STM32 擦除内部 FLASH 时间过长导致 IWDG 复位

1 前言

客户反馈在使用 STM32F412 的时候，擦除 sector 8~11 发现时间过长，从而导致意外触发 IWDG 复位。

2 问题分析

2.1 问题详情

通过与客户邮件和电话沟通，了解到客户主要是想使用内部 FLASH 暂时保存 IAP 升级时的程序数据，在 IAP 升级的过程中，需要首先擦除内部 FLASH 中一块足够大的空间，然后再写入升级数据。客户的工程中有使用到 IWDG，喂狗间隔大约 1.5S，客户的通过 SysTick 的方式计算出擦除 Sector8 大约需要 2ms,因此认为若一次擦除 sector8~11 大约需要 8ms，于是在代码中一次性擦除 sector8~11 后最后再来喂狗，但是，这样会触发 IWDG 复位，这个与预期不一致，固此产生疑问。

2.2 问题重现

使用 NUCLEO-F412ZG 板尝试重现客户问题，主要代码如下：

```
int main(void)
{

  /* USER CODE BEGIN 1 */
  uint32_t beginTick =0,endTick =0;
  uint32_t curSysTick=0,endSysTick =0;
  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_IWDG_Init();

  /* USER CODE BEGIN 2 */
  if (__HAL_RCC_GET_FLAG(RCC_FLAG_IWDGRST) != RESET) //如果是看门狗复位
  {
    /* Clear reset flags */
```

```
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_7,GPIO_PIN_SET);
     __HAL_RCC_CLEAR_RESET_FLAGS();
    Error_Handler();
  }

      HAL_FLASH_Unlock();
  /* Fill EraseInit structure*/
  EraseInitStruct.TypeErase     = FLASH_TYPEERASE_SECTORS;
  EraseInitStruct.VoltageRange  = FLASH_VOLTAGE_RANGE_3;
  EraseInitStruct.Sector        = FLASH_SECTOR_8;
  EraseInitStruct.NbSectors     = 1;

//    if(HAL_FLASHEx_Erase(&EraseInitStruct, &SECTORError) != HAL_OK)
//  {
//          Error_Handler();
//  }

  beginTick =HAL_GetTick();
  HAL_GPIO_WritePin(GPIOC,GPIO_PIN_8,GPIO_PIN_SET);
  curSysTick =SysTick->VAL;
  if(HAL_FLASHEx_Erase_IT(&EraseInitStruct)!= HAL_OK)     //擦除 sector8
  {
      Error_Handler();
  }
  endSysTick =SysTick->VAL; // curSysTick, endSysTick 保存着 SysTick 寄存器的值
  HAL_GPIO_WritePin(GPIOC,GPIO_PIN_8,GPIO_PIN_RESET); //PC8 波形表示擦除 FLASH 的时间间隔
  endTick =HAL_GetTick();       // beginTick, endTick 保存着全局变量 Tick 的值
  g_TickCount =endTick -beginTick; //变量 Tick 的时间差
  HAL_IWDG_Refresh(&hiwdg);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
    if (HAL_IWDG_Refresh(&hiwdg) != HAL_OK)
    {
      /* Refresh Error */
      Error_Handler();
    }
    HAL_Delay(10);
  }
  /* USER CODE END 3 */
}
```

此外，同时在每个 SysTick 中断输出一个波形，用来检测 SysTick 是否正常：

```
void HAL_SYSTICK_Callback(void)
{
	HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_11);//用 PA11 来检测 SysTick 波形
}
```

最终得出的波形如下:

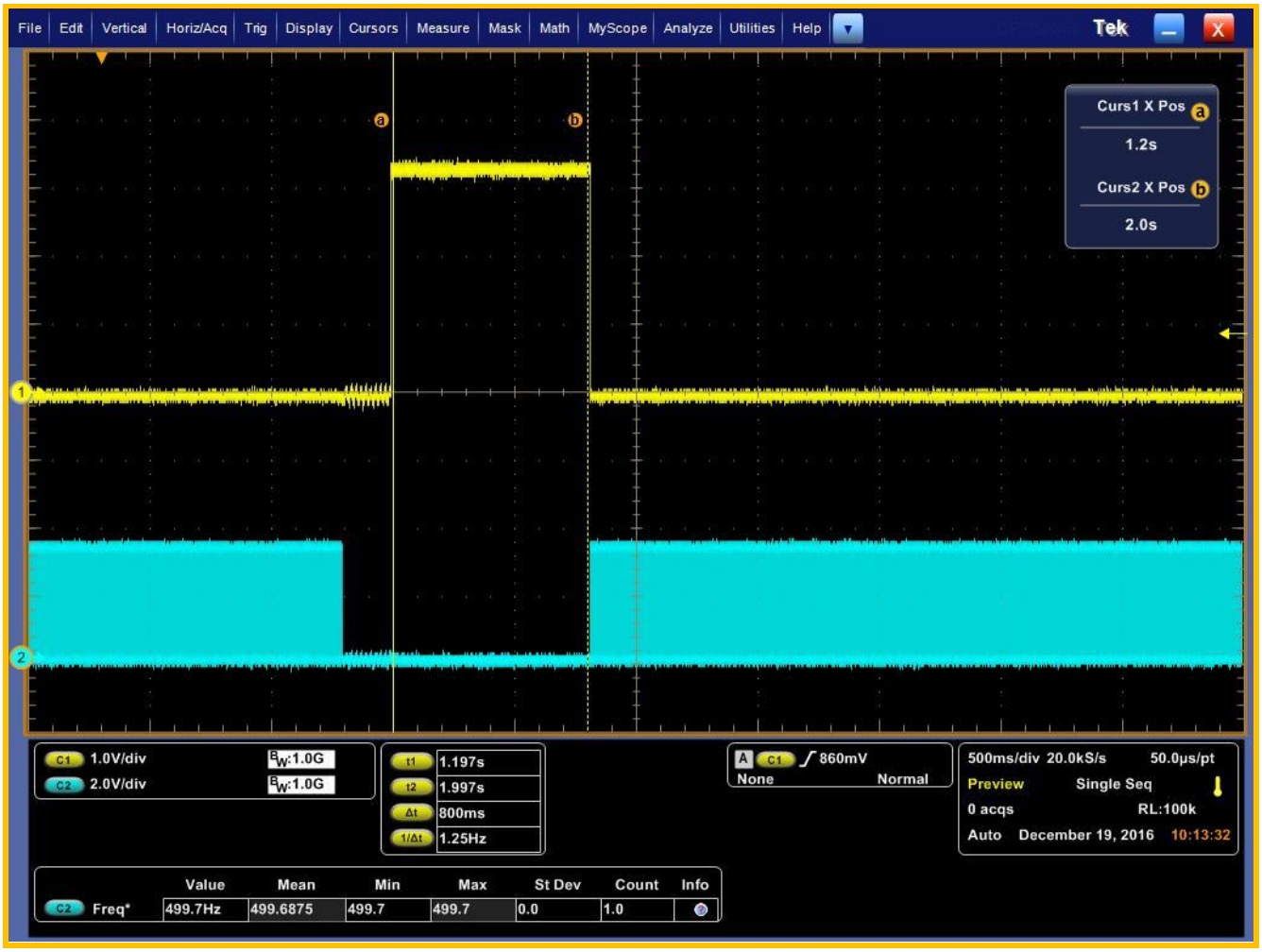

Figure 1 擦除 FLASH 时间与 SysTick 输出波形

如上图，黄色为 PC8 脚波形，表示擦除 FLASH 的时间,下面蓝色为 PA11 管脚波形，表示 SysTick 波形。

从上图可以看出擦除 sector8 所需要的时间是 800ms,这个与客户认为的 2ms 是不一致的。查看 STM32F412 的数据手册，在第 6.3.12 节中可以看到如下信息:

Table 48. Flash memory programming

Symbol	Parameter	Conditions	Min(1)	Typ	Max(1)	Unit
t_{prog}	Word programming time	Program/erase parallelism (PSIZE) = x 8/16/32	-	16	100(2)	µs
$t_{ERASE16KB}$	Sector (16 KB) erase time	Program/erase parallelism (PSIZE) = x 8	-	400	800	ms
		Program/erase parallelism (PSIZE) = x 16	-	300	600	
		Program/erase parallelism (PSIZE) = x 32	-	250	500	
$t_{ERASE64KB}$	Sector (64 KB) erase time	Program/erase parallelism (PSIZE) = x 8	-	1200	2400	ms
		Program/erase parallelism (PSIZE) = x 16	-	700	1400	
		Program/erase parallelism (PSIZE) = x 32	-	550	1100	
$t_{ERASE128KB}$	Sector (128 KB) erase time	Program/erase parallelism (PSIZE) = x 8	-	2	4	s
		Program/erase parallelism (PSIZE) = x 16	-	1.3	2.6	
		Program/erase parallelism (PSIZE) = x 32	-	1	2	
t_{ME}	Mass erase time	Program/erase parallelism (PSIZE) = x 8	-	16	32	s
		Program/erase parallelism (PSIZE) = x 16	-	11	22	
		Program/erase parallelism (PSIZE) = x 32	-	8	16	
V_{prog}	Programming voltage	32-bit program operation	2.7	-	3.6	V
		16-bit program operation	2.1	-	3.6	V
		8-bit program operation	1.7	-	3.6	V

1. Guaranteed by characterization, not tested in production.
2. The maximum programming time is measured after 100K erase operations.

Figure 2 擦除 FLASH 扇区所需要的时间

如上图，在 PSIZE=32 时，擦除一个 128K 的扇区需要大概 1S(典型值)的时间，而我们从图 1 中实际测出的为 800ms，这个基本相差不大，单与客户认为的 2ms 相去甚远，基本上我们认为这里的 800ms 是正确的结果，但是这个又是什么原因导致客户通过 SysTick 测出的值是错误的呢？

实际上，从图 1 我们也可以看出，在擦除 FLASH 的期间，SysTick 是没有波形的(见图 1 下面蓝色波形)，同时在参考手册 3.5 节中有如下信息：

3.5 Erase and program operations

For any Flash memory program operation (erase or program), the CPU clock frequency (HCLK) must be at least 1 MHz. The contents of the Flash memory are not guaranteed if a device reset occurs during a Flash memory operation.

Any attempt to read the Flash memory on STM32F4xx while it is being written or erased, causes the bus to stall. Read operations are processed correctly once the program operation has completed. This means that code or data fetches cannot be performed while a write/erase operation is ongoing.

Figure 3 参考手册中关于擦除扇区的描述

这句话的意思是说，在擦除 FLASH 的期间，若尝试读取 FLASH，则会被暂停，实际这个"读取"是指取指，我们都知道，程序的执行首先得通过从 FLASH 中通过 I-BUS 取出指令后才可以执行。这里 SysTick 之所以会被暂停掉，就是因为在擦除 FLASH 期间，为了执行 SysTick 中断例程，内核会尝试从 FLASH 取指，从而导致被暂停掉，进而全局变量 uwTick 的值没有机会增加。下图是调试界面：

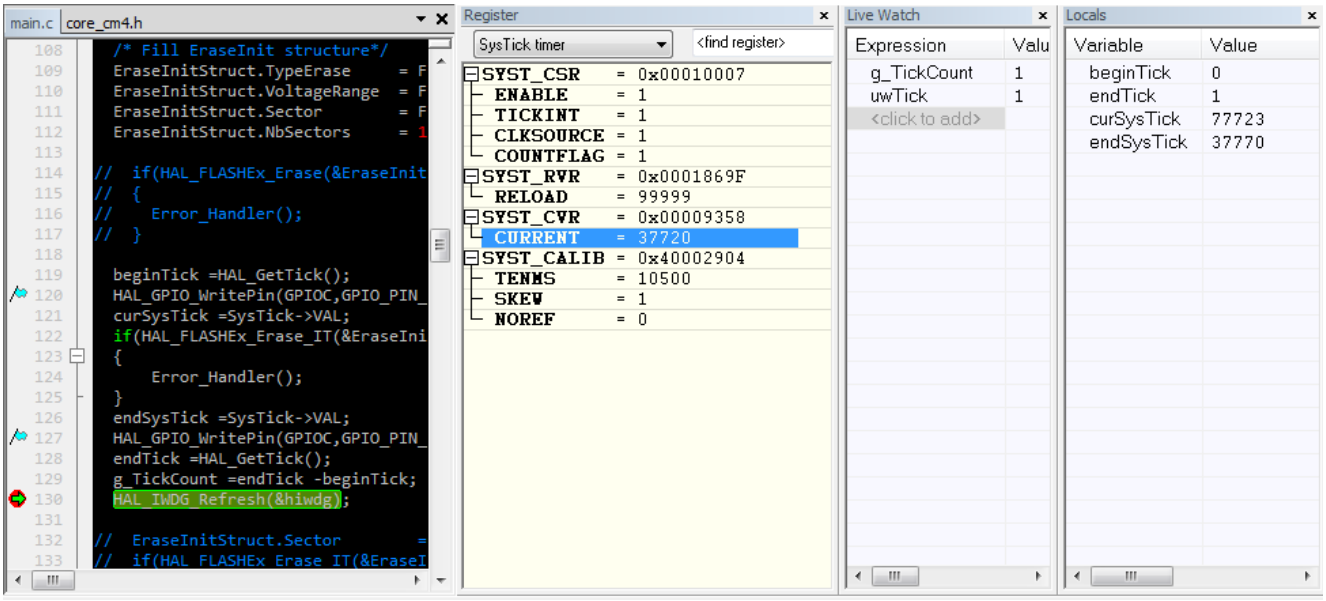

Figure 4 调试

如上图，在执行擦除扇区后，SysTick 的全局变量 uwTick 就增加了 1，但 SysTick 在内核中的寄存器还是有变化的。这个与我们的预想一致。

最后客户通过每擦除一个扇区喂一次狗的方式解决了问题，而在此期间不能依靠 SysTick 的值来计算时间。

3 结论

- 在擦除 FLASH 期间，取指操作会被暂停掉，且 SysTick 所对应的全局变量 uwTick 值是不会增加的。

- 另外，通过函数 HAL_FLASHEx_Erase_IT()来执行擦除 FLASH 和通过函数 HAL_FLASHEx_Erase()所花费时间没有差别，只不过前者在擦除完成后会产生一个中断，而后者没有。
- 可以通过外设 RTC 来计算擦除 FLASH 的时间，从而绕开限制。