32-bit read/write register for each DMA channel common to Secure and Non-secure worlds
- accessible in user and privileged modes.

The DMA Internal Start Address Register bits [31:0] contain the Internal Start VA.

Access in the Non-secure world depends on the DMA bit, see *c1, Non-Secure Access Control Register* on page 3-55. The processor can access this register in User mode if the U bit, see *c11, DMA User Accessibility Register* on page 3-107, for the currently selected channel is set to 1. Table 3-114 lists the results of attempted access for each mode.

Table 3-114 Results of access to the DMA Internal Start Address Register

U bit	DMA bit	Secure Privileged Read or Write	Non-secure Privileged Read or Write	Secure User Read or Write	Non-secure User Read or Write
0	0	Data	Undefined exception	Undefined exception	Undefined exception
	1	Data	Data	Undefined exception	Undefined exception
1	0	Data	Undefined exception	Data	Undefined exception
	1	Data	Data	Data	Data

To access the DMA Internal Start Address Register set the DMA Channel Number Register to the appropriate DMA channel and read or write CP15 c11 with:

- Opcode_1 set to 0
- CRn set to c11
- CRm set to c5
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c11, c5, 0    ; Read DMA Internal Start Address Register
MCR p15, 0, <Rd>, c11, c5, 0    ; Write DMA Internal Start Address Register
```

The Internal Start Address is a VA. Page tables describe the physical mapping of the VA when the channel starts.

The memory attributes for that VA are used in the transfer, so memory permission faults might be generated. The Internal Start Address must lie within a TCM, otherwise an error is reported in the DMA Channel Status Register. The marking of memory locations in the TCM as being Device results in Unpredictable effects. The global system behavior, but not the security, can be affected.

The contents of this register do not change while the DMA channel is Running. When the channel is stopped because of a Stop command, or an error, it contains the address required to restart the transaction. On completion, it contains the address equal to the Internal End Address.

The Internal Start Address must be aligned to the transaction size set in the DMA Control Register or the processor generates a bad parameter error.

3.2.39 c11, DMA External Start Address Register

The purpose of the DMA External Start Address Register for each channel is to define the first address in external memory for that DMA channel. That is, it defines the first address that data transfers go to or from.

The DMA External Start Address Register is:

- in CP15 c11
- one 32-bit read/write register for each DMA channel common to Secure and Non-secure worlds
- accessible in user and privileged modes.

The DMA External Start Address Register bits [31:0] contain the External Start VA.

Access in the Non-secure world depends on the DMA bit, see *c1, Non-Secure Access Control Register* on page 3-55. The processor can access this register in User mode if the U bit, see *c11, DMA User Accessibility Register* on page 3-107, for the currently selected channel is set to 1. Table 3-115 lists the results of attempted access for each mode.

Table 3-115 Results of access to the DMA External Start Address Register

U bit	DMA bit	Secure Privileged Read or Write	Non-secure Privileged Read or Write	Secure User Read or Write	Non-secure User Read or Write
0	0	Data	Undefined exception	Undefined exception	Undefined exception
	1	Data	Data	Undefined exception	Undefined exception
1	0	Data	Undefined exception	Data	Undefined exception
	1	Data	Data	Data	Data

To access the DMA External Start Address Register set the DMA Channel Number Register to the appropriate DMA channel and read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c11
- CRm set to c6
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c11, c6, 0    ; Read DMA External Start Address Register
MCR p15, 0, <Rd>, c11, c6, 0    ; Write DMA External Start Address Register
```

The External Start Address is a VA, the physical mapping that you must describe in the page tables at the time that the channel is started. The memory attributes for that VA are used in the transfer, so memory permission faults might be generated.

The External Start Address must lie in the external memory outside the level one memory system otherwise the results are Unpredictable. The global system behavior, but not the security, can be affected.

This register contents do not change while the DMA channel is Running. When the channel stops because of a Stop command, or an error, it contains the address that the DMA requires to restart the transaction. On completion, it contains the address equal to the final address of the transfer accessed plus the Stride.

If the External Start Address does not align with the transaction size that is set in the Control Register, the processor generates a bad parameter error.

3.2.40 c11, DMA Internal End Address Register

The purpose of the DMA Internal End Address Register for each channel is to define the final internal address for that channel. This is, the end address of the data transfer.

The DMA Internal End Address Register is:

- in CP15 c11
- one 32-bit read/write register for each DMA channel common to Secure and Non-secure worlds
- accessible in user and privileged modes.

The DMA Internal End Address Register bits [31:0] contain the Internal End VA.

Access in the Non-secure world depends on the DMA bit, see *c1, Non-Secure Access Control Register* on page 3-55. The processor can access this register in User mode if the U bit, see *c11, DMA User Accessibility Register* on page 3-107, for the currently selected channel is set to 1. Table 3-116 lists the results of attempted access for each mode.

Table 3-116 Results of access to the DMA Internal End Address Register

U bit	DMA bit	Secure Privileged Read or Write	Non-secure Privileged Read or Write	Secure User Read or Write	Non-secure User Read or Write
0	0	Data	Undefined exception	Undefined exception	Undefined exception
	1	Data	Data	Undefined exception	Undefined exception
1	0	Data	Undefined exception	Data	Undefined exception
	1	Data	Data	Data	Data

To access the DMA Internal End Address Register set the DMA Channel Number Register to the appropriate DMA channel and read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c11
- CRm set to c7
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c11, c7, 0    ; Read DMA Internal End Address Register
MCR p15, 0, <Rd>, c11, c7, 0    ; Write DMA Internal End Address Register
```

The Internal End Address is the final internal address, modulo the transaction size, that the DMA is to access plus the transaction size. Therefore, the Internal End Address is the first, incremented, address that the DMA does not access.

If the Internal End Address is the same of the Internal Start Address, the DMA transfer completes immediately without performing transactions.

When the transaction associated with the final internal address has completed, the whole DMA transfer is complete.

The Internal End Address is a VA. Page tables describe the physical mapping of the VA when the channel starts.

The memory attributes for that VA are used in the transfer, so memory permission faults might be generated. The Internal End Address must lie within a TCM, otherwise an error is reported in the DMA Channel Status Register. The marking of memory locations in the TCM as being Device results in Unpredictable effects. The global system behavior, but not the security, can be affected.

The Internal End Address must be aligned to the transaction size set in the DMA Control Register or the processor generates a bad parameter error.

3.2.41 c11, DMA Channel Status Register

The purpose of the DMA Channel Status Register for each channel is to define the status of the most recently started DMA operation on that channel.

The DMA Channel Status Register is:

- in CP15 c11
- one 32-bit read-only register for each DMA channel common to Secure and Non-secure worlds
- accessible in user and privileged modes.

Figure 3-63 shows the bit arrangement for the DMA Channel Status Register.

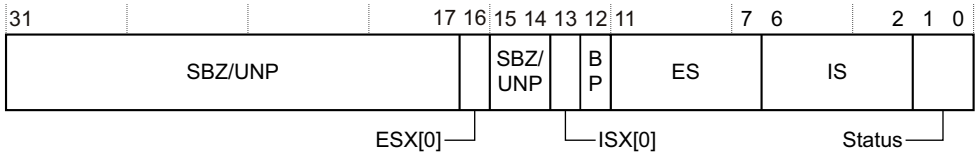

Figure 3-63 DMA Channel Status Register format

Table 3-117 lists the functions of the bits in the DMA Channel Status Register.

Table 3-117 DMA Channel Status Register bit functions

Bits	Field name	Function
[31:17]	-	UNP/SBZ.
[16]	ESX[0]	The ESX[0] bit adds a SLVERR or DECERR qualifier to the ES encoding. Only predictable on ES encodings of b11010, b11100, and b1.1110, otherwise UNP/SBZ. For the predictable encodings:0 = DECERR1 = SLVERR.
[15:14]	-	UNP/SBZ.
[13]	ISX[0]	The ISX[0] bit adds a SLVERR or DECERR qualifier to the IS encoding. Only predictable on IS encodings of b11100 and b11110, otherwise UNP/SBZ. For the predictable encodings:0 = DECERR1 = SLVERR.

Table 3-117 DMA Channel Status Register bit functions (continued)

Bits	Field name	Function
[12]	BP[a]	Indicates whether the DMA parameters are conditioned inappropriately or acceptable: 0 = DMA parameters are acceptable, reset value 1 = DMA parameters are conditioned inappropriately.
[11:7]	ES	Indicates the status of the External Address Error. All other encodings are Reserved: b00000 = No error, reset value b00xxx = No error b01001 = Unshared data error b11010 = External Abort, can be imprecise b11100 = External Abort on translation of first-level page table b11110 = External Abort on translation of second-level page table b10011 = Access Bit fault on section b10110 = Access Bit fault on page b10101 = Translation fault, section b10111 = Translation fault, page b11001 = Domain fault, section b11011 = Domain fault, page b11101 = Permission fault, section b11111 = Permission fault, page.
[6:2]	IS	Indicates the status of the Internal Address Error. All other encodings are Reserved: b00000 = No error, reset value b00xxx = No error b01000 = TCM out of range b11100 = External Abort on translation of first-level page table b11110 = External Abort on translation of second-level page table b10011 = Access Bit fault on section b10110 = Access Bit fault on page b10101 = Translation fault, section b10111 = Translation fault, page b11001 = Domain fault, section b11011 = Domain fault, page b11101 = Permission fault, section b11111 = Permission fault, page.
[1:0]	Status	Indicates the status of the DMA channel: b00 = Idle, reset value b01 = Queued b10 = Running b11 = Complete or Error.

a. The external start and end addresses and the Stride must all be multiples of the transaction size. If this is not the case, the BP bit is set to 1, and the DMA channel does not start.

Access in the Non-secure world depends on the DMA bit, see *c1, Non-Secure Access Control Register* on page 3-55. These registers can be accessed in User mode if the U bit, see *c11, DMA User Accessibility Register* on page 3-107, for the currently selected channel is set to 1. Table 3-118 lists the results of attempted access for each mode.

Table 3-118 Results of access to the DMA Channel Status Register

U bit	DMA bit	Secure Privileged		Non-secure Privileged		Secure User		Non-secure User	
		Read	Write	Read	Write	Read	Write	Read	Write
0	0	Data	Undefined exception	Undefined exception	Undefined exception	Undefined exception	Undefined exception	Undefined exception	Undefined exception
	1	Data	Undefined exception	Data	Undefined exception	Undefined exception	Undefined exception	Undefined exception	Undefined exception
1	0	Data	Undefined exception	Undefined exception	Undefined exception	Data	Undefined exception	Undefined exception	Undefined exception
	1	Data	Undefined exception	Data	Undefined exception	Data	Undefined exception	Data	Undefined exception

To access the DMA Channel Status Register set DMA Channel Number Register to the appropriate DMA channel and read CP15 with:

- Opcode_1 set to 0
- CRn set to c11
- CRm set to c8
- Opcode_2 set to 0.

```
MRC p15, 0, <Rd>, c11, c8, 0    ; Read DMA Channel Status Register
```

In the event of an error, the appropriate Start Address Register contains the address that faulted, unless the error is an external error that is set to b11010 by bits [11:7].

A channel with the state of Queued changes to Running automatically if the other channel, if implemented, changes to Idle, or Complete or Error, with no error.

When a channel completes all of the transfers of the DMA, so that all changes to memory locations caused by those transfers are visible to other observers, its status is changed from Running to Complete or Error. This change does not happen before the external accesses from the transfer complete.

If the processor attempts to access memory locations that are not marked as shared, then the ES bits signal an Unshared error for either:

- a DMA transfer in User mode
- a DMA transfer that has the UM bit set in the DMA Control Register.

A DMA transfer where the external address is within the range of the TCM also results in an Unshared data error.

If the DMA channel is configured Secure, in the event of an error the processor asserts the **nDMASIRQ** pin provided it is not masked. If the channel is configured Non-secure, in the event of an error the processor asserts the **nDMAIRQ** pin, provided it is not masked. In the event of an external abort on a page table walk caused by the DMA, the processor asserts the **nDMAEXTERRIRQ** output.

3.2.42 c11, DMA Context ID Register

The DMA Context ID Register for each channel contains the processor 32-bit Context ID of the process that uses that channel.

The DMA Context ID Register is:

- in CP15 c11
- a 32-bit read/write register for each DMA channel common to Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-64 shows the arrangement of bits in the DMA Context ID Register.

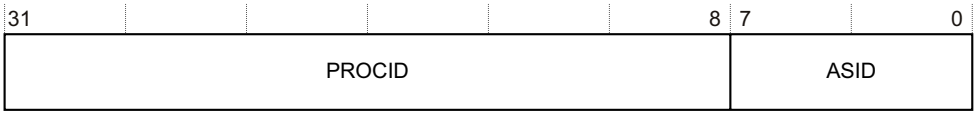

Figure 3-64 DMA Context ID Register format

Table 3-119 lists how the bit values correspond with the DMA Context ID Register functions.

Table 3-119 DMA Context ID Register bit functions

Bits	Field name	Function
[31:8]	PROCID	Extends the ASID to form the process ID and identify the current process Holds the process ID value
[8:0]	ASID	Holds the ASID of the current process and identifies the current ASID Holds the ASID value

Access in the Non-secure world depends on the DMA bit, see *c1, Non-Secure Access Control Register* on page 3-55. Table 3-120 lists the results of attempted access for each mode.

Table 3-120 Results of access to the DMA Context ID Register

DMA bit	Secure Privileged		Non-secure Privileged		User
	Read	Write	Read	Write	
0	Data	Data	Undefined exception	Undefined exception	Undefined exception
1	Data	Data	Data	Data	Undefined exception

To access the DMA Context ID register in a privileged mode set the DMA Channel Number Register to the appropriate DMA channel and read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c11
- CRm set to c15
- Opcode_2 set to 0.

```
MRC p15, 0, <Rd>, c11, c15, 0    ; Read DMA Context ID Register
MCR p15, 0, <Rd>, c11, c15, 0    ; Write DMA Context ID Register
```

As part of the initialization of the DMA channel, the process that uses that channel writes the processor Context ID to the DMA Context ID Register. Where the channel is designated as a User-accessible channel, the privileged process, that initializes the channel for use in User mode, must write the Context ID at the same time that the software writes to the U bit for the channel.

The process that translates VAs to physical addresses uses the ASID stored in the bottom eight bits of the Context ID register to enable different VA maps to co-exist. Attempts to write this register while the DMA channel is Running or Queued has no effect.

Only privileged processes can read this register. This provides anonymity of the DMA channel usage from User processes. On a context switch, where the state of the DMA is stacked and restored, the saved state must include this register.

If a user process attempts to access this privileged register the processor takes an Undefined instruction trap.

3.2.43 c12, Secure or Non-secure Vector Base Address Register

The purpose of the Secure or Non-secure Vector Base Address Register is to hold the base address for exception vectors in the Secure and Non-secure worlds. For more information, see *Exceptions* on page 2-36.

The Secure or Non-secure Vector Base Address Register is:

- in CP15 c12
- a 32-bit read/write register banked in Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-65 shows the arrangement of bits in the register.

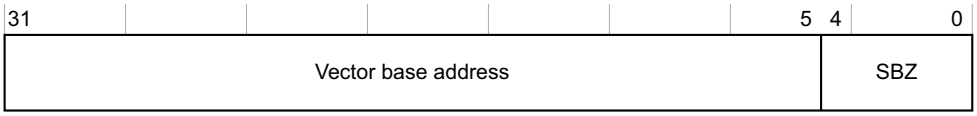

Figure 3-65 Secure or Non-secure Vector Base Address Register format

Table 3-121 lists how the bit values correspond with the Secure or Non-secure Vector Base Address Register functions.

Table 3-121 Secure or Non-secure Vector Base Address Register bit functions

Bits	Field name	Function
[31:5]	Vector base address	Determines the location that the core branches to on an exception Holds the base address. The reset value is 0.
[4:0]	SBZ	UNP/SBZ.

When an exception occurs in the Secure world, the core branches to address:

Secure Vector_Base_Address + Exception_Vector_Address.

When an exception occurs in the Non-secure world, the core branches to address:

Non-secure Vector_Base_Address + Exception_Vector_Address.

When high vectors are enabled, regardless of the value of the register the core branches to:

`0xFFFF0000` + Exception_Vector_Address

You can configure IRQ, FIQ, and External abort exceptions to branch to Secure Monitor mode, see *c1, Secure Configuration Register* on page 3-52. In this case the processor uses the Monitor Vector Base Address, see *c12, Monitor Vector Base Address Register*, to calculate the branch address. The Reset exception always branches to `0x00000000`, regardless of the value of the Vector Base Address except when the processor uses high vectors.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Table 3-122 lists the results of attempted access for each mode.

Table 3-122 Results of access to the Secure or Non-secure Vector Base Address Register

Secure Privileged		Non-secure Privileged		User
Read	**Write**	**Read**	**Write**	
Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception

To use the Secure or Non-secure Vector Base Address Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c12
- CRm set to c0
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c12, c0, 0    ; Read Secure or Non-secure Vector Base Address Register
MCR p15, 0, <Rd>, c12, c0, 0    ; Write Secure or Non-secure Vector Base Address Register
```

3.2.44 c12, Monitor Vector Base Address Register

The purpose of the Monitor Vector Base Address Register is to hold the base address for the Secure Monitor exception vector. For more information, see *Exceptions* on page 2-36.

The Monitor Vector Base Address Register is:

- in CP15 c12
- a 32-bit read/write register in the Secure world only
- accessible in Secure privileged modes only.

Figure 3-66 shows the arrangement of bits in the register.

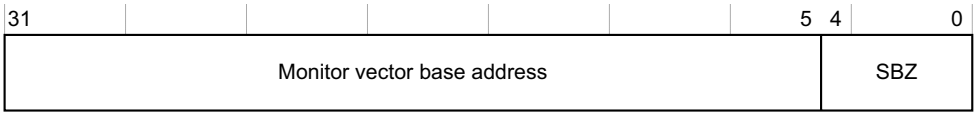

Figure 3-66 Monitor Vector Base Address Register format

Table 3-123 lists how the bit values correspond with the Monitor Vector Base Address Register functions.

Table 3-123 Monitor Vector Base Address Register bit functions

Bits	Field name	Function
[31:5]	Monitor vector base address	Determines the location that the core branches to on a Secure Monitor mode exception. Holds the base address. The reset value is 0.
[4:0]	SBZ	UNP/SBZ.

When an exception branches to the Secure Monitor mode, the core branches to address:

Monitor_Base_Address + Exception_Vector_Address.

The Secure Monitor Call Exception caused by an SMC instruction branches to Secure Monitor mode. You can configure IRQ, FIQ, and External abort exceptions to branch to Secure Monitor mode, see *c1, Secure Configuration Register* on page 3-52. These are the only exceptions that can branch to Secure Monitor mode and that use the Monitor Vector Base Address Register to calculate the branch address. For more information about exceptions, see *Exception vectors* on page 2-48.

Note

The Monitor Vector Base Address Register is `0x00000000` at reset. The Secure boot code must program the register with an appropriate value for the Secure Monitor.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Table 3-124 lists the results of attempted access for each mode.

Table 3-124 Results of access to the Monitor Vector Base Address Register

Secure Privileged		Non-secure Privileged	User
Read	Write		
Data	Data	Undefined exception	Undefined exception

To use the Monitor Vector Base Address Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c12
- CRm set to c0
- Opcode_2 set to 1.

For example:

```
MRC p15, 0, <Rd>, c12, c0, 1    ; Read Monitor Vector Base Address Register
MCR p15, 0, <Rd>, c12, c0, 1    ; Write Monitor Vector Base Address Register
```

3.2.45 c12, Interrupt Status Register

The purpose of the Interrupt Status Register is to:

- reflect the state of the **nFIQ** and **nIRQ** pins on the processor
- to reflect the state of external aborts.

The Interrupt Status Register is:

- in CP15 c12
- a 32-bit read-only register common to Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-67 shows the arrangement of bits in the register.

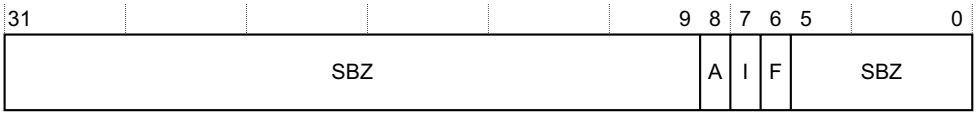

Figure 3-67 Interrupt Status Register format

Table 3-125 lists how the bit values correspond with the Interrupt Status Register functions.

Table 3-125 Interrupt Status Register bit functions

Bits	Field name	Function[a]
[31:9]	-	SBZ.
[8]	A	Indicates when an external abort is pending: 0 = No abort, reset value 1 = Abort pending.
[7]	I	Indicates when an IRQ is pending: 0 = no IRQ, reset value 1 = IRQ pending.
[6]	F	Indicates when an FIQ is pending: 0 = no FIQ, reset value 1 = FIQ pending.
[5:0]	-	SBZ.

a. The reset values depend on external signals.

Note

- The F and I bits directly reflect the state of the **nFIQ** and **nIRQ** pins respectively, but are the inverse state.
- The A bit is set when an external abort occurs and automatically clears when the abort is taken.

Table 3-126 lists the results of attempted access for each mode.

Table 3-126 Results of access to the Interrupt Status Register

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Undefined exception	Data	Undefined exception	Undefined exception

The A, I, and F bits map to the same format as the CPSR so that you can use the same mask for these bits.

The Secure Monitor can poll these bits to detect the exceptions before it completes context switches. This can reduce interrupt latency.

To use the Interrupt Status Register read CP15 with:

- Opcode_1 set to 0
- CRn set to c12
- CRm set to c1
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c12, c1, 0    ; Read Interrupt Status Register
```

3.2.46 c13, FCSE PID Register

The *c13, Context ID Register* on page 3-128 replaces the FCSE PID Register. Use of the FCSE PID Register is deprecated.

The FCSE PID Register is:

- in CP15 c13
- a 32-bit read/write register banked for Secure and Non-secure worlds
- accessible in privileged modes only.

Writing to this register globally flushes the BTAC.

Figure 3-68 shows the arrangement of bits in the register.

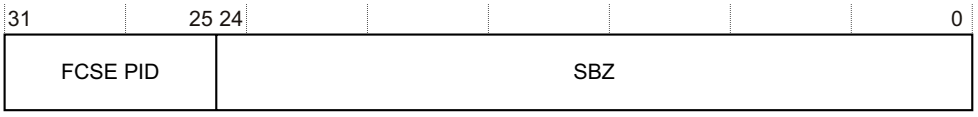

Figure 3-68 FCSE PID Register format

Table 3-127 lists how the bit values correspond with the FCSE PID Register functions.

Table 3-127 FCSE PID Register bit functions

Bits	Field name	Function
[31:25]	FCSE PID	The purpose of the FCSE PID Register is to provide the ProcID for fast context switch memory mappings. The MMU uses the contents of this register to map memory addresses in the range 0-32MB. Identifies a specific process for fast context switch. Holds the ProcID. The reset value is 0.
[24:0]	-	Reserved. SBZ.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Table 3-128 lists the results of attempted access for each mode.

Table 3-128 Results of access to the FCSE PID Register

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception

To use the FCSE PID Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c13
- CRm set to c0
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c13, c0, 0    ; Read FCSE PID Register
MCR p15, 0, <Rd>, c13, c0, 0    ; Write FCSE PID Register
```

To change the ProcID and perform a fast context switch, write to the FCSE PID Register. You do not have to flush the contents of the TLB after the switch because the TLB still holds the valid address tags.

From zero to six instructions after the MCR that writes the ProcID might be fetched with the old ProcID:

```
{ProcID = 0}
MOV R0, #1                              ; Fetched with ProcID = 0
MCR p15,0,R0,c13,c0,0                   ; Fetched with ProcID = 0
A0    (any instruction)                 ; Fetched with ProcID = 0/1
A1    (any instruction)                 ; Fetched with ProcID = 0/1
A2    (any instruction)                 ; Fetched with ProcID = 0/1
A3    (any instruction)                 ; Fetched with ProcID = 0/1
A4    (any instruction)                 ; Fetched with ProcID = 0/1
A5    (any instruction)                 ; Fetched with ProcID = 0/1
A6    (any instruction)                 ; Fetched with ProcID = 1
```

Note

You must not rely on this behavior for future compatibility. An IMB must be executed between changing the ProcID and fetching from locations that are translated by the ProcID.

Addresses issued by the ARM1176JZF-S processor in the range 0-32MB are translated by the ProcID. Address A becomes A + (ProcID x 32MB). This translated address, the MVA, is used by the MMU. Addresses higher than 32MB are not translated. The ProcID is a seven-bit field, enabling 128 x 32MB processes to be mapped.

Note

If ProcID is 0, as it is on Reset, then there is a flat mapping between the ARM1176JZF-S processor and the MMU.

Figure 3-69 shows how addresses are mapped using the FCSE PID Register.

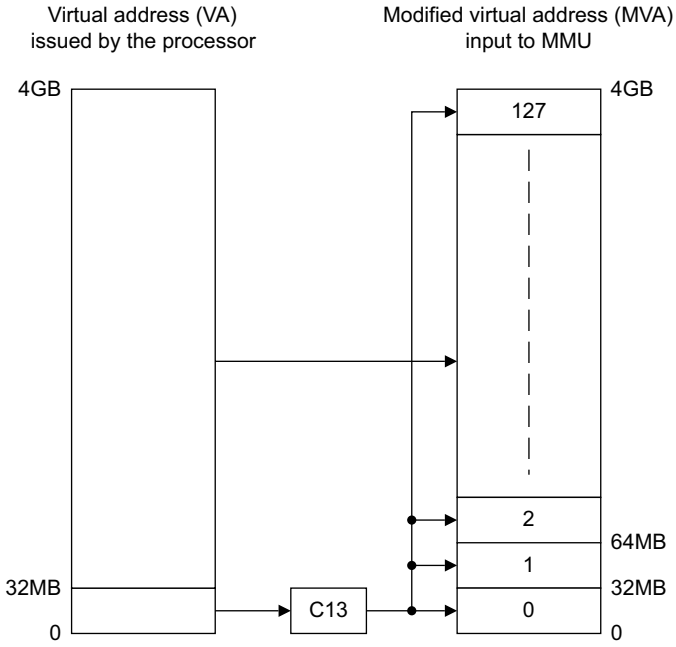

Figure 3-69 Address mapping with the FCSE PID Register

3.2.47 c13, Context ID Register

The purpose of the Context ID Register is to provide information on the current ASID and process ID, for example for the ETM and debug logic.

Table 3-129 lists the purposes of the individual bits of the Context ID Register.

Debug logic uses the ASID information to enable process-dependent breakpoints and watchpoints.

The Context ID Register is:
- in CP15 c13
- a 32-bit read/write register banked for Secure and Non-secure worlds
- accessible in privileged modes only.

Writing to this register globally flushes the BTAC.

Figure 3-70 shows the arrangement of bits in the Context ID Register.

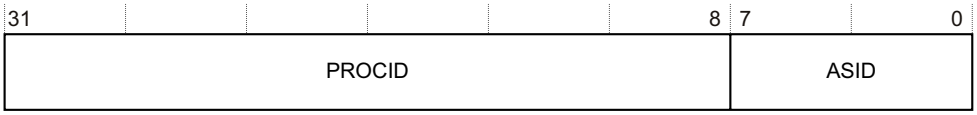

Figure 3-70 Context ID Register format

Table 3-129 lists how the bit values correspond with the Context ID Register functions.

Table 3-129 Context ID Register bit functions

Bits	Field name	Function
[31:8]	PROCID	Extends the ASID to form the process ID and identify the current process. The value is the Process ID. The reset value is 0.
[8:0]	ASID	Holds the ASID of the current process to identify the current ASID. The value is the ASID. The reset value is 0.

Table 3-130 lists the results of attempted access for each mode.

Table 3-130 Results of access to the Context ID Register

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception

The current ASID value in the ID Context Register is exported to the MMU.

To use the Context ID Register read or write CP15 with:
- Opcode_1 set to 0
- CRn set to c13
- CRm set to c0
- Opcode_2 set to 1.

For example:

```
MRC p15, 0, <Rd>, c13, c0, 1    ;Read Context ID Register
```

```
MCR p15, 0, <Rd>, c13, c0, 1    ;Write Context ID Register
```

You must ensure that software performs a Data Synchronization Barrier operation before changes to this register. This ensures that all accesses are related to the correct context ID.

You must execute an IMB instruction immediately after changes to the Context ID Register. You must not attempt to execute any instructions that are from an ASID-dependent memory region between the change to the register and the IMB instruction. Code that updates the ASID must execute from a global memory region.

You must program each process with a unique number to ensure that ETM and debug logic can correctly distinguish between processes.

3.2.48 c13, Thread and process ID registers

The purpose of the thread and process ID registers is to provide locations to store the IDs of software threads and processes for OS management purposes.

The thread and process ID registers are:

- in CP15 c13
- three 32-bit read/write registers banked for Secure and Non-secure worlds:
  - User Read/Write Thread and Process ID Register
  - User Read Only Thread and Process ID Register
  - Privileged Only Thread and Process ID Register.
- each accessible in different modes:
  - User Read/Write: read/write in User and privileged modes
  - User Read Only: read only in User mode, read/write in privileged modes
  - Privileged Only: read/write in privileged modes only.

Table 3-131 lists the results of attempted access to each register for each mode.

Table 3-131 Results of access to the thread and process ID registers

Thread and Process ID Register	Secure Privileged		Non-secure Privileged		Secure User		Non-secure User	
	Read	Write	Read	Write	Read	Write	Read	Write
User Read/ Write[a]	Secure data	Secure data	Non-secure data	Non-secure data	Secure data	Secure data	Non-secure data	Non-secure data
User Read Only[a]	Secure data	Secure data	Non-secure data	Non-secure data	Secure data	Undefined exception	Non-secure data	Undefined exception
Privileged Only[a]	Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception	Undefined exception	Undefined exception	Undefined exception

a. The register names are:
- User Read/Write Thread and Process ID Register
- User Read Only Thread and Process ID Register
- Privileged Only Thread and Process ID Register.

To use the thread and process ID registers read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c13
- CRm set to c0

- Opcode_2 set to:
  - 2, User Read/Write Thread and Process ID Register
  - 3, User Read Only Thread and Process ID Register
  - 4, Privileged Only Thread and Process ID Register.

For example:

```
MRC p15, 0, <Rd>, c13, c0, 2    ;Read User Read/Write Thread and Proc. ID Register
MCR p15, 0, <Rd>, c13, c0, 2    ;Write User Read/Write Thread and Proc. ID Register
MRC p15, 0, <Rd>, c13, c0, 3    ;Read User Read Only Thread and Proc. ID Register
MCR p15, 0, <Rd>, c13, c0, 3    ;Write User Read Only Thread and Proc. ID Register
MRC p15, 0, <Rd>, c13, c0, 4    ;Read Privileged Only Thread and Proc. ID Register
MCR p15, 0, <Rd>, c13, c0, 4    ;Write Privileged Only Thread and Proc. ID Register
```

Reading or writing the thread and process ID registers has no effect on processor state or operation. These registers provide OS support and must be managed by the OS.

You must clear the contents of all thread and process ID registers on process switches to prevent data leaking from one process to another. This is important to ensure the security of secure data. The reset value of these registers is 0.

3.2.49 c15, Peripheral Port Memory Remap Register

The purpose of the Peripheral Port Memory Remap Register is to remap the memory attributes to Non-Shared Device. This forces access to the peripheral port and overrides what is programmed in the page tables. The remapping happens both with the MMU enabled and with the MMU disabled, therefore you can remap the peripheral port even when you do not use the MMU. The Peripheral Port Memory Remap Register has the highest priority, higher than that of the Primary and Normal memory remap registers.

Table 3-132 on page 3-131 lists the purposes of the individual bits in the Peripheral Port Memory Remap Register.

The Peripheral Port Memory Remap Register is:

- in CP15 c15
- a 32-bit read/write register banked for Secure and Non-secure worlds
- accessible in privileged modes only.

Figure 3-71 shows the arrangement of the bits in the register.

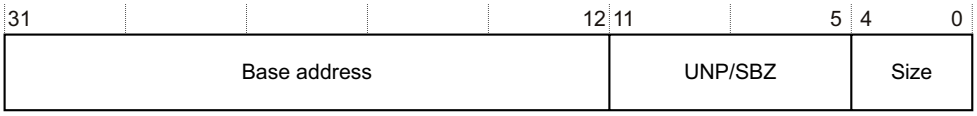

Figure 3-71 Peripheral Port Memory Remap Register format

Table 3-132 lists how the bit values correspond with the functions of the Peripheral Port Memory Remap Register.

Table 3-132 Peripheral Port Memory Remap Register bit functions

Bits	Field name	Function
[31:12]	Base Address	Gives the physical base address of the region of memory for remapping to the peripheral port. If the processor uses the Peripheral Port Memory Remap Register while the MMU is disabled, the virtual base address is equal to the physical base address that is used. The assumption is that the Base Address is aligned to the size of the remapped region. Any bits in the range [($\log_2$(Region size)-1):12] are ignored. The value is the base address. The reset value is 0.
[11:5]	-	UNP/SBZ
[4:0]	Size	Indicates the size of the memory region that the peripheral port is remapped to. All other values are reserved: b00000 = 0KB[a] b00011 = 4KB b00100 = 8KB b00101 = 16KB b00110 = 32KB b00111 = 64KB b01000 = 128KB b01001 = 256KB b01010 = 512KB b01011 = 1MB b01100 = 2MB b01101 = 4MB b01110 = 8MB b01111 = 16MB b10000 = 32MB b10001 = 64MB b10010 = 128MB b10011 = 256MB b10100 = 512MB b10101 = 1GB b10110 = 2GB.

a. The reset value, indicating that no remapping is to take place.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Table 3-133 lists the results of attempted access for each mode.

Table 3-133 Results of access to the Peripheral Port Remap Register

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Secure data	Secure data	Non-secure data	Non-secure data	Undefined exception

To use the memory remap registers read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c15
- CRm set to c2
- Opcode_2 set to 4.

For example:

```
MRC p15, 0, <Rd>, c15, c2, 4    ; Read Peripheral Port Memory Remap Register
MCR p15, 0, <Rd>, c15, c2, 4    ; Write Peripheral Port Memory Remap Register
```

3.2.50 c15, Secure User and Non-secure Access Validation Control Register

The purpose of the Secure User and Non-secure Access Validation Control Register is to control:

- access to the system validation registers in User mode and in the Non-secure world
- access to the performance monitor unit registers in User mode.

Table 3-134 lists the purpose of the individual bits in the register.

The Secure User and Non-secure Access Validation Control Register is:

- in CP15 c15
- a 32-bit read/write register in the Secure world only
- accessible in privileged modes only.

Figure 3-72 shows the bit arrangement for the Secure User and Non-secure Access Validation Control Register.

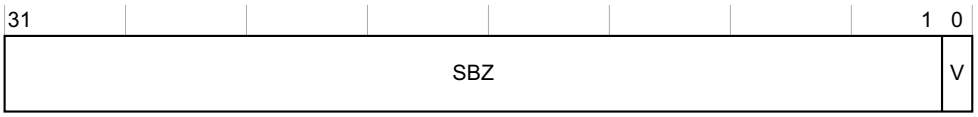

Figure 3-72 Secure User and Non-secure Access Validation Control Register format

Table 3-134 lists how the bit values correspond with the Secure User and Non-secure Access Validation Control Register functions.

Table 3-134 Secure User and Non-secure Access Validation Control Register bit functions

Bits	Field name	Function
[31:1]	-	UNP/SBZ.
[0]	V	Controls access to system validation registers from User and Non-secure modes, and to performance monitor registers in User mode. 0 = system validation registers accessible only from Secure privileged modes, performance monitor registers accessible only from privileged modes. The reset value is 0. 1 = system validation and performance monitor registers accessible from any mode.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Table 3-135 lists the results of attempted access for each mode.

Table 3-135 Results of access to the Secure User and Non-secure Access Validation Control Register

Secure Privileged		Non-secure Privileged	User
Read	Write		
Data	Data	Undefined exception	Undefined exception

To access the Secure User and Non-secure Access Validation Control Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c15
- CRm set to c9
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c15, c9, 0    ; Read Secure User and Non-secure Access Validation Control Register
MCR p15, 0, <Rd>, c15, c9, 0    ; Write Secure User and Non-secure Access Validation Control Register
```

3.2.51 c15, Performance Monitor Control Register

The purpose of the Performance Monitor Control Register is to control the operation of:

- the Cycle Counter Register
- the Count Register 0
- the Count Register 1.

Table 3-136 on page 3-134 lists the purpose of the individual bits in the register.

The Performance Monitor Control Register is:

- in CP15 c15
- a 32-bit read/write register common to Secure and Non-secure worlds
- accessible in User and Privileged modes.

Figure 3-73 shows the bit arrangement for the Performance Monitor Control Register.

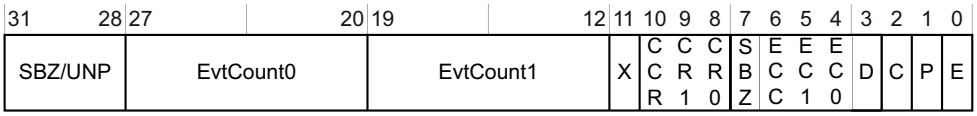

Figure 3-73 Performance Monitor Control Register format

Table 3-136 lists how the bit values correspond with the Performance Monitor Control Register.

Table 3-136 Performance Monitor Control Register bit functions

Bits	Field name	Function
[31:28]	-	UNP/SBZ.
[27:20]	EvtCount0	Identifies the source of events for Count Register 0. Table 3-137 on page 3-135 lists the values, functions and EVNTBUS bit position for Count Register 0. The reset value is 0.
[19:12]	EvtCount1	Identifies the source of events for Count Register 1. Table 3-137 on page 3-135 lists the values and the bit functions for Count Register 1. The reset value is 0.
[11]	X	Enable Export of the events to the event bus to an external monitoring block, such as the ETM to trace events: 0 = Export disabled, **EVNTBUS** held at `0x0`, reset value 1 = Export enabled, **EVNTBUS** driven by the events.
[10]	CCR	Cycle Counter Register overflow flag: 0 = For reads No overflow, reset value. For writes No effect. 1 = For reads, overflow occurred. For writes Clear this bit.
[9]	CR1	Count Register 1 overflow flag: 0 = For reads No overflow, reset value. For writes No effect. 1 = For reads, overflow occurred. For writes Clear this bit.
[8]	CR0	Count Register 0 overflow flag: 0 = For reads No overflow, reset value. For writes No effect. 1 = For reads overflow occurred. For writes Clear this bit.
[7]	-	UNP/SBZ.
[6]	ECC	Used to enable and disable Cycle Counter interrupt reporting: 0 = Disable interrupt, reset value 1 = Enable interrupt.
[5]	EC1	Used to enable and disable Count Register 1 interrupt reporting: 0 = Disable interrupt, reset value 1 = Enable interrupt.
[4]	EC0	Used to enable and disable Count Register 0 interrupt reporting: 0 = Disable interrupt, reset value 1 = Enable interrupt.
[3]	D	Cycle count divider: 0 = Counts every processor clock cycle, reset value 1 = Counts every 64th processor clock cycle.

Table 3-136 Performance Monitor Control Register bit functions (continued)

Bits	Field name	Function
[2]	C	Cycle Counter Register Reset. Reset on write, Unpredictable on read: 0 = No action, reset value 1 = Reset the Cycle Counter Register to `0x0`.
[1]	P	Count Register 1 and Count Register 0 Reset. Reset on write, Unpredictable on read: 0 = No action, reset value 1 = Reset both Count Registers to `0x0`.
[0]	E	Enable all counters: 0 = All counters disabled, reset value 1 = All counters enabled.

The Performance Monitor Control Register:

- controls the events that Count Register 0 and Count Register 1 count
- indicates the counter that overflowed
- enables and disables the report of interrupts
- extends Cycle Count Register counting by six more bits, cycles between counter rollover = 2^{38}
- resets all counters to zero
- enables the entire performance monitoring mechanism.

Table 3-137 lists the events that can be monitored using the Performance Monitor Control Register.

Table 3-137 Performance monitoring events

EVNTBUS bit position	Event number	Event definition
-	`0xFF`	An increment each cycle.
-	`0x26`	Procedure return instruction executed and return address predicted incorrectly. The procedure return address was restored to the return stack following the prediction being identified as incorrect.
-	`0x25`	Procedure return instruction executed and return address predicted. The procedure return address was popped off the return stack and the core branched to this address.
-	`0x24`	Procedure return instruction executed. The procedure return address was popped off the return stack.
-	`0x23`	Procedure call instruction executed. The procedure return address was pushed on to the return stack.
-	`0x22`	If both **ETMEXTOUT[0]** and **ETMEXTOUT[1]** signals are asserted then the count is incremented by two. If either signal is asserted then the count increments by one.
-	`0x21`	**ETMEXTOUT[1]** signal was asserted for a cycle.
-	`0x20`	**ETMEXTOUT[0]** signal was asserted for a cycle.
[19]	`0x12`	Write Buffer drained because of a Data Synchronization Barrier operation or Strongly Ordered operation.

Table 3-137 Performance monitoring events (continued)

EVNTBUS bit position	Event number	Event definition
[18]	0x11	Stall because of a full Load Store Unit request queue. This event takes place each clock cycle when the condition is met. A high incidence of this event indicates the LSU is often waiting for transactions to complete on the external bus.
[17]	0x10	Explicit external data accesses, Data Cache linefills, Noncacheable, write-through.
[16]	0xF	Main TLB miss.
[15:14]	0xD	Software changed the PC. This event occurs any time the PC is changed by software and there is not a mode change. For example, a MOV instruction with PC as the destination triggers this event. Executing an SVC from User mode does not trigger this event, because it incurs a mode change. If **EVENTBUS** bit [15] is HIGH, two software PC changes occurred in this clock cycle and the count increments by two.
[13]	0xC	Data cache write-back. This event occurs once for each half line of four words that are written back from the cache.
[12]	0xB	Data cache miss. Does not include Cache Operations.
[11]	0xA	Data cache access. Does not include Cache Operations. This event occurs for each nonsequential access to a cache line, regardless of whether or not the location is cacheable.
[10]	0x9	Data cache access. Does not include Cache Operations. This event occurs for each nonsequential access to a cache line, for cacheable locations.
[9:8]	0x7	Instruction executed. If **EVENTBUS** bit [9] is HIGH, two instructions were executed in this clock cycle and the count is increments by two.
[7]	0x6	Branch mispredicted.
[6]	-	Reserved.
[5]	0x5	Branch instruction executed, branch might or might not have changed program flow.
[4]	0x4	Data MicroTLB miss.
[3]	0x3	Instruction MicroTLB miss.
[2]	0x2	Stall because of a data dependency. This event occurs every cycle when the condition is present.
[1]	0x1	Stall because instruction buffer cannot deliver an instruction. This can indicate an Instruction Cache miss or an Instruction MicroTLB miss. This event occurs every cycle when the condition is present.
[0]	0x0	Instruction cache miss. **Note** This event counts all instruction cache misses, including any speculative access that would be a cache miss. If the instruction that caused a speculative access is not executed then there might not be a fetch from external memory. This can happen, for example, if the code branches round the instruction. This means that the value returned in this counter can be much larger than the number of external memory accesses caused by instruction cache misses.
-	All other values	Reserved. Unpredictable behavior.

Access to the Performance Monitor Control Register in User mode depends on the V bit, see *c15, Secure User and Non-secure Access Validation Control Register* on page 3-132. The Performance Monitor Control Register is always accessible in Privileged modes. Table 3-138 lists the results of attempted access for each mode.

Table 3-138 Results of access to the Performance Monitor Control Register

V bit	Secure Privileged		Non-secure Privileged		User	
	Read	Write	Read	Write	Read	Write
0	Data	Data	Data	Data	Undefined exception	Undefined exception
1	Data	Data	Data	Data	Data	Data

To access the Performance Monitor Control Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c15
- CRm set to c12
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c15, c12, 0    ; Read Performance Monitor Control Register
MCR p15, 0, <Rd>, c15, c12, 0    ; Write Performance Monitor Control Register
```

If this unit generates an interrupt, the processor asserts the pin **nPMUIRQ**. You can route this pin to an external interrupt controller for prioritization and masking. This is the only mechanism that signals this interrupt to the core. When asserted, this interrupt can only be cleared if bit 0 of the Performance Monitor Control Register is high.

There is a delay of three cycles between an enable of the counter and the start of the event counter. The information used to count events is taken from various pipeline stages. This means that the absolute counts recorded might vary because of pipeline effects. This has negligible effect except in cases where the counters are enabled for a very short time.

In addition to the two counters within the processor, most of the events that Table 3-137 on page 3-135 lists are available on an external bus, **EVNTBUS**. You can connect this bus to the ETM unit or other external trace hardware to enable the events to be monitored. If you do not want this functionality, set the X bit in the Performance Monitor Control Register to 0. In Debug state, the **EVNTBUS** is masked to zero.

3.2.52 c15, Cycle Counter Register

The purpose of the Cycle Counter Register is to count the core clock cycles.

The Cycle Counter Register:

- is in CP15 c15
- is a 32-bit read/write register common to Secure and Non-secure worlds
- counts up and can trigger an interrupt on overflow.

The Cycle Counter Register bits[31:0] contain the count value. The reset value is 0.

You can use this register in conjunction with the Performance Monitor Control Register and the two Counter Registers to provide a variety of useful metrics that enable you to optimize system performance.

Access to the Cycle Counter Register in User mode depends on the V bit, see *c15, Secure User and Non-secure Access Validation Control Register* on page 3-132. The Cycle Counter Register is always accessible in Privileged modes. Table 3-139 lists the results of attempted access for each mode.

Table 3-139 Results of access to the Cycle Counter Register

V bit	Secure Privileged		Non-secure Privileged		User	
	Read	Write	Read	Write	Read	Write
0	Data	Data	Data	Data	Undefined exception	Undefined exception
1	Data	Data	Data	Data	Data	Data

To access the Cycle Counter Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c15
- CRm set to c12
- Opcode_2 set to 1.

For example:

```
MRC p15, 0, <Rd>, c15, c12, 1    ; Read Cycle Counter Register
MCR p15, 0, <Rd>, c15, c12, 1    ; Write Cycle Counter Register
```

The value in the Cycle Counter Register is zero at Reset.

You can use the Performance Monitor Control Register to set the Cycle Counter Register to zero.

You can use the Performance Monitor Control Register to configure the Cycle Counter Register to count every 64th clock cycle.

3.2.53 c15, Count Register 0

The purpose of the Count Register 0 is to count instances of an event that the Performance Monitor Control Register selects.

The Count Register 0:

- is in CP15 c15
- is a 32-bit read/write register common to Secure and Non-secure worlds
- counts up and can trigger an interrupt on overflow.

Count Register 0 bits [31:0] contain the count value. The reset value is 0.

You can use this register in conjunction with the Performance Monitor Control Register, the Cycle Count Register, and Count Register 1 to provide a variety of useful metrics that enable you to optimize system performance.

Note

- In Debug state the counter is disabled.
- When the core is in a mode where noninvasive debug is not permitted, set by **SPNIDEN** and the SUNIDEN bit, see *c1, Secure Debug Enable Register* on page 3-54, the processor does not count events.

Access to the Count Register 0 in User mode depends on the V bit, see *c15, Secure User and Non-secure Access Validation Control Register* on page 3-132. The Count Register 0 is always accessible in Privileged modes. Table 3-140 lists the results of attempted access for each mode.

Table 3-140 Results of access to the Count Register 0

V bit	Secure Privileged		Non-secure Privileged		User	
	Read	Write	Read	Write	Read	Write
0	Data	Data	Data	Data	Undefined exception	Undefined exception
1	Data	Data	Data	Data	Data	Data

To access Count Register 1 read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c15
- CRm set to c12
- Opcode_2 set to 2.

For Example:

```
MRC p15, 0, <Rd>, c15, c12, 2    ; Read Count Register 0
MCR p15, 0, <Rd>, c15, c12, 2    ; Write Count Register 0
```

The value in Count Register 0 is 0 at Reset.

You can use the Performance Monitor Control Register to set Count Register 0 to zero.

3.2.54 c15, Count Register 1

The purpose of the Count Register 1 is to count instances of an event that the Performance Monitor Control Register selects.

The Count Register 1:

- is in CP15 c15
- is a 32-bit read/write register common to Secure and Non-secure worlds
- counts up and can trigger an interrupt on overflow.

Count Register 1 bits [31:0] contain the count value. The reset value is 0.

You can use this register in conjunction with the Performance Monitor Control Register, the Cycle Count Register, and Count Register 0 to provide a variety of useful metrics that enable you to optimize system performance.

Note

- In Debug state the counter is disabled.
- When the core is in a mode where non-invasive debug is not permitted, set by **SPNIDEN** and the SUNIDEN bit, see *c1, Secure Debug Enable Register* on page 3-54, the processor does not count events.

Access to the Count Register 1 in User mode depends on the V bit, see *c15, Secure User and Non-secure Access Validation Control Register* on page 3-132. The Count Register 1 is always accessible in Privileged modes. Table 3-141 lists the results of attempted access for each mode.

Table 3-141 Results of access to the Count Register 1

V bit	Secure Privileged		Non-secure Privileged		User	
	Read	Write	Read	Write	Read	Write
0	Data	Data	Data	Data	Undefined exception	Undefined exception
1	Data	Data	Data	Data	Data	Data

To access Count Register 1 read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c15
- CRm set to c12
- Opcode_2 set to 3.

For example:

```
MRC p15, 0, <Rd>, c15, c12, 3    ; Read Count Register 1
MCR p15, 0, <Rd>, c15, c12, 3    ; Write Count Register 1
```

The value in Count Register 1 is 0 at Reset.

You can use the Performance Monitor Control Register to set Count Register 1 to zero.

3.2.55 c15, System Validation Counter Register

The purpose of the System Validation Counter Register is to count core clock cycles to trigger a system validation event.

The System Validation Counter Register is:

- in CP15 c15
- a 32 bit read/write register common to the Secure and Non-secure worlds
- accessible in User and Privileged modes.

The System Validation Counter Register consists of one 32-bit register that performs four functions. Table 3-142 lists the arrangement of the functions in this group. The reset value is 0.

Table 3-142 System validation counter register operations

CRn	Opcode_1	CRm	Opcode_2	R/W	Operation
c15	0	c12	1	R/W	Reset counter
			2	R/W	Interrupt counter
			3	R/W	Fast interrupt counter
			7	W	External debug request counter

The reset, interrupt, and fast interrupt counters are 32-bits wide. The external debug request counter is 6 bits wide. Figure 3-74 on page 3-141 shows the arrangement of bits for the external debug request counter.

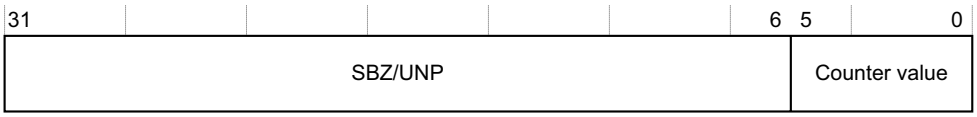

Figure 3-74 System Validation Counter Register format for external debug request counter

Table 3-143 lists the results of attempted access for each mode. Access in Secure User mode and in the Non-secure world depends on the V bit, see *c15, Secure User and Non-secure Access Validation Control Register* on page 3-132.

Table 3-143 Results of access to the System Validation Counter Register

Function	V bit	Secure Privileged		Non-secure Privileged		User	
		Read	Write	Read	Write	Read	Write
Reset, interrupt, and fast interrupt counters	0	Data	Data	Undefined exception	Undefined exception	Undefined exception	Undefined exception
	1	Data	Data	Data	Data	Data	Data
External debug request counter	0	Unpredictable	Data	Undefined exception	Undefined exception	Undefined exception	Undefined exception
	1	Unpredictable	Data	Unpredictable	Data	Unpredictable	Data

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

To use the System Validation Counter Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c15
- CRm set to c12
- Opcode_2 set to:
  - 1, Read/write reset counter
  - 2, Read/write interrupt counter
  - 3, Read/write fast interrupt counter
  - 7, Write external debug request counter.

For example:

```
MRC p15, 0, <Rd>, c15, c12, 1    ;Read reset counter
MCR p15, 0, <Rd>, c15, c12, 1    ;Write reset counter
MRC p15, 0, <Rd>, c15, c12, 2    ;Read interrupt counter
MCR p15, 0, <Rd>, c15, c12, 2    ;Write interrupt counter
MRC p15, 0, <Rd>, c15, c12, 3    ;Read fast interrupt counter
MCR p15, 0, <Rd>, c15, c12, 3    ;Write fast interrupt counter
MCR p15, 0, <Rd>, c15, c12, 7    ;Write external debug request counter
```

A read or write to the System Validation Counter Register with a value of Opcode_2 other than 1, 2, 3, or 7 has no effect.

When the system starts the counters they count up, incrementing by one on each core clock cycle, until they wrap around. When the counters wrap around they cause the specified event to occur. See *c15, System Validation Operations Register* on page 3-142.

The reset, interrupt, and fast interrupt counters reuse the Cycle Count Register, Count Register 0 and Count Register 1 of the System performance monitor registers respectively, see *System performance monitor* on page 3-10. You must not use the System Validation Count Register when the System Performance Monitor Registers are in use.

The reset, interrupt, and fast interrupt counters are read/write. The external debug request counter is write only. Attempts to read the external debug request counter return `0x00000000` regardless or the actual value of the counter.

3.2.56 c15, System Validation Operations Register

The purpose of the System Validation Operations Register is to start and stop system validation counters to trigger a system validation event.

The System Validation Operations Register is:

- in CP15 c15
- a 32 bit read/write register common to the Secure and Non-secure worlds
- accessible in user and privileged modes.

The System Validation Operations Register consists of one 32-bit register that performs 16 functions. Table 3-144 lists the arrangement of the functions in this group.

Table 3-144 System Validation Operations Register functions

CRn	Opcode_1	CRm	Opcode_2	R/W	Operation
c15	0	c13	1	W	Start reset counter
			2	W	Start interrupt counter
			3	W	Start reset and interrupt counters
			4	W	Start fast interrupt counter
			5	W	Start reset and fast interrupt counters
			6	W	Start interrupt and fast interrupt counters
			7	W	Start reset, interrupt and fast interrupt counters
c15	1	c13	0-7	W	Start external debug request counter
c15	2	c13	1	W	Stop reset counter
			2	W	Stop interrupt counter
			3	W	Stop reset and interrupt counters
			4	W	Stop fast interrupt counter
			5	W	Stop reset and fast interrupt counters
			6	W	Stop interrupt and fast interrupt counters
			7	W	Stop reset, interrupt and fast interrupt counters
c15	3	c13	0-7	W	Stop external debug request counter

A write to the System Validation Operations Register with a combination of Opcode_1 and Opcode_2 that Table 3-144 does not list has no effect. A read from the System Validation Operations Register returns `0x00000000`.

The reset value of this register is 0.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Table 3-145 lists the results of attempted access for each mode. Access in Secure User mode and in the Non-secure world depends on the V bit, see *c15, Secure User and Non-secure Access Validation Control Register* on page 3-132.

Table 3-145 Results of access to the System Validation Operations Register

V bit	Secure Privileged		Non-secure Privileged		User	
	Read	Write	Read	Write	Read	Write
0	Unpredictable	Data	Undefined exception	Undefined exception	Undefined exception	Undefined exception
1	Unpredictable	Data	Unpredictable	Data	Unpredictable	Data

To use the System Validation Operations Register write CP15 with <Rd> set to SBZ and:

- Opcode_1 set to:
  - 0, Start reset, interrupt, or fast interrupt counters
  - 1, Start external debug request counter
  - 2, Stop reset, interrupt, or fast interrupt counters
  - 3, Stop external debug request counter.
- CRn set to c15
- CRm set to c13
- Opcode_2 set to:
  - 1, Reset counter
  - 2, Interrupt counter
  - 3, Reset and interrupt counters
  - 4, Fast interrupt counter
  - 5, Reset and fast interrupt counters
  - 6, Interrupt and fast interrupt counters
  - 7, Reset, interrupt and fast interrupt counters
  - Any value, External debug request counter.

For example:

```
MCR p15, 0, <Rd>, c15, c13, 1    ; Start reset counter
MCR p15, 0, <Rd>, c15, c13, 2    ; Start interrupt counter
MCR p15, 0, <Rd>, c15, c13, 3    ; Start reset and interrupt counters
MCR p15, 0, <Rd>, c15, c13, 4    ; Start fast interrupt counter
MCR p15, 0, <Rd>, c15, c13, 5    ; Start reset and fast interrupt counters
MCR p15, 0, <Rd>, c15, c13, 6    ; Start interrupt and fast interrupt counters
MCR p15, 0, <Rd>, c15, c13, 7    ; Start reset, interrupt and fast interrupt counters
MCR p15, 1, <Rd>, c15, c13, 0    ; Start external debug request counter
MCR p15, 2, <Rd>, c15, c13, 1    ; Stop reset counter
MCR p15, 2, <Rd>, c15, c13, 2    ; Stop interrupt counter
MCR p15, 2, <Rd>, c15, c13, 3    ; Stop reset and interrupt counters
MCR p15, 2, <Rd>, c15, c13, 4    ; Stop fast interrupt counter
MCR p15, 2, <Rd>, c15, c13, 5    ; Stop reset and fast interrupt counters
MCR p15, 2, <Rd>, c15, c13, 6    ; Stop interrupt and fast interrupt counters
MCR p15, 2, <Rd>, c15, c13, 7    ; Stop reset, interrupt and fast interrupt counters
MCR p15, 3, <Rd>, c15, c13, 0    ; Stop external debug request counter
```

You use the System Validation Operations Register to start and stop the reset, interrupt, fast interrupt, and external debug request counters. When the system starts any of these counters, they count up incrementing by one every core clock cycle, until they wrap around. When the counters wrap around they cause **nVALRESET**, **nVALIRQ**, **nVALFIQ**, or **VALEDBGRQ** to go LOW depending on the operation. You can use these outputs to generate system Reset, Interrupt request, Fast Interrupt request, or External Debug Request events. You can use the System Validation Counter Register to set the start value of the counters, see *c15, System Validation Counter Register* on page 3-140. Any number of events can occur simultaneously.

When you use the Validation Trickbox Operations Register to start a counter, there is one clock cycle delay, that generally corresponds to one instruction, before the count begins. If you require an event to occur on the next instruction, insert a NOP instruction between the MCR instruction, to the System Validation Operations Register, that starts the counter and the instruction on which you want the event to occur.

You must leave two clock cycles, that generally corresponds to two instructions, between a write to a counter with the System Validation Counter Register and the start of that count with the System Validation Operations Register.

After the system stops the reset, interrupt or fast interrupt counters, or after handling the events they cause, you must explicitly clear the counters to return them to their System performance monitoring function. To do this set bits in <Rn> and write to the Performance Monitor Control Register to clear the relevant overflow flags:

- bit [10] to clear the reset counter
- bit [9] to clear the fast interrupt counter
- bit [8] to clear the interrupt counter.

You must carry out this operation with a read-modify-write sequence to avoid changes to other bits, see *c15, Performance Monitor Control Register* on page 3-133. You do not have to clear the external debug request counter explicitly in this way because it is not used for system performance monitoring.

The reset, interrupt, and fast interrupt counters reuse the Cycle Count Register, Count Register 0 and Count Register 1 of the System performance monitor registers respectively, see *System performance monitor* on page 3-10. As a result you must not perform read or write operations to the System Validation Counter Register when the System performance monitor registers are in use.

The System Validation Operations Register is write only and attempts to read this register are reserved and return `0x00000000`.

To schedule system validation events follow this procedure:

1. Modify the Secure User and Non-secure Access Validation Control Register to permit access from User or Non-secure modes if this is required.
2. Use the Validation Trickbox Counter Register to load the required counter with `0xFFFFFFFF` minus the number of core clock cycles to wait before the event occurs.
3. Use the Validation Trickbox Operations Register to start the required counter.
4. Use the appropriate Validation Trickbox Operations Register to stop the required counter, after the event has occurred or as necessary.
5. Use the Performance Monitor Control Register to reset the counters and return them to System performance monitoring functionality.

3.2.57 c15, System Validation Cache Size Mask Register

The purpose of the System Validation Cache Size Mask Register is to change the apparent size of the caches and TCMs as they appear to the processor, for validation by simulation. It does not change the physical size of the caches and TCMs in a manufactured device.

The System Validation Cache Size Mask Register is:

- in CP15 c15
- a 32 bit read/write register common to the Secure and Non-secure worlds
- accessible in User and Privileged modes.

Figure 3-75 shows the arrangement of bits for the System Validation Cache Size Mask Register.

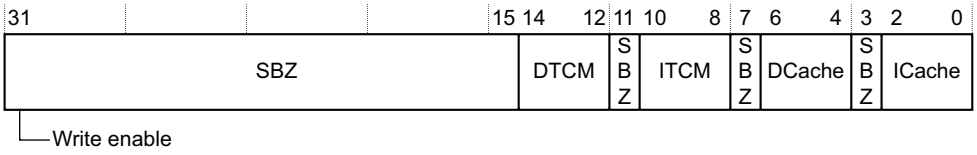

Figure 3-75 System Validation Cache Size Mask Register format

Table 3-146 lists how the bit values correspond with the System Validation Cache Size Mask Register functions.

Table 3-146 System Validation Cache Size Mask Register bit functions

Bits	Field name	Function
[31]	Write enable	Enables the update of the Cache and TCM sizes: 0 = The Cache and TCM sizes are not changed, reset value. 1 = The Cache and TCM sizes take the new values that the DTCM, ITCM, DCache and ICache fields of this register specify. **Note** This is bit is write access only and Read As Zero.
[30:15]	SBZ	UNP/SBZ.
[14:12]	DTCM	Specifies apparent size of Data TCM and apparent number of Data TCM banks, as it appears to the processor. All other values are reserved: b000 = Not present b011 = 1 bank, 4KB b100 = 2 banks, 4KB each b101 = 2 banks, 8KB each b110 = 2 banks, 16KB each b111 = 2 banks, 32KB each.
[11]	SBZ	UNP/SBZ.
[10:8]	ITCM	Specifies apparent size of Instruction TCM and apparent number of Instruction TCM banks, as it appears to the processor. All other values are reserved: b000 = Not present b011 = 1 bank, 4KB b100 = 2 banks, 4KB each b101 = 2 banks, 8KB each b110 = 2 banks, 16KB each b111 = 2 banks, 32KB each.

Table 3-146 System Validation Cache Size Mask Register bit functions (continued)

Bits	Field name	Function
[7]	SBZ	UNP/SBZ.
[6:4]	DCache	Specifies apparent size of Data Cache, as it appears to the processor. All other values are reserved: b011 = 4KB b100 = 8KB b101 = 16KB b110 = 32KB b111 = 64KB.
[3]	SBZ	UNP/SBZ.
[2:0]	ICache	Specifies apparent size of Instruction Cache, as it appears to the processor. All other values are reserved: b011 = 4KB b100 = 8KB b101 = 16KB b110 = 32KB b111 = 64KB.

At reset, the values in the System Validation Cache Size Mask Register are the correct values for the implemented caches and TCMs.

Access to the System Validation Cache Size Mask Register in Secure User mode and in the Non-secure world depends on the V bit, see *c15, Secure User and Non-secure Access Validation Control Register* on page 3-132. Table 3-147 lists the results of attempted access for each mode.

Table 3-147 Results of access to the System Validation Cache Size Mask Register

V bit	Secure Privileged		Non-secure Privileged		User	
	Read	Write	Read	Write	Read	Write
0	Data	Data	Undefined exception	Undefined exception	Undefined exception	Undefined exception
1	Data	Data	Data	Data	Data	Data

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

To use the System Validation Cache Size Mask Register read or write CP15 with:

- Opcode_1 set to 0
- CRn set to c15
- CRm set to c14
- Opcode_2 set to 0.

For example:

```
MRC p15, 0, <Rd>, c15, c14, 0    ; Read System Validation Cache Size Mask Register
MCR p15, 0, <Rd>, c15, c14, 0    ; Write System Validation Cache Size Mask Register
```

You can use the System Validation Cache Size Mask Register, in a validation simulation environment, to perform validation with cache and TCM sizes that appear to be a different size from those that are actually implemented. The validation environment for the processor contains validation RAMs that support cache and TCM size masking using this register. When you write to the System Validation Cache Size Mask Register, the processor behaves as though the caches and TCMs are the sizes that are written to the register. The sizes written to the register are reflected in:

- The sizes of the cache and TCM RAMs.
- The sizes of the caches in the Cache Type Register, see *c0, Cache Type Register* on page 3-21, the number of Instruction and Data TCM banks in the TCM Status Register, see *c0, TCM Status Register* on page 3-24, the sizes of the TCMs in the Instruction TCM Region Register, see *c9, Instruction TCM Region Register* on page 3-91, and the Data TCM Region Register, see *c9, Data TCM Region Register* on page 3-89.
- The number and use of cache master valid bits, see *Cache Master Valid Registers* on page 3-8.
- The hazard detection logic that prevents the same line being allocated twice into the caches.
- The DMA. If the TCMs are both masked as not present, then the DMA also appears not to be present.

Note

You must not modify the System Validation Cache Size Mask Register in a manufactured device. Physical RAMs do not support cache and TCM size masking. Therefore, any attempt to mask cache and TCM sizes using this register causes address aliasing effects and problems with cache master valid bits, that result in incorrect operation and Unpredictable effects.

3.2.58 c15, Instruction Cache Master Valid Register

The purpose of the Instruction Cache Master Valid Register is to save and restore the instruction cache master valid bits on entry to and exit from dormant mode, see *Dormant mode* on page 10-4. You might also use this register during debug.

The Instruction Cache Master Valid Register is:

- in CP15 c15
- a 32-bit read/write register in Secure world only
- accessible in privileged modes only.

The number of Master Valid bits in the register is a function of the cache size. There is one Master Valid bit for each 8 cache lines:

$$\text{Master Valid bits} = \frac{\text{cache size}}{\text{line length in bytes} \times 8}$$

For instance, there are 64 Master Valid bits for a 16KB cache. You can access Master Valid bits through 32-bit registers indexed using Opcode_2. The maximum number of 32-bit registers required for the largest cache size, 64KB, is 8. The Master Valid bits fill the registers from the LSB of the lowest numbered register upwards.

Writes to unimplemented Valid bits have no effect, and reads return 0. The reset value is 0.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Attempts to access the register in modes other than Secure privileged result in an Undefined exception.

To use the Instruction Cache Master Valid Register write CP15 with:

- Opcode_1 set to 3
- CRn set to c15
- CRm set to c8
- Opcode_2 set to <Register Number>.

```
MRC p15, 3, <Rd>, c15, c8, <Register Number>    ; Read Instruction Cache Master Valid Register
MCR p15, 3, <Rd>, c15, c8, <Register Number>    ; Write Instruction Cache Master Valid Register
```

The <Register Number> field of the instruction designates one of the registers required to capture all the Valid bits. The highest Register Number is one less than the number of times 8KB divides into the cache size.

3.2.59 c15, Data Cache Master Valid Register

The purpose of the Data Cache Master Valid Register is to save and restore the Data cache master valid bits on entry to and exit from dormant mode, see *Dormant mode* on page 10-4. You might also use this register during debug.

The Data Cache Master Valid Register is:

- in CP15 c15
- a 32-bit read/write register in the Secure world only
- accessible in privileged modes only.

The number of Master Valid bits in the register is a function of the cache size. There is one Master Valid bit for each 8 cache lines:

$$\text{Master Valid bits} = \frac{\text{cache size}}{\text{line length in bytes x 8}}$$

For instance, there are 64 Master Valid bits for a 16KB cache. You can access Master Valid bits through 32-bit registers indexed using Opcode_2. The maximum number of 32-bit registers required for the largest cache size, 64KB, is 8. The Master Valid bits fill the registers from the LSB of the lowest numbered register upwards.

Writes to unimplemented Valid bits have no effect, and reads return 0. The reset value is 0.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Attempts to access the register in modes other than Secure privileged result in an Undefined exception.

To use the Data Cache Master Valid Register write CP15 with:

- Opcode_1 set to 3
- CRn set to c15
- CRm set to c12
- Opcode_2 set to <Register Number>.

```
MRC p15, 3, <Rd>, c15, c12, <Register Number>    ; Read Data Cache Master Valid Register
MCR p15, 3, <Rd>, c15, c12, <Register Number>    ; Write Data Cache Master Valid Register
```

The <Register Number> field of the instruction designates one of the registers required to capture all the Valid bits. The highest Register Number is one less than the number of times 8KB divides into the cache size.

3.2.60 c15, TLB lockdown access registers

The purpose of the TLB lockdown access registers is to provide read and write access to the contents of the lockdown region of the TLB. The processor requires these registers to enable it to save state before it enters Dormant mode, see *Dormant mode* on page 10-4. You might also use this register for debug.

The TLB lockdown access registers are:

- in CP15 c15
- four 32-bit read/write registers in the Secure world only:
  - TLB Lockdown Index Register
  - TLB Lockdown VA Register
  - TLB Lockdown PA Register
  - TLB Lockdown Attributes Register.
- accessible in privileged modes only.

The four registers have different bit arrangements and functions. Figure 3-76 shows the arrangement of bits in the TLB Lockdown Index Register.

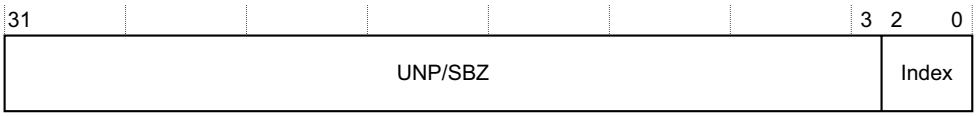

Figure 3-76 TLB Lockdown Index Register format

Table 3-148 lists how the bit values correspond with the TLB Lockdown Index Register functions.

Table 3-148 TLB Lockdown Index Register bit functions

Bits	Field name	Function
[31:3]	-	UNP/SBZ.
[2:0]	Index	Selects the lockdown entry of the eight TLB lockdown entries to read or write when accessing other TLB lockdown access registers. Select lockdown entry 0 to 7.

Figure 3-77 shows the arrangement of bits in the TLB Lockdown VA Register.

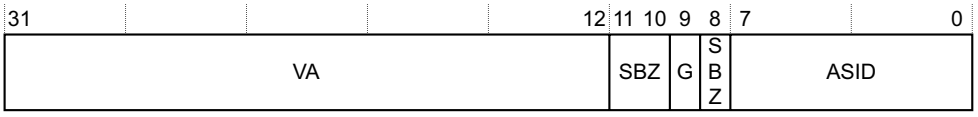

Figure 3-77 TLB Lockdown VA Register format

Table 3-149 lists how the bit values correspond with the TLB Lockdown VA Register functions.

Table 3-149 TLB Lockdown VA Register bit functions

Bits	Field name	Function
[31:12]	VA	Holds the VA of this page table entry.
[11:10]	-	UNP/SBZ.
[9]	G	Defines if this page table entry is global, applies to all ASIDs, or application-specific, ASID must match on lookups: 0 = Application-specific entry 1 = Global entry.
[8]	-	UNP/SBZ.
[7:0]	ASID	Holds the ASID for application-specific page table entries. For global entries, this field Should Be Zero.

Figure 3-78 shows the arrangement of bits in the TLB Lockdown PA Register.

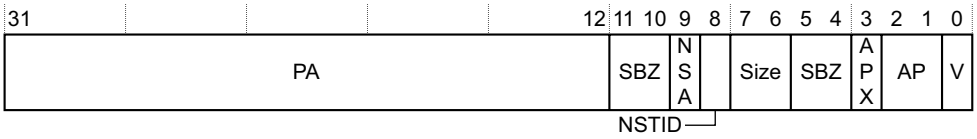

Figure 3-78 TLB Lockdown PA Register format

Table 3-150 lists how the bit values correspond with the TLB Lockdown PA Register functions.

Table 3-150 TLB Lockdown PA Register bit functions

Bits	Field name	Function
[31:12]	PA	Holds the PA of this page table entry.
[11:10]	-	UNP/SBZ.
[9]	NSA	Defines whether memory accesses in the memory region that this page table entry describes are Secure or Non-secure accesses. This matches the Secure or Non-secure state of the memory being accessed. If the NSTID bit is set, the NSA bit is also set regardless of the written value. This ensures that Non-secure page table entries can only access Non-secure memory, but Secure page table entries can access Secure or Non-secure memory: 0 = Memory accesses are Secure 1 = Memory accesses are Non-secure.
[8]	NSTID	Defines page table entry as Secure or Non-secure: 0 = Entry is Secure 1 = Entry is Non-secure.
[7:6]	Size	Defines the size of the memory region that this page table entry describes: b00 = 16MB supersection b01 = 4KB page b10 = 64KB page b11 = 1M section.
[5:4]	-	UNP/SBZ.

Table 3-150 TLB Lockdown PA Register bit functions (continued)

Bits	Field name	Function
[3]	APX	Access permissions extension bit. Defines the access permissions for this page table entry. See Table 3-151.
[2:1]	AP	Access permissions, or first sub-page access permissions if the page table entry supports sub-pages.
[0]	V	Indicates if this page table entry is valid: 0 = Entry is not valid 1 = Entry is valid.

Table 3-151 lists the encoding for the access permissions for bit fields APX and AP.

Table 3-151 Access permissions APX and AP bit fields encoding

APX	AP	Supervisor permissions	User permissions	Access type
0	b00	No access	No access	All accesses generate a permission fault
0	b01	Read/write	No access	Supervisor access only
0	b10	Read/write	Read only	Writes in user mode generate permission faults
0	b11	Read/write	Read/write	Full access
1	b00	No access	No access	Domain fault encoded field
1	b01	Read only	No access	Supervisor read only
1	b10	Read only	Read only	Supervisor/User read only
1	b11	Read only	Read only	Supervisor/User read only

Figure 3-79 shows the arrangement of bits in the TLB Lockdown Attributes Register.

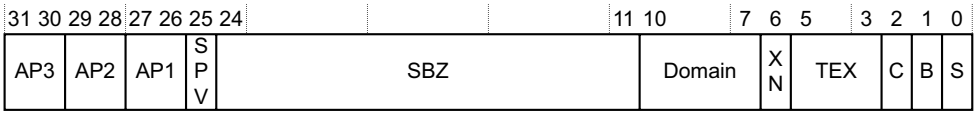

Figure 3-79 TLB Lockdown Attributes Register format

Table 3-152 lists how the bit values correspond with the TLB Lockdown Attributes Register functions.

Table 3-152 TLB Lockdown Attributes Register bit functions

Bits	Field name	Function
[31:30]	AP3	Sub-page access permissions for the fourth sub-page. If the page table entry does not support sub-pages this field Should Be Zero.
[29:28]	AP2	Sub-page access permissions for the third sub-page. If the page table entry does not support sub-pages this field Should Be Zero.
[27:26]	AP1	Sub-page access permissions for the second sub-page. If the page table entry does not support sub-pages this field Should Be Zero.

Table 3-152 TLB Lockdown Attributes Register bit functions (continued)

Bits	Field name	Function
[25]	SPV	Indicates that this page table entry supports sub-pages. Page table entries that support sub-pages must be marked as Global, see *c15, TLB lockdown access registers* on page 3-149: 0 = Sub-pages are not valid 1 = Sub-pages are valid.
[24:11]	SBZ	UNP/SBZ.
[10:7]	Domain	Specifies the Domain number for the page table entry.
[6]	XN	Specifies Execute Never attribute: when set, the contents of the memory region that this page table entry describes cannot be executed as code. An attempt to execute an instruction in this region results in a permission fault: 0 = Can execute 1 = Cannot execute.
[5:3]	TEX	TEX[2:0] bits. Describes the memory region attributes. See *Memory region attributes* on page 6-14.
[2]	C	C bit. Describes the memory region attributes. See *Memory region attributes* on page 6-14.
[1]	B	B bit. Describes the memory region attributes. See *Memory region attributes* on page 6-14.
[0]	S	Indicates if the memory region that this page table entry describes is shareable: 0 = Region is not shared 1 = Region is shared.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Table 3-153 lists the results of attempted access for each mode.

Table 3-153 Results of access to the TLB lockdown access registers

Secure Privileged		Non-secure Privileged		User
Read	Write	Read	Write	
Data	Data	Undefined exception	Undefined exception	Undefined exception

To read or write a TLB Lockdown entry, you must use this procedure:

1. Write TLB Lockdown Index Register to select the required TLB Lockdown entry.
2. Read or write TLB Lockdown VA Register.
3. Read or write TLB Lockdown Attributes Register.
4. Read or write TLB Lockdown PA Register. For writes, this sets the valid bit, enabling the complete new entry to be used.

This procedure must not be interruptible, so your code must disable interrupts before it accesses the TLB lockdown access registers.

Note

Software must avoid the creation of inconsistencies between the main TLB entries and the entries already loaded in the micro-TLBs.

To use the TLB lockdown access registers read or write CP15 with:

- Opcode_1 set to 5
- CRn set to c15
- CRm set to:
  - — c4, TLB Lockdown Index Register
  - — c5, TLB Lockdown VA Register
  - — c6, TLB Lockdown PA Register
  - — c7, TLB Lockdown Attributes Register.

  Opcode_2 set to 2.

For example:

```
MRC p15, 5, <Rd>, c15, c4, 2    ; Read TLB Lockdown Index Register
MCR p15, 5, <Rd>, c15, c4, 2    ; Write TLB Lockdown Index Register
MRC p15, 5, <Rd>, c15, c5, 2    ; Read TLB Lockdown VA Register
MCR p15, 5, <Rd>, c15, c5, 2    ; Write TLB Lockdown VA Register
MRC p15, 5, <Rd>, c15, c6, 2    ; Read TLB Lockdown PA Register
MCR p15, 5, <Rd>, c15, c6, 2    ; Write TLB Lockdown PA Register
MRC p15, 5, <Rd>, c15, c7, 2    ; Read TLB Lockdown Attributes Register
MCR p15, 5, <Rd>, c15, c7, 2    ; Write TLB Lockdown Attributes Register
```

Example 3-3 is a code sequence that stores all 8 TLB Lockdown entries to memory, and later restores them to the TLB Lockdown region. You might use sequences similar to this for entry into Dormant mode.

Example 3-3 Save and restore all TLB Lockdown entries

```
                ADR     r1,TLBLockAddr          ; Set r1 to save address
                MOV     R0,#0                   ; Initialize counter
                CPSID   aif                     ; Disable interrupts
TLBLockSave     MCR     p15,5,R0,c15,c4,2       ; Set TLB Lockdown Index
                MRC     p15,5,R2,c15,c5,2       ; Read TLB Lockdown VA
                MRC     p15,5,R3,c15,c7,2       ; Read TLB Lockdown Attrs
                MRC     p15,5,R4,c15,c6,2       ; Read TLB Lockdown PA
                STMIA   r1!,{R2-R4}             ; Save TLB Lockdown entry
                ADD     R0,R0,#1                ; Increment counter
                CMP     R0,#8                   ; Saved all 8 entries?
                BNE     TLBLockSave             ; Loop until all saved
                CPSIE   aif                     ; Re-enable interrupts

; insert other code here

                ADR     r1,TLBLockAddr          ; Set r1 to save address
                MOV     R0,#0                   ; Initialize counter
                CPSID   aif                     ; Disable interrupts
TLBLockLoad     LDMIA   r1!,{R2-R4}             ; Load TLB Lockdown entry
                MCR     p15,5,R0,c15,c4,2       ; Set TLB Lockdown Index
                MCR     p15,5,R2,c15,c5,2       ; Write TLB Lockdown VA
                MCR     p15,5,R3,c15,c7,2       ; Write TLB Lockdown Attrs
                MCR     p15,5,R4,c15,c6,2       ; Write TLB Lockdown PA
                ADD     R0,R0,#1                ; Increment counter
                CMP     R0,#8                   ; Restored all 8 entries?
                BNE     TLBLockLoad             ; Loop until all restored
                CPSIE   aif                     ; Re-enable interrupts
```