【注意事項】
CS+用 RX コード生成
e2 studio Code Generator プラグイン
RX コード生成支援ツール AP4

R20TS0426JJ0110
Rev.1.10
2019.05.16 号

概要

タイトルに記載している製品の使用上の注意事項を連絡します。

1. システムクロック(ICLK)を 12MHz を超える周波数に設定する場合の注意事項
2. リアルタイムクロック使用時の注意事項

1. システムクロック(ICLK)を 12MHz を超える周波数に設定する場合の注意事項

1.1 該当製品

- CS+用 RX コード生成 V1.07.00 (CS+ for CC V3.01) 以降
- Code Generator プラグイン V2.0.1 (e2 studio V4.0.1.007) 以降
- RX コード生成支援ツール AP4 V1.06.00 以降

1.2 該当デバイス

- RX ファミリ：
  RX230、RX231、および RX23T グループ

1.3 内容

システムクロック(ICLK)を 12MHz を超える周波数に設定する場合、動作電力制御モードの切り替え、および MEMWAIT レジスタの設定の完了を確認する処理が適切に行われないため、指定した周波数で動作しない場合があります。

1.4 発生条件

システムクロック(ICLK)を 12MHz を超える周波数に設定する場合、デバイスとコード生成ツールバージョンの組み合わせにより発生します。発生条件の詳細と回避策については、1.5 項をご参照ください。

1.5　回避策

コード生成を行う度に、下記ソースファイルを修正してください。

・ソースファイル：r_cg_cgc.c

・関数：void R_CGC_Create(void)

デバイスとコード生成ツールバージョンの組み合わせにより修正方法が異なります。

以下の表 1 にしたがって、修正を行ってください。

表 1. デバイスとコード生成ツールバージョンの組み合わせと修正方法

デバイス	システムクロック (ICLK)	CS+用コード生成 V1.16.00 Code Generator プラグイン V2.12.0 以降 AP4 V1.15.00, V1.15.01	その他のバージョン
RX230, RX231	32MHz < ICLK	修正方法 1	修正方法 2
	12MHz <ICLK≦32MHz	修正不要	修正方法 5
RX23T	32MHz < ICLK	修正方法 3	修正方法 4
	12MHz <ICLK≦32MHz	修正不要	修正方法 5

修正方法１：動作電力モード設定処理の移動、メモリウェイト設定完了待ち処理の修正
「修正前」の**青枠**のコードを、「修正後」の**赤枠**の箇所に移動してください。
また、「修正前」の**青文字**を「修正後」の**赤文字**に修正してください。

修正前：

```
/***********************************************************************
* 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);
    …
}
```

修正後：

```
/***********************************************************************
* 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);
    …
}
```

修正方法 2：動作電力モード設定処理の追加、メモリウェイト設定完了待ち処理の修正
「修正後」の**赤枠**のコードを追加してください。
また、「修正前」の**青文字**を「修正後」の**赤文字**に修正してください。

修正前：

```
/***********************************************************************
* 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);
    …
}
```

修正後：

```
/***********************************************************************
* 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);
    …
}
```

修正方法 3：動作電力モード設定処理の移動

「修正前」の**青枠**のコードを、「修正後」の**赤枠**の箇所に移動してください。

修正前：

```
/***********************************************************************
* 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);
    ...
}
```

修正後：

```
/***********************************************************************
* 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);
    ...
}
```

修正方法 4：動作電力モード設定処理の追加
「修正後」の**赤枠**のコードを追加してください。

修正前：

```
/***********************************************************************
* 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);
    …
}
```

修正後：

```
/***********************************************************************
* 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);
    …
}
```

修正方法５：動作電力モード設定処理の追加
「修正後」の**赤枠**のコードを追加してください。

修正前：

```
/***********************************************************************
* 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);
    …
}
```

修正後：

```
/***********************************************************************
* 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　恒久対策

改修予定はありません。

2. リアルタイムクロック使用時の注意事項

2.1 該当製品

- CS+用 RX コード生成 V1.00.00 (CS+ for CC V2.01) 以降
- Code Generator プラグイン V1.00.00 (e^2 studio V2.1.0) 以降
- RX コード生成支援ツール AP4 V1.00.00 以降

2.2 該当デバイス

- RX ファミリ：
  RX111、RX113、RX130、RX230、RX231、RX64M、RX71M、RX651、および RX65N グループ

2.3 内容

クロックソース供給開始後に 6 クロック分のクロック供給待ち処理がないため、リアルタイムクロックが動作しない場合があります。

2.4 発生条件

リアルタイムクロックを使用する場合に発生します。

2.5 回避策

コード生成を行う度に、下記ソースファイルを修正してください。

・ソースファイル：r_cg_rtc.c

・関数：void R_RTC_Create(void)

以下に RX231(ICLK=16MHz)の場合の修正例を記します。赤文字の部分の処理を追加してください。
6 カウントソース供給を待つループ処理の回数は、お客様のクロック設定に合わせて、最適な回数に設定してください。

修正前：

```
/***********************************************************************
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);
    …
}
```

修正後：

```
/***********************************************************************
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　恒久対策

改修予定はありません。

以上

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	May.16.19	-	新規発行
1.10	Jun. 07. 19	1 - 7	概要、1.1、および 1.3~1.5 の修正

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/