RZ/A2Mグループ
イーサネットドライバ

要旨

本アプリケーションノートは、RZ/A2M グループ用イーサネットドライバについて説明します。本ドライバはイーサネットコントローラ、イーサネットコントローラ用 DMA コントローラを使用して、イーサネットフレームの送受信を行います。以降、本モジュールをイーサネットドライバと称します。

イーサネットドライバを使用するためにはユーザプログラムでイーサネットコントローラの入出力信号を I/O ポートに割り当ててください。詳細は、4節を参照してください。

対象デバイス

以下は、この API によってサポートできるデバイスの一覧です。

RZ/A2M

目次

1. 概要

イーサネットドライバは、イーサネットコントローラ（以降、ETHERC と呼称）とイーサネットコントローラ用 DMA コントローラ（以降、EDMAC と呼称）を使用し、イーサネットフレームの送受信行うための手段を提供します。以下にイーサネットドライバがサポートしている機能を列挙します。

- MII（Media Independent Interface）および RMII（Reduced Media Independent Interface）に対応しています。
- イーサネット PHY-LSI のリンクには、自動交渉機能を用います。
- イーサネット PHY-LSI から出力されるリンク信号を用いて、リンク状態を検出します。
- イーサネット PHY-LSI からの自動交渉結果を取得し、接続モード（全二重モードまたは半二重モード、転送速度 10Mbps または 100Mbps）を ETHERC に設定します。

1.1 イーサネットドライバとは

イーサネットドライバは API として、プロジェクトに組み込んで使用します。イーサネットドライバの組み込み方については、「5 使用方法」を参照してください。

1.2 API の概要

表1.1にイーサネットドライバに含まれる API 関数を示します。

表1.1　API 関数一覧

関数	関数説明
R_ETHER_Initial()	イーサネットドライバの初期化を行います。
R_ETHER_Open_ZC2()	ETHERC と EDMAC および PHY-LSI をソフトウェアリセットした後、PHY-LSI のオートネゴシエーションを開始してリンク信号変化割り込みを許可します。
R_ETHER_Close_ZC2()	ETHERC の送信、受信機能をディゼーブル状態とします。ETHERC、EDMAC をモジュールストップにしません。
R_ETHER_Read()	指定した受信バッファへデータを受信します。
R_ETHER_Read_ZC2()	受信データが格納されたバッファの先頭アドレスへのポインタを返します。
R_ETHER_Read_ZC2_BufRelease()	R_ETHER_Read_ZC2 関数で読み出したバッファを開放します。
R_ETHER_Write()	指定した送信バッファからデータを送信します。
R_ETHER_Write_ZC2_GetBuf()	送信データの書き込み先の先頭アドレスへのポインタが返されます。
R_ETHER_Write_ZC2_SetBuf()	EDMAC に送信バッファのデータの送信を許可します。
R_ETHER_CheckLink_ZC()	物理的なイーサネットのリンク状態を、PHY 管理インタフェースを使用してチェックします。PHY が適切に初期化されている相手デバイスとケーブルが接続されていれば、イーサネットのリンク状態がリンクアップとなります。
R_ETHER_LinkProcess()	リンク信号変化割り込み処理およびマジックパケット検出割り込み処理を行います。
R_ETHER_WakeOnLAN()	ETHERC の設定を通常の送受信動作からマジックパケット検出動作に切り替えます。
R_ETHER_CheckWrite()	データ送信が完了したことを確認します。
R_ETHER_Control()	コントロールコードに対応した処理を行います。
R_ETHER_GetVersion()	イーサネットドライバのバージョン番号を返します。

2.　API 情報

イーサネットドライバの API はルネサスの API の命名基準に従っています。

2.1　サポートされているツールチェーン

イーサネットドライバは、「6 動作確認環境」に示すツールチェーンで動作確認を行っています。

2.2　使用する割り込み

引数にチャネル番号を指定して、R_ETHER_Open_ZC2 関数を実行するとチャネルに対応した EINT0 割り込み、EINT1 割り込みが有効になります。

2.3　ヘッダファイル

すべての API 呼び出しと使用されるインタフェース定義は r_ether_rza2_if.h に記載しています。

2.4　整数型

このプロジェクトは ANSI C99 を使用しています。これらの型は stdint.h で定義されています。

2.5　コンパイル時の設定

イーサネットドライバのコンフィギュレーションオプションの設定は、Smart Configurator を用いて変更します。変更された設定は r_ether_drv_sc_cfg.h に反映されます。オプション名および設定値に関する説明を、下表に示します。

Configuration options configured by Smart Configurator	
Ethernet channel 0 【注】デフォルト値は “Enabled”	ETHERC チャネル 0 を使用するかを選択します。 “Disabled”(0) の場合、チャネル 0 を使用しません “Enabled”(1) の場合、チャネル 0 を使用します
Ethernet channel 1 【注】デフォルト値は “Enabled”	ETHERC チャネル 1 を使用するかを選択します。 “Disabled”(0) の場合、チャネル 1 を使用しません “Enabled”(1) の場合、チャネル 1 を使用します
Ethernet interface select. 【注】デフォルト値は “RMII”	ETHERC とイーサネット PHY-LSI 間のインタフェースを設定してください。 “MII”(0) の場合、MII(Media Independent Interface)を選択します。 “RMII”(1) の場合、RMII（Reduced Media Independent Interface）を選択します。
PHY-LSI address setting for ETHER0. 【注】デフォルト値は “7”	ETHERC チャネル 0 が使用する PHY-LSI に割り当てられた PHY アドレスを設定してください。 “0” ～ “31” の範囲で設定しください。
PHY-LSI address setting for ETHER1. 【注】デフォルト値は “7”	ETHERC チャネル 1 が使用する PHY-LSI に割り当てられた PHY アドレスを設定してください。 “0” ～ “31” の範囲で設定しください。
Loop count for write PIR register. 【注】デフォルト値は “8”	PHY-LSI のリード/ライトに使用しているソフトウェアループのループ回数を設定します。ループ回数はご使用する PHY-LSI に合わせて設定してください。 “1” 以上の値を設定してください。
Define the waiting time for reset completion of PHY-LSI 【注】デフォルト値は “0x00020000”	PHY-LSIのリセット完了待ちのタイムアウト処理に使用しているループ回数を設定します。ループ回数はご使用する PHY-LSI に合わせて設定してください。
Link status read from LMON bit of ETHERC PSR register. 【注】デフォルト値は “(Link up/Link down)=(Falling/Rising)”(0)	PHY-LSI から出力されるリンク信号の極性を設定してください。 “(Link up/Link down)=(Falling/Rising)”(0) の場合、LINKSTA 信号の立ち下がり／立ち上がりで、リンクアップ／リンクダウンとなります。 "(Link up/Link down)=(Rising/Falling)”(1) の場合、LINKSTA 信号の立ち上がり／立ち下がりで、リンクアップ／リンクダウンとなります。
LINKSTA signal for detect link status changes 【注】デフォルト値は “the PHY-LSI status register is used.”(0)	リンク状態変化の検出において、LINKSTA 信号の代わりに PHY-LSIのステータスレジスタを使用するかを設定してください。 “the PHY-LSI status register is used.”(0) の場合、PHY-LSI のステータスレジスタを使用します。 “the LINKSTA signal is used.”(1) の場合、LINKSTA 信号を使用します。
Difference between physical address and virtual address of un-cached RAM. 【注】デフォルト値は “0x02000000”	非キャッシュ領域の論理 RAM アドレスと物理アドレスの差分を設定します。

Configuration options configured by macros in r_ether_drv_sc_cfg.h	
#define ETHER_CFG_EMAC_RX_DESCRIPTORS 【注】デフォルト値は “8”	受信ディスクリプタの数を設定してください。 “1” 以上の値を設定してください。
#define ETHER_CFG_EMAC_TX_DESCRIPTORS 【注】デフォルト値は “8”	送信ディスクリプタの数を設定してください。 “1” 以上の値を設定してください。
#define ETHER_CFG_BUFSIZE 【注】デフォルト値は “1536”	送信バッファ、受信バッファのサイズを設定してください。 バッファは 32 バイト境界で配置しますので、32 バイト単位の値を設定してください。

2.6　引数

API 関数の引数である列挙体、共用体、構造体を示します。これらは API 関数のプロトタイプ宣言とともに r_ether_rza2_if.h で記載されています。

```
typedef enum
{
  CONTROL_SET_CALLBACK,                 /* コールバック関数の登録 */
  CONTROL_SET_PROMISCUOUS_MODE,
                                        /* プロミスキャスモード設定 */
  CONTROL_SET_INT_HANDLER,              /* 割り込みハンドラ関数の登録 */
  CONTROL_POWER_ON,                     /* ETHERC/EDMAC モジュールストップ解除 */
  CONTROL_POWER_OFF,                    /* ETHERC/EDMAC モジュールストップ遷移 */
  CONTROL_MULTICASTFRAME_FILTER,        /* マルチキャストフレームフィルタ設定       */
  CONTROL_BROADCASTFRAME_FILTER         /* ブロードキャストフレームフィルタ連続     */
                                        /* 受信回数設定 */
} ether_cmd_t;

typedef union
{
    ether_cb_t            ether_callback;         /* コールバック関数ポインタ */
    ether_promiscuous_t   * p_ether_promiscuous;  /* プロミスキャスモード設定 */
    ether_cb_t            ether_int_hnd;          /* 割り込みハンドラ関数ポインタ */
    uint32_t              channel;                /* ETHERC のチャネル番号*/
    ether_multicast_t     * p_ether_multicast;
                                                  /* マルチキャストフレームフィルタ設定*/
    ether_broadcast_t     * p_ether_broadcast;
                                                  /*ブロードキャストフレームフィルタ設定 */
} ether_param_t;

typedef struct
{
  void    (*pcb_func)(void *);          /* コールバック関数ポインタ */
  void    (*pcb_int_hnd)(void *);       /* 割り込みハンドラ関数ポインタ */
} ether_cb_t;

typedef enum
{
  ETHER_PROMISCUOUS_OFF,                /* ETHERC は標準モード */
  ETHER_PROMISCUOUS_ON                  /* ETHERC はプロミスキャスモード */
} ether_promiscuous_bit_t;

typedef enum
{
    ETHER_MC_FILTER_OFF,                /* マルチキャストフレームフィルタは無効 */
    ETHER_MC_FILTER_ON                  /* マルチキャストフレームフィルタは有効 */
} ether_mc_filter_t;

typedef struct
{
  uint32_t                  channel;    /* ETHERC チャネル */
  ether_promiscuous_bit_t   bit;        /* プロミスキャスモード */
} ether_promiscuous_t;
```

```
typedef struct
{
    uint32_t                channel;    /* ETHERC チャネル */
    ether_mc_filter_t         flag;     /* マルチキャストフレームフィルタ設定 */
} ether_multicast_t;

typedef struct
{
    uint32_t                channel;    /* ETHERC チャネル */
    uint32_t                counter;    /* ブロードキャストフレーム連続受信回数 */
} ether_broadcast_t;

typedef enum
{
  ETHER_CB_EVENT_ID_WAKEON_LAN,         /* マジックパケット検出 */
  ETHER_CB_EVENT_ID_LINK_ON,            /* Link up 検出 */
  ETHER_CB_EVENT_ID_LINK_OFF            /* Link down 検出 */
} ether_cb_event_t;
typedef struct
{
  uint32_t            channel;     /* ETHERC チャネル */
  ether_cb_event_t    event_id;    /* コールバック関数用イベントコード */
  uint32_t            status_ecsr;/* 割り込みハンドラ関数用 ETHERC ステータスレジスタ */
  uint32_t            status_eesr;/* 割り込みハンドラ関数用 */
                                  /* ETHERC/EDMAC ステータスレジスタ */
} ether_cb_arg_t;
```

2.7　戻り値

API 関数の戻り値を示します。この列挙型は API 関数のプロトタイプ宣言とともに r_ether_rza2_if.h で記載されています。

```
typedef enum                /* Ether API のエラーコード*/
{
ETHER_SUCCESS,              /* 問題なく処理が終了した場合 */
ETHER_ERR_INVALID_PTR,      /* ポインタの値が、NULL の場合 */
ETHER_ERR_INVALID_DATA,     /* 引数のとり得る値が、範囲外の場合 */
ETHER_ERR_INVALID_CHAN,     /* 存在しないチャネルの場合 */
ETHER_ERR_INVALID_ARG,      /* 不正な引数の場合 */
ETHER_ERR_LINK,             /* オートネゴシエーション処理が完了しておらず受信が */
                            /* 許可されていない場合 */
ETHER_ERR_MPDE,             /* マジックパケットの検出状態のため、*/
                            /* 送信と受信が許可されていない場合*/
ETHER_ERR_TACT,             /* 送信バッファに空きがない場合 */
ETHER_ERR_CHAN_OPEN,        /* 他のアプリケーションが使用しているため */
                            /* Ether を Open できない場合 */
ETHER_ERR_MC_FRAME,         /* マルチキャストフレームフィルタ有効時に、マルチキャスト
                            /* フレームを検出した場合 */
ETHER_ERR_RECV_ENABLE,      /* 受信機能有効のため設定が変更できない場合 */
ETHER_ERR_OTHER             /* その他エラー */
} ether_return_t;
```

2.8　コールバック関数

2.8.1　API 関数 R_ETHER_LinkProcess から呼び出すコールバック関数

イーサネットドライバでは、マジックパケットの検出、または、リンク信号変化の検出があったとき、コールバック関数を呼び出します。

コールバック関数の設定は、後述の関数 R_ETHER_Control を用いて、「2.6　引数」に記載の列挙体（第 1 引数）には、コントロールコード “CONTROL_SET_CALLBACK” を、構造体（第 2 引数）には、コールバック関数として登録したい関数のアドレスを設定してください。

コールバック関数が呼び出されるとき、検出があったチャネル番号と表2.1に示す定数を格納した変数を、引数として渡します。引数の値をコールバック関数外で使用する場合は、グローバル変数などの変数にコピーしてください。

表2.1　コールバック関数の引数一覧

定数定義	意味
ETHER_CB_EVENT_ID_WAKEON_LAN	マジックパケットを検出した
ETHER_CB_EVENT_ID_LINK_ON	リンク信号変化（リンクアップ）を検出した
ETHER_CB_EVENT_ID_LINK_OFF	リンク信号変化（リンクダウン）を検出した

2.8.2　EINT0/EINT1 ステータス割り込みから呼び出す割り込みハンドラ関数

イーサネットドライバでは、以下に示した内容の割り込みがあったとき、割り込みハンドラ関数を呼び出します。

- イーサネットドライバがマジックパケット検出動作の場合
  - リンク信号変化の検出*1
  - マジックパケットの検出
- イーサネットドライバが通常動作の場合
  - リンク信号変化の検出*1
  - フレーム受信の検出、フレーム送信完了の検出

割り込みハンドラ関数の設定は、後述の関数 R_ETHER_Control を用いて、「2.6　引数」に記載の列挙体（第 1 引数）には、コントロールコード “CONTROL_SET_INT_HANDLER” を、構造体（第 2 引数）には、割り込みハンドラ関数として登録したい関数のアドレスを設定してください。

割り込みハンドラ関数が呼び出されるとき、割り込みがあったチャネル番号と ETHERC ステータスレジスタの値、ETHERC/EDMAC ステータスレジスタの値を格納した変数を、引数として渡します。引数の値をコールバック関数以外で使用する場合は、グローバル変数などの変数にコピーしてください。

【注】 *1 #define ETHER_CFG_USE_LINKSTA を値 0 に設定している場合には、リンク信号変化の検出による割り込みハンドラ関数の呼び出しは発生しません。

2.9　イーサネットフレームのフレーム形式

イーサネットドライバは、Ethernet II/IEEE802.3 のフレーム形式をサポートしています。

2.9.1　データ送受信時のフレーム形式

図 2.1に Ethernet II/IEEE802.3 のフレーム形式を示します。

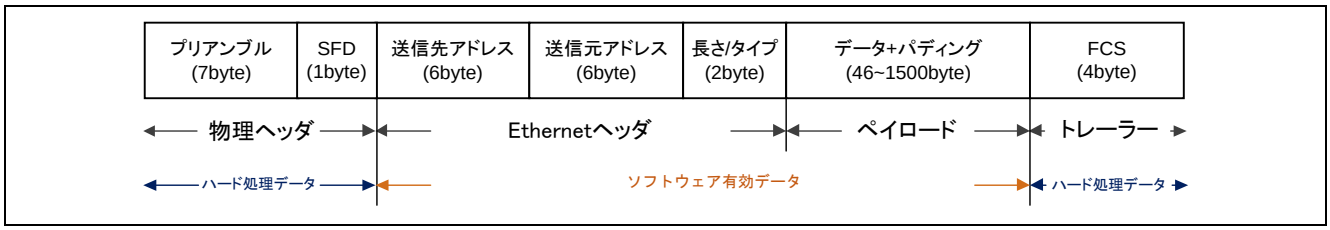

図 2.1　Ethernet II /IEEE802.3 のフレーム形式

- プリアンブルおよび SFD は、イーサネットフレームの始まりを合図するための信号です。また FCS は、送信側で計算したイーサネットフレームの CRC 値は格納されており、ハードウェアがデータ受信時に同様に CRC 値を計算して一致しない場合のイーサネットフレームは破棄されます。
- ハードウェアが正常データと判断した場合における受信データの有効範囲は、（送信先アドレス）+（送信元アドレス）+（長さ/タイプ）+（データ）となります。

2.9.2　PAUSE フレームのフレーム形式

図 2.2に PAUSE フレームのフレーム形式を示します。

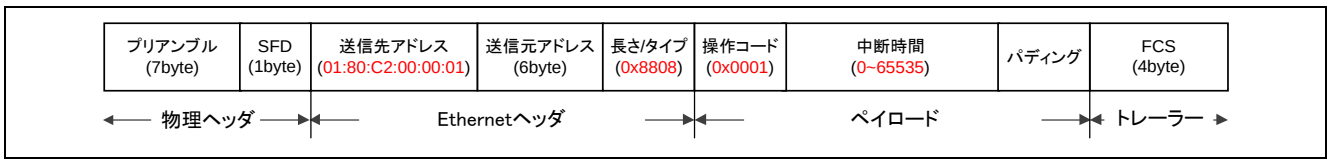

図 2.2　PAUSE フレームのフレーム形式

- 送信先アドレスには「01:80:C2:00:00:01」（PAUSE フレーム用に予約されているマルチキャストアドレス）が指定されます。また、長さ/タイプには「0x8808」、ペイロードの先頭に操作コードとして「0x0001」が指定されます。
- ペイロードの中断時間は「自動 PAUSE フレーム設定レジスタ（APR）」の「自動 PAUSE ビット（AP）」もしくは「手動 PAUSE フレーム設定レジスタ（MPR）」の「手動 PAUSE ビット（MP）」の値が指定されます。

2.9.3　マジックパケットのフレーム形式

図 2.3にマジックパケットのフレーム形式を示します。

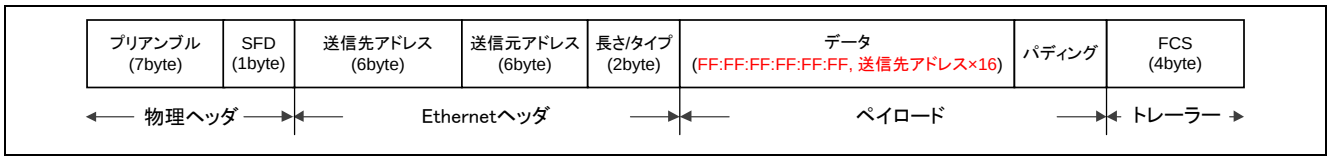

図 2.3　マジックパケットのフレーム形式

- マジックパケットはイーサフレームのデータのどこかに、「FF:FF:FF:FF:FF:FF」の後に「送信先アドレスを 16 回繰り返した値」を挿入します。

3.　API 関数

3.1　R_ETHER_Initial()

イーサネットドライバの初期設定を行う関数です。

Format

void R_ETHER_Initial(void);

Parameters

なし

Return Values

なし

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

イーサネット通信を開始するため、使用するメモリの初期化を行います。

Reentrant

- 不可

Example

```
#include "r_ether_rza2_if.h"

void callback_sample(void*);
void int_handler_sample(void*);

ether_return      ret;
ether_param_t     param;
ether_cb_t           cb_func;

/* Ethernet channel number
 * ETHER_CHANNEL_0 = Ethernet channel number is 0
 * ETHER_CHANNEL_1 = Ethernet channel number is 1
 */
uint32_t          channel;

/* Initialize memory which ETHERC/EDMAC is used */
R_ETHER_Initial();

channel        = ETHER_CHANNEL_0
param.channel  = channel;

/* Set the callback function */
cb_func.pcb_func    = &callback_sample;
param.ether_callback = cb_func;
ret = R_ETHER_Control(CONTROL_SET_CALLBACK, param);

/* Set the interrupt handler */
cb_func.pcb_int_hnd = &int_handler_sample;
param.ether_int_hnd = cb_func;
ret = R_ETHER_Control(CONTROL_SET_INT_HANDLER, param);

/* Release ETHERC and EDMAC module stop, port settings using ETHERC */
ret = R_ETHER_Control(CONTROL_POWER_ON, param);
if(ETHER_SUCCESS == ret)
{
  /* Initialized successfully completed without ETHERC, EDMAC*/
}
```

Special Notes:

R_ETHER_Open_ZC2 関数よりも前で呼び出してください。

3.2　R_ETHER_Open_ZC2()

ETHER の API を使用する際に、最初に使用する関数です。

Format

```
ether_return_t R_ETHER_Open_ZC2(
        uint32_t    channel              /* ETHERC のチャネル番号 */
        const       uint8_t  mac_addr[]  /* ETHERC の MAC アドレス */
        uint8_t     pause                /*  フロー制御機能の有効/無効 */
);
```

Parameters

channel

ETHERC/EDMAC のチャネル番号（0、1）を指定します。ETHERC/EDMAC を 1 チャネルのみ搭載する製品の場合は必ずチャネル番号 0 を指定してください。

mac_addr

ETHERC の MAC アドレスを指定します。

pause

PHY-LSI のレジスタ 4（Auto-Negotiation Advertisement）のビット 10（Pause）に設定する値を指定します。ユーザが使用する PHY-LSI が Pause 機能に対応している場合のみ ETHER_FLAG_ON の指定が可能です。この値はオートネゴシエーション時に相手側の PHY-LSI に引き渡されます。オートネゴシエーションの結果、自分の PHY-LSI と相手側の PHY-LSI の両方が Pause 機能に対応している場合はフロー制御が有効となります。

Pause 機能に対応していることをオートネゴシエーション時に相手側の PHY-LSI に伝達したい場合は、ETHER_FLAG_ON を、Pause 機能対応していない場合または対応していても使わない場合は、ETHER_FLAG_OFF を指定してください。

Return Values

```
ETHER_SUCCESS              /*  問題なく処理が完了した場合 */
ETHER_ERR_INVALID_CHAN     /*  存在しないチャネルの場合 */
ETHER_ERR_INVALID_PTR      /*  ポインタの値が、NULL の場合 */
ETHER_ERR_INVALID_DATA     /*  引数のとり得る値が、範囲外の場合 */
ETHER_ERR_OTHER            /* PHY-LSI の初期化に失敗した場合 */
```

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

R_ETHER_Open_ZC2 関数は ETHERC と EDMAC および PHY-LSI をソフトウェアリセットした後、PHY-LSI のオートネゴシエーションを開始し、リンク信号変化割り込みを許可します。

MAC アドレスは ETHERC の MAC アドレスレジスタを初期化するために使用されます。

Reentrant

- 異なるチャネルからリエントラントは可能です。

Example

- サンプルコードに含まれるMACアドレスはルネサスエレクトロニクス株式会社のベンダIDから割り当てられたアドレスを使用しています。お客様が製品化する際には必ずIEEEに申請したMACアドレスを使用するようにしてください。

```
#include "r_ether_rza2_if.h"

ether_return   ret;

/* Source MAC Address */
static uint8_t   mac_addr_src[6] = {0x74,0x90,0x50,0x00,0x79,0x01};

/* Flow control function
 * ETHER_FLAG_ON  = Use flow control function
 * ETHER_FLAG_OFF = No use flow control function
 */
static volatile uint8_t  pause_enable = ETHER_FLAG_OFF;

/* Ethernet channel number
 * ETHER_CHANNEL_0 = Ethernet channel number is 0
 * ETHER_CHANNEL_1 = Ethernet channel number is 1
 */
uint32_t        channel;

channel = ETHER_CHANNEL_0;

/* Initialize ETHERC, EDMAC */
ret = R_ETHER_Open_ZC2(channel, mac_addr_src, pause_enable);
if(ETHER_SUCCESS == ret)
{
  while(1)
  {
    /* Check Link status when Initialized successfully completed */
    R_ETHER_LinkProcess(channel);
  }
}
```

Special Notes:

- パワーオンリセット後にR_ETHER_Initial関数を実行した後、およびR_ETHER_Close_ZC2関数を実行した後は、必ず本関数を実行して戻り値がETHER_SUCCESSであることを確認した後、他のAPIをご使用ください。

-

3.3　R_ETHER_Close_ZC2()

R_ETHER_Close_ZC2 関数は ETHERC の送信、受信機能をディゼーブル状態にします。この関数は ETHERC、EDMAC をモジュールストップにしません。

Format

```
ether_return_t R_ETHER_Close_ZC2(
        uint32_t    channel    /* ETHERC のチャネル番号 */
);
```

Parameters

channel

ETHERC/EDMAC のチャネル番号（0、1）を指定します。ETHERC/EDMAC を 1 チャネルのみ搭載する製品の場合は必ずチャネル番号 0 を指定してください。

Return Values

ETHER_SUCCESS	/* 問題なく処理が完了した場合 */
ETHER_ERR_INVALID_CHAN	/* 存在しないチャネルの場合 */

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

R_ETHER_Close_ZC2 関数は ETHERC の送信、受信機能およびイーサネット割り込みをディゼーブル状態にします。ETHERC、EDMAC をモジュールストップにしません。

本関数はイーサネット通信を終了する場合に実行してください。

Reentrant

- 異なるチャネルからリエントラントは可能です。

Example

```
#include "r_ether_rza2_if.h"

ether_return   ret;

/* Ethernet channel number
 * ETHER_CHANNEL_0 = Ethernet channel number is 0
 * ETHER_CHANNEL_1 = Ethernet channel number is 1
 */
uint32_t        channel;

channel = ETHER_CHANNEL_0;

/* Disable transmission and receive function */
ret = R_ETHER_Close_ZC2(channel);
if(ETHER_SUCCESS == ret)
{
  goto end;
}
```

Special Notes:

なし

3.4　R_ETHER_Read_ZC2()

R_ETHER_Read_ZC2 関数は受信データが格納されたバッファの先頭アドレスへのポインタを返します。

Format

```
int32_t R_ETHER_Read_ZC2(
        uint32_t    channel     /* ETHERC のチャネル番号 */
        void**      pbuf        /* 受信データが格納されたバッファポインタ */
);
```

Parameters

channel

ETHERC/EDMAC のチャネル番号（0、1）を指定します。ETHERC/EDMAC を 1 チャネルのみ搭載する製品の場合は必ずチャネル番号 0 を指定してください。

** pbuf

受信データが格納されたバッファの先頭アドレスへのポインタを返します。

Return Values

1 以上の値	*/* 受信したバイト数 */*
ETHER_NO_DATA	*/* ゼロが返されたときは、データが受信されていません */*
ETHER_ERR_INVALID_CHAN	*/* 存在しないチャネルの場合 */*
ETHER_ERR_INVALID_PTR	*/* ポインタの値が、NULL の場合 */*
ETHER_ERR_LINK	*/* オートネゴシエーション処理が完了しておらず受信が */*
	/ 許可されていない場合 */*
ETHER_ERR_MPDE	*/* マジックパケットの検出状態のため、 */*
	/ 送信と受信が許可されていない場合 */*
ETHER_ERR_MC_FRAME	*/* マルチキャストフレームフィルタ有効時に、 */*
	/ マルチキャストフレームを受信した場合 */*

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

受信データが格納されたバッファの先頭アドレスへのポインタはパラメータ pbuf に格納して返されます。返されたポインタを利用して、ゼロコピーで操作が行えます。

戻り値は受信されたバイト数を示しています。呼び出し時に、データが存在しないときには値 ETHER_NO_DATA が返されます。オートネゴシエーション処理が完了しておらず受信が許可されていないときには値 ETHER_ERR_LINK が返されます。マジックパケット検出状態となっているときには値 ETHER_ERR_MPDE が返されます。

EDMAC は R_ETHER_Read_ZC2 関数とは独立して動作します。EDMAC は受信ディスクリプタで指定されたバッファにデータを読み込みます。EDMAC の受信ディスクリプタが指しているバッファはイーサネットドライバによって静的に割り当てられます。

R_ETHER_Control 関数で指定チャネルのマルチキャストフレームフィルタを有効にしている場合、マルチキャストフレームを検出すると直ちにバッファを開放します。また値 ETHER_ERR_MC_FRAME が返されます。

受信 FIFO オーバフロー、端数ビットフレーム受信エラー、ロングフレーム受信エラー、ショートフレーム受信エラー、PHY-LSI 受信エラー、受信フレーム CRC エラーが発生したフレームは受信フレームエラーとなります。受信フレームエラーが発生したディスクリプタのデータは破棄され、ステータスをクリアして読み込みを継続します。

Reentrant

- 異なるチャネルからリエントラントは可能です。

Example

```
#include <string.h>
#include "r_ether_rza2_if.h"

ether_return   ret;
uint8_t      * pread_buffer_address;
uint8_t      * pbuf;

/* Ethernet channel number
 * ETHER_CHANNEL_0 = Ethernet channel number is 0
 * ETHER_CHANNEL_1 = Ethernet channel number is 1
 */
uint32_t        channel;

channel = ETHER_CHANNEL_0;

ret = R_ETHER_Read_ZC2(channel, (void **)&pread_buffer_address);
/* When there is data to receive */
if(ETHER_NO_DATA < ret)
{
  memcpy(pbuf, pread_buffer_address, (uint32_t)ret);

  /* Release the receive buffer after reading the receive data. */
  R_ETHER_Read_ZC2_BufRelease(channel);
}
```

Special Notes:

- 本関数は R_ETHER_Read_ZC2_BufRelease 関数とセットで使用されますので、必ず R_ETHER_Read_ZC2 関数、R_ETHER_Read_ZC2_BufRelease 関数の順序で呼び出してください。また、本関数を呼び出して値 ETHER_ERR_LINK が返却された場合は、イーサネットドライバを初期化してください。

3.5　R_ETHER_Read_ZC2_BufRelease()

R_ETHER_Read_ZC2_BufRelease 関数は R_ETHER_Read_ZC2 関数で読み出したバッファを開放します。

Format

```
int32_t R_ETHER_Read_ZC2_BufRelease(
        uint32_t    channel     /* ETHERC のチャネル番号を指定します */
);
```

Parameters

channel

ETHERC/EDMAC のチャネル番号（0、1）を指定します。ETHERC/EDMAC を 1 チャネルのみ搭載する製品の場合は必ずチャネル番号 0 を指定してください。

Return Values

ETHER_SUCCESS	*/* 問題なく処理が完了した場合 */*
ETHER_ERR_INVALID_CHAN	*/* 存在しないチャネルの場合 */*
ETHER_ERR_LINK	*/* オートネゴシエーション処理が完了しておらず受信が */*
	/ 許可されていない場合 */*
ETHER_ERR_MPDE	*/* マジックパケットの検出状態のため、 */*
	/ 送信と受信が許可されていない場合 */*

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

R_ETHER_Read_ZC2_BufRelease 関数は R_ETHER_Read_ZC2 関数で読み出したバッファを開放します。

Reentrant

- 異なるチャネルからリエントラントは可能です。

Example

```
#include <string.h>
#include "r_ether_rza2_if.h"

ether_return   ret;
uint8_t      * pread_buffer_address;
uint8_t      * pbuf;

/* Ethernet channel number
 * ETHER_CHANNEL_0 = Ethernet channel number is 0
 * ETHER_CHANNEL_1 = Ethernet channel number is 1
 */
uint32_t        channel;

channel = ETHER_CHANNEL_0;

ret = R_ETHER_Read_ZC2(channel, (void **)&pread_buffer_address);
/* When there is data to receive */
if(ETHER_NO_DATA < ret)
{
  memcpy(pbuf, pread_buffer_address, (uint32_t)ret);
  /* Release the receive buffer after reading the receive data. */
  R_ETHER_Read_ZC2_BufRelease(channel);
}
```

Special Notes:

- 本関数は R_ETHER_Read_ZC2 関数でデータを読み出し、1 以上の値が返却された後に呼び出してください。
- 本関数は R_ETHER_Read_ZC2 関数とセットで使用されますので、必ず R_ETHER_Read_ZC2 関数、R_ETHER_Read_ZC2_BufRelease 関数の順序で呼び出してください。また、本関数を呼び出して値 ETHER_ERR_LINK が返却された場合は、イーサネットドライバを初期化してください。

3.6　R_ETHER_Write_ZC2_GetBuf()

R_ETHER_Write_ZC2_GetBuf 関数は送信データの書き込み先の先頭アドレスへのポインタが返されます。

Format

```
ether_return_t R_ETHER_Write_ZC2_GetBuf(
        uint32_t        channel     /* ETHERC のチャネル番号 */
        void        **  pbuf        /* 送信データの書き込み先の先頭アドレスへのポインタ*/
        uint16_t    *   pbuf_size   /* バッファに書き込み可能な上限サイズ */
);
```

Parameters

channel

ETHERC/EDMAC のチャネル番号（0、1）を指定します。ETHERC/EDMAC を 1 チャネルのみ搭載する製品の場合は必ずチャネル番号 0 を指定してください。

** pbuf

送信データの書き込み先の先頭アドレスへのポインタが返されます。

* pbuf_size

バッファに書き込み可能な上限サイズが返されます。

Return Values

ETHER_SUCCESS	*/* 問題なく処理が完了した場合 */*
ETHER_ERR_INVALID_CHAN	*/* 存在しないチャネルの場合 */*
ETHER_ERR_INVALID_PTR	*/* ポインタの値が、NULL の場合 */*
ETHER_ERR_LINK	*/* オートネゴシエーション処理が完了しておらず受信が */*
	/ 許可されていない場合 */*
ETHER_ERR_MPDE	*/* マジックパケットの検出状態のため、 */*
	/ 送信と受信が許可されていない場合 */*
ETHER_ERR_TACT	*/* 送信バッファに空きがない場合 */*

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

送信データの書き込み先の先頭アドレスへのポインタはパラメータ pbuf に格納して返されます。またバッファに書き込み可能な上限サイズはパラメータ pbuf_size に返されます。返されたポインタを利用して、ゼロコピーで操作が行えます。

戻り値は送信バッファ（pbuf）へ書き込みが可能であるか示しています。呼び出し時に、書き込みが可能なときには ETHER_SUCCESS が返されます。オートネゴシエーション処理が完了しておらず送信が許可されていないときには値 ETHER_ERR_LINK が返されます。マジックパケット検出状態となっているときには値 ETHER_ERR_MPDE が返されます。送信バッファに空きがないときには値 ETHER_ERR_TACT が返されます。

EDMAC は R_ETHER_Write_ZC2_GetBuf 関数とは独立して動作します。EDMAC は送信ディスクリプタで指定されたバッファのデータを書き出します。EDMAC の送信ディスクリプタが指しているバッファはイーサネットドライバによって静的に割り当てられます。

Reentrant

- 異なるチャネルからリエントラントは可能です。

Example

- サンプルコードに含まれるMACアドレスはルネサスエレクトロニクス株式会社のベンダIDから割り当てられたアドレスを使用しています。お客様が製品化する際には必ずIEEEに申請したMACアドレスを使用するようにしてください。

```
#include <string.h>
#include "r_ether_rza2_if.h"

ether_return   ret;
uint8_t      * pwrite_buffer_address;
uint8_t      * pbuf;
uint16_t       buf_size;

/* Transmit data */
static uint8_t send_data[60] =
{
    0x74,0x90,0x50,0x00,0x79,0x02,      /* Destination MAC address    */
    0x74,0x90,0x50,0x00,0x79,0x01,      /* Source MAC address         */
    0x00,0x00,                          /* The type field is not used    */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    /* Data field     */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00
};

/* Ethernet channel number
 * ETHER_CHANNEL_0 = Ethernet channel number is 0
 * ETHER_CHANNEL_1 = Ethernet channel number is 1
 */
uint32_t        channel;

channel = ETHER_CHANNEL_0;

ret = R_ETHER_Write_ZC2_GetBuf(channel, (void **)&pwrite_buffer_address,
&buf_size);
/* When transmission buffer is empty */
if(ETHER_SUCCESS == ret)
{
  /* Write the transmit data to the transmission buffer. */
  memcpy(pwrite_buffer_address, send_data, sizeof(send_data));

  R_ETHER_Write_ZC2_SetBuf(channel, sizeof(send_data));

  /* Verifying that the transmission is completed */
  ret = R_ETHER_CheckWrite(channel);
  if(ETHER_SUCCESS == ret)
  {
    /* Transmission is completed */
  }
}
```

Special Notes:

- 本関数はR_ETHER_Write_ZC2_SetBuf関数とセットで使用されますので、必ずR_ETHER_Write_ZC2_GetBuf関数、R_ETHER_Write_ZC2_SetBuf関数の順序で呼び出してください。また、本関数を呼び出して値ETHER_ERR_LINKが返却された場合は、イーサネットドライバを初期化してください。

3.7　R_ETHER_Write_ZC2_SetBuf()

R_ETHER_Write_ZC2_SetBuf 関数は EDMAC に送信バッファのデータの送信を許可します。

Format

```
ether_return_t R_ETHER_Write_ZC2_SetBuf(
        uint32_t        channel     /* ETHERC のチャネル番号 */
        const uint32_t  len         /* イーサネットフレーム長から CRC の 4 バイトを */
                                    /* 除いたサイズ（60～1514） */
);
```

Parameters

channel

ETHERC/EDMAC のチャネル番号（0、1）を指定します。ETHERC/EDMAC を 1 チャネルのみ搭載する製品の場合は必ずチャネル番号 0 を指定してください。

len

イーサネットフレーム長から CRC の 4 バイトを除いたサイズ（60～1514）を指定します。

Return Values

ETHER_SUCCESS	*/* 問題なく処理が完了した場合 */*
ETHER_ERR_INVALID_CHAN	*/* 存在しないチャネルの場合 */*
ETHER_ERR_INVALID_DATA	*/* 引数のとり得る値が、範囲外の場合 */*
ETHER_ERR_LINK	*/* オートネゴシエーション処理が完了しておらず受信が */*
	/ 許可されていない場合 */*
ETHER_ERR_MPDE	*/* マジックパケットの検出状態のため、 */*
	/ 送信と受信が許可されていない場合 */*

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

本関数は 1 フレームの送信データの書き込みが完了した後、呼び出してください。

バッファ長に指定する値は、イーサネットフレームの最小値 64 バイトから CRC の 4 バイトを除いた 60 バイト以上かつイーサネットフレームの最大値 1518 バイトから CRC の 4 バイトを除いた 1514 バイト以下までの範囲としてください。

60 バイト未満のデータを送信する場合は、データを 0 パディングで埋めて 60 バイトとなるようにしてください。

戻り値は送信バッファに書き込んだデータの送信許可状態を示しています。呼び出し時に、送信バッファのデータの送信が許可されたときには ETHER_SUCCESS が返されます。オートネゴシエーション処理が完了しておらず送信が許可されていないときには値 ETHER_ERR_LINK が返されます。マジックパケット検出状態となっているときには値 ETHER_ERR_MPDE が返されます。

Reentrant

- 異なるチャネルからリエントラントは可能です。

Example

- サンプルコードに含まれるMACアドレスはルネサスエレクトロニクス株式会社のベンダIDから割り当てられたアドレスを使用しています。お客様が製品化する際には必ずIEEEに申請したMACアドレスを使用するようにしてください。

```
#include <string.h>
#include "r_ether_rza2_if.h"

ether_return   ret;
uint8_t      * pwrite_buffer_address;
uint8_t      * pbuf;
uint16_t       buf_size;

/* Transmit data */
static uint8_t send_data[60] =
{
    0x74,0x90,0x50,0x00,0x79,0x02,      /* Destination MAC address    */
    0x74,0x90,0x50,0x00,0x79,0x01,      /* Source MAC address         */
    0x00,0x00,                          /* The type field is not used    */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    /* Data field     */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00
};

/* Ethernet channel number
 * ETHER_CHANNEL_0 = Ethernet channel number is 0
 * ETHER_CHANNEL_1 = Ethernet channel number is 1
 */
uint32_t        channel;

channel = ETHER_CHANNEL_0;

ret = R_ETHER_Write_ZC2_GetBuf(channel, (void **)&pwrite_buffer_address,
&buf_size);
/* When transmission buffer is empty */
if(ETHER_SUCCESS == ret)
{
  /* Write the transmit data to the transmission buffer. */
  memcpy(pwrite_buffer_address, send_data, sizeof(send_data));

  R_ETHER_Write_ZC2_SetBuf(channel, sizeof(send_data));

  /* Verifying that the transmission is completed */
  ret = R_ETHER_CheckWrite(channel);
  if(ETHER_SUCCESS == ret)
  {
    /* Transmission is completed */
  }
}
```

Special Notes:

- 本関数は 1 フレームの送信データの書き込みが完了した後、呼び出してください。
- 60 バイト未満のデータを送信する場合は、データを 0 パディングで埋めて 60 バイトとなるようにしてください。
- 本関数は R_ETHER_Write_ZC2_GetBuf 関数でデータを読み出し、値 ETHER_SUCCESS が返却された後に、呼び出してください。
- 本関数は R_ETHER_Write_ZC2_GetBuf 関数とセットで使用されますので、必ず R_ETHER_Write_ZC2_GetBuf 関数、R_ETHER_Write_ZC2_SetBuf 関数の順序で呼び出してください。また、本関数を呼び出して値 ETHER_ERR_LINK が返却された場合は、イーサネットドライバを初期化してください。

3.8　R_ETHER_CheckLink_ZC()

R_ETHER_CheckLink_ZC は物理的なイーサネットのリンク状態を、PHY 管理インタフェースを使用してチェックします。PHY が適切に初期化されている相手デバイスとケーブルが接続されていれば、イーサネットのリンク状態がリンクアップとなります。

Format

```
ether_return_t    R_ETHER_CheckLink_ZC(
        uint32_t       channel          /* ETHERC のチャネル番号 */
);
```

Parameters

channel

ETHERC/EDMAC のチャネル番号（0、1）を指定します。ETHERC/EDMAC を 1 チャネルのみ搭載する製品の場合は必ずチャネル番号 0 を指定してください。

Return Values

ETHER_SUCCESS	*/* リンク状態がリンクアップの場合 */*
ETHER_ERR_OTHER	*/* リンク状態がリンクダウンの場合 */*
ETHER_ERR_INVALID_CHAN	*/* 存在しないチャネルの場合 */*

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

R_ETHER_CheckLink_ZC 関数はイーサネットのリンク状態を知るために PHY 管理インタフェースを使用します。この情報は PHY-LSI の Basic Status レジスタ（レジスタ 1）から読み出されます。リンク状態がリンクアップのときには ETHER_SUCCESS が返され、リンク状態がリンクダウンのときには ETHER_ERR_OTHER が返されます。

Reentrant

- 異なるチャネルからリエントラントは可能です。

Example

```
#include "r_ether_rza2_if.h"

ether_return   ret;

/* Ethernet channel number
 * ETHER_CHANNEL_0 = Ethernet channel number is 0
 * ETHER_CHANNEL_1 = Ethernet channel number is 1
 */
uint32_t        channel;

channel = ETHER_CHANNEL_0;

ret = R_ETHER_CheckLink_ZC(channel);
if(ETHER_SUCCESS == ret)
{
  /* Link is up */
  LED1 = LED_ON;
}
else
{
  /* Link is down */
  LED1 = LED_OFF;
}
```

Special Notes:

なし

3.9　R_ETHER_LinkProcess()

R_ETHER_LinkProcess 関数はリンク信号変化割り込み処理およびマジックパケット検出割り込み処理を行います。

Format

```
void    R_ETHER_LinkProcess(
            uint32_t        channel        /* ETHERC のチャネル番号  */
);
```

Parameters

channel

ETHERC/EDMAC のチャネル番号（0、1）を指定します。ETHERC/EDMAC を 1 チャネルのみ搭載する製品の場合は必ずチャネル番号 0 を指定してください。

Return Values

なし

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

R_ETHER_LinkProcess 関数はリンク信号変化割り込み処理およびマジックパケット検出割り込み処理を行います。ただし ETHER_CFG_USE_LINKSTA を値 0 に設定している場合はリンク信号変化割り込み処理は発生せずに、リンク状態変化検出処理を行います。

- マジックパケット検出割り込みが発生していた場合
  - R_ETHER_Control 関数で登録したコールバック関数により、マジックパケットを検出したことを通知します。
- リンク信号変化（リンク状態がリンクアップ）割り込みが発生していた場合
  - ディスクリプタと送受信バッファの内容を削除します。
  - ETHERC および EDMAC を初期化した後、オートネゴシエーション結果から全二重／半二重、リンク速度、フロー制御に関して適切なコンフィグレーションを決定して送受信機能を有効にします。
  - EDMAC のディスクリプタを初期状態にセットアップします。
  - R_ETHER_Control 関数で登録したコールバック関数により、リンク信号変化（リンクアップ）を検出したことを通知します。
- リンク信号変化（リンク状態がリンクダウン）割り込みが発生していた場合
  - 送受信機能を無効にした後、R_ETHER_Control 関数で登録したコールバック関数により、リンク信号変化（リンクダウン）を検出したことを通知します。
- ETHER_CFG_USE_LINKSTA を値 0 に設定している場合
  - イーサネットのリンク状態を PHY-LSI の Basic Status レジスタ（レジスタ 1）を読みだして確認します。リンク状態変化を検出した場合に以下の処理を行います。
  - リンク状態変化（リンク状態がリンクアップ）の場合
    - ディスクリプタと送受信バッファの内容を削除します。
    - ETHERC および EDMAC を初期化した後、オートネゴシエーション結果から全二重／半二重、リンク速度、フロー制御に関して適切なコンフィグレーションを決定して送受信機能を有効にします。
    - EDMAC のディスクリプタを初期状態にセットアップします。
    - R_ETHER_Control 関数で登録したコールバック関数により、リンク状態変化（リンクアップ）を検出したことを通知します。
  - リンク状態変化（リンク状態がリンクダウン）の場合
    - 送受信機能を無効にした後、R_ETHER_Control 関数で登録したコールバック関数により、リンク状態変化（リンクダウン）を検出したことを通知します。

Reentrant

- 異なるチャネルからリエントラントは可能です。

Example

```
#include "r_ether_rza2_if.h"

/* Ethernet channel number
 * ETHER_CHANNEL_0 = Ethernet channel number is 0
 * ETHER_CHANNEL_1 = Ethernet channel number is 1
 */
uint32_t        channel;

channel = ETHER_CHANNEL_0;

while(1)
{
  /* Perform link signal change interrupt processing and
   * Magic Packet detection interrupt processing
   */
  R_ETHER_LinkProcess(channel);
}
```

Special Notes:

- ETHER_CFG_USE_LINKSTA を値 1 に設定している場合は、本関数は通常処理ルーチンで定期的に呼び出してください。本関数がコールされない場合、送受信およびマジックパケット検出モードへの変更が正常に動作致しませんのでご注意ください。
- ETHER_CFG_USE_LINKSTA を値 0 に設定している場合は、本関数は必ず通常処理ルーチンで定期的に呼び出すか、定期的に発生する割り込み要因で処理される割り込み関数から呼び出してください。本関数がコールされない場合、送受信およびマジックパケット検出モードへの変更が正常に動作致しませんのでご注意ください。
- R_ETHER_Control 関数を用いて、コールバック関数を登録していない場合は、コールバック関数による通知はありません。

3.10　R_ETHER_WakeOnLAN()

R_ETHER_WakeOnLAN 関数は ETHERC の設定を通常の送受信動作からマジックパケット検出動作に切り替えます。

Format

```
ether_return_t    R_ETHER_WakeOnLAN(
        uint32_t       channel          /* ETHERC のチャネル番号 */
);
```

Parameters

channel

ETHERC/EDMAC のチャネル番号（0、1）を指定します。ETHERC/EDMAC を 1 チャネルのみ搭載する製品の場合は必ずチャネル番号 0 を指定してください。

Return Values

ETHER_SUCCESS	*/* 問題なく処理が完了した場合 */*
ETHER_ERR_INVALID_CHAN	*/* 存在しないチャネルの場合 */*
ETHER_ERR_LINK	*/* オートネゴシエーション処理が完了しておらず受信が */*
	/ 許可されていない場合 */*
ETHER_ERR_OTHER	*/* リンク状態がリンクダウンでマジックパケット検出動作に */*
	/ 切り替えた場合 */*

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

R_ETHER_WakeOnLAN 関数は ETHERC と EDMAC を初期化した後、ETHERC の設定をマジックパケット検出動作に切り替えます。

戻り値は ETHERC がマジックパケット検出動作への切り替えが成功したか否かを示しています。呼び出し時に、オートネゴシエーション処理が完了しておらず送受信が許可されていないときには値 ETHER_ERR_LINK が返されます。設定をマジックパケット検出動作に切り替えた後、リンク状態がリンクダウンとなっていたときには値 ETHER_ERR_OTHER が返されます。

Reentrant

- 異なるチャネルからリエントラントは可能です。

Example

```
#include "r_ether_rza2_if.h"

ether_return   ret;

/* Ethernet channel number
 * ETHER_CHANNEL_0 = Ethernet channel number is 0
 * ETHER_CHANNEL_1 = Ethernet channel number is 1
 */
uint32_t        channel;

channel = ETHER_CHANNEL_0;

while(1)
{
  /* Perform link signal change interrupt processing and
   * Magic Packet detection interrupt processing
   */
  R_ETHER_LinkProcess(channel);

  /* Enter Magic Packet detection mode. */
  ret = R_ETHER_WakeOnLAN(channel);
  if(ETHER_SUCCESS == ret)
  {
    /*
     * Set the MCU in sleep mode as low power consumption mode when the MCU is
     * awaiting a Magic Packet detection.
    */
    R_LPM_DStandbyTransition( &LPM_SC_TABLE[config_no]);

    wait();
  }
}
```

Special Notes:

なし

3.11　R_ETHER_CheckWrite()

データ送信が完了したことを確認する関数です。

Format

```
ether_return_t    R_ETHER_CheckWrite(
        uint32_t       channel          /* ETHERC のチャネル番号 */
);
```

Parameters

channel

ETHERC/EDMAC のチャネル番号（0、1）を指定します。ETHERC/EDMAC を 1 チャネルのみ搭載する製品の場合は必ずチャネル番号 0 を指定してください。

Return Values

ETHER_SUCCESS	*/* 問題なく処理が完了した場合 */*
ETHER_ERR_INVALID_CHAN	*/* 存在しないチャネルの場合 */*

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

R_ETHER_CheckWrite 関数は、データが送信されたことを確認します。

送信が完了した場合には、戻り値 ETHER_SUCCESS を返します。

Reentrant

- 異なるチャネルからリエントラントは可能です。

Example

- サンプルコードに含まれるMACアドレスはルネサスエレクトロニクス株式会社のベンダIDから割り当てられたアドレスを使用しています。お客様が製品化する際には必ずIEEEに申請したMACアドレスを使用するようにしてください。

```
#include <string.h>
#include "r_ether_rza2_if.h"

ether_return   ret;
uint8_t      * pwrite_buffer_address;
uint8_t      * pbuf;
uint16_t       buf_size;

/* Transmit data */
static uint8_t send_data[60] =
{
    0x74,0x90,0x50,0x00,0x79,0x02,      /* Destination MAC address    */
    0x74,0x90,0x50,0x00,0x79,0x01,      /* Source MAC address         */
    0x00,0x00,                          /* The type field is not used    */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    /* Data field     */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00
};

/* Ethernet channel number
 * ETHER_CHANNEL_0 = Ethernet channel number is 0
 * ETHER_CHANNEL_1 = Ethernet channel number is 1
 */
uint32_t         channel;

channel = ETHER_CHANNEL_0;

ret = R_ETHER_Write_ZC2_GetBuf(channel, (void **)&pwrite_buffer_address,
&buf_size);
/* When transmission buffer is empty */
if(ETHER_SUCCESS == ret)
{
  /* Write the transmit data to the transmission buffer. */
  memcpy(pwrite_buffer_address, send_data, sizeof(send_data));

  R_ETHER_Write_ZC2_SetBuf(channel, sizeof(send_data));

  /* Verifying that the transmission is completed */
  ret = R_ETHER_CheckWrite(channel);
  if(ETHER_SUCCESS == ret)
  {
    /* Transmission is completed */
  }
}
```

Special Notes:

- 本関数は､R_ETHER_Write_ZC2_SetBuf 関数で送信するデータを書き込みした後､呼び出してください。
- R_ETHER_Write_ZC2_SetBuf 関数を呼び出した後、実際のデータ送信が完了するまでには数十 μsec 必要になります。そのため、データ送信後に R_ETHER_Close_ZC2 関数にて、イーサネットモジュールを終了する場合は、R_ETHER_Write_ZC2_SetBuf 関数を呼び出した後、本関数を呼び出し、データ送信が完了したことを待ってから R_ETHER_Close_ZC2 関数を呼び出してください。本関数を呼び出さずに R_ETHER_Close_ZC2 関数を呼び出した場合、データ送信が中断されるときがあります。

3.12　R_ETHER_Read()

R_ETHER_Read 関数は指定した受信バッファへデータを受信します。

Format

```
int32_t   R_ETHER_Read(
          uint32_t      channel     /* ETHERC のチャネル番号 */
          void*         pbuf        /*  受信データの保存先 */
);
```

Parameters

channel

ETHERC/EDMAC のチャネル番号（0、1）を指定します。ETHERC/EDMAC を 1 チャネルのみ搭載する製品の場合は必ずチャネル番号 0 を指定してください。

* pbuf

受信バッファの（受信データの保存先）を指定します。

最大 1514 バイトの書き込みがあります。本関数を呼び出す際には、1514 バイト確保した配列の先頭アドレスを指定してください。

Return Values

1 以上の値	*/* 受信したバイト数 */*
ETHER_NO_DATA	*/* ゼロが返されたときは、データが受信されていません */*
ETHER_ERR_INVALID_CHAN	*/* 存在しないチャネルの場合 */*
ETHER_ERR_INVALID_PTR	*/* ポインタの値が、NULL の場合 */*
ETHER_ERR_LINK	*/* オートネゴシエーション処理が完了しておらず受信が */*
	/ 許可されていない場合 */*
ETHER_ERR_MPDE	*/* マジックパケットの検出状態のため、 */*
	/ 送信と受信が許可されていない場合 */*
ETHER_ERR_MC_FRAME	*/* マルチキャストフレームフィルタ有効時に */*
	/ マルチキャストフレームを検出した場合 */*

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

指定した受信バッファに受信データを保存します。

戻り値は受信されたバイト数を示しています。呼び出し時に、データが存在しないときには値 ETHER_NO_DATA が返されます。オートネゴシエーション処理が完了しておらず受信が許可されていないときには値 ETHER_ERR_LINK が返されます。マジックパケット検出状態となっているときには値 ETHER_ERR_MPDE が返されます。

R_ETHER_Control 関数で指定チャネルのマルチキャストフレームフィルタを有効にしている場合、マルチキャストフレームを検出すると直ちにバッファを開放します。また値 ETHER_ERR_MC_FRAME が返されます。

受信 FIFO オーバフロー、端数ビットフレーム受信エラー、ロングフレーム受信エラー、ショートフレーム受信エラー、PHY-LSI 受信エラー、受信フレーム CRC エラーが発生したフレームは受信フレームエラーとなります。受信フレームエラーが発生したディスクリプタのデータは破棄され、ステータスをクリアして読み込みを継続します。

Reentrant

- 異なるチャネルからリエントラントは可能です。

Example

```
#include "r_ether_rza2_if.h"
#include "r_ether_rza2_config.h"

ether_return   ret;
uint8_t        read_buffer[ETHER_BUFSIZE];

/* Ethernet channel number
 * ETHER_CHANNEL_0 = Ethernet channel number is 0
 * ETHER_CHANNEL_1 = Ethernet channel number is 1
 */
uint32_t         channel;

channel = ETHER_CHANNEL_0;

ret = R_ETHER_Read(channel, (void *)read_buffer);
if(ETHER_NO_DATA < ret)
{
  /* Reading the receive data is completed */
}
```

Special Notes:

- 本関数は内部で R_ETHER_Read_ZC2 関数および、R_ETHER_Read_ZC2_BufRelease 関数を呼び出しております。このため、EDMAC の受信ディスクリプタが指しているバッファと R_ETHER_Read 関数経由で指定した受信バッファの間でデータのコピーが行われます。(最大 1514 バイトの書き込みがありますので、指定する受信バッファは 1514 バイト確保してください。)
- R_ETHER_Read 関数を使用する場合は R_ETHER_Read_ZC2 関数および、R_ETHER_Read_ZC2_BufRelease 関数は使わないようにお願いいたします。
- 本関数では、標準関数 memcpy を使用するため、string.h をインクルードしています。
- 本関数を呼び出して値 ETHER_ERR_LINK が返却された場合は、イーサネットドライバを初期化してください。

3.13　R_ETHER_Write()

R_ETHER_Write 関数は指定した送信バッファからデータを送信します。

Format

```
ether_return_t   R_ETHER_Write(
        uint32_t          channel     /* ETHERC のチャネル番号 */
        void*             pbuf        /* 送信バッファポインタ*/
        const uint32_t    len         /* イーサネットフレーム長から CRC の 4 バイトを */
                                      /* 除いたサイズ（60～1514） */
);
```

Parameters

channel

ETHERC/EDMAC のチャネル番号（0、1）を指定します。ETHERC/EDMAC を 1 チャネルのみ搭載する製品の場合は必ずチャネル番号 0 を指定してください。

* pbuf

送信バッファ（送信データの書き込み先）を指定します。

len

イーサネットフレーム長から CRC の 4 バイトを除いたサイズ（60～1514）を指定します。

Return Values

ETHER_SUCCESS	*/* 問題なく処理が完了した場合 */*
ETHER_ERR_INVALID_CHAN	*/* 存在しないチャネルの場合 */*
ETHER_ERR_INVALID_DATA	*/* 引数のとり得る値が、範囲外の場合 */*
ETHER_ERR_INVALID_PTR	*/* ポインタの値が、NULL の場合 */*
ETHER_ERR_LINK	*/* オートネゴシエーション処理が完了しておらず受信が */*
	/ 許可されていない場合 */*
ETHER_ERR_MPDE	*/* マジックパケットの検出状態のため、 */*
	/ 送信と受信が許可されていない場合 */*
ETHER_ERR_TACT	*/* 送信バッファに空きがない場合 */*

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

指定した送信バッファからデータを送信します。

バッファ長に指定する値は、イーサネットフレームの最小値 64 バイトから CRC の 4 バイトを除いた 60 バイト以上かつイーサネットフレームの最大値 1518 バイトから CRC の 4 バイトを除いた 1514 バイト以下までの範囲としてください。

60 バイト未満のデータを送信する場合は、データを 0 パディングで埋めて 60 バイトとなるようにしてください。

戻り値は送信バッファに書き込んだデータの送信許可状態を示しています。呼び出し時に、送信バッファのデータの送信が許可されたときには ETHER_SUCCESS が返されます。オートネゴシエーション処理が完了しておらず送信が許可されていないときには値 ETHER_ERR_LINK が返されます。マジックパケット検出状態となっているときには値 ETHER_ERR_MPDE が返されます。送信バッファに空きがないときには値 ETHER_ERR_TACT が返されます。

Reentrant

- 異なるチャネルからリエントラントは可能です。

Example

- サンプルコードに含まれるMACアドレスはルネサスエレクトロニクス株式会社のベンダIDから割り当てられたアドレスを使用しています。お客様が製品化する際には必ず IEEE に申請した MAC アドレスを使用するようにしてください。

```
#include "r_ether_rza2_if.h"

ether_return   ret;

/* Transmit data */
static uint8_t send_data[60] =
{
    0x74,0x90,0x50,0x00,0x79,0x02,      /* Destination MAC address    */
    0x74,0x90,0x50,0x00,0x79,0x01,      /* Source MAC address         */
    0x00,0x00,                          /* The type field is not used    */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    /* Data field     */
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00
};

/* Ethernet channel number
 * ETHER_CHANNEL_0 = Ethernet channel number is 0
 * ETHER_CHANNEL_1 = Ethernet channel number is 1
 */
uint32_t        channel;

channel = ETHER_CHANNEL_0;

ret = R_ETHER_Write(channel, (void *)send_data, sizeof(send_data));
if (ETHER_SUCCESS == ret)
{
  /* Transmission is completed */
}
```

Special Notes:

- 60 バイト未満のデータを送信する場合は、データを 0 パディングで埋めて 60 バイトとなるようにしてください。
- 本関数は内部で R_ETHER_Write_ZC2_GetBuf 関数および、R_ETHER_Write_ZC2_SetBuf 関数を呼び出しております。このため、EDMAC の送信ディスクリプタが指しているバッファと R_ETHER_Write 関数経由で指定した送信バッファの間でデータのコピーが行われます。
- R_ETHER_Write 関数を使用する場合は R_ETHER_Write_ZC2_GetBuf 関数および、R_ETHER_Write_ZC2_SetBuf 関数は使わないようにお願いいたします。
- 本関数では、標準関数 memset、memcpy を使用するため、string.h をインクルードしています。
- 本関数を呼び出して値 ETHER_ERR_LINK が返却された場合は、イーサネットドライバを初期化してください。

3.14　R_ETHER_Control()

コントロールコードに対応した処理を行う関数です。

Format

```
ether_return_t R_ETHER_Control(
        ether_cmd_t const       cmd     /* コントロールコード */
        ether_param_t const     control /* コートロールコードに応じたパラメータ */
);
```

Parameters

cmd
コントロールコードを指定します。

control
コントロールコードに応じたパラメータを指定します。

Return Values

ETHER_SUCCESS	*/* 問題なく処理が完了した場合 */*
ETHER_ERR_INVALID_CHAN	*/* 存在しないチャネルの場合 */*
ETHER_ERR_CHAN_OPEN	*/* 他のアプリケーションが使用しているため */*
	/ Ether を Open できない場合 */*
ETHER_ERR_INVALID_ARG	*/* 不正な引数の場合 */*
ETHER_ERR_RECV_ENABLE	*/* ETHERC の受信機能が有効の場合 */*

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

コントロールコードに対応した処理を行います。対応していないコントロールコードの場合、戻り値 ETHER_ERR_INVALID_ARG を返します。

以下に、対応するコントロールコードを示します。

コントロールコード	概要
CONTROL_SET_CALLBACK	リンク信号変化割り込みがあったとき、もしくはマジックパケット検出割り込みがあったときにコールバックされる関数を登録します。 第 2 引数で指定した関数を登録します。
CONTROL_SET_PROMISCUOUS_MODE	ETHERC モードレジスタ（ECMR）のプロミスキャスモードビット（PRM）を設定します。 第 2 引数には、PRM を設定する側の ETHERC のチャネル番号および、PRM の値を格納している変数のアドレスを設定します。
CONTROL_SET_INT_HANDLER	EINT0/1 ステータス割り込みがあったときにコールバックされる関数を登録します。 第 2 引数で指定した関数を登録します。
CONTROL_POWER_ON	ETHERC/EDMAC のモジュールストップを解除します。 第 2 引数にモジュールストップを解除する ETHERC のチャネルを指定します。
CONTROL_POWER_OFF	ETHERC/EDMAC のモジュールストップに遷移させます。 第 2 引数にモジュールストップに遷移させる ETHERC のチャネルを指定します。
CONTROL_MULTICASTFRAME_FILTER	ディスクリプタの情報を読み込んでマルチキャストフレームを検出してフレームを破棄する機能（マルチキャストフレームフィルタ）を設定します。 第 2 引数にマルチキャストフレームフィルタ機能の設定値を指定してください。
CONTROL_BROADCASTFRAME_FILTER	ETHERC が連続で受信できるブロードキャストフレーム数を設定します。設定値以上のブロードキャストフレームを ETHERC が受信した場合はそれ以降のブロードキャストフレームは破棄されます。 第 2 引数に使用する ETHERC のチャネル番号および、ETHERC が連続で受信可能なブロードキャストフレーム数を指定してください。ブロードキャストフレーム数に 0 が指定された場合に本設定は無効になります。

Reentrant

- 異なるチャネルからリエントラントは可能です。

Example

コールバック関数を登録する場合）

```
void callback(void*);

ether_return_t   ret;
ether_param_t   param;
ether_cb_t     cb_func;

cb_func.pcb_func   = &callback;
param.ether_callback = cb_func;

ret = R_ETHER_Control(CONTROL_SET_CALBACK, param);
```

プロミスキャスモードモードを設定する場合）

```
ether_return        ret;
ether_param_t       param;
ether_promiscuous_t  promiscuous;

promiscuous.channel      = ETHER_CHANNEL_0;
promiscuous.bit          = ETHER_PROMISCUOUS_ON;
param.p_ether_promiscuous = &promiscuous;

ret = R_ETHER_Control(CONTROL_SET_PROMISCUOUS_MODE, param);
```

割り込みハンドラ関数を登録する場合)

```
void int_handler(void*);

ether_return_t  ret;
ether_param_t   param;
ether_cb_t     cb_func;

cb_func. pcb_int_hnd = &int_handler;
param.ether_callback = cb_func;

ret = R_ETHER_Control(CONTROL_SET_INT_HANDLER, param);
```

割り込みハンドラ関数)

```
static uint32_t   status_ecsr[2];
static uint32_t   status_eesr[2];

void int_handler(void * p_param)
{
  ether_cb_arg_t  *p_arg;

  p_arg = (ether_cb_arg_t *)p_param;

  if (ETHER_CANNEL_MAX > p_arg->channel)
  {
    status_ecsr[p_arg->channel] = p_arg->status_ecsr;
    status_eesr[p_arg->channel] = p_arg->status_eesr;
  }
}
```

```
ETHERC/EDMAC モジュールストップの解除)
ether_return_t  ret;
ether_param_t   param;

param.channel = channel;
ret = R_ETHER_Control(CONTROL_POWER_ON, param);

ETHERC/EDMAC モジュールストップへの遷移)
ether_return_t  ret;
ether_param_t   param;

param.channel = channel;
ret = R_ETHER_Control(CONTROL_POWER_OFF, param);

マルチキャストフレームフィルタの有効/無効設定)
ether_return_t    ret;
ether_param_t     param;
ether_multicast_t   multicast;

multicast.channel       = channel;
multicast.flag          = ETHER_MC_FILTER_ON;
param.p_ether_multicast = &multicast;

ret = R_ETHER_Control(CONTROL_MULTICASTFRAME_FILTER, param);

ブロードキャストフレームフィルタの連続受信回数の設定)
ether_return_t    ret;
ether_param_t     param;
ether_broadcast_t   broadcast;

broadcast.channel       = channel;
broadcast.counter       = 10;
param.p_ether_broadcast = &broadcast;

ret = R_ETHER_Control(CONTROL_BROADCASTFRAME_FILTER, param);
```

Special Notes:

コールバック関数の登録や割り込みハンドラ関数の登録は、R_ETHER_Open_ZC2 関数を呼び出す前に登録してください。R_ETHER_Open_ZC2 関数を呼び出してから登録した場合は、最初の割り込みを検出できない場合があります。

プロミスキャスモードを設定する場合、コントロールコードに CONTROL_POWER_ON を設定し、本関数を呼び出してから、設定してください。コントロールコードに CONTROL_POWER_ON を設定し、本関数を呼び出しせず、プロミスキャスモードを設定した場合は、意図した値が ETHERC モードレジスタに設定されません。

マルチキャストフレームフィルタおよびブロードキャストフレームフィルタは ETHERC の受信機能が有効のときは設定できません。設定する場合は R_ETHER_LinkProcess 関数を呼び出す前に設定してください。R_ETHER_LinkProcess 関数を呼び出してイーサネットドライバがリンクアップ状態になると受信機能が有効になるため、コントロールコードに CONTROL_MULTICASTFRAME_FILTER および CONTROL_BROADCASTFRAME_FILTER を設定して本関数を呼び出しても、設定されずに値「ETHER_ERR_RECV_ENABLE」が返却されます。

3.15　R_ETHER_GetVersion()

API のバージョンを返す関数です。

Format

uint32_t R_ETHER_GetVersion(void);

Parameters

なし

Return Values

バージョン番号

Properties

r_ether_rza2_if.h にプロトタイプ宣言されています。

Description

本 API のバージョン番号を返します。

Reentrant

- 異なるチャネルからリエントラントは可能です。

Example

```
#include "r_ether_rza2_if.h"

uint32_t   version;

version = R_ETHER_GetVersion();
```

Special Notes:

この関数は“#pragma inline”を使用してインライン化されています。

4.　端子設定

イーサネットドライバを使用するためには、周辺機能の入出力信号を端子に割り付ける（以下、端子設定と称す）必要があります。端子設定は、R_ETHER_Open_ZC2 関数を呼び出す前に行ってください。

e² studio の場合は「Smart Configurator」の端子設定機能を使用することができます。Smart Configurator の端子設定機能を使用すると、端子設定画面で選択したオプションに応じて、ソースファイルが出力されます。そのソースファイルで定義された関数を呼び出すことにより端子を設定できます。

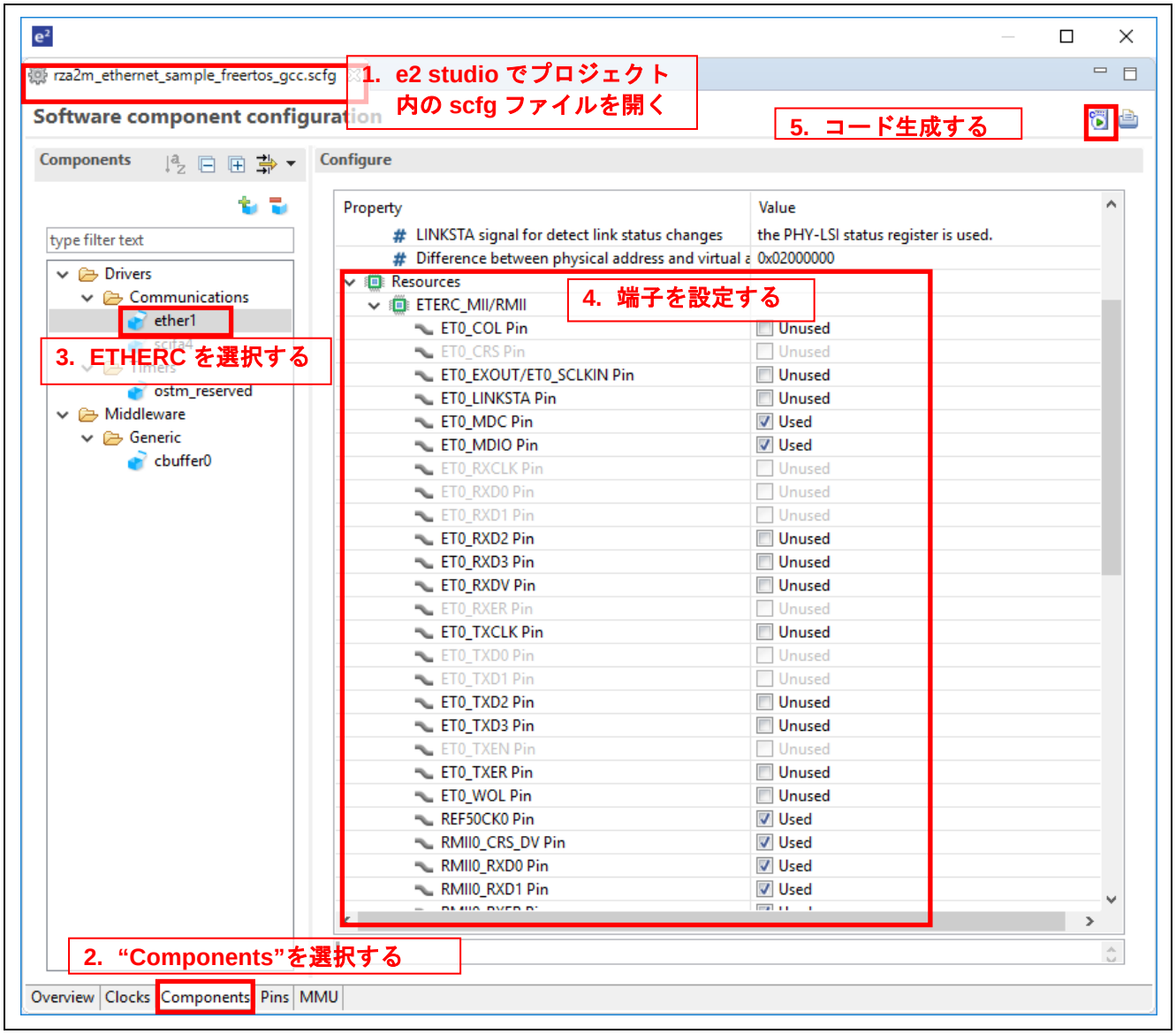

図 4.1　Smart Configurator による端子設定

5.　使用方法

5.1　セクション配置

表5.1にイーサネットドライバのセクション配置例を示します。

表5.1　プログラムのセクション配置例

種別	セクション	説明
キャッシュ RAM	.data	初期値付き変数
	.bss	初期値なし変数
非キャッシュ RAM	UNCACHED_BSS	ディスクリプタ格納領域
	_ETHERNET_BUFFERS	通信バッファ格納領域
ROM	.text	プログラムコード領域
	.rodata	定数領域

5.1.1　セクション配置の注意点

- キャッシュ RAM 領域の先頭アドレス(0x80000000 を想定)と非キャッシュ RAM の先頭アドレス(0x82000000 を想定)の差を r_ether¥src¥r_ether_rza2.c の MMU_UNCAHCED_DIFF(初期値 0x02000000) マクロに設定してください。

5.2　イーサネットドライバの初期設定方法

図 5.1にイーサネットドライバの初期設定方法のフローチャートを示します。

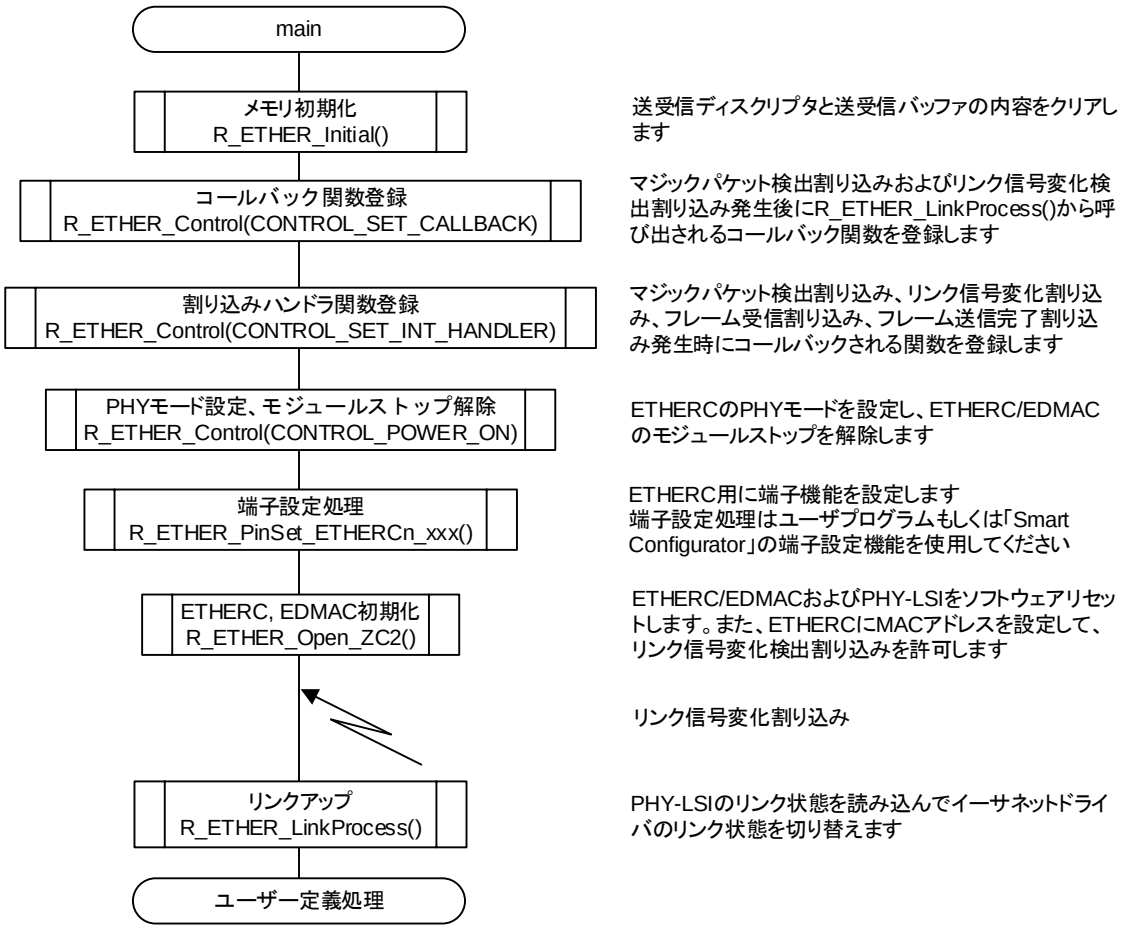

図 5.1　イーサネットドライバの初期設定方法のフローチャート

5.2.1　イーサネットドライバの初期設定方法の注意点

- R_ETHER_Initial 関数を呼び出すことで、全てのチャネルのメモリの内容がクリアされます。

5.3　マジックパケット検出動作

図 5.2にマジックパケット検出動作モードに遷移後、マジックパケットを検出して ETHERC,EDMAC を初期化するまでのフローチャートを示します。

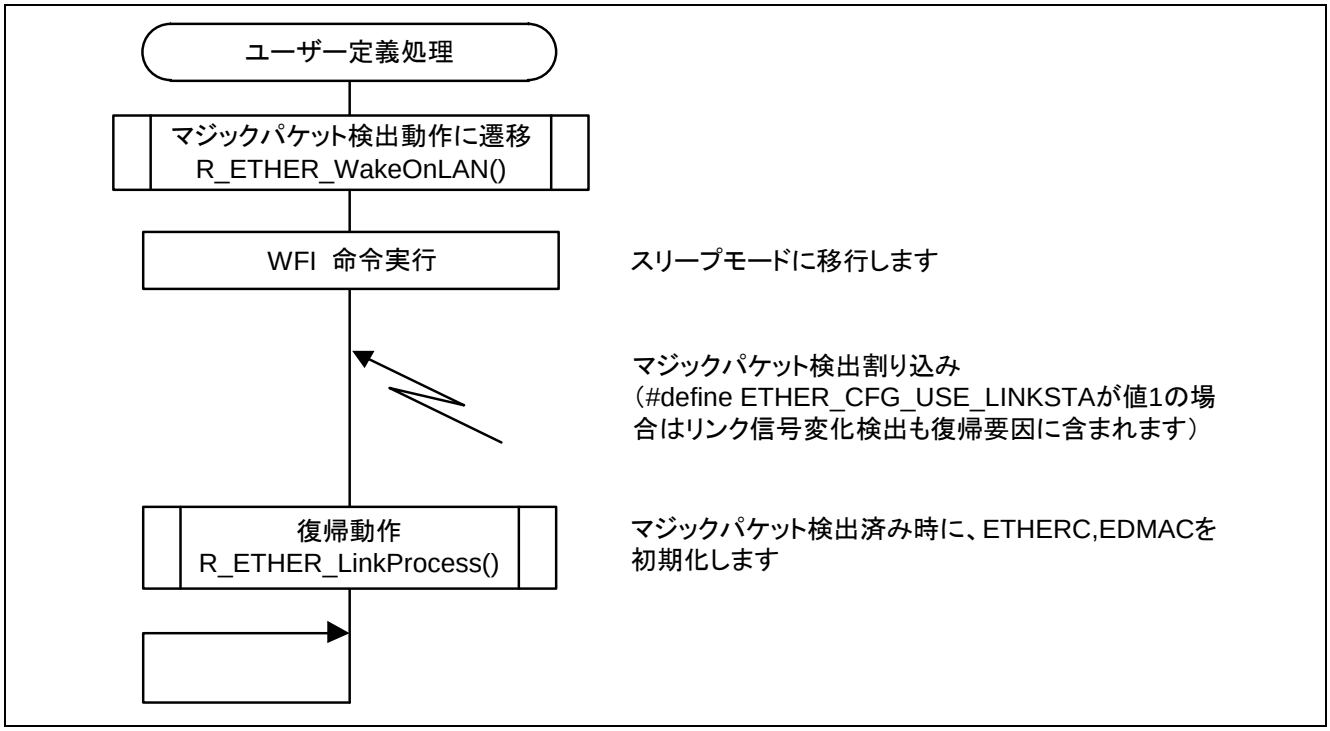

図 5.2　マジックパケット検出動作のフローチャート

5.3.1　マジックパケット検出動作の注意点

- マジックパケット検出動作に切り替えた後に ETHERC、EDMAC をモジュールストップ状態に遷移させないでください。ETHERC がマジックパケットを検出できなくなるため WFI 命令後に CPU がスリープモードから復帰できなくなる場合があります。
- マジックパケットを検出したときには、それ以前に受信していたブロードキャストフレーム等によって受信 FIFO にはデータが蓄積され、ETHERC には受信ステータスなどが報告されています。そのため R_ETHER_LinkProcess 関数を呼び出して ETHERC、EDMAC を初期化します。
- #define ETHER_CFG_USE_LINKSTA を値 1 に設定している場合は、リンク信号の変化検出時に割り込みハンドラ関数の呼び出しが発生します。そのためリンク信号の変化検出時に CPU がスリープモードに遷移していた場合は、マジックパケット検出の有無に関係なく CPU は通常動作に復帰します。

6. 動作確認環境

本アプリケーションノートのサンプルコードは、下記の条件で動作を確認しています。

表6.1　動作確認条件

項目	内容
使用 MCU	RZ/A2M
動作周波数（注）	CPU クロック（Iφ）：528MHz 画像処理クロック（Gφ）：264MHz 内部バスクロック（Bφ）：132MHz 周辺クロック 1（P1φ）：66MHz 周辺クロック 0（P0φ）：33MHz QSPI0_SPCLK：66MHz CKIO：132MHz
動作電圧	電源電圧（I/O）：3.3V 電源電圧（1.8/3.3V 切替 I/O（PVcc_SPI））：3.3V 電源電圧（内部）：1.2V
統合開発環境	e2 studio V7.4.0
C コンパイラ	GNU Arm Embedded Toolchain 6-2017-q2-update
	コンパイラオプション（ディレクトリパスの追加は除く） Release: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Os -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -Wstack-usage=100 -fabi-version=0 Hardware Debug: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Og -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -g3 -Wstack-usage=100 -fabi-version=0
動作モード	ブートモード 3 （シリアルフラッシュブート 3.3V 品）
ターミナルソフトの通信設定	• 通信速度：115200bps • データ長：8 ビット • パリティ：なし • ストップビット長：1 ビット • フロー制御：なし
使用ボード	RZ/A2M CPU ボード(RTK7921053C00000BE) RZ/A2M SUB ボード(RTK79210XXB00000BE)
使用デバイス （ボード上で使用する機能）	• シリアルフラッシュメモリ（SPI マルチ I/O バス空間に接続） メーカ名：Macronix 社、型名：MX25L51245GXD • RL78/G1C（USB 通信とシリアル通信を変換し、ホスト PC との通信に使用）

【注】 クロックモード 1（EXTAL 端子からの 24MHz のクロック入力）で使用時の動作周波数です。

7. ドライバのインポート方法

7.1 e2 studio

Smart Configurator ツールを使用して e2 studio のプロジェクトにドライバをインポートする方法の詳細については、RZ/A2M Smart Configurator ユーザーガイド：e2 studio R20AN0583JJ を参照してください。

7.2 e2 studio 以外で作成されたプロジェクトの場合

このセクションでは、ドライバをプロジェクトにインポートする方法について説明します。

一般的に、どの IDE にも 2 つのステップがあります。

1) プロジェクトに必要なソースツリー内の場所にドライバをコピーします。

2) ドライバをコピーした場所へのリンクをコンパイラに追加します。

他に必要なドライバがある場合（例えば r_cbuffer など）、同様にインポートする必要があります。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2018.12.28	-	初版
1.01	2019.4.15	7	カスタマイズ方法をヘッダファイルの編集から、Smart Configurator に変更
		55	e^2 studio のバージョンを 7.4.0 に変更
		-	Smart Configurator によるカスタマイズに対応
1.10	2019.5.17	55	表6.1　動作確認条件 コンパイラオプション"-mthumb-interwork"を削除
		56	「7.ドライバのインポート方法」追加

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 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・システムの設計において、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て生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 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。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   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、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合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。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、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、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等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、お客様の機器・システムとしての出荷保証を行ってください。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、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の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。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。
9. 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・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
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禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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直接的、間接的に支配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。

(Rev.4.0-1　2017.11)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。