78K0R/Kx3-L
Clock Synchronous Single Master Control Software
Using CSI Mode of Serial Array Unit

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 78K0R/KE3-L 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 as well.

Target Device
Corresponding MCU: 78K0R/KE3-L
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.

Contents

1. Specifications ................................................................................................................................. 2
2. Conditions of Checking the Operation of the Software .............................................................. 3
3. Related Application Notes ........................................................................................................... 3
4. Hardware Description .................................................................................................................. 4
5. Software Description .................................................................................................................... 5
6. Application Example ................................................................................................................... 31
7. Usage Notes ............................................................................................................................... 38
1. Specifications

This software program uses the 3-wire serial I/O communications (CSI mode) of the serial array unit (SAU) of the 78K0R/KE3-L 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 78K0R/KE3-L 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

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

![Figure 1.1 Sample Configuration](image-url)
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.

Table 2.1 Operating Conditions

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcomputer used for evaluation</td>
<td>78K0R/KE3-L (program ROM 64 KB/RAM 3 KB)</td>
</tr>
<tr>
<td>Memory used for evaluation</td>
<td>Renesas Electronics R1EX25xxx Series SPI serial EEPROM</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>Main system clock: 20 MHz</td>
</tr>
<tr>
<td></td>
<td>CPU/peripheral hardware clock: 20 MHz</td>
</tr>
<tr>
<td></td>
<td>Serial clock: 2.5 MHz</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>3.3 V</td>
</tr>
<tr>
<td>Integrated development environment</td>
<td>Renesas Electronics Project manager (PM+ Ver. 6.31)</td>
</tr>
<tr>
<td>C compiler, assembler</td>
<td>Renesas Electronics 78K0R C compiler (CC78K0R Ver. 2.13)</td>
</tr>
<tr>
<td></td>
<td>78K0R assembler package (RA78K0R Ver. 1.33)</td>
</tr>
<tr>
<td></td>
<td>Compiler options: The default settings (-a. -zp) for the integrated</td>
</tr>
<tr>
<td></td>
<td>development environment are used.</td>
</tr>
<tr>
<td>Emulator</td>
<td>Minicube2</td>
</tr>
<tr>
<td>Integrated debugger</td>
<td>Renesas Electronics ID78K0R-QB integrated debugger, Ver. 3.61</td>
</tr>
<tr>
<td>Version of the sample code</td>
<td>Ver.2.02</td>
</tr>
<tr>
<td>Software used for evaluation</td>
<td>Renesas Electronics The R1EX25xxx Series' SPI serial EEPROM control software,</td>
</tr>
<tr>
<td></td>
<td>Ver. 2.01</td>
</tr>
<tr>
<td>Evaluation board used</td>
<td>78K0R/KE3-L Target board (QB-78K0RKE3L-TB)</td>
</tr>
</tbody>
</table>

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)
4. Hardware Description

4.1 List of Pins

Table 4.1 lists the pins that are used and their uses.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCK (CLK of figure 4.1)</td>
<td>Output</td>
<td>Clock output</td>
</tr>
<tr>
<td>SO (DataOut of figure 4.1)</td>
<td>Output</td>
<td>Master data output</td>
</tr>
<tr>
<td>SI (DataIn of figure 4.1)</td>
<td>Input</td>
<td>Master data input</td>
</tr>
<tr>
<td>Port (Port(CS#) of figure 4.1)</td>
<td>Output</td>
<td>Slave device select output</td>
</tr>
</tbody>
</table>

Note: Port (CS#) of figure 4.1 is 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 an 78K0R/Kx3-L 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](image-url)

<table>
<thead>
<tr>
<th>Byte offset value on the device</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  1  ...</td>
</tr>
<tr>
<td>508 509 510 511</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte offset value in memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  1  ...</td>
</tr>
<tr>
<td>508 509 510 511</td>
</tr>
</tbody>
</table>
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 78K0R/KE3-L must be set for type 1 (DAPmn=0, CKPmn=0).

<table>
<thead>
<tr>
<th>CLK</th>
<th>DataOut</th>
<th>DataIn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D7</td>
<td>D7</td>
</tr>
<tr>
<td></td>
<td>D6</td>
<td>D6</td>
</tr>
<tr>
<td></td>
<td>D5</td>
<td>D5</td>
</tr>
<tr>
<td></td>
<td>···</td>
<td>···</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D0</td>
</tr>
</tbody>
</table>

- 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.

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 78K0R/KE3-L. This enables the SPI slave device to be controlled by using general port output from the 78K0R/KE3-L.

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.

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.

5.3 Sizes of Required Memory
Table 5.1 lists the sizes of the required memory areas.

Table 5.1 Sizes of Required Memory

<table>
<thead>
<tr>
<th>Memory Used</th>
<th>Size</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM</td>
<td>527 bytes</td>
<td>R_SIO_csi.c</td>
</tr>
<tr>
<td>RAM</td>
<td>0 bytes</td>
<td>R_SIO_csi.c</td>
</tr>
<tr>
<td>Maximum user stack size</td>
<td>30 bytes</td>
<td></td>
</tr>
<tr>
<td>Maximum interrupt stack size</td>
<td>—</td>
<td>No interrupts are used.</td>
</tr>
</tbody>
</table>

Note: The sizes of required memory areas vary with the version and compiler options of the C compiler. The above-mentioned memory sizes vary with the endian mode adopted. The maximum user stack size value shown is that when using the serial EEPROM control software and includes the stack of the serial EEPROM control software.
5.4 File Configuration

Table 5.2 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 5.2 File Configuration**

<table>
<thead>
<tr>
<th>Directory Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\an_r01an0708ej_78k0rkx3l</td>
<td>Folder for the sample code</td>
</tr>
<tr>
<td>r01an0708ej0102_78k0rkx3l.pdf</td>
<td>Application note</td>
</tr>
<tr>
<td>\source</td>
<td>Folder for storing the programs</td>
</tr>
<tr>
<td>\com*1</td>
<td>Folder for storing the common functions</td>
</tr>
<tr>
<td>\com*1</td>
<td>Miscellaneous common function definitions</td>
</tr>
<tr>
<td>mtl_com.c</td>
<td>Common header file</td>
</tr>
<tr>
<td>mtl_com.h.common</td>
<td>Common function header file</td>
</tr>
<tr>
<td>mtl_com.h.78K0R</td>
<td>Common file (related to endian setting)</td>
</tr>
<tr>
<td>mtl_endi.c</td>
<td>Common file (standard library function)</td>
</tr>
<tr>
<td>mtl_mem.c</td>
<td>Common file (standard library function)</td>
</tr>
<tr>
<td>mtl_os.c mtl_os.h</td>
<td>Common file (related to loop timer)</td>
</tr>
<tr>
<td>mtl_str.c</td>
<td>Sample for setting the value in the loop timer</td>
</tr>
<tr>
<td>mtl_tim.c mtl_tim.h</td>
<td>Sample for setting the value in the loop timer</td>
</tr>
<tr>
<td>\r_sio_csi_78k0r</td>
<td>Folder for clock synchronous single master control software using the SCI for the 78K0R</td>
</tr>
<tr>
<td>R_SIO.h</td>
<td>Header file</td>
</tr>
<tr>
<td>R_SIO_csi.c</td>
<td>I/F module</td>
</tr>
<tr>
<td>R_SIO_csi.h.78k0r</td>
<td>I/F module common definitions</td>
</tr>
</tbody>
</table>

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
Table 5.3 lists the return values that are returned by the sample code.

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIO_OK</td>
<td>(error_t)0</td>
<td>Successful operation</td>
</tr>
<tr>
<td>SIO_ERR_PARAM</td>
<td>(error_t)-1</td>
<td>Parameter error</td>
</tr>
<tr>
<td>SIO_ERR_HARD</td>
<td>(error_t)-2</td>
<td>Hardware error</td>
</tr>
<tr>
<td>SIO_ERR_OTHER</td>
<td>(error_t)-7</td>
<td>Other error</td>
</tr>
</tbody>
</table>

5.5.2 Miscellaneous Definitions
Table 5.4 lists miscellaneous definitions that are used in the sample code.

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIO_LOG_ERR</td>
<td>1</td>
<td>Log type: Error</td>
</tr>
<tr>
<td>SIO_TRUE</td>
<td>(uint8_t)0x01</td>
<td>Flag &quot;ON&quot;</td>
</tr>
<tr>
<td>SIO_FALSE</td>
<td>(uint8_t)0x00</td>
<td>Flag &quot;OFF&quot;</td>
</tr>
<tr>
<td>SIO_HI</td>
<td>(uint8_t)0x01</td>
<td>Port &quot;H&quot;</td>
</tr>
<tr>
<td>SIO_LOW</td>
<td>(uint8_t)0x00</td>
<td>Port &quot;L&quot;</td>
</tr>
<tr>
<td>SIO_OUT</td>
<td>(uint8_t)0x01</td>
<td>Port output setting</td>
</tr>
<tr>
<td>SIO_IN</td>
<td>(uint8_t)0x00</td>
<td>Port input setting</td>
</tr>
<tr>
<td>SIO_TX_WAIT</td>
<td>(uint16_t)50000</td>
<td>SIO transmission completion waiting time 50000 × 1 μs = 50 ms</td>
</tr>
<tr>
<td>SIO_RX_WAIT</td>
<td>(uint16_t)50000</td>
<td>SIO receive completion waiting time 50000 × 1 μs = 50 ms</td>
</tr>
<tr>
<td>SIO_DMA_TX_WAIT</td>
<td>(uint16_t)50000</td>
<td>DMA transmission completion waiting time 50000 × 1 μs = 50 ms</td>
</tr>
<tr>
<td>SIO_DMA_RX_WAIT</td>
<td>(uint16_t)50000</td>
<td>DMA receive completion waiting time 50000 × 1 μs = 50 ms</td>
</tr>
<tr>
<td>SIO_T_SIO_WAIT</td>
<td>(uint16_t)MTL_T_1US</td>
<td>SIO transmit and receive completion waiting polling time</td>
</tr>
<tr>
<td>SIO_T_DMA_WAIT</td>
<td>(uint16_t)MTL_T_1US</td>
<td>DMA transmit and receive completion waiting polling time</td>
</tr>
<tr>
<td>SIO_T_BRR_WAIT</td>
<td>(uint16_t)MTL_T_10US</td>
<td>BRR setting wait time</td>
</tr>
</tbody>
</table>
5.6 Structures and Unions

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

```c
/* 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

Table 5.5 lists the functions that are used in the sample code.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_SIO_Init_Driver()</td>
<td>Driver initialization processing</td>
</tr>
<tr>
<td>R_SIO_Disable()</td>
<td>Serial I/O disable setting processing</td>
</tr>
<tr>
<td>R_SIO_Enable()</td>
<td>Serial I/O enable setting processing</td>
</tr>
<tr>
<td>R_SIO_Open_Port()</td>
<td>Serial I/O open setting processing</td>
</tr>
<tr>
<td>R_SIO_Tx_Data()</td>
<td>Serial I/O data transmit processing</td>
</tr>
<tr>
<td>R_SIO_Rx_Data()</td>
<td>Serial I/O data receive processing</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th><strong>R_SIO_Init_Driver</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview</strong></td>
</tr>
<tr>
<td><strong>Header</strong></td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
</tr>
</tbody>
</table>
| **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](image)

Disables the serial I/O function and sets pins as ports.
### 5.8.2 Serial I/O Disable Setting Processing

<table>
<thead>
<tr>
<th>R_SIO_Disable</th>
<th>Serial I/O disable setting processing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td>R_SIO.h, R_SIO_csi.h, mtl_com.h</td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
<td>error_t R_SIO_Disable(void)</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>• 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.</td>
</tr>
<tr>
<td></td>
<td>• Set the slave device select signal high before calling this function.</td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Return values</strong></td>
<td>SIO_OK ; Successful operation</td>
</tr>
<tr>
<td><strong>Notes</strong></td>
<td>• Enables supply of the input clock to the serial array unit.</td>
</tr>
<tr>
<td></td>
<td>• Waits a minimum of 4 cycles of fCLK.</td>
</tr>
<tr>
<td></td>
<td>• Sets STm, S0m, and SOEm to stop operation and switches pins to function as ports.</td>
</tr>
<tr>
<td></td>
<td>• Sets SCRmn to set communication disabled as the communication mode.</td>
</tr>
<tr>
<td></td>
<td>• Writes 0020h to SMRmn (value after a reset) to initialize it.</td>
</tr>
<tr>
<td></td>
<td>• Sets SOLm.</td>
</tr>
<tr>
<td></td>
<td>• This function can be called to disable the serial I/O function when serial I/O is not used.</td>
</tr>
<tr>
<td></td>
<td>• This function does not control stopping supply of the input clock to the serial array unit.</td>
</tr>
</tbody>
</table>
Figure 5.5 Serial I/O Disable Setup Processing Outline

Start

Enable serial array unit input clock

Enable serial array unit input clock

End

Disable serial I/O function

SIO_DISABLE()

SIO_DISABLE()

End

Start

Enable serial array unit input clock

Enable serial array unit input clock

End

Disable serial I/O function

SIO_DISABLE()

SIO_DISABLE()

End

Start

Enable serial array unit input clock

Enable serial array unit input clock

End

Disable serial I/O function

SIO_DISABLE()

SIO_DISABLE()

End

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 PMxx ← 0b: 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

Set 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 PMxx ← 0b: 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

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 PMxx ← 0b: 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
5.8.3 Serial I/O Enable Setting Processing

| R_SIO_Enable | 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.
Start

Enable serial array unit input clock

- Enables the serial array unit input clock.
  - PERn SAUmEN ← 1b: Enables the input clock supply.

- Enables reading/writing to registers.
  - PERn SAUmEN ← 1b: Enables the input clock supply.

- Enables reading/writing to registers.
  - PERn SAUmEN ← 1b: Enables the input clock supply.

Initialize port

SIO_IO_INIT()

- 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

Enable serial I/O function

SIO_ENABLE(BrgData)

- Enables the serial I/O function and sets the bit rate.
  - Sets CKm0 as the operating clock in SPSm.
  - 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.

End

Figure 5.6 Serial I/O Enable Setup Processing Outline
5.8.4 Serial I/O Open Setting Processing

<table>
<thead>
<tr>
<th>R_SIO_Open_Port</th>
<th>Serial I/O open setting processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headers</td>
<td>R_SIO.h, R_SIO_csi.h, mtl_com.h</td>
</tr>
<tr>
<td>Declaration</td>
<td><code>error_t R_SIO_Open_Port(void)</code></td>
</tr>
</tbody>
</table>
| Explanation     | - 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 value    | `SIO_OK` ; Successful operation   |
| Remarks         | 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. |

![Figure 5.7 Serial I/O Open Setup Processing Outline](image.png)
5.8.5 Serial I/O Data Transmit Processing

<table>
<thead>
<tr>
<th>R_SIO_Tx_Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview</strong></td>
</tr>
<tr>
<td><strong>Header</strong></td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
</tr>
</tbody>
</table>
| **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

Start

Enable serial I/O transmission
SIO_TX_ENABLE()

Set TxWait (timeout count)

Clear transfer-end interrupt flag

Write transmit data

Transfer-end interrupt flag

IFxx = 0
Wait the TxWait subtraction for 1 μs

IFxx = 1

Update data storage pointer

Decrement transmit byte count

TxCnt! = 0

Transmit end?

TxCnt = 0

Disable serial I/O transmission
SIO_TX_DISABLE()

End

Enables serial I/O transmission and serial communication.
• Enables transmit mode in SCRmn.
  TEXmn=1b, REXmn=0b
• SOm CKOmn ← 1b: Sets clock output high (for use as port function).
  SOm SOmn ← 1b: Sets data output high (for use as port function).
• SOEm SOEmn ← 1b: Enables data output by communication.
• SSm SSmn ← 1b: Enables clock output by communication.

Sets to 50,000.

IFxx ← 0b

Writes transmit data to SIPp.

pData++

TxCnt--

Disables serial I/O transmission.
• STm STmn ← 1b: Stops channel operation.
  (Stops clock output by communication.)
  SEmn is cleared to 0, causing the state to transition to operation stopped.
• SOEm SOEmn ← 0b: Stops data output by communication.
• SCRmn TEXmn ← 0b, REXmn ← 0b: Stops communication.
5.8.6 Serial I/O Data Receive Processing

<table>
<thead>
<tr>
<th>R_SIO_Rx_Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview</strong></td>
</tr>
<tr>
<td><strong>Header</strong></td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
</tr>
</tbody>
</table>
| **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=0b 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 5.9  Serial I/O Data Reception Processing Outline**

---

Start

Enable serial I/O transmission/reception  
**SIO_TRX_ENABLE()**

- Enables serial I/O transmission/reception and serial communication.
- Enables transmit mode in SCRmn.  
  TEXmn=1b, REXmn=0b
- SOm CKOmн ← 1b: Sets clock output high (for use as port function).
- SOm SOmn ← 1b: Sets data output high (for use as port function).
- SOEm SOEmn ← 1b: Enables data output by communication.
- SSm SSmn ← 1b: Enables clock output by communication.

Set RxWait (timeout count)

Clear transfer-end interrupt flag

Write transmit dummy data

Transfer-end interrupt flag

IFxx = 0  
Wait the RxWait subtraction for 1 μs

IFxx = 1

Read receive data

Update data storage pointer  
pData++

Decrement receive byte count  
RxCnt--

RxCnt! = 0

Transfer-end interrupt flag

RxCnt = 0

Disable serial I/O transmission/reception  
**SIO_TRX_DISABLE()**

- Disables serial I/O transmission/reception.
- STm STmn ← 1b: Stops channel operation.  
  (Stops clock output by communication.)
- SEmн is cleared to 0, causing the state to transition to operation stopped.
- SOEm SOEmn ← 0b: Stops data output by communication.
- SCRmn TEXmn ← 0b, REXmn ← 0b: Stops communication.

End
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) Sets the DataIn pin to noise filter off for CSI mode.
   (2) Sets the DataIn pin to the normal input buffer by using the port input mode register (PIMxx).
   (3) 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) Sets the DataOut pin to the normal output mode by using the port output mode register (POMxx).
   (2) 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) Sets the DataOut pin to the normal output mode by using the port output mode register (POMxx).
   (2) 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 78K0R/KE3-L.
   (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 78K0R/KE3-L.

(1) Sets the channel to operation stopped mode and switches the pins to function as ports.
- Sets CKOm=1b and SOhm=1b in SOh so that the pins function as ports.*1
- Sets STm=1b in STm.
  → Cleared the SEm bit to 0 and stopped clock output by serial communication operation.
  → Put the channel into the operation stopped state.
  → The value set in the CKOm bit in SOh is output from the serial clock output pin.
- Sets SOEm=0b in SOEm, stopping data output by serial communication operation.
- Sets CKOm=1b and SOhm=1b in SOh so that the pins function as ports.*2
- Sets SOLm=0b in SOLm (depends on the channel).

(2) Sets TEXm=0b and REXm=0b in SCRm, setting the operation mode to communication stopped.

(3) Sets SMRm 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 (SOh) 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 CKOm and SOhm bits in the serial output register (SOh).
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 SEm 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 SOh before stopping clock output by using STm and stopping data output by using SOEm. However, writing to SOh is ignored if the values of both SEm and SOEm are 1, so the function’s effects depend on the settings of SEm and SOEm immediately before it is executed.
Since the effects depend on the preceding state, during initialization SOh 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 SOh setting can take effect.

2. SOh is set after stopping clock output by using STm and stopping data output by using SOEm.
This ensures that the SOh 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 78K0R/KE3-L.
   (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).
   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.

Perform the following processing in the 78K0R/KE3-L.

(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 78K0R/KE3-L.
   (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 78K0R/KE3-L.

   (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 5.10 shows the state transition diagram. Do not perform serial transmission or reception before the serial I/O function has been initialized. For details, see 7.7, Prohibition of Data Transmission and Reception.

![State Transition Diagram]

Figure 5.10  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 78K0R uses a different method to make these definitions, so the header file should be commented out in the sample code. Refer to 7.4, Method of Manipulating SFR Area, for instructions for making the necessary settings.

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 78K0R/Kx3-L, define the endian mode as little-endian.

```c
/* 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 78K0R/Kx3-L, leave this commented out so that the definition is not made.

```c
/* 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.

```c
/* 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 78K0R/Kx3-L.

```c
/* 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.

```c
/* Define the counter value for the timer. */
/* Specify according to the user MCU, clock and wait requirements. */
#if 1
/* Setting for 20 MHz no wait (Compile Option "-a. -zp" at PM+ Ver.6.31, 
CC78K0R Ver.2.13, RA78K0R Ver.1.33)/
#define MTL_T_1US            3  /* loop Number of 1 us */
#define MTL_T_2US            6  /* loop Number of 2 us */
#define MTL_T_4US            12 /* loop Number of 4 us */
#define MTL_T_5US            15 /* loop Number of 5 us */
#define MTL_T_10US           30 /* loop Number of 10 us */
#define MTL_T_20US           60 /* loop Number of 20 us */
#define MTL_T_30US           90 /* loop Number of 30 us */
#define MTL_T_50US          150 /* loop Number of 50 us */
#define MTL_T_100US         300 /* loop Number of 100 us */
#define MTL_T_200US         600 /* loop Number of 200 us */
#define MTL_T_300US         900 /* loop Number of 300 us */
#define MTL_T_400US     ( MTL_T_200US * 2 ) /* loop Number of 400 us */
#define MTL_T_1MS         3000 /* 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 $\mu$s.

Supposing transfer at 100 kHz and usage with Multimedia Cards, make the setting for 10 $\mu$s.

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

The 78K0RKE3-L 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_CRC7_USED    /* CRC7 used         */
#define SIO_CRCCCITT_USED /* CRC-CCITT used */
```
(3) Defining the Pins to Be Used

Define the pins to be used.

/*--------------------------------------------------------------------------*/
/*  Define the control port.                                               */
/*  Delete comment of a related macrodefinition, and please validate setting. */
/*--------------------------------------------------------------------------*/
#define SIO_PM_DATAO  PM7.0   /* SIO DataOut */
#define SIO_PM_DATAI  PM7.1   /* SIO DataIn */
#define SIO_PM_CLK   PM7.2   /* SIO CLK */
#define SIO_P_DATAO  P7.0    /* SIO DataOut */
#define SIO_P_DATAI  P7.1    /* SIO DataIn */
#define SIO_P_CLK   P7.2    /* SIO CLK */
#define SIO_PIM_DATAI  PIM7.1  /* SIO DataIn */
#define SIO_PIM_CLK  PIM7.2  /* SIO CLK */
#define SIO_POM_DATAO  POM7.0  /* SIO DataOut */
#define SIO_POM_CLK  POM7.2  /* SIO CLK */

(4) Defining the Peripheral Enable Register

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

#define SIO_SAUEN   SAU0EN   /* Control of CSI0 input clock supply */

(5) Defining the CSI Channel to Be Used

Specify the CSI channel to be used. CSI01 is used in the example below.

/*----------------- SIO definitions -----------------*/
/* CSI01 setting example - Set the following for the system. */
#define SIO_SPS   SPS0    /* Serial clock select register */
#define SIO_SMR   SMR01   /* Serial mode register */
#define SIO_SCR   SCR01   /* Serial communication operation setting register */
#define SIO_SDR   SDR01   /* Serial data register */
#define SIO_TXBUF  SIO01   /* SIOp data register */
#define SIO_RXBUF  SIO01   /* SIOp data register */
#define SIO_SIR   SIR01   /* Serial flag clear trigger register */
#define SIO_SSR   SSR01   /* Serial status register */
#define SIO_SS    SSOL1.1  /* Serial channel start register SSmn */
#define SIO_ST    STOL1.1  /* Serial channel stop register STmn */
#define SIO_SE    SEOL1.1  /* Serial channel enable status register SEmn */
#define SIO_SOE   SOEOL1.1 /* Serial output enable register SOEmn */
#define SIO_SO    SO0      /* Serial output register SOmn */
#define SIO_SOL   SOL0     /* Serial output level register */
#define SIO_SNFEN  NFEN0.0 /* Use of noise filter of RXD pin SNFEN */

#define SIO_TXNEXT  (SSR01L & 0x20) /* CSI Transmit data empty */
#define SIO_RXNEXT  IF0H.6  /* CSI Receive completion */
#define SIO_TXEND   IF0H.6   /* 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.

```c
#define SIO_USPS_INIT   (uint16_t)0x0000
/* 0000000000000000B */ /* SPS CSI initial setting */
/* 11111111111111111111B */ /* CKm0:No division of fclk */
/* 11111111111111111111B */ /* CKm1:No division of fclk */
/* ++++++++++++++++++++++ Reserved */
```

(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.

```c
#define SIO_USMR_INIT   (uint16_t)0x0020
/* 0000000000100000B */ /* SMR CSI initial setting */
/* 11111111111111111111B */ /* Interrupt source : Transfer end interrupt */
/* 11111111111111111111B */ /* Operation mode : CSI mode */
/* 11111111111111111111B */ /* Reserved */
/* 11111111111111111111B */ /* Reserved (Controls in UART mode) */
/* 11111111111111111111B */ /* Start trigger source : Software trigger */
/* 11111111111111111111B */ /* Reserved */
/* 11111111111111111111B */ /* 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.

```c
#define SIO_USO_INIT   (uint16_t)0x0A0A
/* 0000101000001010B */ /* SO0 initial setting */
/* 11111111111111111111B */ /* SO00 output  : 0 */
/* 11111111111111111111B */ /* SO01 output  : 1  CSI01 */
/* 11111111111111111111B */ /* SO02 output  : 0 */
/* 11111111111111111111B */ /* Reserved   : 1  Fixed */
/* 11111111111111111111B */ /* Reserved   : 0 */
/* 11111111111111111111B */ /* CK00 output : 0 */
/* 11111111111111111111B */ /* CK001 output : 1  CSI01 */
/* 11111111111111111111B */ /* CK002 output : 0 */
/* 11111111111111111111B */ /* Reserved   : 1  Fixed */
/* ++++++++++++++++++++++ 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 CSI01 is used is shown in the example below.

```c
#define SIO_USOL_INIT   (unit16_t)0xFFFFA /* 1111111111111010B */ /* 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)**

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

```c
#ifndef SIO_DATAI_INIT
#define SIO_DATAI_INIT() do {
    SIO_SNFEN_ = 0; /* Noise filter OFF */
    SIO_PIM_DATAI = 0; /* Normal input buffer */
    SIO_PM_DATAI = 1; /* DataIn Input */
} while (0)
#endif
```
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 Method of Manipulating SFR Area
The SFR area contains registers that perform special functions, such as mode registers and control registers for the peripheral hardware of the 78K0R.
It is possible to manipulate this register area at the C source code level by means of declarations using the SFR.
The SFR can be used in the C source code by means of the #pragma instruction. (It does not matter if the keyword SFR is capitalized or lowercased in the source code.)

```
#pragma SFR
```
Place #pragma SFR at the beginning of the C source code. Note that if the specification #pragma PC (device type) is also included in the source code, #pragma SFR should come after it. The following items may also precede #pragma SFR:
1. Comments
2. Pre-processing instructions that do not define or reference variables and do not define or reference functions.

7.5 Port Control for Serial Data and Clock Output Pins
To set these pins to function as ports, set to 1 the CKOm 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 CKOm bit and SOmn bit are set to 1, the unmodified port register (Pxx) setting value becomes the output value of the corresponding pin.

7.6 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.
7.7 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.

7.8 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).
Website and Support

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

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

All trademarks and registered trademarks are the property of their respective owners.
### Revision Record

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Jan.19.12</td>
<td>— First edition issued</td>
</tr>
<tr>
<td>1.01</td>
<td>Jan.31.12</td>
<td>3 Table 2.1 Version of the sample code changed to Ver.2.01.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 Table 5.2 Application note updated.</td>
</tr>
<tr>
<td>1.02</td>
<td>Aug.31.12</td>
<td>3 Table 2.1 Version of the sample code and Version of Software used for evaluation changed to Ver.2.02.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 5.4 File Configuration updated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19 Figure 5.8 changed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21 Figure 5.9 changed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32 6.1 mtl_com.h (common header file) (7) Defining the RAM Area to Be Accessed changed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>37 6.2.2 R_SIO_csi.h (9) Defining the Serial Output Level Register (SOLm) changed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>37 6.2.2 R_SIO_csi.h (10) Defining the Port Input Mode Register (PIM) added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>39 7.8 Setting Serial Output Level Register (SOLm) added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Last page updated.</td>
</tr>
</tbody>
</table>
General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord 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 one with a different type number, confirm that the change will not lead to problems.
   - The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.
SALES OFFICES

Renesas Electronics Corporation

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

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Inc.
1151 Nicholl Road Ne, Markham, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

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
Arcadistrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-60530, Fax: +49-211-60531-127

Renesas Electronics (Shanghai) Co., Ltd.
7F East, Quantum Plaza, No.27 Zhichu Ru Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8236-7679

Renesas Electronics (China) Co., Ltd.
7F North, Quantum Plaza, No.27 Zhichu Ru Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8234-1698, Fax: +86-10-8234-1699

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16F, Tower 3, Grand Century Place, 193 Prince Edward Road West, Mong kok, Kowloon, Hong Kong
Tel: +852-2886-9316, Fax: +852-2886-9022-9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 96, 1st floor North Road, Taipei, Taiwan
Tel: +886-2-8177-9000, Fax: +886-2-8175-9570

Renesas Electronics Malaysia Sdn.Bhd.
Unit 905, 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 Korea Co., Ltd.
11F, Samik-Lavied or Bldg., 220-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-559-3727, Fax: +82-2-559-5141

© 2012 Renesas Electronics Corporation. All rights reserved.
Colophon 1.1