应用笔记

Application Note

文档编号：**AN1108**

APM32F411 系列 QSPI 应用笔记

版本：**V1.0**

1 引言

本应用笔记提供如何在 APM32F4xx 系列上配置和应用 QSPI 接口的指南，包括代码实现以及应用实例。

QSPI 支持三种帧格式以及四种传输模式，使其能够连接单，双线或者四线外部 SPI Flash 存储介质。

目前，APM32F411 的 QSPI 模块仅用于与 QSPI Flash 进行通信，并不支持 XIP 功能。

目录

1 引言 1

2 QSPI 简介 3

2.1 QSPI 模式 3

2.2 QSPI 帧格式组成 5

2.3 QSPI 传送模式 8

2.4 QSPI FIFO 使用 8

2.5 QSPI 中断以及 DMA 9

3 QSPI 配置 10

3.1 QSPI 初始化 10

3.2 标准 SPI 模式 12

3.3 Dual SPI 模式 13

3.4 Quad SPI 模式 16

3.5 QPI 模式 17

4 Quad SPI 读取 FLASH ID 18

4.1 轮询读取 18

4.2 中断读取 19

5 版本历史 21

2 QSPI 简介

QSPI（Quad Serial Peripheral Interface）是一个串行数据总线接口，由时钟（SCLK），片选信号（CS），和 4 条数据线（IO[0:3]）组成。它是 SPI 的增强版本，常用于与 QSPI Flash 存储设备通信。

2.1 QSPI 模式

为了在不同的环境下满足设计需求，QSPI 支持三种不同的传输模式，分别为标准 SPI（单线）模式，Dual SPI（双线）模式以及 Quad SPI（四线）模式。三种不同模式的帧格式使用的 IO 引脚也有差别，具体如下：

- **标准 SPI 模式**：时钟（SCLK）,片选信号（CS）,MOSI（IO0）,MISO（IO1）。
- **Dual SPI 模式**：时钟（SCLK）,片选信号（CS），2 条数据线（IO[0:1]）。
- **Quad SPI 模式**：时钟（SCLK），片选信号（CS），4 条数据线（IO[0:3]）。

2.1.1 标准 SPI 模式

QSPI 的标准 SPI 模式支持全双工传输，但受连接存储器件限制，通常为半双工传输。在标准模式下，数据在 IO0 和 IO1 线上进行传输，IO0 线为 MOSI,而 IO1 线为 MISO。而 IO2 和 IO3 连接到 QSPI Flash 的 nWP 以及 nHOLD 上，并输出高电平来取消写保护功能以及激活保持功能。具体硬件连线图如图 1 所示：

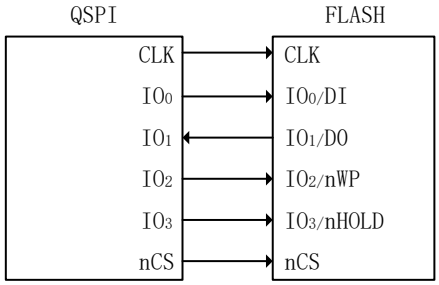

图 1 标准 SPI 模式硬件连线图

通过上图可知，在标准 SPI 模式下，QSPI 通过 IO0 给 Flash 传输数据，通过 IO1 接收 Flash 的数据。如果 nWP 以及 nHOLD 通过 QSPI Flash 硬件进行连接时，QSPI 的 IO3 和 IO4 可当其他 IO 使用。

应用标准 SPI 模式需将控制寄存器 1（QSPI_CTRL1）的帧格式（FRF）配置成 0x0。

2.1.2 Dual SPI 模式

在 Dual SPI 模式下，硬件连线与标准 SPI 模式类似，但是 Dual SPI 模式使用两根数据线同时发送和接收。QSPI Flash 也将 DI、DO 两个单向引脚变为双向引脚 IO0 和 IO1。具体硬件连线

如图 2 所示。

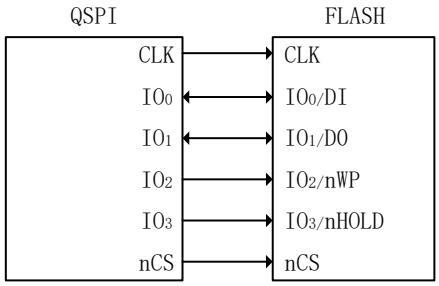

图 2 Dual SPI 模式硬件连线图

通过上图可知，IO0 和 IO1 共同完成数据的传输，缩短了传输数据的时间。与标准 SPI 模式相比，大大提高了数据的吞吐量。

应用 Dual SPI 模式需将控制寄存器 1（QSPI_CTRL1）的帧格式（FRF）配置成 0x1。

2.1.3 Quad SPI 模式

在 Quad 模式下，QSPI 使用四根数据线同时发送或者接收数据。在此模式下，QSPI Flash 将使能四路模式，nWP 以及 nHOLD 引脚复用成 IO 口，作数据传输，无写保护和保持功能。具体硬件连线如图 3 所示。

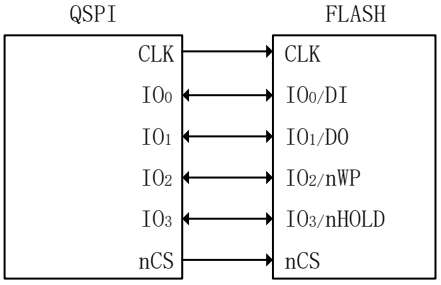

图 3 Quad SPI 模式硬件连线图

对比 Quad SPI 模式的硬件连接图和 Dual SPI 模式的硬件连接图可知，数据从 IO0 和 IO1 两个 IO 口进行传输变成从 IO0~3 四个 IO 口进行传输数据，吞吐量得到明显提升。

应用 Quad SPI 模式需将控制寄存器 1（QSPI_CTRL1）的帧格式（FRF）配置成 0x2。

2.2 QSPI 帧格式组成

在 QSPI 与存储设备进行信息传输过程中，每一次操作的数据流称为帧。QSPI 的帧通常由四个阶段组成，分别指令段，地址段，空闲字节段以及数据段。QSPI 的帧的详细情况如图 4 所示。

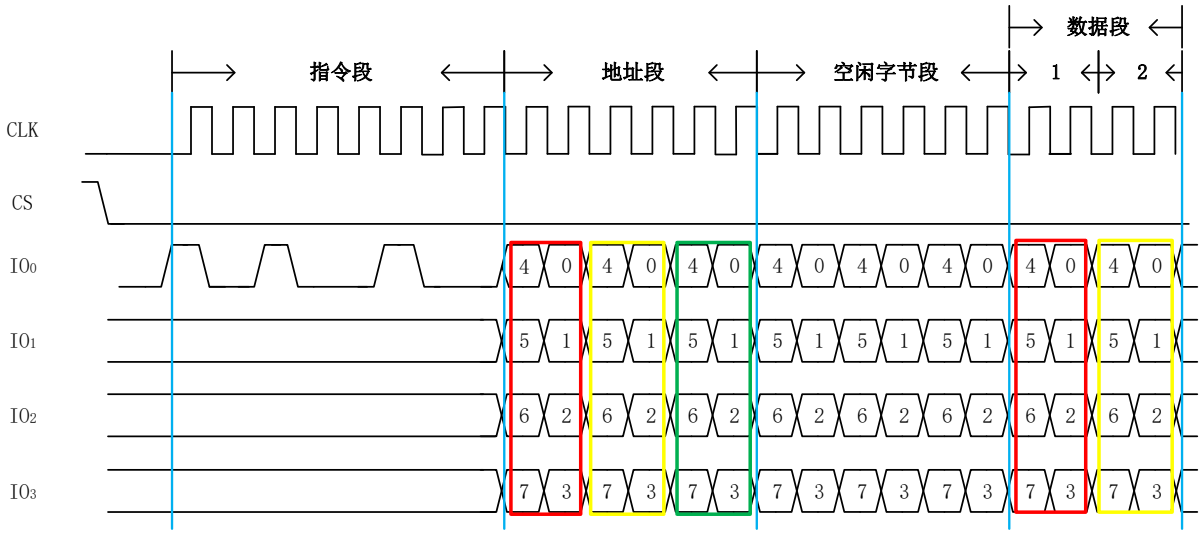

图 4 QPSI 四路读取序列

上图为 QSPI 四路读取 QSPI Flash ID 的序列逻辑图。其中指令段为单线模式，地址段以及数据段为四路模式。每一段的传输模式由控制寄存器 1（QSPI_CTRL1）的 FRF 以及控制寄存器 3（QSPI_CTRL3）的 IAT 进行配置，具体见后续章节。

2.2.1 指令段

在指令段中，QSPI 会向 QSPI Flash 发送一条 8bit 的指令，具体指令可由用户根据软件设计以及硬件需求进行配置。QSPI 可以选择在单线模式下或者双线模式以及四线模式下将指令发送出去，具体配置见下文。

1. 在单线模式下发送指令，需要将控制寄存器 3（QSPI_CTRL3）的 IAT 位配置成 0x00 或者 0x01。单线模式下的指令发送序列如图 5 所示。

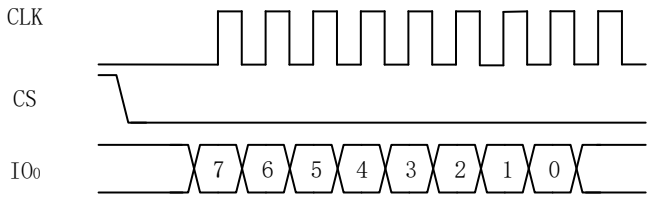

图 5 单线模式指令发送序列

2. 在双线模式（Dual SPI 模式）下发送指令，需要将控制寄存器 3（QSPI_CTRL3）的 IAT

位配置成 0x2，同时将控制寄存器 1（QSPI_CTRL1）的 FRF 位配置成 0x1。双线模式下的指令发送序列如图 6 所示。

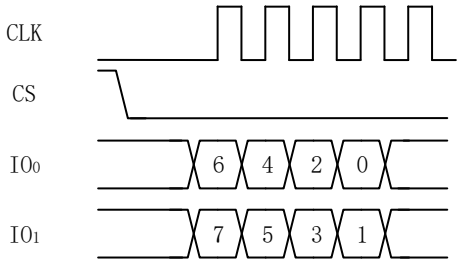

图 6 双线模式指令发送序列

3. 在四线模式（Quad SPI 模式）下发送指令，需要将控制寄存器 3（QSPI_CTRL3）的 IAT 位配置成 0x2，同时将控制寄存器 1（QSPI_CTRL1）的 FRF 位配置成 0x2。四线模式下的指令发送序列如图 7 所示。

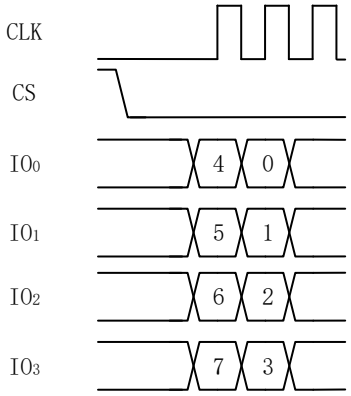

图 7 四线模式指令发送序列

除了指令的发送模式可以配置，指令的长度也可进行配置，具体配置见表格 1。

表格 1 指令长度配置选择

指令长度选择	寄存器相关配置
没有指令	CTRL3[9:8] INSLEN = 0X00
4 位指令	CTRL3[9:8] INSLEN = 0X01
8 位指令	CTRL3[9:8] INSLEN = 0X02
16 位指令	CTRL3[9:8] INSLEN = 0X03

注意：单线模式不需要对指令长度进行配置。

2.2.2 地址段

在地址段中，QSPI 会向 QSPI Flash 发送需要写入或者读取数据的地址。地址的长度以及发送的模式均可由用户进行配置，其中地址的发送模式配置与指令的类似。具体如下：

(1). 单线模式（标准 SPI 模式）：将控制寄存器 3（QSPI_CTRL3）的 IAT 配置成 0x00，即可在单线模式下发送地址。

(2). 双线模式（Dual SPI 模式）：将控制寄存器 3（QSPI_CTRL3）的 IAT 配置成 0x01 或者 0x02，控制寄存器 1（QSPI_CTRL1）的 FRF 配置成 0x1，即可在双线模式下发送地址。

(3). 四线模式（Quad SPI 模式）：将控制寄存器（QSPI_CTRL3）的 IAT 配置成 0x01 或者 0x02，控制寄存器 1（QSPI_CTRL1）的 FRF 配置成 0x2，即可在四线模式下发送地址。

而地址的长度由控制寄存器 3（QSPI_CTRL3）的 ADDRLEN 进行配置，可以将其配置成没有地址发送以及 4~60 位的地址长度（4 的倍数）。单线模式也不需要进行地址长度的配置，具体请查看 APM32F411xCxE 用户手册。

2.2.3 空闲字节段

空闲字节段主要在快速读取或者快速写入 QSPI Flash 使用。在快速操作时，QSPI Flash 有可能需要额外的时钟周期来准备数据发送。同样，QSPI 也需要延迟若干个时钟周期来进行数据读取，进而确保读取数据的正确性。

控制寄存器 3（QSPI_CTRL3）的 WAITCYC 位可对空闲字节占用的时钟周期进行配置，最多可以给空闲字节配置 31 个时钟周期。

2.2.4 数据段

数据段主要进行数据的写入以及读取。其传输模式以及传输数量也可以由用户进行配置。其中，传输模式配置在控制寄存器 1（QSPI_CTRL1）的 FRF 中进行配置，具体为：

- 0x00：单线模式（标准 SPI 模式）
- 0x01：双线模式（Dual SPI 模式）
- 0x02：四线模式（Quad SPI 模式）

在双线以及四线模式下，数据帧的数量由控制寄存器 2（QSPI_CTRL2）的 NDF 进行配置。通过配置该位来决定 QSPI 写入或者接收数据帧的个数。单线模式下也不需要对该寄存器进行配置。

2.3 QSPI 传送模式

APM32F411 的 QSPI 模块一共支持四种传输模式，分别为发送和接收、仅发送、仅接收以及 EEPROM（电可擦除只读存储器）读取。传输模式（TXMODE）由控制寄存器 1（QSPI_CTRL1）的 TXMODE 确定。四种传输模式具体区别如下：

(1). 发送和接收模式：仅适用于标准 SPI 模式下进行发送和接收。

(2). 仅发送模式：在标准 SPI 模式下只能发送，接收数据线上的数据会被忽略。或用于增强模式下（Dual SPI 模式以及 Quad SPI 模式）的写入操作。

(3). 仅接收模式：在标准 SPI 模式下只能接收。或用于增强模式下（Dual SPI 模式以及 Quad SPI 模式）的读取操作。

(4). EEPROM 读取模式：仅用于操作 EEPROM。

2.4 QSPI FIFO 使用

QSPI 集成用于发送和接收的 FIFO 缓冲器。该 FIFO 缓冲器深度为 8，长度为 32bit。因此，传输和接收的数据位数固定为 32 位。小于 32 位的数据帧在被写入到传输 FIFO 缓冲器时会进行右对齐。

往 QSPI 的 DATA 寄存器写入数据后，如果发送 FIFO 未满，则会将 DATA 的数据填充到发送 FIFO 中。从 DATA 寄存器读取数据时，如果接收 FIFO 有数据，则会把数据读取出来。

2.5 QSPI 中断以及 DMA

2.5.1 QSPI 中断

QSPI 一共支持 6 种中断。关于中断的使能、状态位查看以及清除标志位等详情如下表所示。

表格 2 QSPI 中断详情

中断类型	使能位	状态位	清除位	中断触发详情
传输中断	TFEIE	TFEIF	ICF	往发送 FIFO 填充数据会触发中断
发送 FIFO 溢出中断	TFOIE	TFOIF	TFOIC	往满载的发送 FIFO 填充数据会触发中断
接收 FIFO 下溢中断	RFUIE	RFUIF	RFUIC	读取空的接收 FIFO 会触发中断
接收 FIFO 溢出中断	RFOIE	RFOIF	RFOIC	当满载的接收 FIFO 接收存储器发送的数据时会触发中断
接收 FIFO 满中断	RFFIE	RFFIF	ICF	当接收 FIFO 的数据超过 RFT 设定的值会触发中断
多主机争用中断	MSTIE	MSTIF	MIC	当串行总线上出现争用会触发中断

2.5.2 QSPI DMA 使用

通过配置 DMA 控制寄存器（QSPI_DMACTRL）的 RDMANE 位和 TDMAEN 位来使能 QSPI 的 DMA 功能。

3 QSPI 配置

本章节将介绍使用 QSPI 模块时的基础配置，以及介绍 QSPI 三种不同模式的配置方法。其中一些涉及 QSPI Flash 的内容在本章节不做讲解，请读者自行查询。

3.1 QSPI 初始化

本小节将对 QSPI 初始化进行讲解，主要分为 GPIO 初始化，时钟配置以及 QSPI 初始化。具体请看后续内容。

3.1.1 GPIO 初始化

在 GPIO 初始化中，先打开使用到的引脚的相应的 GPIO 时钟，并配置其 IO 口复用功能。将 CS 引脚配置成输出模式，其他引脚配置成复用推挽模式。最后，将 CS 引脚上拉。具体代码配置如下：

```
GPIO_Config_T GPIO_ConfigStruct;
RCM_EnableAHB1PeriphClock((RCM_APB2_PERIPH_T)(RCM_AHB1_PERIPH_GPIOE | RCM_AHB1_PERIPH_GPIOB ));
GPIO_ConfigPinAF(GPIOE, GPIO_PIN_SOURCE_7, GPIO_AF10_QSPI); //QSPI_IO0
GPIO_ConfigPinAF(GPIOE, GPIO_PIN_SOURCE_8, GPIO_AF10_QSPI); //QSPI_IO1
GPIO_ConfigPinAF(GPIOE, GPIO_PIN_SOURCE_9, GPIO_AF10_QSPI); //QSPI_IO2
GPIO_ConfigPinAF(GPIOE, GPIO_PIN_SOURCE_10, GPIO_AF10_QSPI); //QSPI_IO2
GPIO_ConfigPinAF(GPIOB, GPIO_PIN_SOURCE_1, GPIO_AF_QSPI); //QSPI_CLK
GPIO_ConfigPinAF(GPIOB, GPIO_PIN_SOURCE_6, GPIO_AF10_QSPI); //QSPI_CS
GPIO_ConfigStruct.mode = GPIO_MODE_AF;
GPIO_ConfigStruct.speed = GPIO_SPEED_50MHz;
GPIO_ConfigStruct.otype = GPIO_OTYPE_PP;
GPIO_ConfigStruct.pupd = GPIO_PUPD_NOPULL;
GPIO_ConfigStruct.pin = GPIO_PIN_7 | GPIO_PIN_8 | GPIO_PIN_9 | GPIO_PIN_10;
GPIO_Config(GPIOE, &GPIO_ConfigStruct);
GPIO_ConfigStruct.pin = GPIO_PIN_1;
GPIO_Config(GPIOB, &GPIO_ConfigStruct);
GPIO_ConfigStruct.mode = GPIO_MODE_OUT;
GPIO_ConfigStruct.pin = GPIO_PIN_6;
GPIO_Config(GPIOB, &GPIO_ConfigStruct);

GPIO_SetBit(GPIOB, GPIO_PIN_6);
```

注意：APM32F411 的 QSPI 相关 IO 口作 BK1 和 BK2 区分，如 QSPI_BK1_IO1 和 QSPI_BK2_IO1。两者在配置复用功能时参数会有差异。BK1 的 IO 配置复用功能的参数为 AF10（GPIO_AF10_QSPI）,BK2 的 IO 配置复用功能的参数为 AF9（GPIO_AF_QSPI）。除了配置复用功能时参数不同，BK1 和 BK2 的 IO 在 QSPI 使用上没有区别。

3.1.2 QSPI 时钟分频

QSPI 时钟源由 AHB 提供，其中需要经过 QSPI_BR 寄存器分频，得到最终的 QSPI 时钟。详情如下图所示。

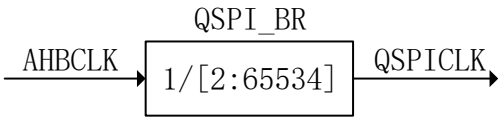

图 8 QSPI 时钟分频配置图

注意：QSPI_BR 的 CLKDIV 字段的 LSB 始终被设置为 0。因此，QSPI 的时钟分频值为 2 到 65534 之间的任意偶数，最高分频值为 65534。例如：AHB 的时钟为 100MHz，QSPI 的分频值设置为 65534，则 QSPI 的时钟 QSPICLK = 100/65534=1.5259KHz。

3.1.3 QSPI 初始化

在 QSPI 初始化中，需先打开 QSPI 的时钟，并将 IOSW 位置 1 开启 QSPI 对应的 GPIO 端口，然后在波特率寄存器（QSPI_BR）的 CLKDIV 配置 QSPI 的时钟分频，在控制寄存器 1（QSPI_CTRL1）中配置数据帧的大小（DFS）、时钟相位和极性（CPHA 和 CPOL）、片选的反转使能（SSTEN）以及帧格式的选择（FRF）。最后使能从机（SLAEN）从而使数据能够发送。需要注意在使能 QSPI（SSIEN）后，控制寄存器 1（QSPI_CTRL1）不能被写入。因此，每次切换 QSPI 的工作模式前，需先将 QSPI 失能，即将使能寄存器（QSPI_SSIEN）的 EN 位置 0。根据需要的工作模式更改完成控制寄存器 1 后再将 QSPI 使能。具体代码配置如下：

```
QSPI_Config_T configStruct;
RCM_EnableAHB2PeriphClock(RCM_AHB2_PERIPH_QSPI);
QSPI_GPIOInit();
QSPI_Reset();
QSPI_OpenIO();
configStruct.selectSlaveToggle = QSPI_SST_DISABLE;
configStruct.clockDiv = 0X64;
configStruct.clockPhase = QSPI_CLKPHA_1EDGE;
configStruct.clockPolarity = QSPI_CLKPOL_LOW;
configStruct.dataFrameSize = QSPI_DFS_8BIT;
configStruct.frameFormat = QSPI_FRF_STANDARD;
QSPI_Config(&configStruct);
QSPI_ConfigTxFifoThreshold(0);
QSPI_ConfigRxFifoThreshold(1);
QSPI_EnableClockStretch();

QSPI_EnableSlave();
```

通过以上两个步骤便可完成 QSPI 的初始化工作，后续将根据不同 QSPI 模式进行讲解。

需要注意的是，使用 QSPI 操作 Flash 除了要使能 QSPI 外，还需要进行从机（SLAEN）的使能。如果没有使能从机的话，数据将会在 FIFO 区中无法发送出去。

3.2 标准 SPI 模式

在进行标准 SPI 模式的 QSPI 通信前，需要先将 QSPI 配置成标准 SPI 模式，具体步骤如下：

1. 失能 QSPI;；
2. 修改 QSPI 的帧格式为标准 SPI 模式；
3. 根据自身需求修改 QSPI 的传输方向；
4. 使能 QSPI；

完成上述步骤后即可对 QSPI Flash 发送相应的命令或者数据进行操作 QSPI Flash。在标准 SPI 模式下，可以使用缓冲队列 FIFO 进行多发多收，也可以与 SPI 一样单次发送与接收。两种方式具体实现步骤如下：

a. 单次发送接收：在标准 SPI 模式下进行单次发送与接收时，QSPI 的处理流程与 SPI 的一致。在 QSPI 完成初始化后将从机使能打开，即将 SLAEN 位进行置 1，接着就往 FIFO 里面填充数据。每填充一次数据，QSPI 都会将其发送出去，然后等待 RFNEF 置 1（接收 FIFO 非空），最后将接收 FIFO 中的数据读取出来。具体流程见图 9。

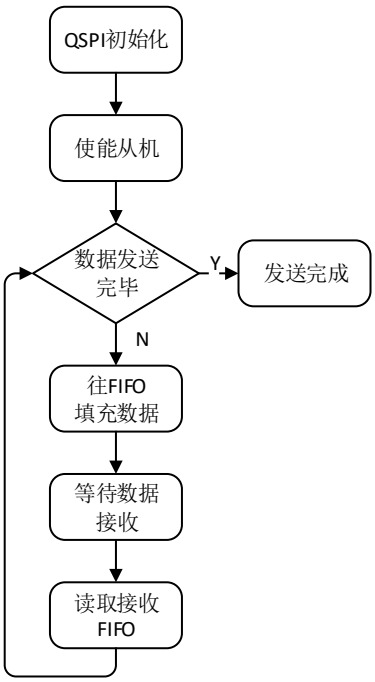

图 9 标准 SPI 模式单次传输流程图

b. 批量发送接收：在标准 SPI 模式下进行批量发送和接收，QSPI 的操作流程如图 10 所示。首先失能从机，即将 SLAEN 置 0，再将数据填充到发送 FIFO 中。QSPI 的 FIFO 大小为 8*32bit。因此，在往 FIFO 填充 8 次数据后，QSPI_STS 寄存器的 TFNF 会置零，表示发送 FIFO 已满。如果发送 FIFO 已满，则使能从机（SLAEN 置 1）将 FIFO 中的数据发送出去并等待数据的接收与读取。完成数据的读取后将从机失能，继续往发送 FIFO 填充数据。当所有数据都填充到发送 FIFO 中后，则使能从机，将剩余数据发送出去并等待数据的接收与读取。

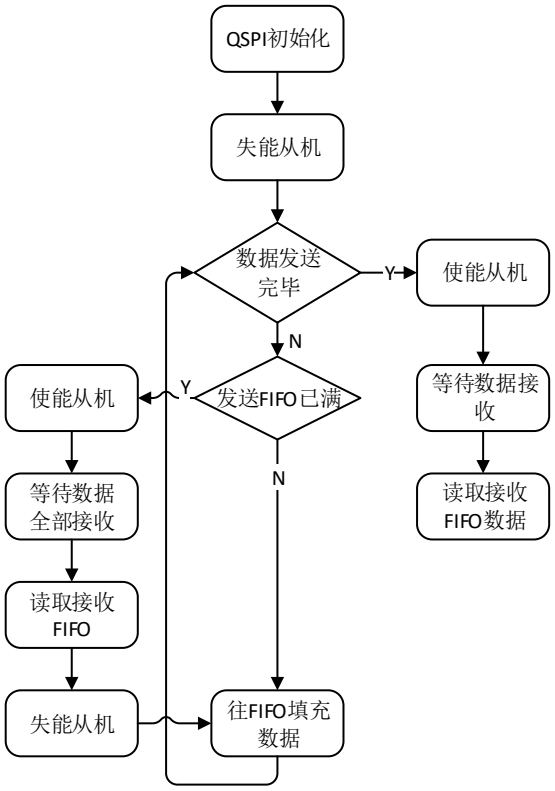

图 10 标准 SPI 模式批量传输流程图

3.3 Dual SPI 模式

在进行 Dual SPI（双线）模式通信前，需要先将 QSPI 配置成 Dual SPI 模式，同时还需将指令长度、地址长度等参数进行配置，具体配置步骤如下：

1. 失能 QSPI（QSPI_SSIEN 的 EN 位）；
2. 配置数据传输模式（QSPI_CTRL1 的 TXMODE 位）；
3. 修改 QSPI 的帧格式为 Dual SPI 模式（QSPI_CTRL1 的 FPF 位）；
4. 配置 QSPI 的指令长度（QSPI_CTRL3 的 INSLEN）；
5. 配置 QSPI 的地址长度（QSPI_CTRL3 的 ADDRLEN）；
6. 配置 QSPI 的等待周期（QSPI_CTRL3 的 WAITCYC）；

7. 配置 QSPI 的数据大小（QSPI_CTRL2 的 NDF）；

8. 配置 QSPI 的指令地址类型（QSPI_CTRL3 的 IAT）；

9. 使能 QSPI（QSPI_SSIEN 的 EN 位）；

其中，指令长度和地址长度决定 QSPI 发送指令和地址的时钟周期个数。如果这两项没有配置正确，可能导致 QSPI 不能发送完整指令或者指令不被 QSPI Flash 识别。等待周期决定空闲字节的时钟周期长度，此项保证 QSPI 读取数据的正确性。数据大小则是需要读取或者写入数据的数量。

需要注意的是，数据阶段的数据类型是跟随 QSPI 的模式，但是指令阶段以及地址阶段的数据类型是根据 CTRL3 的 IAT 决定的。一共有三个选择，分别是（1）指令以及地址为单线模式；（2）指令为单线模式，地址跟随 QSPI 模式；（3）指令以及地址都跟随 QSPI 模式。这一项的选择主要是根据 QSPI 存储设备是否支持相关的传输模式。通常来说，指令都是单线模式，地址则跟随 QSPI 模式。

完成上述配置后，就可以将需要写入的指令以及地址填充到发送 FIFO 中。如果是读取指令则接着进行数据读取，如果是写入指令，则进行数据的写入。下面代码以读取数据为例：

```
void QSPI_Dual_Read(QSPI_ReadWriteParam_T *rParam)
{
    uint16_t i;
    QSPI_CS_LOW;
    QSPI_Disable();
    QSPI_ConfigTansMode(QSPI_TRANS_MODE_RX);
    QSPI_ConfigAddrLen((QSPI_ADDR_LEN_T)rParam->addrLen);
    QSPI_ConfigInstLen((QSPI_INST_LEN_T)rParam->instLen);
    QSPI_ConfigWaitCycle(rParam->waitCycle);
    QSPI_ConfigInstAddrType(rParam->instAddrType);
    QSPI_ConfigFrameNum(rParam->dataLen - 1);
    QSPI_ConfigFrameFormat(QSPI_FRF_DUAL);
    QSPI_Enable();
    if(rParam->instLen)
    {
        QSPI_TxData(rParam->instruction);
    }
    if(rParam->addrLen)
    {
        QSPI_TxData(rParam->addr);
    }
    for(i = 0; i < rParam->dataLen; i++)
    {
        while(QSPI_ReadStatusFlag(QSPI_FLAG_RFNE) == RESET);
        rParam->dataBuf[i] = QSPI_RxData();
    }
    while(QSPI_ReadStatusFlag(QSPI_FLAG_BUSY) == SET);
    QSPI_CS_HIGH;
}
```

通过上面的 Dual SPI 模式的读取操作代码可以知道，指令阶段，地址阶段以及数据阶段都是可以省略的，具体情况需要视 QSPI Flash 是否支持以及所选指令而定。

Dual SPI 模式下读取数据的传输逻辑图如图 11 所示，读取的数据为 QSPI Flash 的 ID 信息，指令长度和地址长度分别为 8bit 以及 24bit，数据个数为 4，数据帧大小为 8bit。

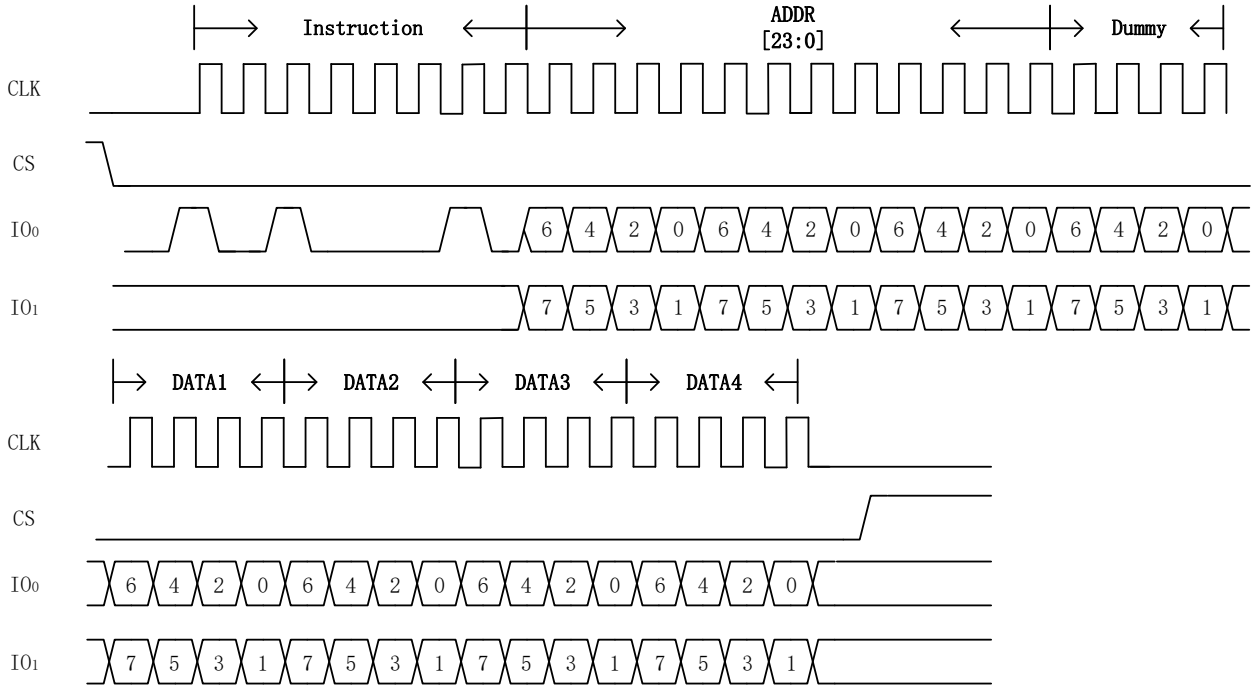

图 11 Dual SPI 模式传输逻辑图

通过上图可以看出，在 Dual SPI 模式下，数据在 IO0 和 IO1 线上传输。因此 8bit 数据传输需要 4 个时钟周期，其中 IO1 为 bit7，bit5，bit3 以及 bit1，而 IO0 为 bit6，bit4，bit2 以及 bit0。同时，在发送地址后，没有马上进行数据读取，而是隔了四个时钟周期后才进行数据读取。这四个时钟周期即为空闲字节。在快速操作中，读取数据前需要间隔空闲字节来保证数据的正确性。

3.4 Quad SPI 模式

Quad SPI 模式的配置和 Dual SPI 模式大致相同，需要注意修改的地方是 QSPI 模式的配置，以及等待周期个数的修改。Quad SPI 模式和 Dual SPI 模式的需要的等待周期是不一样的。具体配置步骤如下：

1. 失能 QSPI（QSPI_SSIEN 的 EN 位）；
2. 配置数据传输模式（QSPI_CTRL1 的 TXMODE 位）；
3. 修改 QSPI 的帧格式为 Quad SPI 模式（QSPI_CTRL1 的 FPF 位）；
4. 配置 QSPI 的指令长度（QSPI_CTRL3 的 INSLEN）；
5. 配置 QSPI 的地址长度（QSPI_CTRL3 的 ADDRLEN）；
6. 配置 QSPI 的等待周期（QSPI_CTRL3 的 WAITCYC）；
7. 配置 QSPI 的数据个数（QSPI_CTRL2 的 NDF）；
8. 配置 QSPI 的指令地址类型（QSPI_CTRL3 的 IAT）；

9. 使能 QSPI（QSPI_SSIEN 的 EN 位）；

完成上述配置后，就可以往 FIFO 写入相应的指令和地址了。这里配置的指令长度为 8，地址长度为 24，6 个空闲周期以及读取 2 个 8 位的数据。Quad SPI 模式的传输逻辑图如图 12 所示。

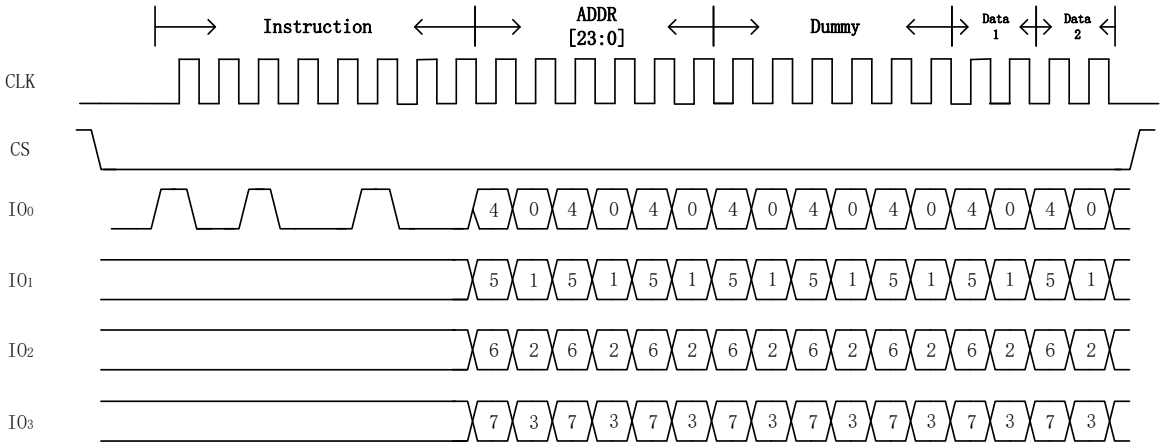

图 12 Quad SPI 模式传输逻辑图

由上图可知，Quad SPI 模式下，数据由四根数据线并行输出。传输速度跟 Dual SPI 模式相比有明显提升。

3.5 QPI 模式

QPI 模式为 Quad SPI 模式下的一个特殊模式。它与 Quad SPI 模式的区别为指令阶段是四线模式传输的。因此，QPI 模式的配置是在 Quad SPI 模式的配置的基础上，将 QSPI_CTRL3 的 IAT 修改成 0x10，即选择在 SPI_FRF 规定的模式下发送指令和地址，即四线模式下发送指令和地址。

4 Quad SPI 读取 FLASH ID

本章节将介绍如何设计代码读取 QSPI Flash W25Q64FV 的 ID 信息。其中，QSPI Flash 默认配置成四线使能模式。对于如何将 QSPI Flash 配置四线使能模式，本章不做讲解。

4.1 轮询读取

在设计轮询读取 Flash ID 代码前，需了解读取该 Flash ID 的步骤。四线模式下读取 QSPI Flash 的 ID 的指令为 0x94，地址为 24bit 的 0x000000，需要等待的空闲字节周期为 6 个 byte。最终读取的 ID 为 16bit 数据。

根据上述读取 ID 的步骤以及 3.4 章节的 QSPI 配置步骤，对 QSPI 寄存器进行配置。首先将 QSPI 失能，接着将 QSPI_CTRL1 的 TXMODE 配置为 0x2（仅接收模式）、FRF 配置为 0x2（Quad SPI 模式）。因为读取 ID 指令为 0x94，因此将 QSPI_CTRL3 的 INSLEN 配置为 0x2（8 位指令）。同时将 QSPI_CTRL3 的 ADDRLEN 配置为 0x6（24 位地址长度）、WAITCYC 配置为 0x6（6 个等待时钟周期）。指令和地址的传输模式分别为单线模式和四线模式，因此 QSPI_CTRL3 的 IAT 配置为 0x1（标准 SPI 模式发送指令，FRF 规定模式发送地址）。最后还需配置 QSPI 读取数据的数量，因为 Flash ID 是一个 16bit 的数据，而一开始 QSPI 的数据位数配置为 8bit，所以将 QSPI_CTRL2 的 NDF 配置为 0x1（两个数据帧-1）。

完成以上配置后，使能 QSPI，拉低片选，往发送 FIFO 写入相应的指令和地址，然后判断接收 FIFO 不为空，并读取相应数量的数据。具体代码如下：

```
uint8_t id[2] = {0};
uint8_t i = 0;
QSPI_Disable();
QSPI->CTRL1_B.TXMODE = 0X2;
QSPI->CTRL1_B.FRF = 0X2;
QSPI->CTRL3_B.INSLEN = 0X2;
QSPI->CTRL3_B.ADDRLEN = 0X6;
QSPI->CTRL3_B.WAITCYC = 0X6;
QSPI->CTRL3_B.IAT = 0X1;
QSPI->CTRL2_B.NDF = 0X1;
QSPI_Enable();
QSPI_CS_LOW;
QSPI_TxData(0x94);
QSPI_TxData(0x000000);
for(;i<2;i++){
    while((QSPI->STS & QSPI_FLAG_RFNE) == 0){};
    id[i] = (uint32_t)QSPI->DATA;  }
while(QSPI_ReadStatusFlag(QSPI_FLAG_BUSY) == SET);

QSPI_CS_HIGH;
```

4.2 中断读取

中断读取 Flash ID 时用到的中断为接收 FIFO 满中断，因此将 QSPI_INTEN 的 RFFIE 置 1（使能接收 FIFO 满中断），其他位置 0。接收 FIFO 满中断触发条件是接收 FIFO 中的数据帧的数量超过 RFTL（接收 FIFO 阈值电平寄存器）设定的值。因此，将 QSPI_RFTL 配置为 0，每接收一个数据帧都会触发 QSPI 中断。最后，使能 QSPI 中断。QSPI 其他配置和轮询读取一致。

在 QSPI 中断函数，如果是接收 FIFO 满中断，则清除相应状态位并将接收 FIFO 的值读取出来。当读取相应数量的数据后，将片选引脚拉高，表示当前通信结束。具体代码如下：

```
uint8_t id[2] = {0};
GPIO_ResetBit(GPIOB, GPIO_PIN_6);
QSPI_Disable();
QSPI->CTRL1_B.TXMODE = 0X2;
QSPI->CTRL1_B.FRF = 0X2;
QSPI->CTRL3_B.INSLEN = 0X2;
QSPI->CTRL3_B.ADDRLEN = 0X6;
QSPI->CTRL3_B.WAITCYC = 0X6;
QSPI->CTRL3_B.IAT = 0X1;
QSPI->CTRL2_B.NDF = 0X1;
QSPI->INTEN = 0x10;
QSPI->RFTL = 0;
QSPI_Enable();
NVIC_EnableIRQ(QSPI_IRQn);
Datanum = 2;
GPIO_ResetBit(GPIOB, GPIO_PIN_6);
QSPI_TxData(0x94);
QSPI_TxData(0x000000);
while(GPIO_ReadOutputBit(GPIOB, GPIO_PIN_6) == RESET)
{

};
```

中断代码为：

```
uint16_t i = 0;
uint8_t temp[128];
void QSPI_IRQHandler(void)
{
    if(QSPI_ReadIntFlag(QSPI_INT_FLAG_RFF) == SET)
    {
        temp[i] = QSPI_RxData();
        QSPI_ClearStatusFlag();
        i++;
        if(i==datanum)
        {
            GPIO_SetBit(GPIOB, GPIO_PIN_6);
        }
    }
}
```

在发送指令和地址后或者下次发送指令或者拉低片选引脚前需要判断 CS 引脚是否已经拉高从而确保上一次通信已经完成。

5 版本历史

表格 3 文件版本历史

日期	版本	变更历史
2024.01.24	1.0	新建

声明

本手册由珠海极海半导体有限公司（以下简称“极海”）制订并发布，所列内容均受商标、著作权、软件著作权相关法律法规保护，极海保留随时更正、修改本手册的权利。使用极海产品前请仔细阅读本手册，一旦使用产品则表明您（以下称“用户”）已知悉并接受本手册的所有内容。用户必须按照相关法律法规和本手册的要求使用极海产品。

1、权利所有

本手册仅应当被用于与极海所提供的对应型号的芯片产品、软件产品搭配使用，未经极海许可，任何单位或个人均不得以任何理由或方式对本手册的全部或部分内容进行复制、抄录、修改、编辑或传播。

本手册中所列带有“®”或“TM”的“极海”或“Geehy”字样或图形均为极海的商标，其他在极海产品上显示的产品或服务名称均为其各自所有者的财产。

2、无知识产权许可

极海拥有本手册所涉及的全部权利、所有权及知识产权。

极海不应因销售、分发极海产品及本手册而被视为将任何知识产权的许可或权利明示或默示地授予用户。

如果本手册中涉及任何第三方的产品、服务或知识产权，不应被视为极海授权用户使用前述第三方产品、服务或知识产权，除非在极海销售订单或销售合同中另有约定。

3、版本更新

用户在下单购买极海产品时可获取相应产品的最新版的手册。

如果本手册中所述的内容与极海产品不一致的，应以极海销售订单或销售合同中的约定为准。

4、信息可靠性

本手册相关数据经极海实验室或合作的第三方测试机构批量测试获得，但本手册相关数据难免会出现校正笔误或因测试环境差异所导致的误差，因此用户应当理解，极海对本手册中可能出现的该等错误无需承担任何责任。本手册相关数据仅用于指导用户作为性能参数参照，不构成极海对任何产品性能方面的保证。

用户应根据自身需求选择合适的极海产品，并对极海产品的应用适用性进行有效验证和测试，以确认极海产品满足用户自身的需求、相应标准、安全或其它可靠性要求；若因用户未充分对极海产品进行有效验证和测试而致使用户损失的，极海不承担任何责任。

5、合规要求

用户在使用本手册及所搭配的极海产品时，应遵守当地所适用的所有法律法规。用户应了解产品可能受到产品供应商、极海、极海经销商及用户所在地等各国有关出口、再出口或其它法律的限制，用户（代表其本身、子公司及关联企业）应同意并保证遵守所有关于取得极海产品及／或技术与直接产品的出口和再出口适用法律与法规。

6、免责声明

本手册由极海“按原样”（as is）提供，在适用法律所允许的范围内，极海不提供任何形式的明示或暗示担保，包括但不限于对产品适销性和特定用途适用性的担保。

对于用户后续在针对极海产品进行设计、使用的过程中所引起的任何纠纷，极海概不承担责任。

7、责任限制

在任何情况下，除非适用法律要求或书面同意，否则极海和/或以“按原样”形式提供本手册的任何第三方均不承担损害赔偿责任，包括任何一般、特殊因使用或无法使用本手册相关信息而产生的直接、间接或附带损害（包括但不限于数据丢失或数据不准确，或用户或第三方遭受的损失）。

8、适用范围

本手册的信息用以取代本手册所有早期版本所提供的信息。