RTL8762C Flash User Guide

V1.1

2018/09/12

修订历史（Revision History）

日期	版本	修改	作者	Reviewer
2018/06/11	V1.0	发布第一版	Grace	
2018/09/12	V1.1	修改第 2 节	Grace	Serval

目录

图目录

表目录

1 概述

Flash 作为一种非易失性存储器，与 RAM 不同。RAM 可以直接读写，直接执行代码或存储数据。而 flash 的写入操作只能在空的或已擦除的单元内进行。因此，在执行 flash 写访问之前常常需要先执行擦除访问，而 flash 具有最高 100K 次的编程次数限制。如果频繁对某个区块擦写，将容易产生坏块。当用户需要利用 flash 存储数据时，Flash 驱动程序的接口并不首推给 APP 使用，更建议使用 FTL（Flash Translation Layer）接口。如果 FTL 无法满足需求而必须使用 Flash driver，请调用 RTL8762C SDK 中带“_locked”后缀的 flash API。

RTL8762C 使用的是外部 SPI flash，用户可以根据需求灵活选型。对于只要求基本的读、写、擦访问，多种 flash 型号可以选用。但是 RTL8762C flash 驱动程序中同时还支持一些高级特性，例如支持切换到四位模式以提升 flash 访问的效率、支持 flash 进深度睡眠模式来降低功耗、支持 Flash 软件块保护以阻止非预期的擦写操作以及提供 flash 高速读取接口，这些功能并不是所有的 flash 型号都适用。因此， Realtek 将会同步发布 RTL8762C 支持的 flash AVL（Approved Vendor List），要求用户从 AVL 中选择 Flash 模型，以保证 flash 相关的功能都能够支持。

RTL8762C 中集成了 SPI 控制器以支持在 SPI Flash 上直接执行代码（XIP），但使用上有一些限制条件。

2 Flash AVL

RTL8762C 采用 SPI 接口的外部 flash，方便用户根据不同的需求去选用不同供应商的不同型号的 flash。但是由于 flash 的供应商众多，有如 MXIC、Winbond、GD、FUDAN、ESMT 等。即使是同一家供应商，也会存在不同型号的 flash 对同一个操作使用不同的命令情况，甚至具有不同的功能和使用限制。

因此，RTL8762C 为了支持 flash 的一些高级功能，例如四位模式（Quad mode）、软件块保护（Software Block Protect，BP）等待，需要对选用的 flash 进行筛查。因此 Realtek 提供了一个 AVL，列出 Realtek 已经验证通过的 flash 供应商的型号，具体请参考 Realtek 同步发布的 RTL8762C AVL。

3 基本操作

Flash 的三大基本操作——读、写、擦，用户都必须调用带“_locked”后缀的 API。由于所有访问 flash 的操作都需要通过 flash 控制器（SPIC）来完成，而 SPIC 是唯一的。这些带“_locked”后缀的 API 将保证访问 flash 之前必须先获取信号量，访问完成后释放信号量，以防止多任务或者中断打断时破坏 SPIC 命令序列的原子性。

3.1 Flash API

- **读操作**

bool flash_auto_read_locked(uint32_t addr, uint32_t *data);

bool flash_read_locked(uint32_t start_addr, uint32_t data_len, uint8_t *data);

- **写操作**

bool flash_auto_write_locked(uint32_t start_addr, uint32_t data);

bool flash_auto_write_buffer_locked(uint32_t start_addr, uint32_t *data, uint32_t len);

bool flash_write_locked(uint32_t start_addr, uint32_t data_len, uint8_t *data);

- **擦操作**

bool flash_erase_locked(T_ERASE_TYPE type, uint32_t addr);

3.2 Flash 访问模式

flash 控制器（SPIC）支持两种模式——用户模式和自动模式来执行三种基本操作。在用户模式下，访问 flash 需要先设置相关的寄存器，然后将串行数据传输到 flash，这一系列的操作都由使用者自行控制。而自动模式下，是配置 flash 控制器（SPIC）将这一系列的操作由硬件自动完成，使用户使用起来更为简单。在 RTL8762C 的 SDK 中两种模式的接口都有提供，其中自动模式的接口均带有“auto”字样，其他的接口则是以用户模式访问 flash。例如 flash_auto_read_locked 是以自动模式读访问 flash，而 flash_read_locked 是以用户模式读访问 flash。

为加速 flash 的访问效率，RTL8762C 集成了 16KB 的 cache。flash 控制器同步支持两组最大 8MB 大小的映射地址空间，对应 cache 地址空间 [0x00800000, 0x01000000)和 non-cache 地址空间[0x01800000, 0x02000000)。将 mem_config.h 中宏“SHARE_CACHE_RAM_SIZE”配置为 8K 或者 0 可以打开 flash 的 cache 功能。在 cache 被使能的情况下，以自动模式读写 flash 需要注意以下两点：

1. 禁止以自动模式写访问 flash 的 cache 地址空间，因此在调用接口 flash_auto_write_locked 和

flash_auto_write_buffer_locked 时不管传入的是 cache 地址还是 non-cache 地址空间，在底层都将统一处理成 non-cache 地址。

2. 以自动模式读访问 flash， 如果访问 cache 地址，只能读取 RO 类型，如代码或 const 数据。一旦读取 RW 类型数据，有可能读回的是 cache 中未 flush 的旧值。

可以看出，以自动模式访问 flash 使得 flash 的读写操作像 RAM 访问那样简单，但是也存在着安全隐患。访问 flash 的起始地址为 0x01800000，结束地址是依据所选 flash 的大小决定的。如果用户使用不当，操作到一些非预期的地址空间，例如存放代码和重要数据的空间，将会导致程序运行错误或者运行不起来。因此，RTL8762C 还支持 flash 软件块保护功能，一旦使能可以将 flash 低地址开始的部分空间上锁，被上锁的区域将无法执行 flash 的写和擦的操作，从而保护代码区和重要数据区域。

4 位模式

除了标准串行外围接口（SPI）之外，大多数 flash 模型还支持高性能的双位/四位模式 I/O SPI，由以下六个引脚控制。

- 串行时钟(CLK)
- 片选 (CS#)
- 串行数据 IO0 (DI)
- 串行数据 IO1 (DO)
- 串行数据 IO2 (WP#)
- 串行数据 IO3 (HOLD#)

4.1 三种位模式

1. 单位模式：标准的 SPI 模式，也称为 1 位模式，只使用 CLK, CS#, DI 和 DO。WP#作为写保护的输入，而 HOLD#作为保持功能的输入.
2. 双位模式：使用 CLK, CS#, 并将 DI、DO 分别用作 IO0、IO1。WP#和 HOLD#的功能和和单位模式中一致。
3. 四位模式：使用 CLK, CS#, 并将 DI、DO、WP#、HOLD#分别用作 IO0、IO1、IO2、IO3。由于 6 根管脚全部被使用，所以四位模式下写保护和保持功能不可用。

虽然几乎所有的 flash 都支持双位和四位模式，但是 flash 的命令集和位模式切换原则并不相同，这也是需要 AVL 的原因之一。

4.2 位模式切换

RTL8762C 为支持更多 flash 模型，在启动时默认 flash 都是在 1 位模式。如果用户需要切换至高速位模式（2 或 4 位模式），可调用 SDK 中提供的接口：flash_try_high_speed（可参考 SDK 中 Bee2-SDK.chm）切换至高速位模式，并由参数 “bit_mode”配置 flash 尝试切换至的高位模式，函数返回值表示是否切换成功，如果切换失败将会切回 1 位模式。需要注意的是，如果是切换至 4 位模式，将要额外占用引脚 P1_3 和 P1_4 用作 IO2 和 IO3，同时硬件电路上也应做支持。

SDK 中提供的进行位模式切换的接口函数原型如下：

uint32_t flash_try_high_speed(T_FLASH_MODE bit_mode);

5 软件块保护

虽然 Flash 支持通过硬件保护管脚（#WP）来锁定整个 flash 以防止写入和擦除操作，但仍有以下两个缺点：

1. 如果使用管脚#WP 用作保护功能，就无法使用 flash 的四位模式。
2. 硬件保护只能选择保护整颗 flash 或者全部不保护，不能保护部分 flash 空间。

RTL8762C 是通过一个更为灵活的机制——软件块保护（Software Block Protect，BP）来防止意外的 flash 的擦写操作。其原理是利用 flash 状态寄存器中的一些 BP 位来选择要保护的范围。如表所示，Flash 使用状态寄存器中的 BX（X）位来识别要锁定的块的数量，以及 TB 位来决定锁定的方向。RTL8762C 只支持从 flash 的低地址端开始上锁。

STATUS REGISTER(1)					W25Q16DV (16M-BIT) MEMORY PROTECTION(3)			
SEC	TB	BP2	BP1	BP0	PROTECTED BLOCK(S)	PROTECTED ADDRESSES	PROTECTED DENSITY	PROTECTED PORTION(2)
X	X	0	0	0	NONE	NONE	NONE	NONE
0	0	0	0	1	31	1F0000h – 1FFFFFh	64KB	Upper 1/32
0	0	0	1	0	30 and 31	1E0000h – 1FFFFFh	128KB	Upper 1/16
0	0	0	1	1	28 thru 31	1C0000h – 1FFFFFh	256KB	Upper 1/8
0	0	1	0	0	24 thru 31	180000h – 1FFFFFh	512KB	Upper 1/4
0	0	1	0	1	16 thru 31	100000h – 1FFFFFh	1MB	Upper 1/2
0	1	0	0	1	0	000000h – 00FFFFh	64KB	Lower 1/32
0	1	0	1	0	0 and 1	000000h – 01FFFFh	128KB	Lower 1/16
0	1	0	1	1	0 thru 3	000000h – 03FFFFh	256KB	Lower 1/8
0	1	1	0	0	0 thru 7	000000h – 07FFFFh	512KB	Lower 1/4
0	1	1	0	1	0 thru 15	000000h – 0FFFFFh	1MB	Lower 1/2
X	X	1	1	X	0 thru 31	000000h – 1FFFFFh	2MB	ALL

图 5-1 Flash Block Protect

RTL8762C 默认是利用 BP 功能来保护一些重要的数据（如配置的参数）和代码段，而不是整个 flash 空间。不过，BP 函数并没有开放给客户，而是提供一个配置选项“bp_enable”去控制是否打开 BP 功能（默认是关闭的）。这是由于不同的 Flash 供应商和模型有不同的规则和限制，而且 Flash 状态寄存器默认使用 NVRAM 类型来保持数据，BP 函数需要改变 BP 位来切换不同的保护级别，但是 NVRAM 有 100K 次编程限制。虽然大多数厂商支持使用 0x50 命令将 Flash 状态寄存器切换至 SRAM 类型，但也不是所有的 flash 厂商都支持，例如 MXIC。那么，频繁访问状态寄存器将会会损坏 flash。

目前，如果选择打开 BP 功能，将会依据所配置的 flash layout 和所选 flash 的型号去锁到一个最大可以上锁的范围。Flash BP 上锁原则和 flash layout 配置注意事项详见 RTL8762C Memory User Guide 中 5.1 节。通过修改 APP 工程目录下的 otp_config.h 中的宏 FLASH_BLOCK_PROTECT_ENABLE 为 1 将启用该功能。

6 省电

Flash 的功耗模式主要分以下三种场景：工作、待机和 Power Down。工作模式耗电一般在 10 mA 数量级，待机模式一般是几十 uA 左右，Power Down 模式的耗电更低，甚至低于 1 uA。

Flash 在没有访问时会自动进入待机模式，当需要再访问时会自动进入工作状态。而要进入 Power Down 模式，需要使用特定的命令：大多数 Flash 使用命令 0xB9 控制其进入 Power Down 模式，使用命令 0xAB 退出 Power Down 模式，而 MXIC 是通过切换#CS 引脚退出 Power Down 模式。

需要注意的是，Flash 在进入 Power Down 模式之后，除了退出 Power Down 命令（0xAB）之外，接收其他的命令都是危险的。因为 Flash 在进入 Power Down 模式之后只接受唤醒命令退出 Power Down 模式，其他命令都将被忽略，但 Flash 控制器可能会进入无限循环等待 Flash 的响应。

为了防止滥用 Flash Power Down 模式带来的风险，系统的 DLPS 机制中已经加入 Flash Power Down 模式的控制：进入 DLPS 时自动下指令使 Flash 进入 Power Down 模式，在退出 DLPS 时唤醒 Flash。

7 Flash 高速读取接口

添加 sdk\src\flash\flash_hs_read.c 到工程中，使用以下三个接口可以支持 flash 的高速读取。

7.1 flash_split_read_locked

受限于 Flash 控制器的 FIFO 大小（64 bytes），用户模式单次操作只能读取 60 字节数据，因为额外还要 1 字节的读取命令和 3 字节的目标地址。如果需要读取的长度大于 60 字节，就需要重新设置下一次读操作的起始地址和长度。这样的操作增加了数据开销。针对读取连续空间的一长串数据的场景，可以利用 Flash 控制器来加速 flash 读操作的效率。通过该接口的函数原型如下，需要保证读取长度必须是 4 字节对齐：

bool flash_split_read_locked(uint32_t start_addr, uint32_t data_len, uint8_t *data, uint32_t *counter)

7.2 flash_auto_dma_read_locked

每个自动读操作可以一次读取 1 个字（4 字节）。如果读取连续空间的一长串数据，将比用户模式花费更多的开销。为了提高自动读写的性能，引入 DMA，接口的函数原型如下：

```
bool flash_auto_dma_read_locked(T_FLASH_DMA_TYPE dma_type, FlashCB flash_cb,
                                uint32_t src_addr, uint32_t dst_addr, uint32_t data_len);
```

7.3 flash_auto_seq_trans_dma_read_locked

通过 DMA 读取 flash 完成后会执行 FLASH_GDMA_HANDLER，函数原型如下：

```
bool flash_auto_seq_trans_dma_read_locked(T_FLASH_DMA_TYPE dma_type, FlashCB flash_cb,
                                          uint32_t src_addr, uint32_t dst_addr, uint32_t data_len);
```

8 XIP

RTL8762C 剩余大约 75K Bytes 的 RAM 空间可提供给 APP 开发使用，RAM 使用情况详情可参考 RTL8762C Memory User Guide。如果所剩的 RAM 空间足够，可将 APP 直接执行在 RAM 上，有利于提升性能和降低功耗。但是，如果 APP 代码较大，所剩的 RAM 空间不够，需要将部分或者全部 APP 代码放在 flash 上执行。RTL8762C 上执行 XIP 的相关配置总结如下：

1. 宏 FEATURE_RAM_CODE（mem_config.h 中配置）：配置为 1 时，默认不加任何 section 修饰的代码都将在 RAM 上执行。相反，配置为 0 时，默认不加任何 section 修饰的代码都将在 flash 上执行。
2. Section 修饰（参考 app_section.h）
   a) APP_FLASH_TEXT_SECTION：指定代码在 flash 上执行；
   b) DATA_RAM_FUNCTION：指定代码在 data RAM 上执行。如果 RAM 空间不足，要优先将时间敏感的代码放在 RAM 上执行以保证效率。
3. 场景切换（参考 app_section.h 和 overlay_mgr.h）： APP 开发时首先根据需要划分出不同的场景并定义加载场景信息表，然后手动对不同的场景代码用不同的 section 关键字修饰，最后在切换场景的地方调用加载函数 load_overlay。目前 RT8762C SDK 默认支持如下三种场景，APP 可遵循一定的原则扩充。

```
#define OVERLAY_SECTION_BOOT_ONCE   __attribute__((section(".app.overlay_a")))
#define OVERLAY_B_SECTION           __attribute__((section(".app.overlay_b")))
#define OVERLAY_C_SECTION           __attribute__((section(".app.overlay_c")))
```

4. 提高 XIP 执行代码的效率的两种方式：
   a) 打开 cache 功能：配置 SHARE_CACHE_RAM_SIZE 为 8K Bytes 或者 0；分别表示 cache 大小为 8K Bytes 或者 16K Bytes。
   b) 将 flash 切换至双位或者四位模式：调用 flash_try_high_speed。

RTL8762C 通过 SPIC 的支持可以通过自动模式去访问 flash 并且直接在 SPI Flash 上执行代码。但是用户有可能通过调用提供的接口在用户模式下对 flash 执行读写擦操作，而用户模式访问 flash 的操作不是原子性的，如果被另一个自动模式访问操作打断，就会造成 flash 访问异常。为保证用户模式访问 flash 操作的原子性，XIP 时需要遵循以下的使用限制和注意事项：

1. RTL8762C 中提供的用户模式访问 flash 的接口都带“_locked”后缀，默认加临界区保护，会关闭 0 和 1 优先级以下的中断，也就是中断优先级数值设置为 2 到 7 的中断（中断优先级数值越大，中断优先级越低）。如果 flash 操作的时间过久，比如一次写大量的数据，建议将一次写的动作拆分成多次少量数据写的动作，以避免关闭中断时间过长而丢中断。
2. 如果有时间非常关键的中断（中断优先级 0 和 1 的中断）需要处理，需要保证中断服务例程本身不能 XIP，中断服务例程中也禁止访问 flash。因为中断服务例程中不能存在会引起任务切换到一

个 XIP 任务的操作。