[Notes]
Smart Configurator for RH850

R20TS0744EC0100
Rev.1.00
Sep. 16, 2021

Outline

When using Smart Configurator for RH850, note the following points.

1. Notes on selecting PLL0 clock CPLL0OUT as source of CPU Subsystem clock
2. Notes on using CSIH Master receive and Master transmit/receive operation mode

1. Notes on selecting PLL0 clock CPLL0OUT as source of CPU Subsystem clock

1.1 Applicable Products

Smart Configurator for RH850 V1.4.0

1.2 Applicable Devices

RH850 family: RH850/F1KH-D8 group

- RH850/F1KH-D8 (176-pin, 233-pin, 324-pin product)

1.3 Details

When selecting PLL0 clock CPLL0OUT as source of CPU Subsystem clock (refer to Figure1-1), the clock initialization enters an infinite loop, and the program cannot continue to run.

- RH850/ F1KH-D8: 176-pin, 233-pin, 324-pin product

  Clocks

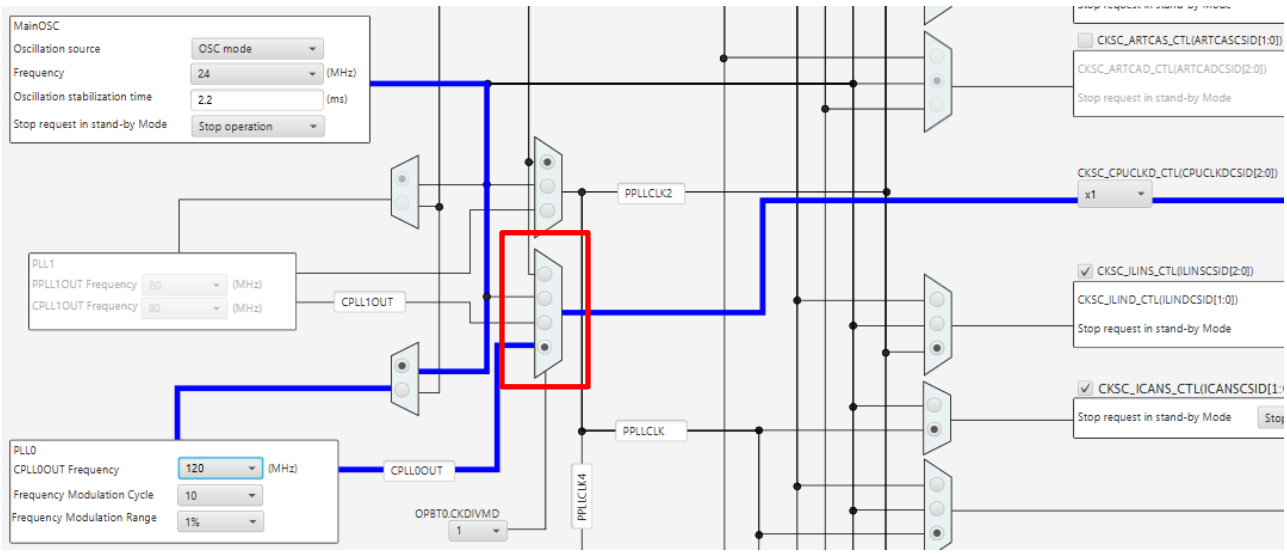

Figure 1-1 Setting of CPU Subsystem clock source

1.4 Workaround

User can manually modify the code in the following source file

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

Note: If code is generated again, the previous state is restored. Modification is necessary each time you perform code generation.

This is an example of the required modification. User should manually set protect register and modify CLKCTL.CKSC_CPUCLKD_CTL register setting statement. In the following example, the code in red color is wrong code before modification, while the code in blue color is correct code after modification.

Before modification:

```
 void R_CGC_Create(void)
 {
     ……
   /* CPU clock setting */
     ……
   while (CLKCTL.CKSC_CPUCLKD_ACT != (_CGC_CPLLOUT_DIVIDER_4 |
_CGC_CPU_CLK_DIVIDER_2))
   {
      NOP();
   }
   CLKCTL.CKSC_CPUCLKD_CTL = _CGC_CPLLOUT_DIVIDER_4 | _CGC_CPU_CLK_DIVIDER_1;
   CLKCTL.CKSC_CPUCLKD_CTL = (uint32_t) _CGC_CPLLOUT_DIVIDER_4 |
_CGC_CPU_CLK_DIVIDER_1;
   CLKCTL.CKSC_CPUCLKD_CTL = _CGC_CPLLOUT_DIVIDER_4 | _CGC_CPU_CLK_DIVIDER_1;
     ……
 }
```

After modification:

```
 void R_CGC_Create(void)
 {
     ……
   /* CPU clock setting */
     ……
   while (CLKCTL.CKSC_CPUCLKD_ACT != (_CGC_CPLLOUT_DIVIDER_4 |
_CGC_CPU_CLK_DIVIDER_2))
   {
      NOP();
   }
   WPROTR.PROTCMD1 = _WRITE_PROTECT_COMMAND;       --> add this line
   CLKCTL.CKSC_CPUCLKD_CTL = _CGC_CPLLOUT_DIVIDER_4 | _CGC_CPU_CLK_DIVIDER_1;
   CLKCTL.CKSC_CPUCLKD_CTL = (uint32_t)~(_CGC_CPLLOUT_DIVIDER_4 |
_CGC_CPU_CLK_DIVIDER_1);                        --> add "~" before macro name
   CLKCTL.CKSC_CPUCLKD_CTL = _CGC_CPLLOUT_DIVIDER_4 | _CGC_CPU_CLK_DIVIDER_1;
     ……
 }
```

1.5 Schedule for Fixing the Problem

This problem will be fixed in the next version. (Scheduled to be released in Nov 2021.)

2. Notes on using CSIH Master receive and Master transmit/receive operation mode

2.1 Applicable Products

Smart Configurator for RH850 V1.4.0 and earlier version

2.2 Applicable Devices

RH850 family: RH850/F1KH-D8, RH850/F1KM-S4, RH850/F1KM-S1 group

- RH850/F1KM-S1 (48-pin, 64-pin, 80-pin, and 100-pin products)
- RH850/F1KM-S4 (100-pin, 144-pin, 176-pin, and 233-pin products)
- RH850/F1KH-D8 (176-pin, 233-pin, 324-pin product)

2.3 Details

When using CSI Master component with operation mode of “Master receive” and “Master transmit/receive” (refer to figure 2-1), selecting the master supplies chip select (CS) signal to the slaves (refer to figure 2-2), master can only receive the first data correctly. From the second data, master fails to receive.

- RH850/F1KM-S1: 48-pin, 64-pin products

  CSIH0
- RH850/F1KM-S1: 80-pin products

  CSIH0, CSIH1, CSIH2
- RH850/F1KM-S1: 100-pin products

  CSIH0, CSIH1, CSIH2, CSIH3
- RH850/F1KM-S4: 100-pin, 144-pin, 176-pin, 233-pin products

  CSIH0, CSIH1, CSIH2, CSIH3
- RH850/F1KH-D8: 176-pin, 233-pin, 324-pin product

  CSIH0, CSIH1, CSIH2, CSIH3

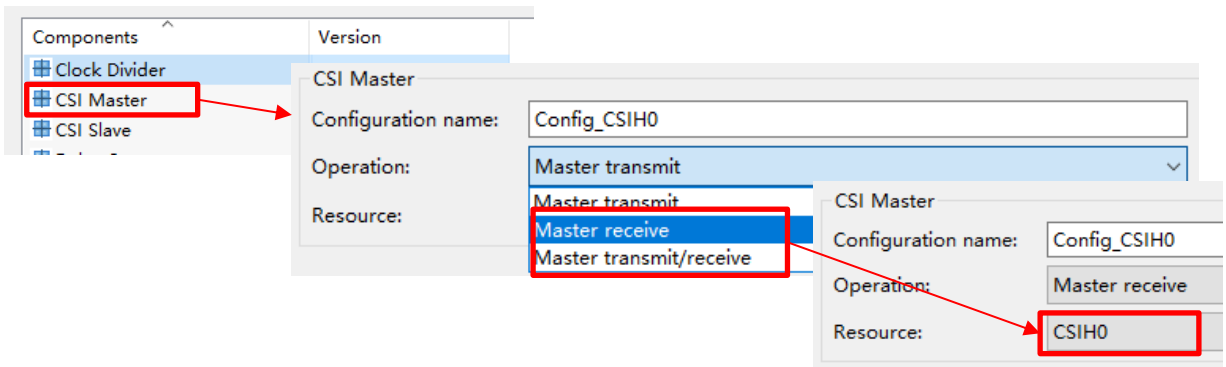

Figure 2-1 Setting of CSI Master operation mode (using CSIH0 as an example)

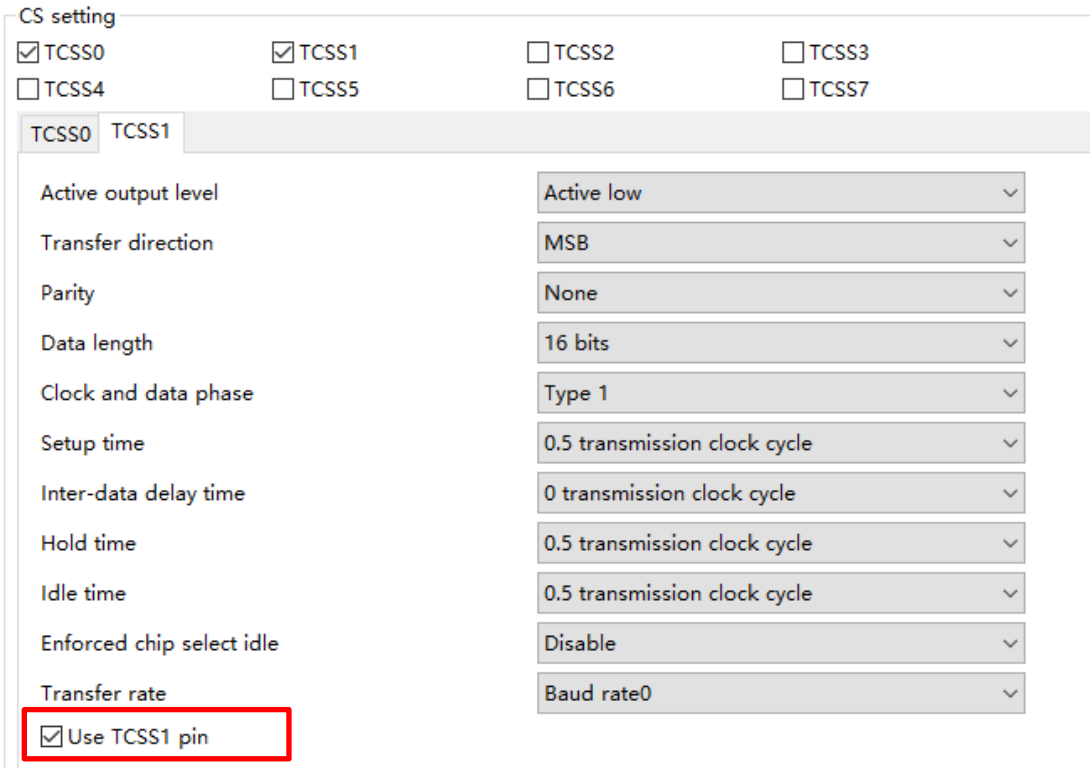

Figure 2-2 Setting of CSIHn master supplies chip select signal to slave

2.4 Workaround

User can manually modify the register setting code in the following source file

- Source file: “<Configuration-name>_user.c”.
- Function: “void r_<*Configuration-name*>_ interrupt_receive (void)”

Note: If code is generated again, the previous state is restored. Modification is necessary each time you perform code generation.

The following is an example of the required modification when <*Configuration-name*> is Config_CSIH0 in the RH850/F1KM group. The code in red color is wrong code before modification, while the code in blue color is correct code after modification.

Before modification:

```
void r_Config_CSIH0_interrupt_receive(void)
{
    uint8_t err_type;
    uint16_t temp;
     ……
    else
    {
        temp = g_csih0_rx_total_num;
        if (temp > g_csih0_rx_num)
        {
            *gp_csih0_rx_address = CSIH0.RX0W;
            gp_csih0_rx_address++;
            g_csih0_rx_num++;
            CSIH0.TX0W = 0xFFFFFFFFUL;
        }
        ……
    }
}
```

After modification:

```
void r_Config_CSIH0_interrupt_receive(void)
{
    uint8_t err_type;
    uint16_t temp;
     ……
    else
    {
        temp = g_csih0_rx_total_num;
        if (temp > g_csih0_rx_num)
        {
            *gp_csih0_rx_address = CSIH0.RX0W;
            gp_csih0_rx_address++;
            g_csih0_rx_num++;
            CSIH0.TX0W |= 0xFFFFUL;
        }
        ……
    }
}
```

2.5 Schedule for Fixing the Problem

This problem will be fixed in the next version. (Scheduled to be released in Nov 2021.)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Sep.16.21	-	First edition issued

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

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/

TS Colophon 4.2