RX111 Group

Quick Design Guide

R01AN1725EU0100
Rev.1.00
Dec 2, 2013

Introduction

This document answers common questions and points 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.

Target Device

RX111 Group

Contents

1. Power Supplies

The RX family has digital power supplies and analog power supplies. The power supplies use the following pins:

Digital Power Supplies

Symbol	Name	Description
VCC	Power supply	1.8V to 3.3V power supply. Connect to the system power supply. Connect this pin to VSS via a 0.1 uF capacitor placed close to the VCC pin.
VSS	Ground	Ground
VCL	Power supply	Connect this pin to VSS via a 4.7uF capacitor close to the VCL pin.
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.
VCC_VSS	USB ground	USB ground pin. Connect this pin to VSS.

Analog Power Supplies

Symbol	Name	Description
AVCC0	12-bit ADC power supply	Analog voltage supply pin for the 12-bit A/D converter. Connect this pin to VCC if the 12-bit ADC is not used.
AVSS0	12-bit ADC ground	Analog ground for the 12-bit A/D converter. Connect this pin to VSS if the 12-bit ADC is not used.
VREFH0	12-bit ADC high reference voltage	Reference power supply pin for the 12-bit A/D converter. Connect this pin to VCC if the 12-bit ADC is not used.
VREFL0	12-bit ADC low reference voltage	Analog reference ground pin for the 12-bit A/D converter. Connect this pin to VSS if the 12-bit ADC is not used.

1.1 References

Further information regarding the power supply for the RX can be found in the following chapters of the Hardware Manual:

Table 1 R01UH0365EJ0100 - RX111 Group User's Manual: Hardware

Chapter	Name	Description
1	Overview	Lists power pins in each package with notes on termination and bypassing.
6	Resets	Discusses the Power-on Reset and how to differentiate this from other reset sources.
8	Voltage Detection Circuit	Provides details on the Low-Voltage Detection Circuit that can be used to monitor the power supply.
11	Low Power Consumption	Using low power modes may allow you to reduce the voltage of the power supply. See this chapter for details on how operating modes affect power supply requirements.
30 31	12-Bit A/D Converter D/A Converter	If you plan to use the on-chip A/D or D/A converters, these chapters give details on how to provide filtered power supplies for these peripherals.

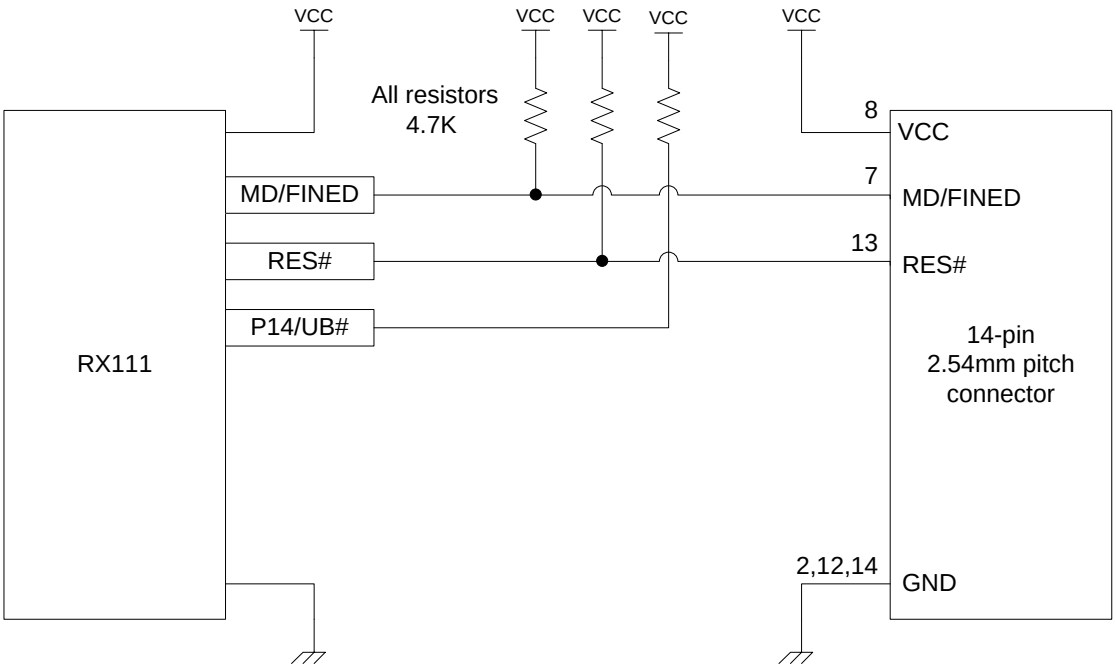
2. Emulator Support

The RX111 supports a 1-wire FINE debug interface. FINE supports full on-chip debugging, including branch trace, but does not support real-time RAM monitoring or hot plug-in. Renesas emulators support a FINE connection through the standard 14-pin E1 interface.

2.1 Debug interface

To use an E1 debugger, the MD/FINED and RES# pins from the MCU must be connected to the E1 connector as shown in the diagram below. Note that P14/UB# may be connected to a pull-up as shown or left open (do not pull down).

Figure 1 – Connections for FINE debug and FINE programming via E1 connector



2.2 Notes on Emulator Connections

Pin P14/UB# on the MCU must be pulled high during debugging. If your design uses the special USB boot mode then you will need to add additional circuitry to switch P14/UB# to ground at reset during USB boot. See for section 3, “MCU Operating Modes,” for more details.

2.3 Production Programming Requirements

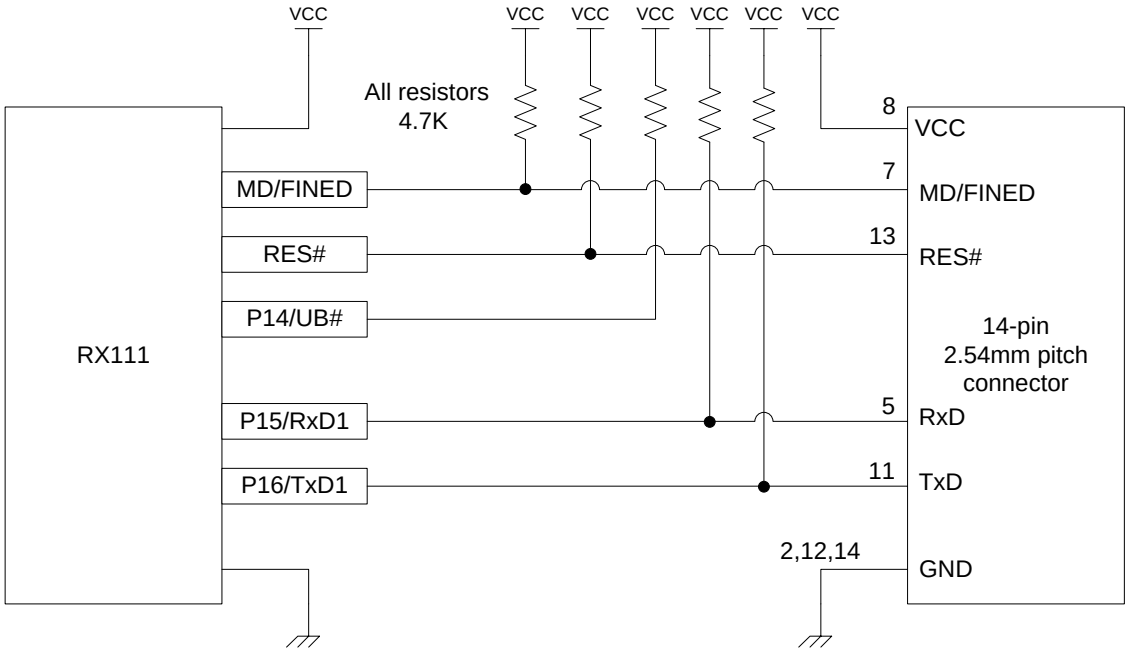
The RX111 supports three interfaces for production programming: FINE, serial boot mode, and USB boot mode. FINE programming requires an E1 emulator and the circuit shown above in section 2.1.

Serial programming requires connections to the P15/RxD1 and P16/TxD1 pins, either through the E1 connector as shown below or through an application-specific connector. During reset the MD/FINED and P14/UB# pins must be pulled to appropriate levels as detailed in section 3, “MCU Operating Modes”. The E1 emulator and the PG-FP5 stand-alone programmer both support serial boot mode programming.

Flash Memory Access Disable Function (ID Code)

The program memory can be protected with an ID code to prevent unauthorized reading of program memory. See section 7.3.4 - ID Code Protection for more details.

Figure 2 - Connections for FINE debug and serial programming via E1 interface



3. MCU Operating Modes

The RX111 MCUs can enter one of three modes after reset: single-chip mode, serial boot mode, or USB boot mode. The boot mode is selected by the MD pin and, optionally, the state of the P14/UB# pin.

Table 2 – Operating Modes Available at Reset

Mode Setting Pin		
MD[*1]	**P14/UB#**	**Mode**
High	Don't care	Single-chip mode
Low	High or open	Boot mode (SCI mode)
Low	Low	Boot mode (USB interface mode)

Note 1: Do not change the level on the MD pin while the MCU is operating.

3.1 USB Boot Mode

Pin P35/UPSEL selects whether the device is self-powered or bus-powered in USB boot mode.

Table 3 - Selecting Self-Powered or Bus-Powered USB Boot Mode

Mode Setting Pin			
MD	**P14/UB#**	**P35/UPSEL**	**Mode**
Low	Low	Low	USB Boot mode in Self-Powered mode
Low	Low	High	USB Boot mode in Bus-Powered mode

4. Option Setting Memory

The RX100 Series MCUs have Option Setting Memory. These are flash-based registers that are set when the device is programmed, and that govern the operation of the chip immediately after reset. The registers are detailed in Chapter 7 of the Hardware Manual: “Option Setting Memory”.

The flash option registers occupy space in a reserved area of the memory map; they are located as part of the fixed vector table. The image below shows the Option Setting Memory which consists of the two Option Function Select registers (OFSx) and the Endian Select Register (MDE). These registers are read by the MCU at reset to determine whether the part will run in big-endian or little-endian mode, and if peripherals like the Independent Watchdog Timer (IWDT), the High-speed On-chip Oscillator (HOCO), and low-voltage detection circuit are operational or not at boot time.

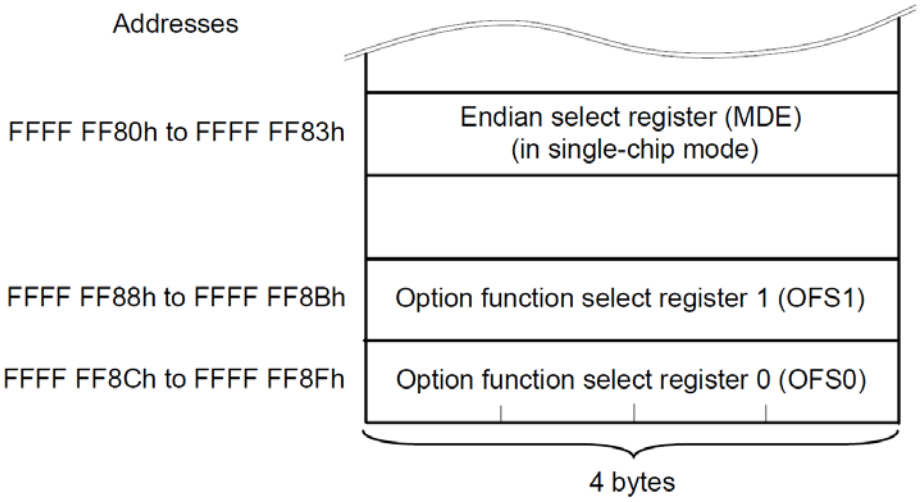


Figure 3 - Option Function Select registers

Enabling HOCO via these registers means that the HOCO is powered up and will start stabilizing immediately after reset. This reduces the wait time when switching from the LOCO (the default clock source on startup) to the HOCO. If, however, power savings is a requirement, then the registers can be configured to leave the HOCO off on power-up. The OFS registers are also used to configure all aspects of the IWDT operation.

4.1 Option Setting Memory Registers

Below is a summary of the Option Setting Memory registers. Make sure that they are configured properly before startup.

- OFS0 register
  - Independent Watchdog Timer (IWDT) auto start
  - IWDT timeout, frequency, windowing, interrupt type, and low power mode behavior
  - Watchdog Timer (WDT) auto start
  - WDT timeout, frequency, windowing, and interrupt type
- OFS1 register
  - LVD0 enable after reset
  - HOCO startup after reset
- MDE register
  - Big/little endian mode

Below is an example on how to set the option setting memory register, MDE, which is at the address 0xFFFFFF80. Note that unused/reserved bits are written to a ‘1’ as instructed in the Hardware Manual; be sure to set unused/reserved bits per the hardware manual.

```
/* Allocate the name MDEreg to the address 0xFFFFFF80 */
#pragma address MDEreg = 0xFFFFFF80

#ifdef __BIG
/* Set as Big Endian */
const unsigned long MDEreg = 0xFFFFFFF8u
#else
/* Set as Little Endian */
const unsigned long MDEreg = 0xFFFFFFFFu
#endif
```

Most sample projects and demo code from Renesas includes code to set the option registers.

5. Clock Circuits

The RX111 group MCUs have five oscillators (see Table 4 - RX111 Oscillators). Four of these may be used as source for the main system clock; the remaining oscillator is dedicated to the Independent Watchdog Timer. In a typical system the main clock is driven with an external crystal or clock. This input is directed to a divider to bring it into the 4-8 MHz input range required by the PLL, then multiplied by the PLL up to a maximum speed of 48 MHz.

Table 4 - RX111 Oscillators

Oscillator	Input source	Frequency	Primary uses
Main clock	External crystal/resonator -or- External clock	1 MHz to 20 MHz (VCC >= 2.4 V) 1 MHz to 8 MHz (VCC < 2.4V) Up to 20 MHz	PLL input, main system clock, peripherals clocks, flash clock, USB clock
Sub-clock	External crystal/resonator	32.768 kHz	Real-time clock, system clock in low power modes
High-speed on-chip (HOCO)	On-chip oscillator	32 MHz	Main system clock, peripheral clocks in low power modes
Low-speed on-chip (LOCO)	On-chip oscillator	4 MHz	System clock at startup, in low power modes, & during main oscillator stop detection
Independent Watchdog (IWDT)	On-chip oscillator	15 kHz	Independent watchdog timer clock

5.1 Reset Conditions

At reset the RX111 MCU uses the low-speed on-chip oscillator (LOCO) at 4 MHz as the main system clock. The LOCO provides a good balance between performance and low-power. At reset, the main oscillator and the PLL are off by default. The HOCO and IWDT may be on or off depending on the settings in the Option Setting Memory (see section 4).

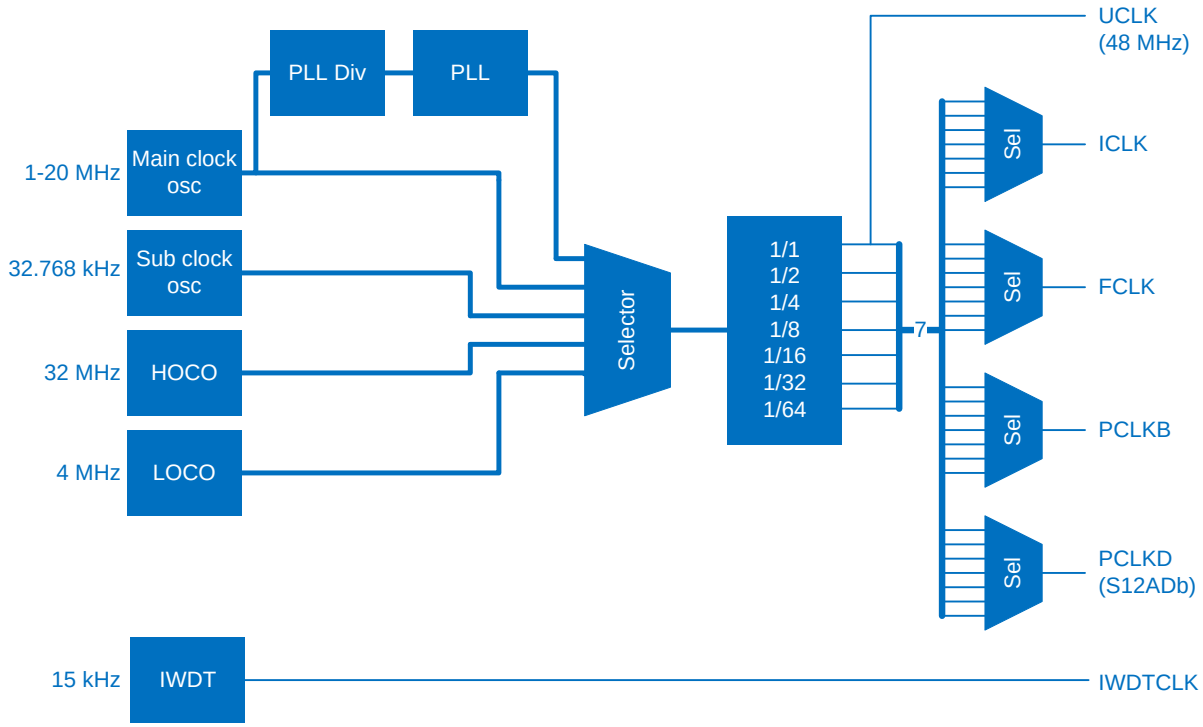
5.2 Main Clock Oscillator Voltage and Speed Dependency

The Main Clock Oscillator Forced Oscillation Control Register (MOFCR) is used to configure the main oscillator circuit for variances in MCU operating voltage, oscillator speed, and oscillator type. The MODRV21 bit selects the drive capability of the main clock oscillator, and the MOSEL bit selects whether an external resonator (crystal) or external oscillator is used. Refer to chapter 9 of the Hardware Manual for more details on the MOFCR register.

5.3 Clock Domains

The diagram below shows a simplified view of the RX111 clock domains. On the left are the five oscillators (main clock osc, sub-clock osc, HOCO, LOCO, and IWDT). The main clock can be routed into a PLL circuit to boost its speed. A selector chooses between five sources as the main system clock (PLL output, main clock, sub-clock, HOCO, and LOCO), and the output of this selector is routed through a divider and then on to individual on-chip clock domains.

Figure 4 - RX111 Clock Chain



The internal clock domains are as follows:

Table 5 - RX111 Internal Clock Domains

Clock domain	Description	Frequency
ICLK	Main system (instruction) clock Provide clock for CPU, DTC, ROM, and RAM	Up to 32 MHz
FCLK	Flash Clock. Paces reads/writes of data flash and writes to program flash when self-programming.	32 MHz max for reading data flash, >= 1 MHz for writing, <= ICLK (see section 5.4.2)
PCLKB	Peripheral Clock B. Clocks all on-chip peripherals except ADC, Data flash, and USB	32 MHz max, <= ICLK
PCLKD	Peripheral Clock D. Clocks the S12ADb	32 MHz max, <= ICLK
UCLK	USB clock (for devices with USB)	Must be 48 MHz.
IWDTCLK	Independent watchdog clock.	Fixed at 15 kHz

5.4 Clock Frequency Requirements

The ICLK must always be greater than or equal to the peripheral clocks (PCLKB, PCLKD) and the flash clock (FCLK).

5.4.1 USB

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 the PLL, which must be set to generate a 48 MHz output. This

means that all other clock domains (ICLK, PCLKB, PCLKD, FCLK) must use at least a divide by 2 divider since their maximum speed is 32 MHz. Therefore, when using USB the maximum ICLK speed is 24 MHz (48 MHz output from the PLL divided by 2).

5.4.2 Programming and Erasing ROM or Data Flash

The FCLK must be at least 1 MHz to perform programming and erasing on internal ROM and data flash. When using FCLK at lower than 4 MHz, during programming or erasing the flash memory, the frequency must be set to 1, 2, or 3 MHz.

5.5 Lowering CGC Power Consumption

The CGC area can account for 30%-40% of the power consumption of the chip. To aid in saving power, set the dividers for any unused clocks (i.e. FCLK while not using data flash or performing self-programming) to the highest possible value. Ensure that unused oscillators are powered off if not in use (i.e. if your application runs off the HOCO and the main clock and PLL are not used, then power the main clock and PLL off). The registers for controlling each clock source are shown in the table below:

Oscillator	Register	Description
Main clock	MOSCCR	Starts/stops main clock oscillator
Sub-clock	SOSCCR	Starts/stops sub-clock oscillator
High-speed on-chip (HOCO)	HOCOCR	Starts/stops HOCO
Low-speed on-chip (LOCO)	LOCOCR	Starts/stops LOCO
Independent Watchdog (IWDT)	ILOCOCR	Starts/stops IWDT on-chip oscillator

5.6 Sample Code for Clock Setup

See the sample code for the RSK RX111 board for an example of how to setup the clock.

5.7 HOCO Accuracy

The internal high-speed on-chip oscillator (HOCO) runs at 32 MHz +/-1% over the -20 to 85° C range. Accuracy decrease slightly between -20° and -40° C. Refer to the Electrical Specifications in the hardware manual for details.

5.8 FlashIF Clock

The FlashIF Clock (FCLK) is used as the operating clock for when programming and erasing internal flash (ROM and data flash) and for reading from the data flash. Therefore, the frequency setting of the FCLK will have a direct impact on the amount of time it takes to read from the data flash. If the user's program is reading from the data flash, or performing programming or erasures on internal flash, then using the maximum FCLK frequency is recommended.

Please note that the FCLK frequency does not have any impact upon reading from ROM (i.e. executing instructions) or reading and writing to RAM. Both of these memory areas are always single-cycle access.

5.9 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. A separate application note, "Design Guide for Low CL Sub-Clock Circuits" (R01AN1012EJ0100), provides details on board layout for clock circuits.

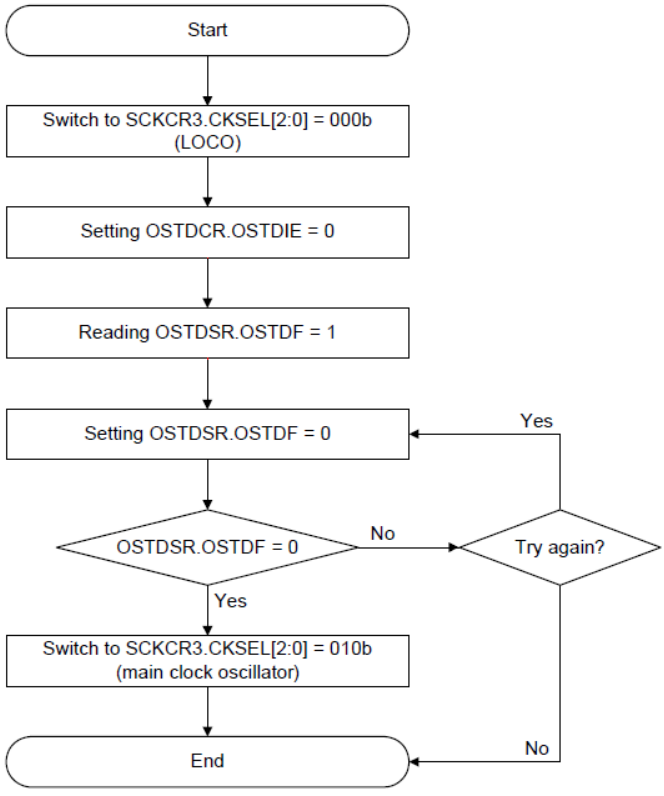
5.10 Oscillation Stop Detect

The Oscillation Stop Detect function detects when the main oscillator stops and then automatically switches the MCU's system clock over to the LOCO in response.

5.10.1 Important notes regarding Oscillation Stop Detect

- The Oscillation Stop Detect circuit is disabled by default after reset, but may be enabled by writing to the OSTDE bit in the Oscillator Stop Detection Control Register (OSCTDR)
- When the main oscillator is selected as the system clock, and oscillation stop detect is enabled, if an oscillation stop occurs the system clock source is switched to the LOCO. The SCKCR3.CKSEL bits remain unchanged, however. The system will stay on the LOCO until the OSTDF flag in the OSTDSR register is cleared. Clear this flag by changing the SCKCR3.CKSEL bits to the LOCO, then clear the flag, then switch the clock source back to the main oscillator. The steps are shown in the flowchart from the hardware manual below.
- When the PLL is selected as the system clock, and oscillation stop detect is enabled, if an oscillation stop occurs the system clock source remains the PLL, the SCKCR3.CKSEL bits remain unchanged, and the system frequency becomes the PLL free-running oscillation frequency.
- When transitioning to the LOCO, the frequency divisors in the SCKCR register for ICLK, PCLKB, PCLKD, and FCLK may need to be adjusted.
- Because the main clock oscillator is turned off in Software Standby Mode, the Oscillation Stop Detect circuit must be disabled before entering this mode.
- The Oscillation Stop Detect circuit must be disabled before the main clock oscillator is stopped by setting the MOSTP bit in the MOSCCR.
- To use the Non-Maskable Interrupt for Oscillation Stop Detect, the OSTEN bit in the Non-Maskable Interrupt Enable Register (NMIER) must be set.
- Oscillation Stop Detect should only be enabled after the main clock oscillator has properly stabilized.
- 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 after an Oscillation Stop Detect event.

Figure 5 - Oscillation Stop Detect Recovery



Note: • On return from the oscillation-stopped state, the factor responsible for stopping the main clock oscillation circuit must be removed on the user system to allow the return of oscillation.

6. Reset Requirements and Reset Circuit

There are six types of resets:

Reset Name	Source
Pin reset	RES# is driven low
Power-on reset	VCC rices (voltage detection: VPOR)
Voltage-monitoring 1 reset	VCC falls (voltage detection Vdet1)
Voltage-monitoring 2 reset	VCC falls (voltage detection Vdet2)
Independent watchdog timer reset	The independent watchdog timer underflows, or a refresh occurs outside the permitted window (refresh error)
Software reset	Register setting

6.1 Pin Reset

When the RES# pin is driven low, all processing is aborted and the RX enters a reset state. To reset the MCU while it is running, RES# should be held low for the specified reset pulse width (minimum of 30 us). Refer to the “Electrical Characteristics” chapter of the Hardware Manual for more detailed timing requirements. Also refer to section 2, “Emulator Support” for details on reset circuitry in relation to debug support.

If you do not plan on using the internal Power-On Reset function (see next section), you must ensure that RES# is held low for at least 3 ms after VCC rises above Vpor. A much simpler solution is to use the internal POR circuit.

6.2 Power-On Reset

The built-in Power-On Reset circuit reduces the number of external components needed to safely start the MCU at power-up. To enable the Power-On Reset circuit, simply tie the RES# pin to VCC with a 4.7k resistor. 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 power-on reset time, tPOR 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.

Because the POR circuit relies on having RES# high concurrently with VCC, don’t place a capacitor on the reset pin. This will slow the rise time of RES# in relation to VCC, preventing the POR circuit from properly recognizing the power-on condition.

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 tPOR has elapsed.

After a power on reset, the PORF bit in RSTSR0 is set to 1; following a pin reset PORF is cleared to 0.

6.3 Voltage Monitoring Reset

The RX111 group includes 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. The trip levels of Vdet1 and Vdet2 are configurable.

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

Low Voltage Detection (Vdet1 and Vdet2) is disabled by default after reset; Vdet1 can be enabled out of reset by using the Option Function register OFS1. For more details, see the chapter “Voltage Detection Circuit (LVD)” in the hardware manual for details.

After an LVD Reset, the LVDnRF (n= 1, 2) bit in RSTSR0 is set to 1

6.4 Independent Watchdog Timer Reset

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

When the IWDT underflows, or if data is written outside the refresh-permitted period, an independent watchdog timer reset is optionally generated (NMI can be generated instead) and the UNDFF bit in the IWDTSR is set to a 1. After a short delay (tRESW2 = 300 µsec), the IWDT reset is canceled.

6.5 Software Reset

This is an internal reset generated by writing 0xA501 to the SWRR register. The internal reset time when using software reset is typically of 168 µsec. When using software reset, make sure that the watchdogs are serviced first before issuing the software reset command.

6.6 Determination of Cold/Warm Start

The CWSF bit in Reset Status Register 1 (RSTSR1) is cleared automatically by the MCU at power-on; it is not cleared by other reset sources. By reading and then setting this bit, software can tell whether the MCU was just powered on or whether the reset was caused by some other source (i.e. a pin reset or an independent watchdog reset). If the bit is clear, then a power-on reset has occurred; if the bit is set, then some other reset occurred.

6.7 Determining the Reset Source

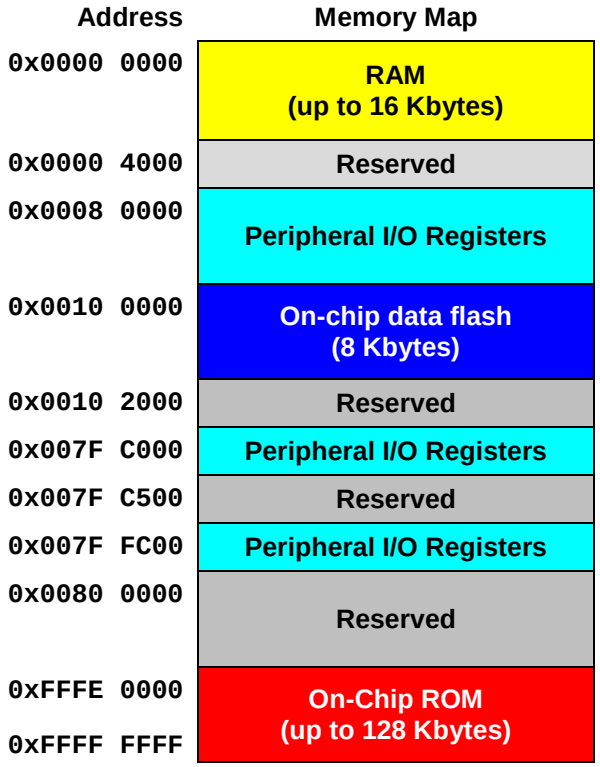
The RX MCU allows the user to determine the reset signal generation source. Refer to the hardware manual section 6.3.9 Determination of Reset Generation Source for the flow diagram.

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

```
#define RST_SRC_SW            0x001   /* Software reset */
#define RST_SRC_VDET2         0x004   /* Voltage monitor 2 reset */
#define RST_SRC_VDET1         0x008   /* Voltage monitor 1 reset */
#define RST_SRC_IWDT          0x020   /* Independent watchdog reset */
#define RST_SRC_POR           0x080   /* Power on reset */
#define RST_SRC_PIN           0x100   /* Pin reset */

int ResetSource ()
{
  /* Check for software reset */
  if (SYSTEM.RSTSR2.BIT.SWRF == 1) return (RST_SRC_SW) ;

   /* Check for voltage monitoring reset on Vdet2 */
  if (SYSTEM.RSTSR0.BIT.LVD2RF == 1) return (RST_SRC_VDET2) ;

  /* Check for voltage monitoring reset on Vdet1 */
  if (SYSTEM.RSTSR0.BIT.LVD1RF == 1) return (RST_SRC_VDET1) ;

  /* Check for independent watchdog timer (IWDT) reset */
  if (SYSTEM.RSTSR2.BIT.IWDTRF == 1) return (RST_SRC_IWDT) ;

  /* Check for power on reset */
  if (SYSTEM.RSTSR0.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) ;
}
```

7. Memory

The RX100 Series of MCU's have a 32-bit memory space spanning 4 Gbyte that includes areas for on-chip memory and peripherals. 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.



7.1 On-Chip RAM

Members of the RX100 include high-speed on-chip RAM that can be accessed in a single cycle at CPU speeds up to 32 MHz. Data stored in RAM is retained in all low-power modes of the CPU. Depending on the RX100 device, up to 16K of on-chip RAM is accessed starting at address 0x00000000.

7.2 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. 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.

7.3 Program ROM & Data Flash

The RX100 Series of MCUs feature two flash memory sections: program ROM and data flash. The program ROM is designed to store user application code and constant data. The data flash is designed to store 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 smaller than that of the program ROM (1 bytes for Data Flash versus 4 bytes for ROM). This makes the data flash better 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 APIs that can be used for modifying both ROM and data flash.

7.3.1 Enabling Data Flash Memory

Out of reset the data flash memory cannot be read, programmed, or erased. In order to allow read access the data flash must be enabled by setting the DFLEN bit in the DFLCTL register.

7.3.2 Blank Checking of Data Flash Memory

A data flash memory area can be checked to see if it is erased by verifying it contains all 0xFF's. This is different than RX600 & RX200 MCUs where erased memory had an undefined value.

7.3.3 Background Operation

RX100 MCUs support background operations for ROM and data flash. This means that when a program or erase is started, the user can keep executing and accessing memory from memory areas other than the one being operated on. For example, the CPU can execute application code from ROM while the data flash memory is being erased or programmed. Also, the CPU can execute application code from RAM while the ROM memory is being erased or programmed. The only exception to this rule is that the data flash cannot be accessed during ROM programming or erasing.

7.3.4 ID Code Protection

RX100 MCUs have a 32-byte memory area that is used as an ID code. If this ID code is left blank (0xFF's) then no protection is enabled and access to the MCU is allowed through boot mode or using the on-chip debugger. If the ID code is set then the user can control access to these modes. The user can choose to always disallow connections, or can choose to allow connections when a matching ID code is input. Refer to the "Flash Memory >> Flash Memory Access Disable Function" section of the hardware manual for more information.

7.4 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 RX100 Series, which is 32MHz. The FlashIF Clock (FCLK) controls the speed when reading data flash memory. The number of FCLK cycles it takes to read the data flash depends on the access width.

Access Size	8-bit	16-bit	32-bit
Number of FCLK cycles to read data from data flash	4 to 5	6 to 7	14 to 15

7.5 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.

7.6 Runtime ROM Protection

The RX111 has two different mechanisms to help protect user applications that perform self-programming. Startup-Up Program Protection is one feature that allows the top two 16KB areas of ROM to be swapped. This is useful when writing the fixed-vector table because the new table can be written in the bottom 16KB block, verified that it has been written, and then the blocks swapped. If a power down or reset were to occur during the writing of the bottom block, the fixed-vector table would still be intact in the top block. This feature is only available on MCUs that have at least 32KB of ROM. For more information on this feature please see the "Flash Memory >> Start-Up Program Protection" section of the hardware manual.

The Area Protection feature can be also be used for runtime ROM protection. This feature works by setting an upper and lower address for ROM blocks that can be rewritten using self-programming. When set, only ROM blocks that are below the upper address and above the lower address can be modified. If self-programming is attempted on a ROM block that is outside of these addresses then the operation will fail. This feature does not pertain to Boot Mode where all ROM blocks can always be rewritten. For more information on this feature please see the "Flash Memory >> Area Protection" section of the hardware manual.

8. Register Write Protection

The Register Write Protection function protects certain registers in the MCU from inadvertent changes by runaway or buggy software. The system Protect Register (PRCR) contains bits that enable writing to other registers in the MCU. PRCR is a sixteen bit register with a key in the upper byte and the protection bits in the lower byte. A key code of A5 hex must be written to the upper 8 bits of PRCR to modify any of the lock bits in the lower byte. Setting a PRC bit to a 1 allows writing of the protect registers. Protection is enabled by default after reset (all PRC bits are zero after reset).

Figure 6 - PRCR Register

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PRKEY[7:0]								—	—	—	—	PRC3	PRC2	PRC1	PRC0

PRCR bit	Description
PRC0	• Registers in the **Clock Generation Circuit** (CGC) that control operation of the MCU's clocks: SCKCR, SCKCR3, PLLCR, PLLCR2, MOSCCR, SOSCCR, LOCOCR, ILOCOCR, HOCOCR, OSTDCR, OSTDSR, CKOCR
PRC1	• Registers related to the **operating modes**:SYSCR1 • Registers related to the **low power consumption** functions: SBYCR, MSTPCRA, MSTPCRB, MSTPCRC, OPCCR, RSTCKCR, SOPCCR • Registers related to **clock generation circuit**: MOFCR, MOSCWTCR • **Software reset** register: SWRR
PRC2	• Registers related to the **clock generation circuit** • HOCOWTCR
PRC3	• Registers related to the **Low Voltage Detection** circuit (LVD): LVCMPCR, LVDLVLR, LVD1CR0, LVD1CR1, LVD1SR, LVD2CR0, LVD2CR1, LVD2SR
PRKEY[7:0]	Write 0xA5 to these bits to allow writing to the PRC bits. To modify the system protection, write 0xA5 to the PRKEY bits while setting the PRC bits.

8.1 System Protection Example Code

To unlock all registers, write 0xA50F to the PRCR. Set it back to 0xA500 to protect them all again.

```
/* Disable write protection for all protected registers */
SYSTEM.PRCR.WORD = 0xA50F;
/* Change system clock divisors */
SYSTEM.SCKCR.LONG = 0x21000202 ;
/* Turn write protection back on for all protected registers */
SYSTEM.PRCR.WORD = 0xA500;
```

9. I/O Ports and 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: SCI5, ICU, S12AD, etc.

Register is the register abbreviation for a specific register such as: SCR, IPR, ADCSR, 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 ADIE.

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 5, Receive Data Register (RDR)	8	SCI5.RDR
SCI Channel 5, Serial Control Register (SCR)	8	SCI5.SCR.BYTE
SCI Channel 5, Receive enable bit in SCR	1	SCI5.SCR.BIT.RE
Port 2, Pin 5, Port Direction Register Bit	1	PORT2.PDR.BIT.B5
CMT0 Control Register clock selection bits	2	CMT0.CMCR.BIT.CKS
CMT0 Control Register – whole register	16	CMT0.CMCR.WORD

9.1 I/O Register Macros

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(SCI5,TXI5) == 1)…`
`IEN("module name", "bit name")`	`IEN(SCI5,TXI5) = 1 ;`
`IPR("module name", "bit name")`	`IPR(SCI5,TXI5) = 0x02 ;`
`MSTP("module name")`	`MSTP(SCI5) = 0 ;`
`VECT("module name", "bit name")`	`#pragma interrupt (MySciTxIsr(vect=VECT(SCI5,TXI5))`

9.2 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[28].BIT.IR = 0;`	`IR(CMT0, CMI0) = 0;`
`ICU.DTCER[28].BIT.DTCE = 1;`	`DTCE(CMT0, CMI0) = 1;`
`ICU.IER[3].BIT.IEN4 = 1;`	`IEN(CMT0, CMI0) = 1;`
`ICU.IPR[4].BIT.IPR = 3;`	`IPR(CMT0, CMI0) = 3;`

9.3 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_CMI0(vect=28))
void INT_CMI0 (void) ;
```

The above example hooks the C language function “INT_CMI0” to interrupt vector number 28, which is the compare match interrupt for CMT0. This same interrupt source (CMI0) may not use the same vector number (28) 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: SCI5, CMT0, AD0, etc.

Source is the name of an interrupt source in that peripheral such as: RXI5, CMI0, ADI0, etc.

Example:

Without Macro

```
/* Declare ISR for CMT0 – CMI0 */
#pragma interrupt CMT0_CMI0(vect=28)
```

With Macro

```
/* Declare ISR for TMR0 – CMI0 */
#pragma interrupt CMT0_CMI0(vect=VECT(CMT0,CMI0))
```

9.4 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 SCI5 */`	`/* Turn on SCI5 */`
`SYSTEM.MSTPCRB.BIT.MSTPA26 = 0;`	`MSTP(SCI5) = 0;`

Care should be taken when using the MSTP() macro because sometimes multiple peripheral channels will map to the same MSTP bit. For example, the MSTPA15 bit controls CMT0 and CMT1 (both channels are part of CMT unit 0). This means that both MSTP(CMT0) and MSTP(CMT1) will resolve to SYSTEM.MSTPCRA.BIT.MSTPA15. This is not a problem when powering on a peripheral but could cause a problem when powering down. If the user turns off CMT0 to save power by using 'MSTP(CMT0) = 1;' then they will also turn off CMT1 even if they did not intend to. The user can avoid this problem by always checking to make sure both channels are not in use before powering down.

9.5 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.

10. I/O Port Configuration and the Multifunction Pin Controller (MPC)

The I/O Ports and Multifunction Pin Controller (MPC) sections of the Hardware Manual describe exact pin configurations based on peripheral selection and other register settings. Some general information is listed below.

10.1 Setting Up and Using Port as GPIO

- Select a pin as an output by writing a "1" to the corresponding Port Direction Register (PDR)
- The Port Direction Register (PDR) is read/write. Setting the value to a "1" selects the pin as an output. Default state for I/O Ports is "0" (input). The port direction registers can be read on the RX.
- The Port Output Data Register (PODR) is read/write. When the PODR is read the state of the output data latch (not the pin level) is read.
- The Port Input Register (PIDR) is read only. Read the PIDR register to read the pin state.
- The Port Mode Register (PMR) is read/write and is used to specify whether individual pins function as GPIO or as peripheral pins. Out of reset all PMR registers are set to 0 which sets all pins to work as GPIO. If a PMR register is set to 1 then that corresponding pin will be used for peripheral functions. The peripheral function is defined by that pin's MPC setting.
- 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.PDR.BIT.B1 = 0;
/* See if input is high */
if (PORT0.PIDR.BIT.B1 == 1) …
```

  Set up Port 0, bit 1 as an output:

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

10.1.1 Internal Pull-Ups

- Most pins have optional internal pull-up resistors that can be enabled with the Pull-up Control Registers (PCR). Each bit in the PCR register controls the corresponding bit on the port. Set the PCR bit to "1" to enable the pull-up and "0" to disable it.
- Out of reset all PCR registers are cleared to 0; therefore, all on-chip pull-ups are disabled.
- The pull-up is automatically turned off whenever a pin is designated as an output, or assigned as a peripheral function output pin.

  Enable pull-up on Port 0, bit 1:

```
/* Port pin P01 requires a pull-up */
PORT0.PCR.BIT.B1 = 1;
```

10.1.2 Open-Drain Output

- Pins configured as outputs normally operate as CMOS outputs
- Many pins have the option of being configured as N-channel open-drain outputs, and some can be configured as P-channel open-drain outputs
- The Open Drain Control Registers (ODR0 and ODR1) control which pins operate in open-drain mode. The ODR0 registers control the settings for pins 0 through 3 on a port, and the ODR1 registers control the settings for pins 4 through 7 on each port. Check the "I/O Ports" section in the hardware manual for your device for specific pin settings.

Set Port A, bit 3 as N-channel open-drain output:

```
/* Set the output level first to prevent glitches */
PORTA.PODR.BIT.B3 = 1;
/* Set PA3 to open-drain */
PORTA.ODR0.BIT.B3 = 1;
/* Make pin an output */
PORTA.PDR.BIT.B1 = 1;
```

10.2 Setting Up and Using Port Peripheral Functions

The Multifunction Pin Controller (MPC) allows flexible run-time assignment of peripheral functions to I/O pins. It allows mapping of peripheral functions to various places around the chip's package, making board design and layout easier.

- Each pin has a Pin Function Control register (PmnPFS where m = port, n = pin). For example, the P10PFS register allows you to choose pin functions for pin 0 on port 1.
- Each PmnPFS register can contain up to 3 fields: ASEL, ISEL, and PSEL.

- The ASEL bit is present only for pins that support analog functions. Set this bit to a 1 to use the pin in analog mode. In addition, the pin's bit in the Port Mode Register (PMR) should be set to 0, and the pin's direction (set in the PDR) should be set to input.
- The ISEL bit is present only for pins that support IRQ interrupt functionality. Set this bit to a 1 to use the pin as an IRQ pin.
- The PSEL bits are present in every PmnPFS register. These bits determine the peripheral function to assign to the pin. Refer to the appropriate register in the MPC chapter of the hardware manual for a table of the available peripheral functions for each pin for your device and packages.
- In order to ensure that no unexpected edges are input or output on peripheral pins make sure to clear the PMR bit for the targeted pin before modifying the pin's PmnPFS register.
- All PmnPFS registers are write protected out of reset. In order to write to these registers the Write-Protect Register (PWPR) must first be used to enable writing.
- Care should be taken when setting PmnPFS registers such that a single function is not assigned to multiple pins. The user should not do this but the MCU will allow it. If this occurs the function on the pins will be undefined.
- The example below shows the steps for setting port C, bit 2 to be a SCI receiver input pin (RXD5). These steps are defined in the Multi-Function Pin Controller (MPC) >> Usage Notes >> Procedure for Specifying Input/Output Pin Function section of the RX111 HW manual.

Example - Enabling SCI5 to use port C, bit C as SCI receiver input pin

```
/* Allow writing to MSTP registers. */
SYSTEM.PRCR.WORD = 0xA50B;
/* Enable SCI5 (take out of stop mode) */
MSTP(SCI5) = 0;
/* Configure SCI5 */
...
/* Clear PMR bit for PC2 before changing PC2PFS register. */
PORTC.PMR.BIT.B2 = 0;
/* Set PC2 as input pin. */
PORTC.PDR.BIT.B2 = 0;
/* Unlock protection register */
MPC.PWPR.BIT.B0WI = 0;
/* Unlock MPC registers */
MPC.PWPR.BIT.PFSWE = 1;
/* Set PC2 to be used for RXD5 function. */
MPC.PC2PFS.BYTE = 0x0A;
/* Assign PC2 to be used for peripheral function. */
PORTC.PMR.BIT.B2 = 1;

/* Set other port registers and re-enable MPC & MSTP register protection. */
```

10.3 Setting Up and Using IRQ Pins

- Certain port pins can be used as hardware interrupt lines (IRQ). See the Multi-Function Pin Controller (MPC) “Overview” section of the HW Manual for information on which pins are available for your MCU.
- To set a port pin to be used as an IRQ pin, the Interrupt Input Function Select bit (ISEL) in the pin’s PFS register must be set to “1”.
- Pins can be used for both IRQ and peripheral functions simultaneously. To enable this the user should set both the ISEL and PSEL bits in the pin’s PFS register.
- IRQ pins can trigger interrupts on detection of:
    - Low level
    - Falling edge
    - Rising edge
    - Rising and falling edges

  Which trigger is selected is chosen using the IRQ Control Registers (IRQCRi).
- Digital filtering is available for IRQ pins. The filters are based on repetitive sampling of the signal at one of four selectable clock rates (PCLK, PCLK/8, PCLK/32, PCLK/64). They filter out short pulses: any high or low pulse less than 3 samples at the filter rate. The filters are useful for filtering out ringing and noise in these lines, but are much too quick for filtering out long events like mechanical switch bounce. Enabling filtering adds a short bit of latency (the filter time) to the hardware IRQ lines.
- Digital filtering can be enabled for each IRQ pin independently. This is done by setting the IRQ Pin Digital Filter Enable Registers (IRQFLTEi).
- The clock rate for digital filtering is configurable for each IRQ pin independently. This is done by setting the IRQ Pin Digital Filter Setting Registers (IRQFLTCi).
- The example below shows code to enable IRQ3 with falling edge detection on port 2, pin 7.

Example - Enabling port 2, bit 7 as IRQ3 input

```
/* P27 is not being used for peripheral function. */
PORT2.PMR.BIT.B7 = 0;
/* Make pin an input */
PORT2.PDR.BIT.B7 = 0;
/* Unlock protection register */
MPC.PWPR.BIT.B0WI = 0;
/* Unlock MPC registers */
MPC.PWPR.BIT.PFSWE = 1;
/* Set P27 to be used for IRQ function. */
MPC.P27PFS.BYTE = 0x40;
/* Set IRQ type (falling edge) */
ICU.IRQCR[3].BIT.IRQMD = 0x01;
/* Clear any pending interrupts. */
IR(ICU,IRQ3) = 0;
/* Set interrupt priority to 3 */
IPR(ICU,IRQ3) = 0x03;
/* Enable the interrupt */
IEN(ICU,IRQ3) = 1;
/* Be sure to write an interrupt handler!!! */
```

10.4 Unused Pins

NOTE:

Some pins require specific termination: See the "I/O Ports: Handling 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.

10.5 Non-Existent Pins

When using a MCU with less than 64 pins, set the corresponding bits of nonexistent ports in the PDR register to "1" (output) and in the PODR register to "0". The user can see which ports are available on each MCU package by reviewing the "Specifications of I/O Ports" table in the I/O Ports section of the HW Manual. For example pins 3 and 5 on port 0 are only available on the 64 pin package. A separate application note covers software startup of the chip including initialization of nonexistent pins. See the "Initial Setting" application note in the References section of this document.

10.6 Electrical Characteristics

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

10.7 MPC Register Setting Summary

The following table can be found in section 19.3 of the Hardware Manual. Additional information can be found there.

Table 6 - Register Settings for GPIO

			PmnPFS			
Item	**PMR.Bn**	**PDR.Bn**	**ASEL**	**ISEL**	**PSEL[4:0]**	**Point to note**
After a reset	0	0	0	0	00000b	Pins function as general input port pins after release from the reset state.
General input ports	0	0	0	0/1	X	Set the PmnPFS.ISEL bit to 1 if these are multiplexed with interrupt inputs.
General output ports	0	1	0	0	X	
Peripheral functions	1	X	0	0/1	Peripheral function setting	Set the PmnPFS.ISEL bit to 1 if these are multiplexed with interrupt inputs.
Interrupt inputs	0	0	0	1	X	
NMI	X	X	X	X*	X	Register settings are not required
Analog inputs and outputs	0	0	1	X*	X	Set these as general input port pins so that the output buffers are turned off
XCIN	0	0	X	X*	X	Set these as general input port pins so that the output buffers are turned off.

X: setting not required

0/1: Setting the PmnPFS.ISEL bit to 0 makes the pin incapable of functioning as an IRQ pin.

Setting the PmnPFS.ISEL bit to 1 makes the pin capable of functioning as an IRQ pin

*Even if the PmnPFS.ISEL bit is set to 1, the pin will not function as an IRQn input pin.

11. Module Stop Function

To maximize power efficiency, the RX family of 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 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 out of stop mode by writing a '0' to the corresponding bit in the MSTPCRi register. The MSTPCRi registers are protected registers and they have to be unprotected by writing to the PRCR registers first (see section 8 - Register Write Protection). See example below.

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.

Example – Turning on SCI5 using the MSTP macro

```
/* Disable write protection for the MSTP registers */
SYSTEM.PRCR.WORD = 0xA502;
/* Enable SCI5 (take out of stop mode) */
MSTP(SCI5) = 0 ;
/* Enable write protection for the MSTP registers */
SYSTEM.PRCR.WORD = 0xA500;
/* You can now access SCI5 control registers */
```

12. Interrupts

The RX family has a sophisticated Interrupt Control Unit (ICU) that handles asynchronous events from over 200 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.

To handle simultaneous interrupt requests from multiple sources, the ICU 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 its 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 in both the Port setup registers and the Multifunction Pin Controller Registers.
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.

12.1 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.

12.2 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.

12.2.1 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. Also included in the fixed vector table are some locations that are reserved for system configuration and ROM protection. Figure 7 - Fixed Vector Table shows the fixed vector table.

Figure 7 - Fixed Vector Table

Address	Description
FFFFFF80h	Endian select register (MDE) in single-chip mode
FFFFFF84h	(Reserved)
FFFFFF88h	Option Function Select Register 1 (OFS1)
FFFFFF8Ch	Option Function Select Register 2 (OFS2)
FFFFFF90h – FFFFFFCCh	(Reserved)
FFFFFFA0h – FFFFFFACh	ID Code for flash protection
FFFFFFD0h	Privileged instruction exception
FFFFFFD4h	(Reserved)
FFFFFFD8h	(Reserved)
FFFFFFDCh	Undefined instruction exception
FFFFFFE0h – FFFFFFF4h	(Reserved)
FFFFFFF8h	Non-maskable interrupt
FFFFFFFCh	Reset

12.2.2 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 8 - Relocatable Vector Table 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.

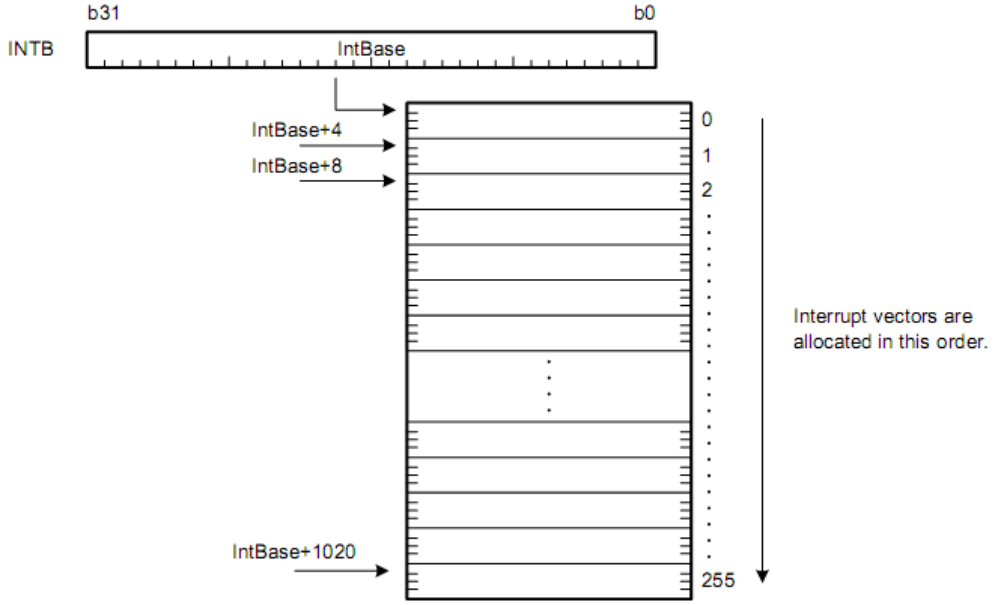


Figure 8 - Relocatable Vector Table

12.3 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.

12.4 Interrupt Stack Pointers

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.

13. Low Power Consumption

The RX111 has register settings that allow the MCU to operate with lower power consumption. These modes are referred to as the Operating Power Control Modes and are controlled by the OPCCR and SOPCCR registers. The three Operating Power Control Modes, High, Medium, and Low speed operating mode, switch on-chip regulators on and off which limit maximum system frequencies in lower power modes. Power savings are realized by setting an appropriate power control mode for the current system clock settings.

Below is a summary of these modes and the maximum permissible clocking and voltage levels under each mode.

Table 7 - Operating Frequency and Voltage Ranges in Operating Power Control Modes

Operating Power Control Mode	OPCM [2:0] Bits	SOPCM Bit	Operating Voltage Range	Operating Frequency Range				
				Flash Memory Read Frequency				Flash Memory Programming/ Erasure Frequency
				ICLK	FCLK	PCLKD	PCLKB	FCLK
High-speed operating mode	000b	0	2.7 to 3.6V	Up to 32 MHz	Up to 32 MHz	Up to 32 MHz	Up to 32 MHz	1 to 32 MHz
			2.4 to 2.7V	Up to 16 MHz	Up to 16 MHz	Up to 16 MHz	Up to 16 MHz	---
			1.8 to 2.4V	Up to 8 MHz	Up to 8 MHz	Up to 8 MHz	Up to 8 MHz	---
Middle-speed operating mode	010b	0	2.4 to 3.6V	Up to 12 MHz	Up to 12 MHz	Up to 12 MHz	Up to 12 MHz	1 to 12 MHz
			1.8 to 2.4V	Up to 8 MHz	Up to 8 MHz	Up to 8 MHz	Up to 8 MHz	1 to 8 MHz
Low-speed operating mode	000b	1	1.8 to 3.6V	Up to 32.768 kHz	Up to 32.768 kHz	Up to 32.768 kHz	Up to 32.768 kHz	---
	010b	1	1.8 to 3.6V					

In order to achieve the lowest power numbers, use the maximum possible dividers in the clock generation circuits.

Note that when switching from a lower power, lower speed mode to a higher power, higher speed mode, change the Operating Power Control Mode first by writing to the OPCM and SOPCM bits first (which enables on-chip regulators) and then switch to the higher frequency clock. Conversely, when switching from high power, high speed mode to lower power, lower speed mode, make the clock source and speed settings first and then write the OPCM and SOPCM bits to switch off the regulators.

There are some restrictions on which oscillators can be used in each Operating Power Control Mode:

Table 8 - Oscillator Usability in Power Control Modes

	PLL	HOCO	LOCO	Main Clock Oscillator	Sub-Clock Oscillator
High-speed operating mode	Usable if VCC > 2.4V	Usable	Usable	Usable	Usable
Middle-speed operating mode	Usable if VCC > 2.4V	Usable	Usable	Usable	Usable
Low-speed operating mode	Not usable	Not usable	Not usable	Not usable	Usable

In addition to the Operating Power Control Modes, the RX111 also has Low Power Consumption modes where the CPU is stopped and the other peripherals/clocks are switched on or off depending on how much power needs to be conserved. These modes are activated by configuring the appropriate control registers (refer to Figure 11.1 in the hardware manual) and then executing the wait() instruction. Since the low power modes can be exited on receiving an interrupt, all pending requests should be handled and the I flag cleared prior to executing the wait() instruction.

Table 9 - Low Power Consumption CPU Modes

	Sleep Mode	Deep Sleep Mode	Software Standby Mode
Entry trigger	Control register + instruction	Control register + instruction	Control register + instruction
Exit trigger	Interrupt	Interrupt	Interrupt
After exiting from each mode, CPU begins from	Interrupt handling	Interrupt handling	Interrupt handling
Main clock oscillator	Operating possible	Operating possible	Stopped
Sub-clock oscillator	Operating possible	Operating possible	Operating possible
High-speed on-chip oscillator	Operating possible	Operating possible	Stopped
Low-speed on-chip oscillator	Operating possible	Operating possible	Stopped
IWDT-dedicated on-chip oscillator	Operating possible	Operating possible	Operating possible
PLL	Operating possible	Operating possible	Stopped
CPU	Stopped (retained)	Stopped (retained)	Stopped (retained)
RAM0 (0000 0000h to 0000 3FFFh)	Operating possible (retained)	Stopped (retained)	Stopped (retained)
DTC	Operating possible	Stopped (retained)	Stopped (retained)
Flash memory	Operating	Stopped (retained)	Stopped (retained)
Independent watchdog timer	Operating possible	Operating possible	Operating possible
Real time clock (RTC)	Operating possible	Operating possible	Operating possible
Voltage detection circuit (LVD)	Operating possible	Operating possible	Operating possible
Power-on reset circuit	Operating	Operating	Operating
Peripheral modules	Operating possible	Operating possible	Stopped (retained)
I/O ports	Operating	Operating	Retained
RTCOUT	Operating possible	Operating possible	Operating possible
CLKOUT	Operating possible	Operating possible	Operating possible

14. References

The following documents were used in creating this Quick Design Guide:

Reference	Document Number	Description
1	R01UH0365EJ0100_RX111	RX111 Group User's Manual: Hardware
2	R20UT0398EJ0300_E1E20	E1 Emulator E20 Emulator User's Manual
3	R20UT0399EJ0600_E1E20_RX	E1/E20 Emulator Additional Document for User's Manual (RX User System Design)
4	R01AN0252ET0101_RX	RX Family Debug Console Function Using E1

Website and Support

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

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

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Dec 2, 2013	—	Initial release

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation 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.
2. 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.
3. 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.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.
6. 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.
7. 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 systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. 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.
9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
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: +44-1628-651-700, Fax: +44-1628-651-804

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.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

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 2.2