RENESAS ツールニュース

RENESAS TOOL NEWS 2016年1月16日：160116/tn5

CS+用 RL78コード生成(CS+ for CC)
CS+用 RL78コード生成(CS+ for CA,CX)
e2 studio Code Generatorプラグイン
RL78コード生成支援ツール Applilet3
ご使用上のお願い

CS+用 RL78コード生成(CS+ for CC)、CS+用 RL78コード生成(CS+ for CA,CX)、 e2 studio (Code Generatorプラグイン) および RL78コード生成支援ツール Applilet3 の使用上の注意事項を連絡します。

- シリアルアレイユニットCSI および UARTで10ビット以上のデータ長を送受信する時の注意事項
  対象: RL78/F12グループ、RL78/F13グループ、RL78/F14グループ、
  RL78/F15グループおよび RL78/D1Aグループ

---

1. 該当製品
   - CS+用 RL78コード生成 (CS+ for CC) V2.08.00以降
   - CS+用 RL78コード生成 (CS+ for CA,CX) V2.08.00以降
   - e2 studio V4.0.1.007以降 (Code Generator プラグイン V2.00.01以降)
   - RL78コード生成支援ツール Applilet3 V1.08.00以降

2. 該当マイコン
   - RL78/F12グループ
   - RL78/F13グループ
   - RL78/F14グループ
   - RL78/F15グループ
   - RL78/D1Aグループ (Applilet3のみサポート)

3. 内容
   シリアルアレイユニットを、3線シリアル(CSI)として使用しデータ長を10ビット
   以上に指定した場合 または UARTとして使用しデータ長を16ビットに指定した

場合、生成コードに誤りがあります。

4. 回避策

r_cg_serial.c および r_cg_serial_user.c にある関数を下記に従い修正してください。なお、コード生成後は常に修正が必要です。

(1) CSIとして使用する場合

r_cg_serial.c にあるCSI送信関数 R_CSImn_Send(void) または
CSI送受信関数 R_CSImn_Send_Receive(void) と、
r_cg_serial_user.c にあるCSI通信完了割り込み関数
r_csimn_interrupt(void) の2つの関数内の送受信時のデータ演算を
修正してください。
注: mはユニット番号、nはチャネル番号を意味します。

- ユニット0、チャネル1を使用したCSI送信受信関数の例

修正前:

```
------------------------------------------------------------------
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 /* 修正前 */
    | ((*(gp_csi01_tx_address + 1U) & 0x01U) << 8U)); /* 修正前 */
    gp_csi01_tx_address += 2U;
  }
.............
}
------------------------------------------------------------------
```

修正後:

```
------------------------------------------------------------------
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 | (       /* 修正後 */
```

```
     (*(gp_csi01_tx_address + 1U) << 8U) & 0xFF00UL); /* 修正後 */
    gp_csi01_tx_address += 2U;
  }
.............
}
-----------------------------------------------------------------
```

- ユニット0、チャネル1を使用したCSI通信完了割り込み関数の例

修正前:

```
-----------------------------------------------------------------
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) =                  /* 修正前 */
       (uint8_t)((SDR01 & 0x0100U) >> 8U);           /* 修正前 */
      gp_csi01_rx_address += 2U;
    }
    else
    {
       sio_dummy = SDR01;
    }
    if(0U != gp_csi01_tx_address)
    {
     SDR01 = (SDR01 & 0xFE00U) | (*gp_csi01_tx_address /* 修正前 */
     | ((*(gp_csi01_tx_address + 1U) & 0x01U) << 8));  /* 修正前 */
      gp_csi01_tx_address += 2U;
    }
.............
}
-----------------------------------------------------------------
```

修正後:

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

(2) UARTとして使用する場合

r_cg_serial.c にあるUART送信関数 R_UARTn_Send(void) 、
r_cg_serial_user.c にあるUART送信完了割り込み関数
r_uartn_interrupt_send(void) とUART受信完了割り込み関数
r_uartn_interrupt_receive(void) の3つの関数内の送受信時の
データ演算を修正してください。
(nはチャネル番号を意味します。)

- チャネル0を使用したUART送信関数の例

修正前:

```
------------------------------------------------------------------
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 /* 修正前 */
    | ((*(gp_uart0_tx_address + 1U) & 0x01U) << 8U)); /* 修正前 */
    ..............
  }
..............
```

```
}
-----------------------------------------------------------------
```

修正後:

```
-----------------------------------------------------------------
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 | (       /* 修正後 */
     (*(gp_uart0_tx_address + 1U) << 8U) & 0xFF00UL); /* 修正後 */
    .............
  }
.............
}
-----------------------------------------------------------------
```

- チャネル0を使用したUART送信完了割り込み関数の例

修正前:

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

修正後:

```
-----------------------------------------------------------------
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 | (      /* 修正後 */
     (*(gp_uart0_tx_address + 1U) << 8U) & 0xFF00UL); /* 修正後 */
    ..............
  }
..............
}
```

- チャネル0を使用したUART受信完了割り込み関数の例

修正前:

```
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) =                  /* 修正前 */
      (uint8_t)((rx_data & 0x0100U) >> 8U);        /* 修正前 */
    ..............
  }
}
```

修正後:

```
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) =                  /* 修正後 */
          (uint8_t)((rx_data >> 8U) & 0x00FFUL);   /* 修正後 */
    ..............
  }
}
```

---

5. 恒久対策

次期バージョンで改修する予定です。

[免責事項]
過去のニュース内容は発行当時の情報をもとにしており、現時点では変更された情報や無効な情報が含まれている場合があります。
ニュース本文中のURLを予告なしに変更または中止することがありますので、あらかじめご承知ください。