Date: Feb. 25, 2026

RENESAS TECHNICAL UPDATE

TOYOSU FORESIA, 3-2-24, Toyosu, Koto-ku, Tokyo 135-0061, Japan
Renesas Electronics Corporation

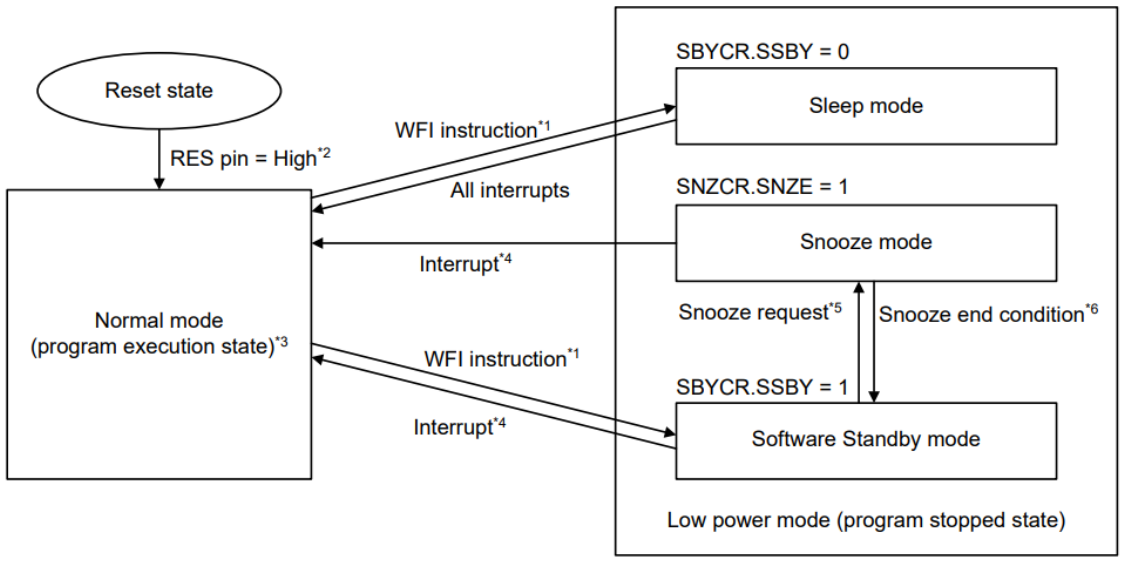
Product Category	MPU/MCU		Document No.	TN-RIS-A0001B/E	Rev.	2.00
Title	Note about the interruption during the transition to low power modes		Information Category	Technical Notification		
Applicable Product	Renesas R9A02G021	Lot No. All	Reference Document	Renesas R9A02G021 User's Manual: Hardware R01UH1036EJ0110 Rev.1.10		

Corrections are made to the figures and tables in the user's manual hardware as shown in 1 and 2 below.
If the software meets the applicable condition listed in 3 below, you may not be able to enter the intended low power mode and transit to unintended states described in 3 below.
If the unintended states described in the following 3 notes cannot be tolerated, use the following 4 workaround.

Contents

1. Correction of the Figure “Mode transitions”

Fig10.1 is corrected as follows. Note 1 sentence is corrected and note 7 is added.



Note 1. When an interrupt that acts as a trigger for cancel is received during a transition to the program stopped state after the execution of a WFI instruction, the MCU executes interrupt exception handling instead of a transition to low power mode.
Note 2. The MOCO is the source of the operating clock following a transition from the reset state to Normal mode.
Note 3. The transition to Normal mode is made from an interrupt in Sleep mode, Software Standby mode, or Snooze mode. The clock source is the same as before entering the low power mode.
Note 4. See Table 10.3.
Note 5. See section 10.5.1. Setting Operating Power Control Mode.
Note 6. See section 10.8.3. Returning from Snooze Mode to Software Standby Mode.

Figure 10.1 Low power mode transitions

Before correction

Note 1. When an interrupt that acts as a trigger for cancel is received during a transition to the program stopped state after the execution of a WFI instruction, the MCU executes interrupt exception handling instead of a transition to low power mode.

After correction (correction of Note 1. and addition of Note 7.)

Note 1. When an interrupt that acts as a trigger for cancellation of the Sleep mode state is received during a transition to the Sleep mode, the MCU does not transit to Software standby mode state but go back to after the execution of a WFI instruction, the MCU executes interrupt exception handling instead of a transition to low power mode.

Note. 7 When an interrupt acts as a trigger for cancellation of the Sleep mode state is received during a transition to the Software Standby mode state, the MCU does not transit to Software Standby mode state but go back to Normal mode state.

2. Correction of the table “Operating conditions of each low power mode”

Tables 10.2 is corrected.

Before correction

Parameter	Sleep mode	Software Standby mode	Snooze mode
Transition condition	WFI instruction while SBYCR.SSBY = 0	WFI instruction while SBYCR.SSBY = 1	Snooze request trigger in Software Standby mode. SNZCR.SNZE = 1
State after cancellation by an interrupt	Program execution state (interrupt processing)	Program execution state (interrupt processing)	Program execution state (interrupt processing)

After correction

Parameter	Sleep mode	Software Standby mode	Snooze mode
Transition condition	When below conditions are met while SBYCR.SSBY=0 ・WFI instruction ・A valid interrupt request(*1) cannot be accepted to CPU. (including a transition from the time WFI instruction is executed to the time the transition to Sleep mode is completed)	When below conditions are met while SBYCR.SSBY=1 ・WFI instruction ・A valid interrupt request(*1) cannot be accepted to CPU. (including a transition from the time WFI instruction is executed to the time the transition to Software Standby mode is completed)	Snooze request trigger in Software Standby mode. SNZCR.SNZE = 1
State after cancellation by an interrupt	Program execution state	Program execution state	Program execution state

(*1) Valid interrupt requests are any interrupt/exception that are not masked by the priority level of current exception and the priority level set by mintthresh. In addition, if the interrupt request is maskable interrupt, the interrupt must be enabled by clicintie[i].

3. Applicable condition and notes

[Applicable condition]

Transition to Software Standby mode is started by a trigger (WFI instruction or SLEEPONEXIT) with SBYCR.SSBY=1 set to use Software Standby, or Snooze mode.

During the specified interval (ICLK 2cycle) of transitioning to Software Standby mode, one of the following interrupt requests that is not an interrupt source to return from Software Standby mode is accepted by CPU.

1) maskable interrupt which is not for cancelling the Software Standby mode (all of the following are applicable)

・Cancelling Software Standby mode is not accepted by WUPEN0/WUPEN1 and exception number ID is either 3 or 7 or 19 - 50 in Interrupt vector table .

・Interrupt requests are enabled by CLIC Interrupt Enable Register (clicintie[i]).

・Interrupt requests are not masked by Machine interrupt-level threshold (mintthresh) .

(mintthresh=0 or mintthresh < defined level by clicintctl[i])

2) Non-maskable interrupt request triggered by the following sources

SRAM parity error

SRAM ECC failure

[Notes]

If the above conditions are met, the MCU will transit to following unintended states.

These unintended states can be resolved by a reset or returning to Normal mode with an interrupt request of an interrupt source to return from Software Standby mode.

Adapt a workaround if these unintended states are not acceptable.

1) When transitioning to Software Standby mode (SBYCR.SSBY=1, SNZCR.SNZE=0)

Only CPU clock is stopped, and the remaining clocks continue to operate as they were prior to transitioning Software Standby mode.

- As before the transition to Software Standby mode is started, depending on the setting, timer or other peripherals continue to operate, and an interrupt request related to the peripheral is generated.
- Because the IWDT and WDT clock-stop function is disabled, a reset or an interrupt for the IWDT and WDT is generated depending on the settings before starting the transition to Software Standby mode.
- Interrupt requests are held in IR flag (IELSRn).

2) When transitioning to Snooze mode (SBYCR.SSBY=1, SNZCR.SNZE=1)

The transition to Snooze mode is not possible, and the states shown in “1) When transitioning to Software Standby mode” is continued.

To return to Normal mode by an interrupt source (SELSR0) from Snooze mode depends on whether the interrupt request (SELSR0) to returning from Snooze mode can be generated while DTC operation is disabled.

If DTC operation is disabled (SNZCR.SNZDTCEN=0) in Snooze mode, Normal mode can be returned because an interrupt source for the interrupt source (SELSR0) to return from Snooze mode can be generated.

If DTC operation is enabled (SNZCR.SNZDTCEN=1) in Snooze mode, the Normal mode cannot be returned because an interrupt request for the interrupt source (SELSR0) to return from Snooze mode cannot be generated.

4. Workaround

[Workaround]

To avoid the unintended states described above, apply the following before the condition for transition to Software Standby mode, or Snooze mode are met: (For the setting procedure, see "Setting Procedure for Transition to Software Standby, or Snooze Mode")

1) Disable maskable interrupt requests that are not interrupt sources to return from Software Standby mode.

Exception number 3,7,19-50 of Interrupt vector table that WUPEN0/WUPEN1 does not allow to return from Software Standby mode

3) Stop access from the bus master other than the CPU so that the non-maskable interrupt is not triggered by the following sources.

SRAM parity error

SRAM ECC failure

Setting Procedure for Transition to Software Standby, or Snooze Mode

This section describes procedures for avoiding unintended states.

The handling of interrupt requests after returning from Software Standby or Snooze mode varies depending on the method used to disable the maskable interrupt request. Either one or the other should be applied.

Procedure A) Disable maskable interrupt request acceptance.

Any interrupt request that occurs while interrupt request acceptance is disabled is discarded.

Before transitioning to Software Standby, Snooze mode

Step1: Stop the bus access from the bus master other than CPU. (*1)

Step2: Set CLIC clicintie[7].ie to 0 to disable Machine timer interrupt request. (*2)

Read CLIC clicintie[7].ie to confirm CLIC clicintie[7].ie has been cleared.

Step3: Clear IELSRn in ICU to disable acceptance of maskable interrupt requests that are not interrupt sources to return from Software Standby mode.

Step4: Read IELSRn in ICU to confirm that IELSRn in ICU has been cleared.

Step5: Transition to Software Standby mode (WFI instruction)

After returning from Software Standby or Snooze mode

Step6: Set CLIC clicintie[7].ie to 1 to enable Machine timer interrupt request.

Step7: Set IELSRn in ICU to enable acceptance of maskable interrupt requests that are not interrupt sources to return from Software Standby mode.

Step8: Enable bus access from bus masters other than CPU.

Procedure B) Disable the maskable interrupt request

The interrupt request generated while the interrupt request is disabled is retained in IELSRn.IR flag. Therefore, after returning from Software Standby or Snooze mode and enabling the maskable interrupt, it is possible to process the interrupt.

Before transitioning to Software Standby, Snooze mode

Step1: Stop the bus access from the bus master other than CPU. (*1)

Step2: Set CLIC clicintie[i].ie to 0 to disable maskable interrupt requests that are not interrupt sources to return from Software Standby mode. (*2)

Step3: Read CLIC clicintie[i].ie to confirm CLIC clicintie[i].ie has been cleared.

Step4: Transition Software Standby mode (WFI instruction)

After returning from Software Standby or Snooze mode

Step5: Set CLIC clicintie[i].ie to 1 to enable maskable interrupt requests that are not interrupt sources to return from Software Standby mode.

Step6: Enable bus access from bus masters other than CPU.

*1: SRAM parity error interrupt, SRAM ECC error interrupt, is enabled as a non-maskable interrupt.

*2: Disabling a Machine timer interrupt request may cause Machine timer interrupt request to be delayed by one cycle of Machine timer without generating the latest Machine timer interrupt request.