RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group

Clock Synchronous Single Master Control Software Using CSI Mode of Serial Array Unit

R01AN1195EJ0105
Rev.1.05
Mar 31, 2016

Introduction

This application note explains clock synchronous control of a single master by using the 3-wire serial I/O communications (CSI mode) of the serial array unit (SAU) of the RL78/G14, RL78/G1C, RL78/L12, RL78/L13, RL78/L1C Group and describes how to use the sample code for this application.

The SPI mode single master can be controlled by adding control of SPI slave device selection through port control.

This sample code lies in a lower-level layer of the software for controlling a SPI device as a slave device.

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 of 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 Device

Corresponding MCU: RL78/G14, RL78/G1C Group

RL78/L12, RL78/L13, RL78/L1C Group

Device used for checking the operation of the sample code: Renesas Electronics R1EX25xxx Series SPI Serial EEPROM

When applying the contents of this application note to other series of microcomputers, make necessary modifications to and make extensive evaluations of the sample code according to the specifications for the microcomputer to be used.

Note that the term “RL78 Family microcontroller” is used in this document for ease of description since the target devices come from multiple groups.

Contents

1. Specifications

This software program uses the 3-wire serial I/O communications (CSI mode) of the serial array unit (SAU) of the RL78 Family microcontroller to control clock synchronous communication. The SPI mode single master can be controlled by adding control of SPI slave device selection through port control.

Table 1-1 summarizes the peripheral devices to be used and their uses. Figure 1.1 illustrates a sample configuration.

The major functions are summarized below.

- This software is a block-type device driver that uses the 3-wire serial I/O communications (CSI mode) of the SAU of the RL78 Family microcontroller as the master device in clock synchronous single master communication.
- The MCU's internal clock synchronous (3-wire) serial communication function is used. It can only be used with a single user-configured channel; that is, it cannot be used with multiple channels.
- The sample code does not support chip-select control. To control the SPI device, the chip-select control must be separately embedded.
- This software supports MSB-first transfer.
- The software supports transfer by the CPU but not by the DMAC.
- It does not support using an interrupt to start the transfer.

Table 1-1 Peripheral Devices Used and their Uses

Peripheral Device	Use
SAU	Clock synchronous (3-wire method) serial 1 channel (required)
Port	For SPI slave device select control signals. As many ports as there are SPI slave devices in use are necessary (required). Not used by this sample code.

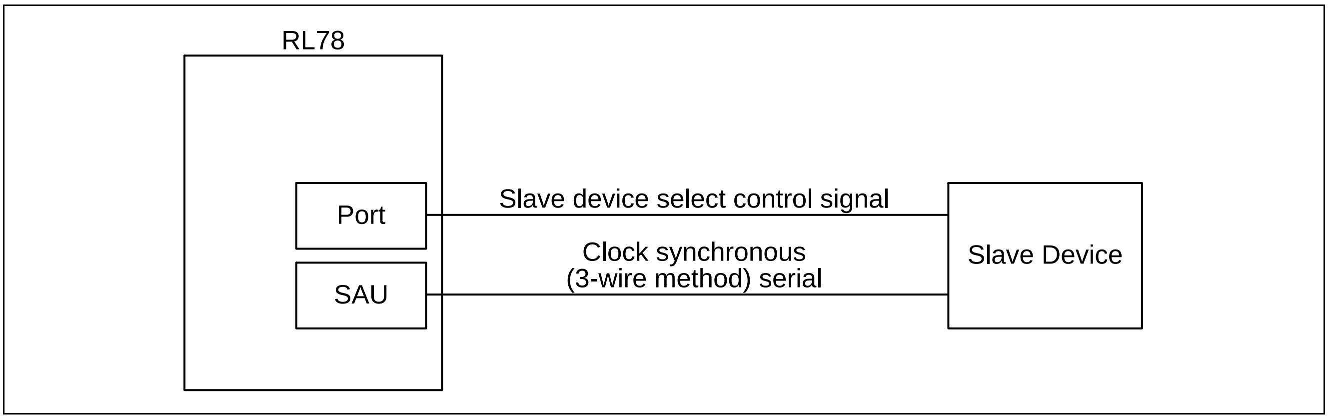


Figure 1.1 Sample Configuration

2. Conditions of Checking the Operation of the Software

The sample code described in this application note has been confirmed to run normally under the operating conditions given below.

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

Table 2-1 Operating Conditions

Item	Description
Microcomputer used for evaluation	RL78/G14 Group (Program ROM: 256 KB, RAM: 24 KB)
Memory used for evaluation	Renesas Electronics R1EX25xxx Series SPI Serial EEPROM
Operating frequency	Main system clock: 32 MHz Peripheral hardware clock: 32 MHz Serial clock: 4 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics CS+ for CA, CX V3.01.00
C compiler	Renesas Electronics RL78,78K0R compiler CA78K0R V1.71
	Compiler options: The default settings (-qx2) for the integrated development environment are used.
Version of the sample code	Ver.2.05
Software used for evaluation	RX Family, RL78 Family, 78K0R/Kx3-L Renesas R1EX25xxx Series Serial EEPROM Control Software, (R01AN0565EJ) Ver.2.04
Evaluation board used	Renesas Starter Kit for RL78/G14

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

Table 2-2 Operating Conditions

Item	Description
Microcomputer used for evaluation	RL78/G14 Group (Program ROM: 256 KB, RAM: 24 KB)
Memory used for evaluation	Renesas Electronics R1EX25xxx Series SPI Serial EEPROM
Operating frequency	Main system clock: 32 MHz Peripheral hardware clock: 32 MHz Serial clock: 4 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics CS+ for CC V3.03.00
C compiler	Renesas Electronics RL78 compiler CC-RL V1.02.00
	Compiler options: The default settings (Perform the default optimization(None)) for the integrated development environment are used.
Version of the sample code	Ver.2.05
Software used for evaluation	RX Family, RL78 Family, 78K0R/Kx3-L Renesas R1EX25xxx Series Serial EEPROM Control Software, (R01AN0565EJ) Ver.2.04
Evaluation board used	Renesas Starter Kit for RL78/G14

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

Table 2-3 Operating Conditions

Item	Description
Microcomputer used for evaluation	RL78/G14 Group (Program ROM: 256 KB, RAM: 24 KB)
Memory used for evaluation	Renesas Electronics R1EX25xxx Series SPI Serial EEPROM
Operating frequency	Main system clock: 32 MHz Peripheral hardware clock: 32 MHz Serial clock: 4 MHz
Operating voltage	3.3 V
Integrated development environment	IAR Systems IAR Embedded Workbench for Renesas RL78 (Ver.1.30.2)
C compiler, assembler	IAR Systems IAR Assembler for Renesas RL78 (Ver.1.30.2.50666) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.2.50666)
	Compiler options: The default settings ("level: low") for the integrated development environment are used.
Version of the sample code	Ver.2.03
Software used for evaluation	Renesas Electronics The R1EX25xxx Series SPI Serial EEPROM Control Software, (R01AN0565EJ) Ver.2.02
Evaluation board used	Renesas Starter Kit for RL78/G14

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

Table 2-4 Operating Conditions

Item	Description
Microcomputer used for evaluation	RL78/G1C Group (Program ROM: 32 KB, RAM: 5.5 KB)
Memory used for evaluation	Renesas Electronics R1EX25xxx Series SPI Serial EEPROM
Operating frequency	Main system clock: 24 MHz Peripheral hardware clock: 24 MHz Serial clock: 4 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics CubeSuite+ V2.01.00
C compiler	Renesas Electronics RL78,78K0R compiler CA78K0R V1.70
	Compiler options: The default settings (-qx2) for the integrated development environment are used.
Version of the sample code	Ver.2.03
Software used for evaluation	Renesas Electronics The R1EX25xxx Series SPI Serial EEPROM Control Software, (R01AN0565EJ) Ver.2.03 R01
Evaluation board used	Renesas RL78/G1C Target Board QB-R5F10JGC-TB

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

Table 2-5 Operating Conditions

Item	Description
Microcomputer used for evaluation	RL78/G1C Group (Program ROM: 32 KB, RAM: 5.5 KB)
Memory used for evaluation	Renesas Electronics R1EX25xxx Series SPI Serial EEPROM
Operating frequency	Main system clock: 24 MHz Peripheral hardware clock: 24 MHz Serial clock: 4 MHz
Operating voltage	3.3 V
Integrated development environment	IAR Systems IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5)
C compiler, assembler	IAR Systems IAR Assembler for Renesas RL78 (Ver.1.30.4.50715) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715)
	Compiler options: The default settings ("level: low") for the integrated development environment are used.
Version of the sample code	Ver.2.03
Software used for evaluation	Renesas Electronics The R1EX25xxx Series SPI Serial EEPROM Control Software, (R01AN0565EJ) Ver.2.03 R01
Evaluation board used	Renesas RL78/G1C Target Board QB-R5F10JGC-TB

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

Table 2-6 Operating Conditions

Item	Description
Microcomputer used for evaluation	RL78/L12 Group (Program ROM: 32 KB, RAM: 1.5 KB)
Memory used for evaluation	Renesas Electronics R1EX25xxx Series SPI Serial EEPROM
Operating frequency	Main system clock: 24 MHz Peripheral hardware clock: 24 MHz Serial clock: 4 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics CubeSuite+ V2.01.00
C compiler	Renesas Electronics RL78,78K0R compiler CA78K0R V1.70
	Compiler options: The default settings (-qx2) for the integrated development environment are used.
Version of the sample code	Ver.2.03
Software used for evaluation	Renesas Electronics The R1EX25xxx Series SPI Serial EEPROM Control Software, (R01AN0565EJ) Ver.2.03 R01
Evaluation board used	Renesas Starter Kit for RL78/L12

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

Table 2-7 Operating Conditions

Item	Description
Microcomputer used for evaluation	RL78/L12 Group (Program ROM: 32 KB, RAM: 1.5 KB)
Memory used for evaluation	Renesas Electronics R1EX25xxx Series SPI Serial EEPROM
Operating frequency	Main system clock: 24 MHz Peripheral hardware clock: 24 MHz Serial clock: 4 MHz
Operating voltage	3.3 V
Integrated development environment	IAR Systems IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5)
C compiler, assembler	IAR Systems IAR Assembler for Renesas RL78 (Ver.1.30.4.50715) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715)
	Compiler options: The default settings ("level: low") for the integrated development environment are used.
Version of the sample code	Ver.2.03
Software used for evaluation	Renesas Electronics The R1EX25xxx Series SPI Serial EEPROM Control Software, (R01AN0565EJ) Ver.2.03 R01
Evaluation board used	Renesas Starter Kit for RL78/L12

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

Table 2-8 Operating Conditions

Item	Description
Microcomputer used for evaluation	RL78/L13 Group (Program ROM: 128 KB, RAM: 8 KB)
Memory used for evaluation	Renesas Electronics R1EX25xxx Series SPI Serial EEPROM
Operating frequency	Main system clock: 24 MHz Peripheral hardware clock: 24 MHz Serial clock: 4 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics CubeSuite+ V2.01.00
C compiler	Renesas Electronics RL78,78K0R compiler CA78K0R V1.70
	Compiler options: The default settings (-qx2) for the integrated development environment are used.
Version of the sample code	Ver.2.03
Software used for evaluation	Renesas Electronics The R1EX25xxx Series SPI Serial EEPROM Control Software, (R01AN0565EJ) Ver.2.03 R01
Evaluation board used	Renesas Starter Kit for RL78/L13

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

Table 2-9 Operating Conditions

Item	Description
Microcomputer used for evaluation	RL78/L13 Group (Program ROM: 128 KB, RAM: 8 KB)
Memory used for evaluation	Renesas Electronics R1EX25xxx Series SPI Serial EEPROM
Operating frequency	Main system clock: 24 MHz Peripheral hardware clock: 24 MHz Serial clock: 4 MHz
Operating voltage	3.3 V
Integrated development environment	IAR Systems IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5)
C compiler, assembler	IAR Systems IAR Assembler for Renesas RL78 (Ver.1.30.4.50715) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715)
	Compiler options: The default settings ("level: low") for the integrated development environment are used.
Version of the sample code	Ver.2.03
Software used for evaluation	Renesas Electronics The R1EX25xxx Series SPI Serial EEPROM Control Software, (R01AN0565EJ) Ver.2.03 R01
Evaluation board used	Renesas Starter Kit for RL78/L13

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

Table 2-10 Operating Conditions

Item	Description
Microcomputer used for evaluation	RL78/L1C Group (Program ROM: 256 KB, RAM: 16 KB)
Memory used for evaluation	Renesas Electronics R1EX25xxx Series SPI Serial EEPROM
Operating frequency	Main system clock: 24 MHz Peripheral hardware clock: 24 MHz Serial clock: 4 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics CubeSuite+ V2.01.00
C compiler	Renesas Electronics RL78,78K0R compiler CA78K0R V1.70
	Compiler options: The default settings (-qx2) for the integrated development environment are used.
Version of the sample code	Ver.2.03
Software used for evaluation	Renesas Electronics The R1EX25xxx Series SPI Serial EEPROM Control Software, (R01AN0565EJ) Ver.2.03 R01
Evaluation board used	Renesas Starter Kit for RL78/L1C

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

Table 2-11 Operating Conditions

Item	Description
Microcomputer used for evaluation	RL78/L1C Group (Program ROM: 256 KB, RAM: 16 KB)
Memory used for evaluation	Renesas Electronics R1EX25xxx Series SPI Serial EEPROM
Operating frequency	Main system clock: 24 MHz Peripheral hardware clock: 24 MHz Serial clock: 4 MHz
Operating voltage	3.3 V
Integrated development environment	IAR Systems IAR Embedded Workbench for Renesas RL78 (Ver.1.30.5)
C compiler, assembler	IAR Systems IAR Assembler for Renesas RL78 (Ver.1.30.4.50715) IAR C/C++ Compiler for Renesas RL78 (Ver.1.30.5.50715)
	Compiler options: The default settings ("level: low") for the integrated development environment are used.
Version of the sample code	Ver.2.03
Software used for evaluation	Renesas Electronics The R1EX25xxx Series SPI Serial EEPROM Control Software, (R01AN0565EJ) Ver.2.03 R01
Evaluation board used	Renesas Starter Kit for RL78/L1C

3. Related Application Notes

The applications notes that are related to this application note are listed below. Reference should also be made to those application notes.

- Renesas R1EX25xxx Series Serial EEPROM Control Software (R01AN0565EJ)
- Micron Technology M25P Series Serial Flash Memory Control Software (R01AN0566EJ0101)
- Micron Technology M45PE Series Serial Flash Memory Control Software (R01AN0567EJ0101)
- Micron Technology P5Q Serial Phase Change Memory Control Software (R01AN1439EJ)
- Micron Technology N25Q Serial NOR Flash Memory Control Software (R01AN1528EJ)
- Spansion S25FLxxxS MirrorBit® Flash Non-Volatile Memory Control Software (R01AN1529EJ)

4. Hardware Description

4.1 List of Pins

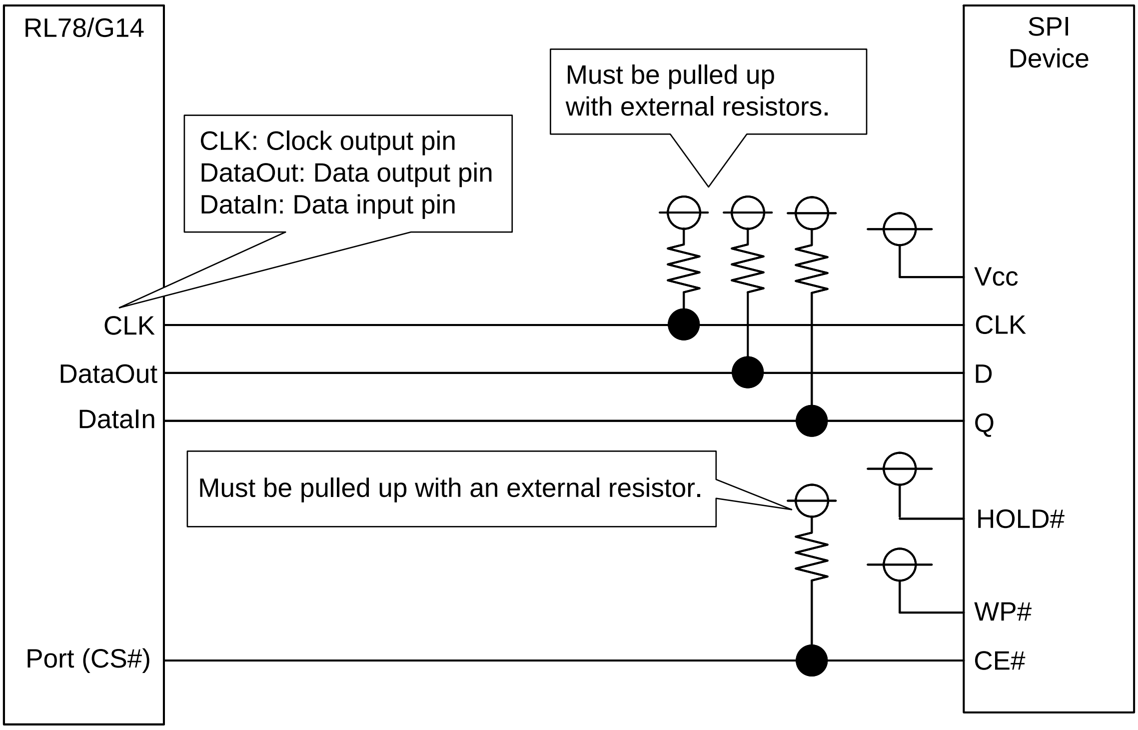
The following table lists the pins that are used and their uses.

Table 4.1 List of Pins Used

Pin Name	I/O	Description
SCK (CLK of Figure 4.1)	Output	Clock output
SO (DataOut of Figure 4.1)	Output	Master data output
SI (DataIn of Figure 4.1)	Input	Master data input
Port (Port(CS#) of Figure 4.1)	Output	Slave device select output Not used by this sample code.

4.2 Reference Circuit

Figure 4.1 shows a sample wiring configuration.



• The names of the MCU pins used for serial I/O are dependent on the individual MCU.
• In this application note, pin names CLK, DataIn, DataOut, and Port (CS#) are used in accordance with the sample code.

Figure 4.1 Sample Wiring Diagram for a RL78 Family microcontroller Serial Array Unit and an SPI Slave Device

5. Software Description

5.1 Operation Outline

The 3-wire serial I/O communications (CSI mode) of the SAU are used to implement clock synchronous single master control.

The sample code provides the following control functions:

- Controls the input/output of the data in the clock synchronous mode (using an internal clock).

In this sample code, the byte offset value of the data on the device is made equal to the byte offset value in the source or destination memory as illustrated in the figure below.

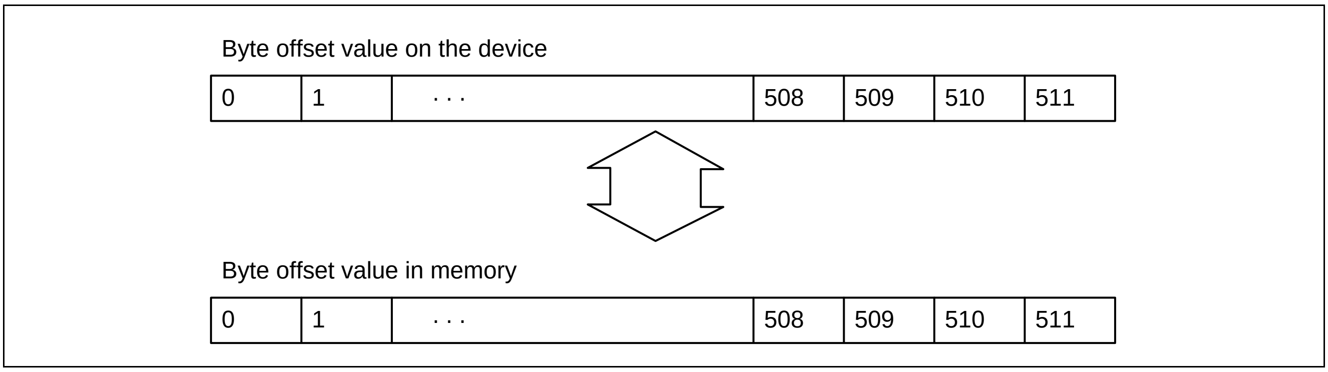
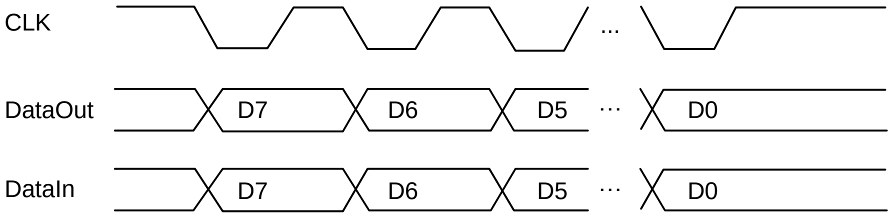


Figure 5.1 Storage Format of the Transferred Data

5.1.1 Clock Synchronous Mode Timing

The SPI mode 3 (CPOL=1, CPHA=1) timing shown in Figure 5.2 is used to control the SPI slave device. Therefore, the data and clock phase select bits (DAPmn and CKPmn) in the serial communication operation setting register (SCRmn) of the RL78 Family microcontroller must be set for type 1 (DAPmn=0, CKPmn=0).



- MCU → SPI slave device transmission: Transmission of transmit data is started on the falling edge of the transfer clock.
- SPI slave device → MCU reception: The receive data is taken in on the rising edge of the transfer clock.
- MSB first mode transfer

The level of the CLK pin is held high when no transfer processing is in progress.

Figure 5.2 Clock Synchronous Mode Timing Setup

For available serial clock frequencies, see the datasheets for the individual MCUs and SPI slave devices.

5.1.2 SPI Slave Device CE# Pin Control

It is recommended that the CE# pin of the SPI slave device be connected to the Port pin of the RL78 Family microcontroller. This enables the SPI slave device to be controlled by using general port output from the RL78 Family microcontroller.

Secure the time between the falling edge of the CE# signal of the SPI device (the Port signal of the MCU (CS#)) and that of the CLK signal of the SPI device (the clock signal of the MCU) as the setup time of the CE# pin of the SPI device.

Secure the time between the rising edge of the CLK signal of the SPI device (the CLK signal of the MCU) and that of the /S signal of the SPI device (the Port signal of the MCU (CS#)) as the hold time of the CE# pin of the SPI device.

Check the datasheet for the SPI device in use and set up the software wait times that are appropriate to your system.

5.2 Software Control Outline

5.2.1 Software Configuration

The sample code ranks in the lower-level layer of the SPI device control software as a slave device.

The sample code realizes the control the clock synchronous single master by using SPI mode 3 (CPOL = 1 and CPHA = 1) without controlling the CE# pin of the SPI slave device.

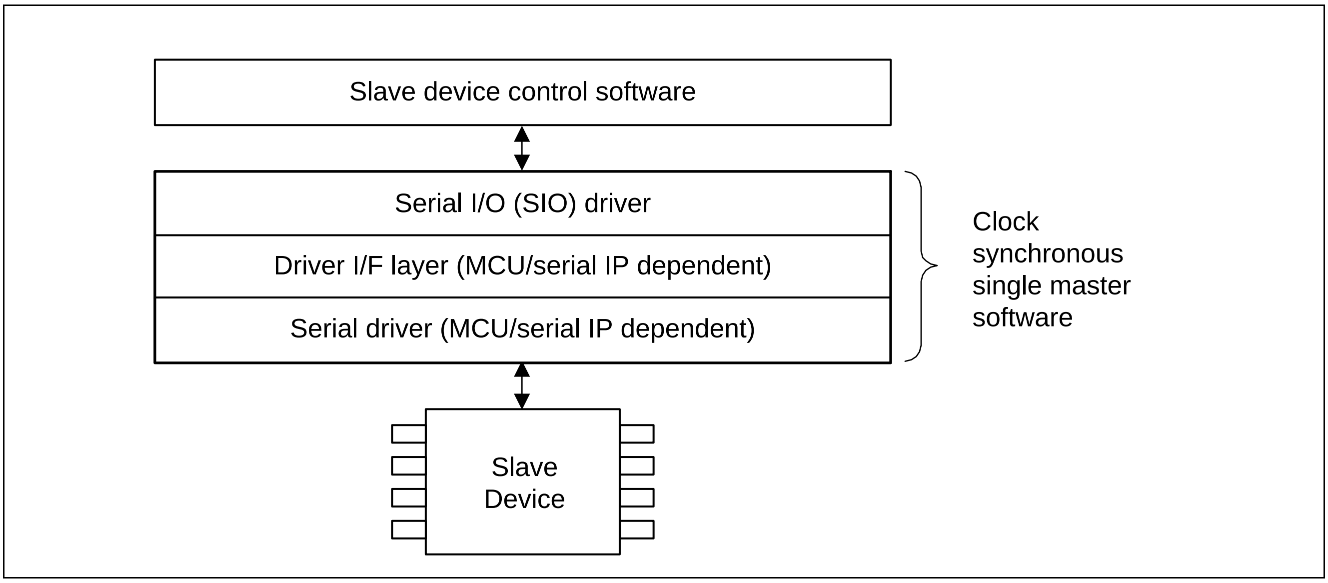


Figure 5.3 Software Configuration

The following transmission and reception are realized.

(1) Sends data using the clock synchronous single master software.
(2) Receives data using the clock synchronous single master software.

This sample code is made up of the following five basic routines:

- Serial enabling
  Sets the DataIn pin for port input, sets the DataOut and CLK pins high, enables serial I/O and set the baud rate.
- Serial disabling
  Disables serial I/O, sets the DataIn pin for port input, sets the DataOut and CLK pins high.
- Serial opening
  Disables serial I/O, sets the DataIn pin for port input, sets the DataOut and CLK pins for port input.
- Data transmission
  Sends data to the SPI device.
- Data reception
  Receives data from the SPI device.

5.2.2 Serial Enabling (R_SIO_Enable())

Sets the DataIn pin to be used for serial I/O for port input and set the DataOut and CLK pins high.

Enables the serial I/O function and switches the DataIn pin for data input, the DataOut pin for data output, and the CLK pin for clock output.

Sets the communication speed (baud rate) to be used for serial I/O.

5.2.3 Serial Disabling (R_SIO_Disable())

Switches the pins to be used for serial I/O to function as ports, sets the DataIn pin to port input, and sets the DataOut and CLK pins to high output.

5.2.4 Serial Opening (R_SIO_Open_Port())

Switches the pins to be used for serial I/O to function as ports, sets the DataIn, DataOut, and CLK pins to port input.

5.2.5 Data Transmission (R_SIO_Tx_Data())

Sends data using the serial I/O function.

Sends data according to the transmission setting.

5.2.6 Data Reception (R_SIO_Rx_Data())

Receives data using the serial I/O function.

Receives data according to the transmission/reception settings.

5.2.7 Data Transmission/Reception (R_SIO_TRx_Data())

Sends and Receives data using the serial I/O function.

Sends and Receives data according to the transmission/reception settings.

5.3 Sizes of Required Memory

The sizes of the required memory areas for each MCU of different instructions are given below. Investigate the instructions of MCU to be used and give by reference.

See chapter 2, Conditions of Checking the Operation of the Software, for the environment.

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

Table 5-1 Sizes of Required Memory

Memory Used	Size	Remarks
ROM	653 bytes	R_SIO_csi.c
RAM	0 bytes	R_SIO_csi.c
Maximum user stack size	24 bytes	
Maximum interrupt stack size	—	Not interrupt used

Note: The required memory size differs with the C compiler version and the compiler options used.

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

Table 5-2 Sizes of Required Memory

Memory Used	Size	Remarks
ROM	598 bytes	R_SIO_csi.c
RAM	0 bytes	R_SIO_csi.c
Maximum user stack size	20 bytes	
Maximum interrupt stack size	—	Not interrupt used

Note: The required memory size differs with the C compiler version and the compiler options used.

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

Table 5-3 Sizes of Required Memory

Memory Used	Size	Remarks
ROM	547 bytes	R_SIO_csi.c
RAM	0	R_SIO_csi.c
Maximum user stack size	94 bytes	
Maximum interrupt stack size	—	Not interrupt used

Note: The required memory size differs with the C compiler version and the compiler options used.
The maximum user stack size is the stack size for the whole project.

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

Table 5-4 Sizes of Required Memory

Memory Used	Size	Remarks
ROM	525 bytes	R_SIO_csi.c
RAM	0 bytes	R_SIO_csi.c
Maximum user stack size	22 bytes	
Maximum interrupt stack size	—	Not interrupt used

Note: The required memory size differs with the C compiler version and the compiler options used.

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

Table 5-5 Sizes of Required Memory

Memory Used	Size	Remarks
ROM	516 bytes	R_SIO_csi.c
RAM	0	R_SIO_csi.c
Maximum user stack size	94 bytes	
Maximum interrupt stack size	—	Not interrupt used

Note: The required memory size differs with the C compiler version and the compiler options used.
The maximum user stack size is the stack size for the whole project.

5.4 File Configuration

The following table lists the files that are used for the sample code. The table excludes the files that are automatically generated by the integrated development environment.

Table 5-6 File Configuration

\an_r01an1195ej0105_rl78_serial				<DIR>	Folder for the sample code
	r01an1195ej0105_rl78.pdf				Application note
	\source			<DIR>	Folder for storing the programs
		\com*1		<DIR>	Folder for storing the common functions
			mtl_com.c		Miscellaneous common function definitions
			mtl_com.h.common		Common header file
			mtl_com.h.RL78		Common function header file
			mtl_endi.c		Common file (related to endian setting)
			mtl_mem.c		Common file (standard library function)
			mtl_os.c	mtl_os.h	Common file (standard library function)
			mtl_str.c		Common file (standard library function)
			mtl_tim.c	mtl_tim.h	Common file (related to loop timer)
			mtl_tim.h.sample		Sample for setting the value in the loop timer
		\r_sio_csi_rl78		<DIR>	Folder for clock synchronous single master control software using the SCI for the RL78
			R_SIO.h		Header file
			R_SIO_csi.c		I/F module
			R_SIO_csi.h.rl78g1c		I/F module common definitions (for RL78/G1C)
			R_SIO_csi.h.rl78g14		I/F module common definitions (for RL78/G14)
			R_SIO_csi.h.rl78l1c		I/F module common definitions (for RL78/L1C)
			R_SIO_csi.h.rl78l12		I/F module common definitions (for RL78/L12)
			R_SIO_csi.h.rl78l13		I/F module common definitions (for RL78/L13)

Note: *1 The files in the com folder are used in the slave device control software, too.
Use the latest files.

5.5 List of Constants

5.5.1 Return Values

The following table lists the return values that are returned by the sample code.

Table 5-7 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

5.5.2 Miscellaneous Definitions

The following table lists miscellaneous definitions that are used in the sample code.

Table 5-8 Miscellaneous Definitions

Constant Name	Value	Description
SIO_LOG_ERR	1	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 µs = 50 ms
SIO_RX_WAIT	(uint16_t)50000	SIO receive completion waiting time 50000 × 1 µs = 50 ms
SIO_DMA_TX_WAIT	(uint16_t)50000	DMA transmission completion waiting time 50000 × 1 µs = 50 ms
SIO_DMA_RX_WAIT	(uint16_t)50000	DMA receive completion waiting time 50000 × 1 µs = 50 ms
SIO_T_SIO_WAIT	(uint16_t)MTL_T_1US	SIO transmit and receive completion waiting polling time
SIO_T_DMA_WAIT	(uint16_t)MTL_T_1US	DMA transmit and receive completion waiting polling time
SIO_T_BRR_WAIT	(uint16_t)MTL_T_10US	BRR setting wait time

5.6 Structures and Unions

Shown below are the structures that are used in the sample code.

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


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

5.7 List of Functions

The following table lists the functions that are used in the sample code.

Table 5-9 List of Functions

Function Name	Description
R_SIO_Init_Driver()	Driver initialization processing
R_SIO_Disable()	Serial I/O disable setting processing
R_SIO_Enable()	Serial I/O enable setting processing
R_SIO_Open_Port()	Serial I/O open setting processing
R_SIO_Tx_Data()	Serial I/O data transmit processing
R_SIO_Rx_Data()	Serial I/O data receive processing
R_SIO_TRx_Data()	Serial I/O data transmit/receive processing

5.8 Function Specifications

The sample code enables supply of the input clock to the serial array unit but does not include processing to control stopping of the input clock.

Therefore, the user should provide additional program code with the necessary control functions if there is a need to stop operation of individual units in order to reduce power consumption and noise, taking into account the control of channels other than the one used by the sample code.

Note that the sample code does not provide the capability to stop operation of individual units, but it can be used to stop operation of a specific channel.

Operating clock CKm0, specified in the serial clock select register (SPSm), is used as the operating clock in the sample code. If necessary, a different clock can be selected by changing the settings in the serial mode register (SMRmn) and serial clock select register (SPSm).

5.8.1 Driver Initialization Processing

R_SIO_Init_Driver	
Overview	Driver initialization processing
Header	R_SIO.h, R_SIO_csi.h, mtl_com.h
Declaration	error_t R_SIO_Init_Driver(void)
Description	• Initializes the driver. Disables the serial I/O function and set the pin in the port. • This function must be called only once at system start time. • Set the slave device select signal high before calling this function.
Arguments	None
Return values	SIO_OK ; Successful operation
Notes	Performs the following processing, considering the previous use conditions. • Enables supply of the input clock to the serial array unit. • Stops transmission/reception. • Sets the pins to be used for serial I/O to function as ports.

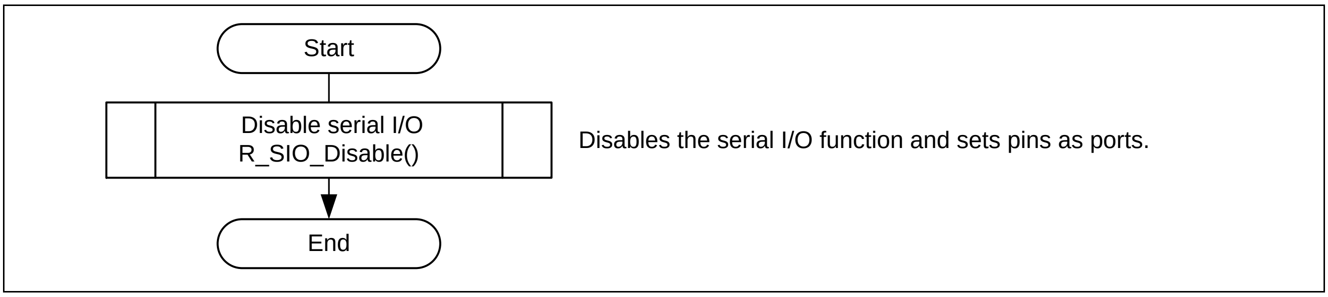
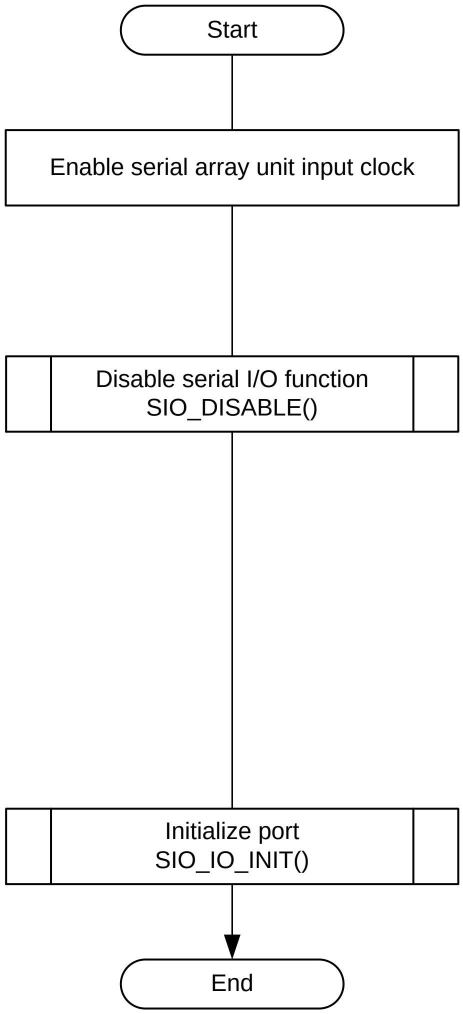


Figure 5.4 Driver Initialization Processing Outline

5.8.2 Serial I/O Disable Setting Processing

R_SIO_Disable	
Overview	Serial I/O disable setting processing
Header	R_SIO.h, R_SIO_csi.h, mtl_com.h
Declaration	error_t R_SIO_Disable(void)
Description	• Disables the serial I/O function and sets the pins to function as ports. Enables supply of the input clock to the serial array unit. Disables serial I/O. Sets the pins to be used for serial I/O to function as ports. • Set the slave device select signal high before calling this function.
Arguments	None
Return values	SIO_OK ; Successful operation
Notes	• Enables supply of the input clock to the serial array unit. • Waits a minimum of 4 cycles of fCLK. • Sets STm, SOm, and SOEm to stop operation and switches pins to function as ports. • Sets SCRmn to set communication disabled as the communication mode. • Writes 0020h to SMRmn (value after a reset) to initialize it. • Sets SOLm. • This function can be called to disable the serial I/O function when serial I/O is not used. • This function does not control stopping supply of the input clock to the serial array unit.



Enables the serial array unit input clock.
- PERn SAUmEN ← 1b: Enables the input clock supply.
  Enables reading/writing to registers.
- Executes the NOP instruction 4 times
  (to allow duration of min. 4 (fCLK clock cycles).

Disables the serial I/O function.
- SOm CKOmn ← 1b: Sets clock output high (for use as port function).
  SOm SOmn ← 1b: Sets data output high (for use as port function).
- STm STmn ← 1b: Stops channel operation.
  (Stops clock output by communication.)
  SEmn is cleared to 0, and moves to the operation stop mode.
- SOEm SOEmn ← 0b: Stops data output by communication.
- SOm CKOmn ← 1b: Sets clock output high (for use as port function).
  SOm Somn ← 1b: Sets data output high (for use as port function).
- SCRmn TEXmn ← 0b, REXmn ← 0b: Stops communication.
- SMRmn ← 0020h: Sets to the value after a reset.
- Sets SOLm.

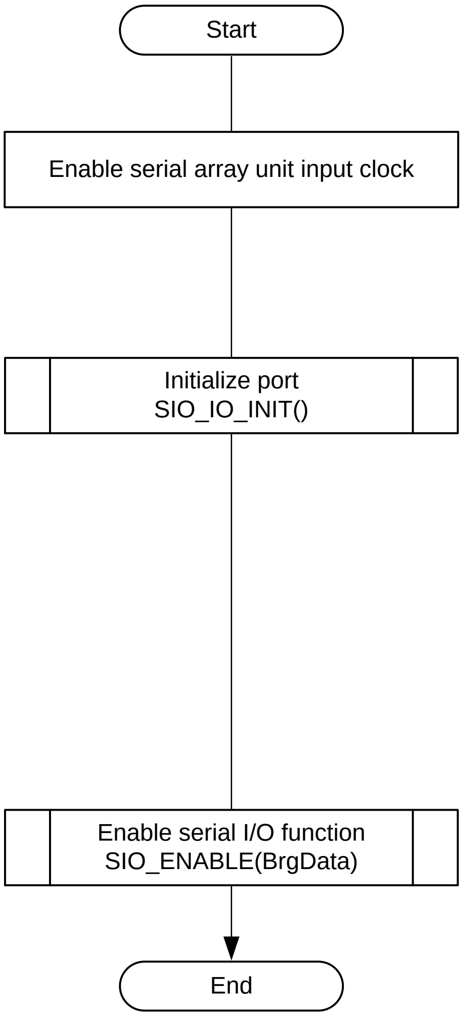
Sets the DataIn pin to port input, the DataOut pin to port high output, and the CLK pin to port high output.
- DataIn pin NFEN SNFEN ← 0b: Noise filter off
  DataIn pin PIMxx ← 0b: Normal input buffer
  DataIn pin PMxx ← 1b: Input mode (output buffer off)
- DataOut pin POMxx ← 0b: Normal output mode
  DataOut pin Pxx ← 1b: HIgh output
  DataOut pin PMxx ← 0b: Output mode (output buffer on)
  DataOut pin Pxx ← 1b: HIgh output
- CLK pin POMxx ← 0b: Normal output mode
  CLK pin Pxx ← 1b: HIgh output
  CLK pin PMxx ← 0b: Output mode (output buffer on)
  CLK pin Pxx ← 1b: HIgh output

Figure 5.5 Serial I/O Disable Setup Processing Outline

5.8.3 Serial I/O Enable Setting Processing

R_SIO_Enable

Overview	Serial I/O enable setting processing
Header	R_SIO.h, R_SIO_csi.h, mtl_com.h
Declaration	error_t R_SIO_Enable(uint8_t BrgData)
Description	• Enables the serial I/O function and sets the baud rate. Enables supply of the input clock to the serial array unit. Sets the pin to be used for serial I/O in the port. Enables the serial I/O and sets the baud rate. • Call this function after calling R_SIO_Disable() • Call this function once before performing serial I/O data transmit processing and serial I/O data receive processing. • To change the baud rate, disable serial I/O setting, and then, use this function.
Arguments	uint8_t BrgData ; Bit rate setting value
Return values	SIO_OK ; Successful operation
Notes	Executes the following processing according to the initial setting procedure for master transmission and master transmission/reception described in the hardware manual. (Assumes that R_SIO_Disable() has been called.) (1) Sets PER SAUmEN. Enables supply of the input clock to the serial array unit. (2) Waits for at least 4 fCLK clock cycles. (3) Initializes ports. (4) Sets the operating clock in SPSm. (5) Sets the operating mode in SSMRmn. (6) Sets the communication format in SCRmn. (7) Sets the baud rate in SDRmn. (8) Clears the error flags in SIRmn. (9) Sets SOLm.



Enables the serial array unit input clock.
- PERn SAUmEN ← 1b: Enables the input clock supply.
  Enables reading/writing to registers.
- Executes the NOP instruction 4 times
  (to allow duration of min. 4 (fCLK clock cycles).

Sets the DataIn pin to port input, the DataOut pin to port high output, and the CLK pin to port high output.
- DataIn pin NFEN SNFEN ← 0b: Noise filter off
  DataIn pin PIMxx ← 0b: Normal input buffer
  DataIn pin PMxx ← 1b: Input mode (output buffer off)
- DataOut pin POMxx ← 0b: Normal output mode
  DataOut pin Pxx ← 1b: High output
  DataOut pin PMxx ← 0b: Output mode (output buffer on)
  DataOut pin Pxx ← 1b: High output
- CLK pin POMxx ← 0b: Normal output mode
  CLK pin Pxx ← 1b: High output
  CLK pin PMxx ← 0b: Output mode (output buffer on)
  CLK pin Pxx ← 1b: High output

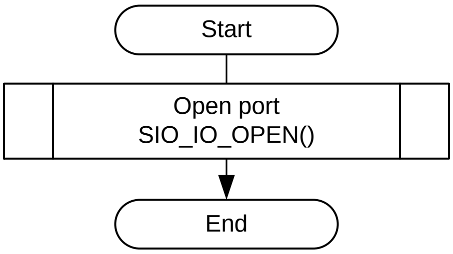
Enables the serial I/O function and sets the bit rate.
- Sets CKm0 as the operating clock in SPSm.
- Sets the operating mode in SMRmn.
  CKm0 selected, software trigger, CSI mode, transfer-end interrupt
- Sets the communication format in SCRmn.
  Data and clock phases, MSB-first, 8-bit
- Sets the baud rate in SDRmn.
- SIRmn ← 0007h: Clears FECT, PBCT, and OVCT.
- Sets SOLm.

Figure 5.6 Serial I/O Enable Setup Processing Outline

5.8.4 Serial I/O Open Setting Processing

R_SIO_Open_Port

Overview	Serial I/O open setting processing
Header	R_SIO.h, R_SIO_csi.h, mtl_com.h
Declaration	error_t R_SIO_Open_Port(void)
Description	• Sets the pin used for serial I/O to "open" (input state). • Set the slave device select signal high before calling this function.
Arguments	None
Return values	SIO_OK ; Successful operation
Notes	Prepared to connect and disconnect removable media. Use this function before connecting and disconnecting the removable media. Perform serial I/O disable setup processing before disconnecting the removable media.



Sets DataIn pin for port input and sets DataOut pin for port input,
Sets CLK pin for port input.

- DataIn pin NFEN SNFEN ← 0b: Noise filter off
  DataIn pin PIMxx ← 0b: Normal input buffer
  DataIn pin PMxx ← 1b: Input mode (output buffer off)
- DataOut pin PMxx ← 1b: Input mode (output buffer off)
- CLK pin PIMxx ← 0b: Normal input buffer
  CLK pin PMxx ← 1b: Input mode (output buffer off)

Figure 5.7 Serial I/O Open Setup Processing Outline

5.8.5 Serial I/O Data Transmit Processing

R_SIO_Tx_Data

Overview	Serial I/O data transmit processing			
Header	R_SIO.h, R_SIO_csi.h, mtl_com.h			
Declaration	error_t R_SIO_Tx_Data(uint16_t TxCnt, uint8_t FAR* pData)			
Description	• Transmits a specified number of bytes of pData. • Perform serial I/O enable setup processing before calling this function. Perform serial I/O disable setup processing in case of unsuccessful operation after calling this function.			
Arguments	uint16_t	TxCnt	;	Number of transmitted bytes
	uint8_t FAR*	pData	;	Transmit data storage buffer pointer
Return values	SIO_OK		;	Successful operation
	SIO_ERR_HARD		;	Hardware error
Notes	• Makes the following initialization settings, following serial I/O enable setting processing, according to the initial setting procedure for master transmission described in the hardware manual. (1) Sets TEXmn=1b and REXmn=0b in SCRmn to enable transmission. (2) Sets data output high and clock output high in SOm. (3) Sets SOEm to enable data output by serial communication operation. (4) Sets SSm to enable clock output by serial communication operation. • After transmit-end, executes the following processing according to the procedure for stopping master transmission described in the hardware manual. (1) Sets STm to disable clock output by serial communication operation. (2) Sets SOEm to disable data output by serial communication operation. (3) Sets TEXmn=0b and REXmn=0b in SCRmn to disable communication. • Recommended to perform serial I/O disable setup processing if this function is not continuously used.			

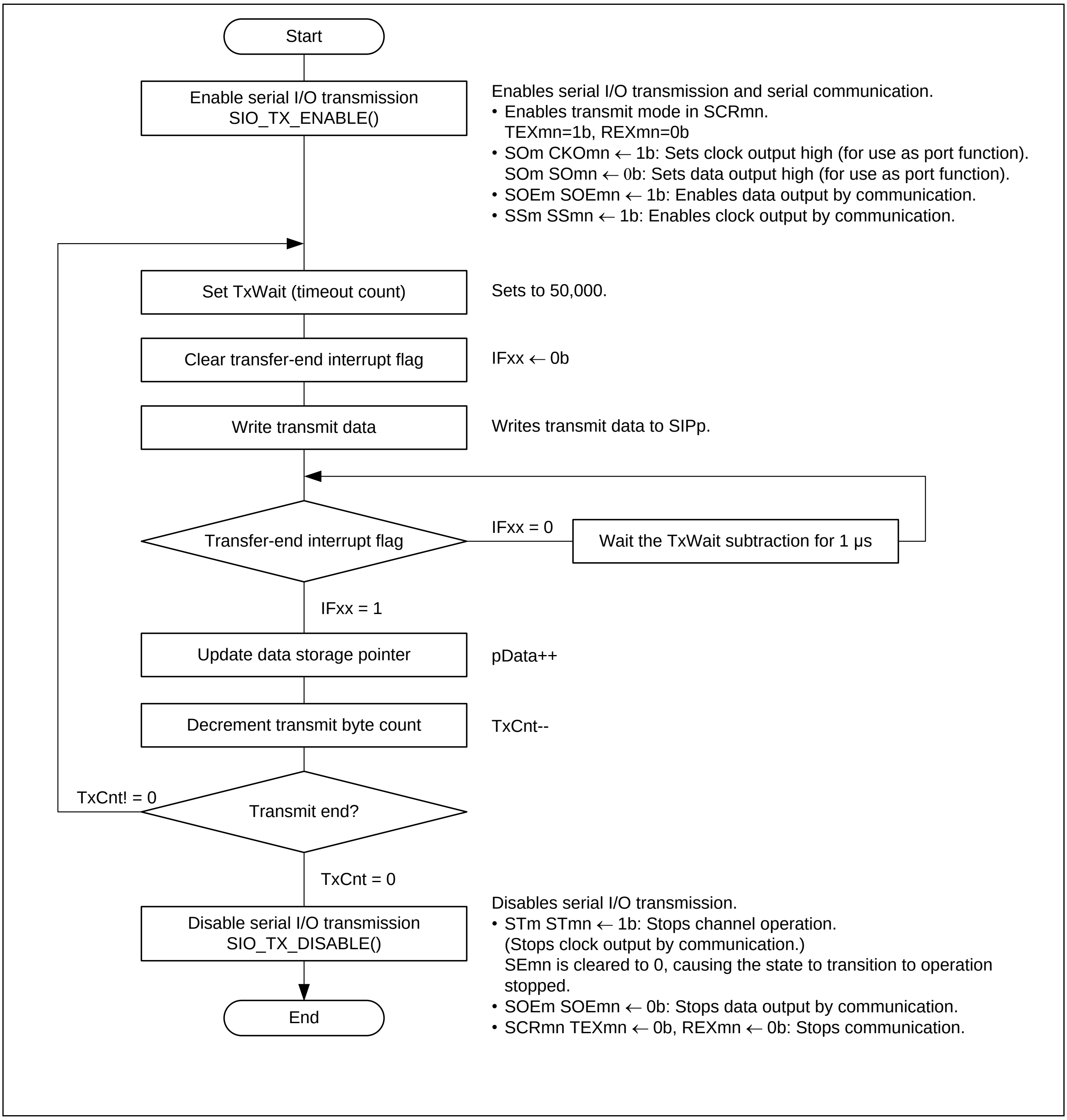


Figure 5.8 Serial I/O Data Transmission Processing Outline

5.8.6 Serial I/O Data Receive Processing

R_SIO_Rx_Data

Overview	Serial I/O data receive processing
Header	R_SIO.h, R_SIO_csi.h, mtl_com.h
Declaration	error_t R_SIO_Rx_Data(uint16_t RxCnt, uint8_t FAR* pData)
Description	• Receives a specified number of data and stores it in pData. • Perform serial I/O enable setup processing before calling this function. • Perform serial I/O disable setup processing in case of unsuccessful operation after calling this function.
Arguments	uint16_t RxCnt ; Number of received bytes uint8_t FAR* pData ; Receive data storage buffer pointer
Return values	SIO_OK ; Successful operation SIO_ERR_HARD ; Hardware error
Notes	• Makes the following initialization settings, following serial I/O enable setting processing, according to the initial setting procedure for master transmission/reception described in the hardware manual. (1) Sets TEXmn=1b and REXmn=1b in SCRmn to enable transmission/reception. (2) Sets data output high and clock output high in SOm. (3) Sets SOEm to enable data output by serial communication operation. (4) Sets SSm to enable clock output by serial communication operation. • After transmit-end, executes the following processing according to the procedure for stopping master transmission/master reception described in the hardware manual. (1) Sets STm to disable clock output by serial communication operation. (2) Sets SOEm to disable data output by serial communication operation. (3) Sets TEXmn=0b and REXmn=0b in SCRmn to disable communication. • Recommended to perform serial I/O disable setup processing if this function is not continuously used.

Transfer-end interrupt flag

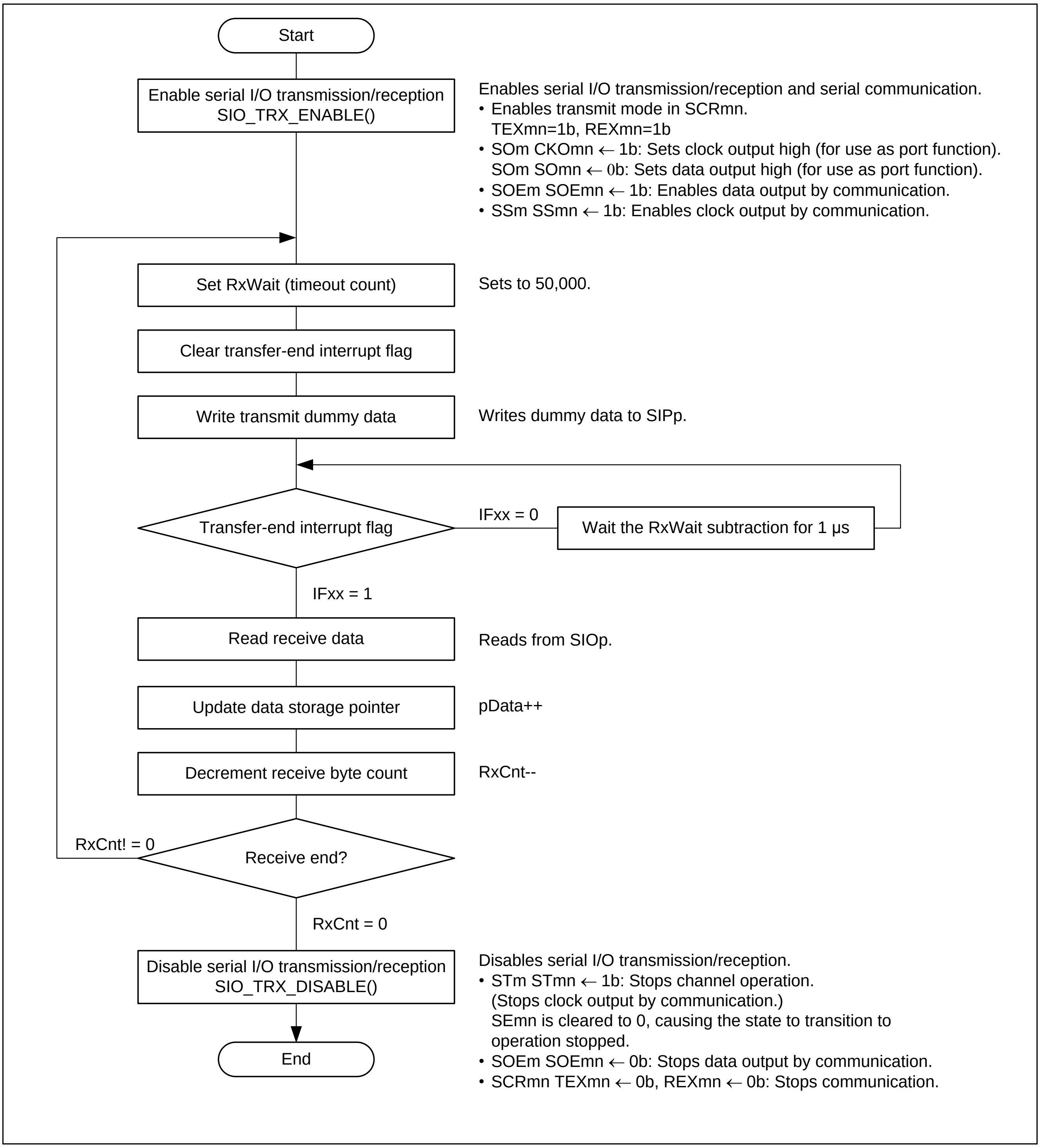


Figure 5.9 Serial I/O Data Reception Processing Outline

5.8.7 Serial I/O Data Transmit/Receive Processing

R_SIO_TRx_Data

Overview Serial I/O data transmit/receive processing

Header R_SIO.h, R_SIO_csi.h, mtl_com.h

Declaration error_t R_SIO_TRx_Data(uint16_t TRxCnt, uint8_t FAR* pTxData, uint8_t FAR* pRxData)

Description

- Transmits a specified number of bytes of pTxData.
- Receives a specified number of data and stores it in pRxData.
- Perform serial I/O enable setup processing before calling this function.
- Perform serial I/O disable setup processing in case of unsuccessful operation after calling this function.

Arguments

uint16_t	TRxCnt	;	Number of transmit/received bytes
uint8_t FAR*	pTxData	;	Transmit data storage buffer pointer
uint8_t FAR*	pRxData	;	Receive data storage buffer pointer

Return values

SIO_OK	;	Successful operation
SIO_ERR_HARD	;	Hardware error

Notes

- Makes the following initialization settings, following serial I/O enable setting processing, according to the initial setting procedure for master transmission/reception described in the hardware manual.
  (1) Sets TEXmn=1b and REXmn=1b in SCRmn to enable transmission/reception.
  (2) Sets data output high and clock output high in SOm.
  (3) Sets SOEm to enable data output by serial communication operation.
  (4) Sets SSm to enable clock output by serial communication operation.
- After transmit-end, executes the following processing according to the procedure for stopping master transmission/master reception described in the hardware manual.
  (1) Sets STm to disable clock output by serial communication operation.
  (2) Sets SOEm to disable data output by serial communication operation.
  (3) Sets TEXmn=0b and REXmn=0b in SCRmn to disable communication.
- Recommended to perform serial I/O disable setup processing if this function is not continuously used.

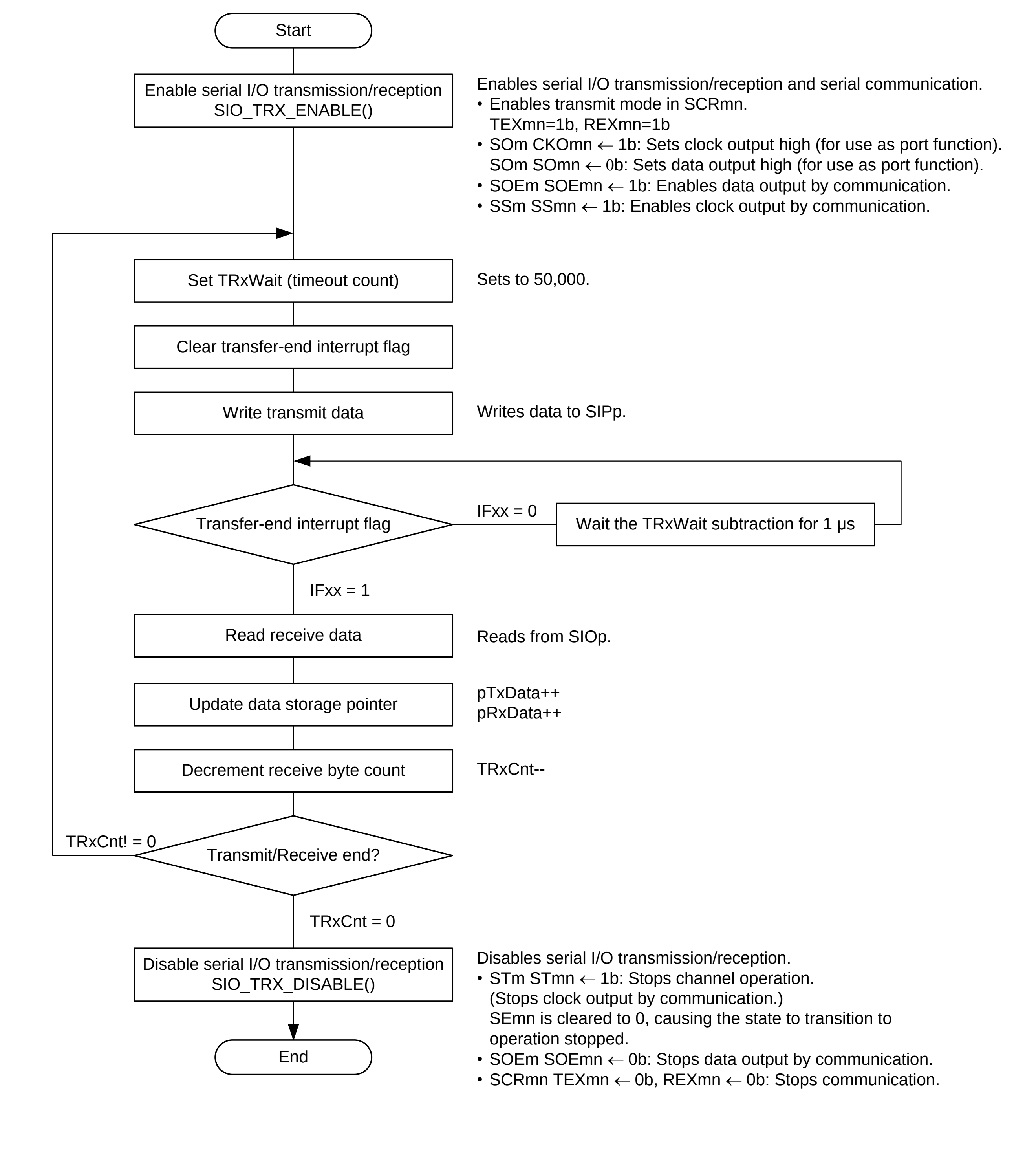


Figure 5.10 Serial I/O Data Reception Processing Outline

5.9 Macro Function Specifications

The macro functions used in this sample code are described below.

5.9.1 Macro Function SIO_IO_INIT()

1. Purpose
   Sets the input pin to the port input state and the output pin to the port output state.
2. Function
   Sets the DataIn pin to the port input state and the DataOut and CLK pins to the port output state.
   Performs the following processing. Review the processing as necessary.
   (1) Sets the DataIn pin to port input.
   (2) Sets the DataOut pin to port high output.
   (3) Sets the CLK pin to port high output.
3. Remarks
   Before executing this function, ensure that the pins can be used as ports.
   When in the serial communication output stopped state, the output pin state is determined by an AND operation according to the setting of the serial output register (SOm) and the output latch setting of the port register (Pxx). Executing this function sets the corresponding port registers (Pxx) to high output, so the output pin states are dependent on the settings of the corresponding CKOmn and SOmn bits in the serial output register (SOm). Before executing this function, execute SIO_DISABLE() and set the corresponding CKOmn and SOmn bits in the serial output register (SOm) to 1 to enable the pins to function as ports.
   To manipulate the registers of the serial array unit, first enable clock supply by setting the appropriate SAUmEN bit in PERn (n corresponds to the register number).

5.9.2 Macro Function SIO_IO_OPEN()

1. Purpose
   Sets the input and output pins to the port input state or output buffer off state.
2. Function
   Sets the DataIn, DataOut, and CLK input pins to the port input state.
   Performs the following processing. Review the processing as necessary.
   (1) Sets the DataIn pin to the port input.
   (2) Sets the DataOut pin to input mode (output buffer off).
   (3) Sets the CLK pin to the port input.
3. Remarks
   Use this function to put all the pins in the Hi-z state before connecting and after disconnecting the removable media.
   Execute SIO_IO_INIT() before executing this function.
   To manipulate the registers of the serial array unit, first enable clock supply by setting the appropriate SAUmEN bit in PERn (n corresponds to the register number).

5.9.3 Macro Function SIO_DATAI_INIT()

1. Purpose
   Sets the DataIn pin to the port input state.
2. Function
   Performs the following processing. Review the processing as necessary.
   (1) In case of the RL78/L1x, sets the DataIn pin to port (other than segment output) by using the LCD port function register (PFSEGx).
   (2) Sets the DataIn pin to noise filter off for CSI mode.
   (3) Sets the DataIn pin to the normal input buffer by using the port input mode register (PIMxx).
   (4) Sets the DataIn pin to the port input by the port mode register (PMxx).
3. Remarks
   It may be necessary to modify the port input mode register (PIMxx) value to match the connected device.
   To manipulate the registers of the serial array unit, first enable clock supply by setting the appropriate SAUmEN bit in PERn (n corresponds to the register number).

5.9.4 Macro Function SIO_DATAO_INIT()

1. Purpose
   Sets the DataOut pin to port high output.
2. Function
   Performs the following processing. Review the processing as necessary.
   (1) In case of the RL78/L1x, sets the DataOut pin to port (other than segment output) by using the LCD port function register (PFSEGx).
   (2) Sets the DataOut pin to the normal output mode by using the port output mode register (POMxx).
   (3) Sets the DataOut pin to port high output by using the port mode register (PMxx) and port register (Pxx).
3. Remarks
   When in the serial communication output stopped state, the output pin state is determined by an AND operation according to the setting of the serial output register (SOm) and the output latch setting of the port register (Pxx). Executing this function sets the corresponding port register (Pxx) to high output, so the output pin state is dependent on the setting of the corresponding SOmn bit in the serial output register (SOm). Before executing this function, execute SIO_DISABLE() and set the corresponding SOmn bit in the serial output register (SOm) to 1 to enable the pin to function as a port.

5.9.5 Macro Function SIO_DATAO_OPEN()

1. Purpose
   Sets the DataOut pin to the port input state.
2. Function
   Performs the following processing. Review the processing as necessary.
   (1) Sets the DataIn pin to the port input state or output buffer off state by using the port output mode register (POMxx).
3. Remarks
   None.

5.9.6 Macro Function SIO_CLK_INIT()

1. Purpose
   Sets the CLK pin to port high output.
2. Function
   Performs the following processing. Review the processing as necessary.
   (1) In case of the RL78/L1x, sets the CLK pin to port (other than segment output) by using the LCD port function register (PFSEGx).
   (2) Sets the DataOut pin to the normal output mode by using the port output mode register (POMxx).
   (3) Sets the CLK pin to port high output by using the port mode register (PMxx) and port register (Pxx).
3. Remarks
   When in the serial communication output stopped state, the output pin state is determined by an AND operation according to the setting of the serial output register (SOm) and the output latch setting of the port register (Pxx). Executing this function sets the corresponding port register (Pxx) to high output, so the output pin state is dependent on the setting of the corresponding SOmn bit in the serial output register (SOm). Before executing this function, execute SIO_DISABLE() and set the corresponding CKOmn bit in the serial output register (SOm) to 1 to enable the pin to function as a port.

5.9.7 Macro Function SIO_CLK_OPEN()

1. Purpose
   Sets the CLK pin to the port input state.
2. Function
   Performs the following processing. Review the processing as necessary.
   (1) Sets the CLK pin to the normal input buffer by using the port input mode register (PIMxx).
   (2) Sets the CLK pin to the port input by using the port mode register (PMxx).
3. Remarks
   It may be necessary to modify the port input mode register (PIMxx) value to match the connected device.

5.9.8 Macro Function SIO_ENABLE()

1. Purpose
   Initializes serial I/O and enables the function. Note that common processing is used up to the point at which transmission, reception, and transmission/reception is enabled. Also sets the baud rate.
2. Function
   Initializes serial I/O according to the hardware manual. Make modifications to the processing as necessary.
   Performs the following processing in the RL78 Family microcontroller.
   (1) Performs common processing to enable transmission and transmission/reception settings.
   - Sets the operating clock in SPSm.
     Sets CKm0. This register can be used to specify two operating clocks (CKm0 and CKm1), so the setting is determined by an OR operation.
   - Sets the operating mode in SMRmn.
     Sets the CKm0 prescaler output clock specified by SPSm in CKSmn.
     Sets the CSI mode in MDmn2 and MDmn1.
     Sets transfer-end interrupt as the interrupt source in MDmn0.
   - Sets the communication format in SCRmn.
     Sets the data and clock phases (DAPmn=0, CKPmn=0: SPI mode 3 compatible) in DAPmn and CKPmn.
     Sets the data transfer sequence (MSB-first) in DIRmn.
     Sets the data length (8 bits) in DLSmn2 to DLSmn0.
   - Sets the baud rate by writing to the operating clock (fMCK) division ratio setting bit field (bits 15 to 9 in SDRmn).
   - Writes 1 to flags FECTmn, PECTmn, and OVCTmn in SIRmn to clear them.
   - Sets SOLmn=0b in SOLm (depends on the channel).
3. Remarks
   This function is the counterpart to SIO_DISABLE(). After executing this function, execute SIO_DISABLE() to end processing.
   SEmn must be cleared to 0 in order to set SPSm, SMRm, and SOLm. Execute SIO_DISABLE() before executing this function.
   CKm0 is used as the operating clock.

5.9.9 Macro Function SIO_DISABLE()

1. Purpose
   Disables the serial I/O function.
2. Function
   Disables the serial I/O function. Performs the common processing to disable transmission and transmission/reception setups. Reconsider the processing as necessary.
   Performs the following processing in the RL78 Family microcontroller.
   (1) Sets the channel to operation stopped mode and switches the pins to function as ports.
   - Sets CKOmn=1b and SOmn=1b in SOm so that the pins function as ports.*[1]
   - Sets STmn=1b in STm.
     → Cleared the SEmn bit to 0 and stopped clock output by serial communication operation.
     → Put the channel into the operation stopped state.
     → The value set in the CKOmn bit in SOm is output from the serial clock output pin.
   - Sets SOEmn=0b in SOEm, stopping data output by serial communication operation.
   - Sets CKOmn=1b and SOmn=1b in SOm so that the pins function as ports.*[2]
   - Sets SOLmn=0b in SOLm (depends on the channel).

   (2) Sets TEXmn=0b and REXmn=0b in SCRmn, setting the operation mode to communication stopped.
   (3) Sets SMRmn to 0020h (value after a reset).
3. Remarks
   This function is the counterpart to SIO_ENABLE(). After executing SIO_ENABLE(), execute this function to end processing.
   SIO_TX_DISABLE() and SIO_TRX_DISABLE() use control by STm to stop communication operation, and this function also uses control by STm to stop communication operation.
   When in the serial communication output stopped state, the output pin state is determined by an AND operation according to the setting of the serial output register (SOm) and the output latch setting of the port register (Pxx). Executing this function sets the corresponding port registers (Pxx) to high output, so the output pin states are dependent on the settings of the corresponding CKOmn and SOmn bits in the serial output register (SOm).
   To manipulate the registers of the serial array unit, first enable clock supply by setting the appropriate SAUmEN bit in PERn (n corresponds to the register number).
   Initially, writing to registers SPSm, SMRm, SDRm, etc., is enabled by clearing SEmn to 0.
   This function is intended to be called during initialization processing and after transmission or reception has ended.

Notes: 1. This function is executed in order to set SOm before stopping clock output by using STm and stopping data output by using SOEm. However, writing to SOm is ignored if the values of both SEmn and SOEmn are 1, so the function's effects depend on the settings of SEmn and SOEmn immediately before it is executed.
Since the effects depend on the preceding state, during initialization SOm is set once again after stopping clock output by using STm and stopping data output by using SOEm (see note 2 below).
When transmission or reception ends, SIO_TX_DISABLE() or SIO_TRX_DISABLE() use control by STm to stop clock output and control by SOEm to stop data output, so the SOm setting can take effect.
2. SOm is set after stopping clock output by using STm and stopping data output by using SOEm.
This ensures that the SOm setting takes effect.

5.9.10 Macro Function SIO_TX_ENABLE()

1. Purpose
   Enables serial I/O transmission.
2. Function
   Enables serial I/O transmission according to the hardware manual. Enables the transmission after switching the pin from the port function to serial I/O function. Reconsider the processing as necessary.
   Performs the initialization procedure for the rest after SIO_ENABLE() and for transmission setting only.
   Performs the following processing in the RL78 Family microcontroller.
   (1) Sets the operating mode to transmission.
   Sets TEXmn=1b and REXmn=0b in SCRmn, enabling transmission.
   (2) Switches the pins to the serial I/O function.
   - Sets data output high and clock output high in SOm, enabling pin output.
   - Sets SOEmn=1b in some, enabling data output by serial communication operation.
     → The values reflected by communication operation are output from the serial data output pin
   (3) Enables serial communication operation.
   Sets SSmn=1b in SSm.
   → The SEmn bit is set to 1, enabling clock output by serial communication.
   → The values reflected by communication operation are output from the serial clock output pin.
3. Remarks
   This function is the counterpart to SIO_TX_DISABLE().After executing this function, execute SIO_TX_DISABLE() to end processing.
   Before executing this function, execute SIO_DISABLE(), SIO_TX_DISABLE(), or SIO_TRX_DISABLE() (each of which use control by STm to stop communication operation) to stop communication operation.
   When in the serial communication output stopped state, the output pin state is determined by an AND operation according to the setting of the serial output register (SOm) and the output latch setting of the port register (Pxx).
   Before executing this function, execute SIO_DISABLE() and SIO_IO_INIT() to set the corresponding CKOmn and SOmn bits in the serial output register (SOm) and the port registers (Pxx) to 1.
   To manipulate the registers of the serial array unit, first enable clock supply by setting the appropriate SAUmEN bit in PERn (n corresponds to the register number).

5.9.11 Macro Function SIO_TX_DISABLE()

1. Purpose
   Disables the serial I/O transmission function.
2. Function
   Disables transmission according to the inverse processing of SIO_TX_ENABLE(). Switches the pin from the serial I/O function to the port function after disabling transmission. Reconsider the processing as necessary.
   Performs the following processing in the RL78 Family microcontroller.
   (1) Sets serial communication to the operation stopped state.
   Sets STmn=1b in STm.
   → Cleared the SEmn to 0 and stopped clock output by serial communication operation.
   → Put the channel into the operation stopped state.
   → The value set in the CKOmn bit in SOm is output from the serial clock output pin.
   (2) Stops output by serial communication operation.
   Sets SOEmn=0b in SOEm, stopping data output by serial communication operation.
   → The value set in the SOmn bit in SOm is output from the serial data output pin.
   (3) Sets the operating mode to communication disabled.
   Sets TEXmn=0b and REXmn=0b in SCRmn, disabling communication.
3. Remarks
   This function is the counterpart to SIO_TX_ENABLE(). After executing SIO_TX_ENABLE(), execute this function to end processing.
   When in the serial communication output stopped state, the output pin state is determined by an AND operation according to the setting of the serial output register (SOm) and the output latch setting of the port register (Pxx).
   Before executing this function, execute SIO_DISABLE() and SIO_IO_INIT() to set the corresponding CKOmn and SOmn bits in the serial output register (SOm) and the port registers (Pxx) to 1.
   To manipulate the registers of the serial array unit, first enable clock supply by setting the appropriate SAUmEN bit in PERn (n corresponds to the register number).

5.9.12 Macro Function SIO_TRX_ENABLE()

1. Purpose

   Enables serial I/O transmission/reception.

2. Function

   Enables serial I/O transmission/reception according to the hardware manual. Enables the transmission/reception after switching the pin from the port function to serial I/O function. Reconsider the processing as necessary.

   Performs the initialization procedure for the rest after SIO_ENABLE() and for transmission/reception setting only.

   Performs the following processing in the RL78 Family microcontroller.

   (1) Sets the operating mode to transmission/reception.
   Sets TEXmn=1b and REXmn=1b in SCRmn, enabling transmission/reception.

   (2) Switches the pins to the serial I/O function.
   - Sets data output high and clock output high in SOm, enabling pin output.
   - Sets SOEmn=1b in some, enabling data output by serial communication operation.
     → The values reflected by communication operation are output from the serial data output pin

   (3) Enables serial communication operation.
   Sets SSmn=1b in SSm.
   → The SEmn bit is set to 1, enabling clock output by serial communication.
   → The values reflected by communication operation are output from the serial clock output pin.

3. Remarks

   This function is the counterpart to SIO_TRX_DISABLE().After executing this function, execute SIO_TRX_DISABLE() to end processing.

   Before executing this function, execute SIO_DISABLE(), SIO_TX_DISABLE(), or SIO_TRX_DISABLE() (each of which use control by STm to stop communication operation) to stop communication operation.

   When in the serial communication output stopped state, the output pin state is determined by an AND operation according to the setting of the serial output register (SOm) and the output latch setting of the port register (Pxx).

   Before executing this function, execute SIO_DISABLE() and SIO_IO_INIT() to set the corresponding CKOmn and SOmn bits in the serial output register (SOm) and the port registers (Pxx) to 1.

   To manipulate the registers of the serial array unit, first enable clock supply by setting the appropriate SAUmEN bit in PERn (n corresponds to the register number).

5.9.13 Macro Function SIO_TRX_DISABLE()

1. Purpose
   Disables the serial I/O transmission/reception function.
2. Function
   Disables transmission/reception according to the inverse processing of SIO_TRX_ENABLE(). Switches the pin from the serial I/O function to the port function after disabling transmission/reception. Reconsider the processing as necessary.
   Performs the following processing in the RL78 Family microcontroller.
   (1) Sets serial communication to the operation stopped state.
   Sets STmn=1b in STm.
   → Cleared the SEmn to 0 and stopped clock output by serial communication operation.
   → Put the channel into the operation stopped state.
   → The value set in the CKOmn bit in SOm is output from the serial clock output pin.
   (2) Stops output by serial communication operation.
   Sets SOEmn=0b in SOEm, stopping data output by serial communication operation.
   → The value set in the SOmn bit in SOm is output from the serial data output pin.
   (3) Sets the operating mode to communication disabled.
   Sets TEXmn=0b and REXmn=0b in SCRmn, disabling communication.
3. Remarks
   This function is the counterpart to SIO_TRX_ENABLE(). After executing SIO_TRX_ENABLE(), execute this function to end processing.
   When in the serial communication output stopped state, the output pin state is determined by an AND operation according to the setting of the serial output register (SOm) and the output latch setting of the port register (Pxx). Before executing this function, execute SIO_DISABLE() and SIO_IO_INIT() to set the corresponding CKOmn and SOmn bits in the serial output register (SOm) and the port registers (Pxx) to 1.
   To manipulate the registers of the serial array unit, first enable clock supply by setting the appropriate SAUmEN bit in PERn (n corresponds to the register number).

5.10 State Transition Diagram

Figure 5.11 shows the state transition diagram. Do not perform serial transmission or reception before the serial I/O function has been initialized. For details, see 7.6, Prohibition of Data Transmission and Reception.

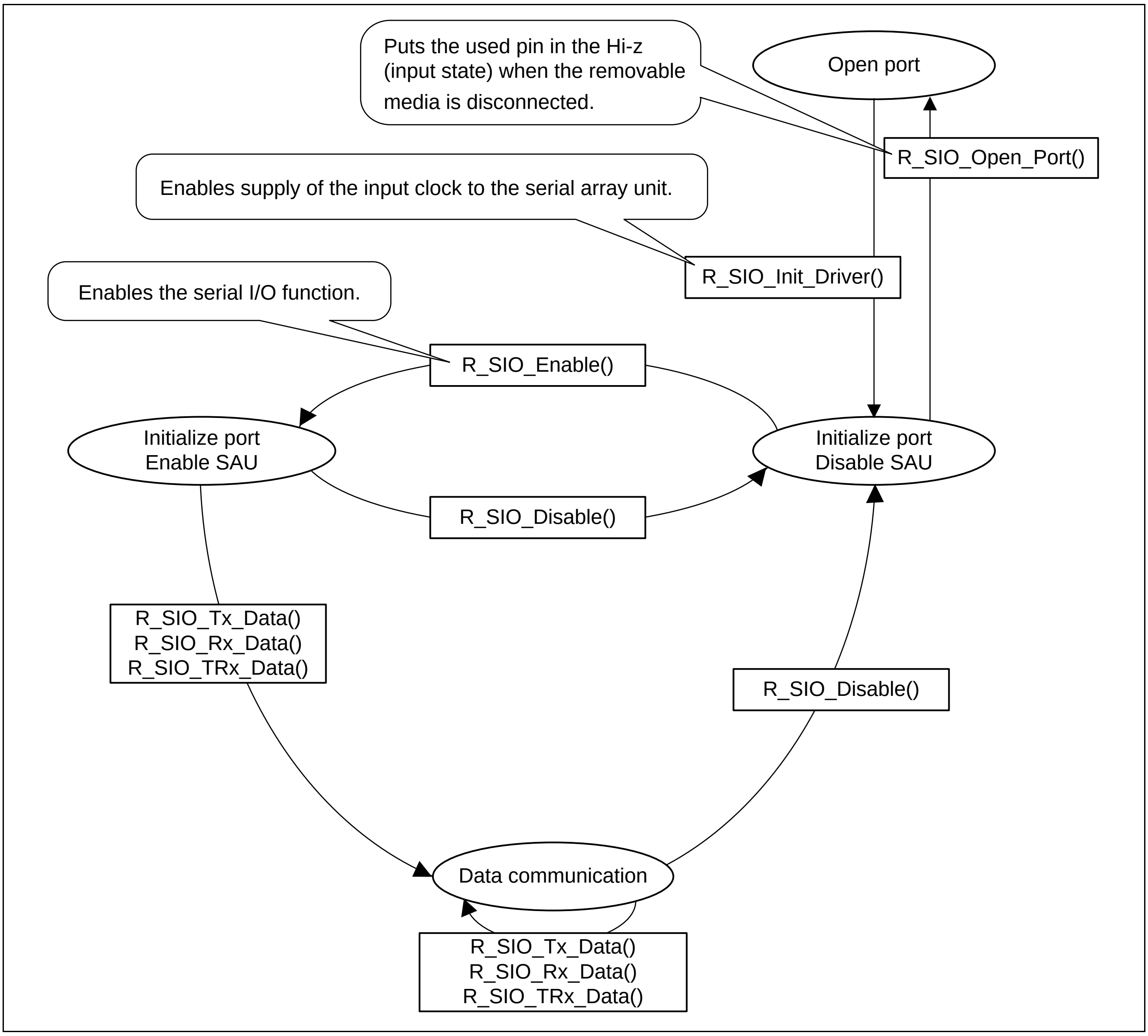


Figure 5.11 State Transition Diagram

6. Application Example

This section gives an example of settings for the serial I/O control section.

Examples of the settings for usage are given below.

The locations where settings are made are identified by the comments header "/** SET **/" in the defining file.

6.1 mtl_com.h (common header file)

This is the header file for functions to be in common use.

Each mtl_com.h.XXX (excluding mtl_com.h.common) is made for the evaluation of a given MCU. Use the appropriate header file after renaming it mtl_com.h. If there is no header file for the MCU to be evaluated, make mtl_com.h with reference to mtl_com.h.XXX.

(1) Defining the Header Files for the OS

This sample code is independent of the OS.

In the example given below, the OS is not to be used.

That is, the settings in the sample code are for when the OS is not to be used, so the code is independent of the OS. This sample code does, however depend on other software.

```
/* In order to use wai_sem/sig_sem/dly_tsk for microITRON (Real-Time OS)-compatible, */
/* include the OS header file that contains the prototype declaration.       */
/* When not using the OS, put the following 'define' and 'include' as comments.      */
//#define MTL_OS_USE                    /* Use OS                                  */
//#include <RTOS.h>                     /* OS header file                          */
//#include "mtl_os.h"
```

(2) Defining the Header File with the Common Access Area Defined

It is possible to include a header file of MCU function register definitions. The main reason it would be necessary to include this header file is to enable port control, etc., by the device driver. The RL78 uses a different method to make these definitions, so the header file should be commented out in the sample code.

In the example below, the header file is not included.

```
/* In order to use definitions of MCU SFR area,                            */
/* include the header file of MCU SFR definition.                          */
//#include "iodefine.h"                      /* definition of MCU SFR      */
```

(3) Defining the Loop Timer

The following header file is included so that the software loop timer is available for use.

This is used to secure waiting time for the device driver.

Comment out the "#include" directive if the software loop timer is not to be used.

The software loop timer is to be used in this example.

This header file must be included if the sample code is to be used.

```
/* When not using the loop timer, put the following 'include' as comments.   */
#include "mtl_tim.h"
```

(4) Defining the Endian Mode

Either little-endian or big-endian mode can be specified.

For the RL78 Family microcontroller, define the endian mode as little-endian.

```
/* When using M16C or SuperH for Little Endian setting, define it.            */
/* When using other MCUs, put 'define' as a comment.                          */
#define MTL_MCU_LITTLE                /* Little Endian                        */
```

(5) Defining High-Speed Endian Processing

High-speed processing by mtl_end.c can be specified. Processing becomes high-speed if the M16C is in use.

In the case of the RL78 Family microcontroller, leave this commented out so that the definition is not made.

```
/* When using M16C, define it.                                                */
/* It performs the fast processes of 'mtl_endi.c'.                            */
//#define MTL_ENDI_HISPEED            /* Uses the high-speed function.        */
```

(6) Defining the Standard Library to Be Used

Define the type of standard library to be used.

Leave the "#define" below commented out if the library attached to the compiler is to handle the indicated processing.

The library attached to the compiler is to be used in the example below.

```
/* Specify the type of user standard library.                                 */
/* When using the compiler-bundled library for the following processes,       */
/* put the following 'define' as comments.                                    */
/* memcmp() / memmove() / memcpy() / memset() / strcat() / strcmp() / strcpy() / strlen()     */
//#define MTL_USER_LIB                /* use optimized library                */
```

(7) Defining the RAM Area to Be Accessed

Define the RAM area to be accessed.

This obtains more efficient processing by standard functions and some other processes.

Define MTL_MEM_NEAR in the case of the RL78 Family microcontroller.

```
/* Define the RAM area to be accessed by the user process.                    */
/* Efficient operations for standard functions and processes are applied.     */
//#define MTL_MEM_FAR    /* Defines 'FAR' as 'far' attribute for RAM area.(For M16C Family)*/
#define MTL_MEM_NEAR     /* No far/near attribute for RAM area.               */
```

6.1.1 mtl_tim.h

This is included by the include directive for the loop timer in mtl_com.h.

The effects of the settings depend on the MCU, clock, and compiler options in use.

If the system is cache-equipped, make settings on the assumption that the instruction cache is enabled and that the code for loop-timer processing is stored in the cache.

Repeat measurement and adjust the settings according to the conditions of usage.

```
/* Define the counter value for the timer.                                      */
/* Specify according to the user MCU, clock and wait requirements.              */
#if 1
/* Setting for 32MHz no wait (Compile Option : "-qx" at CubeSuite+ V1.01.01a,
CA78K0R Ver.1.30)*/
#define MTL_T_1US                          4    /* loop Number of   1 us   */
#define MTL_T_2US                          8    /* loop Number of   2 us   */
#define MTL_T_4US                         16    /* loop Number of   4 us   */
#define MTL_T_5US                         20    /* loop Number of   5 us   */
#define MTL_T_10US                        40    /* loop Number of  10 us   */
#define MTL_T_20US                        80    /* loop Number of  20 us   */
#define MTL_T_30US                       120    /* loop Number of  30 us   */
#define MTL_T_50US                       200    /* loop Number of  50 us   */
#define MTL_T_100US                      400    /* loop Number of 100 us   */
#define MTL_T_200US                      800    /* loop Number of 200 us   */
#define MTL_T_300US                     1200    /* loop Number of 300 us   */
#define MTL_T_400US      ( MTL_T_200US  *   2 ) /* loop Number of 400 us   */
#define MTL_T_1MS                        000    /* loop Number of   1 ms   */
#endif
```

Times for the above values have not been measured, so the settings are not necessarily appropriate. Perform evaluation as required.

6.2 Setting up the Control Software for Clock Synchronous Single Master Operation

The locations where settings are made are identified by the comments header "/** SET **/" in the defining file.

6.2.1 R_SIO.h

(1) Defining the Wait Time after Setting Up the BRR

Setting the BRR of the SAU is followed by a software wait until one bit of data is transferred. Set this wait time as required.

The default setting is for 10 μs.

Supposing transfer at 100 kHz and usage with Multimedia Cards, make the setting for 10 μs.

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

The RL78 Family microcontroller does not require any wait after setting BRR. Since no wait processing is specified in the sample code, this line is ignored.

6.2.2 R_SIO_csi.h

This is the definition file for the SAU.

Each R_SIO_csi.h.XXX is made for the evaluation of a given MCU. Use the appropriate header file after renaming it R_SIO_csi.h. If there is no header file for the MCU to be evaluated, make R_SIO_csi.h with reference to the R_SIO_csi.h.XXX files.

(1) Defining the Operating Mode to Be Used

The resources of the MCU to be used can be set.

If processing is to be of MSB-first CRC-CCITT calculations, specify SIO_OPTION_2 as in the following example.

CRC-CCITT calculations are unnecessary when control is of serial EEPROM or serial Flash memory. In such cases, comment the definition out.

The separate R_SIO_csi_rx_mmc.c file is needed to perform CRC-CCITT calculations for controlling Multimedia Cards.

```
/*-------------------------------------------------------------------------- */
/*  Define the combination of the MCU's resources.                           */
/*-------------------------------------------------------------------------- */
#define SIO_OPTION_1       /* Low speed*/ /* SI/O                            */
//#define SIO_OPTION_2     /*          */ /* SI/O   + CRC calculation (S/W)  */
```

(2) Defining the Form of CRC Calculation to Be Used

Define the form of CRC calculation to be used.

CRC-CCITT calculation is not used when control is of serial EEPROM or serial Flash memory. In such cases, comment the definition out.

To control multimedia cards, define both CRC-CCITT calculation and CRC-CCITT calculation at the same time.

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

(3) Defining the Pins to Be Used

Define the pins to be used.

The following example is for the CubeSuite+ integrated development environment (Renesas Electronics Corporation RL78, 78K0R compiler, CA78K0R).

```
/*----------------------------------------------------------------------------*/
/*  Define the control port.                                                  */
/*  Delete comment of a related macrodefinition, and please validate setting. */
/*----------------------------------------------------------------------------*/
#define SIO_PM_DATAO    PM14.4      /* SIO DataOut                            */
#define SIO_PM_DATAI    PM14.3      /* SIO DataIn                             */
#define SIO_PM_CLK      PM14.2      /* SIO CLK                                */
#define SIO_P_DATAO     P14.4       /* SIO DataOut                            */
#define SIO_P_DATAI     P14.3       /* SIO DataIn                             */
#define SIO_P_CLK       P14.2       /* SIO CLK                                */
#define SIO_PIM_DATAI   PIM14.3        /* SIO DataIn                          */
#define SIO_PIM_CLK     PIM14.2        /* SIO CLK                             */
#define SIO_POM_DATAO   POM14.4        /* SIO DataOut                         */
#define SIO_POM_CLK     POM14.2        /* SIO CLK                             */
```

(4) Defining the Peripheral Enable Register

Specify the peripheral enable register related to the SAU to be used.

```
#define SIO_SAUEN       SAU1EN      /* Control of CSI30 input clock supply   */
```

(5) Defining the CSI Channel to Be Used

Specify the used CSI channel. The following example is for using CS130 with the CubeSuite+ integrated development environment (Renesas Electronics Corporation RL78, 78K0R compiler, CA78K0R).

```
/*----------------- SIO definitions -----------------*/

/* CSI30 setting example - Set the following for the system. */
#define SIO_SPS      SPS1     /* Serial clock select register               */
#define SIO_SMR      SMR12    /* Serial mode register                       */
#define SIO_SCR      SCR12    /* Serial communication operation setting register*/
#define SIO_SDR      SDR12    /* Serial data register                       */
#define SIO_TXBUF    SIO30    /* SIOp data register                         */
#define SIO_RXBUF    SIO30    /* SIOp data register                         */
#define SIO_SIR      SIR12    /* Serial flag clear trigger register         */
#define SIO_SSR      SSR12    /* Serial status register                     */
#define SIO_SS       SS1L.2   /* Serial channel start register SSmn         */
#define SIO_ST       ST1L.2   /* Serial channel stop register STmn          */
#define SIO_SE       SE1L.2   /* Serial channel enable status register SEmn */
#define SIO_SOE      SOE1L.2  /* Serial output enable register SOEmn        */
#define SIO_SO       SO1      /* Serial output register SOmn                */
#define SIO_SOL      SOL1     /* Serial output level register               */
#define SIO_SNFEN    NFEN0.6  /* Use of noise filter of RXD pin SNFEN       */

#define SIO_TXNEXT   (SSR12L & 0x20)/* CSI Transmit data empty              */
#define SIO_RXNEXT   IF1H.4   /* CSI Receive completion                     */
#define SIO_TXEND    IF1H.4   /* CSI Transmit completion                    */
```

(6) Defining the Operating Clock to Be Used in the Serial Clock Select Register (SPSm)

Specify the operating clock selection in the serial clock select register (SPSm). CKm0 is used in the example below.

```
#define SIO_USPS_INIT      (uint16_t)0x0000
                  /* 0000000000000000B */ /* SPS CSI initial setting       */
                  /* ||||||||||||++++-- CKm0:No division of fclk         */
                  /* ||||||||++++------ CKm1:No division of fclk         */
                  /* ++++++++---------- Reserved      : 0   Fixed        */
```

(7) Defining the Operating Clock (fMCK) Selection for the Channel to Be Used

Specify the operating clock (fMCK) selection used for the CKSmn bit in the serial mode register (SMRmn). CKm0 is used in the example below.

```
#define SIO_USMR_INIT      (uint16_t)0x0020
            /* 0000000000100000B */ /* SMR CSI initial setting             */
            /* ||||||||||||||+-- Interrupt source : Transfer end interrupt */
            /* |||||||||||||++--- Operation mode : CSI mode               */
            /* |||||||||||++----- Reserved         : 0      Fixed         */
            /* ||||||||||+------- Reserved         : 1      Fixed         */
            /* |||||||||+-------- Reserved (Controls in UART mode)        */
            /* ||||||||+--------- Reserved         : 0      Fixed         */
            /* |||||||+---------- Start trigger source : Software trigger  */
            /* ||+++++----------- Reserved         : 0      Fixed         */
            /* |+---------------- ftclk clock channel setting : Divided fmck */
            /* +----------------- fMCK clock channel setting : CKm0 set    */
```

(8) Defining the Serial Output Value

Set to 1 the serial output register for the channel to be used.

To accomplish this, set to 1 the SOmn bit in the serial output register (SOm) of the channel to be used. This sets to 1 the SOmn bit corresponding to the location set to 1. The setting used for the CSI01 data output pin and clock output pin is shown in the example below.

```
#define SIO_USO_INIT       (uint16_t)0x0404                               */
            /* 0000010000000100B */ /* SO0 initial setting                 */
            /* ||||||||||||||+-- SOm0 output                               */
            /* |||||||||||||+--- SOm1 output                               */
            /* ||||||||||||+---- SOm2 output                               */
            /* |||||||||||+----- SOm3 output                               */
            /* ||||||||++++------ Reserved      : 0   Fixed                */
            /* |||||||+---------- CKOm0 output                             */
            /* ||||||+----------- CKOm1 output                             */
            /* |||||+------------ CKOm2 output                             */
            /* ||||+------------- CKOm3 output                             */
            /* ++++-------------- Reserved      : 0                        */
```

(9) Defining the Serial Output Level Register (SOLm)

In CSI mode the inversion setting is prohibited. Set 1 in order to write 0 to the relevant SOLmn bit and reserved bits.

The reason for setting 1 is to contain the operation of writing 0 to the point that is set to 1 in the sample code,

The setting when CSI30 is used is shown in the example below.

```
#define SIO_USOL_INIT       (unit16_t)0xFFFE
         /* 1111111111111110B */ /* SOLm initial setting(CSI mode setting) */
         /* |||||||||||||||+-- SOLm0 Communication data is output :        */
         /* ||||||||||||||+--- Reserved    : 1      Fixed                  */
         /* |||||||||||||+---- SOLm2 Communication data is output :        */
         /* +++++++++++++----- Reserved    : 1      Fixed                  */

         /* Caution: Refer to the application note for Setting method.     */
         /*          Set Unit/Channel No. and reserved bit to use to 1.    */
         /*          Because 0 is written to a register by setting 1.      */
```

(10) Defining the Port Input Mode Register (PIM) and the Port Output Mode Register (POM)

According to the pin to be used, specify PIM and POM.

```
/*------------------- DataIn control -------------------*/
#define SIO_DATAI_INIT() do {           /* DataIn initial setting        */  \
    SIO_SNFEN        = 0;               /* Noise filter OFF              */  \
    SIO_PIM_DATAI    = 0; /** SET **/   /* Normal input buffer           */  \
    SIO_PM_DATAI     = 1;               /* DataIn           Input        */  \
} while (0)


/*------------------ DataOut control -------------------*/
#define SIO_DATAO_INIT() do {           /* DataOut initial setting       */  \
    SIO_POM_DATAO    = 0; /** SET **/   /* Normal output mode            */  \
    SIO_P_DATAO      = SIO_HI;          /* DataOut          "H"          */  \
    SIO_PM_DATAO     = 0;               /* DataOut          Output       */  \
    SIO_P_DATAO      = SIO_HI;          /* DataOut          "H"          */  \
} while (0)

#define SIO_DATAO_OPEN() do {           /* DataOut open setting          */  \
    SIO_PM_DATAO     = 1;               /* DataOut          Input        */  \
} while (0)


/*-------------------- CLK control --------------------*/
#define SIO_CLK_INIT() do {             /* CLK initial setting           */  \
    SIO_POM_CLK      = 0; /** SET **/   /* Normal output mode            */  \
    SIO_P_CLK        = SIO_HI;          /* CLK              "H"          */  \
    SIO_PM_CLK       = 0;               /* CLK              Output       */  \
    SIO_P_CLK        = SIO_HI;          /* CLK              "H"          */  \
} while (0)

#define SIO_CLK_OPEN() do {             /* CLK open setting              */  \
    SIO_PIM_CLK      = 0; /** SET **/   /* Normal input buffer           */  \
    SIO_PM_CLK       = 1;               /* CLK              Input        */  \
} while (0)
```

6.3 R_SIO_csi.c

An example of usage settings is shown below.

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

6.4 Setting the definition of SFR

There will be predefined preprocessor symbols in the C compiler used. The program is coded using these predefined preprocessor symbols.

Also, if the IAR Systems integrated development environment is used, it will be necessary to set the header file in which the SFRs for the microcontroller used are defined.

Table 6.1 Microcontroller and SFR Area Define Settings

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

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

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

7. Usage Notes

7.1 Usage Notes to be Observed when Building the Sample Code

To incorporate the sample code, include R_SIO.h and R_SIO_csi.h (after renaming R_SIO_csi.h.XXX).

7.2 Unnecessary Functions

Unused functions waste ROM capacity, so we recommend excluding them by commenting them out and so on.

7.3 Using Other MCUs

Other MCUs can easily be used.

The files to be prepared are as follows:

- A common I/O module definition file corresponding to R_SIO_csi.h.XXX
- A header definition file corresponding to mtl_com.h.XXX

Make them by referring the attachment.

7.4 Port Control for Serial Data and Clock Output Pins

To set these pins to function as ports, set to 1 the CKOmn and SOmn bits in the serial output register (SOm). The output from these pins is determined by an AND operation using the serial output register (SOm) setting and the output latch setting of the corresponding port registers (Pxx). When the CKOmn bit and SOmn bit are set to 1, the unmodified port register (Pxx) setting value becomes the output value of the corresponding pin.

7.5 Enabling/Disabling Clock Supply to the Serial Array Unit

In the sample code, supply of the clock is started by the serial I/O enable setting processing (R_SIO_Enable()), but no control over stopping the clock is provided by the serial I/O disable setting processing (R_SIO_Disable()). This is because it is assumed that other programs may be using the other channels of the same unit.

Therefore, the user should provide additional program code with the necessary control functions if there is a need to stop operation of individual units in order to reduce power consumption and noise, taking into account the control of channels other than the one used by the sample code.

Note that the sample code does provide the capability to stop operation of the channel used by the application.

7.6 Prohibition of Data Transmission and Reception

Do not perform serial data transmission or reception if the serial I/O function has not been enabled.

In the sample code, supply of the clock to the serial array unit starts when driver initialization processing (R_SIO_Init_Driver()) is performed. Executing serial I/O data transmit processing (R_SIO_Tx_Data()) or serial I/O data receive processing (R_SIO_Rx_Data()) in this state will cause transmission or reception processing to start even though the correct register settings for the serial I/O function have not been completed. It is not possible for transmission or reception processing to proceed properly in this state because the register settings for items such as the baud rate are not correct.

To perform serial I/O data transmit processing (R_SIO_Tx_Data()) or serial I/O data receive processing (R_SIO_Rx_Data()), first execute serial I/O enable setting processing (R_SIO_Enable()) to make the necessary register settings related to serial I/O. Also refer to 5.10, State Transition Diagram.

7.7 Setting Serial Output Level Register (SOLm)

In CSI mode the inversion setting is prohibited. Set 1 in order to write 0 to the relevant SOLmn bit and reserved bits.

The reason for setting 1 is to contain the operation of writing 0 to the point that is set to 1 in the sample code,

Also refer to 6.2.2 (9), Defining the Serial Output Level Register (SOLm).

7.8 About Warnings of Duplicate Type Declaration

This driver has declared the intN_t and uintN_t that are declared in the "stdint.h". There is a possibility that the warning occurs when including the "stdint.h". If the type of declaration is unnecessary, delete the declaration of this driver.

Website and Support

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

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

Revision History

Rev.	Date	Description: Page	Description: Summary
1.02	Aug.31, 2012	—	First edition issued
1.04	Apr. 30, 2014	—	The Product name was 'RL78/G14 Group'. The Application Note Number, Date and Revision are changed.
		1	The MCU name in Introduction was 'RL78/G14 Group'.
		1	Added the combination information URL of the latest slave device control software.
		1	The Target Device was 'RL78/G14 Group'.
		1	Added the following. Note that the term "RL78 Family microcontroller" is used in this document for ease of description since the target devices come from multiple groups.
		3-	The 'RL78 Family microcontroller' was 'RL78/G14 Group'.
		4	Added the following title to section 2. (1) RL78/G14 SAU Integrated Development Environment CubeSuite+
		4-8	Added the following conditions to section 2. (2) RL78/G14 SAU Integrated Development Environment IAR Embedded Workbench (3) RL78/G1C SAU Integrated Development Environment CubeSuite+ (4) RL78/G1C SAU Integrated Development Environment IAR Embedded Workbench (5) RL78/L12 SAU Integrated Development Environment CubeSuite+ (6) RL78/L12 SAU Integrated Development Environment IAR Embedded Workbench (7) RL78/L13 SAU Integrated Development Environment CubeSuite+ (8) RL78/L13 SAU Integrated Development Environment IAR Embedded Workbench (9) RL78/L1C SAU Integrated Development Environment CubeSuite+ (10) RL78/L1C SAU Integrated Development Environment IAR Embedded Workbench
		9	The following added to section 3. • Micron Technology P5Q Serial Phase Change Memory Control Software (R01AN1439EJ) • Micron Technology N25Q Serial NOR Flash Memory Control Software (R01AN1528EJ) • Spansion S25FLxxxS MirrorBit® Flash Non-Volatile Memory Control Software (R01AN1529EJ)
		12	Section 5.1.1 and 5.1.2 The 'RL78 Family microcontroller' was 'RL78/G14'.
		15	Section 5.3 'The sizes of the required memory areas for each MCU of different instructions are given below. Investigate the instructions of MCU to be used and give by reference. See chapter 2, Conditions for Checking the Operation of the Software, for the environment.' was 'The sizes of the required memory areas are given below.'.
		15	Added the following title to section 5.3. (1) RL78/G14 SAU Integrated Development Environment CubeSuite+

		15	Added the following sizes to section 5.3. (2) RL78/G14 SAU Integrated Development Environment IAR Embedded Workbench (3) RL78/L13 SAU Integrated Development Environment CubeSuite+ (4) RL78/L13 SAU Integrated Development Environment IAR Embedded Workbench
		16	Section 5.4 Changed Application Note Number. Changed Folder names. 'R_SIO_csi.h.rl78g14' was 'R_SIO_csi.h.rl78'. Added the following. R_SIO_csi.h.rl78g1c R_SIO_csi.h.rl78l12 R_SIO_csi.h.rl78l13 R_SIO_csi.h.rl78l1c
		30	1.Function of sections 5.9.3 Added (1).
		31	1.Function of sections 5.9.4 Added (1).
		32	1.Function of sections 5.9.6 Added (1).
		36	Changed content of sections 6.2.2 (5).
		46	Added Section 6.3, R_SIO_csi.c.
		48	Original section 7.4, Method of Manipulating SFR Area, removed.
1.05	Mar. 31, 2016	5	Section 2 Changed the following conditions. (1) RL78/G14 SAU Integrated Development Environment CS+ for CA,CX (Compiler: CA78K0R) Added the following conditions. (2) RL78/G14 SAU Integrated Development Environment CS+ for CC (Compiler: CC-RL)
		16	Section 5.2 Added the following. 5.2.7 Data Transmission/ Reception (R_SIO_TRx_Data())
		17	Section 5.3 Changed the following sizes. (1) RL78/G14 SAU Integrated Development Environment CS+ for CA,CX (Compiler: CA78K0R) Added the following sizes. (2) RL78/G14 SAU Integrated Development Environment CS+ for CC (Compiler: CC-RL)
		19	Changed the following table to Section 5.4.
		21	5.7 List of Functions Added function R_SIO_TRx_Data().
		32	Added the following to Section 5.8. 5.8.7 Serial I/O Data Transmit/Receive Processing
		43	Changed the following diagrams to Section 5.10.
		53	Section 7 Added the following. 7.8 About Warnings of Duplicate Type Declaration

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

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

1. Handling of Unused Pins

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

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

2. Processing at Power-on

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

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

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

4. Clock Signals

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

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

5. Differences between Products

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

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

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