RX Family, RL78 Family, 78K0R/Kx3-L

R01AN1439EJ0104
Rev.1.04
Mar 31, 2016

Micron Technology P5Q Serial Phase Change Memory Control Software

Introduction

This application note describes how to control P5Q serial phase change memory, manufactured by Micron Technology, Inc., using an MCU manufactured by Renesas Electronics, and it explains the usage of the sample code provided for that purpose.

Note that the sample code is upper-layer software for controlling the serial phase change memory as a slave device.

Lower-layer software (clock synchronous single master control software) for controlling the SPI modes specific to individual MCU models is available separately, and should be obtained by the user, so please obtain this from the following URL as well. In addition, when a new microcontroller is added to the clock synchronous single-master control software, update of this application note may not be in time. Refer to 'Clock Synchronous Single Master Control Software (Lower-level layer of the software)' information in the following URL for the combination information on the latest supported microcontroller and its single-master control software.

- SPI/QSPI Serial Flash Memory, QSPI Serial Phase Change Memory Driver
  http://www.renesas.com/driver/spi_serial_flash

Target Device

Serial phase change memory: P5Q serial phase change memory, manufactured by Micron Technology, Inc.

Contact Micron Technology, Inc., for information on obtaining P5Q serial phase change memory products.

MCUs on which operation has been confirmed:

RX600 series	: RX63N (using the RSPI)
RX100 series	: RX111 (using the SCI)
	: RX111 (using the RSPI)
RL78/G1x	: RL78/G14, RL78/G1C group (using the SAU)
RL78/L1x	: RL78/L12, RL78/L12, RL78/L1C group (using the SAU)

See 3, Reference Application Notes, regarding MCU models other than those listed above.

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

Note that the following abbreviations are used in this application note:

- Single-SPI (communication in single-SPI mode)
- Dual-SPI (communication in single-SPI mode)
- Quad-SPI (communication in single-SPI mode)

Contents

1. Specifications 4
2. Operation Confirmation Conditions 5
2.1 RX Family 5
2.2 RL78 Family, 78K0R/Kx3-L 7
3. Reference Application Notes 13
3.1 RX Family: List of Related Application Notes 13
3.2 RL78 Family, 78K0R Family: List of Related Application Notes 13
4. Hardware 14
4.1 Hardware Configuration 14
4.1.1 Pin Assignments for Single-SPI Configuration 14
4.1.2 Single-SPI Connection Example 14
4.1.3 Pin Assignments for Dual-SPI Configuration 15
4.1.4 Dual-SPI Connection Example 15
4.1.5 Pin Assignments for Quad-SPI Configuration 16
4.1.6 Quad-SPI Connection Example 16
5. Software 17
5.1 Operation Overview 17
5.1.1 Relationship Between Data Buffers and Transmit/Receive Data 17
5.1.2 Timing Generation in Clock Synchronous Mode 18
5.1.3 Serial Phase Change Memory S# Pin Control 20
5.1.4 Serial Phase Change Memory Instruction Codes 21
5.2 Software Configuration 22
5.3 Required Memory Size 23
5.3.1 RX Family 23
5.3.2 RL78 Family, 78K0R/Kx3-L 25
5.4 File Structure 28
5.5 Constants 29
5.5.1 Return Value 29
5.5.2 Command Definitions 29
5.5.3 Other Definitions 30
5.6 Structure/Union List 33
5.7 Variable 34
5.8 Functions 34
5.9 Function Specifications 35
5.9.1 Driver Initialization Processing 35
5.9.2 Status Register Read Processing 36

5.9.3 Write Protect Setting Processing 37
5.9.4 WRDI Command Issue Processing 40
5.9.5 Data Read Processing 41
5.9.6 Data Write Processing 42
5.9.7 Data Write Processing (for Single-Page Write) 44
5.9.8 Erase Processing 47
5.9.9 ID Read Processing 50
5.9.10 Busy Wait Processing 51

6. Application Example 53
6.1 Serial Phase Change Memory Control Software Settings 54
6.1.1 r_qspi_pcm_p5q.h 54
6.1.2 r_qspi_pcm_p5q_sfr.h 56
6.1.3 r_qspi_pcm_p5q_sub.h 59
6.1.4 r_qspi_pcm_p5q_sub.c 60
6.1.5 r_qspi_pcm_p5q_drvif.c 62
6.1.6 r_qspi_pcm_p5q_sfr_rl78.c 65

7. Usage Notes 66
7.1 Notes on Integrating Sample Code 66
7.2 Using an MCU with On-Chip Cache 66
7.3 Support for Other Capacities 66
7.4 Using Other Slave Devices 66
7.5 Voltage Stabilization Time After Power-On 66
7.6 Serial Phase Change Memory Usage Limitations 67

1. Specifications

A Renesas Electronics MCU is used to control P5Q serial phase change memory, manufactured by Micron Technology, Inc.

Separate MCU-specific clock synchronous single master control software is required.

Table 1-1 lists the peripheral functions used and their applications, and Figure 1.1 shows a usage example.

Summaries of the functions are provided below:

- The software functions as a device driver, with a Renesas Electronics MCU operating as the master device and the Micron Technology, Inc., P5Q serial phase change memory operating as the slave device.
- The MCU's on-chip serial communication function (clock synchronous mode) is used in a single-SPI, dual-SPI, or quad-SPI configuration to control operation.
- One serial communication function channel can be specified by the user for use. It is not possible to use multiple channels.
- It is possible to control up to two serial phase change memory devices of the same type name.
- The communication speed can be specified by the user.
- Both big-endian and little-endian operation are supported. (The choice depends on the MCU used.)

Table 1-1 Peripheral Functions and Their Applications

Peripheral Function	Application
MCU's on-chip serial communication functionality (clock synchronous mode)	Communication with SPI slave device by means of serial communication function (clock synchronous mode) 1 channel (required)
Port	For SPI slave device select control signal Number of ports equal to number of devices (required)

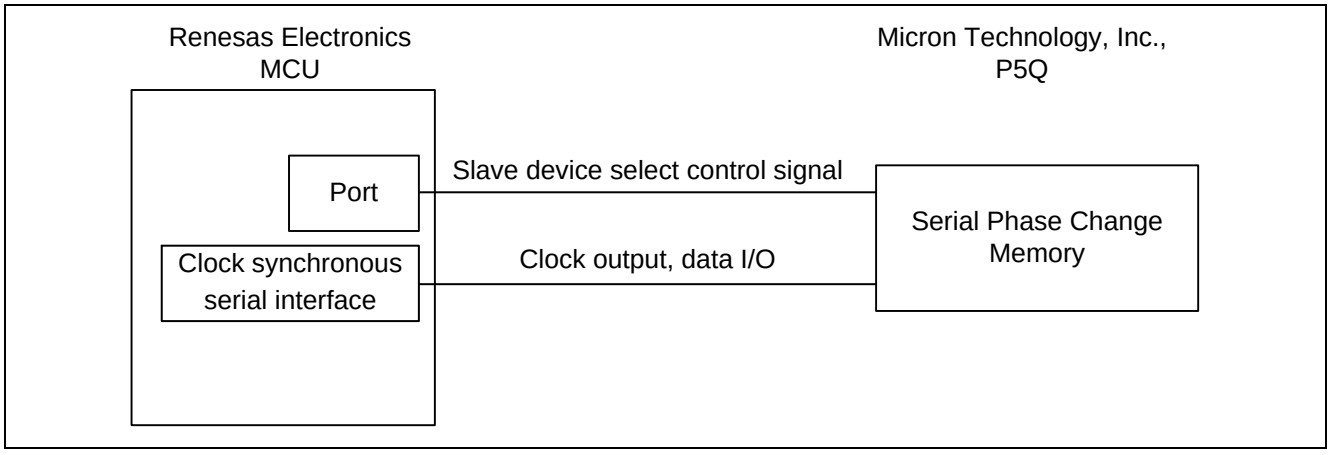


Figure 1.1 Usage Example

2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

2.1 RX Family

(1) **RX63N RSPI**

Table 2-1 Operation Confirmation Conditions

Item	Description
Memory	Micron Technology P5Q Serial Phase Change Memory
Microcontroller used	RX63N Group (Program ROM: 1 MB/RAM: 128 KB)
Operating frequency	ICLK: 96 MHz, PCLK: 48 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance embedded Workshop Version 4.09.01.007
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 1.2.1.0)
	Compiler options The integrated development environment default settings are used.
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.20
Software	Clock synchronous single master control software using the RSPI, for RX63N, version 2.04
Board	Renesas Starter Kit for RX63N

(2) **RX111 RSPI**

Table 2-2 Operation Confirmation Conditions

Item	Description
Memory	Micron Technology P5Q Serial Phase Change Memory
Microcontroller used	RX111 Group (Program ROM: 128 KB, RAM: 16 KB)
Operating frequency	ICLK: 32 MHz, PCLK: 32 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics CubeSuite+ V2.01.00
C compiler	Renesas Electronics RX family C/C++ compiler package (Toolchain 2.01.00)
	Compiler options: The default settings (Optimize Level: 2, Optimize for size) for the integrated development environment are used.
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.21.R01
Software	RX210, RX21A, RX220, RX63N, RX63T, RX111 Group Clock Synchronous Single Master Control Software Using the RSPI (R01AN1196EJ), Ver. 2.04.R04
Board	Renesas Starter Kit for RX111

(3) **RX111 SCI**

Table 2-3 Operation Confirmation Conditions

Item	Description
Memory	Micron Technology P5Q Serial Phase Change Memory
Microcontroller used	RX111 Group (Program ROM: 128 KB, RAM: 16 KB)
Operating frequency	ICLK: 32 MHz, PCLK: 32 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics CubeSuite+ V2.01.00
C compiler	Renesas Electronics RX family C/C++ compiler package (Toolchain 2.01.00)
	Compiler options: The default settings (Optimize Level: 2, Optimize for size) for the integrated development environment are used.
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.21.R01
Software	RX210, RX21A, RX220, RX63N, RX63T,RX111 Group Clock Synchronous Single Master Control Software Using the SCI (R01AN1229EJ), Ver. 2.01.R05
Board	Renesas Starter Kit for RX111

2.2 RL78 Family, 78K0R/Kx3-L

(1) **RL78/G14 Integrated Development Environment CS+ for CA,CX (Compiler: CA78K0R)**

Table 2-4 Operation Confirmation Conditions

Item	Description
Memory	Micron Technology P5Q Serial Phase Change Memory
Microcontroller used	RL78/G14 Group (Program ROM: 256 KB/RAM: 24 KB)
Operating frequency	Main system clock: 24 MHz CPU/peripheral hardware clock: 24 MHz Serial clock: 6 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics CS+ for CA, CX V3.01.00
C compiler	Renesas Electronics RL78,78K0R compiler CA78K0R V1.71
	Compiler options: The default settings (-qx2) for the integrated development environment are used.
Endian order	Little endian
Sample code version number	Ver. 2.22
Software	RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ), Ver. 2.05
Board	Renesas Starter Kit for RL78/G14

(2) **RL78/G14 Integrated Development Environment CS+ for CC (Compiler: CC-RL)**

Table 2-5 Operation Confirmation Conditions

Item	Description
Memory	Micron Technology P5Q Serial Phase Change Memory
Microcontroller used	RL78/G14 Group (Program ROM: 256 KB/RAM: 24 KB)
Operating frequency	Main system clock: 24 MHz CPU/peripheral hardware clock: 24 MHz Serial clock: 6 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics CS+ for CC V3.03.00
C compiler	Renesas Electronics RL78 compiler CC-RL V1.02.00
	Compiler options: The default settings (Perform the default optimization(None)) for the integrated development environment are used.
Endian order	Little endian
Sample code version number	Ver. 2.22
Software	RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ), Ver. 2.05
Board	Renesas Starter Kit for RL78/G14

(3) **RL78/G14 Integrated Development Environment IAR Embedded Workbench**

Table 2-6 Operation Confirmation Conditions

Item	Description
Memory	Micron Technology P5Q Serial Phase Change Memory
Microcontroller used	RL78/G14 Group (Program ROM: 256 KB/RAM: 24 KB)
Operating frequency	Main system clock: 24 MHz CPU/peripheral hardware clock: 24 MHz Serial clock: 6 MHz
Operating voltage	3.3 V
Integrated development environment	IAR Systems IAR Embedded Workbench for Renesas RL78 (Ver.1.30.2)
C compiler	IAR Systems IAR Assembler for Renesas RL78 (Ver.1.30.2.50666) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.2.50666)
	Compiler options: The default settings (Low) for the integrated development environment are used.
Endian order	Little endian
Sample code version number	Ver. 2.21
Software	Clock synchronous single master control software using CSI mode of serial array unit, version 2.03
Board	Renesas Starter Kit for RL78/G14

(4) **RL78/G1C Integrated Development Environment CubeSuite+**

Table 2-7 Operation Confirmation Conditions

Item	Description
Memory	Micron Technology P5Q Serial Phase Change Memory
Microcontroller used	RL78/G1C Group (Program ROM: 32 KB/RAM: 5.5 KB)
Operating frequency	Main system clock: 24 MHz CPU/peripheral hardware clock: 24 MHz Serial clock: 12 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation CubeSuite+ V2.01.00
C compiler	Renesas Electronics Corporation CubeSuite+ RL78,78K0R Compiler CA78K0R V1.70
	Compiler options The integrated development environment default settings ("-qx2") are used.
Endian order	Little endian
Sample code version number	Ver. 2.21.R01
Software	RL78/G14,RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), Ver. 2.03
Board	Renesas RL78/G1C Target Board QB-R5F10JGC-TB

(5) **RL78/G1C Integrated Development Environment IAR Embedded Workbench**

Table 2-8 Operation Confirmation Conditions

Item	Description
Memory	Micron Technology P5Q Serial Phase Change Memory
Microcontroller used	RL78/G1C Group (Program ROM: 256 KB/RAM: 24 KB)
Operating frequency	Main system clock: 24 MHz CPU/peripheral hardware clock: 24 MHz Serial clock: 12 MHz
Operating voltage	3.3 V
Integrated development environment	IAR Systems IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5)
C compiler	IAR Systems IAR Assembler for Renesas RL78 (Ver.1.30.4.50715) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715)
	Compiler options: The default settings (Low) for the integrated development environment are used.
Endian order	Little endian
Sample code version number	Ver. 2.21.R01
Software	RL78/G14,RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), Ver. 2.03
Board	Renesas RL78/G1C Target Board QB-R5F10JGC-TB

(6) **RL78/L12 Integrated Development Environment CubeSuite+**

Table 2-9 Operation Confirmation Conditions

Item	Description
Memory	Micron Technology P5Q Serial Phase Change Memory
Microcontroller used	RL78/L12 Group (Program ROM: 32 KB, RAM:1.5 KB)
Operating frequency	Main system clock: 24 MHz CPU/peripheral hardware clock: 24 MHz Serial clock: 6 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation CubeSuite+ V2.01.00
C compiler	Renesas Electronics Corporation CubeSuite+ RL78,78K0R Compiler CA78K0R V1.70
	Compiler options The integrated development environment default settings ("-qx2") are used.
Endian order	Little endian
Sample code version number	Ver. 2.21.R01
Software	RL78/G14,RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), Ver. 2.03
Board	Renesas Starter Kit for RL78/L12

(7) **RL78/L12 Integrated Development Environment IAR Embedded Workbench**

Table 2-10 Operation Confirmation Conditions

Item	Description
Memory	Micron Technology P5Q Serial Phase Change Memory
Microcontroller used	RL78/L12 Group (Program ROM: 32 KB, RAM:1.5 KB)
Operating frequency	Main system clock: 24 MHz CPU/peripheral hardware clock: 24 MHz Serial clock: 6 MHz
Operating voltage	3.3 V
Integrated development environment	IAR Systems IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5)
C compiler	IAR Systems IAR Assembler for Renesas RL78 (Ver.1.30.4.50715) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715)
	Compiler options: The default settings (Low) for the integrated development environment are used.
Endian order	Little endian
Sample code version number	Ver. 2.21.R01
Software	RL78/G14,RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), Ver. 2.03
Board	Renesas Starter Kit for RL78/L12

(8) **RL78/L13 Integrated Development Environment CubeSuite+**

Table 2-11 Operation Confirmation Conditions

Item	Description
Memory	Micron Technology P5Q Serial Phase Change Memory
Microcontroller used	RL78/L13 Group (Program ROM: 128 KB/RAM: 8 KB)
Operating frequency	Main system clock: 24 MHz CPU/peripheral hardware clock: 24 MHz Serial clock: 6 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation CubeSuite+ V2.01.00
C compiler	Renesas Electronics Corporation CubeSuite+ RL78,78K0R Compiler CA78K0R V1.70
	Compiler options The integrated development environment default settings ("-qx2") are used.
Endian order	Little endian
Sample code version number	Ver. 2.21.R01
Software	RL78/G14,RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), Ver. 2.03
Board	Renesas Starter Kit for RL78/L13

(9) **RL78/L13 Integrated Development Environment IAR Embedded Workbench**

Table 2-12 Operation Confirmation Conditions

Item	Description
Memory	Micron Technology P5Q Serial Phase Change Memory
Microcontroller used	RL78/L13 Group (Program ROM: 128 KB/RAM: 8 KB)
Operating frequency	Main system clock: 24 MHz CPU/peripheral hardware clock: 24 MHz Serial clock: 6 MHz
Operating voltage	3.3 V
Integrated development environment	IAR Systems IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5)
C compiler	IAR Systems IAR Assembler for Renesas RL78 (Ver.1.30.4.50715) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715)
	Compiler options: The default settings (Low) for the integrated development environment are used.
Endian order	Little endian
Sample code version number	Ver. 2.21.R01
Software	RL78/G14,RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), Ver. 2.03
Board	Renesas Starter Kit for RL78/L13

(10) **RL78/L1C Integrated Development Environment CubeSuite+**

Table 2-13 Operation Confirmation Conditions

Item	Description
Memory	Micron Technology P5Q Serial Phase Change Memory
Microcontroller used	RL78/L1C Group (Program ROM: 256 KB/RAM: 16 KB)
Operating frequency	Main system clock: 24 MHz CPU/peripheral hardware clock: 24 MHz Serial clock: 6 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation CubeSuite+ V2.01.00
C compiler	Renesas Electronics Corporation CubeSuite+ RL78,78K0R Compiler CA78K0R V1.70
	Compiler options The integrated development environment default settings ("-qx2") are used.
Endian order	Little endian
Sample code version number	Ver. 2.21.R01
Software	RL78/G14,RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), Ver. 2.03
Board	Renesas Starter Kit for RL78/L1C

(11) **RL78/L1C Integrated Development Environment IAR Embedded Workbench**

Table 2-14 Operation Confirmation Conditions

Item	Description
Memory	Micron Technology P5Q Serial Phase Change Memory
Microcontroller used	RL78/L1C Group (Program ROM: 256 KB/RAM: 16 KB)
Operating frequency	Main system clock: 24 MHz CPU/peripheral hardware clock: 24 MHz Serial clock: 6 MHz
Operating voltage	3.3 V
Integrated development environment	IAR Systems IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5)
C compiler	IAR Systems IAR Assembler for Renesas RL78 (Ver.1.30.4.50715) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715)
	Compiler options: The default settings (Low) for the integrated development environment are used.
Endian order	Little endian
Sample code version number	Ver. 2.21.R01
Software	RL78/G14,RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), Ver. 2.03
Board	Renesas Starter Kit for RL78/L1C

3. Reference Application Notes

For additional information associated with this document, refer to the following application notes.

In the related application notes listed below, refer to the “Target Device” item on the cover for a listing of MCU models on which operation has been confirmed.

3.1 RX Family: List of Related Application Notes

- RX610 Group Clock Synchronous Single Master Control Software Using the SCI (R01AN0534EJ)
- RX62N Group Clock Synchronous Single Master Control Software Using the RSPI (R01AN0323EJ)
- RX62N Group Clock Synchronous Single Master Control Software Using the SCI (R01AN1088EJ)
- RX210, RX21A, RX220, RX63N, RX63T, RX111 Group Clock Synchronous Single Master Control Software Using the RSPI (R01AN1196EJ)
- RX210, RX21A, RX220, RX63N, RX63T, RX111 Group Clock Synchronous Single Master Control Software Using the SCI (R01AN1229EJ)

3.2 RL78 Family, 78K0R Family: List of Related Application Notes

- 78K0R/Kx3-L Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN0708EJ)
- RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ)

4. Hardware

4.1 Hardware Configuration

An example hardware configuration is shown below.

4.1.1 Pin Assignments for Single-SPI Configuration

The following table lists the MCU pins used for single-SPI operation and their functions.

Table 4-1 Single-SPI Pins and Functions

MCU Pin Name	I/O	Description
CLK	Output	Clock output
DataOut	Output	Master data output
DataIn	Input	Master data input
Port (CS#)	Output	Slave device select output

4.1.2 Single-SPI Connection Example

A connection example for single-SPI operation is shown below:

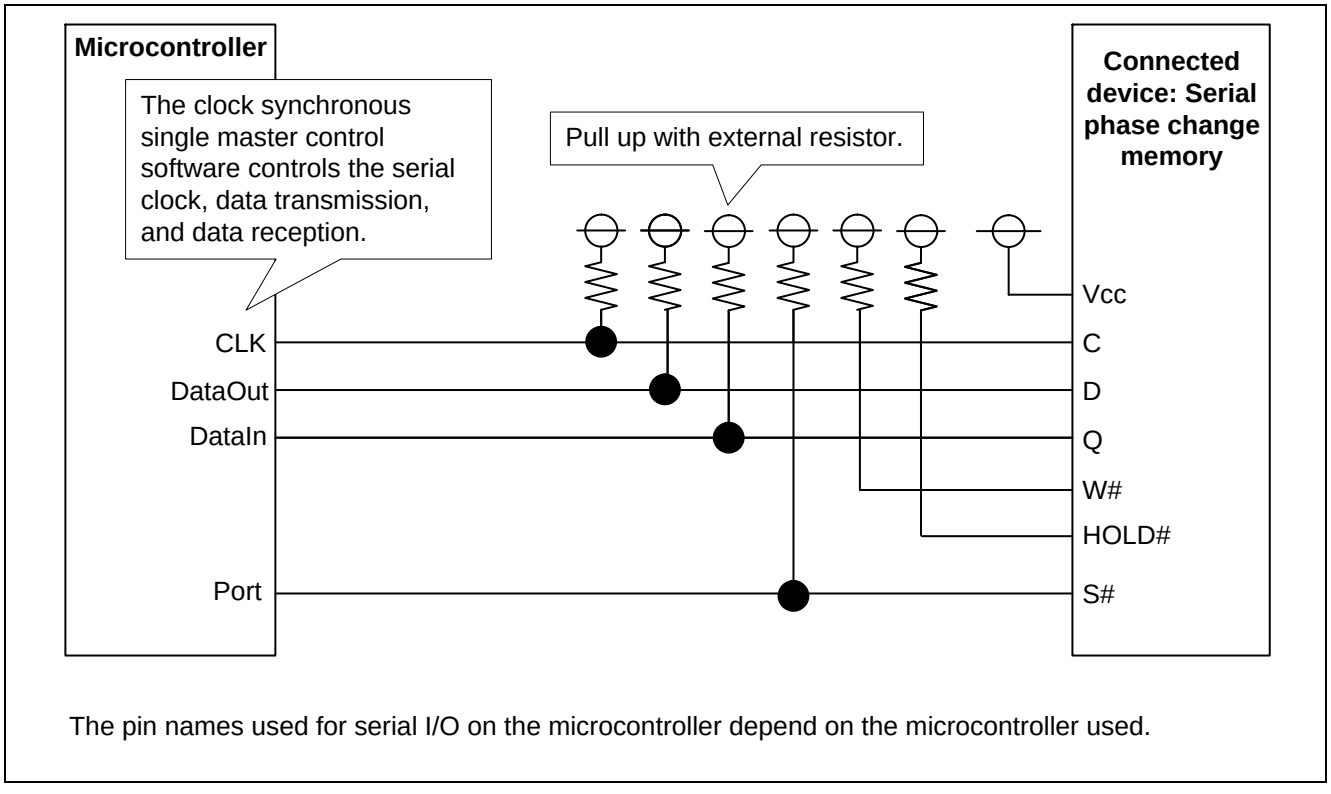


Figure 4.1 MCU and SPI Slave Device Connection Example for Single-SPI

4.1.3 Pin Assignments for Dual-SPI Configuration

The following table lists the MCU pins used for dual-SPI operation and their functions.

In order to use a dual-SPI configuration, the MCU must have a quad serial peripheral interface function.

Table 4-2 Dual-SPI Pins and Functions

MCU Pin Name	I/O	Description
CLK	Output	Clock output
DataIn/Out0	Input/output	Master data input/output 0
DataIn/Out1	Input/output	Master data input/output 1
Port(CS#)	Output	Slave device select output

4.1.4 Dual-SPI Connection Example

A connection example for dual-SPI operation is shown below:

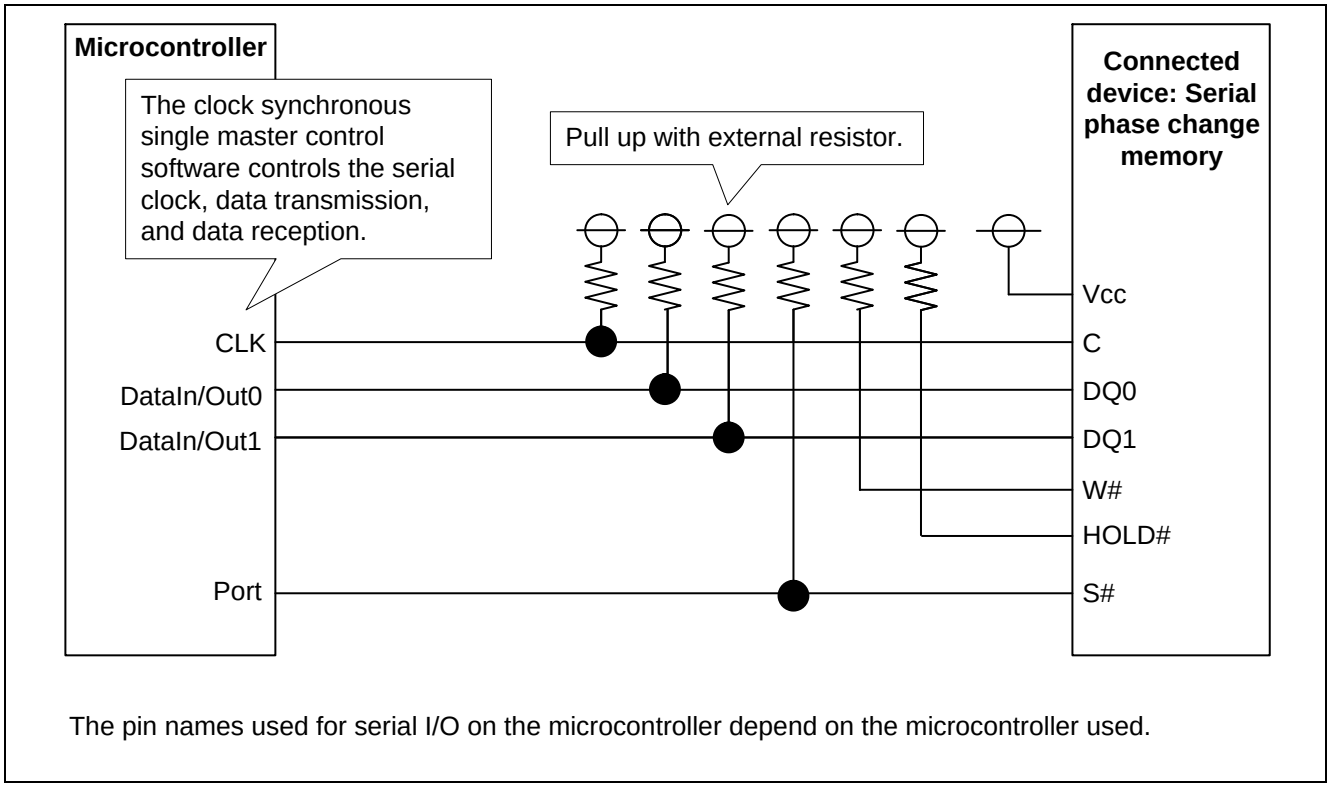


Figure 4.2 MCU and SPI Slave Device Connection Example for Dual-SPI

4.1.5 Pin Assignments for Quad-SPI Configuration

The following table lists the MCU pins used for quad-SPI operation and their functions.

In order to use a quad-SPI configuration, the MCU must have a quad serial peripheral interface function.

Table 4-3 Quad-SPI Pins and Functions

MCU Pin Name	I/O	Description
CLK	Output	Clock output
DataIn/Out0	Input/output	Master data input/output 0
DataIn/Out1	Input/output	Master data input/output 1
DataIn/Out2	Input/output	Master data input/output 2
DataIn/Out3	Input/output	Master data input/output 3
Port (CS#)	Output	Slave device select output

4.1.6 Quad-SPI Connection Example

A connection example for quad-SPI operation is shown below:

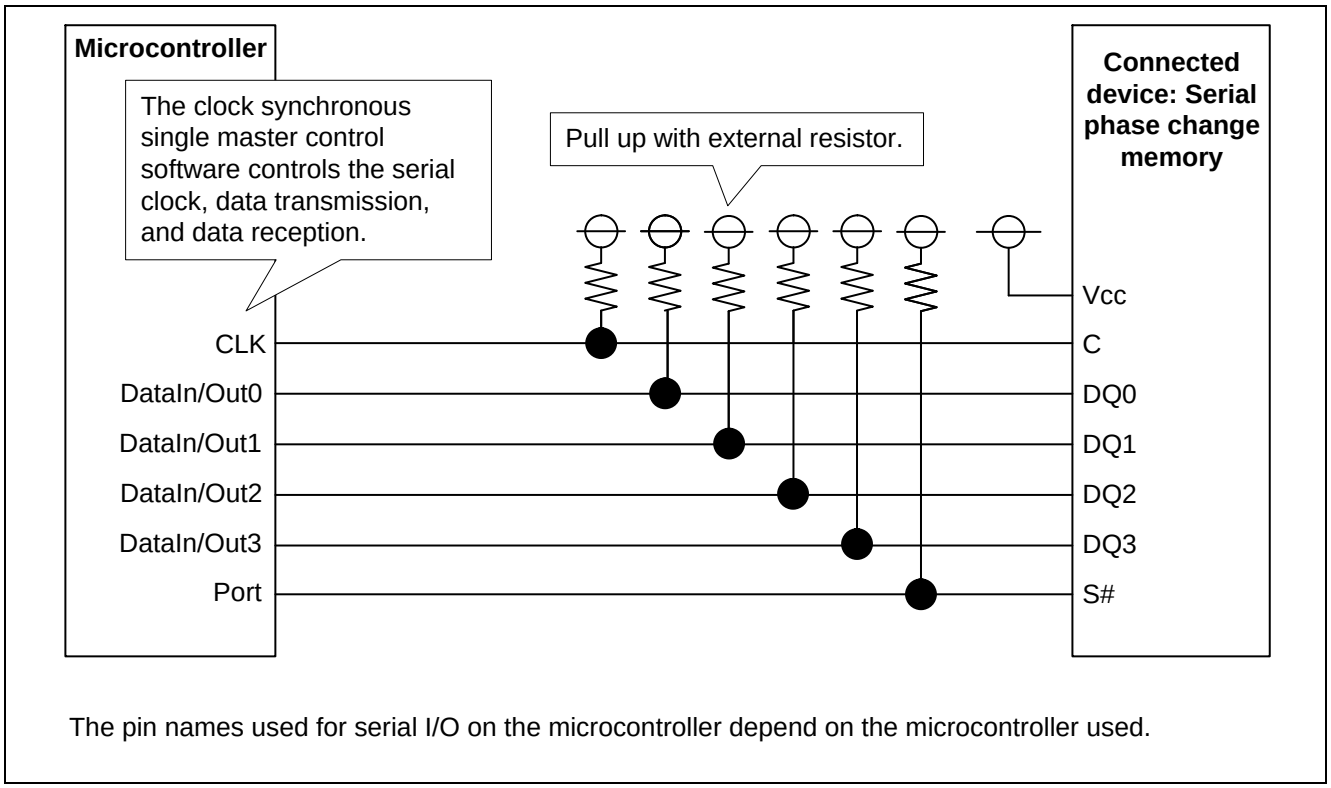


Figure 4.3 MCU and SPI Slave Device Connection Example for Quad-SPI

5. Software

5.1 Operation Overview

The MCU's clock synchronous serial communication function is used to control the serial phase change memory.

The sample code performs the following types of control:

- The S# pin of the SPI slave device is connected to the port of the MCU and is controlled by using MCU general port output. (This control is implemented by the sample code.)
- Data input and output is controlled in clock synchronous mode (using the internal clock of the MCU). (The sample code makes use of the MCU-specific clock synchronous single master control software.)

5.1.1 Relationship Between Data Buffers and Transmit/Receive Data

This sample code is a block type device driver and passes the transmit or receive data pointer as an argument. The relationship between the data ordering in the data buffer in RAM and the transmit/receive order is shown below and this sample code both transmits in the order data is stored in the transmit buffer and writes data to the receive data buffer in the order received regardless of the endian order or serial communication function used

Figure 5.1 illustrates the storage of transfer data.

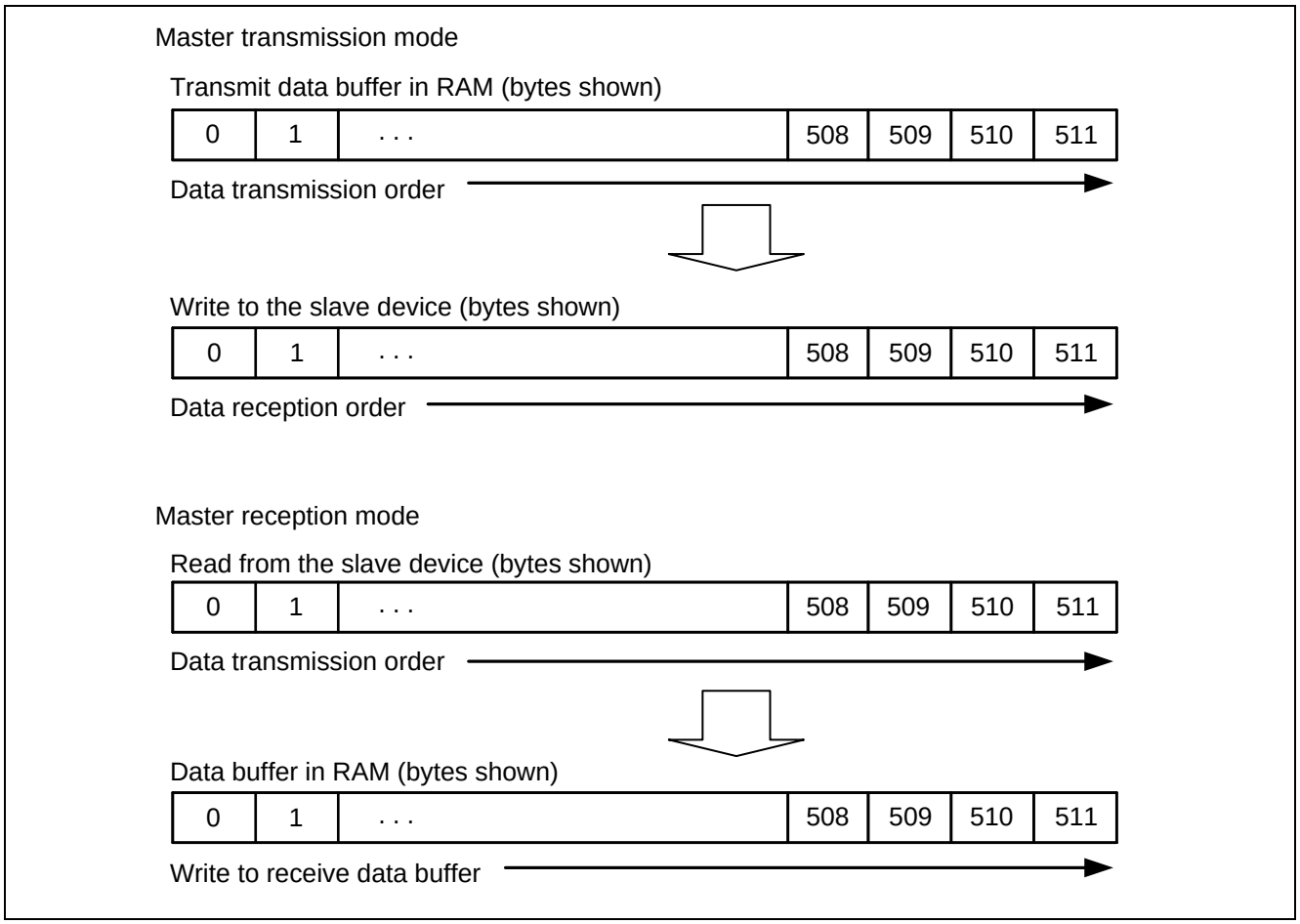


Figure 5.1 Storage of Transfer Data

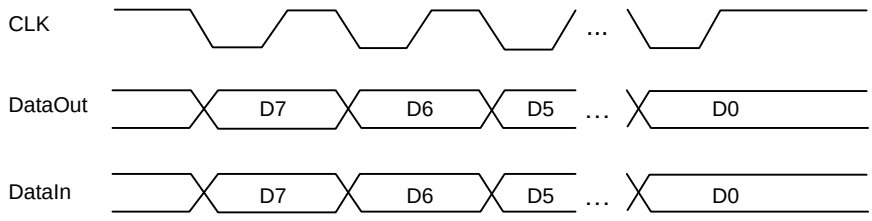
5.1.2 Timing Generation in Clock Synchronous Mode

The timings generated in clock synchronous mode are shown below.

Refer to the data sheets of the MCU and SPI device when determining the serial clock frequency to be used.

(1) **Single-SPI Operation**

To control serial phase change memory, the SPI mode 3 (CPOL = 1, CPHA = 1) shown in Figure 5.2 is generated.



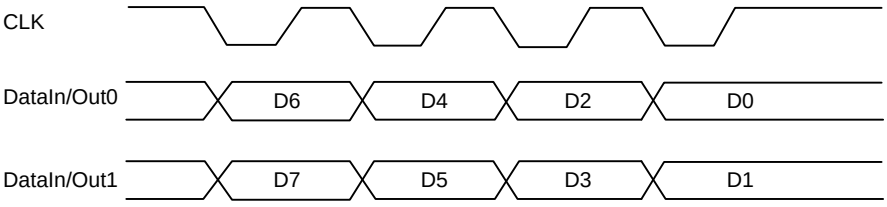
- Transmission from MCU to serial phase change memory: Output of transmit data starts at falling edge of transfer clock.
- Reception from serial phase change memory to MCU: Input of receive data starts at rising edge of transfer clock.
- MSB-first transfer

The CLK pin is high-level when no transfer is taking place.

Figure 5.2 Single-SPI Clock Synchronous Mode Timing Settings

(2) **Dual-SPI Operation**

To control serial phase change memory, the SPI mode 3 (CPOL = 1, CPHA = 1) shown in Figure 5.3 is generated.



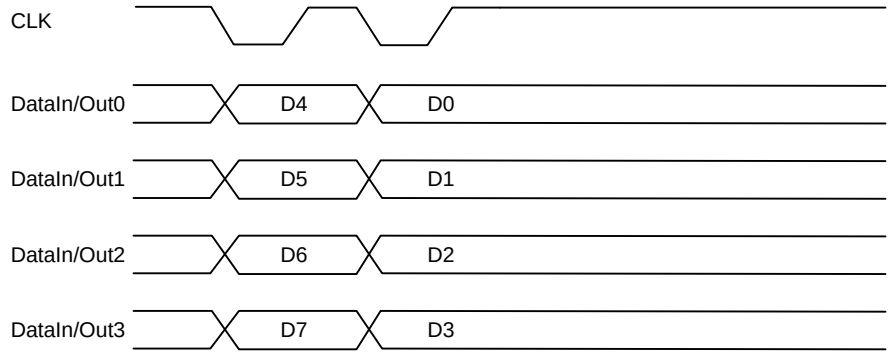
- Transmission from MCU to serial phase change memory: Output of transmit data starts at falling edge of transfer clock.
- Reception from serial phase change memory to MCU: Input of receive data starts at rising edge of transfer clock.
- MSB-first transfer

The CLK pin is high-level when no transfer is taking place.

Figure 5.3 Dual-SPI Clock Synchronous Mode Timing Settings

(3) **Quad-SPI Operation**

To control serial phase change memory, the SPI mode 3 (CPOL = 1, CPHA = 1) shown in Figure 5.4 is generated.



- Transmission from MCU to serial phase change memory: Output of transmit data starts at falling edge of transfer clock.
- Reception from serial phase change memory to MCU: Input of receive data starts at rising edge of transfer clock.
- MSB-first transfer

The CLK pin is high-level when no transfer is taking place.

Figure 5.4 Quad-SPI Clock Synchronous Mode Timing Settings

5.1.3 Serial Phase Change Memory S# Pin Control

The S# pin of the serial phase change memory is connected to the port of the MCU, and it is controlled by MCU general port output.

The duration from the falling edge of the S# (MCU port (CS#)) signal of the serial phase change memory to the falling edge of the C (MCU CLK) signal of the serial phase change memory is controlled by means of software wait to accommodate the S# setup time of the serial phase change memory.

The duration from the rising edge of the C (MCU CLK) signal of the serial phase change memory to the rising edge of the S# (MCU port (CS#)) signal of the serial phase change memory controlled by means of software wait to accommodate the S# hold time of the serial phase change memory.

Check the data sheet of the serial phase change memory and set the software wait time as appropriate for the system.

5.1.4 Serial Phase Change Memory Instruction Codes

Instruction codes are used to control the serial phase change memory, and command control is implemented by using these codes.

Table 5-1 Instruction Set

Instruction	Description	Instruction format
WREN	Write Enable	0000 0110 (06 h)
WRDI	Write Disable	0000 0100 (04 h)
RDSR	Read Status-Register	0000 0101 (05 h)
WRSR	Write Status-Register	0000 0001 (01 h)
FAST READ	Read Data Bytes at Higher Speed	0000 1011 (0b h)
READ	Read Data Bytes	0000 0011 (03 h)
DOFR	Dual output fast read	0011 1011 (3b h)
QOFR	Quad output fast read	0110 1011 (6b h)
PP	Page Program (legacy program)	0000 0010 (02 h)
	Page Program (bit-alterable write)	0010 0010 (22 h)
	Page Program (on all 1s)	1101 0001 (d1 h)
DIFP	Dual Input fast program (legacy program)	1010 0010 (a2 h)
	Dual Input fast program (bit-alterable write)	1011 0011 (d3 h)
	Dual Input fast program (on all 1s)	1101 0101 (d5 h)
QIFP	Quad Input fast program (legacy program)	0011 0010 (32 h)
	Quad Input fast program (bit-alterable write)	1101 0111 (d7 h)
	Quad Input fast program (on all 1s)	1101 1001 (d9 h)
SE	Sector Erase	1101 1000 (d8 h)
BE	Bulk Erase	1100 0111 (c7 h)
RDID	Read Identification	1001 1111 (9f h)

5.2 Software Configuration

The sample code operates as upper-layer control software for controlling the serial phase change memory (indicated as serial phase change memory control software in Figure 5.5).

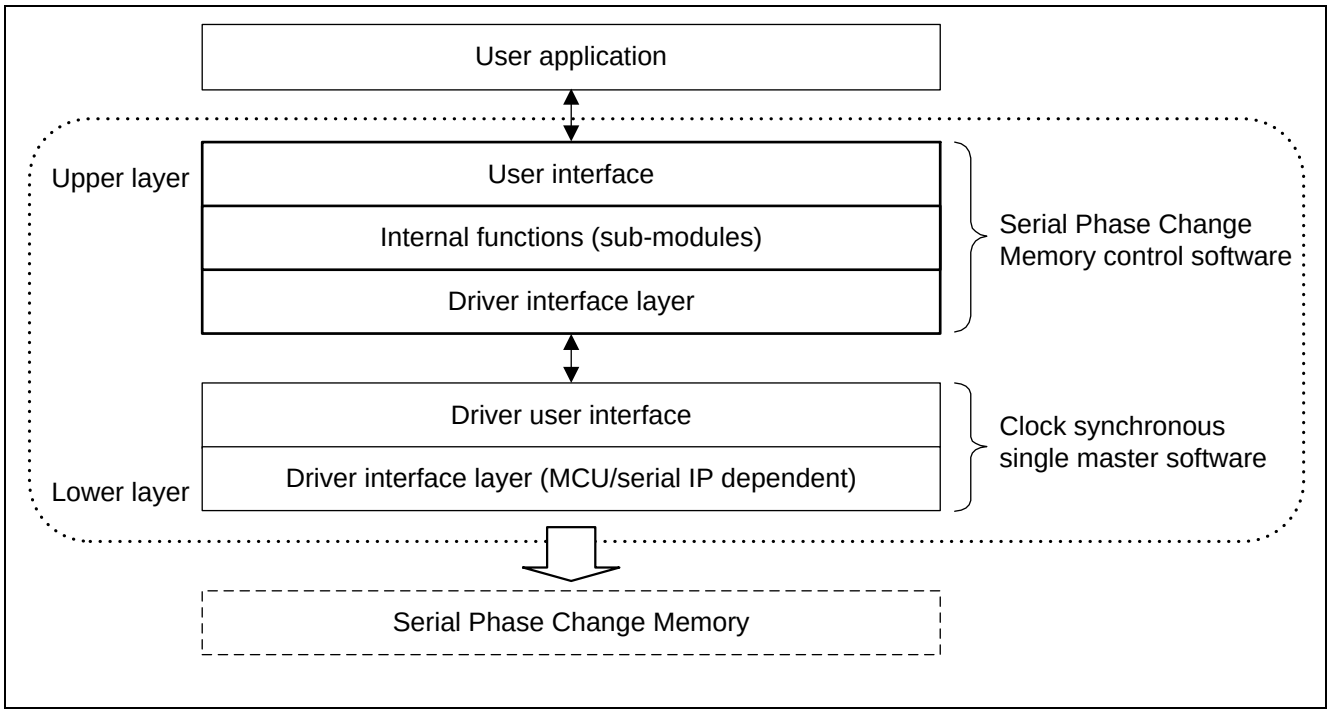


Figure 5.5 Software Configuration

The control procedure is as follows:

1. Port (CS#) signal falling edge
2. Software wait
3. Transmission and reception of commands and data using clock synchronous single master software
4. Software wait
5. Port (CS#) rising edge

5.3 Required Memory Size

The following table lists the Required Memory Size.

5.3.1 RX Family

(1) **RX63N**

Table 5-2 Required Memory Size

Memory Used	Size	Remarks
ROM	2,901 bytes (little endian)	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c
RAM	5 bytes (little endian)	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c
Maximum user stack usage	136 bytes	
Maximum interrupt stack usage	—	No interrupts used

Note: The required memory size varies depending on the C compiler version and compile options.
The indicated ROM and RAM sizes do not include the memory used by the lower-layer clock synchronous single master software.
The memory sizes listed above differ depending on the MCU type name.
The maximum usable user stack size includes the stack size of the lower-layer clock synchronous single master software.

(2) **RX111 RSPI**

Table 5-3 Required Memory Size

Memory Used	Size	Remarks
ROM	2,850 bytes (little endian)	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c
RAM	5 bytes (little endian)	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c
Maximum user stack usage	152 bytes	
Maximum interrupt stack usage	—	No interrupts used

Note: The required memory size varies depending on the C compiler version and compile options.
The indicated ROM and RAM sizes do not include the memory used by the lower-layer clock synchronous single master software.
The memory sizes listed above differ depending on the MCU type name.
The maximum usable user stack size includes the stack size of the lower-layer clock synchronous single master software.

(3) **RX111 SCI**

Table 5-4 Required Memory Size

Memory Used	Size	Remarks
ROM	2,850 bytes (little endian)	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c
RAM	5 bytes (little endian)	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c
Maximum user stack usage	148 bytes	
Maximum interrupt stack usage	—	No interrupts used

Note: The required memory size varies depending on the C compiler version and compile options.
The indicated ROM and RAM sizes do not include the memory used by the lower-layer clock synchronous single master software.
The memory sizes listed above differ depending on the MCU type name.
The maximum usable user stack size includes the stack size of the lower-layer clock synchronous single master software.

5.3.2 RL78 Family, 78K0R/Kx3-L

Show the memory size of the different MCU of the order.Check the order of the MCU and please refer to the following memory sizes.

(1) **RL78/G14 Integrated Development Environment CS+ for CA,CX (Compiler: CA78K0R)**

Table 5-5 Required Memory Size

Memory Used	Size	Remarks
ROM	5,291 bytes	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c r_qspi_pcm_p5q_sfr_rl78.c
RAM	6 bytes	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c r_qspi_pcm_p5q_sfr_rl78.c
Maximum user stack usage	110 bytes	
Maximum interrupt stack usage	—	No interrupts used

Note: The required memory size varies depending on the C compiler version and compile options.
The indicated ROM and RAM sizes do not include the memory used by the lower-layer clock synchronous single master software.
The memory sizes listed above differ depending on the MCU type name.
The maximum usable user stack size includes the stack size of the lower-layer clock synchronous single master software.

(2) **RL78/G14 Integrated Development Environment CS+ for CC (Compiler: CC-RL)**

Table 5-6 Required Memory Size

Memory Used	Size	Remarks
ROM	3,867 bytes	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c r_qspi_pcm_p5q_sfr_rl78.c
RAM	6 bytes	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c r_qspi_pcm_p5q_sfr_rl78.c
Maximum user stack usage	80 bytes	
Maximum interrupt stack usage	—	No interrupts used

Note: The required memory size varies depending on the C compiler version and compile options.
The indicated ROM and RAM sizes do not include the memory used by the lower-layer clock synchronous single master software.
The memory sizes listed above differ depending on the MCU type name.
The maximum usable user stack size includes the stack size of the lower-layer clock synchronous single master software.

(3) **RL78/G14 Integrated Development Environment IAR Embedded Workbench**

Table 5-7 Required Memory Size

Memory Used	Size	Remarks
ROM	5,534 bytes	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c r_qspi_pcm_p5q_sfr_rl78.c
RAM	6 bytes	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c r_qspi_pcm_p5q_sfr_rl78.c
Maximum user stack usage	154 bytes	
Maximum interrupt stack usage	—	No interrupts used

Note: The required memory size varies depending on the C compiler version and compile options.
The indicated ROM and RAM sizes do not include the memory used by the lower-layer clock synchronous single master software.
The memory sizes listed above differ depending on the MCU type name.
The maximum user stack size is the stack size for the entire project. It includes the stack of the lower-layer clock synchronous single-master control software.

(4) **RL78/L13 Integrated Development Environment CubeSuite+**

Table 5-8 Required Memory Size

Memory Used	Size	Remarks
ROM	4,801 bytes	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c r_qspi_pcm_p5q_sfr_rl78.c
RAM	6 bytes	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c r_qspi_pcm_p5q_sfr_rl78.c
Maximum user stack usage	102 bytes	
Maximum interrupt stack usage	—	No interrupts used

Note: The required memory size varies depending on the C compiler version and compile options.
The indicated ROM and RAM sizes do not include the memory used by the lower-layer clock synchronous single master software.
The memory sizes listed above differ depending on the MCU type name.
The maximum usable user stack size includes the stack size of the lower-layer clock synchronous single master software.

(5) **RL78/L13 Integrated Development Environment IAR Embedded Workbench**

Table 5-9 Required Memory Size

Memory Used	Size	Remarks
ROM	4,058 bytes	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c r_qspi_pcm_p5q_sfr_rl78.c
RAM	6 bytes	r_qspi_pcm_p5q_usr.c r_qspi_pcm_p5q_sub.c r_qspi_pcm_p5q_drvif.c r_qspi_pcm_p5q_sfr_rl78.c
Maximum user stack usage	120 bytes	
Maximum interrupt stack usage	—	No interrupts used

Note: The required memory size varies depending on the C compiler version and compile options.
The indicated ROM and RAM sizes do not include the memory used by the lower-layer clock synchronous single master software.
The memory sizes listed above differ depending on the MCU type name.
The maximum user stack size is the stack size for the entire project. It includes the stack of the lower-layer clock synchronous single-master control software.

5.4 File Structure

The following table lists the files used by the sample code.

Table 5-10 File Structure

\an_r01an1439ej0104_mcu_serial <DIR>	Sample code folder
r01an1439ej0104_mcu.pdf	Application note
\source <DIR>	Program folder
\r_qspi_pcm_p5q <DIR>	Serial Phase Change Memory control software folder
r_qspi_pcm_p5q.h	Header file
r_qspi_pcm_p5q_drvif.c	Driver interface source file
r_qspi_pcm_p5q_drvif.h	Driver interface header file
r_qspi_pcm_p5q_sfr.h.rl78g14	Common definition for registers (RL78/G14)
r_qspi_pcm_p5q_sfr.h.rl78g1c	Common definition for registers (RL78/G1C)
r_qspi_pcm_p5q_sfr.h.rl78l1c	Common definition for registers (RL78/L1C)
r_qspi_pcm_p5q_sfr.h.rl78l12	Common definition for registers (RL78/L12)
r_qspi_pcm_p5q_sfr.h.rl78l13	Common definition for registers (RL78/L13)
r_qspi_pcm_p5q_sfr.h.rx63n	Common definition for registers (RX63N)
r_qspi_pcm_p5q_sfr.h.rx111	Common definition for registers (RX111)
r_qspi_pcm_p5q_sfr_rl78g14.c	Common definition source file for registers (RL78/G14)
r_qspi_pcm_p5q_sfr_rl78g1c.c	Common definition source file for registers (RL78/G1C)
r_qspi_pcm_p5q_sfr_rl78l1c.c	Common definition source file for registers (RL78/L1C)
r_qspi_pcm_p5q_sfr_rl78l12.c	Common definition source file for registers (RL78/L12)
r_qspi_pcm_p5q_sfr_rl78l13.c	Common definition source file for registers (RL78/L13)
r_qspi_pcm_p5q_sub.c	Internal function source file
r_qspi_pcm_p5q_sub.h	Internal function header file
r_qspi_pcm_p5q_usr.c	User interface source file
\sample <DIR>	Operation verification program storage folder
testmain.c	Sample source file for operation verification

Note: In addition, separate MCU-specific clock synchronous single master control software is required.

5.5 Constants

5.5.1 Return Value

The following table lists the return value used in the sample code.

Table 5-11 Return Value

Constant Name	Setting Value	Contents
PCM_OK	(error_t)(0)	Successful operation
PCM_ERR_PARAM	(error_t)(–1)	Parameter error
PCM_ERR_HARD	(error_t)(–2)	Hardware error
PCM_ERR_TIMEOUT	(error_t)(–6)	Time out error
PCM_ERR_OTHER	(error_t)(–7)	Other error

5.5.2 Command Definitions

The following table lists the command definitions used in the sample code.

Table 5-12 Command Definitions (Refer to r_qspi_pcm_p5q_sub.c)

Constant Name	Setting Value	Contents
PCM_CMD_WREN	(uint8_t)(0x06)	Write Enable
PCM_CMD_WRDI	(uint8_t)(0x04)	Write Disable
PCM_CMD_RDSR	(uint8_t)(0x05)	Read Status Register
PCM_CMD_WRSR	(uint8_t)(0x01)	Write Status Register
PCM_CMD_FREAD	(uint8_t)(0x0b)	Read for Memory Array at Higher Speed
PCM_CMD_READ	(uint8_t)(0x03)	Read for Memory Array
PCM_CMD_DOFR	(uint8_t)(0x3b)	Dual Output Fast Read
PCM_CMD_QOFR	(uint8_t)(0x6b)	Quad Output Fast Read
PCM_CMD_PP_L	(uint8_t)(0x02)	Page Program (Legacy Program)
PCM_CMD_PP_BA	(uint8_t)(0x22)	Page Program (Bit-alterable Write)
PCM_CMD_PP_OA	(uint8_t)(0xd1)	Page Program (on all 1s)
PCM_CMD_DIFP_L	(uint8_t)(0xa2)	Dual Input Fast Program (Legacy Program)
PCM_CMD_DIFP_BA	(uint8_t)(0xd3)	Dual Input Fast Program (Bit-alterable Write)
PCM_CMD_DIFP_OA	(uint8_t)(0xd5)	Dual Input Fast Program (on all 1s)
PCM_CMD_QIFP_L	(uint8_t)(0x32)	Quad Input Fast Program (Legacy Program)
PCM_CMD_QIFP_BA	(uint8_t)(0xd7)	Quad Input Fast Program (Bit-alterable Write)
PCM_CMD_QIFP_OA	(uint8_t)(0xd9)	Quad Input Fast Program (on all 1s)
PCM_CMD_SE	(uint8_t)(0xd8)	Sector Erase for Memory Array
PCM_CMD_BE	(uint8_t)(0xc7)	Bulk Erase for Memory Array
PCM_CMD_RDID	(uint8_t)(0x9f)	Read Identification

5.5.3 Other Definitions

The values of other definitions used in the sample code are listed below.

Table 5-13 Values Defined in r_qspi_pcm_p5q.h

Constant Name	Setting Value	Contents
PCM_DEV_NUM	(1)	Number of connected devices
PCM_DEV0	(0)	Device number 0
PCM_DEV1	(1)	Device number 1
PCM_DELAY_TASK	(uint8_t)(1)	Wait time of delay task [unit: ms]*1
PCM_LOG_ERR	(1)	Log Type: Error
PCM_TRUE	(uint8_t)(0x01)	Flag "ON"
PCM_FALSE	(uint8_t)(0x00)	Flag "OFF"
PCM_MODE_PP_LEGACY	(uint8_t)(1)	Page Program Type: Legacy
PCM_MODE_PP_BIT_ALTERABLE	(uint8_t)(2)	Page Program Type: Bit-alterable
PCM_MODE_ON_ALL_1S	(uint8_t)(3)	Page Program Type: on all 1s (preset writes)
PCM_B_ERASE	(uint8_t)(1)	Erasure Type: Bulk Erase
PCM_S_ERASE	(uint8_t)(2)	Erasure Type: Sector Erase
PCM_MEM_SIZE	(uint32_t)(16777216)	Memory size (byte units) Value at left corresponds to size of 128 Mbit.
PCM_SECT_ADDR	(uint32_t)(0xfffe0000)	Sector address mask value for sector erase Value at left corresponds to size of 128 Mbit.
PCM_PAGE_SIZE	(uint32_t)(64)	Page size (byte units) Value at left corresponds to size of 128 Mbit.
PCM_ADDR_SIZE	(uint8_t)(3)	Address size (byte units) Value at left corresponds to size of 128 Mbit.
PCM_WP_WHOLE_MEM	(uint8_t)(0x1f)	Whole-chip write protect

Note: 1. The delay task for OS control. The OS control used in the sample code assumes µITRON 4.0.

Table 5-14 Values Defined in r_qspi_pcm_p5q_sfr.h.rx63n

Constant Name	Setting Value	Contents
PCM_DR_CS0	PORTA.PODR.BIT.B0	Device number 0 port output data register SFR definition
PCM_DDR_CS	PORTA.PDR.BIT.B0	Device number 0 port direction register SFR definition
PCM_DR_CS1	—	Device number 1 port output data register SFR definition (This setting is needed when controlling two devices.)
PCM_DDR_CS1	—	Device number 1 port direction register SFR definition (This setting is needed when controlling two devices.)
PCM_HI	(uint8_t)(0x01)	Port "H"
PCM_LOW	(uint8_t)(0x00)	Port "L"
PCM_OUT	(uint8_t)(0x01)	Port Output Setting
PCM_IN	(uint8_t)(0x00)	Port Input Setting
PCM_BR	(uint8_t)(0x01)	Transfer rate for command transmission*1
PCM_BR_WRITE_DATA	(uint8_t)(0x01)	Transfer rate for data transmission*1
PCM_BR_READ_DATA	(uint8_t)(0x01)	Transfer rate for data reception*1

Note: 1. This value is set in the RSPI bit rate register (SPBR) when using the clock synchronous single master control software with the RSPI. The value shown is for a peripheral module clock setting of 48 [MHz] and a transfer rate of 12 [MHz].
This value is set in the bit rate register (BRR) when using the clock synchronous single master control software with SCI. The value shown is for a peripheral module clock setting of 48 [MHz] and a transfer rate of 6 [MHz].

Table 5-15 Values Defined in r_qspi_pcm_p5q_sfr.h.rl78

Constant Name	Setting Value	Contents
PCM_DR_CS0	P8.0	Device number 0 port register SFR definition
PCM_DDR_CS	PM8.0	Device number 0 port mode register SFR definition
PCM_DR_CS1	—	Device number 1 port output data register SFR definition (This setting is needed when controlling two devices.)
PCM_DDR_CS1	—	Device number 1 port direction register SFR definition (This setting is needed when controlling two devices.)
PCM_HI	(uint8_t)(0x01)	Port "H"
PCM_LOW	(uint8_t)(0x00)	Port "L"
PCM_OUT	(uint8_t)(0x00)	Port Output Setting
PCM_IN	(uint8_t)(0x01)	Port Input Setting
PCM_BR	(uint8_t)(0x01)	Transfer rate for command transmission*1
PCM_BR_WRITE_DATA	(uint8_t)(0x01)	Transfer rate for data transmission*1
PCM_BR_READ_DATA	(uint8_t)(0x01)	Transfer rate for data reception*1

Note: 1. This value is set in bits 15 to 9 of the serial data register (SDR) when using the clock synchronous single master control software in the serial array unit CSI mode. The sample code uses this value with an operation clock setting of 24 [MHz] and a transfer rate or 6 [MHz].

Table 5-16 Values Defined in r_qspi_pcm_p5q_sub.c

Constant Name	Setting Value	Contents
PCM_SHORT_SIZE	(uint32_t)(0x00008000)	Maximum transfer size setting for low-level functions (max.: 32 KB)

Table 5-17 Values Defined in r_qspi_pcm_p5q_sub.h

Constant Name	Setting Value	Contents
PCM_BE_BUSY_WAIT	(uint32_t)(100000)	Bulk Erase Busy Timeout 100000 × 1 ms = 100 s
PCM_SE_BUSY_WAIT	(uint32_t)(800)	Sector Erase Busy Timeout 800 × 1 ms = 800 ms
PCM_WBUSY_WAIT	(uint32_t)(400)	Write Ready Timeout 400 × 1 µs = 400 µs
PCM_T_WBUSY_WAIT	(uint16_t)MTL_T_1US	Write Busy Polling Time
PCM_T_EBUSY_WAIT	(uint16_t)MTL_T_1MS	Erase Busy Polling Time
PCM_T_CS_HOLD	(uint16_t)MTL_T_1US	CS Stability Waiting Time
PCM_T_R_ACCESS	(uint16_t)MTL_T_1US	Reading Start Waiting Time
PCM_REG_SRWD	(uint8_t)(0x80)	Status Register Write Disable
PCM_REG_BP3	(uint8_t)(0x40)	Block Protection Bit3
PCM_REG_TB	(uint8_t)(0x20)	Top/Bottom Bit
PCM_REG_BP2	(uint8_t)(0x10)	Block Protection Bit2
PCM_REG_BP1	(uint8_t)(0x08)	Block Protection Bit1
PCM_REG_BP0	(uint8_t)(0x04)	Block Protection Bit0
PCM_REG_WEL	(uint8_t)(0x02)	Write Enable Latch Bit
PCM_REG_WIP	(uint8_t)(0x01)	Write In Progress Bit
PCM_REG	(uint8_t)(0xfc)	Write status fixed data

5.6 Structure/Union List

Show the Structure/Union Used in the Sample Code.

```
typedef union {
   uint32_t          ul;
   uint8_t           uc[4];
} PCM_EXCHG_LONG;              /* total 4bytes                                    */
```

Figure 5.6 Union Used in the Sample Code (Refer to r_qspi_pcm_p5q_sub.c)

```
typedef struct
{
   uint32_t          Addr        /* Address to issue a command                   */
   uint32_t          Cnt         /* Number of bytes to be read/written           */
   uint16_t          DataCnt;    /* Temporary counter or Number of bytes to be written in a page */
   uint8_t           rsv[2];     /* Reserved                                     */
   uint8_t FAR*      pData;      /* Data storage buffer pointer                  */
} r_qspi_pcm_info_t;
```

Figure 5.7 Structure Used in the Sample Code (Refer to r_qspi_pcm_p5q.h)

Table 5-18 Description of Structure "r_qspi_pcm_info_t"

Structure Member	Allowable Setting Range	Description
Addr	0000 0000h to FFFF FFFFh	Write/read start address
Cnt	0000 0000h to FFFF FFFFh	Write/read data counter (byte units)
DataCnt	(Setting prohibited.)	Write: Write data counter temp. (max. 1 page) Read: Read data counter temp. (max. 32 KB)
rsv[2]	(Setting has no effect.)	For alignment adjustment
pData	—	Data storage buffer pointer Write: Storage source of data to be written in serial phase change memory Read: Storage destination of data to be read from serial phase change memory

5.7 Variable

The following table lists the Static Variable.

Table 5-19 Static Variable (Refer to r_qspi_flash_p5q_sub.c)

Type	Variable Name	Contents	Function Used
STATIC uint8_t	g_pcm_cmdbuf[5]	Command buffer	r_qspi_pcm_send_cmd r_qspi_pcm_set_cmd

5.8 Functions

The following table lists the Functions.

Table 5-20 Functions

Function Name	Outline
R_QSPI_PCM_Init_Driver()	Driver initialization processing
R_QSPI_PCM_Read_Status()	Status register read processing
R_QSPI_PCM_Set_Write_Protect()	Write protect setting processing
R_QSPI_PCM_Write_Di()	WRDI command issue processing
R_QSPI_PCM_Read_Data()	Data read processing
R_QSPI_PCM_Write_Data()	Data write processing
R_QSPI_PCM_Write_Data_Page()	Data write processing (for single-page write)
R_QSPI_PCM_Erase()	Erase processing
R_QSPI_PCM_ReadID()	ID read processing
R_QSPI_PCM_Wait()	Busy wait processing

On cache-equipped MCUs, specify a non-cached area as the location of the read/write data storage buffer.

The read/write data storage buffer address is dependent on the lower-layer MCU-specific clock synchronous single master control software, and in some cases it is necessary to specify an address on a 4-byte boundary. For details, refer to the application note for the MCU-specific clock synchronous single master control software.

5.9 Function Specifications

The following tables list the sample code function specifications.

5.9.1 Driver Initialization Processing

R_QSPI_PCM_Init_Driver	
Outline	Driver initialization processing
Header	r_qspi_pcm_p5q.h, r_qspi_pcm_p5q_sub.h, r_qspi_pcm_sfr.h, r_qspi_pcm_drvif.h
Declaration	error_t R_QSPI_PCM_Init_Driver(void)
Description	• Calls the R_QSPI_PCM_Init_Port() function to initialize the CS# pin. • Calls the initialization function of the clock synchronous single master control software to initialize the I/O ports. • Call this function once at system startup
Arguments	None
Return Value	The initialization result is returned. PCM_OK ; Successful operation PCM_ERR_OTHER ; Other error

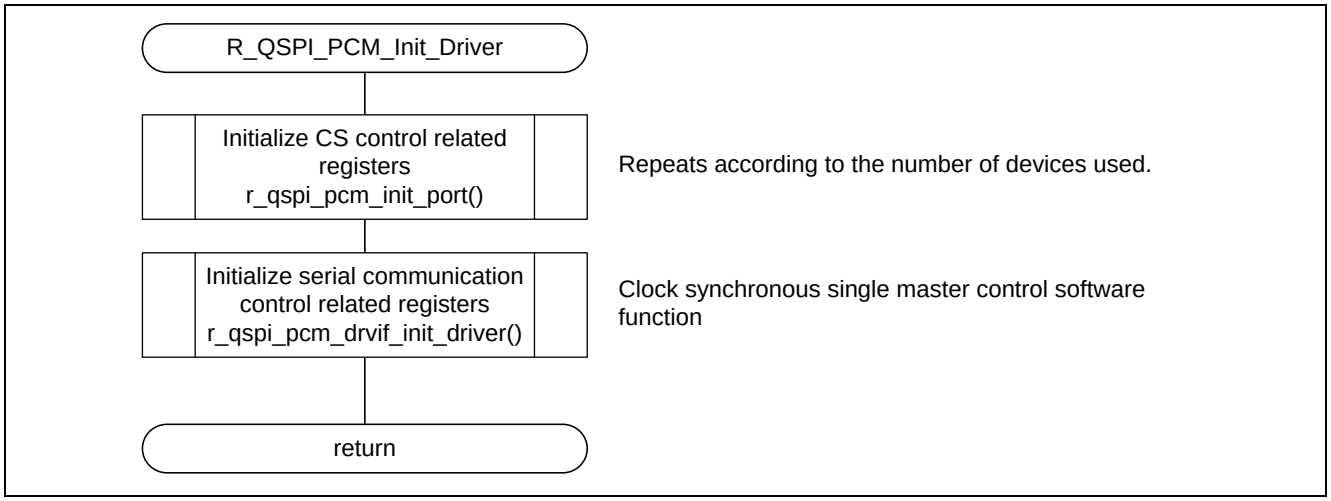


Figure 5.8 Overview of Driver Initialization Processing

5.9.2 Status Register Read Processing

R_QSPI_PCM_Read_Status

Outline Status register read processing

Header r_qspi_pcm_p5q.h, r_qspi_pcm_p5q_sub.h, r_qspi_pcm_sfr.h, r_qspi_pcm_drvif.h

Declaration error_t R_QSPI_PCM_Read_Status(uint8_t DevNo, uint8_t FAR* pStatus)

Description

- Reads the status register and stores the result in pStatus.
  Set 1 byte as a read buffer.
- Stores the following information in the read status storage buffer (pStatus):
  Bit 7:SRWD
  1: TB, BP3, BP2, BP1, BP0 are read-only bits
  0: TB, BP3, BP2, BP1, BP0 are read/writable
  Bits 6 to 2: BP3, TB, BP2, BP1, BP0
  Bit 1:WEL
  1: Internal Write Enable Latch is set
  0: Internal Write Enable Latch is reset
  Bit 0:WIP
  1: Program or Erase cycle is in progress
  0: No Program or Erase cycle is in progress
- Refer to the data sheet of the serial phase change memory for the relationship between protect areas and protect bits. It is possible that the BP bits may not be allocated.

Arguments

uint8_t	DevNo	; Device number
uint8_t FAR*	pStatus	; Read status storage buffer pointer

Return Value The status register fetch result is returned.

PCM_OK	; Successful operation
PCM_ERR_PARAM	; Parameter error
PCM_ERR_HARD	; Hardware error
PCM_ERR_OTHER	; Other error

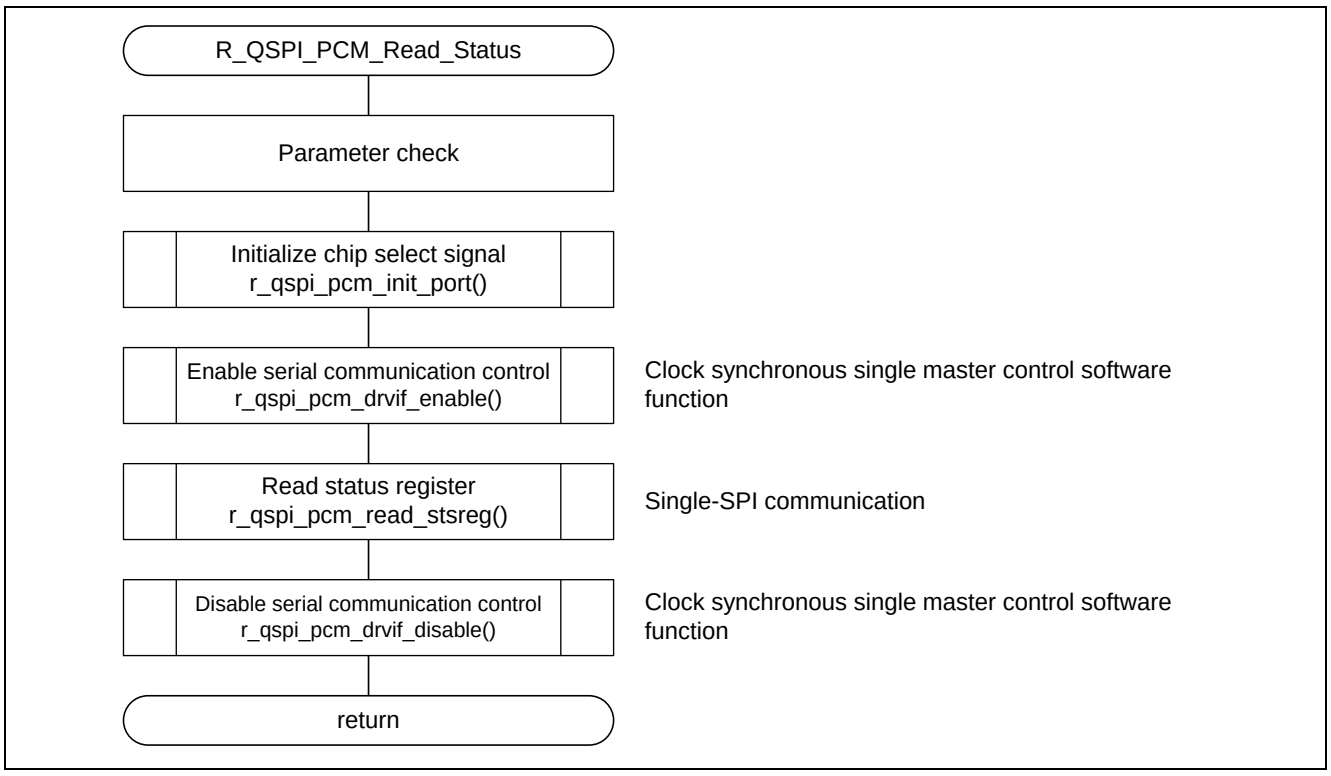


Figure 5.9 Overview of Status Register Read Processing

5.9.3 Write Protect Setting Processing

R_QSPI_PCM_Set_Write_Protect

Outline Write protect setting processing

Header r_qspi_pcm_p5q.h, r_qspi_pcm_p5q_sub.h, r_qspi_pcm_sfr.h, r_qspi_pcm_drvif.h

Declaration error_t R_QSPI_PCM_Set_Write_Protect(uint8_t DevNo, uint8_t WpSts)

Description

- Due to the usage limitations*[1] of 32 Mb, 64 Mb, and 128 Mb serial phase change memory, it is not possible to set the write protect bit to 1. When using this user API, make sure to check the product information contained in the latest data sheet of the serial phase change memory.
  Note 1. Refer to the information on usage limitations in the following documents: 32Mb/64Mb P5Q Serial Phase Change Memory Errata Rev. A, 128Mb P5Q Serial Phase Change Memory Errata Rev. D
- Makes write protect settings
- Make settings using the following write protect setting data (WpSts):

WpSts	BP3	TB1	BP2	BP1	BP0	Protected Area
0x00	0	0	0	0	0	None
0x01	0	0	0	0	1	Upper 32nd (sector 31)
0x02	0	0	0	1	0	Upper 16th (sectors 30 to 31)
0x03	0	0	0	1	1	Upper 8th (sectors 28 to 31)
0x04	0	0	1	0	0	Upper 4th (sectors 24 to 31)
0x05	0	0	1	0	1	Upper half (sectors 16 to 31)
0x06	0	0	1	1	0	All sectors (sectors 0 to 31)
0x07	0	0	1	1	1	All sectors (sectors 0 to 31)
0x10 – 0x16	1	0	0/1	0/1	0/1	Setting prohibited for this user API.
0x17	1	0	1	1	1	All sectors (sectors 0 to 31)
0x08	0	1	0	0	0	None
0x09	0	1	0	0	1	Lower 32nd (sector 0)
0x0a	0	1	0	1	0	Lower 16th (sectors 0 to 1)
0x0b	0	1	0	1	1	Lower 8th (sectors 0 to 3)
0x0c	0	1	1	0	0	Lower 4th (sectors 0 to 7)
0x0d	0	1	1	0	1	Lower half (sectors 0 to 15)
0x0e	0	1	1	1	0	All sectors (sectors 0 to 31)
0x0f	0	1	1	1	1	All sectors (sectors 0 to 31)
0x18 – 0x1e	1	1	0/1	0/1	0/1	Setting prohibited for this user API.
0x1f	1	1	1	1	1	All sectors (sectors 0 to 31)

- Clears SRWD to 0.
- Refer to the data sheet of the serial phase change memory for the relationship between protect areas and protect bits. It is possible that the BP bits may not be allocated.
- There are two ways to wait for write completion. These are described below. Note that the next processing task (write, read, erase, etc.) should be executed after confirming write completion.
- To use the user API to wait for write completion, enable PCM_WAIT_READY in r_qspi_pcm_p5q.h.
- To wait for write completion without using the user API, disable PCM_WAIT_READY in r_qspi_pcm_p5q.h and call R_QSPI_PCM_Wait() after processing by the user API finishes. This processing method allows the use of a user-defined duration when waiting for write completion. Refer to figure 5.11 for the usage method.

Arguments	uint8_t	DevNo	; Device number
	uint8_t	WpSts	; Write protect setting data
Return Value	The write protect setting result is returned.		
	PCM_OK		; Successful operation
	PCM_ERR_PARAM		; Parameter error
	PCM_ERR_HARD		; Hardware error
	PCM_ERR_TIMEOUT		; Time out error (PCM_WAIT_READY enabled)
	PCM_ERR_OTHER		; Other error

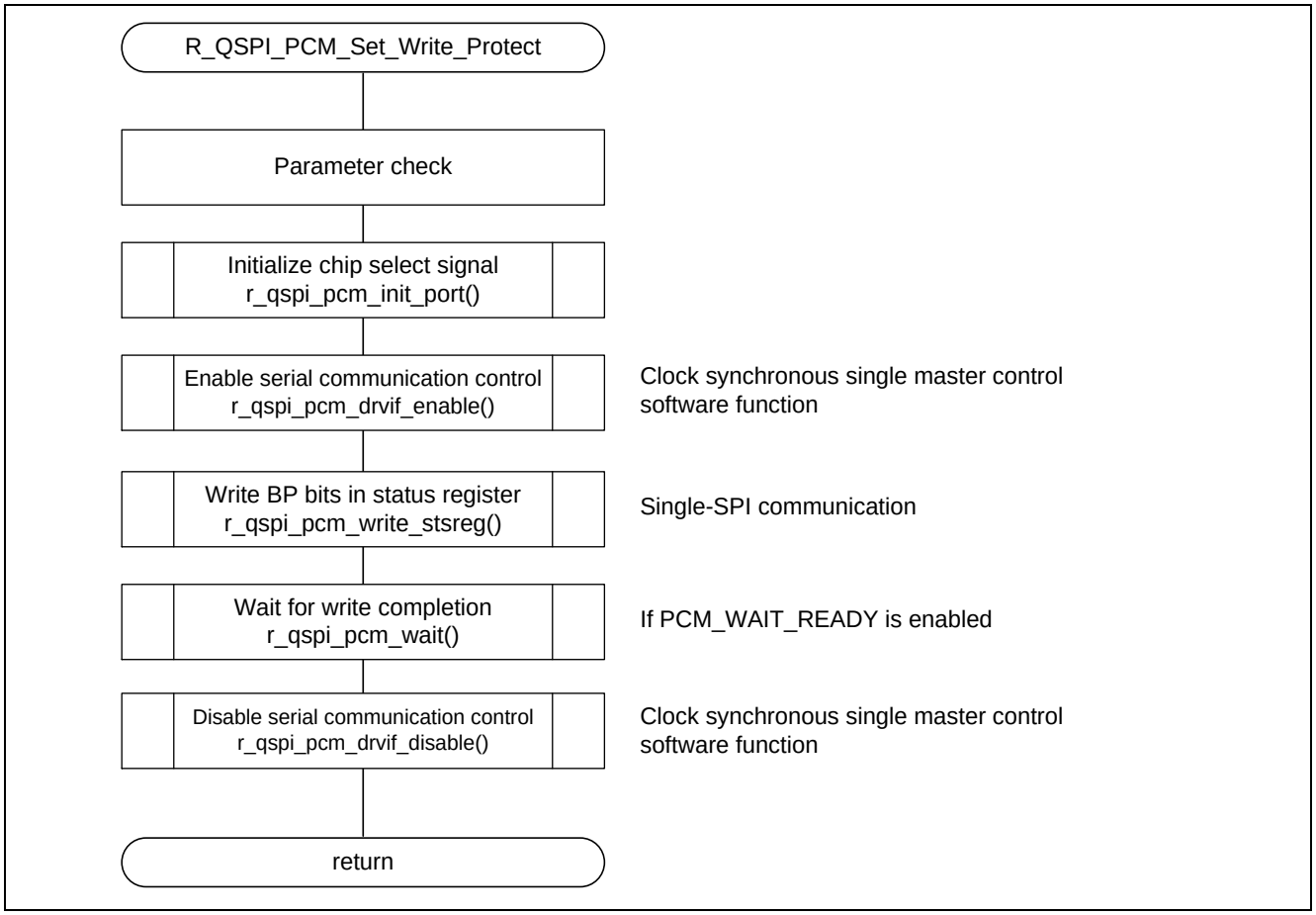


Figure 5.10 Overview of Write Protect Setting Processing

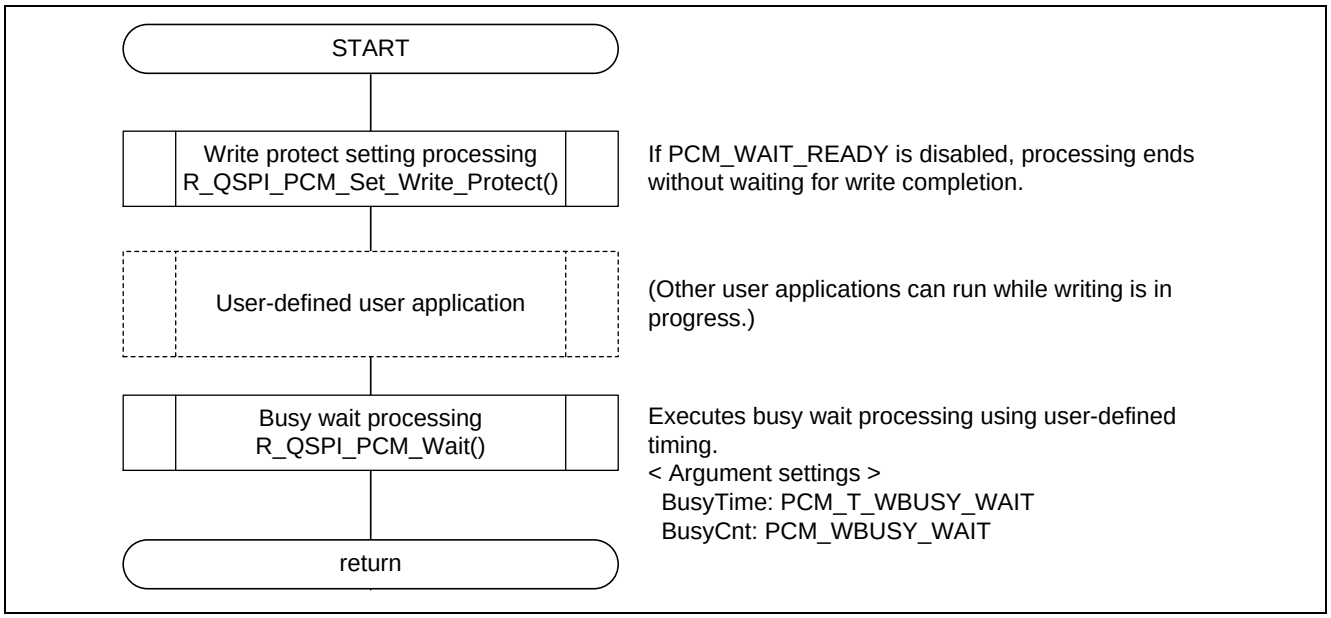


Figure 5.11 Using R_QSPI_PCM_Wait() to Wait for Write Protect Setting Completion

5.9.4 WRDI Command Issue Processing

R_QSPI_PCM_Write_Di

Outline	WRDI command issue processing
Header	r_qspi_pcm_p5q.h, r_qspi_pcm_p5q_sub.h, r_qspi_pcm_p5q_sfr.h, r_qspi_pcm_p5q_drvif.h
Declaration	error_t R_QSPI_PCM_Write_Di(uint8_t DevNo)
Description	• Clears the WEL bit in the status register. • When a programming error or erase error occurs, this function must be called to clear the WEL bit.
Arguments	uint8_t DevNo ; Device number
Return Value	The clearing result is returned. PCM_OK ; Successful operation PCM _ERR_PARAM ; Parameter error PCM _ERR_HARD ; Hardware error PCM _ERR_OTHER ; Other error

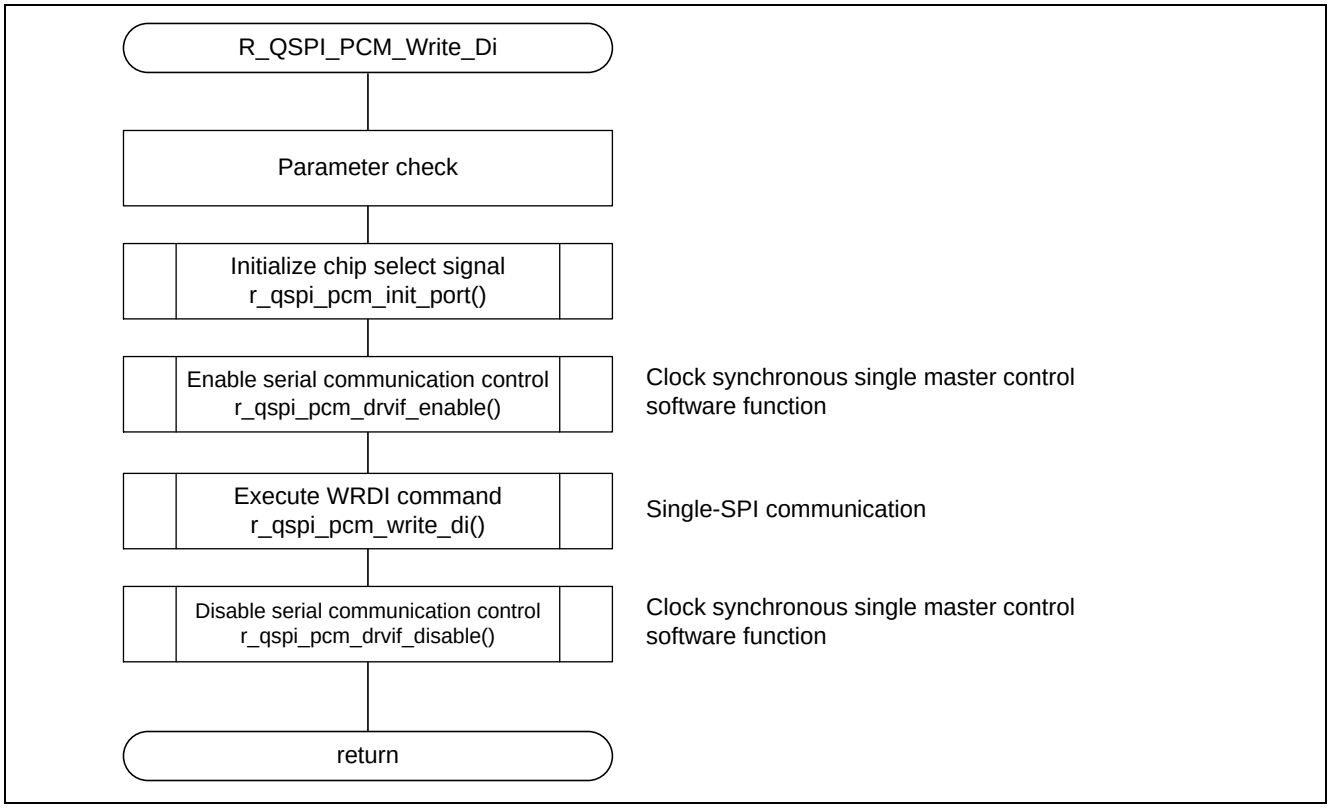


Figure 5.12 Overview of WRDI Command Issue Processing

5.9.5 Data Read Processing

R_QSPI_PCM_Read_Data

Outline Data read processing

Header r_qspi_pcm_p5q.h, r_qspi_pcm_p5q_sub.h, r_qspi_pcm_sfr.h, r_qspi_pcm_drvif.h

Declaration error_t R_QSPI_PCM_Read_Data(uint8_t DevNo, r_qspi_pcm_info_t FAR* pPcm_Info)

Description

- Reads the specified number of bytes of data from the specified address in the serial phase change memory, and stores it in pData.
- The final read address is equal to the serial phase change memory capacity – 1.
- It is not possible to continue reading by means of a rollover. After reading the final address, end processing once and then call the user API again after specifying a new address.
- Due to the usage limitations*[1] of 32 Mb and 64 Mb serial phase change memory, it is not possible to continue reading from the start address because the address is not rolled over at the next read operation.

  Note: 1. Refer to the information on usage limitations in the following document: 32Mb/64Mb P5Q Serial Phase Change Memory Errata Rev. A.

Arguments

```
uint8_t              DevNo          ; Device number
r_qspi_pcm_info_t FAR* pPcm_Info    ; PCM communication information structure
        uint32_t     Addr           ; Read start address
        uint32_t     Cnt            ; Read byte count
        uint16_t     DataCnt        ; Read byte temp. (setting prohibited)
        uint8_t FAR* pData          ; Read data storage buffer pointer
```

Return Value The read result is returned.

```
PCM_OK              ; Successful operation
PCM_ERR_PARAM       ; Parameter error
PCM_ERR_HARD        ; Hardware error
PCM_ERR_OTHER       ; Other error
```

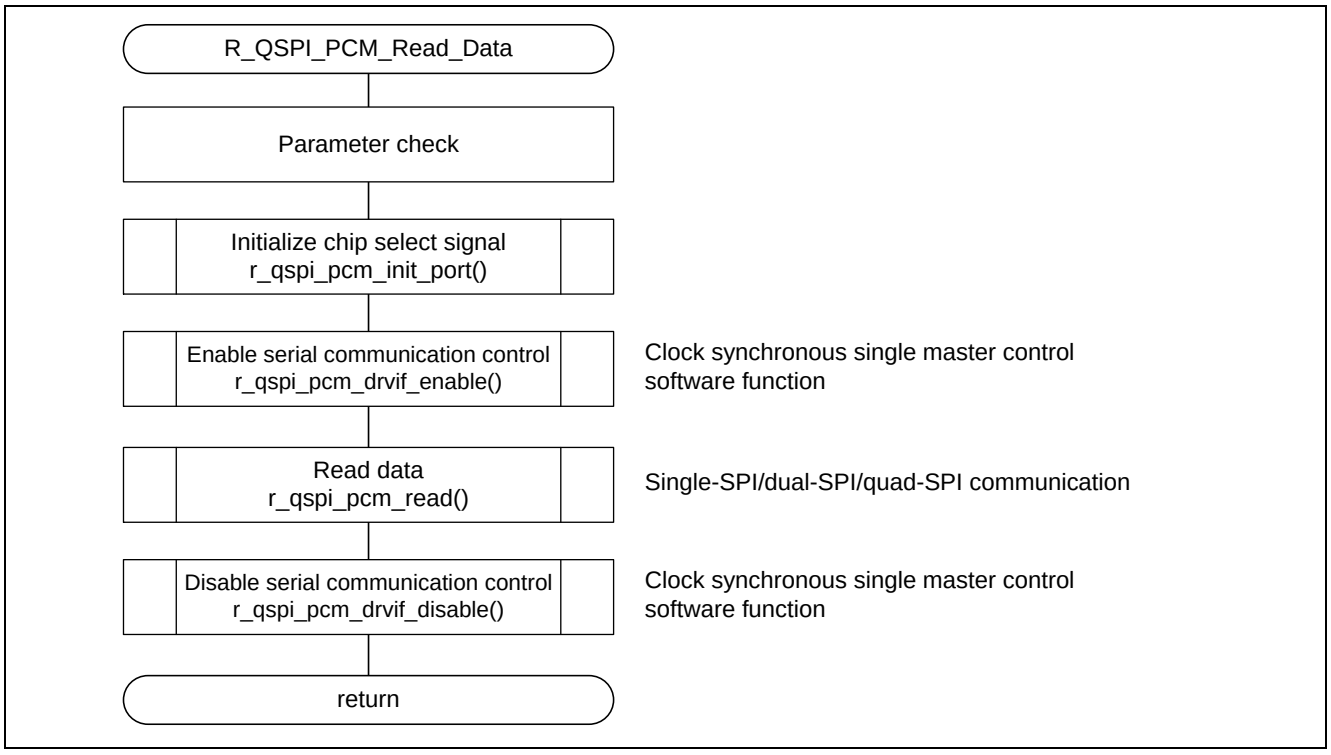


Figure 5.13 Overview of Data Read Processing

5.9.6 Data Write Processing

R_QSPI_PCM_Write_Data

Outline Data write processing

Header r_qspi_pcm_p5q.h, r_qspi_pcm_p5q_sub.h, r_qspi_pcm_sfr.h, r_qspi_pcm_drvif.h

Declaration error_t R_QSPI_PCM_Write_Data(uint8_t DevNo, r_qspi_pcm_info_t FAR* pPcm_Info, uint8_t Mode)

Description

- Writes the specified number of bytes of the data in pData to the specified address in the serial phase change memory.
- Writes using legacy program, bit-alterable write, or on all 1s, according to the Mode setting.
- Writing to the serial phase change memory can only be performed to areas with write protect disabled. Also, no error is returned. The WEL bit is in the set state.
- It is not possible to write to areas where protect is enabled. Also, no error is returned. The WEL bit is in the set state.
- The final write address is equal to the serial phase change memory capacity – 1.
- The maximum value that can be set for the write byte count (Cnt) is equal to the serial phase change memory capacity.
- The user API performs a wait for write completion regardless of the setting of PCM_WAIT_READY in r_qspi_pcm_p5q.h.

Arguments

```
uint8_t             DevNo                   ; Device number
r_qspi_pcm_info_t FAR* pPcm_Info            ; PCM communication information structure
        uint32_t        Addr                ; Write start address
        uint32_t        Cnt                 ; Write byte count
        uint16_t        DataCnt             ; Write byte temp. (setting prohibited)
        uint8_t FAR*    pData               ; Write data storage buffer pointer
uint8_t             Mode                    ; Write mode (selectable from the following):
                                            ;     PCM_MODE_PP_LEGACY
                                            ;     PCM_MODE_PP_BIT_ALTERABLE
                                            ;     PCM_MODE_PP_ON_ALL_1S
```

Return Value The read result is returned.

```
PCM_OK                  ; Successful operation
PCM_ERR_PARAM           ; Parameter error
PCM_ERR_HARD            ; Hardware error
PCM_ERR_TIMEOUT         ; Time out error
PCM_ERR_OTHER           ; Other error
```

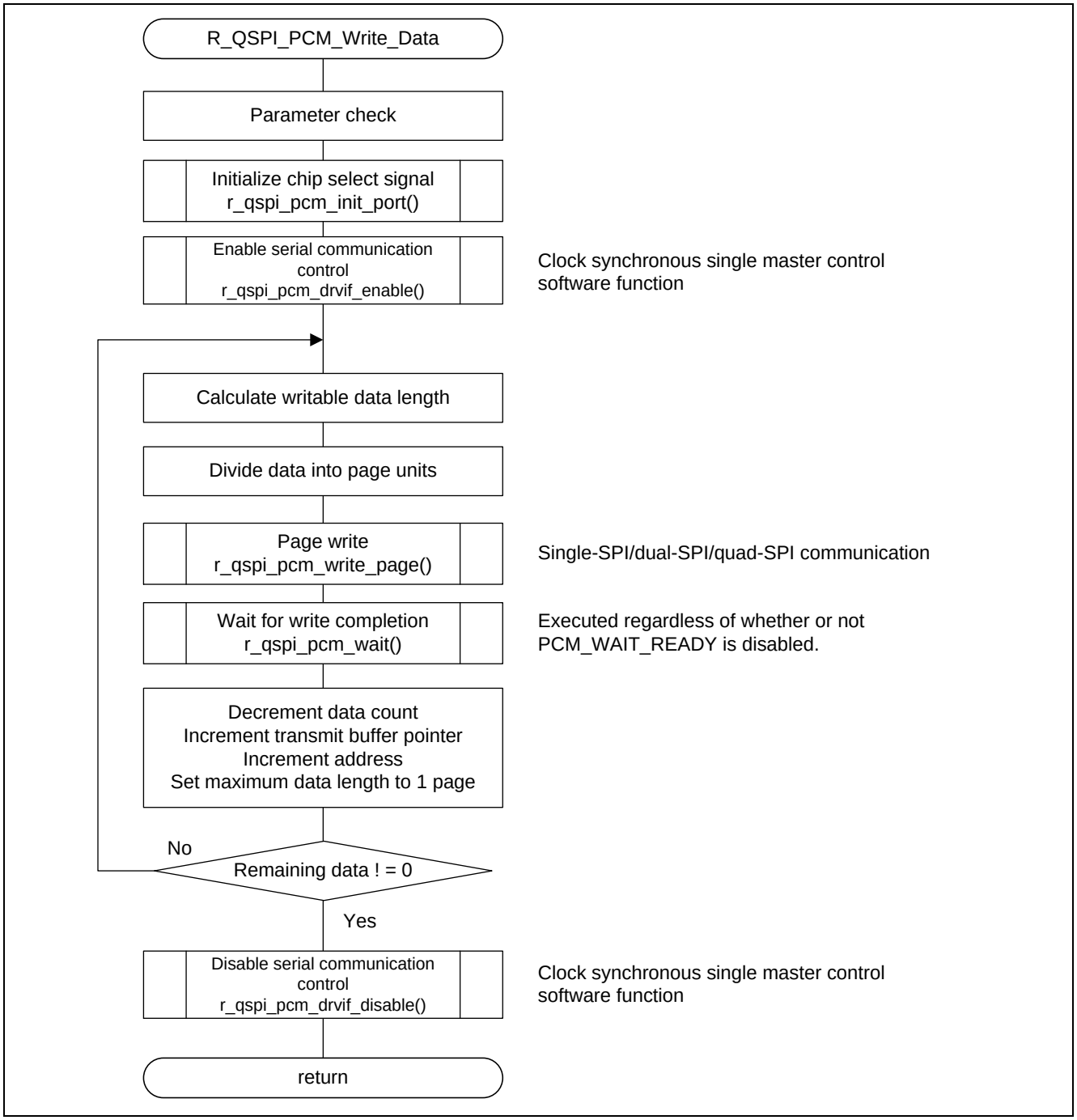


Figure 5.14 Overview of Data Write Processing

5.9.7 Data Write Processing (for Single-Page Write)

R_QSPI_PCM_Write_Data_Page

Outline Data write processing (for single-page write)

Header r_qspi_pcm_p5q.h, r_qspi_pcm_p5q_sub.h, r_qspi_pcm_sfr.h, r_qspi_pcm_drvif.h

Declaration error_t R_QSPI_PCM_Write_Data_Page(uint8_t DevNo, r_qspi_pcm_info_t FAR* pPcm_Info, uint8_t Mode)

Description

- Writes the specified number of bytes (maximum: 1 page) of the data in pData to the specified address in the serial phase change memory.
- Writes using legacy program, bit-alterable write, or on all 1s, according to the Mode setting.
- It is possible to avoid situations in which other processing cannot be performed during data communication because the communication is divided into page units when writing large volumes of data.
- Writing to the serial phase change memory can only be performed to areas with write protect disabled. Also, no error is returned. The WEL bit is in the set state.
- It is not possible to write to areas where protect is enabled. Also, no error is returned. The WEL bit is in the set state.
- The final write address is equal to the serial phase change memory capacity – 1.
- The maximum value that can be set for the write byte count (Cnt) is equal to the serial phase change memory capacity.
- Even if a byte count that exceeds one page is specified, the remaining byte count and the next address information remains in the PCM communication information structure (pPcm_Info) after write processing of one page finishes. It is possible to write the remaining byte count by setting pPcm_Info once again without modification.
- There are two ways to wait for write completion. These are described below. Note that the next processing task (write, read, erase, etc.) should be executed after confirming write completion.
- To use the user API to wait for write completion, enable PCM_WAIT_READY in r_qspi_pcm_p5q.h.
- To wait for write completion without using the user API, disable PCM_WAIT_READY in r_qspi_pcm_p5q.h and call R_QSPI_PCM_Wait() after processing by the user API finishes. This processing method allows the use of a user-defined duration when waiting for write completion. Refer to figure 5.16 for the usage method.

Arguments

```
uint8_t             DevNo                ; Device number
r_qspi_pcm_info_t FAR* pPcm_Info         ; PCM communication information structure
        uint32_t        Addr             ; Write start address
        uint32_t        Cnt              ; Write byte count
        uint16_t        DataCnt          ; Write byte temp. (setting prohibited)
        uint8_t FAR*    pData            ; Write data storage buffer pointer
uint8_t             Mode                 ; Write mode (selectable from the following):
                                         ;     PCM_MODE_PP_LEGACY
                                         ;     PCM_MODE_PP_BIT_ALTERABLE
                                         ;     PCM_MODE_PP_ON_ALL_1S
```

Return Value The read result is returned.

```
PCM_OK                  ; Successful operation
PCM_ERR_PARAM           ; Parameter error
PCM_ERR_HARD            ; Hardware error
PCM_ERR_TIMEOUT         ; Time out error (PCM_WAIT_READY enabled)
PCM_ERR_OTHER           ; Other error
```

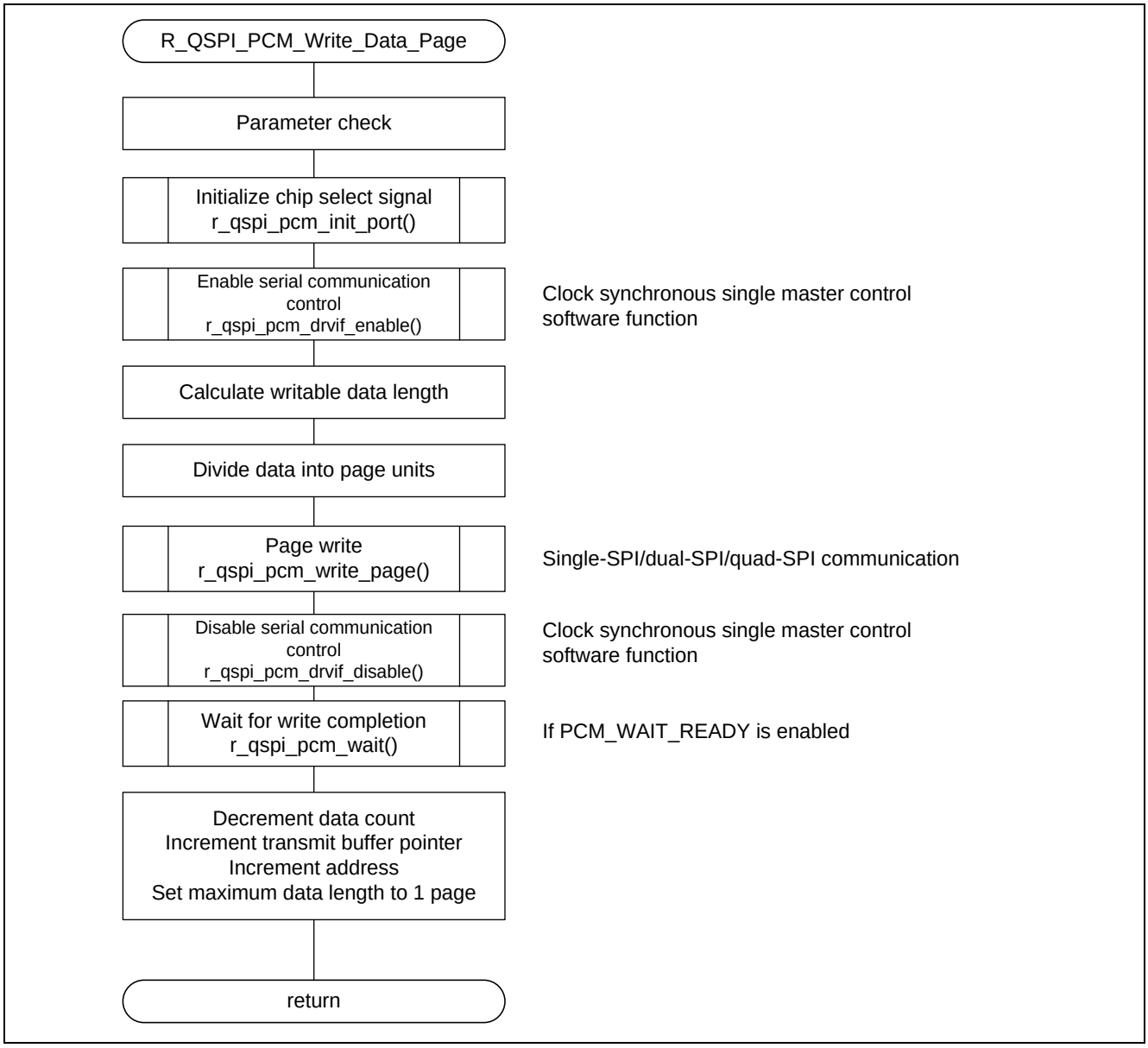


Figure 5.15 Overview of Data Write Processing (for Single-Page Write)

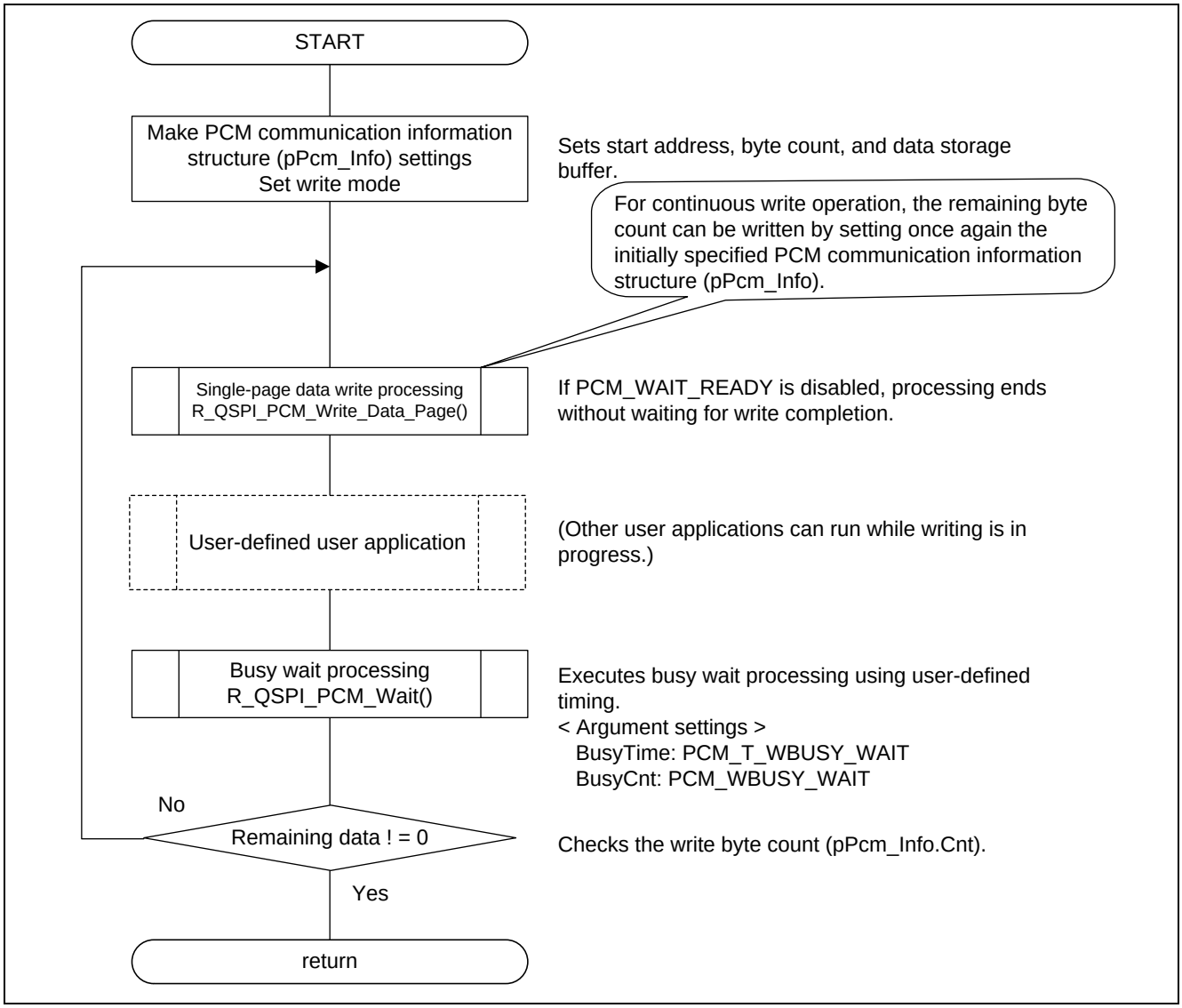


Figure 5.16 Using R_QSPI_PCM_Wait() to Wait for Data Write Processing (for Single-Page Write) Completion

5.9.8 Erase Processing

R_QSPI_PCM_Erase

Outline Erase processing

Header r_qspi_pcm_p5q.h, r_qspi_pcm_p5q_sub.h, r_qspi_pcm_sfr.h, r_qspi_pcm_drvif.h

Declaration error_t R_QSPI_PCM_Erase(uint8_t DevNo, uint32_t Addr, uint8_t Mode)

Description

- Erases all the data in the memory (bulk erase) or all the data in a specified sector (sector erase).
- Bulk erase or sector erase may be selected by using the Mode setting.
- For bulk erase, set Addr to 0x00000000.
- Erasing the serial phase change memory can only be performed on areas with write protect disabled. Also, no error is returned. The WEL bit is in the set state.
- It is not possible to erase areas of the serial phase change memory where protect is enabled. Also, no error is returned. The WEL bit is in the set state.
- There are two ways to wait for erase completion. These are described below. Note that the next processing task (write, read, erase, etc.) should be executed after confirming erase completion.
- To use the user API to wait for completion, enable PCM_WAIT_READY in r_qspi_pcm_p5q.h.
- To wait for completion without using the user API, disable PCM_WAIT_READY in r_qspi_pcm_p5q.h and call R_QSPI_PCM_Wait() after processing by the user API finishes. This processing method allows the use of a user-defined duration when waiting for completion. Refer to figure 5.18 for the usage method.
- The argument setting (BusyCnt) when calling R_QSPI_PCM_Wait() differs for bulk erase and sector erase.
  Bulk Erase: BusyCnt = PCM_BE_BUSY_WAIT
  Sector Erase: BusyCnt = PCM_SE_BUSY_WAIT

Arguments

```
uint8_t     DevNo   ; Device number
uint32_t    Addr    ; Erase address
uint8_t     Mode    ; Erase mode (selectable from the following):
                    ;     PCM_MODE_B_ERASE
                    ;     PCM_MODE_S_ERASE
```

Return Value The erase result is returned.

```
PCM_OK              ; Successful operation
PCM_ERR_PARAM       ; Parameter error
PCM_ERR_HARD        ; Hardware error
PCM_ERR_TIMEOUT     ; Time out error (PCM_WAIT_READY enabled)
PCM_ERR_OTHER       ; Other error
```

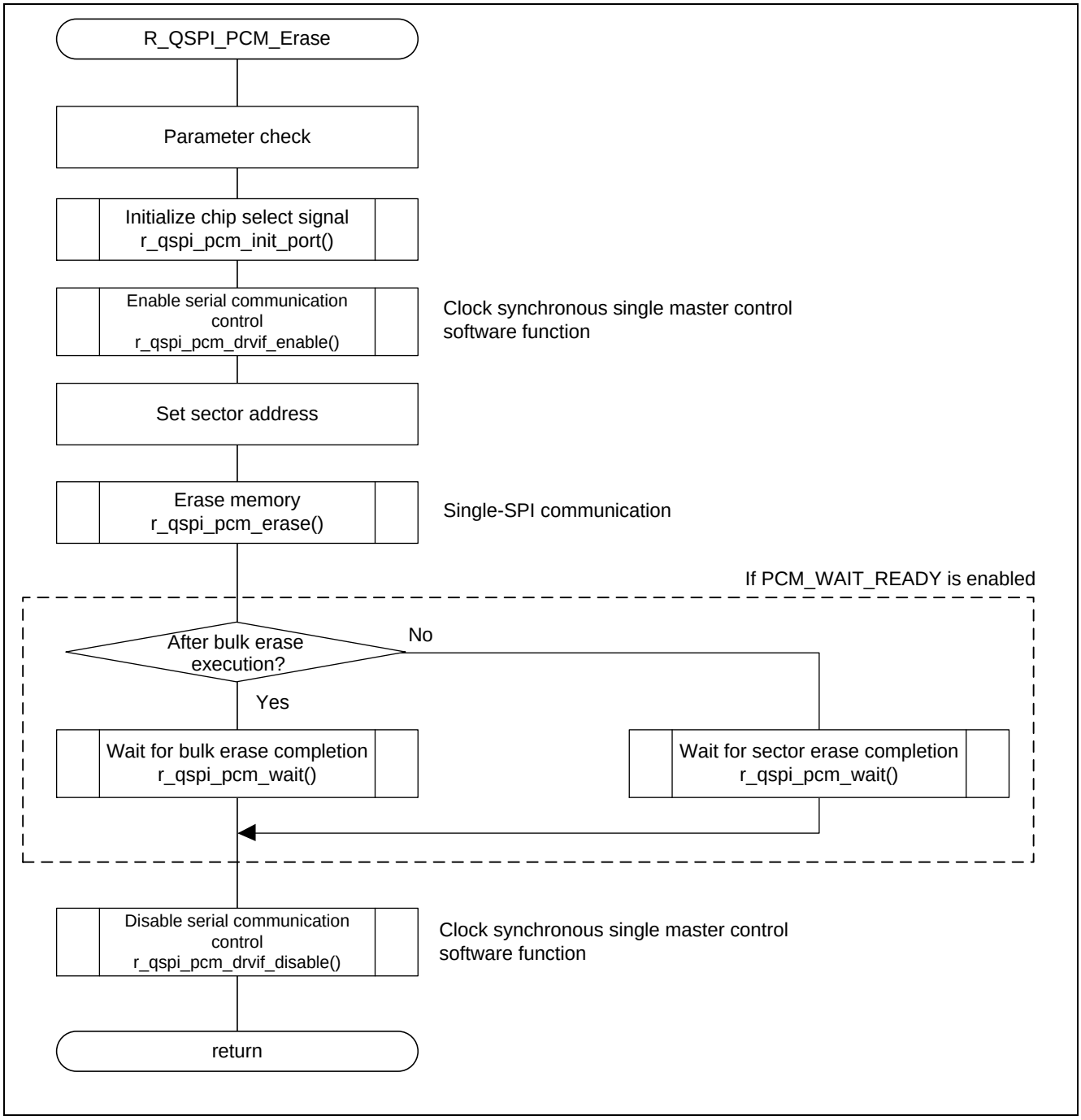


Figure 5.17 Overview of Erase Processing

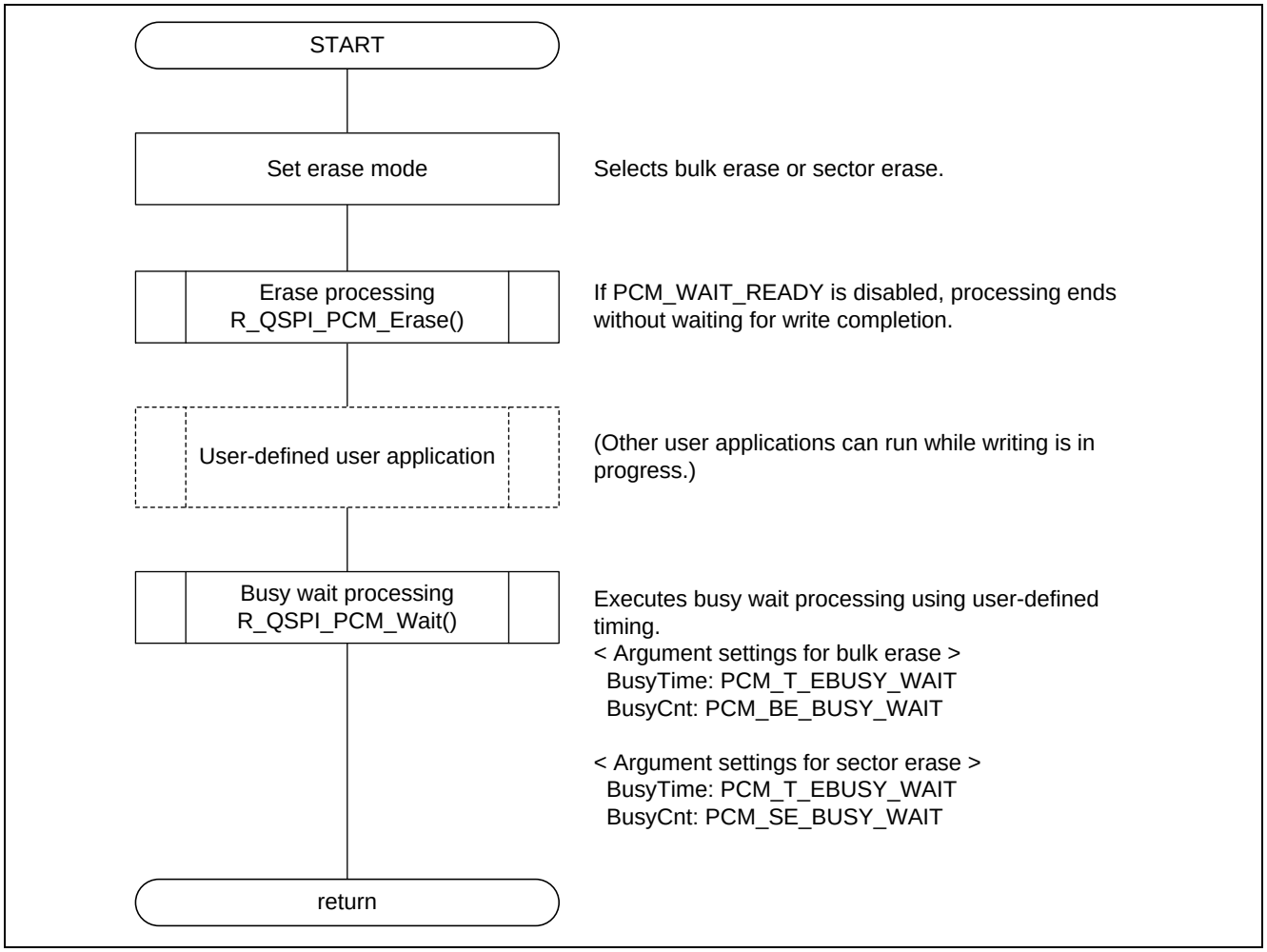


Figure 5.18 Using R_QSPI_PCM_Wait() to Wait for Erase Processing Completion

5.9.9 ID Read Processing

R_QSPI_PCM_ReadID

Outline	ID read processing
Header	r_qspi_pcm_p5q.h, r_qspi_pcm_p5q_sub.h, r_qspi_pcm_sfr.h, r_qspi_pcm_drvif.h
Declaration	error_t R_QSPI_PCM_Read_ID(uint8_t DevNo, uint8_t FAR* pData)
Description	• Reads the manufacturer ID and device ID, and stores them in pData. Set 3 bytes as a read buffer. • Due to the usage limitations*[1] of 32 Mb and 64 Mb serial phase change memory, reading the memory capacity (lower byte) of the device ID returns a value of 18h. The correct values are 16h for 32 Mb, 17h for 64 Mb, and 18h for 128 Mb. Note: 1. Refer to the information on usage limitations in the following document: 32Mb/64Mb P5Q Serial Phase Change Memory Errata Rev. A.
Arguments	uint8_t DevNo ; Device number uint8_t FAR* pData ; Read data storage buffer pointer
Return Value	The read result is returned. PCM_OK ; Successful operation PCM_ERR_PARAM ; Parameter error PCM_ERR_HARD ; Hardware error PCM_ERR_OTHER ; Other error

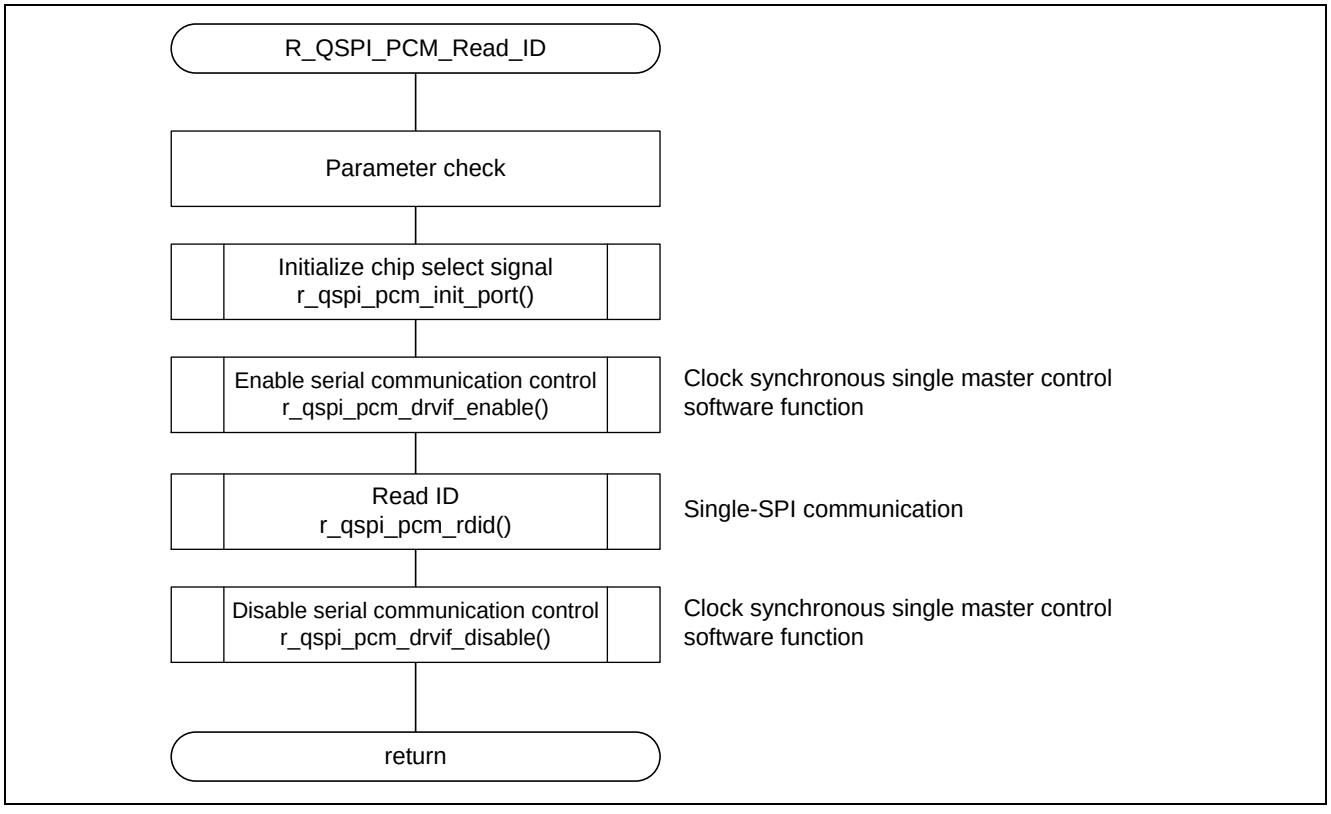


Figure 5.19 Overview of ID Read Processing

5.9.10 Busy Wait Processing

R_QSPI_PCM_Wait

Outline Busy wait processing

Header r_qspi_pcm_p5q.h, r_qspi_pcm_p5q_sub.h, r_qspi_pcm_sfr.h, r_qspi_pcm_drvif.h

Declaration error_t R_QSPI_PCM_Wait(uint8_t DevNo, uint16_t BusyTime, uint32_t BusyCnt)

Description

- Use this function to confirm completion of write or erase when PCM_WAIT_READY is disabled.
- When BusyCnt = 0, a wait is performed for a busy period equal to the BusyTime interval.
- When BusyCnt ≠ 0, a wait is performed for a busy period equal to the BusyTime interval multiplied by BusyCnt. If the busy state exceeds the BusyCnt, FLASH_ERR_TIMEOUT is returned.
- The BusyCnt and BusyTime setting values are different for writing and erasing. A timeout error may occur if busy wait takes place using other than the expected settings. Make settings according to the following table:

State	BusyTime	BusyCnt
Status register write in progress (write protect bit set)	PCM_T_WBUSY_WAIT	PCM_WBUSY_WAIT
Data write in progress	PCM_T_WBUSY_WAIT	PCM_WBUSY_WAIT
Erase in progress (bulk erase)	PCM_T_EBUSY_WAIT	PCM_BE_BUSY_WAIT
Erase in progress (sector erase)	PCM_T_EBUSY_WAIT	PCM_SE_BUSY_WAIT

Arguments

```
uint8_t     DevNo     ; Device number
uint16_t    BusyTime  ; Wait duration (selectable from the following):
                          PCM_T_WBUSY_WAIT: Write
                          PCM_T_EBUSY_WAIT: Erase
uint32_t    BusyCnt   ; Counter (selectable from the following):
                          PCM_WBUSY_WAIT: Write
                          PCM_BE_BUSY_WAIT: Erase (Bulk Erase)
                          PCM_SE_BUSY_WAIT: Erase (Sector Erase)
```

Return Value The read result is returned.

```
PCM_OK              ; Successful operation
PCM_ERR_PARAM       ; Parameter error
PCM_ERR_HARD        ; Hardware error
PCM_ERR_TIMEOUT     ; Time out error (when BusyCnt ≠ 0)
PCM_ERR_OTHER       ; Other error
```

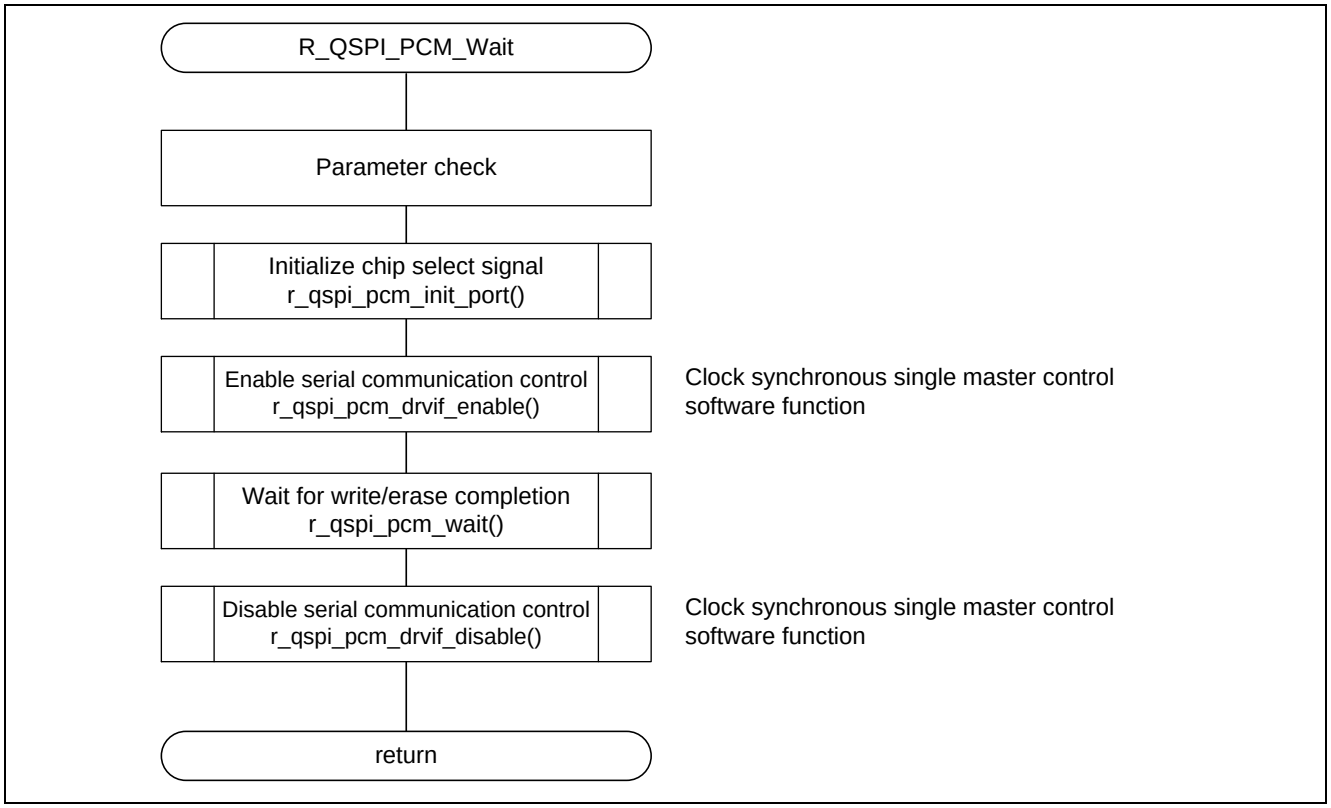


Figure 5.20 Overview of Busy Wait Processing

6. Application Example

Example settings for the control portion of the serial phase change memory are shown below. (Clock synchronous single master control software portion is not covered.)

Refer to the MCU-specific application note for details of the clock synchronous single master control software portion.

Note that the communication speed requires settings for each individual slave device, and these setting are included in the sample code.

The setting locations are designated in each file by the comment /** SET **/.

In addition, for functions used in common (mtl_wait_lp(), etc.), make sure to use the versions included in the MCU-specific clock synchronous single master control software.

6.1 Serial Phase Change Memory Control Software Settings

The setting locations are designated in each file by the comment /** SET **/.

6.1.1 r_qspi_pcm_p5q.h

This is the definition file for the serial phase change memory.

The setting locations are designated in each file by the comment /** SET **/.

(1) **Definition of Number of Devices Used and Device Numbers**

Specify the number of devices to be used, and allocate a number to each device.

In the example below, one device is used, and it is allocated the device number 0.

Up to two devices can be controlled.

```
/*------------------------------------------------------------------------- */
/* Define number of required serial PCM devices.(1~N devices)               */
/* Define the device number in accordance with the number of serial PCM     */
/* devices to be connected.                                                 */
/*------------------------------------------------------------------------- */
/* Define no. of devices */
#define PCM_DEV_NUM             1               /* 1device                  */

/* Define no. of slots */
#define PCM_DEV0                0               /* Device 0                 */
#define PCM_DEV1                1               /* Device 1                 */
```

(2) **Definition of Capacity of Device Used**

Specify the capacity of the device(s) used.

In the example below, a device with a capacity of 64 Mbit is used.

```
/*-------------------------------------------------------------------------*/
/*  Define the serial PCM device.                                          */
/*-------------------------------------------------------------------------*/

//#define P5Q32M                                /* 32Mbit   ( 4MByte)      */
#define P5Q64M                                  /* 64Mbit   ( 8MByte)      */
//#define P5Q32M                                /* 128Mbit  (16MByte)      */
```

(3) **Delay Task Wait Time Setting (Valid when OS Control is Used)**

This setting specifies the OS control* delay task wait time. The unit is ms.

In the example below, a setting of 1 ms is used.

```
/*-------- Definitions of delay task wait time ---------*/
#define PCM_DELAY_TASK    (uint8_t)1    /* OS delay task wait time (Uint:ms)  */
```

Note: * The OS control used in the sample code assumes µITRON 4.0.

(4) **Write/Erase Completion Wait Processing Integration Setting**

The functions listed below support a setting designating waiting for completion following execution of a command. To designate waiting for completion, enable the setting.

Affected functions:

- Write protect setting processing (R_QSPI_PCM_Set_Write_Protect())
- Data write processing (for single-page write) (R_QSPI_PCM_Write_Data_Page())
- Erase processing (R_QSPI_PCM_Erase())

In the example below, waiting for completion is enabled.

```
/*-------- Definitions of using wait ---------*/
/* When you wait completion a PCM writing or erasing, please define it. */
#define PCM_WAIT_READY
```

6.1.2 r_qspi_pcm_p5q_sfr.h

A separate version of r_qspi_pcm_p5q_sfr.h.XXX is provided for each MCU model. Rename the version appropriate for the system to r_qspi_pcm_p5q_sfr.h in order to use it. If there is no available version corresponding to the MCU to be used, refer to the information below and create an appropriate version of r_qspi_pcm_p5q_sfr.h.

The setting locations are designated in each file by the comment /** SET **/.

(1) **Chip Select Signal Setting**

Define the port SFR of the chip select signal to be used.

When connecting two devices, two ports must be defined.

In the example below, port A0 is used on the RX63N.

```
/*------------------------------------------------------------------------*/
/*   Define the CS port.                                                  */
/*------------------------------------------------------------------------*/
#define PCM_DR_CS0    PORTA.PODR.BIT.B0      /* PCM CS0(Negative-true logic)*/
#define PCM_DDR_CS0   PORTA.PDR.BIT.B0       /* PCM CS0(Negative-true logic)*/

#if (PCM_DEV_NUM > 1)
#define PCM_DR_CS1                           /* PCM CS1(Negative-true logic)*/
#define PCM_DDR_CS1                          /* PCM CS1(Negative-true logic)*/
#endif  /* #if (PCM_DEV_NUM > 1) */
```

In the example below, port 80 is used on the RL78/G14.

```
/*------------------------------------------------------------------------- */
/*   Define the CS port.                                                    */
/*------------------------------------------------------------------------- */
#ifdef __CA78K0R__                    /* Renesas RL78 Compiler               */
    #define PCM_DR_CS0  P8.0          /* PCM CS0    (Negative-true logic)    */
    #define PCM_DDR_CS0 PM8.0         /* PCM CS0    (Negative-true logic)    */

    #if (PCM_DEV_NUM > 1)
    #define PCM_DR_CS1                /* PCM CS1    (Negative-true logic)    */
    #define PCM_DDR_CS1               /* PCM CS1    (Negative-true logic)    */
    #endif  /* #if (PCM_DEV_NUM > 1) */
#endif  /* __CA78K0R__ */
#ifdef __CCRL__                       /* Renesas CC-RL Compiler              */
    #define PCM_DR_CS0  P8_bit.no0    /* PCM CS0    (Negative-true logic)    */
    #define PCM_DDR_CS0 PM8_bit.no0   /* PCM CS0    (Negative-true logic)    */

    #if (PCM_DEV_NUM > 1)
    #define PCM_DR_CS1                /* PCM CS1    (Negative-true logic)    */
    #define PCM_DDR_CS1               /* PCM CS1    (Negative-true logic)    */
    #endif  /* #if (PCM_DEV_NUM > 1) */
#endif  /* __CCRL__ */
#ifdef __ICCRL78__                    /* IAR RL78 Compiler                   */
    #define PCM_DR_CS0  P8_bit.no0    /* PCM CS0    (Negative-true logic)    */
    #define PCM_DDR_CS0 PM8_bit.no0   /* PCM CS0    (Negative-true logic)    */

    #if (PCM_DEV_NUM > 1)
    #define PCM_DR_CS1                /* PCM CS1    (Negative-true logic)    */
    #define PCM_DDR_CS1               /* PCM CS1    (Negative-true logic)    */
    #endif  /* #if (PCM_DEV_NUM > 1) */
#endif  /* __ICCRL78__ */
```

(2) **Communication Speed Setting**

These settings define the communication speed. The unit is bits per second.

The appropriate setting values depend on the MCU and serial I/O interface used. Separate settings are provided for different communication applications. See Table 6-1 for details.

Table 6-1 Communication Speed Settings

#define Definition	Application
PCM_BR	Communication processing for other than the following two items (command transmission, etc.)
PCM_BR_WRITE_DATA	Data write processing
PCM_BR_READ_DATA	Data read processing

In the example below, the RSPI of the RX63N is used.

```
    /* PCLK = 48MHz, n=0 for RX63N RSPI */
#define PCM_BR        (uint8_t)(0x01)            /* SPBR initial setting     */
                    /*       ++---------------- 12.00MHz                    */

    /* PCLK = 48MHz, n=0 for RX63N RSPI Write Data */
#define PCM_BR_WRITE_DATA   (uint8_t)(0x01)      /* SPBR initial setting     */
                    /*       ++---------------- 12.00MHz                    */

    /* PCLK = 48MHz, n=0 for RX63N RSPI Read Data */
#define PCM_BR_READ_DATA    (uint8_t)(0x01)      /* SPBR initial setting     */
                    /*       ++---------------- 12.00MHz                    */
```

In the example below, the CSI of the RL78/G14 is used.

```
    /* fMCK = 24MHz for RL78 CSI */
#define PCM_BR              (uint8_t)(0x01)      /* SDR[15:9] initial setting*/
                    /*       ++---------------- 6.00MHz                     */

    /* fMCK = 24MHz for RL78 CSI Write Data */
#define PCM_BR_WRITE_DATA   (uint8_t)(0x01)      /* SDR[15:9] initial setting*/
                    /*       ++---------------- 6.00MHz                     */

    /* fMCK = 24MHz for RL78 CSI Read Data */
#define PCM_BR_READ_DATA    (uint8_t)(0x01)      /* SDR[15:9]initial setting */
                    /*       ++---------------- 6.00MHz                     */
```

Refer to the hardware manual of the MCU when determining the setting values.

6.1.3 r_qspi_pcm_p5q_sub.h

The setting locations are designated in each file by the comment /** SET **/.

(1) **Erase Timeout Duration Settings**

These settings specify the timeout duration when erasing all the data in the memory (bulk erase) and when erasing all the data in a specified sector (sector erase)

The settings below should be reevaluated if the erase duration differs according to the device.

In the example below, the bulk erase timeout duration is set to 100 seconds, and the sector erase timeout duration is set to 800 ms.

```
/*----------------------------------------------------------------------------*/
/*   Define the software timer value of erase or page program busy waiting.  */
/*----------------------------------------------------------------------------*/

/*------- Definitions of software timer value --------*/
/* Bulk Erase : 100s                                                          */
/* Sector Erase : 800ms                                                       */
/* Page (64 bytes) Program (Legacy Program and Bit-alterable Write) : 360us  */
#define PCM_BE_BUSY_WAIT    (uint32_t)(100000)
                              /* Bulk Erase busy timeout  100,000*1ms = 100s */
#define PCM_SE_BUSY_WAIT    (uint32_t)(800)
                              /* Sector Erase busy timeout   800*1ms = 800ms */
```

(2) **Write Timeout Duration Setting**

The settings below should be reevaluated if the write duration differs according to the device.

In the example below, the write timeout duration is set 400 µs.

```
#define PCM_WBUSY_WAIT  (uint32_t)(400) /* Write busy timeout 400*1us = 400us */
```

6.1.4 r_qspi_pcm_p5q_sub.c

This is the source file for internal functions of the serial phase change memory.

The setting locations are designated in each file by the comment /** SET **/.

(1) **Macro Function R_QSPI_PCM_CMD_READ() Definition**

This specifies the operation command for read processing. Define one item from the table below.

Table 6-2 Macro Function R_QSPI_PCM_CMD_READ() Definition

No.	#define Definition	Instruction Code on Data Sheet	Processing Details
1	r_qspi_pcm_send_cmd(**PCM_CMD_READ**, (uint32_t)Addr, PCM_CMD_SIZE + PCM_ADDR_SIZE)	READ	Single-SPI read (normal)
2	r_qspi_pcm_send_cmd(**PCM_CMD_FREAD**,(uint32_t) Addr, PCM_CMD_SIZE + PCM_ADDR_SIZE + 1)	FAST_READ	Single-SPI read (high-speed)
3	r_qspi_pcm_send_cmd(**PCM_CMD_DOFR**, (uint32_t)Addr, PCM_CMD_SIZE + PCM_ADDR_SIZE + 1)	DOFR	Dual-SPI read (high-speed)
4	r_qspi_pcm_send_cmd(**PCM_CMD_QOFR**, (uint32_t) Addr, PCM_CMD_SIZE + PCM_ADDR_SIZE + 1)	QOFR	Quad-SPI read (high-speed)

(2) **Macro Function R_QSPI_PCM_CMD_PP_LEGACY() Setting**

This specifies the operation command for write processing by legacy program. Define one item from the table below.

Table 6-3 Macro Function R_QSPI_PCM_CMD_PP_LEGACY() Definition

No.	#define Definition	Instruction Code on Data Sheet	Processing Details
1	r_qspi_pcm_send_cmd(**PCM_CMD_PP_L**, (uint32_t)Addr, PCM_CMD_SIZE + PCM_ADDR_SIZE)	PP	Single-SPI write (legacy program)
2	r_qspi_pcm_send_cmd(**PCM_CMD_DIPP_L**, (uint32_t)Addr, PCM_CMD_SIZE + PCM_ADDR_SIZE)	DIFP	Dual-SPI write (legacy program)
3	r_qspi_pcm_send_cmd(**PCM_CMD_QIPP_L**, (uint32_t)Addr, PCM_CMD_SIZE + PCM_ADDR_SIZE)	QIFP	Quad-SPI write (legacy program)

(3) **Macro Function R_QSPI_PCM_CMD_PP_BIT_ALTERABLE() Setting**

This specifies the operation command for write processing by bit-alterable write. Define one item from the table below.

Table 6-4 Macro Function R_QSPI_PCM_CMD_PP_ BIT_ALTERABLE() Definition

No.	#define Definition	Instruction Code on Data Sheet	Processing Details
1	r_qspi_pcm_send_cmd(**PCM_CMD_PP_BA**, (uint32_t)Addr, PCM_CMD_SIZE + PCM_ADDR_SIZE)	PP	Single-SPI write (bit-alterable write)
2	r_qspi_pcm_send_cmd(**PCM_CMD_DIPP_BA**, (uint32_t)Addr, PCM_CMD_SIZE + PCM_ADDR_SIZE)	DIFP	Dual-SPI write (bit-alterable write)
3	r_qspi_pcm_send_cmd(**PCM_CMD_QIPP_BA**, (uint32_t)Addr, PCM_CMD_SIZE + PCM_ADDR_SIZE)	QIFP	Quad-SPI write (bit-alterable write)

(4) **Macro Function R_QSPI_PCM_CMD_PP_ON_ALL_1S() Setting**

This specifies the operation command for write processing by on all 1s. Define one item from the table below.

Table 6-5 Macro Function R_QSPI_PCM_CMD_PP_ON_ALL_1S() Definition

No.	#define Definition	Instruction Code on Data Sheet	Processing Details
1	r_qspi_pcm_send_cmd(**PCM_CMD_PP_OA**, (uint32_t)Addr, PCM_CMD_SIZE + PCM_ADDR_SIZE)	PP	Single-SPI write (on all 1s)
2	r_qspi_pcm_send_cmd(**PCM_CMD_DIPP_OA**, (uint32_t)Addr, PCM_CMD_SIZE + PCM_ADDR_SIZE)	DIFP	Dual-SPI write (on all 1s)
3	r_qspi_pcm_send_cmd(**PCM_CMD_QIPP_OA**, (uint32_t)Addr, PCM_CMD_SIZE + PCM_ADDR_SIZE)	QIFP	Quad-SPI write (on all 1s)

6.1.5 r_qspi_pcm_p5q_drvif.c

This is the source file for the clock synchronous single control software interface of the serial phase change memory.

The setting locations are designated in each file by the comment /** SET **/.

(1) **r_qspi_pcm_drvif_init_driver() Setting**

This specifies the driver initialization processing of the clock synchronous single master control software used.

If there is no corresponding item, add one as necessary.

```
error_t r_qspi_pcm_drvif_init_driver(void)
{
   return R_SIO_Init_Driver();
}
```

(2) **r_qspi_pcm_drvif_disable() Setting**

This specifies the communication disable setting processing of the clock synchronous single master control software used.

If there is no corresponding item, add one as necessary.

```
error_t r_qspi_pcm_drvif_disable(void)
{
   return R_SIO_Disable();
}
```

(3) **r_qspi_pcm_drvif_enable() Setting**

This specifies the communication enable setting processing of the clock synchronous single master control software used.

If there is no corresponding item, add one as necessary.

```
error_t r_qspi_pcm_drvif_enable(uint8_t BrgData)
{
   return R_SIO_Enable(BrgData);
}
```

(4) **r_qspi_pcm_drvif_enable_tx_data() Setting**

This specifies the data write-only communication enable setting processing of the clock synchronous single master control software used.

If there is no corresponding item, add one as necessary.

```
error_t r_qspi_pcm_drvif_enable_tx_data(uint8_t BrgData)
{
   return R_SIO_Enable(BrgData);
}
```

(5) **r_qspi_pcm_drvif_enable_rx_data() Setting**

This specifies the data read-only communication enable setting processing of the clock synchronous single master control software used.

If there is no corresponding item, add one as necessary.

```
error_t r_qspi_pcm_drvif_enable_rx_data(uint8_t BrgData)
{
   return R_SIO_Enable(BrgData);
}
```

(6) **r_qspi_pcm_drvif_open() Setting**

This specifies the communication open setting processing of the clock synchronous single master control software used.

If there is no corresponding item, add one as necessary.

```
error_t r_qspi_pcm_drvif_open(void)
{
   return R_SIO_Open_Port();
}
```

(7) **r_qspi_pcm_drvif_tx() Setting**

This specifies the data transmit processing of the clock synchronous single master control software used. It is used mainly for command transmission and writing to the status register.

If there is no corresponding item, add one as necessary.

```
error_t r_qspi_pcm_drvif_tx(uint16_t TxCnt, uint8_t FAR * pData)
{
   return R_SIO_Tx_Data(TxCnt, pData);
}
```

(8) **r_qspi_pcm_drvif_tx_data() Setting**

This specifies the write-only data transmit processing of the clock synchronous single master control software used.

If there is no corresponding item, add one as necessary.

```
error_t r_qspi_pcm_drvif_tx_data(uint16_t TxCnt, uint8_t FAR * pData)
{
   return R_SIO_Tx_Data(TxCnt, pData);
}
```

(9) **r_qspi_pcm_drvif_rx() Setting**

This specifies the data receive processing of the clock synchronous single master control software used. It is used mainly for reading the status register.

If there is no corresponding item, add one as necessary.

```
error_t r_qspi_pcm_drvif_rx(uint16_t RxCnt, uint8_t FAR * pData)
{
   return R_SIO_Rx_Data(RxCnt, pData);
}
```

(10) **r_qspi_pcm_drvif_rx_data() Settings**

This specifies the read-only data receive processing of the clock synchronous single master control software used.

If there is no corresponding item, add one as necessary.

```
error_t r_qspi_pcm_drvif_rx_data(uint16_t RxCnt, uint8_t FAR * pData)
{
   return R_SIO_Rx_Data(RxCnt, pData);
}
```

6.1.6 r_qspi_pcm_p5q_sfr_rl78.c

This is an I/O module file for this Serial Flash memory.

The settings to be made are identified by the comments header "/** SET **/" in the file.

1. Setting the definition of SFR

When an RL78 family or 78K0R family microcontroller is used, there will be predefined preprocessor symbols in the C compiler used. The program is coded using these predefined preprocessor symbols.

Also, when the microcontroller used is an RL78 family or 78K0R family product and furthermore, the IAR Systems integrated development environment is used, it will be necessary to set the header file in which the SFRs for the microcontroller used are defined.

See the clock synchronous single master control software for the individual microcontroller.

These settings are used for the SPI slave device select control signals.

Table 6-6 Microcontroller and SFR Area Define Settings

Integrated development environment	Microcontroller	SFR setting required?	Method
CubeSuite+ CS+	RL78	Not required	Not required
	78K0R	Not required	Not required
	RX	Not required	Not required
IAR Embedded Workbench	RL78	Required	#ifdef __ICCRL78__ #include <ior5f104pj.h> ← Change to match the microcontroller used. #include <ior5f104pj_ext.h> ← Change to match the microcontroller used. #endif
	78K0R	Required	#ifdef __ICC78K__ #include <io78f1009_64.h> ← Change to match the microcontroller used. #include <io78f1009_64_ext.h> ← Change to match the microcontroller used. #endif
	RX	(Not supported by this software)	(Not supported by this software)

The example below is for the 100-pin RL78/G14 microcontroller.

```
#ifdef __ICCRL78__                      /* IAR RL78 Compiler                */
   #include <ior5f104pj.h>              /* for RL78/G14 100pin (R5F104PJ)   */
   #include <ior5f104pj_ext.h>          /* for RL78/G14 100pin (R5F104PJ)   */
#endif   /* __ICCRL78__ */
```

7. Usage Notes

7.1 Notes on Integrating Sample Code

To integrate the sample code, include the following header files:

r_qspi_pcm_p5q.h
r_qspi_pcm_p5q_sub.h
r_qspi_pcm_p5q _sfr.h
r_qspi_pcm_p5q _drvif.h

7.2 Using an MCU with On-Chip Cache

Specify a non-cached area for the read/write data storage buffer.

7.3 Support for Other Capacities

To support other capacities, the following definitions must be reevaluated:

PCM_MEM_SIZE
PCM_SECT_ADDR
PCM_PAGE_SIZE
PCM_ADDR_SIZE
PCM_WP_WHOLE_MEM

It may be necessary to reevaluate definitions other than those listed above as well. Obtain the data sheet of the memory, and reevaluate the definitions as appropriate.

7.4 Using Other Slave Devices

It is possible to control other slave devices connected to the same SPI bus.

Refer to the sample code when creating slave device control software.

Note that the communication speed may be set individually for each slave device control software program.

7.5 Voltage Stabilization Time After Power-On

Make sure to allow sufficient time for the voltage to stabilize after power-on before calling the initialization function.

Check the data sheet of the slave device regarding the voltage stabilization wait time after power-on.

7.6 Serial Phase Change Memory Usage Limitations

As of the date of authorship of this application note, Micron Technology has announced the usage limitations listed below. When using the affected functions, make sure to check the latest version of the data sheet and carry out adequate evaluation.

(1) **128Mb P5Q Serial Phase Change Memory Errata**

No.	Errata Rev.	Usage Limitation Details	Effect on Sample Code
1	D	The write protect bits (BP3, BP2, BP1, and BP0) cannot be set to 1.	It is not possible to set write protect bits to 1 by means of R_QSPI_PCM_Set_Write_Protect() in the user API.

(2) **64Mb P5Q Serial Phase Change Memory Errata**

No.	Errata Rev.	Usage Limitation Details	Effect on Sample Code
1	A	The write protect bits (BP3, BP2, BP1, and BP0) cannot be set to 1.	It is not possible to set write protect bits to 1 by means of R_QSPI_PCM_Set_Write_Protect() in the user API.
2	A	Reading the device ID memory capacity (lower byte) returns a value of 18h. The correct value is 17h.	When a read is performed using R_QSPI_PCM_RDID() in the user API, the value of the third byte of pData is 18h.
3	A	During data read operations (READ, READ_FAST, DUAL OUTPUT FAST READ, and QUAD OUTPUT FAST READ), the address does not roll over to 000000h after 7FFFFFh is read.	No effect. This is because read operations with rollover are not enabled in the R_QSPI_PCM_Read_Data() read processing function of the sample code.

(3) **32Mb P5Q Serial Phase Change Memory Errata**

No.	Errata Rev.	Usage Limitation Details	Effect on Sample Code
1	A	The write protect bits (BP3, BP2, BP1, and BP0) cannot be set to 1.	It is not possible to set write protect bits to 1 by means of R_QSPI_PCM_Set_Write_Protect() in the user API.
2	A	Reading the device ID memory capacity (lower byte) returns a value of 18h. The correct value is 16h.	When a read is performed using R_QSPI_PCM_RDID() in the user API, the value of the third byte of pData is 18h.
3	A	During data read operations (READ, READ_FAST, DUAL OUTPUT FAST READ, and QUAD OUTPUT FAST READ), the address does not roll over to 000000h after 3FFFFFh is read.	No effect. This is because read operations with rollover are not enabled in the R_QSPI_PCM_Read_Data() read processing function of the sample code.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.01	Sep. 26, 2013	—	First edition issued
1.03	Apr. 30, 2014	1	Modified Introduction to add short address.
		1	Added RX63N, RX63T, RX210, RX21A, RX220, RX111 RL78/G1C, RL78/L12, RL78/L13, RL78/L1C and RL78/G14 as supported devices.
		5, 7	Added 2.1 RX Family and 2.2 RL78 Family, 78K0R/Kx3-L.
		6	Added the following conditions to section 2.1. (2) RX111 RSPI (3) RX111 SCI
		7 to 11	Added the following conditions to section 2.1. (2) RL78/G14 SAU Integrated Development Environment IAR Embedded Workbench (3) RL78/G1C SAU Integrated Development Environment CubeSuite+ (4) RL78/G1C SAU Integrated Development Environment IAR Embedded Workbench (5) RL78/L12 SAU Integrated Development Environment CubeSuite+ (6) RL78/L12 SAU Integrated Development Environment IAR Embedded Workbench (7) RL78/L13 SAU Integrated Development Environment CubeSuite+ (8) RL78/L13 SAU Integrated Development Environment IAR Embedded Workbench (9) RL78/L1C SAU Integrated Development Environment CubeSuite+ (10) RL78/L1C SAU Integrated Development Environment IAR Embedded Workbench
		12	Updated application note title in section 3, Related Application Notes. -RX210, RX21A, RX220, RX63N, RX63T, RX111 Group Clock Synchronous Single Master Control Software Using the RSPI (R01AN1196EJ) -RX210, RX21A, RX220, RX63N, RX63T, RX111 Group Clock Synchronous Single Master Control Software Using the SCI (R01AN1229EJ) -RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ)
		22, 24	Added 5.3.1 RX Family and 5.3.2 RL78 Family, 78K0R/Kx3-L.
		23	Added the following to section 5.3.1, Sizes of Required Memory. (2) RX111 RSPI (3) RX111 SCI
		24 to 25	Added the following to section 5.3.2, Sizes of Required Memory. (2) RL78/G14 SAU Integrated Development Environment IAR Embedded Workbench (3) RL78/L13 SAU Integrated Development Environment CubeSuite+ (4) RL78/L13 SAU Integrated Development Environment IAR Embedded Workbench

		26	5.4 File Structure Changed name for folder for the sample code. Changed application note number. Added new device register common definitions.
		62	Added 6.1.6 r_qspi_pcm_p5q_sfr_rl78.c.
		-	Changed “Table No” format.
1.04	Mar. 31, 2016	7	Section 2.2 RL78 Family, 78K0R/Kx3-L Changed the following conditions. (1) RL78/G14 Integrated Development Environment CS+ for CA,CX (Compiler: CA78K0R) Added the following conditions. (2) RL78/G14 Integrated Development Environment CS+ for CC (Compiler: CC-RL)
		25	Section 5.3.2 RL78 Family, 78K0R/Kx3-L Changed the following sizes. (1) RL78/G14 Integrated Development Environment CS+ for CA,CX (Compiler: CA78K0R) Added the following sizes. (2) RL78/G14 Integrated Development Environment CS+ for CC (Compiler: CC-RL)
		28	Changed the following table to Section 5.4.
		56 to 57	6.1.2 r_qspi_flash_s25fl_sfr.h Changed the example.

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. 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended 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; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor 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 damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be 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, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 5.0