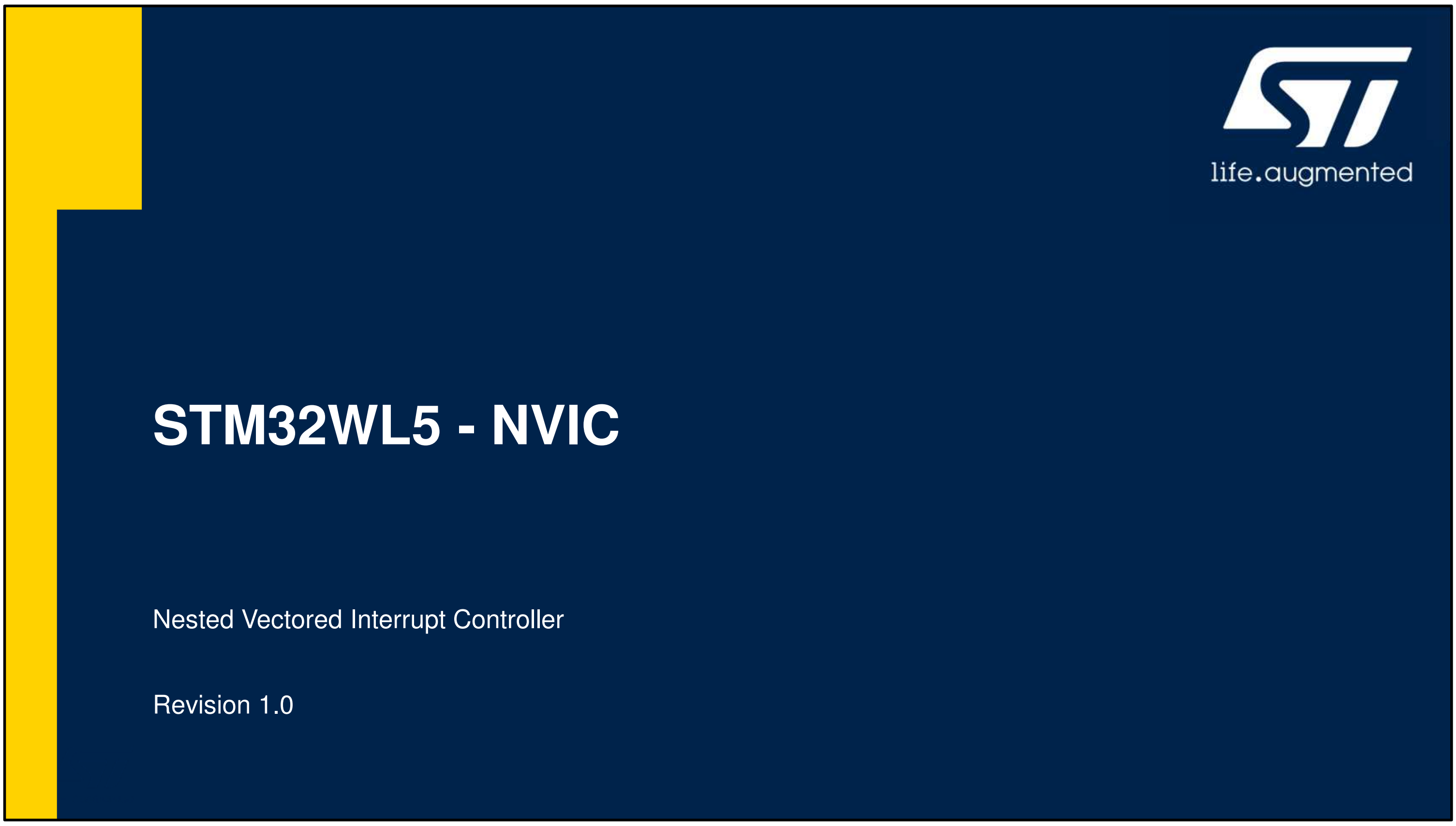
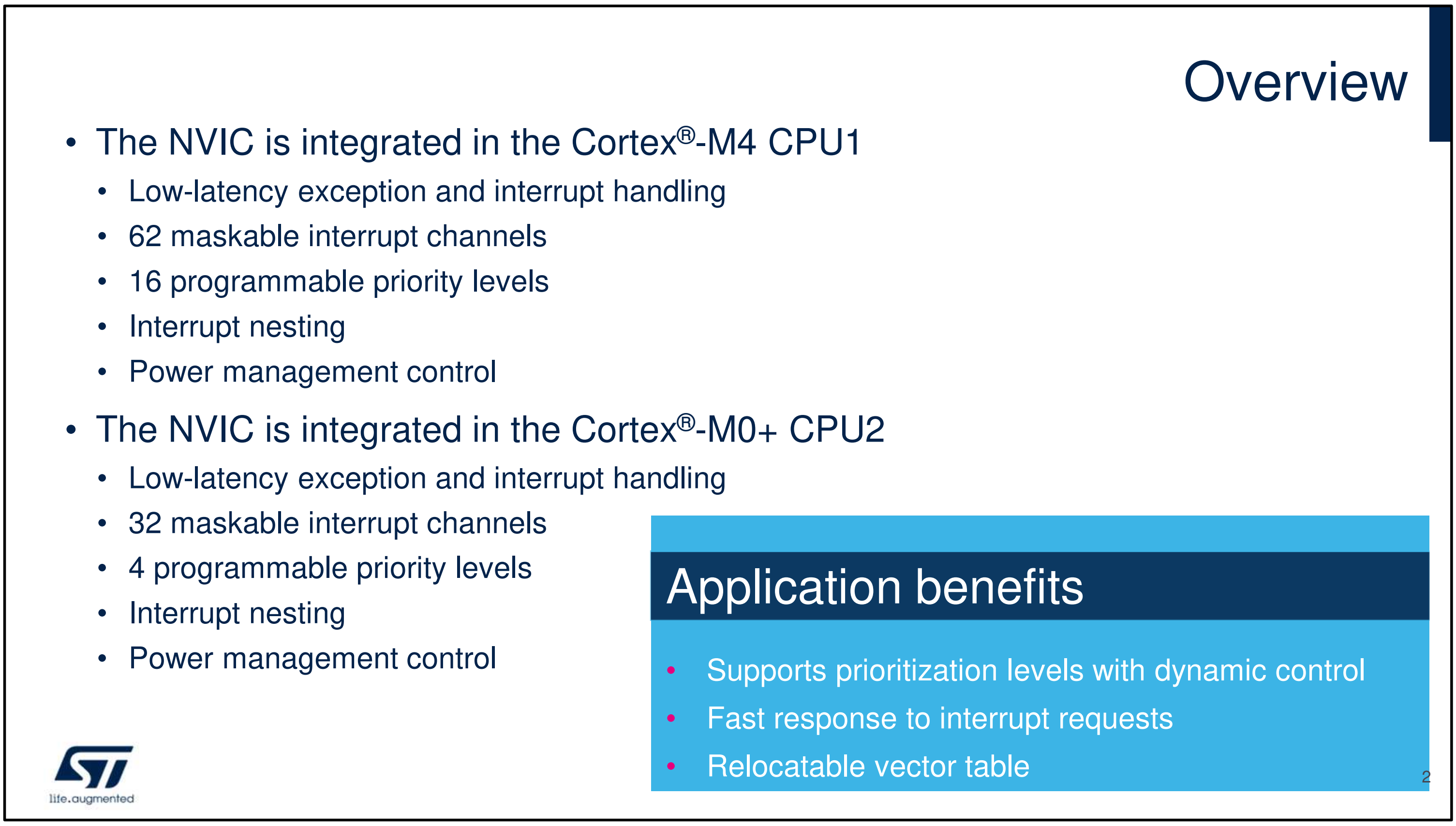


STM32WL5 - NVIC

Nested Vectored Interrupt Controller

Revision 1.0

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



Overview

- The NVIC is integrated in the Cortex®-M4 CPU1
  - Low-latency exception and interrupt handling
  - 62 maskable interrupt channels
  - 16 programmable priority levels
  - Interrupt nesting
  - Power management control
- The NVIC is integrated in the Cortex®-M0+ CPU2
  - Low-latency exception and interrupt handling
  - 32 maskable interrupt channels
  - 4 programmable priority levels
  - Interrupt nesting
  - Power management control

Application benefits

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

The interrupt controller belongs to the CPU enabling a close coupling with the processor core.

The Cortex®-M4 CPU1 main features are:

- 62 interrupt sources
- 16 programmable priority levels

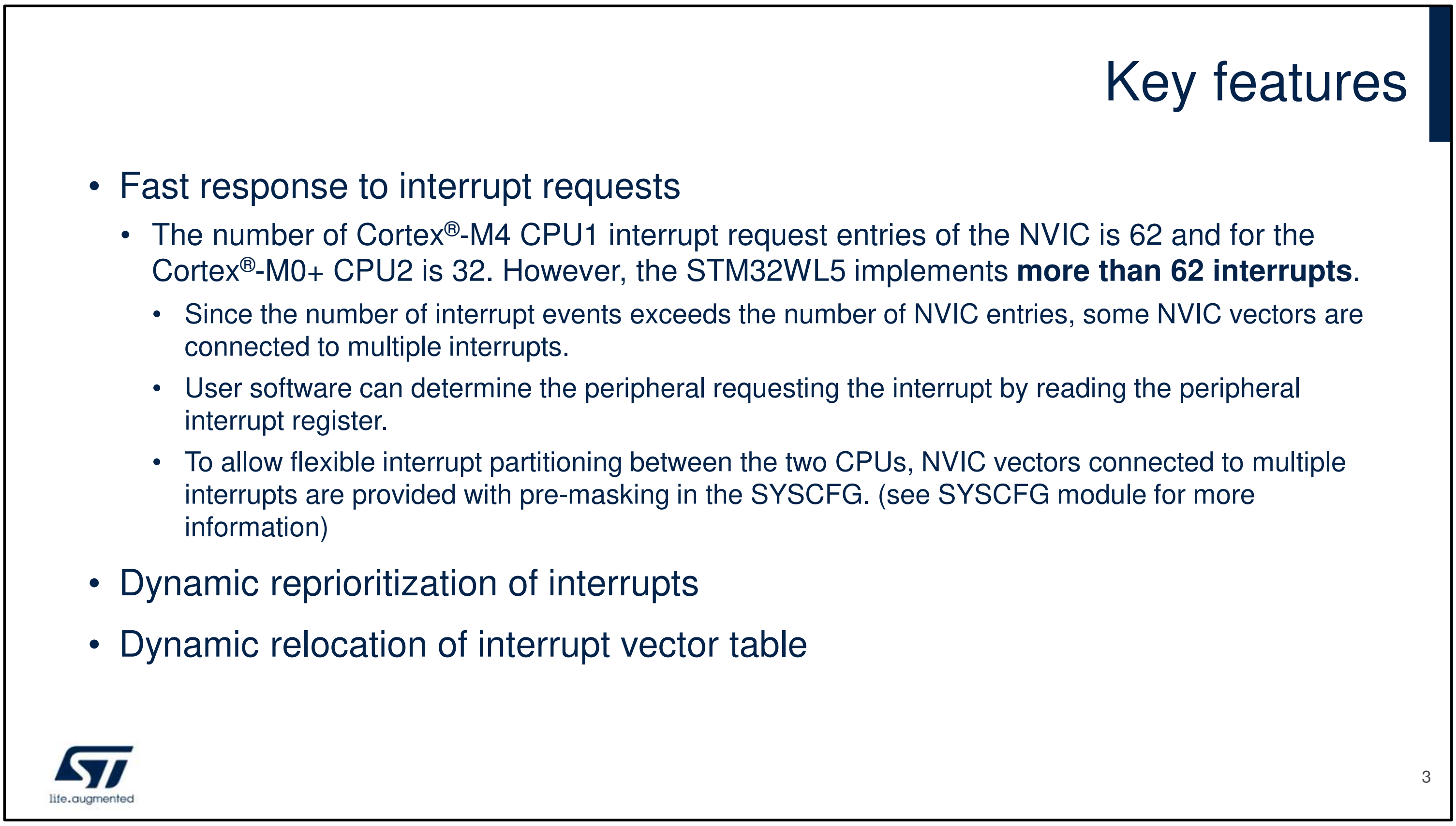
The Cortex®-M0+ CPU2 main features are:

- 32 interrupt sources
- 4 programmable priority levels

The NVIC of both CPUs allow:

- low-latency exception and interrupt handling
- Automatic nesting
- Power management control

Applications can benefit from dynamic prioritization of the interrupt levels, fast response to the requests thanks to low latency responses and tail chaining as well as from vector table relocation.



Key features

- Fast response to interrupt requests
  - The number of Cortex®-M4 CPU1 interrupt request entries of the NVIC is 62 and for the Cortex®-M0+ CPU2 is 32. However, the STM32WL5 implements **more than 62 interrupts**.
    - Since the number of interrupt events exceeds the number of NVIC entries, some NVIC vectors are connected to multiple interrupts.
    - User software can determine the peripheral requesting the interrupt by reading the peripheral interrupt register.
    - To allow flexible interrupt partitioning between the two CPUs, NVIC vectors connected to multiple interrupts are provided with pre-masking in the SYSCFG. (see SYSCFG module for more information)
- Dynamic reprioritization of interrupts
- Dynamic relocation of interrupt vector table

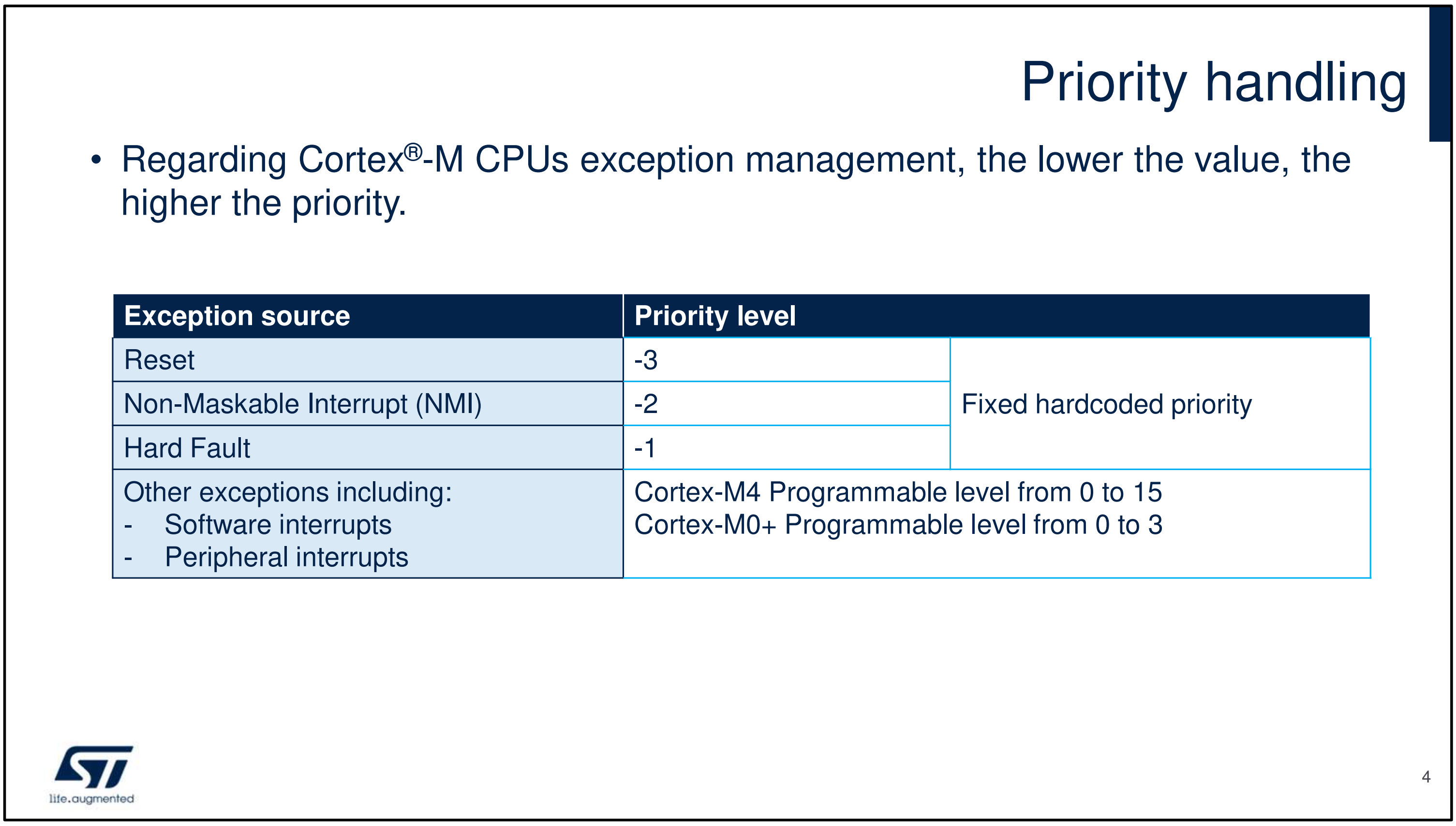
The Nested Vector Interrupt Controller provides a fast response to interrupt requests, allowing an application to quickly serve incoming events.

The STM32WL5 implements more interrupts than the number of NVIC entries. Some interrupts are combined on the same Nested Vector Interrupt Controller vector. By reading the peripheral interrupt register, the software can determine the peripheral that requested the interrupt.

The interrupts combined on the same Nested Vector Interrupt Controller vector are provided with a pre-masking in the SYSCFG to allow flexible interrupt partitioning between the Cortex-M4 and Cortex-M0+ CPUs.

The priority assigned to each interrupt request is programmable and can be dynamically changed.

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. For instance, the vector table can be relocated in RAM.



Priority handling

- Regarding Cortex®-M CPUs exception management, the lower the value, the higher the priority.

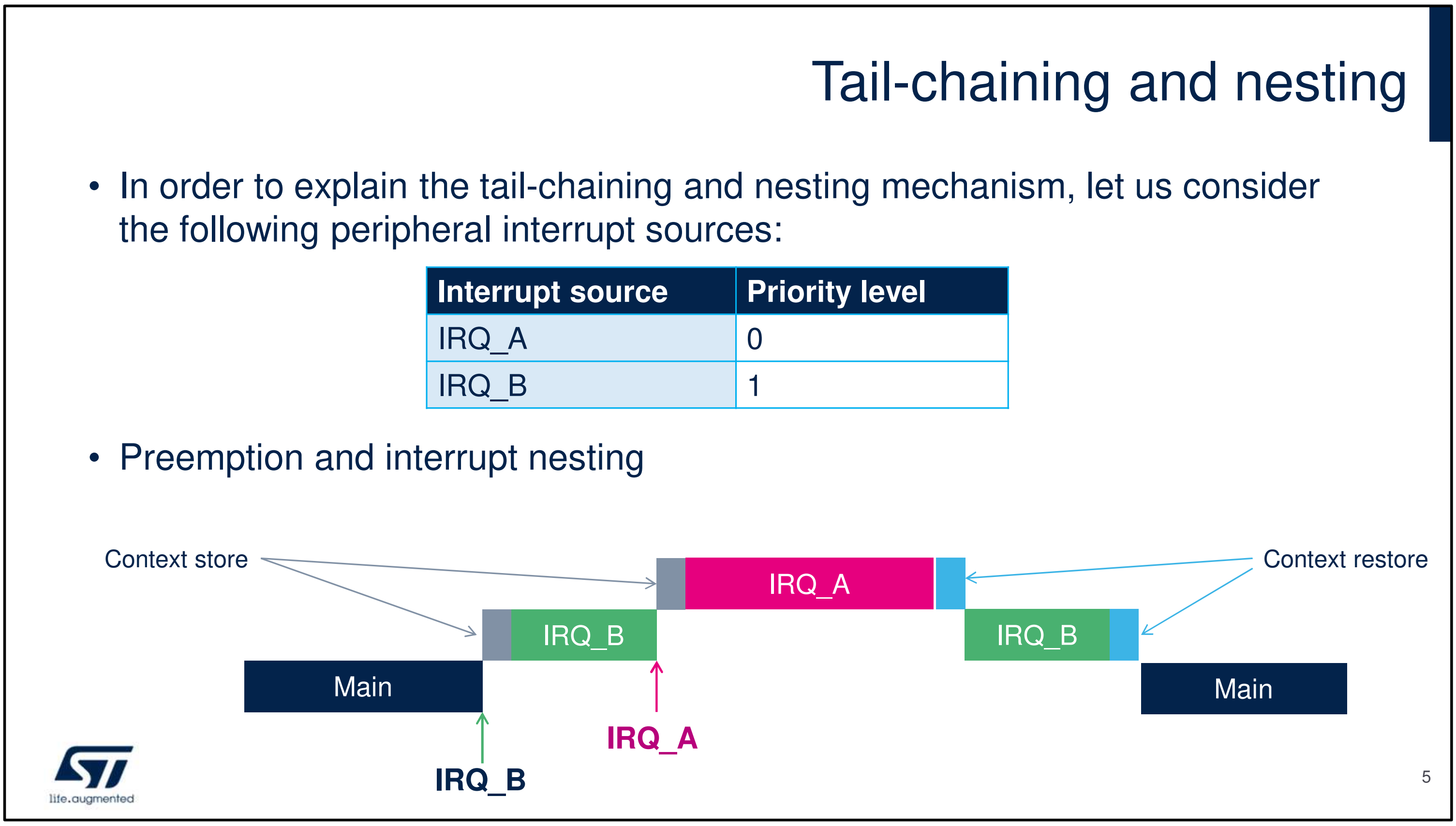
Exception source	Priority level	
Reset	-3	Fixed hardcoded priority
Non-Maskable Interrupt (NMI)	-2	
Hard Fault	-1	
Other exceptions including: - Software interrupts - Peripheral interrupts	Cortex-M4 Programmable level from 0 to 15 Cortex-M0+ Programmable level from 0 to 3	

Software is in charge of assigning a priority level to each interrupt as well as to all exception sources excluding a reset, Non Maskable Interrupt and hard fault.

Whenever a peripheral interrupt is requested at the same time a supervisor call instruction is executed, the relative priority of these hardware and software exceptions dictates which one is handled first.

Regarding the STM32WL5 microcontroller, a Non-Maskable Interrupt (NMI) is caused either by a SRAM2 parity error, a Flash double ECC error or a clock failure.

The priority of any of the software and peripheral interrupt requests is programmable in a dedicated priority field located in the Nested Vector Interrupt Controller registers.



Tail-chaining and nesting

- In order to explain the tail-chaining and nesting mechanism, let us consider the following peripheral interrupt sources:

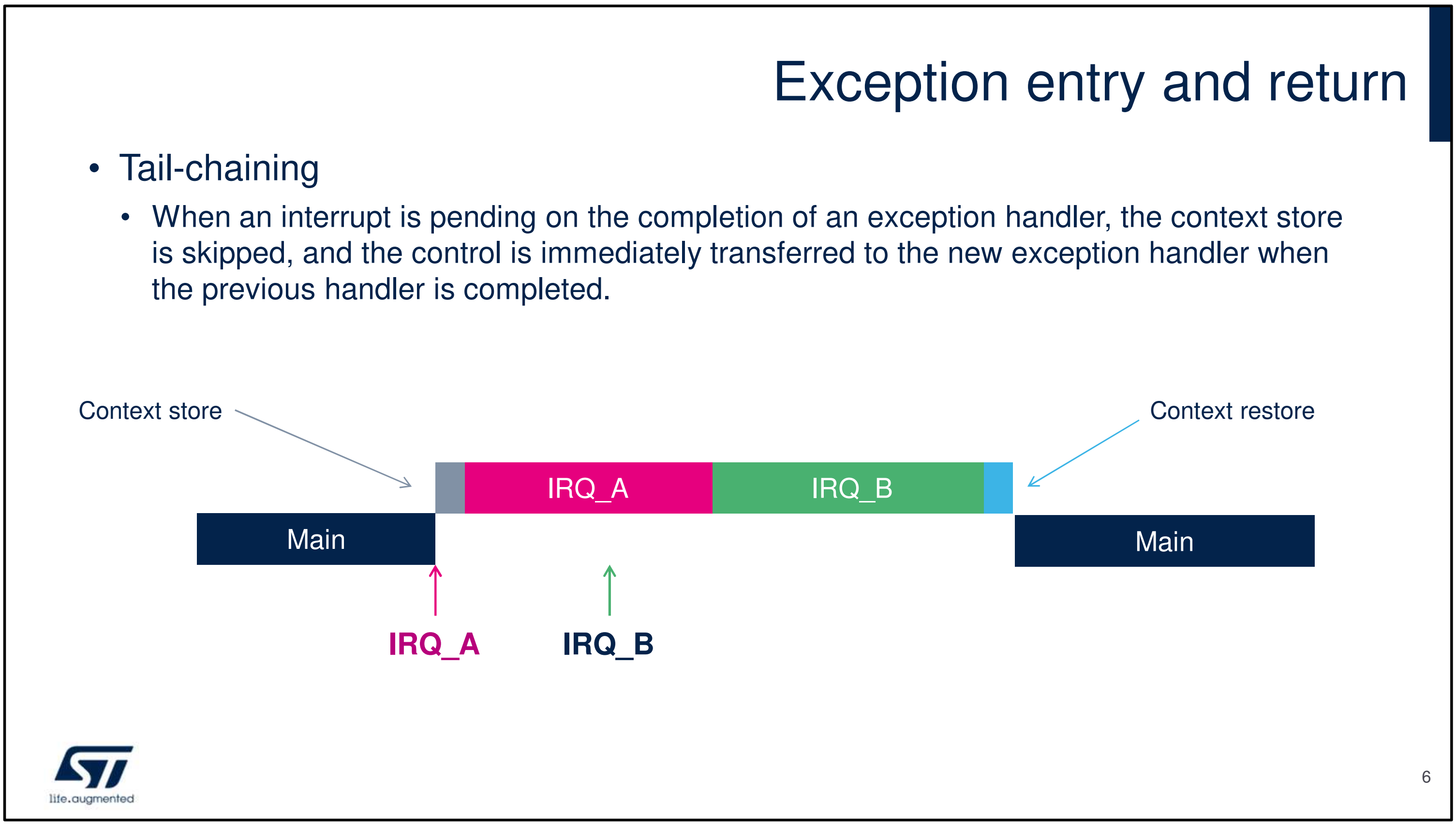
Interrupt source	Priority level
IRQ_A	0
IRQ_B	1

- Preemption and interrupt nesting

The Nested Vector Interrupt Controller 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.

A microcode present in the Cortex®-M4 automatically pushes the context to the current stack and restores it upon interrupt return.



Exception entry and return

- 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.

Context store

Context restore

IRQ_A

IRQ_B

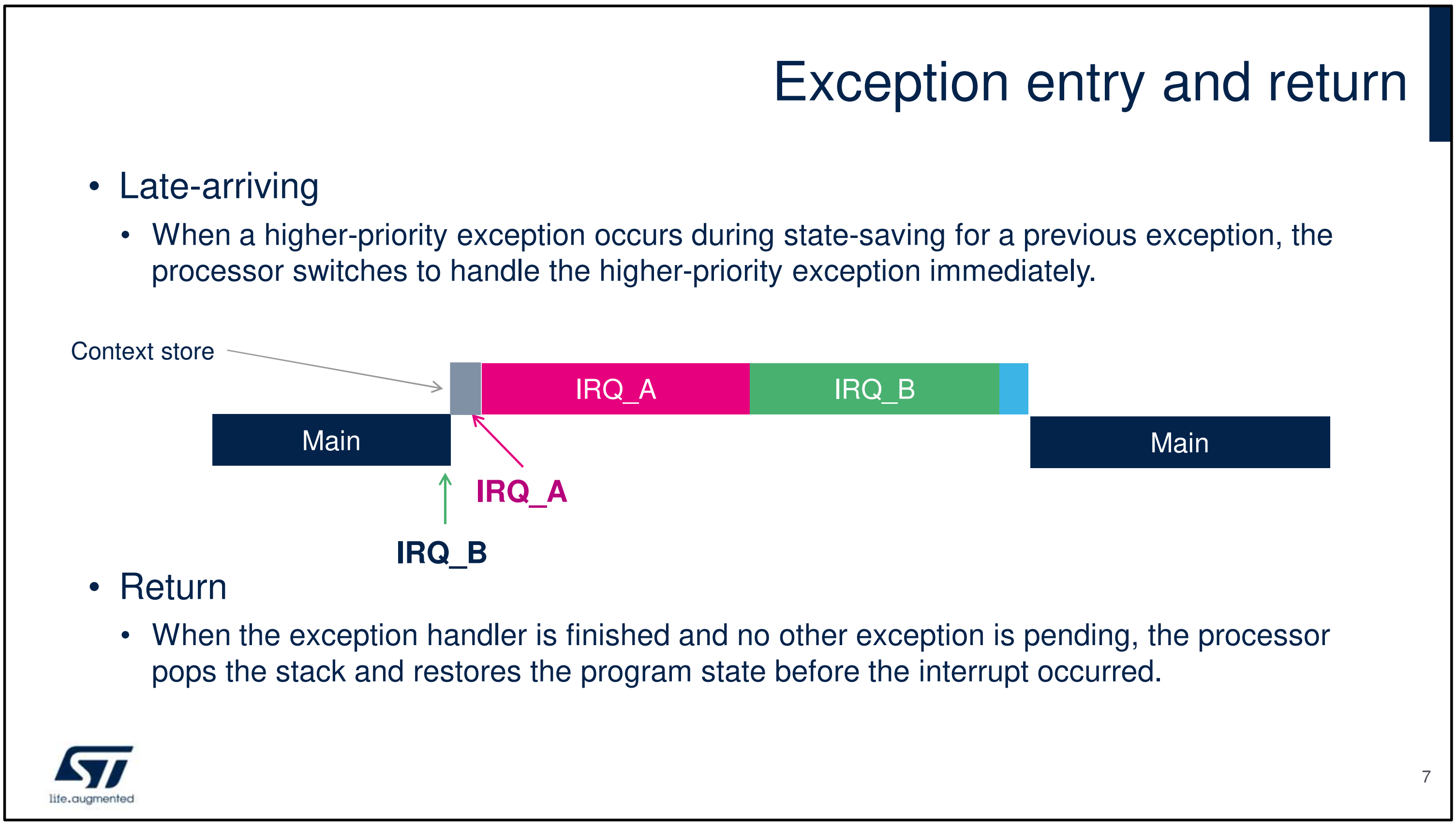
Main

Main

IRQ_A

IRQ_B

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.
So back-to-back interrupts with decreasing priorities (higher priority values) are chained with a very short latency of only a few clock cycles.

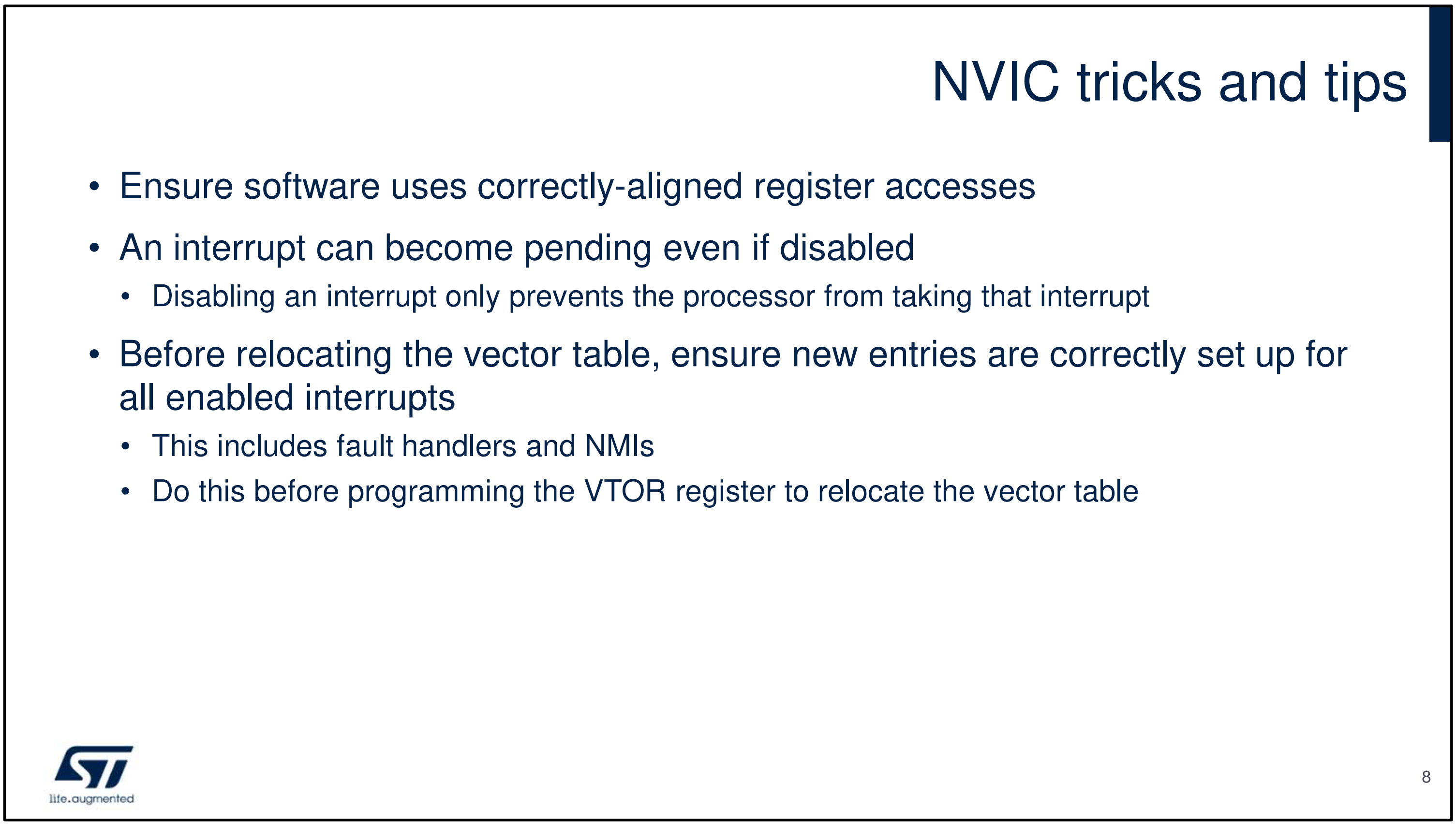


Exception entry and return

- 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.

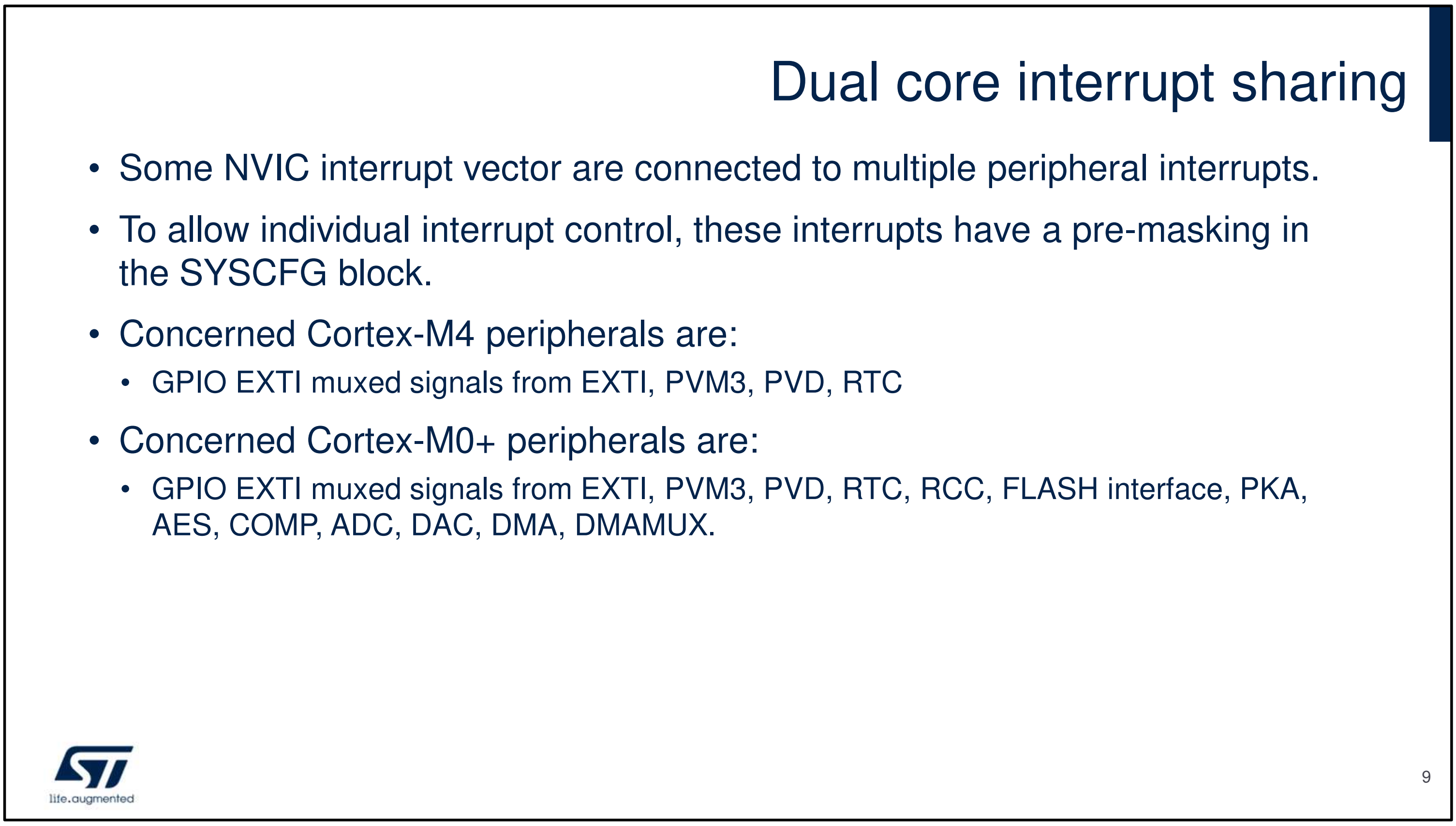
When an interrupt arrives, the processor first saves the program context before executing the interrupt handler. If the processor is performing this context-saving operation when an interrupt of higher priority arrives, the processor switches directly to handling the higher-priority interrupt when it is finished saving the program context. Then tail-chaining is used prior to executing the IRQ_B interrupt service routine. 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 tricks 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

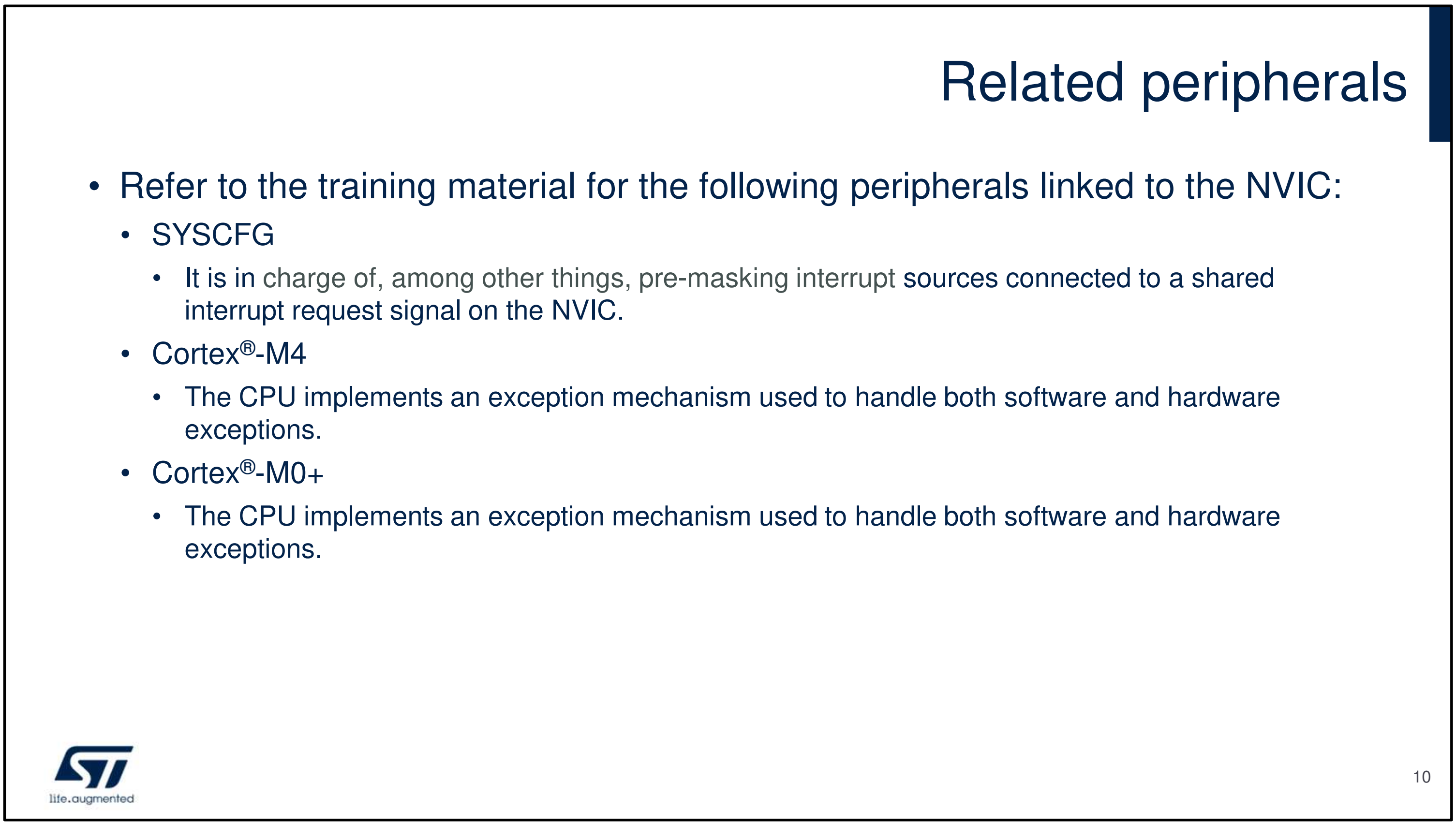
When accessing the Nested Vector Interrupt Controller registers, ensure that your code uses a correctly-aligned register access. Unaligned access is not supported for Nested Vector Interrupt Controller registers as well as all memory-mapped registers located in the Cortex®-M4.
An interrupt becomes pending 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, Non Maskable Interrupt and all enabled interrupts are correctly set up on the new location.



Dual core interrupt sharing

- Some NVIC interrupt vector are connected to multiple peripheral interrupts.
- To allow individual interrupt control, these interrupts have a pre-masking in the SYSCFG block.
- Concerned Cortex-M4 peripherals are:
  - GPIO EXTI muxed signals from EXTI, PVM3, PVD, RTC
- Concerned Cortex-M0+ peripherals are:
  - GPIO EXTI muxed signals from EXTI, PVM3, PVD, RTC, RCC, FLASH interface, PKA, AES, COMP, ADC, DAC, DMA, DMAMUX.

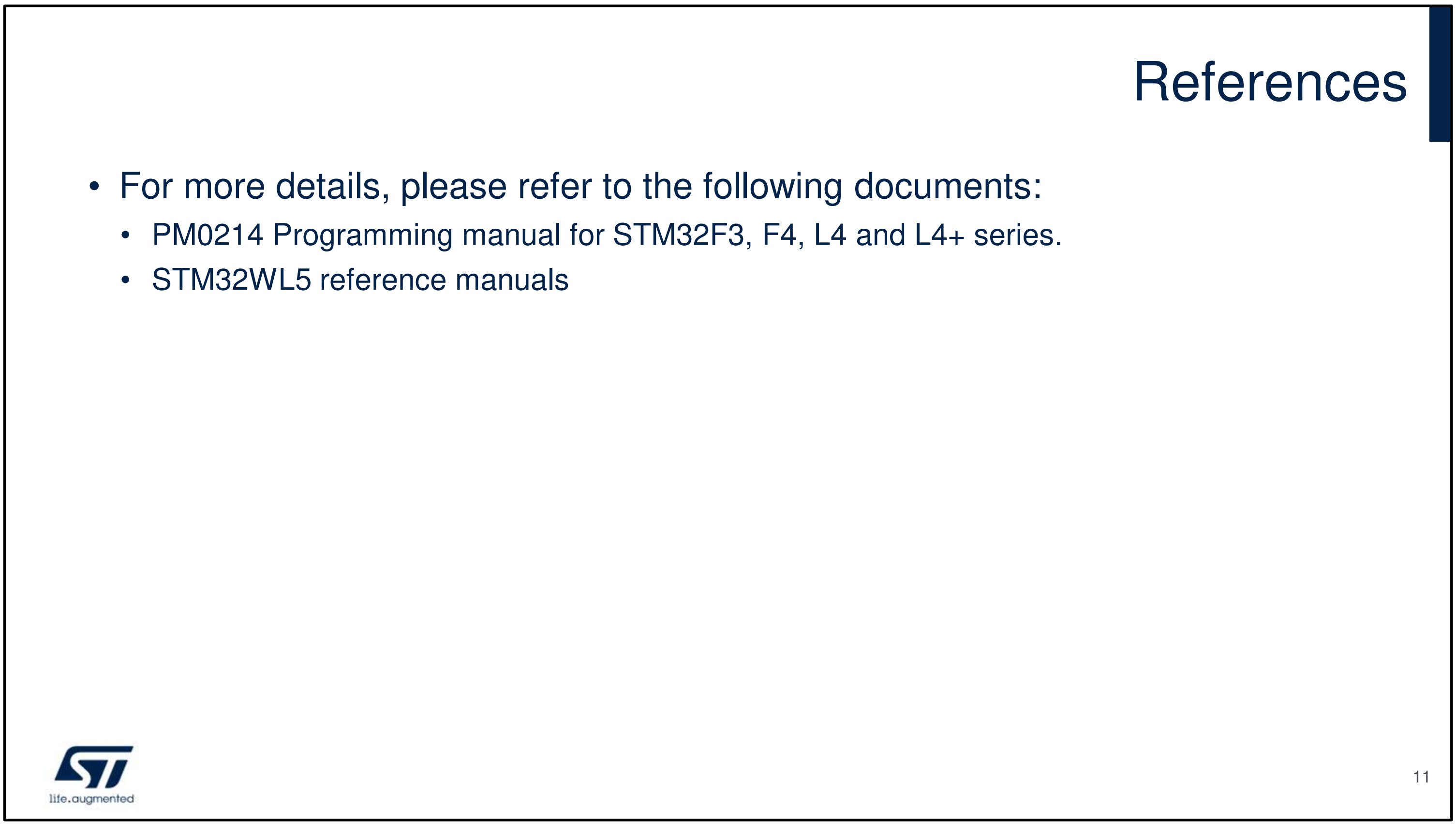
In the dual-core STM32WL5 MCU which embeds a Cortex®-M4 core and a Cortex®-M0+ core, the peripherals interrupts are connected to both cores. To prevent unwanted interruptions, the interrupts mapped on a single Nested Vector Interrupt Controller entry can be pre-masked in the system configuration controller (SYSCFG). The SYSCFG interrupt mask registers ensure that the interrupt sources are only forwarded to the wanted CPU, (Cortex®-M4 or Cortex®-M0+). The impacted peripherals are listed for each CPU.



Related peripherals

- Refer to the training material for the following peripherals linked to the NVIC:
  - SYSCFG
    - It is in charge of, among other things, pre-masking interrupt sources connected to a shared interrupt request signal on the NVIC.
  - Cortex®-M4
    - The CPU implements an exception mechanism used to handle both software and hardware exceptions.
  - Cortex®-M0+
    - The CPU implements an exception mechanism used to handle both software and hardware exceptions.

The Nested Vector Interrupt Controller is linked with the SYSCFG module, the Cortex-M4 CPU1, and the Cortex-M0+ CPU2. Please refer to the related presentations.



References

- For more details, please refer to the following documents:
  - PM0214 Programming manual for STM32F3, F4, L4 and L4+ series.
  - STM32WL5 reference manuals

life.augmented

11

For detailed information, please also refer to the programming manual for the STM32F3, F4, L4 and L4+ series and the reference **manual of the STM32WL5.**