Date: May 13, 2025

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-RA*-A0146A/E	Rev. 1.00
Title	Note about the interruption during the transition to low power modes		Information Category	Technical Notification	
Applicable Product	RA4M1, RA4M2, RA4M3, RA4E1, RA4E2, RA4T1, RA4W1, RA4L1, RA6M1, RA6M2, RA6M3, RA6M4, RA6M5, RA6E1, RA6E2, RA6T1, RA6T2, RA6T3 Group	Lot No. All	Reference Document	Refer table at the end of this document	

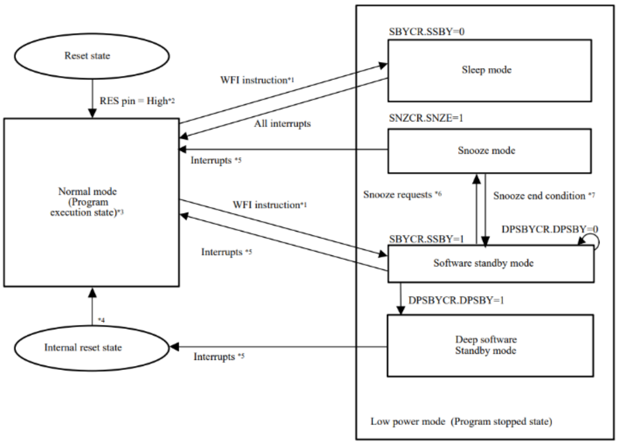
Corrections are made to the figures and tables in the user's manual hardware as shown in 1 and 2 below.
If the software meets the applicable condition listed in 4 below, you may not be able to enter the intended low power mode and transit to unintended states described in 4 below.
If the unintended states described in the following 4 notes cannot be tolerated, use the following 5 workaround.

Contents

1. Correction of the Figure “Mode transitions”

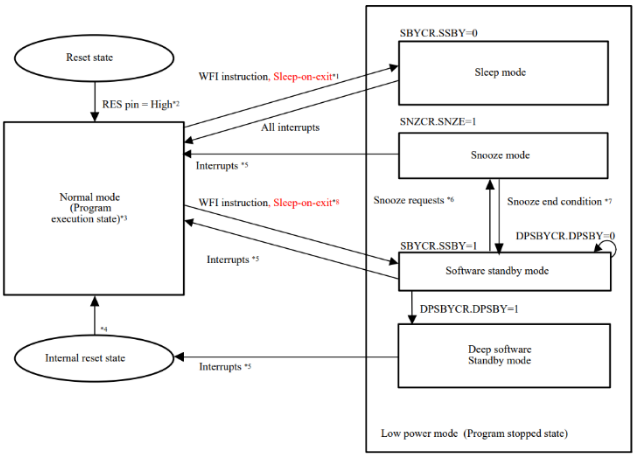
1) Figure 10.1 of RA4M2, RA4M3, RA4E1, RA4E2, RA4T1, RA6M4, RA6M5, RA6E1, RA6E2, RA6T2, and RA6T3 groups

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 transitioning to low power mode.
Note 2. The MOCO clock 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 because of an interrupt from Sleep mode, Snooze mode, or Software Standby mode. The clock source is the same as before entering the low power mode.
Note 4. When an available interrupt request is generated, an internal reset (Deep Software Standby reset) is generated over a fixed period. Canceling of Deep Software Standby mode accompanies release from the internal reset state, and then the MCU transitions to Normal mode and executes reset exception processing with the MOCO clock as the source of the operating clock.
Note 5. See Table 10.3
Note 6. See Table 10.7
Note 7. See Table 10.9

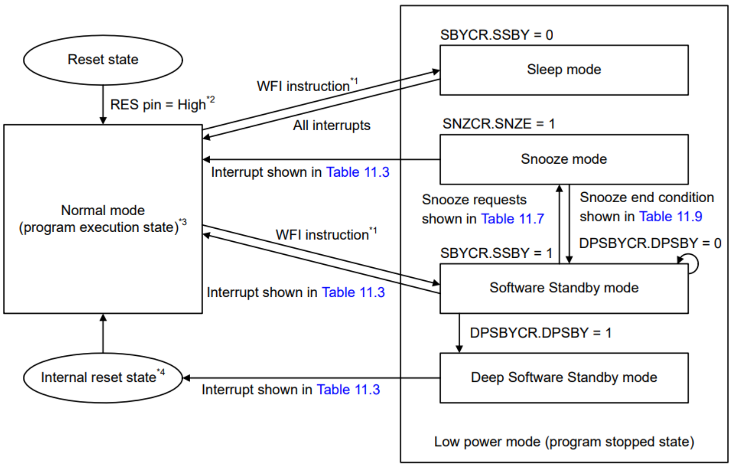
After correction



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 state, the MCU does not transit to Sleep mode state but go back to Normal mode state.
Note 2. The MOCO clock 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 because of an interrupt from Sleep mode, Snooze mode, or Software Standby mode. The clock source is the same as before entering the low power mode.
Note 4. When an available interrupt request is generated, an internal reset (Deep Software Standby reset) is generated over a fixed period. Canceling of Deep Software Standby mode accompanies release from the internal reset state, and then the MCU transitions to Normal mode and executes reset exception processing with the MOCO clock as the source of the operating clock.
Note 5. See Table 10.3
Note 6. See Table 10.7
Note 7. See Table 10.9
Note 8. When an interrupt that 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) Figure 11.1 of RA6M1, RA6M2, RA6M3, and RA6T1 groups

Before correction



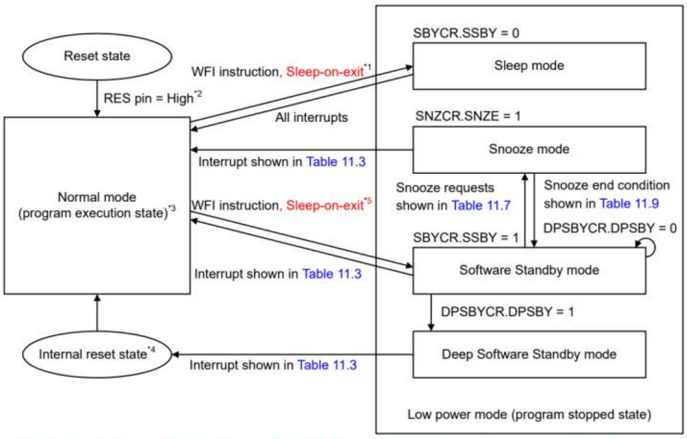
Note 1. When an interrupt that acts as a trigger for cancel is received during a transition to the program-stopped state after the execution of a WFI instruction, the MCU executes interrupt exception handling instead of transitioning to low power mode.

Note 2. The MOCO clock 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, Snooze mode, or Software Standby mode. The clock source is the same as before entering the low power mode.

Note 4. When an available interrupt request is generated, an internal reset (Deep Software Standby reset) is generated over a fixed period. Canceling of Deep Software Standby mode accompanies release from the internal reset state, and the MCU transitions to Normal mode and executes a reset exception processing with the MOCO clock as the source of the operating clock.

After correction



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 state, the MCU does not transit to Sleep mode state but go back to Normal mode state.

Note 2. The MOCO clock is the source of the operating clock following a transition from the reset state to Normal mode.

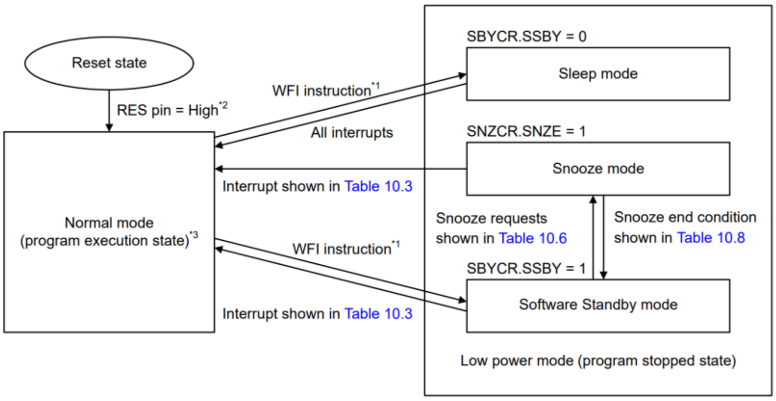
Note 3. The transition to Normal mode is made from an interrupt in Sleep mode, Snooze mode, or Software Standby mode. The clock source is the same as before entering the low power mode.

Note 4. When an available interrupt request is generated, an internal reset (Deep Software Standby reset) is generated over a fixed period. Canceling of Deep Software Standby mode accompanies release from the internal reset state, and the MCU transitions to Normal mode and executes a reset exception processing with the MOCO clock as the source of the operating clock.

Note 5. When an interrupt that 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.

3) Figure 10.1 of RA4M1group

Before correction

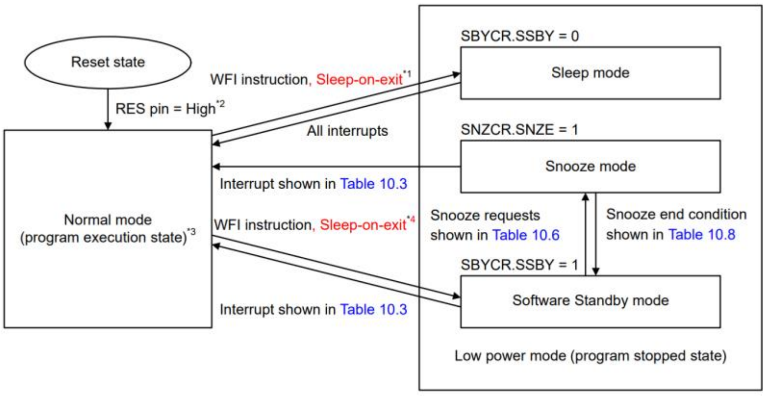


Note 1. When an interrupt that acts as a trigger for cancel is received during a transition to the program stopped state after the execution of a WFI instruction, the MCU executes interrupt exception handling instead of a transition to low power mode.

Note 2. The MOCO clock 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.

After correction



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 state, the MCU does not transit to Sleep mode state but go back to Normal mode state.

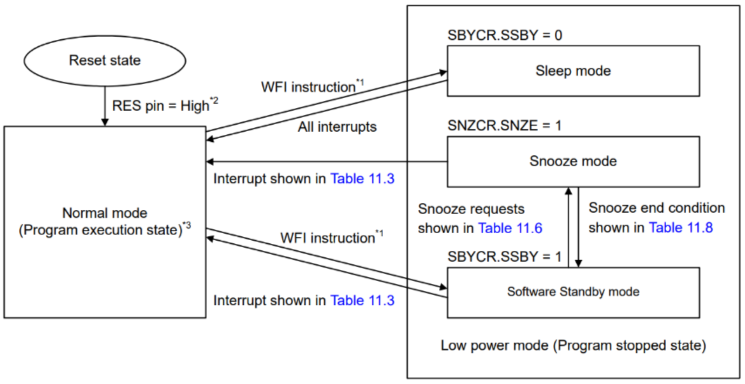
Note 2. The MOCO clock is the source of the operating clock following a transition from the reset state to Normal mode.

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. When an interrupt that 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.

4) Figure 11.1 of RA4W1 group

Before correction

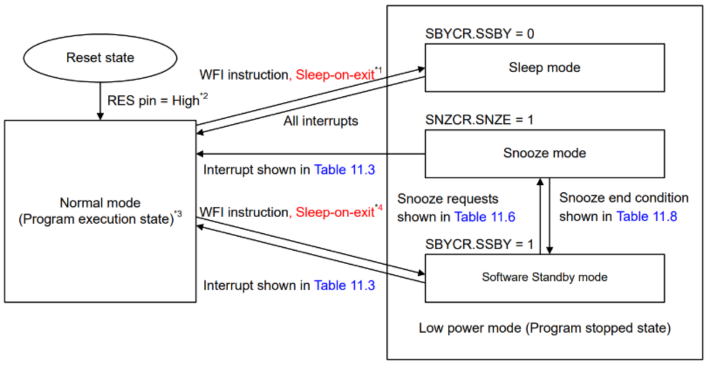


Note 1. When an interrupt that acts as a trigger for cancel is received during a transition to the program stopped state after the execution of a WFI instruction, the MCU executes interrupt exception handling instead of a transition to low power mode.

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.

After correction



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 state, the MCU does not transit to Sleep mode state but go back to Normal mode state.

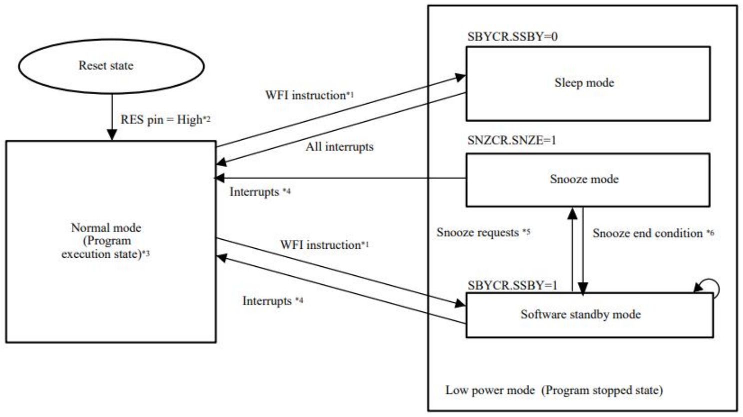
Note 2. The MOCO is the source of the operating clock following a transition from the reset state to Normal mode.

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. When an interrupt that 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.

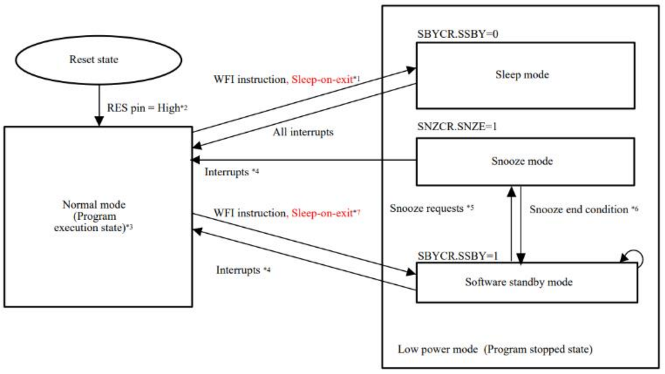
5) Figure 10.1 of RA4L1 group

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 transitioning to low power mode.
Note 2. The MOCO clock 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 because of an interrupt from Sleep mode, Snooze mode, or Software Standby mode. The clock source is the same as before entering the low power mode.
Note 4. See Table 10.3.
Note 5. See Table 10.8.
Note 6. See Table 10.10.

After correction



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 state, the MCU does not transit to Sleep mode state but go back to Normal mode state.
Note 2. The MOCO clock 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 because of an interrupt from Sleep mode, Snooze mode, or Software Standby mode. The clock source is the same as before entering the low power mode.
Note 4. See Table 10.3.
Note 5. See Table 10.8.
Note 6. See Table 10.10.
Note 7. When an interrupt that 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”

1) Tables 10.2 of RA4M2, RA4M3, RA4E1, RA4E2, RA4T1, RA6M4, RA6M5, RA6E1, RA6E2, RA6T2, and RA6T3 groups and Tables 11.2 for RA6M1, RA6M2, RA6M3 and RA6T1 groups

Before correction

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

After correction

Parameter	Sleep mode	Software Standby mode	Snooze mode	Deep Software Standby mode
Transition condition	When [Condition 1] or [Condition 2] while SBYCR.SSBY=0 [Condition 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 Sleep mode is completed) [Condition 2] ・SCR.SLEEPONEXIT=1 ・Complete execution of all exception handlers ・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 [Condition 1] or [Condition 2] while SBYCR.SSBY=1 and DPSBYCR.DPSBY=0 [Condition 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 ~~sleep~~ Software Standby mode is completed) [Condition 2] ・SCR.SLEEPONEXIT=1 ・Complete execution of all exception handlers ・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	When [Condition 1] or [Condition 2] while SBYCR.SSBY=1 and DPSBYCR.DPSBY=1 [Condition 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) [Condition 2] ・SCR.SLEEPONEXIT=1 ・Complete execution of all exception handlers ・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)
State after cancellation by an interrupt	Program execution state	Program execution state	Program execution state	Reset 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 BASEPRI. In addition, if the interrupt request is based on IELSRn, the interrupt must be enabled by NVIC_ISERn.

2) Table 10.2 for RA4M1, RA4L1 groups and table 11.2 for RA4W1 group

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 [Condition 1] or [Condition 2] while SBYCR.SSBY=0 [Condition 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 Sleep mode is completed) [Condition 2] ・SCR.SLEEPONEXIT=1 ・Complete execution of all exception handlers ・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 [Condition 1] or [Condition 2] while SBYCR.SSBY=1 and DPSBYCR.DPSBY=0 [Condition 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) [Condition 2] ・SCR.SLEEPONEXIT=1 ・Complete execution of all exception handlers ・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 BASEPRI. In addition, if the interrupt request is based on IELSRn, the interrupt must be enabled by NVIC_ISERn.

3. Notes about the Sleep-on-exit function

There are 2 ways to transition to low power modes. One is WFI instruction and the other is Sleep-on-exit. When Sleep-on-exit is used for transition to low power modes, WFI instruction comments written in User's Manual Hardware is applicable to Sleep-on-exit.

4. 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, Snooze, or Deep Software Standby 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) SysTick interrupt (all of the following are applicable)

・Exception number 15 of Interrupt vector table .

・Interrupt requests are not masked by Base Priority Mask Register (BASEPRI)

(BASEPARI=0 or BASEPRI > SHPR3.PRI_15)

2) Maskable interrupt requests that are not interrupt source to return from Software Standby mode (all of the following are applicable)

・By WUPEN in exception numbers 16 to 111 in the interrupt vector table those not permitted to return from Software Standby mode

・Interrupt requests are enabled by Interrupt Set-Enable Register (NVIC_ISERn).

・Interrupt requests are not masked by Base Priority Mask Register (BASEPRI)

(BASEPARI=0 or BASEPRI > NVIC_IPRn.PRI_N)

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

SRAM parity error

SRAM ECC failure

MPU bus master error

MPU bus slave error

TrustZone filtering error

[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, DPSBYCR.DPSBY=0, 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, DELSRn).

2) When transitioning to Snooze mode (SBYCR.SSBY=1, DPSBYCR.DPSBY=0, 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.

3) When transitioning to Deep Software Standby mode (SBYCR.SSBY=1, DPSBYCR.DPSBY=1)

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

As for the interrupt source (DPSIERn) settings to returning from Deep Software Standby mode, these interrupt requests can return to Normal mode only when the interrupt source (WUPEN) to return from Software Standby mode is set.

5. Workaround

[Workaround]

To avoid the unintended states described above, apply the following before the terms for transition to Software

Standby mode, Snooze mode, or Deep Software Standby mode are met: (For the setting procedure, see "Setting Procedure for Transition to Software Standby, Snooze, or Deep Software Standby Mode")

1) Disable SysTick interrupt requests.

Exception number 15 of Interrupt vector table

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

Exception number 16～111 of Interrupt vector table that WUPEN 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

MPU bus master error

MPU bus slave error

TrustZone filtering error

Setting Procedure for Transition to Software Standby, Snooze, or Deep Software Standby 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, or Deep Software Standby mode

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

Step2: Disable the SysTick interrupt request. (*2)

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, SLEEPONEXIT)

After returning from Software Standby ~~mode~~ or Snooze mode

Step6: Enable the SysTick 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.

Step 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, or Deep Software Standby mode

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

Step2: Disable the SysTick interrupt request. (*2)

Step3: Write 1 to the corresponding bit in NVIC_ICERn in CPU to disable maskable interrupt requests that are not interrupt sources to return from Software Standby mode.

Step4: Execute Data Synchronization Barrier (DSB) instruction.

Step5: Transition Software Standby mode (WFI instruction, SLEEPONEXIT)

After returning from Software Standby or Snooze mode

Step6: Enable the SysTick interrupt request.

Step7: Write 1 to the corresponding bit in NVIC_ISERn in CPU to enable maskable interrupt requests that are not interrupt sources to return from Software Standby mode.

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

*1: SRAM parity error interrupt, SRAM ECC error interrupt, MPU bus master error interrupt, MPU bus slave error interrupt, or TrustZone filter error interrupt is enabled as a non-maskable interrupt.

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

Reference for each product table number, register name, mode, error name.

1.Interrupt vector table is as follows.

Table 12.3 shows the interrupt vector table for RA4E2, RA4T1, RA4L1, RA6E2, RA6T2, and RA6T3 groups.

Table 13.3 shows the interrupt vector table for RA4M1, RA4M2, RA4M3, RA4E1, RA6M4, RA6M5, RA6E1, and RA6T1 groups.

Table 14.3 shows the interrupt vector table for RA4W1, RA6M1, RA6M2, and RA6M3 groups.

2. TrustZone filtering errors and MPU bus slave errors are as follows

RA4M1, RA4W1, RA6M1, RA6M2, RA6M3, and RA6T1 groups have MPU bus slave error and do not have TrustZone filtering error.

RA4M2, RA4M3, RA4E1, RA4E2, RA4T1, RA4L1, RA6M4, RA6M5, RA6E1, RA6E2, RA6T2, and RA6T3 groups have TrustZone filtering errors and do not have MPU bus slave error.

3. Deep Software Standby mode existence is as follows.

RA4M1, RA4W1, and RA4L1 groups do not have Deep Software Standby mode.

4.The maximum exception number in the interrupt vector table is as follows:

47 for RA4M1, and RA4W1 groups. 79 for RA4L1 group.

111 for RA4M2, RA4M3, RA4E1, RA4E2, RA4T1, RA6M1, RA6M2, RA6M3, RA6M4, RA6M5, RA6E1, RA6E2, RA6T1, RA6T2 and RA6T3 groups.

5. WUPEN resister name for each product is as follows.

WUPEN0 is used for RA6T2 group.

WUPEN0/1 is used for RA4M2, RA4M3, RA4E1, RA4E2, RA4T1, RA6M4, RA6M5, RA6E1, RA6E2, and RA6T3 groups.

WUPEN0/1/2 is used for RA4L1 group.

Related Documentation

Product	Document Name
RA4M1 Group	Renesas RA4M1 Group User’s Manual: Hardware Rev. 1.10
RA4M2 Group	Renesas RA4M2 Group User’s Manual: Hardware Rev. 1.30
RA4M3 Group	Renesas RA4M3 Group User’s Manual: Hardware Rev. 1.40
RA4E1 Group	Renesas RA4E1 Group User’s Manual: Hardware Rev. 1.20
RA4E2 Group	Renesas RA4E2 Group User’s Manual: Hardware Rev. 1.30
RA4T1 Group	Renesas RA4T1 Group User’s Manual: Hardware Rev. 1.20
RA4W1 Group	Renesas RA4W1 Group User’s Manual: Hardware Rev. 1.00
RA4L1 Group	Renesas RA4L1 Group User’s Manual: Hardware Rev. 1.00
RA6M1 Group	Renesas RA6M1 Group User’s Manual: Hardware Rev. 1.20
RA6M2 Group	Renesas RA6M2 Group User’s Manual: Hardware Rev.1.20
RA6M3 Group	Renesas RA6M3 Group User’s Manual: Hardware Rev.1.20
RA6M4 Group	Renesas RA6M4 Group User’s Manual: Hardware Rev.1.40
RA6M5 Group	Renesas RA6M5 Group User’s Manual: Hardware Rev.1.30
RA6E1 Group	Renesas RA6E1 Group User’s Manual: Hardware Rev.1.20
RA6E2 Group	Renesas RA6E2 Group User’s Manual: Hardware Rev. 1.30
RA6T1 Group	Renesas RA6T1 Group User’s Manual: Hardware Rev. 1.20
RA6T2 Group	Renesas RA6T2 Group User’s Manual: Hardware Rev. 1.40
RA6T3 Group	Renesas RA6T3 Group User’s Manual: Hardware Rev. 1.20