

32-bit microcontrollers

Four-wire Serial Peripheral Interface for HC 32F 460 Series QSPI

Applicable objects

F Series	HC 32F460
----------	-----------

Table of Contents Table of Contents

1	Abstract	3
2	QSPI Introduction	3
2.1	Main Features.....	3
2.2	Memory Mapping	4
3	QSPI for HC32F460 Series.....	5
3.1	Frame format.....	5
3.1.1	Instruction	6
3.1.2	Address	7
3.1.3	Virtual Cycle	8
3.1.4	Data.....	8
3.2	Communication protocol	9
3.2.1	Extended SPI Protocol.....	9
3.2.2	2-wire SPI protocol	10
3.2.3	Four-wire SPI protocol	10
3.3	Bus mode	11
3.3.1	ROM Access Mode.....	11
3.3.2	Direct communication mode.....	11
3.4	Special Features.....	12
3.4.1	Flash pre-read function.....	12
3.4.2	XIP mode.....	13
3.5	Register Introduction.....	14
3.6	Cautions	14
3.6.1	Setting order of registers.....	14
3.6.2	Setting of module stop signal	14
4	Sample Code	15
4.1	Code Introduction	15
4.2	Workflow.....	17
4.3	Code Run	18
5	Summary	19
6	Version Information & Contact	20

1 Abstract

This application note introduces the HC32F460 family's four-wire serial peripheral interface (QSPI) module and briefly explains how to use it via the

How QSPI quad-wire input/output fast read mode communicates with external Flash.

2 QSPI Introduction

The HC32F460 series' four-wire serial peripheral interface (QSPI) is a memory control module used to communicate with serial ROMs with SPI-compatible interfaces, mainly for serial flash memory, serial EEPROMs, and serial EEPROMs. FeRAM.

2.1 Main Features

- Supports multiple protocols such as extended SPI, 2-wire SPI and 4-wire SPI
- Address line width selectable 8-bit/16-bit/24-bit/32-bit
- Timing can be adjusted to support various serial flash memories
- Support multiple reading methods
 - Standard Read/Quick Read
 - 2-wire output fast readout / 2-wire input and output fast readout
 - Four-wire output fast readout / Four-wire input and output fast readout
- Adjustable number of virtual cycles
- 16-byte pre-read function
- Bus cycle extension function
- XIP control function
- Flexible and extensive support for a wide range of serial flash software control commands, including erase, write, ID read and power-down control

2.2 Memory Mapping

The location of the serial flash memory and associated control registers in the AHB bus space is determined by the overall address range configuration. The QSPI space is distinguished into 2 segments, including 64MB of QSPI I/O register space and 64MB of external QSPI device space, refer to the following table for allocation relationships:

QSPI	0x98000000	0x9FFFFFFF	128MB
QSPI I/O Registers	0x9C000000	0x9FFFFFFF	64MB
External QSPI Devices	0x98000000	0x9BFFFFFF	64MB

Whenever a read access is made to the ROM space of the QSPI, the QSPI bus automatically starts working to transfer the data read from the serial flash memory. The QSPI can read the serial flash memory by automatically converting the MCU's external ROM read bus cycle to a QSPI bus cycle.

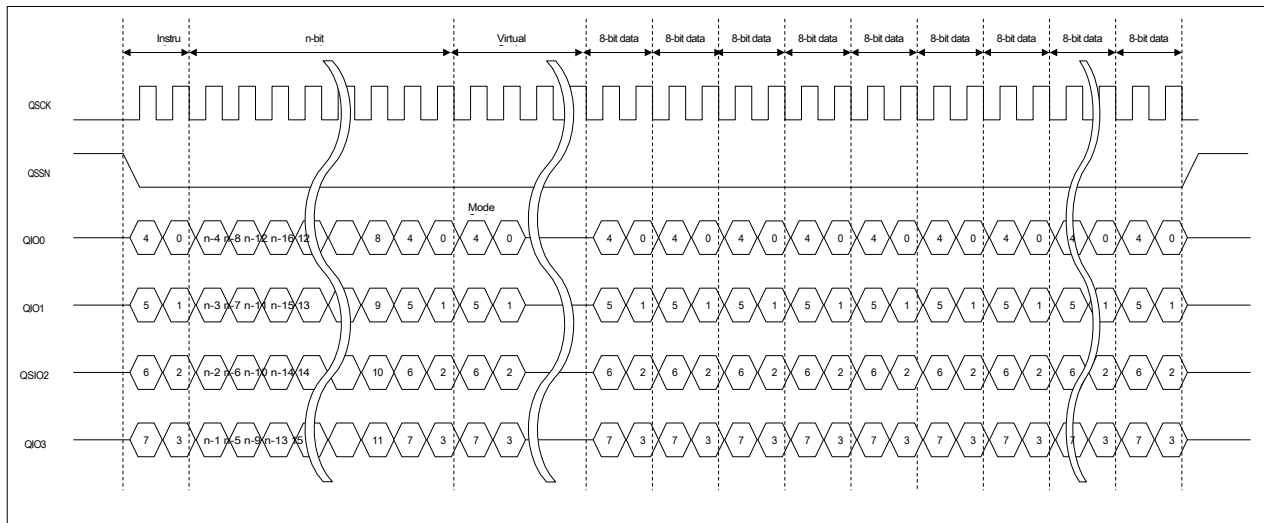
3 QSPI for HC32F460 Series

3.1 Frame format

The four-wire serial peripheral interface (QSPI) of HC32F460 series supports three protocols: extended SPI, two-wire SPI and four-wire SPI. The initial default protocol is the extended SPI protocol, and the protocols for the command send phase, address send phase, and data receive phase can be configured by setting the IPRSL[1:0]/APRSL[1:0]/DPRSL[1:0] bits in the QSCR register, respectively.

The MDSEL in the QSCR register can be used to configure the QSPI read mode. Generally, the recommended mode can be configured for normal communication, but if it is configured as a custom mode, the IPRSL[1:0]/APRSL[1:0]/DPRSL[1:0] bits in the QSCR register must be configured for the same protocol to ensure proper QSPI operation. And the direct communication mode does not support multi-line actions.

Four-wire SPI protocol action schematic:



3.1.1 Instruction

When a serial bus cycle begins, the serial flash select signal is set to active and the QSPI begins to output the instruction code, which is an 8-bits of data that can send any valid instruction value. The instruction code needs to be configured before the serial bus cycle begins, via the QSCCMD register.

The command phases are configured in the following table:

Register Configuration		ROM Access Mode	Direct communication mode	Command Format
QSCR [9:8]	Expanded SPI	IPRSL[1:0] = 00	IPRSL[1:0]=00	
	Two-wire SPI	IPRSL[1:0] = 01	Not supported	
	Four-wire SPI	IPRSL[1:0] = 10	Not supported	

3.1.2 Address

At this stage, an address is sent to the flash memory. QSPI has a 32-bit address bus width to match the serial flash memory and can be selected to use 8-bit/16-bit/24-bit/32-bit as the address bus width by setting AWSL[1:0] in the QSFCE register. If the 8-bit/16-bit/24-bit address bus width is selected, only the lower space with the matching address can be accessed normally, and accessing the serial flash mirror space of the higher bits in the QSPI will repeatedly result in the contents of the lower space.

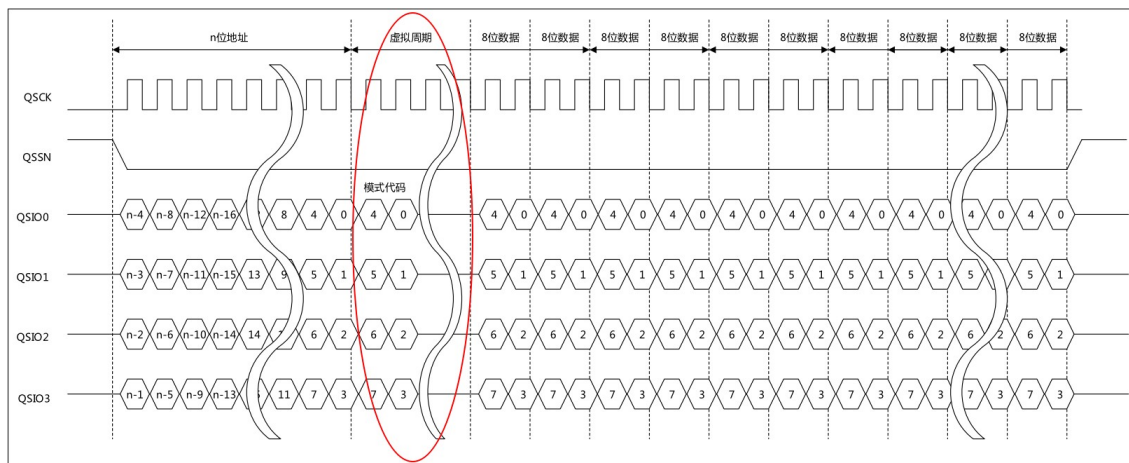
The address phase is configured in the following table:

Register Configuration		ROM Access Mode	Direct communication mode
QSFCE[1:0]	1 byte	AWSL[1:0] = 00	AWSL[1:0] = 00
	2 bytes	AWSL[1:0] = 01	AWSL[1:0] = 01
	3 bytes	AWSL[1:0] = 10	AWSL[1:0] = 10
	4 bytes	AWSL[1:0] = 11	AWSL[1:0] = 11
QSCRE [11:10]	Expanded SPI	APRSL[1:0] = 00	APRSL[1:0] = 00
	Two-wire SPI	APRSL[1:0] = 01	Not supported
	Four-wire SPI	APRSL[1:0] = 10	Not supported

3.1.3 Virtual Cycle

In the case of fast read instructions, a certain number of virtual cycles are added after the transmit address, as determined by DMCYCN[3:0] in the QSFCR register. The first two virtual cycles are used to determine whether XIP mode is selected.

The virtual cycle phase is configured as follows:



3.1.4 Data

During this phase, data is sent to or received from the QSPI flash memory, and a complete QSPI bus cycle in direct communication mode begins with the first operation on DCOM[7:0] of register QSDCOM and ends with a write operation to the QSCR register. A write to DCOM[7:0] converts to a single byte of data transfer on the QSPI bus, while a read to DCOM[7:0] converts to a single byte of data reception on the QSPI bus.

The data phases are configured in the following table:

Register Configuration		ROM Access Mode	Direct communication mode
Data	Read data	Direct access to memory mapped addresses	QSDCOM
	Write data	Not supported	QSDCOM
QSCR [13:12]	Expanded SPI	APRSL[1:0] = 00	APRSL[1:0] = 00
	Two-wire SPI	APRSL[1:0] = 01	Not supported
	Four-wire	APRSL[1:0] = 10	Not supported

	SPI		
--	-----	--	--

3.2 Communication protocols

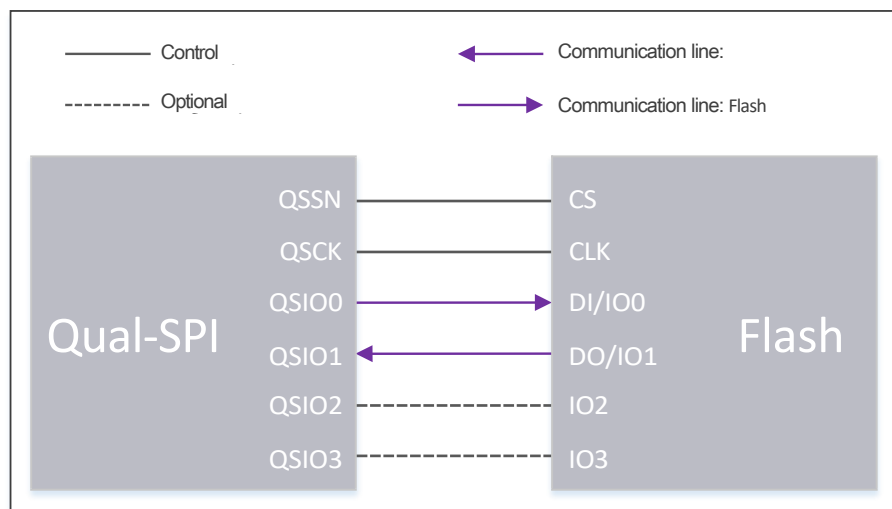
The HC32F460 family's four-wire serial peripheral interface (QSPI) supports flexible configuration of the target address and the number of virtual cycles, where the width of the target address is set by the AWSL[1:0] bits in the QSFCCR register and the virtual cycle is set by the DMCYCN[3:0] bits in the QSFCCR register.

3.2.1 Extended SPI Protocol

The extended SPI protocol uses only a single line of the QSIO0 pin for command output, after which the address and data are output using single line / two line / four line depending on the specific read mode command.

The QSIO2 pin can also be used as the WP function of the serial flash, and the QSIO3 pin can also be used as the HOLD or RESET function of the serial flash.

The hardware connection schematic is as follows:

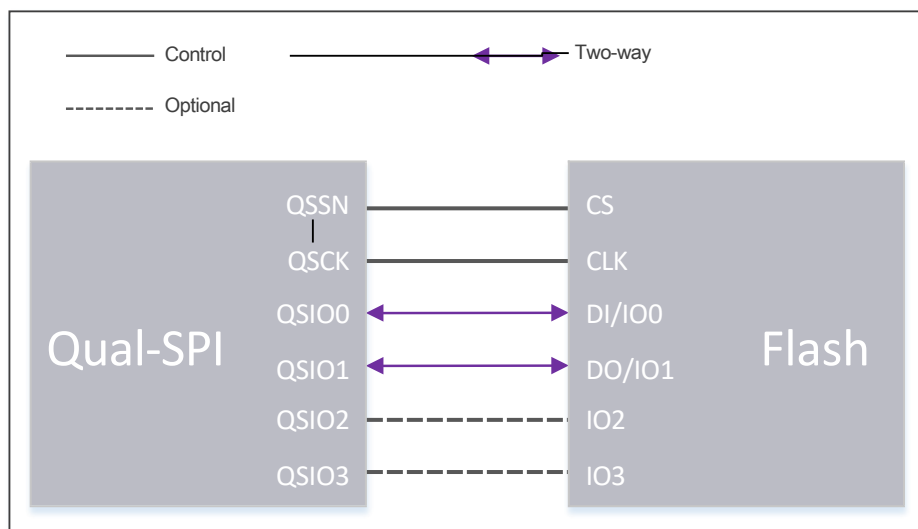


3.2.2 Two-wire SPI Protocol

The two-wire SPI protocol uses the QSI00 and QSI01 pins to perform the corresponding operations, including issuing commands, addressing, and receiving data.

QSI02 is the output state and the output level is determined by the WPOL bit in the QSFCR register, the initial output is low, QSI03 is also the output state and the output is high, QSI02 pin can also be used as the WP function of the serial flash, QSI03 pin can also be used as the HOLD or RESET function of the serial flash.

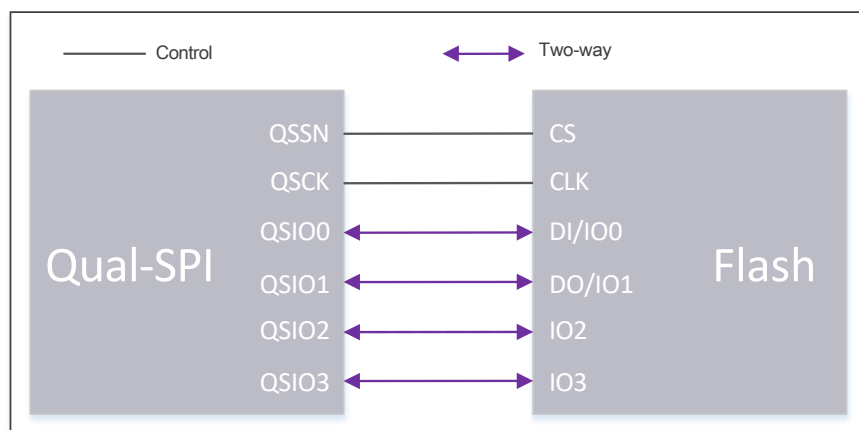
The hardware connection schematic is as follows:



3.2.3 Four-wire SPI Protocol

The four-wire SPI protocol uses the four pins QSI00, QSI01, QSI02, and QSI03 to perform all related operations such as issuing commands, addressing, and receiving data.

The hardware connection schematic is as follows:



3.3 Bus mode

3.3.1 ROM Access Mode

The location of the serial flash and associated control registers in the AHB bus space is determined by the overall address range configuration, and QSPI can read the serial flash by automatically converting the MCU's external ROM read bus cycle to a QSPI bus cycle.

Fetch. In this mode, after configuring the relevant parameters, reading the flash memory is the same as reading the internal flash, no register manipulation is required, and the QSPI Direct access to the mapped address of the connected flash memory is sufficient.

A single read instruction for a ROM is independently converted one-to-one from the chip's internal bus cycle to a QSPI bus cycle. When a ROM read bus cycle is detected, the QSSN signal is set to the active state, initiating a QSPI bus cycle. When the data from the serial flash is received, the QSSN signal becomes invalid and the QSPI bus cycle is declared complete.

3.3.2 Direct communication mode

Serial Flash has many different additional functions, such as ID reading, erasing, writing, and status reading. To address this, QSPI provides a direct communication mode that allows the user to control the serial flash directly through software, which can generate as many QSPI bus cycles as desired.

Once in direct communication mode, normal flash reads are not possible. To perform regular flash reads, clear the DCOME bits to exit direct communication mode. A complete QSPI bus cycle in direct communication mode starts with the first operation on DCOM[7:0] of register QSDCOM and ends with a write operation on the QSCR register, during which the QSSN signal remains active low.

Writes to registers other than QSCR and QSDCOM are not possible in direct communication mode, and writes to other registers will exit direct communication mode.

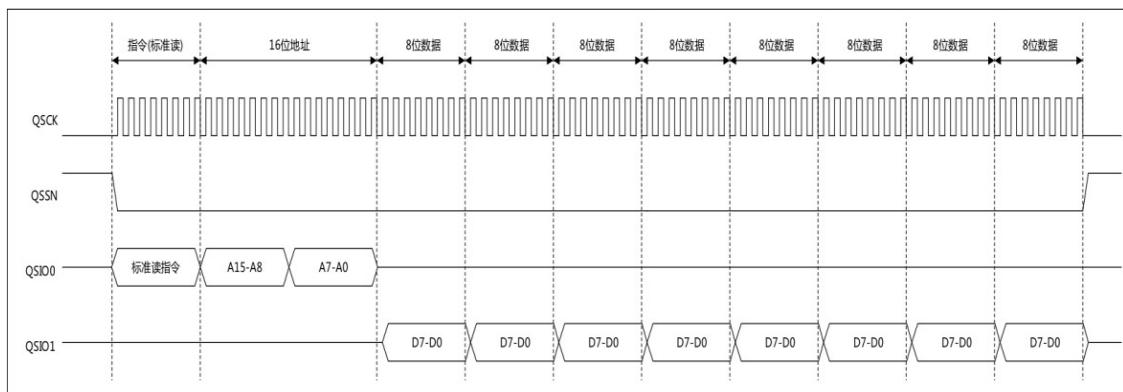
3.4 Special Features

3.4.1 Flash pre-read function

For transfers such as CPU instructions or data blocks, the system typically reads data in a sequential, incremental sequence of flash addresses, and serial flash has the ability to transfer data continuously without sending instruction codes and addresses again. When this function is active, data is continuously received and stored in the buffer without waiting for another flash read request. The pre-read buffer can store up to 16 bytes of data, in addition to the 2-byte data receive buffer that can also store the pre-read data.

In the Pre-Read Status Register QSSR, the PFAN bits show the current pre-reading operation status, the PFFUL bits indicate that the pre-reading data buffer is full, and PFNUM[4:0] shows the number of bytes of data that have been read into the buffer so far.

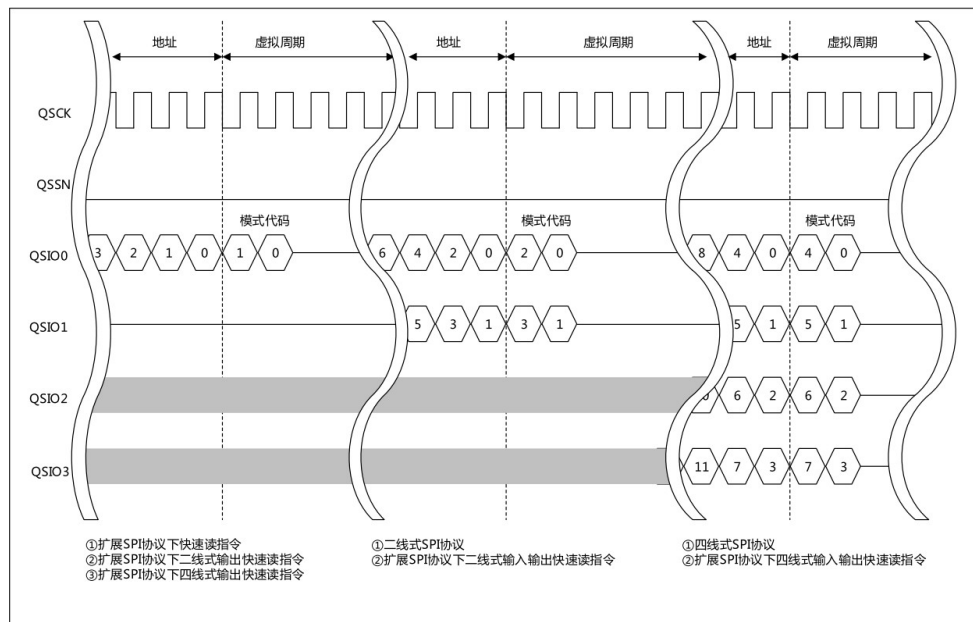
Schematic diagram of the data reading operation when the pre-read function is active:



3.4.2 XIP mode

Some serial flash devices can reduce latency by omitting the receive read instruction. This capability can be achieved by sending the mode generation during the virtual cycle during the virtual cycle for fast speed instructions. The QSPI controls the XIP mode of the serial flash by sending the XIP mode code during the first two cycles, which can be targeted through the XIPMC[7:0] bits of register QSXCMD.

XIP mode control schematic:



Starting the XIP mode of the serial flash requires setting the corresponding mode code in QSXCMD[7:0]. The XIP mode of the control section only requires setting the XIPE position to 1, independent of the value of QSXCMD[7:0].

Exiting the XIP mode of the serial flash requires setting the appropriate exit mode code in QSXCMD[7:0], the XIP of the control section mode only requires clearing the XIPE bits to zero, independent of the value of QSXCMD[7:0].

3.5 Register Introduction

The registers of the Quad Serial Peripheral Interface (QSPI) module are shown in the following table, for more details, please refer to the user manual:

Register abbreviation	Register Function
QSCR	Control register
QSCSCR	Chip Select Control Register
QSFCR	Format Control Register
QSSR	Status Register
QSDCOM	Direct communication command register
QSCCMD	Instruction code register
QSXCMD	XIP mode code register
QSSR2	Flag clear register (write only)
QSEXAR	External address register

3.6 Cautions

3.6.1 Order of register settings

The QSPI control registers can be dynamically set or changed in use, but not paying attention to the order in which the registers are set may cause the QSPI bus cycle to start before the registers are fully set, so please configure the register setting order carefully to avoid such situations.

3.6.2 Setting of module stop signal

The QSPI is in the module stop state after a system reset, and the register can only be set if the QSPI module stop signal in the module stop control register is cleared to zero.

4 Sample Code

4.1 Code Introduction

Users can write their own code according to the above workflow to learn and verify the module, or they can download the Device Driver Library (DDL) of HC32F460 series MCUs directly from the website of UW Semiconductors to experience the advantages of QSPI communication with external Flash.

The following section is based on the DDL's QSPI module using four-wire input and output fast read instructions to read and write Flash samples

qspi_four_wire_io_fast_read code, which briefly describes the use of QSPI:

- 1) Enables QSPI clocking:

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

- 2) Configure the IOs used by the QSPI function:

```
/* 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) Initializing QSPI:

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



```
stcQspiInit.stcCommProtocol.enReceProtocol = QspiProtocolExtendSpi;
stcQspiInit.u8RomAccessInstr = QSPI_3BINSTR_FOUR_WIRES_IO_READ; QSPI
_Init(&stcQspi Init).
```

- 4) Switching QSPI to standard read mode for erase and write operations:

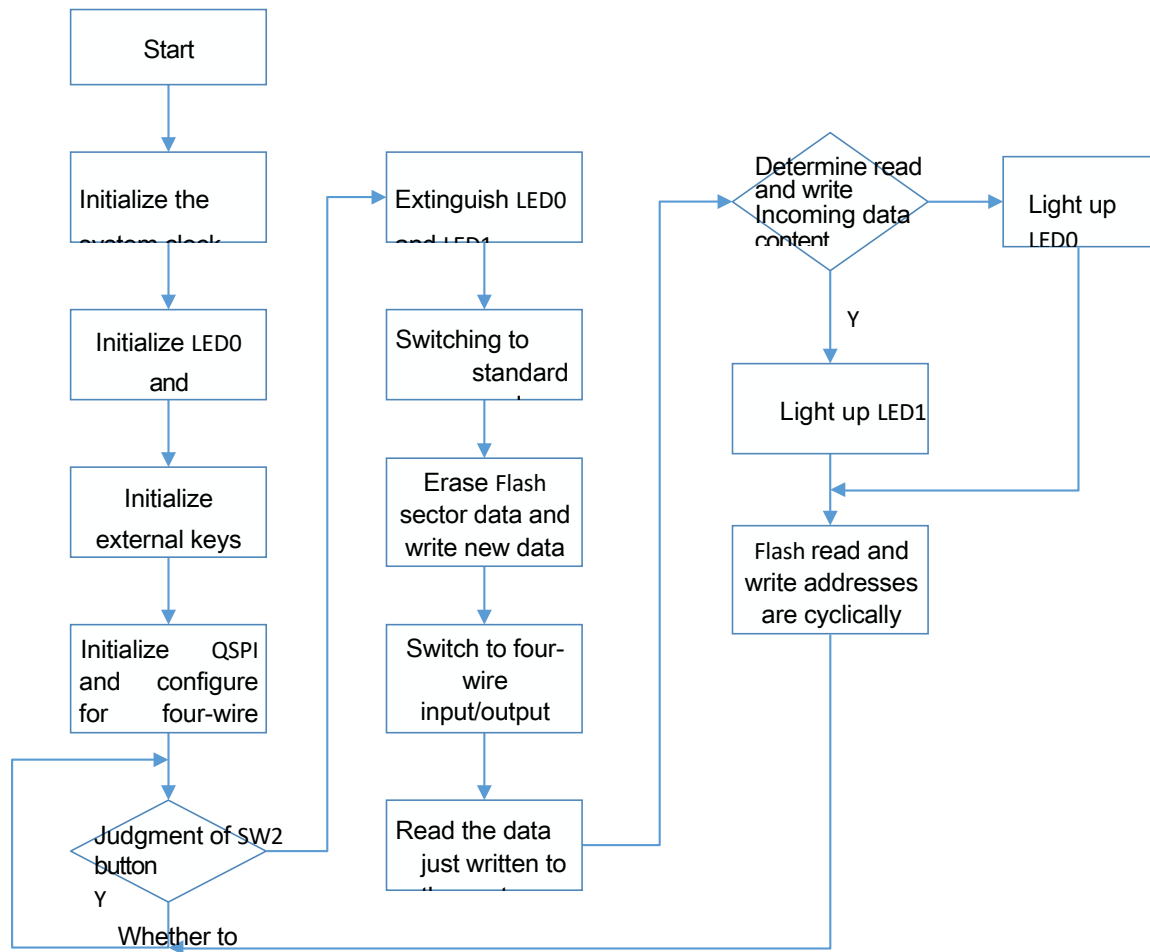
```
/* Switch to standard read mode */
stcQspiCommProtocol.enReadMode = QspiReadModeStandard;
QSPI_CommProtocolConfig(&stcQspiCommProtocol).
/* Erase sector */
QspiFlash_Erase4KbSector(flashAddr).
/* Write data to flash */
QspiFlash_WritePage(flashAddr, &txBuffer[0], bufferLen).
```

- 5) Switch QSPI to four-wire input-output fast read mode for data reading and comparison, if the written data and the read data are identical, LED1 is lit, otherwise LED0 is lit:

```
/* 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 Workflow

The QSPI operation flow in the sample code is shown in the following diagram:

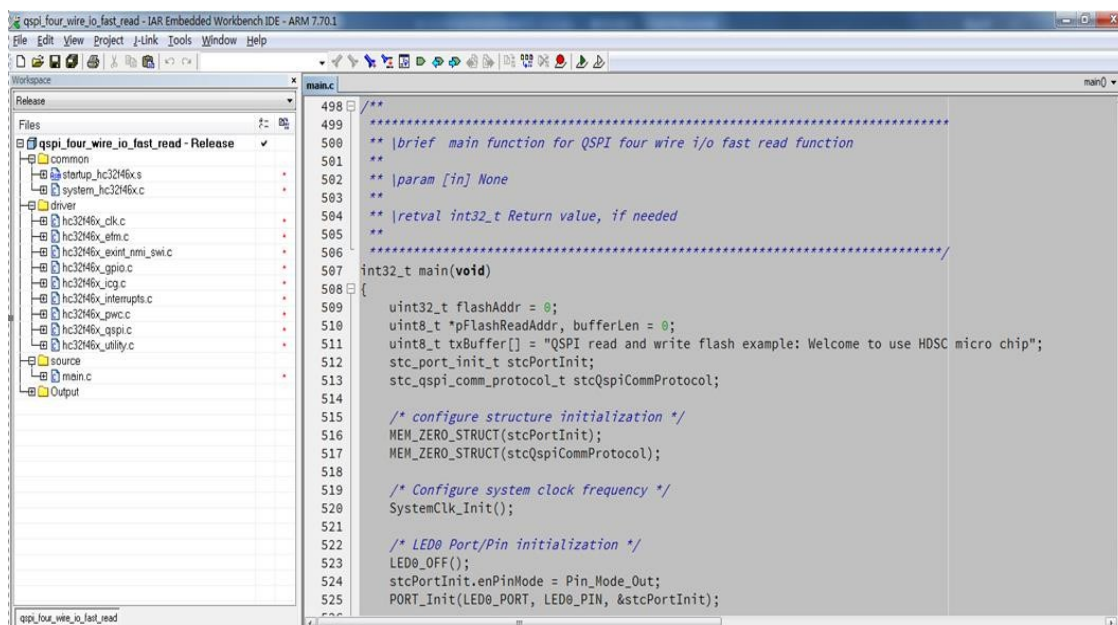




4.3 Code Run

Users can download sample DDL code (qspi_four_wire_io_fast_read, qspi_two_wire_io_fast_read, qspi_standard_read, qspi_fast_read) with evaluation boards (e.g. 'EV-HC32F460-LQFP100-050-V1.1') to run the relevant code to learn to use the QSPI module.

The following section describes how to compile and run the qspi_four_wire_io_fast_read sample code on the 'EV-HC32F460-LQFP100-050-V1.1' evaluation board using IAR EWARM and observe the results:

- Verify that the correct IAR EWARM v7.7 tool is installed (please download the appropriate installation package from the official IAR website and refer to the user manual for installation).
 - Get the 'EV-HC32F460-LQFP100-050-V1.1' evaluation board.
 - Download the HC32F460 DDL code from the UW Semiconductors website.
 - Download and run the project file in qspi\qspi_four_wire_io_fast_read \ at
- 1) Open the qspi_four_wire_io_fast_read\ project and open the 'main.c' view as follows:



- 2) Click  Recompile the entire project;
- 3) Click  Download the code to the evaluation board and run it at full speed;
- 4) Press SW2 to trigger flash erase, write, read and compare functions;
- 5) Observe the test board, compare the results are the same then LED1 is on, different then

LED0 is on.

5 Summary

The above section briefly introduces the QSPI registers, function modes, and notes of HC32F460 series. It demonstrates how to operate the

QSPI read and write Flash sample code, users can use QSPI module according to their actual needs in the development.

6 Version Information & Contact

Date	Versions	Modify records
2019/3/15	Rev1.0	Initial Release
2020/8/26	Rev1.1	Update supported models



If you have any comments or suggestions in the process of purchase and use, please feel free to contact us.

Email: mcu@hdsc.com.cn

Website: <http://www.hdsc.com.cn/mcu.htm>

Address: 10/F, Block A, 1867 Zhongke Road, Pudong

New Area, Shanghai, 201203, P.R. China

