To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 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

April 1st, 2010
Renesas Electronics Corporation

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

Send any inquiries to http://www.renesas.com/inquiry.

RENESAS

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: Master

Introduction

LIN (Local Interconnect Network) Application Note: Slave provides specifications and setting examples that use the on-chip peripheral functions of the H8/36057F 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 H8/36057F CPU

Contents

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 by the master node, as well as the transmission and reception of response frames, is performed.

1.1.1 System Configuration

A sample LIN bus network system configuration is shown in Figure 1.

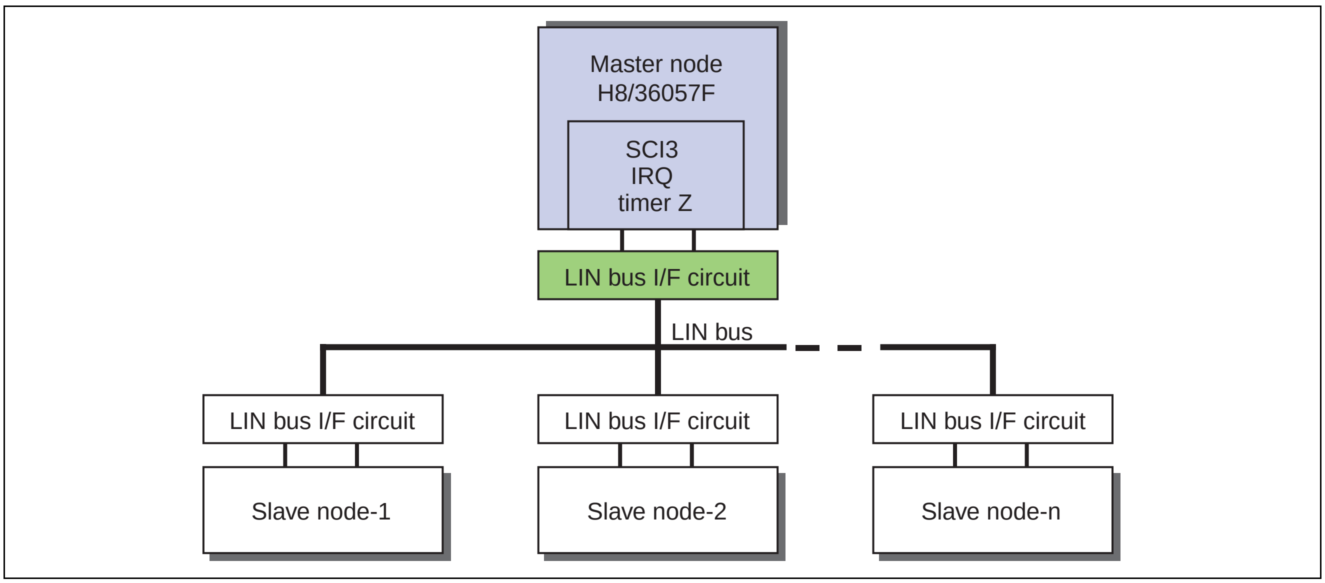


Figure 1 Block Diagram of a System Connected Through the LIN Bus

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 H8/36057F microcomputer (hereinafter referred to as the microcomputer).

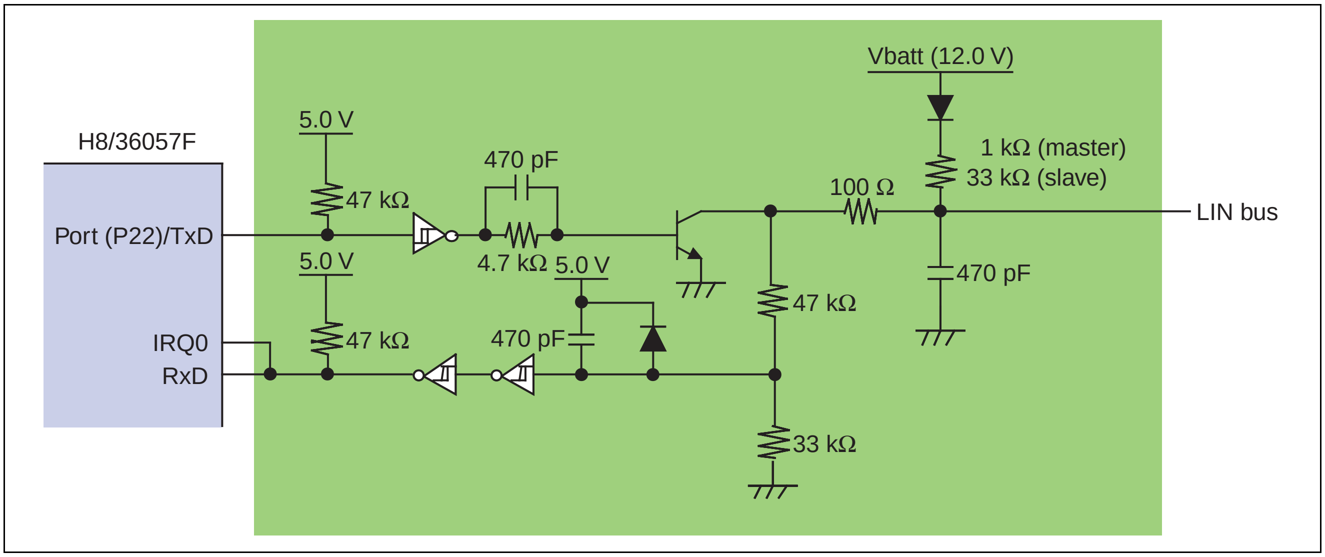


Figure 2 Sample LIN Bus Interface Circuit

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

The structure of a message frame is shown in Figure 3. 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.

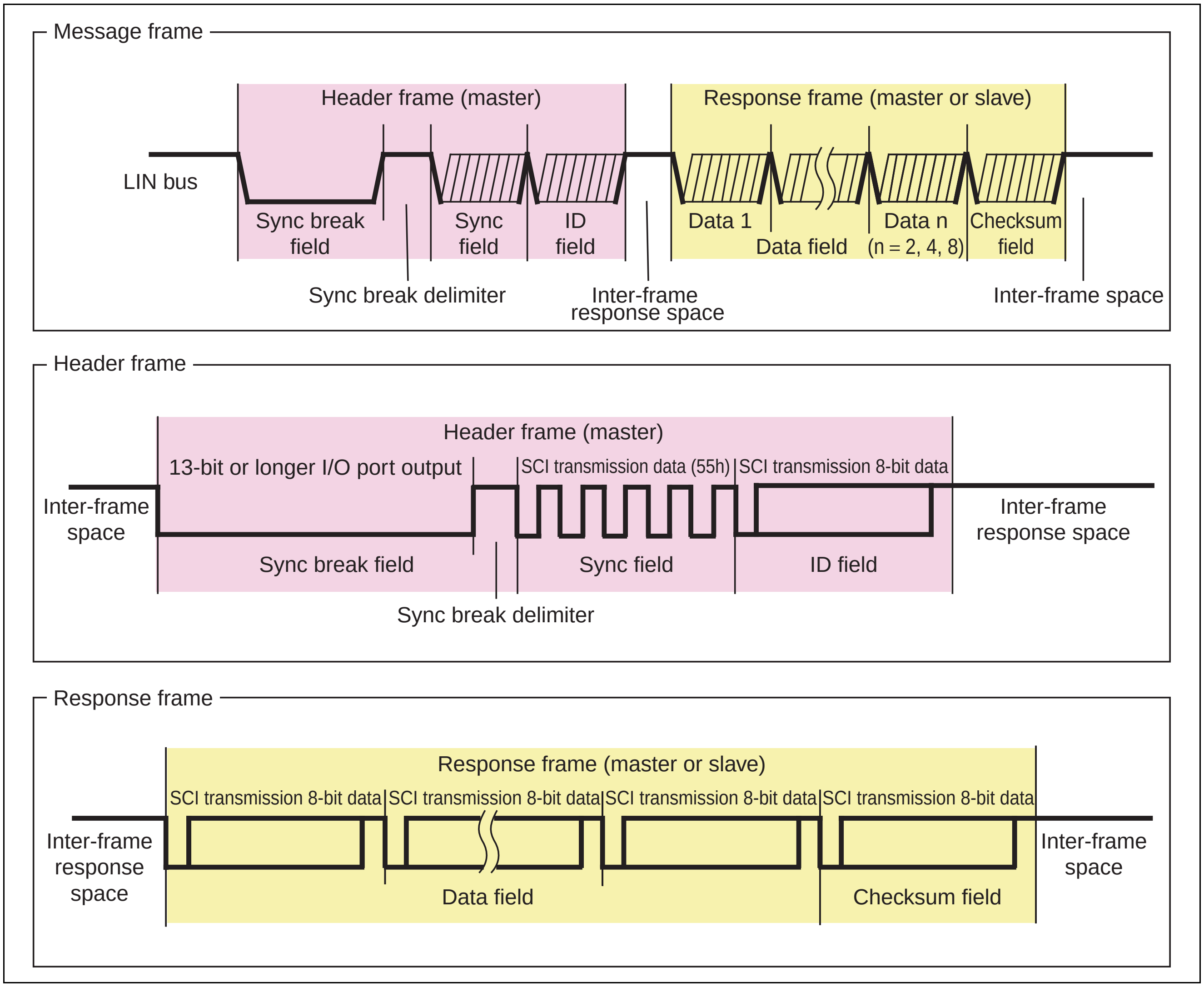


Figure 3 Message Frame Structure

1.2.2 Transmission and Reception of Message Frames

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

1. The master node transmits a header frame.
2. 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 ID is determined by the master node at transmission.

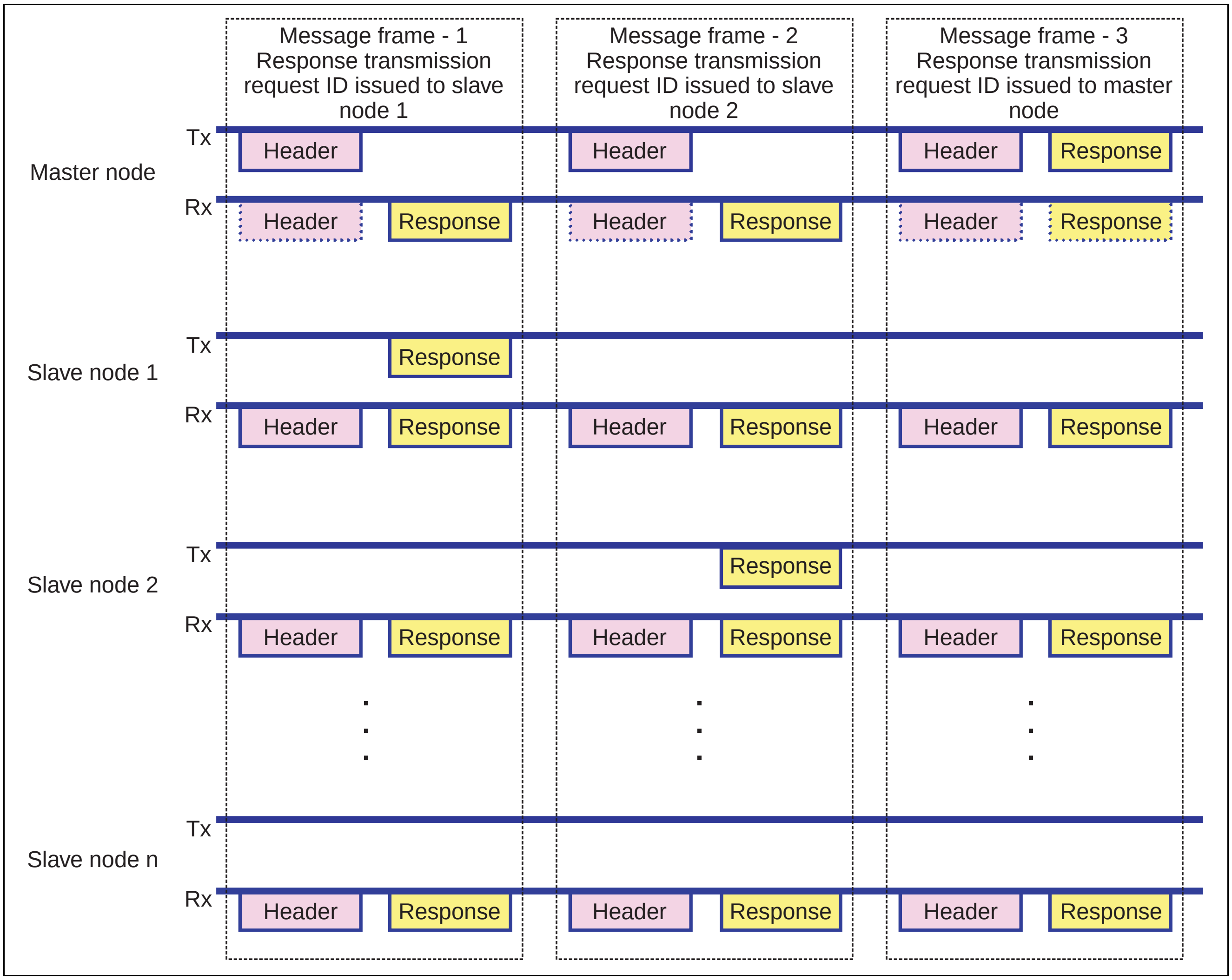


Figure 4 Transmission and Reception of Message Frames

2. Library Software Specifications

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

2.1 Operating Environment

- Device used: H8/36037/57 Group microcomputer
- Operating frequency range (system clock (ϕ osc)): Range equivalent to device operating frequencies. It is necessary to define ϕ osc in LIND.h by considering the LIN communication speed and processing conditions of the user application program. (Refer to Section 2.4.2, LINID.h File Setting Example.)
- Functions used: Table 1 is a list of on-chip peripheral functions to be used with the library, together with their uses.

Table 1 Use of On-Chip (H8/36057) Peripheral Functions

Function		Pin function (pin No.)	Use	Description
I/O port		P22 (pin 46)	Transmission of sync break field	Sync break field is formed by the I/O port (high or low) output.
SCI3 (channel-0)	Transmission	TXD (pin 46)	Transmission of sync field Transmission of ID field Transmission of response frame Transmission of wake-up signal	Asynchronous mode Data length: 8 bits No parity bits One stop bit (with start bit added) LSB first
	Reception	RXD (pin 465)	Reception of response frame	
			Communication error detection	Error detection function in a module
Timer Z (channel-1)		—	Measurement of sync break field dominant period Measurement of sync break delimiter period Measurement of wait period (internal function of the library) Timeout detection	Counting is performed at cycles of ϕ osc/8, and each period is measured.
IRQ		/IRQ0 (pin 51)	Wake-up signal detection	In the standby state, the LIN bus is monitored to detect a falling-edge.

2.2 File Configuration

- LINmtrT.c (Ver.1.00)
  C source file used for (master) microcomputer function setting and communication control for LIN communication in the H8/36057 Group.
- LINID.h (Ver.1.00)
  Include file used to include user-defined items such as the communication transfer rate and ID settings at LINmtrT.c (Ver.1.00) 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_36057.h (Ver.1.00)
  Internal I/O register definition file for the H8/36057 Group

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: Approximately 1.5 Kbytes
- RAM size: Approximately 35 bytes

2.4 Functional Specifications

2.4.1 LIN Communication Specifications

- Node: Slave node supported
- ID: User-defined ID
  A. Response transmission ID
     Zero to 60 IDs (00h to 3bh) 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)
        The master automatically transmits a response frame (8-byte data).
        The data field is set by the user. If the first byte of the data field is 00h, a sleep command is assumed.
     b. Slave response frame ID 3dh (ID field data: 7Dh)
        A response frame (8-byte data) is received.
     c. Extended frame ID 3eh, 3fh (ID field data: FEh, BFh)
        Not supported by this library (Ver.1.00).
  C. ID setting method
     In LINID.h, delete the definition statements (#define __IDm 0xnn (m = 00h to 3bh)) 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 LINmtrT.c.
- Response data length: Specified by the DLC (data length control) bits in the ID (LIN_tx_id) field data to be transmitted.
  LIN_tx_id = 00h to 1fh : 2 bytes
  = 20h to 2fh : 4 bytes
  = 30h to 3Dh : 8 bytes
- Communication transfer rate: The communication transfer rate used is defined in LINID.h.
  From the system clock (ϕ 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 ϕ 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. By calling the function (LIN_transmit_wake_up) from the user application program, the wake-up signal is transmitted on the LIN bus.
  Including the wake-up signal reception function
  A definition statement (#define __R_WAKEUP __ON) in LINID.h includes the wake-up reception function. Even when the H8/36057 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 node does not transmit a wake-up signal.
2. Wake-up signal detection (falling-edge detection) through IRQ0 (external interrupt) is performed.
3. Response frames are transmitted to the following four IDs:

ID (ID bit + DLC bits)	(including parity bits)
01h	(C1h)
12h	(92h)
23h	(A3h)
34h	(B4h)

4. The system clock (ϕ osc) is 20 [MHz].
5. The LIN communication transfer rate is 9600 [bit/sec].

An example of the settings made based on the specifications described in items 1. to 5., above, is given below.

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

```
/******************************************************************************************/
/*                                                                                        */
/*                        LINID.h    Ver.1.00                                             */
/*                                                                                        */
/******************************************************************************************/
#define     __ON    1               /*  This line must not be changed or deleted.       */
#define     __OFF   0               /*  This line must not be changed or deleted.       */

/******************************************************************************************/
/*      Setting of wake-up signal transmission function                                   */
/******************************************************************************************/
/*#define   __T_WAKEUP __ON         /*  When transmitting a wake-up signal, define this line.   */
#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                                              */
/******************************************************************************************/
/*      2-byte data                                                                       */
/*----------------------------------------------------------------------------------------*/

#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            /*  Transmit response to ID field C1h.                */
/*#define  __ID02  0x42                                          /*                       */
/*#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            /*  Transmit response to ID field 92h.                */
/*#define  __ID13  0xD3                                          /*                       */
/*#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 2-byte response data transmission ID,
/*define this line.                                                                  */

#if    __Res4byte_ID  ==  __ON

/*#define   __ID20  0x20                                  /*                         */
/*#define   __ID21  0x61                                  /*                         */
/*#define   __ID22  0xE2                                  /*                         */
#define     __ID23  0xA3            /*  Transmit response to ID field A3h.           */
/*#define   __ID24  0x64                                                             */
/*#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            /*  Transmit response to ID field B4h.           */
/*#define   __ID35  0xF5                                  /*                         */
/*#define   __ID36  0x76                                  /*                         */
/*#define   __ID37  0x37                                  /*                         */
/*#define   __ID38  0x78                                  /*                         */
/*#define   __ID39  0x39                                  /*                         */
/*#define   __ID3a  0xBA                                  /*                         */
/*#define   __ID3b  0xFB                                  /*                         */

#endif

/*****************************************************************************************/
/*      System clock (φosc) definition section                                            */
/*****************************************************************************************/
#define     OSC_Hz      20000000    /* φ osc=20.000MHz → 20000000                        */
/*#define   OSC_Hz      16000000                          /* φ osc=16.000MHz → 16000000 */
```

```
/*#define  OSC_Hz     10486000                                   /* ϕ osc=10.486MHz → 10486000 */
/*#define  OSC_Hz     10000000                                   /* ϕ osc=10MHz     → 10000000 */
/*#define  OSC_Hz     9830400                                    /* ϕ osc=9.8304MHz →  9830400 */
/*#define  OSC_Hz     8000000                                    /* ϕ osc=8.0000MHz →  8000000 */
/*#define  OSC_Hz     7372800                                    /* ϕ osc=7.3728MHz →  7372800 */
/*#define  OSC_Hz     4915200                                    /* ϕ osc=4.9152MHz →  4915200 */
/*#define  OSC_Hz     2457600                                    /* ϕ osc=2.4576MHz →  2457600 */

/***********************************************************************************************/
/*        Baud rate definition module                                                          */
/***********************************************************************************************/
/*#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     19200                                      /* 19200bps    → 19200        */
/*#define  B_rate     20000                                      /* 20000bps    → 20000        */

/***********************************************************************************************/
/*        Library constant calculation module    The following must not be changed or deleted.  */
/***********************************************************************************************/
#define    t_1_data        ((((OSC_Hz) / (B_rate)) + 0x04) >>3)
#define    t_2_data        ((((OSC_Hz) / (B_rate)) + 0x02) >>2)
#define    t_3_data        (t_1_data + t_2_data)
#define    t_13_data       ((((13  * ((OSC_Hz) >>2)) / (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 module       The following must not be changed or
deleted.                                                                                    */
/**********************************************************************************************/
#ifndef     __LIN_LIB

extern     void   LIN_end(void);
extern     void   LIN_data_set(void);
extern     void   LIN_error(void);
extern     void   LIN_initialize(void);
extern     void   LIN_transmit_header(void);
#if     __T_WAKEUP     ==     __ON
extern     void   LIN_transmit_wake_up(void);
#endif
#if     __R_WAKEUP     ==     __ON
extern     void   LIN_wake_up(void);
extern     void   LIN_wake_up_PR(void);
#endif

extern     volatile unsigned char    LIN_tx_id;
extern     volatile unsigned char    LIN_tx_data[8];
extern     volatile unsigned char    LIN_rx_id;
```

```
extern     volatile unsigned char    LIN_rx_data[8];
extern     volatile union {
              unsigned char   BYTE;
              struct {
                unsigned char   wk7            :1;
                unsigned char   CSE            :1;
                unsigned char   wk5            :2;
                unsigned char   SNRE           :1;
                unsigned char   SCI            :1;
                unsigned char   SUC            :1;
                unsigned char   Ready          :1;
            }    BIT;
}    LIN_status;
extern     volatile union {
             unsigned char BYTE;
             struct {
               unsigned char   SB_DEL          :2;
               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 the user application program.

- Interface by function (module) call
  The user application program calls functions from the library to initialize the necessary on-chip (H8/36057) peripheral functions for LIN communication control, transmitting a header frame, controlling wake-up signal transmission, and preparing for wake-up signal reception.

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

Function name	Description
LIN_initialize	Initializes necessary on-chip (H8/36057) peripheral functions for LIN communication control.
LIN_transmit_header	Starts header frame transmission.
LIN_transmit_wake_up	Transmits a wake-up signal.
LIN_wake_up_PR	Makes the preparations needed to receive the wake-up signal.

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

Function name	Description
LIN_wake_up	Function for controlling the user application program when a wake-up signal is detected
LIN_data_set	Function for controlling the user application program before response-frame transmission
LIN_end	Function for controlling the user application program after the completion of message-frame transmission or reception
LIN_error	Function for controlling the user application program when a LIN communication error is detected

- Operation overview
  Operation when the message frame is transmitted or received is shown in Figure 5, and the operation when the error is detected or a wake-up signal is transmitted or received is shown in figure 5.

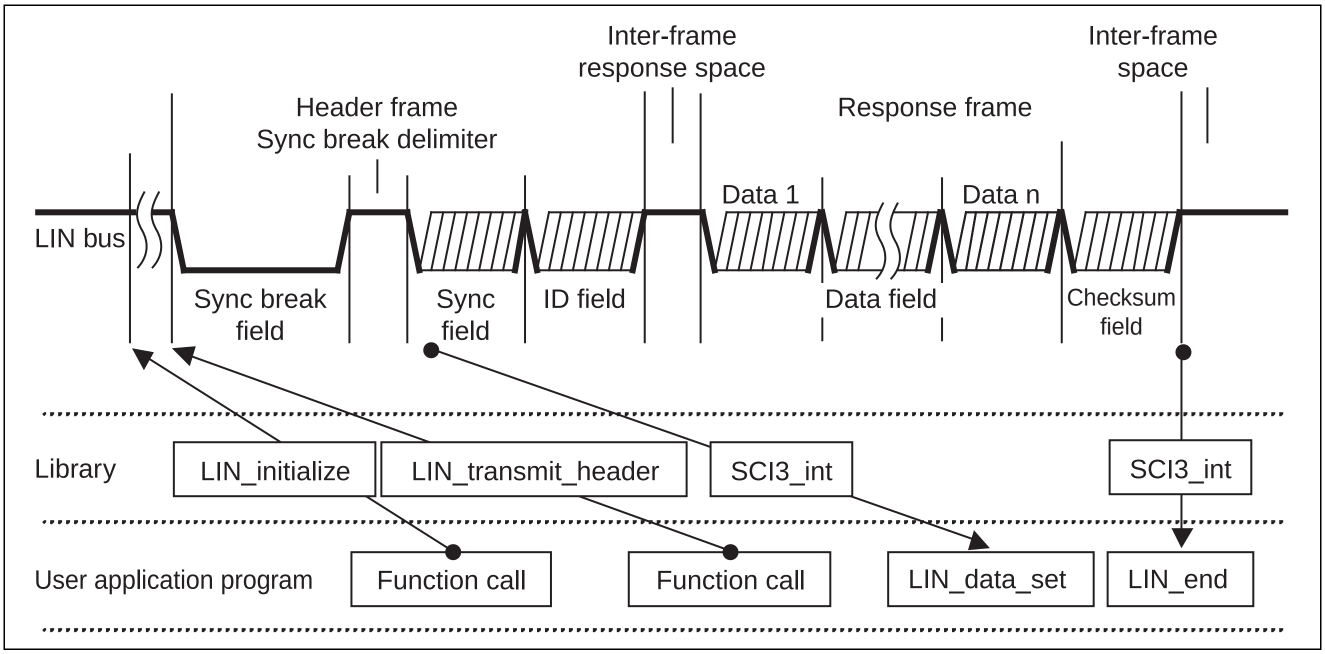


Figure 5 Operation Overview at Message Frame Transmission or Reception

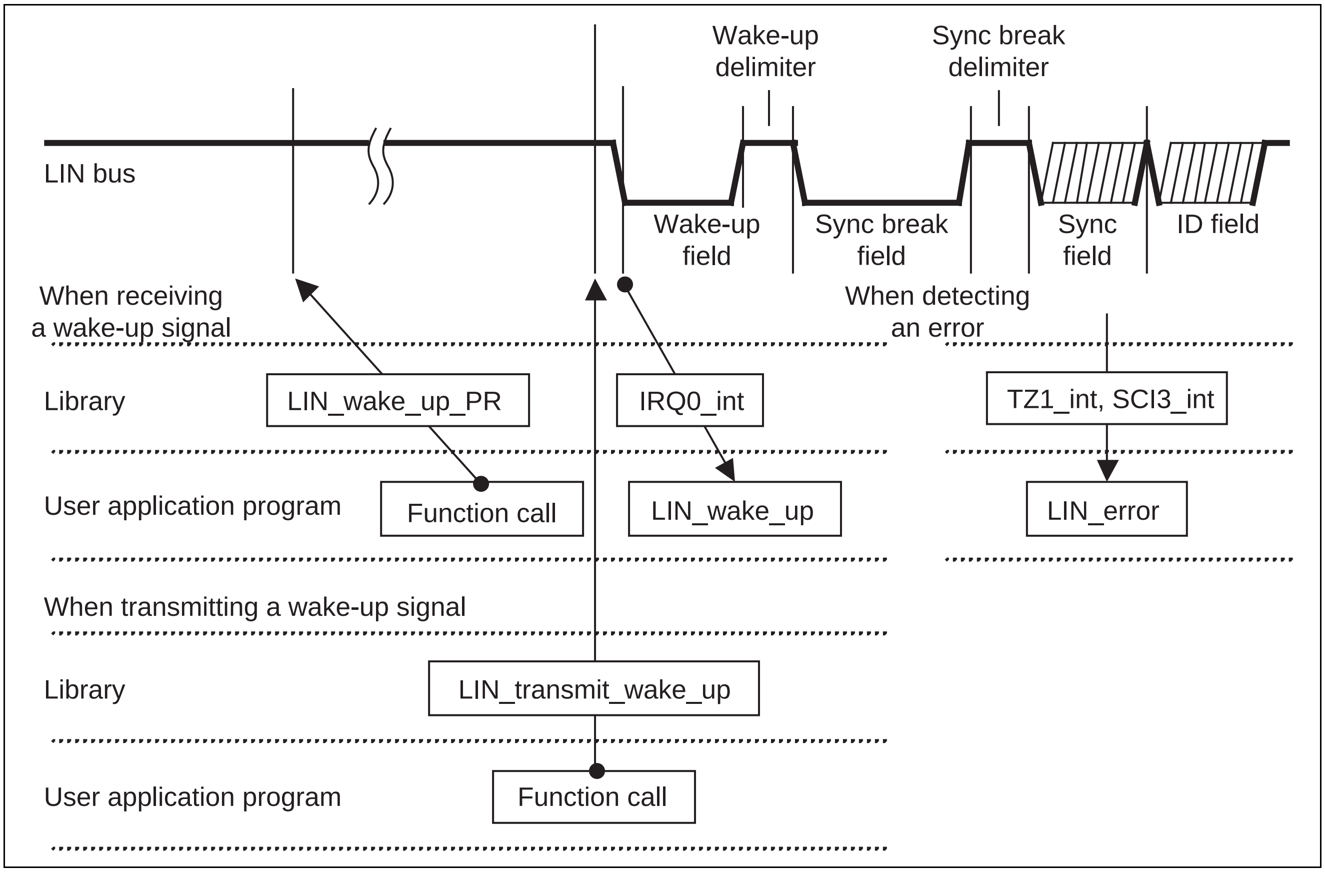


Figure 6 Operation at Error Detection and Wake-Up Signal Transmission or Reception

- 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

Label name (variable name)	Data type	Description
LIN_tx_id	unsigned char	Sets the transmit ID (ID bit + DLC bit) used when a header frame is transmitted. (See Table 5, "ID List".)
LIN_tx_data[0] to [7]	unsigned char (array)	Sets the transmission data when transmitting a response frame.
LIN_rx_id	unsigned char	Stores a received ID.
LIN_rx_data[0] to [7]	unsigned char (array)	Stores received response data.
LIN_status	(Structure)	Communication state
LIN_status.BYTE	Byte access unsigned char	
	Bit access	
LIN_status.BIT.wk7	Bit 7	Reserved bit

Label name (variable name)	Data type	Description
LIN_status.BIT.CSE	Bit 6	Checksum error flag Set condition : A checksum error is detected when a response is received.
LIN_status.BIT.wk5	Bit 5 to 4	Reserved bits
LIN_status.BIT.SNRE	Bit 3	Slave-not-responding error Set condition : Reception of a response frame from a slave is not completed within a specified period.
LIN_status.BIT.SCI	Bit 2	SCI error Set condition : An error in the SCI3 module (overrun error or framing error) is detected.
LIN_status.BIT.SUC	Bit 1	Message-frame normal-reception completion flag Set condition : A response frame has been received normally. Clearing condition : Transmission of the next response reception ID is started.
LIN_status.BIT.Ready	Bit 0	Header frame transmission ready flag Set condition : Initialization has been completed. Message frame transmission/ reception has been completed. A communication error is detected. Clearing condition : A message frame is transmitted or received.
LIN_control	(Structure)	Communication control
LIN_control.BYTE	Byte access unsigned char	
	Bit access	
LIN_control.BIT.SB_DEL	Bit 7 to 6	Bits for setting the sync break delimiter length Bit 7 Bit 6 0 0 : 1 bit 0 1 : 1 bit 1 0 : 2 bits 1 1 : 3 bits
LIN_control.BIT.WU	Bit 5	(Wake-up control bit)
LIN_control.BIT.wk4	Bits 4 to 0	Reserved bits

Table 5 ID List

LIN_tx_id setting		ID field transmission data		Response data length	LIN_tx_id setting		ID field transmission data		Response data length
Dec.	Hex.	Dec.	Hex.		Dec.	Hex.	Dec.	Hex.	
0	0x00	128	0x80	2	32	0x20	32	0x20	4
1	0x01	193	0xC1	2	33	0x21	97	0x61	4
2	0x02	66	0x42	2	34	0x22	226	0xE2	4
3	0x03	3	0x03	2	35	0x23	163	0xA3	4
4	0x04	196	0xC4	2	36	0x24	100	0x64	4
5	0x05	133	0x85	2	37	0x25	37	0x25	4
6	0x06	6	0x06	2	38	0x26	166	0xA6	4
7	0x07	71	0x47	2	39	0x27	231	0xE7	4
8	0x08	8	0x08	2	40	0x28	168	0xA8	4
9	0x09	73	0x49	2	41	0x29	233	0xE9	4
10	0x0A	202	0xCA	2	42	0x2A	106	0x6A	4
11	0x0B	139	0x8B	2	43	0x2B	43	0x2B	4
12	0x0C	76	0x4C	2	44	0x2C	236	0xEC	4
13	0x0D	13	0x0D	2	45	0x2D	173	0xAD	4
14	0x0E	142	0x8E	2	46	0x2E	46	0x2E	4
15	0x0F	207	0xCF	2	47	0x2F	111	0x6F	4
16	0x10	80	0x50	2	48	0x30	240	0xF0	8
17	0x11	17	0x11	2	49	0x31	177	0xB1	8
18	0x12	146	0x92	2	50	0x32	50	0x32	8
19	0x13	211	0xD3	2	51	0x33	115	0x73	8
20	0x14	20	0x14	2	52	0x34	180	0xB4	8
21	0x15	85	0x55	2	53	0x35	245	0xF5	8
22	0x16	214	0xD6	2	54	0x36	118	0x76	8
23	0x17	151	0x97	2	55	0x37	55	0x37	8
24	0x18	216	0xD8	2	56	0x38	120	0x78	8
25	0x19	153	0x99	2	57	0x39	57	0x39	8
26	0x1A	26	0x1A	2	58	0x3A	186	0xBA	8
27	0x1B	91	0x5B	2	59	0x3B	251	0xFB	8
28	0x1C	156	0x9C	2	60	0x3C	60	0x3C	8
29	0x1D	221	0xDD	2	61	0x3D	125	0x7D	8
30	0x1E	94	0x5E	2	(62)	(0x3E)	(254)	(0xFE)	—
31	0x1F	31	0x1F	2	(63)	(0x3F)	(191)	(0xBF)	—

2.5 Operation

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

2.5.1 Transmission of a Header Frame

When the header frame transmission ready flag (LIN_status.BIT.Ready) is set to 1, the user program starts header frame transmission by calling the LIN_transmit_header function.

Before the LIN_transmit_header function can be called, the bit length of the sync break delimiter (LIN_control.BIT.SB_DEL) and the transmission ID (LIN_tx_id) must be set. (Refer to Table 4 and Table 5.)

1. Transmission of a Sync Break Field:
   The I/O port (P22 (a multiplexed pin also functioning as TxD)) output function outputs the sync break field dominant state for a period of about 13 bits.

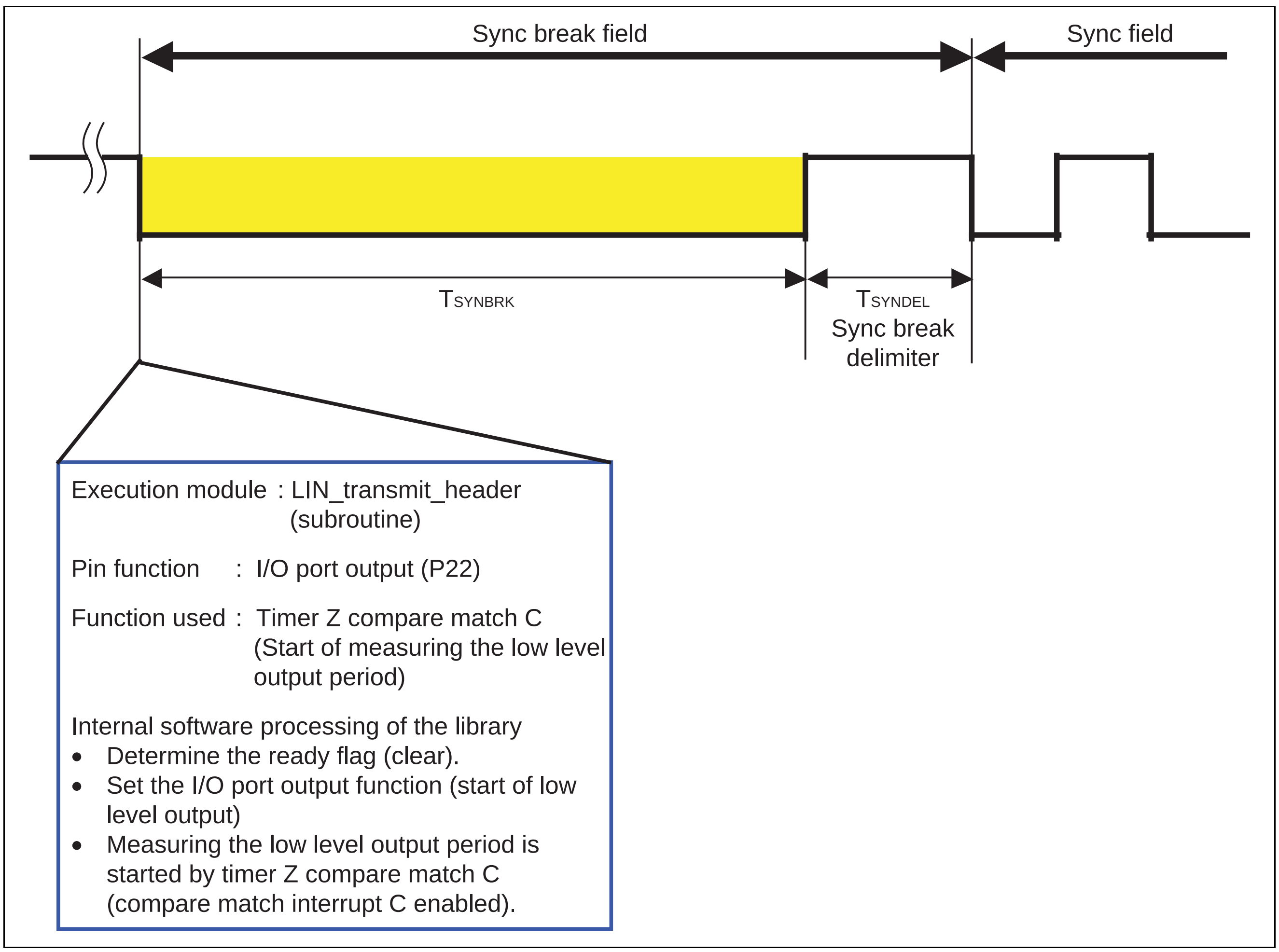


Figure 7 Output of the Sync Break Field Dominant Period

2. Transmission of a Sync Break Delimiter:
   The I/O port output function outputs a sync break delimiter (recessive period).
   After the output of the recessive state for the period set by the LIN_control.BIT.SB_DEL setting, which is about 1 to 3 bits long, the port function changes to the TxD pin function (SCI3: channel-0 (called SCI hereafter) transmission function).

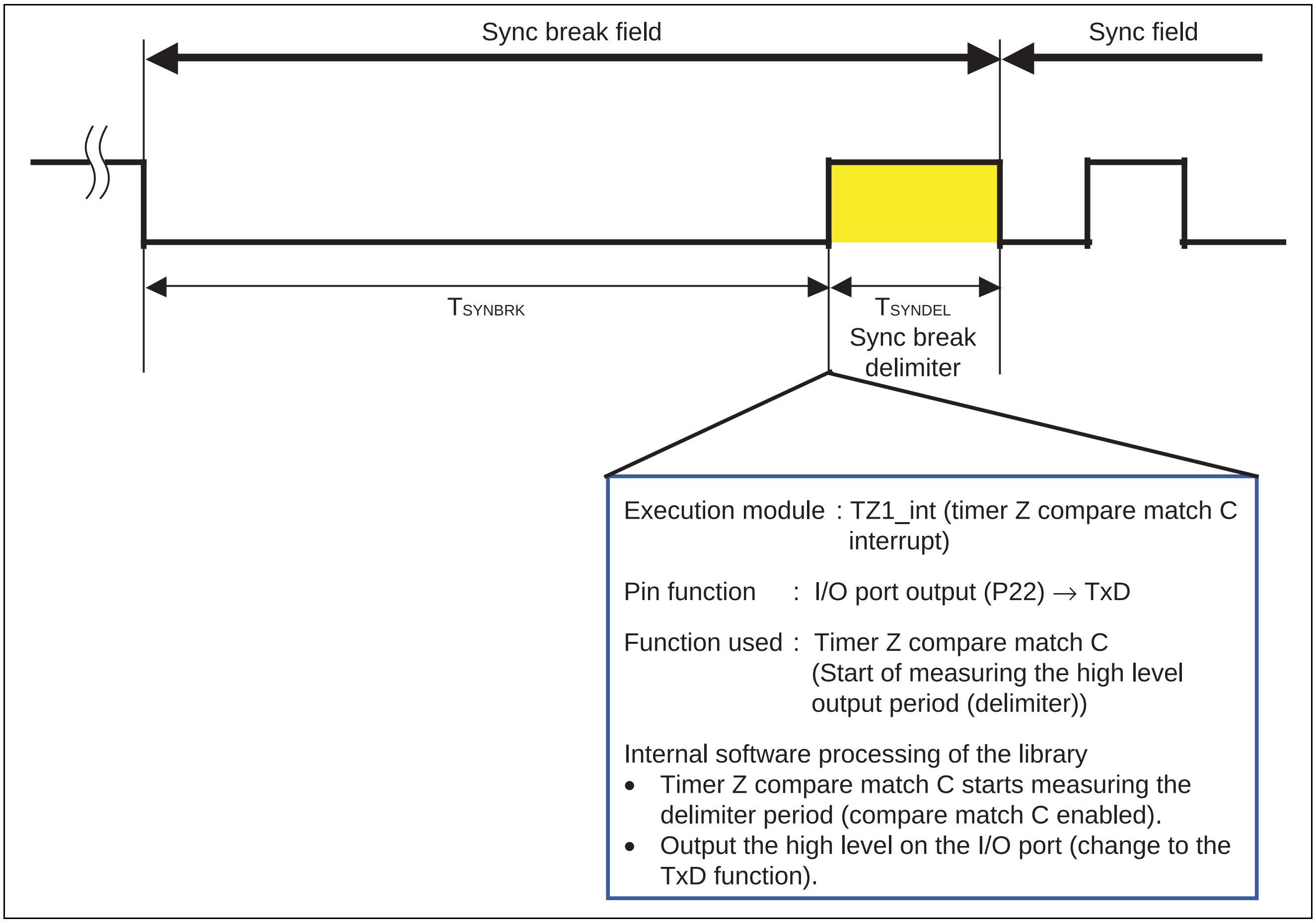


Figure 8 Sync Break Delimiter Output

3. Transmission of a Sync Field:
   The SCI transmission function transmits data 55h.

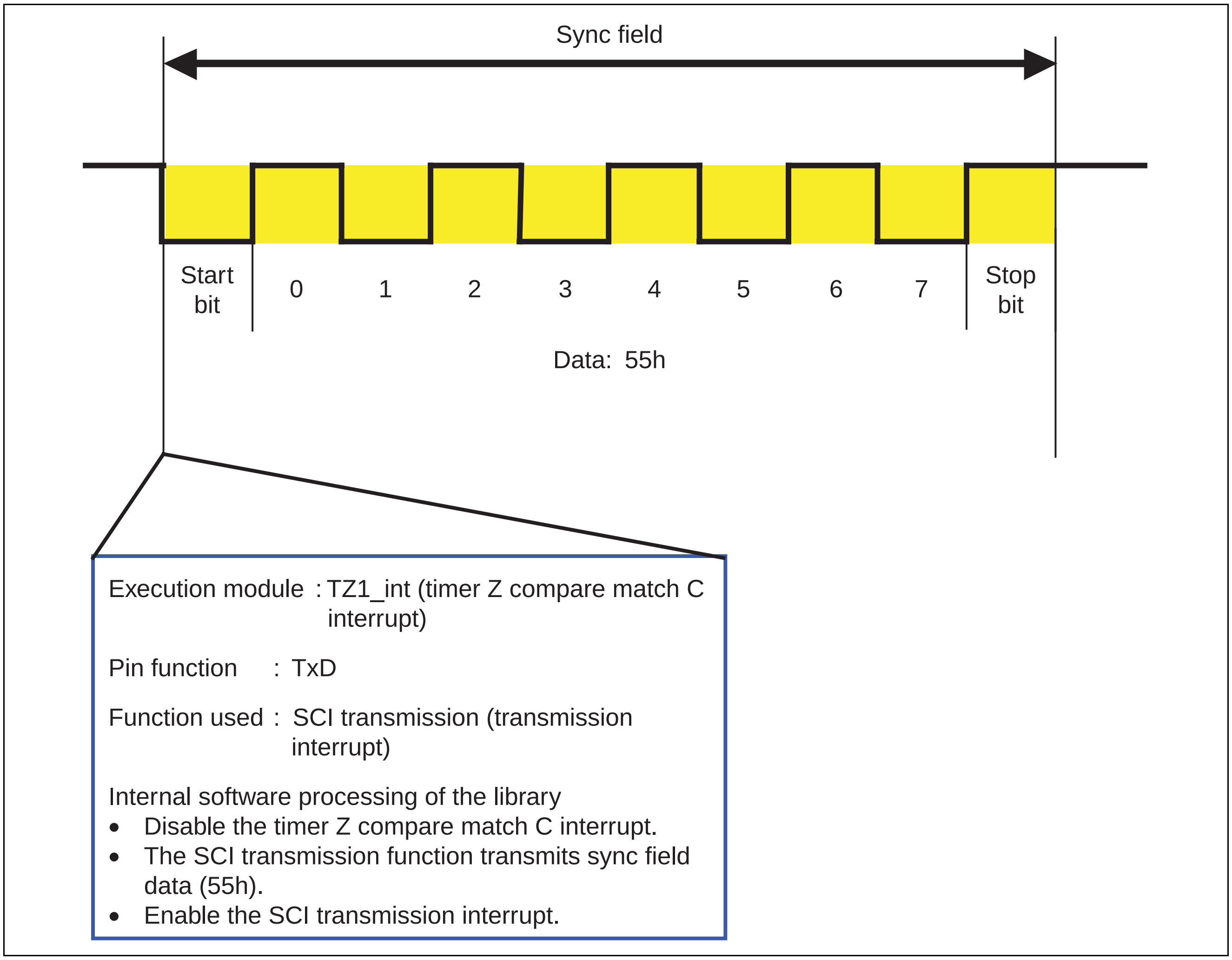


Figure 9 Sync Field Transmission

4. Transmission of an ID Field:
   The ID field data is transmitted by the SCI transmission function. The ID field data includes the LIN_tx_id setting, value and parity bits are automatically added. (Refer to Table 5.)

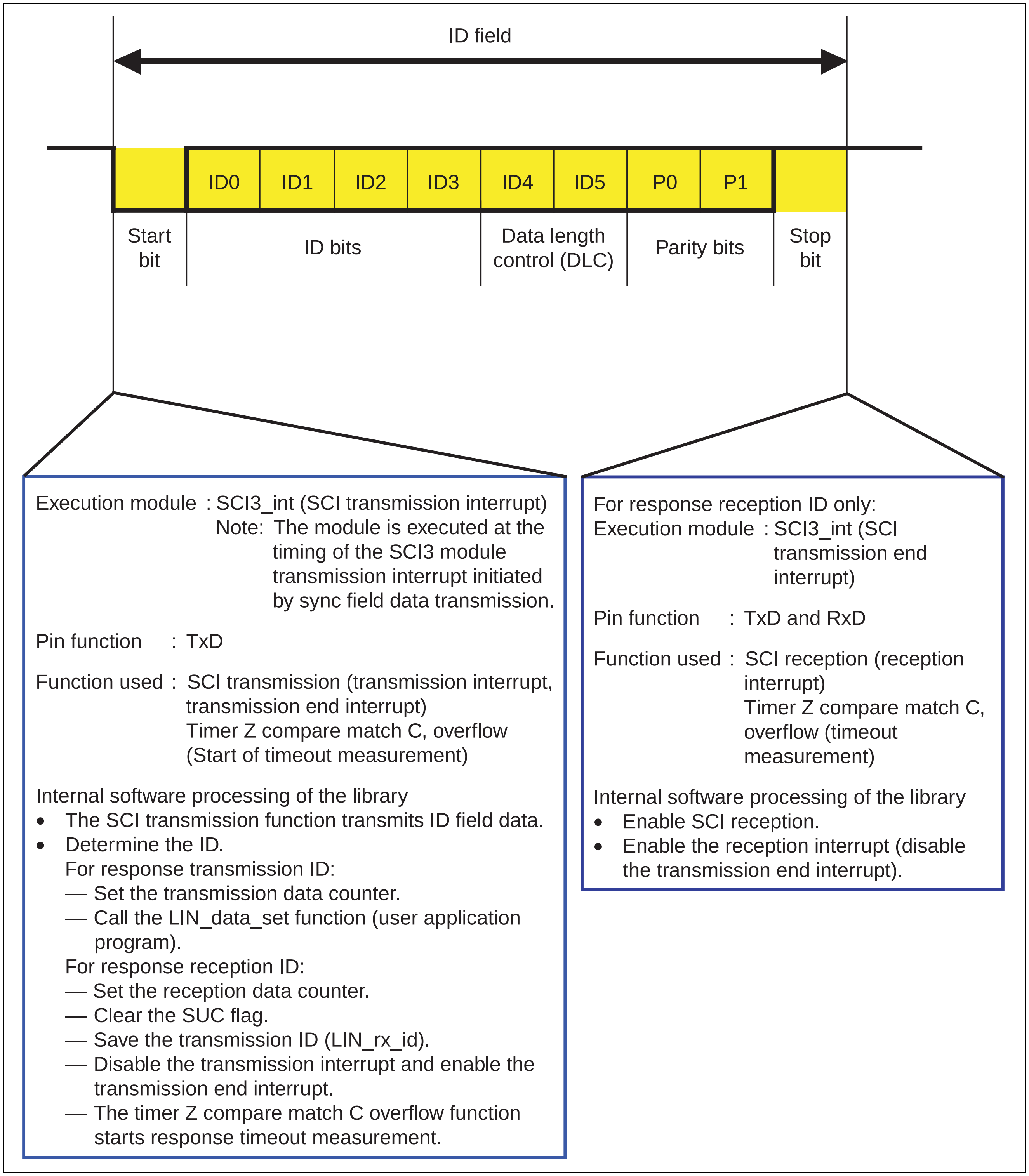


Figure 10 ID Field Data Transmission and Determination

2.5.2 Transmission and Reception of a Response Frame

When the ID field transmission data includes a response transmission ID, the SCI transmission function transmits a response frame. When the ID field transmission data includes a response reception ID, the SCI reception function receives a response frame.

1. Transmission of a Data Field:
   The SCI transmission function transmits a data field.
   The transmission data is set in LIN_tx_data[0] to LIN_tx_data [7], and as many bytes of data as the value (2, 4, or 8 bytes) set by the DLC bits of the ID field data are transmitted sequentially from LIN_tx_data[0].

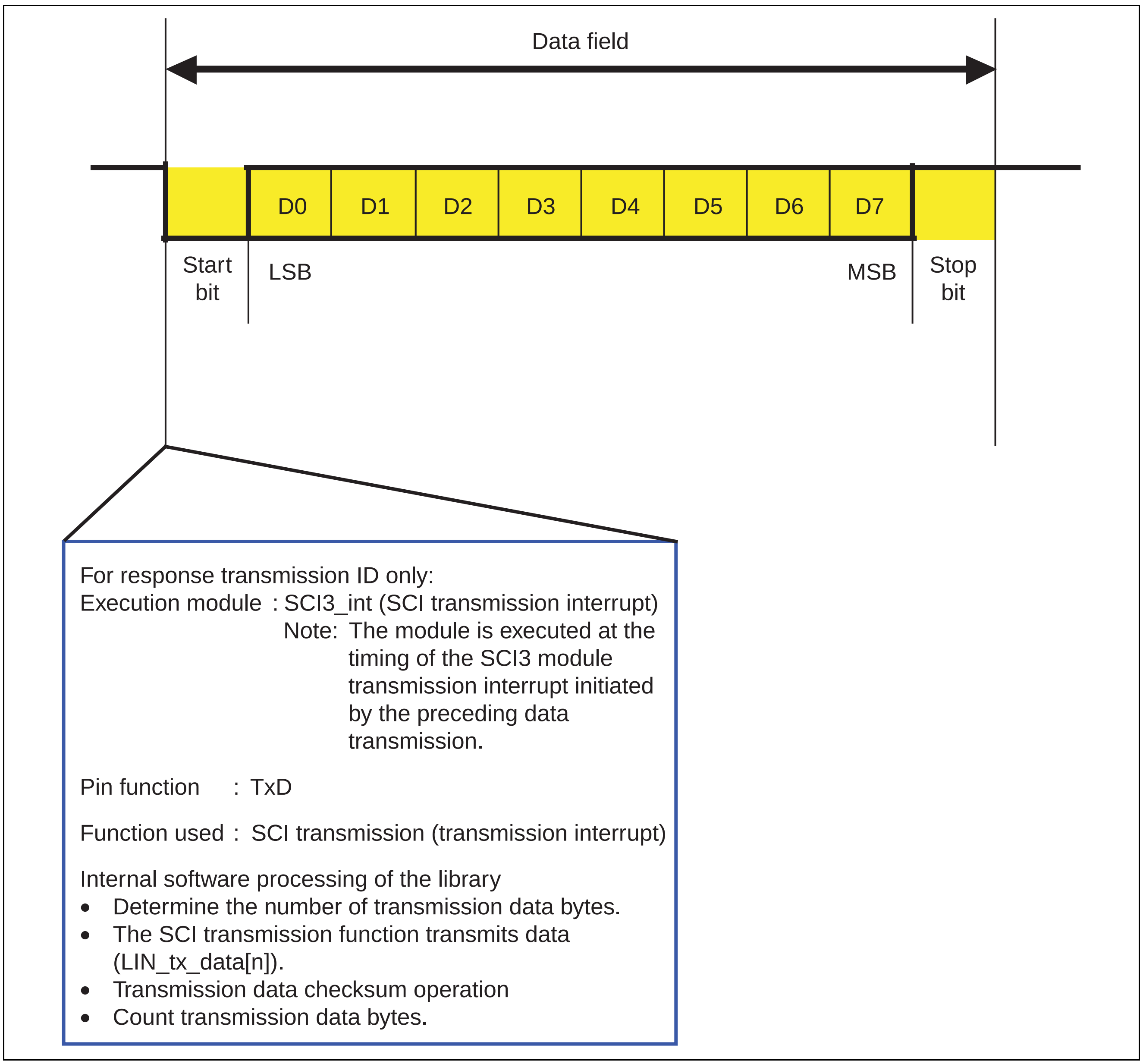


Figure 11 Transmission of a Data Field

2. Transmission of a Checksum Field:
   The SCI transmission function transmits a checksum field.

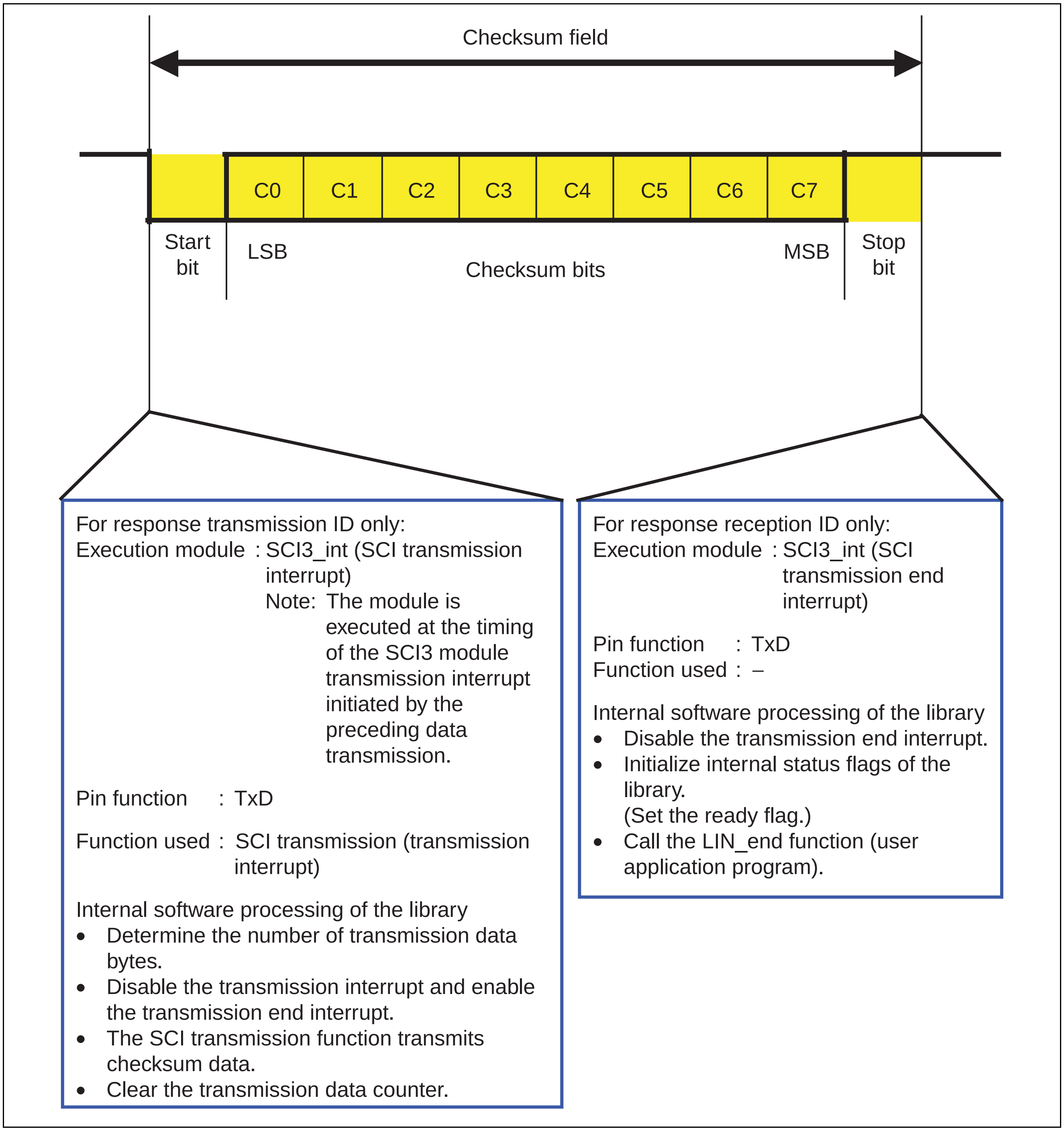


Figure 12 Checksum Field Transmission

3. Reception of the Data Field:
   The SCI reception function receives the data field.
   The received data is saved in LIN_rx_data[0] to LIN_rx_data[7] sequentially from LIN_rx_data[0]. Only the bytes that are received are saved.

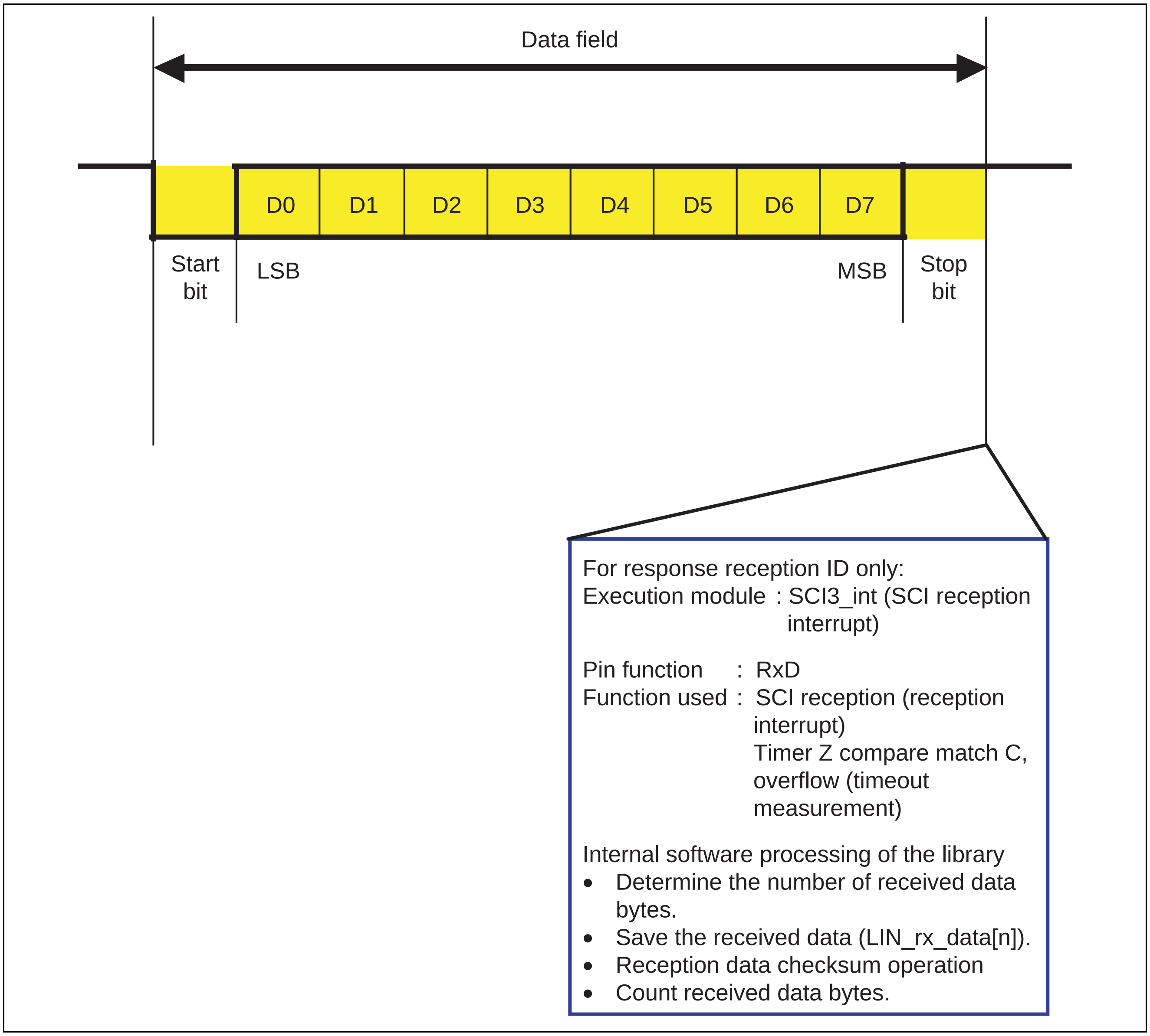
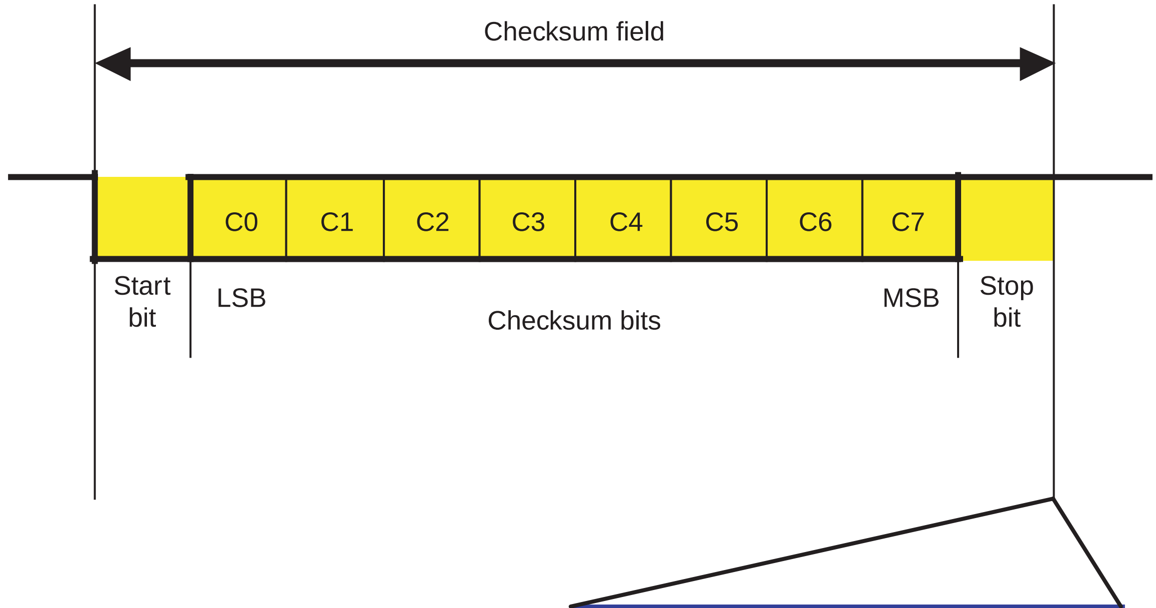


Figure 13 Reception of a Data Field

4. Reception of a Checksum Field:
   The SCI reception function receives a checksum field, compares it with the operation result from the received data field, then makes a decision.



For response reception ID only:
Execution module : SCI3_int (SCI reception interrupt)

Pin function : RxD
Function used : –

Internal software processing of the library

- Determine the number of received data bytes.
- Disable the timer Z compare match C interrupt and overflow interrupt (stop response timeout measurement).
- Disable SCI reception (disable the reception interrupt).
- Initialize internal status flags of the library.
- Determine the checksum.
  If the determination is acceptable
  –– Set the SUC flag and ready flag.
  –– Call the LIN_end function (user application program).
  If the determination is not acceptable
  –– Set the CSE flag and ready flag.
  –– Call the LIN_eror function (user application program).

Figure 14 Checksum Field Reception and Determination

2.5.3 Transmission and Reception of a Wake-Up Signal

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

The IRQ0 (hereafter called IRQ) 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 at compilation, allowing the SCI transmission function to transmit a wake-up signal (transmit data : 80h) when the user application program calls the LIN_transmit_wake_up function.
   The library does not control the wake-up delimiter output and the retry transmission. Although the ready flag is set during the transmission of a wake-up signal, if header frame transmission is started before the other nodes (slave nodes) complete the preparation needed for LIN communication, communication may not be performed normally.

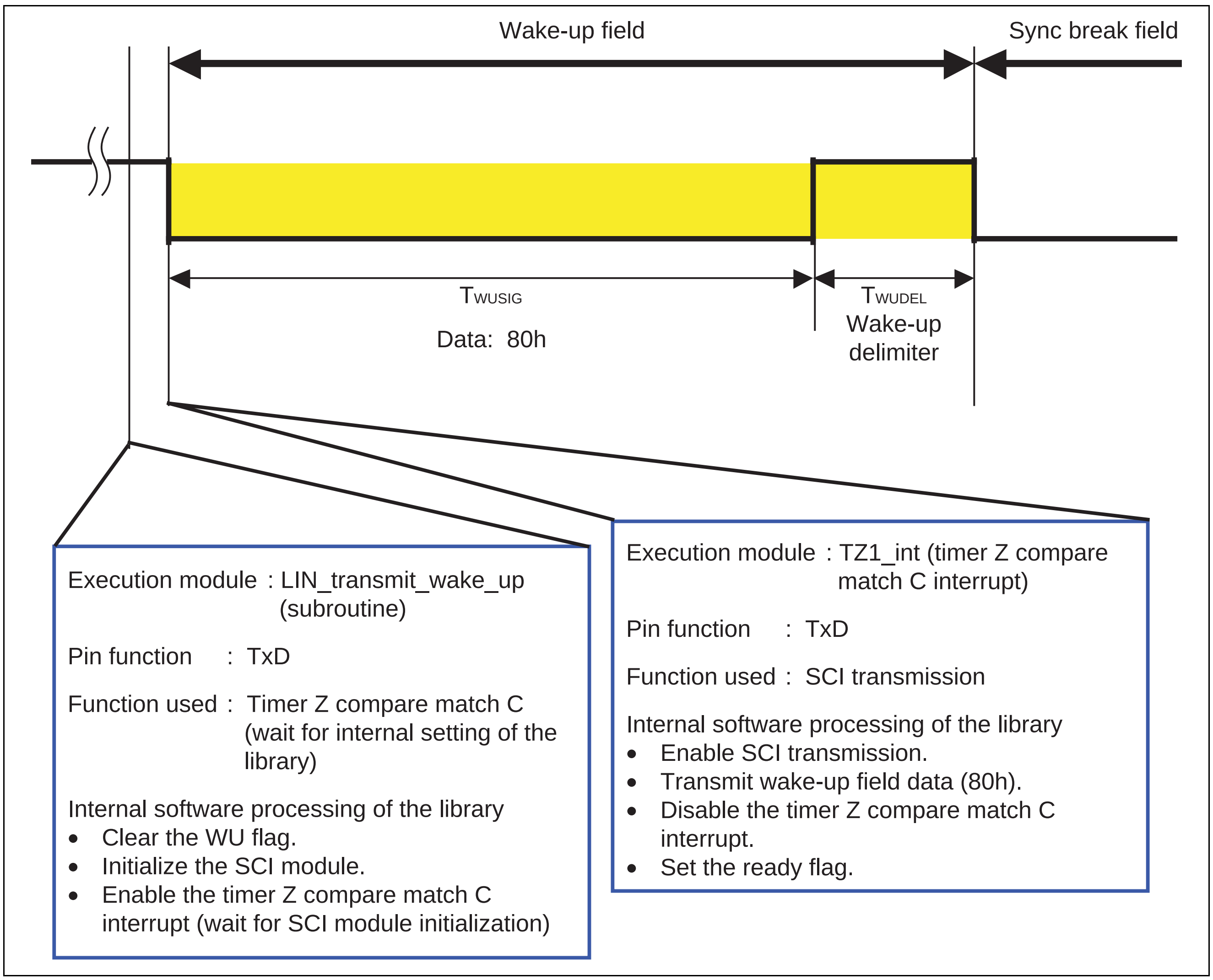


Figure 15 Transmission of a Wake-Up Signal

2. Reception of a Wake-up Signal:
   A definition statement in LINID.h (#define __R_WAKEUP __ON) includes the wake-up signal reception function at compilation, 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.
   The library detects only falling-edges, without verifying the wake-up field data.

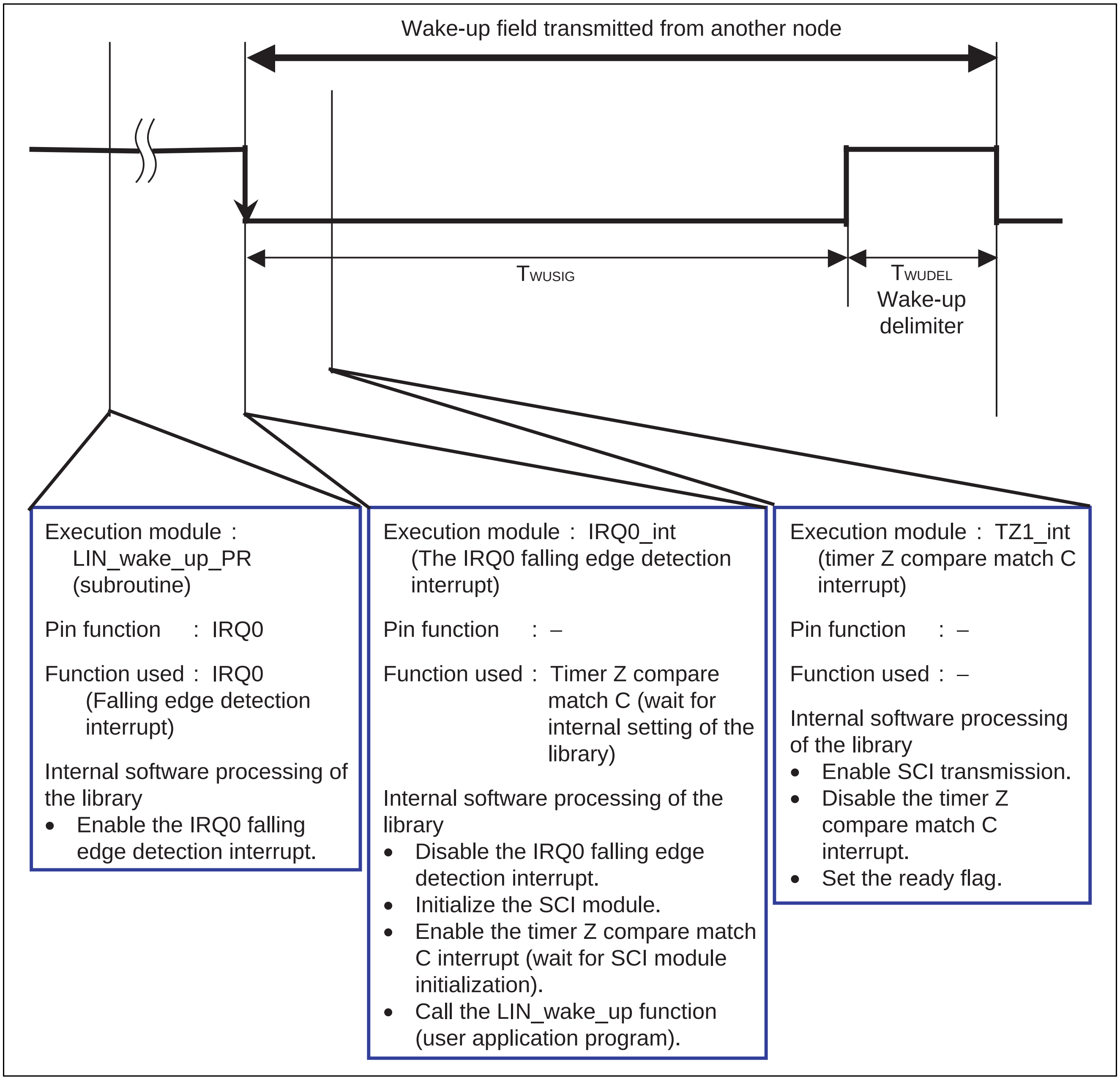


Figure 16 Reception of a Wake-Up Signal

2.5.4 Issuing a Sleep Command

The sleep command (when a command frame ID (3CH) is transmitted, and the first byte of the response transmission data is 00h), which is defined by the LIN communication protocol (LIN Protocol Specification Rev 1.2 , Rev 1.3 Draft 7), is transmitted by calling the LIN_transmit_header function with 3Ch set in LIN_tx_id and 00h set in LIN_tx_data[0].

The library does not contain any special operations that are to be performed after the transmission of the sleep command (flag setting, microcomputer operation mode change, and so on).

2.6 Software Description

This section explains the library software.

2.6.1 Including Header Files

The standard library (machine.h), the H8/36057 on-chip peripheral register definition file (H8_36057.h), and the LIN library definition file (LINID.h) are included.

```
#include    <machine.h>
#include    "H8_36057.h"
#define    __LIN_LIB
#include    "LINID.h"
```

2.6.2 Defining Functions

Functions (modules) in the library must be defined.

The inclusion of the LIN_transmit_wake_up function is selected by the __T_WAKEUP definition in LINID.h. Similarly, 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.

```
void    LIN_initialize(void);
void    LIN_system_init(void);
void    LIN_port_init(void);
void    LIN_sci_init(void);
void    LIN_timerZ_init(void);
void    LIN_Sflag_init(void);
void    LIN_end(void);
void    LIN_data_set(void);
void    LIN_error(void);
void    LIN_transmit_header(void);

#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 Internal Constants for the Library

Constants used in the library must be defined.

Table 6 Internal Constants for the Library

Label name (variable name)	Data type	Description
id_field[0] to [63]	unsigned char (array)	ID field transmission data (refer to the list of IDs in Table 5.)
wait_time[0] to [3]	unsigned short (array)	Wait settings for internal control of the library (used at SCI3 module initialization and sync break delimiter period setting)
t_13	unsigned short	Setting of sync break field dominant period (13-bit period)
flame_max_2	unsigned long	Maximum response timeout value
flame_max_4		
flame_max_8		
baudrate	unsigned short	Baud rate setting for the SCI3 module

```
const  unsigned char  id_field[64]  =  { 0x80, 0xC1, 0x42, 0x03, 0xC4, 0x85, 0x06, 0x47,
                                          0x08, 0x49, 0xCA, 0x8B, 0x4C, 0x0D, 0x8E, 0xCF,
                                          0x50, 0x11, 0x92, 0xD3, 0x14, 0x55, 0xD6, 0x97,
                                          0xD8, 0x99, 0x1A, 0x5B, 0x9C, 0xDD, 0x5E, 0x1F,
                                          0x20, 0x61, 0xE2, 0xA3, 0x64, 0x25, 0xA6, 0xE7,
                                          0xA8, 0xE9, 0x6A, 0x2B, 0xEC, 0xAD, 0x2E, 0x6F,
                                          0xF0, 0xB1, 0x32, 0x73, 0xB4, 0xF5, 0x76, 0x37,
                                          0x78, 0x39, 0xBA, 0xFB, 0x3C, 0x7D, 0xFE, 0xBF };
const  unsigned short  wait_time[4]  =  { t_1_data, t_1_data, t_2_data, t_3_data };
const  unsigned short  t_13    =    t_13_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;
const     union {
```

```
            unsigned short  WORD;
            struct {
              unsigned char   smr;
              unsigned char   brr;
            }   BYTE;
  }   baudrate   =   baudrate_data;
```

2.6.4 Defining Internal Variables for the Library

Variables used in the library must be defined. Refer to Table 7.

Table 7 Internal Variables for the Library

Label name (variable name)	Data type	Description
ex_counter	unsigned long	Timer Z extended counter
flame_max	unsigned short	Response timeout setting (timer Z overflow count value)
t_counter	unsigned char	Transmission data counter
r_counter	unsigned char	Reception data counter
t_checksum	(Structure)	Transmission data checksum operation value
t_checksum.WORD	unsigned short	
t_checksum.BYTE.carry	unsigned char	
t_checksum.BYTE.data	unsigned char	
r_checksum	(Structure)	Reception data checksum operation value
r_checksum.WORD	unsigned short	
r_checksum.BYTE.carry	unsigned char	
r_checksum.BYTE.data	unsigned char	
in_status	(Structure)	Internal state of the library
in_status.BYTE	unsigned char	
in_status.BIT.sync_break	Bit 7	Sync break field transmission flag
in_status.BIT.sync_break_delimiter	Bit 6	Sync break delimiter transmission flag
in_status.BIT.sync_field	Bit 5	Sync field transmission flag
in_status.BIT.response_id	Bit 4	Response ID determination flag At response data transmission: 1 At reception: 0
in_status.BIT.wk3	Bits 3 to 2	Reserved bits
in_status.BIT.wu	Bits 1 to 0	Wake-up signal transmission flag (transmission counter for internal settings)

```
  static   union {
           unsigned long   LONG;
           struct {
             unsigned short   h;
             unsigned short   l;
           }  WORD;
  }   ex_counter;
  static   unsigned short   flame_max;
  static   unsigned char   t_counter;
```

```
static   unsigned char   r_counter;
static   union {
         unsigned short   WORD;
         struct {
           unsigned char   carry;
           unsigned char   data;
         }  BYTE;
}   t_checksum;
static   union {
         unsigned short   WORD;
         struct {
           unsigned char   carry;
           unsigned char   data;
         }  BYTE;
}   r_checksum;
static   union {
         unsigned char   BYTE;
         struct {
           unsigned char   sync_break            :1;
           unsigned char   sync_break_delimiter  :1;
           unsigned char   sync_field            :1;
           unsigned char   response_id           :1;
           unsigned char   dummy2                :2;
           unsigned char   wu                    :2;
         }  BIT;
}  In_status;
```

2.6.5 Defining Global Variables

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

(Refer to Table 4.)

```
volatile  unsigned char   LIN_tx_id;
volatile  unsigned char   LIN_tx_data[8];
volatile  unsigned char   LIN_rx_id;
volatile  unsigned char   LIN_rx_data[8];
volatile  union {
            unsigned char   BYTE;
            struct {
              unsigned char   wk7    :1;
              unsigned char   CSE    :1;
              unsigned char   wk5    :2;
              unsigned char   SNRE   :1;
              unsigned char   SCI    :1;
              unsigned char   SUC    :1;
              unsigned char   Ready  :1;
            }  BIT;
}  LIN_status;
volatile  union {
            unsigned char   BYTE;
            struct {
              unsigned char   SB_DEL  :2;
```

```
                unsigned char   WU      :1;
                unsigned char   wk4     :5;
              }  BIT;
  }  LIN_control;
```

2.6.6 Initialization Function

This function initializes the H8/36057 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), and P22 (TxD) are used for LIN communication. When the user application program uses other pins (P10 to P12, P15 to P17, P20, P23, and P24) in ports 1 and 2, the pin settings may be changed by the setting statement of PCR2 in the LIN_port_init function and the setting statement of 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, and then either delete the setting statements of PCR1 and PCR2 in the source file below or write them as comments.

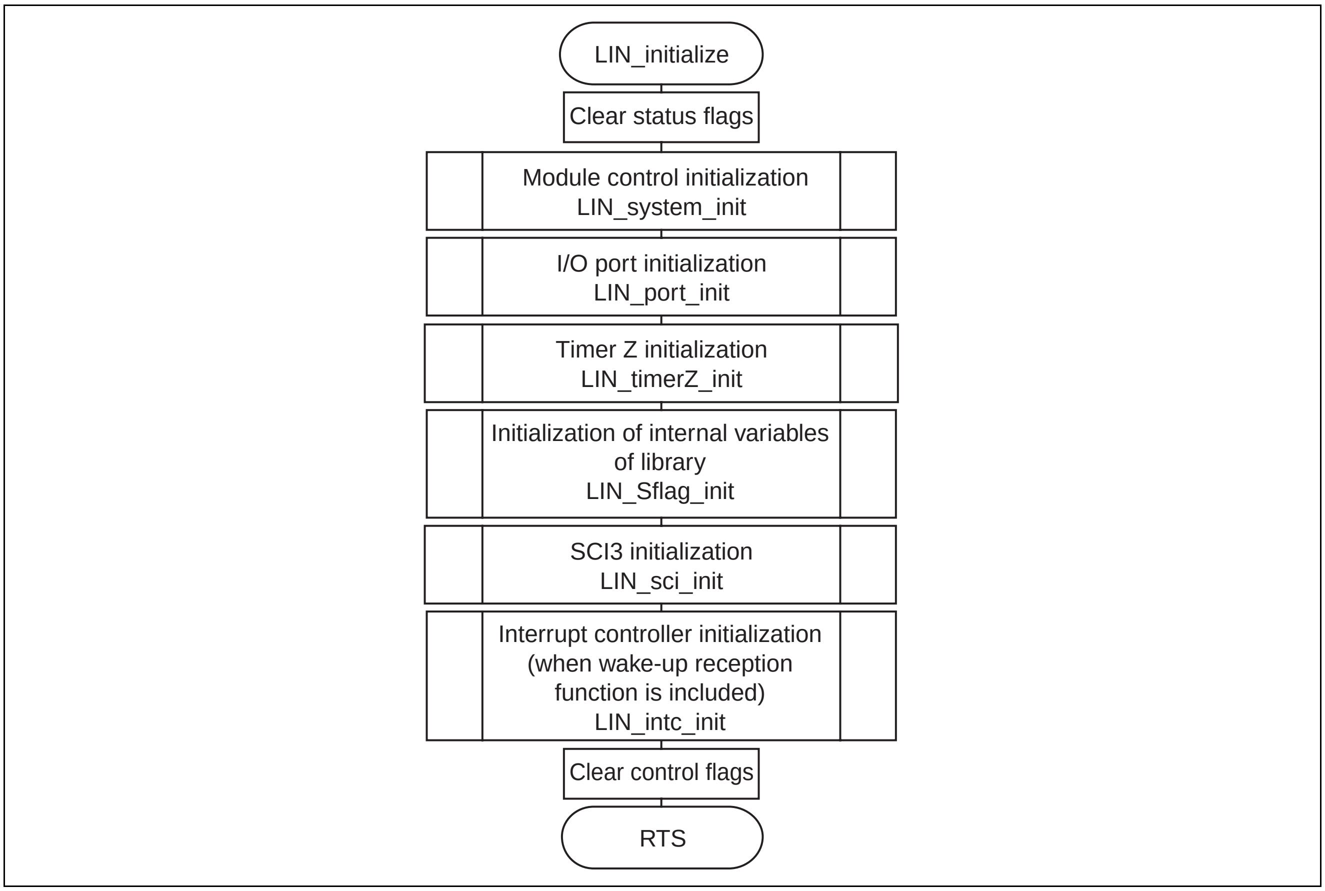


Figure 17 Initialization Function Flowchart

```
void   LIN_initialize(void) {
         LIN_status.BYTE   =   0x00;
         LIN_system_init();
         LIN_port_init();
         LIN_timerZ_init();
         LIN_Sflag_init();
         LIN_sci_init();

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

         LIN_control.BYTE   =   0x00;
}

void   LIN_system_init(void) {
         MSTCR1.BIT.MSTS3   =   0;
         MSTCR2.BIT.MSTTZ   =   0;
}

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   =   0x04;
}

void   LIN_sci_init(void) {
         SCI3.SCR3.BYTE   =   0x00;
         SCI3.SMR.BYTE   =   baudrate.BYTE.smr;
         SCI3.BRR   =   baudrate.BYTE.brr;
         TZ.GRC1   =   TZ.TCNT1   +   wait_time[1];
         TZ.TSR1.BIT.IMFC   =   0;
         TZ.TIER1.BIT.IMIEC   =   1;
         In_status.BIT.wu   +=   1;
}

void   LIN_timerZ_init(void) {
         TZ.TSTR.BIT.STR1   =   0;
         TZ.TCR1.BYTE   =   0x03;
         TZ.TIORC1.BIT.IOC2   =   0;
         TZ.TIORC1.BIT.IOC1   =   0;
         TZ.TIORC1.BIT.IOC0   =   0;
         TZ.GRC1   =   0x0000;
         TZ.TIER1.BYTE   &=   0xEB;
         TZ.TSTR.BIT.STR1   =   1;
}

#if   __R_WAKEUP   ==   __ON
void   LIN_intc_init(void) {
```

```
        IO.PCR1   =   0x00;
        IEGR1.BIT.IEG0   =   0;
        IRR1.BIT.IRRI0   =   0;
        IENR1.BIT.IEN0   =   0;
}
#endif

void   LIN_Sflag_init(void) {
        t_counter   =   0;
        r_counter   =   0;
        In_status.BYTE   =   0;
}
```

2.6.7 Header Frame Transmission Function

This function starts transmitting a header frame (the sync break field, sync field, and ID field).

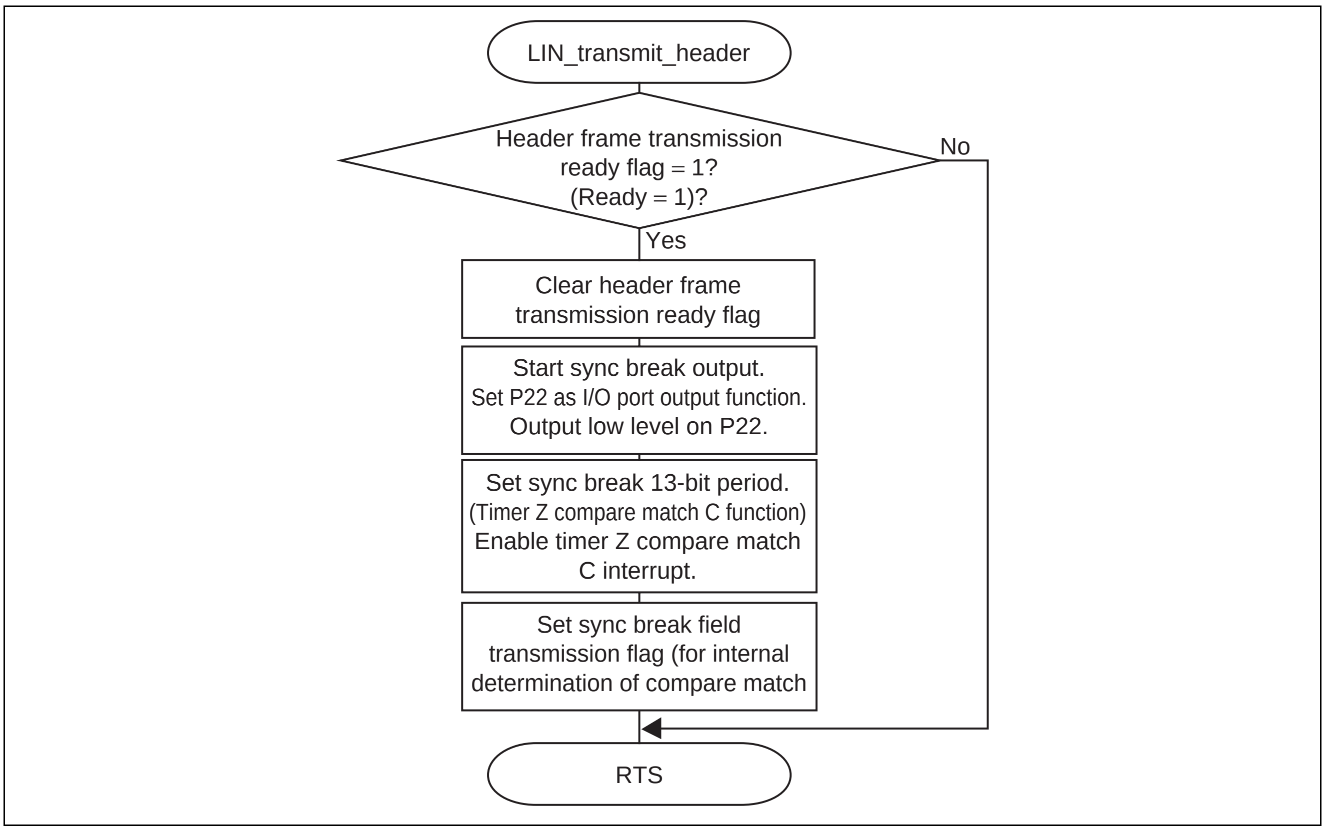


Figure 18 Flowchart of the Header Frame Transmission Function

```
void   LIN_transmit_header(void) {
        if(LIN_status.BIT.Ready) {
           LIN_status.BIT.Ready   =   0;
           IO.PMR1.BIT.TXD   =   0;
           IO.PDR2.BIT.B2   =   0;
           TZ.GRC1   =   TZ.TCNT1   +   t_13;
```

```
            TZ.TSR1.BIT.IMFC   =   0;
            TZ.TIER1.BIT.IMIEC   =   1;
            In_status.BYTE   =   0x80;
         }
}
```

2.6.8 Wake-Up Signal Transmission Function

This function transmits a wake-up signal.

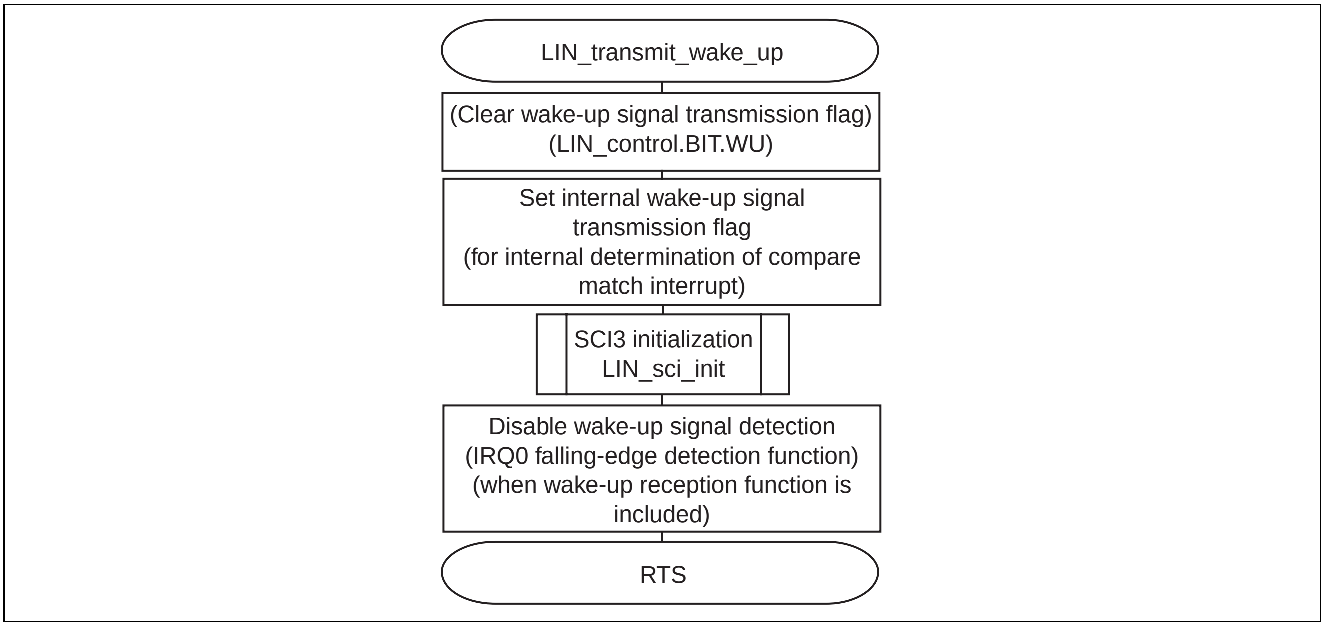


Figure 19 Flowchart of the Wake-up Signal Transmission Function

```
#if   __T_WAKEUP   ==   __ON
void   LIN_transmit_wake_up(void) {
         LIN_control.BIT.WU   =   0;
         In_status.BIT.wu   =   1;
         LIN_sci_init();

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

}
#endif
```

2.6.9 Function for Preparing for Wake-up Signal Reception

This function performs the necessary preparations prior to receiving a wake-up signal from another node (slave node).

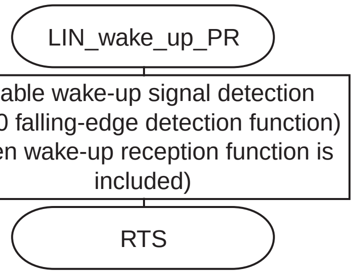


Figure 20 Flowchart of the Wake-Up Signal Reception Preparation Function

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

2.6.10 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 described in Section 2.7.9, this function detects a falling edge (such as a wake-up signal from another node) on the LIN bus and makes the necessary preparations for LIN communication control.

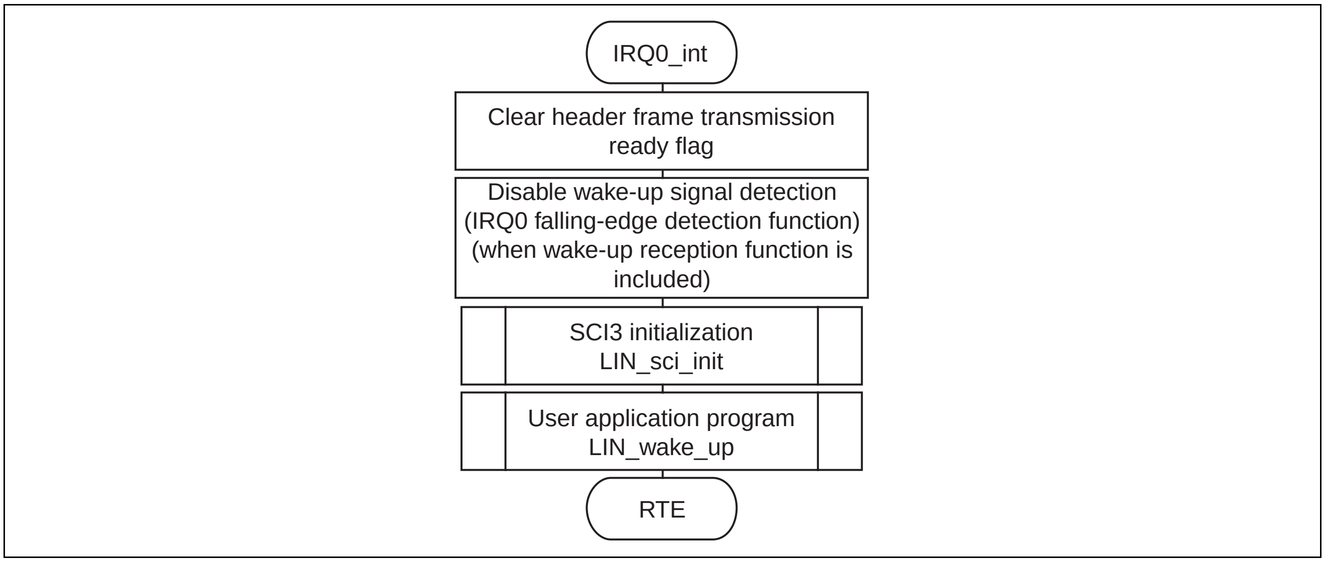


Figure 21 Flowchart of the IRQ Interrupt Function

```
#if   __R_WAKEUP   ==   __ON
#pragma   interrupt( IRQ0_int )
void   IRQ0_int(void) {
        LIN_status.BIT.Ready   =   0;
        IRR1.BIT.IRRI0   =   0;
        IENR1.BIT.IEN0   =   0;
        LIN_sci_init();
        LIN_wake_up();
}
#endif
```

2.6.11 Timer Z Interrupt Function

This function processes the timer Z (channel-1) overflow interrupt and compare match C interrupt.

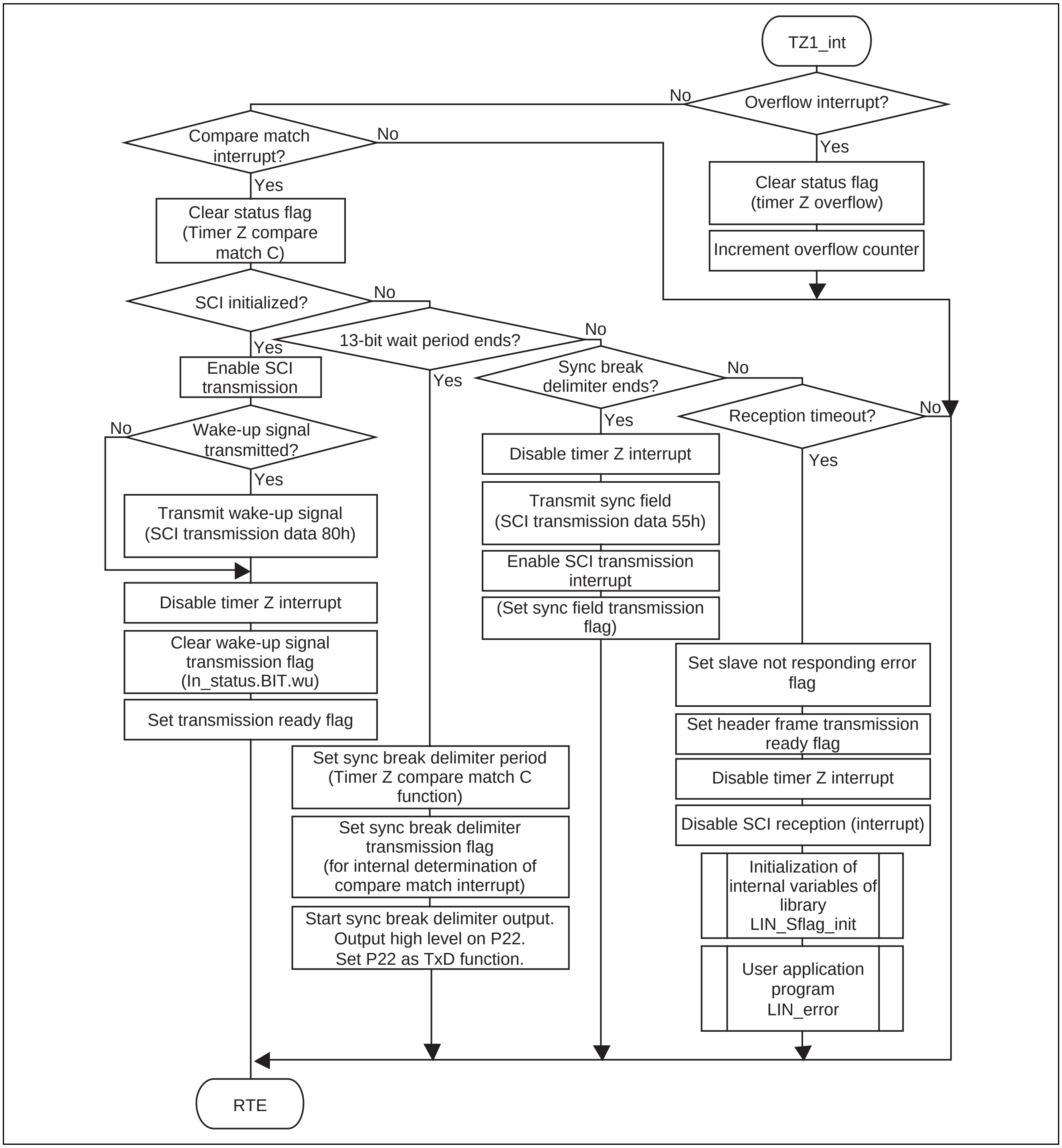


Figure 22 Flowchart of the Timer Z Interrupt Function

```
#pragma   interrupt( TZ1_int )
void  TZ1_int(void) {

        if((TZ.TSR1.BIT.OVF)  &&  (TZ.TIER1.BIT.OVIE)) {
            TZ.TSR1.BIT.OVF   =   0;
            ex_counter.WORD.h   +=   1;
         } else if((TZ.TSR1.BIT.IMFC)  &&  (TZ.TIER1.BIT.IMIEC)) {
           TZ.TSR1.BIT.IMFC   =   0;
           if(In_status.BIT.wu) {
              SCI3.SSR.BYTE   &=   0x80;
              SCI3.SCR3.BIT.TE   =   1;
              if(In_status.BIT.wu   ==   2) {
                 SCI3.TDR   =   0x80;
             }
             TZ.TIER1.BYTE   &=   0xEB;
             In_status.BIT.wu   =   0;
             LIN_status.BYTE   =   0x01;
           } else if(In_status.BIT.sync_break) {
             TZ.GRC1   =   TZ.TCNT1   +   wait_time[LIN_control.BIT.SB_DEL];
             In_status.BYTE   =   0x40;
             IO.PDR2.BIT.B2   =   1;
             IO.PMR1.BIT.TXD   =   1;
           } else if(In_status.BIT.sync_break_delimiter) {
             TZ.TIER1.BYTE   &=   0xEB;
             SCI3.SSR.BYTE   &=   0x80;
             SCI3.TDR   =   0x55;
             SCI3.SCR3.BIT.TIE   =   1;
             In_status.BYTE   =   0x20;
           } else if(r_counter) {
             if(ex_counter.WORD.h   >=   flame_max) {
                LIN_status.BYTE   =   0x09;
                TZ.TIER1.BYTE   &=   0xEB;
                SCI3.SCR3.BYTE   =   0x20;
                LIN_Sflag_init();
                LIN_error();
             }
           }
         }
}
```

2.6.12 SCI3 Interrupt Function

This function processes the SCI3 (channel-0) error detection interrupt, reception interrupt, transmission interrupt, and transmission-end interrupt.

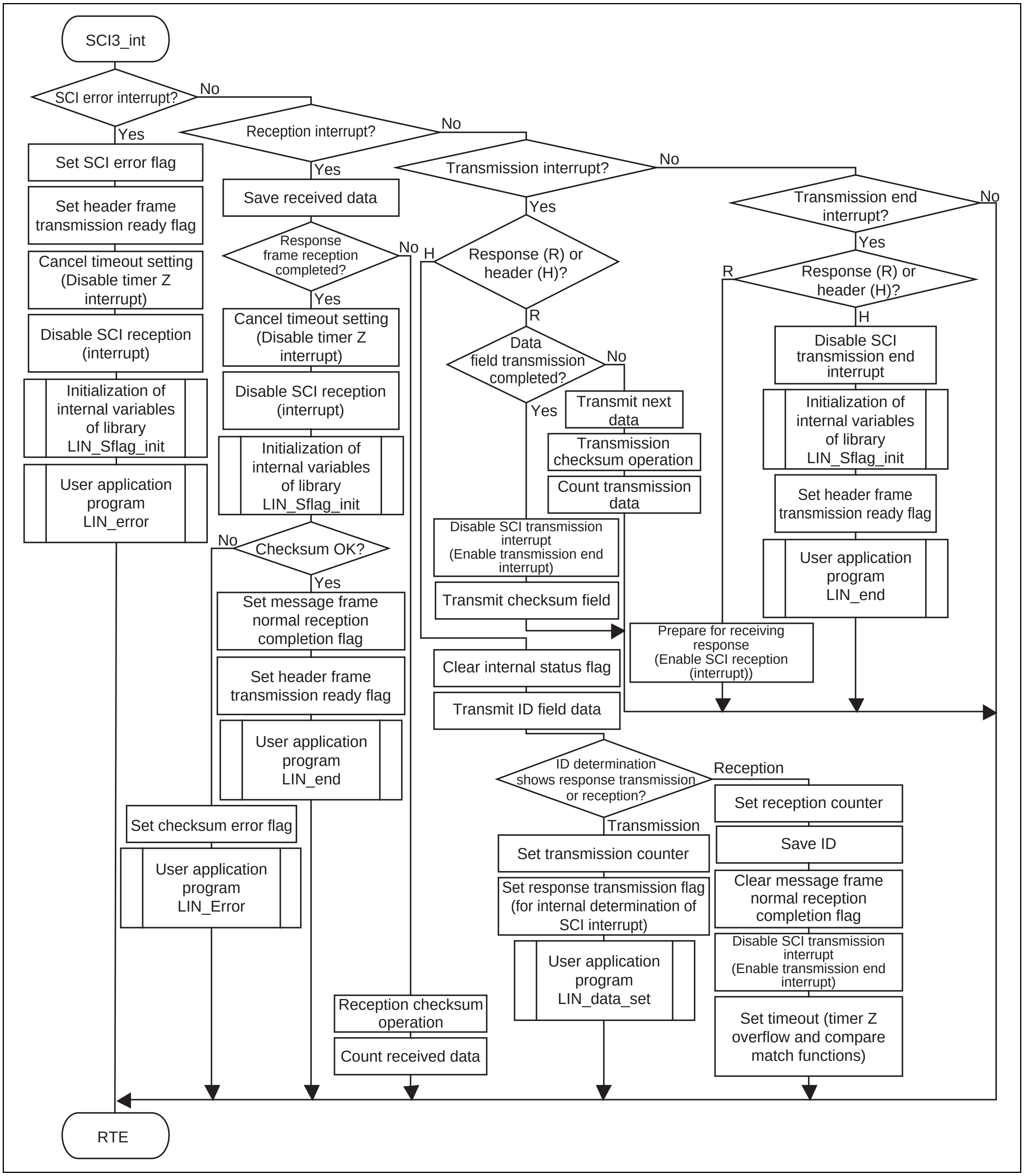


Figure 23 Flowchart of the SCI3 Interrupt Function

```
#pragma   interrupt( SCI3_int )
void   SCI3_int(void) {
         unsigned char   buff, nmbr, nm, dlc;

         if(SCI3.SSR.BYTE   &   0x38) {
            LIN_status.BYTE   =   0x05;
            TZ.TIER1.BYTE   &=   0xEB;
            SCI3.SCR3.BYTE   =   0x20;
            LIN_Sflag_init();
            LIN_error();
          } else if(SCI3.SSR.BIT.RDRF) {
          buff   =   SCI3.RDR;
          nm   =   r_counter   &   0x0F;
          nmbr   =   (r_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;
            r_counter   -=   1;
          } else {
            TZ.TIER1.BYTE   &=   0xEB;
            SCI3.SCR3.BYTE   =   0x20;
            LIN_Sflag_init();
            if((r_checksum.BYTE.data   ^   buff)   !=   0xFF) {
                LIN_status.BYTE   =   0x41;
                LIN_error();
            } else {
              LIN_status.BYTE   =   0x03;
              LIN_end();
            }
          }
        } else if((SCI3.SSR.BIT.TDRE)  &&  (SCI3.SCR3.BIT.TIE)) {
          if(In_status.BIT.response_id) {
             nm   =   t_counter   &   0x0F;
             nmbr   =   (t_counter   >>  4)   -   nm;
             if(nm) {
                buff   =   LIN_tx_data[(nmbr)];
                SCI3.TDR   =   buff;
                t_checksum.WORD   +=   buff;
                t_checksum.BYTE.data   +=   t_checksum.BYTE.carry;
                t_checksum.BYTE.carry   =   0;
                t_counter   -=   1;
              } else {
                SCI3.SSR.BYTE   &=   0x80;
                SCI3.SCR3.BYTE   =   0x24;
                t_checksum.BYTE.data   =   ~(t_checksum.BYTE.data);
                SCI3.TDR   =   t_checksum.BYTE.data;
                t_counter   =   0;
            }
          } else {
            In_status.BYTE   =   0x00;
            buff   =   id_field[LIN_tx_id];
```

```
            SCI3.TDR   =   buff;
            switch(buff) {

#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

#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

                t_counter   =   0x22;
                In_status.BIT.response_id   =   1;
                t_checksum.WORD   =   0;
                LIN_data_set();
                break;
#endif

#if __Res4byte_ID == __ON
#ifdef __ID20
              case __ID20:
#endif

#ifdef __ID21
              case __ID21:
#endif

#ifdef __ID22
              case __ID22:
#endif

#ifdef __ID23
              case __ID23:
#endif
```

```
#ifdef __ID24
              case __ID24:
#endif

#ifdef __ID25
              case __ID25:
#endif

#ifdef __ID26
              case __ID26:
#endif

#ifdef __ID27
              case __ID27:
#endif

#ifdef __ID28
              case __ID28:
#endif

#ifdef __ID29
              case __ID29:
#endif

#ifdef __ID2a
              case __ID2a:
#endif

#ifdef __ID2b
              case __ID2b:
#endif

#ifdef __ID2c
              case __ID2c:
#endif

#ifdef __ID2d
              case __ID2d:
#endif

#ifdef __ID2e
              case __ID2e:
#endif

#ifdef __ID2f
              case __ID2f:
#endif

                t_counter   =   0x44;
                In_status.BIT.response_id   =   1;
                t_checksum.WORD   =   0;
                LIN_data_set();
                break;
```

```
#endif

#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

                t_counter   =   0x88;
```

```
                In_status.BIT.response_id   =   1;
                t_checksum.WORD   =   0;
                LIN_data_set();
                break;
#endif

              case 0x3C:
                t_counter   =   0x88;
                In_status.BIT.response_id   =   1;
                t_checksum.WORD   =   0;
                LIN_data_set();
                break;
              case 0x7D:
              case 0xFE:
              case 0xBF:
                r_counter   =   0x88;
                r_checksum.WORD   =   0;
                LIN_status.BIT.SUC   =   0;
                LIN_rx_id   =   buff;
                SCI3.SSR.BYTE   &=   0x80;
                SCI3.SCR3.BYTE   =   0x24;
                TZ.GRC1   =   flame_max_8.WORD.l;
                flame_max   =   flame_max_8.WORD.h;
                ex_counter.WORD.h   =   0;
                TZ.TSR1.BYTE   &=   0xEB;
                TZ.TIER1.BYTE   |=   0x14;
                break;
              default :
                dlc   =   buff   &   0x30;
                if(dlc   ==   0x20) {
                   r_counter   =   0x44;
                   TZ.GRC1   =   flame_max_4.WORD.l;
                   flame_max   =   flame_max_4.WORD.h;
                 } else if(dlc   ==   0x30) {
                   r_counter   =   0x88;
                   TZ.GRC1   =   flame_max_8.WORD.l;
                   flame_max   =   flame_max_8.WORD.h;
                 } else {
                   r_counter   =   0x22;
                   TZ.GRC1   =   flame_max_2.WORD.l;
                   flame_max   =   flame_max_2.WORD.h;
                 }
                 r_checksum.WORD   =   0;
                 LIN_status.BIT.SUC   =   0;
                 LIN_rx_id   =   buff;
                 SCI3.SSR.BYTE   &=   0x80;
                 SCI3.SCR3.BYTE   =   0x24;
                 ex_counter.WORD.h   =   0;
                 TZ.TSR1.BYTE   &=   0xEB;
                 TZ.TIER1.BYTE   |=   0x14;
                 break;
            }
          }
```

```
        } else if((SCI3.SSR.BIT.TEND)  &&  (SCI3.SCR3.BIT.TEIE)) {
          if(In_status.BIT.response_id) {
             SCI3.SCR3.BYTE   =   0x20;
             LIN_Sflag_init();
             LIN_status.BIT.Ready   =   1;
             LIN_end();
           } else {
             SCI3.SSR.BYTE   &=   0x80;
             SCI3.SCR3.BYTE   =   0x70;
          }
        }
}
```

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Dec.20.03	—	First edition issued
1.01	Jun.15.07	Pages 1, 6, 7, 14, 49 and 50	Content correction

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.