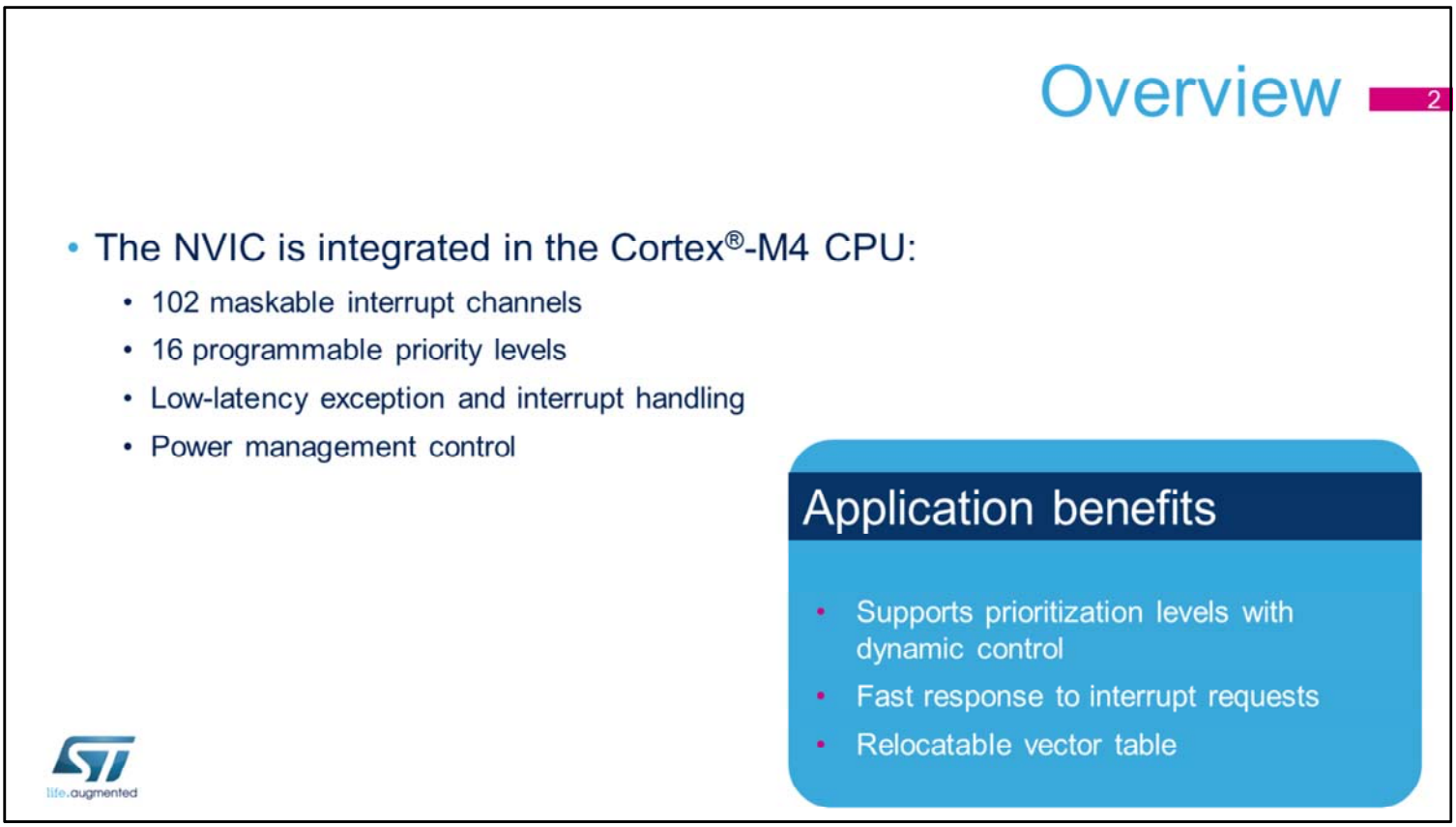
Hello, and welcome to this presentation of the STM32 Nested Vectored Interrupt Controller (or NVIC). We will be presenting the features of this controller.

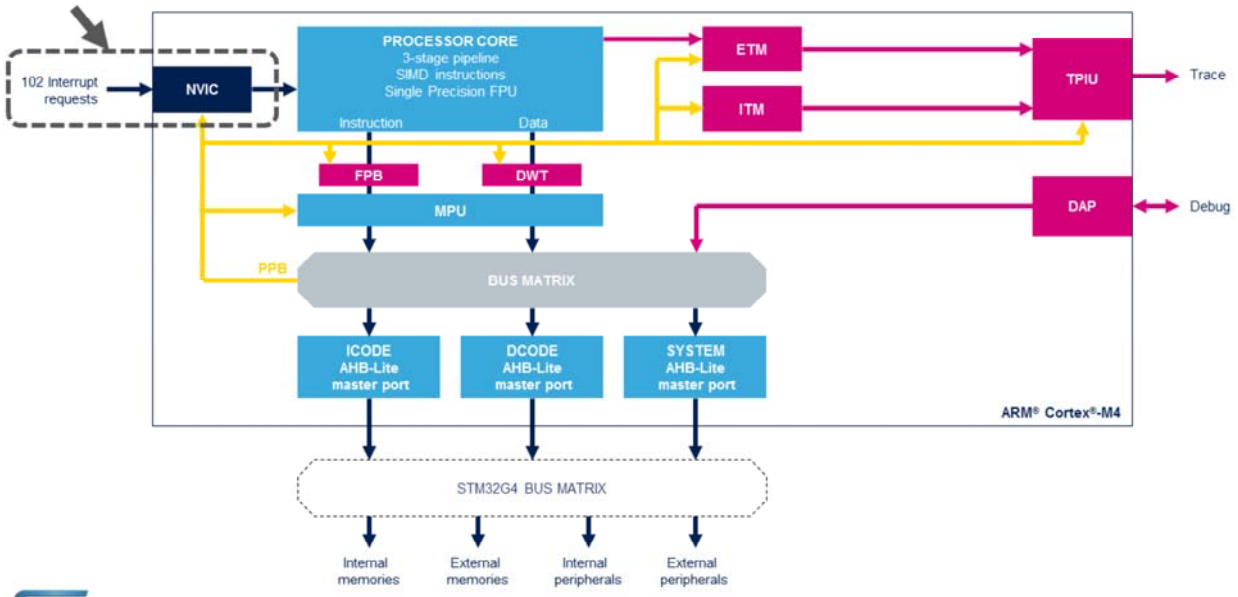


The interrupt controller belongs to the Cortex®-M4 CPU enabling a close coupling with the processor core.
The main features are:

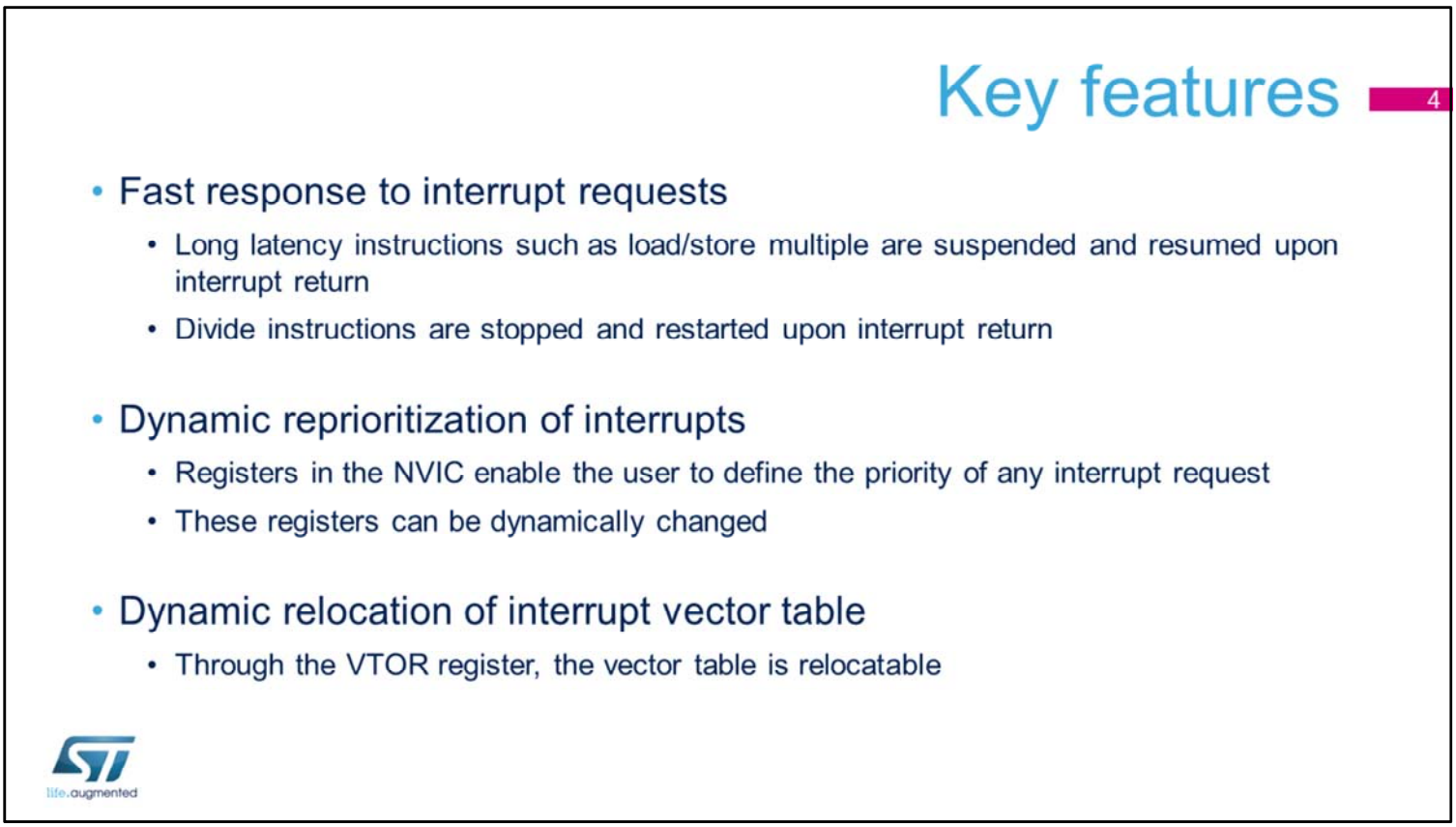
- 102 interrupt sources,
- 16 programmable priority levels,
- Low-latency exception and interrupt handling,
- Automatic nesting,
- Power management control.

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.

Core architecture overview



All interrupts including the core exceptions are managed by the NVIC.
The NVIC and the processor core interface are closely coupled, which ensures a low interrupt latency and enables the efficient processing of late-arriving interrupts.
Access to the NVIC's control and status registers is performed through the Private Peripheral Bus (or PPB) internal to the Cortex®-M4 CPU.



The NVIC provides a fast response to interrupt requests, allowing an application to quickly serve incoming events. An interrupt is handled without waiting for the completion of a long instructions sequence. These instructions will be either restarted or resumed upon the interrupt return. The priority assigned to each interrupt request is programmable and can be dynamically changed.
The vector table, containing the address of the exception handlers, can also be relocated, which allows the system designer to adapt the placement of the 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: - **Peripheral interrupts** - Software exceptions	Programmable level from 0 to 15	

life.augmented

Software is in charge of assigning a priority level to each interrupt as well as to all exception sources not including reset, non-maskable interrupt (or NMI) and hard fault. Whenever a peripheral interrupt is requested at the same time as a supervisor call instruction is executed, the relative priority of these hardware and software exceptions will dictate which one will be taken first. Regarding the STM32G4, NMI is caused by a SRAM parity error, a flash double ECC error or clock failure. The priority of any of the 102 peripheral interrupt requests is programmable in a dedicated priority field located in Cortex®-M4 NVIC 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 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.
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 (1/2)

- 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

life.augmented

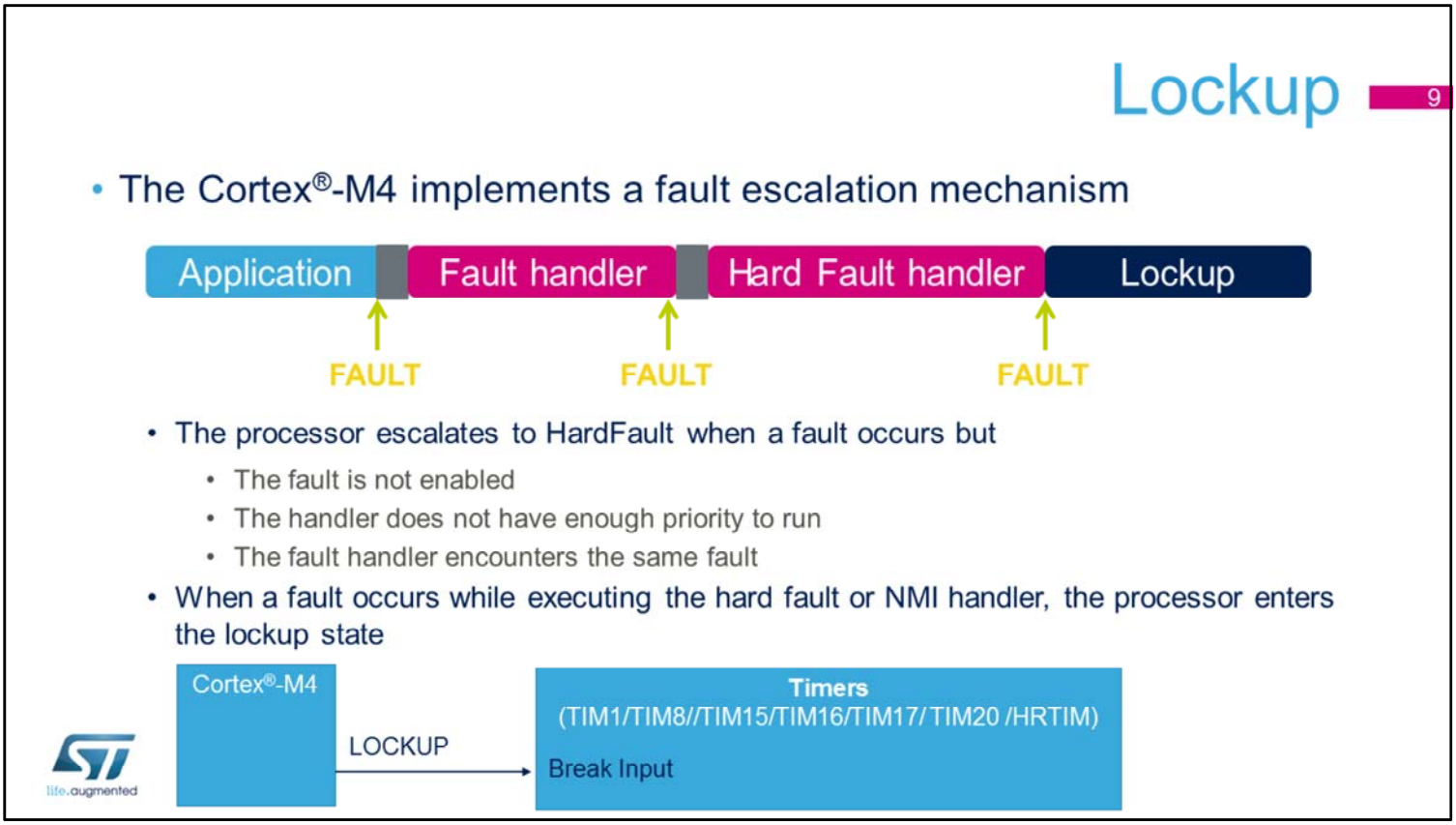
When an interrupt request with lower or equal priority is raised during execution of an interrupt handler, it becomes pending. Once the current interrupt handler is finished, the context saving and restoring process is skipped and control is transferred directly to the new exception handler to decrease interrupt latency.
So back-to-back interrupts with decreasing priorities (higher priority values) are chained with a very short latency of a few clock cycles.

Exception entry and return (2/2)

- 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 will be 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.



When a fault occurs while executing either the bus fault handler or the memory management fault handler or the usage fault handler, the processor escalates this event to the hard fault handler.
Note that bus fault, memory management fault and usage fault have to be explicitly enabled. By default, any fault leads to the hard fault handler.
When a fault occurs while executing the hard fault handler, the processor enters a state named lockup.
It asserts the LOCKUP output to indicate that it has encountered a serious non recoverable error.
In the STM32G4 microcontroller, this LOCKUP output is internally connected to timer break inputs, so that a safe state can be entered when the STM32G4 is used to control a power electronic equipment, such as a motor.

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 NVIC registers, ensure that your code uses a correctly-aligned register access. Unaligned access is not supported for NVIC 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, NMI and all enabled interrupts are correctly set up on the new location.

Related peripherals

- Refer to the training material for the following peripherals linked to the timers:
  - TIMERS
    - The break input of some timers is connected to the Cortex®-M4 LOCKUP output
  - Cortex®-M4
    - The CPU implements an exception mechanism used to handle both software and hardware exceptions.

life.augmented

The NVIC is linked with the TIMERS and the Cortex-M4 CPU modules. Please refer to the related presentations.

References

- For more details, please refer to the following documents:
  - PM0214 Programming manual for Cortex®-M4
  - STM32G4 reference manuals

life.augmented

For detailed information, please refer to the programming manual for the Cortex®-M4 core and the reference manual of the STM32G4.