

# Smart Analog

R20AN0246EJ0100

Rev.1.00

## System development procedure by using SA-Designer (RX family)

Apr 3, 2013

### Introduction

This application note is intended to explain the steps involved in developing a simple system in an environment that uses the Smart Analog, assuming the RX63N.

- Design of the Analog Front-end Circuit
- Creating Program
- Register and Build the Circuit Data
- Testing

### Contents

1.	Introduction.....	2
1.1	Development Environment.....	3
1.1.1	Hardware.....	3
1.1.2	Software.....	3
2.	Development Procedure .....	4
2.1	Overview .....	4
2.1.1	Design of the Analog Front-end Circuit.....	5
(1)	Starting the SA-Designer .....	5
(2)	The Design of the New Circuit .....	5
(3)	Creating a Circuit Diagram.....	7
(4)	Generation Source File .....	8
2.1.2	Creating Program.....	9
(1)	Starting CubeSuite+ .....	9
(2)	New Project.....	10
(3)	Creating Program.....	13
2.1.3	Register and Build the Circuit Data.....	16
(1)	Register the Source Files to CubeSuite+.....	16
(2)	Build .....	18
2.1.4	Testing .....	19
(1)	Download the Load Module .....	19
(2)	Registration Variable to the Watch Window.....	24
(3)	Run Program .....	26
3.	Sample Programs.....	28
(1)	Function main (In addition to the main function of RX63N.c) .....	28
(2)	Initialization function (Add to RX63N.c).....	29
(3)	Interrupt function (Add to intprg.c).....	30
(4)	Function SPI (Add to RX63N.c) .....	30

## 1. Introduction

This application note is intended to explain the steps involved in developing a simple system in an environment that uses the Smart Analog, assuming the RX63N. The system uses a temperature sensor built into the "smart analog IC". Depending on the temperature, change the blink rate of the LED. The system uses "Smart Analog IC" and "GR-SAKURA" as CPU board. The application note explains the procedures of the load module that uses the High-performance Embedded Workshop and the SA-Designer, and the procedures of testing program.

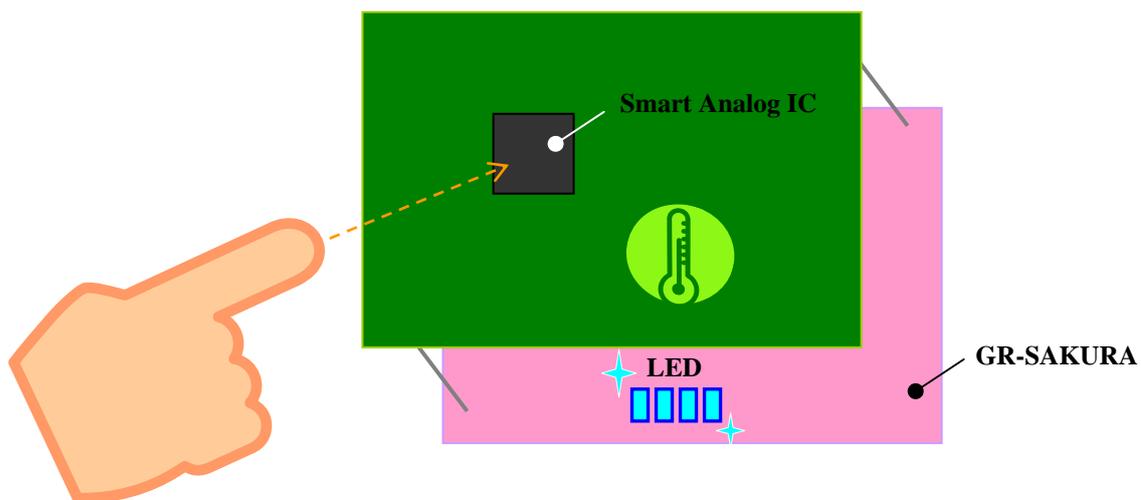


Figure 1 System summary

## 1.1 Development Environment

The application note uses the following development environments.

### 1.1.1 Hardware

- Host PC
- evaluation board: GR-SAKURA (RX63N), Smart Analog IC500
- E1 emulator

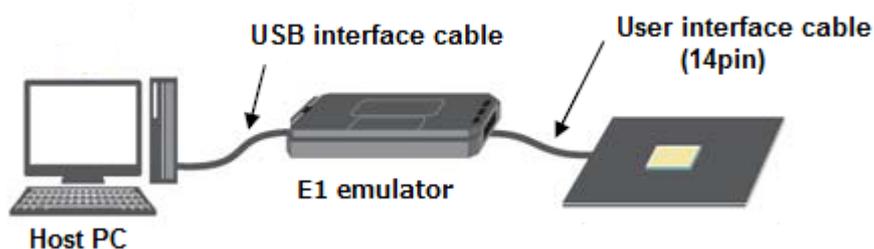


Figure 2 Hardware construction

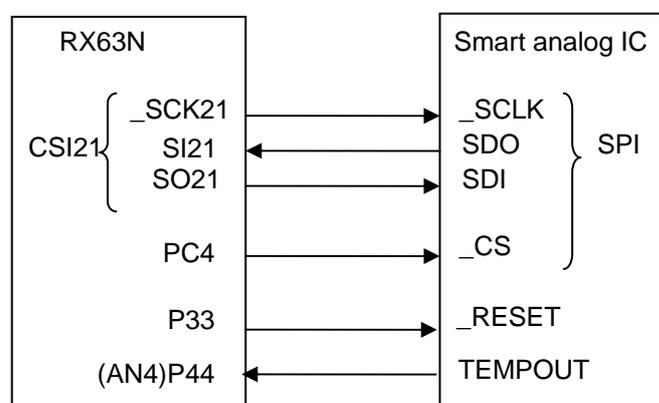


Figure 3 Constructing RX63N and Analog chip

### 1.1.2 Software

- SA-Designer (V1.00.00)
- IDE CubeSuite+ (Version 1.02.01)

### **2. Development Procedure**

#### **2.1 Overview**

The following instructions describe the construction procedures of the system.

It will be used the CubeSuite+ and the SA-Designer for the construction procedures. Followings are the steps of the system development.

##### **(1) Design the analog front-end circuit**

Design the analog front-end circuit using with the SA-Designer.



##### **(2) Programming the source codes**

Program the source codes that set the clock, the ports, the A/D conversion functions of the microcomputer, also for the operation of the system.



##### **(3) Registering the setting program of the circuit data.**

Register the C source codes made by the SA-Designer to the CubeSuite+, then build the codes.



##### **(4) Operation check**

Connect the E1 emulator and write the program to the microcomputer, then check the operations of the program.

Note: It will be needed to install the CubeSuite+ (At least Version V1.02.00) and the SA-Designer for the program operations.

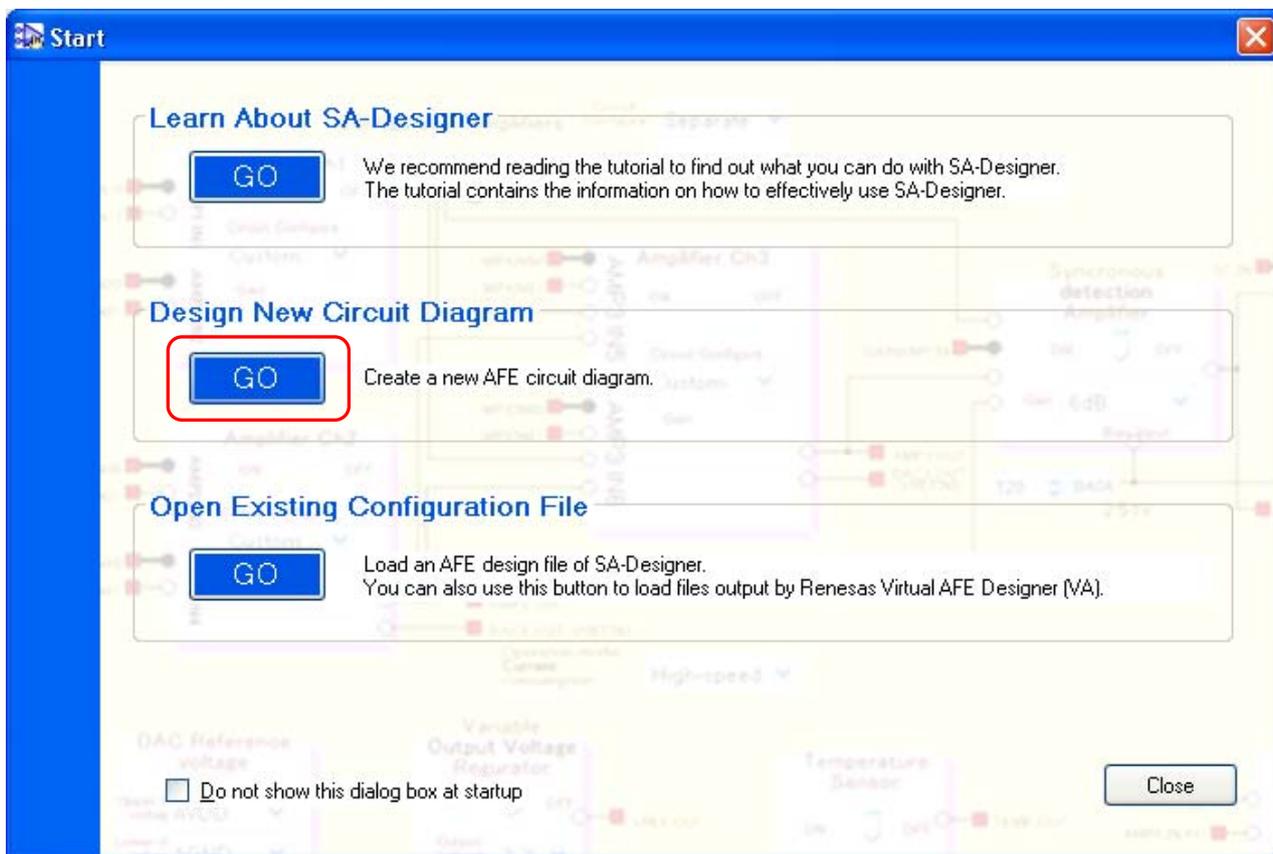
### 2.1.1 Design of the Analog Front-end Circuit

#### (1) Starting the SA-Designer

Start the SA-Designer by selecting the [SA-Designer] from the Start menu.

#### (2) The Design of the New Circuit

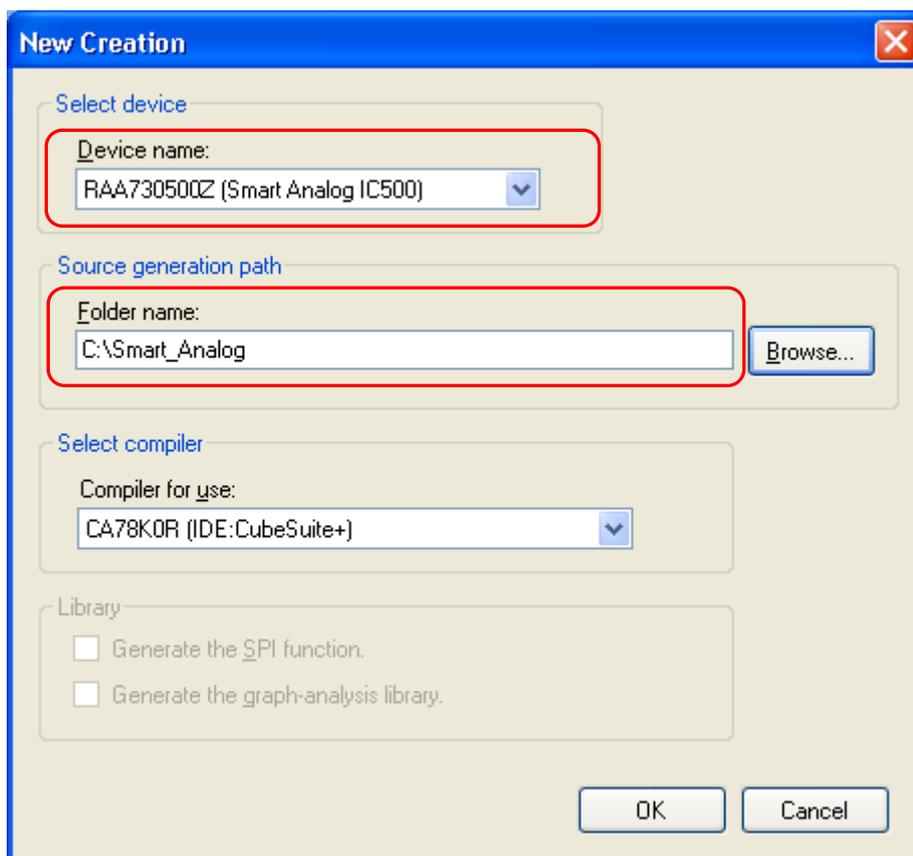
For the design of the analog front-end circuit, choose the destination device and the folder.



Click the GO in "Design New Circuit Diagram".

## Smart Analog System development procedure by using SA-Designer (RX family)

Choose the device and the folder for creating the codes in the “New dialog”.

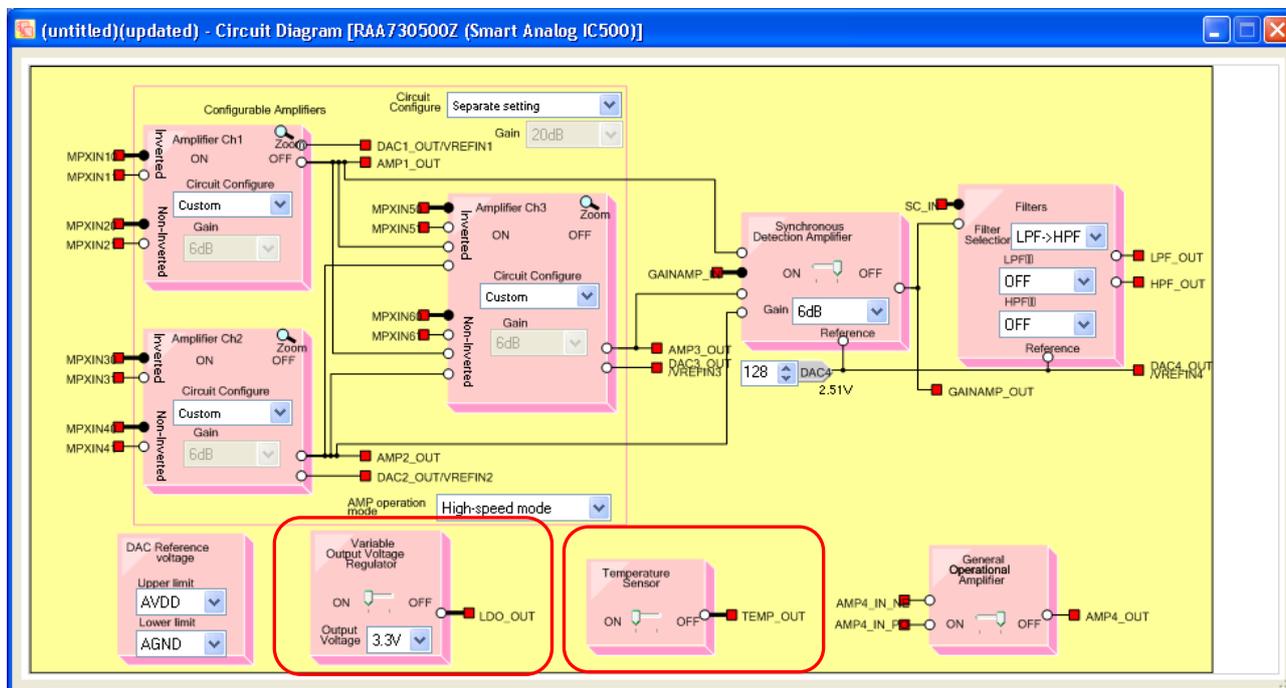


[Device]	Choose the device “RAA730500Z (Smart Analog IC500)”
[Folder name]	Choose the folder arbitrarily. Choose "Smart_Analog" as above image. Choose the folder as above image. The folder must be existed in the computer.

## Smart Analog System development procedure by using SA-Designer (RX family)

### (3) Creating a Circuit Diagram

Design the analog front-end circuit to use a temperature sensor. Change the settings from the initial state of the circuit diagram as follows.



[Variable Output Voltage Regulator]

Set the switch to "ON".

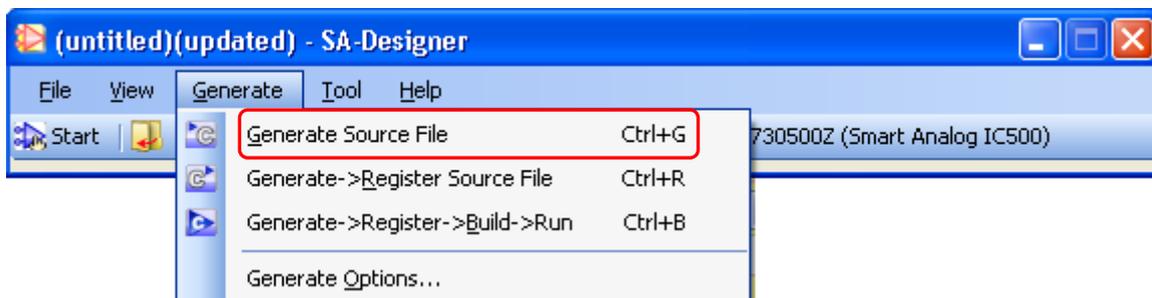
[Temperature Sensor]

Set the switch to "ON".

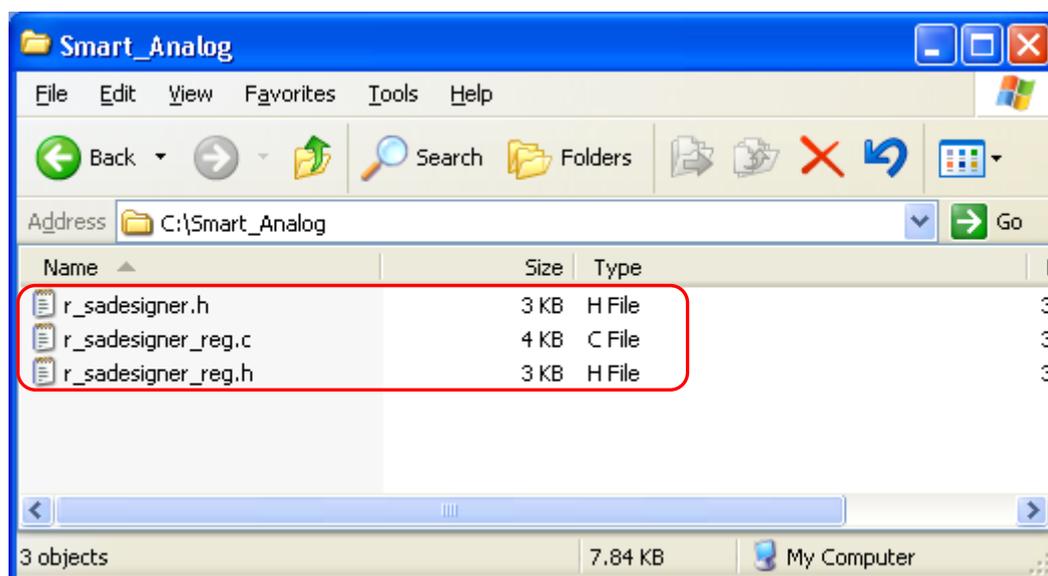
## Smart Analog System development procedure by using SA-Designer (RX family)

### (4) Generation Source File

Program the source codes to set the data of the designed circuits. Completion dialog will be displayed when you click the "[Generate] - [Generate Source File]".



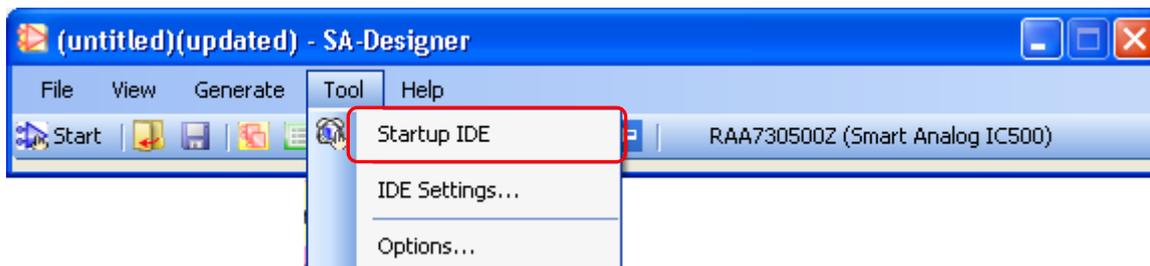
Three "C source files" will be made in the folder that you choose.



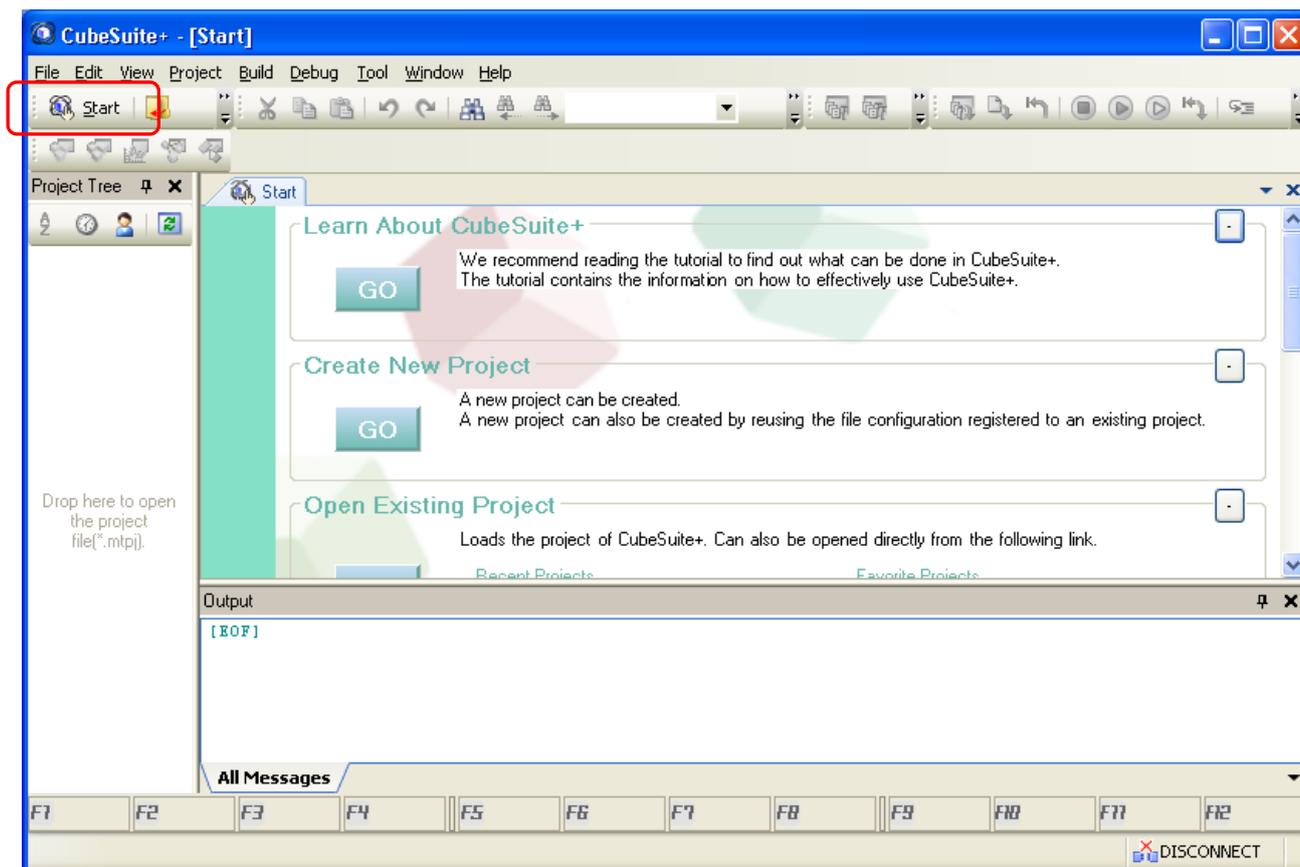
## 2.1.2 Creating Program

### (1) Starting CubeSuite+

Launch the CubeSuite+, from menu of SA-Designer. It also can be launched from “Windows Start menu”. And you need to install ”CubeSuite+” beforehand. Click [Startup IDE] of SA-Designer from “Tool”, then “CubeSuite+” will be launched.



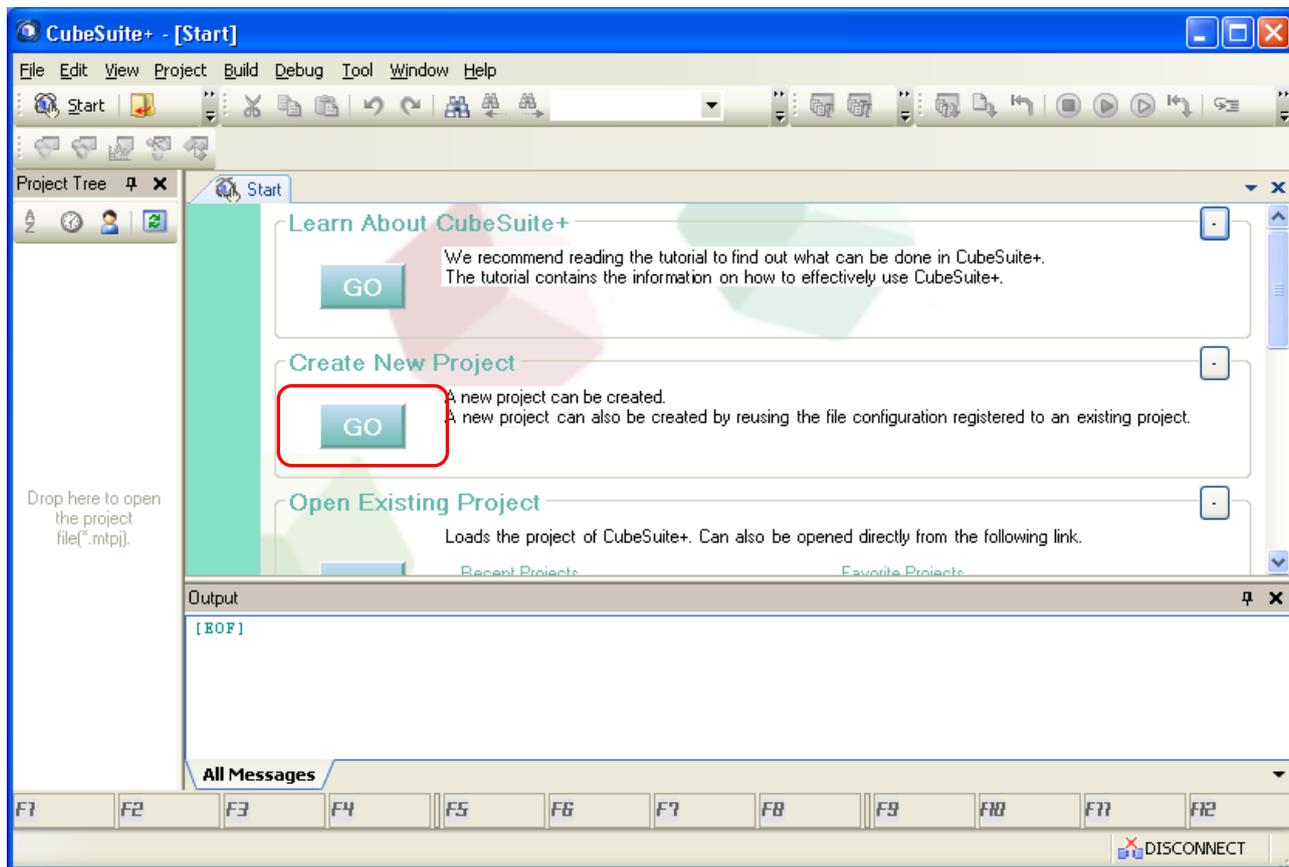
Open “Menu Window” with pressing [Start] after starting CubeSuite+.



## Smart Analog System development procedure by using SA-Designer (RX family)

### (2) New Project

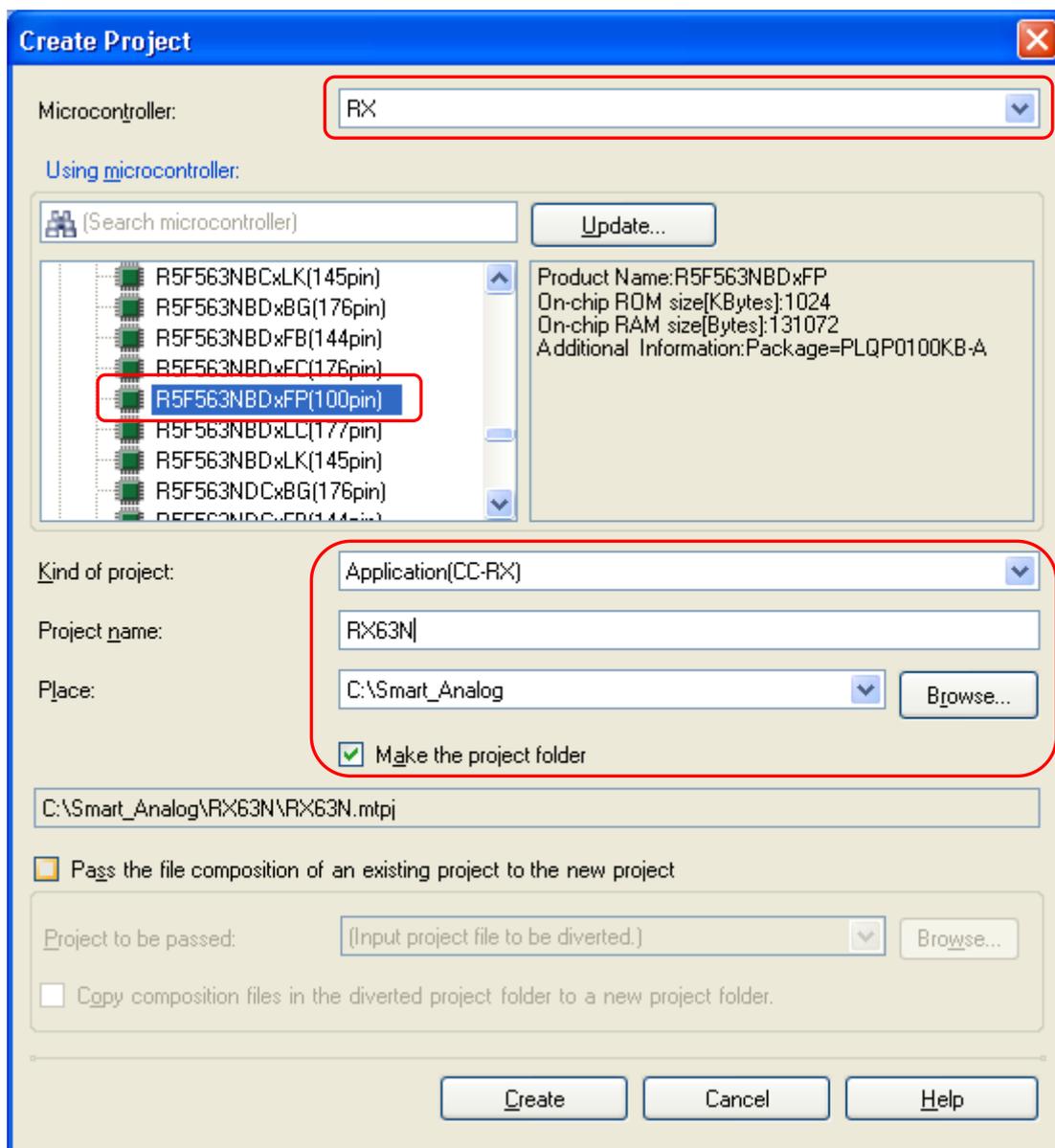
Create a project workspace in the CubeSuite+.



Press "GO" button in "Create New Project".

## Smart Analog System development procedure by using SA-Designer (RX family)

Set the project information in [Create Project] dialog.

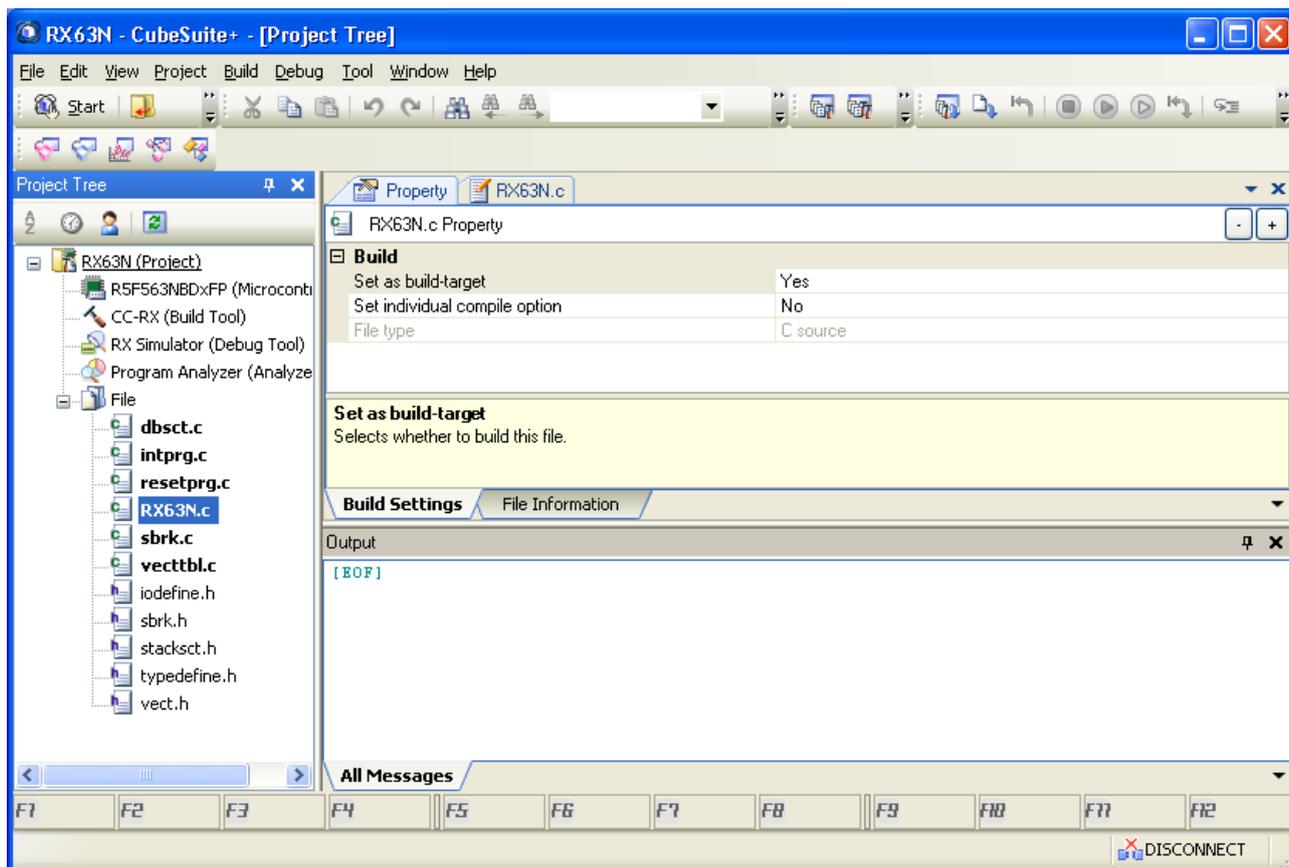


[Microcontroller]	Choose "RX".
[Using microcontroller]	Choose "R5F563NEDxFP(100pin)" in "RX63N".
[Kind of project]	Choose "Applications(CC-RX)".
[Project name]	Type "RX63N" as above image.
[Place]	Choose "Smart_Analog" as above image. Check "Make the project folder".

Press "Create" button.

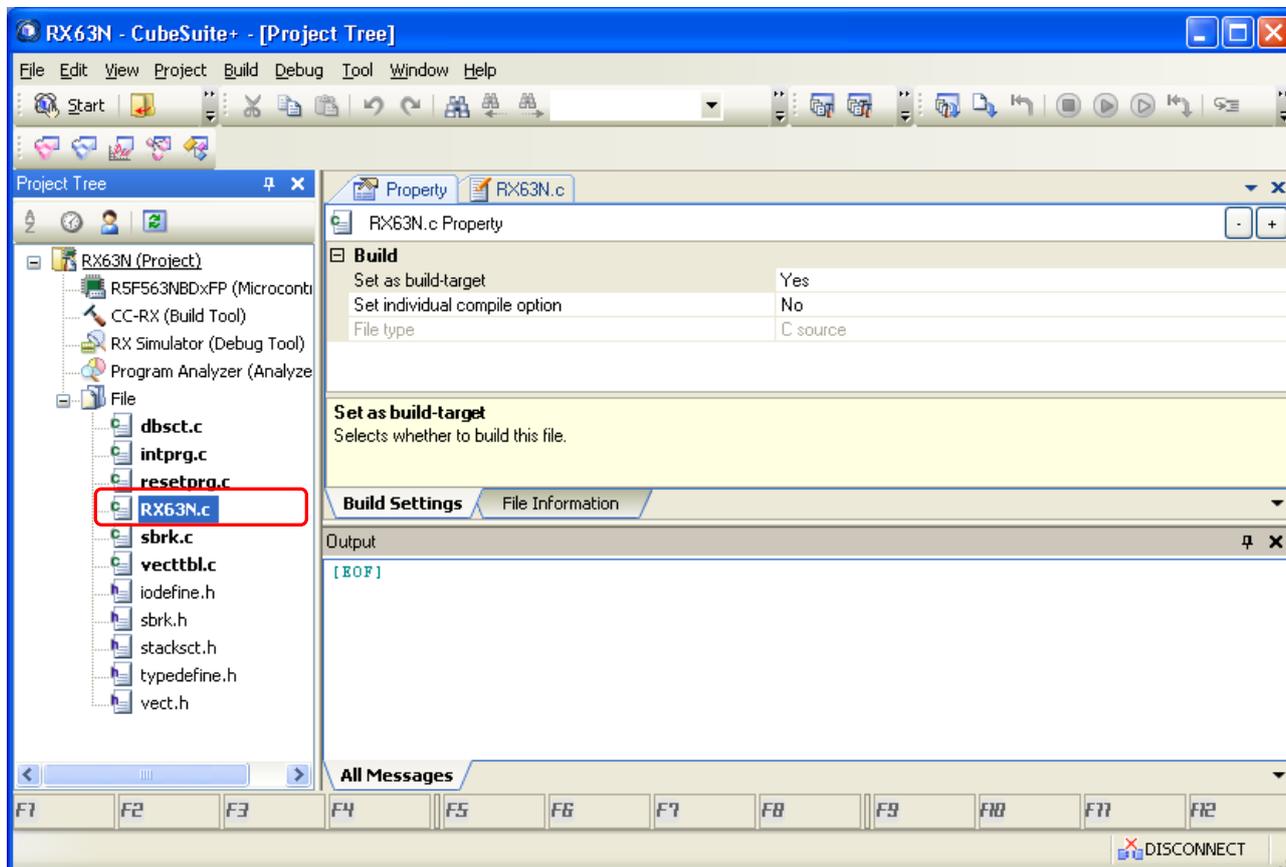
## Smart Analog System development procedure by using SA-Designer (RX family)

The Project will be made and be displayed in the tree of the Project Tree panel.



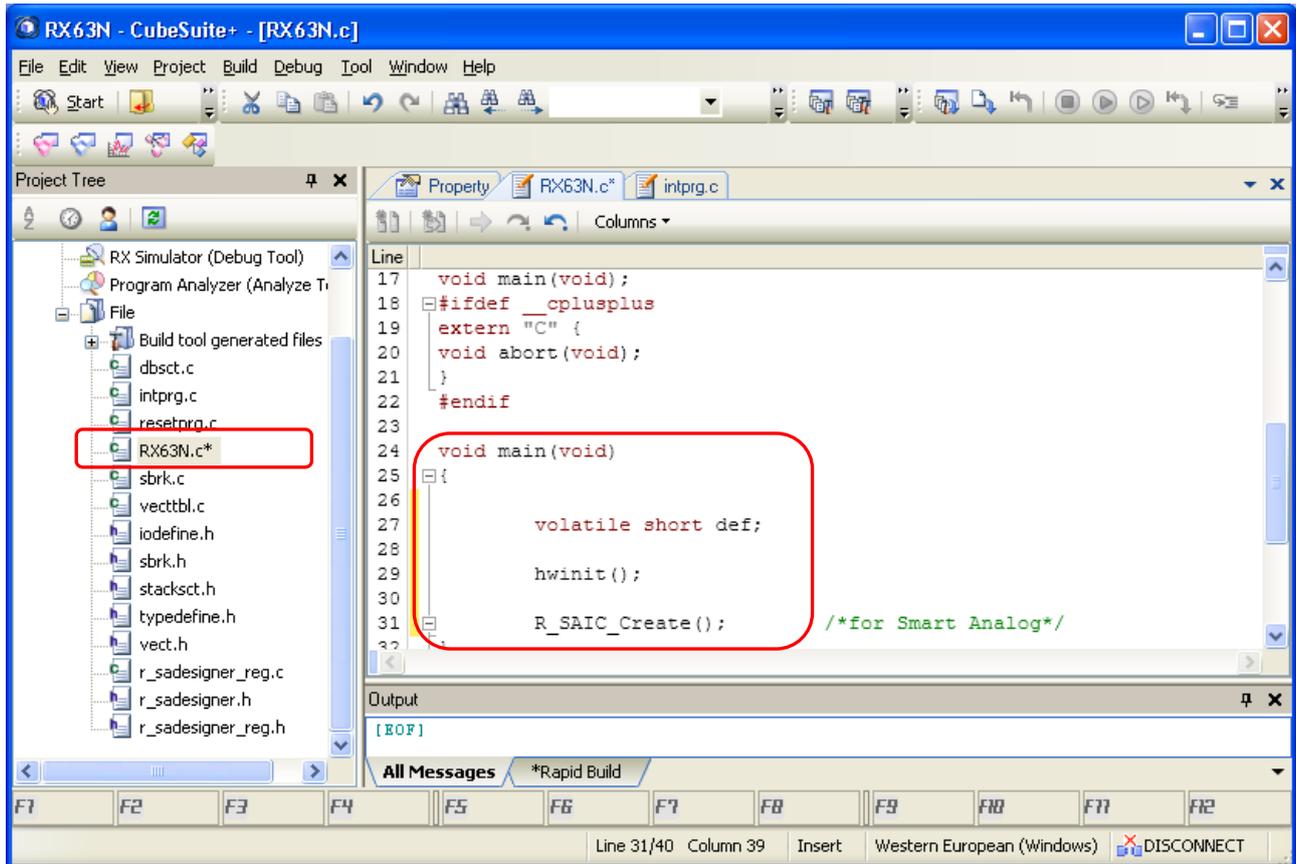
## (3) Creating Program

Program the codes for using the clock settings and the function of A/D. As follows, the codes will be programmed in the RX63N.c and ntrpg.c samples which are made by the CubeSuite+ as sample. Refer to “3. Sample Programs” for the programs that you need.



Open the source file “RX63N.c.”

## Smart Analog System development procedure by using SA-Designer (RX family)

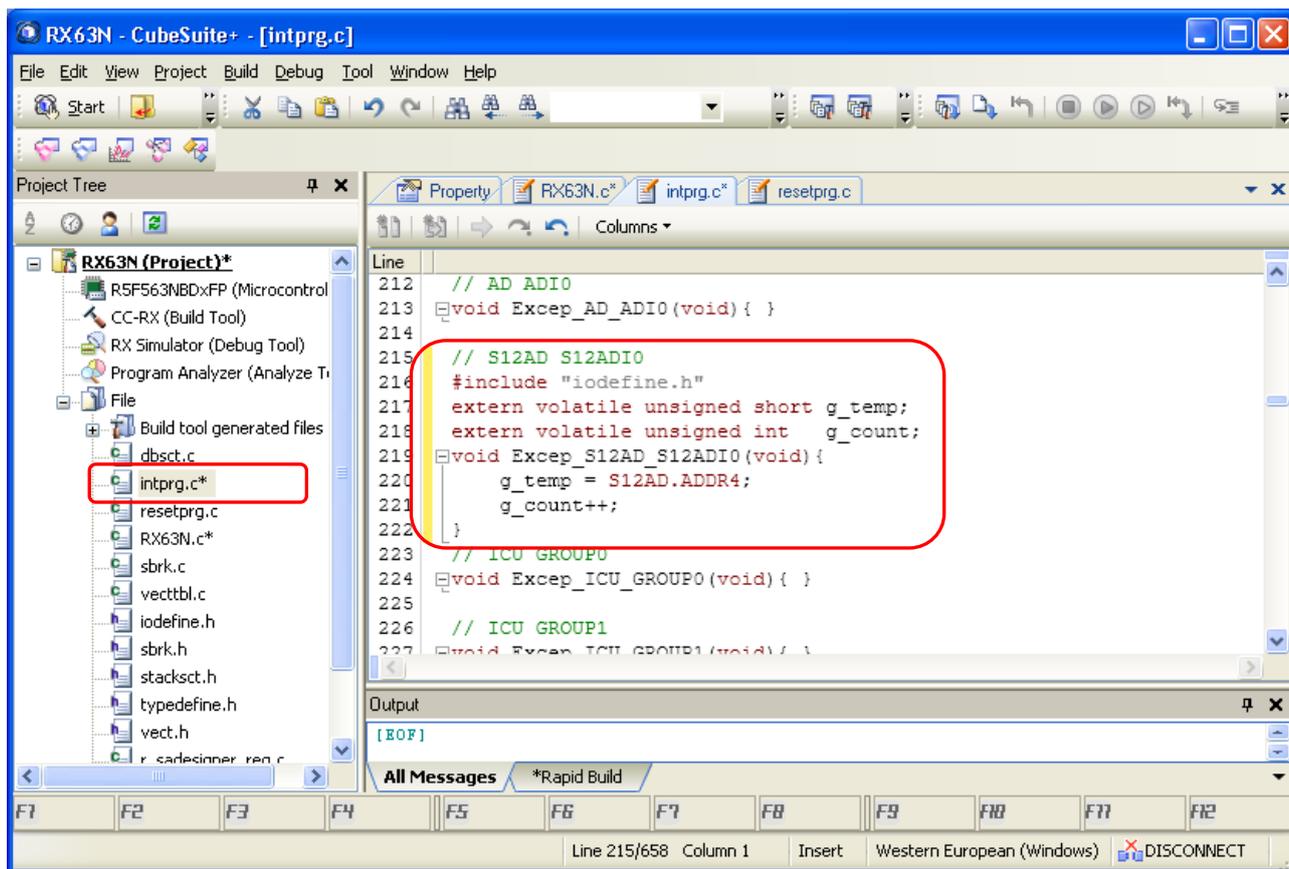


Add procedures to the main function of the source file RX63N.c.

Also add the function hwinit and SPI initialization function to the RX63N.c.

Refer to “3. Sample Programs” for the programs that you need.

## Smart Analog System development procedure by using SA-Designer (RX family)



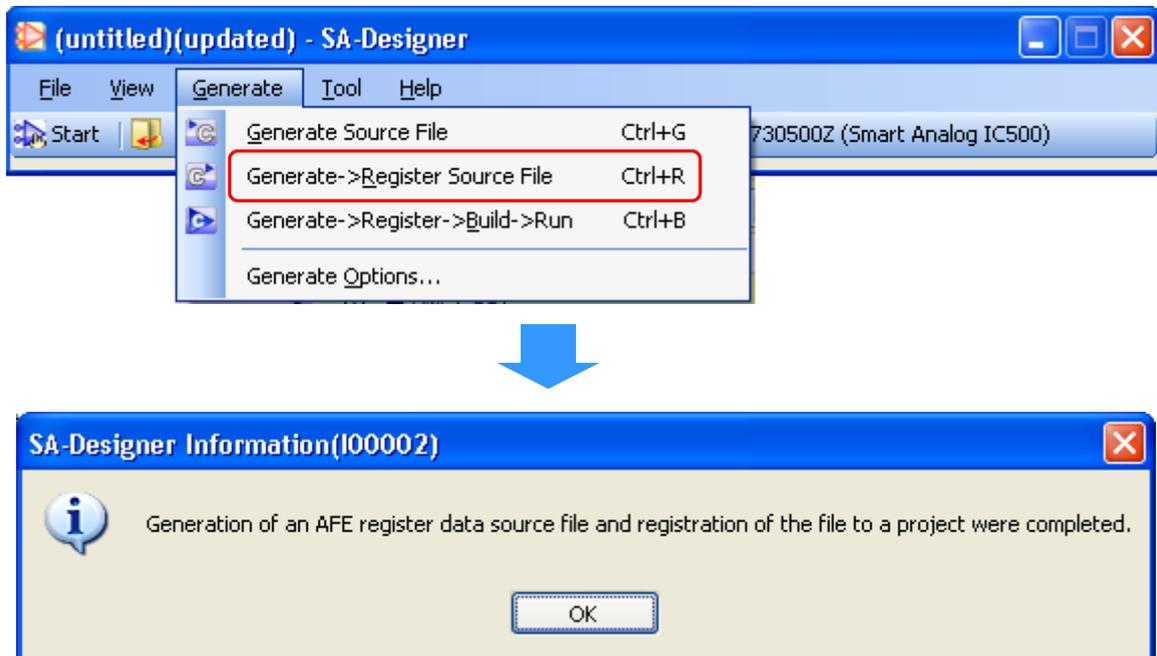
Add processing of Excep\_S12AD\_S12ADI0 interrupt function of the source file intrpg.c.  
Refer to “3. Sample Programs” for the programs that you need.

## Smart Analog System development procedure by using SA-Designer (RX family)

### 2.1.3 Register and Build the Circuit Data

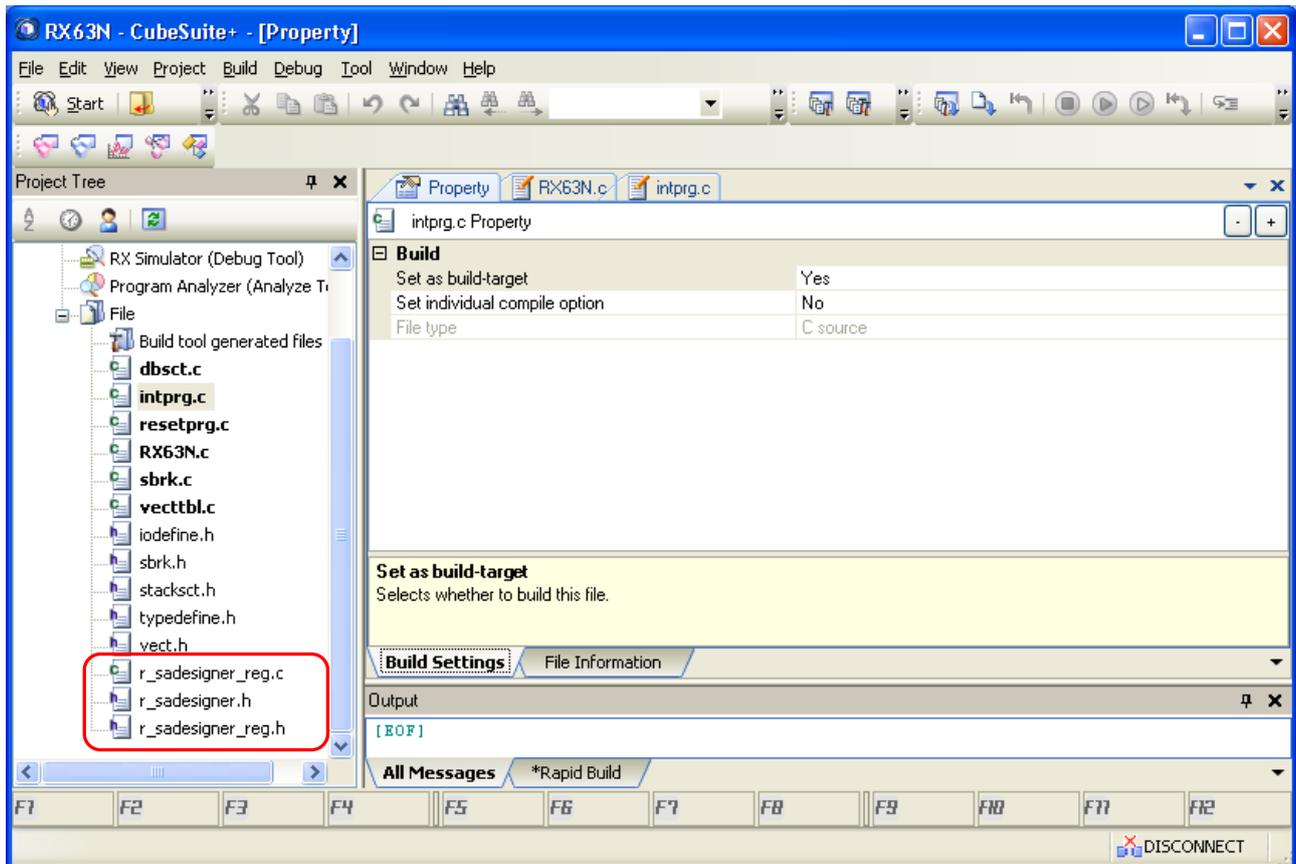
#### (1) Register the Source Files to CubeSuite+

Register the source file, which is made by the SA-Designer to the project that you created in the CubeSuite+.



## Smart Analog System development procedure by using SA-Designer (RX family)

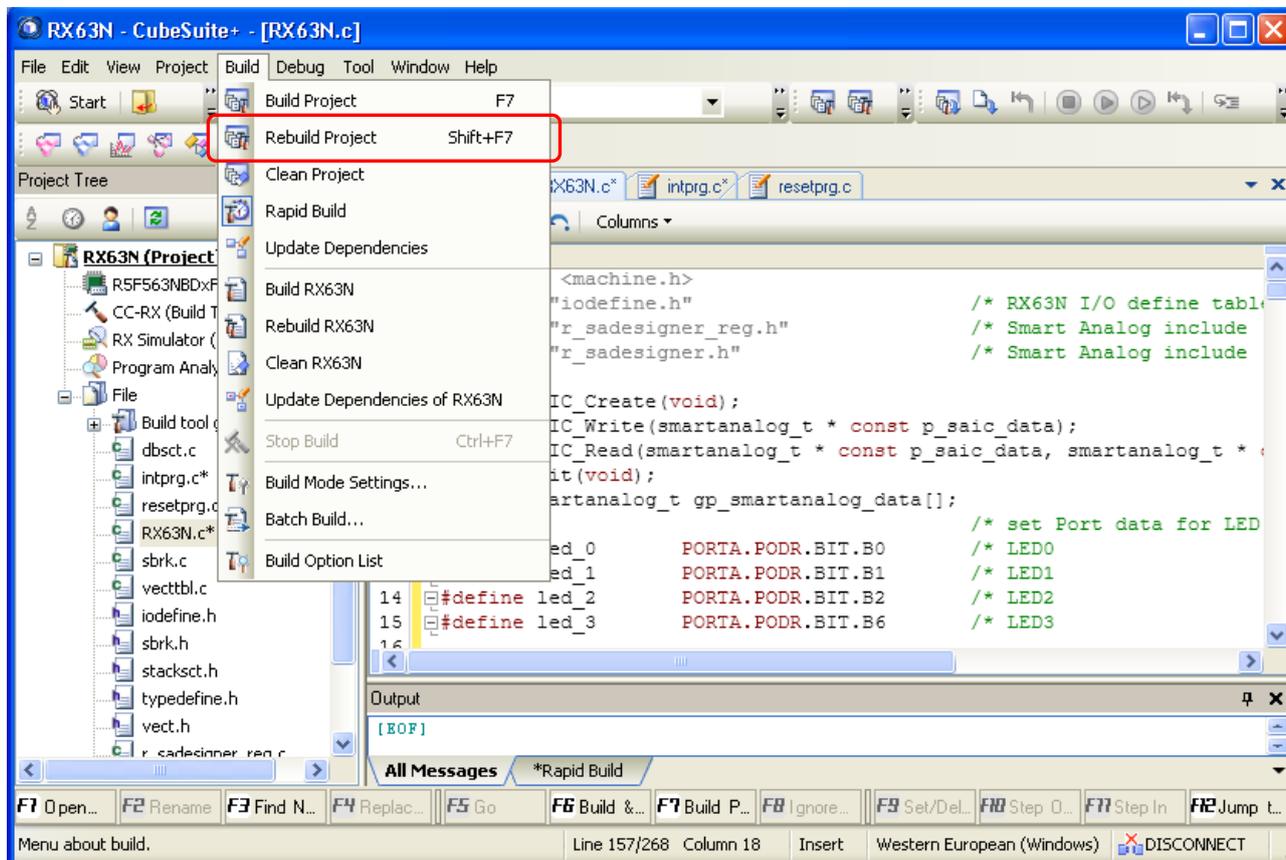
The Circuit data source file will be registered in the project tree.



## Smart Analog System development procedure by using SA-Designer (RX family)

### (2) Build

Choose “Rebuild Project”, then make the load module file.



#### 【Caution】

The following warning message will be shown after the link is started.

```
** L1100 (W) Cannot find "L" specified in option "start"
```

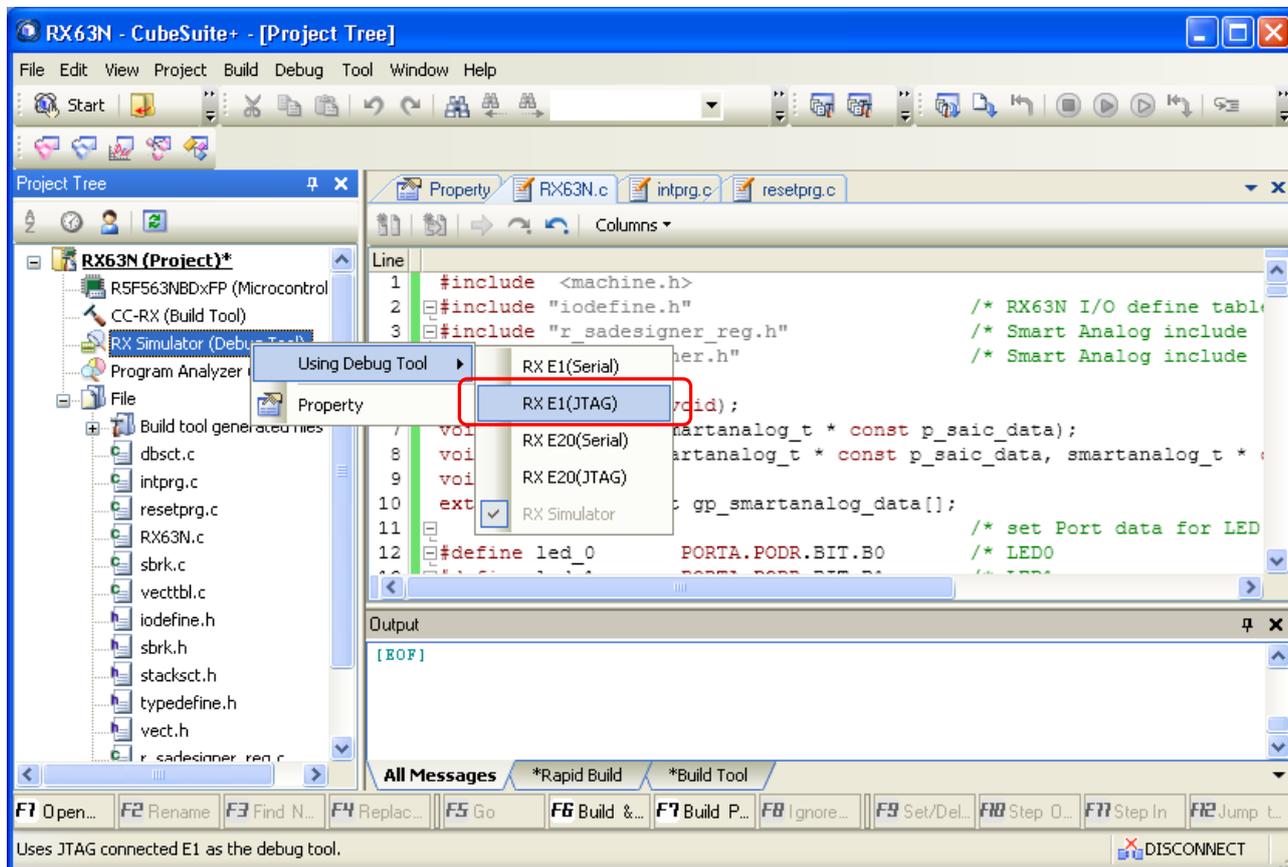
This message can be removed by deleting the section "L" which is specified by default during project creation.

To delete this section, go to [CC-RX(Build Tools)] - [Properties] - [Link Options] - [Section] in the project tree.

## 2.1.4 Testing

### (1) Download the Load Module

Set the debug tool to use from the "Using Debug Tool" in the project tree of the CubeSuite+.

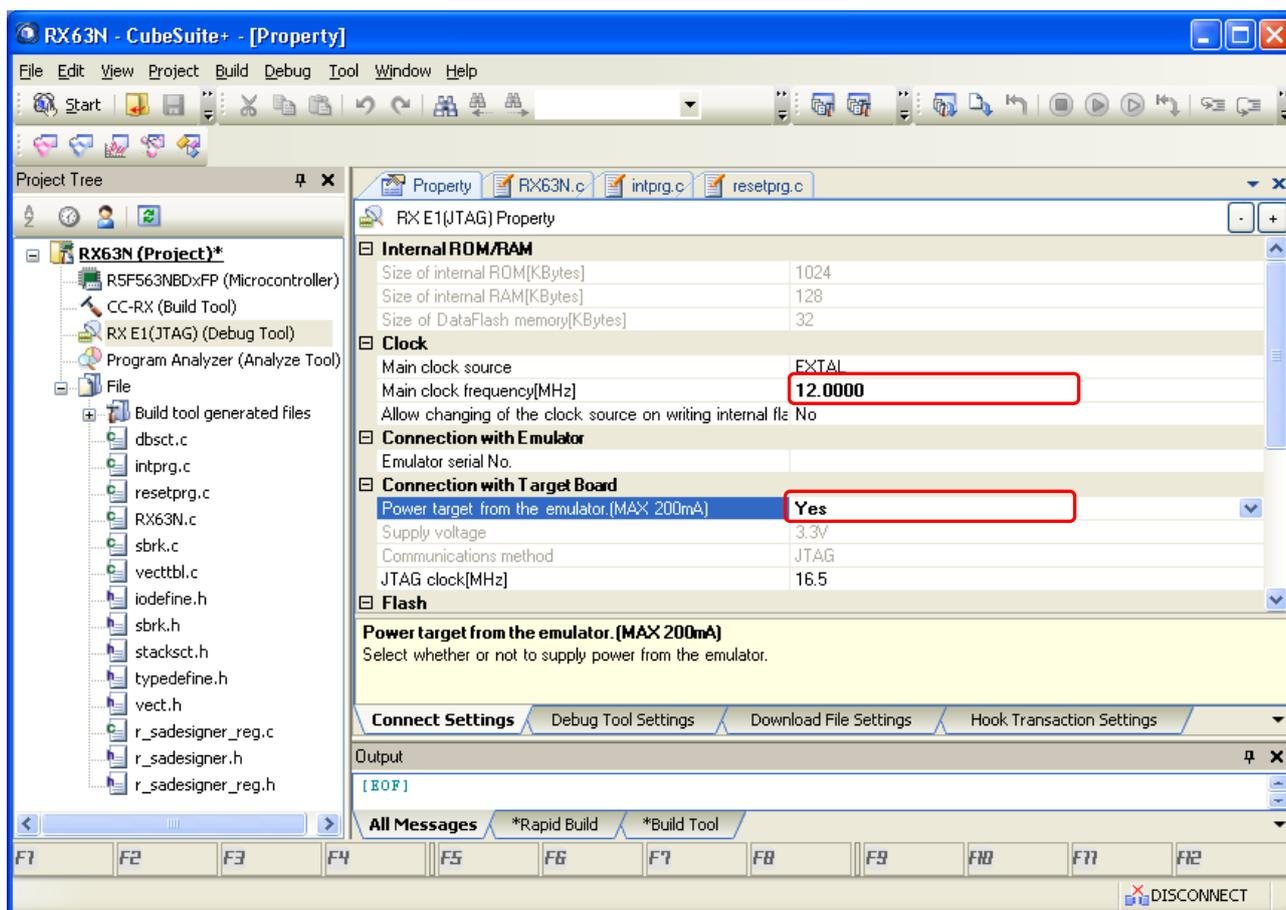


[RX Simulator(Debug Tool)] - [Using Debug Tool]

Choose the "RX E1(JTAG)".

## Smart Analog System development procedure by using SA-Designer (RX family)

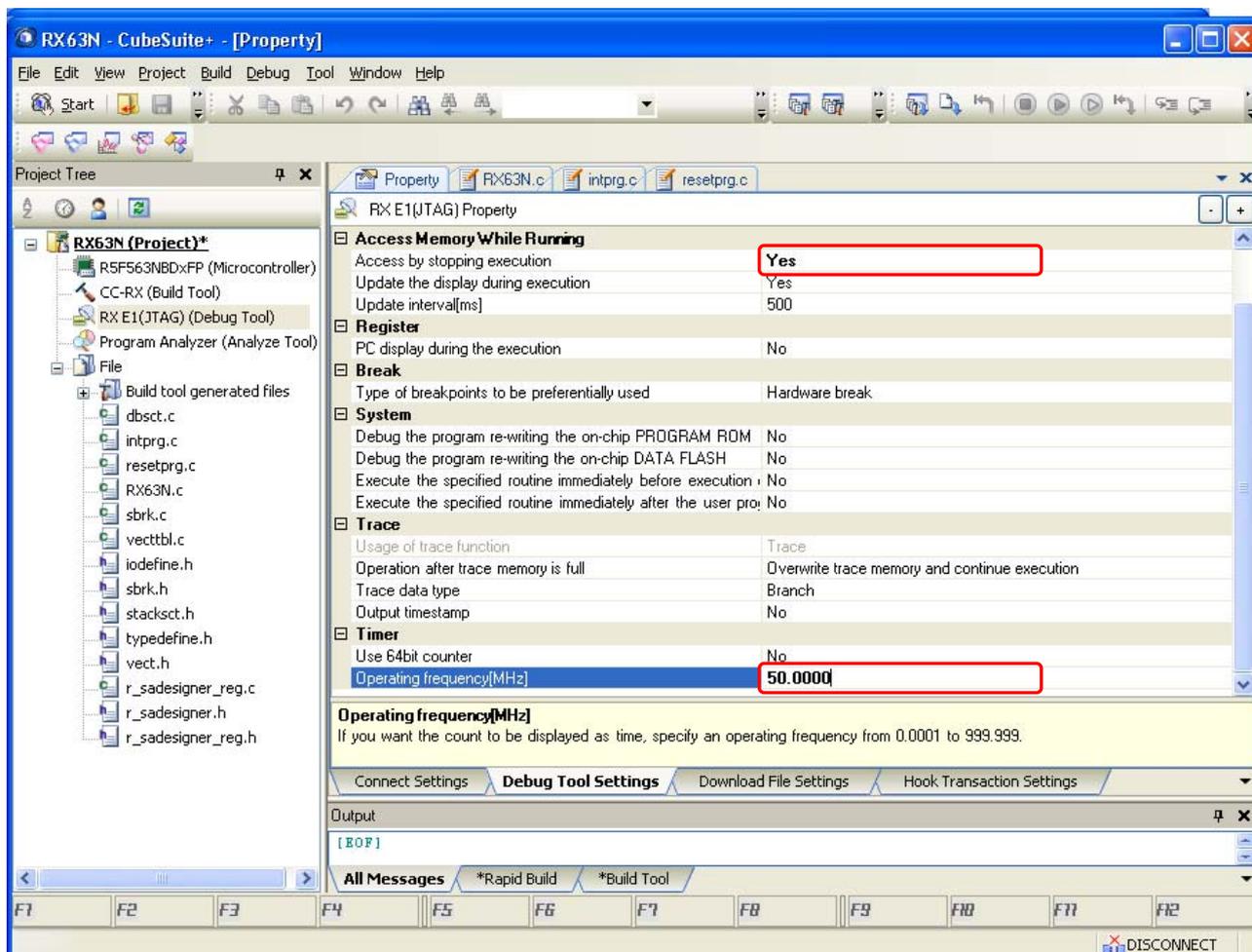
Set the clock and choose “Power target from the emulator” in Property window of Debug Tool.



[Clock] - [Main clock frequency]	Specify "12.0000".
[Connection with Target Board] - [Power target from the emulator]	Choose "YES".

## Smart Analog System development procedure by using SA-Designer (RX family)

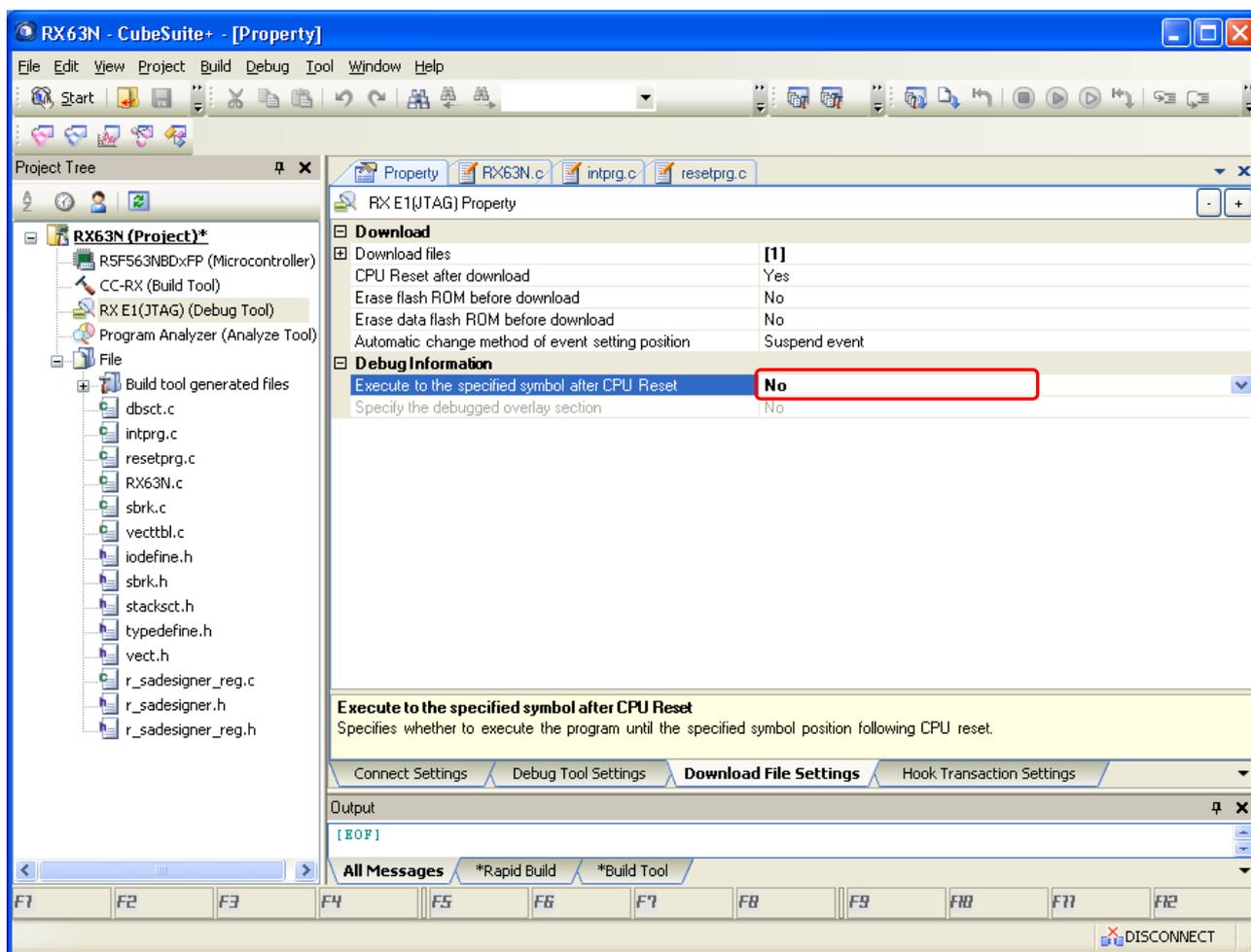
Set the Timer and Memory access in Property window of Debug Tool.



[Access Memory While Running] - [ Access by stopping execution]	Choose "YES".
[Timer] - [Operating frequency]	Specify "50.0000".

## Smart Analog System development procedure by using SA-Designer (RX family)

Set the “debug information” in Property window of Debug Tool.

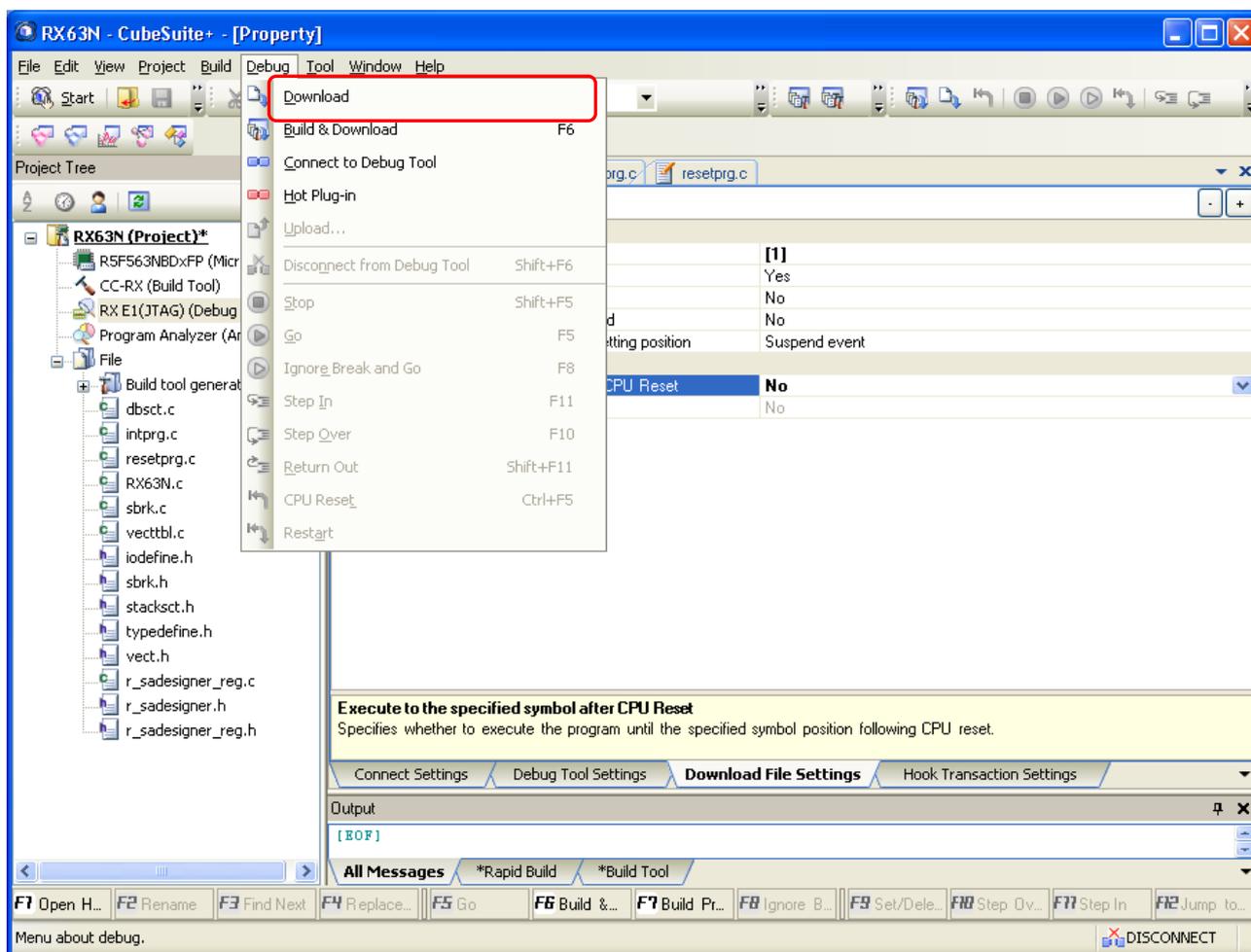


[Debug Information] - [Execute to the specified symbol after CPU Reset]

Choose "No".

## Smart Analog System development procedure by using SA-Designer (RX family)

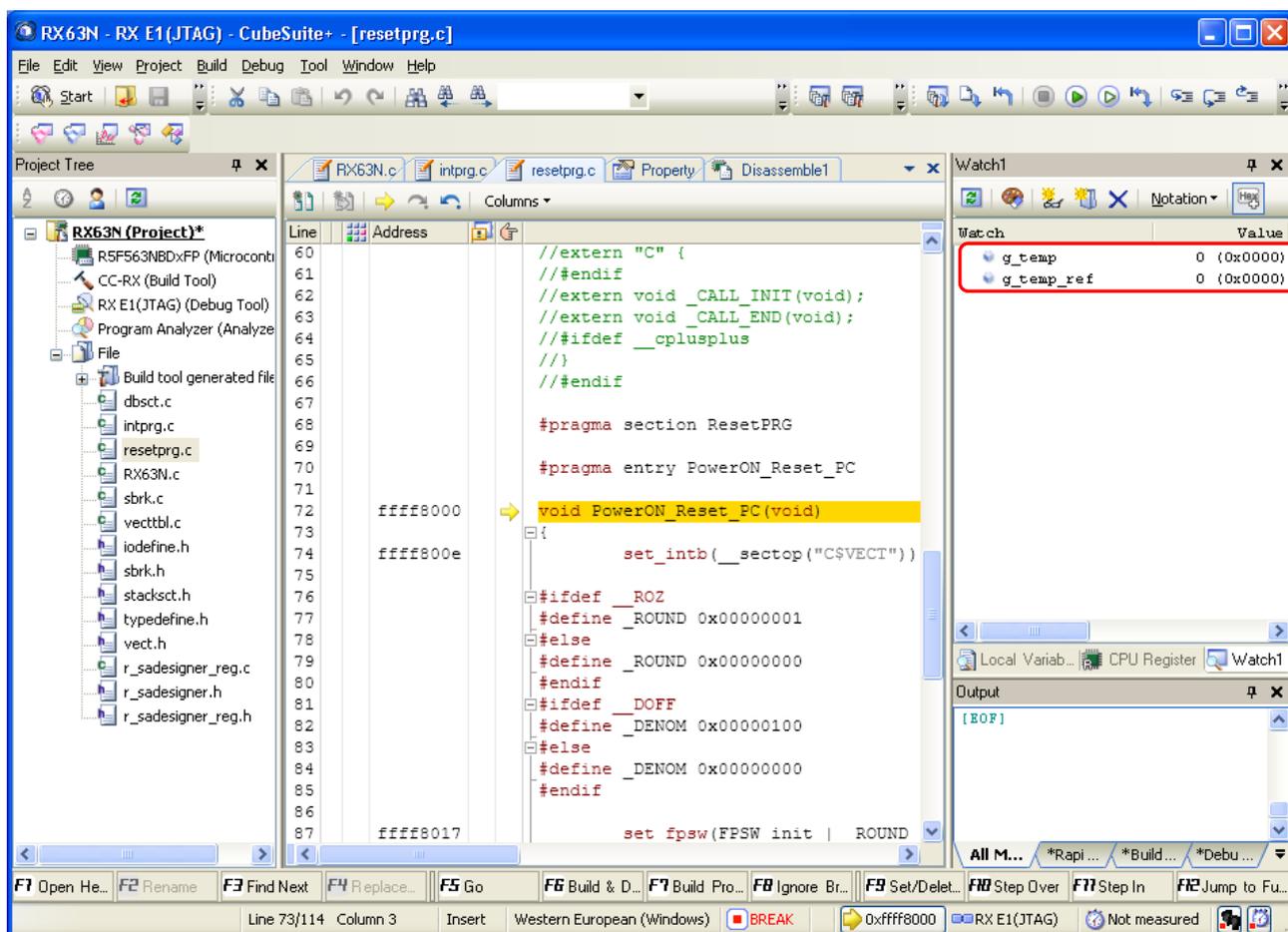
Choose [Download] from the [Debug]. Connect to the Debug Tool for downloading the load module.





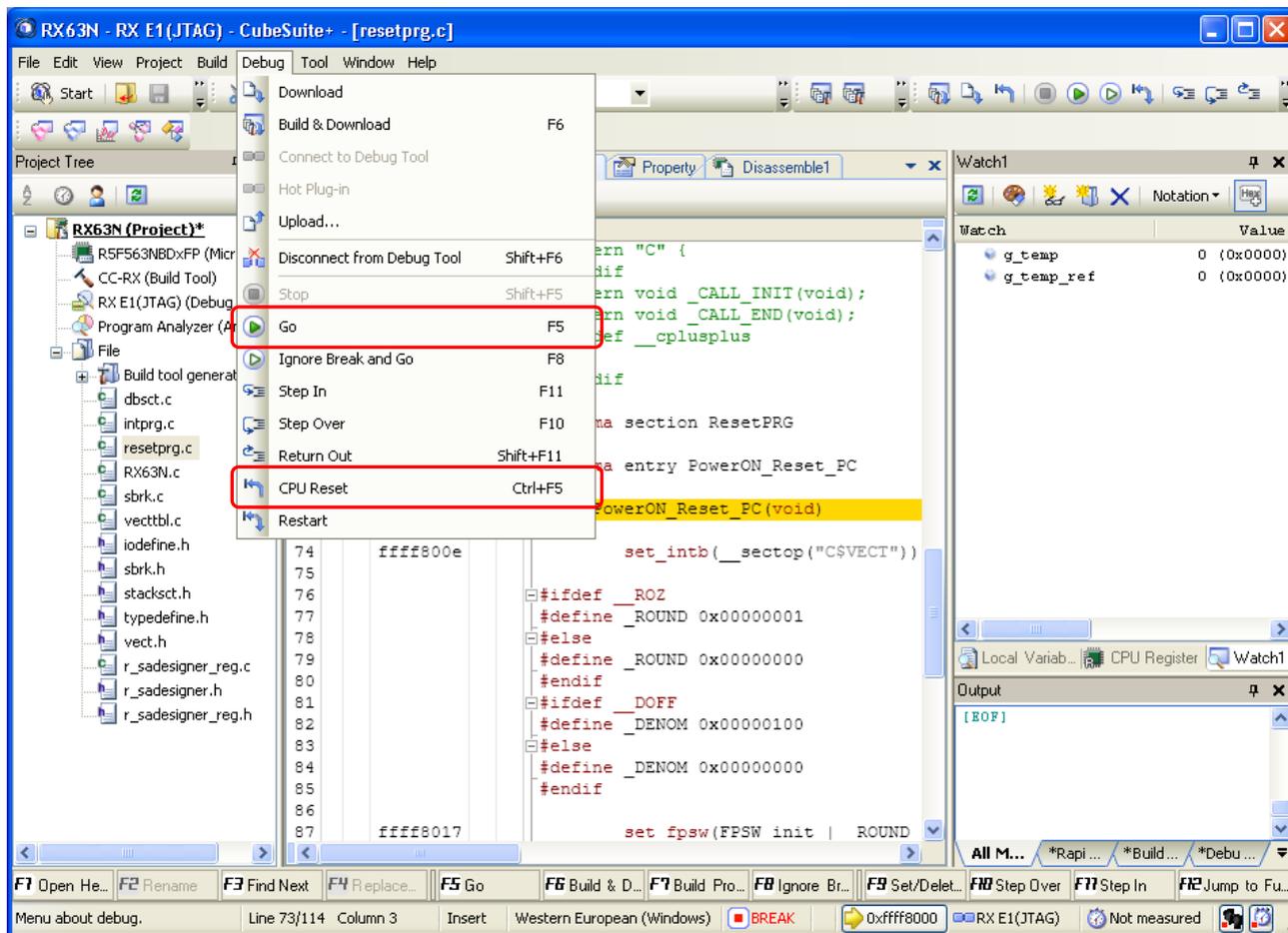
## Smart Analog System development procedure by using SA-Designer (RX family)

The variables will be displayed in the “Watch1 window”.



## (3) Run Program

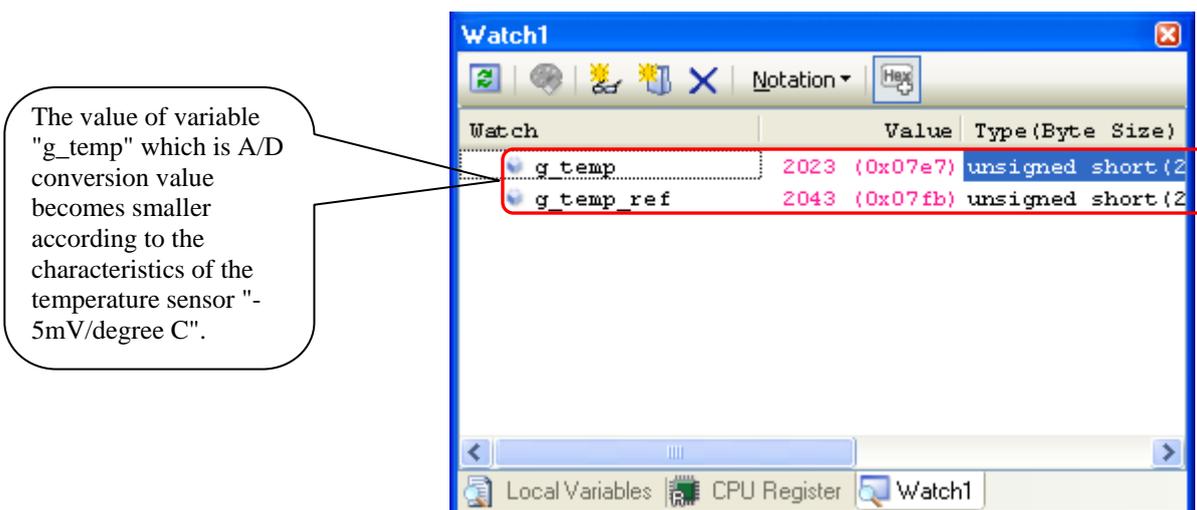
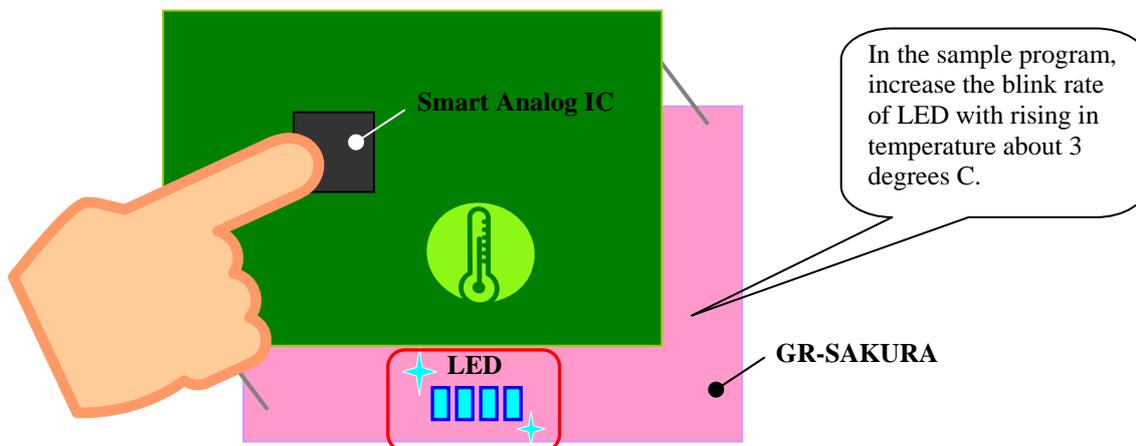
Check the system operations. "CPU Reset" must be chosen before execute the program.



Choose "CPU Reset" from "Debug" and then choose "Go" to execute the program.

## Smart Analog System development procedure by using SA-Designer (RX family)

Touch your finger to the microcomputer of "Smart Analog IC500". When you touch the "Smart Analog IC", then the temperature of the microcomputer will increase and get lower value of the variable "g\_temp" of A/D conversion. Also, the blink rate of LED will be increased.



### 3. Sample Programs

The followings are the sample programs which are used in the application note.

(1) **Function main (In addition to the main function of RX63N.c)**

```

#include <machine.h>
#include "iodefine.h"          /* RX63N I/O define table */
#include "r_sadesigner_reg.h"  /* Smart Analog include */
#include "r_sadesigner.h"      /* Smart Analog include */

void R_SAIC_Create(void);
void R_SAIC_Write(smartanalog_t * const p_saic_data);
void R_SAIC_Read(smartanalog_t * const p_saic_data, smartanalog_t * const p_saic_read_buf);
void hwinit(void);
extern smartanalog_t gp_smartanalog_data[];

#define led_0    PORTA.PODR.BIT.B0    /* LED0 */
#define led_1    PORTA.PODR.BIT.B1    /* LED1 */
#define led_2    PORTA.PODR.BIT.B2    /* LED2 */
#define led_3    PORTA.PODR.BIT.B6    /* LED3 */

/* Change the value according to the system * * * * * */
#define DEF_TMP 20
/* * * * * * */

volatile unsigned short g_temp      = 0;
volatile unsigned short g_temp_ref;
volatile unsigned int   g_count     = 0;
volatile unsigned int   g_timeofswitch = 10000;

void main(void)
{
    volatile short def;

    hwinit();

    R_SAIC_Create();          /* for Smart Analog */

    while(!g_temp) {
        nop();
    }
    g_temp_ref = g_temp;     /* read start condition */

    while(1) {
        def = g_temp_ref - g_temp;
        if ( g_count > g_timeofswitch ) {
            led_0 = ~led_0;
            led_1 = ~led_1;
            led_2 = ~led_2;
            led_3 = ~led_3;

            g_count = 0;

            if ( def > DEF_TMP ) {
                g_timeofswitch = 5000;
            } else {
                g_timeofswitch = 25000;
            }
        }
    }
}

```

### (2) Initialization function (Add to RX63N.c)

```

#define PSW_I_FLG 0x00010000
#define PSW_I_CLR 0x00000000
void hwinit(void)
{
    set_psw(PSW_I_CLR);

    SYSTEM.PRCR.WORD = 0xA503;          /* disable Register protection */
    SYSTEM.MSTPCRA.LONG = 0xFFFDFFFF;  /* enable MSTP S12AD */
    SYSTEM.MSTPCRB.LONG = 0xFFFDFFFF;  /* enable MSTP RSP10 */
    SYSTEM.MSTPCRC.LONG = 0xFFFF0000; /* */

    SYSTEM.SCKCR3.WORD = 0x0200;       /* select Main Clock */
    SYSTEM.MOSCCR.BIT.MOSTP = 0;       /* enable Main Clock */

    MPC.PWPR.BIT.BOWI = 0;            /* disable MPC protection */
    MPC.PWPR.BIT.PFSWE = 1;

    MPC.PC5PFS.BYTE = 0x0D;           /* set SPI RSPCKA/MOSIA/MISOA */
    MPC.PC6PFS.BYTE = 0x0D;
    MPC.PC7PFS.BYTE = 0x0D;
    MPC.P44PFS.BIT.ASEL = 1;         /* set Smart Analog TEMP_OUT */

    MPC.PWPR.BIT.BOWI = 1;            /* enable MPC protection */
    MPC.PWPR.BIT.PFSWE = 0;

    PORT1.PDR.BIT.B2 = 1;            /* init Port for SAIC RESET */
    PORT1.PMR.BIT.B2 = 0;
    PORT1.PODR.BIT.B2 = 1;

    SYSTEM.PRCR.WORD = 0xA500;       /* enable Register protection */

    PORTA.PODR.BYTE = 0;              /* set Port for LED */
    PORTA.PDR.BYTE = 0x47;

    PORT4.PDR.BIT.B4 = 0;            /* set Port for TEMP_OUT */
    PORT4.PMR.BIT.B4 = 0;

    PORTC.PDR.BYTE = 0x70;           /* set Port Output PC4, PC5, PC6 */
    PORTC.PMR.BYTE = 0xE0;           /* set Port General PC5, PC6, PC7 */

    PORTC.PODR.BIT.B4 = 1;           /* set Port PC4 for CS */
    /* set S12AD */
    S12AD.ADCSR.BYTE = 0;            /* clear ADST, CKS */
    S12AD.ADANSO.WORD = 0x0010;     /* set AN004 */
    S12AD.ADCSR.BYTE = 0xD0;        /* set ADST, ADCS, ADIE */

    led_0 = 1;
    led_2 = 1;

    set_psw(PSW_I_FLG);
}

```

### (3) Interrupt function (Add to intrpg.c)

```
// S12AD S12AD10
#include "iodefine.h"
extern volatile unsigned short g_temp;
extern volatile unsigned int g_count;
void Excep_S12AD_S12AD10(void) {
    g_temp = S12AD.ADDR4;
    g_count++;
}
```

### (4) Function SPI (Add to RX63N.c)

#### (a) R\_SAIC\_Create()

```
/* R_SAIC_Create() */
void R_SAIC_Create(void)
{
    volatile uint16_t w_count;

    PORT1.PODR.BIT.B2 = 0; /* Analog IC Reset */

    /* Change the waiting time according to the system */
    for (w_count = 0U; w_count < 0x82; w_count++)
    {
        nop();
    }

    PORT1.PODR.BIT.B2 = 1; /* Analog IC Reset release */

    R_SAIC_Write(gp_smartanalog_data);
}
```

### (b) R\_SAIC\_Write()

```

/*****
/* R_SAIC_Write(gp_smartanalogs_data); */
/*****
void R_SAIC_Write(smartanalogs_t * const p_saic_data)
{
    volatile uint8_t adrs;
    volatile uint8_t dat;
    volatile uint8_t wait;

    smartanalogs_t *p_saic_write;
    p_saic_write = p_saic_data;

    RSPIO.SPCR.BYTE = 0;          /* clear SPE, SPTIE */
                                /* set ICU */
    ICU.IPR[102].BYTE = 1;      /* set interrupt priority S12AD */
    ICU.IR[102].BIT.IR = 0;     /* clear interrupt S12AD */
    ICU.IR[40].BIT.IR = 0;      /* clear interrupt RSPI SPTI */
    ICU.IER[5].BIT.IEN0 = 1;    /* enable interrupt RSPI SPTI */

    ICU.IER[12].BIT.IEN6 = 1;   /* enable interrupt S12AD */

    RSPIO.SPCR.BYTE = 0x0B;     /* set SPMS, MSTR, TXMD */
    RSPIO.SPCMD0.WORD = 0x0703; /* set SPB, LSBF, CPOL, CPHA */
    RSPIO.SPBR = 0x05;

    RSPIO.SPCR.BIT.SPE = 1;     /* set SPE */
    RSPIO.SPCR.BIT.SPTIE = 1;  /* set SPTIE */

    while (p_saic_write->address != 0xff)
    {
        PORTC.PODR.BIT.B4 = 0;
        for (wait = 0U; wait < 10U; wait++) /* SA Stable waiting time (tSKA) */
        {
            nop();
        }

        adrs = (p_saic_write->address & 0x7f) | 0x80; /* 0x80 data write mode*/
        RSPIO.SPDR.WORD.H = adrs; /* send SAIC Address data */

        while (ICU.IR[40].BIT.IR == 0U); /* wait for CSI send */
        ICU.IR[40].BIT.IR = 0U;

        dat = p_saic_write->data;
        RSPIO.SPDR.WORD.H = dat;

        while (ICU.IR[40].BIT.IR == 0U); /* wait for CSI send */
        ICU.IR[40].BIT.IR = 0U;

        for (wait = 0U; wait < 10U; wait++) /* SA Stable waiting time (tKSA) */
        {
            nop();
        }
        PORTC.PODR.BIT.B4 = 1; /* SAIC CS=H */
        for (wait = 0U; wait < 10U; wait++) /* SA Stable waiting time (tSHA) */
        {
            nop();
        }
        p_saic_write++;
    }
    RSPIO.SPCR.BIT.SPTIE = 0;
    RSPIO.SPCR.BIT.SPE = 0;
}

```

## (c) R\_SAIC\_Read()

```

/*****
/* R_SAIC_Read(gp_smartanalog_data, saic_read_buf);
/*****
void R_SAIC_Read(smartanalog_t * const p_saic_data, smartanalog_t * const p_saic_read_buf)
{
    volatile uint8_t adrs;
    volatile uint8_t wait;
    smartanalog_t *p_saic_write;
    smartanalog_t *p_saic_read;

    p_saic_write = p_saic_data;
    p_saic_read = p_saic_read_buf;

    RSP10. SPCR. BYTE = 0;          /* clear SPE, SPTIE          */
                                   /* set ICU                   */
    ICU. IPR[102]. BYTE = 1;        /* set interrupt priority S12AD */
    ICU. IR[102]. BIT. IR = 0;      /* clear interrupt S12AD      */
    ICU. IR[40]. BIT. IR = 0;       /* clear interrupt RSP1 SPTI   */
    ICU. IER[5]. BIT. IEN0 = 1;     /* enable interrupt RSP1 SPTI */

    RSP10. SPCR. BYTE = 0x09;       /* set SPMS, MSTR           */
    RSP10. SPCMD0. WORD = 0x0703;   /* set SPB, LSBF, CPOL, CPHA */
    RSP10. SPBR = 0x05;

    RSP10. SPCR. BIT. SPE = 1;      /* set SPE                   */
    RSP10. SPCR. BIT. SPTIE = 1;    /* set SPTIE                 */

    while (p_saic_write->address != 0xff)
    {
        PORTC. PODR. BIT. B4= 0;
        for (wait = 0U; wait < 10U; wait++) /* SA Stable waiting time (tSKA) */
        {
            nop();
        }

        adrs = (p_saic_write->address) & 0x7f;
        p_saic_read->address = adrs;          /* send SAIC Address data      */
        RSP10. SPDR. WORD. H = adrs;         /* send SAIC Address data      */

        while (ICU. IR[40]. BIT. IR == 0U); /* wait for CSI send          */
        ICU. IR[40]. BIT. IR = 0U;

        RSP10. SPDR. WORD. H = 0xff;         /* send CSI dummy data        */

        while (ICU. IR[40]. BIT. IR == 0U); /* wait for CSI send          */
        ICU. IR[40]. BIT. IR = 0U;

        p_saic_read->data = (unsigned char)RSP10. SPDR. WORD. H;

        for (wait = 0U; wait < 10U; wait++) /* SA Stable waiting time (tKSA) */
        {
            nop();
        }
        PORTC. PODR. BIT. B4= 1;            /* SAIC CS=H                  */
        for (wait = 0U; wait < 10U; wait++) /* SA Stable waiting time (tSHA) */
        {
            nop();
        }
        p_saic_write++;
        p_saic_read++;
    }
    RSP10. SPCR. BIT. SPTIE = 0;
    RSP10. SPCR. BIT. SPE = 0;
}

```

**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 Record

Rev.	Date	Description	
		Page	Summary
1.00	Apr 3, 2013	—	First edition issued

## General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

### 1. Handling of Unused Pins

Handle unused pins in accord 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 one with a different type number, confirm that the change will not lead to problems.

- The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

## Notice

1. Descriptions of circuits, 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 circuits, 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 circuits, 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 in 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 license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should 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 products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.  
"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.  
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime 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 (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it 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, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
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 use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them 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 hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products 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 regulate the inclusion or use of controlled 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 domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development 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 procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify 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 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 product(s)" 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.

**Renesas Electronics America Inc.**  
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.  
Tel: +1-408-588-6000, Fax: +1-408-588-6130

**Renesas Electronics Canada Limited**  
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada  
Tel: +1-905-898-5441, Fax: +1-905-898-3220

**Renesas Electronics Europe Limited**  
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K  
Tel: +44-1628-651-700, Fax: +44-1628-651-804

**Renesas Electronics Europe GmbH**  
Arcadiastrasse 10, 40472 Düsseldorf, Germany  
Tel: +49-211-65030, Fax: +49-211-6503-1327

**Renesas Electronics (China) Co., Ltd.**  
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China  
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

**Renesas Electronics (Shanghai) Co., Ltd.**  
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China  
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

**Renesas Electronics Hong Kong Limited**  
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong  
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

**Renesas Electronics Taiwan Co., Ltd.**  
13F, No. 363, Fu Shing North Road, Taipei, Taiwan  
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

**Renesas Electronics Singapore Pte. Ltd.**  
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949  
Tel: +65-6213-0200, Fax: +65-6213-0300

**Renesas Electronics Malaysia Sdn.Bhd.**  
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia  
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

**Renesas Electronics Korea Co., Ltd.**  
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea  
Tel: +82-2-558-3737, Fax: +82-2-558-5141