测试 STM32C031 的 BOOT_LOCK 位

关键字：STM32C0， STM32G0， 选项字节，锁死， BOOT_LOCK，PCROP 全片保护

1. 前言

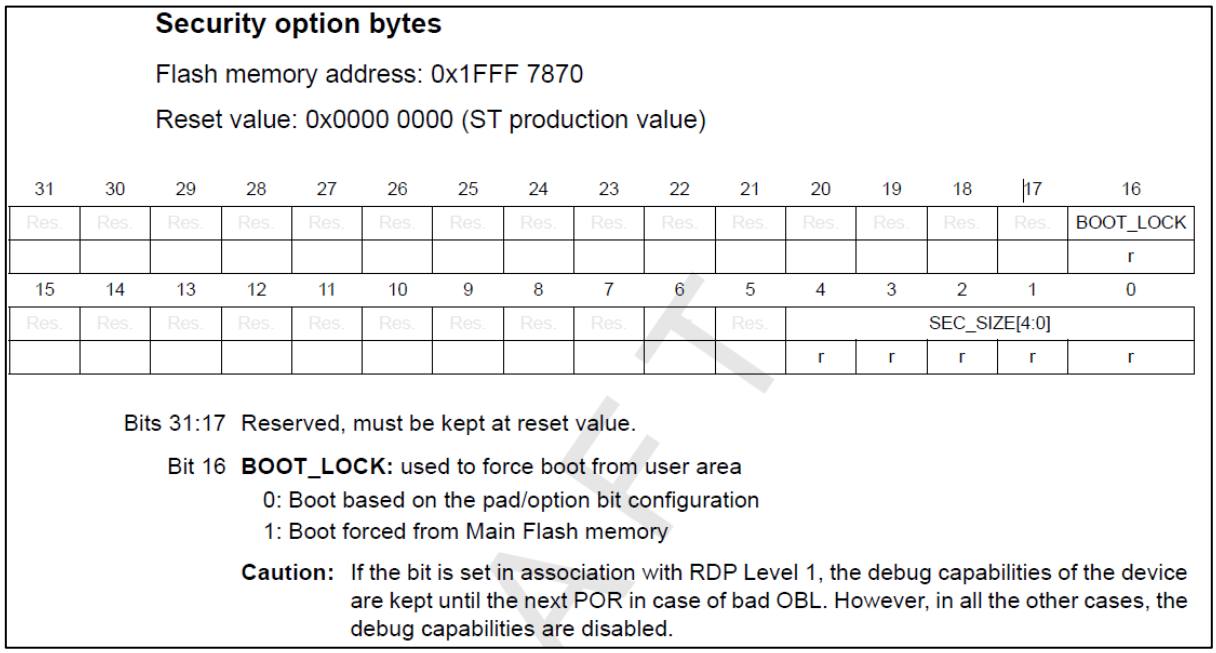
STM32C031 继承了 STM32G0 选项字节的安全特性，但是 STM32G0 在选项字节失配时会出现芯片永久锁死无法恢复的情况（由于 STM32G0 的一个 Weakness，导致可以通过添加一些处理来避免这种情况）。所以需要确认 STM3C031 中是否也会出现相同的情况，以及出现这种情况时是否也可以使用与 STM32G0 相同的处理来解决。

2. 描述

STM32G0 在选项字节失配时，会导致芯片锁死（PCROP 全片保护+BOOT_LOCK 置位+RDP Level 1），并且 DBG_SWEN 的复位值变为 0，也就是复位后默认关闭调试口，所以在选项字节失配时无法通过调试口修改选项字节。并且由于 PCROP 全片保护导致中断向量表无法被加载（被 PCROP 保护的区域，只能访问指令，不能访问数据），所以也不能通过执行代码来修改选项字节。由于选项字节不能通过调试口和执行代码这两种方法进行修改，这时芯片就会被会永久锁死无法恢复。但是由于 STM32G0 的一个问题 (Erratasheet 中的“Flash memory PCROP area weakness ”章节有描述)，会导致在 PCROP 全片保护的情况下可以执行 Reset_Handler。所以可以在 STM32G0 的 Reset_Handler 中添加一段将 DBG_SWEN 置位（打开调试口）的汇编代码，然后再通过调试口修改选项字节对芯片进行恢复。

而 STM32C031 的选项字节与 STM32G0 的选项字节特性类似，但是在 BOOT_LOCK 的描述中有一些区别。在 STM32C031 中如果 BOOT_LOCK 置位与 RDP1 相关联，则会保留调试能力；在其它情况下如果 BOOT_LOCK 置位+RDP1，则会默认关闭调试口。STM32C031 的 BOOT_LOCK 描述如图 1 所示。

图1. STM32C031 中 BOOT_LOCK 位描述



Security option bytes

Flash memory address: 0x1FFF 7870

Reset value: 0x0000 0000 (ST production value)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	SEC_SIZE[4:0]				
											r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **BOOT_LOCK:** used to force boot from user area

0: Boot based on the pad/option bit configuration

1: Boot forced from Main Flash memory

Caution: If the bit is set in association with RDP Level 1, the debug capabilities of the device are kept until the next POR in case of bad OBL. However, in all the other cases, the debug capabilities are disabled.

对图 1 的描述有个疑问：由于选项字节失配导致的 RDP1 与 BOOT_LOCK 置位，是否可以算的上相关联？

分析：如果在 OBL 阶段检测到选项字节失配，会将 USR OPT 的所有位都配置为 1。也就是 RDP 为 0xFF（RDP Level 1），PCROP_START 与 PCROP_END 都为 0x3F(全片保护，bit0~5)，BOOT_LOCK 置位。所以这时 BOOT_LOCK 置位与 RDP Level 1 都是由于选项字节失配导致的，是相关联的。

如果分析正确的话，选项字节失配导致的 BOOT_LOCK 置位与 RDP Level 1 就是相关联的，这样就可以通过调试口修改选项字节，避免芯片永久锁死无法恢复的问题。

为了验证文档描述与分析是否正确，设计了下面三个测试进行验证。

- 测试一：用来测试 BOOT_LOCK 置位与 RDP Level 1 无关联时，设备是否失去调试能力。
- 测试二：用来测试 PCROP 全片保护时能否执行 Reset_Handler。这个测试是为了测试三做准备。如果可以执行 Reset_Handler 的话（STM32G0 由于自身的 weakness 导致可以执行），则可以在 Reset_Handler 中添加打开调试口的代码，避免在测试三出现芯片永久锁死的情况。
- 测试三：用来测试在选项字节失配时能否连接调试口，如果能够连接，则说明分析是正确的。

3. 测试

下面几个测试分别是为了验证文档的描述与分析是否正确进行设计的。其中用到的设备与工具为：NUCLEO-C031C6 + STM32CubeMX + IAR + STM32CubeProgrammer。

3.1. 测试一：在 RDP1+BOOT_LOCK 置位时，测试 DBU_SWEN 的默认值

3.1.1. 测试目的

测试 BOOT_LOCK 置位与 RDP1 无关联时，调试口在上电时是否关闭。验证是否与文档描述一致。

3.1.2. 测试思路

使用 STM32CubeProgrammer 配置选项字节，将 BOOT_LOCK 置位，然后将 RDP 设置为 Level 1。根据文档描述，这种情况下会默认关闭调试口，所以在修改选项字节之前，需要将能够打开调试口的代码下载到 STM32C031 中，否则芯片无法进行恢复。

3.1.3. 测试代码

下面代码的功能为：通过 NUCLEO-C031C6 板上的 LD4 来显示 STM2C031 的调试口状态（LD4 亮：调试口打开； LD4 灭：调试口关闭），通过按钮 B1 来控制调试口的开启与关闭（拨动一次按钮，调试口的状态翻转一次）

```
/* USER CODE BEGIN 3 */
    if(!!READ_BIT(FLASH->ACR, FLASH_ACR_DBG_SWEN))
    {
      if(HAL_GPIO_ReadPin(LD4_GPIO_Port, LD4_Pin) == GPIO_PIN_RESET)
      {
        HAL_GPIO_WritePin(LD4_GPIO_Port, LD4_Pin, GPIO_PIN_SET);
      }
    }
    else
    {
      if(HAL_GPIO_ReadPin(LD4_GPIO_Port, LD4_Pin) == GPIO_PIN_SET)
      {
        HAL_GPIO_WritePin(LD4_GPIO_Port, LD4_Pin, GPIO_PIN_RESET);
      }
    }

    if(HAL_GPIO_ReadPin(B1_GPIO_Port, B1_Pin) == GPIO_PIN_SET)
    {
      while(HAL_GPIO_ReadPin(B1_GPIO_Port, B1_Pin) == GPIO_PIN_SET);
      HAL_GPIO_TogglePin(B1_GPIO_Port, B1_Pin);
      if(!!READ_BIT(FLASH->ACR, FLASH_ACR_DBG_SWEN))
      {
        MODIFY_REG(FLASH->ACR, FLASH_ACR_DBG_SWEN, 0);
      }
      else
      {
        MODIFY_REG(FLASH->ACR, FLASH_ACR_DBG_SWEN, FLASH_ACR_DBG_SWEN);
      }
```

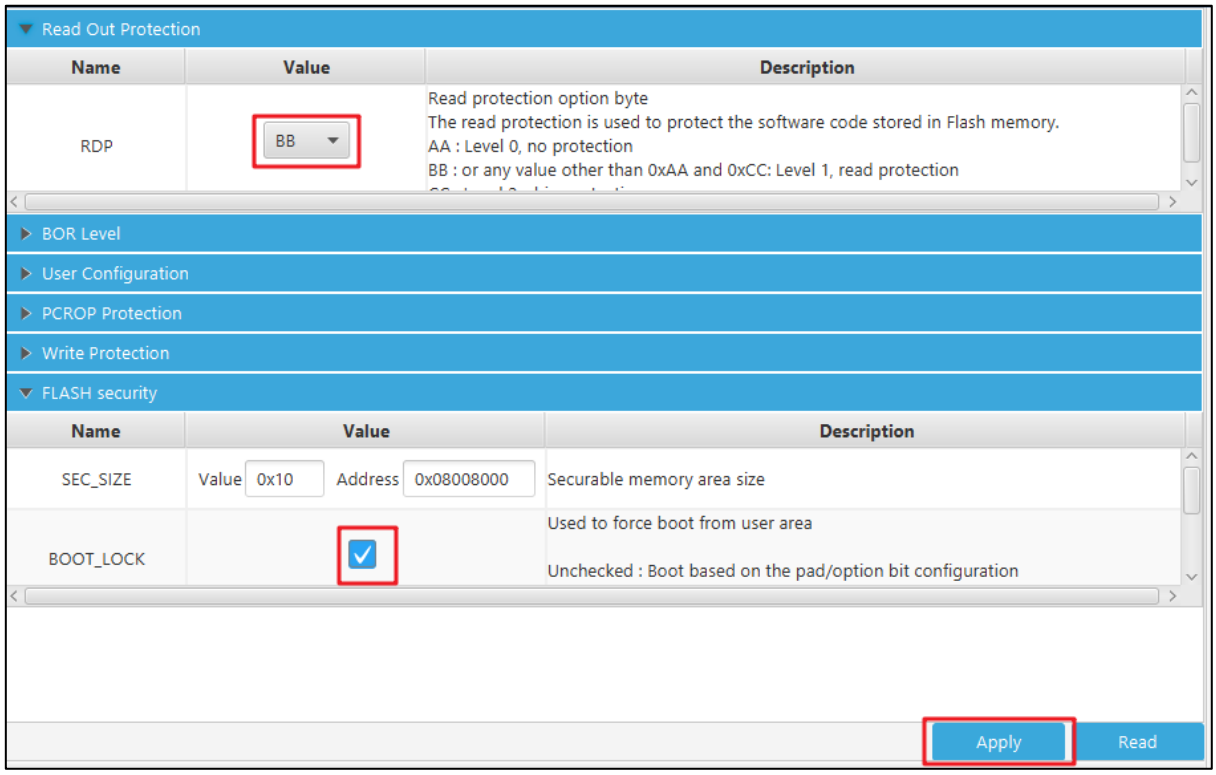
```
}
```

3.1.4. 测试过程

将 3.1.3 中的代码编译完成之后下载到 NUCLEO-C031C6 中会发现，每拨动一次按钮 B1，LD4 灯的状态就会翻转一次，并且在 LD4 灯不亮的时候不能通过 Stm32CubeProgrammer 连接或调试。

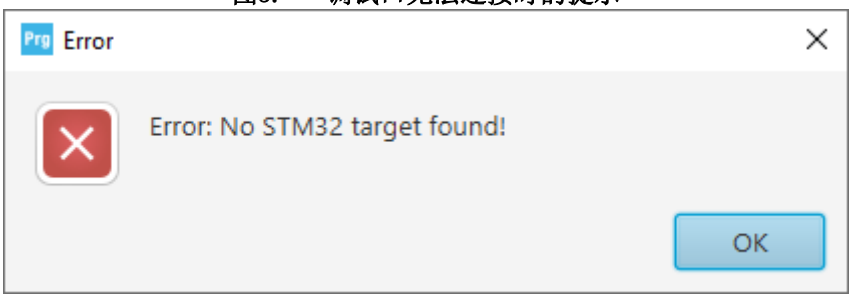
然后通过 STM32CubeProgrammer 修改选项字节，将 BOOT_LOCK 置位并且将 RDP 设置为 Level 1，如图 2 所示。

图2. 置位 BOOT_LOCK 并且设置 RDP 为 Level 1



修改以后，会发现 NUCLEO-C031C6 板上的 LD4 上电后不亮（代表上电调试口默认是关闭状态），并且此时 STM32CubeProgrammer 连接不到 STM32C031，如图 3 所示。

图3. 调试口无法连接时的提示



然后拨动 NUCLEO-C031C6 上的 B1 按钮，会发现 LD4 变亮（代表调试口打开），此时可以通过 STM32CubeProgrammer 进行连接，也就是调试口成功打开。然后再使用 STM32CubeProgrammer 将选项字节恢复（先将 RDP 设置为 Level 0，然后再取消 BOOT_LOCK）。

3.1.5. 结论

如果 BOOT_LOCK 置位与 RDP Level 1 无关联的话，则上电时默认关闭调试口，也就是将 DBU_SWEN 的复位值变为 0，与文档描述一致。

3.2. 测试二：测试 STM32C031 在 PCROP 全片保护的情况下是否可以执行 Reset_Handler

3.2.1. 测试目的

STM32C031 如果与 STM32G0 一样，在 PCROP 全片保护的情况下可以执行 Reset_Handler，则为了保险起见可以在测试选项字节失配的时候，在其中添加将 DBG_SWEN 置位（打开调试口）的代码，以避免出现芯片永久锁死无法恢复的情况。

3.2.2. 测试思路

在 Reset_Handler 中添加使 NUCLEO-C031C6 板上的 LD4 灯不停闪烁的代码。如果在 PCROP 全片保护的情况下，LD4 灯仍然可以不停闪烁，则说明 STM32C031 在 PCROPO 全片保护的情况下，可以执行 Reset_Handler；反之，则说明 STM32C031 在 PCROP 全片保护的情况下无法执行 Reset_Handler。

3.2.3. 测试代码

下面代码的功能为：配置 PA5 引脚（NUCLEO-C031C6 上的 LD4），不停翻转 PA5 引脚的状态，也就是令 LD4 灯不停闪烁。

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;; Configura PA5(LED4 pin)

      MOVS R0, #0x40 ; R0 = 0x40
      LSLS R0, R0, #24 ; R0 = 0x4000 0000
      MOVS R1, #0x21 ; R1 = 0x21
      LSLS R1, R1, #12
      ADDS R0, R0, R1 ; R0 = 0x4002 1000 ; RCC

      MOVS R1, #0x50 ; R1 = 0x50
      LSLS R1, R1, #24 ; R1 = 0x5000 0000 ; GPIOA

      ;;GPIOA Ports Clock Enable
      LDR R2, [R0, #0x34] ; RCC_IOPENR
      MOVS R3, #0x01
      ORRS R2, R2, R3
      STR R2, [R0, #0x34]

      ;;configure GPIO pin Output Level
      MOVS R2, #0x20
      STR  R2, [R1, #0x28] ; GPIOA_BRR
```

```
;;Configure IO Direction mode
LDR R2, [R1] ; GPIOA_MODER
MOVS R3, #0x05
LSLS R4, R3, #1 ; bit10(5*2)

MOVS R5, #0x03;
LSLS R5, R5, R4
MVNS R6, R5
ANDS R2, R6 ; Clear bit10~bit11

MOVS R5, #0x01
LSLS R5, R5, R4
ORRS R2, R2, R5
STR R2, [R1] ; Output

;;Configure the IO Speed(Low speed 0x00)
LDR R2, [R1, #0x08] ; GPIOA_OSPEEDR
MOVS R3, #0x05
LSLS R4, R3, #1 ; bit10(5*2)

MOVS R5, #0x03
LSLS R5, R5, R4
MVNS R6, R5
ANDS R2, R6
STR R2, [R1, #0x08]; Very low speed

;;Configure the IO Output Type(Output Push-pull)
LDR R2, [R1, #0x04] ; GPIOA_OTYPER
MOVS R4, #0x05

BICS R2, R4 ; Clear OTYPER_5
STR R2, [R1, #0x04] ; Output push-pull (reset state)

;;Configure the Pull parameters
LDR R2, [R1, #0x0C] ; GPIOA_PUPDR
MOVS R3, #0x05
LSLS R4, R3, #1 ; bit10(5*2)

MOVS R5, #0x03
LSLS R5, R5, R4
MVNS R6, R5
ANDS R2, R6
STR R2, [R1, #0x0C] ; No pull-up, pull-down

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Blink LED4(PA5_pin)

MOVS R0, #0x50 ; R0 = 0x50
LSLS R0, R0, #24 ; R0 = 0x5000 0000 (GPIOA)

MOVS R1, #0x01
```

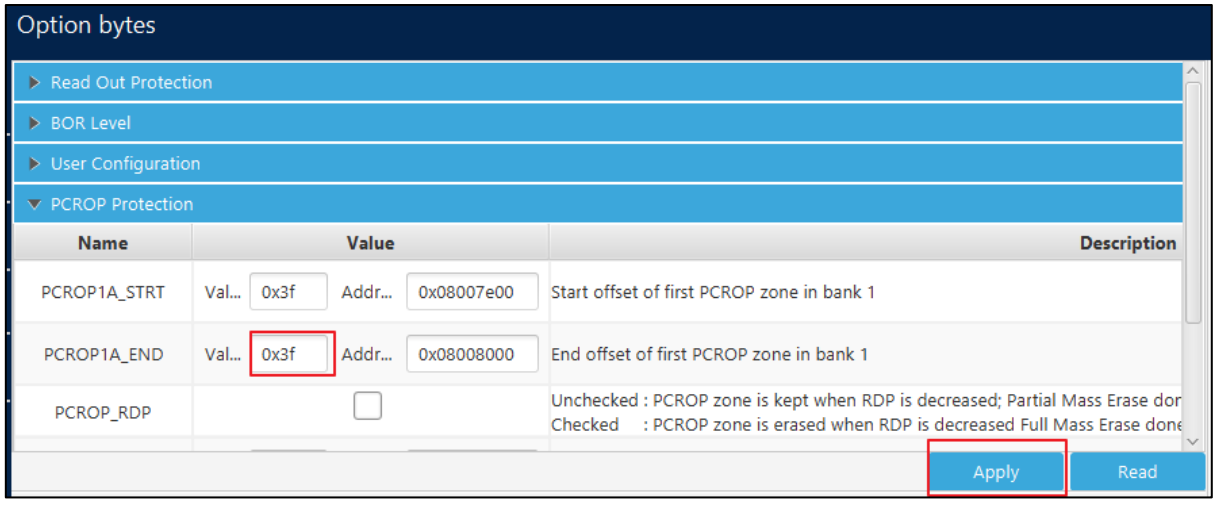
```
      LSLS R1, R1, #0x05 ; LD4(PA5: position 5)

ENDLESS_LOOP:
      ;;Toggle LED4
      LDR R2, [R0, #0x14] ; R2 = GPIOA_ODR
      ANDS R2, R1 ; R2 = R2 & 0x05
      MOVS R3, R2 ; R3 = R2
      EORS R2, R1 ; R2 = R2 ^ 0x05
      LSLS R3, R3, #16 ; R3 = R3 << 16
      ORRS R3, R3, R2 ; R3 = R3 | R2
      STR R3, [R0, #0x18] ; GPIOA_BSRR = R3

      ;; Delay
      MOVS R2, #0
      MOVS R3, #0xFF
      LSLS R3, R3, #10
DELAY:
      NOP
      ADDS R2, R2, #1
      CMP R2, R3
      BGT CLEAR_R2
      B DELAY
CLEAR_R2:
      MOVS R2, #0
      B ENDLESS_LOOP
```

3.2.4. 测试过程

将 3.2.3 中的代码添加到 startup_stm32c031xx.s 文件的 Reset_Handler 中，然后在 IAR 中编译并下载到 NUCLEO-C031C6 板中，会看到 LD4 灯不停闪烁。

接着通过 STM32CubeProgrammer 设置 STM32C031 为 PCROP 全片保护（令 PCROP1A_END 与 PCROP1A_START 相等），如图 4 所示。

图4. 配置 PCROP 全片保护



将 STM32C031 设置为 PCROP 全片保护后重启，NUCLEO-C031C6 板上的 LD4 灯停止闪烁。说明 STM32C031 在 PCROP 全片保护之后，没有执行 Reset Handler。

3.2.5. 结论

STM32C031 在 PCROP 全片保护的情况下，不能执行 Reset_Handler，没有出现 STM32G0 的 weakness。

3.3. 测试三：测试 STMC031 在选项字节失配时能否连接调试口

3.3.1. 测试目的

测试 STM32C031 在选项字节失配时，上电后调试口是否默认被打开。若在选项字节失配时，调试口默认是打开的，则说明分析正确。

3.3.2. 测试思路

执行程序不停修改选项字节，在芯片运行的时候突然断电，靠这种方法来使选项字节失配。

3.3.3. 测试代码

修改选项字节的步骤如下：

1. 清零 OPTLOCK 选项锁定位
2. 清零控制寄存器(FLASH_CR)中的 LOCK 位
3. 写入 FLASH 选项密钥寄存器(FLASH_OPTKYER)的 OPTKEY1=0x08192A3B
4. 写入 FLASH 选项密钥寄存器(FLASH_OPTKYER)的 OPTKEY2= 0x4C5D6E7F
5. 向 FLASH 选项寄存器(FLASH_OPTR)中写入需要修改的值
6. 检查 FLASH 状态寄存器(FLASH_SR)中的 BSY1 位，并确认当前未执行任何 FLASH 操作
7. 将 FLASH 控制寄存器(FLASH_CR)中的选项启动位 OPTSTRT 置 1.
8. 等待 BSY1 位清零。

当 BSY1 位清零后，所有的选项都将被更新到 Flash 中，但是并不会被应用到系统中。只有在选项字节加载后，更新的选项才会对系统有影响。在将 OPTSTRT 置 1 后，会自动计算补码值并将其写入补码选项字节。

选项字节有两种加载方式：

1. 将 FLASH 控制寄存器(FLASH)的 OBL_LAUNCH 置 1
2. 上电复位后(BOR 复位或退出 Standby/Shutdown 模式)

注意：将 OBL_LAUNCH 置位时会使芯片复位。

下面代码的功能是，串口不停打印选项字节中 nBoot1 位的状态，并且不停修改选项字节的 nBoot1 位的值。

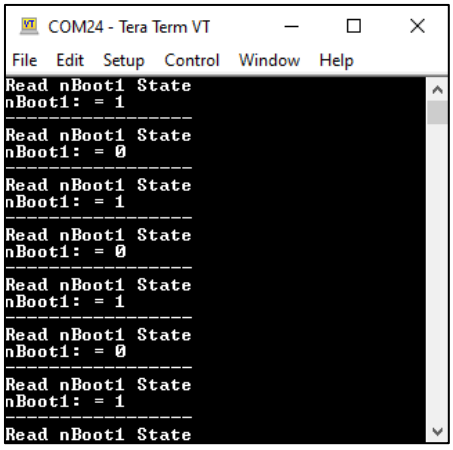
```
/* USER CODE BEGIN 2 */
  /* LED4 on */
  HAL_GPIO_WritePin(LD4_GPIO_Port, LD4_Pin, GPIO_PIN_SET);
```

```
/* Print the state of Option Bytes */
printf("Read nBoot1 State\r\n");
printf("nBoot1: = %d\r\n", !!READ_BIT(FLASH->OPTR, OB_USER_NBOOT1));
printf("----------------\r\n");

FLASH_OBProgramInitTypeDef OBInit;
HAL_FLASH_Unlock();
HAL_FLASH_OB_Unlock();
HAL_FLASHEx_OBGetConfig(&OBInit);

if(!READ_BIT(OBInit.USERConfig, OB_USER_NBOOT1))
{
  SET_BIT(OBInit.USERConfig, OB_USER_NBOOT1);
}
else
{
  CLEAR_BIT(OBInit.USERConfig, OB_USER_NBOOT1);
}

if(HAL_FLASHEx_OBProgram(&OBInit) != HAL_OK)
{
  printf("Failed to OBProgram\r\n");
}

if(HAL_FLASH_OB_Launch() != HAL_OK)
{
  printf("Failed to launch\r\n");
}

HAL_FLASH_OB_Lock();
HAL_FLASH_Lock();
/* USER CODE END 2 */
```

3.3.4. 测试步骤

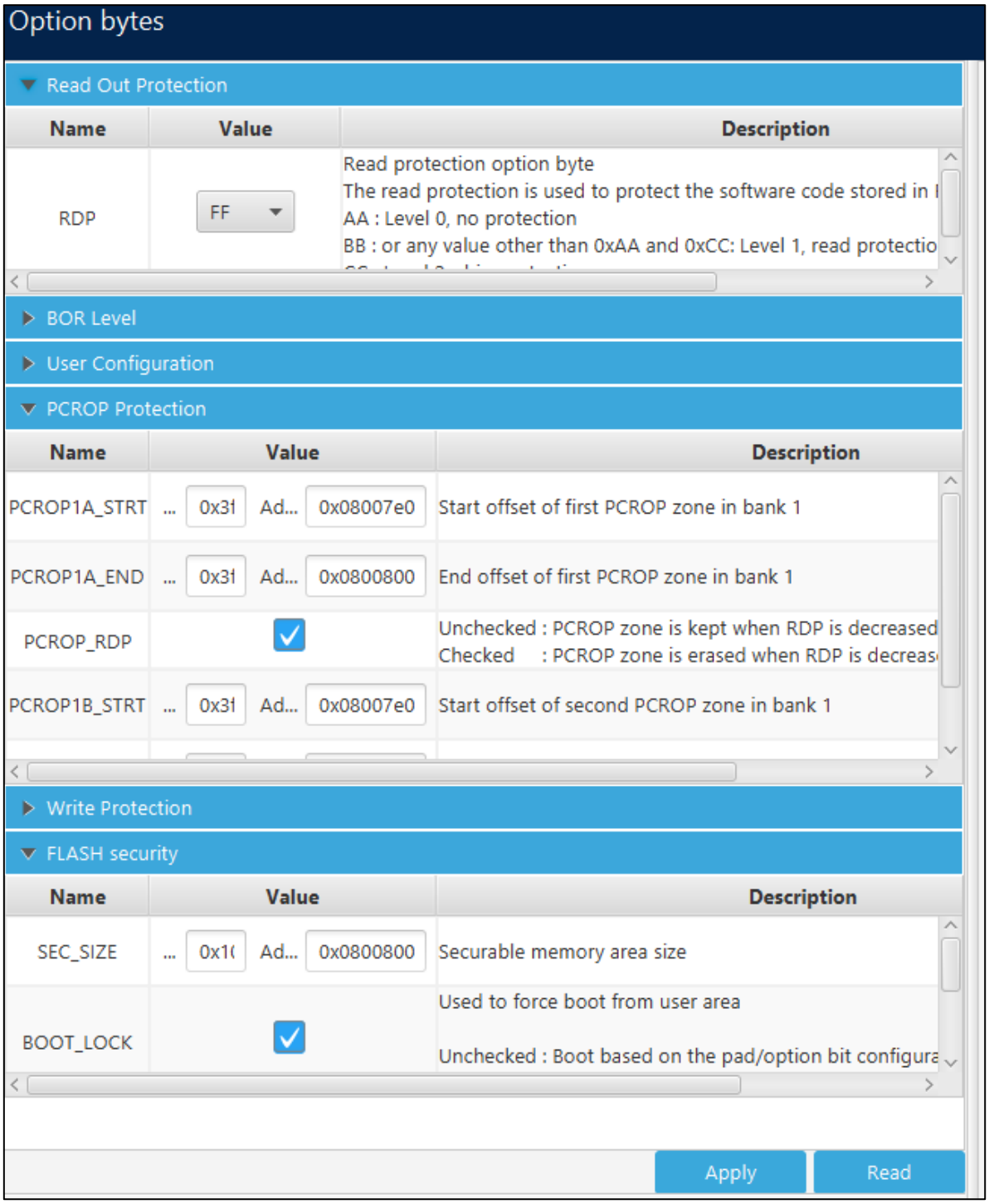
在 main 函数中添加 3.3.3 中的代码，编译后下载到 NUCLEO-C031C6 板中。然后连接串口工具（波特率 921600）后会看到 nBoot1 的状态不断被输出，如图 5 所示。

图5. 串口输出的信息



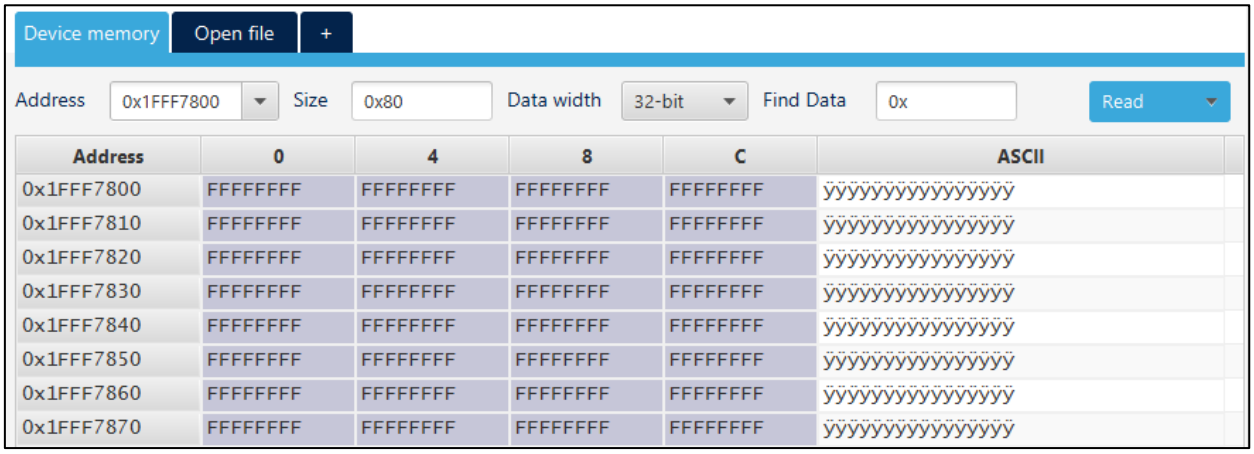
可以看到，选项字节中的 nBoot1 这个位的值不断被改变。这时给板子断电然后重新上电，如果上电后在串口中仍然有数据输出，则重复断电上电的操作。若上电后如果串口中没有数据输出，说明程序执行出了问题，很可能是选项字节失配导致的。如果分析正确，在选项字节失配时，调试口是可以连接的。使用 STM32CubePrammer 通过调试口进行连接时，发现确实是可以连接的。此时读出来选项字节的结果为：RDP1+PCROP 全片保护+BOOT_LOCK 置位。如图 6 所示。

图6. 选项字节失配时的状态



此时选项字节中的内容为如图 7 所示，通过图 6 和图 7 可以看出，此时选项字节失配。

图7.　选项字节中的内容

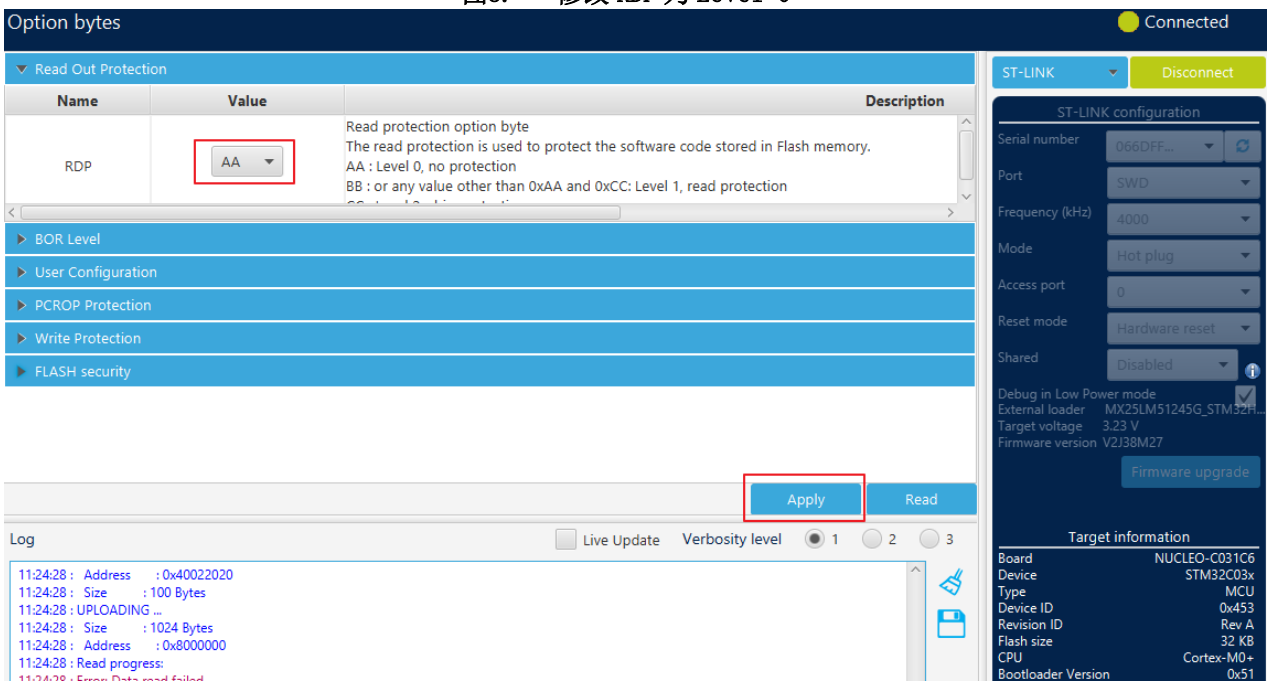


所以 STM32C031 在选项字节失配时，调试口仍然可以连接。

3.3.5. 选项字节恢复

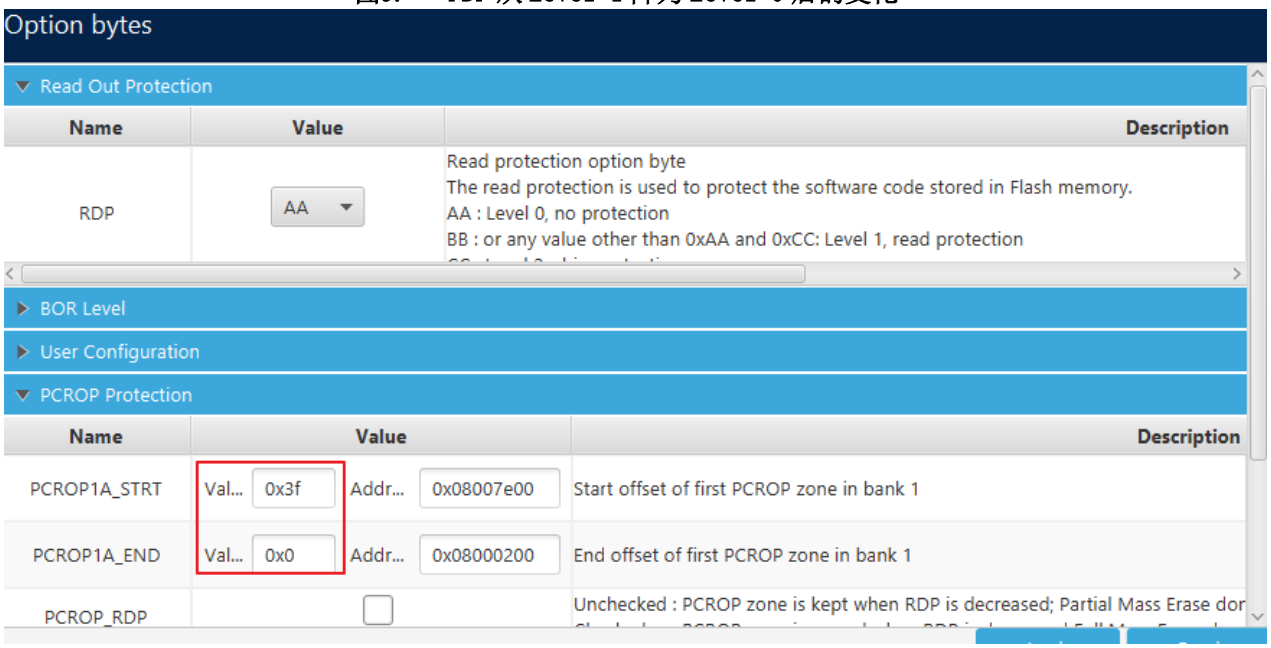
1. STM32CubeProgrammer 使用 SWD 的模式与 NUCLEO-C031C6 板连接
2. 将 RDP 修改为 Level 0，然后点击“Apply”选项，如图 8 所示。

图8.　修改 RDP 为 Level 0



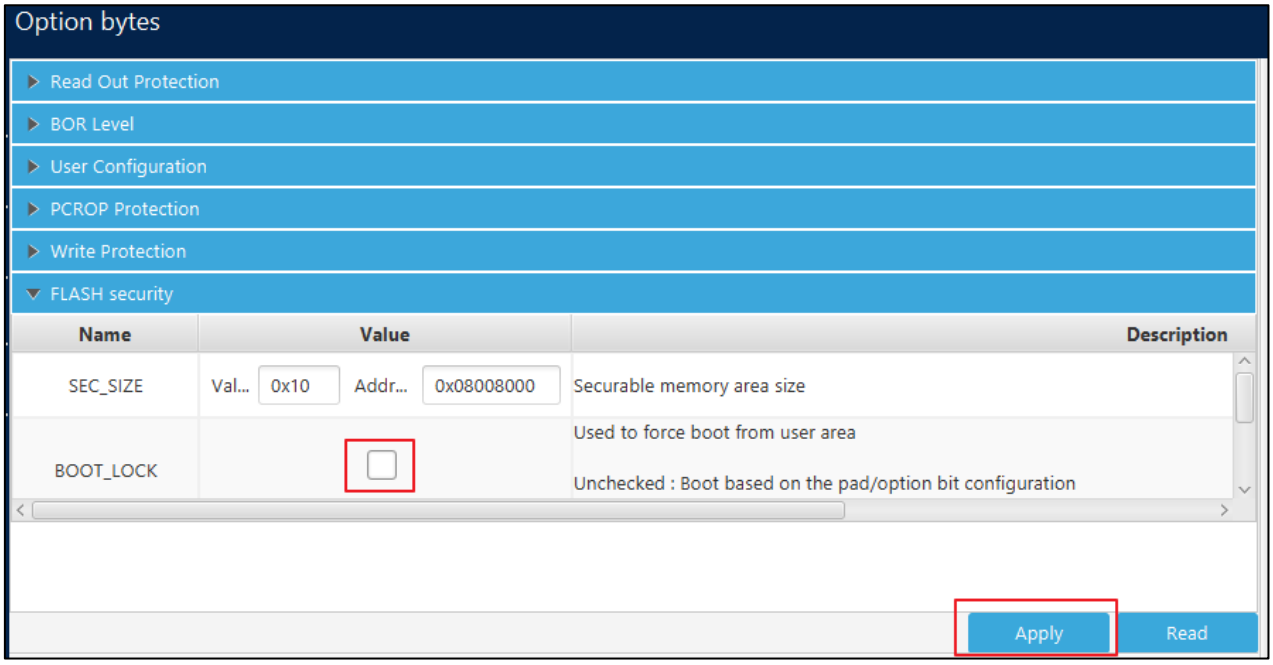
3. 将 RPD 修改为 Level 0 之后，可以看到 PCROP 全片保护已被取消（因为在选项字节失配时，PCROP_RDP 被置位），如图 9 所示。

图9.　PDP 从 Level 1 降为 Level 0 后的变化



4. 此时在 RDP Level 0 的时候取消 BOOT_LOCK 的勾选，然后点击“Apply”选项，如图 10 所示。执行成功后，BOOT_LOCK 已经被取消。选项字节已经被成功恢复。

图10.　取消 BOOT_LOCK



3.3.6. 结论

STM32C031 在选项字节失配后，仍然可以通过调试口来恢复选项字节，与分析一致。在选项字节失配时，导致的 BOOT_LOCK 置位与 RDP Level 1 是相关联的，这时芯片仍有调试能力。

4. 小结

STM32C031 在选项字节失配时，不会出现向 STM32G0 那样芯片永久锁死不能恢复的情况，默认会打开调试口，可以通过调试口来修改选项字节进行恢复。

参考文献

文件编号	文件标题	版本号	发布日期
RM0490	STM32C0x1 advanced Arm®-based 32-bit MCUs	Rev 0.2	22-Sep-2021

文档中所用到的工具及版本

[1]STM32MX, v6.99.16-B1
[2]STM32CubeProgrammer, v2.9.0-A02
[3]STM32Cube_FW_C0, v0.2.1

LAT 中的附件

[1] 测试 STM32C031 的 BOOT_LOCK 位_v0.2.zip

版本历史（内部参考）
(仅供内部参考，评审完成后 Auditor 需要删除这部分内容）

日期	版本	变更	撰写/变更者
2020 年 12 月 14 日	0.1	初始版本	Leo ZHANG
2022 年 01 月 20 日	0.2	更新文档版本和内容 review 与验证	Ke DENG

版本历史

日期	版本	变更
YYYY 年 MM 月 DD 日	1.0	详细变更信息

重要通知 – 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。若需 ST 商标的更多信息，请参考 *www.st.com/trademarks*。所有其他产品或服务名称均为其 各自所有者的财产。

本文档是 ST 中国本地团队的技术性文章，旨在交流与分享，并期望借此给予客户产品应用上足够的帮助或提醒。若文中内容存有局限或与 ST 官网资料不一致，请以实际应用验证结果和 ST 官网最新发布的内容为准。您拥有完全自主权是否采纳本文档（包括代码，电路图等）信息，我们也不承担因使用或采纳本文档内容而导致的任何风险。

本文档中的信息取代本文档所有早期版本中提供的信息。