[Notes]
CS+ Code Generator for RL78 (CS+ for CC),
CS+ Code Generator for RL78 (CS+ for CA, CX),
e^2 studio Code Generator Plug-in,
Applilet3 Coding Assistance Tool for RL78,
AP4 Coding Assistance Tool for RL78

R20TS0459EJ0101
Rev.1.01
Aug. 01, 2019

Outline

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

1. When using IICA0 or IICA1 as a Single Master System
2. When using the R_ADC_Set_ADChannel function in the A/D converter

1. When Using IICA0 or IICA1 as a Single Master System

1.1 Applicable Products

- CS+ Code Generator for RL78 V3.02.00 (CS+ for CC V3.00) or later
- CS+ Code Generator for RL78 V3.02.00 (CS+ for CA,CX V3.00) or later
- Code Generator plug-in V1.0.1 (e^2 studio V2.2.0) or later
- Applilet3 for RL78 V1.01.00 or later
- AP4 for RL78 V1.00.00 or later

1.2 Applicable Devices

- RL78 family:
  RL78/F12, RL78/F13, RL78/F14, RL78/F15,
  RL78/G10, RL78/G11,
  RL78/G1A, RL78/G1C, RL78/G1D, RL78/G1F, RL78/G1H,
  RL78/H1D,
  RL78/I1A, RL78/I1B, RL78/I1C,
  RL78/L12, RL78/L13, RL78/L1A and RL78/L1C groups

1.3 Details

When using the Serial Interface IICA0 (IICA0) or IICA1 (IICA1) as a single master system (*), incorrect code will be generated. Because the order of the bit setting (below) differs from that in User's Manual: Hardware of applicable devices, it may affect the operation of the ‘control of wait and interrupt request generation of serial interface IICA’.

* Displayed as ‘Single Master’ in the code generator GUI.

- For IICA0:

  IICCTL00 Register: WTIM0 and WREL0 bit

- For IICA1:

  IICCTL10 Register: WTIM1 and WREL1 bit

1.4 Conditions

The error occurs when using the Serial Interface IICA0 (IICA0) or IICA1 (IICA1) as a single master system. This error does not occur in Slave Mode.

1.5 Workaround

Fix the code to correct the order of the bit setting.

Note: When code is generated again, generated code returns to the state before correction. Therefore, correct the source file each time you generate code.

- When using 'IICA0' as a single master
  After code is generated, open iica0_masterhandler(void), and then modify the code shown in the red box below.

Before modification

```
static void iica0_masterhandler(void)
{
        /* Master receive control */
        else
        {
            if (g_iica0_rx_cnt < g_iica0_rx_len)
            {
                *gp_iica0_rx_address = IICA0;
                gp_iica0_rx_address++;
                g_iica0_rx_cnt++;

                if (g_iica0_rx_cnt == g_iica0_rx_len)
                {
                    ACKE0 = 0U;
                    WREL0 = 1U;
                    WTIM0 = 1U;
```

After modification

```
static void iica0_masterhandler(void)
{
        /* Master receive control */
        else
        {
            if (g_iica0_rx_cnt < g_iica0_rx_len)
            {
                *gp_iica0_rx_address = IICA0;
                gp_iica0_rx_address++;
                g_iica0_rx_cnt++;

                if (g_iica0_rx_cnt == g_iica0_rx_len)
                {
                    ACKE0 = 0U;
                    WTIM0 = 1U;
                    WREL0 = 1U;
```

➢ When using 'IICA1' as a single master
After code is generated, open iica1_master_handler(void), and then modify the code shown in the red box below.

Before modification

```
static void iica1_master_handler(void)
{
        /* Master receive control */
        else
        {
            if (g_iica1_rx_cnt < g_iica1_rx_len)
            {
                *gp_iica1_rx_address = IICA1;
                gp_iica1_rx_address++;
                g_iica1_rx_cnt++;

                if (g_iica1_rx_cnt == g_iica1_rx_len)
                {
                    ACKE1 = 0U;
                    WREL1 = 1U;
                    WTIM1 = 1U;
```

After modification

```
static void iica1_master_handler(void)
{
        /* Master receive control */
        else
        {
            if (g_iica1_rx_cnt < g_iica1_rx_len)
            {
                *gp_iica1_rx_address = IICA1;
                gp_iica1_rx_address++;
                g_iica1_rx_cnt++;

                if (g_iica1_rx_cnt == g_iica1_rx_len)
                {
                    ACKE1 = 0U;
                    WTIM1 = 1U;
                    WREL1 = 1U;
```

1.6 Schedule for Fixing the Problem

For RL78/F12, RL78/F13, RL78/F14, RL78/F15, RL78/G10, and RL78/G11 groups, this problem will be fixed in the next version. (Scheduled to be released in January 2020.)

For groups that are not listed above, no fixes are scheduled.

2. When Using the R_ADC_Set_ADChannel Function in the A/D Converter

2.1 Applicable Products

- CS+ Code Generator for RL78 V1.04.00/V3.02.00 (CS+ for CC V3.00) or later
- CS+ Code Generator for RL78 V1.04.00/V3.02.00 (CS+ for CA,CX V3.00) or later
- Code Generator plug-in V1.0.1 (e^2 studio V2.2.0) or later
- Applilet3 for RL78 V1.00.00 or later
- AP4 for RL78 V1.04.00 or later

2.2 Applicable Devices

- RL78/D1A group: 48- and - 64-pin products
- RL78/G1A group: 25- and 32- pin products
- RL78/G1F group: 24-pin products
- RL78/I1D group: 48-pin products

2.3 Details

If you select [Used] for [A/D converter operation setting], inappropriate enum definitions are generated for channel definitions for the A/D converter. For this reason, A/D conversion channels may not be changed correctly by the R_ADC_Set_ADChannel function that uses enum definitions.

Examples of inappropriate enum definitions (RL78/G1A group: 32-pin products)

In the following enum definitions, ADCHANNEL26 does not have a required definition of the correct value (initial value = 26). Therefore, ADCHANNEL26 results in an inappropriate value (25). This further causes ADCHANNEL27 to ADCHANNEL29 to result in inappropriate values.

```
/************************************************************************************************
Typedef definitions
************************************************************************************************/
typedef enum
{
    ADCHANNEL0, ADCHANNEL1, ADCHANNEL2, ADCHANNEL3, ADCHANNEL4, ADCHANNEL16 = 16U, ADCHANNEL17,
    ADCHANNEL18, ADCHANNEL19, ADCHANNEL20, ADCHANNEL21, ADCHANNEL22, ADCHANNEL23, ADCHANNEL24,
    ADCHANNEL26, ADCHANNEL27, ADCHANNEL28, ADCHANNEL29, ADTEMPERSENSOR0 = 128U, ADINTERREFVOLT
} ad_channel_t;
```

2.4 Conditions

The problem occurs when the R_ADC_Set_ADChannel function makes a change to one of the following A/D conversion channels.

Devices	A/D conversion channels which cannot be changed correctly
RL78/D1A group: 48- and 64-pin products	ADCHANNEL7
RL78/G1A group: 25-pin products	ADCHANNEL20, 21, 25, 26, 27 and 29
RL78/G1A group: 32-pin products	ADCHANNEL26, 27, 28 and 29
RL78/G1F group: 24-pin products	ADCHANNEL20 and 21 and ADPGA0
RL78/I1D group: 48-pin products	ADCHANNEL16, 17 and 18

2.5 Workaround

When changing an A/D conversion channel by the R_ADC_Set_ADChannel function, specify a literal instead of an enum definition.

Example: Specifying ADCHANNEL7

```
R_ADC_Set_ADChannel((ad_channel_t) 7 );
```

2.6 Schedule for Fixing the Problem

This problem will be fixed in the next version. (Scheduled to be released in January 2020.)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Aug.01.19	-	First edition issued
1.01	Dec.23.19	1	Corrected 1.2 Applicable Devices
		3	Corrected 1.6 Workaround

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

TS Colophon 4.0