Contents
Cortex-M4 Devices Generic User Guide

Routines (ISRs), dramatically reducing the interrupt latency. This is achieved through the hardware stacking of registers, and the ability to suspend load-multiple and store-multiple

Base Priority Mask Register

The BASEPRI register defines the minimum prio

2.2.3 Behavior of memory accesses

The behavior of accesses to each region in the memory map is:

The Code, SRAM, and external RAM regions can hold programs. However, ARM recommends that programs always use the Code region. This is because the processor has separate buses that enable instruction fetches and data accesses to occur simultaneously.

The optional MPU can override the default memory access behavior described in this section. For more information, see *Optional Memory Protection Unit* on page 4-37.

Additional memory acc

-

2.3 Exception model

This section describes the exception model. It describes:

- *Exception states*
- *Exception types*
- *Exception handlers* on page 2-23
- *Vector table*ge543
  - *Except*

Late-arriving This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the state saved is the same for both exceptions. Therefore the state saving continues uninterrupted. The processor can accept a late arriving exception until the first instruction of the exception handler of the original exception enters the execute stage of the processor. On return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply.

Exception entry

Exception entry occurs when there is a pending exception with sufficient priority and either:

Figure 2-3 Exception stack frame

Immediately after stacking, the stack pointer indicates the lowest address in the stack frame. The alignment of the stack frame is controlled via the STKALIGN bit of the *Configuration Control*

2.4 Fault handling

Faults are a subset of the exceptions, see *Exception model* on page 2-21. Faults are generated by:

- a bus error on:
  - an instruction fetch or vector table load
  - a data access.
- an internally-detected error such as an undefined instruction

2.5.2 Wakeup from sleep mode

The conditions for the processor to wakeup depend on the mechanism that cause it to enter sleep mode.

Wakeup from WFI or sleep-on-exit

See *Wait for event* on page 2-32 and the documentation supplied by your device vendor for more information about this signal.

2.5.5 Power management programming hints

ISO/IEC C cannot directly generate the

3.1 Instruction set summary

The processor implements a version of the Thumb instruction set. Table 3-1 lists the supported instructions.

Note

In Table 3-1:

3.3.5 Address alignment

An aligned access is an operation where a word-aligned address is used for a word, dual word, or multiple word access, or where a halfword-aligned address is used for a halfword access. Byte accesses are always aligned.

The Cortex-M4 processor supports unaligned access only for the following instructions:

•

3.4.1 ADR

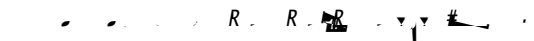
Generate PC-relative address.

Syntax

3.4.3 LDR and STR, register offset

Load and Store with register offset.

Syntax



where:

Is one of:

Load Register.

Store Register.

Is one of:

unsigned byte, zero extend to 32 bits on loads.

signed byte, sign extend to 32 bits (

3.4.8 LDREX and STREX

Restrictions

In these instructions:

- do not use PC
- do not use SP for R_d and R_m

3.4.9 CLREX

Clear Exclusive.

Restrictions

You can use SP and PC only in the

3.5.13 SSUB16 and SSUB8

Signed Subtract 16 and Signed Subtract 8.

3.6.2 UMULL, UMAAL, UMLAL

Unsigned Long Multiply, with optional Accumulate, using 32-bit operands and producing a

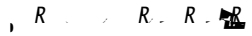
3.6.7 SMMLA and SMMLS

Signed Most Significant Word Multiply Accumulate and Signed Most Significant Word

3.6.8 SMMUL

Signed Most Significant Word Multiply.

Syntax



where:

nxt

Loading a 64-bit float

3.7.3 QADD and QSUB

Saturating Add and Saturating Subtract, signed.

Syntax

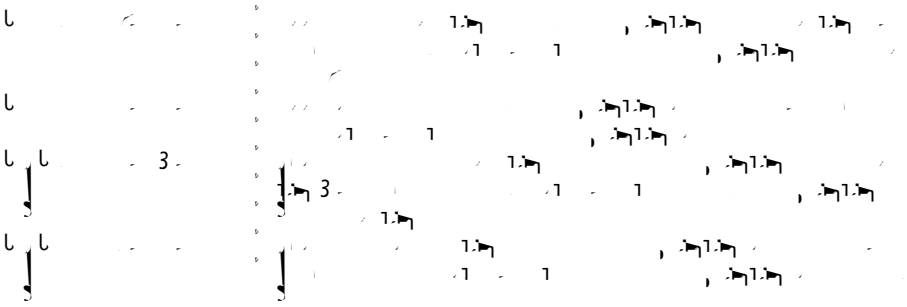
3.7.6 UQASX and UQSAX

3.7.7 UQADD and UQSUB

Condition flags

These instructions do not affect the condition code flags.

Examples



3.10.3 IT

If-Then condition instruction.

Syntax

where:

3.11.3 VCMP, VCMPE

Compares two floating-point registers, or

3.12.1 BKPT

Breakpoint.

Syntax

where:

3.12.11 WFE

Chapter 4
Cortex-M4 Peripherals

4.2 Nested Vectored Interrupt Controller

This section describes the NVIC and the registers it uses. The NVIC supports:

- An implementation-defined number of interrupts, in the range 1-240 interrupts.
- A programmable priority level of 0-255 for each interrupt. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.

4.2.3 Interrupt Clear-enable Registers

The NVIC_ICER0-NVIC_ICER7 registers disable interrupts, and show which interrupts are enabled. See the register summary in Table 4-2 on page 4-3 for the register attributes.

The bit assignments are:

4.2.4 Interrupt Set-pending R7.587/TT.002

4.2.5 Interrupt Clear-pending Registers

4.3.2 CPUID Base Register

4.3.8 System Handler Priority Registers

The SHPR1-SHPR3 registers set the priority level, 0 to 255, of the exception handlers that have configurable priority.

SHPR1-SHPR3 are byte accessible. See the register summary in Table 4-12 on page 4-11 for their attributes.

System Handler Priority Register 2

The bit assignments are:

System Handler Priority Register 3

The bit assignments are:

When an unaligned access faults, the address is the actual address that faulted. Because a single read or write instruction can be split into multiple aligned accesses, the fault address can be any address in the range of the requested access size.

Flags in the MMFSR indicate the cause of the fault, and whether the value in the MMFAR is valid. See *MemManage Fault Status Register* on page 4-25.

4.5 Optional Memory Protection Unit

This section describes the optional

Use the MPU registers to define the MPU regions and their attributes. The MPU registers are:

4.5.2 MPU Control Register

The MPU_CTRL register:

- enables the MPU
- enables the default memory map background region
- enables use of the MPU when in the hard fault, *Non-maskable Interrupt* (NMI), and FAULTMASK escalated handlers.

Glossary