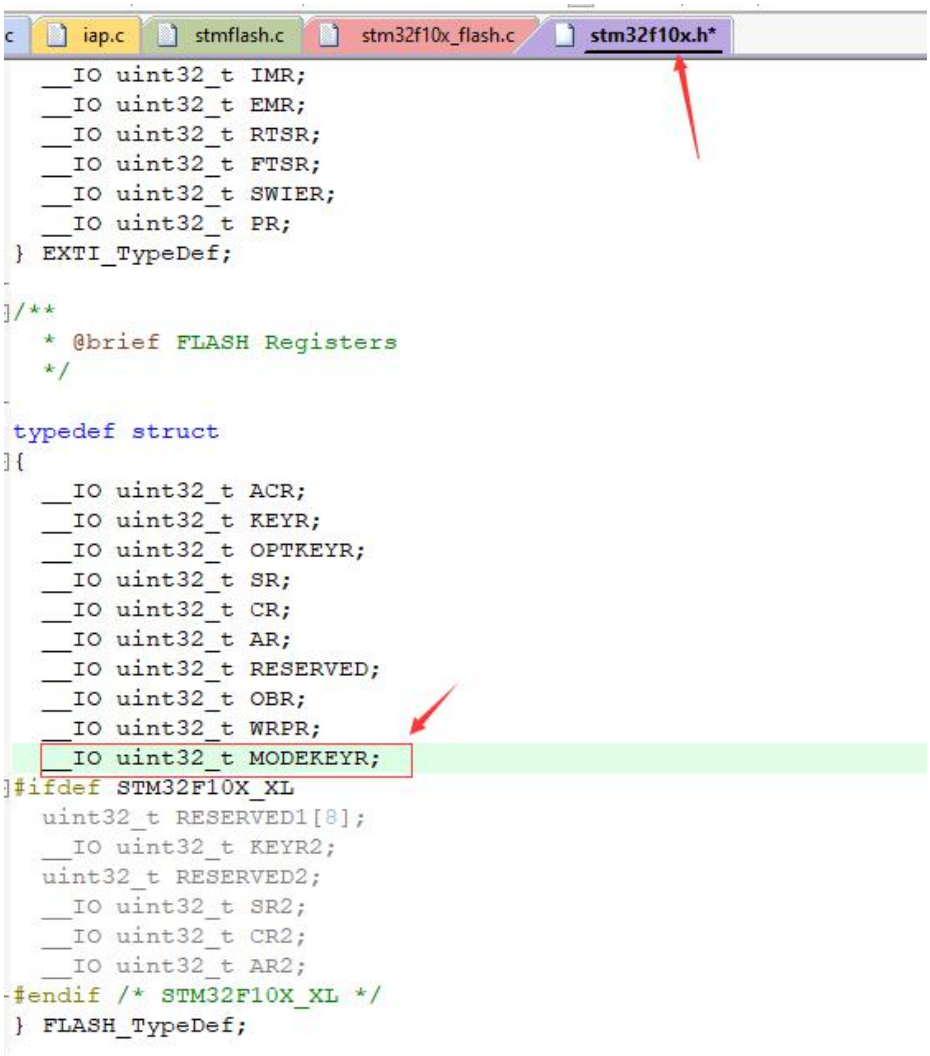
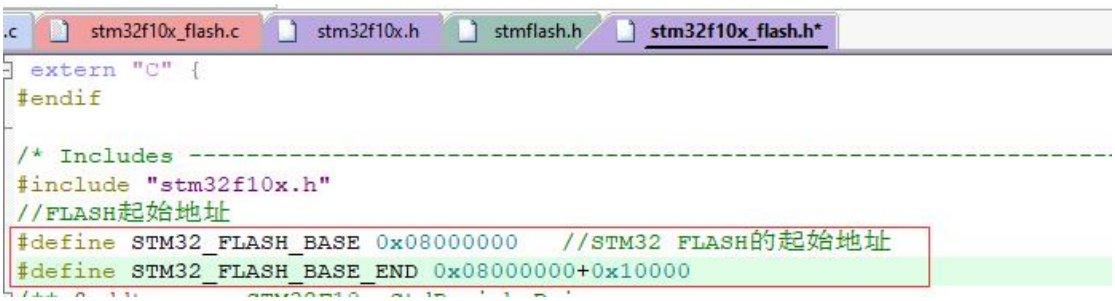
因 CH32 相对于 STM32 flash 操作多了快速编程模式，该文档说明主要目的是为了方便客户在原先 ST 工程的基础上实现 flash 快速编程模式的快速移植。

1、 在 ch32f10x.h 文件中在 FLASH_TypeDef 结构体中增加 MODEKEYR 成员定义，如图所示：



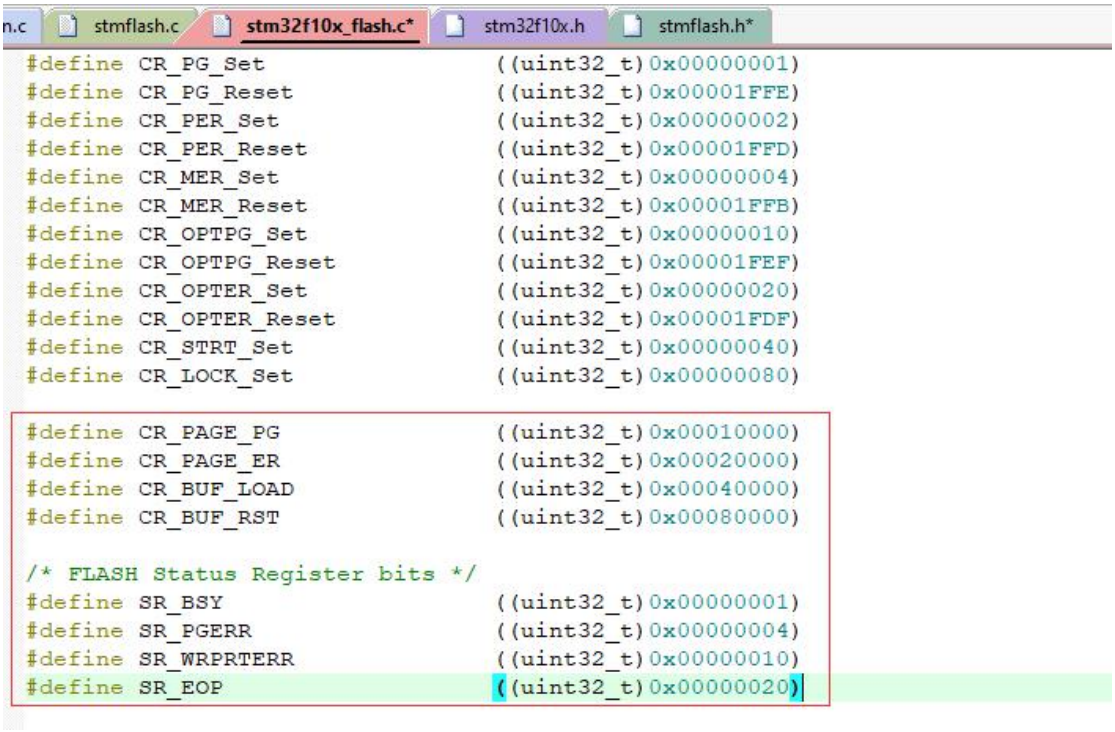
2、 修改 FLASH_Unlock 函数，增加 MODEKEYR 成员的操作，如图所示：

3、 目前 CH32F103 的 FLASH 最大为 64K，在 stm32f10x_flash.h 中增加 flash 容量的定义，如图所示：



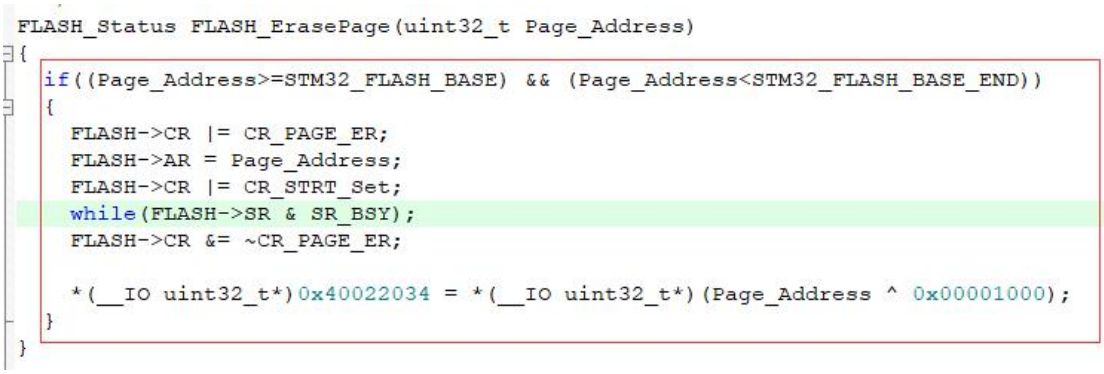
（大容量芯片可以相应修改“0x10000”数据即可）

4、 在 stm32f10x_flash.c 中增加 flash 寄存器快速编程位定义，如图所示：




```
#define CR_PAGE_PG                ((uint32_t)0x00010000)
#define CR_PAGE_ER                ((uint32_t)0x00020000)
#define CR_BUF_LOAD               ((uint32_t)0x00040000)
#define CR_BUF_RST                ((uint32_t)0x00080000)
/* FLASH Status Register bits */
#define SR_BSY                     ((uint32_t)0x00000001)
#define SR_PGERR                   ((uint32_t)0x00000004)
#define SR_WRPRTERR                  ((uint32_t)0x00000010)
#define SR_EOP                       ((uint32_t)0x00000020)
```

5、 修改 FLASH_ErasePage()函数，如图所示：



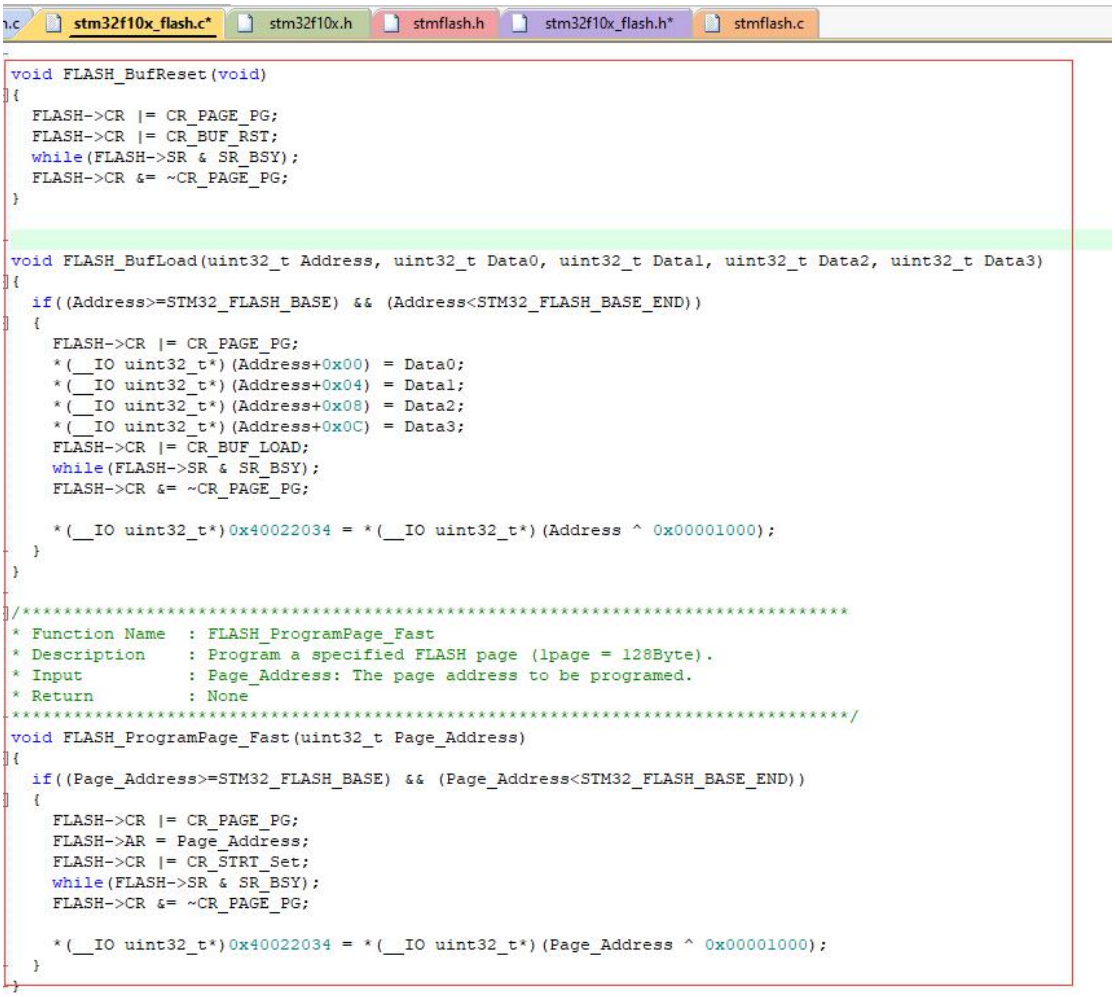
```
FLASH_Status FLASH_ErasePage(uint32_t Page_Address)
{
  if((Page_Address>=STM32_FLASH_BASE) && (Page_Address<STM32_FLASH_BASE_END))
  {
    FLASH->CR |= CR_PAGE_ER;
    FLASH->AR = Page_Address;
    FLASH->CR |= CR_STRT_Set;
    while(FLASH->SR & SR_BSY);
    FLASH->CR &= ~CR_PAGE_ER;

    *(__IO uint32_t*)0x40022034 = *(__IO uint32_t*)(Page_Address ^ 0x00001000);
  }
}
```

```
FLASH_Status FLASH_ErasePage(uint32_t Page_Address)
{
    if((Page_Address>=STM32_FLASH_BASE) && (Page_Address<STM32_FLASH_BASE_END))
    {
        FLASH->CR |= CR_PAGE_ER;
        FLASH->AR = Page_Address;
        FLASH->CR |= CR_STRT_Set;
        while(FLASH->SR & SR_BSY);
        FLASH->CR &= ~CR_PAGE_ER;

        *(__IO uint32_t*)0x40022034 = *(__IO uint32_t*)(Page_Address ^ 0x00001000);
    }
}
```

6、 在 stm32f10x_flash.c 中增加 FLASH_BufReset()函数、FLASH_BufLoad()函数以及快速编程函数 FLASH_ProgramPage_Fast()，如图所示：




```
void FLASH_BufReset(void)
{
  FLASH->CR |= CR_PAGE_PG;
  FLASH->CR |= CR_BUF_RST;
  while(FLASH->SR & SR_BSY);
  FLASH->CR &= ~CR_PAGE_PG;
}

void FLASH_BufLoad(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)
{
  if((Address>=STM32_FLASH_BASE) && (Address<STM32_FLASH_BASE_END))
  {
    FLASH->CR |= CR_PAGE_PG;
    *(__IO uint32_t*)(Address+0x00) = Data0;
    *(__IO uint32_t*)(Address+0x04) = Data1;
    *(__IO uint32_t*)(Address+0x08) = Data2;
    *(__IO uint32_t*)(Address+0x0C) = Data3;
    FLASH->CR |= CR_BUF_LOAD;
    while(FLASH->SR & SR_BSY);
    FLASH->CR &= ~CR_PAGE_PG;

    *(__IO uint32_t*)0x40022034 = *(__IO uint32_t*)(Address ^ 0x00001000);
  }
}

/*******************************************************************************
* Function Name  : FLASH_ProgramPage_Fast
* Description    : Program a specified FLASH page (1page = 128Byte).
* Input          : Page_Address: The page address to be programed.
* Return         : None
*******************************************************************************/
void FLASH_ProgramPage_Fast(uint32_t Page_Address)
{
  if((Page_Address>=STM32_FLASH_BASE) && (Page_Address<STM32_FLASH_BASE_END))
  {
    FLASH->CR |= CR_PAGE_PG;
    FLASH->AR = Page_Address;
    FLASH->CR |= CR_STRT_Set;
    while(FLASH->SR & SR_BSY);
    FLASH->CR &= ~CR_PAGE_PG;

    *(__IO uint32_t*)0x40022034 = *(__IO uint32_t*)(Page_Address ^ 0x00001000);
  }
}
```

（stm32f10x_flash.h 增加相应函数声明）

```
    void FLASH_BufReset(void)
    {
     FLASH->CR |= CR_PAGE_PG;
     FLASH->CR |= CR_BUF_RST;
     while(FLASH->SR & SR_BSY);
     FLASH->CR &= ~CR_PAGE_PG;
}
    void FLASH_BufLoad(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2,
uint32_t Data3)
    {
     if((Address>=STM32_FLASH_BASE) && (Address<STM32_FLASH_BASE_END))
     {
          FLASH->CR |= CR_PAGE_PG;
          *(__IO uint32_t*)(Address+0x00) = Data0;
          *(__IO uint32_t*)(Address+0x04) = Data1;
          *(__IO uint32_t*)(Address+0x08) = Data2;
          *(__IO uint32_t*)(Address+0x0C) = Data3;
          FLASH->CR |= CR_BUF_LOAD;
          while(FLASH->SR & SR_BSY);
```

```
        FLASH->CR &= ~CR_PAGE_PG;

        *(__IO uint32_t*)0x40022034 = *(__IO uint32_t*)(Address ^ 0x00001000);
     }
}
    void FLASH_ProgramPage_Fast(uint32_t Page_Address)
    {
     if((Page_Address>=STM32_FLASH_BASE)                                    &&
    (Page_Address<STM32_FLASH_BASE_END))
     {
        FLASH->CR |= CR_PAGE_PG;
        FLASH->AR = Page_Address;
        FLASH->CR |= CR_STRT_Set;
        while(FLASH->SR & SR_BSY);
        FLASH->CR &= ~CR_PAGE_PG;

        *(__IO uint32_t*)0x40022034 = *(__IO uint32_t*)(Page_Address ^ 0x00001000);

        }
      }
```

7、 演示函数如图所示

```c
void Flash_Test_Fast(void)
{
  u8 i, Verity_Flag=0;
  u32 buf[32];

  for(i=0; i<32; i++)
  {
    buf[i] = i;
  }

  FLASH_Unlock();

  FLASH_ErasePage(0x0800E000);

//  printf("128Byte Page Erase Sucess\r\n");

  FLASH_BufReset();
  FLASH_BufLoad(0x0800E000, buf[0], buf[1], buf[2], buf[3]);
  FLASH_BufLoad(0x0800E000 + 0x10, buf[4], buf[5], buf[6], buf[7]);
  FLASH_BufLoad(0x0800E000 + 0x20, buf[8], buf[9], buf[10], buf[11]);
  FLASH_BufLoad(0x0800E000 + 0x30, buf[12], buf[13], buf[14], buf[15]);
  FLASH_BufLoad(0x0800E000 + 0x40, buf[16], buf[17], buf[18], buf[19]);
  FLASH_BufLoad(0x0800E000 + 0x50, buf[20], buf[21], buf[22], buf[23]);
  FLASH_BufLoad(0x0800E000 + 0x60, buf[24], buf[25], buf[26], buf[27]);
  FLASH_BufLoad(0x0800E000 + 0x70, buf[28], buf[29], buf[30], buf[31]);
  FLASH_ProgramPage_Fast(0x0800E000);

//  printf("128Byte Page Program Sucess\r\n");

  FLASH_Lock();

  for(i=0; i<32; i++)
  {
    if(buf[i] == *(u32*)(0x0800E000 + 4*i)){
      Verity_Flag = 0;
    }
    else
    {
      Verity_Flag = 1;
      break;
    }
  }

  if(Verity_Flag) printf("128Byte Page Verity Error\r\n");
  else printf("128Byte Page Verity Sucess\r\n");
}
```

: