Introduction
This document describes how to use the RLIN3 hardware in slave 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 and Resource Consumption .................................................................. 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 Slave transmit flowchart ............................................................................... 12
     3.4.4 Slave receive flowchart ................................................................................ 12
     3.4.5 Slave interrupt flowchart ............................................................................... 13
4. Demo system .............................................................................................................. 14
5. Sample code ............................................................................................................. 15
  5.1 RLIN_driver.c ........................................................................................................ 15
  5.2 RIN_driver_user.c ............................................................................................... 19
  5.3 RLIN_driver.h ...................................................................................................... 21
  5.4 RLIN_main.c ......................................................................................................... 22

Website and Support <website and support,ws> ........................................................... 25

Revision Record <RL78/F1x Application note RLIN3 in slave 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 Slave mode support**
  - Basic LIN functionality
    - Conform to LIN Specification Package Revision 1.3, 2.0, 2.1 and SAEJ2602
    - Automatic baud rate detection or fixed baud rate mode
    - Wakeup transmission and reception (LIN WakeUp mode)
    - Automatic classic or enhanced checksum generation/verification
    - Break field reception while frame is being transmitted/received

- **Advanced features for LIN**
  - Extended response reception and transmission (extension to any data count by software)
  - 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 two interrupts are generated. The first one after the successful PID reception and the second one after the complete message.

- Due to this enhanced LIN functionality the interrupt load will be drastically reduced compared to a standard UART.
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&lt;br&gt;System clock: 32MHz (PLL)&lt;br&gt;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</td>
</tr>
</tbody>
</table>

Figure 2.1 Evaluation board
3. **Software**

The sample code demonstrates the usage of the RLIN3 interface in LIN slave 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 slave mode, the RLIN3 waits for reception of header frame from the master. Upon detection of the header frame, the slave checks ID and response with a transmission or reception according to ID. A proper communication will be indicated by LED1 and LED2 mounted on the target boards.

### 3.1 Operation overview

**Settings:**

- Use channel of the RLIN3 to perform LIN communication in slave 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 auto baud rate for slave mode, RLIN3 can automatically measure synch field and setting baud rate by itself.
- Use the INTLIN0RVC interrupt; The INTLIN0RVC interrupt is generated after a LIN successful header reception or response reception.
- Use the INTLIN0TRM interrupt; The INTLIN0TRM interrupt is generated after a LIN successful response transmission.
- Use the INTLIN0 interrupt; The INTLIN0 interrupt is generated when an Error on the bus was detected. A complete error handling is not implemented.
- Communication direction and number of transmit/receive data at a response field are determined by the ID data received at the ID field.
- ID data store in the ID buffer register LIDB0.
- Auto store data received at the field to data buffer register LDB01-LDB08, then get data from ID Buffer and store to Slave_RxData1[], Slave_RxData2[], Slave_RxData3[] according to ID and clear the Data buffer.
- Set Slave_TxData[] to data buffer LDB01-LDB08 and setting RTS bit to start transmission.
3.2 Functions and Resource Consumption

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
<th>Code size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLIN_Slave_Init</td>
<td>Initial setting</td>
<td>91</td>
</tr>
<tr>
<td>RLIN_Slave_HeaderReceive</td>
<td>Header receive preparation</td>
<td>11</td>
</tr>
<tr>
<td>RLIN_Slave_Transmit</td>
<td>Data transmission preparation</td>
<td>57</td>
</tr>
<tr>
<td>RLIN_Slave_Receive</td>
<td>Data reception preparation</td>
<td>22</td>
</tr>
<tr>
<td>RLIN_Slave_NoResponse</td>
<td>No response to LIN bus</td>
<td>5</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_Slave_Init

<table>
<thead>
<tr>
<th>Outline</th>
<th>Initial setting of RLIN3’s registers in slave mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void RLIN_Slave_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_Slave_Init

### RLIN_Slave_HeaderReceive

<table>
<thead>
<tr>
<th>Outline</th>
<th>Header receive preparation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>None</td>
</tr>
<tr>
<td>Declaration</td>
<td>void RLIN_Slave_headerReceive(void)</td>
</tr>
<tr>
<td>Description</td>
<td>Set RLIN3 to slave mode, set header reception start</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Returned value</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3.3 RLIN_Slave_HeaderReceive
### RLIN_Slave_Transmit

<table>
<thead>
<tr>
<th>Outline</th>
<th>Data transmission preparation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
<td>void RLIN_Slave_Transmit(uint8_t * databuf, uint8_t data_length)</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Setting data buffer and response transmission start</td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td></td>
</tr>
<tr>
<td>uint8_t * databuf</td>
<td>Transmit data</td>
</tr>
<tr>
<td>uint8_t data_length</td>
<td>Transmission data length</td>
</tr>
<tr>
<td><strong>Returned value</strong></td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3.4 RLIN_Slave_Transmit

### RLIN_Slave_Receive

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

Table 3.5 RLIN_Slave_Receive

### RLIN_NoResponse

<table>
<thead>
<tr>
<th>Outline</th>
<th>No response to LIN bus</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Header</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Declaration</strong></td>
<td>void RLIN_Slave_NoResponse(void)</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Slave node does not response anything when ID invalid</td>
</tr>
<tr>
<td><strong>Arguments</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Returned value</strong></td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3.6 RLIN_NoResponse
### 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 Data variable array</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 3.1 show the main processing

1. main
2. Disable maskable interrupt
3. Initial setting RLIN
4. Enable maskable interrupt
5. Start timer channel0
6. While(1)

See figure 3.2 slave initial flowcharts

Delay for transceiver enabling.
Ready for receive header frame.
3.4.2 Initial RLIN flowchart

Figure 3.2 show the RLIN initial processing
### 3.4.3 Slave transmit flowchart

1. **Slave response transmit**
2. **Setting LDFC0**
3. **Setting transmission data to data buffer**
4. **Setting response transmission start (LTRC0)**
5. **Waiting interrupt and Return**

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

Load data to data buffer

Setting RTS=1, Response transmission started

---

**Figure 3.3** show the slave transmit processing

### 3.4.4 Slave receive flowchart

1. **Slave response receive**
2. **Clear data buffer**
3. **Setting LTRFC0 and reception data length**
4. **Setting response reception start (LTRC0)**
5. **Waiting interrupt and Return**

Setting all data buffer to 0

Setting checksum mode, reception mode, data length by register LTRFC0

Setting RTS=1 for response reception is started

---

**Figure 3.4** show the slave receive processing
3.4.5 Slave interrupt flowchart

Figure 3.5 show the reception interrupt processing.
4. Demo system

The below pictures 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 slave mode is part of this application note, where the master mode is described in a separate document. Both boards are connected via the LIN interface. The slave is indicating proper data communication via the two LEDs mounted on the target boards.

![Picture of the demo system](image)

Figure 4.1 Picture of the demo system

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

**Slave Demo:**

![State diagram of slave](image)

Figure 4.2 State diagram of slave
5. Sample code

5.1 RLIN_driver.c

/******************************************************************************
* File Name : RLIN_driver.c
* Device(s) : R5F10PPJ
* Tool-Chain : IAR Systems iccrl78
* Description : This file implements device driver for PORT module.
* Creation Date: 15.07.2013
*******************************************************************************/

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

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

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

void RLIN_Slave_Init(void)
{

    LCHSEL = 0x00;       /*  Selects RLIN0 */
    PER2  |= 0x04;       /*  Enable input clock supply RLIN0*/
    LINCKSEL=0x00;      /*  selects the fclk=32MHz clock to RLIN0.*/
    LWBR0 = 0x34;       /*  b3-b1=010: Prescaler Clock Selct 32/4, bit sampling count select b7-b4=0011 : 4 sampling. */
    LBRP00 = 0x67;      /*  lower 8bit : 0X67=103D, Baud rate= 32M/ (103+1)*16= 19230 bps*/
    LBRP01 = 0x00;      /*  upper 8 bits in the 16bit coun
    LIN0RVCIF = 0U;      /*  Clear interrupt request signal */
    LIN0TRMIF = 0U;      /*  Clear interrupt request signal */
    LIN0WUPIF = 0U;      /*  Clear interrupt request signal */
    LIN0RVCMK = 0U;      /*  interrupt servicing enable */
    LIN0TRMMK = 0U;      /*  interrupt servicing enable */
LIN0WUPMK = 0U; /* interrupt servicing enable */
LIE0 |= 0x0F; /* Enable successful response/wake-up reception interrupt, enable all interrupt*/
LEDE0 |= 0xC9; /* Enable error detection */

/*Header format setting*/
LMD0 = 0x12; /* b1b0=10; LIN Slave mode (Auto baud rate),transmission interrupt,sucessful reception interrupt,..., The noise filter is enable.*/
LBFC0 = 0x00; /* Reception break of 9.5/10 or more Tbits*/
LSC0 = 0x24; /* Response space 4bit; inter-byte space 1bit;*/
LWUP0 = 0x30; /* Wake-up Transmission low width 4 bits.*/
LIDB0 &= 0x00; /* Clear the ID buffer */
ISC = 0x04; /* LRXD0 pin input signal is set as external interrupt input.*/
LINCKSEL |= 0x10; /* Enable RLIN0 engine clock supply,*/

ISR0  |= 0x0F; /* Enable successful response/wake-up reception interrupt, enable all interrupt*/
LEDE0 |= 0xC9; /* Enable error detection */
LMD0 = 0x12; /* b1b0=10; LIN Slave mode (Auto baud rate),transmission interrupt,sucessful reception interrupt,..., The noise filter is enable.*/
LBFC0 = 0x00; /* Reception break of 9.5/10 or more Tbits*/
LSC0 = 0x24; /* Response space 4bit; inter-byte space 1bit;*/
LWUP0 = 0x30; /* Wake-up Transmission low width 4 bits.*/
LIDB0 &= 0x00; /* Clear the ID buffer */
ISC = 0x04; /* LRXD0 pin input signal is set as external interrupt input.*/
LINCKSEL |= 0x10; /* Enable RLIN0 engine clock supply,*/

所提供之代码示例和注释。

/*Function Name: RLIN_Slave_HeaderReceive(void)
 * Description : This function is setting in slave mode, enable header reception is started.
 * Arguments : None
 * Return Value : None
 */

void RLIN_Slave_HeaderReceive(void)
{
    LCUC0 = 0x03; /* 01: RLIN rest mode is canceled; 03:RLIN operation mode */
    LTRC0 |= 0x01; /* FTS=1; Header reception or wake up transmission/reception is started.*/
}

所提供之代码示例和注释。

/*Function Name: RLIN_Slave_Transmit(void)
 * Description : This function setting data buffer for response transmission start
 * Arguments : uint8_t* databuf : variable array data.
 *             : transmit data length.
 * Return Value : None
 */

void RLIN_Slave_Transmit(uint8_t* databuf,uint8_t Data_length)
{
uint8_t i;
uint16_t Databuf_adr;
LDFC0=0x30;                    /*b5=1:enhanced checksum mode; b4=1:transmission*/
LDFC0|=Data_length;            /* b4-b0=Data_length: response data length select byte*/
Databuf_adr=RLIN_DateBuffer;   /* get the data buffer address*/
for(i=0;i<Data_length;i++)             /* setting transmission data to date buffer*/
{
    *((uint8_t *)(Databuf_adr+i))=databuf[i];
}
LTRC0=0x02;                     /*setting RTS=1:Response transmission start*/

/***********************************************************************************************
* Function Name: RLIN_Slave_Receive(void)
* Description  : This function clear data buffer for response reception start
* Arguments    : uint8_t Data_length : receive data length.
* Return Value : None
***********************************************************************************************/
void RLIN_Slave_Receive(uint8_t Data_length)
{
    Clear_DataBuffer();
    LDFC0=0x20;                    /*b5=1:enhanced checksum mode; b4=0:Reception*/
    LDFC0|=Data_length;           /* b4-b0=Data_length: response data length select byte*/
    LTRC0=0x02;                   /*setting RTS=1,response reception is started*/
}
void RLIN_Slave_NoResponse(void)
{
    LTRC0=0x04;                     /* setting LNRR=0, No response request*/
}

/***************************************************************************/
* 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 avriable array for store Data */
/* Return Value : RxData[1] */
/*******************************************************************************/
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 \textbf{RLIN\_driver\_user.c}

```c
#include "RLIN\_macrodriver.h"
#include "RLIN\_driver.h"
#include "RLIN\_userdefine.h"

uint8_t GetIDbuffer;
uint8_t Slave\_RxData1[8]; /* reception data store array */
uint8_t Slave\_RxData2[8]; /* reception data store array */
uint8_t Slave\_RxData3[8]; /* reception data store array */
uint8_t Slave\_TxData[] = \{0x55,0xC0\}; /* transmission data store array */

#pragma vector = INTLIN0TRM\_vect
__interrupt static void RLIN0\_Transmission\_interrupt(void)
{
    LST0 &= 0xFE;
}

#pragma vector = INTLIN0RVC\_vect
__interrupt static void RLIN0\_Reception\_interrupt(void)
{
    uint8_t receive\_header\_flag;
    uint8_t receive\_reponse\_flag;
    receive\_header\_flag = LST0 & 0x80; /* get header reception flag */
    ...
receive_reponse_flag=LST0 & 0X02; /* get response reception flag*/

GetIDbuffer=LIDB0;
if(receive_header_flag) /* Header successful receive*/
{
    LST0 &= 0X7F; /*clear successful header reception flag*/

    switch(GetIDbuffer)
    {
        case 0x08: RLIN_Slave_Receive(2);
                     break;
        case 0x49: RLIN_Slave_Receive(2);
                     break;
        case 0xCA: RLIN_Slave_Receive(2);
                     break;
        case 0x8B: RLIN_Slave_Transmit(Slave_TxData,2);
                    P6=Slave_TxData[1];
                    break;
        default: RLIN_Slave_NoResponse();
                    break;
    }
}

if(receive_reponse_flag)
{
    LST0 &= 0xFD; /* clear response reception successful flag*/

    switch(GetIDbuffer)
    {
        case 0x08: P6 = Get_reponse_RxData(Slave_RxData1);
                     break;
        case 0x49: P6 = Get_reponse_RxData(Slave_RxData2);
                     break;
        case 0xCA: P6 = Get_reponse_RxData(Slave_RxData3);
                     break;
        default: break;
    }
}

LTRC0=0x01; /*enabled header reception interrupt*/
5.3  RLIN _driver.h

***********************************************************************************************
* File Name    : RLIN_Driver.h
* Device(s)    : R5F10PPJ
***********************************************************************************************
#include "RLIN_userdefine.h"

void Clear_DataBuffer(void);

uint_8 Get_reponse_RxData(uint8_t * RxData);

void RLIN_Slave_Init(void);  /* init Slave RLIN0*/

void RLIN_Slave_HeaderReceive(void);

void RLIN_Slave_Transmit(uint8_t* databuf,uint8_t Data_length);

void RLIN_Slave_Receive(uint8_t Data_length);

void RLIN_Slave_NoResponse(void);


5.4 RLIN_main.c

INCLUDES

#include "RLIN_macrodriver.h"
#include "RLIN_cgc.h"
#include "RLIN_port.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_TAU0_Channel0_Start();  /*Delay times for transiver wake up */
    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_Slave_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.
## Revision History of RL78/F13, F14 Group, LIN Slave Mode (RLIN3)

<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 20, control of LIE0 register removed.</td>
<td></td>
</tr>
<tr>
<td>1.02</td>
<td>24 Jan 2017</td>
<td>2nd revision setting of LWUP and ISC corrected</td>
<td></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.