RX65N, H8SX/1668

Serial Communications Interface Migration Guide: H8SX/1668 to RX65N

Introduction

This application note describes the differences in the serial communications interface (SCI) between the RX65N and H8SX/1668 devices.

Target Devices

RX65N

H8SX/1668

Contents

1. Features

Table 1.1 shows the features of the SCIg modules* of the RX65N and H8SX/1668 devices. Differences between the devices are shaded.

Table 1.1 Features of the SCIg Module (1/2)

Item		Specifications	
		RX65N	H8SX/1668
Number of channels		10: SCI0, SCI1, SCI2, SCI3, SCI4, SCI5, SCI6, SCI7, SCI8, and SCI9	6: SCI_0, SCI_1, SCI_2, SCI_4, SCI_5, and SCI_6
Communication mode		5 modes: asynchronous, clock synchronous, smart card, simplified I2C, and simplified SPI	3 modes: asynchronous, clock synchronous, and smart card
Full duplex communication		Available. Both the transmitter and receiver are double-buffered.	
Data transfer		MSB-first or LSB-first can be selected.	
Interrupt source		Transmit end, transmit data empty, receive data full, receive error, and end of generating start conditions, restart conditions, or stop conditions (for simplified I2C mode)	Transmit end, transmit data empty, receive data full, and receive error
Low power consumption		The module stop state can be set for each channel.	
Asynchronous mode	Channel	All	
	Data length	7, 8, or 9 bits	7 or 8 bits
	Transmission stop bit	1 bit or 2 bits	
	Parity	Even, odd, or none	
	Receive error detection	Parity, overrun, and framing errors	
	Hardware flow control	Transmission and reception can be controlled by using the CTSn# and RTSn# pins.	
	Start bit detection	The Low level or falling edge can be selected.	The Low level is detected.
	Break detection	• A break can be detected by directly reading the level of the RxDn pin when a framing error occurs. • A break can be detected by reading the SPTR.RXDMON flag.	• A break can be detected by directly reading the level of the RxD pin when a framing error occurs.
	Clock source	The internal or external clock can be selected. The transfer rate clock can be input from TMR (SCI5 or SCI6).	
	Double-speed mode	The baud rate generator double-speed mode can be selected.	
	Multiprocessor communication	Serial communications among multiple processors are possible.	
	Noise elimination	On-chip digital noise filter on the RXDn pin input route	

Table 1.1 Features of the SCIg Module (2/2)

Item		Specifications	
		RX65N	**H8SX/1668**
Clock synchronous mode	Channel	All	0, 1, 2, 4
	Data length	8 bits	
	Receive error detection	Overrun errors	
	Hardware flow control	Transmission and reception can be controlled by using the CTSn# and RTSn# pins.	
Smart card interface mode	Error handling	• An error signal is automatically transmitted when a parity error is detected during reception. • Data is automatically resent when an error signal is received during transmission.	
	Data type	Both direct convention and inverse convention are supported.	
IrDA mode		Not available.	Available with SCI5 only.
Simplified I2C mode	Data length	8 bits	
	Error detection	Overrun errors	
	SS input pin function	The output pin can be driven at high impedance when the SSn# pin is at High level.	
	Clock settings	The clock phase and polarity settings can be selected from four options.	
Bit rate modulation		The error can be reduced by correcting the output of the on-chip baud rate generator.	
Event linkage (available with SCI5 only)		Output of error events (detection of receive errors and error signals)	
		Output of the receive data full event	
		Output of the transmit data empty event	
		Output of the transmit end event	

Note: * The RX651 device has the SCIi (SCI10 or SCI11) and SCIh (SCI12) modules, as well as the SCIg module. For details on the SCHi and SHCh modules, refer to the chapter on serial communications interfaces in the manual “RX65N Group, RX651 Group User’s Manual: Hardware”.

2. General Notes

The RX65N device does not support IrDA. To use an application that needs IrDA support, the external IrDA controller must be added.

For the RX65N device, all of its 10 channels support clock synchronous mode. For the HS8X/1668 device, only four channels support clock synchronous mode.

3. References

- Hardware manual for the RX65N:
  R01UH0590EJ0230: RX65N Group, RX651 Group User's Manual: Hardware
- Software manual for the RX65N:
  R01US0071EJ0100: RX Family RXv2 Instruction Set Architecture User's Manual: Software

(The latest versions of the above manuals are available on the Renesas website.)

3.1 Related Chapters in the Hardware Manual

- Clock Generation Circuit
  Provides details on how to set up the peripheral clocks used for the SCI.
- I/O Registers
  Shows a list of all registers.
- Low Power Consumption
  Provides details on the module stop control registers.
- Interrupt Controller
  Describes how to enable interrupts from the SCI to the interrupt controller.
- I/O Ports
  Provides details on the interrupt control registers and port function registers related to SCI-related pins.

3.2 Related Registers

The following table lists the registers related to the operation of the serial communications interfaces (SCIg, SCIi, and SCIh) of the RX65N.

Table 3.1 Registers Related to the Operation of the Serial Communications Interfaces (1/2)

Name	Description	Chapter in the Hardware Manual
SYSTEM.SCKCR	System clock control register	Clock Generation Circuit
SYSTEM.MSTPCRB	Module stop control register B	Low Power Consumption
ICU.IRx	Interrupt request register	Interrupt Controller
ICU.IERx	Interrupt request enable register	
ICU.IPRx	Interrupt source priority register	
PORTx.PDR	Port direction register	I/O Ports
PORTx.PMR	Port mode register	
MPC.PWPR	Write protection register	Multi-Function Pin Controller
MPC.PxxPFS	Pin function control register	
SCIx.RSR	Receive shift register	Serial Communications Interface
SCIx.RDR	Receive data register	
SCIx.RDRH	Receive data register H	
SCIx.RDRL	Receive data register L	
SCIx.RDRHL	Receive data register HL	
SCI10.FRDR SCI11.FRDR	Receive FIFO data register	
SCIx.TDR	Transmit data register	
SCIx.TDRH	Transmit data register H	
SCIx.TDRL	Transmit data register L	
SCIx.TDRHL	Transmit data register HL	
SCI10.FTDR SCI11.FTDR	Send FIFO data register	
SCIx.SMR	Serial mode register	
SCIx.SCR	Serial control register	
SCIx.SSR/SSRFIFO	Serial status register	
SCIx.SCMR	Smart card mode register	
SCIx.BRR	Bit rate register	
SCIx.MDDR	Modulation duty register	
SCIx.SEMR	Serial extended mode register	
SCIx.SNFR	Noise filter setting register	
SCIx.SIMR1	I2C mode register 1	
SCIx.SIMR2	I2C mode register 2	
SCIx.SIMR3	I2C mode register 3	
SCIx.SISR	I2C status register	
SCIx.SPMR	SPI mode register	
SCI10.FCR SCI11.FCR	FIFO control register	
SCI10.FDR SCI11.FDR	FIFO data count register	
SCI10.LSR SCI11.LSR	Line status register	
SCI10.CDR SCI11.CDR	Comparison data register	

Table 3.1 Registers Related to the Operation of the Serial Communications Interfaces (2/2)

Name	Description	Chapter in the Hardware Manual
SCI10.DCCR SCI11.DCCR	Data comparison control register	Serial Communications Interface
SCI10.SPTR SCI11.SPTR	Serial port register	
SCI12.ESMER	Extended serial mode enable register	
SCI12.CR0	Control register 0	
SCI12.CR1	Control register 1	
SCI12.CR2	Control register 2	
SCI12.CR3	Control register 3	
SCI12.PCR	Port control register	
SCI12.ICR	Interrupt control register	
SCI12.STR	Status register	
SCI12.STCR	Status clearing register	
SCI12.CF0DR	Control Field 0 data register	
SCI12.CF0CR	Control Field 0 compare enable register	
SCI12.CF0RR	Control Field 0 receive data register	
SCI12.PCF1DR	Primary Control Field 1 data register	
SCI12.SCF1DR	Secondary Control Field 1 data register	
SCI12.CF1CR	Control Field 1 compare enable register	
SCI12.CF1RR	Control Field 1 receive data register	
SCI12.TCR	Timer control register	
SCI12.TMR	Timer mode register	
SCI12.TPRE	Timer prescaler register	
SCI12.TCNT	Timer count register	

4. Summary of Differences of Registers

Table 4.1 lists the SCI-related registers of the RX65N. The registers that have been changed from those of the H8SX/1668 are shaded in the table. For details on the changed registers, refer to the relevant sections in this chapter. For details on the other registers, refer to the hardware manual for the RX65N.

Table 4.1 SCI-Related Registers (1/2)

Register Name	Symbolic Name
Receive shift register	RSR
Receive data register	RDR
Receive data register H	RDRH
Receive data register L	RDRL
Receive data register HL	RDRHL
Receive FIFO data register	FRDR
Transmit data register	TDR
Transmit data register H	TDRH
Transmit data register L	TDRL
Transmit data register HL	TDRHL
Send FIFO data register	FTDR
Serial mode register	SMR
Serial control register	SCR
Serial status register	SSR
Smart card mode register	SCMR
Bit rate register	BRR
Modulation duty register	MDDR
Serial extended mode register	SEMR
Noise filter setting register	SNFR
I2C mode register 1	SIMR1
I2C mode register 2	SIMR2
I2C mode register 3	SIMR3
I2C status register	SISR
SPI mode register	SPMR
FIFO control register	FCR
FIFO data count register	FDR
Line status register	LSR
Comparison data register	CDR
Data comparison control register	DCCR
Serial port register	SPTR
Extended serial mode enable register	ESMER
Control register 0	CR0
Control register 1	CR1
Control register 2	CR2
Control register 3	CR3
Port control register	PCR
Interrupt control register	ICR
Status register	STR
Status clearing register	STCR
Control Field 0 data register	CF0DR
Control Field 0 compare enable register	CF0CR
Control Field 0 receive data register	CF0RR
Primary Control Field 1 data register	PCF1DR

Table 4.1 SCI-Related Registers (2/2)

Register Name	Symbolic Name
Secondary Control Field 1 data register	SCF1DR
Control Field 1 compare enable register	CF1CR
Control Field 1 receive data register	CF1RR
Timer control register	TCR
Timer mode register	TMR
Timer prescaler register	TPRE
Timer count register	TCNT

4.1 Changes to the Serial Mode Register (SMR)

As shown below, bits renamed during migration from the H8SX to the RX65N are shaded. However, the functions of these bits do not change.

- SMR (for the RX65N)
  (when SCMR.SMIF = 0)

b7	b6	b5	b4	b3	b2	b1	b0
CM	CHR	PE	PM	STOP	MP	CKS[1:0]	

- SMR (for the H8SX/1668)
  (when SCMR.SMIF = 0)

b7	b6	b5	b4	b3	b2	b1	b0
C/-A	CHR	PE	O/-E	STOP	MP	CKS[1:0]	

- SMR (for the RX65N)
  (when SCMR.SMIF = 1)

b7	b6	b5	b4	b3	b2	b1	b0
GM	BLK	PE	PM	BCP[1:0]		CKS[1:0]	

- SMR (for the H8SX/1668)
  (when SCMR.SMIF = 1)

b7	b6	b5	b4	b3	b2	b1	b0
GM	BLK	PE	O/-E	BCP[1:0]		CKS[1:0]	

4.2 Changes to the SCI Smart Card Mode Register (SCMR)

On the RX56N, the BCP2 and CHR1 bits have been added.

- SCMR (for the RX65N)

b7	b6	b5	b4	b3	b2	b1	b0
BCP2	—	—	CHR1	SDIR	SINV	—	SMIF

- SCMR (for the H8SX/1668)

b7	b6	b5	b4	b3	b2	b1	b0
—	—	—	—	SDIR	SINV	—	SMIF

4.3 Changes to the Serial Extended Mode Register (SEMR_2/SEMR)

The name of this register was changed from "SEMR_2" to "SEMR".

The position of the ABCS bit was changed from "b3" to "b4".

The ACS2 and ACS1 bits (average transfer rate function) were deleted.

The RXDESEL, BGDM, and NFEN bits were added to this register of the RX65N.

- SEMR (for the RX65N)

b7	b6	b5	b4	b3	b2	b1	b0
RXDESEL	BGDM	NFEN	ABCS	—	BRME	—	ACS0

- SEMR_2 (for the H8SX/1668)

b7	b6	b5	b4	b3	b2	b1	b0
—	—	—	—	ABCS	ACS[2:0]		

5. Software Details

5.1 Summary of the Software Migration Procedure

1. Change register name references where applicable (if the Renesas iodefine.h file is used).
2. Adjust the baud rate if the peripheral clock rate has been changed.
3. Adjust the port pin settings.
4. Adjust the module stop control registers.
5. Touch up interrupt setup and handler code.

5.2 Step 1: Change Register Name References

The Renesas toolchain includes utilities that automatically generate header files containing the register definitions of the hardware resources for the target chip. When migrating code, carefully note that the underlying structures used to reference these resources in some cases have changed. These definitions are contained in the iodefine.h file.

With the iodefine.h file generated for an H8SX family device, separate structures are defined for the SCI registers for different modes of operation. For example, the structure “st_sci” is used to define the registers for an SCI in normal mode (smart code mode disabled), while the structure “st_smci” is used to reference the same registers when the SCI is in smart code mode.

A different approach to defining structures for referencing control registers is taken for the RX series of parts: A single structure is defined for a peripheral such as a serial port. Within that structure where a control register has two different bit maps based on the mode of the peripheral, a union of two structures is created.

- Register changes (SMIF in SMCR = 0, smart card interface disabled)
- From the H8SX To the RX65N

```
SCIx.SMR.BIT.CA          SCIx.SMR.BIT.CM
SCIx.SMR.BIT.CHR         SCIx.SMR.BIT.CHR
SCIx.SMR.BIT._PE         SCIx.SMR.BIT.PE
SCIx.SMR.BIT.OE          SCIx.SMR.BIT.PM
SCIx.SMR.BIT.STOP        SCIx.SMR.BIT.STOP
SCIx.SMR.BIT.MP          SCIx.SMR.BIT.MP
SCIx.SMR.BIT.CKS         SCIx.SMR.BIT.CKS
```

- Register changes (SMIF in SMCR = 1, smart card interface enabled)
- From the H8SX To the RX65N

```
SMCIx.SMR.BIT.GM         SCIx.SMR.BIT.GM
SMCIx.SMR.BIT.CHR        SCIx.SMR.BIT.BLK
SMCIx.SMR.BIT._PE        SCIx.SMR.BIT.PE
SMCIx.SMR.BIT.OE         SCIx.SMR.BIT.PM
SMCIx.SMR.BIT. BCP       SCIx.SMR.BIT.BCP
SMCIx.SMR.BIT.CKS        SCIx.SMR.BIT.CKS
```

With new macros defined in the iodefine.h file for RX family members, The ICU control register, module stop register, DTC enable register, and interrupt vector numbers can easily be referenced by using the logical name associated with a peripheral module. These macros allow specific registers and vector numbers to be hidden, thus achieving migration between RX family members. For details, refer to the code contained in iodefine.h.

Macro	Usage Example
IR(<module-name>, <bit-name>)	if (IR(SCI0, TXI0)) == 1)…
IEN(<module-name>, <bit-name>)	IEN(SCI0, TXI0) = 1 ;
IPR(<module-name>, <bit-name>)	IPR(SCI0, TXI0) = 0x02 ;
MSTP(<module-name>)	MSTP(SCI0) = 0 ;
VECT(<module-name>, <bit-name>)	#pragma interrupt MySciTxIsr(vect=VECT(SCI0, TXI0))

5.3 Step 2: Adjust the Baud Rate If the Peripheral Clock Rate Has Been Changed

The peripheral clock (PCLK) provides the SCI's time base. By accelerating the clock of the RX65N cores, the peripheral module clock can operate at a maximum of 60 MHz (120 MHz, with SCIi). The maximum peripheral clock frequency of the H8SX/1668 is 35 MHz. In applications that use improved peripheral clock performance, the settings of the timing parameters related to baud rate generation must be adjusted. As a result, the minimum and maximum allowable baud rates may change.

The RX65N's improved clock generation circuit and high-speed core enable applications to take advantage of increased performance while minimizing the migration effort for timing-related changes to software for peripheral modules. RX65N system clock speed, which governs code execution speed, can be doubled over H8SX/1668 applications while maintaining the same peripheral clock speed. Code can run twice as fast with minimal impact to driver code for peripheral modules.

5.4 Step 3: Adjust Port Pin Settings

Because the allocation of port pins has been changed, perform the following operations for the serial ports to be used: Set the corresponding bit of the port mode register (PMR) so that the pin is used as an I/O port for peripheral modules; and configure the multi-function pin controller (MCP) so that the pin is used as an I/O port for serial communications.

5.5 Step 4: Adjust the Module Stop Control Registers

To maximize the power efficiency, the on-chip peripheral functions of the H8SX and RX cores can be individually stopped by performing a write to the corresponding module stop control registers. By default, the serial port is inactivated after the processor is reset. Therefore, to use the serial port, it must be enabled. The bit that determines the state of the SCI channel has been changed. Therefore, when migrating the application code from the H8SX/1668 to the RX65N, this change must also be applied accordingly.

- From the H8SX To the RX65N

  SCI0:
  ```
  MSTP.CRB.BIT._SCI0        MSTP(SCI0)
  ```
  SCI1:
  ```
  MSTP.CRB.BIT._SCI1        MSTP(SCI1)
  ```
  SCI2:
  ```
  MSTP.CRB.BIT._SCI2        MSTP(SCI2)
  ```
  SCI4:
  ```
  MSTP.CRB.BIT._SCI4        MSTP(SCI4)
  ```
  SCI5:
  ```
  MSTP._CRC.BIT._SCI5       MSTP(SCI5)
  ```
  SCI6:
  ```
  MSTP._CRC.BIT._SCI6       MSTP(SCI6)
  ```

5.6 Step 5: Touch Up Interrupt Setup and Handler Code

5.6.1 Enabling Interrupt Sources

If there is an application that uses interrupts for SCI communications, its code must be modified. The step described in this section must be added during SCI initialization so that the appropriate interrupt source is enabled to set the priority level by using the RX65N interrupt controller (ICU). For details, refer to the chapter on the interrupt controller in the manual “RX65N Group, RX651 Group User’s Manual: Hardware”.

- The following shows the code that must be included to enable interrupts for the SCI0:

```
IEN(SCI0,RXI0)      = 1U;     // Enable Rx interrupts
IEN(SCI0,TXI0)      = 1U;     // Enable Tx interrupts
ICU.GENBL0.BIT.EN0 = 1U;     // Enable Tx complete interrupts
ICU.GENBL0.BIT.EN1 = 1U;     // Enable Rx error interrupts

IPR(SCI0,RXI0)      = 0x01;  // Set priority for SCI0 RXI0 interrupts
IPR(SCI0,TXI0)      = 0x01;  // Set priority for SCI0 TXI0 interrupts
IPR(ICU,GROUPBL0)  = 0x01;  // Set priority for GENBL0 interrupts
```

5.6.2 Setting Interrupt Priority Levels

Each SCI channel has an associated interrupt priority that must be set. Active interrupts of a priority higher than the current interrupt in the CPU’s PSW will be fired. Interrupt priorities are set using the macros defined in the iodefine.h file.

Table 5.1 Interrupt Source Priority Registers

SCI Channel	Interrupt Source Priority Register (IPR)	Macro Defined in IODEFINE.H
RXI0	IPR58	IPR(SCI0, RXDI0) = <priority>
TXI0	IPR59	IPR(SCI0, TXDI0) = <priority>
RXI1	IPR60	IPR(SCI1, RXDI1) = <priority>
TXI1	IPR61	IPR(SCI1, TXDI1) = <priority>
RXI2	IPR62	IPR(SCI2, RXDI2) = <priority>
TXI2	IPR63	IPR(SCI2, TXDI2) = <priority>
RXI3	IPR80	IPR(SCI3, RXDI3) = <priority>
TXI3	IPR81	IPR(SCI3, TXDI3) = <priority>
RXI4	IPR82	IPR(SCI4, RXDI4) = <priority>
TXI4	IPR83	IPR(SCI4, TXDI4) = <priority>
RXI5	IPR84	IPR(SCI5, RXDI5) = <priority>
TXI5	IPR85	IPR(SCI5, TXDI5) = <priority>
RXI6	IPR86	IPR(SCI6, RXDI6) = <priority>
TXI6	IPR87	IPR(SCI6, TXDI6) = <priority>

5.6.3 Adjusting the Interrupt Protocol

The use of the "__interrupt" keyword to identify an interrupt service routine (ISR) is no longer supported in the latest versions of the Renesas C compiler. Prototypes for ISR's should now use the "#pragma interrupt" directive. Rather than using vector numbers, use the macros defined in the iodefine.h file to ensure portability across RX family members:

- Old syntax:

```
__interrupt(vect=215) void INT_RXI0_SCI0(void)
```

- New syntax:

```
#pragma interrupt (INT_RXI0(vect=VECT(SCI0,RXI0)))
void INT_RXI0 (void) ;
```

5.6.4 Adjusting Vector Numbers

Interrupt vector numbers for the RX65N are different from the vector numbers for the H8SX. When the application code is compiled using the Renesas compiler, interrupt service routines written in C language are hooked to specific interrupts vectors using the #pragma interrupt directive as follows:

```
#pragma interrupt (INT_RXI0(vect=145))
void INT_RXI0 (void) ;
```

The vector number following "vect=" needs to be changed to the new vector number for the RX65N according to Table 5.2. The VECT macro defined in the iodefine.h file provides a syntax that allows easy migration to other RX-family devices without changing the application code.

Table 5.2 Interrupt Vector Numbers

Interrupt Source	H8SX Vector	RX65N	Macro Defined in iodefine.h
SCI0 receive	145	58	`VECT(SCI0, RXI0)`
SCI0 transmit	146	59	`VECT(SCI0, TXI0)`
SCI1 receive	149	60	`VECT(SCI1, RXI1)`
SCI1 transmit	150	61	`VECT(SCI1, TXI1)`
SCI2 receive	153	62	`VECT(SCI2, RXI2)`
SCI2 transmit	154	63	`VECT(SCI2, TXI2)`
SCI4 receive	161	82	`VECT(SCI4, RXI4)`
SCI4 transmit	162	83	`VECT(SCI4, TXI4)`
SCI5 receive	220	84	`VECT(SCI5, RXI5)`
SCI5 transmit	221	85	`VECT(SCI5, TXI5)`
SCI6 receive	224	86	`VECT(SCI6, RXI6)`
SCI6 transmit	225	87	`VECT(SCI6, TXI6)`
SCI0 receive error	144	110	`VECT(ICU,GROUPBL0)`
SCI0 transmit end/complete	147		
SCI1 receive error	148		
SCI1 transmit end/complete	151		
SCI2 receive error	152		
SCI2 transmit end/complete	155		
SCI4 receive error	160		
SCI4 transmit end/complete	163		
SCI5 receive error	222		
SCI5 transmit end/complete	223		
SCI6 receive error	226		
SCI6 transmit end/complete	227		

6. Usage Notes

6.1 RX Smart Configurator

On an RX-family device, RX Smart Configurator can be used when creating code for the SCI. With RX Smart Configurator, when a user selects or sets the SCI function from the GUI, the corresponding driver code is automatically generated. When you migrate to an RX-family device, we recommend that you use Smart Configurator.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar. 27, 2023	—	First edition issued

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

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/.