RENESAS Tool News

RENESAS TOOL NEWS on January 16, 2016: 160116/tn5

Note on Using the Following Tools:
CS+ Code Generator for RL78 (CS+ for CC),
CS+ Code Generator for RL78 (CS+ for CA and CX),
e2 studio Code Generator Plug-in,
Applilet3 Coding Assistance Tool for RL78

When using the CS+ Code Generator for RL78 (CS+ for CC), the CS+ Code Generator for RL78 (CS+ for CA and CX), the e2 studio (Code Generator Plug-in), and the Applilet3 coding assistance tool for RL78, take note of the problem on the following point that is described in this note.

- Transfer of data with a length of 10 or more bits through an element of a serial array unit configured as a CSI or data with a length of 16 bits through an element configured as a UART
  Applicable MCUs: RL78/F12, RL78/F13, RL78/F14, RL78/F15, and RL78/D1A groups

---

1. Applicable Products
   - V2.08.00 and later versions of the CS+ Code Generator for RL78 (CS+ for CC)
   - V2.08.00 and later versions of the CS+ Code Generator for RL78 (CS+ for CA and CX)
   - V4.0.1.007 and later versions of the e2 studio (V2.00.01 and later versions of the Code Generator Plug-in)
   - V1.08.00 and later versions of the Applilet3 coding assistance tool for RL78

2. Applicable MCUs
   - RL78/F12 group
   - RL78/F13 group
   - RL78/F14 group
   - RL78/F15 group
   - RL78/D1A group (only supported by Applilet3)

3. Description

Generated code has an error when an element of a serial array unit is set up for use as a 3-line serial (CSI) port and the length of data is specified as 10 or more bits, or the unit is set up for use as a UART and the length of data is specified as 16 bits.

4. Workaround

Modify functions in r_cg_serial.c and r_cg_serial_user.c in the ways shown below. These modifications are required every time code is generated.

(1) In usage as a CSI

Modify the data calculation for transmission by the CSI sending function R_CSImn_Send(void) or the CSI transmission function R_CSImn_Send_Receive(void) in r_cg_serial.c, and the CSI communications completed interrupt function r_csimn_interrupt(void) in r_cg_serial_user.c.

Note: m and n mean the unit number and channel number respectively.

- Example of CSI transmission for channel 1 of unit 0

Before modification:

```
-----------------------------------------------------------------------
MD_STATUS R_CSI01_Send_Receive(uint8_t * const tx_buf,
                               uint16_t tx_num,
                               uint8_t * const rx_buf)
{
.............
  if (0U != gp_csi01_tx_address)
  {
     /* started by writing data to SDR[15:0] */
     SDR01 = (SDR01 & 0xFE00U) | (*gp_csi01_tx_address /* Before */
                                                   /* modification */
     | ((*(gp_csi01_tx_address + 1U) & 0x01U) << 8U)); /* Before */
                                                   /* modification */
     gp_csi01_tx_address += 2U;
  }
.............
}
-----------------------------------------------------------------------
```

After modification:

```
-----------------------------------------------------------------------
MD_STATUS R_CSI01_Send_Receive(uint8_t * const tx_buf,
                               uint16_t tx_num,
                               uint8_t * const rx_buf)
{
.............
  if (0U != gp_csi01_tx_address)
  {
     /* started by writing data to SDR[15:0] */
     SDR01 = (uint16_t)(*gp_csi01_tx_address | (       /* After */
                                                   /*  modification */
      (*(gp_csi01_tx_address + 1U) << 8U) & 0xFF00UL); /* After */
                                                    /* modification */
     gp_csi01_tx_address += 2U;
  }
.............
}
-----------------------------------------------------------------
```

- Example of CSI communications completed interrupt function for channel 1 of unit 0

Before modification:

```
-----------------------------------------------------------------
static void r_csi01_interrupt(void)
{
.............
  if (g_csi01_tx_count > 0U)
  {
    if (0U != gp_csi01_rx_address)
    {
      *gp_csi01_rx_address = (uint8_t)(SDR01 & 0x00FFU);
      *(gp_csi01_rx_address + 1U) =                  /* Before */
                                                    /* modification */
       (uint8_t)((SDR01 & 0x0100U) >> 8U);            /* Before */
                                                    /* modification */
      gp_csi01_rx_address += 2U;
    }
    else
    {
       sio_dummy = SDR01;
    }
    if(0U != gp_csi01_tx_address)
    {
     SDR01 = (SDR01 & 0xFE00U) | (*gp_csi01_tx_address /* Before */
                                                    /* modification */
     | ((*(gp_csi01_tx_address + 1U) & 0x01U) << 8));  /* Before */
                                                    /* modification */
      gp_csi01_tx_address += 2U;
    }
.............
}
------------------------------------------------------------------
```

After modification:

```
------------------------------------------------------------------
static void r_csi01_interrupt(void)
{
.............
  if (g_csi01_tx_count > 0U)
  {
    if (0U != gp_csi01_rx_address)
    {
      *gp_csi01_rx_address = (uint8_t)(SDR01 & 0x00FFUL);
      *(gp_csi01_rx_address + 1U) =                   /* After */
                                        /* modification */
       (uint8_t)((rSDR01 >> 8U) & 0x00FFUL);          /* After */
                                        /* modification */
      gp_csi01_rx_address += 2U;
    }
    else
    {
       sio_dummy = SDR01;
    }
    if(0U != gp_csi01_tx_address)
    {
      SDR01 = (uint16_t)(*gp_csi01_tx_address | (      /* After */
                                        /* modification */
      (*(gp_csi01_tx_address + 1U) << 8U) & 0xFF00UL); /* After */
                                        /* modification */
      gp_csi01_tx_address += 2U;
    }
.............
}
------------------------------------------------------------------
```

(2) In usage as a UART

Modify the data calculation for transmission by the UART sending function R_UARTn_Send(void) in r_cg_serial.c, and the UART transmission and reception completed interrupt functions r_uartn_interrupt_send(void) and r_uartn_interrupt_receive(void) in r_cg_serial_user.c.

n means the channel number.

- Example of a UART sending function for channel 0

Before modification:

```
-----------------------------------------------------------------------

MD_STATUS R_UART0_Send(uint8_t * const tx_buf, uint16_t tx_num)
{
.............
  if ((tx_num < 1U) || ((tx_num & 0x1U) == 1U))
  .............
  else
  {
    .............
    SDR00 = (SDR00 & 0xFE00U) | (*gp_uart0_tx_address /* Before */
                                                  /* modification */
    | ((*(gp_uart0_tx_address + 1U) & 0x01U) << 8U)); /* Before */
                                                  /* modification */
    .............
  }
.............
}
-----------------------------------------------------------------------
```

After modification:

```
-----------------------------------------------------------------------
MD_STATUS R_UART0_Send(uint8_t * const tx_buf, uint16_t tx_num)
{
.............
  if ((tx_num < 1U) || ((tx_num & 0x1U) == 1U))
  .............
  else
  {
    .............
    SDR00 = (uint16_t)(*gp_uart0_tx_address | (       /* After */
                                                  /* modification */
     (*(gp_uart0_tx_address + 1U) << 8U) & 0xFF00UL); /* After */
                                                  /* modification */
    .............
  }
.............
}
-----------------------------------------------------------------------
```

- Example of a UART sending completed interrupt function for channel 0

Before modification:

```
-------------------------------------------------------------------
static void __near r_uart0_interrupt_send(void)
 {
.............
  if (g_uart0_tx_count > 0U)
  {
    SDR00 = (SDR00 & 0xFE00U) | (*gp_uart0_tx_address /* Before */
                                                  /* modification */
    | ((*(gp_uart0_tx_address + 1U) & 0x01U) << 8U)); /* Before */
                                                  /* modification */
    .............
  }
.............
}
-------------------------------------------------------------------
```

After modification:

```
-------------------------------------------------------------------
MD_STATUS R_UART0_Send(uint8_t * const tx_buf, uint16_t tx_num)
{
.............
  if (g_uart0_tx_count > 0U)
  {
    SDR00 = (uint16_t)(*gp_uart0_tx_address | (       /* After */
                                                  /* modification */
     (*(gp_uart0_tx_address + 1U) << 8U) & 0xFF00UL); /* After */
                                                  /* modification */
    .............
  }
.............
}
-------------------------------------------------------------------
```

- Example of a UART reception completed interrupt function
  for channel 0

Before modification:

```
-------------------------------------------------------------------
static void __near r_uart0_interrupt_receive(void)
{
  uint16_t rx_data;
  .............
  if (g_uart0_rx_length > g_uart0_rx_count)
```

```
    {
      *gp_uart0_rx_address = (uint8_t)(rx_data & 0x00FFU);
      *(gp_uart0_rx_address + 1U) =                   /* Before */
                                                 /* modification */
         (uint8_t)((rx_data & 0x0100U) >> 8U);        /* Before */
                                                 /* modification */
      .............
    }
  }
  ----------------------------------------------------------------------
```

After modification:

```
  ----------------------------------------------------------------------
  static void __near r_uart0_interrupt_receive(void)
  {
    uint16_t rx_data;
    .............
    if (g_uart0_rx_length > g_uart0_rx_count)
    {
      *gp_uart0_rx_address = (uint8_t)(rx_data & 0x00FFUL);
      *(gp_uart0_rx_address + 1U) =                    /* After */
                                                 /* modification */
            (uint8_t)((rx_data >> 8U) & 0x00FFUL);    /* After */
                                                 /* modification */
      .............
    }
  }
  ----------------------------------------------------------------------
```

5. Schedule for Fixing the Problem

This problem will be fixed in the next version.

[Disclaimer]

The past news contents have been based on information at the time of publication. Now changed or invalid information may be included. The URLs in the Tool News also may be subject to change or become invalid without prior notice.