



基于 Hifmcv100 控制器的 Flash 移植指南

文档版本 00B02
发布日期 2018-05-15

版权所有 © 深圳市海思半导体有限公司 2016-2018。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市海思半导体有限公司

地址：深圳市龙岗区坂田华为基地华为电气生产中心 邮编：518129

网址：<http://www.hisilicon.com>

客户服务电话：+86-755-28788858

客户服务传真：+86-755-28357515

客户服务邮箱：support@hisilicon.com



前言

概述

本文档主要介绍如何基于 HiFMCV100 驱动新增 Flash 器件及常见问题分析。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3519	V100
Hi3519	V101
Hi3516A	V200
Hi3516C	V300
Hi3516E	V100
Hi3559	V100
Hi3556	V100
Hi3536C	V100
Hi3536D	V100
Hi3531D	V100
Hi3521D	V100
Hi35520D	V400
Hi3559A	V100
Hi3559C	V100
Hi3518E	V200/V201
Hi3516C	C200
Hi3521A	V100
Hi3531A	V100



产品名称	产品版本
Hi3519A	V100
Hi3556A	V100
Hi3516C	V500
Hi3516D	V300
Hi3559	V200
Hi3556	V200






读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
 警告	表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 注意	表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 窍门	表示能帮助您解决某个问题或节省您的时间。
 说明	表示是正文的附加信息，是对正文的强调和补充。

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。



文档版本 00B02 (2018-05-15)

第 2 次临时发布。

新增 1.2.1.4 小节

1.2.2.2 涉及修改

文档版本 00B01 (2017-10-10)

第 1 次临时发布。



目 录

前 言.....	i
1 新 Flash 器件移植.....	1
1.1 导读.....	1
1.2 SPI Nor Flash 器件的移植	1
1.2.1 SPI-Nor U-boot 下的移植	1
1.2.2 SPI-Nor 内核下的移植	13
1.3 SPI Nand Flash 器件的移植.....	18
1.3.1 相关驱动路径.....	18
1.3.2 如何新增一颗 SPI Nand Flash.....	18
1.4 并口 Nand 驱动	27
1.4.1 相关驱动路径.....	27
1.4.2 并口 Nand ID 表系统.....	27
1.4.3 如何新增一颗并口 Nand 器件	29
2 常见问题整理.....	32
2.1 SPI Nor Flash 3byte/4byte 启动和 SPI Nand Flash 1/4 线启动.....	32
2.2 Hibern 显示烧写成功，但串口没打印.....	33
2.3 4bit ECC、8bit ECC 和 8bit/1k ECC、16bit/1k ECC.....	34
2.4 使用大容量 NAND 应该注意的地方	34
2.5 为什么在 NAND 上, u-boot 保存环境变量后，系统无法启动.....	35
2.6 如何正确使用 mtd-utils 的 nandwrite 裸写工具.....	35
2.7 有些 Flash 器件 ID 不变工艺更新导致参数变化对兼容性的影响.....	36



1 新 Flash 器件移植

1.1 导读

本文档基于 HiFMCV100（Hisilicon Flash Memory Controller 以下简称 FMC）控制器，提供 FLASH 器件的移植方法。FMC 集成 SPI Nor Flash、SPI Nand Flash、并口 Nand Flash 三个控制器，部分平台上裁剪成二合一的控制器，即包含 SPI Nor Flash 和 SPI Nand Flash 两个控制器。

- 当选用的 SPI Nor Flash 器件的型号与 SPI Nor Flash 器件兼容性列表中型号不同时，需要在 SPI Nor Flash 驱动的 ID 列表中新增器件 ID 节点以及器件相关的功能函数，主要包括读写擦类型、时钟、ID 值、实现并指定器件相关的功能函数（4Byte 寻址，四线读写使能、复位函数等）。
- 当选用的 SPI Nand Flash 器件的型号与 SPI Nand Flash 器件兼容性列表中型号不同时，需要在 SPI Nand Flash 驱动的 ID 列表中新增 ID 节点以及器件相关的功能函数，主要包括读写类型、时钟大小，ID 值、实现并指定四线使能处理函数。
- 当选用的并口 Nand Flash 器件的型号与并口 Nand Flash 器件兼容性列表中型号不同时，第一步先匹配 Nand Flash 协议层提供的公共器件列表，若公共表匹配出的器件信息不满足器件的实际情况，需要在并口 Nand Flash 驱动的特殊器件列表中新增器件 ID 节点。

1.2 SPI Nor Flash 器件的移植

由于 Flash 驱动都在标准化，内核下的 SPI Nor Flash 驱动已经匹配最新的标准化驱动框架，而 u-boot 下的 SPI Nor Flash 驱动使用的是非标准化的版本。

当前有 u-boot-2010.06 和 u-boot-2016.11 两个版本，两者的差异主要是 u-boot-2016.11 新增支持使用 **4 Byte 命令** 访问大于 16MB 的 SPI Nor Flash，详见 [1.2.1.4 u-boot-2016.11 下大于 16MB SPI Nor Flash 的移植](#) 小节。

1.2.1 SPI-Nor U-boot 下的移植

1.2.1.1 相关驱动路径

目前 u-boot 下的 SPI Nor Flash 驱动使用的是没有标准化的版本。



u-boot 下的路径：drivers/mtd/spi/hifmc100/

非标准化 SPI NorFlash 驱动新器件移植有关系的目录如下表 1-1 所示。

表1-1 SPI Nor Flash 器件相关的主要目录结构

Hifmc100 目录/文件名	描述
hifmc_spi_nor_ids.c	SPI Nor Flash 器件 ID 表，移植新的器件主要修改的文件，主要包含 SPI Nor Flash 器件的读写擦的参数和功能函数钩子的赋值。
hifmc100_spi_general.c	SPI Nor Flash 驱动大部分器件通用的功能函数代码。包含写使能等操作，4Byte 寻址与四线读写，但是只适应部分厂商，特殊器件需要根据器件手册增加相应的实现函数
hifmc100_spi_gd25qxxx.c hifmc100_spi_micron.c hifmc100_spi_mx25l25635e.c hifmc100_spi_s25fl256s.c hifmc100_spi_w25q256fv.c	部分特殊器件中的功能函数与 hifmc100_spi_general.c 中的驱动不匹配，只能额外增加这些器件具体功能的驱动实现，主要是 4byte 寻址（32M 或以上容量器件），QUAD 四线读写，QE 位使能函数。

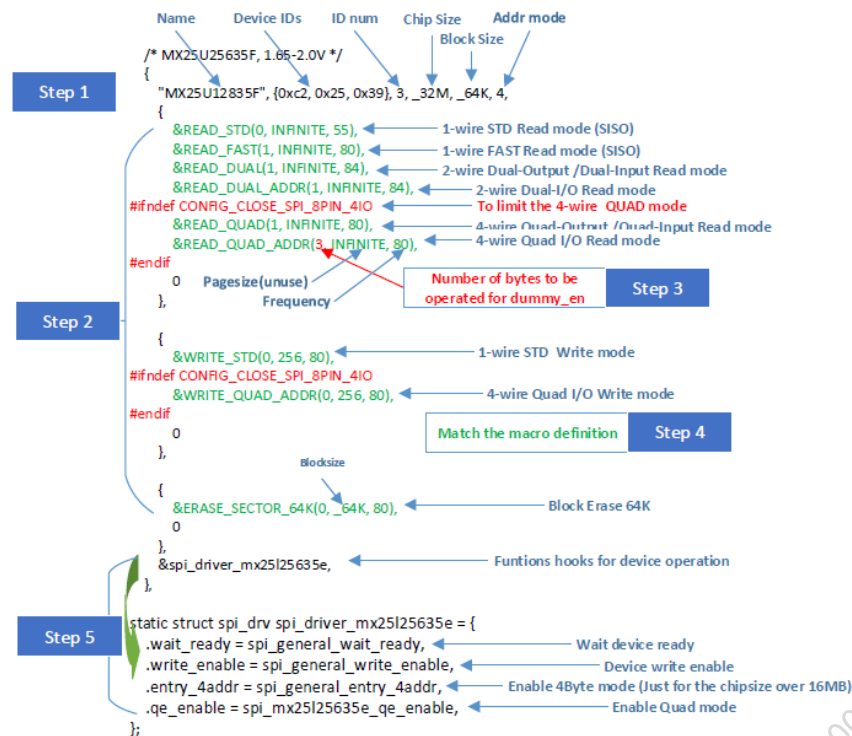
1.2.1.2 如何新增一颗 SPI Nor Flash

新移植一个 SPI Nor Flash 器件，简单来说就是往 *hifmc_spi_nor_ids.c* 的 ID 注册信息结构体中增加一个 ID 节点，如图 1-1 所示。

接下来我们以 **Macronix MX25U25635F** 为例，详细讲述如何在 ID 表里面新增一颗 SPI Nor Flash。



图1-1 非标驱动中 SPI Nor ID 注册信息详解



步骤 1. 查阅 **MX25U25635F** 的器件手册，新增 ID 节点，在节点的相应位置填 ID 信息。

找到 **9Fh** 命令下的 **ID** 信息

从图中可以看出该器件有 3 个 ID，分别是：0xC2、0x20、0x19。

Table 6. ID Definitions

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

获取器件的 chip size、block size 信息

- 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

步骤 2. 从器件手册的 **Features** 章节中获取器件所支持的接口类型。



- 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

通过查询表 1-2，可以匹配出 **MX25U25635F** 支持的接口及其对应的驱动中的宏定义，如表 1-2 底色是灰色的行所示。

此外，Erase 操作我们默认匹配 Block Erase 64K 命令。

表1-2 SPI 接口匹配查询表

器件支持的接口	命令字	对应的 FMC 的接口	驱动中的宏定义
READ	03h	Standard SPI	READ_STD
FAST READ	0Bh	Standard SPI	READ_FAST
2READ	3Bh	Dual-Output/Dual-Input SPI	READ_DUAL
DREAD	BBh	Dual I/O SPI	READ_DUAL_ADDR
4READ	6Bh	Quad-Output/Quad-Input SPI	READ_QUAD
QREAD	EBh	Quad I/O SPI	READ_QUAD_ADDR
PP	02h	Standard SPI	WRITE_STD
DPP(page program)	A2h	Dual-Output/Dual-Input SPI	WRITE_DUAL
2PP	D2h	Dual I/O SPI	WRITE_DUAL_ADDR
QPP	32h	Quad-Output/Quad-Input SPI	WRITE_QUAD
4PP	38h	Quad I/O SPI	WRITE_QUAD_ADDR
Block Erase 4K	02h	Standard SPI	ERASE_SECTOR_4K
Block Erase 32K	52h	Standard SPI	ERASE_SECTOR_32K
Block Erase 64K	D8h	Standard SPI	ERASE_SECTOR_64K
Block Erase 128K	D8h	Standard SPI	ERASE_SECTOR_128K



器件支持的接口	命令字	对应的 FMC 的接口	驱动中的宏定义
Block Erase 256K	D8h	Standard SPI	ERASE_SECTOR_25 6K

步骤 3. 确定接口的 dummy 值和工作时钟。

- Dummy num 的确认

根据 SPI Nor Flash 的 SPI 接口的定义的特点，可以有以下结论。

- Standard SPI STD Read 和所有 Write 及 Erase 接口的 Dummy 值为 0;
- Dual-Output/Dual-Input SPI 和 Quad-Output/Quad-Input SPI 接口 Dummy 值为 1;
- Dual I/O SPI 和 Quad I/O SPI 的接口 Dummy 值需要参考手册计算:

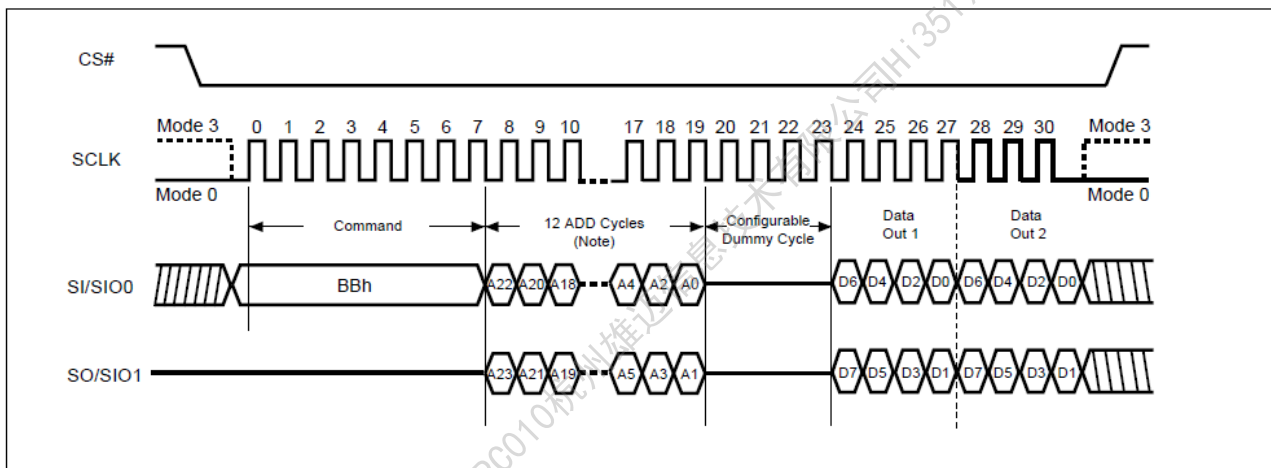
🔑 窍门

因为有的器件 dummy 的计算还要依据 I/O 的驱动能力 (详见手册 Output Driver Strength Table) 做调整，比较复杂，所以建议用户在移植器件接口时可以适当舍去 Dual I/O SPI (2READ/2PP) 和 Quad I/O SPI(4READ/4PP)两种接口类型。对于性能的需求不是很苛刻的场景，这两种接口相对于 Dual-Output/Dual-Input SPI 和 Quad-Output/Quad-Input SPI 并不会有很大的性能收益。

这里只举例在器件默认 DC 值为 0x00 时的 dummy 值计算方法：

a) Dual I/O SPI Read

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

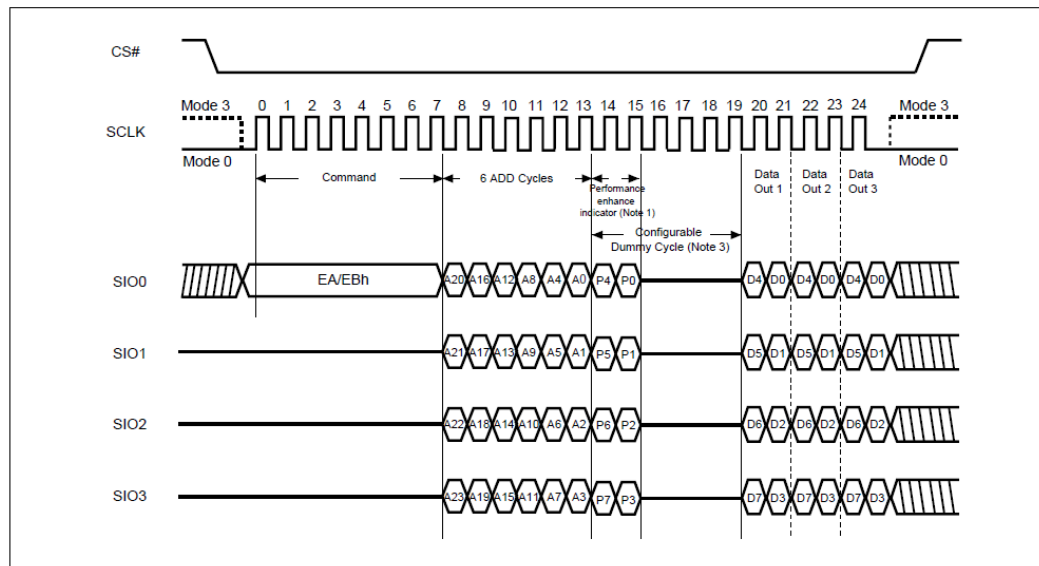


从器件手册中的波形图可以看出 Dual I/O SPI Read 需要 20-23 的 4 个 dummy cycle 的时钟周期，又因为接口是两线，所以相当于要有 8 dummy cycle bit，相当于 1 dummy cycle Byte。根据芯片手册 dummy_num 的定义：&READ_DUAL_ADDR(1, INFINITE, 84)中 dummy 值填 1。

b) Quad I/O SPI Read



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



从器件手册中的波形图可以看出 Quad I/O SPI Read 需要 14-15 的 2 个 cycle 的 Performance enhance (有的厂家的器件是 M7-M0) 的时钟周期, 16-19 的 4 个 dummy cycle 的时钟周期, 加起来 6 个 cycle。又因为接口是四线, 所以相当于要有 24 dummy cycle bit, 相当于 3 dummy cycle Byte。根据芯片手册 dummy_num 的定义: &READ_QUAD_ADDR (3, INFINITE, 80) 中 dummy 值填 3。

- 接口工作时钟

接口 &READ_QUAD_ADDR(3, INFINITE, 80) 中的工作频率是依据每个手册里面的 AC CHARACTERISTICS 表获取。

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
fRSCLK	fR	Clock Frequency for READ instructions			50	MHz
fTSCLK	fT	Clock Frequency for 2READ instructions			84 ⁽⁷⁾	MHz
	fQ	Clock Frequency for 4READ instructions			84 ⁽⁷⁾	MHz



注意

从手册里面可以看到器件所能支持的最高接口时钟 84MHz/133MHz, 跟以上举例的 MX25U25635F ID 节点的时钟信息有出入, 这是因为目前 FMC 对于 1.8V IO 接口时钟最高只能支持 150MHz (也就是 75MHz), 所以这里填个 80MHz, 一方面是为了限制接口时钟, 一方面是为了匹配步骤 4 中的宏定义。

步骤 4. 当根据上面几个步骤填好如下图接口的信息之后, 要匹配宏定义。



```
{
    &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
},
```

在 hifmc_spi_nor_ids.c 源代码路径开头几行中匹配宏定义。

```
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);
```

步骤 5. 匹配完接口信息，最后一步要匹配器件相关功能函数钩子 &spi_driver_mx25l25635e (结构体 struct spi_drv)，我们需要关注的以下几个接口赋值：

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

目前驱动中对于大部分 SPI Nor Flash 厂家的所使用的功能函数都是有匹配的，如果新增器件是表格中所列厂家的器件可以尝试使用现成的，不需要额外匹配。

表1-3 相关功能函数与厂家对应表

厂家	struct spi_drv 赋值	源文件
GigaDevice (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
Spansion	&spi_driver_s25fl256s	hifmc100_spi_s25fl256s.c



厂家	struct spi_drv 赋值	源文件
Winbond	&spi_driver_w25q256fv	hifmc100_spi_w25q256fv.c
其他通用适配	&spi_driver_general	hifmc100_spi_general.c

- .wait_ready = spi_general_wait_ready, 等待器件 ready。

通过读器件状态寄存器的 WIP bit 来判断器件是否处于空闲状态，一般所有 SPI Nor Flash 的满足这个机制。

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, 写使能接口。

通过写器件状态寄存器的 WEL bit 来改变器件是否可操作，一般所有 SPI Nor Flash 的满足这个机制。

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.

说明

因为各个厂家对器件状态寄存器的命名方法有差异，但是访问寄存器的命令字是一样的，所以下面我们使用命令字来区分 SPI Nor Flash 的状态寄存器：05 状态寄存器、35 状态寄存器和 15 状态寄存器。

如上所诉 WIP bit 和 WEL bit 基本上在所有的 SPI Nor Flash 的 05 状态寄存器上的位置都一致，都是第 0 和第 1 个 bit 位置；

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

但下面的 3Byte/4Byte 切换和 QE 使能所涉及的 35 状态寄存器和 15 状态寄存器不同厂家和器件存在很大的差异，所以下面将以一个专题来讲解。



---结束

1.2.1.3 3Byte/4Byte 切换和 QE 使能专题

.entry_4addr = spi_general_entry_4addr 和 qe_enable = spi_mx25l25635e_qe_enable，这两个接口内容比较多，作为一个专题来讲解。

说明

用户在移植新的 Flash 器件时，需要根据器件的具体情况来适配 3Byte/4Byte 模式切换的接口。

- .entry_4addr = spi_general_entry_4addr，切换器件 3Byte/4Byte 模式的接口，这个接口只有那个**器件容量大于 16MB**才会调用。

一般的器件 SPI Nor Flash 都是提供 Enter 4-byte mode (EN4B)和 Exit 4-byte mode (EX4B)来实现器件 3Byte/4Byte 模式的切换，并通过判断 15 状态寄存器的 bit5 来判断是否切换成功。

图1-2 15 状态寄存器

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

而其他器件在切换 3Byte/4Byte 模式之间的差异总体上分为以下三类：

- 切换 3Byte/4Byte 模式的方式不一样：

典型的就是 Spansion（现在的 Cypress）的.entry_4addr = spi_s25fl256s_entry_4addr，不是通过发 EN4B 和 EX4B 命令来切换 3Byte/4Byte 模式的，而是通过改变 Bank Address Register 的 bit7 的值来切换。

图1-3 Bank 地址寄存器

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

- 通过命令来切换 3Byte/4Byte 模式，但是检查是否切换成功的寄存器不是 15 状态寄存器，典型的就是 Micron 的 SPI Nor Flash 的.entry_4addr =



spi_micron_entry_4addr 接口是通过检查器件的 FLAG STATUS REGISTER 的 bit0 来判断切换 3Byte/4Byte 模式是否成功。

- 不能简单通过 EX4B 命令来将 4Byte 模式切换回 3Byte 模式，得通过发复位命令来切换，典型的就 Winbond 的 W25Q256 系列器件

.entry_4addr = spi_w25q256fv_entry_4addr, 的接口得用直接发复位组合命令的方式来将 4Byte 模式切换回 3Byte 模式。

- .qe_enable = spi_mx25l25635e_qe_enable, 器件四线使能的函数接口。



注意

驱动中是通过 CONFIG_CLOSE_SPI_8PIN_4IO 宏来打开和关闭四线接口，BVT 的很多平台上是默认不开放四线接口的，这是因为对于 8 PIN 封装的器件 IO_3 和 RESET 引脚是复用在一起的，我们一般默认复用为 RESET 功能。

使能器件的 QE (QUAD ENABLE) bit 的差异主要是不同厂家不同器件的 QE bit 分布不一样导致的。这些不一样只能通过实际的器件手册来分析。

接下来我们以 Macronix MX25U25635F 为例详细讲解一下这个接口的函数实现，其他器件都是大同小异。



图1-4 Macronix MX25U25635F 4 线使能函数实现

```
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); ← Judge whether 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 value of QE bit

    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);
    if (MX_SPI_NOR_GET_QE_BY_SR(status) == op) Tag 3
        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 one-to-one correspondence command between read and write status register, please check from datasheet

Check whether the set successfully

如果需要新增一个 QE 使能接口，可以参考上面的代码实现，代码流程不需要更改，需要关注的是代码中的几处标识：

- tag1：读 QE bit 所在器件寄存器的命令；
- tag2：写 QE bit 所在器件寄存器的命令；
- tag3：判断 QE bit 是否设置成功。

1.2.1.4 u-boot-2016.11 下大于 16MB SPI Nor Flash 的移植

目前使用 u-boot-2016.11 版本的芯片平台优化了 HiFMCV100 控制器，解除需要硬件接 reset 管脚的限制。但对于容量大于 16MB 的器件必须要求支持 4 Byte 命令。目前，驱动中只适配了 Winbond、Macronix (MXIC) 和 Micron 三个厂家，用户若想适配其他厂家的支持 4Byte 命令的器件，在 drivers/mtd/spi/hifmc100/hifmc_spi_nor_ids.c 路径下，找到下面代码段，增加相应的 case 代码分支。



```
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");
```



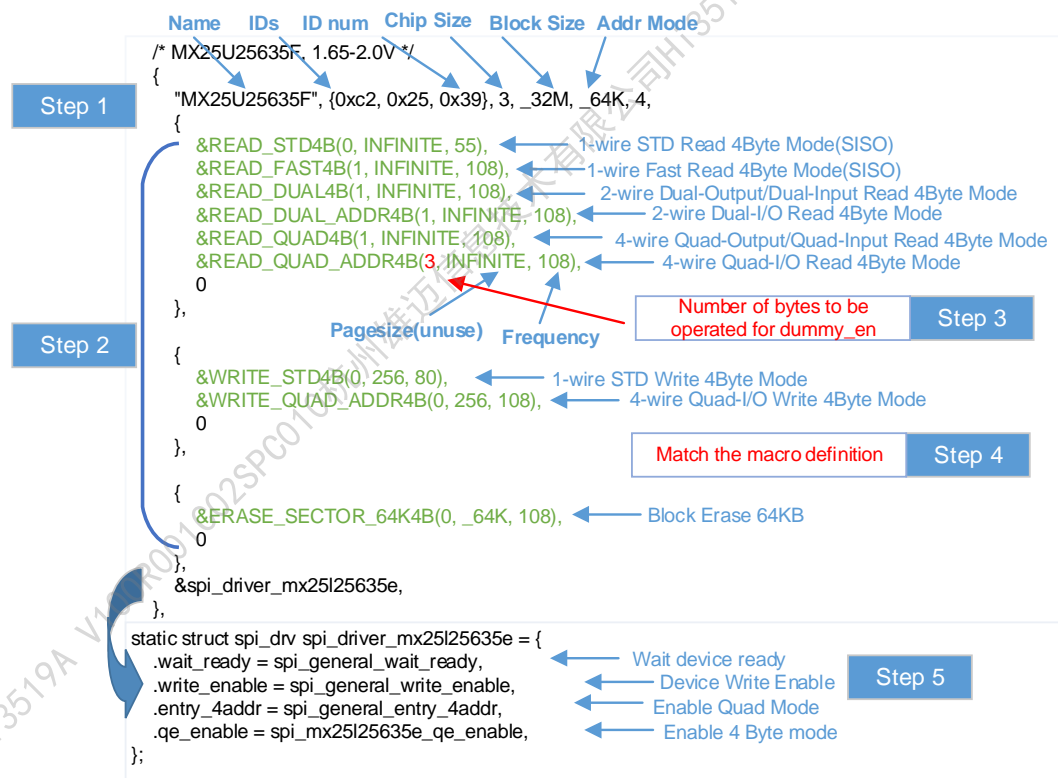
注意

如果移植的大于 16MB 的 SPI Nor Flash 器件不支持 4Byte 命令，请参考 1.2.1.2 “如何新增一颗 SPI Nor Flash”进行移植，但是必须把接口限制成 2 线且硬件连 reset 引脚。

如图 1-5，接下来我们还是以 **Macronix MX25U25635F** 为例，详细讲述如何在 ID 表里面新增一颗支持 4Byte 命令的 SPI Nor Flash。

因为步骤 1,3,4,5 分别跟 1.2.1.2 “如何新增一颗 SPI Nor Flash” 章节一样，这里不再赘述，下面主要阐述步骤 2 的差异。

图1-5 非标驱动中 SPI Nor ID 注册信息详解





从器件手册的 **command set** 章节中确认是否支持 4Byte 命令。

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

通过查询表 1-4，可以匹配出 **MX25U25635F** 支持的接口及其对应的驱动中的宏定义，如表 1-4 底色是灰色的行所示。

此外，Erase 操作我们默认匹配 Block Erase 64K4B 命令。

表1-4 SPI 接口匹配查询表

器件支持的接口	命令字	对应的 FMC 的接口	驱动中的宏定义
READ4B	13h	Standard SPI	READ_STD4B
FAST READ4B	0Ch	Standard SPI	READ_FAST4B
2READ4B	3Ch	Dual-Output/Dual-Input SPI	READ_DUAL4B
DREAD4B	BCh	Dual I/O SPI	READ_DUAL_ADDR4B
4READ4B	6Ch	Quad-Output/Quad-Input SPI	READ_QUAD4B
QREAD4B	EBh	Quad I/O SPI	READ_QUAD_ADDR4B
PP4B	12h	Standard SPI	WRITE_STD4B
4PP4B	3Eh	Quad I/O SPI	WRITE_QUAD_ADDR4B
BE4K4B Block Erase 4K4B	21h	Standard SPI	ERASE_SECTOR_4K4B
BE32K4B Block Erase 32K	5Ch	Standard SPI	ERASE_SECTOR_324B K
BE4B Block Erase 64K	DCh	Standard SPI	ERASE_SECTOR_64K4B

1.2.2 SPI-Nor 内核下的移植

1.2.2.1 相关驱动路径

目前内核下的 SPI Nor 驱动使用的是标准化的版本。

驱动路径：drivers/mtd/spi-nor/

标准化 SPI Nor 驱动新器件移植有关系的目录如表 1-5 所示。



表1-5 标准化 SPI Nor 器件相关的主要目录结构

Hifmc100 目录/文件名	描述
spi-nor.c	标准驱动框架，定义 SPI Nor 驱动中主要的数据结构；定义 SPI Nor 器件 id 表，移植新 SPI Nor 器件主要修改的文件；区分不同器件 3/4byte 切换及其四线使能等操作。

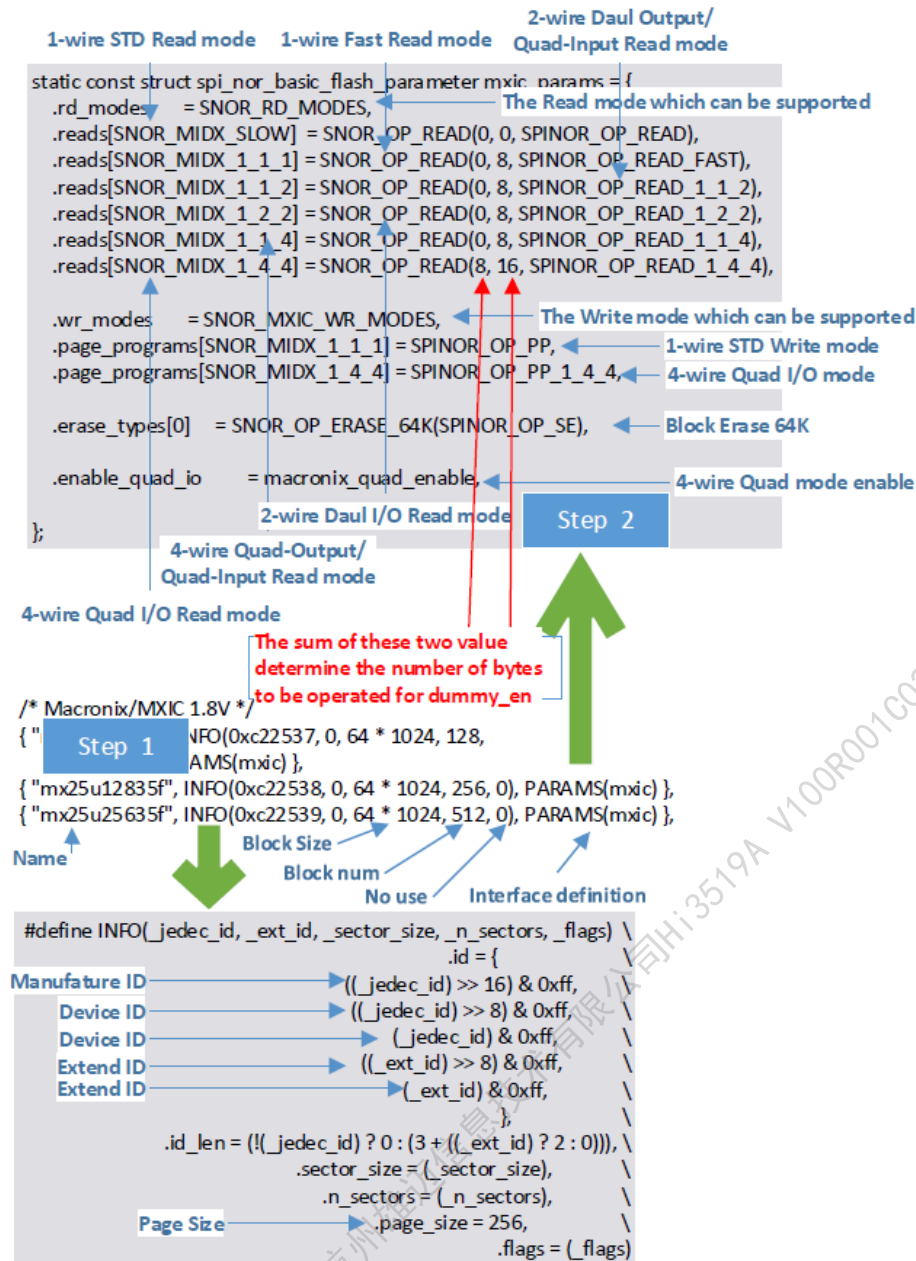
1.2.2.2 如何新增一颗 SPI Nor Flash

新移植一个 SPI Nor Flash 器件，简单来说就是往 *spi-nor.c* 的 ID 注册信息结构体中增加一个 ID 节点，如图 1-6 所示。

接下来我们以 **Macronix MX25U25635F** 为例，详细讲述如何在 ID 表里面新增一颗 SPI Nor Flash。



图1-6 标准驱动中 SPI Nor ID 注册信息示例



注意

不像 SPI Nor 的非标准化驱动，可以对同一个厂家的不同器件适配不同的接口函数，标准驱动中，只对厂家做出区分，如果同一个厂家不同的器件在接口类型的匹配上存在差异，可能需要限制接口来兼容。

步骤 1. 查阅 **MX25U25635F** 的器件手册，新增 ID 节点，在节点的相应位置填信息；



找到 **9Fh** 命令下的 ID 信息：

Table 6. ID Definitions

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

从图中可以看出该器件有 3 个 ID，分别是：0x2C、0x20、0x19。

获取器件的 **chip size**、**block size** 信息，分别是 32MB、64KB：

- 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

步骤 2. 由于 FMC 已经下表大部分厂家的 SPI Nor Flash 的接口类型、dummy 值、3Byte/4Byte 切换和 QE 使能做出限定。当新增器件是下表所列出的器件时，请直接使用厂家对于的 PARAMS 宏定义。

厂家	宏定义
ESMT	PARAMS(esmt)
GigaDevice	PARAMS(gd)
Macronix/MXIC	PARAMS(mxlc)
Micron	PARAMS(micron)
Spansion	PARAMS(spansion)
Winbond	PARAMS(winbond)
Paragon	PARAMS(paragon)

步骤 3. 确定的工作时钟。



注意

版本默认发布地工作时钟的值是经过兼容性测试的，所以这个值不建议修改。

参考器件手册中 AC CHARACTERISTICS 表获取器件的工作时钟，根据实际使用的接口类型选择相应的时钟：



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

Symbol	Alt.	Parameter	Min.	Typ.	Max.	Unit
fCLK	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 ⁽⁷⁾	MHz
	fQ	Clock Frequency for 4READ instructions			84 ⁽⁷⁾	MHz

说明

由于目前兼容性列表里面的所有器件都支持 FAST READ (由 DTS 中的 m25p,fast-read 指定) , 所以上图中 READ 命令 (对应 STD READ) 支持的最大时钟 50MHz 可以忽略。从器件手册上, 我们可以看到, 器件所能支持的最大时钟频率是 133MHz, 但是 2READ/4READ 接口只能支持到 84MHz, 由于对于该器件我们是全接口支持 (通过 PARAMS(mx1c)定义的 都是全接口支持) , 所以我们设置器件的时钟频率应该取较小的 84MHz 时钟, 如下图 (路径: drivers/mtd/spi-nor/spi-nor.c) :

```
{ "mx25u25635f", INFO(0xc22539, 0, 64 * 1024, 512,
    SPI_NOR_QUAD_READ | SPI_NOR_4B_OPCODES), PARAMS(mx1c), CLK_MHZ_2X(84) },
```

说明

如图 1-7, PARAMS(mx1c)全接口支持表示的是 MX1C 的器件支持 1-wire (STD 和 FAST READ)、2-wire (DUAL-OUTPUT/DUAL-INPUT 和 DUAL-I/O)、4-wire (QUAD-OUTPUT/QUAD-INPUT 和 QUAD-I/O) 所有接口。当 ID 节点中 PARAMS(mx1c)参数缺省时, 器件的接口类型由节点里面的 SPI_NOR_QUAD_READ/SPI_NOR_DUAL_READ 指定。所以, 当新增器件无法全接口匹配时, 可以使用 SPI_NOR_QUAD_READ/SPI_NOR_DUAL_READ 指定器件的接口。

图1-7 PARAMS(mx1c)释义

```
static const struct spi_nor_basic_flash_parameter mx1c_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_MX1C_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 64KB

    .enable_quad_io = macronix_quad_enable, ← Enable Quad Mode
};
```

确定完器件所能支持的最大时钟, 接下来就要确定芯片 CRG 提供给 FMC 控制器的最大时钟, 参考对应芯片的芯片手册的系统章节, 假如 FMC 的时钟源最佳可以匹配到 150MHz 的时钟, 在 arch/arm/boot/dts/hi35xx-demb.dts 文件中, 找到以下设备节点:



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

并更改 spi-max-frequency = <150000000>; 的值，**注意：**真实填写到这个节点的频率值都是 2X 时钟，也就是说当 spi-max-frequency 参数填写 150000000（Hz）时，CRG 实际提供的时钟是 75MHz。

综合上面提到的**器件时钟**和**CRG 时钟**，这两个的最小值就是实际接口时钟 75MHz。

---结束

1.3 SPI Nand Flash 器件的移植

1.3.1 相关驱动路径

目前 SPI Nand 驱动使用的都是没有标准化的版本。

SPI Nand 驱动的路径：drivers/mtd/nand/hifmc100/

非标准化 SPI Nand 驱动新器件移植有关系的目录如下表 1-6 所示。

表1-6 SPI Nand 器件相关的主要路径

Hifmc100 目录/文件名	描述
hifmc_spi_nand_ids.c	SPI Nand 器件 ID 表，移植新 SPI Nand 器件主要修改的文件。主要包含 SPI Nand 器件的读写擦除参数；
hifmc100_spi_general.c	SPI Nand 驱动大部分器件通用的功能驱动代码，包含写使能操作、QE bit 使能函数。

1.3.2 如何新增一颗 SPI Nand Flash



注意

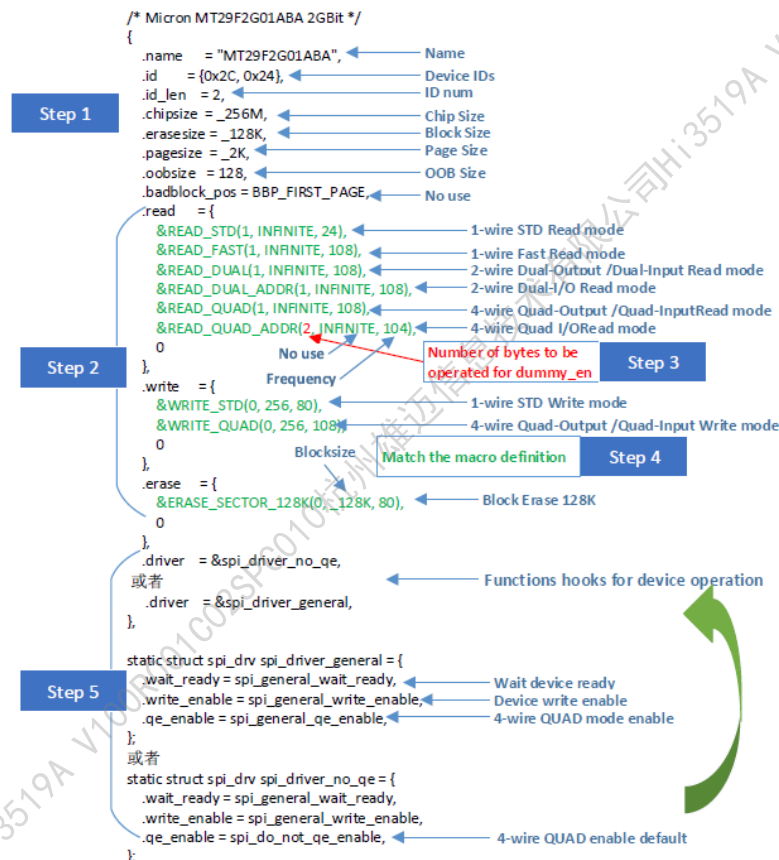
FMC 控制器自身集成 ECC 纠错功能，FMC 启动之后会关掉 SPI Nand Flash 器件的 ECC，而关闭 ECC 这个动作，只能支持将 B0h Feature 寄存器的 bit4 置成 0。所以，判断一颗 SPI Nand Flash 器件 FMC 能不能支持的第一步就是先看看这颗器件的 B0h Feature 寄存器的 bit4 是否是 ECC_EN bit，如果不是，那就不支持。

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	COH	Reserved	Reserved	ECSS1	ECSS0	P_FAIL	E_FAIL	WEL	OIP

新移植一个 SPI Nand Flash 器件，简单来说就是往 *hifmc_spi_nand_ids.c* 的 ID 注册信息结构体中增加一个 ID 节点，如图 1-8 所示。

接下来我们以 Micron MT29F2G01ABA 为例，详细讲述如何在 ID 表里面新增一颗 SPI Nand Flash：

图1-8 非标驱动中 SPI Nand ID 注册信息示例





步骤 1. 查阅 **MT29F2G01ABA** 的器件手册，新增 ID 节点，在节点的相应位置填信息

找到 **9Fh** 命令下的 ID 信息：

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

从 ID 表里可以看出，MT29F2G01ABA 有 2 个 ID：0x2C、0x24。

获取器件的 **chip size**、**block size**、**page size**、**OOB size** 信息，其他信息不确定的情况下，请保持跟示例中的 ID 信息一致：

- 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)

步骤 2. 从手册的 **Features** 章节中获取器件所支持的接口类型：

- 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

通过查询表 1-7，可以匹配出 MT29F2G01ABA 支持的接口及其对应的驱动中的宏定义，如下表底色是灰色的行所示。根据步骤 1 中我们可以知道 MT29F2G01ABA 的 block size 是 128K，所以我们应该匹配 Block Erase 128K 命令。

表1-7 SPI 接口匹配查询表

器件支持的接口	命令字	对应的 FMC 的接口	驱动中的宏定义
READ	03h	Standard SPI	READ_STD
FAST READ	0Bh	Standard SPI	READ_FAST
2READ	3Bh	Dual-Output/Dual-Input SPI	READ_DUAL
DREAD	BBh	Dual I/O SPI	READ_DUAL_ADDR
4READ	6Bh	Quad-Output/Quad-Input SPI	READ_QUAD
QREAD	EBh	Quad I/O SPI	READ_QUAD_ADDR



器件支持的接口	命令字	对应的 FMC 的接口	驱动中的宏定义
PP	02h	Standard SPI	WRITE_STD
DPP	A2h	Dual-Output/Dual-Input SPI	WRITE_DUAL
2PP	D2h	Dual I/O SPI	WRITE_DUAL_ADDR
QPP	32h	Quad-Output/Quad-Input SPI	WRITE_QUAD
4PP	38h	Quad I/O SPI	WRITE_QUAD_ADDR
Block Erase 4K	02h	Standard SPI	ERASE_SECTOR_4K
Block Erase 32K	52h	Standard SPI	ERASE_SECTOR_32K
Block Erase 64K	D8h	Standard SPI	ERASE_SECTOR_64K
Block Erase 128K	D8h	Standard SPI	ERASE_SECTOR_128K
Block Erase 256K	D8h	Standard SPI	ERASE_SECTOR_256K

步骤 3. 确定接口的 dummy 值和工作时钟:

1. Dummy num 的确认

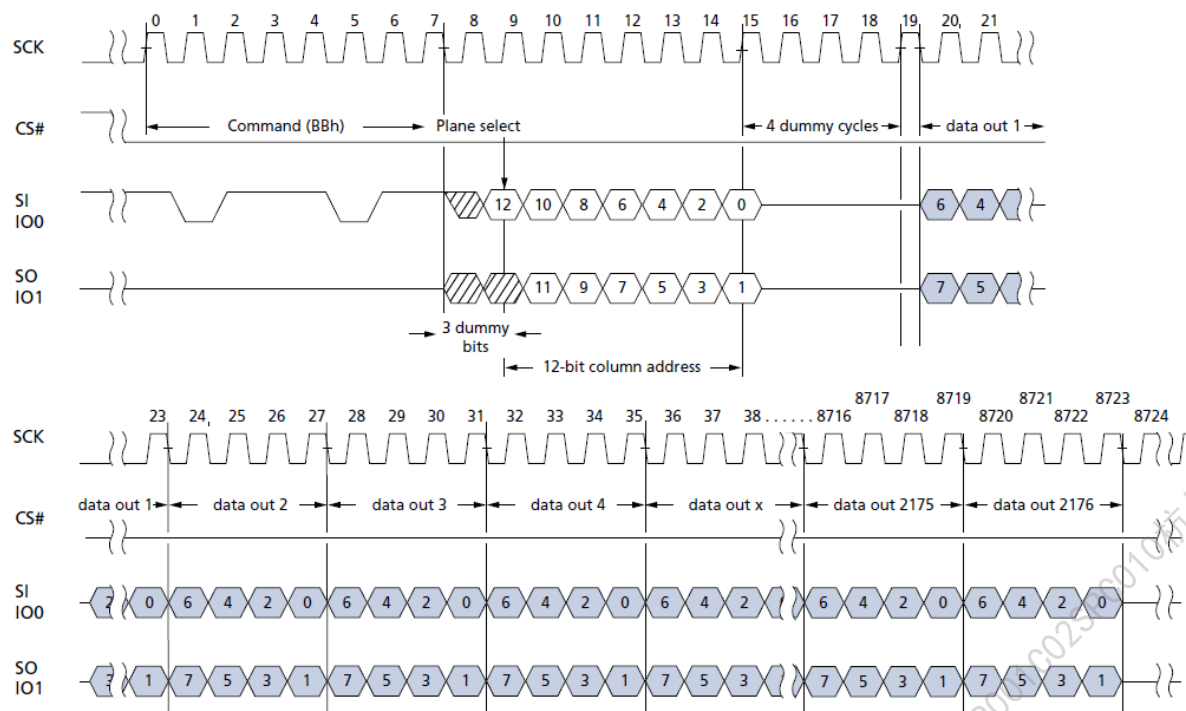
参照手册 SPI Nand Flash 的 SPI 接口的定义:

- (1) Standard SPI STD Read 和所有 Write 及 Erase 接口的 Dummy 值为 0;
- (2) Dual-Output/Dual-Input SPI 和 Quad-Output/Quad-Input SPI 接口 Dummy 值为 1;
- (3) Dual I/O SPI 和 Quad I/O SPI 的接口 Dummy 值要参考手册计算:

- Dual I/O SPI Read

图1-9 Dual I/O SPI Read 时序图

Figure 16: READ FROM CACHE Dual I/O



从器件手册中的波形图可以看出 Dual I/O SPI Read 需要 15-18 的 4 个 dummy cycle 的时钟周期，又因为接口是两线，所以相当于要有 8 dummy cycle bit，相当于 1 dummy cycle Byte。根据芯片手册 dummy_num 的定义：`&READ_DUAL_ADDR(1, INFINITE, 108)`中 dummy 值填 1。

- Quad I/O SPI Read

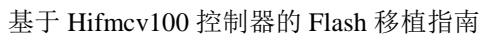
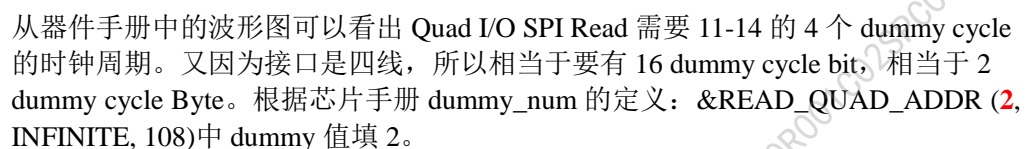


Figure 17: READ FROM CACHE Quad I/O



举个例子，接口`&READ_QUAD_ADDR(2, INFINITE, 108)`，中的工作频率是依据每个手册里面的 `AC CHARACTERISTICS` 表获取。

表1-8 AC CHARACTERISTICS 表

Table 17: AC Characteristics

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.

步骤 4. 当根据上面几个步骤填好如下图接口的信息之后，要匹配宏定义。



```
.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
},
```

在 hifmc_spi_nand_ids.c 源代码路径开头几行中匹配宏定义：

```
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);
```

步骤 5. 匹配完接口信息，最后一步要匹配器件相关函数钩子 &spi_driver_no_qe, (结构体 struct spi_drv)，我们需要关注的以下几个：

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

🔑 窍门

目前驱动中对于大部分 SPI Nand Flash 厂家的所使用的功能函数都是有匹配的，如果新增器件是表格中所列厂家的器件可以尝试使用现成的，不需要额外匹配：

厂家	struct spi_drv 赋值
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



厂家	struct spi_drv 赋值
Paragon	&spi_driver_general
All-flash	&spi_driver_general
TOSHIBA	&spi_driver_no_qe
HeYangTek	&spi_driver_general

- .wait_ready = spi_general_wait_ready, 等待器件 ready。

通过读器件 C0h feature 寄存器的 bit0 WIP 来判断器件是否处于空闲状态，目前所有所有 SPI Nand Flash 的满足这个机制。

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, 写使能接口。

通过写 C0h feature 寄存器的 bit1 WEL 来改变器件是否可操作，目前所有所有 SPI Nand Flash 的满足这个机制。

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.

说明

区别于 SPI Nor Flash，SPI Nand Flash 器件是不需要复位脚的，因为 FMC 会自动下发复位命令，所以，SPI Nand Flash 默认匹配为四线模式。但是，不是所有的厂家的 SPI Nand Flash 器件出厂时都默认四线，根据 QE bit 的情况分为两类：

图1-11 不带 QE bit 的 Feature 寄存器

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



图1-12 带 QE bit 的 Feature 寄存器

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	ECSS1	ECSS0	P_FAIL	E_FAIL	WEL	OIP

- .qe_enable = spi_do_not_qe_enable, 器件四线接口使能的函数接口:
Micron MT29F2G01ABA 器件默认上电是四线使能, 所以不需要使能四线。
但是, 有的 SPI Nand Flash 需要使能四线才能使用 QUAD 接口, 对应的函数接口是: .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); ← Judge whether 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 %s!\n", str[op]);
        return op;
    } ← Get the status register to check the value of QE bit

    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);

    reg = spi_nand_feature_op(spi, GET_OP, FEATURE_ADDR, 0);
    if ((reg & FEATURE_QE_ENABLE) == op) ← Check whether the set successfully
        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;
}
```




说明

当前市面上的 SPI Nand Flash 的 QE bit 使能都是遵循上面驱动中的操作。如果所新增的器件 QE bit 使能的方式有所差异，可以参考上面的驱动代码进行修改。

步骤 6. 将 ID 合入到 hifmc_spi_nand_ids.c 文件中后，编译烧写上电启动查看 U-boot 的打印信息，查看 ECC 是否跟手册要求的 ECC 类型匹配：

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



警告

驱动匹配出的 ECC 大小必须保证不小于 SPI Nand Flash 手册要求的 ECC 大小。建议用户在实际使用过程中尽量匹配更高的 ECC 类型，这样有利于提高器件的寿命。

在使用量产工具（spinand_product/nand_product）制作烧录镜像时，所输入 ECC type 参数：

ECC type ECC size:

- | | |
|---|-----------|
| 1 | 4bit/512B |
| 2 | 16bit/1K |
| 3 | 24bit/1K |
| 4 | 28bit/1K |

一定要匹配实际使用的 ECC type，可以从 u-boot 驱动打印信息中获取。

---结束

1.4 并口 Nand 驱动

1.4.1 相关驱动路径

- 并口 Nand 驱动使用是非标准化版本。
- 并口 Nand 驱动的路径：drivers/mtd/nand/hifmc100_nand/
- 并口 Nand 驱动新器件移植有关系的目录，如表 1-9 所示。

表1-9 并口 Nand 驱动 ID 移植相关的目录结构

Hifmc100 目录/文件名	描述
hifmc_nand_spl_ids.c	并口 Nand 特殊器件 ID 表，移植新并口 Nand 器件主要修改的文件。



1.4.2 并口 Nand ID 表系统



注意

FMC 控制器集成 ECC 纠错功能，不需要并口 Nand 器件自身进行 ECC 纠错。目前发现，有的厂家出厂的并口 Nand 器件自身携带 ECC 纠错功能，且还是无法关闭的，对于这种器件 FMC 不建议支持，一则需要下发特殊的命令字去读取 ECC 纠错的 bit 数，会增加驱动的复杂度，不利于维护；再则，实际使用过程中，发现经过两次 ECC (器件自身 ECC+FMC ECC) 之后会存在一些很难解释的出错现象。所以，对于这类并口 Nand 器件，不建议支持。

并口 Nand 驱动中包括了两个表：

- 公共 ID 表：内核 nand 适配层通过识别并解析 ID 信息，获取器件的 chip size、block size、page size、OOB size 信息。
- 特殊 ID 表：但是不是所有的并口 Nand 通过 nand 适配层识别出来的信息都能满足器件的实际需求，所以又定义了一套特别器件 ID 表。

公共 ID 表

读取的并口 Nand 的 ID 中，是读取好几个字节，一般最少是 4 个，新的芯片，支持 5 个甚至更多，从这些字节中，可以解析出很多相关的信息，比如此 Nand Flash 内部是几个芯片 (chip) 所组成的，每个 chip 包含了几片 (Plane)，每一片中的页大小，块大小，等等。如图 1-13 所示，在并口 Nand 的 ID 信息中，可以识别出此 flash 是 SLC 还是 MLC，以及 chip size、block size、page size、OOB size 信息。

图1-13 并口 Nand Flash ID 含义

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



特殊 ID 表

图1-14 并口 Nand ID 注册信息示例

```

{ /* 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:

- 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 如何新增一颗并口 Nand 器件

步骤 1. 首先，尝试一下匹配公共 ID 表。参考器件手册 Feature 章节，检查 u-boot（或者 kernel）下打印出的器件信息是否满足器件手册中的要求，如果满足那么就不需要额外增加 ID 信息来支持该器件。

FEATURES

- Organization

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

**警告**

- 驱动匹配出的 ECC 大小必须保证大于或等于 Nand Flash 手册要求的 ECC 大小。
- 在使用量产工具 (nand_product) 制作烧录镜像时，所输入 ECC type 参数：

ECC type ECC size:

1	4bit/512B
2	16bit/1K
3	24bit/1K
4	28bit/1K
5	40bit/1K
6	64bit/1K



一定要匹配实际使用的 ECC type，可以从 u-boot 驱动打印信息中获取。

- 步骤 2. 如果通过步骤一匹配公共 ID 表的方式，匹配到的器件信息不满足器件手册的要求，那么就必须特殊 ID 表中增加相应的 ID 信息。
- 步骤 3. 接下来我们以 **Spansion S34ML02G200TFI000** 为例，详细讲述如何在 ID 表里面新增并口 Nand Flash：
- 步骤 4. 详细阅读前两个小节的内容，找到相应的 ID 表的路径，并了解 ID 表信息的含义；
- 步骤 5. 查阅 S34ML02G200TFI000 器件手册，找到 90h 命令下的 ID 信息：

Table 3.3 Read ID for Supported Configurations

Density	Org	V _{CC}	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

可以看出 **S34ML02G200TFI000** 有 5 个 ID：0x01、0xDA、0x90、0x95、0x46。

- 步骤 6. 从手册的 **Features** 章节中获取器件信息，新增 ID 节点，在节点的相应位置填信息。

■ 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

根据上述信息填写相应的 ID 表，如下：



```
/* ***** 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,
},
```

步骤 7. 将 ID 合入到 hifmc_nand_spl_ids.c 文件中后，编译烧写上电启动查看 U-BOOT 的打印信息，查看 ECC 是否跟手册要求的 ECC 类型匹配（必须大于手册要求的 ECC 类型）。

■ Reliability

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



注意

对于 Spansion S34ML02G200TFI000，虽然通过匹配公共 ID 表也能满足手册中 ECC 类型的要求，但是由于 FMC 在 OOBSize 为 128Byte 情况下，可以匹配到更高的 24bit/1K。这边我们会建议用户在实际使用过程中尽量匹配更高的 ECC 类型，这样有利于提高器件的寿命。

---结束



2 常见问题整理

2.1 SPI Nor Flash 3byte/4byte 启动和 SPI Nand Flash 1/4 线启动

3Byte/4Byte 地址位宽这个概念是针对于 SPI Nor 的。芯片在上电启动的时候，会去读取 SPI Nor Flash 的前 1MB 空间，单板上的 3Byte/4Byte 决定了控制器读取时下发的地址宽度。

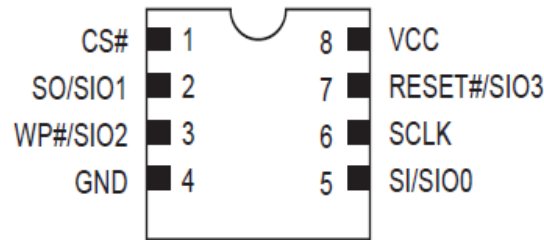
- 3Byte 地址位宽最大的寻址空间是 16MB，所以对于容量不大于 16MB 的 SPI Nor Flash 的 3Byte/4Byte 模式拨码必须是 3Byte；
- 对于大部分 32MB 及其更大容量的 SPI Nor Flash，出厂时器件默认是 3Byte 模式，意味着只能访问前 16MB 空间，所以此时 3Byte/4Byte 模式拨码必须是 3Byte；
- 对于一小部分 32MB 及其更大容量的 SPI Nor Flash，比如 MXIC MX25L25735E 和 Winbond W25Q257FV，出厂时器件默认是 4Byte 模式，这类器件 3Byte/4Byte 模式拨码必须是 4Byte。

1/4 线启动是针对 SPI Nand Flash 的，这里必须强调 SPI Nor Flash 是默认 1 线启动，几乎所有的 SPI Nand Flash 都支持 1 线和 4 线，4 线接口向下兼容 1 线。1/4 线拨码取决于硬件设计：

- 当图 2-1 SIO3 接到芯片上有效 IO 上时，表示支持 4 线，1/4 线拨码应拨成 4 线启动，可以得到更快的启动速度；
- 当图 2-1 SIO3 没有接有效 IO 时，这种场景是有的，因为 BVT 的产品 SPI Nor 和 SPI Nand 是共焊盘的，而 SPI Nor Flash IO3 一般复用成 reset 信号，此时 1/4 线拨码应拨成 1 线启动。



图2-1 8-WSO 封装



2.2 Hibern 显示烧写成功，但串口没打印

实际使用过程中，我们都会出现这样一个场景，我们可能会遇到这样一个场景，命名 Hibern 提示写成功，但是串口却狂打空格。一般情况下有两个情况可能导致这种问题：

- SPI Nor 3Byte/4Byte 问题

确认 3Byte/4Byte 问题的方法是，Fastboot 先选择烧写 DDR，启动之后读写数据，如果发现读出来的数据跟原始数据对比发现数据发现错位，基本上可以确认是 3Byte/4Byte 接口问题。还有一种厂家是掉电能启动，发复位命令发现 Flash 启动不了，一般也是这个问题。

图2-2 SPI Nor 3Byte/4Byte 问题数据对比图

00000000	17 05 00 EA 14 F0 9F E5 14 F0 9F E5 14 F0 9F E5	...è.ðYä.ðYä.ðYä	00000000	FF 17 05 00 EA 14 F0 9F E5 14 F0 9F E5 14 F0 9F	ÿ...è.ðYä.ðYä.ðYä
00000010	14 F0 9F E5 14 F0 9F E5 14 F0 9F E5 14 F0 9F E5	.ðYä.ðYä.ðYä.ðYä	00000010	E5 14 F0 9F E5 14 F0 9F E5 14 F0 9F E5 14 F0 9F	ä.ðYä.ðYä.ðYä.ðYä
00000020	A0 16 80 80 00 17 80 80 60 17 80 80 C0 17 80 80	.ëë..ëë'.ëëÄ.ëë	00000020	E5 A0 16 80 80 00 17 80 80 60 17 80 80 C0 17 80	ä'.ëë..ëë'.ëëÄ.ëë
00000030	20 18 80 80 80 18 80 80 E0 18 80 80 78 56 34 12	.ëëë.ëëä.ëëxV4.	00000030	80 20 18 80 80 80 18 80 80 E0 18 80 80 78 56 34	ë'.ëëë.ëëä.ëëxV4.
00000040	50 00 04 12 00 00 00 00 00 00 00 00 FD 00 00 00	P.....ÿ..	00000040	12 50 00 04 12 00 00 00 00 00 00 00 FD 00 00 00	.P.....ÿ..
00000050	54 00 04 12 00 00 00 00 00 00 00 00 FD 00 00 00	T.....ÿ..	00000050	00 54 00 04 12 00 00 00 00 00 00 00 FD 00 00 00	.T.....ÿ..
00000060	00 00 00 00 00 00 00 00 64 00 00 00 00 00 00 00d.....	00000060	00 00 00 00 00 00 00 00 00 64 00 00 00 00 00d.....
00000070	00 00 04 12 00 00 00 12 00 00 00 00 FD 00 00 00ÿ..	00000070	00 00 04 12 00 00 00 12 00 00 00 00 FD 00 00 00ÿ..
00000080	04 00 04 12 45 61 00 01 00 00 00 00 FD 00 00 00Ea.....ÿ..	00000080	00 04 00 04 12 45 61 00 01 00 00 00 FD 00 00 00Ea.....ÿ..
00000090	98 01 04 12 06 00 00 00 00 00 00 00 15 00 00 00	~.....ÿ..	00000090	00 98 01 04 12 06 00 00 00 00 00 00 15 00 00	~.....ÿ..
000000A0	30 00 04 12 00 00 00 12 00 00 00 00 FD 00 00 00	0.....ÿ..	000000A0	00 30 00 04 12 00 00 00 12 00 00 00 00 FD 00 00	0.....ÿ..
000000B0	34 00 04 12 15 C2 00 00 00 00 00 00 FD 00 00 00	4....Ä.....ÿ..	000000B0	00 34 00 04 12 15 C2 00 00 00 00 00 00 FD 00 00	4....Ä.....ÿ..
000000C0	98 01 04 12 49 00 00 00 00 00 00 00 45 18 00 00	~...I.....E...	000000C0	00 98 01 04 12 49 00 00 00 00 00 00 45 18 00	~...I.....E...
000000D0	98 01 04 12 01 00 00 00 00 00 00 00 0D 00 00 00	~.....ÿ..	000000D0	00 98 01 04 12 01 00 00 00 00 00 00 0D 00 00	~.....ÿ..
000000E0	98 01 04 12 00 00 00 00 00 00 00 00 05 10 00 00	~.....ÿ..	000000E0	00 98 01 04 12 00 00 00 00 00 00 00 05 10 00	~.....ÿ..
000000F0	58 01 04 12 FF 00 00 00 00 00 00 00 00 3D 00 00	X...ÿ.....=	000000F0	00 58 01 04 12 FF 00 00 00 00 00 00 00 3D 00	.X...ÿ.....=
00000100	00 00 00 00 10 02 00 00 00 00 00 00 1D 40 00 000.....	00000100	00 3D 92 00 EB 00 50 50 E2 03 00 00 1A 04 10 A0	.='..ë.PPä.....
00000110	50 00 04 12 7C FF 01 00 00 00 00 00 8D 00 00 00	P... ÿ.....	00000110	E1 3C 08 9F E5 0D 23 00 EB 09 00 00 EA 1F 00 55	ä<.Vä.#...ë..U
00000120	54 00 04 12 7C FF 01 00 00 00 00 00 8D 00 00 00	T... ÿ.....	00000120	E3 07 00 00 9A 2C 60 A0 E3 05 20 A0 E1 96 74 26	ä...S,' ä. ä-t&
00000130	00 00 00 00 00 00 00 F4 01 00 00 00 00 00 00 00ö.....	00000130	E0 04 10 A0 E1 1C 08 9F E5 2C 30 96 E5 03 23 00	ä..ä..Vä,0-ä.#
00000140	4C 01 04 12 03 00 00 00 00 00 00 00 0D 28 00 00	L.....(.....	00000140	EB 2C 50 96 E5 00 00 54 E3 05 00 00 1A 2C 50 87	ë,P-ä..Tä....,P#
00000150	48 01 04 12 01 00 00 00 00 00 00 00 0D 10 00 00	H.....ÿ.....	00000150	E5 2C 10 A0 E3 24 00 9D E5 01 40 A0 E3 BB 8F 00	ä.. ä\$.ä.ä.ä..
00000160	90 00 05 12 FF FF FF 00 00 00 00 FD 00 00 00 00ÿÿÿÿ.....ÿ..	00000160	EB DC FF FF EA 04 30 9A E5 DC 57 9F E5 00 00 53	eÜÿÿÿ.0sääÜVä..S
00000170	94 00 05 12 C9 91 73 81 00 00 00 00 FD 00 00 00	"...É'st.....ÿ..	00000170	E3 0E 00 00 0A B4 37 9F E5 DC 07 9F E5 08 10 93	ä....'7VäU.Vä.."

SPI Nor 3Byte/4Byte 问题处理方法：检查一下单板拨码，或者要检查一下软件复位流程是否成功。

- SPI Nor/SPI Nand 4 线问题

确认 4 线接口问题的方法是，Fastboot 先选择烧写 DDR，启动之后读写数据，如果发现读出来的数据跟原始数据对比出现很多这种 1bit 翻转的情况，基本上可以确认是 4 线接口问题。



图2-3 SPI Nor/SPI Nand 4 线问题数据对比图

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	8889fffc	88888888	8888888d	18	12040050	0001ff7c	00000000	0000008d
19	9a8c88dc	8889fffc	88888888	888d8888	19	12040054	0001ff7c	00000000	008d0000
20	88888888	88888888	888889fc	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

SPI Nor/SPI Nand 4 线问题处理方法：检查一下硬件的接口连线。

- IO3 比较接有效的 IO 信息线；
- IO3 必须加上拉。

2.3 4bit ECC、8bit ECC 和 8bit/1k ECC、16bit/1k ECC

由于历史原因，用户可能会对 4bit ECC 和 8bit/1K ECC，8bit ECC 和 16bit/1k ECC 产生混淆。器件说的 4bit ECC 和 8bit ECC 每个纠错单元是 512B，而 HiFMCv100 的 8bit 和 16bit 每个纠错单元是 1KB（1KB 指 1KB 数量级，并不是严格意义上的 1024Byte；512B 指 512 数量级，并不是严格的 512Byte，一般是 526Byte）。所以，其实 4bit/512B 等价于 8bit/1K，8bit/512B 等价于 16bit/1K。

建议我们的开发和支撑人员在跟客户交流 ECC 的时候，把纠错单元的大小也指明，免得造成误解。

2.4 使用大容量 NAND 应该注意的地方

大容量 NAND 是指容量>4G 的 NAND 器件。4G 是一个 32 位无符号整型所能表示的最大极限，超过这个界限，32 位变量会发生回绕。当前测试过的最大容量为 8G。使用>4G 的器件时，应该注意文件系统（UBI/YAFFS2）分区大小不能超过 4G，否则文件系统可能回绕。

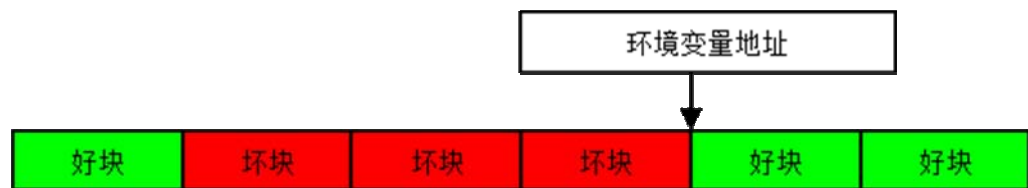


2.5 为什么在 NAND 上, u-boot 保存环境变量后, 系统无法启动.

有些用户在 NAND 上烧写完 u-boot 后, 系统能正常启动, 但保存环境变量后, 系统无法启动。

原因如图 2-4 所示, u-boot 占三个好块, NAND 起始位置有三个坏块, 保存环境变量后, fastboot 内容被擦除, 系统无法启动。这个时候, 要把环境变量地址往后挪。

图2-4 坏块分布示意图



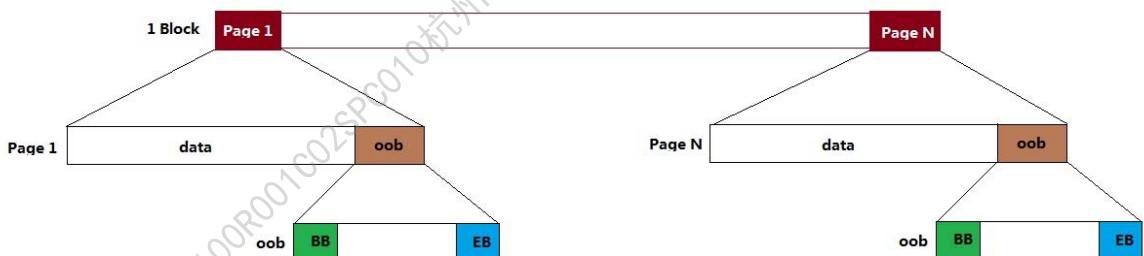
2.6 如何正确使用 mtd-utils 的 nandwrite 裸写工具

使用 mtd-utils 的 nandwrite 裸写工具时, 如果写的是 u-boot.bin 镜像, 且镜像大小大于 Nand Flash 的一个块的大小, 一定要保证 u-boot.bin 镜像的数据按块对齐填充。否则, 写进去的 u-boot.bin 镜像无法正常启动。

具体原因如下:

由于个别 Nand Flash 出厂时坏块标记位 (BB, Bad Block) 被标记为非全 0 的数, 例如 0xFE, 在判断坏块的时候, 容易被 FMC 控制器利用 ECC 纠错为 0xFF(因为全 0xFF 在控制器的 ECC 算法上是合法可纠错的), 故在每一个 page 的 OOB 信息的最后两个 byte 设置空块标记 (EB, Empty Block) 位。如图 2-5 所示, 在 u-boot 启动时, 逻辑上把 block 视为好块的前提条件是: block 的第一个 page 1 和最后一个 page N 的 BB=0xFF, EB=0x00。

图2-5 Nand Flash 块结构图



nandwrite 是根据镜像文件大小逐页写数据的, 并且写 page 时, 会自动将当页的 EB 位配置为 0x00。当写的最后一个 page 不是所在 block 的最后一个 page 时, 由于所在



block 最后一个 page 的 EB=0xFF，逻辑就是视这个 block 为空块不会去读数据，导致 uboot 启动失败。

值得注意的是，由于 Nand Flash 器件出厂时保证第一个块为好块，故逻辑不会去判断第一个块的情况，所以当 u-boot.bin 镜像大小小于一个块大小时，uboot 是可以正常启动的。

此外，一旦 uboot 正常启动，软件上我们不会再去判断 EB 位，这也是为什么使用 nandwrite 写内核镜像和文件系统镜像时可以不考虑镜像大小是否块对齐。

2.7 有些 Flash 器件 ID 不变工艺更新导致参数变化对兼容性的影响

伴随着 Flash（SPI Nor/SPI Nand/并口 Nand）工艺的不断更新，Flash 器件的接口、OOB 和性能等参数在不断的改变和优化。

某些厂家为了图方便，虽然 Flash 制作工艺升级了，但是 Flash ID 没有变化。而驱动是通过 ID 去识别器件的，不会通过厂家建议的 SFDP 寄存器去识别器件的批次、工艺等信息，因为 SFDP 不是标准的，各个厂家存在差异，即使同一个厂家，新老器件也会有差异。

因此，底下列出的 Flash 器件都是 ID 一致，参数不一致导致影响兼容性的几种情况。假如客户用到以下的（或者有相同情况）器件，可以参考下面的例子进行修改驱动，从而保证器件有正常的功能和优越的性能。

- SPI Nor Flash 的 ID 不变，接口类型变化

MXIC MX25L6436F 跟 MXIC MX25L6406E 比较，ID 一致，但是 MX25L6436F 接口类型增加了 2x I/O Read Mode/4x I/O Read Mode/4x I/O Page Program 的支持。

对于 MXIC MX25L6406E，hifmc_spi_nor_ids.c（hifmcv100）或 hisfc350_spi_ids.c（hisfc350）文件中的器件信息结构体 hifmc_spi_nor_info_table（hifmcv100）或 spi_info_table（hisfc350）应该定义为：



图2-6 MXIC MX25L6406E ID 注册表

```
{
    "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,
},
```

对于 MXIC MX25L6436F, hifmc_spi_nor_ids.c (hifmcv100) 或 hisfc350_spi_ids.c (hisfc350) 文件中的器件信息结构体 hifmc_spi_nor_info_table (hifmcv100) 或 spi_info_table (hisfc350) 应该定义为:

图2-7 MXIC MX25L6436F 注册表

```
{
    "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,
},
```

- SPI Nor Flash 的 ID 不变, 命令字变化
Micron SPI Nor 器件 MT25Q 系列和 N25Q 系列 EXTENDED QUAD INPUT FAST PROGRAM 命令不一样, 如图 2-8 所示。



图2-8 MT25Q 系列和 N25Q 系列命令差异

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

对于 Micron 的器件，如果用更高性能的 WRITE_QUAD_ADDR 接口类型，对于 MT25Q 系列，可以直接通过配置 WRITE_QUAD_ADDR 使用，因为现在驱动默认是匹配 38h 的命令。