ARM Cortex-M technologies

Each Cortex-M series processor delivers specific benefits; underpinned by fundamental technologies that make Cortex-M processors ideal for a broad range of embedded applications.

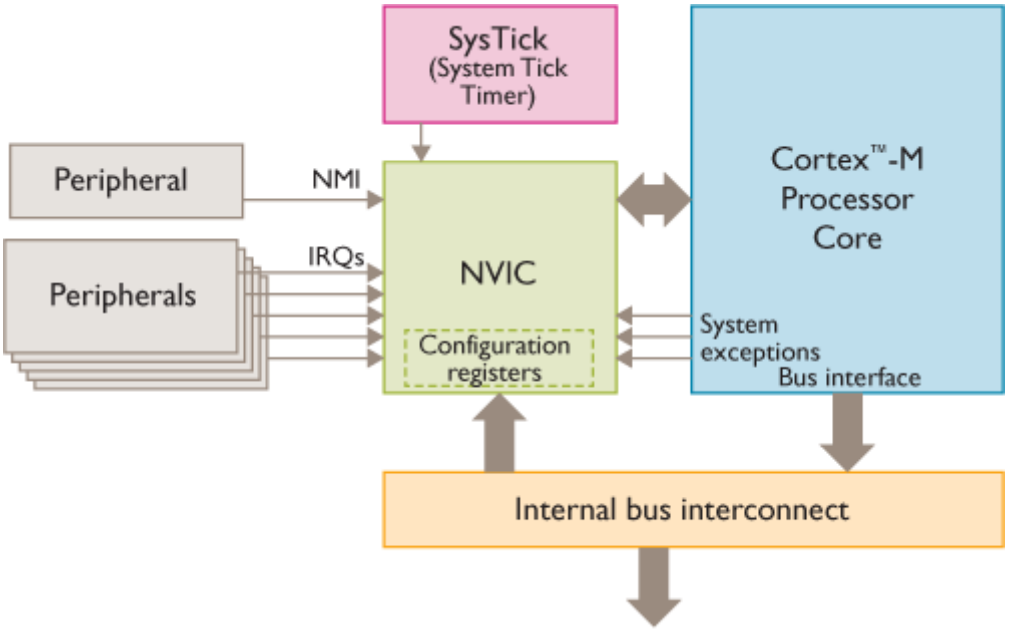
RISC processor core	Thumb-2® technology
• High performance 32-bit CPU • Deterministic operation • Compact, low latency pipeline	• Optimal blend of 16/32-bit instructio • 30% smaller code size than 8-bit de • No compromise on performance
Low power modes	**Nested Vectored Interrupt Controller**
• Integrated sleep state support • Multiple power domains • Architected software control	• Low latency, low jitter interrupt resp • No need for assembly programming • Interrupt service routines in pure C
Tools and RTOS support	**CoreSight debug and trace**
• Broad 3rd party tools support • Cortex Microcontroller Software Interface Standard (CMSIS) • Maximizes software reuse	• JTAG or 2-pin Serial Wire Debug (SWD)connection • Support for multiple processors

CMSIS

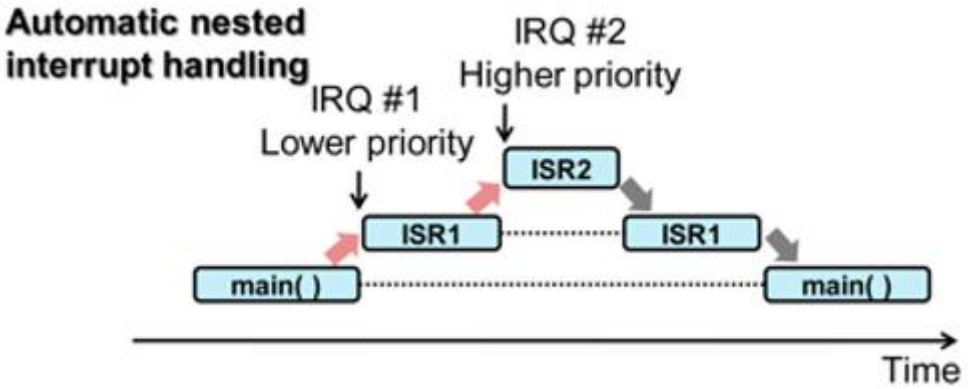
The ARM Cortex Microcontroller Software Interface Standard (CMSIS) is a vendor-independent hardware abstraction layer for the Cortex-M processor series.The CMSIS enables consistent and simple software interfaces to the processor for interface peripherals, real-time operating systems, and middleware, simplifying software re-use. With a reduced learning curve for new microcontroller developers, CMSIS shortens the time to market for new products.

In-depth: Nested Vectored Interrupt Controller (NVIC)

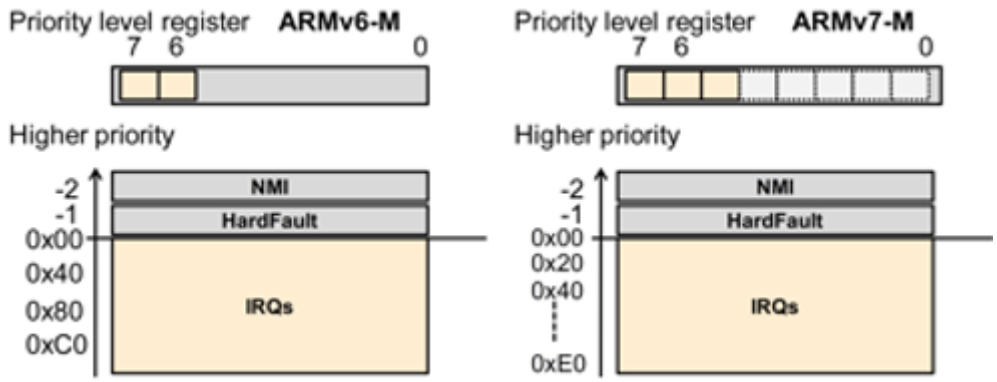
The NVIC is an integral part of all Cortex-M processors and provides the processors' outstanding interrupt handling abilities. In the Cortex-M0, Cortex-M0+ and Cortex-M1 processors, the NVIC support up to 32 interrupts (IRQ), a Non-Maskable Interrupt (NMI) and various system exceptions. The Cortex-M3 and Cortex-M4 processors extend the VIC to support up to 240 IRQs, 1 NMI and further system exceptions.



Most of the NVIC settings are programmable. The configuration registers are part of the memory map and can be accessed as C pointers. The CMSIS library also provided various helper functions to make interrupt control easier.
Inside the NVIC, each interrupt source is assigned an interrupt priority. A few of the system exceptions like such as NMI haves a fixed priority level, and others hashave programmable priority levels. By assigning different priorities to each interrupt, the NVIC can support Nested Interrupts automatically without any software intervention.



The architecture provides 8-bits of priority level settings for each programmable interrupt or exception. To reduce gate count, only parts of these registers are implemented. In the Cortex-M0, Cortex-M0+ and Cortex-M1 processors (ARMv6-M architecture), 4 programmable levels are provided. In the Cortex-M3 and Cortex-M4 processors (ARMv7-M architecture), the designs allow from 8 priority levels to 256 levels.

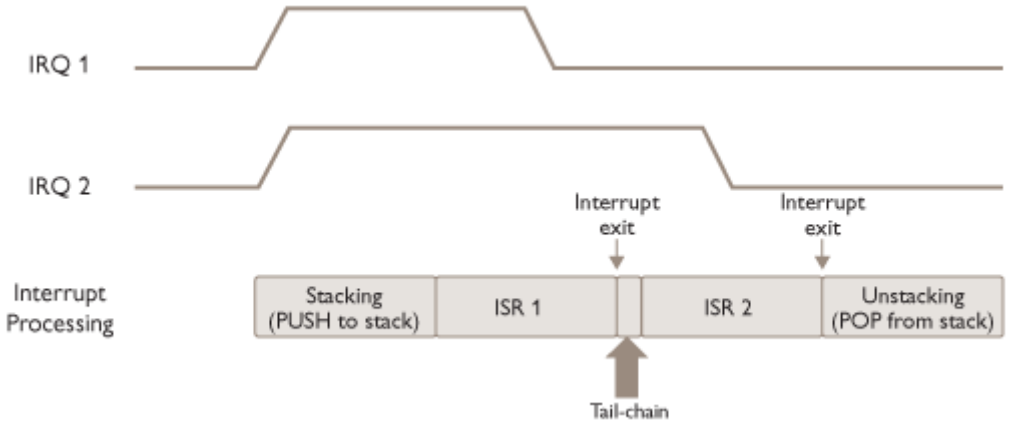


To make the Cortex-M processors easier to use, the Cortex-M processor uses a stack based exception model. When an exception takes place a number of registers are pushed on to the stack. These registers are restored to their original values when the exception handler completes. This allows the exception handlers to be written as normal C functions, and also reduce the hidden software overhead ofin interrupt processing.

In addition, the Cortex-M processors use a vector table that contains the address of the function to be executed for eacha particular interrupt handler. On accepting an interrupt, the processor fetches the address from the vector table. Again, this avoids software overhead and reduces interrupt latency.

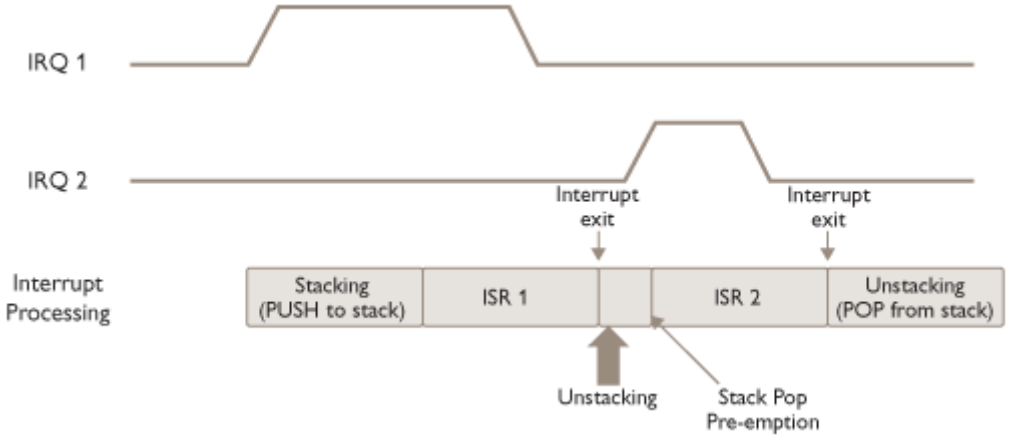
Various optimization techniques are also used in the Cortex-M processor implementationss to make interrupt processing more efficiency and make the system more responsive:

Tail chaining – If another exception is pending when an ISR exits, the processor does not restore all saved registers from the stack and instead moves on to the next ISR. This reduces the latency when switching from one exception handler to another.

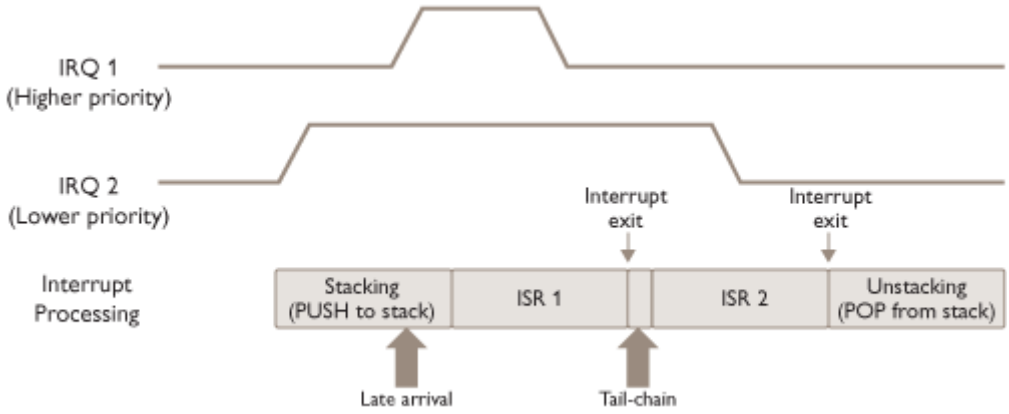


Stack pop pre-emption – If another exception occurs during the unstacking process of an exception, the processor abandons the stack Pop and services the new interrupt

immediately as shown above. By pre-empting and switching to the second interrupt without completing the state restore and save, the NVIC achieves lower latency in a deterministic manner.



Late arrival – If a higher priority interrupt arrives during the stacking of a lower priority interrupt, the processor fetches a new vector address and processes the higher priority interrupt first.



With these optimizations, the interrupt overhead reduces as the interrupt loading increases, allowing high interrupt processing throughput in embedded systems.

**Get more on ARM Webs:
http://www.arm.com/products/