目录

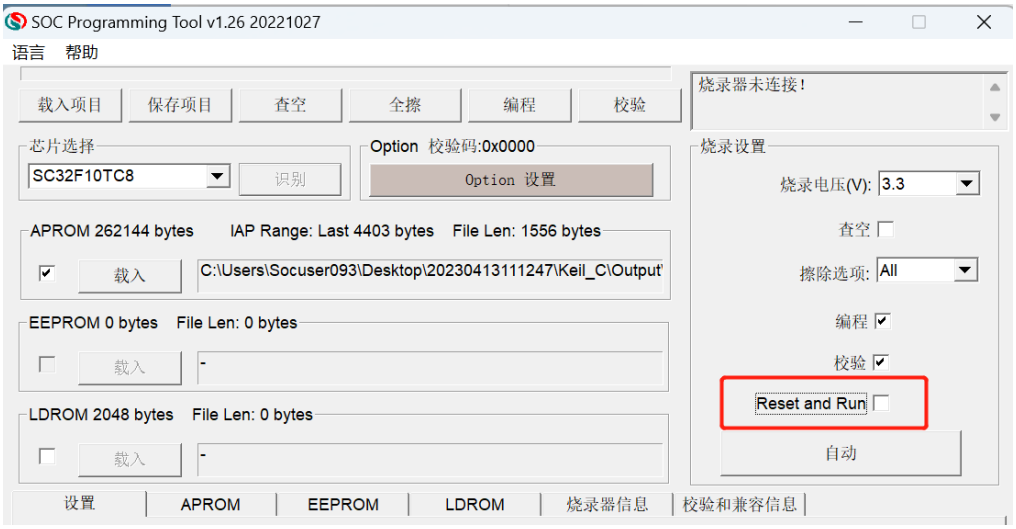
1 赛元 SC32F1XXX 系列 MCU 电气参数注意项

工作电压：2.0V~5.5V
工作温度：-40 ~ 105℃
内核：Cortex®-M0+内核，带 WIC 模块和 MPU 模块
Flash ROM：可重复写入 10 万次, 25℃环境下数据可保存 100 年以上
LDROM：出厂固化 BootLoader 程序
时钟源：

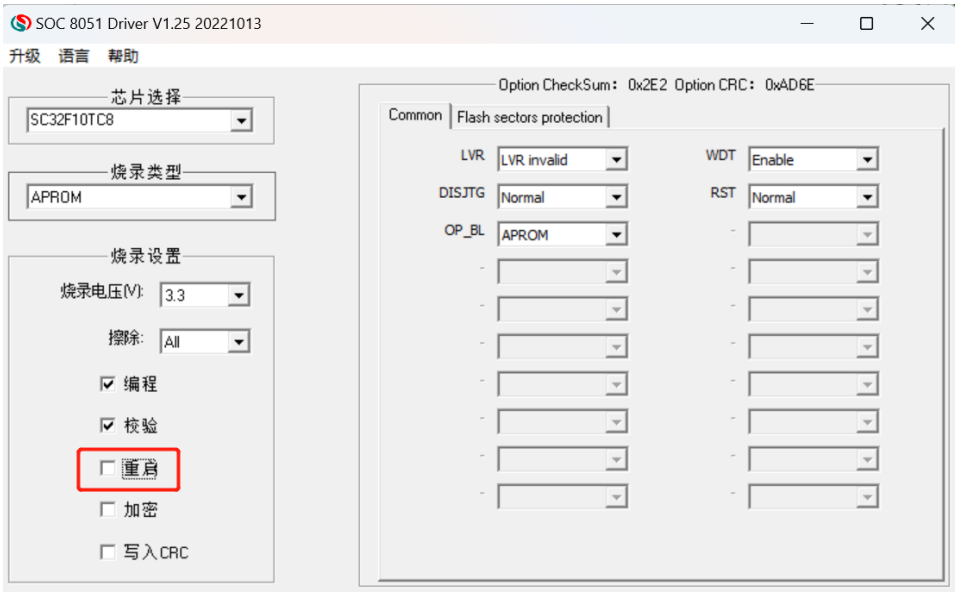
1. 内建高频振荡器（HIRC）
   可作为系统时钟源和 PLL 时钟源。频率误差：跨越 (2.0V~5.5V) 及 (-40 ~ 105℃) 应用环境, 不超过 ±1%。
2. 内建低频 32kHz 低频振荡器（LIRC）
   可作为系统时钟源、Base Timer、LCD/LED 时钟源，固定为 WDT 时钟源。频率误差：跨越 (4.0 ~ 5.5V) 及 (-20 ~ 85℃) 应用环境，经寄存器修正后频率误差不超过 ±4%。
3. 可外接 2~16MHz 高频晶振（HXT）
   可作为系统时钟源和 PLL 时钟源。用户可以选择外接晶振振荡频率<12MHz 或≥12MHz。
4. 可外接 32.768 KHz 低频晶振（LXT）
   可作为系统时钟源、Base Timer、LCD/LED 时钟源。可通过 LXT 对 HIRC 进行自动校准。
5. PLL
   可作为系统时钟源。PLL 时钟来源可选 HIRC 或 HXT，输出 PLLRCLK 可多达 64MHz，可作为系统时钟。

2 赛元 SC32F1XXX 系列 MCU 烧写注意事项

1. 赛元 SC32F1XXX 系列芯片的 CLK 或 DIO 管脚对 GND 的电容不得超过 100pF，VDD 对 GND 的电容不可超过 1000uF。
2. 烧录引出点与芯片之间尽量不要串电阻，如无法避免，应保证串接电阻的阻值不超过 100R，且烧录时要尽量缩短烧录排线。
3. 电路设计时应避免将芯片的 CLK 和 DIO 连到同一个数码管上。
4. 请使用 SC LINK Pro 配合烧录上位机 SOC Programming Tool 进行烧录。
5. SC32F1XXX 系列芯片脱机烧录时，如果程序上电有 IAP 写的操作，则会校验失败，原因为 SC32F1XXX 系列脱机校验为 CRC 校验，程序跑起来导致 ROM 数据改变，从而校验失败。解决方法两种：
   (1)在初始化之前加一段时间(建议 100ms)的延时才能烧录成功。
   (2)在烧录上位机和 Keil 插件的配置页面中，取消勾选“Reset and Run”或者在 KEIL 设置界面取消勾选“重启”：



烧录上位机 SOC Programming Tool 设置页面



KEIL 插件的设置页面

3 仿真注意事项

“工程取消勾选 MicroLIB 后无法 Debug”问题解决：

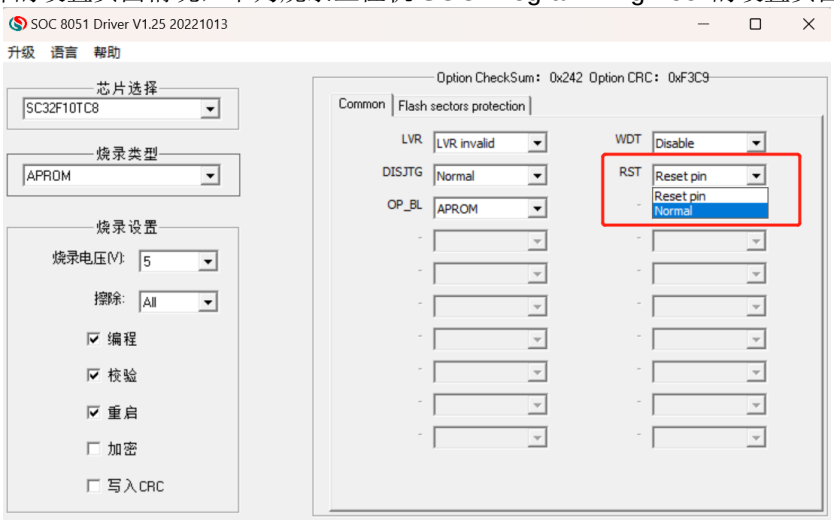
若 Keil 工程中使用到了 printf()和 scanf()等函数，需要加入以下代码退出半主机模式（STM32 默认是开启半主机模式）：

```
/* 告知连接器不从 C 库链接使用半主机的函数 */
// 标准库需要的支持函数
struct __FILE
{
    int handle;
};

// 定义_sys_exit()以避免使用半主机模式
void _sys_exit(int x)
{
    x = x;
}

FILE __stdout;
```

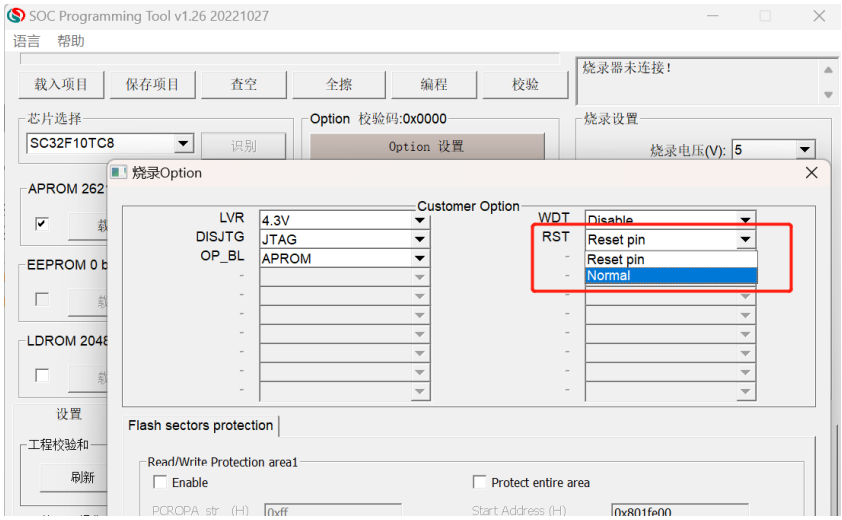
4 电路设计的注意事项

赛元 SC32F1XXX 系列 MCU 的 GPIO 上电默认模式为高阻输入模式。

赛元 SC32F1XXX 系列 MCU 的 RST 管脚，通过 Option 选项可设置为 Reset pin 或 GPIO。RST 管脚作为 Reset pin 时，低电平使能；作为 GPIO 时，管脚低电平不会产生复位。Option 的设置如图（以 SC32F10TC8 为例）。上为 KEIL 插件的设置页面情况，下为烧录上位机 SOC Programming Tool 的设置页面情况。



KEIL 插件的设置页面情况



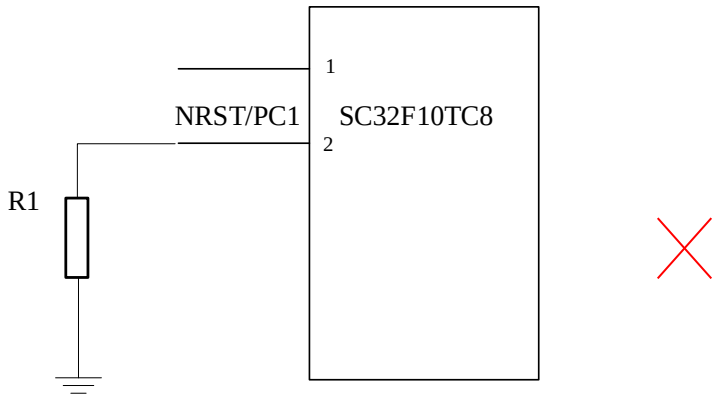
烧录上位机 SOC Programming Tool 的设置页面情况

4.1 电路设计实例

4.1.1 RST 管脚电路

赛元 SC32F1XXX 系列 MCU 的 NRST 引脚，与 I/O 复用，既可以做输入，又可以做输出，有别于传统 MCU 的 RST 引脚（传统的 RST 的引脚只能做输入，不能做输出）。当将 IO 口设置为复位口时，上电后，用户电路的 NRST 口不能一直为低，否则会一直复位，无法正常工作。因此，用户设计电路时，需要注意：

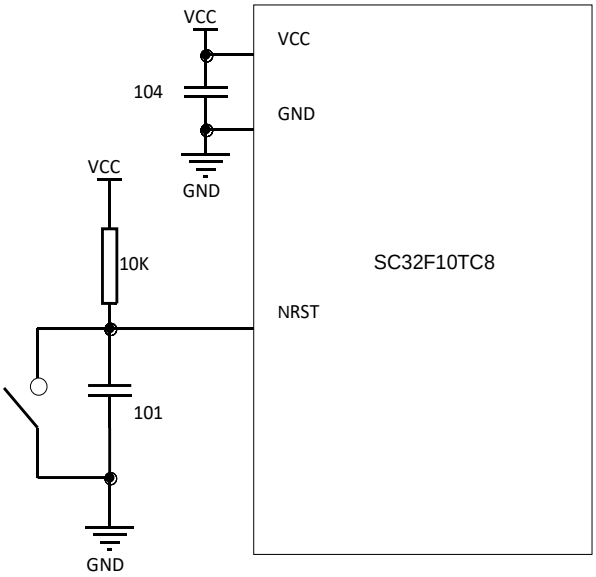
错误接法：



错误接法图示

说明：以上电路，若 NRST 外接一个电阻 R1，系统在上电时与内部上拉判为低电平，造成系统一直复位，无法正常工作。

建议接法：

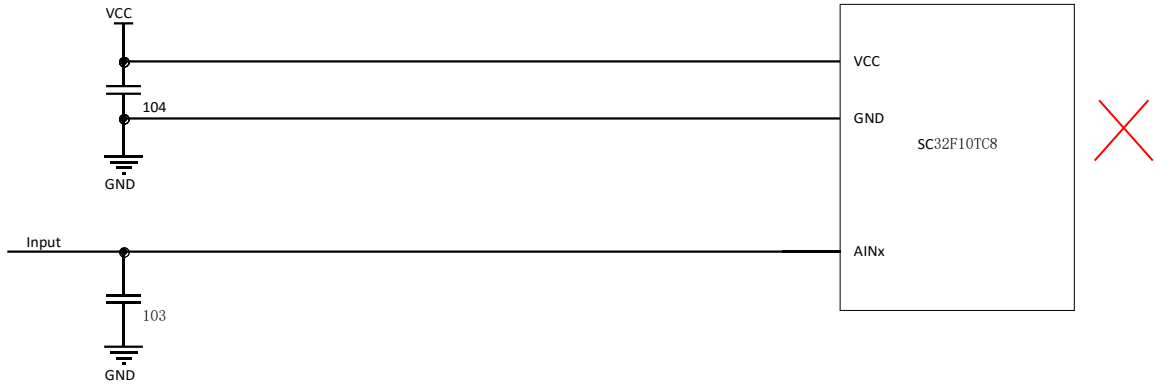


建议接法图示

4.1.2 ADC 采样管脚电路

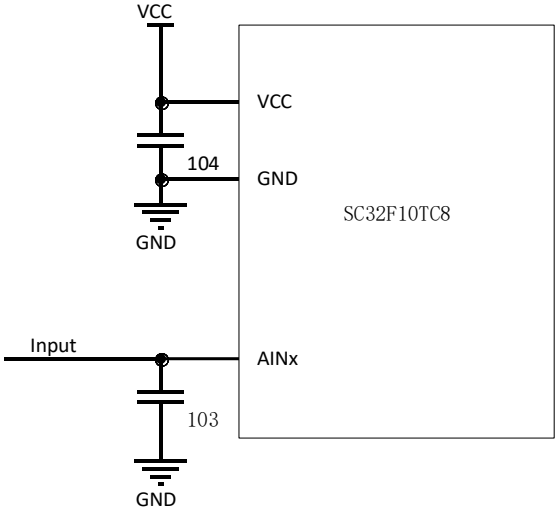
赛元 SC32F1XXX 系列 MCU 的 ADC 采样口需要在靠近管脚处加 103 电容， ADC 转换需要让电源电压稳定，所以在使用 ADC 功能时，请在靠近 IC 的 VCC 和 GND 处加 104 电容，以保证转换结果准确。

错误接法：104 电容离电源脚太远， 103 电容离 ADC 采样口太远。



错误接法图示

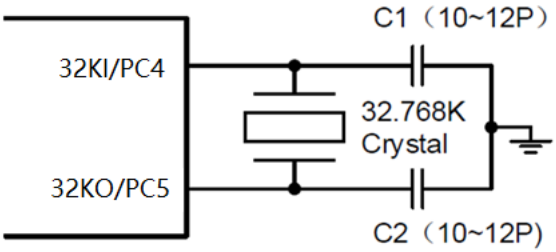
建议接法：104 电容靠近电源脚， 103 电容靠近 ADC 采样口。



建议接法图示

4.1.3 外部晶振电路

赛元 SC32F1XXX 系列 MCU 提供了高频外部晶振接口（OSCI/OSCO）和低频外部晶振接口（32KI/32KO），用户如需使用外部晶振，匹配电容请根据所选晶振的要求进行选择，请将晶振电路靠近芯片引脚处。如下为 32.768K 外部晶振连接图：



32.768K 外部晶振连接图

4.1.4 TOUCHKEY 电路

赛元触控 MCU 的触控架构为高灵敏度触控模式。
高灵敏度触控模式外接的 CMOD 电容容值范围为 472~104，推荐使用 103 电容，电容材质无特殊要求。
CMOD 电容需要尽量靠近 CMOD 芯片管脚。

4.2 IO 口各模式设置注意事项

赛元 SC32F1XXX 系列 MCU 的 GPIO，有三种工作模式：

1. 带上拉输入模式
2. 高阻输入模式
3. 强推挽输出模式

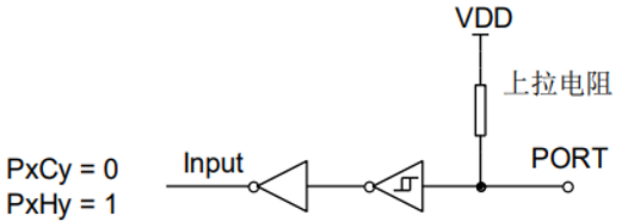
注意：未使用及封装未引出的 IO 口均要设置为强推挽输出模式。

4.2.1 I/O 设为高阻，实现电路设计

通常来说，对于某些特定场合的应用，譬如电压检测，过零检测，LCD 的应用等，都是采用高阻工作模式来实现的。因此说，用户可以从赛元 MCU 体系中按需选择。

4.2.2 带上拉输入模式

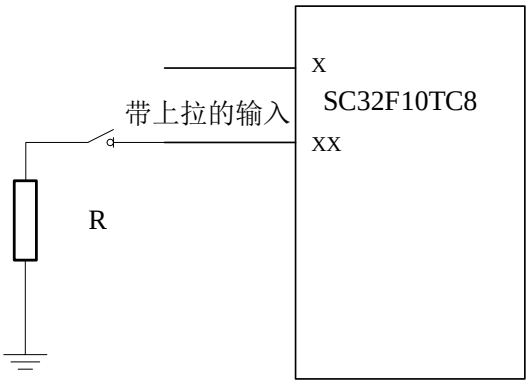
带上拉的输入模式下，输入口恒定接一个上拉电阻，仅当输入口上拉电平被拉低时，才会检测到低电平信号。带上拉的输入模式的端口结构示意图如下：



带上拉的输入模式

4.2.3 带上拉输入模式检测按键

I/O 口作按键输入时，串接的下拉电阻 R 需小于 10K。



上拉模式的按键输入接法图示

4.2.4 I/O 开漏输出模式的实现方式

赛元 SC32F1XXX 系列 MCU 没有开漏输出设置项，若用户想让 IO 口实现开漏输出功能，需要通过切换模式以达到开漏输出的效果，需要引脚输出低时切换为强推挽输出模式，需要引脚保持悬空状态时，则将 IO 口切换为高阻输入模式即可。

代码示例如下：

```
GPIOC->PXPH &= 0XFFFE;        //去除 PC0 的上拉电阻
GPIOC->PIN  &= ~0X0001;       //将 PC0 输出 0
GPIOC->PXCON &=  (uint32_t)~ 0X0001;     //PC0 设置为输入模式，等效为开漏输出的开漏状态
GPIOC->PXCON |=  (uint32_t) 0X0001;      //PC0 设置为输出模式，等效为开漏输出低
```

4.2.5 I/O 与或操作注意事项

赛元 SC32F1XXX 的 CPU 执行指令速度快，在执行下一条指令时 GPIO 上的电平可能还没来得及完成改变，所以请不要对一个 GPIO 寄存器通过做连续的与或操作的方式来改变 IO 口的输出状态，如下：

```
GPIOC->PIN |= 0x04;
GPIOC->PIN |= 0x08;
```

以上操作请合并为一条实现，如下：

```
GPIOC->PIN |= 0x0C;
```

或者改为位操作，如下：

```
PC_BIT(2) = 1;
PC_BIT(3) = 1;
```

5 软件编写的注意事项

赛元 SC32F1XXX 系列 MCU 内含有丰富的外设，只要配置相应的寄存器即可对其实现操作，但一些操作需要按要求进行，用户编程过程中需要注意以下几点。

5.1 ADC 多通道切换采集注意事项

赛元 SC32F1XXX 系列 MCU 拥有多路 ADC 通道，但每次转换只能转换一个通道，若想实现多路通道的 ADC 信号的采集，需要在一路 ADC 通道转换完毕后将转换口切换至另一路 ADC，如此反复以实现多通道的 ADC 转换。若在 ADC 通道切换后马上进行 AD 转换，通道口线上的电压可能存在不稳定的现象，在切换通道后转换的第一个值可能会存在异常，建议用户对某个通道做连续的多次采集和转换后，将切换通道后转换的第一个值或几个值去除或将最大值及最小值去除，再将剩余的 AD 转换值求平均值得到采集结果。

使用示例如下：

```
unsigned int ADC_Value0,ADC_Value1,ADC_Value2;
unsigned int ADC_Convert(void)
{
    unsigned  int Tad=0,MinAd=0xffff,MaxAd=0x0000,TempAdd=0;
    unsigned char t=0;
    for(t=0;t<10;t++)
    {
        ADC->ADC_CON |= 0X80;                       //开始 ADC 转换
        while(!(ADC->ADC_STS & 0X01));              //等待 ADC 转换完成
        ADC->ADC_STS &= ~0X01;                  //清中断标志位
        Tad = (unsigned int) ADC->ADC_VALUE;        //取得一次转换值
        if (Tad>MaxAd)
            MaxAd=Tad;                              //获得当前的最大值
        if (Tad<MinAd)
            MinAd=Tad;                              //获得当前的最小值
        TempAdd+=Tad;                               //转换值累加
    }
    TempAdd-=MinAd;                             //去掉最小值
    TempAdd-=MaxAd;                             //去掉最大值
    TempAdd>>=3;                                //求平均值
    return(TempAdd);
}

void ADC_SetChannel(uint16_t ADC_Channel)
{
    ADC->ADC_CON &= (uint32_t) ~0X0F;
    ADC->ADC_CON |= ADC_Channel;            //ADC 输入选择为 ADCchannel 口
}

void ADC_Multichannel()
{
    ADC->ADC_CFG = 0X07;    //设置 AIN0、AIN1、AIN2 设置为 ADC 口，并自动将上拉电阻移除
    ADC->ADCCON  |= 0X8000;     //开启 ADC 模块电源
    ADC_ SetChannel (0);            //ADC 入口切换至 AIN0 口
    ADC_Value0 = ADC_Convert();  //启动 ADC 转换，获得转换值
    ADC_ SetChannel (1);            //ADC 入口切换至 AIN1 口
    ADC_Value1 = ADC_Convert();  //启动 ADC 转换，获得转换值
    ADC_ SetChannel (2);            //ADC 入口切换至 AIN2 口
    ADC_Value2 = ADC_Convert();  //启动 ADC 转换，获得转换值
}
```

5.2 软件操作 CODE OPTION 的注意事项

赛元 SC32F1XXX 系列的 MCU 内部有单独的一块 Flash 区域用于保存客户的上电初始值设置，此区域称为 Code Option 区域。用户在烧写 IC 时将此部分代码写入 IC 内部，IC 在复位初始化时，就会将此设置调入 SFR 作为初始设置。

Option 相关 SFR 的读写操作由 OPINX 和 OPREG 两个寄存器进行控制，各 Option SFR 的具体位置由 OPINX 确定，各 Option SFR 的写入值由 OPREG 确定：

寄存器	地址	说明	复位值
OPINX	0x4000_03F8	Customer Option 指针	0x0000_0000
OPREG	0x4000_03FC	Customer Option 寄存器	0x0000_0000
OPT_CON0	0XC1 @ OPINX	Customer Option 映射寄存器 0	0x0000_0000
OPT_CON1	0XC2 @ OPINX	Customer Option 映射寄存器 1	0x0000_0000

操作 Option 相关 SFR 时 OPINX 寄存器存放相关 OPTION 寄存器的地址，OPREG 寄存器存放对应的值。

例如：要将 OPT_CON1 配置为 0x40，具体操作方法如下：

C 语言例程：

```
RCC->RCC_KEY = 0xFF;    //开启 option 时钟
RCCAHB->AHB_CFG |= 0x04;
OPT->OPINX = 0xC2;                  //将 OPT_CON1 的地址写入 OPINX 寄存器
OPT->OPREG = 0x40;                  //对 OPREG 寄存器写入 0x40（待写入 OPT_CON1 寄存器的值）
```

5.3 CRC 使用注意事项

赛元 SC32F1XXX 系列 MCU 内建 1 个硬件 CRC 模块，仅支持软件送数计算模式，对指定数据进行 CRC 运算处理。将需要进行 CRC 计算的数据写入 CRC 数据寄存器 CRCDR，当需要读取 CRC 计算结果时，再从 CRCDR 读出。

用户可通过修改 CRC 多项式设置寄存器值选择多项式公式，在 CRC 控制寄存器选择多项式的大小，支持 7bit/8bit/16bit/32bit。首次进行 CRC 计算前，必须先对 CRCRST 写 1，将 CRCDR 复位为全 1。

注意：

1. CRCDR 写入数据和读出不是同一数据。
2. 写入 CRCDR 寄存器，写入的数据长度与指针所指向数据类型的长度一致。例如使用 u8 类型指针写入 CRCDR 寄存器，每次则写入 1 个 u8 类型数据。

使用示例如下：

```
uint32_t index;
uint32_t CRC_Result = 0;
uint32_t CRC_Buff[16] = {0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0xaa,0xbb,0xcc,0xdd,
                         0xee,0xff,0x00};
RCCAHB->AHB_CFG |= 0X02;                    //开启 CRC 时钟
RCCAHB->AHB_RST |= 0X02;
RCCAHB->AHB_RST &= ~ 0X02;                  //将 CRC 相关寄存器复位至缺省值
CRC->CRC_CON &= (uint32_t) ~(0X03<<6);      //32 位多项式
CRC->CRC_POL =  0x04C11DB7;                 //多项式默认值
CRC->CRC_INT =  0xFFFFFFFF;                 //CRC 默认初始值
CRC->CRC_CON |= 0X01;                       //将 CRCDR 复位为全 1
for(index = 0U; index < 16; index++)
{
   * ( (uint8_t *)CRC->CRC_DR) = CRC_Buff [index];          //将计算值填入 CRC
}
CRC_Result = CRC->CRC_DR;                   //获取 32 位 CRC 计算值
```

5.4 RCC 时钟系统注意事项

赛元 SC32F1XXX 系列 MCU 共有 5 个时钟源：HIRC、LIRC、HXT、LXT 和 PLL。上电默认时钟源是 HIRC, 内置的系统时钟监控模块可在系统时钟异常时将时钟源切换至 HIRC。

时钟总线分为 HCLK 和 PCLK0/1/2，其中 HCLK 默认为二分频，PCLK0/1/2 默认为一分频。

在切换系统时钟源时，需要注意：

1. 切换时钟前，相关时钟源（预选和新选）必须打开。
2. 切换任何系统时钟时，应先切换回 HIRC，再设定所需时钟。
3. 必须先配置系统时钟源选择位（SYSCLKSEL[1:0]），再配置系统时钟源切换位（SYSCLKSW）。最好不要两个寄存器位同时赋值，否则时钟源需要有一个先切换为 LIRC 后再切换为所选择时钟源的过程，如果此时 LIRC 不使能，则时钟源会切换失败。
4. 对系统时钟源切换位（SYSCLKSW）改写后，必须内部电路切换成功才会更新改写的值，否则读到的一直是改写前的状态，用户可以通过读取此位的方式判断时钟源是否已切换成功。

使用示例如下：

```
RCC->RCC_KEY = 0X40;                    //打开 RCC_CFG0 寄存器写操作功能，写入值大于等于 0X40
RCC->RCC_CFG0 = (uint32_t)0x00000040;   //系统时钟源切换回 HIRC
RCC->RCC_CFG0 |= 0X20;                  //使能 HXT
uint32_t tmpreg;
tmpreg = RCC->RCC_CFG0;
tmpreg &= (uint32_t)~(0X0380);          //将系统时钟源选择位（SYSCLKSEL 和 SYSCLKSW）清零
tmpreg |= (uint32_t) 0X0100;            //系统时钟源选择 HXT
RCC->RCC_CFG0 = tmpreg;
RCC->RCC_CFG0 |= 0X80;                  //系统时钟源切换为 HXT
```

5.5 PLL 时钟配置注意事项

系统运行时钟可通过 PLL 倍频至 64MHz，使用需要注意：

1. 需要确保经过 PLL 输入源预分频后频率为 2MHz；
2. PLL 倍频系数 N 的取值范围为 2 到 192，其他配置值不支持，倍频后要保证倍频的频率保持在 200Mhz~400Mhz；
3. PLLRDY 标志位置起代表 PLL 时钟就绪，此时 PLL 时钟才能作为系统时钟源；

5.6 INT 外部中断设置注意事项

赛元 SC32F1XXX 系列 MCU 在使用外部中断功能时，请将对应的 IO 口设置为输入带上拉模式！IO 口需要先设置，再设置相应的外部中断配置。反过来操作有可能会误产生一次边沿中断。

同组外部中断共用一个中断向量，用户需要在中断服务函数内检测边沿标志，判断中断的来源，再执行对应的操作。

使用示例如下：

```
GPIOC->PXCON &= (uint16_t) ~(0X0C00);     //将 INT10(PC10)和 INT11(PC11)设置为输入模式
GPIOC->PXPH |= 0X0C00;                    //打开 PC10 和 PC11 的上拉电阻
INT->INTF_CON |= 0x0C00;                  //使能 INT10、INT11 下降沿触发
INT->INT_SEL1 = (uint32_t) ((0X02<<8) | (0X02<<12));  //使能 PC10 和 PC11 的外部中断
INT->INTF_IE |= 0x0C00;                   //使能 INT10 和 INT11 的下降沿中断
NVIC_EnableIRQ(2);;                       //开总中断
void EXTI8_11_IRQHandler()
{
    if(INT->INTF_STS & 0x0400)    //判断 INT10 是否产生下降沿中断
    {
        RT_BIT(10);  //清标志
    }
    if(INT->INTF_STS & 0x0800)    //判断 INT11 是否产生下降沿中断
    {
        RT_BIT(11);  //清标志
```

```
    }
}
```

5.7 LEDPWM 使用注意事项

赛元 SC32F1XXX 系列 MCU 拥有 32 路 8 位常规 PWM，分为 PWMAn（n 取 0~31）。

注意：LEDPWM 的通道设置寄存器与 LCD/LED 的 SEG 口使能寄存器共用，因此一旦用户使用了 LEDPWM 资源，就不能再使能 LCD/LED，否则会导致 LEDPWM 周期输出异常！

当 LEDPWM 输出波形时，若需改变占空比，可通过改变 LEDPWM_DTn（n = 0~31）的值实现。但需要注意：更改 LEDPWM_DTn 的值，占空比不会立即改变，而是等待本周期结束，在下个周期改变。

5.8 PWM 设置及使用注意事项

赛元 SC32F1XXX 系列 MCU PWM0 在互补模式下，PWM0/PWM1, PWM2/PWM3, PWM4/PWM5 和 PWM6/PWM7 分为四组，通过 PWM0_DTn（n = 0~7）调节占空比。

赛元 SC32F1XXX 系列 MCU PWM0 提供故障检测功能。用户使能故障检测功能后，不能悬空 FLT 管脚，否则 PWM 输出异常！当故障发生时，PWM 停止输出，PWM 口处于高阻状态。故障检测模式分为立即模式和锁存模式。锁存模式下，故障信号满足失能条件后，硬件不会自动清除故障检测状态标志位，用户可通过对 PWM0_STS 寄存器的 FLTSTA 位置 1 进行软件清零。

注意：

在配置 PWM 时需要先配置好 PWM 周期和占空比后再打开 PWM 电源，且 PWM 周期和占空比不能为 0！

5.9 SPI 使用注意事项

赛元 SC32F1XXX 系列 MCU 具有两个独立的 SPI 通信口（SPI0~1），其中 SPI0 有 16 位 8 级 FIFO 缓存。

1. 为了避免 SPI 功能打开时误产生 CLK（此时管脚从 IO 切换到 SPI 功能），必须在打开 SPI 前根据 CLK 的空闲电平设置，先将 IO 口设置为对应的电平状态。如设置 CPOL = 0 空闲为低电平时，需要将 CLK 对应的 IO 设置输出 0 再打开 SPI(SPEN = 1)，反之亦然，否则会导致通讯错位。
2. 在 SPI 通讯时，程序中需要经常重置 SPI，以保证不会因为噪声等原因误接收 CLK 而导致一直通讯错误。
3. SPI 配合 DMA 使用时，需要确保触发 DMA 的标志/状态位为清零状态，否则标志/状态位不能正常置起会影响 DMA 触发，影响数据搬运。
4. 在使用带 FIFO 的 SPI0 时，使能 RXNEIE、RXIE、RXHIE 和 TXHIE 中断后，状态位 TXHIF、RXHIF、RXFIF、RXNEIF 不能通过软件清零，所以符合状态描述后，会不停进入中断影响正常通信。

5.10TWI 使用注意事项

赛元 SC32F1XXX 系列 MCU 具有两个独立的 TWI 通信口（TWI0~1），用户可根据需要将 TWI 设置为主机或者从机，TWI 最大通信速率为 1Mbps；TWI0 可以产生 DMA 请求，TWI1 不可以产生 DMA 请求。

1. 使用 TWI 功能时，建议用户在中断服务函数中通过状态标志位 STATE[2:0]查询通信状态；
2. 由于 TWI 作主机时没有中断标志判断 STOP，用户需在发送 STOP 信号前后加短暂延时防止通信出错。

5.11UART 使用注意事项

赛元 SC32F1XXX 系列的 MCU 具有四个功能一致的 UART（UART0~3），UART 是标准的全双工串行通信接口，方便用于同外部设备的 UART 接口通信，UART 模块有如下几种功能：

1. 有 3 种通信模式：模式 0（同步通信，UART 只能做主机），模式 1/3：异步串行通信；
2. 在模式 1/3 下可以唤醒 CPU 的深度睡眠功能；
3. 能够触发 DMA 功能。其中 UART0 和 UART1 的接收（RXIF 置起）和发送（TXIF 置起）均可产生 DMA 请求，而 UART2 和 UART3 不能产生 DMA 请求。
4. UART 发送和接收完成可产生中断并置起对应的标志位 TXIF 和 RXIF，中断标志需要软件清除，清除方式为“写 1 清零”；DMA 请求响应后，buffer 数据发送/读取后会自动清除对应 TXIF 和 RXIF 标志位，不需要手动清零。
5. 使用 UART 配合 DMA 进行发送，DMA 把最后一个数据写入 UARTn_DATA 寄存器后，DMA 传输完成中断标志位（TCIF）就会置起，但此时 UART 外设并没有把最后一个数据完全发送。UART 需要继续发

送数据，有以下解决办法：

a) 在 DMA 传输完成中断加个延时确保 UART 数据发送完毕；

b) 配置 DMA 时，把 UART-DMA 中断的优先级调最高。进入 DMA 传输完成中断后，先把 UART 发送中断标志位（TXIF）清除，然后把 UART 发送中断打开。进入 UART 发送中断则证明最后一个数据完全发送。

5.12 DMA 注意事项

直接存储器访问(DMA)控制器用于高速数据传输，使用需要注意：

1. 对于一个双向数据传输应用，需要 2 个 DMA 通道分别完成发送和接收；
2. 用 DMA 请求 DMA 时需要把作为请求源 DMA 的 CHRQ 位置起；
3. DMA 暂停后，需要先 PAUSE 位置 0，再使能 DMA，才能从 DMA 暂停恢复；

```
DMA0->DMA_CFG &= (~DMA_CFG_PAUSE);  //置 0PAUSE 位
DMA0->DMA_CFG |= DMA_CFG_CHEN;      //重新使能 DMA0
```

4. 在非循环模式下使用 DMA 通道，重新赋值该 DMA 的 DMACNT 后，需要把该 DMA 请求源所在外设的 DMA 重新失能后再使能，DMA 才能重新响应请求；

```
/* 重新写入 DMA 通道传输值 */
DMA0->DMA_CNT = (uint32_t)16;
/* 重新使能 SPI0 的 DMA 发送功能，才能开启新一轮通信 */
SPI0->SPI_IDE &= (uint16_t)~SPI_IDE_TXDMAEN;    //先失能 SPI0 的 DMA 发送功能
SPI0->SPI_IDE |= (uint16_t)SPI_IDE_TXDMAEN;     //再使能 SPI0 的 DMA 发送功能
```

5. 在非循环模式下使用到多个 DMA 通道时，当其中一个 DMA 搬移完成后，需要把该 DMA 关闭或重新赋值 DMACNT，否则优先级和该 DMA 一致或小于的 DMA 通道无法响应请求；

```
if((DMA0->DMA_STS & DMA_FLAG_TCIF) != 0)        //DMA0 搬运完成
{
   DMA0->DMA_STS = DMA_FLAG_TCIF;               //标志清零
   DMA0->DMA_CFG &= (~DMA_CFG_CHEN);    //失能 DMA0，避免影响其他 DMA 响应请求
}
```

5.13 SysTick 寄存器注意事项

SysTick 的时钟源不能高于系统时钟源，否则 SysTick 不生效。

5.14 省电模式设置注意事项

赛元 SC32F1XXX 系列芯片提供三种省电模式：低速模式、IDLE Mode 和 STOP Mode 模式。

1. 使用 WFI 或 WFE 都可以进入 IDLE Mode 或 STOP Mode。通过把修改 SCB->SCR 寄存器 SLEEPDEEP 位置 1，执行 WFI 或 WFE 就会进入 STOP Mode，否则进入 IDLE Mode。

```
/* 进入 STOP Mode */
SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;
__WFI();
```

2. 使用 WFE 时，需要先写 SEV，然后写入两次 WFE。使用示例如下：

```
//进入 IDLE Mode
   __SEV();
   __WFE();
   __WFE();
```

3. 从 STOP Mode 唤醒后，SYSCLKSEL[1:0]、SYSCLKSW 和 HIRCEN 都会恢复为保留值，系统时钟源为 HIRC。若进入 STOP Mode 前系统时钟源不是 HIRC，唤醒后需要重新配置时钟。

5.15 软件安全加密功能注意事项

赛元 SC32F1XXX 系列芯片允许用户选择是否开启安全加密功能，该功能需要使用 SC LINK PRO，并配合 SOC Programming Tool 上位机软件使用，具体的操作说明见《赛元 LINK 系列量产开发工具使用手册》4.4. 5 安全加密章节，可到赛元网站下载。

5.16 中断关闭注意事项

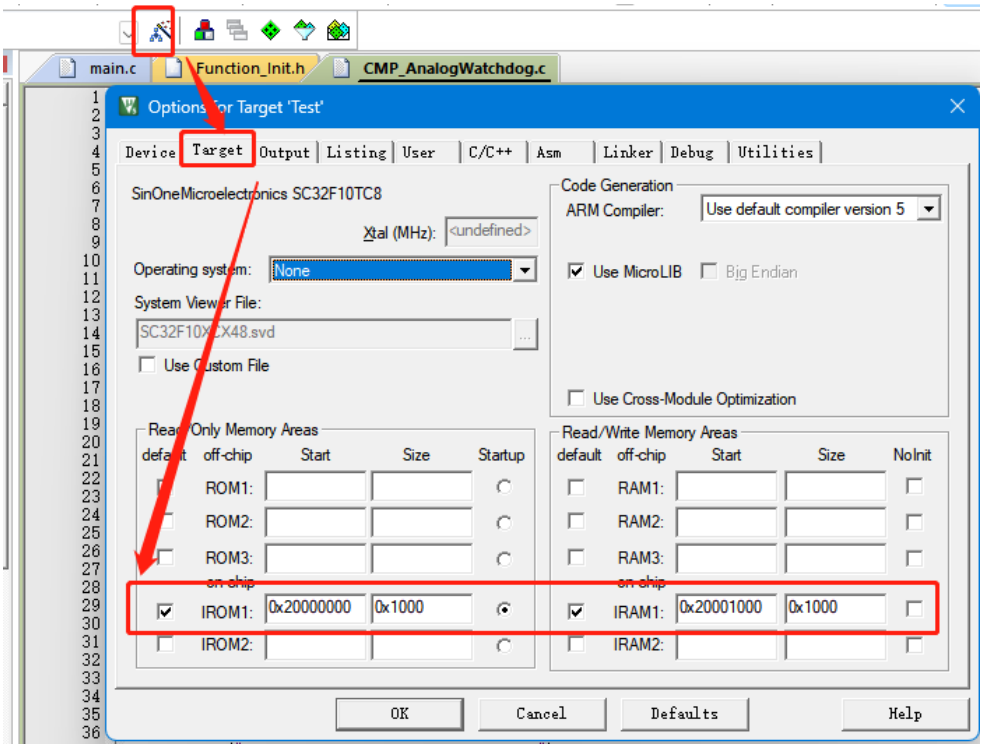
在实际应用中开启子中断后一般不需要再次关闭子中断，如果用户需要关闭子中断，必须在关闭子中断前先将总中断关闭，然后再关闭子中断，如下是子中断关闭步骤，请用户务必按照此步骤进行子中断的关闭：

```
__disable_irq();
NVIC_DisableIRQ(); //关闭子中断开关;
__enable_irq();
```

5.17 赛元 SC32F1XXX 系列在 SRAM 启动

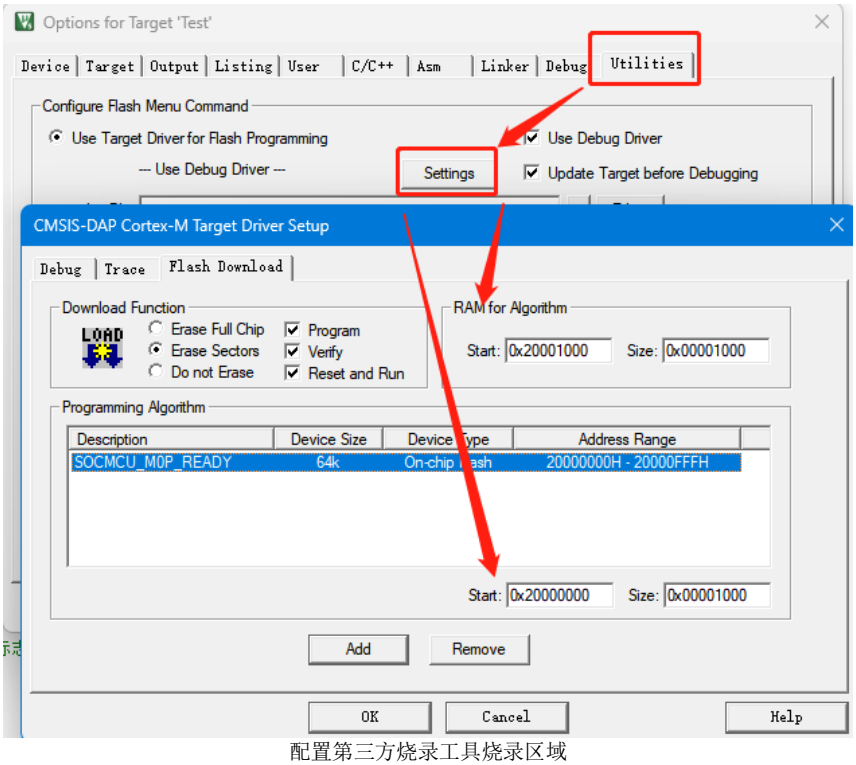
赛元 SC32F1XXX 系列芯片在 SRAM 启动并运行 SRAM 代码，需要完成以下步骤：

1. 在 SRAM 运行代码，需要把代码先存放在 SRAM 区域，赛元 SC32F1XXX 系列芯片 SRAM 的起始地址是 0x20000000。打开 Keil 工程的 Option 配置页面，在 Target 窗中可以需要两个区域的起始地址和区域大小。“Read Only Memory Areas”代表则只读内存区域，存放的是代码和常量；而“Read/Write Memory Areas”则是数据区域。配置 IROM1 区域的起始地址为 0x20000000，则编译生成代码则从 0x20000000 开始存放，也就是 SRAM 区域开始存放。区域大小可自由灵活划分，注意代码区域和数据区域不能有重叠，而且两者区域大小之和不能超过 SRAM 区域的大小。



代码区域为 0x20000000~0x20001000

2. 选择第三方烧录器烧录时，打开 Keil 工程的 Option 配置页面，在 Utilties 窗中打开“Settings”，把 RAM 和 ROM 区域配置成和第一步一致。



配置第三方烧录工具烧录区域

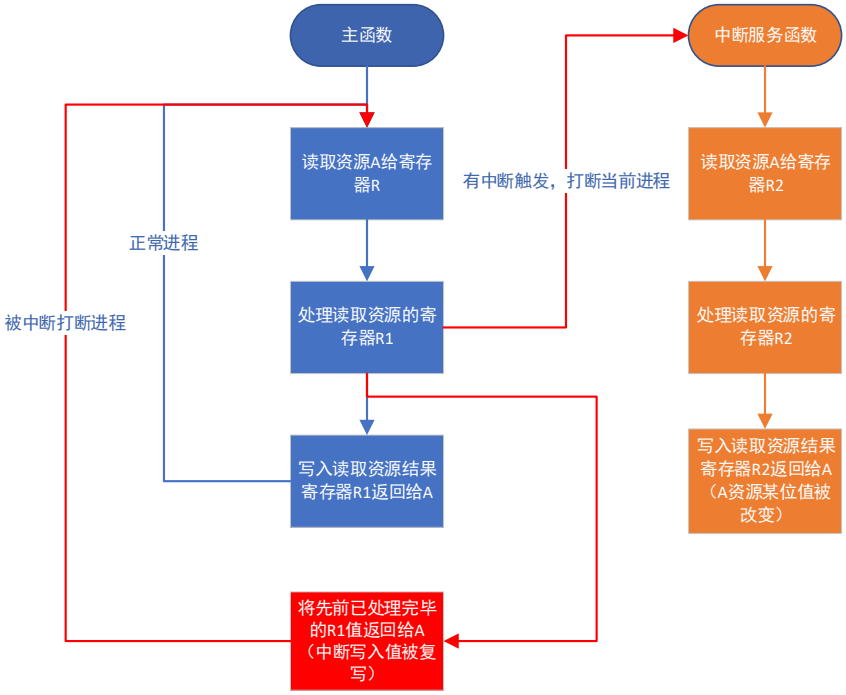
3. 编译烧录后，代码即可在 SRAM 中运行。代码运行中注意不能使能下电复位，否则需要重新烧录代码才能运行。

5.18临界区问题概述及解决方式

1. 临界区问题概述

临界区指的是访问多个任务共享资源的一段代码。当有任务进入临界区时，其它任务必须等待直至该任务离开临界区，以确定共享资源的访问不会冲突。

常见的临界区问题是由于 ARM 单片机寄存器的“&”操作赋值语句导致的，如在 main 主程序及中断服务函数中，分别对同一寄存器或全局变量，进行读写操作，比较容易出现临界区操作问题。问题的出现流程大致如下：



临界区操作问题出现的根本原因在于，存在两个及以上任务对同一外设资源或者变量进行读写操作，在一个任务对特定资源进行访问操作时，另一个任务可以打断该进程或者硬件同步对该资源进行读写操作，导致打断过

程返回时或者在要实时读取的场所，该资源写的效果被覆盖或者读取出错。

2. 具体问题及解决方法

使用赛元 SC32F1XXX 系列 MCU 时，应注意寄存器临界区操作的问题。以下是临界区操作问题，常用的外设场景及解决方法：

外设资源	出现场景	解决方法
GPIO	在主函数和中断中，对同一端口的不同 PIN 使用 BSP 函数接口或者直接操作端口寄存器，对整个端口进行写操作，会出现引脚状态复原的现象	使用位带操作语句：PX_BIT(n)、PX_OT(n)，X=A，B，C，n=0，1，..15，操作 GPIO
INT	启用多个外部中断时，在中断中使用 INT_ClearFlag 函数有可能会将另一个 INT 的标志位给清除	使用位带操作语句：FT_BIT(n)、RT_BIT(n)，n=0，1，..15，清除 INT 的标志位
TIM	使用捕获模式时，如果溢出中断和捕获中断同时触发，捕获值有可能是 0xFFFF 或者 0x0000，而溢出次数为 1， 导致读到的计数值可能比实际值多了 0xffff	在进入中断的第一时间，如果脉冲时间大于 TIM 溢出时间，中断中先关闭 TIM 计数，再处理捕获计数。（TIMER 捕获功能时，建议采样周期大于被测波形的周期。如时间要求特别严格，请自行加上补偿）
SPI	SPI 使用 FIFO 进行传输时，在中断中，如先处理 TXEIF，最后再处理 SPIF，那么当 TXEIF 处理时间过长时，有可能将会遗漏下一个数据的 SPIF	进入中断时，应先判断和处理 SPIF 标志，再处理 TXEIF 标志，不能在返回前才清 SPIF

5.19TOUCHKEY 设置注意事项

应用程序初始化时需要将 TK 对应的 IO 口设置为强推挽输出模式，输出高/低电平。在 TK 扫描过程中，不能去操作使用 TK 对应的 IO。另外，如果同一块 PCB 上有两颗以上的触摸芯片，为了防止同频干扰，建议开启“抗干扰设置”。

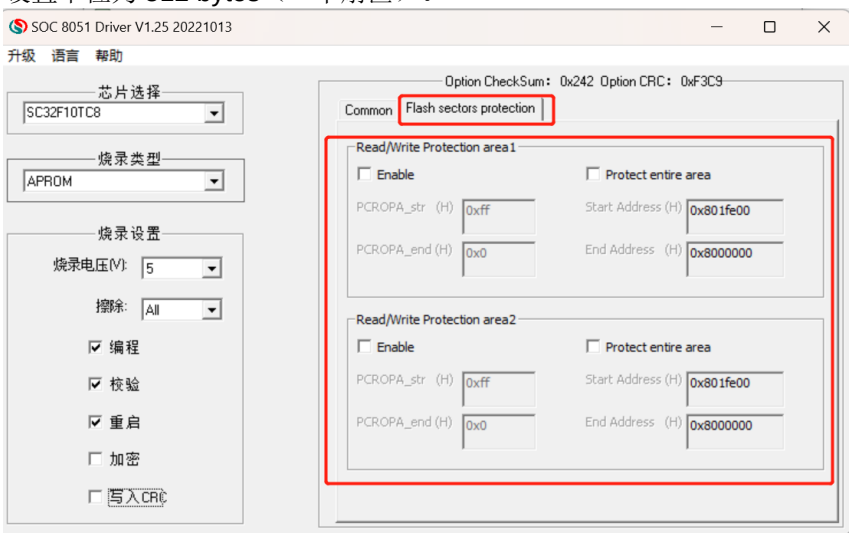
更多 TouchKey 的注意事项请参考《赛元 SC32F1XXX 系列 TouchKey MCU 应用指南》。

6 赛元 SC32F1XXX 系列 MCU 的 IAP 及算法解析

6.1 IAP 操作

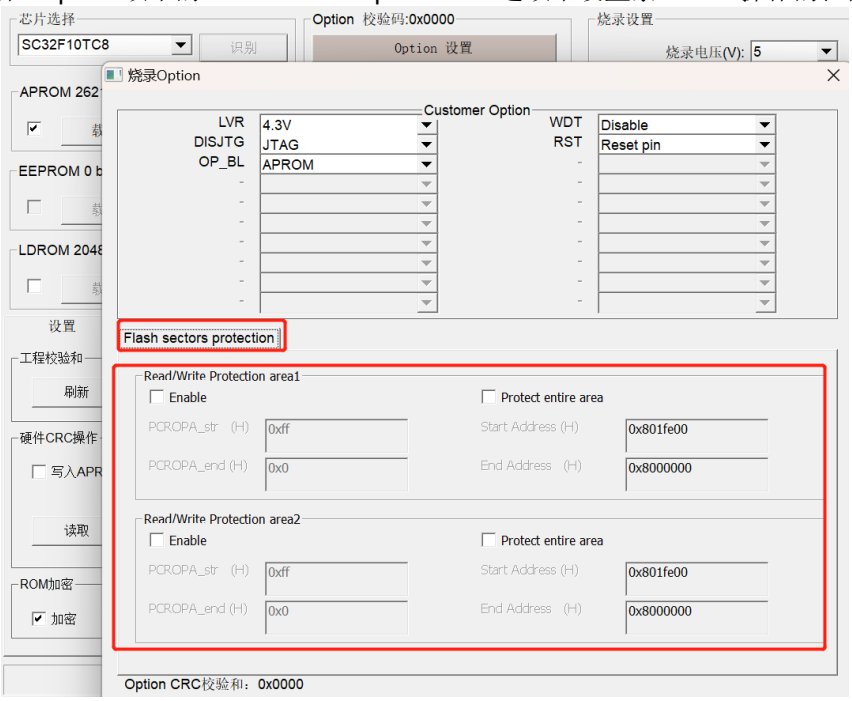
以 SC32F10TC8 为例，说明赛元 SC32F1XXX 系列 MCU CODE 区 IAP 操作的方法。

SC32F10TC8 内部 256Kbytes Flash 均可以进行 In Application Programming（IAP）操作，即允许用户程序动态的把数据写入内部的 Flash。用户使用 IAP 时，可通过 Code Option 设置 APROM 区域禁止 IAP 操作的范围，该范围分 2 段，最小设置单位为 512 bytes（一个扇区）。



KEIL IAP 操作范围配置界面

在上位机烧录时也是在 Option 项中的 Flash sectors protection 选项中设置禁止 IAP 操作的范围。如图。



烧录上位机 IAP 操作范围配置界面

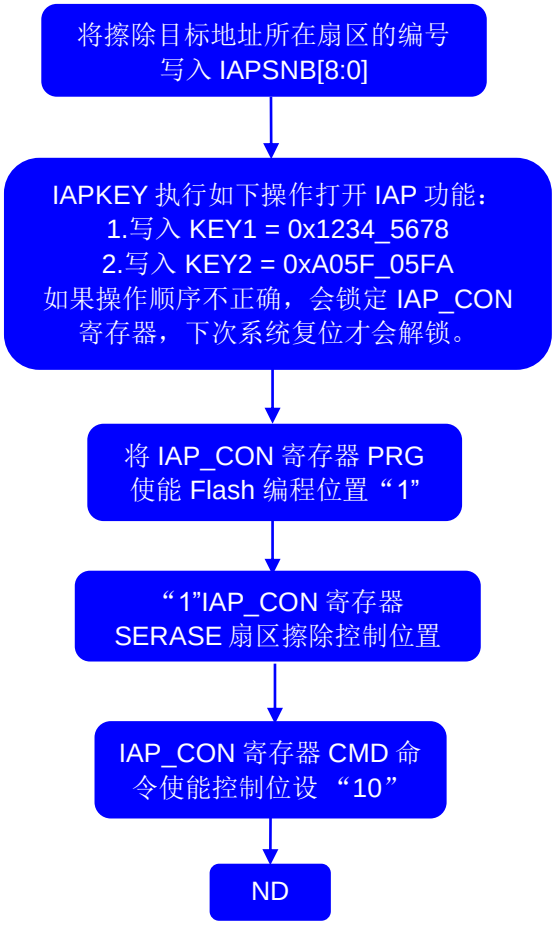
FLASH 读写特点：

1. 可进行单 Byte 读写操作；
2. Flash ROM 分为 512 个扇区（800 0000h ~ 803 FFFFh），写前需对操作地址所属扇区进行扇区擦除（扇区擦除：512bytes），擦除写入该扇区首地址即可；

FLASH 的寿命：10 万次以上

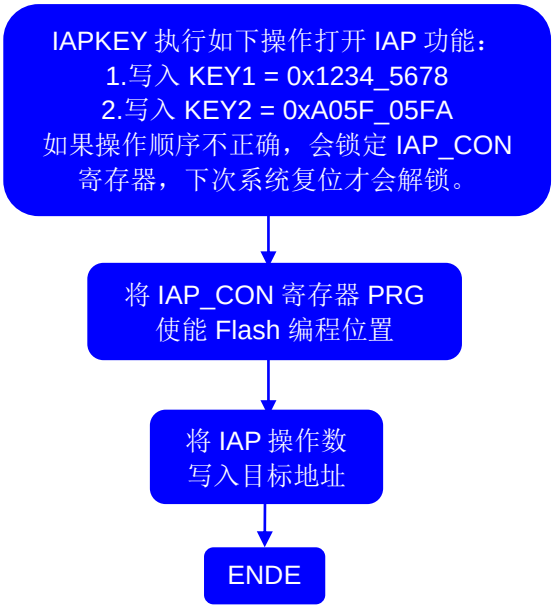
6.1.1 IAP 扇区擦除流程

为保证写入字节成功，写入前需要确保写入目标地址已被擦除为 0x00；赛元 SC32F1XXX 系列 MCU 支持扇区擦除，具体 IAP 扇区擦除流程如下：



6.1.2 IAP 写入流程

每写入一个字节，需要指定一个地址；具体 IAP 写入流程如下：



6.1.3 特别提醒

ROM 区域内的 IAP 操作有一定的风险，需要用户在软件中做相应的安全处理措施，如果操作不当可能会造成用户程序被改写！除非用户必需此功能(比如用于远程程序更新等)，否则不建议用户使用。

6.2 IAP 的使用建议及注意事项

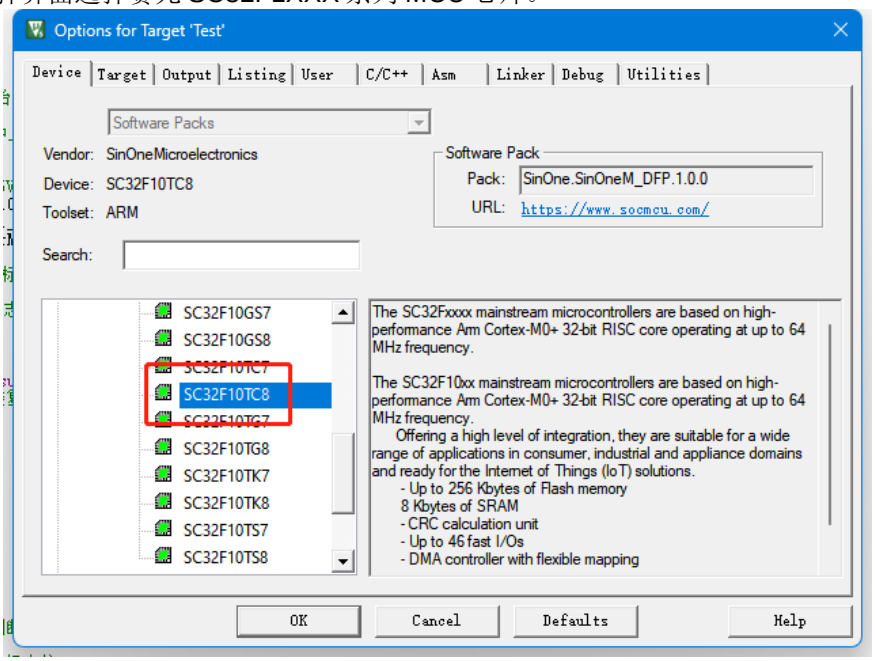
1. 赛元 SC32F1XXX 系列 MCU 写入数据前需要对目标地址所在扇区进行扇区擦除，建议用户在擦除前将数据备份到另一个扇区，防止擦除过程中掉电而发生旧数据被擦除而新数据未写入的意外情况。
2. IAP 写入需要遵守字节对齐。
3. SC32F1XXX 系列芯片可以进行 In Application Programming（IAP）操作，而实际产品的应用中，只是需要把仅几个 Bytes 的数据写到 Flash，采用固定地址写入数据会使某些地址过早达到 IAP 寿命次数，同时每次写数据前需进行数据备份和扇区擦除，因此建议用户可采用分扇区、循环地址写入的方法操作 Flash。
4. 执行 IAP 操作时会同步执行 WDT 清零，WDT 从零开始计数。
5. IAP 操作前，需使能 HIRC；若系统时钟源不为 HIRC，IAP 操作完成后，可以关闭 HIRC，以避免功率损耗。
6. IAP 扇擦前，需将写使能关闭。
7. IAP 擦除过程中 CPU 掉电不工作，若外设中断在擦除操作前，则程序会先响应外设中断在进行 IAP 的擦除操作。

7 第三方烧录器烧录

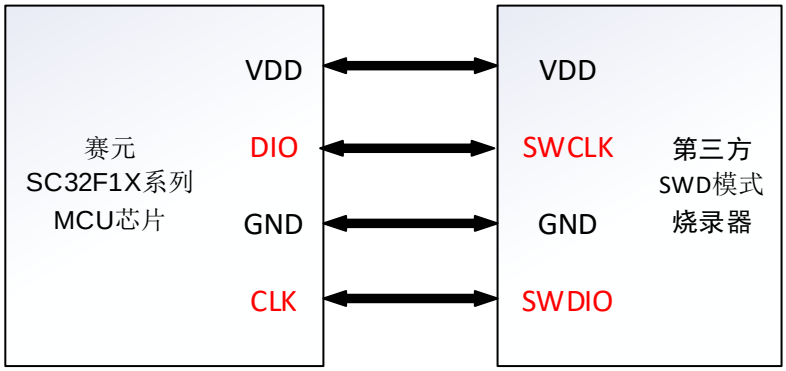
赛元 SC32F1XXX 系列 MCU 支持使用 JLink、DAP 等第三方烧录器进行代码烧录。

7.1 第三方烧录器烧录步骤

1. 在芯片型号选择界面选择赛元 SC32F1XXX 系列 MCU 芯片。

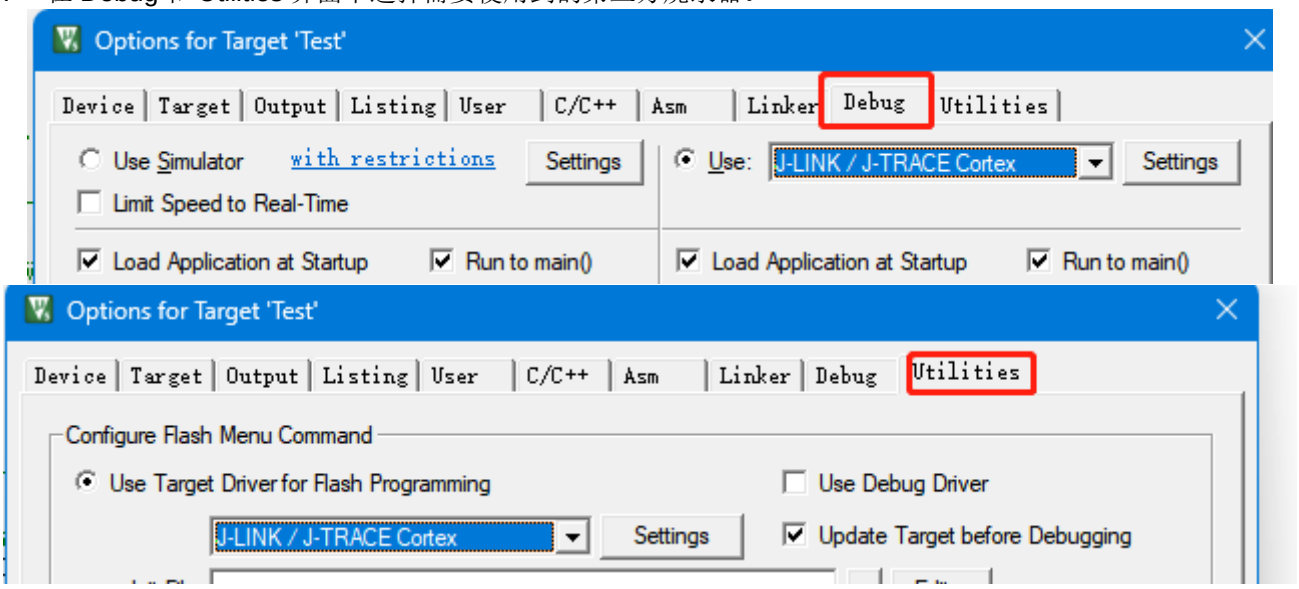


2. 连接赛元 SC32F1XXX 系列 MCU 芯片和第三方烧录器。
使用 SWD 模式接口烧录，芯片的 DIO 管脚连接到烧录器的 SWCLK 接口，而 CLK 管脚连接到烧录器的 SWDIO 接口，连接示意图如下：

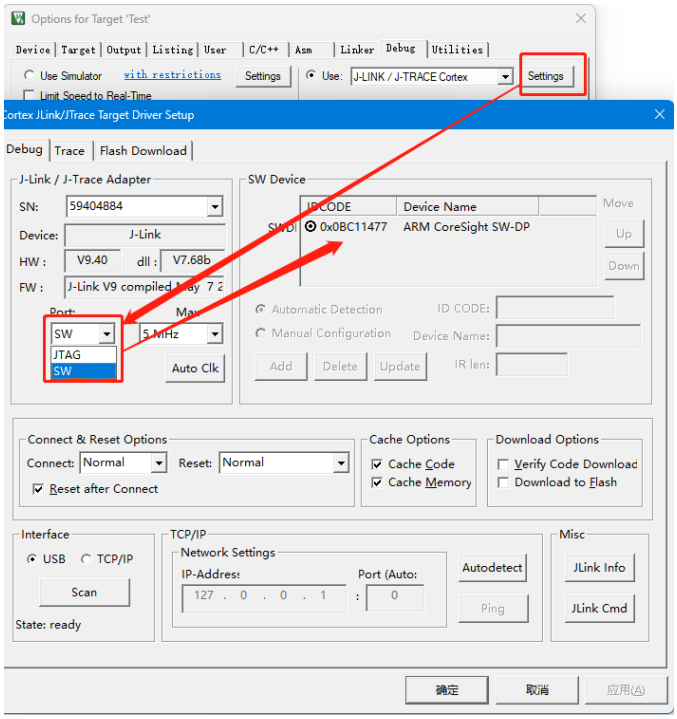


SWD 模式烧录连接图

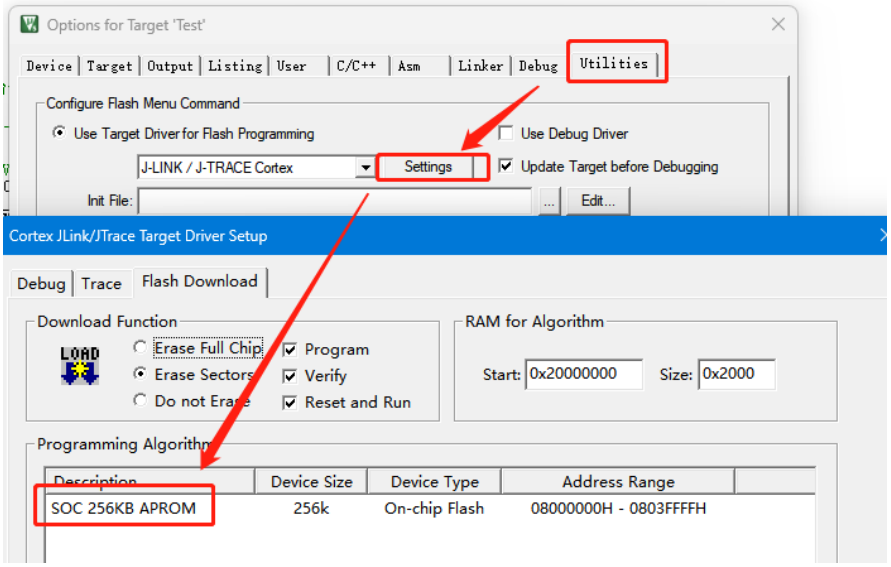
3. 在 Debug 和 Utilities 界面中选择需要使用到的第三方烧录器。



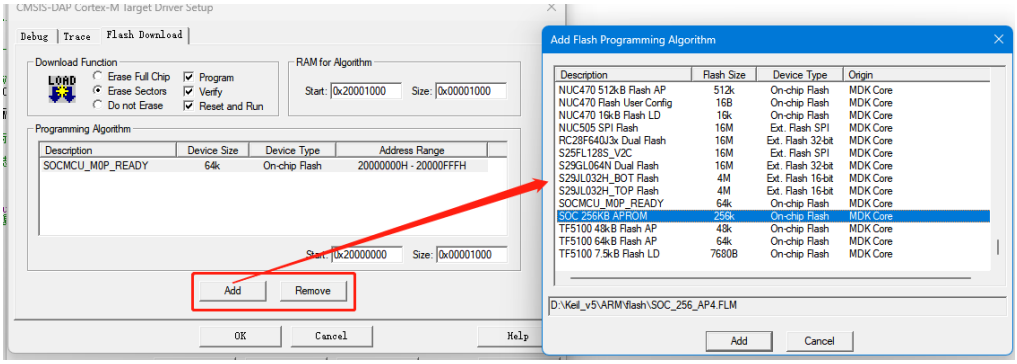
4. 打开 Debug 界面的 Settings 按键，在 Port 中选择 SW。在 SW Device 中显示到 IPCODE 和 Device Name，则证明第三方烧录器与芯片连接成功并成功识别。



5. 在 Utilties 窗中打开“Settings”，查看 Description 是否为赛元 SC32F1XXX 系列烧录算法。赛元算法描述的命名规则为“IC 的系列+ROM+对象区域”，如烧录 SC32F10TC8 的 ARPOM 区域，赛元算法需要选择为“SC32F10XX_256K_APROM”。



6. 如果烧录算法中没有选择正确，可以通过 Add/Remove 来进行添加/移除烧录算法。

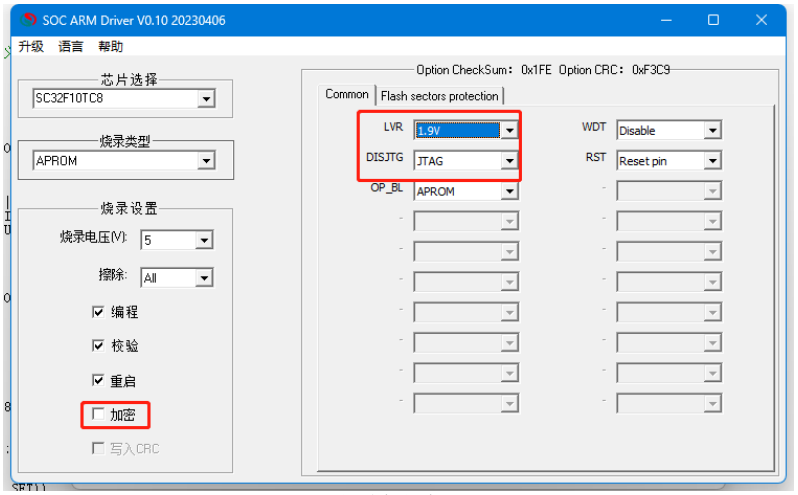


7. 程序编译通过后，即可使用第三方烧录器烧录程序。

7.2 使用第三方烧录器注意事项

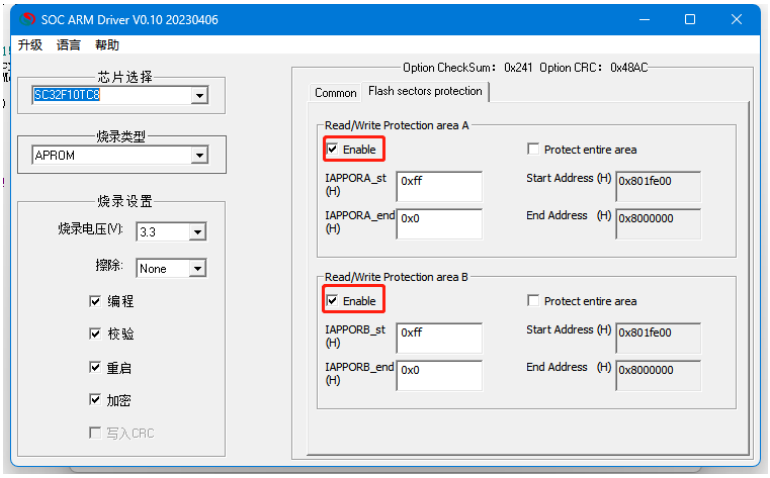
1. 使用 SWD 方式接口烧录，芯片的 DIO 管脚连接到烧录器的 SWCLK 接口，而 CLK 管脚连接到烧录器的 SWDIO 接口。
2. 在使用第三方烧录器烧录前，使用 SC LINK Pro 烧录芯片时需要注意：
   1) 不能把 option 中 DISJTG 配置为 Normal;
   2) LVR 复位电压需要小于第三方烧录器工具的供电电压;
   3) 不能选择加密烧录；

   否则经过 SC LINK Pro 烧录后，无法被第三方烧录器识别。



可用配置界面

3. 在使用第三方烧录器烧录前，使用 SC LINK Pro 烧录芯片时，不能使能代码写保护区域 A 或 B。否则经过 SC LINK Pro 烧录后，会提示擦除失败或编程失败。



4. 烧录 APROM 和 SRAM 区域的算法不一致，烧录前需要进行烧录区域确认。

8 规格更改记录

版本	记录	日期
V1.2	1.修正项 （1）修正 TWI 注意事项 2.优化项 （1）优化 INT 中断标志位清 0 方式为位操作 （2）芯片型号统一，从 SC32 修正为 SC32F1XXX 3.新增项 （1）新增 IAP 操作注意事项的条例	2023 年 12 月
V1.1	1.修正项 （1）修正 INT 外部中断注意事项 （2）修正 DMA 注意事项代码 2.优化项 （1）应用指南 TOUCHKEY 章节移到 5.19 后 3.新增项 （1）新增临界区问题概述及解决方式	2023 年 10 月
V1.0	初版	2023 年 4 月

9 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 2023 年 04 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。