AN12949

Flash Programming Tips for LPC5500 Series

Rev. 0 — 08/2020 Application Note

Contents

1 Introduction

1.1 LPC5500 series introduction

The LPC55S00 is a family of Arm® Cortex® -M33-based MCUs for embedded applications. It features a rich peripheral set leveraging the new ARMv8M architecture. This document focuses on LPC5500's series flash operation.

Compared with legacy LPC parts, LPC4300, LPC54000, LPC1800, LPC800, the LPC5500 uses new flash IP and is enabled with ROM API driver. Lots of new features have been introduced into this new series. At the same time, it might be a little complicated for new users to operate LPC5500 series' flash. This document aimes to removing those barriers and giving some useful tips for users.

If you have already used LPC5500 series and met some issues with operating internal flash, go to Summary first to check if you follow all the rules and tips mentioned in this document.

1.2 Related UM chapters

1. Chapter 2: LPC55S6x/LPC55S2x/LPC552x Memory Map
2. Chapter 5: LPC55S6x/LPC55S2x/LPC552x Flash
3. Chapter 9: LPC55S6x/LPC55S2x/LPC552x Flash API
4. Chapter 10: LPC55S6x/LPC55S2x/LPC552x Protected Flash Region
5. UM attached file: LPC55S6x_LPC55S2x_LPC552x Protected Flash Region v1.1.xls

2 Implementation

2.1 Flash memory layout

LPC5500 series have three flash configurations:

- 640 KB FLASH/320 KB RAM
- 512 KB FLASH/256 KB RAM
- 256 KB FLASH/144 KB RAM

In some configurations, users cannot use all flash memory to store code and data. Because LPC5500 series use last few pages as Protected Flash Region (PFR) to store important information and some internal manufacture settings.

2.1.1 Page size and sector size

The sector size of the LPC5500 series internal flash memory is 32 KB and the page size is 512 bytes.

The minimum programming size is the page size, 512 bytes.

The minimum erase size is page size, 512 bytes.

2.1.2 PFR region in last pages of flash memory

Unlike legacy LPC parts, all LPC5500 series use last few pages of flash memory as PFR. PFR is used to store some important user and manufacture configuration information. PFR cannot be used as normal flash to store data and code. Programming PFR needs special cautions, or the chip might be malfunctioned. Figure 1 shows the flash memory layout.

Figure 1. LPC5500 series flash memory layout

As shown in Figure 1:

The PFR size of LPC55S6x/LPC55S2x is 10 KB, located in 0x0009_D800 – 0x0009_FFFF.

The PFR size of LPC55S1x is 12 KB, located in 0x0003_D000 – 0x0003_FFFF.

- For LPC55S6x/LPC55S2x 256/512 KB part, the total flash size can be used by customer is 256/512 KB.
- For LPC55S6x/LPC55S2x 640 KB part, the total flash size can be used by customer is 630 KB.
- For LPC55S1x 64/128 KB part, the total flash size can be used by customer is 64/128 KB.
- For LPC55S1x 256 KB part, the total flash size can be used by customer is 244 KB.

2.2 Flash operation

LPC5500 series' flash implement internal ECC management, including single bit correction and error correction logging. This makes flash more secure and robust than others. But this feature also brings an issue: when performing AHB read, memcpy, of an erased or corrupted flash memory, a HardFault will occur. This may happen when users try to read, AHB read, an erased while not programmed flash page.

See Notes in **Block diagram** in *LPC55S6x/LPC55S2x/LPC552x User manual* (document UM11126).

NOTE

When performing AHB reads of the flash memory contents, a hardware fault will occur if an unrecoverable error is detected. Read operations performed using flash controller commands (see **Command listing**) will not cause a hard fault.

The following section will discuss how to work around this issue.

Also, PFR region resides the last pages of flash, making it vulnerable to be erased accidently. To avoid this, we recommend using ROM API to operate flash instead of manipulating flash register directly. ROM API provide internal protection to avoid user erase PFR accidently. While the flash register gives you free access to all flash data, it is a little "dangerous".

We suggest using ROM API to access data flash and PFR. The flash register operation is not recommended.

Finally, flash erase and program can only be done when core clock <=100 MHz. if the CPU running at higher than 100 Mhz, user must lower CPU frequency first than erase and program flash.

2.2.1 Flash ROM API

The BootROM of LPC5500 series provides a set of Flash APIs for users. ROM API table is located at the address of `0x130010f0`.

Figure 2 shows the ROM API address layout.

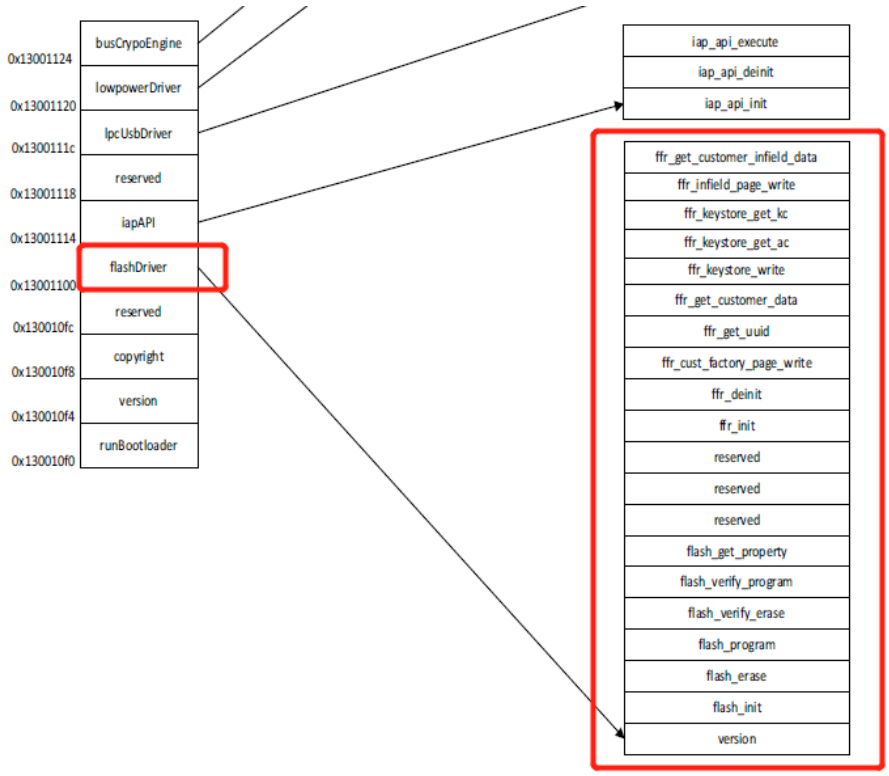

Figure 2. LPC5500 flash API layout

SDK provides full examples of how to use flash driver to operate internal flash. The example is located at:

\boards\lpcxpresso55s28\driver_examples\flashiap

We suggest using ROM API to access data flash and PFR. The flash register operation is not recommended.

2.2.2 Flash Init

To use Flash ROM API to operate flash, you need first initialized flash instance. The `FLASH_Init` API in SDK needs to pass a `flash_config_t` structure. When calling this API, the structure returns some flash parameters, such as, `FLashSecterSize`, `TotalFlashSize`, `PageSize`, etc. Users should check and confirm that the parameters are correct.

```
#define ROM_API_TREE ((*uint32_t)0x130010f0)
#define FLASH_API_TREE ((flash_driver_interface_t*) ROM_API_TREE[3])
flash_config_t flashConfig;
status = FLASH_API_TREE->flash_init(&flashConfig);
```

Figure 3. Flash Init code

2.2.3 Flash erase

The flash erase API is easy to use. The **start** and **lengthInBytes** filed need to be 512 bytes aligned.

```
#define ERASE_KEY 0x6b65666b

status = FLASH_API_TREE->flash_erase(&flashConfig, 0x0, 0x4000, ERASE_KEY);
```

Figure 4. Flash erase

2.2.4 Flash program

Same as Flash Erase API, the **start** and **lengthInBytes** filed need to be 512 bytes aligned.

```
uint8_t programBuffer[512];
for (uint32_t i=0; i<sizeof(programBuffer); i++)
{
    programBuffer[i] = (uint8_t)(i & 0xFF);
}
status = FLASH_API_TREE->flash_program(&flashConfig, 0x0, programBuffer,
                                       sizeof(programBuffer));
```

Figure 5. Flash program

2.2.5 Flash read

Accessing or reading blank flash area will hit hardfault, so before performing the access/read, check whether the flash region is blank or not.

SDK provides an API named **FLASH_VerifyErase** for blank checking. Before accessing or reading a flash page that we don't know blank or not, use **FLASH_VerifyErase** to confirm. If the page is blank, the content in this page is all `0xFF`, and reading or accessing it directly will hit hardfault. If the page is not blank, it can be read/access as usual.

```
status = FLASH_VerifyErase(&flashInstance, flash_addr, 512);

if (status == kStatus_Success)
{
    /* it's a erased flash, just return all 0xFF */
    memset(read_buf, 0xFF, sizeof(read_buf));
}
else
{
    /* flash not erased, AHB read */
    memcpy(read_buf, (void*)flash_addr, sizeof(read_buf));
}
```

Figure 6. Flash read

3 Summary

3.1 Make sure core clock under 100 MHz when operating flash

All flash operations must be done when the core clock is under 100 MHz.

3.2 Caution on PFR region

Special caution needs to be taken when you operate last few pages near PFR. Do not perform a sector erase (erase 32 KB) of the last flash sector.

3.3 Using ROM API to erase/program flash

Use ROM API to access data flash and PFR. The flash register operation is not recommended because it’s easy to erase PFR accidently.

3.4 Using ROM API: FLASH_VerifyErase before read flash data to void Hardfault issue

Call **FLASH_VerifyErase** before reading flash content. To make sure it’s not an erased/empty page. Otherwise, it might trigger HardFault.

3.5 Align

ROM API: The **start** and **lengthInBytes** fileds of **Flash_Program** and **Flash_Erase** need to be page aligned.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 08/2020

Document identifier: AN12949

arm