[Notes]
e² studio Smart Configurator Plug-in,
Smart Configurator for RX

R20TS0466ES0100
Rev.1.00
Aug. 01, 2019

Outline

When using the e² studio Smart Configurator Plug-in and Smart Configurator for RX, note the following point.

1. When using the NACK reception transfer suspension function on the I²C bus interface

1. When Using the NACK Reception Transfer Suspension Function on the I²C Bus Interface

1.1 Applicable Products

- e² studio V6.0.0 (Smart Configurator Plug-in V1.2.0) or later
- Smart Configurator for RX V1.2.0 or later

1.2 Applicable Devices

- RX family:
  RX110, RX111, RX113, RX130, RX230, RX231, RX23T, RX24T, RX24U,
  RX64M, RX651, RX65N, RX66T, RX71M, RX72M and RX72T 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: "<*I²C-configuration-name*>_user.c"
- Function: "void r_<*I²C- configuration-name*>_error_interrupt(void)"

  The <*I²C- configuration-name*> varies depending on the selected component of I²C master mode or I²C slave mode.

  Below is an example of modification when the <*I²C-configuration-name*> is Config_RIIC0 (initial value) for RX64M. 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 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();
}
```

- 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 these lines.

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

This problem will be fixed in the next version. (Scheduled to be released in November 2019.)

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