RX610, RX62N, RX621 Group

Quick Design Guide

R01AN0622EU0140
Rev.1.40
Dec.15, 2011

Introduction

The purpose of this document is to help answer common questions and point out subtleties of the MCU that might be missed unless the hardware manual was extensively reviewed. The document is not intended to be a replacement for the hardware manual; it is intended to supplement the manual by highlighting some key items most engineers will need to start their own design. It also discusses some design decisions from an application point of view.

Target Device

RX600 Group

RX62N Group

RX621 Group

Contents

1. Power Supplies 2

2. MCU Operating Modes 3

3. Endianness 4

4. Clock Circuits 7

5. Reset Requirements and the Reset Circuit 10

6. Memory 13

7. I/O Register Structures 15

8. I/O Port Configuration 18

9. Module Stop Function 21

10. Interrupts 22

11. Low Power Consumption 25

12. Emulator Support 27

13. References 29

1. Power Supplies

The RX family has digital power supplies and analog power supplies. The power supplies use the following pins (see Figure 1.1 - RX Power Supply).

Digital Power Supplies

Symbol	Name	Description
VCC	Power supply	3.3V power supply
VSS	Ground	Ground
VCL	Core voltage	Connect this pin to VSS with a 0.1uF capacitor. The capacitor should be placed close to the pin. Do not apply voltage to this pin.
PLLVCC	PLL power supply	Power supply for the PLL circuit. Connect this pin to VCC.
PLLVSS	PLL power ground	Ground for PLL. Connect this pin to VSS.
CNVSS		Connect this pin to VSS via a 10K pull-down resistor.
VCC_USB	USB power supply	USB power supply pin. Connect this pin to VCC. If USB is not used, it is safe to omit the 10uF cap on VCC_USB in figure 1.1.
VCC_VSS	USB ground	USB ground pin. Connect this pin to VSS.

Analog Power Supplies

Symbol	Name	Description
AVCC	Analog power supply	Analog supply pin for the A/D and D/A converters. When the A/D and D/A converters are not in use, connect this pin to VCC.
AVSS	Analog ground	Ground pin for the A/D and D/A converters. Connect this pin to the system power supply (0 V).
VREFH	A/D high reference voltage	Reference power supply pin for the A/D and D/A converters. When the A/D and D/A converters are not in use, connect this pin to the system power supply.
VREFL	A/D low reference voltage	Reference ground pin for the A/D and D/A converters. Connect this pin to the analog ground (AVSS). When the A/D and D/A converters are not in use, connect this pin to the system power supply (VSS).

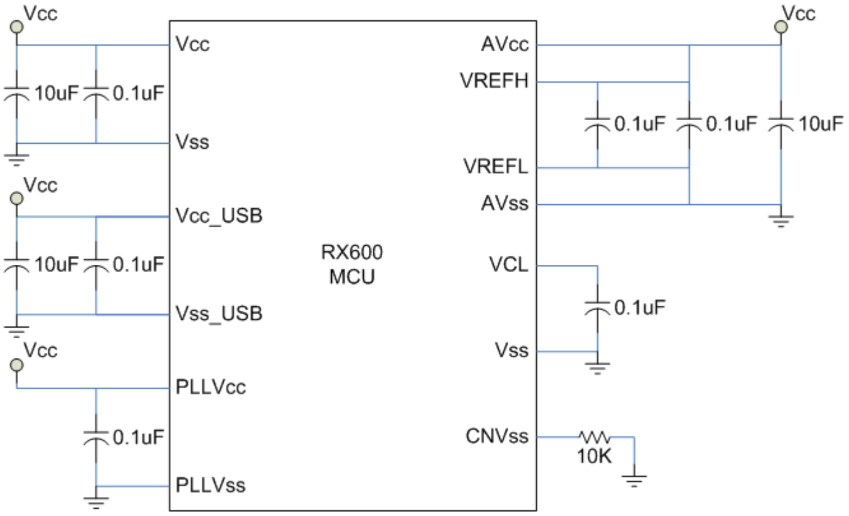


Figure 1.1 - RX Power Supply

2. MCU Operating Modes

The RX610, RX62N, and RX621 offer 5 different operating modes. Three of these modes are available immediately after reset; two additional modes may be selected by application software. The state of the MD1 and MD0 pins determine the processor's initial operating mode after reset. Details of these 3 modes are shown in the tables below. The MDMONR register can be used by application software to read the state of MD1, MD0 and MDE.

Table 2.1 - Operating Modes Available at Reset – RX610

Mode	MD0	MD1	Execution starts at	Memory available for P/E
Boot Mode	1	0	Serial Bootloader *[1]	• User Flash Area • Data Flash Area • User Boot Area
User Boot Mode	0	1	Address located at 0xFF7FFFFC	• User Flash Area • Data Flash Area
Single-Chip Mode	1	1	Address located at 0xFFFFFFFC	• User Flash Area • Data Flash Area

Notes: 1. This is the embedded bootloader that comes from the factory. It cannot be read or modified.

Table 2.2 - Operating Modes Available at Reset – RX62N/621

Mode	MD0	MD1	Execution starts at	Memory available for P/E
Boot Mode	1	0	Serial Bootloader *[1]	• User Flash Area • Data Flash Area
USB Boot Mode	0	1	USB Bootloader *[1]	• User Flash Area • Data Flash Area
Single-Chip Mode	1	1	Address located at 0xFFFFFFFC	• User Flash Area • Data Flash Area

Notes: 1. This is the embedded bootloader that comes from the factory. It cannot be read or modified.

User Boot Mode on the RX610, along with the 16KB User Boot Mat flash area, provides the user with a convenient way to implement a custom bootloader. For more information on this refer to the "Simple Flash API for RX" application note. On the RX62N, RX621 the user boot area has been replaced with an integrated factory USB bootloader.

The MCU can transition into 2 other operating modes after reset by modifying the ROME and EXBE bits in the System Control Register 0 (SYSCR0). Clearing the ROME bit disables the on-board flash ROM areas. Setting the EXBE bit enables the external memory bus. The table below shows the details of each mode.

Table 2.3 - Software Selectable Operating Modes

Mode	ROME	EXBE	On-Chip ROM	External Bus
After Reset	1	0	Enabled	Disabled
On-Chip ROM Enabled Extended Mode	1	1	Enabled	Enabled
On-Chip ROM Disabled Extended Mode	0	1	Disabled *[1]	Enabled

Notes: 1. After disabling the On-Chip ROM by clearing the ROME bit, it cannot be re-enabled.

3. Endianness

While the RX CPU's instructions are always little endian, the treatment of data is selectable as little endian or big endian. In little endian storage, the least-significant byte of multi-byte data is stored at the lower address. Big endian storage places the most-significant byte at the lower address. See Figure 3.1 – Big and Little Endian Data.

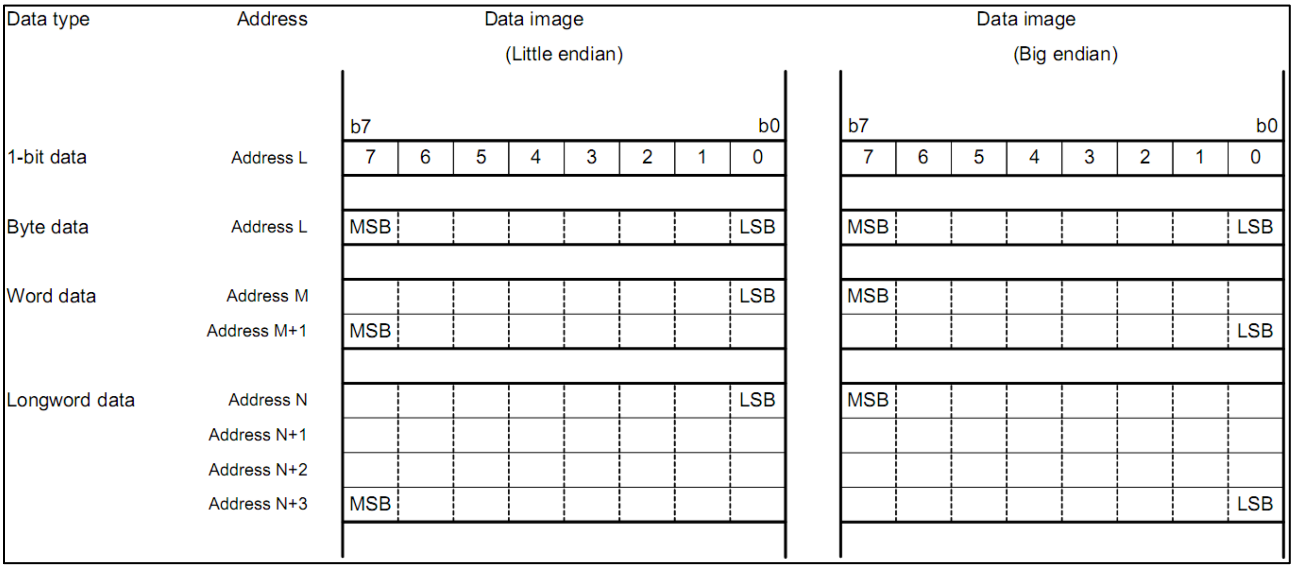


Figure 3.1 – Big and Little Endian Data

3.1 Configuring the MCU's Endianness

The voltage level on the MDE pin at reset determines run-time endianness of the MCU (see Table 3.1 - MDE Pin Settings). MDE must not change once the processor is running. Performance is unaffected by choice of big or little endian mode; it is offered as a convenience for developers migrating software from various platforms. Access to peripheral I/O register access is also unaffected by this setting.

Table 3.1 - MDE Pin Settings

Mode Pin MDE	Endian
0	Little endian
1	Big endian

3.2 Memory External to the MCU

Some members of the RX family include external bus controllers and SDRAM controllers. Memory accesses for devices connected to these external buses may be configured by software for big endian or little endian access. The access mode is set on a per-chip-select basis by setting the EMODE bit in the chip select control register (CSiCNT) or the SDC mode register (SDCMOD). Note that the EMODE bit changes the access **relative to** the MCU's main operating mode (see Table 3.2 - Setting Endianness for External Devices).

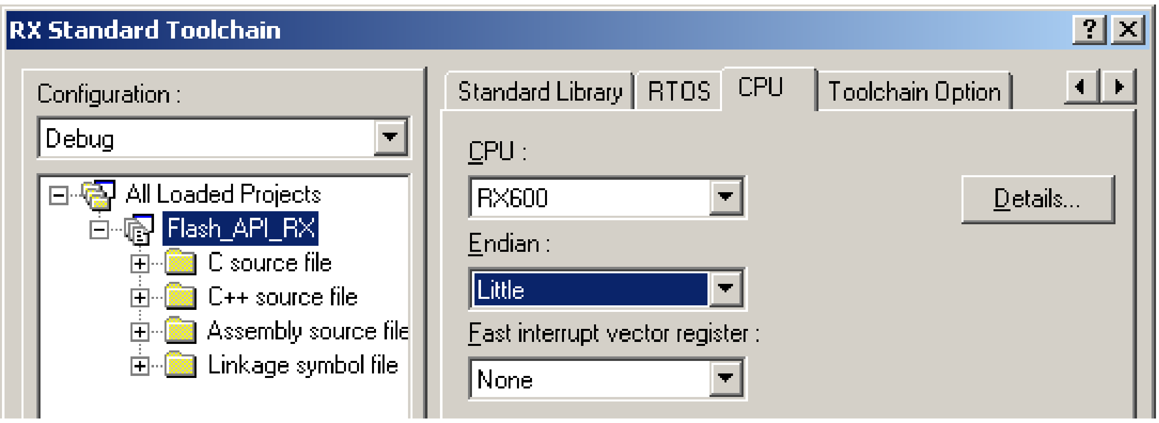
Table 3.2 - Setting Endianness for External Devices

MDE	EMODE	Access to external devices
0	0	Little endian
0	1	Big endian
1	0	Big endian
1	1	Little endian

3.3 Configuring Endianness in the Tool Chain

The tool chain in HEW must be set to match the target hardware with regard to endianness. The following steps configure the tool chain properly:

1. Open the appropriate workspace in HEW.
2. Under the “Build” menu, select “RX Standard Toolchain…”.
3. Select the ‘CPU’ tab (this may require scrolling the top tabs to the right)
4. Select ‘Little’ or ‘Big’ as appropriate from the ‘Endian’ drop down box.


RX Standard Toolchain

Configuration : Debug

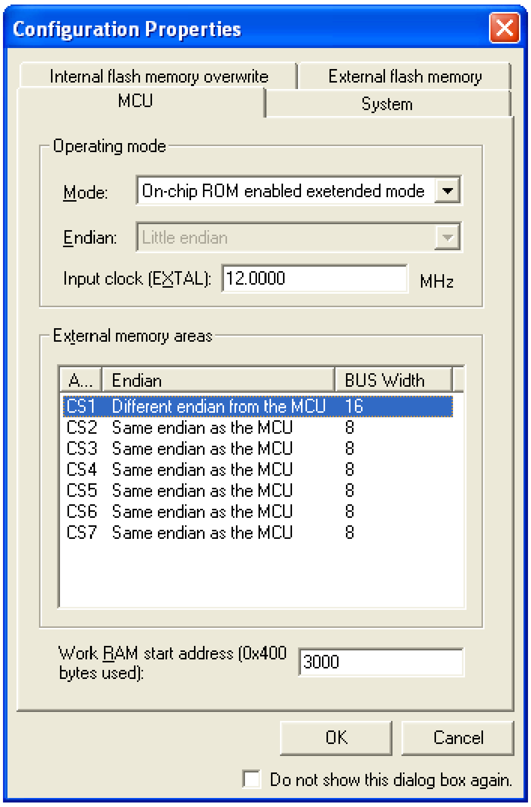
CPU : RX600

Endian : Little

Fast interrupt vector register : None

If the target has devices connected to the external buses, the debugger must be configured to access these correctly:

1. From the Debug menu, choose “Connect” or “Initialize” (only one will be active).
2. From the debugger “Configuration Properties” window, set the Operating Mode to “On-chip ROM enabled extended mode” (see Figure 3.2 - Configuring External Buses).
3. Double click on each active chip select to set the endian mode and bus width for each.



Configuration Properties

Operating mode

Mode: On-chip ROM enabled exetended mode

Endian: Little endian

Input clock (EXTAL): 12.0000 MHz

External memory areas

A...	Endian	BUS Width
CS1	Different endian from the MCU	16
CS2	Same endian as the MCU	8
CS3	Same endian as the MCU	8
CS4	Same endian as the MCU	8
CS5	Same endian as the MCU	8
CS6	Same endian as the MCU	8
CS7	Same endian as the MCU	8

Work RAM start address (0x400 bytes used): 3000

□ Do not show this dialog box again.

Figure 3.2 - Configuring External Buses

4. Clock Circuits

The main clock inputs for members of the RX family are the EXTAL and XTAL pins. The Clock Generation Circuit (CGC) uses a PLL and divider to provide frequencies of EXTAL, (EXTAL x 2), (EXTAL x 4), and (EXTAL x 8) to internal clock domains. A maximum CPU frequency of 100 MHz is achieved using a 12.5 MHz crystal. Depending on the peripheral mix needed by the application, a lower CPU frequency may be required (see Requirements for USB Communications below).

Up to 7 clock domains are generated by the CGC in the RX. Not all domains are available on every RX family member. See Table 4.1 - RX Clock Domains.

Table 4.1 - RX Clock Domains

Clock domain	Symbol	RX610	RX62N, RX621	Source
System clock	ICLK	X	X	EXTAL x [1|2|4|8]
Peripheral clock	PCLK	X	X	EXTAL x [1|2|4|8]
External bus clock	BCLK	X	X	EXTAL x [1|2|4|8]
SDRAM clock	SDCLK		X	BCLK
USB clock (48 MHz)	UCLK		X	EXTAL x 4
RTC clock	SUBCLK		X	OSC1 (32 kHz)
On-chip oscillator	OCOCLK		X	On-chip 125 kHz

4.1 Clock Frequency Requirements

The ICLK must always be greater than or equal to the PCLK and BCLK. Minimum and maximum frequencies are shown in the table below.

Table 4.2 - Frequency Range for MCU Clocks

	ICLK	PCLK	BCLK	SDCLK	UCLK
Maximum Frequency [MHz]	100	50	100*	50	48
Minimum Frequency [MHz]	8	8	8	8	48

* While BLCK can be set to 100 MHz, the maximum frequency that can be output on the BCLK pin is 50 MHz.
A 50 MHz clock can be output on the BCLK pin with a BCLK frequency of 100MHz by setting the BCLKDIV bit.

4.1.1 Requirements for USB Communications

The USB 2.0 Host/Function Module (USB) available on some members of the RX family requires a 48 MHz USB clock signal (UCLK). UCLK is generated internally by multiplying the external clock by 4. **Applications that require USB communications must use a 12 MHz clock input to the chip**; maximum system clock (ICLK) if using USB is 96 MHz. Additionally, the peripheral clock (PCLK) must be set to a minimum of 24 MHz when USB is enabled.

4.1.2 Requirements for SDRAM Controller

When the SDCLK is used, BCLK cannot exceed 50 MHz.

4.1.3 Requirements for Ethernet Controller

When the Ethernet controller (EtherC) and Ethernet DMA Controller (E-DMAC) are used, system clock (ICLK) must be a minimum of 12.5 MHz.

4.2 Reset Conditions

On reset the following multipliers of the input clock are used:

- ICLK = Frequency(EXTAL) x 2
- PCLK = Frequency(EXTAL) x 2
- BCLK = SDCLK = Frequency(EXTAL) x 2
- UCLK = Frequency(EXTAL) x 4

4.3 On-Chip Oscillator & Independent Watchdog Timer

Some members of the RX family include an Independent Watchdog Timer (IWDT) that is clocked from a separate source: the on-chip oscillator. The clock supplied by the on-chip oscillator, OCOCLK, is fixed at 125 kHz.

4.4 Sub-clock Oscillator

Members of the RX family that feature a real-time clock (RTC) require an external 32.768 kHz crystal connected to the OSC1 and OSC2 pins. An on-board sub-clock oscillator uses this crystal to generate the 32.768 kHz SUBCLK signal that drives the RTC. The RTC can be used in all low power states to wake up the MCU even when the main clock is stopped.

4.5 Oscillation Stop Detection

An on-chip Oscillation Stop Detection circuit monitors the output of the main clock oscillator (MCO) that is driven by the EXTAL and XTAL pins. It detects when the MCO is locked at a 0 or 1 state for too long indicating a stoppage of the MCO.

When a stoppage of the oscillator is detected, a number of events occur:

1. The MCU switches to the internal on-chip oscillator (OCOCLK = 125 kHz).
2. The MCU operates at a reduced speed defined by the multipliers in the SYSCKR register: the MCU’s maximum speed when running off the on-chip oscillator is 1 MHz (125 kHz OCOCLK x 8).
3. Optionally, MTU outputs can be forced to high-impedance.
4. Optionally, a non-maskable interrupt can be generated.

4.5.1 Important notes regarding the Oscillation Stop Detection Circuit

- Once the MCU has switched from the main clock oscillator (MCO) to the on-chip oscillator (OCOCLK), it continues to run on OCOCLK even if MCO is re-established; a reset of the MCU is required to switch back to the main clock oscillator.
- The Oscillation Stop Detection circuit is enabled by default after reset, but may be disabled by writing to the Oscillator Stop Detection Control Register (OSTDCR).
- Because the main clock oscillator is turned off in the Software Standby and Deep Software Standby low power modes, the Oscillation Stop Detection circuit must be disabled before entering these modes.
- To use the Non-Maskable Interrupt for oscillator stop, the OSTEN bit in Non-Maskable Interrupt Enable Register (NMIER) must be set.
- Application code servicing the Independent Watchdog Timer (IWDT) must take into account that the IWDT continues to run at the same rate even though the MCU is running at a reduced rate.

4.6 Writing the System Clock Control Register

Care should be taken when writing to the individual bit fields in the SCKCR register. The hardware manual states:

"After writing to the SCKCR, further writing to the same register before completion of the change in frequency is ignored. In the case of continued writing to the SCKCR, confirm that values read from the SCKCR are actually the most recently written values."

The easiest way to avoid this situation and to ensure clock settings are correct, is to write the entire register at once:

Unsafe	Safe
`/*ICLK=EXTAL*8=12.5M*8=100MHz*/`	`/* ICLK=0=EXTAL*8=12.5M*8=100MHz`
`SYSTEM.SCKCR.BIT.ICK = 0;`	`   PCLK=1=EXTAL*4=12.5M*4=50MHz`
`/*PCLK=EXTAL*4=12.5M*4=50MHz*/`	`   BCLK=2=EXTAL*2=12.5M*2=25MHz */`
`SYSTEM.SCKCR.BIT.PCK = 1;`	`SYSTEM.SCKCR.LONG = (unsigned long)0x00020100;`
`/*BCLK=EXTAL*2=12.5M*2=25MHz*/`	
`SYSTEM.SCKCR.BIT.BCK = 2;`	

4.7 Board Design

Refer to the "Usage Notes" section of the Clock Generation Circuit chapter in the Hardware Manual for more information on using the CGC and for board design recommendations.

5. Reset Requirements and the Reset Circuit

Up to six reset sources are available for members of the RX family. Not all members support all six sources; check the hardware manual for details pertaining to specific chips.

Table 5.1 - Reset Sources

Reset name	Source
Pin Reset	The RES# pin is driven low.
Power On Reset	VCC rises or falls (voltage detection: VPOR)
Voltage-monitoring Reset	Vcc falls (voltage detection: Vdet1 and Vdet2)
Deep software standby reset	Deep software standby mode is canceled by an interrupt.
Independent watchdog timer reset	The independent watchdog timer underflows, or a refresh error occurs.
Watchdog timer reset	The watchdog timer overflows.

5.1 Pin Reset

When the RES# pin is driven low, all processing is aborted and the RX enters a reset state. During a power on sequence, the RES# should be held low for the specified oscillation stabilization time (typically 10 ms). To reset the MPU while it is running, RES# should be held low for the specified reset pulse width (typically a minimum of 1.5 uS); RES# must be held low longer during ROM or data flash programming and erasure (as long as 35 uS). Refer to the "Electrical Characteristics" chapter of the Hardware Manual for the timing requirements for a specific RX family member. Also refer to section 12 - Emulator Support for details on reset circuitry in relation to debug support.

5.2 Power On Reset

The Power On Reset occurs when the RES# pin is high as power is applied to the MCU. After VCC has exceeded the power on voltage (Vpor) and the specified period (power-on reset time, t_{POR}) has elapsed, the chip is released from the power-on reset state. The power-on reset time is a period that allows for stabilization of the external power supply and the MCU.

If the RES# pin is high when the power supply (VCC) falls to or below Vpor, a power-on reset is generated. The chip is released from the power-on state after VCC has risen above Vpor and the t_{POR} has elapsed.

After a power on reset, the PORF bit in RSTSR is set to 1; following a pin reset PORF is cleared to 0.

5.3 Voltage-monitoring Reset

Some members of the RX family include circuitry that allows the MCU to protect against unsafe operation during brownouts. On-board comparators check the supply voltage against two reference voltages, Vdet1 and Vdet2. As the supply dips below each reference voltage an interrupt or a reset can be generated.

When Vcc subsequently rises above Vdet1 or Vdet2, release from the voltage-monitoring reset proceeds after a stabilization time has elapsed.

Low Voltage Detection is disabled by default after reset; see the chapter "Voltage Detection Circuit (LVD)" in the hardware manual for details.

5.4 Deep Software Standby Reset

This is an internal reset generated when deep software standby mode is canceled by an interrupt.

When deep software standby mode is canceled, a deep software standby reset is generated, and clock oscillation starts. A time delay specified in the deep software standby wait time setting bits (DPSWCR.WTSTS[5:0]) gives the oscillator time to start up before reset is canceled and normal processing starts. For details of the deep software standby mode refer to the "Low Power Consumption" chapter of the hardware manual.

5.5 Independent Watchdog Timer Reset

This is an internal reset generated by the Independent Watchdog Timer (IWDT).

When the IWDT underflows, an independent watchdog timer reset is generated and the UNDFF bit in the IWDTSR is set to a 1. After a short delay, the IWDT reset is canceled.

5.6 Watchdog Timer Reset

This is an internal reset generated by the Watchdog Timer (WDT).

When the WDT overflows, a watchdog timer reset is optionally generated, the WDTOVF# is driven low, and the WOVF bit in RSTCSR is set to a 1. After a short delay the WDT reset is canceled.

Do not connect the WDTOVF# pin, which is an output, to the RES# pin, which is an input. The RX MCU will not be initialized properly if WDTOVF# is connected to RES#. To reset the entire system by means of the WDTOVF# signal, use a circuit like the one shown in Figure 5.1 - Example of System Reset Circuit Using WDTOVF# Signal.

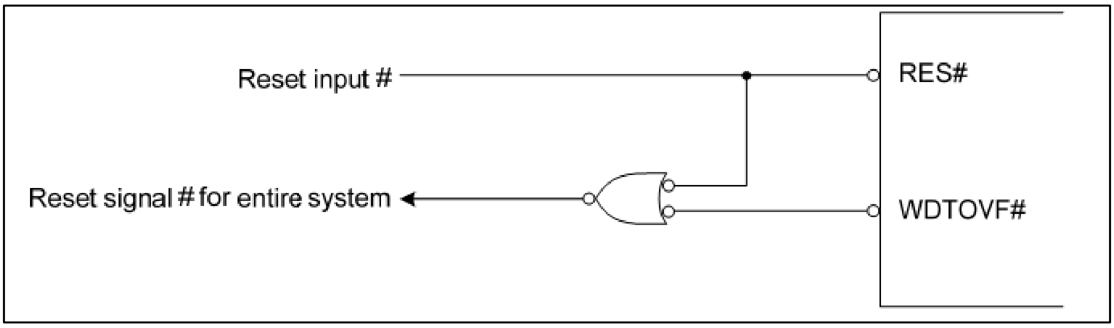


Figure 5.1 - Example of System Reset Circuit Using WDTOVF# Signal

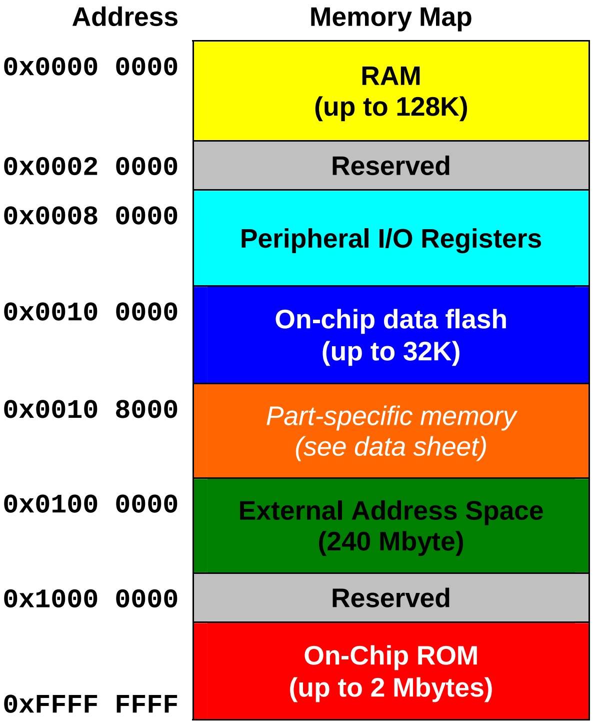
5.7 Determining the Reset Source

The following code sample shows how to determine the source that caused a reset.

```
#define RST_SRC_WDT           0x01    /* Watchdog timer reset */
#define RST_SRC_IWDT          0x02    /* Independent watchdog reset */
#define RST_SRC_DSSTDBY       0x04    /* Deep software standby reset */
#define RST_SRC_VDET2         0x08    /* Voltage monitor 2 reset */
#define RST_SRC_VDET1         0x10    /* Voltage monitor 1 reset */
#define RST_SRC_POR           0x20    /* Power on reset */
#define RST_SRC_PIN           0x40    /* Pin reset */

int ResetSource ()
{
  /* Check for watchdog timer (WDT) reset */
  if ((WDT.READ.RSTCSR.BIT.RSTE==1)&&(WDT.READ.RSTCSR.BIT.WOVF==1)) return (RST_SRC_WDT) ;

  /* Check for independent watchdog timer (IWDT) reset */
  if (IWDT.IWDTSR.BIT.UNDFF == 1) return (RST_SRC_IWDT) ;

  /* Check for deep software standby reset */
  if (SYSTEM.RSTSR.BIT.DPSRSTF == 1) return (RST_SRC_DSSTDBY) ;

  /* Check for voltage monitoring reset on Vdet2 */
  if ((SYSTEM.LVDCR.BIT.LVD2E==1) && (SYSTEM.LVDCR.BIT.LVD2RI==0) && (SYSTEM.RSTSR.BIT.LVD2F==1))
    return (RST_SRC_VDET2) ;

  /* Check for voltage monitoring reset on Vdet1 */
  if ((SYSTEM.LVDCR.BIT.LVD1E==1) && (SYSTEM.LVDCR.BIT.LVD1RI==0) && (SYSTEM.RSTSR.BIT.LVD1F==1))
    return (RST_SRC_VDET1) ;

  /* Check for power on reset */
  if (SYSTEM.RSTSR.BIT.PORF == 1) return (RST_SRC_POR) ;

  /* If no other reset sources were indicated, then it must have been a pin reset */
  return (RST_SRC_PIN) ;
}
```

6. Memory

The RX600 Series of MCU's have a 32-bit memory space spanning 4 Gbyte that includes areas for on-chip memory and peripherals. Some members of the family include a 256 Mbyte region that allows access to devices connected to external memory buses. Program and data memory share the address space; separate buses are used to access each, increasing performance and allowing same-cycle access of program and data. Contained within the memory map are regions for on-chip RAM, peripheral I/O registers, program ROM and data flash, and external memory.



6.1 On-Chip RAM

Members of the RX family include high-speed on-chip RAM that can be accessed in a single cycle at CPU speeds up to 100 MHz. Data stored in RAM is retained in all low-power modes of the CPU; the entire RAM or a portion of it may be powered down during Deep Software Standby Mode to further reduce power consumption. Depending on the RX device, up to 128K of on-chip RAM is accessed starting at address 0x00000000.

6.2 Peripheral I/O Registers

Blocks of peripheral I/O registers appear at various locations in the memory map depending on the device and the current operating mode. The majority of peripheral I/O registers occupy a region from address 0x00080000 to 0x00100000. This region contains registers that are available at all times in all modes of operation. Other blocks of peripheral registers, such as those to control access flash memory, vary in location and size by device; consult the hardware manual for specifics. The Renesas tool chain generates C header files that map all of the peripheral I/O registers for a specific device to easily accessible C data structures; see section 7 - I/O Register Structures for details.

6.3 Program ROM & Data Flash

The RX600 Series of MCUs feature two flash memory sections: program ROM and data flash. The program ROM stores user application code and constant data. The data flash stores information that may be updated from time to time such as configuration parameters, user settings, or logged data. The units of programming and erasure in the data flash area are much smaller than that of the program ROM (8 bytes for Data Flash versus 256 bytes for ROM). This makes the data flash more suited for storing information that would benefit from the finer granularity of the data flash area, such as configuration parameters.

Both the data flash and ROM areas can be programmed or erased by application code. This enables field firmware updates without having to connect an external programming tool. To speed development of code that supports in-application programming of the flash, Renesas supplies a “Simple Flash API for RX Application Note” that includes sample code. This Application Note can be found on the Renesas web site; see the section Website and Support at the end of this manual.

One some RX600 Series MCUs another flash area is available to the user. This small flash area can be used along with User Boot Mode to hold a custom bootloader that the user designs. This area cannot be erased or programmed during normal user application execution.

6.3.1 Blank Checking of Flash Memory

Reading data from blank/erased sectors of flash memory returns undefined (i.e. non-“0xFF”) values. To determine if a sector of flash memory has been erased, use the blank check command in the Flash Control Unit.

6.3.2 Background Operation

The CPU can execute application code from ROM while the data flash memory is being erased or programmed. The CPU is able to execute program code from areas other than the ROM or data flash while the ROM is being programmed or erased.

6.4 External Memory & Chip Selects

Some members of the RX family include an external data bus for connection to external memory and devices. Some members also include a built-in SDRAM controller that allows the use of up to 128 Mbytes of external SDRAM. Eight programmable chip selects provide a number of options that are settable on a per-chip select basis to allow connection to a wide range of external devices. The external chip select area of the memory map begins at address 0x0100 0000. Consult the hardware manual for details.

6.4.1 Special note about CS0

Chip select zero is mapped in memory to the same space as the internal ROM of the device. When using chip select zero, the device boots out of internal ROM. Application code must then enable CS0 and disable on-chip ROM. Once on-chip ROM is disabled it cannot be re-enabled without resetting the chip. ***Users are advised not to use chip select zero unless they have very specific reasons for doing so.***

6.4.2 Using External 16-bit Memory Devices

When connecting an external 16-bit memory device that has a byte select line, connect A1 of the MCU to A0 of the memory and A0 of the MCU to the byte select line.

6.5 Memory Access Speed

Both the RAM and internal ROM can be accessed in a single cycle with no wait states. This is true to up to the current maximum operating frequency of the RX600 Series, which is 100MHz. The system peripheral clock limits speed when accessing peripheral I/O registers and data flash memory. An example: if the clocks are set at their maximums (System clock: 100MHz, Peripheral Clock: 50MHz), it will take 2 CPU cycles to access a peripheral I/O register. The data flash memory takes 3 cycles of the peripheral clock to read 1 or 2 bytes.

6.6 Data Alignment

There are no limits for aligning data. The MCU is capable of doing byte, word, and long accesses on odd memory locations. While it is still optimal to align data accesses, it is not required.

7. I/O Register Structures

Renesas supplies a C language header file named ‘iodefine.h’ that allows users to easily access I/O registers through unions and structures. The syntax of using these unions and structures to access hardware registers is:

```
Peripheral.Register<.AccessWidth>.<Bit>
```

Where:

Peripheral is the name of a specific peripheral such as: SCI0, ICU, AD0, etc.

Register is the register abbreviation for a specific register such as: SCR, IPR, ADCR, etc.

AccessWidth is an optional field used when an I/O register has more than one field. One of four keywords specifies how to access the register: LONG, WORD, BYTE, or BIT.

Bit is an optional field that is only used when *AccessWidth* is BIT. It specifies the name of a single bit or range of bits in a register such as: TIE, IPR, or MODE.

Note that *Peripheral*, *Register*, and *Bit* match the mnemonics used in the RX Hardware Manual.

If accessing a register that does not have bit fields, use the peripheral and register name only. An example is ‘MTU0 Timer Counter’ shown in the table below.

What to access	Bits to Access	How to access
System Clock Control Register (SCKCR)	32	SYSTEM.SCKCR.LONG
MTU0 Timer Counter	16	MTU0.TCNT
SCI Channel 3, Receive Data Register (RDR)	8	SCI3.RDR
SCI Channel 3, Serial Control Register (SCR)	8	SCI3.SCR.BYTE
SCI Channel 3, Receive enable bit in SCR	1	SCI3.SCR.BIT.RE
Port 2, Pin 5, Data Direction Register Bit	1	PORT2.DDR.BIT.B5
Counter Clear bit field in TMR0 TCR register	2	TMR0.TCR.BIT.CCLR
CMT0 Compare Match Timer Control Register	16	CMT0.CMCR.WORD

7.1 I/O Register Macros

New macros in the iodefine.h for RX family parts make it easier to refer to ICU control registers, module stop registers, DTC enable registers, and interrupt vector numbers by the logical names associated with the peripherals. These macros allow portability across RX family members by hiding specific register and vector numbers. See the documentation contained in iodefine.h and sections below for details.

Some examples:

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)**))

7.1.1 ICU Register Macros

These macros help with accesses to the following registers in the ICU:

- Interrupt Request Registers (IRn)
- DTC Activation Enable Register (DTCERn)
- Interrupt Request Enable Register (IERm)
- Interrupt Priority Register (IPRm)

Instead of having to refer to the values for ‘n’ and ‘m’, the user can specify the desired peripheral and interrupt. Application code then becomes portable across members of the RX family that share the same peripheral.

Examples are below.

Without Macro	With Macro
`ICU.IR[180].BIT.IR = 0;`	`IR(TMR2, CMIA2) = 0;`
`ICU.DTCER[180].BIT.DTCE = 1;`	`DTCE(TMR2, CMIA2) = 1;`
`ICU.IER[0x16].BIT.IEN4 = 1;`	`IEN(TMR2, CMIA2) = 1;`
`ICU.IPR[0x6A].BIT.IPR = 3;`	`IPR(TMR2, CMIA2) = 3;`

7.1.2 Vector Number Macro

When using the Renesas compiler, interrupt service routines written in C language are hooked to specific interrupts vectors using the #pragma interrupt directive:

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

The above example hooks the C language function “INT_RXI0” to interrupt vector number 145, which is the receive interrupt for SCI0. This same interrupt source (RXI0) may not use the same vector number (145) on other members of the RX family. To provide portability, the **VECT()** macro allows the user to specify a logical name for an interrupt source which is then expanded by a part-specific iodefine.h file to the correct vector number.

The syntax is:

VECT(*Peripheral*, *Source*)

Where:

Peripheral is the name of a specific peripheral such as: SCI0, TMR2, AD0, etc.

Source is the name of an interrupt source in that peripheral such as: RXI0, CMIA2, ADI0, etc.

Example:

Without Macro

```
/* Declare ISR for TMR2 – CMIA2 */
#pragma interrupt TMR2_CMIA2(vect=180)
```

With Macro

```
/* Declare ISR for TMR2 – CMIA2 */
#pragma interrupt TMR2_CMIA2(vect=VECT(TMR2,CMIA2))
```

7.1.3 Module Stop Control Macro

The Module Stop Control Registers allow individual peripherals to be turned on or off for power savings. By default, most peripherals are off at power up and must be powered on before accessing their control registers (see hardware manual for details). The Module Stop Control Registers contain bit fields for a number of peripherals; these registers change in layout from part to part in the RX family. The **MSTP( )** macro simplifies control of the stop state of peripherals and makes code portable.

To use this macro, specify the name of the peripheral:

MSTP (*Peripheral*)

Example:

Without Macro	With Macro
`/* Turn on TMR2 */`	`/* Turn on TMR2 */`
`SYSTEM.MSTPCRA.BIT.MSTPA4 = 0;`	`MSTP(TMR2) = 0;`

7.2 I/O Registers and Endian Settings

The RX I/O Registers are at fixed locations and byte orders in memory regardless of the endian setting of the processor. When accessing data memory, the most significant byte of a 16-bit word can be stored at either an odd or even address depending on the endian setting; this is not the case with the RX I/O Registers.

> ***Always access I/O registers using the proper access instruction for the size of the register; do not access word or long word registers with byte instructions, or long word registers with word instructions. Do not assume that registers for a particular peripheral are big-endian or little-endian.***

This can confuse some compilers depending on the data structures used to access I/O Registers, particularly when using bit fields in 16-bit or wider registers. The iodefine.h file generated by the Renesas tools uses directives specific to the Renesas compiler (such as "__evenaccess") to ensure that access to the I/O registers is correct regardless of the endian setting of the processors.

Because of this:

The user is strongly advised to use only the structures in iodefine.h file to access I/O registers

and

to check the compiler output at the assembly language level if changes are made to the file.

8. I/O Port Configuration

The I/O Ports section of the Hardware Manual describes exact pin configurations based on peripheral selection and other register settings. Some general information is listed below.

8.1 Setting Up and Using port as GPIO

- Select a pin as an output by writing a "1" to the corresponding Data Direction Register (DDR)
- The Data Direction Register (DDR) is read/write. Setting the value to a 1 selects the pin as an output. Default state for I/O Ports is "0" (input). Unlike the H8 family, the DDR's can be read on the RX.
- When using a pin as an input the corresponding bit in the port's Input Buffer Control Register (ICR) must be set to a 1.
- The Data Register (DR) is read/write. When the Data Register is read the state of the output data latch (not the pin level) is read.
- The PORT register is read only. Read the PORT register to read the pin state. The RX610 is an exception to this rule. On the RX610 when the corresponding pin is selected as output, the DR register is read out instead of the pin state. When the pin is selected as input on the RX610, it acts the same as the other MCUs in the RX600 Series.
- When setting a pin as an output it is recommended that the desired output value of the port be written to the data latch first, then the direction register is set to an output. Though not important in all systems, this prevents an unintended output glitch on the port being setup.

Examples:

Set up Port 0, bit 1 as an input:

```
/* Make pin an input */
PORT0.DDR.BIT.B1 = 0 ;
/* Enable the input buffer */
PORT0.ICR.BIT.B1 = 1 ;
/* See if input is high */
if (PORT0.PORT.BIT.B1 == 1) …
```

Set up Port 0, bit 1 as an output:

```
/* Set the output level first to prevent glitches */
PORT0.DR.BIT.B1 = 0 ;
/* Make pin an output */
PORT0.DDR.BIT.B1 = 1 ;
```

8.1.1 Internal Pull-Ups

- Each pin on ports A through E has the option of enabling a pull-up MOS. The pull-up is controlled by the Pull-Up MOS Control Register (PCR). Each bit in the PCR register controls the corresponding pin on the port. Set the PCR bit to "1" to enable the pull-up and to "0" to disable it.
- The pull-up is automatically turned off whenever a pin is designated as an output.

8.1.2 Open-Drain Output

Pins configured as outputs normally operate as CMOS outputs. Some port pins can be configured as NMOS open-drain outputs (consult the hardware manual for your specific part to see which pins have this capability). The Open Drain Control Register (ODR) controls which pins operate in open-drain mode; setting the applicable bit in the register to a

‘1’ makes the output open drain. Because of parasitic diodes on the RX port pins, maximum voltage to open drain outputs must be limited to VCC.

8.2 Setting Up and Using Port Peripheral Functions

- Since many pins have multiple functions the RX600 Series has Port Function Control Registers (PFCR#) that allow you to change the function assigned to a pin. For example, PFCR8 on the RX610 allows you to choose whether you want the IRQ8 pin to be P0_0 or P4_0.
- To help save power the RX600 Series has Input Buffers on the I/O Ports. These buffers control whether peripherals in the MCU are connected to the I/O pins. The buffers are controlled using the Input Buffer Control Registers (ICR). **After reset the default value for the register is “0” which means there is no connection between the peripheral and the pin on the MCU.** Therefore, if using a peripheral that uses an input pin on the MCU, make sure to set the appropriate ICR to “1” to enable the connection. Setting the ICR register for an output pin of a peripheral is not required.
- The “I/O Ports >> Settings of Ports” section of the Hardware Manual details the different functions that are available on each pin, and how to enable each function.

Example - Enabling SCI6 to use port 0, bit 1 as SCI receiver input pin

```
/* Enable SCI6 (take out of stop mode) */
MSTP(SCI6) = 0 ;
/* Enable the input buffer to the peripheral */
PORT0.ICR.BIT.B1 = 1 ;
/* Make the pin an input */
PORT0.DDR.BIT.B1 = 0 ;

/* Continue with SCI peripheral initialization… */
```

8.3 Setting Up and Using IRQ Pins

Certain port pins can be used as hardware interrupt lines (see hardware manual for specific pins).

Example - Enabling port 0, bit 1 as IRQ9 input

```
/* Set port function: IRQ9 is assigned to P0.1 */
IOPORT.PF8IRQ.BIT.ITS9 = 0;
/* Make the pin an input */
PORT0.DDR.BIT.B1 = 0 ;
/* Enable the input buffer */
PORT0.ICR.BIT.B1 = 1 ;
/* Set IRQ type (falling edge) */
ICU.IRQCR[9].BIT.IRQMD = 0x01 ;
/* Set interrupt priority to 3 */
IPR(ICU,IRQ9) = 0x03 ;
/* Enable the interrupt */
IEN(ICU,IRQ9) = 1 ;

/* Be sure to write an interrupt handler!!! */
```

8.4 Unused Pins

NOTE:

Some pins require specific termination: See the "I/O Ports: Treatment of Unused Pins" section of the Hardware Manual for specific recommendations.

Unused pins that are left floating can consume extra power and leave the system more susceptible to noise problems. Terminate unused pins with one of the methods detailed here:

1. The first option is to set the pin to an input (the default state after Reset) and connect the pin to Vcc or Vss using a resistor. There is no difference from a MCU standpoint between one connection or another; however, there may be an advantage from a system noise perspective. Vss is probably the most typical choice. Avoid connecting a pin directly to Vcc or Vss since an accidental write to the port's direction register that sets the pin to an output could create a shorted output.
2. A second method is to set the pin to an output. It does not matter whether the pin level is set high or low; however, setting the pin as an output and making the output low connects the pin internally to the ground plane. This may help with overall system noise concerns. A disadvantage of setting unused pins to outputs is that the configuration of the port must be done via software control. While the MCU is held in Reset and until the direction register is set for output the pin will be a floating input and may draw extra current. If the extra current can be tolerated during this time, this method eliminates the external resistors required in the first method.
3. A variation on leaving the pins as inputs and terminating them with external resistors uses the internal pull-ups available on some ports of the MCU. This has the same limitation as setting the pins to outputs (requires the program to set up the port) but it does limit the effect of accidental pin shorts to ground, adjacent pins or Vcc since the device will not be driving the pin.

8.5 Electrical Characteristics

- GPIO require CMOS level inputs (High ≥0.8 * Vcc, Low≤ 0.2*Vcc) see electrical characteristics for more information

9. Module Stop Function

To maximize power efficiency, the RX family MCU's allow on-chip peripherals to be shut down individually by writing to the Module Stop Control Registers (MSTPCRi, i = A,B,C). After reset most of the modules are stopped (exceptions are DMAC, DTC, and on-chip RAM, see hardware manual for details).

Before accessing any of the registers for a peripheral, it must be enabled by taking it out of stop mode by writing a '0' to the corresponding bit in the MSTPCRi register.

Peripherals may be shut down by writing a '1' to the proper bit in the MSTPCRi register.

The **MSTP ()** macro in iodefine.h makes it easy to enable and disable peripherals using their name. See section 7.1.3 - Module Stop Control Macro for details. Details on the MSTPCRi registers can be found in the "Low Power Consumption" chapter of the hardware manual.

Example - Enabling SCI6 to use port 0, bit 1 as RXD input

```
/* Enable SCI6 (take out of stop mode) */
MSTP(SCI6) = 0 ;

/* Enable the input buffer to the peripheral */
PORT0.ICR.BIT.B1 = 1 ;

/* Make the pin an input */
PORT0.DDR.BIT.B1 = 0 ;

/* Continue with SCI peripheral initialization… */
```

10. Interrupts

10.1 Interrupt Control Unit Basics

The RX family has a sophisticated Interrupt Control Unit (ICU) that handles asynchronous events from over 100 sources. These sources include on-board peripherals, external hardware, and software requests. The Interrupt Control Unit chapter of the Hardware Manual lists each source for specific parts.

Local interrupt enable flags in each peripheral gate a signal from the peripheral to the ICU. These signals set Interrupt Status Flags in individual ICU Interrupt Request registers (IRx) that exist for each interrupt source. Within the ICU, individual bits in the Interrupt Request Enable Registers (IERx) determine whether an interrupt is taken when the Status Flag becomes set.

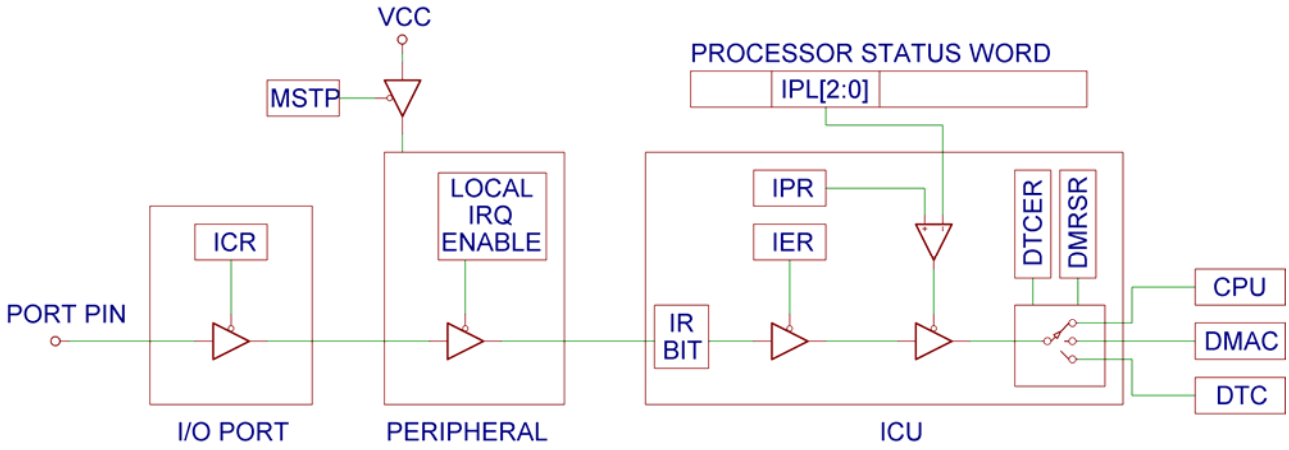


Figure 10.1 - RX Interrupt Flow

To handle simultaneous interrupt requests from multiple sources, the ICU also allows each interrupt source to be assigned a priority. These priorities are compared to the current priority level in the CPU status register IPL bits, and an interrupt is only serviced if it's priority is greater than the CPU's current IPL and all other active requests. Two active sources with the same priority level are serviced in vector number order, lowest vector first.

The steps to enable an interrupt are:

1. The peripheral or port pin generating the interrupt must be enabled and configured.
2. Set an interrupt priority for the interrupt source (IPR macro) to a value greater than zero (zero = disabled).
3. Enable the interrupt in the peripheral (local enable bit)
4. Enable the interrupt in the ICU (IEN macro)

For edge-triggered interrupts, the Interrupt Status Flags in the IR registers are cleared automatically when an interrupt fires and the CPU vectors to the Interrupt Service Routine (ISR). The flags must be manually cleared when using polled operation rather than interrupts.

For level-sensitive interrupts, the Interrupt Status Flag in the IR register stays set until the interrupt source is cleared.

10.2 Nesting Interrupts

The global interrupt enable bit in the Processor Status Word (PSW), the I bit, is cleared whenever an interrupt is taken, disabling all further interrupts including higher priority interrupts. To allow nesting of interrupts and pre-emption of the ISR by higher priority interrupts, the I bit must be set in the ISR. When declaring an interrupt in C (#pragma interrupt), use the 'enable' keyword to automatically set the 'I' bit when the interrupt is taken. Refer to RX compiler manual for more info.

10.3 Interrupt Vector Tables

The RX family has a fixed interrupt vector table and a relocatable interrupt vector table. Each vector in the vector table consists of four bytes and specifies the address where the corresponding exception handler starts.

10.3.1 Fixed Vector Table

The fixed vector table is allocated to a fixed address range. The individual vectors for the privileged instruction exception, undefined instruction exception, floating-point exception, non-maskable interrupt, and reset are allocated to addresses in the range from FFFFFF80h to FFFFFFFFh. Figure 10.2 shows the fixed vector table.

	MSB LSB
FFFF FF80	(Reserved)
...	...
FFFF FF98	(Reserved)
FFFF FF9C	ROM Code Protection
FFFF FFA0	ID Code Protection [0-3]
FFFF FFA4	ID Code Protection [4-7]
FFFF FFA8	ID Code Protection [8-11]
FFFF FFAC	ID Code Protection [12-15]
FFFF FFB0	(Reserved)
...	...
FFFF FFC8	(Reserved)
FFFF FFD0	Privileged Instruction Exception
FFFF FFD4	(Reserved)
FFFF FFD8	(Reserved)
FFFF FFDC	Undefined Instruction Exception
FFFF FFE0	(Reserved)
FFFF FFE4	Floating-Point Exception
FFFF FFE8	(Reserved)
FFFF FFEC	(Reserved)
FFFF FFC0	(Reserved)
FFFF FFF4	(Reserved)
FFFF FFF8	Non-maskable Interrupt
FFFF FFFC	Reset

Figure 10.2 - Fixed Vector Table

Do not store data in areas marked “Reserved” in the fixed vector table; some of these areas are used by the RX for specific functions such as the code protection mechanism. User data must be stored below address 0xFFFF FF80.

10.3.2 Relocatable Vector Table

The address where the relocatable vector table is placed can be adjusted. The table is a 1,024-byte region that contains all vectors for unconditional traps and interrupts and starts at the address (IntBase) specified in the interrupt table register (INTB). Figure 10.3 shows the relocatable vector table.

Each vector in the relocatable vector table has a vector number from 0 to 255. Each of the INT instructions, which act as the sources of unconditional traps, is allocated to the vector that has the same number as that of the instruction itself (from 0 to 255). The BRK instruction is allocated to the vector with number 0. Furthermore, vector numbers within the set from 0 to 255 are also allocated to other interrupt sources, such as on-chip peripherals, on a per-product basis.

Note that the value of the Interrupt Table Register (INTB) is undefined after reset. The Renesas tool chain can automatically generate startup code that initializes the INTB register. INTB can only be changed when the MCU is in supervisor mode.

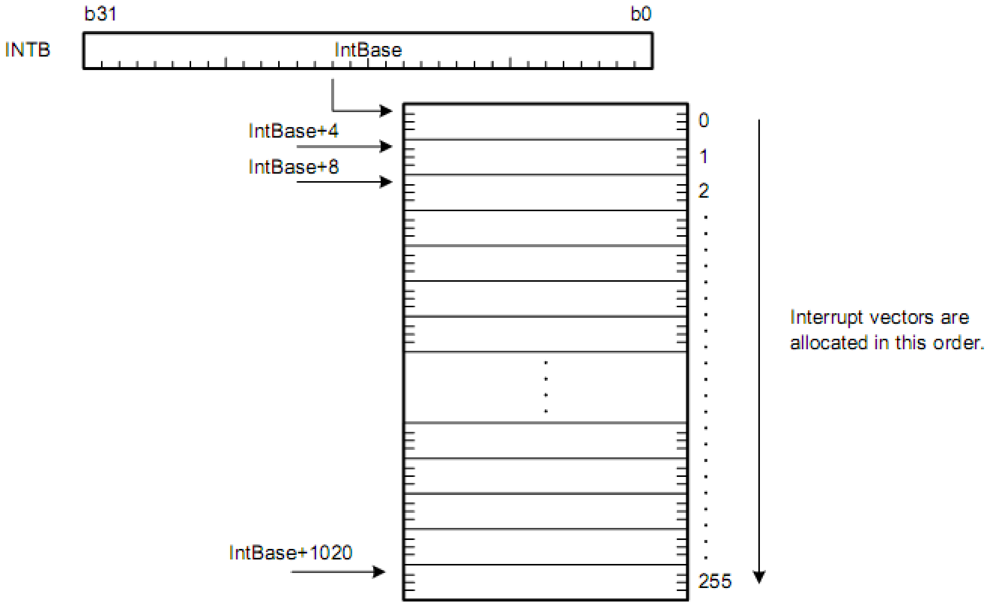


Figure 10.3 - Relocatable Vector Table

10.4 Fast Interrupts

For applications where interrupt response is critical, interrupt latency can be reduced through the use of the Fast Interrupt. The Fast Interrupt specifies one interrupt source in the Fast Interrupt Vector register (FINTV) as a high-priority interrupt, and uses dedicated registers for saving the Program Status Word (BPSW) and Program Counter (BPC). Further speed enhancements can be realized by instructing the compiler to reserve some of the general purpose CPU registers for exclusive use by the Fast Interrupt service routine. With a dedicated set of CPU registers reserved for its sole use, the response of Fast Interrupt service routine is improved by eliminating the need to save and restore processor context on the stack during entry and exit. The performance of the main application code may be slightly degraded due to the smaller register set available to it.

10.5 Interrupt Stack Pointer

A separate Interrupt Stack Pointer (ISP) is used during exception processing. This greatly reduces RAM requirements when using an RTOS since room for an interrupt stack does not need to be allocated as part of each task's stack. The ISP is automatically set by the startup code generated by the Renesas tool chain (see “Startup Program Creation” in the RX Software manual for details). Register R0 is used as the stack pointer and contains the current value of the active stack pointer (ISP or USP) depending on the processor mode.

11. Low Power Consumption

RX family MCUs include many features that allow the designer to minimize power consumption. These include a sophisticated clock generation circuit, the module stop function that allows for granular shutdown of individual peripherals, switchable external SDCLK and BCLK signals, and low power processor modes.

After reset, the MCU enters normal program execution state (full power), although all peripheral modules except the DTC, DMAC, and EXDMAC are in the stop state. See section 9 - Module Stop Function for details on starting individual peripherals.

MCU Low Power Modes

The MCU has five operating modes: normal program execution, sleep mode, all-module clock stop mode, software standby mode, and deep software mode. The state of various portions of the MCU in each mode is show in the table below. Refer to the Electrical Characteristics section of the Hardware Manual for details on current consumption in each mode.

State of operation	**Sleep Mode**	**All-Module Clock Stop Mode**	**Software Standby Mode**	**Deep Software Standby Mode**
Transition condition	Control register + instruction	Control register + instruction	Control register + instruction	Control register + instruction
Canceling method other than resets	Interrupt	Interrupt [*1]	Interrupt [*2]	Interrupt [*3]
State after cancellation [*4]	Program execution state (interrupt processing)	Program execution state (interrupt processing)	Program execution state (interrupt processing)	Program execution state (reset processing)
Oscillator	Operating	Operating	Stopped	Stopped
CPU	Stopped (retained)	Stopped (retained)	Stopped (retained)	Stopped (undefined)
On-chip RAM 1 (0001 0000h to 0001 7FFFh)	Operating (retained)	Operating (retained)	Stopped (retained)	Stopped (undefined)
On-chip RAM 2 (0000 0000h to 0000 FFFFh)	Operating (retained)	Stopped (retained)	Stopped (retained)	Stopped (retained / undefined) [*5]
USB 2.0 host / function module (USB)	Operating	Stopped [*6]	Stopped [*6]	Stopped (retained / undefined) [*7]
Watchdog timer (WDT)	Operating	Operating	Stopped (retained)	Stopped (undefined)
8-bit timer (unit 0, unit 1)	Operating	Operating [*8]	Stopped (retained)	Stopped (undefined)
Realtime clock (RTC)	Operating	Operating	Operating	Operating
Voltage detection circuit	Operating	Operating	Operating	Operating
Power-on reset circuit	Operating	Operating	Operating	Operating
Peripheral modules	Operating	Stopped [*9]	Stopped [*9]	Stopped (undefined)
I/O pin state	Operating	Retained [*11]	Retained [*10]	Retained [*10]

Notes: "Stopped (retained)" means that the internal register values are retained and internal operations are suspended.

"Stopped (undefined)" means that internal register values are undefined and power is not supplied to the internal circuit.

1. An external interrupt or some internal interrupts (8-bit timer, WDT, RTC, oscillation stop detection, USB suspend/resume, and voltage monitoring).
2. An external interrupt or some internal interrupts (voltage monitoring, RTC, and USB suspend/resume).
3. NMI, only side A of IRQ0-A to IRQ3-A, or some internal interrupts (voltage monitor circuit, RTC, and USB suspend/resume).
4. Cancellation by the RES# pin, power-on reset, voltage monitoring reset, watchdog timer reset, or independent watchdog timer reset, the MCU enters the reset state.
5. "Retained" or "undefined" can be selected by setting the on-chip RAM Off 2, on-chip RAM off 1, and on-chip RAM Off 0 bits (RAMCUT2/RAMCUT1/RAMCUT0) in DPSBYCR. These bits apply on to parts with RAM1 and RAM2 areas (like the RX62N/621).
6. Resume detecting operation is valid.
7. The USB resume detection function is enabled or disabled by setting the on-chip RAM Off 2, on-chip RAM Off 1, and on-chip RAM Off 0 bits (RAMCUT2/RAMCUT1/RAMCUT0) in DPSBYCR.
8. "Operating" or "Stopped" can be selected by setting the 8-bit timer 3/2 (unit 1) module stop and 8-bit timer 1/0 (unit 0) module stop bits (MSTPA5/MSTPA4) in MSTPCRA.
9. Peripheral modules retain the state.
10. "Retained" or "High impedance" for the address bus and bus control signals (CS0# to CS7#, RD#, WR#, WR0# to WR3#, and BC0# to BC3#) can be selected by the setting of the output port enable bit (OPE) in SBYCR.
11. When pin P53 is being used as the output pin for the BCLK signal, operation as the BCLK output is maintained. When pin P70 is being used as the output pin for the SDCLK signal, operation as the SDCLK output is maintained. For details, see 9.6, BCLK and SDCLK Output Control.

12. Emulator Support

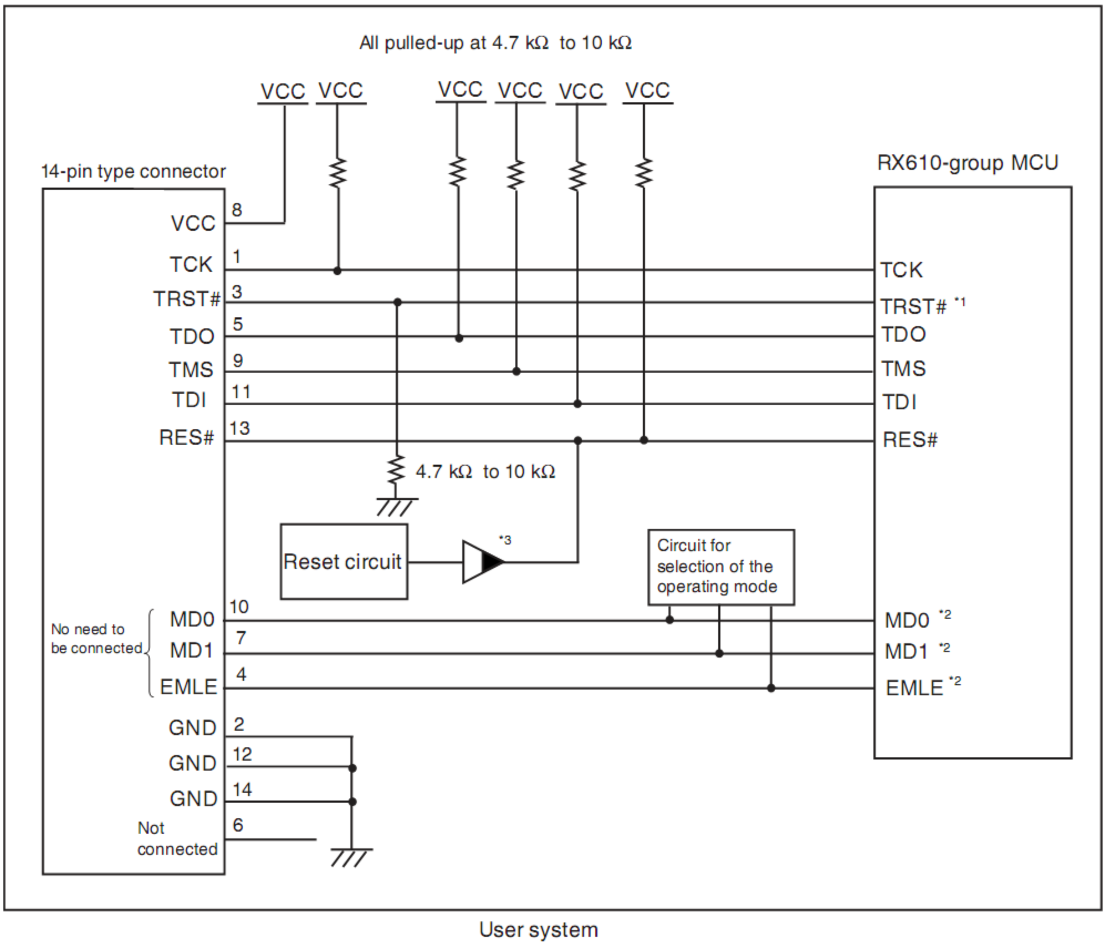
Two debug interfaces are available for members of the RX family:

- **14-pin E1 interface** that supports only basic functions using JTAG communications. It does not provide external trace-output function. These connectors are general-purpose connectors with a pitch of 2.54 mm.
- **38-pin type** that supports basic functions that employ JTAG and other communications, plus the external trace-output function for acquiring large amounts of trace data in real time. The fine-pitch connector is as compact as the 14-pin connectors.

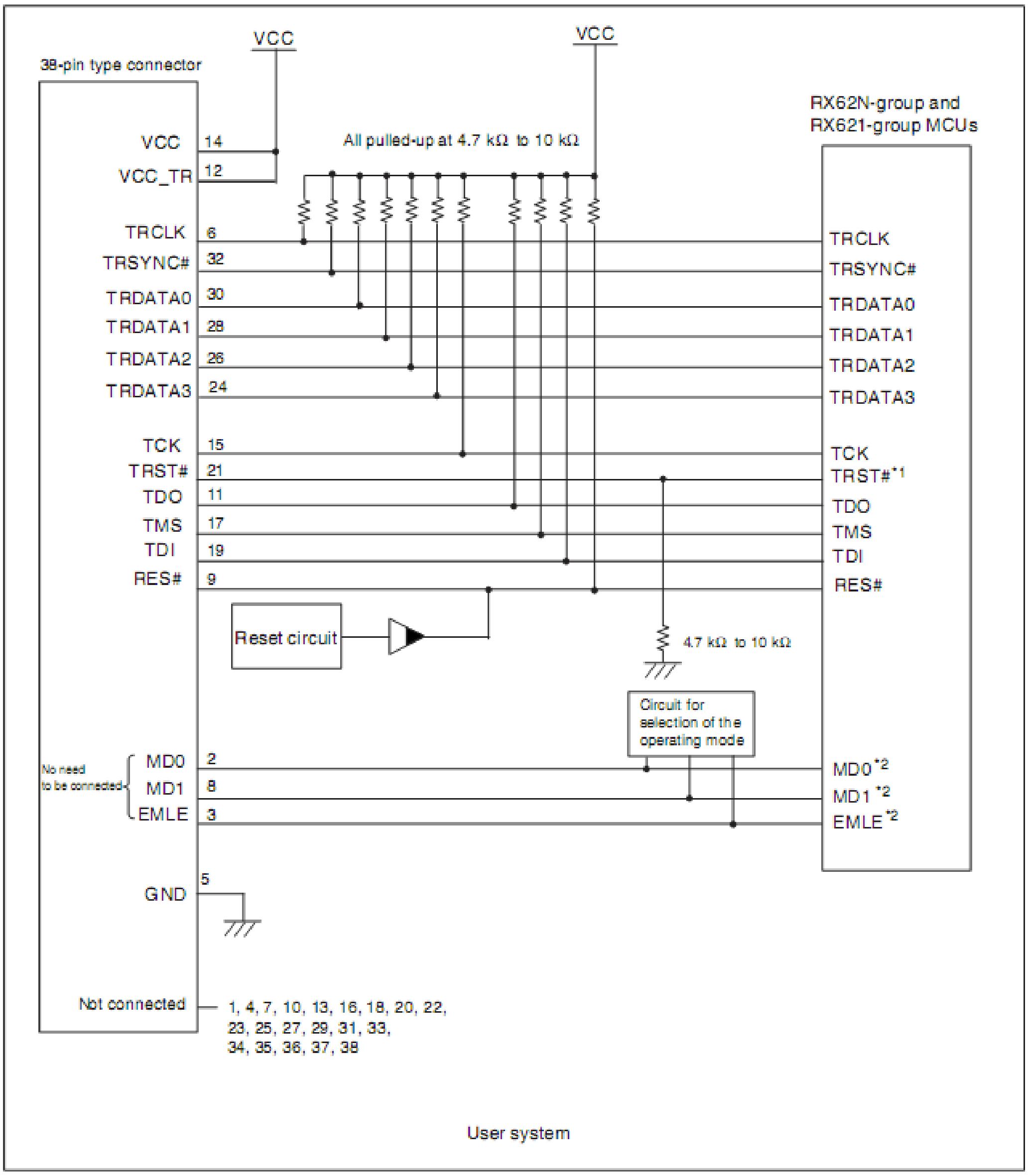
Renesas currently offers two emulators for the RX: the E1, which supports only the 14-pin connection, and the E20 that supports both the 14-pin connection and the 38-pin connection with full trace capability. Refer to "RX Family E1/E20 Emulator Additional Document" for more information (see section 13 - References).

The debug signals are multiplexed with other signals on the MCU; signals used by the debugger are generally not available for use by the application. These assignments vary with the MCU package; refer to the hardware manual.

12.1 E1 Emulator 14-Pin Interface



12.2 E20 Emulator 38-Pin Interface



12.3 Notes on Emulator Connections

RES# circuitry on the target must be open-collector.

Use 4.7K to 10K pull-ups on TCK, TDO, TMS, and TDI. Use a pull-down on TRST#.

Use pull-ups on trace connections: TRCLK, TRSYNC#, TRDATA0-3.

Connect MD0, MD1, and EMLE to the debug connector to use flash programming. MD0 and MD1 should be pulled to levels appropriate to the application; the emulator will override these when connected. Pull the EMLE pin low with a pull-down resistor; Renesas emulators will pull this signal high during debugging.

13. References

13.1 Hardware Manuals

The hardware manuals contain detailed descriptions of all hardware including memory maps, I/O register addresses, peripherals, and pin outs.

- REJ09B0460 – RX610 Group Hardware Manual
- REJ09B0552 – RX62N Group, RX621 Group Hardware Manual

13.2 Software Manual

This software manual contains details on the instruction set, MCU modes of operation (user & supervisor), CPU registers, and other information.

- REJ09B0435 – RX Family Software Manual

13.3 Emulator Manuals

These manuals discuss in-system debugging using the E1 and E20 emulators. The latter two documents include hardware schematics that show how to design a target system that supports in-circuit debug.

- REJ10J2089 - RX Family E1/E20 Emulator User's Manual
- REJ10J2090 - RX Family E1/E20 Emulator, Additional Document for User's Manual, Supplementary Information on Using the RX610 Group

Website and Support

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

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

Revision Record

Rev.	Date	Description Page	Summary
1.00	Apr.21.2010	All	Added Renesas document number & version number
1.10	July.15.2010	All	Updated document numbers, corrected RX621 references
1.20	Sept 27, 2010	14	Added notes on CS0 in section 6.4.1
1.30	March 14, 2011	2	Updated notes on USB power supply & power supply drawing
		14	Added section 6.4.2 on 16-bit external memory
		28	Added note on EMLE pin pull-down
		All	Updated format
1.40	Dec. 30, 2011	18	Updated section 8.1 with note to set ICR when using GPIO as an input. Updated example code to show ICR setting.

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 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 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 a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU 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. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of 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. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may 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.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be 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, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

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

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

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

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

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, 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
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, 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., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.1