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。

R8C/Tiny シリーズ
家電向け一線式通信プロトコル

要旨

家電向け一線式通信（以下一線式通信と称す）プロトコルは，R8C/Tiny シリーズの内蔵機能を使用し，LIN（Local Interconnect Network）通信のメッセージフレーム仕様に準拠した通信プロトコルであり，製品内でのチップ間通信のハードウエア設計およびソフトウエア設計の際，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

R8C/Tiny シリーズ　－R8C/18，R8C/24－

目次

1. 一線式通信システム概要

本アプリケーションノートに掲載している一線式通信用ソフトウエ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一線式通信の概要を示す。

1.1 シングルワイヤバスへの接続

本ライブラリはシングルワイヤ（一線式）のバス接続形態での接続を前提に構成されており，シングルワイヤでの接続であれば，特にその形態を規定するものではない。シングルワイヤによりネットワーク接続されたシステムに任意のバスインタフェース回路（またはトランシーバ回路）を介して接続することにより，メッセージフレームの送受信などの一線式通信を行います。

1.1.1 システム構成例

図１にシングルワイヤバスネットワークシステム構成例を示します。

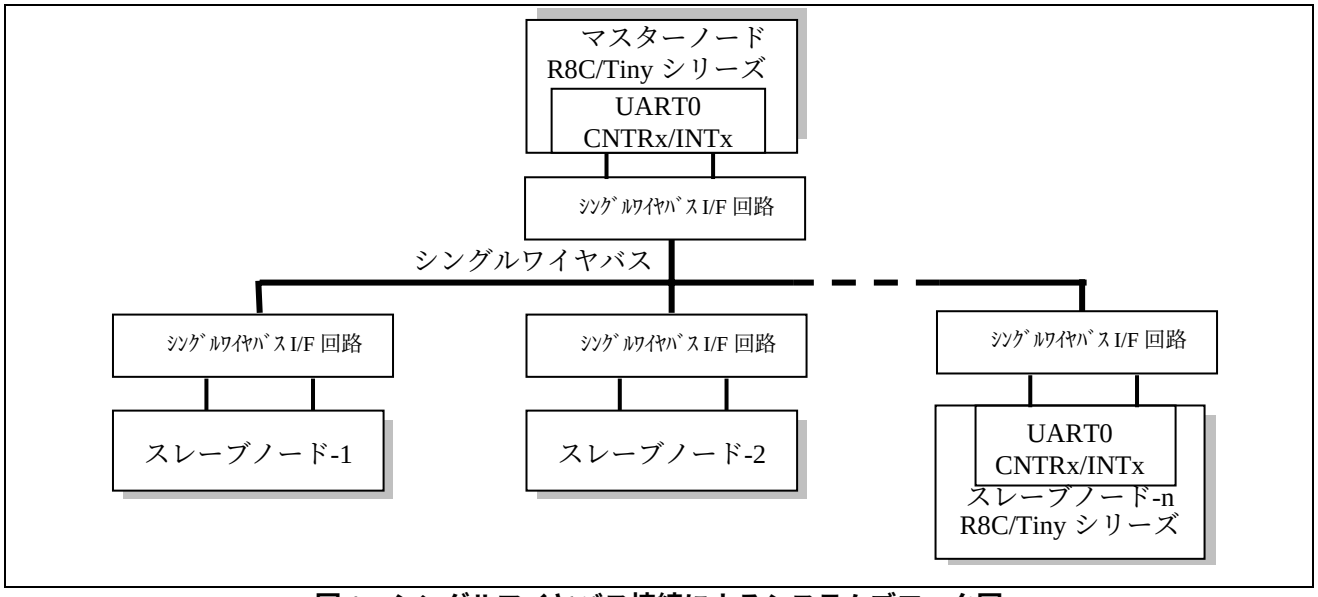


図１　シングルワイヤバス接続によるシステムブロック図

1.1.2 一線式バスインタフェース

図２に R8C/Tiny シリーズマイコン（以下マイコンと称す）内蔵機能の入出力端子とシングルワイヤバスとのインタフェース回路例を示します。

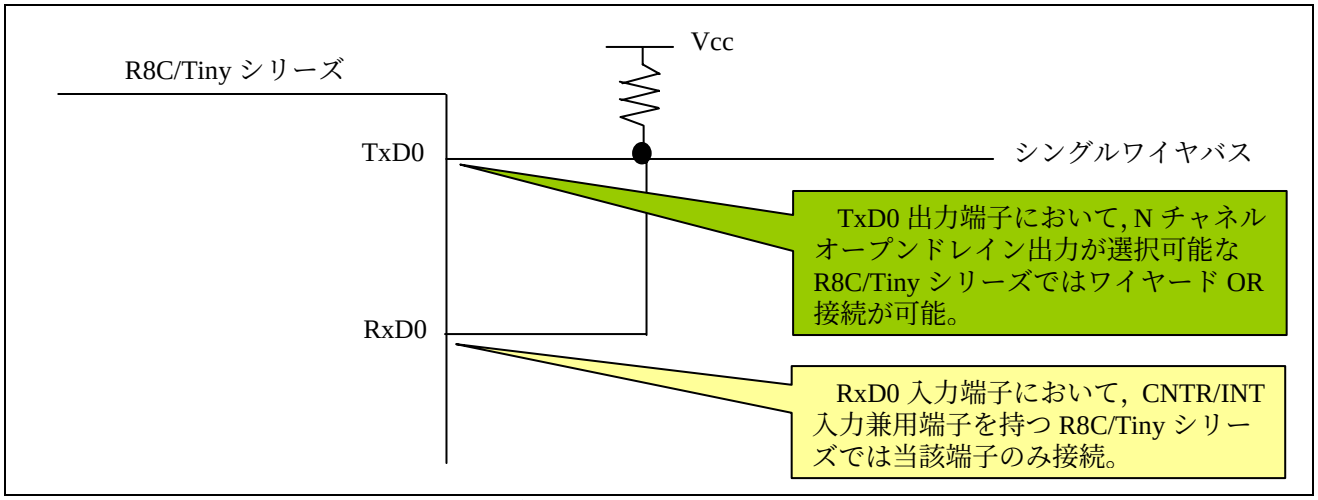


図２　シングルワイヤバスインタフェース回路例

1.2 一線式通信概要

一線式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，マスター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。

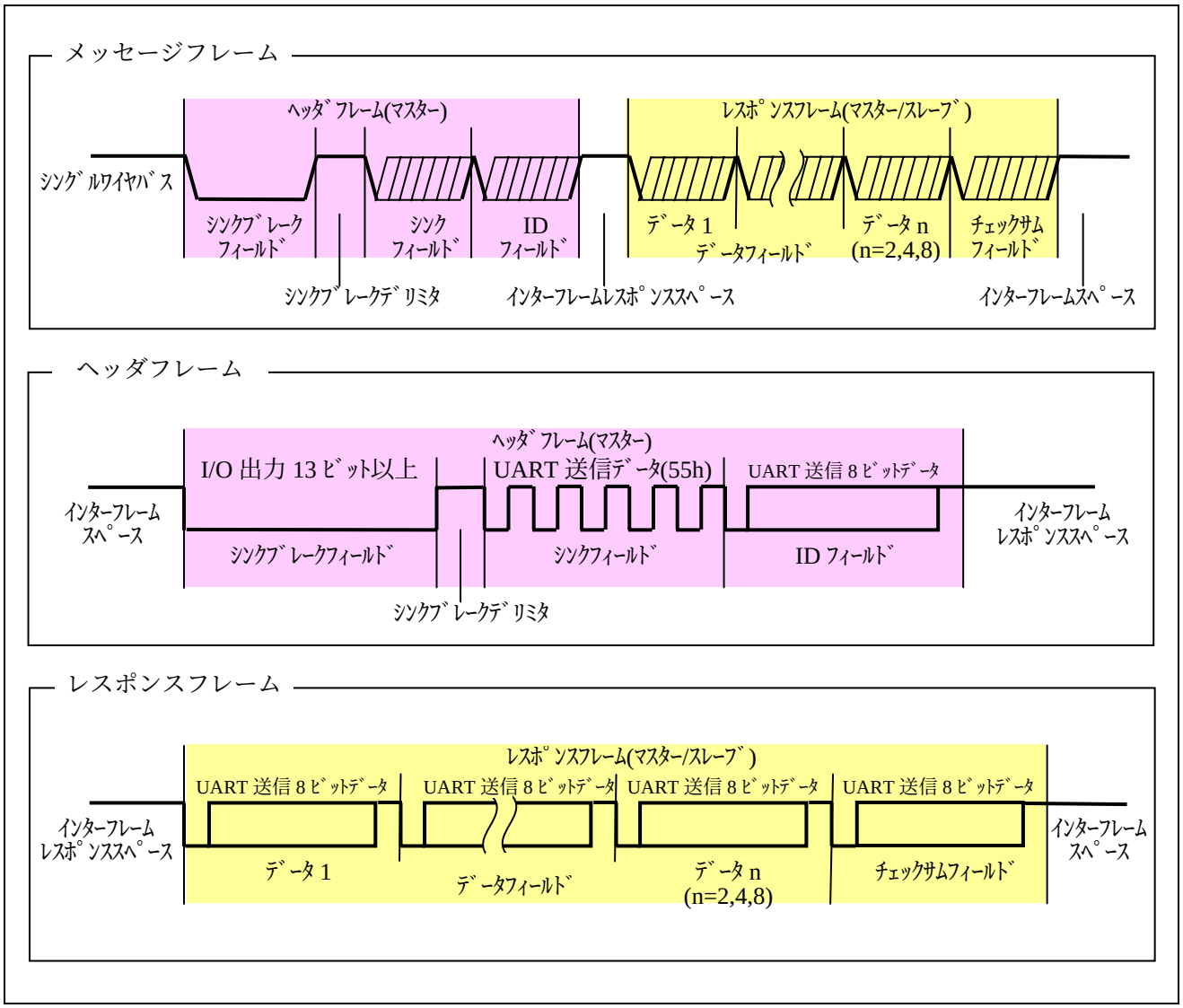


図３　メッセージフレームの構成

1.2.2 メッセージフレーム送受信

図 4 にマスター／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。

－マスターノードがヘッダー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。（マスターノードは送信時に ID を判定）

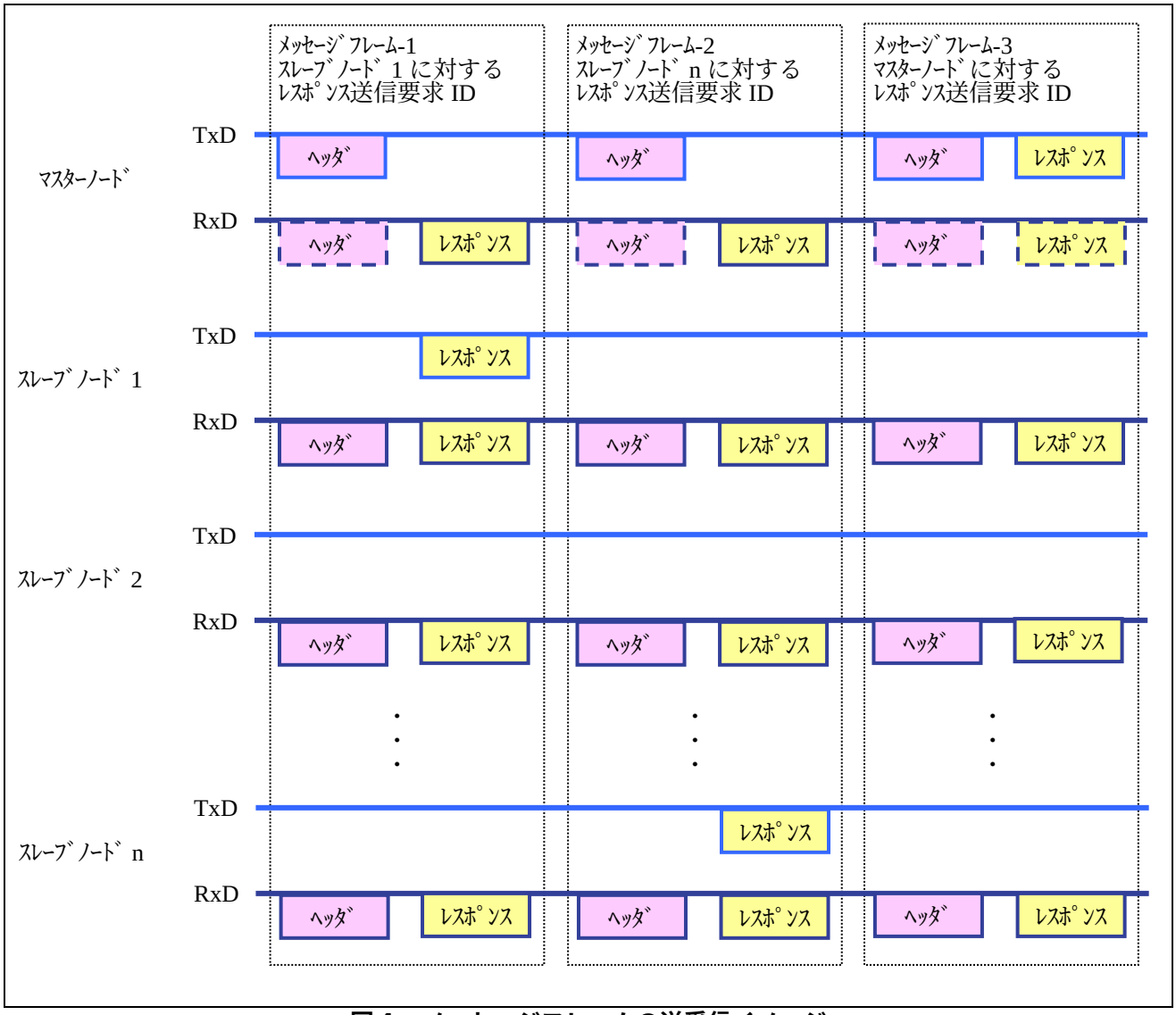


図４　メッセージフレームの送受信イメージ

2. ライブラリソフトウエア仕様マスターノード編

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，R8C/Tiny の内蔵機能を使用し，マスターノードとして一線式通信を行います。

2.1 動作環境

- 使用デバイス：R8C/24 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，一線式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，HASW_MST.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　R8C/24 内蔵周辺機能使用表

機能		端子機能（端子番号）	用途	説明
UART0	送信	TxD0(24)	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	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	RxD0(23)	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
ﾊｰﾄﾞｳｴｱ LIN ﾀｲﾏ RA		TxD0(24)	シンクブレークフィールド送信 シンクブレークデリミタ送信	TxD0 出力によりシンクブレークフィールドを形成 Φosc 周期でカウント動作し，各時間間隔を計測

2.2 ファイル構成

- HASW_MST.c（Ver1.00）

  R8C/24 グループにおける一線式通信（マスター）マイコン機能設定，および通信制御を行うための C ソースファイルです。

- HASW_MST.h（Ver1.00）

  ユーザーにより通信転送速度の設定や ID の設定などを行い HASW_MST .c（Ver1.00）コンパイル時に組み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み込む必要があります。

- SFR_R825.h

  R8C/24 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

（R8C シリーズ C コンパイラ M3T-NC30WAVer5.40R0 使用時）

ROM/RAM 使用容量は，ID 設定数などにより変化します。

- ROM 容量：約 1KByte(オプティマイズ無し)，約 0.8KByte(オプティマイズ有り：ROM サイズ重視)
- RAM 容量：37Byte

2.4 機能仕様

2.4.1 一線式通信仕様

- ノード：マスターノードに対応
- ID：ユーザ定義 ID

レスポンス送信 ID
HASW_MST.h より 0 個から 60 個（00h～3ch）まで設定可能。
（但し，同一シングルワイヤバス上に同じ ID を受け付けるノードを設定すると正常動作しません）
一線式通信プロトコル定義 ID

a) マスターリクエストフレーム ID3ch（ID フィールドデータ：3Ch）
自動的にマスター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。

b) スレーブレスポンスフレーム ID3dh（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 ID3eh，3fh（ID フィールドデータ：FEh，BFh）
本ライブラリ（Ver1.00）では，対応しておりません。

ID 設定方法
HASW_MST.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 HASW_MST.c をコンパイルします。

- レスポンスデータ長：送信する 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。

ID フィールドデータ = 00h～1fh ： ２バイト
= 20h～2fh ： ４バイト
= 30h～3Dh ： ８バイト

- 通信転送速度：HASW_MST.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 UART0 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。詳しくはハードウエアマニュアルのクロック非同期式シリアル I/O（UART0）モード：ビットレートに対する U0BRG 設定例（UART モード）をご参考ください。

なお，φosc（20MHz，8MHz）および通信速度（19200bps，9600bps，2400bps）以外の組み合わせを使用される場合は，初期設定処理関数内にて通信速度の自動計算を行いますので，初期設定処理関数の処理時間がかかります。

- ウェイクアップ信号送受信：本ライブラリ（Ver1.00）では，対応しておりません。

2.4.2 HASW_MST.h ファイル設定例

HASW_MST.h の設定例を示します。

a) 使用するマイコンは R8C/24 グループとする。

b) 以下の 4 個の ID に対してレスポンスフレームを送信する。

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

c) システム搭載クロック（φosc）を 8[MHz](内蔵 OSC)とする。

d) 一線式通信転送速度を 19200[bit/sec]とする。

上記 a)～d)の仕様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下さい）

```
/*""FILE COMMENT""*********************************************************************
 * File Name    : hasw_mst.h
 * Version      : 1.00
 * Contents     : 一線式通信ユーザ定義ファイル
*""FILE COMMENT END""*********************************************************************/
#ifdef REAL_MEM                                 /* この行は変更または削除しないでください */
    #define     GLOBAL                        /* この行は変更または削除しないでください */
#else                                           /* この行は変更または削除しないでください */
    #define     GLOBAL      extern            /* この行は変更または削除しないでください */
#endif                                          /* この行は変更または削除しないでください */
#define     SBDATA_ON                   /* SB 相対アドレッシングを使用する場合はこの行を定義 */
/*****************************************************************************************/
/*                                                                                       */
/*     CPU 選択                                                                          */
/*                                                                                       */
/*=======================================================================================*/
#define     __CPU_R8C_24                            /* R8C/24 グループの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           /* 2 バイトレスポンスデータ送信 ID を持つ場合はこの行を定義 */
#ifdef      __Res2Byte_ID
//      #define     __ID00      0x80            /*                                       */
    #define     __ID01      0xC1            /* ID フィールド C1h に対してレスポンスを送信 */
//      #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    __ID10    0xCA              /*                                  */
//    #define    __ID11    0x8B              /*                                  */
//    #define    __ID12    0x4C              /*                                  */
//    #define    __ID13    0x0D              /*                                  */
//    #define    __ID14    0x8E              /*                                  */
//    #define    __ID15    0xCF              /*                                  */
//    #define    __ID16    0x50              /*                                  */
//    #define    __ID17    0x11              /*                                  */
    #define    __ID18    0x92              /* ID フィールド 92h に対してレスポンスを送信 */
//    #define    __ID19    0xD3              /*                                  */
//    #define    __ID20    0x14              /*                                  */
//    #define    __ID21    0x55              /*                                  */
//    #define    __ID22    0xD6              /*                                  */
//    #define    __ID23    0x97              /*                                  */
//    #define    __ID24    0xD8              /*                                  */
//    #define    __ID25    0x99              /*                                  */
//    #define    __ID26    0x1A              /*                                  */
//    #define    __ID27    0x5B              /*                                  */
//    #define    __ID28    0x9C              /*                                  */
//    #define    __ID29    0xDD              /*                                  */
//    #define    __ID30    0x5E              /*                                  */
//    #define    __ID31    0x1F              /*                                  */
#endif

/*==================================================================================*/
/*      4 バイトデータ                                                              */
/*----------------------------------------------------------------------------------*/
#define    __Res4Byte_ID          /* 4 バイトレスポンスデータ送信 ID を持つ場合はこの行を定義 */
#ifdef     __Res4Byte_ID
//    #define    __ID32    0x20              /*                                  */
//    #define    __ID33    0x61              /*                                  */
//    #define    __ID34    0xE2              /*                                  */
    #define    __ID35    0xA3              /* ID フィールド A3h に対してレスポンスを送信 */
//    #define    __ID36    0x64              /*                                  */
//    #define    __ID37    0x25              /*                                  */
//    #define    __ID38    0xA6              /*                                  */
//    #define    __ID39    0xE7              /*                                  */
//    #define    __ID40    0xA8              /*                                  */
//    #define    __ID41    0xE9              /*                                  */
//    #define    __ID42    0x6A              /*                                  */
//    #define    __ID43    0x2B              /*                                  */
//    #define    __ID44    0xEC              /*                                  */
//    #define    __ID45    0xAD              /*                                  */
//    #define    __ID46    0x2E              /*                                  */
//    #define    __ID47    0x6F              /*                                  */
#endif
```

```
/*=========================================================================================*/
/*    8バイトデータ                                                                        */
/*-----------------------------------------------------------------------------------------*/
#define     __Res8Byte_ID          /*8バイトレスポンスデータ送信IDを持つ場合はこの行を定義 */
#ifdef      __Res8Byte_ID
//    #define     __ID48     0xF0              /*                                        */
//    #define     __ID49     0xB1              /*                                        */
//    #define     __ID50     0x32              /*                                        */
//    #define     __ID51     0x73              /*                                        */
    #define     __ID52     0xB4              /*IDフィールドF5hに対してレスポンスを送信 */
//    #define     __ID53     0xF5              /*                                        */
//    #define     __ID54     0x76              /*                                        */
//    #define     __ID55     0x37              /*                                        */
//    #define     __ID56     0x78              /*                                        */
//    #define     __ID57     0x39              /*                                        */
//    #define     __ID58     0xBA              /*                                        */
//    #define     __ID59     0xFB              /*                                        */
    #define     __ID60     0x3C              /*IDフィールド3Chに対してレスポンスを送信 */
#endif
/*****************************************************************************************/
/*                                                                                       */
/*      システム搭載クロック定義                                                         */
/*                                                                                       */
/*=========================================================================================*/
//#define     __OSC_Hz     20000000                    /* φosc = 20.000MHz → 20000000 */
#define     __OSC_Hz      8000000                    /* φosc =  8.000MHz →  8000000 */
//#define     __OVER_10M_Hz                  /* φosc >= 10.000MHz の場合はこの行を定義*/
/*****************************************************************************************/
/*                                                                                       */
/*      通信転送速度ボーレート定義                                                       */
/*                                                                                       */
/*=========================================================================================*/
//#define     __B_rate      2400                          /*  2400bps →  2400 */
//#define     __B_rate      9600                          /*  9600bps →  9600 */
#define     __B_rate     19200                          /* 19200bps → 19200 */
/*****************************************************************************************/
/*                                                                                       */
/*       関数・変数定義          以下は変更または削除しないでください                    */
/*                                                                                       */
/*=========================================================================================*/
/*      レスポンスデータ有無確認定義                                                     */
/*-----------------------------------------------------------------------------------------*/
#ifndef     __RESPONSE
    #ifdef      __Res2Byte_ID
        #define     __RESPONSE
    #endif
#endif
#ifndef     __RESPONSE
    #ifdef      __Res4Byte_ID
        #define     __RESPONSE
```

```
    #endif
#endif
#ifndef    __RESPONSE
    #ifdef    __Res8Byte_ID
        #define    __RESPONSE
    #endif
#endif

/*=========================================================================================*/
/*     関数定義                                                                            */
/*-----------------------------------------------------------------------------------------*/
GLOBAL    void    l_sys_init_HASW_MST(void);
GLOBAL    void    l_ifc_init_HASW_MST(void);
GLOBAL    void    l_ifc_connect_HASW_MST(void);
GLOBAL    void    l_ifc_disconnect_HASW_MST(void);
#ifdef REAL_MEM
    extern    void    l_ifc_rx_HASW_MST(void);
#else
    void    l_ifc_rx_HASW_MST(void);
#endif

#ifdef    __RESPONSE
    #ifdef REAL_MEM
        extern    void    l_ifc_tx_HASW_MST(void);
    #else
        void    l_ifc_tx_HASW_MST(void);
    #endif
#endif

/*=========================================================================================*/
/*     変数定義                                                                            */
/*-----------------------------------------------------------------------------------------*/
#ifdef    __RESPONSE
    GLOBAL volatile unsigned char    uc_bHASW_tx_data[8];
    #ifdef    SBDATA_ON
        #pragma  SBDATA  uc_bHASW_tx_data
    #endif
#endif
GLOBAL volatile unsigned char    uc_dHASW_tx_id;
GLOBAL volatile unsigned char    uc_dHASW_rx_id;
GLOBAL volatile unsigned char    uc_bHASW_rx_data[8];
GLOBAL volatile union {
    struct {
        unsigned char    Bit0:1;
        unsigned char    Bit1:1;
        unsigned char    Bit2:1;
        unsigned char    Bit3:1;
        unsigned char    Bit4:1;
        unsigned char    Bit5:1;
        unsigned char    Bit6:1;
```

```
        unsigned char    Bit7:1;
        unsigned char    Bit8:1;
        unsigned char    Bit9:1;
        unsigned char    Bit10:1;
        unsigned char    Bit11:1;
        unsigned char    Bit12:1;
        unsigned char    Bit13:1;
        unsigned char    Bit14:1;
        unsigned char    Bit15:1;
    }StBIT;
    struct {
        unsigned char    Byte0;
        unsigned char    Byte1;
    }StBYTE;
    unsigned int uiAll;
}ui_sHASW_status;
#ifdef    SBDATA_ON
    #pragma  SBDATA  uc_dHASW_tx_id
    #pragma  SBDATA  uc_dHASW_rx_id
    #pragma  SBDATA  uc_bHASW_rx_data
    #pragma  SBDATA  ui_sHASW_status
#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，一線式通信制御に必要なマイコン（R8C/24 グループ）内蔵周辺機能の初期化，一線式通信制御の停止・再開，ヘッダフレーム送信を行います。

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
l_sys_init_HASW_MST	**一線式通信制御に必要なマイコン（R8C/24 グループ）内蔵周辺機能の初期設定を行います。**
l_ifc_init_HASW_MST	**一線式通信制御を初期状態にします。** **・通信エラーなどの発生時に一線式通信制御を初期状態に戻したいときに使用します。**
l_ifc_connect_HASW_MST	**一線式通信制御を開始状態にし，ヘッダフレームの送信を開始します。**
l_ifc_disconnect_HASW_MST	**一線式通信制御を停止状態にします。** **・レスポンスフレーム送受信後に使用します。**

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，一線式通信中の各タイミング（送受信完了時，通信エラー検出時など）での処理を行います。

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
l_ifc_rx_HASW_MST	**レスポンスフレーム送受信後のユーザアプリケーション制御関数** **・レスポンスフレーム受信後，レスポンスフレーム送信後，エラー発生時に呼び出されます。イベントの内容については通信ステータスフラグで判定してください。**
l_ifc_tx_HASW_MST	**ヘッダフレーム送信後のユーザアプリケーション制御関数** **・レスポンスフレームを送信する ID を使用した場合，ヘッダフレーム送信後に呼び出されます。**

- 動作概要

  図５，図６に動作概要を示します。

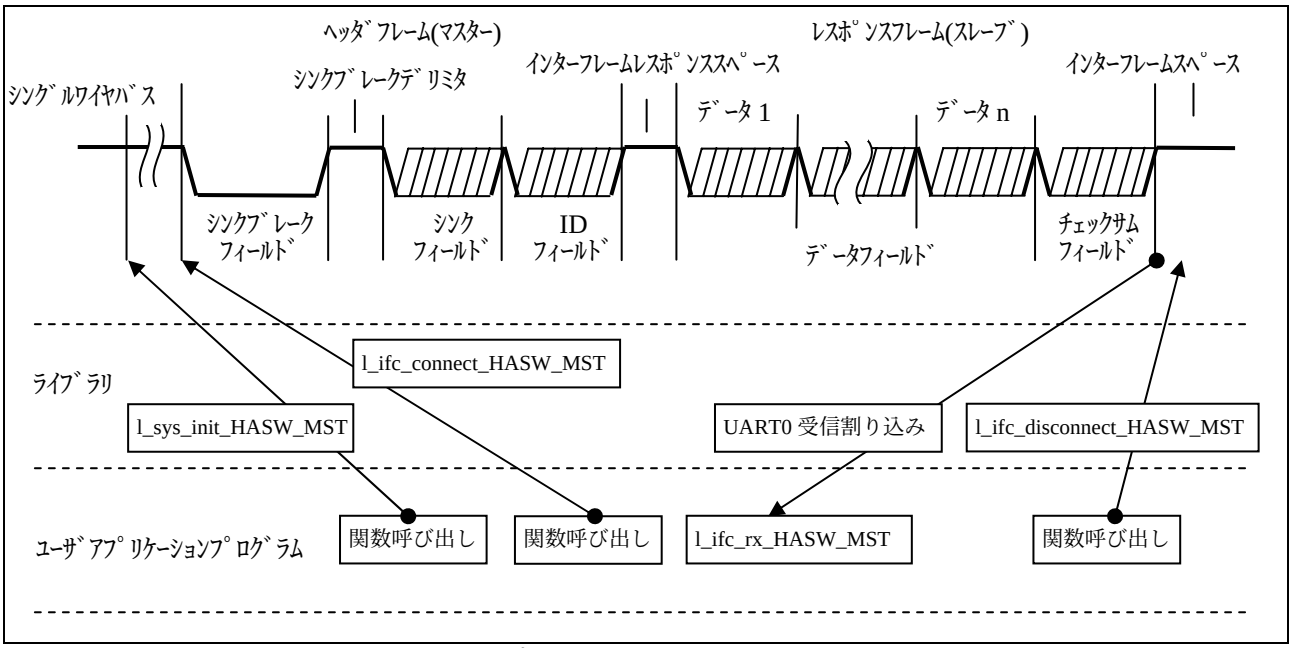


図５　レスポンスフレーム受信時の動作概要

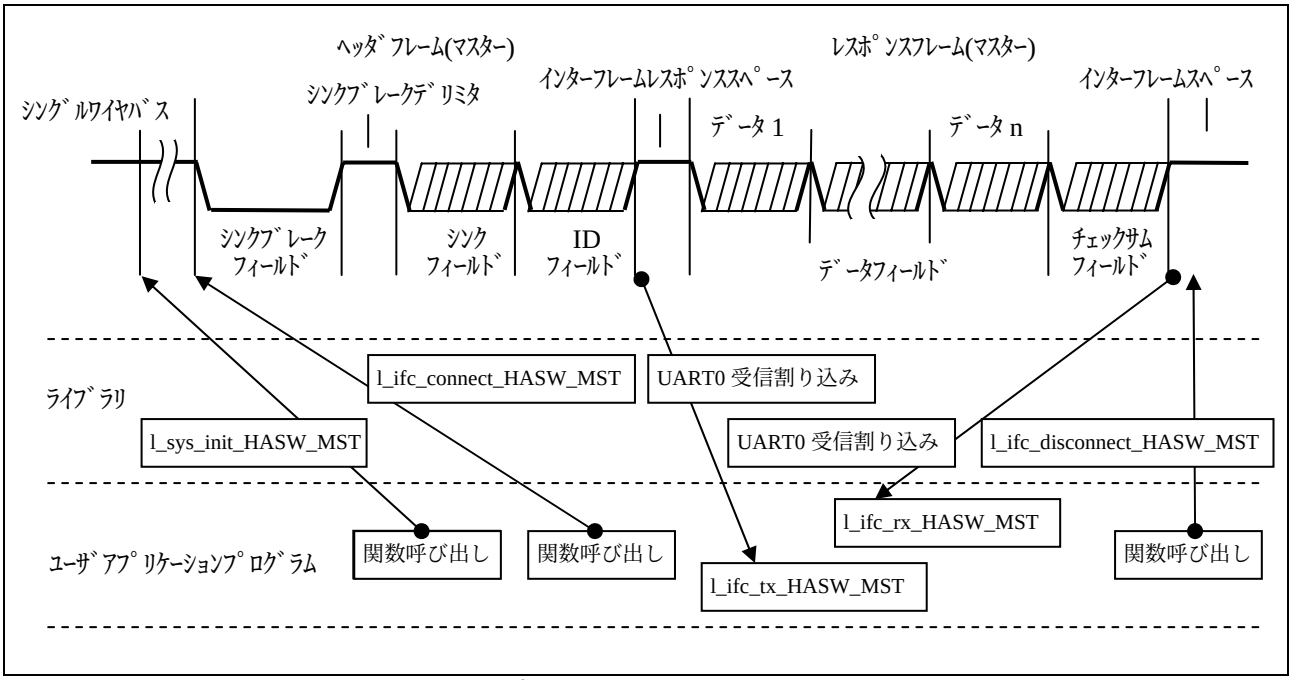


図６　レスポンスフレーム送信時の動作概要

- 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と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
uc_dHASW_tx_id	unsigned char	ヘッダーフレーム送信時の送信 ID を設定(ID ビット＋DLC ビット)を設定（表５　ID 一覧表参照）
uc_bHASW_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
uc_dHASW_rx_id	unsigned char	レスポンスデータ受信時の ID を格納
uc_bHASW_rx_data[0]〜[7]	unsigned char (配列)	受信したレスポンスデータを格納
ui_sHASW_status	（構造体）	通信ステータス
ui_sHASW_status.uiAll	ワードアクセス unsigned int	
	ビットアクセス	
ui_sHASW_status.StBIT.Bit15	ビット-15	ビットエラーフラグ
ui_sHASW_status.StBIT.Bit14	ビット-14	フレーミングエラーフラグ
ui_sHASW_status.StBIT.Bit13	ビット-13	オーバーランエラーフラグ
ui_sHASW_status.StBIT.Bit12	ビット-12	パリティエラーフラグ
ui_sHASW_status.StBIT.Bit11	ビット-11	リザーブビット
ui_sHASW_status.StBIT.Bit10	ビット-10	チェックサムエラーフラグ
ui_sHASW_status.StBIT.Bit9	ビット-9	送受信エラー発生フラグ
ui_sHASW_status.StBIT.Bit8	ビット-8	メッセージフレーム正常受信完了フラグ
ui_sHASW_status.StBIT.Bit7	ビット-7	UART0，タイマ RA 割り込みレベル設定ビット UART0，タイマ RA 割り込みのレベル 000〜111 に設定する。
ui_sHASW_status.StBIT.Bit6	ビット-6	
ui_sHASW_status.StBIT.Bit5	ビット-5	
ui_sHASW_status.StBIT.Bit4	ビット-4	リザーブビット
ui_sHASW_status.StBIT.Bit3	ビット-3	シンクブレークデリミタビット長設定ビット 00：1 ビット長 01：2 ビット長 10：3 ビット長 11：4 ビット長
ui_sHASW_status.StBIT.Bit2	ビット-2	
ui_sHASW_status.StBIT.Bit1	ビット-1	リザーブビット
ui_sHASW_status.StBIT.Bit0	ビット-0	ヘッダーフレーム送信許可フラグ

表 5　ID 一覧表

uc_dHASW_tx_id 設定値		ID ﾌｨｰﾙﾄﾞ送信ﾃﾞｰﾀ		ﾚｽﾎﾟﾝｽﾃﾞｰﾀ長	uc_dHASW_tx_id 設定値		ID ﾌｨｰﾙﾄﾞ送信ﾃﾞｰﾀ		ﾚｽﾎﾟﾝｽﾃﾞｰﾀ長
10 進	16 進	10 進	16 進		10 進	16 進	10 進	16 進	
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	213	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 内蔵周辺機能の初期設定

一線式通信の実行を行うことに先立って，ユーザアプリケーションプログラムから l_sys_init_HASW_MST 関数を呼び出す事により，マイコンの内蔵周辺機能の初期化を行います。

l_sys_init_HASW_MST 関数をユーザアプリケーションプログラムから呼び出す時には，予め割り込みレベル設定（ui_sHASW_status.StBIT.Bit7～ui_sHASW_status.StBIT.Bit5）を設定してください。

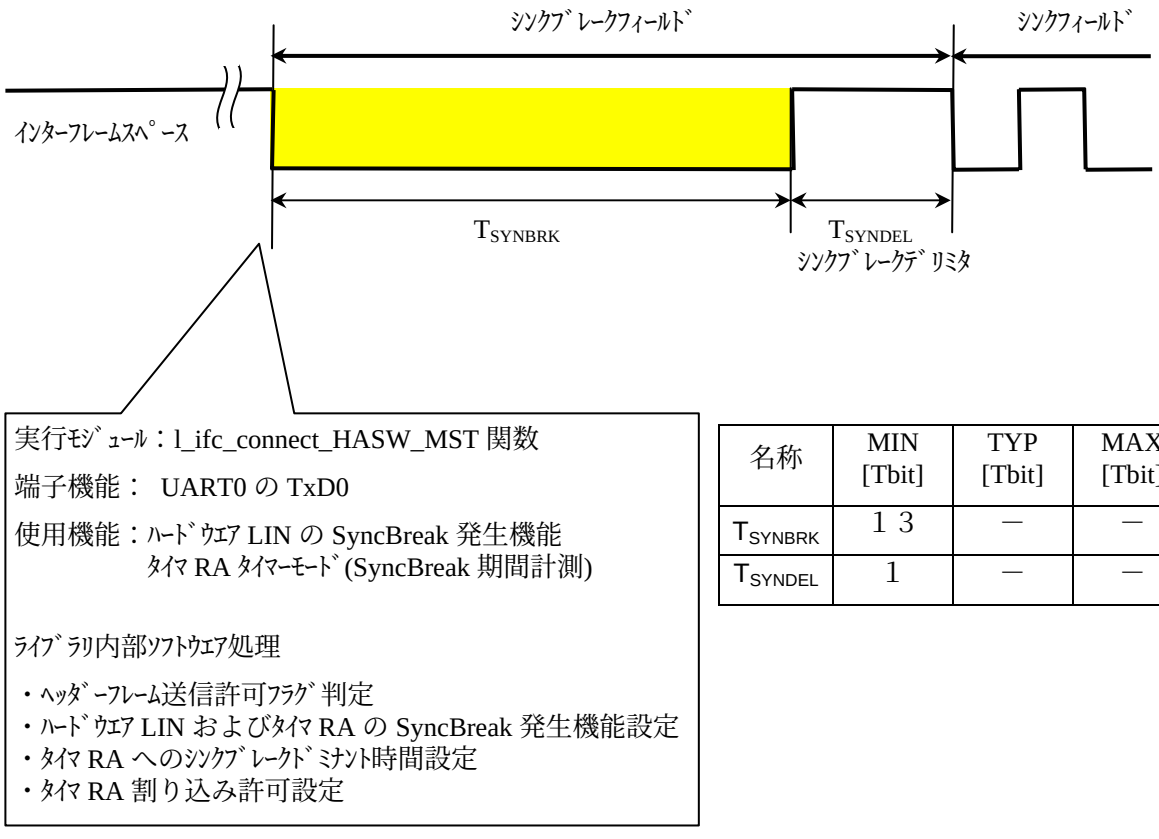
2.5.2 ヘッダフレームの送信

ヘッダフレーム送信許可フラグ（ui_sHASW_status.StBIT.Bit0）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 l_ifc_connect_HASW_MST 関数呼び出す事により，ヘッダーフレームの送信を開始します。

l_ifc_connect_HASW_MST 関数をユーザアプリケーションプログラムから呼び出す時には，予めシンクブレークデリミタのビット長（ui_sHASW_status.StBIT.Bit3，ui_sHASW_status.StBIT.Bit2），送信 ID（uc_dHASW_tx_id）を設定してください。（表４，表５参照）

2.5.2.1 シンクブレークフィールドの送信

ハードウエア LIN の SyncBreak 発生機能(TxD0 端子出力)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。



名称	MIN [Tbit]	TYP [Tbit]	MAX [Tbit]
T_{SYNBRK}	13	—	—
T_{SYNDEL}	1	—	—

図７　シンクブレークフィールドドミナント期間出力

2.5.2.2 シンクブレークデリミタの送信

ハードウエア LIN の SyncBreak 発生機能(TxD0 端子出力)によりシンクブレークデリミタ（レセンシブ期間）を出力します。

シンクブレークデリミタのビット長（ui_sHASW_status.StBIT.Bit3，ui_sHASW_status.StBIT.Bit2）設定値により約１～４ビット期間レセンシブ出力します。

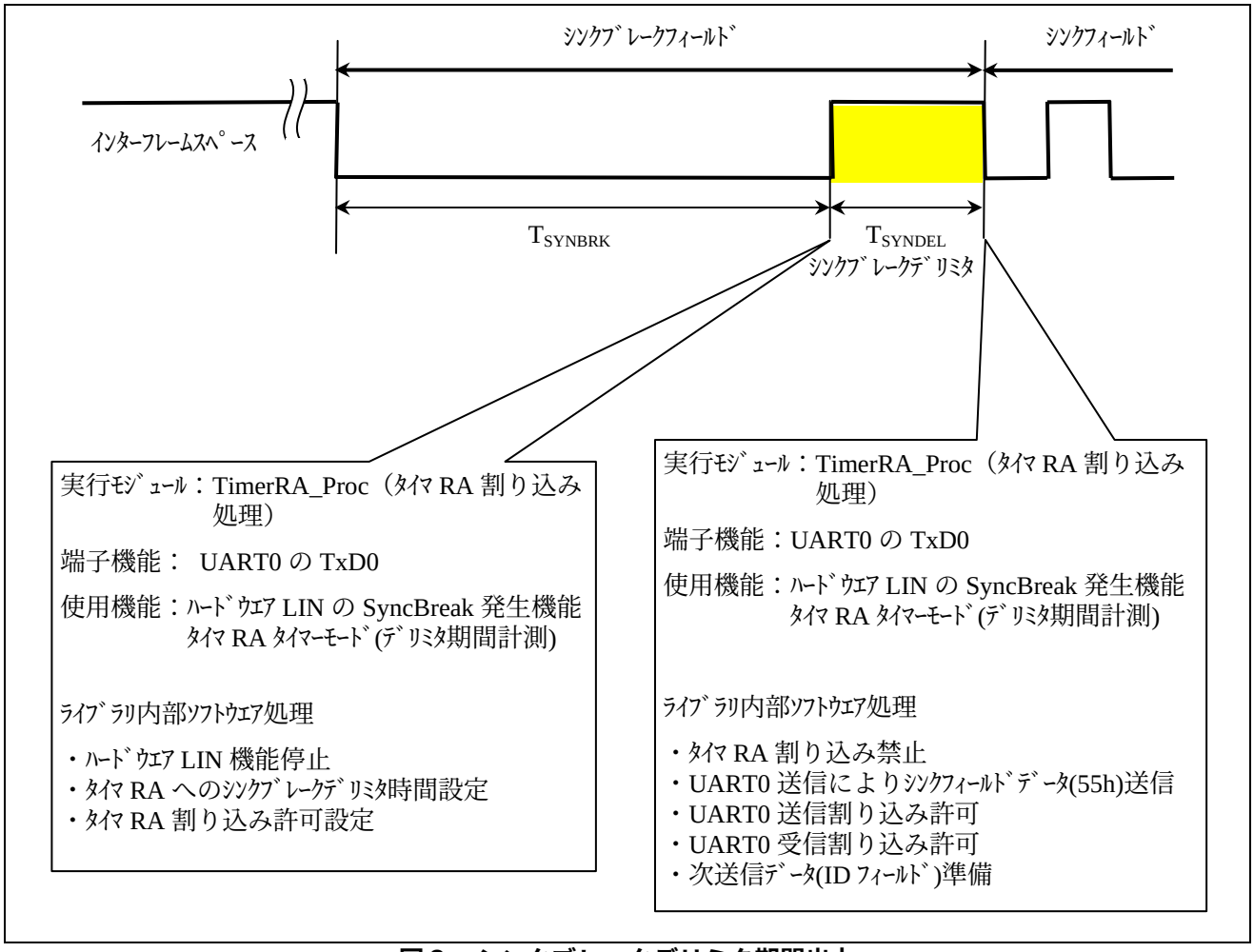


図８　シンクブレークデリミタ期間出力

2.5.2.3 シンクフィールドの送信

UART0 送信機能によりデータ 55h を送信します。

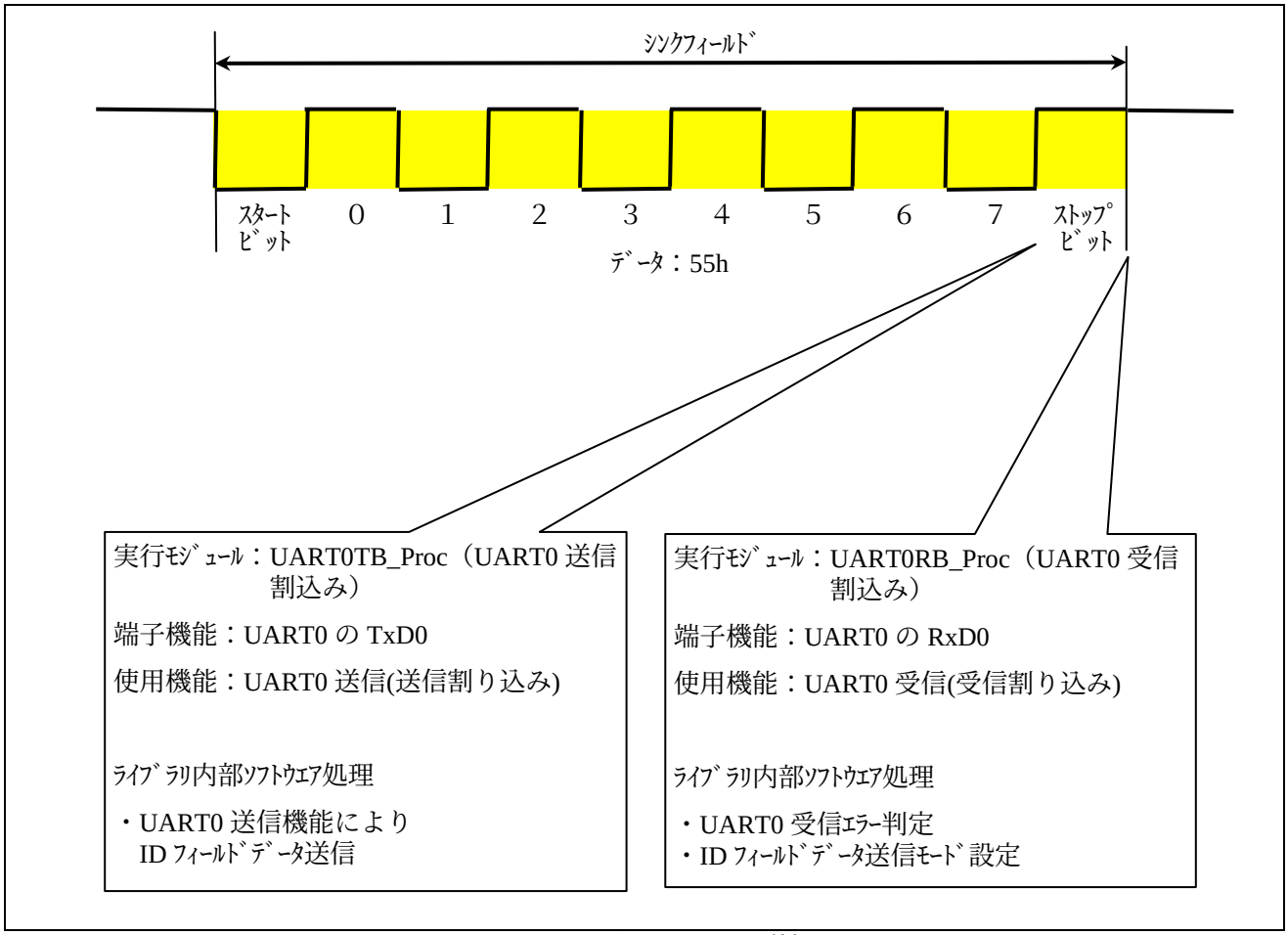


図９　シンクフィールド送信

2.5.2.4 ID フィールドの送信

UART0 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，uc_dHASW_tx_id 設定値にパリティビットが自動的に付加されます。

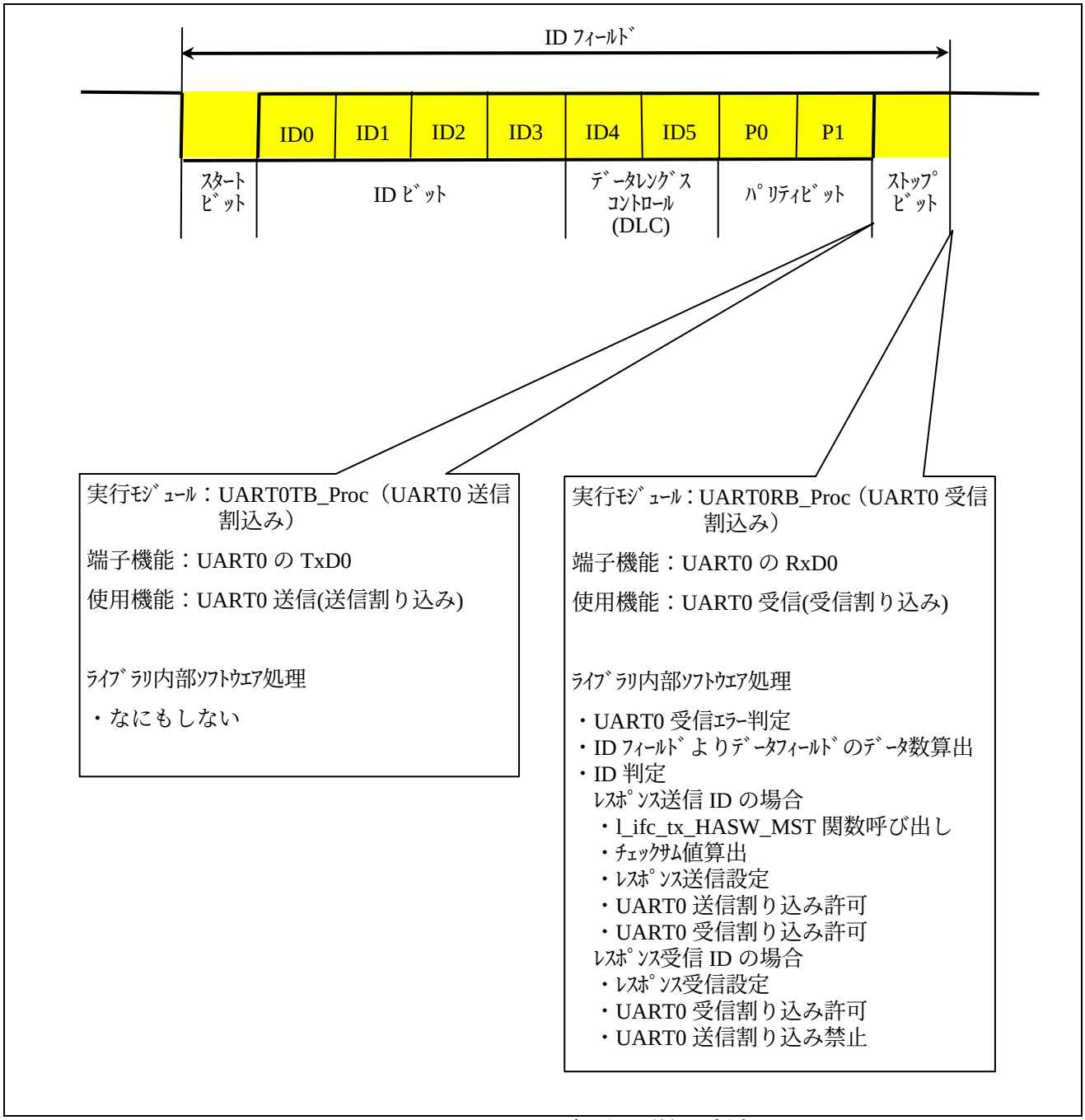


図１０　ID フィールドデータの送信と判定

2.5.3 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，UART0 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，UART0 受信機能によりレスポンスフレームを受信します。

2.5.3.1 データフィールドの送信

UART0 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，uc_bHASW_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。

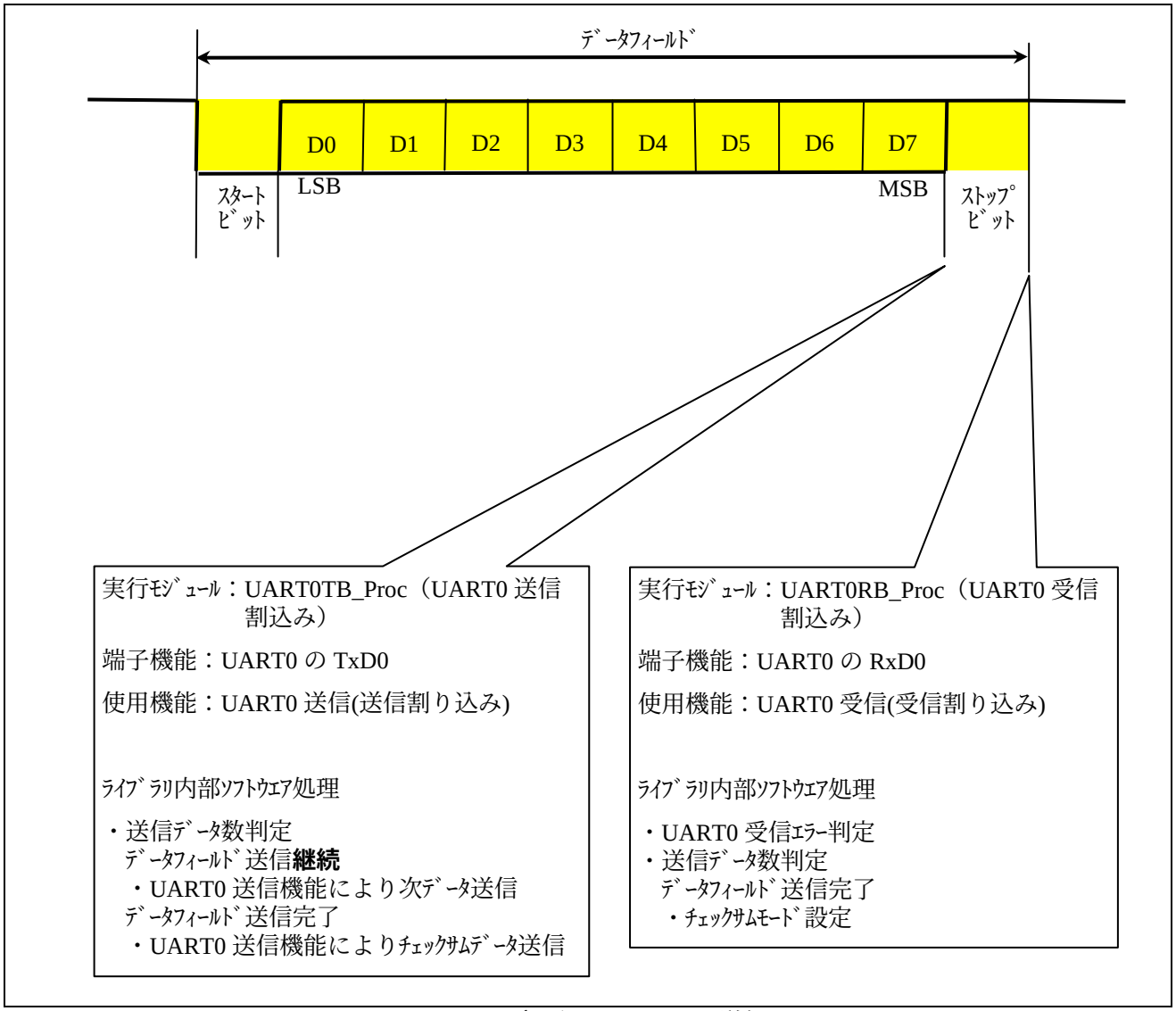


図１１　データフィールドの送信

2.5.3.2 チェックサムフィールドの送信

UART0 送信機能によりチェックサムフィールドを送信します。

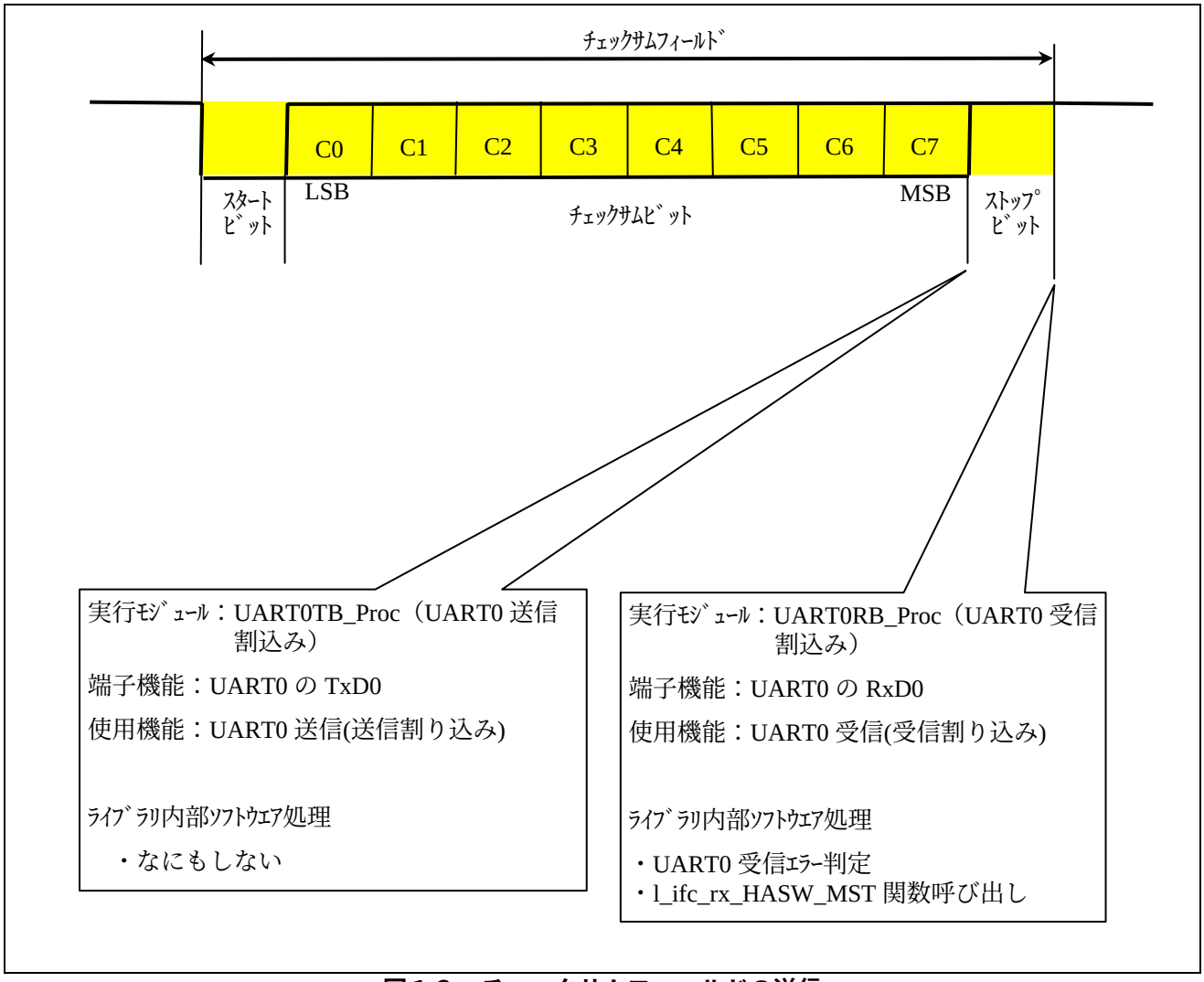


図１２　チェックサムフィールドの送信

2.5.3.3 データフィールドの受信

UART0 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，uc_bHASW_rx_data[0]から順に受信したデータ数のみ uc_bHASW_rx_data[0]～[7]に格納します。

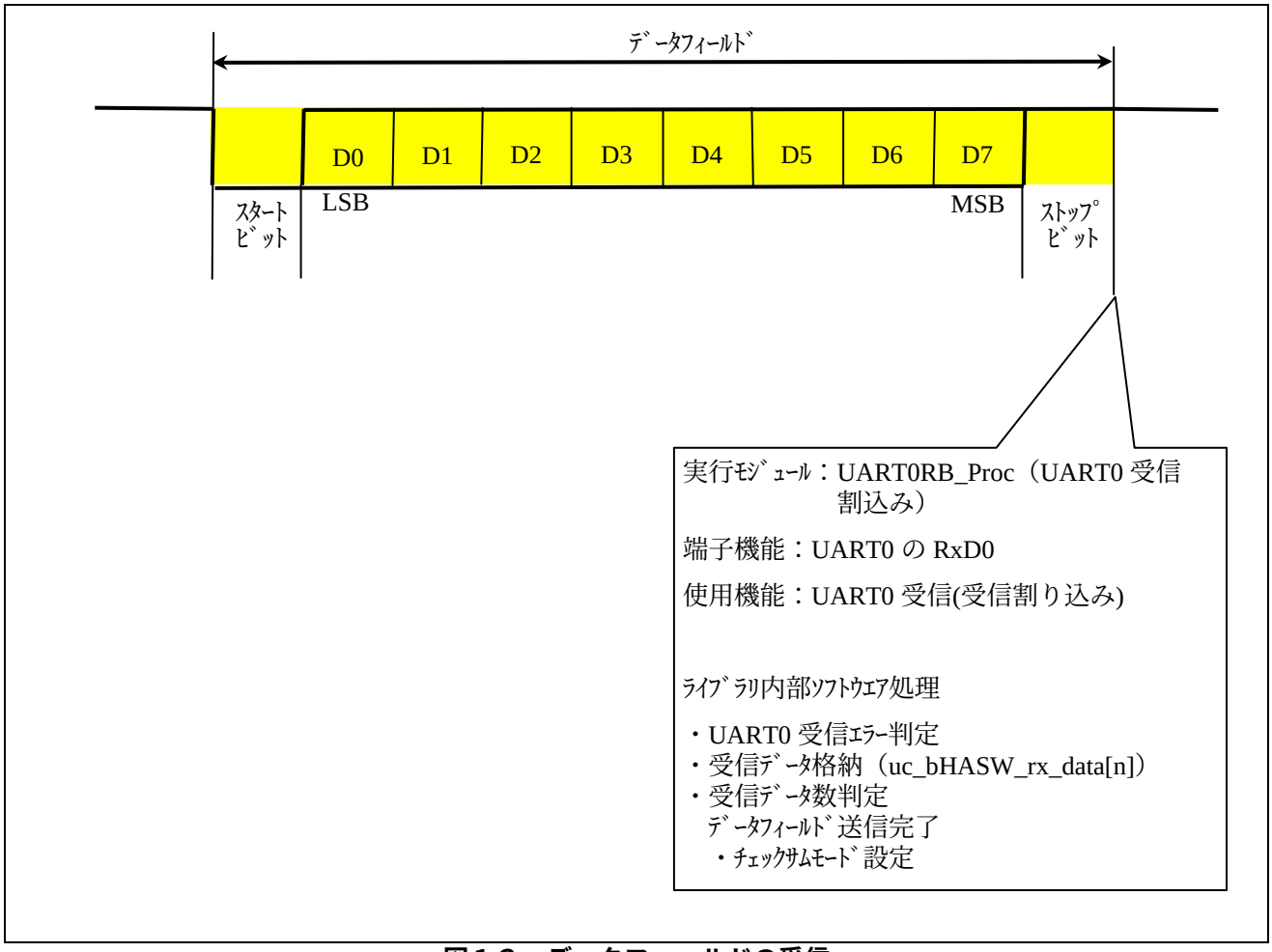


図１３　データフィールドの受信

2.5.3.4 チェックサムフィールドの受信

UART0 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。

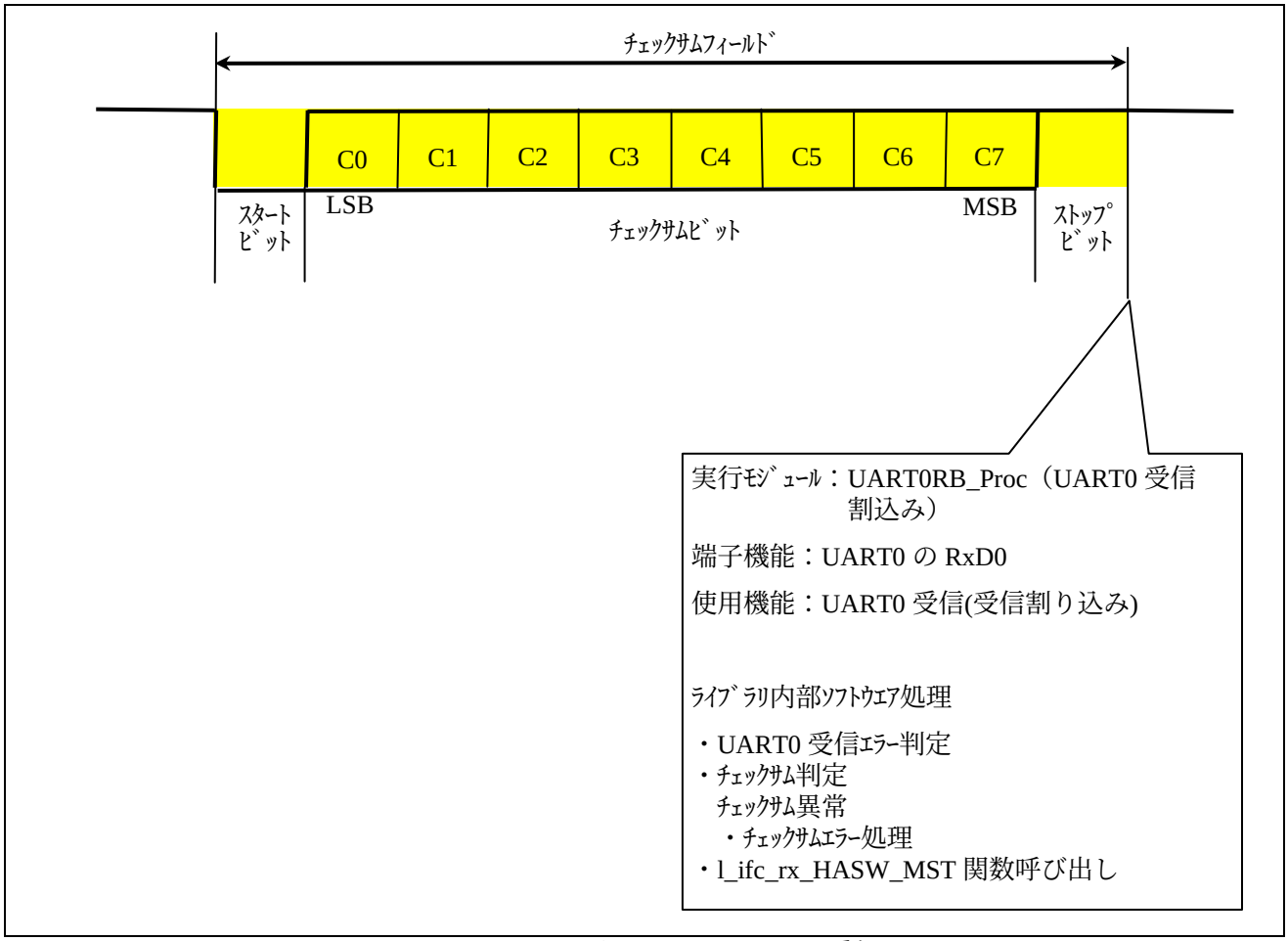


図１４　チェックサムフィールドの受信

2.6 ソフトウエア説明

以下に，本ライブラリソフトウエアの説明を示します。

2.6.1 ヘッダファイルの組み込み

R8C/24 グループ内蔵周辺レジスタ定義ファイル（SFR_R825.h），一線式通信ライブラリ用定義ファイル（HASW_MST.h）の組み込みを行います。

```
#include    "hasw_mst.h"
#ifdef    __CPU_R8C_24
#include    "sfr_r825.h"
#endif
```

2.6.2 関数定義

一線式通信ライブラリ用定義ファイル（HASW_MST.h）内にて，l_sys_init_HASW_MST 関数，l_ifc_init_HASW_MST 関数，l_ifc_connect_HASW_MST 関数，l_ifc_disconnect_HASW_MST 関数，l_ifc_rx_HASW_MST 関数，l_ifc_tx_HASW_MST 関数の定義を行っています。

なお，TimerRA_Proc 関数(タイマ RA 割り込み関数)，Uart0RB_Proc 関数(UART0 受信割り込み関数)，Uart0TB_Proc 関数(UART0 送信割り込み関数)についてはそれぞれの割り込み要因で呼び出されますので，ユーザアプルケーションプログラム内のそれぞれの割り込みベクタテーブルへ組み込みを行ってください。

```
void TimerRA_Proc(void);                        // ベクタ番地+88～+91 に設定
void Uart0RB_Proc(void);                        // ベクタ番地+72～+75 に設定
void Uart0TB_Proc(void);                        // ベクタ番地+68～+71 に設定
static void ChangeMeasureMode(unsigned char );
static unsigned char CheckIdentParity(unsigned char );
static unsigned char CheckMasterCommand(void);
static void MakeErrorStatus(unsigned char );
static unsigned char MakeHASW_CheckSUM(unsigned char );
```

2.6.3 ライブラリ内部定数・変数定義

ライブラリ内部で使用する定数および変数を定義します。

表６　ライブラリ内部定数定義

ラベル名（変数名）	データ型	機能説明
ErrorFlagTable[6]	unsigned char (配列)	ステータスフラグ設定データ
MasterCommandTable []	unsigned char (配列)	レスポンス送信 ID データ
HASW_STATUS	enum	通信状態管理データ
HASW_BUFFER	enum	通信バッファ管理データ
__TBIT_PULS_BASE	unsigned int	1 秒間カウント値(φosc 換算)
__TBIT_PULS_WIDTH	unsigned int	1 ビット期間カウント値(φosc 換算)
__SYNC_BRK_WIDTH	unsigned int	13 ビット期間カウント値(φosc 換算)
__SYNBRK_TM	unsigned int	13 ビット期間カウント値(ﾀｲﾏ RA 設定値)
__SYNBRK_TM_PRE	unsigned int	13 ビット期間カウント値(ﾀｲﾏ RA ﾌﾟﾘｽｹｰﾗ設定値)
__TBIT_TM	unsigned int	1 ビット期間カウント値(ﾀｲﾏ RA 設定値)
__TBIT_TM_PRE	unsigned int	1 ビット期間カウント値(ﾀｲﾏ RA ﾌﾟﾘｽｹｰﾗ設定値)
ERROR_MODE	enum	エラー管理データ

表７　ライブラリ内部変数定義

ラベル名（変数名）	データ型	機能説明
uc_cHaswStatus	unsigned char	通信インタフェース状態管理 （enum HASW_STATUS）
uc_cTransCnt	unsigned char	通信データフィールド送受信カウンタ
uc_cTransPtr	unsigned char	通信データフィールドバッファポインタ
un_sUART0	（構造体）	UART0 送受信テンポラリバッファ
un_sUART0.uiAll	ワードアクセス unsigned int	
	ビットアクセス	
un_sUART0.StBIT.Bit15	ビット-15	エラーサムフラグ
un_sUART0.StBIT.Bit14	ビット-14	パリティエラーフラグ
un_sUART0.StBIT.Bit13	ビット-13	フレーミングエラーフラグ
un_sUART0.StBIT.Bit12	ビット-12	オーバーランエラーフラグ
un_sUART0.StBIT.Bit11	ビット-11	リザーブビット
un_sUART0.StBIT.Bit10	ビット-10	リザーブビット
un_sUART0.StBIT.Bit9	ビット-9	リザーブビット
un_sUART0.StBIT.Bit8	ビット-8	送受信データ
un_sUART0.StBIT.Bit7	ビット-7	
un_sUART0.StBIT.Bit6	ビット-6	
un_sUART0.StBIT.Bit5	ビット-5	
un_sUART0.StBIT.Bit4	ビット-4	
un_sUART0.StBIT.Bit3	ビット-3	
un_sUART0.StBIT.Bit2	ビット-2	
un_sUART0.StBIT.Bit1	ビット-1	
un_sUART0.StBIT.Bit0	ビット-0	
un_sU0BRGdata	（構造体）	UART0 BRG 値データ格納バッファ
un_sU0BRGdata.uiAll	ワードアクセス unsigned int	
	バイトアクセス	
un_sU0BRGdata.StBYTE. Byte0	下位 8 ビット	U0BRG 設定データ
un_sU0BRGdata.StBYTE. Byte1	上位 8 ビット	U0BRG 設定データ
uc_bHaswBuff [0]～[9]	unsigned char (配列)	通信バッファ （enum HASW_BUFFER）
uc_cTimerRA [0]～[1]	unsigned char (配列)	[0]：13 ビット期間カウント値(ﾀｲﾏ RA 設定値) [1]：1 ビット期間カウント値(ﾀｲﾏ RA 設定値)
uc_cTimerPreRA [0]～[1]	unsigned char (配列)	[0]：13 ビット期間カウント値(ﾀｲﾏ RA ﾌﾟﾘｽｹｰﾗ設定値) [1]：1 ビット期間カウント値(ﾀｲﾏ RA ﾌﾟﾘｽｹｰﾗ設定値)

```
static enum HASW_STATUS {
        enHASW_INIT,                    // 初期状態モード
        enSYNCH_BREAK,                  // SynchBreak 期間送信モード
        enSYNCH_MEASURE,                // SynchBreak デリミタ期間送信モード
        enSYNCH_SYNCH,                  // Synch フィールド送信モード
        enIDENT,                        // Ident 送信モード
        enRESPONSE,                     // Response 受信モード
        enMASTERCMD,                    // Command 送信モード
```

```
        enCHECKSUM,                                     // Checksum 送受信モード
        enHASW_STATUS
};
static unsigned char    uc_cHaswStatus;             // 通信インタフェース状態管理
#ifdef    SBDATA_ON
    #pragma  SBDATA        uc_cHaswStatus
#endif
static unsigned char    uc_cTransCnt;               // 通信データフィールド送受信カウンタ
#ifdef    SBDATA_ON
    #pragma  SBDATA  uc_cTransCnt
#endif
static unsigned char    uc_cTransPtr;               // 通信データフィールドバッファポインタ
#ifdef    SBDATA_ON
    #pragma  SBDATA  uc_cTransPtr                   // [enHASW_DATA0～enHASW_DATA7]
#endif

typedef union {
    struct{
        char    Bit0:1;
        char    Bit1:1;
        char    Bit2:1;
        char    Bit3:1;
        char    Bit4:1;
        char    Bit5:1;
        char    Bit6:1;
        char    Bit7:1;
        char    Bit8:1;
        char    Bit9:1;
        char    Bit10:1;
        char    Bit11:1;
        char    Bit12:1;
        char    Bit13:1;
        char    Bit14:1;
        char    Bit15:1;
    }StBIT;
    struct {
        unsigned char    Byte0;
        unsigned char    Byte1;
    }StBYTE;
    unsigned int uiAll;
}UnWORD;
static UnWORD    un_sUART0;                             // UART0 送受信テンポラリバッファ
#ifdef    SBDATA_ON
    #pragma  SBDATA  un_sUART0
#endif
static UnWORD    un_sU0BRGdata;                         // UART0 BRG データ格納バッファ
#ifdef    SBDATA_ON
    #pragma  SBDATA  un_sU0BRGdata
#endif

static enum HASW_BUFFER {
        enHASW_ID,
        enHASW_DATA0,
        enHASW_DATA1,
        enHASW_DATA2,
        enHASW_DATA3,
        enHASW_DATA4,
```

```
        enHASW_DATA5,
        enHASW_DATA6,
        enHASW_DATA7,
        enHASW_CHECKSUM,
        enHASW_BUFFER
};
static unsigned char    uc_bHaswBuff[enHASW_BUFFER];    // 通信バッファ
#ifdef    SBDATA_ON
    #pragma  SBDATA  uc_bHaswBuff
#endif

/*=========================================================================================*/
/*        定数定義                                                                         */
/*-----------------------------------------------------------------------------------------*/
#ifdef __OVER_10M_Hz    // OSC が 10MHz 以上の場合, TimerRA のソースは 2/f
    #define    __TBIT_PULS_BASE        (__OSC_Hz / 2)        // 1sec / 2/f(Timer Count Source)
#else                   // OSC が 10MHz 以下の場合, TimerRA のソースは 1/f
    #define    __TBIT_PULS_BASE        __OSC_Hz              // 1sec / 1/f(Timer Count Source)
#endif
#define        __TBIT_PULS_WIDTH        (__TBIT_PULS_BASE / __B_rate)    // 1 ビットパルス幅
#define        __SYNC_BRK_WIDTH         (__TBIT_PULS_WIDTH * 13)         // SYCBRK 信号パルス幅

#if __OSC_Hz == 8000000         // φOSC=8MHz
    #if __B_rate == 19200    // ボーレート=19200bps
        #define         __SYNBRK_TM          24          // 13 ビットパルス幅(Timer 値)
        #define         __SYNBRK_TM_PRE     226          // 13 ビットパルス幅(TimerPre 値)
        #define         __TBIT_TM            3           // 1 ビットパルス幅(Timer 値)
        #define         __TBIT_TM_PRE      139           // 1 ビットパルス幅(TimerPre 値)
    #elif __B_rate == 9600    // ボーレート=9600bps
        #define         __SYNBRK_TM          44          // 13 ビットパルス幅(Timer 値)
        #define         __SYNBRK_TM_PRE     246          // 13 ビットパルス幅(TimerPre 値)
        #define         __TBIT_TM            7           // 1 ビットパルス幅(Timer 値)
        #define         __TBIT_TM_PRE      119           // 1 ビットパルス幅(TimerPre 値)
    #elif __B_rate == 2400    // ボーレート=2400bps
        #define         __SYNBRK_TM         174          // 13 ビットパルス幅(Timer 値)
        #define         __SYNBRK_TM_PRE     249          // 13 ビットパルス幅(TimerPre 値)
        #define         __TBIT_TM            33          // 1 ビットパルス幅(Timer 値)
        #define         __TBIT_TM_PRE      101           // 1 ビットパルス幅(TimerPre 値)
    #else
        #define     __AUTO_CALCURATION                  // 転送時間自動計算設定
    #endif
#elif __OSC_Hz == 20000000    // φOSC=20MHz？
    #if __B_rate == 19200
        #define         __SYNBRK_TM          28          // 13 ビットパルス幅(Timer 値)
        #define         __SYNBRK_TM_PRE     242          // 13 ビットパルス幅(TimerPre 値)
        #define         __TBIT_TM            9           // 1 ビットパルス幅(Timer 値)
        #define         __TBIT_TM_PRE       58           // 1 ビットパルス幅(TimerPre 値)
    #elif __B_rate == 9600
        #define         __SYNBRK_TM          63          // 13 ビットパルス幅(Timer 値)
        #define         __SYNBRK_TM_PRE     215          // 13 ビットパルス幅(TimerPre 値)
        #define         __TBIT_TM            7           // 1 ビットパルス幅(Timer 値)
        #define         __TBIT_TM_PRE      149           // 1 ビットパルス幅(TimerPre 値)
    #elif __B_rate == 2400    // ボーレート=2400bps
        #define         __SYNBRK_TM         222          // 13 ビットパルス幅(Timer 値)
        #define         __SYNBRK_TM_PRE     244          // 13 ビットパルス幅(TimerPre 値)
        #define         __TBIT_TM           17           // 1 ビットパルス幅(Timer 値)
        #define         __TBIT_TM_PRE      245           // 1 ビットパルス幅(TimerPre 値)
```

```
    #else
        #define    __AUTO_CALCURATION                  // 転送時間自動計算設定
    #endif
#else
    #define    __AUTO_CALCURATION                      // 転送時間自動計算設定
#endif
#ifdef __AUTO_CALCURATION                             // 転送時間自動計算？
    static unsigned char    uc_cTimerRA[2];           // タイマ RA 設定値[0]:13 ビット時, [1]1 ビット時
    static unsigned char    uc_cTimerPreRA[2];        // タイマ RA 設定値[0]:13 ビット時, [1]1 ビット時
    #ifdef    SBDATA_ON
        #pragma  SBDATA  uc_cTimerRA
        #pragma  SBDATA  uc_cTimerPreRA
    #endif
    typedef union {
        struct {
            unsigned int    quot;
            unsigned int    rem;
        }StINT;
        unsigned long    ulAll;
    }div_u;
    static unsigned long udiv(unsigned long ,unsigned int );
    #pragma __ASMMACRO    udiv(R2R0,R1)
    #pragma ASM
        _udiv    .macro
        DIVU.W    R1
        .endm
    #pragma ENDASM
#endif
/*=========================================================================================*/
/*        コマンドテーブル定義                                                             */
/*-----------------------------------------------------------------------------------------*/
const unsigned char MasterCommandTable[]={
#ifdef    __ID00
    __ID00,
#endif
#ifdef    __ID01
    __ID01,
#endif
#ifdef    __ID02
    __ID02,
#endif
#ifdef    __ID03
    __ID03,
#endif
#ifdef    __ID04
    __ID04,
#endif
#ifdef    __ID05
    __ID05,
#endif
#ifdef    __ID06
    __ID06,
#endif
#ifdef    __ID07
    __ID07,
#endif
#ifdef    __ID08
```

```
    __ID08,
#endif
#ifdef    __ID09
    __ID09,
#endif
#ifdef    __ID10
    __ID10,
#endif
#ifdef    __ID11
    __ID11,
#endif
#ifdef    __ID12
    __ID12,
#endif
#ifdef    __ID13
    __ID13,
#endif
#ifdef    __ID14
    __ID14,
#endif
#ifdef    __ID15
    __ID15,
#endif
#ifdef    __ID16
    __ID16,
#endif
#ifdef    __ID17
    __ID17,
#endif
#ifdef    __ID18
    __ID18,
#endif
#ifdef    __ID19
    __ID19,
#endif
#ifdef    __ID20
    __ID20,
#endif
#ifdef    __ID21
    __ID21,
#endif
#ifdef    __ID22
    __ID22,
#endif
#ifdef    __ID23
    __ID23,
#endif
#ifdef    __ID24
    __ID24,
#endif
#ifdef    __ID25
    __ID25,
#endif
#ifdef    __ID26
    __ID26,
#endif
#ifdef    __ID27
```

```
    __ID27,
#endif
#ifdef    __ID28
    __ID28,
#endif
#ifdef    __ID29
    __ID29,
#endif
#ifdef    __ID30
    __ID30,
#endif
#ifdef    __ID31
    __ID31,
#endif
#ifdef    __ID32
    __ID32,
#endif
#ifdef    __ID33
    __ID33,
#endif
#ifdef    __ID34
    __ID34,
#endif
#ifdef    __ID35
    __ID35,
#endif
#ifdef    __ID36
    __ID36,
#endif
#ifdef    __ID37
    __ID37,
#endif
#ifdef    __ID38
    __ID38,
#endif
#ifdef    __ID39
    __ID39,
#endif
#ifdef    __ID40
    __ID40,
#endif
#ifdef    __ID41
    __ID41,
#endif
#ifdef    __ID42
    __ID42,
#endif
#ifdef    __ID43
    __ID43,
#endif
#ifdef    __ID44
    __ID44,
#endif
#ifdef    __ID45
    __ID45,
#endif
#ifdef    __ID46
```

```
    __ID46,
#endif
#ifdef    __ID47
    __ID47,
#endif
#ifdef    __ID48
    __ID48,
#endif
#ifdef    __ID49
    __ID49,
#endif
#ifdef    __ID50
    __ID50,
#endif
#ifdef    __ID51
    __ID51,
#endif
#ifdef    __ID52
    __ID52,
#endif
#ifdef    __ID53
    __ID53,
#endif
#ifdef    __ID54
    __ID54,
#endif
#ifdef    __ID55
    __ID55,
#endif
#ifdef    __ID56
    __ID56,
#endif
#ifdef    __ID57
    __ID57,
#endif
#ifdef    __ID58
    __ID58,
#endif
#ifdef    __ID59
    __ID59,
#endif
#ifdef    __ID60
    __ID60
#endif
};

/*=============================================================================================*/
/*        define 定義                                                                          */
/*---------------------------------------------------------------------------------------------*/
#define     pTxD            p1_4                // I : Hi-Z : TxD 出力
#define     pdTxD            pd1_4
#define     pRxD            p1_5                // I : Hi_Z : RxD 入力
#define     pdRxD            pd1_5
#define        uc_sUART0LowBuffer        un_sUART0.StBYTE.Byte0
#define        uc_sUART0HighBuffer        un_sUART0.StBYTE.Byte1
#define        ui_sUART0Buffer            un_sUART0.uiAll
```

```
#define         uc_fMessageFrame        ui_sHASW_status.StBIT.Bit0    // フレーム送信許可フラグ
static enum ERROR_MODE {
        enERROR_CHECKSUM,                               // チェックサムエラー
        enERROR_SYNCH,                                  // ノーレスポンスエラー
        enERROR_PARITY,                                 // IDパリティエラー
        enERROR_OVER_RUN,                               // オーバーランエラー
        enERROR_FRAME,                                  // フレームエラー
        enERROR_BIT,                                    // ビットエラー
        enERROR_MODE
};
#define         FALSE    0
#define         TRUE    1
#define         SET         1
#define         P_IN    0
```

2.6.4 グローバル変数定義

一線式通信ライブラリ用定義ファイル（HASW_MST.h）内にて，ユーザー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（uc_dHASW_tx_id,uc_bHASW_tx_data[8], uc_dHASW_rx_id, uc_bHASW_rx_data[8], ui_sHASW_status）を定義しています。

2.6.5 初期設定用関数

一線式通信制御に使用する R8C/24 グループ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エアフラグなどの初期化を行います。

注意：端子 P1_4(TxD0)，P1_5(RxD0)は一線式通信で使用します。ユーザアプリケーションでポート 1 におけるその他の端子(P1_0～P1_3，P1_6～P1_7)を使用する場合，端子 P1_4(TxD0)，P1_5(RxD0)の方向レジスタを必ず入力方向に設定するようにユーザアプリケーションで設定してください。

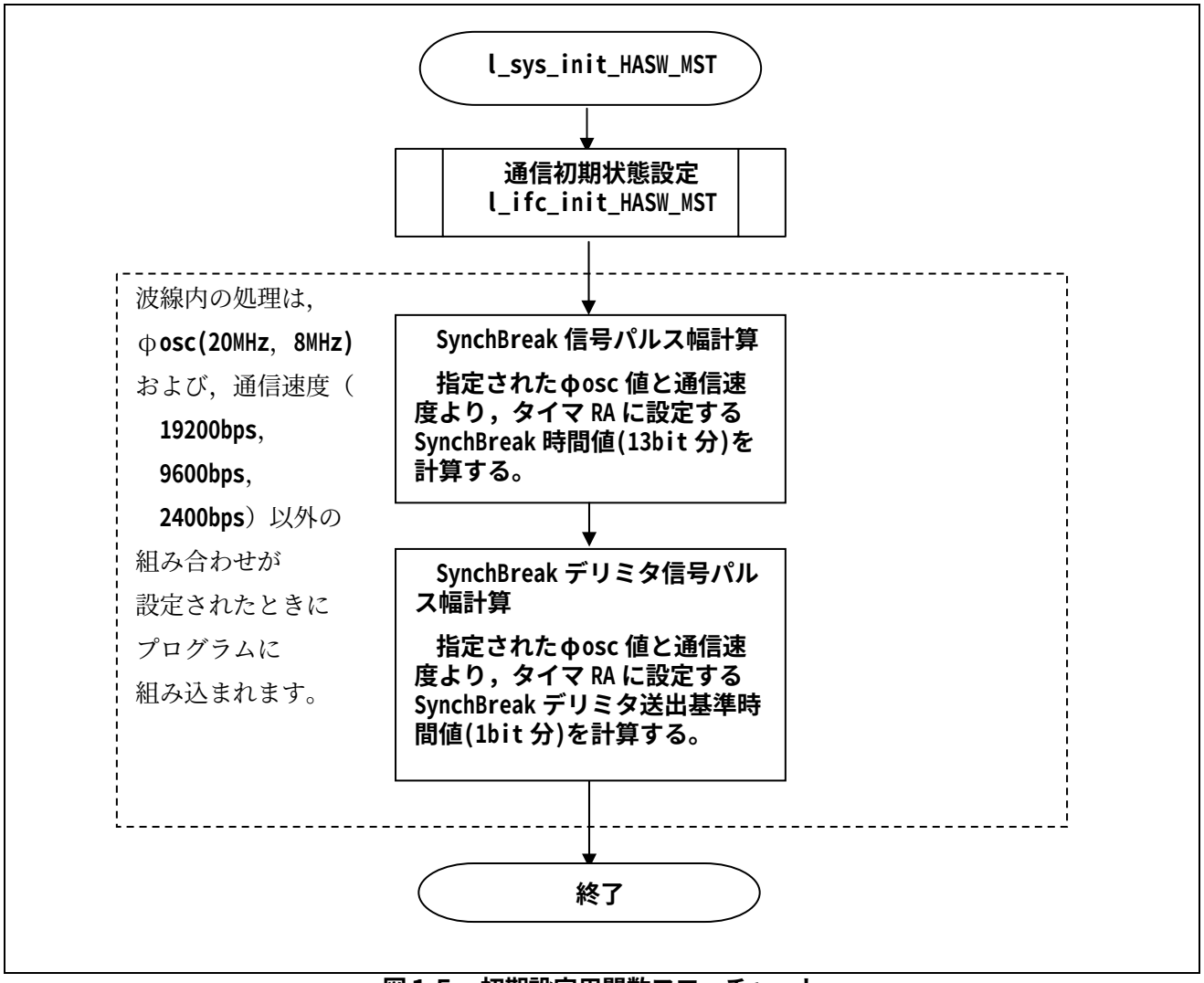


図１５　初期設定用関数フローチャート

```
void l_sys_init_HASW_MST(void)
{
#ifdef __AUTO_CALCURATION                    // 転送時間自動計算？
    div_u    tmp_div;
```

```
    unsigned char    denom,rem,min_rem,abs_rem;
#endif

    l_ifc_init_HASW_MST();                      // 初期状態設定
#ifdef __AUTO_CALCURATION                       // 転送時間自動計算？
    /* SynchBreak 信号幅計算 */
    min_rem = 255;
    for(denom=255;denom>0;--denom){
        tmp_div.ulAll = udiv(__SYNC_BRK_WIDTH,(unsigned int)denom);
        rem = (unsigned char)tmp_div.StINT.rem;
        abs_rem = denom - rem;
        if (tmp_div.StINT.quot > 255) break;    // 商が 255 を越えたら，その時点で終わる。
        else if (rem <= 2) {        // 差が 2 以下？
            uc_cTimerPreRA[0] = denom;                          // 除数
            uc_cTimerRA[0] = (unsigned char)tmp_div.StINT.quot;    // 商
            break;
        }
        else if(abs_rem <= 2) {        // 差が 2 以下？
            uc_cTimerPreRA[0] = denom;                          // 除数
            uc_cTimerRA[0] = (unsigned char)tmp_div.StINT.quot + 1;    // 商
            break;
        }
        else {                                  // 差が 2 以上の場合
            if (rem > abs_rem) {
                rem = abs_rem;
                abs_rem = 1;
            }
            else {
                abs_rem = 0;
            }
            if (rem < min_rem) {                // 今までの除算で最小の差か？
                min_rem = rem;
                uc_cTimerPreRA[0] = denom;        // 最小の差の時の商と除数を保持
                uc_cTimerRA[0] = (unsigned char)tmp_div.StINT.quot + abs_rem;
            }
        }
    }
    /* 1 ビット分パルス幅計算 */
    min_rem = 255;
    for(denom=255;denom>0;--denom){
        tmp_div.ulAll = udiv(__TBIT_PULS_WIDTH,(unsigned int)denom);
        rem = (unsigned char)tmp_div.StINT.rem;
        abs_rem = denom - rem;
        if (tmp_div.StINT.quot >= 64) break;        // 商が 64 を越えたら，その時点で終わる。
        else if (rem <= 1) {        // 差が 1 以下？
            uc_cTimerPreRA[1] = denom;                          // 除数
            uc_cTimerRA[1] = (unsigned char)tmp_div.StINT.quot;    // 商
            break;
        }
        else if (abs_rem <= 1) {        // 差が 1 以下？
            uc_cTimerPreRA[1] = denom;                          // 除数
            uc_cTimerRA[1] = (unsigned char)tmp_div.StINT.quot + 1;    // 商
            break;
        }
        else {                                  // 差が 1 以上の場合
            if (rem > abs_rem) {
                rem = abs_rem;
```

```
                abs_rem = 1;
            }
            else {
                abs_rem = 0;
            }
            if (rem < min_rem) {                // 今までの除算で最小の差か？
                min_rem = rem;
                uc_cTimerPreRA[1] = denom;              // 最小の差の時の商と除数を保持
                uc_cTimerRA[1] = (unsigned char)tmp_div.StINT.quot + abs_rem;
            }
        }
    }
    uc_cTimerPreRA[0] -= 1;                         // ダウンカウンタへの設定のため-1 する。
    uc_cTimerRA[0] -= 1;
    uc_cTimerPreRA[1] -= 1;                         // uc_cTimerRA[1]は後の計算のため,
                                                    // デクリメントせずに計算値のまま保持
#endif
}
```

2.6.6 一線式通信制御初期設定関数

一線式通信制御に使用する R8C/24 グループ内蔵周辺機能の初期状態設定, およびライブラリ内部で使用するソフトウエアフラグなどの初期化を行います。通信中にエラーが発生した場合，本関数を呼び出すことにより，通信制御状態を初期状態に戻します。

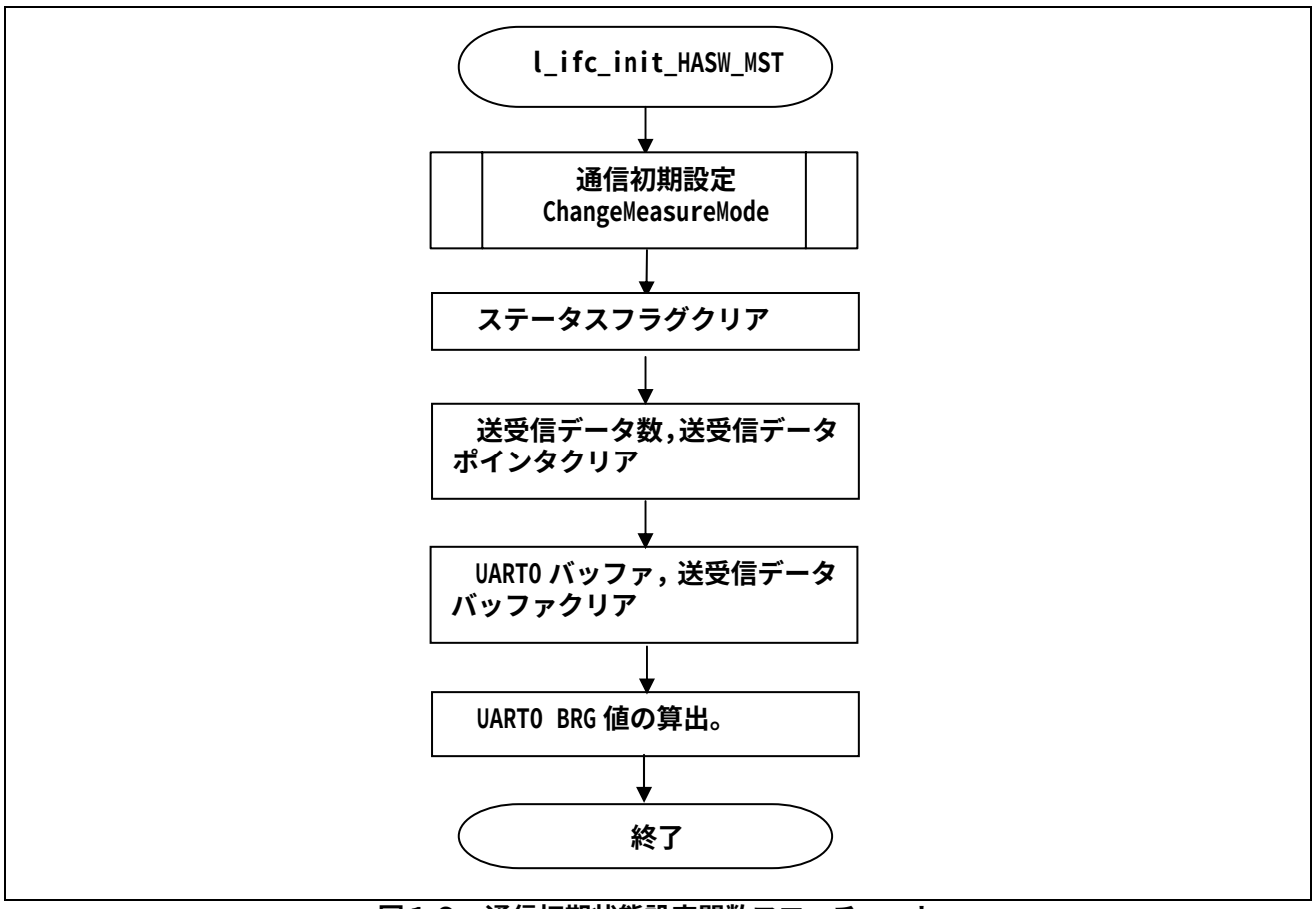


図１６　通信初期状態設定関数フローチャート

```
void l_ifc_init_HASW_MST(void)
{
    unsigned char    c;

    ChangeMeasureMode(enHASW_INIT);                 // 初期状態設定
    ui_sHASW_status.StBYTE.Byte1 = 0x00;            // ステータスフラグクリア
    uc_cTransCnt = 0;                               // 送受信データ数クリア
    uc_cTransPtr = 0;                               // 送受信データポインタクリア
    ui_sUART0Buffer = 0;                            // UART0 バッファクリア
    for (c=0;c<enHASW_BUFFER;++c) {                 // 送受信データバッファクリア
        uc_bHaswBuff[c] = 0;
    }
#ifdef __OVER_10M_Hz    // OSC が 10MHz 以上の場合，8 で割ると BRG 値が算出される
    un_sU0BRGdata.uiAll  = (__TBIT_PULS_WIDTH + (__TBIT_PULS_WIDTH & 0x0004)) >> 3; // 四捨五入
#else                    // OSC が 10MHz 以下の場合，16 で割ると BRG 値が算出される
    un_sU0BRGdata.uiAll  = (__TBIT_PULS_WIDTH + (__TBIT_PULS_WIDTH & 0x0008)) >> 4; // 四捨五入
#endif
}
```

2.6.7 一線式通信制御ヘッダーフレーム送信用関数

ヘッダーフレーム（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。

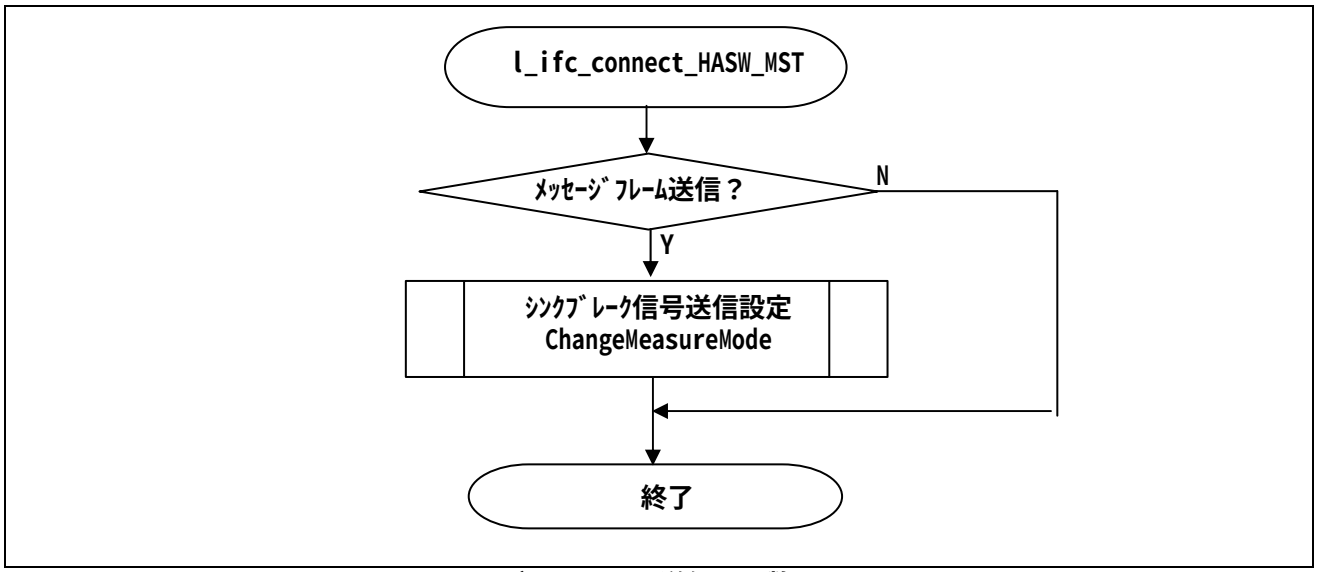


図１７　ヘッダーフレーム送信用関数フローチャート

```
void l_ifc_connect_HASW_MST(void)
{
    if (uc_fMessageFrame == SET) {
        ChangeMeasureMode(enSYNCH_BREAK);                    // SynchBreak 信号送信
    }
}
```

2.6.8 一線式通信制御停止関数

一線式通信制御を停止し，接続されている一線式通信バス上の通信に関与しなくなります。

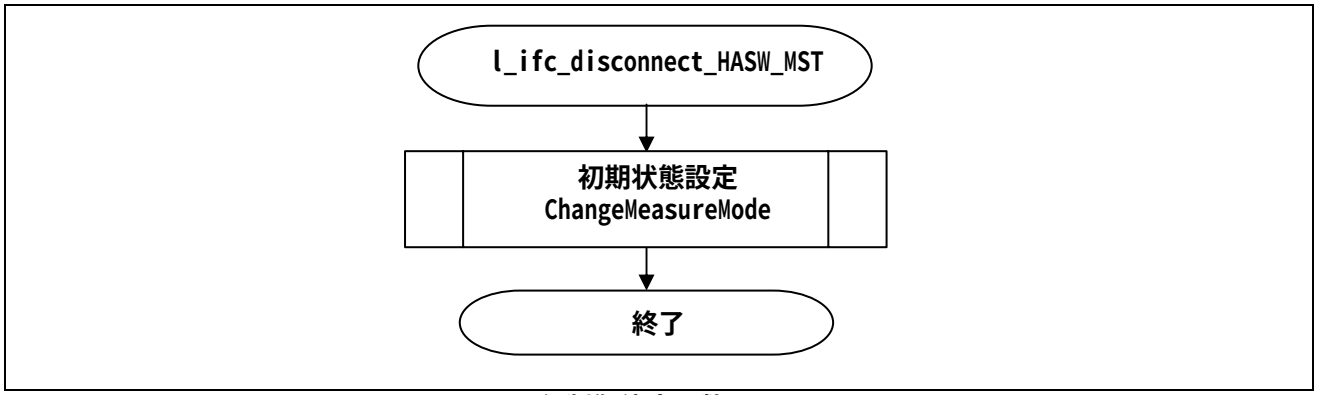


図１８　通信制御停止関数フローチャート

```
void l_ifc_disconnect_HASW_MST(void)
{
    ChangeMeasureMode(enHASW_INIT);                          // 初期状態設定
}
```

2.6.9 タイマ RA 割り込み関数

シンクブレーク信号出力タイマ RA のオーバーフロー割り込み，シンクブレークデリミタ信号出力タイマ RA のオーバーフロー割り込み処理を行います。

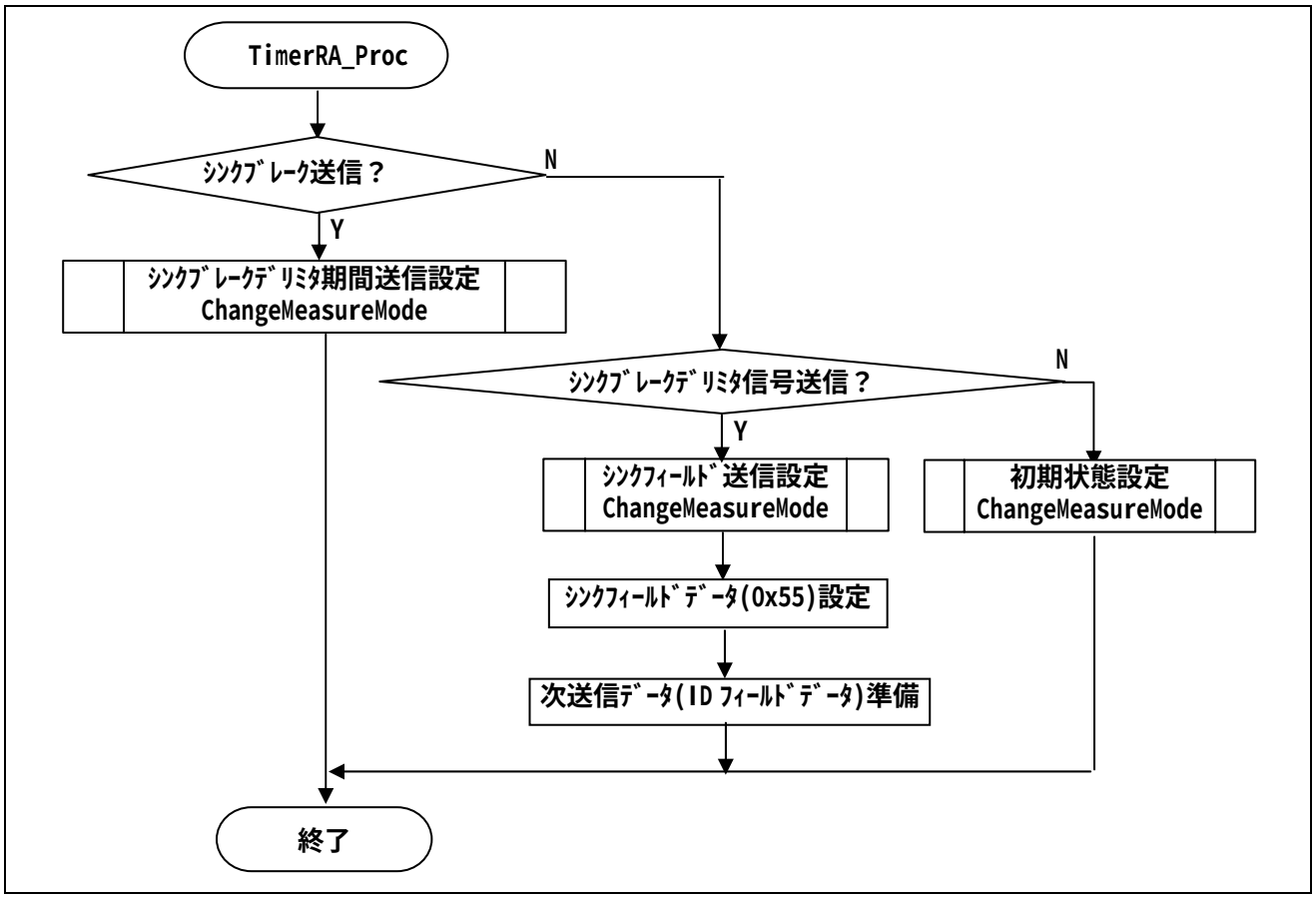


図１９　タイマ RA 割り込み関数フローチャート

```
#pragma INTERRUPT /B TimerRA_Proc
void TimerRA_Proc(void)
{
    switch(uc_cHaswStatus){
    case enSYNCH_BREAK:         // SynchBreak 期間送信モード
        ChangeMeasureMode(enSYNCH_MEASURE);                 // SynchBreak デリミタ期間送信設定
        break;
    case enSYNCH_MEASURE:     // SynchBreak デリミタ期間送信モード
        ChangeMeasureMode(enSYNCH_SYNCH);                   // Synch フィールド送信状態設定
        ui_sUART0Buffer = 0x0055;
        u0tb = ui_sUART0Buffer;
        uc_bHaswBuff[enHASW_ID] = uc_dHASW_tx_id | CheckIdentParity(uc_dHASW_tx_id);
        break;
    default:
        ChangeMeasureMode(enHASW_INIT);                             // 初期状態設定
        break;
    }
}
```

2.6.10 UART0 送信割り込み関数

UART0 送信割り込み処理を行います。

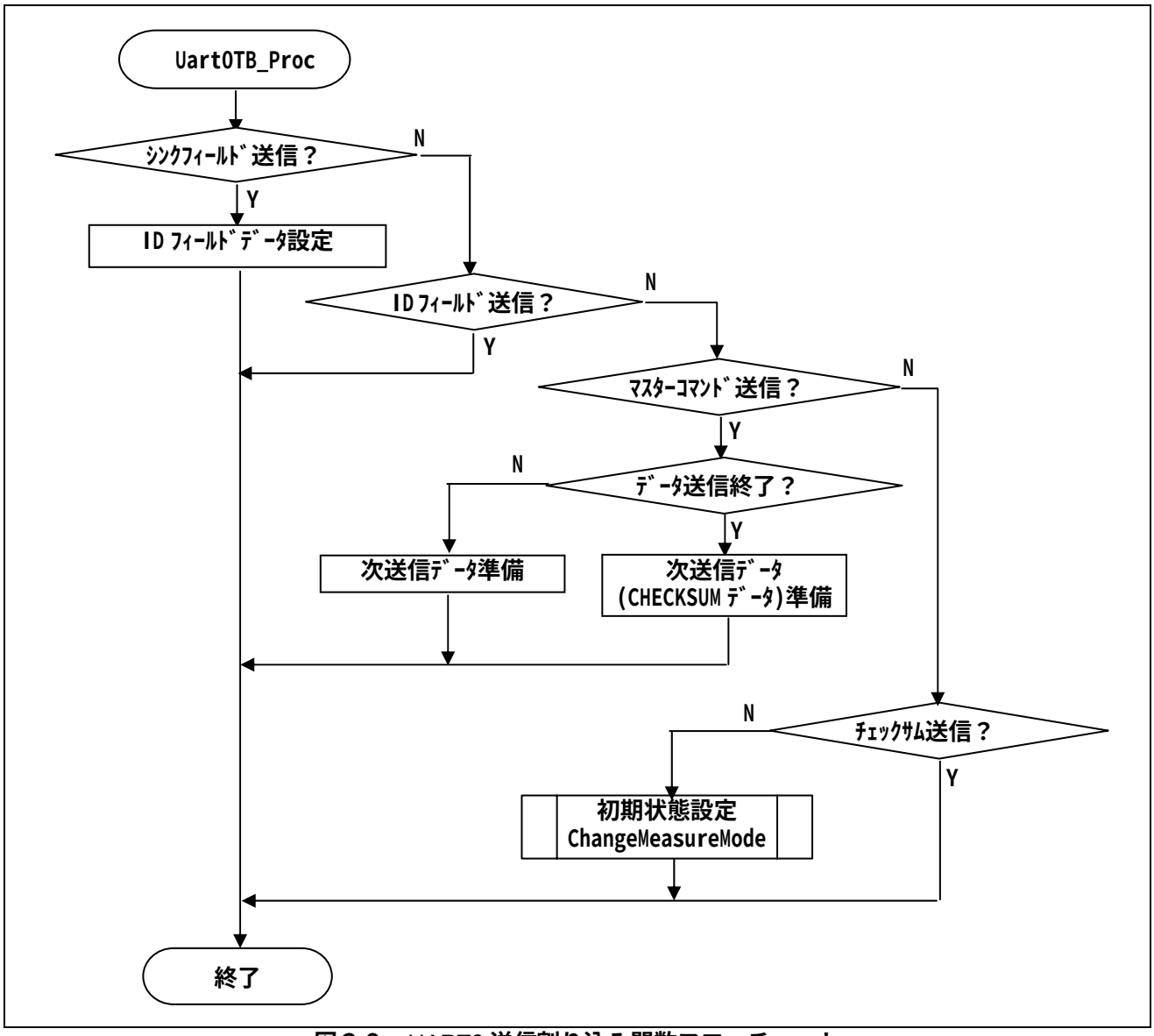


図２０　UART0 送信割り込み関数フローチャート

```
#pragma INTERRUPT /B UartOTB_Proc
void UartOTB_Proc(void)
{
    switch(uc_cHaswStatus){
    case enSYNCH_SYNCH:     // Synch フィールド送信モード
        uc_sUART0HighBuffer = 0x00;
        uc_sUART0LowBuffer = uc_bHaswBuff[enHASW_ID];
        u0tb = ui_sUART0Buffer;
        break;
    case enIDENT:           // Ident 送信モード
        break;
    case enMASTERCMD:       // Command 送信モード
        uc_sUART0HighBuffer = 0x00;
```

```
            if ((uc_cTransPtr) >= (uc_cTransCnt + enHASW_DATA0)) {
                uc_sUART0LowBuffer = uc_bHaswBuff[enHASW_CHECKSUM];
            }
            else uc_sUART0LowBuffer = uc_bHaswBuff[uc_cTransPtr];
            u0tb = ui_sUART0Buffer;
            break;
        case enCHECKSUM:    // Checksum 送受信モード
            break;
        default:
            ChangeMeasureMode(enHASW_INIT);                         // 初期状態設定
            break;
        }
    }
```

2.6.11 UART 受信割り込み関数

UART0 受信エラー割り込み，UART0 受信割り込み処理を行います。

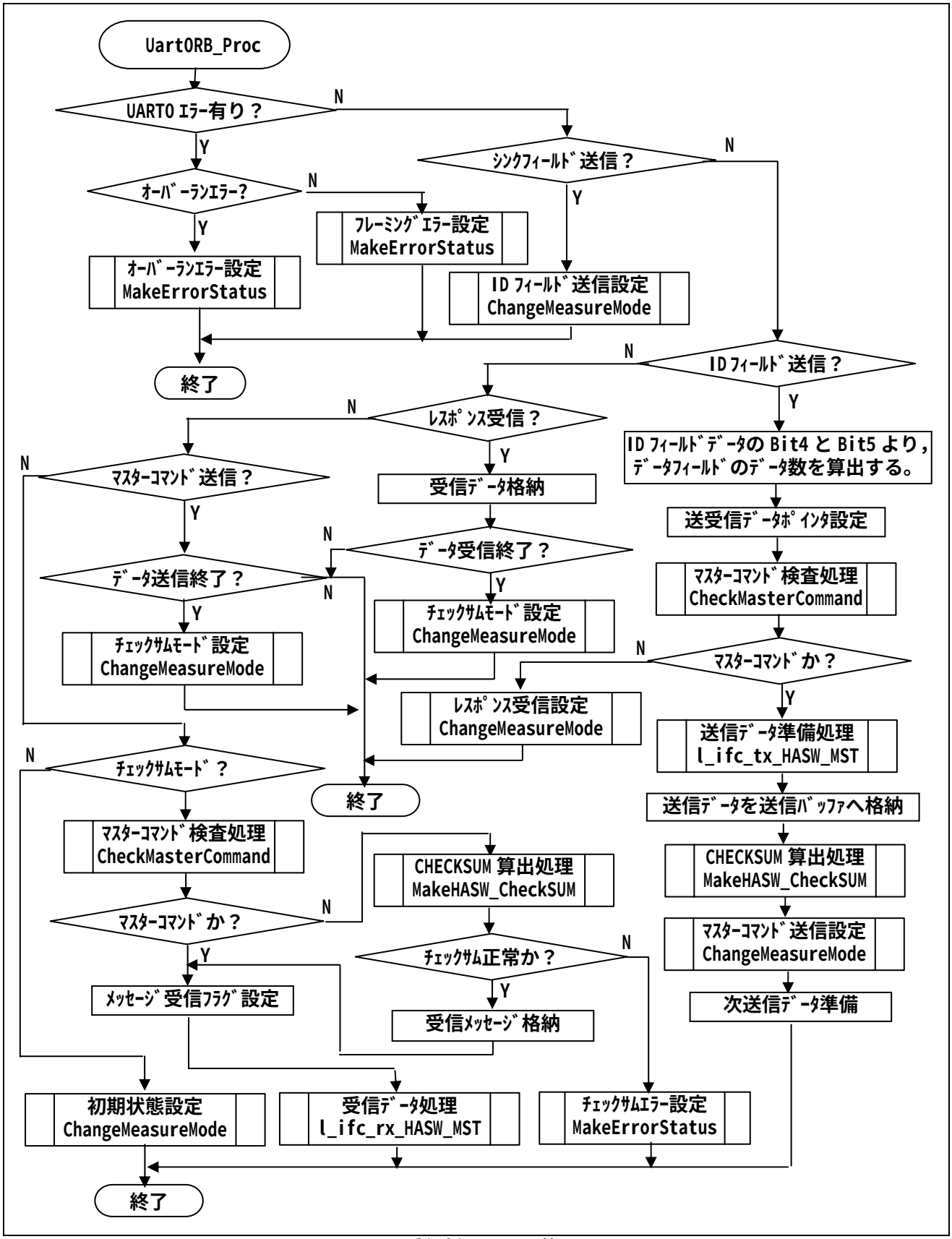


図２１　UART0 受信割り込み関数フローチャート

```
#pragma INTERRUPT /B Uart0RB_Proc
void Uart0RB_Proc(void)
{
    unsigned char    c;

    ui_sUART0Buffer = u0rb;
    if (uc_sUART0HighBuffer&0x80) {
        if (uc_sUART0HighBuffer&0x10) {
            MakeErrorStatus(enERROR_OVER_RUN);            // オーバーランエラー設定
        }
        else {
            MakeErrorStatus(enERROR_FRAME);               // フレーミングエラー設定
        }
        return;
    }

    switch(uc_cHaswStatus){
    case enSYNCH_SYNCH:    // Synch フィールド送信モード
        ChangeMeasureMode(enIDENT);
        break;
    case enIDENT:          // Ident 送信モード
        uc_sUART0LowBuffer = uc_bHaswBuff[enHASW_ID];
        if (un_sUART0.StBIT.Bit5 == 1) {
            if (un_sUART0.StBIT.Bit4 == 1) uc_cTransCnt = 8;
            else uc_cTransCnt = 4;
        }
        else uc_cTransCnt = 2;
        uc_cTransPtr = enHASW_DATA0;
        if (CheckMasterCommand() == TRUE) {        // マスター送信コマンドか？
            l_ifc_tx_HASW_MST();                   // l_ifc_tx_HASW_MST 関数呼び出し
            for (c=0;c<uc_cTransCnt;++c) {
                uc_bHaswBuff[c+enHASW_DATA0] = uc_bHASW_tx_data[c];
            }
            uc_bHaswBuff[enHASW_CHECKSUM] = MakeHASW_CheckSUM(uc_cTransCnt);
            ChangeMeasureMode(enMASTERCMD);
            uc_sUART0HighBuffer = 0x00;
            uc_sUART0LowBuffer = uc_bHaswBuff[uc_cTransPtr++];
            u0tb = ui_sUART0Buffer;
        }
        else {
            ChangeMeasureMode(enRESPONSE);
        }
        break;
    case enRESPONSE:    // Response 受信モード
        uc_bHaswBuff[uc_cTransPtr] = uc_sUART0LowBuffer;
        if (++uc_cTransPtr >= (uc_cTransCnt + enHASW_DATA0)) ChangeMeasureMode(enCHECKSUM);
        break;
    case enMASTERCMD:    // Command 送信モード
        if (uc_cTransPtr++ >= (uc_cTransCnt + enHASW_DATA0)) ChangeMeasureMode(enCHECKSUM);
        break;
    case enCHECKSUM:    // Checksum 送受信モード
        if (CheckMasterCommand() == FALSE) {           // マスター送信コマンドではない？
            uc_bHaswBuff[enHASW_CHECKSUM] = uc_sUART0LowBuffer;
            if (uc_bHaswBuff[enHASW_CHECKSUM] == MakeHASW_CheckSUM(uc_cTransCnt)) {
                for(c=0;c<uc_cTransCnt;c++) {          // 受信データ格納
                    uc_bHASW_rx_data[c] = uc_bHaswBuff[c+enHASW_DATA0];
                }
```

```
            }
            else {
                MakeErrorStatus(enERROR_CHECKSUM);          // チェックサムエラー設定
                break;
            }
        }
        ui_sHASW_status.StBYTE.Byte1 = 0x01;              // メッセージ正常送受信フラグ設定
        l_ifc_rx_HASW_MST();                              // l_ifc_rx_HASW_MST 関数呼び出し
        break;
    default:
        ChangeMeasureMode(enHASW_INIT);                             // 初期状態設定
        break;
    }
}
```

2.6.12 通信状態制御関数

要求された一線式通信の状態の設定処理を行います。

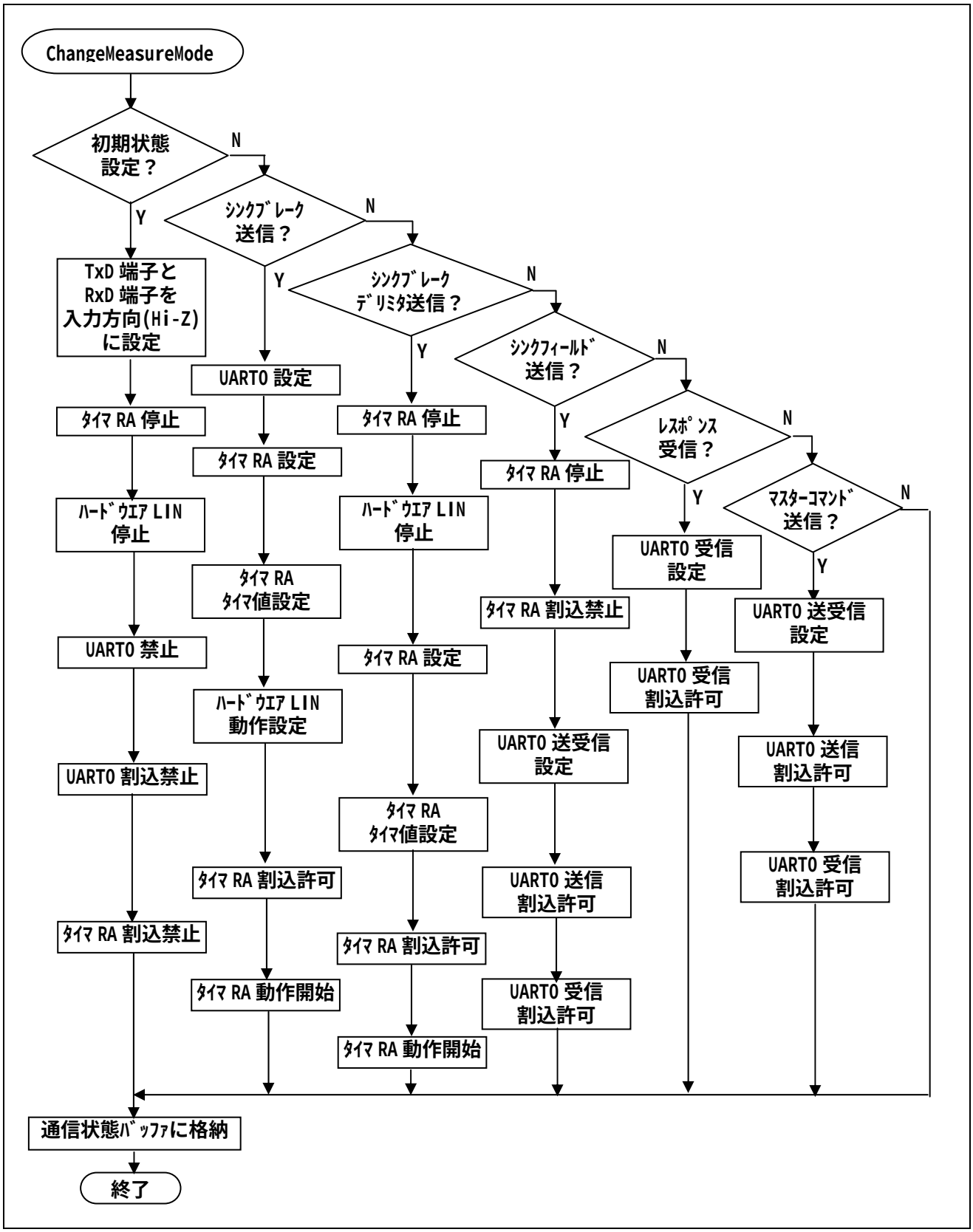


図２２　通信状態制御関数フローチャート

```
void ChangeMeasureMode(unsigned char mode)
{
    unsigned char TimerRALevel,Uart0Level,SynBrkHigh;

    Uart0Level = (ui_sHASW_status.StBYTE.Byte0 & 0xe0) >> 5;
    TimerRALevel = Uart0Level;
    SynBrkHigh = (ui_sHASW_status.StBYTE.Byte0 & 0x0c) >> 2;

    switch(mode){
    case enHASW_INIT:        // 初期状態モード
        pdTxD = P_IN;                                   /* TxD Hi-Z */
        pdRxD = P_IN;                                   /* RxD Hi-Z */
        tracr = 0x00;                                   /* TimerRA 動作停止 */
        while(tcstf_tracr==1);
        tramr = 0x00;                                   /* タイマーモード設定 */
        traioc = 0x00;                                  /* タイマーモード設定 */
        lincr = 0x00;                                   /* LIN モード停止 */
        u0mr = 0x00;                                   /* UART モード禁止 */
        u0c0 = 0x20;                                   /* TxD 端子 Nch-O.D.設定 */
        u0c1 = 0x00;                                   /* UART0 送受信禁止 */
        traic = 0x00;                                   /* TimerRA 割り込み禁止 */
        s0tic = 0x00;                                   /* UART0 送信割り込み禁止 */
        s0ric = 0x00;                                   /* UART0 受信割り込み禁止 */
        break;
    case enSYNCH_BREAK:         // SynchBreak 期間送信モード
        u0mr = 0x05;                                   /* UART モード、8 ビット、1ST、NP */
        tramr = 0x00;                                   /* タイマーモード設定 */
#ifdef __OVER_10M_Hz    // OSC が 10MHz 以上の場合，TimerRA のソースは 2/f
        tramr = 0x30;                                   /* f2、タイマーモード設定 */
#else                    // OSC が 10MHz 以下の場合，TimerRA のソースは 1/f
        tramr = 0x00;                                   /* f1、タイマーモード設定 */
#endif
        tedgsel_traioc = 1;                             /* "L"出力設定 */
        tiosel_traioc = 1;                              /* P1_5 出力 */
#ifdef __AUTO_CALCURATION        // 転送時間自動計算モード？
        trapre = uc_cTimerPreRA[0];
        tra = uc_cTimerRA[0];
#else
        trapre = __SYNBRK_TM_PRE - 1;
        tra = __SYNBRK_TM - 1;
#endif
        line_lincr = 1;                                 /* LIN モード設定 */
        mst_lincr = 1;                                 /* MST モード設定 */
        sbie_lincr = 1;                                 /* SynchBreak 検出設定 */
        linst = 0x38;                                  /* LIN ステータスフラグクリア */
        traic = TimerRALevel;                          /* TimerRA 割り込み許可 */
        tstart_tracr = 1;                              /* TimerRA 動作開始 */
        while(tcstf_tracr==0);
        break;
    case enSYNCH_MEASURE:    // SynchBreak デリミタ期間送信モード
        tracr = 0x00;                                   /* TimerRA 動作停止 */
        while(tcstf_tracr==1);
        lincr = 0x00;                                   /* LIN モード停止 */
#ifdef __OVER_10M_Hz    // OSC が 10MHz 以上の場合，TimerRA のソースは 2/f
        tramr = 0x30;                                   /* f2、タイマーモード設定 */
#else                    // OSC が 10MHz 以下の場合，TimerRA のソースは 1/f
        tramr = 0x00;                                   /* f1、タイマーモード設定 */
```

```
#endif
        traioc = 0x00;                                  /* タイマーモード設定 */
#ifdef __AUTO_CALCURATION       // 転送時間自動計算モード？
        tra = uc_cTimerRA[1];
        trapre = uc_cTimerPreRA[1];
        for (;SynBrkHigh>0;--SynBrkHigh) tra = tra + uc_cTimerRA[1];
#else
        tra = __TBIT_TM;
        trapre = __TBIT_TM_PRE - 1;
        for (;SynBrkHigh>0;--SynBrkHigh) tra = tra + __TBIT_TM;
#endif
        --tra;
        traic = TimerRALevel;                           /* TimerRA 割り込み許可 */
        tstart_tracr = 1;                               /* TimerRA 動作開始 */
        while(tcstf_tracr==0);
        break;
    case enSYNCH_SYNCH:         // Synch フィールド送信モード
        tracr = 0x00;                                   /* TimerRA 動作停止 */
        while(tcstf_tracr==1);
        traic = 0x00;                                   /* TimerRA 割り込み禁止 */
        u0mr = 0x05;                                    /* UART モード、8 ビット、1ST、NP */
        if (un_sU0BRGdata.StBYTE.Byte1 != 0x00) {
            u0c0 = 0x21;                                /* LSB,f8,TxD 端子 Nch-O.D.設定 */
        }
        else {
            u0c0 = 0x20;                                /* LSB,f1,TxD 端子 Nch-O.D.設定 */
        }
        u0c1 = 0x15;                                    /* UART0 送受信許可,送信完了時割り込み */
        u0brg = un_sU0BRGdata.StBYTE.Byte0 - 1;         /*  */
        s0ric = Uart0Level;                             /* UART0 受信割り込み許可 */
        s0tic = Uart0Level;                             /* UART0 送信割り込み許可 */
    case enIDENT:             // Ident 送信モード
        break;
    case enRESPONSE:          // Response 受信モード
        u0c1 = 0x14;                                    /* UART0 受信許可,送信完了時割り込み */
        s0ric = Uart0Level;                             /* UART0 受信割り込み許可 */
        s0tic = 0x00;                                   /* UART0 送信割り込み禁止 */
        break;
    case enMASTERCMD:         // Command 送信モード
        u0c1 = 0x15;                                    /* UART0 送受信許可,送信完了時割り込み */
        s0tic = Uart0Level;                             /* UART0 送信割り込み許可 */
        s0ric = Uart0Level;                             /* UART0 受信割り込み許可 */
        break;
    case enCHECKSUM:          // Checksum 送受信モード
    default:
        break;
    }
    uc_cHaswStatus = mode;
}
```

2.6.13 ID パリティ計算処理関数

ID フィールドデータのパリティ値を計算処理を行います。

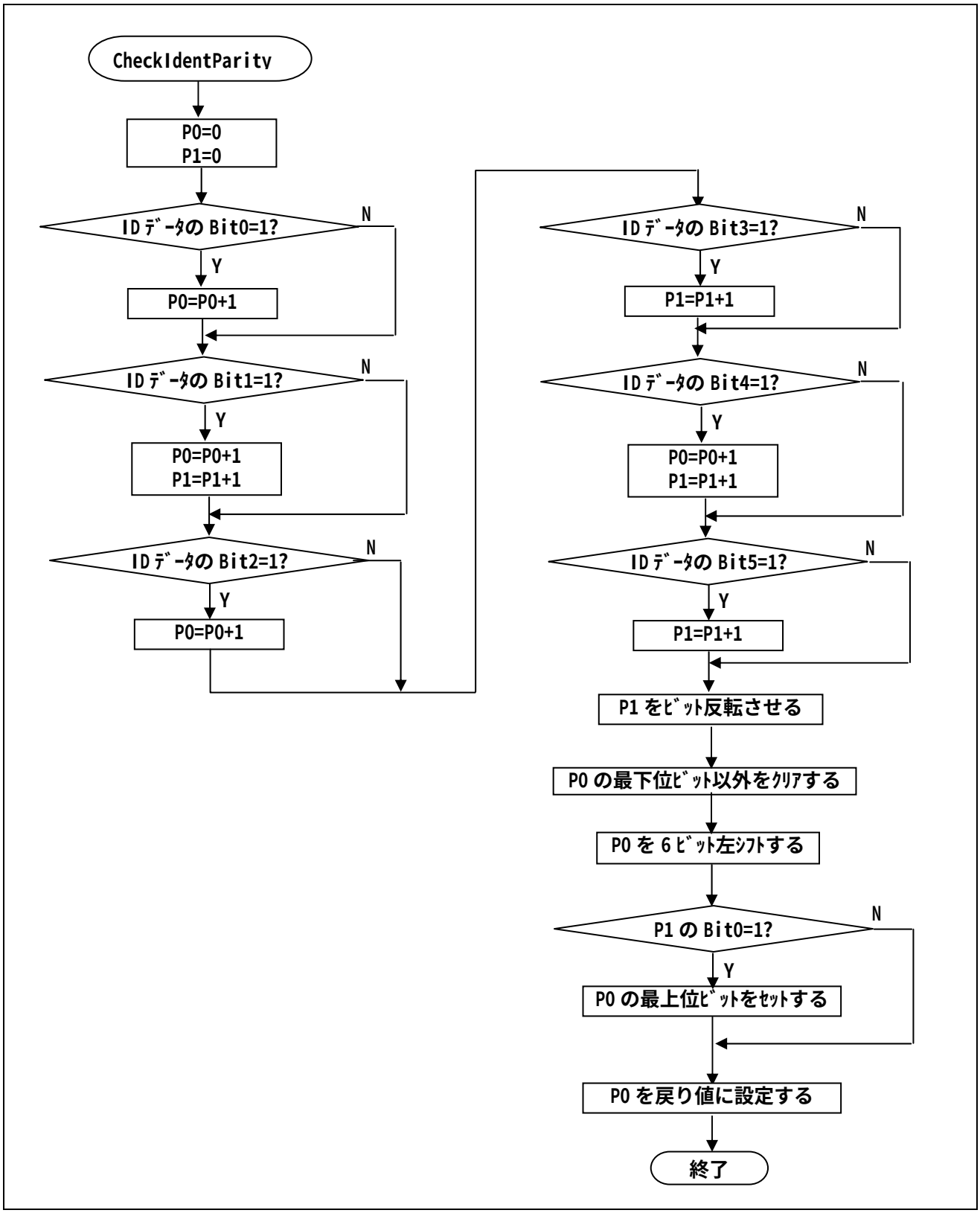


図２３　ID フィールドパリティ計算関数フローチャート

```
unsigned char CheckIdentParity(unsigned char ident_data)
{
    unsigned char    uc_p0,uc_p1;

    uc_p0 = uc_p1 = 0;
    if ((ident_data & 0x01) == 0x01) ++uc_p0;
    if ((ident_data & 0x02) == 0x02) {
        ++uc_p0;
        ++uc_p1;
    }
    if ((ident_data & 0x04) == 0x04) ++uc_p0;
    if ((ident_data & 0x08) == 0x08) ++uc_p1;
    if ((ident_data & 0x10) == 0x10) {
        ++uc_p0;
        ++uc_p1;
    }
    if ((ident_data & 0x20) == 0x20) ++uc_p1;
    uc_p1 = ~uc_p1;
    uc_p0 &= 0x01;
    uc_p0 <<= 6;
    if ((uc_p1&0x01) == 0x01) uc_p0 += 0x80;
    return uc_p0;
}
```

2.6.14 マスターコマンド検査処理関数

送信する ID がマスターコマンドかどうかを検査する処理を行います。

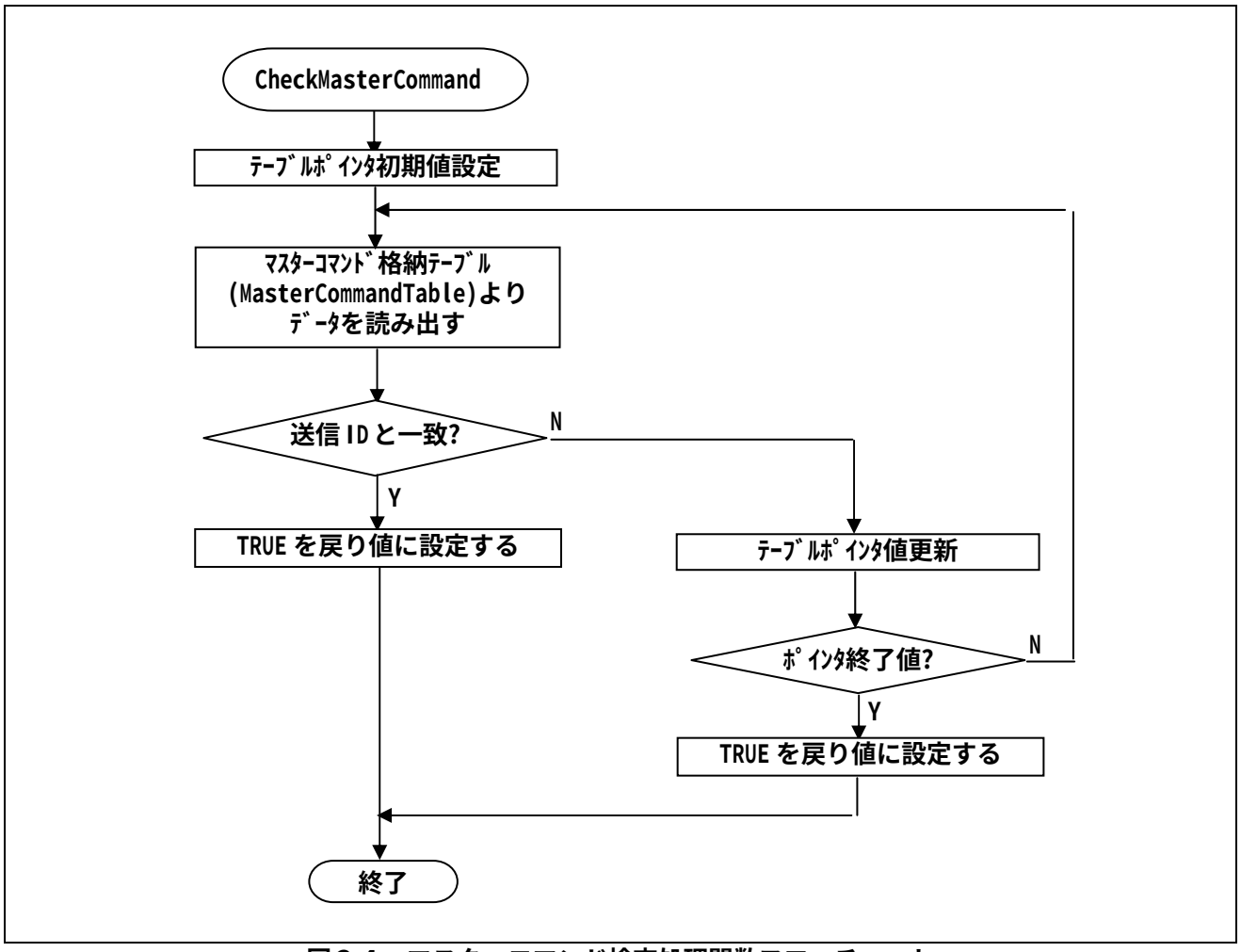


図２４　マスターコマンド検査処理関数フローチャート

```
unsigned char CheckMasterCommand(void)
{
    unsigned char    c;

    for (c=0;c < (unsigned char )(sizeof MasterCommandTable);++c) {
        if (uc_bHaswBuff[enHASW_ID] == MasterCommandTable[c]) {
            return TRUE;
        }
    }
    return FALSE;
}
```

2.6.15 チェックサム値計算処理関数

送受信するデータフィールドのチェックサム値を計算する処理を行います。

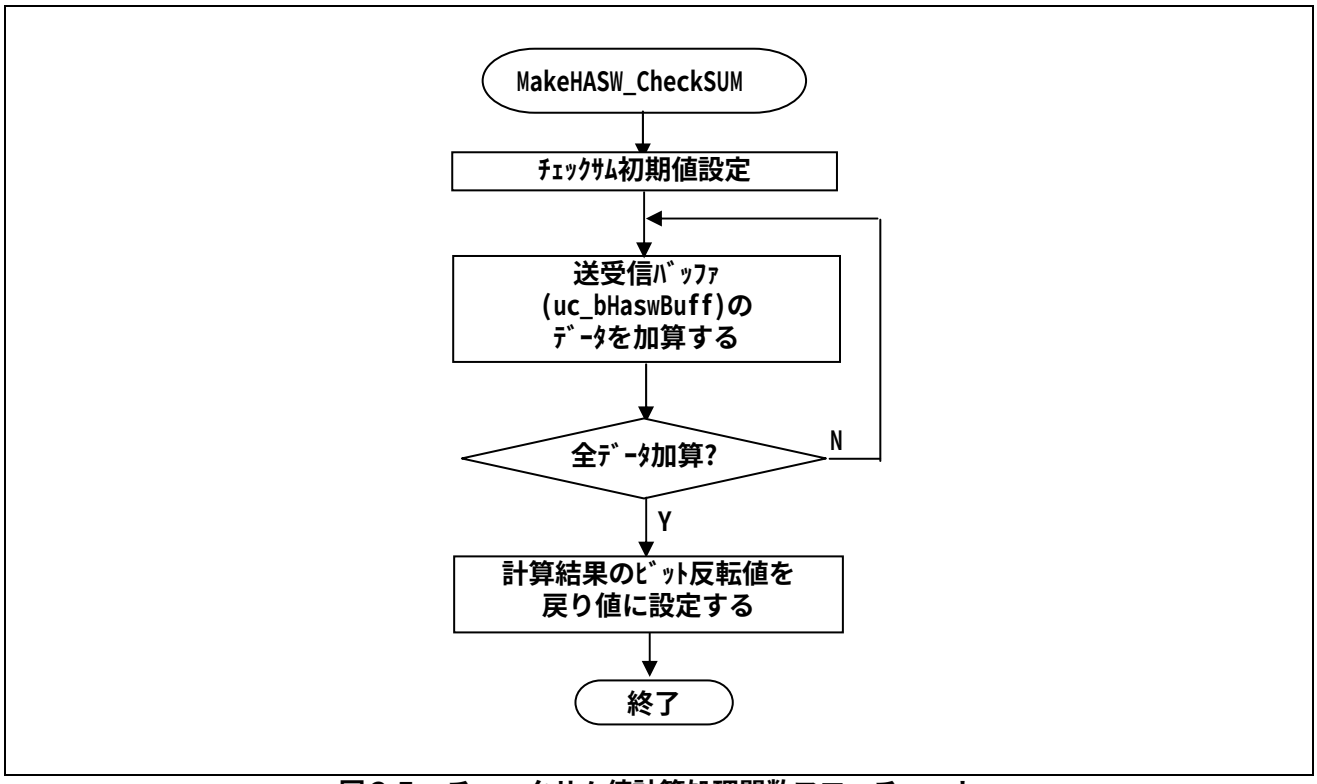


図２５　チェックサム値計算処理関数フローチャート

```
unsigned char MakeHASW_CheckSUM(unsigned char cnt)
{
    unsigned char    c;
    UnWORD    sum;

    sum.uiAll = 0;
    for(c=0;c<cnt;++c){
        sum.uiAll += (unsigned int )uc_bHaswBuff[c+enHASW_DATA0];
        sum.StBYTE.Byte0 += sum.StBYTE.Byte1;
        sum.StBYTE.Byte1 = 0;
    }
    return ~sum.StBYTE.Byte0;
}
```

2.6.16 通信エラー処理関数

一線式通信においてエラーが発生した場合の処理を行います。

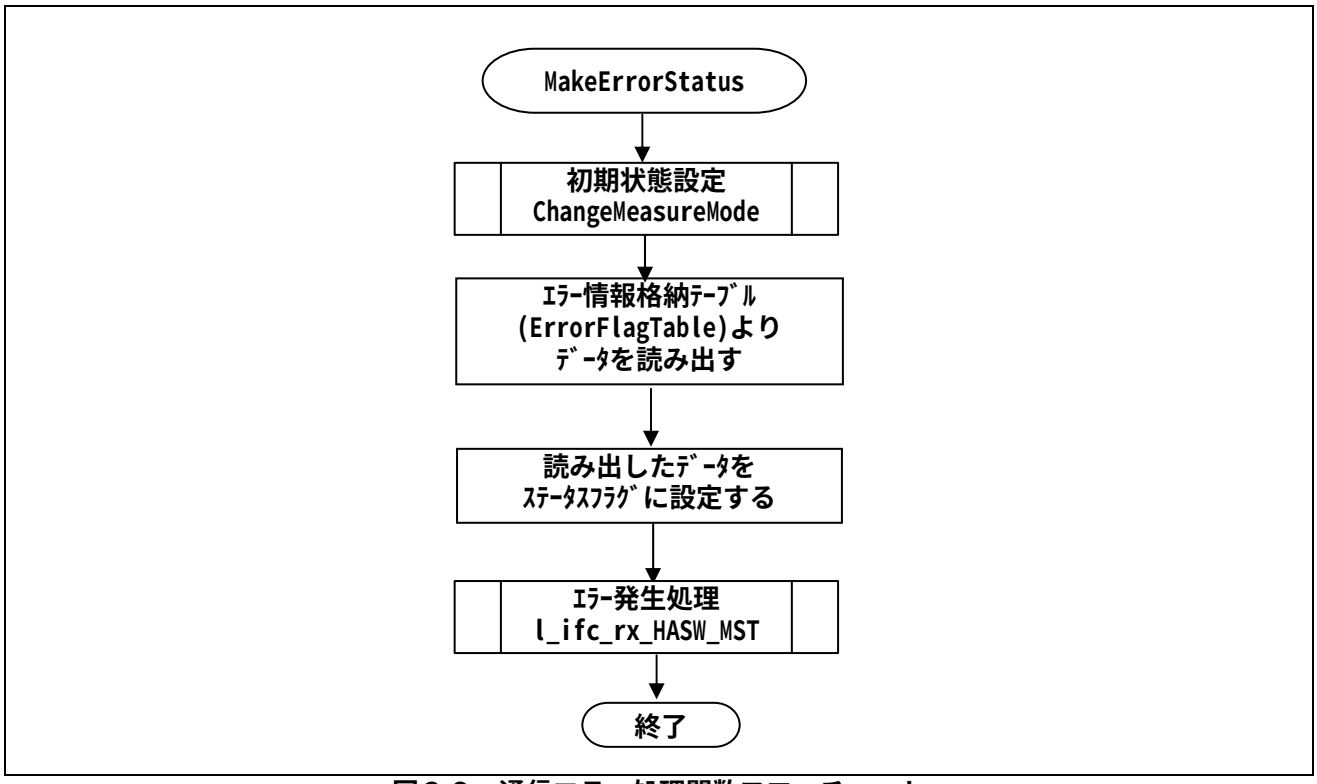


図２６　通信エラー処理関数フローチャート

```
void MakeErrorStatus(unsigned char ErrorCode)
{
    const unsigned char ErrorFlagTable[enERROR_MODE]={0x06,0x0a,0x12,0x22,0x42,0x82};

    ChangeMeasureMode(enHASW_INIT);                          // 初期状態設定
    ui_sHASW_status.StBYTE.Byte1 = ErrorFlagTable[ErrorCode];    // エラーフラグ設定
    l_ifc_rx_HASW_MST();                                  // l_ifc_rx_HASW_MST 関数呼び出し
}
```

3. ライブラリソフトウエア仕様スレーブノード編

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，R8C/Tiny の内蔵機能を使用し，スレーブノードとして一線式通信を行います。

3.1 動作環境

- 使用デバイス：R8C/18 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，一線式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，HASW_SLV.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８に示します。

表８　R8C/18 内蔵周辺機能使用表

機能		端子機能（端子番号）	用途	説明
UART0	送信	TxD0(13)	レスポンスフレーム送信	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	RxD0(12)	シンクフィールド受信 ID フィールド受信 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 X INT1		CNTR01/INT11 (12)	シンクブレークフィールド ドミナント期間計測 シンクフィールド期間計測	Φosc 周期でカウント動作し，各時間間隔を計測

3.2 ファイル構成

- HASW_SLV.c（Ver1.00）

  R8C/18 グループにおける一線式通信（スレーブ）マイコン機能設定，および通信制御を行うための C ソースファイルです。

- HASW_SLV.h（Ver1.00）

  ユーザーにより通信転送速度の設定や ID の設定などを行い HASW_SLV.c（Ver1.00）コンパイル時に組み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み込む必要があります。

- SFR_R819.h

  R8C/18 グループ用内蔵 I/O レジスタ定義ファイルです。

3.3 ROM/RAM 使用容量

（R8C シリーズ C コンパイラ M3T-NC30WAVer5.40R0 使用時）

ROM/RAM 使用容量は，ID 設定数などにより変化します。

- ROM 容量：約 1KByte(自動速度補正なし，オプティマイズ無し)，約 1KByte(自動速度補正有り，オプティマイズ有り：ROM サイズ重視)
- RAM 容量：36Byte

3.4 機能仕様

3.4.1 一線式通信仕様

- ノード：スレーブノードに対応
- ID：ユーザ定義 ID

レスポンス受信 ID
HASW_SLV.h より 0 個から 60 個（00h～3b,3dh）まで設定可能。
（但し，同一シングルワイヤバス上に同じ ID を受け付けるノードを設定すると正常動作しません）
一線式通信プロトコル定義 ID

a) マスターリクエストフレーム ID　3ch（ID フィールドデータ：3Ch）
マスターノードからレスポンスフレーム（データ長８バイト）が送信されます。

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）
この ID を持つスレーブノードがレスポンスフレーム（データ長８バイト）を送信します。

c) 拡張フレーム ID3eh，3fh（ID フィールドデータ：FEh，BFh）
本ライブラリ（Ver1.00）では，対応しておりません。

ID 設定方法
HASW_SLV.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～3b,3dh））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 HASW_SLV.c をコンパイルします。

- レスポンスデータ長：受信した ID フィールドデータ内 DLC（データレングスコントロール）ビットにより判定。

ID フィールドデータ	= 00h～1fh	：	２バイト
	= 20h～2fh	：	４バイト
	= 30h～3dh	：	８バイト

- 通信転送速度：HASW_SLV.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 UART0 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。詳しくはハードウエアマニュアルのクロック非同期式シリアル I/O（UART0）モード：ビットレートに対する U0BRG 設定例（UART モード）をご参考ください。

  なお, 自動通信速度補正機能を使用される場合は, INT1 割り込み処理関数内にて通信速度の自動判定, 計算を行いますので，処理時間がかかります。

- ウェイクアップ信号送受信：本ライブラリ（Ver1.00）では，対応しておりません。

3.4.2 HASW_SLV.h ファイル設定例

HASW_SLV.h の設定例を示します。

e) 使用するマイコンは R8C/18 グループとする。

f) 以下の 4 個の ID に対してレスポンスフレームを送信する。

ID（ID ビット＋DLC ビット）	（パリティビット含む）
02h	42h
13h	D3h
24h	64h
35h	F5h

g) システム搭載クロック（φosc）を 8[MHz](内蔵 OSC)とする。

h) 一線式通信転送速度を 19200[bit/sec]とし，通信速度の自動補正（±5%）をする。

上記 a)～d)の仕様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下さい）

```
/*""FILE COMMENT""*************************************************************************
 * File Name    : hasw_slv.h
 * Version      : 1.00
 * Contents     : 一線式通信ユーザ定義ファイル
 *""FILE COMMENT END""*********************************************************************/
#ifdef REAL_MEM                                     /* この行は変更または削除しないでください */
    #define     GLOBAL                            /* この行は変更または削除しないでください */
#else                                               /* この行は変更または削除しないでください */
    #define     GLOBAL      extern                /* この行は変更または削除しないでください */
#endif                                              /* この行は変更または削除しないでください */
#define     SBDATA_ON                   /* SB 相対アドレッシングを使用する場合はこの行を定義 */
/*************************************************************************************************/
/*                                                                                               */
/*     CPU 選択                                                                                  */
/*                                                                                               */
/*===============================================================================================*/
#define     __CPU_R8C_18                            /* R8C/18 グループの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           /* 2 バイトレスポンスデータ送信 ID を持つ場合はこの行を定義 */
#ifdef      __Res2Byte_ID
//      #define     __ID00      0x80                /*                                           */
//      #define     __ID01      0xC1                /*                                           */
    #define     __ID02      0x42                /* ID フィールド 42h に対してレスポンスを送信 */
//      #define     __ID03      0x03                /*                                           */
//      #define     __ID04      0xC4                /*                                           */
//      #define     __ID05      0x85                /*                                           */
```

```
//    #define    __ID06    0x06              /*                                  */
//    #define    __ID07    0x47              /*                                  */
//    #define    __ID08    0x08              /*                                  */
//    #define    __ID09    0x49              /*                                  */
//    #define    __ID10    0xCA              /*                                  */
//    #define    __ID11    0x8B              /*                                  */
//    #define    __ID12    0x4C              /*                                  */
//    #define    __ID13    0x0D              /*                                  */
//    #define    __ID14    0x8E              /*                                  */
//    #define    __ID15    0xCF              /*                                  */
//    #define    __ID16    0x50              /*                                  */
//    #define    __ID17    0x11              /*                                  */
//    #define    __ID18    0x92              /*                                  */
    #define    __ID19    0xD3              /* ID フィールド D3h に対してレスポンスを送信 */
//    #define    __ID20    0x14              /*                                  */
//    #define    __ID21    0x55              /*                                  */
//    #define    __ID22    0xD6              /*                                  */
//    #define    __ID23    0x97              /*                                  */
//    #define    __ID24    0xD8              /*                                  */
//    #define    __ID25    0x99              /*                                  */
//    #define    __ID26    0x1A              /*                                  */
//    #define    __ID27    0x5B              /*                                  */
//    #define    __ID28    0x9C              /*                                  */
//    #define    __ID29    0xDD              /*                                  */
//    #define    __ID30    0x5E              /*                                  */
//    #define    __ID31    0x1F              /*                                  */
#endif

/*==================================================================================*/
/*      4 バイトデータ                                                              */
/*----------------------------------------------------------------------------------*/
#define    __Res4Byte_ID          /* 4 バイトレスポンスデータ送信 ID を持つ場合はこの行を定義 */
#ifdef     __Res4Byte_ID
//    #define    __ID32    0x20              /*                                  */
//    #define    __ID33    0x61              /*                                  */
//    #define    __ID34    0xE2              /*                                  */
//    #define    __ID35    0xA3              /*                                  */
    #define    __ID36    0x64              /* ID フィールド 64h に対してレスポンスを送信 */
//    #define    __ID37    0x25              /*                                  */
//    #define    __ID38    0xA6              /*                                  */
//    #define    __ID39    0xE7              /*                                  */
//    #define    __ID40    0xA8              /*                                  */
//    #define    __ID41    0xE9              /*                                  */
//    #define    __ID42    0x6A              /*                                  */
//    #define    __ID43    0x2B              /*                                  */
//    #define    __ID44    0xEC              /*                                  */
//    #define    __ID45    0xAD              /*                                  */
//    #define    __ID46    0x2E              /*                                  */
//    #define    __ID47    0x6F              /*                                  */
#endif
```

```
/*=========================================================================================*
/
/*      8 バイトデータ                                                                     */
/*-----------------------------------------------------------------------------------------*/
#define     __Res8Byte_ID           /* 8 バイトレスポンスデータ送信 ID を持つ場合はこの行を定義 */
#ifdef      __Res8Byte_ID
//      #define     __ID48      0xF0            /*                                        */
//      #define     __ID49      0xB1            /*                                        */
//      #define     __ID50      0x32            /*                                        */
//      #define     __ID51      0x73            /*                                        */
//      #define     __ID52      0xB4            /*                                        */
    #define     __ID53      0xF5            /* ID フィールド F5h に対してレスポンスを送信 */
//      #define     __ID54      0x76            /*                                        */
//      #define     __ID55      0x37            /*                                        */
//      #define     __ID56      0x78            /*                                        */
//      #define     __ID57      0x39            /*                                        */
//      #define     __ID58      0xBA            /*                                        */
//      #define     __ID59      0xFB            /*                                        */
    #define     __ID61      0x7D            /* ID フィールド 7Dh に対してレスポンスを送信 */
#endif
/*****************************************************************************************/
/*                                                                                       */
/*      システム搭載クロック定義                                                         */
/*                                                                                       */
/*=======================================================================================*/
//#define     __OSC_Hz      20000000                  /* φosc = 20.000MHz → 20000000 */
#define     __OSC_Hz      8000000                   /* φosc =  8.000MHz →  8000000 */
//#define     __OVER_10M_Hz                   /* φosc >= 10.000MHz の場合はこの行を定義*/
/*****************************************************************************************/
/*                                                                                       */
/*      通信転送速度ボーレート定義                                                       */
/*                                                                                       */
/*=======================================================================================*/
//#define     __B_rate      2400                      /*  2400bps →  2400 */
//#define     __B_rate      9600                      /*  9600bps →  9600 */
#define     __B_rate      19200                     /* 19200bps → 19200 */
#define     __BPS_AUTO                      /* 通信速度の自動補正を行う場合はこの行を定義*/
/*****************************************************************************************/
/*                                                                                       */
/*      関数・変数定義          以下は変更または削除しないでください                     */
/*                                                                                       */
/*=======================================================================================*/
/*      レスポンスデータ送信の有無定義      以下は変更または削除しないでください        */
/*-----------------------------------------------------------------------------------------*/
#ifndef     __RESPONSE
    #ifdef      __Res2Byte_ID
        #define     __RESPONSE
    #endif
#endif
#ifndef     __RESPONSE
```

```
    #ifdef    __Res4Byte_ID
        #define    __RESPONSE
    #endif
#endif
#ifndef    __RESPONSE
    #ifdef    __Res8Byte_ID
        #define    __RESPONSE
    #endif
#endif

/*=========================================================================================*/
/*     関数定義                    以下は変更または削除しないでください                    */
/*-----------------------------------------------------------------------------------------*/
GLOBAL    void    l_sys_init_HASW_SLV(void);
GLOBAL    void    l_ifc_init_HASW_ SLV(void);
GLOBAL    void    l_ifc_connect_HASW_ SLV(void);
GLOBAL    void    l_ifc_disconnect_HASW_ SLV(void);
#ifdef REAL_MEM
    extern    void    l_ifc_rx_HASW_SLV(void);
    extern    void    l_ifc_aux_HASW_SLV(void);
#else
    void    l_ifc_rx_HASW_SLV(void);
    void    l_ifc_aux_HASW_SLV(void);
#endif

#ifdef    __RESPONSE
    #ifdef REAL_MEM
        extern    void    l_ifc_tx_HASW_SLV(void);
    #else
        void    l_ifc_tx_HASW_SLV(void);
    #endif
#endif

/*=========================================================================================*/
/*     変数定義                    以下は変更または削除しないでください                    */
/*-----------------------------------------------------------------------------------------*/
#ifdef    __RESPONSE
    GLOBAL volatile unsigned char    uc_bHASW_tx_data[8];
    #ifdef    SBDATA_ON
        #pragma  SBDATA  uc_bHASW_tx_data
    #endif
#endif
GLOBAL volatile unsigned char    uc_dHASW_rx_id;
GLOBAL volatile unsigned char    uc_bHASW_rx_data[8];
GLOBAL volatile union {
    struct {
        unsigned char    Bit0:1;
        unsigned char    Bit1:1;
        unsigned char    Bit2:1;
        unsigned char    Bit3:1;
```

```
        unsigned char    Bit4:1;
        unsigned char    Bit5:1;
        unsigned char    Bit6:1;
        unsigned char    Bit7:1;
        unsigned char    Bit8:1;
        unsigned char    Bit9:1;
        unsigned char    Bit10:1;
        unsigned char    Bit11:1;
        unsigned char    Bit12:1;
        unsigned char    Bit13:1;
        unsigned char    Bit14:1;
        unsigned char    Bit15:1;
    }StBIT;
    struct {
        unsigned char    Byte0;
        unsigned char    Byte1;
    }StBYTE;
    unsigned int uiAll;
}ui_sHASW_status;
#ifdef    SBDATA_ON
    #pragma  SBDATA  uc_dHASW_tx_id
    #pragma  SBDATA  uc_dHASW_rx_id
    #pragma  SBDATA  uc_bHASW_rx_data
    #pragma  SBDATA  ui_sHASW_status
#endif
```

3.4.3 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，一線式通信制御に必要なマイコン（R8C/18 グループ）内蔵周辺機能の初期化，一線式通信制御の停止・再開レスポンスフレーム送受信を行います。

表９　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
l_sys_init_HASW_SLV	**一線式通信制御に必要なマイコン（R8C/18 グループ）内蔵周辺機能の初期設定を行います。**
l_ifc_init_HASW_ SLV	**一線式通信制御を初期状態にします。** **・通信エラーなどの発生時に一線式通信制御を初期状態に戻したいときに使用します。**
l_ifc_connect_HASW_ SLV	**一線式通信制御を開始状態にし，ヘッダフレームの受信を開始します。**
l_ifc_disconnect_HASW_ SLV	**一線式通信制御を停止状態にします。** **・レスポンスフレーム送受信後に使用します。**

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，一線式通信中の各タイミング（送受信完了時，通信エラー検出時など）での処理を行います。

表１０　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
l_ifc_rx_HASW_SLV	**レスポンスフレーム送受信後のユーザアプリケーション制御関数** **・レスポンスフレーム受信後，レスポンスフレーム送信後，エラー発生時に呼び出されます。イベントの内容については通信ステータスフラグで判定してください。**
l_ifc_tx_HASW_SLV	**ヘッダフレーム送信後のユーザアプリケーション制御関数** **・レスポンスフレームを送信する ID の場合，ヘッダフレーム受信後に呼び出されます。**
l_ifc_aux_HASW_SLV	**シンクフィールド受信後のユーザアプリケーション制御関数** **・シンクフィールドにて同期が取れなかった場合，シンクフィールド受信後に呼び出されます。（この後のメッセージフレームは受信されません）**

- 動作概要

  図２７，図２８に動作概要を示します。

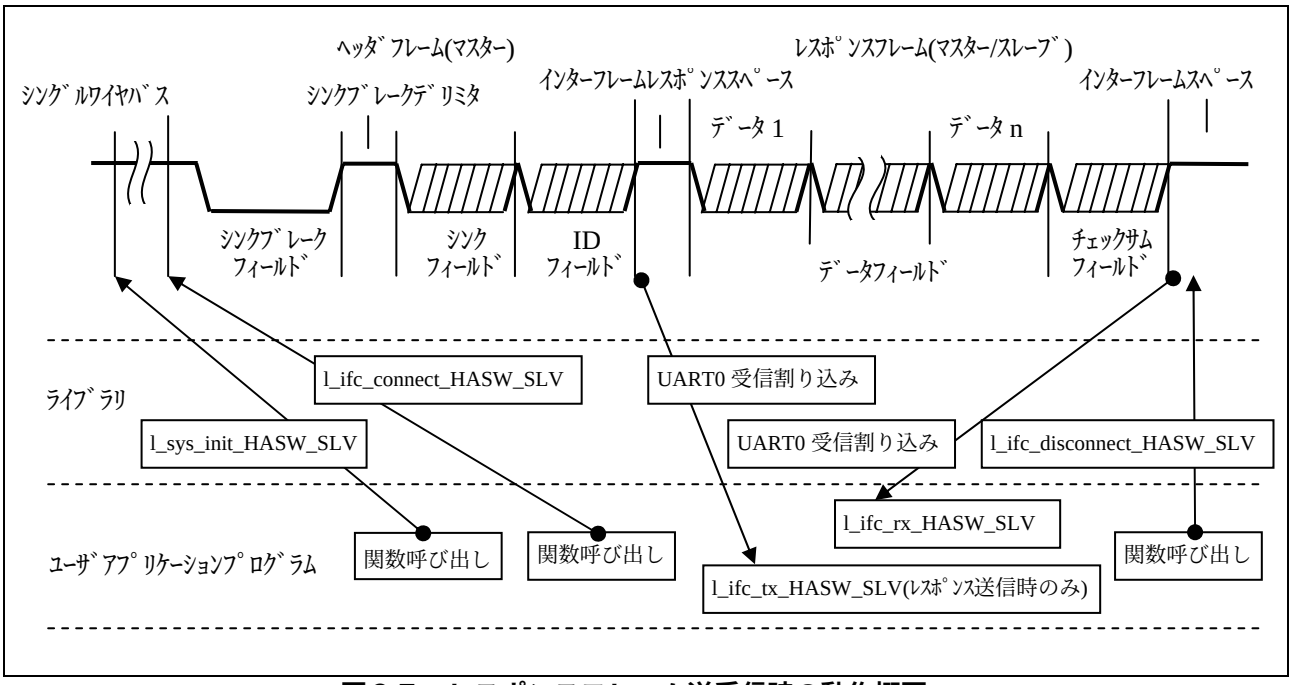


図２７　レスポンスフレーム送受信時の動作概要

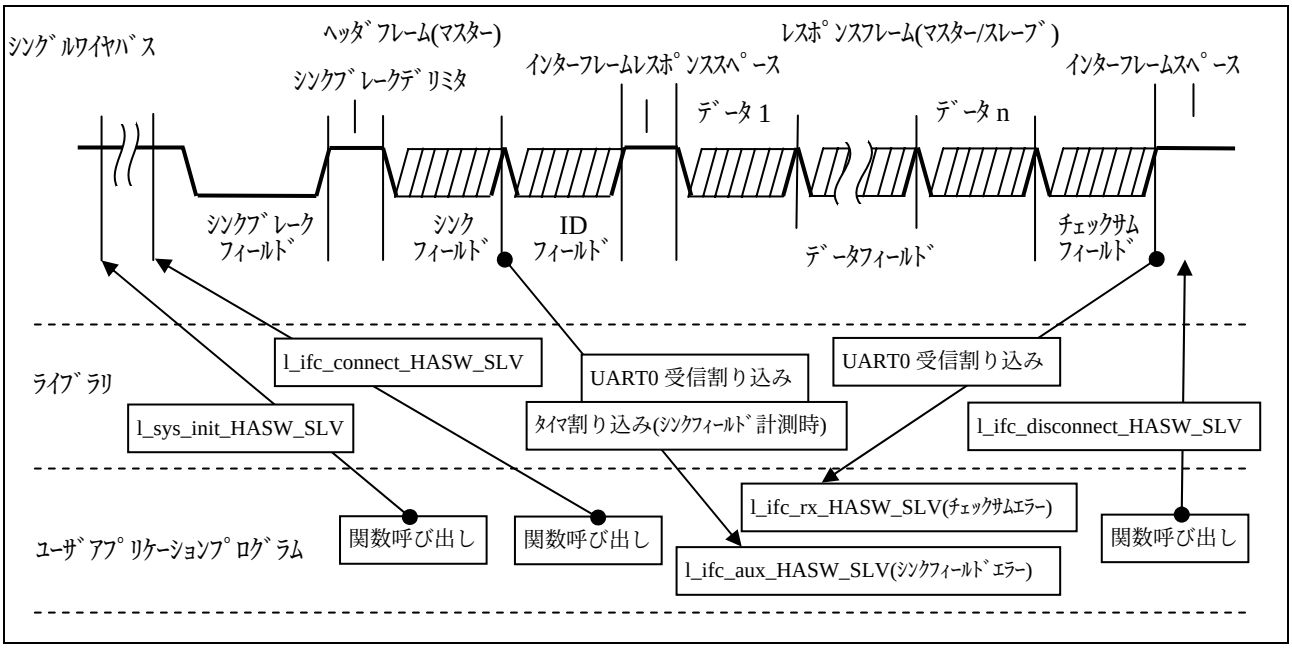


図６　シンクフィールドエラー，チェックサムエラー発生時の動作概要

- 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と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います。

表１１　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
uc_bHASW_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**
uc_dHASW_rx_id	unsigned char	**レスポンスデータ受信時の** ID **を格納**
uc_bHASW_rx_data[0]〜[7]	unsigned char (**配列**)	**受信したレスポンスデータを格納**
ui_sHASW_status	**（構造体）**	**通信ステータス**
ui_sHASW_status.uiAll	**ワードアクセス** unsigned int	
	ビットアクセス	
ui_sHASW_status.StBIT.Bit15	**ビット**-15	**ビットエラーフラグ**
ui_sHASW_status.StBIT.Bit14	**ビット**-14	**フレーミングエラーフラグ**
ui_sHASW_status.StBIT.Bit13	**ビット**-13	**オーバーランエラーフラグ**
ui_sHASW_status.StBIT.Bit12	**ビット**-12	**パリティエラーフラグ**
ui_sHASW_status.StBIT.Bit11	**ビット**-11	**シンクフィールドエラーフラグ**
ui_sHASW_status.StBIT.Bit10	**ビット**-10	**チェックサムエラーフラグ**
ui_sHASW_status.StBIT.Bit9	**ビット**-9	**送受信エラー発生フラグ**
ui_sHASW_status.StBIT.Bit8	**ビット**-8	**メッセージフレーム正常送受信完了フラグ**
ui_sHASW_status.StBIT.Bit7 ui_sHASW_status.StBIT.Bit6 ui_sHASW_status.StBIT.Bit5	**ビット**-7〜5	UART0 **割り込みレベル設定ビット** UART0 **割り込みのレベルを** 000〜111 **に設定する。**
ui_sHASW_status.StBIT.Bit4 ui_sHASW_status.StBIT.Bit3 ui_sHASW_status.StBIT.Bit2	**ビット**-4〜2	INT1 **割り込みレベル設定ビット** INT1 **割り込みのレベルを** 000〜111 **に設定する。** **注）通信転送レート補正機能制御ビットを設定した場合，最優先の割り込みレベルに設定する。**
ui_sHASW_status.StBIT.Bit1	**ビット**-1	**通信転送レート補正機能制御ビット**
ui_sHASW_status.StBIT.Bit0	**ビット**-0	**リザーブビット**

3.5 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。

3.5.1 内蔵周辺機能の初期設定

一線式通信の実行を行うことに先立って，ユーザアプリケーションプログラムから l_sys_init_HASW_SLV 関数を呼び出す事により，マイコンの内蔵周辺機能の初期化を行います。

l_sys_init_HASW_SLV 関数をユーザアプリケーションプログラムから呼び出す時には，予め割り込みレベル設定（ui_sHASW_status.StBIT.Bit7～ui_sHASW_status.StBIT.Bit5），通信転送レート補正機能制御ビット（ui_sHASW_status.StBIT.Bit1）を設定してください。

3.5.2 ヘッダフレームの受信

3.5.2.1 シンクブレークフィールドの送信

タイマ X のパルス幅測定機能(CNTR 端子入力)によりシンクブレークフィールドドミナント期間を計測します。

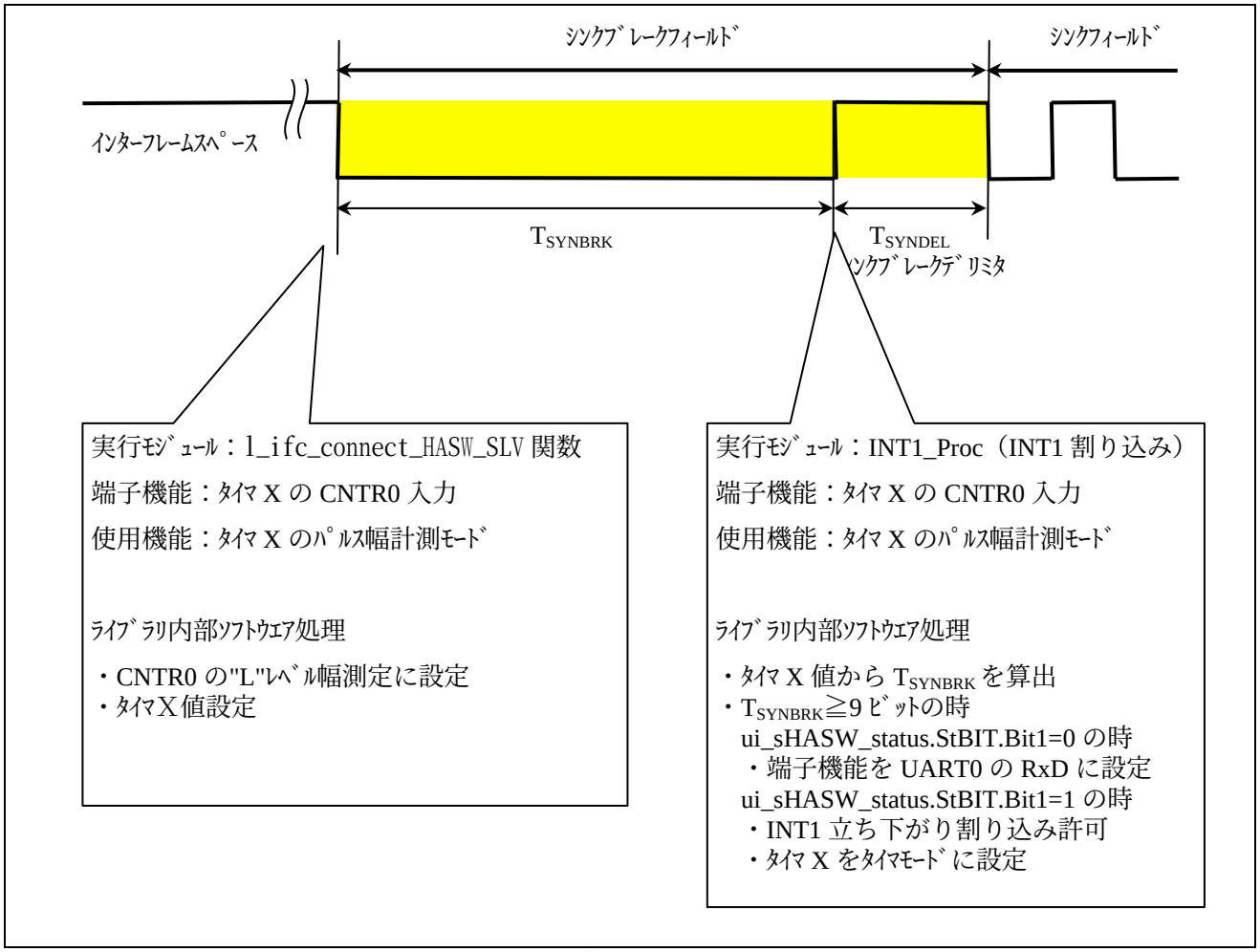


図２９　シンクブレークフィールドの検出

3.5.2.2 シンクフィールドの受信

ui_sHASW_status.StBIT.Bit1 ビットの設定により，シンクフィールドの受信制御方法は以下のようになります。

ui_sHASW_status.StBIT.Bit1=0：UART0 受信機能によりシンクフィールド受信データ(55h)の判定を行います。

ui_sHASW_status.StBIT.Bit1=1：タイマ X のパルス周期計測モード機能により，シンクフィールドのビット幅を計測し，通信転送レートを補正（UART0 モジュール内の U0BRG などの再設定）します。

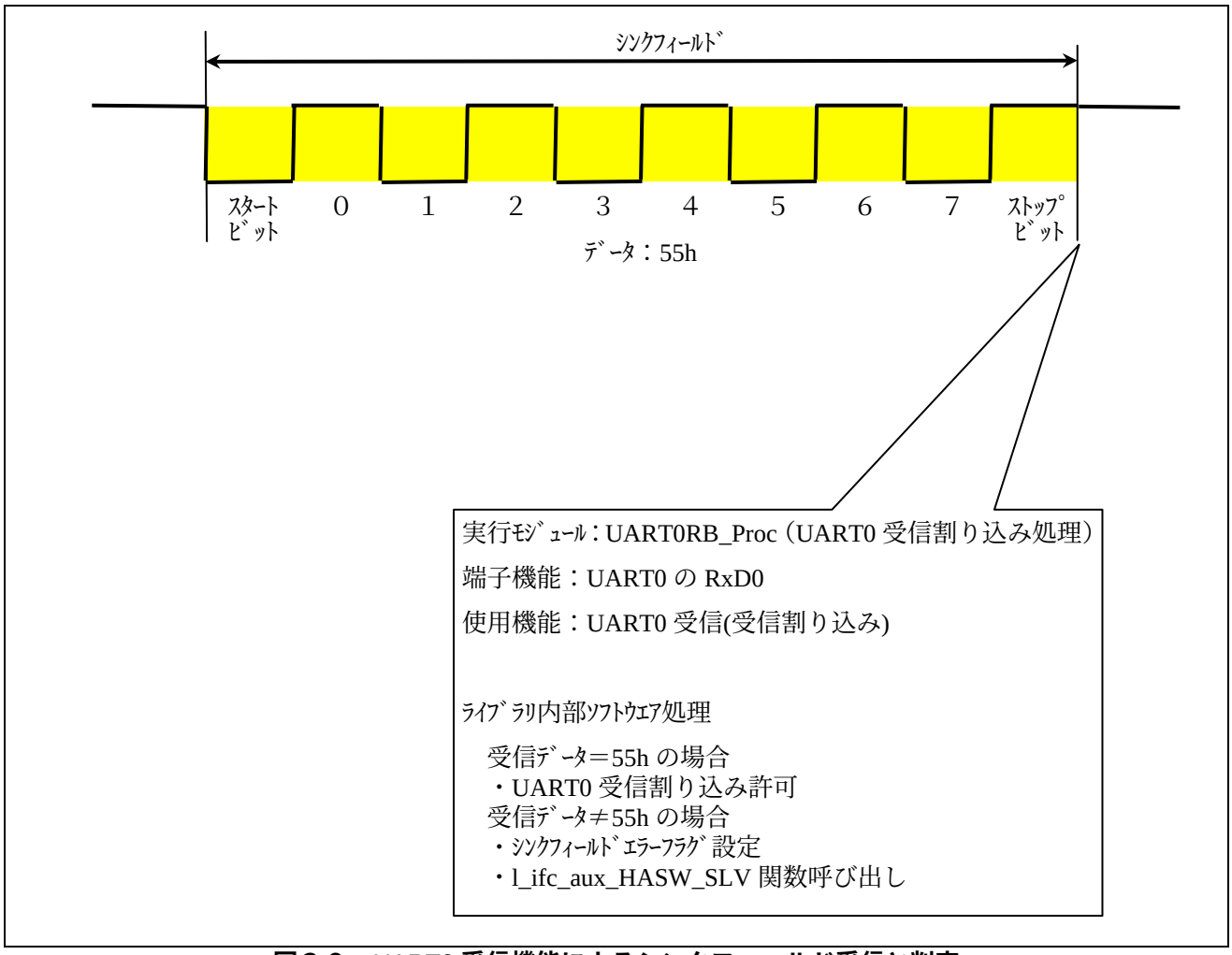


図３０　UART0 受信機能によるシンクフィールド受信と判定

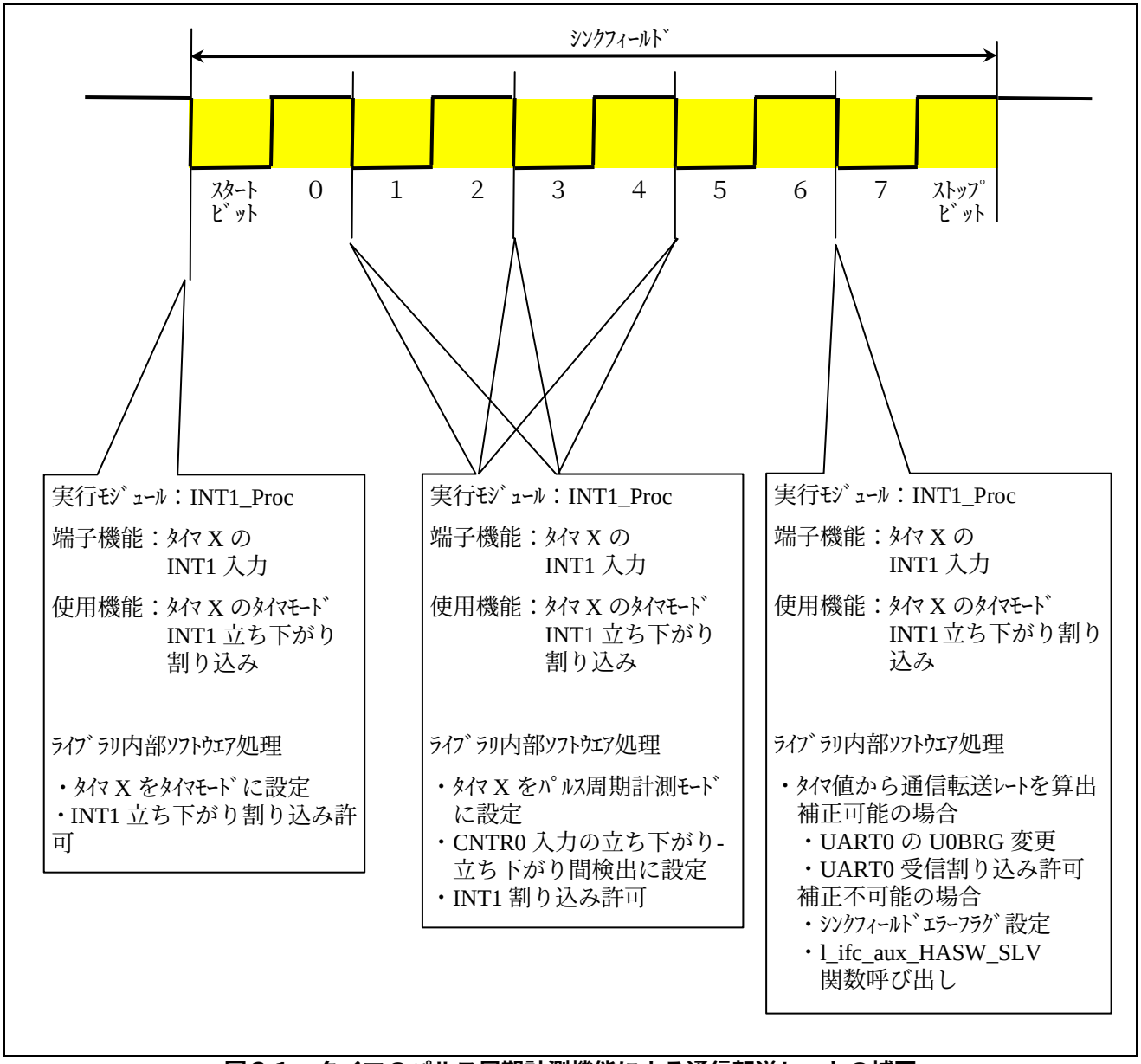


図３１　タイマのパルス周期計測機能による通信転送レートの補正

3.5.2.3 ID フィールドの受信

UART0 受信機能により ID フィールドデータを受信し, ID フィールドデータ中 ID（DLC, パリティを含む）を判定します。自ノードへのレスポンス送信要求 ID であった場合，uc_dHASW_rx_id にデータを格納し，レスポンスフレームの送信を開始します。

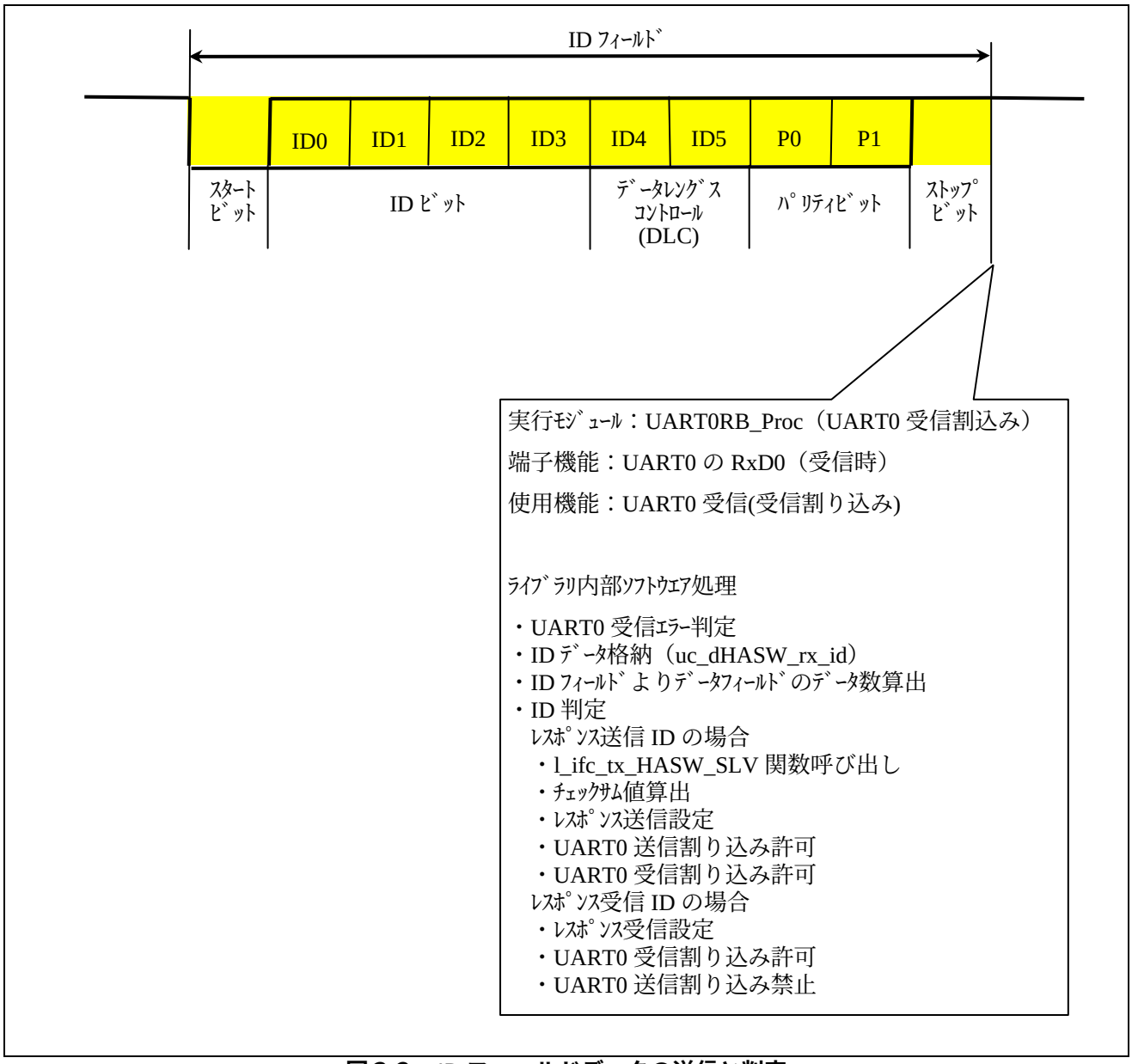


図３２　ID フィールドデータの送信と判定

3.5.3 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，UART0 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，UART0 受信機能によりレスポンスフレームを受信します。

3.5.3.1 データフィールドの送信

UART0 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，uc_bHASW_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。

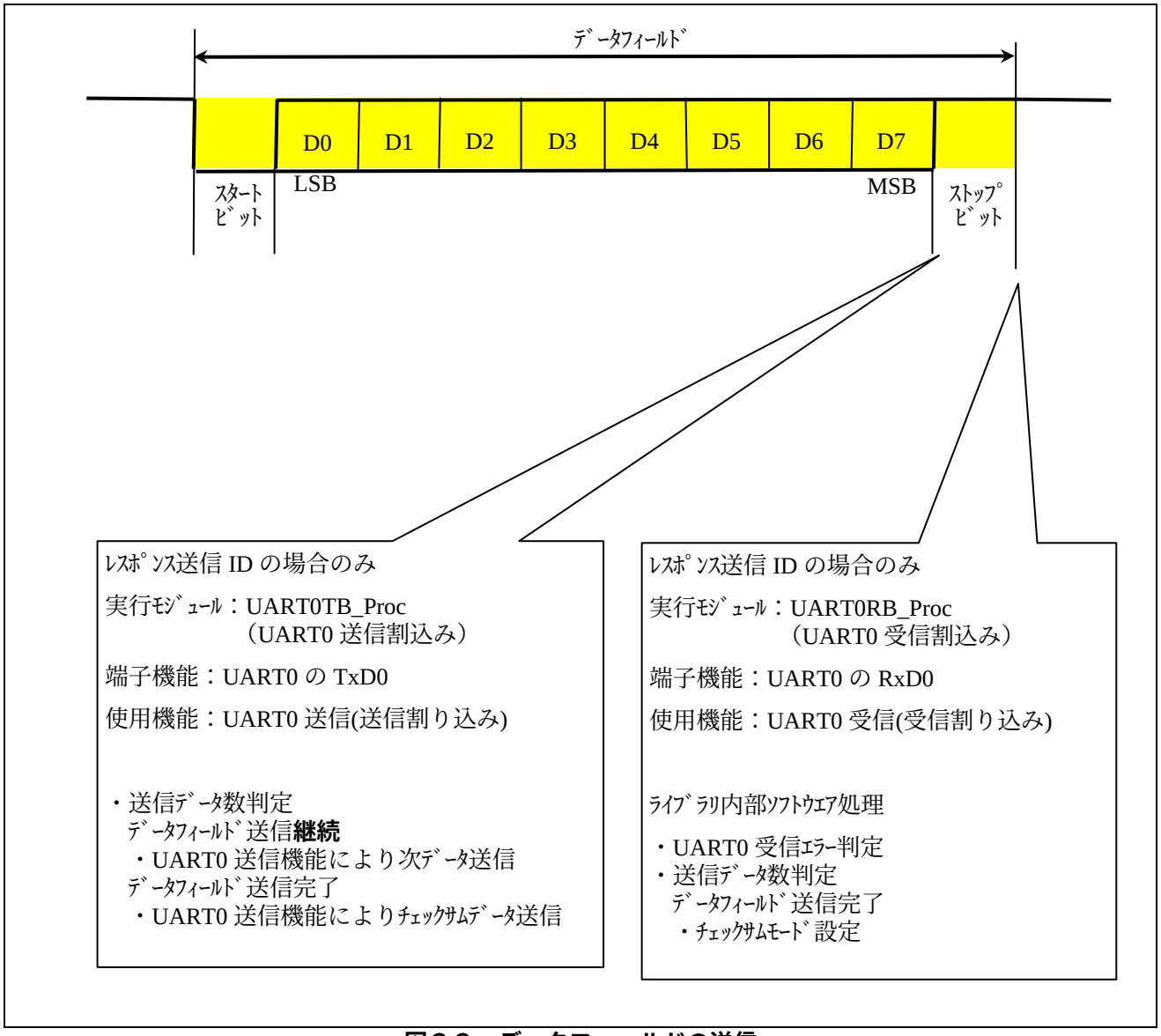


図３３　データフィールドの送信

3.5.3.2 チェックサムフィールドの送信

UART0 送信機能によりチェックサムフィールドを送信します。

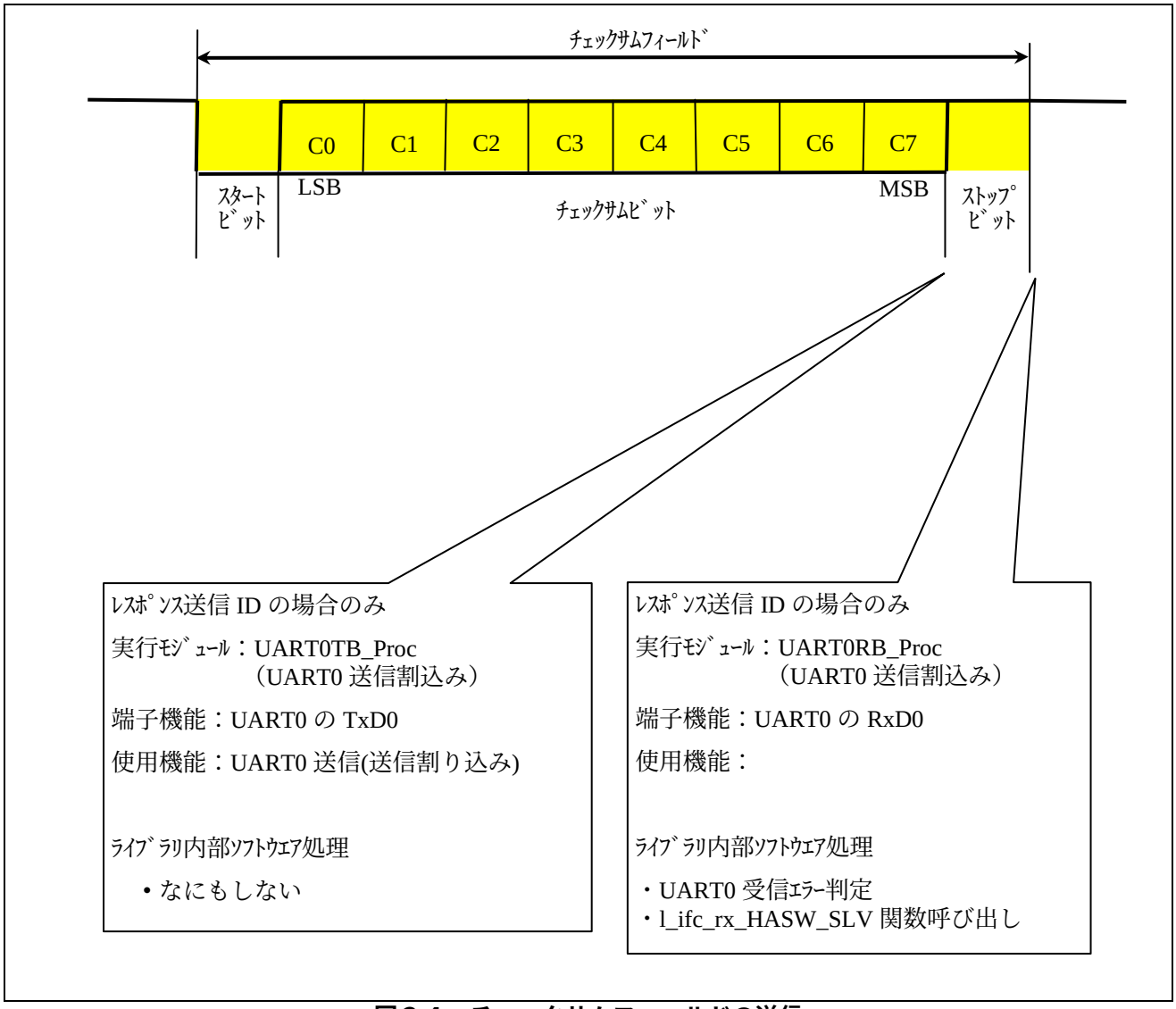


図３４　チェックサムフィールドの送信

3.5.3.3 データフィールドの受信

UART0 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，uc_bHASW_rx_data[0]から順に受信したデータ数のみ uc_bHASW_rx_data[0]～[7]に格納します。

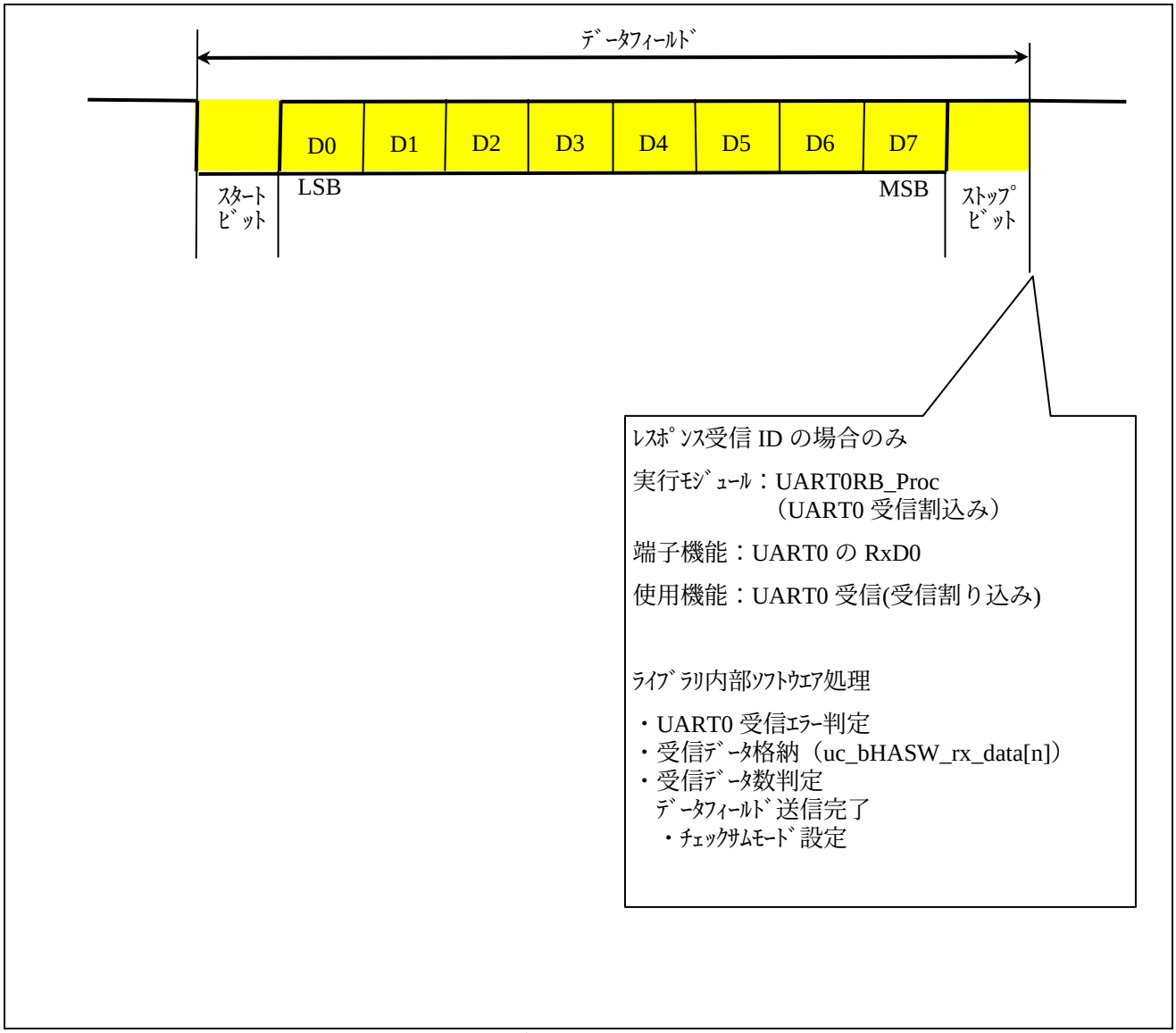


図３５　データフィールドの受信

3.5.3.4 チェックサムフィールドの受信

UART0 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。

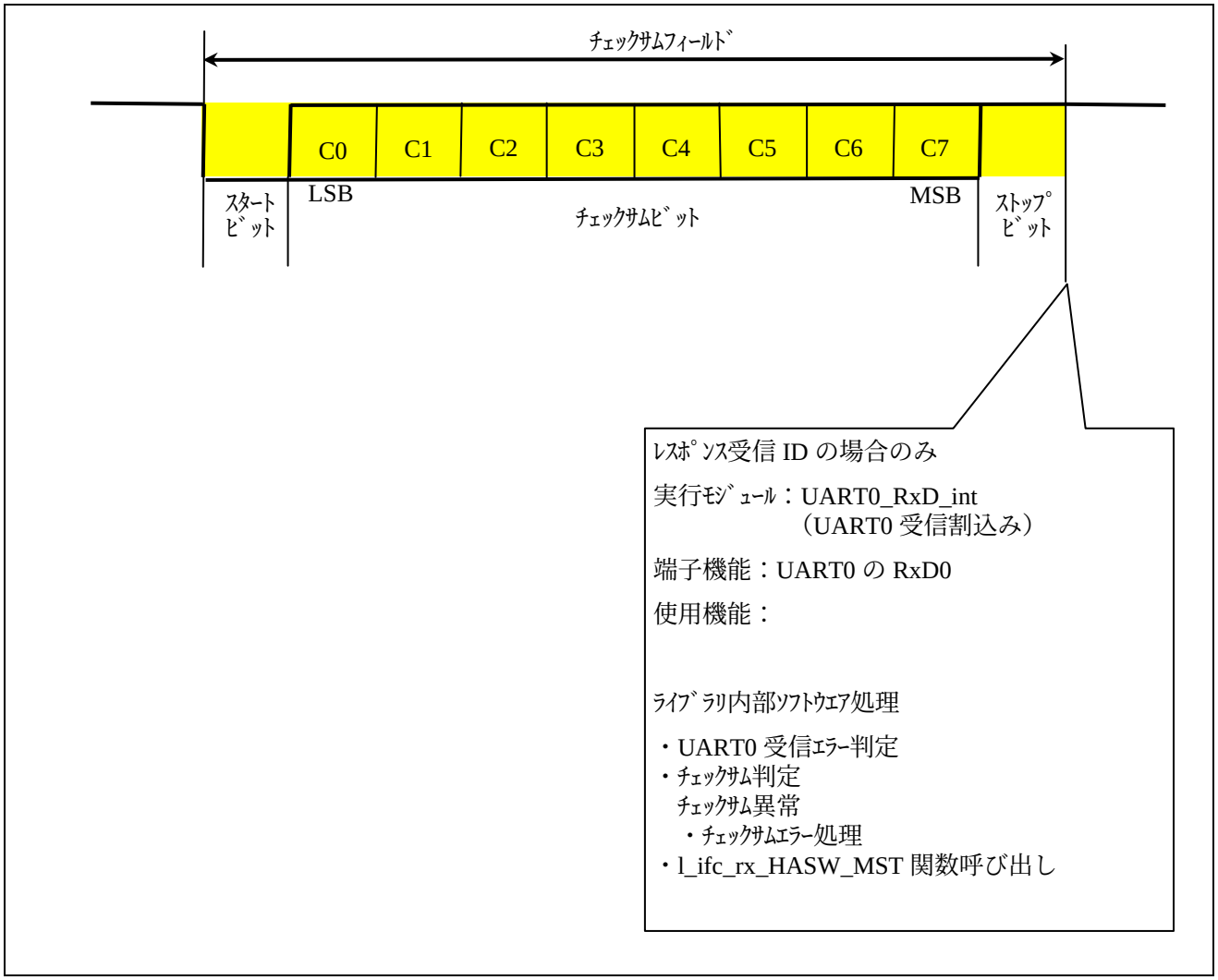


図３６　チェックサムフィールドの受信

3.6 ソフトウエア説明

以下に，本ライブラリソフトウエアの説明を示します。

3.6.1 ヘッダファイルの組み込み

R8C/18 グループ内蔵周辺レジスタ定義ファイル（SFR_R819.h），一線式通信ライブラリ用定義ファイル（HASW_SLV.h）の組み込みを行います。

```
#include    "hasw_slv.h"
#ifdef __CPU_R8C_18
    #include    "sfr_r819.h"
#endif
```

3.6.2 関数定義

一線式通信ライブラリ用定義ファイル（HASW_SLV.h）内にて，l_sys_init_HASW_SLV 関数，l_ifc_init_HASW_SLV 関数，l_ifc_connect_HASW_SLV 関数，l_ifc_disconnect_HASW_SLV 関数，l_ifc_rx_HASW_SLV 関数，l_ifc_aux_HASW_SLV 関数，l_ifc_tx_HASW_SLV 関数の定義を行っています。

なお，INT1_Proc 関数(INT1 割り込み関数)，Uart0RB_Proc 関数(UART0 受信割り込み関数)，Uart0TB_Proc 関数(UART0 送信割り込み関数)についてはそれぞれの割り込み要因で呼び出されますので，ユーザアプリケーションプログラム内のそれぞれの割り込みベクタテーブルへ組み込みを行ってください。

```
void Int1_Proc(void);                           // ベクタ番地+100～+103 に設定
void Uart0RB_Proc(void);                        // ベクタ番地+72～+75 に設定
void Uart0TB_Proc(void);                        // ベクタ番地+68～+71 に設定
void ChangeMeasureMode(unsigned char );
unsigned char CheckIdentParity(unsigned char );
unsigned char CheckResponseCommand(void);
void MakeErrorStatus(unsigned char );
unsigned char MakeHASW_CheckSUM(unsigned char );
```

3.6.3 ライブラリ内部定数・変数定義

ライブラリ内部で使用する定数および変数を定義します。

表１２　ライブラリ内部定数定義

ラベル名（変数名）	データ型	機能説明
ErrorFlagTable[6]	unsigned char (配列)	ステータスフラグ設定データ
SlaveCommandTable []	unsigned char (配列)	レスポンス送信 ID データ
HASW_STATUS	enum	通信状態管理データ
HASW_BUFFER	enum	通信バッファ管理データ
__TBIT_PULS_BASE	unsigned int	1 秒間カウント値(φosc 換算)
__TBIT_PULS_WIDTH	unsigned int	1 ビット期間カウント値(φosc 換算)
__TBIT_PULS_CRCT_MAX	unsigned int	1 ビット期間カウント値(ﾀｲﾏ X 値)補正最大値[+5%]
__TBIT_PULS_MAX	unsigned int	1 ビット期間カウント値(ﾀｲﾏ X 値)有効最大値[+1.5%]
__TBIT_PULS_TYP	unsigned int	1 ビット期間カウント値(ﾀｲﾏ X 値)有効基準値
__TBIT_PULS_MIN	unsigned int	1 ビット期間カウント値(ﾀｲﾏ X 値)有効最小値[-1.5%]
__TBIT_PULS_CRCT_MIN	unsigned int	1 ビット期間カウント値(ﾀｲﾏ X 値)補正最小値[-5%]
__SYNC_BRK_WIDTH_MIN	unsigned int	9 ビット期間カウント値(φosc 換算)
ERROR_MODE	enum	エラー管理データ

表１３　ライブラリ内部変数定義

ラベル名（変数名）	データ型	機能説明
uc_cHaswStatus	unsigned char	通信インタフェース状態管理 （enum HASW_STATUS）
uc_cTransCnt	unsigned char	通信データフィールド送受信カウンタ
uc_cTransPtr	unsigned char	通信データフィールドバッファポインタ
uc_cPulsCnt	unsigned char	パルス周期回数バッファ(uc_cTransCnt 兼用)
un_sUART0	（構造体）	UART0 送受信テンポラリバッファ
un_sUART0.uiAll	ワードアクセス unsigned int	
	ビットアクセス	
un_sUART0.StBIT.Bit15	ビット-15	エラーサムフラグ
un_sUART0.StBIT.Bit14	ビット-14	パリティエラーフラグ
un_sUART0.StBIT.Bit13	ビット-13	フレーミングエラーフラグ
un_sUART0.StBIT.Bit12	ビット-12	オーバーランエラーフラグ
un_sUART0.StBIT.Bit11	ビット-11	リザーブビット
un_sUART0.StBIT.Bit10	ビット-10	リザーブビット
un_sUART0.StBIT.Bit9	ビット-9	リザーブビット
un_sUART0.StBIT.Bit8	ビット-8	送受信データ
un_sUART0.StBIT.Bit7	ビット-7	
un_sUART0.StBIT.Bit6	ビット-6	
un_sUART0.StBIT.Bit5	ビット-5	
un_sUART0.StBIT.Bit4	ビット-4	
un_sUART0.StBIT.Bit3	ビット-3	
un_sUART0.StBIT.Bit2	ビット-2	
un_sUART0.StBIT.Bit1	ビット-1	
un_sUART0.StBIT.Bit0	ビット-0	
un_sU0BRGdata	（構造体）	UART0 BRG 値データ格納バッファ
un_sU0BRGdata.uiAll	ワードアクセス unsigned int	
	バイトアクセス	
un_sU0BRGdata.StBYTE. Byte0	下位 8 ビット	U0BRG 設定データ
un_sU0BRGdata.StBYTE. Byte1	上位 8 ビット	U0BRG 設定データ
uc_bHaswBuff [0]〜[9]	unsigned char (配列)	通信バッファ （enum HASW_BUFFER）

```
enum HASW_STATUS
        enHASW_INIT,                                    // 初期状態モード
        enSYNCH_BREAK,                                  // SynchBreak モード
        enSYNCH_MEASURE,                                // Synch 計測モード
        enSYNCH_SYNCH,                                  // Synch 同期モード
        enIDENT,                                        // Ident 受信モード
        enRESPONSE,                                     // Response 送信モード
        enMASTERCMD,                                    // Command 受信モード
        enCHECKSUM,                                      // Checksum 送信モード
        enHASW_STATUS
};
GLOBAL    unsigned char    uc_cHaswStatus;              // 通信インタフェース状態管理
#ifdef    SBDATA_ON
  #pragma  SBDATA        uc_cHaswStatus
```

```
#endif
GLOBAL    unsigned char    uc_cTransCnt;          // 通信データフィールド送受信カウンタ[0,2,4,8]
#ifdef    SBDATA_ON
    #pragma  SBDATA  uc_cTransCnt
#endif
GLOBAL    unsigned char    uc_cTransPtr;              // 通信データフィールドバッファポインタ
#ifdef    SBDATA_ON
    #pragma  SBDATA  uc_cTransPtr                  // [enHASW_DATA0～enHASW_DATA7]
#endif

typedef union {
    struct{
        char    Bit0:1;
        char    Bit1:1;
        char    Bit2:1;
        char    Bit3:1;
        char    Bit4:1;
        char    Bit5:1;
        char    Bit6:1;
        char    Bit7:1;
        char    Bit8:1;
        char    Bit9:1;
        char    Bit10:1;
        char    Bit11:1;
        char    Bit12:1;
        char    Bit13:1;
        char    Bit14:1;
        char    Bit15:1;
    }StBIT;
    struct {
        unsigned char    Byte0;
        unsigned char    Byte1;
    }StBYTE;
    unsigned int uiAll;
}UnWORD;
GLOBAL    UnWORD    un_sUART0;                          // UART0 送受信テンポラリバッファ
#ifdef    SBDATA_ON
    #pragma  SBDATA  un_sUART0
#endif
GLOBAL    UnWORD    un_sU0BRGdata;                      // UART0 BRG データ格納バッファ
#ifdef    SBDATA_ON
    #pragma  SBDATA  un_sU0BRGdata
#endif

enum HASW_BUFFER {
        enHASW_ID,
        enHASW_DATA0,
        enHASW_DATA1,
        enHASW_DATA2,
        enHASW_DATA3,
        enHASW_DATA4,
        enHASW_DATA5,
        enHASW_DATA6,
        enHASW_DATA7,
        enHASW_CHECKSUM,
        enHASW_BUFFER
};
```

```
GLOBAL    unsigned char    uc_bHaswBuff[enHASW_BUFFER];    // 通信バッファ
#ifdef    SBDATA_ON
    #pragma  SBDATA  uc_bHaswBuff
#endif

/*=========================================================================================*/
/*        定数定義                                                                         */
/*-----------------------------------------------------------------------------------------*/
#ifdef __OVER_10M_Hz    // OSC が 10MHz 以上の場合，TimerX のソースは 2/f
    #define    __TBIT_PULS_BASE        (__OSC_Hz / 2)        // 1sec / 2/f(TimerX Count Source)
#else                   // OSC が 10MHz 以下の場合，TimerX のソースは 1/f
    #define    __TBIT_PULS_BASE        __OSC_Hz              // 1sec / 1/f(TimerX Count Source)
#endif
#define        __TBIT_PULS_WIDTH        (__TBIT_PULS_BASE / __B_rate)    // 1 ビットパルス幅
#define        __TBIT_PULS_CRCT_MAX    (unsigned int)(__TBIT_PULS_WIDTH * 1.050) // 補正 max 値
#define        __TBIT_PULS_MAX           (unsigned int)(__TBIT_PULS_WIDTH * 1.015) // max 値
#define        __TBIT_PULS_TYP           (unsigned int)(__TBIT_PULS_WIDTH * 1.000) // typ 値
#define        __TBIT_PULS_MIN           (unsigned int)(__TBIT_PULS_WIDTH * 0.985) // min 値
#define        __TBIT_PULS_CRCT_MIN    (unsigned int)(__TBIT_PULS_WIDTH * 0.950) // 補正 min 値
#define        __SYNC_BRK_WIDTH_MIN    (__TBIT_PULS_WIDTH * 9)     // SYCBRK 信号パルス幅 min 値
/*=========================================================================================*/
/*        コマンドテーブル定義                                                             */
/*-----------------------------------------------------------------------------------------*/
const unsigned char SlaveCommandTable[]={
#ifdef    __ID00
    __ID00,
#endif
#ifdef    __ID01
    __ID01,
#endif
#ifdef    __ID02
    __ID02,
#endif
#ifdef    __ID03
    __ID03,
#endif
#ifdef    __ID04
    __ID04,
#endif
#ifdef    __ID05
    __ID05,
#endif
#ifdef    __ID06
    __ID06,
#endif
#ifdef    __ID07
    __ID07,
#endif
#ifdef    __ID08
    __ID08,
#endif
#ifdef    __ID09
    __ID09,
#endif
#ifdef    __ID10
    __ID10,
#endif
```

```
#ifdef    __ID11
    __ID11,
#endif
#ifdef    __ID12
    __ID12,
#endif
#ifdef    __ID13
    __ID13,
#endif
#ifdef    __ID14
    __ID14,
#endif
#ifdef    __ID15
    __ID15,
#endif
#ifdef    __ID16
    __ID16,
#endif
#ifdef    __ID17
    __ID17,
#endif
#ifdef    __ID18
    __ID18,
#endif
#ifdef    __ID19
    __ID19,
#endif
#ifdef    __ID20
    __ID20,
#endif
#ifdef    __ID21
    __ID21,
#endif
#ifdef    __ID22
    __ID22,
#endif
#ifdef    __ID23
    __ID23,
#endif
#ifdef    __ID24
    __ID24,
#endif
#ifdef    __ID25
    __ID25,
#endif
#ifdef    __ID26
    __ID26,
#endif
#ifdef    __ID27
    __ID27,
#endif
#ifdef    __ID28
    __ID28,
#endif
#ifdef    __ID29
    __ID29,
#endif
```

```
#ifdef    __ID30
    __ID30,
#endif
#ifdef    __ID31
    __ID31,
#endif
#ifdef    __ID32
    __ID32,
#endif
#ifdef    __ID33
    __ID33,
#endif
#ifdef    __ID34
    __ID34,
#endif
#ifdef    __ID35
    __ID35,
#endif
#ifdef    __ID36
    __ID36,
#endif
#ifdef    __ID37
    __ID37,
#endif
#ifdef    __ID38
    __ID38,
#endif
#ifdef    __ID39
    __ID39,
#endif
#ifdef    __ID40
    __ID40,
#endif
#ifdef    __ID41
    __ID41,
#endif
#ifdef    __ID42
    __ID42,
#endif
#ifdef    __ID43
    __ID43,
#endif
#ifdef    __ID44
    __ID44,
#endif
#ifdef    __ID45
    __ID45,
#endif
#ifdef    __ID46
    __ID46,
#endif
#ifdef    __ID47
    __ID47,
#endif
#ifdef    __ID48
    __ID48,
#endif
```

```
#ifdef    __ID49
    __ID49,
#endif
#ifdef    __ID50
    __ID50,
#endif
#ifdef    __ID51
    __ID51,
#endif
#ifdef    __ID52
    __ID52,
#endif
#ifdef    __ID53
    __ID53,
#endif
#ifdef    __ID54
    __ID54,
#endif
#ifdef    __ID55
    __ID55,
#endif
#ifdef    __ID56
    __ID56,
#endif
#ifdef    __ID57
    __ID57,
#endif
#ifdef    __ID58
    __ID58,
#endif
#ifdef    __ID59
    __ID59,
#endif
#ifdef    __ID61
    __ID61
#endif
};

/*=========================================================================================*/
/*        define 定義                                                                      */
/*-----------------------------------------------------------------------------------------*/
#define     pTxD            p1_4                // I : Hi-Z : TxD 出力
#define     pdTxD            pd1_4
#define     pRxD            p1_5                // I : Hi_Z : RxD 入力
#define     pdRxD            pd1_5
#define        uc_sUART0LowBuffer        un_sUART0.StBYTE.Byte0
#define        uc_sUART0HighBuffer        un_sUART0.StBYTE.Byte1
#define        ui_sUART0Buffer            un_sUART0.uiAll
#define        uc_cPulsCnt                uc_cTransCnt    // パルス周期計測回数バッファ（兼用）

#define        uc_fBaudrateCheck    ui_sHASW_status.StBIT.Bit1      // 転送レート補正フラグ
enum ERROR_MODE {
        enERROR_CHECKSUM,                           // チェックサムエラー
        enERROR_SYNCH,                              // シンクフィールドエラー
        enERROR_PARITY,                             // ID パリティエラー
        enERROR_OVER_RUN,                           // オーバーランエラー
        enERROR_FRAME,                              // フレームエラー
```

```
        enERROR_BIT,                        // ビットエラー
        enERROR_MODE
  };
#define       FALSE    0
#define       TRUE    1
#define       SET       1
#define       P_HIGH    1
#define       P_IN    0
```

3.6.4 グローバル変数定義

一線式通信ライブラリ用定義ファイル（HASW_SLV.h）内にて，ユーザー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（uc_bHASW_tx_data[8], uc_dHASW_rx_id, uc_bHASW_rx_data[8], ui_sHASW_status）を定義しています。

3.6.5 初期設定用関数

一線式通信制御に使用する R8C/18 グループ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エアフラグなどの初期化を行います。

注意：端子 P1_4(TxD0)，P1_5(RxD0)は一線式通信で使用します。ユーザアプリケーションでポート 1 におけるその他の端子(P1_0～P1_3，P1_6～P1_7)を使用する場合，端子 P1_4(TxD0)，P1_5(RxD0)の方向レジスタを必ず入力方向に設定するようにユーザアプリケーションで設定してください。

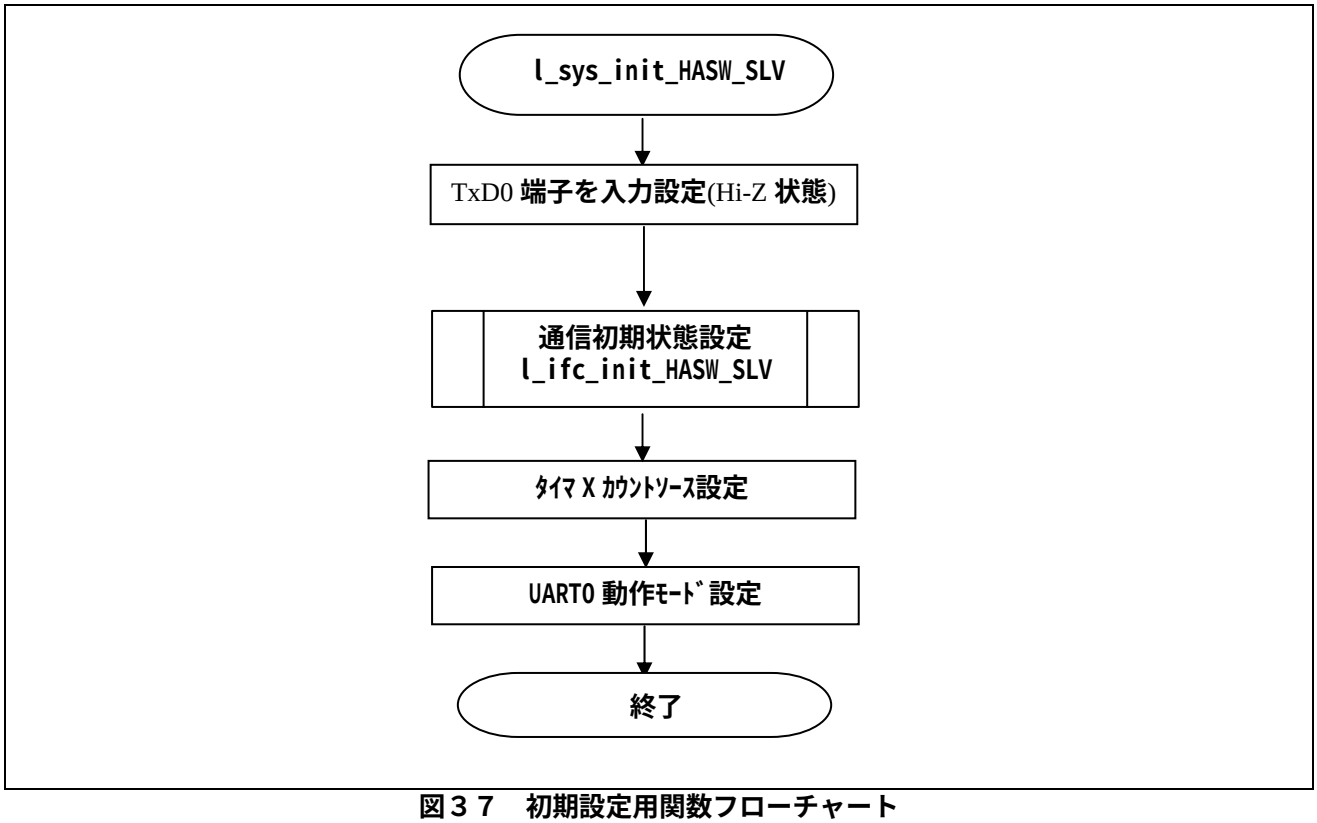


図３７　初期設定用関数フローチャート

```
void l_sys_init_HASW_SLV(void)
{
    pTxD = P_HIGH;                              /* TxD port Level "L"設定 */
    l_ifc_init_HASW_SLV();                      // 初期状態設定
    #ifdef __OVER_10M_Hz    // OSC が 10MHz 以上の場合，TimerX のソースは 2/f
    tcss = (tcss & 0xf0) | 0x03;            /* TimerX=f2 クロックソース設定 */
    #else                     // OSC が 10MHz 以下の場合，TimerX のソースは 1/f
    tcss = (tcss & 0xf0) | 0x00;            /* TimerX=f1 クロックソース設定 */
    #endif
    ucon = 0x85;     /* P1_5/RxD0/CNTR01/INT11,連続受信モード,送信完了時割り込み設定 */
}
```

3.6.6 一線式通信制御初期設定関数

一線式通信制御に使用する R8C/18 グループ内蔵周辺機能の初期状態設定, およびライブラリ内部で使用するソフトウエアフラグなどの初期化を行います。通信中にエラーが発生した場合，本関数を呼び出すことにより，通信制御状態を初期状態に戻します。

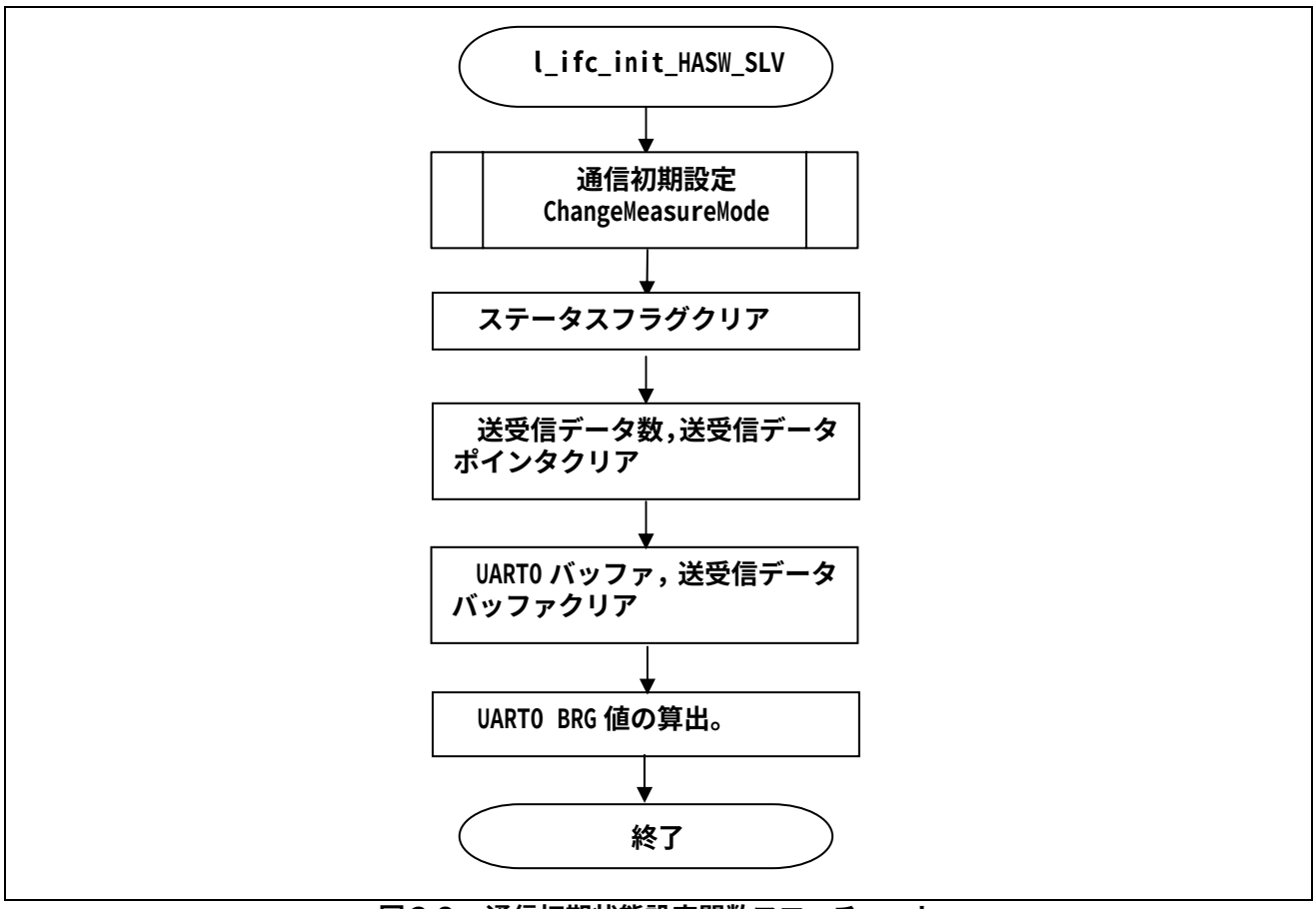


図３８　通信初期状態設定関数フローチャート

```
void l_ifc_init_HASW_SLV(void)
{
    unsigned char    c;

    ChangeMeasureMode(enHASW_INIT);                 // 初期状態設定
    ui_sHASW_status.StBYTE.Byte1 = 0x00;            // ステータスフラグクリア
    uc_cPulsCnt = 0;                                // パルス周期計測カウンタクリア
    uc_cTransCnt = 0;                               // 送受信データ数クリア
    uc_cTransPtr = 0;                               // 送受信データポインタクリア
    ui_sUART0Buffer = 0;                            // UART0 バッファクリア
    for (c=0;c<enHASW_BUFFER;++c) {                 // 送受信データバッファクリア
        uc_bHaswBuff[c] = 0;
    }
#ifdef __OVER_10M_Hz    // OSC が 10MHz 以上の場合，8 で割ると BRG 値が算出される
    un_sU0BRGdata.uiAll  = (__TBIT_PULS_TYP + (__TBIT_PULS_TYP & 0x0004)) >> 3;     // 四捨五入
#else                    // OSC が 10MHz 以下の場合，16 で割ると BRG 値が算出される
    un_sU0BRGdata.uiAll  = (__TBIT_PULS_TYP + (__TBIT_PULS_TYP & 0x0008)) >> 4;     // 四捨五入
#endif
```

3.6.7 一線式通信制御開始準備用関数

シンクブレークフィールドの受信待ち状態に設定します。

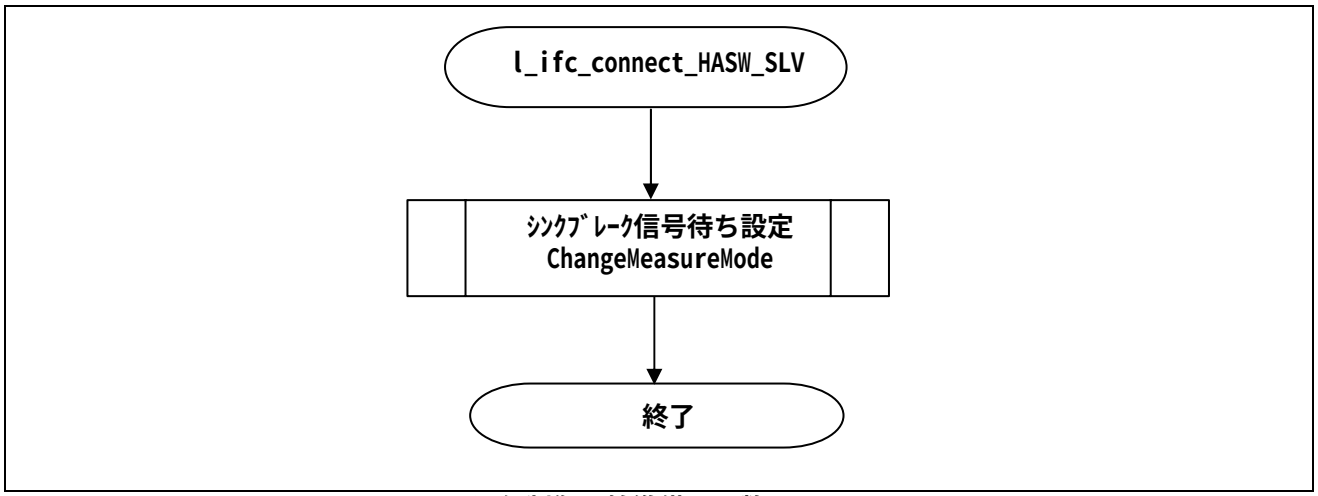


図３９　通信制御開始準備用関数フローチャート

```
void l_ifc_connect_HASW_SLV(void)
{
    ChangeMeasureMode(enSYNCH_BREAK);                    // SynchBreak 待ち設定
}
```

3.6.8 一線式通信制御停止関数

一線式通信制御を停止し，接続されている一線式通信バス上の通信に関与しなくなります。

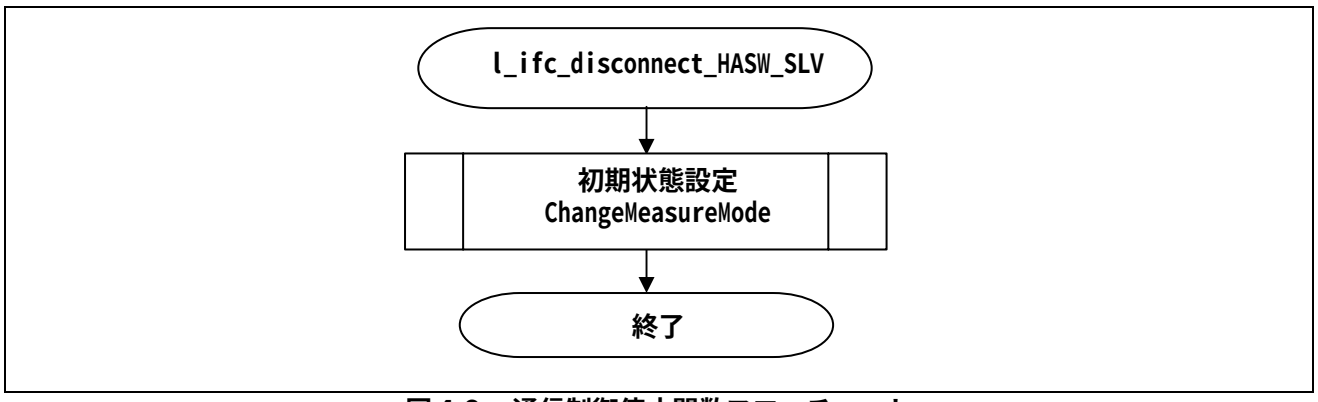


図４０　通信制御停止関数フローチャート

```
void l_ifc_disconnect_HASW_SLV(void)
{
    ChangeMeasureMode(enHASW_INIT);                      // 初期状態設定
}
```

3.6.9 INT1 割り込み関数

シンクブレーク信号の検出およびシンクブレーク信号ドミナント幅計測，自動速度補正有効時のシンクフィールド信号立ち下がりエッジ検出割り込み処理を行います。

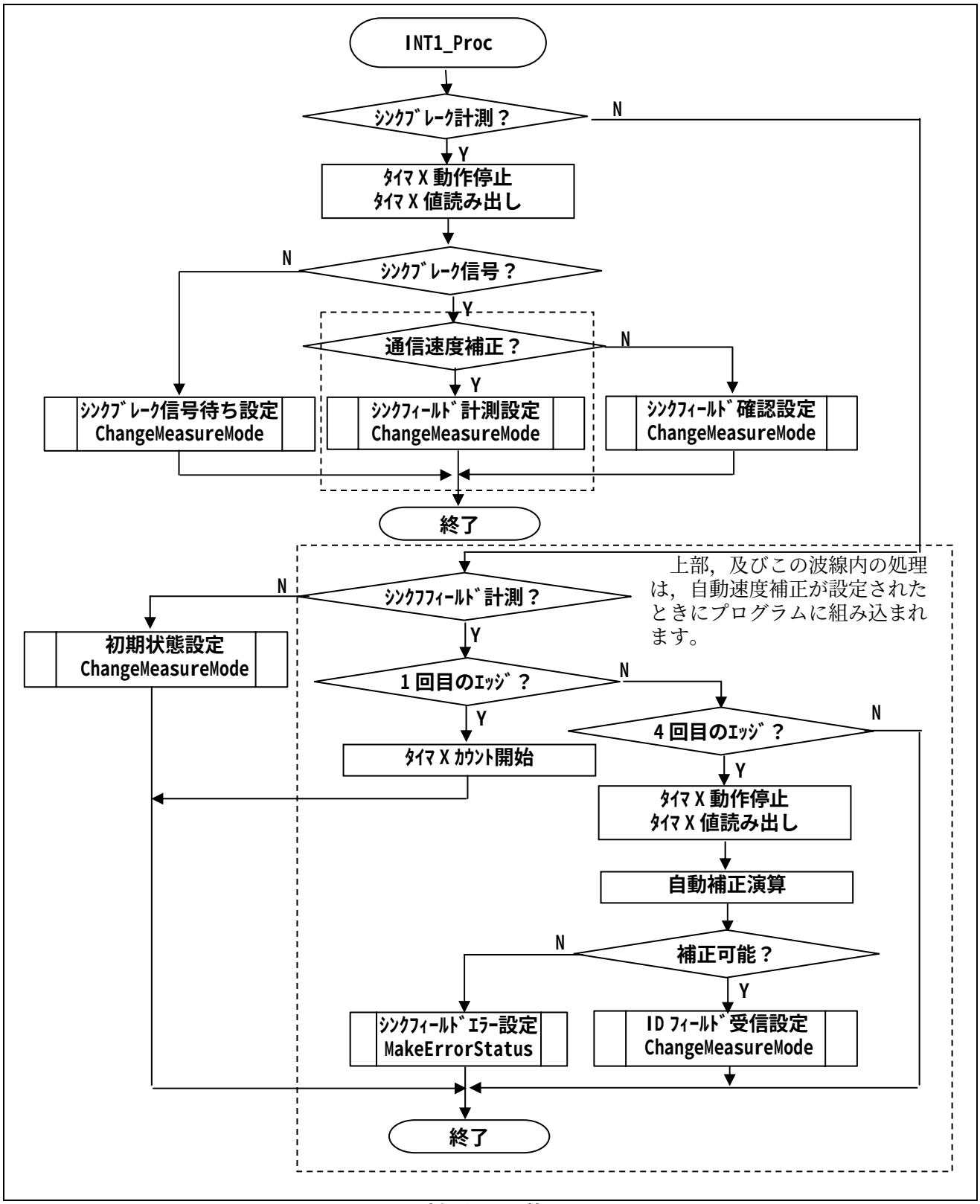


図４１　INT1 割り込み関数フローチャート

```
#pragma INTERRUPT /B Int1_Proc
void Int1_Proc(void)
{
    UnWORD    ui_tmp;

    switch(uc_cHaswStatus){
    case enSYNCH_BREAK:     // SynchBreak 計測
        txmr = 0x03;                                        /* TimerX 動作停止 */
        while(txs==1);                                        /* TimerX 動作停止確認 */
        ui_tmp.StBYTE.Byte0 = ~prex;
        ui_tmp.StBYTE.Byte1 = ~tx;                                    /* TimerX 読みだし */
        /* SynchBreak パルス幅規定判定 */
        if (ir_txic == 0) {                                 // TimerX オーバーフローしていない？
            if (ui_tmp.uiAll < __SYNC_BRK_WIDTH_MIN) {        // SynchBreak 信号が短い？
                ChangeMeasureMode(enSYNCH_BREAK);         // SynchBreak 信号と判定せず
                break;
            }
        }
#ifdef    __BPS_AUTO
        if (uc_fBaudrateCheck == SET) {                     // 転送レート補正あり？
            ChangeMeasureMode(enSYNCH_MEASURE);             // Synch 信号計測状態設定
            uc_cPulsCnt = 0;                               // パルス周期計測カウンタクリア
        }
        else {
            ChangeMeasureMode(enSYNCH_SYNCH);             // Synch 信号同期状態設定
        }
#else
        ChangeMeasureMode(enSYNCH_SYNCH);                 // Synch 信号同期状態設定
#endif
        break;
#ifdef    __BPS_AUTO
    case enSYNCH_MEASURE:     // Synch 信号計測
        if (uc_cPulsCnt == 0) {                             // パルス周期計測 1 回目？
            txmr = 0x0c;                                        /* TimerX 動作開始 */
        }
        if (uc_cPulsCnt == 4) {                             // パルス周期計測 4 回目終了？
            txmr = 0x04;                                        /* TimerX 動作停止 */
            while(txs==1);                                        /* TimerX 動作停止確認 */
            ui_tmp.StBYTE.Byte0 = ~prex;
            ui_tmp.StBYTE.Byte1 = ~tx;                                     /* TimerX 読みだし */

            /* 補正演算 */
            if (ir_txic == 0) {                             // TimerX オーバーフローしていない？
                ui_tmp.uiAll  = ui_tmp.uiAll + (ui_tmp.uiAll & 0x0002);     // 四捨五入
                ui_tmp.uiAll >>= 3;                                 // 8 パルス分なので 8 で割る
                if ((ui_tmp.uiAll >= __TBIT_PULS_CRCT_MIN)
                    &&(ui_tmp.uiAll < __TBIT_PULS_CRCT_MAX)) {     // ±5%?
                    if ((ui_tmp.uiAll < __TBIT_PULS_MIN)
                        ||(ui_tmp.uiAll >= __TBIT_PULS_MAX)) {     // ±1.5%ではない?
                        /* BRG 値補正 */
    #ifdef __OVER_10M_Hz     // OSC が 10MHz 以上の場合，8 で割ると BRG 値が算出される
                        ui_tmp.uiAll  = ui_tmp.uiAll + (ui_tmp.uiAll & 0x0004);     // 四捨五入
                        ui_tmp.uiAll >>= 3;
    #else                       // OSC が 10MHz 以下の場合，16 で割ると BRG 値が算出される
                        ui_tmp.uiAll  = ui_tmp.uiAll + (ui_tmp.uiAll & 0x0008);     // 四捨五入
                        ui_tmp.uiAll >>= 4;
    #endif
```

```
                        un_sU0BRGdata.uiAll = ui_tmp.uiAll;
                    }
                    ChangeMeasureMode(enIDENT);             // Ident 受信モード設定
                    break;
                }
            }
            MakeErrorStatus(enERROR_SYNCH);                 // シンクフィールドエラー設定
        }
        else {
            ++uc_cPulsCnt;
        }
        break;
#endif
    default:
        ChangeMeasureMode(enHASW_INIT);                     // 初期状態設定
        break;
    }
}
```

3.6.10 UART 送信割り込み関数

UART0 送信割り込み処理を行います。

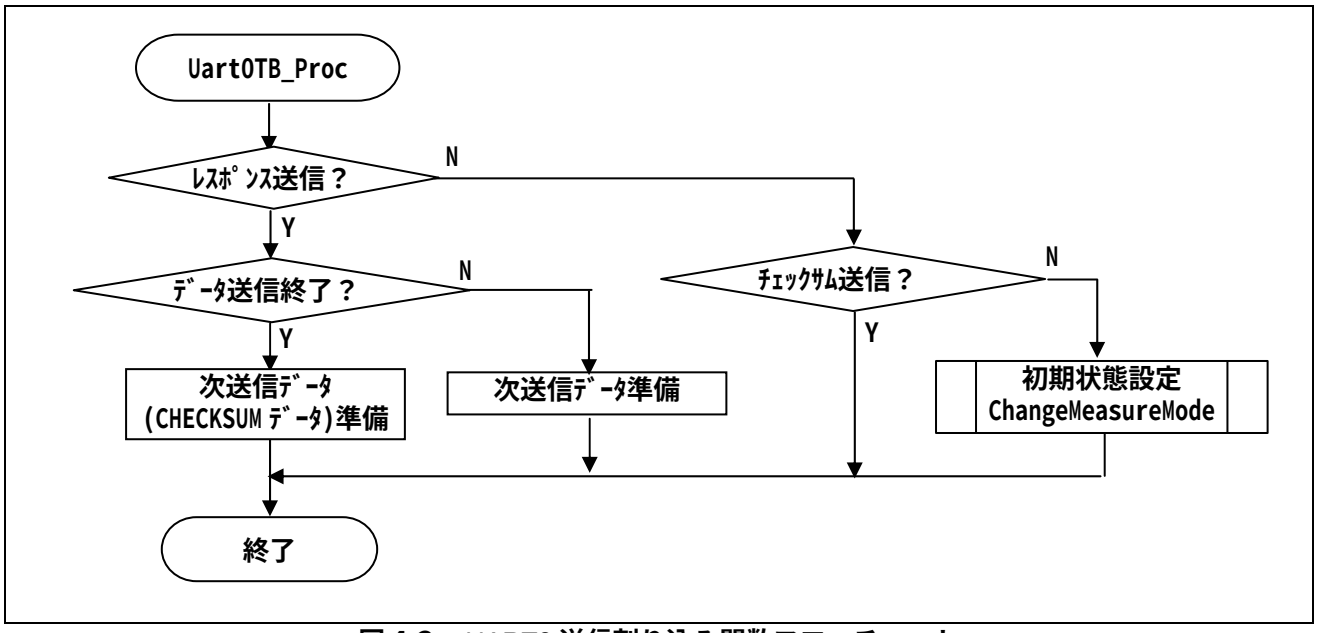


図４２　UART0 送信割り込み関数フローチャート

```
#pragma INTERRUPT /B UartOTB_Proc
void UartOTB_Proc(void)
{
    switch(uc_cHaswStatus){
    case enRESPONSE:    // Response モード
        ui_sUART0Buffer = 0x0000;
        if ((uc_cTransPtr) >= (uc_cTransCnt + enHASW_DATA0)) {
            uc_sUART0LowBuffer = uc_bHaswBuff[enHASW_CHECKSUM];
        }
        else {
            uc_sUART0LowBuffer = uc_bHaswBuff[uc_cTransPtr];
        }
        u0tb = ui_sUART0Buffer;
    case enCHECKSUM:    // Checksum モード
        break;
    default:
        ChangeMeasureMode(enHASW_INIT);                    // 初期状態設定
        break;
    }
}
```

3.6.11 UART 受信割り込み関数

UART0 受信エラー割り込み，UART0 受信割り込み処理を行います。

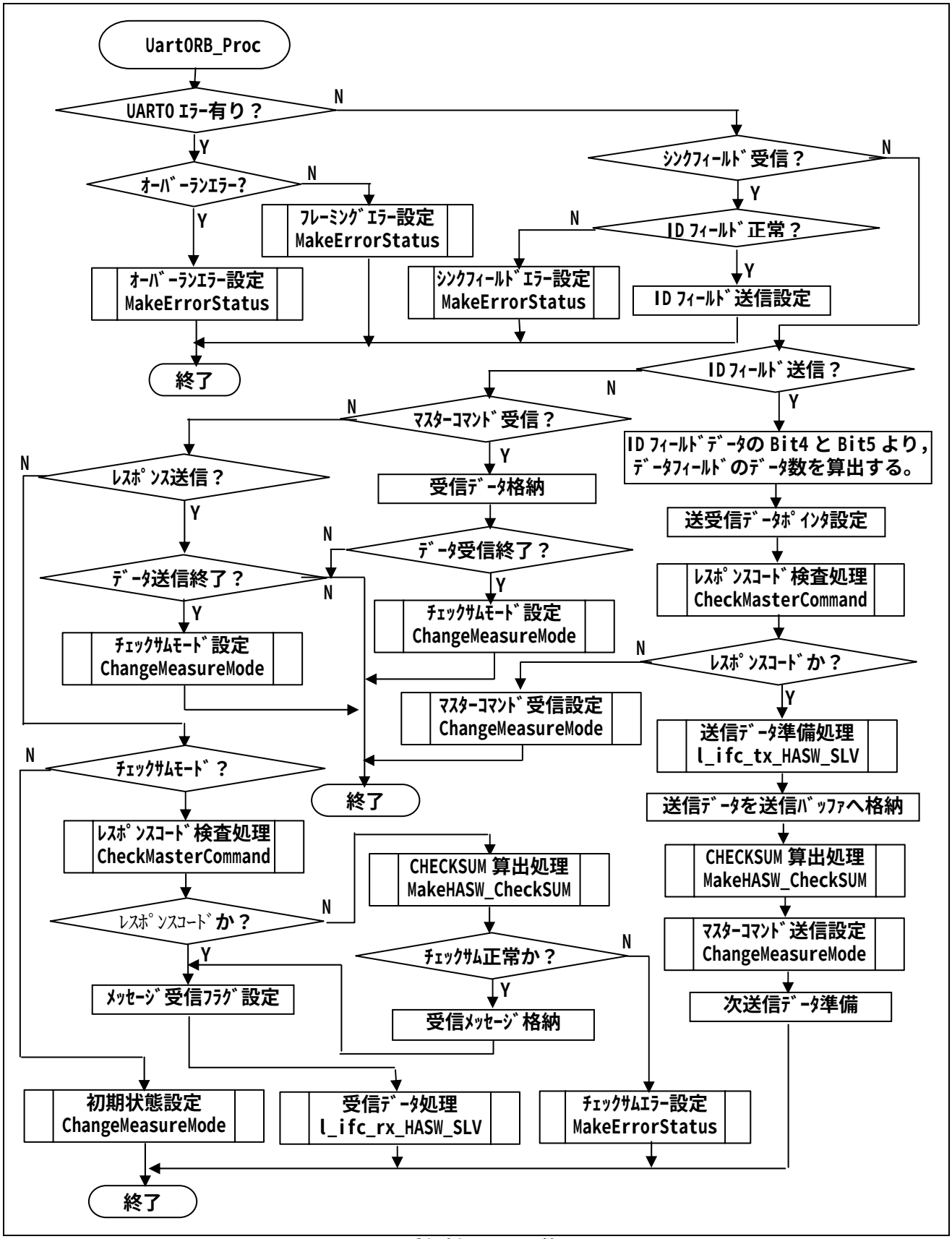


図４３　UART0 受信割り込み関数フローチャート

```
#pragma INTERRUPT /B Uart0RB_Proc
void Uart0RB_Proc(void)
{
    unsigned char    c;

    ui_sUART0Buffer = u0rb;
    if (uc_sUART0HighBuffer&0x80) {
        if (uc_sUART0HighBuffer&0x10) {
            MakeErrorStatus(enERROR_OVER_RUN);              // オーバーランエラー設定
        }
        else {
            MakeErrorStatus(enERROR_FRAME);                 // フレーミングエラー設定
        }
        return;
    }
    switch(uc_cHaswStatus){
    case enSYNCH_SYNCH:     // Synch 同期モード
        if (uc_sUART0LowBuffer != 0x55) {
            MakeErrorStatus(enERROR_SYNCH);                 // シンクフィールドエラー設定
        }
        else {
            uc_cHaswStatus = enIDENT;                       // Ident 受信モード切り替え
        }
        break;
    case enIDENT:         // Ident モード
        uc_cTransCnt = 0;
        if ((uc_sUART0LowBuffer&0xc0) == CheckIdentParity(uc_sUART0LowBuffer)) {
            if (un_sUART0.StBIT.Bit5 == 1) {
                if (un_sUART0.StBIT.Bit4 == 1) uc_cTransCnt = 8;
                else uc_cTransCnt = 4;
            }
            else uc_cTransCnt = 2;
        }
        uc_bHaswBuff[enHASW_ID] = uc_sUART0LowBuffer;
        uc_dHASW_rx_id = uc_bHaswBuff[enHASW_ID] & 0x3f;
        if (uc_cTransCnt == 0) {
            MakeErrorStatus(enERROR_PARITY);                // ID パリティエラー設定
        }
        else {
            uc_cTransPtr = enHASW_DATA0;
            if (CheckResponseCommand() == TRUE) {           // レスポンス応答コマンドか？
                l_ifc_tx_HASW_SLV();                        // l_ifc_tx_HASW_SLV 関数呼び出し
                for (c=0;c<uc_cTransCnt;++c) {
                    uc_bHaswBuff[c+enHASW_DATA0] = uc_bHASW_tx_data[c];
                }
                uc_bHaswBuff[enHASW_CHECKSUM] = MakeHASW_CheckSUM(uc_cTransCnt);
                ChangeMeasureMode(enRESPONSE);              // レスポンス応答設定
                ui_sUART0Buffer = 0x0000;
                uc_sUART0LowBuffer = uc_bHaswBuff[uc_cTransPtr++];
                u0tb = ui_sUART0Buffer;
            }
            else {
                ChangeMeasureMode(enMASTERCMD);             // マスターコマンド受信設定
            }
        }
        break;
    case enRESPONSE:    // Response モード
```

```
        if (uc_cTransPtr++ >= (uc_cTransCnt + enHASW_DATA0)) {    // 前データ送信完了？
            ChangeMeasureMode(enCHECKSUM);                  // 送信後処理設定
        }
        break;
    case enMASTERCMD:    // Command モード
        uc_bHaswBuff[uc_cTransPtr] = uc_sUART0LowBuffer;
        if (++uc_cTransPtr >= (uc_cTransCnt + enHASW_DATA0)) {    // 前データ受信完了？
            ChangeMeasureMode(enCHECKSUM);                  // チェックサム受信設定
        }
        break;
    case enCHECKSUM:    // Checksum モード
        if (CheckResponseCommand() == FALSE) {              // レスポンス応答コマンドではない？
            uc_bHaswBuff[enHASW_CHECKSUM] = uc_sUART0LowBuffer;
            if (uc_bHaswBuff[enHASW_CHECKSUM] == MakeHASW_CheckSUM(uc_cTransCnt)) {
                for(c=0;c<uc_cTransCnt;c++) {               // 受信データ格納
                    uc_bHASW_rx_data[c] = uc_bHaswBuff[c+enHASW_DATA0];
                }
            }
            else {
                MakeErrorStatus(enERROR_CHECKSUM);          // チェックサムエラー設定
                break;
            }
        }
        ui_sHASW_status.StBYTE.Byte1 = 0x01;                // メッセージ正常送受信フラグ設定
        l_ifc_rx_HASW_SLV();                                // l_ifc_rx_HASW_SLV 関数呼び出し
        break;
    default:
        ChangeMeasureMode(enHASW_INIT);                     // 初期状態設定
        break;
    }
}
```

3.6.12 通信状態制御関数

要求された一線式通信の状態の設定処理を行います。

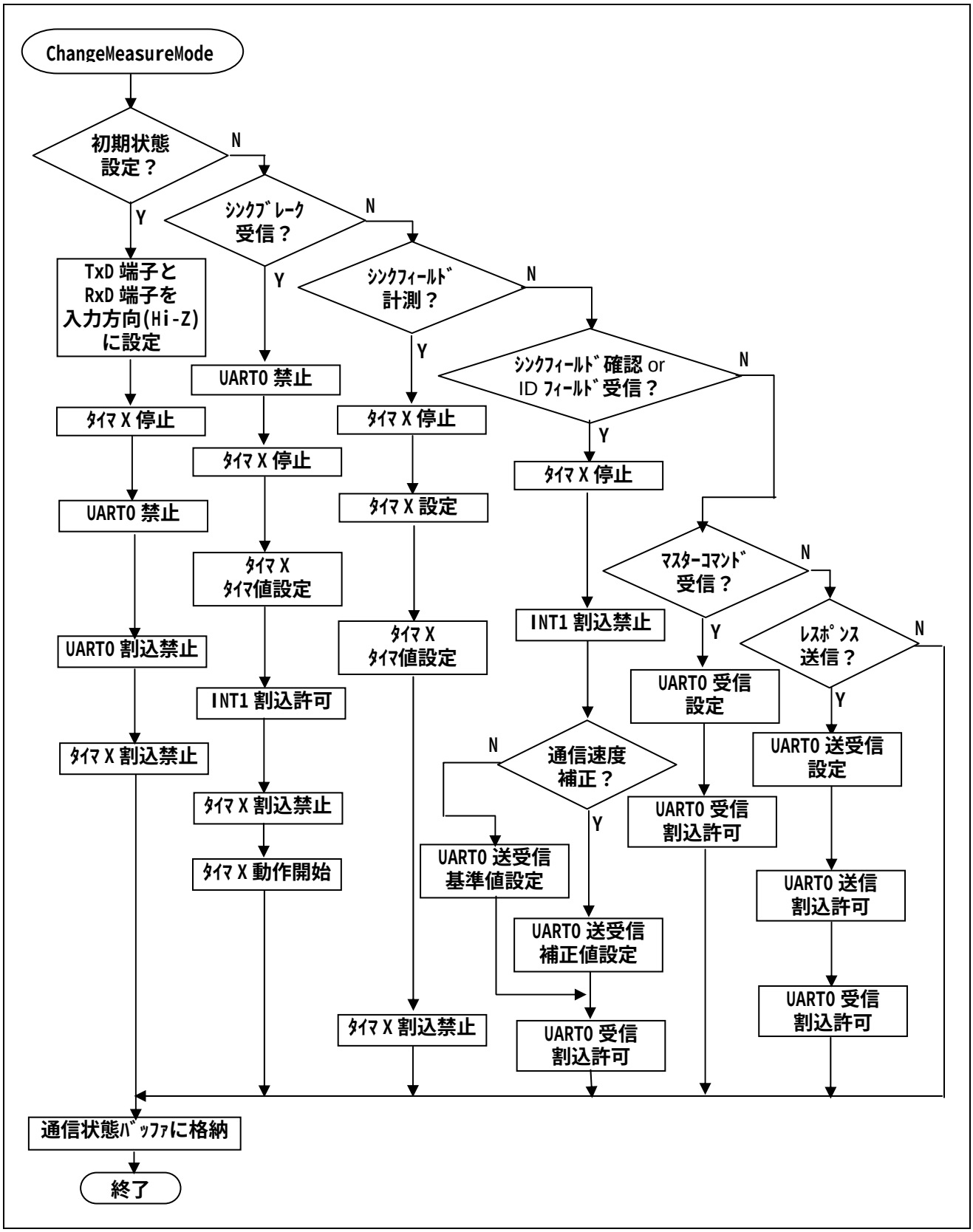


図４４　通信状態制御関数フローチャート

```
void ChangeMeasureMode(unsigned char mode)
{
    unsigned char Int1Level,Uart0Level;

    Int1Level = (ui_sHASW_status.StBYTE.Byte0 & 0x01c) >> 2;
    Uart0Level = (ui_sHASW_status.StBYTE.Byte0 & 0xe0) >> 5;

    switch(mode){
    case enHASW_INIT:          // 初期状態モード
        pdTxD = P_IN;                                   /* TxD Hi-Z */
        pdRxD = P_IN;                                   /* RxD Hi-Z */
        txmr = 0x00;                                   /* TimerX 動作停止 */
        while(txs==1);
        u0mr = 0x00;                                   /* UART モード禁止 */
        u0c0 = 0x20;                                   /* TxD 端子 Nch-O.D.設定 */
        u0c1 = 0x00;                                   /* UART0 送受信禁止 */
        intlic = 0x00;                                   /* INT1 割り込み禁止 */
        txic = 0x00;                                   /* TimerX 割り込み禁止 */
        s0tic = 0x00;                                   /* UART0 送信割り込み禁止 */
        s0ric = 0x00;                                   /* UART0 受信割り込み禁止 */
        break;
    case enSYNCH_BREAK:          // SynchBreak モード
        u0c1 = 0x00;                                   /* UART0 送受信禁止 */
        s0tic = 0x00;                                   /* UART0 送信割り込み禁止 */
        s0ric = 0x00;                                   /* UART0 受信割り込み禁止 */
        txmr = 0x03;                                /* TimerX 動作停止、パルス幅(L)計測モード設定 */
        while(txs==1);
        prex = 0xff;                                   /* TimerX = 0xffff */
        tx = 0xff;
        txic = 0x00;                                   /* TimerX 割り込み禁止 */
        intlic = Int1Level;                                   /* INT1 割り込み許可 */
        txmr = 0x0b;                                /* TimerX 動作開始、パルス幅(L)計測モード設定 */
        break;
#ifdef    __BPS_AUTO
    case enSYNCH_MEASURE:     // Synch 計測モード
        txmr = 0x04;                                   /* TimerX 動作停止、INT1↓エッジ設定 */
        while(txs==1);
        prex = 0xff;                                   /* TimerX = 0xffff */
        tx = 0xff;
        txic = 0x00;                                   /* TimerX 割り込み禁止 */
        break;
#endif
    case enSYNCH_SYNCH:          // Synch 同期モード
    case enIDENT:              // Ident 受信モード
        txmr = 0x00;                                   /* TimerX 動作停止 */
        while(txs==1);
        intlic = 0x00;                                   /* INT1 割り込み禁止 */
        u0mr = 0x05;                                   /* UART モード、8 ビット、1ST、NP */
        if (uc_fBaudrateCheck == SET) {                        // 転送レート補正あり？
            if (un_sU0BRGdata.StBYTE.Byte1 != 0x00) {
                un_sU0BRGdata.uiAll = un_sU0BRGdata.uiAll + (un_sU0BRGdata.uiAll & 0x0004);
                un_sU0BRGdata.uiAll >>= 3;
                u0c0 = 0x21;                           /* LSB,f8,TxD 端子 Nch-O.D.設定 */
            }
            else u0c0 = 0x20;                           /* LSB,f1,TxD 端子 Nch-O.D.設定 */
        }
        else {
```

```
            if (un_sU0BRGdata.StBYTE.Byte1 != 0x00) u0c0 = 0x21;
                                                    /* LSB,f8,TxD 端子 Nch-O.D.設定 */
            else u0c0 = 0x20;                        /* LSB,f1,TxD 端子 Nch-O.D.設定 */
        }
        u0brg = un_sU0BRGdata.StBYTE.Byte0 - 1;     /*  */
        u0c1 = 0x04;                                /* UART0 受信許可 */
        s0ric = Uart0Level;                            /* UART0 受信割り込み許可 */
        break;
    case enRESPONSE:          // Response 送信モード
        u0c1 = 0x05;                                /* UART0 送受信許可 */
        s0tic = Uart0Level;                            /* UART0 送信割り込み許可 */
        s0ric = Uart0Level;                            /* UART0 受信割り込み許可 */
        break;
    case enMASTERCMD:          // Command 受信モード
        u0c1 = 0x04;                                /* UART0 受信許可 */
        s0ric = Uart0Level;                            /* UART0 受信割り込み許可 */
        s0tic = 0x00;                                /* UART0 送信割り込み禁止 */
        break;
    case enCHECKSUM:
    default:
        break;
    }
    uc_cHaswStatus = mode;
}
```

3.6.13 ID パリティ計算処理関数

ID フィールドデータのパリティ値を計算処理を行います。

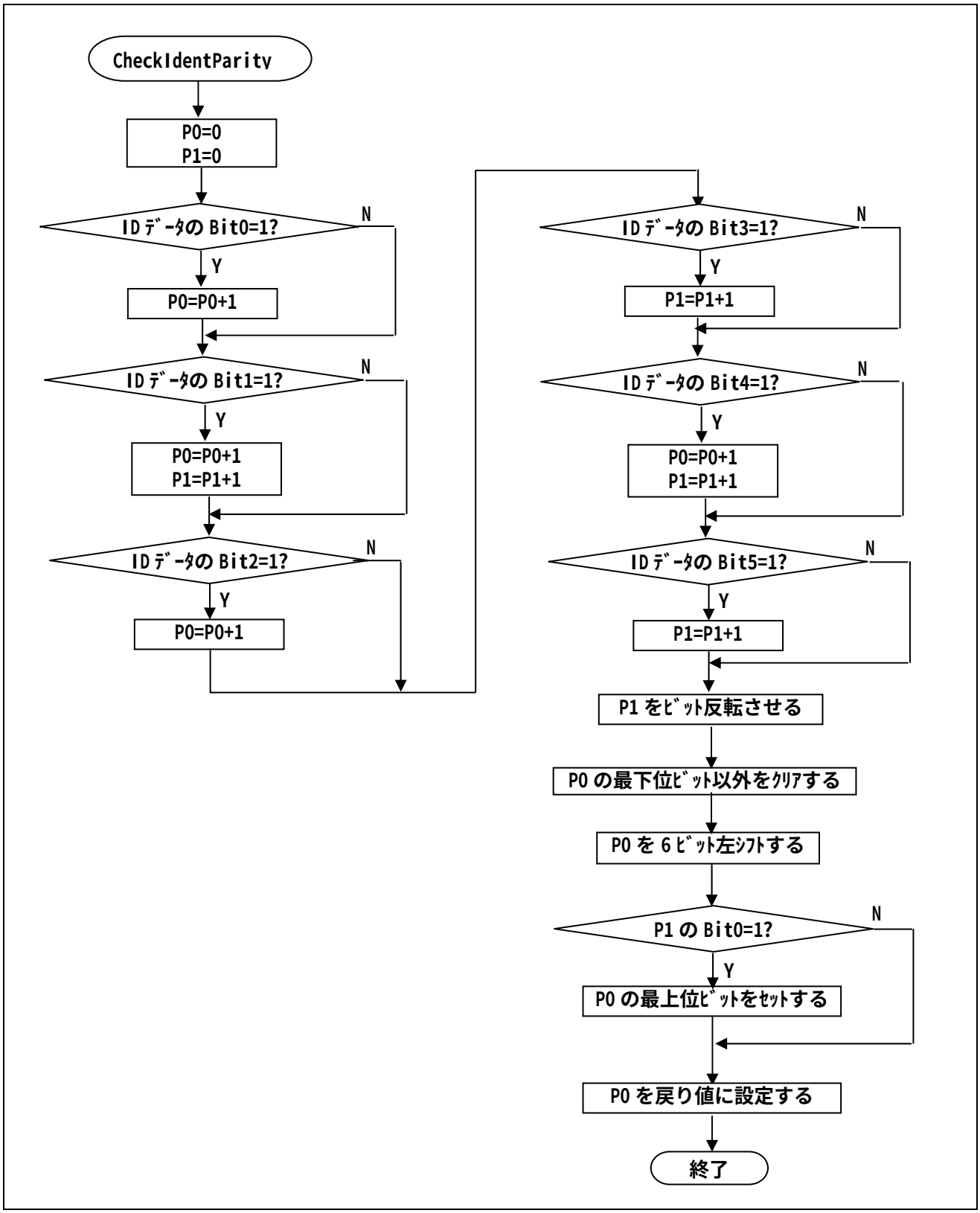


図４５　ID フィールドパリティ計算関数フローチャート

```
unsigned char CheckIdentParity(unsigned char ident_data)
{
    unsigned char    uc_p0,uc_p1;

    uc_p0 = uc_p1 = 0;
    if ((ident_data & 0x01) == 0x01) ++uc_p0;
    if ((ident_data & 0x02) == 0x02) {
        ++uc_p0;
        ++uc_p1;
    }
    if ((ident_data & 0x04) == 0x04) ++uc_p0;
    if ((ident_data & 0x08) == 0x08) ++uc_p1;
    if ((ident_data & 0x10) == 0x10) {
        ++uc_p0;
        ++uc_p1;
    }
    if ((ident_data & 0x20) == 0x20) ++uc_p1;
    uc_p1 = ~uc_p1;
    uc_p0 &= 0x01;
    uc_p0 <<= 6;
    if ((uc_p1&0x01) == 0x01) uc_p0 += 0x80;
    return uc_p0;
}
```

3.6.14 レスポンスコード検査処理関数

受信した ID コードがレスポンスコードかどうかを検査する処理を行います。

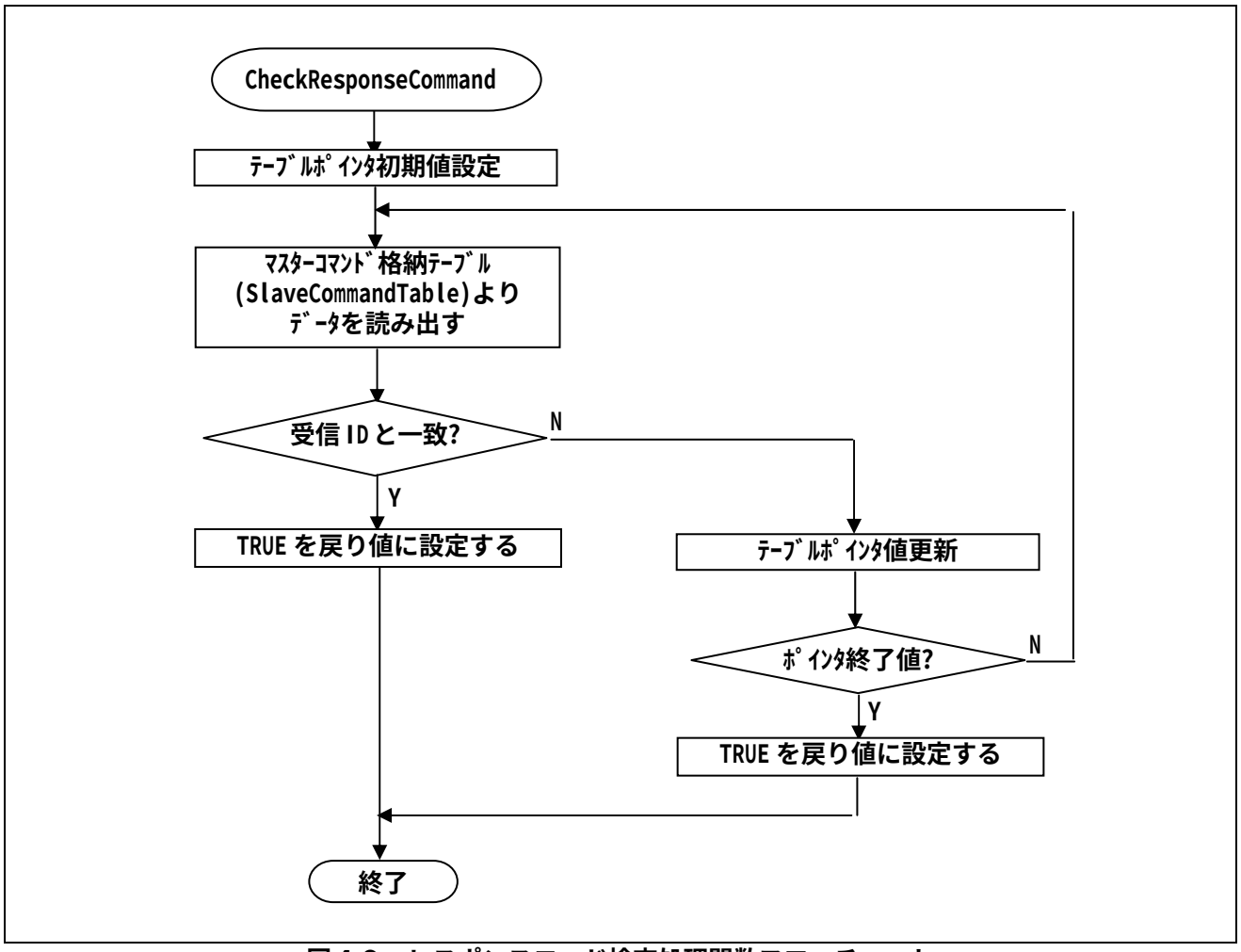


図４６　レスポンスコード検査処理関数フローチャート

```
unsigned char CheckResponseCommand(void)
{
    unsigned char    c;

    for (c=0;c < (unsigned char )(sizeof SlaveCommandTable);++c) {
        if (uc_bHaswBuff[enHASW_ID] == SlaveCommandTable[c]) {
            return TRUE;
        }
    }
    return FALSE;
}
```

3.6.15 チェックサム値計算処理関数

送受信するデータフィールドのチェックサム値を計算する処理を行います。

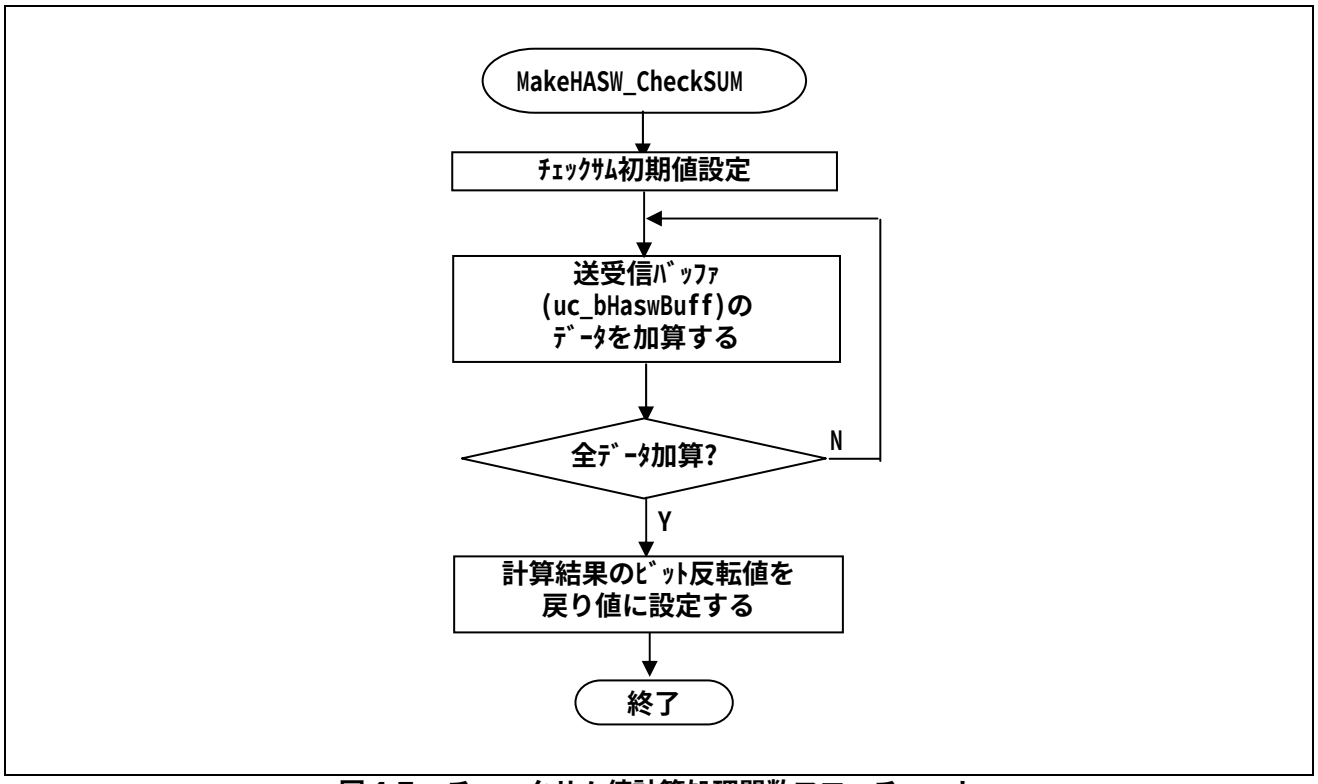


図４７　チェックサム値計算処理関数フローチャート

```
unsigned char MakeHASW_CheckSUM(unsigned char cnt)
{
    unsigned char    c;
    UnWORD    sum;

    sum.uiAll = 0;
    for(c=0;c<cnt;++c){
        sum.uiAll += (unsigned int )uc_bHaswBuff[c+enHASW_DATA0];
        sum.StBYTE.Byte0 += sum.StBYTE.Byte1;
        sum.StBYTE.Byte1 = 0;
    }
    return ~sum.StBYTE.Byte0;
}
```

3.6.16 通信エラー処理関数

一線式通信においてエラーが発生した場合の処理を行います。

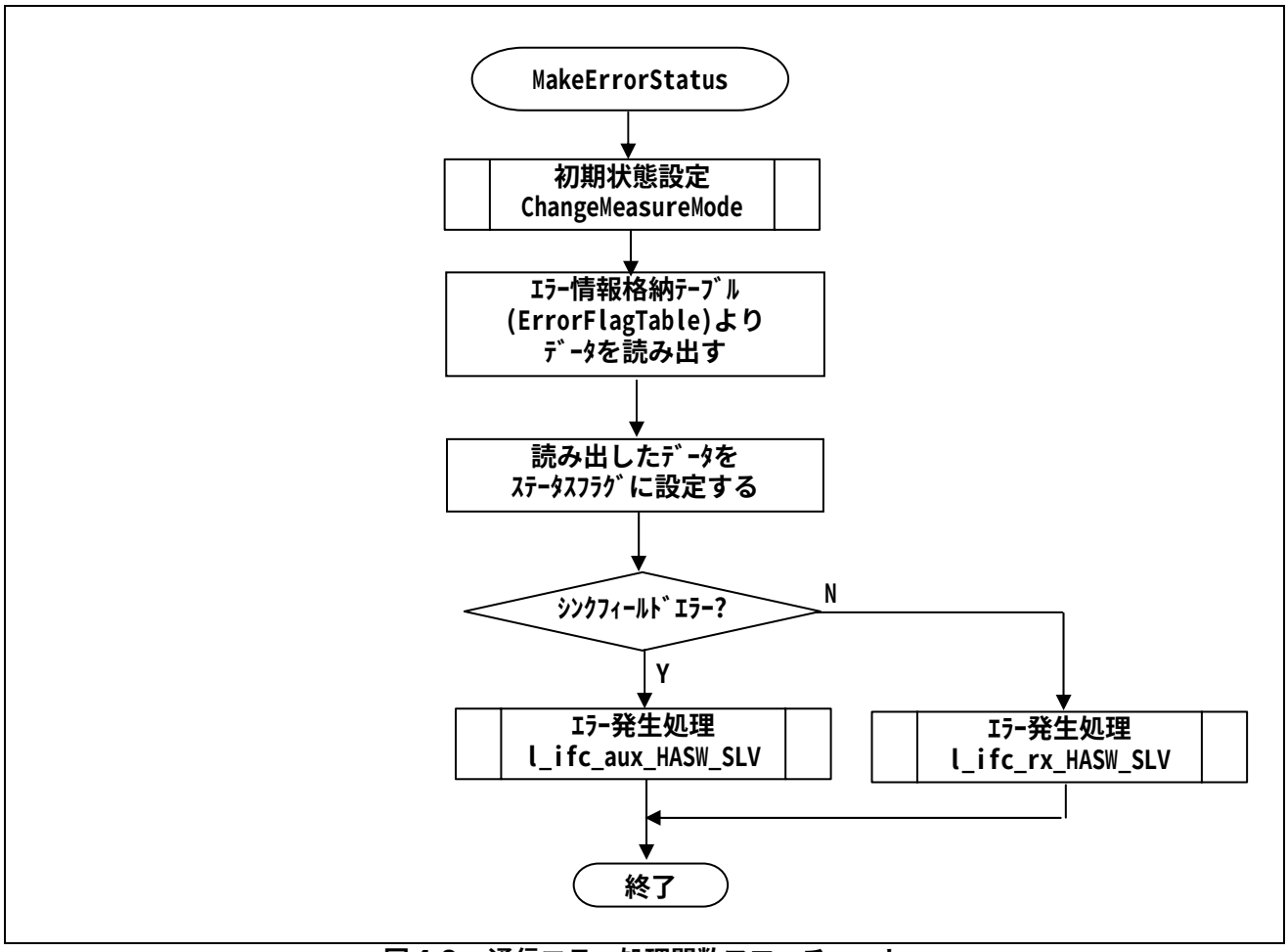


図４８　通信エラー処理関数フローチャート

```
void MakeErrorStatus(unsigned char ErrorCode)
{
    const unsigned char ErrorFlagTable[enERROR_MODE]={0x06,0x0a,0x12,0x22,0x42,0x82};
    ChangeMeasureMode(enHASW_INIT);                          // 初期状態設定
    ui_sHASW_status.StBYTE.Byte1 = ErrorFlagTable[ErrorCode];    // エラーフラグ設定
    if (ErrorCode == enERROR_SYNCH) {
        l_ifc_aux_HASW_SLV();                              // l_ifc_aux_HASW_SLV 関数呼び出し
    }
    else {
        l_ifc_rx_HASW_SLV();                               // l_ifc_rx_HASW_SLV 関数呼び出し
    }
}
```

4. LIN 通信への応用

LIN 通信への接続を行う場合，スリープ機能，ウェイクアップ機能，タイムアウト判定機能の実装が必要になります。本ライブラリではこれらの機能を実装しておりません。以下にそれぞれの機能を組み込む場合の応用方法について示します。

4.1 スリープ機能について

4.1.1 マスターノードの場合

マスターリクエストフレーム(ID=3dh)において，データフィールドの 1 バイト目が 00h の場合，スリープコマンドとなります。メイン処理にて，スリープ機能をマスターノードに実装する場合，マスターリクエストフレーム送信時にスリープコマンドを送信する機能を組み込みます。

4.1.2 スレーブノードの場合

マスターリクエストフレーム(ID=3dh)において，データフィールドの 1 バイト目が 00h の場合，スリープコマンドとなります。メイン処理にて，スリープ機能をスレーブノードに実装する場合，マスターリクエストフレーム受信時にスリープコマンドを受信した場合の処理を組み込みます。

4.2 ウェイクアップ機能について

4.2.1 マスターノードの場合

ウェイクアップ信号の送信を行う場合は，本ライブラリの l_sys_init_HASW_MST 関数の実行前もしくは，l_ifc_disconnect_HASW_MST 関数実行後に，UART0 送信機能を用いてウエイクアップ信号を送信する機能を組み込みます。また，ウェイクアップ信号の受信を行う場合は，l_sys_init_HASW_MST 関数の実行前もしくは，l_ifc_disconnect_HASW_MST 関数実行後に，INT 割り込み機能を用いて，ウェイクアップ信号の受信を行う機能を組み込みます。

4.2.2 スレーブノードの場合

ウェイクアップ信号の送信を行う場合は，本ライブラリの l_sys_init_HASW_SLV 関数の実行前もしくは，l_ifc_disconnect_HASW_SLV 関数実行後に，UART0 送信機能を用いてウエイクアップ信号を送信する機能を組み込みます。また，ウェイクアップ信号の受信を行う場合は，l_sys_init_HASW_SLV 関数の実行前もしくは，l_ifc_disconnect_HASW_SLV 関数実行後に，INT 割り込み機能を用いて，ウェイクアップ信号の受信を行う機能を組み込みます。

4.3 タイムアウト判定機能について

4.3.1 マスターノードの場合

データフィールド送受信時のタイムアウト判定はメイン処理にて判定してください。

4.3.2 スレーブノードの場合

データフィールド送受信時のタイムアウト判定はメイン処理にて判定してください。

5. 参考文献

- アプリケーションノート　H8/3664F/3694F/36014F シリーズ LIN マスター編
- アプリケーションノート　H8/3664F/3694F/36014F シリーズ LIN スレーブ編
- LIN Protocol Specification Revision 1.3
- R8C/18 グループハードウエアマニュアル　第一版
- R8C/24 グループハードウエアマニュアル　第一版

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	2006.09.06	—	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しては、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が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は負いません。
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には、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。