RX Family and M16C Family

Guide for Migration from the M16C to the RX: DMAC and DTC

Abstract

This document describes migration from the DMAC in the M16C Family to the DMAC and DTC in the RX Family.

Products

RX Family

M16C Family

When this document explains migration from the M16C Family to the RX Family, the M16C/65C Group MCU is used as an example of the M16C Family MCU, and the RX231 Group and RX660 Group MCUs are used as examples of the RX Family MCU. When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Differences in Terminology Between the RX Family and M16C Family MCUs

Item	M16C Family	RX Family	
	DMAC	DMAC	DTC
Mode for transferring specific data once	Single transfer mode	Normal transfer mode	Normal transfer mode
Mode for transferring specific data multiple times	Repeat transfer mode	Repeat transfer mode	Repeat transfer mode
Peripheral function registers	Special function registers (SFRs)	I/O registers	

Contents

1. General Differences Between the DMAC/DTC on the RX231/RX660 and the DMAC on the M16C

Table 1.1 shows General Differences Among the DMACs of the M16C, RX231, and RX660, Table 1.2 shows General Differences Between the DTC on the RX231 or RX660 and the DMAC on the M16C, and Table 1.3 shows Differences in the Number of Cycles Between the DMAC/DTC and the DMAC.

To transfer data giving priority to speed, select the DMAC; to transfer data by using a specific request source or transfer method you choose, select the DTC.

- For the DMAC, the transfer address, transfer mode, and other settings are specified in I/O registers provided for each channel. After a request source is generated, data transfer starts according to the I/O register setting values. Therefore, the processing time until transfer starts is shorter than that for the DTC.
- For the DTC, the transfer address, transfer mode, and other information is set in memory (ROM and RAM). After a request source is generated, the information is read from the memory, and then data transfer starts. Therefore, the processing time until transfer starts is longer than that for the DMAC. However, compared to the DMAC, more request sources and transfer methods can be selected, and there is no limit on the number of transfer channels.

Table 1.1 General Differences Among the DMACs of the M16C, RX231, and RX660

<table>
<tr><th rowspan="2">Item</th><th>M16C (M16C/65C)</th><th>RX (RX231)</th><th>RX (RX660)</th></tr>
<tr><th>DMAC</th><th>DMAC</th><th>DMAC</th></tr>
<tr><td>Number of channels</td><td>4 channels</td><td>4 channels</td><td>8 channels</td></tr>
<tr><td>Transfer space</td><td>00000h to FFFFFh</td><td>0000 0000h to 0FFF FFFFh and F000 0000h to FFFF FFFFh</td><td>0000 0000h to 0FFF FFFFh and F000 0000h to FFFF FFFFh</td></tr>
<tr><td>Request source</td><td>43 sources
• INT pin (falling edge) (8 sources)
• INT pin (both edges) (8 sources)
• Timers (11 sources)
• Communications (14 sources)
• A/D conversion (1 source)
• Software trigger</td><td>56 sources
• INT pin (4 sources): Choose low, falling edge, rising edge, or both edges
• Timers (15 sources)
• Communications (24 sources)
• A/D conversion and comparator (6 sources)
• ELC (2 sources)
• Capacitive touch sensing unit (2 sources)
• Serial sound interface (2 sources)
• Software trigger</td><td>112 sources
• INT pin (8 sources): Choose low, falling edge, rising edge, or both edges
• Timers (47 sources)
• Communications (39 sources)
• A/D conversion and comparator (7 sources)
• ELC (2 sources)
• Software trigger</td></tr>
<tr><td>Transfer units</td><td>8 or 16 bits</td><td>8, 16, or 32 bits</td><td>8, 16, or 32 bits</td></tr>
<tr><td>Transfer direction</td><td>Non-change → Non-change, Non-change → Forward, or Forward → Non-change</td><td>4 conditions selectable for transfer source and transfer destination
• Non-change
• Increment-by-offset *1
• Increment
• Decrement</td><td>4 conditions selectable for transfer source and transfer destination
• Non-change
• Increment-by-offset *1
• Increment
• Decrement</td></tr>
<tr><td>Transfer modes</td><td>Single transfer mode
Repeat transfer mode</td><td>Normal transfer mode
Repeat transfer mode
Block transfer mode</td><td>Normal transfer mode
Repeat transfer mode
Block transfer mode</td></tr>
<tr><td>Interrupts</td><td>Generated when the DMAi transfer counter underflows.</td><td>• Generated when the number of data units set by the transfer counter are transferred (transfer end interrupt).
• Generated when data of the repeat size is transferred or when the extended repeat area overflows (transfer escape end interrupt).</td><td>• Normal transfer mode: Generated when the specified number of transfers end; Repeat transfer mode: Generated when the specified number of times a transfer is repeated; Block transfer mode: Generated when the specified number of blocks are transferred (transfer end interrupt).
• Generated when data of the repeat size is transferred or when the extended repeat area overflows (transfer escape end interrupt).</td></tr>
</table>

Note: 1. Increment-by-offset can be set for DMAC0 only.

Table 1.2 General Differences Between the DTC on the RX231 or RX660 and the DMAC on the M16C

Item	M16C (M16C/65C)	RX (RX231)	RX (RX660)
	DMAC	DTC	DTC
Number of channels	4 channels	— *1	— *1
Transfer space	00000h to FFFFFh	• Short-address mode: 0000 0000h to 007F FFFFh and FF80 0000h to FFFF FFFFh • Full-address mode: 0000 0000h to FFFF FFFFh	• Short-address mode: 0000 0000h to 007F FFFFh and FF80 0000h to FFFF FFFFh • Full-address mode: 0000 0000h to FFFF FFFFh
Request source	43 sources • INT pin (falling edge) (8 sources) • INT pin (both edges) (8 sources) • Timers (11 sources) • Communications (14 sources) • A/D conversion (1 source) • Software trigger	98 sources • INT pin (9 sources): Choose low, falling edge, rising edge, or both edges • Timers (48 sources) • Communications (28 sources) • A/D conversion and comparator (6 sources) • ELC (2 sources) • Capacitive touch sensing unit (2 sources) • Serial sound interface (2 sources) • Software trigger	126 sources • INT pin (18 sources): Choose low, falling edge, rising edge, or both edges • Timers (55 sources) • Communications (43 sources) • A/D conversion and comparator (7 sources) • ELC (2 sources) • Software trigger
Transfer units	8 or 16 bits	8, 16, or 32 bits	8, 16, or 32 bits
Transfer direction	Non-change → Non-change, Non-change → Forward, or Forward → Non-change	3 conditions selectable for transfer source and transfer destination • Non-change • Increment • Decrement	3 conditions selectable for transfer source and transfer destination • Non-change • Increment • Decrement
Transfer modes	Single transfer mode Repeat transfer mode	Normal transfer mode Repeat transfer mode Block transfer mode Chain transfer	Normal transfer mode Repeat transfer mode Block transfer mode Chain transfer Sequence transfer
Interrupts	Generated when the DMAi transfer counter underflows.	• Generated when the DTC starts • Generated after a single data transfer ends • Generated after the specified number of data units are transferred	• Generated when the DTC starts • Generated after a single data transfer ends • Generated after the specified number of data units are transferred

Note: 1. There is no concept of channels in the DTC. Transfer corresponding to the interrupt source is possible.

Table 1.3 Differences in the Number of Cycles Between the DMAC/DTC and the DMAC

Item	M16C (M16C)	RX (RX231/RX660)	
	DMAC	DMAC	DTC
Number of cycles *1	7 cycles (includes 1 dummy cycle)	6 cycles	18 cycles

Note: 1. This table assumes the following: 8-bit data transfer, transfer source is an I/O register (fixed), the transfer destination is the RAM (increment), the DTC is in full-address mode, and DTC vector table is the ROM, and ICLK = PCLK multiplied by 2.

2. Peripheral Functions Used

Table 2.1 shows Peripheral Functions and Modes Used in the Example for Operating the DMAC and DTC.

Table 2.1 Peripheral Functions and Modes Used in the Example for Operating the DMAC and DTC

No.	Operation	M16C		RX	
		Peripheral function	Mode	Peripheral function	Mode
1	Data transmission	DMAC	Single transfer mode	DMAC	Normal transfer mode
2	Data repeatedly transmitted		Repeat transfer mode		Repeat transfer mode
3	Data transmission		Single transfer mode	DTC	Normal transfer mode
4	Data repeatedly transmitted		Repeat transfer mode		Normal transfer mode

Also, asynchronous serial communications in the serial communications interface is used for the DMAC/DTC request source. Table 2.2 shows Transfer Conditions for the DMAC/DTC (Asynchronous Serial Communications).

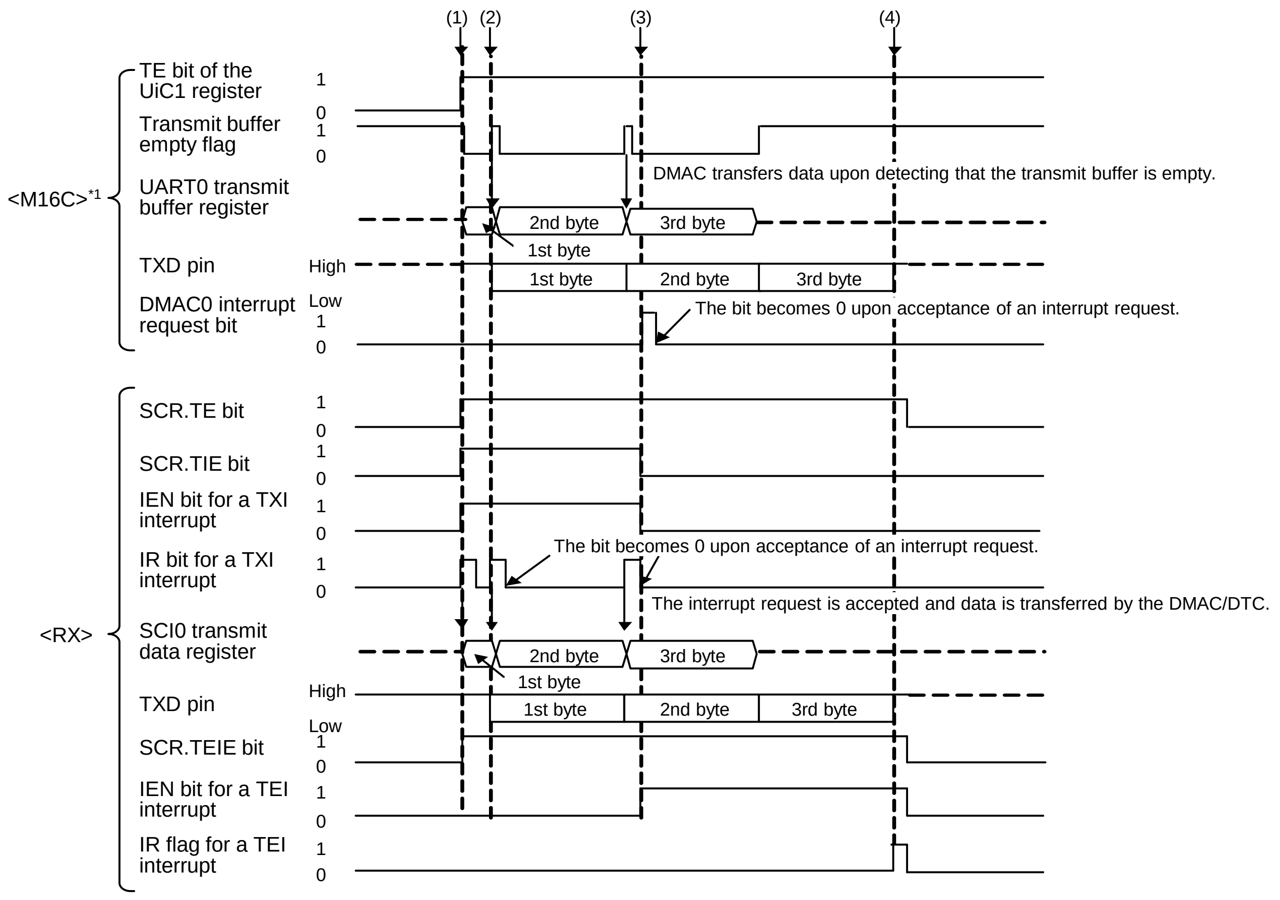
Table 2.2 Transfer Conditions for the DMAC/DTC (Asynchronous Serial Communications)

Item	Conditions for Transmission and Reception
Request source	Serial communications interface (SCI) Transmit data empty of the asynchronous serial communications
Channel used for asynchronous serial communications	RX Family: SCI0 M16C Family: UART0

3. Transfer Timing

Figure 3.1 shows Differences in Timing Between the RX and the M16C. Table 3.1 shows Comparison of Operation and Processing at Various Timings.

Both Figure 3.1 and Table 3.1 assume 3 bytes of serial data are transferred using the DMAC or DTC.



Note 1. For the M16C, only either the time at which the transmit buffer becomes empty or the time at which transmission ends can be used as a trigger to generate a transmit interrupt.
The above timing chart is an example of operation when the time at which the transmit buffer becomes empty is selected as a trigger.

Figure 3.1 Differences in Timing Between the RX and the M16C

Table 3.1 Comparison of Operation and Processing at Various Timings Between the RX and the M16C

Timing	M16C (M16C/65C)	RX (RX231, RX660)
(1) Transmission starts	Transmission is enabled. However, even if transmission is enabled, a transmit interrupt is not generated, so an instruction like the MOV instruction of the CPU must be used in the main processing in order to write the first byte of data to the transmit buffer.	When transmission is enabled, the transmit interrupt (TXI interrupt) is generated. The interrupt request triggers the DMAC/DTC to transfer the first byte of data to the transmit buffer.
(2) Transmit data is transferred to the transmit shift register	When a transmit interrupt request is generated, the DMAC transfers the second byte of data to the transmit buffer.	When a transmit interrupt request is generated, the DMAC/DTC transfers the second byte of data to the transmit buffer.
(3) DMA transfer interrupt (DMAC) or TXI interrupt (DTC) generated when the last data is written	DMAC interrupt request is generated.	When the last data is written to the transmit buffer, the DMA transfer interrupt (DMAC) or TXI interrupt (DTC) is generated. This interrupt enables the transmit end interrupt (TEI interrupt) and disables the transmit interrupt.
(4) After the last data is output	—	The transmit end interrupt is generated. In the interrupt handling, the transmit end interrupt and transmission are disabled. When transmission is disabled, the TXD pin becomes high-impedance.

4. Settings to Use the DTC

When using the DTC, the DTC vector table and DTC transfer information must be prepared. Figure 4.1 shows a Memory Map When the DTC Vector Table is Allocated to the ROM Area. The start address of the DTC transfer information is stored in the DTC vector table, and the DTC vector table is allocated to the ROM or RAM. The base address of the vector table is allocated so the lower 12 bits are 0. The start address of the DTC transfer information with vector number *n* for an interrupt to be used as an activation trigger is stored in the +4*n*-th address from the base address.

The DTC transfer information is allocated to the RAM, and Figure 4.2 shows a Memory Map and Structure of the DTC Transfer Information.

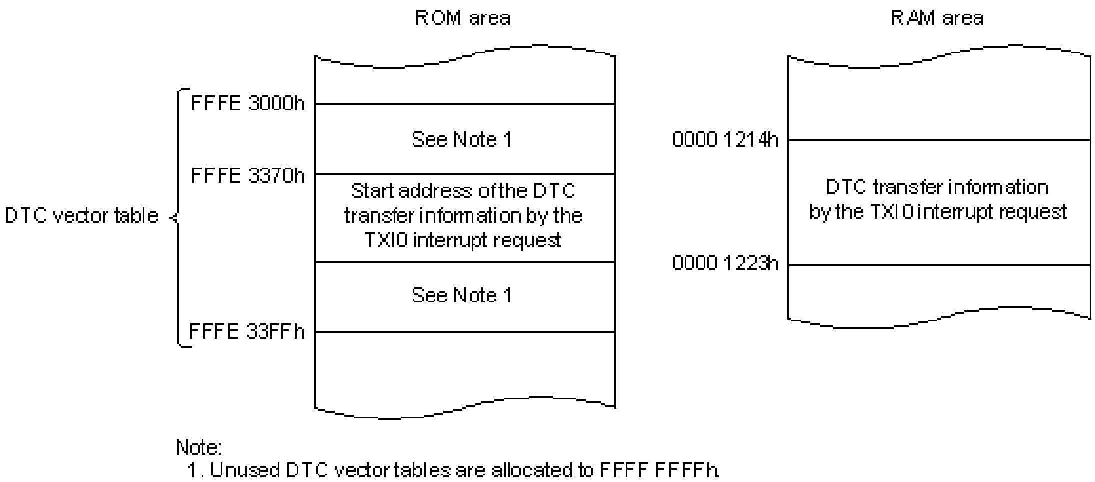
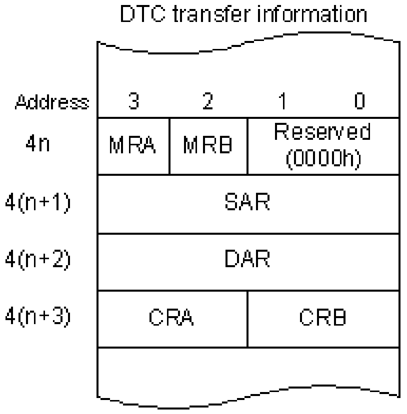


Figure 4.1 Memory Map When the DTC Vector Table is Allocated to the ROM Area




```
/* **** DTC Transfer information data **** */
#pragma bit_order left
#pragma unpack
struct st_dtc_full{
        union{
                unsigned long LONG;
                struct{
                        unsigned long MRA_MD :2;
                        unsigned long MRA_SZ :2;
                        unsigned long MRA_SM :2;
                        unsigned long :2;
                        unsigned long MRB_CHNE :1;
                        unsigned long MRB_CHNS :1;
                        unsigned long MRB_DISEL :1;
                        unsigned long MRB_DTS :1;
                        unsigned long MRB_DM :2;
                        unsigned long :2;
                        unsigned long :16;
                }BIT;
        }MR;
        void * SAR;
        void * DAR;
        struct{
                unsigned long CRA:16;
                unsigned long CRB:16;
        }CR;
}
#pragma bit_order
#pragma packoption
```

Figure 4.2 Memory Map and Structure of the DTC Transfer Information

5. Appendix

5.1 Points on Migration From the M16C Family to the RX Family

This chapter explains points on migration from the M16C Family to the RX Family.

5.1.1 Interrupts

For the RX Family, when an interrupt request is received while all of the following conditions are met, the interrupt occurs.

- The I flag (PSW.I bit) is 1.
- Registers IER and IPR in the ICU are set to enable interrupts.
- The interrupt request is enabled by the interrupt request enable bits for peripheral functions.

Table 5.1 shows Comparison of Conditions for Interrupt Generation Between the RX and the M16C.

Table 5.1 Comparison of Conditions for Interrupt Generation Between the RX and the M16C

Item	M16C	RX
I flag	When the I flag is set to 1 (enabled), the maskable interrupt request can be accepted.	
Interrupt request flag	When an interrupt request is generated by a peripheral function, the interrupt request flag becomes 1 (interrupt requested).	
Interrupt priority level	Selected by setting bits ILVL2 to ILVL0.	Selected by setting the IPR[3:0] bits.
Interrupt request enable	—	Specified by setting the IER register.
Interrupt enable for peripheral functions	—	Interrupts can be enabled or disabled in each peripheral function.

For more information, refer to sections Interrupt Controller (ICU), CPU, and sections for other peripheral functions used in the User's Manual: Hardware.

5.1.2 I/O Ports

In the RX Family, the MPC must be configured in order to assign I/O signals from peripheral functions to pins.

Before controlling the I/O pins in the RX Family, the following two items must be set.

- In the MPC.PFS register, select the peripheral functions that are assigned to the appropriate pins.
- In the PMR register for I/O ports, select the function for the pin to be used as a general I/O port or I/O port for a peripheral function.

Table 5.2 shows Comparison of I/O Settings for Peripheral Function Pins Between the RX and the M16C.

Table 5.2 Comparison of I/O Settings for Peripheral Function Pins Between the RX and the M16C

Function	M16C (in the case of the M16C/65C)	RX (in the case of the RX660/RX231)
Select the pin function	These are not available in the M16C. *1 When a mode is set for a peripheral function, appropriate pins are assigned as I/O pins for the peripheral function.	With the PFS register, I/O ports for peripheral functions can be assigned by selecting from multiple pins.
Switch between general I/O port and peripheral function		With the PMR register, the corresponding pin function can be selected as a general I/O port or a peripheral function.

Note: 1. Register for similar functions are available in the M32C Group and R32C Group.

For more information, refer to the Multi-Function Pin Controller (MPC) and I/O port sections in the User's Manual: Hardware.

5.1.3 Module Stop Function

The RX Family has the ability to stop each peripheral module individually.

By transitioning unused peripheral modules to the module stop state, power consumption can be reduced.

After a reset is released, all modules (with a few exceptions) are in the module stop state.

Registers for modules in the module stop state cannot be written to or read.

For more information, refer to the Low Power Consumption section in the User's Manual: Hardware.

5.2 I/O Register Macros

Macro definitions listed in Table 5.3 can be found in the RX I/O register definitions (iodefine.h).

The readability of programs can be achieved with these macro definitions.

Table 5.3 shows Macro Usage Examples.

Table 5.3 Macro Usage Examples

Macro	Usage Example
IR("module name", "bit name")	**IR(MTU0, TGIA0)** = 0 ; The IR bit corresponding to MTU0.TGIA0 is cleared to 0 (no interrupt request is generated).
DTCE("module name", "bit name")	**DTCE (MTU0, TGIA0)** = 1 ; The DTCE bit corresponding to MTU0.TGIA0 is set to 1 (DTC activation is enabled).
IEN("module name", "bit name")	**IEN(MTU0, TGIA0)** = 1 ; The IEN bit corresponding to MTU0.TGIA0 is set to 1 (interrupt enabled).
IPR("module name", "bit name")	**IPR(MTU0, TGIA0)** = 0x02 ; The IPR bit corresponding to MTU0.TGIA0 is set to 2 (interrupt priority level 2).
MSTP("module name")	**MSTP(MTU)** = 0 ; The MTU0 Module Stop bit is set to 0 (module stop state is canceled).
VECT("module name", "bit name")	#pragma interrupt (Excep_MTU0_TGIA0 (vect = **VECT(MTU0, TGIA0)**) The interrupt function is declared for the corresponding MTU0.TGIA0 register.

5.3 Intrinsic Functions

The RX Family has intrinsic functions for setting control registers and special instructions. When using intrinsic functions, include machine.h.

Table 5.4 shows Examples of Differences in the Settings of Control Registers and Descriptions of Special Instructions Between the RX and the M16C.

Table 5.4 Examples of Differences in the Settings of Control Registers and Descriptions of Special Instructions Between the RX and the M16C

Item	Description	
	M16C	RX
Set the I flag to 1	asm("fset i");	setpsw_i (); *1
Set the I flag to 0	asm("fclr i");	clrpsw_i (); *1
Expanded into the WAIT instruction	asm("wait");	wait(); *1
Expanded into the NOP instruction	asm("nop");	nop(); *1

Note: 1. The machine.h file must be included.

6. Reference Documents

User's Manual: Hardware

RX230/RX231 Group User's Manual: Hardware (R01UH0496EJ)
RX660 Group User's Manual: Hardware (R01UH0037EJ)
M16C/65C Group User's Manual: Hardware (R01UH0093EJ)
If you are using a product that does not belong to the RX231, RX660, or M16C/65C Group, refer to the applicable user's manual for hardware.
The latest versions can be downloaded from the Renesas Electronics website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools

RX Family CC-RX Compiler User's Manual (R20UT3248)
M16C Series, R8C Family C Compiler Package V5.45 (M3T-NC30WA)
The latest versions can be downloaded from the Renesas Electronics website.

REVISION HISTORY

Rev.	Date	Description	
		Page	Summary
1.00	Sep. 22, 2014	—	First edition issued
2.00	June 16, 2023	—	The product model of the target device for the RX MCU was changed: From RX210 to RX231/RX660

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Corporate Headquarters

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

Trademarks

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

Contact information

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