[Notes]
CS+ Code Generator for RX,
e^2 studio Code Generator Plug-in,
AP4 Coding Assistance Tool for RX

R20TS0462EJ0100
Rev.1.00
Aug. 01, 2019

Outline

When using the products in the title, note the following point.

1. When using the NACK reception transfer suspension function on the I^{2}C bus interface
2. When using sleep mode

1. When Using the NACK Reception Transfer Suspension Function on the I^{2}C Bus Interface

1.1 Applicable Products

- CS+ Code Generator for RX V1.00.00 (CS+ for CC V2.01) or later
- Code Generator plug-in V1.00.00 (e^2 studio V2.1.0) or later
- AP4 for RX V1.00.00 or later

1.2 Applicable Devices

- RX family:
  RX110, RX111, RX113, RX130, RX230, RX231, RX23T, RX24T, RX24U,
  RX64M, RX651, RX65N, and RX71M groups

1.3 Details

When the NACK reception transfer suspension function on the I²C bus interface is used, error interrupt processing when receiving NACK is incorrect, which affects operation as follows.

In the master mode, communication is not suspended because code to resume communication is generated.

In the slave mode, unnecessary processing is performed.

➢ Error location

Master mode

```
/***********************************************************************************
* Function Name: r_Config_RIIC0_error_interrupt
* Description  : This function is EEI0 interrupt service routine
* Arguments    : None
* Return Value : None
***********************************************************************************/
void r_Config_RIIC0_error_interrupt(void)
{
    volatile uint8_t dummy;
    …
    else if ((1U == RIIC0.ICIER.BIT.NAKIE) && (1U == RIIC0.ICSR2.BIT.NACKF))
        …
        RIIC0.ICSR2.BIT.NACKF = 0U;        <-- Communication resumes with this code

        r_Config_RIIC0_callback_receiveerror(MD_ERROR3);
    }
    …
}
```

Slave mode

```
/***********************************************************************************
* Function Name: r_Config_RIIC0_error_interrupt
* Description  : This function is EEI0 interrupt service routine
* Arguments    : None
* Return Value : None
***********************************************************************************/
void r_Config_RIIC0_error_interrupt(void)
{
    volatile uint8_t dummy;
    …
    else if ((1U == RIIC0.ICIER.BIT.NAKIE) && (1U == RIIC0.ICSR2.BIT.NACKF))
    {
        dummy = RIIC2.ICDRR;               <-- These codes are unnecessary


        RIIC0.ICSR2.BIT.NACKF = 0U;
        r_Config_RIIC0_callback_receiveerror(MD_ERROR3);
    }
    …
}
```

1.4 Workaround

Modify the error process when receiving NACK in the error interrupt function in the source file below.

Note: When code is generated again, generated code returns to the state before correction. Therefore, correct the source file each time you generate code.

- Source file: r_cg_riic_user.c
- Function: void r_riic*n*_error_interrupt(void)

  n = Channel number

Below is an example of modification for RX64M group. Modification is shown in red.

We have also included an example of a user program when receiving NACK. Add the code by referring to this sample program.

➢ For Channel0 (Master mode)

Workaround

```
/***********************************************************************
* Function Name: r_Config_RIIC0_error_interrupt
* Description  : This function is EEI0 interrupt service routine
* Arguments    : None
* Return Value : None
***********************************************************************/
void r_Config_RIIC0_error_interrupt(void)
{
    volatile uint8_t dummy;
    …
    else if ((1U == RIIC0.ICIER.BIT.NAKIE) && (1U == RIIC0.ICSR2.BIT.NACKF))
    {
        RIIC0.ICSR2.BIT.NACKF = 0U;      <-- Remove this line.

        r_Config_RIIC0_callback_receiveerror(MD_ERROR3);
    }
    …
}
```

Example of a user program when receiving NACK:

Below is a sample program to prepare for the next communication when receiving NACK.

```
void NackReceive(void)
{
    volatile uint8_t dummy;

    /* Disable the interrupt for RIIC communication */
    R_Config_RIIC0_Stop();

    /* Stop condition issuance */
    RIIC0.ICSR2.BIT.STOP = 0U;
    RIIC0.ICCR2.BIT.SP = 1U;

    /* Dummy read when master reception */
    dummy = RIIC0.ICDRR;

    /* Wait for stop condition issuance */
    while (0U == RIIC0.ICSR2.BIT.STOP);

    /* Clear status flags */
    RIIC0.ICSR2.BIT.NACKF = 0U;
    RIIC0.ICSR2.BIT.STOP = 0U;

    /* Enable the interrupt for RIIC communication */
    R_Config_RIIC0_Start();
}
```

➢ For Channel0 (Slave mode)

Workaround

```
/***********************************************************************
* Function Name: r_Config_RIIC0_error_interrupt
* Description  : This function is EEI0 interrupt service routine
* Arguments    : None
* Return Value : None
***********************************************************************/
void r_Config_RIIC0_error_interrupt(void)
{
    volatile uint8_t dummy;
    …
    else if ((1U == RIIC0.ICIER.BIT.NAKIE) && (1U == RIIC0.ICSR2.BIT.NACKF))
    {
        dummy = RIIC2.ICDRR;

        RIIC0.ICSR2.BIT.NACKF = 0U;
        r_Config_RIIC0_callback_receiveerror = MD_ERROR3;
    }
    …
}
```

Remove this line.

Example of a user program when receiving NACK:

Below is a sample program to prepare for the next communication when receiving NACK.

```
void NackReceive(void)
{
    volatile uint8_t dummy;

    /* Disable the interrupt for RIIC communication */
    R_Config_RIIC0_Stop();

    /* Dummy read */
    dummy = RIIC0.ICDRR;

    /* Wait for stop condition issuance */
    while (0U == RIIC0.ICSR2.BIT.STOP);

    /* Clear status flags */
    RIIC0.ICSR2.BIT.NACKF = 0U;
    RIIC0.ICSR2.BIT.STOP = 0U;

    /* Enable the interrupt for RIIC communication */
    R_Config_RIIC0_Start();
}
```

1.5 Schedule for Fixing the Problem

There is no schedule for fixing this problem.

2. When Using Sleep Mode

2.1 Applicable Products

- CS+ Code Generator for RX V1.00.00 (CS+ for CC V2.01) or later
- Code Generator plug-in V1.00.00 (e^2 studio V2.1.0) or later
- AP4 for RX V1.00.00 or later

2.2 Applicable Devices

- RX family:
  RX64M, RX651, RX65N, and RX71M groups

2.3 Details

If you enable clock source switching on return from sleep mode, a clock source error identification error occurs in the R_LPC_Sleep function, causing transition to sleep mode to not be performed correctly. For details of incorrect operation, see Table 2.1.

Figure 2.1 Incorrect operation of the R_LPC_Sleep function

Applicable devices	Clock source when transitioning to sleep mode				
	MAINCLK	SUBCLK	PLLCLK	HOCO	LOCO
RX64M RX651 RX65N RX71M	**2**	**3**	**1**	**2**	**3**

[Incorrect operation]

1. Transition to sleep mode occurs when using a clock source which does not allow transition to sleep mode.
2. Although transition to sleep mode occurs when using a clock source which does not allow transition to sleep mode, the return value is normal end (MD_OK).
3. After transitioning to sleep mode, the return value is abnormal end (MD_ERROR1).

2.4 Workaround

In the following source file, correct the error identification processing when sleep mode return clock source switching is enabled

Note: When code is generated again, generated code returns to the state before correction. Therefore, correct the source file each time you generate code.

- Source file: r_cg_lpc.c
- Function: void R_LPC_Sleep(void)

If the operating clock is not a sub-clock or LOCO, correct the source file in a way that an error code is set and the program is aborted. Modification is shown in red.

Before modification

```
/***********************************************************************
* Function Name: R_LPC_Sleep
* Description  : This function enables sleep mode.
* Arguments    : None
* Return Value : status -
*                 MD_OK or MD_ERROR1
***********************************************************************/
MD_STATUS R_LPC_Sleep(void)
{
    …
    /* When RSTCKEN is enable, the possible clock source is HOCO or
       main clock oscillator */
    if ((1U == SYSTEM.RSTCKCR.BIT.RSTCKEN) &&
         (SYSTEM.SCKCR3.BIT.CKSEL != _0100_LPC_CLOCKSOURCE_HOCO) &&
         (SYSTEM.SCKCR3.BIT.CKSEL  != _0200_LPC_CLOCKSOURCE_MAINCLK))
    {
        status = MD_ERROR1;
    }
    …
}
```

After modification

```
/***********************************************************************
* Function Name: R_LPC_Sleep
* Description  : This function enables sleep mode.
* Arguments    : None
* Return Value : status -
*                 MD_OK or MD_ERROR1
***********************************************************************/
MD_STATUS R_LPC_Sleep(void)
{
    …
    /* When RSTCKEN is enable, the possible clock source is LOCO or
       sub clock oscillator */
    if ((1U == SYSTEM.RSTCKCR.BIT.RSTCKEN) &&
         (SYSTEM.SCKCR3.BIT.CKSEL != 0U) &&
         (SYSTEM.SCKCR3.BIT.CKSEL  != 3U))
    {
        status = MD_ERROR1;
        return status;
    }
    …
}
```

2.5 Schedule for Fixing the Problem

There is no schedule for fixing this problem.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Aug.01.19	-	First edition issued

Corporate Headquarters

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

www.renesas.com

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/

Trademarks

TS Colophon 4.0