RX Family

Clock Synchronous Single Master Control Software Using the RSPI

R01AN1196EJ0111
Rev.1.11
Mar. 31, 2016

Abstract

This application note describes a clock synchronous single master control method that uses RX Series Renesas serial peripheral interface (RSPI) clock synchronous (three-wire method) serial communication and sample code that uses that method.

SPI mode single master control can be implemented by adding SPI slave device selection control using port control.

This sample code implements the single master basic control method that is unique to these microcontrollers. The user should implement the software required to control the slave devices using this sample code.

Software in the upper-level layer for controlling the slave device is separately available, so please obtain this from the following URL as well. When the slave device control software is added, update of this application note may not be in time. Refer to the following URL for the combination information on the latest slave device control software.

- SPI Serial EEPROM Control Software

  http://www.renesas.com/driver/spi_serial_eeprom

- SPI/QSPI Serial Flash Memory Control Software, QSPI Serial Phase Change Memory Control Software

  http://www.renesas.com/driver/spi_serial_flash

Target Devices

Target microcontroller: RX210 Group, RX21A Group, RX220 Group,
RX63N Group, RX63T Group, RX634 Group, RX64M Group,
RX111 Group
RX71M Group

Devices used in verifying operation

- Renesas Electronics Corporation R1EX25xxx Series SPI Serial EEPROM
- Micron Technology M25P Series Serial Flash Memory Control Software 64 Mbits
- Micron Technology M45PE Series Serial Flash Memory Control Software 1 Mbit

When using this application note's sample code with another microcontroller, the code must be modified to match the specifications of the microcontroller used and tested thoroughly.

Note that the term "RX Family microcontroller" is used in this document for ease of description since the target devices come from multiple groups.

Contents

1. Specifications

This sample program uses RX family microcontrollers RSPI clock synchronous (three-wire method) serial communications to perform clock synchronous control. SPI mode single master control can be implemented by adding SPI slave device selection control using port control.

Table 1.1 lists the used peripheral functions and their uses and figure 1.1 shows an example of the use of this application.

In the following, we present an overview of these functions.

- The sample program implements a clock synchronous single master block type device driver that uses the RSPI with an RX family microcontroller used as the master device.
- The microcontroller's built-in clock synchronous (three-wire method) serial communications function is used. A single channel set up by the user can also be used. Multiple channels cannot be used.
- This sample code does not support chip select control. If an SPI device is controlled, it will be necessary to provide device select control code separately.
- This sample code supports both big endian and little endian byte orders.
- Data is transferred in an MSB first format.
- Only CPU transfers are supported. DMAC, EXDMAC, and DTC transfers are not supported.
- Using interrupts to start transfers is not supported.
- Support for clock synchronous (three-wire method) single master transmit, single master receive, and single master transmit/receive
- Either normal receive mode or high-speed receive mode can be selected as the reception method.
- RSPI module without RSPCK auto-stop function: ALLOWS selection of normal mode or high-speed mode for reception or .transmit/receive
  Operation in supervisor mode is enabled when high-speed mode is selected. Operation in user mode is disabled.
  Also, NMI interrupts are disabled. In addition, there are intervals when interrupts are disabled during continuous receive operation or transmit/receive operation.
  Operation in either supervisor mode or user mode is supported when normal mode is selected.
- RSPI module with RSPCK auto-stop function: Supports reception in high-speed mode only.
  Operation in either supervisor mode or user mode is supported.

Table 1.1 Peripheral Devices and Uses

Peripheral Device	Use
RSPI	Clock synchronous (three-wire method) serial communications: 1 channel (required)
Port	Used for SPI slave device selection control A number of ports corresponding to the number of devices used are needed (required). Note, however, that ports are not used in this sample code.

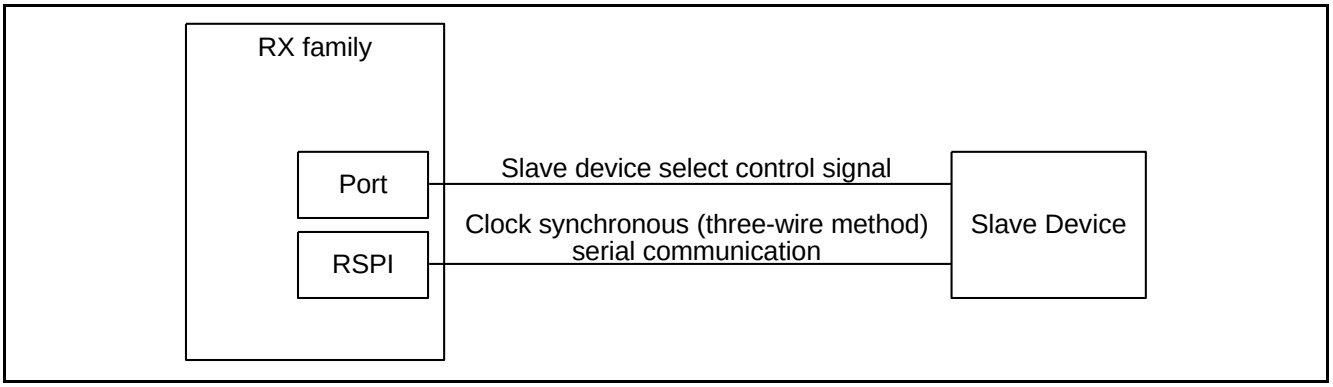


Figure 1.1 Usage Example

2. Verified Operating Conditions

Operation of this application note's sample code has been verified under the following conditions.

(1) For the RX210

Table 2.1 Verified Operating Conditions

Item	Description
Microcontroller used	RX210 Group (Program ROM: 512 KB, RAM: 64 KB)
Memory	Renesas Electronics Corporation R1EX25xxx Series SPI Serial EEPROM
Operating frequency	ICLK: 50 MHz, PCLKB: 25 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance embedded Workshop Version 4.09.00.007
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 1.2.1.0)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Renesas R1EX25xxx Series Serial EEPROM Control Software (R01AN0565EJ), Version 2.02
Board	Renesas Starter Kit for RX210

Table 2.2 Verified Operating Conditions

Item	Description
Microcontroller used	RX210 Group (Program ROM: 512 KB, RAM: 64 KB)
Memory	Micron Technology M25P Series Serial Flash Memory: 64 Mbits
Operating frequency	ICLK: 50 MHz, PCLKB: 25 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance embedded Workshop Version 4.09.00.007
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 1.2.1.0)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Micron Technology M25P Series Serial Flash Memory Control Software (R01AN0566EJ), Version 2.01
Board	Renesas Starter Kit for RX210

Table 2.3 Verified Operating Conditions

Item	Description
Microcontroller used	RX210 Group (Program ROM: 512 KB, RAM: 64 KB)
Memory	Micron Technology M45PE Series Serial Flash Memory: 1 Mbit
Operating frequency	ICLK: 50 MHz, PCLKB: 25 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance embedded Workshop Version 4.09.00.007
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 1.2.1.0)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Micron Technology M45PE Series Serial Flash Memory Control Software (R01AN0567EJ), Version 2.01
Board	Renesas Starter Kit for RX210

(2) For the RX21A

Table 2.4 Verified Operating Conditions

Item	Description
Microcontroller used	RX21A Group (Program ROM: 512 KB, RAM: 64 KB)
Memory	Renesas Electronics Corporation R1EX25xxx Series SPI Serial EEPROM
Operating frequency	ICLK: 50 MHz, PCLKB: 25 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance embedded Workshop Version 4.09.00.007
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 1.2.1.0)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Renesas R1EX25xxx Series Serial EEPROM Control Software (R01AN0565EJ), Version 2.02
Board	HSBRX21AP-B (Hokuto Denshi Co., Ltd.)

Table 2.5 Verified Operating Conditions

Item	Description
Microcontroller used	RX21A Group (Program ROM: 512 KB, RAM: 64 KB)
Memory	Micron Technology M25P Series Serial Flash Memory: 64 Mbits
Operating frequency	ICLK: 50 MHz, PCLKB: 25 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance embedded Workshop Version 4.09.00.007
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 1.2.1.0)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Micron Technology M25P Series Serial Flash Memory Control Software (R01AN0566EJ), Version 2.01
Board	HSBRX21AP-B (Hokuto Denshi Co., Ltd.)

Table 2.6 Verified Operating Conditions

Item	Description
Microcontroller used	RX21A Group (Program ROM: 512 KB, RAM: 64 KB)
Memory	Micron Technology M45PE Series Serial Flash Memory: 1 Mbit
Operating frequency	ICLK: 50 MHz, PCLKB: 25 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance embedded Workshop Version 4.09.00.007
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 1.2.1.0)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Micron Technology M45PE Series Serial Flash Memory Control Software (R01AN0567EJ), Version 2.01
Board	HSBRX21AP-B (Hokuto Denshi Co., Ltd.)

(3) For the RX220

Table 2.7 Verified Operating Conditions

Item	Description
Microcontroller used	RX220 Group (Program ROM: 256 KB, RAM: 16 KB)
Memory	Renesas Electronics Corporation R1EX25xxx Series SPI Serial EEPROM
Operating frequency	ICLK: 32 MHz, PCLKB: 32 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance embedded Workshop Version 4.09.01.007
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 1.2.1.0)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Renesas R1EX25xxx Series Serial EEPROM Control Software (R01AN0565EJ), Version 2.02
Board	Renesas Starter Kit for RX220

Table 2.8 Verified Operating Conditions

Item	Description
Microcontroller used	RX220 Group (Program ROM: 256 KB, RAM: 16 KB)
Memory	Micron Technology M25P Series Serial Flash Memory: 64 Mbits
Operating frequency	ICLK: 32 MHz, PCLKB: 32 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance embedded Workshop Version 4.09.01.007
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 1.2.1.0)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Micron Technology M25P Series Serial Flash Memory Control Software (R01AN0566EJ), Version 2.01
Board	Renesas Starter Kit for RX220

Table 2.9 Verified Operating Conditions

Item	Description
Microcontroller used	RX220 Group (Program ROM: 256 KB, RAM: 16 KB)
Memory	Micron Technology M45PE Series Serial Flash Memory: 1 Mbit
Operating frequency	ICLK: 32 MHz, PCLKB: 32 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance embedded Workshop Version 4.09.01.007
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 1.2.1.0)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Micron Technology M45PE Series Serial Flash Memory Control Software (R01AN0567EJ), Version 2.01
Board	Renesas Starter Kit for RX220

(4) For the RX63N

Table 2.10 Verified Operating Conditions

Item	Description
Microcontroller used	RX63N Group (Program ROM: 1 MB, RAM: 128 KB)
Memory	Renesas Electronics Corporation R1EX25xxx Series SPI Serial EEPROM
Operating frequency	ICLK: 96 MHz, PCLKB: 48 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance embedded Workshop Version 4.09.00.007
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 1.2.1.0)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Renesas R1EX25xxx Series Serial EEPROM Control Software (R01AN0565EJ), Version 2.02
Board	Renesas Starter Kit for RX63N

Table 2.11 Verified Operating Conditions

Item	Description
Microcontroller used	RX63N Group (Program ROM: 1 MB, RAM: 128 KB)
Memory	Micron Technology M25P Series Serial Flash Memory: 64 Mbits
Operating frequency	ICLK: 96 MHz, PCLKB: 48 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance embedded Workshop Version 4.09.00.007
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 1.2.1.0)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Micron Technology M25P Series Serial Flash Memory Control Software (R01AN0566EJ), Version 2.01
Board	Renesas Starter Kit for RX63N

Table 2.12 Verified Operating Conditions

Item	Description
Microcontroller used	RX63N Group (Program ROM: 1 MB, RAM: 128 KB)
Memory	Micron Technology M45PE Series Serial Flash Memory: 1 Mbit
Operating frequency	ICLK: 96 MHz, PCLKB: 48 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation High-performance embedded Workshop Version 4.09.00.007
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 1.2.1.0)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Micron Technology M45PE Series Serial Flash Memory Control Software (R01AN0567EJ), Version 2.01
Board	Renesas Starter Kit for RX63N

(5) For the RX63T

Table 2.13 Verified Operating Conditions

Item	Description
Microcontroller used	RX63T Group (Program ROM: 512 KB, RAM: 48 KB)
Memory	Renesas Electronics Corporation R1EX25xxx Series SPI Serial EEPROM
Operating frequency	ICLK: 96 MHz, PCLKB: 48 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation CubeSuite+ V2.00.00
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.00.00)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Renesas R1EX25xxx Series Serial EEPROM Control Software (R01AN0565EJ), Version 2.02
Board	Renesas Starter Kit for RX63T

Table 2.14 Verified Operating Conditions

Item	Description
Microcontroller used	RX63T Group (Program ROM: 512 KB, RAM: 48 KB)
Memory	Micron Technology M25P Series Serial Flash Memory: 64 Mbits
Operating frequency	ICLK: 96 MHz, PCLKB: 48 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation CubeSuite+ V2.00.00
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.00.00)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Micron Technology M25P Series Serial Flash Memory Control Software (R01AN0566EJ), Version 2.01
Board	Renesas Starter Kit for RX63T

Table 2.15 Verified Operating Conditions

Item	Description
Microcontroller used	RX63T Group (Program ROM: 512 KB, RAM: 48 KB)
Memory	Micron Technology M45PE Series Serial Flash Memory: 1 Mbit
Operating frequency	ICLK: 96 MHz, PCLKB: 48 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation CubeSuite+ V2.00.00
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.00.00)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R01
Software	Micron Technology M45PE Series Serial Flash Memory Control Software (R01AN0567EJ), Version 2.01
Board	Renesas Starter Kit for RX63T

(6) For the RX111

Table 2.16 Verified Operating Conditions

Item	Description
Microcontroller used	RX111 Group (Program ROM: 128KB, RAM: 16KB)
Memory	Renesas Electronics Corporation R1EX25xxx Series SPI Serial EEPROM
Operating frequency	ICLK: 32 MHz, PCLKB: 32 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation CubeSuite+ V2.01.00
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.01.00)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R03
Software	Renesas R1EX25xxx Series Serial EEPROM Control Software (R01AN0565EJ), Version 2.03
Board	Renesas Starter Kit for RX111

Table 2.17 Verified Operating Conditions

Item	Description
Microcontroller used	RX111 Group (Program ROM: 128KB, RAM: 16KB)
Memory	Micron Technology M25P Series Serial Flash Memory: 64 Mbits
Operating frequency	ICLK: 32 MHz, PCLKB: 32 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation CubeSuite+ V2.01.00
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.01.00)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R03
Software	Micron Technology M25P Series Serial Flash Memory Control Software (R01AN0566EJ), Version 2.02
Board	Renesas Starter Kit for RX111

Table 2.18 Verified Operating Conditions

Item	Description
Microcontroller used	RX111 Group (Program ROM: 128KB, RAM: 16KB)
Memory	Micron Technology M45PE Series Serial Flash Memory: 1 Mbit
Operating frequency	ICLK: 32 MHz, PCLKB: 32 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation CubeSuite+ V2.01.00
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.01.00)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.04.R03
Software	Micron Technology M45PE Series Serial Flash Memory Control Software (R01AN0567EJ), Version 2.02
Board	Renesas Starter Kit for RX111

(7) For the RX64M

Table 2.19 Verified Operating Conditions

Item	Description
Microcontroller used	RX64M Group (Program ROM: 4MB, RAM: 512KB)
Memory	Renesas Electronics Corporation R1EX25xxx Series SPI Serial EEPROM
Operating frequency	ICLK: 120 MHz, PCLKA: 120 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation e2studio V3.1.0.24
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.01.00)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.05
Software	Renesas R1EX25xxx Series Serial EEPROM Control Software (R01AN0565EJ), Version 2.03
Board	Renesas Starter Kit for RX64M

Table 2.20 Verified Operating Conditions

Item	Description
Microcontroller used	RX64M Group (Program ROM: 4MB, RAM: 512KB)
Memory	Micron Technology M25P Series Serial Flash Memory: 64 Mbits
Operating frequency	ICLK: 120 MHz, PCLKB: 120 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation e2studio V3.1.0.24
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.01.00)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.05
Software	Micron Technology M25P Series Serial Flash Memory Control Software (R01AN0566EJ), Version 2.02
Board	Renesas Starter Kit for RX64M

Table 2.21 Verified Operating Conditions

Item	Description
Microcontroller used	RX64M Group (Program ROM: 4MB, RAM: 512KB)
Memory	Micron Technology M45PE Series Serial Flash Memory: 1 Mbits
Operating frequency	ICLK: 120 MHz, PCLKB: 120 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation e2studio V3.1.0.24
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.01.00)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.05
Software	Micron Technology M45PE Series Serial Flash Memory Control Software (R01AN0567EJ), Version 2.02
Board	Renesas Starter Kit for RX64M

(8) For the RX71M

Table 2.22 Verified Operating Conditions

Item	Description
Microcontroller used	RX71M Group (Program ROM: 4MB, RAM: 512KB)
Memory	Renesas Electronics Corporation R1EX25xxx Series SPI Serial EEPROM
Operating frequency	ICLK: 240 MHz, PCLKA: 120 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation e2studio V4.3.0.8
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.04.01)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.05.R01
Software	Renesas R1EX25xxx Series Serial EEPROM Control Software (R01AN0565EJ), Version 2.04
Board	Renesas Starter Kit for RX71M

Table 2.23 Verified Operating Conditions

Item	Description
Microcontroller used	RX71M Group (Program ROM: 4MB, RAM: 512KB)
Memory	Micron Technology M25P Series Serial Flash Memory: 64 Mbits
Operating frequency	ICLK: 240 MHz, PCLKB: 120 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation e2studio V4.3.0.8
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.04.01)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.05.R01
Software	Micron Technology M25P Series Serial Flash Memory Control Software (R01AN0566EJ), Version 2.03
Board	Renesas Starter Kit for RX71M

Table 2.24 Verified Operating Conditions

Item	Description
Microcontroller used	RX71M Group (Program ROM: 4MB, RAM: 512KB)
Memory	Micron Technology M45PE Series Serial Flash Memory: 1 Mbits
Operating frequency	ICLK: 240 MHz, PCLKB: 120 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation e2studio V4.3.0.8
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.04.01)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.05.R01
Software	Micron Technology M45PE Series Serial Flash Memory Control Software (R01AN0567EJ), Version 2.04
Board	Renesas Starter Kit for RX71M

(9) For the RX634

Table 2.25 Verified Operating Conditions

Item	Description
Microcontroller used	RX634 Group (Program ROM: 2MB, RAM: 128KB)
Memory	Renesas Electronics Corporation R1EX25xxx Series SPI Serial EEPROM
Operating frequency	ICLK: 50 MHz, PCLKB: 25 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation e2studio V4.3.0.8
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.04.01)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.05.R01
Software	Renesas R1EX25xxx Series Serial EEPROM Control Software (R01AN0565EJ), Version 2.04
Board	Renesas Starter Kit for RX634

Table 2.26 Verified Operating Conditions

Item	Description
Microcontroller used	RX634 Group (Program ROM: 2MB, RAM: 128KB)
Memory	Micron Technology M25P Series Serial Flash Memory: 64 Mbits
Operating frequency	ICLK: 50 MHz, PCLKB: 25 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation e2studio V4.3.0.8
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.04.01)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.05.R01
Software	Micron Technology M25P Series Serial Flash Memory Control Software (R01AN0566EJ), Version 2.03
Board	Renesas Starter Kit for RX634

Table 2.27 Verified Operating Conditions

Item	Description
Microcontroller used	RX634 Group (Program ROM: 2MB, RAM: 128KB)
Memory	Micron Technology M45PE Series Serial Flash Memory: 1 Mbits
Operating frequency	ICLK: 50 MHz, PCLKB: 25 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics Corporation e2studio V4.3.0.8
C compiler	Renesas Electronics Corporation RX Family C/C++ Compiler Package (Toolchain 2.04.01)
	Compiler options The integrated development environment default settings*1 are used. Note: 1. Optimization level: 2, optimization method: Size priority
Endian order	Big endian / Little endian
Sample code version number	Ver. 2.05.R01
Software	Micron Technology M45PE Series Serial Flash Memory Control Software (R01AN0567EJ), Version 2.04
Board	Renesas Starter Kit for RX634

3. Related Application Notes

Related application notes are listed below. Refer to these when using this application note.

- Renesas R1EX25xxx Series Serial EEPROM Control Software (R01AN0565EJ)
- Micron Technology M25P Series Serial Flash Memory Control Software (R01AN0566EJ)
- Micron Technology M45PE Series Serial Flash Memory Control Software (R01AN0567EJ)
- Micron Technology N25Q Serial NOR Flash Memory Control Software (R01AN1528EJ)
- Micron Technology P5Q Serial Phase Change Memory Control Software (R01AN1439EJ)
- Spansion S25FLxxxS MirrorBit® Flash Non-Volatile Memory Control Software (R01AN1529EJ)
- Macronix International MX25/66L Family Serial NOR Flash Memory Control Software (R01AN1967EJ)

4. Peripheral Functions

The RSPI module supports two types of operation: SPI operation (four-wire method) and clock synchronous operation (three-wire method).

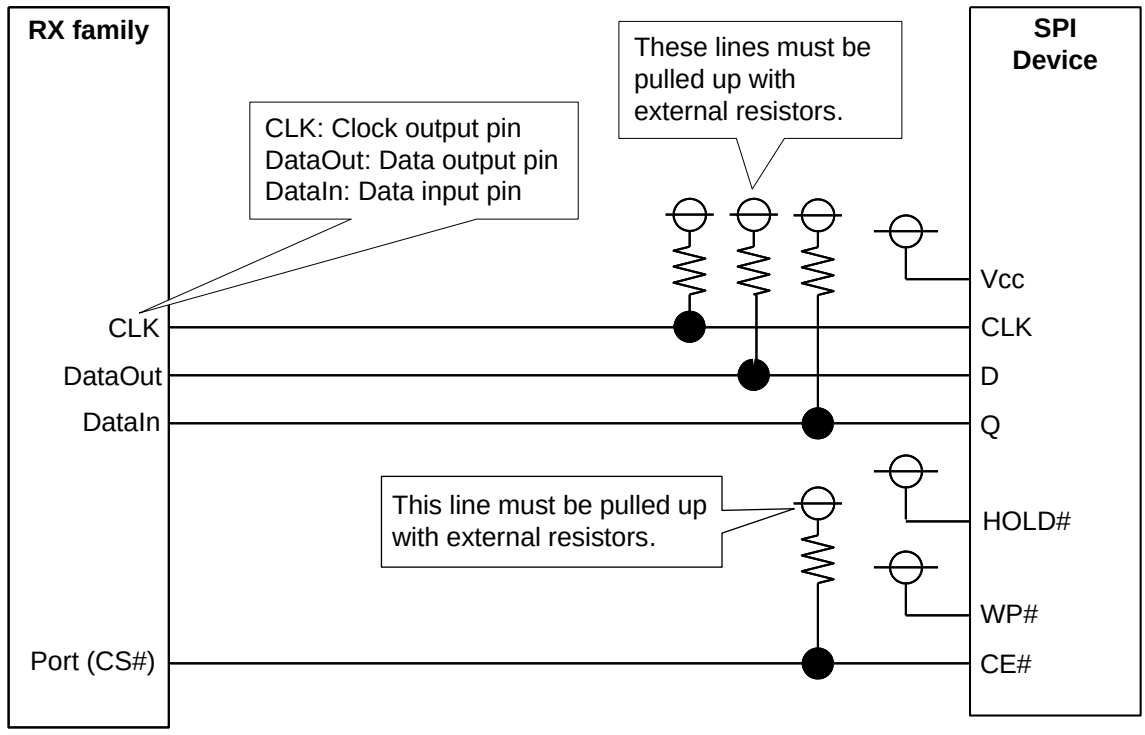
This application note uses clock synchronous operation (three-wire method). In this sample code, a port is allocated as the SPI slave device select pin when an SPI device is controlled.

The SSL pin used with the RSPI four-wire method can be allocated as a CE# pin for port control when three-wire method is used.

5. Hardware Description

5.1 Reference Circuit

Figure 5.1 shows the device connection circuit diagram. Note that if the hardware will be operated at high speed, a damping resistor and capacitor should be added for circuit matching for each signal line.



- The pin names used for serial I/O on the microcontroller depend on the microcontroller used.
- In this application note, the pins used are expressed as the CLK, DataIn, DataOut, and Port (CS#) pins to match the notation used in the sample code.

Figure 5.1 Connection Between RX Family Microcontrollers RSPI and SPI Slave Device

5.2 List of Pins

Table 5.1 lists the pins used and their functions.

Table 5.1 Pins and Usage

Pin Name	I/O	Description
RSPCK (CLK in figure 5.1)	Output	Clock output
MOSI (DataOut in figure 5.1)	Output	Master data output
MISO (DataIn in figure 5.1)	Input	Master data input
Port (Port(CS#) in figure 5.1)	Output	Slave device select output Note, however, that this pin is not handled by this sample code.

6. Software Description

6.1 Operation Overview

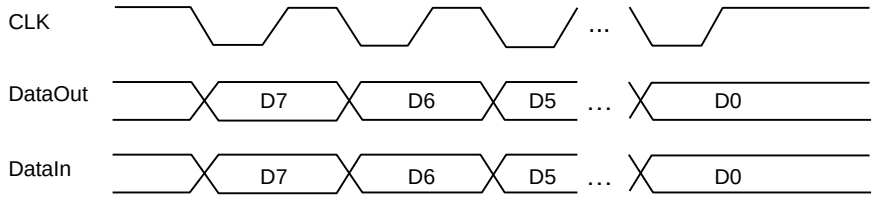
This sample code uses the RSPI module's clock synchronous (three-wire method) serial communication function to implement clock synchronous single master control.

This sample code implements the following control operation.

- Control of data transmit/receive operations in clock synchronous operation (using an internal clock).

6.1.1 Timing Generated in Clock Synchronous Operation

This sample code generates the SPI mode 3 (CPOL = 1, CPHA = 1) timing shown in figure 6.1, which is required for SPI slave device control.



- For microcontroller to SPI slave device transmission: Transmit data output starts on the falling edge of the transfer clock
- For SPI slave device to microcontroller reception: Receive data input is acquired on the rising edge of the transfer clock
- Data is transferred MSB first.

The CLK pin is held at the high level when no transfer is being performed.

Figure 6.1 Timing Settings for Clock Synchronous Operation

Check the microcontroller and SPI slave device datasheets for the serial clock frequencies that can be used.

6.1.2 SPI Slave Device CE# Pin Control

This sample code does not control the SPI slave device CE# pin. To control an SPI device, the user must provide SPI slave device CE# pin control separately.

As the control method, we recommend connecting to a microcontroller port and controlling the SPI device with the microcontroller general-purpose port output.

Also, the application must provide time from the fall of the SPI device CE# (microcontroller port CS#) signal to the fall of the SPI device CLK (the microcontroller CLK) signal.

Similarly, the application must provide time from the rise of the SPI device CLK (the microcontroller CLK) signal to the rise of the SPI device CE# (microcontroller port CS#) signal.

Check the SPI device data sheet, and implement the application with software wait times appropriate for the system.

The SSL pin used in four-wire method with the RSPI module may be allocated to the CE# pin in port control for three-wire method.

6.2 Software Control Outline

6.2.1 Software Structure

This sample code implements a single master basic control method that is unique to the microcontroller.

In particular, this sample code implements control that uses SPI mode 3 (CPOL = 1, CPHA = 1) without control of the SPI slave device CE# pin.

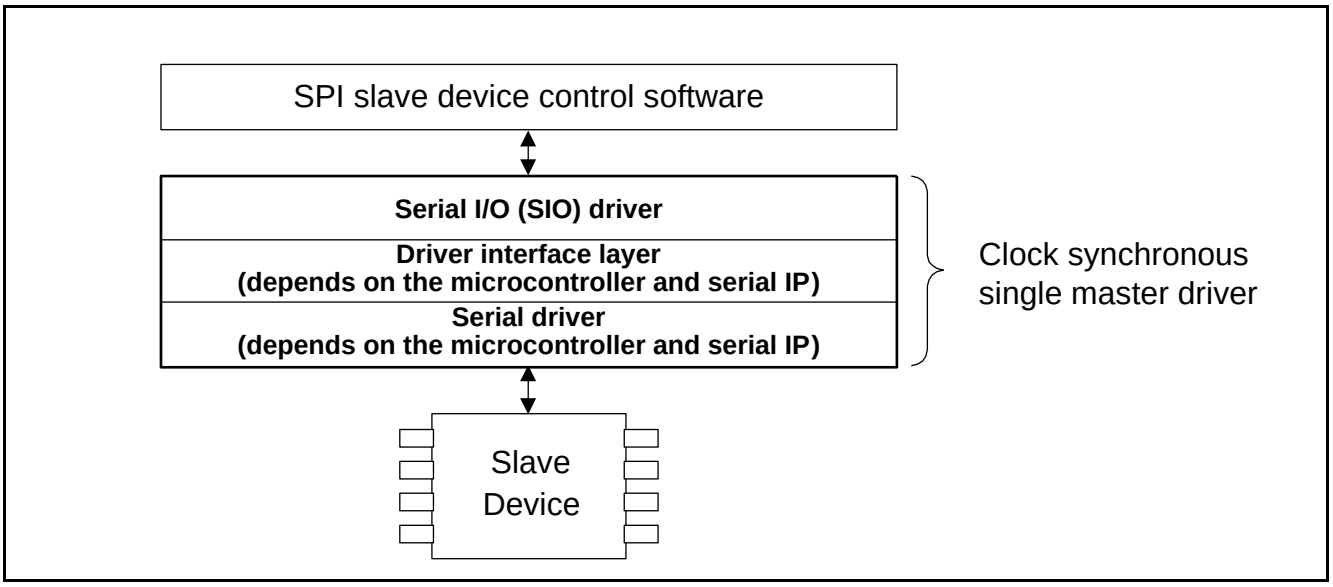


Figure 6.2 Software Structure

The user must implement slave device access by referring to the functions shown in section 6.8, State Transition Diagram, and section 6.9, Function Specifications.

Refer to the previously mentioned section 3, Related Application Notes for specific application examples.

6.2.2 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.

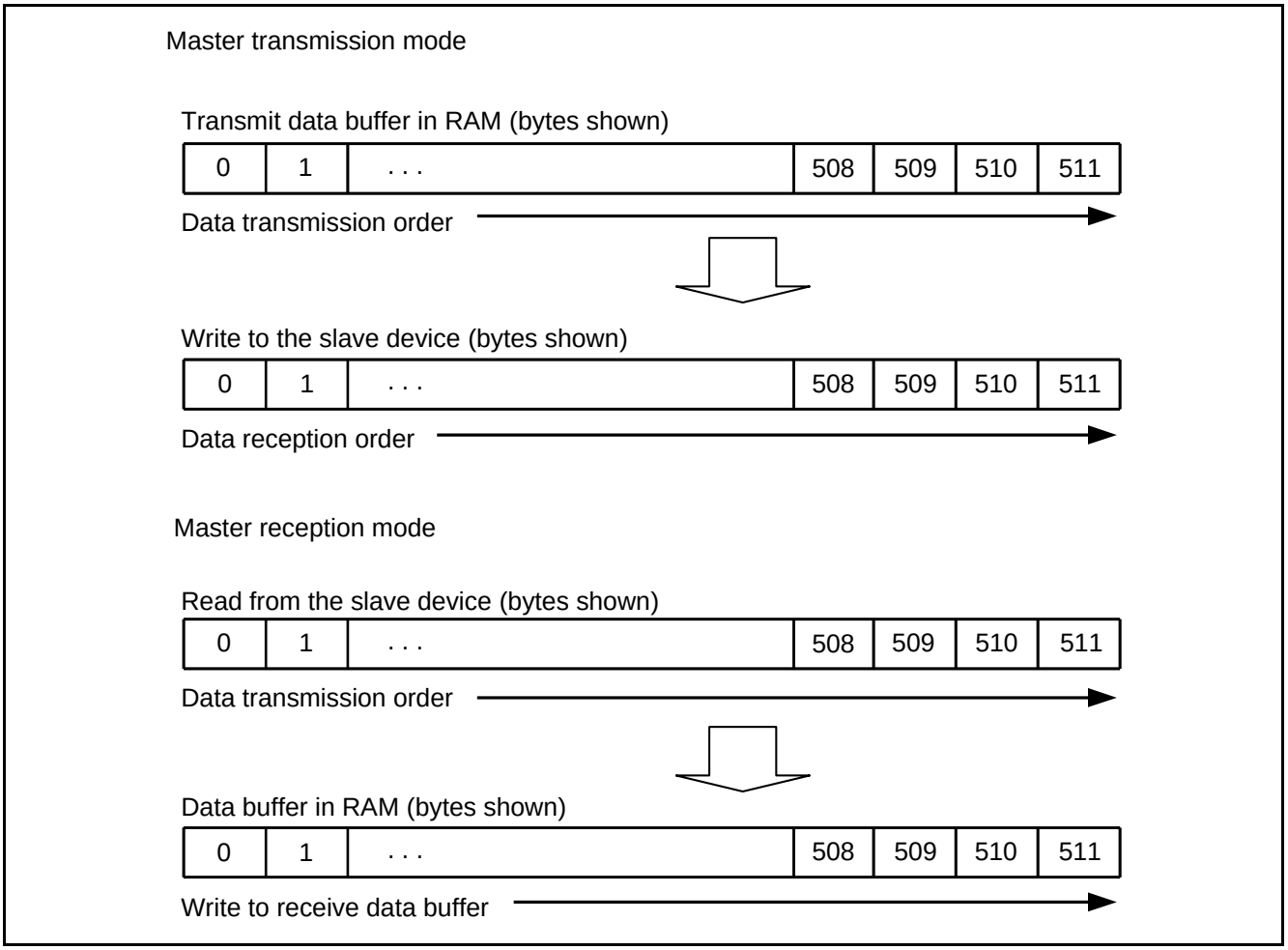


Figure 6.3 Relationship Between Data Buffers and Transmit/Receive Data

6.3 Size of Required Memory

Table 6.1 lists the memory requirements.

The memory sizes listed in table 6.1 apply when SIO_OPTION_4 is selected with the operating mode definition used in section 7.2.2, R_SIO_rspi.h (1). The memory requirements differ depending on the selected definition.

(1) For the RX210

Table 6.1 Memory Requirements

Memory Used	Size	Remarks
ROM	2,617 bytes (little endian)	R_SIO_rspi_rx.c
RAM	0 byte (little endian)	R_SIO_rspi_rx.c
Maximum user stack usage	84 bytes	
Maximum interrupt stack usage	—	

Note: The memory requirements may differ with the version of the C compiler used or with the compiler options specified.
The above memory requirements may differ depending on the endian order selected.

(2) For the RX21A

Table 6.2 Memory Requirements

Memory Used	Size	Remarks
ROM	2,545 bytes (little endian)	R_SIO_rspi_rx.c
RAM	0 byte (little endian)	R_SIO_rspi_rx.c
Maximum user stack usage	84 bytes	
Maximum interrupt stack usage	—	

Note: The memory requirements may differ with the version of the C compiler used or with the compiler options specified.
The above memory requirements may differ depending on the endian order selected.

(3) For the RX220

Table 6.3 Memory Requirements

Memory Used	Size	Remarks
ROM	2,533 bytes (little endian)	R_SIO_rspi_rx.c
RAM	0 byte (little endian)	R_SIO_rspi_rx.c
Maximum user stack usage	84 bytes	
Maximum interrupt stack usage	—	

Note: The memory requirements may differ with the version of the C compiler used or with the compiler options specified.
The above memory requirements may differ depending on the endian order selected.

(4) For the RX63N

Table 6.4 Memory Requirements

Memory Used	Size	Remarks
ROM	2,617 bytes (little endian)	R_SIO_rspi_rx.c
RAM	0 byte (little endian)	R_SIO_rspi_rx.c
Maximum user stack usage	84 bytes	
Maximum interrupt stack usage	—	

Note: The memory requirements may differ with the version of the C compiler used or with the compiler options specified.
The above memory requirements may differ depending on the endian order selected.

(5) For the RX63T

Table 6.5 Memory Requirements

Memory Used	Size	Remarks
ROM	2,328 bytes (little endian)	R_SIO_rspi_rx.c
RAM	0 byte (little endian)	R_SIO_rspi_rx.c
Maximum user stack usage	72 bytes	
Maximum interrupt stack usage	—	

Note: The memory requirements may differ with the version of the C compiler used or with the compiler options specified.
The above memory requirements may differ depending on the endian order selected.

(6) For the RX111

Table 6.6 Memory Requirements

Memory Used	Size	Remarks
ROM	2,156 bytes (little endian)	R_SIO_rspi_rx.c
RAM	0 byte (little endian)	R_SIO_rspi_rx.c
Maximum user stack usage	72 bytes	
Maximum interrupt stack usage	—	

Note: The memory requirements may differ with the version of the C compiler used or with the compiler options specified.
The above memory requirements may differ depending on the endian order selected.

(7) For the RX64M

Table 6.7 Memory Requirements

Memory Used	Size	Remarks
ROM	4,313 bytes (little endian)	R_SIO_rspi_rx.c
RAM	0 byte (little endian)	R_SIO_rspi_rx.c
Maximum user stack usage	87 bytes	
Maximum interrupt stack usage	—	

Note: The memory requirements may differ with the version of the C compiler used or with the compiler options specified.
The above memory requirements may differ depending on the endian order selected.
ROM size has increased than previous version because of adding transmit/receive operation.

(8) For the RX71M

Table 6.8 Memory Requirements

Memory Used	Size	Remarks
ROM	5,037 bytes (little endian)	R_SIO_rspi_rx.c
RAM	0 byte (little endian)	R_SIO_rspi_rx.c
Maximum user stack usage	68 bytes	
Maximum interrupt stack usage	—	

Note: The memory requirements may differ with the version of the C compiler used or with the compiler options specified.
The above memory requirements may differ depending on the endian order selected.
ROM size has increased than previous version because of adding transmit/receive operation.

(9) For the RX634

Table 6.9 Memory Requirements

Memory Used	Size	Remarks
ROM	3,663 bytes (little endian)	R_SIO_rspi_rx.c
RAM	0 byte (little endian)	R_SIO_rspi_rx.c
Maximum user stack usage	68 bytes	
Maximum interrupt stack usage	—	

Note: The memory requirements may differ with the version of the C compiler used or with the compiler options specified.
The above memory requirements may differ depending on the endian order selected.
ROM size has increased than previous version because of adding transmit/receive operation.

6.4 File Configuration

Table 6.6 lists the files used by the sample code. Note that the files automatically generated by the integrated development environment are not included.

Table 6.10 File Configuration

\an_r01an1196ej0111_rx_serial		<DIR>	Sample code folder
r01an1196ej0111_rx.pdf			Application note
\source		<DIR>	Program folder
\com		<DIR>	Common function folder
Note 1	mtl_com.c		Common function definitions
	mtl_com.h.common		Common header file
	mtl_com.h.RX		Common functions header file
	mtl_endi.c		Common files (endian setting related)
	mtl_mem.c		Common files (Standard library functions)
	mtl_os.c	mtl_os.h	Common files (Standard library functions)
	mtl_str.c		Common files (Standard library functions)
	mtl_tim.c	mtl_tim.h	Common files (Loop timer related)
	mtl_tim.h.sample		Sample loop timer settings
\r_sio_rspi_rx		<DIR>	Folder for clock synchronous single master control software using the RSPI
	R_SIO.h		Header file
	R_SIO_rspi.h.rx21a		Interface module common definitions (RX21A)
	R_SIO_rspi.h.rx63n		Interface module common definitions (RX63N)
	R_SIO_rspi.h.rx63t		Interface module common definitions (RX63T)
	R_SIO_rspi.h.rx64m		Interface module common definitions (RX64M)
	R_SIO_rspi.h.rx71m		Interface module common definitions (RX71M)
	R_SIO_rspi.h.rx111		Interface module common definitions (RX111)
	R_SIO_rspi.h.rx210		Interface module common definitions (RX210)
	R_SIO_rspi.h.rx220		Interface module common definitions (RX220)
	R_SIO_rspi.h.rx634		Interface module common definitions (RX634)
	R_SIO_rspi_rx.c		Interface module

Note: 1. The files held in the com folder are also used by the slave device control software. Use the latest versions of these files.

6.5 List of Constants

6.5.1 Return Values

Table 6.7 lists the return values used in the sample code.

Table 6.11 Return Values

Constant Name	Value	Description
SIO_OK	(error_t)(0)	Successful operation
SIO_ERR_PARAM	(error_t)(-1)	Parameter error
SIO_ERR_HARD	(error_t)(-2)	Hardware error
SIO_ERR_OTHER	(error_t)(-7)	Other error

6.5.2 Definitions

Table 6.8 lists the values for certain definitions used in the sample code.

Table 6.12 Return Values

Constant Name	Value	Description
SIO_LOG_ERR	(uint8_t)0x01	Log type: Error
SIO_TRUE	(uint8_t)0x01	Flag "ON"
SIO_FALSE	(uint8_t)0x00	Flag "OFF"
SIO_HI	(uint8_t)0x01	Port "H"
SIO_LOW	(uint8_t)0x00	Port "L"
SIO_OUT	(uint8_t)0x01	Port output setting
SIO_IN	(uint8_t)0x00	Port input setting
SIO_TX_WAIT	(uint16_t)50000	SIO transmission completion waiting time 50000* 1 us = 50 ms
SIO_RX_WAIT	(uint16_t)50000	SIO reception completion waiting time 50000* 1 us = 50 ms
SIO_DMA_TX_WAIT	(uint16_t)50000	DMA transmission completion waiting time 50000* 1 us = 50 ms
SIO_DMA_RX_WAIT	(uint16_t)50000	DMA reception completion waiting time 50000* 1 us = 50 ms
SIO_T_SIO_WAIT	(uint16_t)MTL_T_1US	SIO transmission&reception completion waiting polling time
SIO_T_DMA_WAIT	(uint16_t)MTL_T_1US	DMA transmission&reception completion waiting polling time
SIO_T_BRR_WAIT	(uint16_t)MTL_T_10US	BRR setting wait time

6.5.3 Other Definitions

Table 6.9 lists the values of certain other definitions used in the sample code.

Table 6.13 Return Values

Constant Name	Value	Description
SIO_TRAN_SIZE	(uint8_t)0x04	4 bytes (This value may not be changed.)

6.6 Structures and Unions

The structures used in the sample code are shown below.

```
/* uint32_t <-> uint8_t conversion */
typedef union {
  uint32_t  ul;
  uint8_t uc[4];
} SIO_EXCHG_LONG;                          /* total 4byte                */

/* uint16_t <-> uint8_t conversion */
typedef union {
  uint16_t  us;
  uint8_t uc[2];
} SIO_EXCHG_SHORT;                         /* total 2byte                */
```

6.7 List of Functions

Table 6.10 lists the functions in the sample code.

Table 6.14 List of Functions

Function Name	Outline
R_SIO_Init_Driver()	Driver initialization
R_SIO_Disable()	Disables serial I/O
R_SIO_Enable()	Enables serial I/O
R_SIO_Open_Port()	Releases serial I/O
R_SIO_Tx_Data()	Transmits serial I/O data
R_SIO_Rx_Data()	Receives serial I/O data
R_SIO_TRx_Data()	Transmits and Receives serial I/O data

To increase the speed of RSPI control operations, 32-bit access is used for the SPDR registers. When specifying a transmit/receive data buffer pointer, we recommend assuring that the start address falls on a 4-byte boundary to increase the speed of this processing.

6.8 State Transition Diagram

Figure 6.4 shows the state transition diagram for this system.

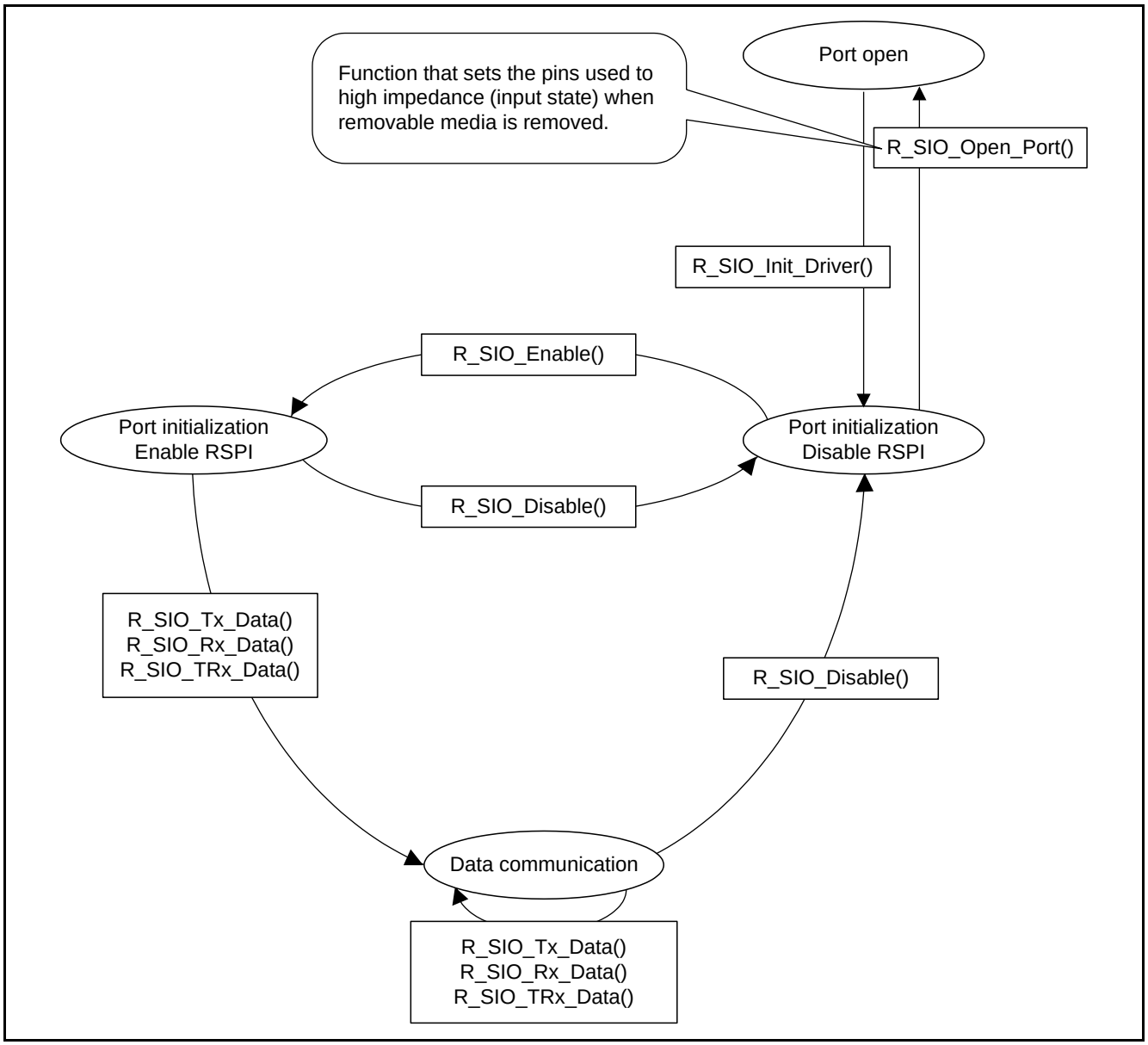


Figure 6.4 State Transition Diagram

6.9 Function Specifications

6.9.1 Driver Initialization Processing

R_SIO_Init_Driver	
Outline	Driver initialization processing
Header	R_SIO.h, R_SIO_rspi.h, mtl_com.h
Declaration	error_t R_SIO_Init_Driver(void)
Description	Initializes the driver. Disables the serial I/O function and sets the pins to their port function. This function must be called exactly once when the system starts. Set the slave device select control signal to the high level before calling this function.
Arguments	None
Return value	SIO_OK ; Successful operation
Notes	The following processing, which takes into account the previous state, is performed. The function R_SIO_Disable() is called.

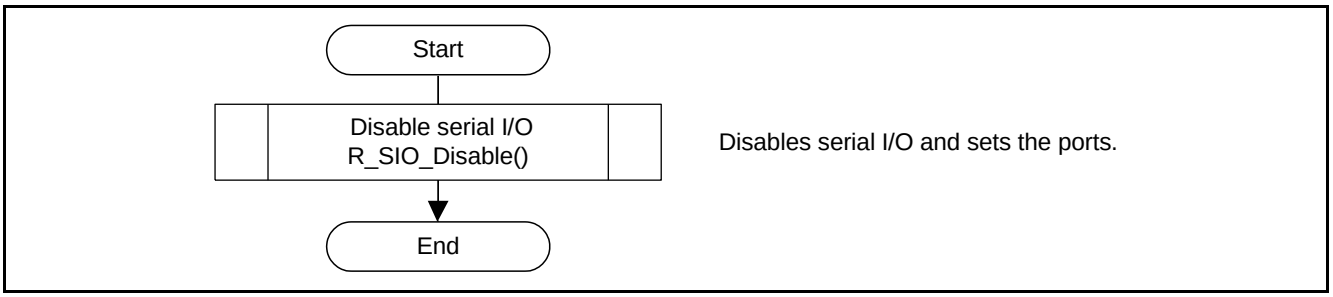


Figure 6.5 Driver Initialization Processing Outline

6.9.2 Serial I/O Disable Setup Processing

R_SIO_Disable	
Outline	Serial I/O disable setup processing
Header	R_SIO.h, R_SIO_rspi.h, mtl_com.h
Declaration	error_t R_SIO_Disable(void)
Description	Disables the serial I/O function and sets the pins to their port function. Disables serial I/O. Sets the pins used for serial I/O to their port function. Set the slave device select control signal to the high level before calling this function.
Arguments	None
Return value	SIO_OK ; Successful operation
Notes	The RSPI module stop state is canceled temporarily to write to the RSPI related registers. After setting the RSPI related registers, the module is set back to the module stop state. If not used, this function can be called to disable the serial I/O function.

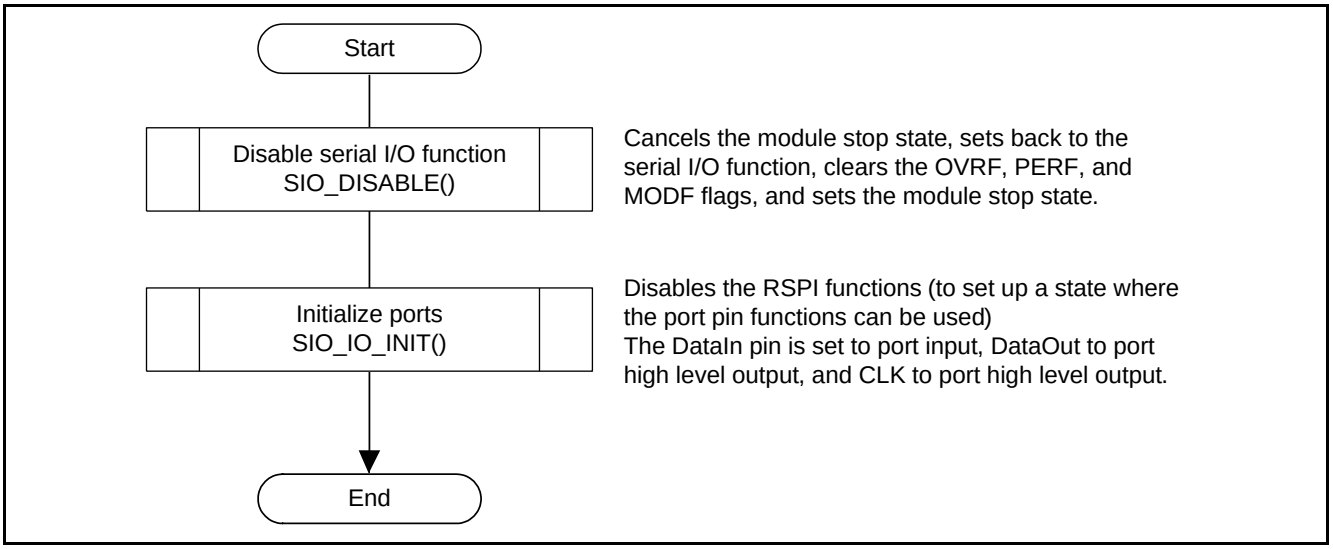


Figure 6.6 Serial I/O Disable Setup Processing Outline

6.9.3 Serial I/O Enable Setup Processing

R_SIO_Enable

Outline	Serial I/O enable setup processing
Header	R_SIO.h, R_SIO_rspi.h, mtl_com.h
Declaration	error_t R_SIO_Enable(uint8_t BrgData)
Description	Enables the serial I/O function and sets the bit rate. Sets the pins used by serial I/O to their port function. Enables serial I/O and sets the bit rate. Call this function only after calling R_SIO_Disable(). This function must be called before performing either serial I/O data transmission or serial I/O data reception. Use this function to change the bit rate. But before doing that, first call the disable serial I/O function.
Arguments	uint8_t BrgData ; Bit rate setting
Return value	SIO_OK ; Successful operation
Notes	This function sets the serial I/O module used to the module stop canceled state. The software wait (10 µs) is the wait time required to set the bit rate.

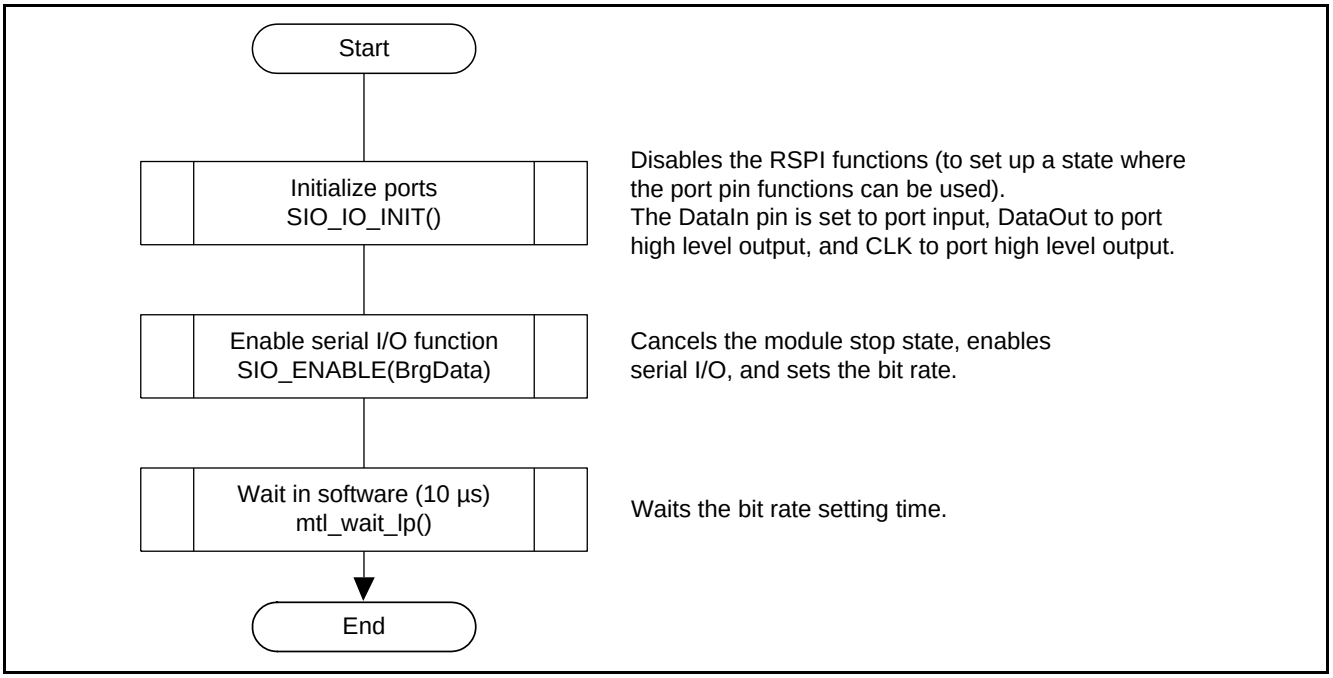


Figure 6.7 Serial I/O Enable Setup Processing Outline

6.9.4 Serial I/O Open Setup Processing

R_SIO_Open_Port

Outline	Serial I/O open setup processing
Header	R_SIO.h, R_SIO_rspi.h, mtl_com.h
Declaration	error_t R_SIO_Open_Port(void)
Description	Sets the pins used for serial I/O to open (the input state). Set the slave device select control signal to the high level before calling this function.
Arguments	None
Return value	SIO_OK ; Successful operation
Notes	This function is provided for inserting and removing removable media. Use this function before inserting or removing removable media. Perform the serial I/O disable setup processing before removing removable media.

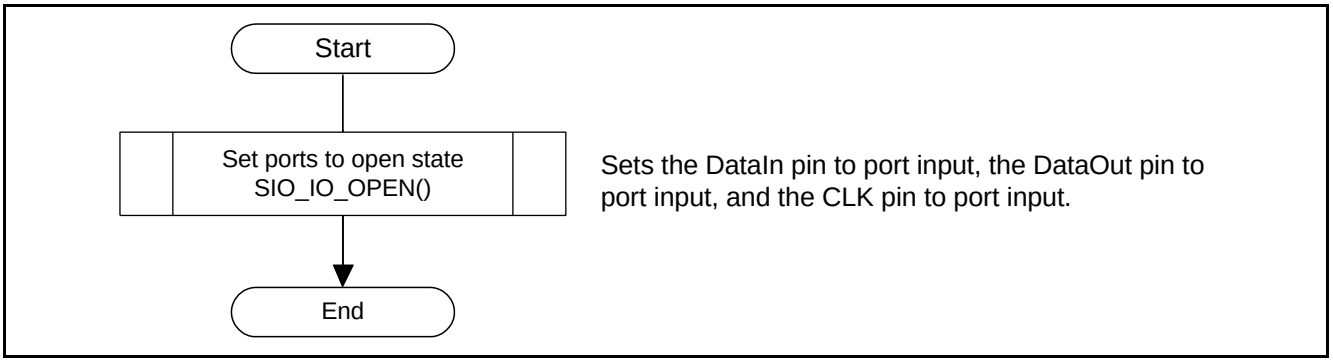


Figure 6.8 Serial I/O Open Setup Processing Outline

6.9.5 Serial I/O Data Transmission Processing

R_SIO_Tx_Data

Outline	Serial I/O data transmission processing
Header	R_SIO.h, R_SIO_rspi.h, mtl_com.h
Declaration	error_t R_SIO_Tx_Data(uint16_t TxCnt, uint8_t FAR* pData)
Description	Transmits the specified number of bytes of data from pData. The serial I/O enable setup processing must be performed prior to calling this function. The serial I/O disable setup processing must be performed if the result of this function indicates that an error occurred.
Arguments	uint16_t TxCnt ; Number of bytes to transmit uint8_t FAR* pData ; Pointer to transmit data buffer
Return value	SIO_OK ; Successful operation SIO_ERR_HARD ; Hardware error
Notes	Use this function for half-duplex transmission. The following operations, which follow the initialization flowchart shown in the hardware manual, are performed. (the inline function SIO_TX_ENABLE()) (1) Sets SPCR2 (enables RSPI idle interrupt requests). (2) Sets SPCMD. (3) Clears the IR flags and the internally held interrupt requests. (4) Sets the multi-function pin controller (MPC) (enables RSPI pins). (5) Sets SPCR (enables transmission). (6) Reads SPCR. After transmission completes, serial communication is disabled by the reverse of the enable processing shown above. (The inline function SIO_TX_DISABLE()) (1) Sets SPCR (stops transmission and reception). (2) Reads SPCR. (3) Sets the multi-function pin controller (MPC) (disables RSPI pins). (4) Clears the IR flags and the internally held interrupt requests. (5) Sets SPCR2 (disables RSPI idle interrupt requests). Both the transmit buffer empty IR and the RSPI idle IR are used to verify the completion of data transmission. We recommend performing the serial I/O disable setup processing if serial I/O is not to be used sequentially.

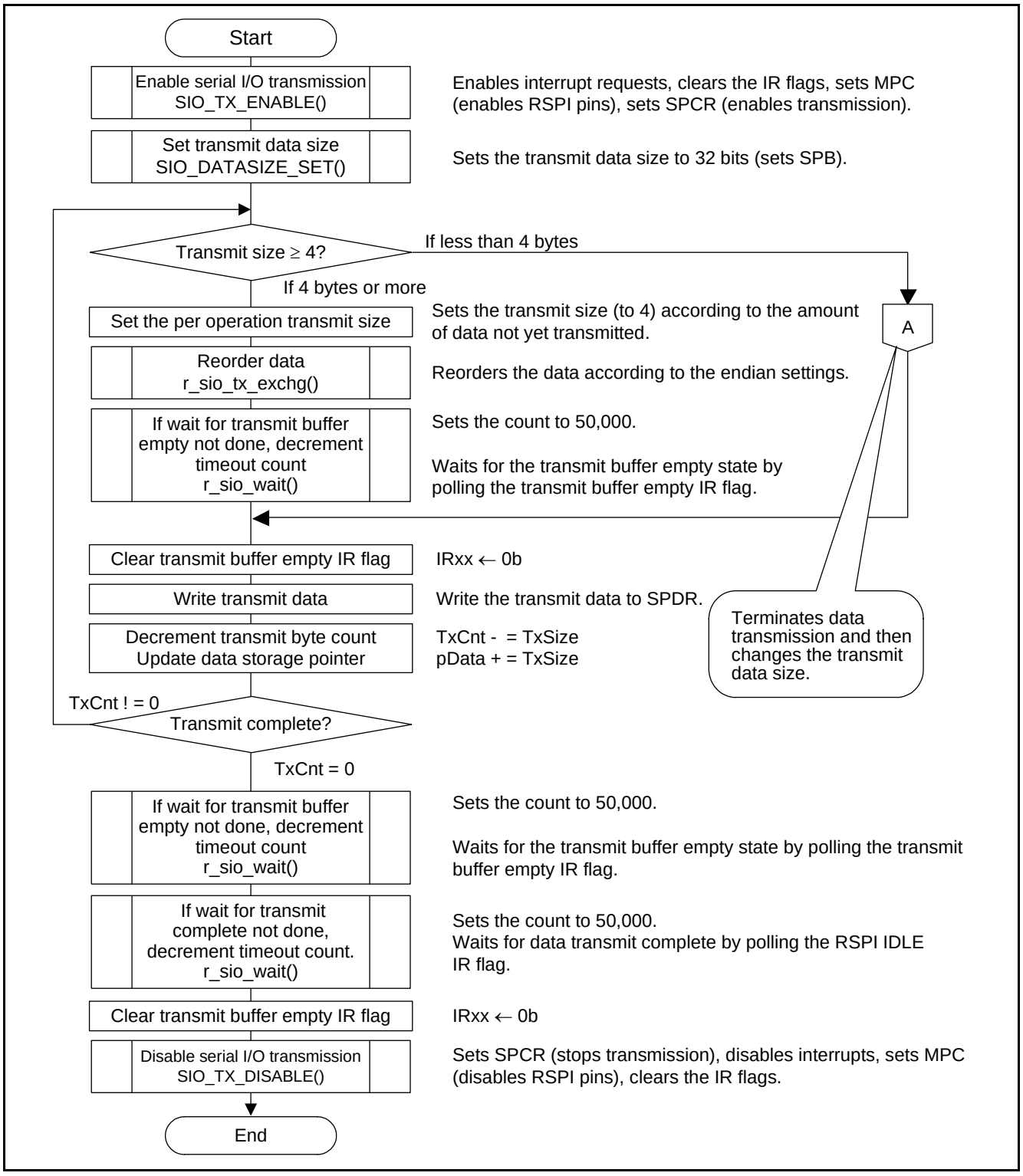


Figure 6.9 Serial I/O Data Transmission Processing Outline 1

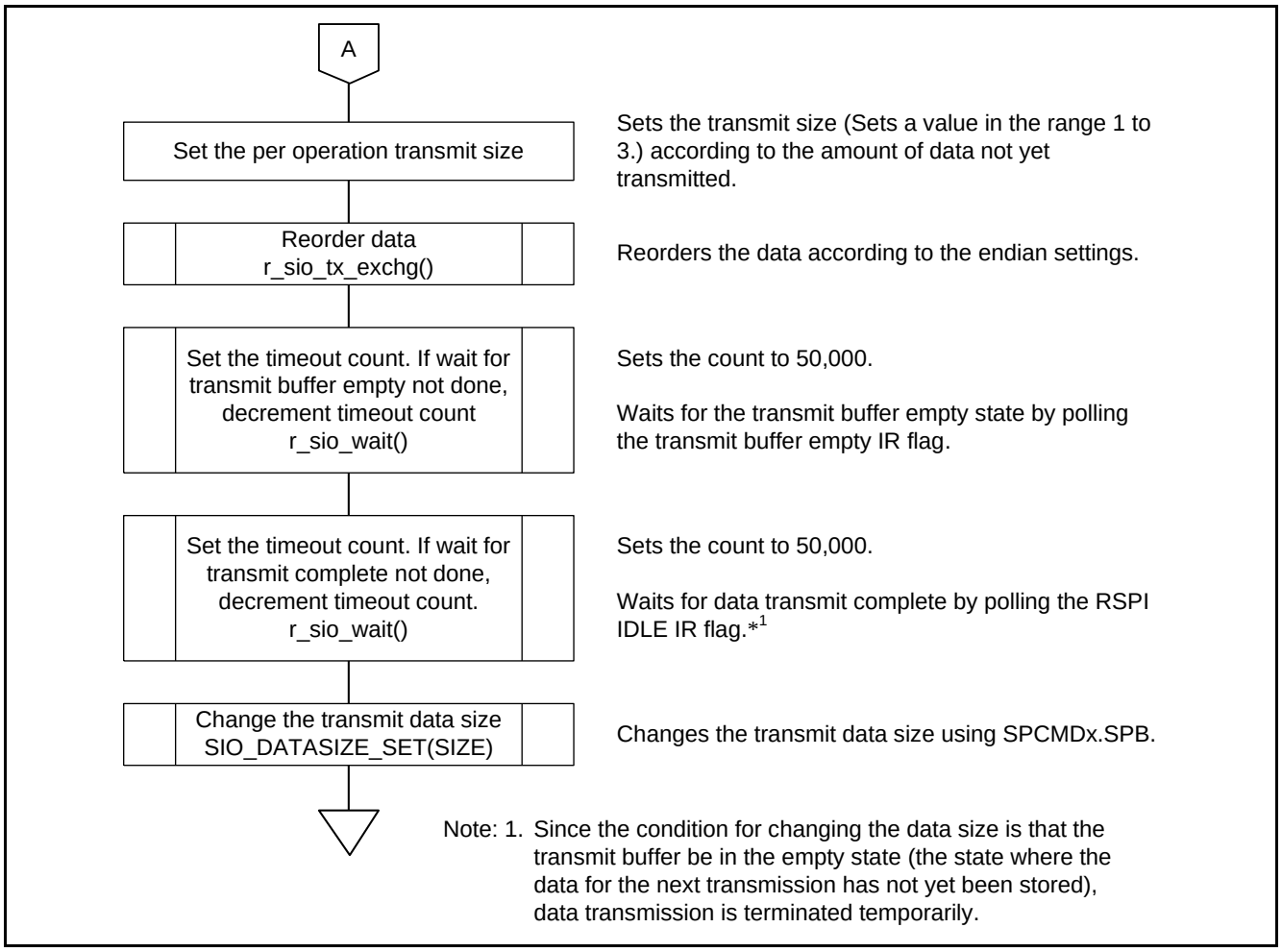


Figure 6.10 Serial I/O Data Transmission Processing Outline 2

6.9.6 Serial I/O Data Reception Processing

R_SIO_Rx_Data

Outline	Serial I/O data reception processing
Header	R_SIO.h, R_SIO_rspi.h, mtl_com.h
Declaration	error_t R_SIO_Rx_Data(uint16_t RxCnt, uint8_t FAR* pData)
Description	Receives the specified number of bytes of data and stores it in pData. The serial I/O enable setup processing must be performed prior to calling this function. The serial I/O disable setup processing must be performed if the result of this function indicates that an error occurred. Either normal reception or high-speed reception may be selected. For the selection method, see section 7.2.2, R_SIO_rspi.h (1) for the definitions of the operation modes used. An overview of normal reception is shown in figure 6.11, Serial I/O Data Reception Processing Outline — 1 (Normal) and figure 6.12, Serial I/O Data Reception Processing Outline — 2 (Normal). An overview of normal reception is shown in figure 6.13, Serial I/O Data Reception Processing Outline — 3 (High-Speed) and figure 6.14, Serial I/O Data Reception Processing Outline — 4 (High-Speed).
Arguments	uint16_t RxCnt ; Reception byte count uint8_t FAR* pData ; Pointer to receive data storage buffer
Return value	SIO_OK ; Successful operation SIO_ERR_HARD ; Hardware error
Notes	Use this function for half-duplex reception. The following operations, which follow the initialization flowchart shown in the hardware manual, are performed. (the inline function SIO_TRX_ENABLE()) (1) Sets SPCMD. (2) Clears the IR flags and the internally held interrupt requests. (3) Sets the multi-function pin controller (MPC) (enables RSPI pins). (4) Sets SPCR (enables transmission/reception). (5) Reads SPCR. After reception completes, serial communication is disabled by the reverse of the enable processing shown above. (The inline function SIO_TRX_DISABLE()) (1) Sets SPCR (stops transmission/reception). (2) Reads SPCR. (3) Sets the multi-function pin controller (MPC) (disables RSPI pins). (4) Clears the IR flags and the internally held interrupt requests. We recommend performing the disable serial I/O processing if serial I/O is not to be used sequentially. The following processing is added for high-speed reception. (1) To prevent overrun errors*1 from occurring during continuous reception, interrupts are disabled from the immediately before the next dummy write to the point where the previous receive data has been acquired. The interrupts disabled state is implemented by setting the processor interrupt priority level (IPL[3:0]) to the highest level. (2) The dummy data writes for the third and following continuous reception operations are performed after data reception has completed. This allows other interrupts to be accepted during continuous reception operations. Note: 1. Overrun errors may occur if there is contention for a shared bus between a DMAC, EXDMAC, or DTC transfer performed by another programs and this reception operation, or if a high-priority NMI interrupt occurs.

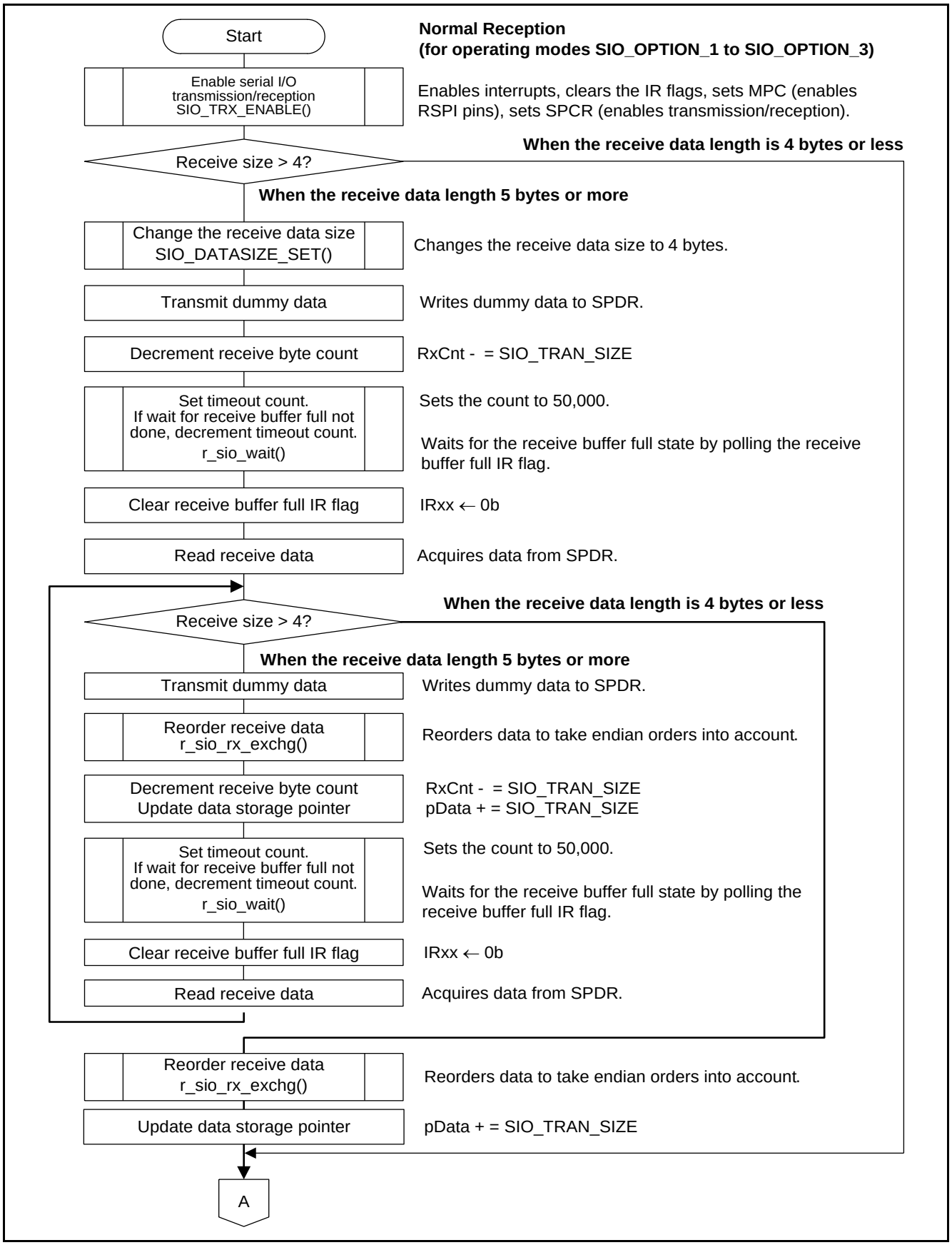


Figure 6.11 Serial I/O Data Reception Processing Outline 1 (Normal)

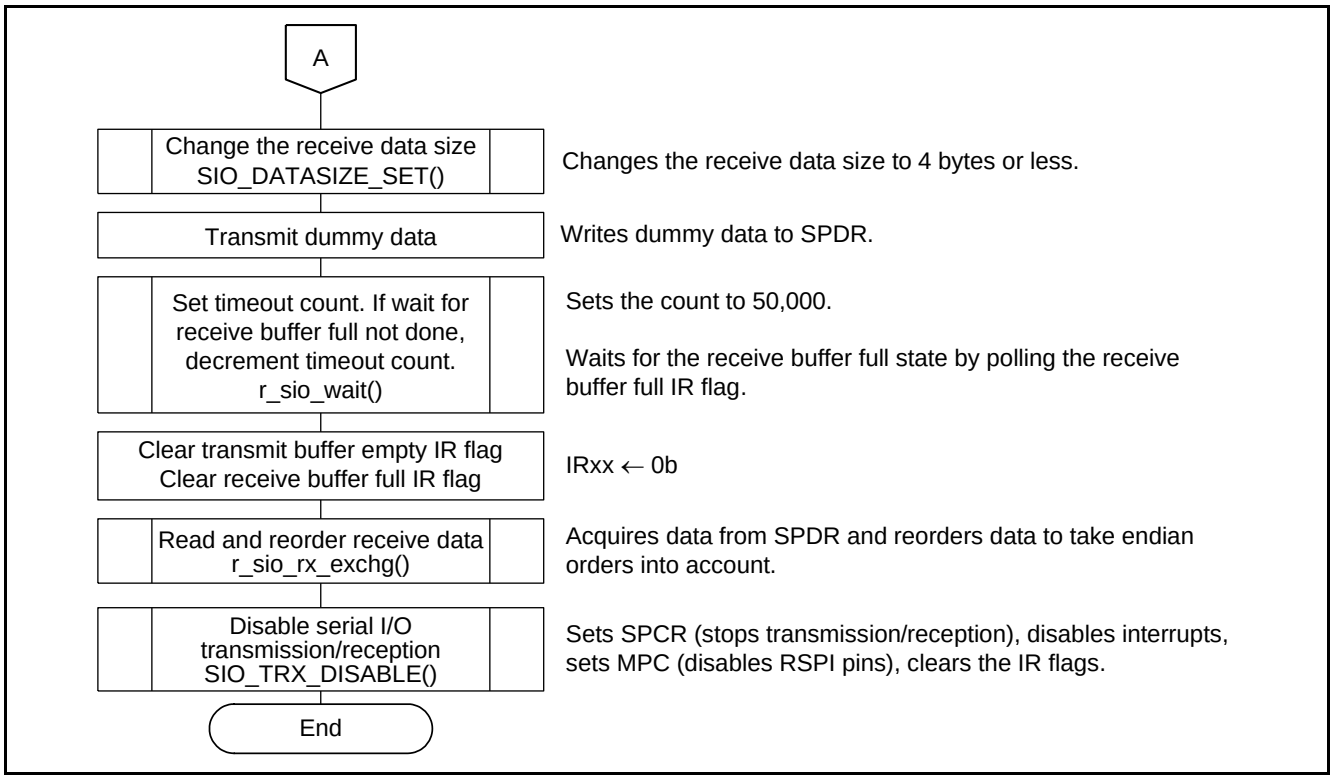


Figure 6.12 Serial I/O Data Reception Processing Outline 2 (Normal)

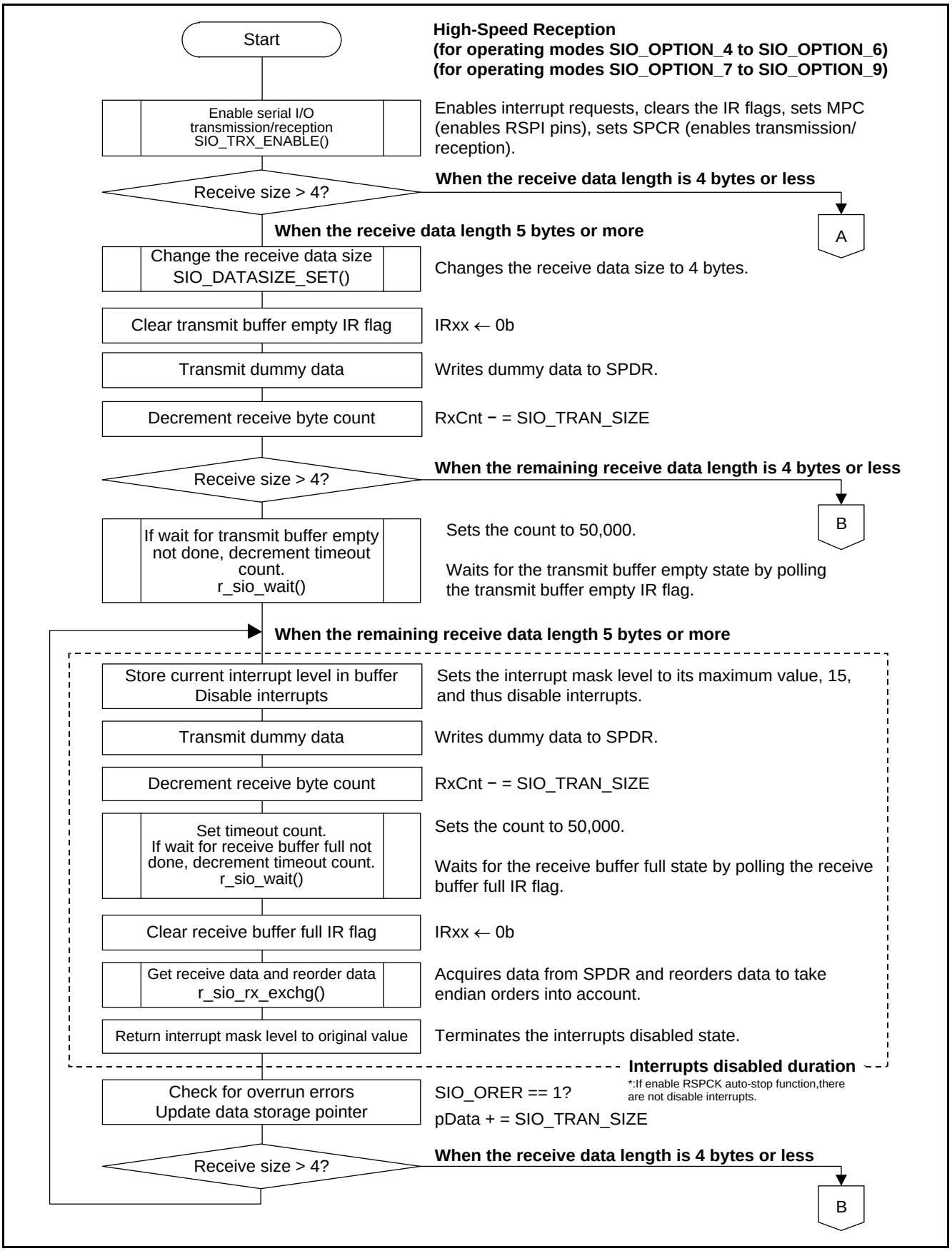


Figure 6.13 Serial I/O Data Reception Processing Outline 3 (High-Speed)

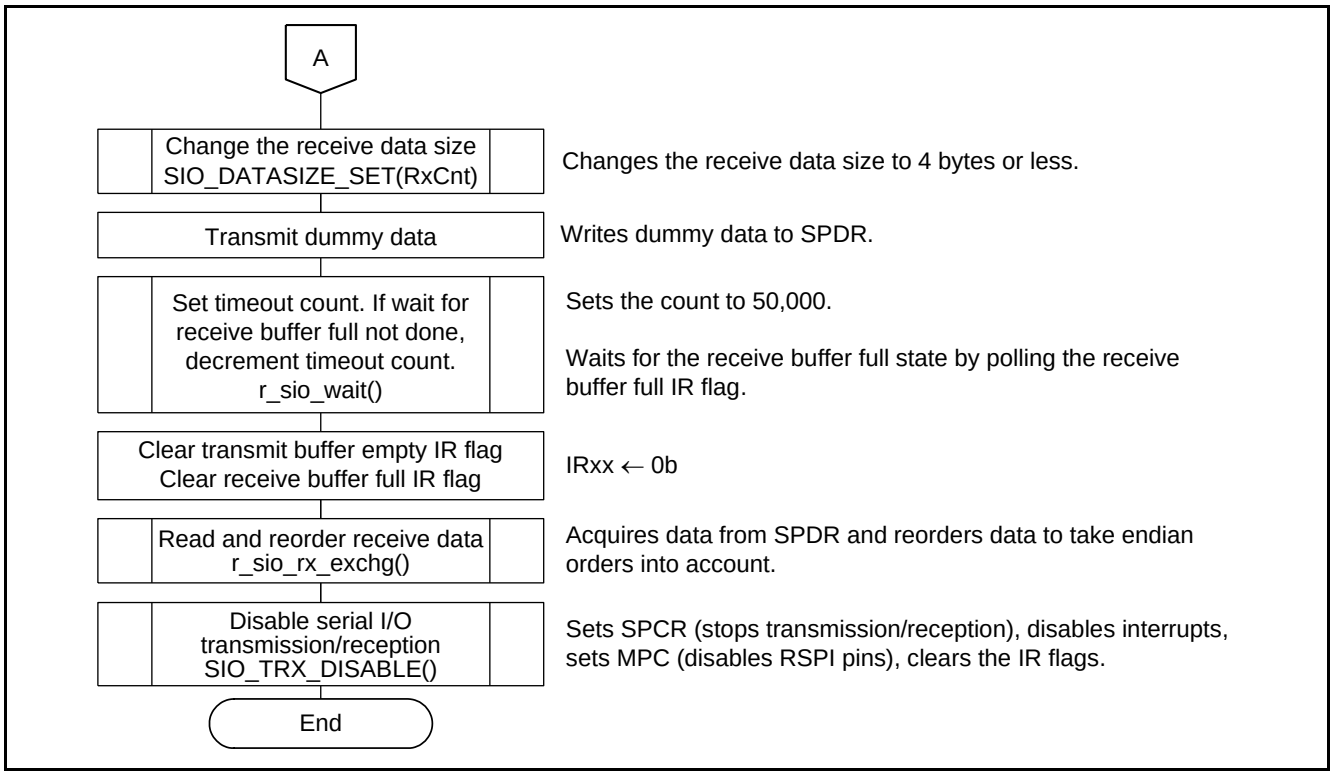


Figure 6.14 Serial I/O Data Reception Processing Outline 4 (High-Speed)

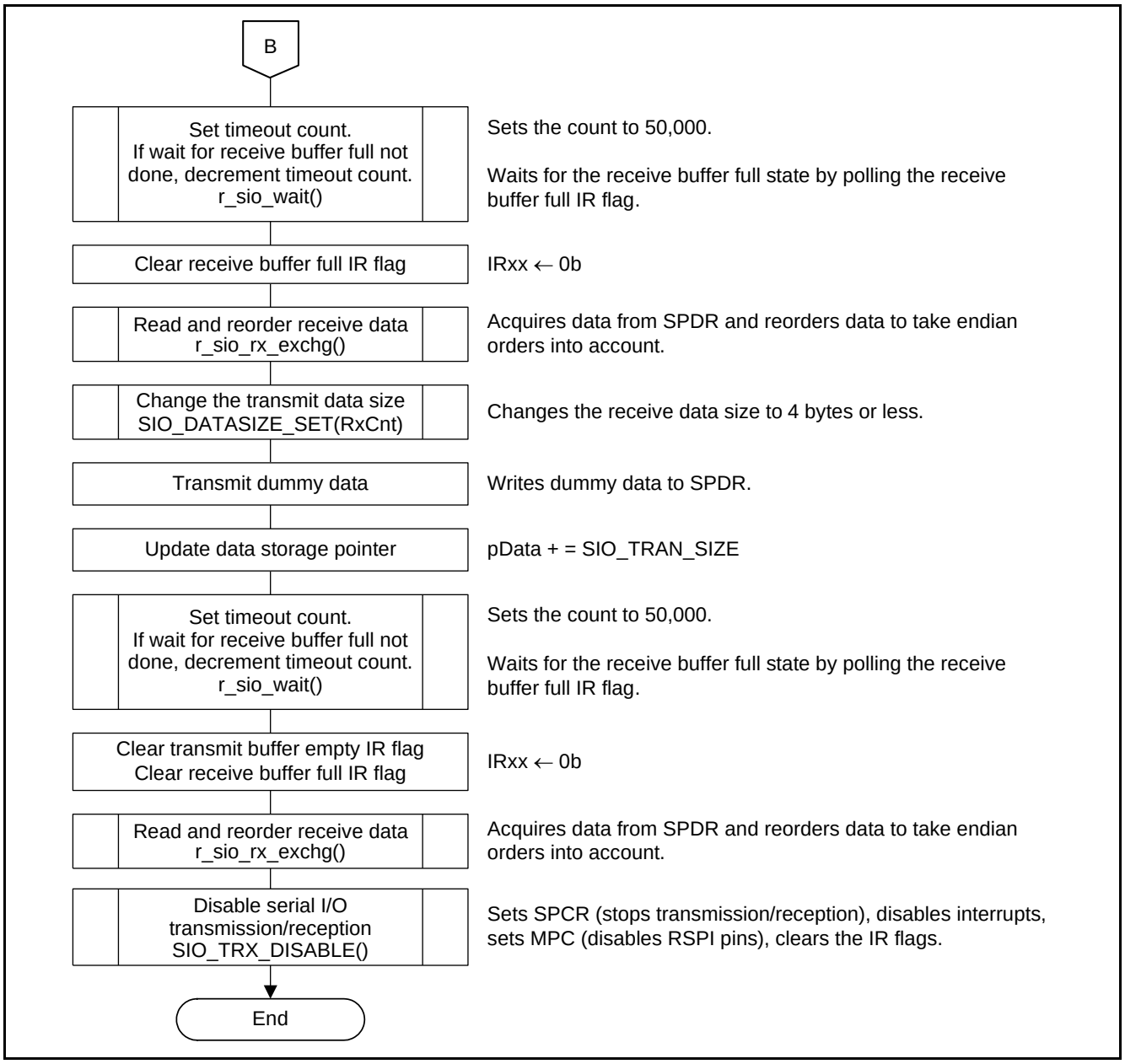


Figure 6.15 Serial I/O Data Reception Processing Outline 5 (High-Speed)

6.9.7 Serial I/O Data Transmission/Reception Processing

R_SIO_Rx_Data

Outline	Serial I/O data transmission/reception processing
Header	R_SIO.h, R_SIO_rspi.h, mtl_com.h
Declaration	error_t R_SIO_Rx_Data(uint16_t TRxCnt, uint8_t FAR* pTxData, uint8_t FAR* pRxData)
Description	Transmits the specified number of bytes of data from pData and receives the specified number of bytes of data and stores it in pData. The serial I/O enable setup processing must be performed prior to calling this function. The serial I/O disable setup processing must be performed if the result of this function indicates that an error occurred. Either normal transmission/reception or high-speed transmission/reception may be selected. For the selection method, see section 7.2.2, R_SIO_rspi.h (1) for the definitions of the operation modes used. An overview of normal transmission/reception is shown in figure 6.16, Serial I/O Data Reception Processing Outline — 1 (Normal) and figure 6.17, Serial I/O Data Transmission/Reception Processing Outline — 2 (Normal). An overview of normal Transmission/reception is shown in figure 6.18, Serial I/O Data Transmission/Reception Processing Outline — 3 (High-Speed) and figure 6.19, Serial I/O Data Transmission/Reception Processing Outline — 4 (High-Speed).
Arguments	uint16_t TRxCnt ; Transmission/Reception byte count uint8_t FAR* pTxData; Pointer to transmit data buffer uint8_t FAR* pRxData; Pointer to receive data storage buffer
Return value	SIO_OK ; Successful operation SIO_ERR_HARD ; Hardware error
Notes	Use this function for half-duplex reception. The following operations, which follow the initialization flowchart shown in the hardware manual, are performed. (the inline function SIO_TRX_ENABLE()) (1) Sets SPCMD. (2) Clears the IR flags and the internally held interrupt requests. (3) Sets the multi-function pin controller (MPC) (enables RSPI pins). (4) Sets SPCR (enables transmission/reception). (5) Reads SPCR. After transmission/reception completes, serial communication is disabled by the reverse of the enable processing shown above. (The inline function SIO_TRX_DISABLE()) (1) Sets SPCR (stops transmission/reception). (2) Reads SPCR. (3) Sets the multi-function pin controller (MPC) (disables RSPI pins). (4) Clears the IR flags and the internally held interrupt requests. We recommend performing the disable serial I/O processing if serial I/O is not to be used sequentially. The following processing is added for high-speed reception. (1) To prevent overrun errors*1 from occurring during continuous reception, interrupts are disabled from the immediately before the next dummy write to the point where the previous receive data has been acquired. The interrupts disabled state is implemented by setting the processor interrupt priority level (IPL[3:0]) to the highest level. (2) The dummy data writes for the third and following continuous reception operations are performed after data reception has completed. This allows other interrupts to be accepted during continuous reception operations. Note: 1. Overrun errors may occur if there is contention for a shared bus between a DMAC, EXDMAC, or DTC transfer performed by another programs and this reception operation, or if a high-priority NMI interrupt occurs.

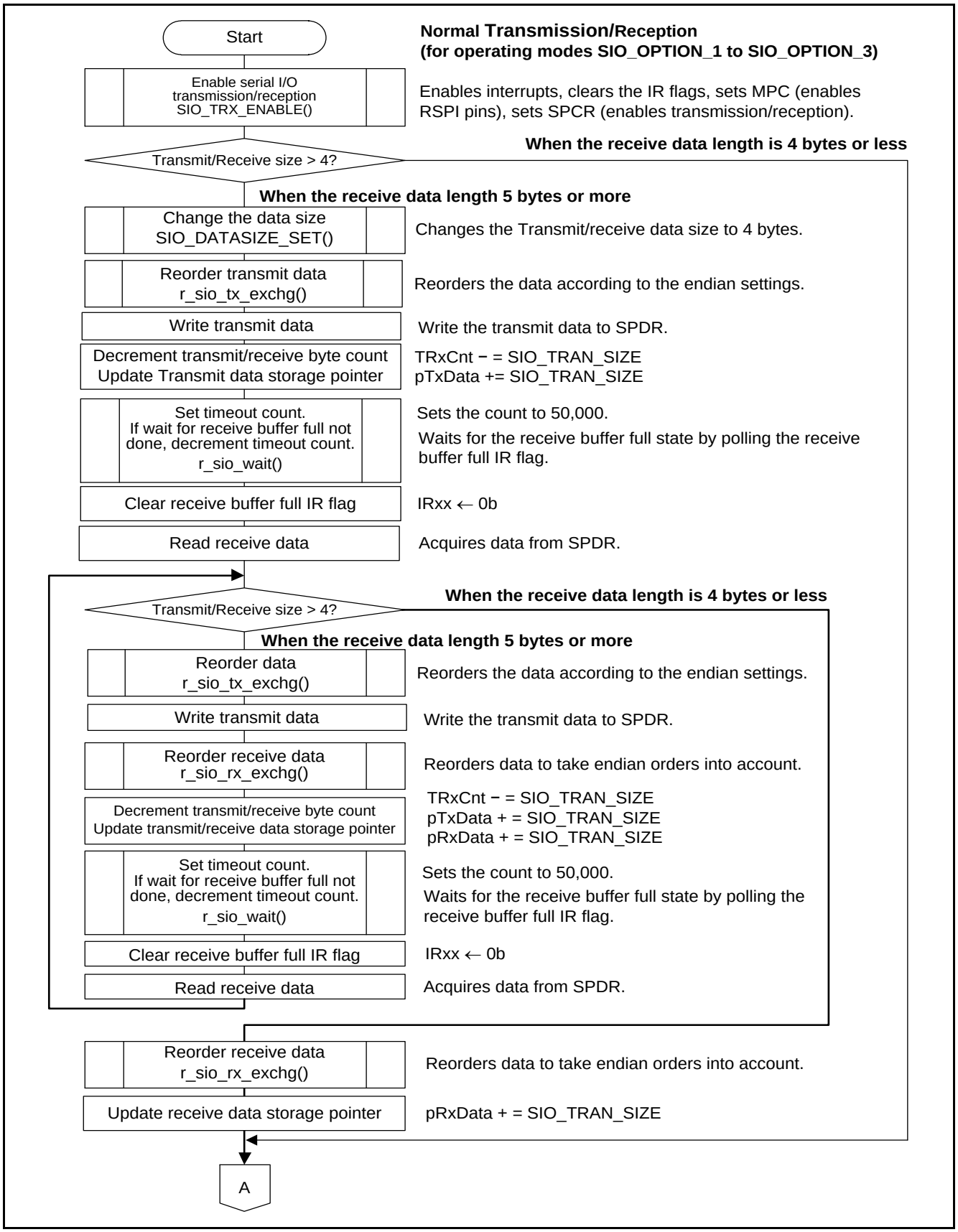


Figure 6.16 Serial I/O Data Transmission/Reception Processing Outline 1 (Normal)

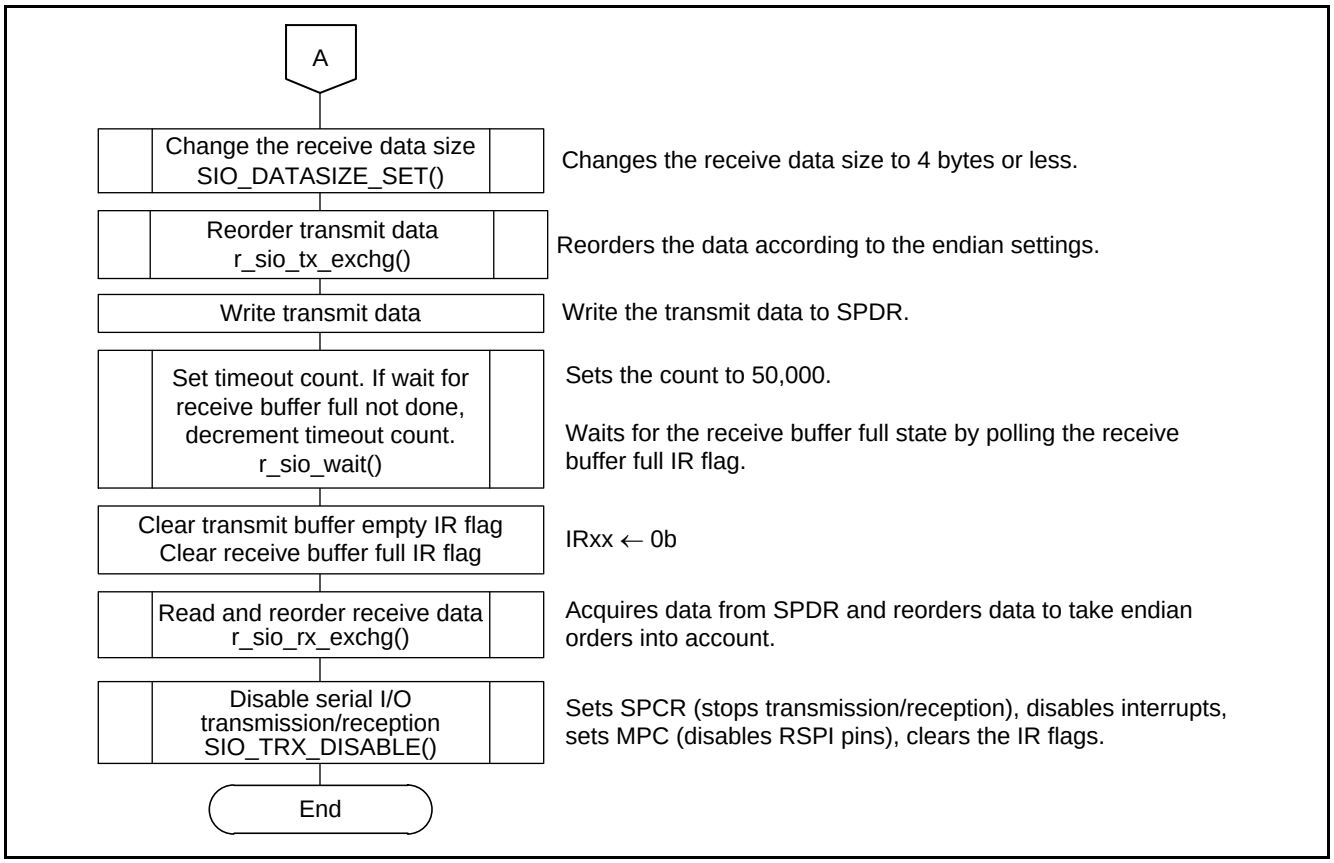


Figure 6.17 Serial I/O Data Transmission/Reception Processing Outline 2 (Normal)

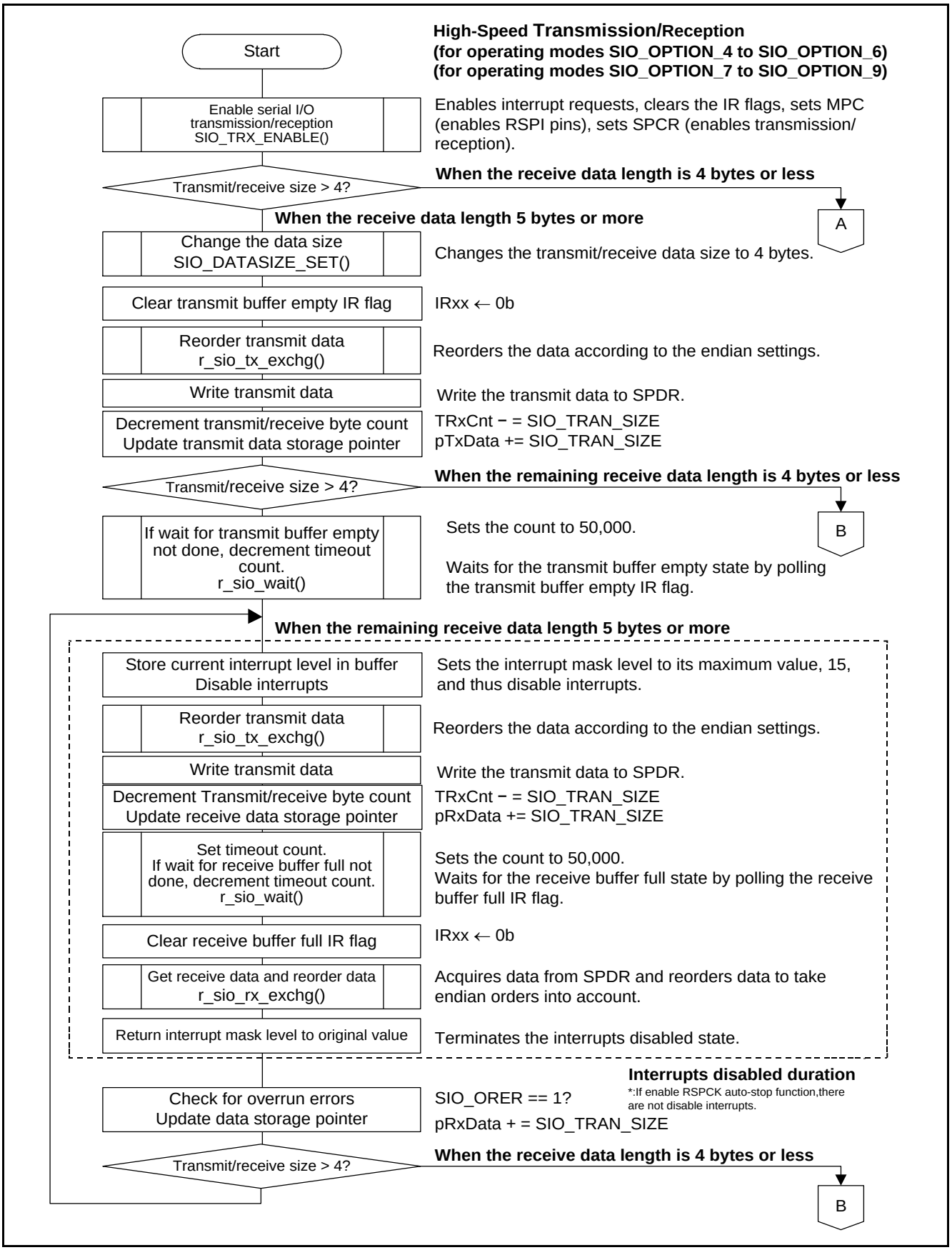


Figure 6.18 Serial I/O Data Transmission/Reception Processing Outline 3 (High-Speed)

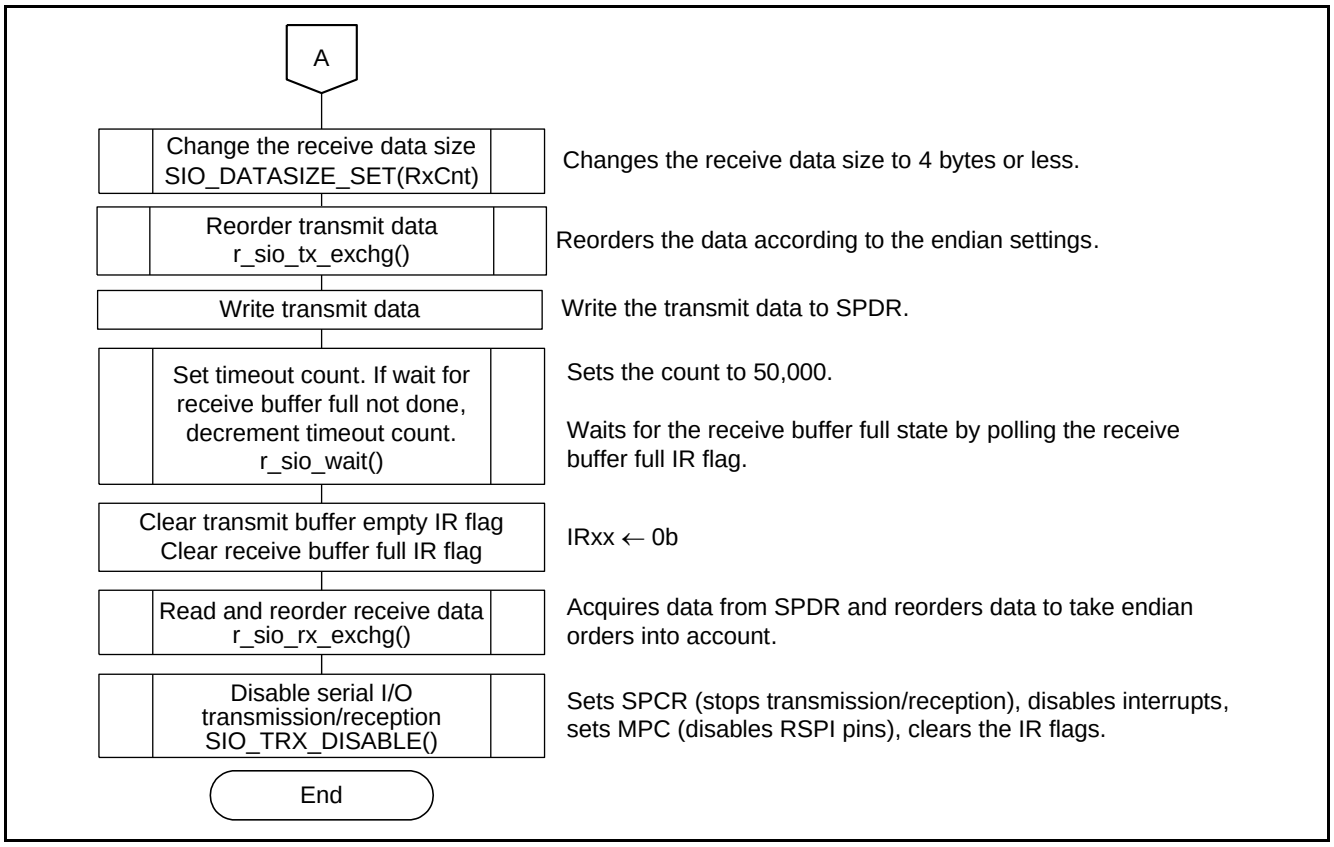


Figure 6.19 Serial I/O Data Transmission/Reception Processing Outline 4 (High-Speed)

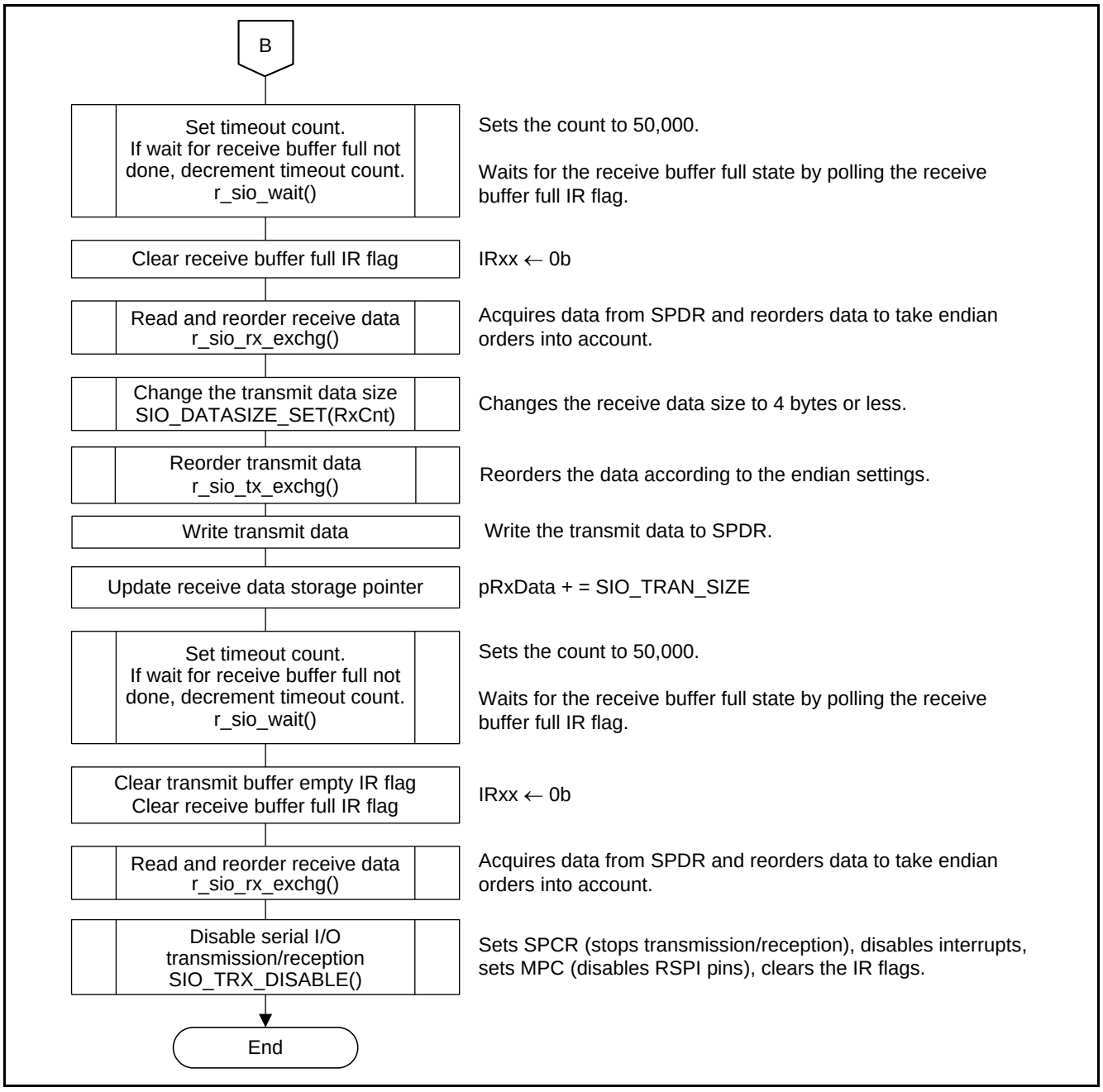


Figure 6.20 Serial I/O Data Transmission/Reception Processing Outline 5 (High-Speed)

6.10 Inline Function Specifications

This section describes the inline functions used in this sample code.

6.10.1 SIO_IO_INIT()

(1) Purpose

This function disables the RSPI functions for the corresponding pins, sets input pins to the port input state, and sets output pins to the port output state.

(2) Function

This function disables the RSPI functions for the corresponding pins, sets the DataIn pin to the port input state, and sets the DataOut and CLK pins to the port output state.

The following processing is implemented. If necessary, revise this processing.

1. Sets the pins to be used to their port function.
   See the SIO_MPC_DISABLE() function.
2. Sets the DataIn pin to port input.
   See the SIO_DATAI_INIT() function.
3. Sets the DataOut pin to port high output.
   See the SIO_DATAO_INIT() function.
4. Sets the DataOut pin to port high output.
   See the SIO_CLK_INIT() function.

(3) Remarks

This inline function changes the pins from their peripheral function to their port function. Applications should first verify that other peripheral functions are not being used before executing this function.

6.10.2 SIO_IO_OPEN()

(1) Purpose

Sets the input pins and output pins to the port input state.

(2) Function

Sets the DataIn pin, the DataOut pin, and the CLK pin to the port input state.

The following processing is implemented. If necessary, revise this processing.

1. Sets the DataIn pin to port input.
   See the SIO_DATAI_INIT() function.
2. Sets the DataOut pin to port input.
   See the SIO_DATAO_OPEN() function.
3. Sets the CLK pin to port input.
   See the SIO_CLK_OPEN() function.

(3) Remarks

Use this function to set all pins to high impedance before removable media is inserted or removed. Execute this function after executing SIO_IO_INIT().

6.10.3 SIO_DATAI_INIT()

(1) Purpose

Sets the DataIn pin to the port input state.

(2) Function

The following processing is implemented. If necessary, revise this processing.

1. Disables the DataIn pin input pull-up resistor with the pull-up resistor control register (PCR).
   — DataIn pin PCR ← 0b: Input pull-up resistor disabled
2. Sets the DataIn pin to port input using the port direction register (PDR).
   — DataIn pin PDR ← 0b: Input port

(3) Remarks

None

6.10.4 SIO_DATAO_INIT()

(1) Purpose

Sets the DataOut pin to port high output.

(2) Function

The following processing is implemented. If necessary, revise this processing.

1. Sets the DataOut pin output type to CMOS output using the open drain control register (ODRn).
   — DataOut pin ODR ← 0b: CMOS output
2. Sets the DataOut pin port drive capacity using the drive capacity control register (DSCR). We recommend the following settings*[1] according to the AC timing characteristics conditions of the microcontroller used.
   — Supplement: For the RX210, RX21A, RX63N, RX63T, RX64M, RX71M and RX634:
   Set the DataOut pin port drive capacity to “high drive output”.
   • DataOut pin DSCR ← 1b: High drive output
   — Supplement: For the RX220
   Set the DataOut pin port drive capacity to “normal drive output”.
   • DataOut pin DSCR ← 0b: Normal drive output*[2]
   — Supplement: For the RX111
   • Setting is not required because DSCR is not supported.
3. Sets the DataOut pin to high output using the port output data register (PODR).
   — DataOut pin PODR ← 1b: High output
4. Sets the DataOut pin to port output using the port direction register (PDR) and the port output data register (PODR).
   — DataOut pin PDR ← 1b: Output port
   — DataOut pin PODR ← 1b: High output

(3) Remarks

Notes: 1. The permissible output low current (I_{OL}) and the output low voltage (V_{OL}) characteristics for the normal output and high drive output differ depending on the microcontroller used. Set this item to an appropriate value for the connected output.
2. The pins that can be used with the drive capacity control register (DSCR) are limited.

6.10.5 SIO_DATAO_OPEN()

(1) Purpose

Sets the DataOut pin to the port input function.

(2) Function

The following processing is implemented. If necessary, revise this processing.

1. Sets the DataOut pin to port input using the port direction register (PDR).
   — DataOut pin PDR ← 0b: Input port (input buffer disabled)

(3) Remarks

None

6.10.6 SIO_CLK_INIT()

(1) Purpose

Sets the CLK pin to port high output.

(2) Function

The following processing is implemented. If necessary, revise this processing.

1. Sets the CLK pin output type to CMOS output using the open drain control register (ODRn).
   — CLK pin ODR ← 0b: CMOS output*1
2. Sets the CLK pin port drive capacity using the drive capacity control register (DSCR). We recommend the following settings*2 according to the AC timing characteristics conditions of the microcontroller used.
   — Supplement: For the RX210, RX21A, RX63N, RX63T, RX64M, RX71M and RX634:
   Set the CLK pin port drive capacity to “high drive output”.
   • CLK pin DSCR ← 1b: High drive output
   — Supplement: For the RX220
   Set the CLK pin port drive capacity to “normal drive output”.
   • CLK pin DSCR ← 0b: Normal drive output*3
   — Supplement: For the RX111
   • Setting is not required because DSCR is not supported.
3. Sets the CLK pin to high output using the port output data register (PODR).
   — CLK pin PODR ← 1b: High output
4. Sets the CLK pin to port output using the port direction register (PDR) and the port output data register (PODR).
   — CLK pin PDR ← 1b: Output port
   — CLK pin PODR ← 1b: High output

(3) Remarks

Notes: 1. For the RX63N, when setting open drain control register 0 (ODR0), it is necessary to set 2 bits (bits 2 and 3) for port PE1 only. Revise this setting if required.
2. The permissible output low current (I_{OL}) and the output low voltage (V_{OL}) characteristics for the normal output and high drive output differ depending on the microcontroller used. Set this item to an appropriate value for the connected output.
3. The pins that can be used with the drive capacity control register (DSCR) are limited.

6.10.7 SIO_CLK_OPEN()

(1) Purpose

Sets the CLK pin to the port input function.

(2) Function

The following processing is implemented. If necessary, revise this processing.

1. Sets the CLK pin to port input using the port direction register (PDR).
   — CLK pin PDR ← 0b: Input port

(3) Remarks

None

6.10.8 SIO_ENABLE()

(1) Purpose

Initializes serial I/O and enables its functions. Note that this function performs the common processing through enabling transmission or transmission/reception. It also sets the bit rate.

(2) Function

Initializes serial I/O as stipulated in the hardware manual. If necessary, revise this processing.
This function performs the following processing when an RX family microcontroller is used.

1. Sets the module to the module stop canceled state using the protect register (PRCR) and module stop control register (MSTPCRB).
   — PRCR ← A502h: Cancels protect of module stop control register.
   — MSTPCRB.MSTPBxx ← 0b: Cancels module stop and enables reading and writing of the RSPI registers.
   — Reads MSTPCRB.MSTPBxx.
   — PRCR ← A500h: Enables protect of module stop control register.
2. Performs the common processing for enabling transmission and transmission/reception.
   The common processing for enabling transmission and transmission/reception consists of the following operations.
   — SPPCR ← 30h: Sets up normal mode, CMOS output, and a MOSI idle fixed value of 1.
   — Sets the SPBR bit rate.
   — SPDCR ← 20h: Setting 1.1, SSL0 to SSL3 output, read receive buffer, longword access.
   — SPCKD ← 00h: SPCKD delay value setting (initial value)*[1]
   — SSLND ← 00h: SSL negation delay value setting (initial value)*[1]
   — SPND ← 00h: Next access delay value setting (initial value)*[1]
   — SPCR2 ← 00h: Parity function disabled, idle interrupt disabled.
   — Clears the SPSR OVFR, MODF, and PERF flags.
   See the SIO_SPSR_CLEAR() function.
   — SPCR ← 09h: three-wire method, master mode, transmit interrupts disabled, RSPI functions disabled, receive interrupts disabled.
   — SPSCR ← 00h: Sequence length: only SPCMD0 is used.

(3) Remarks

The user should insert wait processing after this inline function completes for serial I/O that requires a wait after setting the bit rate.

This function forms a pair with SIO_DISABLE(). If this function is run, call SIO_DISABLE() to terminate processing.

Call one of SIO_DISABLE(), SIO_TX_DISABLE(), or SIO_TRX_DISABLE() (to disable communication operation using SPCR) to stop communication operation before calling this function.

Note: 1. Not used in this sample code.

6.10.9 SIO_DISABLE()

(1) Purpose

Disables the serial I/O functions.

(2) Function

Disables the serial I/O functions. This function performs the common processing in the procedures for disabling transmission or transmission/reception. If necessary, revise this processing.

This function performs the following processing when an RX family microcontroller is used.

1. Sets the module to the module stop canceled state using the protect register (PRCR) and module stop control register (MSTPCRB) so that the RSPI related registers can be set.*1
   - PRCR ← A502h: Cancels protect of module stop control register.
   - MSTPCRB.MSTPBxx ← 0b: Cancels module stop and enables reading and writing of the RSPI registers.
   - Reads MSTPCRB.MSTPBxx.
2. Disables the RSPI functions.
   - SPCR ← 09h: three-wire method, master mode, transmit interrupts disabled, RSPI functions disabled, receive interrupts disabled.
3. Clears the SPSR OVFR, MODF, and PERF flags.
   See the SIO_SPSR_CLEAR() function.
4. Sets the module to the module stop state using the protect register (PRCR) and module stop control register (MSTRCRB).
   - MSTPCRB MSTPBxx ← 1b: Sets module stop state and disables reading or writing the RSPI registers. (The RSPI register states are retained.)
   - Reads MSTPCRB MSTPBxx.
   - PRCR ← A500h: Enables protect of module stop control register.

(3) Remarks

This function forms a pair with SIO_ENABLE(). If SIO_ENABLE() is run, call this function to terminate processing.

Note: 1. With RX family microcontrollers, registers for a module in the module stop state cannot be read or written. In this inline function, the module stop state is canceled temporarily to use SPCR to disable the RSPI functions. After setting SPCR, this function sets module stop state. Note that register values are retained while a module is in the module stop state.

6.10.10 SIO_DATASIZE_SET()

(1) Purpose

Sets SPB[3:0] in the SPCMD0 to SPCMD7 registers.

(2) Function

Sets the data length (8, 16, 24, or 32 bits).

(3) Remarks

None

6.10.11 SIO_TX_ENABLE()

(1) Purpose

Enables serial I/O transmission.

(2) Function

Enables serial I/O according to the specifications in the hardware manual. After switching the pins from their port functions to their serial I/O functions, it enables serial I/O. If necessary, revise this processing.

This function performs the processing from the initialization procedure following SIO_ENABLE() to the dedicated initialization processing for transmission.

The following processing is performed when an RX family microcontroller is used.

1. SPCR2 settings (Sets the RSPI idle interrupt request to enabled.)
   — SPCR2 ← 04h: Parity function disabled, idle interrupt enabled (for detection of end of transmission)
2. SPCMD settings
   — SPCMD0 ← 0203h: CPHA = 1, CPOL = 1, base bit rate, SSL0*[1], SSL signal negated on end of transfer*[2], 32 bits, MSB first*[3].
3. Clears the IR flags and the internally held interrupt request*[4].
   See the SIO_IR_CLEAR() function.
4. Sets the used pins to their RSPI function.
   See the SIO_MPC_ENABLE() function.
5. SPCR settings (Enables transmission)
   Enables transmission by setting TXMD, SPTIE, and SPE in the SPCR register.
   — SPCR ← 6Bh: three-wire method, transmission operation only, master mode, transmit interrupts enabled, RSPI functions enabled.
6. Reads SPCR.

(3) Remarks

This function forms a pair with SIO_TX_DISABLE(). If this function is run, call SIO_TX_DISABLE() to terminate processing.

Notes: 1. Since three-wire method is used, the SSL functions are not used. Since the SSL pins can be allocated to other functions, sets the SSL pins other than SSL0 to I/O and allocates them to the other functions.
2. Since the SSL functions are not used, this setting is ignored.
3. Since SPCKD, SSLND, and SPND are not used, bits b15 to b13 in the SPCMD0 register should be set to 0b.
4. Interrupt requests to the ICU are held internally and not output even when the interrupt generation condition for the SPTI interrupt and SPRI interrupt is that the value of the IR flag is 1. Unexpected behavior can result if communication starts while these internally held flags remain set to 1, so the internal flags are cleared.

6.10.12 SIO_TX_DISABLE()

(1) Purpose

Stops the serial I/O data transmission function.

(2) Function

This function stops the transmission function with the reverse procedure from that used by SIO_TX_ENABLE(). After performing the settings to stop transmission, it switches the pins from their serial I/O functions to their port functions. If necessary, revise this processing.

This function performs the following processing when an RX family microcontroller is used.

1. SPCR settings (Stops transmission and reception)
   Clears the TXMD, SPTIE, SPE, and SPRIE bits in the SPCR register to stop transmission and reception.
   — SPCR ← 09h: Master mode, transmit interrupts disabled, RSPI functions disabled, receive interrupts disabled.
2. Reads SPCR.
3. Disables the pin peripheral functions.
   See the SIO_MPC_DISABLE() function.
4. Clears the IR flags and the internally held interrupt requests.
   See the SIO_IR_CLEAR() function.
5. SPCR2 settings (Sets the RSPI idle interrupt request to disabled.)
   — SPCR2 ← 00h: Parity function disabled, idle interrupt disabled.

(3) Remarks

This function forms a pair with SIO_TX_ENABLE(). After SIO_TX_ENABLE() is run, call this function to terminate processing.

6.10.13 SIO_TRX_ENABLE()

(1) Purpose

Enables serial I/O transmission/reception.

(2) Function

Enables serial I/O according to the specifications in the hardware manual. After switching the pins from their port functions to their serial I/O functions, it enables serial I/O transmission/reception. If necessary, revise this processing.

This function performs the processing from the initialization procedure following SIO_ENABLE() to the dedicated initialization processing for transmission/reception.

The following processing is performed when an RX family microcontroller is used.

1. SPCMD settings
   — SPCMD0 ← 0203h: CPHA = 1, CPOL = 1, base bit rate, SSL0*1, SSL signal negated on end of transfer*2, 32 bits, MSB first*3.
2. Clears the IR flags and the internally held interrupt requests.*4
   See the SIO_IR_CLEAR() function.
3. Sets the used pins to their RSPI function.
   See the SIO_MPC_ENABLE() function.
4. SPCR settings (Enables transmission/reception)
   Enables transmission/reception by setting SPTIE, SPE and SPRIE in the SPCR register.
   — SPCR ← E9h: three-wire method, full-duplex operation, master mode, transmit and receive interrupts enabled, RSPI functions enabled.
5. Reads SPCR.

(3) Remarks

This function forms a pair with SIO_TRX_DISABLE(). If this function is run, call SIO_TRX_DISABLE() to terminate processing.

Notes: 1. Since three-wire method is used, the SSL functions are not used. Since the SSL pins can be allocated to other functions, sets the SSL pins to I/O and allocates them to the other functions.
2. Since the SSL functions are not used, this setting is ignored.
3. Since SPCKD, SSLND, and SPND are not used, bits b15 to b13 in the SPCMD0 register should be set to 0b.
4. Interrupt requests to the ICU are held internally and not output even when the interrupt generation condition for the SPTI interrupt and SPRI interrupt is that the value of the IR flag is 1. Unexpected behavior can result if communication starts while these internally held flags remain set to 1, so the internal flags are cleared.

6.10.14 SIO_TRX_DISABLE()

(1) Purpose

Stops the serial I/O data transmission/reception function.

(2) Function

This function stops the transmission/reception function with the reverse procedure from that used by SIO_TRX_ENABLE(). After performing the settings to stop transmission/reception, it switches the pins from their serial I/O functions to their port functions. If necessary, revise this processing.

This function performs the following processing when an RX family microcontroller is used.

1. SPCR settings (Stops transmission/reception)
   Clears the TXMD, SPTIE, SPE, and SPRIE bits in the SPCR register to stop transmission/reception.
   — SPCR ← 09h: Master mode, transmit interrupts disabled, RSPI functions disabled, receive interrupts disabled.
2. Reads SPCR.
3. Disables the pin peripheral functions.
   See the SIO_MPC_DISABLE() function.
4. Clears the IR flags and the internally held interrupt requests.
   See the SIO_IR_CLEAR() function.

(3) Remarks

This function forms a pair with SIO_TRX_ENABLE(). After SIO_TRX_ENABLE() is run, call this function to terminate processing.

6.10.15 SIO_SPSR_CLEAR()

(1) Purpose

Clears the SPSR error flags.

(2) Function

Clears the OVRF, MODF, and PERF flags.

1. If a flag is 1, it is cleared to 0.
2. The flag is then read to verify that it is 0.

(3) Remarks

None

6.10.16 SIO_IR_CLEAR()

(1) Purpose

Clears the IR flags and the internally held interrupt requests.*1

(2) Function

The procedure below is used to clear the flags according to “Points to Note on Starting Transfer” in the description of the RSPI in the hardware manual. If necessary, revise this processing.

The processing is as follows on RX family.

1. Confirms that SPE in SPCR is cleared to 0. If it is set to 1, clears SPE to 0.
2. Sets SPCR (disables interrupt requests).
   Clears SPTIE and SPRIE in SPCR to 0 to disable transmit and receive interrupt requests.
3. Reads SPTIE and SPRIE in SPCR to confirm that their value is 0.
4. Clears IR flags.

(3) Remarks

Note: 1. Interrupt requests to the ICU are held internally and not output even when the interrupt generation condition for the SPTI interrupt and SPRI interrupt is that the value of the IR flag is 1. Unexpected behavior can result if communication starts while these internally held flags remain set to 1, so the internal flags are cleared.

6.10.17 SIO_MPC_ENABLE()

(1) Purpose

Sets the pins used to their RSPI functions.

(2) Function

The procedure below is used to make register settings according to “Procedure for Specifying Input/Output Pin Function” in the description of the multi-function pin controller (MPC) in the hardware manual. If necessary, revise this processing.

1. Clears the appropriate bits in the port mode register (PMR) to 0 to set the pins to their general I/O function.
   — DataIn pin, DataOut pin, and CLK pin PMR ← 0b: Set as general I/O port.
2. Sets the write protect register (PWPR) to enable writing to the port pin function select registers (PxnPFS).
   — PWPR.B0WI ← 0b: Writing to PFSWE bit enabled.
   — PWPR.PFSWE ← 1b: Writing to PFS registers enabled.
3. Sets the RSPI pin functions using bits PxnPFS.PSEL[4:0].
   — DataIn pin PxnPFS ← 0Dh: Use as MISO pin enabled.*1
   — DataOut pin PxnPFS ← 0Dh: Use as MOSI pin enabled.*1
   — CLK pin PxnPFS ← 0Dh: Use as RSPCK pin enabled.*1
4. Clears the PFSWE bit in PWPR to 0 to disable writing to the PxnPFS registers.
   — PWPR.PFSWE ← 0b: Writing to PFS registers disabled.
   — PWPR.B0WI ← 1b: Writing to PFSWE bit disabled.
5. Sets PMR to 1 for each pin to switch to the RSPI pin function.
   — DataIn pin, DataOut pin, and CLK pin PMR ← 1b: Used as RSPI function.

(3) Remarks

Make settings to the port pin function select (PxnPFS) registers while the bits for the relevant pins in the PMR register are cleared to 0. Making settings to the PxnPFS registers while the bits for the relevant pins in the PMR register are set to 1 can cause unanticipated edge input, in the case of the input function, and unanticipated pulse output, in the case of the output function.

Note: 1. The setting values may differ depending on the microcontroller used. If necessary, revise these values.

6.10.18 SIO_MPC_DISABLE()

(1) Purpose

Sets the pins used to their port function.

(2) Function

The procedure below is used to make register settings according to “Procedure for Specifying Input/Output Pin Function” in the description of the multi-function pin controller (MPC) in the hardware manual. If necessary, revise this processing.

1. Clears the appropriate bits in the port mode register (PMR) to 0 to set the pins to their general I/O function.
   — DataIn pin, DataOut pin, and CLK pin PMR ← 0b: Set as general I/O port.
2. Sets the write protect register (PWPR) to enable writing to the port pin function select registers (PxnPFS).
   — PWPR.B0WI ← 0b: Writing to PFSWE bit enabled.
   — PWPR.PFSWE ← 1b: Writing to PFS registers enabled.
3. Sets the port pin I/O function using bits PxnPFS.PSEL[4:0].
   — DataIn pin, DataOut pin, and CLK pin PxnPFS ← 00h: Hi-z (initial value)
4. Clears the PFSWE bit in PWPR to 0 to disable writing to the PxnPFS registers.
   — PWPR.PFSWE ← 0b: Writing to PFS registers disabled.
   — PWPR.B0WI ← 1b: Writing to PFSWE bit disabled.

(3) Remarks

Make settings to the port pin function select (PxnPFS) registers while the bits for the relevant pins in the PMR register are cleared to 0. Making settings to the PxnPFS registers while the bits for the relevant pins in the PMR register are set to 1 can cause unanticipated edge input, in the case of the input function, and unanticipated pulse output, in the case of the output function.

7. Sample Application

This section presents a sample application that sets up the serial I/O control block.

The sample settings for actual usage are shown below.

The places in each file that need to be set are marked with the comment "/** SET **/".

7.1 mtl_com.h (Common header file)

Common header file for common functions.

Files (except for mtl_com.h.common) with the filename mtl_com.h.XXX have been created for each microcontroller. Rename one of these to mtl_com.h and use that file. If there is no corresponding file for the microcontroller used, refer to these files and create a file appropriate for the microcontroller used.

(1) OS Header File Definitions

This sample code does not use any settings for OS system calls.

The example below is for the case where no OS is used.

Set these up to be unused settings with this sample code. They depend on other software.

```
/* To use system calls,                                                    */
/* include the OS header files with the prototype declaration.             */
/* If no OS is used, comment out the following define and includes.        */
//#define MTL_OS_USE                    /* Use OS                          */
//#include <RTOS.h>                     /* OS header file                  */
//#include "mtl_os.h"
```

(2) Header File Definitions that Define the Common Access areas

A header file in which the MCU function registers are defined is included.

This file is mainly used by device drivers for port control and must be included.

Include the header file that matches the microcontroller used.

In the example below, the header file for an RX family microcontroller is included.

This header file must be included when this sample code is used.

```
/* To use the SFR area define values for the microcontroller,              */
/* include the header file the has the I/O peripheral definitions.         */
#include "iodefine.h"                     /* definition of MCU SFR         */
```

(3) Loop Timer Definitions

Include the following header file if the software loop timer is used.

This file is mainly used for device drivers to provide wait times.

Comment out the following include statement if the software loop timer is not used.

The example shown below is for the case where the software loop timer is used.

This header file must be included when this sample code is used.

```
/* Comment out the following include statement if the software loop timer is not used.   */
#include "mtl_tim.h"
```

(4) Endian Order Definition

Either little endian or big endian may be specified.

The example below shows how big endian is specified.

```
/* Specify little endian for (1) SuperH or (2) M16C microcontrollers by enabling this definition.  */
/* For other microcontrollers, comment out the little endian definition.     */
//#define MTL_MCU_LITTLE                    /* Little Endian                  */
```

(5) Definition for Fast Endian Processing

High-speed processing can be specified for mtl_end.c. If an M16C microcontroller is used, this will speed up processing.

For RX family microcontrollers, comment out this definition so that the symbol is not defined.

```
/* Enable this definition if an M16C microcontroller is used.                */
/* High-speed processing can be specified for mtl_end.c.                     */
//#define MTL_ENDI_HISPEED              /* Uses the high-speed function.      */
```

(6) Standard Library Type Definition

The type of standard library used must be defined.

If the library included with the compiler will be used for the processing shown below, comment out the following definition.

The example shown below is for the case where the library included with the compiler is used.

```
/* Specify the type of standard library used.                                */
/* If the library included with the compiler will be used for the processing shown below,*/
/* comment out the following definition.                                     */
/* memcmp() / memmove() / memcpy() / memset() / strcat() / strcmp() / strcpy() / strlen()*/
//#define MTL_USER_LIB                    /* use optimized library          */
```

(7) Definition of the RAM Area to be Accessed

The RAM area used must be defined.

Highly efficient processing can be applied to standard functions and certain other operations.

For RX family microcontrollers, MTL_MEM_NEAR should be defined.

```
/* The processing group used and the RAM area used must be defined.          */
/* Highly efficient processing can be applied to standard functions and certain other operations.  */
//#define MTL_MEM_FAR          /* Supports Far RAM area of M16C/60            */
#define MTL_MEM_NEAR           /* Supports Near RAM area.       (Others)      */
```

7.1.2 mtl_tim.h

This file is included if the loop timer is defined in mtl_com.h.

This file depends on the microcontroller used, the clock, the compiler options, and other items.

In systems in which the instruction cache is enabled, the loop timer should be set up assuming that it is running from the instruction cache.

Measure the loop timer performance and set it up according to the operating environment used.

Sample settings for the RX Family microcontrollers are shown below.

```
/* The timer counter value must be defined.                                  */
/* Set up the timer according to the microcontroller and clock used.         */
#if 1
/* Setting for 12 MHz no wait Ix16/2 = 96 MHz(Compile Option "-optimize=2",com.V406R00)
  */
#define MTL_T_1US                          10    /* loop Number of   1us     */
#define MTL_T_2US                          20    /* loop Number of   2us     */
#define MTL_T_4US                          40    /* loop Number of   4us     */
#define MTL_T_5US                          50    /* loop Number of   5us     */
#define MTL_T_10US                        100    /* loop Number of  10us     */
#define MTL_T_20US                        200    /* loop Number of  20us     */
#define MTL_T_30US                        300    /* loop Number of  30us     */
#define MTL_T_50US                        500    /* loop Number of  50us     */
#define MTL_T_100US                      1000    /* loop Number of 100us     */
#define MTL_T_200US                      2000    /* loop Number of 200us     */
#define MTL_T_300US                      3000    /* loop Number of 300us     */
#define MTL_T_400US      ( MTL_T_200US   *  2 )   /* loop Number of 400us     */
#define MTL_T_1MS                       10000    /* loop Number of   1ms     */
#endif
```

Note that the values above have not been measured and thus appropriate values have not been determined. Through testing should be performed to determine these values.

7.2 Settings for the Clock Synchronous Single Master Control Software

The places in each file that need to be set are marked with the comment "/** SET **/".

7.2.1 R_SIO.h

(1) Definition of the Wait Following the BRR Setting

After the RSPI SPBR register is set, the application waits in software for the period to transfer 1 bit. This wait time must be set.

A time of 10 µs is set as an initial value.

When a MultiMediaCard is used, a value of 10 µs should be set assuming a communications rate of 100 kHz.

```
#define SIO_T_BRR_WAIT        (uint16_t)MTL_T_10US /* BRR setting wait time */
```

7.2.2 R_SIO_rspi.h

This is the definitions file for the RSPI module.

Files with the filename R_SIO_rspi.h.XXX have been created for each microcontroller. Rename one of these to R_SIO_rspi.h and use that file. If there is no corresponding file for the microcontroller used, refer to these files and create a file appropriate for the microcontroller used.

(1) Operating Mode Definitions

The resources for the microcontroller used can be set up. Select the one required definition. In the example below, SIO_OPTION_4 has been selected. Table 7.1 lists operating modes and their functions.

```
/*------------------------------------------------------------------------- */
/*  Define the combination of the MCU's resources.                          */
/*------------------------------------------------------------------------- */
//#define SIO_OPTION_1
//#define SIO_OPTION_2
//#define SIO_OPTION_3
#define SIO_OPTION_4
//#define SIO_OPTION_5
//#define SIO_OPTION_6
//#define SIO_OPTION_7
//#define SIO_OPTION_8
//#define SIO_OPTION_9
```

Table 7.1 Operating Modes

#define Definition	Operating Mode				
	SI/O (RSPI)	CRC Calculation (on-chip functional unit of microcontroller)	CRC Calculation (using software)	Interrupt disable period	Receive Mode Transmit/Receive Mode
SIO_OPTION_1	○	—	—	No	Normal receive mode
SIO_OPTION_2	○	○	—	No	Normal receive mode
SIO_OPTION_3	○	—	○	No	Normal receive mode
SIO_OPTION_4	○	—	—	Yes	High-speed receive mode
SIO_OPTION_5	○	○	—	Yes	High-speed receive mode
SIO_OPTION_6	○	—	○	Yes	High-speed receive mode
SIO_OPTION_7	○	—	—	No	High-speed receive mode
SIO_OPTION_8	○	○	—	No	High-speed receive mode
SIO_OPTION_9	○	—	○	No	High-speed receive mode

When one of SIO_OPTION_1 to SIO_OPTION_3 is selected, normal receive mode or normal transmit/receive mode is used. In normal receive mode or normal transmit/receive mode the next data receive operation is performed only after full data reception has been verified and the data output. Therefore, overrun errors do not occur and reliable reception is possible. This mode is designed to avoid software processing during data reception as much as possible. For example, the endian conversion processing for data during continuous reception is performed only after the dummy write of the following data.

When one of SIO_OPTION_4 to SIO_OPTION_9 is selected, high-speed receive mode or high-speed transmit/receive mode is used. In high-speed receive mode or high-speed transmit/receive mode, writing the dummy data for the next data reception is performed during the current reception, which allows the next data reception operation to be performed immediately after the received data is acquired. However, note that a period in which interrupts are disabled occurs during continuous data reception or transmission/reception in case of SIO_OPTION_4 to SIO_OPTION_9. During this period an overrun error may occur if this application does not acquire the received data in time due to any of the following conditions:

- [Supplement] If contention for the bus occurs between this reception and a DMAC, EXDMAC, or DTC transfer by another application
- [Supplement] If a high-priority NMI interrupt occurs
- [Supplement] If the system clock is set to low-speed operation

Select one of SIO_OPTION_1 to SIO_OPTION_3 or SIO_OPTION_7 to SIO_OPTION_9 to avoid this problem. Note that it is possible to select in case of RSPI module with RSPCK auto-stop function.

If the microcontroller's internal CRC unit is used to perform MSB-first CRC CCITT calculations, select one of SIO_OPTION_2, SIO_OPTION_5 or SIO_OPTION_8.

If software processing is used to perform MSB-first CRC CCITT calculations, select one of SIO_OPTION_3, SIO_OPTION_6 or SIO_OPTION_9.

(2) CRC Calculation Type Definition

The CRC calculation type must be specified.

If either serial EEPROM or serial flash memory is controlled, comment these settings so that no CRC CCITT calculation is used.

Both of these must be defined if a MultiMediaCard is used.

```
/*------------------------------------------------------------------------*/
/*  Define the CRC calculation.                                           */
/*------------------------------------------------------------------------*/
#define SIO_CRCCCITT_USED              /* CRC-CCITT used                  */
#define SIO_CRC7_USED                  /* CRC7 used                       */
```

(3) Used RSPI Channel Definition

The RSPI channel used must be defined.

```
/*------------------------------------------------------------------------*/
/*  Define the RSPI channel.                                              */
/*------------------------------------------------------------------------*/
#define SIO_RSPI_CHANNEL   0              /* RSPI Channel Select          */
```

(4) Used Pin Definitions

The definitions of the serial pins used are shown below. Specify the pin numbers for the used pins by referring to table 7.2, Used Pin Definitions.

- For the RX210, RX220, RX63N and RX634

```
/*------------------------------------------------------------------------*/
/*  Define the control port.                                              */
/*------------------------------------------------------------------------*/
/* Set to use port numbers and bit numbers */
#define SIO_DATAI_PORTNO      A     /* SIO DataIn Port No.                */
#define SIO_DATAI_BITNO       7     /* SIO DataIn Bit No.                 */
#define SIO_CLK_PORTNO        A     /* SIO CLK Port No.                   */
#define SIO_CLK_BITNO         5     /* SIO CLK Bit No.                    */
#define SIO_CLK_REGNO         1     /* SIO CLK ODR Register No(Set '0'or'1') */
#define SIO_CLK_ODRBITNO      2     /* SIO CLK ODR bit No.                */
#define SIO_DATAO_PORTNO      A     /* SIO DataOut Port No.               */
#define SIO_DATAO_BITNO       6     /* SIO DataOut Bit No.                */
#define SIO_DATAO_REGNO       1     /* SIO DataOut ODR Register No(Set '0'or'1') */
#define SIO_DATAO_ODRBITNO    4     /* SIO DataOut ODR bit No.            */
```

- For the RX21A

```
/*------------------------------------------------------------------------*/
/*  Define the control port.                                              */
/*------------------------------------------------------------------------*/
/* Set to use port numbers and bit numbers */
#define SIO_DATAI_PORTNO      C     /* SIO DataIn Port No.                */
#define SIO_DATAI_BITNO       7     /* SIO DataIn Bit No.                 */
#define SIO_CLK_PORTNO        C     /* SIO CLK Port No.                   */
#define SIO_CLK_BITNO         5     /* SIO CLK Bit No.                    */
#define SIO_CLK_REGNO         1     /* SIO CLK ODR Register No(Set '0'or'1') */
#define SIO_CLK_ODRBITNO      2     /* SIO CLK ODR bit No.                */
#define SIO_DATAO_PORTNO      C     /* SIO DataOut Port No.               */
#define SIO_DATAO_BITNO       6     /* SIO DataOut Bit No.                */
#define SIO_DATAO_REGNO       1     /* SIO DataOut ODR Register No(Set '0'or'1') */
#define SIO_DATAO_ODRBITNO    4     /* SIO DataOut ODR bit No.            */
```

- For the RX63T

```
/*----------------------------------------------------------------------*/
/*  Define the control port.                                            */
/*----------------------------------------------------------------------*/
/* Set to use port numbers and bit numbers */
#define SIO_DATAI_PORTNO      A     /* SIO DataIn Port No.                */
#define SIO_DATAI_BITNO       5     /* SIO DataIn Bit No.                 */
#define SIO_CLK_PORTNO        A     /* SIO CLK Port No.                   */
#define SIO_CLK_BITNO         4     /* SIO CLK Bit  No.                   */
#define SIO_CLK_REGNO         1     /* SIO CLK ODR Register No(Set '0'or'1') */
#define SIO_CLK_ODRBITNO      0     /* SIO CLK ODR bit No.                */
#define SIO_CLK_DSCR2BITNO    7     /* SIO CLK DSCR2 bit No.              */
#define SIO_DATAO_PORTNO      B     /* SIO DataOut Port No.               */
#define SIO_DATAO_BITNO       0     /* SIO DataOut Bit No.                */
#define SIO_DATAO_REGNO             /* SIO DataOut ODR Register No(Set '0'or'1') */
#define SIO_DATAO_ODRBITNO          /* SIO DataOut ODR bit No.            */
#define SIO_DATAO_DSCR2BITNO  7     /* SIO DataOut DSCR2 bit No.          */
```

- For the RX111

```
/*----------------------------------------------------------------------*/
/*  Define the control port.                                            */
/*----------------------------------------------------------------------*/
/* Set to use port numbers and bit numbers */
#define SIO_DATAI_PORTNO      1     /* SIO DataIn Port No.                */
#define SIO_DATAI_BITNO       7     /* SIO DataIn Bit No.                 */
#define SIO_CLK_PORTNO        1     /* SIO CLK Port No.                   */
#define SIO_CLK_BITNO         5     /* SIO CLK Bit No.                    */
#define SIO_CLK_REGNO         1     /* SIO CLK ODR Register No(Set '0'or'1') */
#define SIO_CLK_ODRBITNO      2     /* SIO CLK ODR bit No.                */
#define SIO_DATAO_PORTNO      1     /* SIO DataOut Port No.               */
#define SIO_DATAO_BITNO       6     /* SIO DataOut Bit No.                */
#define SIO_DATAO_REGNO       1     /* SIO DataOut ODR Register No(Set '0'or'1') */
#define SIO_DATAO_ODRBITNO    4     /* SIO DataOut ODR bit No.            */
```

- For the RX64M and RX71M

```
/*----------------------------------------------------------------------*/
/*  Define the control port.                                            */
/*----------------------------------------------------------------------*/
/* Set to use port numbers and bit numbers */
#define SIO_DATAI_PORTNO      C     /* SIO DataIn Port No.                */
#define SIO_DATAI_BITNO       7     /* SIO DataIn Bit No.                 */
#define SIO_CLK_PORTNO        C     /* SIO CLK Port No.                   */
#define SIO_CLK_BITNO         5     /* SIO CLK Bit No.                    */
#define SIO_CLK_REGNO         1     /* SIO CLK ODR Register No(Set '0'or'1') */
#define SIO_CLK_ODRBITNO      2     /* SIO CLK ODR bit No.                */
#define SIO_DATAO_PORTNO      C     /* SIO DataOut Port No.               */
#define SIO_DATAO_BITNO       6     /* SIO DataOut Bit No.                */
#define SIO_DATAO_REGNO       1     /* SIO DataOut ODR Register No(Set '0'or'1')*/
#define SIO_DATAO_ODRBITNO    4     /* SIO DataOut ODR bit No.            */
```

Table 7.2 Used Pin Definitions

#define Definition	Set Value
SIO_DATAI_PORTNO	DataIn pin port number
SIO_DATAI_BITNO	DataIn pin bit number
SIO_CLK_PORTNO	CLK pin port number
SIO_CLK_BITNO	CLK pin bit number
SIO_CLK_REGNO	CLK pin open drain control register setting “x”: ODRx (x = 0 or 1)
SIO_CLK_ODRBITNO	CLK pin open drain control register bit number
SIO_CLK_DSCR2BITNO	CLK pin drive capacity control register 2 bit number*1
SIO_DATAO_PORTNO	DataOut pin port number
SIO_DATAO_BITNO	DataOut pin bit number
SIO_DATAO_REGNO	DataOut pin open drain control register setting “x”: ODRx (x = 0 or 1)
SIO_DATAO_ODRBITNO	DataOut pin open drain control register bit number
SIO_DATAO_DSCR2BITNO	DataOut pin drive capacity control register 2 bit number*1

Note: 1. This setting is only for the RX63T.

(5) Definition of Multi-Function Pin Controller (MPC) Use

Set the Pxn pin function control (PxnPFS) registers to match the serial pins to be used.

Sample settings for the RX Family microcontrollers are shown below.

```
/* Set to use Multi-Function Pin Controller */
#define SIO_MPCDATAO_ENABLE   (uint8_t)(0x0D)   /* Setting for RSPI MOSIA   */
      /* 00001101B*/ /* Port Pin Function Control Register RSPI pin setting */
      /* |||++++---------- Pin Function Select  : MOSIA                     */
      /* ||+--------------- Reserved            : Sets 0.                   */
      /* |+---------------- Interrupt Input Function Select  : Sets 0.       */
      /* +----------------- Analog Function Select/Reserved  : Sets 0.       */

#define SIO_MPCDATAI_ENABLE   (uint8_t)(0x0D)   /* Setting for RPSI MISOA   */
      /* 00001101B*/ /* Port Pin Function Control Register RSPI pin setting */
      /* |||++++---------- Pin Function Select  : MISOA                     */
      /* ||+--------------- Reserved            : Sets 0.                   */
      /* |+---------------- Interrupt Input Function Select  : Sets 0.       */
      /* +----------------- Analog Function Select/Reserved  : Sets 0.       */

#define SIO_MPCCLK_ENABLE     (uint8_t)(0x0D)   /* Setting for RPSI RSPCKA  */
      /* 00001101B*/ /* Port Pin Function Control Register RSPI pin setting */
      /* |||++++---------- Pin Function Select  : RSPCKA                    */
      /* ||+--------------- Reserved            : Sets 0.                   */
      /* |+---------------- Interrupt Input Function Select  : Sets 0.       */
```

(6) Mask Level Definition

Processing with interrupts disabled occurs during high-speed receive mode operation. To disable interrupts, specify the mask level with the highest priority. Note that this depends on the microcontroller used.

Set this to 15 for RX Family microcontrollers.

```
/*------------------------------------------------------------------------*/
/*  Define the interrupt mask level.                                      */
/*  Set interrupt mask level due to protect an overrun error by interrupt.*/
/*------------------------------------------------------------------------*/
#define SIO_INT_MASK_LEVEL    (uint8_t)(15)     /* Interrupt Mask Level     */
```

(7) Software Timer Definition

Set up the software timer that is used only by this sample code.

Set a value of 0.1 µs or larger as the initial value.

```
/*------------------------------------------------------------------------*/
/*  Define the wait time for timeout.                                     */
/*  Time out is occurred after 50000 times loop process of wait time.     */
/*------------------------------------------------------------------------*/
#define SIO_T_RSPI_WAIT      (uint16_t)(1)   /* 0.1us wait When CPU clock = 96MHz */
```

(8) Open Drain Control Register (ODR) Definitions

The inline functions SIO_DATAO_INIT() and SIO_CLK_INIT() can define an ODR.

With the RX63T, if an ODR definition for a used pin is possible, remove the comment from the ODR definition. If a pin for which an ODR cannot be allocated is used, a compiler error will occur if this definition is enabled.

For the RX63N, RX64M, RX111 and RX634, if open drain control register 0 (ODR0) is set by the inline function SIO_CLK_INIT(), it is necessary to set 2 bits (bits 2 and 3) for port PE1 only. Revise this setting if required.

For all other microcontrollers, since an ODR setting is possible for all the used RSPI pins, these statements should be enabled and the value set to 0 (CMOS output).

See section 8.10 for details on the RSPI pin port functions for each microcontroller.

Sample settings for the case where pins other than port PE1 are used and the definitions are enabled are shown below.

```
/*------------------- DataOut control ---------------------*/
#pragma inline(SIO_DATAO_INIT)
static void SIO_DATAO_INIT(void)       /* DataOut Initial Setting          */
{
  SIO_ODR_DATAO    = 0;                /* Open Drain Control Register : CMOS*/

/*-------------------- CLK control ---------------------*/
#pragma inline(SIO_CLK_INIT)
static void SIO_CLK_INIT(void)         /* CLK Initial Setting              */
{
  SIO_ODR_CLK      = 0;                /* Open Drain Control Register : CMOS*/
```

(9) Drive Capacity Control Register (DSCR) Settings

The inline functions SIO_DATAO_INIT() and SIO_CLK_INIT() can define a DSCR.

With the RX63N and RX220, if a DSCR setting is possible for a used pin, remove the comment from the DSCR definition. If a pin for which the DSCR setting is fixed is used, a compiler error will occur if this definition is enabled.

For the RX210, RX21A, RX63N, RX63T, RX64M, RX71M and RX634, we recommend the value 1 (high drive output).

For the RX220, we recommend the value 0 (normal output).

For the RX111, The DSCR is not supported.

See section 8.10 for details on the RSPI pin port functions for each microcontroller.

The recommended settings are shown below.

- For the RX210, RX21A, RX63N, RX64M, RX71M and RX634 (high drive output)

```
/*----------------- DataOut control ------------------*/
#pragma inline(SIO_DATAO_INIT)
static void SIO_DATAO_INIT(void)       /* DataOut Initial Setting           */
{
  SIO_DSCR_DATAO   = 1;     /* Drive Capacity Control :  High-drive output    */

/*------------------- CLK control --------------------*/
#pragma inline(SIO_CLK_INIT)
static void SIO_CLK_INIT(void)         /* CLK Initial Setting               */
{
  SIO_DSCR_CLK     = 1;     /* Drive Capacity Control : High-drive output     */
```

- For the RX63T (high drive output)

```
/*----------------- DataOut control ------------------*/
#pragma inline(SIO_DATAO_INIT)
static void SIO_DATAO_INIT(void)       /* DataOut Initial Setting           */
{
  SIO_DSCR2_DATAO  = 1;     /* Drive Capacity Control :  High-drive output    */

/*------------------- CLK control --------------------*/
#pragma inline(SIO_CLK_INIT)
static void SIO_CLK_INIT(void)         /* CLK Initial Setting               */
{
  SIO_DSCR2_CLK    = 1;     /* Drive Capacity Control : High-drive output     */
```

- For the RX220 (normal output)

```
/*----------------- DataOut control ------------------*/
#pragma inline(SIO_DATAO_INIT)
static void SIO_DATAO_INIT(void)       /* DataOut Initial Setting           */
{
  SIO_DSCR_DATAO   = 0;     /* Drive Capacity Control :  Normal-drive output */

/*------------------- CLK control --------------------*/
#pragma inline(SIO_CLK_INIT)
static void SIO_CLK_INIT(void)         /* CLK Initial Setting               */
{
  SIO_DSCR_CLK     = 0;     /* Drive Capacity Control : Normal-drive output  */
```

8. Usage Notes

8.1 Notes on Embedding

When embedding this sample code in an application, include the files R_SIO.h and R_SIO_rspi.h (the renamed R_SIO_rspi.h.XXX).

8.2 Unused Functions

We recommend commenting out unused functions so that they do not consume ROM capacity unnecessarily.

8.3 Using a Different Microcontroller

Other microcontrollers can be handled easily.

Only the following two files need to be provided.

- A common I/O module definitions file corresponding to R_SIO_rspi.h.XXX
- A header definitions file corresponding to mtl_com.h.XXX.

Create these files based on the provided samples.

8.4 CRC Calculator Unit Stop Setting (option)

While functions that use the CRC calculator unit cancel the module stop state in initialization, there is no function that sets this module stop state. If it is necessary to set up the module stop state, the user must implement code that performs this control.

8.5 Compiler Options

Operation has been verified with optimization level set to 2 and optimization method set to "prioritize size".

Operation has not been verified with optimization level set to 2 and optimization method set to "prioritize speed".

8.6 When Other Applications Use DMAC, EXDMAC, or DTC Transfers

When one of SIO_OPTION_4 to SIO_OPTION_6 is selected as the operating mode and contention for the bus that this sample code uses or a high-priority NMI interrupt occurs when another application uses DMAC, EXDMAC, or DTC transfers, overrun errors may occur due to receive data not being acquired in time.

To avoid this problem, select one of SIO_OPTION_1 to SIO_OPTION_3 as the operating mode.

8.7 System Clock

When SIO_OPTION_4 to SIO_OPTION_6 is selected as the operating mode and the system clock is set to low-speed operation, an overrun error may occur if this application does not acquire the received data in time due to the slowness of CPU processing.

Select one of SIO_OPTION_1 to SIO_OPTION_3 or SIO_OPTION_7 to SIO_OPTION_9as the operating mode definition to avoid this problem.

8.8 Open Drain Control Register 0 (ODR0) Settings when Using Port PE1

For the RX63N, RX64M, RX111 and RX634, when setting open drain control register 0 (ODR0), it is necessary to set 2 bits (bits 2 and 3) for port PE1 only. Revise this setting if required.

8.9 Notes on Drive Capacity Control Register (DSCR) Settings

The permissible output low current (I_{OL}) and the output low voltage (V_{OL}) characteristics for the normal output and high drive output differ depending on the microcontroller used. Set the drive capacity to an appropriate value for the connected output.

8.10 RSPI Pin Port Functions for Each Microcontroller

The open drain control register (ODR) and drive capacity control register (DSCR) control methods differ depending on the microcontroller used. Table 8.1 lists the port functions for the RSPI pins for each microcontroller.

Table 8.1 RSPI Pin Functions for Each Microcontroller (1)

Microcontroller	Channel	Pin	Port	Open Drain Control Register (ODR)	Drive Capacity Control Register (DSCR)
RX63N	RSPI0	RSPCKA	PA5	CMOS/Open drain	Normal/high drive output
			PB0	CMOS/Open drain	Normal/high drive output
			PC5	CMOS/Open drain	Normal/high drive output
		MOSIA	P16	CMOS/Open drain	High drive output (fixed)
			PA6	CMOS/Open drain	Normal/high drive output
			PC6	CMOS/Open drain	Normal/high drive output
		MISOA	P17	CMOS/Open drain	High drive output (fixed)
			PA7	CMOS/Open drain	Normal/high drive output
			PC7	CMOS/Open drain	Normal/high drive output
	RSPI1	RSPCKB	P27	CMOS/Open drain	High drive output (fixed)
			PE1*1	CMOS/Open drain	Normal/high drive output
			PE5	CMOS/Open drain	Normal/high drive output
		MOSIB	P26	CMOS/Open drain	High drive output (fixed)
			PE2	CMOS/Open drain	Normal/high drive output
			PE6	CMOS/Open drain	Normal/high drive output
		MISOB	P30	CMOS/Open drain	High drive output (fixed)
			PE3	CMOS/Open drain	Normal/high drive output
			PE7	CMOS/Open drain	Normal/high drive output
	RSPI2	RSPCKC	PD3	CMOS/Open drain	Normal/high drive output
		MOSIC	PD1	CMOS/Open drain	Normal/high drive output
		MISOC	PC2	CMOS/Open drain	Normal/high drive output

Note: 1. When setting open drain control register 0 (ODR0), it is necessary to set 2 bits (bits 2 and 3) for port PE1 only.

Table 8.2 RSPI Pin Functions for Each Microcontroller (2)

Microcontroller	Channel	Pin	Port	Open Drain Control Register (ODR)	Drive Capacity Control Register (DSCR)
RX63T	RSPI0/ RSPI1	RSPCKA/ RSPCKB	P24	No register definition	Normal/high drive output
			PA4	CMOS/Open drain	Normal/high drive output
			PD0	No register definition	Normal/high drive output
		MOSIA/ MOSIB	P23	CMOS/Open drain	Normal/high drive output
			PB0	No register definition	Normal/high drive output
			PD2	No register definition	Normal/high drive output
		MISOA/ MISOB	P22	CMOS/Open drain	Normal/high drive output
			PA5	CMOS/Open drain	Normal/high drive output
			PD1	No register definition	Normal/high drive output
RX210	RSPI0	RSPCKA	PA5	CMOS/Open drain	Normal/high drive output
			PB0	CMOS/Open drain	Normal/high drive output
			PC5	CMOS/Open drain	Normal/high drive output
		MOSIA	P16	CMOS/Open drain	Normal/high drive output
			PA6	CMOS/Open drain	Normal/high drive output
			PC6	CMOS/Open drain	Normal/high drive output
		MISOA	P17	CMOS/Open drain	Normal/high drive output
			PA7	CMOS/Open drain	Normal/high drive output
			PC7	CMOS/Open drain	Normal/high drive output
RX21A	RSPI0	RSPCKA	PA5	CMOS/Open drain	Normal/high drive output
			PB0	CMOS/Open drain	Normal/high drive output
			PC5	CMOS/Open drain	Normal/high drive output
		MOSIA	P16	CMOS/Open drain	Normal/high drive output
			PA6	CMOS/Open drain	Normal/high drive output
			PC6	CMOS/Open drain	Normal/high drive output
		MISOA	P17	CMOS/Open drain	Normal/high drive output
			PA7	CMOS/Open drain	Normal/high drive output
			PC7	CMOS/Open drain	Normal/high drive output
	RSPI1	RSPCKB	P27	CMOS/Open drain	Normal/high drive output
		MOSIB	P26	CMOS/Open drain	Normal/high drive output
			PE6	CMOS/Open drain	Normal/high drive output
		MISOB	P30	CMOS/Open drain	Normal/high drive output
			PE7	CMOS/Open drain	Normal/high drive output
RX220	RSPI0	RSPCKA	PA5	CMOS/Open drain	Normal drive output (fixed)
			PB0	CMOS/Open drain	Normal/high drive output
			PC5	CMOS/Open drain	Normal/high drive output
		MOSIA	P16	CMOS/Open drain	Normal/high drive output
			PA6	CMOS/Open drain	Normal drive output (fixed)
			PC6	CMOS/Open drain	Normal/high drive output
		MISOA	P17	CMOS/Open drain	Normal/high drive output
			PA7	CMOS/Open drain	Normal drive output (fixed)
			PC7	CMOS/Open drain	Normal/high drive output

Table 8.3 RSPI Pin Functions for Each Microcontroller (3)

Microcontroller	Channel	Pin	Port	Open Drain Control Register (ODR)	Drive Capacity Control Register (DSCR)
RX111	RSPI0	RSPCKA	PA5	CMOS/Open drain	No register definition
			PB0	CMOS/Open drain	No register definition
			PC5	CMOS/Open drain	No register definition
			PE3	CMOS/Open drain	No register definition
		MOSIA	P16	CMOS/Open drain	No register definition
			PA6	CMOS/Open drain	No register definition
			PE4	CMOS/Open drain	No register definition
			PC6	CMOS/Open drain	No register definition
		MISOA	P17	CMOS/Open drain	No register definition
			PC7	CMOS/Open drain	No register definition
			PA3	CMOS/Open drain	No register definition
RX64M	RSPI0	RSPCKA	PA5	CMOS/Open drain	Normal/high drive output
			PC5	CMOS/Open drain	Normal/high drive output
		MOSIA	PA6	CMOS/Open drain	Normal/high drive output
			PC6	CMOS/Open drain	Normal/high drive output
		MISOA	PA7	CMOS/Open drain	Normal/high drive output
			PC7	CMOS/Open drain	Normal/high drive output
RX71M	RSPI0	RSPCKA	PA5	CMOS/Open drain	Normal/high drive output
			PC5	CMOS/Open drain	Normal/high drive output
		MOSIA	PA6	CMOS/Open drain	Normal/high drive output
			PC6	CMOS/Open drain	Normal/high drive output
		MISOA	PA7	CMOS/Open drain	Normal/high drive output
			PC7	CMOS/Open drain	Normal/high drive output
	RSPI1	RSPCKB	P27	CMOS/Open drain	Normal/high drive output
			PE5	CMOS/Open drain	Normal/high drive output
		MOSIB	P26	CMOS/Open drain	Normal/high drive output
			PE6	CMOS/Open drain	Normal/high drive output
		MISOB	P30	CMOS/Open drain	Normal/high drive output
			PE7	CMOS/Open drain	Normal/high drive output
RX634	RSPI0	RSPCKA	PA5	CMOS/Open drain	Normal/high drive output
			PB0	CMOS/Open drain	Normal/high drive output
			PC5	CMOS/Open drain	Normal/high drive output
		MOSIA	P16	CMOS/Open drain	Normal/high drive output
			PA6	CMOS/Open drain	Normal/high drive output
			PC6	CMOS/Open drain	Normal/high drive output
		MISOA	P17	CMOS/Open drain	Normal/high drive output
			PA7	CMOS/Open drain	Normal/high drive output
			PC7	CMOS/Open drain	Normal/high drive output

Table 8.4 RSPI Pin Functions for Each Microcontroller (3)

Microcontroller	Channel	Pin	Port	Open Drain Control Register (ODR)	Drive Capacity Control Register (DSCR)
RX634	RSPI1	RSPCKB	P27	CMOS/Open drain	Normal/high drive output
			PE1*1	CMOS/Open drain	Normal/high drive output
			PE5	CMOS/Open drain	Normal/high drive output
		MOSIB	P26	CMOS/Open drain	Normal/high drive output
			PE2	CMOS/Open drain	Normal/high drive output
			PE6	CMOS/Open drain	Normal/high drive output
		MISOB	P30	CMOS/Open drain	Normal/high drive output
			PE3	CMOS/Open drain	Normal/high drive output
			PE7	CMOS/Open drain	Normal/high drive output

Note: 1. When setting open drain control register 0 (ODR0), it is necessary to set 2 bits (bits 2 and 3) for port PE1 only.

8.11 Differences Between Microcontrollers Used

Table 8.3 lists the differences between the microcontrollers used.

Table 8.5 Differences

Section	Item	Remarks
2	Verified operating conditions	
6.3	Size of required memory	
6.4	File: Common interface module definitions	
7.2.2 (4)	Used pin definitions	
7.2.2 (9)	Drive capacity control register (DSCR) settings	
8.8	Notes on drive capacity control register (DSCR) settings	
—	Modules timings that can be set	*1

Note: 1. When setting the bit rate, be sure to fully verify the settings with the hardware manual for the microcontroller used.

Website and Support

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

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

REVISION HISTORY	RX Series Application Note Clock Synchronous Single Master Control Software Using the RSPI

Rev.	Date	Description	
		Page	Summary
1.08	Sep. 26, 2013	—	First edition issued
1.09	Dec. 13, 2013	—	Added RX111 Group.
		13-14	2. Verified Operating Conditions: Added (6) For the RX111.
		15	3. Related Application Notes: Added the following. Micron Technology N25Q Serial NOR Flash Memory Control Software (R01AN1528EJ) Micron Technology P5Q Serial Phase Change Memory Control Software (R01AN1439EJ) Spansion S25FLxxxS MirrorBit® Flash Non-Volatile Memory Control Software (R01AN1529EJ)
		21	6.3 Size of Required Memory: Added (6) For the RX111.
		22	6.4 File Configuration: Changed application note numbers. Added for the RX111.
		40	6.10.4 SIO_DATAO_INIT() (2) Function; Added for the RX111.
		41	6.10.6 SIO_CLK_INIT() (2) Function; Added for the RX111.
		56	7.2.2 R_SIO_rspi.h (4) Used Pin Definition; Added for the RX111.
		58	7.2.2 R_SIO_rspi.h (9) Drive Capacity Control Register (DSCR) Settings; Added for the RX111.
		63	8.10 RSPI Pin Port Functions for Each Microcontroller Added for the RX111.
		67	Section 8.10 Added for the RX64M.
1.10	Dec. 13, 2013	—	Added RX64M Group.
		5	Section 1 Added “Support for clock synchronous (three-wire method) single master transmit, single master receive, and single master transmit/receive”. Added “RSPI module without RSPCK auto-stop function”. Added “RSPI module with RSPCK auto-stop function”.
		15-16	Section 2 Added (6) For the RX64M.
		16	Section 3 Added “Macronix International MX25/66L Family Serial NOR Flash Memory Control Software (R01AN1967EJ)“.
		22	Section 6.3 Added (7) For the RX64M.
		23	Section 6.4 Changed application note numbers. Added “R_SIO_rspi.h.rx64m”.
		25	Section 6.7 Added “R_SIO_TRx_Data()”.
		26	Section 6.8 Added “R_SIO_TRx_Data()”.
		25	Section 6.7 Added “R_SIO_TRx_Data()”.
		26	Section 6.8 Added “R_SIO_TRx_Data()”.

		40 - 45	Added “6.9.7 Serial I/O Data Transmission/Reception Processing”.
		47	Section 6.10.4 (2) Added for the RX64M.
		48	Section 6.10.6 (2) Added for the RX64M.
		56	Section 7.2.2 (4) Added for the RX64M.
		65	Section 7.2.2 (8) and (9) Added for the RX64M.
		67	Section 8.8 Added for the RX64M.
1.11	Mar. 31, 2015	—	Added RX71M and RX634 Group.
		—	Modified the tittle. Before : RX210, RX21A, RX220, RX63N, RX63T, RX111, RX64M Group Clock Synchronous Single Master Control Software Using the RSPI
		16-19	Section 2 Added (8) For the RX71M and (9) For the RX634.
		26	Section 6.3 Added (8) For the RX71M and (9) For the RX634.
		27	Section 6.4 Changed application note numbers. Added “R_SIO_rspi.h.rx71m” and “R_SIO_rspi.h.rx634”.
		51	Section 6.10.4 (2) Added for the RX71M and RX634.
		52	Section 6.10.6 (2) Added for the RX71M and RX634.
		66,67	Section 7.2.2 (4) Added for the RX71M and RX634.
		69	Section 7.2.2 (8) Added for the RX634.
		70	Section 7.2.2 (9) Added for the RX71M and RX634.
		71	Section 8.8 Added for the RX634.
		74-75	Section 8.10 Added for the RX71M and RX634.

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

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

4. Clock Signals

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

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

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

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

RENESAS

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Colophon 5.0