



# **Guide for Porting the Flash Based on HiFMC V100**

**Issue**            **00B03**

**Date**            **2019-01-04**

**Copyright © HiSilicon (Shanghai) Technologies Co., Ltd. 2019. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of HiSilicon (Shanghai) Technologies Co., Ltd.

## **Trademarks and Permissions**



**HISILICON**, and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

## **Notice**

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

## **HiSilicon (Shanghai) Technologies Co., Ltd.**

Address: New R&D Center, 49 Wuhe Road, Bantian,  
Longgang District,  
Shenzhen 518129 P. R. China

Website: <http://www.hisilicon.com>

Email: [support@hisilicon.com](mailto:support@hisilicon.com)



## About This Document

---

### Purpose

This document provides guidance for flash memory porting based on the HiFMC V100 and analysis on common issues.

### Related Version

The following table lists the product versions related to this document.

Product Name	Version
Hi3519	V100
Hi3519	V101
Hi3516A	V200
Hi3518E	V200/V201
Hi3516C	C200
Hi3516C	V300
Hi3516E	V100
Hi3516E	V200
Hi3516E	V300
Hi3518E	V300
Hi3516D	V200
Hi3559	V100
Hi3556	V100
Hi3536C	V100
Hi3536D	V100
Hi3531D	V100
Hi3521D	V100



Product Name	Version
Hi3531A	V100
Hi3521A	V100
Hi3559A	V100
Hi3559C	V100
Hi3519A	V100
Hi3556A	V100
Hi3516C	V500
Hi3516D	V300
Hi3516A	V300
Hi3559	V200
Hi3556	V200





## Intended Audience

This document is intended for:


- Technical support personnel
- Software development engineers

## Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
	Indicates an imminently hazardous situation which, if not avoided, will result in death or serious injury.
	Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.
	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury.
	Indicates a potentially hazardous situation which, if not avoided, could result in equipment damage, data loss, performance deterioration, or unanticipated results. NOTICE is used to address practices not related to personal injury.



Symbol	Description
 <b>NOTE</b>	<p>Calls attention to important information, best practices and tips.</p> <p>NOTE is used to address information not related to personal injury, equipment damage, and environment deterioration.</p>

## Change History

Changes between document issues are cumulative. The latest document issue contains all the changes made in earlier issues.

### Issue 00B03(2019-01-04)

This issue is the third draft release, which incorporates the following changes:

Section 1.2.1.5 is added.

### Issue 00B02(2018-05-15)

This issue is the second draft release, which incorporates the following changes:

Section 1.2 is modified.

Section 1.2.1.4 is added.

### Issue 00B01 (2017-10-18)

This issue is the first draft release.



# Contents

<b>1 Porting New SPI Flash Memories.....</b>	<b>1</b>
1.1 Introduction .....	1
1.2 Porting an SPI NOR Flash Memory .....	1
1.2.1 Porting an SPI-NOR Flash Memory Under U-Boot .....	2
1.2.2 Porting an SPI-NOR Flash Memory Under the Kernel .....	16
1.3 Porting an SPI NAND Flash Memory .....	20
1.3.1 Driver Path.....	20
1.3.2 Porting an SPI NAND Flash Memory .....	20
1.4 Porting a Parallel NAND Flash Memory .....	29
1.4.1 Driver Path.....	29
1.4.2 Parallel NAND ID System.....	29
1.4.3 Porting a Parallel NAND Flash Memory.....	30
<b>2 Common Issues .....</b>	<b>34</b>
2.1 3-Byte/4-Byte Boot of the SPI NOR Flash and 1-Wire/4-Wire Boot of the SPI NAND Flash .....	34
2.2 HiBurn Displays "Burn Success" but No Information Displayed Over the Serial Ports .....	35
2.3 4-Bit ECC and 8-Bit/1 KB ECC, 8-Bit ECC, and 16-Bit/1 KB ECC .....	36
2.4 What Should Be Paid Attention to When a Large-Capacity NAND Flash Is Used? .....	37
2.5 Why Cannot the System Start after U-Boot Saves Environment Variables in the NAND Flash?.....	37
2.6 How Do I Use the nandwrite Naked-Write Tool of mtd-utils? .....	37
2.7 What Do I Do If the Flash Compatibility Is Affected by Parameter Changes Caused by Updated Process but Unchanged ID of the Flash? .....	38



## Figures

<b>Figure 1-1</b> Registering the SPI NOR flash ID in the non-standardized driver .....	3
<b>Figure 1-2</b> 15 status register .....	9
<b>Figure 1-3</b> Bank address register .....	9
<b>Figure 1-4</b> Implementation of the quad-wire enable function of Macronix MX25U25635F .....	11
<b>Figure 1-5</b> Registering the SPI NOR flash ID in the non-standard driver .....	13
<b>Figure 1-6</b> Example of registration information .....	15
<b>Figure 1-7</b> Registering the SPI NOR flash ID in the standardized driver .....	17
<b>Figure 1-8</b> PARAMS(mxlc) description .....	19
<b>Figure 1-9</b> Registering the SPI NAND flash ID in the non-standardized driver .....	21
<b>Figure 1-10</b> Timing in dual I/O SPI Read mode .....	24
<b>Figure 1-11</b> Timing in quad I/O SPI Read mode .....	24
<b>Figure 1-12</b> Feature registers without the QE bit .....	27
<b>Figure 1-13</b> Feature registers with the QE bit .....	27
<b>Figure 1-14</b> ID information meaning of parallel NAND flash .....	30
<b>Figure 1-15</b> ID registration steps for parallel NAND flash .....	30
<b>Figure 2-1</b> 8-WSON package .....	34
<b>Figure 2-2</b> Comparison between 3-byte and 4-byte problem data of the SPI NOR flash .....	35
<b>Figure 2-3</b> Comparison between 4-wire problem data of the SPI NOR and SPI NAND flash .....	36
<b>Figure 2-4</b> Distribution of bad blocks .....	37
<b>Figure 2-5</b> Structure of NAND flash blocks .....	38
<b>Figure 2-6</b> MXIC MX25L6406E ID registry .....	39
<b>Figure 2-7</b> MXIC MX25L6436F ID registry .....	39
<b>Figure 2-8</b> Command difference between MT25Q series and N25Q series .....	40



## Tables

<b>Table 1-1</b> Directory structure related to the SPI NOR flash memories.....	2
<b>Table 1-2</b> SPI match table.....	4
<b>Table 1-3</b> Mapping between vendors and implementation functions .....	7
<b>Table 1-4</b> SPI interface mapping table.....	13
<b>Table 1-5</b> SPI match table.....	15
<b>Table 1-6</b> Directory structure related to the standardized SPI NOR flash memories.....	16
<b>Table 1-7</b> Mapping between vendors and the PARAMS macro definition of the SPI NOR flash.....	18
<b>Table 1-8</b> Directory structure related to the SPI NAND flash memories.....	20
<b>Table 1-9</b> SPI match table.....	22
<b>Table 1-10</b> AC characteristics table .....	25
<b>Table 1-11</b> Directory structure related to the ID porting of parallel NAND driver.....	29





# 1 Porting New SPI Flash Memories

---

## 1.1 Introduction

This document provides guidance on how to port new SPI flash memories based on the HiSilicon flash memory controller (HiFMC) V100 (FMC for short). The FMC has embedded three controllers for the SPI NOR flash, the SPI NAND flash, and the parallel NAND flash. In some platforms, the three controllers are clipped into a two-in-one controller for the SPI NOR flash and the SPI NAND flash.

- If the model of a selected SPI NOR flash memory is different from those listed in the SPI NOR flash compatibility list, you should add the flash ID node and implementation functions to the SPI NOR flash ID list. The modifications include read, write, and erase types, clock, ID, and the implementation functions (such as 4-byte addressing, quad-wire read/write enable, and reset function).
- If the model of a selected SPI NAND flash memory is different from those listed in the SPI NAND flash compatibility list, you should add the flash ID node and implementation functions to the SPI NAND flash ID list. The modifications include read and write types, clock size, ID, and quad-wire enable processing function.
- If the model of a selected parallel NAND flash memory is different from those listed in the parallel NAND flash compatibility list, you should first review the common flash list provided by the parallel NAND flash protocol layer. If the selected parallel NAND flash memory does not comply with the listed flash information, you should add the flash ID node to the special parallel NAND flash list.

## 1.2 Porting an SPI NOR Flash Memory

With the standardization of flash drivers, the SPI NOR flash driver under the kernel has been optimized to be compatible with the latest standardized driver architecture. However, the SPI NOR flash driver under U-Boot still uses the non-standardized versions.

Currently, two versions are available: u-boot-2010.06 and u-boot-2016.11. The difference is that, in u-boot-2016.11, you can access to the SPI NOR flash memories larger than 16 MB by using the 4-byte command. For details, see section [1.2.1.4 "Porting an SPI NOR Flash Memory Larger Than 16 MB in the u-boot-2016.11 SPI NOR Flash Driver."](#)



## 1.2.1 Porting an SPI-NOR Flash Memory Under U-Boot

### 1.2.1.1 Driver Path

The SPI NOR flash driver under U-Boot uses the non-standardized version currently.

The driver path is **drivers/mtd/spi/hifmc100/**.

[Table 1-1](#) lists the directories related to the component porting under the non-standardized SPI NOR flash drivers.

**Table 1-1** Directory structure related to the SPI NOR flash memories

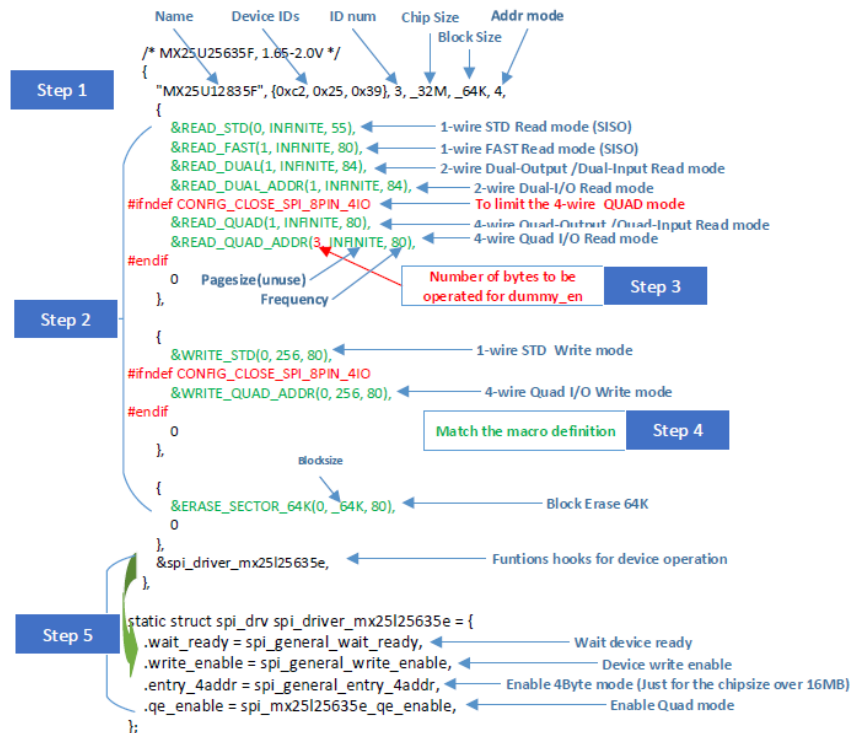
HiFMC V100 Directory/File Name	Description
hifmc_spi_nor_ids.c	SPI NOR flash ID list. It is the key file to be modified for component porting. It also lists the read, write, and erase parameters and the hook value of the implementation functions of the SPI NOR flash memories.
hifmc100_spi_general.c	A file containing universal function driver codes for most SPI NOR flash memories. This file supports write enable, 4-byte addressing, and quad-wire read/write operations. However, the operations apply only to the SPI NOR flash memories provided by some vendors. For special SPI NOR flash memories, implementation functions must be added based on user manuals.
<ul style="list-style-type: none"><li>hifmc100_spi_gd25qxxx.c</li><li>hifmc100_spi_micron.c</li><li>hifmc100_spi_mx25l25635e.c</li><li>hifmc100_spi_s25fl256s.c</li><li>hifmc100_spi_w25q256fv.c</li></ul>	The drivers in <b>hifmc100_spi_general.c</b> do not support some implementation functions for special SPI NOR flash memories. Specific drivers must be added to realize the special functions including 4-byte addressing (for the SPI flash with 32 MB or larger capacity), quad-wire read/write, and the Quad Enable (QE) bit enable function.

### 1.2.1.2 Porting an SPI NOR Flash Memory

Porting an SPI NOR flash memory is realized by adding a new ID node to the ID registration structure in **hifmc\_spi\_nor\_ids.c**.

In this section, Macronix MX25U25635F is used as an example to describe how to port a new SPI NOR flash memory.

**Figure 1-1** Registering the SPI NOR flash ID in the non-standardized driver



**Step 1** Refer to the MX25U25635F user manual and add an ID node by filling the ID information in the corresponding position.

Obtain the **ID** information under the **9Fh** command.

As shown in the figure below, this component has three IDs: 0xC2, 0x20, and 0x19.

**Table 6. ID Definitions**

Command Type		MX25L25635F		
RDID	9Fh	Manufactory ID	Memory type	Memory density
		C2	20	19

Obtain the chip size and block size:

- 256Mb: 268,435,456 x 1 bit structure or 134,217,728 x 2 bits (two I/O mode) structure or 67,108,864 x 4 bits (four I/O mode) structure
- Equal Sectors with 4K byte each, or Equal Blocks with 32K byte each or Equal Blocks with 64K byte each

**Step 2** Obtain the SPI types that the component supports from the **Features** chapter in the user manual:



- Fast read for SPI mode
  - Support clock frequency up to 133MHz for all protocols
  - Support Fast Read, 2READ, DREAD, 4READ, QREAD instructions.
  - Configurable dummy cycle number for fast read operation
- Quad Peripheral Interface (QPI) available
- Equal Sectors with 4K byte each, or Equal Blocks with 32K byte each or Equal Blocks with 64K byte each
  - Any Block can be erased individually

Refer to [Table 1-2](#) for the SPI types that MX25U25635F supports and their corresponding macro definitions in the driver. The results are marked with a gray background.

Note that the 64-KB block erase command is matched with the erase operation by default.

**Table 1-2** SPI match table

SPI Supported	Command Word	Corresponding FMC SPI	Macro Definition in the Driver
READ	<b>03h</b>	Standard SPI	<b>READ_STD</b>
FAST READ	<b>0Bh</b>	Standard SPI	<b>READ_FAST</b>
2READ	<b>3Bh</b>	Dual-output/dual-input SPI	<b>READ_DUAL</b>
DREAD	<b>BBh</b>	Dual I/O SPI	<b>READ_DUAL_ADDR</b>
4READ	<b>6Bh</b>	Quad-output/quad-input SPI	<b>READ_QUAD</b>
QREAD	<b>EBh</b>	Quad I/O SPI	<b>READ_QUAD_ADDR</b>
PP	<b>02h</b>	Standard SPI	<b>WRITE_STD</b>
DPP (page program)	<b>A2h</b>	Dual-output/dual-input SPI	<b>WRITE_DUAL</b>
2PP	<b>D2h</b>	Dual I/O SPI	<b>WRITE_DUAL_ADDR</b>
QPP	<b>32h</b>	Quad-output/quad-input SPI	<b>WRITE_QUAD</b>
4PP	<b>38h</b>	Quad I/O SPI	<b>WRITE_QUAD_ADDR</b>
Block Erase 4 KB	<b>02h</b>	Standard SPI	<b>ERASE_SECTOR_4K</b>
Block Erase 32 KB	<b>52h</b>	Standard SPI	<b>ERASE_SECTOR_32K</b>
Block Erase 64 KB	<b>D8h</b>	Standard SPI	<b>ERASE_SECTOR_64K</b>
Block Erase 128 KB	<b>D8h</b>	Standard SPI	<b>ERASE_SECTOR_128K</b>



SPI Supported	Command Word	Corresponding FMC SPI	Macro Definition in the Driver
Block Erase 256 KB	<b>D8h</b>	Standard SPI	<b>ERASE_SECTOR_256K</b>

### Step 3 Specify the dummy value and working clock.

- Specifying the dummy value:

The SPI definitions of the SPI NOR Flash show that:

- The dummy value of the standard SPI STD read, and all write and erase SPIs is **0**.
- The dummy value of the dual-output/dual-input SPIs and quad-output/quad-input SPIs is **1**.
- The dummy values of the dual I/O SPIs and quad I/O SPIs should be calculated according to the user manual.



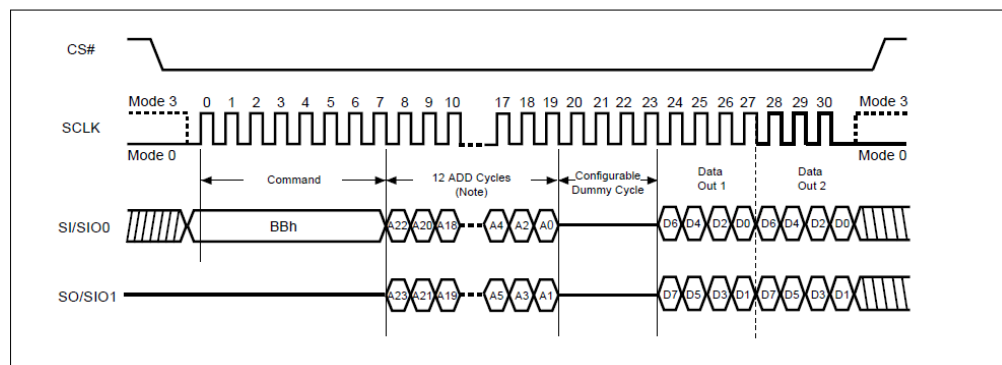
#### NOTE

The I/O driver strength (see the *Output Driver Strength Table*) has to be taken into the calculation of the dummy value for some components. For simplicity, you are advised to omit the dual I/O SPIs (2READ/2PP) and quad I/O SPIs (4READ/4PP) during the SPI porting. These two types of SPIs, comparing with the dual-output/dual-input SPIs and quad-output/quad-input SPIs, make little performance difference in scenarios where high performance is not strictly necessary.

This section describes how to calculate the dummy value when the default DC value is **0x00**.

#### a) Dual I/O SPI Read mode

Figure 30. 2 x I/O Read Mode Sequence (SPI Mode only)

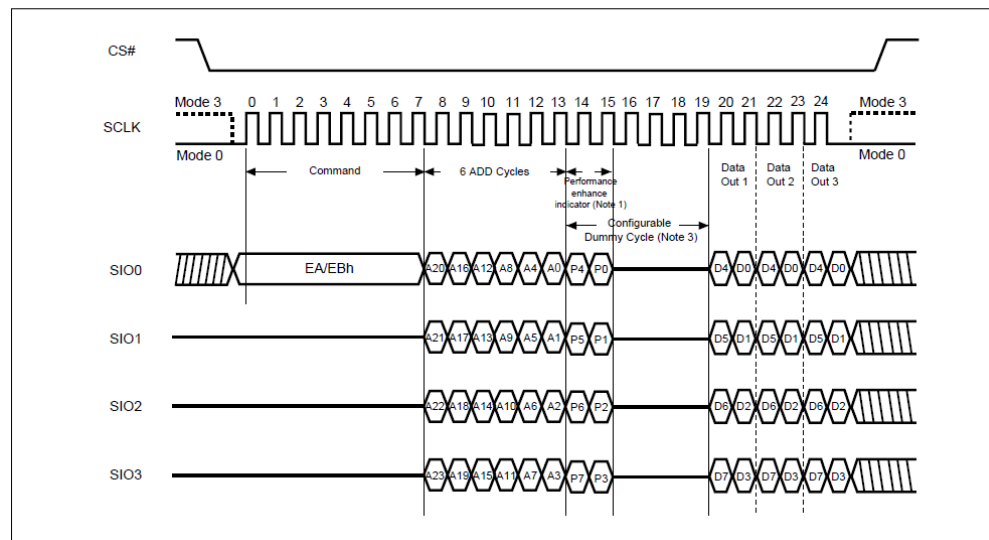


As shown in the waveforms excerpted from the user manual, the dual I/O SPI Read mode requires four dummy cycles (20–23). Since the SPI is a 2-wire interface, four dummy cycles take eight dummy cycle bits, or one dummy cycle byte. According to `dummy_num` defined in the chip data sheet, the dummy value in `&READ_DUAL_ADDR(1, INFINITE, 84)` should be **1**.

#### b) Quad I/O SPI Read mode



Figure 32. 4 x I/O Read Mode Sequence (SPI Mode)



As shown in the waveforms excerpted from the user manual, the quad I/O SPI Read mode requires two cycles (14–15) for performance enhancement (or M7–M0 because of vendor difference) and four dummy cycles (16–19), which are six cycles in total. Since the SPI is a 4-wire interface, six cycles take 24 dummy cycle bits, or three dummy cycle bytes. According to dummy\_num defined in the chip data sheet, the dummy value in &READ\_DUAL\_ADDR(3, INFINITE, 80) should be 3.

- SPI working clock

The clock frequency value in &READ\_QUAD\_ADDR(3, INFINITE, 80) is obtained from the AC characteristics table in the user manual.

Table 16. AC CHARACTERISTICS (Temperature = -40°C to 85°C, VCC = 2.7V ~ 3.6V)

Symbol	Alt.	Parameter	Min.	Typ.	Max.	Unit
fSCLK	fC	Clock Frequency for all commands (except Read)	D.C.		133	MHz
fRCLK	fR	Clock Frequency for READ instructions			50	MHz
fTCLK	fT	Clock Frequency for 2READ instructions			84 <sup>(7)</sup>	MHz
	fQ	Clock Frequency for 4READ instructions			84 <sup>(7)</sup>	MHz

## NOTICE

As shown in the table above, the maximum clock frequency supported is 84 MHz or 133 MHz, which is different from the clock information of the MX25U25635F ID node. **The reason is that the maximum frequency of the 1.8 V I/O SPI clock supported by the current FMC is 150 MHz (or 75 MHz).** 80 MHz is written here to limit the SPI clock frequency and match the macro definitions in step 4 as well.

**Step 4** Once the SPI information is filled out as the figure shown below, you may begin to match the macro definition.



```
{
    &READ_STD(0, INFINITE, 55),
    &READ_FAST(1, INFINITE, 80),
    &READ_DUAL(1, INFINITE, 84),
    &READ_DUAL_ADDR(1, INFINITE, 84),
#ifdef CONFIG_CLOSE_SPI_8PIN_4IO
    &READ_QUAD(1, INFINITE, 80),
    &READ_QUAD_ADDR(3, INFINITE, 80),
#endif
    0
},

{
    &WRITE_STD(0, 256, 80),
#ifdef CONFIG_CLOSE_SPI_8PIN_4IO
    &WRITE_QUAD_ADDR(0, 256, 80),
#endif
    0
},

{
    &ERASE_SECTOR_64K(0, _64K, 80),
    0
},
},
```

Match the macro definition in the beginning lines of the **hifmc\_spi\_nor\_ids.c** source code path.

```
SET_READ_STD(0, INFINITE, 66);
SET_READ_FAST(1, INFINITE, 80);
SET_READ_DUAL(1, INFINITE, 84);
SET_READ_DUAL_ADDR(1, INFINITE, 84);
SET_READ_QUAD(1, INFINITE, 80);
SET_READ_QUAD_ADDR(3, INFINITE, 80);
SET_WRITE_STD(0, 256, 80);
SET_WRITE_QUAD(0, 256, 80);
SET_ERASE_SECTOR_64K(0, 64K, 80);
```

**Step 5** Match the **&spi\_driver\_mx25l25635e** hook (the struct **spi\_drv** structure) of the implementation functions. The following interface values require attentions:

```
.wait_ready = spi_general_wait_ready,
.write_enable = spi_general_write_enable,
.entry_4addr = spi_general_entry_4addr,
.qe_enable = spi_mx25l25635e_qe_enable,
```

The current driver is compatible with the implementation functions used by most SPI NOR flash vendors. If the new component to be ported is listed in [Table 1-3](#), you may use the value listed directly to save troubles.

**Table 1-3** Mapping between vendors and implementation functions

Vendor	struct spi_drv Value	Source File
GigaComponent (GD)	&spi_driver_gd25qxxx	hifmc100_spi_gd25qxxx.c
Micron	&spi_driver_micron	hifmc100_spi_micron.c
MXIC (Macronix)	&spi_driver_mx25l25635e	hifmc100_spi_mx25l25635e.c





Vendor	struct spi_drv Value	Source File
Spansion	&spi_driver_s25fl256s	hifmc100_spi_s25fl256s.c
Winbond	&spi_driver_w25q256fv	hifmc100_spi_w25q256fv.c
Others (universally applicable)	&spi_driver_general	hifmc100_spi_general.c

- **wait\_ready = spi\_general\_wait\_ready** is used to wait for the component to be ready.

Check whether the component is busy by reading the WIP bit of the component status register.

Normally, this mechanism is applicable to all SPI NOR flash memories.

**WIP bit.** The Write in Progress (WIP) bit, a volatile bit, indicates whether the device is busy in program/erase/write status register progress. When WIP bit sets to 1, which means the device is busy in program/erase/write status register progress. When WIP bit sets to 0, which means the device is not in progress of program/erase/write status register cycle.

- **.write\_enable = spi\_general\_write\_enable** is the write enable SPI.

Set whether the component is operationable by writing the WEL bit of the status register.

Normally, this mechanism is applicable to all SPI NOR flash memories.

**WEL bit.** The Write Enable Latch (WEL) bit, a volatile bit, indicates whether the device is set to internal write enable latch. When WEL bit sets to 1, which means the internal write enable latch is set, the device can accept program/erase/write status register instruction. When WEL bit sets to 0, which means no internal write enable latch; the device will not accept program/erase/write status register instruction. The program/erase command will be ignored if it is applied to a protected memory area. To ensure both WIP bit & WEL bit are both set to 0 and available for next program/erase/operations, WIP bit needs to be confirm to be 0 before polling WEL bit. After WIP bit confirmed, WEL bit needs to be confirm to be 0.

#### NOTE

Although different vendors may have different naming systems for the component status registers, the command words for accessing the registers are the same. Therefore, the command words are used to differentiate the status registers of SPI NOR flash: the 05 status register, the 35 status register, and the 15 status register.

As mentioned above, the WIP bit and WEL bit are located at bit 0 and bit 1 of the 05 status register, respectively. Their positions are the same for all SPI NOR flash memories.

**Status Register**

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
SRWD (status register write protect)	QE (Quad Enable)	BP3 (level of protected block)	BP2 (level of protected block)	BP1 (level of protected block)	BP0 (level of protected block)	WEL (write enable latch)	WIP (write in progress bit)
1=status register write disable	1=Quad Enable 0=not Quad Enable	(note 1)	(note 1)	(note 1)	(note 1)	1=write enable 0=not write enable	1=write operation 0=not in write operation
Non-volatile bit	Non-volatile bit	Non-volatile bit	Non-volatile bit	Non-volatile bit	Non-volatile bit	volatile bit	volatile bit

However, different vendors and components differ greatly in the 35 status registers and 15 status registers (involving the 3-byte/4-byte mode switch and the QE bit enable).

----End





### 1.2.1.3 Special Topic: 3-Byte/4-Byte Mode Switch and QE Bit Enable

This section describes the `.entry_4addr = spi_general_entry_4addr` and `qe_enable = spi_mx25l25635e_qe_enable` interfaces.



#### NOTE

When porting an SPI flash memory, you should adapt the SPI for 3-byte/4-byte mode switch based on the application scenario.

- `.entry_4addr = spi_general_entry_4addr` is used to switch the 3-byte/4-byte mode. This SPI is called only when **the component capacity is larger than 16 MB**.

Generally, SPI NOR flash memories provide enter 4-byte mode (EN4B) and exit 4-byte mode (EX4B) instructions to realize the 3-byte/4-byte mode switch. Bit 5 of the 15 status register is used to indicate whether the switch operation succeeds.

**Figure 1-2** 15 status register

**Configuration Register**

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
DC1 (Dummy cycle 1)	DC0 (Dummy cycle 0)	4 BYTE	Reserved	TB (top/bottom selected)	ODS 2 (output driver strength)	ODS 1 (output driver strength)	ODS 0 (output driver strength)
(note 2)	(note 2)	0=3-byte address mode 1=4-byte address mode (Default=0)	x	0=Top area protect 1=Bottom area protect (Default=0)	(note 1)	(note 1)	(note 1)
volatile bit	volatile bit	volatile bit	x	OTP	volatile bit	volatile bit	volatile bit

The differences of other components in 3-byte/4-byte mode switch come in three different ways.

- The 3-byte/4-byte mode switch methods are different.

A typical example is `.entry_4addr = spi_s25fl256s_entry_4addr` from Spansion (now Cypress). It realizes the 3-byte/4-byte mode switch by changing the value of bit 7 in the bank address register.

**Figure 1-3** Bank address register

**Table 8.16** Bank Address Register (BAR)

Bits	Field Name	Function	Type	Default State	Description
7	EXTADD	Extended Address Enable	Volatile	0b	1 = 4-byte (32-bits) addressing required from command. 0 = 3-byte (24-bits) addressing from command + Bank Address
6 to 1	RFU	Reserved	Volatile	00000b	Reserved for Future Use
0	BA24	Bank Address	Volatile	0	A24 for 256-Mbit device, RFU for lower density device

- The 3-byte/4-byte mode switch is realized by instructions while the register indicating whether the switch operation succeeds is not the 15 status register. A typical example is the `.entry_4addr = spi_micron_entry_4addr` SPI of the SPI NOR flash from Micron. Bit 0 of the flag status register is used as the indicator instead.



- The operation of switching back to the 3-byte mode from the 4-byte mode is not realized by the EX4B instruction but the reset command. A typical example is the W25Q256 series from Winbond.

The `.entry_4addr = spi_w25q256fv_entry_4addr` SPI directly sends reset group commands to switch back to the 3-byte mode from the 4-byte mode.

- The `.qe_enable = spi_mx25l25635e_qe_enable` SPI is the quad-wire enable function interface.

---

## NOTICE

In the driver, the `CONFIG_CLOSE_SPI_8PIN_4IO` macro is used to open or close the quad-wire SPI. The quad-wire SPI in many of the Broadband Video Terminal (BVT) platforms is not open by default, because the `IO_3` pin is multiplexed with the reset pin. The default function is reset.

Components from different vendors may have different distributions of the QE bit. Such differences can only be analyzed based on the user manuals.

Macronix MX25U25635F is used as an example to describe the function implementation of this SPI.

**Figure 1-4** Implementation of the quad-wire enable function of Macronix MX25U25635F

```

static int spi_mx25l25635e_qe_enable(struct hifmc_spi *spi)
{
    unsigned char status, op;
    unsigned int regval;
    const char *str[] = {"Disable", "Enable"};
    struct hifmc_host *host = (struct hifmc_host *)spi->host;

    op = spi_is_quad(spi); ← Check whether the 4-wire mode should be supported

    status = spi_general_get_flash_register(spi, SPI_CMD_RDSR);
    if (MX_SPI_NOR_GET_QE_BY_SR(status) == op) {} ← Tag 1
    return op;
} ← Get the status register to check the QE bit value

spi->driver->write_enable(spi);

if (op)
    status |= MX_SPI_NOR_SR_QE_MASK; ← Turn on QE
else
    status &= ~MX_SPI_NOR_SR_QE_MASK; ← Turn off QE
writeb(status, host->iobase); ← Tag 2
regval = FMC_CMD_CMD1(SPI_CMD_WRSR);
hifmc_write(host, FMC_CMD, regval);
regval = OP_CFG_FM_CS(spi->chipselect);
hifmc_write(host, FMC_OP_CFG, regval);
regval = FMC_DATA_NUM_CNT(SPI_NOR_SR_LEN);
hifmc_write(host, FMC_DATA_NUM, regval);
regval = FMC_OP_CMD1_EN(ENABLE)
    | FMC_OP_WRITE_DATA_EN(ENABLE)
    | FMC_OP_REG_OP_START;
hifmc_write(host, FMC_OP, regval);
FMC_CMD_WAIT_CPU_FINISH(host);
spi->driver->wait_ready(spi);
status = spi_general_get_flash_register(spi, SPI_CMD_RDSR); ← Check whether the
if (MX_SPI_NOR_GET_QE_BY_SR(status) == op) ← Tag 3 setting is successful
    FMC_PR(QE_DBG, "\t| -%s Quad success, status: %#x.\n", str[op],
        status);
else
    DB_MSG("Error: %s Quad failed! reg: %#x\n", str[op], status);
return op;
}

```

The flow of writing back to status register above.  
Note: there is an one-to-one correspondence command between the read and write status register, please refer to the datasheet.

If you need to add a new QE enable SPI, you may refer to the code realization in [Figure 1-4](#). You do not need to modify the code flow while the following tags require attentions.

- tag1: This command is used to read the component register where the QE bit is located.
- tag2: This command is used to write the component register where the QE bit is located.
- tag3: This command is used to indicate whether the QE bit is set successfully.

### 1.2.1.4 Porting an SPI NOR Flash Memory Larger Than 16 MB in the u-boot-2016.11 SPI NOR Flash Driver

The HiFMC V100 is optimized for the chip with the u-boot-2016.11 SPI NOR flash driver, so that the restriction on the hardware-based connection to the reset pin is removed. However, the support of the 4-byte command must be configured for the components larger than 16 MB.



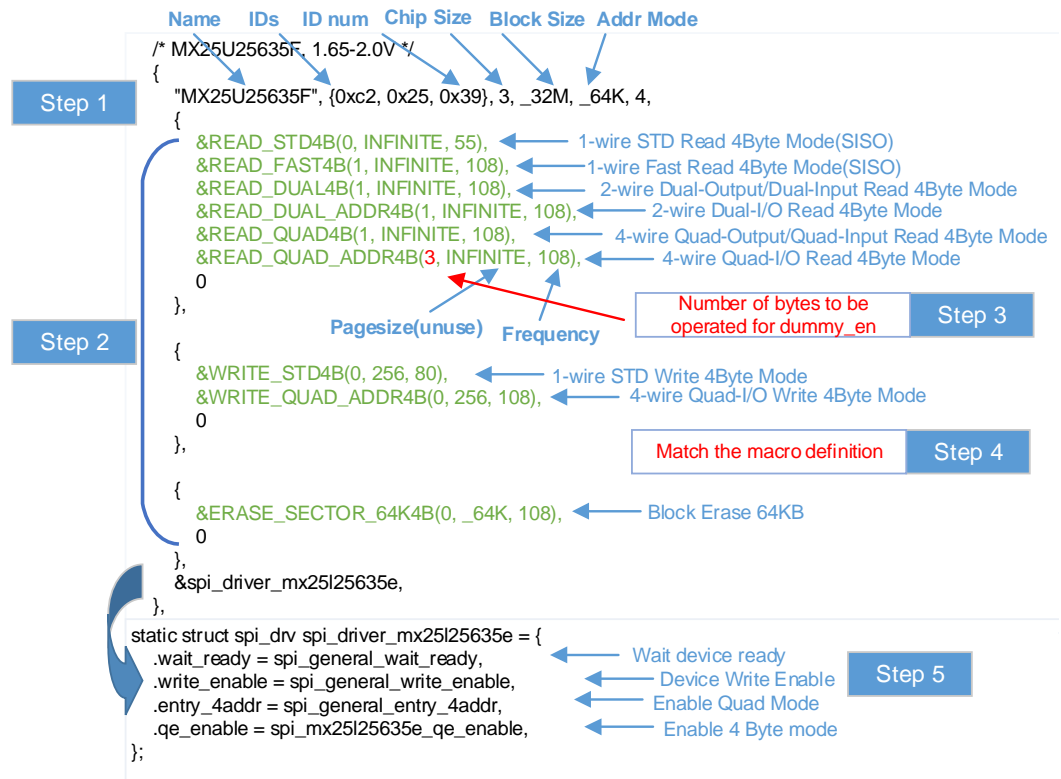
Currently, the driver adapts only to three vendors: Winbond, Macronix (MXIC), and Micron. To adapt to the components that support the 4-byte command from other vendors, add the corresponding case code branch to the following code segment in the directory **drivers/mtd/spi/hifmc100/hifmc\_spi\_nor\_ids.c**.

```
if ((spi->addrcycle == SPI_NOR_4BYTE_ADDR_LEN)
    && (start_up_addr_mode == SPI_NOR_ADDR_MODE_3_BYTES)) {
    switch (ids[0]) {
        case MID_WINBOND:
        case MID_MXIC:
        case MID_MICRON:
            FMC_PR(BT_DBG, "\t|||-4-Byte Command Operation\n");
            break;
        default:
            FMC_PR(BT_DBG, "\t|||-start up: 3-Byte mode\n");
            spi->driver->entry_4addr(spi, ENABLE);
            break;
    }
} else
    FMC_PR(BT_DBG, "\t|||-start up: 4-Byte mode or 4-Byte Command\n");
```

## NOTICE

If the SPI NOR flash memory (larger than 16 MB) to be ported does not support the 4-byte command, perform the porting by following the steps specified in section [1.2.1.2 "Porting an SPI NOR Flash Memory"](#). Note that only the 2-wire interface with hardware-based connection to the reset pin can be used.

The following uses Macronix MX25U25635F as an example to describe how to register a SPI NOR flash that supports the 4-byte command in the ID table. The only difference in porting a common SPI NOR flash memory described in section [1.2.1.2 "Porting an SPI NOR Flash Memory"](#) is Step 2. Therefore, steps 1, 3, 4, and 5 are omitted here.

**Figure 1-5** Registering the SPI NOR flash ID in the non-standard driver

Check whether the 4-byte command is supported by refereeing to the command set section in the Macronix MX25U25635F data sheet.

**Table 6. Read/Write Array Commands (4 Byte Address Command Set)**

Command (byte)	READ4B	FAST READ4B	2READ4B	DREAD4B	4READ4B	QREAD4B	4DTRD4B (Quad I/O DT Read)
Mode	SPI	SPI	SPI	SPI	SPI/QPI	SPI	SPI/QPI
Address Bytes	4	4	4	4	4	4	4

By querying [Table 1-4](#), you can find the interfaces supported by MX25U25635F and the macro definitions in the drivers, as shown in the text highlighted in gray.

In addition, the **Block Erase 64K4B** command is the default command for the erase operation.

**Table 1-4** SPI interface mapping table

Interfaces Supported by the Component	Command Word	Corresponding FMC Interface	Macro Definition in the Driver
READ4B	13h	Standard SPI	READ_STD4B
FAST READ4B	0Ch	Standard SPI	READ_FAST4B
2READ4B	3Ch	Dual-Output/Dual-Input SPI	READ_DUAL4B



Interfaces Supported by the Component	Command Word	Corresponding FMC Interface	Macro Definition in the Driver
DREAD4B	<b>BCh</b>	Dual I/O SPI	<b>READ_DUAL_ADDR4B</b>
4READ4B	<b>6Ch</b>	Quad-Output/Quad-Input SPI	<b>READ_QUAD4B</b>
QREAD4B	<b>EBh</b>	Quad I/O SPI	<b>READ_QUAD_ADDR4B</b>
PP4B	<b>12h</b>	Standard SPI	<b>WRITE_STD4B</b>
4PP4B	<b>3Eh</b>	Quad I/O SPI	<b>WRITE_QUAD_ADDR4B</b>
BE4K4B Block Erase 4K4B	<b>21h</b>	Standard SPI	<b>ERASE_SECTOR_4K4B</b>
BE32K4B Block Erase 32K	<b>5Ch</b>	Standard SPI	<b>ERASE_SECTOR_324B K</b>
BE4B Block Erase 64K	<b>DCh</b>	Standard SPI	<b>ERASE_SECTOR_64K4B</b>

### 1.2.1.5 Porting an SPI NOR Flash Memory in DTR Mode

This section uses Macronix MX25L25645G as an example to describe how to port an SPI NOR flash memory that supports the double transfer rate (DTR) mode.

#### NOTICE

Only a DTR-capable flash memory supports adaption to the DTR mode. A flash memory in DTR mode supports only write operations.

- The DTR interface is enabled and disabled by using the macro `CONFIG_DTR_MODE_SUPPORT` in the driver. By default, the DTR interface is disabled on many BVT platforms. To use the DTR mode, ensure that the four-wire interface is enabled. In u-boot-2010.06, the macro that controls the enabling of the DTR mode is stored in `include/configs/hi35xx.h`.
- In u-boot-2016.11, the macro that controls the enabling of the DTR mode can be enabled by choosing **Device Drivers-> SPI Flash Support-> Hisilicon Spi Nor Device DTR mode Support** in menuconfig.

The adaption to the DTR mode is different from that to the STR mode. For details, see the part marked in the red box in [Figure 1-6](#).

**Figure 1-6** Example of registration information

```
{
    "MX25L(256/257)XX",
    {0xc2, 0x20, 0x19}, 3, _32M, _64K, 4,
    {
        &READ_STD(0, INFINITE, 40/*50*/),
        &READ_FAST(1, INFINITE, 104),
        &READ_DUAL(2, INFINITE, 104),
        &READ_DUAL_ADDR(1, INFINITE, 84),
#ifdef CONFIG_CLOSE_SPI_8PIN_4IO
        &READ_QUAD_ADDR(3, INFINITE, 75),
#endif
#ifdef CONFIG_DTR_MODE_SUPPORT
        &READ_QUAD_DTR(10, INFINITE, 84),
#endif
        0
    },
    {
        &WRITE_STD(0, 256, 75),
#ifdef CONFIG_CLOSE_SPI_8PIN_4IO
        &WRITE_QUAD_ADDR(0, 256, 104),
#endif
        0
    },
    {
        &ERASE_SECTOR_64K(0, _64K, 80),
        0
    },
    &spi_driver_mx25l25635e,
},
```

The procedure of porting an SPI NOR flash memory in DTR mode is similar to that described in section 1.2.1.2 "Porting an SPI NOR Flash Memory". The following describes the differences between the procedure of porting an SPI NOR flash memory in DTR mode and that described in Step 2 in section 1.2.1.2 "Porting an SPI NOR Flash Memory".

Obtain the SPI types supported by the flash memory from the chapter titled "**Features**" in the user manual provided by the vendor:

- Fast read for SPI mode
  - Support clock frequency up to 133MHz for all protocols
  - Support Fast Read, 2READ, DREAD, 4READ, QREAD instructions
  - Support DTR (Double Transfer Rate) Mode
  - Configurable dummy cycle number for fast read operation

Table 1-5 describes the SPI types that MX25U25645G DTR supports and their macro definitions in the driver.

**Table 1-5** SPI match table

SPI Supported	Command Word	Corresponding FMC SPI	Macro Definition in the Driver
4DTRD4B	<b>EEh</b>	DTR Fast Read Quad I/O	<b>READ_QUAD_DTR</b>
4DTRD	<b>EDh</b>	DTR Fast Read Quad I/O	<b>READ_QUAD_DTR_WINB OND</b>



## 1.2.2 Porting an SPI-NOR Flash Memory Under the Kernel

### 1.2.2.1 Driver Path

The SPI NOR flash driver under the kernel uses the standardized version currently.

The driver path is **drivers/mtd/spi-nor/**.

[Table 1-6](#) describes the directory related to the component porting under the standardized SPI NOR flash drivers.

**Table 1-6** Directory structure related to the standardized SPI NOR flash memories

HiFMC V100 Directory/File Name	Description
spi-nor.c	Standard driver architecture. It defines the main data structure of the SPI NOR flash driver and the SPI NOR component ID list. It is the key file to be modified for component porting. It also differentiates the 3-byte/4-byte mode switch and quad-wire enable operations of different components.

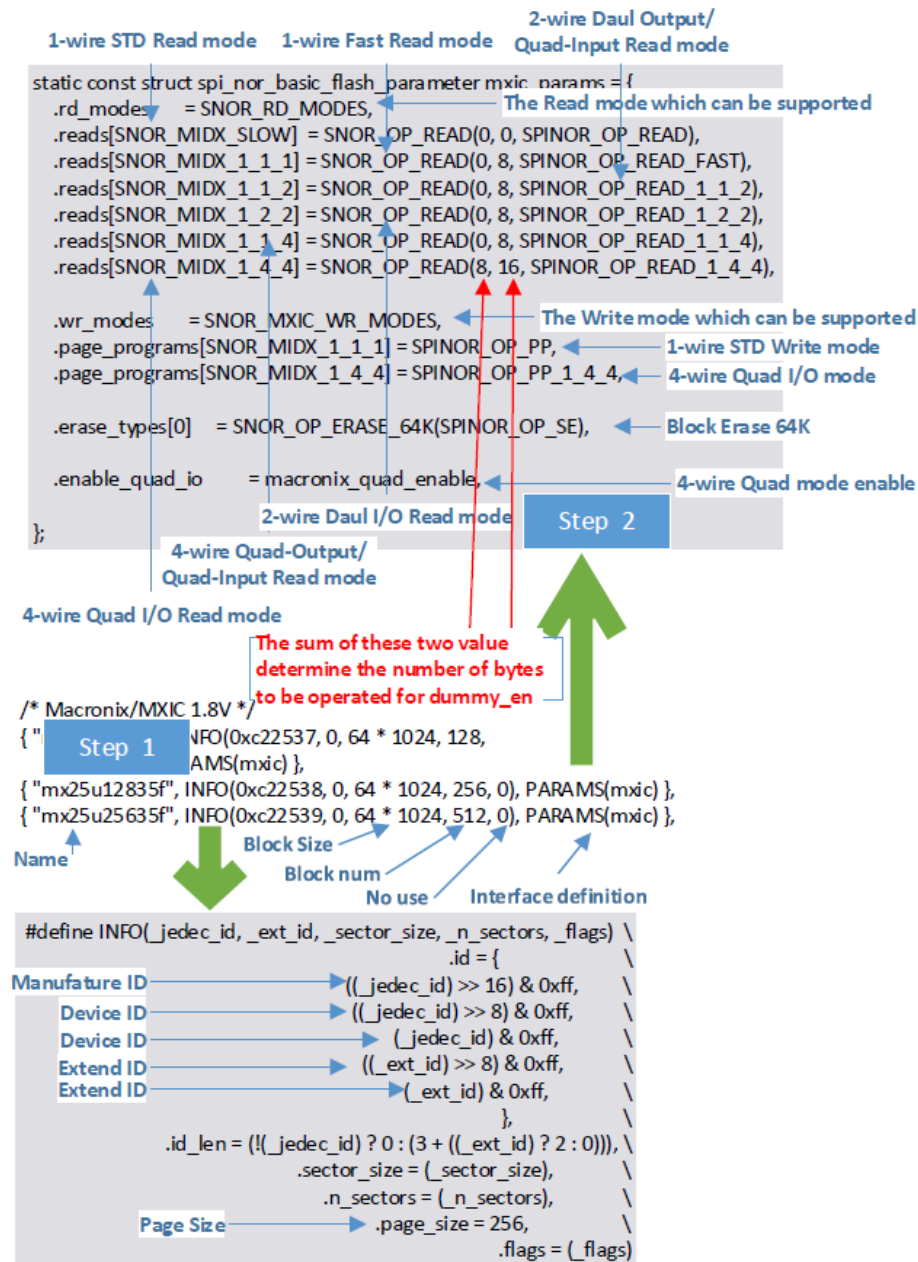
### 1.2.2.2 Porting an SPI NOR Flash Memory

Porting an SPI NOR flash memory is realized by adding a new ID node to the ID registration structure in **spi-nor.c**.

In this section, Macronix MX25U25635F is used as an example to describe how to port a new SPI NOR flash memory.



**Figure 1-7** Registering the SPI NOR flash ID in the standardized driver



## NOTICE

Unlike the SPI NOR flash under the non-standardized driver where different components produced by one vendor can have different interfaces, the interface adaption only applies to the vendor level for the SPI NOR flash under the standardized driver. If different components produced by one vendor require different interface types, you may have to close some of the interfaces for compatibility purpose.

**Step 1** Refer to the MX25U25635F user manual and add an ID node by filling the ID information in the corresponding position.



Obtain the ID information under the **9Fh** command. As shown in the figure below, this component has three IDs: 0xC2, 0x20, and 0x19.

**Table 6. ID Definitions**

Command Type		MX25L25635F		
RDID	9Fh	Manufactory ID	Memory type	Memory density
		C2	20	19

Obtain the **chip size and block size**. They are 32 MB and 64 KB, respectively.

- 256Mb: 268,435,456 x 1 bit structure or 134,217,728 x 2 bits (two I/O mode) structure or 67,108,864 x 4 bits (four I/O mode) structure
- Equal Sectors with 4K byte each, or Equal Blocks with 32K byte each or Equal Blocks with 64K byte each

**Step 2** The FMC has specified the SPI types, dummy values, 3-byte/4-byte mode switch, and QE bit enable for the SPI NOR flash memories by most vendors. If the new component is listed in the following table, use the PARAMS macro definition specified by the vendor directly.

**Table 1-7** Mapping between vendors and the PARAMS macro definition of the SPI NOR flash

Vendor	Macro Definition
ESMT	PARAMS(esmt)
GigaDevice	PARAMS(gd)
Macronix/MXIC	PARAMS(mxic)
Micron	PARAMS(micron)
Spansion	PARAMS(spansion)
Winbond	PARAMS(winbond)
Paragon	PARAMS(paragon)

**Step 3** Specify the working clock.

## NOTICE

The value of the working clock of the default version release place has passed the compatibility test. Therefore, it is not advised to change this value.

Obtain the component working clock from the AC characteristics table in the user manual. Select the clock frequency based on the SPI to be used.

**Table 16. AC CHARACTERISTICS** (Temperature = -40°C to 85°C, VCC = 2.7V ~ 3.6V)

Symbol	Alt.	Parameter	Min.	Typ.	Max.	Unit
fSCLK	fC	Clock Frequency for all commands (except Read)	D.C.		133	MHz
fRCLK	fR	Clock Frequency for READ instructions			50	MHz
fTCLK	fT	Clock Frequency for 2READ instructions			84 <sup>(7)</sup>	MHz
	fQ	Clock Frequency for 4READ instructions			84 <sup>(7)</sup>	MHz

**NOTE**

All the components in the compatibility list support FAST READ (specified by **m25p,fast-read** in the DTS file). Therefore, the maximum clock frequency 50 MHz supported by the READ instructions (corresponding to STD READ) in the preceding figure can be ignored.

From the component data sheet, we can tell that the maximum clock frequency supported by the component is 133 MHz, but the 2READ/4READ interfaces can support up to 84 MHz only. Because this component supports PARAMS(mx) interfaces, that is, this component supports all types of interfaces. Therefore, the 84 MHz clock frequency should be adopted, as shown in the following figure: directory: **drivers/mtd/spi-nor/spi-nor.c**

```
{ "mx25u25635f", INFO(0xc22539, 0, 64 * 1024, 512,
    SPI_NOR_QUAD_READ | SPI_NOR_4B_OPCODES), PARAMS(mx), CLK_MHZ_2X(84) },
```

**NOTE**

As shown in Figure 1-8, the PARAMS(mx) interfaces supported by the MXIC component include the 1-wire (STD and FAST READ), 2-wire (dual-output/dual-input and dual-I/O), and 4-wire (quad-output/quad-input and quad-I/O) interfaces. When the PARAMS(mx) parameter in the ID node is set to the default value, the interface type supported by the component is specified by

**SPI\_NOR\_QUAD\_READ/SPI\_NOR\_DUAL\_READ** in the node. Therefore, when a newly added component cannot support all interfaces, you can use

**SPI\_NOR\_QUAD\_READ/SPI\_NOR\_DUAL\_READ** to specify the supported interface types.

**Figure 1-8** PARAMS(mx) description

```
static const struct spi_nor_basic_flash_parameter mxic_params = {
    .rd_modes      = SNOR_RD_MODES,
    .reads[SNOR_MIDX_SLOW] = SNOR_OP_READ(0, 0, SPINOR_OP_READ), ← 1-wire STD read mode (SISO)
    .reads[SNOR_MIDX_1_1_1] = SNOR_OP_READ(0, 8, SPINOR_OP_READ_FAST), ← 1-wire fast read mode (SISO)
    .reads[SNOR_MIDX_1_1_2] = SNOR_OP_READ(0, 8, SPINOR_OP_READ_1_1_2), ← 2-wire dual-output/dual-input read mode
    .reads[SNOR_MIDX_1_2_2] = SNOR_OP_READ(0, 8, SPINOR_OP_READ_1_2_2), ← 2-wire dual-I/O read mode
    .reads[SNOR_MIDX_1_1_4] = SNOR_OP_READ(0, 8, SPINOR_OP_READ_1_1_4), ← 4-wire quad-output/quad-input read mode
    .reads[SNOR_MIDX_1_4_4] = SNOR_OP_READ(8, 16, SPINOR_OP_READ_1_4_4), ← 4-wire quad-I/O read mode

    .wr_modes      = SNOR_MXIC_WR_MODES,
    .page_programs[SNOR_MIDX_1_1_1] = SPINOR_OP_PP, ← 1-wire STD write mode
    .page_programs[SNOR_MIDX_1_4_4] = SPINOR_OP_PP_1_4_4, ← 4-wire quad-I/O write mode

    .erase_types[0] = SNOR_OP_ERASE_64K(SPINOR_OP_SE), ← Block erase 64 KB

    .enable_quad_io = macronix_quad_enable, ← Quad mode enable
};
```

Specify the maximum clock frequency supported by the component. Then refer to the chip data sheet and specify the maximum clock frequency for the FMC provided by the clock and reset generator (CRG) of the chip. Assume that the clock source of the FMC can support a clock frequency of 150 MHz in maximum. Go to **arch/arm/boot/dts/hi35xx-demb.dts** and find the following component node:



```
&hisfc {
    hi_sfc {
        compatible = "jedec,spi-nor";
        reg = <0>;
        spi-max-frequency = <150000000>;
        m25p,fast-read;
    };
};
```

Change the value in **spi-max-frequency** = <150000000>. Note that only the frequency value of the 2X clock is changed in this node.

That is, when **spi-max-frequency** is set to **150000000** (Hz), the actual clock frequency provided by the CRG is 75 MHz.

Based on the frequencies of the component clock and CRG clock, the smaller value, 75 MHz, is the actual interface clock frequency.

----End

## 1.3 Porting an SPI NAND Flash Memory

### 1.3.1 Driver Path

The SPI NAND flash driver uses the non-standardized version currently.

The driver path is **drivers/mtd/nand/hifmc100/**.

[Table 1-8](#) lists the directories related to the component porting under the non-standardized SPI NAND flash drivers.

**Table 1-8** Directory structure related to the SPI NAND flash memories

HiFMC V100 Directory/File Name	Description
hifmc_spi_nand_ids.c	SPI NAND component ID list. It is the key file to be modified to port a new SPI NAND component. This file contains the parameters for reading, reading, and erasing SPI NAND components.
hifmc100_spi_general.c	Driver code universally used by most SPI NAND drivers, including the write enable operation and QE bit enable function.

### 1.3.2 Porting an SPI NAND Flash Memory

**NOTICE**

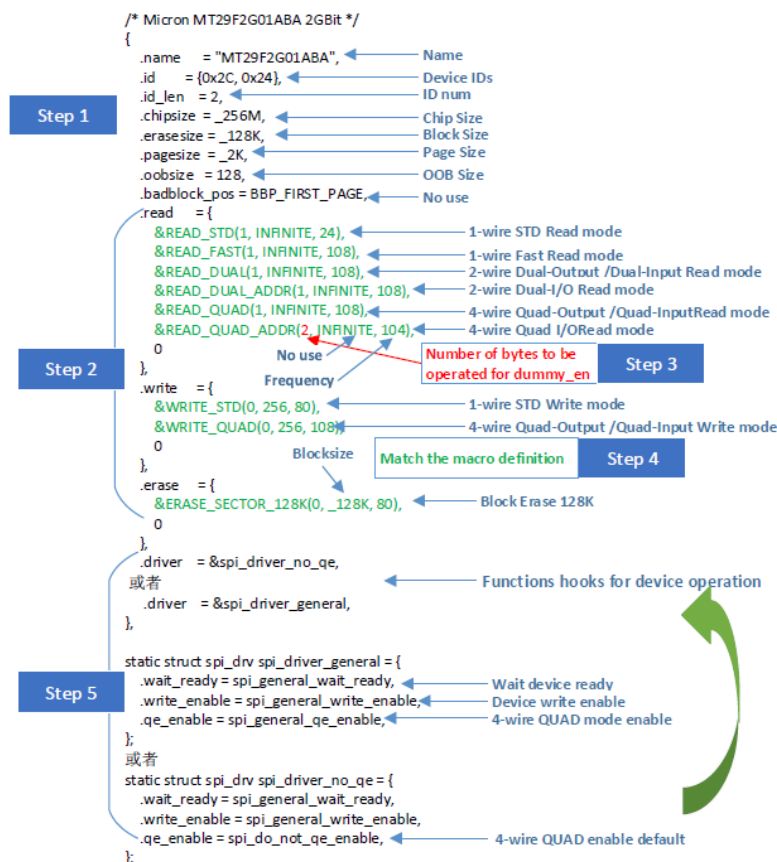
The FMC disables the error checking and correction function of the SPI NAND component when it starts, because the FMC has the error checking and correction function itself. However, disabling the error checking and correction function can only set bit 4 of the B0H feature register to 0. Therefore, to determine if the FMC of an SPI NAND component can support the error checking and correction function, check whether the bit 4 of the B0H feature register is ECC\_EN bit or not.

Register	Address	Data Bits							
		7	6	5	4	3	2	1	0
Block Lock	AOH	BRWD	Reserved	BP2	BP1	BP0	INV	CMP	Reserved
OTP	B0H	OTP_PRT	OTP_EN	Reserved	ECC_EN	Reserved	Reserved	Reserved	QE
Status	COH	Reserved	Reserved	ECSS1	ECSS0	P_FAIL	E_FAIL	WEL	OIP

Porting an SPI NAND flash memory is realized by adding a new ID node to the ID registration structure in `hifmc_spi_nand_ids.c`, as shown in Figure 1-9.

In this section, Macronix MT29F2G01ABA is used as an example to describe how to port a new SPI NAND flash memory.

**Figure 1-9** Registering the SPI NAND flash ID in the non-standardized driver





**Step 1** Refer to the MT29F2G01ABA user manual and add an ID node by filling the ID information in the corresponding position.

Obtain the ID information under the **9Fh** command.

**Table 3: READ ID Table**

Byte	Description	7	6	5	4	3	2	1	0	Value
Byte 0	Manufacturer ID (Micron)	0	0	1	0	1	1	0	0	2Ch
Byte 1	2Gb 3.3V Device ID	0	0	1	0	0	1	0	0	24h

As shown in the figure below, MT29F2G01ABA has two IDs: 0x2C and 0x24.

Obtain the **chip size, block size, page size, and OOB size**. Use the information given in the example for the items you are not sure with.

- 2Gb density
- Organization
  - Page size x1: 2176 bytes (2048 + 128 bytes)
  - Block size: 64 pages (128K + 8K bytes)
  - Plane size: 2Gb (2 planes, 1024 blocks per plane)

**Step 2** Obtain the SPI types that the component supports from the **Features** chapter in the user manual.

- Standard and extended SPI-compatible serial bus interface
  - Instruction, address on 1 pin; data out on 1, 2, or 4 pins
  - Instruction on 1 pin; address, data out on 2 or 4 pins
  - Instruction, address on 1 pin; data in on 1 or 4 pins

Refer to [Table 1-9](#) for the SPI types that MX25U25635F supports and their corresponding macro definitions in the driver. The results are marked with a gray background. According to step 1, the block size of MT29F2G01ABA is 128 KB. Therefore, the 128-KB block erase command should be matched.

**Table 1-9** SPI match table

SPI Supported	Command Word	Corresponding FMC SPI	Macro Definition in the Driver
READ	<b>03h</b>	Standard SPI	<b>READ_STD</b>
FAST READ	<b>0Bh</b>	Standard SPI	<b>READ_FAST</b>
2READ	<b>3Bh</b>	Dual-output/dual-input SPI	<b>READ_DUAL</b>
DREAD	<b>BBh</b>	Dual I/O SPI	<b>READ_DUAL_ADDR</b>
4READ	<b>6Bh</b>	Quad-output/quad-input	<b>READ_QUAD</b>



SPI Supported	Command Word	Corresponding FMC SPI	Macro Definition in the Driver
		SPI	
QREAD	<b>EBh</b>	Quad I/O SPI	<b>READ_QUAD_ADDR</b>
PP	<b>02h</b>	Standard SPI	<b>WRITE_STD</b>
DPP	<b>A2h</b>	Dual-output/dual-input SPI	<b>WRITE_DUAL</b>
2PP	<b>D2h</b>	Dual I/O SPI	<b>WRITE_DUAL_ADDR</b>
QPP	<b>32h</b>	Quad-output/quad-input SPI	<b>WRITE_QUAD</b>
4PP	<b>38h</b>	Quad I/O SPI	<b>WRITE_QUAD_ADDR</b>
Block Erase 4K	<b>02h</b>	Standard SPI	<b>ERASE_SECTOR_4K</b>
Block Erase 32K	<b>52h</b>	Standard SPI	<b>ERASE_SECTOR_32K</b>
Block Erase 64K	<b>D8h</b>	Standard SPI	<b>ERASE_SECTOR_64K</b>
Block Erase 128K	<b>D8h</b>	Standard SPI	<b>ERASE_SECTOR_128K</b>
Block Erase 256K	<b>D8h</b>	Standard SPI	<b>ERASE_SECTOR_256K</b>

**Step 3** Specify the dummy value and working clock of each SPI.

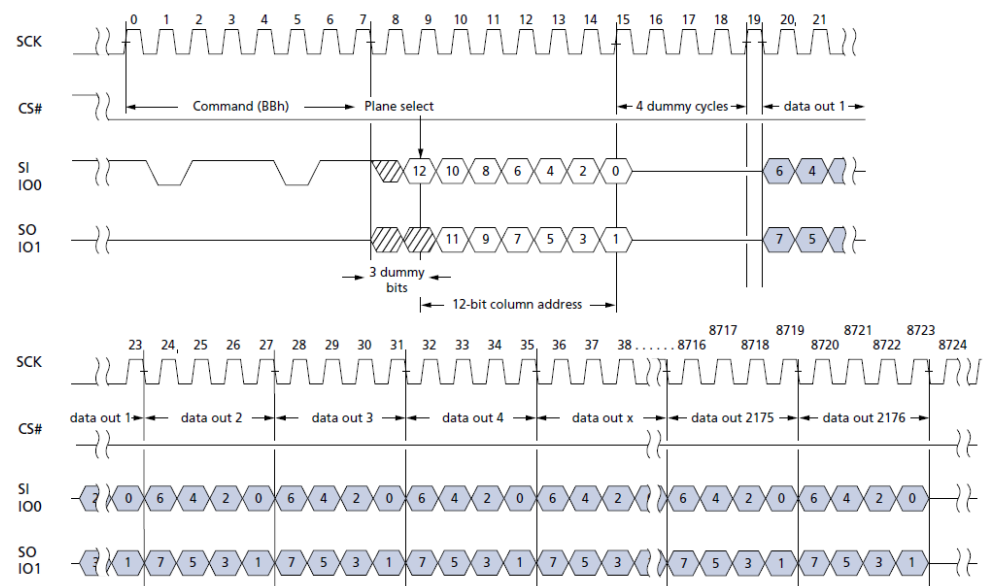
**1. Specifying dummy\_num:**

Refer to the user manual for the SPI definitions of SPI NAND flash:

- The dummy value of the standard SPI STD read, and all write and erase SPIs is **0**.
- The dummy value of the dual-output/dual-input SPIs and quad-output/quad-input SPIs is **1**.
- The dummy values of the dual I/O SPIs and quad I/O SPIs should be calculated according to the user manual.
- Dual I/O SPI Read mode



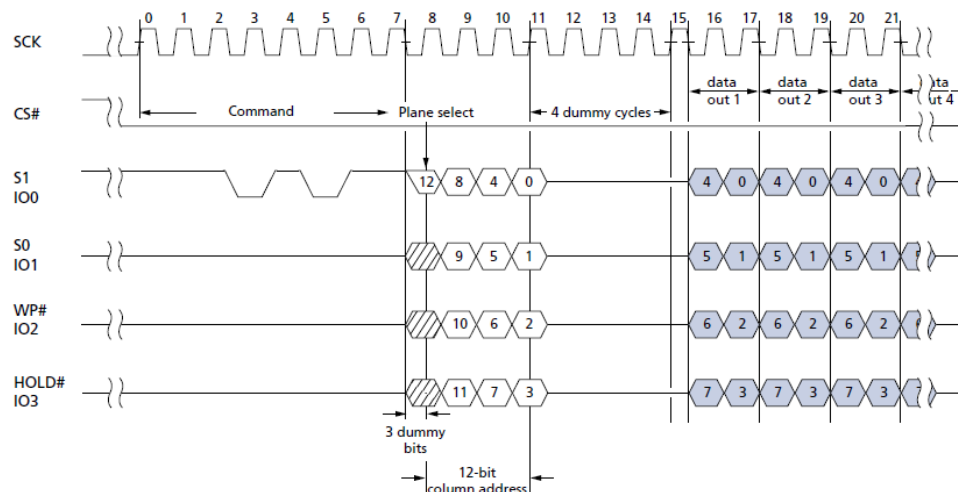
**Figure 1-10** Timing in dual I/O SPI Read mode

**Figure 16: READ FROM CACHE Dual I/O**


As shown in the waveforms excerpted from the user manual, the dual I/O SPI Read mode requires four dummy cycles (15–18). Since the SPI is a 2-wire interface, four dummy cycles take eight dummy cycle bits, or one dummy cycle byte. According to `dummy_num` defined in the user manual, the dummy value in `&READ_DUAL_ADDR` (1, INFINITE, 108) should be 1.

- Quad I/O SPI Read mode

**Figure 1-11** Timing in quad I/O SPI Read mode

**Figure 17: READ FROM CACHE Quad I/O**


As shown in the waveforms excerpted from the user manual, the quad I/O SPI Read mode requires four dummy cycles (11–14). Since the SPI is a 4-wire interface, six cycles take 16 dummy cycle bits, or two dummy cycle bytes. According to `dummy_num`





defined in the user manual, the dummy value in &READ\_DUAL\_ADDR (2, INFINITE, 108) should be 2.

## 2. SPI working clock

For example, the working frequency value in &READ\_QUAD\_ADDR (2, INFINITE, 108) is obtained from the AC characteristics table in the user manual.

**Table 1-10** AC characteristics table

**Table 17: AC Characteristics**

Parameter	Symbol	Min	Max	Unit
Clock frequency <sup>1,2</sup>	$f_C$	–	133	MHz
Clock LOW time	$t_{WL}$	3.375	–	ns
Clock HIGH time	$t_{WH}$	3.375	–	ns
Clock rise time	$t_{CRT}$	1.3	–	V/ns
Clock fall time	$t_{CFT}$	1.3	–	V/ns
Command deselect time	$t_{CS}$	30	–	ns
Chip select# active setup/hold time relative to SCK	$t_{CSS}/t_{CSH}$	3.375	–	ns
Chip select# non-active setup/hold time relative to SCK	$t_{CSH}$	2.5	–	ns
Output disable time	$t_{DIS}$	–	6	ns
Data input setup time	$t_{SUDAT}$	2.5	–	ns
Data input hold time	$t_{HDDAT}$	1.75	–	ns
Clock LOW to output valid (30pF)	$t_V$	–	6	ns
Clock LOW to output valid (10pF)	$t_V$	–	5	ns
Clock LOW to output valid (similar to SPI NOR 20 MHz read 30pF)	$t_V$	–	30	ns

Notes: 1. Read from Cache Dual I/O (BBh) and Quad I/O (EBh) can run up to 108 MHz.  
2. When read protocol similar to SPI NOR is enabled, Read from Cache 03h command can run up to 20 MHz, while read from Cache 0Bh command can run up to 133 MHz.

**Step 4** Once the SPI information is filled out as the figure shown below, you may begin to match the macro definition.

```
.read = {
    &READ_STD(1, INFINITE, 24),
    &READ_FAST(1, INFINITE, 108),
    &READ_DUAL(1, INFINITE, 108),
    &READ_DUAL_ADDR(1, INFINITE, 108),
    &READ_QUAD(1, INFINITE, 108),
    &READ_QUAD_ADDR(2, INFINITE, 104),
    0
},
.write = {
    &WRITE_STD(0, 256, 80),
    &WRITE_QUAD(0, 256, 108),
    0
},
.erase = {
    &ERASE_SECTOR_128K(0, _128K, 80),
    0
},
}
```

Match the macro definition in the beginning lines of the source code path **hifmc\_spi\_nand\_ids.c**.



```
SET_READ_STD(1, INFINITE, 24);  
SET_READ_FAST(1, INFINITE, 108);  
SET_READ_DUAL(1, INFINITE, 104);  
SET_READ_DUAL_ADDR(1, INFINITE, 108);  
SET_READ_QUAD(1, INFINITE, 108);  
SET_READ_QUAD_ADDR(2, INFINITE, 104);  
SET_WRITE_STD(0, 256, 80);  
SET_WRITE_QUAD(0, 256, 108);  
SET_ERASE_SECTOR_128K(0, 128K, 80);
```

**Step 5** Then, match the **&spi\_driver\_no\_qe** hook (the **struct spi\_drv** structure) of the related functions. The following hook values require attentions:

```
.wait_ready = spi_general_wait_ready,  
.write_enable = spi_general_write_enable,  
.qe_enable = spi_general_qe_enable, or .qe_enable = spi_do_not_qe_enable,
```

**NOTE**

The current driver is compatible with the implementation functions used by most SPI NAND flash vendors. If the new component is listed in the table, you may use the existing match to save troubles.

Vendor	struct spi_drv Value
GigaDevice	&spi_driver_general
Micron	&spi_driver_no_qe
MXIC	&spi_driver_general
Winbond	&spi_driver_general
ESMT	&spi_driver_no_qe
ATO	&spi_driver_general
Paragon	&spi_driver_general
All-flash	&spi_driver_general
TOSHIBA	&spi_driver_no_qe
HeYangTek	&spi_driver_general

- **wait\_ready = spi\_general\_wait\_ready** is used to wait for the component to be ready.

Check whether the component is busy by reading the bit 0 WIP of the C0H feature register. **Normally, this mechanism is applicable to all SPI NAND flash memories.**

**WIP bit.** The Write in Progress (WIP) bit, a volatile bit, indicates whether the device is busy in program/erase/write status register progress. When WIP bit sets to 1, which means the device is busy in program/erase/write status register progress. When WIP bit sets to 0, which means the device is not in progress of program/erase/write status register cycle.

- **.write\_enable = spi\_general\_write\_enable** is the write enable SPI.

Set whether the component is operationable by writing the bit 1 WEL of the C0H feature register. **Normally, this mechanism is applicable to all SPI NAND flash memories.**



**WEL bit.** The Write Enable Latch (WEL) bit, a volatile bit, indicates whether the device is set to internal write enable latch. When WEL bit sets to 1, which means the internal write enable latch is set, the device can accept program/erase/write status register instruction. When WEL bit sets to 0, which means no internal write enable latch; the device will not accept program/erase/write status register instruction. The program/erase command will be ignored if it is applied to a protected memory area. To ensure both WIP bit & WEL bit are both set to 0 and available for next program/erase/operations, WIP bit needs to be confirm to be 0 before polling WEL bit. After WIP bit confirmed, WEL bit needs to be confirm to be 0.

**NOTE**

Unlike the SPI NOR flash, the SPI NAND flash does not require the reset pin, because the FMC sends the reset command automatically. That is why the SPI NAND flash is matched with the quad-wire mode by default. However, not all SPI NAND flash memories are matched with the quad-wire mode by default before delivery. These exceptions come in two types based on the QE bit.

**Figure 1-12** Feature registers without the QE bit**Table 5: Feature Address Settings and Data Bits**

Register	Feature Address	Feature Data Bits								Notes
		7	6	5	4	3	2	1	0	
Block lock	Address = A0h; Access = R/W	BRWD	BP3	BP2	BP1	BP0	TB	WP#/HOLD# Disable	–	1, 2
Configuration	Address = B0h; Access = R/W	CFG2	CFG1	LOT_EN	ECC_EN	–	–	CFG0	–	1
Status	Address = C0h; Access = R	CRBSY	ECCS2	ECCS1	ECCS0	P_Fail	E_Fail	WEL	OIP	1

**Figure 1-13** Feature registers with the QE bit**Feature Register Table**

Register	Address	Data Bits							
		7	6	5	4	3	2	1	0
Block Lock	A0H	BRWD	Reserved	BP2	BP1	BP0	INV	CMP	Reserved
OTP	B0H	OTP_PRT	OTP_EN	Reserved	ECC_EN	Reserved	Reserved	Reserved	QE
Status	C0H	Reserved	Reserved	ECCS1	ECCS0	P_FAIL	E_FAIL	WEL	OIP

- **.qe\_enable = spi\_do\_not\_qe\_enable** is the quad-wire enable function interface.

The quad-wire enable function of Micron MT29F2G01ABA is enabled after power-on by default. Therefore, you do not have to enable the function.

However, for some SPI NAND flash memories, the function needs to be enabled to use the quad SPI. The corresponding SPI is **.qe\_enable = spi\_general\_qe\_enable**.

```
static int spi_general_qe_enable(struct hifmc_spi *spi)
{
    unsigned int reg, op;
    const char *str[] = {"Disable", "Enable"};

    op = spi_is_quad(spi); ← Check whether the 4-wire mode should be supported

    reg = spi_nand_feature_op(spi, GET_OP, FEATURE_ADDR, 0);
    if ((reg & FEATURE_QE_ENABLE) == op) {
        FMC_PR(QE_DBG, "\t| | *-SPI Nand quad was %sd!\n", str[op]);
        return op;
    } Get the status register to check the QE bit value

    if (op == ENABLE)
        reg |= FEATURE_QE_ENABLE; ← Turn on QE
    else
        reg &= ~FEATURE_QE_ENABLE; ← Turn off QE

    spi_nand_feature_op(spi, SET_OP, FEATURE_ADDR, reg);
    Write back to feature register ←
    spi->driver->wait_ready(spi);
    Check whether the setting is successful ←

    reg = spi_nand_feature_op(spi, GET_OP, FEATURE_ADDR, 0);
    if ((reg & FEATURE_QE_ENABLE) == op)
        FMC_PR(QE_DBG, "\t| | *-SPI Nand %s Quad succeed!\n", str[op]);
    else
        DB_MSG("Error: %s Quad failed! reg: %#x\n", str[op], reg);

    FMC_PR(QE_DBG, "\t| | *-End SPI Nand %s Quad.\n", str[op]);

    return op;
}
```



#### NOTE

Currently, the QE bit enable of most SPI NAND flash memories follow the preceding driver operations. If the component to be ported uses different QE bit enable method, you may make modifications according to the driver code given above.

**Step 6** Integrate the ID into the **hifmc\_spi\_nand\_ids.c** file, compile and burn U-Boot, start U-Boot and check the information displayed. Make sure that the ECC type matches with the ECC type specified in the user manual.

- User-selectable internal ECC supported
  - 8 bits/sector



The ECC size matched by the driver should be greater than or equal to that specified in the user manual. You are advised to match an ECC type with higher strength, which is good for prolonging the lifecycle of the component.

The ECC type parameters to be inputted for the image burning with the mass production tool (such as `spinand_product` and `nand_product`) are provided as follows:

ECC Type    ECC Size:

1            4-bit/512 bytes



- 2 16-bit/1 KB
- 3 24-bit/1 KB
- 4 28-bit/1 KB

You must match the actual ECC type which can be obtained from the information displayed by U-Boot driver.

----End

## 1.4 Porting a Parallel NAND Flash Memory

### 1.4.1 Driver Path

- The parallel NAND driver uses the non-standardized version currently.
- The driver path is **drivers/mtd/nand/hifmc100\_nand/**.
- [Table 1-11](#) lists the directory related to the component porting under the non-standardized parallel NAND driver.

**Table 1-11** Directory structure related to the ID porting of parallel NAND driver

HiFMC V100 Directory/File Name	Description
hifmc_nand_spl_ids.c	Special component ID list of parallel NAND. It is the key file to be modified to port a new parallel NAND component.

### 1.4.2 Parallel NAND ID System

#### NOTICE

The parallel NAND component does not need to conduct the ECC operation because the FMC has the error checking and correction function itself. There is a kind of parallel NAND components whose integrated error checking and correction function cannot be disabled. It is not advised to port this kind of parallel NAND components for two reasons. First, special command words are needed to read the ECC bit number, which will increase the complexity of the driver and the maintenance difficulty. Second, during the actual operation process, unexplainable errors will appear when two ECC operations (the component ECC and the FMC ECC) are implemented.

The parallel NAND driver contains two lists.

- Common ID list: The kernel NAND adaption layer identifies and decodes the ID information including the chip size, block size, page size, and OOB size.
- Special ID list: Not all ID information obtained from the NAND adaption layer can satisfy the actual needs of the parallel NAND component. Therefore, a special component ID list is defined.



## Common ID List

The ID information of the parallel NAND flash obtained is of several bytes (generally four bytes at least). Later releases support five or more bytes. The ID bytes provide a wealth of information. For example, the number of chips inside the NAND flash memory, the number of planes each chip contains, the page size and block size of each plane, and so on. As shown in [Figure 1-14](#), the ID information of the parallel NAND flash indicates whether the flash is single-level cell (SLC) or multi-level cell (MLC) NAND flash. It can also provide the chip size, block size, page size, and OOB size.

**Figure 1-14** ID information meaning of parallel NAND flash

Device Identifier Byte	Description
1st	Manufacturer Code
2nd	Device Identifier
3rd	Internal chip number, cell type, etc.
4th	Page Size, Block Size, Spare Size, Serial Access Time, Organization
5th (S34ML02G1, S34ML04G1)	ECC, Multiplane information

## Special ID List

**Figure 1-15** ID registration steps for parallel NAND flash

```
{ /* SLC S34ML02G200TFI000 */
.name = "S34ML02G200TFI000",
.id = {0x01, 0xDA, 0x90, 0x95, 0x46, 0x00, 0x00, 0x00},
.length = 5,
.chipsize = _256M,
.probe = NULL,
.pagesize = _2K,
.erasize = _128K,
.oobsize = 128,
.options = 0,
.read_retry_type = NAND_RR_NONE,
.badblock_pos = BBP_FIRST_PAGE,
.flags = 0,
},
```

Annotations for Figure 1-15:

- Name: .name = "S34ML02G200TFI000"
- Device IDs: .id = {0x01, 0xDA, 0x90, 0x95, 0x46, 0x00, 0x00, 0x00}
- ID num: .length = 5
- Chip Size: .chipsize = \_256M
- No use: .probe = NULL
- Page Size: .pagesize = \_2K
- Block Size: .erasize = \_128K
- OOB Size: .oobsize = 128
- No use: .options = 0
- Read retry Flag( No use): .read\_retry\_type = NAND\_RR\_NONE
- No use: .badblock\_pos = BBP\_FIRST\_PAGE
- NAND\_RANDOMIZER etc. Flags: .flags = 0

### 1.4.3 Porting a Parallel NAND Flash Memory

**Step 1** First, try to match the common ID list. Refer to the **Features** chapter and check if the information displayed by U-Boot or kernel driver satisfies the requirements specified by the user manual. If the requirements are satisfied, no extra ID information is needed to support this component.



## FEATURES

- Organization

	x8
Memory cell array	2176 × 64K × 8
Register	2176 × 8
Page size	2176 bytes
Block size	(128K + 8K) bytes



- The ECC size matched by the driver should be greater than or equal to that specified in the user manual.
- The ECC type parameters to be inputted for the image burning with spinand\_product are provided as follows:

ECC type    ECC size:

1	4-bit/512 bytes
2	16-bit/1 KB
3	24-bit/1 KB
4	28-bit/1 KB
5	40-bit/1 KB
6	64-bit/1 KB

You must match the actual ECC type which can be obtained from the information displayed by U-Boot driver.

- Step 2** If the ID information matched by the common ID list does not satisfy the requirements specified in the user manual, further ID information should be added to the special ID list.
- Step 3** In this section, Spansion S34ML02G200TFI000 is used as an example to describe how to port a new parallel NAND flash memory.
- Step 4** Read the contents of sections 1 and 2, find the corresponding ID list path, and understand the meaning of the information provided by the ID list.
- Step 5** Refer to the S34ML02G200TFI000 user manual and find the ID information under the **90h** command.

**Table 3.3** Read ID for Supported Configurations

Density	Org	V <sub>CC</sub>	1st	2nd	3rd	4th	5th
1 Gb	x8	3.3V	01h	F1h	80h	1Dh	—
2 Gb			01h	DAh	90h	95h	46h
4 Gb			01h	DCh	90h	95h	56h
1 Gb	x16		01h	C1h	80h	5Dh	—
2 Gb			01h	CAh	90h	D5h	46h
4 Gb			01h	CCh	90h	D5h	56h

S34ML02G200TFI000 has five IDs: 0x01, 0xDA, 0x90, 0x95, and 0x46.



**Step 6** Obtain the component ID information from the **Features** chapter, add new ID node, and fill the information at the corresponding position of the node.

■ **Density**

- 1 Gbit / 2 Gbit / 4 Gbit

■ **Architecture**

- Input / Output Bus Width: 8-bits / 16-bits
- Page Size:
  - x8:
    - 1 Gbit: (2048 + 64) bytes; 64-byte spare area
    - 2 Gbit / 4 Gbit: (2048 + 128) bytes; 128-byte spare area
- Block Size: 64 Pages
  - x8:
    - 1 Gbit: 128k + 4k bytes
    - 2 Gbit / 4 Gbit: 128k + 8k bytes
- Plane Size
  - x8
    - 1 Gbit: 1024 Blocks per Plane or (128M + 4M) bytes
    - 2 Gbit: 1024 Blocks per Plane or (128M + 8M) bytes
- Device Size
  - 1 Gbit: 1 Plane per Device or 128 Mbyte
  - 2 Gbit: 2 Planes per Device or 256 Mbyte

As shown in the figure below, fill out the corresponding ID list according to the information above.

```
/****** Spansion *****/
{
    /* SLC S34ML02G200TFI000 */
    .name      = "S34ML02G200TFI000",
    .id        = {0x01, 0xDA, 0x90, 0x95, 0x46, 0x00, 0x00, 0x00},
    .length    = 5,
    .chipsize  = 256M,
    .probe     = NULL,
    .pagesize  = 2K,
    .erasesize = 128K,
    .oobsize   = 128,
    .options   = 0,
    .read_retry_type = NAND_RR_NONE,
    .badblock_pos = BBP_FIRST_PAGE,
    .flags     = 0,
},
```

**Step 7** Integrate the ID into the **hifmc\_nand\_spl\_ids.c** file, compile and burn U-Boot, start U-Boot, and check the information displayed. Make sure that the ECC type matches with the ECC type specified in the user manual. (The ECC size should be greater than that specified in the user manual.)

■ **Reliability**

- 100,000 Program / Erase cycles (Typ)  
(with 4-bit ECC per 528 bytes (x8) or 264 words (x16))



**NOTICE**

For Spansion S34ML02G200TFI000, the ECC type requirements specified in the user manual can be satisfied by matching the common ID list. However, the FMC with 128-byte OOB can match the 24-bit/1-KB ECC. You are advised to match an ECC type with higher strength, which is good for prolonging the lifecycle of the component.

---

----End

## 2 Common Issues

### 2.1 3-Byte/4-Byte Boot of the SPI NOR Flash and 1-Wire/4-Wire Boot of the SPI NAND Flash

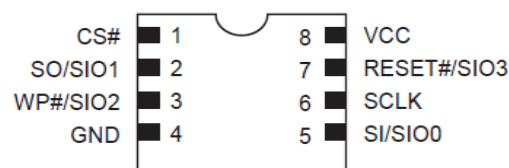
The 3-byte/4-byte address width applies to the SPI NOR flash. When the chip is powered on, it reads the first 1-MB space of the SPI NOR flash. During the reading, the address loaded is either 3-byte or 4-byte wide.

- For the 3-byte address width, the maximum addressing space is 16 MB. Therefore, the 3-byte mode must be used for the SPI NOR flash memories of 16-MB or lower capacity.
- Most SPI NOR flash memories of 32-MB or higher capacity support 3-byte mode by default before delivery, which means that only the first 16-MB space can be accessed. Therefore, the 3-byte mode must be used.
- Some of the SPI NOR flash memories of 32-MB or higher capacity (such as MXIC MX25L25735E and Winbond W25Q257FV) support 4-byte mode by default before delivery. Therefore, the 4-byte mode must be used.

The 1-wire/4-wire boot applies to the SPI NAND flash. The SPI NOR flash uses 1-wire boot by default. Nearly all SPI NAND flash memories support both options. The 4-wire SPI is backward compatible with the 1-wire SPI. The 1-wire/4-wire switch is decided by the hardware design:

- When SIO 3 is connected to a valid I/O interface of the chip, 4-wire boot is supported. Therefore, the 4-wire boot mode must be used. As a result, you gain a faster startup speed.
- When SIO 3 is not connected to a valid I/O SPI (This is possible because the SPI NOR and SPI NAND of the BVT products share the soldering pads and I/O 3 of the SPI NOR flash is generally multiplexed as the reset signal.), the 1-wire boot mode must be used.

**Figure 2-1** 8-WSON package





## 2.2 HiBurn Displays "Burn Success" but No Information Displayed Over the Serial Ports

It is not rare to find nothing but spaces are displayed over the serial ports while the HiBurn displays "Burn Success." There are two possible causes.

- The 3-byte/4-byte error of the SPI NOR flash:  
Burn the DDR SDRAM under FastBoot, and then read and write the data after boot. If you find there is data misplacement between the data read and the original data, it is confirmed that the cause is the 3-byte/4-byte interface error. There is a kind of flash memories by some of the vendors. They can be restarted after power-off while they cannot be restarted with the reset command. This is also a possible cause.

Figure 2-2 Comparison between 3-byte and 4-byte problem data of the SPI NOR flash

00000000	17 05 00 EA 14 F0 9F E5	14 F0 9F E5 14 F0 9F E5	...è.òYà.òYà.òYà
00000010	14 F0 9F E5 14 F0 9F E5	14 F0 9F E5 14 F0 9F E5	.òYà.òYà.òYà.òYà
00000020	A0 16 80 80 00 17 80 80	60 17 80 80 C0 17 80 80	.ëë.ëë.ëë.ëë
00000030	20 18 80 80 80 18 80 80	E0 18 80 80 78 56 34 12	.ëëë.ëë.ëëxV4.
00000040	50 00 04 12 00 00 00 00	00 00 00 00 FD 00 00 00	P.....ý..
00000050	54 00 04 12 00 00 00 00	00 00 00 00 00 00 00 00	T.....ý.
00000060	00 00 00 00 00 00 00 00	64 00 00 00 00 00 00 00	.....d.....
00000070	00 00 04 12 00 00 00 12	00 00 00 00 FD 00 00 00	.....ý..
00000080	04 00 04 12 45 61 00 01	00 00 00 00 FD 00 00 00	....Ea.....ý..
00000090	98 01 04 12 06 00 00 00	00 00 00 00 15 00 00 00	.....ý..
000000A0	30 00 04 12 00 00 00 12	00 00 00 00 FD 00 00 00	0.....ý..
000000B0	34 00 04 12 15 C2 00 00	00 00 00 00 FD 00 00 00	4.....Ä.....ý..
000000C0	98 01 04 12 49 00 00 00	00 00 00 00 45 18 00 00	.....I.....E..
000000D0	98 01 04 12 01 00 00 00	00 00 00 00 00 00 00 00	.....ý..
000000E0	98 01 04 12 00 00 00 00	00 00 00 00 05 10 00 00	.....ý..
000000F0	58 01 04 12 FF 00 00 00	00 00 00 00 00 00 3D 00	X...ý.....=.
00000100	08 00 00 10 02 00 00 00	00 00 00 00 1D 40 00 00	.....@..
00000110	50 00 04 12 7C FF 01 00	00 00 00 00 8D 00 00 00	P... ý.....
00000120	54 00 04 12 7C FF 01 00	00 00 00 00 00 00 8D 00	T... ý.....
00000130	00 00 00 00 00 00 00 00	F4 01 00 00 00 00 00 00	.....ð.....
00000140	4C 01 04 12 03 00 00 00	00 00 00 00 0D 28 00 00	L.....(.....
00000150	48 01 04 12 01 00 00 00	00 00 00 00 0D 10 00 00	H.....
00000160	90 00 05 12 FF FF FF FF	00 00 00 00 FD 00 00 00	.....ýyyý.....ý..
00000170	94 00 05 12 C9 91 73 B1	00 00 00 00 FD 00 00 00	"...É'st.....ý..

**Solution to the 3-byte/4-byte error of the SPI NOR flash:** Check the DIP switch or the software reset process.

- SPI NOR and SPI NAND 4-wire error  
Burn DDR under FastBoot, and then read and write the data after boot. If you find there is data misplacement of 1 bit between the data read and the original data, it is confirmed that the cause is the 4-wire interface error.



**Figure 2-3** Comparison between 4-wire problem data of the SPI NOR and SPI NAND flash

3	88889ea8	88889f88	88889fe8	88889fc8	3	808016a0	80801700	80801760	808017c0
4	888898a8	88889888	888898e8	9abcdef8	4	80801820	80801880	808018e0	12345678
5	9a8c88d8	88888888	88888888	888888fd	5	12040050	00000000	00000000	000000fd
6	9a8c88dc	88888888	88888888	88fd8888	6	12040054	00000000	00000000	00fd0000
7	88888888	88888888	888888ec	88888888	7	00000000	00000000	00000064	00000000
8	9a8c8888	9a888888	88888888	888888fd	8	12040000	12000000	00000000	000000fd
9	9a8c888c	8988e9cd	88888888	888888fd	9	12040004	01006145	00000000	000000fd
10	9a8c8998	8888888e	88888888	8888889d	10	12040198	00000006	00000000	00000015
11	9a8c88b8	9a888888	88888888	888888fd	11	12040030	12000000	00000000	000000fd
12	9a8c88bc	8888ca9d	88888888	888888fd	12	12040034	0000c215	00000000	000000fd
13	9a8c8998	888888c9	88888888	888898cd	13	12040198	00000049	00000000	00001845
14	9a8c8998	88888889	88888888	8888888d	14	12040198	00000001	00000000	0000000d
15	9a8c8998	88888888	88888888	8888988d	15	12040198	00000000	00000000	00001005
16	9a8c89d8	888888ff	88888888	88bd8888	16	12040158	000000ff	00000000	003d0000
17	98080000	8888888a	88888888	8888c89d	17	10000008	00000002	00000000	0000401d
18	9a8c88d8	8889ffff	88888888	8888888d	18	12040050	0001ff7c	00000000	0000008d
19	9a8c88dc	8889ffff	88888888	888d8888	19	12040054	0001ff7c	00000000	008d0000
20	88888888	88888888	88889ffc	88888888	20	00000000	00000000	000001f4	00000000
21	9a8c89cc	8888888b	88888888	8888a88d	21	1204014c	00000003	00000000	0000280d
22	9a8c89c8	88888889	88888888	8888988d	22	12040148	00000001	00000000	0000100d
23	9a8d8898	ffffffff	88888888	888888fd	23	12050090	ffffffff	00000000	000000fd
24	9a8d889c	b9fb99c9	88888888	888888fd	24	12050094	b17391c9	00000000	000000fd
25	9a8d8898	88888888	88888888	888888fd	25	12050098	00000000	00000000	000000fd
26	9a8d88a8	88889fff	88888888	888888fd	26	120500a0	000017ff	00000000	000000fd
27	9a8e8888	88c988cf	88888888	888888fd	27	120e0000	004100c7	00000000	000000fd
28	9a8e888c	88a888cf	88888888	888888fd	28	120e0004	002800c7	00000000	000000fd
29	9a8e8898	8888888d	88888888	8888889d	29	120e0010	00000005	00000000	0000001d
30	88888888	88888888	889eebe8	88888888	30	00000000	00000000	0016e360	00000000

**Solution to the 4-wire error of the SPI NOR and SPI NAND flash:** check the hardware interfaces.

- The I/O information line connected to IO 3 should be valid.
- IO 3 should be connected to pull-up resistors.

## 2.3 4-Bit ECC and 8-Bit/1 KB ECC, 8-Bit ECC, and 16-Bit/1 KB ECC

Users may be confused with the 4-bit ECC and the 8-bit/1 KB ECC, the 8-bit ECC and the 16-bit/1 KB ECC. In the sense of components, the length of the error correction unit of the 4-bit ECC and 8-bit ECC is 512 bytes. However, for the HiFMC V100, the length of the error correction unit of the 8-bit ECC and the 16-bit ECC is 1 KB. (The 1 KB here refers to the 1 KB level but not exactly 1024 bytes; 512 B refers to the 512 B level (526 bytes) but not exactly 512 bytes). Therefore, the 4-bit/512 bytes ECC mode is equivalent to the 8-bit/1 KB ECC mode, and the 8-bit/512 bytes ECC mode is equivalent to the 16-bit/1 KB ECC mode.

The development engineers and support personnel are advised to explain the error correction unit of the error checking and correction function to the customers to avoid misunderstanding.

## 2.4 What Should Be Paid Attention to When a Large-Capacity NAND Flash Is Used?

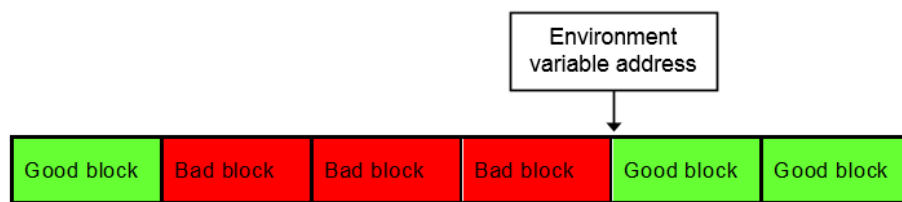
The large-capacity NAND flash here refers to the NAND flash larger than 4 GB. The maximum 32-bit unsigned integer is 4 GB. If the capacity is larger than 4 GB, the 32-bit variable may be wrapped. The maximum capacity of the tested NAND flash is 8 GB. If a NAND flash memory larger than 4 GB is used, ensure that the partition size of the file system (UBI or YAFFS2) is less than or equal to 4 GB. Otherwise, the file system may be wrapped.

## 2.5 Why Cannot the System Start after U-Boot Saves Environment Variables in the NAND Flash?

The system can start normally after U-Boot is burnt in the NAND flash. However, the system cannot start after the environment variables are saved.

The reason is shown in [Figure 2-4](#). U-Boot occupies three good blocks while there are three bad blocks at the start address for the NAND flash. The FastBoot content is erased after the environment variables are saved. As a result, the system cannot start. To resolve this problem, move environment variable address backwards.

**Figure 2-4** Distribution of bad blocks

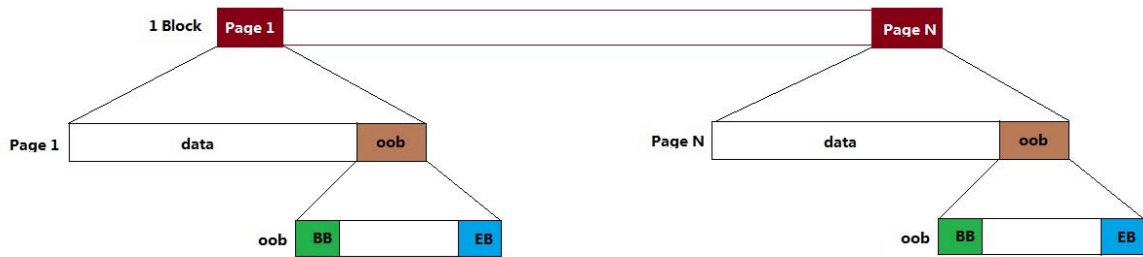


## 2.6 How Do I Use the nandwrite Naked-Write Tool of mtd-utils?

When the nandwrite naked-write tool of mtd-utils is used, if U-Boot.bin image is written and the image size is greater than the size of a block in the NAND flash, ensure that the data of U-Boot.bin image is aligned and filled by block. Otherwise, the written u-boot.bin image cannot boot normally.

The reason is as follows:

The bad block (BB) flag bits of some NAND flash are marked as non-all-0 values such as 0xFE upon delivery. As a result, during BB judgment, the FMC easily corrects the BB flag bits to 0xFF by using ECC correction (because the ECC algorithm of the controller considers 0xFF to be valid and correctable). Therefore, the empty block (EB) flag bits are set at the last two bytes of the OOB information on each page. As shown in [Figure 2-5](#), when U-Boot is started, the prerequisite for the logic to consider a block as a good block is as follows: BB is 0xFF and EB is 0x00 on page 1 and page N.

**Figure 2-5** Structure of NAND flash blocks

The nandwrite tool writes data by page based on the size of the image file, and automatically sets the EB flag bits of the current page to 0x00 during page writing. If the last page to be written is not the last page of the block, the logic considers this block to be empty and does not read because the EB flag bits on the last page of the block are 0xFF. As a result, U-Boot fails to be started.

Note that the first block is guaranteed to be a good block during the delivery of the NAND flash, and the logic does not judge the first block. Therefore, if the size of U-Boot.bin image is less than the size of the first block, U-Boot is started normally.

In addition, once U-Boot is started normally, the EB bits are not judged. This is why the image size does not need to be block-aligned when the nandwrite tool is used to write the kernel image and file system image.

## 2.7 What Do I Do If the Flash Compatibility Is Affected by Parameter Changes Caused by Updated Process but Unchanged ID of the Flash?

With the constant upgrade of the flash (SPI NOR, SPI NAND, or parallel NAND) process, the flash interface, OOB, and performance parameters are continuously changed and optimized as well.

However, some vendors do not change the flash ID even if the flash manufacturing process is upgraded. The driver identifies a component based on its ID. It does not identify the component information (such as batch number and manufacturing process) based on the SFDP register recommended by the vendor. The reason is that the SFDP is not standard, and varies according to different vendors. Even for the same vendor, the SFDP register for the new component differs from that for the old component.

The flash memories listed as follows have the same ID but inconsistent parameters, affecting the compatibility. If the component you used is list in the table or similar problems occur, you may modify the driver accordingly to guarantee the function and performance of the component.

- The ID of the SPI NOR flash remains unchanged but the interface type is changed.  
The ID of MXIC MX25L6436F is the same as that of MXIC MX25L6406E. However, MXIC MX25L6406E supports the interface types 2x I/O Read Mode, 4x I/O Read Mode, and 4x I/O Page Program.

For MXIC MX25L6406E, the component information structure `hifmc_spi_nor_info_table` (hifmcv100) or `spi_info_table` (hisfc350) in the file `hifmc_spi_nor_ids.c` (hifmcv100) or `hisfc350_spi_ids.c` (hisfc350) should be defined as follows:

**Figure 2-6** MXIC MX25L6406E ID registry

```
{
    "MX25L6406E", {0xc2, 0x20, 0x17}, 3, _8M, _64K, 3,
    {
        &READ_STD(0, INFINITE, 50),
        &READ_FAST(1, INFINITE, 86),
        &READ_DUAL(1, INFINITE, 80),
        0
    },
    {
        &WRITE_STD(0, 256, 86),
        0
    },
    {
        &ERASE_SECTOR_64K(0, _64K, 86),
        0
    },
    &spi_driver_mx25l25635e,
},
```

For MXIC MX25L6436F, the component information structure `hifmc_spi_nor_info_table` (hifmcv100) or `spi_info_table` (hisfc350) in the file `hifmc_spi_nor_ids.c` (hifmcv100) or `hisfc350_spi_ids.c` (hisfc350) should be defined as follows:

**Figure 2-7** MXIC MX25L6436F ID registry

```
{
    "MX25L6436F", {0xc2, 0x20, 0x17}, 3, _8M, _64K, 3,
    {
        &READ_STD(0, INFINITE, 50),
        &READ_FAST(1, INFINITE, 133),
        &READ_DUAL(1, INFINITE, 133),
        &READ_DUAL_ADDR(1, INFINITE, 133),
#ifdef CONFIG_CLOSE_SPI_8PIN_4IO
        &READ_QUAD(1, INFINITE, 133),
        &READ_QUAD_ADDR(3, INFINITE, 133),
#endif
        0
    },
    {
        &WRITE_STD(0, 256, 133),
#ifdef CONFIG_CLOSE_SPI_8PIN_4IO
        &WRITE_QUAD_ADDR(0, 256, 133),
#endif
        0
    },
    {
        &ERASE_SECTOR_64K(0, _64K, 133),
        0
    },
    &spi_driver_mx25l25635e,
},
```



- The ID of the SPI NOR flash remains unchanged but the command word is changed.  
For the Micron SPI NOR flash, the **EXTENDED QUAD INPUT FAST PROGRAM** command for the MT25Q series differs from that for the N25Q series. The difference is described in [Figure 2-8](#):

**Figure 2-8** Command difference between MT25Q series and N25Q series

15. The code 38h is valid only for part numbers N25Q256A83ESF40x, N25Q256A83E1240x, and N25Q256A83ESFA0F; the code 12h is valid for the other part numbers.

The Micron MT25Q series components can use the higher-performance WRITE\_QUAD\_ADDR interface type by directly configuring WRITE\_QUAD\_ADDR because currently the drive matches the 38h command by default.