お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

H8/300H Tiny シリーズ

LIN (Local Interconnect Network) マスタ編

要旨

LIN (Local Interconnect Network) アプリケーションノート H8/300H Tiny シリーズ マスタ編は，H8/36057F グループマイコンの内蔵周辺機能を使用し，LIN 通信プロトコルによる通信を行なう仕様例，設定例を記しており，ユーザにてソフトウェア設計およびハードウェア設計の際，ご参考として役立てていただけるようにまとめたものです。

動作確認デバイス

H8/36057F

目次

1. LIN 通信システム概要

本アプリケーションノートに掲載している LIN 通信用ソフトウェアライブラリサンプル (以下ライブラリと称す) を組み込んだシステムによる LIN 通信の概要を示します。

1.1 LIN バスへの接続

LIN バスによりネットワーク接続されたシステム (図 1) に LIN バスインタフェース回路 (または LIN トランシーバ) を介して接続することにより，マスタノードとしてヘッダフレームの送信，レスポンスフレームの送受信等の LIN 通信を行ないます。

1.1.1 システム構成

図 1 に LIN バスネットワークシステム構成例を示します。

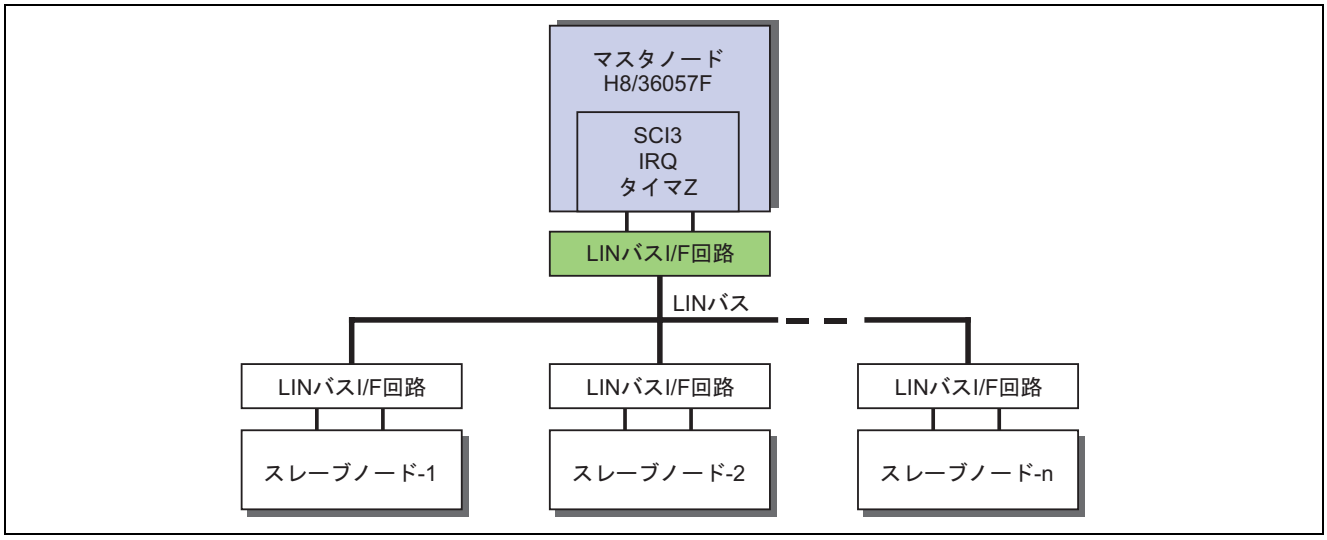

図 1 LIN バス接続によるシステムブロック図

1.1.2 LIN バス (シングルワイヤバス) インタフェース

図 2 にマイコン (H8/36057F) 内蔵機能の入出力端子と LIN バスとのインタフェース回路例を示します。

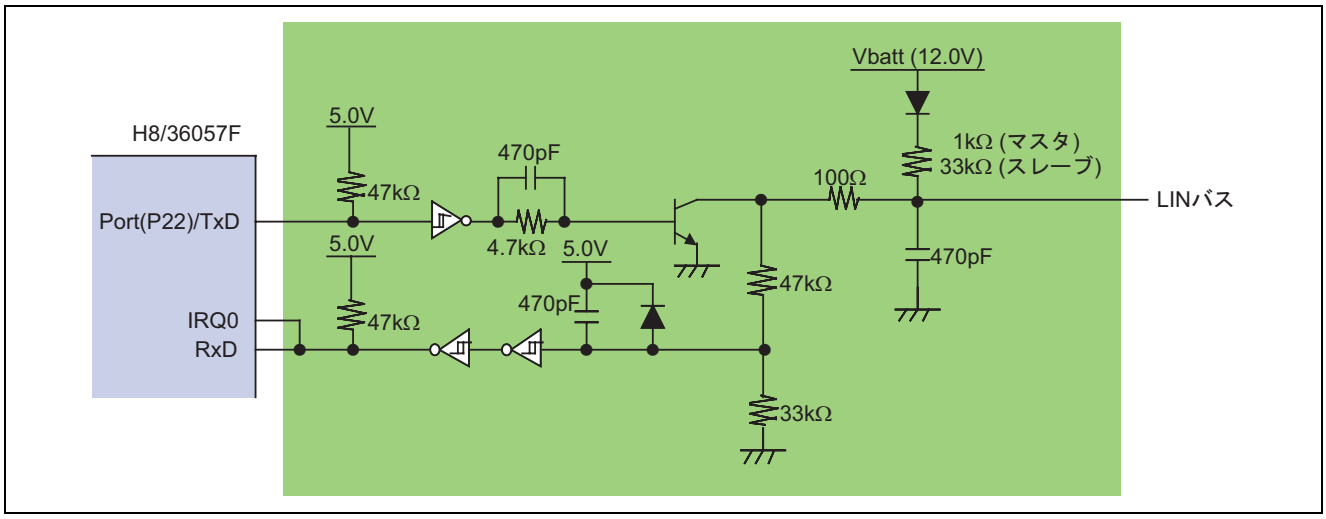

図 2 LIN バスインタフェース回路例

1.2 LIN 通信概要

LIN 通信プロトコルにより送受信されるメッセージフレームの概要を示します。

1.2.1 メッセージフレーム構成

図 3 にメッセージフレームの構成を示します。

メッセージフレームは，マスタノードから送信されるヘッダフレームと，マスタノードもしくはスレーブノードから送信されるレスポンスフレームから構成されます。

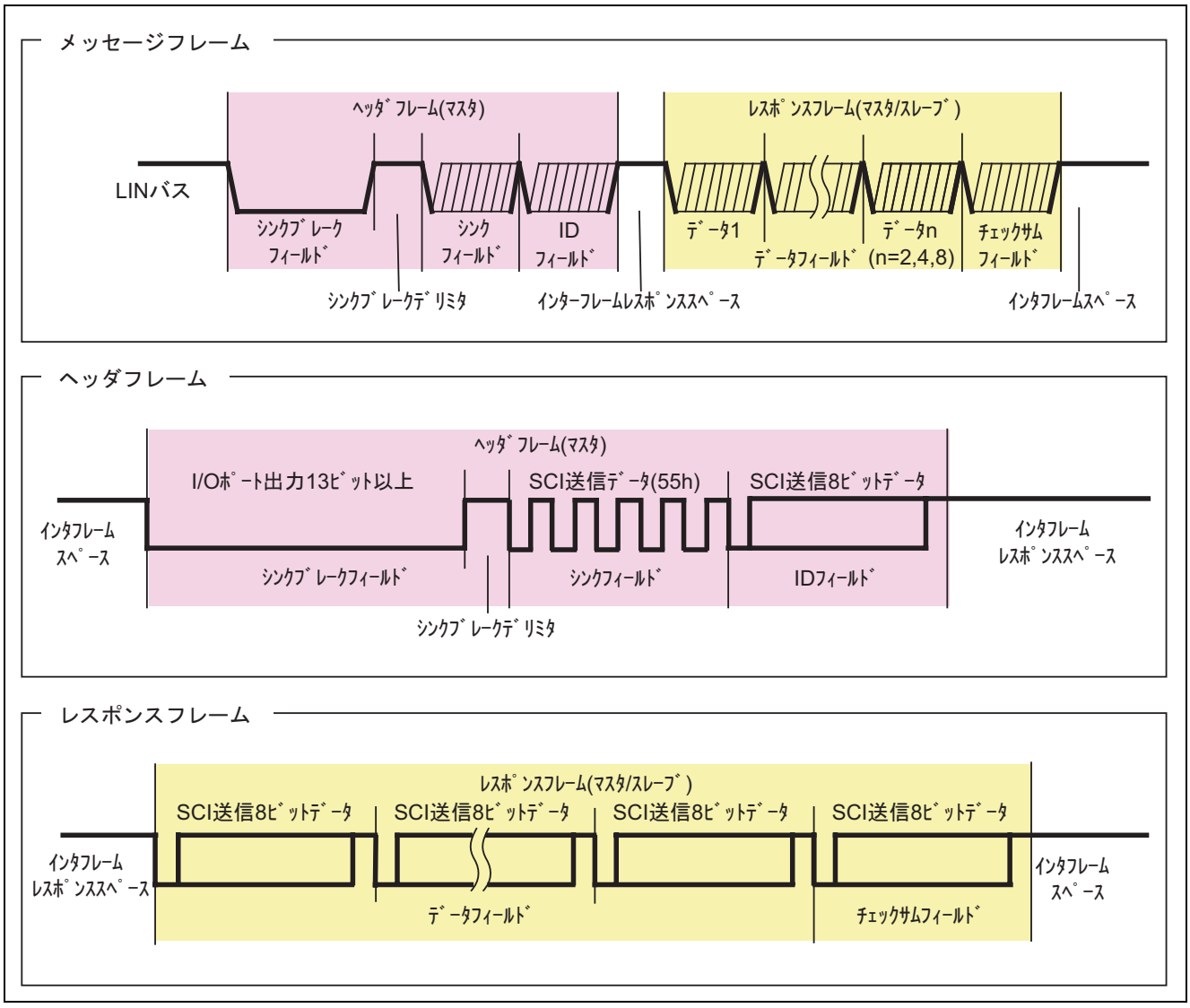

図 3　メッセージフレームの構成

1.2.2 メッセージフレームの送受信

図 4 にマスタ／スレーブ各ノードにおけるメッセージフレームの送受信イメージを示します。

— マスタノードがヘッダフレームを送信します。
— スレーブノードは，受信したヘッダフレームから ID を判定し，自ノードに対する ID の時レスポンスを送信します。(マスタノードは送信時に ID を判定)

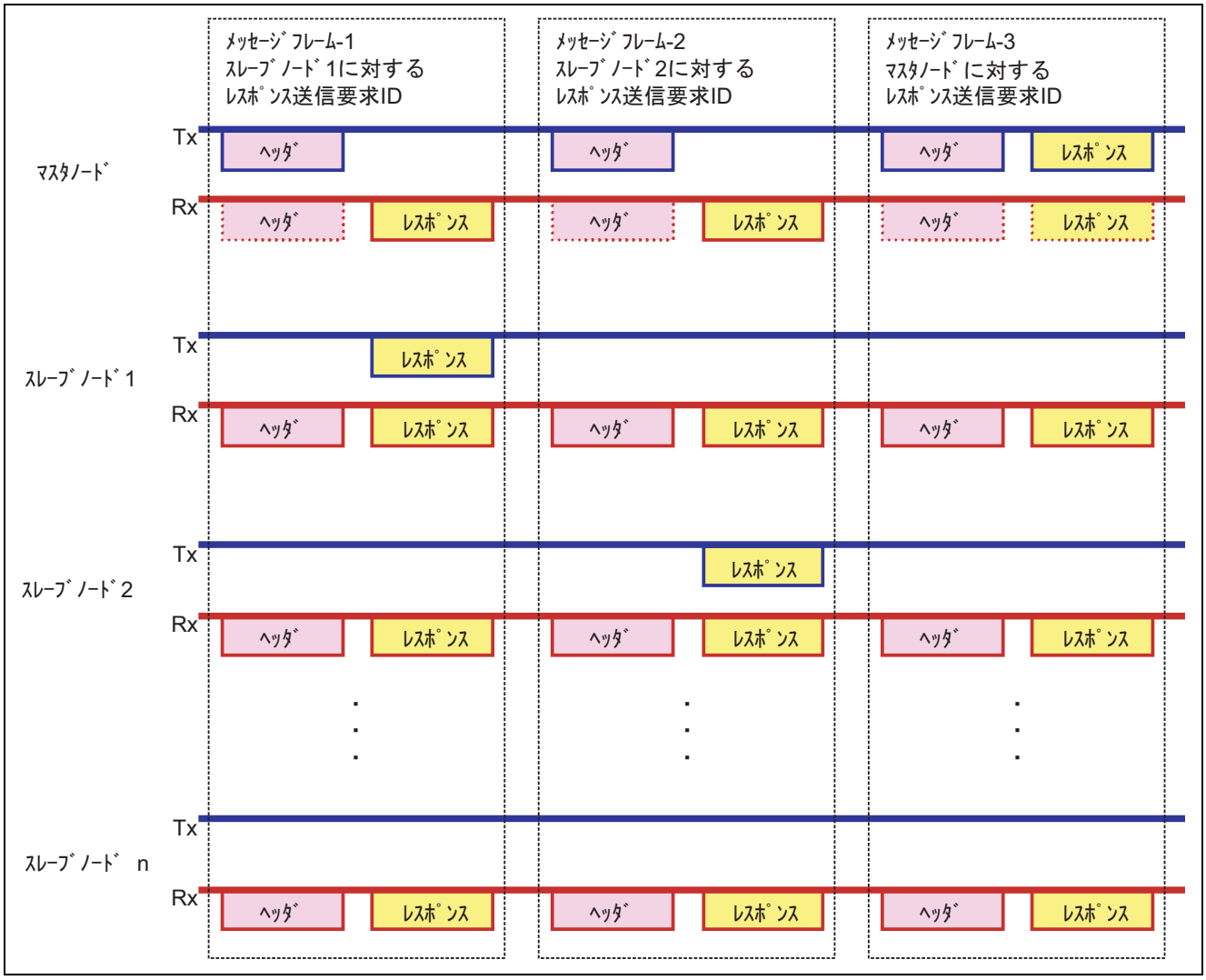

図 4　メッセージフレームの送受信イメージ

2. ライブラリソフトウェア仕様

ライブラリをユーザアプリケーションプログラムに組み込む事により H8/36057F の内蔵機能を使用し，マスタノードとして LIN 通信を行ないます。

2.1 動作環境

- 使用デバイス：H8/36037/57 グループマイコン
- 動作周波数範囲 (システム搭載クロック (φosc))：デバイス動作周波数と同等範囲
  ただし，LIN 通信速度およびユーザアプリケーションプログラムの処理条件を考慮し，LINID.h 内にφosc を定義する必要があります。(詳細は後述)
- 使用機能：ライブラリにおいて使用するマイコンの内蔵周辺機能と使用用途を表 1 に示します

表 1　H8/36057 内蔵周辺機能使用表

機能		端子機能 (端子 No.)	用途	説明
I/O ポート		P22 (46pin)	シンクブレークフィールド送信	I/O ポート (High/Low) 出力によりシンクブレークフィールドを形成
SCI3 (channel-0)	送信	TXD (46pin)	シンクフィールド送信 ID フィールド送信 レスポンスフレーム送信 ウェイクアップ信号送信	調歩同期式モード データ長 8 ビット パリティービット無し 1 ストップビット (スタートビット付加) LSB ファースト
	受信	RXD (45pin)	レスポンスフレーム受信	
			通信エラー検出	モジュール内エラー検出機能
タイマ Z (channel-1)		―	シンクブレークフィールドドミナント期間計測 シンクブレークデリミタ期間計測 ウェイト期間計測（ライブラリ内部機能） タイムアウト検出	φosc/8 周期でカウント動作し，各時間間隔を計測
IRQ		/IRQ0 (51pin)	ウェイクアップ信号検出	スタンバイ時に LIN バスを監視し立下りエッジを検出

2.2 ファイル構成

- LINmtrT.c (Ver.1.00)
  H8/36057 グループにおける LIN 通信用 (マスタ) マイコン機能設定，および通信制御を行なうための C ソースファイルです。
- LINID.h (Ver.1.00)
  ユーザにより通信転送速度の設定や ID の設定等を行ない LINmtrT.c（Ver.1.00）コンパイル時に組込むためのインクルードファイルです。また, ユーザアプリケーションインタフェース用関数, 変数定義も同ファイル内で行っているため，ユーザアプリケーションプログラムコンパイル時にも組込む必要があります。
- H8_36057.h (Ver.1.00)
  H8/36057 グループ用内蔵 I/O レジスタ定義ファイルです。

2.3 ROM/RAM 使用容量

(H8S, H8/300 シリーズ C コンパイラ CH38.exe Ver.2.0C 使用時)

ROM/RAM 使用容量は，ID 設定数等により変化します。

- ROM 容量：約 1.5 kByte
- RAM 容量：約 35 Byte

2.4 機能仕様

2.4.1 LIN 通信仕様

- ノード：マスタノードに対応
- ID：ユーザ定義 ID

  レスポンス送信 ID

  LINID.h により 0 個から 60 個 (00h～3bh) まで設定可能。
  (ただし，同一 LIN バス上に同じ ID を持つノードを設定すると正常動作しません)

  LIN プロトコル定義 ID

  a) マスタリクエストフレーム ID 3ch (ID フィールドデータ：3Ch)
  自動的にマスタがレスポンスフレーム (データ長 8 バイト) を送信します。
  データフィールドの設定はユーザによる設定とします。データフィールド 1 バイト目が 00h の時，スリープコマンドとなります。

  b) スレーブレスポンスフレーム ID 3dh (ID フィールドデータ：7Dh)
  レスポンスフレーム (データ長 8 バイト) を受信します。

  c) 拡張フレーム ID 3eh, 3fh (ID フィールドデータ：FEh, BFh)
  本ライブラリ (Ver.1.00) では，対応しておりません。

  ID 設定方法

  LINID.h 内でレスポンス送信 ID として設定する ID 以外の定義文 (#define __IDm 0xnn (m=00h～3bh)) を削除，またはコメント文とし設定したい ID のみを定義した状態で LINmtrT.c をコンパイルします。

- レスポンスデータ長：送信する ID (LIN_tx_id) フィールドデータ内 DLC (データレングスコントロール) ビットにより指定。

  LIN_tx_id = 00h～1fh ： 2 バイト
  = 20h～2fh ： 4 バイト
  = 30h～3Dh ： 8 バイト

- 通信転送速度：LINID.h 内で使用する通信転送速度を定義します。
  システム搭載クロック (φosc) 定義値，および通信転送速度定義値からライブラリ内で使用する定数，および SCI3 モジュール設定値を自動的に算出します。(注意：通信転送速度はφosc により制限されることがあります。詳しくはハードウェアマニュアルの SCI3 モジュール：ビットレートに対する BRR 設定例 (調歩同期式モード) をご参考ください)
- ウェイクアップ信号送受信：ウェイクアップ信号の送信機能，受信機能を組込みます。

  ウェイクアップ信号送信機能の組込み

  LINID.h 内定義文 (#define __T_WAKEUP __ON) により，ウェイクアップ送信機能を組込み，ユーザアプリケーションプログラム等から関数 (LIN_transmit_wake_up) を呼び出すことにより LIN バス上にウェイクアップ信号を送信します。

  ウェイクアップ信号受信機能の組込み

  LINID.h 内定義文 (#define __R_WAKEUP __ON) により，ウェイクアップ受信機能を組込み，マイコン (H8/36057) がスタンバイ状態等にあっても LIN バス上のウェイクアップ信号を IRQ0 (外部割込み入力) により検出 (立下りエッジ検出) します。

2.4.2 LINID.h ファイル設定例

LINID.h の設定例を示します。

a) ウェイクアップ信号の送信は行なわない。
b) IRQ0 (外部割込み) によるウェイクアップ信号の検出 (立下りエッジ検出) を行なう。
c) 以下の ID に対してレスポンスデータを送信する。

ID (ID ビット+DLC ビット)	(パリティビット含む)
01h	(C1h)
12h	(92h)
23h	(A3h)
34h	(B4h)

d) システム搭載クロック (φosc) を 20[MHz]とする。
e) LIN 通信転送速度を 9600[bit/s]とする。

上記 a)～e)に基づき設定した例を以下に示します。

(太字以外の定義文を削除もしくはコメント行としてください。)

```
/************************************************************************/
/*                      */
/*                   LINID.h  Ver. 1.00*/
/*                      */
/************************************************************************/
#define    __ON   1       /* 変更または削除しないでください    */
#define    __OFF  0       /* 変更または削除しないでください    */


/************************************************************************/
/*    ウェイクアップ送信機能設定    */
/************************************************************************/
/*#define  __T_WAKEUP  __ON /* ウェイクアップ信号の送信を行なう場合はこの行を定義    */
#define    __T_WAKEUP  __OFF/* ウェイクアップ信号の送信を行なわない場合はこの行を定義    */


/************************************************************************/
/*    ウェイクアップ受信機能設定    */
/************************************************************************/
#define    __R_WAKEUP  __ON /* ウェイクアップ信号の検出(立下りエッジ検出)を行なう場合はこの行を定義   */
/*#define  __R_WAKEUP  __OFF/* ウェイクアップ信号の検出を行なわない場合はこの行を定義  */

/************************************************************************/
/*    レスポンス送信 ID 設定 */
/************************************************************************/
/*    2 バイトデータ       */
/*----------------------------------------------------------------------*/

#define    __Res2byte_ID  __ON /* 2 バイトのレスポンスデータ送信 ID を持つ場合はこの行を定義    */
/*#define  __Res2byte_ID  __OFF/* 2 バイトのレスポンスデータ送信 ID を持たない場合はこの行を定義    */

#if   __Res2byte_ID   ==  __ON

/*#define__ID000x80  /* */
#define      __ID010xC1  /* ID フィールド C1h に対してレスポンスを送信   */
/*#define__ID020x42  /* */
/*#define__ID030x03  /* */
```

```
/*#define__ID040xC4  /* */
/*#define__ID050x85  /* */
/*#define__ID060x06  /* */
/*#define__ID070x47  /* */
/*#define__ID080x08  /* */
/*#define   __ID090x49  /* */
/*#define__ID0a0xCA  /* */
/*#define__ID0b0x8B  /* */
/*#define__ID0c0x4C  /* */
/*#define__ID0d0x0D  /* */
/*#define__ID0e0x8E  /* */
/*#define__ID0f0xCF  /* */
/*#define__ID100x50  /* */
/*#define__ID110x11  /* */
#define     __ID12 0x92  /* ID フィールド 92h に対してレスポンスを送信  */
/*#define__ID130xD3  /* */
/*#define__ID140x14  /* */
/*#define__ID150x55  /* */
/*#define__ID160xD6  /* */
/*#define__ID170x97  /* */
/*#define__ID180xD8  /* */
/*#define__ID190x99  /* */
/*#define__ID1a0x1A  /* */
/*#define__ID1b0x5B  /* */
/*#define__ID1c0x9C  /* */
/*#define__ID1d0xDD  /* */
/*#define__ID1e0x5E  /* */
/*#define__ID1f0x1F  /* */

#endif


/*-------------------------------------------------------------------------------*/
/*    4 バイトデータ       */
/*-------------------------------------------------------------------------------*/
#define    __Res4byte_ID  __ON /* 4 バイトのレスポンスデータ送信 ID を持つ場合はこの行を定義    */
/*#define  __Res4byte_ID  __OFF  /* 2 バイトのレスポンスデータ送信 ID を持たない場合はこの行を定義  */

#if   __Res4byte_ID   ==   __ON

/*#define__ID200x20  /* */
/*#define__ID210x61  /* */
/*#define__ID220xE2  /* */
#define     __ID230xA3  /* ID フィールド A3h に対してレスポンスを送信  */
/*#define__ID240x64  /* */
/*#define__ID250x25  /* */
/*#define__ID260xA6  /* */
/*#define__ID270xE7  /* */
/*#define__ID280xA8  /* */
/*#define__ID290xE9  /* */
/*#define__ID2a0x6A  /* */
/*#define__ID2b0x2B  /* */
/*#define__ID2c0xEC  /* */
/*#define__ID2d0xAD  /* */
/*#define__ID2e0x2E  /* */
/*#define__ID2f0x6F  /* */
```

```
#endif


/*-------------------------------------------------------------------------*/
/*    8 バイトデータ       */
/*-------------------------------------------------------------------------*/
#define    __Res8byte_ID  __ON /* 8 バイトのレスポンスデータ送信 ID を持つ場合はこの行を定義    */
/*#define  __Res8byte_ID  __OFF  /* 8 バイトのレスポンスデータ送信 ID を持たない場合はこの行を定義   */

#if   __Res8byte_ID   ==  __ON

/*#define__ID300xF0  /* */
/*#define__ID310xB1  /* */
/*#define__ID320x32  /* */
/*#define__ID330x73  /* */
#define     __ID340xB4  /* ID フィールド B4h に対してレスポンスを送信   */
/*#define__ID350xF5  /* */
/*#define__ID36   0x76  /* */
/*#define__ID370x37  /* */
/*#define__ID380x78  /* */
/*#define__ID39   0x39  /* */
/*#define__ID3a0xBA  /* */
/*#define__ID3b0xFB  /* */

#endif


/*************************************************************************/
/*    システム搭載クロック（φosc）定義部  */
/*************************************************************************/
#define    OSC_Hz   20000000   /* φosc=20.000MHz →20000000   */
/*#defineOSC_Hz16000000 /* φosc=16.000MHz →16000000   */
/*#defineOSC_Hz10486000 /* φosc=10.486MHz →10486000   */
/*#defineOSC_Hz10000000 /* φosc=10.000MHz →10000000   */
/*#defineOSC_Hz 9830400 /* φosc=9.8304MHz →  9830400   */
/*#defineOSC_Hz 8000000 /* φosc=8.0000MHz →  8000000   */
/*#defineOSC_Hz 7372800 /* φosc=7.3728MHz →  7372800   */
/*#defineOSC_Hz 4915200 /* φosc=4.9152MHz →  4915200   */
/*#defineOSC_Hz 2457600 /* φosc=2.4576MHz →  2457600   */


/*************************************************************************/
/*    ボーレート定義部     */
/*************************************************************************/
/*#defineB_rate 2400 /*  2400bps →  2400 */
/*#defineB_rate 4800 /*  4800bps →  4800 */
#define    B_rate   9600   /* 9600 bps → 9600*/
/*#defineB_rate10000 /* 10000bps → 10000 */
/*#defineB_rate14400 /* 14400bps → 14400 */
/*#defineB_rate19200 /* 19200bps → 19200 */
/*#defineB_rate20000 /* 20000bps  →20000 */


/*************************************************************************/
/*    ライブラリ用定数算出部    以下は変更または削除しないでください */
```

```
/*****************************************************************************/
#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)


/*****************************************************************************/
/*     関数・変数定義部      以下は変更または削除しないでください */
/*****************************************************************************/
#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 ユーザアプリケーションインタフェース

本ライブラリとユーザアプリケーションプログラムとのインタフェース仕様を示します。

- 関数 (モジュール) 呼び出しによるインタフェース
  ユーザアプリケーションプログラムからライブラリ内の関数を呼び出すことにより, LIN 通信制御に必要なマイコン (H8/36057) 内蔵周辺機能の初期化, ヘッダフレームの送信, ウェイクアップ信号の送信制御, ウェイクアップ信号の受信準備を行ないます。

表 2　ユーザアプリケーションプログラムからのライブラリ内関数呼び出し

関数名	機能説明
LIN_initialize	LIN 通信制御に必要なマイコン (H8/36057) 内蔵周辺機能の初期設定を行ないます。
LIN_transmit_header	ヘッダフレームの送信を開始します。
LIN_transmit_wake_up	ウェイクアップ信号を送信します。
LIN_wake_up_PR	ウェイクアップ信号の受信準備を行ないます。

ライブラリ内から呼び出される関数をユーザアプリケーションプログラム内に準備することにより，LIN 通信中の各タイミング (送受信完了時，通信エラー検出時等) での処理を行ないます。

表 3　ライブラリからのユーザアプリケーション制御用関数呼び出し

関数名	機能説明
LIN_wake_up	ウェイクアップ信号検出時のユーザアプリケーション制御用関数
LIN_data_set	レスポンスフレーム送信前のユーザアプリケーション制御用関数
LIN_end	メッセージフレーム送受信完了後のユーザアプリケーション制御用関数
LIN_error	LIN 通信エラー検出時のユーザアプリケーション制御用関数

- 動作概要
  図 5，図 6 に動作概要を示します。

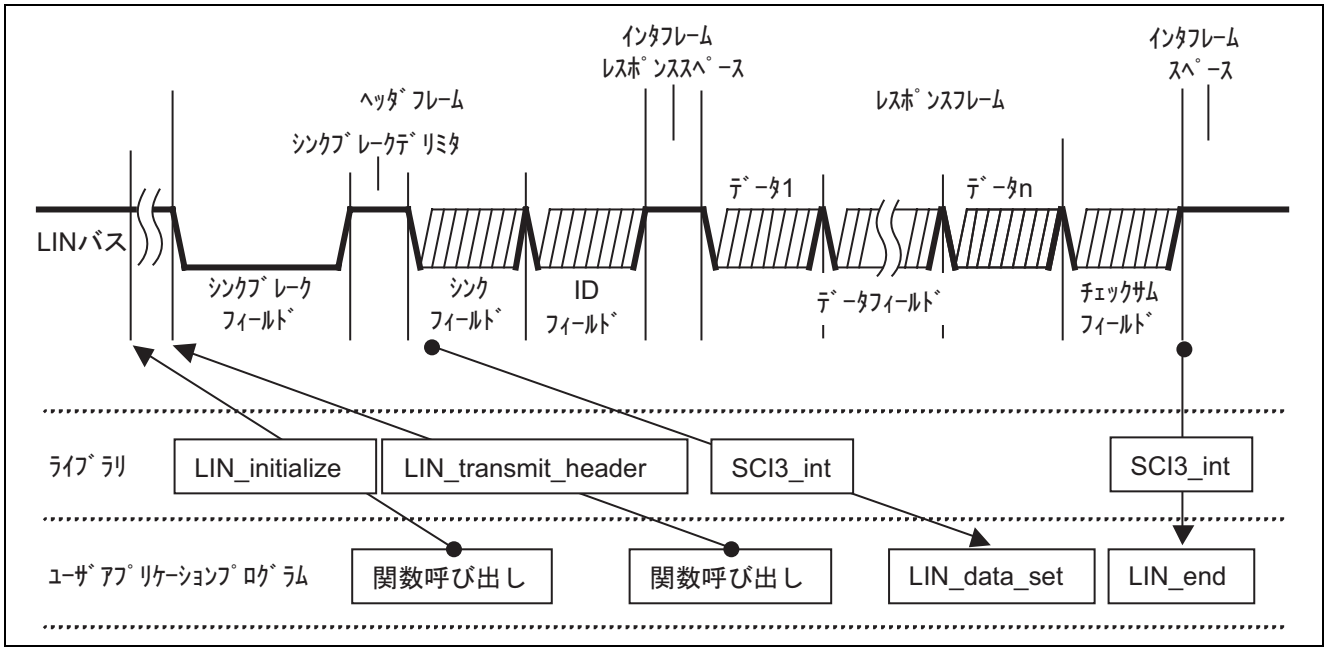

図 5　メッセージフレーム送受信時動作概要

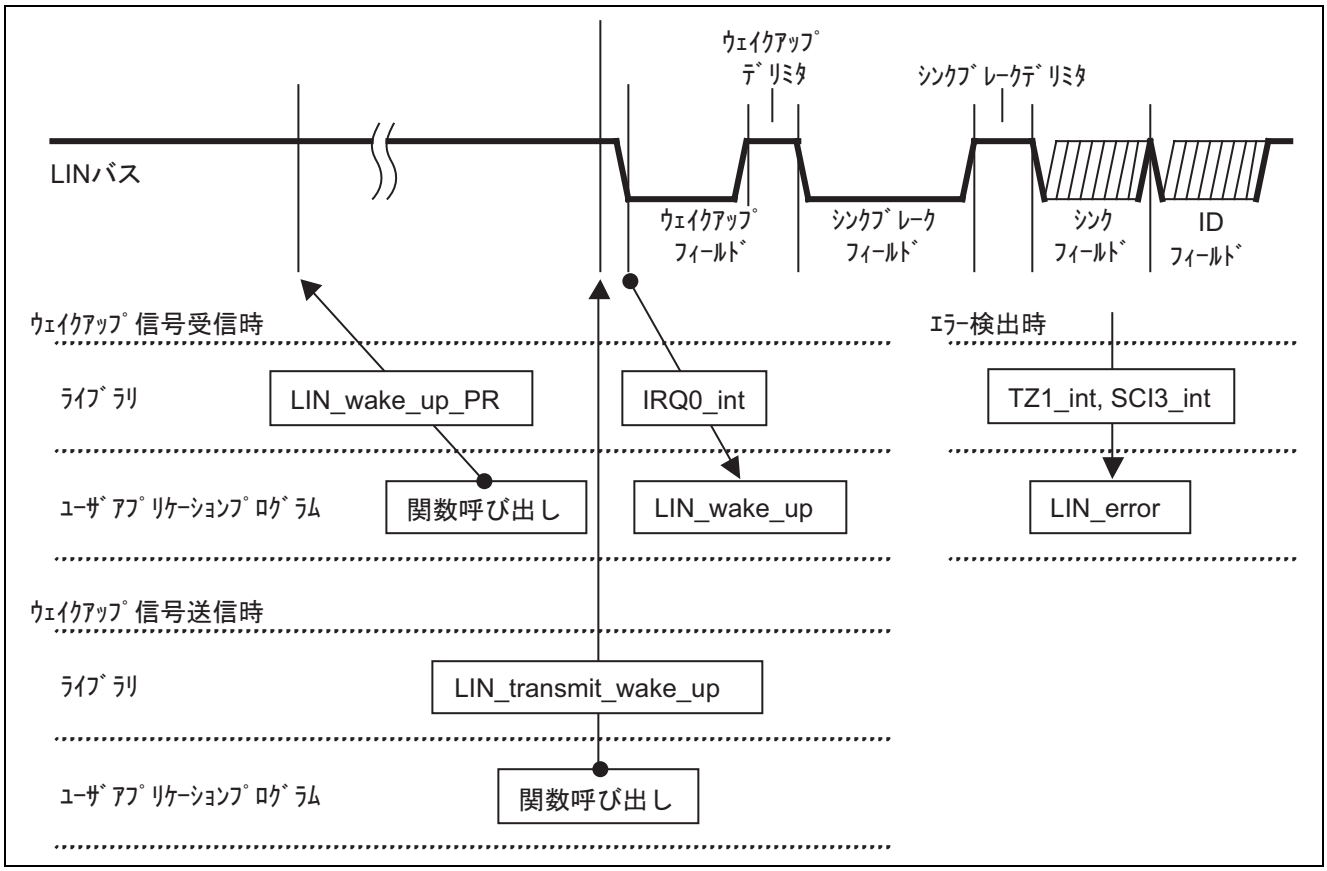

図 6　エラー検出，ウェイクアップ信号送受信時動作概要

- グローバル変数 (RAM 領域格納データ) によるインタフェース
  ユーザアプリケーションプログラム, ライブラリでデータを共有することによりインタフェースを行ないます。

表 4　ユーザアプリケーション・ライブラリ共有データ (グローバル変数)

ラベル名 (変数名)	データ型	機能説明
LIN_tx_id	unsigned char	ヘッダフレーム送信時の送信 ID (ID ビット+DLC ビット) を設定 (表 5　ID 一覧表参照)
LIN_tx_data[0]〜[7]	unsigned char (配列)	レスポンスフレーム送信時の送信データを設定
LIN_rx_id	unsigned char	レスポンスデータ受信時の ID を格納
LIN_rx_data[0]〜[7]	unsigned char (配列)	受信したレスポンスデータを格納
LIN_status	(構造体)	通信ステータス
LIN_status.BYTE	バイトアクセス unsigned char	
	ビットアクセス	
LIN_status.BIT.wk7	ビット- 7	リザーブビット
LIN_status.BIT.CSE	ビット- 6	チェックサムエラーフラグ セット条件：レスポンス受信時にチェックサムエラーを検出した場合
LIN_status.BIT.wk5	ビット- 5〜4	リザーブビット
LIN_status.BIT.SNRE	ビット- 3	スレーブノットレスポンディングエラー セット条件：一定時間内にスレーブからのレスポンスフレームが受信完了されなかった場合
LIN_status.BIT.SCI	ビット- 2	SCI エラー セット条件：SCI3 モジュールによるエラー（オーバランエラー，フレーミングエラー）を検出した場合
LIN_status.BIT.SUC	ビット- 1	メッセージフレーム正常受信完了フラグ セット条件：レスポンスフレームを正常受信完了した場合 クリア条件：次のレスポンス受信用 ID が送信開始された時
LIN_status.BIT.Ready	ビット- 0	ヘッダフレーム送信許可フラグ セット条件：初期設定完了時 メッセージフレーム送受信完了時 通信エラー検出時 クリア条件：メッセージフレーム送受信時
LIN_control	(構造体)	通信コントロール
LIN_control.BYTE	バイトアクセス unsigned char	
	ビットアクセス	
LIN_control.BIT.SB_DEL	ビット- 7〜6	シンクブレークデリミタビット長設定ビット ビット 7　ビット 6 0　0　：　1 ビット 0　1　：　1 ビット 1　0　：　2 ビット 1　1　：　3 ビット
LIN_control.BIT.WU	ビット- 5	(ウェイクアップコントロールビット)
LIN_control.BIT.wk4	ビット- 4〜0	リザーブビット

表 5　ID 一覧表

LIN_tx_id 設定値		ID フィールド送信データ		レスポンス	LIN_tx_id 設定値		ID フィールド送信データ		レスポンス
10 進	16 進	10 進	16 進	データ長	10 進	16 進	10 進	16 進	データ長
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 動作説明

以下にライブラリによる送受信動作説明を示します。

2.5.1 ヘッダフレームの送信

ヘッダフレーム送信許可フラグ (LIN_status.BIT.Ready) が“1”にセットされている時に，ユーザアプリケーションプログラムから LIN_transmit_header 関数を呼び出すことにより，ヘッダフレームの送信を開始します。

LIN_transmit_header 関数呼び出し時には，あらかじめシンクブレークデリミタのビット長(LIN_control.BIT.SB_DEL)，送信 ID (LIN_tx_id) を設定してください。(表 4，表 5 参照)

1. シンクブレークフィールドの送信
   I/O ポート (P22 (TxD 機能兼用端子)) 出力機能によりシンクブレークフィールドドミナント期間を約 13 ビット出力します。

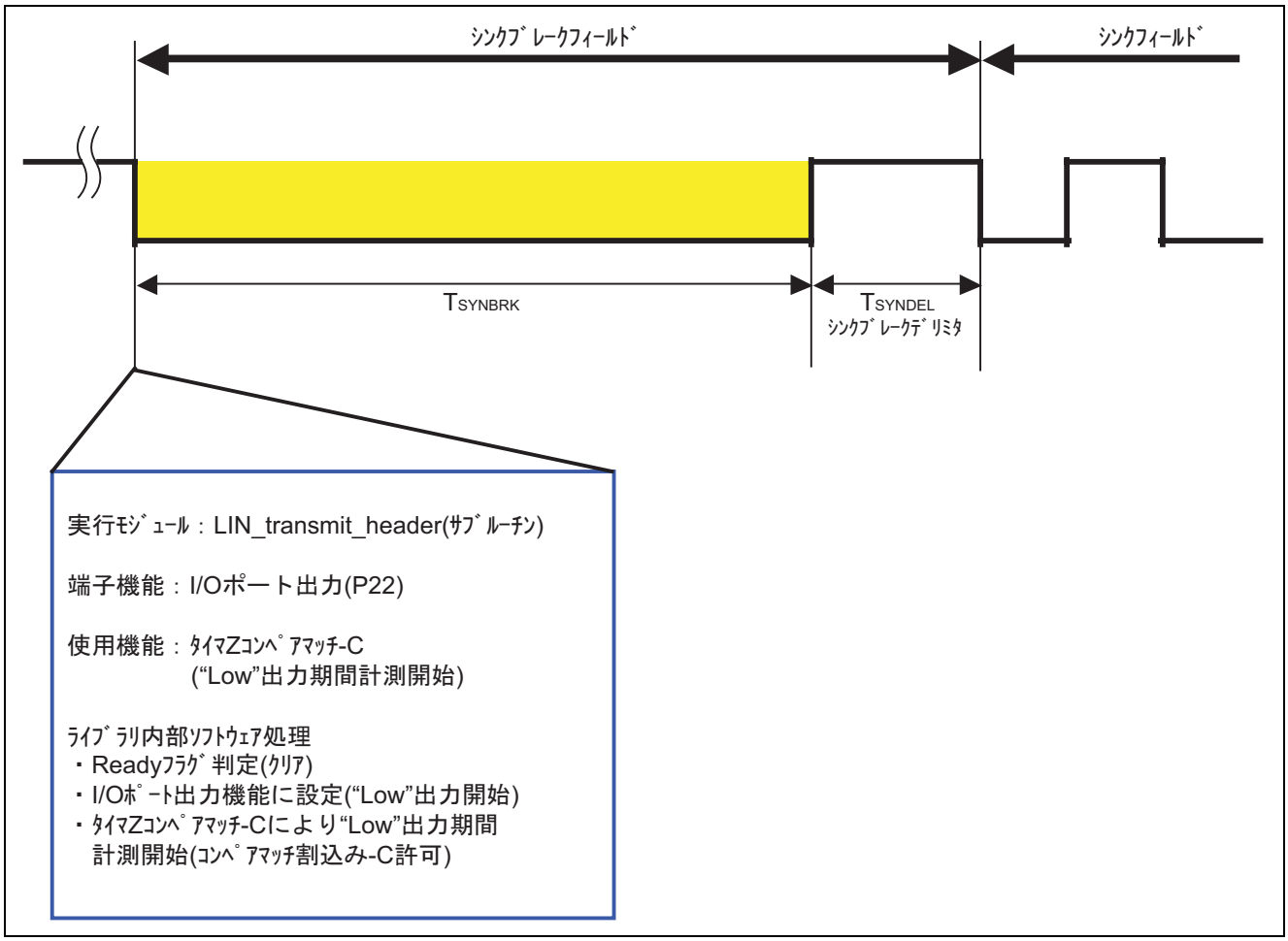

図 7　シンクブレークフィールドドミナント期間出力

2. シンクブレークデリミタの送信
   I/O ポート出力機能によりシンクブレークデリミタ (レセッシブ期間) を出力します。
   LIN_control.BIT.SB_DEL 設定値により約 1～3 ビット期間レセッシブ出力し，ポートを TxD 端子機能 (SCI3:channel-0 (以下 SCI と称す) 送信機能) に切り替えます。

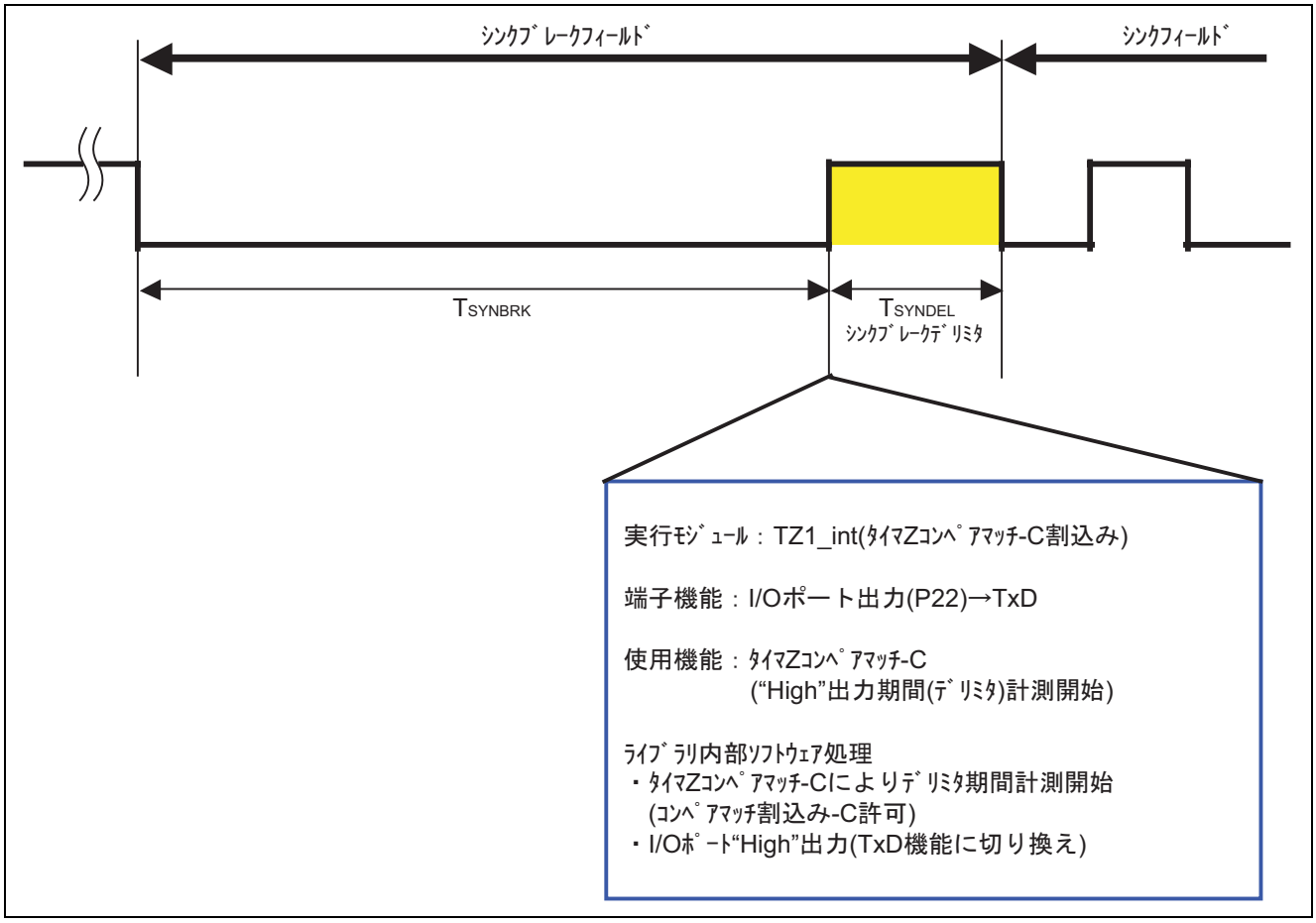

図 8　シンクブレークデリミタ出力

3. シンクフィールドの送信
   SCI 送信機能によりデータ 55h を送信します。

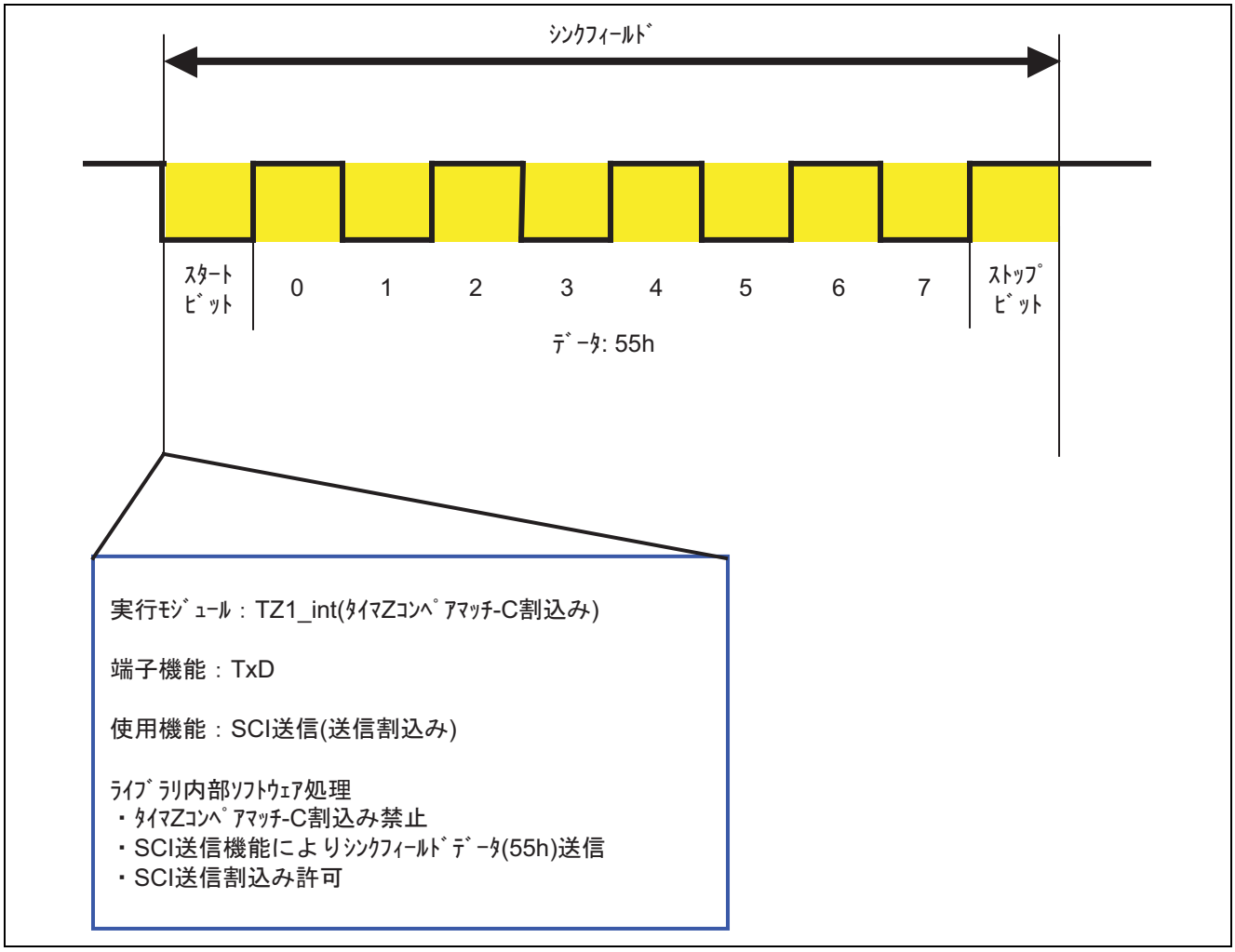

図 9　シンクフィールド送信

4. ID フィールドの送信
SCI 送信機能により ID フィールドデータを送信します。ID フィールドデータは，LIN_tx_id 設定値にパリティビットが自動的に付加されます。(表 5 参照)

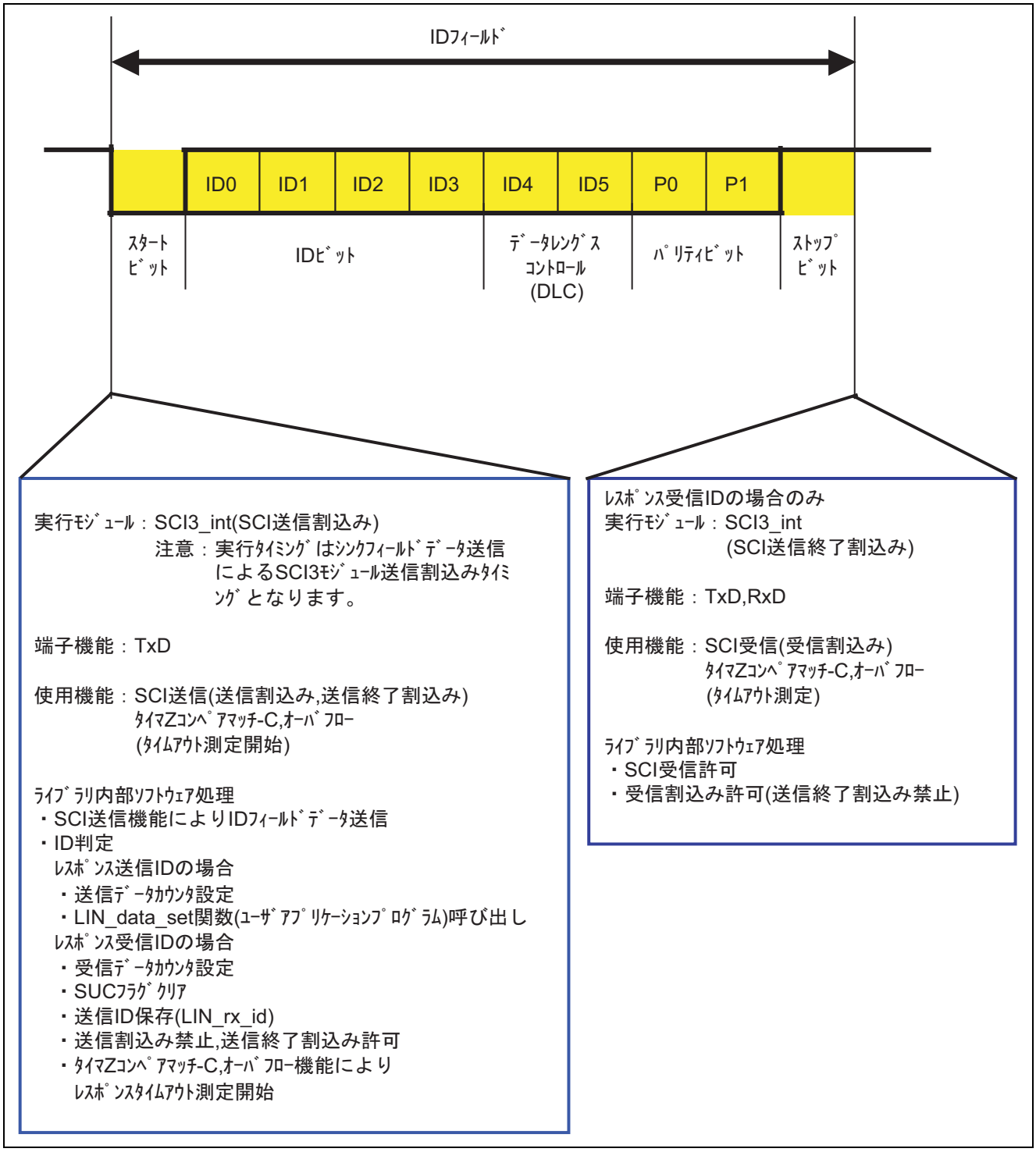

図 10　ID フィールドデータの送信と判定

2.5.2 レスポンスフレームの送受信

ID フィールド送信データがレスポンス送信用 ID であった場合，SCI 送信機能によりレスポンスフレームを送信します。また，レスポンス受信用 ID であった場合，SCI 受信機能によりレスポンスフレームを受信します。

1. データフィールドの送信
   SCI 送信機能によりデータフィールドを送信します。
   送信データは LIN_tx_data[0]～[7]に設定し, LIN_tx_data[0]から順に ID フィールドデータ DLC ビットに従い，それぞれのデータ数 (2, 4, 8 バイト) 送信します。

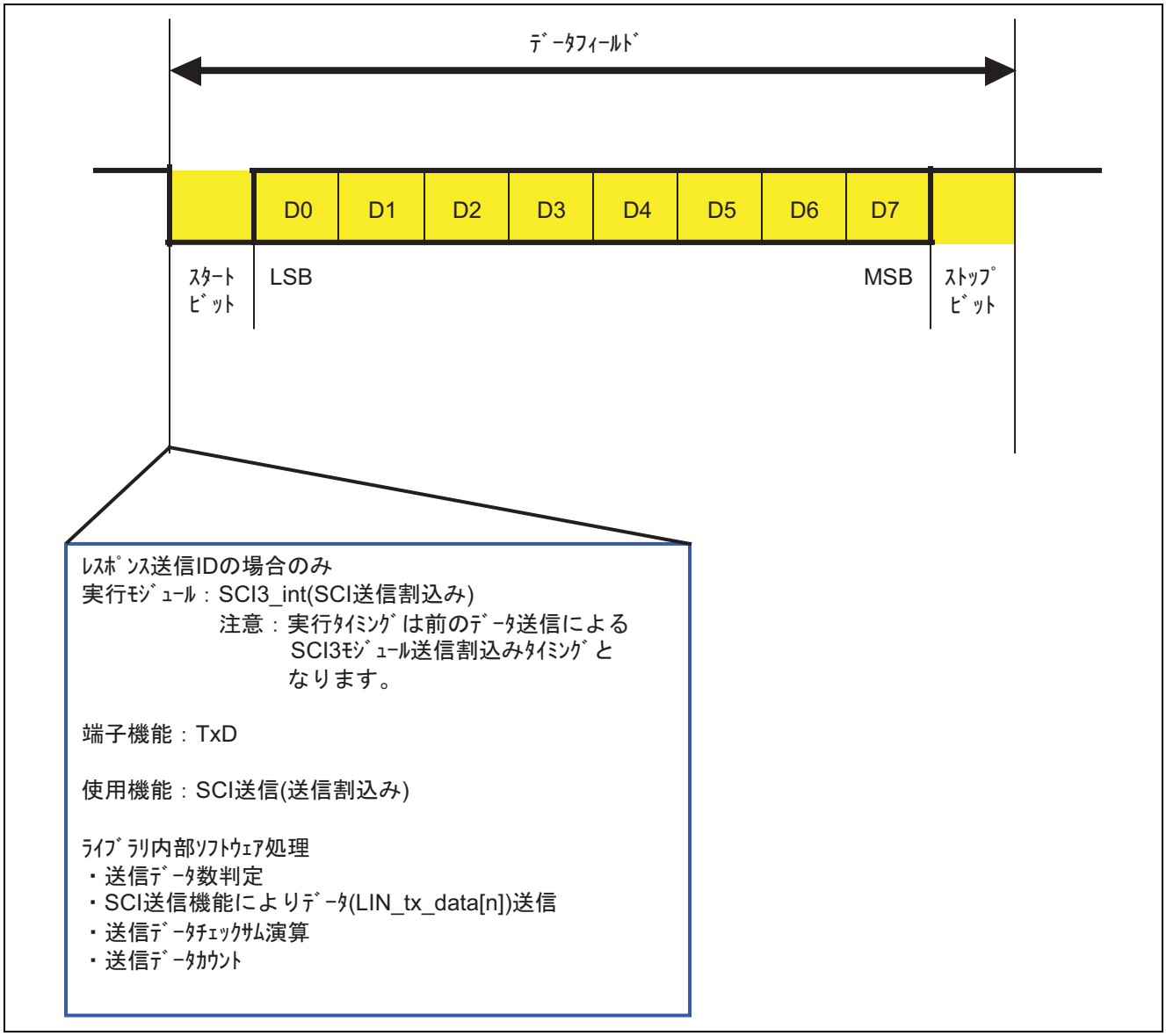

図 11　データフィールドの送信

2. チェックサムフィールドの送信
SCI 送信機能によりチェックサムフィールドを送信します。

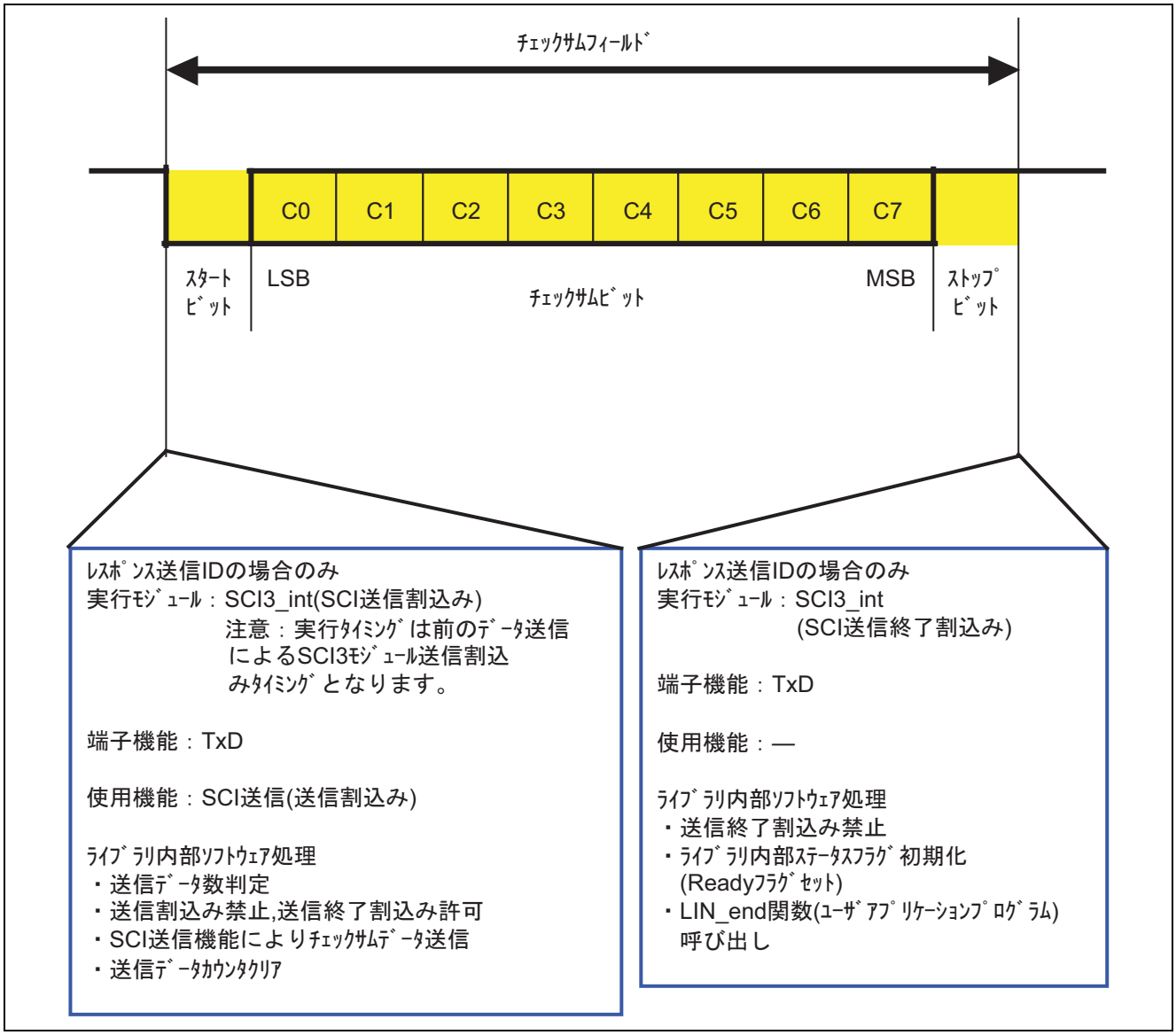

図 12　チェックサムフィールドの送信

3. データフィールドの受信
   SCI 受信機能によりデータフィールドを受信します。
   受信データは LIN_rx_data[0]から順に受信したデータ数のみ LIN_rx_data[0]～[7]に保存します。

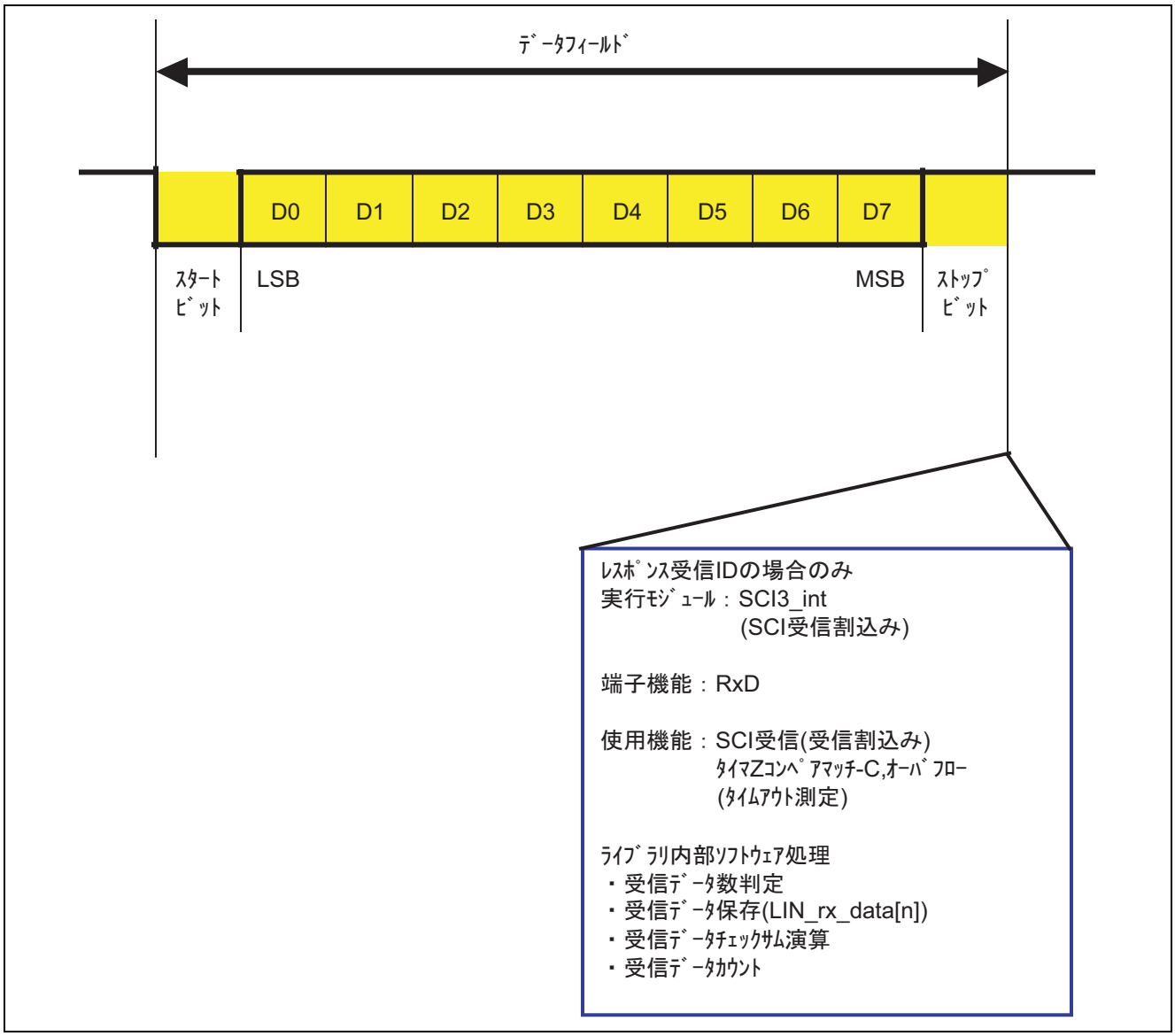

図 13　データフィールドの受信

4. チェックサムフィールドの受信
SCI 受信機能によりチェックサムフィールドを受信し，受信したデータフィールドからの演算結果と比較，判定します。

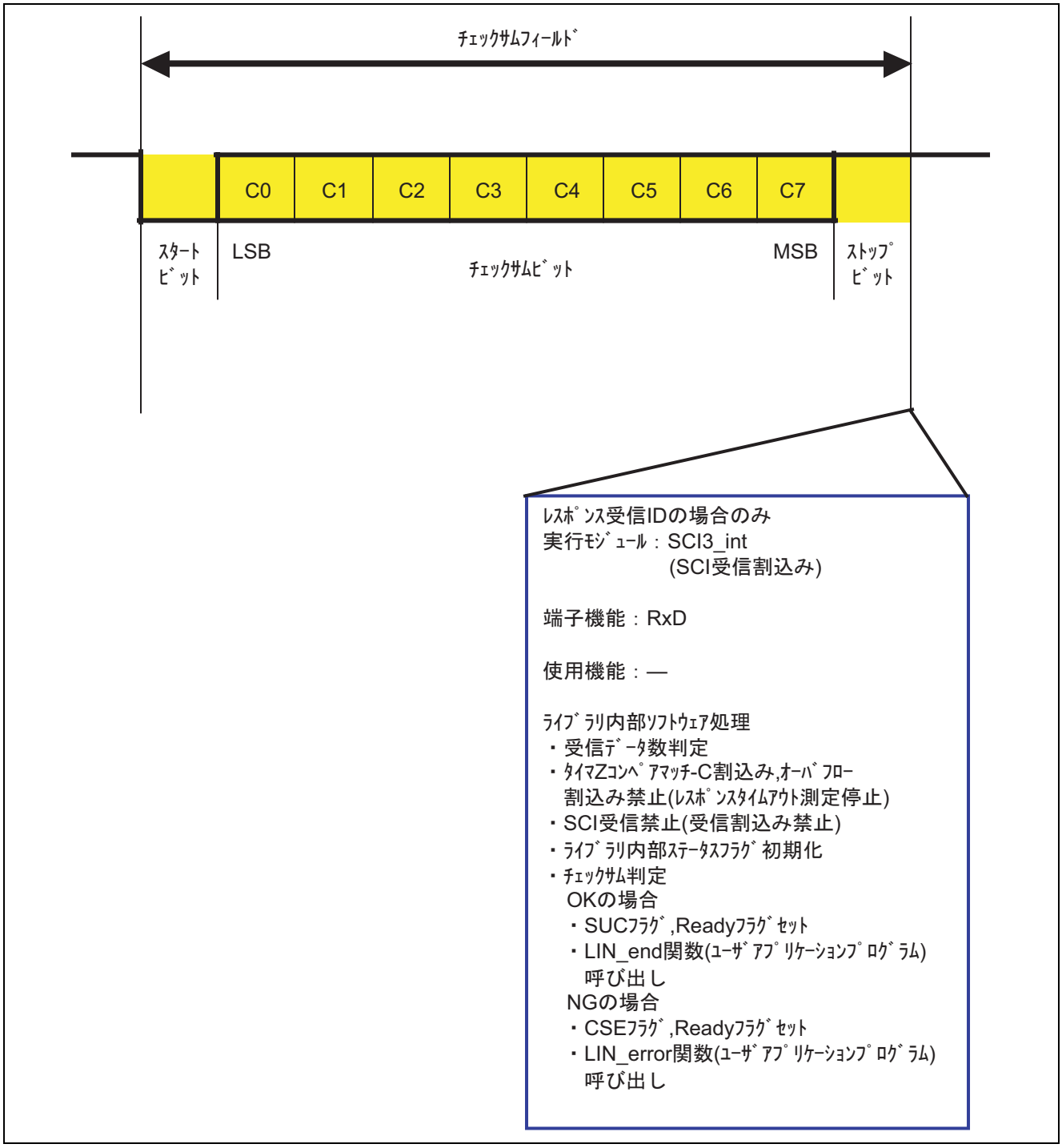

図 14　チェックサムフィールドの受信

2.5.3 ウェイクアップ信号の送受信

SCI 送信機能によりウェイクアップ信号 (送信データ：80h) を送信し, IRQ0 (以下 IRQ と称す) 立下りエッジ検出機能により他ノードからのウェイクアップ信号を検出します。

1. ウェイクアップ信号の送信

   LINID.h 内定義文 (#define __T_WAKEUP __ON) によりコンパイル時にウェイクアップ信号送信機能が組み込まれ, ユーザアプリケーションプログラムから LIN_transmit_wake_up 関数を呼び出すことで SCI 送信機能によりウェイクアップ信号 (送信データ：80h) を送信します。

   本ライブラリは, ウェイクアップデリミタ出力制御, およびリトライ送信制御は行なっておりません。また, Ready フラグはウェイクアップ信号送信中にセットされますが, 他ノード (スレーブノード) の LIN 通信準備が完了する前にヘッダフレーム送信を開始した場合, 正常通信されないことがあります。

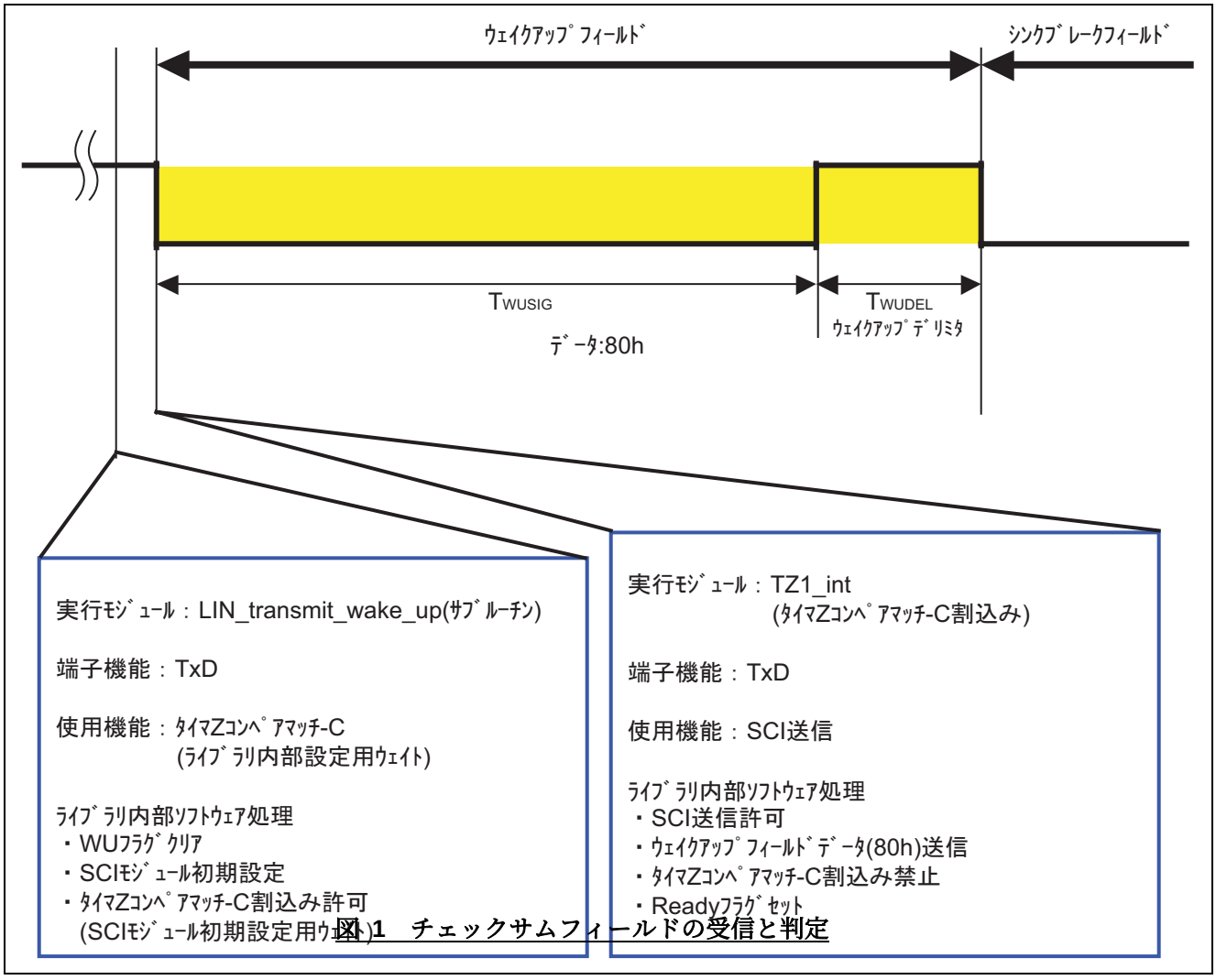

図 1 チェックサムフィールドの受信と判定

図 15 ウェイクアップ信号の送信

2. ウェイクアップ信号の受信

LINID.h 内定義文 (#define __R_WAKEUP __ON) によりコンパイル時にウェイクアップ信号受信機能が組み込まれ，ユーザアプリケーションプログラムから LIN_wake_up_PR 関数を呼び出すことで IRQ 立下りエッジ検出機能による他ノードからのウェイクアップ信号の受信待ち状態となります。
本ライブラリは，ウェイクアップフィールドデータの検証は行なわず，立下りエッジのみの検出となります。

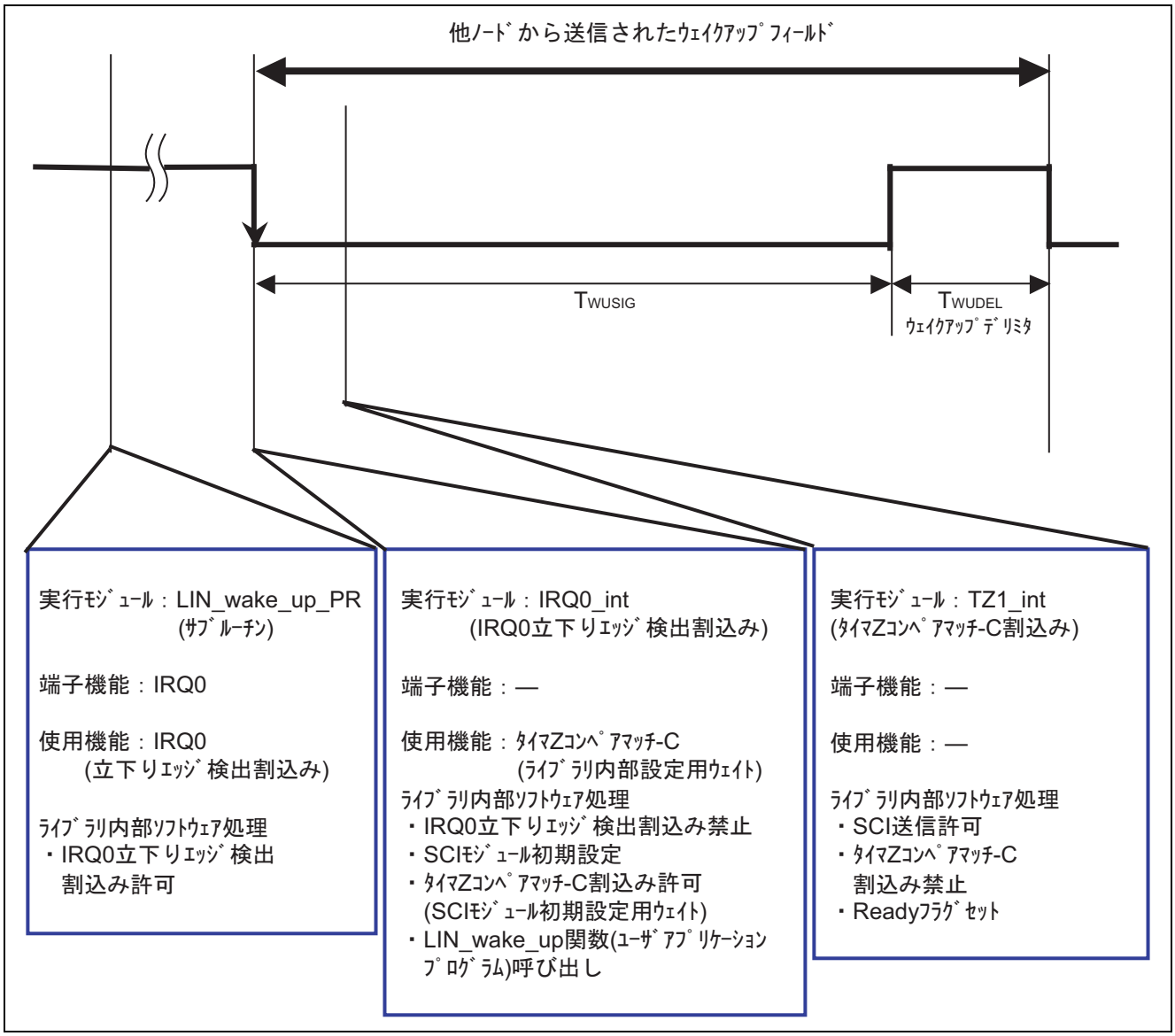

図 16　ウェイクアップ信号の受信

2.5.4 スリープコマンドの発行

LIN の通信プロトコル (LIN Protocol Specification Rev 1.2 , Rev1.3 Draft7) において定義されているスリープコマンド (コマンドフレーム ID (3Ch) 送信，かつレスポンス送信データ 1 バイト目が 00h) は，LIN_tx_id に 3Ch, LIN_tx_data[0]に 00h をそれぞれ設定し, LIN_transmit_header 関数を呼び出すことにより送信されます。

ライブラリにおいて，スリープコマンド送信後の特別な動作 (フラグへの反映やマイコン動作モードの遷移等) はありません。

2.6 ソフトウェア説明

以下に本ライブラリソフトウェアの説明を示します。

2.6.1 ヘッダファイルの組み込み

標準ライブラリ (machine.h), H8/36057 内蔵周辺レジスタ定義ファイル (H8_36057.h), LIN ライブラリ用定義ファイル (LINID.h) の組み込みを行ないます。

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

2.6.2 関数定義

ライブラリ内の関数 (モジュール)定義を行ないます。

LIN_transmit_wake_up 関数は，LINID.h 内の__T_WAKEUP 定義により関数の組み込みを選択されます。LIN_intc_init 関数，LIN_wake_up 関数，LIN_wake_up_PR 関数は，同じく__R_WAKEUP 定義により関数の組み込みを選択されます。

```
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 ライブラリ内部定数定義

ライブラリ内部で使用する定数を定義します。

表 6　ライブラリ内部定数

ラベル名 (定数名)	データ型	機能説明
id_field[0]～[63]	unsigned char (配列)	ID フィールド送信データ (表 5　ID 一覧表参照)
wait_time[0]～[3]	unsigned short (配列)	ライブラリ内部制御用ウェイト設定値 (SCI3 モジュール初期設定，シンクブレークデリミタ期間設定時に使用)
t_13	unsigned short	シンクブレークフィールドドミナント期間 (13 ビット期間) 設定値
flame_max_2	unsigned long	レスポンスタイムアウト MAX 値
flame_max_4		
flame_max_8		
baudrate	unsigned short	SCI3 モジュール用ボーレート設定値

```
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,  xA6, 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 ライブラリ内部変数定義

ライブラリ内部で使用する変数を定義します。

表 7　ライブラリ内部変数

ラベル名（変数名）	データ型	機能説明
ex_counter	unsigned long	タイマ Z 拡張カウンタ
flame_max	unsigned short	レスポンスタイムアウト設定値 (タイマ Z オーバフローカウント値)
t_counter	unsigned char	送信データカウンタ
r_counter	unsigned char	受信データカウンタ
t_checksum	(構造体)	送信データチェックサム演算値
t_checksum.WORD	unsigned short	
t_checksum.BYTE.carry t_checksum.BYTE.data	unsigned char unsigned char	
r_checksum	(構造体)	受信データチェックサム演算値
r_checksum.WORD	unsigned short	
r_checksum.BYTE.carry r_checksum.BYTE.data	unsigned char unsigned char	
in_status	(構造体)	ライブラリ内部ステータス
in_status.BYTE	unsigned char	
in_status.BIT.sync_break	ビット- 7	シンクブレークフィールド送信フラグ
in_status.BIT.sync_break_delimiter	ビット- 6	シンクブレークデリミタ送信フラグ
in_status.BIT.sync_field	ビット- 5	シンクフィールド送信フラグ
in_status.BIT.response_id	ビット- 4	レスポンス ID 判定フラグ レスポンスデータ送信時：1 受信時：0
in_status.BIT.wk3	ビット- 3～2	リザーブビット
in_status.BIT.wu	ビット- 1～0	ウェイクアップ信号送信用フラグ (内部設定用フラグ)

```
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 グローバル変数定義

ユーザアプリケーションプログラム，およびライブラリで共有する変数を定義します。

(表 4 参照)

```
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 初期設定用関数

LIN 通信制御に使用する H8/36057 内蔵周辺機能の初期設定，およびライブラリ内部で使用するソフトウェアフラグ等の初期化を行ないます。

注意：端子 P14 (IRQ0), P21 (RxD), P22 (TxD) は LIN 通信で使用します。ユーザアプリケーションでポート 1，ポート 2 におけるその他の端子 (P10～P12, P15～P17, P20, P23, P24) を使用する場合，下記ソースファイル内 LIN_port_init 関数における PCR2，および LIN_intc_init 関数における PCR1 の設定文により端子設定が変更されるおそれがあります。前述の端子をご使用の際には，PCR の設定をユーザアプリケーションプログラム内で行なったうえ，下記ソースファイル内の PCR1, PCR2 の設定文を削除またはコメントアウトしてください。

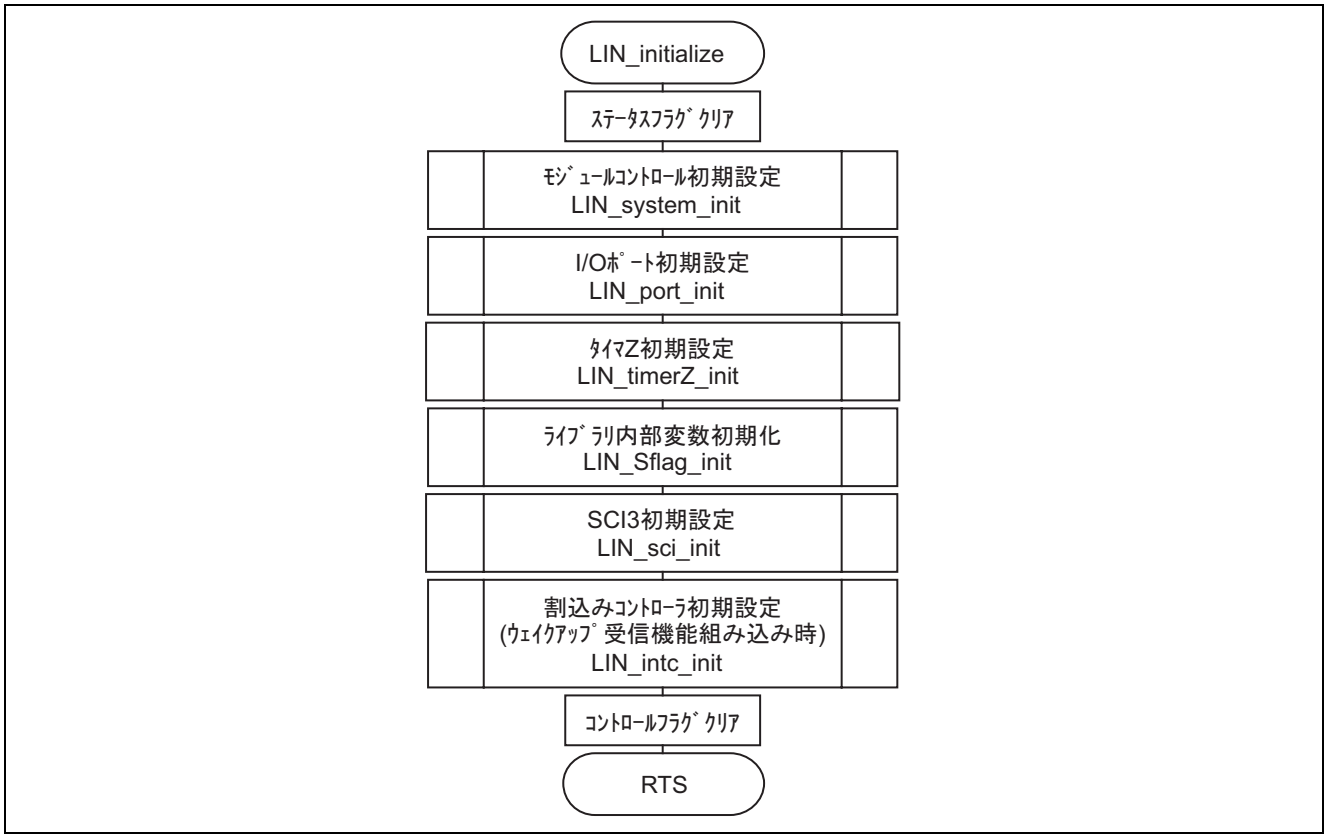

図 17　初期設定用関数フローチャート

```
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 ヘッダフレーム送信用関数

ヘッダフレーム (シンクブレークフィールド，シンクフィールド，ID フィールド) の送信を開始します。

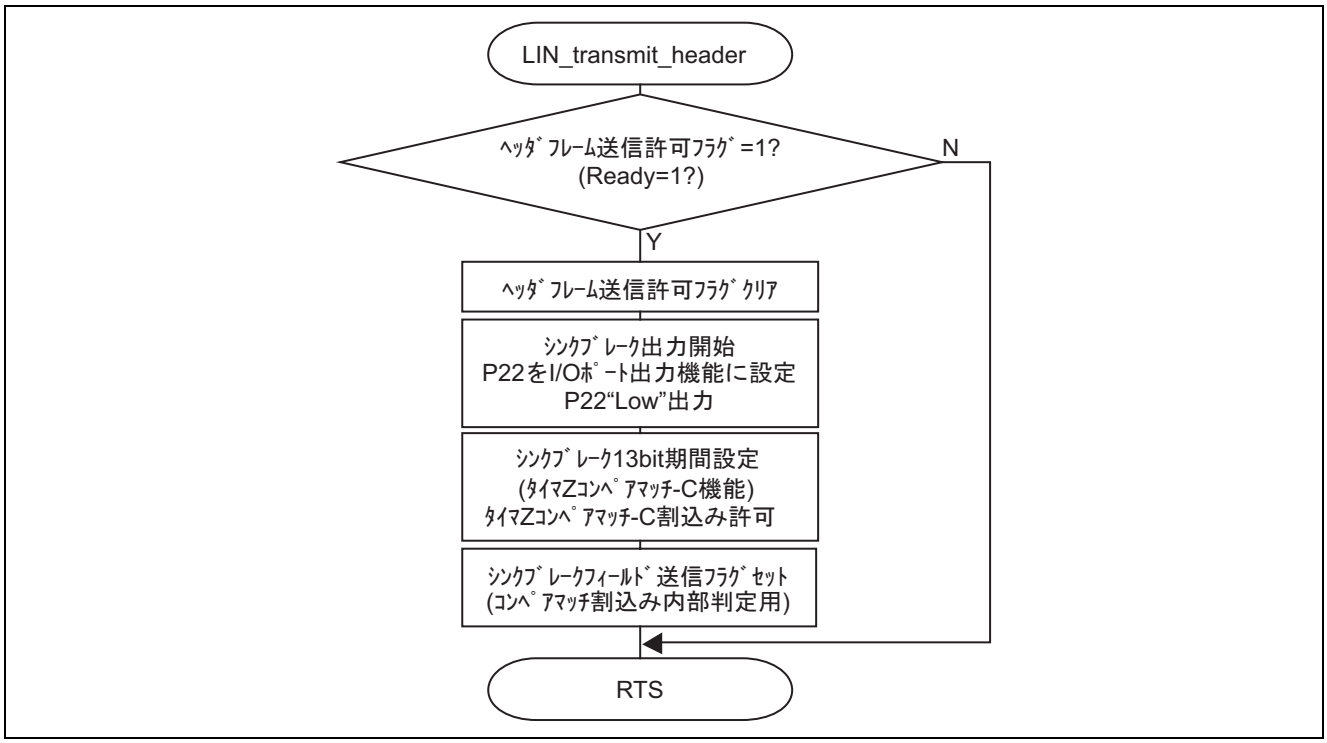

図 18　ヘッダフレーム送信用関数フローチャート

```
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 ウェイクアップ信号送信用関数

ウェイクアップ信号を送信します。

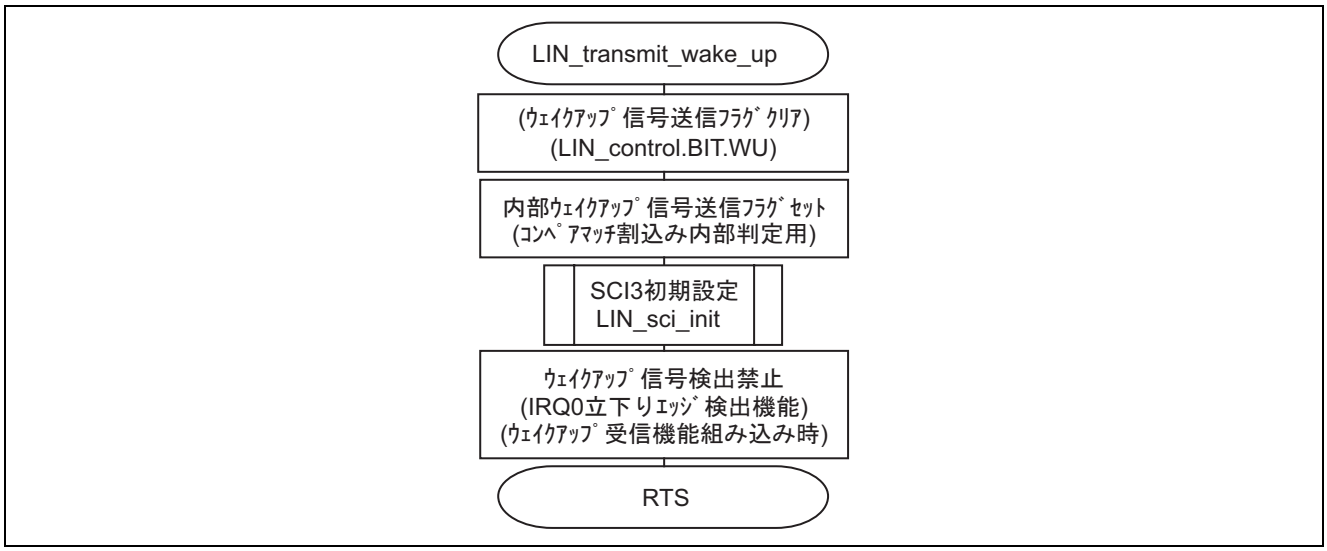

図 19 ウェイクアップ信号送信用関数フローチャート

```
#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 ウェイクアップ信号受信準備用関数

他ノード (スレーブノード) からのウェイクアップ信号受信準備を行ないます。

図 20 ウェイクアップ信号受信準備用関数フローチャート

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

2.6.10 IRQ 割込み関数

IRQ0 立下りエッジ検出割込み処理を行ないます。2.7.9　ウェイクアップ信号受信準備用関数による設定後, LIN バス上の立下りエッジ (他ノードからのウェイクアップ信号等) を検出し, LIN 通信制御の準備を行ないます。

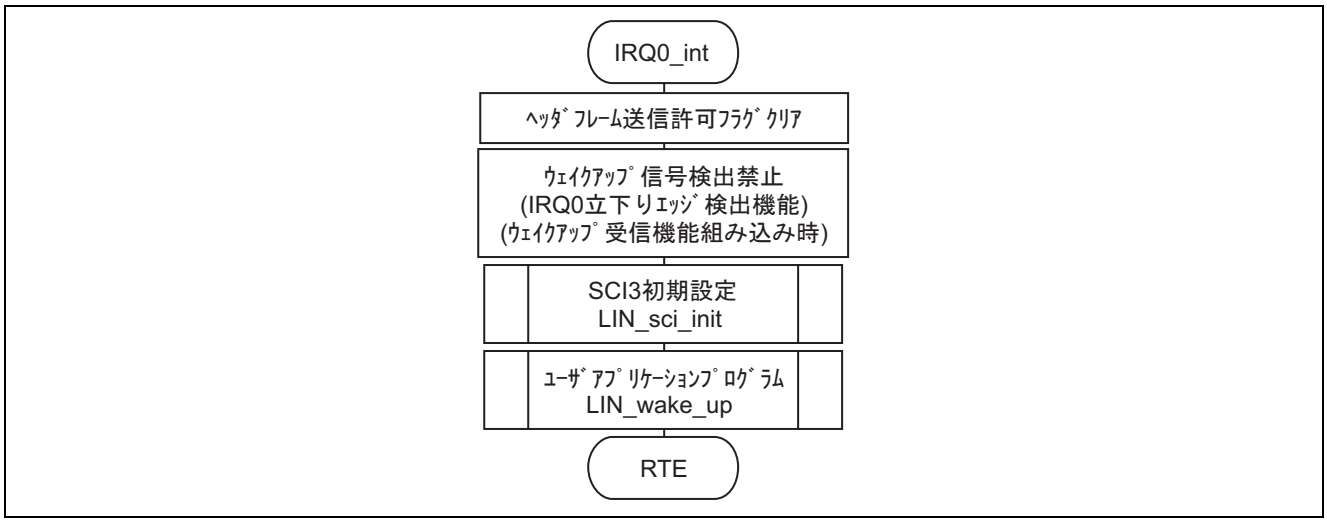

図 21　IRQ 割込み関数フローチャート

```
#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 タイマ Z 割込み関数

タイマ Z (channel-1) のオーバフロー割込み，およびコンペアマッチ-C 割込み処理を行ないます。

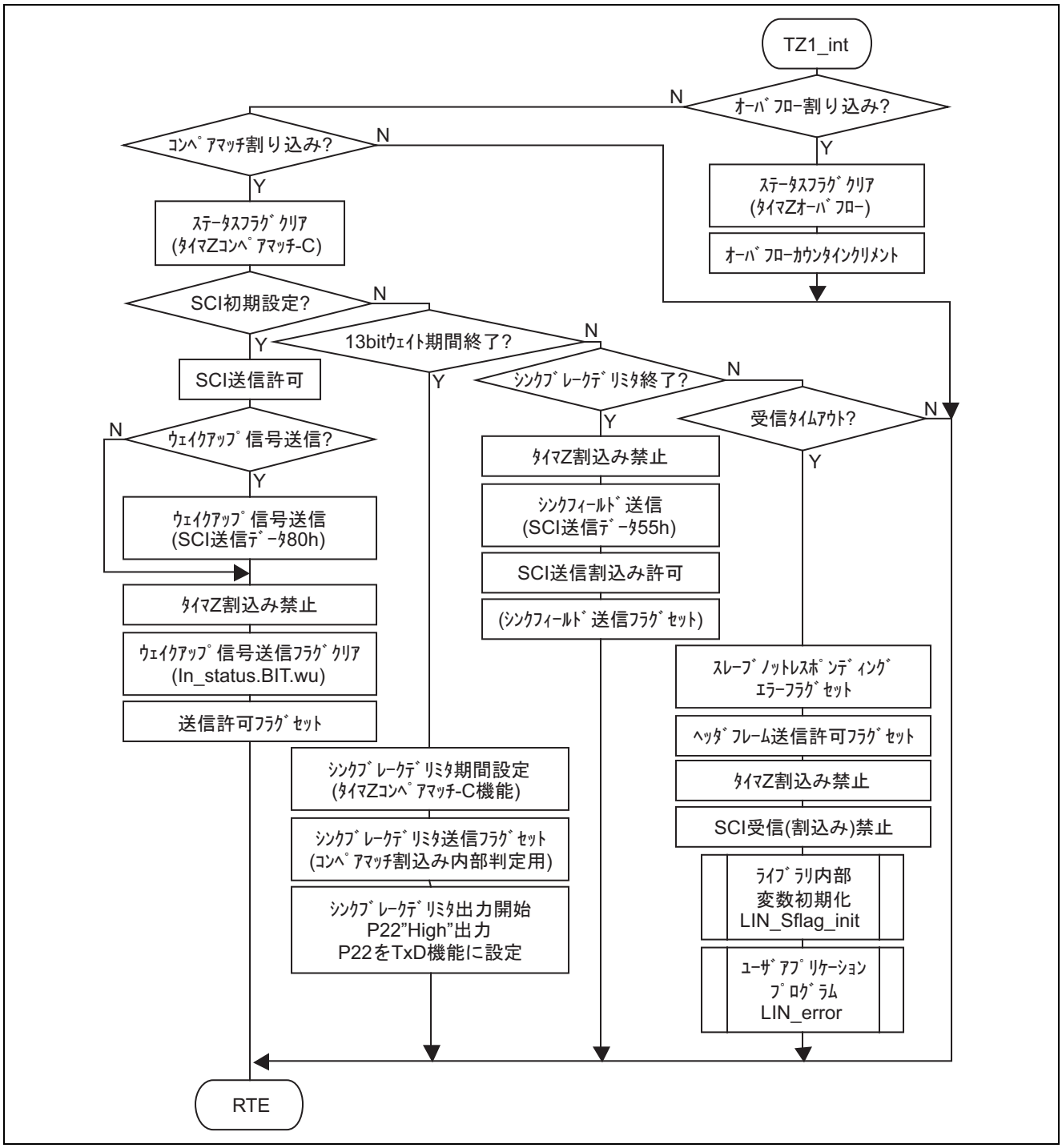

図 22　タイマ Z 割込み関数フローチャート

```
#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 割込み関数

SCI3 (channel-0) のエラー検出割込み，受信割込み，送信割込み，送信終了割込み処理を行ないます。

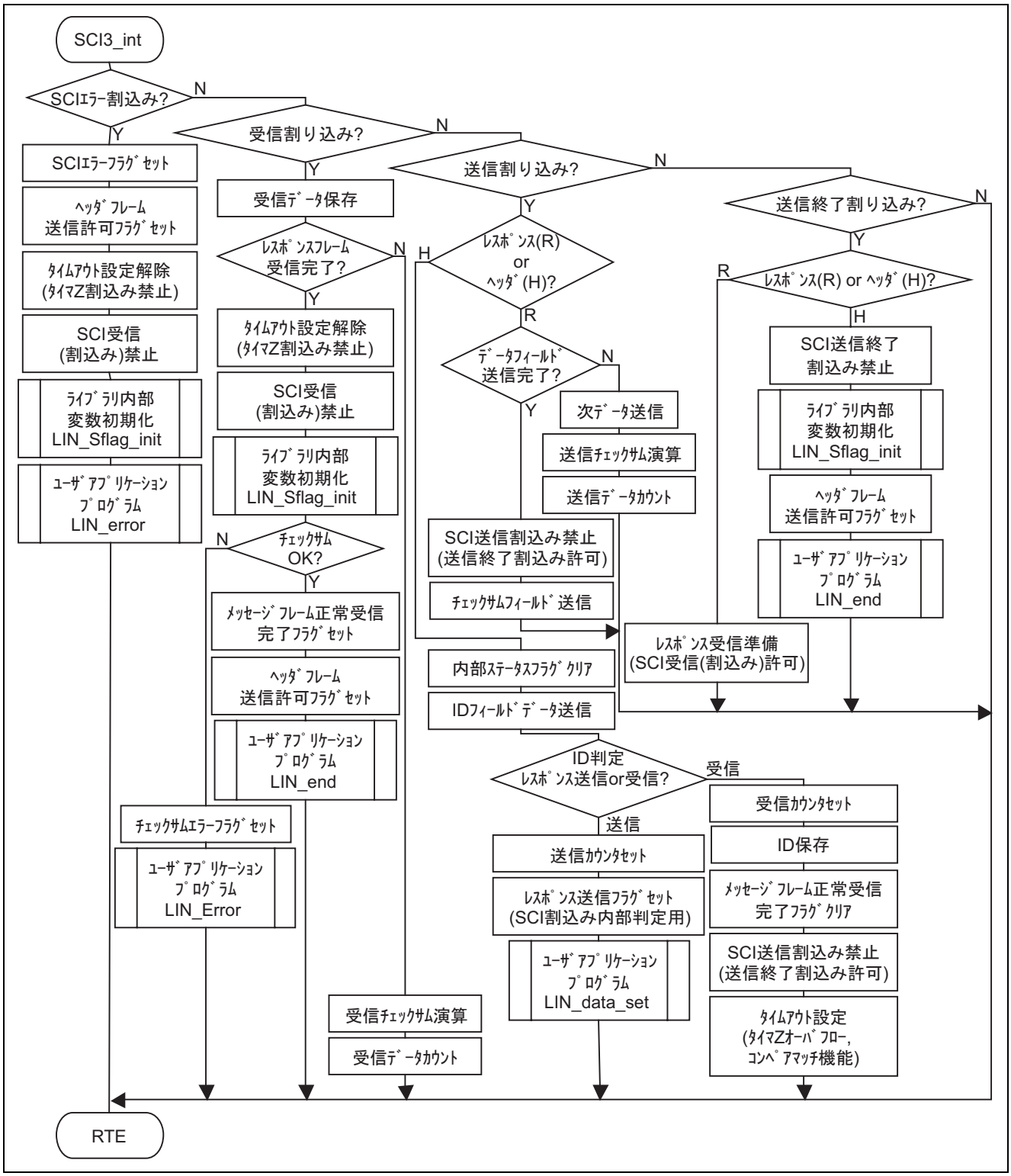

図 23　SC13 割込み関数フローチャート

```
#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;
                }
             }
}
```

3. 参考文献

- LIN Protocol Specification Revision 1.2
- LIN Protocol Specification Revision 1.3 Draft 7
- H8/36057 グループ，H8/36037 グループハードウェアマニュアル　Rev.4.00
  (最新版をルネサス テクノロジホームページから入手してください。)

ホームページとサポート窓口

ルネサステクノロジホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry
csc@renesas.com

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2003.12.22	—	初版発行
1.01	2007.06.15	12	誤記訂正

本資料ご利用に際しての留意事項

1. 本資料は、お客様に用途に応じた適切な弊社製品をご購入いただくための参考資料であり、本資料中に記載の技術情報について弊社または第三者の知的財産権その他の権利の実施、使用を許諾または保証するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例など全ての情報の使用に起因する損害、第三者の知的財産権その他の権利に対する侵害に関し、弊社は責任を負いません。
3. 本資料に記載の製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的、あるいはその他軍事用途の目的で使用しないでください。また、輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、それらの定めるところにより必要な手続を行ってください。
4. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例などの全ての情報は本資料発行時点のものであり、弊社は本資料に記載した製品または仕様等を予告なしに変更することがあります。弊社の半導体製品のご購入およびご使用に当たりましては、事前に弊社営業窓口で最新の情報をご確認頂きますとともに、弊社ホームページ（http://www.renesas.com）などを通じて公開される情報に常にご注意下さい。
5. 本資料に記載した情報は、正確を期すため慎重に制作したものですが、万一本資料の記述の誤りに起因する損害がお客様に生じた場合においても、弊社はその責任を負いません。
6. 本資料に記載の製品データ、図、表などに示す技術的な内容、プログラム、アルゴリズムその他応用回路例などの情報を流用する場合は、流用する情報を単独で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断して下さい。弊社は、適用可否に対する責任は負いません。
7. 本資料に記載された製品は、各種安全装置や運輸・交通用、医療用、燃焼制御用、航空宇宙用、原子力、海底中継用の機器・システムなど、その故障や誤動作が直接人命を脅かしあるいは人体に危害を及ぼすおそれのあるような機器・システムや特に高度な品質・信頼性が要求される機器・システムでの使用を意図して設計、製造されたものではありません（弊社が自動車用と指定する製品を自動車に使用する場合を除きます）。これらの用途に利用されることをご検討の際には、必ず事前に弊社営業窓口へご照会下さい。なお、上記用途に使用されたことにより発生した損害等について弊社はその責任を負いかねますのでご了承願います。
8. 第7項にかかわらず、本資料に記載された製品は、下記の用途には使用しないで下さい。これらの用途に使用されたことにより発生した損害等につきましては、弊社は一切の責任を負いません。
   1）生命維持装置。
   2）人体に埋め込み使用するもの。
   3）治療行為（患部切り出し、薬剤投与等）を行なうもの。
   4）その他、直接人命に影響を与えるもの。
9. 本資料に記載された製品のご使用につき、特に最大定格、動作電源電圧範囲、放熱特性、実装条件およびその他諸条件につきましては、弊社保証範囲内でご使用ください。弊社保証値を越えて製品をご使用された場合の故障および事故につきましては、弊社はその責任を負いません。
10. 弊社は製品の品質および信頼性の向上に努めておりますが、特に半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。弊社製品の故障または誤動作が生じた場合も人身事故、火災事故、社会的損害などを生じさせないよう、お客様の責任において冗長設計、延焼対策設計、誤動作防止設計などの安全設計（含むハードウエアおよびソフトウエア）およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特にマイコンソフトウエアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願い致します。
11. 本資料に記載の製品は、これを搭載した製品から剥がれた場合、幼児が口に入れて誤飲する等の事故の危険性があります。お客様の製品への実装後に容易に本製品が剥がれることがなきよう、お客様の責任において十分な安全設計をお願いします。お客様の製品から剥がれた場合の事故につきましては、弊社はその責任を負いません。
12. 本資料の全部または一部を弊社の文書による事前の承諾なしに転載または複製することを固くお断り致します。
13. 本資料に関する詳細についてのお問い合わせ、その他お気付きの点等がございましたら弊社営業窓口までご照会下さい。