STM32 MCU IAP 例程跳转到 APP 代码简要分析

问题：

有客户在初次使用 STM32 MCU IAP 的例程的，可能会对跳转到 APP 部分的函数的实现产生疑问：

问题 1：JumpAddress 地址为什么指向 APPLICATION_ADDRESS + 4；

MSP 主堆栈指针为什么指向 APPLICATION_ADDRESS；

问题 2：为什么需要做以下判断：

```
if (((*(__IO uint32_t*)APPLICATION_ADDRESS) & 0x2FFE0000 ) == 0x20000000)
{

}
```

此 Tips 对于这两个个问题做简要分析.进行所以说主要代码如下：

```
/* Test if user code is programmed starting from address "APPLICATION_ADDRESS" */
  if (((*(__IO uint32_t*)APPLICATION_ADDRESS) & 0x2FFE0000 ) == 0x20000000)
  {
    /* Jump to user application */
    JumpAddress = *(__IO uint32_t*) (APPLICATION_ADDRESS + 4);
    Jump_To_Application = (pFunction) JumpAddress;
    /* Initialize user application's Stack Pointer */
    __set_MSP(*(__IO uint32_t*) APPLICATION_ADDRESS);
    Jump_To_Application();
  }
```

调研：

问题 1 分析：

- 从 startup_stm32f4xx.s 中的启动代码可以看出:程序开始第一条指令地址为 CSTACK，第二条指令地址为复位指令 ,参考代码中的红色部分(蓝色为注释)；

```
      EXTERN  __iar_program_start
      EXTERN  SystemInit
      PUBLIC  __vector_table

      DATA
__vector_table
      DCD     sfe(CSTACK)                     ; APPLICATION_ADDRESS
      DCD     Reset_Handler          ; Reset Handler    ; APPLICATION_ADDRESS + 4

      DCD     NMI_Handler            ; NMI Handler
      DCD     HardFault_Handler      ; Hard Fault Handler
      DCD     MemManage_Handler      ; MPU Fault Handler
      DCD     BusFault_Handler       ; Bus Fault Handler
      DCD     UsageFault_Handler     ; Usage Fault Handler
      DCD     0                      ; Reserved
      DCD     0                      ; Reserved
      DCD     0                      ; Reserved
      DCD     0                      ; Reserved
      DCD     SVC_Handler            ; SVCall Handler
      DCD     DebugMon_Handler       ; Debug Monitor Handler
```

```
        DCD     0                       ; Reserved
        DCD     PendSV_Handler          ; PendSV Handler
        DCD     SysTick_Handler         ; SysTick Handler
```

当程序启动时首先要执行复位程序，因此 JumpAddress 地址指向复位指令地址(即 APPLICATION_ADDRESS + 4) ;

- MSP 对应的是主堆栈指针，指向 CSTACK 地址(即 APPLICATION_ADDRESS).

问题 2 分析 :

- ApplicationAddress 存放的是程序的主堆栈地址，CSTACK 堆栈地址指向 RAM，而 RAM 的起始地址是 0x20000000；

  因此上面的判断语句执行：判断用户代码的堆栈地址是否落在:0x20000000~0x2001ffff 区间中，这个区间的大小为 128K.

  例程中使用芯片 RAM=128K，因此做上面的判断；如果芯片 RAM 比 128K 大的话，可以在此判断语句做调整.

结论 :

- 以上分析可以看出，程序跳转部分的设计与 APP 启动代码和芯片 RAM 的大小有关系;移植和调试时需加以注意.

处理 :