RX62T Group

Asynchronous SCIb Communication Using the DTC Module

Introduction

This application note presents a sample program that performs SCI (serial communications interface) asynchronous serial communication using the Renesas MCU's DTC (data transfer controller) module.

Target Devices

RX62T Group

Other members of the RX Family that have the same I/O registers (peripheral unit control registers) as the RX62T Group products can also use the code from this application note. Note, however, that since certain aspects of the functions used may be changed in other devices due to function additions or other differences, the documentation for the device used must be checked carefully before using this code. When using this code in an end product or other application, its operation must be tested and evaluated thoroughly.

Contents

1. Specifications .................................................................................................................................................. 2
2. Operation Confirmation Environment ............................................................................................................ 3
3. Functions Used ................................................................................................................................................ 4
4. Operation ......................................................................................................................................................... 4
5. Software ........................................................................................................................................................ 8
6. Reference Documents .................................................................................................................................... 18
1. **Specifications**

The code presented in this application note performs serial communication (transmission and reception) by performing data transfers between RAM and the SCI (serial communications interface) module using the data transfer controller (DTC). Figure 1 presents an overview of this asynchronous serial data transmission and reception that uses the DTC.

1. This sample program uses the SCI channel 0 and the DTC module.
2. An 8-bit length, one stop bit, no parity bit communication format is used.
3. For the transmit operation, the DTC is activated by the transmit data empty interrupt request, and it transfers the transmit data, which is prepared in advance from an arbitrary transfer source to the SCI transmit data register (TDR).
4. For the transmit operation, the DTC is activated by the receive data full interrupt request, and it transfers the receive data from the SCI receive data register (RDR) to an arbitrary transfer destination that is prepared in advance.
5. When the specified number of transfers have been performed, these operations are reinitialized.

---

**Figure 1** Overview of Asynchronous Serial Data Transmission/Reception Using the DTC
### 2. Operation Confirmation Environment

Table 1 lists the environment required for confirming master operation.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller used</td>
<td>R5F562TAADFP (RX62T Group)</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>Main clock: 12.5 MHz System clock (ICLK): 100 MHz Peripheral module clock (PCLK): 50 MHz</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>5.0 V</td>
</tr>
<tr>
<td>Integrated development environment</td>
<td>Renesas Electronics Corporation High-performance Embedded Workshop Version 4.09.01.007</td>
</tr>
<tr>
<td>C compiler</td>
<td>Renesas Electronics Corporation RX Standard Toolchain (V1.2.1.0)</td>
</tr>
<tr>
<td>iodefine.h version</td>
<td>Version 2.0</td>
</tr>
<tr>
<td>Endian order</td>
<td>Little Endian, Big Endian</td>
</tr>
<tr>
<td>Operating mode</td>
<td>Single-chip mode</td>
</tr>
<tr>
<td>Processor mode</td>
<td>User mode</td>
</tr>
<tr>
<td>Sample code version</td>
<td>Version 1.01</td>
</tr>
<tr>
<td>Board used</td>
<td>Renesas Starter Kit for RX62T (R0K5562T0S000BE)</td>
</tr>
</tbody>
</table>
3. Functions Used

- Clock generation circuit
- Low power consumption functions
- Interrupt control unit (ICU)
- Serial communication interface (SCI)
- Data transfer controller (DTC)

See the RX62T Group User’s Manual: Hardware for detailed information.

4. Operation

4.1 Operation Mode Settings

In the sample program, mode pins are set to MD1 = 1, MD0 = 1 to select single-chip mode as the operating mode and the ROME bit in system control register 0 (SYSCR0) is set to 1 to enable the on-chip ROM.

Table 2 lists the operating mode settings used in the sample program.

<table>
<thead>
<tr>
<th>Mode Pin</th>
<th>SYSCR0 Register</th>
<th>Operating Mode</th>
<th>On-Chip ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD1</td>
<td>MD0  ROME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1   1</td>
<td>Single-chip mode</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

Note: The initial setting of the ROME bit in the SYSCR0 register is SYSCR0.ROME = 1, so it is not necessary for the sample program to make settings to the SYSCR0 register.

4.2 Clock Settings

The evaluation board used for this application note includes a 12.5 MHz crystal oscillator.

Therefore this application note uses the following settings for the system clock (ICLK) and the peripheral module clock (PCLK): 8× (100 MHz) and 4× (50 MHz).

4.3 Endian Mode Setting

The sample program presented in this application note supports both big- and little-endian mode. Table 3 lists the hardware endian mode settings of the master device.

<table>
<thead>
<tr>
<th>MDE pin</th>
<th>Endian</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Little endian</td>
</tr>
<tr>
<td>1</td>
<td>Big endian</td>
</tr>
</tbody>
</table>
Table 4 lists the endian settings used in the compiler options.

**Table 4  Endian Mode Settings (Compiler Options)**

<table>
<thead>
<tr>
<th>MCU Option</th>
<th>Endian</th>
</tr>
</thead>
<tbody>
<tr>
<td>endian = little</td>
<td>Little endian</td>
</tr>
<tr>
<td>endian = big</td>
<td>Big endian</td>
</tr>
</tbody>
</table>

Note: Set the MDE pin to match the endian mode selected as a compiler option.

### 4.4 Bit Order Settings

The program in this application note supports both right and left as the bit order. Table 5 lists the bit order settings in the microcontroller option in the compiler options.

**Table 5  Bit Order Settings (Compiler Options)**

<table>
<thead>
<tr>
<th>MCU Option</th>
<th>Bit Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit_order = right</td>
<td>Bit field members are allocated in order starting with the low-order bit. (Default)</td>
</tr>
<tr>
<td>bit_order = left</td>
<td>Bit field members are allocated in order starting with the high-order bit.</td>
</tr>
</tbody>
</table>

Notes:
1. In this application note, bit fields are used in the I/O register definitions file (iodefine.h). In the I/O register definitions file, "left" is specified with the #pragma bit_order extension, and the bit field members are allocated in order starting with the high-order bit.
2. If both the bit_order compiler option and the #pragma bit_order extension are specified, the #pragma bit_order extension specification takes precedence. Thus the bit fields defined in the I/O register definitions file will be allocated in order starting with the high-order bit, regardless of the compiler options bit_order specification.

### 4.5 SCI Settings

Table 6 lists the SCI communication function settings used in this sample program.

**Table 6  SCI Settings and Conditions**

<table>
<thead>
<tr>
<th>Channel Used</th>
<th>SCI 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication mode</td>
<td>Asynchronous serial communication mode</td>
</tr>
<tr>
<td>Interrupts</td>
<td>• Receive error interrupt (ERI)</td>
</tr>
<tr>
<td></td>
<td>• Receive data full interrupt (RXI)</td>
</tr>
<tr>
<td></td>
<td>• Transmit data empty interrupt (TXI)</td>
</tr>
<tr>
<td></td>
<td>• Transmit complete interrupt (TEI)</td>
</tr>
<tr>
<td>Communication speed</td>
<td>38,400 bps (PCLK = 50 MHz)</td>
</tr>
<tr>
<td>Data length</td>
<td>8-bit data</td>
</tr>
<tr>
<td>Stop bits</td>
<td>1 stop bit</td>
</tr>
<tr>
<td>Parity</td>
<td>None</td>
</tr>
</tbody>
</table>
### 4.6 DTC Settings

Table 7 lists the DTC transfer conditions used in this sample program.

<table>
<thead>
<tr>
<th>Condition</th>
<th>SCI Transmit DTC Transfer Conditions (TXI0)</th>
<th>SCI Receive DTC Transfer Conditions (RXI0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer information</td>
<td>Full address mode</td>
<td>Full address mode</td>
</tr>
<tr>
<td>Transfer mode</td>
<td>Normal transfer mode</td>
<td>Normal transfer mode</td>
</tr>
<tr>
<td>Transfer count</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>Transfer data</td>
<td>Size: byte</td>
<td>Size: byte</td>
</tr>
<tr>
<td></td>
<td>Data content: 256 bytes with values from H'00 to H'FF</td>
<td>Data content: An arbitrary 256 bytes of data</td>
</tr>
<tr>
<td>Transfer source</td>
<td>Internal RAM</td>
<td>Receive data register (SCI0.RDR)</td>
</tr>
<tr>
<td>Transfer destination</td>
<td>Transmit data register (SCI0.TDR)</td>
<td>Internal RAM</td>
</tr>
<tr>
<td>Transfer source address</td>
<td>The transfer source address is incremented after the transfer</td>
<td>The transfer source is fixed</td>
</tr>
<tr>
<td>Transfer destination address</td>
<td>The transfer destination is fixed</td>
<td>The transfer destination address is incremented after the transfer</td>
</tr>
<tr>
<td>Start event</td>
<td>Started on the SCI transmit data empty interrupt</td>
<td>Started on the SCI receive data full interrupt</td>
</tr>
<tr>
<td>Interrupts</td>
<td>An interrupt is enabled to the CPU after the specified data transfer completes.</td>
<td>An interrupt is enabled to the CPU after the specified data transfer completes.</td>
</tr>
</tbody>
</table>
4.7 Operation Timing

Figure 2 shows the timing of the operations performed by this sample program.

**Transmit Operation Example**

- Initialization
- TE, TIE (SCI)
- TXI IER (ICU)
- TXI IR (ICU)
- DTC transfer
  - TDR (SCI) HFF
  - Automatic transfer from TDR to TSR
- TSR (SCI) ...
- TxD pin D1 D2 D3 D4 D255 D256 D1 D2

**Receive Operation Example**

- Initialization
- RE, RIE (SCI)
- RXI IER (ICU)
- RXI IR (ICU)
- RxD pin D1 D2 D254 D255 D256
- RSR (SCI) ...
  - Automatic transfer from RSR to RDR
- RDR (SCI) H00 D1 D254 D255 D256

**Figure 2 Operation Timing**

Note: Note that when the DTC module is used in combination with communication functions in the RX62T Group microcontroller, if the next transfer request occurs before the IR flag is cleared automatically, the transfer request will be lost.

5. Software

5.1 Constants

Table 8 lists the constants used in the sample code.

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Set Value</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXD_MAX</td>
<td>256</td>
<td>DTC transfer count</td>
</tr>
</tbody>
</table>

5.2 Structures and Unions

Figure 3 shows the structures and unions used in the sample program.

```c
#pragma bit_order left
#pragma unpack
struct st_dtc_full{
    union{
        unsigned long LONG;
        struct{
            unsigned long MRA_MD :2; /* MRA.MD bits */
            unsigned long MRA_SZ :2; /* MRA.SZ bits */
            unsigned long MRA_SM :2; /* MRA.SM bits */
            unsigned long :2;
            unsigned long MRA_CHNE :1; /* MRA.CHNE bit */
            unsigned long MRA_CHNS :1; /* MRA.CHNS bit */
            unsigned long MRA_DISEL :1; /* MRA.DISEL bit */
            unsigned long MRA_DTS :1; /* MRA.DTS bit */
            unsigned long MRA_DM :2; /* MRA.DM bits */
            unsigned long :2;
            unsigned long :16;
        }BIT;
    }MR;
    void * SAR; /* SAR register */
    void * DAR; /* DAR register */
    struct{
        unsigned long CRA:16; /* CRA register */
        unsigned long CRB:16; /* CRB register */
    }CR;
};
```

Figure 3 Structures and Unions Used in the Sample Code (DTC Transfer Information)

Note: The DTC transfer information alignment count is stipulated to be 4 by the "#pragma unpack" statement.
5.3 Variables

Table 9 lists the variables used in the sample program.

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable</th>
<th>Usage</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned char</td>
<td>recvBuf[BUF_SIZE]</td>
<td>Array variable that holds the serial receive data</td>
<td>main, dtc_init, int_sci_rxi0</td>
</tr>
<tr>
<td>unsigned char</td>
<td>trnsBuff[BUF_SIZE]</td>
<td>Array variable that holds the serial transmit data</td>
<td>dtc_init, int_sci_tei0</td>
</tr>
<tr>
<td>st_dtc_full</td>
<td>dtc_rx</td>
<td>Structure variable that holds the DTC transfer information for SCI reception</td>
<td>dtc_init, int_sci_rxi0</td>
</tr>
<tr>
<td>st_dtc_full</td>
<td>dtc_tx</td>
<td>Structure variable that holds the DTC transfer information for SCI transmission</td>
<td>dtc_init, int_sci_tei0</td>
</tr>
<tr>
<td>void*</td>
<td>dtc_table[256]</td>
<td>DTC vector table that allocates addresses for the dtc_rx/dtc_tx DTC transfer information (The start address is set to 0x00000000.)</td>
<td>dtc_init</td>
</tr>
</tbody>
</table>

5.4 Functions

Table 10 lists the functions used in this application note's sample program.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>HardwareSetup</td>
<td>Initialization, clock settings, and clearing the module stop state</td>
</tr>
<tr>
<td>main</td>
<td>Main processing</td>
</tr>
<tr>
<td>icu_init</td>
<td>ICU initialization and setting the interrupt levels</td>
</tr>
<tr>
<td>sci0_init</td>
<td>SCI initialization, transfer clock DTC settings</td>
</tr>
<tr>
<td>dtc_init</td>
<td>DTC initialization, transfer information setup, DTC vector base register setting, and enabling DTC activation</td>
</tr>
<tr>
<td>int_sci_txi0</td>
<td>Transfer interrupt</td>
</tr>
<tr>
<td>int_sci_tei0</td>
<td>Transmit complete interrupt</td>
</tr>
<tr>
<td>int_sci_rxi0</td>
<td>Receive interrupt</td>
</tr>
<tr>
<td>int_sci_eri0</td>
<td>Receive error interrupt</td>
</tr>
</tbody>
</table>
5.5 Processing Flow

Figures 4 to 12 show the processing flow of the sample program.

**Figure 4 Initialization Processing**

- **HardwareSetup**
  - Set the operating frequency
  - Clear the SCI module stop states
  - **main**

- **System clock register (SCKCR) setting**
  - Operating frequencies: ICLK = 100 MHz, PCLK = 50 MHz

- **Module stop control register B (MSTPDRB) settings**
- **Module stop state clear for SCI0**

- Jumps to the main routine.

**Figure 5 Main Processing**

- **main**
- **Set up the transfer data**
- **Initialize the ICU module (icu_init)**
- **Initialize the SCI module (sci_init)**
- **Initialize the DTC module (dtc_init)**
- **Set the interrupt request enable register (IERm)**

- Stores the transfer data in trnsBuf[256].

- **Set the SCI registers.**
- **Initialize the I/O ports.**
- **Set the bit rate.**

- **Sets the DTC registers.**
- **Sets up the DTC transfer information**
- **Sets the DTC start bit**

- Enables the ERI0, RXI0, and TXI0 interrupt requests.
Set the interrupt request priority register (IPRm)

Set the DTC start enable register (DTCEERn)

icu_init

Sets up DTC activation by RXI0 and TXI0.

Interrupt level = 1

return

Figure 6  ICU Initialization
The transfer rate is set so that the period for 1 bit is 16 cycles of the base clock.

The RXD0 pin input noise exclusion function is enabled.

Enable the PB port (b1) input buffer.

PCLK (50 MHz) is set as the internal baud rate generator clock source.

One stop bit, no parity bit, 8-bit data length, and asynchronous mode.

Serial communications interface mode

BRR = 40 (38,400 bps)

• TE = 1 (serial transfer operations enabled)
• TIE = 1 (TXI interrupt request enabled)
• TEIE = 1 (TEI interrupt request enabled)
• RE = 1 (serial reception operations enabled)
• RIE = 1 (RXI and ERI interrupt requests enabled)

Figure 7 SCI Initialization
Set the transfer information addresses for the DTC start events

Start events: RXI0 and TXI0

Read skip disabled.

Set the DTC control register (DTCCR)

• Sets the DTC register transfer information (transmit)
  MRA: Normal transfer mode, byte size transfer, SAR incremented after transfer
  MRB: Chaining disabled, interrupts enabled after the data transfer, fixed DAR
  SAR: The transfer source start address is the internal RAM transmit data storage area.
  DAR: The transfer destination start address is the SCI transmit data register TDR.
  CRA: Transfer count: 256 operations
  CRB: Set to FFFFh.

Set the DTC register transfer information (MRA, MRB, SAR, DAR, CRA, and CRB)

• Sets the DTC register transfer information (receive)
  MRA: Normal transfer mode, byte size transfer, SAR held fixed after transfer
  MRB: Chaining disabled, interrupts enabled after the data transfer, DAR incremented after transfer
  SAR: The transfer source start address is the SCI receive data register RDR.
  DAR: The transfer destination start address is the internal RAM receive data storage area.
  CRA: Transfer count: 256 operations
  CRB: Set to FFFFh.

Set the DTC vector base register (DTCVBR)

Sets the base address used when calculating the DTC vector table address.

Read skip enabled.

Set the DTC control register (DTCCR)

Set the DTC module start register (DTCST)

DTC module activation.

Figure 8  DTC Initialization
Disables the RXI0 interrupt request.

Read skip disabled.

Sets the transfer destination start address.

Sets the transfer count (256 in this case).

Read skip enabled.

Verifies the DTC transfer operating state
• Checks the ACT bit in the DTC status register (DTCSTS).

Stops the DTC module.

Sets up DTC activation by RXI0.

The DTC module will be operating.

Enables the RXI0 interrupt request.

Figure 9 Receive Interrupt
Set the interrupt request enable register (IERm)

- Disables the TXI0 interrupt request.
- Enables the TEI0 interrupt request.

return

**Figure 10  Transmit Interrupt**
int_sci_tei0

Set the interrupt request enable register (IERm)

Set the DTC control register (DTCCR)

Set the DTC source address register (SAR)

Set the DTC transfer count register (CRA)

Set the DTC control register (DTCCR)

Is the DTC stopped?

Yes

Set the DTC module start register (DTCST)

Set the DTC start enable register (DTCERn)

Set the DTC module start register (DTCST)

Set the interrupt request enable register (IERm)

No

Disables the TEI0 interrupt request.

Read skip disabled.

Sets the transfer source start address

Sets the transfer count (256 in this case).

Read skip enabled.

• Verifies the DTC transfer operating state
• Checks the ACT bit in the DTC status register (DTCSTS).

Stops the DTC module.

Sets up DTC activation by TXI0.

The DTC module will be operating.

Enables the TXI0 interrupt request.

Figure 11  Transmit Complete Interrupt
• ORER: Overrun error flag
• FER: Framing error flag
• PER: Parity error flag

Note: * Not performed in this sample program.

Figure 12  Receive Error Interrupt
6. Reference Documents

- Hardware Manual
  RX62T Group User’s Manual: Hardware  Rev.1.10
  (The latest version can be downloaded from the Renesas Electronics Web site.)

- Software Manual
  RX Family User’s Manual: Software  Rev.1.00
  (The latest version can be downloaded from the Renesas Electronics Web site.)

- Development Environment Manual  Rev.1.01
  RX Family C/C++ Compiler Package User’s Manual
  (The latest version can be downloaded from the Renesas Electronics Web site.)

- Technical Updates
  (The latest information can be downloaded from the Renesas Electronics Web site.)
Website and Support

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

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

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

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Sep.27.11</td>
<td>—</td>
<td>First edition issued</td>
</tr>
<tr>
<td>1.01</td>
<td>Nov.01.18</td>
<td>3</td>
<td>Table 1  Operation Confirmation Environment changed</td>
</tr>
</tbody>
</table>
General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

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

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

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

3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   - The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable.
   When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   - When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal.
   Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

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

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against liability for infringement of any claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.

3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.

5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality." The intended applications for each Renesas Electronics product depend on the product's quality grade, as indicated below.

   "Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment, industrial robots, etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.), traffic control (traffic lights), large-scale communication equipment, key financial terminal systems, safety control equipment, etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations, etc.), or may cause serious property damage (space system; underwater repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment, etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third party arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.

7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.

8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accidents arising out of the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

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

© 2018 Renesas Electronics Corporation. All rights reserved.

Colophon 7.2