RX Family

QSPIX Module Using Firmware Integration Technology

Introduction

This document covers the QSPIX Module Using Firmware Integration Technology (FIT) for the supported RX family MCUs. Details are provided that describe the QSPIX driver's architecture, integration of the FIT module into a user's application, and how to use the API.

The RX family MCUs supported by this module have a built-in Quad Serial Peripheral Interface (QSPIX) for only one channel. The Quad-SPI Memory Interface (QSPIX) module is a memory controller for connecting serial ROM that has an SPI compatible interface. This includes nonvolatile memory, such as a serial flash memory, serial EEPROM, or serial FeRAM.

Target Device

The following is a list of devices that are currently supported by this API:

- RX671 Group

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Target Compilers

- Renesas Electronics C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX

For details of the confirmed operation contents of each compiler, refer to "6.1 Operation Confirmation Environment".

Related Documents

- Firmware Integration Technology User's Manual (R01AN1833)
- RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685)

Contents

1. Overview

The QSPIX FIT module can be combined with other FIT modules for easy integration into the target system.

The functions of the QSPIX FIT module can be incorporated into software programs by means of APIs.

It is recommended to review the QSPIX peripheral chapter in the RX MCU hardware user's manual before using this software.

1.1 QSPIX FIT Module

The QSPIX FIT module can be used by being implemented in a project as an API. See section 2.13, Adding the FIT Module to Your Project for details on methods to implement this FIT module into a project.

1.2 Overview of the QSPIX FIT Module

After adding the QSPIX FIT module to your project you will need to modify the r_qspix_rx_config.h file to configure the software for your installation. See Section 2.8 Configuration Overview for details on configuration options.

The QSPIX FIT module does not have a function to initialize a register of the I/O port. The setting of the I/O port must be accomplished other than this module. See Section 4, Pin Setting for setting of the I/O port.

The functions of the module are described briefly below.

Table 1-1 QSPIX Specifications

Parameter	Specifications
Number of channels	1 channel
SPI	• Support for Extended SPI, Dual SPI, and Quad SPI protocols • Configurable to SPI mode 0 and SPI mode 3 • Address width selectable to 8, 16, 24, or 32 bits
Timing adjustment function	Configurable to support a wide range of serial flash
Memory mapped mode	• Support for Read, Fast Read, Fast Read Dual Output, Fast Read Dual I/O, Fast Read Quad Output, and Fast Read Quad I/O instructions • Substitutable instruction code • Adjustable number of dummy cycles • Prefetch function • Polling processing • SPI bus cycle extension function
Indirect access mode	Flexible support for a wide variety of serial flash instructions and functions through software control, including erase, write, ID read, and power-down control
Interrupt source	Error interrupts

1.3 Using the FIT QSPIX module

Using FIT QSPIX module in C++ project

For C++ project, add FIT QSPIX module interface header file within extern “C”{}:

```
Extern “C”
{
  #include “r_smc_entry.h”
  #include “r_qspix_rx_if.h”
}
```

1.4 API Overview

Table 1-2 API Functions lists the API functions included in this module. Also, section 2.9, Code Size, lists the size of the code sections used by this module.

Table 1-2 API Functions

Function	Function Description
R_QSPIX_Open()	This function applies power to the QSPIX channel, initializes the associated registers, enables interrupts, and provides the channel handle for use with other API functions.
R_QSPIX_Control()	This function is used to change settings.
R_QSPIX_Write_Indirect()	Writes raw data directly to the QSPIX.
R_QSPIX_Read_Indirect()	Reads raw data directly from the QSPIX. This API can only be called after R_QSPIX_Write_indirect() with read_after_write set to true.
R_QSPIX_Enter_XIP()	Enters XIP(execute in place) mode.
R_QSPIX_Exit_XIP()	Exits XIP(execute in place) mode.
R_QSPIX_BankSet()	Selects the bank to access.
R_QSPIX_Set_Spi_Protocol()	Selects the protocol to access.
R_QSPIX_Get_Status()	Get status of Prefetch, busy flag, ROM access error flag.
R_QSPIX_Cloes()	Close the QSPIX driver module.
R_QSPIX_GetVersion()	This function returns the version number of the API.

1.5 Hardware Settings

1.5.1 Hardware Configuration Example

Figure 1-1 is a connection diagram. To achieve high-speed operation, consider adding damping resistors or capacitors to improve the circuit matching of the various signal lines.
Do not fail to perform pull-up processing. Without pull-up processing, the slave device may enter the write protect state or hold state when a data line is in the Hi-Z state.

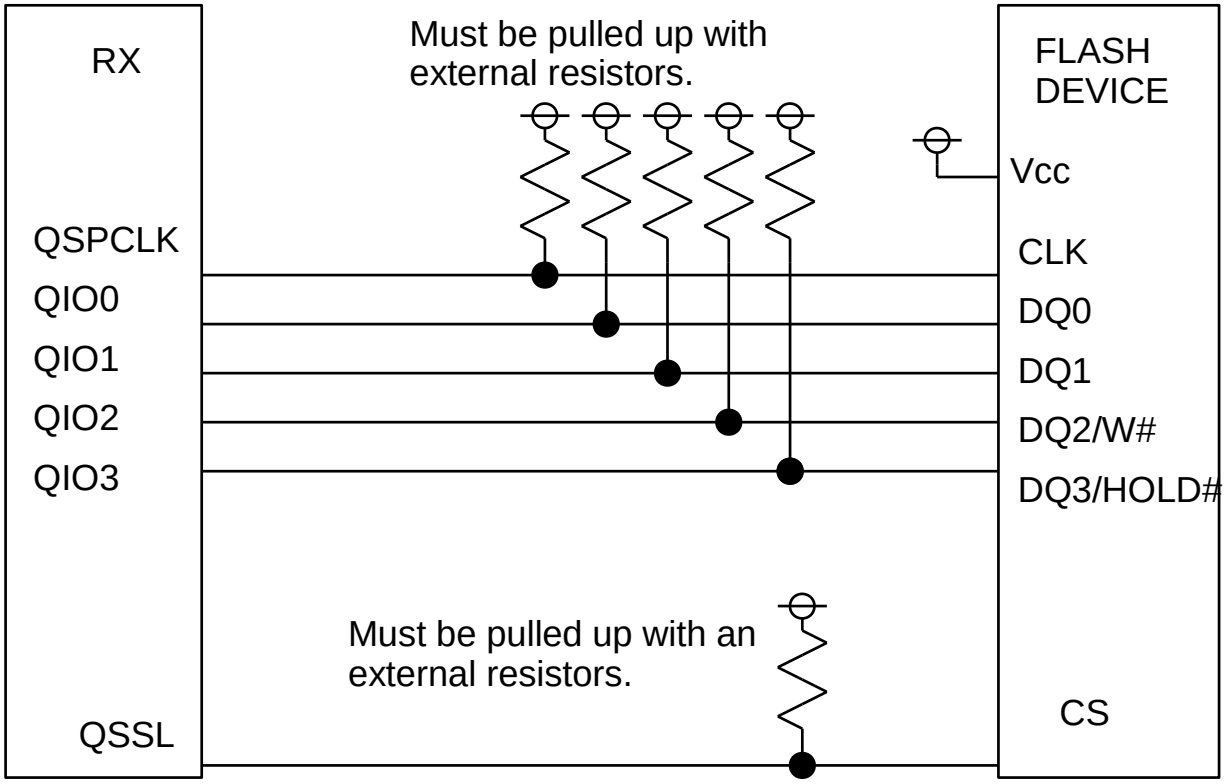

Figure 1.1 : Sample Wiring Diagram for a RX Family MCU QSPI and a Flash Device

1.5.2 List of Pins

Table 1-3 lists the pins that are used and their uses.

Pin Name	I/O	Description
QSPCLK	Output	QSPIX clock output pin
QSSL	Output	QSPIX slave select pin
QIO0	I/O	Data 0 input/output
QIO1	I/O	Data 1 input/output
QIO2	I/O	Data 2 input/output
QIO3	I/O	Data 3 input/output

2. API Information

This Driver API follows the Renesas API naming standards.

2.1 Hardware Requirements

The microcontroller used must support the following functionality.

- QSPIX

2.2 Software Requirements

This driver is dependent on the following packages.

- Renesas Board Support Package (r_bsp) v6.10 or higher

2.3 Limitations

2.3.1 RAM Location Limitations

In FIT, if a value equivalent to NULL is set as the pointer argument of an API function, error might be returned due to parameter check. Therefore, do not pass a NULL equivalent value as pointer argument to an API function.

The NULL value is defined as 0 because of the library function specifications. Therefore, the above phenomenon would occur when the variable or function passed to the API function pointer argument is located at the start address of RAM (address 0x0). In this case, change the section settings or prepare a dummy variable at the top of the RAM so that the variable or function passed to the API function pointer argument is not located at address 0x0.

In the case of the CCRX project (e2 studio V21.1.0), the RAM start address is set as 0x4 to prevent the variable from being located at address 0x0. In the case of the GCC project (e2 studio V21.1.0) and IAR project (EWRX V4.20.1), the start address of RAM is 0x0, so the above measures are necessary.

The default settings of the section may be changed due to the IDE version upgrade. Please check the section settings when using the latest IDE.

2.4 Supported Toolchain

The operation of QSPIX FIT module has been confirmed with the toolchain listed in “6.1 Operation Confirmation Environment”.

2.5 Interrupt Vector

The QSPIX (ROM access error) is enabled by executing the R_QSPIX_Open function (while the macro definition QSPIX_CFG_ERI_INCLUDED is 1.

Table 2.1 lists the interrupt vectors used in the FIT Module.

Table 2-1 Interrupt Vector

Device	Interrupt Vector
RX671	GROUPAL0 interrupt (vector no.: 30)

2.6 Header Files

All the API calls and interface definitions used are listed in r_qspix_rx_if.h.
Configuration options for individual builds are selected in r_qspix_rx_config.h.

2.7 Integer Types

This project uses ANSI C99. These types are defined in stdint.h.

2.8 Configuration Overview

The configuration option settings of this module are located in r_qspix _rx_config.h. The option names and setting values are listed in the table below:

Configuration options in r_qspix_rx_config.h	
Definition	**Description**
QSPIX_CFG_PARAM_CHECKING_ENABLE (1)	1: Parameter checking is included in the build. 0: Parameter checking is omitted from the build.
QSPIX_CFG_USE_CH0 (1)	Enable the QSPIX channels. (0) = not used. (1) = used.
QSPIX_CFG_ERI_INCLUDED (1)	1: ROM access error is included in the build. 0: ROM access error is omitted in the build.
QSPIX_CFG_ERI_IR_PRIORITY (3)	Sets the ROM access error interrupt priority for the channel. The value selected by a value from 1 to 15 is set as the interrupt level.

2.9 Code Size

The sizes of ROM, RAM, and maximum stack usage associated with this module are listed below.

The ROM (code and constants) and RAM (global data) sizes are determined by the build-time configuration options described in 2.8 Configuration Overview

The values in the table below are confirmed under the following conditions.

Module Revision: r_qspix_rx rev1.00

Compiler Version: Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00

(The option of “-lang = c99” is added to the default settings of the integrated development environment.)

GCC for Renesas RX 8.3.0.202004

(The option of “-std = gnu99” is added to the default settings of the integrated development environment.)

IAR C/C++ Compiler for Renesas RX version 4.20.3

(The default settings of the integrated development environment)

Configuration Options: Default settings

ROM, RAM and Stack Code Sizes							
Device	Category	Memory Used					
		Renesas Compiler		GCC		IAR Compiler	
		With Parameter Checking	Without Parameter Checking	With Parameter Checking	Without Parameter Checking	With Parameter Checking	Without Parameter Checking
RX671	ROM	**1609 bytes**	**1291 bytes**	**3360 bytes**	**2944 bytes**	**2857 bytes**	**2378 bytes**
	RAM	**72 bytes**		**72 bytes**		**4 bytes**	
	STACK	**16 bytes**		**-**		**100 bytes**	

2.10 Arguments

The structure for the arguments of the API functions is shown below. This structure is listed in r_qspix_ rx_if.h, along with the prototype declarations of the API functions.

```
typedef struct st_qspix_cfg
{
    /* Read Instruction Select */
    qspix_read_mode_t                       read_mode;

    /* Slave Select Extension */
    qspix_slave_select_extension_t          slave_select_extension;

    /* Slave Select Extension */
    qspix_prefetch_function_t               prefetch_function;

    /* Clock Mode Select */
    qspix_clock_mode_t                      clock_mode;

    /* Data Output Duration Extension */
    qspix_data_output_select_t              data_output_select;

    /* Special Read Instruction Select */
    qspix_special_instruction_select_t      special_instruction_select;

    /* Slave Select High Width Setting */
    qspix_slave_select_high_width_t         slave_select_high_width;

    /* Slave Select Hold Time Setting */
    qspix_slave_select_hold_time_t          slave_select_hold_time;

    /* Slave Select Setup Time Setting */
    qspix_slave_select_setup_time_t         slave_select_setup_time;

    /* Clock Divisor Select */
    qspix_clock_divisor_t                   clock_divisor;

    /* Special Read Instruction Setting used when
       special_instruction_select = QSPIX_INSTRUCTION_CODE_IN_SPRIR_REGISTER */
    uint32_t qspix_special_read_instruction;

    /* Address Size Setting */
    qspix_address_size_t                    address_size;

    /* Instruction with 4-Byte Address Enable */
```

```
    qspix_instruction_4_Byte_address_t      instruction_4_Byte_address;

    /* Number of Dummy Cycle */
    qspix_dummy_clocks_t                    dummy_clocks;

    /* SPI protocol */
    qspix_protocol_t                        protocol;

    /* WP Pin Control */
    qspix_WP_pin_control_t                  WP_pin_control;

    void (*p_callback)(void *p_cbdat); /* pointer to user callback function. */

} qspix_cfg_t;
```

2.11 Return Values

The API function return values are shown below. This enumerated type is listed in r_qspix_rx_if.h, along with the prototype declarations of the API functions.

```
typedef enum e_qspix_err       /* QSPIX API error codes */
{
    QSPIX_SUCCESS,             /* QSPIX processing completed without problem. */
    QSPIX_ERR_OPENED,             /* QSPIX was initialized already. */
    QSPIX_ERR_NOT_OPEN,           /* QSPIX module is not initialized yet. */
    QSPIX_ERR_INVALID_ARG,        /* Arguments are invalid. */
    QSPIX_ERR_INVALID_COMMAND,    /* Command parameters are invalid. Or, forced
data change failed. */
    QSPIX_ERR_NULL_PTR,        /* Argument pointers are NULL. */
    QSPIX_ERR_BUSY,     /* The QSPIX resources are locked by another process. */
    QSPIX_ERR_HW                  /* HW error. */
} qspix_err_t;
```

2.12 Callback Functions

In this module, the callback function specified by the user is called when the ROM access error interrupt occurs.

The callback function is specified by storing the address of the user function in the “void (*p_callback)(void *p_cbdat)” structure member (see 2.10, Arguments).

2.13 Adding the FIT Module to Your Project

This module must be added to each project in which it is used. Renesas recommends the method using the Smart Configurator described in (1) or (3) below. However, the Smart Configurator only supports some RX devices. Please use the methods of (2) or (4) for RX devices that are not supported by the Smart Configurator.

(1) Adding the FIT module to your project using the Smart Configurator in e^2 studio
By using the Smart Configurator in e^2 studio, the FIT module is automatically added to your project. Refer to “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)” for details.

(2) Adding the FIT module to your project using the FIT Configurator in e^2 studio
By using the FIT Configurator in e^2 studio, the FIT module is automatically added to your project. Refer to “Adding Firmware Integration Technology Modules to Projects (R01AN1723)” for details.

(3) Adding the FIT module to your project using the Smart Configurator in CS+
By using the Smart Configurator Standalone version in CS+, the FIT module is automatically added to your project. Refer to “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)” for details.

(4) Adding the FIT module to your project in CS+
In CS+, please manually add the FIT module to your project. Refer to “Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)” for details.

3. API Functions

R_QSPIX_Open()

This function applies power to the QSPIX channel, initializes the associated registers, enables interrupts, and provides the channel handle for use with other API functions.

Format

```
qspix_err_t  R_QSPIX_Open (uint8_t channel, qspix_cfg_t * p_cfg)
```

Parameters

channel

Number of the QSPIX channel to be initialized.

**p_cfg*

Pointer to QSPIX channel configuration data structure.

Return Values

QSPIX_SUCCESS	/* QSPIX processing completed without problem. */
QSPIX_ERR_OPENED	/* QSPIX was initialized already. */
QSPIX_ERR_INVALID_ARG	/* Arguments are invalid. */
QSPIX_ERR_NULL_PTR	/* Argument pointers are NULL. */
QSPIX_ERR_BUSY	/* The QSPIX resources are locked by another process. */

Properties

The declaration is located in r_qspix_rx_if.h.

Description

The Open function is responsible for preparing an QSPIX channel for operation. This function must be called once prior to calling any other QSPIX API functions (except R_QSPIX_GetVersion). Once successfully completed, the status of the selected channel QSPIX will be set to "open". After that, this function should not be called again for the same QSPIX channel without first performing a "close" by calling R_QSPIX_Close(). Communication is not yet available upon completion of this processing.
After the driver is open, the QSPIX can be accessed like internal flash memory starting at address 0x70000000.

Example

Condition: Channel not yet open

```
void rom_error_access(void *p_cbdat);
void rom_error_access(void *p_cbdat)
{
  /* Code for handle ROM error access */

  R_BSP_NOP();
}
qspix_err_t ret = QSPIX_SUCCESS;
qspix_cfg_t p_cfg;
/* Set read mode */
p_cfg.read_mode = QSPIX_READ_MODE_STANDARD;


/* Set Slave Select Extension */
p_cfg.slave_select_extension = QSPIX_DO_NOT_EXTEND_QSSL;

/* Set Prefetch Function Enable */
p_cfg.prefetch_function = QSPIX_PREFETCH_ENABLE;

/* Set Clock Mode */
p_cfg.clock_mode = QSPIX_SPI_MODE_0;
```

```
    /* Set Data Output Duration Extension */
    p_cfg.data_output_select = QSPIX_DO_NOT_EXTEND;

    /* Set Special Read Instruction Select */
    p_cfg.special_instruction_select=QSPIX_DEFAULT_INSTRUCTION_CODE;

    /* Set Slave Select High Width Setting */
    p_cfg.slave_select_high_width=QSPIX_1_CYCLE_QSPCLK;

    /* Set Slave Select Hold Time Setting */
    p_cfg.slave_select_hold_time=QSPIX_RELEASE_QSSL_0_5_QSPCLK;

    /* Set Slave Select Setup Time Setting */
    p_cfg.slave_select_setup_time=QSPIX_OUTPUT_QSSL_0_5_QSPCLK;

    /* Set Clock Divisor Select */
    p_cfg.clock_divisor=QSPIX_ICLK_DIV_2;

    /* Set Address Size */
    p_cfg.address_size=QSPIX_3_BYTES;

    /* Set Instruction with 4-Byte Address Enable */
    p_cfg.instruction_4_Byte_address=QSPIX_INSTRUCTION_4_BYTE_ADDRESS_DISABLE;

    /* Set Number of Dummy Cycle */
    p_cfg.dummy_clocks=QSPIX_DEFAULT_DUMMY_CYCLES;

    /* Set SPI protocol */
    p_cfg.protocol=QSPIX_EXTENDED_SPI_PROTOCOL;

    /* Set WP Pin Control */
    p_cfg.WP_pin_control=QSPIX_LOW_LEVEL;

    /* Set callback function */
    p_cfg.p_callback = rom_error_access;

    /* Call R_QSPIX_Open */
    ret = R_QSPIX_Open(QSPIX_CH0,&p_cfg);

    /* If there were an error this would demonstrate error detection of API
calls. */
    if (QSPIX_SUCCESS != ret)
    {
        return error(); // Your error handling code would go here.
    }

    /* Initialize I/O port pins for use with the QSPIX peripheral.
     * This is specific to the MCU and ports chosen. */
    R_QSPIX_PinSet_QSPIX0();
```

R_QSPIX_Control()

This function is used to change settings.

Format

```
qspix_err_t  R_QSPIX_Control(uint8_t channel, qspix_cfg_t *p_cfg);
```

Parameters

channel

Number of the QSPIX channel to be control.

**p_cfg*

Pointer to QSPIX channel configuration data structure.

Return Values

QSPIX_SUCCESS	/* QSPIX processing completed without problem. */
QSPIX_ERR_NOT_OPEN	/* QSPIX module is not initialized yet. */
QSPIX_ERR_INVALID_ARG	/* Arguments are invalid. */
QSPIX_ERR_NULL_PTR	/* Argument pointers are NULL. */

Properties

The declaration is located in r_qspix_rx_if.h.

Description

This function modify the operating of QSPIX channel for a particular mode. This function must be called after R_QSPIX_Open.

Example

```
void rom_error_access(void *p_cbdat);
void rom_error_access(void *p_cbdat)
{
  /* Code for handle ROM error access */

  R_BSP_NOP();
}
qspix_err_t ret = QSPIX_SUCCESS;
qspix_cfg_t p_cfg;

/* Set read mode */
p_cfg.read_mode = QSPIX_READ_MODE_STANDARD;

/* Set Slave Select Extension */
p_cfg.slave_select_extension = QSPIX_DO_NOT_EXTEND_QSSL;

/* Set Prefetch Function Enable */
p_cfg.prefetch_function = QSPIX_PREFETCH_ENABLE;

/* Change Clock Mode */
p_cfg.clock_mode = QSPIX_SPI_MODE_3;

/* Set Data Output Duration Extension */
p_cfg.data_output_select = QSPIX_DO_NOT_EXTEND;

/* Set Special Read Instruction Select */
p_cfg.special_instruction_select=QSPIX_DEFAULT_INSTRUCTION_CODE;

/* Set Slave Select High Width Setting */
```

```
p_cfg.slave_select_high_width=QSPIX_1_CYCLE_QSPCLK;

/* Set Slave Select Hold Time Setting */
p_cfg.slave_select_hold_time=QSPIX_RELEASE_QSSL_0_5_QSPCLK;

/* Set Slave Select Setup Time Setting */
p_cfg.slave_select_setup_time=QSPIX_OUTPUT_QSSL_0_5_QSPCLK;

/* Set Clock Divisor Select */
p_cfg.clock_divisor=QSPIX_ICLK_DIV_2;

/* Set Address Size */
p_cfg.address_size=QSPIX_3_BYTES;

/* Set Instruction with 4-Byte Address Enable */
p_cfg.instruction_4_Byte_address=QSPIX_INSTRUCTION_4_BYTE_ADDRESS_DISABLE;

/* Set Number of Dummy Cycle */
p_cfg.dummy_clocks=QSPIX_DEFAULT_DUMMY_CYCLES;

/* Set SPI protocol */
p_cfg.protocol=QSPIX_EXTENDED_SPI_PROTOCOL;

/* Set WP Pin Control */
p_cfg.WP_pin_control=QSPIX_LOW_LEVEL;

/* Set callback function */
p_cfg.p_callback = rom_error_access;

/* Call R_QSPIX_Control */
ret = R_QSPIX_Control(QSPIX_CH0,&p_cfg);

/* If there were an error this would demonstrate error detection of API
calls. */
if (QSPIX_SUCCESS != ret)
{
    return error(); // Your error handling code would go here.
}
```

R_QSPIX_Write_Indirect()

Writes raw data directly to the QSPIX.

Format

```
qspix_err_t  R_QSPIX_Write_Indirect(uint8_t channel,
                                    uint8_t *p_src_addr,
                                    uint32_t bytes,
                                    bool read_after_write);
```

Parameters

channel
Channel of QSPIX use to write data.
**p_src_addr*
Pointer to data to write.
bytes
Number of bytes to write.
read_after_write
Whether or not to close SPI bus cycle.

Return Values

QSPIX_SUCCESS	/* Processing completed without problem */
QSPIX_ERR_NOT_OPEN	/* QSPIX module is not initialized yet */
QSPIX_ERR_INVALID_ARG	/* Invalid argument */
QSPIX_ERR_NULL_PTR	/* Argument pointers are NULL */

Properties

Prototyped in file “r_qspix_rx_if.h”

Description

Uses the QSPIX of the channel number specified by the argument channel to write data.
Number of bytes is depend on the Flash(maximum size of 1 page) is used.
When writing a large volume of data, communication is divided into page units.

Example

```
#define QSPIX_COMMAND_READ_MODE_STANDARD (0x03)
/* Conditions: Channel currently open. */
/* Pointer to store data */
uint8_t data[4];
/* Pointer to data to write */
uint8_t buffer [10];

data[0] = QSPIX_COMMAND_READ_MODE_STANDARD;
data[1] = 0x00; Input the higher-order byte at address 0x000000.
data[2] = 0x00; Input the middle byte at target address 0x000000.
data[3] = 0x00; Input the lower-order byte at target address 0x000000.

/* Send the standard 0x03 command followed by 3 address bytes */
R_QSPIX_Write_Indirect(QSPIX_CH0, &data[0],4,true);

/* Read 10 bytes from Flash */
R_QSPIX_Read_Indirect(QSPIX_CH0, &buffer[0],10);
```

Special Notes:

When Extended SPI protocol is used in indirect access mode, the standard Read or Fast Read instruction must be used to reference the contents of the serial flash. The QSPIX does not support Fast Read Dual Output, Fast Read Dual I/O, Fast Read Quad Output, or Fast Read Quad I/O transfers in this configuration. When these high-speed read operations are required, use ordinary flash access.

R_QSPIX_Read_Indirect()

Reads raw data directly from the QSPIX. This API can only be called after R_QSPIX_Write_Indirect() with read_after_write set to true.

Format

```
qspix_err_t  R_QSPIX_Read_Indirect(uint8_t channel,
                                   uint8_t *p_des_addr,
                                   uint32_t bytes);
```

Parameters

channel
Channel of QSPIX use to read data.
**p_des_addr*
Pointer to store data.
bytes
Number of bytes to read.

Return Values

QSPIX_SUCCESS	/* Processing completed without problem*/
QSPIX_ERR_NOT_OPEN	/* QSPIX module is not initialized yet */
QSPIX_ERR_INVALID_ARG	/* Invalid argument */
QSPIX_ERR_NULL_PTR	/* Argument pointers are NULL */

Properties

The declaration is located in r_qspix_rx_if.h.

Description

Uses the QSPIX of the channel number specified by the argument channel to read data.
Number of bytes is depend on the Flash is used.

Example

```
#define QSPIX_COMMAND_READ_MODE_STANDARD (0x03)
/* Conditions: Channel currently open. */
/* Pointer to store data */
uint8_t data[4];
/* Pointer to data to write */
uint8_t buffer [10];

data[0] = QSPIX_COMMAND_READ_MODE_STANDARD;
data[1] = 0x00; Input the higher-order byte at address 0x000000.
data[2] = 0x00; Input the middle byte at target address 0x000000.
data[3] = 0x00; Input the lower-order byte at target address 0x000000.

/* Send the standard 0x03 command followed by 3 address bytes */
R_QSPIX_Write_Indirect(QSPIX_CH0, &data[0],4,true);

/* Read 10 bytes from Flash */
R_QSPIX_Read_Indirect(QSPIX_CH0, &buffer[0],10);
```

Special Notes:

When Extended SPI protocol is used in indirect access mode, the standard Read or Fast Read instruction must be used to reference the contents of the serial flash. The QSPIX does not support Fast Read Dual Output, Fast Read Dual I/O, Fast Read Quad Output, or Fast Read Quad I/O transfers in this configuration. When these high-speed read operations are required, use ordinary flash access.

R_QSPIX_Enter_XIP()

Enters XIP(execute in place) mode.

Format

```
qspix_err_t  R_QSPIX_Enter_XIP(uint8_t channel, uint8_t mode);
```

Parameters

channel

Channel of QSPIX Enters XIP (execute in place) mode.

mode

Enter XIP Mode.

Return Values

```
QSPIX_SUCCESS          /* Processing completed without problem */
QSPIX_ERR_NOT_OPEN     /* QSPIX module is not initialized yet */
QSPIX_ERR_HW           /* Hardware error */
```

Properties

Prototyped in file “r_qspix_rx_if.h”

Description

To select the XIP mode, specify the XIP mode configuration for the serial flash device in the MODE [7:0] bits in the SPDCR register.

Example

```
/* Conditions: Channel currently open. */
/* Enter XIP */
R_QSPIX_Enter_XIP(QSPIX_CH0,0xA5);
```

R_QSPIX_Exit_XIP()

Exits XIP(execute in place) mode.

Format

```
qspix_err_t  R_QSPIX_Exit_XIP(uint8_t channel, uint8_t mode);
```

Parameters

channel

Channel of QSPIX Exits XIP (execute in place) mode

mode

Exit XIP Mode

Return Values

```
QSPIX_SUCCESS          /* Processing completed without problem */
QSPIX_ERR_NOT_OPEN     /* QSPIX module is not initialized yet */
QSPIX_ERR_HW           /* Hardware error  */
```

Properties

Prototyped in file “r_qspix_rx_if.h”

Description

To select the XIP mode, specify the XIP mode configuration for the serial flash device in the MODE [7:0] bits in the SPDCR register.

Example

```
/* Conditions: Channel currently open. */
/* Exits XIP */
R_QSPIX_Exit_XIP(QSPIX_CH0, 0xFF);
```

R_QSPIX_BankSet()

Selects the bank to access.

Format

```
qspix_err_t  R_QSPIX_BankSet(uint8_t channel, uint8_t bank);
```

Parameters

channel

Channel for switch bank.

bank

Bank need to set.

Return Values

```
QSPIX_SUCCESS           /* Processing completed without problem */
QSPIX_ERR_INVALID_ARG   /* Arguments are invalid */
QSPIX_ERR_NOT_OPEN      /* QSPIX module is not initialized yet */
```

Properties

Prototyped in file “r_qspix_rx_if.h”

Description

A bank is a 64MB sliding access window into the QSPI device flash memory space. To access chip address 0x4000000, select bank 1, then read from internal flash address 0x70000000. To access chip address 0x8001000, select bank 2, then read from internal flash address 0x70001000.

This function is not required for memory devices less than or equal to 512 Mb (64MB).

Example

```
/* Conditions: Channel currently open. */
/* Select bank 1*/
R_QSPIX_BankSet(QSPIX_CH0,1);
```

R_QSPIX_Set_Spi_Protocol()

Selects the protocol to access.

Format

```
qspix_err_t  R_QSPIX_Set_Spi_Protocol(uint8_t channel,
                                      qspix_protocol_t protocol);
```

Parameters

channel

Channel for set protocol.

protocol

Protocol need to set.

Return Values

```
QSPIX_SUCCESS           /* Processing completed without problem */
QSPIX_ERR_INVALID_ARG   /* Arguments are invalid */
QSPIX_ERR_NOT_OPEN      /* QSPIX module is not initialized yet */
```

Properties

Prototyped in file “r_qspix_rx_if.h”

Description

Example

```
/* Conditions: Channel currently open. */
/* Select protocol Quad SPI protocol */
R_QSPIX_Set_Spi_Protocol(QSPIX_CH0, QSPIX_QUAD_SPI_PROTOCOL);
```

R_QSPIX_Get_Status()

Get status of Prefetch, busy flag, ROM access error flag.

Format

```
qspix_err_t  R_QSPIX_Get_Status(qspix_cmd_t cmd, uint8_t *return_status);
```

Parameters

cmd

Command to get status

**return_status*

Pointer store status corresponding with command

Return Values

```
QSPIX_SUCCESS                 /* Processing completed without problem */
QSPIX_ERR_INVALID_COMMAND     /* Arguments are invalid */
```

Properties

Prototyped in file “r_qspix_rx_if.h”

Description

This function return status corresponding with input cmd.

```
typedef enum e_qspix_cmd
{
   QSPIX_GET_PREFETCH_BUFFER_FILL_LEVEL, /* Get Prefetch Buffer Fill Level */
   QSPIX_GET_PREFETCH_BUFFER_FULL_FLAG, /* Get Prefetch Buffer Full Flag */
   QSPIX_GET_PREFETCH_FUNCTION_OPERATING_STATUS_FLAG, /* Get Prefetch Function
                                    Operating Status Flag */
   QSPIX_GET_BUS_BUSY_FLAG,        /* Get Bus Busy Flag */
   QSPIX_GET_ROM_ACCESS_ERROR_FLAG, /* Get ROM Access Error Flag */
   QSPIX_GET_XIP_STATUS_FLAG       /* Get XIP Status Flag */
} qspix_cmd_t;
```

Example

```
    /* Conditions: Channel currently open. */
    /* Variable store to status */
    uint8_t status;

    /* Get prefetch buffer fill level and store to variable status*/
    R_QSPIX_Get_Status(QSPIX_GET_PREFETCH_BUFFER_FILL_LEVEL, &status);
```

R_QSPIX_Close()

Close the QSPIX driver module.

Format

```
qspix_err_t  R_QSPIX_Close(uint8_t channel);
```

Parameters

channel

Channel of QSPIX need to close.

Return Values

```
QSPIX_SUCCESS           /* Successful; channel initialized */
QSPIX_ERR_NOT_OPEN      /* QSPIX module is not initialized yet */
QSPIX_ERR_INVALID_ARG   /* Invalid argument */
```

Properties

The declaration is located in r_qspix_rx_if.h.

Description

Disables the QSPIX channel and enters module-stop state. The QSPIX channel cannot be used again until it has been reopened with the R_QSPIX_Open function. If this function is called for an QSPIX that is not in the open state then an error code is returned.

Example

```
/* Conditions: Channel currently open. */

qspix_err_t ret;
ret = R_QSPIX_Close(QSPIX_CH0);
if (QSPIX_SUCCESS != result)
{
    return result;
}
```

R_QSPIX_GetVersion()

This function returns the version number of the API.

Format

```
uint32_t  R_QSPIX_GetVersion(void);
```

Parameters

None

Return Values

Version Number

Properties

The declaration is located in r_qspix_rx_if.h.

Description

The function returns the version of this module. The version number is encoded such that the top two bytes are the major version number and the bottom two bytes are the minor version number.

Example

```
/* Retrieve the version number and convert it to a string. */
uint32_t version, version_high, version_low;
char version_str[9];

version = R_QSPIX_GetVersion();
version_high = (version >> 16)&0xf;
version_low = version & 0xff;


sprintf(version_str, "QSPIXv%1.1hu.%2.2hu", version_high, version_low);.
```

4. Pin Setting

To use the QSPIX FIT module, assign input/output signals of the peripheral function to pins with the multi-function pin controller (MPC). The pin assignment is referred to as the “Pin Setting” in this document.

Please perform the pin setting after calling the R_QSPIX_Open function.

When performing the pin setting in the e^2 studio, the Pin Setting feature of the FIT Configurator or the Smart Configurator can be used. When using the Pin Setting feature, a source file is generated according to the option selected in the Pin Setting window in the FIT Configurator or the Smart Configurator. Then pins are configured by calling the function defined in the source file. Refer to Table 4.1 Function Output by the FIT Configurator

Table 4.1 Function Output by the FIT Configurator

MCU Used	Function to be Output	Remarks
All MCUs	R_QSPIX_PinSet_QSPIXx	x: Channel number

5. Sample program

This application note includes one or more sample program to demonstrate basic usage of the FIT QSPIX Module. The sample program is intended to provide a quick functional example of common API function calls in use.

The provided sample application execute flash erasing, blank check, and programming. Each write function is verified with a read-back of data.

5.1 Adding the Sample program to a Workspace

Currently, Sample program is not supported..

5.2 Running the Sample program

Currently, Sample program is not supported.

6. Appendices

6.1 Operation Confirmation Environment

This section describes for detailed the operating test environments of this module.

Table 6-1 Confirmed Operation Environment (Rev.1.00)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 2021-07 IAR Embedded Workbench for Renesas RX 4.20.3
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.202004 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99 Linker option: The following user defined option should be added to the default settings of the integrated development environment, if “Optimize size (-Os)” is used: -Wl,--no-gc-sections This is to work around a GCC linker issue whereby the linker erroneously discard interrupt functions declared in FIT peripheral module
	IAR C/C++ Compiler for Renesas RX version 4.20.3 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.1.00
Board used	Renesas Starter Kit+ for RX671 (product No.: RTK55671xxxxxxxxx)

6.2 Troubleshooting

(1) Q: I have added the FIT module to the project and built it. Then I got the error: Could not open source file "platform.h".

A: The FIT module may not be added to the project properly. Check if the method for adding FIT modules is correct with the following documents:

- Using CS+:

  Application note "Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)"

- Using e^2 studio:

  Application note "Adding Firmware Integration Technology Modules to Projects (R01AN1723)"

When using this FIT module, the board support package FIT module (BSP module) must also be added to the project. Refer to the application note "Board Support Package Module Using Firmware Integration Technology (R01AN1685)".

(2) Q: I have added the FIT module to the project and built it. Then I got the error: This MCU is not supported by the current r_qspix_rx module.

A: The FIT module you added may not support the target device chosen in your project. Check the supported devices of added FIT modules.

(3) Q: I have added the FIT module to the project and built it. Then I got an error for when the configuration setting is wrong.

A: The setting in the file "r_qspix_rx_config.h" may be wrong. Check the file "r_qspix_rx_config.h". If there is a wrong setting, set the correct value for that. Refer to 2.8 Configuration Overview for details.

7. Reference Documents

User's Manual: Hardware

Technical Update/Technical News

User's Manual: Development Tools
The latest version can be downloaded from the Renesas Electronics website.

Related Technical Update

Not applicable technical update for this module.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar 31, 2021	--	First edition issued

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.