STM32H7 level -1入门培训

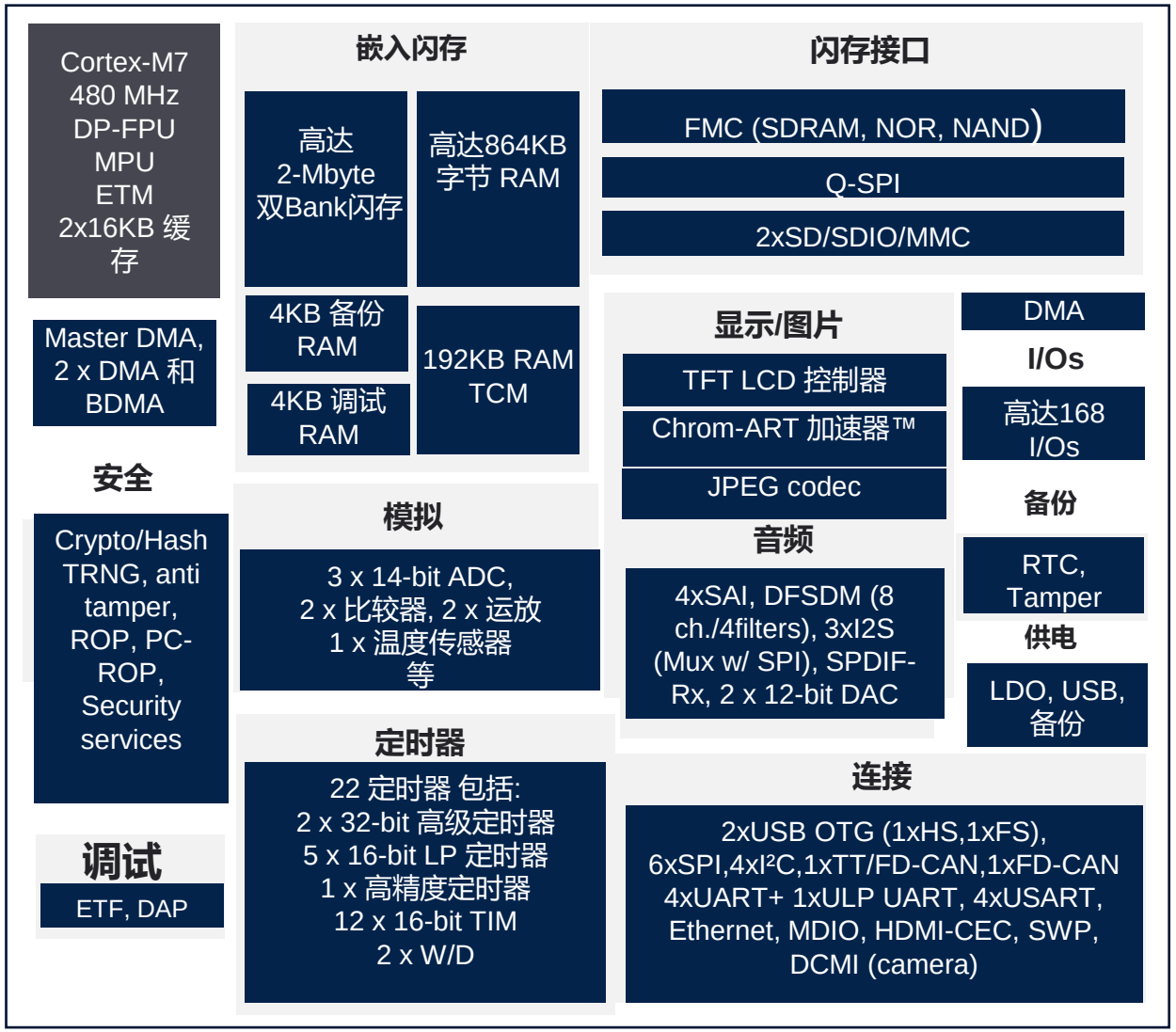
https://www.st.com/content/st_com/en/support/learning/stm32-education/stm32-online-training/stm32h7-online-training.html

V1.0

STM32H7-系统概览

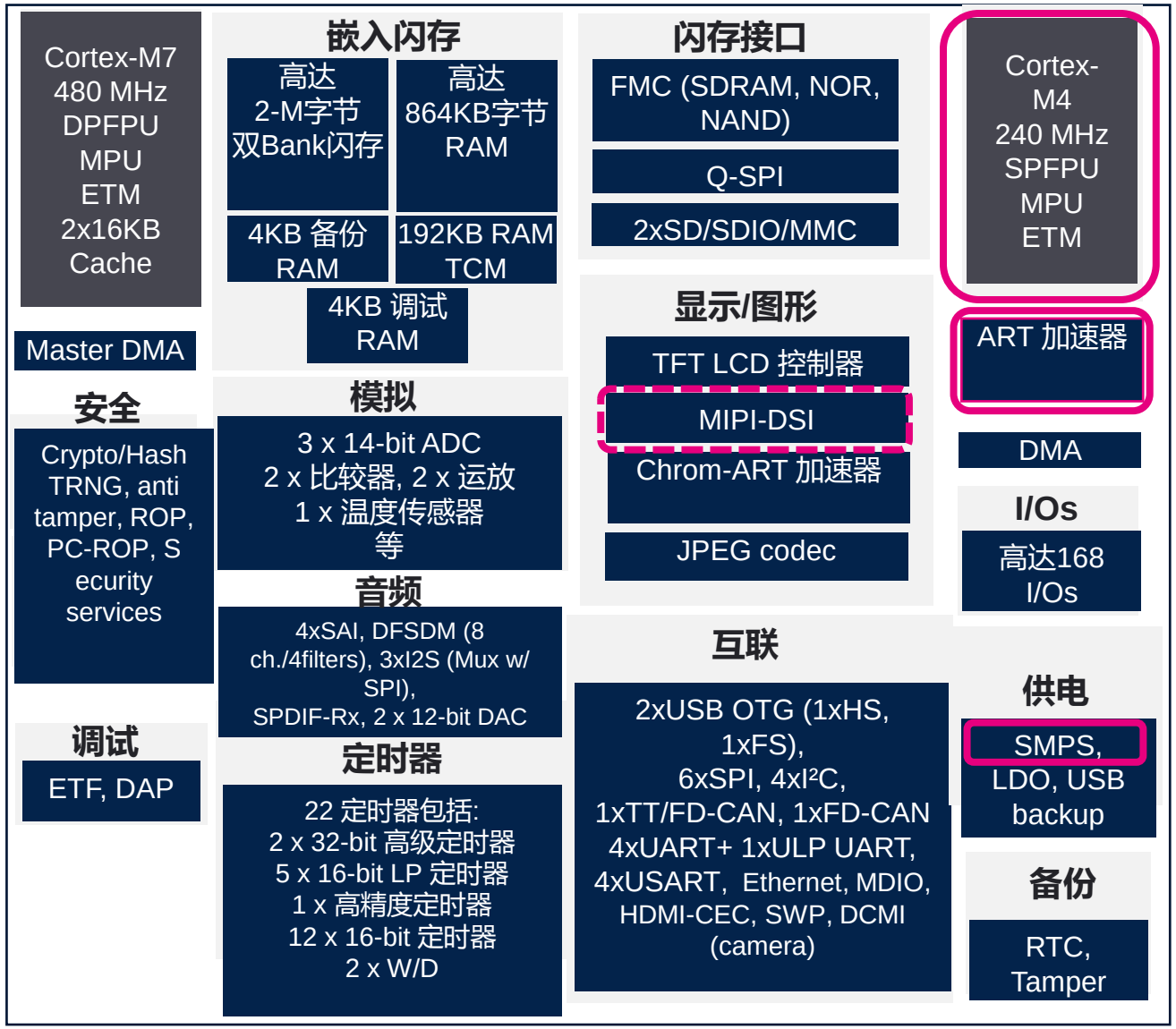
STM32H7x3 框架

- 它包含哪些资源?
  - Cortex®-M7 内核，高达480 MHz
    - 16 K字节 I-$&D-$
    - 16 MPU 单元
    - 单精度和双精度浮点单元
  - 3x 互联矩阵
  - 4 DMA 控制器用于减轻CPU负荷
  - 高达35个通信外设
  - 11模拟外设
  - 高达22个定时器和看门狗
  - 增强的时钟管理



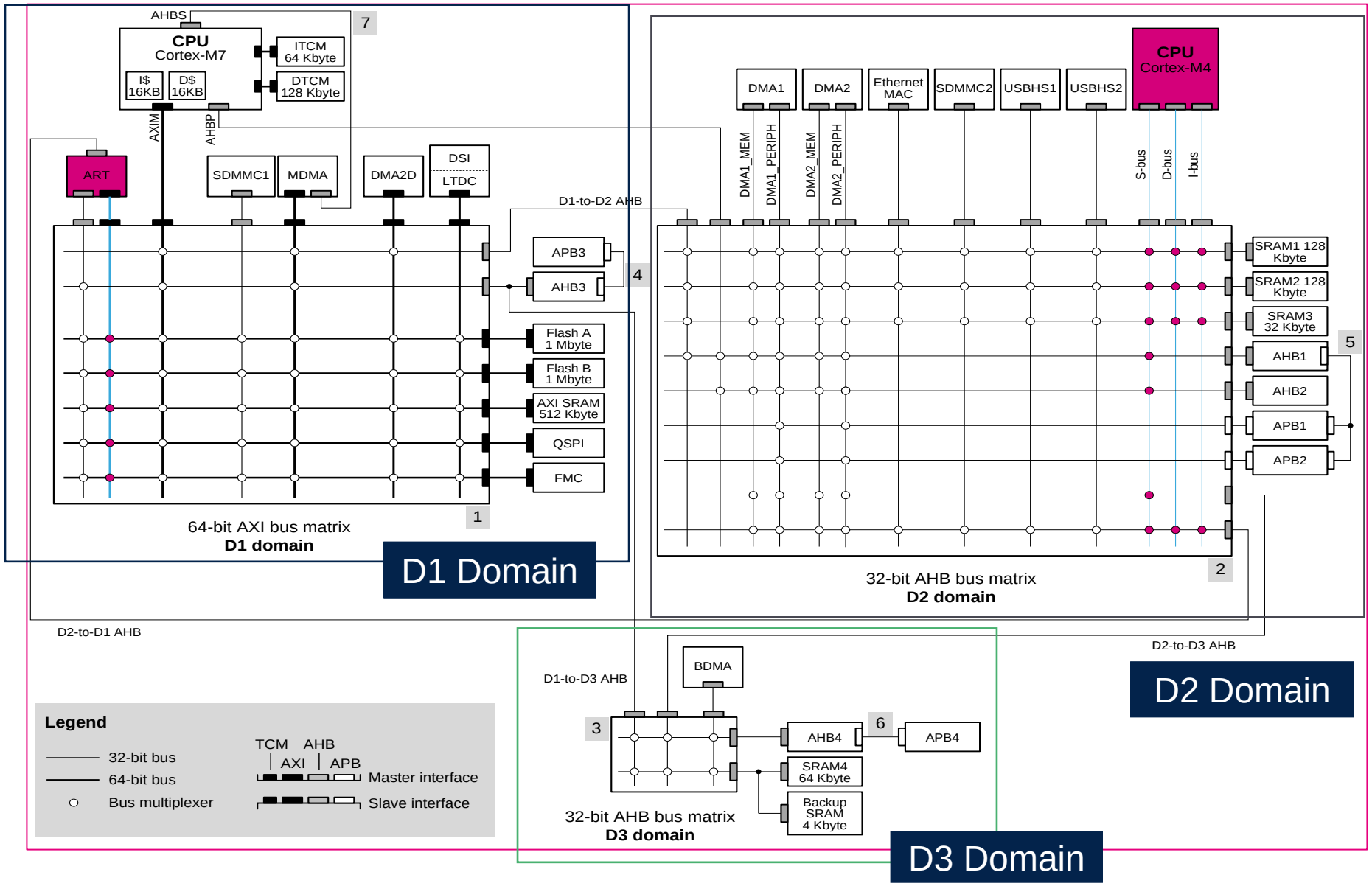
STM32H7x5/x7 框架

- 它包含哪些资源
  - Cortex®-M7 core 高达480MHz
  - Cortex®-M4 core 高达240 MHz
  - 3 个互联矩阵 matrix
  - 4 DMA控制器 用于减轻CPU的负荷
  - 高达35个通信外设
  - 11 个模拟外设
  - 高达22定时器和看门狗
  - 加强的时钟管理
  - 图形显示
    - MIPI DSI host (仅存在于STM32H7x7)



life.augmented

系统架构概览



life.augmented

系统架构概览：总线级联

大量互联可能性	超过20个总线主控

• 总线矩阵
  • D1域中的AXI总线矩阵
  • D2域和D3域中的AHB总线矩阵
• 总线-总线之间的桥接总线
  • 5 个 AHB-to-APB, 2 个 AXI-to-AHB
• 内部域之间的互联
  • D2-to-D1 AHB
  • D1-to-D2 AHB
  • D1-to-D3 AHB
  • D2-to-D3 AHB
	• CPU(s) 总线
	• 高性能外设
	  • SDMMC1, SDMMC2
	  • MDMA 控制器
	  • DMA1 和 DMA2 控制器
	  • BDMA 控制器
	  • Chrom-Art Accelerator™ (DMA2D)
	  • LCD-TFT 控制器 (LTDC)
	  • USBHS1 和 USBHS2 外设

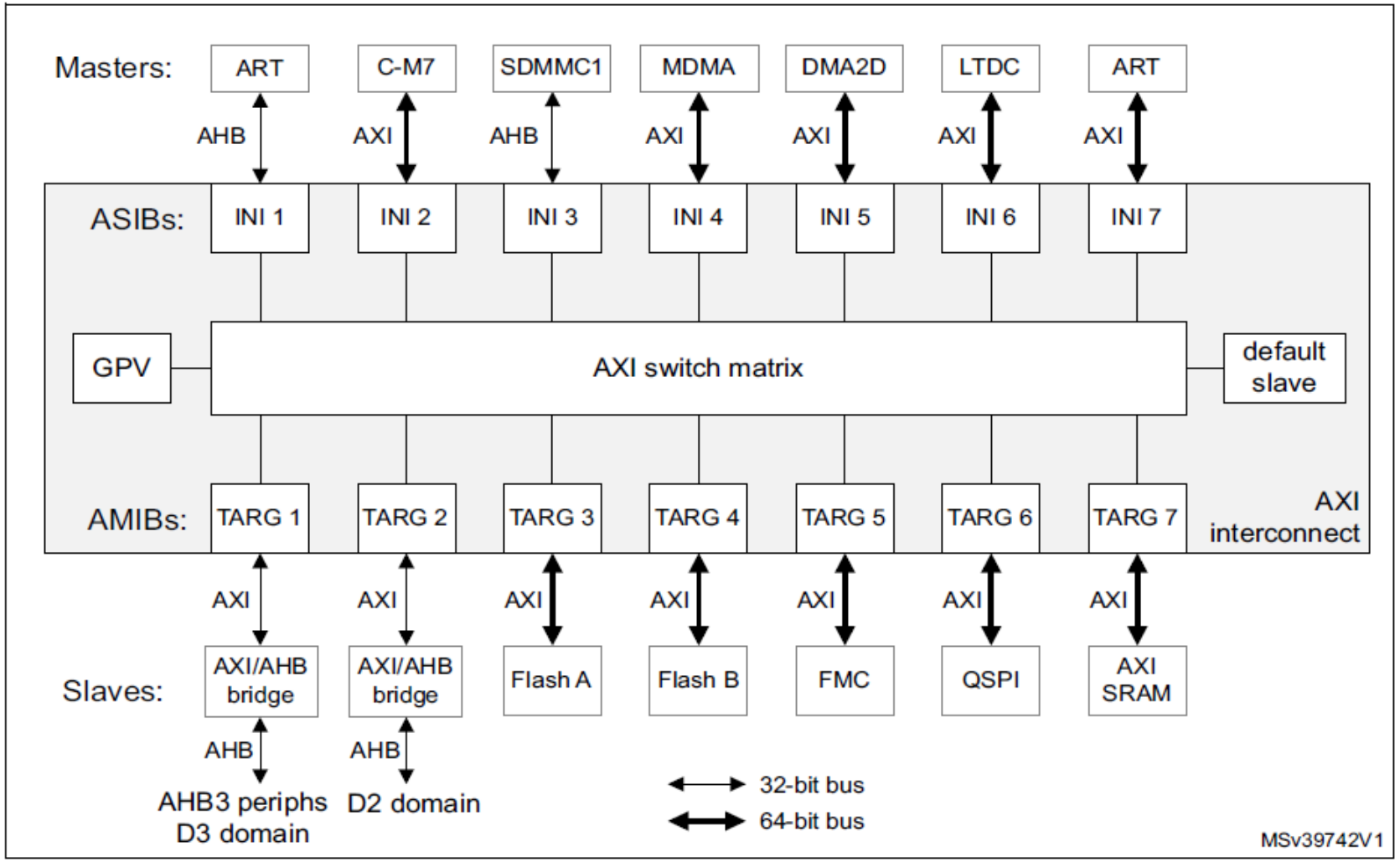
STM32H7xx - AXI 级联

AXI级联

2020年10月7日星期

模块框图

Figure 2. AXI interconnect



主要特点

- 当两个AXI主控同时尝试访问同一AXI从设备时，基于优先级实现仲裁。
- AXI读通道和写通道优先级可以独立配置
  - 可配置的优先级值从0~15
    - 值越高，优先级越高.
  - 如果两个同时发生的事务到达同一个AMIB, 则优先级较高的事务在优先级较低的事务之前通过开关矩阵.
  - 如果两个事务具有相同的QoS值, 那么采用最近最少使用原则 (LRU)去仲裁.
- QoS对于图形处理（LTDC、DMA2D）等任务非常有用，应仔细研究优先权分配对应用的影响
- 如果将对同一从设备进行多次频繁访问的主设备配置为高优先级别，会阻止其他低优先级主设备对该从设备的访问

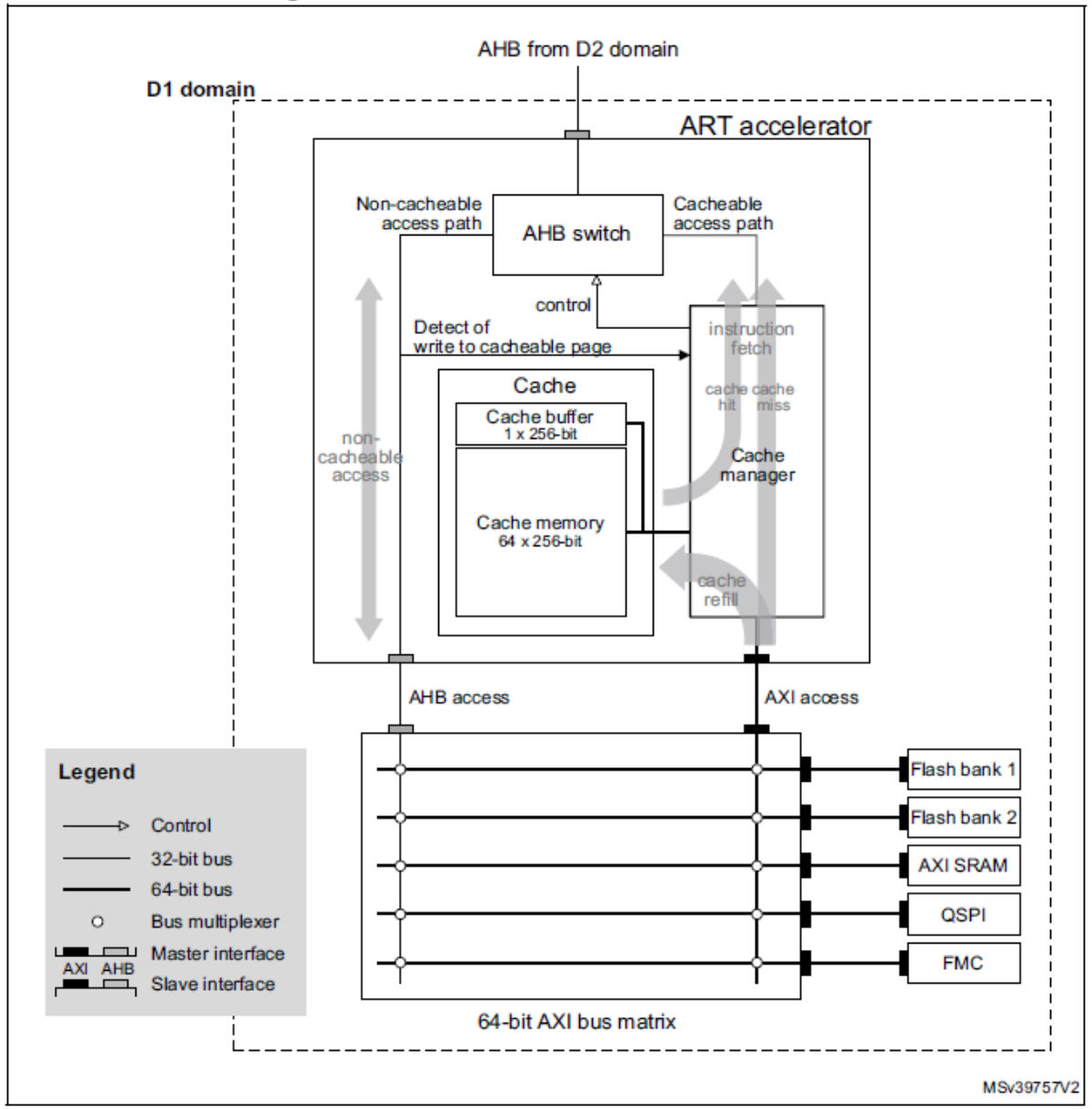
STM32H7xx – ART

ART (Accelerator real-time) 指令缓存

2020/10/7 rev4

ART模块框图

Figure 18. ART accelerator - block schematic



- 32-bit AHB 从接口在 D2 域
- 32-bit AHB 主 接口 用于non-cacheable 内存访问
- 64-bit AXI 主 接口用于将代码从memory加载到cache
- 完全关联的缓存
- 最近最少使用的替换策略
- 带有内容一致性检查器的可编程可缓页

life.augmented

ART 操作

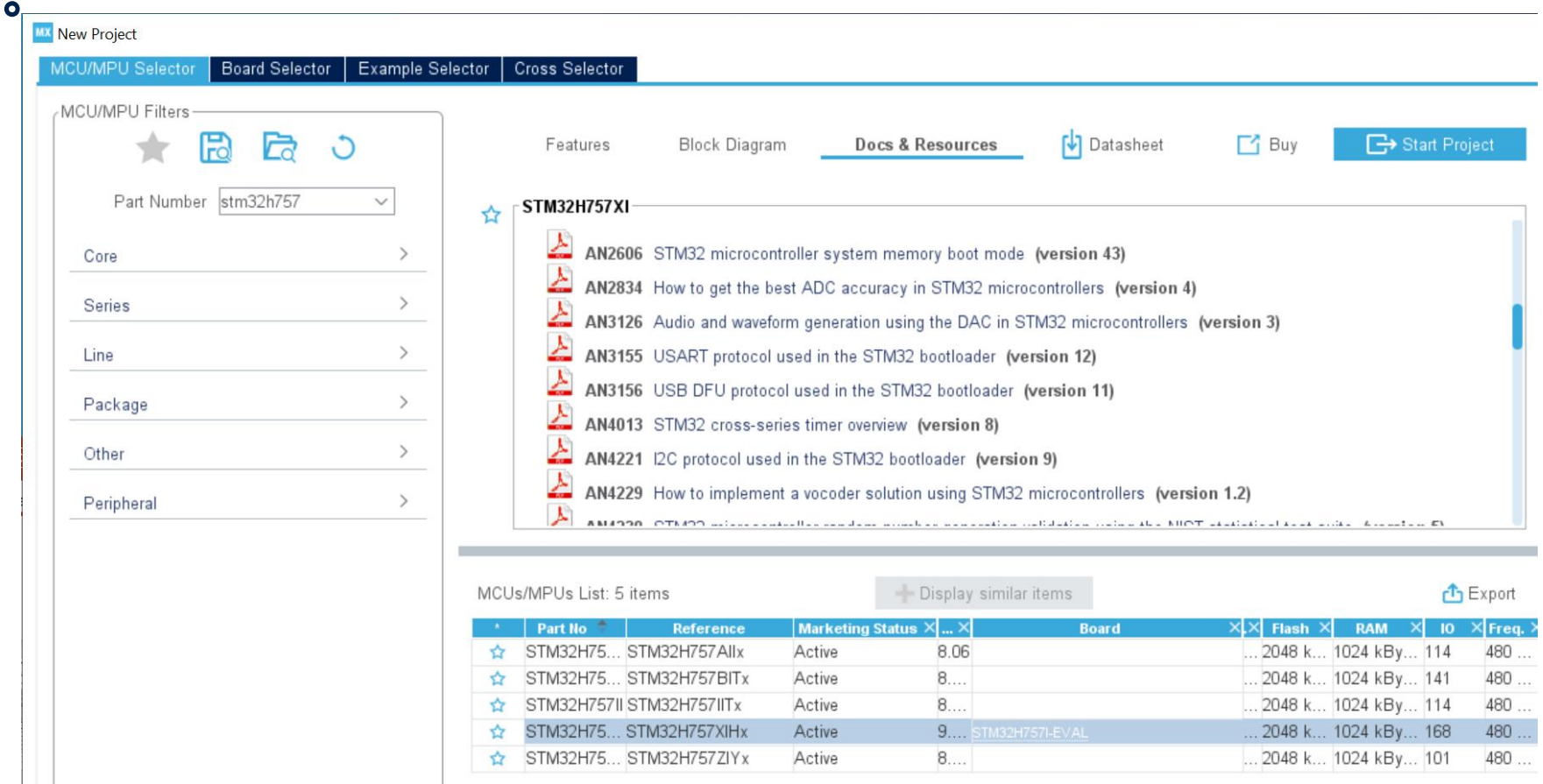
- 为了缓存一致性（缓存中的代码与寻址存储器的相应地址上的代码不一致），如果检测到以下操作之一，则所有cache line都将认为失效：
  - 写访问修改可缓存页中的一段数据
  - 软件修改ART加速器控制寄存器中的可缓存页索引。
  - 在缓存缓冲区和/或缓存线重新填充期间发生数据传输错误

- 该设备采用了一种机制来防止由于高速缓存而导致的软件不一致。它检测上述情况并标记所有缓存线和缓存缓冲区无效。因此，接着的下一个可缓存访问将导致缓存未命中，在随后的缓存缓冲区和缓存线中重新填充新数据。

小结问题

1、能简单看懂架构框图、知道单核与双核之间的区别等。

2、确保安装好了Cube相关工具，会操作，并知道如何快速搜索相关资源文档。

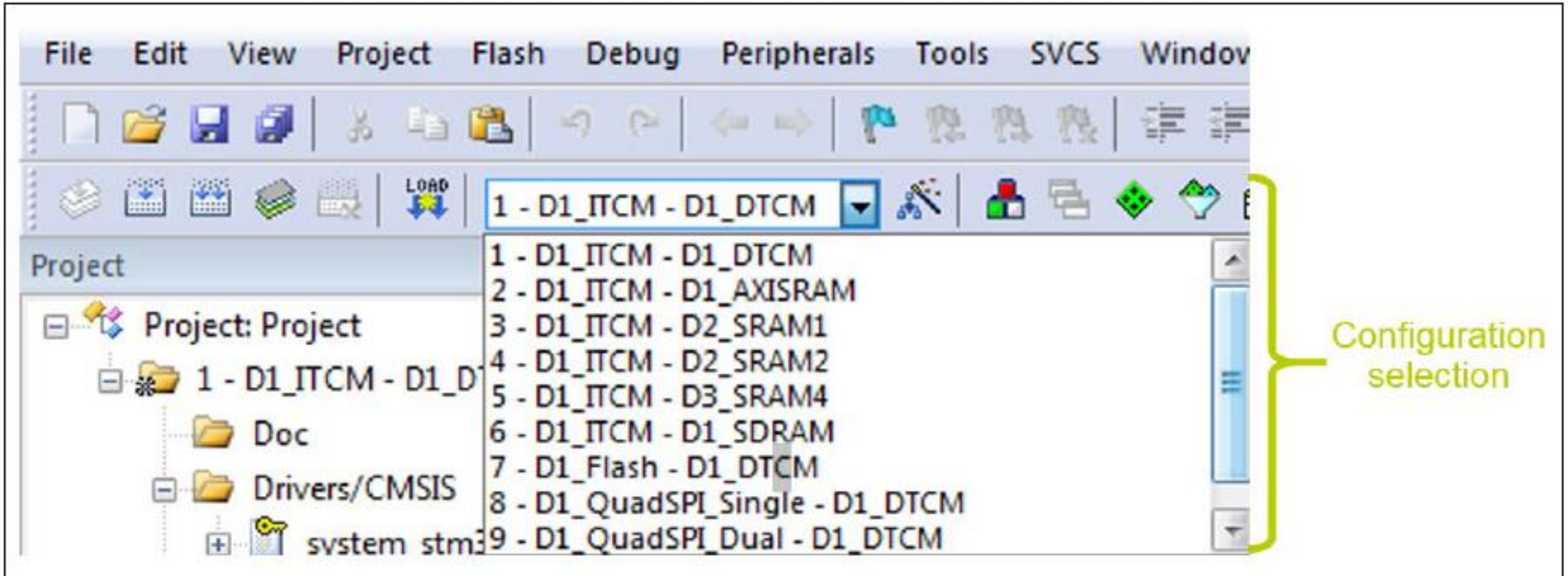


life.augmented

STM32H7-性能
X-CUBE-PERF-H7《AN4891》

项目配置

Figure 9. Keil® (MDK-ARM) configuration selection



项目配置

Figure 10. MDK-ARM flags configuration

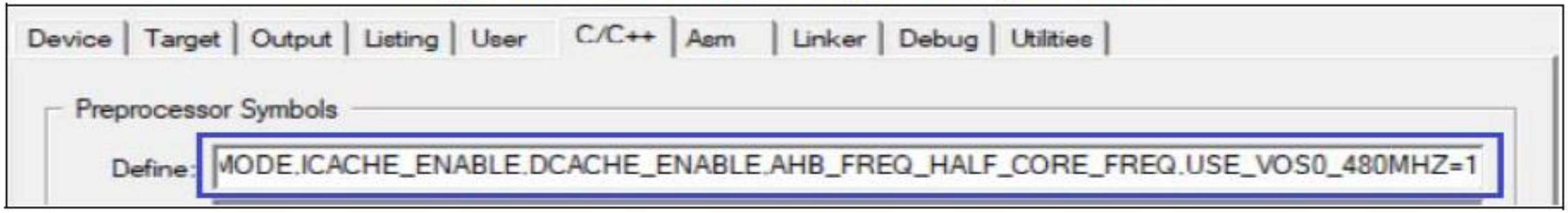
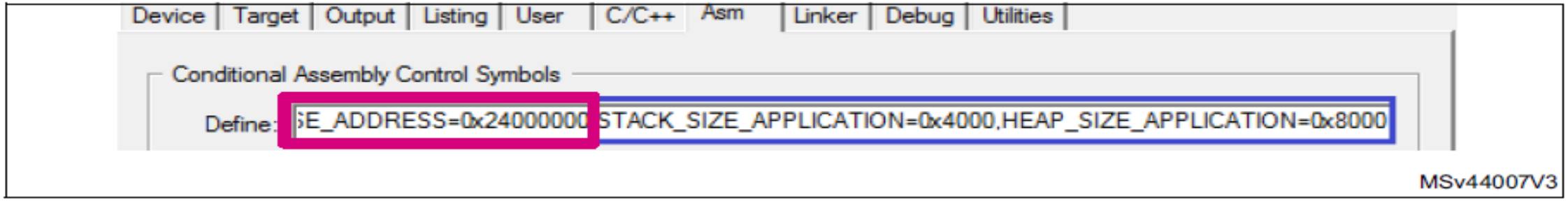
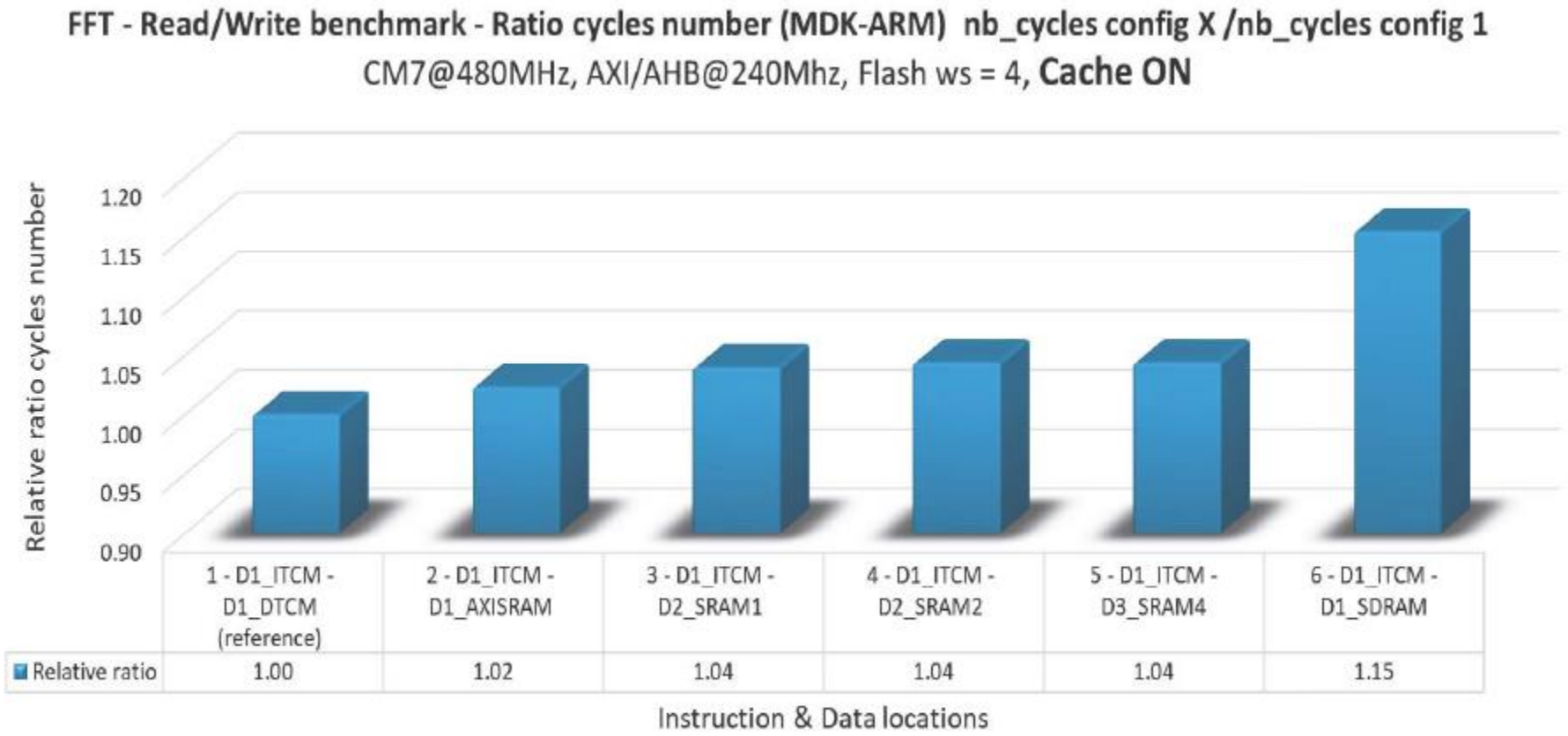


Figure 11. MDK-ARM heap and stack configurations



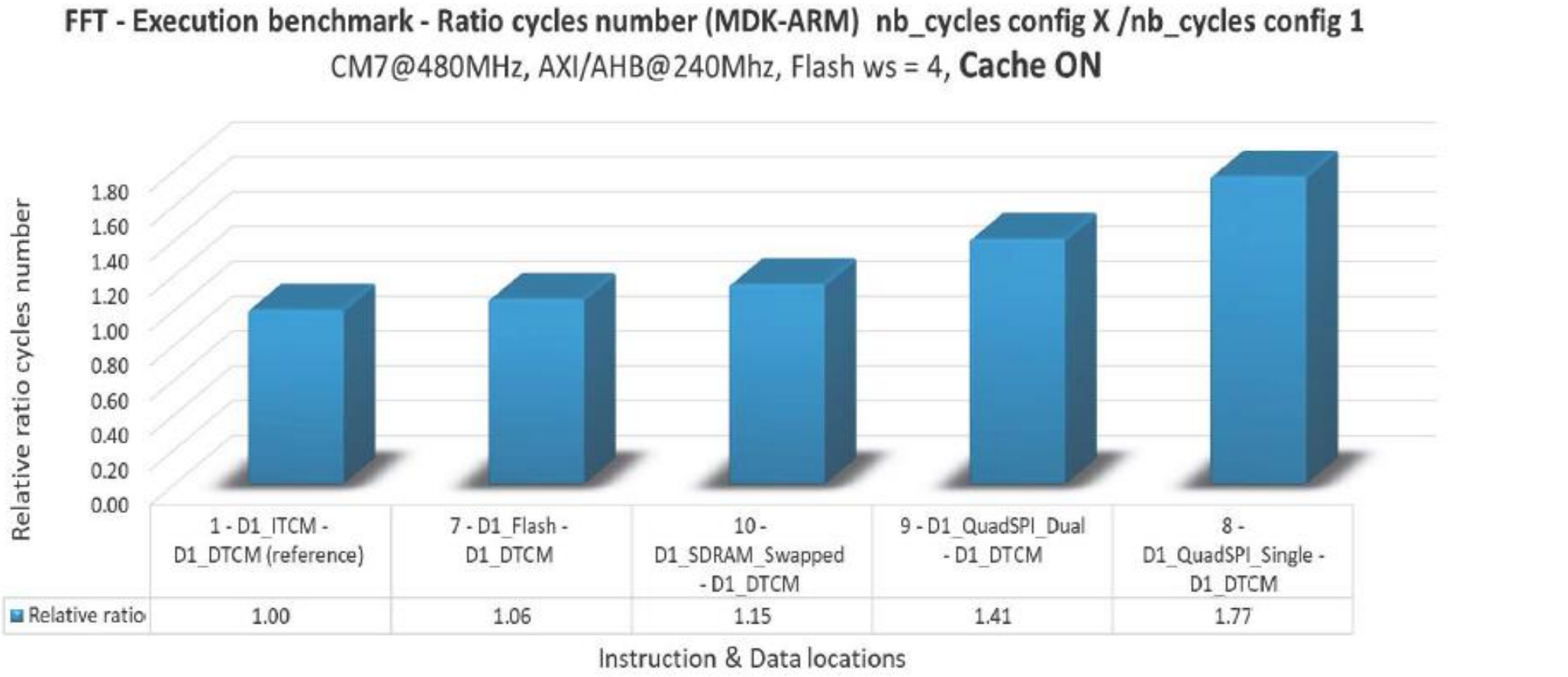
life.augmented

M7 - FFT 数据读写 Benchmark位于不同位置 (代码指令执行固定在ITCM)



life.augmented

M7 - FFT 执行 Benchmark代码指令位于不同位置(数据固定在DTCM)



life.augmented

结果分析

- DTCM-RAM和ITCM-RAM位置分别是读取/写入数据和获取指令的最佳位置。
- ITCM-RAM是为具有确定性执行的关键代码保留的，例如无法等待缓存未命中的中断处理程序，以及一些以电机控制应用程序为目标的关键控制循环。STM32H72x/73x ITCM-RAM的大小是可配置的，因此可以根据应用要求进行调整。
- DTCM-RAM用于常规访问R/W数据和关键实时数据，如stack和heap。为了更高速度的计算，如果数据位于例如D1域中的AXI SRAM、D2域中的SRAM1或D3域中的SRAM4中，则可以使用MDMA将数据从这些存储器移动到DTCM-RAM以便以CPU时钟速度进行处理。

结果分析

- 位于D1域的AXI SRAM可以用于16位模式下QVGA TFT的图形应用中的图形帧缓冲区，这些应用需要大量图形数据和高性能显示（由LCD-TFT和DMA2D DMA保证）。为了达到此效果，必须将STM32H72x/73x共享SRAM配置为AXI SRAM。
- 位于D2域中的SRAM1、SRAM2和SRAM3可用作缓冲区来存储位于D2域中的外围设备（例如以太网和USB）的输入/输出数据。通过使用MDMA将数据从SRAM1、SRAM2或SRAM3传输到DTCM-RAM，可以在DTCM-RAM中处理数据。
- 位于D3域中的SRAM4通常用于存储用户应用程序低功耗部分的数据。当D1和D2进入DStandby模式时，它可以用来保留一些应用程序数据。低功耗应用数据可以是CPU的R/W数据或要由诸如LPUART1、I2C4等外设（位于D3域中）传输的缓存。
- 在MDMA和Cortex®-M7同时访问DTCM-RAM的情况下，如果Cortex®-M7没有最高优先级访问，则应用程序速度可能会降低。该优先级可由软件修改CM7_AHBSCR寄存器进行配置。

life.augmented

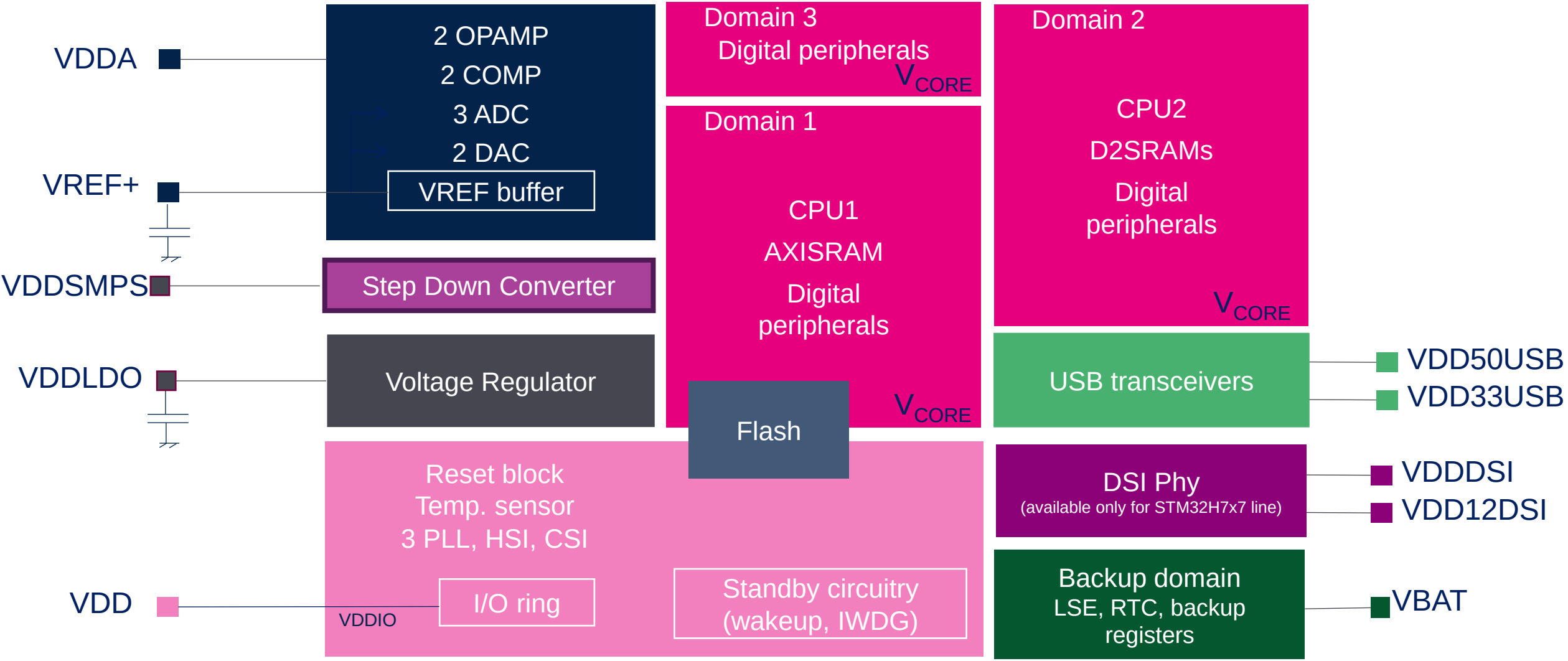
小结

- 知道我们High performance 的例程，以及相应的应用笔记，
- 基于例程X-CUBE-PERF-H7《AN4891》，能够帮助客户porting，
- 结合系统框图选择合适的指令存储和数据存储的路径，合理分配资源

life.augmented

STM32H7- PWR

电源方案


VDDA
2 OPAMP
2 COMP
3 ADC
2 DAC
VREF+
VREF buffer
VDDSMPS
Step Down Converter
VDDLDO
Voltage Regulator
Domain 3
Digital peripherals
VCORE
Domain 1
CPU1
AXISRAM
Digital peripherals
VCORE
Flash
Domain 2
CPU2
D2SRAMs
Digital peripherals
VCORE
USB transceivers
VDD50USB
VDD33USB
DSI Phy
(available only for STM32H7x7 line)
VDDDSI
VDD12DSI
Reset block
Temp. sensor
3 PLL, HSI, CSI
VDD
VDDIO
I/O ring
Standby circuitry
(wakeup, IWDG)
Backup domain
LSE, RTC, backup registers
VBAT

仅双核

SD降压转换器

- SD的用法:
  - SD直接供电给V_{CORE} 域
    - SD 的工作模式遵循H7的工作模式 (Run, Stop, Standby)..
    - SD 的输出电压由VOS和SVOS决定.
  - SD提供级联的供电给LDO，LDO供电给Core
    - SD 的工作模式遵循H7的工作模式 (Run, Stop, Standby).
    - SD 的输出电压由SDLEVEL (1.8 V or 2.5 V)决定.
  - SD用于给外部供电.
    - SD 强制性的一直处于High Performance工作模式.
    - SD 的输出电压由SDLEVEL (1.8 V or 2.5 V)决定.
- 工作模式
    - Main(Run or External supply)
    - Low Power (Stop)
    - Off (Standby)

life.augmented

仅双核

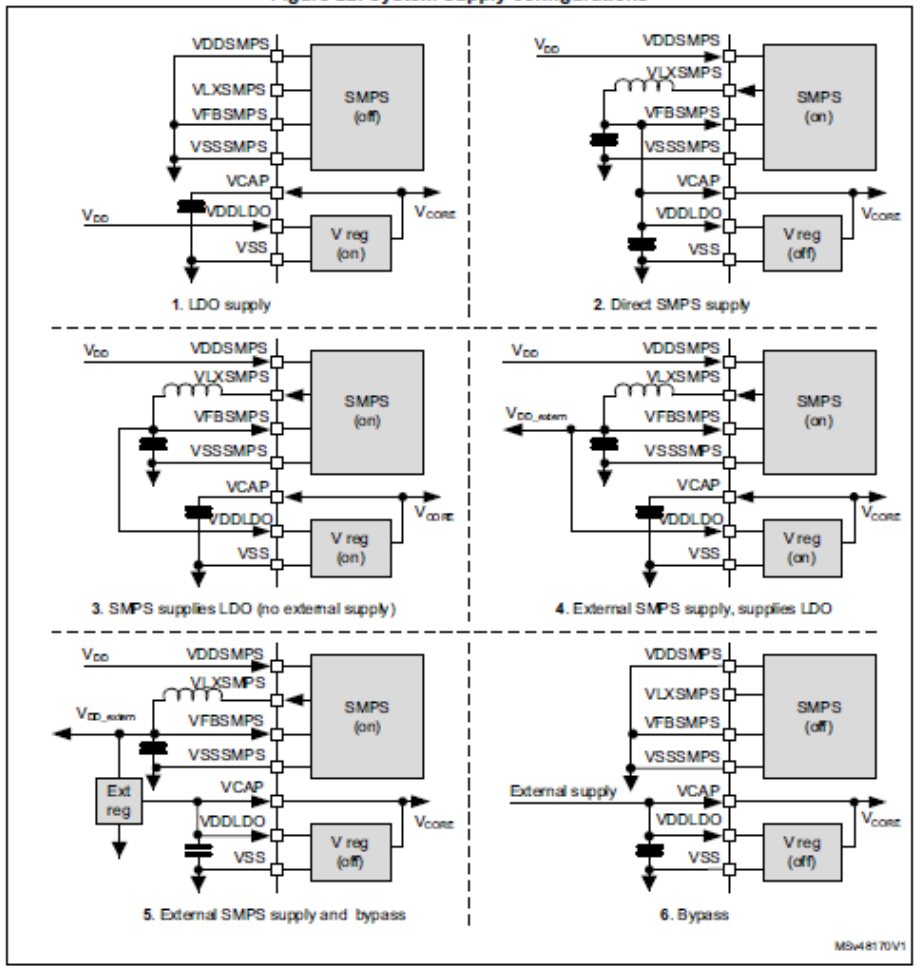
供电配置

介于性能和成本之间的灵活的配置选择

- Core 供电来源于SD convertor 且/或 Voltage regulator
  - 1. LDO 供电 (SD disabled)
    - LDO 的供电来源和VDD一致
  - 2. SD输出直接给Vcore供电 (LDO disabled)
    - Regulator的模式遵循 H7的工作模式
  - 3. SD 供电给 LDO (没有外部的供电)
    - 这两个regulators的模式都遵循H7的工作模式
  - 4. SD供给 LDO的同时，也给外部供电
    - SD 强制性的需要处于high performance模式
    - LDO的模式遵循H7的工作模式
  - 5. SD给外部供电，LDO 旁路模式 (LDO disabled)
    - SD 强制性的需要处于high performance模式
  - 6. 旁路模式 (SD 和LDO都是 disabled)

(SD 不能够给VDD供电)

Figure 22. System supply configurations

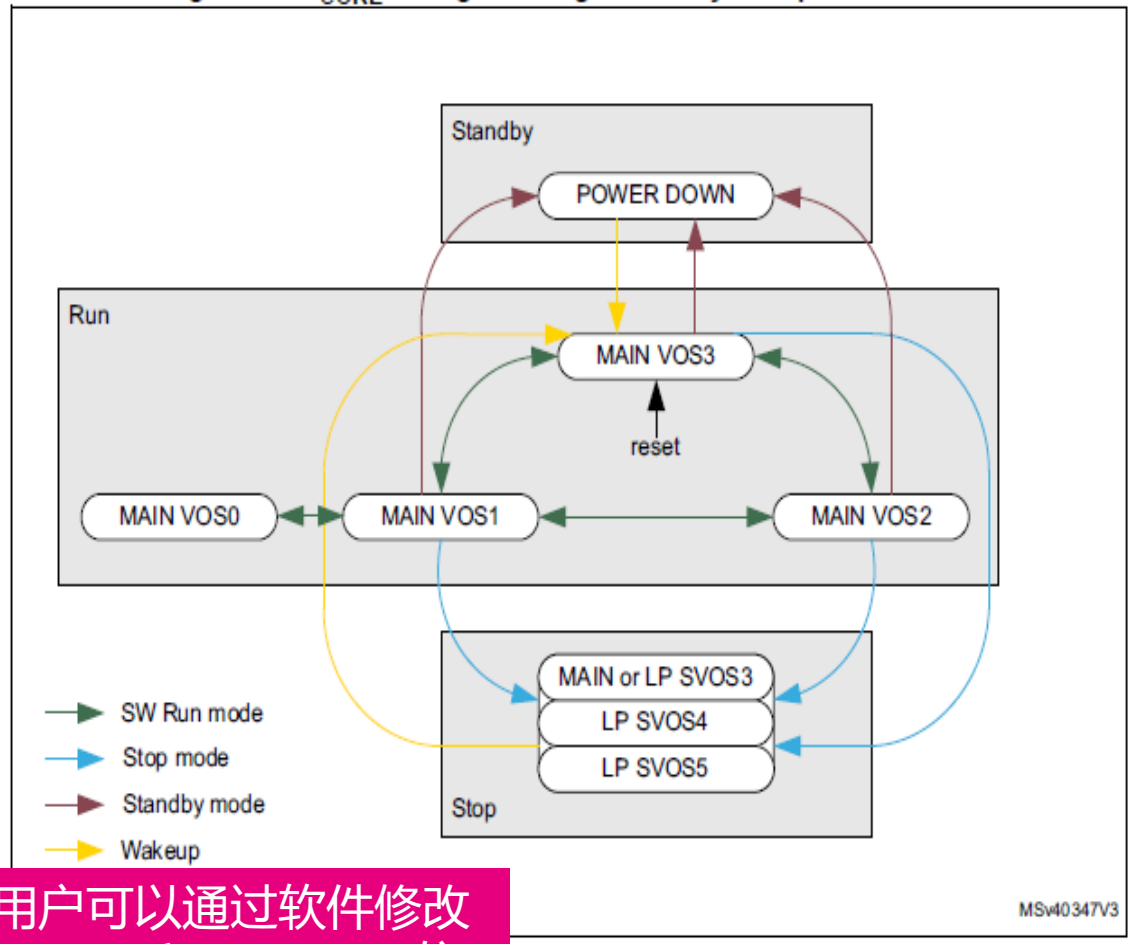


降压转换器

• 供电配置控制

ID	供电配置	SDLEVEL	SDEXTHP	SDEN	LDOEN	BYPASS
0	默认配置	00	0	1	1	0
1	LDO供电	x	x	0	1	0
2	直接SD转换器供电	x	0	1	0	0
3	SD 转换器给LDO供电	01 或 10	0	1	1	0
4	SD 转换器给外部和LDO供电	01 或 10	1	1	1	0
5	SD 转换器给外部供电，LDO被旁路	01 或 10	1	1	0	1
6	SD 禁能并且 LDO 旁路	x	x	0	0	1

Figure 36. V_{CORE} voltage scaling versus system power modes



在系统启动 (上电), MCU的默认配置为LDO供电，之后用户可以通过软件修改**PWR_CR3寄存器**中的SDLEVEL, SDEXTHP, SDEN, LDOEN 和 BYPASS 位进行供电配置

life.augmented

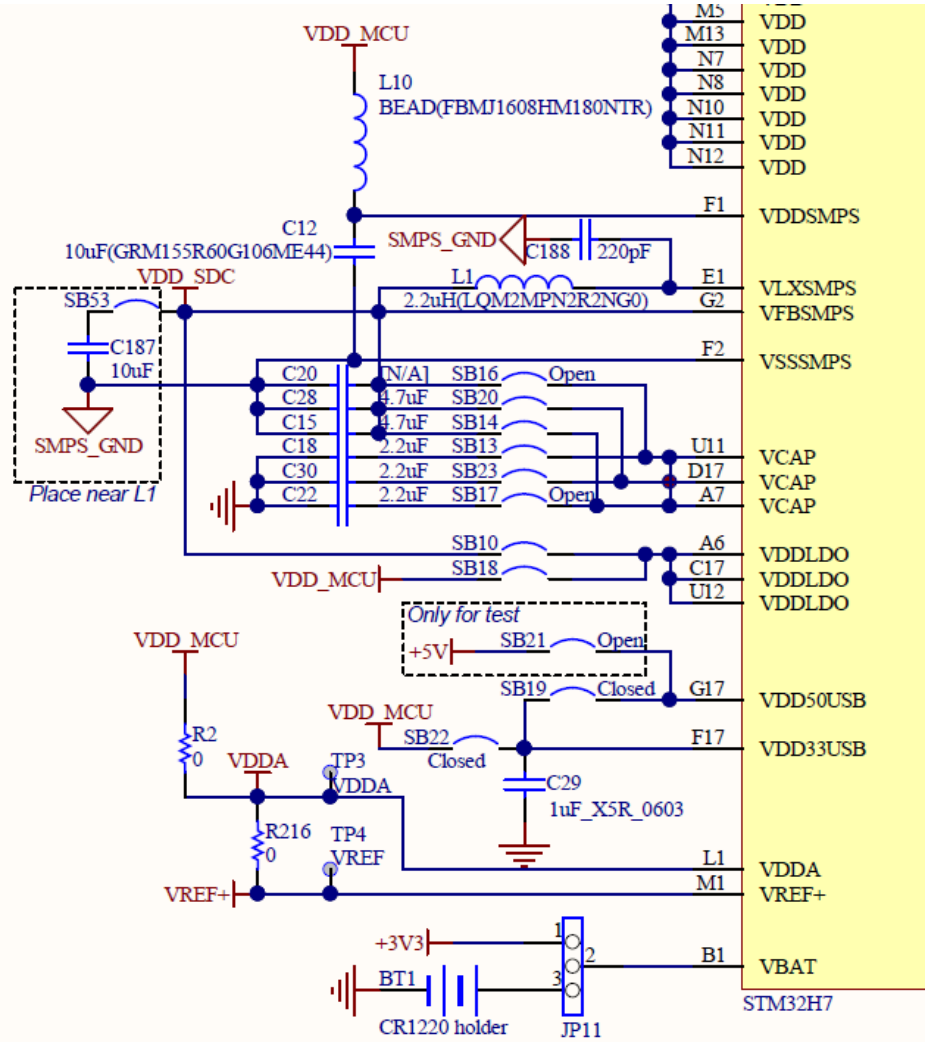
评估版:
电源配置

- 灵活的 V_{core}-供电配置STM32H7x7I-EVAL
  - LDO 供电
  - SD直接给Vcore
  - SD 供电给 LDO，LDO供电给Vcore

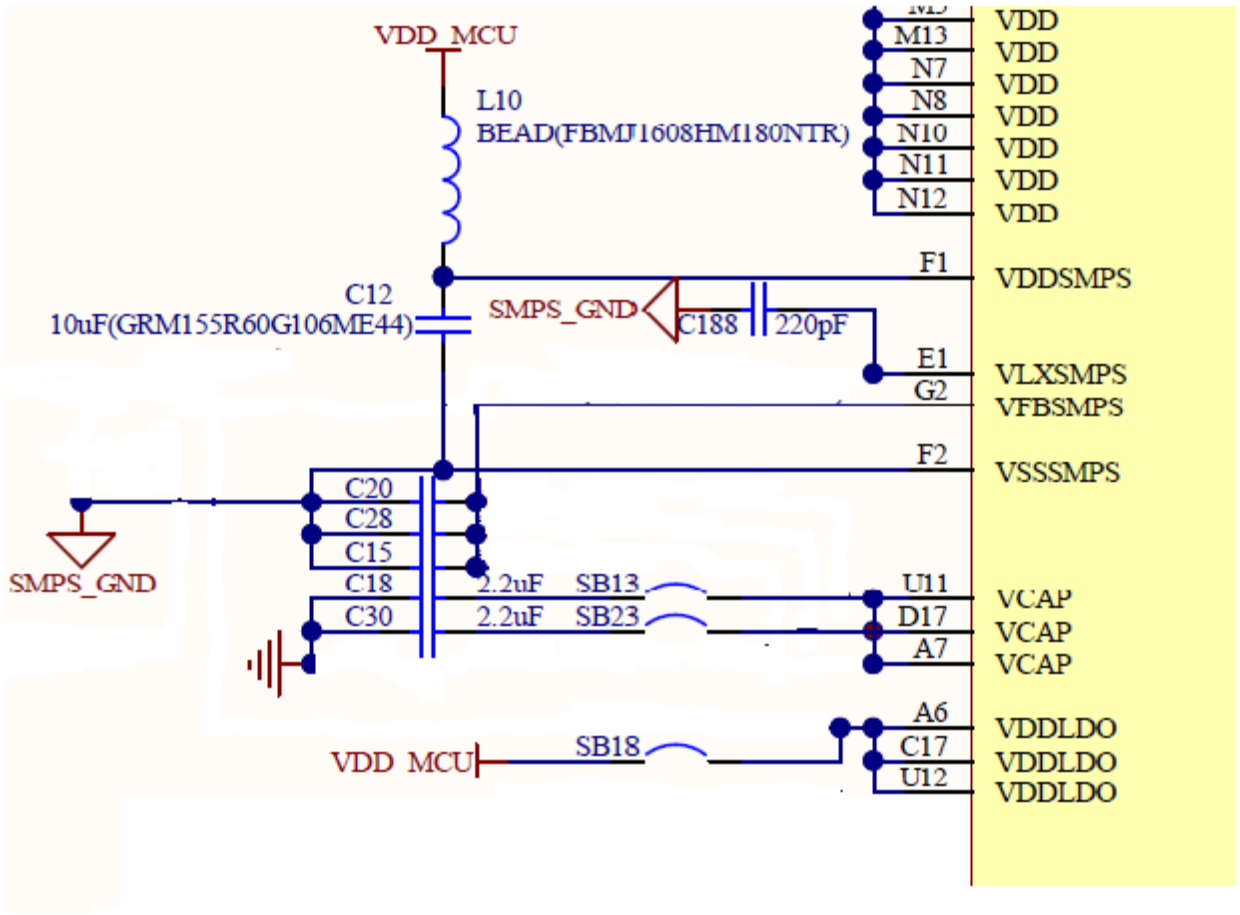
- 可以通过桥接电阻进行灵活的配置

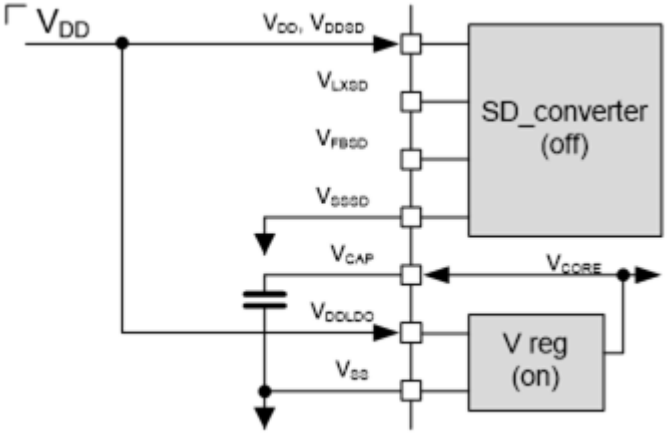
Power configurations:

Mode	Mount	Removed
LDO mode STM32H7x3 Default setting	SB13,SB23,SB18	SB14,SB16,SB20 SB10,SB17,SB53 L1
SMPS mode STM32H7x7 Default setting	SB14,SB20,SB10 SB53 L1	SB13,SB17,SB23 SB16,SB18
SMPS+LDO mode	SB13,SB23,SB10 SB53, L1	SB14,SB16,SB17 SB18,SB20

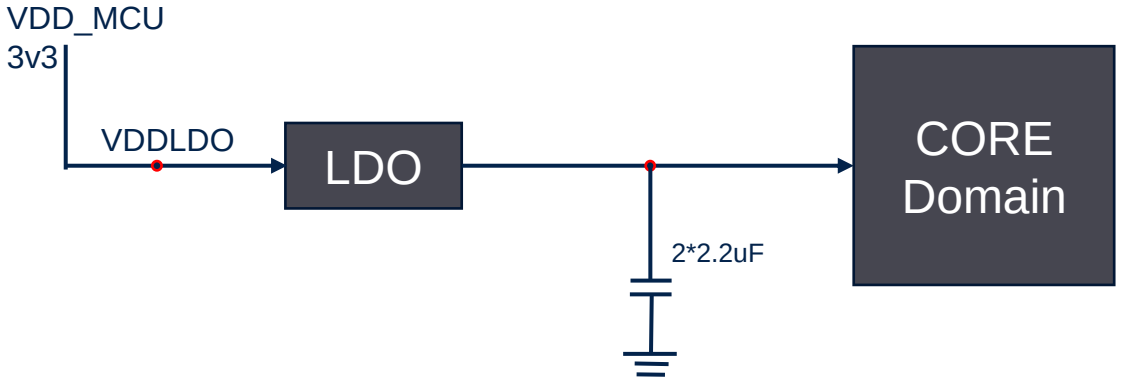


1、LDO Mode

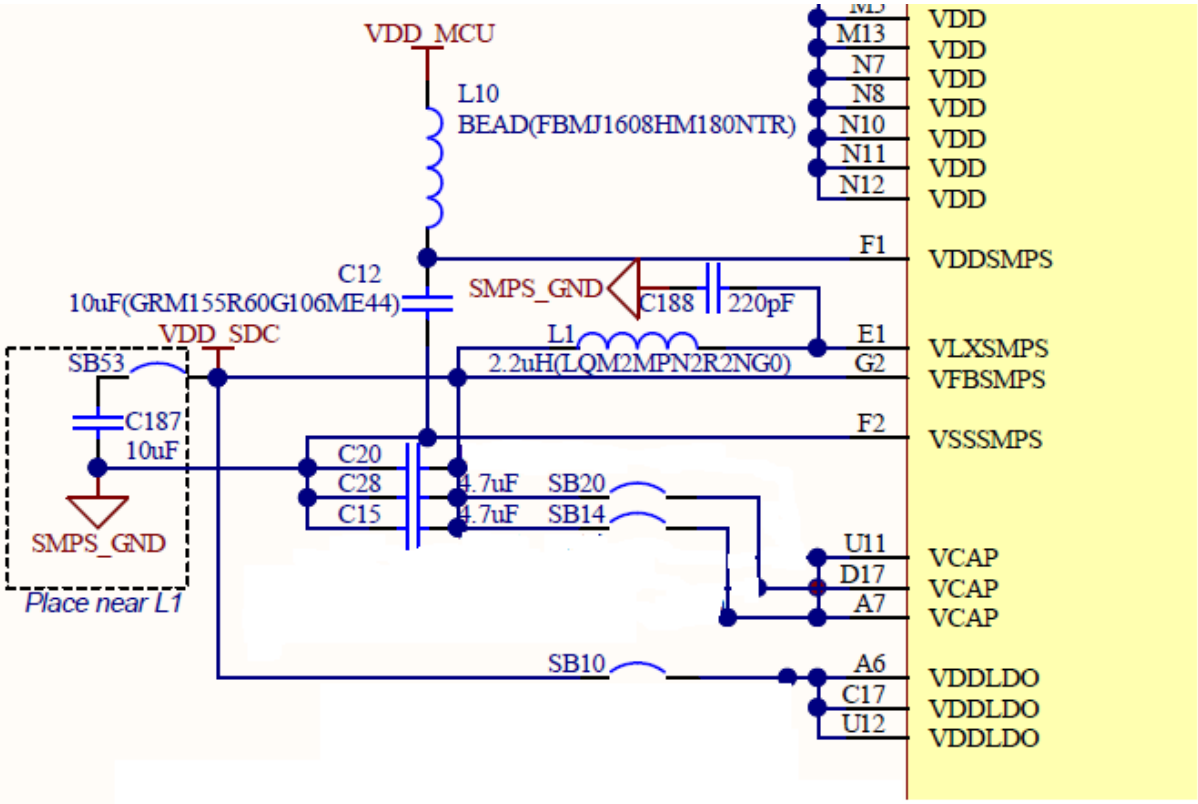


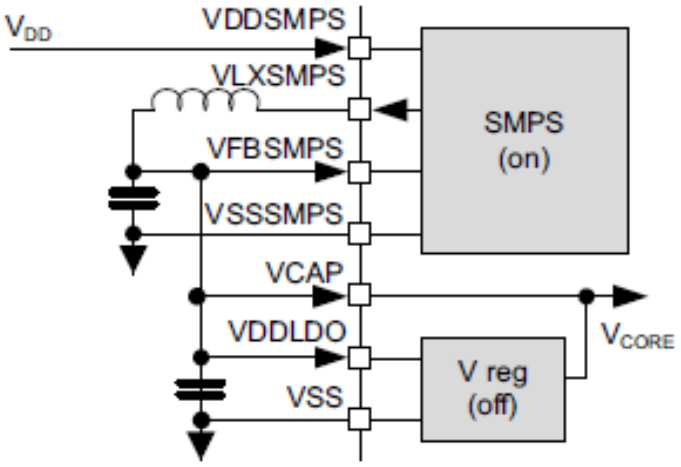


1. LDO Supply

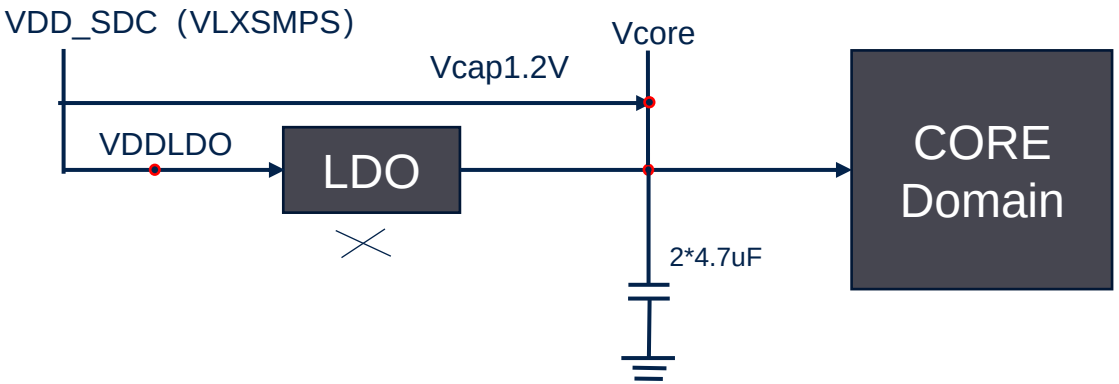


2、SMPS Mode

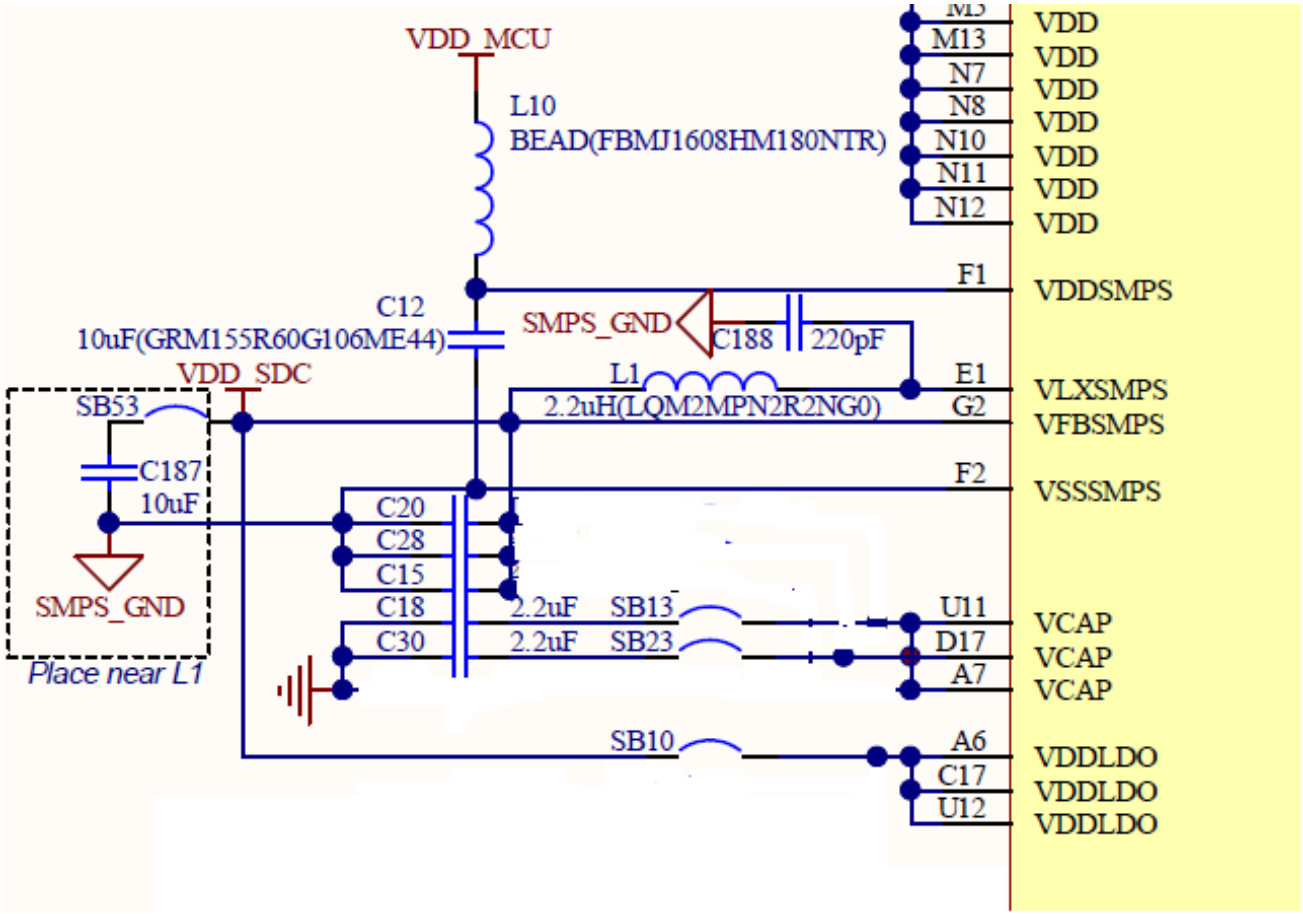


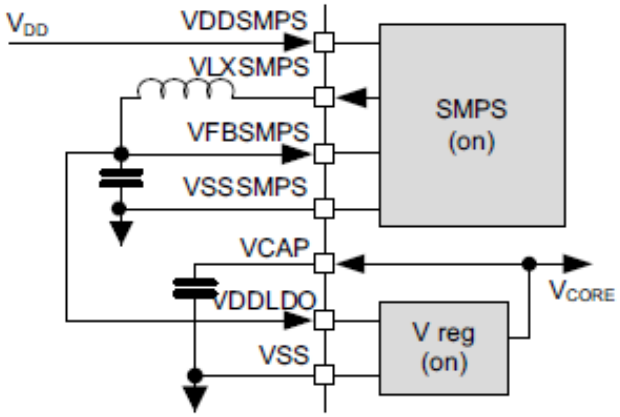


2. Direct SMPS supply

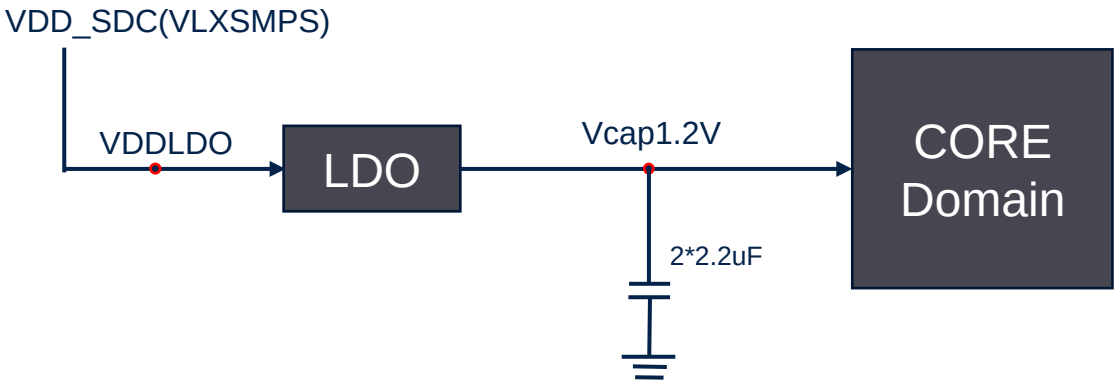


3、SMPS +LDO Mode



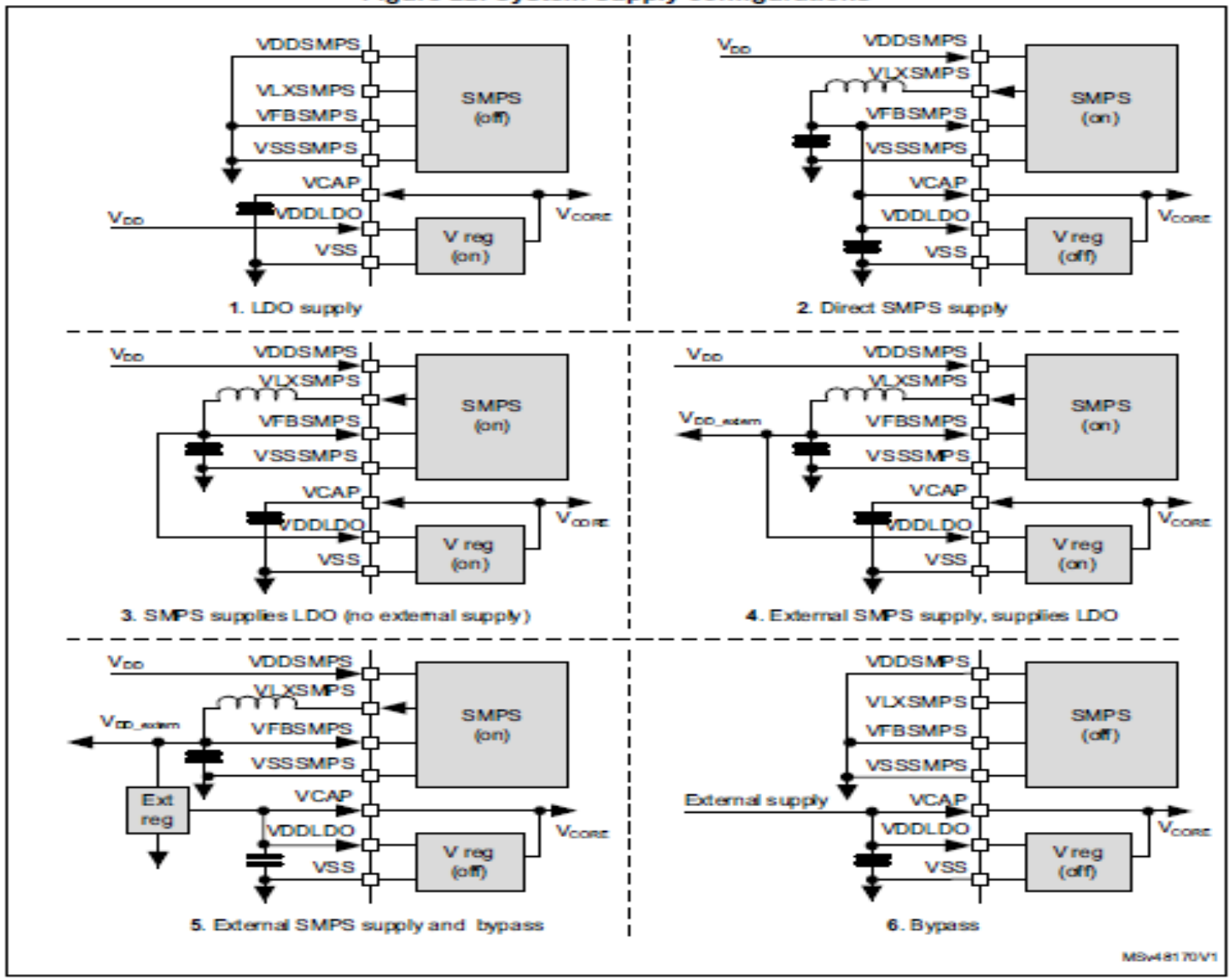


3. SMPS supplies LDO (no external supply)



life.augmented

Figure 22. System supply configurations



life.augmented

Bypass模式常见的问题

- 1、外部电源的需求（电流和电压斜率）

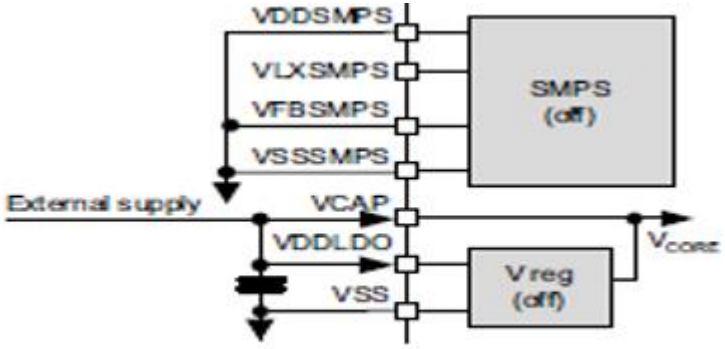
VDDLDO (和VCAP)的斜率要求必须满足经典的常数10us/V最大。和如下手册中的电源要求一样：

6.3.4 Operating conditions at power-up / power-down

Subject to general operating conditions for T_A.

Table 27. Operating conditions at power-up / power-down (regulator ON)

Symbol	Parameter	Min	Max	Unit
t_{VDD}	V_{DD} rise time rate	0	∞	µs/V
	V_{DD} fall time rate	10	∞	
t_{VDDA}	V_{DDA} rise time rate	0	∞	
	V_{DDA} fall time rate	10	∞	
t_{VDDUSB}	V_{DDUSB} rise time rate	0	∞	
	V_{DDUSB} fall time rate	10	∞	



6. Bypass

- 2、 外部电源和VDD等的上电先后时许的要求

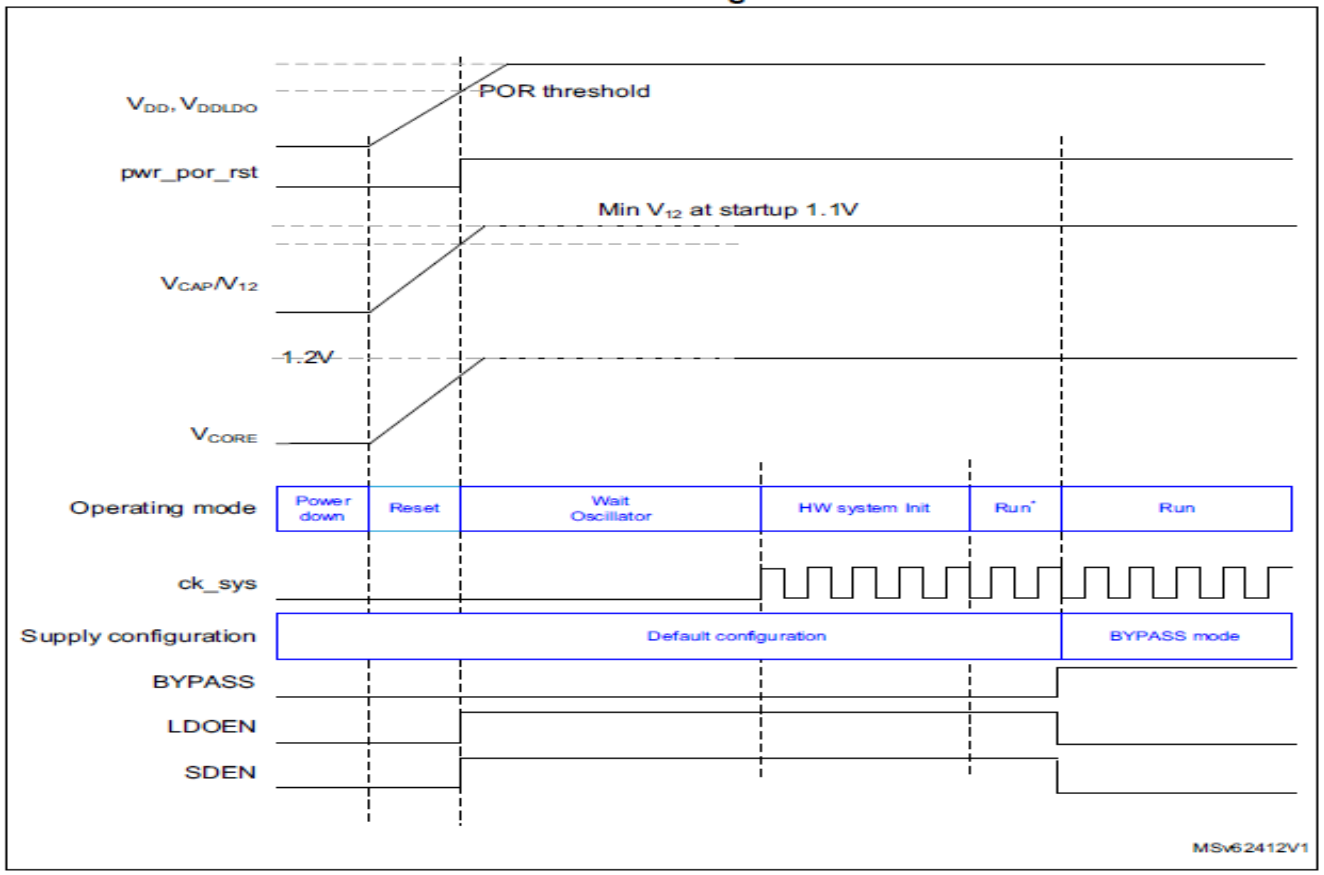
bypass模式解析

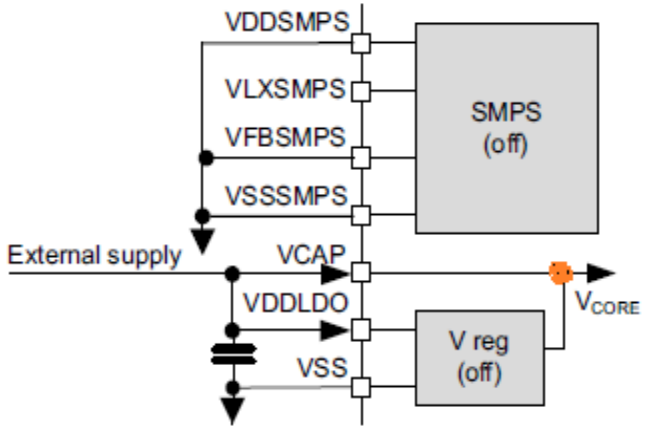
V_{CORE} supplied in Bypass mode (LDO and SMPS OFF)

When V_{CORE} is supplied in Bypass mode (LDO and SMPS OFF), the V_{CORE} voltage must first settles at a default level higher than 1.1 V. Due to the LDO default state after power-up (enabled by default), the external V_{CORE} voltage must remain higher than 1.1 V until the LDO is disabled by software.

When the LDO is disabled, the external V_{CORE} voltage can be adjusted according to the user application needs (refer to section *General operating conditions* of the datasheet for details on V_{CORE} level versus the maximum operating frequency).

Figure 25. Device startup with V_{CORE} supplied in Bypass mode from external regulator





6. Bypass

MSv48170V1

小结

- 1、原理图的设计参考EVAL板的原理图
- 2、《getting start with xxxxx hardware development》
- 3、特殊的供电方式，比如bypass模式，有疑问请发TSR

life.augmented

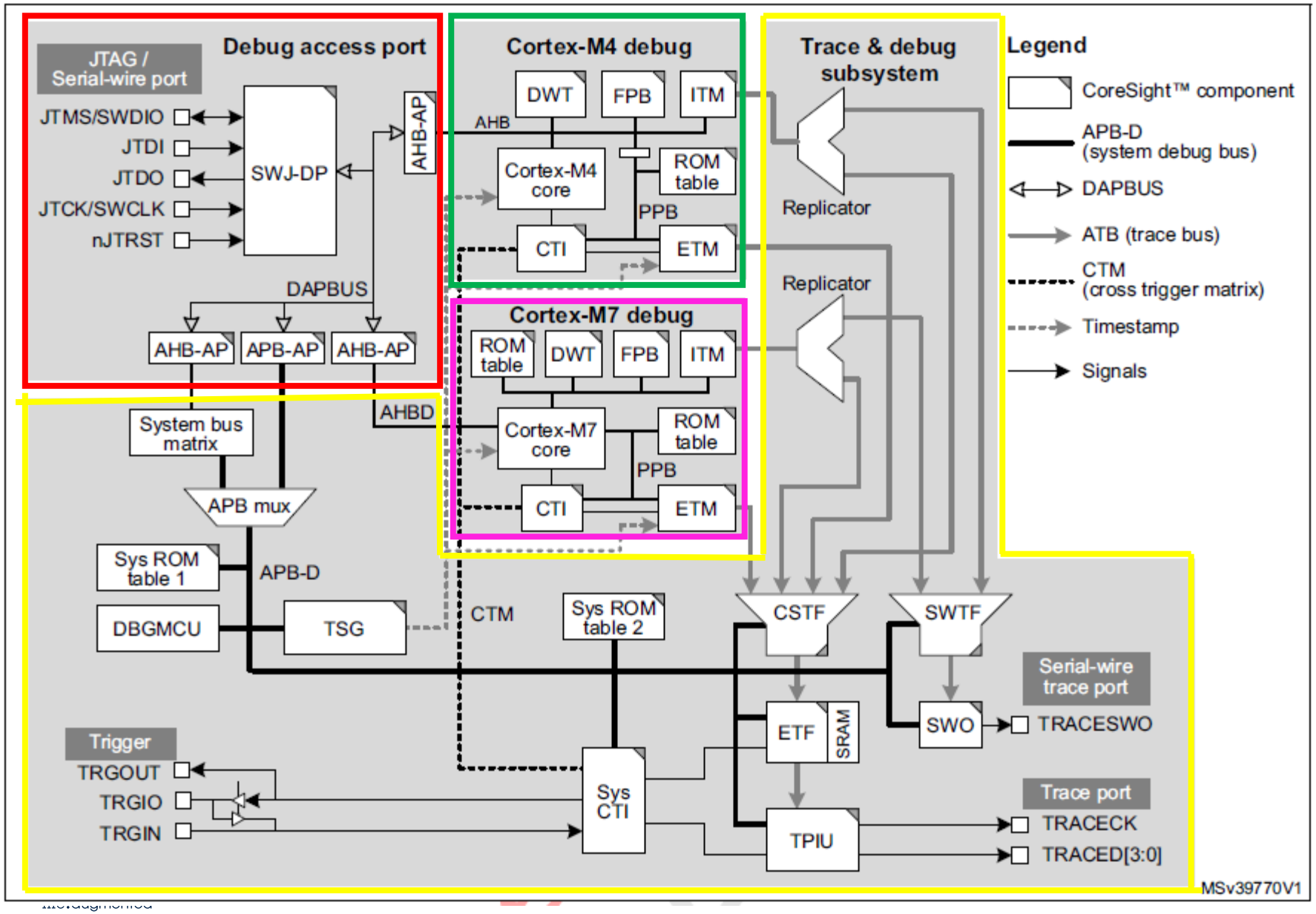
STM32H7- Debug

- CoreSight：ARM公司于2014年推出的一种新的调试体系架构，它可对完整的片上系统（SoC）设计的性能进行调试、监视、优化，即实现对内核的调试和跟踪功能。
  - 跟踪和调试子系统的CoreSight组件
  - Cortex-M7内核的CoreSight组件
  - Cortex-M4内核的CoreSight组件

- 编译器IDEs对双核调试的支持(请保持使用最新版本的编译器)
  - IAR对于STM32H7双核的支持和演示
  - Keil对于STM32H7双核的支持和演示
  - CubeIDE 对于STM32H7双核的支持和演示（IDE 1.4.0）
    - 可参看往期培训：www.stmcu.com.cn

life.augmented

STM32H7 双核调试块框图

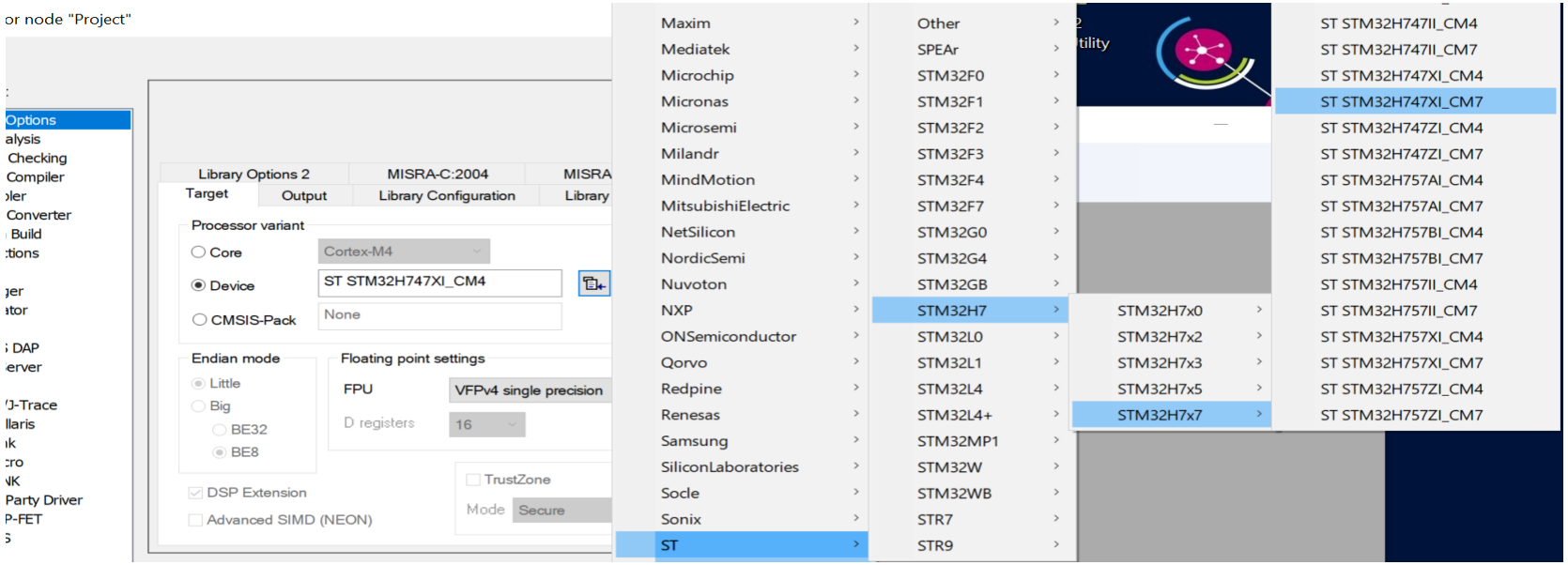


APSEL[3:0]: Access port select bits
These bits select the access port for the next transaction.

0x0: AP0 - Cortex-M7 debug access port (AHB-AP)
0x1: AP1 - D3 access port (AHB-AP)
0x2: AP2 - System debug access port (APB-AP)
0x3: AP3 - Cortex-M4 debug access port (AHB-AP)
0x4 to 0xF: Reserved

IAR 对STM32H7双核调试的支持

- 请下载最新版本的IAR，最新版本能够很好的支持双核调试，不需要安装额外的补丁包
  - IAR Version 8.3，开始支持双核，不需要额外安装补丁。
    - https://www.iar.com/iar-embedded-workbench/#!?currentTab=free-trials
    - IAR打开一个项目工程，右击项目，在弹出菜单中Option, 查看是否支持双核MCU型号。
    - IAR下支持STLINK V2、STLINK V3、I-Jet等对双核的调试。
  - 如果是低版本的IAR，则需要打补丁去支持双核调试，这里不做介绍。

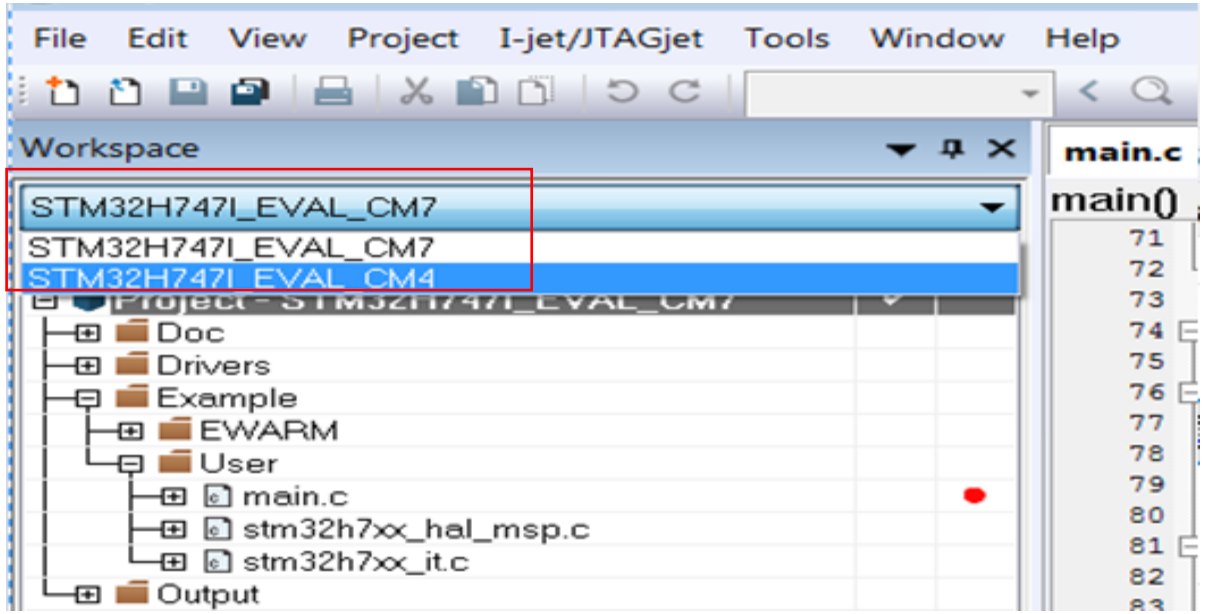


life.augmented

IAR 下I-Jet对STM32H7双核调试的支持

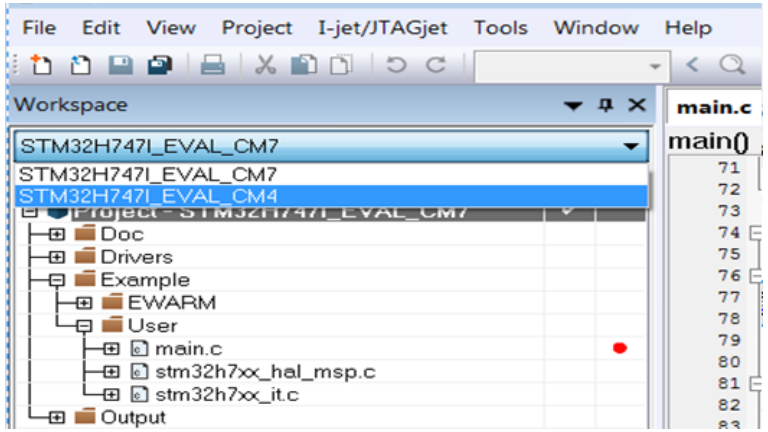
- 一个I-Jet调试工具可以完美的支持双核的调试(CM7与CM4)。
- 需要在一个workspace下为每个内核建立一个项目工程，也就是要建立两个项目工程, 需要设定主项目工程指向从项目工程。
  - 2个projects在同1个workspace下面，或者1个project在1个workspace下，但是这1个project它有2个configuration
  - Configuration可以通过菜单栏里面的Project->Edit Configurations修改

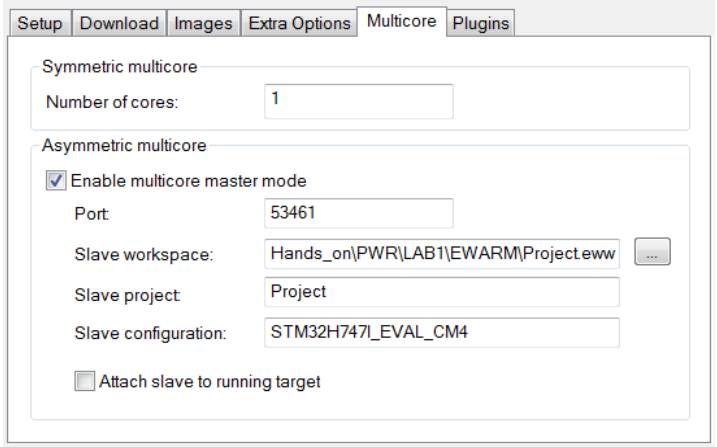
Project（Master/slave）
Configuration



IAR 下I-Jet对STM32H7双核调试的支持

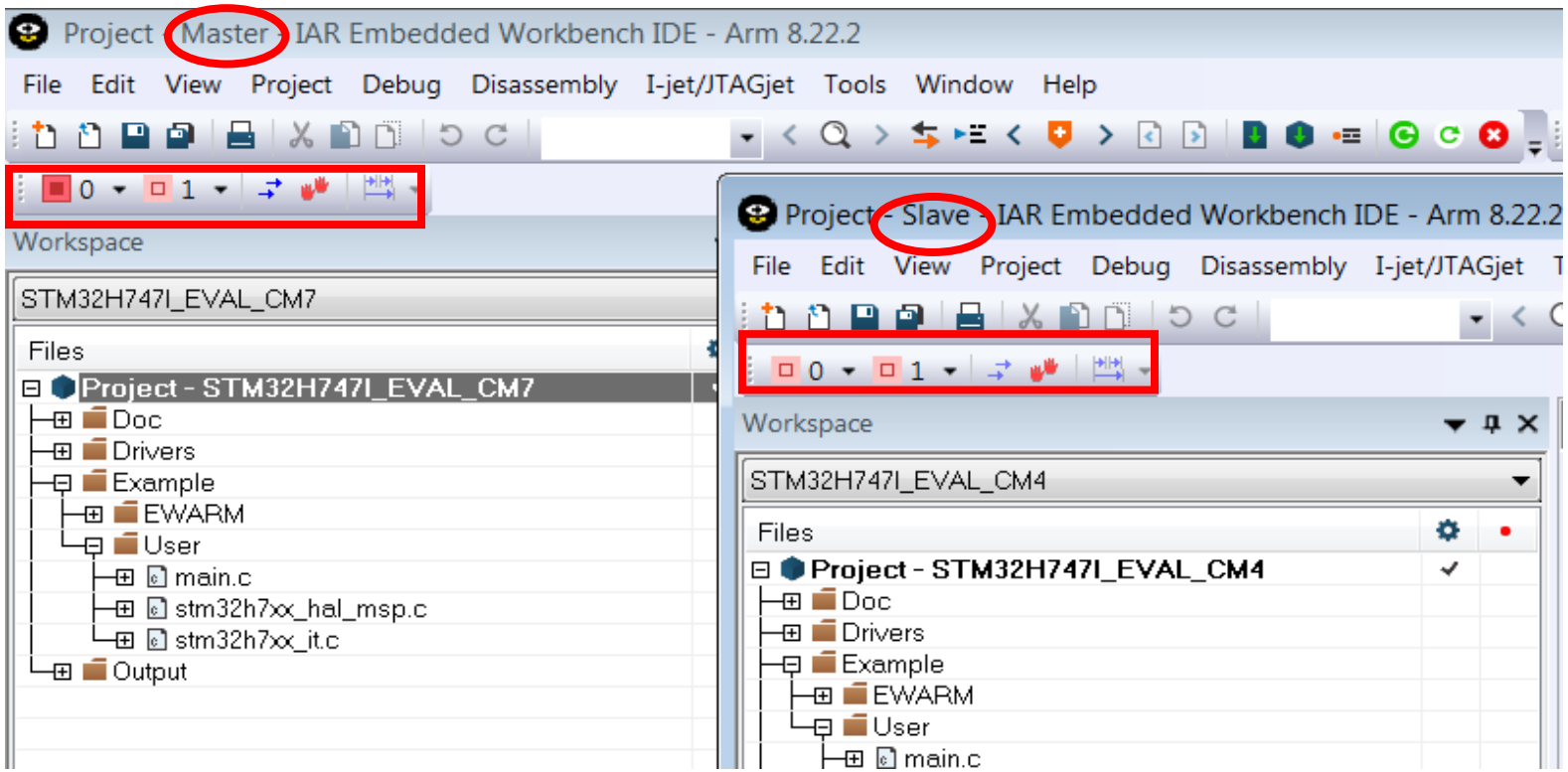
- 对于Master主项目工程(一般是CM7)
  - Project>Options>Debugger>Enable multicore master mode
    - Symmetric multicore: 对称的多核调试，常见用于菊花链daisy chain的调试，如多个CM4的内核。
    - Asymmetric multicore: 非对称多核调试，如H7x7内两个内核是不一样的，一个CM7，一个CM4
  - 指定port：可以为任意数值，但是需要和从项目工程在该处的port号一致，才能保证双核之间通信。
  - 指定slave workspace path：.eww的路径
  - Salve project name:.eww后缀前面的名字，如右下角图，填入Project即可
  - Slave configuration 名称：
- 对于Slave从项目工程(一般是CM4)
  - 仅仅需要你选择的是I-Jet的调试器，其他保持默认值设定即可。





IAR 下I-Jet对STM32H7双核调试的支持

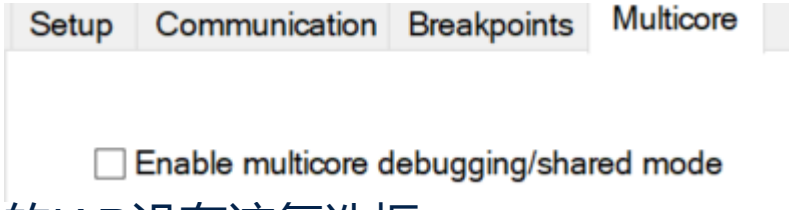
- 主和从的关系属性可以通过主窗口的标题栏显现出来。
- 可以通过主或从项目工程中的工具栏菜单独立的控制run/halt一个核或者两个核一起控制。
  - 下图的IAR版本8.22.2较低，是通过安装补丁包的方式去实现支持双核调试的，最新版本的IAR不需要额外安装补丁.

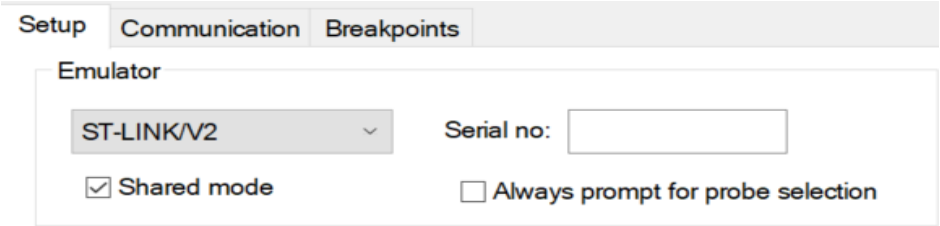


life.augmented

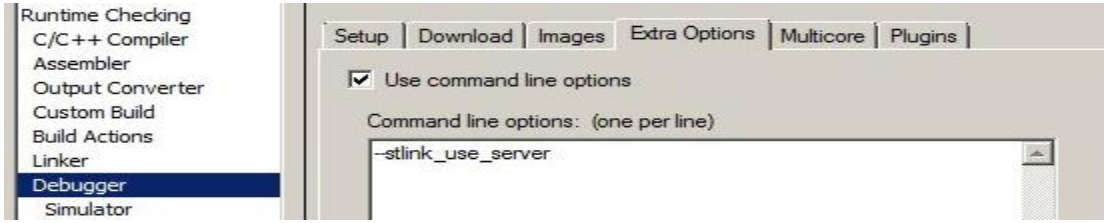
IAR 下STLINK V2对STM32H7双核调试的支持

- 请使用IAR最新版本并利用ST-LINK FW v2J28 或更新版本
- 需要在一个workspace下建立为每个内核建立一个项目工程。
- 使能Dual core双核的调试通过以下两种方式实现:
  - Option->ST-LINK->Setup->Shared mode
    - Shared mode该复选框存在于较新的IAR编译器版本中，低版本的IAR没有该复选框



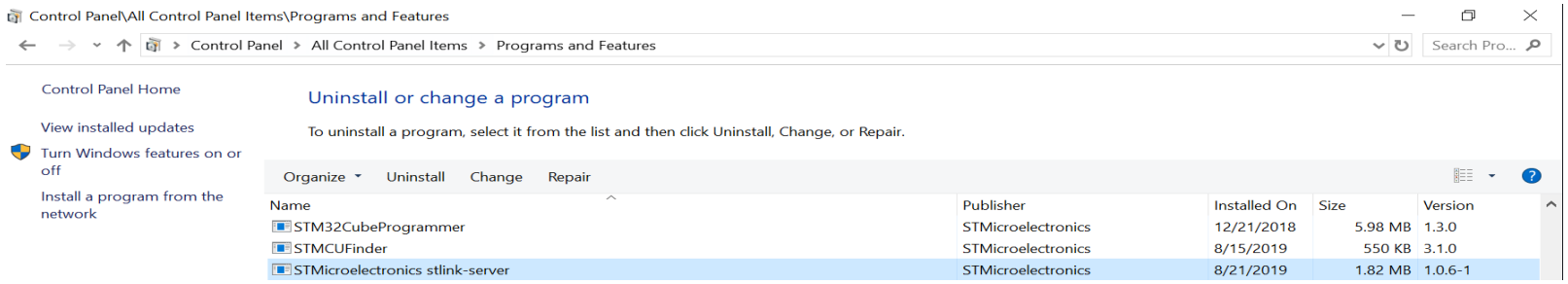


  - 命令行的形式--stlink_use_server
    - 不管是新版本还是旧版本，都可以使用命令行的形式，需要手动输入命令行—stlink_use_server
    - 使用复选框较方便，不需要输入命令行，推荐使用shared mode的方式去调试双核

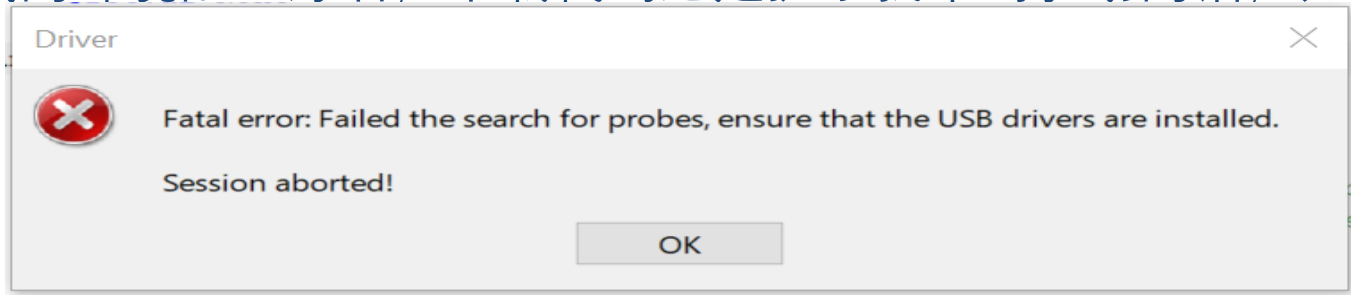


IAR 下STLINK V2对STM32H7双核调试的支持

- 不管是使用shared mode还是命令行形式，都需要确保stlink server驱动被正确安装：



- 可查看C:\Program Files (x86)\STMicroelectronics处是否有stlink_server文件夹。
- 如果双核调试的环境配置好后，下载代码还是提示找不到驱动的话，如下图：



- 可卸载原先的stlink server的驱动并手动删除C:\Program Files (x86)\STMicroelectronics处stlink_server的文件夹，并于ST的官网下载解压后安装。

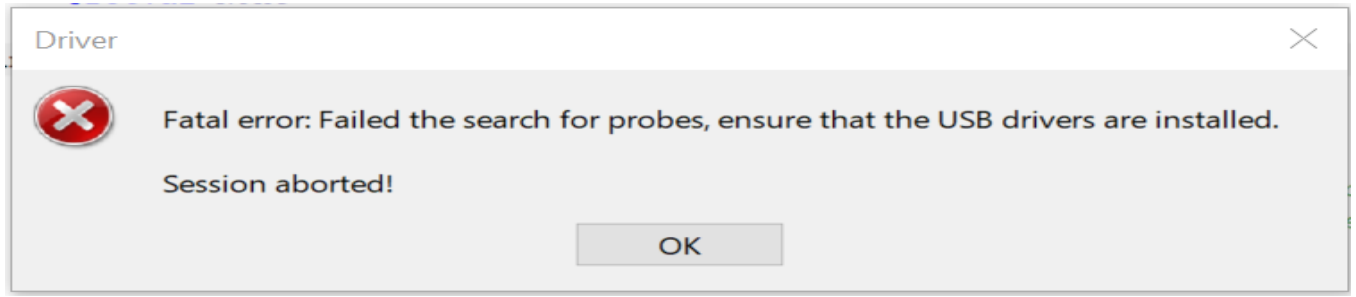
Get Software

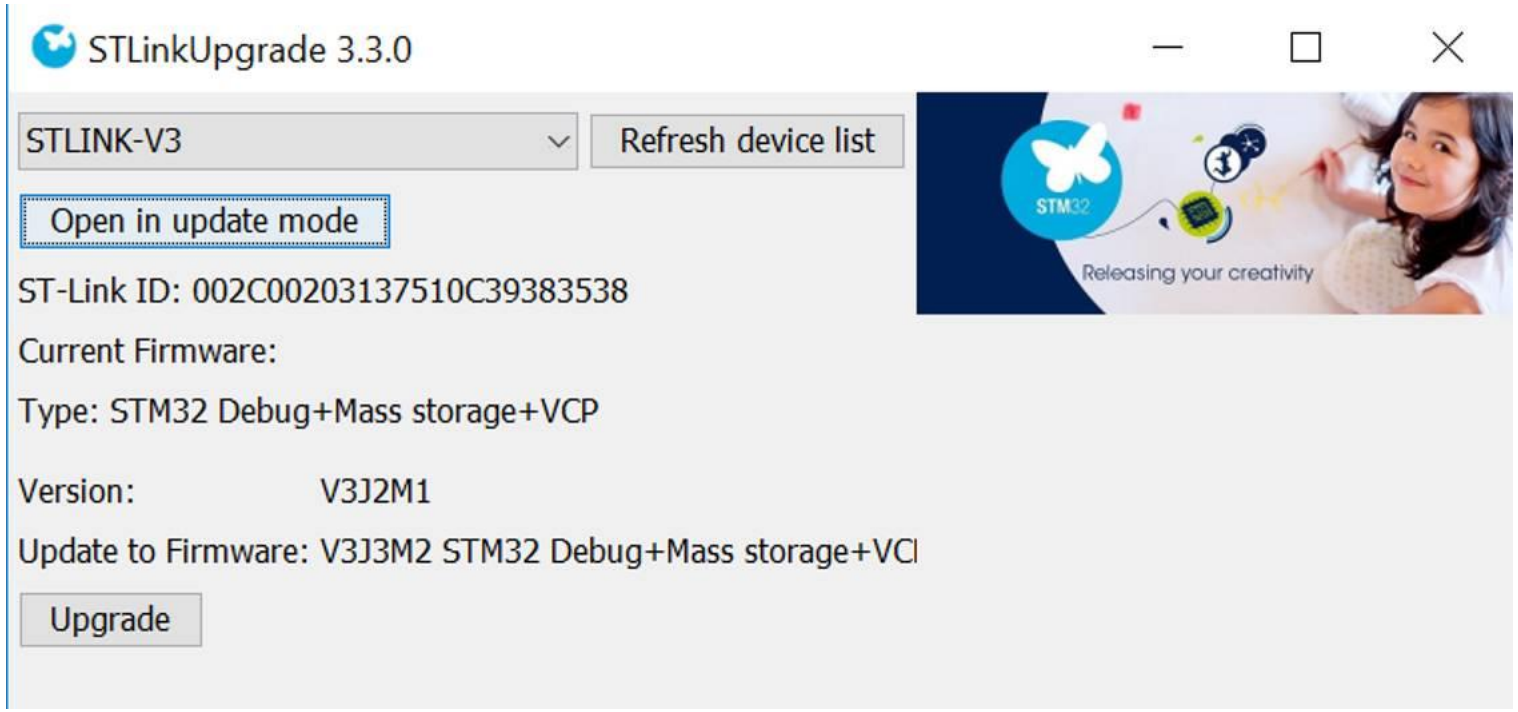
Part Number	Marketing Status	Supplier	Unit Price (US$)	Software Version	Download
ST-LINK-SERVER	Active	ST	-	1.1.1	Download

life.augmented

IAR 下STLINK V2对STM32H7双核调试的支持

如果感觉各个配置都正确，但是还是报下面的错误，请升级STLINK本身的固件，也可以解决该问题。





life.augmented

IAR 下STLINK V2对STM32H7双核调试的支持

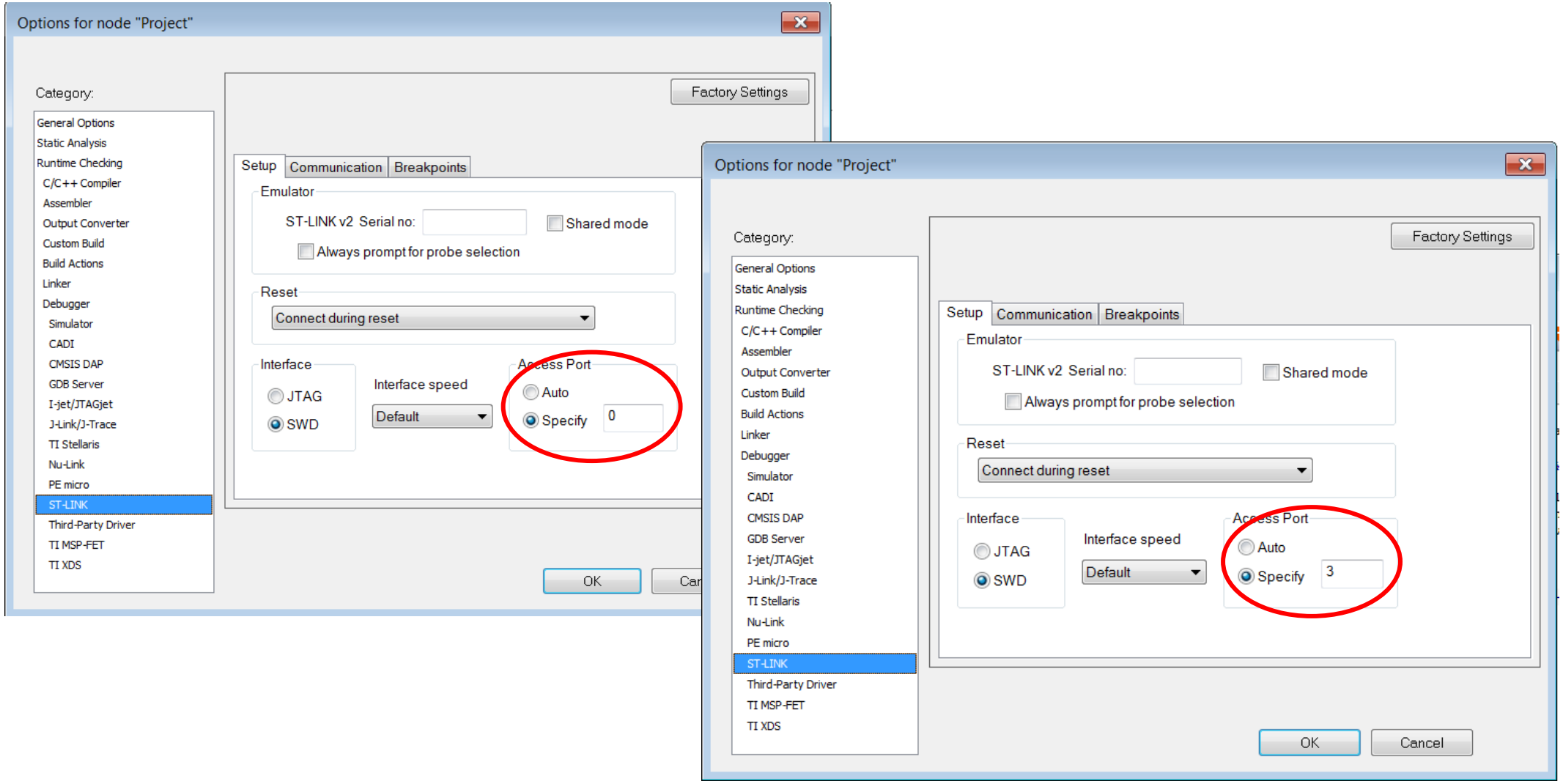
- 命令行的形式如下：
- 对于主项目CM7工程的环境配置：
  - 选择 ST-LINK 从 Project > Options > Debugger > Driver
  - 选择 ST-LINK/V2 从 Project->Options->ST-LINK->Enulator.
  - 添加命令行--stlink_use_server 在Debugger->Extra Options 为CM7(Master) 项目工程.
  - SWD interface and Specified the Access port 设为0 对于CM7 :
- 对于从项目CM4工程的环境：
  - 选择 ST-LINK from Project > Options > Debugger > Driver
  - 选择 ST-LINK/V2 from Project->Options->ST-LINK->Enulator.
  - 添加命令行--stlink_use_server 在Debugger->Extra Options 为CM4(Slave) 项目工程.
  - SWD interface and Specified the Access port 设为3 对于 CM4:

life.augmented

IAR 下STLINK V2对STM32H7双核调试的支持

- Shared mode的形式如下：
- 对于主项目CM7工程的环境配置：
  - 选择 ST-LINK 从 Project > Options > Debugger > Driver
  - 选择 ST-LINK/V2 从 Project->Options->ST-LINK->Enulator.
  - 选择 Share mode 从 Project->Options->ST-LINK->Enulator
  - SWD interface and Specified the Access port 设为0 对于CM7 :
- 对于从项目CM4工程的环境：
  - 选择 ST-LINK from Project > Options > Debugger > Driver
  - 选择 ST-LINK/V2 from Project->Options->ST-LINK->Enulator.
  - 选择 Share mode from Project->Options->ST-LINK->Enulator
  - SWD interface and Specified the Access port 设为3 对于 CM4:

IAR 下STLINK V2对STM32H7双核调试的支持



life.augmented

Project - IAR Embedded Workbench IDE - ARM 8.11.3

File Edit View Project Debug Disassembly ST-Link Tools Window Help

Workspace: STM32H747I_EVAL_CM7

Files

- Project - STM32H747...
  - Doc
  - Drivers
    - BSP
    - CMSIS
      - system_stm32h7...
    - STM32H7xx_HAL_...
  - Example
    - EWARM
      - startup_stm32h7...
  - User
    - CM4
    - CM7
    - Common
  - Output

main.c | system_stm32h7xx.c

main()

```
76   CPU_CACHE_Enable();
77
78
79   /* STM32H7xx HAL lib
80         - Systick time
81           can eventual
82           timer for ex
83           duration sho
84           handled in m
85         - Set NVIC Gro
86         - Low Level In
87       */
88   HAL_Init();
89
90   /* Configure the sy
91   SystemClock_Config(
92  //  HAL_RCCEx_EnableB
93  //  HAL_RCCEx_EnableB
94   /* When system init
95      HSEM notificatio
96   */
97
98    /*HW semaphore Clo
99   __HAL_RCC_HSEM_CLK_
100
101     /*Take HSEM */
102     HAL_HSEM_FastTake
103    /*Release HSEM in
104     HAL_HSEM_Release(
105
106   /* -1- Initialize L
107   BSP_LED_Init(LED1);
108
109   /* -2- Configure EX
110   EXTI15_10_IRQHandle
```

Project - IAR Embedded Workbench IDE - ARM 8.11.3

File Edit View Project Debug Disassembly ST-Link Tools Window Help

Workspace: STM32H747I_EVAL_CM4

Files

- Project - STM32H747...
  - Doc
  - Drivers
    - BSP
    - CMSIS
      - system_stm32h7...
    - STM32H7xx_HAL_...
  - Example
    - EWARM
      - startup_stm32h7...
  - User
    - CM4
    - CM7
    - Common
  - Output

main.c | system_stm32h7xx.c | main.c | stm32h7xx_hal_rcc.h

main()

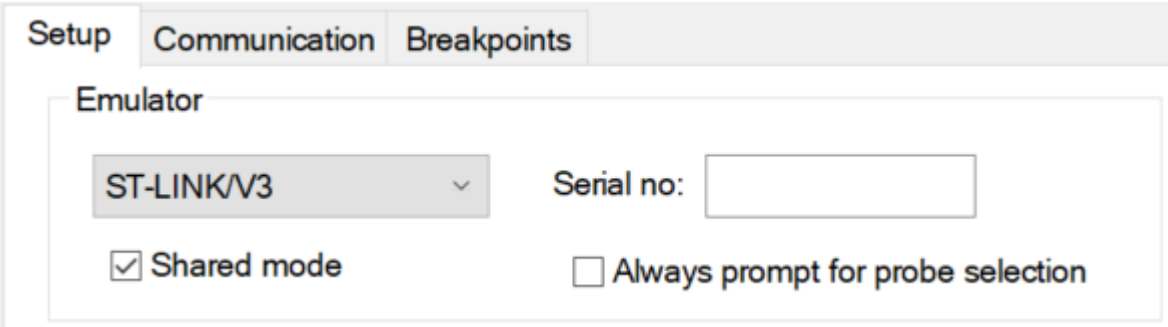
```
72
73   /*
74     Domain D2 goes to STOP mode (Cortex-M4 in deep-sleep) waiting for Cortex-M7 to
75     perform system initialization (system clock config, external memory configuratio
76   */
77 // HAL_PWREx_EnterSTOPMode(PWR_MAINREGULATOR_ON, PWR_SLEEPENTRY_WFE, PWR_D2_DOMAIN);
78
79  /* STM32H7xx HAL library initialization:
80        - Systick timer is configured by default as source of time base, but user
81          can eventually implement his proper time base source (a general purpose
82          timer for example or other time source), keeping in mind that Time base
83          duration should be kept 1ms since PPP_TIMEOUT_VALUEs are defined and
84          handled in milliseconds basis.
85        - Set NVIC Group Priority to 4
86        - Low Level Initialization
87      */
88   HAL_Init();
89
90   /* -1- Initialize LEDs mounted on STM32H747I-EVAL board */
91   BSP_LED_Init(LED3);
92
93   /* Enable and set EXTI lines 15 to 10 Interrupt to the lowest priority */
94   HAL_NVIC_SetPriority(EXTI15_10_IRQn, 2, 0);
95   HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
96
97   /* Infinite loop */
98   while (1)
99   {
100  }
101 }
102
103 HAL_StatusTypeDef HAL_InitTick(uint32_t TickPriority)
104 {
105   /*Configure the SysTick to have interrupt in 1ms time basis*/
106   HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);
```

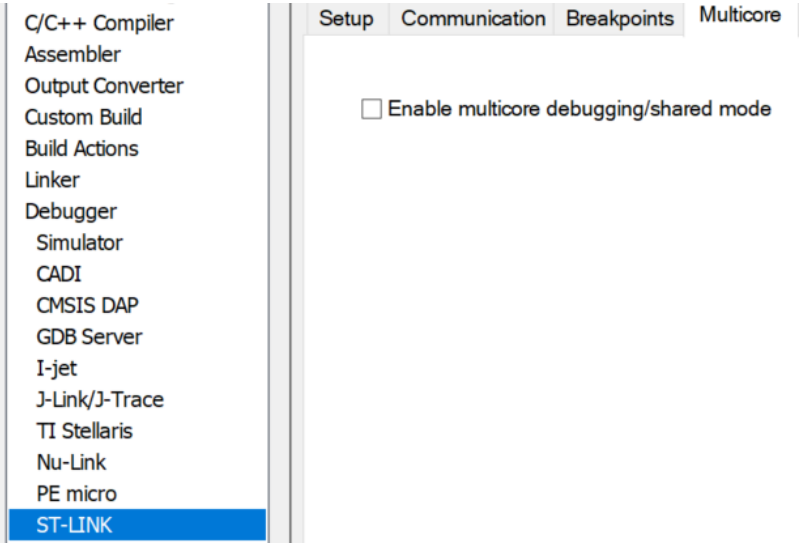
life.augmented

life.augmented

IAR 下STLINK V3对STM32H7双核调试的支持

- 同IAR下STLINK V2的说明和配置。
- 需要确保stlink server是否正确成功安装，可以使用shared mode和命令行的形式，
- 区别仅在于Emulator下面要选择ST-LINK/V3，当前IAR版本默认的是ST-LINK/V2，所以要主动的修改过来，以便与你的调试器相一致。





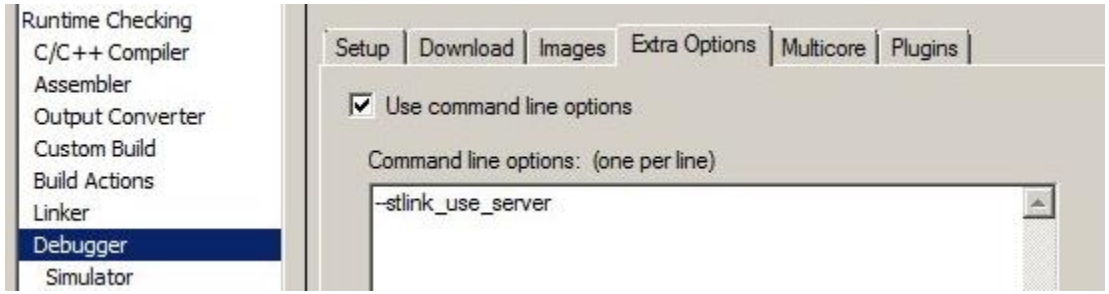
IAR 下STLINK V3对STM32H7双核调试的支持

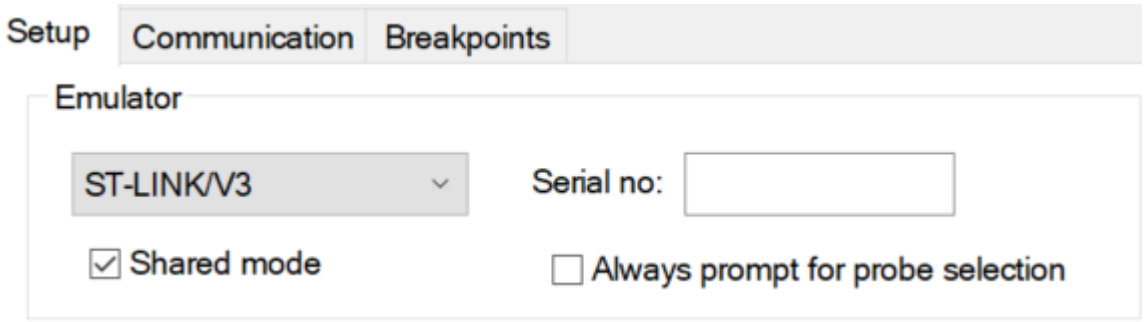
- 命令行的形式如下：
- 对于主项目CM7工程的环境配置：
  - 选择 ST-LINK 从 Project > Options > Debugger > Driver
  - 选择 ST-LINK/V3 从 Project->Options->ST-LINK->Enulator.
  - 添加命令行--stlink_use_server 在Debugger->Extra Options 为CM7(Master) 项目工程.
  - SWD interface and Specified the Access port 设为0 对于CM7 :
- 对于从项目CM4工程的环境：
  - 选择 ST-LINK from Project > Options > Debugger > Driver
  - 选择 ST-LINK/V3 from Project->Options->ST-LINK->Enulator.
  - 添加命令行--stlink_use_server 在Debugger->Extra Options 为CM4(Slave) 项目工程.
  - SWD interface and Specified the Access port 设为3 对于 CM4:

IAR 下STLINK V3对STM32H7双核调试的支持

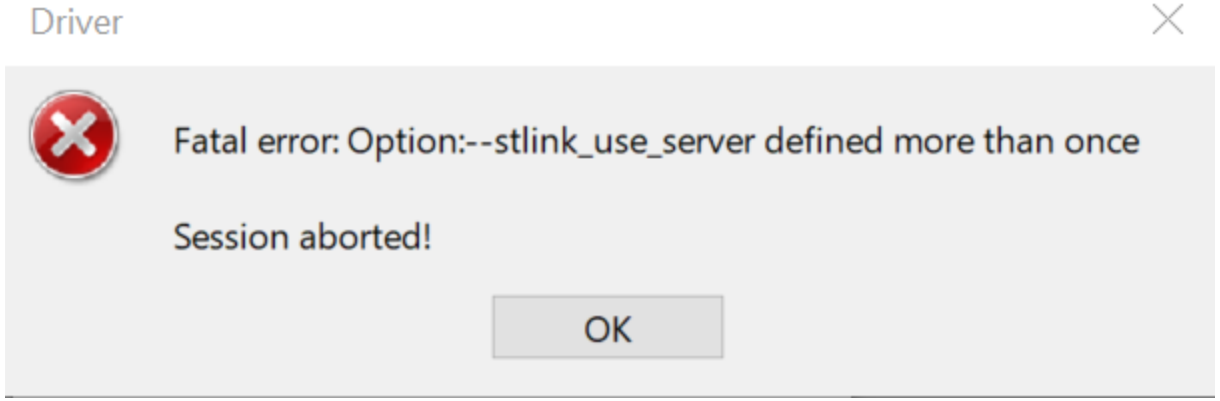
- Shared mode的形式如下：
- 对于主项目CM7工程的环境配置：
  - 选择 ST-LINK 从 Project > Options > Debugger > Driver
  - 选择 ST-LINK/V3 从 Project->Options->ST-LINK->Enulator.
  - 选择 Share mode 从 Project->Options->ST-LINK->Enulator
  - SWD interface and Specified the Access port 设为0 对于CM7 :
- 对于从项目CM4工程的环境：
  - 选择 ST-LINK from Project > Options > Debugger > Driver
  - 选择 ST-LINK/V3 from Project->Options->ST-LINK->Enulator.
  - 选择 Share mode from Project->Options->ST-LINK->Enulator
  - SWD interface and Specified the Access port 设为3 对于 CM4:

IAR 下STLINK V3对STM32H7双核调试的支持



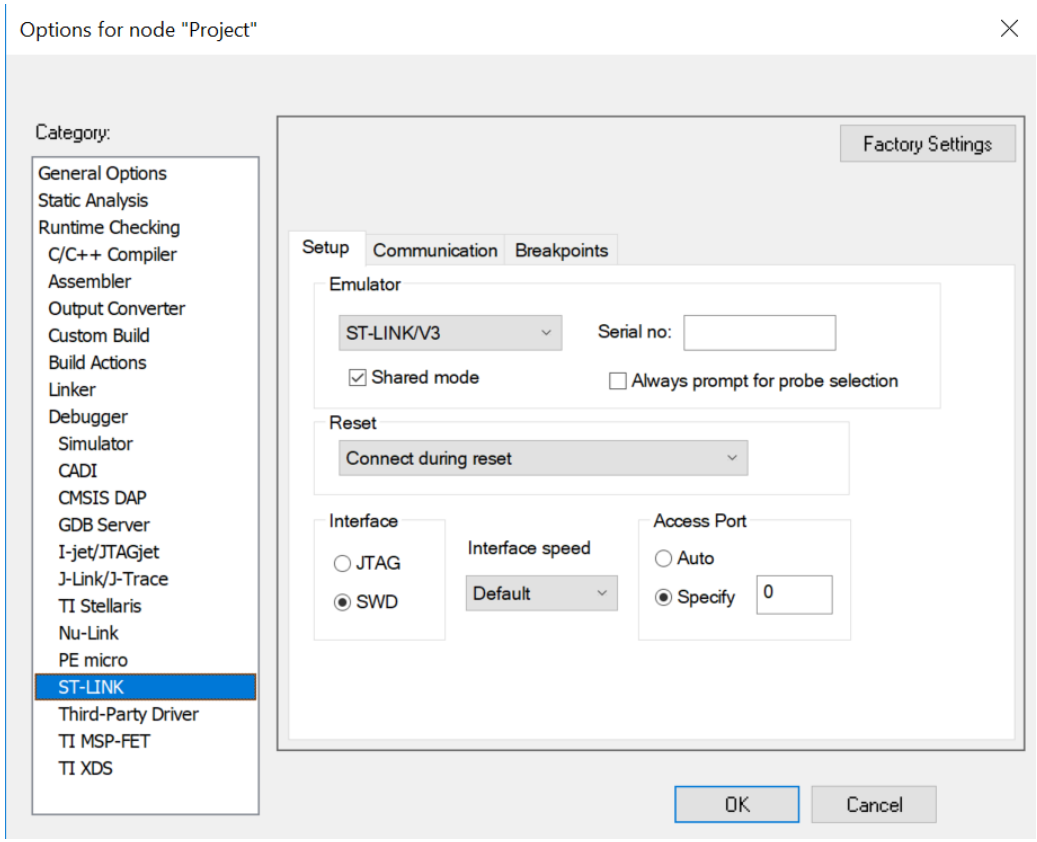


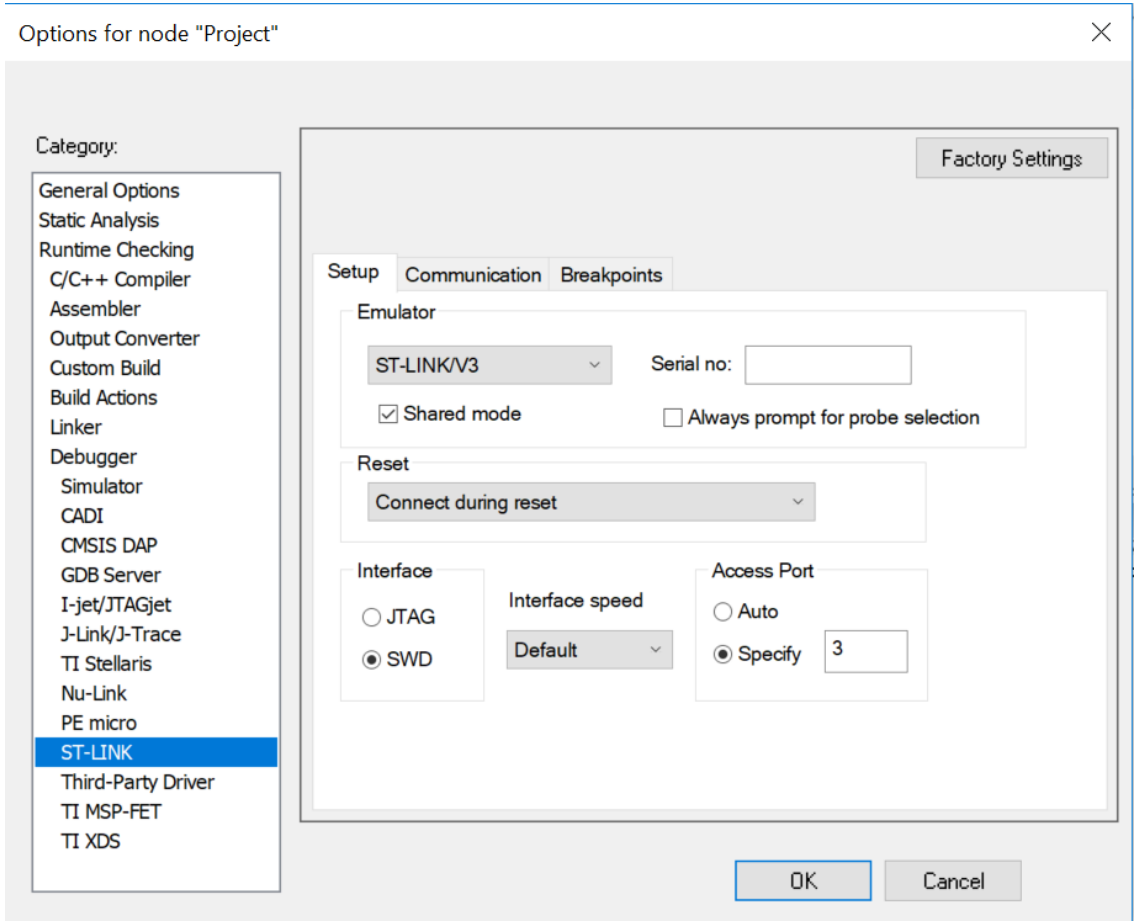
STLINK V2或STLINK V3选择shared mode之后不再需要--stlink_use_server的命令行，否则会报错。从这一点可以看出shared mode和命令行的作用是一样，推荐使用复选框shared mode



life.augmented

IAR 下STLINK V3对STM32H7双核调试的支持





life.augmented

IAR 下STLINK V3对STM32H7双核调试的支持

Project - IAR Embedded Workbench IDE - Arm 8.32.2

File Edit View Project Debug Disassembly ST-Link Tools Window Help

Workspace

STM32H745I_DISCO_CM7

Files

Project - STM32H745I_...

Doc

Drivers

Example

Output

main.c

main()

```
43  /* Private functions ---------------------------------------------------------*/
44
45  /**
46    * @brief  Main program
47    * @param  None
48    * @retval None
49    */
50  int main(void)
51  {
52    /* System Init, System clock, voltage scaling and L1-Cache configuration are d
```

Disassembly

Go to

Memory

```
0x8000ec0: 0x58020c00
0x8000ec4: 0x58021000
0x8000ec8: 0x58021400
0x8000ecc: 0x58021800
0x8000ed0: 0x58021c00
0x8000ed4: 0x58022000
0x8000ed8: 0x58000088
int main(void)
```

Project - IAR Embedded Workbench IDE - Arm 8.32.2

File Edit View Project Debug Disassembly ST-Link Tools Window Help

Workspace

STM32H745I_DISCO_CM4

Files

Project - STM32H745I_...

Doc

Drivers

Example

Output

Project

main.c

main()

```
39  /* Private functions ---------------------------------------------------------*/
40
41  /**
42    * @brief  Main program
43    * @param  None
44    * @retval None
45    */
46  int main(void)
47  {
48   /*HW semaphore Clock enable*/
49    __HAL_RCC_HSEM_CLK_ENABLE();
50
51    /* Activate HSEM notification for Cortex-M4*/
52    HAL_HSEM_ActivateNotification(__HAL_HSEM_SEMID_TO_MASK(HSEM_ID_0));
53
54    /*
```

Disassembly

Go to

Memory

```
0x81004ac: 0x58020c00
0x81004b0: 0x58021000
0x81004b4: 0x58021400
0x81004b8: 0x58021800
0x81004bc: 0x58021c00
0x81004c0: 0x58022000
0x81004c4: 0x580000c8
int main(void)
{
main:
0x81004c8: 0xb580
__HAL_RCC_HSEM_CLK_ENABLE
0x81004ca: 0x4819
```

life.augmented

MDK-ARM 对于STM32H7双核调试的支持

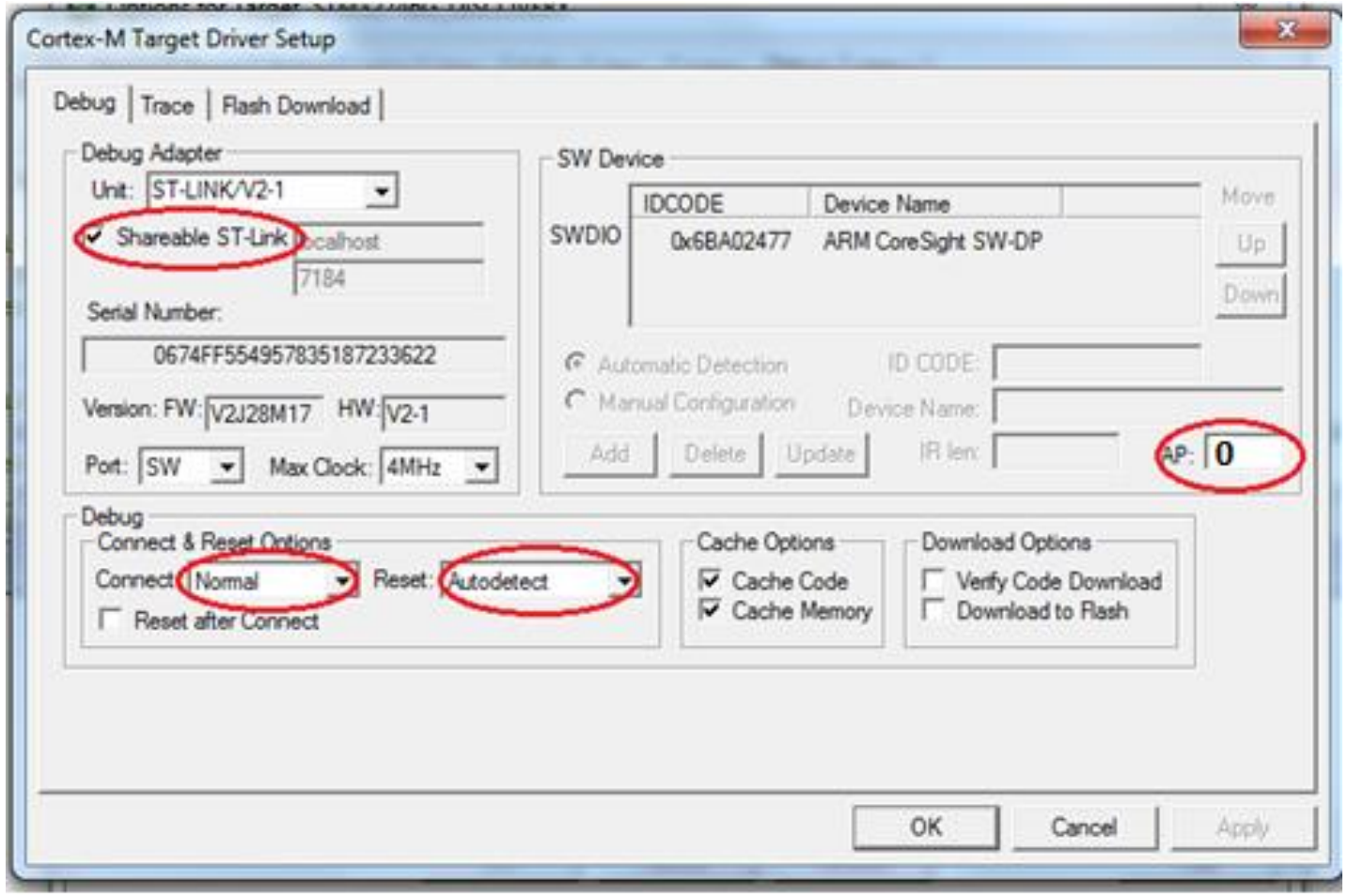
- 请使用最新版本的Keil,并确保使用的是FW v2J28或更新版本的ST-LINK/V2 。
  - Keil 当前最新版本Version 5.28a
    - https://www.keil.com/download/product/

- 需要在一个workspace下建立为每个内核建立一个项目工程
- 使能Dual core双核的调试通过勾选复选框实现：Shareable ST-LINK
  - 确保stlink_server的安装
  - STLINK/V2、STLINK/V3均可支持双核的调试
  - 可以利用ULINK-ME 进行双核的调试。(还未自测)

MDK-ARM 下STLINK V2对STM32H7双核调试的支持

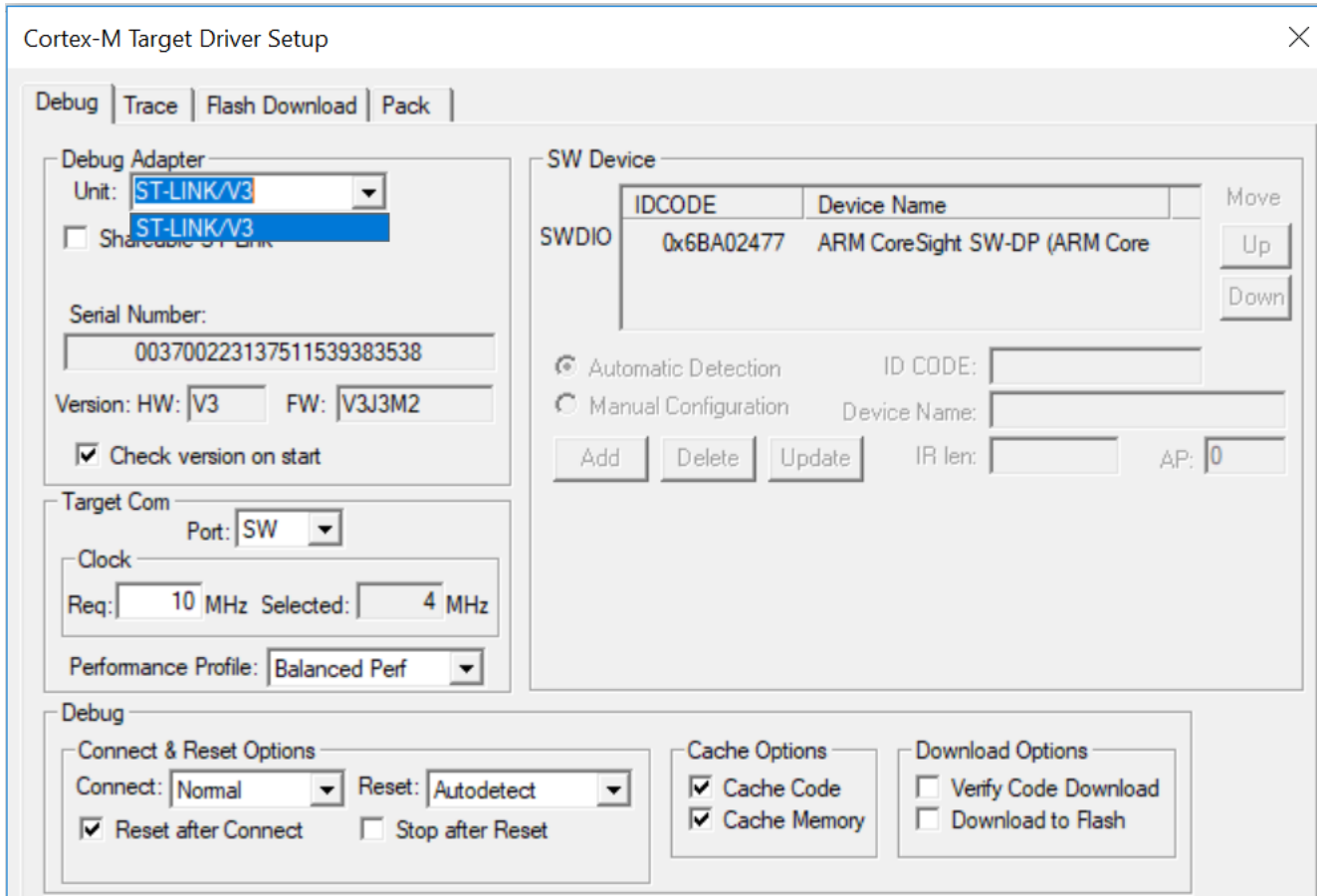
旧版本的Keil如果需要支持双核调试，需要安装补丁包，安装后才会出现shareable ST-LINK，这里不做说明，请使用最新版的Keil。

勾选上shareable ST-LINK并配置AP = 0或3即可。



MDK-ARM 下STLINK V3对STM32H7双核调试的支持

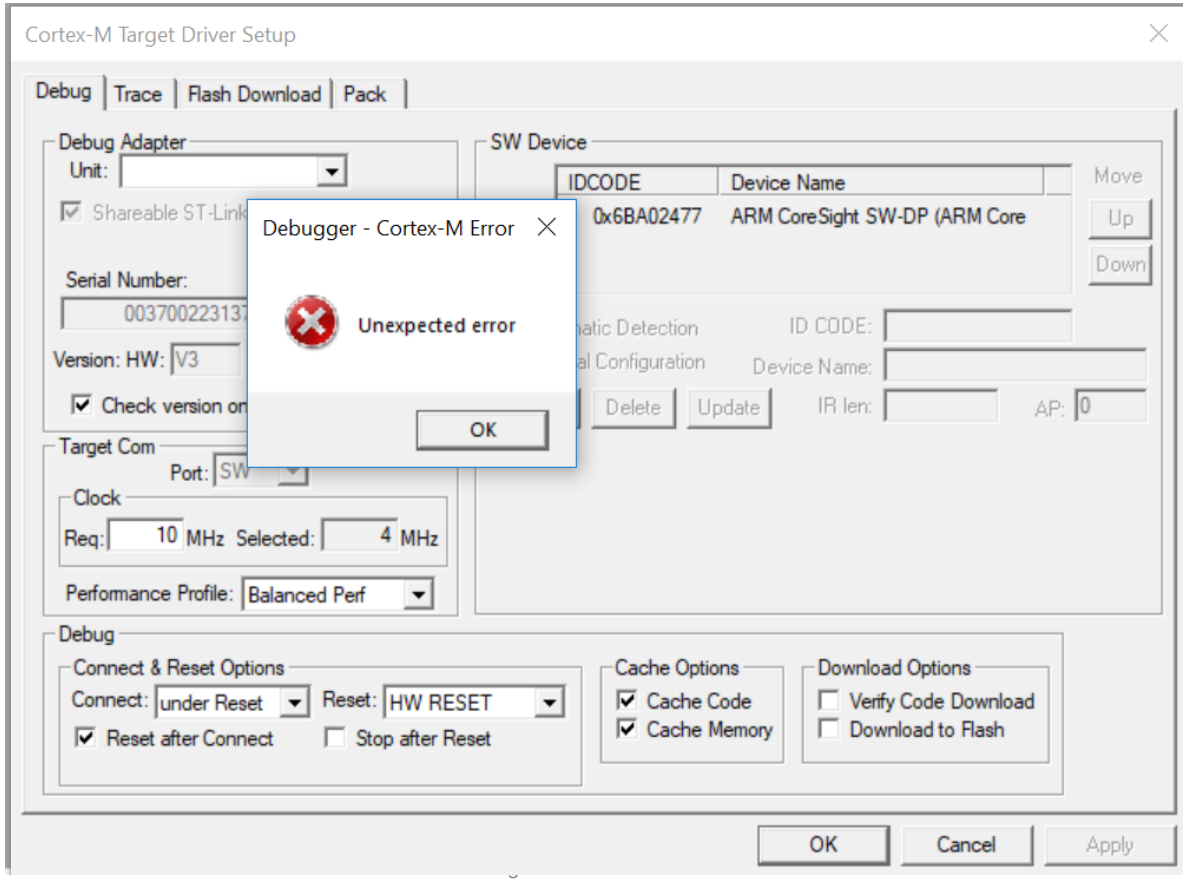
- 新版的Keil支持双核调试，不需要安装补丁包，会自动出现shareable ST-LINK 但是从下图的下拉框可以看出，新版的Keil不支持ST-LINK/V2，仅仅支持ST-LINK/V3
- 勾选上shareable ST-LINK即可。AP = 0或3的配置Keil自动完成，为灰色，用户无法修改。



life.augmented

MDK-ARM 下STLINK V3对STM32H7双核调试的支持

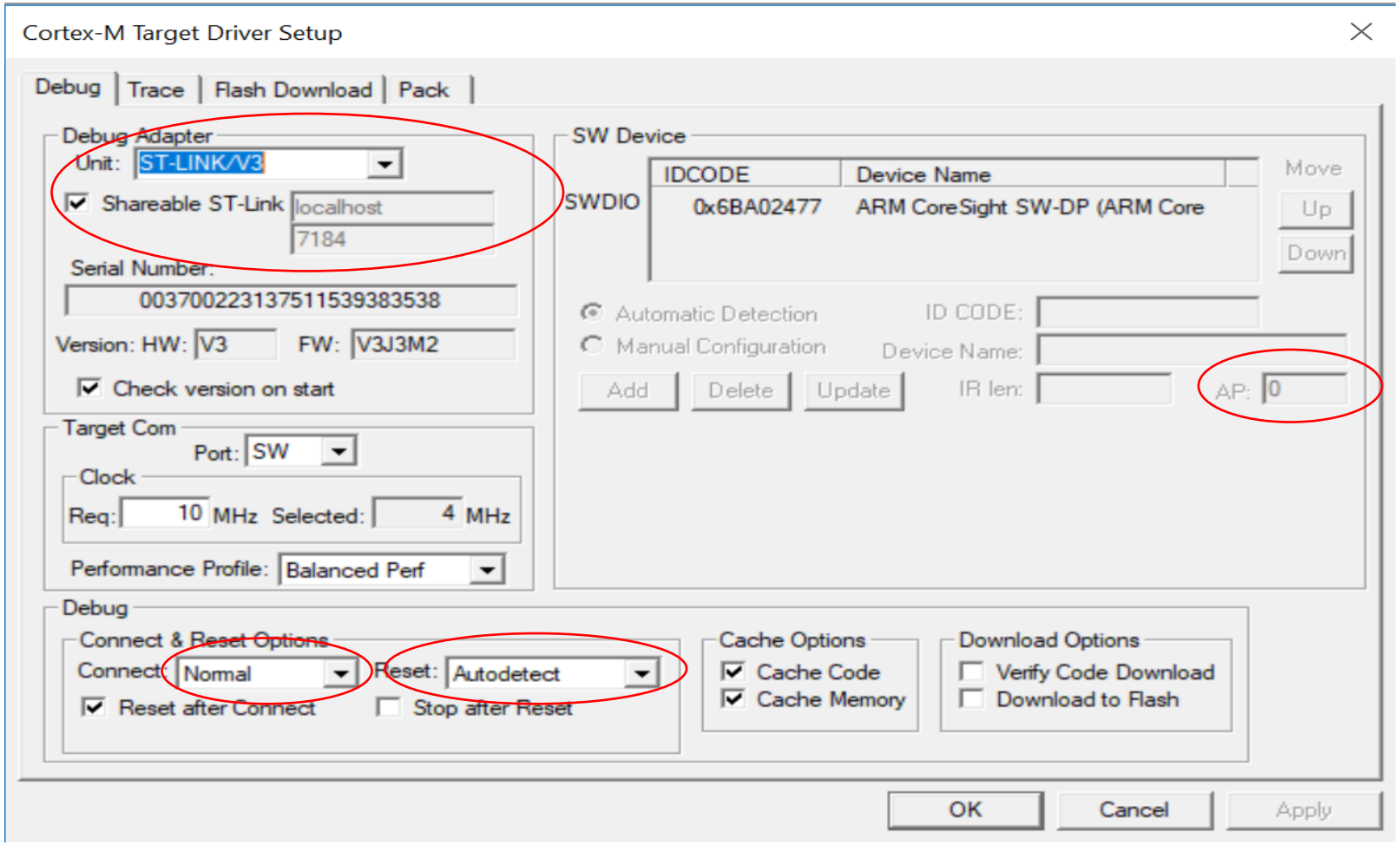
- 勾选shareable ST-LINK是如果弹出如下的错误，且Unit处的内容空白，则说明stlink server没有安装好。



life.augmented

MDK-ARM 下STLINK V3对STM32H双核调试的支持

- 此时需要的是安装ST官方的stlink server安装包。
- 对于Keil只需勾选上ST-LINK/V3即可，它会自动配置AP(灰色，用户不可配置)



life.augmented

MDK-ARM 下STLINK V3对STM32H7双核调试的支持

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

USBD_ITF_MAX_NUM

Registers

Register

Core

R0 0x0
R1 0x1
R2 0x0
R3 0x0
R4 0x0
R5 0x0
R6 0x0
R7 0x0
R8 0x0
R9 0x0
R10 0x0
R11 0x0
R12 0x0
R13 (SP) 0x1
R14 (LR) 0x0
R15 (PC) 0x0
xPSR 0x6

Banked
System
Internal

Project Registers

Disassembly

```
158:    CPU_CACHE_Enable()
159:
160:    /* Wait until CPU2
161:    timeout = 0xFFFF;
```

startup_stm32h747xx.s | main

```
153    /* System Init, Sy
154       in the meantime
155    */
156
157    /* Enable the CPU
158    CPU_CACHE_Enable()
159
160    /* Wait until CPU2
161    timeout = 0xFFFF;
162    while((__HAL_RCC_
163    if ( timeout < 0 )
164    {
165      Error_Handler()
166    }
167
168    /* STM32H7xx HAL
169       - Systick ti
```

```
48:    __HAL_RCC_HSEM_CLK_ENABLE();
49:
50:   /* Activate HSEM notification for Cortex-M4*/
0x08100700 4812      LDR      r0,[pc,#72]   ; @0x0810074C
45: {
46:
47:   /*HW semaphore Clock enable*/
```

main.c

```
43    */
44   int main(void)
45   {
46
47    /*HW semaphore Clock enable*/
48      __HAL_RCC_HSEM_CLK_ENABLE();
49
50    /* Activate HSEM notification for Cortex-M4*/
51     HAL_HSEM_ActivateNotification(__HAL_HSEM_SEMID_TO_MASK(HSEM_ID_0));
52
53    /*
54      Domain D2 goes to STOP mode (Cortex-M4 in deep-sleep) waiting for
55      perform system initialization (system clock config, external memo
56    */
```

Command

```
_DISCO_CM7\\Exe\\STM32H747I_EVAL_CM7.axf"
```

Command

```
Load "STM32H747I_DISCO_CM4\\Exe\\STM32H747I_EVAL_CM4.axf"
```

Call Stack + Locals

Name	Locatio
main	0x08100
tmpreg	<not in

小结

- 1、Access port的正确选择
- 2、STLINK-Server的正确安装
- 3、share mode的勾选
- 4、AN4989: STM32 microcontroller debug toolbox
- 5、AN5286: STM32H7x5/x7 dual-core microcontroller debugging
- 6 、要说明一点的是，目前STLINK V2或V3还不支持菊花链的链接

life.augmented

STM32H7- HSEM

HSEM概述

- 管理访问权限和用于同步
  - 不同的进程运行在相同的CPU上
  - 不同的CPUs之间
- 32个信号量
- 2种锁的机制
  - 2-step write, read back lock
  - 1-step read lock
- 信号量释放中断生成

运用优势

- 防止共享资源访问冲突
- 确保进程之间同步

Boot启动模式

- 在STM32H7x5和STM32H7x7产品线, 两个核CM7与CM4根据Option Bytes选项字节的配置可以启动其中一个内核 或两个内核同时启动：

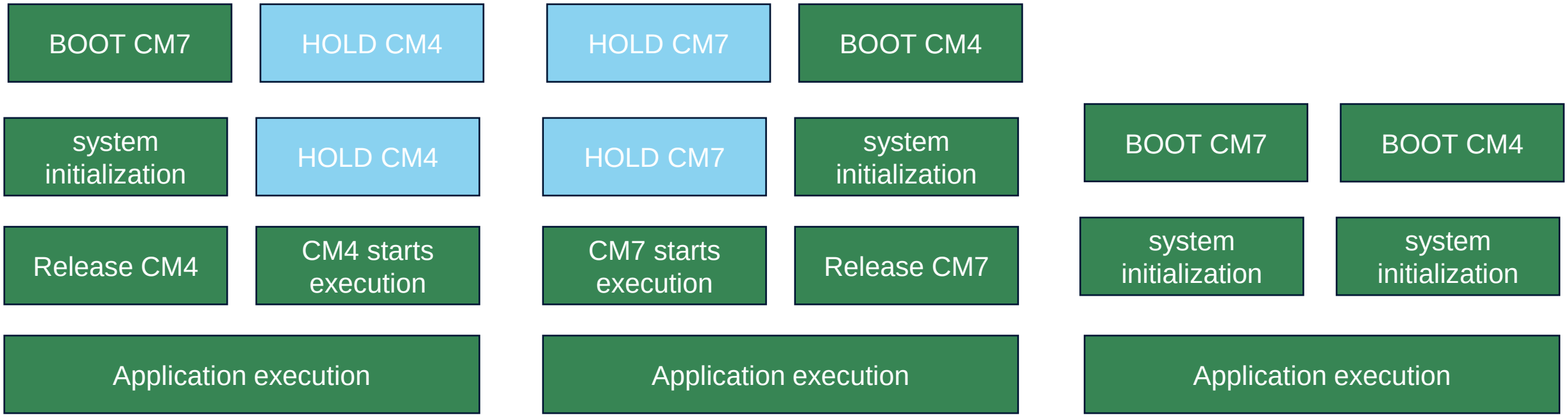
BCM7	BCM4	Boot order
0	0	Cortex®-M7 启动、Cortex®-M4 clock is gated
0	1	Cortex®-M7 clock is gated 、Cortex®-M4 启动
1	0	Cortex®-M7 启动、Cortex®-M4 clock is gated
1	1	Cortex®-M7 和 Cortex®-M4 一起启动

- 启动的CPU被定义为主机master, 它负责系统初始化. 而另一个CPU则执行特定的初始化.

- 允许在上电时实现安全启动和正确的初始化。

life.augmented

Boot 模式：boot 顺序

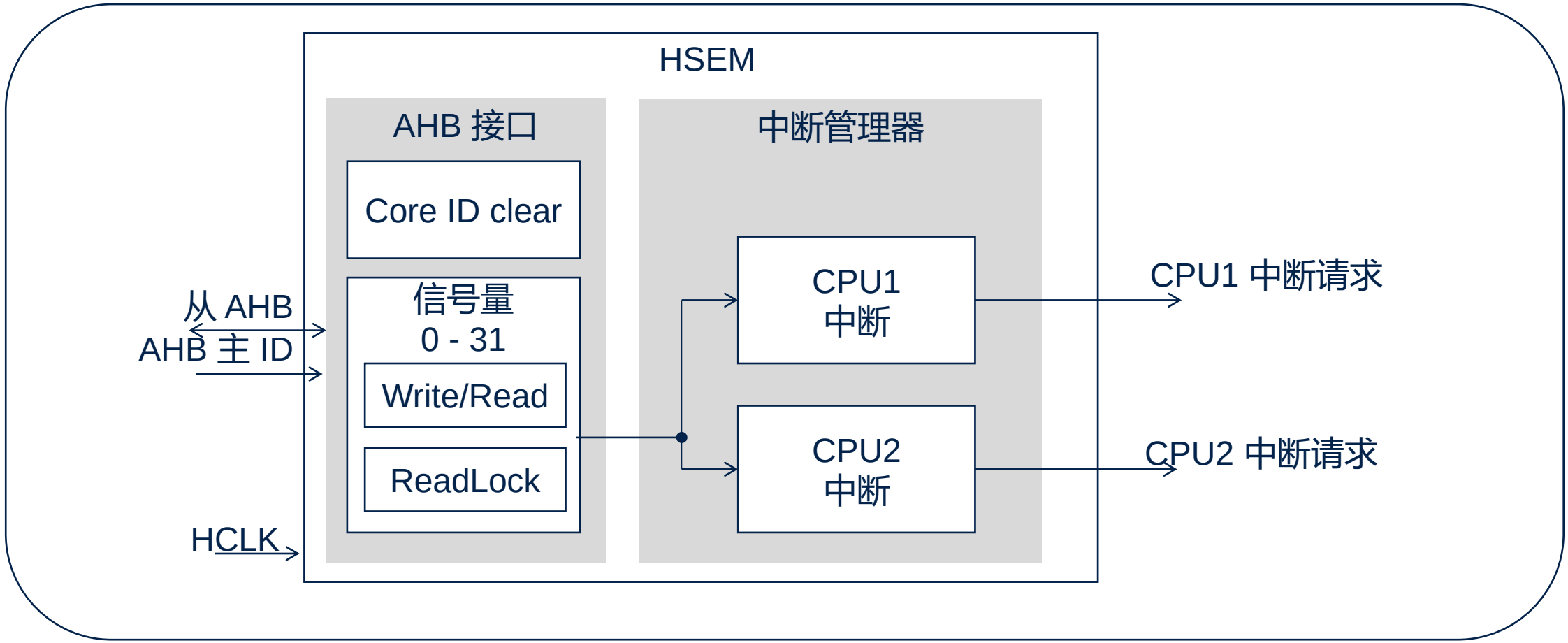


BCM7=1 & BCM4=0

BCM7=0 & BCM4=1

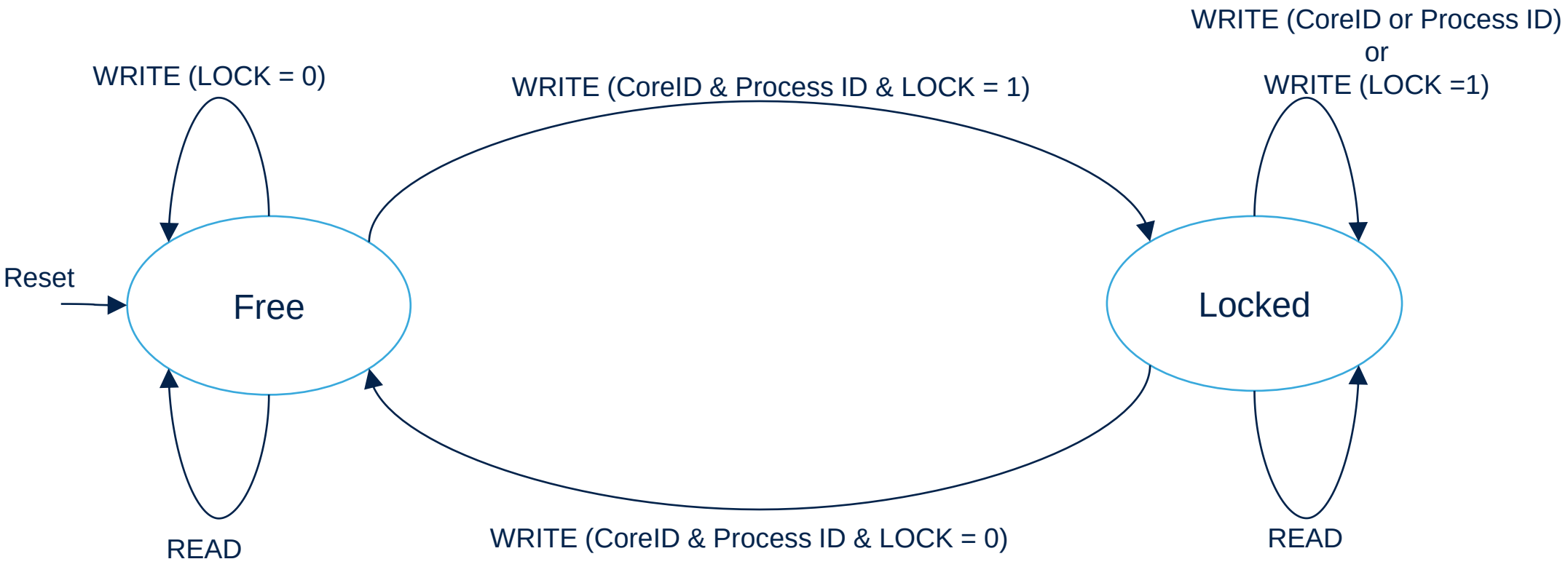
BCM7=BCM4

模块框图


HSEM
AHB 接口
Core ID clear
信号量
0 - 31
Write/Read
ReadLock
中断管理器
CPU1
中断
CPU2
中断
从 AHB
AHB 主 ID
HCLK
CPU1 中断请求
CPU2 中断请求

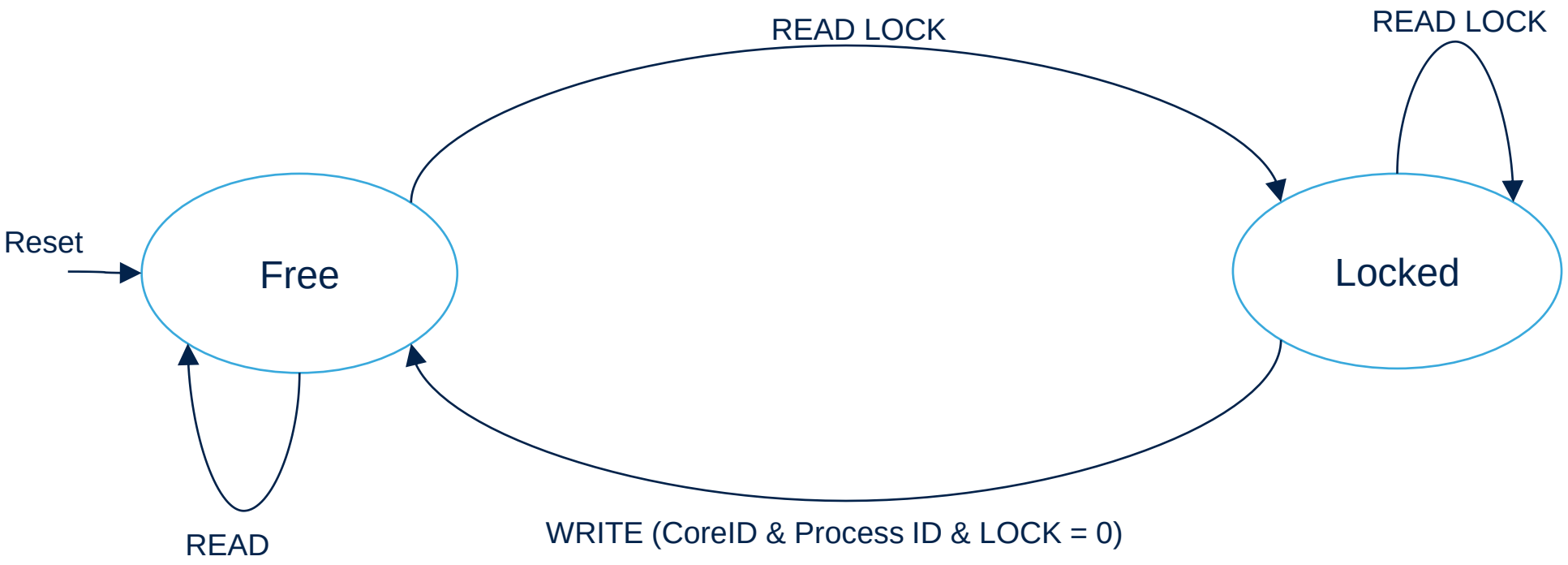
信号量的处理过程

2-步写锁



信号量的处理过程

1-步读锁



1 步读锁

信号量只有在空闲时才能被锁定。当一步读锁定一个自由信号量时，PROCID将为'0’。

如果一步读锁锁的是一个已经被锁定的信号量时将返回锁定该信号量的的COREID和PROCID。所有Read锁，包括锁定信号量的第一个锁，都将返回锁定或锁定信号量的COREID。

如果同一AHB总线主站的多个进程使用一步读锁过程，则使用相同信号量的所有进程将读取相同的状态。当只有一个进程锁定信号量时，该AHB总线主机的每个进程都将以COREID自身锁定的方式读取信号量。

AHB总线master ID 验证

HSEM只允许授权的AHB总线master IDs去锁定和解锁信号量。

- – 来自未经授权的AHB master IDs 的访问会被丢弃，不会锁定信号量.

- –未经授权的AHB master IDs 的访问读取HSEM_RLRx寄存器的值，会根据信号量状态返回下列值:
  - 当信号量为自由空闲状态时，返回所有为 ‘0’s
  - 当信号量被锁定后，将返回HSEM_RLRx寄存器的值.

- 只有有效的总线master IDs才能够写入HSEM_CR 寄存器并清除COREID的相关任何信号量.
  –来自未经授权的AHB master IDs 的访问会被丢弃，不会清除COREID相关的任何信号量.

Table 92. Authorized AHB bus master IDs

Bus master 0 (CPU1)	Bus master 1 (CPU2)
COREID = 3	COREID = 1

信号量的处理过程

General Core ID 清除

- 所有通过 CPU锁定的信号量，可以通过Semaphore clear register清除.
  - 当CPU不再正常工作时，允许清除锁定信号量.
- 写Core ID and Key value.
  - HSEM clear register (HSEM_CR) and HSEM interrupt clear register (HSEM_KEYR),也就是先往HSEM_KEYR寄存器里预设一个KEY值，然后再往HSEM_CR寄存器里写预设的KEY值，则相应的CoreID的所有信号量将被释放。

```
/**
  * @brief  Release All semaphore used by a given Master .
  * @param  Key: Semaphore Key , value from 0 to 0xFFFF
  * @param  CoreID: CoreID of the CPU that is using semaphores to be released
  * @retval None
  */
void HAL_HSEM_ReleaseAll(uint32_t Key, uint32_t CoreID)
{
  assert_param(IS_HSEM_KEY(Key));
  assert_param(IS_HSEM_COREID(CoreID));

  HSEM->CR = ((Key << HSEM_CR_KEY_Pos) | (CoreID << HSEM_CR_COREID_Pos));
}
```

```
/**
  * @brief  Set semaphore Key .
  * @param  Key: Semaphore Key , value from 0 to 0xFFFF
  * @retval None
  */
void  HAL_HSEM_SetClearKey(uint32_t Key)
{
  assert_param(IS_HSEM_KEY(Key));

  MODIFY_REG(HSEM->KEYR, HSEM_KEYR_KEY, (Key << HSEM_KEYR_KEY_Pos));

}
```

life.augmented

主要寄存器的说明

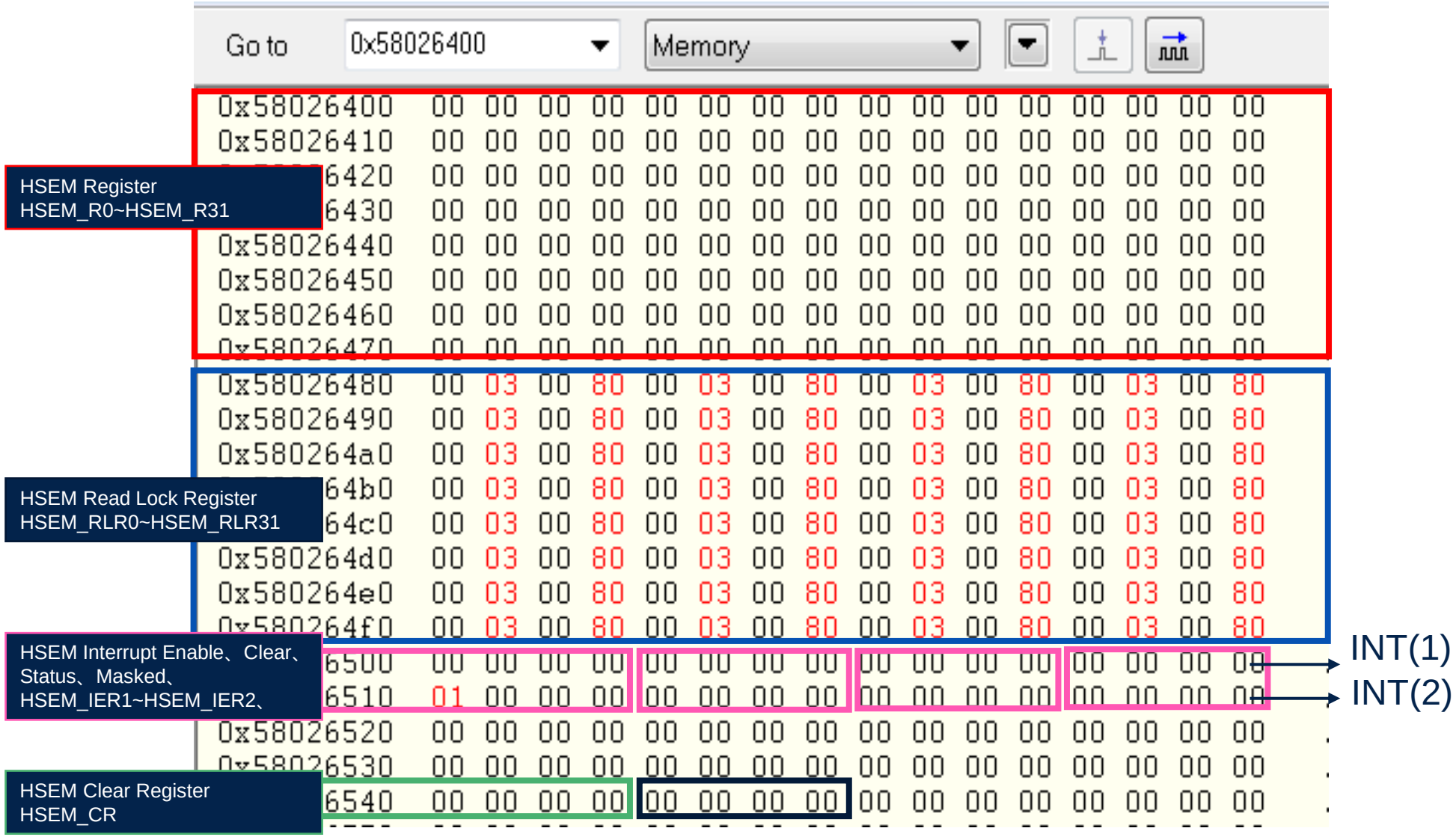
寄存器	说明	备注
HSEM_CnIER	With the Interrupt enable (HSEM_CnIER) the semaphores affecting the interrupt line can be enabled. Disabled (masked) semaphore interrupts will not set the masked interrupt status for that semaphore, and will not generate an interrupt on the interrupt line	设置0~31号各号的信号量的释放中断的使能， 0表示释放无中断， 1表示释放会产生中断。
HSEM_CnICR	will clear the Interrupt status and Masked interrupt status of the associated semaphore for the interrupt line	清除HSEM_CnISR中的Status与HSEM_CnMISR中的MaskedStatus状态的标志， 读始终为0，写1清0。
HSEM_CnISR	The Interrupt status (HSEM_CnISR) mirrors the semaphore Interrupt status of the interrupt line before the Enable.	该状态寄存器代表的是HSEM_CnIER使能之前的信号量的状态，因为对于未使能中断的信号量，如果释放，虽然不能产生中断，但是Status会置位。可以结合RM0399 Figure 74理解，被置位代表相应的信号量从锁住的状态被释放了。只读。
HSEM_CnMISR	The Masked interrupt status (HSEM_CnMISR) only mirrors the semaphore Interrupt status of the enabled semaphore interrupts on the interrupt line. All Masked interrupt status of the enabled semaphore need to be cleared in order to clear the interrupt line.	该状态寄存器代表的是使能中断之后的信号量，如果被释放了，MaskedStatus将被置位，也就是说如果信号量未使能中断，被释放后只置位HSEM_CnISR中的Status，但是如果信号量使能中断，被释放后，不仅置位HSEM_CnISR中的Status，也会置位HSEM_CnMISR中的MaskedStatus。只读。

life.augmented

共享读、互斥写

- 其实就是共享读，互斥写
- 保证了数据的一致性
  - 一步读锁
    - 一步读锁的具体实现函数只关心内核即CoreID，所以更多的用于不同的核之间。
    - 一步锁共享读，意味着一个资源可以同时被一个core上的多个进程/线程共享读取，访问。
      譬如，这个资源是一个缓冲区，这个core上的每一个进程共享读的数据都是相同的。
    - 如CM7用一步锁锁定了一个LED1的资源，则CM7内的多个线程或进程都可以对LED1进行读写操作访问，而CM4则最多只能读取（最好也不要读操作，等待信号量释放再去执行），不应写操作，否则会造成数据不一致，也会失去HSEM这个类似于硬件全局变量的作用。
  - 两步写锁
    - 两步写锁不仅关系CoreID，也关心ProcessID，所以不同的或者相同的核之间都适合。
    - 两步互斥写锁，则是意味着仅有一个core上的一个进程可以访问这个资源，
      例如，cpu 1上的 process 1把一个缓冲区使用了写锁，则其他进程（无论是cpu 1还是cpu 2）上的进程无法访问，这就保证了数据的一致性

life.augmented

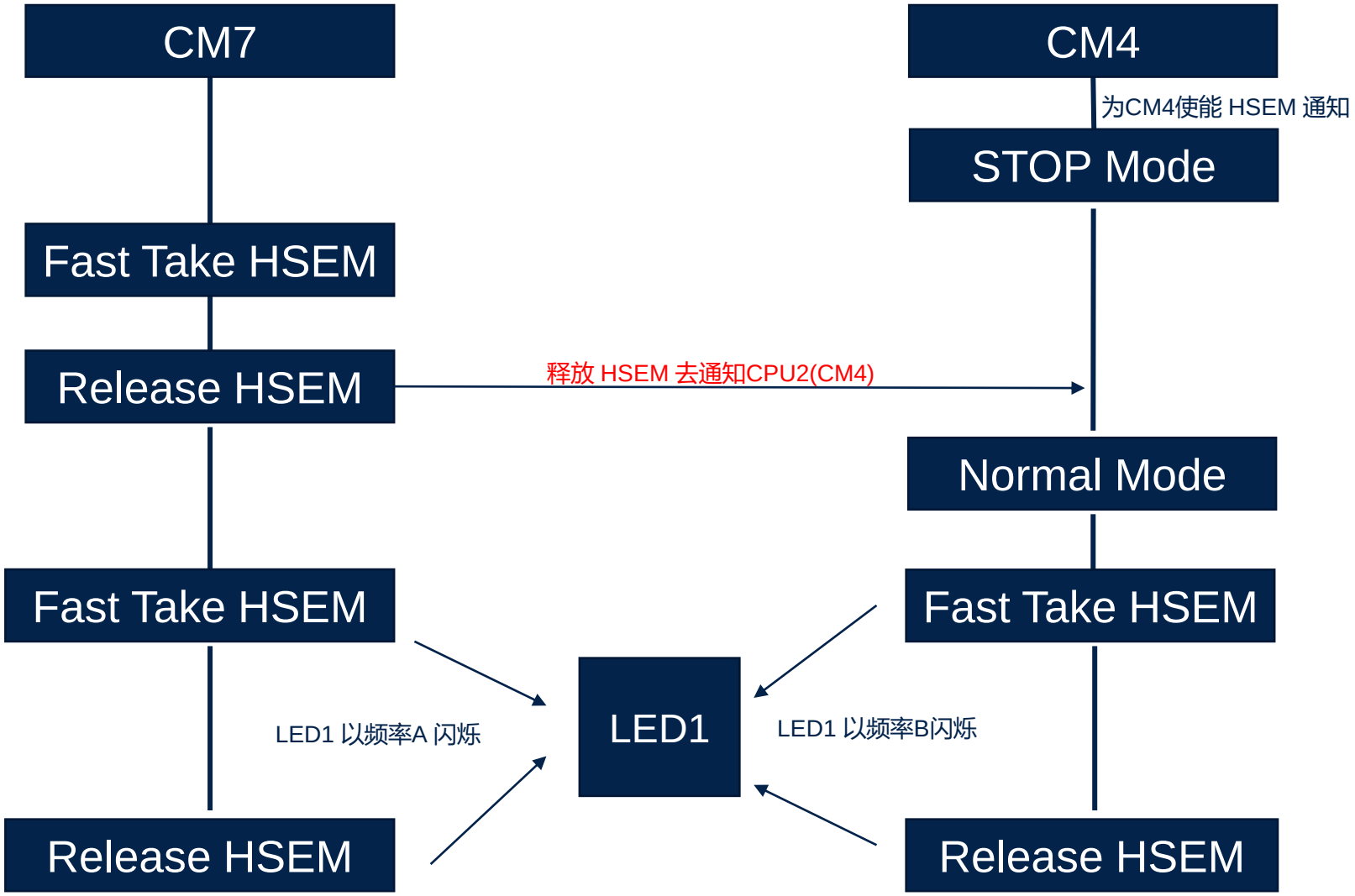


调试的时候不适合通过编译器观察寄存器，因为通过编译器看寄存器其实就是编译器读寄存器的一种，因为一步读锁，一读就会锁住寄存器，所以和调试的目的相违背，可通过串口打印的无关消息的方法查看。

life.augmented

HSEM应用例程

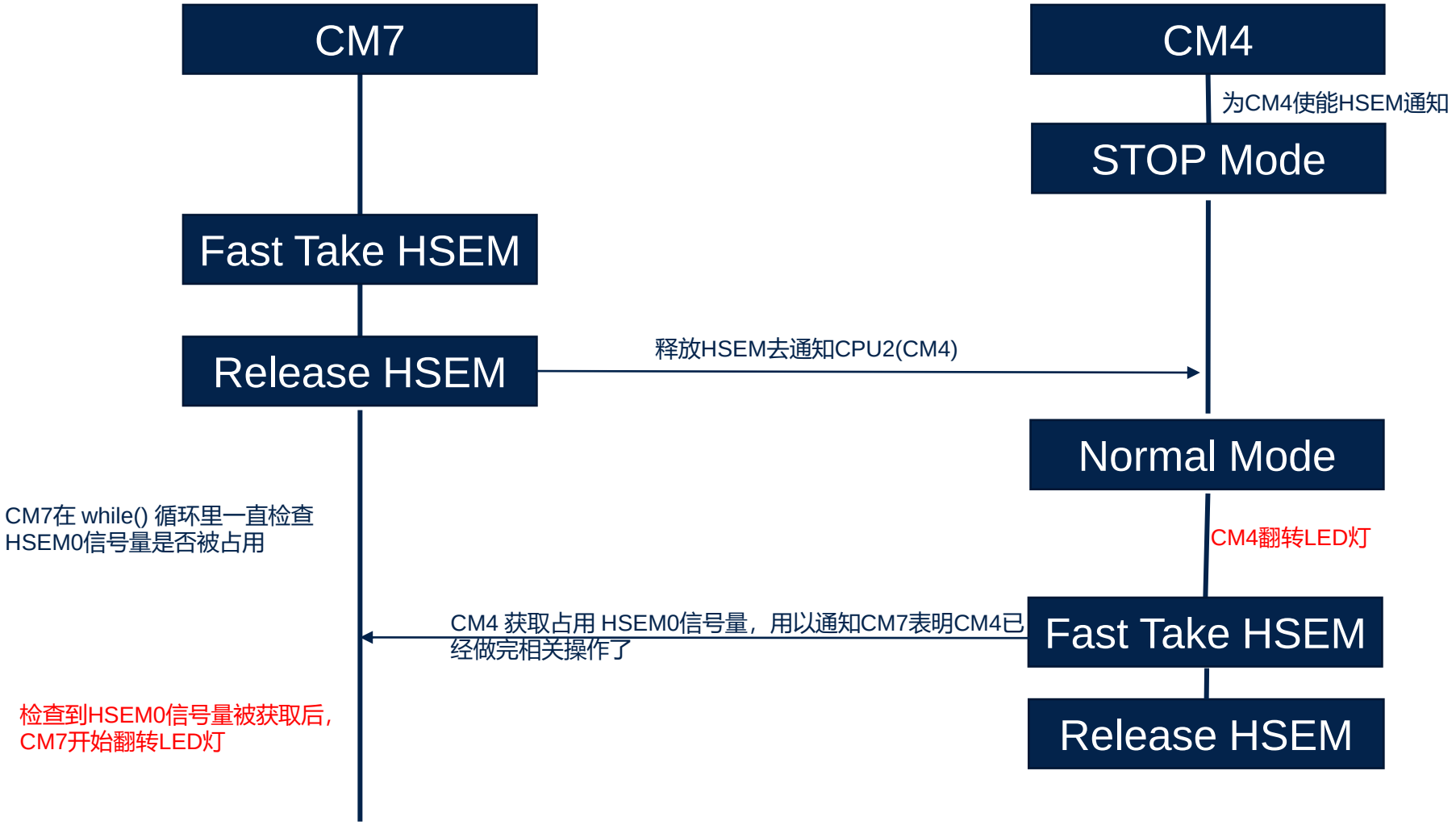
1、HSEM_CoreNotification



life.augmented

HSEM应用例程

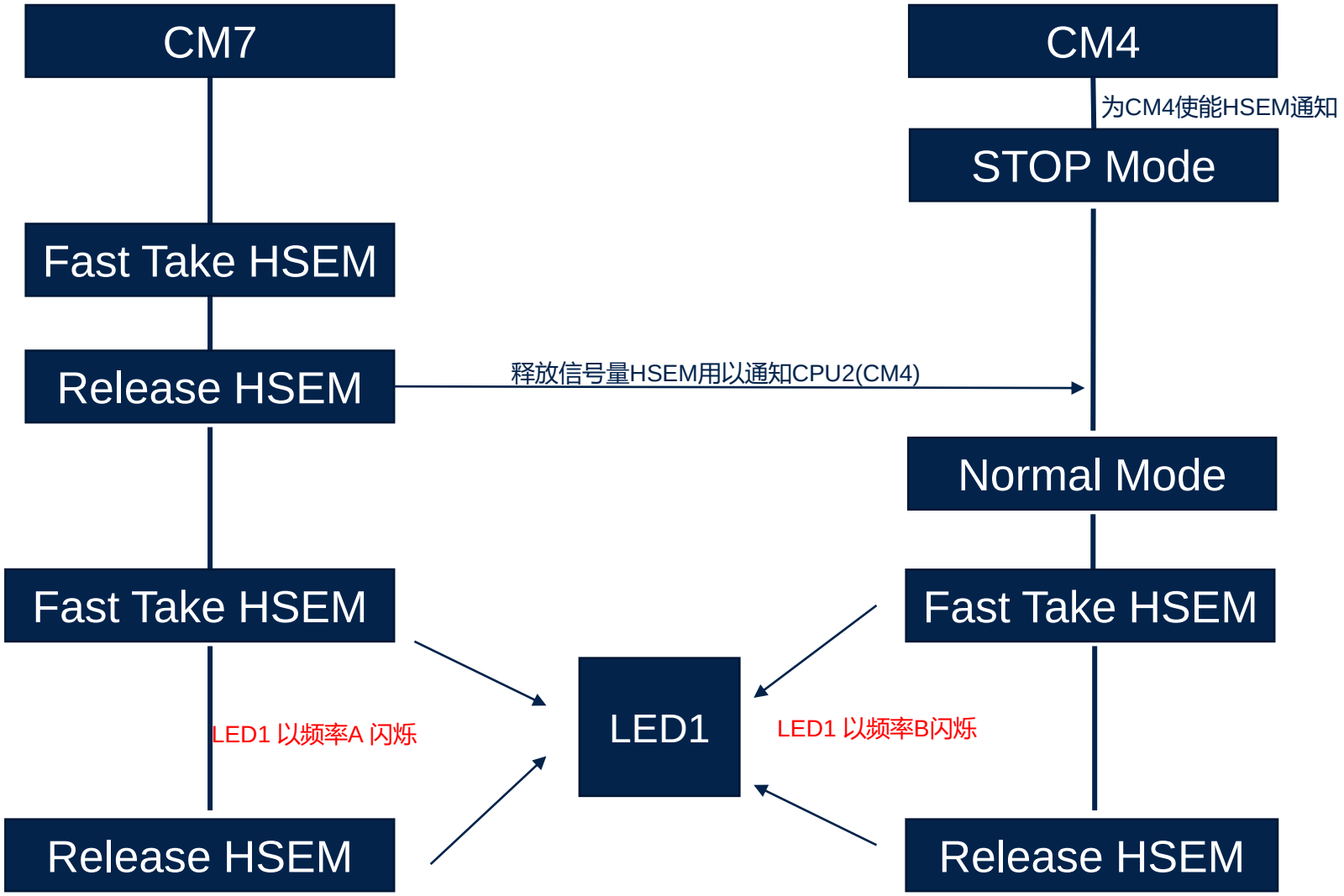
2、HSEM_CoreSync



life.augmented

HSEM应用例程

3、 HSEM_ResourceSharing



life.augmented

life.augmented

STM32H7 – IPC

Inter-Processor Communication内核通信

为什么 Inter-procesors communication (IPC)?

- 运行在不同CPU内核上的应用可能是独立的，也可能是相互协作的
- 很多情况下，需要一种机制来实现：
  - 数据共享
  - 资源共享和同步
- IPC 是一种机制，用于不同内核时间的通信和数据交换
- 这种机制可以保证：
  - 安全的并行运算Safe parallel computation
  - 同步
  - 应用程序的模块化

IPC 机制

- 消息的传输
  - **Send** 消息到其他的处理器
  - 从其他的处理器 **Receive** 消息
- 共享存储器
  - 一个处理器和其他处理器共用一块公共的存储区域
- 处理器同步的实现方式
  - 使用信号量
  - 使用中断

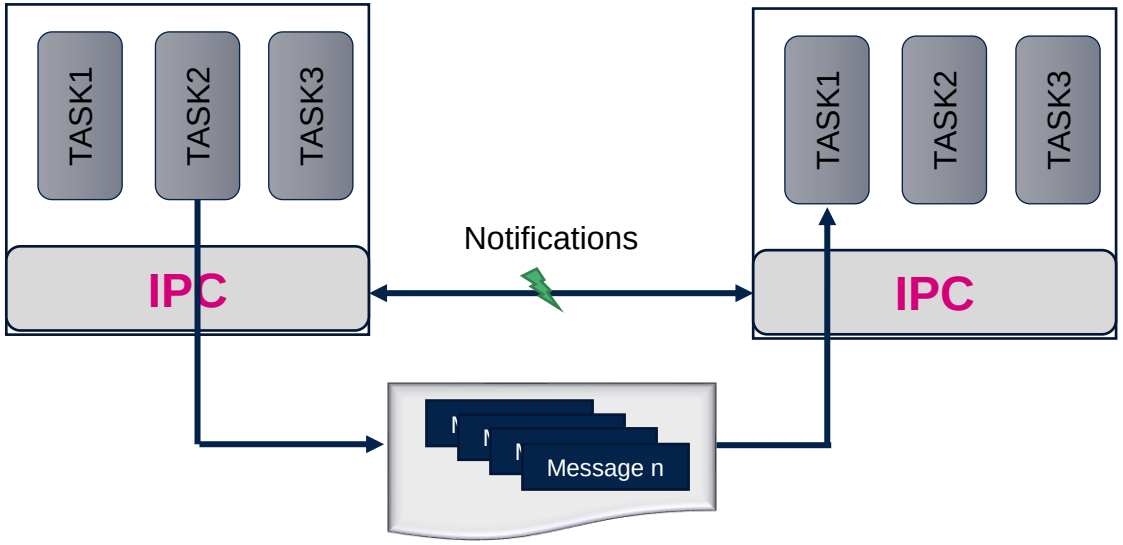
STM32H7 IPC 解决办法

- STM32H7两种可以使用的解决方案
  - 使用设备资源建立通信/ 同步协议协议
  - 使用 FreeRTOS IPC 模块 (在STM32Cube H7 Firmware)
- IPC 模块可以用在 :
  - FreeRTOS OS
  - 或者 OS-less 的应用
- IPC 模块用 :
  - 用Stream buffers 在不同 CPUs之间数据传输
  - 使用内核共享的存储器区域来分配 Stream buffers

License : same as FreeRTOS license.

FreeRTOS IPC 模块

- The IPC 模块是一整套 APIs， 用于不同应用任务之间交换数据
  - 每个Task有一个特定的 Stream buffer (一个发送，一个接收)



Stream Buffer
(共享存储器)

FreeRTOS IPC 模块

- 目前的IPC 模块定义了一个很友好的 APIs

功能	描述
pxStreamBufferCreateBlocking ()	在共享存储区shared Memory创立一个Stream Buffer
uxStreamBufferSend()	发送消息到另外一个Core内核
uxStreamBufferReceive()	从另外一个Core内核接收到消息

FreeRTOS IPC 模块

• IPC 模块用户自定义

- 部分的代码可以由用户自定义:
  - 在配置阶段，用户需要定义通知功能（在运行时，Stream Buffer API将生成通知）
- 通知模式:
  - 即当发送方消息发送完成时，发送方需要产生一个中断去通知接收发表明消息已经发送了
  - 即当接收方消息接收好消息后，接收方需要产生一个中断去通知发送方表明消息已经接收了
- 在用户端的两个macros宏定义

函数	描述
sbSEND_COMPLETED()	产生一个中断去通知接收方表明消息已发送了
sbRECEIVE_COMPLETED()	产生一个中断去通知发送方表明消息被接收了

中断的产生可以由EXTI线/硬件信号量或 SEV 指令产生

life.augmented

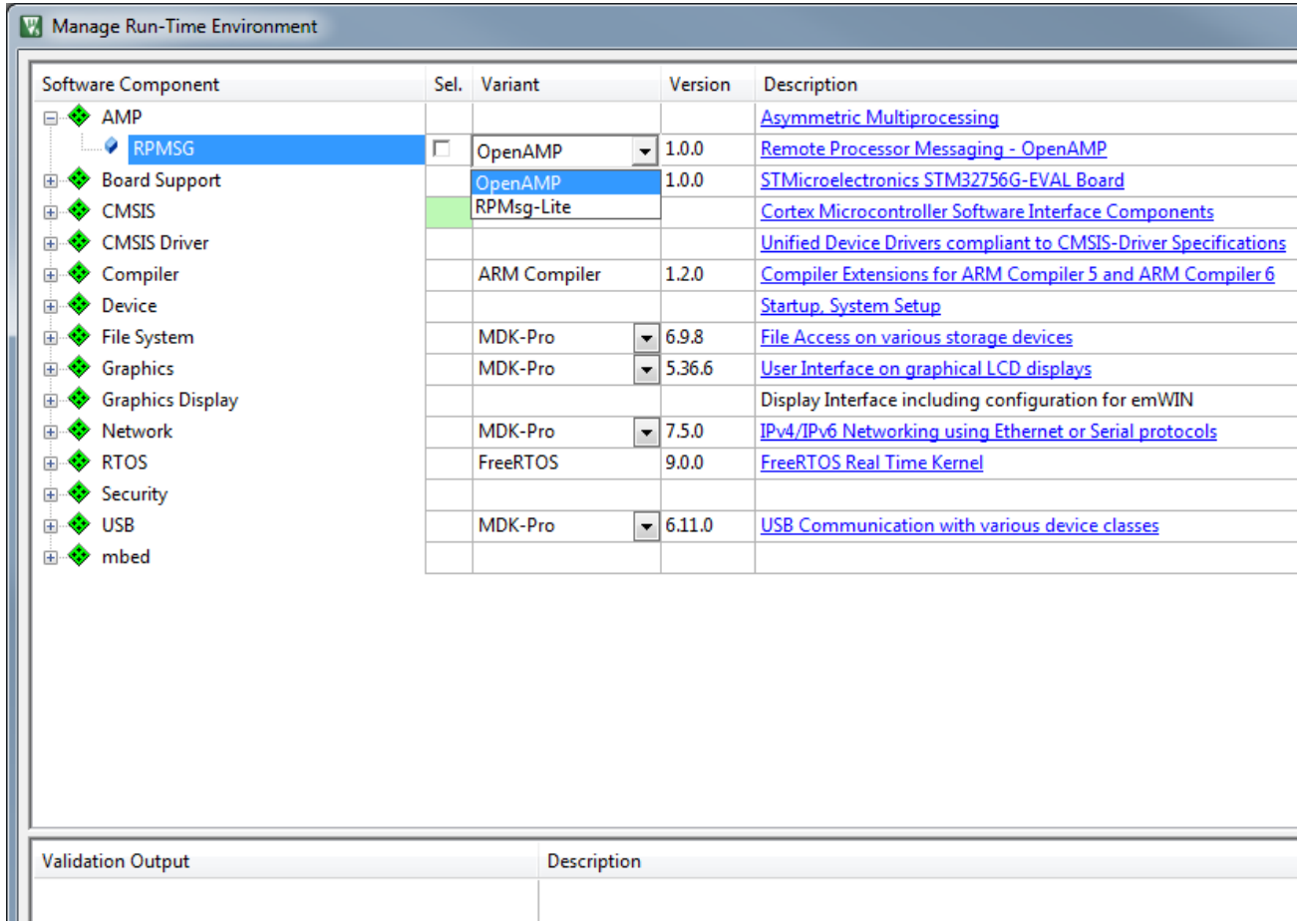
FreeRTOS IPC 优缺点

- FreeRTOS IPC 优势:
  - Small footprint代码小.
  - 简单易用(用户API友好)
  - 易集成,
  - 可以灵活的运用实施

- FreeRTOS IPC 缺点:
  - 依赖于FreeRTOS的资源

IPC 解决方案

- FreeRTOS IPC :
  - 需要移植到STM32H7
  - 需要更多的资源 (RAM/Flash)
- 示例 **OpenAMP**, 是一种开源的open-source framework IPC解决方案



life.augmented

life.augmented

STM32H747x IPC相关示例解析

示例的路径

- xxx\Projects\STM32H747I_EVAL\Hands_on\IPC\ ------双核通信
  - FreeRTOS IPC
  - 使用 EWARM Toolchain工具
- \STM32Cube_FW_H7_V1.5.0\Projects\STM32H747I-DISCO\Applications\
  - FreeRTOS --------FreeRTOS版本的CoreSync，类似HSEM中的例子
    - FreeRTOS_HwSemaphoreCoreSync
  - OpenAMP --------双核通信
    - OpenAMP_PingPong
    - OpenAMP_RTOS_PingPong
  - ResourceManager ---------资源共享，类似于HSEM里面的例子
    - ResourcesManager_SharedResources
    - ResourcesManager_UsageWithNotification

STM32H7- DMAMux

DMA Request

AN5224 (AN5224 STM32 DMAMUX: the DMA request router)

F3未使用DMAMUX

Table 25. Summary of DMA1 requests for each channel

Peripheral	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel6	Channel7
ADC	ADC1	-	-	-	-	-	-
SPI	-	SPI3_RX	SPI3_TX	SPI2_RX	SPI2_TX	-	-
USART	-	USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I2C	I2C3_TX	I2C3_RX	-	I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1	-	TIM1_CH1	TIM1_CH2	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	-
TIM2	TIM2_CH3	TIM2_UP	-	-	TIM2_CH1	-	TIM2_CH2 TIM2_CH4
TIM6/DAC	-	-	TIM6_UP DAC_CH1 (1)	-	-	-	-
TIM15	-	-	-	-	TIM15_CH1 TIM15_UP TIM15_TRIG TIM15_COM	-	-
TIM16	-	-	TIM16_CH1 TIM16_UP	-	-	TIM16_CH1 TIM16_UP(1)	-
TIM17	TIM17_CH1 TIM17_UP	-	-	-	-	-	TIM17_CH1 TIM17_UP(1)

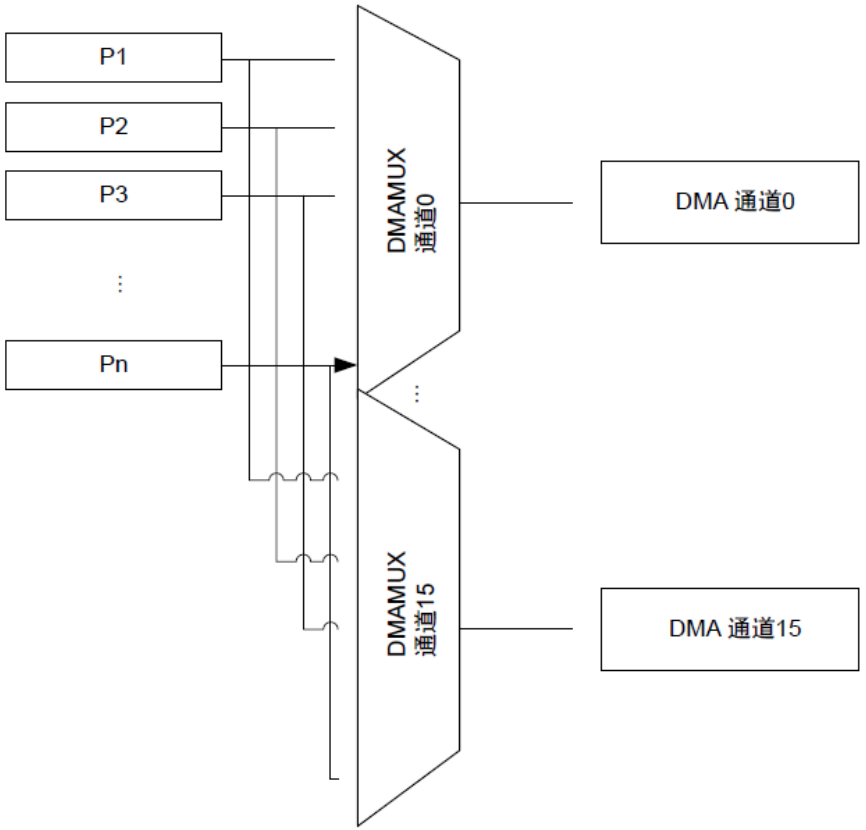
F4未使用DMAMUX

Table 43. DMA2 request mapping

Peripheral requests	Stream 0	Stream 1	Stream 2	Stream 3	Stream 4	Stream 5	Stream 6	Stream 7
Channel 0	ADC1	SAI1_A(1)	TIM8_CH1 TIM8_CH2 TIM8_CH3	SAI1_A(1)	ADC1	SAI1_B(1)	TIM1_CH1 TIM1_CH2 TIM1_CH3	-
Channel 1	-	DCMI	ADC2	ADC2	SAI1_B(1)	SPI6_TX(1)	SPI6_RX(1)	DCMI
Channel 2	ADC3	ADC3	-	SPI5_RX(1)	SPI5_TX(1)	CRYP_OUT	CRYP_IN	HASH_IN
Channel 3	SPI1_RX	-	SPI1_RX	SPI1_TX	-	SPI1_TX	-	-
Channel 4	SPI4_RX(1)	SPI4_TX(1)	USART1_RX	SDIO	-	USART1_RX	SDIO	USART1_TX
Channel 5	-	USART6_RX	USART6_RX	SPI4_RX(1)	SPI4_TX(1)	-	USART6_TX	USART6_TX
Channel 6	TIM1_TRIG	TIM1_CH1	TIM1_CH2	TIM1_CH1	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	-
Channel 7	-	TIM8_UP	TIM8_CH1	TIM8_CH2	TIM8_CH3	SPI5_RX(1)	SPI5_TX(1)	TIM8_CH4 TIM8_TRIG TIM8_COM

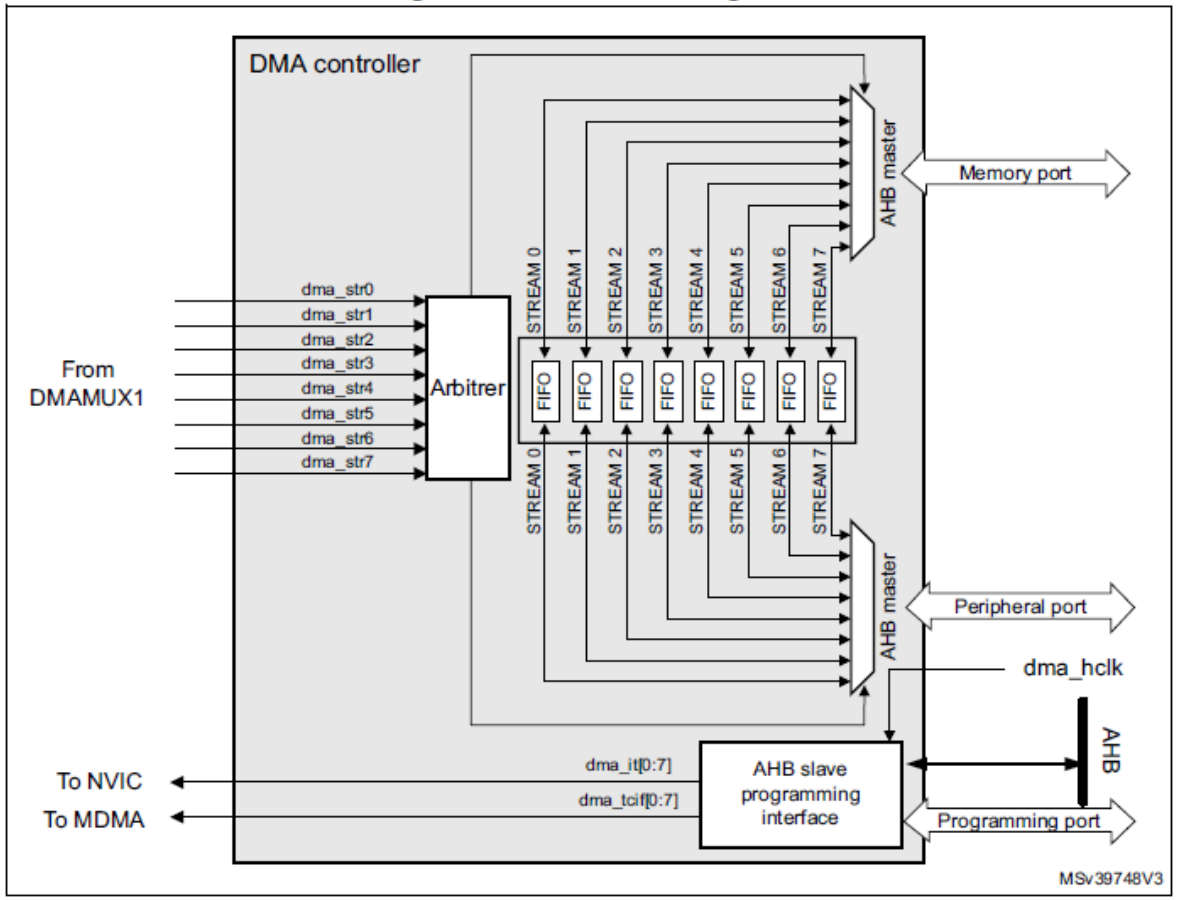
life.augmented

DMAMux模块框图

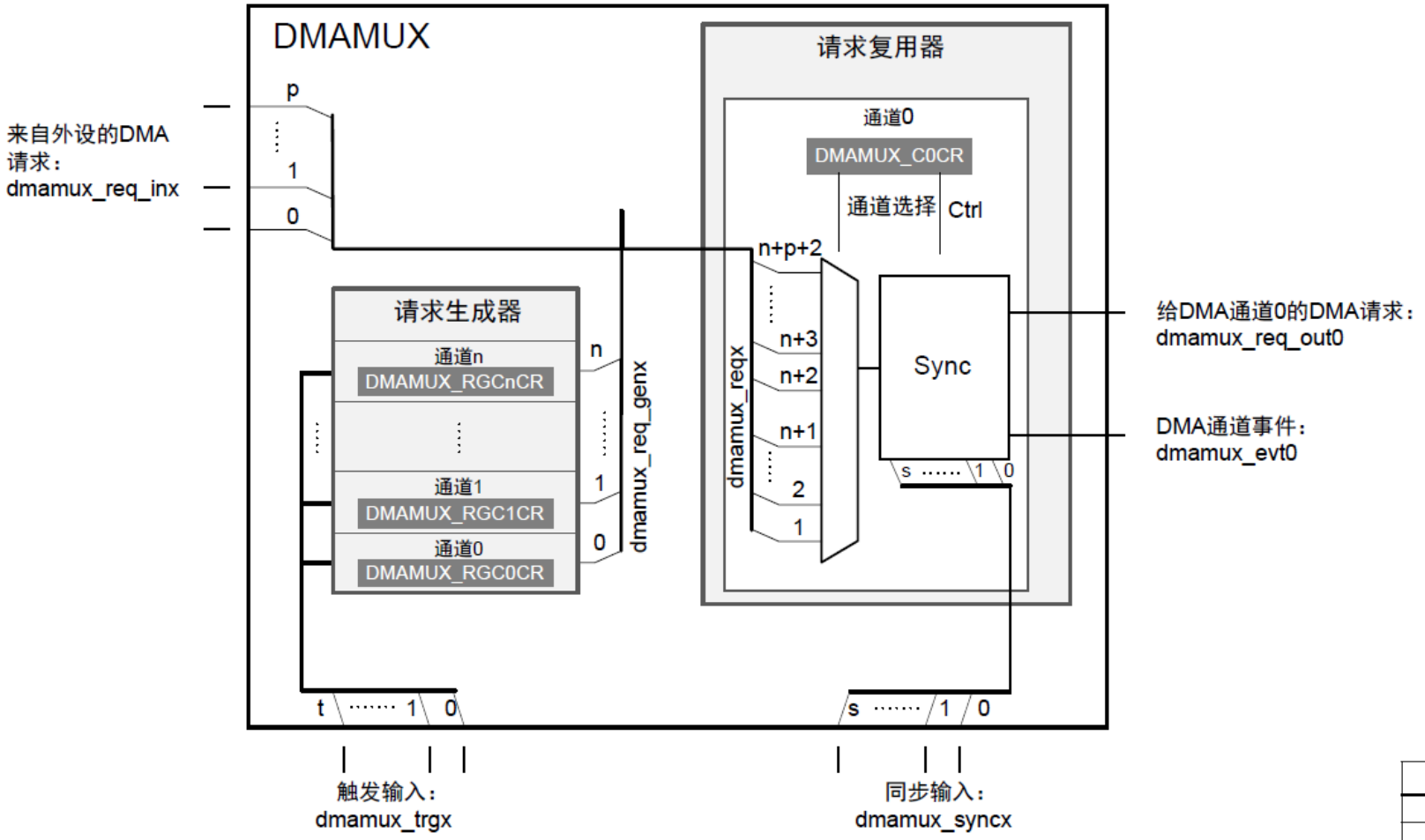


Px　Px 外设X请求（例如LPUART1_TX或LPUART1_RX）

Figure 78. DMA block diagram



DMAMux模块框图



注：简化框图仅有一个请求复用器

Table 120. DMAMUX1 and DMAMUX2 instantiation

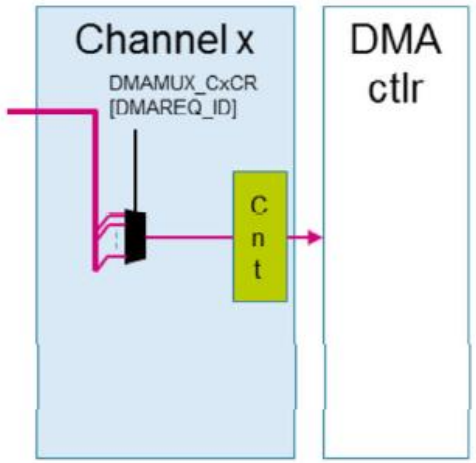
Feature	DMAMUX1	DMAMUX2
Number of DMAMUX output request channels	16	8
Number of DMAMUX request generator channels	8	8
Number of DMAMUX request trigger inputs	8	32
Number of DMAMUX synchronization inputs	8	16
Number of DMAMUX peripheral request inputs	107	12

life.augmented

DMAMux 操作模式

• 无条件请求转发操作模式

- 在无条件请求转发工作模式下，
  - 为了执行外设到内存或内存到外设的传输，DMA 控制器通道每次都需要一个外设 DMA 请求线。每次发生请求时， DMA 通道将数据传入/传出外设。
  - DMAMUX 请求复用器通道 X 允许选择外设 DMA 请求线到 DMA 通道 X。通过通道控制寄存器 （DMAMUX_CxCR）的 DMAREQ_ID 位中的编码 ID 选择外设 DMA 请求到复用器通道输出的连接。
  - 当复用器设置之后（DMAREQ_ID 不等于零），它确保 DMA 请求线的实际路由。
  - 对于产品中的每个外设 DMA 请求线，都会影响唯一的 ID；DMAREQ_ID = 0x00，则对应于没有选择 DMA 请 求线。
  - 在配置 DMAMUX 通道之后，可以依次配置相应的 DMA 控制器通道。
  - 不能将两个不同的 DMAMUX 通道配置为选 择相同的外设 DMA 请求线作为源。



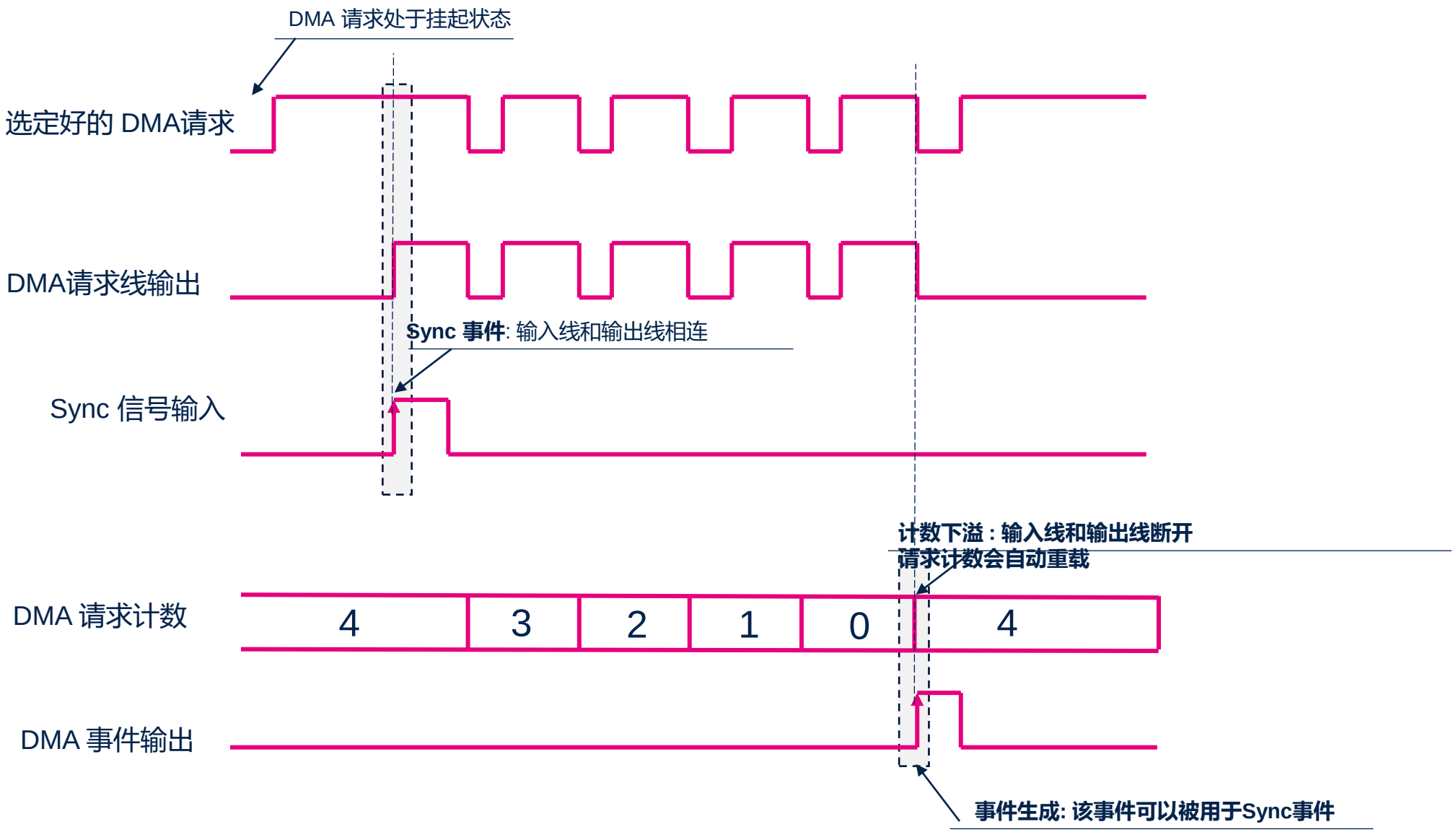
DMAMux 操作模式

• 有条件请求转发操作模式

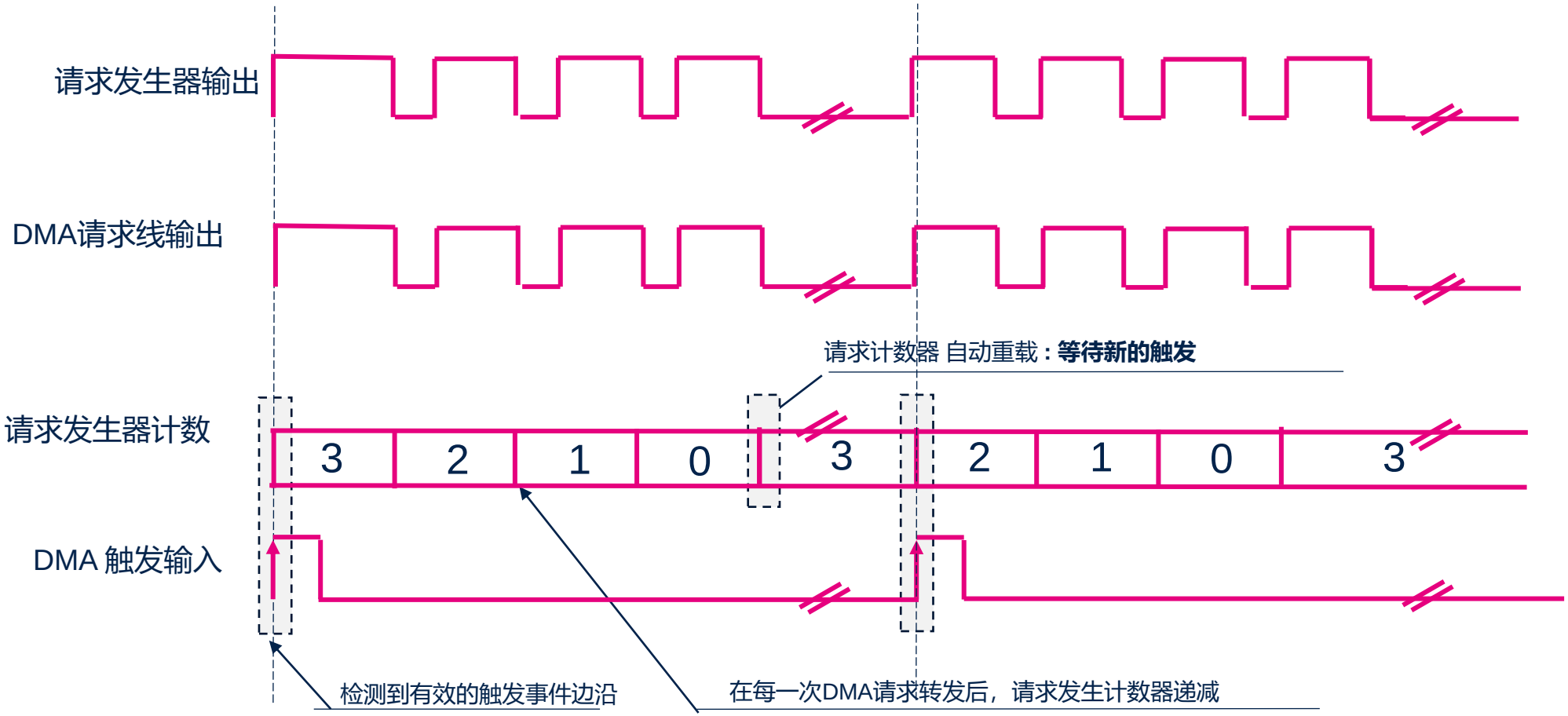
- 在有条件请求转发工作模式下，
  - 除了无条件请求转发，同步单元还允许软件实施有条件请求转发。只有在检测到定义的条件时，路由才能有效完成。DMA 传输可以与内部或外部信号同步。每个通道均可以独立的配置为同步操作模式。
- 例如，用户软件可以使用同步单元启动或调整数据传输吞吐量。可以通过以下方式之一转发 DMA 请求（同步信号输入源）：
  - 每次在GPIO 引脚（EXTI）上检测到沿时
  - 响应来自计时器的周期性事件
  - 响应来自外设的异步事件
  - 响应来自另一个请求路由器（请求链）的事件
- 根据DMA 请求条件，同步单元允许生成其他DMAMUX 子块（如请求生成器或另一个DMAMUX 请求复用器通道）可能使用的事件；

life.augmented

DMAMux ----事件生成


DMA 请求处于挂起状态
选定好的 DMA请求
DMA请求线输出
Sync 事件: 输入线和输出线相连
Sync 信号输入
计数下溢 : 输入线和输出线断开
请求计数会自动重载
DMA 请求计数
4
3
2
1
0
4
DMA 事件输出
事件生成: 该事件可以被用于Sync事件

DMAMux ----请求生成

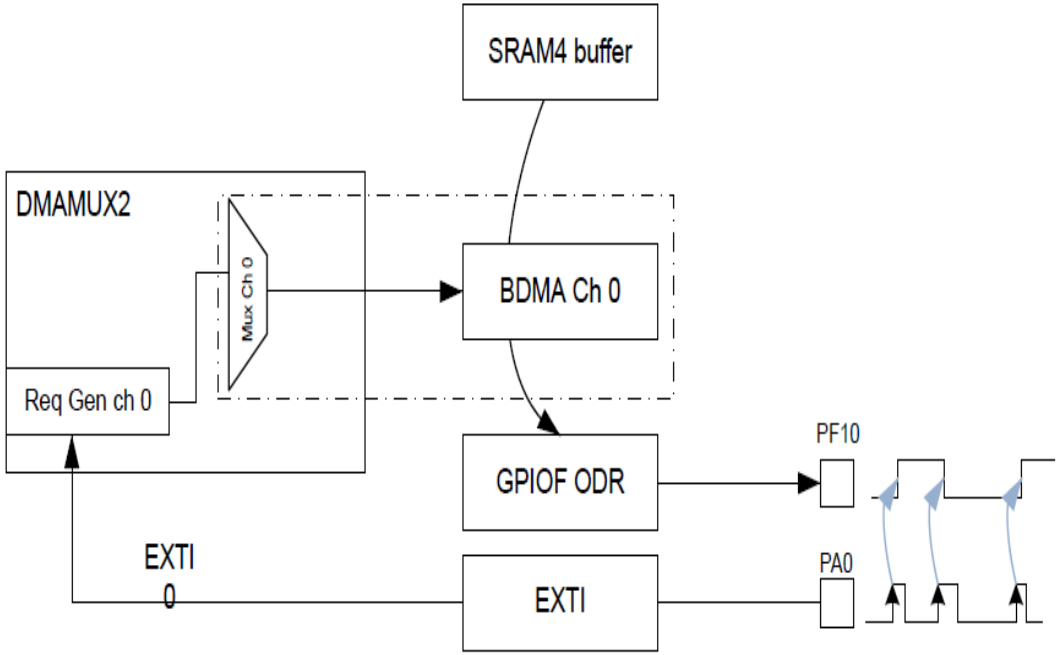

请求发生器输出
DMA请求线输出
请求计数器 自动重载：**等待新的触发**
请求发生器计数
3
2
1
0
3
2
1
0
3
DMA 触发输入
检测到有效的触发事件边沿
在每一次DMA请求转发后，请求发生计数器递减

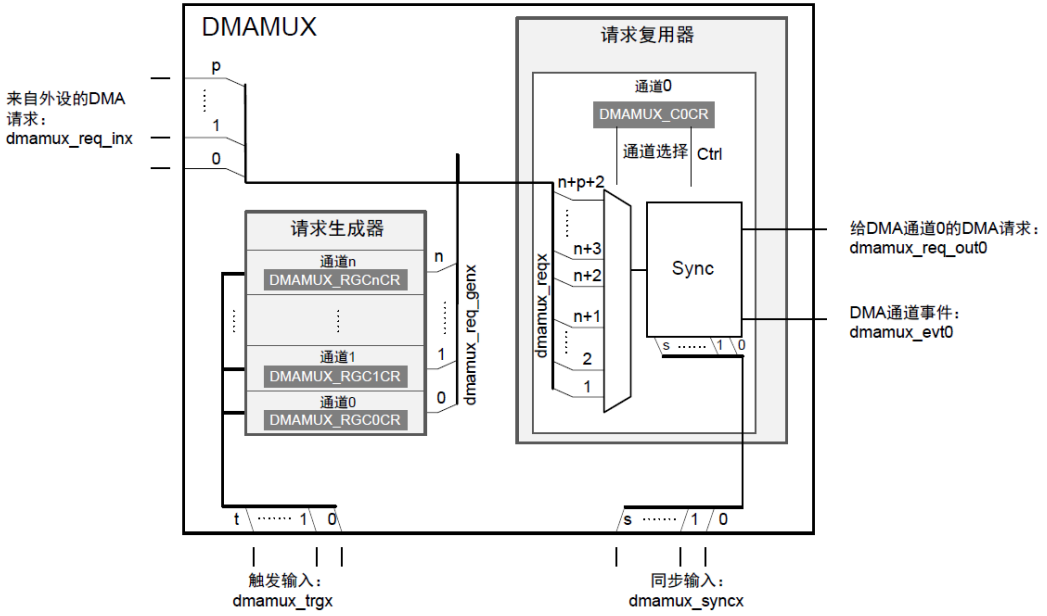
示例例程代码解析

- \STM32Cube\Repository\STM32Cube_FW_H7_V1.8.0\Projects\STM32H743I-EVAL\Examples\DMA
- DMAMUX_RequestGen
  - 本例使用EXTI0线路触发DMAMUX请求生成器，并执行从SRAM缓冲区数据到GPIO输出数据寄存器的DMA数据传输，在每次EXTI0上升沿发生时改变输出引脚状态。
- DMAMUX_SYNC
  - 这个例子在DMA同步模式下使用USART1发送从10到00的倒计时，周期为2秒。DMAMUX同步块被配置为使DMA传输与LPTIM1输出信号同步。同步信号的每个上升沿（LPTIM1输出信号）授权使用DMA将4个USART1请求发送到USART1外围设备。这四个请求代表两个字符'\n\r'加上这两个字符从10到00倒数。LPTIM1是配置为产生2秒周期的PWM。

DMAMUX_RequestGen

Figure 7. DMAMUX_RequestGen





注：简化框图仅有一个请求复用器

```
//请求产生器的输入（EXTI0）和对应的输出HAL_DMAMUX2_REQ_GEN_EXTI0（Syscfg_exti0_mux）
dmamux_ReqGenParams.SignalID  = HAL_DMAMUX2_REQ_GEN_EXTI0;
//请求产生器的输出Syscfg_exti0_mux，直接连到请求复用器的输入，从而产生DMA触发源
BDMA_REQUEST_GENERATOR；
hdma_bdma_generator[0,7].Init.Request = BDMA_REQUEST_GENERATOR[0,7];
// SRAM和外设之间的DMA传输通道
 hdma_bdma_generator[0,7].Instance = BDMA_Channel[0:7];
// 使能请求生成器，用于产生DMA传输请求信号；
HAL_DMAEx_EnableMuxRequestGenerator (&hdma_bdma_generator[0,7]);
```

life.augmented

DMAMUX_SYNC

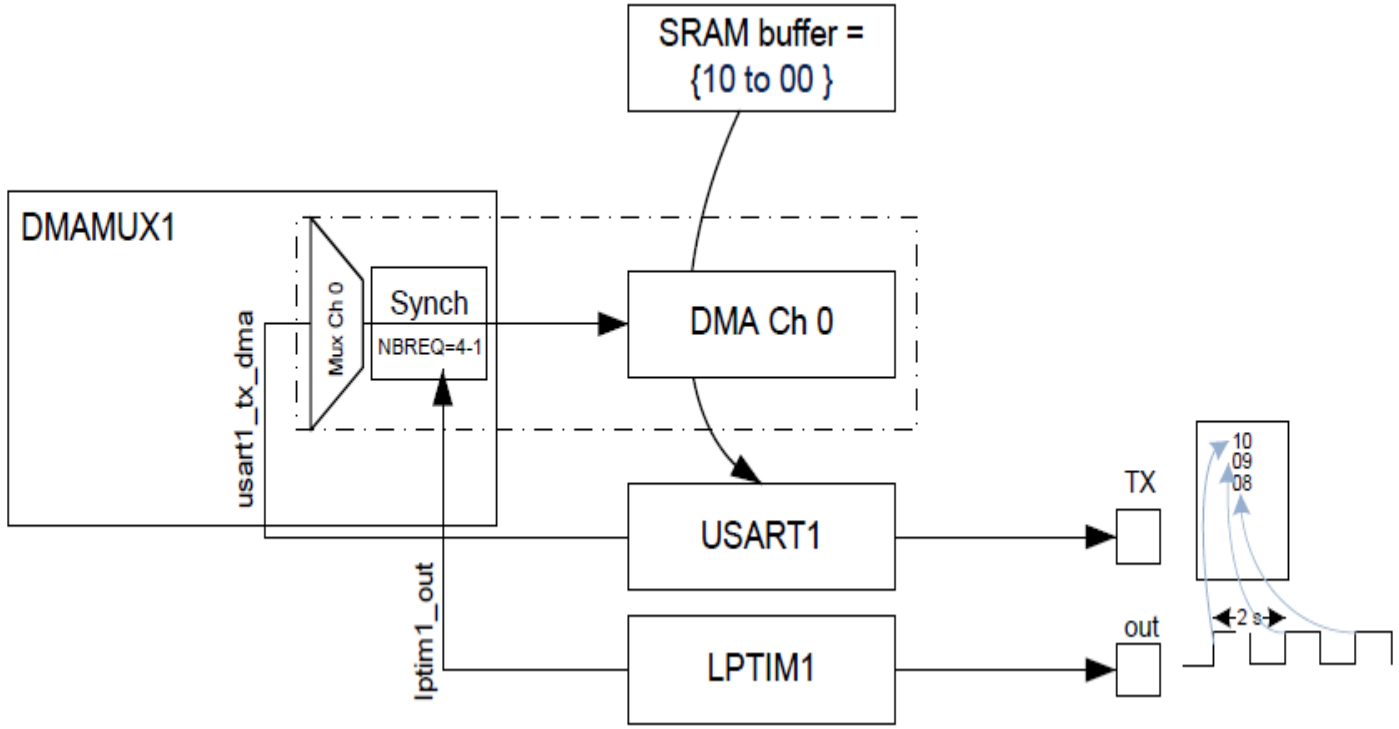


Table 118. DMAMUX1: assignment of synchronization inputs to resources

Sync. input	Resource	Sync. input	Resource
0	dmamux1_evt0	4	lptim2_out
1	dmamux1_evt1	5	lptim3_out
2	dmamux1_evt2	6	extit0
3	lptim1_out	7	TIM12_TRGO

This example shows how to use the DMA with the DMAMUX to synchronize a transfer with LPTIM1 output signal.
The USART1 is used in DMA synchronized mode to send a countdown from 10 to 00 with a period of 2sec

Start countdown :

10
09
08
07
06
05
04
03
02
01
00

Example Finished

```
//4 requests are autorized after each edge of the sync signal
dmamux_syncParams.RequestNumber = 4;
//使能DMAMUX同步模式
HAL_DMAEx_ConfigMuxSync(&DMA_Handle, &dmamux_syncParams);
```

life.augmented

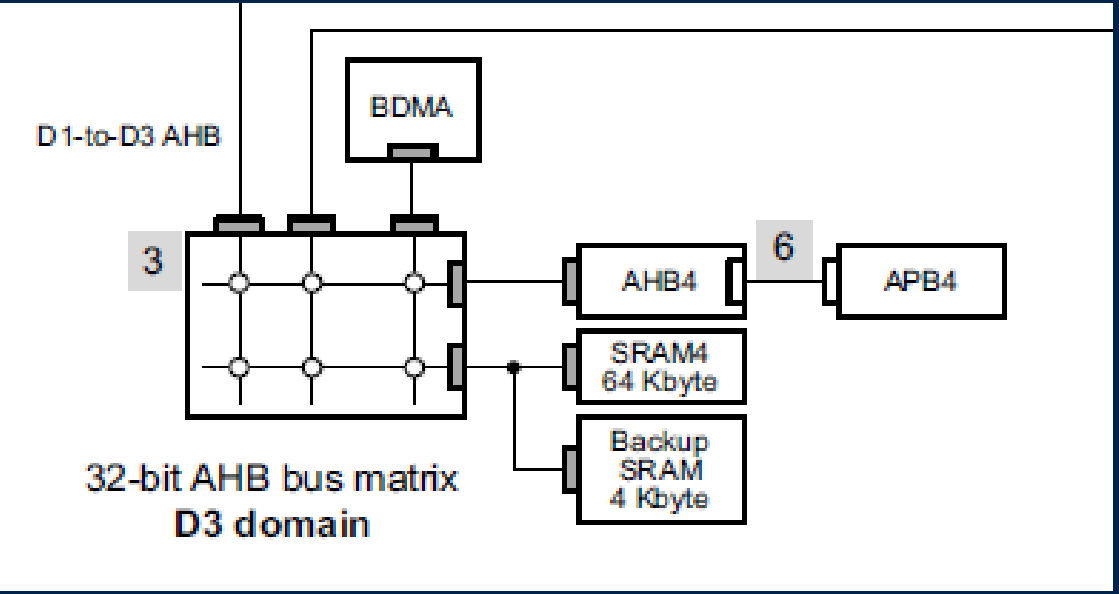
STM32H7- DMAs

标准DMA

AN4031《Using the STM32F2, STM32F4 and STM32F7 Series DMA controller 》

STM32H7- BDMA

Basic DMA

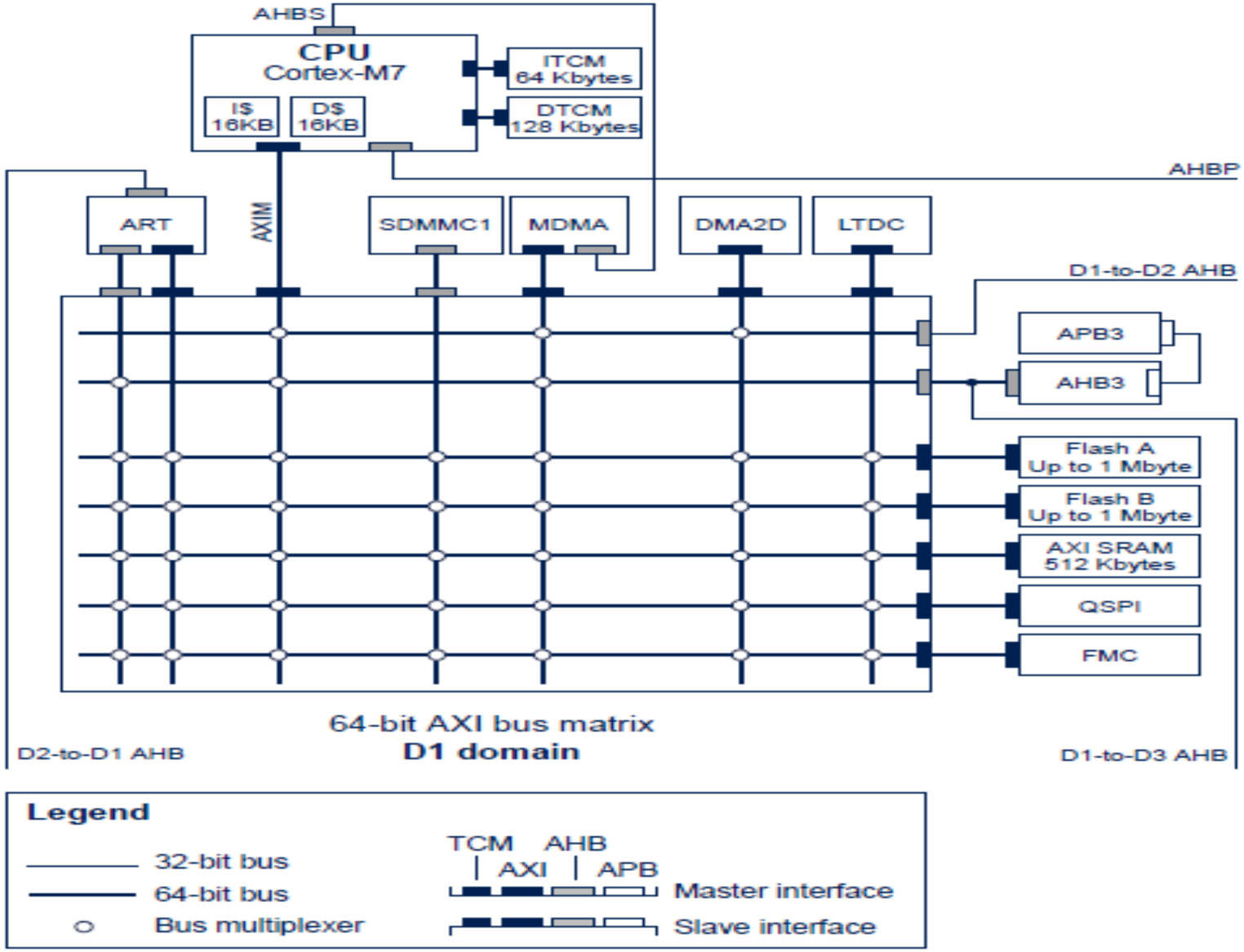


STM32H7- **MDMA**

3HDMA

AN5001 (STM32Cube Expansion Package for STM32H7 Series MDMA)

Figure 1. **Master connections of the AHB/AXI matrix for STM32H74xxx and STM32H75xxx**



life.augmented

主要特性

- 每一个 MDMA通道可以执行:
  - **单块传输:** 传输一个块（最多64K字节）.在块传输结束时，DMA通道将会被关闭disabled.
  - **重复块传输:** 在禁用通道前，传输多个块(最多4096个块)
  - **链表传输:**当前数据传输完成后，从内存加载新的块控制结构，并开始新的块的传输

- 灵活的独立通道:
  - 对于源和目的地址，可以独立的 递增, 递减或 不递增
  - 对于源和目地地址，可以独立的设置传输数据长度 ，以及源和目的的增量大小
  - 支持大小端数据格式: 字节, 半字和字

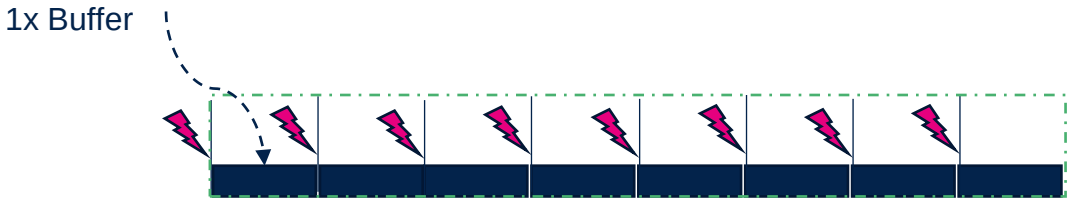
life.augmented

传输类型

- MDMA 支持 单一 或 增量突发 传输,
- 软件可配置突发传输的大小，最多可达到128字节,
- 最大突发数据传输的大小是128字节. 对于更大的数据量的传输，突发模式的传输受到长度128级的FIFO的限制，该FIFO将用于存储要传输的临时数据
  - e.g. 16x64-bit or 32x32-bit = 1024bit = 128B
- 对于TCM内存访问，只有当增量和数据大小相同且小于等于32位时才允许突发访问

MDMA 通道触发模式

- 单个请求，数据的大小 组成 传输，为下列中的一种：
  - 请求的模式
    - TRGM=’00’



每次请求会传输一个buffer

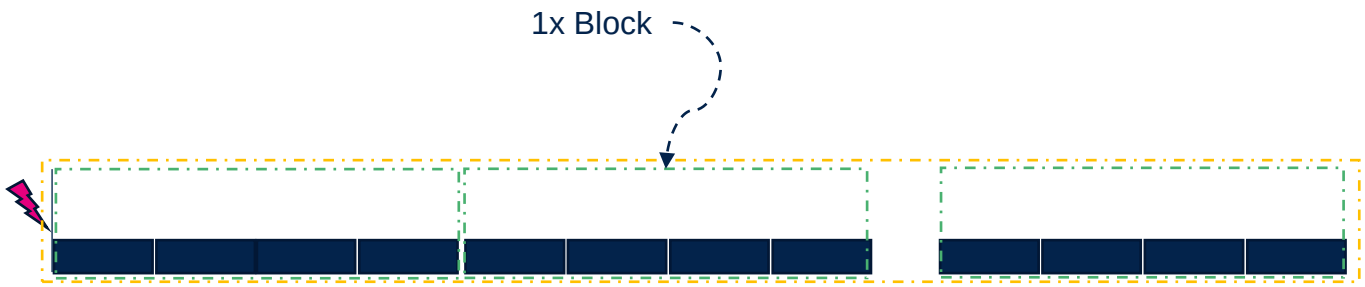
MDMA 通道触发模式

- 单个请求，数据的大小 组成 传输，为下列中的一种:
  1. 请求的模式
     - TRGM=’01’
  2. Block块的大小
     - BNDT[16:0]的值决定

每次请求传输一个 block块

MDMA通道触发模式

- 单个请求，数据的大小 组成 传输，为下列中的一种:
  1. 请求的模式
     - TRGM=’10’
  2. 每个block块传输的大小
     - BNDT[16:0]的值决定
  3. 重复repeat blockd的次数
     - BRC[11:0]的值决定



每次请求传输多个block块

MDMA 通道触发模式

- 单个请求，数据的大小 组成 传输，为下列中的一种:
  1. 请求的模式
     - TRGM=’11’

一个请求开始传输，直到该链表的指针指向INULL才停止

请求仲裁(1/2)

- buffer传输是通过MDMA进行的最小的数据长度的传输，在未传输完buffer数据时，即使MDMA通道之间有新的请求，传输过程也不会有新的仲裁

- 在块传输的情况下，在传输单个buffer之后，MDMA将在新的外部请求和内部存储的请求之间,进入一个新的仲裁阶段。
  - 如果没有其他更高的优先级,通道请求被激活时,一个新的 buffer 传输将启动同样的通道
  - 在超过一个buffer传输周期的时间外，不会阻止更高级别的MDMA请求，因为在每次buffer传输之后都会进行新的通道仲裁

life.augmented

请求仲裁(2/2)

- 两种数据的数组大小对 MDMA 和 应用会有影响：
- Buffer transfer size:
  - 是数据传输长度，是不可中断的最大的数据传输长度。
  - 它是在检查到其他通道上的MDMA请求之前，要在一个通道上完成的传输数据的长度
- Burst size:
  - 是突发传输模式的数组大小
  - 它定义了在总线仲裁级别下，不可中断的最大的数据传输长度，并不是MDMA仲裁器级别；
  - 它可以阻塞其它的 masters 获得总线的访问；

- 在实时性要求很高的应用中，选择正确的突发和缓冲区buffer传输大小，对于其他MDMA通道和主控的是非常重要

MDMA buffer 传输

- Buffer transfer 是在一个通道上，一个MDMA请求事件，传输的最小逻辑上传输的数据量。
  - 在当前通道上，需要传输的数据总量,是在 MDMA请求后由触发模式决定。

- 要传输的数据项的数目，宽度（8位，16位，32位或64位）和用于数据传输的burst 长度都是软件可编程。

- 在为DMA、外设请求事件提供服务之后，通过将掩码数据值写入 给定掩码地址的地址来确认请求。
  - 主要用于清除请求事件源标志.

MDMA 块传输 (1/2)

- 块是一个连续的数据数组，单个数据块最大传输为64 KB字节，通过连续的buffer传输进行传输。
- 每个数据块由“起始地址”和“块长度”定义。
- 根据配置的触发模式，一个外设的DMA请求可以触发整个块传输。
  - 如果没有其他更高优先级的通道请求处于活动状态，则将为同一通道启动新的buffer传输

life.augmented

MDMA 块传输 (2/2)

- 当一个块传输完成后,可以执行三个操作:
  - 单个块是块重复传输的一部分：可以重新加载块长度，并根据块重复地址的 更新寄存器 中的信息，去计算新的块的起始地址 。
  - 它是单个块或块重复传输中的最后一个块，将会从内存加载下一个块的信息
  - 它是当前MDMA信道需要传输的最后一个块：该通道被禁用，并且不再接受该信道的MDMA请求;

MDMA 块重复传输模式

- 块重复模式允许为源和目标地址使用不同的“起始地址”进行块重复传输。

- 当块重复模式处于活动状态（重复计数器非0）的情况下，在当前块的传输完成时，块的参数将更新：
  - 将会重新加载要传输的数据字节数（BNDT）值
  - 将会根据块重复的源和目标地址的更新模式的配置，去更新块的源地址和目标地址的地址值，其中更新模式在BRSUM和BRDUM中设定
  - 将会让重复计数器递减1

- 当重复块计数器达到0，最后一块将被视为一个单独的块传输.

MDMA Linked list 模式

- 链表模式允许从通道 Link Address register 即CLAR寄存器中所给的地址， 加载新的MDMA的配置，
- 在该操作之后，通道准备好接受新的请求，比如接受block块、repeat block重复块模式中所定义的请求，或者如果触发模式设置为完全数据传输（即TRGM=11），则继续传输
- 当加载触发和总线选择寄存器(TBR)值时，自动触发源可能会改变。
- 如果触发模式设置为TRGM=11 的情况下所选触发模式和软件请求的选择不得更改

life.augmented

MDMA Linked list 模式

- 通道配置(channel link address LAR)必须在AXI地址空间
- LAR value 必须用 Double Word 地址对齐, i.e. LAR[2:0] = 0x0

寄存器 (32-bit word)	从Link Address register的偏移量	描述
CTCR	0x00	传输配置寄存器
CBNDTR	0x04	Block块数据寄存器
CSAR	0x08	源地址寄存器
CDAR	0x0C	目的地址寄存器
CBRUR	0x10	Block Repeat 地址更新寄存器
CLAR	0x14	Link 地址寄存器: 下一个描述符
CTBR	0x18	触发和总线选择寄存器
CMAR	0x1C	Mask 地址寄存器
CMDR	0x20	Mask 数据寄存器

life.augmented

每一个通道的中断事件

中断事件	描述
CTCIF	MDMA 通道传输完成. 它会将通道的enable使能位写为0（通道禁能）
BTIF	MDMA 块传输完成
BTRIF	MDMA 块重复传输完成
TCIF	MDMA buffer 传输完成.
TEIF	MDMA传输出错

MDMA 在低功耗模式下

模式	描述
Run	激活。
Sleep	激活. MDMA 中断能够唤醒STM32H7。
Stop	冻结。MDMA 寄存器的内容是保持的。
Standby	MDMA Powered-down. 从Standby模式下返回，MDMA 必须重新初始化。

life.augmented

MDMA 用法用例

- MDMA 可以用于从其他 (DMA1 / DMA2和BDMA)处收集数据，并将数据提供给D1域的Cortex-M7的DTCM或AXI-SRAM处，供M7内核可以以更快的速度处理 。

- DMA链表模式，执行一组DMA传输，且不需要CPU干预
  - 该模式可用于为其他DMA准备数据，然后将DMA配置设置为开始传输。
  - 该模式用来支持分散/聚合。这意味着源和目标区域不需要占用连续的内存区域。源和目标数据区域是由一系列的链表描述符定义，这些描述符控制数据块的传输。

life.augmented

小结

- 1、了解MDMA的架构以及他的Linked list mode模式，MDMA和DMA1,DMA2可以协同工作

- 2、关于MDMA的示例，我们以前线下培训的时候也做过相关的动手实验，大家可以www.stmcu.com.cn上查找，进一步了解，如何用CubeMX生成相应MDMA代码

- 3、AN5001 《STM32Cube Expansion Package for STM32H7 Series MDMA》

Thank you

© STMicroelectronics - All rights reserved.
ST logo is a trademark or a registered trademark of STMicroelectronics International NV or its affiliates in the EU and/or other countries.
For additional information about ST trademarks, please refer to www.st.com/trademarks.
All other product or service names are the property of their respective owners.

life.augmented