FAQ0048

常见问题解答

擦除页大小不同的内部FLASH注意事项

Questions：擦除页大小不同的内部 FLASH 有何注意事项？

Answer：

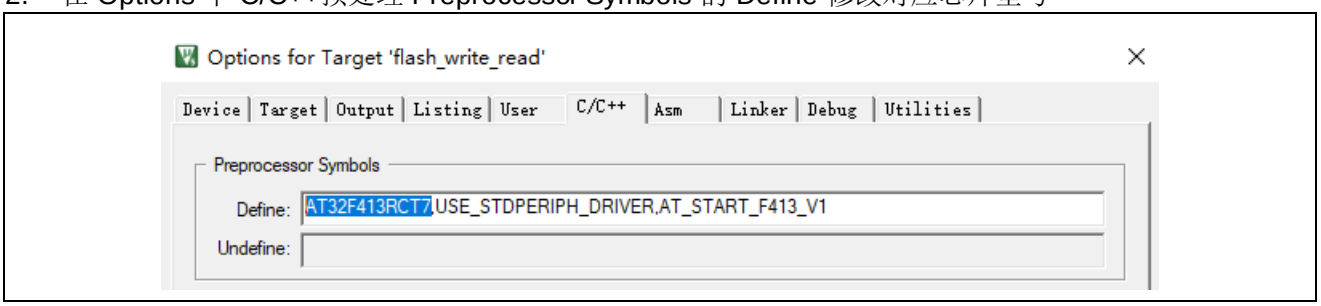
AT32F4 全系列内部 FLASH 大小为 256K 及以上的闪存扇区大小是 2K，内部 FLASH 大小小于 256K 的闪存扇区大小是 1K，在擦除时有所区别：

1. 擦除闪存扇区步骤：解锁闪存->擦除闪存扇区->锁定闪存
2. 擦除闪存扇区起始地址（Page_Address）及页内的任何地址都可以将该页擦除
3. 擦除位于 512K 之内的闪存扇区，是操作寄存器 FLASH->CTRL/FLASH->ADDR；擦除位于 512K 及以外的闪存扇区，是操作寄存器 FLASH->CTRL2/FLASH->ADDR2
4. 将闪存扇区大小是 1K 的擦除代码移植到闪存扇区大小是 2K 时，若连续擦除多个闪存扇区，需要将起始地址递增长度改为 2K（0x800）。如果起始地址递增长度还是 1K（0x400），那么一个 2K 的闪存扇区将会被擦除两次。例如，在 IAP 升级 APP 时，一般操作是擦除 1 页，写 1 页。如果页大小 2K 还是按页大小 1K 的擦除代码操作，比如擦除起始地址是 0x08001000 的闪存扇区，第一次会将该页擦除，写入 1K 数据后，第二次擦除长度只增加 1K，写入的起始地址是 0x08001400，那么还是会将该页擦除，将上一次写入的 1K 数据擦除
5. 将闪存扇区大小是 2K 的擦除代码移植到闪存扇区大小是 1K 时，若连续擦除多个闪存扇区，需要将起始地址递增长度改为 1K（0x400）。如果起始地址递增长度还是 2K（0x800），那么每隔一页将会有一页被漏擦除。例如，在 IAP 升级 APP 时，一般操作是擦除 1 页，写 1 页。如果闪存扇区大小 1K 还是按闪存扇区大小 2K 的擦除代码操作，比如擦除起始地址是 0x08001000 的页，将该页 1K 空间擦除，但写入 2K 数据，那么起始地址是 0x08001400 的页在没有被擦除就写入数据，会导致写入的数据错误

由于上述原因，在使用 AT32F413/AT32F415 系列 FLASH 小于 256K 的型号时，需要特别注意 BSP 程序基于该系列最大资源设计（FLASH 大小 256K），直接编译下载可能会造成下载失败或不全片擦除无法二次下载等问题。

此时处理方法如下：

1. 参考 BSP document 路径下《AT32F4xx 固件库 BSP&Pack 应用指南》在 Options 下 Device 修改对应 AT32 芯片型号及 FLASH 算法文件
2. 在 Options 下 C/C++预处理 Preprocessor Symbols 的 Define 修改对应芯片型号



3. 如果使用的是 FLASH 读写例程，还需要注意在 flash.c/flash.h 文件中修改对应 SECTOR_SIZE 大小

```
#define SECTOR_SIZE 2048
```

擦除程序示例：

```
flash_unlock();
flash_sector_erase(sector_address);
flash_lock();
```

其中 flash_sector_erase();函数中针对位于 512K 以内的闪存扇区擦除代码如下：

```
#define FLASH_BANK1_START_ADDR          ((uint32_t)0x08000000) /*!< flash start address of bank1 */
#define FLASH_BANK1_END_ADDR            ((uint32_t)0x0807FFFF) /*!< flash end address of bank1 */
if((sector_address >= FLASH_BANK1_START_ADDR) && (sector_address <= FLASH_BANK1_END_ADDR))
{
  /* wait for last operation to be completed */
  status = flash_bank1_operation_wait_for(ERASE_TIMEOUT);

  if(status == FLASH_OPERATE_DONE)
  {
    /* if the previous operation is completed, continue to erase the sector */
    FLASH->ctrl_bit.secers = TRUE;
    FLASH->addr = sector_address;
    FLASH->ctrl_bit.erstr = TRUE;

    /* wait for operation to be completed */
    status = flash_bank1_operation_wait_for(ERASE_TIMEOUT);

    /* disable the secers bit */
    FLASH->ctrl_bit.secers = FALSE;
  }
}
```

针对位于 512K 及以外的闪存扇区擦除代码如下：

```
#define FLASH_BANK2_START_ADDR          ((uint32_t)0x08080000) /*!< flash start address of bank2 */
#define FLASH_BANK2_END_ADDR            ((uint32_t)0x080FFFFF) /*!< flash end address of bank2 */
if((sector_address >= FLASH_BANK2_START_ADDR) && (sector_address <= FLASH_BANK2_END_ADDR))
{
  /* wait for last operation to be completed */
```

```
  status = flash_bank2_operation_wait_for(ERASE_TIMEOUT);

  if(status == FLASH_OPERATE_DONE)
  {
    /* if the previous operation is completed, continue to erase the sector */
    FLASH->ctrl2_bit.secers = TRUE;
    FLASH->addr2 = sector_address;
    FLASH->ctrl2_bit.erstr = TRUE;

    /* wait for operation to be completed */
    status = flash_bank2_operation_wait_for(ERASE_TIMEOUT);

    /* disable the secers bit */
    FLASH->ctrl2_bit.secers = FALSE;
  }
}
```

类型： MCU 应用
适用型号： AT32 全系列
主功能： FLASH
次功能： 无

文档版本历史

日期	版本	变更
2022.2.16	2.0.0	最初版本

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途(及其依据任何司法管辖区的法律的对应情况)，或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：(A) 对安全性有特别要求的应用，如：生命支持、主动植入设备或对产品功能安全有要求的系统；(B) 航空应用；(C) 汽车应用或汽车环境；(D) 航天应用或航天环境，且/或(E) 武器。因雅特力产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险由购买者单独承担，并且独力负责在此类相关使用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。