RX ファミリ

イーサネットモジュール Firmware Integration Technology

要旨

本アプリケーションノートは、Firmware Integration Technology（FIT）を使用したイーサネットモジュールについて説明します。本モジュール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 FIT モジュールと称します。

Rev1.11 からイーサネット FIT モジュール内の端子設定処理を削除しました。イーサネット FIT モジュール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。

- RX64M
- RX71M
- RX65N
- RX72M
- RX72N
- RX66N

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

対象コンパイラ

- Renesas Electronics C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX

各コンパイラの動作確認内容については「6.2 動作確認環境」を参照してください。

関連ドキュメント

- ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)

目次

1. 概要

イーサネット FIT モジュールは、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）と PHY マネージメントインターフェース（以降、PMGI と呼称）*1 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 FIT モジュール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。
-

【注】*1：ETHER_CFG_NON_BLOCKING を値 1 に設定している場合、PMGI を使用します。

1.1 イーサネット FIT モジュールとは

イーサネット FIT モジュールは 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 FIT モジュールの組み込み方については、「2.12　FIT モジュールの追加方法」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 FIT モジュールに含まれる 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_WritePHY()	PHY 管理インタフェースを使用して、PHY-LSI 内のレジスタにライトアクセスします。
R_ETHER_ReadPHY()	PHY 管理インタフェースを使用して、PHY-LSI 内のレジスタにリードアクセスします。
R_ETHER_GetVersion()	イーサネット FIT モジュールのバージョン番号を返します。

1.3　制限事項

イーサネット FIT モジュールには以下の制限事項があります。

- RX64M、RX71M、RX72M、RX72N、RX66N でイーサネット FIT モジュールを使用する場合は、0000 0000h ～0000 001Fh 番地を使用できません。

2. API 情報

イーサネット FIT モジュールの API はルネサスの API の命名基準に従っています。

2.1 ハードウェアの要求

ご使用になる MCU が以下の機能をサポートしている必要があります。

- ETHERC
- EDMAC
- PMGI[*1]

【注】*1：ETHER_CFG_NON_BLOCKING を値 1 に設定している場合、PMGI を使用します。

2.2 ソフトウェアの要求

イーサネット FIT モジュールは以下のパッケージに依存しています。

- Renesas Board Support Package (r_bsp) Rev.5.20 以上

2.3 サポートされているツールチェーン

イーサネット FIT モジュールは、「6.2 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。

2.4　使用する割り込みベクタ

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 EINT 割り込み、EINT0 割り込み、EINT1 割り込みが有効になります。表 2.1にイーサネット FIT モジュールが使用する割り込みベクタを示します。

表 2.1　使用する割り込みベクター覧

デバイス	割り込みベクタ
RX64M RX71M RX72M RX72N	GROUPAL1 割り込み（ベクタ番号: 113） ● EINT0 割り込み[チャネル 0]（グループ割り込み要因番号：4） ● EINT1 割り込み[チャネル 1]（グループ割り込み要因番号：5）
RX65N RX66N	GROUPAL1 割り込み（ベクタ番号: 113） ● EINT0 割り込み[チャネル 0]（グループ割り込み要因番号：4）
RX72M RX72N	● PMGI0I 割り込み[チャネル 0]（ベクタ番号：252[*1]） ● PMGI1I 割り込み[チャネル 1]（ベクタ番号：253[*1]）
RX66N	● PMGI0I 割り込み[チャネル 0]（ベクタ番号：252[*1]）

【注】*1：選択型割り込み A に割り当てられている割り込みのベクタ番号については、ボードサポートパッケージ FIT モジュール(BSP モジュール)で割り当てられているデフォルト設定を記載しています。

2.5　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_rx_if.h に記載しています。

2.6　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。

2.7　コンパイル時の設定

イーサネット FIT モジュールのコンフィギュレーションオプションの設定は、r_ether_rx_config.h で行います。Smart Configurator を使用する場合は、ソフトウェアコンポーネント設定画面でコンフィグレーションオプションを設定できます。設定値はモジュールを追加する際に、自動的に r_ether_rx_config.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 in r_ether _rx_config.h	
#define ETHER_CFG_MODE_SEL 【注】デフォルト値は “0”	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。 “0” の場合、MII(Media Independent Interface)を選択します。 “1” の場合、RMII（Reduced Media Independent Interface）を選択します。
#define ETHER_CFG_CH0_PHY_ADDRESS 【注】デフォルト値は “0” *6	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。
#define ETHER_CFG_CH1_PHY_ADDRESS 【注】デフォルト値は “1” *8	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。
#define ETHER_CFG_EMAC_RX_DESCRIPTORS 【注】デフォルト値は “1”	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 “1”	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。
#define ETHER_CFG_AL1_INT_PRIORTY 【注】デフォルト値は “2”	グループ AL1 割り込みの優先レベルを設定してください。 “1” ～ “15” の範囲で設定してください。*4
#define ETHER_CFG_CH0_PHY_ACCESS 【注】デフォルト値は “1” *1*7	ETHERC チャネル 0 が使用する PHY のアクセスチャネルを設定してください。 “0” の場合、PHY のレジスタアクセスは ETHERC0 を使用します。*2 “1” の場合、PHY のレジスタアクセスは ETHERC1 を使用します。*3
#define ETHER_CFG_CH1_PHY_ACCESS 【注】デフォルト値は “1” *1*7	ETHERC チャネル 1 が使用する PHY のアクセスチャネルを設定してください。 “0” の場合、PHY のレジスタアクセスは ETHERC0 を使用します。*2 “1” の場合、PHY のレジスタアクセスは ETHERC1 を使用します。*3
#define ETHER_CFG_PHY_MII_WAIT 【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 ETHER_CFG_PHY_DELAY_RESET 【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。
#define ETHER_CFG_LINK_PRESENT 【注】デフォルト値は “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。 “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。 “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。
#define ETHER_CFG_USE_LINKSTA 【注】デフォルト値は “1”	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。*5 “0” の場合、PHY-LSI のステータスレジスタを使用します。 “1” の場合、LINKSTA 信号を使用します。
#define ETHER_CFG_USE_PHY_KSZ8041NL 【注】デフォルト値は “0”	Micrel 社の PHY-LSI KSZ8041NL を使用するかしないかを設定してください。 “0” の場合、KSZ8041 を使用しません。 “1” の場合、KSZ8041 を使用します。
#define ETHER_CFG_NON_BLOCKING 【注】デフォルト値は “0”*10	一部の API 関数の呼び出しをブロッキングコールにするかノンブロッキングコールにするかを設定してください。 “0” の場合、ブロッキングです。 “1” の場合、ノンブロッキングです。
#define ETHER_CFG_PMGI_CLOCK 【注】デフォルト値は “2500000”*9	PHY マネジメントステーションのクロックを設定してください。 “97657”～”60000000”の範囲で設定してください。
#define ETHER_CFG_PMGI_ENABLE_PREAMBLE 【注】デフォルト値は “0”*9	PHY マネジメントステーションのプリアンブルフィールドの有無を設定してください。 “0” の場合、プリアンブルフィールド有です。 “1” の場合、プリアンブルフィールド無です。
#define ETHER_CFG_PMGI_HOLD_TIME 【注】デフォルト値は “0”*9	PHY マネジメントステーションの MDIO 入出力のキャプチャ時間を設定してください。 “0”～”7”の範囲で設定してください。
#define ETHER_CFG_PMGI_CAPTURE_TIME 【注】デフォルト値は “0”*9	PHY マネジメントステーションの MDIO 入出力のホールド時間を設定してください。 “0”～”7”の範囲で設定してください。
#define ETHER_CFG_PMGI_INT_PRIORTY 【注】デフォルト値は “2”*9	グループ PMGI 割り込みの優先レベルを設定してください。 “1” ～ “15” の範囲で設定してください。

【注】 *1 Renesas Starter Kit+ for RX64M（製品型名：R0K50564MSxxxBE）上でイーサネット FIT モジュールを動かす場合の設定は表 2.2を参照ください。
また Renesas Starter Kit+ for RX71M（製品型名：R0K50571MSxxxBE）上でイーサネット FIT モジュールを動かす場合の設定は表 2.3 を参照ください。

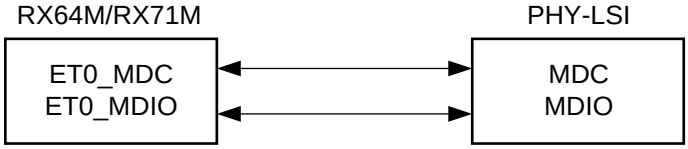
表2.2　ETHER_CFG_CH0_PHY_ACCESS／ETHER_CFG_CH1_PHY_ACCESS 設定　その 1

ショートピン J3	ショートピン J4	ETHER_CFG_CH0_PHY_ACCESS ETHER_CFG_CH1_PHY_ACCESS の設定値
1-2 間ショート	1-2 間ショート	0 0
2-3 間ショート	2-3 間ショート	1 1

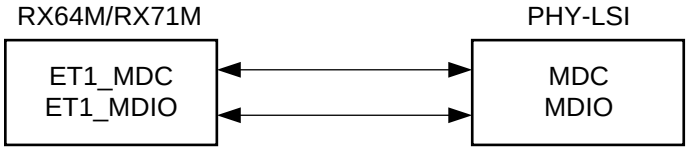
表2.3　ETHER_CFG_CH0_PHY_ACCESS／ETHER_CFG_CH1_PHY_ACCESS 設定　その 2

ショートピン J13	ショートピン J9	ETHER_CFG_CH0_PHY_ACCESS ETHER_CFG_CH1_PHY_ACCESS の設定値
1-2 間ショート	1-2 間ショート	0 0
2-3 間ショート	2-3 間ショート	1 1

*2 ETHERC と PHY-LSI の接続が下記の場合の設定です。



*3 ETHERC と PHY-LSI の接続が下記の場合の設定です。



*4 本設定はターゲット MCU が RX64M/RX71M/RX65N/RX72M/RX72N/RX66N の時のみ有効です。

*5 ターゲット MCU が RX64M/RX71M/RX72M/RX72N の場合は、全てのチャネルで本設定が有効になります。

*6 デフォルト値は Renesas Starter Kit+ for RX64M および Renesas Starter Kit+ for RX71M の初期設定に従った数値です。Renesas Starter Kit+ for RX65N（製品型名：RTK500565NSxxxxxBE）もしくは Renesas Starter Kit+ for RX65N-2MB（製品型名：RTK50565N2SxxxxxBE）を使用する場合は、値を 30 に設定してください。Renesas Starter Kit+ for RX72M（製品型名：RTK5572Mxxxxxxxxxx）を使用する場合は、値を 1 に設定してください。

*7 デフォルト値は Renesas Starter Kit+ for RX64M および Renesas Starter Kit+ for RX71M および Renesas Starter Kit+ for RX72N の初期設定に従った数値です。Renesas Starter Kit+ for RX65N、Renesas Starter Kit+ for RX65N-2MB および Renesas Starter Kit+ for RX72M を使用する場合は、値を 0 に設定してください。

*8 デフォルト値は Renesas Starter Kit+ for RX64M および Renesas Starter Kit+ for RX71M および Renesas Starter Kit+ for RX72N の初期設定に従った数値です。Renesas Starter Kit+ for RX72M（製品型名：RTK5572Mxxxxxxxxxx）を使用する場合は、値を 2 に設定してください。

*9　ETHER_CFG_NON_BLOCKING == 1 の場合のみ有効です。

*10　RX72M、RX72N、RX66N の時のみ有効です。

2.8　コードサイズ

ツールチェーン（セクション2.3記載）でのコードサイズは、最適化レベル 2、およびコードサイズ重視の最適化を前提としたサイズです。ROM (コードおよび定数) と RAM (グローバルデータ) のサイズは、本モジュールのコンフィギュレーションヘッダファイルで設定される、ビルド時のコンフィギュレーションオプションによって決まります。

下表の値は下記条件で確認しています。

モジュールリビジョン: r_ether_rx rev1.20

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00
(統合開発環境のデフォルト設定に”-lang = c99”オプションを追加)
GCC for Renesas RX 4.08.04.201902
(統合開発環境のデフォルト設定に” -std=gnu99”オプションを追加)
IAR C/C++ Compiler for Renesas RX version 4.12.1
(統合開発環境のデフォルト設定)

コンフィグレーションオプション: デフォルト設定

ROM、RAM およびスタックのコードサイズ							
デバイス	分類	使用メモリ					
		Renesas Compiler		GCC		IAR Compiler	
		ﾌﾞﾛｯｷﾝｸﾞ (注 3)	ﾉﾝﾌﾞﾛｯｷﾝｸﾞ (注 4)	ﾌﾞﾛｯｷﾝｸﾞ	ﾉﾝﾌﾞﾛｯｷﾝｸﾞ	ﾌﾞﾛｯｷﾝｸﾞ	ﾉﾝﾌﾞﾛｯｷﾝｸﾞ
RX72M (注 1)	ROM	4492 バイト	6487 バイト	10776 バイト	14852 バイト	6546 バイト	9849 バイト
	RAM	6281 バイト	6349 バイト	6444 バイト	6512 バイト	6285 バイト	6337 バイト
	スタック (注 2)	148 バイト	148 バイト	-		216 バイト	216
RX65N (注 1)	ROM	4197 バイト		7776 バイト		5927 バイト	
	RAM	3146 バイト		3144 バイト		3150 バイト	
	スタック (注 2)	148 バイト		-		168 バイト	

注 1. ETHER_CFG_EMAC_RX_DESCRIPTORS = 1, ETHER_CFG_EMAC_TX_DESCRIPTORS = 1, ETHER_CFG_BUFSIZE = 1536 の場合のサイズです。
注 2. 割り込み関数の最大使用スタックサイズを含みます。
注 3. ETHER_CFG_NON_BLOCKING = 0 の場合のサイズです。
注 4. ETHER_CFG_NON_BLOCKING = 1 の場合のサイズです。

2.9　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_rx_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 */
  CONTROL_RECEIVE_DATA_PADDING,         /* 受信データにパディング挿入 */
  CONTROL_SET_PMGI_CALLBACK             /* PMGI コールバック関数の登録 */

} 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_cb_t             pmgi_callback;         /* PMGI コールバック関数ポインタ */
   ether_recv_padding_t * padding_param;         /* 受信データパディング挿入用パラメータ */
} 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 */
  void    (*pcb_pmgi_hnd) (void *);     /* PMGI コールバック関数ポインタ */
} 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;

typedef struct
{
  uint32_t      channel;              /* ETHERC チャネル */
  uint8_t       position;             /* パディング挿入ポジション */
  uint8_t       size;                 /* パディング挿入サイズ */
} ether_recv_padding_t;

typedef enum
{
  OPEN_ZC2 = 0,                 /* R_ETHER_Open_ZC2 関数を実行中 */
  CHECKLINK_ZC,                 /* R_ETHER_CheckLink_ZC 関数を実行中 */
  LINKPROCESS,                  /* R_ETHER_LinkProcess 関数を実行中 */
  WAKEONLAN,                    /* R_ETHER_WakeOnLAN 関数を実行中 */
  LINKPROCESS_OPEN_ZC2,         /* R_ETHER_LinkProcess 関数を実行中 */
  LINKPROCESS_CHECKLINK_ZC0,    /* R_ETHER_LinkProcess 関数を実行中 */
  LINKPROCESS_CHECKLINK_ZC1,    /* R_ETHER_LinkProcess 関数を実行中 */
  LINKPROCESS_CHECKLINK_ZC2,    /* R_ETHER_LinkProcess 関数を実行中 */
  WAKEONLAN_CHECKLINK_ZC,       /* R_ETHER_WakeOnLAN 関数を実行中 */
  WRITEPHY,                     /* R_ETHER_WritePHY 関数を実行中 */
  READPHY,                      /* R_ETHER_ReadPHY 関数を実行中 */
  PMGI_MODE_NUM                 /* PMGI 動作モード数 */
} pmgi_mode_t;
```

```
typedef enum
{
  STEP0 = 0,                    /* PMGI 動作ステップ 0 */
  STEP1,                        /* PMGI 動作ステップ 1 */
  STEP2,                        /* PMGI 動作ステップ 2 */
  STEP3,                        /* PMGI 動作ステップ 3 */
  STEP4,                        /* PMGI 動作ステップ 4 */
  STEP5,                        /* PMGI 動作ステップ 5 */
  STEP6,                        /* PMGI 動作ステップ 6 */
  PMGI_STEP_NUM                 /* PMGI 動作ステップ数 */
} pmgi_step_t;

typedef enum
{
  PMGI_IDLE = 0,                    /* PMGI アイドル中 */
  PMGI_RUNNING = 1,                 /* PMGI 動作中 */
  PMGI_COMPLETE = 2,                /* PMGI 動作完了 */
  PMGI_ERROR = -1                   /* PMGI 動作エラー */
} pmgi_event_t;

typedef struct
{
  ether_return_t (* p_func)(uint32_t ether_channel);
                                    /* ファンクションポインタ配列のタイプ */
} st_pmgi_interrupt_func_t;

typedef struct
{
  bool           locked;          /* PMGI ロック状態を示すフラグ */
  pmgi_event_t   event;           /* PMGI カレントオプレション状態 */
  pmgi_mode_t    mode;            /* PMGI 動作モード */
  pmgi_step_t    step;            /* PMGI 動作ステップ */
  uint16_t       read_data;       /* PMGI レジスタから読んだ値 */
  uint32_t       reset_counter;   /* リセットレジスタ読み取りのカウント */
  uint32_t       ether_channel;   /* ETHERC チャネル番号 */
} pmgi_param_t;

typedef struct
{
  uint32_t         channel;       /* ETHERC チャネル */
  pmgi_event_t     event;         /* コールバック関数のイベントコード */
  pmgi_mode_t      mode;          /* PMGI 動作モード */
  uint16_t         reg_data;      /* 割り込みハンドラの PHY レジスタデータ */
} pmgi_cb_arg_t;
```

2.10　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_rx_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 もしくは FIT_NO_PTR 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_LOCKED,               /* ノンブロッキングモードが有効時かつ PHY アクセス中の場合 */
ETHER_ERR_OTHER                 /* その他エラー */
} ether_return_t;
```

2.11　コールバック関数

(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 FIT モジュールでは、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、「2.9　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 2.4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2.4　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)　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 FIT モジュールでは、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 FIT モジュール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 FIT モジュールが通常動作の場合
  - 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、「2.9　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。

(3)　PMGI 割り込みから呼び出すコールバック関数

イーサネット FIT モジュールでは、ノンブロッキングコールした API 関数の処理が完了の時、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 R_ETHER_Control を用いて、「2.9　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_PMGI_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、API の処理が完了したチャネル番号と表 2.5に示す定数を格納した変数、表 2.6に示す定数を格納した変数、PHY レジスタリードデータ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2.5　コールバック関数の引数一覧

定数定義	意味
OPEN_ZC2	R_ETHER_Open_ZC2 関数の処理が完了した。
CHECKLINK_ZC	R_ETHER_CheckLink_ZC 関数の処理が完了した。
LINKPROCESS	R_ETHER_LINKPROCESS 関数の処理が完了した。
WAKEONLAN	R_ETHER_WAKEONLAN 関数の処理が完了した。
WRITEPHY	R_ETHER_WritePHY 関数の処理が完了した。
READPHY	R_ETHER_ReadPHY 関数の処理が完了した。

表2.6　コールバック関数の引数一覧

定数定義	意味
PMGI_COMPLETE	API 関数の処理が正常に完了した。 R_ETHER_CheckLink_ZC 関数の場合、リンクアップを検出した。
PMGI_ERROR	API 関数の処理が異常終了した。 R_ETHER_CheckLink_ZC 関数の場合、リンクダウンを検出した。
PMGI_IDLE	API 関数処理は、PMGI 操作なしで正常に完了しました。 R_ETHER_LinkProcess 関数の場合、関数の実行中に PMGI 操作は実行されません。

2.12　FIT モジュールの追加方法

本モジュールは、使用するプロジェクトごとに追加する必要があります。ルネサスでは、Smart Configurator を使用した(1)、(3)の追加方法を推奨しています。ただし、Smart Configurator は、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)、(4)の方法を使用してください。

(1) e^2 studio 上で Smart Configurator を使用して FIT モジュールを追加する場合
e^2 studio の Smart Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「Renesas e^2 studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(2) e^2 studio 上で FIT Configurator を使用して FIT モジュールを追加する場合
e^2 studio の FIT Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加することができます。詳細は、アプリケーションノート「RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)」を参照してください。

(3) CS+上で Smart Configurator を使用して FIT モジュールを追加する場合
CS+上で、スタンドアロン版 Smart Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「Renesas e^2 studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(4) CS+上で FIT モジュールを追加する場合
CS+上で、手動でユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」を参照してください。

2.13　イーサネットフレームのフレーム形式

イーサネット FIT モジュールは、Ethernet II/IEEE802.3 のフレーム形式をサポートしています。

2.13.1　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。

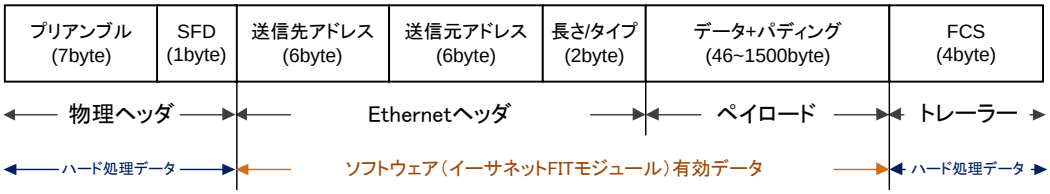


図 2.1　Ethernet II /IEEE802.3 のフレーム形式

- 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。

2.13.2　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。

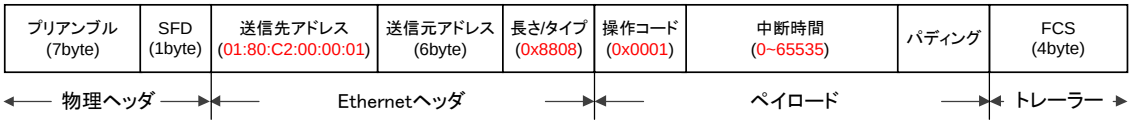


図 2.2　PAUSE フレームのフレーム形式

- 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。

2.13.3　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。

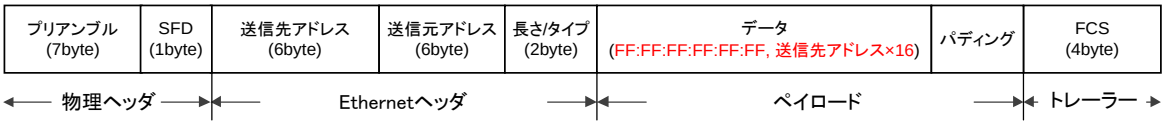


図 2.3　マジックパケットのフレーム形式

- 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。

2.14　for 文、while 文、do while 文について

本モジュールでは、レジスタの反映待ち処理等で for 文、while 文、do while 文（ループ処理）を使用しています。これらループ処理には、「WAIT_LOOP」をキーワードとしたコメントを記述しています。そのため、ループ処理にユーザがフェイルセーフの処理を組み込む場合は、「WAIT_LOOP」で該当の処理を検索できます。

以下に記述例を示します。

```
while文の例：
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}

for文の例：
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}

do while文の例：
/* Reset completion waiting */
do
{
    reg = phy_read(ether_channel, PHY_REG_CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /*
WAIT_LOOP */
```

3. API 関数

3.1 R_ETHER_Initial()

イーサネット FIT モジュール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_rx_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。

Example

```
#include "platform.h"
#include "r_ether_rx_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（Pasuse）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 Pasuse 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。
Pasuse 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またはノンブロッキング*/
                            /* モードが有効のとき、動作が正常に開始された場合  */
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 もしくはFIT_NO_PTR 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または */
                            /* ノンブロッキングモードが有効のとき、*/
                            /* PMGI コールバック関数が登録されていない場合 */
ETHER_ERR_LOCKED            /* ノンブロッキングモードが有効のとき、PHY アクセス中*/
                            /* の場合 */
```

Properties

r_ether_rx_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。
ノンブロッキングモードが有効の場合、関数の処理結果が PMGI コールバック関数の引数として渡されます。

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 "platform.h"
#include "r_ether_rx_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_rx_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。

Example

```
#include "platform.h"
#include "r_ether_rx_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 もしくはFIT_NO_PTR 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_rx_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。なお、RX64M/RX71M/RX72M/RX72N/RX66N でハードウェアによるマルチキャストフレームフィルタを有効にした場合、マルチキャストフレームはハードウェアにより破棄され検出はできなくなります。詳細は6.1節を参照ください。

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。

Example

```
#include <string.h>
#include "platform.h"
#include "r_ether_rx_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 FIT モジュール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_rx_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。

Example

```
#include <string.h>
#include "platform.h"
#include "r_ether_rx_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 FIT モジュール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 もしくは FIT_NO_PTR 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_rx_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。

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 "platform.h"
#include "r_ether_rx_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 FIT モジュール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_rx_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。

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 "platform.h"
#include "r_ether_rx_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 FIT モジュール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またはノンブロッキングモードが*/
                       /* 有効のとき、動作が正常に開始された場合   */
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 */
ETHER_ERR_LOCKED       /* ノンブロッキングモードが有効のとき、PHY アクセス中の場合 */
```

Properties

r_ether_rx_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。ノンブロッキングモードが無効の場合、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。ノンブロッキングモードが有効の場合、リンク状態のチェック完了後にチェック結果が割り込みハンドラ関数の引数として渡されます。

Example

```
#include "platform.h"
#include "r_ether_rx_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_rx_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。ノンブロッキングモードが有効の場合、関数の処理結果が PMGI コールバック関数の引数として渡されます。

- 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。

Example

```
#include "platform.h"
#include "r_ether_rx_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 */*

ETHER_ERR_LOCKED　　/ ノンブロッキングモードが有効のとき、PHY アクセス中の場合 */*

Properties

r_ether_rx_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。

ノンブロッキングコールが無効の場合、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。

ノンブロッキングモードが有効の場合、関数の処理結果が PMGI コールバック関数の引数として渡されます。

Example

```
#include "platform.h"
#include "r_ether_rx_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)
  {
    R_BSP_RegisterProtectDisable(BSP_REG_PROTECT_LPC_CGC_SWR);
    /*
     * Set the MCU in sleep mode as low power consumption mode when the MCU is
     * awaiting a Magic Packet detection.
     */
    SYSTEM.SBYCR.BIT.SSBY = 0;
    R_BSP_RegisterProtectEnable(BSP_REG_PROTECT_LPC_CGC_SWR);

    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_rx_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。

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 "platform.h"
#include "r_ether_rx_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 もしくはFIT_NO_PTR 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_rx_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。なお、RX64M/RX71M/RX72M/RX72N/RX66N でハードウェアによるマルチキャストフレームフィルタを有効にした場合、マルチキャストフレームはハードウェアにより破棄され検出はできなくなります。詳細は6.1節を参照ください。

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。

Example

```
#include "platform.h"
#include "r_ether_rx_if.h"
#include "r_ether_rx_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 FIT モジュール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 もしくは FIT_NO_PTR 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_rx_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。

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 "platform.h"
#include "r_ether_rx_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 FIT モジュール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     contorl /* 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_rx_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。
CONTROL_RECEIVE_DATA_PADDING	受信データパディング挿入レジスタ（PRADIR）のパラメータを設定します。 2 番目の引数に、ETHERC チャネル、パディング挿入位置、およびパディング挿入サイズを設定します。
CONTROL_SET_PMGI_CALLBACK	ノンブロッキングコールが有効の場合、API 関数の処理完了後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。

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_Contorl(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_Contorl(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_Contorl(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_Contorl(CONTROL_BROADCASTFRAME_FILTER, param);
```

受信データに対するパディング挿入位置とパディング挿入サイズの設定)

```
ether_return_t                      ret;
ether_param_t                       param;
ether_recv_padding_t                pad_param;

pad_param.channel             = channel;
pad_param.position            = 0x3f;
pad_param.size                = 0x3;
param.padding_param           = &pad_param;

ret = R_ETHER_Contorl(CONTROL_RECEIVE_DATA_PADDING, param);
```

PMGI コールバック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_pmgi_hnd = &int_handler;
param.ether_callback = cb_func;

ret = R_ETHER_Contorl(CONTROL_SET_PMGI_CALLBACK, param);
```

PMGI コールバック関数)

```
static pmgi_event_t   pmgi_event;
static pmgi_mode_t   pmgi_mode;
static uint16_t    phy_reg_data;

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

  p_arg = (pmgi_cb_arg_t *)p_param;

  if (ETHER_CANNEL_MAX > p_arg->channel)
  {
    pmgi_event     = p_arg->event;
    pmgi_mode      = p_arg->mode;
    pmgi_reg_data  = p_arg->reg_data;
  }
}
```

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 FIT モジュール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_WritePHY()

R_ETHER_WritePHY 関数は PHY 管理インタフェースを使用して、PHY-LSI 内のレジスタにライトアクセスします。

Format

```
ether_return_t R_ETHER_WritePHY(
        uint32_t channel        /* ETHERC のチャネル番号 */
        uint16_t address        /* アクセスする PHY-LSI のレジスタアドレス */
        uint16_t data           /* PHY-LSI のレジスタにライトするデータ */
);
```

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。

address

アクセスする PHY-LSI のレジスタのアドレスを指定します。詳細は使用する PHY-LSI のデータシートを確認してください。

data

PHY-LSI のレジスタにライトするデータを指定します。詳細は使用する PHY-LSI のデータシートを確認してください。

Return Values

ETHER_SUCCESS　　/ アクセスが正常完了した場合またはノンブロッキングモードが有効の*/ /* とき、動作が正常に開始された場合 */*

ETHER_ERR_OTHER　　/ ノンブロッキングモードが有効のとき、割り込みハンドラ関数が登録 */ /* されていない場合 */*

ETHER_ERR_INVALID_CHAN / 存在しないチャネルの場合 */*

ETHER_ERR_LOCKED　　/ ノンブロッキングモードが有効のとき、PHY アクセス中の場合 */*

Properties

r_ether_rx_if.h にプロトタイプ宣言されています。

Description

R_ETHER_WritePHY 関数は PHY 管理インタフェースを使用して、PHY-LSI 内のレジスタにライトアクセスします。 ノンブロッキングモードが無効の場合、ライトアクセスが正常に完了したら ETHER_SUCCESS が返されます。
ノンブロッキングモードが有効の場合、ライトアクセス完了後にコールバック関数が実行されます。

Example

```
#include "platform.h"
#include "r_ether_rx_if.h"
ether_return_t          ret;
uint32_t                      channel;
uint16_t                      address;
```

```
uint16_t                        data;

channel = ETHER_CHANNEL_0;
address = PHY_REG_CONTROL;
data    = PHY_CONTROL_RESET;
ret = R_ETHER_WritePHY(channel, address, data);
```

Special Notes:

なし。

3.16　R_ETHER_ReadPHY()

R_ETHER_ReadPHY 関数は PHY 管理インタフェースを使用して、PHY-LSI 内のレジスタにリードアクセスします。

Format

```
ether_return_t R_ETHER_ReadPHY(
        uint32_t channel /* ETHERC のチャネル番号 */
        uint16_t address /* アクセスする PHY-LSI のレジスタアドレス */
        uint16_t * p_data /* リードしたレジスタの値を格納する変数のポインタ */
);
```

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。

address

アクセスする PHY-LSI のレジスタのアドレスを指定します。詳細は使用する PHY-LSI のデータシートを確認してください。

*p_data

PHY-LSI から読み出したレジスタの値を格納する変数のポインタを指定します。詳細は使用する PHY-LSI のデータシートを確認してください。

Return Values

```
ETHER_SUCCESS           /* アクセスが正常完了した場合またはノンブロッキングモードが有効の *
                        /* とき、動作が正常に開始された場合 */
ETHER_ERR_OTHER         /* ノンブロッキングモードが有効のとき、割り込みハンドラ関数が登録*/
                        /* されていない場合 */
ETHER_ERR_INVALID_CHAN /* 存在しないチャネルの場合 */
ETHER_ERR_LOCKED        /* ノンブロッキングモードが有効のとき、PHY アクセス中の場合 */
```

Properties

r_ether_rx_if.h にプロトタイプ宣言されています。

Description

R_ETHER_ReadPHY 関数は PHY 管理インタフェースを使用して、PHY-LSI 内のレジスタにリードアクセスします。 ノンブロッキングモードが無効の場合、PHY-LSI から読み出したレジスタの値は引数 p_data に格納されます。また、リードアクセスが正常に完了したら ETHER_SUCCESS が返されます。
ノンブロッキングモードが有効の場合、読み出した値がコールバック関数の引数として渡されます。

Example

```
#include "platform.h"
#include "r_ether_rx_if.h"
ether_return_t          ret;
uint32_t                      channel;
uint16_t                      address;
```

```
uint16_t                        data;
channel = ETHER_CHANNEL_0;

address = PHY_REG_CONTROL;
ret = R_ETHER_ReadPHY(channel, address, &data);
```

Special Notes:
なし。

3.17　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_rx_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。

Example

```
#include "platform.h"
#include "r_ether_rx_if.h"

uint32_t   version;

version = R_ETHER_GetVersion();
```

Special Notes:

なし。

4. 端子設定

イーサネット FIT モジュールを使用するためには、マルチファンクションピンコントローラ（MPC）で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。

e^2 studio の場合は「FIT Configurator」または「Smart Configurator」の端子設定機能を使用することができます。FIT Configurator、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。詳細は表 4.1を参照してください。

表 4.1　FIT コンフィグレータが出力する関数一覧

使用マイコン	選択したオプション	出力される関数名	備考
RX64M RX71M RX65N RX72M RX72N RX66N	チャネル 0 MII モード	R_ETHER_PinSet_ETHERC0_MII()	MII モードでチャネル 0 を使用する場合
	チャネル 0 RMII モード	R_ETHER_PinSet_ETHERC0_RMII()	RMII モードでチャネル 0 を使用する場合
	チャネル 1 MII モード	R_ETHER_PinSet_ETHERC1_MII()	MII モードでチャネル 1 を使用する場合
	チャネル 1 RMII モード	R_ETHER_PinSet_ETHERC1_RMII()	RMII モードでチャネル 1 を使用する場合

4.1　RSK+RX64M/RSK+RX71M/RSK+RX72M を使用する場合の端子設定例

表 4.3、表 4.4に RSK+RX64M, RSK+RX71M の端子設定例を示します、表 4.5、表 4.6に RSK+RX72M の端子設定例を示します。使用するチャネルとイーサネット FIT モジュールのコンフィギュレーションオプションの設定によって端子設定するチャネルが決まります。詳細は表 4.2を参照してください。また、記載してある設定値以外では使用しないでください。

表 4.2　使用チャネルとコンフィギュレーションオプションによる必要な端子設定の組み合わせ

使用するチャネル	コンフィグレーションオプションの設定	端子設定するチャネル
チャネル 0	ETHER_CFG_CH0_PHY_ACCESS (0) ETHER_CFG_CH1_PHY_ACCESS (0)	チャネル 0
チャネル 0	ETHER_CFG_CH0_PHY_ACCESS (1) ETHER_CFG_CH1_PHY_ACCESS (1)	チャネル 0 チャネル 1
チャネル 1	ETHER_CFG_CH0_PHY_ACCESS (0) ETHER_CFG_CH1_PHY_ACCESS (0)	チャネル 0 チャネル 1
チャネル 1	ETHER_CFG_CH0_PHY_ACCESS (1) ETHER_CFG_CH1_PHY_ACCESS (1)	チャネル 1
チャネル 0 チャネル 1	Don’t Care	チャネル 0 チャネル 1

表 4.3　RSK+RX64M/RSK+RX71M のチャネル 0 の端子設定例

MII モードを使用する場合	RMII モードを使用する場合	I/O ポート
ET0_TX_CLK		PC4
ET0_RX_CLK	REF50CK0	P76
ET0_TX_EN	RMII0_TXD_EN	P80
ET0_ETXD3		PC6
ET0_ETXD2		PC5
ET0_ETXD1	RMII0_TXD1	P82
ET0_ETXD0	RMII0_TXD0	P81
ET0_TX_ER		PC3
ET0_RX_DV		PC2
ET0_ERXD3		PC0
ET0_ERXD2		PC1
ET0_ERXD1	RMII0_RXD1	P74
ET0_ERXD0	RMII0_RXD0	P75
ET0_RX_ER	RMII0_RX_ER	P77
ET0_CRS	RMII0_CRS_DV	P83
ET0_COL		PC7
ET0_MDC		P72
ET0_MDIO		P71
ET0_LINKSTA		P34*[1]
ET0_EXOUT		-*[2]
ET0_WOL		-*[2]

【注】 *1 ETHER_CFG_USE_LINKSTA を値 0 に設定している場合は設定不要です。
【注】 *2 イーサネット FIT モジュールでは使用しない端子なので設定不要です。

表 4.4　RSK+RX64M/RSK+RX71M のチャネル 1 の端子設定例

MII モードを使用する場合	RMII モードを使用する場合	I/O ポート
ET1_TX_CLK		PG2
ET1_RX_CLK	REF50CK1	PG0
ET1_TX_EN	RMII1_TXD_EN	P60
ET1_ETXD3		PG6
ET1_ETXD2		PG5
ET1_ETXD1	RMII1_TXD1	PG4
ET1_ETXD0	RMII1_TXD0	PG3
ET1_TX_ER		PG7
ET1_RX_DV		P90
ET1_ERXD3		P97
ET1_ERXD2		P96
ET1_ERXD1	RMII1_RXD1	P95
ET1_ERXD0	RMII1_RXD0	P94
ET1_RX_ER	RMII1_RX_ER	PG1
ET1_CRS	RMII1_CRS_DV	P92
ET1_COL		P91
ET1_MDC		P31
ET1_MDIO		P30
ET1_LINKSTA		P93*[1]
ET1_EXOUT		-*[2]
ET1_WOL		-*[2]

【注】 *1 ETHER_CFG_USE_LINKSTA を値 0 に設定している場合は設定不要です。
【注】 *2 イーサネット FIT モジュールでは使用しない端子なので設定不要です。

表 4.5　RSK+RX72M のチャネル 0 の端子設定例

MII モードを使用する場合	RMII モードを使用する場合	I/O ポート
CLKOUT25M		PH7
ET0_TX_CLK		PM6
ET0_RX_CLK	REF50CK0	PL3
ET0_TX_EN	RMII0_TXD_EN	PL6
ET0_ETXD3		PM5
ET0_ETXD2		PM4
ET0_ETXD1	RMII0_TXD1	PL5
ET0_ETXD0	RMII0_TXD0	PL4
ET1_TX_ER		-*2
ET0_RX_DV		PK2
ET0_ERXD3		PK5
ET0_ERXD2		PK4
ET0_ERXD1	RMII0_RXD1	P74
ET0_ERXD0	RMII0_RXD0	P75
ET0_RX_ER	RMII0_RX_ER	PL2
ET0_CRS	RMII0_CRS_DV	PM7
ET0_COL		PK1
ET0_MDC		PK0
ET0_MDIO		PL7
ET0_LINKSTA		P34*1
ET0_EXOUT		-*2
ET0_WOL		-*2

【注】 *1 ETHER_CFG_USE_LINKSTA を値 0 に設定している場合は設定不要です。
【注】 *2 イーサネット FIT モジュールでは使用しない端子なので設定不要です。

表 4.6　RSK+RX72M のチャネル 1 の端子設定例

MII モードを使用する場合	RMII モードを使用する場合	I/O ポート
CLKOUT25M		PH7
ET1_TX_CLK		PN2
ET1_RX_CLK	REF50CK1	PQ4
ET1_TX_EN	RMII1_TXD_EN	PQ7
ET1_ETXD3		PN1
ET1_ETXD2		PN0
ET1_ETXD1	RMII1_TXD1	PQ6
ET1_ETXD0	RMII1_TXD0	PQ5
ET1_TX_ER		-*2
ET1_RX_DV		PQ2
ET1_ERXD3		PM3
ET1_ERXD2		PM2
ET1_ERXD1	RMII1_RXD1	PM1
ET1_ERXD0	RMII1_RXD0	PM0
ET1_RX_ER	RMII1_RX_ER	PN3
ET1_CRS	RMII1_CRS_DV	PQ0
ET1_COL		PQ1
ET0_MDC		PK0
ET0_MDIO		PL7
ET1_LINKSTA		P84*1
ET1_EXOUT		-*2
ET1_WOL		-*2

【注】 *1 ETHER_CFG_USE_LINKSTA を値 0 に設定している場合は設定不要です。

【注】 *2 イーサネット FIT モジュールでは使用しない端子なので設定不要です。

4.2　RSK+RX65N/RSK+RX65N-2M を使用する場合の端子設定例

表 4.7に RSK+RX65N、RSK+RX65N-2M の端子設定例を示します。

表 4.7　RSK+RX65N,RSK+RX65N-2M での端子設定例

MII モードを使用する場合	RMII モードを使用する場合	I/O ポート
ET0_TX_CLK		PC4
ET0_RX_CLK	REF50CK0	P76
ET0_TX_EN	RMII0_TXD_EN	P80
ET0_ETXD3		PC6
ET0_ETXD2		PC5
ET0_ETXD1	RMII0_TXD1	P82
ET0_ETXD0	RMII0_TXD0	P81
ET0_TX_ER		PC3
ET0_RX_DV		PC2
ET0_ERXD3		PC0
ET0_ERXD2		PC1
ET0_ERXD1	RMII0_RXD1	P74
ET0_ERXD0	RMII0_RXD0	P75
ET0_RX_ER	RMII0_RX_ER	P77
ET0_CRS	RMII0_CRS_DV	P83
ET0_COL		PC7
ET0_MDC		P72
ET0_MDIO		P71
ET0_LINKSTA		P54 (RSK+RX65N の場合)*1 P34 (RSK+RX65N-2M の場合)*1
ET0_EXOUT		-*2
ET0_WOL		-*2

【注】 *1 ETHER_CFG_USE_LINKSTA を値 0 に設定している場合は設定不要です。
【注】 *2 イーサネット FIT モジュールでは使用しない端子なので設定不要です。

4.3　RSK+RX72N を使用する場合の端子設定例

表 4.8に RSK+RX72N の端子設定例を示します。

表 4.8　RSK+RX72N での端子設定例

MII モードを使用する場合	RMII モードを使用する場合	I/O ポート
CLKOUT25M		PH7
ET1_TX_CLK		PN2
ET1_RX_CLK	REF50CK1	PQ4
ET1_TX_EN	RMII1_TXD_EN	PQ7
ET1_ETXD3		PN1
ET1_ETXD2		PN0
ET1_ETXD1	RMII1_TXD1	PQ6
ET1_ETXD0	RMII1_TXD0	PQ5
ET1_TX_ER		-*2
ET1_RX_DV		P90
ET1_ERXD3		P97
ET1_ERXD2		P96
ET1_ERXD1	RMII1_RXD1	P95
ET1_ERXD0	RMII1_RXD0	P94
ET1_RX_ER	RMII1_RX_ER	PN3
ET1_CRS	RMII1_CRS_DV	PQ0
ET1_COL		P91
ET1_MDC		P31
ET1_MDIO		P30
ET1_LINKSTA		P93*1
ET1_EXOUT		-*2
ET1_WOL		-*2

【注】 *1 ETHER_CFG_USE_LINKSTA を値 0 に設定している場合は設定不要です。
【注】 *2 イーサネット FIT モジュールでは使用しない端子なので設定不要です。

5. 使用方法

5.1 セクション配置

表 5.1にイーサネット FIT モジュール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

アドレス	デバイス	セクション	説明
0x00000020	内蔵 RAM	SI	割り込みスタック領域
		SU	ユーザスタック領域
		B_1	1byte 境界の未初期化データ領域
		R_1	1byte 境界の初期化データ領域（変数）
		B_2	2byte 境界の未初期化データ領域
		R_2	2byte 境界の初期化データ領域（変数）
		B	4byte 境界の未初期化データ領域
		R	4byte 境界の初期化データ領域（変数）
0x00010000		B_ETHERNET_BUFFERS_1	送信バッファおよび受信バッファ領域
		B_RX_DESC_1	受信ディスクリプタ領域
		B_TX_DESC_1	送信ディスクリプタ領域
0xFFFF8000	内蔵 ROM	C_1	1byte 境界の定数領域
		C_2	2byte 境界の定数領域
		C	4byte 境界の定数領域
		C$*	C$*セクション（C$DEC、C$BSEC、C$VECT）の定数領域
		D*	初期化データ領域
		P*	プログラム領域
		W*	switch 文分岐テーブル領域
		L	文字列リテラル領域
0xFFFFFF80		EXCEPTVECT	割り込みベクタ領域
0xFFFFFFFC		RESETVECT	リセットベクタ領域

5.1.1　GCC for Renesas RX のセクション設定例

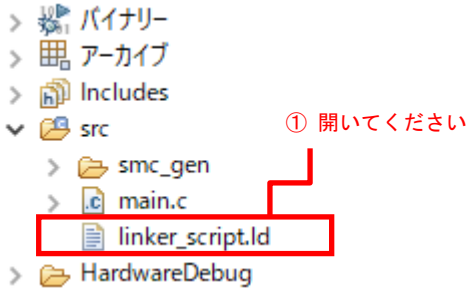
linker_script.ld ファイルを編集して、セクションおよびシンボルを追加してください。

セクションおよびシンボルの追加

下記のコードを追加。

```
      B_ETHERNET_BUFFERS_1 0x00010000 (NOLOAD) : AT(0x00010000)
      {
            _B_ETHERNET_BUFFERS_1_start = .;
            *(B_ETHERNET_BUFFERS_1)
            _B_ETHERNET_BUFFERS_1_end = .;
      } >RAM
      B_RX_DESC_1 (NOLOAD) :
      {
            _B_RX_DESC_1_start = .;
            *(B_RX_DESC_1)
            _B_RX_DESC_1_end = .;
      } >RAM
      B_TX_DESC_1 (NOLOAD) :
      {
            _B_TX_DESC_1_start = .;
            *(B_TX_DESC_1)
            _B_TX_DESC_1_end = .;
      } >RAM
```

①プロジェクトエクスプローラーから「linker_script.ld」を開く。



②「linker_script.ld」をクリック。

③ コードを入力。


```
125         {
126             *(.gcc_exc)
127         } > RAM
128         .bss :
129         {
130             _bss = .;
131             *(.bss)
132             *(.bss.**)
133             *(COMMON)
134             *(B)
135             *(B_1)
136             *(B_2)
137             _ebss = .;
138             _end = .;
139         } > RAM
140         B_ETHERNET_BUFFERS_1 0x00010000 (NOLOAD) : AT(0x00010000)
141         {
142             _B_ETHERNET_BUFFERS_1_start = .;
143             *(B_ETHERNET_BUFFERS_1)
144             _B_ETHERNET_BUFFERS_1_end = .;
145         } >RAM
146         B_RX_DESC_1 (NOLOAD) :
147         {
148             _B_RX_DESC_1_start = .;
149             *(B_RX_DESC_1)
150             _B_RX_DESC_1_end = .;
151         } >RAM
152         B_TX_DESC_1 (NOLOAD) :
153         {
154             _B_TX_DESC_1_start = .;
155             *(B_TX_DESC_1)
156             _B_TX_DESC_1_end = .;
157         } >RAM
158         .ofs1 0xFE7F5D40: AT(0xFE7F5D00)
159         {
160             KEEP(*(.ofs1))
161         } > OFS
```

5.1.2　IAR C/C++ Compiler for Renesas RX のセクション設定例

icf ファイルを編集して、セクション設定を追加してください。

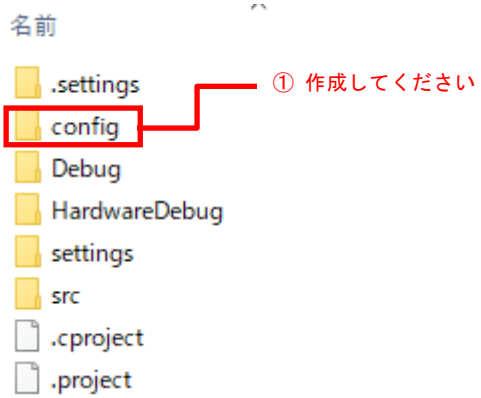
編集する icf ファイルはプロジェクトのターゲットデバイスにより異なりますので、使用するデバイスの型名上 8 桁を確認して編集してください。

例として RX65N(R5F565NEDDFC)では「lnkr5f565ne.icf」を編集します。

以下に RX65N(R5F565NEDDFC)での編集例を説明します。

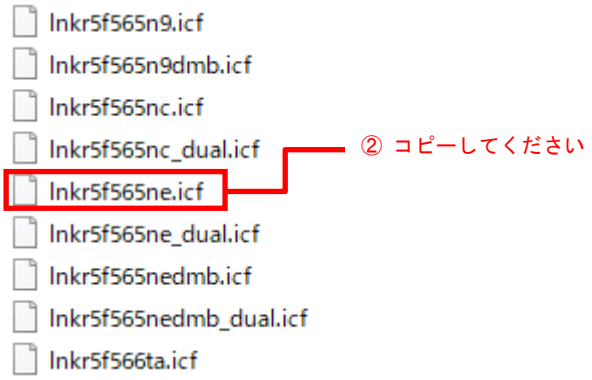
セクション設定の追加

①プロジェクトフォルダに「config」フォルダを作成。



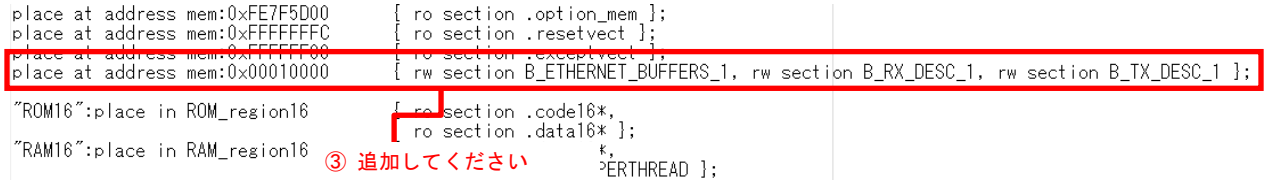
②IAR C/C++ Compiler for Renesas RX(以下「EWRX」と記載)をインストールしたフォルダの「¥rx¥config」から「lnkr5f565ne.icf」をプロジェクトフォルダの「config」フォルダにコピー。

インストール時のデフォルトは「C:¥Program Files (x86)¥IAR Systems¥Embedded Workbench 8.1」です。



③コピーした「lnkr5f565ne.icf」ファイルを開き以下のコードを追加。

```
place at address mem:0x00010000      { rw section B_ETHERNET_BUFFERS_1, rw section
B_RX_DESC_1, rw section B_TX_DESC_1 };
```



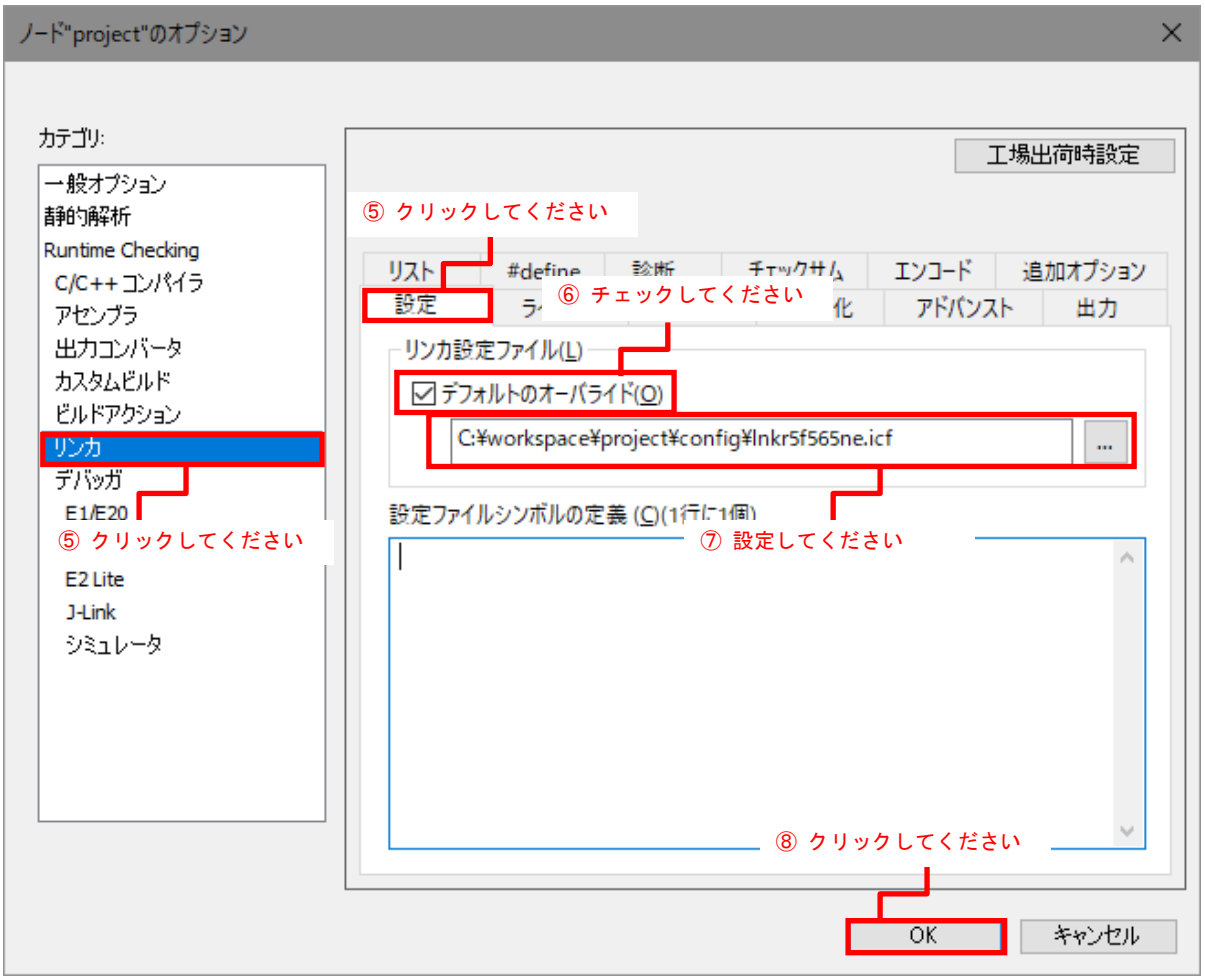
④EWRX から使用するプロジェクトを開き、ワークスペースのプロジェクトを右クリックし
オプションを開く。

⑤「カテゴリ：リンカ」を選択し、「設定」をクリック。

⑥「デフォルトのオーバライド」にチェックを入れる。

⑦③で編集したファイルを参照先に設定する。

⑧ 「OK」をクリック。



5.1.3　セクション配置の注意点

- 受信ディスクリプタ領域および送信ディスクリプタ領域は、EDMAC モードレジスタ（EDMR）の送受信ディスクリプタ長指定ビット（DL）を、16byte 設定にしているため、16byte 境界になるよう配置してください。
- 送信バッファおよび受信バッファ領域は 32byte 境界になるよう配置してください。
- セクション配置例は、e^2 studio の「FIT Configurator」、「Smart Configurator」を使用してイーサネット FIT モジュールをユーザプロジェクトにインストールした際に、自動で設定されます。ユーザプログラムに応じて設定を変更してください。
- RX64M、RX71M、RX72M、RX72N、RX66N でイーサネット FIT モジュールを使用する場合は、0000 0000h ～0000 001Fh 番地を使用しないでください。

5.2　イーサネット FIT モジュールの初期設定方法

図 5.1にイーサネット FIT モジュールの初期設定方法のフローチャートを示します。

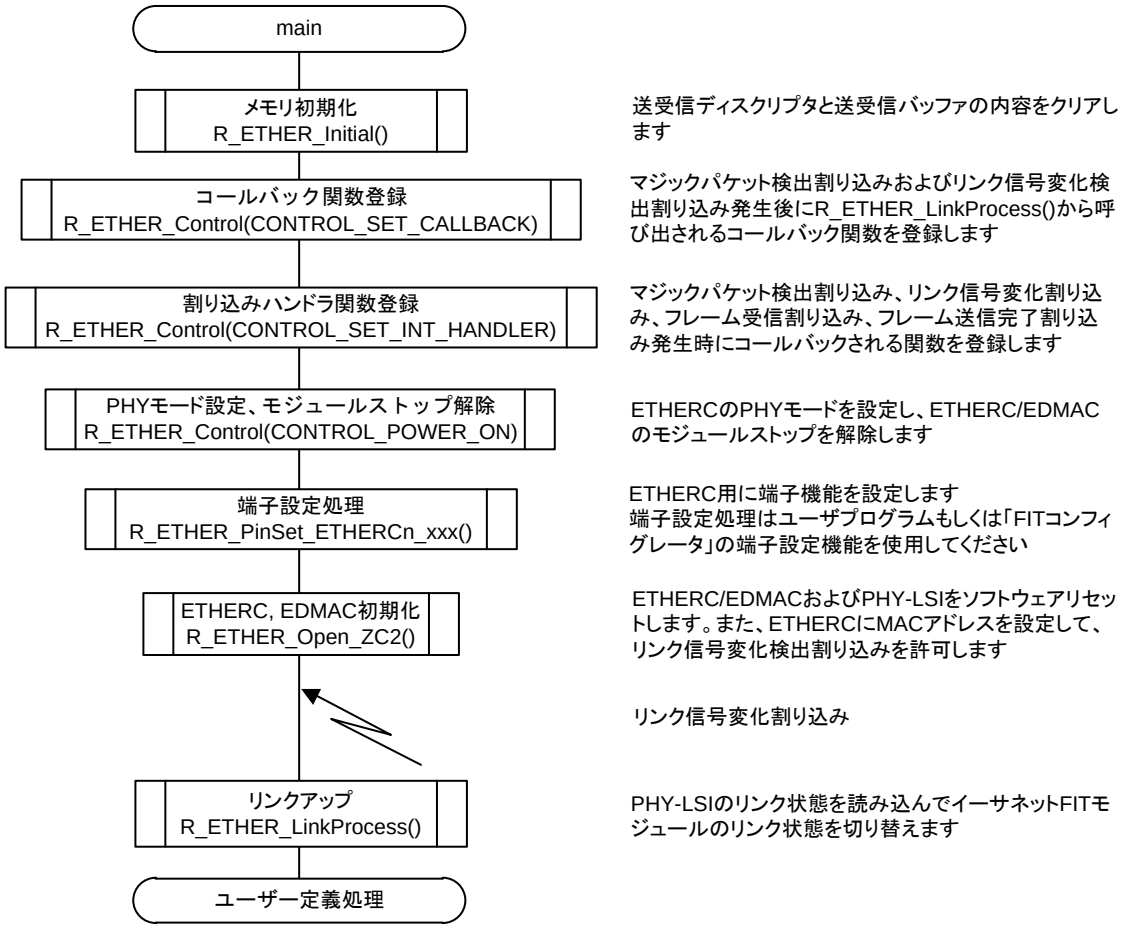


図 5.1　イーサネット FIT モジュールの初期設定方法のフローチャート

5.2.1　イーサネット FIT モジュール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。

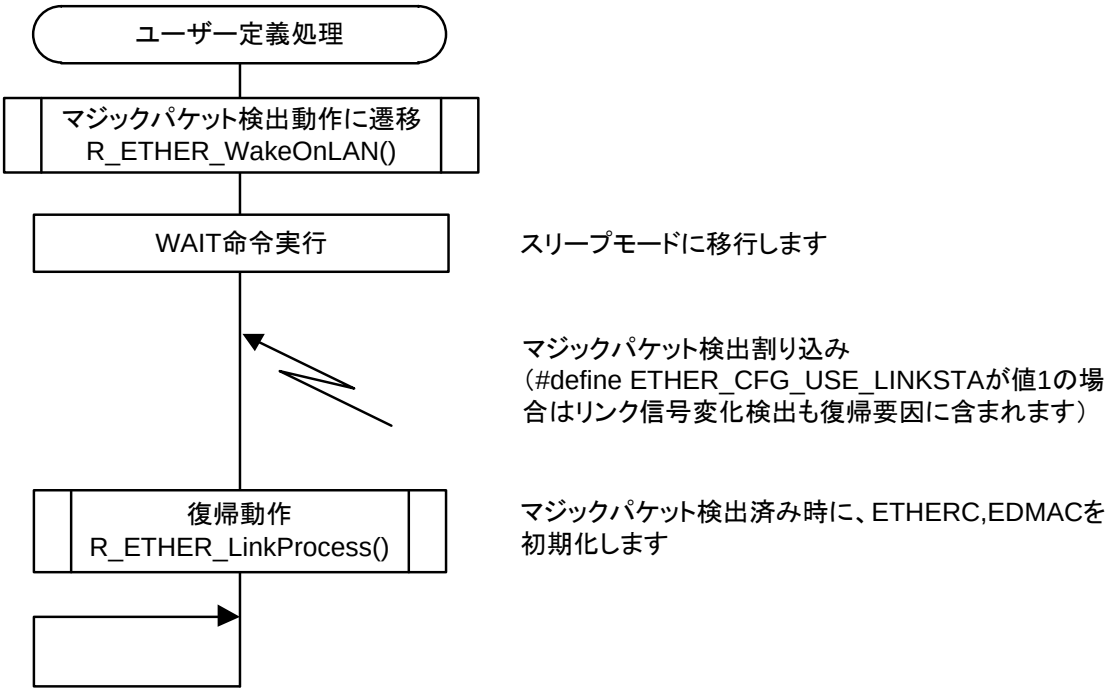


図 5.2　マジックパケット検出動作のフローチャート

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 WAIT 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. 付録

6.1 EPTPC Light FIT モジュール

RX64M/RX71M/RX72M/RX72N では、イーサネット FIT モジュールを EPTPC Light FIT モジュールと組み合わせることで、以下に示すハードウェアによる簡易スイッチ機能とマルチキャストフレームフィルタ機能が使用できます。

(1) 簡易スイッチ

2 チャネルの ETHERC を使用時、チャネル間のフレーム転送をハードウェアで行います。

転送方向は、チャネル 0 からチャネル 1、チャネル 1 からチャネル 0、双方向を選択でき、転送方式はストア&フォワードとカットスルーを選択できます。

(2) マルチキャストフレームフィルタ

マルチキャストフレームを ETHERC が受信した場合、受信または破棄の処理をハードウェアで行います。

全て受信する、全て受信しない、または、特定の宛先アドレス（2 種まで登録可能）を持つフレームのみ受信することができます。

詳細は EPTPC Light FIT モジュールのアプリケーションノート「RX ファミリ EPTPC Light モジュール Firmware Integration Technology, ドキュメント No.R01AN3035」を参照ください。

6.1.1 使用上の注意点

イーサネット FIT モジュールを EPTPC Light FIT モジュールと組み合わせて使用する場合、IEEE1588 準拠の時刻同期機能を持つ EPTPC FIT モジュール（完全版）*[1] との同時使用はできません。

RX64M/RX71M/RX72M/RX72N の簡易スイッチとマルチキャストフレームフィルタを使用する場合、下記のどちらかを選択してください。

- IEEE1588 時刻同期機能を使用しない

  EPTPC Light FIT モジュールを選択（モジュール名：r_ptp_light_rx）
- IEEE1588 時刻同期機能を使用する

  EPTPC FIT モジュール（完全版）を選択（モジュール名：r_ptp_rx）

【注】 *1RX ファミリ EPTPC モジュール Firmware Integration Technology, ドキュメント No.R01AN1943

6.2　動作確認環境

本 FIT モジュールの動作確認環境を以下に示します。

表 6.1　動作確認環境 (Rev.1.13)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V6.00.001
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V2.07.00
	コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev1.13
使用ボード	Renesas Starter Kit for RX64M（型名：R0K50564MSxxxBE） Renesas Starter Kit for RX65N（型名：RTK500565NSxxxxxBE） Renesas Starter Kit for RX65N-2MB（型名：RTK50565N2SxxxxxBE）

表 6.2　動作確認環境 (Rev.1.15)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V6.2.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V2.08.00
	コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev1.15
使用ボード	Renesas Starter Kit for RX64M（型名：R0K50564MSxxxBE） Renesas Starter Kit for RX71M（型名：R0K50571MCxxxBE） Renesas Starter Kit for RX65N（型名：RTK500565NSxxxxxBE）

表 6.3　動作確認環境 (Rev.1.16)

項目	内容
統合開発環境	ルネサス エレクトロニクス製 e^2 studio V7.3.0 IAR Embedded Workbench for Renesas RX 4.10.01
C コンパイラ	ルネサス エレクトロニクス製 C/C++ Compiler for RX Family V3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.08.04.201803 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.10.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev1.16
使用ボード	Renesas Starter Kit+ for RX65N（型名：RTK500565Nxxxxxx）

表 6.4 動作確認環境 (Rev.1.17)

項目	内容
統合開発環境	ルネサス エレクトロニクス製 e^2 studio V7.4.0 IAR Embedded Workbench for Renesas RX 4.12.01
C コンパイラ	ルネサス エレクトロニクス製 C/C++ Compiler for RX Family V3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.08.04.201902 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.12.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev1.17
使用ボード	Renesas Starter Kit+ for RX72M（型名：RTK5572Mxxxxxxxxx）

表 6.5 動作確認環境 (Rev.1.20)

項目	内容
統合開発環境	ルネサス エレクトロニクス製 e^2 studio V7.4.0 IAR Embedded Workbench for Renesas RX 4.12.01
C コンパイラ	ルネサス エレクトロニクス製 C/C++ Compiler for RX Family V3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.08.04.201902 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.12.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev1.20
使用ボード	Renesas Starter Kit+ for RX72N（型名：RTK5572Nxxxxxxxxx）

6.3　トラブルシューティング

(1) Q：本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「Could not open source file "platform.h"」エラーが発生します。

A：FIT モジュールがプロジェクトに正しく追加されていない可能性があります。プロジェクトへの追加方法をご確認ください。

- CS+を使用している場合
  アプリケーションノート RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」
- e^2 studio を使用している場合
  アプリケーションノート RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)」

また、本 FIT モジュールを使用する場合、ボードサポートパッケージ FIT モジュール(BSP モジュール)もプロジェクトに追加する必要があります。BSP モジュールの追加方法は、アプリケーションノート「ボードサポートパッケージモジュール(R01AN1685)」を参照してください。

(2) Q：本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「This MCU is not supported by the current r_ether_rx module.」エラーが発生します。

A：追加した FIT モジュールがユーザプロジェクトのターゲットデバイスに対応していない可能性があります。追加した FIT モジュールの対象デバイスを確認してください。

(3) Q：本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「ERROR-ETHER_CFG_xxx_xxx - …」エラーが発生します。

A：“r_ether_rx_config.h”ファイルの設定値が間違っている可能性があります。“r_ether_rx_config.h”ファイルを確認して正しい値を設定してください。詳細は「2.7 コンパイル時の設定」を参照してください。

(4) Q：データの送受信が開始されない。

A：正しく端子設定が行われていない可能性があります。本 FIT モジュールを使用する場合は端子設定が必要です。詳細は「4 端子設定」を参照してください。

7. 提供するモジュール

提供するモジュールは、ルネサス エレクトロニクスホームページから入手してください。

8. イーサネット FIT モジュール使用時の注意事項

イーサネット FIT モジュールを使用する際には、以下の注意事項があります。

- RX64M/RX71M/RX72M/RX72N/RX66N において外部回線上の破損フレームおよびノイズにより、ETHERC および EPTPC が受信中に異常フレームを検出すると、それ以降に正常フレームを受信しても正しく受信できない場合があります。詳細は以下のテクニカルアップデートおよびアプリケーションノートを参照してください。
- イーサネットコントローラの使用上の注意事項（テクニカルアップデート No. TN-RX*-A125A/J）
- RX ファミリ イーサネットコントローラ INFABT 発生時の推奨復帰処理（ドキュメント No.R01AN2604）
- RX64M/RX71M/RX72M/RX72N/RX66N において EDMAC0、EDMAC1、PTPEDMAC がデータ転送を実施しているときに EDMR.SWR ビットを”1”にすると、0000 0000h～0000 001Fh 番地のデータが破壊されることがあります。詳細は以下のテクニカルアップデートを参照してください。
- RX64M グループ、RX71M グループ イーサネットコントローラ用 DMA コントローラ (EDMAC) のソフトウェアリセットに関する注意事項（テクニカルアップデート No. TN-RX*-A0212A/J）

9. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

RX64M グループ ユーザーズマニュアル ハードウェア編（ドキュメント No.R01UH0377）
RX71M グループ ユーザーズマニュアル ハードウェア編（ドキュメント No.R01UH0493）
RX65N グループ、RX651 グループ　ユーザーズマニュアル　ハードウェア編（ドキュメント No. R01UH0590）
RX72M グループ　ユーザーズマニュアル　ハードウェア編（ドキュメント No. R01UH0804）
RX72N グループ　ユーザーズマニュアル　ハードウェア編（ドキュメント No. R01UH0824）
RX66N グループ　ユーザーズマニュアル　ハードウェア編（ドキュメント No. R01UH0825）
（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート／テクニカルニュース

（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：開発環境

RX ファミリ C/C++コンパイラ、アセンブラ、最適化リンケージエディタ コンパイラパッケージ（R20UT0570）
（最新版をルネサス エレクトロニクスホームページから入手してください。）

改訂記録	RXファミリ　イーサネットモジュール Firmware Integration Technology

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2014.07.29	—	初版発行
1.01	2015.01.27	1	「対象デバイス」　RX71M を追加
		5	「2.6　コンパイル時の設定」 注*1～*3 を修正。表 2.2 を追加
		9	「2.10.1　イーサネット FIT モジュールの追加手順」 手順 7、8 を変更
		10	「3.1 R_ETHER_Initial()」　「Special Notes:」を修正
		22	「3.11 R_ETHER_CheckWrite()」　「Special Notes:」を修正
		29	「3.14 R_ETHER_Control()」　「Special Notes:」を修正
		31	「表 4.1　プログラムのセクション配置例」 アドレス 0x00120064 を削除
		52	「6.　参考ドキュメント」 RX71M のユーザーズマニュアルを追加 開発環境のユーザーズマニュアルを変更
1.02	2015.3.27	—	“r_ether_rx.c”内の R_ETHER_LinkProcess()を変更
1.10	2016.3.31	1	「対象デバイス」　RX63N を追加
		3	「概要」　制限事項の記述を削除
		5	「2.6　コンパイル時の設定」 #define ETHER_CFG_EINT_INT_PRIORITY を追加
		6	「2.6　コンパイル時の設定」 #define ETHER_CFG_USE_LINKSTA を追加 注*4～*8 を追加
		7	「2.7 コードサイズを追加」
		9	「2.8　引数」 ether_cmd_t の内容を変更
		10	「2.9　戻り値」 ether_return_t の内容を変更
		11	「2.10　コールバック関数」 「（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」の内容を変更。注*1 を追加
		12	「2.11　FIT モジュールの追加方法」の内容を変更
		13	「2.12　イーサネットフレームのフレーム形式」を追加
		14	「3.　API 関数」 各 API 関数の内容を変更
		47	「4.2　イーサネット FIT モジュールの初期設定方法」を追加
		48	「4.3　EPTPC Light FIT モジュール」を追加
		49	「4.4　マジックパケット検出動作」を追加
		-	「4.2　サンプルコード」を削除
		50	「6.　イーサネット FIT モジュール使用時の注意事項」を追加
1.11	2016.10.01	-	e2 studio の端子設定機能対応に伴い、イーサネット FIT モジュール内の端子設定処理を削除

		1	「対象デバイス」　RX65N を追加
		7	「2.6　コンパイル時の設定」の注*5、*7、*8 の内容を変更
		40	「3.14　R_ETHER_Control()」の Description の内容を変更
		47	「4.2　イーサネット FIT モジュールの初期設定方法」 図 4.1 初期設定方法のフローチャートを一部変更
		48	「4.3　イーサネット FIT モジュールの端子設定方法」を追加
		51	5. 付録を追加
1.12	2016.11.11	ﾌﾟﾛｸﾞﾗﾑ	ソフトウェア不具合のため、イーサネット FIT モジュールを改修 ■内容 R_ETHER_LinkProcess 関数が呼び出されたとき、リンクアップ/リンクダウンが正常に処理されない場合がある。 ■発生条件 ETHER_CFG_USE_LINKSTA を値 0 に設定している。 ■対策 イーサネット FIT モジュール Rev1.12 を使用してください。
1.13	2017.10.01	-	RX65N-2MB 版に対応
		60	「2.3 動作確認環境」を「6.2 動作確認環境」に移動
		6	「2.4 使用する割り込みベクタ」の誤記を修正
		9	「2.6　コンパイル時の設定」の注*7、*8 の内容を変更
		14	「2.12 FIT モジュールの追加方法」の内容を変更
		36	「3.9 R_ETHER_LinkProcess()」 Special Notes:の誤記を修正
		50	「4.3 イーサネット FIT モジュールの端子設定方法」を「4. 端子設定」に移動
		52,53	「4.2 RSK+RX63N の端子設定例」、「4.3 RSK+RX65N/RSK+RX65N-2M の端子設定例」を追加
		54	「5.1 セクション配置」の内容を変更
		55	「5.2 イーサネット FIT モジュールの初期設定方法」の内容を変更
		58	「6.3 トラブルシューティング」を追加
1.14	2018.01.08	-	Smart Configurator での GUI によるコンフィグオプション設定機能に対応
		7	「2.7 コンパイル時の設定」の内容を修正
		50	「4. 端子設定」 表 4.1 の誤記を修正
		59	「9. 参考ドキュメント」の内容を変更
1.15	2018.05.07	ﾌﾟﾛｸﾞﾗﾑ	ソフトウェア不具合のため、イーサネット FIT モジュールを改修 ■内容 R_ETHER_Read_ZC2 関数または R_ETHER_Read 関数が呼び出されたとき、正常にイーサネットフレームを受信できない場合がある。 ■発生条件 割り込み関数内で R_ETHER_Read_ZC2 関数または R_ETHER_Read 関数を呼び出している。 ■対策 イーサネット FIT モジュール Rev1.15 を使用してください。 本修正により、以下の関数を変更しています。 R_ETHER_LinkProcess 関数 対応ツールニュース番号 ：R20TS0307

		ﾌﾟﾛｸﾞﾗﾑ	ソフトウェア不具合のため、イーサネット FIT モジュールを改修 ■内容 R_ETHER_LinkProcess 関数が呼び出されたとき、正常にリンクアップ処理が完了しない場合がある。 ■発生条件 R_ETHER_LinkProcess 関数が呼び出されたとき、PHY のオートネゴシエーションが完了していない。 ■対策 イーサネット FIT モジュール Rev1.15 を使用してください。 本修正により、以下の関数を変更しています。 R_ETHER_LinkProcess 関数 対応ツールニュース番号 ：R20TS0307
		ﾌﾟﾛｸﾞﾗﾑ	ソフトウェア不具合のため、イーサネット FIT モジュールを改修 ■内容 R_ETHER_Read_ZC2 関数または R_ETHER_Read 関数が呼び出されたとき、関数の実行が終了しない場合がある。 ■発生条件 割り込み関数内で R_ETHER_LinkProcess 関数を呼び出している。 ■対策 イーサネット FIT モジュール Rev1.15 を使用してください。 本修正により、以下の関数を変更しています。 R_ETHER_Read_ZC2 関数 対応ツールニュース番号 ：R20TS0307
		10	「2.8 コードサイズ」の内容を変更
		58	「6.2 動作確認環境」の内容を変更
1.16	2019.05.20	-	以下のコンパイラに対応 ・GCC for Renesas RX ・IAR C/C++ Compiler for Renesas RX
		1	「対象コンパイラ」を追加
		1	「関連ドキュメント」　R01AN1723、R01AN1826、R20AN0451 を削除
		5	「2.2 ソフトウェアの要求」依存する r_bsp モジュールのリビジョンを追加
		9	「2.8 コードサイズ」を更新
		50	3.1.15　R_ETHER_GetVersion に関数のインライン展開を削除
		56-59	5.1.1 と 5.1.2 を追加
		64	「6.2 動作確認環境」に、表 6.3 動作確認環境を追加
1.17	2019.07.30	-	RX72M 版に対応
		4	「1.3 制限事項」を追加
		6	「表 2.1　使用する割り込みベクター覧」を更新
		9	注 *5～*8 を更新 注 *9 を追加
		10	「2.8 コードサイズ」を更新
		17	「2.14 for 文、while 文、do while 文について」を追加
		18-50	API 説明ページの「Reentrant」項目を削除

		51-54	「表 4.5 RSK+RX72M のチャネル 0 の端子設定例」、 「表 4.6 RSK+RX72M のチャネル 1 の端子設定例」を追加
		57	「5.1 セクション配置」を変更
		62	「5.1.3 セクション配置の注意点」を変更
		65	「6.1 EPTPC Light FIT モジュール」に RX72M を追加
		66	「6.2 動作確認環境」の内容を変更
		69	「8. イーサネット FIT モジュール使用時の注意事項」を変更 「9.参考ドキュメント」を変更
		ﾌﾟﾛｸﾞﾗﾑ	ソフトウェア不具合のため、イーサネット FIT モジュールを改修 ■内容 R_ETHER_Read_ZC2_BufRelease 関数または R_ETHER_Read 関数におけるバッファ開放処理後にイーサネットフレームを受信すると以下の現象が発生する場合がある。 (1)　受信したイーサネットフレームを読み出せない場合がある。 (2)　受信したエラーフレームを正常なイーサネットフレームとして読み出す場合がある。 ■発生条件 R_ETHER_Read_ZC2_BufRelease 関数または R_ETHER_Read 関数におけるバッファ開放処理後に、イーサネットフレームを受信する。 ■対策 イーサネット FIT モジュール Rev1.17 を使用してください。 本修正により、以下の関数を変更しています。 R_ETHER_Read_ZC2_BufRelease 関数 対応ツールニュース番号：R20TS0447
1.20	2019.11.22	-	RX72N/RX66N 版に対応
		5	「表 1.1　API 関数一覧」を更新
		6	「表 2.1　使用する割り込みベクター覧」を更新
		8	「Configuration options in r_ether _rx_config.h」を更新
		10	注 *4～*8 を更新 注 *9 を追加
		10-11	「2.8 コードサイズ」を更新
		17-18	「2.11 コールバック関数」を更新
		24	「3.2 R_ETHER_Open_ZC2()」を更新
		37	「3.8 R_ETHER_CheckLink_ZC()」を更新
		39	「3.9 R_ETHER_LinkProcess()」を更新
		42	「3.10 R_ETHER_WakeOnLAN()」を更新
		50-54	「3.14 R_ETHER_Control()」を更新
		55	「3.15 R_ETHER_WritePHY()」を追加
		57	「3.16 R_ETHER_ReadPHY()」を追加
		59	4.4 RSK+RX72N を使用する場合の端子設定例
		69	「6.2 動作確認環境」の内容を変更
		72	「9.参考ドキュメント」を変更
		ﾌﾟﾛｸﾞﾗﾑ	ソフトウェア不具合のため、イーサネット FIT モジュールを改修 ■内容 受信ディスクリプタを 1 個に設定したときに “R_ETHER_Read_ZC2_BufRelease”関数または “R_ETHER_Read”関数におけるバッファ開放処理後、イーサネットフレームを受信できない場合がある。

			■発生条件 R_ETHER_Read_ZC2_BufRelease 関数または R_ETHER_Read 関数を使用して、イーサネットフレームを受信する。 ■対策 イーサネット FIT モジュール Rev1.20 を使用してください。 本修正により、以下の関数を変更しています。 R_ETHER_Read_ZC2 R_ETHER_Read_ZC2_BufRelease 関数 対応ツールニュース番号：R20TS0481
		ﾌﾟﾛｸﾞﾗﾑ	ソフトウェア不具合のため、イーサネット FIT モジュールを改修 ■内容 送信ディスクリプタを 1 個に設定したときに、“R_ETHER_Write_ZC2_SetBuf”関数または“R_ETHER_Write”関数における送信開始処理後、イーサネットフレームを送信できない場合がある。 ■発生条件 R_ETHER_Read_ZC2_SetBuf 関数または R_ETHER_Write 関数を使用して、イーサネットフレームを送信する。 ■対策 イーサネット FIT モジュール Rev1.20 を使用してください。 本修正により、以下の関数を変更しています。 R_ETHER_Write_ZC2_GetBuf 関数 対応ツールニュース番号：R20TS0481

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。