Introduction

This application note describes how to replace the programs for R8C with the programs for RL78.

Target Device

R8C Family

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.
Contents

1. How to Replace Programs from R8C Family to RL78 Family ........................................... 3

2. Program Conversion Using CcnvNC30 .................................................................................. 3
   2.1 About CcnvNC30 .......................................................................................................... 3
   2.2 How to Use CcnvNC30 .................................................................................................. 4
   2.3 When CcnvNC30 is Not Used ....................................................................................... 6

3. Converting Programs for Peripheral Functions ................................................................. 7
   3.1 Generating Programs Automatically ............................................................................. 7
   3.2 Adding Programs .......................................................................................................... 9
   3.3 When Code Generator is Not Used ............................................................................... 9

4. Replacement Examples ....................................................................................................... 10
   4.1 R8C Sample Program (Clock Operation Using RTC) .................................................... 10
      4.1.1 Porting Source to CC-RL with CcnvNC30 ............................................................... 10
      4.1.2 Generating Programs Automatically ..................................................................... 12
      4.1.3 Adding Programs .................................................................................................. 14
      4.1.4 Other Items to be Corrected .................................................................................. 18
      4.1.5 Sample Code After Replacement ............................................................................ 19
   4.2 R8C Sample Program (DTC Operation) ....................................................................... 20
      4.2.1 Porting Source to CC-RL with CcnvNC30 ............................................................... 20
      4.2.2 Generating Programs Automatically ..................................................................... 21
      4.2.3 Adding Programs .................................................................................................. 24
      4.2.4 Other Modifications ............................................................................................... 25
      4.2.5 Sample Code After Replacement ............................................................................ 26
   4.3 Conditions for Confirming Operations of Sample Programs ....................................... 27

5. Sample Code ...................................................................................................................... 28

6. Reference Documents ....................................................................................................... 28
1. How to Replace Programs from R8C Family to RL78 Family

   This section explains how to replace the programs for R8C family with the programs for RL78 family.
   First, use the C source converter CcnvNC30 to convert the extended functions for the C compiler NC30 to the
   extended functions for the C compiler CC-RL.
   Next, use the integrated development environment CS+ or e2studio to create a project. Because the R8C family and
   the RL78 family have different peripheral functions, use the code generator for the RL78 family to generate programs
   for peripheral functions of the RL78 family instead of using the programs for peripheral functions of the R8C family.
   Combine the programs converted with CcnvNC30 and the above programs for peripheral functions to replace
   programs.

2. Program Conversion Using CcnvNC30

2.1 About CcnvNC30

   CcnvNC30 converts extended language specifications (such as macro names, reserved words, #pragma directives,
   and extended functions) in C source programs for NC30 into extended language specifications for CC-RL.
   CcnvNC30 is the software that supports the porting of the programs for NC30 to the programs for CC-RL. Since we
do not guarantee the correct operation of the programs converted by CcnvNC30, be sure to check the operation of the
   program after conversion.
   In addition, the device-dependent codes such as location addresses, access to an SFR, and assembly-language codes
   cannot be converted. Convert these codes manually into the code for the RL78 family as required.
   For details, see "CcnvNC30 C Source Converter User's Manual (R20UT3685E)".
2.2 How to Use CcnvNC30

The method of converting a program with CcnvNC30 is shown below.

1. Place CcnvNC30 (CcnvNC30.exe) and a program for NC30 in the same folder of your choice.
2. Launch Command Prompt in Windows.
3. Change the current directory to the folder where CcnvNC30 is stored.

(4) Specify an output file name with the -o option before execution. After the execution, a program for CC-RL is output. In addition, when outputting messages in a specified file, use the -r option.
When converting multiple files at the same time, create a list file and execute conversion with the -l option specified. After the execution, programs for CC-RL are output to the specified folder.

Figure 2.3 CcnvNC30 Execution Window (Multiple Files)

The example below shows the description in a list file.

Figure 2.4 Example of Description in List File
(6) Correct the parts that are not converted by CcnvNC30. For the parts that require corrections, refer to "CONVERSION SPECIFICATIONS" of "CcnvNC30 C source converter User's Manual (R20UT3685E)".

2.3 When CcnvNC30 is Not Used

When CcnvNC30 is not used, extended functions of NC30 need to be converted manually into extended functions of CC-RL. For the extended language specifications supported by CC-RL, see "CC-RL Compiler User's Manual (R20UT3123E)".
3. Converting Programs for Peripheral Functions

3.1 Generating Programs Automatically

Programs are automatically generated for the RL78 family peripheral functions equivalent to the peripheral functions that were used by the R8C family by using the code generator for the RL78 family provided in the integrated development environment CS+ or e2studio. For how to use the code generator, see "CS+ Code Generator Integrated Development Environment User’s Manual: Peripheral Function Operation [CS+ for CC][CS+ for CA,CX] (R20UT3104E)".

(1) Under [Project Tree], click [Clock Generator] in [Code Generator (Design Tool)] and perform “pin assignment”. When the pin assignment setting is decided once, it is not possible to be changed it later.

(2) Refer to the program for the R8C family and set each function.
(3) On completion of all the peripheral function settings, click the [Generate Code] button at the top of the window to generate codes (automatic program generation). Use the automatically generated functions for peripheral functions to replace programs.

![Code Generator Setting Window](image)

**Figure 3.2 Code Generator Setting Window (2)**
3.2 Adding Programs

Add the programs that cannot be automatically generated by the code generator (such as main function, interrupt function process, and variables).

Add a program between "/* Start user code for adding. Do not edit comment generated here */" and "/* End user code. Do not edit comment generated here */" in each file that was automatically generated. A program needs to be added manually. Note that any program added outside this range is automatically deleted during automatic generation of a program.

Be sure to confirm the operation of the system using the added programs.

3.3 When Code Generator is Not Used

When the code generator is not used, you need to create a new project first with the integrated development environment CS+ or e2studio and then manually create a program for a peripheral function. For details of peripheral functions, see the user’s manual for the RL78 family.
Replacement Guide for R8C Family to RL78 Family (CcnvNC30)

4. Replacement Examples

4.1 R8C Sample Program (Clock Operation Using RTC)

The program of "R8C/35A Group Clock Operation Using RTC (R01AN0079E)" is replaced with the program for RL78/G14. The project file after replacement is "r01an3508_rl78g14_rtc".

This program uses timer RE (Real-Time Clock Mode) to operate a clock. Use the 20 MHz XIN clock for the CPU clock. Use fC4 (XCIN clock (32.768 kHz) divided by 4) for the timer RE count source.

4.1.1 Porting Source to CC-RL with CcnvNC30

1) Create a list file to specify a C source file to be converted.

```
|c=sjis r01an0079_src.c output\rtc.c
```

Figure 4.1 Example of Description in List File (Clock Operation Using RTC)

2) Launch Command Prompt to convert the C source file specified with the list file. In addition, the output conversion result file indicates changes.

```
c:\CcnvNC30\CcnvNC30 -l=listfile_rtc.txt -r=output\rtc.txt
```

Figure 4.2 CcnvNC30 Execution Window (Clock Operation Using RTC)

The conversion result file indicates the conversion result as shown below. For details of the conversion result, see "CcnvNC30 C source converter User's Manual (R20UT3685E)".

```
CcnvNC30 C Source Converter V1.00.00.01 [07 Mar 2016]
r01an0079_src.c[106]:N0593124:[Info]The language specification dependent on R3C or M6C.
r01an0079_src.c[111]:N0593124:[Info]The language specification dependent on R8C or M16C.
r01an0079_src.c[116]:N0593146:[Info]The language specification dependent on R8C or M16C.
r01an0079_src.c[116]:N0593146:[Info]The language specification dependent on R8C or M16C.
r01an0079_src.c[120]:N0593124:[Info]The language specification dependent on R8C or M16C.
r01an0079_src.c[120]:N0593124:[Info]The language specification dependent on R8C or M16C.
r01an0079_src.c[120]:N0593124:[Info]The language specification dependent on R8C or M16C.
r01an0079_src.c[120]:N0593124:[Info]The language specification dependent on R8C or M16C.
n:
```
Figure 4.3 Details of Conversion Result (Clock Operation Using RTC)
(3) Correct the converted C source file.

There may be redundant interrupt function declarations. As CC-RL produces an error in this case, delete the converted #pragma directive.

```c
// [CnvNC30] #pragma interrupt/B _timer_re(vect-10)
#pragma interrupt _timer_re(vect-10, bank-RB1)
void _timer_re(void)
{
  sec = tsec & 0x7f;  /* Set second to RAM */
  min = tmin & 0x7f;  /* Set minute to RAM */
  hr = tchr & 0x3f;  /* Set hour to RAM */
  wk_old = wk;       /* Set last-time value */
  wk = tewk & 0x07;  /* Set day to RAM */
  up_flg = UPDATE;   /* Set update flag */
}
```

Delete the #pragma directive.

**Figure 4.4 Changing Interrupt Function Declaration**
4.1.2 Generating Programs Automatically

(1) Create a new project with the integrated development environment CS+ or e2studio.
(2) Set each function with the code generator.

Set the CPU clock to 16MHz high-speed OCO clock.
Set RTC and interval timer operation clock is 32.768kHz.

![Setting Window in Code Generator (Clock)](image-url)

Figure 4.5 Setting Window in Code Generator (Clock)
Set Real-time Clock (RTC) which is the equivalent function to Timer RE of the R8C family.

![Image of the setting window in Code Generator](image)

**Figure 4.6 Setting Window in Code Generator (Real-time Clock)**

(3) Set ports, watchdog timer, voltage detection circuit, etc. based on your environment.

(4) Click [Generate Code] to generate a file.
4.1.3 Adding Programs

Add the processes of symbol definition and the main function to the program with generated code. Use the programs with generated code for other programs (such as clock setting and RTC function setting).

- Symbol definition

Add symbol definition to r_cgrtc.h.

Program for R8C

```c
#define NO_UPDATE 0 /* No update */
#define UPDATE 1 /* Update */
#define COMM 0 /* Common year */
#define LEAP 1 /* Leap year */
#define DEC 0x12 /* December */
#define WEEK 0x04 /* A day of the week(thursday) */
```

Add contents corresponding to the red box manually.

r_cgrtc.h file for RL78/G14

```c
/* Start user code for function. Do not edit comment generated here */
#define NO_UPDATE 0 /* No update */
#define UPDATE 1 /* Update */
#define COMM 0 /* Common year */
#define LEAP 1 /* Leap year */
#define DEC 0x12 /* December */
```

Figure 4.7 Replacement of Symbol Definition
Replacement Guide for R8C Family to RL78 Family (CcnvNC30)

- main function

When the code generator for RL78/G14 is used, the R_Systeminit function is executed before the main function is executed. The R_Systeminit function performs the initial setting of the clock and RTC. Thus, only the process indicated in the red box is added manually. The R_RTC_Start function starts the operation of RTC.

In R8C/35A, customers can use Timer RE for real-time clock. In RL78/G14, customer can use RTC for real-time clock. In the main function of "R8C/35A Group Clock Operation Using RTC (R01AN0079E)", the initialization function and the start command of Timer RE are executed. So in main function of "r01an3508_rl78g14_rtc", the initialization function (RTC generate the interrupt per one second) and the start function of RTC are executed.

Program for R8C

```c
void main(void)
{
    asm("PCLR I"); /* Interrupt disabled */
    mcu_init(); /* MCU initialize */
    timer_re_init(); /* Timere initialize */
    asm("FSER I"); /* Interrupt enabled */

    leap_flg = leap_chk(); /* Leap year check */
    tstart_trecrl = 1; /* TRE1 count start */
    while(tstart_trecrl == 0); /* if TRE1 count start? else wait */

    while(1)
    {
        if(up_flg == UPDATE){
            update(); /* Update processing */
        }
    }
}
```

r_main.c file for RL78/G14

```c
void main(void)
{
    R_MAIN_Initial(); /* START user code. Do not edit comment generated here */
    leap_flg = leap_chk(); /* Leap year check */
    R_RTC_Set_ConstPeriodInterruptOn(SEC); /* Interrupt turned on */
    R_RTC_Start(); /* Enabled */

    while (1U)
    {
        if(up_flg == UPDATE){
            update(); /* Update processing */
        }
    }
    /* End user code. Do not edit comment generated here */
}
```

Figure 4.8 Replacement of main Function
Additionally, "update" and "leap_chk" functions, which are used by "main" function, should be added into r_main.c file. And the variables that are used by these two functions are also needs to be added in r_main.c file.

Variables for R8C

```
/*----------------------------------------------------------*/
/*   R01                                             */
/*----------------------------------------------------------*/
unsigned short year = 0x2008;     /* Year */
unsigned char month = 0x01;       /* Month */
unsigned char day = 0x01;          /* Day */
unsigned char wk = WEEK;           /* Sun,Mon,Tue,Wed,Thu,Fri,Sat */
unsigned char hr = 0x00;           /* Hour */
unsigned char min = 0x00;          /* Minute */
unsigned char sec = 0x00;          /* Second */
/* Work area */
unsigned char wk_sld = 0x00;       /* Last-time value of 'wk' */
/* Flags */
unsigned char up_flg = NO_UPDATE; /* Update flag */
unsigned char leap_flg = CONN;    /* Leap flag */
/*----------------------------------------------------------*/
/*   R01                                             */
/*----------------------------------------------------------*/
unsigned char const MONTH_DAYS[13] = {
   /* This table contains the number of days in different months */
   0x00,0x31,0x30,0x31,0x30,0x31,0x30,0x31,0x30,0x31,0x30,0x31
};
unsigned char const MONTH_DAYS_L[13] = {
   /* This table contains the number of days in different months (leap year) */
   0x00,0x31,0x30,0x31,0x30,0x31,0x30,0x31,0x30,0x31,0x30,0x31
};
```

r_main.c file for RL78/G14

```
global variables and functions
/*----------------------------------------------------------*/
/*   R01                                             */
/*----------------------------------------------------------*/
unsigned short year = 0x2008;     /* Year */
unsigned char month = 0x01;       /* Month */
unsigned char day = 0x01;          /* Day */
unsigned char wk = 0x04;           /* Sun,Mon,Tue,Wed,Thu,Fri,Sat */
unsigned char hr = 0x00;           /* Hour */
unsigned char min = 0x00;          /* Minute */
unsigned char sec = 0x00;          /* Second */
unsigned char wk_sld = 0x00;       /* Last-time value of 'wk' */
unsigned char up_flg = NO_UPDATE; /* Update flag */
unsigned char leap_flg = CONN;    /* Leap flag */
unsigned char const MONTH_DAYS[13] = {
   /* This table contains the number of days in different months */
   0x00,0x31,0x30,0x31,0x30,0x31,0x30,0x31,0x30,0x31,0x30,0x31
};
unsigned char const MONTH_DAYS_L[13] = {
   /* This table contains the number of days in different months (leap year) */
   0x00,0x31,0x30,0x31,0x30,0x31,0x30,0x31,0x30,0x31,0x30,0x31
};
```

Figure 4.9 Add Variables
Program for R8C

```c
#ifndef FUNC_COMMENT
#define FUNC_COMMENT

/* Start user code for adding. Do not edit comment generated here */

/* Function Name: update
 * Description : Update check
 * Arguments : None
 * Return Value : None
 */
void update(void)
{

/* Function Name: leap_chk
 * Description : Leap year check
 * Arguments : None
 * Return Value : None
 */
unsigned char leap_chk(void)
{

Add contents corresponding to the red box manually.
```

Figure 4.10 Add Functions
### 4.1.4 Other Items to be Corrected

If RTC is set with the code generator, interrupt processes are automatically generated. Add the user program to interrupt routine.

**Program for R8C**

```c
/* Func Comment */

#define void_timer_r8(void);

/** Func Comment **/

/* Outline */

/* Declaration */

/* Description */

/* Argument */

/* Return Value */

/* Func Comment End */

/** Func Comment **/

/* void _timer_r8(void) */

/* void _timer_r8(void) */

sec = t sec & 0xFF;
min = t min & 0xFF;
hr = t hr & 0xFF;
wk old = wk;
wk = t wk & 0x07;
up_flg = UPDATE;
/* Set second to RAM */
/* Set minute to RAM */
/* Set hour to RAM */
/* Set last-time value */
/* Set day to RAM */
/* Set update flag */

**Figure 4.11 Add Interrupt Program**

Additionally, these variables need external declaration in `r_cg_RTC_user.c` file.

```c
/* Start user code. Do not edit comment generated here */

extern unsigned short year;
/* Year */

extern unsigned char month;
/* Month */

extern unsigned char day;
/* Day */

extern unsigned char wk;
/* Sun,Mon,Tue,Wed,Thu,Fri,Sat */

extern unsigned char ha;
/* Hour */

extern unsigned char mn;
/* Minute */

extern unsigned char sec;
/* Second */

extern unsigned char wk old;
/* Last-time value of `wk` */

extern unsigned char up_flg;
/* End user code. Do not edit comment generated here */
```

**Figure 4.12 Add External Declaration**
4.1.5 Sample Code After Replacement

Obtain the sample code "an-r01an3508ec0100-rl78-migrate.zip" from the Renesas Electronics Website. "rl78g14_migrate_rtc" in the "workspace" folder is the sample code that replaces the program of "R8C/35A Group Clock Operation Using RTC (R01AN0079E)".
4.2 R8C Sample Program (DTC Operation)

The program of "R8C/35C Group DTC Operation in Chain Transfers (R01AN0372E)" is replaced with the program for RL78/G14. The project file after replacement is "r01an3508_rl78g14_dtc".

This program uses DTC operation in chain transfers.

4.2.1 Porting Source to CC-RL with CcnvNC30

(1) Create a list file to specify a C source file to be converted.

```
-c=sjis r01an0372_src.c output\dtc.c
```

Figure 4.13 Example of Description in List File (DTC Operation in Chain Transfers)

(2) Launch Command Prompt to convert the C source file specified with the list file. In addition, the output conversion result file indicates changes.

Figure 4.14 CcnvNC30 Execution Window (DTC Operation in Chain Transfers)

The conversion result file indicates the conversion result as shown below. For details of the conversion result, see "CcnvNC30 C source converter User's Manual (R20UT3685E)".

Figure 4.15 Details of Conversion Result (DTC Operation in Chain Transfers)
(3) Correct the converted C source file.

There may be redundant interrupt function declarations. As CC-RL produces an error in this case, delete the converted #pragma directive.

```c
86     #pragma address ad value 0x00600E /* Destination address of A/D conversion data */
87     #pragma address ad value=0x00600E

Delete the #pragma directive.
```

**Figure 4.16  Changing Interrupt Function Declaration**

### 4.2.2 Generating Programs Automatically

1. Create a new project with the integrated development environment CS+ or e2studio.
2. Set each function with the code generator.

Set the CPU clock to 16MHz high-speed OCO clock.

**Figure 4.17  Setting Window in Code Generator (Clock)**
Set A/D Converter (ADC) which is the equivalent function to ADC of the R8C family.
Use A/D converter in software trigger mode, scan mode, and one-shot conversion mode.

Figure 4.18 Setting Window in Code Generator (ADC)
Set Data Transfer Controller (DTC) which is the equivalent function to DTC of the R8C family.

In the sample program of “R8C/35C Group DTC Operation in Chain Transfers (R01AN0372E)”, A/D converted values that have been stored to A/D registers (AD0 to AD3) are transferred to the internal RAM using the DTC chain transfer. But, in RL78/G14 Group, there is only one A/D conversion result register, so use DTC (normal mode, not use chain transfer). Perform A/D conversion on analog input voltage input to pins ANI0 to ANI3 while in scan mode, and use DTC transfer to store the A/D converted value to the RAM. Perform A/D conversion for individual pins successively. Every time A/D conversion for a pin is completed, store the converted result to the 10-bit A/D conversion result register (ADCR), activate the DTC, and transfer the A/D converted result from the ADCR register to the RAM. When A/D conversion and DTC transfer for all of the above pins are completed, an A/D conversion end interrupt request is generated.

Click the DTCD0 tab to open the detailed settings.

Figure 4.19  Setting Window in Code Generator (DTC)

(3)  Set ports, watchdog timer, voltage detection circuit, etc. based on your environment.
(4)  Click [Generate Code] to generate a file.
4.2.3 Adding Programs

Add the processes of symbol definition and the main function to the program with generated code. Use the programs with generated code for other programs (such as clock setting, ADC function, and DTC function setting).

- main function

When the code generator for RL78/G14 is used, the R_Systeminit function is executed before the main function is executed. The R_Systeminit function performs the initial setting of the clock, ADC, and DTC. Thus, only the process indicated in the red box is added manually. The R_DTCD0_Start function starts the operation of DTC. The R_ADC_Start function starts the operation of ADC.

Program for R8C

```c
void main(void)
{
    /* === Interrupt disabled === */
    asm("PCLR I");

    /* === Set High-speed on-chip oscillator clock to System clock === */
    ncv_init();

    /* === A/D converter initialize === */
    ad_converter_enable();

    /* === DTC initialize === */
    dtc_enable();

    /* === interrupt enabled === */
    asm("PSET 1");

dtcen16 = 1;                      /* Activation of A/D interruption enable */
adst = 1;                        /* A/D conversion starts */

    /* === Main loop === */
    while(1){
    }
}
```

r_main.c file for RL78/G14

```c
void main(void)
{

    R_MAIN_UserInit();                /* Start user code. Do not edit comment generated here */
    R_DTCD0_Start();                 /* DTC enabled */
    R_ADC_Set OperationOn();         /* A/D converter enabled */

    while (1U)
    {
        NOP();
    }

    /* End user code. Do not edit comment generated here */
}
```

Even if a DTC transfer request is generated, DTC transfer is held pending immediately when there are no commands in the instruction "while (1U)", so add the command "NOP ()" into the "while (1U)".

---

**Figure 4.20 Replacement of main Function**
4.2.4 Other Modifications

If ADC is set with the code generator, interrupt processes are automatically generated. Add the user program to interrupt routine.

Variables for R8C

```c
/* **** Global Variables **** */

#pragma ADDRESS ad_value 00600H /* Destination address of A/D conversion data */
unsigned short ad_value[4]; /* A/D data from AN8 to AN11 addressed from */

unsigned short an8_value; /* AN8 value */
unsigned short an9_value; /* AN9 value */
unsigned short an10_value; /* AN10 value */
unsigned short an11_value; /* AN11 value */
```

Add contents corresponding to the red box manually.

```
#pragma address ad_value=0xFF500
unsigned short ad_value[4]; /* A/D data from AN10 to AN13 addressed from */

unsigned short an0_value; /* AN10 value */
unsigned short an1_value; /* AN11 value */
unsigned short an2_value; /* AN12 value */
unsigned short an3_value; /* AN13 value */
```

**Figure 4.21  Add Variables for Interrupt Program**
4.2.5 Sample Code After Replacement

Obtain the sample code "an-r01an3508ec0100-rl78-migrate.zip" from the Renesas Electronics Website. "rl78g14_migrate_dtc" in the "workspace" folder is the sample code that replaces the program of "R8C/35C Group DTC Operation in Chain Transfers (R01AN0372E)".
4.3 Conditions for Confirming Operations of Sample Programs

The operations of the sample codes after replacement are confirmed under the following conditions.

Table 4.1 Conditions for Confirming Operations

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller used</td>
<td>RL78/G14 (R5F104PJ)</td>
</tr>
<tr>
<td>Integrated development environment (CS+)</td>
<td>CS+ for CC V4.01.00 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>C compiler (CS+)</td>
<td>CC-RL V1.03.00 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>Integrated development environment (e2 studio)</td>
<td>e2 studio V5.2.0.020 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>C compiler (e2 studio)</td>
<td>CC-RL V1.03.00 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>Board used</td>
<td>Renesas original</td>
</tr>
</tbody>
</table>
5. **Sample Code**

The sample code is available on the Renesas Electronics website.

6. **Reference Documents**

User’s Manual

- RL78 Family User’s Manual: Software (R01US0015E)
- CC-RL Compiler User’s Manual (R20UT3123E)
- CcnvNC30 C Source Converter User's Manual (R20UT3685E)

Application Note

- R8C/35A Group Clock Operation Using RTC (R01AN0079E)
- R8C/35C Group DTC Operation in Chain Transfers (R01AN0372E)

(The latest versions can be downloaded from the Renesas Electronics website.)

**Website and Support**

Renesas Electronics Website

http://www.renesas.com/

Inquiries

http://www.renesas.com/contact/

All trademarks and registered trademarks are the property of their respective owners.
### Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Dec. 22, 2016</td>
<td>-</td>
<td>First edition issued</td>
</tr>
</tbody>
</table>
General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins
   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable.
   When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal.
   Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products
   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.
   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.
Notice

1. Data, software, and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these devices, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these products, software, or information.

2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors or omissions from the information included herein.

3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No licenses, express or implied, are granted hereby unless any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You shall not, alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.

5. Renesas Electronics product are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depend on the products quality grade, as indicated below.

   "Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronics appliances, machine tools, personal electronics equipment, and industrial robots etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-arms systems, and safety equipment etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (critical life support devices or systems, surgical implantations etc.) or may cause serious property damage (nuclear reactor control systems, military equipment etc.). You should check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use Renesas Electronics product for any application the which is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.

6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, as possibly with respect to the maximum rating, operating supply voltage range, maximum power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall not have any liability for malfunctions or damages arising out of the use of the Renesas Electronics products beyond this specified range.

7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard against the possibility of physical injury and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardwares and softwares including but not limited to redundancy, fire control and radiation prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the validity of the final product or systems manufactured by you.

8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that require the installation or use of contaminated substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable sanctions or foreign laws or regulations. You must not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by military, including but not limited to the development and testing of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedure required by such laws and regulations.

10. In the event of the buyer or distributor of Renesas Electronics products, which distributes, resells or otherwise places the products with a third party, to notice such third party, in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of any unauthorized use of Renesas Electronics products.

11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics products” means any product developed or manufactured by or for Renesas Electronics.

SALES OFFICES

Renesas Electronics Corporation
http://www.renesas.com

Refer to “http://www.renesas.com” for the latest and detailed information.

© 2015 Renesas Electronics Corporation. All rights reserved.