Renesas RA8 Series

Getting Started with RA8 Power Management Techniques

Introduction

This application note is intended to provide details on how to best utilize Power Management resources of the RA8 family of devices. This application note highlights the Low power features/modes of the RA8x1 MCUs. With the high-performance capability of the RA8 devices, power consumption can be a critical consideration for customers, and minimizing the power will be important for many designs.

This Application note includes techniques to minimize overall power consumption,

- Using Low speed active modes and CPU sleep modes
- Using the retention feature for context save and faster wake-up
- Using clock and timers
- Using voltage and frequency scaling and multiple power domains
- Low Power modes with full SRAM retention.
- Using the MCU peripherals such as DMAC, DTC, ELC, DCDC converter etc. in power saving modes.
- Using lower operating voltage range of (down to 1.8V) and the second Vcc2 domain to interface to 1.8V external devices.
- Any additional methods/tips/techniques such as hardware PCB techniques, software options, or innovative use of RA8 peripherals.

Required Resources

- Flexible Software Package (FSP) v5.2.0

Target devices

- RA8x1 (RA8M1, RA8D1, RA8T1)

Contents

1. Overview of Power Management

Power Management techniques are crucial when working with MCU to extend battery life and optimize energy consumption. Efficient management of power is vital when designing embedded systems, and it involves controlling and optimizing the consumption and distribution of electrical power within a device. Effective power management is particularly important in battery-operated devices, portable electronics, and energy-conscious applications.

In embedded designs, power management and power efficiency go hand-in-hand, and are crucial for ensuring that you have a viable end product, as well as an environmentally-sound one. This is true for a vast range of embedded products today across verticals from industrial applications to medical devices and other mission critical use cases.

2. Power Saving Modes

Power-saving modes are specific operational states or configurations in the embedded devices, especially MCUs, which can minimize power consumption when the device is not actively performing tasks or when full performance is not required. These modes help extend battery life, improve energy efficiency, and reduce heat generation.

The RA MCUs have several functions for reducing power consumption, such as setting clock dividers, clock gating, stopping peripheral modules, power gating control, operating MCU in normal mode and transitioning to low power modes during the CPU idle etc.

Power-saving modes are important for several reasons. Some of the key reasons are extended Battery Life, Environmental Impact, Cost savings, Compliance and regulation.

2.1 Low Power Modes

MCUs provide various low-power modes to reduce power consumption when the device is not actively running. The specific low-power modes available in a MCU can vary depending on the architecture. Here are some common low-power modes found in Renesas RA MCU

Sleep Mode: An operational CPU is typically the primary cause of power consumption. In Sleep mode, the CPU stops operating, but the contents of its internal registers are retained. Other peripheral functions in the MCU do not stop. Available resets or interrupts, in Sleep mode can cause the MCU to cancel Sleep mode.

Software Standby mode: In Software Standby mode, the CPU, most of the on-chip peripheral functions and the oscillators stop operation. However, the contents of the CPU internal registers and the SRAM data, the states of the on-chip peripheral functions, and the I/O port states are retained. Software Standby mode allows a significant reduction in power consumption since most of the oscillators stop in this mode.

Deep Software Standby Mode: In this mode, the MCU disables the CPU clock and shuts down the majority of peripherals components, achieving further power savings. This mode is suitable for applications with infrequent wake-up events.

Apart from the 3 commonly used modes, some RA MCU also provides other modes such as snooze mode, standby mode which are small variation to the commonly used LPM modes.

RA MCU also supports different power control modes to reduce the power consumption as part of the LPM. Power consumption can be reduced in Normal, Sleep, and standby mode by selecting an appropriate operating power control mode according to the operating frequency and voltage.

Four operating power control modes are available:

- High-speed mode
- Middle-speed mode
- Low-speed mode
- Subosc-speed mode

2.2 CPU Sleep Modes

CPU sleep mode is a low-power state, in which the central processing unit (CPU) of the MCU temporarily halts its normal operation to conserve power. Instead of actively executing instructions, the CPU enters a sleep state, where it consumes less power and generates less heat.

In RA8 MCU low power mode can be achieved using the CPU Sleep mode

- CPU Sleep mode
- CPU Deep Sleep mode

When RA MCU enters CPU Sleep mode, the CPU stops operating but the contents of its internal registers are retained. Other peripheral functions do not stop. Available resets or interrupts, in CPU Sleep mode can cause the MCU to cancel CPU Sleep mode.

Whereas when RA MCU enters CPU Deep Sleep mode, the CPU stops operating, but the contents of its internal registers are retained. Other peripheral functions do not stop. However, accessing to TCM is not available and SysTick also stops in this mode. Available resets or wakeup enabled interrupts in CPU Deep Sleep mode can cause the MCU to cancel CPU Deep Sleep mode.

2.3 Power Gating

Turning off power to entire blocks or subsystems when they are not needed helps conserve power. Power gating is commonly used in low-power designs to completely shut down sections of the device during idle periods. All blocks are not operational all the time inside the MCU; it depends on their application in the device. There is no need to supply power to a block if it is not functional in a particular instance. By turning off the power supply to non-functional blocks, power consumption can be reduced. To efficiently use this technique designers can use isolation blocks to prevent unnecessary signals coming from power-gated blocks.

2.4 Frequency Gating

Frequency gating, also known as dynamic frequency scaling (DFS), is a power-saving technique in MCUs that involves adjusting the operating frequency of the CPU based on the workload or system requirements. By dynamically scaling the frequency, the MCU can match its processing speed to the current demands of the application, resulting in reduced power consumption during periods of low activity. Here are key aspects of frequency gating to save power in MCUs:

As we know, MCUs consist of multiple blocks, and each block does not require the highest possible frequency to operate. The best technique can be to segregate the blocks based on their frequency requirement and provide a different clock signal to a different block. This method can greatly reduce the localized dynamic power consumption of the MCU.

2.5 Clock Gating

Clock gating is one of the techniques used in MCUs to reduce dynamic power consumption by selectively disabling the clock to specific functional blocks or modules when they are not in use. By controlling the clock signal to these modules, clock gating helps to minimize the power consumed by the digital circuits during idle or low-activity periods.

Each functional block or peripheral in an MCU is associated with a clock enable or disable option. When a particular module is not in use, the clock enable signal can be turned off effectively stopping the clock signal to that module. When the clock is gated, the logic elements within the module do not switch states, resulting in reduced power consumption associated with dynamic power dissipation.

RA MCUs and FSP offer software-configurable clock gating, allowing the firmware to control which modules have their clocks gated based on the application's needs.

2.6 Multiple Voltage/Power Domains

Multiple power domains in a MCUs are employed to save power by selectively turning off power to specific sections or peripherals when they are not in use. This technique is particularly useful in battery-powered and energy-efficient devices.

The MCU's CPU and associated components are placed in a separate power domain. The core power domain can be turned off during periods of inactivity, and the CPU can be put into a low-power state.

Multiple voltage domains in MCU is another advanced power management technique that helps optimize power consumption. By providing different voltage levels to various components or sections of the MCU.

Different peripheral modules may have varying voltage requirements. Grouping peripherals with similar voltage needs into separate voltage domains allows for independent voltage scaling and power optimization.

2.7 Voltage and Frequency Scaling

Reducing the operating voltage of the device is a useful step to reduce the overall power consumption. When running, power consumption is mainly influenced by the clock speed. When sleeping, the most significant factor is leakage in the transistors. At lower voltages, less charge is required to switch the system clocks and transistors leak less current.

RA MCU has multiple internal and external clock sources, and it allows switching between the available clock sources as the main system clock. This allows for great power savings by choosing different clocks for different portions of code. For example, an application can use the slower internal oscillator when executing non-critical code and then switch to a fast high-accuracy oscillator for time or frequency sensitive code.

Clock switching offers significantly more flexibility in applications compared to being constrained to a single clock source. The sequences for clock switching differ across device families, thus it's essential to refer to RA MCU device datasheets for the precise clock switching procedures.

2.8 Power Efficient Hardware PCB Design

Power-efficient hardware Printed Circuit Board (PCB) design is crucial for optimizing energy consumption in electronic devices. The following techniques can be considered while designing the HW PCB

- Choose low-power components or modules which supports Low power modes
- Minimize trace length and width
- Usage of proper grounding and decoupling
- Reduce switching frequency and duty cycle
- Implement power management strategies
- Implement Efficient Power Sequencing using Power sequencers

When powering circuits, set the system voltage as low as possible for lower power consumption. For example, a MCU running at 1 MIPS at 3.3 V can draw 700 µA. Lower the voltage to 1.8 V while keeping the same speed, and the current draw becomes 370 µA—almost a 50% savings.

Low Power components: Select components that are specifically designed for low power consumption. This includes MCU, voltage regulators, and other ICs (Integrated Circuits) with low standby and active power requirements.

Efficient Power Supplies: Use switch-mode power supplies (SMPS) instead of linear regulators whenever possible. SMPS are generally more energy-efficient because they can convert voltage with less power loss. Additionally, choose power supplies that have low standby power consumption.

Efficient Ground and Power Planes: Design the PCB with well-planned ground and power planes to minimize resistance and inductance, reducing power losses and improving overall efficiency.

Reduced Parasitic Capacitance and Inductance: Minimize parasitic capacitance and inductance in the layout, as these can contribute to power losses. Properly manage trace lengths, keep components close together, and use short and wide traces where possible.

Optimized PCB Layout: Carefully design the PCB layout to minimize signal crosstalk, impedance mismatches, and unnecessary trace lengths. A well-optimized layout can reduce power losses and improve signal integrity.

Implement a well-defined power sequencing strategy to ensure that components power up and power down in a controlled manner, minimizing the risk of voltage spikes and unnecessary power consumption.

Smart Sensors and Actuators: Use sensors and actuators that can operate in low-power modes and wake up only when needed. This is especially relevant in IoT (Internet of Things) and sensor node applications.

Energy Harvesting: Explore energy harvesting techniques, such as solar cells or vibration-based harvesters, to supplement or recharge the power source in energy-constrained applications.

2.9 Firmware Based Power Efficiency Optimization Technique

Designing firmware for the power efficient systems requires good usage of available MCU Hardware resources, well tested software algorithms, protocols and Middleware, RTOS etc.

To start with, Bootloader plays a small role in the power saving of an embedded system, and minimizing boot time is often a critical requirement.

On top of the bootloader, firmware update algorithm needs to be efficient to perform the updates as required. While updating the firmware, the wired or wireless communication module has to be reliable and support the Low power modes when not performing the Job.

Application code also needs to be designed by taking the Power saving mode into consideration. Code needs to use the efficient algorithms, well tested libraries and reasoned RTOS when developing the application.

Implement an efficient task scheduler to ensure that the processor enters low-power modes during idle times. Group and prioritize tasks to minimize wake-ups and transitions between power states.

Implement intelligent sensor sampling strategies. Use interrupts or event-driven mechanisms to wake the system only when sensor data is required. Also configure the sensors to low power mode if it is available.

Choose communication protocols that are optimized for low power. Minimize the use of high-power communication interfaces and prefer low-power alternatives like I2C or SPI.

3. RA8 MCU Power Saving Features

From the above sections it is clear that including and implementing the power saving features of MCUs can greatly contribute to the power efficient designs. RA8 MCUs provides Low Power modes, Clock gating, Power Domains, Peripheral Control to achieve this. In this section, we will describing more details of the RA8 MCU specific features and its usage for saving the power in the system.

3.1 RA8 Low Power Modes

RA8 MCU provides different Low power mode options for the users. Below Figure 1 shows the different Low power functions and its specifications.

Item	Specification
Reducing power consumption by switching clock signals	The frequency division ratio can be selected independently for the CPU clock (CPUCLK), system clock (ICLK), peripheral module clocks (PCLKA, PCLKB, PCLKC, PCLKD, PCLKE), external bus clock (BCLK), and flash interface clock (FCLK). *1
EBCLK output control	BCLK output or high-level output can be selected.*1
SDCLK output control	SDCLK output or high-level output can be selected.
Module stop	Functions can be stopped independently for each peripheral module
Power gating control	This function can be controlled the power state of the power domain. • Control the turning On/OFF for the power domain • Control the retention of specific circuits during power gating
Processor low power modes	• CPU Sleep mode • CPU Deep Sleep mode
Low-power modes	• Software Standby mode*2 • Deep Software Standby mode 1, 2, 3*2
Operating power control modes	• Power consumption can be reduced in Normal and Processor low power mode by selecting an appropriate operating power control mode according to the operating frequency. • Two operating power control modes are available: High-speed mode Low-speed mode*2
TrustZone Filter	Security and Privilege attribution can be set

Note 1. For details, see section 8, Clock Generation Circuit
Note 2. This mode is not supported in external VDD mode.

Figure 1. Specifications of the low power mode functions in RA8M1

RA8 provides different clock sources such as XTAL, HOCO, MOCO, LOCO and SUBCLK. Furthermore, it offers various clock division ratios that can be chosen for CPU Clock , System Clock, Peripheral clocks and external bus clocks. This can be selected using the clocks tab of the FSP configurator as shown in the Figure 2. Operating the MCU and its peripherals from different clock sources and selectively lowering frequencies can save the power by lowering the speed as required by the system.

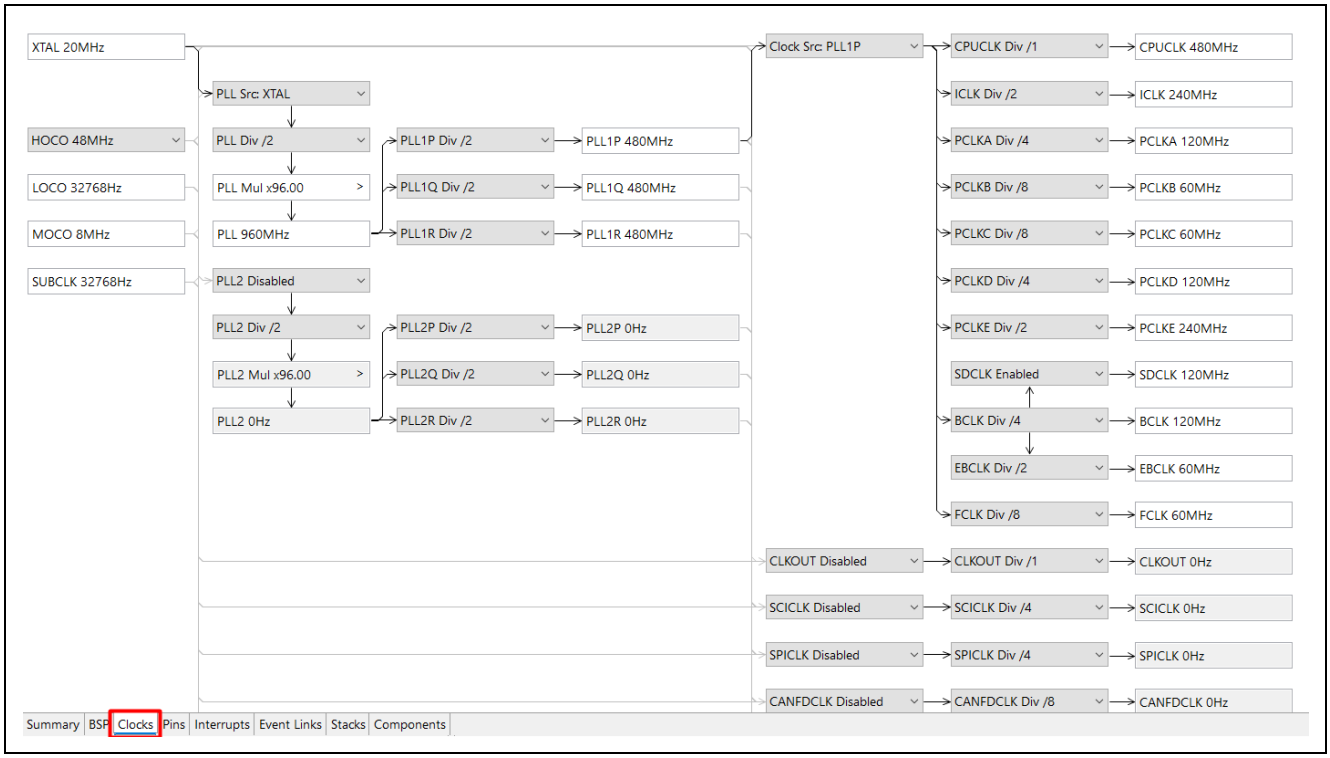

Figure 2. RA8 Clock configurations using FSP configurator

Clock gating by controlling the unused clock can also contribute to additional power savings. The EBCLK and SDCLK can be controlled when they are not in operation. FSP configurator provides an option to disable the clock to the peripherals which is not in use. Clock Gating can also be implemented by setting the Module-stop function to which can stop the clock supply to RA8 peripherals such as SRAM, DMA, DTC and ELC, POEG, GPT, ULPT, AGT, ETHERC, EDMAC, USB, SCI, CAN etc.

Module Stop Control Registers A-E (MSTPCRA, MSTPCRB, MSTPCRC, MSTPCRD, MSTPCRE) provides a control to configure the individual peripherals to implement Module-stop function. The details for individual modules can be found in the section 10 of the RA8M1 User's Manual.

Processor Low power modes : RA8 MCU provides Processor Low power modes by taking the CPU to Sleep mode and Deep Sleep mode.

When a WFI instruction is executed while SCR.SLEEPDEEP bit is 0, the MCU enters CPU Sleep mode. In this mode, the CPU stops operating but the contents of its internal registers are retained. Other peripheral functions do not stop. Available resets or interrupts in CPU Sleep mode can cause the MCU to cancel CPU Sleep mode. All interrupt sources are available, if using an interrupt to cancel CPU Sleep mode, you must set the associated IELSRn register before executing a WFI instruction.

When a WFI instruction is executed while SCR.SLEEPDEEP is set as 1, the MCU enters CPU Deep Sleep mode. In this mode, the CPU stops operating but the contents of its internal registers are retained. Other peripheral functions do not stop. However, accessing to Tightly coupled Memory (TCM) is not available and SysTick also stops in this mode. Available resets or wakeup enabled interrupts in CPU Deep Sleep mode can cause the MCU to cancel CPU Deep Sleep mode. User must also set WUPEN0 and WUPEN1 registers located in ICU to enable wakeup from CPU Deep Sleep mode. For details, refer to the section 13, Interrupt Controller Unit (ICU) in the RA8M1 User's Manual.

Item	CPU Sleep mode	CPU Deep Sleep mode
Transition condition	WFI instruction after set CPU0.SCR. SLEEPDEEP = 0.	WFI instruction after set CPU0.SCR. SLEEPDEEP = 1
Canceling method	All interrupts. Any reset available in the mode.	Interrupts shown in Table 10.4 Any reset available in the mode.
State after cancellation by an interrupt	Program execution state (interrupt processing)	Program execution state (interrupt processing)
State after cancellation by a reset	Reset state	Reset state
CPU	Stop (Retained)	Stop (Retained)
DMA Controller (DMAC)	Selectable	Selectable
Data Transfer Controller (DTC)	Selectable	Selectable
Watchdog Timer (WDT)	Selectable*1	Selectable*1
Independent Watchdog Timer (IWDT)	Selectable*1	Selectable*1
ARM Debug function	Stop*2	Stop*2
Trace function	Stop*3	Stop*3
Other peripheral modules	—	—

Figure 3. RA8M1 Operating state of Processor Low Power Mode

Software Standby Mode: When a WFI instruction is executed while Low Power State Control Register (LPSCR.LPMD) bit is 4 and CPU0.SCR.SLEEPDEEP bit is 1, the MCU enters Software Standby mode depending on LPSCR.LPMD setting. In this mode, the CPU, most of the on-chip peripheral functions and most of the oscillators stops.

However, the contents of the CPU internal registers and SRAM data, the states of on-chip peripheral functions and the I/O ports are retained. Software Standby mode allows significant reduction in power consumption because most of the oscillators stops in this mode.

The state of the oscillators depends on the setting of the control register for each oscillator. For details, refer to the chapter of clock generation circuit. The status of address bus and bus control signals in Software Standby mode can be selected by SBYCR.OPE bit. The SBYCR.OPE bit specifies whether to set to the high-impedance state or to retain the output of the address bus and bus control signals in Software Standby mode. Clear DMAST.DMST bit and DTCST.DTCST bit to 0 before executing WFI instruction.

Example of Software Standby Mode Application

Figure 4 shows an example of entry to Software Standby mode on detection of a falling edge of the IRQn pin, and exit from Software Standby mode by a rising edge of the IRQn pin. In this example, an IRQn pin interrupt is accepted with the IRQCRi.IRQMD[1:0] (i = 0 to 15) bits of the ICU set to 00b (falling edge) in Normal mode, and the IRQCRi.IRQMD[1:0] bits are set to 01b (rising edge). Then, if MOCO is not operated, it must set MOCOCR.MCSTP to 0 (MOCO is operated). After that, the LPSCR.LPMD bit is set to 0x4 and CPU.SCR.SLEEPDEEP bit is set to 1, and then a WFI instruction is executed. As a result, entry to Software Standby mode completes and exit from Software Standby mode is initiated by a rising edge of the IRQn pin.

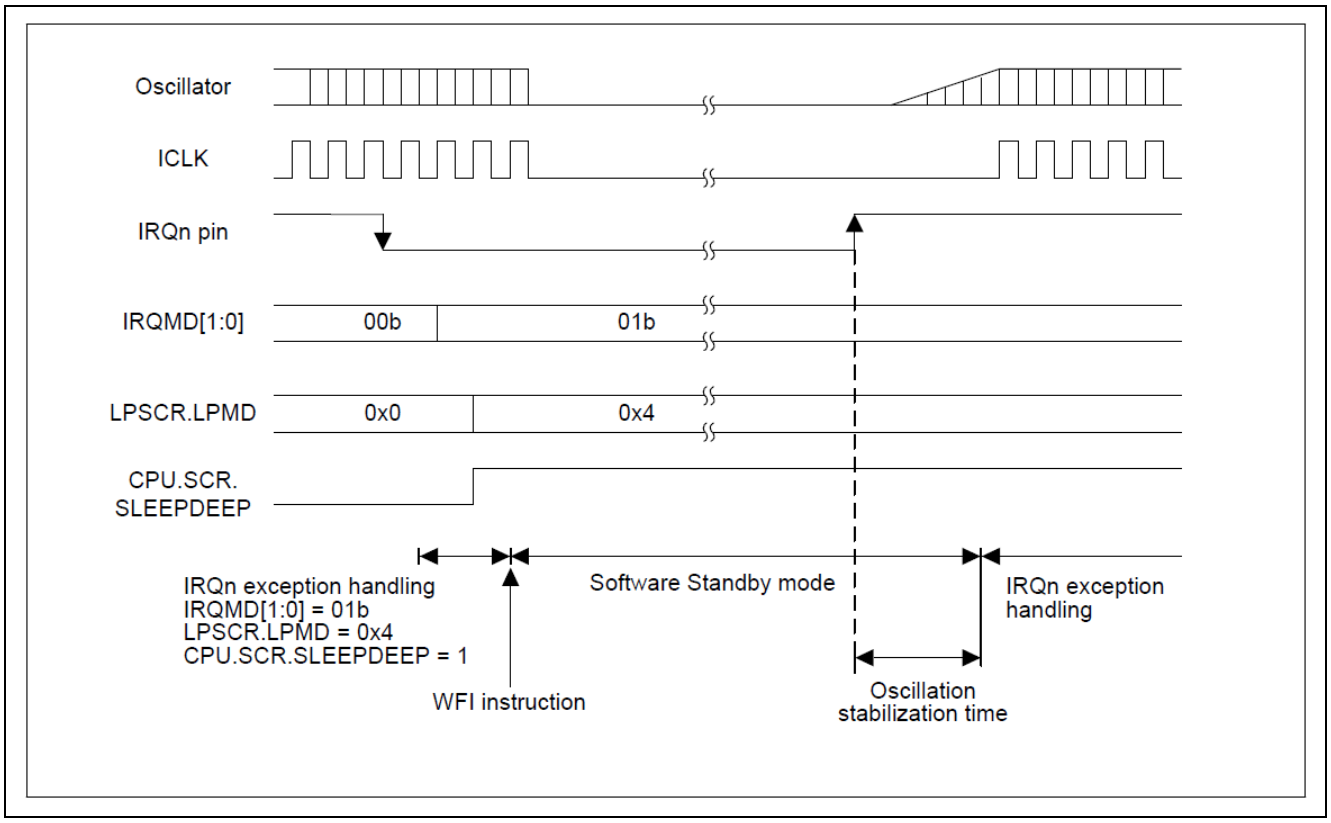

Figure 4. RA8M1 Software Standby Example

Deep Software Standby Mode: When a WFI instruction is executed while LPSCR.LPMD bit is 0x8, 0x9, 0xA and CPU.SCR.SLEEPDEEP bit is 1, the MCU enters Deep Software Standby mode 1 or 2 or 3 depending on LPSCR.LPMD setting. It must be set MOCOCR.MCSTP to 0 (MOCO is operated) before executed the WFI instruction to entry this mode. Figure 5 shows the setting of the related control bits. In this mode, the CPU, on-chip peripheral functions, SRAM, and oscillators are almost stopped as shown in Figure 6. Furthermore, since the internal power supply to these inactive modules is stopped, power consumption is remarkably reduced. The contents of all the registers of the CPU and specific internal peripheral modules become undefined. Figure 6 shows the status of each on-chip peripheral functions and oscillators.

Data in the standby SRAM are preserved in Deep Software Standby mode 1. When OFS1.PVDAS = 0 and OFS1.PVDLPSEL = 0, the low-power-consumption function of PVD0 is enabled, so power consumption is further reduced. In Deep Software Standby mode 2, the internal supply of power to the standby SRAM, the USB resume detecting unit and IWDT are cut off, reducing power consumption. Data in the standby SRAM become undefined at this time. When OFS1.PVDAS = 0 and OFS1.PVDLPSEL = 0, the low-power-consumption function of PVD0 is enabled, so power consumption is further reduced. In Deep Software Standby mode 3, the PVD is also stopped.

		Setting mode of LPSCR.LPMD bit and CPU.SCR.SLEEPDEEP bit			
		Software Standby	**Deep Software Standby 1**	**Deep Software Standby 2**	**Deep Software Standby 3**
		(LPSCR.LPMD = 0x4, CPU0.SCR. SLEEPDEEP = 1)	**(LPSCR.LPMD = 0x8, CPU0.SCR. SLEEPDEEP = 1)**	**(LPSCR.LPMD = 0x9, CPU0.SCR. SLEEPDEEP = 1)**	**(LPSCR.LPMD = 0xA, CPU0.SCR. SLEEPDEEP = 1)**
FENTRYR.FENTRYC FENTRYR.FENTRYD	0	Software Standby	Deep Software Standby 1	Deep Software Standby 2	Deep Software Standby 3
	1	CPU Power gating mode and Keep System Active	CPU Power gating mode and Keep System Active	CPU Power gating mode and Keep System Active	CPU Power gating mode and Keep System Active
OFS0.IWDTSTPCTL (auto-start mode) IWDTCSTPR.SLCSTP (register start mode)	0	Software Standby	Deep Software Standby 1	Deep Software Standby 1	Deep Software Standby 1
	1			Deep Software Standby 2	Deep Software Standby 3
PVD1CR0.RI PVD2CR0.RI	0	Software Standby	Deep Software Standby 1	Deep Software Standby 2	Deep Software Standby 3
	1			Deep Software Standby 1	Deep Software Standby 1

Figure 5. RA8 Software Standby and Deep Software Standby Mode setting

Item	Software Standby Mode(SSTBY)	Deep Software Standby mode(DSTBY)		
	SSTBY	DSTBY1	DSTBY2	DSTBY3
Transition condition	WFI instruction after set LPSCR and CPU0.SCR.SLEEPDEEP=1	WFI instruction after set LPSCR and CPU0.SCR.SLEEPDEEP=1.		
Canceling method	Interrupts shown in Table 10.4 Any reset available in the mode	Interrupts shown in Table 10.4 Any reset available in the mode		
State after cancellation by an interrupt	Program execution state (interrupt processing)	Reset state		
State after cancellation by a reset	Reset state	Reset state		
Main clock oscillator	Selectable*10	Stop		
Sub-clock oscillator	Selectable	Selectable		
High-speed on-chip oscillator	Selectable*11	Stop		
Middle-speed on-chip oscillator	Stop*19	Stop		
Low-speed on-chip oscillator	Selectable*2	Selectable*2	Stop	
PLL1	Stop	Stop		
PLL2	Stop	Stop		
Oscillation stop detection function	Selectable*12	Stop		
Clock/buzzer output function	Selectable*3	Stop (Undefined)		
External Bus (EBCLK)	Stop (Retained)	Stop (Retained)		
CPU	Stop (Retained)	Stop (Undefined)		
TCM (SRAM)	Stop (Retained)*15	Stop (Undefined)		
User SRAM	Stop (Retained)*13	Stop (Undefined)		
Standby SRAM	Stop (Retained)*14	Stop (Retained)*14	Stop (Undefined)	
Backup register	Stop (Retained)	Stop (Retained)		
Flash memory	Stop (Retained)	Stop (Retained)		
Memory Protection Unit (MPU)	Stop (Retained)	Stop (Undefined)		
DMA Controller (DMAC)	Stop (Retained)	Stop (Undefined)		
Data Transfer Controller (DTC)	Stop (Retained)	Stop (Undefined)		
Watchdog Timer (WDT)	Stop (Retained)	Stop (Undefined)		
Independent Watchdog Timer (IWDT)	Selectable*1	Selectable*1	Stop (Undefined)	
ARM Debug function	Stop*16	Stop*16		
Trace function	Stop*17	Stop*17		
Clock Frequency Accuracy Measurement Circuit (CAC)	Stop (Undefined)	Stop (Undefined)		
Ethernet MAC Controller (ETHERC)	Stop (Undefined)	Stop (Undefined)		
Ethernet DMA Controller (EDMAC)	Stop (Undefined)	Stop (Undefined)		
USB 2.0 Full-Speed Module (USBFS)	Stop (Retained) Detection of USB resumption is possible.	Stop (Retained) Detection of USB resumption is possible.	Stop (Undefined)	

Figure 6. RA8 Operating state in Software Standby and Deep Software Standby Mode

Item	Software Standby Mode(SSTBY)	Deep Software Standby mode(DSTBY)		
	SSTBY	DSTBY1	DSTBY2	DSTBY3
USB 2.0 High-Speed Module (USBHS)	Stop (Retained) Detection of USB resumption is possible.	Stop (Retained) Detection of USB resumption is possible.	Stop (Undefined)	
Realtime clock (RTC)	Selectable	Selectable	Selectable*4	
CAN-FD	Stop (Undefined)	Stop (Undefined)		
CANFD ECC (CNECC)	Stop (Undefined)	Stop (Undefined)		
Serial Peripheral Interface (SPI0)	Stop (Retained)	Stop (Undefined)		
Serial Peripheral Interface (SPI1)	Stop (Undefined)	Stop (Undefined)		
Octa Serial Peripheral Interface (OSPI)	Stop (Retained)	Stop (Undefined)		
Decryption On The Fly (DOTF)	Stop (Retained)	Stop (Undefined)		
Serial Sound Interface Enhanced (SSIE0)	Stop (Retained)	Stop (Undefined)		
Serial Sound Interface Enhanced (SSIE1)	Stop (Undefined)	Stop (Undefined)		
SD/MMC Host Interface (SDHI0)	Stop (Retained)	Stop (Undefined)		
SD/MMC Host Interface (SDHI1)	Stop (Undefined)	Stop (Undefined)		
Cyclic Redundancy Check (CRC) Calculator	Stop (Undefined)	Stop (Undefined)		
Port Output Enable for GPT (POEG)	Stop (Undefined)	Stop (Undefined)		
General PWM Timer (GPT)	Stop (Undefined)	Stop (Undefined)		
Ultra low power Timer (ULPTn, n = 0, 1)	Selectable	Selectable	Stop (Undefined)	
Asynchronous General Purpose Timer (AGTn, n = 0, 1)	Selectable*5	Stop (Undefined)		
12-Bit A/D Converter (ADC12)	Stop (Undefined)	Stop (Undefined)		
12-Bit D/A Converter (DAC12)	Stop (Retained)	Stop (Undefined)		
Data Operation Circuit (DOC)	Stop (Undefined)	Stop (Undefined)		
Serial Communications Interface (SCI0)	Stop (Retained)	Stop (Undefined)		
Serial Communications Interface (SCIn, n = 1 to 4, 9)	Stop (Undefined)	Stop (Undefined)		
I2C Bus Interface (IIC0)	Selectable*6	Stop (Undefined)		
I2C Bus Interface (IIC1)	Stop (Undefined)	Stop (Undefined)		
I3C Bus Interface (I3C)	Selectable	Stop (Undefined)		
Event Link Controller (ELC)	Stop (Undefined)	Stop (Undefined)		
Renesas Secure IP (RSIP-E51A)	Stop (Retained)	Stop (Undefined)		
Capture Engine Unit(CEU)	Stop (Undefined)	Stop (Undefined)		
Temperature Sensor (TSN)	Stop (Undefined)	Stop (Undefined)		
High-Speed Analog Comparator 0 (ACMPHS0)	Selectable	Stop (Undefined)		
High-Speed Analog Comparator 1 (ACMPHS1)	Selectable *20	Stop (Undefined)		
IRQn (n = 0 to 15) pin interrupt	Selectable	Stop (Undefined)		
NMI, IRQn-DS (n = 0 to 15) pin interrupt	Selectable	Selectable		
Programable Voltage Detect (PVD)	Selectable	Selectable*18	Selectable*18	Stop (Undefined)*7
VBATT_R voltage drop detection	Selectable	Selectable	Selectable	Stop (Undefined)
Power-on reset circuit	Operating	Operating	Operating	Operating*8
I/O Ports	Retained*9	Retained*9		

Figure 7. RA8 Operating state in Software Standby and Deep Software Standby Mode

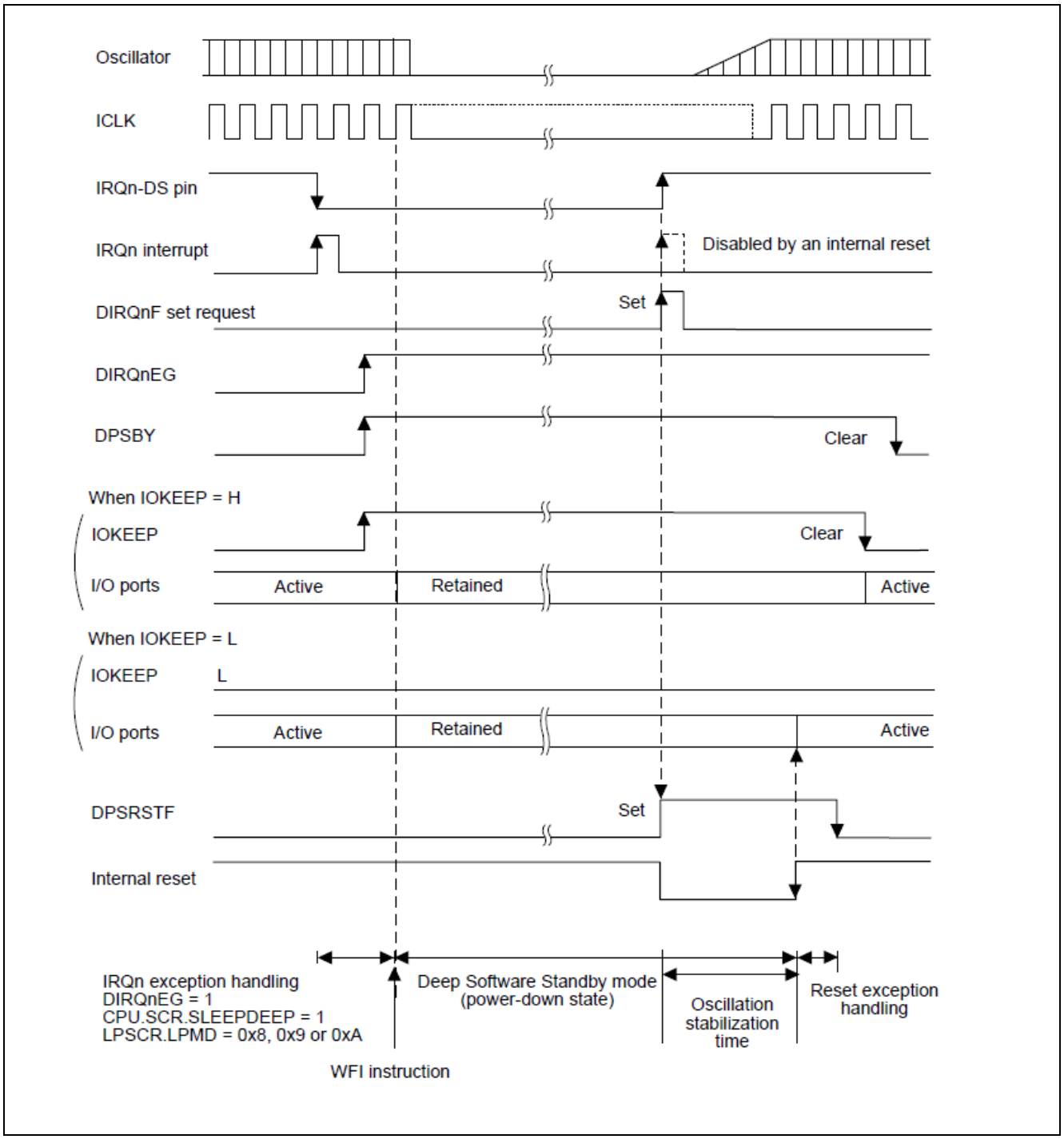

Figure 8. RA8M1 Deep Software Standby Entry and Exit Example

Figure 8 illustrates an instance where the transition to Deep Software Standby mode occurs at the falling edge of the IRQn-DS pin, and exiting Deep Software Standby mode happens at the rising edge of the IRQn-DS pin. In this scenario, an IRQn interrupt is acknowledged with the IRQCRi.IRQMD[1:0] bits of the ICU set to 00b (falling edge). If MOCO is not active, MOCOCR.MCSTP must be set to 0 (MOCO is active).

Then, after the DPSIEGRy.DIRQnEG (y = 0 or 1, n = 0 to 15) bit is set to 1 (rising edge) and the LPSCR.LPMD bit is set to 0x8 or 0x9 or 0xA and CPU.SCR.SLEEPDEEP bit is set to 1, the WFI instruction is executed. Thus, a transition to Deep Software Standby mode is made. After that, exiting Deep Software Standby mode is made at the rising edge of the IRQn-DS pin.

Power Control Mode

Power consumption can be reduced in Normal and Processor low power mode by selecting an appropriate operating power control mode according to the operating frequency.

Two operating power control modes are available in RA8M1 are 1) High-speed mode 2) Low-speed mode. This can be selected using the OPCCR:OPCM[1:0] register is used to reduce power consumption in Normal and Processor low power mode by specifying a lower operating frequency.

Mode	Oscillator					
	PLL1, PLL2	High-speed on-chip oscillator	Middle-speed on-chip oscillator	Low-speed on-chip oscillator	Main clock oscillator	Sub-clock oscillator
High-speed	Available	Available	Available	Available	Available	Available
Low-speed	Not Available	Available	Available	Available	Available	Available

Figure 9. RA8M1 Available Oscillator in Power Control Mode

Procedure to change the operating power control modes.

Table 1. Operating Power Control Modes

From a higher power to a lower power mode	From a lower power to a higher power mode
• Change the oscillator to what is used in Low-speed mode. Set the frequency of each clock lower than or equal to the maximum operating frequency in Low-speed mode.	• Confirm that the OPCCR.OPCMTSF flag is 0 (indicates transition completed).
• Turn off the oscillator that is not required in Low-speed mode.	• Set the OPCCR.OPCM[1:0] bits to 00b (High-speed mode).
• Confirm that the OPCCR.OPCMTSF flag is 0 (indicates transition completed).	• Confirm that the OPCCR.OPCMTSF flag is 0 (indicates transition completed).
• Set the OPCCR.OPCM[1:0] bits to 11b (Low-speed mode).	• Turn on any required oscillator in High-speed mode.
• Confirm that OPCCR.OPCMTSF flag is 0 (indicates transition completed).	• Set the frequency of each clock lower than or equal to the maximum operating frequency for High-speed mode.

3.2 RA8 Clocks and Oscillators

One of the primary contributors to power consumption in MCUs is the utilization of various clock sources. The power consumed by an MCU is directly correlated with the number of clock sources employed and their operational frequencies. These diverse clock sources cater to the varying speed requirements of components within the MCU. Opting for the lowest feasible frequency when employing these clock sources can yield maximal power savings, albeit at the expense of processing speed. The operational speed of the MCU is contingent upon system demands, particularly interrupt service duration and data transfer rates. The RA8 MCU offers a range of clock sources for system use. Depending on the specific system requisites, clock sources can be dynamically selected and adjusted using the application, thereby optimizing power consumption.

In the past, system clocks were driven by oscillator circuits using external ceramic or crystal resonators. Nowadays, MCUs feature multiple internal oscillators with varying levels of accuracy. Internal oscillator accuracy typically ranges from 1% at 25ºC to 3% to 5% over temperature. When higher accuracy is required by the application, the external oscillator circuit can be activated in software. Conversely, when accuracy isn't critical, the MCU can switch to its internal oscillator, resulting in a slower clock speed and power savings.

RA8 provides the following Clock sources for the internal clock signals

- Main clock oscillator
- Sub-clock oscillator
- HOCO clock
- MOCO clock
- LOCO clock
- PLL1 clock (PLL1P, PLL1Q and PLL1R)
- PLL2 clock (PLL2P, PLL2Q and PLL2R)
- External clock input for JTAG
- External clock input for SWD

Different components within an MCU may have vastly different speed requirements. For instance, while the core processor might demand high-speed operation for rapid computation and real-time processing, certain peripherals such as timers or watchdog timers might function perfectly well at lower clock frequencies. By providing a range of clock sources, MCUs allow developers to tailor performance to the specific needs of each component. From the above clock sources, the internal clock is produced for the internal peripheral components.

Table 2. Clock Source and its Components

Operating clock for the Components	Internal Clock Name
CPU	CPUCLK
DMAC, DTC, Flash, SRAM, System Bus , I/O Port and ICU	System clock (ICLK)
Debug Subsystem	Debug clock (DCLK)
Peripheral modules	Peripheral module clocks (PCLKA, PCLKB, PCLKC, PCLKD, and PCLKE)
Flash Interface	Flash interface clock (FCLK)
External bus controller and external pin output	External bus clock (BCLK, EBCLK)
External bus controller and external pin output for the SDRAM	SDRAM clock (SDCLK)
Trace function and external pin output	Trace clock (TRCLK)
SCI	SCI clock (SCICLK)
SPI	SPI clock (SPICLK)
Octal SPI	Octal-SPI clock (OCTACLK, OCTADIVCLK)
CANFD Core	CANFD Core clock (CANFDCLK)
USBFS and USBHS	USB clock (USBCLK)
I3C	I3C clock (I3CCLK)
CAN	CAN clock (CANMCLK)
ULPT	ULPT LOCO clock (ULPTLCLK) and ULPT sub-clock (ULPTSCLK)
AGT	AGT LOCO clock (AGTLCLK)
CAC	CAC clock (CACCLK)
RTC	RTC LOCO clock (RTCLCLK
IWDT	IWDT clock (IWDTCLK)
SysTick Timer	SysTick timer clock (SYSTICKCLK)
JTAG	JTAG clock (JTAGTCK)

The internal clocks mentioned in the Table 2 serve various functions within the RA8 MCU. To optimize power usage and tailor performance according to the system's requirements, it's essential to control these clocks effectively.

By enabling or activating specific internal clocks only when they are needed and disabling them when they are not in use, developers can effectively manage power consumption. For example, clocks associated with peripherals that are inactive during certain periods can be disabled to conserve power during idle states. This dynamic control over clock sources allows for efficient utilization of resources and minimizes unnecessary power consumption.

In pursuit of power savings within a RA8 MCU, one effective strategy is to scale down the frequencies of internal clocks. This can be achieved through the utilization of available clock dividers, which allow developers to reduce the frequency of clock signals supplied to various peripherals and components within the MCU.

By scaling down the clocks, peripherals can operate at slower speeds while still maintaining functionality. This approach facilitates power savings without compromising the overall availability and required performance of the system. For instance, peripherals such as timers, communication modules, and sensor interfaces may not always require high-speed clock signals for their operations. By reducing their clock frequencies, these peripherals can operate efficiently while consuming less power.

Furthermore, slower clock sources can be particularly beneficial when the MCU enters lower power states, such as low-power or sleep modes. During these states, the MCU can switch to using slower clock sources, further reducing power consumption while still maintaining basic functionality. This allows the MCU to remain operational in a reduced-power state, ready to quickly respond to wake-up events or resume normal operation when required.

Importantly, the ability to dynamically switch between slower and higher clock speeds based on the operational mode of the MCU ensures optimal power efficiency across different usage scenarios. For example, when the MCU transitions from a low-power sleep mode to normal operation, it can seamlessly switch back to higher clock frequencies to meet performance demands.

RA FSP also provides a API level functions to change the Clock and its settings, these can be accessed using “RA_Clock/ra/fsp/src/r_cgc/r_cgc.c”

```
fsp_err_t R_CGC_ClocksCfg(cgc_ctrl_t * const p_ctrl, cgc_clocks_cfg_t const * const p_clock_cfg);
fsp_err_t R_CGC_ClockStart(cgc_ctrl_t * const p_ctrl, cgc_clock_t clock_source, cgc_pll_cfg_t const * const p_pll_cfg);
fsp_err_t R_CGC_ClockStop(cgc_ctrl_t * const p_ctrl, cgc_clock_t clock_source);
fsp_err_t R_CGC_ClockCheck(cgc_ctrl_t * const p_ctrl, cgc_clock_t clock_source);
fsp_err_t R_CGC_SystemClockSet(cgc_ctrl_t * const              p_ctrl,
                               cgc_clock_t                     clock_source,
                               cgc_divider_cfg_t const * const p_divider_cfg);
fsp_err_t R_CGC_SystemClockGet(cgc_ctrl_t * const        p_ctrl,
                               cgc_clock_t * const       p_clock_source,
                               cgc_divider_cfg_t * const p_divider_cfg);
fsp_err_t R_CGC_OscStopDetectEnable(cgc_ctrl_t * const p_ctrl);
fsp_err_t R_CGC_OscStopDetectDisable(cgc_ctrl_t * const p_ctrl);
fsp_err_t R_CGC_OscStopStatusClear(cgc_ctrl_t * const p_ctrl);
fsp_err_t R_CGC_CallbackSet(cgc_ctrl_t * const          p_api_ctrl,
                            void (                    * p_callback)(cgc_callback_args_t *),
                            void const * const          p_context,
                            cgc_callback_args_t * const p_callback_memory);
```

Figure 10. RA8 MCU Clock Configuration using FSP API

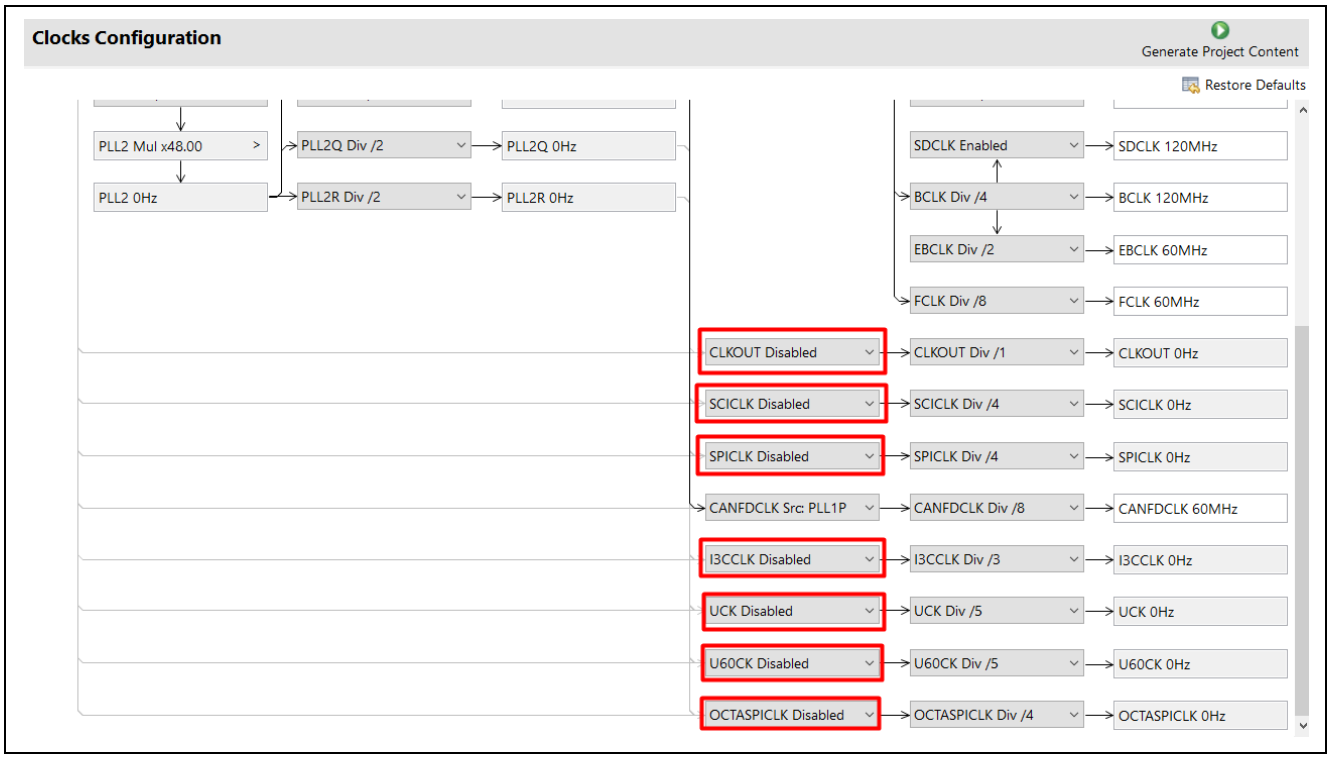

Figure 11. RA8 MCU Clock Configuration to Enable/Disable individual clock

Renesas Flexible Software Package (FSP) enhances this control by providing a configurator tool. This tool enables developers to selectively configure the performance and power-saving features associated with each internal clock. With the configurator, developers can fine-tune parameters such as clock frequencies, clock gating strategies, and power-saving modes to optimize both performance and power consumption according to the specific requirements of their application.

In RA8 MCU the frequency division ratio can be selected independently for the CPU clock (CPUCLK), system clock (ICLK), peripheral module clocks (PCLKA, PCLKB, PCLKC, PCLKD, PCLKE), external bus clock (BCLK), and flash interface clock (FCLK). Clocks Tab in the FSP configurator can be used to set up the settings for the Clock selection and its division. Also the clock settings can be done dynamically using the application code as well.

Output of clock signal can be controlled by disabling them to conserve the power. For instance EBCLK and SDCLK output control can be selected to enable or disable depending on the usage.

In summary, effective management of internal clocks, coupled with the configurability provided by Renesas FSP, empowers developers to strike a balance between performance and power efficiency in MCU-based systems. This ensures that the system operates efficiently while minimizing energy consumption, ultimately extending battery life and enhancing overall system reliability.

For more details on clock configurations and its generation, please refer to the section 8 (Clock Generation Circuit) of the RA8M1 User's Manual.

3.3 Designing Power Efficient System with RA8 Power Domains

When designing a power-efficient system with MCUs, selecting the power supply is crucial. Power domains are distinct sections of a circuit that can be powered on or off independently. This allows for selective power management, conserving energy by shutting down parts of the system when they're not in use. Here's how you can consider power domains while designing a power-efficient system with RA8 MCU.

RA8 MCU provides operating voltage range from 1.65V to 3.6V. System designers take advantage of the Low voltage designs to achieve power saving.

RA8 MCU also provides two power supply for providing core voltage (VDD):

- Switching regulator (DCDC)
- External power supply from external power device (External VDD)

With the inbuilt independent power supply for the CPU core features an integrated voltage regulator capable of supplying the core voltage. Alternatively, external provision of core voltage is feasible by deactivating the internal DCDC regulator. There are instances where utilizing an external source for the core proves more power-efficient. This is attributed to the internal regulator powering the core at nominal voltage, enabling full-speed operation. However, if the application doesn't necessitate full-speed operation, employing a lower voltage for core power proves advantageous. Additionally, disabling the internal regulator also deactivates LVD circuits, resulting in power savings.

Note: Using Inbuilt DCDC regulator it takes 150us to stabilize when coming out of Low Power modes compared to 10usec using External power supply.

Lowering the device's operating voltage, Vdd, is an effective measure to minimize overall power consumption. During active operation, power usage is primarily determined by the clock speed, whereas during sleep states, transistor leakage becomes the dominant factor. Operating at lower voltages necessitates less charge for switching system clocks and results in reduced transistor leakage.

It's crucial to consider the trade-off between reducing operating voltage and its impact on the maximum attainable operating frequency. Determine the optimal voltage level that permits the application to operate at its highest speed. Consult the device datasheet to ascertain the maximum operating frequency achievable at the specified voltage.

3.4 Usage of RA8 Peripherals and its Power Management Techniques

3.4.1 DMA Controller (DMAC)

RA8 DMA enhances power efficiency by reducing CPU utilization, accelerating data transfers, optimizing peripheral usage, and enabling flexible system design. These benefits collectively contribute to minimizing power consumption and maximizing energy efficiency. RA8 MCU's DMA decreases CPU utilization. This allows the CPU to enter low-power modes or sleep states more frequently, effectively reducing overall power consumption by the MCU.

Power efficiency be achieved using Module stop function, when DMA is not in use, this disables the power to the DMA engine providing maximum power savings.

Before entering the module-stop state, Software Standby mode or Deep Software Standby mode, you must first set the DMAST.DMST bit to 0 (the DMAC module suspended) and use the settings in the sections that follow.

Software Standby mode and Deep Software Standby mode

If DMA transfer operations are in progress when the WFI instruction is executed, the DMA transfer completes before the transition to Software Standby mode or Deep Software Standby mode.

Module-stop function

Writing 1 to the MSTPCRA.MSTPA22 bit enables the module-stop function of the DMAC. If a DMA transfer is in progress when 1 is written to the MSTPA22 bit, the transition to the module-stop state proceeds after the DMA transfer ends. Access to the DMAC registers is prohibited while the MSTPA22 bit is 1. Writing 0 to the MSTPA22 bit releases the DMAC from the module-stop state.

3.4.2 Data Transfer Controller (DTC)

RA8 DTC enhances power efficiency by reducing CPU utilization, accelerating data transfers, optimizing peripheral usage, and enabling flexible system design. These benefits collectively contribute to minimizing power consumption and maximizing energy efficiency. RA8 MCUs DTC decreases CPU utilization. This allows the CPU to enter low-power modes or sleep states more frequently, effectively reducing overall power consumption by the MCU.

Power efficiency can be achieved using Module stop function, when DTC is not in use, this disables the power to the DTC engine providing maximum power savings.

Before transitioning to the module-stop state, Software Standby mode or Deep Software Standby mode, set the DTCST.DTCST bit to 0, and then perform the operations described in the following sections.

(1) Module-stop function: Writing 1 to the MSTPCRA.MSTPA22 bit enables the module-stop function of the DTC. If a DTC transfer is in progress. when 1 is written to the MSTPCRA.MSTPA22 bit, the transition to the module-stop state proceeds after the DTC transfer ends. While the MSTPCRA.MSTPA22 bit is 1, accessing the DTC registers is prohibited. Writing 0 to the MSTPCRA.MSTPA22 bit releases the DTC from the module-stop state.

(2) Software Standby mode and Deep Software Standby mode : If DTC transfer operations are in progress when the WFI instruction is executed, the transition to Software Standby mode or Deep Software Standby mode follows the completion of the DTC transfer.

3.4.3 Event Link Controller (ELC)

The Event Link Controller (ELC) can be used for event requests generated by various peripheral modules as source signals to connect them to different modules, allowing direct link between the modules without CPU intervention. In CPU Sleep mode, it can be used to run the system where CPU intervention to transact the data is not needed.

ELC module can be stopped using the Module-stop function, Module-stop state can be set in Module Control Register C (MSTPCRC) to reduce power consumption.

Power conservation when using the ELC module can be accomplished through the utilization of Software Standby Mode (SSTBY) and Deep Software Standby Mode (DSTBY). In SSTBY and DSTBY mode, the ELC module halts while register contents become undefined.

3.4.4 Port Output Enable for GPT (POEG)

POEG module can be stopped using the Module-stop function, Module-stop state can be set in Module Control Register D (MSTPCRD) to reduce power consumption.

When using the POEG, do not invoke Software Standby mode. In this mode, the POEG stops and therefore output disable of the pins cannot be controlled.

3.4.5 General PWM Timer (GPT)

GPT module can be stopped using the Module-stop function, Module-stop state can be set in Module Control Register D (MSTPCRD) to reduce power consumption.

Power conservation when using the GPT module can be accomplished through the utilization of Software Standby Mode (SSTBY) and Deep Software Standby Mode (DSTBY). In SSTBY and DSTBY mode, the GPT module stops while register contents are not retained. Users are required to backup the registers in standby memory during the low power modes.

3.4.6 Ultra-Low-Power Timer (ULPT)

This ultra-low-power timer (ULPT) as name suggests is power efficient and can be selected to operate in the Low Power mode. It can also be configured to operate during the Low power modes. The register contents are retained during the Low Power modes.

The ULPT can operate in Software Standby mode or Deep Software Standby mode1. Set it to each Standby mode with count operation start (ULPTCR.TSTART = 1, ULPTCR.TCSTF = 1)

ULPT module can be stopped using the Module-stop function, Module-stop state can be set in Module Control Register D (MSTPCRD) to reduce power consumption.

3.4.7 Low Power Asynchronous General Purpose Timer (AGT)

AGT can be selected to operate in the Software standby mode. It can also be configured to stop during the Software standby mode. The register contents are retained during the Software standby mode. Register contents are not retained in deep software standby mode.

AGTMR2 : AGT Mode Register 2, bit 7 sets the low power operation, which impacts access to certain AGT registers. Set this bit to 1 to operate in low power. When this bit is 1, access to the some of the registers is prohibited. For more details refer to the AGTMR2 register settings in the User's Hardware manual

AGT operation can be disabled or enabled using Module Stop Control Register D (MSTPCRD). The AGT module is initially stopped after reset. Releasing the module-stop state enables access to the registers.

3.4.8 Realtime Clock (RTC)

This RTC can be selected to operate in the Low Power mode. It can also be configured to stop during the Low power modes. The register contents are retained during the Low Power modes and using battery backup.

A transition to a low power state (Software Standby mode, Deep Software Standby mode, or battery backup state) during a write to an RTC register might corrupt the value of the register. After setting the register, confirm that the setting is in place before initiating a transition to a low power state.

3.4.9 Ethernet MAC Controller (ETHERC)

Power conservation when using the ETHERC module can be accomplished through the utilization of Software Standby Mode (SSTBY) and Deep Software Standby Mode (DSTBY). In SSTBY and DSTBY mode, the ETHERC module stops while register contents are not retained.

ETHERC module can be stopped using the Module-stop function, Module-stop state can be set in Module Control Register B (MSTPCRB) to reduce power consumption.

The ETHERC supports Wake-on-LAN (WOL). WOL is a function to detect a Magic Packet transmitted from a host device or other device, and to wake the MCU from a low power mode such as Sleep. When the ETHERC detects a Magic Packet, it outputs high on the ET0_WOL pin. Write 1 to the EDMAC0.EDMR.SWR bit to drive the ET0_WOL pin low.

As an example, designers can take advantage of the WOL feature along with ELC to handover the data reception to DMA and once complete frame is received, MCU can wake up from the sleep mode and process the data.

3.4.10 Ethernet DMA Controller (EDMAC)

Power conservation when using the ETHERC module can be accomplished through the utilization of Software Standby Mode (SSTBY) and Deep Software Standby Mode (DSTBY). In SSTBY and DSTBY mode, the ETHERC module stops while register contents are not retained.

Module Stop Control Register B (MSTPCRB:15) enables or disables EDMAC module operation. The modules are initially stopped after reset. Releasing the module-stop state enables access to the registers.

Stopping the EDMAC during Operation, when stopping EDMAC operation by using a Software Standby mode or the module-stop function while the EDMAC is running, confirm that the EDTRR.TR and EDRRR.RR bits are 0. If the EDMAC is stopped while the EDTRR.TR or EDRRR.RR bit is 1, data for the frame that is being transmitted or received might not be complete, and EDMAC operation after exiting Software Standby mode or the module-stop state is not guaranteed.

3.4.11 Universal serial Bus (USB)

USBFS and USBHS operation can be disabled or enabled using Module Stop Control Register B (MSTPCRB). The USBFS is initially stopped after reset. Releasing the module-stop state enables access to the registers. For details, see section 10, Low Power Modes.

Deep Software Standby mode1 can be canceled by a USB suspend/resume interrupt. USB suspend/resume interrupts are detected by the USB resume detecting unit, which controls and monitors the USB I/O pins to detect the interrupts. Clearing the Interrupt Status Register on Canceling Software Standby Mode

Because the input buffer is always enabled in Software Standby mode, an unexpected interrupt might occur under the following conditions:

- When the interrupt is enabled in Normal mode
- When the interrupt is disabled in Software Standby mode
- When the input level of the pin that cancels software standby mode is changed in Software Standby mode

These conditions might cause the associated interrupt flag in the Interrupt Status Register to set unexpectedly. After the MCU cancels the Software Standby mode, the unexpected interrupt might be sent to the interrupt controller. To avoid this, always clear the INTSTS0 and INTSTS1 registers in the canceling sequence.

Setup for Transitioning to Deep Software Standby Mode 1

Before transitioning to Deep Software Standby mode 1, clear the DVSTCTR0.VBUSEN bit to 0.

3.4.12 I2C Bus Interface (IIC)

Power conservation when utilizing the I2C module can be accomplished through the utilization of Software Standby Mode (SSTBY) and Deep Software Standby Mode (DSTBY). In SSTBY mode, the IIC channel 0 can be configured to operate, whereas in DSTBY mode, Channel 0 stops and the register contents are not retained. In SSTBY and DSTBY mode IIC Channel 1, stops and register contents are not retained.

The Module Stop Control Register B (MSTPCRB) can enable or disable IIC operation. The IIC is initially stopped after reset. Releasing the module-stop state enables access to the registers.

The IIC0 provides a wakeup function that causes the MCU to transition from Software Standby mode to normal operation. The wakeup function generates a wakeup interrupt signal on a match of the slave address of the received data. This wakeup interrupt signal triggers the return to normal operation.

3.4.13 I3C Bus Interface (I3C)

Power conservation when utilizing the I3C module can be accomplished through the utilization of Software Standby Mode (SSTBY) and Deep Software Standby Mode (DSTBY). I3C can be configured to operate in the SSTBY mode, whereas in DSTBY mode, it stops and the register contents are not retained.

The Module Stop Control Register B (MSTPCRB) can enable or disable I3C operation. The I3C is initially stopped after reset. Releasing the module-stop state enables access to the registers.

Additionally The I3C Clock(I3CCLK), can be deactivated when the module is not in operation, thus conserving power.

The I3C provides a wakeup function that causes the MCU to transition from Software Standby mode to normal operation. The wakeup function generates a wakeup interrupt signal on a match of the slave address of the received data. This wakeup interrupt signal triggers the return to normal operation.

3.4.14 Serial Communications Interface (SCI)

Power conservation when utilizing the SCI module can be accomplished through the utilization of Software Standby Mode (SSTBY) and Deep Software Standby Mode (DSTBY). In SSTBY mode, the SCI channel 0 stops while retaining register contents, whereas in DSTBY mode, the register contents become undefined. For SCI Channel 1,2,3,4, and 9 its stops but register contents are undefined. Reception in standby mode is only available for SCI0.

Additionally, The SCI Clock, SCICLK, can be deactivated when the module is not in operation, thus conserving power.

The Module Stop Control Register B (MSTPCRB) can enable or disable SCI operation. The SCI is initially stopped after reset. Releasing the module-stop state enables access to the registers.

For Transmission, before using the power consumption reduction function to reduce SCI's power consumption, do the following to confirm transmission end (CSR.TEND = 1)

- Set the output pin state after transmission operation is stopped by CCR1.SPB2DT, SPB2IO.
- Stop the transmission (CCR0.TIE = 0, TE = 0, TEIE = 0)

When transitions to these states are made during transmission, the data being transmitted become indeterminate. To transmit data in the same operation mode after cancellation of the low power consumption state, set the TE bit to 1, read CSR, and write data to TDR sequentially to start data transmission.

To transmit data with a different operation mode, initialize the SCI first. To start transmission using the DMAC or DTC after cancellation from software standby mode, set the CCR0.TE and CCR0.TIE bit to at the same time. Then SCIn_TXI interrupt flag is generated, which causes the DMAC or DTC to write the transmit data, which starts transmission.

For reception, before specifying the module stop state or making a transition to software standby mode, stop the receive operations (CCR0.RE = 0). If transition is made during data reception, the data being received will be invalid.

3.4.15 Controller Area Network (CAN)

The CANFD Clock, CANFDCLK, can be deactivated when the module is not in operation, thus conserving power.

CANFD operation can be disabled or enabled using Module Stop Control Register C (MSTPCRC). The CANFD module is initially stopped after reset. Releasing the module-stop state enables access to the registers. CANFD also provides Power down function Module start stop function for each CAN node (Channel and Global Sleep mode).

After the release of a hardware reset or after setting and clearing a CFDGRSTC.SRST bit, the CANFD module automatically enters Global Sleep mode. The CANFD module also enters the Global Sleep mode when the Global Sleep Request bit is set while it is in Global Reset mode. This control bit cannot be set in Global Halt mode or Global Operation mode.

Setting the Global Sleep Request bit sets Channel Sleep Request bit and forces the channel into the Channel Sleep mode. Sleep mode is used for power saving purpose. When CANFD module is in Global Sleep mode, only the clock for CPU write access to the Global Sleep Mode Request bit is active. All other clocks are stopped and all other functions of the CANFD module are suspended. Read access from all registers is still possible and all register values are preserved.

After setting the Global Sleep Request bit, it is necessary to confirm that the Global Sleep status has been updated, indicating successful transition to Global Sleep mode before the Global Sleep Request bit can be cleared again.

Global Halt Mode is supported in the CANFD module, CANFD enters this mode when the Global Mode Control bit CFDGCTR.GMDC in the Global Control Register is configured for Global Halt mode while the CANFD module is in Global Reset mode. In this mode the channel in either Channel Reset or Channel Sleep mode remains in this mode

Global Mode Control bit CFDGCTR.GMDC in the Global Control Register is configured for Global Halt mode while the CANFD module is in Global Operation mode. In this mode if

- The channel in Channel Reset, Channel Halt, or Channel Sleep mode remains in this mode
- The channel in Channel Operation mode transitions to Channel Halt mode
- Global Halt Mode Status bit is set when the channel has left Channel Operation mode.

If a transmission or reception is ongoing for a channel, the transition to Channel Halt mode is delayed until completion of the communication. Similarly, if a channel is in bus-off, the full bus-off recovery sequence may be delayed depending on the channel configuration.

In Global Halt mode, all communications are suspended and CANFD logic does not cause any change to the Status and Flag registers (only when a channel is in the bus-off that its REC and TEC values are cleared). Additionally, the test mode configuration and control registers are not initialized in this mode. The Global Halt mode should be used to configure global module test modes.

3.4.16 Serial Peripheral Interface (SPI)

Power conservation when utilizing the SPI module can be accomplished through the utilization of Software Standby Mode (SSTBY) and Deep Software Standby Mode (DSTBY). In SSTBY mode, the SPI module halts while retaining register contents, whereas in DSTBY mode, the register contents become undefined.

The SPI Clock, SPICLK, can be deactivated when the module is not in operation, thus conserving power.

The activation or deactivation of SPI operation can be managed through the Module Stop Control Register B (MSTPCRB). Following a reset, the SPI module is initially halted. Releasing the module-stop state allows access to the registers.

Constraint on Low-Power Functions When using the module-stop function and entering a low-power mode other than CPU Sleep mode or CPU Deep Sleep mode, set the SPCR.SPE bit to 0 before completing communication.

3.4.17 Octal Serial Peripheral Interface (OSPI)

Power saving when using OSPI module can be achieved using the Software Standby Mode(SSTBY) and Deep Software standby Mode(DSTBY). In the SSTBY mode OSPI Module is stopped and the register contents are retained, whereas in the DSTBY mode the register contents are not retained.

OSPI Clock OCTASPICLK can be disabled when the module is not required to be operated and there by saving the power.

OSPI operation can also be disabled or enabled using Module Stop Control Register B (MSTPCRB:16). The OSPI module is initially stopped after reset. Releasing the module-stop state enables access to the registers

4. Factors to Consider When Designing a System with RA8 MCU

The RA8 MCU provides a battery backup function that maintains partial battery powering in the event of a power loss. Switching between VCC and VBATT, the battery-powered area includes RTC, SOSC, backup register, tamper detection and VBATT_R voltage drop detection. VBATT_R is the output voltage of the battery power supply switch. This is the power supply of the battery powered area. During normal operation, the battery-powered area is powered by the main power supply, the VCC pin. When a VCC voltage drop is detected, the power source switches to the dedicated battery backup power pin, the VBATT pin. When the voltage rises again, the power source switches back from VBATT to VCC.

To maximize power efficiency, the RA8 Series of MCUs allow on-chip peripherals to be stopped individually by writing to the Module Stop Control Registers (MSTPCRi, I = A, B, C, D, E). Once a module stops, access to the registers associated to the module is not possible. After a reset, most of the modules are placed in module-stop state, except for DMAC, DTC, and SRAM. See Hardware User's 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 stopped by writing a '1' to the proper bit in the MSTPCRi register. HAL drivers in Renesas FSP handle module start/stop function automatically.

Implement wake-up mechanisms to bring the system back to full power from the Low Power modes when needed. This could include external interrupts, timers, or communication events triggering wake-up signals to transition the system from a low-power mode to an active mode.

When designing system with Ethernet as the communication media, The power savings can be achieved by using the Wake-on-LAN (WOL). The ETHERC supports Wake-on-LAN (WOL) function to detect a Magic Packet transmitted from a host device or other device, and to wake the MCU from a low power mode such as Sleep.

When using the DCDC mode (switching regulator), it takes 150µsec for the clock to stabilize when system changes from LPM mode to Normal mode vs the 10µs in External VDD mode. This needs to be considered while using the DCDC mode for the system power supply.

To conserve power, set the dividers for any unused clocks (for example, BCLK) to the highest possible value whenever possible. Also, if not using a clock then make sure that it has been stopped by setting the appropriate register(s). The registers for controlling each clock source are shown in the following table.

Table 3. Clock Source Configuration Registers

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
Middle-speed on-chip (MOCO)	MOCOCR	Starts/stops MOCO
Low-speed on-chip (LOCO)	LOCOCR	Starts/stops LOCO

Usage of Ultra-low power peripherals such as ULPT timer instead of GPT as it can be operated in the Low power modes and helps to conserve power.

4.1 PCB hardware design techniques in Designing a System using RA8 MCU

4.1.1 Power Supply

Implementing dedicated power planes for different voltage levels can help minimize voltage drops and noise, improving overall power integrity. Separate power planes for digital and analog sections, along with proper grounding techniques, can help reduce crosstalk and interference, contributing to lower power consumption. RA8 MCUs have two primary power supply voltages, VCC and VCC2. Each of these power supply voltages may be sourced from different external power supplies. Each power supply voltage is used internally for multiple peripherals and I/O blocks. Please refer to the table “Recommended operating conditions” in the “Electrical Characteristics” chapter of the Hardware User’s Manual for details of allowed voltage ranges for each power supply voltage.

The peripherals associated with VCC include both USB and SDRAM. When using either one of these peripherals, VCC should be connected to a nominal 3.3 V supply. However, there may be circumstances where a lower voltage rail may be desired for some higher speed peripherals. For example, Octal SPI memory devices often have an I/O voltage of 1.8 V for lower power consumption and improved performance. In this case, VCC2 can be connected to a lower voltage external power supply.

Details of which MCU ports are associated with which power supply voltage can be found in the table “I/O port functions” in the “I/O Ports” chapter of the Hardware User’s Manual. The individual ports associated with each power supply voltage may vary from one RA8 device to another. When connecting different voltage supply levels to VCC and VCC2, ensure that all ports associated with each power supply voltage will operate correctly at the relevant voltage level.

When connecting VCC and VCC2 to their respective voltage supplies, follow the guidelines in the “Internal Voltage Regulator” chapter of the Hardware User’s Manual, with one exception. The diagrams in that chapter show VCC and VCC2 both connected to the same external power supply. Instead of connecting both power supply voltages to the same external power supply, each power supply voltage may be connected to separate external power supplies. Be sure to include the voltage bypass capacitors indicated in the Hardware User’s Manual for every VCC and VCC2 pin.

In DCDC mode, VDD is supplied from VCL through the VLO output and an external inductor and capacitor. DCDC mode has the advantages that no external supply is needed for VDD and all power up timing is handled within the MCU. However, additional components are needed to support this mode, which can add cost and board space requirements.

In DCDC mode, to minimize power loss and maximize efficiency, Renesas recommends using a 2.2 uH inductor with a DC resistance of 100 mΩ or less. VCC and VCC_DCDC should be shorted together.

4.1.2 Component Placement

Carefully place components on the PCB to minimize trace lengths and avoid sharp turns to reduce impedance, signal distortion and power losses. Position critical components, such as the MCU, voltage regulators, and decoupling capacitors, close to each other to minimize parasitic capacitance and inductance. This reduces power losses and improves signal integrity, especially in high-speed or sensitive circuits.

An initial consideration in this practical model pertains to the widespread use of bypass capacitors. In digital circuits, these capacitors are scattered liberally across the printed circuit board (PCB). Despite their static operation, each bypass capacitor adds a small current draw, typically in the range of a few nanoamps. To address this issue, it becomes imperative to minimize the quantity of bypass capacitors employed. This necessitates meticulous attention to PCB design, including the creation of power planes and strategically bypassing circuitry regions.

Considering capacitor leakage, it's important to pick the right type. Electrolytic capacitors have high leakage. Tantalum capacitors also leak, but less than electrolytics. Ceramics offer a good balance between cost and performance, with low leakage and resistance.

Route power and signal traces efficiently to minimize impedance and parasitic effects. Use wide traces for power lines to reduce resistance and voltage drops. Employ differential signaling and impedance matching techniques for high-speed interfaces to minimize signal reflections and power consumption.

Ensure adequate thermal management by using thermal vias, heatsinks, or thermal pads to dissipate heat efficiently. Overheating can degrade component performance and reduce system reliability, leading to increased power consumption and potential failures.

4.1.3 Clock source

Distribute clock signals efficiently to minimize skew and jitter. Use low-power clock sources and distribute clocks using short, impedance-controlled traces. Implement clock gating techniques to disable clocks to unused peripherals or modules during low-power modes, reducing power consumption without sacrificing functionality.

The RA8 MCUs have five primary oscillators. Four of these may be used as the source for the main system clock. In typical systems, requiring higher clock accuracy, the main clock is driven with an external crystal or clock. This input is directed to PLLn (n=1,2) where it is eventually multiplied up to the PLL clock, then supplied into the CPU clock (CPUCLK), system clock (ICLK), flash clock (FCK), peripheral module clocks (PCLKn), external bus clock, and trace clock. Additional selectors and frequency dividers are used to generate the SCI clock, SPI clock, Octal SPI clock, CANFD clock, LCD clock, USB clocks, and I3C clock. Refer to the Hardware User’s Manual “Clock Generation Circuit” chapter for the “Clock generation circuit block diagram”.

Each clock has specific tolerances and timing values. Refer to the Hardware User’s Manual “AC Characteristics” section in the “Electrical Characteristics” chapter for the Frequency and Clock Timing specifications. Refer to the Hardware User’s Manual “Clock Generation Circuit” chapter for the relationship between the various clock frequencies.

4.1.4 Electromagnetic interference

Design the PCB layout to minimize electromagnetic interference (EMI) and ensure compliance with electromagnetic compatibility (EMC) standards. Implement ground planes, shielding techniques, and proper signal routing to reduce EMI emissions and susceptibility to external interference, improving system stability and reducing power consumption.

4.1.5 Signal Drive Capacity

The drive capacity is typically specified in terms of sink current (current the pin can sink when driving a low signal) and source current (current the pin can source when driving a high signal).

The drive capacity of a CMOS output pin is usually higher than that of an n-ch open-drain configuration because CMOS outputs actively drive both high and low logic levels.

Adjusting the drive capacity settings of the MCU peripheral pins, such as low, medium, or high drive strength, can significantly impact power consumption in embedded systems. Here's how each setting can contribute to power savings:

The best practice for saving power by configuring pins as analog inputs involves setting the pins to operate in a mode where they function as high-impedance inputs. When configured as analog inputs, the pins present a

high-impedance load to the circuit, meaning they draw very little current. This is particularly advantageous when the MCU enters sleep modes or low-power states.

Table 4. Drive Strength Configuration

Drive Strength	Description	Applications
Low	Reduces current sourced/sunk by pins. Slower transitions, higher impedance, reduced power consumption.	Driving light loads. Low-frequency operations.
Medium	Balances drive capability and power consumption. Provides sufficient drive with moderate power usage.	Moderate loads. Frequencies needing compromise.
High	Maximizes current sourced/sunk by pins. Faster transitions, better handling of heavy loads.	High-speed interfaces. Long traces, high impedance.

- The drive capacity switching is controlled by the Drive Capacity Control Register (DSCR) bits in each Port mn Pin Function Select (PmnPFS) register.
- Most port pins have the option of enabling low-, middle-, or high-drive output.
- Some ports have the option of enabling low-, middle-, high-, or high-speed high-drive output. Refer to the “Peripheral Select Settings for Each Product” section in the Hardware User’s Manual for details of the options for each port. • Port0 and P201 are limited to low-drive output only.
- P200 is input only • Out of reset all DSCR registers are cleared to 0 therefore all pins are set to low drive output. Setting a value other than “00” will change the drive capacity of the output for the selected pin.
- The maximum total output of all pins summed together is 80 mA.
- The differences between the drive levels are shown in the following table.

Table 5. Drive Strength Configuration and permissible current per pin

Typical output pins	DSCR[1:0]	Drive Capacity	Average (mA)	Max (mA)
Permissible output current per pin	0 0	Low Drive	2.0	4.0
Permissible output current per pin	0 1	Middle Drive	4.0	8.0
Permissible output current per pin	1 0	High-speed high-drive	20.0	40.0
Permissible output current per pin	1 1	High Drive	16	32

The Driver capacity can also be configured using the FSP configurator for individual pins. This can be done by accessing the Pins tab and selecting the desired port and its pin.

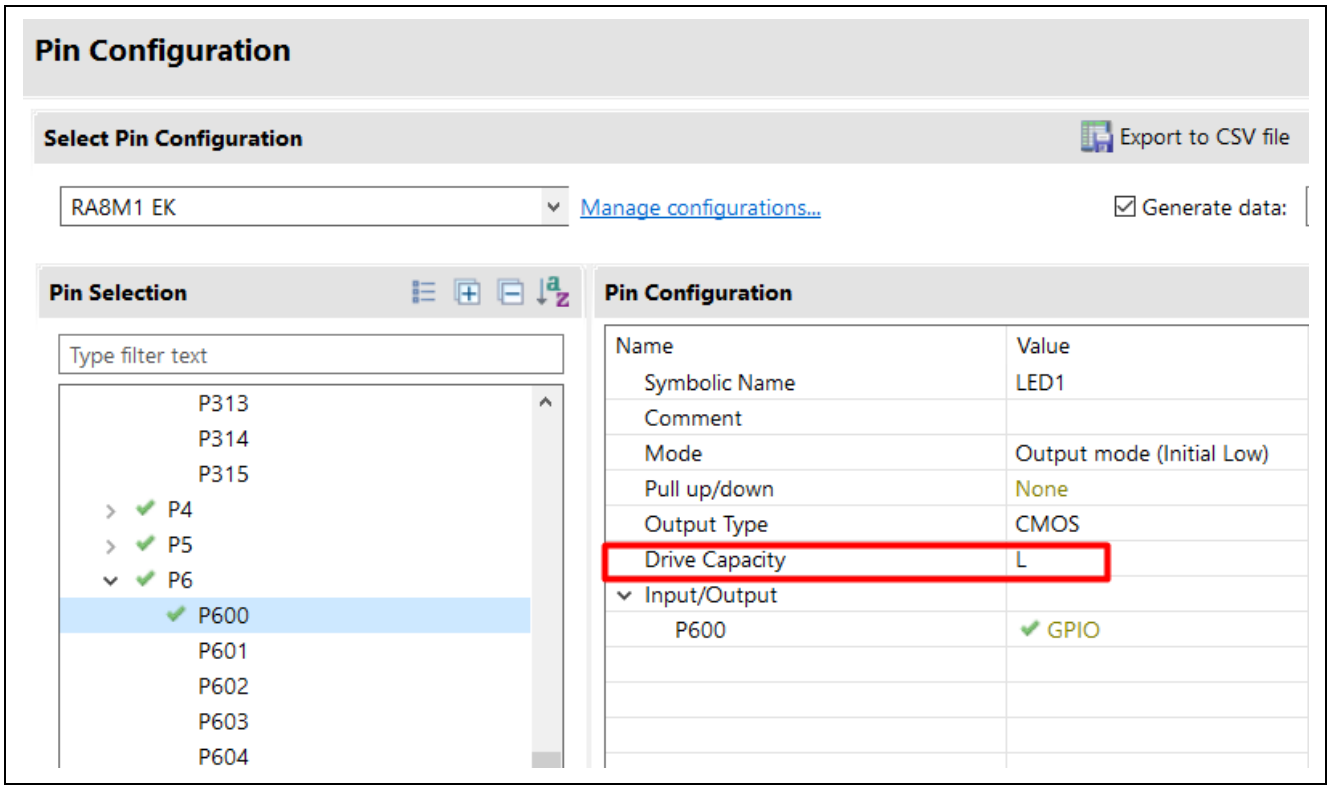

Figure 12. RA8 MCU Pin Drive Capacity Configuration

4.1.6 Unused Pins

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 to the MCU 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 many 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.

Note: Some pins require specific termination: See the "Handling of Unused Pins" section of the Hardware User's Manual for specific recommendations

4.2 Software/Firmware Design using RA8 MCU

When using the RA8 MCU in power-efficient firmware applications, the user needs to focus on crafting highly efficient firmware. This involves not only reducing active mode duration but also maximizing the utilization of low-power modes. This is accomplished through thorough algorithm optimization, eliminating unnecessary polling, and strategically employing interrupts to improve overall system efficiency.

To start with power on reset startup routine, Bootloader plays a small role in the power saving in the system, and minimizing boot time is often a critical requirement. Fine-tuned and tested bootloader can contribute to good amount of power saving during bootup time.

Depending on your system requirements, consider whether it is more efficient to boot directly from flash memory or load the firmware into RAM for execution. The choice may depend on factors like speed and power consumption. Ensure that the bootloader can take advantage of low-power modes provided by the MCU.

On top of the bootloader, if the system has the Over the air (OTA) firmware update, algorithm needs to be efficient to perform the updates. While updating the firmware, the wired or wireless communication module has to be reliable and support the Low power modes when not performing the Job.

It is a Good idea to use the register variables as required instead placing the variable in the memory. The access time of the register variables are shorter compared to the internal/external memories. The frequently used variables inside the function can be stored in the register. It can be accomplished declaring the variable as “register int x; /* Declaring x as a register variable */ “

Write efficient and optimized code to reduce execution time. Minimize unnecessary loops, conditionals, and function calls. Design the application to be event-driven, where tasks are triggered by external events or interrupts rather than polling continuously. This reduces CPU wake-ups and enables the system to remain in low power modes for longer periods.

Reduce the number of function calls, especially within tight loops or frequently executed code sections. Each function call incurs overhead in terms of stack manipulation, parameter passing, and return address management, which can contribute to increased power consumption. Instead of breaking down tasks into small functions, consider consolidating related functionality into larger functions to minimize the overhead associated with function calls.

It's recommended to utilize local variables for frequently accessed data within functions. This practice helps minimize the need to fetch data from external memory, reducing additional clock cycles required for data retrieval. RA MCUs offer faster SRAM blocks that can be effectively utilized for critical code sections in the application.

Avoid floating-point arithmetic if possible, as it generally consumes more power than integer operations. Always use fixed-point arithmetic wherever feasible or use integer-based approximations for certain calculations.

Pointers can be used instead of passing the variable to the function. Inline functions also helps to avoid the parameter passing, while increases the flash code size.

Minimize or avoid dynamic memory allocation and deallocation. Use fixed-size buffers or pre-allocate memory pools to avoid the overhead of dynamic memory management.

Optimize your Data Structures. Choose data structures that minimize memory usage and access times. Utilize bit fields and packed structures to reduce memory footprint.

Firmware optimization is also achieved by intelligently harnessing the capabilities Data cache, Instruction Cache. Data cache and Instruction cache are essential for creating optimized firmware in resource-constrained systems. By leveraging RA8 caches effectively, firmware can improve data access latency, reduce memory traffic, conserve energy, and enhance overall system performance. However, it's crucial to consider cache size, cache coherence, cache replacement policies, and cache management strategies to maximize the benefits of caching in firmware optimization.

RA8 MCUs Tightly Coupled Memories such as ITCM and DTCM are physically located very close to the CPU core, this proximity reduces memory access latency, allowing instructions to be fetched and executed quickly and play important roles in creating optimized firmware by reducing memory access latency, providing deterministic access times, enabling fast interrupt handling, optimizing data processing, improving power efficiency.

Internal static RAM such as SRAM, Standby SRAM (SSRAM), aiming to not only amplify computational efficiency but also contribute to reduce power consumption within the system. RA8 SSRAM can retain the critical data during the Low power modes. Designing application by taking advantage of SSRAM feature can improve the faster mode transition from Low power mode to Normal mode.

Harnessing the RA8 MCUs M85's Helium instruction architecture alongside compatible compilers like LLVM presents a significant opportunity to enhance system performance while simultaneously reducing the CPU's active time. This optimization consequently extends the duration the CPU spends in low-power modes, further enhancing overall energy efficiency of the system.

5. Example Projects and Reference Applications and Application Notes

Renesas RA MCU's Power saving techniques are demonstrated through the Application Projects and Example projects available on Renesas.com. Users can refer to them for details.

5.1 Reference Example Projects for Demonstrating the LPM Modes

The Example project demonstrates the basic functionalities of the Low Power Mode module (LPM) on Renesas RA MCUs using Renesas FSP. The project illustrates methods to reduce MCU power consumption and restore the pre-LPM states of peripheral modules. The MCU will automatically enter each LPM mode and then wait for cancel sources to trigger manually by an external interrupt or automatically by a timer to exit LPM mode.

The Example projects are available at ra-fsp-examples/example_projects/ek_ra8m1/lpm/lpm_ek_ra8m1_ep at master · renesas/ra-fsp-examples · GitHub

This example project also demonstrates the usage of Standby SRAM (SSRAM) by storing the data in the SSRAM and going into Standby mode and retrieving it back in the normal mode.

5.2 Reference Application Projects

The Application projects showcasing the different Low power modes in the application, LPM activation and cancellation of the sources, changing the clocks at run time, LPM transition at runtime are demonstrated. The same methods can be tailored when designing the Applications using Low Power modes. This can be found in the link, https://www.renesas.com/us/en/document/scd/lpm-application.

6. References

- Renesas FSP User's Manual: https://renesas.github.io/fsp
- Renesas RA MCU Datasheets: See http://renesas.com/ra and select the relevant MCU
- RA8 Example Projects on Renesas RA GitHub: https://github.com/renesas/ra-fsp-examples
- RA8 Quick Design Guide (r01an7087eu0100) RA8 MCU Quick Design Guide (renesas.com)

7. Website and Support

Visit the following vanity URLs to learn about key elements of the RA family, download components and related documentation, and get support.

RA Product Information	renesas.com/ra
RA Product Support Forum	renesas.com/ra/forum
RA Flexible Software Package	renesas.com/FSP
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Apr.24.24	-	Initial version

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended 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; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

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

Contact information

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