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

R20TS0426EJ0110
Rev.1.10
May. 16, 2019

Outline

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

1. Setting system clock (ICLK) to a frequency higher than 12MHz
2. Using Realtime Clock

1. Setting System Clock (ICLK) to a Frequency Higher Than 12MHz

1.1 Applicable Products

- CS+ Code Generator for RX V1.07.00 (CS+ for CC V3.01) and later versions
- Code Generator plug-in V2.0.1 (e^2 studio V4.0.1.007) and later versions
- AP4 for RX V1.06.00 and later versions

1.2 Applicable Devices

- RX family:
  RX230, RX231, and RX23T groups

1.3 Details

If the system clock (ICLK) is set to a frequency higher than 12MHz, the clock may not operate at the selected frequency because the procedure for switching operating power control modes and processing to wait for completion of the MEMWAIT register setting are not performed correctly.

1.4 Conditions

When the system clock (ICLK) is set to a frequency higher than 12MHz, the error occurs depending on the combinations of device type and code generator version. For details about the conditions and workaround, see section 1.5.

1.5 Workaround

Modify the following source file after each code generation.

- Source file: r_cg_cgc.c
- Function: void R_CGC_Create(void)

Workaround varies depending on the combination of device type and code generator version.

Modify the code according to Table 1 below.

Table 1. Combinations of device type and code generator version and workarounds

Devices	System Clock (ICLK)	CS+ Code Generator for RX V1.16.00 Code Generator plug-in V2.12.0 and later AP4 for RX V1.15.00 and V1.15.01	Other versions
RX230 RX231	32MHz < ICLK	Workaround 1	Workaround 2
	12MHz <ICLK ≤ 32MHz	No fix required	Workaround 5
RX23T	32MHz < ICLK	Workaround 3	Workaround 4
	12MHz <ICLK ≤ 32MHz	No fix required	Workaround 5

Workaround 1: Move the code for setting operating power control modes and correct the code to wait for completion of the MEMWAIT register setting.
Move the code in the blue box in [Before modification] to where the red box is in [After modification].
Modify the blue code in [Before modification] to the red code in [After modification].

Before modification:

```
/***********************************************************************
* Function Name: R_CGC_Create
* Description  : This function initializes the clock generator.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_CGC_Create(void)
{
    …
    /* Set memory wait cycle setting register */
    SYSTEM.MEMWAIT.BIT.MEMWAIT = 1U;
    memorywaitcycle = SYSTEM.MEMWAIT.BYTE;
    memorywaitcycle++;

    /* Set operating power control */
    SYSTEM.OPCCR.BIT.OPCM = _00_LPC_HIGH_SPEED_MODE;
    while (1U == SYSTEM.OPCCR.BIT.OPCMTSF);

    /* Set clock source */
    SYSTEM.SCKCR3.WORD = _0400_CGC_CLOCKSOURCE_PLL;
    while (SYSTEM.SCKCR3.WORD != _0400_CGC_CLOCKSOURCE_PLL);
    …
}
```

After modification:

```
/***********************************************************************
* Function Name: R_CGC_Create
* Description  : This function initializes the clock generator.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_CGC_Create(void)
{
    …
    /* Set operating power control */
    SYSTEM.OPCCR.BIT.OPCM = _00_LPC_HIGH_SPEED_MODE;
    while (1U == SYSTEM.OPCCR.BIT.OPCMTSF);

    /* Set memory wait cycle setting register */
    SYSTEM.MEMWAIT.BIT.MEMWAIT = 1U;
    while(1U == SYSTEM.MEMWAIT.BIT.MEMWAIT);

    /* Set clock source */
    SYSTEM.SCKCR3.WORD = _0400_CGC_CLOCKSOURCE_PLL;
    while (SYSTEM.SCKCR3.WORD != _0400_CGC_CLOCKSOURCE_PLL);
    …
}
```

Workaround 2: Add the code for setting operating power control modes and correct the code to wait for completion of the MEMWAIT register setting.
Add the code in the red box in [After modification].
Modify the blue code in [Before modification] to the red code in [After modification].

Before modification:

```
/***********************************************************************
* Function Name: R_CGC_Create
* Description  : This function initializes the clock generator.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_CGC_Create(void)
{
    …
    /* Set memory wait cycle setting register */
    SYSTEM.MEMWAIT.BIT.MEMWAIT = 1U;
    memorywaitcycle = SYSTEM.MEMWAIT.BYTE;
    memorywaitcycle++;

    /* Set clock source */
    SYSTEM.SCKCR3.WORD = _0400_CGC_CLOCKSOURCE_PLL;

    while (SYSTEM.SCKCR3.WORD != _0400_CGC_CLOCKSOURCE_PLL);
    …
}
```

After modification:

```
/***********************************************************************
* Function Name: R_CGC_Create
* Description  : This function initializes the clock generator.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_CGC_Create(void)
{
    /* Set operating power control */
    SYSTEM.OPCCR.BIT.OPCM = _00_LPC_HIGH_SPEED_MODE;
    while (1U == SYSTEM.OPCCR.BIT.OPCMTSF);

    /* Set memory wait cycle setting register */
    SYSTEM.MEMWAIT.BIT.MEMWAIT = 1U;
    while(1U == SYSTEM.MEMWAIT.BIT.MEMWAIT);

    /* Set clock source */
    SYSTEM.SCKCR3.WORD = _0400_CGC_CLOCKSOURCE_PLL;

    while (SYSTEM.SCKCR3.WORD != _0400_CGC_CLOCKSOURCE_PLL);
    …
}
```

Workaround 3: Move the code for setting operating power control modes.
Move the code in the blue box in [Before modification] to where the red box is in [After modification].

Before modification:

```
/***********************************************************************
* Function Name: R_CGC_Create
* Description  : This function initializes the clock generator.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_CGC_Create(void)
{
    …
    /* Set memory wait cycle setting register */
    SYSTEM.MEMWAIT.BIT.MEMWAIT = 1U;

    while(1U == SYSTEM.MEMWAIT.BIT.MEMWAIT);

    /* Set operating power control */
    SYSTEM.OPCCR.BIT.OPCM = _00_LPC_HIGH_SPEED_MODE;
    while (1U == SYSTEM.OPCCR.BIT.OPCMTSF);

    /* Set clock source */
    SYSTEM.SCKCR3.WORD = _0400_CGC_CLOCKSOURCE_PLL;

    while (SYSTEM.SCKCR3.WORD != _0400_CGC_CLOCKSOURCE_PLL);
    …
}
```

After modification:

```
/***********************************************************************
* Function Name: R_CGC_Create
* Description  : This function initializes the clock generator.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_CGC_Create(void)
{
    …
    /* Set operating power control */
    SYSTEM.OPCCR.BIT.OPCM = _00_LPC_HIGH_SPEED_MODE;
    while (1U == SYSTEM.OPCCR.BIT.OPCMTSF);

    /* Set memory wait cycle setting register */
    SYSTEM.MEMWAIT.BIT.MEMWAIT = 1U;

    while(1U == SYSTEM.MEMWAIT.BIT.MEMWAIT);

    /* Set clock source */
    SYSTEM.SCKCR3.WORD = _0400_CGC_CLOCKSOURCE_PLL;

    while (SYSTEM.SCKCR3.WORD != _0400_CGC_CLOCKSOURCE_PLL);
    …
}
```

Workaround 4: Add the code for setting operating power control modes.
Add the code in the red box in [After modification]

Before modification:

```
/***********************************************************************
* Function Name: R_CGC_Create
* Description  : This function initializes the clock generator.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_CGC_Create(void)
{
    …
    /* Set memory wait cycle setting register */
    SYSTEM.MEMWAIT.BIT.MEMWAIT = 1U;

    while(1U == SYSTEM.MEMWAIT.BIT.MEMWAIT);

    /* Set clock source */
    SYSTEM.SCKCR3.WORD = _0400_CGC_CLOCKSOURCE_PLL;

    while (SYSTEM.SCKCR3.WORD != _0400_CGC_CLOCKSOURCE_PLL);
    …
}
```

After modification

```
/***********************************************************************
* Function Name: R_CGC_Create
* Description  : This function initializes the clock generator.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_CGC_Create(void)
{
    …
    /* Set operating power control */
    SYSTEM.OPCCR.BIT.OPCM = _00_LPC_HIGH_SPEED_MODE;
    while (1U == SYSTEM.OPCCR.BIT.OPCMTSF);

    /* Set memory wait cycle setting register */
    SYSTEM.MEMWAIT.BIT.MEMWAIT = 1U;

    while(1U == SYSTEM.MEMWAIT.BIT.MEMWAIT);

    /* Set clock source */
    SYSTEM.SCKCR3.WORD = _0400_CGC_CLOCKSOURCE_PLL;

    while (SYSTEM.SCKCR3.WORD != _0400_CGC_CLOCKSOURCE_PLL);
    …
}
```

Workaround 5: Add the code for setting operating power control modes.
Add the code in the red box in [After modification].

Before modification:

```
/***********************************************************************
* Function Name: R_CGC_Create
* Description  : This function initializes the clock generator.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_CGC_Create(void)
{
    …
    /* Set clock source */
    SYSTEM.SCKCR3.WORD = _0400_CGC_CLOCKSOURCE_PLL;

    while (SYSTEM.SCKCR3.WORD != _0400_CGC_CLOCKSOURCE_PLL);
    …
}
```

After modification:

```
/***********************************************************************
* Function Name: R_CGC_Create
* Description  : This function initializes the clock generator.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_CGC_Create(void)
{
    …
    /* Set operating power control */
    SYSTEM.OPCCR.BIT.OPCM = _00_LPC_HIGH_SPEED_MODE;
    while (1U == SYSTEM.OPCCR.BIT.OPCMTSF);

    /* Set clock source */
    SYSTEM.SCKCR3.WORD = _0400_CGC_CLOCKSOURCE_PLL;

    while (SYSTEM.SCKCR3.WORD != _0400_CGC_CLOCKSOURCE_PLL);
    …
}
```

1.6 Schedule for Fixing the Problem

There is no schedule for fixing this problem.

2. Using Realtime Clock

2.1 Applicable Products

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

2.2 Applicable Devices

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

2.3 Details

Realtime clock may not operate because there is no processing to wait for supply of 6 clocks after supply of the clock source is started.

2.4 Conditions

The problem occurs when using the realtime clock.

2.5 Workaround

Modify the following source file after each code generation.

- Source file: r_cg_cgc.c

Function: void R_CGC_Create(void)

The following is an example of modification for RX231 (ICLK=16MHz). Add the processing shown in red. Set the optimum count for the loop processing to wait for supply of 6 clocks of count source according to your clock settings.

Before modification:

```
/***********************************************************************
Includes
***********************************************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefine.h"

/***********************************************************************
* Function Name: R_RTC_Create
* Description  : This function initializes the RTC module.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_RTC_Create(void)
{
    uint32_t read_count;
    volatile uint32_t dummy;
    uint32_t w_count;

    /* Disable ALM, PRD and CUP interrupts */
    IEN(RTC, ALM) = 0U;
    IEN(RTC, PRD) = 0U;
    IEN(RTC, CUP) = 0U;

    /* Set sub-clock oscillator */
    while (RTC.RCR3.BIT.RTCEN != 1U)
    {
        RTC.RCR3.BIT.RTCEN = 1U;
    }

    /* Stop all counters */
    RTC.RCR2.BYTE = 0x00U;
    while (RTC.RCR2.BIT.START != 0U);
    …
}
```

After modification:

```
/***********************************************************************
Includes
***********************************************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_rtc.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefine.h"

/***********************************************************************
* Function Name: R_RTC_Create
* Description  : This function initializes the RTC module.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_RTC_Create(void)
{
    uint32_t read_count;
    volatile uint32_t dummy;
    uint32_t w_count;

    /* Disable ALM, PRD and CUP interrupts */
    IEN(RTC, ALM) = 0U;
    IEN(RTC, PRD) = 0U;
    IEN(RTC, CUP) = 0U;

    /* Set sub-clock oscillator */
    while (RTC.RCR3.BIT.RTCEN != 1U)
    {
        RTC.RCR3.BIT.RTCEN = 1U;
    }

    /* Wait for supply 6 clocks of count source */
    for (w_count = 0U; w_count < 267; w_count++)
    {
        nop();
    }

    /* Stop all counters */
    RTC.RCR2.BYTE = 0x00U;
    while (RTC.RCR2.BIT.START != 0U);
    …
}
```

2.6 Schedule for Fixing the Problem

There is no schedule for fixing this problem.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	May.16.19	-	First edition issued
1.10	Jun. 07. 19	1-7	Corrected Outline, section 1.1 and 1.3 through 1.5.

Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

The past news contents have been based on information at the time of publication. Now changed or invalid information may be included.

URLs in Tool News also may be subject to change or become invalid without prior notice.

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

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

© 2019. Renesas Electronics Corporation. All rights reserved.

TS Colophon 4.0