[Notes]
e^2 studio Smart Configurator Plug-in,
Smart Configurator for RX

R20TS0591ES0100
Rev.1.00
Jun. 16, 2020

Outline

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

1. When using Data Transfer Controller (DTC) component and making configuration for its vector base address
2. When using SCI/SCIF Asynchronous Mode component and making configuration for its bit-rate
3. When using AN007 or AN107 as analog input pins in S12AD components

1. When Using Data Transfer Controller (DTC) Component and Making Configuration for Its Vector Base Address

1.1 Applicable Products

- e^2 studio V6.2.0 (Smart Configurator Plug-in V1.3.0) or later
- Smart Configurator for RX V1.3.0 or later

1.2 Applicable Devices

- RX Family:
  RX230, RX231 groups (Products with 32Kbytes RAM capacity only)
  RX651, RX65N groups (Products with 640Kbytes RAM capacity only)

1.3 Details

When configuring the vector base address in Data Transfer Controller (DTC) component, because the address range for this vector base address is determined incorrectly, error marks do not appear properly.

- In case of RX230, RX231 groups (Products with 32Kbytes RAM capacity only)

  Correct address range: 0x00000000~0x00007C00

  Incorrect address range: 0x00000000~0x0000FC00 (See Figure1.1)

- In case of RX651, RX65N groups (Products with 640Kbytes RAM capacity only)

  Correct address range: 0x00000000~0x0003FC00 and 0x00800000~0x0085FC00

  Incorrect address range: 0x00000000~0x0003FC00 only (See Figure1.2)

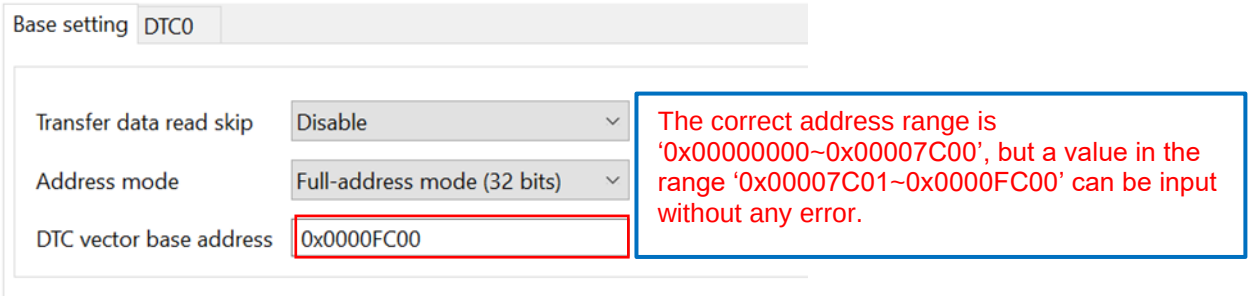


Figure 1.1 DTC vector base address determined based on the incorrect address range (RX230, RX231 groups)

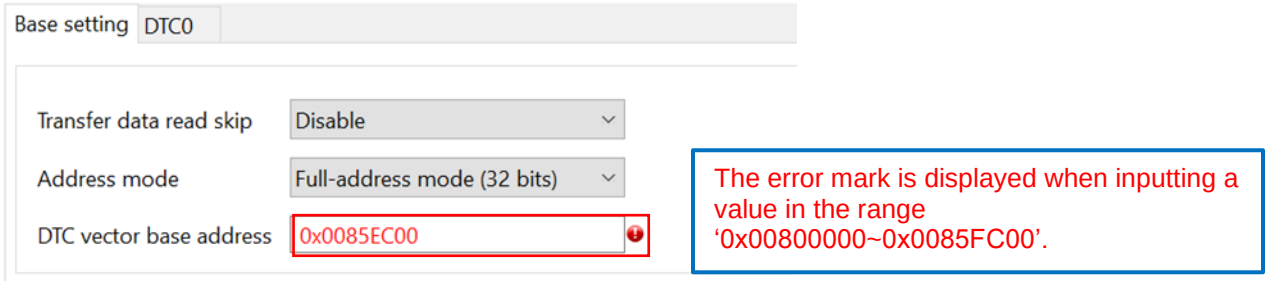


Figure 1.2 DTC vector base address determined based on the incorrect address range (RX651, RX65N groups)

1.4 Workaround

- In case of RX230, RX231 groups (Products with 32Kbytes RAM capacity)

  Manually correct the DTC vector base address to 0x00007C00 or smaller.

- In case of RX651, RX65N groups (Products with 640Kbytes RAM capacity)

  Ignore the error mark besides the DTC vector base address textbox if the input address value falls in the range '0x00800000 ~ 0x0085FC00' and is in 1-KBytes unit.

1.5 Schedule for Fixing the Problem

This problem will be fixed in the following versions. (Scheduled to be released in July 2020.)

- e² studio 2020-07
- Smart Configurator for RX V2.6.0

2. When Using SCI/SCIF Asynchronous Mode Component and Making Configuration for Its Bit-Rate

2.1 Applicable Products

- e² studio V6.0.0 (Smart Configurator Plug-in V1.2.0) or later
- Smart Configurator for RX V1.2.0 or later

2.2 Applicable Devices

- RX Family:
  RX651, RX65N groups

2.3 Details

When making configuration for the bit-rate on ACI/SCIF Asynchronous Mode component by using the textbox, if the value is within the input range but smaller than 8 times of the minimum value, then the generated codes for bit-rate setting are incorrect.

Example: When using SCI channel 0 and inputting 500 bps in the bit-rate textbox

Input value 500 bps is within the input range (114.441~7500000.0) but smaller than 8 times of minimum value (8*114.441=915.528). Therefore, the generated codes for bit-rate setting are incorrect.

GUI configuration (Figure 2.1) and incorrect generated code (Figure 2.2) are shown below.

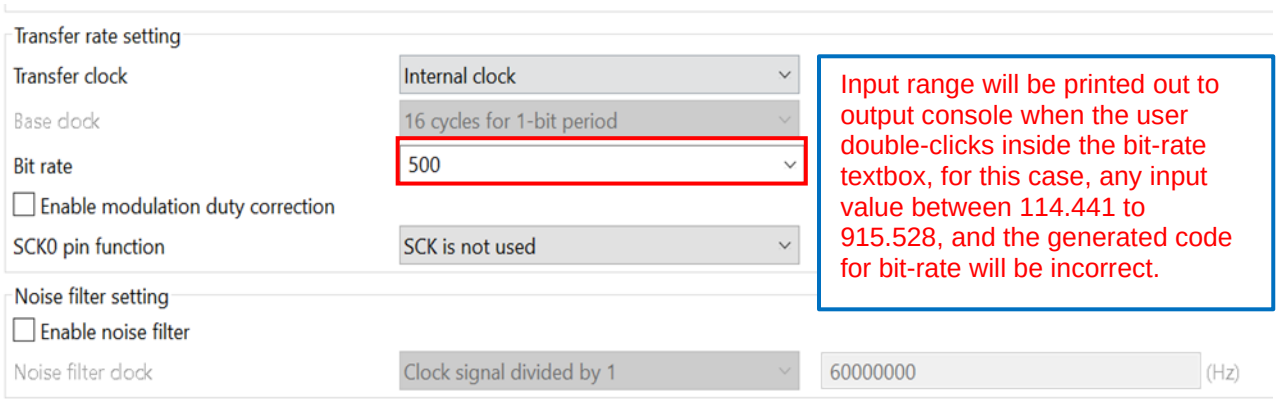


Figure 2.1 GUI configuration for bit-rate on SCI/SCIF Asynchronous Mode

```
/***********************************************************************
* Function Name: R_Config_SCI0_Create
* Description  : This function initializes the SCI0 channel
* Arguments    : None
* Return Value : None
***********************************************************************/

void R_Config_SCI0_Create(void)
{
    /* Cancel SCI stop state */
    MSTP(SCI0) = 0U;
    ......

    /* Set control registers */
    SCI0.SMR.BYTE = _00_SCI_CLOCK_PCLK | _00_SCI_MULTI_PROCESSOR_DISABLE |
_00_SCI_STOP_1 | _00_SCI_PARITY_DISABLE | _00_SCI_DATA_LENGTH_8 |
_00_SCI_ASYNCHRONOUS_OR_I2C_MODE;
    ......

    SCI0.SEMR.BYTE = _00_SCI_BIT_MODULATION_DISABLE | _00_SCI_16_BASE_CLOCK |
_00_SCI_NOISE_FILTER_DISABLE | _00_SCI_BAUDRATE_SINGLE |
_00_SCI_LOW_LEVEL_START_BIT;

    /* Set bit rate */
    SCI0.BRR = 0x2EU;

    ......

    R_Config_SCI0_Create_UserInit();
}
```

CKS value is wrong and it should be '_03_SCI_CLOCK_PCLK_64'

ABCS value is wrong and it should be '_10_SCI_8_BASE_CLOCK'

BRR value is wrong and it should be 'SCI0.BRR = 0x74U'

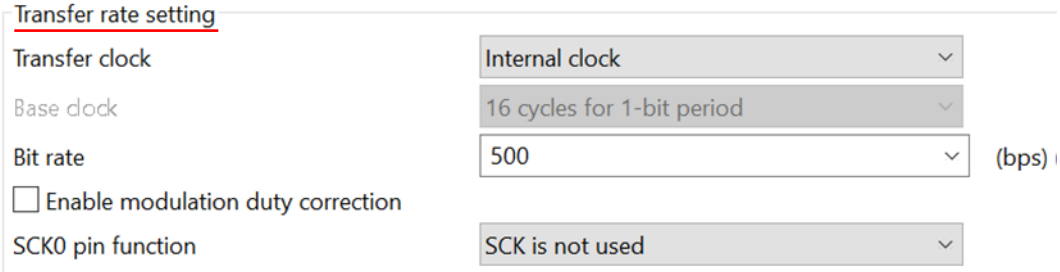
Figure 2.2 Generated code for bit-rate on SCI/SCIF Asynchronous Mode

2.4 Workaround

Follow the steps below by referring to the generated codes for SCI/SCIF Asynchronous Mode component for RX64M project.

(1) Create a Smart Configurator RX64M project and add SCI/SCIF Asynchronous Mode component. Set resource to any one from SCI0 ~ 7, or SCI12.

(2) Configure settings for items within the "Transfer rate setting" group to be same as Smart Configurator RX651/N project*1.



(3) Generate codes and copy SMR.CKS and SEMR.ABCS macro values and BRR register value in the RX651/N project initialization API and replaces the existing values (See Figure 2.2) respectively.

*1: Make sure that PCLK frequency setting values on the clock page for RX651/N and RX64M project are same, for RX651/N, SCI0~9, SCI12 use PLCKB, SCI10 and SCI11 use PCLKA;
for RX64M, all channels use PCLKB.

2.5 Schedule for Fixing the Problem

This problem will be fixed in the following versions. (Scheduled to be released in July 2020.)

- e^2 studio 2020-07
- Smart Configurator for RX V2.6.0

3. When Using AN007 or AN107 as Analog Input Pins in S12AD Components

3.1 Applicable Products

- e² studio V7.2.0 (Smart Configurator Plug-in V1.5.0) or later
- Smart Configurator for RX V1.5.0 or later

3.2 Applicable Devices

- RX Family:
  RX66T, RX72T groups

3.3 Details

When using AN007 or AN107 as analog input pins, although PxDEN bits*1 are supposed to be cleared to 0, incorrect code is generated. Therefore, AN007 and AN107 cannot be used as analog input pins.

*1: x=000 to 002 for AN007, x=100 to 102 for AN107

Below is an example of when using AN007 as an analog input pin.

In case of AN107, read “AN007” as “AN107” below.

■ Example 1

・Conditions

Item		Settings
AN001 and AN002	Pin settings	Analog input pins
	Analog input path	Any one of the top 4 selections from its combo box
Amplifier input setting		Single-ended

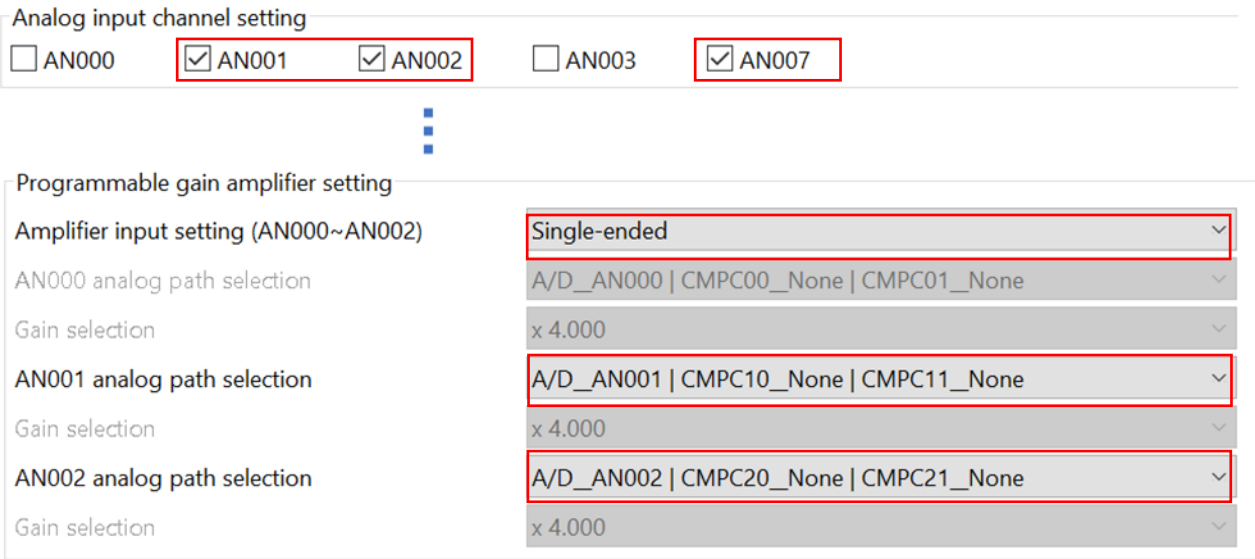


Figure 3.1 GUI settings for Example 1

・Generated code

```
/***********************************************************************
* Function Name: R_Config_S12AD0_Create
* Description  : This function initializes the S12AD0 channel
* Arguments    : None
* Return Value : None
***********************************************************************/

void R_Config_S12AD0_Create(void)
{
    /* Cancel S12AD0 module stop state */
    MSTP(S12AD) = 0U;
    .....

    S12AD.ADANSA0.WORD = _0002_AD_ANx01_USED | _0004_AD_ANx02_USED |
_0080_AD_ANx07_USED;
    S12AD.ADADS0.WORD = _0080_AD_ANx07_

    /* Set AN001 amplifier */
    S12AD.ADPGADCR0.BIT.P001DEN = 0U;
    S12AD.ADPGACR.BIT.P001CR = _0001_AD_PATH_ANx_NONE_NONE;

    /* Set AN002 amplifier */
    S12AD.ADPGADCR0.BIT.P002DEN = 0U;
    S12AD.ADPGACR.BIT.P002CR = _0001_AD_PATH_ANx_NONE_NONE;
    S12AD.ADCER.WORD = _0000_AD_AUTO_CLEARING_DISABLE |
_0000_AD_SELFTDIAGST_DISABLE | _0000_AD_RIGHT_ALIGNMENT;
    S12AD.ADELCCR.BYTE = _02_ALL_SCAN_COMPLETION;
    ......

    R_Config_S12AD0_Create_UserInit();
}
```

Code to clear P000DEN bit:
'S12AD.ADPGADCR0.BIT.P000DEN = 0U' is missing

Figure 3.2 Generated code for Example 1

■ Example 2

・Conditions

Item		Settings
AN001	Pin settings	Analog input pins
	Analog input path	Any one of the top 4 selections from its combo box
Amplifier input setting		Single-ended

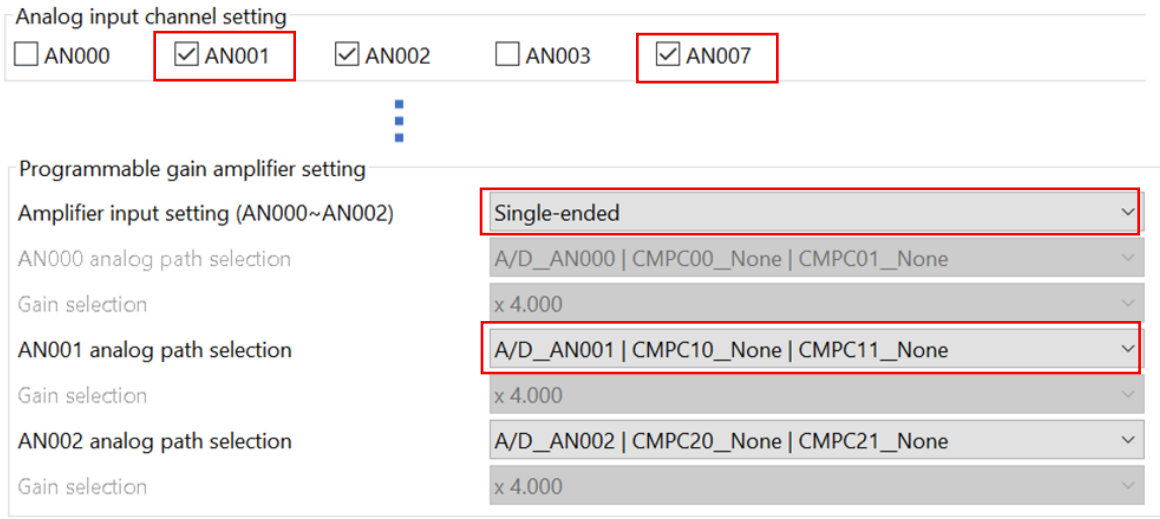


Figure 3.3 GUI settings for Example 2

・Generated code

```
/***********************************************************************
* Function Name: R_Config_S12AD0_Create
* Description  : This function initializes the S12AD0 channel
* Arguments    : None
* Return Value : None
***********************************************************************/

void R_Config_S12AD0_Create(void)
{
    /* Cancel S12AD0 module stop state */
    MSTP(S12AD) = 0U;
    .....

    S12AD.ADANSA0.WORD = _0002_AD_ANx01_USED | _0080_AD_ANx07_USED;
    S12AD.ADADS0.WORD = _0080_AD_ANx07

    /* Set AN001 amplifier */
    S12AD.ADPGADCR0.BIT.P001DEN = 0U;
    S12AD.ADPGACR.BIT.P001CR = _0001_A

    /* Set compare control register */
    S12AD.ADCMPCR.WORD = _0000_AD_WIND
_0000_AD_WINDOWFUNCTION_DISABLE;
    ......

    R_Config_S12AD0_Create_UserInit();
}
```

Code to clear P000DEN bit: 'S12AD.ADPGADCR0.BIT.P000DEN = 0U' is missing

Code to clear P000DEN bit: 'S12AD.ADPGADCR0.BIT.P002DEN = 0U' is missing

Figure 3.4 Generated code for Example 2

■ Example 3

・Conditions

Item		Settings
AN002	Pin settings	Analog input pins
	Analog input path	Any one of the top 4 selections from its combo box
Amplifier input setting		Single-ended

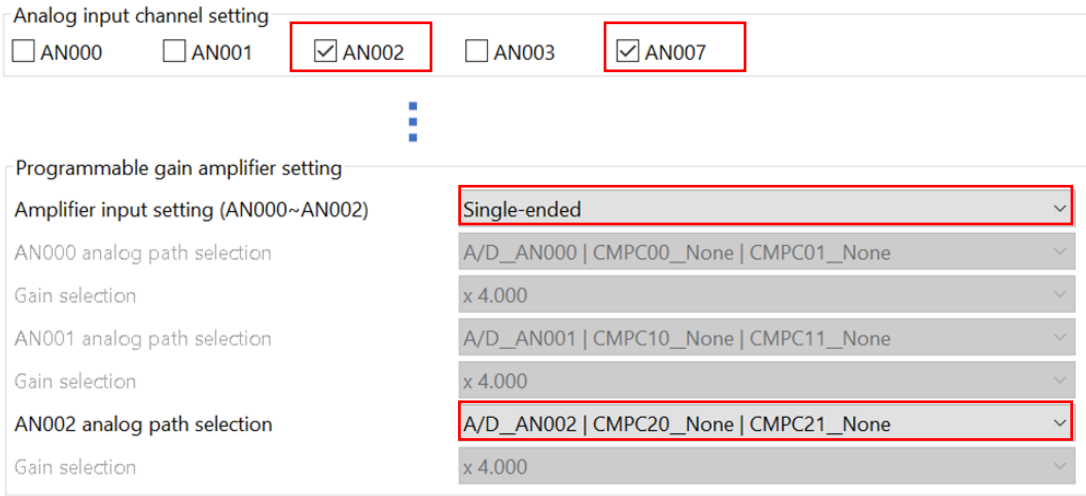


Figure 3.5 GUI settings for Example 3

・Generated code

```
/***********************************************************************
* Function Name: R_Config_S12AD0_Create
* Description  : This function initializes the S12AD0 channel
* Arguments    : None
* Return Value : None
***********************************************************************/

void R_Config_S12AD0_Create(void)
{
    /* Cancel S12AD0 module stop state */
    MSTP(S12AD) = 0U;
    .....

    S12AD.ADANSA0.WORD = _0004_AD_ANx02_USED | _0080_AD_ANx07_USED;
    S12AD.ADADS0.WORD = _0080_AD_ANx07_

    /* Set AN002 amplifier */
    S12AD.ADPGADCR0.BIT.P002DEN = 0U;
    S12AD.ADPGACR.BIT.P002CR = _0001_AD
    S12AD.ADCER.WORD = _0000_AD_AUTO_CLEARING_DISABLE |
_0000_AD_SELFTDIAGST_DISABLE | _0000_AD_RIGHT_ALIGNMENT;
    S12AD.ADELCCR.BYTE = _02_ALL_SCAN_COMPLETION;
    S12AD.ADCSR.WORD |= _1000_AD_SCAN_END_INTERRUPT_ENABLE;
    ......

    R_Config_S12AD0_Create_UserInit();
}
```

Codes to clear P000DEN and P001DEN bit: 'S12AD.ADPGADCR0.BIT.P000DEN = 0U' and 'S12AD.ADPGADCR0.BIT.P001DEN = 0U' are missing.

Figure 3.6 Generated code for Example 3

3.4 Workaround

When using AN007 or AN107 as analog input pins, manually add code to clear PxDEN bits[*1] 'S12AD.ADPGADCR0.BIT.PxDEN = 0U' to the generated file.

*1: x=000 to 002 for AN007, x=100 to 102 for AN107

- Source file: "<*Configuration name*>.c"
- Function: "void R_<*configuration-name*>_Create(void)"

The <*Configuration name*> varies depending on the selected component of S12AD.

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

The following is an example of modification when <*configuration-name*> is Config_S12AD0 (default).

■ Modification example for Example 1 in section 3.3

```
/***********************************************************************
* Function Name: R_Config_S12AD0_Create
* Description  : This function initializes the S12AD0 channel
* Arguments    : None
* Return Value : None
***********************************************************************/

void R_Config_S12AD0_Create(void)
{
    /* Cancel S12AD0 module stop state */
    MSTP(S12AD) = 0U;
    .....

    S12AD.ADANSA0.WORD = _0002_AD_ANx01_USED | _0004_AD_ANx02_USED |
_0080_AD_ANx07_USED;
    S12AD.ADADS0.WORD = _0080_AD_ANx07_ADD_USED;

    S12AD.ADPGADCR0.BIT.P000DEN = 0U;   Add a new line of code to clear P000DEN bit

    /* Set AN001 amplifier */
    S12AD.ADPGADCR0.BIT.P001DEN = 0U;
    S12AD.ADPGACR.BIT.P001CR = _0001_AD_PATH_ANx_NONE_NONE;

    /* Set AN002 amplifier */
    S12AD.ADPGADCR0.BIT.P002DEN = 0U;
    S12AD.ADPGACR.BIT.P002CR = _0001_AD_PATH_ANx_NONE_NONE;
    S12AD.ADCER.WORD = _0000_AD_AUTO_CLEARING_DISABLE |
_0000_AD_SELFTDIAGST_DISABLE | _0000_AD_RIGHT_ALIGNMENT;
    S12AD.ADELCCR.BYTE = _02_ALL_SCAN_COMPLETION;
    ......

    R_Config_S12AD0_Create_UserInit();
}
```

Figure 3.7 Code modification for Example 1 in section 3.3

■ Modification example for Example 2 in section 3.3

```
/***********************************************************************
* Function Name: R_Config_S12AD0_Create
* Description  : This function initializes the S12AD0 channel
* Arguments    : None
* Return Value : None
***********************************************************************/

void R_Config_S12AD0_Create(void)
{
    /* Cancel S12AD0 module stop state */
    MSTP(S12AD) = 0U;
    .....

    S12AD.ADANSA0.WORD = _0002_AD_ANx01_USED | _0080_AD_ANx07_USED;
    S12AD.ADADS0.WORD = _0080_AD_ANx07_ADD_USED;

    S12AD.ADPGADCR0.BIT.P000DEN = 0U;

    /* Set AN001 amplifier */
    S12AD.ADPGADCR0.BIT.P001DEN = 0U;
    S12AD.ADPGACR.BIT.P001CR = _0001_AD_PATH_ANx_NONE_NONE;

    S12AD.ADPGADCR0.BIT.P002DEN = 0U;

    /* Set compare control register */
    S12AD.ADCMPCR.WORD = _0000_AD_WINDOWB_DISABLE | _0000_AD_WINDOWA_DISABLE |
_0000_AD_WINDOWFUNCTION_DISABLE;
    ......

    R_Config_S12AD0_Create_UserInit();
}
```

Add a new line of code to clear P000DEN bit

Add a new line of code to clear P002DEN bit

Figure 3.8 Code modification for Example 2 in section 3.3

■ Modification example for Example 3 in section 3.3

```
/***********************************************************************
* Function Name: R_Config_S12AD0_Create
* Description  : This function initializes the S12AD0 channel
* Arguments    : None
* Return Value : None
***********************************************************************/

void R_Config_S12AD0_Create(void)
{
    /* Cancel S12AD0 module stop state */
    MSTP(S12AD) = 0U;
    .....

    S12AD.ADANSA0.WORD = _0004_AD_ANx02_USED | _0080_AD_ANx07_USED;
    S12AD.ADADS0.WORD = _0080_AD_ANx07_ADD_USED;

    S12AD.ADPGADCR0.BIT.P000DEN = 0U;
    S12AD.ADPGADCR0.BIT.P001DEN = 0U;

    /* Set AN002 amplifier */
    S12AD.ADPGADCR0.BIT.P002DEN = 0U;
    S12AD.ADPGACR.BIT.P002CR = _0001_AD_PATH_ANx_NONE_NONE;
    S12AD.ADCER.WORD = _0000_AD_AUTO_CLEARING_DISABLE |
_0000_AD_SELFTDIAGST_DISABLE | _0000_AD_RIGHT_ALIGNMENT;
    S12AD.ADELCCR.BYTE = _02_ALL_SCAN_COMPLETION;
    S12AD.ADCSR.WORD |= _1000_AD_SCAN_END_INTERRUPT_ENABLE;
    ......

    R_Config_S12AD0_Create_UserInit();
}
```

Add a new line of code to clear P000DENP000DEN bit

Figure 3.9 Code modification for Example 3 in section 3.3

3.5 Schedule for Fixing the Problem

This problem will be fixed in the following versions: (Scheduled to be released in July 2020.)

- e² studio 2020-07
- Smart Configurator for RX V2.6.0

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jun.16.20	-	First edition issued

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/

TS Colophon 4.1