

32 位微控制器

HC32F460 系列的四线式串行外设接口 QSPI

适用对象

F 系列	HC32F460
------	----------

目 录

1	摘要	3
2	QSPI 简介	3
2.1	主要特性.....	3
2.2	内存映射.....	4
3	HC32F460 系列的 QSPI.....	5
3.1	帧格式	5
3.1.1	指令	6
3.1.2	地址	7
3.1.3	虚拟周期	8
3.1.4	数据	8
3.2	通信协议.....	9
3.2.1	扩展式 SPI 协议	9
3.2.2	二线式 SPI 协议	10
3.2.3	四线式 SPI 协议	10
3.3	总线模式.....	11
3.3.1	ROM 访问模式.....	11
3.3.2	直接通信模式	11
3.4	特殊功能.....	12
3.4.1	闪存预读取功能.....	12
3.4.2	XIP 模式	13
3.5	寄存器介绍.....	14
3.6	注意事项.....	14
3.6.1	寄存器的设置顺序	14
3.6.2	模块停止信号的设置.....	14
4	样例代码	15
4.1	代码介绍.....	15
4.2	工作流程.....	17
4.3	代码运行.....	18
5	总结	19
6	版本信息 & 联系方式	20

1 摘要

本篇应用笔记主要介绍 HC32F460 系列的四线式串行外设接口（QSPI）模块，并简要说明通过 QSPI 四线式输入输出快速读模式如何与外部 Flash 通信。

2 QSPI 简介

HC32F460 系列的四线式串行外设接口（QSPI）是一个存储器控制模块，主要用于和带 SPI 兼容接口的串行 ROM 进行通信，其对象主要包括有串行闪存、串行 EEPROM 以及串行 FeRAM。

2.1 主要特性

- 支持扩展 SPI，二线式 SPI 和四线式 SPI 等多种协议
- 地址线宽度可选择 8 位/16 位/24 位/32 位
- 可通过时序调整以支持各种串行闪存
- 支持多种读取方式
 - 标准读/快速读
 - 二线式输出快速读取/二线式输入输出快速读取
 - 四线式输出快速读取/四线式输入输出快速读取
- 数量可调的虚拟周期
- 16 字节的预读取功能
- 总线周期延长功能
- XIP 控制功能
- 灵活而广泛的支持大量串行闪存软件控制指令，包括擦、写、ID 读取及掉电控制等

2.2 内存映射

串行闪存及相关的控制寄存器在 AHB 总线空间的位置由总体的地址范围配置来决定，QSPI 空间被区分为 2 段空间，包括 QSPI I/O 寄存器空间 64MB 和外部 QSPI 设备空间 64MB，分配关系请参考下表：

QSPI	0x98000000	0x9FFFFFFF	128MB
QSPI I/O 寄存器	0x9C000000	0x9FFFFFFF	64MB
外部 QSPI 设备	0x98000000	0x9BFFFFFF	64MB

每当对 QSPI 的 ROM 空间进行读访问时，QSPI 总线自动开始工作，将从串行闪存内读到的数据传送过来，QSPI 可以通过自动将 MCU 的外部 ROM 读取总线周期转换为 QSPI 总线周期来对串行闪存进行读取。

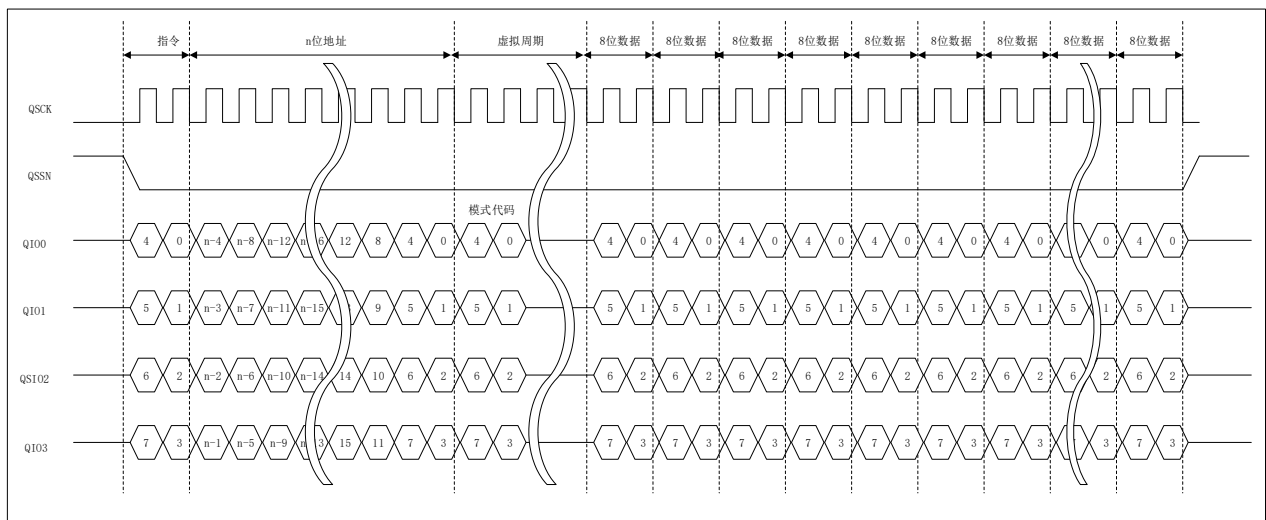
3 HC32F460 系列的 QSPI

3.1 帧格式

HC32F460 系列的四线式串行外设接口（QSPI）支持扩展式 SPI，二线式 SPI 和四线式 SPI 三种协议。初始的默认协议是扩展式 SPI 协议，可通过设置 QSCR 寄存器中的 IPRSL[1:0]/APRSL[1:0]/DPRSL[1:0] 位来分别配置指令发送阶段、地址发送阶段、数据接收阶段的协议。

通过 QSCR 寄存器中的 MDSEL 可以配置 QSPI 的读取模式，一般配置为推荐的模式即可正常通信，如配置为自定义的模式，则 QSCR 寄存器中的 IPRSL[1:0]/APRSL[1:0]/DPRSL[1:0] 位必须配置为同一种协议才能保证 QSPI 正常工作。且直接通信模式不支持多线式动作。

四线式 SPI 协议动作示意图：



3.1.1 指令

当一个串行总线周期开始的时候，串行闪存选择信号被置为有效状态，QSPI 开始输出指令代码，指令代码为一个 8-bits 的数据，可以发送任何有效的指令值，指令代码需要在串行总线周期开始之前配置好，通过 QSCCMD 寄存器进行配置。

指令阶段配置如下表：

寄存器配置		ROM 访问模式	直接通信模式	指令格式
QSCR[9:8]	扩展式 SPI	IPRSL[1:0] = 00	IPRSL[1:0]=00	
	二线式 SPI	IPRSL[1:0] = 01	不支持	
	四线式 SPI	IPRSL[1:0] = 10	不支持	

3.1.2 地址

在这个阶段，一个地址被发送到闪存。QSPI 拥有 32 位地址总线宽度来配合串行闪存，可以通过设置 QSFCR 寄存器内的 AWSL[1:0] 来选择使用 8 位/16 位/24 位/32 为地址总线宽度。如果选择 8 位/16 位/24 位的地址总线宽度，那么只有地址与之匹配的低位空间可以被正常访问，访问 QSPI 中高位的串行闪存镜像空间将会反复出现低位空间的内容。

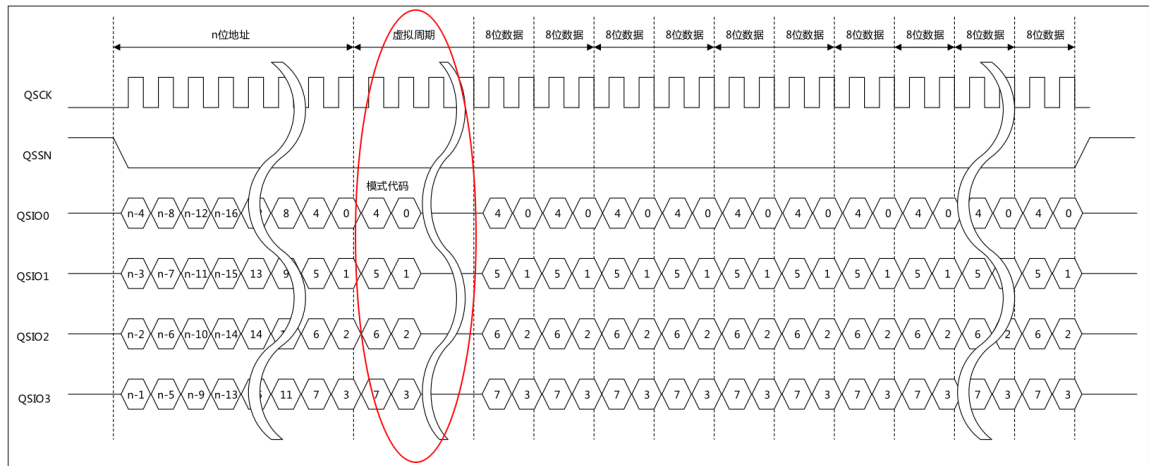
地址阶段配置如下表：

寄存器配置		ROM 访问模式	直接通信模式
QSFCR[1:0]	1 字节	AWSL[1:0] = 00	AWSL[1:0] = 00
	2 字节	AWSL[1:0] = 01	AWSL[1:0] = 01
	3 字节	AWSL[1:0] = 10	AWSL[1:0] = 10
	4 字节	AWSL[1:0] = 11	AWSL[1:0] = 11
QSCR[11:10]	扩展式 SPI	APRSL[1:0] = 00	APRSL[1:0] = 00
	二线式 SPI	APRSL[1:0] = 01	不支持
	四线式 SPI	APRSL[1:0] = 10	不支持

3.1.3 虚拟周期

在快速读指令的情况下，需要在发送地址之后加入一定数量的虚拟周期，其具体数量由 QSFRCR 寄存器中的 DMCYCN[3:0] 决定。虚拟周期最初的两个周期用于决定是否选择 XIP 模式。

虚拟周期阶段配置如下图：



3.1.4 数据

在这个阶段，数据发送到 QSPI 闪存或者从 QSPI 闪存接收数据，直接通信模式下一个完整的 QSPI 总线周期从对寄存器 QSDCOM 的 DCOM[7:0] 第一次操作开始直到对 QSCR 寄存器的进行一次写操作后结束。对 DCOM[7:0] 的写会转换为一次 QSPI 总线的单字节的数据传送，而对 DCOM[7:0] 的读则会转换成一次 QSPI 总线的单字节的数据接收。

数据阶段配置如下表：

寄存器配置		ROM 访问模式	直接通信模式
数据	读数据	直接访问内存映射地址	QSDCOM
	写数据	不支持	QSDCOM
QSCR[13:12]	扩展式 SPI	APRSL[1:0] = 00	APRSL[1:0] = 00
	二线式 SPI	APRSL[1:0] = 01	不支持
	四线式 SPI	APRSL[1:0] = 10	不支持

3.2 通信协议

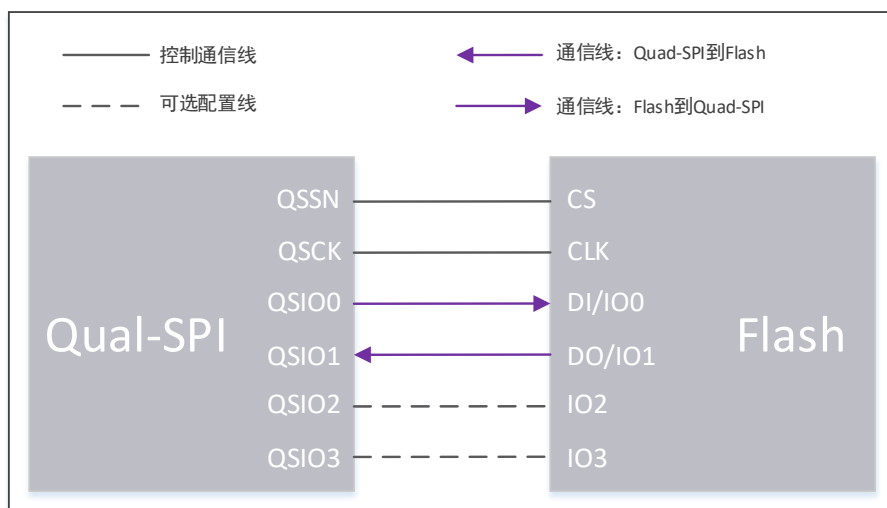
HC32F460 系列的四线式串行外设接口（QSPI）支持灵活的配置目标地址及虚拟周期数，其中目标地址的宽度通过 QSFCR 寄存器中的 AWSL[1:0]位来设置，虚拟周期通过 QSFCR 寄存器中的 DMCYCN[3:0]位来设置。

3.2.1 扩展式 SPI 协议

扩展式 SPI 协议只用 QSIO0 管脚单线进行指令输出，之后的地址及数据则根据具体的读取模式指令使用单线式/二线式/四线式输出。

若配置为单线式及二线式输出则 QSIO2 为输出状态，输出电平由 QSFCR 寄存器的 WPOL 位决定，初始输出为低电平，QSIO3 也为输出状态，输出高电平。QSIO2 管脚也可用作串行闪存的 WP 功能，QSIO3 管脚也可用作串行闪存 HOLD 或 RESET 功能。

硬件连接示意图如下：

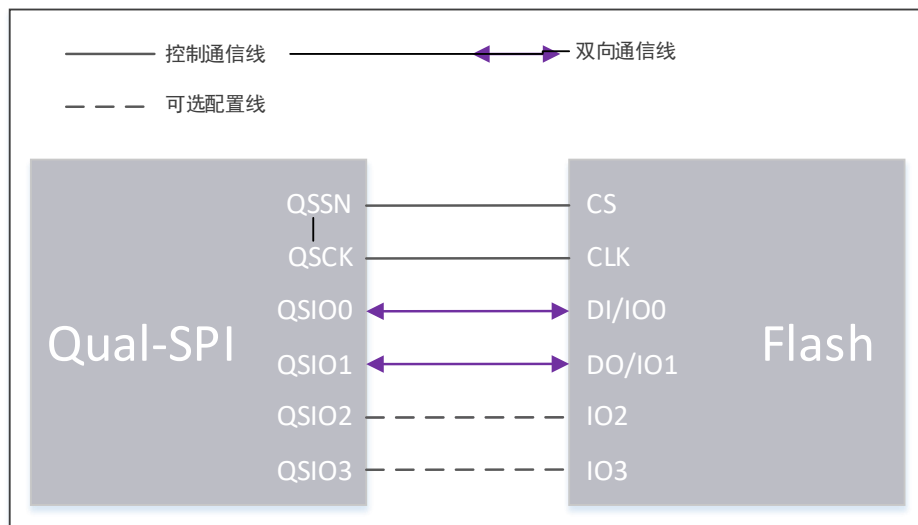


3.2.2 二线式 SPI 协议

二线式 SPI 协议使用 QSIO0, QSIO1 两个管脚实行相应的操作, 包括发出指令, 地址, 接收数据等。

此时 QSIO2 为输出状态, 输出电平由 QSFCR 寄存器的 WPOL 位决定, 初始输出为低电平, QSIO3 也为输出状态, 输出高电平。QSIO2 管脚也可用作串行闪存的 WP 功能, QSIO3 管脚也可用作串行闪存 HOLD 或 RESET 功能。

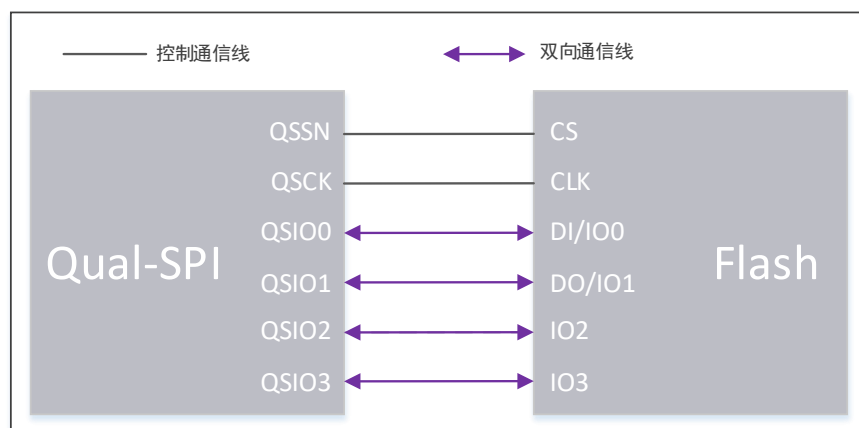
硬件连接示意图如下:



3.2.3 四线式 SPI 协议

四线式 SPI 协议使用 QSIO0, QSIO1, QSIO2, QSIO3 四个管脚实行发出指令, 地址, 接收数据等所有相关操作。

硬件连接示意图如下:



3.3 总线模式

3.3.1 ROM 访问模式

串行闪存及相关的控制寄存器在 AHB 总线空间的位置由总体的地址范围配置来决定，QSPI 可以通过自动将 MCU 的外部 ROM 读取总线周期转换为 QSPI 总线周期来对串行闪存进行读取。此模式下配置好相关参数，读取闪存和读取内置 Flash 一样，不需要操作寄存器，对 QSPI 连接的闪存的映射地址进行直接访问即可。

对于 ROM 的单一读取指令会独立的从芯片内部总线周期一对一的转换为 QSPI 总线周期。当一个 ROM 的读取总线周期被检测到时，QSSN 信号会置为有效状态，从而启动一个 QSPI 总线周期。当接收完串行闪存的数据后，QSSN 信号变成无效状态，该 QSPI 总线周期宣告完成。

3.3.2 直接通信模式

串行闪存还有很多不同的追加功能，诸如 ID 信息读取，擦除，写入及状态信息读取等。针对这种情况，QSPI 提供了直接通信模式，用户可通过软件直接对串行闪存进行控制，由此模式软件可以产生任意所需的 QSPI 总线周期。

将 QSCR 寄存器的 DCOME 位设成 1 可以进入直接通信模式，一旦进入直接通信模式，将无法进行通常的闪存读取操作，如果要进行常规的闪存读取，需要将 DCOME 位清零退出直接通信模式。直接通信模式下一个完整的 QSPI 总线周期从对寄存器 QSDCOM 的 DCOM[7:0]第一次操作开始直到对 QSCR 寄存器的进行一次写操作后结束，这期间 QSSN 信号始终保持低电平的有效状态。

在直接通信模式下是无法对 QSCR 和 QSDCOM 以外的寄存器进行写操作的，对其他寄存器的写操作将会退出直接通信模式。

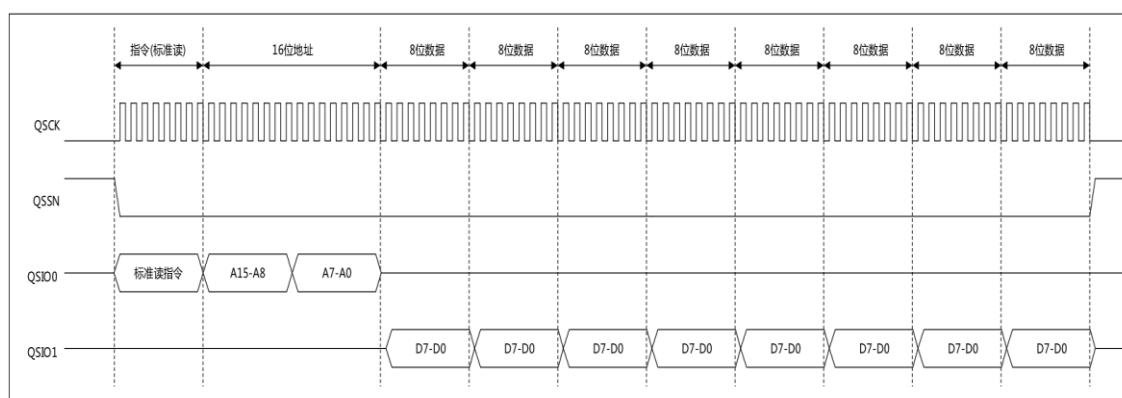
3.4 特殊功能

3.4.1 闪存预读取功能

对于诸如 CPU 指令或是数据块的传输，系统通常是以一个顺次递增的闪存地址顺序进行数据的读取，串行闪存具有连续数据传输能力而不需要再次发送指令代码和地址。QSPI 提供了预读取功能来进行连续的数据接收，通过设置 QSCR 寄存器中的 PFE 位为 1 激活预读取功能，当该功能有效后，数据会被连续接收并储存到缓冲区而不需要等待另一个闪存读取要求，预读取的缓冲区最大可存储 16 个字节的数据，除此以外，还有 2 个字节的数据接收缓冲区也可以存储预读取的数据，当所有的缓冲区数据取满后，QSPI 总线周期结束。

在预读取状态寄存器 QSSR 中，PFAN 位显示了当前的预读取工作状态，PFFUL 位表示预读取数据缓冲区已经放满，而 PFNUM[4:0]则显示了目前已经读取到缓冲区的数据的字节数。

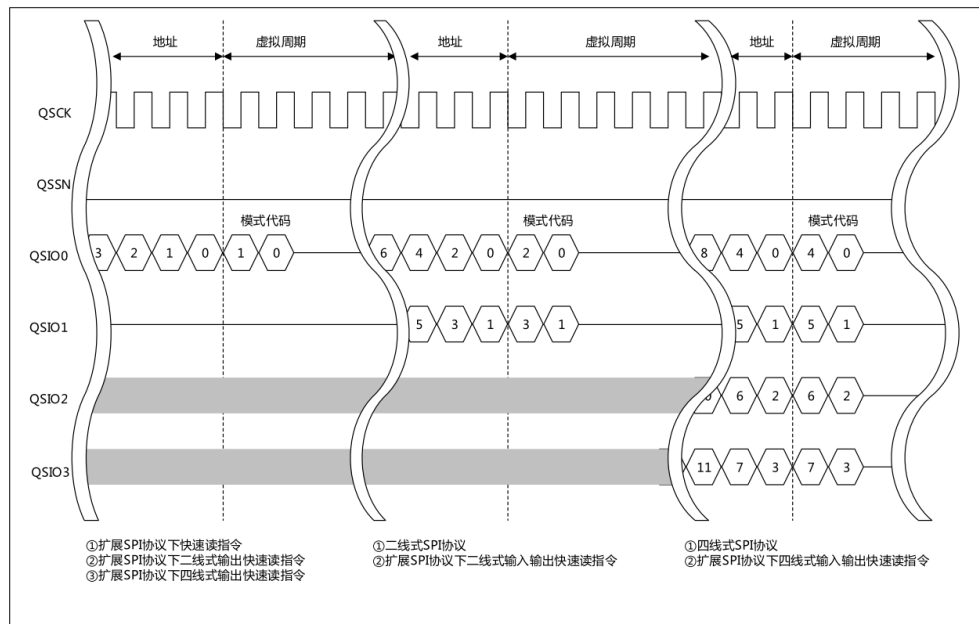
预读取功能有效时数据读取操作示意图：



3.4.2 XIP 模式

一些串行闪存器件可以通过省略接收读取指令来降低延迟时间，该机能可通过虚拟周期期间发送的模式代码在快速度指令时的虚拟周期期间，QSPI 在最初的两个周期通过发送 XIP 模式代码来控制串行闪存的 XIP 模式，可通过寄存器 QSCMD 的 XIPMC[7:0]位进行针对性设置。

XIP 模式控制示意图：



启动串行闪存的 XIP 模式需要在 QSCMD[7:0]中设置相应的模式代码，控制部分的 XIP 模式只需要将 XIPE 位置成 1 就可以，而与 QSCMD[7:0]的值无关。

退出串行闪存的 XIP 模式需要在 QSCMD[7:0]中设置相应的退出模式代码，控制部分的 XIP 模式只需要将 XIPE 位清零就可以，而与 QSCMD[7:0]的值无关。

3.5 寄存器介绍

四线式串行外设接口（QSPI）模块的寄存器如下表所示，若需了解具体细节，请参考用户手册：

寄存器简称	寄存器功能
QSCR	控制寄存器
QSCSCR	片选控制寄存器
QSFCR	格式控制寄存器
QSSR	状态寄存器
QSDCOM	直接通信指令寄存器
QSCCMD	指令代码寄存器
QSXCMD	XIP 模式代码寄存器
QSSR2	标志清除寄存器（只写）
QSEXAR	外部地址寄存器

3.6 注意事项

3.6.1 寄存器的设置顺序

在使用中可以动态的对 QSPI 控制寄存器进行设置或更改，但是不注意寄存器的设置顺序可能会导致 QSPI 总线周期在寄存器还没有完全设置完成时就开始，因此请仔细配置寄存器的设置顺序以避免这类情况的发生。

3.6.2 模块停止信号的设置

QSPI 在系统复位后处于模块停止状态，只有在将模块停止控制寄存器中的 QSPI 模块停止信号清零号才可以对寄存器进行设置。

4 样例代码

4.1 代码介绍

用户可以根据上述的工作流程编写自己的代码来学习验证该模块，也可以直接通过华大半导体的网站下载到 HC32F460 系列 MCU 的设备驱动库（Device Driver Library, DDL）来体验 QSPI 与外部 Flash 通信的优势。

以下部分主要基于 DDL 的 QSPI 模块采用四线式输入输出快速读指令读写 Flash 样例 `qspi_four_wire_io_fast_read` 代码，简要介绍 QSPI 的使用方法：

1) 开启 QSPI 时钟：

```
/* Configuration peripheral clock */
PWC_Fcg2PeriphClockCmd(TIMERA_UNIT1_CLOCK | TIMERA_UNIT2_CLOCK,
Enable);
```

2) 配置 QSPI 功能使用的 IO：

```
/* Configuration QSPI pin */
PORT_SetFunc(QSPCK_PORT, QSPCK_PIN, Func_Qspi, Disable);
PORT_SetFunc(QSNSS_PORT, QSNSS_PIN, Func_Qspi, Disable);
PORT_SetFunc(QSIO0_PORT, QSIO0_PIN, Func_Qspi, Disable);
PORT_SetFunc(QSIO1_PORT, QSIO1_PIN, Func_Qspi, Disable);
PORT_SetFunc(QSIO2_PORT, QSIO2_PIN, Func_Qspi, Disable);
PORT_SetFunc(QSIO3_PORT, QSIO3_PIN, Func_Qspi, Disable);
```

3) 初始化 QSPI：

```
/* Configuration QSPI structure */
stcQspiInit.enClkDiv = QspiHclkDiv3;
stcQspiInit.enSpiMode = QspiSpiMode3;
stcQspiInit.enBusCommMode = QspiBusModeRomAccess;
stcQspiInit.enPrefetchMode = QspiPrefetchStopComplete;
stcQspiInit.enPrefetchFuncEn = Disable;
stcQspiInit.enQssnValidExtendTime = QspiQssnValidExtendSck32;
stcQspiInit.enQssnIntervalTime = QspiQssnIntervalQsck8;
stcQspiInit.enQsckDutyCorr = QspiQsckDutyCorrHalfHclk;
stcQspiInit.enVirtualPeriod = QspiVirtualPeriodQsck6;
stcQspiInit.enWpPinLevel = QspiWpPinOutputHigh;
stcQspiInit.enQssnSetupDelayTime = QspiQssnSetupDelay1Dot5Qsck;
stcQspiInit.enQssnHoldDelayTime = QspiQssnHoldDelay1Dot5Qsck;
stcQspiInit.enFourByteAddrReadEn = Disable;
stcQspiInit.enAddrWidth = QspiAddressByteThree;
stcQspiInit.stcCommProtocol.enReadMode = QspiReadModeFourWiresIO;
stcQspiInit.stcCommProtocol.enTransInstrProtocol = QspiProtocolExtendSpi;
stcQspiInit.stcCommProtocol.enTransAddrProtocol = QspiProtocolExtendSpi;
```

```
stcQspiInit.stcCommProtocol.enReceProtocol = QspiProtocolExtendSpi;  
stcQspiInit.u8RomAccessInstr = QSPI_3BINSTR_FOUR_WIRES_IO_READ;  
QSPI_Init(&stcQspiInit);
```

- 4) 切换 QSPI 为标准读模式进行擦、写操作:

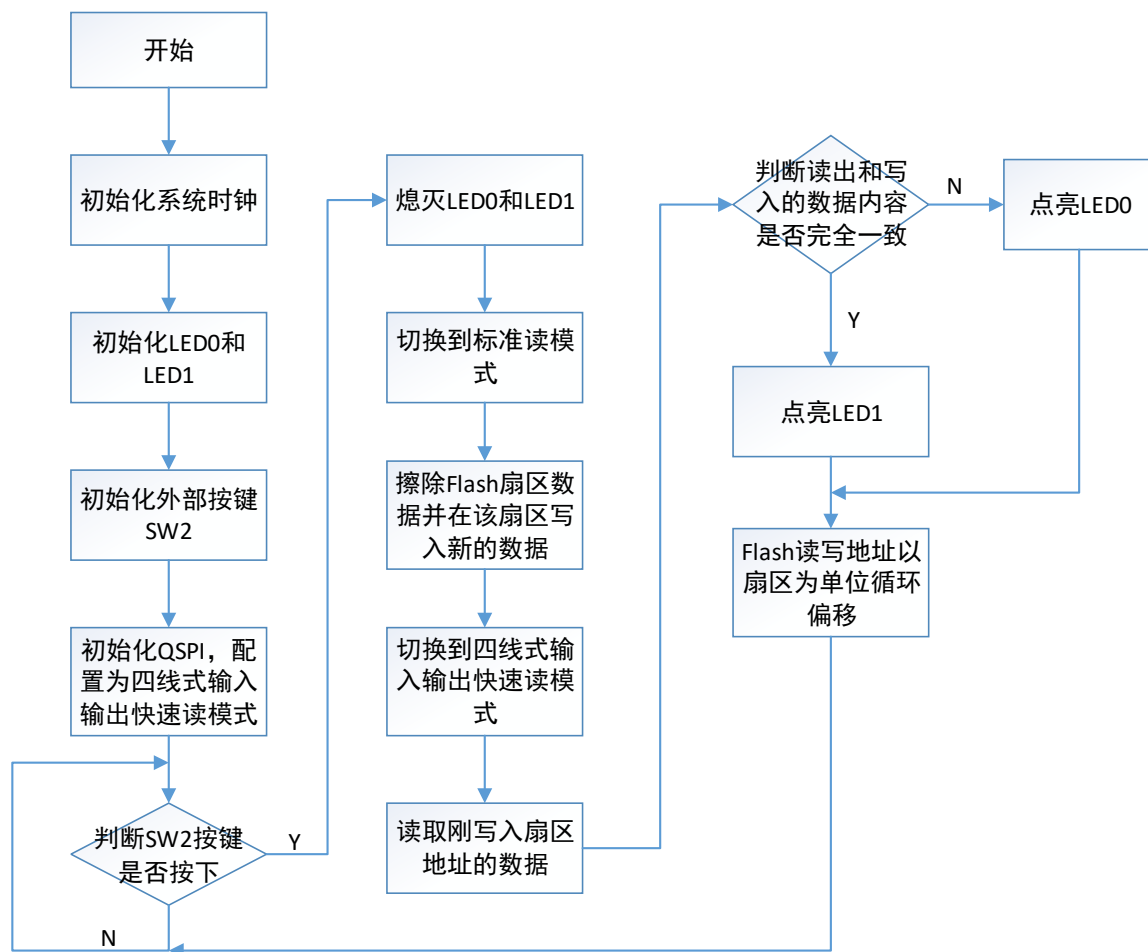
```
/* Switch to standard read mode */  
stcQspiCommProtocol.enReadMode = QspiReadModeStandard;  
QSPI_CommProtocolConfig(&stcQspiCommProtocol);  
/* Erase sector */  
QspiFlash_Erase4KbSector(flashAddr);  
/* Write data to flash */  
QspiFlash_WritePage(flashAddr, &txBuffer[0], bufferLen);
```

- 5) 切换 QSPI 为四线式输入输出快速读模式进行数据读取及对比, 若写入数据和读出数据完全一致则点亮 LED1 灯, 否则点亮 LED0 灯:

```
/* Switch to four wire i/o fast read mode */  
stcQspiCommProtocol.enReadMode = QspiReadModeFourWiresIO;  
QSPI_CommProtocolConfig(&stcQspiCommProtocol);  
/* Pointer to flash address map */  
pFlashReadAddr = (uint8_t *)((uint32_t)QSPI_BUS_ADDRESS + flashAddr);  
/* Compare txBuffer and flash */  
if (memcmp(txBuffer, pFlashReadAddr, bufferLen) != 0)  
{  
    LED0_ON();  
}  
else  
{  
    LED1_ON();  
}
```


4.2 工作流程

样例代码中 QSPI 操作流程如下图所示：

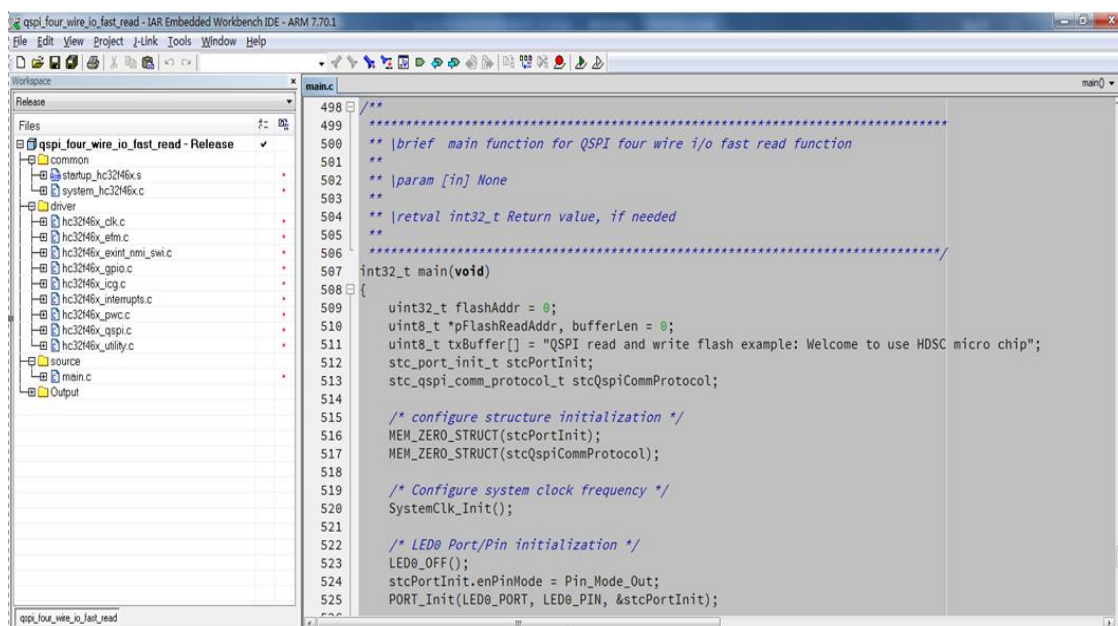




4.3 代码运行

用户可以通过华大半导体的网站下载到 DDL 的样例代码（qspi_four_wire_io_fast_read、qspi_two_wire_io_fast_read、qspi_standard_read、qspi_fast_read），并配合评估用板（比如‘EV-HC32F460-LQFP100-050-V1.1’）运行相关代码学习使用 QSPI 模块。

以下部分主要介绍如何在‘EV-HC32F460-LQFP100-050-V1.1’评估板上，通过 IAR EWARM 编译、运行 qspi_four_wire_io_fast_read 样例代码并观察结果：

- 一 确认安装正确的 IAR EWARM v7.7 工具（请从 IAR 官方网站下载相应的安装包，并参考用户手册进行安装）。
- 一 获取‘EV-HC32F460-LQFP100-050-V1.1’评估板。
- 一 从华大半导体网站下载 HC32F460 DDL 代码。
- 一 下载并运行 qspi\qspi_four_wire_io_fast_read\中的项目文件：
 - 1) 打开 qspi_four_wire_io_fast_read\项目，并打开‘main.c’如下视图：



- 2) 点击  重新编译整个项目；
- 3) 点击  将代码下载到评估板上，全速运行；
- 4) 按下 SW2 触发 flash 擦除、写入、读出后比较功能；
- 5) 观察测试板，比较结果相同则 LED1 亮，不同则 LED0 亮。

5 总结

以上章节简要介绍 HC32F460 系列的 QSPI 寄存器、功能模式、注意事项。演示了如何操作 QSPI 读写 Flash 样例代码，在开发中用户可以根据自己的实际需要使用 QSPI 模块。

6 版本信息 & 联系方式

日期	版本	修改记录
2019/3/15	Rev1.0	初版发布
2020/8/26	Rev1.1	更新支持型号



如果您在购买与使用过程中有任何意见或建议，请随时与我们联系。

Email: mcu@hdsc.com.cn

网址: <http://www.hdsc.com.cn/mcu.htm>

通信地址: 上海市浦东新区中科路 1867 号 A 座 10 层

邮编: 201203

