Introduction
This document describes how to use the RLIN3 hardware in master mode.

Target Device
RL78 F13/F14 Group (R5F10PPJ)
When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Development environment
IAR Embedded workbench for Renesas RL78 V1.30.3
Contents

1. RLIN3 hardware module specifications ........................................................... 3

2. Development environment .............................................................................. 5

3. Software ........................................................................................................... 6
   3.1 Operation overview ....................................................................................... 6
   3.2 Functions ...................................................................................................... 7
   3.3 Function Specifications ................................................................................ 7

3.4 Flowcharts ..................................................................................................... 10
   3.4.1 Main Flowchart ....................................................................................... 10
   3.4.2 Initial RLIN flowchart ............................................................................. 11
   3.4.3 Master Header frame transmit flowchart .................................................... 12
   3.4.4 Master transmit flowchart ....................................................................... 12
   3.4.5 Master receive flowchart ......................................................................... 13
   3.4.6 Master interrupt flowchart .................................................................... 14

4. Demo system .................................................................................................... 15

5. Sample code .................................................................................................... 16
   5.1 RLIN_driver.c ............................................................................................ 16
   5.2 RLIN_driver_user.c .................................................................................... 20
   5.3 RLIN_driver.h ............................................................................................ 22
   5.4 RLIN_main.c ............................................................................................... 23

Website and Support <website and support,ws> .................................................. 26

Revision Record < RL78/F1x Application note RLIN3 in master mode >

General Precautions in the Handling of MPU/MCU Products
1. **RLIN3 hardware module specifications**

The RLIN3 interface is a dedicated UART interface supporting LIN slave and master functionality.

![RLIN3 Module Block Diagram](image)

**Features of the RLIN3 interface**

- **LIN Master mode support by extended hardware features**
  - Inheritance of functionality from RLIN2 IP
  - Conform to LIN Specification Package Revision 1.3, 2.0, 2.1 and SAEJ2602
  - Frame combined operation (header + response processing in 1 step)
  - Frame separate operation (separate steps for header and response)
  - Wakeup transmission and reception
  - Automatic classic or enhanced checksum generation/verification
  - Automatic error detection
  - Automatic frame communication
    - Automatic Header transmission
    - Automatic response transmission/reception

- **Advanced features for LIN**
  - Extended response reception and transmission (extension to any data count by software)
  - LIN Wake Up mode
  - LIN Self-Test mode
  - Various settings for LIN frame timing (spacing, break/delimiter timing)
  - LIN Error detections
    - Bit errors (commonly, or in break/wakeup field ["physical bit error"])
    - Frame error (wrong STOP bit level)
    - Checksum error (received does not match internally calculated)
    - Timeout error (either frame or response, threshold automatically set)
    - Response preparation error (for LIN master – on response if not yet triggered)
- Software processing flow
  - During a complete LIN message only one interrupts are generated. The interrupt are generated when complete message response.

- Due to this enhanced LIN functionality the interrupt load will be drastically reduced compared to a standard UART.

The RLIN3 only gets 1 interrupt while the UART module gets 24 interrupts.
2. Development environment

The sample code described in this application note runs under the conditions below.

Table 2.1 Development environment

<table>
<thead>
<tr>
<th>Item</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU</td>
<td>RL78/F14 R5F10PPJ (WS2.0)</td>
</tr>
<tr>
<td>Operating frequencies</td>
<td>Xin : 4MHz</td>
</tr>
<tr>
<td></td>
<td>System clock: 32MHz (PLL)</td>
</tr>
<tr>
<td></td>
<td>CPU clock: 32MHz</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>5.0V for MCU, 12V for LIN transceiver</td>
</tr>
<tr>
<td>Integrated development environment</td>
<td>IAR Embedded workbench for Renesas RL78   V1.30.3</td>
</tr>
<tr>
<td>LIN protocol versions</td>
<td>V2.1</td>
</tr>
<tr>
<td>Evaluation board</td>
<td>See figure 2.1 below</td>
</tr>
</tbody>
</table>

Figure 2.1 Evaluation board
3. Software

The sample code demonstrates the usage of the RLIN3 interface in LIN master mode. The sample code runs on the QB-R5F10PPJ-TB, which is a target board for the RL78F13,F14 microcontroller family including a LIN transceiver. In master mode, the RLIN3 waits for INTP0 falling edge, and it transmits a header frame to the LIN bus when the external interrupt generated. Otherwise, it prepares a response transmission or reception according to send IDs. LED1 and LED2 are showing that different IDs are contained in the header, and frames (headers and responses) were transmitted or received successfully.

3.1 Operation overview

Settings:

- Use channel of the RLIN3 to perform LIN communication in master mode.
- Use the P1.3/LTXD0 pin for the transmit data output.
- Use the P1.4/LRXD0 pin for the receive data input.
- Set the baud rate to 19200bps.
- Use the INTLIN0RVC interrupt; The INTLIN0RVC interrupt is generated after a LIN successful frame reception.
- Use the INTLIN0TRM interrupt; The INTLIN0TRM interrupt is generated after a LIN successful frame transmission, header transmission or wake up transmission.
- Use the INTLIN0 interrupt; The INTLIN0 interrupt is generated when the LIN bus have any Error was detected. A complete error handling is not implemented.
- Communication direction and number of transmit/receive date at a response field are determined by the ID data received at the ID field.
- ID data store in the ID buffer register LIDB0.
- Load the data to the data buffer before starting the header transmission when master will transmit a response according to ID.
3.2 Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
<th>Code size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLIN_Master_Init</td>
<td>Initial setting</td>
<td>98</td>
</tr>
<tr>
<td>RLIN_Master_HeaderTransmit</td>
<td>Header Transmit preparation</td>
<td>78</td>
</tr>
<tr>
<td>RLIN_Master_Transmit</td>
<td>Data transmission preparation</td>
<td>53</td>
</tr>
<tr>
<td>RLIN_Master_Receive</td>
<td>Data reception preparation</td>
<td>18</td>
</tr>
<tr>
<td>RLIN_Master_GetData</td>
<td>Get data from data buffer</td>
<td>18</td>
</tr>
<tr>
<td>Clear_DataBuffer</td>
<td>Setting all data buffer to 0</td>
<td>26</td>
</tr>
<tr>
<td>Get_response_RxData</td>
<td>Store data to variables array from Data buffer</td>
<td>41</td>
</tr>
</tbody>
</table>

Table 3.1 lists the Functions

3.3 Function Specifications

The following tables list the sample code function specifications

**RLIN_Master_Init**

<table>
<thead>
<tr>
<th>Outline</th>
<th>Initial setting of RLIN3’s registers in master mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void RLIN_Master_Init(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Setting of channel, clock, baud rate, interrupts, header format.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Returned value</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3.2 RLIN_Master_Init

**RLIN_Master_HeaderTransmit**

<table>
<thead>
<tr>
<th>Outline</th>
<th>Header transmit preparation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void RLIN_Master_HeaderTransmit(uint8_t ID)</td>
</tr>
<tr>
<td>Description</td>
<td>Set RLIN3 to master mode, prepare response, setting header transmit start</td>
</tr>
<tr>
<td>Arguments</td>
<td>uint8_t ID Setting to ID buffer</td>
</tr>
<tr>
<td>Returned value</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3.3 RLIN_Master_HeaderTransmit
### RLIN_Master_Transmit

<table>
<thead>
<tr>
<th>Outline</th>
<th>Data transmission preparation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void RLIN_Master_Transmit(uint8_t * databuf, uint8_t data_length)</td>
</tr>
<tr>
<td>Description</td>
<td>Setting data buffer and response transmission start, it was called by RLIN_Master_HeaderTransmit.</td>
</tr>
<tr>
<td>Arguments</td>
<td></td>
</tr>
<tr>
<td></td>
<td>uint8_t * databuf</td>
</tr>
<tr>
<td></td>
<td>uint8_t data_length</td>
</tr>
<tr>
<td>Returned value</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3.4 RLIN_Master_Transmit

### RLIN_Master_Receive

<table>
<thead>
<tr>
<th>Outline</th>
<th>Data reception preparation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void RLIN_Master_Receive(uint8_t data_length)</td>
</tr>
<tr>
<td>Description</td>
<td>Clear data buffer, setting reception format, response reception start</td>
</tr>
<tr>
<td>Arguments</td>
<td></td>
</tr>
<tr>
<td></td>
<td>uint8_t data_length</td>
</tr>
<tr>
<td>Returned value</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3.5 RLIN_Master_Receive

### RLIN_Master_GetData

<table>
<thead>
<tr>
<th>Outline</th>
<th>Get data from data buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void RLIN_Master_GetData(void)</td>
</tr>
<tr>
<td>Description</td>
<td>This function is get data from data buffer according to ID during the response reception</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Returned value</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3.6 RLIN_Master_GetData
Clear_DataBuffer

<table>
<thead>
<tr>
<th>Outline</th>
<th>Clear all data buffer to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void Clear_DataBuffer(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Clear the complete data buffer</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Returned value</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3.7 Clear_DataBuffer

Get_response_RxData

<table>
<thead>
<tr>
<th>Outline</th>
<th>Store data to variable array from ID buffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>uint8_t Get_response_RxData(uint8_t * RxData)</td>
</tr>
<tr>
<td>Description</td>
<td>Get reception data to variable array</td>
</tr>
<tr>
<td>Arguments</td>
<td>Uint8_t * RxData</td>
</tr>
<tr>
<td>Returned value</td>
<td>RxData[1]</td>
</tr>
</tbody>
</table>

Table 3.8 Get_response_RxData
3.4 Flowcharts

3.4.1 Main Flowchart

Figure 3.5 show the main processing
3.4.2 Initial RLIN flowchart

- **RLIN initial**
- **Select Channel RLIN0**
  - 0x00 → LCHSEL selects RLIN0
  - LCHSEL selects RLIN0
- **Select clock supply RLIN0**
- **Baud rate setting and bit sampling**
  - 0x01 → LWBR0, prescaler clock to 32MHz, 16 bit sampling.
  - LMD[1:0] LIN master mode
  - LPRP00=0x67, fa=19230bps
- **Clear interrupt flag and maskable**
- **RLIN0 and interrupt mode select**
  - LIOS=1; Transmission interrupt, successful reception interrupt, and reception status interrupt are used.
  - BLT=5, transmit break width is 18 Tbit; BDT=1, 1Tbit Delimiter.
  - 0x11 → LSC0, setting 1Tbit for response space, 1Tbit for inter-Byte space
- **Header frame format select**
- **Wake-up width select**
  - 0x30 → LWUP0, wake up transmission low width is 4Tbit
- **Enable RLIN0 engine clock supply**
  - LIN0MCKE=1, enable LIN engine clock supply.
- **Return**

Figure 3.1 shows the RLIN initial processing.
3.4.3 Master Header frame transmit flowchart

Figure 3.2 shows the master transmit processing.

3.4.4 Master transmit flowchart

Figure 3.3 shows the master transmit processing.
3.4.5 Master receive flowchart

Master response receive

Clear data buffer

Setting all data buffer to 0

Setting LDFC0 and reception data length

Setting checksum mode, reception mode, data length by register LDFC0

Return

Figure 3.4 show the master receive processing
3.4.6 Master interrupt flowchart

Figure 3.5 show the transmit interrupt processing

Transmission interrupt

Get interrupt flag

Clear the HTRC

HTRC=1, Header Transmission has been completed

Clear the FTC

FTC=1, response transmission has been completed

Return

Figure 3.6 show the reception interrupt processing

Reception interrupt

Clear receive interrupt flag

Check ID

ID match?

N

Y

Get data from data buffer

Return
4. Demo system

The below picture shows the demo system consists out of two RL78 F14 target boards. One board is running in master mode and the second one in slave mode. The software from the master mode is part of this application note, where the slave mode is described in a separate document. Both boards are connected via the LIN interface. The master is indicating proper data communication via the two LEDs mounted on the target boards.

![Picture of the demo system](image)

In the below state diagram of the master you will find the different internal states of the master demo with the corresponding LED.

**Master Demo:**

- **LED0: ON**
  - LED1: OFF
  - Transmit Header frame and response transmit

- **LED0: OFF**
  - LED1: OFF
  - Transmit Header frame and response reception

- **LED0: OFF**
  - LED1: ON
  - Transmit Header frame and response transmit

- **LED0: ON**
  - LED1: ON
  - Transmit Header frame and response transmit
5. Sample code

5.1 RLIN_driver.c

```c
#include "RLIN_macrodriver.h"
#include "RLIN_driver.h"
#include "RLIN_userdefine.h"

uint8_t Master_TxData1[] = {0x08, 0x80};   /*Transmission data store array*/
uint8_t Master_TxData2[] = {0x49, 0x40};   /*Transmission data store array*/
uint8_t Master_TxData3[] = {0xCA, 0x00};   /*Transmission data store array*/
uint8_t Master_RxData1[8];
uint8_t Master_RxData2[8];
uint8_t Master_RxData3[8];
```

```c
#include "RLIN_macrodriver.h"
#include "RLIN_driver.h"
#include "RLIN_userdefine.h"

uint8_t Master_TxData1[] = {0x08, 0x80};   /*Transmission data store array*/
uint8_t Master_TxData2[] = {0x49, 0x40};   /*Transmission data store array*/
uint8_t Master_TxData3[] = {0xCA, 0x00};   /*Transmission data store array*/
uint8_t Master_RxData1[8];
uint8_t Master_RxData2[8];
uint8_t Master_RxData3[8];
```

```c
void RLIN_Master_Init(void)
{
    LCHSEL = 0x00;         /*  Selects RLIN0 */
    PER2 |= 0x04;         /*  Enable input clock supply RLIN0*/
    LINCKSEL=0x00;         /*  selects the fclk=32MHz clock to RLIN0.*/
    LWBR0 = 0x01;         /*  b0=1, LIN2.0 or 2.1; Prescaler Clock Selcet 1/1;  bit sampling count select 0000 : 16 sampling. */
    LBRP00 = 0x67;         /*   fa:0x67=103D, Baud rate= 32M/ (103+1)*16= 19230 bps  fb:9615bps    fc:2403bps */
    LBRP01 = 0x5F;         /*   fd:10416bps*/
}
```
LIN0RVCIF = 0U; /* Clear Reception interrupt request signal */
LIN0TRMIF = 0U; /* Clear Transmission interrupt request signal */
LIN0WUPIF = 0U; /* Clear Wake up interrupt request signal */
LIN0IF = 0U; /* Clear LIN or LIN Status interrupt */
LIN0RVCMK = 0U; /* interrupt reception servicing enable */
LIN0TRMMK = 0U; /* interrupt transmission servicing enable */
LIN0WUPMK = 0U; /* interrupt wake up servicing enable */
LIN0MK = 0U; /* interrupt Status servicing enable */
LIE0 |= 0x0F; /* Enable successful response/wake-up reception interrupt, enable all interrupt*/
LEDE0 |= 0x8F; /* Enable error detection */

/* Header format setting*/
LMD0 = 0x10; /* b0b1=00: LIN master mode ; b3b2=00: fa=LIN sysclock; b4=1:transmission interrupt,successful reception interrupt..; b5=0: The noise filter is enable.*/
LBFC0 = 0x15; /* b3-b0=0101: transmission break width 18Tbits; b5b4=01: break delimiter 2Tbit*/
LSC0 = 0x11; /* b2-b0=001:inter-byte space 1bit or Response space 4bit; b5b4=01: inter-byte space 1Tbit;*/
LWUP0 = 0x30; /* b7-b4=0100: Wake-up Transmission low width 4 bits.*/
LIDB0 &= 0x00; /* Clear the ID buffer */
ISC = 0x00; /* INTP11 pin input signal is set as external interrupt input*/
LINCKSEL|=0x10; /* Enable RLIN0 engine clock supply,*/

}
case 0x08: RLIN_Master_Transmit(Master_TxData1,2); /* ready for response transmit*/
    break;
  case 0x49: RLIN_Master_Transmit(Master_TxData2,2); /* ready for response transmit*/
    break;
  case 0xCA: RLIN_Master_Transmit(Master_TxData3,2); /* ready for response transmit*/
    break;
  case 0x8B: RLIN_Master_Receive(2); /* ready for response receive*/
    break;
  default: break;
}

LTRC0=0x01;
}

/*****************************************************************************************************/
*/ Function Name: RLIN_Master_Transmit(void)
* Description : This function setting data buffer for response transmission start
* Arguments    : uint8_t* databuf : variable array data.
   uint8_t Data_length : transmit data length.
* Return Value : None
*****************************************************************************************************/
void RLIN_Master_Transmit(uint8_t * databuf,uint8_t Data_length)
{
  uint8_t i;
  uint16_t Databuf_adr;
  LDFC0=0x30;
  LDFC0|=Data_length; /* MSB=0011, RCDS=1:Transmission, LCS=1: Enhanced checksum mode; LSB=0100: response data lengh select 4 byte*/
  Databuf_adr=RLIN_DateBuffer;
  for(i=0;i<Data_length;i++)
    {
    *((uint8_t *)(Databuf_adr+i))=databuf[i];
    }
}

/*****************************************************************************************************/
*/ Function Name: RLIN_Master_Receive(void)
* Description : This function clear data buffer ready for response reception start
* Arguments    : uint8_t Data_length : receive data length.
* Return Value : None
*****************************************************************************************************/
void RLIN_Master_Receive(uint8_t Data_length)
{
    Clear_DataBuffer();
    LDFC0=0x20;
    LDFC0|=Data_length;   /* MSB=0011, RCDS=1:Transmission, LCS=1: Enhanced checksum mode; LSB=0100: response data length select 4 byte*/
}

/**************************************************
 * Function Name: RLIN_Master_GetData(void)
 * Description  : This function is get data from data buffer according to ID data during the response reception
 * Arguments    : uint8_t Data_length : receive data length.
 * Return Value : None
 **************************************************/
void RLIN_Master_GetData(void)
{
    switch(LIDB0)
    {
    case 0x8B: Get_response_RxData(Master_RxData1);
        P6=Master_RxData1[1];
        break;
    case 0x4c: Get_response_RxData(Master_RxData2);      /*no used*/
        break;
    case 0x0D: Get_response_RxData(Master_RxData3);      /*no used*/
        break;
    default: break;
    }
}

/**************************************************
 * Function Name: Clear_DataBuffer
 * Description  : This function setting all data buffer to some value
 * Arguments    : uint8_t x : setting data buff value
 * Return Value : None
 **************************************************/
void Clear_DataBuffer()
{
    uint8_t i;
    uint16_t Databuf_adr;
}
Databuf_adr=RLIN_DateBuffer;
for(i=0;i<8;i++)
{
  *((uint8_t *)(Databuf_adr+i))=0U;
}

/**************************************************************************
* Function Name: Get_reponse_RxData
* Description : This function get data buffer value to a variable array
* Arguments   : uint8_t * RxData : a variable array for store Data
* Return Value : None
***************************************************************************/
uint_8  Get_reponse_RxData(uint8_t * RxData)
{
  uint8_t i,k;
  uint16_t Databuf_adr;
  k=LDFC0&0x0F;
  Databuf_adr=RLIN_DateBuffer;
  for(i=0;i<k;i++)
  {
    RxData[i]=*((uint8_t *)(Databuf_adr+i));
  }
  Return RxData[1];
}

5.2  RLIN_driver_user.c

/**************************************************************************
* File Name   : RLIN_driver_user.c
* Device(s)   : R5F10PPJ
* Tool-Chain  : IAR Systems iccrl78
* Description : This file implements device driver for Interrupt module.
* Creation Date: 02.08.2013
***************************************************************************/
#include "RLIN_macrodriver.h"
#include "RLIN_driver.h"
#include "RLIN_userdefine.h"
uint8_t GetIDbuffer;

/*******************************************************************************
* Function Name: RLIN0_Transmission_interrupt
* Description : This function is RLIN0 Transmission interrupt service routine.
* Arguments : None
* Return Value : None
*******************************************************************************/
#pragma vector = INTLIN0TRM_vect
__interrupt static void RLIN0_Transmission_interrupt(void)
{
    uint8_t transmit_header_flag;
    uint8_t transmit_response_flag;
    transmit_header_flag=LST0 & 0x80;
    transmit_response_flag=LST0 & 0x01;

    if(transmit_header_flag)
    {
        LST0&=0x7F;  /*clear successful header reception flag */
    }

    if(transmit_response_flag)
    {
        LST0&=0xFE;
    }
}

/*******************************************************************************
* Function Name: RLIN0_Reception_interrupt
* Description : This function is RLIN0 Reception interrupt service routine.
* Arguments : None
* Return Value : None
*******************************************************************************/
#pragma vector = INTLIN0RVC_vect
__interrupt static void RLIN0_Reception_interrupt(void)
{
    LST0&=0xFD;  /*clear response reception successful flag*/
}

LST0&=0xFD;  /*clear response reception successful flag*/

/*lin_master_mode*/
RLIN_Master_GetData(); /*get the reception data*/

#pragma vector = INTLIN0_vect
__interrupt static void RLIN0_Status_interrupt(void)
{
    while(1U)
    {
        ;
    }
}

#pragma vector = INTLIN0WUP_vect
__interrupt static void RLIN0_Wakeup_interrupt(void)
{
    LCUC0=0x03;
    LED1=ON;
    LED2=ON;
}

5.3  RLIN_driver.h
#include "RLIN_userdefine.h"

void RLIN_Master_Init(void); /* init Master RLIN0*/
void RLIN_Master_HeaderTransmit(uint8_t ID);
void RLIN_Master_Receive(uint8_t Data_length);
void RLIN_Master_Transmit(uint8_t * databuf, uint8_t Data_length);
void RLIN_Master_GetData(void);
void Clear_DataBuffer(void);
uint_8 Get_reponse_RxData(uint8_t * RxData);

5.4 RLIN_main.c

/* File Name : RLIN_main.c
 * Device(s) : R5F10PPJ
 * Tool-Chain : IAR Systems iccrl78
 * Description : This file implements main function.
 * Creation Date: 02.08.2013
 */

#include "RLIN_macrodriver.h"
#include "RLIN_cgc.h"
#include "RLIN_port.h"
#include "RLIN_intc.h"
#include "RLIN_timer.h"
#include "RLIN_wdt.h"
#include "RLIN_driver.h"
#include "RLIN_userdefine.h"

Global variables and functions

/* Set option bytes */
#pragma location = "OPTBYTE"
__root const uint8_t opbyte0 = 0x7AU;
#pragma location = "OPTBYTE"
__root const uint8_t opbyte1 = 0xFFU;
#pragma location = "OPTBYTE"
__root const uint8_t opbyte2 = 0xE8U;
#pragma location = "OPTBYTE"
__root const uint8_t opbyte3 = 0x84U;

/*/ Set security ID */
#pragma location = "SECUID"
__root const uint8_t secuid[10] =
{0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U, 0x00U};

/*/ Secure trace RAM area */
__no_init __root unsigned char ocdtraceram[512] @ 0xFE300U;

/*/ Secure hot plug-in RAM area */
__no_init __root unsigned char hotpluginram[48] @ 0xFE500U;

void R_MAIN_UserInit(void);

void main(void)
{ 
  R_MAIN_UserInit();
  RLIN_Enable=TRUE;
  LED1=OFF;
  LED2=OFF;
  R_INTC0_Start(); /*waiting interrupt send header frame*/

  while (1U)
  {
  }
R_WDT_Restart();
}
}

/**************************************************************************/

* Function Name: R_MAIN_UserInit
* Description : This function adds user code before implementing main function.
* Arguments : None
* Return Value : None
**************************************************************************/

void R_MAIN_UserInit(void)
{
  RLIN_Master_Init();
  EI();
}
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.
<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>25.Sep.2013</td>
<td>First edition issued</td>
<td></td>
</tr>
<tr>
<td>1.01</td>
<td>29.May 2015</td>
<td>1st revision, source code changed on page 21,</td>
<td>control of LI0 register removed.</td>
</tr>
</tbody>
</table>
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 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 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 a product with a different type number, confirm that the change will not lead to problems.
   - The characteristics of an MPU or MCU 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.