Shanghai Fudan Microelectronics Group Company Limited

复旦微电子

FM33XX
低功耗系列 MCU
应用笔记

KEIL&IAR 程序在 RAM 执行

AN014 **V1.0**

目　　录

图目录

1 说明

本文档为 FM33XX 系列低功耗 MCU 的应用笔记，用于说明将程序代码放到 RAM 执行的方法。FM33XX 系列是复旦微电子公司开发的低功耗 MCU 芯片，请联系复旦微电子公司提供更多相关文档支持设计开发。

2 背景

为什么需要将代码放到 RAM 中执行？

因为 CPU 在 Flash 中取指时进行 Flash 擦写，则 CPU 取指将被暂停，直到擦写操作完成(针对 FM33LC0XX 系列，擦 Page 时间为 2ms，写一个 word 的时间 55us)。所以针对将 flash 当成 EE 来用于存储数据，系统中断时间小于 2ms 并且要求实时性很高的应用需要考虑。

解答方法：如果 CPU 跳转到 RAM 中取指运行，则 Flash 擦写不会暂停 CPU 的执行。Flash 擦写过程中，若用户希望在 RAM 中执行代码时仍然能够实时响应中断，应将中断向量表重新映射到 RAM 中。下面关于 KEIL 和 IAR 给出简单的配置流程。

3 KEIL

①RAM 中分配中断向量表空间

修改 SCT 文件，分配 RAM 空间保存中断向量表，如图 3-1 所示

PS：cortex-m0 中断入口一共 48 个，分配 48*4=192=0xC0 字节空间即可,比如分配的空间放到 RAM 起始位置，其他可用 RAM 起始地址为 0x200000C0

```
LR_IROM1 0x00000000 0x00040000  {    ; load region size_region
  ER_IROM1 0x00000000 0x00040000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
  }
  RW_VTOR 0x20000000 0xc0  {  ; RW data
  .ANY (+RW +ZI)
  }
  RW_IRAM1 0x200000c0 0x00006000-0xc0  {  ; RW data
   .ANY (+RW +ZI)
   .ANY (TORAM)
  }
}
```

图 3-1 分配 RAM 空间保存中断向量表

②中断向量表搬移以及重定向

如图 3-2 所示

```
void VTOR2RAM(void)
{
    __disable_irq();
    memcpy( (void*)0x20000000, (void*)0x00000000, 0xc0);
    SCB->VTOR = 0X20000000;
    __enable_irq();
}
```

图 3-2 中断向量表重定向

③将代码定位到 RAM 中

将在 Flash 擦写期间运行的程序(包括擦写函数)，用__attribute__((section("TORAM")))指令使其放到 RAM 中，如图 3-3 所示

```
LR_IROM1 0x00000000 0x00040000  {    ; load region size_region
  ER_IROM1 0x00000000 0x00040000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
  }
  RW_VTOR 0x20000000 0xc0  {  ; RW data
  .ANY (+RW +ZI)
  }
  RW_IRAM1 0x200000c0 0x00006000-0xc0  {  ; RW data
   .ANY (+RW +ZI)
   .ANY (TORAM)
  }
}
```

```
__attribute__((section("TORAM"))) void BSTIM_IRQHandler()
{
    time++;
    BSTIM32->ISR = 0X01;
    if(GPIOC->DO & 0X01)
        GPIOC->DRST =0X01;
    else
        GPIOC->DSET =0X01;

}
```

```
__attribute__((section("TORAM"))) void DataErase(uint32_t Space)
{

  RCC->PCLKCR2 |= BIT5;
  RCC->OPCCR2 |= BIT22;

  FLASH->EPCR = BIT0;
  FLASH->KEY = 0x96969696U;
  FLASH->KEY = 0xEAEAEAEAU;
  *((uint32_t *)Space) =  0x1234ABCDU;

  while(!(FLASH->ISR & BIT0));
  FLASH->ISR |= BIT0;
  FLASH->KEY = 0x00000000;

   RCC->OPCCR2  &= ~BIT22;
   RCC->PCLKCR2 &= ~BIT5;
}
```

图 3-3 单函数在 RAM 执行指令

Tip1：使用__attribute__((section(“name ”)))修饰的函数中调用到的所有函数也要放到section(“name ”)中，否则子函数如果在 FLASH 中取指时，CPU 也会暂停，直到擦写完成。

Tip2：如果很多的代码需要放到 RAM 中执行，可以只用以#pragma arm section code = “RUNRAM” 开头，以#pragma arm section 结尾。将所有需要放到 RUNRAM section 的函数包括进来。如图 3-4 所示

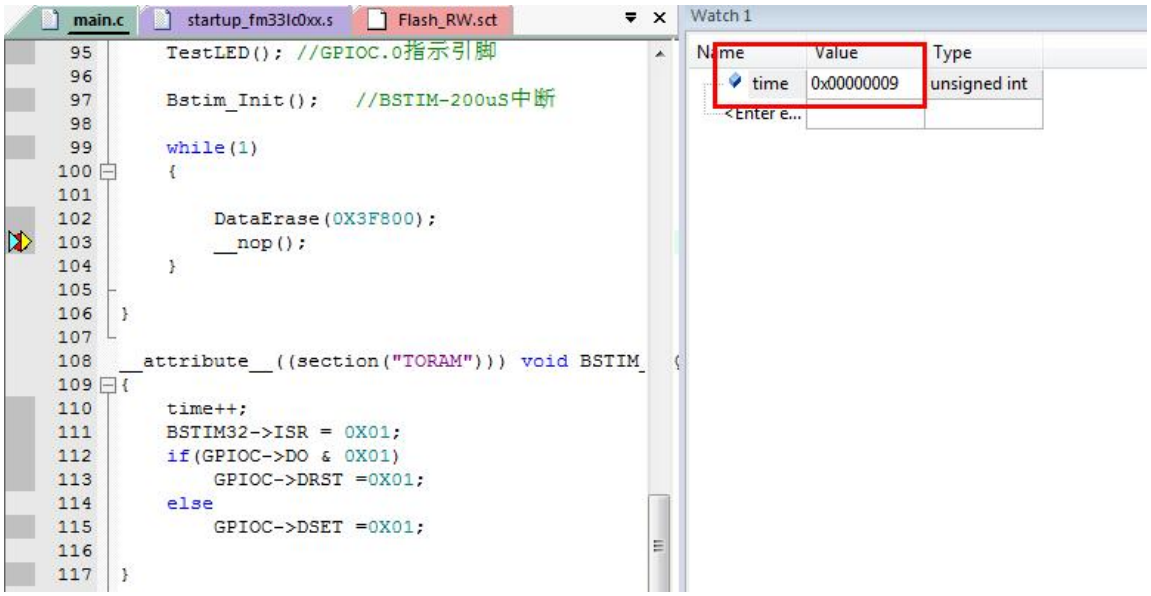
```
#pragma arm section code = "RUNRAM "
void BSTIM_IRQHandler()
{
    time++;
    BSTIM32->ISR = 0X01;
    if(GPIOC->DO & 0X01)
        GPIOC->DRST =0X01;
    else
        GPIOC->DSET =0X01;

}
#pragma arm section
```

图 3-4 多函数在 RAM 中执行指令

④测试结果

→测试条件：BSTIM-200us 中断，使用 PC0 进行翻转。当对 FLASH 进行擦流程时(擦时间 2ms)，通过观察进中断的次数以及翻 IO 的频率，来判断 CPU 是否被暂停。如图 3-5 所示



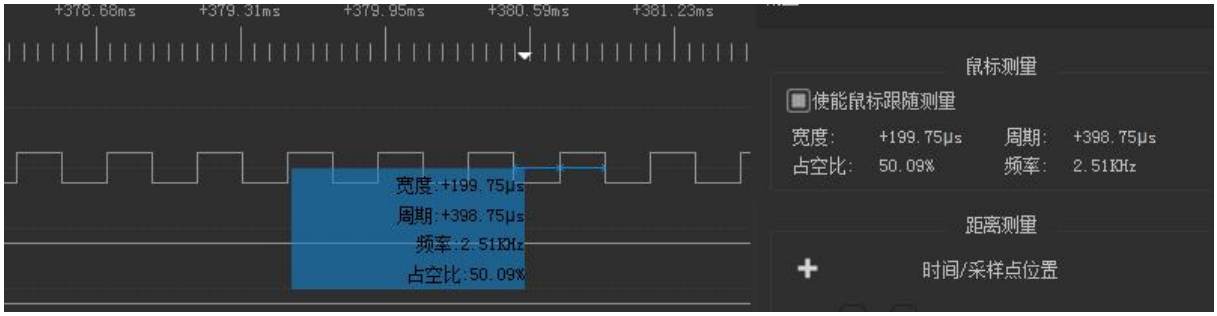


图 3-5 RAM 执行测试结果

4 IAR

①RAM 中分配中断向量表空间

修改 ICF 文件，分配 RAM 空间保存中断向量表，如图 4-1 所示

PS：cortex-m0 中断入口一共 48 个，分配 48*4=192=0xC0 字节空间即可,比如分配的空间放到 RAM 起始位置，其他可用 RAM 起始地址为 0x200000C0

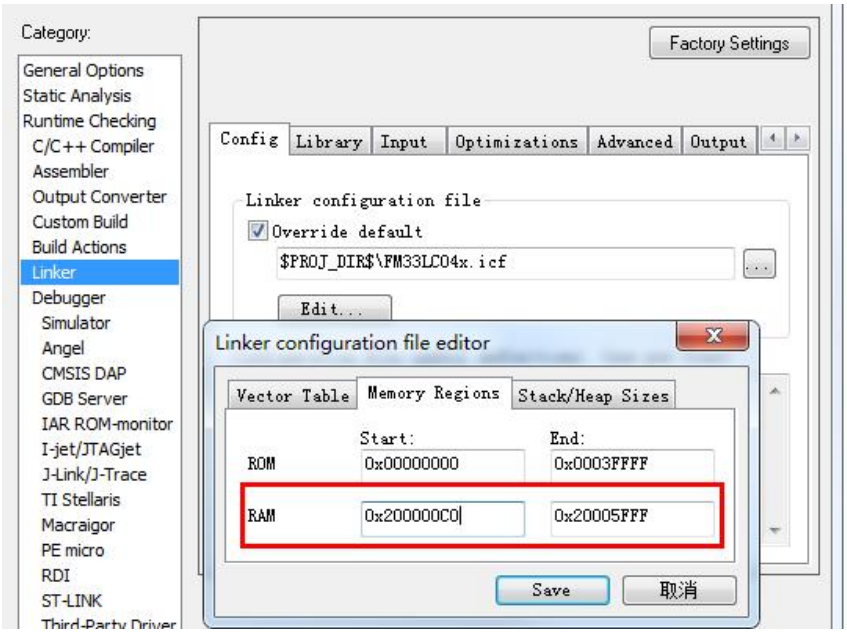


图 4-1 分配 RAM 空间保存中断向量表

②中断向量表搬移以及重定向

如图 4-2 所示

```
void VTOR2RAM(void)
{
    __disable_irq();
    memcpy( (void*)0x20000000, (void*)0x00000000, 0xc0);
    SCB->VTOR = 0X20000000;
    __enable_irq();
}
int main(void)
{

    VTOR2RAM();//中断向量表重定向到RAM
```

图 4-2 中断向量表重定向

③将代码定位到 RAM 中

将需要在 Flash 擦写期间运行的程序(包括擦写函数)，使用__ramfunc 指令使其放到 RAM 中，如图 4-3 所示

```
__ramfunc void DataErase(uint32_t Space)
{

  RCC->PCLKCR2 |= BIT5;
  RCC->OPCCR2 |= BIT22;

  FLASH->EPCR = BIT0;
  FLASH->KEY = 0x96969696U;
  FLASH->KEY = 0xEAEAEAEAU;
  *((uint32_t *)Space) =  0x1234ABCDU;

  while(!(FLASH->ISR & BIT0));
  FLASH->ISR |= BIT0;
  FLASH->KEY = 0x00000000;

   RCC->OPCCR2  &= ~BIT22;
   RCC->PCLKCR2 &= ~BIT5;
}
```

```
__ramfunc void BSTIM_IRQHandler()
{
    time++;
    BSTIM32->ISR = 0X01;
    if(GPIOC->DO & 0X01)
        GPIOC->DRST =0X01;
    else
        GPIOC->DSET =0X01;

}
```

图 4-3 单函数在 RAM 执行指令

Tip1：使用__ramfunc 修饰的函数中调用到的所有函数也要使用__ramfunc 定义，否则子函数如果在 FLASH 中取指时，CPU 也会暂停，直到擦写完成。

Tip2：如果很多的代码需要放到RAM中执行，使用#pragma default_function_attributes = @ " Section Name "的格式作为要指定函数的开始以及#pragma default_function_attributes =的格式作为要指定函数的结束。

④测试结果

→测试条件：BSTIM-200us 中断，使用 PC0 进行翻转。当对 FLASH 进行擦流程时(擦时间 2ms)，通过观察进中断的次数以及翻 IO 的频率，来判断 CPU 是否被暂停。如图 4-4 所示

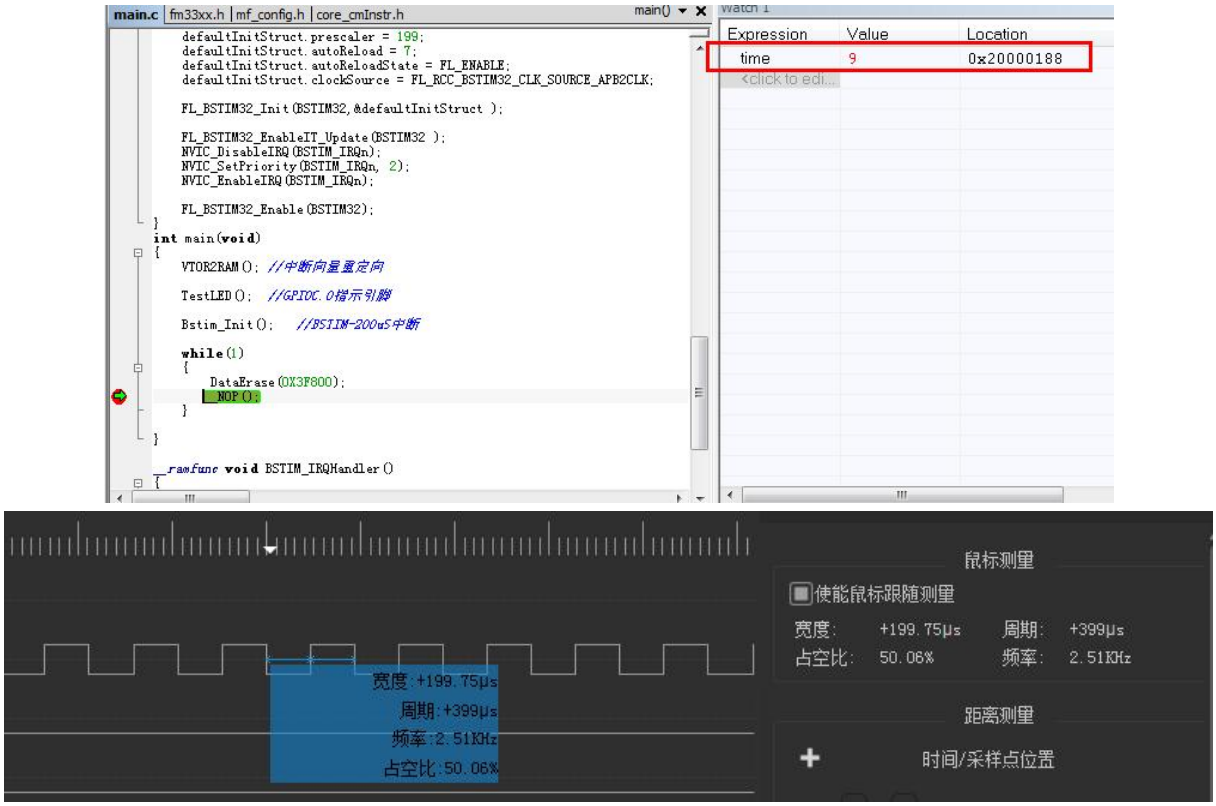


图 4-4 RAM 执行测试结果

版本信息

版本号	发布日期	更改说明
1.0	2021.11	首次发布

上海复旦微电子集团股份有限公司销售及服务网点

上海复旦微电子集团股份有限公司
地址：上海市国泰路 127 号 4 号楼
邮编：200433
电话：(86-021) 6565 5050
传真：(86-021) 6565 9115

上海复旦微电子（香港）股份有限公司
地址：香港九龙尖沙咀东嘉连威老道 98 号东海商业中心 5 楼 506 室
电话：(852) 2116 3288　2116 3338
传真：(852) 2116 0882

北京办事处
地址：北京市东城区东直门北小街青龙胡同 1 号歌华大厦 B 座 423 室
邮编：100007
电话：（86-10）8418 6608
传真：（86-10）8418 6211

深圳办事处
地址：深圳市华强北路 4002 号圣廷苑酒店世纪楼 1301 室
邮编：518028
电话：(86-0755) 8335 0911　8335 1011　8335 2011 8335 0611
传真：(86-0755) 8335 9011

台湾办事处
地址：台北市 114 内湖区内湖路一段 252 号 12 楼 1225 室
电话：(886-2) 7721 1889
传真：(886-2) 7722 3888

新加坡办事处
地址：237, Alexandra Road, #07-01, The Alexcier, Singapore 159929
电话：(65) 6472 3688
传真：(65) 6472 3669

北美办事处
地址：2490 W. Ray Road Suite#2 Chandler, AZ 85224 USA
电话：(480) 857-6500 ext 18
公司网址：http://www.fmsh.com/