乐鑫信息科技
Espressif Systems

Espressif IOT
Flash RW Operation

CONFIDENTIAL

Status	Released
Current version	V0.2
Author	Fei Yu
Completion Date	2014.9.15
Reviewer	JG Wu
Completion Date	2014.9.15

[√] CONFIDENTIAL
[ ] INTERNAL
[ ] PUBLIC

Version info

Date	Version	Author	Comments/Changes
2014.6.27	0.1	Fei Yu	Draft
2014.9.15	0.2	Fei Yu	Revise flash RW APIs

CONFIDENTIAL

Disclaimer and Copyright Notice

Information in this document, including URL references, is subject to change without notice.

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

The Wi-Fi Alliance Member Logo is a trademark of the Wi-Fi Alliance.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

Copyright © 2013 Espressif Systems Inc. All rights reserved.

Table of Contents

CONFIDENTIAL

1、 Foreword

This document introduces flash RW APIs, and related matters needing attention, last but not least, introduces the flash RW protection method used in esp_iot_sdk, and some other methods as reference.

CONFIDENTIAL

2、 Flash APIs

APIs below can read/write/erase the whole flash, flash sectors start counting from 0, 4Kbytes per sector.

spi_flash_erase_sector ： Erase flash sector

spi_flash_write ： Write data to flash

spi_flash_read ： Read data from flash

Return：

```
Typedef enum{
    SPI_FLASH_RESULT_OK,
    SPI_FLASH_RESULT_ERR,
    SPI_FLASH_RESULT_TIMEOUT
}SpiFlashOpResult;
```

2.1. spi_flash_erase_sector

Function：	Erase flash sector.
Prototype：	SpiFlashOpResult spi_flash_erase_sector (uint16 sec) Param： uint16 sec – sector number, start counting from sector 0, 4Kbytes per sector.
Return：	SpiFlashOpResult

2.2. spi_flash_write

Function：	Write data to flash.
Prototype：	SpiFlashOpResult spi_flash_write (uint32 des_addr, uint32 *src_addr, uint32 size)

CONFIDENTIAL

	Param： uint32 des_addr - destination address on flash. uint32 *src_addr – source address of data. Uint32 size - data length.
Return：	SpiFlashOpResult

2.3. spi_flash_read

Function：	Read data from flash.
Prototype：	SpiFlashOpResult spi_flash_read(uint32 src_addr, uint32 * des_addr, uint32 size) Param： uint32 src_addr - source address on flash. uint32 * des_addr – destination address to keep data. uint32 size - data length.
Return：	SpiFlashOpResult

CONFIDENTIAL

3、 Notice

While using flash APIs , please pay attention to follow items:

（1） Don’t change program area.

（2） Don’t change system-data area.

In esp_iot_sdk ：

Don’t support OTA	Program area	eagle.app.v6.flash.bin：	start from 0x00000
		eagle.app.v6.irom0text.bin:	start from 0x40000
	System-data area	The last 4 sectors(16KBytes) on flash	
Support OTA	Program area	boot.bin ：	start from 0x00000
		user1.bin ：	start from 0x01000
		user2.bin ：	start from 0x41000
	System-data area	The last 4 sectors(16KBytes) on flash	

（1） ”Program area” only mentions start address, the space it costs depends on the size of each bin.

（2） System-data area：

esp_init_data_default.bin ： the forth sector from the last on flash

blank.bin ： the second sector from the last on flash

- Take 512KB Flash，for example

esp_init_data_default.bin in 0x7C000; blank.bin in 0x7E000

（3） Take user_light.c in esp_iot_sdk as an example of flash RW. In IOT_Demo, user data area contains 4 sectors which starts from 0x3C000.

- For example, master_device_key.bin(only if using Espressif Cloud need it) burns in 0x3E000

CONFIDENTIAL

4、 Flash RW Protection

4.1. Foreword

Flash is erased sector by sector, which means it has to erase 4Kbytes one time at least. When you want to change some data in flash, you have to erase the whole sector, and then write it back with the new data.

So, if power off during the flash writing, data of the whole sector will be missing.

According to that, Espressif gives an example of flash RW protection and some other advices about flash RW protection as reference。

In esp_iot_sdk, there is an example of flash RW protection.

4.2. Basic Principle

Flash RW protection example in IOT_Demo, uses 3 sectors to provide a reliable storage of 4Kbytes.

Use sector 1 & sector 2 as data sector to save the same data of this time and the last time, this two sector alternate writing, so that there is always a sector to be backup. Use sector 3 as flag sector to keep the flag which sector (sector 1 or 2) saved the latest data.

1） Default sector is sector 2, so at first, copy data from sector 2 to RAM.

2） Then, the first time data changes, save the new data in sector 1.

If power off unexpectedly during this phase, writing sector 1 will fail, but sector 2 & 3 are all fine; then power on, it will still copy data from sector 2 to RAM.

3） Modify sector 3 to change flag to 0， means sector 1 saved the latest data.

If power off unexpectedly during this phase, writing sector 3 will fail, but sector 1 & 2 are all fine; then power on, because of the invalid data in sector

CONFIDENTIAL

3, it will still copy data from sector 2 to RAM by default. Although data in sector 2 is not the latest data, but it can still work, what we lost is just the latest update.

4） The next time we want to change data in flash, we have to read flag from sector 3 first, if flag is 0, means the latest data stored in sector 1 , so we write data to sector 2 this time; if flag is not 0, assume the latest data stored in sector 2, we will write data to sector 1 this time.

If power off unexpectedly during this phase, please look back to step 2 & 3 as reference.

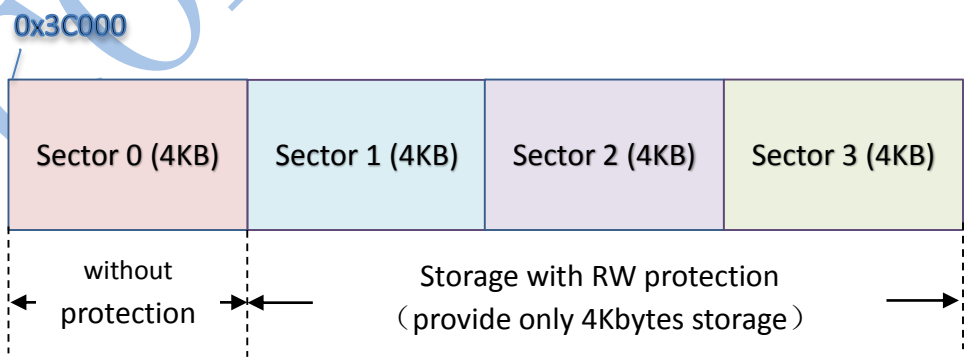
5） Only after writing data sector (sector 1 or 2) accomplished, we will write flag sector (sector 3) to change the flag.

Note：

Write data sector first, then flag sector. This order ensure the integrity of data.

4.3. Example In IoT_Demo

esp_platform_saved_param。In IOT_Demo， use 4 sectors which starts at 0x3C000 as user data area. In user data area, use 3 sectors(0x3D000、0x3E000、0x3F000) to provide flash RW protection, offer 4Kbytes safety storage.



CONFIDENTIAL

Use apis: user_esp_platform_load_param and user_esp_platform_save_param to access to the “Storage with RW protection” area in above picture.

user_esp_platform_load_param - read data from flash

user_esp_platform_save_param - write data to flash

Struct esp_platform_saved_param is the parameter Espressif used that stored in flash as user data. If you want to use this storage, add your parameters in the struct esp_platform_saved_param and call the two apis above to read/write.

4.3.1. user_esp_platform_load_param

Function：	Read user data from flash.
Prototype：	void ICACHE_FLASH_ATTR user_esp_platform_load_param(struct esp_platform_saved_param *param) 参数： struct esp_platform_saved_param *param - data point
Return：	None

4.3.2. user_esp_platform_save_param

Function：	Write user data to flash.
Prototype：	void ICACHE_FLASH_ATTR user_esp_platform_save_param(struct esp_platform_saved_param *param) 参数： struct esp_platform_saved_param *param - data point
Return：	None

CONFIDENTIAL

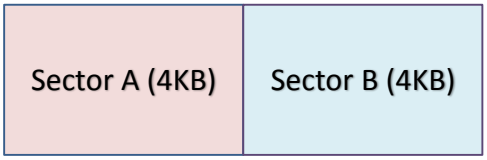
4.4. Advices

Here gives some advices of flash RW protection.

4.4.1. Advice A

Principle: “data sector switch” + “counter in head” + ”check-code in the end”

It takes 2 sectors to provide storage with RW protection of 4Kbytes.



Details：

Two sectors switch to store the user data without flag sector.

Keep counters in the first byte of both data sectors. Every time writing data, count add 1 and write into the first byte of data sector. Compare the value of counter in both data sectors to distinguish which one store the latest data. Keep check-code at the end of both data sectors, such as CRC or checksum, ensure the integrity of data. For example:

1） Data stores in sector A by default, the value of counter in sector A is 0xFF, copy data from sector A to RAM.

2） The first data changing will be written into sector B, the value of counter in sector B will be 1, check-code at the end of sector B.

3） Next time we need to save data, we will compare the value of counter between two sectors, get that the latest data was stored in sector B, so data goes to sector A this time, counter in sector A records 2, check-code at the end.

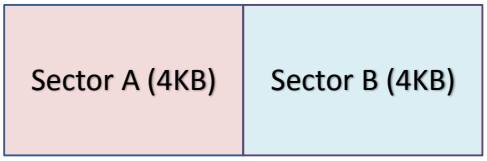
4） If power off unexpectedly, data of sector which is writing maybe lose. When power on, compare the value of counters between sector A and B, read data from the one has large count, check its integrity with the check-code, if pass, use this sector, otherwise, use the other sector, check, and load data.

CONFIDENTIAL

4.4.2. Advice B

Principle: “sector backup” + “check-code in the end”

It takes 2 sectors to provide storage with RW protection of 4Kbytes.



Details：

Always write data to sector A, after that, write the same data to sector B as backup, keep check-codes (such as CRC、checksum) both in the end of sector A and sector B.

1） Read data from sector A，use check-code to confirm its integrity.

2） Write data to sector A, check-code at the end of it.

3） After finish writing sector A, write the same data to sector B as backup, check-code at the end of it.

4） If power off unexpectedly, data of sector which is writing maybe lose. Then power on, read data from sector A, use check-code to confirm its integrity. If pass, all goes as normal; if fail, read data from sector B, check, and program goes on.

CONFIDENTIAL