MT793X IoT SDK for Serial Flash User Guide

Version: 1.0
Release date: 2022-10-27

Version History

Version	Date	Description
0.1	2021-03-26	Initial draft
0.2	2021-03-30	Add API usage
0.3	2022-10-27	Add OTP

Table of Contents

List of Figures

List of Tables

1 Overview

This document introduces the serial flash controller (SFC).

2 SFC Introduction

2.1 Controller Diagram

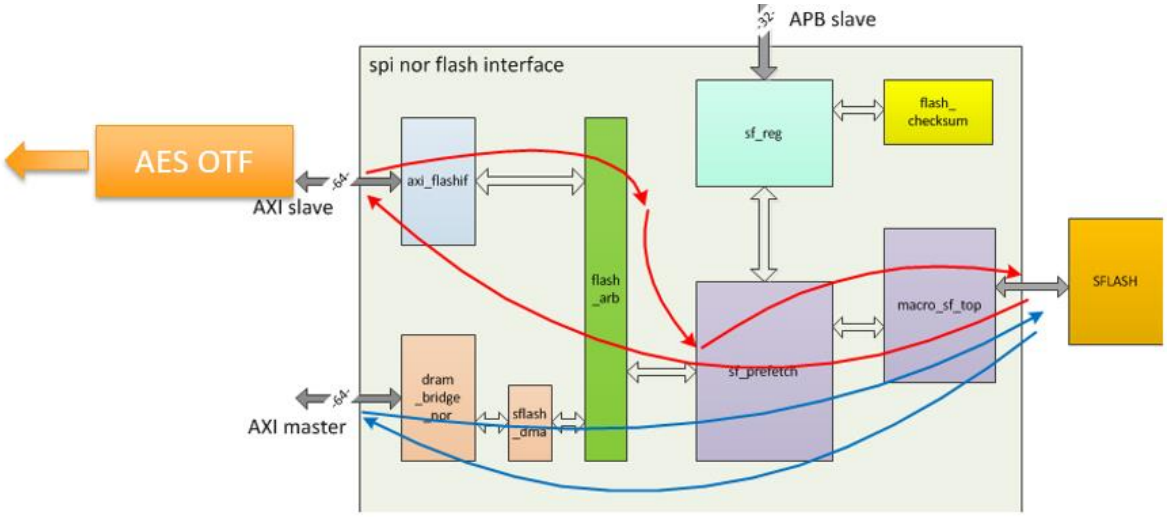

Figure 2-1 Serial NOR Flash Controller Block Diagram

- Map out 512 bytes page program buffer
- Support byte program and page program (SPI mode)
- Support 3/4 bytes address mode
- Support single-bit read, dual output & dual I/O read and quad output & quad I/O read modes
- Read serial NOR flash data through direct read or PIO read
- Support serial NOR flash device of up to 60-MHz frequency

read mode	command	address	data
SPI	1	1	1
Dual output	1	1	2
Dual I/O	1	2	2
Quad output	1	1	4
Quad I/O	1	4	4

Table 2-1 Supported Read Modes

The number in Table 2-1 means the data line used to transfer command/address/data.
In Bootloader, Quad I/O read mode is the default mode.

2.1.1 Access Flash Operation

<1> read flash ID

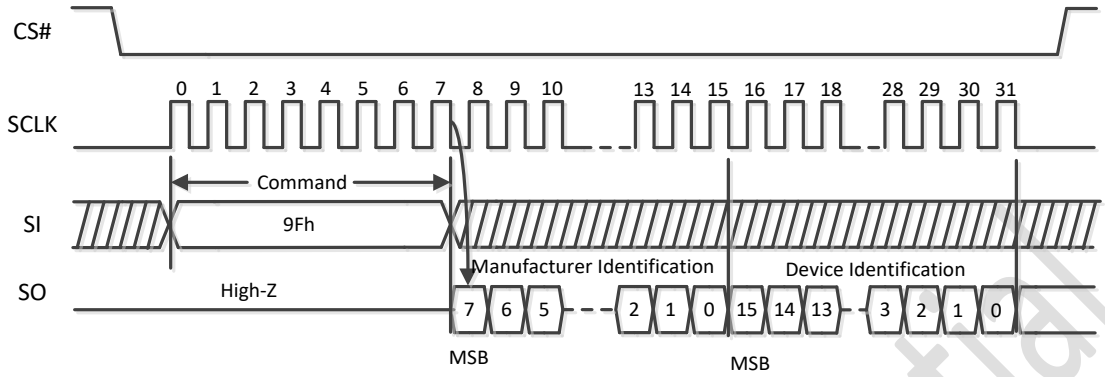

Figure 2-2 Read Flash ID

Step	Address	Register Name	Local Address	R/W	Value	Description
2	SF_Base+0x34	REG_SF_PRGDATA5	REG_SF_PRGDATA5[7:0]	W	8'h9f	**Write the operation command of RDID**
3	SF_Base+0x30	REG_SF_PRGDATA4	REG_SF_PRGDATA4[7:0]	W	8'h00	Write dummy data
4	SF_Base+0x2c	REG_SF_PRGDATA3	REG_SF_PRGDATA3[7:0]	W	8'h00	Write dummy data
5	SF_Base+0x28	REG_SF_PRGDATA2	REG_SF_PRGDATA2[7:0]	W	8'h00	Write dummy data
6	SF_Base+0x04	REG_SF_CNT	REG_SF_CNT[15:0]	W	16'h20	**Write process cycle count**
7	SF_Base+0x00	REG_SF_CMD	REG_SF_CMD[2]	W	1'b1	Trigger controller (Send RDID operation sequence to serial NOR flash)
8	SF_Base+0x00	REG_SF_CMD	REG_SF_CMD[2]	R	1'b0	When this bit is 1'b0, the controller process is done
9	SF_Base+0x38	REG_SF_SHREG0	REG_SF_SHREG0[7:0]	R		Read the JEDEC ID (Device ID-Low Byte)
10	SF_Base+0x3c	REG_SF_SHREG1	REG_SF_SHREG1[7:0]	R		Read the JEDEC ID (Device ID-High Byte)
11	SF_Base+0x40	REG_SF_SHREG2	REG_SF_SHREG2[7:0]	R		Read the JEDEC ID (Manufacturer ID)

<2> erase flash

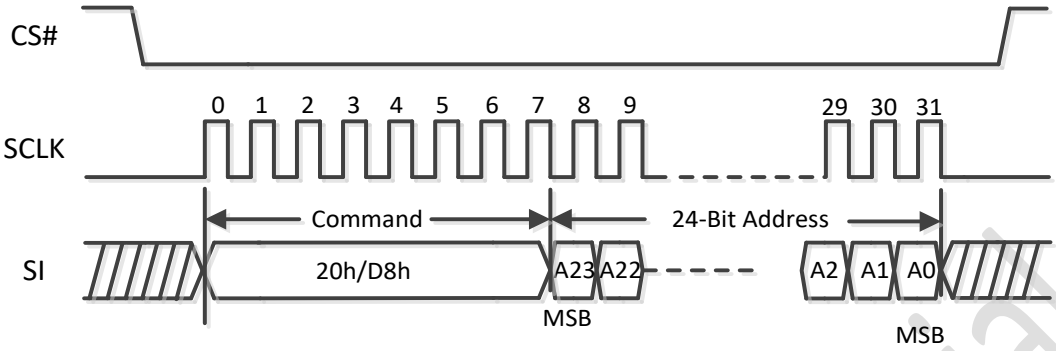

Figure 2-3 Erase Flash (3-byte Address Mode)

Before operations such as read or write operations, which may change flash's content, are performed, SFC must send the write enable command to the flash device.

Step	Address	Register Name	Local Address	R/W	Value	Description
1	SF_Base+0x34	REG_SF_PRGDATA5	REG_SF_PRGDATA5[7:0]	W	8'h20 or 8'hd8	Write the operation command of Sector Erase(0x20) or Block Erase(0xd8)
2	SF_Base+0x30	REG_SF_PRGDATA4	REG_SF_PRGDATA4[7:0]	W	addr[31:24] or addr[23:16]	Write erase address bit31:bit24 when in 4-byte address mode, bit23:bit16 when in 3-byte address mode
3	SF_Base+0x2c	REG_SF_PRGDATA3	REG_SF_PRGDATA3[7:0]	W	Addr[23:16] or addr[15:8]	Write erase address bit23:bit16 when in 4-byte address mode, bit15:bit8 when in 3-byte address mode
4	SF_Base+0x28	REG_SF_PRGDATA2	REG_SF_PRGDATA2[7:0]	W	addr[15:8] or addr[7:0]	Write erase address bit15:bit8 when in 4-byte address mode, bit7:bit0 when in 3-byte address mode
5	SF_Base+0x24	REG_SF_PRGDATA1	REG_SF_PRGDATA1[7:0]	W	addr[7:0]	Write erase address bit7:bit0; this register only needs to be set when in 4-byte address mode
6	SF_Base+0x04	REG_SF_CNT	REG_SF_CNT[15:0]	W	16'h20 or 16'h28	Write process cycle count: set 0x20 for 3-byte address mode set 0x28 for 4-byte address mode
7	SF_Base+0x00	REG_SF_CMD	REG_SF_CMD[2]	W	1'b1	Trigger controller (Send erase operation sequence to serial NOR flash)
8	SF_Base+0x00	REG_SF_CMD	REG_SF_CMD[2]	R	1'b0	When this bit is 1'b0, the controller process is done
9	SF_Base+0x00	REG_SF_CMD	REG_SF_CMD[1]	W	1'b1	(Polling serial NOR flash status) Send read flash status command to serial NOR flash
10	SF_Base+0x08	REG_SF_RDSR	REG_SF_RDSR[0]	R	1'b0	(Polling serial NOR flash status)

Step	Address	Register Name	Local Address	R/W	Value	Description
						Check whether the WIP bit of serial NOR flash status is de-asserted. The erase process completes when this bit is 0.

<3> write flash

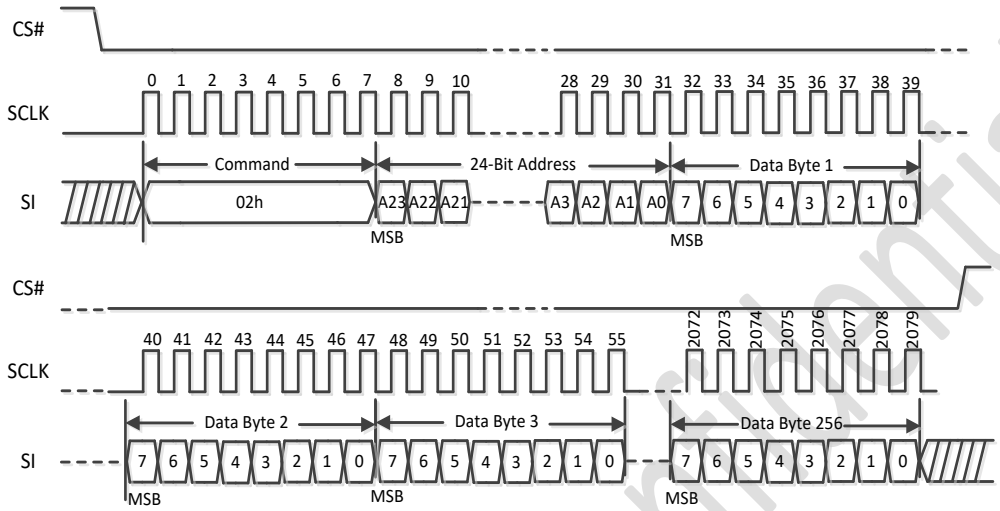

Figure 2-4 Write Flash (3-byte Address Mode)

Step	Address	Register Name	Local Address	R/W	Value	Description
1	SF_Base+0x64	REG_SF_CFG2	REG_SF_CFG2[0]	W	1'b1	Enable the SNFC_Prefetch buffer for write
2	SF_Base+0x10	REG_SF_RADR0	REG_SF_RADR0[7:0]	W	addr[7:0]	Write erase address bit7:bit0
3	SF_Base+0x14	REG_SF_RADR1	REG_SF_RADR1[7:0]	W	addr[15:8]	Write erase address bit15:bit8
4	SF_Base+0x18	REG_SF_RADR2	REG_SF_RADR2[7:0]	W	addr[23:16]	Write erase address bit23:bit16
5	SF_Base+0xc8	REG_SF_RADR3	REG_SF_RADR3[7:0]	W	addr[31:24]	Write erase address bit31:bit24; this register only needs to be set when in 4-byte address mode
6	SF_Base+0x98	REG_SF_PP_DW_DATA	REG_SF_PP_DW_DATA[31:0]	W	data[31:0]	Fill the SNFC_Prefetch buffer by writing program data to this register. Loop N times.(N indicates that data length imported must be an integer multiple of 32 bytes; the maximum data length is 256 bytes)
7	SF_Base+0x00	REG_SF_CMD	REG_SF_CMD[4]	W	1'b1	Trigger page program controller process

Step	Address	Register Name	Local Address	R/W	Value	Description
8	SF_Base+0x00	REG_SF_CMD	REG_SF_CMD[4]	R	1’b0	When this bit is 1'b0, the controller process is done.
9	SF_Base+0x00	REG_SF_CMD	REG_SF_CMD[1]	W	1’b1	(Polling serial NOR flash status) Send read flash status command to serial NOR flash
10	SF_Base+0x08	REG_SF_RDSR	REG_SF_RDSR[0]	R	1’b0	(Polling serial NOR flash status) Check whether the WIP bit of serial NOR flash status is de-asserted. The page program process completes when this bit is 0

Step	Address	Register Name	Local Address	R/W	Value	Description
1	SF_Base+0x1c	REG_SF_WDATA	REG_SF_WDATA[7:0]	W	data[7:0]	One byte data needs to be programmed
2	SF_Base+0x10	REG_SF_RADR0	REG_SF_RADR0[7:0]	W	addr[7:0]	Write program address bit7:bit0
3	SF_Base+0x14	REG_SF_RADR1	REG_SF_RADR1[7:0]	W	addr[15:8]	Write program address bit15:bit8
4	SF_Base+0x18	REG_SF_RADR2	REG_SF_RADR2[7:0]	W	addr[23:16]	Write program address bit23:bit16
5	SF_Base+0xc8	REG_SF_RADR3	REG_SF_RADR3[7:0]	W	addr[31:24]	Write program address bit31:bit24; this register only needs to be programmed when in 4-byte address mode.
6	SF_Base+0x00	REG_SF_CMD	REG_SF_CMD[4]	W	1’b1	Trigger PIO Write controller
7	SF_Base+0x00	REG_SF_CMD	REG_SF_CMD[4]	R	1’b0	When this bit is 1'b0, the controller process is done.
8	SF_Base+0x00	REG_SF_CMD	REG_SF_CMD[1]	W	1’b1	(Polling serial NOR flash status) Send read flash status command to serial NOR flash
9	SF_Base+0x08	REG_SF_RDSR	REG_SF_RDSR[0]	R	1’b0	(Polling serial NOR flash status) Confirm whether the WIP bit of serial NOR flash status is de-asserted. The PIO write process is completed when this bit is 0.

<3> read flash

- call the interface **memcpy**() to read flash
- use CQDMA to read flash

1. When CQDMA or memcpy is used to read data from flash, the source address must contain 0x9000_0000 such as (0x9000_0000 + offset) and must be 8 bytes aligned, interface we provide use CQDMA.
2. The maximum length of address memcpy() can access is 256 MB (0x9000_0000 + 0x10000000)
3. If you want to use interrupt to judge whether CQDMA read finishes,
   - register an interrupt handler;

- trigger the read process;
- poll status until the interrupt handler sets the status.

```
ret = hal_gdma_register_callback(0, GDMA0_ISR_HANDLER, NULL);
if (ret) {
        printf("channel0 hal_gdma_register_callback fail,ret=%d\r\n",ret);
        return UT_STATUS_ERROR;
}
ret = hal_gdma_start_interrupt(0, (uint32_t)dst_addr1, (uint32_t)src_addr, len);
if (ret) {
        printf("channel0 hal_gdma_start_interrupt fail,ret=%d\r\n",ret);
        return UT_STATUS_ERROR;
}
hal_gdma_get_running_status(0, &run_sta);
while(run_sta) {
        hal_gdma_get_running_status(0, &run_sta);
}
```

For specific usage please refer to hal_gdma.c.

2.1.2 API Introduction

hal_flash_status_t hal_flash_init(void);

- Before you access the flash device, hal_flash_init must be called first.

hal_flash_status_t hal_flash_read(
uint32_t start_address,
uint8_t *buffer,
uint32_t length);

- Note that in the MT793X, if you want to use CQDMA to read flash data, the parameter “start_address” must contain the mapping address (0x9000_0000).

hal_flash_status_t hal_flash_write(
uint32_t address,
uint8_t *data,
uint32_t length);

- Note that address + length must not exceed the size of the flash.

hal_flash_status_t hal_flash_erase(

```
                                    uint32_t start_address,
                                    hal_flash_block_t block_type);
```

- This interface provides the granularity of the size to be erased (4 KB, 32 KB, 64 KB); the parameter "start_address" must match the size to be erased， it means that start_address must be integral multiple of erase size.

```
hal_flash_status_t hal_flash_otp_read(
                                    uint32_t addr,
                                    uint32_t len ,
                                    uint8_t *buffer);
```

- Note that address + length must not exceed the size of the flash and different manufacturers may have different otp sizes.

```
hal_flash_status_t hal_flash_otp_write(
                                    uint32_t addr,
                                    uint32_t len ,
                                    uint8_t *buffer);
```

- Note that address + length must not exceed the size of the flash and different manufacturers may have different otp sizes.

```
hal_flash_status_t hal_flash_otp_lock(void);
```

- When the otp area is locked, no more data can be written.

```
hal_flash_status_t hal_flash_otp_lockstatus(uint8_t *lockstatus);
```

- Check if the otp area is locked.

Note: The otp function currently only supports Winbond and MX.

Exhibit 1 Terms and Conditions

Your access to and use of this document and the information contained herein (collectively this "Document") is subject to your (including the corporation or other legal entity you represent, collectively "You") acceptance of the terms and conditions set forth below ("T&C"). By using, accessing or downloading this Document, You are accepting the T&C and agree to be bound by the T&C. If You don't agree to the T&C, You may not use this Document and shall immediately destroy any copy thereof.

This Document contains information that is confidential and proprietary to MediaTek Inc. and/or its affiliates (collectively "MediaTek") or its licensors and is provided solely for Your internal use with MediaTek's chipset(s) described in this Document and shall not be used for any other purposes (including but not limited to identifying or providing evidence to support any potential patent infringement claim against MediaTek or any of MediaTek's suppliers and/or direct or indirect customers). Unauthorized use or disclosure of the information contained herein is prohibited. You agree to indemnify MediaTek for any loss or damages suffered by MediaTek for Your unauthorized use or disclosure of this Document, in whole or in part.

MediaTek and its licensors retain titles and all ownership rights in and to this Document and no license (express or implied, by estoppels or otherwise) to any intellectual propriety rights is granted hereunder. This Document is subject to change without further notification. MediaTek does not assume any responsibility arising out of or in connection with any use of, or reliance on, this Document, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages.

THIS DOCUMENT AND ANY OTHER MATERIALS OR TECHNICAL SUPPORT PROVIDED BY MEDIATEK IN CONNECTION WITH THIS DOCUMENT, IF ANY, ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. MEDIATEK SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, COMPLETENESS OR ACCURACY AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MEDIATEK SHALL NOT BE RESPONSIBLE FOR ANY MEDIATEK DELIVERABLES MADE TO MEET YOUR SPECIFICATIONS OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Without limiting the generality of the foregoing, MediaTek makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does MediaTek assume any liability arising out of the application or use of any product, circuit or software. You agree that You are solely responsible for the designing, validating and testing Your product incorporating MediaTek's product and ensure such product meets applicable standards and any safety, security or other requirements.

The above T&C and all acts in connection with the T&C or this Document shall be governed, construed and interpreted in accordance with the laws of Taiwan, without giving effect to the principles of conflicts of law.