XR871 Flash Support Developer Guide

Revision 1.0

December 1, 2017

Declaration

THIS DOCUMENTATION IS THE ORIGINAL WORK AND COPYRIGHTED PROPERTY OF XRADIO TECHNOLOGY ("XRADIO"). REPRODUCTION IN WHOLE OR IN PART MUST OBTAIN THE WRITTEN APPROVAL OF XRADIO AND GIVE CLEAR ACKNOWLEDGEMENT TO THE COPYRIGHT OWNER.

THE INFORMATION FURNISHED BY XRADIO IS BELIEVED TO BE ACCURATE AND RELIABLE. XRADIO RESERVES THE RIGHT TO MAKE CHANGES IN CIRCUIT DESIGN AND/OR SPECIFICATIONS AT ANY TIME WITHOUT NOTICE. XRADIO DOES NOT ASSUME ANY RESPONSIBILITY AND LIABILITY FOR ITS USE. NOR FOR ANY INFRINGEMENTS OF PATENTS OR OTHER RIGHTS OF THE THIRD PARTIES WHICH MAY RESULT FROM ITS USE. NO LICENSE IS GRANTED BY IMPLICATION OR OTHERWISE UNDER ANY PATENT OR PATENT RIGHTS OF XRADIO. THIS DATASHEET NEITHER STATES NOR IMPLIES WARRANTY OF ANY KIND, INCLUDING FITNESS FOR ANY PARTICULAR APPLICATION.

THIRD PARTY LICENCES MAY BE REQUIRED TO IMPLEMENT THE SOLUTION/PRODUCT. CUSTOMERS SHALL BE SOLELY RESPONSIBLE TO OBTAIN ALL APPROPRIATELY REQUIRED THIRD PARTY LICENCES. XRADIO SHALL NOT BE LIABLE FOR ANY LICENCE FEE OR ROYALTY DUE IN RESPECT OF ANY REQUIRED THIRD PARTY LICENCE. XRADIO SHALL HAVE NO WARRANTY, INDEMNITY OR OTHER OBLIGATIONS WITH RESPECT TO MATTERS COVERED UNDER ANY REQUIRED THIRD PARTY LICENCE.

Revision History

Version	Data	Summary of Changes
1.0	2017-12-1	Initial Version

Table 1-1 Revision History

Contents

Tables

Figures

1 Introduction

1.1 前言

市面上的 SPI Nor Flash 芯片（后文简述为 Flash 芯片）种类繁多。不同厂家不同型号的 flash 芯片在性能方面、命令方面、配置流程方面都会存在差异，而这种差异是通用的 flash 驱动无法很好地支持的。一般在嵌入式 sdk 中 flash 驱动只支持简单操作以及运行在正常的模式如 fast read，针对不同的 flash 需要使用者修改驱动，因此无法做到真正兼容和真正的支持。考虑到上述原因，XR871 sdk 的 flash 驱动采用通信驱动、芯片命令、Flash 控制分离的框架，实现便于扩展，具备高通用性，易于使用的 flash 驱动，以方便使用者进行方案开发，减少开发者工作量。

图注：

Write Status Register	01H	(S7-S0)	(S15-S8)

Figure 1-1 PN25F08 的写寄存器命令

Write Status Register-1	01H	S7-S0	
Write Status Register-2	31H	S15-S8	
Write Status Register-3	11H	S23-S16	

Figure 1-2 GD25Q64C 的写寄存器命令

1.2 Flash Datasheet 阅读指引

阅读 flash data sheet 文档时，建议注意包括以下关注点但不限于，

1. Flash 的存储空间大小。
2. Flash 支持的命令类型及格式。
3. Flash 的 Status 寄存器。
4. Flash 配置 WP、HOLD 与 PIN2、PIN3 切换方式。
5. Flash 进入 XIP 的方式。
6. Flash 的读命令（几种模式下）的 dummy 数以及与频率之间的关系（某些芯片存在关系）。
7. Flash 的最高工作频率以及 READ 命令的最高工作频率。
8. Flash 运行高频率的配置（有些芯片运行高频率需要执行一系列操作）。
9. 其他。

1.3 Flash 驱动框架

XR871 sdk 的 flash 驱动采用通信驱动（FlashDriverBase）、芯片驱动（FlashChipBase）、Flash 控制（Flash）分离的框架。

通信驱动（FlashDriverBase）作为抽象接口供其他两个模块使用。驱动通过继承并实现 FlashDriverBase 可扩展命令传输的驱动，当前已实现基于 SPI 的 SpiDriver 和基于 flash controller 的 FlashcDriver。

芯片驱动（FlashChipBase）作为芯片抽象，提供命令接口给 Flash 控制模块使用，提供如写使能、擦除等接口。扩展具体芯片需继承 FlashChipBase，根据芯片 data sheet 的内容实现对应接口以及参数。对于一些比较简单、命令与 default 一致的芯片，可通过第三章的简单配置实现扩展。

Flash 控制（Flash）模块是基于上述两个模块，进行逻辑控制，提供简便的 flash 操作接口。

图注：

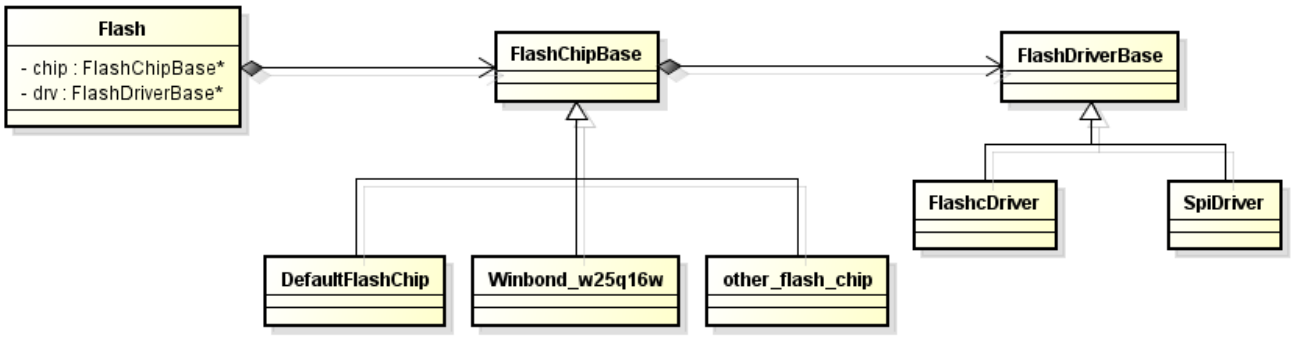

Figure 1-3 Flash 驱动框架示意图

1.4 Board_config Flash 配置

Flash 板级配置在 sdk\project\common\board\xr871_evb_xxx\board_config.c 里，以采用 flash controller 作为 flash 通信驱动为例，在 `g_pinmux_flashc[]`配置 flash controller 引脚后（采用 SWD 进行 Debug 则无法跑 4 线的模式），配置 `g_flash_cfg[]`：

Member	Description	
type	Flash 通信驱动类型，有 FLASH_DRV_FLASHC 和 FLASH_DRV_SPI	
mode	Flash 读模式，具体可参考 Table 3-1 和 3.2 小节	
flashc 或 spi（选其一）	flashc：flash controller 驱动配置	clk：flash 芯片工作频率
	spi：spi 驱动配置	clk：flash 芯片工作频率
		port：spi 通道号
		cs：spi 片选脚

Table 1-1 FlashBoardCfg 结构体成员


```
static const GPIO_PinMuxParam g_pinmux_flashc[] = {
    {
        GPIO_PORT_B,
        GPIO_PIN_4,
        { GPIOB_P4_F5_FLASH_MOSI, GPIO_DRIVING_LEVEL_3, GPIO_PULL_NONE }
    },
    {
        GPIO_PORT_B,
        GPIO_PIN_5,
        { GPIOB_P5_F5_FLASH_MISO, GPIO_DRIVING_LEVEL_3, GPIO_PULL_NONE }
    },
    {
        GPIO_PORT_B,
        GPIO_PIN_7,
        { GPIOB_P7_F5_FLASH_CLK,  GPIO_DRIVING_LEVEL_3, GPIO_PULL_NONE }
    },
    {
        GPIO_PORT_B,
        GPIO_PIN_6,
        { GPIOB_P6_F5_FLASH_CS,   GPIO_DRIVING_LEVEL_3, GPIO_PULL_UP   }
    },

#if (!BOARD_SWD_EN)
    {
        GPIO_PORT_B,
        GPIO_PIN_2,
        { GPIOB_P2_F5_FLASH_WP,   GPIO_DRIVING_LEVEL_3, GPIO_PULL_UP   }
    },
    {
        GPIO_PORT_B,
        GPIO_PIN_3,
        { GPIOB_P3_F5_FLASH_HOLD, GPIO_DRIVING_LEVEL_3, GPIO_PULL_UP   }
    },
#endif

};

/* flash */
static const FlashBoardCfg g_flash_cfg[] = {
    {
       .type = FLASH_DRV_FLASHC,
       .mode = FLASH_READ_DUAL_O_MODE,
       .flashc.clk = (48 * 1000 * 1000),
    },
};
```


2 Flash Normal Support

2.1 Flash 芯片驱动

Flash 芯片驱动在 driver/chip/flashchip/下。FlashChipBase 表示该芯片支持的参数、命令和操作等。扩展新的 flash chip 需要写一个继承 FlashChipBase 的结构体，并重载 FlashChipBase 结构体内的接口，或复用 default flash chip 接口。通过重载可以很简单的实现一个完全不同的指令或者行为，除了重载函数，几乎不需要加其他代码。

表注：

	Member	Description
型号信息	**mJedec**	flash 的 jedec ID，具体参考第三章
	mMaxFreq	除了 READ 命令以外，其他命令（含 FAST READ 等）允许的最高频率
	mMaxReadFreq	READ 命令的最高频率
	mSize	芯片存储容量
	mEraseSizeSupport	flash 芯片支持哪些擦除命令，具体参考第三章
	mReadStatusSupport	flash 芯片支持哪些读状态寄存器，具体参考第三章
	mWriteStatusSupport	flash 芯片支持哪些写状态寄存器，具体参考第三章
	mPageProgramSupport	flash 芯片支持哪些烧写命令，具体参考第三章
	mReadSupport	flash 芯片支持哪些读命令，具体参考第三章
	mFlashStatus	存储该芯片的状态，初始为 0
相关驱动	**mDriver**	通信驱动
	mXip	XIP 驱动
驱动对接接口	**driverWrite**	Chip 与 Driver 对接的驱动接口
	driverRead	Chip 与 Driver 对接的驱动接口
	xipDriverCfg	配 XIP 读命令格式，用于跑 XIP。
芯片接口	**writeEnable**	发送写使能命令
	writeDisable	发送写禁止命令
	readStatus	读状态寄存器
	writeStatus	写状态寄存器
	suspendErasePageprogram	暂停擦除/写操作
	resumeErasePageprogram	恢复擦除/写操作

芯片接口	**powerDown**	进入掉电模式
	releasePowerDown	退出掉电模式
	jedecID	获取 JEDEC ID
	enableQPIMode	进去 QPI 模式
	disableQPIMode	退出 QPI 模式
	reset	复位
	uniqueID	获得 unique ID
	pageProgram	发送烧写命令（多种模式的写命令）
	read	发送读命令（多种模式的读命令）
	erase	发送擦除命令
	setFreq	设置频率
	switchReadMode	切换读模式（Read/Fast Read/.....）
	enableXIP	开启 XIP 功能
	disableXIP	关闭 XIP 功能
	isBusy	Flash 是否在擦除/烧写中
	control	Ioctl 函数，用于扩展。
	minEraseSize	返回最小的擦除大小

Table 2-1 FlashChipBase 结构体成员

FlashChipCtor 用于实例化并初始化指定 Flash Chip。采用普通方式来扩展 flash chip list，需要实现该 FlashChipCtor。

表注：

Member	Description
mJedecId	该 flash 芯片的 JEDEC ID
create	创建该 Flash 芯片的实体
init	初始化 Flash 芯片的实体，替换接口具体实现
destory	删除 Flash 芯片的实体

Table 2-2 FlashChipCtor 结构体成员

Flash Chip Creator 会放进 flashChipList[]里面。注意：DefaultFlashChip 需要放在列表的最后，以保证匹配不到 flash 芯片构造器时，可以采用默认方式进行使用。

```
FlashChipCtor *flashChipList[] = {
        &MX25QH16B_FlashChip,
        &MX25QH32B_FlashChip,
```



```
        &DefaultFlashChip, /*default chip must be at the last*/
};
```

2.2 Flash 支持示例

这里以 MX25QH16B/MX25QH32B 为例，通过 data sheet 了解到这款芯片的行为与我们的 default 接口基本一致，但是我们想提供一个配置 flash IO 驱动能力的功能（该功能大部分芯片都不具备，以及配置的行为不能兼容其他具备该功能的芯片），该功能是通过配置 Status Register 3 的 DRV0,DRV1（5、6bit），于是我们需要采用普通扩展的方式重载 control 接口，以实现我们要的功能。

首先，我们需要要有一个 MX25QH16B/MX25QH32B 芯片的结构体，此处为了复用代码，取名 MX25QHXXB。

```
typedef struct MX25QHXXB_Flash
{
    FlashChipBase base;
} MX25QHXXB_Flash;
```

其次，我们需要实现 MX25QH16B/MX25QH32B 的 FlashChipCtor。在 MX25QHXXB_FlashCtor(uint32_t jedec)中通过判断实际创建的是哪个芯片，再具体配置特定参数（如 flash 的存储大小、支持的读写操作等），就可以实现代码复用。

```
FlashChipCtor MX25QH16B_FlashChip = {
        .mJedecId = 0x154020,
        .create = MX25QHXXB_FlashCtor,
        .init = MX25QHXXB_FlashInit,
        .destory = MX25QHXXB_FlashDeinit,
};

FlashChipCtor MX25QH32B_FlashChip = {
        .mJedecId = 0x164020,   /* QPI: 0x166020 */
        .create = MX25QHXXB_FlashCtor,
        .init = MX25QHXXB_FlashInit,
        .destory = MX25QHXXB_FlashDeinit,
};
```

```
static FlashChipBase *MX25QHXXB_FlashCtor(uint32_t jedec)
{
    uint32_t size;

    if (jedec == MX25QH16B_JEDEC)
    {
        size = 32 * 16 * 0x1000;
    }
    else if (jedec == MX25QH32B_JEDEC)
    {
        size = 64 * 16 * 0x1000;
    }
    else
        return NULL;
```



```
    MX25QHXXB_Flash *impl = malloc(sizeof(*impl));

    PCHECK(impl);
    HAL_Memset(impl, 0, sizeof(*impl));

    impl->base.mJedec = jedec;
    impl->base.mPageSize = 256;
    impl->base.mSize = size;
    impl->base.mMaxFreq = 104 * 1000 * 1000;
    impl->base.mMaxReadFreq = 80 * 1000 * 1000;
    impl->base.mEraseSizeSupport = FLASH_ERASE_64KB
                                 | FLASH_ERASE_32KB
                                 | FLASH_ERASE_4KB
                                 | FLASH_ERASE_CHIP;
    impl->base.mPageProgramSupport = FLASH_PAGEPROGRAM
                                   | FLASH_QUAD_PAGEPROGRAM;
    impl->base.mReadStausSupport = FLASH_STATUS1
                                 | FLASH_STATUS2
                                 | FLASH_STATUS3;
    impl->base.mWriteStatusSupport = FLASH_STATUS1
                                   | FLASH_STATUS2
                                   | FLASH_STATUS3;
    impl->base.mReadSupport = FLASH_READ_NORMAL_MODE
                            | FLASH_READ_FAST_MODE
                            | FLASH_READ_DUAL_O_MODE
                            | FLASH_READ_DUAL_IO_MODE
                            | FLASH_READ_QUAD_O_MODE
                            | FLASH_READ_QUAD_IO_MODE
                            | FLASH_READ_QPI_MODE;
    impl->base.mFlashStatus = 0;
    impl->base.mDummyCount = 1;

    return &impl->base;
}

static int PN25F64B_FlashDeinit(FlashChipBase * base)
{
    PCHECK(base);

    Flash_PN25F64B *impl = __containerof(base, Flash_PN25F64B, base);
    free(impl);

    return 0;
}
```

然后，我们通过重载 control 补充配置 DRV 的功能。通过编写 MX25QHXXB_Control(FlashChipBase *base, int op, void *param)替换原 defaultControl(FlashChipBase *base, int op, void *param)赋值到 FlashChipBase 结构体的 control 成员，以实现重载。

图注

Table 6.3 Status Register-3 (SR3)

Bits	Field	Function	Type	Default State	Description
7	HRSW(1)	HOLD# or RESET# function	Volatile	0	When HRSW=0, the pin acts as HOLD#; when HRSW=1, the pin acts as RESET#. HRSW functions are only available when QE=0.
6	DRV1(1)	Output Driver Strength		0	The DRV1 & DRV0 bits are used to determine the output driver strength for the Read operations.
5	DRV0(1)			0	
4	HFQ	High Frequency Enable Bit		0	0= QPI High Frequency Mode Disabled 1 = QPI High Frequency Mode Enabled
3	Latency Control (LC)(2)	Variable SPI Read Latency Control		0	Defines the number of read latency cycles in Fast Read, Dual Out, Quad Out, Dual IO, and Quad IO commands. Binary values for 1 to 15 latency cycles. A value of zero disables the variable latency mode.
2				0	
1				0	
0				0	

Figure 2-1 XM25QH16B Status Register 3

6.2.12 Output Driver Strength (DRV1, DRV0)

The DRV1 & DRV0 bits are used to determine the output driver strength for the Read operations.

DRV1, DRV0	Driver Strength
0, 0	50%
0, 1	25%
1, 0	75%(default)
1, 1	100%

Figure 2-2 XM25QH16B Status Register 3 DRV 描述

```
int MX25QHXXB_Control(FlashChipBase *base, int op, void *param)
{
    switch (op)
    {
        case DEFAULT_FLASH_SET_DRV:
            uint32_t drv = *(uint32 *)param;
            uint32_t tmp = 0;
            uint8_t status3;

            if (drv > 75)
                tmp = 3;
            else if (drv > 50)
                tmp = 2;
            else if (drv > 25)
                tmp = 0;
            else
                tmp = 1;

            base->readStatus(base, FLASH_STATUS3, &status3);
            FLASH_DEBUG("read status3: 0x%x", status3);
```



```
            status3 &= ~(3 << 5);
            status3 |= tmp << 5;
            base->writeStatus(base, FLASH_STATUS3, &status3);
            FLASH_DEBUG("write status3: 0x%x", status3);

            return 0;
    }

    return defaultControl(base, op, param);
}

static int MX25QHXXB_FlashInit(FlashChipBase * base)
{
    PCHECK(base);
    MX25QHXXB_Flash *impl = __containerof(base, MX25QHXXB_Flash, base);
    impl->base.control = MX25QHXXB_Control;
    return 0;
}
```

最后，在 flash_chip.c 的 flashChipList 里补充 MX25QH16B_FlashChip 和 MX25QH32B_FlashChip 就完成扩展。其他扩展例如指令不同、dummy 长度不同等兼容性问题也可以通过该重载方法解决。

```
FlashChipCtor *flashChipList[] = {
       &DefaultFlashChip, /*default chip must be at the last*/
       &MX25QH16B_FlashChip,
       &MX25QH32B_FlashChip,
};
```


3 Flash Simply support

3.1 基本要求

若 flash 芯片的命令与 default 实现一致，并且没有需要扩展的功能，则可通过简易配置进行扩展以稳定支持上该芯片，否则请参考第 2 章。其中 QPI 模式下是整个指令都是 4 线通信（包括 CMD 也是 4 线通信）。

表注：

Instruction	CMD(IO)	Address Bytes(IO)	Dummy Bytes(IO)	Data Bytes(IO)
Write Enable	0x06(1)	-	-	-
Write Disable	0x04(1)	-	-	-
Volatile SR Write Enable	0x50(1)	-	-	-
Read Status1	0x05(1)	-	-	1(1)
Read Status2	0x35(1)	-	-	1(1)
Read Status3	0x15(1)	-	-	1(1)
Write Status1	0x01(1)	-	-	1(1)
Write Status2	0x31(1)	-	-	1(1)
Write Status3	0x11(1)	-	-	1(1)
Read	0x03(1)	3(1)	-	n(1)
Fast Read	0x0B(1)	3(1)	1(1)	n(1)
Fast Read Dual Output	0x3B(1)	3(1)	1(1)	n(2)
Fast Read Dual IO	0xBB(1)	3(2)	1(2)(含 M0~M7)	n(2)
Fast Read Quad Output	0x6B(1)	3(1)	1(1)	n(4)
Fast Read Quad IO	0xEB(1)	3(4)	3(4)(含 M0~M7)	n(4)
Page Program	0x02(1)	3(1)	-	n(1)
Quad Page Program	0x32(1)	3(1)	-	n(4)
64KB Erase	0xD8(1)	3(1)	-	-
32KB Erase	0x52(1)	3(1)	-	-
4KB Erase	0x20(1)	3(1)	-	-
Chip Erase	0xC7(1)	-	-	-
JEDEC ID	0x9F(1)	-	-	3(1)
Enable Reset	0x66(1)	-	-	-
Reset Device	0x99(1)	-	-	-

Enter QPI Mode	0x38(1)	-	-	-
Set Read Parameters	0xC0(4)	-	-	-
Exit QPI Mode	0xFF(4)	-	-	1(4)

Table 3-1 Default Flash Chip 命令

3.2 Flash 支持示例

简易配置通过扩展 simpleFlashChip 数组实现。在数组里增加 SimpleFlashChipCfg 结构体并配置，以支持需要扩展的 flash 芯片参数。

1. **mJedec** 是 flash 的 jedec ID，24bit 长度：0xX0X1X2X3X4X5，X0X1 是 ID7~ID0，X2X3 是 ID15~ID8，X4X5 是 M7～M0.
2. **mSize** 是芯片存储容量。
3. **mEraseSizeSupport** 是 flash 芯片支持哪些擦除命令，一般有 64KByte 擦除（FLASH_ERASE_64KB）、32KByte 擦除（FLASH_ERASE_32KB）、4KByte 擦除（FLASH_ERASE_4KB）和全片擦除（FLASH_ERASE_CHIP）。
4. **mReadStausSupport** 是 flash 芯片支持哪些读状态寄存器，FLASH_STATUS1，FLASH_STATUS2，FLASH_STATUS3，
5. **mWriteStatusSupport** 是 flash 芯片支持哪些写状态寄存器，FLASH_STATUS1，FLASH_STATUS2，FLASH_STATUS3，
6. **mPageProgramSupport** 是 flash 芯片支持哪些烧写命令，FLASH_PAGEPROGRAM，FLASH_QUAD_PAGEPROGRAM
7. **mReadSupport** 是 flash 芯片支持哪些读命令，FLASH_READ_NORMAL_MODE，FLASH_READ_FAST_MODE，FLASH_READ_DUAL_O_MODE，FLASH_READ_DUAL_IO_MODE，FLASH_READ_QUAD_O_MODE，FLASH_READ_QUAD_IO_MODE，FLASH_READ_QPI_MODE，
8. **mMaxFreq** 是除了 READ 命令以外，其他命令（含 FAST READ 等）允许的最高频率。
9. **mMaxReadFreq** 是 READ 命令的最高频率。

```
typedef struct SimpleFlashChipCfg
{
    uint32_t mJedec;
    uint32_t mSize;

    uint32_t mEraseSizeSupport;
    uint16_t mReadStausSupport;
    uint8_t  mWriteStatusSupport;
    uint8_t  mPageProgramSupport;
    uint16_t mReadSupport;

    uint32_t mMaxFreq;
    uint32_t mMaxReadFreq;
} SimpleFlashChipCfg;
```


以 XTX 的 PN25F16B 为例，JEDEC ID 如图，因此 mJedec 填 0x15405E。该 flash 为 2MBytes 大小的 flash， mSize 填 32 * 16 * 0x1000（0x200000）。F_R 为 100MHz，f_R 为 55MHz，因此 mMaxFreq 填 100 * 1000 * 1000，mMaxReadFreq 填 55 * 1000 * 1000。以及从指令集进行擦除、读、写、读寄存器、写寄存器的支持配置。

图注：

Table 7.2(1) Manufacturer and Device Identification

OP Code	(M7-M0)	(ID15-ID0)	(ID7-ID0)
ABh	-	-	14h
90h	5E	-	14h
9Fh	5E	40h	15h

Figure 3-1 PN25F16B JEDEC ID

图注：

Table 7.1(1) Instruction Set

INSTRUCTION NAME	BYTE1 (CODE)	BYTE2	BYTE3	BYTE4	BYTE5	BYTE6	N-BYTES
Write Enable	06h						
write Disable	04h						
Read Status Register	05h	(S7-S0)(2)					
Write Status Register	01h	(S7-S0)(2)					
Read Data	03h	A23-A16	A15-A8	A7-A0	(D7-D0)	(Next byte)	continuous
Fast Read	0Bh	A23-A16	A15-A8	A7-A0	dummy	(D7-D0)	(Next byte) continuous
Fast Read Dual Output	3Bh	A23-A16	A15-A8	A7-A0	dummy	DIO= (D6,D4,D2,D0) DO= (D7,D5,D3,D1)	(One byte per 4 clocks, continuous)
Page Program	02h	A23-A16	A15-A8	A7-A0	(D7-D0)	(Next byte)	Up to 256 bytes
Block Erase(64KB)	D8h	A23-A16	A15-A8	A7-A0			
Half Block Erase(32KB)	52h	A23-A16	A15-A8	A7-A0			
Sector Erase(4KB)	20h	A23-A16	A15-A8	A7-A0			
Chip Erase	C7h/60h						
Power-down	B9h						
Release Power-down /Device ID	ABh	dummy	dummy	dummy	(ID7-ID0)(4)		
Manufacturer /Device ID(3)	90h	dummy	dummy	00h	(M7-M0)	(ID7-ID0)	
JEDEC ID	9Fh	(M7-M0) Manufacturer	(ID15-ID8) Memory Type	(ID7-ID0) Capacity			

Figure 3-2 PN25F16B 指令集

图注：

Table 8.6 AC Electrical Characteristics

SYMBOL	ALT	Parameter	SPEC			UNIT
			MIN	TYP	MAX	
F_R	f_C	Clock frequency For all instructions, except Read Data (03h) and Dual output(3bh) 2.7V-3.6V V_{CC} & Industrial Temperature	D.C.		100	MHz
f_R		Clock freq. Read Data instruction (03h)	D.C.		55	MHz

Figure 3-3 PN25F16B AC 参数

在 simpleFlashChip 里加入该 flash 型号的具体配置信息，并在 flash_default.h 中加入该宏的定义（若希望节省内存，则可以通过打开对应 flash 型号的宏，其他关闭即可）：

```
static const SimpleFlashChipCfg simpleFlashChip[] =
{
/* ...... */
#ifdef FLASH_PN25F16B
    {
        /* FLASH_PN25F16B */
        .mJedec = 0x15405E,
        .mSize = 32 * 16 * 0x1000,
        .mEraseSizeSupport = FLASH_ERASE_64KB
                           | FLASH_ERASE_32KB
                           | FLASH_ERASE_4KB
                           | FLASH_ERASE_CHIP,
        .mPageProgramSupport = FLASH_PAGEPROGRAM,
        .mReadStausSupport = FLASH_STATUS1,
        .mWriteStatusSupport = FLASH_STATUS1,
        .mReadSupport = FLASH_READ_NORMAL_MODE
                      | FLASH_READ_FAST_MODE
                      | FLASH_READ_DUAL_O_MODE,
        .mMaxFreq = 100 * 1000 * 1000,
        .mMaxReadFreq = 55 * 1000 * 1000,
    },
#endif
/* ...... */
}
```

flash_default.h 中加入该宏的定义：

```
#define FLASH_M25P64

#define FLASH_PN25F16B

#define FLASH_W25Q16FW

#define FLASH_PN25F08

#define FLASH_PN25F16
```


4 附录

Flash chip 驱动：

sdk/src/driver/chip/flashchip/

sdk/include/driver/chip/flashchip/

Flash 驱动：

sdk/src/driver/chip/hal_flash.c

sdk/include/driver/chip/hal_flash.h

Flash controller 驱动：

sdk/src/driver/chip/hal_flashctrl.c

sdk/include/driver/chip/hal_flashctrl.h

Spi 驱动：

sdk/src/driver/chip/hal_spi.c

sdk/include/driver/chip/hal_spi.h

