life.augmented

STM32L5 入门课程系列

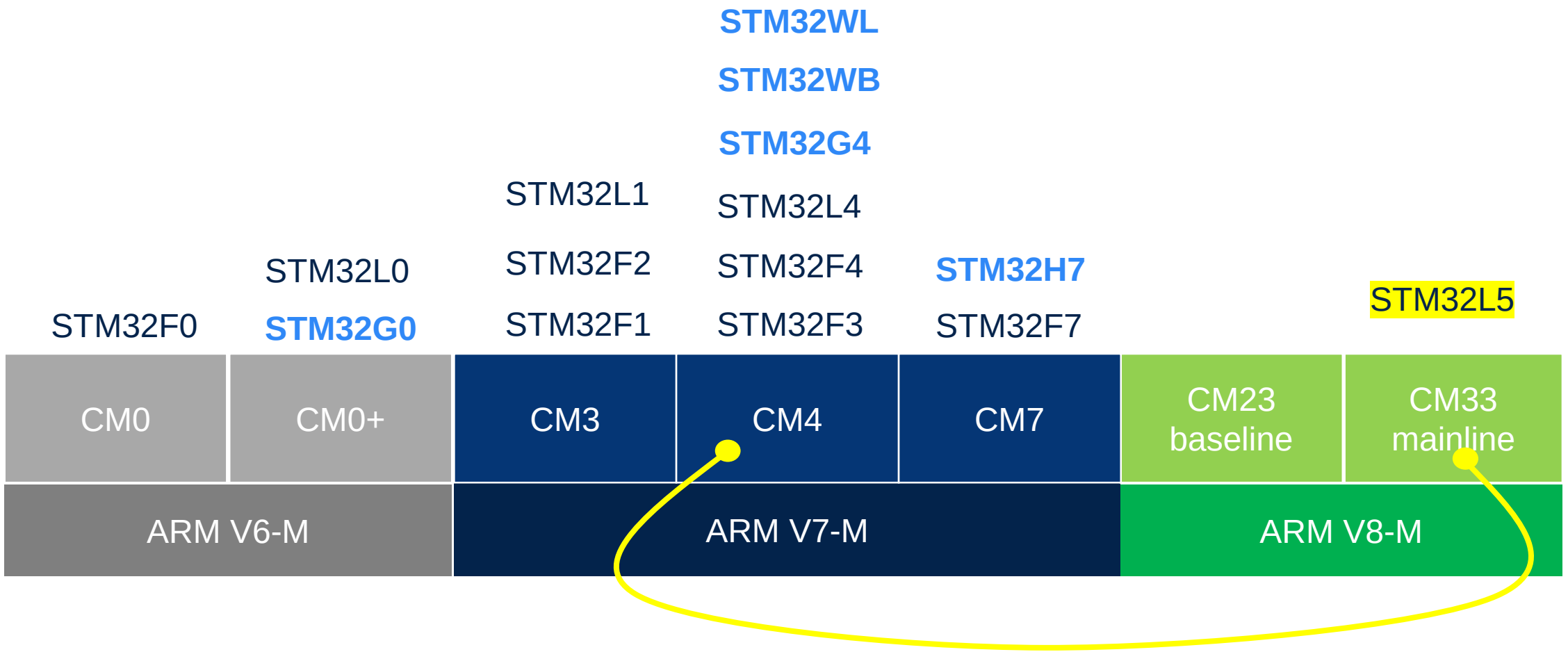
03. 从Cortex-M33内核认识TrustZone

Lilian YAO

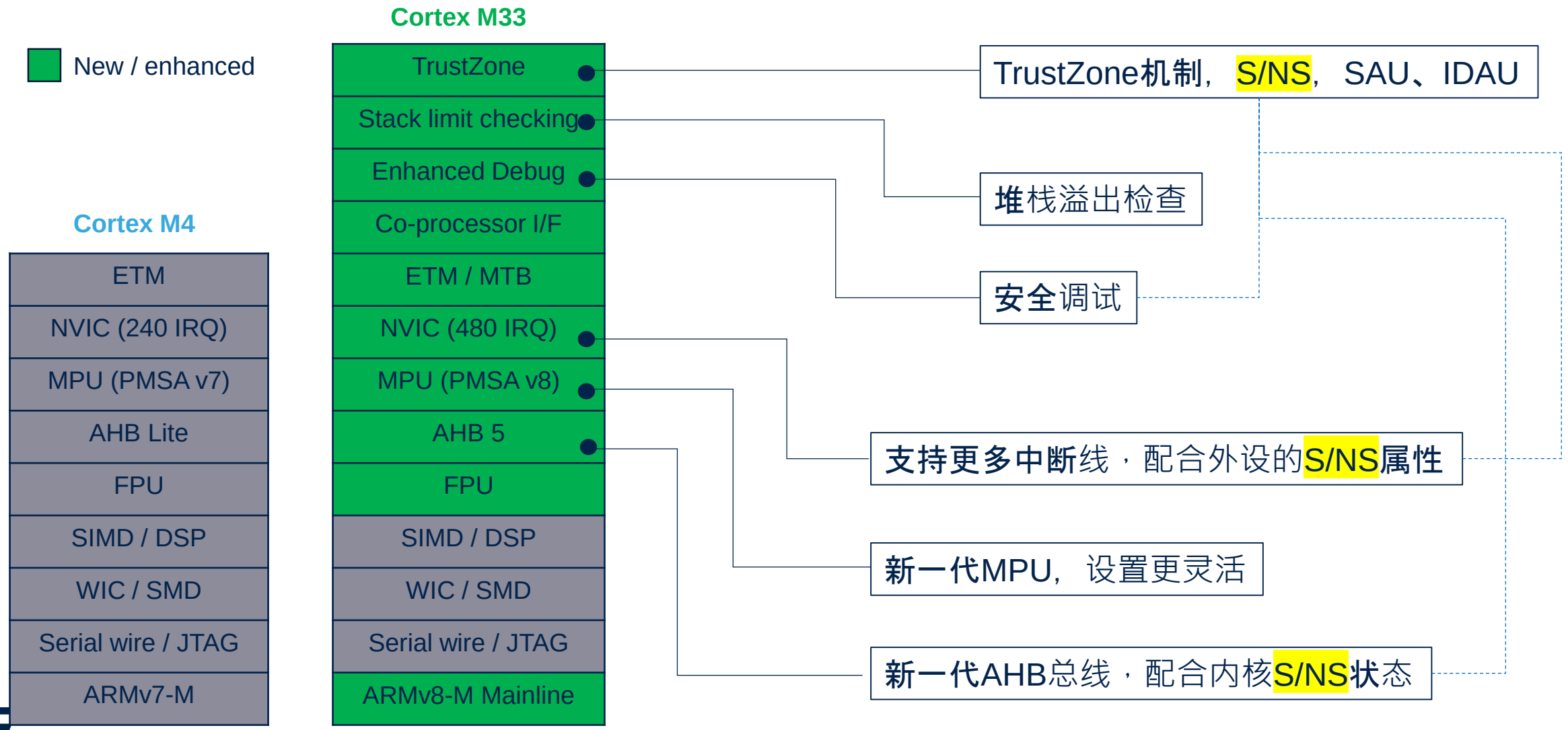
课程内容一览

STM32L5快速入门		
课程章节名称	章节介绍	时长
03. 从Cortex-M33内核认识TrustZone	从内核角度认识TrustZone，包括为了支持TZ功能而在内核里新增的外设，以及原有内核外设为了支持该功能的更新	40 分钟
04. STM32L5的系统新架构	光有内核的TrustZone是不够的，STM32L5把“隔离”的概念和措施从内核延伸出来，部署到了全片系统	20 分钟
05. TrustZone环境下新的用户编程模型	通过GPIO toggle例程，体会新的用户编程模型，深入理解Cortex-M33内核和STM32L5外设对TZ的支持和使用	30 分钟
06. STM32CubeMX对TrustZone环境的支持	使用STM32CubeMX在STM32L5上生成TZ应用初始代码及其框架，添加用户应用逻辑，实现前三节课里讲的知识点	20 分钟
07. STM32L5的低功耗特性	理解L5的供电策略、各种低功耗模式，内置DC/DC SMPS的使用，以及外设对于低功耗方面的设计考虑，BAM模式介绍	

STM32L5 基于ARM Cortex-M33内核



Cortex-M33 在Cortex-M4 基础上的功能扩展



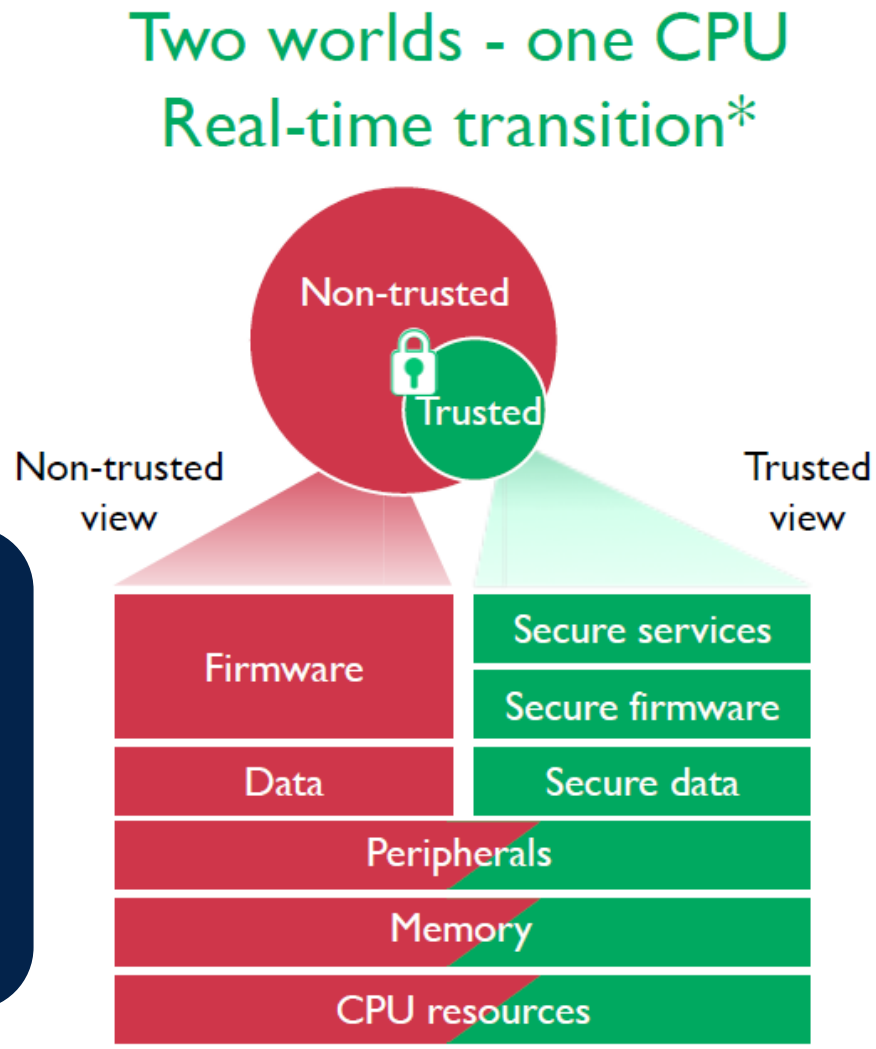
life.augmented

什么是（ARMv8-M架构下的）TrustZone

- 一颗CPU上，硬件隔离出两个世界
- 安全世界里的安全软件
  - 安全相关的敏感操作、审查过的代码
  - 访问放置在安全世界里的敏感资源（数据、外设）
- 安全世界里的安全硬件
  - 加解密相关的硬件、控制关键操作的外设
- 非安全世界里的普通代码、普通硬件
- 两个世界间的切换是实时的

如何设置两个不同的世界？

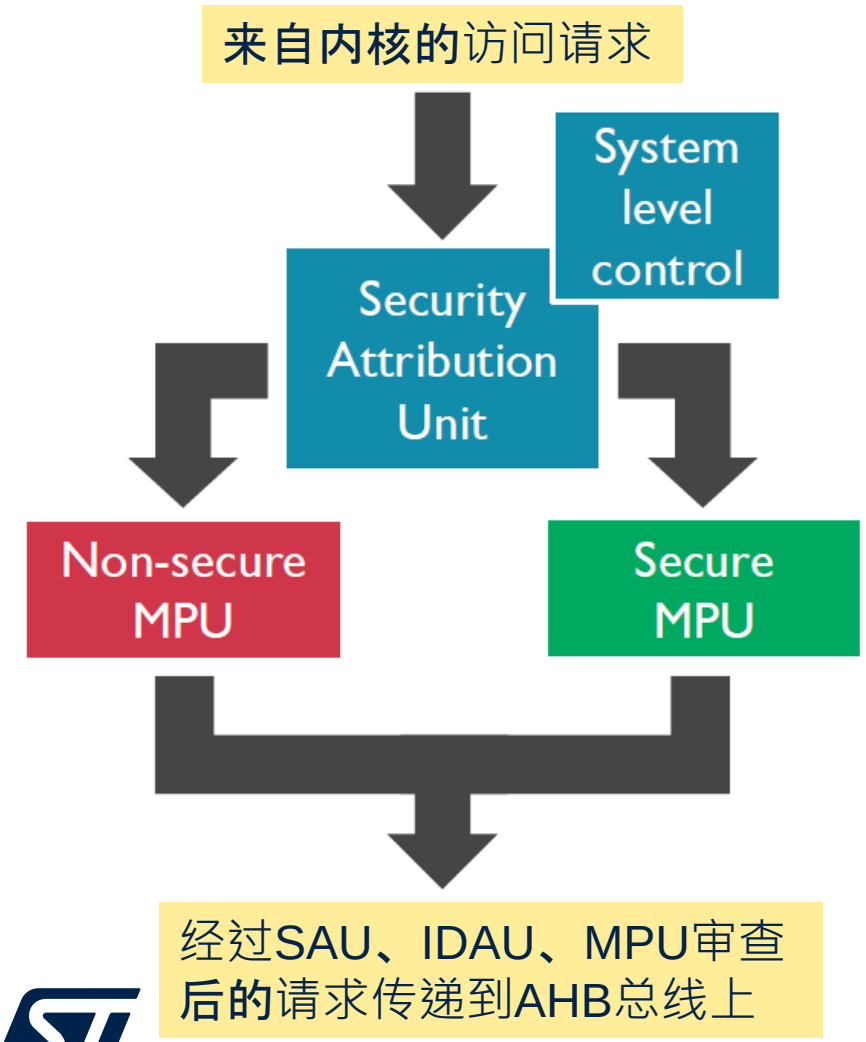
如何在两个世界间切换？

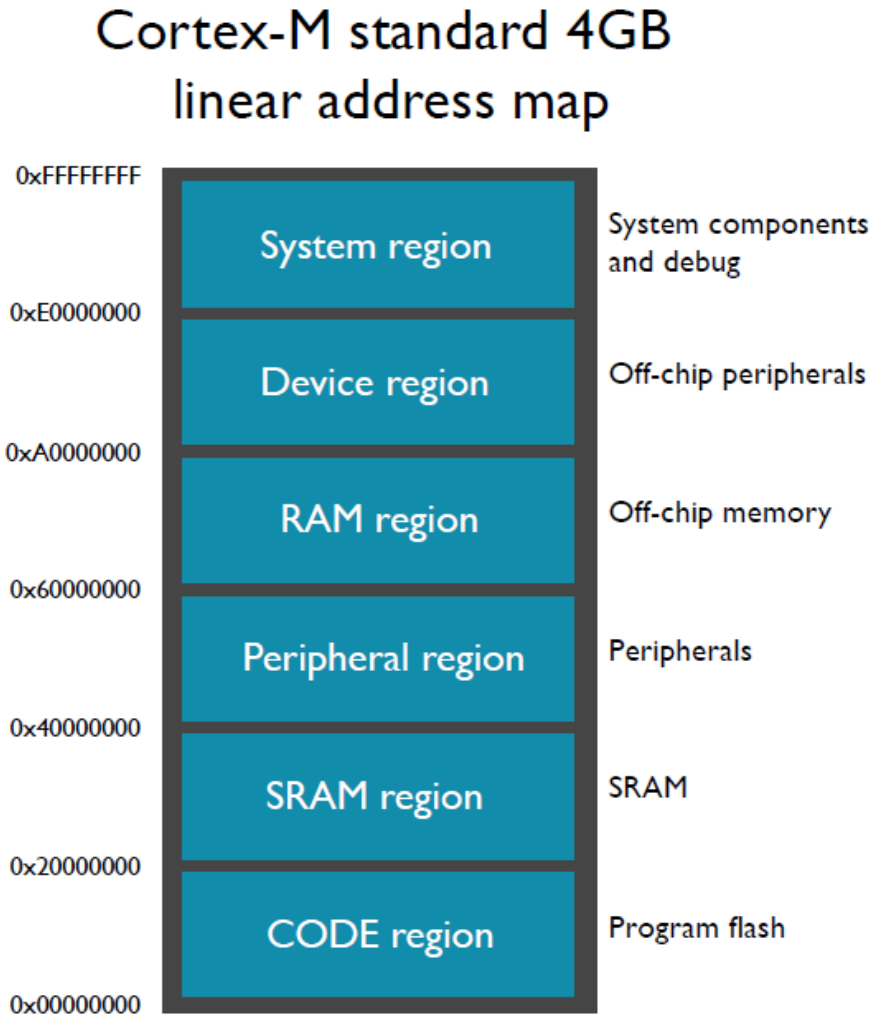


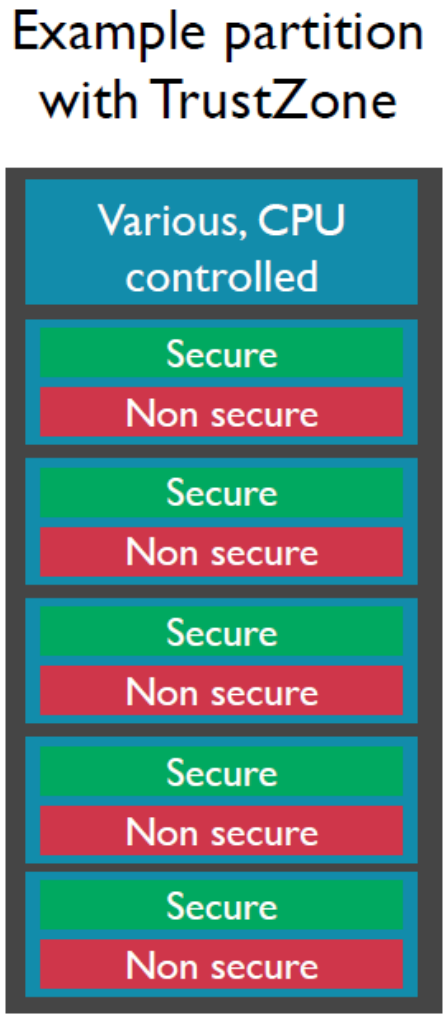
life.augmented

Q：如何设置两个不同的世界？
A：通过SAU和IDAU

按照地址来划分 安全世界 vs. 非安全世界



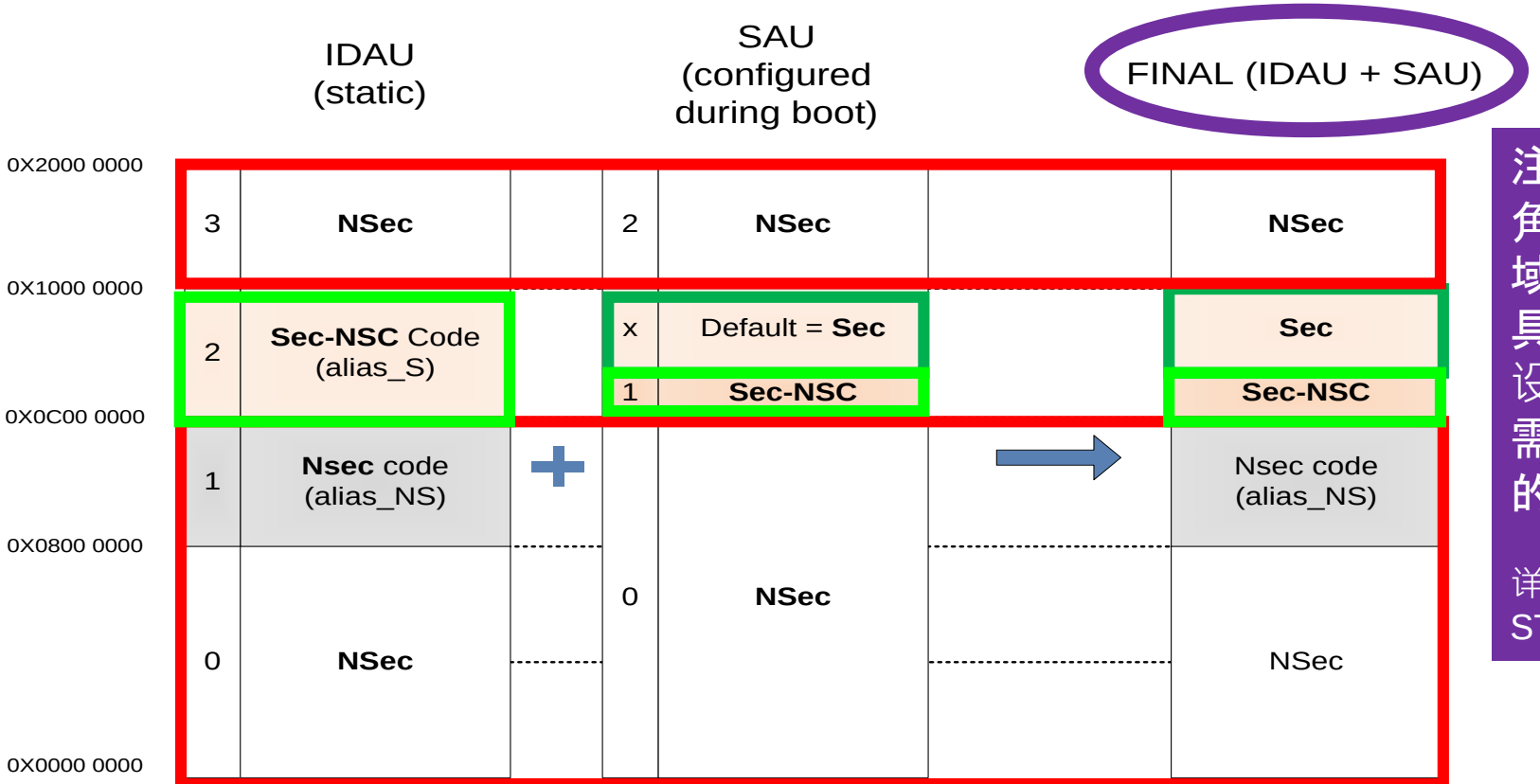




life.augmented

通过SAU和IDAU划分 安全世界 和 非安全世界

- IDAU (Implementation Defined Attribution Unit)
  - **配置区域的属性**：Secure、NSC、Non-Secure
  - **区域大小粒度**：64MB//0x400 0000
  - **静**态配置：上电即生效
- SAU (Security Attribution Unit)
  - （STM32L5）**可**对**8个区域**进行设置
  - 动态配置：上电后可由用户修改配置
  - **二者共同定**义的区域，取高的安全级

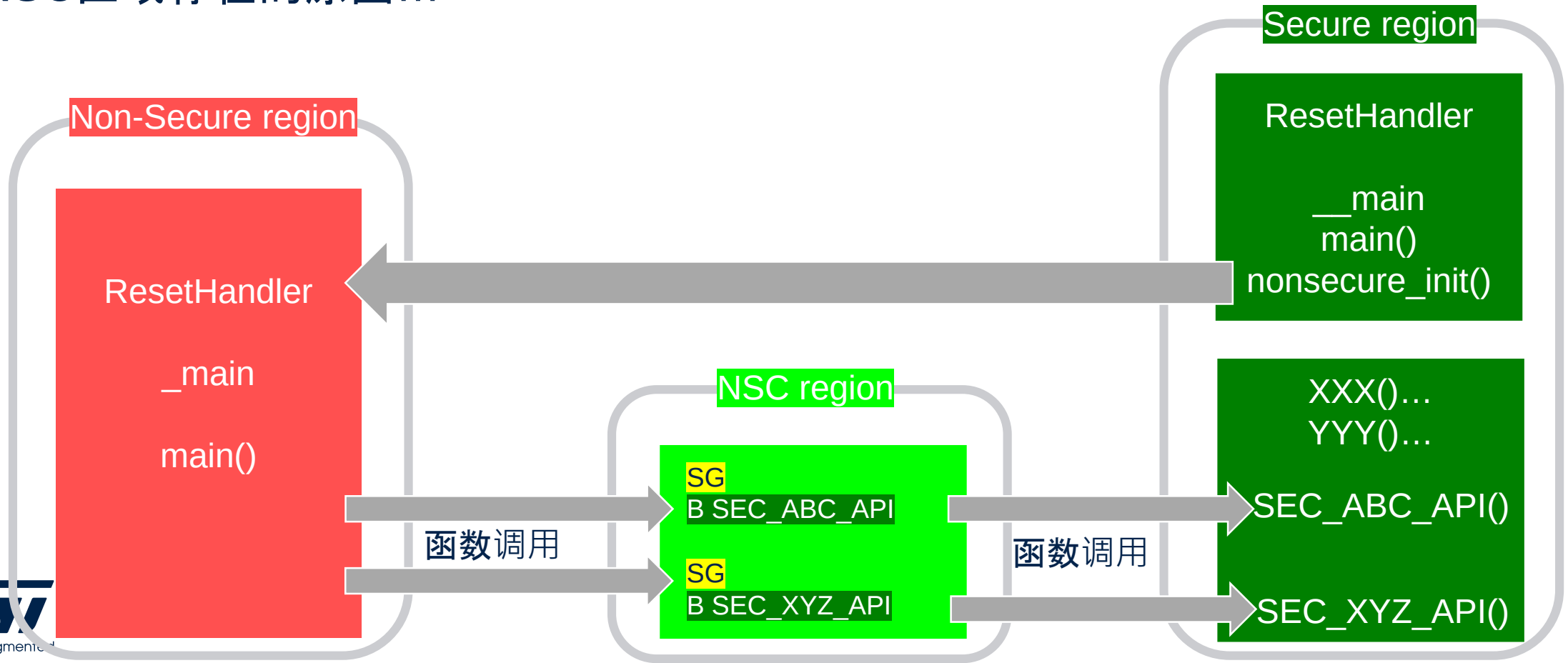


注意：这是从内核角度看到的每个区域的S和NS属性，具体memory和外设的安全属性，还需要看STM32L5上的用户配置如何…

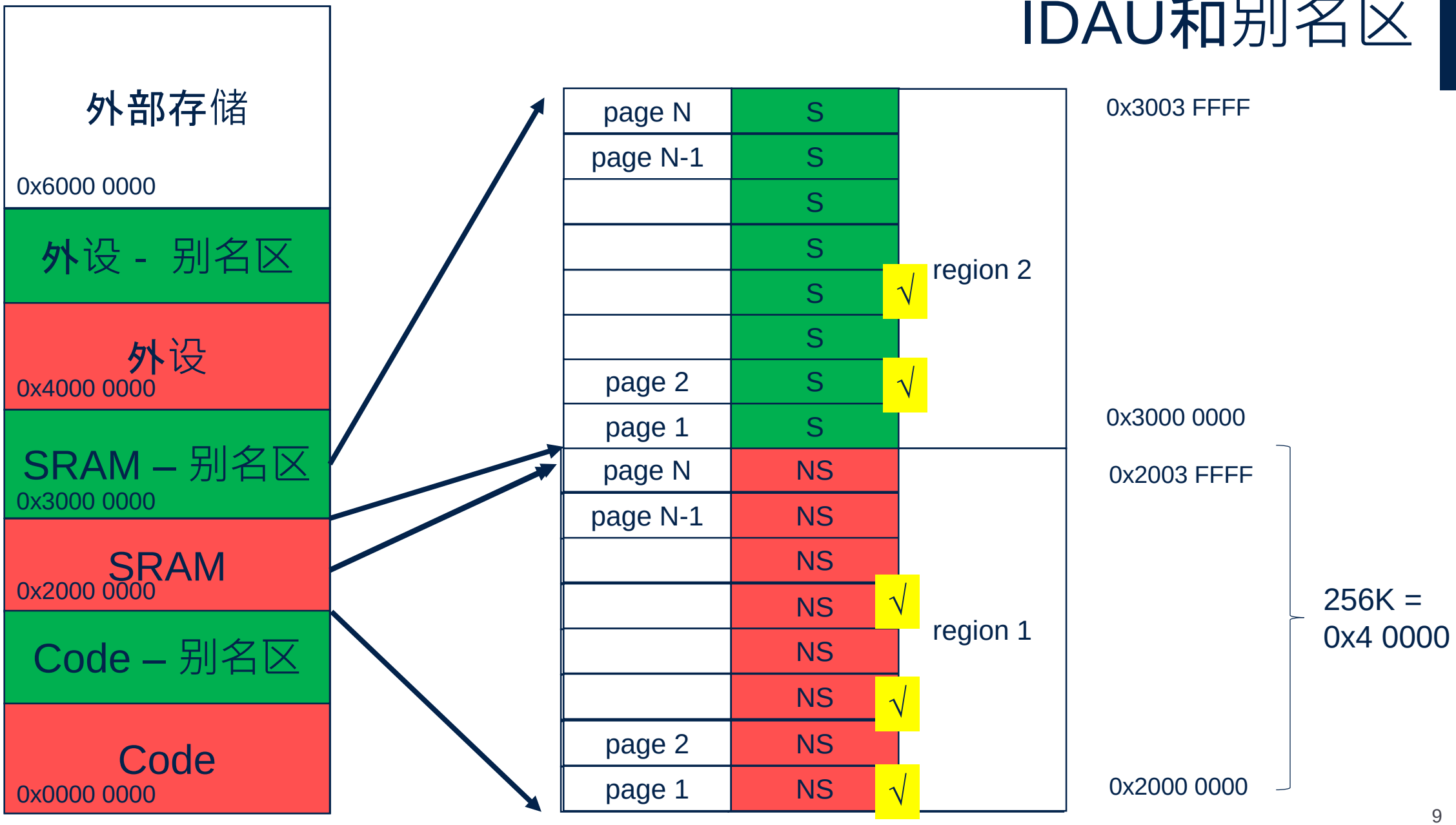
详情参见下一课《04. STM32L5系统架构》

（区域的）三种安全属性

- IDAU/SAU设置区域的安全属性： Secure、NSC、Non-Secure
- NSC区域存在的原因…

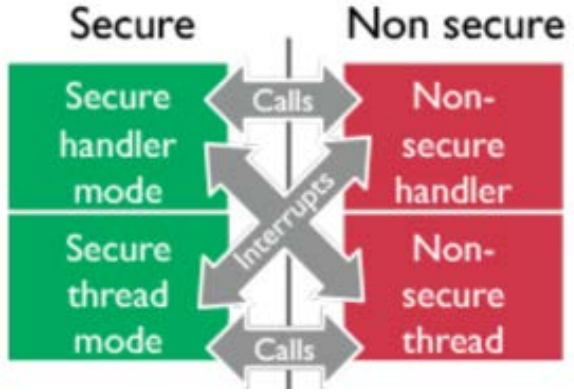


IDAU和别名区

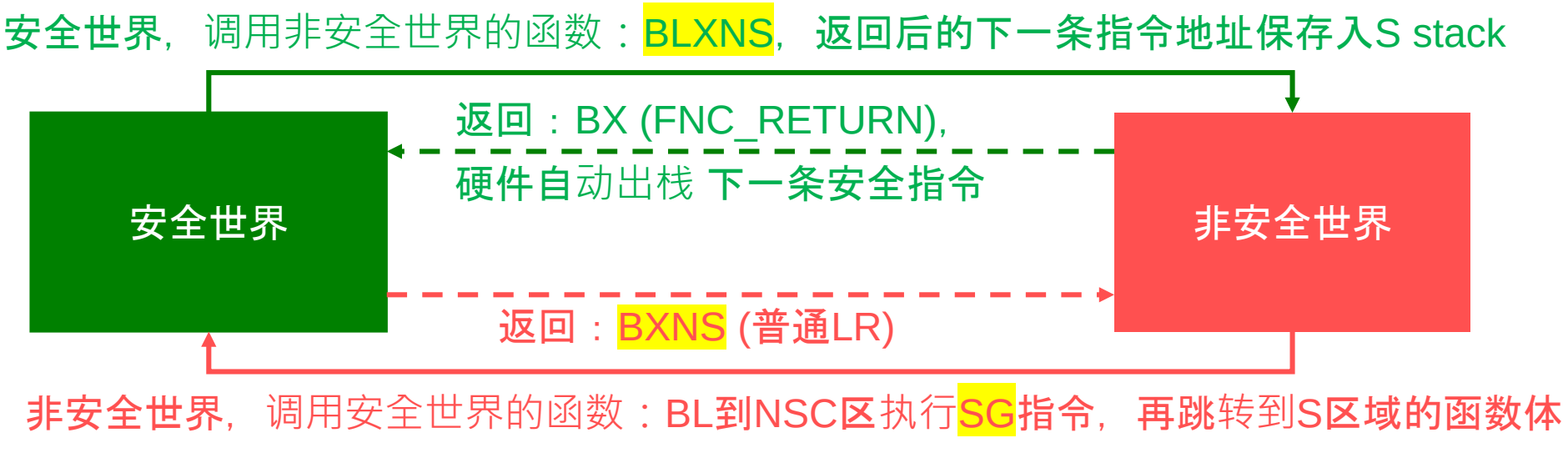

外部存储
0x6000 0000
外设 - 别名区
外设
0x4000 0000
SRAM – 别名区
0x3000 0000
SRAM
0x2000 0000
Code – 别名区
Code
0x0000 0000
page N
page N-1
page 2
page 1
S
NS
√
region 2
region 1
0x3003 FFFF
0x3000 0000
0x2003 FFFF
0x2000 0000
256K =
0x4 0000

Q：如何在两个世界之间切换？
A：对于函数调用 - 新指令、新标志、软硬件配合完成

状态切换．函数调用



- 新指令支持 安全世界 和 非安全世界 之间的函数调用
  - SG、BXNS、BLXNS
- 新操作
  - 软件，清除安全世界的通用寄存器内容
  - 硬件，自动将下一条安全指令入栈到S stack，FNC_RETURN赋值给LR

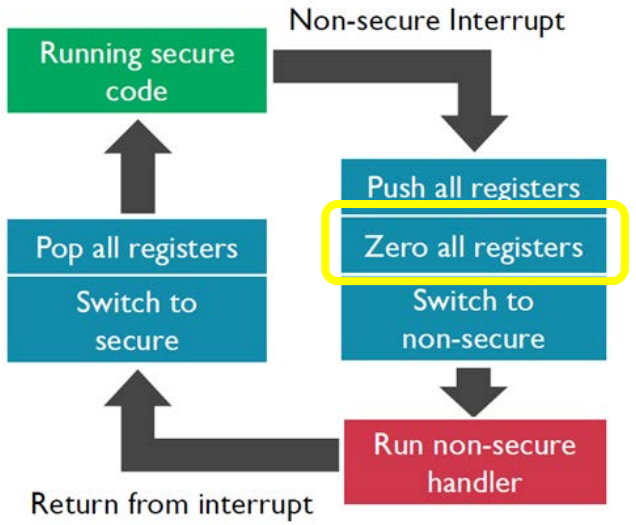


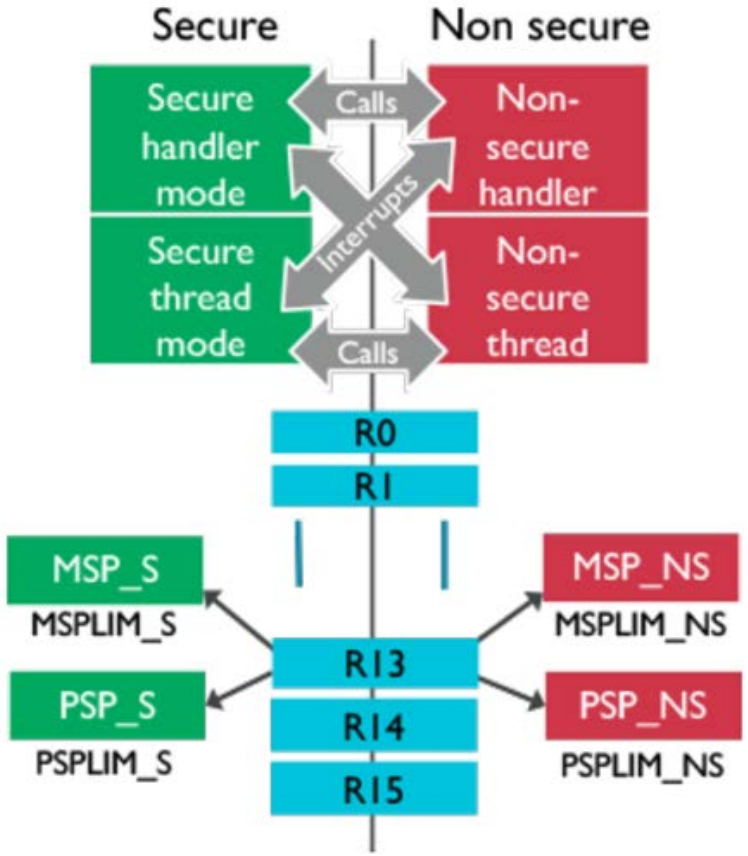
life.augmented

状态切换．中断

Q：**如何在两个世界之**间切换？
A：对于中断 – **自由切**换，硬件入栈上下文并条件性地清零现场

- **只要**优先级许可，彼此可以相互被打断
- **寄存器**值自动入栈
  - **被打断的代**码，当前执行的上下文（通用寄存器）被硬件**入**栈到对应的堆栈
    - **安全世界的堆**栈 （MSP_S、PSP_S）
    - **非安全世界的堆**栈（MSP_NS、PSP_NS）
- **安全代**码被非安全中断打断后
  - **除了硬件自**动入栈寄存器
  - 还会自动清零当前寄存器的内容

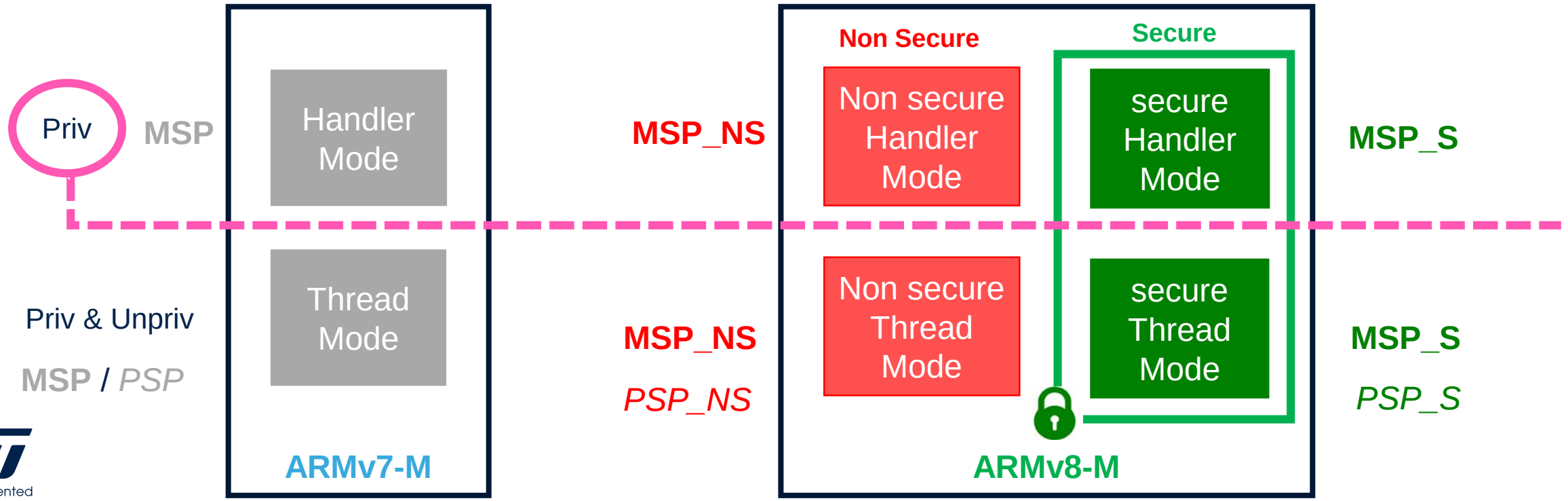




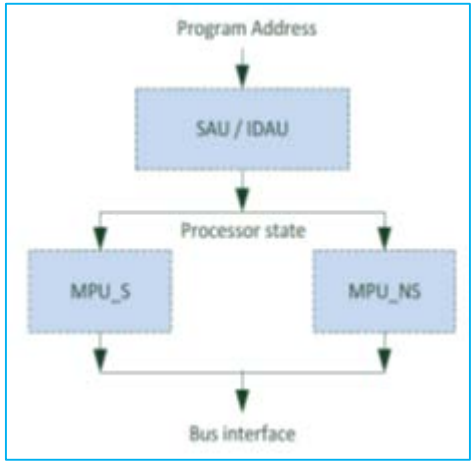
life.augmented

内核所处上下文新增的维度

- 内核所处的 安全**状态**//state：安全 vs. 非安全
- 内核所处的 特权**级别**//level：特权 vs. 非特权
- 内核所处的 处理器**模式**//mode：Handler vs. Thread



安全transaction vs. 非安全transaction



- **内核当前状**态，取决于当前执行代码所在的区域
- **内核当前状**态 **和** 访问目标地址的S/NS**属性，造就了不同属性的** transaction
  - **如果SAU允**许该访问，MPU**会做**进一步检查，决定该transaction**是否有效**

指令访问	**目标地址S**	**目标地址NSC**	**目标地址NS**
	在IDAU/SAU看来		
CPU无论哪种状态	S transaction	S transaction	NS transaction

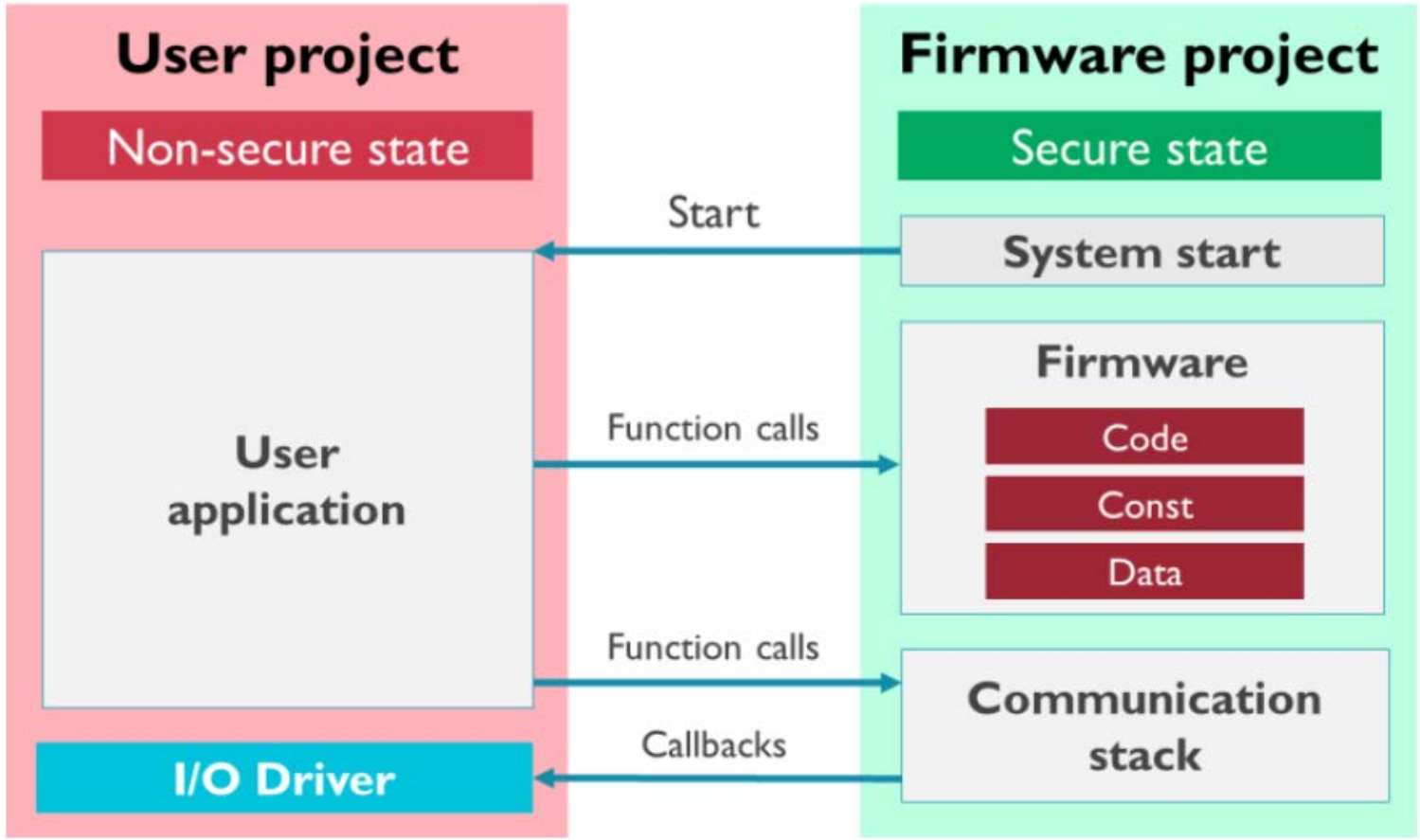
数据访问		**目标地址S**	**目标地址NS**
		在IDAU/SAU看来	
CPU S状态	由当前CPU执行的指令所在的地址区域在IDAU/SAU看来	S transaction	NS transaction
CPU NS状态		Blocked	NS transaction

- MPU的影响
  - **根据**transaction的S和NS**属性，使用**对应MPU（MPU_S、MPU_NS）**管理**
  - **通**过的transaction**才会携**带着“HNONSEC**信号”来到**总线上，最终访问是否被允许还**取决于目**标硬件的检查（参加下一课：L5**系**统安全）

life.augmented

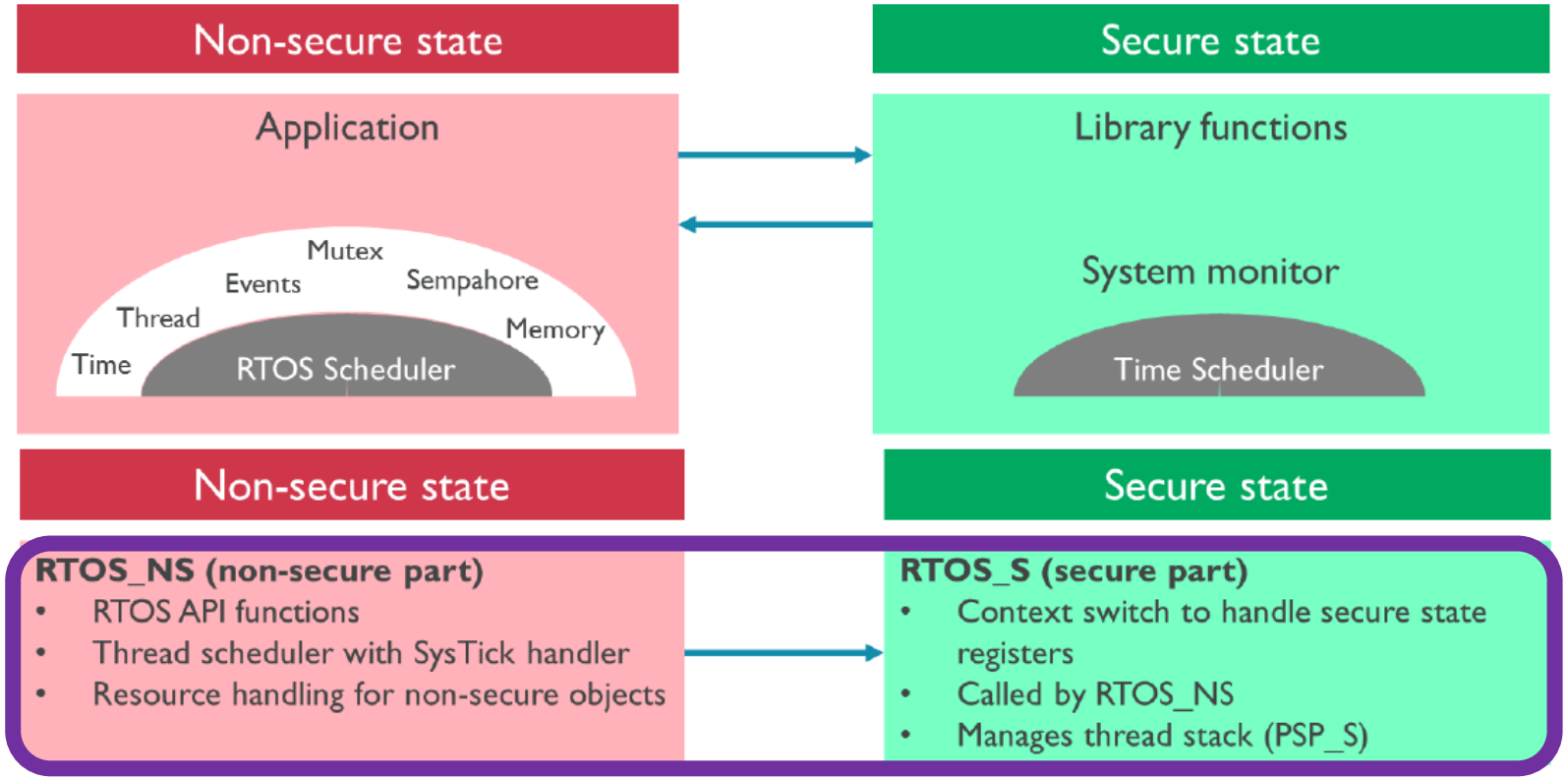
开发者的软件开发模型

- 两个工程
  - 分别运行安全应用和普通应用
  - 复位后先运行安全工程
  - 两个工程间的依赖关系



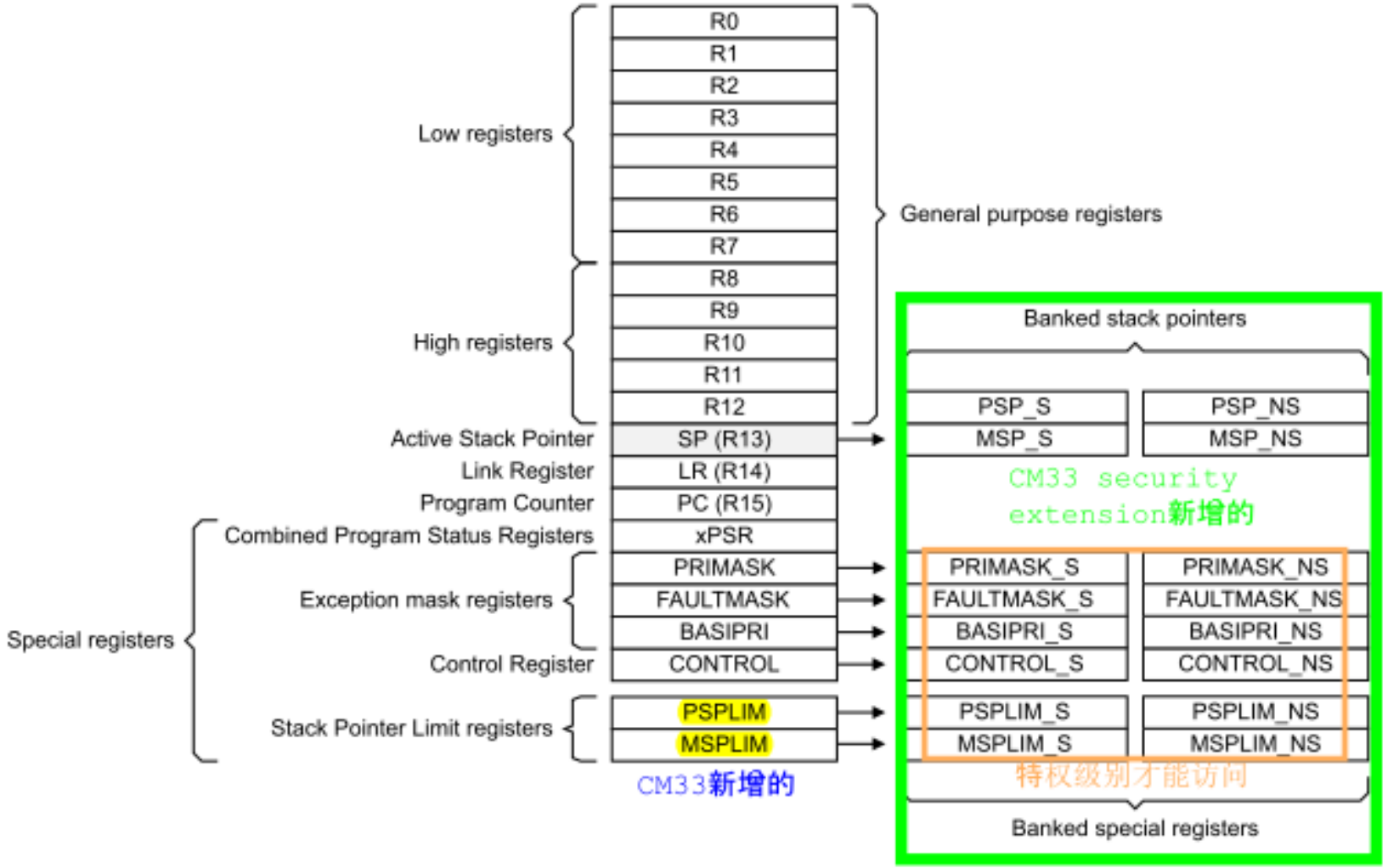
ARMv8-M架构对RTOS的要求

- 两个执行环境：REE和TEE
  - RTOS通常运行在REE
  - RTOS调度的任务，如果使用**了安全世界的**资源，RTOS要负责额外的安全堆栈的分配，**以及安全世界的上下文切**换
  - 在STM32L5上运行FreeRTOS也以往的STM32有什么不同吗？



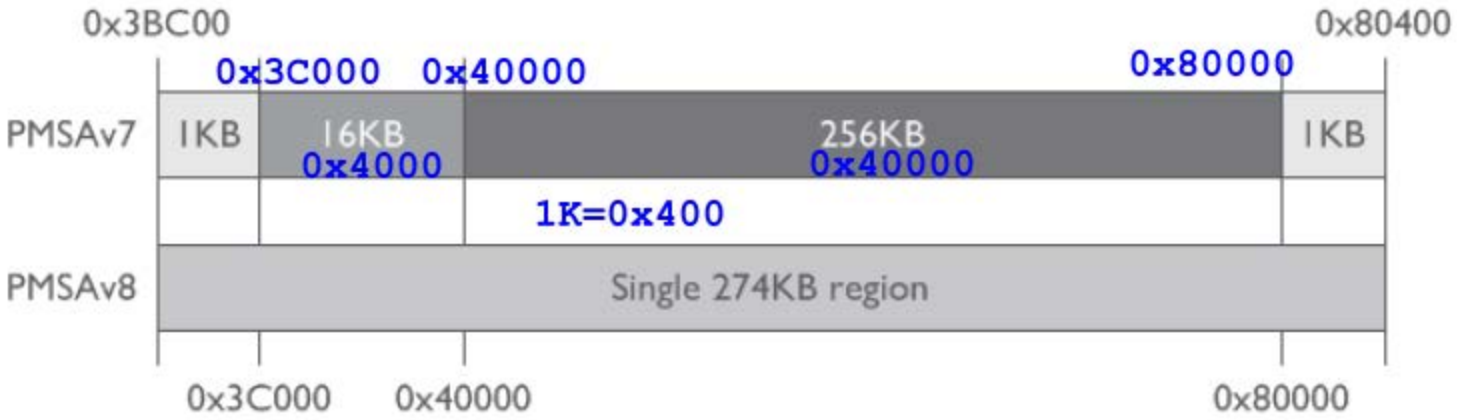
内核寄存器对ARMv8-M和TZ扩展的升级支持

- 通用寄存器
  - 两个世界共享一套
- CM33内核新增寄存器
  - PSPLIM、MSPLIM
  - 两个世界各有一套 (banked)
- 带TZ安全扩展的CM33内核
  - MSP、PSP、CONTROL
  - 异常掩码寄存器组
  - 两个世界各有一套 (banked)



新的MPU，配置更灵活

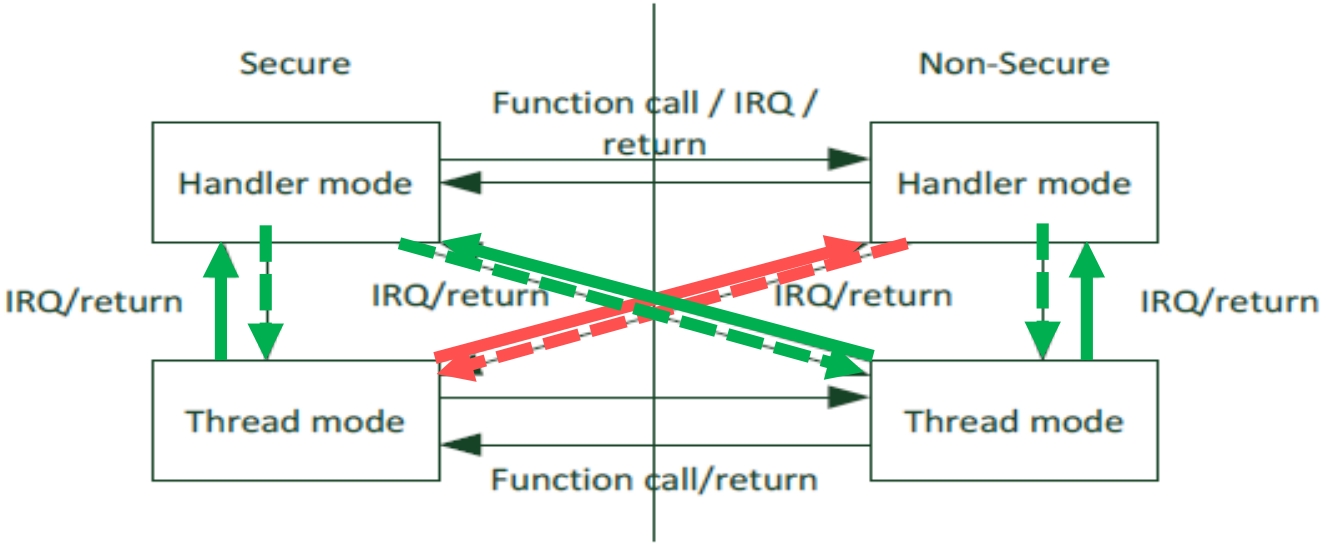
- **区域范**围配置更灵活
  - **以前的MPU，起始地址必**须是该区域大小的整数倍；区域大小要是**2的整次**幂
  - 现在的MPU，PMSA-v8规范：区域起始地址和大小没有限制关系；区域大小只要求**32字**节粒度
  - **寄存器**设置不兼容，对**MPU的**编程需要重写

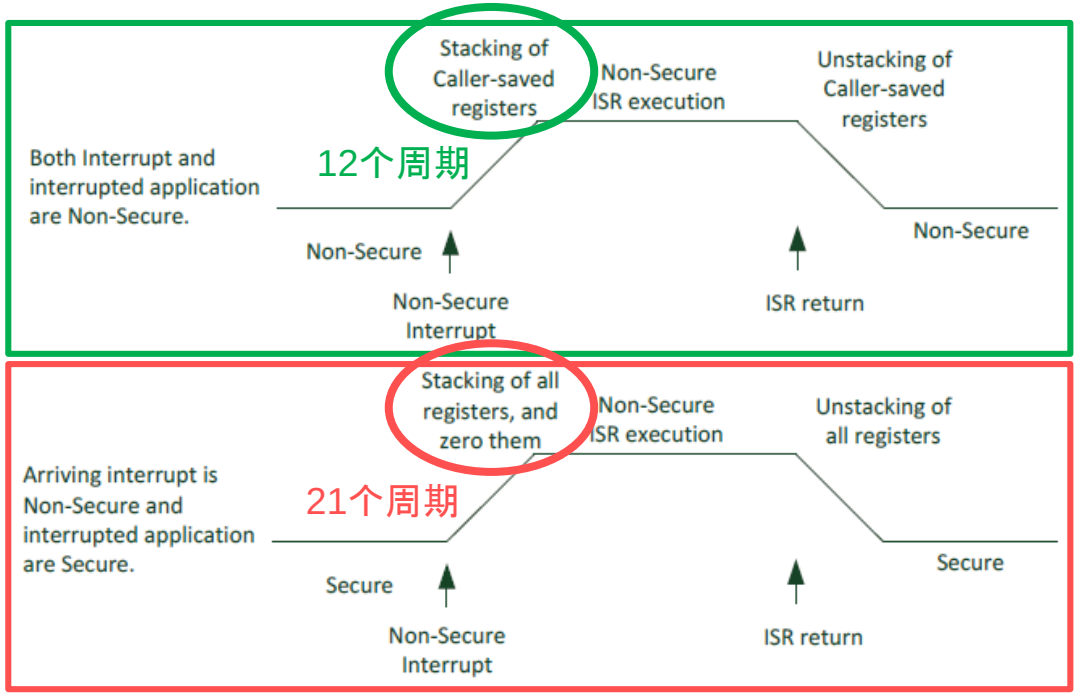


life.augmented

升级的中断模型，更好支持TZ扩展

- 任何中断，都可以由安全世界的代码，将其配置成target to NS or S
- 中断发生时，根据它target到S还是NS，去对应的中断向量表获取ISR地址
  - VTOR_S 和 VTOR_NS
  - 系统复位后，CPU处于安全状态，从VTOR_S中取值，为MSP_S和PC赋值
- 中断带来的S和NS状态切换
  - 12 vs. 21个周期





系统异常和应用中断的安全属性配置

	安全 (上电状态)	非安全	S/NS的条件	使能	是否 active	是否 挂起	软件挂起 和清除	软件 触发
Reset								
NMI	0	1	BFHFNMINS				ICSR	
HardFault								
MemManagement	banked							
BusFault	0	1	BFHFNMINS					
UsageFault	banked							
SecureFault								
SVCall	banked							
DebugMonitor								
PendSV	banked						ICSR	
Systick	banked							
IRQn	0	1	NVIC_ITNSi (i=0~15)	NVIC_ISERi NVIC_ICERi	NVIC_IABRi		NVIC_ISPRi NVIC_ICPRi	STIR

SHCSR
四小Fault

SHCSR
(System handler control and status register)
十大系统异常

SHCSR

系统异常和应用中断的优先级

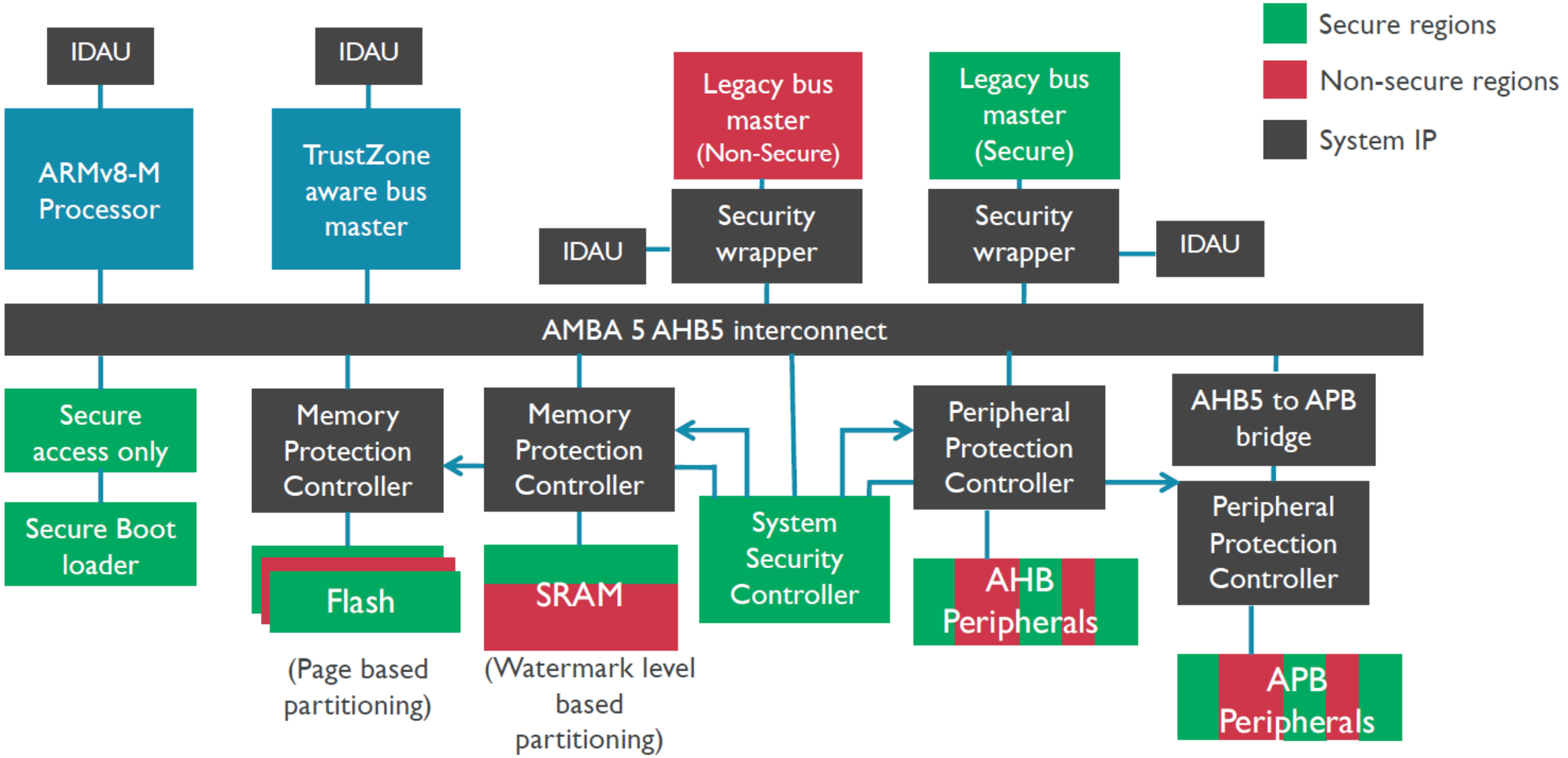
	安全 (上电状态)	非安全	S/NS的条件	优先级	AIRCR. BFHFNMINS =1
Reset				-4	
NMI	0	1	BFHFNMINS	-2	
HardFault				-1	-3
MemManagement	banked				
BusFault	0	1	BFHFNMINS		
UsageFault	banked				
SecureFault					
SVCall	banked				
DebugMonitor					
PendSV	banked				
Systick	banked				
IRQn	0	1	NVIC_ITNS	NVIC.IPR0~119	

优先级可配置

SHPR1~SHPR3
(System handler priority)

AIRCR.PRIS

TZ的安全特性，还需从内核扩展到整个片上系统



Thank you

life.augmented