To our customers,

---

**Old Company Name in Catalogs and Other Documents**

On April 1\textsuperscript{st}, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: [http://www.renesas.com](http://www.renesas.com)

---

April 1\textsuperscript{st}, 2010
Renesas Electronics Corporation

---

Issued by: Renesas Electronics Corporation ([http://www.renesas.com](http://www.renesas.com))

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.

2. 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.

3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.

4. 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.

5. When exporting the 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. You should not use Renesas Electronics products or the 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. 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.

6. 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.

7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. 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 categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. 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 an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

“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.

“High Quality” : Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

“Specific” : Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

8. 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.

9. 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 system manufactured by you.

10. 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.

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.
H8/300H Tiny Series
LIN (Local Interconnect Network) Application Note: Slave

Introduction
LIN (Local Interconnect Network) Application Note: Slave provides specification and setting examples that use the on-chip peripheral functions of the H8/300H Tiny Series microcomputer to enable communication based on the LIN communication protocol. This application note provides reference information for those users who are involved in software and hardware design.

Target Device
H8/300H Tiny Series H8/3664F/3694F/36014F

Contents
1. LIN Communication System Overview .................................................................................................. 2
2. Library Software Specifications......................................................................................................... 6
1. LIN Communication System Overview

This section describes LIN communication for a system that incorporates the sample LIN communication software library (hereinafter referred to as the library) described in this application note.

1.1 Connection to the LIN Bus

When a system is connected to a network through the LIN bus (Figure 1) and via a LIN bus interface circuit (or an LIN transceiver), LIN communication including header frame transmission as the slave node, as well as the transmission and reception of response frames, is performed.

1.1.1 System Configuration

Figure 1 shows a sample LIN bus network system configuration.

![Block Diagram of a System Connected Through the LIN Bus](image-url)
1.1.2 LIN Bus (Single-Wire Bus) Interface

Figure 2 shows a sample circuit for interfacing the LIN bus to the input/output pins of the on-chip functions of the H8/300H Tiny Series microcomputer (hereinafter referred to as the microcomputer).

![Sample LIN Bus Interface Circuit](image-url)
1.2 Overview of LIN Communication

This section describes the message frames that are transmitted and received using the LIN communication protocol.

1.2.1 Message Frame Structure

Figure 3 shows the structure of a message frame. Each message frame consists of a header frame transmitted from the master node and a response frame transmitted from the master node or a slave node.

![Message Frame Structure Diagram]

**Figure 3 Message Frame Structure**
1.2.2 Transmission and Reception of Message Frames

Figure 4 illustrates message frame transmission and reception in the master node and slave nodes.

- The master node transmits a header frame.
- Each slave node determines an ID from the received header frame and, when the ID is of the local node, the node transmits a response.
  (The master node determines the ID at transmission.)

<table>
<thead>
<tr>
<th>Master node</th>
<th>Slave node 1</th>
<th>Slave node 2</th>
<th>Slave node n</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tx</strong></td>
<td><strong>Rx</strong></td>
<td><strong>Tx</strong></td>
<td><strong>Rx</strong></td>
</tr>
<tr>
<td>Header</td>
<td>Response</td>
<td>Header</td>
<td>Response</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Header</td>
<td>Response</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Header</td>
<td>Response</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message frame - 1</td>
<td>Response transmission request ID issued to slave node 1</td>
<td>Message frame - 2</td>
<td>Response transmission request ID issued to slave node n</td>
</tr>
</tbody>
</table>

![Figure 4](image-url)
2. Library Software Specifications

By including the library in a user application program, the user application program can use on-chip functions to perform LIN communication as a slave node.

2.1 Operating Environment

Device used: H8/300H Tiny Series microcomputer (H8/3664F/3694F/36014F)

Operating frequency range (system clock ($\phi$ osc)): Range equivalent to device operating frequencies. It is necessary to define $\phi$ osc in LINID.h by considering the LIN communication speed and processing conditions of the user application program. (See Section 2.4.2, "LINID.h File Setting Example").

Functions used: Table 1 lists the on-chip peripheral functions to be used with the library, together with their uses.

Table 1 Use of On-Chip Peripheral Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Pin function (pin No.)</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCI3 (channel-0)</td>
<td>Transmission TXD (46pin)</td>
<td>Transmission of response frame</td>
<td>Asynchronous mode Data length: 8 bits No parity bit 1 stop bit (with start bit added) LSB first</td>
</tr>
<tr>
<td></td>
<td>Re却ion RXD (45pin)</td>
<td>Reception of response frame</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Communication error detection</td>
<td>Error detection function in module</td>
</tr>
<tr>
<td>Timer W</td>
<td>FTIOA (37pin)</td>
<td>Measurement of sync break field dominant period Measurement of sync field period Measurement of wait period (internal function of library) Timeout detection</td>
<td>Counting is performed at cycles of $\phi$ osc/8, and each period is measured.</td>
</tr>
<tr>
<td>IRQ</td>
<td>/IRQ0 (51pin)</td>
<td>Wake-up signal detection</td>
<td>In the standby state, the LIN bus is monitored to detect a falling-edge.</td>
</tr>
</tbody>
</table>
2.2 File Organization

- **LINslvW.c (Ver.1.40)**
  C source file used for (slave) microcomputer function setting and communication control for LIN communication in the H8/300HTiny Series (versions with built-in timer W).

- **LINID.h (Ver.1.40)**
  Include file used to include user-defined items such as the communication transfer rate and ID settings at LINslvW.c (Ver.1.40) compilation. This file also contains user application interface functions and variable definitions. These must also be included at the time of user application program compilation.

- **H8_3664.h (Ver.1.00)**
  Internal I/O register definition file for the H8/3664F/3694F

- **H8_36014.h (Ver.1.00)**
  Internal I/O register definition file for the H8/36014F

2.3 Required ROM/RAM Capacity

(When H8S or H8/300 Series C compiler CH38.exe Ver.2.0C is used)

The ROM/RAM size used varies depending on the number of IDs that are set and so on.

- **ROM size:** 2.0 Kbytes approximately
- **RAM size:** 40 bytes approximately

2.4 Functional Specifications

2.4.1 LIN Communication Specifications

- **Node:** Slave node supported
- **ID:** User-defined ID
  
  A. Response transmission ID
  
  Zero to 61 IDs (00h to 3bh, 3dh) can be set in LINID.h.
  
  (If nodes having the same ID are set on the same LIN bus, normal operation is impossible.)

  B. LIN protocol definition ID
  
  a. Master request frame ID 3ch (ID field data: 3Ch)
  
  A response frame (8-byte data) is transmitted from the master node. If the first byte of the data field is 00h, the reception of a sleep command is assumed, and a status flag (see Table 4) is set.

  b. Slave response frame ID 3dh (ID field data: 7Dh)
  
  A slave node having this ID transmits a response frame (8-byte data).

  c. Extended frame ID 3eh, 3fh (ID field data: FEh, BFh)
  
  Not supported by this library (Ver.1.40).
  
  (Upon receiving these IDs, the node waits for the next message frame (sync break field detection).)

  C. ID setting method
  
  In LINID.h, delete the definition statements (#define __IDm 0xnn (m = 00h to 3bh, 3dh)) of IDs other than those to be set as response transmission IDs, or set them as comment statements so that only the IDs to be set are defined, and then compile LINslvW.c.

- **Response data length:** The DLC (data length control) bits in the reception ID field are determined.

- **Communication transfer rate:** The communication transfer rate used is defined in LINID.h.

  From the system clock ($\phi_{osc}$) definition value and communication transfer rate definition value, the constants used in the library and the SCI3 module setting value are calculated automatically. (Note: The communication transfer rate may be restricted by $\phi_{osc}$. For details, refer to "SCI3 Module: BRR Setting Example (Asynchronous Mode) for the Bit Rate" in the hardware manual.)
• Wake-up signal transmission and reception: Wake-up signal transmission and reception functions can be included.

Including the wake-up signal transmission function

A definition statement (#define __T_WAKEUP __ON) in LINID.h includes the wake-up transmission function the user application program calls the function (LIN_transmit_wake_up). These enable the wake-up signal to be transmitted on the LIN bus.

Including the wake-up signal reception function

A definition statement (#define __R_WAKEUP __ON) in LINK.h includes the wake-up reception function. Even when the microcomputer is in the standby state, the wake-up signal on the LIN bus is detected (falling-edge detection) through IRQ0 (external interrupt input).

2.4.2 LINID.h File Setting Example

An example of setting LINID.h is shown below.

1. The microcomputer used is the H8/3664F.
2. The node transmits a wake-up signal.
3. Wake-up signal detection (falling-edge detection) through IRQ0 (external interrupt) is not performed. (No wake-up signal is transmitted from other nodes).
4. Response frames are transmitted to the following four IDs:
   
   | ID (ID bit + DLC bits) (including parity bits) |
   |-----------------|-----------------|
   | 02h             | (42h)           |
   | 13h             | (D3h)           |
   | 24h             | (64h)           |
   | 35h             | (F5h)           |

5. The system clock (φ osc) is 16 [MHz].
6. The LIN communication transfer rate is 19200 [bit/sec].
7. Correction of the LIN communication transfer rate by sync field measurement is not performed.

An example of the settings made based on the specifications described in 1. to 7., above, is given below.

(Definition statements other than the statements indicated in boldface must be deleted or set as comment lines.)

```c
/******************************************************************************************************************
/*                                                                                                              */
/*                         LINID.h    Ver.1.40                                                               */
/*                                                                                                              */
/******************************************************************************************************************
#define __ON   1  /*  This line must not be changed or deleted. */
#define __OFF   0  /*  This line must not be changed or deleted. */
#define __H8_3694F 1  /*  This line must not be changed or deleted. */
#define __H8_36014F 0  /*  This line must not be changed or deleted. */

/******************************************************************************************************************
/*      CPU selection                                                                                           */
/*-------------------------------------------------------------------------------------------------------------*/
#define __CPU __H8_3694F  /*  Define this line for the H8/3664F and 3694F series. */
//#define __CPU __H8_36014F  /*  Define this line for the H8/36014F series. */

/******************************************************************************************************************
/*      Setting of wake-up signal transmission function                                                        */
/*-------------------------------------------------------------------------------------------------------------*/
#define __T_WAKEUP __ON  /* When transmitting a wake-up signal, define this line. */
```

REJ06B0232-0101/Rev.1.01 June 2007 Page 8 of 50
/** define __T_WAKEUP __OFF */ /* When not transmitting wake-up signal, define this line. */

 /*******************************************************************************/
 /* Setting of wake-up signal detection function */
 /*******************************************************************************/
/** define __R_WAKEUP __ON */ /* When detecting a wake-up signal (falling-edge detection), define this line. */
#define __R_WAKEUP __OFF /* When not detecting wake-up signal, define this line. */

 /*******************************************************************************/
 /* Setting of response transmission IDs */
 /*******************************************************************************/
#define __Res2byte_ID __ON /* When using a 2-byte response data transmission ID, define this line. */
/** define __Res2byte_ID __OFF */ /* When not using a 2-byte response data transmission ID, define this line. */
#if __Res2byte_ID == __ON
#define __ID00 0x80 /* */
#define __ID01 0xC1 /* */
#define __ID02 0x42 /* Transmit response to ID field 42h. */
#define __ID03 0x03 /* */
#define __ID04 0xC4 /* */
#define __ID05 0x85 /* */
#define __ID06 0x06 /* */
#define __ID07 0x47 /* */
#define __ID08 0x08 /* */
#define __ID09 0x49 /* */
#define __ID0a 0xCA /* */
#define __ID0b 0x8B /* */
#define __ID0c 0x4C /* */
#define __ID0d 0x0D /* */
#define __ID0e 0x8E /* */
#define __ID0f 0xCF /* */
#define __ID10 0x50 /* */
#define __ID11 0x11 /* */
#define __ID12 0x92 /* */
#else define __ID13 0xD3 /* Transmit response to ID field D3h. */
#define __ID14 0x14 /* */
#define __ID15 0x55 /* */
#define __ID16 0xD6 /* */
#define __ID17 0x97 /* */
#define __ID18 0xD8 /* */
#define __ID19 0x99 /* */
#define __ID1a 0x1A /* */
#define __ID1b 0x5B /* */
#define __ID1c 0x9C /* */
#define __ID1d 0xDD /* */
#define __ID1e 0x5E /* */

/*#define __ID1f 0x1F       /*    */
#endif

*-----------------------------------------------------------------------------------------------*/
/*       4-byte data                                                                            */
/*----------------------------------------------------------------------------------------------*/
#define __Res4byte_ID __ON
/*  When using a 4-byte response data transmission ID, define this line.               */
/*#define __Res4byte_ID __OFF /*  When not using a 4-byte response data transmission ID, define this line.             */
#if  __Res4byte_ID == __ON
/*#define __ID20 0x20       /*    */
/*#define __ID21 0x61       /*    */
/*#define __ID22 0xE2       /*    */
/*#define __ID23 0xA3       /*    */
#define __ID24 0x64
/*  Transmit response to ID field 64h.   */
/*#define __ID25 0x25       /*    */
/*#define __ID26 0xA6       /*    */
/*#define __ID27 0xE7       /*    */
/*#define __ID28 0xA8       /*    */
/*#define __ID29 0xE9       /*    */
/*#define __ID2a 0x6A       /*    */
/*#define __ID2b 0x2B       /*    */
/*#define __ID2c 0xEC       /*    */
/*#define __ID2d 0xAD       /*    */
/*#define __ID2e 0x2E       /*    */
/*#define __ID2f 0x6F       */
#endif

/*-----------------------------------------------------------------------------------------------*/
/*       8-byte data                                                                            */
/*-----------------------------------------------------------------------------------------------*/
#define __Res8byte_ID  __ON
/*  When using an 8-byte response data transmission ID, define this line.               */
/*#define __Res8byte_ID __OFF /*  When not using an 8-byte response data transmission ID, define this line.             */
#if  __Res8byte_ID == __ON
/*#define __ID30 0xF0       /*    */
/*#define __ID31 0xB1       /*    */
/*#define __ID32 0x32       /*    */
/*#define __ID33 0x73       /*    */
/*#define __ID34 0xB4       /*    */
#define __ID35 0xF
/*  Transmit response to ID field D3h.   */
/*#define __ID36 0x76       /*    */
/*#define __ID37 0x37       /*    */
/*#define __ID38 0x78       */
#endif
/*#define __ID39 0x39       /*    */
/*#define __ID3a 0xBA       /*    */
/*#define __ID3b 0xFB       /*    */
/*#define __ID3d 0x7D       /*    */
#endif

/*********************************************************************/
/*      System clock (φosc) definition section                                                  */
/*----------------------------------------------------------------------------------------------*/
/*#define OSC_Hz  20000000     /*  φosc=20.000MHz → 20000000 */
#define OSC_Hz  16000000     /*  φosc=16.000MHz → 16000000 */
/*#define OSC_Hz  10486000     /*  φosc=10.486MHz → 10486000 */
/*#define OSC_Hz  10000000     /*  φosc=10.000MHz → 10000000 */
/*#define OSC_Hz  9830400      /*  φosc=9.8304MHz → 9830400 */
/*#define OSC_Hz  8000000      /*  φosc=8.0000MHz → 8000000 */
/*#define OSC_Hz  7372800      /*  φosc=7.3728MHz → 7372800 */
/*#define OSC_Hz  4915200      /*  φosc=4.9152MHz → 4915200 */
/*#define OSC_Hz  2457600      /*  φosc=2.4576MHz → 2457600 */

/*********************************************************************/
/*        Baud rate definition section                                                          */
/*----------------------------------------------------------------------------------------------*/
/*#define B_rate  2400      /*  2400bps → 2400 */
/*#define B_rate  4800      /*  4800bps → 4800 */
/*#define B_rate  9600      /*  9600bps → 9600 */
/*#define B_rate  10000     /*  10000bps → 10000 */
/*#define B_rate  14400     /*  14400bps → 14400 */
/*#define B_rate  15000     /*  15000bps → 15000 */
#define B_rate  19200      /*  19200bps → 19200 */
/*#define B_rate  20000      /*  20000bps → 20000 */

/*********************************************************************/
/*        Setting of baud rate correction function                                              */
/*----------------------------------------------------------------------------------------------*/
/*#define __Corrects_baud_rate __ON /* To correct the baud rate by the sync field measurement, define this line. */
#define __Corrects_baud_rate __OFF /* When not correcting the baud rate by the sync field measurement, define this line. */

/*********************************************************************/
/*        Library constant calculation section    The following must not be changed or deleted. */
/*----------------------------------------------------------------------------------------------*/
#define t_1_data  ((((OSC_Hz) / (B_rate)) + 0x04) >>3)
#define t_11_data  ((((11 * (OSC_Hz) >>2)) / (B_rate)) + 0x01) >>1)
#define t_128_data  (((((OSC_Hz) <<5) / (B_rate)) + 0x01) >>1)
#define t_15k_data  (((0xEA6 *  ((OSC_Hz) / (B_rate))) + 0x01) >>1)
#define t_2byte_data  ((((91 * (OSC_Hz) >>2)) / (B_rate)) + 0x01) >>1)
#define t_4byte_data  ((((119 * (OSC_Hz) >>2)) / (B_rate)) + 0x01) >>1)
#define t_8byte_data  (((((175 * (OSC_Hz) >>2)) / (B_rate)) + 0x01) >>1)
#define baudrate_data (((((OSC_Hz) >>4) / (B_rate)) + 0x01) >>1) - 1)

/*******************************************************************************/
/* Function and variable definition section The following must not be changed or */
/* deleted.                                                                                       */
/******************************************************************************/
#if  ((__Res2byte_ID) ||   (__Res4byte_ID)  ||  (__Res8byte_ID))
#define __RESPONSE       __ON
#else
#define __RESPONSE       __OFF
#endif

#ifndef __LIN_LIB
extern  void LIN_initialize(void);
extern  void LIN_end(void);
extern  void LIN_sleep(void);
extern  void LIN_error(void);
extern  void LIN_data_set(void);
extern  void LIN_start(void);
extern  void LIN_stop(void);
#endif

#ifdef __RESPONSE       == __ON
extern  void LIN_data_set(void);
#endif

#ifdef __T_WAKEUP       == __ON
extern  void LIN_transmit_wake_up(void);
#endif

#ifdef __R_WAKEUP       == __ON
extern  void LIN_wake_up(void);
extern  void LIN_wake_up_PR(void);
#endif

#ifdef __RESPONSE       == __ON
extern  volatile unsigned char LIN_tx_data[8];
#endif

extern  volatile unsigned char LIN_rx_id;
extern  volatile unsigned char LIN_rx_data[8];
extern  volatile union {
    unsigned char BYTE;
    struct {
        unsigned char NBA    :1;
        unsigned char CSE    :1;
        unsigned char ISFE   :1;
        unsigned char TOA3B  :1;
        unsigned char SNRE   :1;
        unsigned char SCI    :1;
        unsigned char SUC    :1;
        unsigned char SLEEP  :1;
    } BIT;
} LIN_status;
extern volatile union {
    unsigned char BYTE;
    struct {
        unsigned char CBR   :1;
        unsigned char wk6   :1;
        unsigned char WU    :1;
        unsigned char wk4   :5;
    } BIT;
} LIN_control;
#endif

2.4.3 User Application Interface

This section describes the specifications of the interface between this library and a user application program.

- Interface by function (module) call

The user application program calls functions in the library to initialize the on-chip peripheral functions that are required for LIN communication control, stop and restart LIN communication control, control wake-up signal transmission, and prepare to receive wake-up signals.

### Table 2 Functions in the Library That are Called by the User Application Program

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIN_initialize</td>
<td>Initializes the required on-chip peripheral functions for LIN communication control and starts communication control. The LIN_start function need not be called.</td>
</tr>
<tr>
<td>LIN_stop</td>
<td>Stops LIN communication control.</td>
</tr>
<tr>
<td>LIN_start</td>
<td>Restarts LIN communication control. (When turning on the power, call the LIN_initialize function.)</td>
</tr>
<tr>
<td>LIN_transmit_wake_up</td>
<td>Transmits a wake-up signal.</td>
</tr>
<tr>
<td>LIN_wake_up_PR</td>
<td>Makes preparations needed to receive a wake-up signal.</td>
</tr>
</tbody>
</table>

If functions called by the library are prepared within the user application program, processing is performed at certain event timings (upon the completion of transmission and reception, upon the detection of a communication error, and so forth) during LIN communication.

### Table 3 User Application Control Functions Called by the Library

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIN_wake_up</td>
<td>Function for user application control when a wake-up signal is detected</td>
</tr>
<tr>
<td>LIN_sleep</td>
<td>Function for user application control when a sleep command is received</td>
</tr>
<tr>
<td>LIN_data_set</td>
<td>Function for user application control before response frame transmission</td>
</tr>
<tr>
<td>LIN_end</td>
<td>Function for user application control after the completion of message frame transmission or reception</td>
</tr>
<tr>
<td>LIN_error</td>
<td>Function for user application control when a LIN communication error is detected</td>
</tr>
</tbody>
</table>
• Operation overview
  Figure 5 and Figure 6 show the operations.

Figure 5   Operation Overview at Message Frame Transmission/Reception
• Interface using global variables (data stored in the RAM area)
  The user application program and the library interface with each other by sharing data.

**Table 4  Data (Global Variables) Shared by the User Application and Library**

<table>
<thead>
<tr>
<th>Label name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIN_tx_data[0] to [7]</td>
<td>unsigned char (array)</td>
<td>Sets the transmission data when transmitting a response frame.</td>
</tr>
<tr>
<td>LIN_rx_id</td>
<td>unsigned char</td>
<td>Holds a received ID.</td>
</tr>
<tr>
<td>LIN_rx_data[0] to [7]</td>
<td>unsigned char (array)</td>
<td>Holds received response data.</td>
</tr>
<tr>
<td>LIN_status (Structure)</td>
<td>Communication status</td>
<td></td>
</tr>
<tr>
<td>LIN_status.BYTE</td>
<td>Byte access unsigned char</td>
<td></td>
</tr>
<tr>
<td>LIN_status.BIT.NBA</td>
<td>Bit 7</td>
<td>No bus active error Set condition : The LIN bus remains inactive for a certain time.</td>
</tr>
<tr>
<td>LIN_status.BIT.CSE</td>
<td>Bit 6</td>
<td>Checksum error flag Set condition : A checksum error is detected when a response is received.</td>
</tr>
<tr>
<td>Label name (variable name)</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>LIN_status.BIT.ISFE</strong></td>
<td>Bit 5</td>
<td>Sync field error&lt;br&gt;Set condition: The received sync field data (data received by the SCI3 module) is other than 55h.</td>
</tr>
<tr>
<td><strong>LIN_status.BIT.TOA3B</strong></td>
<td>Bit 4</td>
<td>Wake-up timeout&lt;br&gt;Set condition: A header frame, transmitted from the master within a certain period after the wake-up retry signal is transmitted (three times, including the first transmission), is not detected.</td>
</tr>
<tr>
<td><strong>LIN_status.BIT.SNRE</strong></td>
<td>Bit 3</td>
<td>Slave not responding error&lt;br&gt;Set condition: Reception of a response frame from a slave is not completed within a certain period during message frame transmission/reception.</td>
</tr>
<tr>
<td><strong>LIN_status.BIT.SCI</strong></td>
<td>Bit 2</td>
<td>SCI error&lt;br&gt;Set condition: An error in the SCI3 module (overrun error or framing error) is detected.</td>
</tr>
<tr>
<td><strong>LIN_status.BIT.SUC</strong></td>
<td>Bit 1</td>
<td>Message frame normal reception completion flag&lt;br&gt;Set condition: A response frame has been received normally.&lt;br&gt;Condition to clear: An ID frame has been received.</td>
</tr>
<tr>
<td><strong>LIN_status.BIT.SLEEP</strong></td>
<td>Bit 0</td>
<td>Sleep command reception flag&lt;br&gt;Set condition: A sleep command has been received.</td>
</tr>
<tr>
<td><strong>LIN_control</strong></td>
<td>(Structure)</td>
<td>Communication control</td>
</tr>
<tr>
<td><strong>LIN_control.BYTE</strong></td>
<td>Byte access&lt;br&gt;unsigned char</td>
<td>Bit access</td>
</tr>
<tr>
<td><strong>LIN_control.BIT.CBR</strong></td>
<td>Bit 7</td>
<td>Control of the communication transfer rate correction function&lt;br&gt;(See Section 2.5.1.2, &quot;Reception of a Sync Field&quot;).</td>
</tr>
<tr>
<td><strong>LIN_control.BIT.wk6</strong></td>
<td>Bit 6</td>
<td>Reserved bit</td>
</tr>
<tr>
<td><strong>LIN_control.BIT.WU</strong></td>
<td>Bit 5</td>
<td>(Wake-up control bit)&lt;br&gt;(See Section 2.5.3, &quot;Transmission and Reception of a Wake-up Signal&quot;).</td>
</tr>
<tr>
<td><strong>LIN_control.BIT.wk4</strong></td>
<td>Bits 4 to 0</td>
<td>Reserved bits</td>
</tr>
</tbody>
</table>
2.5 Operation

This section explains the transmission and reception operations performed with the library.

2.5.1 Reception of a Header Frame

1. Detection of a Sync Break Field

   The timer W input capture function measures the sync break field dominant period.

   - Pin function: FTIOA input (receiving side)
   - Function used: Timer W input capture A (falling-edge detection interrupt)
   - Internal software processing of the library:
     - Set timer W input capture A to enable rising edge detection.
     - Save the captured value (timer W counter value).
     - Clear the timeout counter (timer W overflow counter).

   - Pin function: FTIOA input (receiving side)
   - Function used: Timer W input capture A (rising edge detection interrupt)
   - Internal software processing by the library:
     - Set timer W input capture A to enable falling-edge detection.
     - Calculate $T_{SYNBRK}$ from the captured value (timer W counter value) for falling/rising edges.
     - If $T_{SYNBRK} \geq 11$ bit
       - If LIN_control.BIT.CBR = 0
         - Set RxD as the pin function.
         - SCI3 reception, error interrupt enabled
         - (Timer W input capture A interrupt disabled)
       - If LIN_control.BIT.CBR = 1
         - Set the edge counter for communication transfer rate measurement.

   ![Detection of a Sync Break Field](image_url)

   Figure 7 Detection of a Sync Break Field
2. Reception of a Sync Field

The sync field reception control method is determined according to the setting of the CBR bit in LIN_control, as follows:

CBR = 0: The SCI3 reception function determines the sync field reception data (55h).
(Figure 8  Reception and Determination of a Sync Field by the SCI3 Reception Function)

CBR = 1: The timer W input capture function measures the bit width of a sync field and corrects the communication transfer rate (by setting BRR in the SCI3 module, and so on).
(Figure 9  Correction of the Communication Transfer Rate by the Timer W Input Capture Function)
Figure 9  Correction of the Communication Transfer Rate by the Timer W Input Capture Function

Pin function: FTIOA input (receiving side)
Function used: Timer W input capture A (falling-edge detection interrupt)

Internal software processing by the library
• Save the captured value (timer W counter value).
• Start counting the number of bits.

Pin function: FTIOA input (receiving side)
Function used: Timer W input capture A (falling-edge detection interrupt)

Internal software processing by the library
• Count the number of bits.

Pin function: FTIOA input (receiving side)
Function used: Timer W input capture A (falling-edge detection interrupt)

Internal software processing by the library
• Calculate the communication transfer rate from the captured value (timer W counter value).
• Set the SCI3 transfer rate (SMR, BRR).
• Set RxD as the pin function (SCI3 reception, error interrupt enabled) (Timer W input capture A interrupt disabled)
• Set the sync field reception flag.
• Data operation for determining the sync break period.
• Data operation for determining the timeout period.
3. Reception of an ID Field

The SCI3 reception function determines the ID (including the DLC and parity bits) in the ID field reception data. If the ID is a response transmission request ID intended for the local node, transmission of a response frame starts.

---

**Figure 10  ID Field Reception and Determination**
2.5.2 Transmission and Reception of a Response Frame

1. Transmission and Reception of a Data Field (Transmission of a Checksum Field)
   The SCI3 reception function saves received data and performs a reception checksum data operation. When a response is transmitted, the subsequent data is transmitted, and a transmission checksum data operation is performed. (Within a reception interrupt)

---

**Pin function**: RxD input (receiving side)

**Function used**: SCI3 reception (interrupt)

**Internal software processing by the library**
- Determine the number of received data bytes.
- Save received data (LIN_rx_data[n]).
- Reception checksum data operation
- Count received data.

**At response transmission**
- Transmit the next data (including checksum data).
- Transmission checksum data operation

---

**Figure 11 Transmission/Reception of a Data Field and Transmission of Checksum Data**
2. Reception of a Checksum Field

The SCI3 reception function makes a determination from the received checksum field and reception checksum data obtained by an operation from the data field.

---

**Figure 12   Checksum Field Reception and Determination**
2.5.3 Transmission and Reception of a Wake-up Signal

The SCI3 transmission function transmits a wake-up signal (transmission data: 80h).

The IRQ0 falling-edge detection function detects a wake-up signal from another node.

1. Transmission of a Wake-up Signal

A definition statement in LINID.h (#define __T_WAKEUP __ON) includes the wake-up signal transmission function when compilation is performed, allowing the SCI3 transmission function to transmit a wake-up signal when the user application program calls the LIN_transmit_wake_up function. This library does not perform wake-up delimiter output control.

**Figure 13  Transmission of a Wake-up Signal**
2. Reception of a Wake-up Signal

A definition statement in LINID.h (#define __R_WAKEUP __ON) includes the wake-up signal reception function when compilation is performed, allowing the IRQ falling-edge detection function to wait for a wake-up signal from another node when the user application program calls the LIN_wake_up_PR function. This library detects only falling-edges, without verifying the wake-up field data.

![Diagram of wake-up signal reception](image)

**Figure 14  Reception of a Wake-up Signal**
2.5.4 Reception of a Sleep Command

After the normal reception of a message frame, if the received ID field data is 3Ch and the first byte of the response data is 00h, the reception of a sleep command is recognized. Then, the sleep flag (SLEEP) is set, and the LIN_sleep function (user application program) is called. (In this case, the LIN_end function is not called in the message frame.)

2.6 Software Description

This section explains the library software.

2.6.1 Including Header Files

Includes the standard library (machine.h), the LIN library definition file (LINID.h), and the on-chip peripheral register definition files (H8_3664f.h and H8_36014f.h).

```c
#include    <machine.h>
#define     __LIN_LIB
#include    "LINID.h"

#if      __CPU    ==    __H8_3694F
#include    "H8_3664f.h"
#elif    __CPU    ==    __H8_36014F
#include    "H8_36014f.h"
#endif
```

2.6.2 Defining Functions

Functions (modules) in the library must be defined.

The inclusion of the LIN_data_set function is selected by defining __Res2byte_ID, __Res4byte_ID, or __Res8byte_ID in LINID.h.

The inclusion of the LIN_transmit_wake_up function is selected by the __T_WAKEUP definition.

The inclusion of the LIN_intc_init function, LIN_wake_up function, and LIN_wake_up_PR function is selected by the __R_WAKEUP definition.

```c
void    LIN_initialize(void);
void    LIN_system_init(void);
void    LIN_port_init(void);
void    LIN_sci_init(void);
void    LIN_timerW_init(void);
void    LIN_Sflag_init(void);
void    LIN_end(void);
void    LIN_sleep(void);
void    LIN_error(void);
void    LIN_break_reception_PR(void);
void    LIN_start(void);
void    LIN_stop(void);
```
#if __RESPONSE == __ON
void LIN_data_set(void);
#endif

#if __T_WAKEUP == __ON
void LIN_transmit_wake_up(void);
#endif

#if __R_WAKEUP == __ON
void LIN_intc_init(void);
void LIN_wake_up(void);
void LIN_wake_up_PR(void);
#endif

## 2.6.3 Defining Library Internal Constants and Variables

This section defines the constants and variables that are used in the library.

### Table 5 Definition of Library Internal Constants and Variables

<table>
<thead>
<tr>
<th>Label name (variable name)</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_1</td>
<td>unsigned short</td>
<td>1-bit counter value (for waiting at SCI3 initialization)</td>
</tr>
<tr>
<td>t_11</td>
<td>unsigned long</td>
<td>11-bit counter value (for sync break field detection)</td>
</tr>
<tr>
<td>t_128</td>
<td>(Structure)</td>
<td>128-bit counter value (for sync break field detection timeout at wake-up transmission)</td>
</tr>
<tr>
<td>t_128.LONG</td>
<td>unsigned long</td>
<td></td>
</tr>
<tr>
<td>t_128.WORD.h</td>
<td>unsigned short</td>
<td></td>
</tr>
<tr>
<td>t_128.WORD.l</td>
<td>unsigned short</td>
<td></td>
</tr>
<tr>
<td>t_15k</td>
<td>unsigned long</td>
<td>15000-bit counter value (for timeout after wake-up retry transmission (3 times)) (LIN_status.BIT.TOA3B)</td>
</tr>
<tr>
<td>t_25k</td>
<td>unsigned long</td>
<td>25000-bit counter value (for no bus active detection) (LIN_status.BIT.NBA)</td>
</tr>
<tr>
<td>flame_max_2</td>
<td>(Structure)</td>
<td>Maximum response timeout value (LIN_status.BIT.SNRE)</td>
</tr>
<tr>
<td>flame_max_2.LONG</td>
<td>unsigned long</td>
<td></td>
</tr>
<tr>
<td>flame_max_2.WORD.h</td>
<td>unsigned short</td>
<td></td>
</tr>
<tr>
<td>flame_max_2.WORD.l</td>
<td>unsigned short</td>
<td></td>
</tr>
<tr>
<td>flame_max_4</td>
<td>(Structure)</td>
<td></td>
</tr>
<tr>
<td>flame_max_4.LONG</td>
<td>unsigned long</td>
<td></td>
</tr>
<tr>
<td>flame_max_4.WORD.h</td>
<td>unsigned short</td>
<td></td>
</tr>
<tr>
<td>flame_max_4.WORD.l</td>
<td>unsigned short</td>
<td></td>
</tr>
<tr>
<td>flame_max_8</td>
<td>(Structure)</td>
<td></td>
</tr>
<tr>
<td>flame_max_8.LONG</td>
<td>unsigned long</td>
<td></td>
</tr>
<tr>
<td>flame_max_8.WORD.h</td>
<td>unsigned short</td>
<td></td>
</tr>
<tr>
<td>flame_max_8.WORD.l</td>
<td>unsigned short</td>
<td></td>
</tr>
<tr>
<td>baudrate</td>
<td>(Structure)</td>
<td>Baud rate setting for SCI3 module</td>
</tr>
<tr>
<td>baudrate.WORD</td>
<td>unsigned short</td>
<td></td>
</tr>
<tr>
<td>baudrate.BYTE.smr</td>
<td>unsigned char</td>
<td></td>
</tr>
<tr>
<td>baudrate.BYTE.brr</td>
<td>unsigned char</td>
<td></td>
</tr>
</tbody>
</table>
## Label name (variable name) 
<table>
<thead>
<tr>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ex_counter.LONG</td>
<td>Time W extended counter</td>
</tr>
<tr>
<td>ex_counter.WORD.h</td>
<td>unsigned short</td>
</tr>
<tr>
<td>ex_counter.WORD.l</td>
<td>unsigned short</td>
</tr>
<tr>
<td>flame_max</td>
<td>unsigned short Response timeout setting (timer W overflow count value)</td>
</tr>
<tr>
<td>counter</td>
<td>unsigned char Transmission/reception data counter</td>
</tr>
<tr>
<td>t_checksum.WORD</td>
<td>Transmission data checksum operation value</td>
</tr>
<tr>
<td>t_checksum.BYTE.carry</td>
<td>unsigned char</td>
</tr>
<tr>
<td>t_checksum.BYTE.data</td>
<td>unsigned char</td>
</tr>
<tr>
<td>r_checksum.WORD</td>
<td>Reception data checksum operation value</td>
</tr>
<tr>
<td>r_checksum.BYTE.carry</td>
<td>unsigned char</td>
</tr>
<tr>
<td>r_checksum.BYTE.data</td>
<td>unsigned char</td>
</tr>
<tr>
<td>in_status.BYTE</td>
<td>Internal status of library</td>
</tr>
<tr>
<td>in_status.BIT.wk7</td>
<td>Bit 7 Reserved bit</td>
</tr>
<tr>
<td>in_status.BIT.sync_field</td>
<td>Bit 6 Sync field reception flag</td>
</tr>
<tr>
<td>in_status.BIT.wk5</td>
<td>Bit 5 Reserved bit</td>
</tr>
<tr>
<td>in_status.BIT.r_id</td>
<td>Bit 4 Response ID determination flag</td>
</tr>
<tr>
<td>in_status.BIT.sci</td>
<td>Bit 3 SCI3 module initialization flag</td>
</tr>
<tr>
<td>in_status.BIT.brr</td>
<td>Bit 2 Baud rate correction flag</td>
</tr>
<tr>
<td>in_status.BIT.wu</td>
<td>Bits 1 to 0 Wake-up signal transmission flag (transmission counter for internal setting)</td>
</tr>
</tbody>
</table>

```c
#if    __Corrects_baud_rate    ==    __ON
static   unsigned short    t_1;
static   unsigned long     t_11;
static   union{
    unsigned short    WORD;
    struct{
        unsigned char    smr;
        unsigned char    brr;
    }   BYTE;
}    baudrate;
#elif   __Corrects_baud_rate    ==    __OFF
const    unsigned short    t_1    =    t_1_data;
const    unsigned long     t_11   =    t_11_data;
const    union{
    unsigned short    WORD;
    struct{
        unsigned char    smr;
        unsigned char    brr;
    }   BYTE;
}    baudrate    =    baudrate_data;
```
const unsigned long t_25k = t_25k_data;
const union {
    unsigned long LONG;
    struct {
        unsigned short h;
        unsigned short l;
    } WORD;
} flame_max_2 = t_2byte_data;
const union {
    unsigned long LONG;
    struct {
        unsigned short h;
        unsigned short l;
    } WORD;
} flame_max_4 = t_4byte_data;
const union {
    unsigned long LONG;
    struct {
        unsigned short h;
        unsigned short l;
    } WORD;
} flame_max_8 = t_8byte_data;
static union {
    unsigned long LONG;
    struct {
        unsigned short h;
        unsigned short l;
    } WORD;
} ex_counter;
static unsigned short flame_max;
static unsigned char counter;

#if __T_WAKEUP == __ON
const union {
    unsigned long LONG;
    struct {
        unsigned short h;
        unsigned short l;
    } WORD;
} t_128 = t_128_data;
const unsigned long t_15k = t_15k_data;
#endif

#if __RESPONSE == __ON
static union {
    unsigned short WORD;
    struct {
        unsigned char carry;
        unsigned char data;
    } BYTE;
} t_checksum;
}
2.6.4 Defining Global Variables

The variables that are shared between the user application program and library must be defined.

(See Table 4.)

```c
#if  __RESPONSE      ==    __ON
volatile  unsigned char    LIN_tx_data[8];
#endif

volatile  unsigned char    LIN_rx_id;
volatile  unsigned char    LIN_rx_data[8];
volatile  union {
    unsigned char    BYTE;
    struct {
        unsigned char    NBA      :1;
        unsigned char    CSE      :1;
        unsigned char    ISFE     :1;
        unsigned char    TOA3B    :1;
        unsigned char    SNRE     :1;
        unsigned char    SCI      :1;
        unsigned char    SUC      :1;
        unsigned char    SLEEP    :1;
    }    BIT;
}    LIN_status;
volatile  union {
    unsigned char    BYTE;
    struct {
        unsigned char    CBR    :1;
        unsigned char    wk6    :1;
        unsigned char    WU     :1;
    }    BIT;
}    LIN_status;
```
2.6.5 Initialization Function

This function initializes the H8/3664/3694/36014 on-chip peripheral functions used for LIN communication control and the software flags, as well as other settings used in the library.

Note: Pins P14 (IRQ0), P21 (RxD), P22 (TxD), and P81 (FTIOA) are used for LIN communication. When a user application uses other pins (P10 to P12, P15 to P17, P20, P23, P24, P80, and P82 to P87) with ports 1, 2, and 8, the pin settings may be changed by the setting statements of PCR2 and PCR8 in the LIN_port_init function and PCR1 in the LIN_intc_init function in the source file shown below. When using the above-mentioned pins, set each PCR within the user application program, then delete the setting statements of PCR1, PCR2, and PCR8 in the source file below or set them as comments.

![Initialization Function Flowchart](image.png)

**Figure 15 Initialization Function Flowchart**
void LIN_initialize(void)
{
    LIN_status.BYTE       =    0;
    LIN_system_init();
    LIN_port_init();

    #if    __Corrects_baud_rate == __ON
    t_1      =    t_1_data;
    t_11     =    t_11_data;
    baudrate.WORD    =     baudrate_data;
    #endif

    LIN_timerW_init();
    LIN_Sflag_init();
    LIN_sci_init();

    #if    __R_WAKEUP == __ON
    LIN_intc_init();
    #endif

    ex_counter.WORD.h     =     0;
    LIN_control.BYTE    =    0;
}

void LIN_system_init(void)
{
    MSTCR1.BYTE    &=    0x5B;
}

void LIN_port_init(void)
{
    #if      __R_WAKEUP == __ON
    IO.PMR1.BYTE    |=    0x12;
    #elif    __R_WAKEUP == __OFF
    IO.PMR1.BYTE    |=    0x02;
    #endif
    IO.PDR2.BIT.B2    =    1;
    /*  IO.PCR2    =    0;  /* Note: When using ports 2 and 8 in a user application, set */
    /*  IO.PCR8    =    0;  /* the bits for setting the pins used in LIN to 0(input */
    /*        pins) in the user application and then delete these */
    /*        setting statements to ensure system protection */
}

void LIN_sci_init(void)
{
    SCI3.SCR3.BYTE    =    0;
    SCI3.SMR.BYTE    =    baudrate.BYTE.smr;
    SCI3.BRR    =    baudrate.BYTE.brr;

    #if    ((__RESPONSE  ==  __ON)  ||  (__T_WAKEUP  ==  __ON))
    TW.GRC    =    TW.TCNT    +    t_1;
    TW.TSRW.BIT.IMFC    =    0;
    TW.TIERW.BIT.IMIEC    =    1;
    in_status.BIT.sci    =    1;
    #endif
}
void LIN_timerW_init(void) {
    TW.TMRW.BYTE   =    0x48;
    TW.TCRW.BYTE   =    0x30;
    TW.TIOR0.BYTE  =    0x8D;
    TW.TIOR1.BYTE  =    0xF8;
    TW.TSRW.BYTE   &=    0x70;
    TW.TIERW.BYTE  =    0x81;
    TW.TMRW.BIT.CTS =    1;
}

#if    __R_WAKEUP == __ON
void LIN_intc_init(void) {
    /*  IO.PCR1 = 0;
        Note: When using port 1 in a user application, set the bits */
    /*    for setting the pins used in LIN to 0 (input pins) in */
    /*    the user application and then delete this setting */
    /*    statement to ensure system protection. */
    IEGR1.BIT.IEG0 =    0;
    IRR1.BIT.IRR10 =    0;
    IENR1.BIT.IEN0  =    0;
}
#endif

#if    __R_WAKEUP == __ON
void LIN_wake_up_PR(void) {
    IRR1.BIT.IRRI0  =    0;
    IENR1.BIT.IEN0  =    1;
}
#endif

void LIN_Sflag_init(void) {
    counter    =    0;
    in_status.BYTE    =    0;
}
2.6.6 LIN Communication Control Stop Function
This function stops LIN communication control so that it no longer communicates over the LIN bus.

```c
void LIN_stop(void){
    SCI3.SSR.BYTE &= 0x84;
    SCI3.SCR3.BYTE = 0x00;
    TW.TIERW.BYTE &= 0x70;

#if __R_WAKEUP == __ON
    IENR1.BIT.IEN0 = 0;
#endif
}
```

2.6.7 Function for LIN Communication Restart Preparation
This function restarts LIN communication control (that has previously been placed in the stopped state by the LIN communication control stop function described in Section 2.6.6 or some other reason), and then LIN communication control waits for the reception of a sync break field.

```c
void LIN_start(void){
    SCI3 initialization
    LIN_sci_init

    Clear overflow counter
    (Initialize timeout counter)

    Set sync break field detection
}
```
void LIN_start(void){
    LIN_sci_init();
    ex_counter.WORD.h = 0;
    TW.TSRW.BYTE &= 0x70;
    TW.TIERW.BYTE |= 0x81;
}

### 2.6.8 Wake-up Signal Transmission Function

This function transmits a wake-up signal. If a sync break field is not detected within the 128-bit period after the transmission of the wake-up signal, the function retries transmission up to three times, including the first transmission (transmission is controlled within the timer W interrupt function). If a sync break field is not detected within the 15000-bit period after the signal has been transmitted three times, the function sets the timeout flag (LIN_status.BIT.TOA3B) and calls the LIN_error function (user application program).

![Flowchart of the Wake-up Signal Transmission Function](image)

```c
#if    __T_WAKEUP        ==    __ON
void LIN_transmit_wake_up(void){
    LIN_control.BIT.WU    =    0;
    in_status.BIT.wu    =    1;
    #if    __R_WAKEUP        ==    __ON
    IENR1.BIT.IEN0      =    0;
    #endif
    LIN_start();
} #endif
```
### 2.6.9 Function for Preparing for Wake-up Signal Reception

This function prepares for receiving a wake-up signal from another node.

![Flowchart of the Wake-up Signal Reception Preparation Function](image)

```c
#if    __R_WAKEUP      ==    __ON
void LIN_wake_up_PR(void){
    IRR1.BIT.IRR10    =    0;
    IENR1.BIT.IEN0    =    1;
}
#endif
```

### 2.6.10 Function for Preparing to Detect a Sync Break Field in the Library

When message frame transmission or reception is completed, when an extended frame ID is received, or when an error is detected, this function makes the preparations needed to receive the next message frame (preparation for sync break field detection) in the library.

![Function for Preparing for Sync Break Field Detection and Reception](image)

```c
void LIN_break_reception_PR(void){
    SCI3.SSR.BYTE    &=    0x84;
    #if   (( __RESPONSE  ==  __ON)  ||  (__T_WAKEUP  ==  __ON))
        SCI3.SCR3.BYTE    =    0x20;
    #else
        SCI3.SCR3.BYTE    =    0x00;
    #endif
    ex_counter.WORD.h    =    0;
}
```
2.6.11 IRQ Interrupt Function

This function processes the IRQ0 falling-edge detection interrupt. After the settings have been made by the wake-up signal reception preparation function, as described in Section 2.6.9, this function detects a falling-edge on the LIN bus and makes the preparations required for LIN communication control.

```c
#if __R_WAKEUP == __ON
#pragma interrupt(IRQ0_int)
void IRQ0_int(void)
{
    LIN_status.BIT.SLEEP = 0;
    IRR1.BIT.IRRI0 = 0;
    IENR1.BIT.IENU = 0;
#if __Corrects_baud_rate == __ON
    t_1 = t_1_data;
    t_11 = t_11_data;
    baudrate.WORD = baudrate_data;
#else
    LIN_start();
    LIN_wake_up();
#endif
}
#endif
```
2.6.12 Timer W Interrupt Function

This function processes the timer W input capture A interrupt (I/C-A), compare match B interrupt (O/C-B), compare match C interrupt (O/C-C), and overflow interrupt (OVF).

![Flowchart of the Timer W Interrupt Function](image)

```c
#pragma    interrupt(TmW_int)
void    TmW_int(void){
    unsigned long    i;
    unsigned short    N,w;

    if((TW.TSRW.BIT.IMFA)  &&  (TW.TIERW.BIT.IMIEA)){
        TW.TSRW.BIT.IMFA       =    0;
#if    __Corrects_baud_rate    ==    __ON
```

Figure 22 Flowchart of the Timer W Interrupt Function
if(in_status.BIT.brr == 0){
    #endif

    if(TW.TIOR0.BIT.IOA0){
        TW.TIOR0.BIT.IOA0 = 0;
        ex_counter.LONG = (unsigned long)TW.GRA;
        TW.GRB = TW.GRA;
        if((TW.TSRW.BIT.OVF) && (ex_counter.WORD.l < 0x00FF)){
            TW.TSRW.BIT.OVF = 0;
        }
    }else{
        w = ex_counter.WORD.l;
        ex_counter.WORD.l = TW.GRA;
        if((TW.TSRW.BIT.OVF) && (ex_counter.WORD.l < 0x00FF)){
            ex_counter.WORD.h += 1;
            TW.TSRW.BIT.OVF = 0;
        }
        ex_counter.LONG -= w;
        if(ex_counter.LONG >= t_11){
            #if __Corrects_baud_rate == __ON
                if(LIN_control.BIT.CBR){
                    in_status.BIT.brr = 1;
                    LIN_control.BIT.CBR = 0;
                    counter = 4;
                }
            #endif
        #if __T_WAKEUP == __ON
            in_status.BIT.wu = 0;
            TW.TIERW.BIT.IMIEC = 0;
        #endif
            TW.TIOR0.BIT.IOA0 = 1;
        }
    }
}

#if __Corrects_baud_rate == __ON
    if(LIN_control.BIT.CBR){
        in_status.BIT.brr = 1;
        LIN_control.BIT.CBR = 0;
        counter = 4;
    }else{
        SCI3.SSR.BYTE &= 0x84;
    }
#endif

#if __RESPONSE == __ON
    SCI3.SCR3.BYTE = 0x70;
#else __RESPONSE == __OFF
    SCI3.SCR3.BYTE = 0x50;
#endif

TW.TIERW.BIT.IMIEA = 0;

#if __Corrects_baud_rate == __ON
    }
#endif

#if __T_WAKEUP == __ON
    in_status.BIT.wu = 0;
    TW.TIERW.BIT.IMIEC = 0;
#endif

    TW.TIOR0.BIT.IOA0 = 1;
}

#if __Corrects_baud_rate == __ON
}else{
    if(counter){

if(counter == 4){
    ex_counter.LONG = (unsigned long)TW.GRA;
SCI3.SCR3.BYTE = 0;
SCI3.SMR.BYTE = 0;
if((TW.TSRW.BIT.OVF) && (ex_counter.WORD.l < 0x00FF)){
    TW.TSRW.BIT.OVF = 0;
}
}
counter -= 1;
else{
    TW.TIERW.BYTE = 0xF4;
    w = ex_counter.WORD.l;
ex_counter.WORD.l = TW.GRA;
if((TW.TSRW.BIT.OVF) && (ex_counter.WORD.l < 0x00FF)){
ex_counter.WORD.h += 1;
    TW.TSRW.BIT.OVF = 0;
}
t_1 = (ex_counter.LONG - w) >> 3;
for(N = ((t_1 + 2) >> 2); N > 0x0100; N >>= 2){
    SCI3.SMR.BIT.CKS += 1;
}
SCI3.BRR = N - 1;
TW.GRC = (TW.TCNT + t_1);
TW.TSRW.BYTE ^= 0xF0;
ex_counter.WORD.h = 0;
in_status.BIT.sync_field = 1;
t_11 = t_1 * 11;
}
#endif
}
else if((TW.TSRW.BIT.IMFC) && (TW.TIERW.BIT.IMIEC)){
    TW.TSRW.BIT.IMFC = 0;
    if(in_status.BIT.sci){
        SCI3.SSR.BYTE ^= 0x84;
SCI3.SCR3.BIT.TE = 1;
#endif
    TW.TIERW.BIT.IMIEC = 0;
    #if __T_WAKEUP == __ON
    if(in_status.BIT.wu == 1){
        TW.GRC = TW.TCNT + t_128.WORD.l;
SCI3.TDR = 0x80;
in_status.BIT.wu += 1;
    #endif
    }else{
#endif
    in_status.BIT.sci = 0;
    #if __Corrects_baud_rate == __ON
    }else if(in_status.BIT.brr){
}
SCI3.SSR.BYTE &= 0x84;

#if ((__RESPONSE == __ON) || (__T_WAKEUP == __ON))
    SCI3.SCR3.BYTE = 0x70;
#else
    SCI3.SCR3.BYTE = 0x50;
#endif

TW.TIERW.BIT.IMIEC = 0;
#endif

#if __T_WAKEUP == __ON
#else if((in_status.BIT.wu == 2) && (ex_counter.WORD.h >= t_128.WORD.h)) { 
    SCI3.TDR = 0x80;
    ex_counter.WORD.h = 0;
    TW.GRC = TW.TCNT + t_128.WORD.l;
    in_status.BIT.wu += 1;
}else if((in_status.BIT.wu == 3) && (ex_counter.WORD.h >= t_128.WORD.h)) {
    SCI3.TDR = 0x80;
    ex_counter.WORD.h = 0;
    TW.TIERW.BIT.IMIEC = 0;
#endif
}
#else if((TW.TSRW.BIT.IMFB) && (TW.TIERW.BIT.IMIEB)) {
    TW.TSRW.BIT.IMFB = 0;
    if(ex_counter.WORD.h >= flame_max) {
        LIN_status.BIT.SNRE = 1;
        LIN_Sflag_init();
        LIN_break_reception_PR();
        LIN_error();
    }
}else if((TW.TSRW.BIT.OVF) && (TW.TIERW.BIT.OVIE)) {
    TW.TSRW.BIT.OVF = 0;
    ex_counter.WORD.h += 1;
    if((ex_counter.LONG > t_25k)) {
        LIN_status.BIT.NBA = 1;
        LIN_Sflag_init();
        LIN_break_reception_PR();
        LIN_error();
    }
#endif
#else if((ex_counter.LONG >= t_15k) && (in_status.BIT.wu == 3)) {
    in_status.BIT.wu = 0;
    LIN_status.BIT.TOA3B = 1;
    LIN_error();
#endif
2.6.13 SCI3 Interrupt Function

This function processes the SCI3 error detection and reception interrupts.

Figure 23  Flowchart of the SCI3 Interrupt Function
#pragma interrupt(SCI3_int)

void SCI3_int(void){
    unsigned char buff,nmbr,nm,id,dlc;

    if(SCI3.SSR.BYTE & 0x38){
        LIN_status.BIT.SCI = 1;
        LIN_Sflag_init();
        LIN_break_reception_PR();
        LIN_error();
    }else if(SCI3.SSR.BIT.RDRF){
        buff = SCI3.RDR;
        if(in_status.BIT.sync_field){
            if(counter){
                nm = counter & 0x0F;
                nmbr = (counter >> 4) - nm;
                if(nm){
                    LIN_rx_data[nmbr] = buff;
                    r_checksum.WORD += (unsigned short)LIN_rx_data[nmbr];
                    r_checksum.BYTE.data += r_checksum.BYTE.carry;
                    r_checksum.BYTE.carry = 0;
                    counter -= 1;
                }
            }
        }
    }
}

#if __RESPONSE == __ON
if(in_status.BIT.r_id){
    if(nm - 1){
        buff = LINTx_data[(nmbr + 1)];
        SCI3.TDR = buff;
        t_checksum.WORD += buff;
        t_checksum.BYTE.data += t_checksum.BYTE.carry;
        t_checksum.BYTE.carry = 0;
    }else{
        t_checksum.BYTE.data = ~(t_checksum.BYTE.data);
        SCI3.TDR = t_checksum.BYTE.data;
        in_status.BIT.r_id = 0;
    }
}
#else{
    LIN_Sflag_init();
    LIN_break_reception_PR();
    if((r_checksum.BYTE.data ^ buff) != 0xFF){
        LIN_status.BIT.CSE = 1;
        LIN_error();
    }else{
        if((LIN_rx_id == 0x3C) && (LIN_rx_data[0] == 0)){
            LIN_status.BIT.SLEEP = 1;
            LIN_sleep();
        }else{
            LIN_status.BIT.SUC = 1;
            LIN_end();
        }
    }
}
}
else{
    in_status.BYTE &= 0x40;
    LIN_status.BIT.SUC = 0;
    LIN_rx_id = buff;
    switch(LIN_rx_id){
        #if __Res2byte_ID == __ON
        #ifdef __ID00
            case __ID00:
                #endif
        #ifdef __ID01
            case __ID01:
                #endif
        #ifdef __ID02
            case __ID02:
                #endif
        #ifdef __ID03
            case __ID03:
                #endif
        #ifdef __ID04
            case __ID04:
                #endif
        #ifdef __ID05
            case __ID05:
                #endif
        #ifdef __ID06
            case __ID06:
                #endif
        #ifdef __ID07
            case __ID07:
                #endif
        #ifdef __ID08
            case __ID08:
                #endif
        #ifdef __ID09
            case __ID09:
                #endif
        #ifdef __ID0a
            case __ID0a:
                #endif
        #ifdef __ID0b
            case __ID0b:
                #endif
        #ifdef __ID0c
            case __ID0c:
                #endif
        #ifdef __ID0d
            case __ID0d:
                #endif
        #ifdef __ID0e
            case __ID0e:
                #endif
        #ifdef __ID0f
            case __ID0f:
                #endif
        #endif
    }
```c
#ifdef    __ID10
    case    __ID10:
#endif
#ifdef    __ID11
    case    __ID11:
#endif
#ifdef    __ID12
    case    __ID12:
#endif
#ifdef    __ID13
    case    __ID13:
#endif
#ifdef    __ID14
    case    __ID14:
#endif
#ifdef    __ID15
    case    __ID15:
#endif
#ifdef    __ID16
    case    __ID16:
#endif
#ifdef    __ID17
    case    __ID17:
#endif
#ifdef    __ID18
    case    __ID18:
#endif
#ifdef    __ID19
    case    __ID19:
#endif
#ifdef    __ID1a
    case    __ID1a:
#endif
#ifdef    __ID1b
    case    __ID1b:
#endif
#ifdef    __ID1c
    case    __ID1c:
#endif
#ifdef    __ID1d
    case    __ID1d:
#endif
#ifdef    __ID1e
    case    __ID1e:
#endif
#ifdef    __ID1f
    case    __ID1f:
#endif
    counter    =    0x22;
    in_status.BIT.r_id    =    1;
    r_checksum.WORD    =    0;
    LIN_data_set();
    buff    =    LIN_tx_data[0];
    t_checksum.WORD    =    (unsigned short)buff;
```
TW.GRB += flame_max_2.WORD.l;
flame_max = flame_max_2.WORD.h;
TW.TSRW.BIT.IMFB = 0;
TW.TIERW.BIT.IMIEB = 1;
SCI3.TDR = buff;
break;
#endif
#if __Res4byte_ID == __ON
#endif __ID20
    case __ID20:
#endif
#if __ID21
#endif
    case __ID21:
#endif
#if __ID22
#endif
    case __ID22:
#endif
#if __ID23
#endif
    case __ID23:
#endif
#if __ID24
#endif
    case __ID24:
#endif
#if __ID25
#endif
    case __ID25:
#endif
#if __ID26
#endif
    case __ID26:
#endif
#if __ID27
#endif
    case __ID27:
#endif
#if __ID28
#endif
    case __ID28:
#endif
#if __ID29
#endif
    case __ID29:
#endif
#if __ID2a
#endif
    case __ID2a:
#endif
#if __ID2b
#endif
    case __ID2b:
#endif
#if __ID2c
#endif
    case __ID2c:
#endif
#if __ID2d
#endif
    case __ID2d:
#endif
#if __ID2e
#endif
    case __ID2e:
#endif
#if __ID2f
}
case __ID2f:
  #endif
  counter = 0x44;
  in_status.BIT.r_id = 1;
  r_checksum.WORD = 0;
  LIN_data_set();
  buff = LIN_tx_data[0];
  t_checksum.WORD = (unsigned short)buff;
  TW.GRB += flame_max_4.WORD.l;
  flame_max = flame_max_4.WORD.h;
  TW.TSRW.BIT.IMFB = 0;
  TW.TIERW.BIT.IMIEB = 1;
  SCI3.TDR = buff;
  break;

#if __Res8byte_ID == __ON
#ifdef __ID30
  case __ID30:
#endif
#ifdef __ID31
  case __ID31:
#endif
#ifdef __ID32
  case __ID32:
#endif
#ifdef __ID33
  case __ID33:
#endif
#ifdef __ID34
  case __ID34:
#endif
#ifdef __ID35
  case __ID35:
#endif
#ifdef __ID36
  case __ID36:
#endif
#ifdef __ID37
  case __ID37:
#endif
#ifdef __ID38
  case __ID38:
#endif
#ifdef __ID39
  case __ID39:
#endif
#ifdef __ID3a
  case __ID3a:
#endif
#ifdef __ID3b
  case __ID3b:
#endif
#ifdef __ID3d
  case __ID3d:
counter = 0x88;
in_status.BIT.r_id = 1;
r_checksum.WORD = 0;
LIN_data_set();
buff = LIN_tx_data[0];
t_checksum.WORD = (unsigned short)buff;
TW.GRB += flame_max_8.WORD.l;
flame_max = flame_max_8.WORD.h;
TW.TSRW.BIT.IMFB = 0;
TW.TIERW.BIT.IMIEB = 1;
SCI3.TDR = buff;
break;
#endif

/*                  case 0x3C:
    counter = 0x88;
    r_checksum.WORD = 0;
    TW.GRB += flame_max_8.WORD.l;
    flame_max = flame_max_8.WORD.h;
    TW.TSRW.BIT.IMFB = 0;
    TW.TIERW.BIT.IMIEB = 1;
    break;
*/

    case 0x0C:
        counter = 0x88;
        TW.GRB += flame_max_8.WORD.l;
        flame_max = flame_max_8.WORD.h;
        TW.TSRW.BIT.IMFB = 0;
        TW.TIERW.BIT.IMIEB = 1;
        break;
    case 0x0F:
    case 0xBF:
        LIN_Sflag_init();
        LIN_break_reception_PR();
        break;
    default:
        dlc = buff & 0x30;
        if(dlc == 0x20){
            counter = 0x44;
            TW.GRB += flame_max_4.WORD.l;
            flame_max = flame_max_4.WORD.h;
        } else if(dlc == 0x30){
            counter = 0x88;
            TW.GRB += flame_max_8.WORD.l;
            flame_max = flame_max_8.WORD.h;
        } else{
            counter = 0x22;
            TW.GRB += flame_max_2.WORD.l;
            flame_max = flame_max_2.WORD.h;
        }
        r_checksum.WORD = 0;
        TW.TSRW.BIT.IMFB = 0;
        TW.TIERW.BIT.IMIEB = 1;
        break;
    }
} else if(SCI3.RDR == 0x55){
    in_status.BIT.sync_field = 1;
} else{
    LIN_status.BIT.ISFE = 1;
    LIN_Sflag_init();
}
LIN_break_reception_PR();
LIN_error();
}
}
Website and Support

Renesas Technology Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry
csc@renesas.com

Revision Record

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Dec.20.03</td>
<td>—</td>
<td>—</td>
<td>First edition issued</td>
</tr>
<tr>
<td>1.01</td>
<td>Jun.15.07</td>
<td>Pages 1, 3, 6, 7, 15, 49 and 50</td>
<td></td>
<td>Content correction</td>
</tr>
</tbody>
</table>
Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.

2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.

3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.

4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)

5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.

6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.

7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.

8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.

9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.

10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.

11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.

12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.

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

© 2007. Renesas Technology Corp., All rights reserved.