目录

1 简介

赛元 ARM 系列 MCU ，CPU 跟随系统时钟选择，可选择 HIRC、高频外部晶振、PLL 时钟源（SC32F10 系列），IAP 进行 FLASH 或 EEPROM 写入操作时，固定使用 HIRC 时钟源。当 CPU 与 FLASH 不使用同一时钟源时（系统时钟主频使用高频外部晶振、PLL），在高频率、连续进行 IAP 写操作时，CPU 与 FLASH 的时钟不一致导致时钟异步，有概率引起 CPU 在 FLASH 执行指令与 IAP 写操作冲突，意外进入硬件错误中断，影响系统的稳定性和可靠性。

在干扰环境下，MCU 存在因程序计数器（PC 指针）受到干扰而发生程序跑飞现象，从而有一定概率导致错误地址的 FLASH ROM /EEPROM 区域被误擦写，如果正好擦除/改写了程序代码区域，就会造成不可恢复的程序运行异常。详见《赛元 MCU 干扰环境下 IAP 读写安全应用指南》。

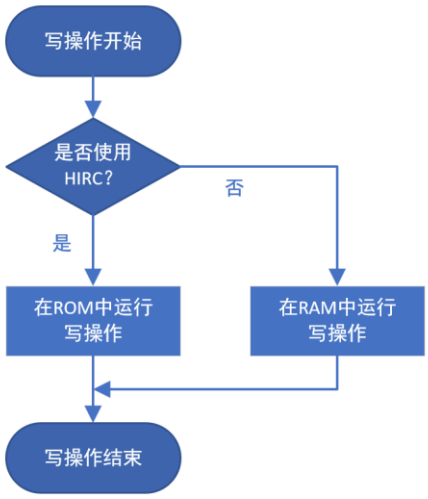
针对以上问题，用户需要按本文档所规范的程序流程进行 SC32 系列 MCU 的 IAP 应用开发。

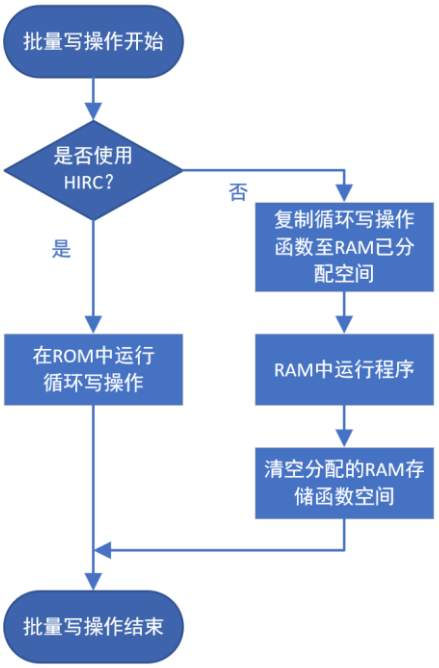
2 解决方案

2.1 使用高频外部晶振、PLL 时 IAP 异常解决方案

在系统主频**使用高频外部晶振、PLL 时钟源**的 IAP 应用中，将 IAP 写操作指令放至 RAM 中运行，以避免 CPU 在 FLASH 执行指令与 IAP 写操作发生冲突。由于指令在 RAM 的运行速度快，为协调 FLASH 速度，需要执行两次写操作指令，单 BYTE 的写入时间将延长至原来的两倍。以上操作已集成至 BSP 中，用户程序 IAP 操作程序的相关接口与此前版本的程序接口一致，不需要特别修改。关于函数的使用请参考 2.2 节。

BSP 中提供了三个单数据写操作函数，即 IAP_ProgramWord、IAP_ProgramHalfWord、IAP_ProgramByte，相关介绍请参阅《赛元 SC32F1XXX 系列固件库使用手册》。在以上函数中对于是否使用高频外部晶振、PLL 做了区分，根据用户使用的系统时钟类型自动选择合适的方式运行程序，流程图如下左图。



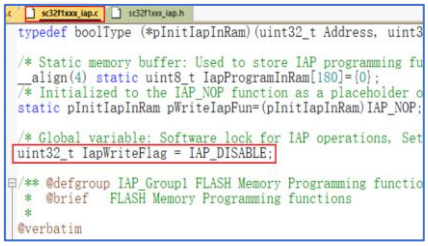


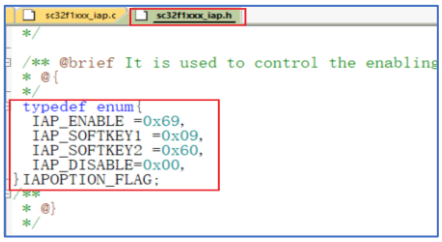
在 BSP 中还提供了三个对批量数据进行写操作的函数，即 IAP_ProgramWordArray 、IAP_ProgramHalfWordArray、IAP_ProgramByteArray，相关介绍请参阅《赛元 SC32F1XXX 系列固件库使用手册》。在这里对于是否使用高频外部晶振、PLL 也同样做了区分，可以根据用户使用的系统时钟类型自动选择合适的方式进行写操作，流程图如上右图。

2.2 干扰环境下 IAP 异常解决方案

在 IAP 原有硬件写保护锁的基础上，再加上软件安全流程锁(IapWriteFlag)，采用最少两重软件安全锁机制，避免因为干扰导致 PC 指针跑飞而产生 FLASH 数据被篡改，因此需要确保程序非法跳转到任意一个单点地址，都无法在同一逻辑分支一次顺序完成解除 IAP 保护锁、擦除/写入 IAP。

安全流程锁的相关定义：

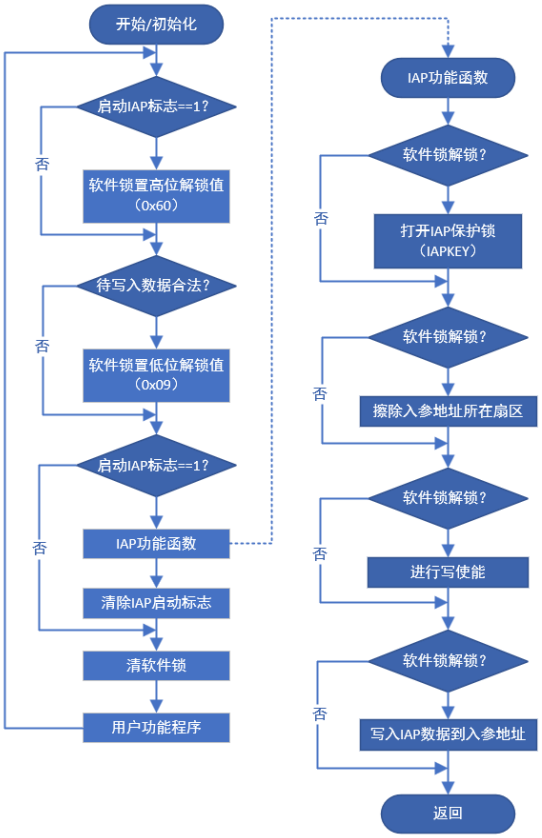




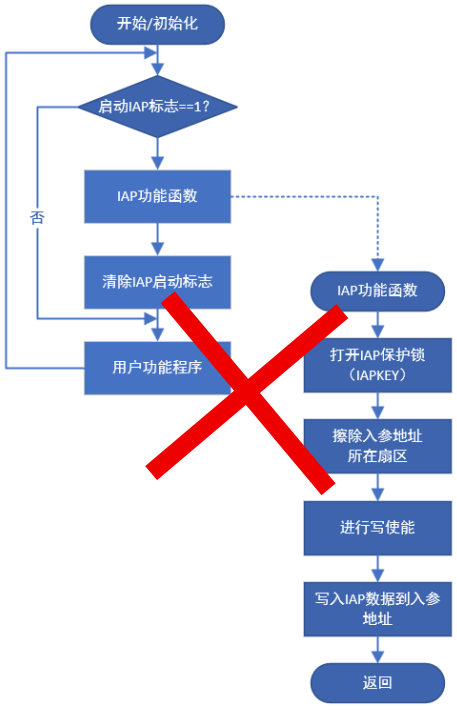
安全流程锁保护功能逻辑已在 BSP 中实现，在操作 IAP 时没有解开安全流程锁（IapWriteFlag!=ENABLE），IAP 写入将失败，用户在进行 IAP 开发时，务必按照以下步骤进行解锁：

A：系统运行中，IAP 擦/写操作的启动，应有两个不一样的条件确认流程来检查是否符合 IAP 写入条件；
B：若符合 IAP 写入条件 1，则写入第一重软件锁；
C：继续检查是否符合 IAP 写入条件 2；
D：若符合 IAP 写入条件 2，则写入第二重软件锁；
E：不同的执行步骤（解锁/擦除/写使能/写入），都需要再次确认软件安全流程锁是否合法；
F：若软件锁合法，才打开 IAP 保护锁（IAPKEY），执行 IAP 擦除/写使能/写入操作；
G：IAP 操作流程返回后，无论是否执行了 IAP 擦除/写入，都将软件锁归零。

用户在进行实际 IAP 开发时可参考以下操作流程（详见《赛元 MCU 干扰环境下 IAP 读写安全应用指南》）：



安全，多重安全确认，不在同一逻辑分支顺序开启IAP保护锁、擦除/写入



不安全，同一逻辑分支顺序一次性开启IAP保护锁、擦除/写入

3 示例说明

以下是一个基于赛元 MCU IAP 读写安全策略的示例说明，展示了如何通过两重软件安全锁（按键和数据有效）来防止 FLASH 在干扰环境下程序跑飞导致的意外篡改。

```
static boolType CheckSaveData(void)
{
  // 实现数据有效性检查逻辑，假设数据合法返回 TRUE，否则返回 0
  return TRUE; // 示例中假设数据总是合法
}
 void main(void)
{
  GPIO_Init(); //GPIO 初始化
  while(1)
   {
    // 检测按键是否按下
    if(GPIO_ReadDataBit(GPIOC,GPIO_Pin_1) == 0)
    {
      if(Key_Press == 0)
       {
        Key_Press = 1;
        Iap_Enflag = 1; // 按下按键，允许 IAP 写入，作为解锁的第一个必要条件
      }
    }
    Key_Press = 0;
     // 检查第一重安全锁标志
    if(Iap_Enflag == 1)
    {
      IapWriteFlag = IAP_SOFTKEY1; // 允许写入 IAP，第一重安全锁解锁
    }
     // 检查数据合法性
    if(CheckSaveData() == TRUE)
    {
      IapWriteFlag |= IAP_SOFTKEY2; // 待写入数据合法性，作为解锁的第二个必要条件，第二重安全锁解锁
    }
     // 执行 IAP 操作（仅在两重安全锁均解锁时，此逻辑已在 BSP 函数中实现）
    if(IapWriteFlag == IAP_ENABLE)
    {
      // 打开 IAP 数据保护锁
    }
    if(IapWriteFlag == IAP_ENABLE)
    {
      // 执行 IAP 擦除
    }
    if(IapWriteFlag == IAP_ENABLE)
    {
      // 执行 IAP 写使能
    }
    if(IapWriteFlag == IAP_ENABLE)
    {
      // 执行 IAP 写入
    }
    IAP_Lock(); // IAP 上锁操作
    // IAP 操作完成后，重置安全锁标志
    Iap_Enflag = 0;
    IapWriteFlag = 0;
  }
}
```

说明：

（1）IapWriteFlag 定义在 BSP 中 sc32f1xxx_iap.c 文件里，默认值为 IAP_DISABLE（0x00）。

（2）按键检测：当检测到按键按下时，设置 Iap_Enflag 为 1，表示第一重安全锁已解锁。

（3）数据有效性检查：调用 CheckSaveData()函数检查数据合法性，若合法则设置 IapWriteFlag 的第二位，表示第二重安全锁已解锁。

（4）IAP 操作执行：只有当两重安全锁均解锁（IapWriteFlag == IAP_ENABLE）时，才执行 IAP 解锁（数据保护锁）/擦除/写使能/写入操作。

（5）重新使能数据保护锁（调用函数 IAP_Lock()）。

（6）安全锁重置：IAP 操作完成后，重置 Iap_Enflag 和 IapWriteFlag，确保下次操作需要重新解锁。

4 注意事项

（1）遵循两重软件安全锁机制，只有两重锁均解锁才允许 IAP 操作；

（2）IAP 操作完成后，无论是否执行了擦除/写入，都要将软件锁归零，并确保 IAPADE 指针指回 CODE 区，防止程序跑飞；

（3）若使用外部晶振或 PLL 作为系统时钟，MCU 会在 RAM 中执行两次写操作，写入耗时会变为两倍时长。

5 常见问题及处理方法

以下是客户在进行 IAP 开发过程中常见的异常问题及其解决方法：

编号	异常现象	处理方法
1	擦除失败	1. 数据保护锁是否通过 IAPKEY 解锁； 2. 软件安全流程锁是否打开（IapWriteFlag = 0x69）； 3. 烧录选项中“Flash sectors protection”是否关闭需要进行 IAP 操作的区域的写保护；
2	写入失败	1. 数据保护锁是否通过 IAPKEY 解锁； 2. 软件锁是否打开（IapWriteFlag = 0x69）； 3. 烧录选项中“Flash sectors protection”是否关闭需要进行 IAP 操作的区域的写保护； 4. 写使能（IAP_WriteCmd）是否开启；
3	程序代码区域出现被部分擦除或改写	1. 烧录选项中“Flash sectors protection”是否对程序代码区域使能了写保护； 2. ISP 或 OTA 更新固件后，是否正确重新在程序代码区域使能了写保护； 3. 是否正确的使用了软件安全流程锁功能；
4	写入时长增加	1. 如果使用了外振或 PLL，那么 IAP 将会在 RAM 中执行两次写操作，导致写入时长增加一倍；

6 规格更改记录

版本	记录	日期
V1.0	初版	2025 年 06 月

7 声明

深圳市赛元微电子股份有限公司（以下简称赛元）保留随时对赛元产品、文档或服务进行变更、更正、增强、修改和改进的权利，恕不另行通知。赛元认为提供的信息是准确可信的。本文档信息于 2025 年 7 月开始使用。在实际进行生产设计时，请参阅各产品最新的数据手册等相关资料。