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

## Macronix International MX25/66L Family Serial NOR Flash Memory Control Software

## Introduction

This application note describes how to control MX25/66L serial NOR flash memory, manufactured by Macronix International Co., Ltd., 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 NOR flash 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. Note that although the clock synchronous single master control software may support newer microcontrollers, there may be cases where the control software presented in this application note has not yet been updated to match. For information on the latest control software releases, see the “Clock Synchronous Single Master Control Software (Lower-level layer of the software)” section of the following webpage:

SPI/QSPI Serial Flash Memory, QSPI Serial Phase Change Memory Driver
SPI/QSPI Serial Flash Memory, QSPI Serial Phase Change Memory Driver | Renesas

## Target Devices

Serial NOR Flash Memory : MX25/66L family serial NOR flash memory, manufactured by Macronix International Co., Ltd.

MX25R1635F (16 Mbit)

MX25L3235E (32 Mbit), MX25L3233F (32 Mbit)

MX25L25635F (256 Mbit), MX66L51235F (512 Mbit)

MX25L51245G (512 Mbit), MX66L1G45G (1 Gbit)

MCUs on which operation has been confirmed:

RX100 Series : RX111 (using SCI), RX111 (using RSPI)

RX600 Series : RX64M (using RSPI), RX64M (using SCI)

RL78/G1x Series : RL78/G14, RL78/G1C (using SAU)

RL78/L1x Series : RL78/L12, RL78/L13, RL78/L1C (using SAU)

RL78/G2x Series : RL78/G23 (using 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 dual-SPI mode)
- Quad-SPI (communication in quad-SPI mode)

## Contents

# 1. Specifications

A Renesas Electronics MCU is used to control MX25/66L serial NOR flash memory, manufactured by Macronix International Co., Ltd.

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 Macronix International Co., Ltd., MX25/66L serial NOR flash 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 NOR flash 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 Devices and Their Applications**

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

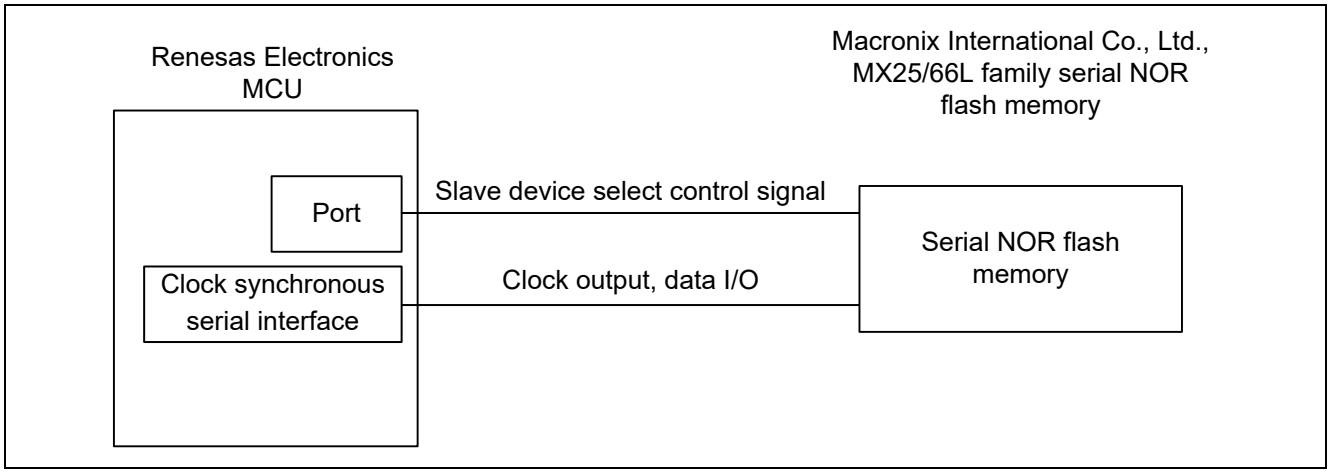



**Figure 1.1 Usage Example**

# 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) **RX111 RSPI**

**Table 2.1 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RX111 Group (Program ROM: 128 KB/RAM: 16 KB) |
| Operating frequency | ICLK: 32 MHz, PCLKB: 32 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>CubeSuite+ V2.01.00 |
| C compiler | Renesas Electronics Corporation<br>RX Family C/C++ Compiler Package (Toolchain 2.01.00) |
| | Compiler options<br>The integrated development environment default settings are used. |
| Endian order | Big endian / Little endian |
| Sample code version number | Ver. 2.21 |
| Software | Clock synchronous single master control software using the RSPI of RX210, RX21A, RX220, RX63N, RX63T, RX111 Group, version 2.04.R04 |
| Board | Renesas Starter Kit for RX111 |

(2) **RX111 SCI**

**Table 2.2 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RX111 Group (Program ROM: 128 KB/RAM: 16 KB) |
| Operating frequency | ICLK: 32 MHz, PCLKB: 32 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>CubeSuite+ V2.01.00 |
| C compiler | Renesas Electronics Corporation<br>RX Family C/C++ Compiler Package (Toolchain 2.01.00) |
| | Compiler options<br>The integrated development environment default settings are used. |
| Endian order | Big endian / Little endian |
| Sample code version number | Ver. 2.21 |
| Software | Clock synchronous single master control software using the SCI of RX210, RX21A, RX220, RX63N, RX63T, RX111 Group, version 2.01.R05 |
| Board | Renesas Starter Kit for RX111 |

### (3) RX64M RSPI

**Table 2.3 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RX64M Group (Program ROM: 4 MB/RAM: 512 KB) |
| Operating frequency | ICLK: 120 MHz, PCLKA: 120 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>e$^2$ studio V3.0.1.08 |
| C compiler | Renesas Electronics Corporation<br>C/C++ compiler for RX family V.2.01.00 |
| | Compiler options<br>The integrated development environment default settings are used, with the following option added:<br>-lang = c99 |
| Endian order | Big endian / Little endian |
| Sample code version number | Ver. 2.21 |
| Software | Clock synchronous single master control software using the RSPI of RX210, RX21A, RX220, RX63N, RX63T,RX111, RX64M Group, version 2.05 |
| Board | Renesas Starter Kit for RX64M |

### (4) RX64M SCI

**Table 2.4 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RX64M Group (Program ROM: 4 MB/RAM: 512 KB) |
| Operating frequency | ICLK: 120 MHz, PCLKA: 120 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>e$^2$ studio V3.0.1.08 |
| C compiler | Renesas Electronics Corporation<br>C/C++ compiler for RX family V.2.01.00 |
| | Compiler options<br>The integrated development environment default settings are used, with the following option added:<br>-lang = c99 |
| Endian order | Big endian / Little endian |
| Sample code version number | Ver. 2.21 |
| Software | Clock synchronous single master control software using the RSPI of RX210, RX21A, RX220, RX63N, RX63T,RX111, RX64M Group, version 2.05 |
| Board | Renesas Starter Kit for RX64M |

## 2.2 RL78 Family, 78K0R/Kx3-L

(1) **RL78/G14 SAU CubeSuite+ Integrated Development Environment (Compiler:CA78K0R)**

**Table 2.5 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/G14 (Program ROM: 256 MB/RAM: 24 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 6 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>CubeSuite+ V2.01.00 |
| C compiler | Renesas Electronics Corporation<br>CubeSuite+ RL78 or 78K0R compiler CA78K0R, version 1.60 |
| | Compiler options<br>The integrated development environment default setting (“-qx2”) is 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 2.02 |
| Board | Renesas Starter Kit for RL78/G14 |

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

**Table 2.6 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/G14 (Program ROM: 256 MB/RAM: 24 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 6 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>CS+ for CC V3.02.00 |
| C compiler | Renesas Electronics Corporation<br>RL78 compiler CC-RL V1.02.00 |
| | Compiler options<br>The default settings (Perform the default optimization(None)) for the integrated development environment are used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| Software | RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group<br>Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0105), version 2.05) |
| Board | Renesas Starter Kit for RL78/G14 |

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

**Table 2.7 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/G14 (Program ROM: 256 KB/RAM: 24 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 6 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | IAR Systems<br>IAR Embedded Workbench for Renesas RL78 (Ver.1.30.2) |
| C compiler | IAR Systems<br>IAR Assembler for Renesas RL78 (Ver.1.30.2.50666)<br>IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.2.50666) |
| | Compiler options<br>The integrated development environment default setting (“level: low”) is 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 SAU CubeSuite+ Integrated Development Environment (Compiler:CA78K0R)**

**Table 2.8 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/G1C (Program ROM: 32 KB/RAM: 5.5 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 12 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>CubeSuite+ V2.01.00 |
| C compiler | Renesas Electronics Corporation<br>CubeSuite+ RL78 or 78K0R compiler CA78K0R, version 1.70 |
| | Compiler options<br>The integrated development environment default setting (“-qx2”) is used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| Software | RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group<br>Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), version 2.03 |
| Board | Renesas RL78/G1C target board, QB-R5F10JGC-TB |

### (5) RL78/G1C SAU CS+ for CC Integrated Development Environment (Compiler:CC-RL)

**Table 2.9 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/G1C (Program ROM: 32 KB/RAM: 5.5 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 12 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>CS+ for CC V3.02.00 |
| C compiler | Renesas Electronics Corporation<br>RL78 compiler CC-RL V1.02.00 |
| | Compiler options<br>The default settings (Perform the default optimization(None)) for the integrated development environment are used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| Software | RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group<br>Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0105), version 2.05) |
| Board | Renesas RL78/G1C target board, QB-R5F10JGC-TB |

### (6) RL78/G1C SAU IAR Embedded Workbench Integrated Development Environment

**Table 2.10 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/G1C (Program ROM: 32 KB/RAM: 5.5 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 12 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | IAR Systems<br>IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5) |
| C compiler | IAR Systems<br>IAR Assembler for Renesas RL78 (Ver.1.30.4.50715)<br>IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715) |
| | Compiler options<br>The integrated development environment default setting (“level: low”) is used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| Software | RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group<br>Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), version 2.03 |
| Board | Renesas RL78/G1C target board, QB-R5F10JGC-TB |

### (7) RL78/L12 SAU CubeSuite+ Integrated Development Environment (Compiler:CA78K0R)

**Table 2.11 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/L12 (Program ROM: 32 KB/RAM: 1.5 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 6 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>CubeSuite+ V2.01.00 |
| C compiler | Renesas Electronics Corporation<br>CubeSuite+ RL78 or 78K0R compiler CA78K0R, version 1.70 |
| | Compiler options<br>The integrated development environment default setting (“-qx2”) is used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| Software | RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group<br>Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), version 2.03 |
| Board | Renesas Starter Kit for RL78/L12 |

### (8) RL78/L12 SAU CS+ for CC Integrated Development Environment (Compiler:CC-RL)

**Table 2.12 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/L12 (Program ROM: 32 KB/RAM: 1.5 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 6 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>CS+ for CC V3.02.00 |
| C compiler | Renesas Electronics Corporation<br>RL78 compiler CC-RL V1.02.00 |
| | Compiler options<br>The default settings (Perform the default optimization(None)) for the integrated development environment are used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| Software | RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group<br>Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0105), version 2.05) |
| Board | Renesas Starter Kit for RL78/L12 |

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

**Table 2.13 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/L12 (Program ROM: 32 KB/RAM: 1.5 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 6 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | IAR Systems<br>IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5) |
| C compiler | IAR Systems<br>IAR Assembler for Renesas RL78 (Ver.1.30.4.50715)<br>IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715) |
| | Compiler options<br>The integrated development environment default setting (“level: low”) is used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| Software | RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group<br>Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), version 2.03 |
| Board | Renesas Starter Kit for RL78/L12 |

(10) **RL78/L13 SAU CubeSuite+ Integrated Development Environment (Compiler:CA78K0R)**

**Table 2.14 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/L13 (Program ROM: 128 KB/RAM: 8 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 6 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>CubeSuite+ V2.01.00 |
| C compiler | Renesas Electronics Corporation<br>CubeSuite+ RL78 or 78K0R compiler CA78K0R, version 1.70 |
| | Compiler options<br>The integrated development environment default setting (“-qx2”) is used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| Software | RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group<br>Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), version 2.03 |
| Board | Renesas Starter Kit for RL78/L13 |

(11) **RL78/L13 SAU CS+ for CC Integrated Development Environment (Compiler:CC-RL)**

**Table 2.15 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/L13 (Program ROM: 128 KB/RAM: 8 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 6 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>CS+ for CC V3.02.00 |
| C compiler | Renesas Electronics Corporation<br>RL78 compiler CC-RL V1.02.00 |
| | Compiler options<br>The default settings (Perform the default optimization(None)) for the integrated development environment are used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| 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 (R01AN1195EJ0105), version 2.05) |
| Board | Renesas Starter Kit for RL78/L13 |

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

**Table 2.16 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/L13 (Program ROM: 128 KB/RAM: 8 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 6 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | IAR Systems<br>IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5) |
| C compiler | IAR Systems<br>IAR Assembler for Renesas RL78 (Ver.1.30.4.50715)<br>IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715) |
| | Compiler options<br>The integrated development environment default setting (“level: low”) is used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| 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), version 2.03 |
| Board | Renesas Starter Kit for RL78/L13 |

(13) **RL78/L1C SAU CubeSuite+ Integrated Development Environment (Compiler:CA78K0R)**

**Table 2.17 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/L1C (Program ROM: 256 KB/RAM: 16 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 6 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>CubeSuite+ V2.01.00 |
| C compiler | Renesas Electronics Corporation<br>CubeSuite+ RL78 or 78K0R compiler CA78K0R, version 1.70 |
| | Compiler options<br>The integrated development environment default setting (“-qx2”) is used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| Software | RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group<br>Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), version 2.03 |
| Board | Renesas Starter Kit for RL78/L1C |

(14) **RL78/L1C SAU CS+ for CC Integrated Development Environment (Compiler:CC-RL)**

**Table 2.18 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/L1C (Program ROM: 256 KB/RAM: 16 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 6 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>CS+ for CC V3.02.00 |
| C compiler | Renesas Electronics Corporation<br>RL78 compiler CC-RL V1.02.00 |
| | Compiler options<br>The default settings (Perform the default optimization(None)) for the integrated development environment are used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| Software | RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group<br>Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0105), version 2.05) |
| Board | Renesas Starter Kit for RL78/L1C |

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

**Table 2.19 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/L1C (Program ROM: 256 KB/RAM: 16 KB) |
| Operating frequency | Main system clock: 24 MHz<br>CPU/peripheral hardware clock: 24 MHz<br>Serial clock: 6 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | IAR Systems<br>IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5) |
| C compiler | IAR Systems<br>IAR Assembler for Renesas RL78 (Ver.1.30.4.50715)<br>IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715) |
| | Compiler options<br>The integrated development environment default setting (“level: low”) is used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| Software | RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group<br>Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ0103), version 2.03 |
| Board | Renesas Starter Kit for RL78/L1C |

(16) **RL78/G23 SAU CS+ for CC Integrated Development Environment (Compiler:CC-RL )**

**Table 2.20 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/G23 (Program ROM: 128 KB/RAM: 16 KB) |
| Operating frequency | Main system clock: 32 MHz<br>CPU/peripheral hardware clock: 32 MHz<br>Serial clock: 8 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>CS+ for CC V8.05.00 |
| C compiler | Renesas Electronics Corporation<br>RL78 compiler CC-RL V1.10.00 |
| | Compiler options<br>The default settings (Perform the default optimization(None)) for the integrated development environment are used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| Software | RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C, RL78/G23 Group<br>Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ) |
| Board | RL78/G23-64p Fast Prototyping Board |

### (17) RL78/G23 SAU IAR Embedded Workbench Integrated Development Environment

**Table 2.21 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/G23 (Program ROM: 128 KB/RAM: 16 KB) |
| Operating frequency | Main system clock: 32 MHz<br>CPU/peripheral hardware clock: 32 MHz<br>Serial clock: 8 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | IAR Systems<br>IAR Embedded Workbench for Renesas RL78 (Ver.4.21.1.2409) |
| C compiler | IAR Systems<br>IAR Assembler for Renesas RL78 (Ver.4.21.1.2409)<br>IAR C/C++ Compiler for Renesas RL78 (Ver. 4.21.1.2409) |
| | Compiler options<br>The integrated development environment default setting (“level: low”) is used. |
| Endian order | Little endian |
| Sample code version number | Ver. 2.21 |
| Software | RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C, RL78/G23 Group Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ) |
| Board | RL78/G23-64p Fast Prototyping Board |

### (18) RL78/G23 SAU e$^2$ studio Integrated Development Environment (Compiler:LLVM )

**Table 2.22 Operation Confirmation Conditions**

| Item | Description |
|---|---|
| Memory | Macronix International Co., Ltd. MX25/66L family serial NOR flash memory |
| Microcontroller used | RL78/G23 (Program ROM: 128 KB/RAM: 16 KB) |
| Operating frequency | Main system clock: 32 MHz<br>CPU/peripheral hardware clock: 32 MHz<br>Serial clock: 8 MHz |
| Operating voltage | 3.3 V |
| Integrated development environment | Renesas Electronics Corporation<br>e$^2$ studio 22.4.0.R20220331-2313 |
| C compiler | Open Source Compiler<br>LLVM for Renesas RL78 10.0.0.202203 |
| | Compiler options<br>The size settings ("Optimize size (-Os)”) 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, RL78/G23 Group Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit (R01AN1195EJ) |
| Board | RL78/G23-64p Fast Prototyping Board |

# 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, RX64M Group Clock Synchronous Single Master Control Software Using the RSPI (R01AN1196EJ)
- RX210, RX21A, RX220, RX63N, RX63T, RX111, RX64M 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, RL78/G23 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

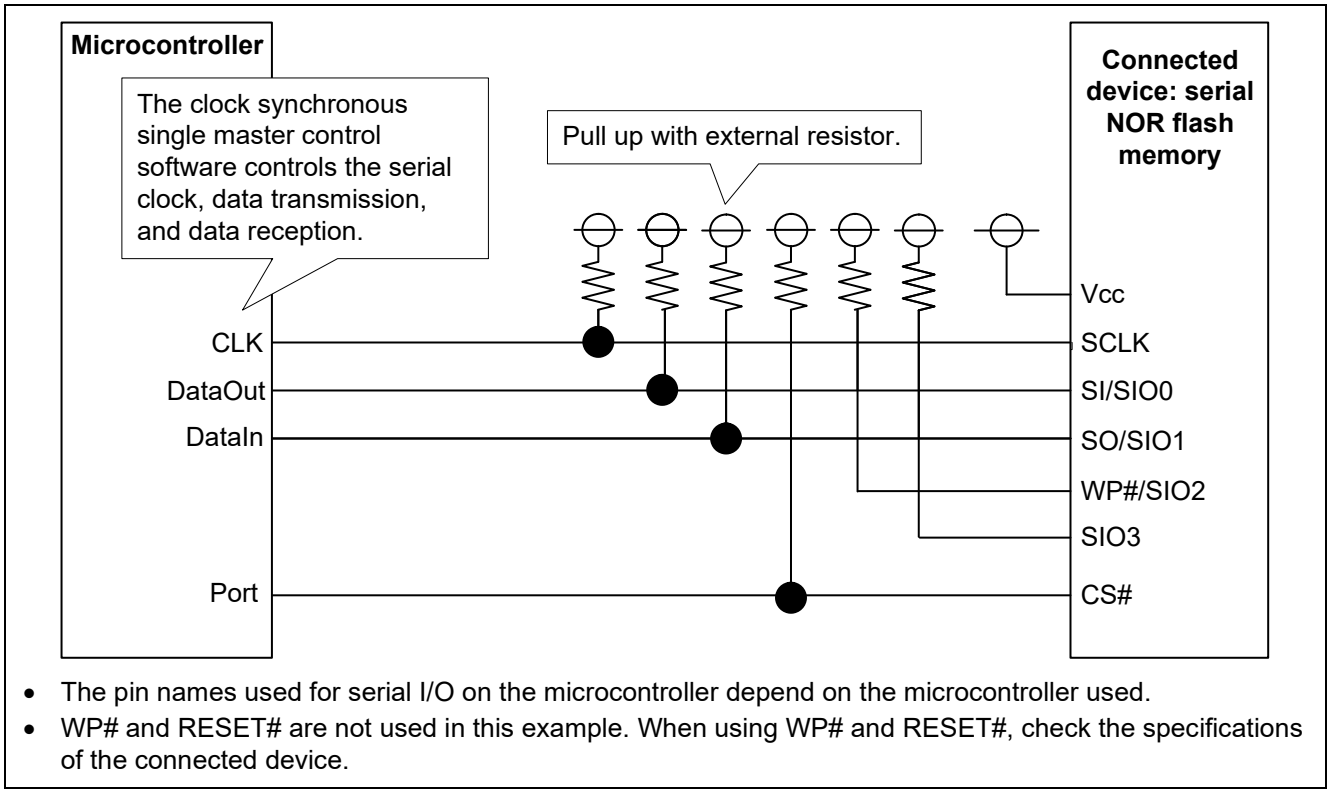
Table 4.1 lists the MCU pins used for single-SPI operation and their functions.

**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:




- The pin names used for serial I/O on the microcontroller depend on the microcontroller used.
- WP# and RESET# are not used in this example. When using WP# and RESET#, check the specifications of the connected device.

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

### 4.1.3 Pin Assignments for Dual-SPI Configuration

Table 4.2 lists the MCU pins used for dual-SPI operation and their functions.

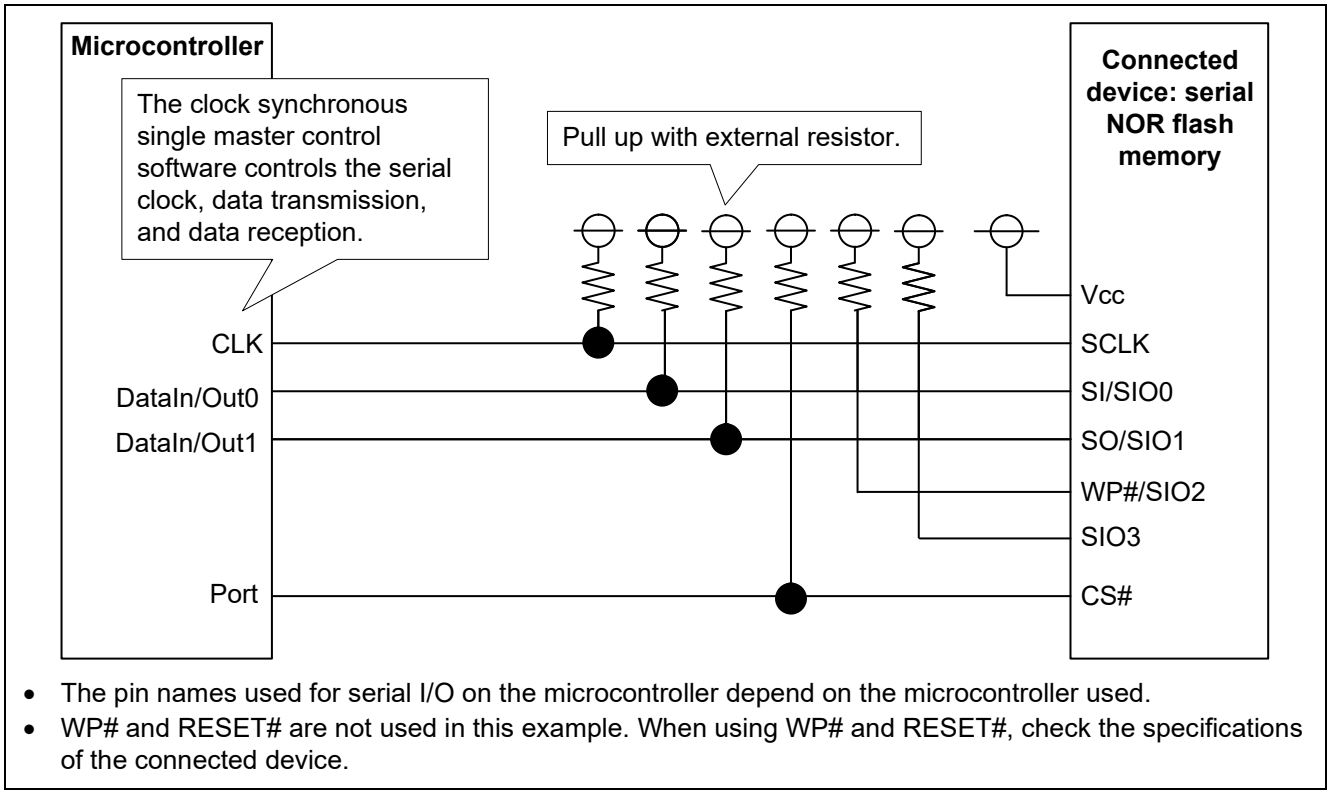
In order to use a dual-SPI configuration, the MCU must have a quad serial peripheral interface function.

**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:




- The pin names used for serial I/O on the microcontroller depend on the microcontroller used.
- WP# and RESET# are not used in this example. When using WP# and RESET#, check the specifications of the connected device.

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

### 4.1.5 Pin Assignments for Quad-SPI Configuration

Table 4.3 lists the MCU pins used for quad-SPI operation and their functions.

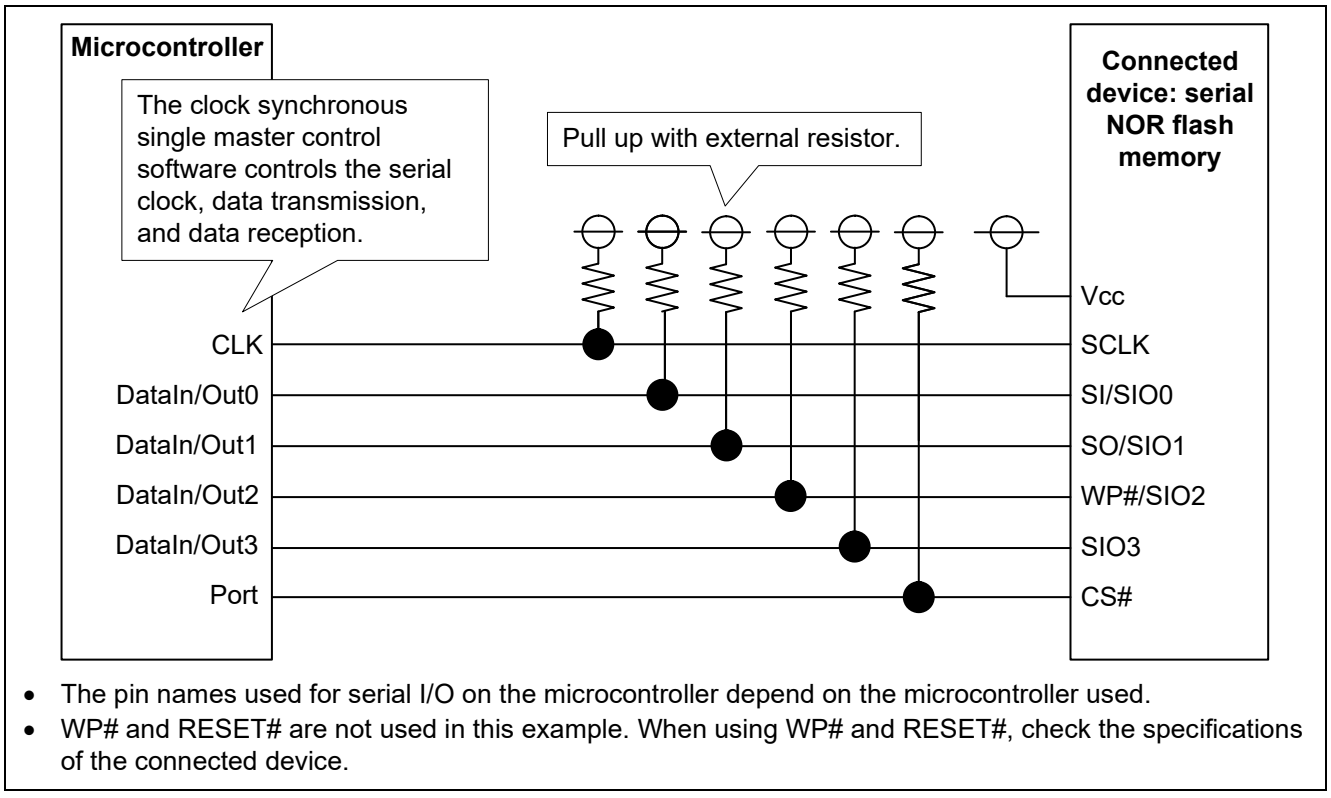
In order to use a quad-SPI configuration, the MCU must have a quad serial peripheral interface function.

**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:




- The pin names used for serial I/O on the microcontroller depend on the microcontroller used.
- WP# and RESET# are not used in this example. When using WP# and RESET#, check the specifications of the connected device.

**Figure 4.3 MCU and 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 NOR flash memory.

The sample code performs the following types of control:

- The CS# pin of the 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.

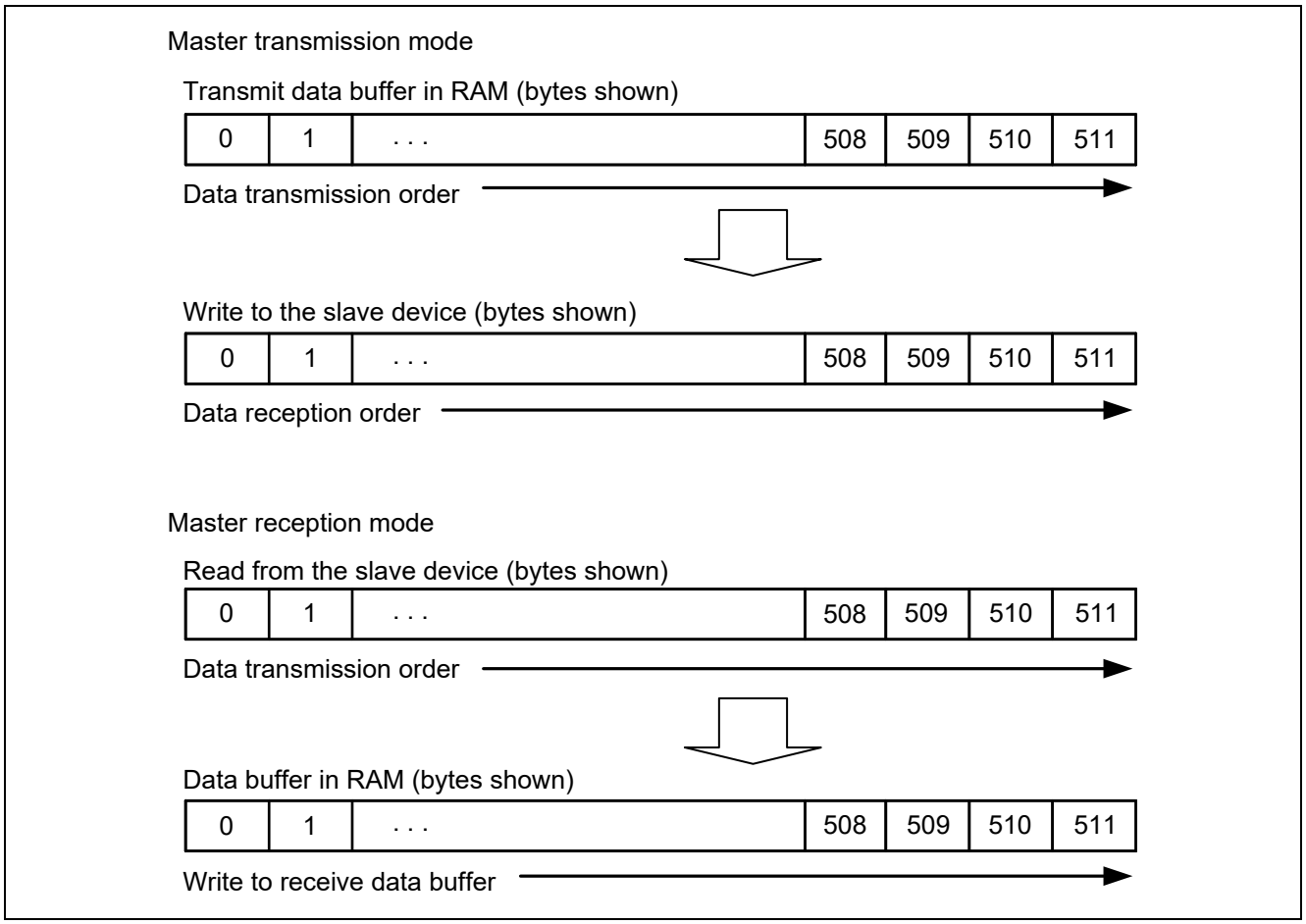


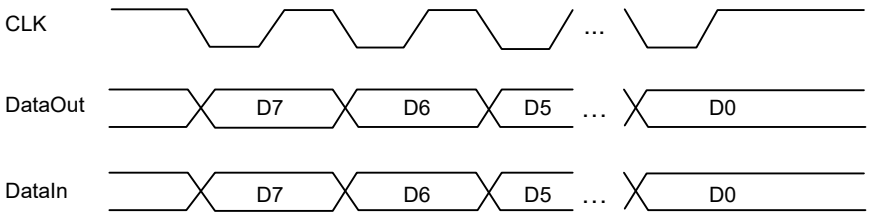

**Figure 5.1 Storage of Transfer Data**

### 5.1.2 Controllable Slave Devices

The timings of controllable slave devices are shown below.

#### (1) Single-SPI Operation

For memory control, the SPI mode 3 (CPOL = 1, CPHA = 1) timings shown in figure 5.2 are used.



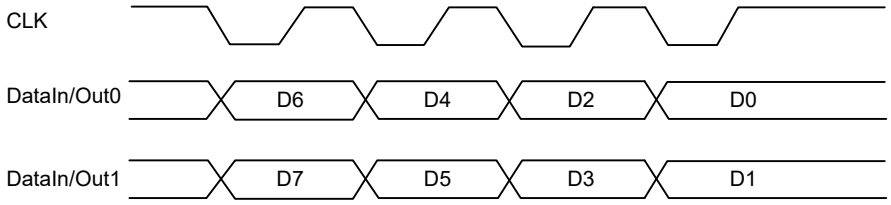

- Transmission from MCU to serial NOR flash memory:
  Output of transmit data starts at falling edge of transfer clock.
- Reception from serial NOR flash memory to MCU:
  Input of receive data starts at rising edge of transfer clock.
- MSB-first transfer

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

For memory control, the SPI mode 3 (CPOL = 1, CPHA = 1) timings shown in figure 5.3 are used.




- Transmission from MCU to serial NOR flash memory:
  Output of transmit data starts at falling edge of transfer clock.
- Reception from serial NOR flash 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**

For memory control, the SPI mode 3 (CPOL = 1, CPHA = 1) timings shown in figure 5.4 are used.

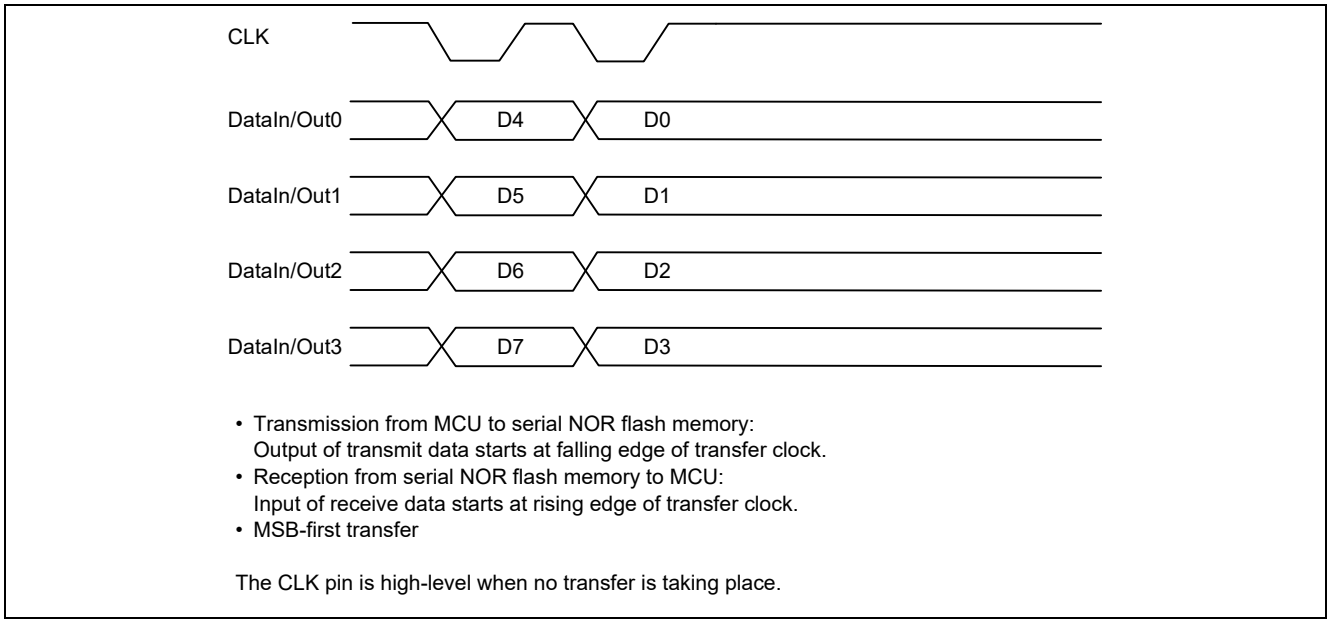



**Figure 5.4 Quad-SPI Clock Synchronous Mode Timing Settings**

### 5.1.3 Serial NOR Flash Memory CS# Pin Control

The CS# pin of the serial NOR flash 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 CS# (MCU port (CS#)) signal of the serial NOR flash memory to the falling edge of the C (MCU CLK) signal of the serial NOR flash memory is controlled by means of software wait to accommodate the CS# setup time of the serial NOR flash memory.

The duration from the rising edge of the C (MCU CLK) signal of the serial NOR flash memory to the rising edge of the CS# (MCU port (CS#)) signal of the serial NOR flash memory controlled by means of software wait to accommodate the CS# hold time of the serial NOR flash memory.

Check the data sheet of the serial NOR flash memory and set the software wait time as appropriate for the system.

### 5.1.4 Serial NOR Flash Memory Instruction Codes

Instruction codes are used to control the serial NOR flash 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) |
| RDCR | Read Configuration Register | 0001 0101 (15 h) |
| RDSCUR | Read Security Register | 0010 1011 (2b h) |
| FAST READ | Read Data Bytes at Higher Speed | 0000 1011 (0b h) |
| DREAD | Dual Output Read | 0011 1011 (3b h) |
| QREAD | Quad Read | 0110 1011 (6b h) |
| PP | Page Program | 0000 0010 (02 h) |
| 4PP | 4 I/O Page Program | 0011 1000 (38 h) |
| SE | Sector Erase (4KB) | 0010 0000 (20 h) |
| BE32K | Block Erase (32KB) | 0101 0010 (52 h) |
| BE | Block Erase (64KB) | 1101 1000 (d8 h) |
| CE | Chip Erase | 0110 0000 (60 h) |
| RDID | Read Identification | 1001 1111 (9f h) |
| EN4B | Enter 4-byte Address Mode | 1011 0111 (b7 h) |
| EXI4B | Exit 4-byte Address Mode | 1110 1001 (e9 h) |

## 5.2 Software Configuration

The sample code operates as upper-layer control software for controlling the serial NOR flash memory (indicated as serial NOR flash memory control software in figure 5.5).

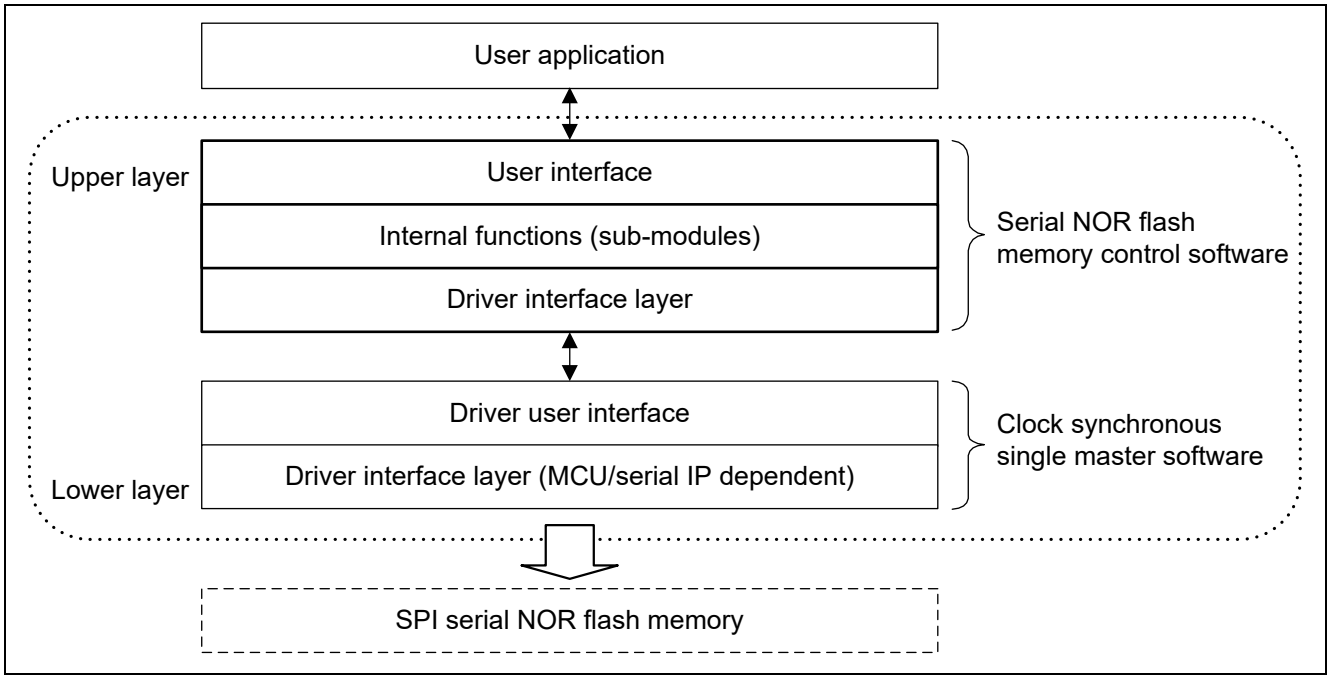



**Figure 5.5 Software Configuration**

## 5.3 Required Memory Size

The required memory sizes are shown below.

### 5.3.1 RX Family

(1) **RX111 RSPI**

**Table 5.2 Required Memory Size**

| Memory Used | Size | Remarks |
|---|---|---|
| ROM | 4,255 bytes (little endian) | r_qspi_flash_mx25l_usr.c<br>r_qspi_flash_mx25l_sub.c<br>r_qspi_flash_mx25l_drvif.c |
| RAM | 6 bytes (little endian) | r_qspi_flash_mx25l_usr.c<br>r_qspi_flash_mx25l_sub.c<br>r_qspi_flash_mx25l_drvif.c |
| Maximum user stack usage | 164 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) **RX64M RSPI**

**Table 5.3 Required Memory Size**

| Memory Used | Size | Remarks |
|---|---|---|
| ROM | 4,251 bytes (little endian) | r_qspi_flash_mx25l_usr.c<br>r_qspi_flash_mx25l_sub.c<br>r_qspi_flash_mx25l_drvif.c |
| RAM | 6 bytes (little endian) | r_qspi_flash_mx25l_usr.c<br>r_qspi_flash_mx25l_sub.c<br>r_qspi_flash_mx25l_drvif.c |
| Maximum user stack usage | 168 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

(1) **RL78/G14 SAU CubeSuite+ Integrated Development Environment (Compiler:CA78K0R)**

**Table 5.4 Required Memory Size**

| Memory Used | Size | Remarks |
|---|---|---|
| ROM | 7,119 bytes | r_qspi_flash_mx25l_usr.c<br>r_qspi_flash_mx25l_sub.c<br>r_qspi_flash_mx25l_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78g14.c |
| RAM | 6 bytes | r_qspi_flash_mx25l_usr.c<br>r_qspi_flash_mx25l_sub.c<br>r_qspi_flash_mx25l_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78g14.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.

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

**Table 5.5 Required Memory Size**

| Memory Used | Size | Remarks |
|---|---|---|
| ROM | 5,672 bytes | r_qspi_flash_mx25l_usr.c<br>r_qspi_flash_mx25l_sub.c<br>r_qspi_flash_mx25l_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78g14.c |
| RAM | 6 bytes | r_qspi_flash_mx25l_usr.c<br>r_qspi_flash_mx25l_sub.c<br>r_qspi_flash_mx25l_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78g14.c |
| Maximum user stack usage | 82 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 SAU IAR Embedded Workbench Integrated Development Environment

**Table 5.6 Required Memory Size**

| Memory Used | Size | Remarks |
|---|---|---|
| ROM | 6,620 bytes | r_qspi_flash_mx25l_usr.c<br>r_qspi_flash_mx25l_sub.c<br>r_qspi_flash_mx25l_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78g14.c |
| RAM | 6 bytes | r_qspi_flash_mx25l_usr.c<br>r_qspi_flash_mx25l_sub.c<br>r_qspi_flash_mx25l_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78g14.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 is the total stack size of the project. It includes the stack size of the lower-layer clock synchronous single master software.

#### (4) RL78/L13 SAU CubeSuite+ Integrated Development Environment (Compiler:CA78K0R)

**Table 5.7 Required Memory Size**

| Memory Used | Size | Remarks |
|---|---|---|
| ROM | 7,218 bytes (little endian) | r_qspi_flash_s25fl_usr.c<br>r_qspi_flash_s25fl_sub.c<br>r_qspi_flash_s25fl_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78l13.c |
| RAM | 6 bytes (little endian) | r_qspi_flash_s25fl_usr.c<br>r_qspi_flash_s25fl_sub.c<br>r_qspi_flash_s25fl_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78l13.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 SAU CS+ for CC Integrated Development Environment (Compiler:CC-RL)

**Table 5.8 Required Memory Size**

| Memory Used | Size | Remarks |
|---|---|---|
| ROM | 5,676 bytes (little endian) | r_qspi_flash_s25fl_usr.c<br>r_qspi_flash_s25fl_sub.c<br>r_qspi_flash_s25fl_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78l13.c |
| RAM | 6 bytes (little endian) | r_qspi_flash_s25fl_usr.c<br>r_qspi_flash_s25fl_sub.c<br>r_qspi_flash_s25fl_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78l13.c |
| Maximum user stack usage | 82 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.

### (6) RL78/L13 SAU IAR Embedded Workbench Integrated Development Environment

**Table 5.9 Required Memory Size**

| Memory Used | Size | Remarks |
|---|---|---|
| ROM | 5,532 bytes (little endian) | r_qspi_flash_s25fl_usr.c<br>r_qspi_flash_s25fl_sub.c<br>r_qspi_flash_s25fl_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78l13.c |
| RAM | 6 bytes (little endian) | r_qspi_flash_s25fl_usr.c<br>r_qspi_flash_s25fl_sub.c<br>r_qspi_flash_s25fl_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78l13.c |
| Maximum user stack usage | 126 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 is the total stack size of the project. It includes the stack size of the lower-layer clock synchronous single master software.

(7) **RL78/G23 SAU CS+ for CC Integrated Development Environment (Compiler:CC-RL)**

**Table 5.10 Required Memory Size**

| Memory Used | Size | Remarks |
|---|---|---|
| ROM | 5,386 bytes | r_qspi_flash_s25fl_usr.c<br>r_qspi_flash_s25fl_sub.c<br>r_qspi_flash_s25fl_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78g23.c |
| RAM | 6 bytes | r_qspi_flash_s25fl_usr.c<br>r_qspi_flash_s25fl_sub.c<br>r_qspi_flash_s25fl_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78g23.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 usable user stack size includes the stack size of the lower-layer clock synchronous single master software.

(8) **RL78/G23 SAU IAR Embedded Workbench Integrated Development Environment**

**Table 5.11 Required Memory Size**

| Memory Used | Size | Remarks |
|---|---|---|
| ROM | 7,527 bytes | r_qspi_flash_s25fl_usr.c<br>r_qspi_flash_s25fl_sub.c<br>r_qspi_flash_s25fl_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78g23.c |
| RAM | 6 bytes | r_qspi_flash_s25fl_usr.c<br>r_qspi_flash_s25fl_sub.c<br>r_qspi_flash_s25fl_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78g23.c |
| Maximum user stack usage | 126 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 is the total stack size of the project. It includes the stack size of the lower-layer clock synchronous single master software.

(9) **RL78/G23 SAU e$^2$ studio Integrated Development Environment (Compiler:LLVM)**

**Table 5.12 Required Memory Size**

| Memory Used | Size | Remarks |
|---|---|---|
| ROM | 4,724 bytes | r_qspi_flash_s25fl_usr.c<br>r_qspi_flash_s25fl_sub.c<br>r_qspi_flash_s25fl_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78g23.c |
| RAM | 6 bytes | r_qspi_flash_s25fl_usr.c<br>r_qspi_flash_s25fl_sub.c<br>r_qspi_flash_s25fl_drvif.c<br>r_qspi_flash_s25fl_sfr_rl78g23.c |
| Maximum user stack usage | 78 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.

## 5.4 File Structure

Table 5.12 lists the files used by the sample code.

**Table 5.13 File Structure**

| | |
|---|---|
| \r01an1967xx0104-mcu-serial <DIR> | Sample code folder |
| r01an1967ej0104-mcu-serial.pdf | Application note (this document) |
| r01an1967jj0104-mcu-serial.pdf | Application note (Japanese) |
| \source <DIR> | Program storage folder |
| \r_qspi_flash_mx25l <DIR> | Serial NOR flash memory control software folder |
| r_qspi_flash_mx25l.h | Header file |
| r_qspi_flash_mx25l_drvif.c | Driver interface source file |
| r_qspi_flash_mx25l_drvif.h | Driver interface header file |
| r_qspi_flash_mx25l_sfr.h.rl78g1c | Common definition for registers (RL78/G1C) |
| r_qspi_flash_mx25l_sfr.h.rl78g14 | Common definition for registers (RL78/G14) |
| r_qspi_flash_mx25l_sfr.h.rl78l1c | Common definition for registers (RL78/L1C) |
| r_qspi_flash_mx25l_sfr.h.rl78l12 | Common definition for registers (RL78/L12) |
| r_qspi_flash_mx25l_sfr.h.rl78l13 | Common definition for registers (RL78/L13) |
| r_qspi_flash_mx25l_sfr.h.rl78g23 | Common definition for registers (RL78/G23) |
| r_qspi_flash_mx25l_sfr.h.rx63n | Common definition for registers (RX63N) |
| r_qspi_flash_mx25l_sfr.h.rx64m | Common definition for registers (RX64M) |
| r_qspi_flash_mx25l_sfr.h.rx111 | Common definition for registers (RX111) |
| r_qspi_flash_mx25l_sfr_rl78g1c.c | Common definition source file for registers (RL78/G1C) |
| r_qspi_flash_mx25l_sfr_rl78g14.c | Common definition source file for registers (RL78/G14) |
| r_qspi_flash_mx25l_sfr_rl78l1c.c | Common definition source file for registers (RL78/L1C) |
| r_qspi_flash_mx25l_sfr_rl78l12.c | Common definition source file for registers (RL78/L12) |
| r_qspi_flash_mx25l_sfr_rl78l13.c | Common definition source file for registers (RL78/L13) |
| r_qspi_flash_mx25l_sfr_rl78g23.c | Common definition source file for registers (RL78/G23) |
| r_qspi_flash_mx25l_sub.c | Internal function source file |
| r_qspi_flash_mx25l_sub.h | Internal function header file |
| r_qspi_flash_mx25l_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

Table 5.13 lists the return value used in the sample code.

**Table 5.14 Return Value (Refer to r_qspi_flash_mx25l.h)**

| Constant Name | Setting Value | Contents |
|---|---|---|
| FLASH_OK | (error_t)( 0) | Successful operation |
| FLASH_ERR_PARAM | (error_t)(-1) | Parameter error |
| FLASH_ERR_HARD | (error_t)(-2) | Hardware error |
| FLASH_ERR_WP | (error_t)(-4) | Write-protection error |
| FLASH_ERR_TIMEOUT | (error_t)(-6) | Timeout error |
| FLASH_ERR_OTHER | (error_t)(-7) | Other error |

### 5.5.2 Command Definitions

Table 5.14 lists the command definitions used in the sample code.

**Table 5.15 Command Definitions (Refer to r_qspi_flash_mx25l_sub.c)**

| Constant Name | Setting Value | Contents |
|---|---|---|
| FLASH_CMD_WREN | (uint8_t)(0x06) | Write Enable |
| FLASH_CMD_WRDI | (uint8_t)(0x04) | Write Disable |
| FLASH_CMD_RDSR | (uint8_t)(0x05) | Read Status Register |
| FLASH_CMD_WRSR | (uint8_t)(0x01) | Write Status Register |
| FLASH_CMD_RDCR | (uint8_t)(0x15) | Read Configuration Register |
| FLASH_CMD_RDSCUR | (uint8_t)(0x2b) | Read Security Register |
| FLASH_CMD_FREAD | (uint8_t)(0x0b) | Read Data at Higher Speed |
| FLASH_CMD_DREAD | (uint8_t)(0x3b) | Dual Read (Single → Dual Output) |
| FLASH_CMD_QREAD | (uint8_t)(0x6b) | Quad Read (Single → Quad Output) |
| FLASH_CMD_PP | (uint8_t)(0x02) | Page Program (Single → Single Input) |
| FLASH_CMD_4PP | (uint8_t)(0x38) | Quad Page Program (Single → Quad Input) |
| FLASH_CMD_SE | (uint8_t)(0x20) | Sector Erase (4KB) |
| FLASH_CMD_BE32K | (uint8_t)(0x52) | Block Erase (32KB) |
| FLASH_CMD_BE64K | (uint8_t)(0xd8) | Block Erase (64KB) |
| FLASH_CMD_CE | (uint8_t)(0x60) | Chip Erase |
| FLASH_CMD_RDID | (uint8_t)(0x9f) | Read Identification |
| FLASH_CMD_EN4B | (uint8_t)(0xb7) | Enter 4-byte Address Mode |
| FLASH_CMD_EX4B | (uint8_t)(0xe9) | Exit 4-byte Address Mode |

### 5.5.3 Other Definitions

The values of other definitions used in the sample code are listed in tables Table 5.15 to Table 5.19.

**Table 5.16 Values Defined in r_qspi_flash_mx25l.h**

| Constant Name | Setting Value | Contents |
|---|---|---|
| FLASH_DEV_NUM | (1) | Number of connected devices |
| FLASH_DEV0 | (0) | Device number 0 |
| FLASH_DEV1 | (1) | Device number 1 |
| FLASH_DELAY_TASK | (uint8_t)(1) | Wait time of delay task [unit: ms]* |
| FLASH_LOG_ERR | (1) | Log Type: Error |
| FLASH_TRUE | (uint8_t)(0x01) | Flag "ON" |
| FLASH_FALSE | (uint8_t)(0x00) | Flag "OFF" |
| FLASH_MODE_S_ERASE | (uint8_t)(1) | Erase Mode: Sector Erase (4 KB) |
| FLASH_MODE_BE32K_ERASE | (uint8_t)(2) | Erase Mode: Block Erase (32 KB) |
| FLASH_MODE_BE64K_ERASE | (uint8_t)(3) | Erase Mode: Block Erase (64 KB) |
| FLASH_MODE_C_ERASE | (uint8_t)(4) | Erase Mode: Chip Erase |
| FLASH_MODE_3BYTE | (uint8_t)(0) | Address Mode: 3-byte Address Mode |
| FLASH_MODE_4BYTE | (uint8_t)(1) | Address Mode: 4-byte Address Mode |
| FLASH_MODE_REG_WRITE | (uint8_t)(0) | Wait Mode: Register write mode |
| FLASH_MODE_PROG_ERASE | (uint8_t)(1) | Wait Mode: Page Program or Erase mode |
| FLASH_MEM_SIZE | (uint32_t)(33554432) | Memory size (byte units)<br>Value at left corresponds to size of 256 Mbit. |
| FLASH_SECT_ADDR | (uint32_t)(0xfffff000) | Sector address mask value for sector erase |
| FLASH_BE32K_ADDR | (uint32_t)(0xffff8000) | Sector address mask value for sector erase |
| FLASH_BE64K_ADDR | (uint32_t)(0xffff0000) | Sector address mask value for sector erase |
| FLASH_PAGE_SIZE | (uint32_t)(256) | Page size (byte units) |
| FLASH_ADDR_SIZE | (uint8_t)(4) | Address size (byte units)<br>Value at left corresponds to size of 256 Mbit or more. |
| FLASH_WP_WHOLE_MEM | (uint8_t)(0x0f) | Whole-chip write protect |
| FLASH_FULL_CHIP_ERASE | FLASH_MODE_C_ ERASE | Supported erase-all command |
| FLASH_ADDR_MODE | FLASH_MODE_4BYTE | Addressability Mode<br>Value at left corresponds to size of 256 Mbit or more. |
| FLASH_CMD_SIZE | (uint8_t)1 | Command size (byte units) |
| FLASH_STSREG_SIZE | (uint16_t)1 | Status register size (byte units) |
| FLASH_CFGREG_SIZE | (uint16_t)1 | Configuration register size (byte units) |
| FLASH_SCURREG_SIZE | (uint16_t)1 | Security register size (byte units) |
| FLASH_WSTSREG_SIZE | (uint16_t)2 | Write status register size (byte units) |
| FLASH_IDDATA_SIZE | (uint16_t)3 | ID data size (byte units) |

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

**Table 5.17 Values Defined in r_qspi_flash_mx25l_sfr.h.rx111**

| Constant Name | Setting Value | Contents |
|---|---|---|
| FLASH_DR_CS0 | PORT5.PODR.BIT.B5 | Device number 0 port output data register SFR definition |
| FLASH_DDR_CS0 | PORT5.PDR.BIT.B5 | Device number 0 port direction register SFR definition |
| FLASH_DR_CS1 | — | Device number 1 port output data register SFR definition (This setting is needed when controlling two devices.) |
| FLASH_DDR_CS1 | — | Device number 1 port direction register SFR definition (This setting is needed when controlling two devices.) |
| FLASH_HI | (uint8_t)(0x01) | Port "H" |
| FLASH_LOW | (uint8_t)(0x00) | Port "L" |
| FLASH_OUT | (uint8_t)(0x01) | Port Output Setting |
| FLASH_IN | (uint8_t)(0x00) | Port Input Setting |
| FLASH_BR | (uint8_t)(0x01) | Transfer rate for command transmission* |
| FLASH_BR_WRITE_DATA | (uint8_t)(0x01) | Transfer rate for data transmission* |
| FLASH_BR_READ_DATA | (uint8_t)(0x01) | Transfer rate for data reception* |

Note: * 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 32 [MHz] and a transfer rate of 16 [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 32 [MHz] and a transfer rate of 4 [MHz].

**Table 5.18 Values Defined in r_qspi_flash_mx25l_sfr.h.rl78**

| Constant Name | Setting Value | Contents |
|---|---|---|
| FLASH_DR_CS0 | P8.0 | Device number 0 port register SFR definition |
| FLASH_DDR_CS0 | PM8.0 | Device number 0 port mode register SFR definition |
| FLASH_DR_CS1 | — | Device number 1 port output data register SFR definition (This setting is needed when controlling two devices.) |
| FLASH_DDR_CS1 | — | Device number 1 port direction register SFR definition (This setting is needed when controlling two devices.) |
| FLASH_HI | (uint8_t)(0x01) | Port "H" |
| FLASH_LOW | (uint8_t)(0x00) | Port "L" |
| FLASH_OUT | (uint8_t)(0x00) | Port Output Setting |
| FLASH_IN | (uint8_t)(0x01) | Port Input Setting |
| FLASH_BR | (uint8_t)(0x01) | Transfer rate for command transmission* |
| FLASH_BR_WRITE_DATA | (uint8_t)(0x01) | Transfer rate for data transmission* |
| FLASH_BR_READ_DATA | (uint8_t)(0x01) | Transfer rate for data reception* |

Note: * 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 5.19 Values Defined in r_qspi_flash_mx25l_sub.c**

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

**Table 5.20 Values Defined in r_qspi_flash_mx25l_sub.h**

| Constant Name | Setting Value | Contents |
|---|---|---|
| FLASH_SE_BUSY_WAIT | (uint32_t)(200) | Sector Erase (4 KB) Busy Timeout<br>200 × 1 ms = 200 ms |
| FLASH_BE32K_BUSY_WAIT | (uint32_t)(1000) | Block Erase (32 KB) Busy Timeout<br>1,000 × 1 ms = 1 s |
| FLASH_BE64K_BUSY_WAIT | (uint32_t)(2000) | Block Erase (64 KB) Busy Timeout<br>20,000 × 1 ms = 1 s |
| FLASH_CE_BUSY_WAIT | (uint32_t)(1200000) | Chip Erase Busy Timeout<br>1,200,000 × 1 ms =1,200 s |
| FLASH_PP_BUSY_WAIT | (uint32_t)(3000) | Page Program Timeout<br>3,000 × 1 us = 3 ms |
| FLASH_WR_BUSY_WAIT | (uint32_t)(40000) | Write Register Timeout<br>40,000 × 1 us = 40 ms |
| FLASH_T_WBUSY_WAIT | (uint16_t)MTL_T_1US | Write Busy Polling Time |
| FLASH_T_PBUSY_WAIT | (uint16_t)MTL_T_1MS | Page Program Busy Polling Time |
| FLASH_T_EBUSY_WAIT | (uint16_t)MTL_T_1MS | Erase Busy Polling Time |
| FLASH_T_CS_HOLD | (uint16_t)MTL_T_1US | CS Stability Waiting Time |
| FLASH_T_R_ACCESS | (uint16_t)MTL_T_1US | Reading Start Waiting Time |
| FLASH_REG_SRWD | (uint8_t)(0x80) | Status Register Write Disable |
| FLASH_REG_QE | (uint8_t)(0x40) | Quad Enable Bit |
| FLASH_REG_BP3 | (uint8_t)(0x20) | Block Protection Bit3 |
| FLASH_REG_BP2 | (uint8_t)(0x10) | Block Protection Bit2 |
| FLASH_REG_BP1 | (uint8_t)(0x08) | Block Protection Bit1 |
| FLASH_REG_BP0 | (uint8_t)(0x04) | Block Protection Bit0 |
| FLASH_REG_WEL | (uint8_t)(0x02) | Write Enable Latch Bit |
| FLASH_REG_WIP | (uint8_t)(0x01) | Write In Progress Bit |
| FLASH_REG_MASK | (uint8_t)(0xfc) | Write status fixed data |
| FLASH_CNFG_DC1 | (uint8_t)(0x80) | Dummy Cycle 1 |
| FLASH_CNFG_DC0 | (uint8_t)(0x40) | Dummy Cycle 0 |
| FLASH_CNFG_4BYTE | (uint8_t)(0x20) | 4-byte Address Mode |
| FLASH_CNFG_RSV | (uint8_t)(0x10) | Reserved |
| FLASH_CNFG_TB | (uint8_t)(0x08) | Top/Bottom |
| FLASH_CNFG_ODS2 | (uint8_t)(0x04) | Output Driver Strength 2 |
| FLASH_CNFG_ODS1 | (uint8_t)(0x02) | Output Driver Strength 1 |
| FLASH_CNFG_ODS0 | (uint8_t)(0x01) | Output Driver Strength 0 |
| FLASH_SCUR_E_FAIL | (uint8_t)(0x40) | Erase Failed |
| FLASH_SCUR_P_FAIL | (uint8_t)(0x20) | Program Failed |

## 5.6 Structure/Union List

Figures 5.6 and 5.7 show the Structure/Union Used in the Sample Code.

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

**Figure 5.6 Union Used in the Sample Code (Refer to r_qspi_flash_mx25l_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_flash_info_t;
```

**Figure 5.7 Structure Used in the Sample Code (Refer to r_qspi_flash_mx25l.h)**

**Table 5.16 Description of Structure "r_qspi_flash_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)<br>Read: Read data counter temp. (max. 32 KB) |
| rsv[2] | (Setting has no effect.) | For alignment adjustment |
| pData | — | Data storage buffer pointer<br>Write: Storage source of data to be written in serial NOR flash memory<br>Read: Storage destination of data to be read from serial NOR flash memory |

## 5.7 Variable

Table 5.17 lists the static variable.

**Table 5.17 Static Variable (Refer to r_qspi_flash_mx25l_sub.c)**

| Type | Variable Name | Contents | Function Used |
|---|---|---|---|
| STATIC uint8_t | g_flash_cmdbuf[6] | Command buffer | r_qspi_flash_send_cmd<br>r_qspi_flash_set_cmd |

## 5.8 Functions

Table 5.18 lists the functions.

**Table 5.18 Functions**

| Function Name | Outline |
|---|---|
| R_QSPI_FLASH_Init_Driver() | Driver initialization processing |
| R_QSPI_FLASH_Read_Status() | Status register read processing |
| R_QSPI_FLASH_Read_Configuration() | Configuration register read processing |
| R_QSPI_FLASH_Write_Configuration() | Configuration register write processing |
| R_QSPI_FLASH_Read_Security() | Security register read processing |
| R_QSPI_FLASH_Set_Write_Protect() | Write protect setting processing |
| R_QSPI_FLASH_Quad_Enable() | Quad mode enable setting processing |
| R_QSPI_FLASH_Quad_Disable() | Quad mode disable setting processing |
| R_QSPI_FLASH_Write_Di() | WRDI command issue processing |
| R_QSPI_FLASH_Read_Data() | Data read processing |
| R_QSPI_FLASH_Write_Data() | Data write processing |
| R_QSPI_FLASH_Write_Data_Page() | Data write processing (for single-page write) |
| R_QSPI_FLASH_Erase() | Erase processing |
| R_QSPI_FLASH_Read_ID() | ID read processing |
| R_QSPI_FLASH_Wait() | Busy wait processing |
| R_QSPI_FLASH_Set_4byte_Address_Mode() | 4-byte address mode setting 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_FLASH_Init_Driver | |
|---|---|
| **Outline** | Driver initialization processing |
| **Header** | r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h |
| **Declaration** | error_t R_QSPI_FLASH_Init_Driver(void) |
| **Description** | • Calls the r_qspi_flash_init_port() function to initialize the CS# pin.<br>• Calls the initialization function of the clock synchronous single master control software to initialize the I/O ports.<br>• Call this function once at system startup |
| **Arguments** | None |
| **Return Value** | The initialization result is returned.<br>FLASH_OK ; Successful operation<br>FLASH_ERR_OTHER ; Other error<br>The return value of r_qspi_flash_drvif_init_driver() is returned. |

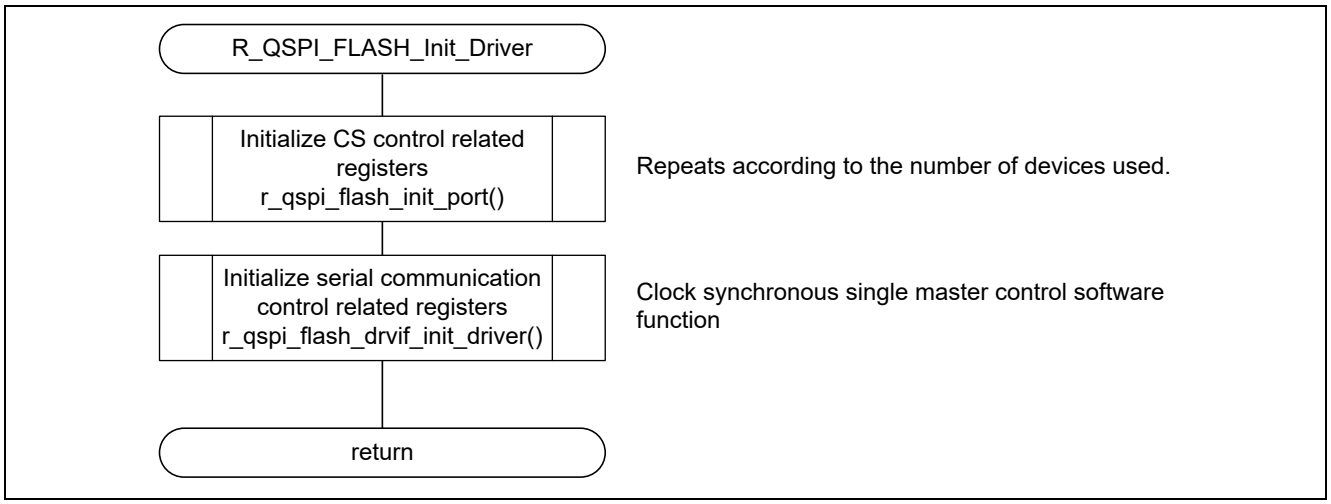



**Figure 5.8 Overview of Driver Initialization Processing**

### 5.9.2 Status Register Read Processing

| R_QSPI_FLASH_Read_Status | |
|---|---|
| **Outline** | Status register read processing |
| **Header** | r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h |
| **Declaration** | error_t R_QSPI_FLASH_Read_Status(uint8_t DevNo, uint8_t FAR* pStatus) |
| **Description** | • Reads the status register and stores the result in pStatus.<br>Set 1 byte as a read buffer.<br>• Stores the following information in the read buffer (pStatus):<br>Bit 7: Status register write enable/disable (SRWD)<br>1: Disable writing to the status register.<br>0: Enable writing to the status register.<br>Bit 6: Quad Enable (QE)<br>1: Quad Enable (Performs Quad I/O mode and WP#, RESET# are disabled.)<br>0: Not Quad Enable (Performs non-Quad I/O mode and WP#, RESET# are enabled.)<br>Bits 5 to 2: Block protect 3-0 (BP3-BP0)<br>Set to 1, a designated memory area is protected from PROGRAM and ERASE operations.<br>Bit 1: Write enable latch (WEL)<br>1: Internal Write Enable Latch is set.<br>0: Internal Write Enable Latch is reset.<br>Bit 0: Write in progress (WIP)<br>1: Program or Erase cycle is in progress.<br>0: No Program or No Erase cycle is in progress<br>• Refer to the data sheet of the serial NOR flash memory for the relationship between protect areas and protect bits. |
| **Arguments** | uint8_t DevNo ; Device number<br>uint8_t FAR* pStatus ; Read buffer pointer |
| **Return Value** | The status register fetch result is returned.<br>FLASH_OK ; Successful operation<br>FLASH_ERR_PARAM ; Parameter error<br>FLASH_ERR_HARD ; Hardware error<br>FLASH_ERR_OTHER ; Other error |

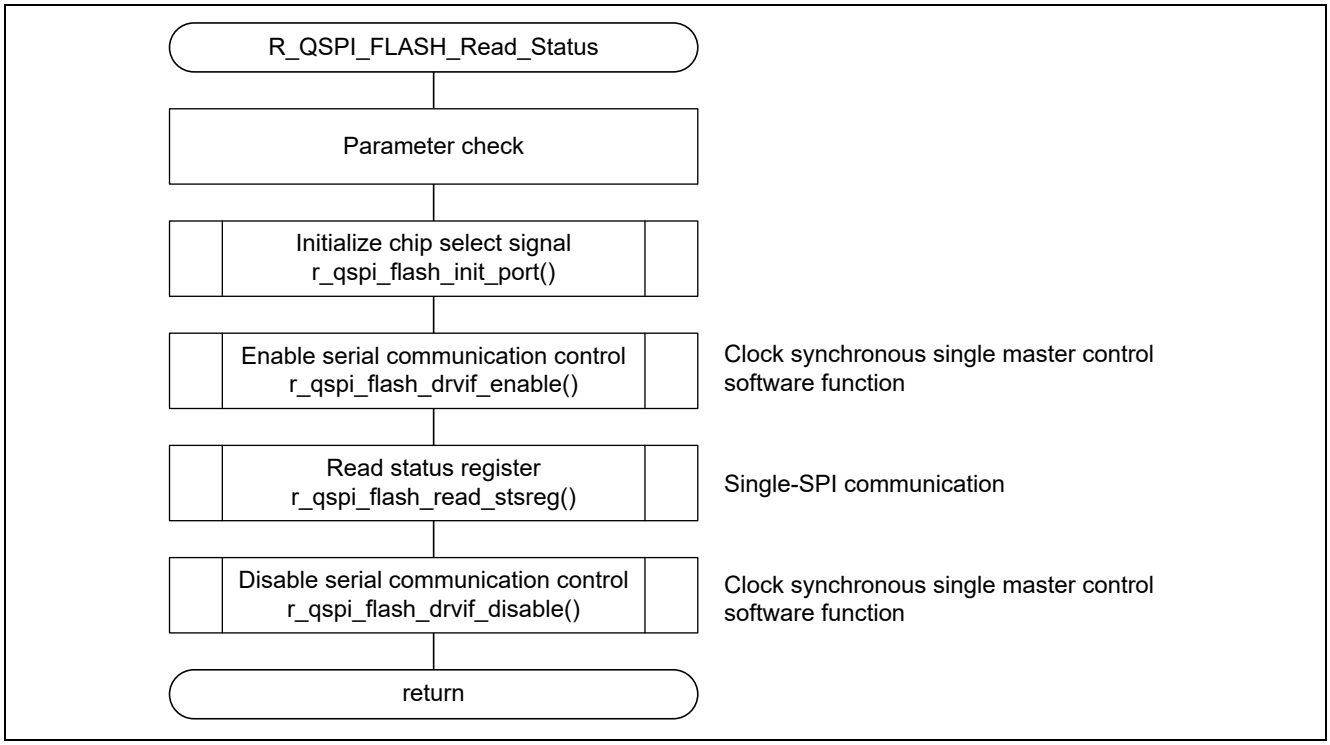



**Figure 5.9 Overview of Status Register Read Processing**

### 5.9.3 Configuration Register Read Processing

| R_QSPI_FLASH_Read_Configuration | |
|---|---|
| **Outline** | Configuration register read processing |
| **Header** | r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h |
| **Declaration** | error_t R_QSPI_FLASH_Read_Configuration(uint8_t DevNo, uint8_t FAR* pConfig) |
| **Description** | • Reads the configuration register and stores the result in pConfig. Set 1 byte as a read buffer.<br>• Stores the following information in the read buffer (pConfig):<br>Bits 7 to 6: DC1-DC0 (Dummy cycle)<br>See the specification of the Flash memory.<br>Bit 5: 4BYTE (4BYTE Indicator)<br>1: 4-byte address mode<br>0: 3-byte address mode<br>Bit 4: Reserved<br>Bit 3: TB (Top/Bottom)<br>1: Bottom area protect<br>0: Top area protect<br>Bits 2 to 0: ODS2-ODS0 (Output driver strength)<br>See the specification of the Flash memory. |
| **Arguments** | uint8_t DevNo ; Device number<br>uint8_t FAR* pConfig ; Read buffer pointer |
| **Return Value** | The configuration register fetch result is returned.<br>FLASH_OK ; Successful operation<br>FLASH_ERR_PARAM ; Parameter error<br>FLASH_ERR_HARD ; Hardware error<br>FLASH_ERR_OTHER ; Other error |

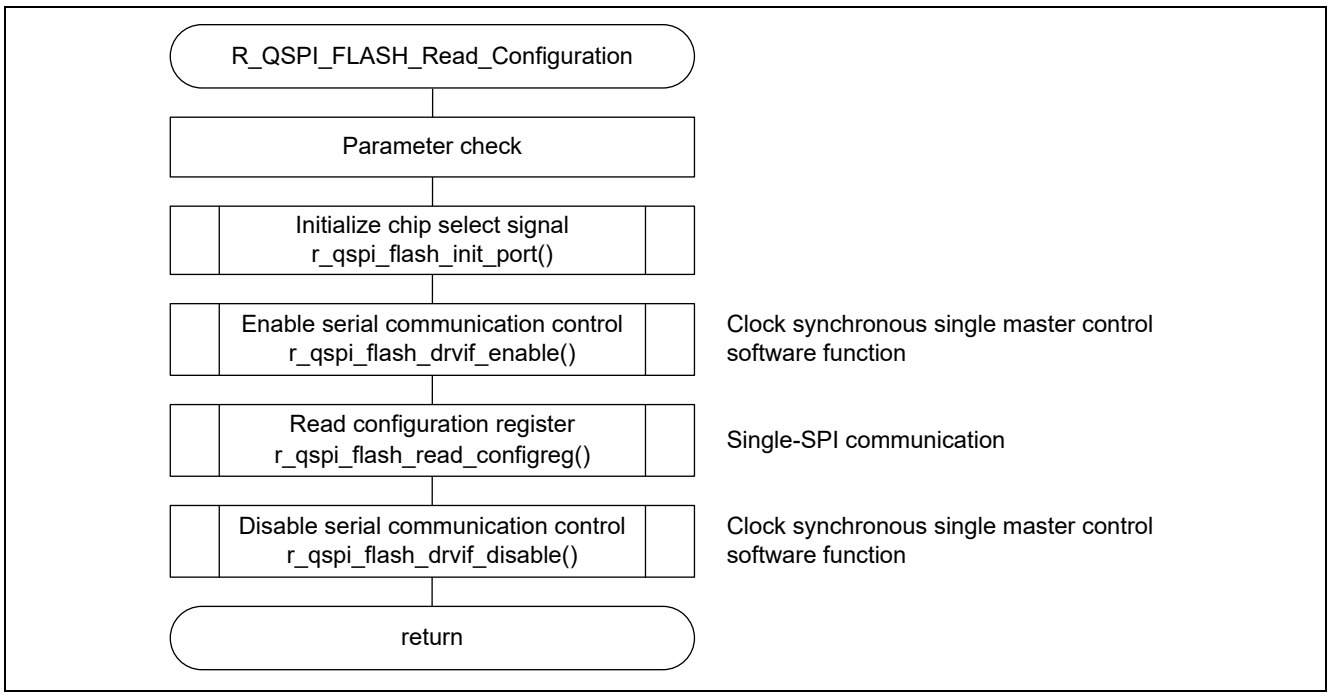



**Figure 5.10 Overview of Configuration Register Read Processing**

### 5.9.4 Configuration Register Write Processing

| R_QSPI_FLASH_Write_Configuration | |
|---|---|
| **Outline** | Configuration register write processing |
| **Header** | r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h |
| **Declaration** | error_t R_QSPI_FLASH_Write_Configuration(uint8_t DevNo, uint8_t FAR* pConfig) |
| **Description** | • Writes the pConfig value to the configuration register. Set one byte as the write buffer.<br>• Store the following information in the write buffer (pConfig). Note that this information will differ according to the device used. Refer to the data sheet of the device.<br>Bits 7 to 6: DC1-DC0 (Dummy cycle)<br>See the specification of the Flash memory.<br>Bit 5: 4BYTE (4BYTE Indicator)<br>1: 4-byte address mode<br>0: 3-byte address mode<br>Bit 4: PBE (Preamble bit Enable)<br>0: Disable<br>1: Enable<br>Bit 3: TB (Top/Bottom)<br>1: Bottom area protect<br>0: Top area protect<br>Bits 2 to 0: ODS2-ODS0 (Output driver strength)<br>See the specification of the Flash memory<br>• When setting the value in the write buffer, first read the value of the configuration register beforehand, and make sure not to alter values other than those you wish to change.<br>• After processing finishes, read the configuration register to confirm that the value was written correctly.<br>• The 4BYTE bit is read-only, so it cannot be written to. The setting is ignored. The value of this bit can be changed using R_QSPI_FLASH_Set_4byte_Address_Mode().<br>• There are two methods of waiting for the write to finish. These are shown below. Note that the next processing (write, read, erase, etc.) should be run after confirming that the write has finished.<br>< Using this user API to wait for the write to finish ><br>Enable FLASH_WAIT_READY in r_qspi_flash_mx25l.h.<br>< Not using this user API to wait for the write to finish ><br>Disable FLASH_WAIT_READY in r_qspi_flash_mx25l.h, and call R_QSPI_FLASH_Wait() after processing by this user API has finished. This processing method allows you to perform write-end wait confirmation at any time you wish. Refer to figure 5.12 for instructions. |
| **Arguments** | uint8_t DevNo ; Device number<br>uint8_t FAR* pConfig ; Write buffer pointer |
| **Return Value** | The configuration register write result is returned.<br>FLASH_OK ; Successful operation<br>FLASH_ERR_PARAM ; Parameter error<br>FLASH_ERR_HARD ; Hardware error<br>FLASH_ERR_TIMEOUT ; Timeout error (FLASH_WAIT_READY enabled)<br>FLASH_ERR_OTHER ; Other error |

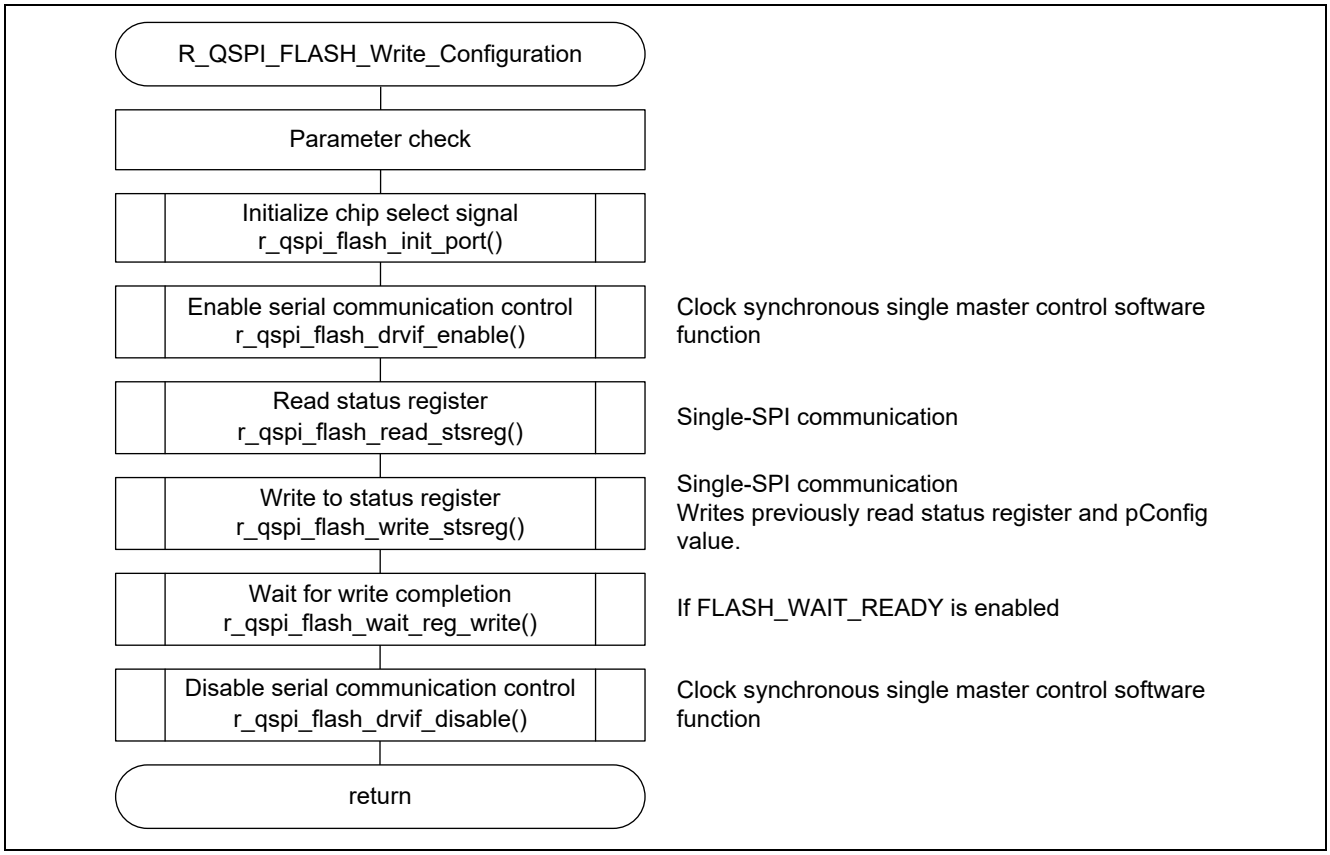



**Figure 5.11 Overview of Configuration Register Write Processing**

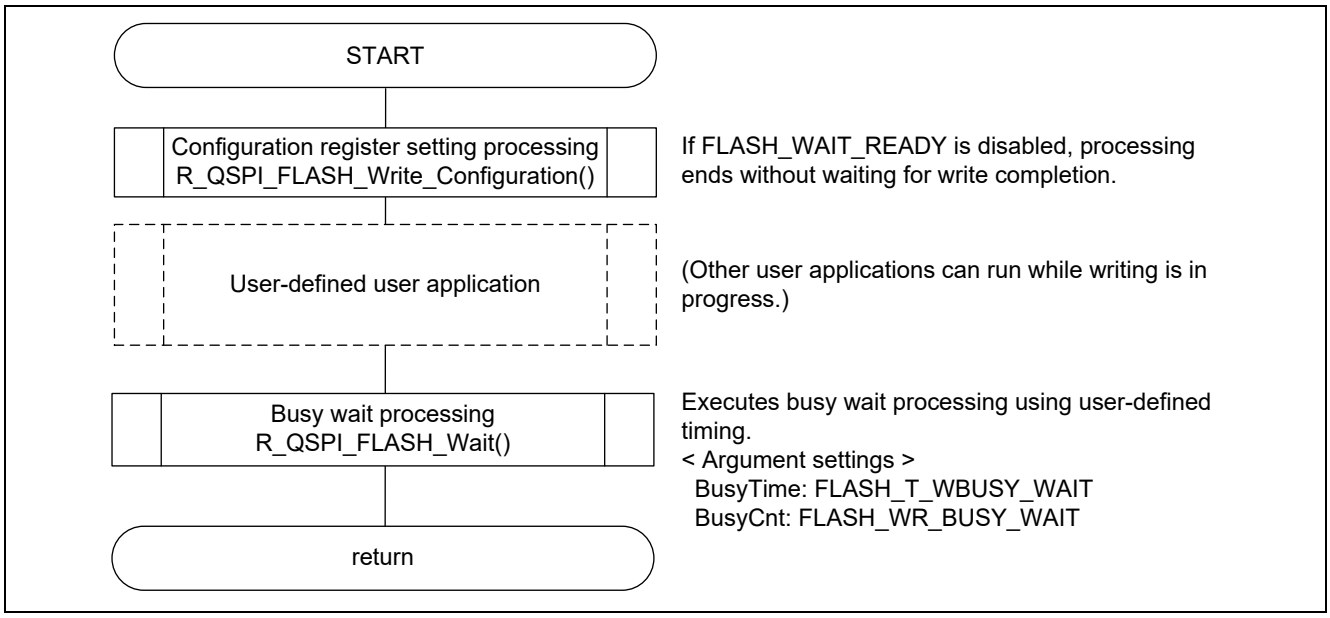



**Figure 5.12 Configuration Register Write-End Wait Processing Using R_QSPI_FLASH_Wait()**

### 5.9.5 Security Register Read Processing

| R_QSPI_FLASH_Read_Security | |
|---|---|
| **Outline** | Security register read processing |
| **Header** | r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h |
| **Declaration** | error_t R_QSPI_FLASH_Read_Security(uint8_t DevNo, uint8_t FAR* pScur) |
| **Description** | • Reads the security register and stores the data in pScur. Set one byte as the read buffer.<br>• The following information is stored in the read buffer (pScur):<br>Bit 7: WPSEL<br>1: Individual mode<br>0: Normal WP mode<br>Bit 6: E_FAIL<br>1: Erase failed<br>0: Erase succeed<br>Bit 5: P_FAIL<br>1: Program failed<br>0: Program succeed<br>Bit 4: Reserved<br>Bit 3: ESB (Erase Suspend Bit)<br>1: Erase Suspended<br>0: Erase is not suspended<br>Bit 2: PSB (Program Suspend Bit)<br>1: Program Suspended<br>0: Program is not suspended<br>Bit 1: LDSO (Indicate if lock-down)<br>1: Lock-down (Cannot program/erase OTP)<br>0: Not lock-down<br>Bit 0: Secured OTP indicator<br>1: Factory lock<br>0: Non-factory lock.<br>• When the value of P_FAIL is 1, it is cleared to 0 the next time program processing succeeds.<br>• When the value of E_FAIL is 1, it is cleared to 0 the next time erase processing succeeds. |
| **Arguments** | uint8_t DevNo ; Device number<br>uint8_t FAR* pConfig ; Read buffer pointer |
| **Return Value** | The security register fetch result is returned.<br>FLASH_OK ; Successful operation<br>FLASH_ERR_PARAM ; Parameter error<br>FLASH_ERR_HARD ; Hardware error<br>FLASH_ERR_OTHER ; Other error |

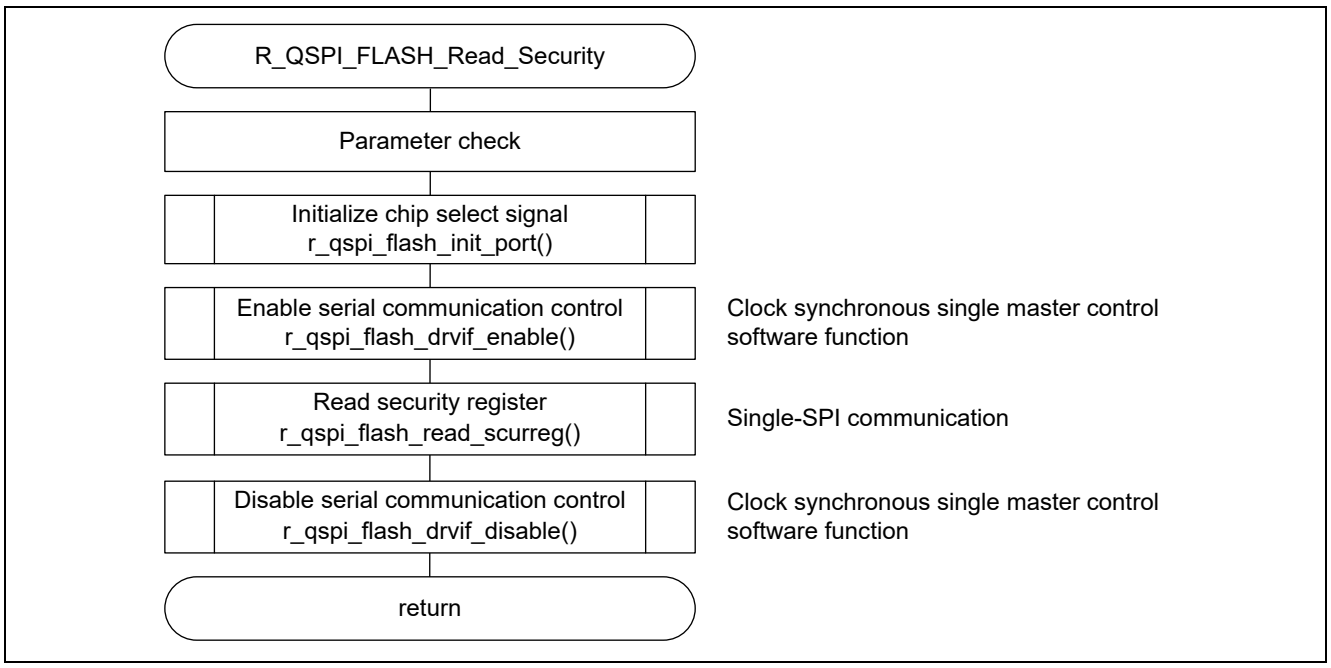



**Figure 5.13 Overview of Security Register Read Processing**

### 5.9.6 Write Protect Setting Processing

R_QSPI_FLASH_Set_Write_Protect

**Outline** Write protect setting processing

**Header** r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h

**Declaration** error_t R_QSPI_FLASH_Set_Write_Protect(uint8_t DevNo, uint8_t WpSts)

**Description**

- Makes write protect settings. Clears SRWD to 0.
- Make settings using the following write protect setting data (WpSts):

| WpSts | BP3 | TB2 | BP1 | BP0 |
|---|---|---|---|---|
| 0x00 | 0 | 0 | 0 | 0 |
| 0x01 | 0 | 0 | 0 | 1 |
| 0x02 | 0 | 0 | 1 | 0 |
| 0x03 | 0 | 0 | 1 | 1 |
| 0x04 | 0 | 1 | 0 | 0 |
| 0x05 | 0 | 1 | 0 | 1 |
| 0x06 | 0 | 1 | 1 | 0 |
| 0x07 | 0 | 1 | 1 | 1 |
| 0x08 | 1 | 0 | 0 | 0 |
| 0x09 | 1 | 0 | 0 | 1 |
| 0x0a | 1 | 0 | 1 | 0 |
| 0x0b | 1 | 0 | 1 | 1 |
| 0x0c | 1 | 1 | 0 | 0 |
| 0x0d | 1 | 1 | 0 | 1 |
| 0x0e | 1 | 1 | 1 | 0 |
| 0x0f | 1 | 1 | 1 | 1 |

- Refer to the data sheet of the serial NOR flash memory for the relationship between protect areas and protect bits.
- After processing finishes, read the status register to confirm that the value was written correctly.
- Make the top/bottom setting during configuration write processing.
- 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.
  < Using this user API to wait for the write to finish >
  Enable FLASH_WAIT_READY in r_qspi_flash_mx25l.h.
  < Not using this user API to wait for the write to finish >
  Disable FLASH_WAIT_READY in r_qspi_flash_mx25l.h, and call R_QSPI_FLASH_Wait() after processing by this user API has finished. This processing method allows you to perform write-end wait confirmation at any time you wish. Refer to figure 5.15 for instructions.

**Arguments**

```
uint8_t     DevNo     ; Device number
uint8_t     WpSts     ; Write protect setting data
```

**Return Value** The write protect setting result is returned.

```
FLASH_OK                ; Successful operation
FLASH_ERR_PARAM         ; Parameter error
FLASH_ERR_HARD          ; Hardware error
FLASH_ERR_TIMEOUT       ; Time out error (FLASH_WAIT_READY enabled)
FLASH_ERR_OTHER         ; Other error
```

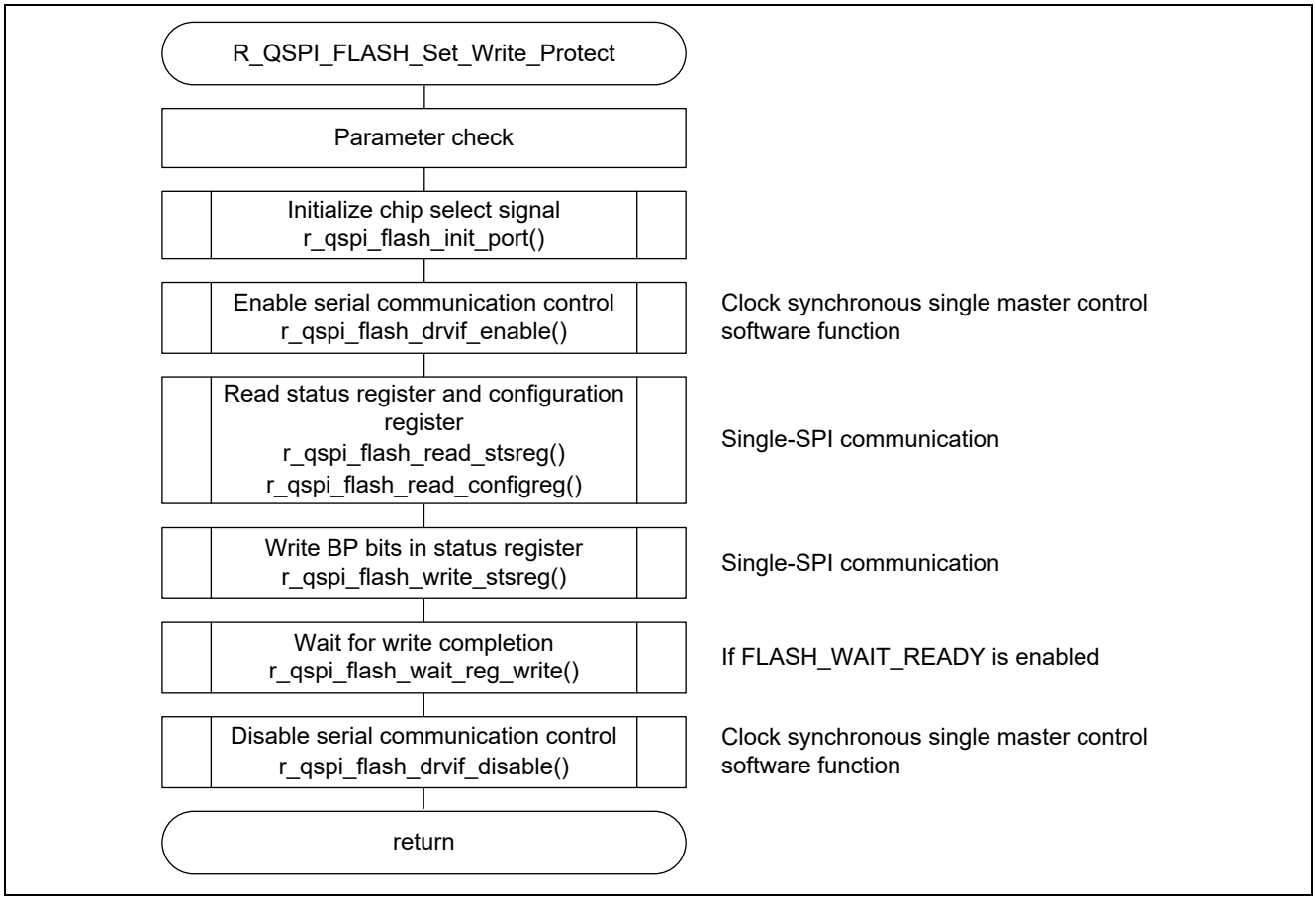



**Figure 5.14 Overview of Write Protect Setting Processing**

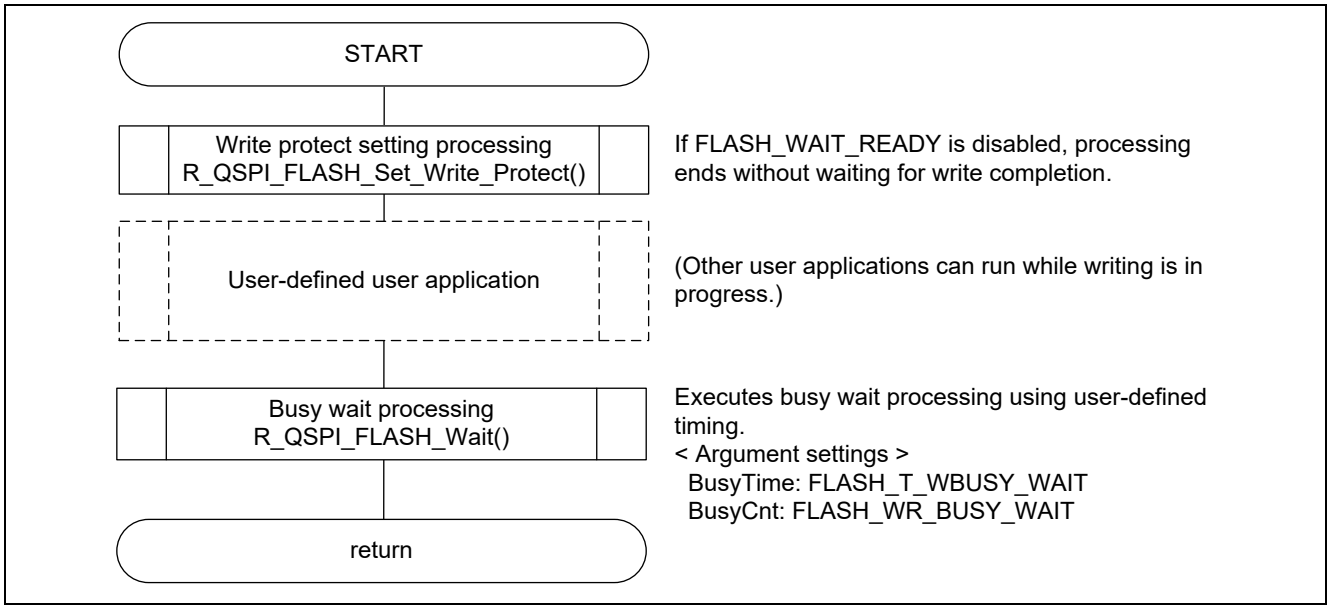



**Figure 5.15 Using R_QSPI_FLASH_Wait() to Wait for Write Protect Setting Completion**

### 5.9.7 Quad Mode Enable Setting Processing

| R_QSPI_FLASH_Quad_Enable | |
|---|---|
| **Outline** | Quad mode enable setting processing |
| **Header** | r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h |
| **Declaration** | error_t R_QSPI_FLASH_Quad_Enable(uint8_t DevNo) |
| **Description** | • Sets the quad enable (QE) bit in the status register to 1 to enable quad mode.<br>• When using quad mode, call this function beforehand.<br>• After processing finishes, read the status register to confirm the setting of the QE bit.<br>• The quad enable (QE) bit is non-volatile. To disable quad mode after it has been enabled once, run quad mode disable setting processing.<br>• There are two methods of waiting for the write to finish. These are shown below. Note that the next processing (write, read, erase, etc.) should be run after confirming that the write has finished.<br>< Using this user API to wait for the write to finish ><br>Enable FLASH_WAIT_READY in r_qspi_flash_mx25l.h.<br>< Not using this user API to wait for the write to finish ><br>Disable FLASH_WAIT_READY in r_qspi_flash_mx25l.h, and call R_QSPI_FLASH_Wait() after processing by this user API has finished. This processing method allows you to perform write-end wait confirmation at any time you wish. Refer to figure 5.17 for instructions. |
| **Arguments** | uint8_t DevNo ; Device number |
| **Return Value** | The setting result is returned.<br>FLASH_OK ; Successful operation<br>FLASH_ERR_PARAM ; Parameter error<br>FLASH_ERR_HARD ; Hardware error<br>FLASH_ERR_TIMEOUT ; Timeout error (FLASH_WAIT_READY enabled)<br>FLASH_ERR_OTHER ; Other error |

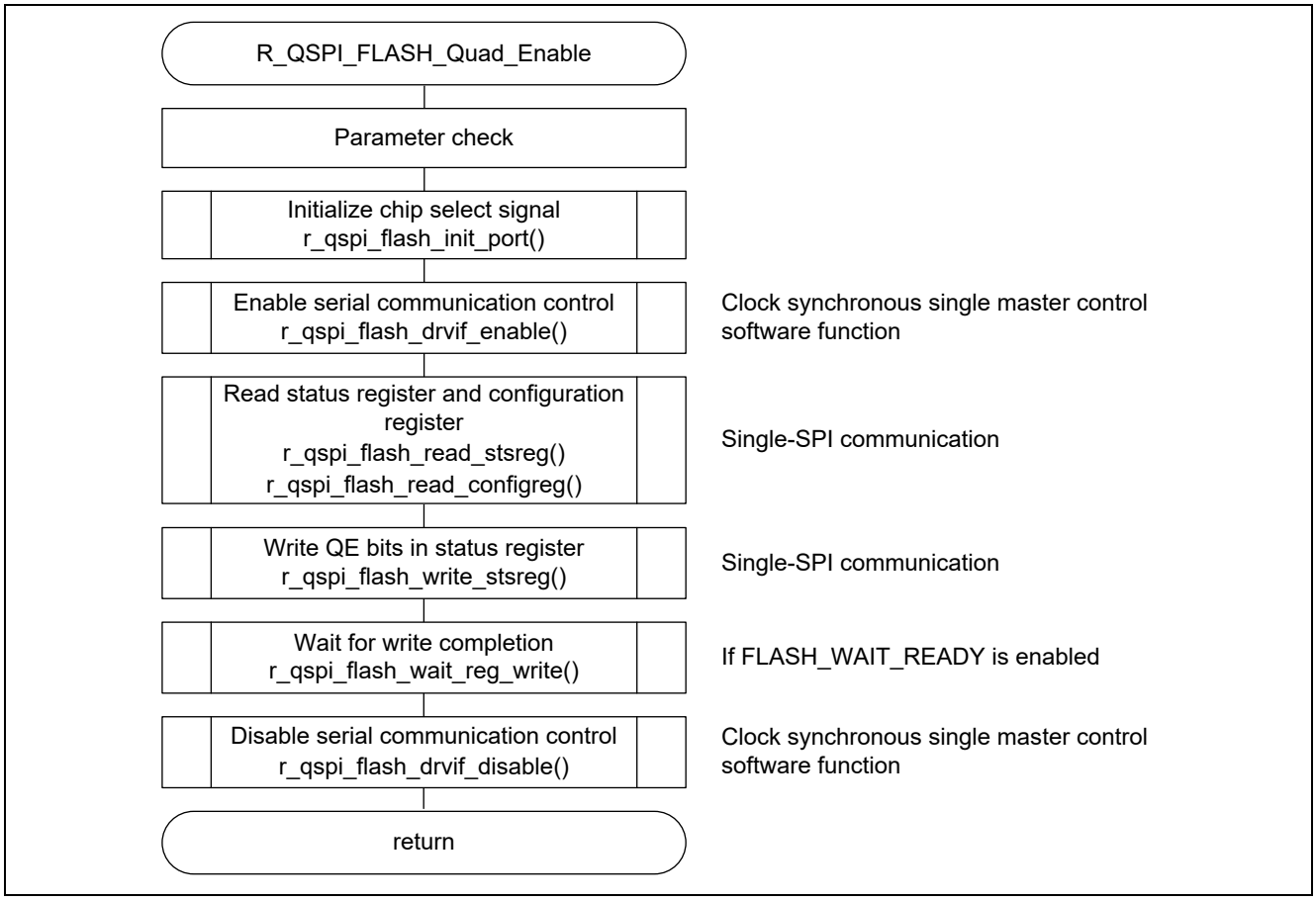



**Figure 5.16 Overview of Quad Mode Enable Setting Processing**

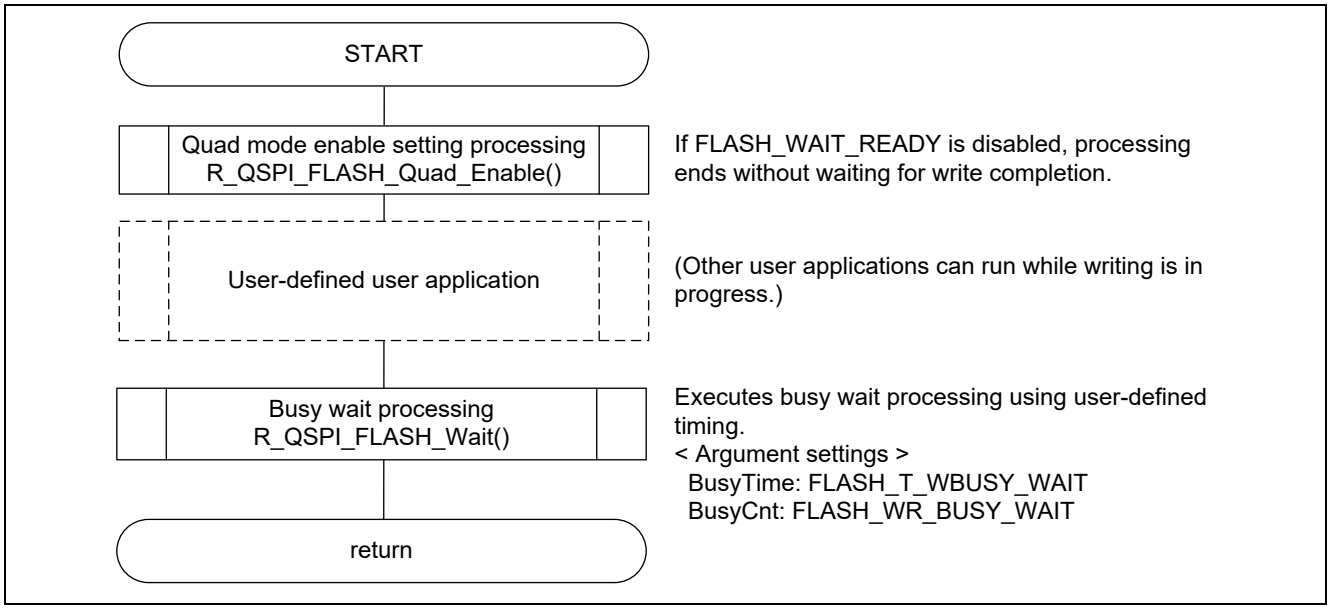



**Figure 5.17 Waiting for Quad Mode Enable Setting to Finish Using R_QSPI_FLASH_Wait()**

### 5.9.8 Quad Mode Disable Setting Processing

| R_QSPI_FLASH_Quad_Disable | |
|---|---|
| **Outline** | Quad mode disable setting processing |
| **Header** | r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h |
| **Declaration** | error_t R_QSPI_FLASH_Quad_Disable(uint8_t DevNo) |
| **Description** | • Clears the quad enable (QE) bit in the status register to 0 to enable quad mode.<br>• After processing finishes, read the status register to confirm the setting of the QE bit.<br>• The quad enable (QE) bit is non-volatile. To disable quad mode after it has been enabled once, run this function.<br>• There are two methods of waiting for the write to finish. These are shown below. Note that the next processing (write, read, erase, etc.) should be run after confirming that the write has finished.<br>< Using this user API to wait for the write to finish ><br>Enable FLASH_WAIT_READY in r_qspi_flash_mx25l.h.<br>< Not using this user API to wait for the write to finish ><br>Disable FLASH_WAIT_READY in r_qspi_flash_mx25l.h, and call R_QSPI_FLASH_Wait() after processing by this user API has finished. This processing method allows you to perform write-end wait confirmation at any time you wish. Refer to figure 5.19 for instructions. |
| **Arguments** | uint8_t DevNo ; Device number |
| **Return Value** | The setting result is returned.<br>FLASH_OK ; Successful operation<br>FLASH_ERR_PARAM ; Parameter error<br>FLASH_ERR_HARD ; Hardware error<br>FLASH_ERR_TIMEOUT ; Timeout error (FLASH_WAIT_READY enabled)<br>FLASH_ERR_OTHER ; Other error |

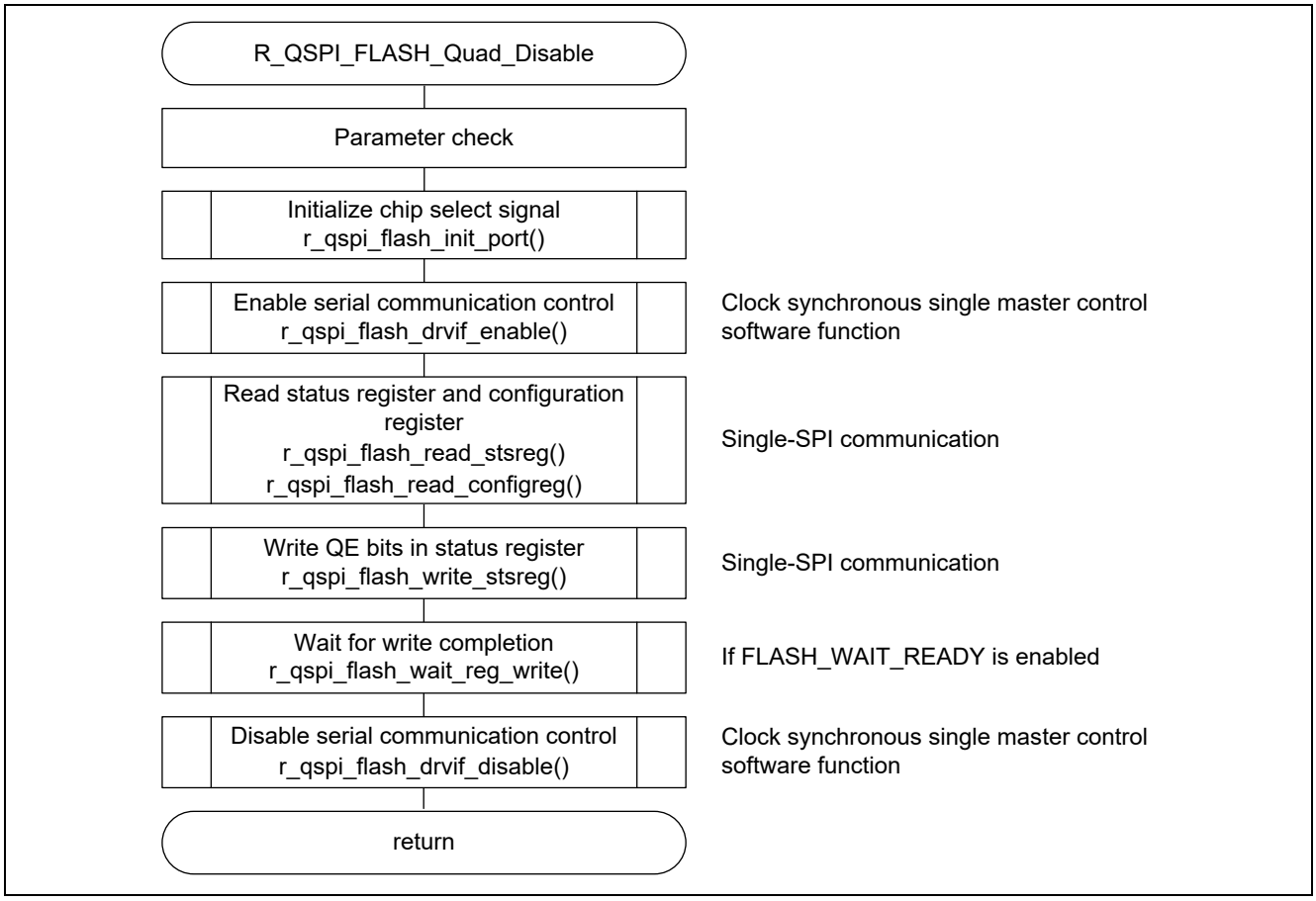



**Figure 5.18 Overview of Quad Mode Disable Setting Processing**

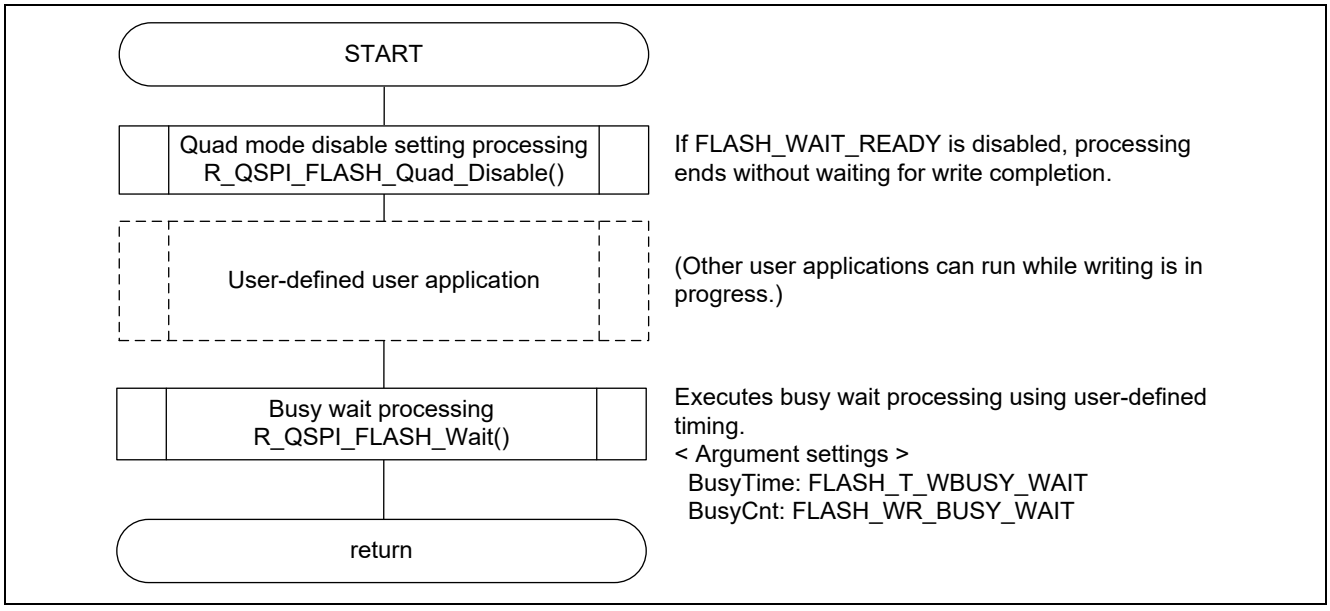



**Figure 5.19 Waiting for Quad Mode Disable Setting to Finish Using R_QSPI_FLASH_Wait()**

### 5.9.9 WRDI Command Issue Processing

| R_QSPI_FLASH_Write_Di | |
|---|---|
| **Outline** | WRDI command issue processing |
| **Header** | r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h |
| **Declaration** | error_t R_QSPI_FLASH_Write_Di(uint8_t DevNo) |
| **Description** | Clears the WEL bit in the status register. |
| **Arguments** | uint8_t DevNo ; Device number |
| **Return Value** | The clearing result is returned.<br>FLASH_OK ; Successful operation<br>FLASH_ERR_PARAM ; Parameter error<br>FLASH_ERR_HARD ; Hardware error<br>FLASH_ERR_OTHER ; Other error |

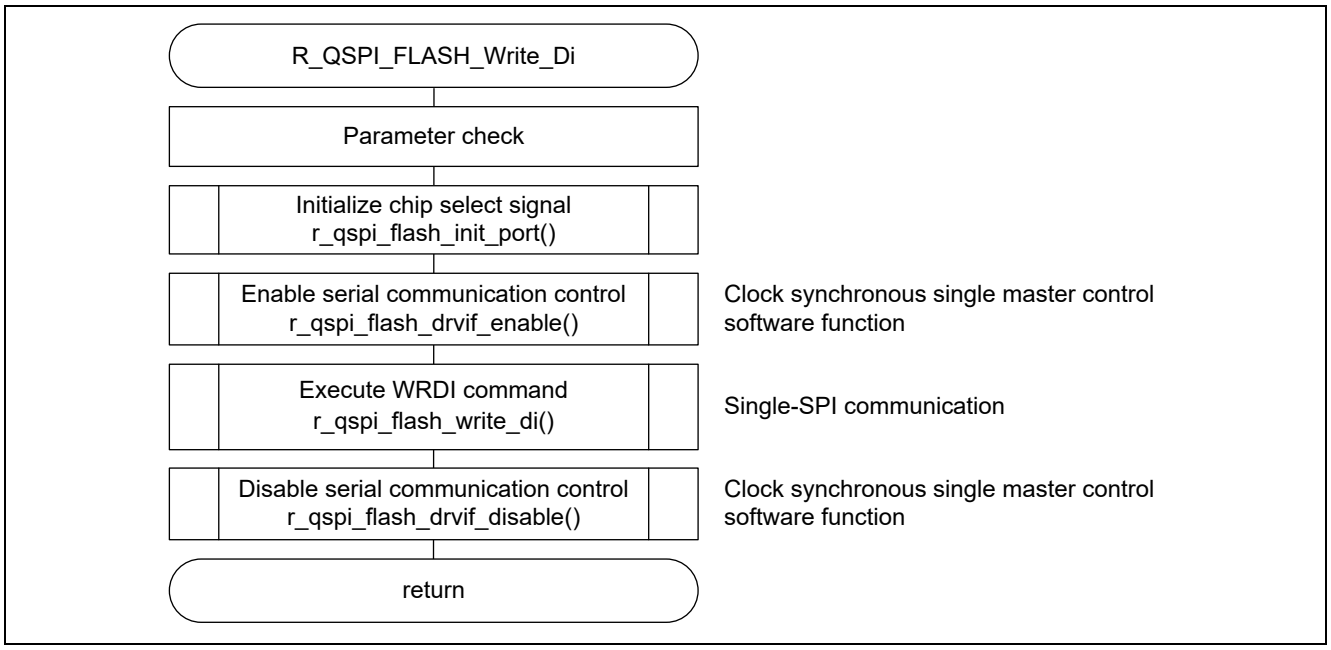



**Figure 5.20 Overview of WRDI Command Issue Processing**

### 5.9.10 Data Read Processing

| R_QSPI_FLASH_Read_Data | |
|---|---|
| **Outline** | Data read processing |
| **Header** | r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h |
| **Declaration** | error_t R_QSPI_FLASH_Read_Data(uint8_t DevNo, r_qspi_flash_info_t FAR* pFlash_Info) |
| **Description** | • Reads the specified number of bytes of data from the specified address in the serial NOR flash memory, and stores it in pData.<br>• The final read address is equal to the serial NOR flash memory capacity – 1.<br>• 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. |
| **Arguments** | uint8_t DevNo ; Device number<br>r_qspi_flash_info_t FAR* pFlash_Info ; FLASH communication information structure<br>uint32_t Addr ; Read start address<br>uint32_t Cnt ; Read byte count<br>uint16_t DataCnt ; Read byte temp. (setting prohibited)<br>uint8_t FAR* pData ; Read data storage buffer pointer |
| **Return Value** | The read result is returned.<br>FLASH_OK ; Successful operation<br>FLASH_ERR_PARAM ; Parameter error<br>FLASH_ERR_HARD ; Hardware error<br>FLASH_ERR_OTHER ; Other error |

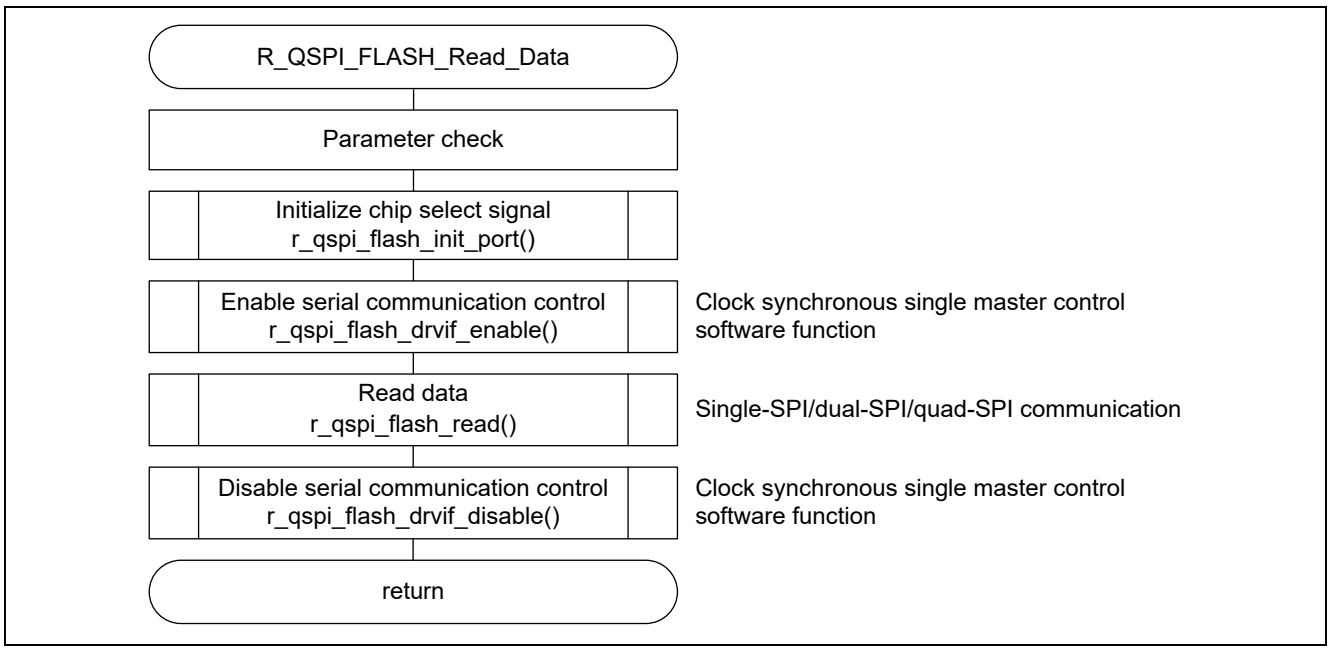



**Figure 5.21 Overview of Data Read Processing**

### 5.9.11 Data Write Processing

| R_QSPI_FLASH_Write_Data | |
|---|---|
| **Outline** | Data write processing |
| **Header** | r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h |
| **Declaration** | error_t R_QSPI_FLASH_Write_Data(uint8_t DevNo, r_qspi_flash_info_t FAR* pFlash_Info) |
| **Description** | • Writes the specified number of bytes of the data in pData to the specified address in the serial NOR flash memory.<br>• Writing to the serial NOR flash memory can only be performed to areas with write protect disabled. It is not possible to write to areas where protect is enabled. FLASH_ERR_WP is returned.<br>• The final write address is equal to the serial NOR flash memory capacity – 1.<br>• The maximum value that can be set for the write byte count (Cnt) is equal to the serial NOR flash memory capacity.<br>• The user API performs a wait for write completion regardless of the setting of FLASH_WAIT_READY in r_qspi_flash_mx25l.h. |
| **Arguments** | uint8_t DevNo ; Device number<br>r_qspi_flash_info_t FAR* pFlash_Info ; Flash communication information structure<br>uint32_t Addr ; Write start address<br>uint32_t Cnt ; Write byte count<br>uint16_t DataCnt ; Write byte temp. (setting prohibited)<br>uint8_t FAR* pData ; Write data storage buffer pointer |
| **Return Value** | The read result is returned.<br>FLASH_OK ; Successful operation<br>FLASH_ERR_PARAM ; Parameter error<br>FLASH_ERR_HARD ; Hardware error<br>FLASH_ERR_TIMEOUT ; Time out error<br>FLASH_ERR_OTHER ; Other error |

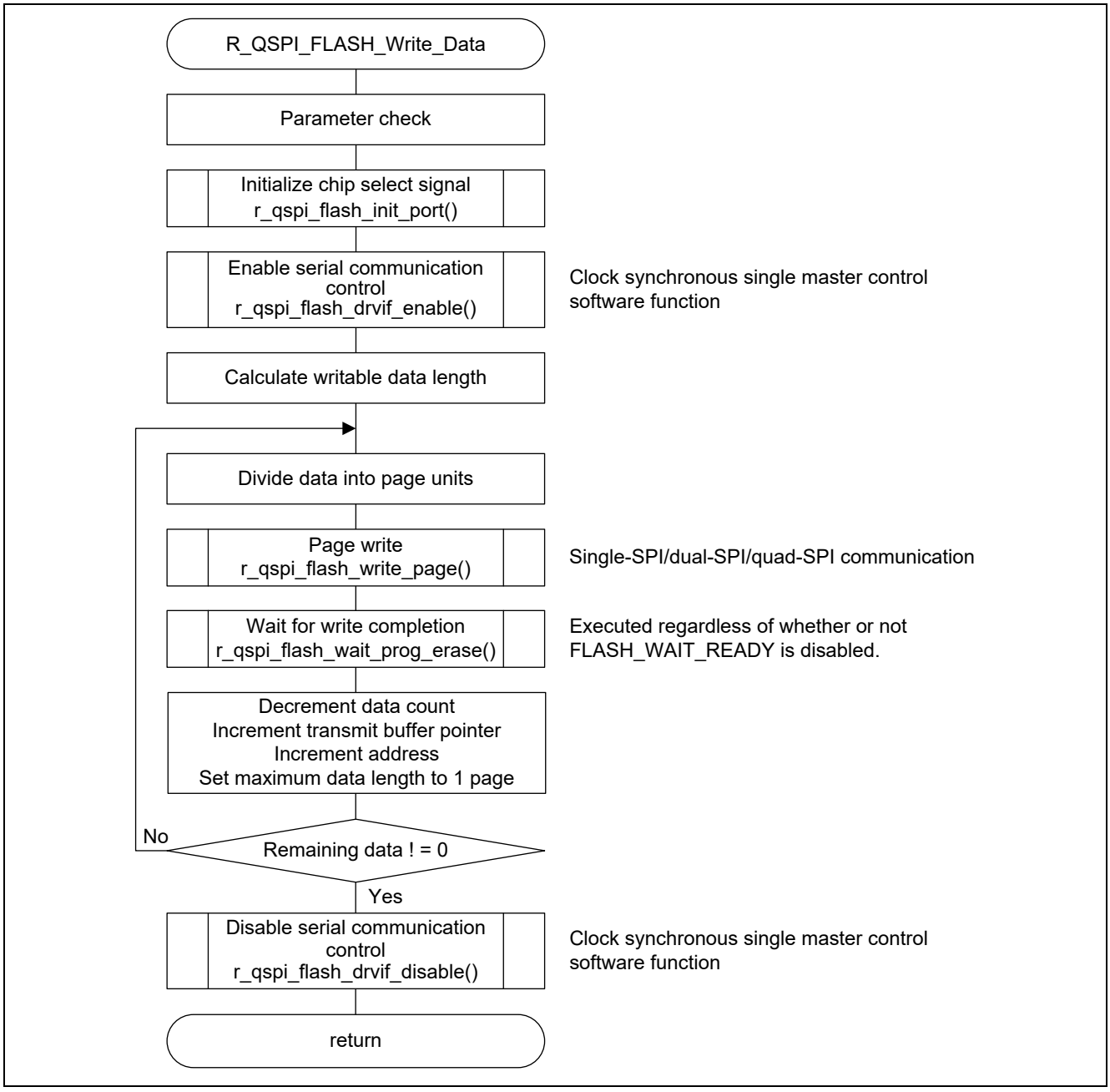



**Figure 5.22 Overview of Data Write Processing**

### 5.9.12 Data Write Processing (for single-page write)

| R_QSPI_FLASH_Write_Data_Page | |
|---|---|
| **Outline** | Data write processing (for single-page write) |
| **Header** | r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h |
| **Declaration** | error_t R_QSPI_FLASH_Write_Data_Page(uint8_t DevNo, r_qspi_flash_info_t FAR* pFlash_Info) |
| **Description** | • Writes the specified number of bytes (maximum: 1 page) of the data in pData to the specified address in the serial NOR flash memory.<br>• When writing a large volume of data, communication is divided into page units. This makes it possible to avoid situations in which other processing cannot be performed while communication is in progress.<br>• Writing to the serial NOR flash memory can only be performed to areas with write protect disabled. It is not possible to write to areas where protect is enabled. FLASH_ERR_WP is returned.<br>• The final write address is equal to the serial NOR flash memory capacity – 1.<br>• The maximum value that can be set for the write byte count (Cnt) is equal to the serial NOR flash memory capacity.<br>• Even if a byte count that exceeds one page is specified, the remaining byte count and the next address information remains in the FLASH communication information structure (pFlash_Info) after write processing of one page finishes. It is possible to write the remaining byte count by setting pFlash_Info once again without modification.<br>• 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.<br>• To use the user API to wait for write completion, enable FLASH_WAIT_READY in r_qspi_flash_mx25l.h.<br>• To wait for write completion without using the user API, disable FLASH_WAIT_READY in r_qspi_flash_mx25l.h and call R_QSPI_FLASH_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 5.24 for the usage method. |
| **Arguments** | uint8_t DevNo ; Device number<br>r_qspi_flash_info_t FAR* pFlash_Info ; FLASH communication information structure<br>uint32_t Addr ; Write start address<br>uint32_t Cnt ; Write byte count<br>uint16_t DataCnt ; Write byte temp. (setting prohibited)<br>uint8_t FAR* pData ; Write data storage buffer pointer |
| **Return Value** | The read result is returned.<br>FLASH_OK ; Successful operation<br>FLASH_ERR_PARAM ; Parameter error<br>FLASH_ERR_HARD ; Hardware error<br>FLASH_ERR_TIMEOUT ; Time out error (FLASH_WAIT_READY enabled)<br>FLASH_ERR_OTHER ; Other error |

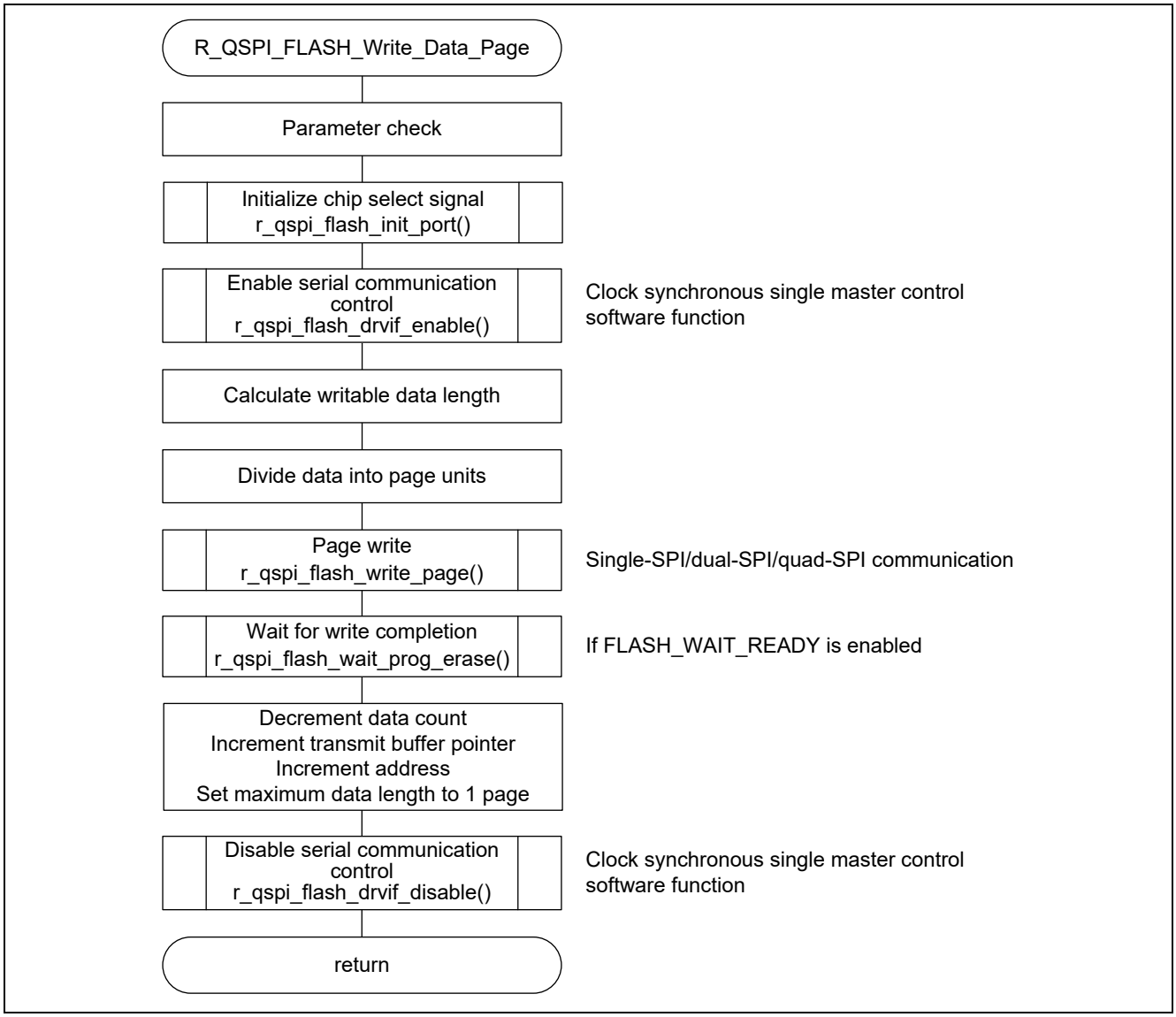



**Figure 5.23 Overview of Data Write Processing (for single-page write)**

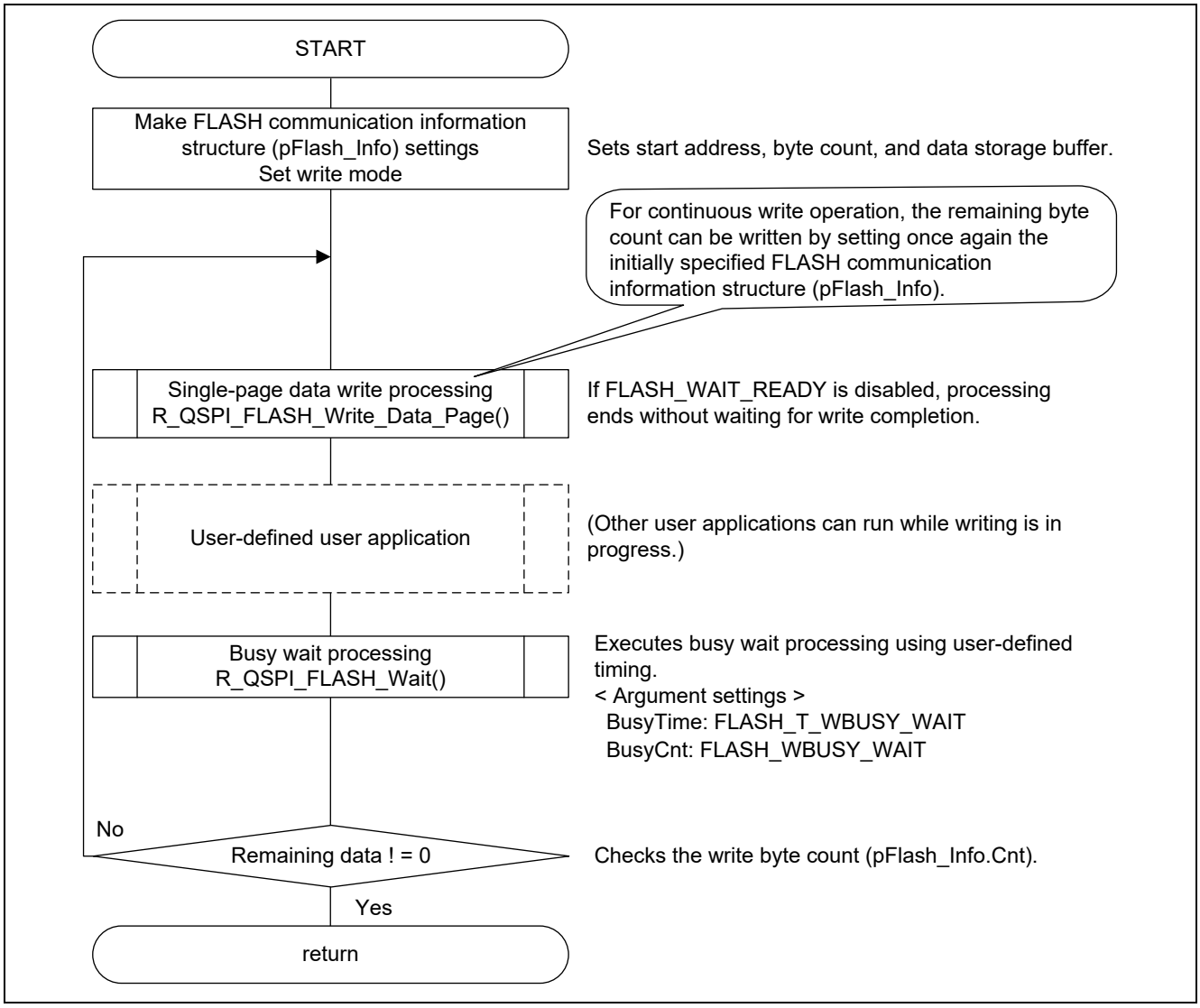



**Figure 5.24 Using R_QSPI_FLASH_Wait() to Wait for Data Write Processing (for Single-Page Write) Completion**

### 5.9.13 Erase Processing

R_QSPI_FLASH_Erase

**Outline** Erase processing

**Header** r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h

**Declaration** error_t R_QSPI_FLASH_Erase(uint8_t DevNo, uint32_t Addr, uint8_t Mode)

**Description**

- Erases all the data in a specified sector (sector erase), all the data in a specified block (block erase: 32 KB block or 64 KB block), or all the data in a specified chip (chip erase), according to the Mode setting.
- For sector erase, set Addr to the start address of the sector.
- For block erase, set Addr to the start address of the block.
- For chip erase, set Addr to 0x00000000.
- Erasing the serial NOR flash memory can only be performed on areas with write protect disabled. If write protect is enabled, erasing fails and FLASH_ERR_OTHER is returned.
- 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 FLASH_WAIT_READY in r_qspi_flash_mx25l.h.
- To wait for completion without using the user API, disable FLASH_WAIT_READY in r_qspi_flash_mx25l.h and call R_QSPI_FLASH_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 5.26 for the usage method.
- The setting of argument BusyCnt when calling R_QSPI_FLASH_Wait() differs depending on the Mode setting.

```
Sector erase (4 KB);      BusyCnt = FLASH_SE_BUSY_WAIT
Block erase (32 KB);      BusyCnt = FLASH_BE32K_BUSY_WAIT
Block erase (64 KB);      BusyCnt = FLASH_BE64K_BUSY_WAIT
Chip erase;               BusyCnt = FLASH_CE_BUSY_WAIT
```

**Arguments**

```
uint8_t     DevNo   ; Device number
uint32_t    Addr    ; Erase address
uint8_t     Mode    ; Erase mode (selectable from the following):
                          FLASH_MODE_S_ERASE
                          FLASH_MODE_B32K_ERASE
                          FLASH_MODE_B64K_ERASE
                          FLASH_MODE_C_ERASE
```

**Return Value** The erase result is returned.

```
FLASH_OK            ; Successful operation
FLASH_ERR_PARAM     ; Parameter error
FLASH_ERR_HARD      ; Hardware error
FLASH_ERR_TIMEOUT   ; Time out error (FLASH_WAIT_READY enabled)
FLASH_ERR_OTHER     ; Other error
```

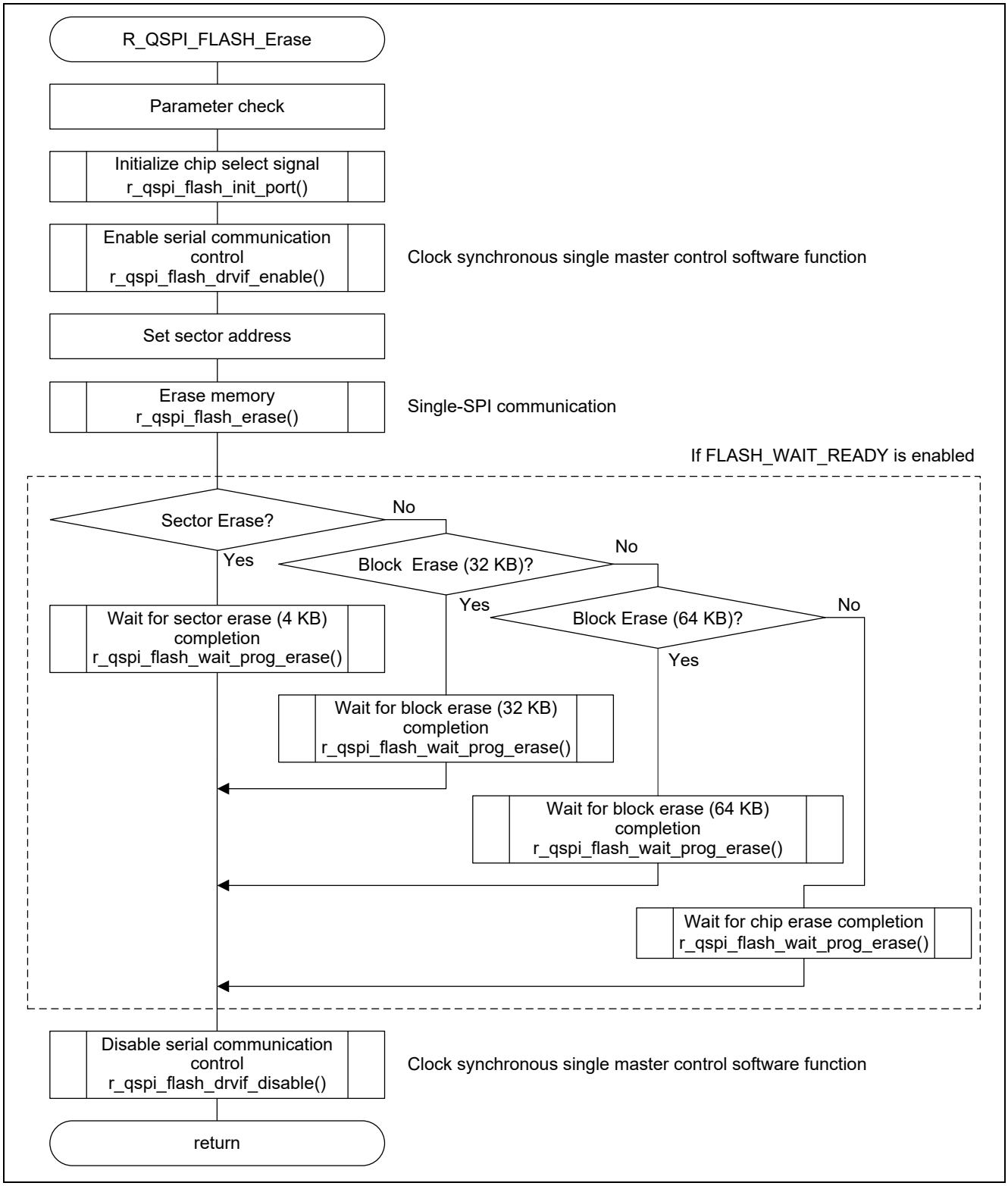



**Figure 5.25 Overview of Erase Processing**

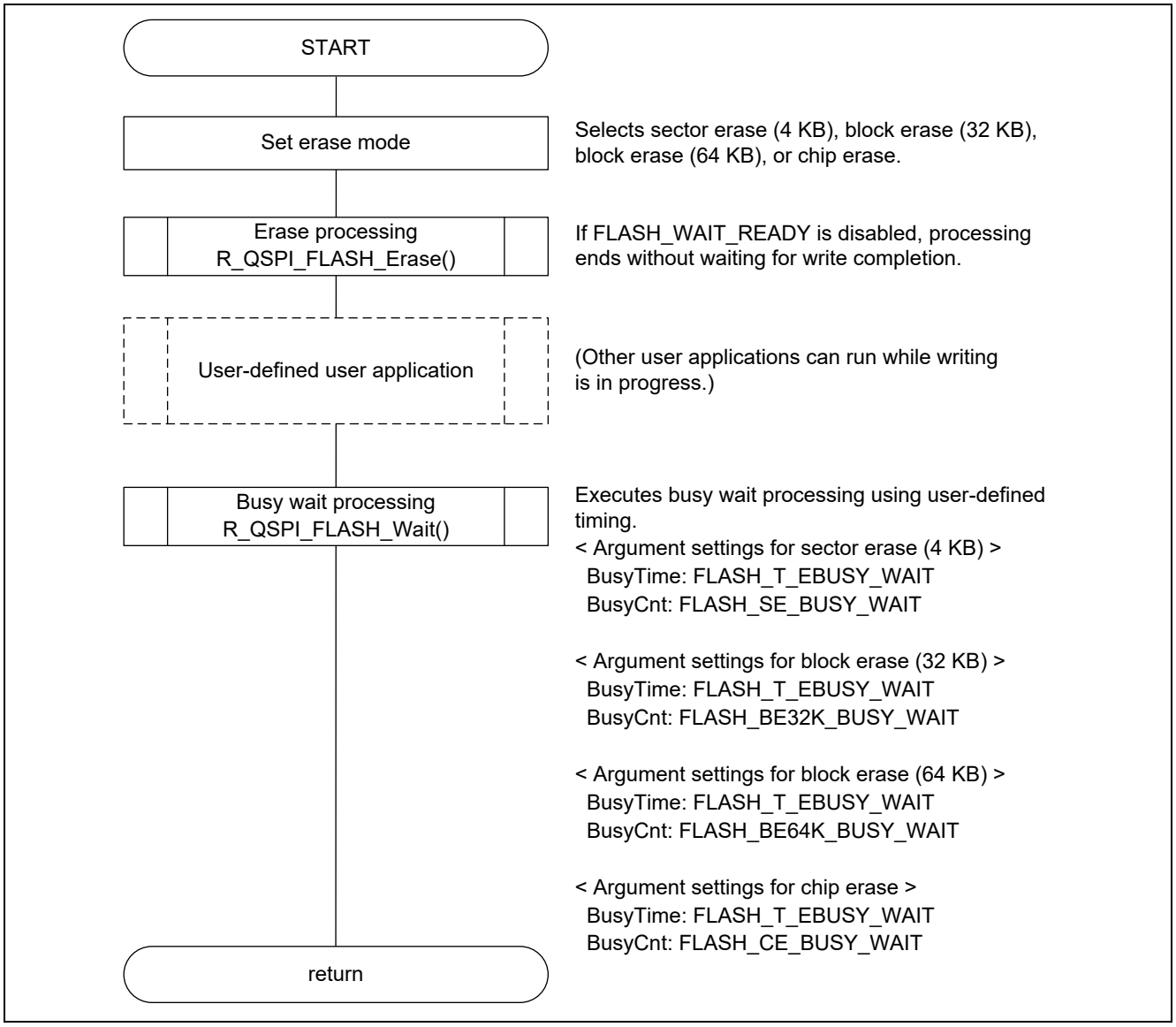



**Figure 5.26 Using R_QSPI_FLASH_Wait() to Wait for Erase Processing Completion**

### 5.9.14 ID Read Processing

| R_QSPI_FLASH_ReadID | | | |
|---|---|---|---|
| **Outline** | ID read processing | | |
| **Header** | r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h | | |
| **Declaration** | error_t R_QSPI_FLASH_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.<br>• Stores the following information in the read status storage buffer (pData):<br>(1) Manufacturer ID<br>(2) Device ID | | |
| **Arguments** | uint8_t | DevNo | ; Device number |
| | uint8_t FAR* | pData | ; Read data storage buffer pointer |
| **Return Value** | The read result is returned. | | |
| | FLASH_OK | | ; Successful operation |
| | FLASH_ERR_PARAM | | ; Parameter error |
| | FLASH_ERR_HARD | | ; Hardware error |
| | FLASH_ERR_OTHER | | ; Other error |

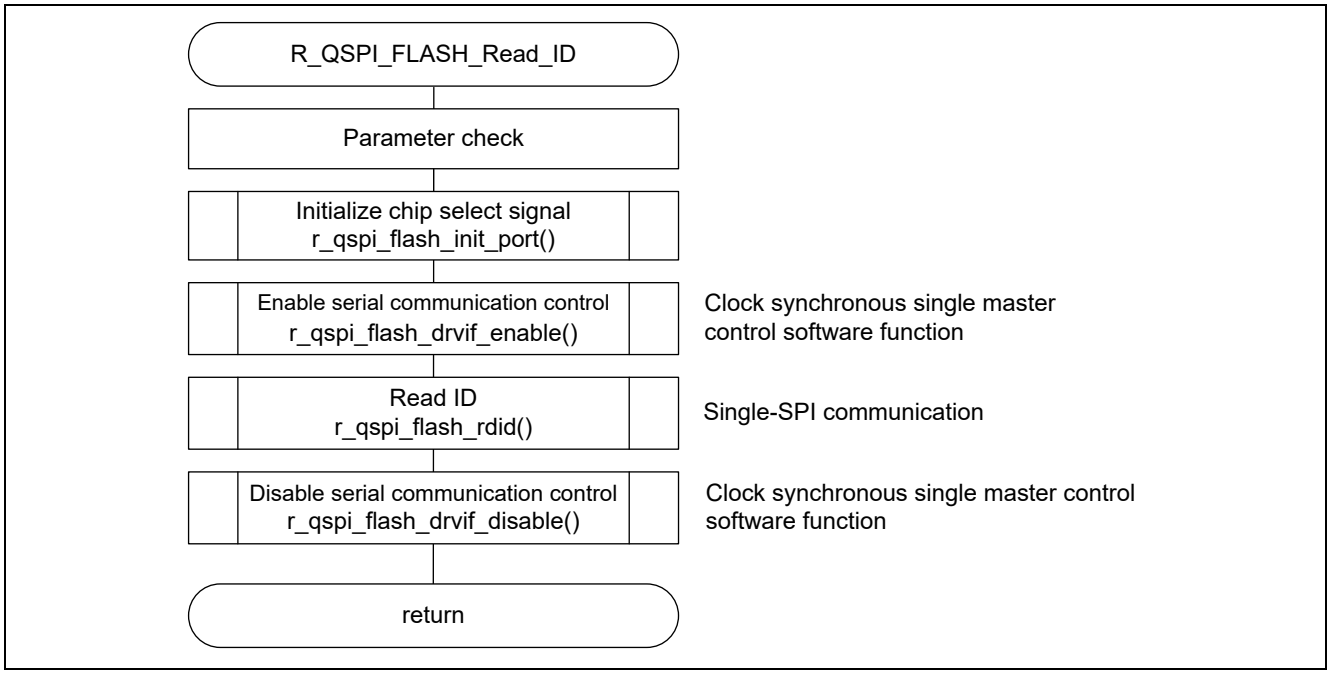



**Figure 5.27 Overview of ID Read Processing**

### 5.9.15 Busy Wait Processing

| R_QSPI_FLASH_Wait | |
|---|---|
| **Outline** | Busy wait processing |
| **Header** | r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h |
| **Declaration** | error_t R_QSPI_FLASH_Wait(uint8_t DevNo, uint16_t BusyTime, uint32_t BusyCnt, uint8_t Mode) |
| **Description** | • Use this function to confirm completion of write or erase when FLASH_WAIT_READY is disabled.<br>• When BusyCnt = 0, a wait is performed for a busy period equal to the BusyTime interval.<br>• 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.<br>• Performs wait for register write, data write, or erase, according to the Mode setting.<br>FLASH_MODE_REG_WRITE: Register write mode<br>FLASH_MODE_PROG_ERASE: Data write and erase mode<br>• In register write mode, the function issues the RDSR command and determines the ready/busy state by means of the WIP bit.<br>• In data write and erase mode, the function first determines the ready/busy state by means of the WIP bit. After confirming the ready state, the function issues the RDSCUR command and uses the E_FAIL or P_FAIL bit to check if an error occurred.<br>• 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 |
|---|---|---|
| Register write in progress | FLASH_T_WBUSY_WAIT | FLASH_WR_BUSY_WAIT |
| Data write in progress | FLASH_T_PBUSY_WAIT | FLASH_PP_BUSY_WAIT |
| Erase in progress (sector erase 4 KB) | FLASH_T_EBUSY_WAIT | FLASH_SE_BUSY_WAIT |
| Erase in progress (block erase 32 KB) | FLASH_T_EBUSY_WAIT | FLASH_BE32K_BUSY_WAIT |
| Erase in progress (block erase 64 KB) | FLASH_T_EBUSY_WAIT | FLASH_BE64K_BUSY_WAIT |
| Erase in progress (chip erase) | FLASH_T_EBUSY_WAIT | FLASH_CE_BUSY_WAIT |

| | | |
|---|---|---|
| **Arguments** | uint8_t DevNo | ; Device number |
| | uint16_t BusyTime | ; Wait duration (selectable from the following):<br>FLASH_T_WBUSY_WAIT: Register write<br>FLASH_T_PBUSY_WAIT: Data write<br>FLASH_T_EBUSY_WAIT: Erase |
| | uint32_t BusyCnt | ; Counter (selectable from the following):<br>FLASH_WR_BUSY_WAIT: Register write<br>FLASH_PP_BUSY_WAIT: Data write<br>FLASH_SE_BUSY_WAIT: Erase (Sector Erase 4 KB)<br>FLASH_BE32K_BUSY_WAIT: Erase (Block Erase 32 KB)<br>FLASH_BE64K_BUSY_WAIT: Erase (Block Erase 64 KB)<br>FLASH_CE_BUSY_WAIT: Erase (Chip Erase) |
| | uint8_t Mode | ; Wait mode (selectable from the following):<br>FLASH_MODE_REG_WRITE: Register write<br>FLASH_MODE_PROG_ERASE: Data write and erase |
| **Return Value** | The wait result is returned. | |
| | FLASH_OK | ; Successful operation |
| | FLASH_ERR_PARAM | ; Parameter error |
| | FLASH_ERR_HARD | ; Hardware error |
| | FLASH_ERR_TIMEOUT | ; Time out error (when BusyCnt ≠ 0) |
| | FLASH_ERR_OTHER | ; Other error |

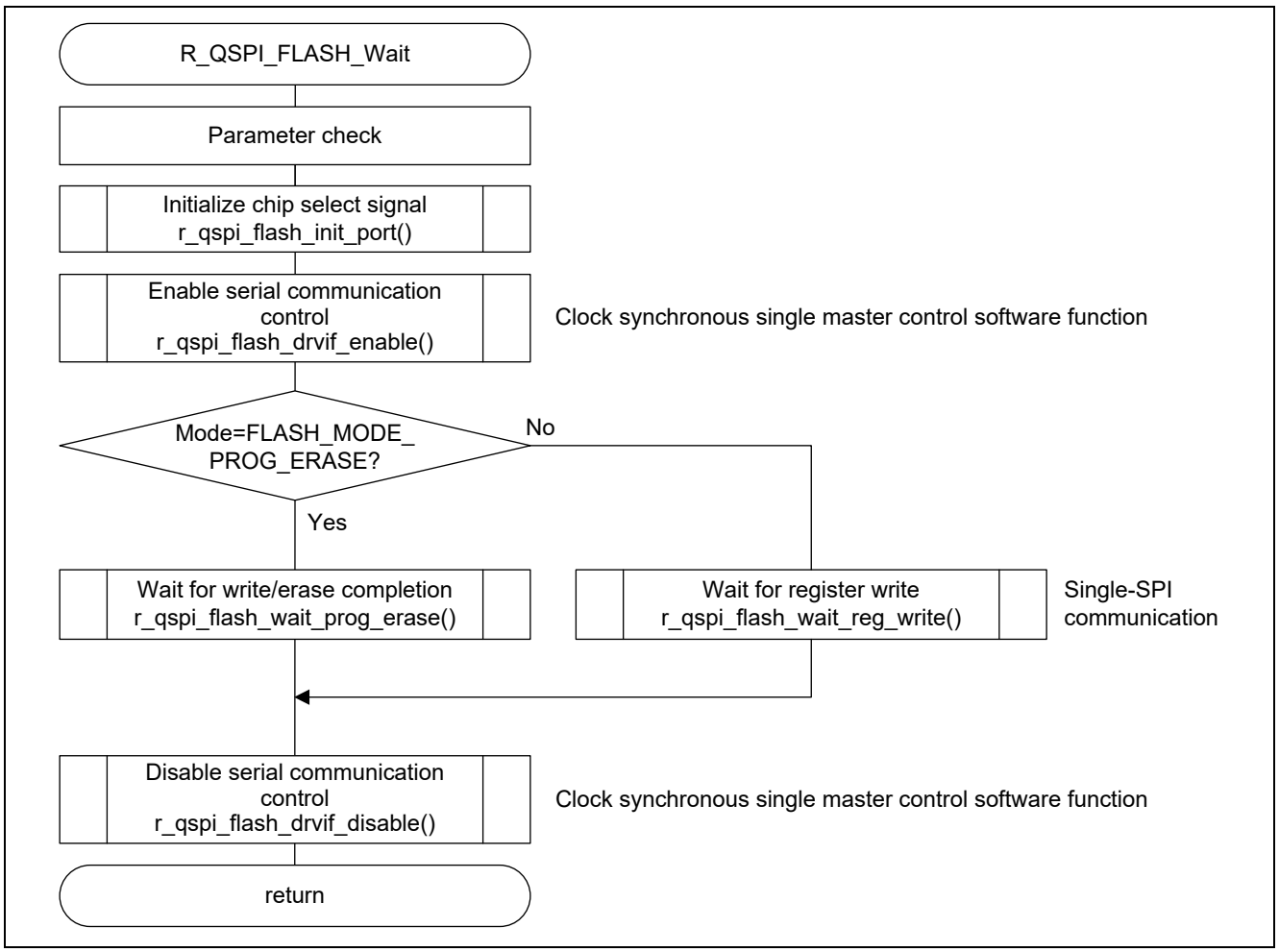



**Figure 5.28 Overview of Busy Wait Processing**

### 5.9.16 4-byte Address Mode Setting Processing

| R_QSPI_FLASH_Set_4byte_Address_Mode | |
|---|---|
| **Outline** | 4-byte address mode setting processing |
| **Header** | r_qspi_flash_mx25l.h, r_qspi_flash_mx25l_sub.h, r_qspi_flash_mx25l_sfr.h, r_qspi_flash_mx25l_drvif.h |
| **Declaration** | error_t R_QSPI_FLASH_Set_4byte_Address_Mode (uint8_t DevNo) |
| **Description** | • Calls the r_qspi_flash_enter_4addr() function to enable 4-byte address mode.<br>• After processing finishes, read the configuration register to confirm the setting of the 4BYTE bit.<br>• At system startup, call this function once after calling the R_QSPI_FLASH_Init_Driver() function. |
| **Arguments** | uint8_t DevNo ; Device number |
| **Return Value** | The address mode setting result is returned.<br>FLASH_OK ; Successful operation<br>FLASH_ERR_PARAM ; Parameter error<br>FLASH_ERR_HARD ; Hardware error<br>FLASH_ERR_OTHER ; Other error |

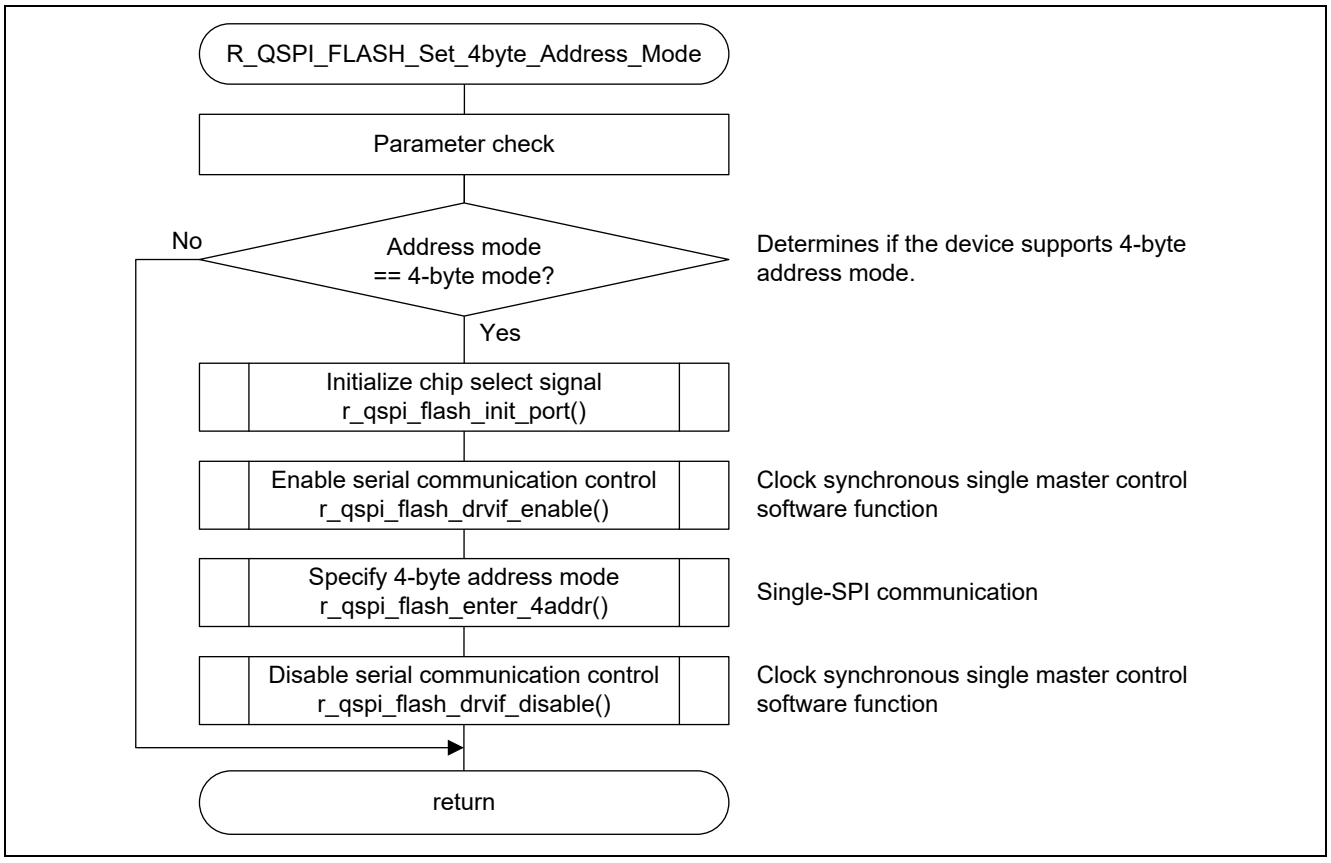



**Figure 5.29 Overview of Address Mode Setting Processing**

## 6. Application Example

Example settings for the serial NOR flash memory control portion are shown below. (The serial I/O control portion is not covered.)

Refer to the MCU-specific application note on the clock synchronous single master control software for details of the serial I/O control 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 NOR Flash Memory Control Software Settings

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

### 6.1.1 r_qspi_flash_mx25l.h

This is the definition file for the serial NOR flash 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 Flash memory. (1~N devices)               */
/*   Define the device number in accordance with the number of Flash memory */
/*   to be connected.                                                    */
/*---------------------------------------------------------------------- */
/* Define number of devices */
#define FLASH_DEV_NUM            (1)                 /* 1device          */

/* Define No. of slots */
#define FLASH_DEV0               (0)                 /* Device 0         */
#define FLASH_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 32 Mbit is used.

```
/*---------------------------------------------------------------------- */
/*   Define the serial Flash memory.                                     */
/*---------------------------------------------------------------------- */

//#define MX25R1635F                          /*  16Mbit (  2MByte)      */
#define MX25L3235E                            /*  32Mbit (  4MByte)      */
//#define MX25L3233F                          /*  32Mbit (  4MByte)      */
//#define MX25L6435E                          /*  64Mbit (  8MByte)      */
//#define MX25L12835F                         /* 128Mbit ( 16MByte)      */
//#define MX25L25635F                         /* 256Mbit ( 32MByte)      */
//#define MX66L51235F                         /* 512Mbit ( 64MByte)      */
//#define MX25L51245G                         /* 512Mbit ( 64MByte)      */
//#define MX66L1G45G                          /*   1Gbit (128MByte)      */
```

### (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 FLASH_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:

- Configuration register write processing (R_QSPI_FLASH_Write_Configuration())
- Write protect setting processing (R_QSPI_FLASH_Set_Write_Protect())
- Quad mode enable setting processing (R_QSPI_FLASH_Quad_Enable())
- Quad mode disable setting processing (R_QSPI_FLASH_Quad_Disable())
- Data write processing (for single-page write) (R_QSPI_FLASH_Write_Data_Page())
- Erase processing (R_QSPI_FLASH_Erase())

In the example below, waiting for completion is enabled.

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

### 6.1.2 r_qspi_flash_mx25l_sfr.h

A separate version of r_qspi_flash_mx25l_sfr.h.XXX is provided for each MCU model. Rename the version appropriate for the system to r_qspi_flash_mx25l_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_flash_mx25l_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 55 is used on the RX111.

```
/*------------------------------------------------------------------------- */
/*   Define the CS port.                                                    */
/*------------------------------------------------------------------------- */
#define FLASH_DR_CS0    PORT5.PODR.BIT.B5 /* FLASH CS0(Negative-true logic)*/
#define FLASH_DDR_CS0   PORT5.PDR.BIT.B5  /* FLASH CS0(Negative-true logic)*/

#if (FLASH_DEV_NUM > 1)
#define FLASH_DR_CS1                      /* FLASH CS1(Negative-true logic)*/
#define FLASH_DDR_CS1                     /* FLASH CS1(Negative-true logic)*/
#endif  /* #if (FLASH_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 FLASH_DR_CS0    P8.0        /* FLASH CS0    (Negative-true logic)*/
    #define FLASH_DDR_CS0   PM8.0       /* FLASH CS0    (Negative-true logic)*/

    #if (FLASH_DEV_NUM > 1)
    #define FLASH_DR_CS1                /* FLASH CS1    (Negative-true logic)*/
    #define FLASH_DDR_CS1               /* FLASH CS1    (Negative-true logic)*/
    #endif  /* #if (FLASH_DEV_NUM > 1) */
#endif  /* __CA78K0R__ */
#ifdef __CCRL__                         /* Renesas CC-RL Compiler           */
    #define FLASH_DR_CS0    P8_bit.no0  /* FLASH CS0    (Negative-true logic)*/
    #define FLASH_DDR_CS0   PM8_bit.no0 /* FLASH CS0    (Negative-true logic)*/

    #if (FLASH_DEV_NUM > 1)
    #define FLASH_DR_CS1                /* FLASH CS1    (Negative-true logic)*/
    #define FLASH_DDR_CS1               /* FLASH CS1    (Negative-true logic)*/
    #endif  /* #if (FLASH_DEV_NUM > 1) */
#endif  /* __CCRL__ */
#ifdef __ICCRL78__                      /* IAR RL78 Compiler                */
    #define FLASH_DR_CS0    P8_bit.no0  /* FLASH CS0    (Negative-true logic)*/
    #define FLASH_DDR_CS0   PM8_bit.no0 /* FLASH CS0    (Negative-true logic)*/

    #if (FLASH_DEV_NUM > 1)
    #define FLASH_DR_CS1                /* FLASH CS1    (Negative-true logic)*/
    #define FLASH_DDR_CS1               /* FLASH CS1    (Negative-true logic)*/
    #endif  /* #if (FLASH_DEV_NUM > 1) */
#endif  /* __ICCRL78__ */
```

### (2) Communication Clock Frequency Settings

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 Clock Frequency Settings**

| #define Definition | Application |
|---|---|
| FLASH_BR | Communication processing for other than the following two items (command transmission, etc.) |
| FLASH_BR_WRITE_DATA | Data write processing |
| FLASH_BR_READ_DATA | Data read processing |

In the example below, the RSPI of the RX111 is used.

```
    /* PCLK = 32MHz, n=0 for RX111 RSPI */
#define FLASH_BR              (uint8_t)(0x00)   /* SPBR initial setting     */
                    /*       ++---------------- 16.00MHz                   */

    /* PCLK = 32MHz, n=0 for RX111 RSPI Write Data */
#define FLASH_BR_WRITE_DATA   (uint8_t)(0x00)   /* SPBR initial setting     */
                    /*       ++---------------- 16.00MHz                   */

    /* PCLK = 32MHz, n=0 for RX111 RSPI Read Data */
#define FLASH_BR_READ_DATA    (uint8_t)(0x00)   /* SPBR initial setting     */
                    /*       ++---------------- 16.00MHz                   */
```

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

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

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

    /* fMCK = 24MHz for RL78 CSI Read Data */
#define FLASH_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_flash_mx25l_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 a specified sector (sector erase), all the data in a specified block (block erase: 32 KB block or 64 KB block), or all the data in a specified chip (chip erase).

The settings below should be reevaluated if the maximum duration is different, depending on the device.

```
/*------- Definitions of software timer value --------*/
/* Write Status/Configuration Register
                            :   30ms - MX25R1635F                                */
/*                          :   40ms - MX25L3233F,  MX25L3235E,  MX25L6435E,     */
/*                          :   40ms - MX25L12835F, MX25L25635F, MX66L51235F     */
/*                          :   40ms - MX25L51245G,              MX66L1G45G      */
/* Sector Erase (4KB)       :   30ms - MX25R1635F                                */
/*                          :  200ms - MX25L3235E,  MX25L3233F,                  */
/*                          :  300ms - MX25L6435E,                               */
/*                          :  120ms - MX25L12835F, MX25L25635F, MX66L51235F     */
/*                          :  400ms - MX25L51245G, MX66L1G45G                   */
/* Block Erase (32KB)       :     3s - MX25R1635F                                */
/*                          :   1.6s - MX25L3235E                                */
/*                          :   0.6s - MX25L3233F                                */
/*                          :     2s - MX25L6435E                                */
/*                          :  0.65s - MX25L12835F, MX25L25635F, MX66L51235F     */
/*                          :     1s - MX25L51245G, MX66L1G45G                   */
/* Block Erase (64KB)       :   3.5s - MX25R1635F                                */
/*                          :     2s - MX25L3235E,  MX25L6435E                   */
/*                          :     1s - MX25L3233F                                */
/*                          :  0.65s - MX25L12835F, MX25L25635F, MX66L51235F     */
/*                          :     2s - MX25L51245G, MX66L1G45G                   */
/* Chip Erase               :    60s - MX25R1635F                                */
/*                          :    50s - MX25L3235E                                */
/*                          :    30s - MX25L3233F                                */
/*                          :    80s - MX25L6435E,  MX25L12835F                  */
/*                          :   150s - MX25L25635F                               */
/*                          :   300s - MX66L51235F                               */
/*                          :   200s - MX25L51245G                               */
/*                          :   600s - MX66L1G45G                                */
/* Page (256 bytes) Program :   10ms - MX25R1635F                                */
/*                          :    3ms - MX25L3235E                                */
/*                          :  1.2ms - MX25L3233F                                */
/*                          :    5ms - MX25L6435E                                */
/*                          :  1.5ms - MX25L12835F, MX25L25635F, MX66L51235F     */
/*                          : 0.75ms - MX25L51245G                               */
/*                          :    3ms - MX66L1G45G                                */

#define FLASH_SE_BUSY_WAIT    (uint32_t)(200)
                      /* Sector Erase busy timeout            200*1ms =  0.2s */
#define FLASH_BE32K_BUSY_WAIT (uint32_t)(1000)
                      /* Block Erase (32KB) busy timeout    1,000*1ms =    1s */
#define FLASH_BE64K_BUSY_WAIT (uint32_t)(2000)
                      /* Block Erase (64KB) busy timeout    2,000*1ms =    2s */
#define FLASH_CE_BUSY_WAIT    (uint32_t)(1200000)
                      /* Chip Erase busy timeout        1,200,000*1ms = 1200s */
#define FLASH_PP_BUSY_WAIT    (uint32_t)(3000)
                      /* Page Program timeout               3,000*1us =   3ms */
#define FLASH_WR_BUSY_WAIT    (uint32_t)(40000)
                      /* Write Register timeout            40,000*1us =  40ms */
```

### (2) Write Timeout Duration Setting

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

```
#define FLASH_PP_BUSY_WAIT     (uint32_t)(3000)
                          /* Page Program timeout            3000*1us =   3ms */
#define FLASH_WR_BUSY_WAIT     (uint32_t)(40000)
                          /* Write Register timeout         40000*1us =  40ms */
```

### 6.1.4 r_qspi_flash_mx25l_sub.c

This is the source file for internal functions of the serial NOR flash memory.

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

#### (1) Macro Function R_QSPI_FLASH_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_FLASH_CMD_READ() Definition**

| No. | #define Definition | Instruction Code on Data Sheet | Processing Details |
|---|---|---|---|
| 1 | r_qspi_flash_send_cmd(**FLASH_CMD_FREAD**,(uint32_t )Addr,FLASH_CMD_SIZE+FLASH_ADDR_SIZE+1) | FAST READ | Single-SPI read (high-speed) |
| 2 | r_qspi_flash_send_cmd(**FLASH_CMD_DREAD**,(uint32_t) Addr,FLASH_CMD_SIZE+FLASH_ADDR_SIZE+1) | DUAL OUTPUT READ | Dual-SPI read (high-speed) |
| 4 | r_qspi_flash_send_cmd(**FLASH_CMD_QREAD**,(uint32_t)Addr,FLASH_CMD_SIZE+FLASH_ADDR_SIZE+1) | QUAD OUTPUT READ | Quad-SPI read (high-speed) |

#### (2) Macro Function R_QSPI_FLASH_CMD_PP() Setting

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

**Table 6.3 Macro Function R_QSPI_FLASH_CMD_PP() Definition**

| No. | #define Definition | Instruction Code on Data Sheet | Processing Details |
|---|---|---|---|
| 1 | r_qspi_flash_send_cmd(**FLASH_CMD_PP**,(uint32_t) Addr,FLASH_CMD_SIZE+FLASH_ADDR_SIZE) | PAGE PROGRAM | Single-SPI write |
| 2 | r_qspi_flash_send_cmd(**FLASH_CMD_4PP**,(uint32_t) Addr,FLASH_CMD_SIZE+FLASH_ADDR_SIZE) | 4 x I/O PAGE PROGRAM | Quad-SPI write |

### 6.1.5 r_qspi_flash_mx25l_drvif.c

This is the source file for the clock synchronous single control software interface of the serial NOR flash memory.

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

(1) **r_qspi_flash_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_flash_drvif_init_driver(void)
{
    return R_SIO_Init_Driver();
}
```

(2) **r_qspi_flash_drvif_disable() Setting**

This specifies the serial I/O 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_flash_drvif_disable(void)
{
    return R_SIO_Disable();
}
```

(3) **r_qspi_flash_drvif_enable() Setting**

This specifies the serial IO enable setting processing used by the clock-synchronous single master control software.

The value of the BrgData argument is set in the bit rate register.

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

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

#### (4) r_qspi_flash_drvif_enable_tx_data() Setting

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

The value of the BrgData argument is set in the bit rate register.

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

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

#### (5) r_qspi_flash_drvif_enable_rx_data() Setting

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

The value of the BrgData argument is set in the bit rate register.

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

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

#### (6) r_qspi_flash_drvif_open_port() Setting

This specifies the serial IO 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_flash_drvif_open_port(void)
{
    return R_SIO_Open_Port();
}
```

#### (7) r_qspi_flash_drvif_tx() Setting

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

The TxCnt argument specifies the transmit data size (bytes), and the pData argument specifies the transmit data storage destination buffer address.

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

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

#### (8) r_qspi_flash_drvif_tx_add() Setting

This specifies the serial IO data transmit processing used by the clock-synchronous single master control software.

This was added for cases in which a separate setting is required, in addition to r_qspi_flash_drvif_tx().

For the serial NOR flash memory in the present example, it is used for address transmission.

The TxCnt argument specifies the transmit data size (bytes), and the pData argument specifies the transmit data storage destination buffer address.

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

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

#### (9) r_qspi_flash_drvif_tx_data() Setting

This specifies the serial IO data transmit processing exclusively for data writes used by the clock-synchronous single master control software. It is mainly used for writing data.

The TxCnt argument specifies the transmit data size (bytes), and the pData argument specifies the transmit data storage destination buffer address.

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

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

#### (10) r_qspi_flash_drvif_rx() Setting

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

The RxCnt argument specifies the receive data size (bytes), and the pData argument specifies the receive data storage destination buffer address.

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

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

#### (11) r_qspi_flash_drvif_rx_add() Setting

This specifies the serial IO data receive processing used by the clock-synchronous single master control software.

This was added for cases in which a separate setting is required, in addition to r_qspi_flash_drvif_rx().

Not used for the serial NOR flash memory in the present example.

The RxCnt argument specifies the receive data size (bytes), and the pData argument specifies the receive data storage destination buffer address.

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

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

#### (12) r_qspi_flash_drvif_rx_data() Settings

This specifies the serial IO data transmit processing exclusively for data reads used by the clock-synchronous single master control software.

The RxCnt argument specifies the receive data size (bytes), and the pData argument specifies the receive data storage destination buffer address.

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

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

### 6.1.6 r_qspi_flash_mx25l_sfr_rl78.c

This program is the SFR module for the RL78.

A separate version of r_qspi_flash_mx25l_sfr_rl78XXX.c is provided for each MCU model. Build the version appropriate for the system 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_flash_mx25l_sfr_rl78.c.

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

#### (1) Settings for Defining the SFR Area

When using the RL78 Family or 78K0R, the C compiler used contains predefined preprocessor symbols. The program code already contains these predefined preprocessor symbols.

When using an RL78 Family or 78K0R microcontroller with the integrated development environment from IAR Systems, it is necessary to specify a header file containing SFR definitions for the microcontroller.

Also refer to the clock synchronous single master control software for the specific microcontroller.

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

**Table 6.4 Microcontrollers and Settings for Defining SFR Area**

| Integrated Development Environment | MCU | SFR Settings Required/ Not Required | Setting Method |
|---|---|---|---|
| CubeSuite+<br>CS+ | RL78 | Not required | Not required |
| | 78K0R | Not required | Not required |
| | RX | Not required | Not required |
| IAR Embedded Workbench | RL78 | Required | #ifdef __ICCRL78__<br>#include <ior5f104pj.h> ← Change to match the microcontroller.<br>#include <ior5f104pj_ext.h> ← Change to match the microcontroller.<br>#endif |
| | 78K0R | Required | #ifdef __ICC78K__<br>#include <io78f1009_64.h> ← Change to match the microcontroller.<br>#include <io78f1009_64_ext.h> ← Change to match the microcontroller.<br>#endif |
| | RX | (Not supported by this software.) | (Not supported by this software.) |
| e$^2$ studio | RL78 | Not required | Not required |
| | 78K0R | (Not supported by this software.) | (Not supported by this software.) |
| | RX | (Not supported by this software.) | (Not supported by this software.) |

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

```
#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_flash_mx25l.h
r_qspi_flash_mx25l_sub.h
r_qspi_flash_mx25l_sfr.h
r_qspi_flash_mx25l_drvif.h

When using Smart Configurator with e$^2$ studio (CC-RL compiler), include iodefine.h.

For RL78 / G23, the path is as follows.

"/src/smc_gen/r_bsp/mcu/rl78_g23/register_access/ccrl"

## 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:

FLASH_MEM_SIZE
FLASH_SECT_ADDR
FLASH_B32K_ADDR
FLASH_B64K_ADDR
FLASH_PAGE_SIZE
FLASH_ADDR_SIZE
FLASH_WP_WHOLE_MEM
FLASH_FULL_CHIP_ERASE
FLASH_ADDR_MODE

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.

## Website and Support

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

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

## Revision History

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Feb. 29, 2015 | — | First edition issued |
| 1.02 | Mar. 31, 2016 | — | Support CC-RL compiler. |
| | | 1 | Modified Introduction to add short address. |
| | | 7 to 13 | Added the following conditions to section 2.2.<br>(2) RL78/G14 SAU CS+ for CC Integrated Development Environment (Compiler:CC-RL)<br>(5) RL78/G1C SAU CS+ for CC Integrated Development Environment (Compiler:CC-RL)<br>(8) RL78/L12 SAU CS+ for CC Integrated Development Environment (Compiler:CC-RL)<br>(11) RL78/L13 SAU CS+ for CC Integrated Development Environment (Compiler:CC-RL)<br>(14) RL78/L1C SAU CS+ for CC Integrated Development Environment (Compiler:CC-RL) |
| | | 26 to 28 | Added the following to section 5.3.2, Required Memory Size.<br>(2) RL78/G14 SAU CS+ for CC Integrated Development Environment (Compiler:CC-RL)<br>(5) RL78/L13 SAU CS+ for CC Integrated Development Environment (Compiler:CC-RL) |
| | | 29 | Updated the software version in Table 5.8 File Structure. |
| | | 69 | Updated example in 6.1.2 (1) Chip Select Signal Setting. |
| | | 78 | Updated Table 6.4 Microcontrollers and Settings for Defining SFR Area. |
| 1.03 | Jul. 14, 2021 | 1 | Updated document links<br>MX25R1635F and MX25L3233F added to operation check devices<br>Added RL78/G23 to the MCU used for operation check |
| | | 16 | Added the operation confirmation condition of RL78/G23 to 2. Operation Confirmation Condition. |
| | | 17 | Changed the title on 3.2 RL78 Family, 78K0R Family: List of Related Application Notes to the latest title |
| | | 31 | Added memory size of RL78/G23 to 5.3.2 RL78/G23 to RL78 Family, 78K0R / Kx3-L |
| | | 32 | Added RL78/G23 files to 5.4 File Structure |
| | | 68 | Added definitions of MX25R1635F and MX25L3233F to 6.1.1 r_qspi_flash_mx25l.h, (2) Definition of Capacity of Device Used |
| | | 73 | Added definitions of MX25R1635F and MX25L3233F to 6.1.3 r_qspi_flash_mx25l_sub.h, (1) Erase Timeout Duration Settings |
| 1.04 | Aug. 4, 2022 | 15 | Added the following conditions to section 2.2.<br>(18) RL78/G23 SAU e$^2$ studio Integrated Development Environment (Compiler:LLVM) |
| | | 31 | Added the following to section 5.3.2, Required Memory Size.<br>(9) RL78/G23 SAU e$^2$ studio Integrated Development Environment (Compiler:LLVM) |

| | | |
|---|---|---|
| | | 32 | Section 5.4<br>Rename the sample code folder.<br>Updated Application Note Number. |
| | | 82 | Added the following to Section 6.1.6 (1) Settings for Defining the SFR Area.<br>e$^2$ studio |
| | | 84 | Section 7.1<br>Added notes on using smart configurator with e$^2$ studio (CC-RL compiler). |

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

# Notice

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

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

## Corporate Headquarters

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

## Trademarks

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

## Contact information

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