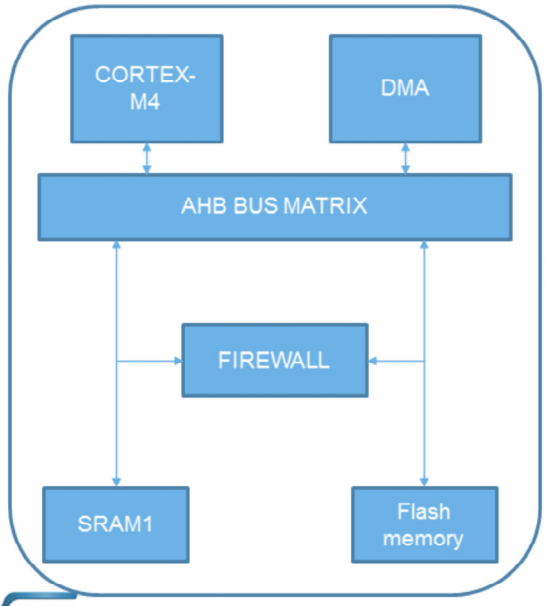
Hello and welcome to this presentation of the STM32L4 Firewall. It covers the main features of this system IP used to secure sensitive code and data.

Overview

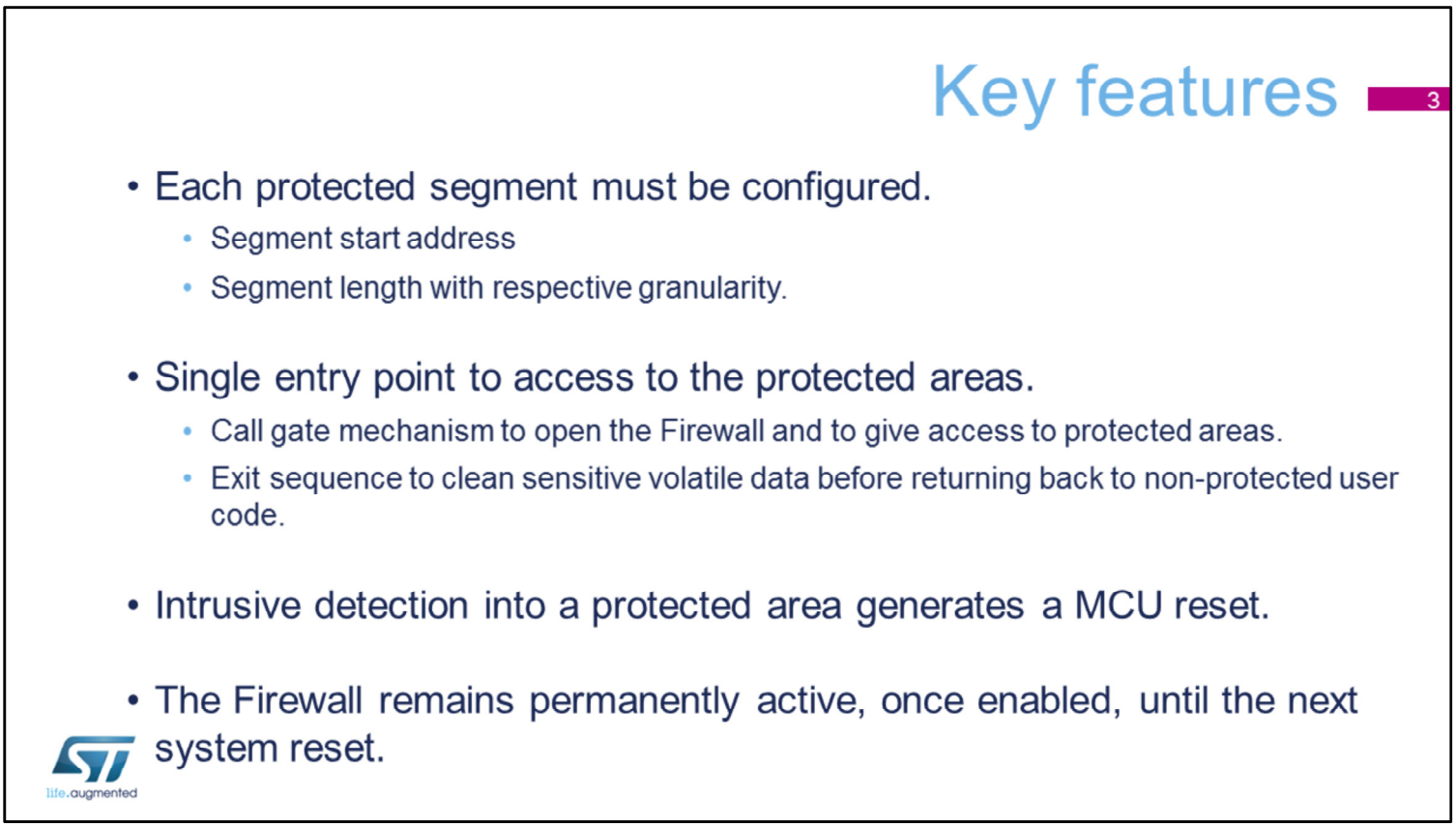


- Protects code and sensitive data from an attack by other processes.
  - Code in Flash memory or SRAM1
  - Constant data in Flash memory
  - Volatile data in SRAM1

Application benefits

- Protects the intellectual property of embedded software interacting with OEM code.
- Prevents attacks designed to dump/execute protected code outside of a protected area.
- Protects access to sensitive data from non-protected user code execution.

Here is an overview of the Firewall's implementation and its benefits for customer applications. The Firewall protects the access to sensitive code and data located in the Flash memory or SRAM1 segments from external processes. Any attack detected by the Firewall causes the MCU to reset, if the Firewall is enabled. The Firewall monitors each access from the AHB masters to the Flash memory or SRAM1 AHB slaves, then depending on the Firewall configuration, allows the access to the memory segment or resets the MCU if not allowed. Having part of the code and data monitored by the Firewall allows users to protect their IP, meaning a third party's intellectual property of embedded software can be protected against code dumping along with any sensitive data stored in SRAM1.



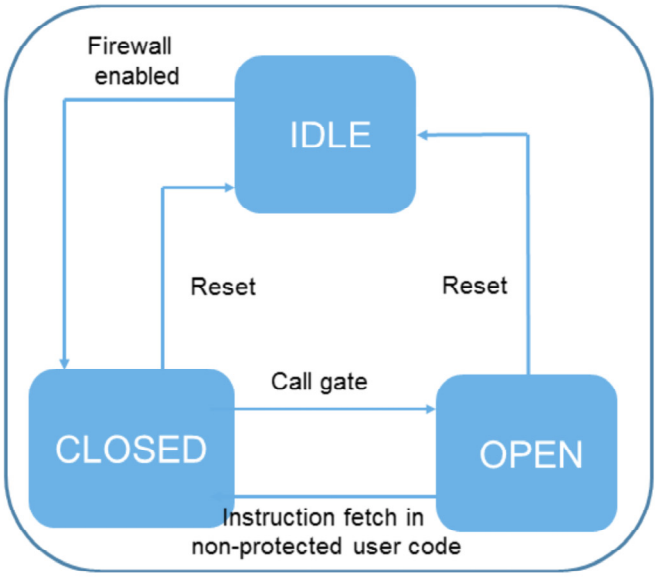
Each memory segment protected by the Firewall is configured independently with a start address and the associated length of the segment. The 3 definable memory segments are the code segment, the volatile data segment and the non-volatile data segment. The Firewall is based on a call gate mechanism used to open the access to these protected segments. The call gate function is the single entry point able to open the Firewall and enable access to the protected segments. To ensure sensitive volatile data is erased before returning back to non-protected user code, the call gate mechanism specifies the exact exit point when jumping back from the protected code segment. The goal is to detect any non-expected code branches and to react by resetting the MCU. To guarantee a maximum level of protection, once the Firewall is enabled it remains active until the next MCU system reset.

Firewall states

4

The Firewall offers a dynamic protection

- Idle state: Firewall is not enabled.
  - No memory bus monitoring by the Firewall
- Closed state: Non-protected user code execution.
  - All access to protected segments is prohibited,
    - SRAM1 included if it is not declared as shared.
  - Only the call gate sequence can be called.
- Open state: protected code execution.
  - Access to the protected segments is allowed.



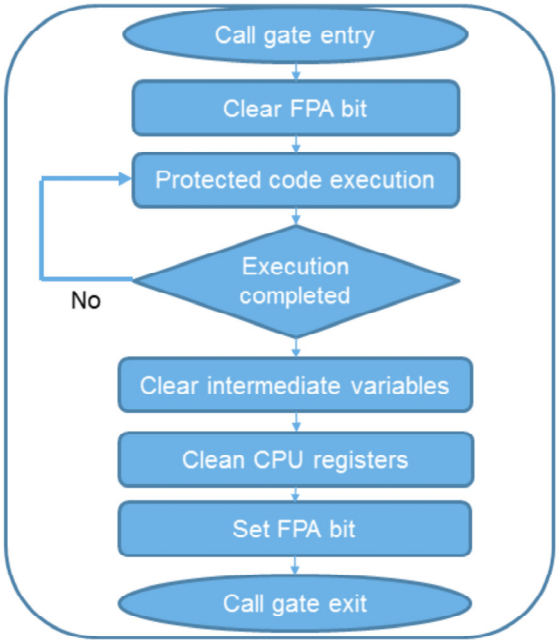
The Firewall is based on 3 states to ensure a dynamic protection of the secure segments. The Idle state is the default state when the Firewall is not enabled. In this state, the AHB memory bus is not monitored. When enabled, the Firewall enters Closed state and all access to the protected segments is prohibited.
The correct call gate entry sequence by non-protected executing code switches the Firewall to the Open state. The protected code can now be executed and access to non-volatile and volatile data segments is allowed. As soon as an instruction fetch is executed, jumping back to the non-protected code area, the Firewall switches back to Closed state. Once closed, all access to the secure areas except for the call gate mechanism, are killed by a MCU RESET.

Call gate function

5

Secure entry and exit to/from protected areas

- Single entry point: call gate function starts at protected code address + 4.
- Clear FPA (Firewall PreArm) bit to generate reset if an unexpected exit sequence occurs.
- Clear the context.
- Allow exit from protected areas by jumping back to the non-protected code.



The Firewall's call gate function architecture offers the best solution for building a secure entry/exit point to the protected memory areas. The call gate function is located in the protected code segment at a mandatory fixed address corresponding to the code segment start address + 4, (Scatter file for Keil, Pragma setup for IAR).The FPA bit has to be cleared immediately in the call gate in order to stop any intrusion that exits the protected code in a non-protected user area when not expected. Before leaving the protected code area in execution, it is recommended to clean/clear the context (variables data) and the CPU registers before requesting the exit sequence and jumping back to the non-protected instruction.

Access to protected segments

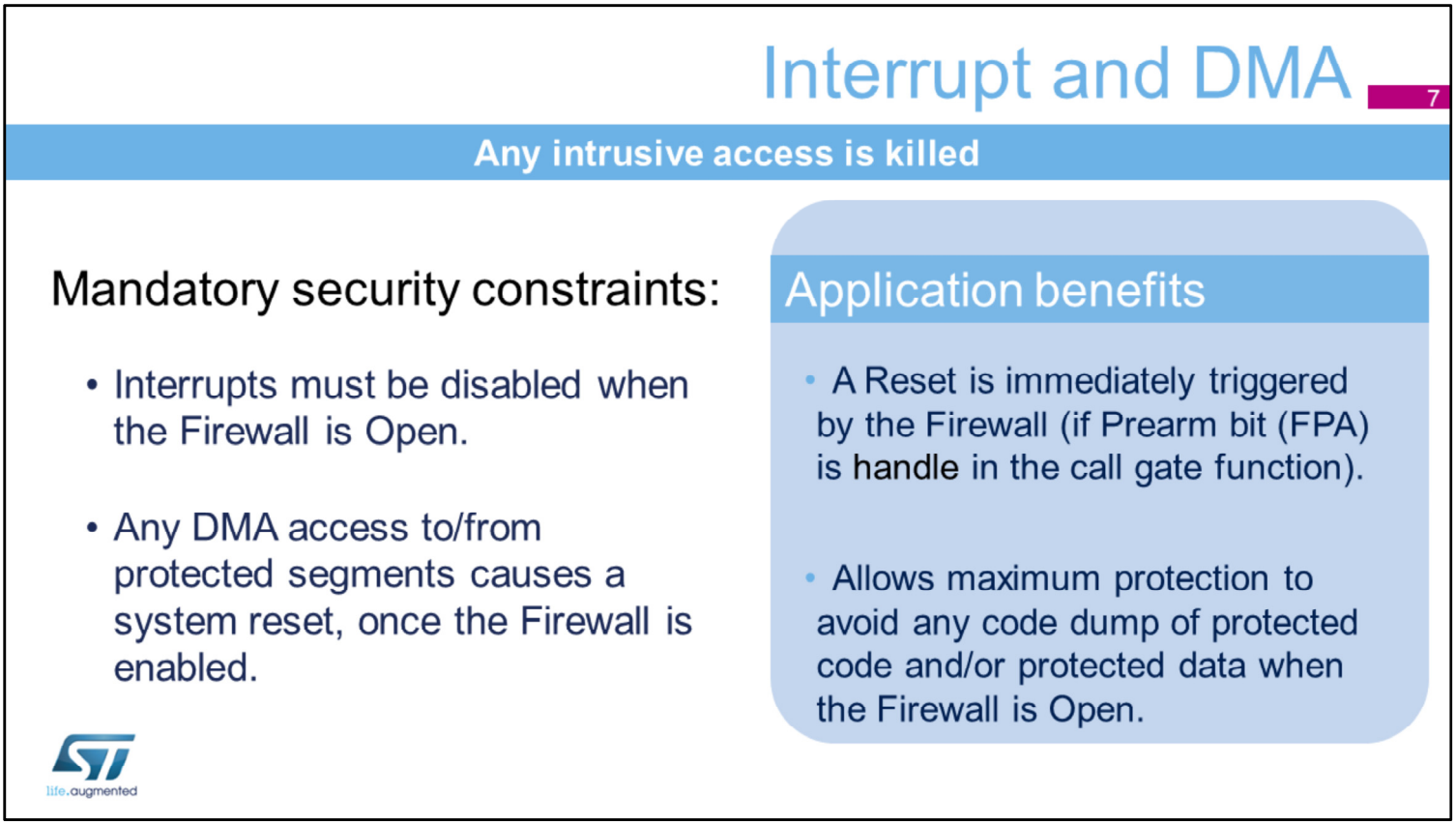
Protected segments		Firewall state		
		Idle (Firewall off)	Closed	Open
Code segment (Flash memory)		R/W/X allowed (*)	**R/W/X not allowed** (except call gate)	R/X allowed **W not allowed**
Constant data segment (Flash memory)			**R/W/X not allowed**	R/W allowed **X not allowed**
Volatile data segment (SRAM1)	**Not shared Not executable**		**R/W/X not allowed**	R/W allowed **X not allowed**
	Not shared Executable		**R/W/X not allowed** (except call gate)	R/W/X allowed
	Shared Not executable		R/W/X allowed	R/W/X allowed
	Shared Executable		R/W/X allowed	R/W/X allowed

R: Read access W: Write access X: eXecute access

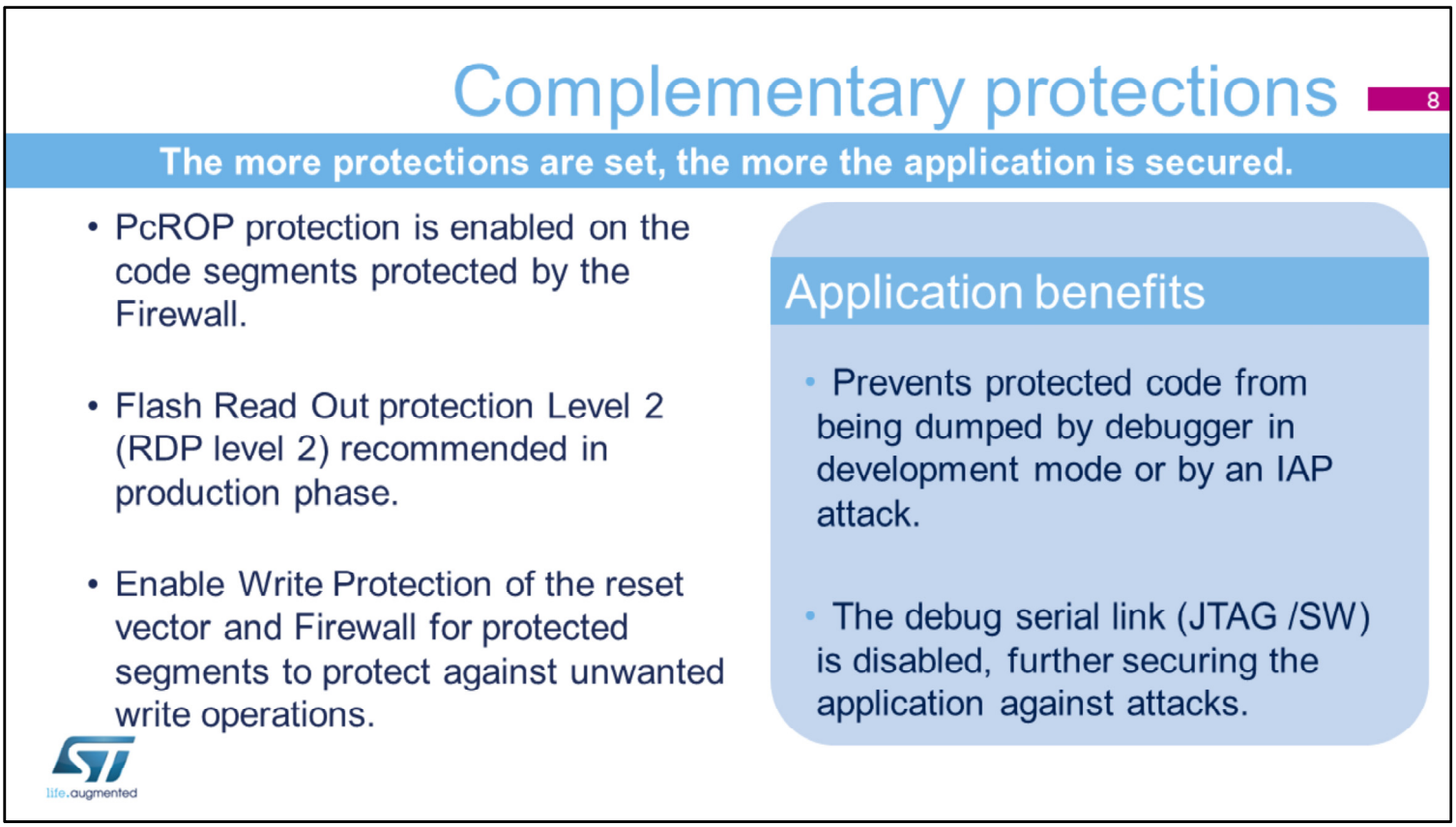
life.augmented

(*) except if other protections are set: Proprietary code readout protection (PCROP), Readout protection (RDP) or Write protection (WRP)

The type of access to the protected segments depends on the Firewall state. When it is closed, any access to the protected area generates a system reset. When the Firewall is open, some access is possible. In the code segment (Flash memory), read operations and instruction fetches are allowed. In the non-volatile data segment (Flash memory), read and write operations are allowed. In the volatile data segment (SRAM1), read, write and execute operations are allowed, if the SRAM1 segment is declared as shared or executable. When the Firewall is disabled (Closed), there is no protection.



Specific constraints must be respected when enabling the Firewall. Interrupts must be disabled from the call gate entry sequence, until the Firewall switches back to the closed state. If an Interrupt Service Routine, ISR occurs, the Firewall generates a reset. All DMA access to/from the protected segments is not allowed and is rejected by the Firewall (system reset). The application benefits are mainly to detect an intrusion, faster, during the protected code execution and to offer a very high level of protection against code dumping using the DMA.



Complementing the Firewall protection, it is required to set the PcROP (Proprietary code Read Out Protection) protection for the protected code segment(s) in the Flash memory. Setting the PcROP stops any code from being dumped by the debugger during the development phase, by external attacks, or from an IAP attack. The PcROP protection mechanism on the STM32L4 is improved over the previous STM32F2/F4/L1 microcontrollers. STM32L4 microcontrollers make it possible to define start/end regions, and there is an option byte which allows a Mass Erase operation, but keeps the PCROP segment protected. In production, ST recommends setting the STM32 Readout protection (RDP) to Level 2, which disables the JTAG link to the MCU. Setting the RDP to Level 2 secures the MCU against any external attacks to the protected segments. ST also recommends enabling Write Protection on the reset vector and the Firewall

configuration to prevent any unwanted write operations to the protected areas.

Related peripherals

- Refer to this training:
  - STM32L4 system memory protection

life.augmented

In addition to this training you can refer to the STM32L4 system memory protection training for more information.