

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のですが、誤りが無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、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と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 H8/36014 グループ

## LIN (Local Interconnect Network) スレーブ編

---

### 要旨

LIN (Local Interconnect Network) アプリケーションノート H8/300H Tiny シリーズ H8/36014 グループ スレーブ編は、H8/300H Tiny シリーズ H8/36014 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使用例、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。

なお、本アプリケーションノートに掲載されているプログラム、回路等の動作は確認しておりますが、実際にご使用になる場合は、必ず動作確認の上ご使用くださいますようお願いいたします。

### 動作確認デバイス

H8/300H Tiny シリーズ H8/36014F

### 目次

1. LIN 通信システム概要 .....	2
2. LIN2.0 ライブラリ仕様 .....	9
3. 参考文献 .....	34

## 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 LIN2.0 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。

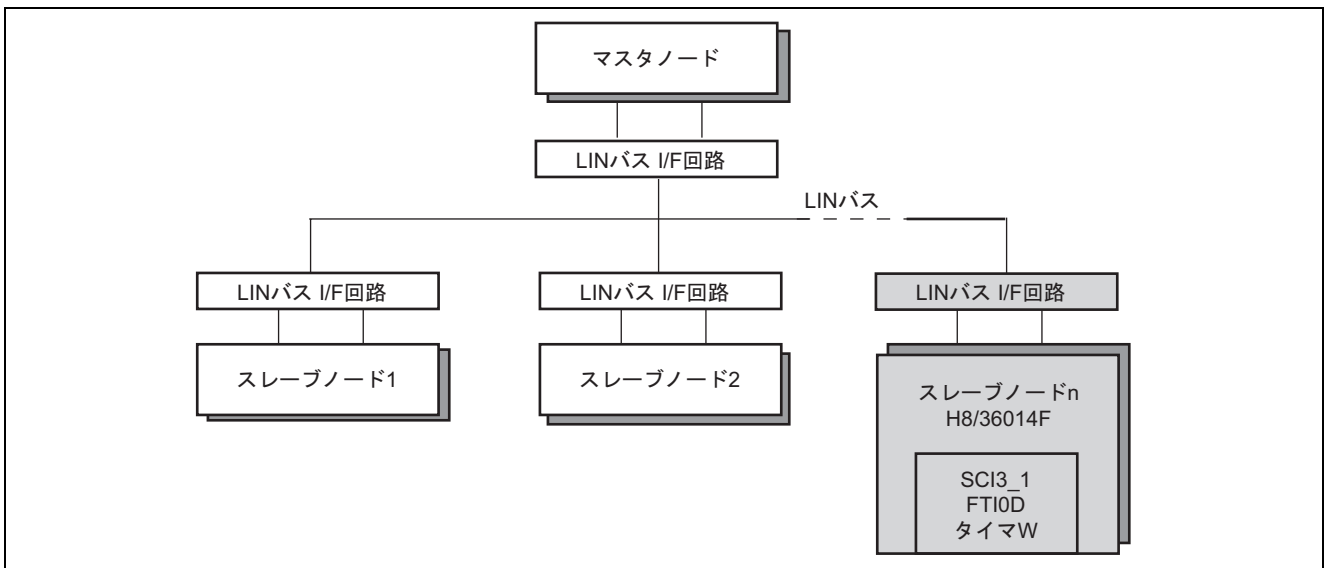


図 1 システム構成

1.1.2 使用リソース

表 1 に H8/36014F で使用するリソースを示します。

表 1 スレーブノードが使用する CPU リソース

機能	端子機能 (端子 No.)	用途	説明
I/O ポート	P80 (36pin)	LIN トランシーバ制御	I/O ポート (High/Low) 出力により LIN トランシーバ有効/無効切り替え (ユーザによりリセット時 High/出力に設定)
SCI3 (channel-1)	送信	TXD (46pin)	レスポンスフレーム送信 WakeUp 信号出力
	受信	RXD (45pin)	レスポンスフレーム受信
			通信エラー検出
タイマ W	FTIOD (40pin)	ヘッダ検出 (インプットキャプチャ) WakeUp 信号期間計測 (アウトプットコンペア)	Break 期間/Sync バイト期間を検出し自動速度調整

## 1.2 LIN 通信概要

LIN 通信プロトコルにより送受信される各フレームの概要を示します。

### 1.2.1 無条件フレーム (Unconditional Frame)

無条件フレームは、信号値更新の有無にかかわらず、必ず送受信されるフレームです。

ヘッダに対するレスポンス送信ノードはマスタノード/スレーブノードのいずれかになります。また、レスポンス受信ノードもマスタノード/スレーブノードのいずれかになります。

図 2 に無条件フレームのシーケンス内容を示します。

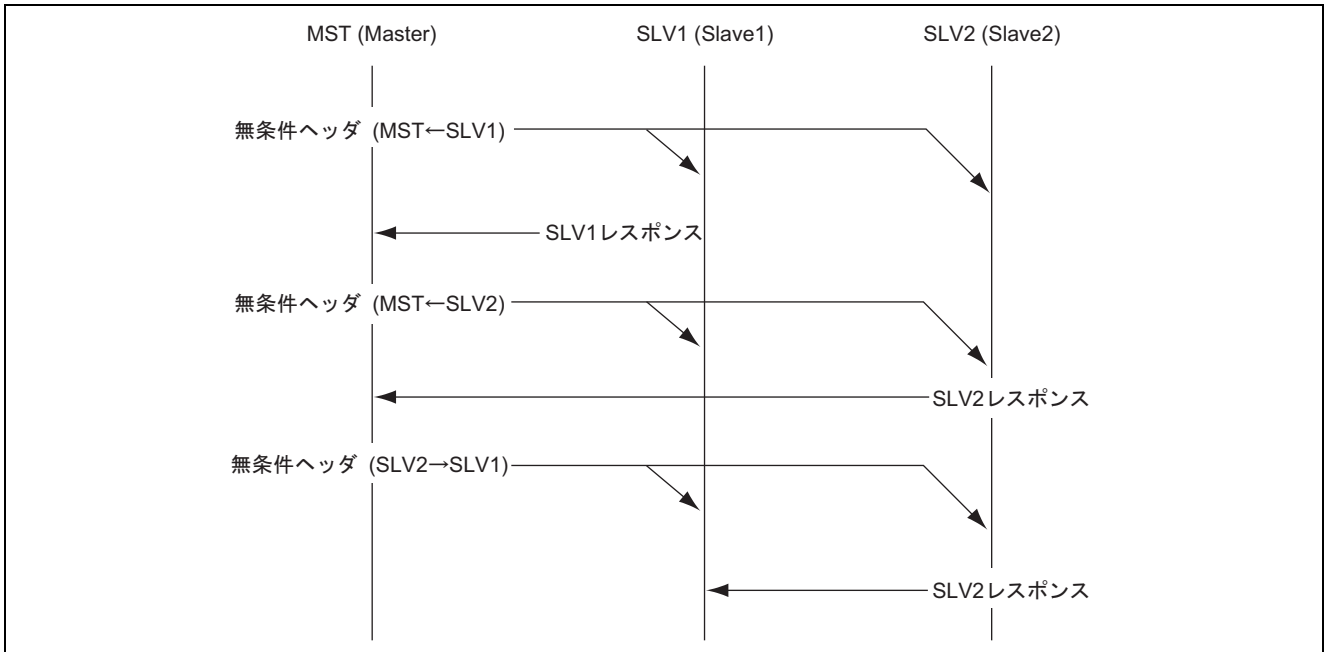


図 2 無条件フレームのシーケンス内容

### 1.2.2 イベントトリガフレーム (Event Triggered Frame)

イベントトリガフレームは、マスタノードがスレーブノードに対し、信号値更新の有無を確認するために送受信されるフレームです。

ヘッダに対するレスポンス送信ノードは信号値に更新があったスレーブノードからのみとなります。このとき、複数のスレーブノードがレスポンスを送信する場合があります、その場合にはコリジョンが発生します。コリジョン発生時にはマスタノードが一旦すべてのスレーブノードに対して、無条件フレームにより信号値の確認を行いません。また、レスポンス受信ノードはマスタノードのみとなります。

図3にイベントトリガフレームのシーケンス内容を示します。

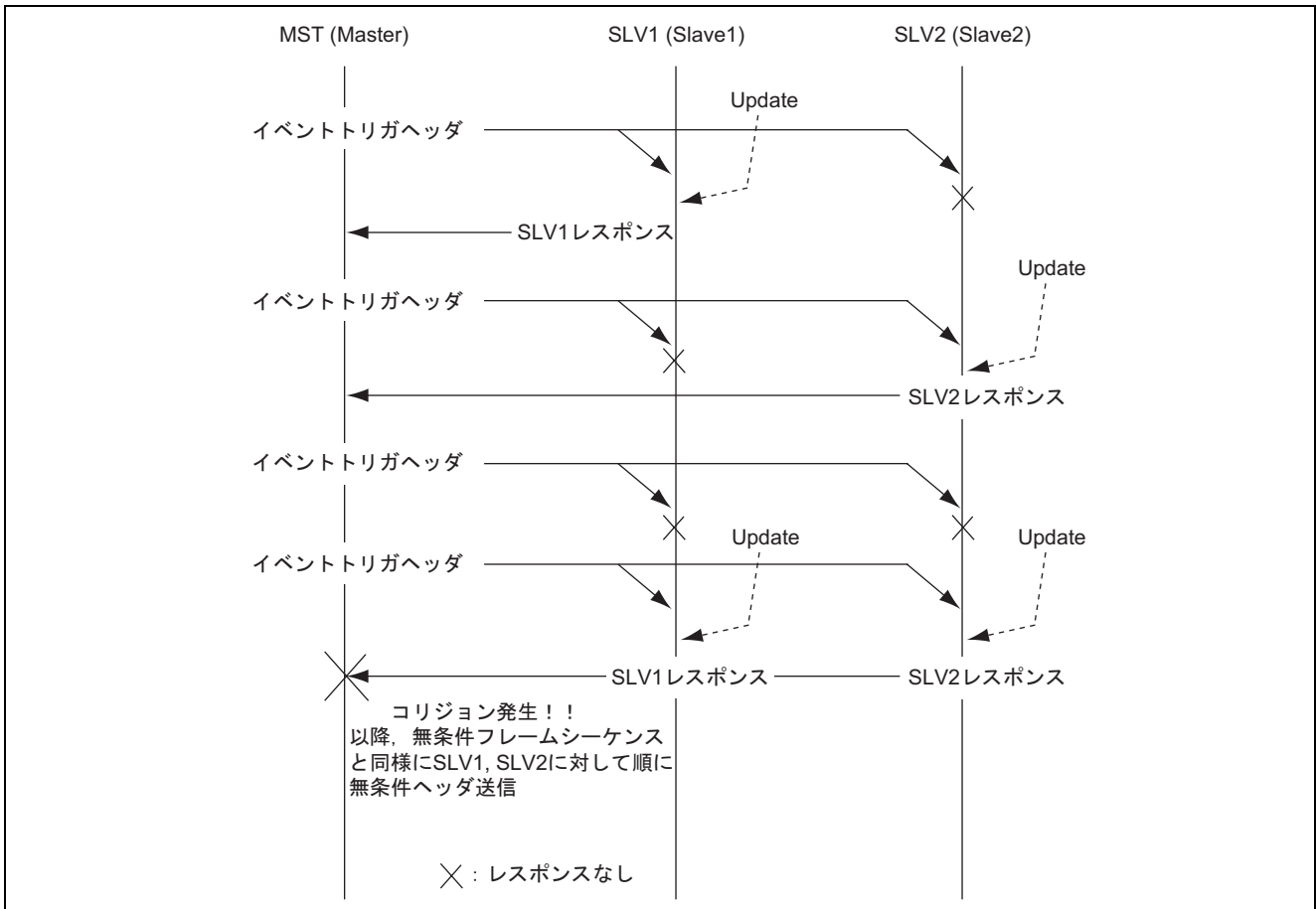


図3 イベントトリガフレームのシーケンス内容

### 1.2.3 散発フレーム (Sporadic Frame)

散発フレームは、マスタノードが管理する信号値に更新があった場合に、関連するスレーブノードすべてに対して、信号値の通知を行ないます。ヘッダに対するレスポンス送信ノードは必ずマスタノードのみからとなります。

図4に散発フレームのシーケンス内容を示します。

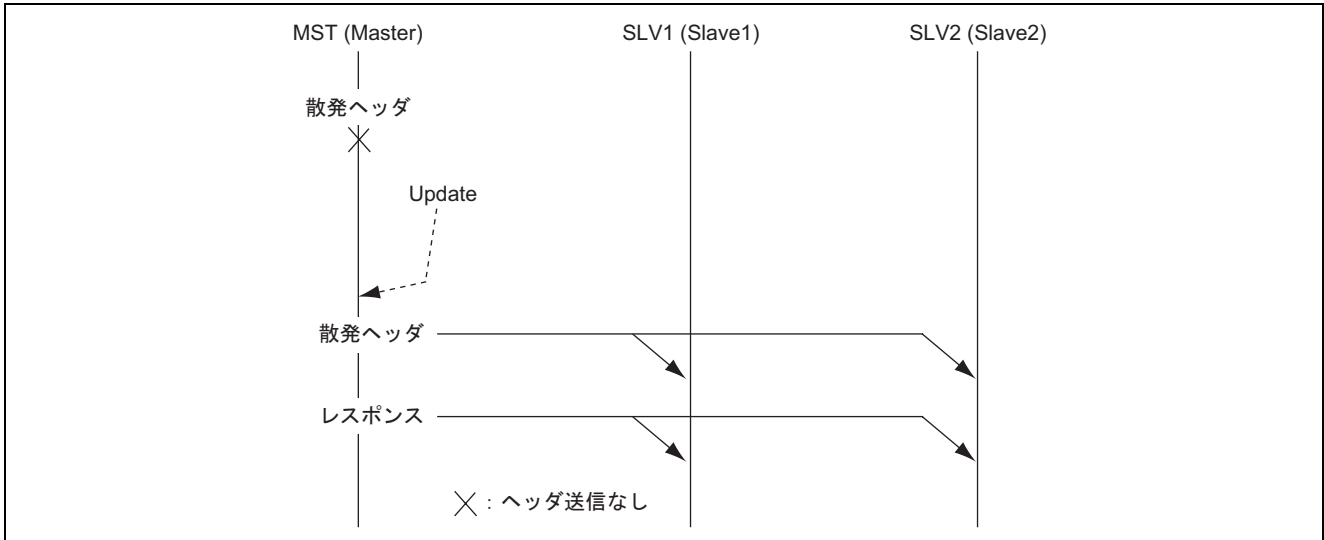


図4 散発フレームのシーケンス内容



#### 1.2.4 マスタリクエストフレーム (Master Request Frame)

マスタリクエストフレームは、マスタノードからスレーブノードに対して、ノード設定フレーム、ノード診断フレームの送信を行ないます。ヘッダに対するレスポンスは必ずマスタノードのみからとなります。

図5にマスタリクエストフレームのシーケンス内容を示します。

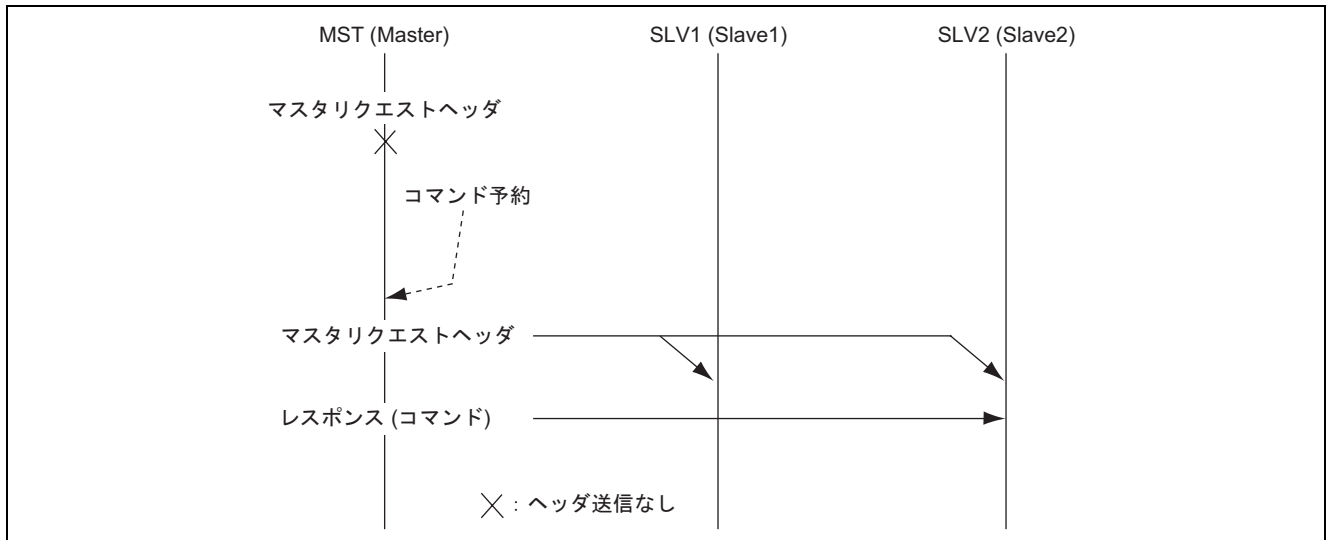


図5 マスタリクエストフレームのシーケンス内容

### 1.2.5 スレーブレスポンスフレーム (Slave Response Frame)

スレーブレスポンスフレームは、マスタノードからスレーブノードに対して、ノード設定フレームに対するレスポンス、ノード診断フレーム有無の確認/受信を行ないます。ヘッダに対するレスポンスは必ずスレーブノードのみからとなります。このとき複数のスレーブノードが反応するクラスタ構成であってはなりません。また、スレーブノードは応答するレスポンスがない場合には、レスポンスの送信を行ないません。

図 6 にスレーブレスポンスフレームのシーケンス内容を示します。

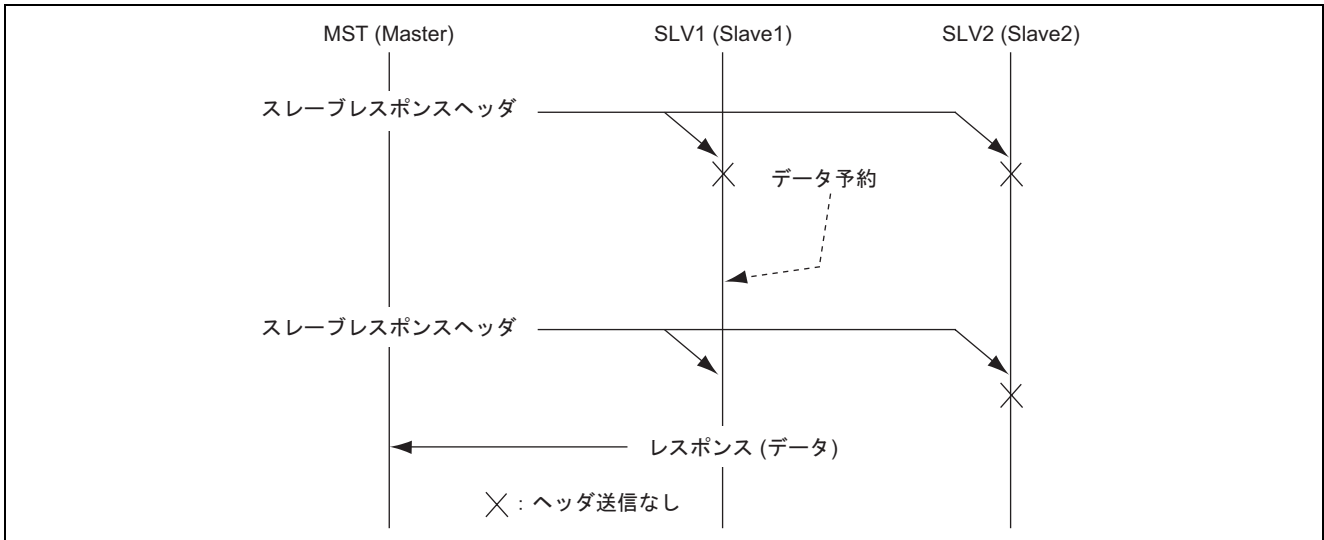


図 6 スレーブレスポンスフレームのシーケンス内容

## 2. LIN2.0 ライブラリ仕様

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ことにより H8/36014F の内蔵機能を使用し、スレーブノードとして LIN 通信を行ないます。

### 2.1 ファイル構成

- **36014s.h (Ver.1.00)**  
H8/36014 グループ用内蔵 I/O レジスタ定義ファイルです。
- **sci\_drv36014.c (Ver.1.00)**  
H8/36014F における LIN 通信用 (スレーブ) SCI3 機能設定、および通信制御を行なう SCI3 ドライバの C ソースファイルです。このファイルはユーザが使用する CPU 環境により自由に変更/改造することができます。このファイルは LIN2.0 ライブラリとは切り離され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組み込む必要があります。
- **sci\_drv36014.h (Ver.1.00)**  
H8/36014F における LIN 通信用 (スレーブ) SCI3 機能設定、および通信制御を行なう SCI3 ドライバのヘッダファイルです。このファイルはユーザが使用する CPU 環境により自由に変更/改造することができます。このファイルは LIN2.0 ライブラリとは切り離され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組み込む必要があります。
- **tmr\_drv36014.c (Ver.1.00)**  
H8/36014F における LIN 通信用 (スレーブ) タイマ W 機能設定、カウント制御を行なうタイマ W ドライバの C ソースファイルです。このファイルはユーザが使用する CPU 環境により自由に変更/改造することができます。このファイルは LIN2.0 ライブラリとは切り離され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組み込む必要があります。
- **tmr\_drv36014.h (Ver.1.00)**  
H8/36014F における LIN 通信用 (スレーブ) タイマ W 機能設定、カウント制御を行なうタイマ W ドライバのヘッダファイルです。このファイルはユーザが使用する CPU 環境により自由に変更/改造することができます。このファイルは LIN2.0 ライブラリとは切り離され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組み込む必要があります。
- **Lin\_Drv36014.c (Ver.1.00)**  
H8/36014F における LIN 通信用 (スレーブ) 機能設定、通信制御を行なう LIN ドライバの C ソースファイルです。このファイルはユーザが使用する CPU 環境により自由に変更/改造することができます。このファイルは LIN2.0 ライブラリとは切り離され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組み込む必要があります。
- **Lin\_Drv36014.h (Ver.1.00)**  
H8/36014F における LIN 通信用 (スレーブ) 機能設定、通信制御を行なう LIN ドライバのヘッダファイルです。このファイルはユーザが使用する CPU 環境により自由に変更/改造することができます。このファイルは LIN2.0 ライブラリとは切り離され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組み込む必要があります。
- **Lin\_Slave\_Cnf.c (Ver.1.00)**  
スレーブノード専用の定義ファイルです。クラスタ内で扱われる信号、フレーム、スケジュールなどの定義がされています。このファイルはユーザが構築するクラスタ環境により生成しますが、通常はコンフィグレータを使用して生成します。
- **Lin\_Com\_Cnf.h (Ver.1.00)**  
スレーブノード定義ファイル (Lin\_Slave\_Cnf.c) にインクルードするヘッダファイルです。

- **lin20.h (Ver.1.00)**  
LIN2.0 ライブラリのヘッダファイルです。このヘッダファイルをユーザアプリケーションプログラムヘインクルードする必要があります。
- **lin20.lib (Ver.1.00)**  
LIN2.0 ライブラリ本体です。このファイルをユーザアプリケーションプログラムとリンクする必要があります。

## 2.2 ROM/RAM 使用容量

(H8S, H8/300 SERIES C/C++ COMPILER (V.6.00.03.000) 使用)

ROM/RAM 使用容量は、定義内容により変化するため、ここでは LIN2.0 ライブラリ (lin20.lib) の使用量のみ示します。

- **ROM 容量:** 8908 Byte
- **RAM 容量:** 155 Byte\*

【注】 ヒープ領域は含みません。「2.2.1 ヒープ領域」参照

### 2.2.1 ヒープ領域

LIN2.0 ライブラリでは初期化時にヒープ領域から動的にバッファを確保します。そのためアプリケーション開発時にはヒープ領域を用意して頂く必要があります。以下では最低限必要とするヒープ領域を示します。また、ヒープ領域を消費する項目を示します。

#### 1. 最低限必要なヒープ (RAM) 領域

- 未処理診断フレーム送信用 FIFO バッファ: 9 byte (1 段確保時)
  - 未処理診断フレーム受信用 FIFO バッファ: 9 byte (1 段確保時)
- 上記により、最低 18 byte のヒープ領域が必要です。

#### 2. ヒープ (RAM) 領域を消費する項目

- 未処理診断フレーム送信用 FIFO バッファ
- 未処理診断フレーム受信用 FIFO バッファ

上記はユーザがコンフィグレータで FIFO 段数を指定することができます。指定できる段数は送信用、受信用共に 1 ~ 65535 段です。

消費するヒープ領域の計算方法は以下となります。

計算式：送信用 (または受信用) FIFO 段数 × 9 byte (FIFO1 段消費数)

例：送信用 FIFO バッファ 30 段，受信用 FIFO バッファ 20 段確保時

【30 (段) × 9 (byte)】 + 【20 (段) × 9 (byte)】 = 450 (byte)

【注】 ヒープ領域が確保できない場合、LIN システム初期化時にエラーとなります。

## 2.3 API 関数

LIN2.0 ライブラリでスレーブノードが使用可能な API 関数について説明します。

図 7 に API 関数の説明形式を示します。

機能の概要を示します。	
ライブラリ関数の型 (リターン値および引数) を示します。	
説 明	ライブラリ関数の機能を説明します。
リターン値	正常：ライブラリ関数が正常終了したときの値です。 異常：ライブラリ関数が異常終了したときの値です。
引 数	引数の意味を説明します。
例	呼び出し手順を説明します。
備 考	補足説明, または使用上の注意事項です。

図 7 API 関数の説明形式

### 2.3.1 API 一覧

以下の表 2 にスレーブノードが使用可能な API 関数 (全 28 関数) の一覧を示します。

表 2 API 関数一覧

API 関数名	用 途
l_sys_init	LIN システム初期化
l_ifc_init	インタフェース初期化
l_ifc_ioctl	I/O ドライバ登録
l_ifc_connect	LIN バス接続
l_ifc_disconnect	LIN バス切断
l_flg_tst	フラグテスト
l_flg_clr	フラグクリア
l_bool_rd	1 bit 信号読み出し
l_u8_rd	2~8 bit 信号読み出し
l_u16_rd	9~16 bit 信号読み出し
l_bytes_rd	バイト配列信号読み出し
l_bool_wr	1 bit 信号書き込み
l_u8_wr	2~8 bit 信号書き込み
l_u16_wr	9~16 bit 信号書き込み
l_bytes_wr	バイト配列信号書き込み
l_ifc_wake_up	ウェイクアップ信号出力
l_ifc_tx	1 フレーム送信
l_ifc_rx	1 フレーム受信
l_ifc_aux	Break/Sync ヘッダ検出
l_ifc_read_status	ステータス取得
ld_put_raw	未処理診断フレーム送信
ld_get_raw	未処理診断フレーム取得
ld_raw_tx_status	未処理診断送信状態取得
ld_raw_rx_status	未処理診断受信状態取得
ld_send_message	処理済診断フレーム送信
ld_receive_message	処理済診断フレーム受信
ld_tx_status	処理済診断送信状態取得
ld_rx_status	処理済診断受信状態取得

2.3.2 コア API

システム初期化

---

**`l_bool l_sys_init(void)`**

---

説明	LIN システムの初期化をします。
リターン値	正常に初期化できたとき: 0 初期化に失敗したとき: 1
引数	なし
例	<code>l_bool ret</code> <code>ret = l_sys_init();</code>
備考	これ以降、説明する API 関数をコールするよりも先に当 API 関数をコールしてください。当 API はリセット時に一度だけコールします。

インタフェース初期化

---

**`void l_ifc_init(l_u8 ifc_name)`**

---

説明	LIN インタフェースの初期化をします。
リターン値	なし
引数	<code>ifc_name</code> インタフェース名
例	<code>ifc_init(0);</code>
備考	「 <code>l_sys_init</code> 」API 関数と「 <code>l_ifc_ioctl</code> 」API 関数コール後に、当 API 関数をコールしてください。 コール前に上記以外の API 関数をコールした場合の動作は不定です。 インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。



**I\_u16 I\_ifc\_ioctl(I\_u8 ifc\_name, I\_ioctl\_op op, void\* hand)**

説 明	各ノードで使用する I/O ドライバを登録します。
リターン値	全てのドライバが登録されたとき: 0 一部のドライバが登録されなかったとき: 登録されなかった数
引 数	ifc_name        インタフェース名 op              オペレーションコード hand            登録するドライバのハンドラ ポインタ
例	<pre> const T_Lib_Slave_Handle Slave_handle = {     Lin_Drv_Init,     Lin_Drv_BreakIn,     Lin_Drv_BreakInFinish,     Lin_Drv_BitInStart,     Lin_Drv_BitIn,     Lin_Drv_BitInEnd,     Lin_Drv_SyncInFinish,     Lin_Drv_Pid_RecvReq,     Lin_Drv_First_SendData,     Lin_Drv_SendData,     Lin_Drv_First_RecvReq,     Lin_Drv_RecvData,     Lin_Drv_SendRecvFinish,     Lin_Drv_LinBus_Enable,     Lin_Drv_LinBus_Disable,     Lin_Drv_WakeUp,     Lin_Drv_WakeUpFinish }; I_u16 ret; ret = I_ifc_ioctl( 0, LIN_ENTRY_SLAVE_DRV, &amp;Slave_handle ); </pre>
備 考	<p>インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。          オペレーションコードは以下の二つから指定してください。          マスタノードドライバ登録: LIN_ENTRY_MASTER_DRV          スレーブノードドライバ登録: LIN_ENTRY_SLAVE_DRV          当 API は「I_ifc_init」API 関数をコールする前に呼び出してください。</p>

---

**I\_bool I\_ifc\_connect( I\_u8 ifc\_name )**

---

説明	LIN バスと接続します。
リターン値	接続に成功したとき: 0 接続に失敗したとき: 1
引数	ifc_name      インタフェース名
例	<pre>I_bool ret; ret = I_ifc_connect( 0 ); if( ret ) {     /* Lin Bus Connect Failed */ }</pre>
備考	LIN 通信を行なう場合には当 API 関数で接続してからスケジュール実行をしてください。 インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。

---

**I\_bool I\_ifc\_disconnect( I\_u8 ifc\_name )**

---

説明	LIN バスを切断します。
リターン値	切断に成功したとき: 0 切断に失敗したとき: 1
引数	ifc_name      インタフェース名
例	<pre>I_bool ret; ret = I_ifc_disconnect( 0 ); if( ret ) {     /* Lin Bus Disconnect Failed */ }</pre>
備考	LIN 通信を終了する場合には当 API 関数で切断してください。 インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。

---

**I\_bool I\_flg\_tst( I\_flag\_handle flag\_name )**

---

説明	フラグのテストを行ないます。
リターン値	フラグの値: 0 or 1
引数	flag_name      フラグ名
例	<pre>I_bool ret; ret = I_flg_tst(&amp;Lin_Frm_FrameU1_flg); if( ret ) {     /* Something is done. */ } else {     /* Something is done. */ }</pre>
備考	インタフェース名は0しか設定できません。0でない値は設定しないでください。 フラグ名はユーザが定義したフラグの名前です。 定義したフラグのアドレスを代入します。

---

**void I\_flg\_clr( I\_flag\_handle flag\_name )**

---

説明	フラグのクリアを行ないます。
リターン値	なし
引数	flag_name      フラグ名
例	<pre>I_flg_clr(&amp;Lin_Frm_FrameU1_flg);</pre>
備考	インタフェース名は0しか設定できません。0でない値は設定しないでください。 フラグ名はユーザが定義したフラグの名前です。 定義したフラグのアドレスを代入します。

---

**`L_bool L_bool_rd( L_signal_handle sig_name )`**

---

説明	1 ビット信号の読み出しを行いません。
リターン値	信号値: 0 or 1
引数	sig_name      信号名
例	L_bool value; value = L_bool_rd( &Lin_Sig_Test0 );
備考	信号名はユーザが定義した信号の名前です。 定義した信号のアドレスを代入します。 1 bit 信号でない信号は当 API で読み出さないでください。 読み出した場合の動作は保証外です。

---

**`L_u8 L_u8_rd( L_signal_handle sig_name )`**

---

説明	2~8 ビット信号の読み出しを行いません。
リターン値	信号値: 0 ~ 255
引数	sig_name      信号名
例	L_u8 value; value = L_u8_rd( &Lin_Sig_Test3 );
備考	信号名はユーザが定義した信号の名前です。 定義した信号のアドレスを代入します。 2~8 ビットの信号でない信号は当 API で読み出さないでください。 読み出した場合の動作は保証外です。

---

**l\_u16 l\_u16\_rd( l\_signal\_handle sig\_name )**

---

説明	9 ~ 16 ビット信号の読み出しを行いません。
リターン値	信号値: 0 ~ 65535
引数	sig_name      信号名
例	l_u16 value; value = l_u16_rd( &Lin_Sig_Test7 );
備考	信号名はユーザが定義した信号の名前です。 定義した信号のアドレスを代入します。 9 ~ 16 ビットの信号でない信号は当 API で読み出さないでください。 読み出した場合の動作は保証外です。

---

**void l\_ifc\_init( l\_u8 ifc\_name )**

---

説明	バイト配列信号の読み出しを行いません。
リターン値	なし
引数	sig_name      信号名 start          読み出し開始バイト位置 count         読み出しバイト数 data          信号値格納用バッファ: 1 ~ 8 byte
例	l_u8 data[8]; l_bytes_rd( &Lin_Sig_Test13, 1, 2 );
備考	信号名はユーザが定義した信号の名前です。 定義した信号のアドレスを代入します。 バイト配列信号でない信号は当 API で読み出さないでください。 読み出した場合の動作は保証外です。 また、読み出しバイト数が定義した信号のサイズを超えないようにしてください。 エラー発生時には読み出しは行いません。このときのバッファの中身は不定となります。

---

**void l\_bool\_wr( l\_signal\_handle sig\_name, l\_bool sig )**

---

説明 1ビット信号の書き込みを行いません。  
リターン値 なし  
引数 sig\_name 信号名  
sig 信号値: 0 or 1  
例 l\_bytes\_wr( &Lin\_Sig\_Test1, 1 );  
備考 信号名はユーザが定義した信号の名前です。  
定義した信号のアドレスを代入します。  
1 bit 信号でない信号は当 API で書き込まないでください。  
書き込んだ場合の動作は保証外です。

---

**void l\_u8\_wr( l\_signal\_handle sig\_name, l\_u8 sig )**

---

説明 2~8ビット信号の書き込みを行いません。  
リターン値 なし  
引数 sig\_name 信号名  
sig 信号値: 0 ~ 255  
例 l\_u8\_wr( &Lin\_Sig\_Test4, 123 );  
備考 信号名はユーザが定義した信号の名前です。  
定義した信号のアドレスを代入します。  
2~8ビットの信号でない信号は当 API で書き込まないでください。  
書き込んだ場合の動作は保証外です。

---

**void l\_u16\_wr( l\_signal\_handle sig\_name, l\_u16 sig )**

---

説明 9～16 ビット信号の書き込みを行いません。

リターン値 なし

引数 sig\_name 信号名  
sig 信号値: 0～65535

例 l\_u16\_wr( &Lin\_Sig\_Test8, 12345 );

備考 信号名はユーザが定義した信号の名前です。  
定義した信号のアドレスを代入します。  
9～16 ビットの信号でない信号は当 API で書き込まないでください。  
書き込んだ場合の動作は保証外です。

---

**void l\_bytes\_wr( l\_signal\_handle sig\_name, l\_u8 start, l\_u8 count, const l\_u8\* const data )**

---

説明 バイト配列信号の書き込みを行いません。

リターン値 なし

引数 sig\_name 信号名  
start 書き込み開始バイト位置  
count 書き込みバイト数  
data 信号値格納用バッファ: 1～8 byte

例 l\_u8 data[8] = { 0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC, 0xDE, 0xF0 };  
l\_bytes\_wr( &Lin\_Sig\_Test15, 0, 8 );

備考 信号名はユーザが定義した信号の名前です。  
定義した信号のアドレスを代入します。  
バイト配列信号でない信号は当 API で書き込まないでください。  
書き込んだ場合の動作は保証外です。  
また、書き込みバイト数が定義した信号のサイズを超えないようにしてください。  
エラー発生時には書き込みは行いません。

---

**void l\_ifc\_wake\_up( l\_u8 ifc\_name )**

---

説明 ウェイクアップ信号の出力を行ないます。  
リターン値 なし  
引数 ifc\_name インタフェース名  
例 l\_ifc\_wake\_up( 0 );  
備考 インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。  
当 API コール時にウェイクアップ信号を出力します。

---

**void l\_ifc\_tx( l\_u8 ifc\_name )**

---

説明 フレームの送信を行ないます。  
リターン値 なし  
引数 ifc\_name インタフェース名  
例 void tx\_isr( void )  
{  
    l\_ifc\_tx( 0 );  
}  
備考 インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。  
通常、当 API はシリアル送信割り込み関数内でコールされます。  
コールされる場所は H/W 構成に依存します。



---

**void l\_ifc\_rx( l\_u8 ifc\_name )**

---

説明	フレームの受信を行いません。
リターン値	なし
引数	ifc_name      インタフェース名
例	<pre>vodi rx_isr( void ) {     l_ifc_rx( 0 ); }</pre>
備考	インタフェース名は0しか設定できません。0でない値は設定しないでください。 通常、当 API はシリアル受信割り込み関数内でコールされます。 コールされる場所は H/W 構成に依存します。

---

**void l\_ifc\_aux( l\_u8 ifc\_name )**

---

説明	ヘッダの検出を行いません。
リターン値	なし
引数	ifc_name      インタフェース名
例	<pre>vodi timer_isr( void ) {     l_ifc_aux( 0 ); }</pre>
備考	インタフェース名は0しか設定できません。0でない値は設定しないでください。 通常、当 API はインプットキャプチャ IRQ の受信割り込み関数内でコールされます。 コールされる場所は H/W 構成に依存します。

---

**L\_u16 l\_ifc\_read\_status( l\_u8 ifc\_name)**

---

説明	ステータス値の取得を行いません。
リターン値	ステータス値: 「3. 参考文献」参照
引数	ifc_name      インタフェース名
例	<pre>l_u16 status; status = l_ifc_read_status( 0 );</pre>
備考	インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。

### 2.3.3 診断フレーム用 API

未処理診断フレーム送信予約

---

#### void Id\_put\_raw( l\_u8 ifc\_name, const l\_u8\* const data )

---

説明	未処理診断フレームを送信 FIFO バッファへ予約します。
リターン値	なし
引数	ifc_name      インタフェース名 data            送信データバッファ
例	l_u8 data[8] = { 0x20u, 0x06u, 0xb1u, 0xffu, 0x7fu, 0x00u, 0x00u, 0x20u }; Id_put_raw( 0, data );
備考	インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。 当 API コール時には送信しません。次のマスタリクエストフレーム送信時に送信します。このときスリープコマンドまたはノード設定コマンドが予約されているときには、スリープコマンド、ノード設定コマンドを優先します。FIFO バッファに空きがない場合にはコールしても予約しません。戻り値がないためエラーチェックは行なっていません。エラーチェックは当 API コール側で行なってください。

未処理診断フレームデータ取得

---

#### void Id\_get\_raw( l\_u8 ifc\_name, l\_u8\* const data )

---

説明	未処理診断フレームのデータを FIFO バッファから取得します。
リターン値	なし
引数	ifc_name      インタフェース名 data            取得データ格納用バッファ
例	l_u8 data[8]; Id_get_raw ( 0, data );
備考	インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。 当 API コール時に FIFO バッファの古いものから順に 1 フレーム分取得します。FIFO バッファが空の場合にはコールしても取得しません。戻り値がないためエラーチェックは行なっていません。エラーチェックは当 API コール側で行なってください。

---

**I\_u8 ld\_raw\_tx\_status( I\_u8 ifc\_name )**

---

説明	未処理診断フレームの送信 FIFO バッファの状況を確認します。
リターン値	FIFO バッファに空きがない場合: LD_QUEUE_FULL FIFO バッファが空の場合: LD_QUEUE_EMPTY 転送エラー発生時: LD_TRANSFER_ERROR
引数	ifc_name      インタフェース名
例	I_u8    rtn; rtn = ld_raw_tx_status ( 0 );
備考	インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。

---

**I\_u8 ld\_raw\_rx\_status( I\_u8 ifc\_name )**

---

説明	未処理診断フレームの受信 FIFO バッファの状況を確認します。
リターン値	FIFO バッファにデータ有り: LD_DATA_AVAILABLE 転送エラー発生時: LD_TRANSFER_ERROR
引数	ifc_name      インタフェース名
例	I_u8    rtn; rtn = ld_raw_rx_status ( 0 );
備考	インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。

**void Id\_send\_message( l\_u8 ifc\_name, l\_u16 length, l\_u8 NAD, const l\_u8\* const data )**

説明	処理済み診断フレームの送信を予約します。	
リターン値	なし	
引数	ifc_name	インタフェース名
	length	送信データ長
	NAD	送信先ノードの NAD 値
	data	送信データバッファ
例	<pre>l_u8 data[5] = { 0x12, 0x34, 0x56, 0x78, 0x9A }; Id_send_message ( 0, 5, 0x23, data );</pre>	
備考	<p>インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。戻り値がないためエラーチェックは行なっていません。エラーチェックは当 API コール側で行なってください。当 API を送信完了前に再度コールした場合の動作は保証できません。</p>	

**void Id\_receive\_message( l\_u8 ifc\_name, l\_u16\* length, l\_u8\* NAD, l\_u8\* const data )**

説明	処理済み診断フレームの受信を予約します。	
リターン値	なし	
引数	ifc_name	インタフェース名
	length	受信データ長格納用バッファ
	NAD	送信元ノードの NAD 値
	data	受信データ格納用バッファ名
例	<pre>l_u8 data[100], nad; l_u16 length = 100; Id_receive_message ( 0, &amp;length, &amp;nad, data );</pre>	
備考	<p>インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。受信データ長は予約時に許容受信データ長を格納してから設定します。戻り値がないためエラーチェックは行なっていません。エラーチェックは当 API コール側で行なってください。当 API を受信完了前に再度コールした場合の動作は保証できません。</p>	

---

**I\_u8 Id\_tx\_status( I\_u8 ifc\_name )**

---

説明	処理済み診断フレームの送信状況を確認します。
リターン値	送信完了: LD_COMPLETED 送信中: LD_IN_PROGRESS 送信エラー発生: LD_FAILED
引数	ifc_name        インタフェース名
例	I_u8    rtn; rtn = Id_tx_status( 0 );
備考	インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。

---

**I\_u8 Id\_rx\_status( I\_u8 ifc\_name )**

---

説明	処理済み診断フレームの受信状況を確認します。
リターン値	受信完了: LD_COMPLETED 受信中: LD_IN_PROGRESS 受信エラー発生: LD_FAILED
引数	ifc_name        インタフェース名
例	I_u8    rtn; rtn = Id_rx_status( 0 );
備考	インタフェース名は 0 しか設定できません。0 でない値は設定しないでください。

## 2.4 LIN2.0 ライブラリ API 関数の使い方

以下ではLIN2.0 ライブラリ API 関数の使用例を示します。

### 2.4.1 LIN システムの初期化

LIN2.0 ライブラリ API 関数を使用する前に LIN システムの初期化を行ないます。

以下の例では、マイコンリセット時に LIN システムの初期化をしています。

**【注】** LIN の API コール箇所を反映します。

```
extern unsigned char lin_SomeCotrol_init( void );
__entry(vect=0) void PowerON_Reset(void)
{
    set_imask_ccr(1);

    _INITSCT();
// _CALL_INIT();           // Remove the comment to use global class object.
// _INIT_IOLIB();         // Remove the comment mark to use SIM I/O.
// errno=0;              // Remove the comment mark to use errno.
// srand(1);             // Remove the comment mark to use rand().
// _slptr=NULL;         // Remove the comment mark to use strtok().
    HardwareSetup();     // Remove the comment mark to use Hardware Setup.

    set_imask_ccr(0);

    /* .....Something to do */

    if( l_sys_init() ) {
        /* LIN System Initialize failed */
        sleep();
    }
    else {
        if( lin_SomeCotrol_init() ) {
            /* SomeSensor Initialize failed */
            sleep();
        }
    }

    /* .....Something to do */

    main();

// _CLOSEALL();           // Remove the comment mark to use SIM I/O.
// _CALL_END();          // Remove the comment mark to use global class object.
    sleep();
}
```

```
/* スレーブドライバエントリ用定義 */
const T_Lib_Slave_Handle Slave_handle = {
    Lin_Drv_Init,
    Lin_Drv_BreakIn,
    Lin_Drv_BreakInFinish,
    Lin_Drv_BitInStart,
    Lin_Drv_BitIn,
    Lin_Drv_BitInEnd,
    Lin_Drv_SyncInFinish,
    Lin_Drv_Pid_RecvReq,
    Lin_Drv_First_SendData,
    Lin_Drv_SendData,
    Lin_Drv_First_RecvReq,
    Lin_Drv_RecvData,
    Lin_Drv_SendRecvFinish,
    Lin_Drv_LinBus_Enable,
    Lin_Drv_LinBus_Disable,
    Lin_Drv_WakeUp,
    Lin_Drv_WakeUpFinish
};

/* Cluster Initialize */
unsigned char lin_SomeCotrol_init( void )
{
    unsigned char rtn;

    rtn = 0;
    if( l_ifc_ioctl( 0, LIN_ENTRY_SLAVE_DRV, &Slave_handle ) ) {
        /* The init of the LIN Slave Driver failed */
        rtn = lu;
    }
    else {
        l_ifc_init(0); /* Interface Initialize */
        if( l_ifc_connect(0) ) {
            /* Connection of the LIN interface failed */
            rtn = lu;
        }
        else {
            /* .....Something to do */
        }
    }
    return rtn;
}
```



## 2.4.2 アプリケーション

ここでは初期化、スケジュール実行以外にアプリケーションからコールされる LIN2.0 ライブラリ API 関数についてのサンプルコードを示します。API 関数コールによって取得したデータの使用方法はアプリケーションに依存するためここでは特に記載しません。LIN バス上で送受信されるデータ (フレーム) の内容は各ノードの状態や周辺デバイスなどから取得したデータです。どのようなデータを送受信するか、データをどのように処理するかは LIN のシステム構成によります。

```
#include "36014s.h"
#include "Lin_Drv36014.h"
#include "lin20.h"
void lin_application( void );
/*****/
/* メイン関数 */
/*****/
void main(void)
{
    while( 1 ) {
        /* .....Something to do */

        lin_application();

        /* .....Something to do */
    }
}
```

```

/*****
/* LIN アプリケーション関数      */
/*****
extern l_flg Lin_Frm_FrameMst0_flg;
extern l_flg Lin_Frm_FrameU1_flg;
extern l_flg Lin_Frm_FrameU2_flg;
extern l_flg Lin_Frm_FrameU3_flg;
extern l_flg Lin_Frm_FrameEve0_flg;
extern l_flg Lin_Frm_FrameSlv0_flg;
extern l_flg Lin_Sig_Command_flg;
extern T_Signal Lin_Sig_Status_Slv0;
extern T_Signal Lin_Sig_Status_Slv1;
extern T_Signal Lin_Sig_Command;
void lin_application( void )
{
    l_u8  data[8];
    union {
        l_u16 Word;
        struct {
            l_u16 lastpid      :8;
            l_u16              :4;
            l_u16 gotosleep    :1;
            l_u16 overrun      :1;
            l_u16 txsucce     :1;
            l_u16 errorrsp     :1;
        } Bit;
    } status;

    /* スレーブノードが受信するフレーム */
    if( 0 != l_flg_tst(&Lin_Frm_FrameU1_flg) ) {
        l_flg_clr( &Lin_Frm_FrameU1_flg );
        /* Something is done */
    }
    else if( 0 != l_flg_tst(&Lin_Frm_FrameMst0_flg) ) {
        l_flg_clr( &Lin_Frm_FrameMst0_flg );
        /* Something is done */
    }

    /* スレーブノードが受信するフレーム */
    if( 0 != l_flg_tst(&Lin_Frm_FrameU1_flg) ) {
        l_flg_clr( &Lin_Frm_FrameU1_flg );
        /* Something is done */
    }
    else if( 0 != l_flg_tst(&Lin_Frm_FrameMst0_flg) ) {
        l_flg_clr( &Lin_Frm_FrameMst0_flg );
        /* Something is done */
    }
}

```

```

    /* スレーブノードが送信するフレーム */
    if( 0 != l_flg_tst(&Lin_Frm_FrameU2_flg) ) {
        l_flg_clr( &Lin_Frm_FrameU2_flg );
        /* Something is done */
    }
    /* スレーブノードが送信するフレーム */
    else if( 0 != l_flg_tst(&Lin_Frm_FrameU3_flg) ) {
        l_flg_clr( &Lin_Frm_FrameU3_flg );
        /* Something is done */
    }
    else if( 0 != l_flg_tst(&Lin_Frm_FrameEve0_flg) ) {
        l_flg_clr( &Lin_Frm_FrameEve0_flg );
        /* Something is done */
    }
    else if( 0 != l_flg_tst(&Lin_Frm_FrameSlv0_flg) ) {
        l_flg_clr( &Lin_Frm_FrameSlv0_flg );
        /* Something is done */
    }
}

status.Word = l_ifc_read_status( 0 );

if( status.Bit.errorrsp ) {
    /* Something Error Response Processing */
}

if( LD_DATA_AVAILABLE == ld_raw_rx_status(0) ) {
    ld_get_raw( 0, data );
}

/* マスタノードから伝えられる信号 */
if( 0 != l_flg_tst(&Lin_Sig_Command_flg) ) {
    l_flg_clr( &Lin_Sig_Command_flg );
    if( 0x1234u == l_ul6_rd(&Lin_Sig_Command) ) {
        l_ul6_wr( &Lin_Sig_Status_Slv0, 0x0101u );
        l_ul6_wr( &Lin_Sig_Status_Slv1, 0x0201u );
        /* Something is done */
    }
    else if( 0x5678u == l_ul6_rd(&Lin_Sig_Command) ) {
        l_ul6_wr( &Lin_Sig_Status_Slv0, 0x0100u );
        l_ul6_wr( &Lin_Sig_Status_Slv1, 0x0200u );
        /* Something is done */
    }
}

if( status.Bit.gotosleep ) {
    /* Something Sleepmode Processing */
}

return;
}

```

### 3. 参考文献

- LIN Specification Package Revision 2.0 <http://www.lin-subbus.org>
- LIN Protocol Specification Revision 2.0 <http://www.lin-subbus.org>
- LIN Diagnostic and Configuration Specification Revision 2.0 <http://www.lin-subbus.org>
- LIN Application Program Interface Specification Revision 2.0 <http://www.lin-subbus.org>
- LIN Physical Layer Specification Revision 2.0 <http://www.lin-subbus.org>
- H8/36024 グループ, H8/36014 グループ ハードウェアマニュアル RJJ09B0153-0400

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2006.01.31	—	初版発行

### 安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

### 本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ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ルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾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の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 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全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりますは、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ(<http://www.renesas.com>)などを通じて公開される情報に常にご注意ください。
4. 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。万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位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ください。ルネサス テクノロジは、適用可否に対する責任を負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際は、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。