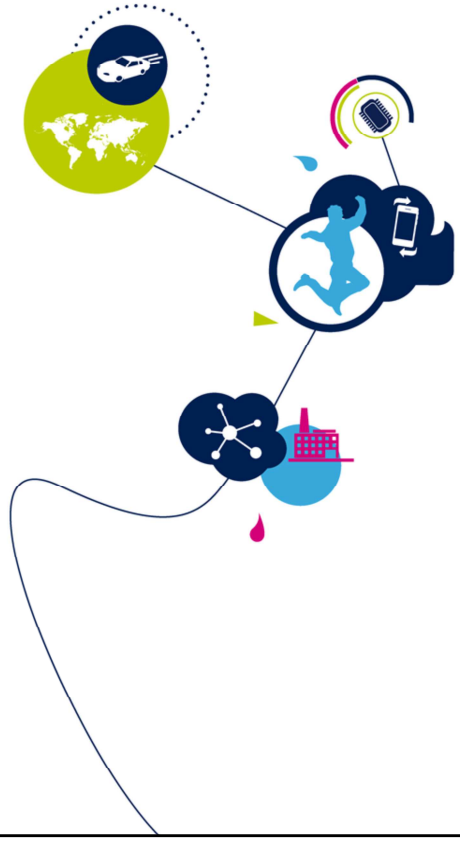
STM32H7 - NVIC

Nested Vectored Interrupt Controller

Revision 1.0



life.augmented

Hello, and welcome to this presentation of the STM32 nested vectored interrupt controller (NVIC). We will be presenting the features of this controller.

Overview

- Both CPU1 (Cortex®-M7) and CPU2 (Cortex®-M4 only for STM32H7x5 and STM32H7x7 lines) cores have their own nested vector interrupt controller (respectively, NVIC1 and NVIC2), which include the following features:
  - up to 150 maskable interrupt channels
  - 16 programmable priority levels
  - low-latency exception and interrupt handling
  - power management control
  - implementation of system control registers

Application benefits

- Supports prioritization levels with dynamic control
- Fast response to interrupt requests
- Relocatable vector table

life.augmented

For STM32H7 microcontrollers, both CPU1 (Cortex®-M7) and CPU2 (Cortex®-M4) cores have their own nested vector interrupt controller (respectively NVIC1 and NVIC2), which include the following features:
• up to 150 maskable interrupt channels (not including the 16 interrupt lines of Cortex®-M7 with FPU)
• 16 programmable priority levels (4 bits of interrupt priority are used)
• low-latency exception and interrupt handling
• power management control
• implementation of system control registers.
Application can benefit from dynamic prioritization of the interrupt levels, fast response to the requests thanks to low latency response and tail chaining as well as from vector table relocation.

Key features

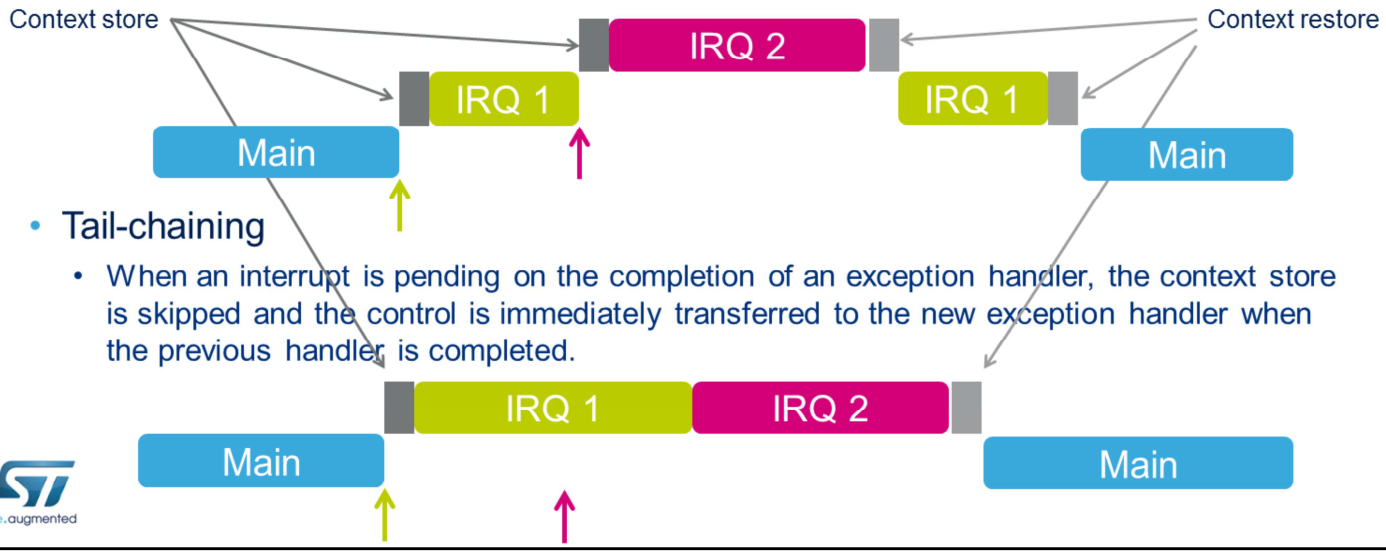
- Fast response to interrupt requests
  - Most peripherals have a dedicated interrupt
- Dynamic reprioritization of interrupts
- Dynamic relocation of interrupt vector table

life.augmented

The NVIC provides a fast response to interrupt requests, allowing an application to quickly serve incoming events. Most of the peripherals have a unique interrupt vector, making development of the application easier (with less need to programmatically determine the source of an interrupt during processing). The interrupt vector table can also be relocated, which allows the system designer to adapt the placement of interrupt service routines to the application's memory layout.

Exception entry and return

- Preemption and interrupt nesting
  - The execution of an interrupt handler can be preempted by an exception having a higher priority



- Tail-chaining
  - When an interrupt is pending on the completion of an exception handler, the context store is skipped and the control is immediately transferred to the new exception handler when the previous handler is completed.

The NVIC provides several features for efficient handling of exceptions.
When an interrupt is served and a new request with higher priority arrives, the new exception can preempt the current one. This is called nested exception handling. The previous exception handler resumes execution after the higher priority exception is handled.
When an interrupt request with lower or equal priority is raised during execution of an interrupt handler, it becomes pending. Once the current interrupt handler is finished, the context saving and restoring process is skipped and control is transferred directly to the new exception handler to decrease interrupt latency.

Exception entry and return

- Late-arriving
  - When a higher-priority exception occurs during state-saving for a previous exception, the processor switches to handle the higher-priority exception immediately.

- Return
  - When the exception handler is finished and no other exception is pending, the processor pops the stack and restores the program state before the interrupt occurred.

life.augmented

When an interrupt arrives, the processor first saves the program context before executing the interrupt handler. If the processor is performing this context-saving operation when an interrupt of higher priority arrives, the processor switches directly to handling the higher-priority interrupt when it is finished saving the program context .
When all of the exception handlers have been run and no other exception is pending, the processor restores the previous context from the stack and returns to normal application execution.

NVIC hints and tips

- Ensure software uses correctly-aligned register accesses
- An interrupt can become pending even if disabled
  - Disabling an interrupt only prevents the processor from taking that interrupt
- Before relocating the vector table, ensure new entries are correctly set up for all enabled interrupts
  - This includes fault handlers and NMIs
  - Do this before programming the VTOR register to relocate the vector table

life.augmented

When accessing the NVIC registers, ensure that your code uses a correctly-aligned register access. Unaligned access is not supported for NVIC registers.
An interrupt becomes pending when the source asks for service. Disabling the interrupt only prevents the processor from taking that interrupt. Make sure the related interrupt flag is cleared before enabling the interrupt vector.
Before relocating the vector table using the VTOR register, ensure that fault handlers, NMI and all enabled interrupts are correctly set up on the new location.

References

- For more details, please refer to following documents:
  - Programming manual for Cortex®-M7 (PM0253)
  - Programming manual for Cortex®-M4 (PM0214)
  - STM32H7 reference manuals

life.augmented

For detailed information, please refer to PM0253 and PM0214 programming manuals for the Cortex®-M7 and for Cortex®-M4 cores, respectively.