Application Note

Document No.: AN1108

APM32F411 Series QSPI Application Note

Version: V1.0

1 Introduction

This application note provides a guide to how to configure and apply QSPI interface on APM32F4xx series, including code implementation and application examples.

QSPI supports three frame formats and four transmission modes, so that it can connect to single-line, dual-line or quad-line external SPI Flash storage media.

At present, the QSPI module of APM32F411 is only used for communication with QSPI Flash and does not support XIP function.

Contents

2 QSPI introduction

QSPI (Quad Serial Peripheral Interface) is a serial data bus interface, which consists of clock (SCLK), chip selection signal (CS), and four data lines (IO[0:3]). It is an enhanced version of SPI, and is commonly used for communication with QSPI Flash storage devices.

2.1 QSPI modes

To meet the design requirements in different environments, QSPI supports three different transmission modes, namely, standard SPI (single-line) mode, Dual SPI (dual-line) mode, and Quad SPI (quad-line) mode. The IO pins used in the three different frame formats are also different, specifically as follows:

- **Standard SPI mode:** Clock (SCLK), chip select signal (CS), MOSI (IO0), MISO (IO1).
- **Dual SPI mode:** Clock (SCLK), chip select signal (CS), 2 data lines (IO [0:1]).
- **Quad SPI mode:** Clock (SCLK), chip select signal (CS), 4 data lines (IO [0:3]).

2.1.1 Standard SPI mode

The standard SPI mode of QSPI supports full duplex transmission, but due to the limitation of the connected storage devices, it is usually half duplex transmission. In the standard mode, the data is transmitted on the IO0 and IO1 lines, the IO0 line is MOSI and the IO1 line is MISO. IO2 and IO3 are connected to the nWP and nHOLD of the QSPI Flash, and output a high level to cancel the write protection function and activate the holding function. The specific hardware connection is shown in Figure 1 Hardware Connection Diagram of Standard SPI Mode:

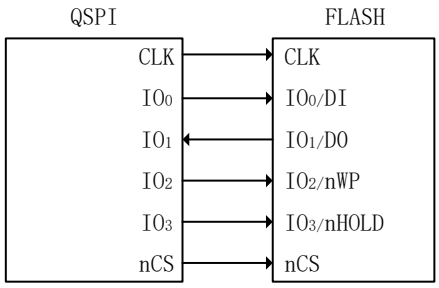

Figure 1 Hardware Connection Diagram of Standard SPI Mode

From the above figure, it can be seen that in the standard SPI mode, QSPI transmits data to Flash through IO0 and receives data from Flash through IO1. If nWP and nHOLD are connected through QSPI Flash hardware, the IO3 and IO4 of QSPI can be used as other IO.

To apply the standard SPI mode, the frame format (FRF) of the control register 1 (QSPI_CTRL1) needs to be configured as 0x0.

2.1.2 Dual SPI mode

In the Dual SPI mode, the hardware connection is similar to that in the standard SPI mode, but the Dual SPI mode uses two data lines to send and receive data simultaneously. QSPI Flash also converts the two unidirectional pins of DI and DO into bidirectional pins IO0 and IO1. The specific hardware connection is shown in Figure 2 Hardware Connection Diagram of Dual SPI Mode:

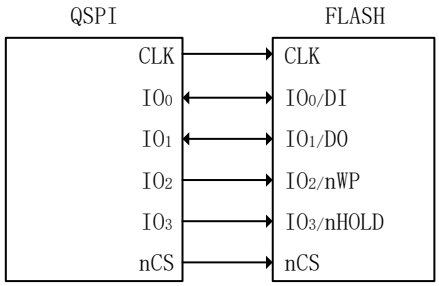

Figure 2 Hardware Connection Diagram of Dual SPI Mode

From the above figure, it can be seen that IO0 and IO1 together complete the data transmission, thus reducing the time for data transmission. Compared with the standard SPI mode, it greatly improves the throughput of data.

To apply the Dual SPI mode, the frame format (FRF) of the control register 1 (QSPI_CTRL1) needs to be configured as 0x1.

2.1.3 Quad SPI mode

In the Quad mode, the QSPI uses four data lines to send or receive data simultaneously. In this mode, the QSPI Flash will enable four-way mode, and nWP and nHOLD pins are multiplexed into IO ports for data transmission, without write protection and holding function. The specific hardware connection is shown in Figure 3 Hardware Connection Diagram of Quad SPI Mode:

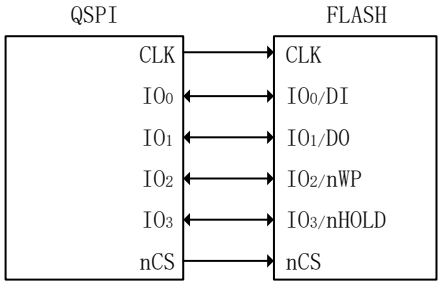

Figure 3 Hardware Connection Diagram of Quad SPI Mode

Comparing the hardware connection diagrams of Quad SPI mode and Dual SPI mode, it can be seen that the data transmission changes from the two IO ports of IO0 and IO1 to transmission from the four IO ports of IO0~3, and the throughput is significantly improved.

To apply the Quad SPI mode, the frame format (FRF) of the control register 1 (QSPI_CTRL1) needs to be configured as 0x2.

2.2 Composition of QSPI frame format

In the process of information transmission between QSPI and storage devices, the data flow of each operation is called a frame. The frame of QSPI usually consists of four stages, namely instruction segment, address segment, idle byte segment, and data segment. The detail of QSPI frame is shown in Figure 4 QPSI Four-way Read Sequence.

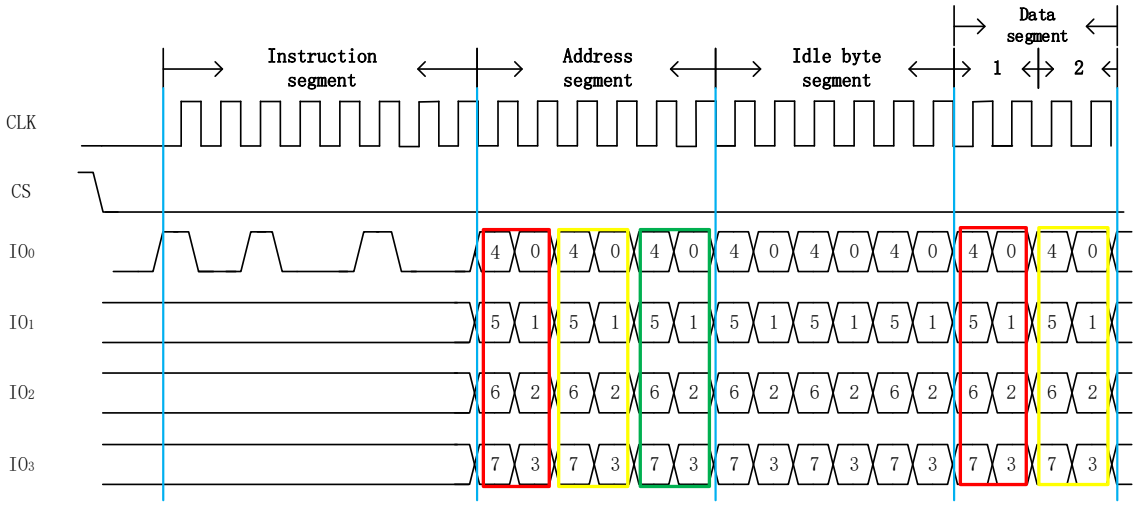

Figure 4 QPSI Four-way Read Sequence

The above figure is the sequential logic diagram of QSPI four-way reading of QSPI Flash ID. The instruction segment is in single-line mode, and the address and data segments are in quad-line mode. The transmission mode of each segment is configured by the FRF of control register 1 (QSPI_CTRL1) and the IAT of control register 3 (QSPI_CTRL3), as detailed in subsequent chapters.

2.2.1 Instruction segment

In the instruction segment, QSPI will send an 8-bit instruction to QSPI Flash, and the specific instrument can be configured by the user based on the software design and hardware requirements. QSPI can choose to send the instruction in single-line mode, double-line mode, or quad-line mode, with the specific configuration described below.

1. To send instructions in the single-line mode, the IAT bit of control register 3 (QSPI_CTRL3) needs to be configured to 0x00 or 0x01. The instruction sending sequence in single-line mode is shown in Figure 5 Instruction Sending Sequence in Single-line Mode.

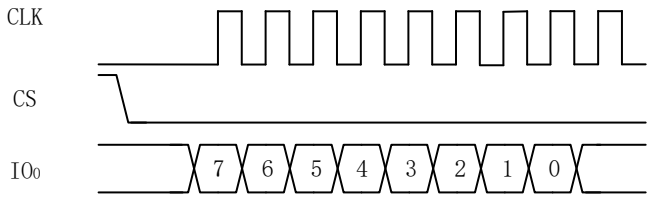

Figure 5 Instruction Sending Sequence in Single-line Mode

2. To send instructions in the Dual-SPI mode, the IAT bit of control register 3 (QSPI_CTRL3) needs to be configured to 0x2 and the FRF bit of control register 1 (QSPI_CTRL1) needs to be configured to 0x1. The instruction sending sequence in dual-line mode is shown in Figure 6 Instruction Sending Sequence in Dual-line Mode.

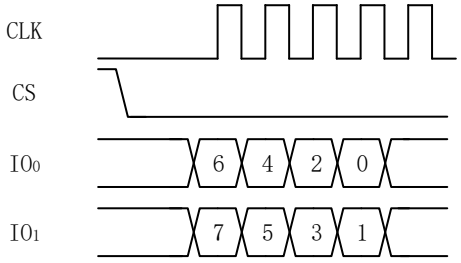

Figure 6 Instruction Sending Sequence in Dual-line Mode

3. To send instructions in the quad-line mode (Quad SPI mode), the IAT bit of control register 3 (QSPI_CTRL3) needs to be configured to 0x2 and the FRF bit of control register 1 (QSPI_CTRL1) needs to be configured to 0x2. The instruction sending sequence in quad-line mode is shown in Figure 7 Instruction Sending Sequence in Quad-line Mode.

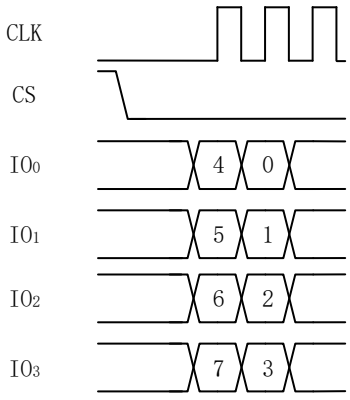

Figure 7 Instruction Sending Sequence in Quad-line Mode

In addition to the sending mode of instructions, the length of instructions can also be configured,

as shown in Table 1 Instruction Length Configuration Selection.

Table 1 Instruction Length Configuration Selection

Instruction length selection	Register related configuration
No instruction	CTRL3[9:8] INSLEN = 0X00
4-bit instruction	CTRL3[9:8] INSLEN = 0X01
8-bit instruction	CTRL3[9:8] INSLEN = 0X02
16-bit instruction	CTRL3[9:8] INSLEN = 0X03

Note: The single-line mode does not require configuration of instruction length.

2.2.2 Address segment

In the address segment, QSPI will send to QSPI Flash the address where the data needs to be written or read. The address length and the sending mode can be configured by the user, and the sending mode configuration of the address is similar to that of the instruction. The details are as follows:

(1). Single-line mode (standard SPI mode): Configure the IAT of control register 3 (QSPI_CTRL3) to 0x00 and the addresses can be sent in single-line mode.

(2). Dual-line SPI mode (Dual SPI mode): Configure the IAT of control register 3 (QSPI_CTRL3) to 0x01 or 0x02, and the FRF of control register 1 (QSPI_CTRL1) to 0x1 and the addresses can be sent in dual-line SPI mode.

(3). Quad-line SPI mode (Quad SPI mode): Configure the IAT of control register (QSPI_CTRL3) to 0x01 or 0x02, and the FRF of control register 1 (QSPI_CTRL1) to 0x2 and the addresses can be sent in quad-line SPI mode.

The length of the address is configured by the ADDRLEN of control register 3 (QSPI_CTRL3), and it can be configured to send without an address and have an address length of 4~60 bits (a multiple of 4). The single-line mode does not require configuration of address length; please refer to the APM32F411xCxE User Manual for details.

2.2.3 Idle byte segment

The idle byte segment is mainly used for fast reading or fast writing to QSPI Flash. During fast operation, QSPI Flash may require additional clock cycles to prepare for data transmission. Similarly, QSPI also needs to delay several clock cycles for data reading to ensure the

correctness of the read data.

The WAITCYC bit of control register 3 (QSPI_CTRL3) can be used to configure the clock cycles occupied by the idle bytes, and a maximum of 31 clock cycles can be configured for the idle bytes.

2.2.4 Data segment

The data segment is mainly used for writing and reading of data. The transmission mode and quantity can also be configured by the user. The transmission mode is configured in the FRF of control register 1 (QSPI_CTRL1), specifically as follows:

- 0x00: Single-line mode (standard SPI mode)
- 0x01: Dual-line mode (Dual SPI mode)
- 0x02: Quad-line mode (Quad SPI mode)

In dual-line and quad-line modes, the quantity of data frames is configured by the NDF of control register 2 (QSPI_CTRL2). The configuration of this bit determines the number of data frames written or received by QSPI. In single-line mode, there is no need to configure this register.

2.3 QSPI transmission mode

The QSPI module of APM32F411 supports a total of four transmission modes, namely, send and receive, send only, receive only, and EEPROM (Electrically Erasable Programmable Read Only Memory) read. The transmission mode (TXMODE) is determined by TXMODE of the control register 1 (QSPI_CTRL1). The specific differences among the four transmission modes are as follows:

(1). Send and receive Mode: Only applicable to sending and receiving in standard SPI mode.

(2). Send only mode: Can send only in standard SPI mode, and receiving data on the data line will be ignored. Or it can be used for write operation in enhanced mode (Dual SPI mode and Quad SPI mode).

(3). Receive only mode: Can receive only in standard SPI mode. Or it can be used for read operation in enhanced mode (Dual SPI mode and Quad SPI mode).

(4). EEPROM read mode: Only used for operating EEPROM.

2.4 Use of QSPI FIFO

QSPI integrates the FIFO buffer used for sending and receiving. The FIFO buffer has a depth of 8 bits and a length of 32 bits. Therefore, the number of transmitted and received data bits is fixed at 32. The data frames with less than 32 bits will be right-aligned when they are written to the transmit FIFO buffer.

After data is written to the DATA register of QSPI, if the transmit FIFO is not full, the data in the DATA will be filled into the transmit FIFO. When reading data from the DATA register, if there is data in the receive FIFO, the data will be read out.

2.5 QSPI interrupt and DMA

2.5.1 QSPI interrupt

QSPI supports 6 types of interrupts. The details about enabling interrupts, checking status bits, and clearing flag bits are shown in the table below.

Table 2 QSPI Interrupt Details

Interrupt type	Enable bit	Status Bit	Clear bit	Interrupt trigger details
Transmission interruption	TFEIE	TFEIF	ICF	Filling data into the transmit FIFO will trigger an interrupt
Transmit FIFO overflow interrupt	TFOIE	TFOIF	TFOIC	Filling data into a fully loaded transmit FIFO will trigger an interrupt
Receive FIFO underflow interrupt	RFUIE	RFUIF	RFUIC	Reading an empty receive FIFO will trigger an interrupt
Receive FIFO overflow interrupt	RFOIE	RFOIF	RFOIC	When the fully loaded receive FIFO receives the data sent by the memory, an interrupt will be triggered
Receive FIFO full interrupt	RFFIE	RFFIF	ICF	When the data of the receive FIFO exceeds the RFT set value, an interrupt will be triggered
Multi-master contention interrupt	MSTIE	MSTIF	MIC	When contention occurs on the serial bus, it will trigger an interrupt

2.5.2 Use of QSPI DMA

The DMA function of QSPI can be enabled by configuring the RDMANE bit and TDMAEN bit of the DMA control register (QSPI_DMACTRL).

3 QSPI configuration

This chapter will introduce the basic configuration when the QSPI module is used, and the configuration methods of three different modes of QSPI. The content involving QSPI Flash will not be discussed in this chapter. Readers can check it themselves.

3.1 QSPI initialization

This section will discuss the QSPI initialization, mainly including GPIO initialization, clock configuration, and QSPI initialization. Please refer to the following content for details.

3.1.1 GPIO initialization

In GPIO initialization, first open the corresponding GPIO clock of the used pin and configure its IO port multiplexing function. Configure the CS pin to output mode and the other pins to multiplex push-pull mode. Finally, pull up the CS pin. The specific code configuration is as follows:

```
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);
```

Note: The QSPI related IO ports of APM32F411 are distinguished for BK1 and BK2, e.g. QSPI_BK1_IO1 and QSPI_BK2_IO1. The parameters may be different between them when configuring the multiplex function. The parameter for configuring the multiplex function for IO of BK1 is AF10 (GPIO_AF10_QSPI), and the parameter for configuring the multiplex function for IO of BK2 is AF9 (GPIO_AF_QSPI). Except for the difference in the parameters for configuring the multiplex function, there is no difference in the use of QSPI between the IO of BK1 and BK2.

3.1.2 Frequency division of QSPI clock

The QSPI clock source is provided by AHB, and it needs to be divided by the QSPI_BR register to obtain the final QSPI clock. The details are shown below.

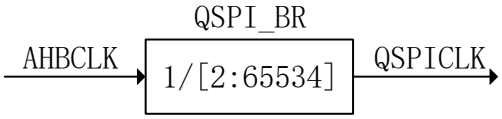

Figure 8 Frequency Division Configuration Diagram of QSPI Clock

Note: The LSB of the CLKDIV field of QSPI_BR is always set to 0. Therefore, the clock division value of QSPI is any even number between 2 and 65534, and the highest division value is 65534. For example, if the clock of AHB is 100MHz and the division value of QSPI is set to 65534, QSPI's clock QSPICLK = 100/65534=1.5259KHz.

3.1.3 QSPI initialization

In QSPI initialization, first turn on the clock of QSPI, set the IOSW bit to 1 to turn on the corresponding GPIO port of QSPI, then configure the clock division of QSPI in CLKDIV of the baud rate register (QSPI_BR), and configure the size (DFS) of the data frame, clock phase and polarity (CPHA and CPOL), reverse enable of chip selection (SSTEN), and frame format selection (FRF) in control register 1 (QSPI_CTRL1). Finally, enable the slave (SLAEN) so that the data can be transmitted. It is important to note that after QSPI (SSIEN) is enabled, the control register 1 (QSPI_CTRL1) cannot be written. Therefore, every time before switching the working mode of QSPI, disable QSPI, namely, enable the EN bit of the register (QSPI_SSIEN) to 0. Change the control register 1 according to the required working mode and then enable the QSPI. The specific code configuration is as follows:

```
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();
```

The initialization of QSPI can be completed through the above two steps, and it will be explained below according to different QSPI modes.

It should be noted that when using QSPI to operate Flash, in addition to enabling QSPI, the slave (SLAEN) also needs to be enabled. If the slave is not enabled, the data cannot be sent out in the FIFO area.

3.2 Standard SPI mode

Before making QSPI communication in standard SPI mode, configure QSPI to standard SPI mode; the specific steps are as follows:

1. Disable QSPI;
2. Modify the frame format of QSPI to standard SPI mode;
3. Modify the transmission direction of QSPI according to your own needs;
4. Enable QSPI;

After completing the above steps, you can send corresponding commands or data to operate the QSPI Flash. In standard SPI mode, the buffer queue FIFO can be used for multiple send and multiple receive, or be used for single send and receive like SPI. The specific implementation steps of these two methods are as follows:

a. Single send and receive: When conducting single send and receive in standard SPI mode, the processing flow of QSPI is consistent with that of SPI. After QSPI completes initialization, enable the slave, set the SLAEN bit to 1, and then fill data into the FIFO. Every time the data is

filled, QSPI will send it out, then wait for RFNEF to be set to 1 (receive FIFO is not empty), and finally read the data from the receive FIFO. The specific flow is shown in Figure 9 Single Transmission Flow Chart of Standard SPI Mode.

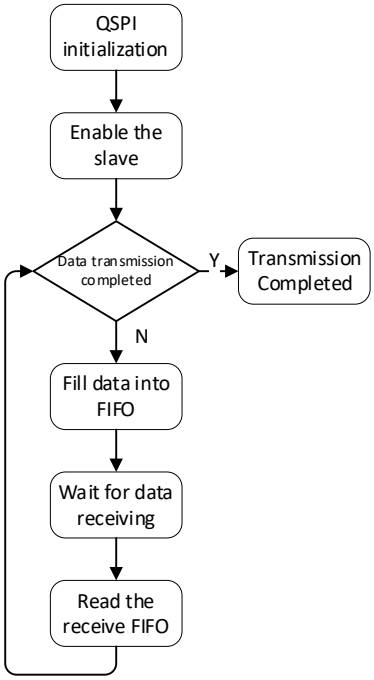

Figure 9 Single Transmission Flow Chart of Standard SPI Mode

b. Batch send and receive: When conducting batch send and receive in standard SPI mode, the operation flow of QSPI is shown in Figure 10 Batch Transmission Flow Chart of Standard SPI Mode. First, disable the slave, set SLAEN to 0, and then fill the data into the transmit FIFO. The FIFO size of QSPI is 8*32bit. Therefore, after filling the FIFO with data 8 times, the TFNF of the QSPI_STS register will be set to zero, indicating that the transmit FIFO has been full. If the transmit FIFO is full, enable the slave (set SLAEN to 1) to send the data from the FIFO and wait for the data to be received and read. After completing the data reading, disable the slave and continue to fill data into the transmit FIFO. When all data has been filled into the transmit FIFO, enable the slave to send the remaining data and wait for the data to be received and read.

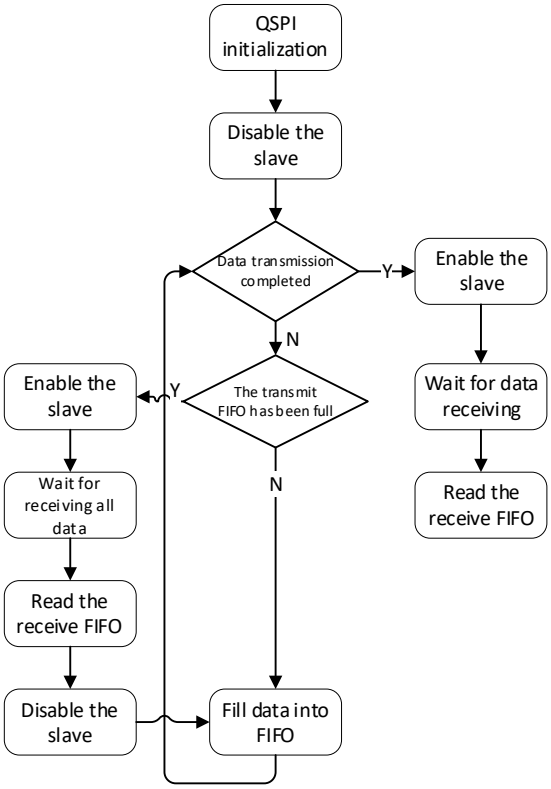

Figure 10 Batch Transmission Flow Chart of Standard SPI Mode

3.3 Dual SPI mode

Before making communication in Dual SPI (dual-line) mode, configure QSPI to Dual SPI mode, and configure such parameters as instruction length and address length; the specific configuration steps are as follows:

1. Disable QSPI (EN bit of QSPI_SSIEN);
2. Configure data transmission mode (TXMODE bit of QSPI_CTRL1);
3. Modify the frame format of QSPI to Dual SPI mode (FPF bit of QSPI_CTRL1);
4. Configure the instruction length of QSPI (INSLEN of QSPI_CTRL3);

5. Configure the address length of QSPI (ADDRLEN of QSPI_CTRL3);
6. Configure the wait cycle of QSPI (WAITCYC of QSPI_CTRL3);
7. Configure the data size of QSPI (NDF of QSPI_CTRL2);
8. Configure the instruction address type of QSPI (IAT of QSPI_CTRL3);
9. Enable QSPI (EN bit of QSPI_SSIEN);

The instruction length and address length determine the number of clock cycles for QSPI to send instructions and addresses. If these two items are not configured correctly, QSPI cannot send complete instructions or the instructions cannot be recognized by QSPI Flash. The wait cycle determines the length of the clock cycle for idle bytes, which ensures QSPI reads data correctly. The data size refers to the amount of data that needs to be read or written.

It should be noted that the data type of the data stage follows the QSPI mode, but the data type of the instruction stage and address stage are determined by the IAT of CTRL3. There are three options in total, namely (1) the instruction and address are in single-line mode; (2) the instruction is in single-line mode, and the address follows the QSPI mode; and (3) both the instruction and the address follow the QSPI mode. This item is mainly selected depending on whether the QSPI storage device supports relevant transmission modes. Generally speaking, instruction is in single-line mode, and the address follows QSPI mode.

After completing the above configuration, the instruction and address that need to be written can be filled into the transmit FIFO. If it is a read instruction, read the data; if it is a write instruction, write the data. The following code takes reading data as an example:

```
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;
}
```

From the above read operation code of the Dual SPI mode, it can be learnt that the instruction stage, address stage, and data stage can all be omitted, depending on whether QSPI Flash supports it and the selected instruction.

The transmission logic diagram for reading data in Dual SPI mode is shown in Figure 11 Transmission Logic Diagram of Dual SPI Mode. The read data is the ID information of QSPI Flash, the instruction length and address length are 8 bits and 24 bits, respectively, the number of data is 4, and the size of data frame is 8 bits.

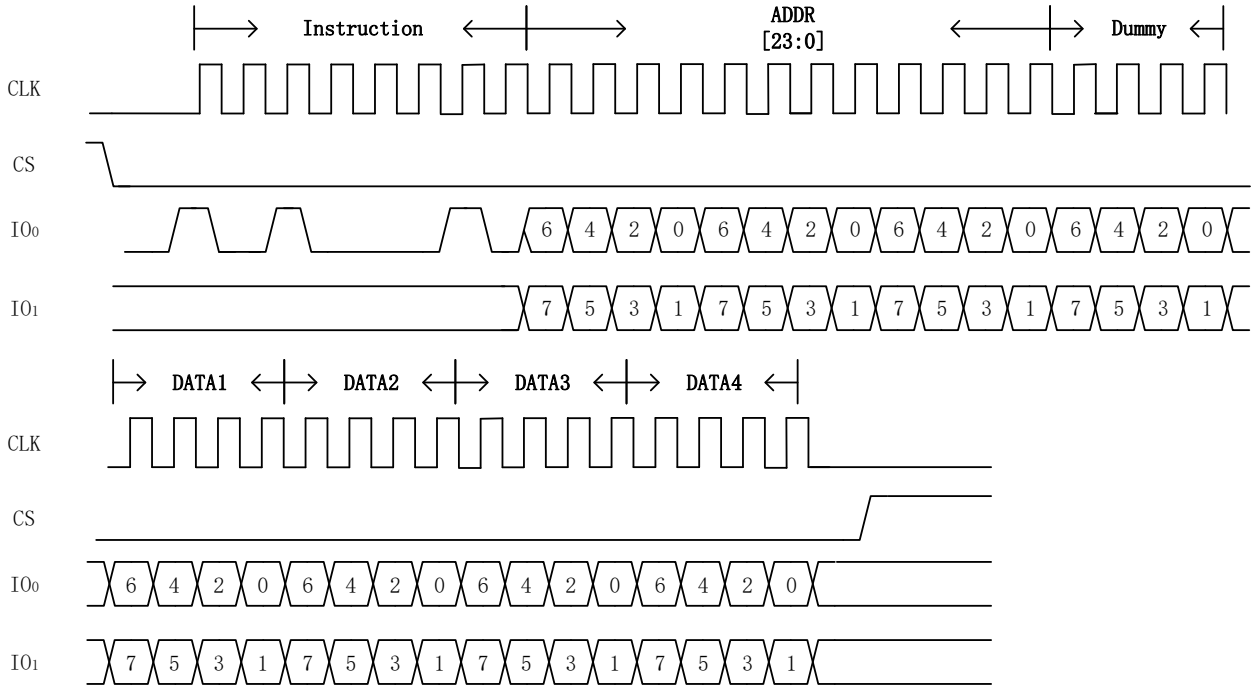

Figure 11 Transmission Logic Diagram of Dual SPI Mode

From the above figure, it can be seen that in Dual SPI mode, the data is transmitted on the IO0 and IO1 lines. Therefore, the transmission of 8-bit data requires 4 clock cycles, IO1 is bit7, bit5, bit3, and bit1, and IO0 is bit6, bit4, bit2, and bit0. At the same time, after sending the address, the data is not read immediately, but is read after four clock cycles. These four clock cycles are idle bytes. During fast operation, free bytes are required before reading data to ensure the correctness of the data.

3.4 Quad SPI mode

The configuration of Quad SPI mode is roughly the same as that of Dual SPI mode, except that the QSPI mode configuration and the number of wait cycles need to be modified. The wait cycles required for Quad SPI mode and Dual SPI mode are different. The specific configuration steps are as follows:

1. Disable QSPI (EN bit of QSPI_SSIEN);
2. Configure data transmission mode (TXMODE bit of QSPI_CTRL1);
3. Modify the frame format of QSPI to Quad SPI mode (FPF bit of QSPI_CTRL1);
4. Configure the instruction length of QSPI (INSLEN of QSPI_CTRL3);
5. Configure the address length of QSPI (ADDRLEN of QSPI_CTRL3);
6. Configure the wait cycle of QSPI (WAITCYC of QSPI_CTRL3);
7. Configure the data size of QSPI (NDF of QSPI_CTRL2);

8. Configure the instruction address type of QSPI (IAT of QSPI_CTRL3);
9. Enable QSPI (EN bit of QSPI_SSIEN);

After completing the above configuration, the corresponding instructions and addresses can be written to the FIFO. The instruction length here configured is 8, the address length is 24, the number of idle cycles is 6, and two 8-bit data is read. The transmission logic diagram of Quad SPI mode is shown in Figure 12 Transmission Logic Diagram of Quad SPI ModeFigure 12 Transmission Logic Diagram of Quad SPI Mode.

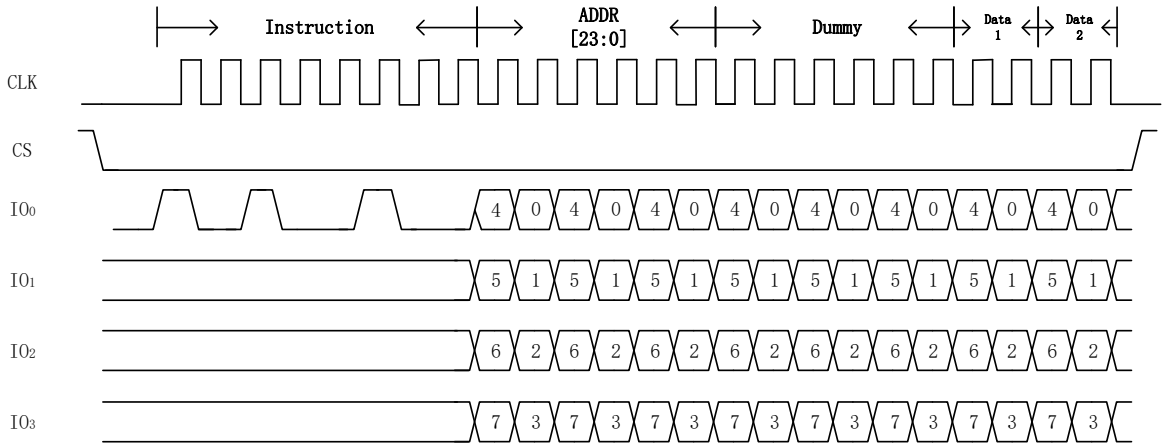

Figure 12 Transmission Logic Diagram of Quad SPI Mode

It can be seen from the above figure that, in Quad SPI mode, the data is output in parallel by four data lines. The transmission speed is significantly improved compared with the Dual SPI mode.

3.5 QPI mode

The QPI mode is a special mode in Quad SPI mode. The difference between it and Quad SPI mode is that the instruction stage is transmitted in quad-line mode. Therefore, the configuration of QPI mode is based on the configuration of Quad SPI mode, modifying the IAT of QSPI_CTRL3 to 0x10, that is, choosing to send instructions and addresses in the mode specified by SPI_FRF, which means sending instructions and addresses in quad-line mode.

4 Quad SPI reading FLASH ID

This chapter will introduce how to design codes to read the ID information of QSPI Flash W25Q64FV. QSPI Flash is configured to quad-line enable mode by default. This chapter will not discuss how to configure QSPI Flash to quad-line enable mode.

4.1 Polling read

Before designing polling read of Flash ID codes, you need to understand the steps for reading this Flash ID. The instruction for reading the QSPI Flash ID in quad-line mode is 0x94, the address is 24bit 0x000000 and an idle byte cycle of 6 bytes needs to be waited for. The final read ID is 16-bit data.

Configure the QSPI register according to the above steps for reading the ID and the QSPI configuration steps in 3.4. First, disable QSPI, and then configure TXMODE of QSPI_CTRL1 to 0x2 (receive-only mode) and FRF to 0x2 (Quad SPI mode). As the instruction for reading ID is 0x94, the INSLEN of QSPI_CTRL3 is configured as 0x2 (8-bit instruction). At the same time, configure the ADDRLEN of QSPI_CTRL3 to 0x6 (24-bit address length) and WAITCYC to 0x6 (6 wait clock cycles). The transmission modes of instructions and addresses are single-line mode and quad-line mode, respectively. Therefore, configure the IAT of QSPI_CTRL3 to 0x1 (sending instructions in standard SPI mode and sending addresses in FRF specified mode). Finally, configure the size of data read by QSPI. As the Flash ID is a 16-bit data, and the data bits of the QSPI are configured to 8 bits at the beginning, configure the NDF of QSPI_CTRL2 to 0x1 (two data frames -1).

After completing the above configuration, enable QSPI, pull down the chip selection pin, write the corresponding instruction and address to the transmit FIFO, then check if the receive FIFO is empty, and read the corresponding amount of data. The specific code is as follows:

```
 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 Interrupt reading

The interrupt used to interrupt reading the Flash ID is the receive FIFO full interrupt, so set the RFFIE of QSPI_INTEN to 1 (enable the receive FIFO full interrupt), and other bits to 0. The trigger condition for receive FIFO full interrupt is that the number of data frames in the receive FIFO exceeds the set value of RFTL (receive FIFO threshold Level register). Therefore, configure QSPI_RFTL to 0, and every time a data frame is received, a QSPI interrupt will be triggered. Finally, enable QSPI interrupt. Other configurations of QSPI are consistent with polling read.

```
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)
{

};
```

In the QSPI interrupt function, if the receive FIFO is full, clear the corresponding status bit and read the value of the receive FIFO. After reading a corresponding amount of data, pull up the chip selection pin to indicate the end of current communication. The specific code is as follows:

The interrupt code is:

```
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);
        }
    }
}
```

After sending the instructions and addresses, or before sending the instructions or pulling down the chip selection pin next time, judge whether the CS pin has been pulled up to ensure that the previous communication has been completed.

5 Revision history

Table 3 Document Revision History

Date	Version	Revision History
January 24, 2024	1.0	New edition

Statement

This manual is formulated and published by Zhuhai Geehy Semiconductor Co., Ltd. (hereinafter referred to as "Geehy"). The contents in this manual are protected by laws and regulations of trademark, copyright and software copyright. Geehy reserves the right to correct and modify this manual at any time. Please read this manual carefully before using the product. Once you use the product, it means that you (hereinafter referred to as the "users") have known and accepted all the contents of this manual. Users shall use the product in accordance with relevant laws and regulations and the requirements of this manual.

1. Ownership of rights

This manual can only be used in combination with chip products and software products of corresponding models provided by Geehy. Without the prior permission of Geehy, no unit or individual may copy, transcribe, modify, edit or disseminate all or part of the contents of this manual for any reason or in any form.

The "Geehy" or "Geehy" words or graphics with "®" or "TM" in this manual are trademarks of Geehy. Other product or service names displayed on Geehy products are the property of their respective owners.

2. No intellectual property license

Geehy owns all rights, ownership and intellectual property rights involved in this manual.

Geehy shall not be deemed to grant the license or right of any intellectual property to users explicitly or implicitly due to the sale and distribution of Geehy products and this manual.

If any third party’s products, services or intellectual property are involved in this manual, it shall not be deemed that Geehy authorizes users to use the aforesaid third party’s products, services or intellectual property, unless otherwise agreed in sales order or sales contract of Geehy.

3. Version update

Users can obtain the latest manual of the corresponding products when ordering Geehy products.

If the contents in this manual are inconsistent with Geehy products, the agreement in Geehy sales order or sales contract shall prevail.

4. Information reliability

The relevant data in this manual are obtained from batch test by Geehy Laboratory or cooperative third-party testing organization. However, clerical errors in correction or errors caused by differences in testing environment may occur inevitably. Therefore, users should understand that Geehy does not bear any responsibility for such errors that may occur in this manual. The relevant data in this manual are only used to guide users as performance parameter reference and do not constitute Geehy's guarantee for any product performance.

Users shall select appropriate Geehy products according to their own needs, and effectively verify and test the applicability of Geehy products to confirm that Geehy products meet their own needs, corresponding standards, safety or other reliability requirements. If loses are caused to users due to the user's failure to fully verify and test Geehy products, Geehy will not bear any responsibility.

5. Compliance requirements

Users shall abide by all applicable local laws and regulations when using this manual and the matching Geehy products. Users shall understand that the products may be restricted by the export, re-export or other laws of the countries of the product suppliers, Geehy, Geehy distributors and users. Users (on behalf of itself, subsidiaries and affiliated enterprises) shall agree and promise to abide by all applicable laws and regulations on the export and re-export of Geehy products and/or technologies and direct products.

6. Disclaimer

This manual is provided by Geehy "as is". To the extent permitted by applicable laws, Geehy does not provide any form of express or implied warranty, including without limitation the warranty of product merchantability and applicability of specific purposes.

Geehy will bear no responsibility for any disputes arising from the subsequent design and use of Geehy products by users.

7. Limitation of liability

In any case, unless required by applicable laws or agreed in writing, Geehy and/or any third party providing this manual "as is" shall not be liable for damages, including any general damages, special direct, indirect or collateral damages arising from the use or no use of the information in this manual (including without limitation data loss or inaccuracy, or losses suffered by users or third parties).

8. Scope of application

The information in this manual replaces the information provided in all previous versions of the manual.