RL78/G14 群

I2C Repeated START 信号

R01AN1947CC0100
Rev.1.00
2014.01.10

要点

本应用说明将从 RL78 的 IICA 模块的主机和从机方面,介绍如何从代码生成器生成的 I2C 示例代码, 添加 Repeated START 状态的对应。

目标设备

RL78/G14

目录

1. I2C Repeated START 信号简介 2
2. I2C 主模式示例程序修改 3
3. I2C 主模式与 Repeated START 状态流程 5
4. I2C 从模式示例程序修改 6
5. I2C 从模式与 Repeated START 状态程序流程 7
6. I2C 主机参考例程 8
7. I2C 从机参考例程 18

公司主页和网咨询窗口 26

1. I2C Repeated START 信号简介

在一个 I2C 传输中，经常需要先发送一个命令，然后读回一个答案。这是需要在没有被另一设备（多主机）中断操作风险下,完成整个工作。 I2C 协议定义了一个所谓的”Repeated START”状态。

RL78 族 IICA 模块能够支持 I2C Repeated START 条件, 如下:

ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	SP

ST:	Start condition
AD6 to AD0:	Address
R/$\overline{W}$:	Transfer direction specification
$\overline{ACK}$:	Acknowledge
D7 to D0:	Data
SP:	Stop condition

然而，通过代码生成器生成的 I2C 示例代码并不支持及对应 I2C Repeated START 状态。但对应 Repeated START 状态可以通过修改示例程序来实现的。本应用笔记描述了如何修改由代码生成器生成，支持 I2C Repeated START（主从）的示例代码。

2. I2C 主模式示例程序修改

“总线忙”检查

在由代码生成器生成的 I2C 主控的示例程序，IICBSY 寄存器将发送 IIC 从地址之前进行检查。如果 IICBSY 为’1’，将返回错误值。

然而，如 Repeated START 状态下，总线状态为忙碌。该程序应忽略 IICBSY 检查。我们应该将下面的 R_IICA0_Master_Send（）和 R_IICA0_Master_Receive（）的部分代码, 移动到一个新的子程序 R_IICA0_Busy_Check（）。

```
MD_STATUS R_IICA0_Master_Send(uint8_t adr, uint8_t * const tx_buf, uint16_t tx_num, uint8_t wait)
{
    MD_STATUS status = MD_OK;

    IICAMK0 = 1U;      /* disable INTIICA0 interrupt */

    if (1U == IICBSY0)
    {
        /* Check bus busy */
        IICAMK0 = 0U;  /* enable INTIICA0 interrupt */
        status = MD_ERROR1;
    }
    else if ((1U == SPT0) || (1U == STT0))
    {
        /* Check trigger */
        IICAMK0 = 0U;  /* enable INTIICA0 interrupt */
        status = MD_ERROR2;
    }
    else
    {
        STT0 = 1U;     /* send IICA0 start condition */
        IICAMK0 = 0U;  /* enable INTIICA0 interrupt */

        /* Wait */
        while (wait--)
        {
            ;
        }
```

此部分代码移到 *R_IICA0_Busy_Check()*

以下代码是新创建的函数 *R_IICA0_Busy_Check().*

```
/***********************************************************************************************************************
* Function Name: R_IICA0_Busy_Check
* Description  : This function starts to check bus busy.
* Arguments    : -
* Return Value : status -
*                    MD_OK or MD_ERROR1 or MD_ERROR2 or MD_ERROR3
***********************************************************************************************************************/
MD_STATUS R_IICA0_Busy_Check(void)
{
    MD_STATUS status = MD_OK;

    IICAMK0 = 1U;      /* disable INTIICA0 interrupt */

    if (1U == IICBSY0)
    {
        /* Check bus busy */
        IICAMK0 = 0U;  /* enable INTIICA0 interrupt */
        status = MD_ERROR1;
    }
    else if ((1U == SPT0) || (1U == STT0))
    {
        /* Check trigger */
        IICAMK0 = 0U;  /* enable INTIICA0 interrupt */
        status = MD_ERROR2;
    }

    return status;
}
```

跳过'停止'状态

对于 Repeated START 状态，'停止'状态应 I2C 命令帧的结束送出去。在 r_iica0_callback_master_sendend（）和 r_iica0_callback_master_receiveend（）函数里的处理停止状态代码, 应该被移动到一个新的子程序 R_IICA0_StopCondition（）。

```
/***********************************************************************************************************************
* Function Name: r_iica0_callback_master_receiveend
* Description  : This function is a callback function when IICA0 finishes master reception.
* Arguments    : None
* Return Value : None
***********************************************************************************************************************/
static void r_iica0_callback_master_receiveend(void)
{
    SPT0 = 1U;
    /* Start user code. Do not edit comment generated here */
    IIC_flg_end = 1;
    /* End user code. Do not edit comment generated here */
}

/***********************************************************************************************************************
* Function Name: r_iica0_callback_master_sendend
* Description  : This function is a callback function when IICA0 finishes master transmission.
* Arguments    : None
* Return Value : None
***********************************************************************************************************************/
static void r_iica0_callback_master_sendend(void)
{
    SPT0 = 1U;
    /* Start user code. Do not edit comment generated here */
    IIC_flg_end = 1;
    /* End user code. Do not edit comment generated here */
}
```

此部分代码移到 *R_IICA0_StopCondition()*

以下代码是新创建的函数 *R_IICA0_StopCondition()*.

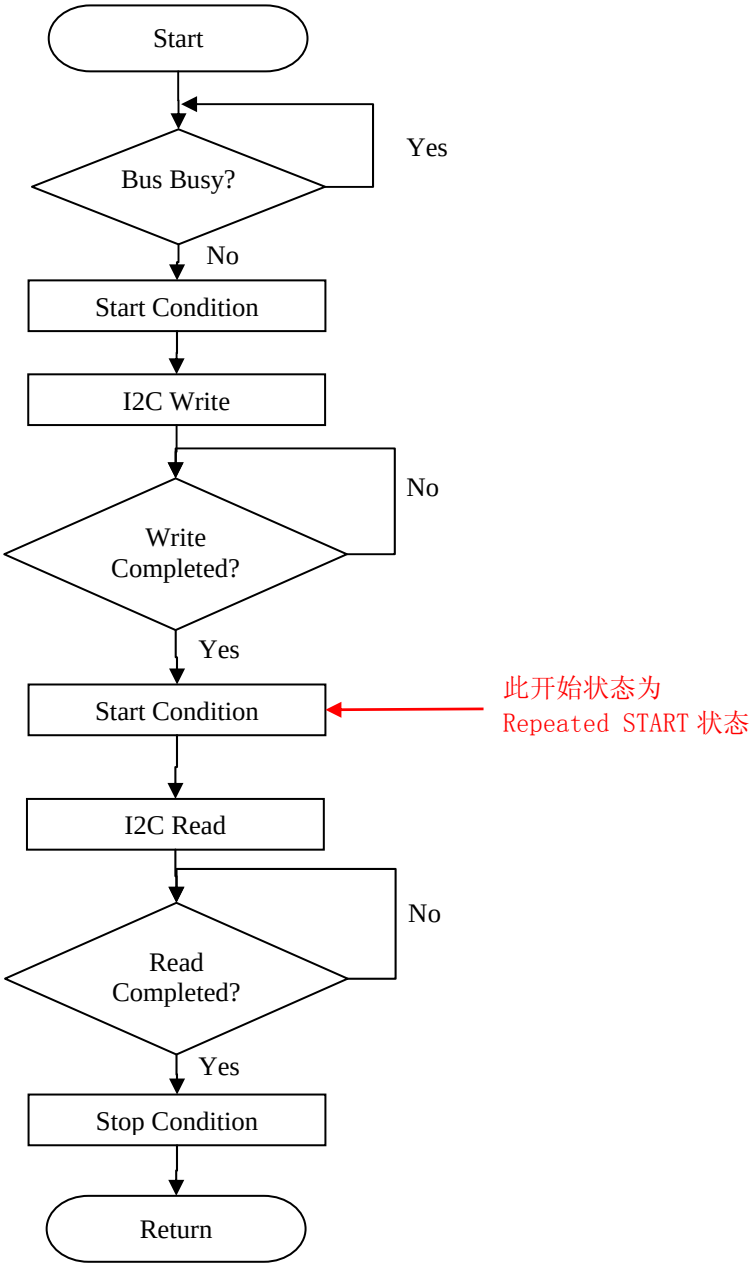
```
/***********************************************************************************************************************
* Function Name: R_IICA0_StopCondition
* Description  : This function sets IICA0 stop condition flag.
* Arguments    : None
* Return Value : None
***********************************************************************************************************************/
void R_IICA0_StopCondition(void)
{
    SPT0 = 1U;    /* set stop condition flag */
}
```

主程序的 I2C 发送命令与 Repeated START 状态

对于使用修改后的代码发送的 I2C，请参阅下面的代码示例。

```
if (R_IICA0_Busy_Check() == MD_OK)  //Check bus busy
{
        IIC_flg_end = 0;
        R_IICA0_Master_Send(0xA0, &IIC_data[0], 1, 0);    //Send data to slave

        while(IIC_flg_end == 0)         //Wait for transmit end
                NOP();
        IIC_flg_end = 0;

        IIC_data[0]= 0;
        R_IICA0_Master_Receive(0xA0, &IIC_data[0], 1, 0);   //Read data from slave
        timer = 0;
        while(IIC_flg_end == 0)         //Wait for recieved end
                NOP();
        IIC_flg_end = 0;

        R_IICA0_StopCondition();        //Send stop condition
}
```

3. I2C 主模式与 Repeated START 状态流程



4. I2C 从模式示例程序修改

添加检查 Repeated START 状态

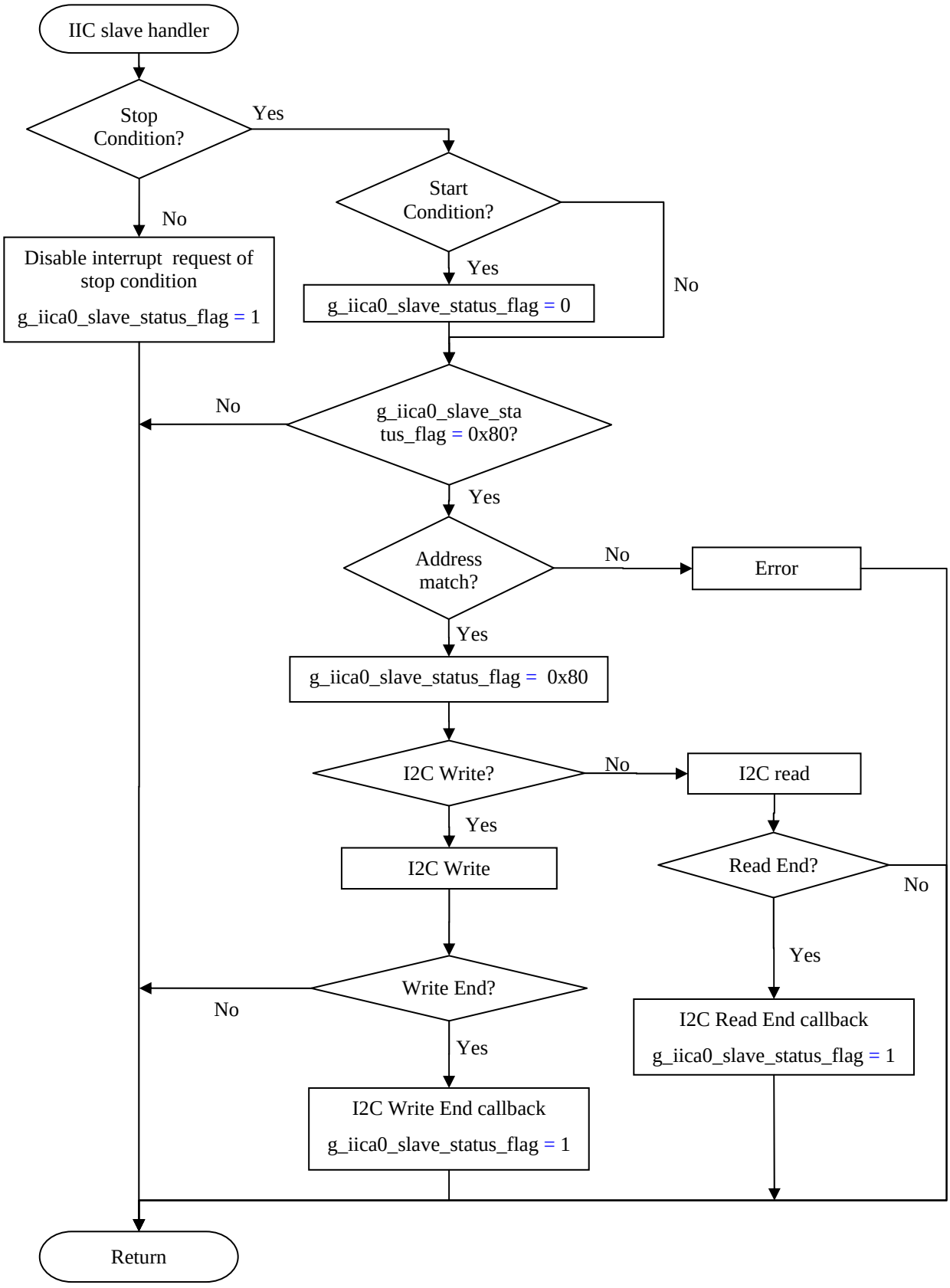
在由代码生成器生成的 I2C 从示例程序，并没有检查 START 状态。当接收到 I2C 主设备 START 或 Repeated START 状态是, I2C 的从状态机将无法复位，并在 Repeated START 状态引起问题。

为了支持 Repeated START 状态，我们可以添加程序检查 Repeated START 状态以及状态机复位手段。修改代码如下所示。

```
/***********************************************************************************************************************
* Function Name: iica0_slave_handler
* Description  : This function is IICA0 slave handler.
* Arguments    : None
* Return Value : None
***********************************************************************************************************************/
static void iica0_slave_handler(void)
{
    /* Control for stop condition */
    if (1U == SPD0)
    {
        /* Get stop condition */
        SPIE0 = 0U;
        g_iica0_slave_status_flag = 1U;
    }
    else
    {
        if(1U == STD0)/* start or restart condition*/
            g_iica0_slave_status_flag = 0U;  //Reset slave status flag

        if ((g_iica0_slave_status_flag & _80_IICA_ADDRESS_COMPLETE) == 0U)
        {
            if (1U == COI0)
            {
                SPIE0 = 1U;
                g_iica0_slave_status_flag |= _80_IICA_ADDRESS_COMPLETE;

                if (1U == TRC0)
                {
                    WTIM0 = 1U;

                    if (g_iica0_tx_cnt > 0U)
                    {
                        IICA0 = *gp_iica0_tx_address;
                        gp_iica0_tx_address++;
                        g_iica0_tx_cnt--;
                    }
                    else
                    {
                        r_iica0_callback_slave_sendend();
                        WREL0 = 1U;
                    }
                }
```

添加代码以检查 START 和 Repeated START 状态。如果发现 START 和 Repeated START 的状态，复位状态标志。

5. I2C 从模式与 Repeated START 状态程序流程



6. I2C 主机参考例程

```
/***********************************************************************
* File Name    : r_main.c
* Version      : CodeGenerator for RL78/I1A V2.00.00.04 [21 Feb 2013]
* Device(s)    : R5F107AE
* Tool-Chain   : CA78K0R
* Description  : This file implements main function.
* Creation Date: 13/9/2013
***********************************************************************/

/***********************************************************************
Pragma directive
***********************************************************************/

/***********************************************************************
Includes
***********************************************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_cgc.h"
#include "r_cg_serial.h"
#include "r_cg_userdefine.h"

/***********************************************************************
Global variables and functions
***********************************************************************/
uint8_t IIC_data[250], timer,i;
uint16_t l;
uint8_t IIC_flg_end = 0;


/***********************************************************************
* Function Name: main
* Description  : This function implements main function.
* Arguments    : None
* Return Value : None
***********************************************************************/
void main(void)
{
    for(i=0; i<250; i++)
       IIC_data[i]=0;

    timer = 0;

    while (1U)
    {
          for(l=0; l<55000; l++)
             NOP();

          if (R_IICA0_Busy_Check() == MD_OK)  //Check bus busy
          {
             timer = 0;
             IIC_flg_end = 0;
             //Send data to slave
             R_IICA0_Master_Send(0xA0, &IIC_data[0], 1, 0);
             //Wait for transmit end
             while(IIC_flg_end == 0)
```

```
                NOP();
            IIC_flg_end = 0;

            IIC_data[0]= 0;
            //Read data from slave
            R_IICA0_Master_Receive(0xA0, &IIC_data[0], 1, 0);
            timer = 0;
            //Wait for recieved end
            while(IIC_flg_end == 0)
                NOP();
            IIC_flg_end = 0;

            R_IICA0_StopCondition();   //Send stop condition
        }
    }
}
```

```
/***********************************************************************
* File Name    : r_cg_serial.c
* Version      : CodeGenerator for RL78/I1A V2.00.00.04 [21 Feb 2013]
* Device(s)    : R5F107AE
* Tool-Chain   : CA78K0R
* Description  : This file implements device driver for Serial module.
* Creation Date: 13/9/2013
***********************************************************************/

/***********************************************************************
Pragma directive
***********************************************************************/

/***********************************************************************
Includes
***********************************************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_serial.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefine.h"

/***********************************************************************
Global variables and functions
***********************************************************************/
volatile uint8_t   g_iica0_master_status_flag; /* iica0 master flag */
volatile uint8_t   g_iica0_slave_status_flag;  /* iica0 slave flag */
volatile uint8_t * gp_iica0_rx_address;        /* iica0 receive buffer address */
volatile uint16_t  g_iica0_rx_len;             /* iica0 receive data length */
volatile uint16_t  g_iica0_rx_cnt;             /* iica0 receive data count */
volatile uint8_t * gp_iica0_tx_address;        /* iica0 send buffer address */
volatile uint16_t  g_iica0_tx_cnt;             /* iica0 send data count */
/* Start user code for global. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/***********************************************************************
* Function Name: R_IICA0_Create
* Description  : This function initializes the IICA0 module.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_IICA0_Create(void)
{
    IICA0EN = 1U;  /* supply IICA0 clock */
    IICE0 = 0U;    /* disable IICA0 operation */
    IICAMK0 = 1U;  /* disable INTIICA0 interrupt */
    IICAIF0 = 0U;  /* clear INTIICA0 interrupt flag */
    /* Set INTIICA0 high priority */
    IICAPR10 = 0U;
    IICAPR00 = 0U;
    /* Set SCLA0, SDAA0 pin */
    P1 &= 0xFCU;
    PM1 &= 0xFCU;
    SMC0 = 1U;
    IICWL0 = _15_IICA0_IICWL_VALUE;
    IICWH0 = _14_IICA0_IICWH_VALUE;
    DFC0 = 0U;     /* digital filter off */
    IICCTL01 |= _01_IICA_fCLK_HALF;
```

```
    SVA0 = _10_IICA0_MASTERADDRESS;
    STCEN0 = 1U;
    IICRSV0 = 1U;
    SPIE0 = 0U;
    WTIM0 = 1U;
    ACKE0 = 1U;
    IICAMK0 = 0U;
    IICE0 = 1U;
    LREL0 = 1U;
    /* Set SCLA0, SDAA0 pin */
    PM1 &= 0xFCU;
}

/***********************************************************************
* Function Name: R_IICA0_Stop
* Description  : This function stops IICA0 module operation.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_IICA0_Stop(void)
{
    IICE0 = 0U;    /* disable IICA0 operation */
}

/***********************************************************************
* Function Name: R_IICA0_StopCondition
* Description  : This function sets IICA0 stop condition flag.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_IICA0_StopCondition(void)
{
    SPT0 = 1U;    /* set stop condition flag */
}

/***********************************************************************
* Function Name: R_IICA0_Busy_Check
* Description  : This function starts to check bus busy.
* Arguments    : -
* Return Value : status -
*                    MD_OK or MD_ERROR1 or MD_ERROR2 or MD_ERROR3
***********************************************************************/
MD_STATUS R_IICA0_Busy_Check(void)
{
    MD_STATUS status = MD_OK;

    IICAMK0 = 1U;      /* disable INTIICA0 interrupt */

    if (1U == IICBSY0)
    {
        /* Check bus busy */
        IICAMK0 = 0U;  /* enable INTIICA0 interrupt */
        status = MD_ERROR1;
    }
    else if ((1U == SPT0) || (1U == STT0))
    {
        /* Check trigger */
        IICAMK0 = 0U;  /* enable INTIICA0 interrupt */
```

```
        status = MD_ERROR2;
    }

    return status;
}


/***********************************************************************
* Function Name: R_IICA0_Master_Send
* Description  : This function starts to send data as master mode.
* Arguments    : tx_buf -
*                    transfer buffer pointer
*                tx_num -
*                    buffer size
*                wait -
*                    wait for start condition
* Return Value : status -
*                    MD_OK or MD_ERROR1 or MD_ERROR2 or MD_ERROR3
***********************************************************************/
MD_STATUS R_IICA0_Master_Send(uint8_t adr, uint8_t * const tx_buf, uint16_t tx_num,
uint8_t wait)
{
    MD_STATUS status = MD_OK;

    IICAMK0 = 1U;      /* disable INTIICA0 interrupt */
    STT0 = 1U;     /* send IICA0 start condition */
    IICAMK0 = 0U;  /* enable INTIICA0 interrupt */

    /* Wait */
    while (wait--)
    {
       ;
    }

    if(0U == STD0)
    {
        status = MD_ERROR3;
    }

    /* Set parameter */
    g_iica0_tx_cnt = tx_num;
    gp_iica0_tx_address = tx_buf;
    g_iica0_master_status_flag = _00_IICA_MASTER_FLAG_CLEAR;
    adr &= (uint8_t)~0x01U; /* set send mode */
    IICA0 = adr;           /* send address */

    return (status);
}
```

```
/***********************************************************************
* Function Name: R_IICA0_Master_Receive
* Description  : This function starts to receive IICA0 data as master mode.
* Arguments    : rx_buf -
*                    receive buffer pointer
*                rx_num -
*                    buffer size
*                wait -
*                    wait for start condition
* Return Value : status -
*                    MD_OK or MD_ERROR1 or MD_ERROR2 or MD_ERROR3
***********************************************************************/
MD_STATUS R_IICA0_Master_Receive(uint8_t adr, uint8_t * const rx_buf, uint16_t rx_num,
uint8_t wait)
{
    MD_STATUS status = MD_OK;

    IICAMK0 = 1U;  /* disable INTIIA0 interrupt */

    {
        STT0 = 1U;     /* set IICA0 start condition */
        IICAMK0 = 0U;  /* enable INTIIA0 interrupt */

        /* Wait */
        while (wait--)
        {
            ;
        }

        if(0U == STD0)
        {
            status = MD_ERROR3;
        }

        /* Set parameter */
        g_iica0_rx_len = rx_num;
        g_iica0_rx_cnt = 0U;
        gp_iica0_rx_address = rx_buf;
        g_iica0_master_status_flag = _00_IICA_MASTER_FLAG_CLEAR;
        adr |= 0x01U;    /* set receive mode */
        IICA0 = adr;    /* receive address */
    }

    return (status);
}
```

```
/***********************************************************************
* File Name    : r_cg_serial_user.c
* Version      : CodeGenerator for RL78/I1A V2.00.00.04 [21 Feb 2013]
* Device(s)    : R5F107AE
* Tool-Chain   : CA78K0R
* Description  : This file implements device driver for Serial module.
* Creation Date: 13/9/2013
***********************************************************************/

/***********************************************************************
Pragma directive
***********************************************************************/
#pragma interrupt INTIICA0 r_iica0_interrupt

/***********************************************************************
Includes
***********************************************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_serial.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefine.h"

/***********************************************************************
Global variables and functions
***********************************************************************/
extern volatile uint8_t   g_iica0_master_status_flag;  /* iica0 master flag */
extern volatile uint8_t   g_iica0_slave_status_flag;   /* iica0 slave flag */
extern volatile uint8_t * gp_iica0_rx_address;         /* iica0 receive buffer
address */
extern volatile uint16_t  g_iica0_rx_cnt;              /* iica0 receive data
length */
extern volatile uint16_t  g_iica0_rx_len;              /* iica0 receive data
count */
extern volatile uint8_t * gp_iica0_tx_address;         /* iica0 send buffer
address */
extern volatile uint16_t  g_iica0_tx_cnt;              /* iica0 send data
count */
extern uint8_t IIC_flg_end;

/***********************************************************************
* Function Name: r_iica0_interrupt
* Description  : This function is INTIICA0 interrupt service routine.
* Arguments    : None
* Return Value : None
***********************************************************************/
__interrupt static void r_iica0_interrupt(void)
{
      P20 = 0xff;
    if ((IICS0 & _80_IICA_STATUS_MASTER) == 0x80U)
    {
        iica0_master_handler();
    }
        P20 = 0x00;
}
```

```
/***********************************************************************
* Function Name: iica0_master_handler
* Description  : This function is IICA0 master handler.
* Arguments    : None
* Return Value : None
***********************************************************************/
static void iica0_master_handler(void)
{
    /* Control for communication */
    if ((0U == IICBSY0) && (g_iica0_tx_cnt != 0U))
    {
        r_iica0_callback_master_error(MD_SPT);
    }
    /* Control for sended address */
    else
    {
        if ((g_iica0_master_status_flag & _80_IICA_ADDRESS_COMPLETE) == 0U)
        {
            if (1U == ACKD0)
            {
                g_iica0_master_status_flag |= _80_IICA_ADDRESS_COMPLETE;
                if (1U == TRC0)
                {
                    WTIM0 = 1U;
                    if (g_iica0_tx_cnt > 0U)
                    {
                        IICA0 = *gp_iica0_tx_address;
                        gp_iica0_tx_address++;
                        g_iica0_tx_cnt--;
                    }
                    else
                    {
                        r_iica0_callback_master_sendend();
                    }
                }
                else
                {
                    ACKE0 = 1U;
                    WTIM0 = 0U;
                    WREL0 = 1U;
                }
            }
            else
            {
                r_iica0_callback_master_error(MD_NACK);
            }
        }
        else
        {
            /* Master send control */
            if (1U == TRC0)
            {
                if ((0U == ACKD0) && (g_iica0_tx_cnt != 0U))
                {
                    r_iica0_callback_master_error(MD_NACK);
                }
                else
                {
```

```
                    if (g_iica0_tx_cnt > 0U)
                    {
                        IICA0 = *gp_iica0_tx_address;
                        gp_iica0_tx_address++;
                        g_iica0_tx_cnt--;
                    }
                    else
                    {
                        r_iica0_callback_master_sendend();
                    }
                }
            }
            /* 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;
                    }
                    else
                    {
                        WREL0 = 1U;
                    }
                }
                else
                {
                    r_iica0_callback_master_receiveend();
                }
            }
        }
    }
}

/***********************************************************************
* Function Name: r_iica0_callback_master_error
* Description  : This function is a callback function when IICA0 master error
occurs.
* Arguments    : flag -
*                    status flag
* Return Value : None
***********************************************************************/
static void r_iica0_callback_master_error(MD_STATUS flag)
{
    /* Start user code. Do not edit comment generated here */
    /* End user code. Do not edit comment generated here */
}
```

```
/***********************************************************************
* Function Name: r_iica0_callback_master_receiveend
* Description  : This function is a callback function when IICA0 finishes
master reception.
* Arguments    : None
* Return Value : None
***********************************************************************/
static void r_iica0_callback_master_receiveend(void)
{
    IIC_flg_end = 1;
}

/***********************************************************************
* Function Name: r_iica0_callback_master_sendend
* Description  : This function is a callback function when IICA0 finishes
master transmission.
* Arguments    : None
* Return Value : None
***********************************************************************/
static void r_iica0_callback_master_sendend(void)
{
    IIC_flg_end = 1;
}
```

7. I2C 从机参考例程

```
/***********************************************************************
* File Name    : r_main.c
* Version      : CodeGenerator for RL78/I1A V2.00.00.04 [21 Feb 2013]
* Device(s)    : R5F107AE
* Tool-Chain   : CA78K0R
* Description  : This file implements main function.
* Creation Date: 13/9/2013
***********************************************************************/

/***********************************************************************
Pragma directive
***********************************************************************/

/***********************************************************************
Includes
***********************************************************************/#
include "r_cg_macrodriver.h"
#include "r_cg_cgc.h"
#include "r_cg_serial.h"
#include "r_cg_userdefine.h"

/***********************************************************************
Global variables and functions
***********************************************************************/
extern unsigned char rx_end;
extern unsigned char tx_end;
uint8_t  g_read_value[250]; //buffer I2C read
uint8_t  g_write_value[250];//buffer I2C write

/***********************************************************************
* Function Name: main
* Description  : This function implements main function.
* Arguments    : None
* Return Value : None
***********************************************************************/
void main(void)
{
     R_IICA0_Slave_Receive(&g_read_value[0], 1);

     g_write_value[0] = 0x55;
    R_IICA0_Slave_Send(&g_write_value[0], 1);
    while (1U)
    {
       if (rx_end == 1)    //1 byte IIC data recieved?
       {
           rx_end = 0;     //clr recieve flag
           //set recieve another IIC data
           R_IICA0_Slave_Receive(&g_read_value[0], 1);    }

           if (tx_end == 1)    //1 byte IIC data transmit?
           {
               tx_end = 0;
               g_write_value[0] = 0x55;
               //set recieve another 250 IIC data
```

```
            R_IICA0_Slave_Send(&g_write_value[0], 1);  }
    }
}
```

```
/***********************************************************************
* File Name    : r_cg_serial.c
* Version      : CodeGenerator for RL78/I1A V2.00.00.04 [21 Feb 2013]
* Device(s)    : R5F107AE
* Tool-Chain   : CA78K0R
* Description  : This file implements device driver for Serial module.
* Creation Date: 13/9/2013
***********************************************************************/

/***********************************************************************
Pragma directive
***********************************************************************/

/***********************************************************************
Includes
***********************************************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_serial.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefine.h"

/***********************************************************************
Global variables and functions
***********************************************************************/
volatile uint8_t   g_iica0_master_status_flag; /* iica0 master flag */
volatile uint8_t   g_iica0_slave_status_flag;  /* iica0 slave flag */
volatile uint8_t * gp_iica0_rx_address;        /* iica0 receive buffer address */
volatile uint16_t  g_iica0_rx_len;             /* iica0 receive data length */
volatile uint16_t  g_iica0_rx_cnt;             /* iica0 receive data count */
volatile uint8_t * gp_iica0_tx_address;        /* iica0 send buffer address */
volatile uint16_t  g_iica0_tx_cnt;             /* iica0 send data count */
/* Start user code for global. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/***********************************************************************
* Function Name: R_IICA0_Create
* Description  : This function initializes the IICA0 module.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_IICA0_Create(void)
{
    IICA0EN = 1U; /* supply IICA0 clock */
    IICE0 = 0U; /* disable IICA0 operation */
    IICAMK0 = 1U; /* disable INTIICA0 interrupt */
    IICAIF0 = 0U; /* clear INTIICA0 interrupt flag */
    /* Set INTIICA0 high priority */
    IICAPR10 = 0U;
    IICAPR00 = 0U;
    /* Set SCLA0, SDAA0 pin */
    P6 &= 0xFCU;
    P6 |= 0x03;
    PM6 |= 0x03U;
    SMC0 = 1U;
    IICWL0 = _15_IICA0_IICWL_VALUE;
    IICWH0 = _14_IICA0_IICWH_VALUE;
    DFC0 = 0U; /* digital filter off */
```

```
    IICCTL01 |= _01_IICA_fCLK_HALF;
    SVA0 = 10_IICA0_SLAVEADDRESS;
    STCEN0 = 1U;
    IICRSV0 = 1U;
    SPIE0 = 0U;
    WTIM0 = 1U;
    ACKE0 = 1U;
    IICAMK0 = 0U;
    IICE0 = 1U;
     LREL0 = 1U;
     /* Set SCLA0, SDAA0 pin */
    PM1 &= 0xFCU;
}

/***********************************************************************
* Function Name: R_IICA0_Stop
* Description  : This function stops IICA0 module operation.
* Arguments    : None
* Return Value : None
***********************************************************************/
void R_IICA0_Stop(void)
{
    IICE0 = 0U;    /* disable IICA0 operation */
}

/***********************************************************************
* Function Name: R_IICA0_Slave_Send
* Description  : This function sends data as slave mode.
* Arguments    : tx_buf -
*                    transfer buffer pointer
*                tx_num -
*                    buffer size
* Return Value : None
***********************************************************************/
void R_IICA0_Slave_Send(uint8_t * const tx_buf, uint16_t tx_num)
{
    g_iica0_tx_cnt = tx_num;
    gp_iica0_tx_address = tx_buf;
    g_iica0_slave_status_flag = 0U;
}

/***********************************************************************
* Function Name: R_IICA0_Slave_Receive
* Description  : This function receives data as slave mode.
* Arguments    : rx_buf -
*                    receive buffer pointer
*                rx_num -
*                    buffer size
* Return Value : None
***********************************************************************/
void R_IICA0_Slave_Receive(uint8_t * const rx_buf, uint16_t rx_num)
{
    g_iica0_rx_len = rx_num;
    g_iica0_rx_cnt = 0U;
    gp_iica0_rx_address = rx_buf;
    g_iica0_slave_status_flag  = 0U;
}
```

```
/***********************************************************************
* File Name    : r_cg_serial_user.c
* Version      : CodeGenerator for RL78/I1A V2.00.00.04 [21 Feb 2013]
* Device(s)    : R5F107AE
* Tool-Chain   : CA78K0R
* Description  : This file implements device driver for Serial module.
* Creation Date: 13/9/2013
***********************************************************************/

/***********************************************************************
Pragma directive
***********************************************************************/
#pragma interrupt INTIICA0 r_iica0_interrupt

/***********************************************************************
Includes
***********************************************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_serial.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefine.h"

/***********************************************************************
Global variables and functions
***********************************************************************/
extern volatile uint8_t   g_iica0_master_status_flag;  /* iica0 master flag */
extern volatile uint8_t   g_iica0_slave_status_flag;   /* iica0 slave flag */
extern volatile uint8_t * gp_iica0_rx_address;         /* iica0 receive buffer address
*/
extern volatile uint16_t  g_iica0_rx_cnt;              /* iica0 receive data length */
extern volatile uint16_t  g_iica0_rx_len;              /* iica0 receive data count */
extern volatile uint8_t * gp_iica0_tx_address;         /* iica0 send buffer address */
extern volatile uint16_t  g_iica0_tx_cnt;              /* iica0 send data count */
uint8_t rx_end = 0;                                      //rx end flag
uint8_t char tx_end = 0;                                    //tx end flag
extern uint8_t  g_read_value[250];                     //buffer for data read
extern uint8_t  g_write_value[250];                    //buffer for data write

/***********************************************************************
* Function Name: r_iica0_interrupt
* Description  : This function is INTIICA0 interrupt service routine.
* Arguments    : None
* Return Value : None
***********************************************************************/
__interrupt static void r_iica0_interrupt(void)
{
    P3 = 0x08;
    if ((IICS0 & _80_IICA_STATUS_MASTER) == 0U)
    {
        iica0_slave_handler();
    }
     P3 = 0x00;
}
```

```
/***********************************************************************
* Function Name: iica0_slave_handler
* Description  : This function is IICA0 slave handler.
* Arguments    : None
* Return Value : None
***********************************************************************/
static void iica0_slave_handler(void)
{
    /* Control for stop condition */
    if (1U == SPD0)
    {
        /* Get stop condition */
        SPIE0 = 0U;
        g_iica0_slave_status_flag = 1U;
    }
    else
    {
  if(1U == STD0)/* start or restart condition*/
   g_iica0_slave_status_flag = 0U;  //Reset slave status flag

        if ((g_iica0_slave_status_flag & _80_IICA_ADDRESS_COMPLETE) == 0U)
        {
            if (1U == COI0)
            {
                SPIE0 = 1U;
                g_iica0_slave_status_flag |= _80_IICA_ADDRESS_COMPLETE;

                if (1U == TRC0)
                {
                    WTIM0 = 1U;

                    if (g_iica0_tx_cnt > 0U)
                    {
                        IICA0 = *gp_iica0_tx_address;
                        gp_iica0_tx_address++;
                        g_iica0_tx_cnt--;
                    }
                    else
                    {
                        r_iica0_callback_slave_sendend();
                        WREL0 = 1U;
                    }
                }
                else
                {
                    ACKE0 = 1U;
                    WTIM0 = 0U;
                    WREL0 = 1U;
                }
            }
            else
            {
                r_iica0_callback_slave_error(MD_ERROR);
            }
        }
        else
        {
```

```
            if (1U == TRC0)
            {
                if ((0U == ACKD0) && (g_iica0_tx_cnt != 0U))
                {
                    r_iica0_callback_slave_error(MD_NACK);
                }
                else
                {
                    if (g_iica0_tx_cnt > 0U)
                    {
                        IICA0 = *gp_iica0_tx_address;
                        gp_iica0_tx_address++;
                        g_iica0_tx_cnt--;
                    }
                    else
                    {
                        r_iica0_callback_slave_sendend();
                        WREL0 = 1U;
                    }
                }
            }
            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;
                        r_iica0_callback_slave_receiveend();
                    }
                    else
                    {
                        WREL0 = 1U;
                    }
                }
                else
                {
                    WREL0 = 1U;
                }
            }
        }
    }
}
```

```
/***********************************************************************
Function Name: r_iica0_callback_slave_error
* Description  : This function is a callback function when IICA0 slave error occurs.
* Arguments    : None
* Return Value : None
***********************************************************************/
static void r_iica0_callback_slave_error(MD_STATUS flag)
{
    /* Start user code. Do not edit comment generated here */
    /* End user code. Do not edit comment generated here */
}

/***********************************************************************
Function Name: r_iica0_callback_slave_receiveend
* Description  : This function is a callback function when IICA0 finishes slave
reception.
* Arguments    : None
* Return Value : None
***********************************************************************/
static void r_iica0_callback_slave_receiveend(void)
{
    rx_end = 1;
}

/***********************************************************************
Function Name: r_iica0_callback_slave_sendend
* Description  : This function is a callback function when IICA0 finishes slave
transmission.
* Arguments    : None
* Return Value : None
***********************************************************************/
static void r_iica0_callback_slave_sendend(void)
{
    tx_end = 1;
}
```

公司主页和咨询窗口

瑞萨电子主页

http://www.renesas.com/

咨询

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

修订记录	RL78/G14 群 应用说明

Rev	发行日	修订内容	
		页	修订处
1.00	Jan 10, 2014	—	初版发行

产品使用时的注意事项

本文对适用于单片机所有产品的“使用时的注意事项”进行说明。有关个别的使用时的注意事项请参照正文。此外，如果在记载上有与本手册的正文有差异之处，请以正文为准。

1. 未使用的引脚的处理

【注意】将未使用的引脚按照正文的“未使用引脚的处理”进行处理。

CMOS产品的输入引脚的阻抗一般为高阻抗。如果在开路的状态下运行未使用的引脚，由于感应现象，外加LSI周围的噪声，在LSI内部产生穿透电流，有可能被误认为是输入信号而引起误动作。

未使用的引脚，请按照正文的“未使用引脚的处理”中的指示进行处理。

2. 通电时的处理

【注意】通电时产品处于不定状态。

通电时， LSI内部电路处于不确定状态，寄存器的设定和各引脚的状态不定。通过外部复位引脚对产品进行复位时，从通电到复位有效之前的期间，不能保证引脚的状态。

同样，使用内部上电复位功能对产品进行复位时，从通电到达到复位产生的一定电压的期间，不能保证引脚的状态。

3. 禁止存取保留地址（保留区）

【注意】禁止存取保留地址（保留区）

在地址区域中，有被分配将来用作功能扩展的保留地址（保留区）。因为无法保证存取这些地址时的运行，所以不能对保留地址（保留区）进行存取。

4. 关于时钟

【注意】复位时，请在时钟稳定后解除复位。

在程序运行中切换时钟时，请在要切换成的时钟稳定之后进行。复位时，在通过使用外部振荡器（或者外部振荡电路）的时钟开始运行的系统中，必须在时钟充分稳定后解除复位。另外，在程序运行中，切换成使用外部振荡器（或者外部振荡电路）的时钟时，在要切换成的时钟充分稳定后再进行切换。

5. 关于产品间的差异

【注意】在变更不同型号的产品时，请对每一个产品型号进行系统评价测试。

即使是同一个群的单片机，如果产品型号不同，由于内部ROM、版本模式等不同，在电特性范围内有时特性值、动作容限、噪声耐量、噪声辐射量等也不同。因此，在变更不认同型号的产品时，请对每一个型号的产品进行系统评价测试。

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.

以下“注意事项”为从英语原稿翻译的中文译文，仅作为参考译文，英文版的“Notice”具有正式效力。

注意事项

1. 本文档中所记载的关于电路、软件和其他相关信息仅用于说明半导体产品的操作和应用实例。用户如在设备设计中应用本文档中的电路、软件和相关信息，请自行负责。对于用户或第三方因使用上述电路、软件或信息而遭受的任何损失，瑞萨电子不承担任何责任。
2. 在准备本文档所记载的信息的过程中，瑞萨电子已尽量做到合理注意，但是，瑞萨电子并不保证这些信息都是准确无误的。用户因本文档中所记载的信息的错误或遗漏而遭受的任何损失，瑞萨电子不承担任何责任。
3. 对于因使用本文档中的瑞萨电子产品或技术信息而造成的侵权行为或因此而侵犯第三方的专利、版权或其他知识产权的行为，瑞萨电子不承担任何责任。本文档所记载的内容不应视为对瑞萨电子或其他人所有的专利、版权或其他知识产权作出任何明示、默示或其它方式的许可及授权。
4. 用户不得更改、修改、复制或者以其他方式部分或全部地非法使用瑞萨电子的任何产品。对于用户或第三方因上述更改、修改、复制或以其他方式非法使用瑞萨电子产品的行为而遭受的任何损失，瑞萨电子不承担任何责任。
5. 瑞萨电子产品根据其质量等级分为两个等级：“标准等级”和“高质量等级”。每种瑞萨电子产品的推荐用途均取决于产品的质量等级，如下所示：
   标准等级： 计算机、办公设备、通讯设备、测试和测量设备、视听设备、家用电器、机械工具、个人电子设备以及工业机器人等。
   高质量等级： 运输设备（汽车、火车、轮船等）、交通控制系统、防灾系统、预防犯罪系统以及安全设备等。
   瑞萨电子产品无意用于且未被授权用于可能对人类生命造成直接威胁的产品或系统及可能造成人身伤害的产品或系统（人工生命维持装置或系统、植埋于体内的装置等）中，或者可能造成重大财产损失的产品或系统（核反应堆控制系统、军用设备等）中。在将每种瑞萨电子产品用于某种特定应用之前，用户应先确认其质量等级。不得将瑞萨电子产品用于超出其设计用途之外的任何应用。对于用户或第三方因将瑞萨电子产品用于其设计用途之外而遭受的任何损害或损失，瑞萨电子不承担任何责任。
6 使用本文档中记载的瑞萨电子产品时，应在瑞萨电子指定的范围内，特别是在最大额定值、电源工作电压范围、移动电源电压范围、热辐射特性、安装条件以及其他产品特性的范围内使用。对于在上述指定范围之外使用瑞萨电子产品而产生的故障或损失，瑞萨电子不承担任何责任。
7. 虽然瑞萨电子一直致力于提高瑞萨电子产品的质量和可靠性，但是，半导体产品有其自身的具体特性，如一定的故障发生率以及在某些使用条件下会发生故障等。此外，瑞萨电子产品均未进行防辐射设计。所以请采取安全保护措施，以避免当瑞萨电子产品在发生故障而造成火灾时导致人身事故、伤害或损害的事故。例如进行软硬件安全设计（包括但不限于冗余设计、防火控制以及故障预防等）、适当的老化处理或其他适当的措施等。由于难于对微机软件单独进行评估，所以请用户自行对最终产品或系统进行安全评估。
8. 关于环境保护方面的详细内容，例如每种瑞萨电子产品的环境兼容性等，请与瑞萨电子的营业部门联系。使用瑞萨电子产品时，请遵守对管制物质的使用或含量进行管理的所有相应法律法规（包括但不限于《欧盟RoHS指令》）。对于因用户未遵守相应法律法规而导致的损害或损失，瑞萨电子不承担任何责任。
9. 不可将瑞萨电子产品和技术用于或者嵌入日本国内或海外相应的法律法规所禁止生产、使用及销售的任何产品或系统中。也不可将本文档中记载的瑞萨电子产品或技术用于与军事应用或者军事用途有关的任何目的（如大规模杀伤性武器的开发等）。在将本文档中记载的瑞萨电子产品或技术进行出口时，应当遵守相应的出口管制法律法规，并按照上述法律法规所规定的程序进行。
10. 向第三方分销或处分产品或者以其他方式将产品置于第三方控制之下的瑞萨电子产品买方或分销商，有责任事先向上述第三方通知本文档规定的内容和条件；对于用户或第三方因非法使用瑞萨电子产品而遭受的任何损失，瑞萨电子不承担任何责任。
11. 在事先未得到瑞萨电子书面认可的情况下，不得以任何形式部分或全部转载或复制本文档。
12. 如果对本文档所记载的信息或瑞萨电子产品有任何疑问，或者用户有任何其他疑问，请向瑞萨电子的营业部门咨询。

(注1) 瑞萨电子：在本文档中指瑞萨电子株式会社及其控股子公司。
(注2) 瑞萨电子产品：指瑞萨电子开发或生产的任何产品。

RENESAS

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.
2801 Scott Boulevard Santa Clara, CA 95050-2549, 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: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

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: +852-2265-6688, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, 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.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.0