RX Family

EPTPC Light 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PTP 簡易版ソフトウェアドライバ（PTP 簡易ドライバ、EPTPC Light FIT モジュール）の内容を説明します。PTP 簡易ドライバはIEEE1588-2008 [1]で規定しPTPドライバ（完全版）[2]に実装されたPrecision Time Protocol（PTP）の機能を削減し、簡易スイッチやマルチキャストフレームフィルタ等の拡張した標準イーサネット機能の提供に特化しています。

動作確認デバイス

以下のデバイスがこのソフトウェアでサポートされています。

- RX64M グループ
- RX71M グループ
- RX72M グループ

本アプリケーションノートを他のルネサス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。

目次

1. 概要

1.1 EPTPC Light FIT モジュールとは

本モジュールは API として、プロジェクトに組み込んで使用します。本モジュールの組み込み方については、「2.14 FIT モジュールの追加方法」を参照してください。

1.2 EPTPC Light FIT モジュールの概要

本アプリケーションノートは、Firmware Integration Technology (FIT）に基づいた PTP 簡易版ソフトウェアドライバ（以後、PTP 簡易ドライバ）と、その使用例を説明します。PTP 簡易ドライバは EPTPC 周辺モジュールを使用することで、イーサネットフレームに関する簡易スイッチ機能やマルチキャストフレームフィルタ機能を提供します。PTP 簡易ドライバは PTP ドライバ（完全版）のサブセットであり、PTP による時刻同期機能には対応しておりません。

1.3 関連ドキュメント

[1] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, Revision of IEEE Std 1588-2008, Mar 2008
[2] RX ファミリ EPTPC モジュール Firmware Integration Technology, Rev.1.16, Document No.R01AN1943JJ0116, Aug 31, 2019
[3] RX ファミリ イーサネットモジュール Firmware Integration Technology, Rev.1.17, Document No. R01AN2009JJ0117, 発行予定
[4] RX ファミリ イーサネットコントローラ：簡易スイッチ動作例、Rev.1.11, Document No. R01AN3036JJ0111, Nov 11, 2016
[5] RX ファミリ イーサネットコントローラ：マルチキャストフレームフィルタ動作例、Rev.1.11, Document No. R01AN3037JJ0111, Nov 11, 2016
[6] Renesas Starter Kit+ for RX64M, ユーザーズマニュアル、Rev.1.20, Document No. R20UT2590JG0102, Jun 25, 2015
[7] Renesas Starter Kit+ for RX71M, ユーザーズマニュアル、Rev.1.00, Document No. R20UT3217JG0100, Jan 23, 2015

1.4 ハードウェアの構成

RX64M/71M/72M グループのイーサネットモジュールは、EPTPC、PTP 用イーサネットコントローラ用 DMA コントローラ（PTPEDMAC）、2 チャネルのイーサネットコントローラ（ETHERC（CH0）、ETHERC（CH1））、および 2 チャネルのイーサネットコントローラ用 DMA コントローラ（EDMAC（CH0）、EDMAC（CH1））で構成しています。EPTPC は、PTP 同期フレーム処理部（CH0）、PTP 同期フレーム処理部（CH1）、パケット中継部（PRC-TC）、および統計的クロック補正部で構成しています。また、EPTPC は、I/O ポートとモータ制御タイマ（MTU3 および GPT 周辺モジュール）に ELC 周辺モジュールを介して接続しており、PTP で時刻同期したパルスを出力できます。

イーサネットモジュールの概要を以下に示します。また、ハードウェアブロック図を図 1.1 に示します。なお、PTP 簡易ドライバは下記に示す機能の EPTPC の標準イーサネット機能のみを使用します。

1. 時刻同期機能
・IEEE1588-2008 バージョン2 に準拠
・PTP メッセージ（イーサネットフレーム[1]とUDP/IPv4 フォーマット[2]）による時刻同期
・マスタまたはスレーブとして動作し、OC、BC、TC デバイスのいずれにも対応
・時刻偏差の統計的な補正（傾き予測時刻補正アルゴリズム）
・タイマイベント出力（6 チャネル、立ち上がり/立ち下がりエッジ、イベントフラグの自動クリア）
・ELC 経由のタイマイベントで、モータ制御タイマ（MTU3、GPT）の同期スタート
・EPTPCとELC経由で接続したI/OポートからPTPで時刻同期したパルスを出力
・PTP メッセージの処理を選択可能（PTP モジュール内部での処理、PTPEDMAC 経由のCPU 処理、他のポートを使用）
2. 標準イーサネット機能
・独立した2 チャネルのイーサネット動作
・ハードウェア簡易スイッチ（フレームの中継はカットスルー方式またはストア&フォワード方式から選

択可能）

・ハードウェアマルチキャストフレームのフィルタリング（全て受信、全て受信しない、特定の2 種類のフレームのみ受信）

[1] RX64M グループではEthernet II フレームフォーマットのみサポートします（IEEE802.3 フレームフォーマットはサポートしません）。

[2] UDP IPv6 はサポートしません。

詳細に関しては「RX64M/71M グループユーザーズマニュアル ハードウェア編」を参照ください。

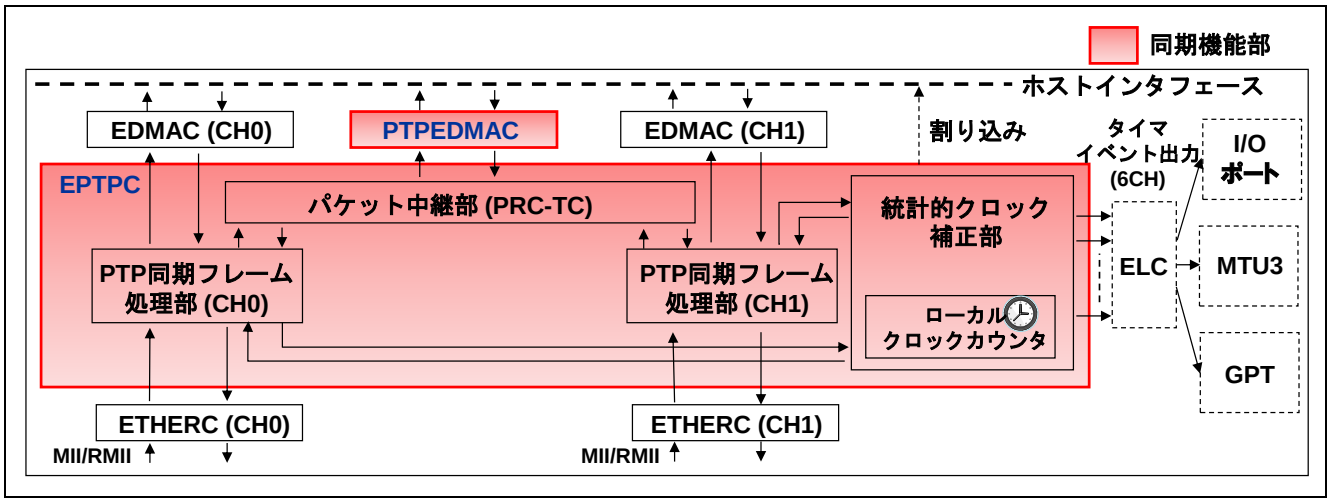


図 1.1　ハードウェアブロック図

1.5　ソフトウェアの構成

PTP 簡易ドライバは常に Ether ドライバ[3]と組み合わせて使用し、TCP/IP システムに適用する場合は、TCP/IP ミドルウェアと組み合わせる必要があります。また、PTP 簡易ドライバは上位層のソフトウェアからの要求に応じ、簡易スイッチ機能やマルチキャストフレームフィルタ機能の設定を行います。ソフトウェアの典型的な構成と機能の概要を図 1.2 に示します。

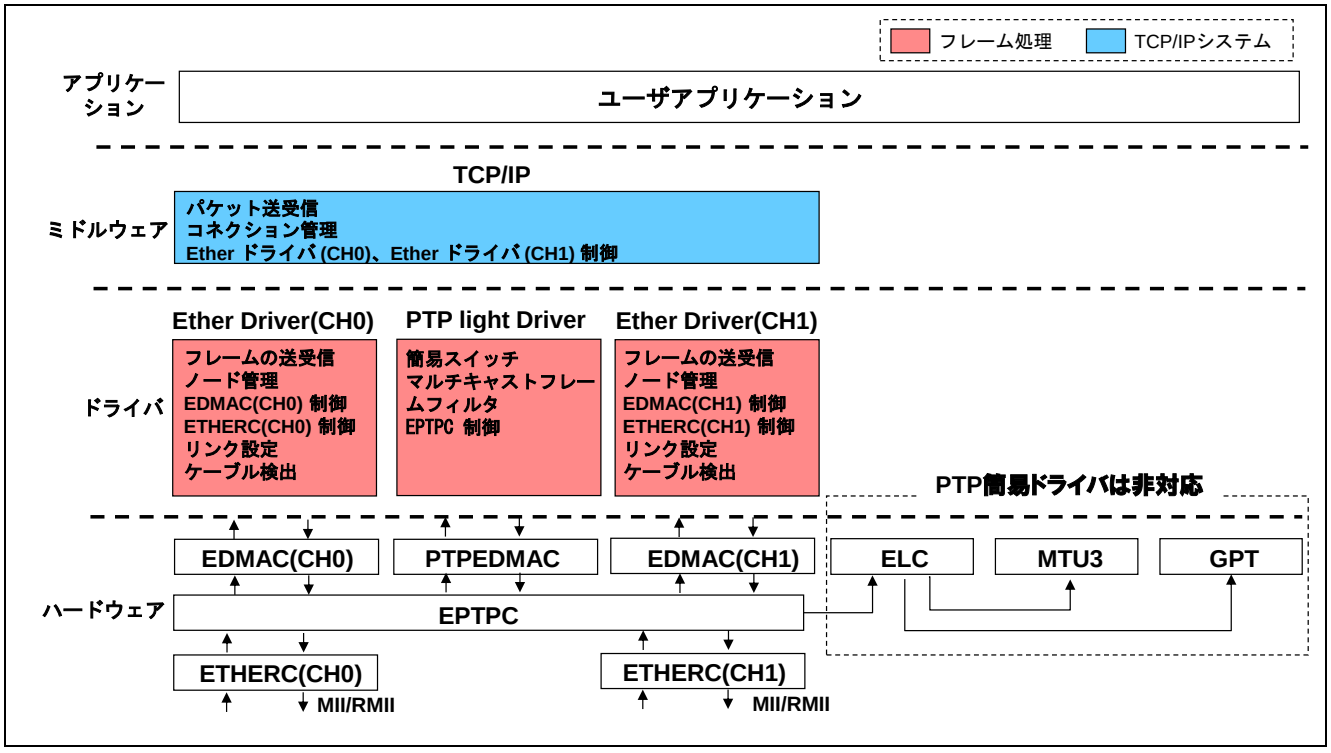


図 1.2　ソフトウェア構成

1.6 ファイル構成

PTP 簡易ドライバは r_ptp_light.c に記述しています。

1.7 API の概要

PTP 簡易ドライバの API 関数の一覧を表 1.1 に示します。

表 1.1 API 関数

関数	内容
R_PTPL_GetVersion()	PTP 簡易ドライバのバージョン番号の取得
R_PTPL_Reset()	EPTPC のリセット
R_PTPL_SetTran()	ポート間の転送モードの設定
R_PTPL_SetMCFilter()	マルチキャストフレーム（MC）のフィルタ（FFLTR）設定
R_PTPL_SetExtPromiscuous()	拡張プロミスキャスモードの設定と解除
R_PTPL_Init()	デバイス構成に従った EPTPC の初期設定
R_PTPL_RegMINTHndr()	EPTPC の MINT 割り込みハンドラへのユーザ関数の登録
R_PTPL_GetSyncConfig()	PTPフレーム制御の構成（SYRFL1R、SYRFL2R、SYTRENR、SYCONFR）取得
R_PTPL_SetSyncConfig()	PTP フレーム制御の構成（SYRFL1R、SYRFL2R、SYTRENR、SYCONFR）設定
R_PTPL_SetInterrupt()	EPTPC のINFABT 要因割り込みの有効化
R_PTPL_ChkInterrupt()	INFABT 割り込み要因の確認
R_PTPL_ClrInterrupt()	INFABT 割り込み要因検出フラグのクリア

2. API 情報

本 FIT モジュールは、下記の条件で動作を確認しています。

2.1 ハードウェアの要件

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。

- EPTPC
- ETHERC
- EDMAC

使用例は「RX ファミリ イーサネットコントローラ：簡易スイッチ動作例 [4]」と「RX ファミリ イーサネットコントローラ：マルチキャストフレームフィルタ動作例[5]」に説明しています。

2.2 ハードウェアリソース要件

ドライバが必要とする周辺回路のハードウェアについて説明します。特に明記しない限り、周辺回路はドライバで制御します。ユーザアプリケーションから直接制御し、使用することはできません。

2.2.1 EPTPC チャネル

ドライバは EPTPC を使用します。この周辺モジュールは、CH0 と CH1 間でのフレームの転送とマルチキャストフレームのフィルタリングに必要です。

2.2.2 ETHERC チャネル

ドライバは ETHEC（CH0）、ETHEC（CH1）、またはその両方を使用します。これらの周辺モジュールはイーサネット MAC 動作に必要です。

2.2.3 EDMAC チャネル

ドライバは EDMAC（CH0）、EDMAC（CH1）、またはその両方を使用します。これらの周辺モジュールは標準イーサネットフレームの送受信での CPU ホストインタフェースとして必要です。

2.3 ソフトウェア要件

ドライバは以下のパッケージ（FIT モジュール）に依存しています。

- r_bsp
- r_ether_rx

2.4 制限事項

ドライバには以下の制限事項があります。

- PTP による時刻同期は対応していません。
- PTP ドライバ（完全版）[2] を同時に使用できません。
- PTP フレームの受信と対応処理はできません [1]。

[1] 転送制御は可能です。

2.5 サポートされているツールチェイン

本 FIT モジュールは「4.5 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。

2.6 使用する割り込みベクタ

R_PTP_Init 関数を実行すると EPTPC MINT 割り込みが有効になります。

表 2.1 に本 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	GROUPAL1 割り込み（ベクタ番号: 113） ● EPTPC MINT 割り込み（グループ割り込み要因番号：0）

2.7　ヘッダファイル

API の呼び出しは、このドライバとともに提供されている r_ptp_light_rx_if.h をインクルードすることで行います。

2.8　整数型

このプロジェクトでは ANSI C99 を使用し、整数型は stdint.h で定義しています。

2.9　コンパイル時の設定

ドライバの構成設定は *r_ptp_light_rx_config.h* で行います。オプション名と設定内容を以下の表に記載します。

構成設定	
#define PTPL_CFG_MODE #define PTPL_CFG_MODE_CH0　(0x01) #define PTPL_CFG_MODE_CH1　(0x02) #define PTPL_CFG_MODE_POLL　(0x10) #define PTPL_CFG_MODE_HWINT (0x20) - Default value = 0x23	PTP簡易ドライバの動作モード設定をします。有効にするチャネルと状態確認の方法を設定します。 • ビット0 を1 に設定すると、チャネル0 が有効になります。 • ビット1 を1 に設定すると、チャネル1 が有効になります。 ビット0とビット1の両方を1 に設定した場合、チャネル0とチャネル1の両方 が有効になります。 • ビット4を1 に設定すると、状態の確認をソフトウェアのポーリングにより行います。このバージョンでは対応していません。 • ビット5を1 に設定すると、状態の確認をEPTPCからの割り込みにより行います。このバージョンではこの値を設定してください。
#define PTPL_CFG_INTERRUPT_LEVEL - Default value = 2	EPTPC の割り込み優先レベルを設定します。 • 設定範囲は 1 から 15 です。

2.10　引数

ドライバの API 関数で使用しているデータ構造について説明します。これらの構造体は r_ptp_light_rx_if.h に API 関数の型宣言と共に定義しています。

2.10.1　定数

```
/* Number of ports */
#define NUM_PORT (2) /* Set 2 in the RX64M/71M */
```

```
/* Inter ports transfer mode */
typedef enum
{
    ST_FOR = 0, /* Store and forward mode (legacy compatible) */
    CT_THR = 1  /* Cut through mode */
} TranMode;
```

```
/* Relay enable directions (bit map form) */
```

```
typedef enum
{
    ENAB_NO = 0x00, /* Prohibit relay */
    ENAB_01 = 0x01, /* Enable CH0 to CH1 */
    ENAB_10 = 0x02, /* Enable CH1 to CH0 */
    ENAB_BT = 0x03  /* Enable CH0 to CH1 and CH1 to CH0 */
} RelEnabDir;
```

```
/* Multicast(MC) frames filter setting */
typedef enum
{
    MC_REC_ALL = 0, /* Receive all MC frames (legacy compatible) */
    MC_REC_NO,      /* Do not receive MC frame */
    MC_REC_REG0,    /* Receive only the MC frame registered FMAC0R(U/L) */
    MC_REC_REG1,    /* Receive only the MC frame registered FMAC1R(U/L) */
} MCRecFil;
```

```
/* MINT interrupt register */
typedef enum
{
    MINT_FUNC_PRC = 0,  /* Interrupt from PRC-TC */
    MINT_FUNC_SYN0,     /* Interrupt from SYNFP0 */
    MINT_FUNC_SYN1,     /* Interrupt from SYNFP1 */
} MINT_Reg;
```

2.10.2 データ型

```
/* Register access structure to SYNFP0 or SYNFP1 part of the EPTPC */
static volatile struct st_eptpc0 _BSP_EVENACCESS_SFR *synfp[2] =
{
    &EPTPC0,
    &EPTPC1,
};
```

```
/* PTP part port related structure */
typedef struct
{
    uint8_t  macAddr[6];
    uint8_t  ipAddr[4];
} PTPLPort;
```

```
/* PTP part configuration structure (port information) */
typedef struct
{
    PTPLPort port[NUM_PORT];
} PTPLConfig;
```

2.11 戻り値

ドライバの API 関数の戻り値を示します。これらの戻り値は r_ptp_light_rx_if.h にプロトタイプ宣言と共に定義しています。

```
/* PTP light driver return value */
typedef enum
{
    PTPL_ERR_TOUT = -3,   /* Timeout error */
    PTPL_ERR_PARAM = -2,  /* Parameter error */
    PTPL_ERR = -1,        /* General error */
    PTPL_OK = 0
} ptpl_return_t;
```

2.12　コールバック関数

本モジュールでは、EPTPC MINT 割り込み発生時にコールバック関数である Eptpc_isr 関数を呼び出します。

2.13　コードサイズ

ROM (コードおよび定数) と RAM (グローバルデータ) のサイズは、ビルド時の「2.9 コンパイル時の設定」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_ptp_light_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 4.08.04.201801
(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	ファイル	使用メモリ		
			Renesas Compiler	GCC	IAR Compiler
RX64M	ROM	r_ptp_light.c	**1222 バイト**	**2392 バイト**	**1842 バイト**
	RAM	r_ptp_light.c	**22 バイト**	**24 バイト**	**22** バイト
	STACK	r_ptp_light.c	**76 バイト**	**-**	**52** バイト
RX72M	ROM	r_ptp_light.c	**1257 バイト**	**2413 バイト**	**1914 バイト**
	RAM	r_ptp_light.c	**22 バイト**	**24 バイト**	**22** バイト
	STACK	r_ptp_light.c	**76 バイト**	**-**	**72** バイト

2.14　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) e² studio 上で Smart Configurator を使用して FIT モジュールを追加する場合
e² 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 e² 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) e² studio 上で FIT Configurator を使用して FIT モジュールを追加する場合
e² 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 e² 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 e² 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。

2.15　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。

「WAIT_LOOP」を記述している対象デバイス

・RX64M グループ
・RX71M グループ
・RX72M グループ

3. API 関数

3.1 R_PTPL_GetVersion ()

PTP 簡易ドライバのバージョン番号を返します。

フォーマット

uint32_t R_PTPL_GetVersion(void);

パラメータ

なし

戻り値

RX_PTPL_VERSION_MAJOR（上位 16 ビット）：メジャーバージョン番号

RX_PTPL_VERSION_MINOR（下位 16 ビット）：マイナーバージョン番号

プロパティ

r_ptp_rx_light_if.h にプロトタイプ宣言しています

説明

PTP 簡易ドライバのメジャーバージョン番号とマイナーバージョン番号を返します。

今回のドライバのバージョン番号は「1.13」です。

- 上位 16 ビットはメジャーバージョン番号を示します。
  RX_PTP_VERSION_MAJOR: 値 = H'1.
- 下位 16 ビットはマイナーバージョン番号を示します。
  RX_PTP_VERSION_MINOR: 値 = H'13.

リエントラント

関数はリエントラントです。

使用例

この関数の使用例を以下に示します。

```
#include <stdio.h>
#include "r_ptp_light_rx_if.h"

  uint32_t ptp_version;

  ptp_version = R_PTPL_GetVersion();

  printf("PTP light driver major version = %d¥n", ptp_light_version >> 16u);
  printf("PTP light driver minor version = %d¥n", ptp_light_version & 0xFFFF);
```

注意

ドライバのバージョンによって、戻り値は異なります。

3.2 R_PTPL_Reset ()

この関数は EPTPC をリセットします。

フォーマット

void R_PTPL_Reset(void);

パラメータ

なし

戻り値

なし

プロパティ

r_ptp_light_rx_if..h にプロトタイプ宣言しています

説明

EPTPC をリセットします。以下の動作を実行します。

- EPTPC をリセットします。
  PTRSTR レジスタの RESET ビットに 1 を書き込みます。
- リセット動作の完了を待ちます。
  PCLKA で 64 サイクル以上
  リセット完了待ちで、R_BSP_SoftwareDelay 関数のループ処理を使用しています。
- EPTPC へのリセットを解除します。
  PTRSTR レジスタの RESET ビットに 0 を書き込みます。
- リセット解除動作の完了を待ちます。
  PCLKA で 256 サイクル以上
  リセット完了待ちで、R_BSP_SoftwareDelay 関数のループ処理を使用しています。

リエントラント

この関数はリエントラントではありません。

使用例

この関数の使用例を以下に示します。

```
/* ==== Ether communication setting ==== */
#define LINK_CH (1) /* 0 or 1(default) */
#define NUM_CH  (2) /* The number of active channel */

/* ==== MAC address ==== */
/* Please change usr own vendor ID */
/* Followings are applied to Renesas vendor ID (= 74-90-50) as sample data */
#define MAC_ADDR_1H (0x00007490)
#define MAC_ADDR_1L (0x50007934)
#define MAC_ADDR_2H (0x00007490)
#define MAC_ADDR_2L (0x50007935)
static uint32_t mac_addr[2][2] = {{MAC_ADDR_1H, MAC_ADDR_1L},{MAC_ADDR_2H,
MAC_ADDR_2L}};

/* ==== IP address ==== */
#define IP_ADDR_1 (0x66676869)
#define IP_ADDR_2 (0x76777879)
static uint32_t ip_addr[2] = {IP_ADDR_1, IP_ADDR_2};
```

```
/* ==== MAC and IP address ==== */
static uint32_t my_mac_addr[2][2] = {{MAC_ADDR_1H, MAC_ADDR_1L},{MAC_ADDR_2H,
MAC_ADDR_2L}};
static uint32_t my_ip_addr[2] = {IP_ADDR_1, IP_ADDR_2};
static uint32_t ether_ch[] = {ETHER_CHANNEL_0, ETHER_CHANNEL_1};

/* ==== PTP message reception filter values ==== */
static uint32_t fil1 = 0x00000000; /* SYRFL1R */
static uint32_t fil2 = 0x00000000; /* SYREL2R */

/* ==== Standard frame received interrupt handler ==== */
extern void EINT_Trig_isr(void *ectrl);

#include "r_ptp_light_rx_if.h"
#include "r_ether_if.h"

   int32_t ch;
   ether_return_t ether_ret;   /* Ether driver return code */
   ether_param_t ectrl;       /* EDMAC and ETHERC control */
   ether_cb_t ecbt;           /* EDMAC callback function structure */
   ptpl_return_t ptpl_ret;    /* PTP light driver return code */

   PTPLConfig ptplc;          /* PTP part configuration structure */
   ether_promiscuous_t pcuous; /* promiscuous control */

   /* Initialize PTP configuration */
   memset(&ptplc, 0, sizeof(ptplc));

   for (ch = 0;ch < NUM_CH;ch++)
   { /* set mac address */
      ptplc.port[ch].macAddr[0] = (uint8_t)(my_mac_addr[ch][0] >> 8u);
      ptplc.port[ch].macAddr[1] = (uint8_t)(my_mac_addr[ch][0]);
      ptplc.port[ch].macAddr[2] = (uint8_t)(my_mac_addr[ch][1] >> 24u);
      ptplc.port[ch].macAddr[3] = (uint8_t)(my_mac_addr[ch][1] >> 16u);
      ptplc.port[ch].macAddr[4] = (uint8_t)(my_mac_addr[ch][1] >> 8u);
      ptplc.port[ch].macAddr[5] = (uint8_t)(my_mac_addr[ch][1]);

      /* set IP address */
      ptplc.port[ch].ipAddr[0] = (uint8_t)(my_ip_addr[ch] >> 24u);
      ptplc.port[ch].ipAddr[1] = (uint8_t)(my_ip_addr[ch] >> 16u);
      ptplc.port[ch].ipAddr[2] = (uint8_t)(my_ip_addr[ch] >> 8u);
      ptplc.port[ch].ipAddr[3] = (uint8_t)(my_ip_addr[ch]);
   }

   /* Initialize resources of the Ether driver */
   R_ETHER_Initial();

   /* Register trigger packet received event to EDMAC interrupt handler */
   ecbt.pcb_int_hnd = EINT_Trig_isr;
   ectrl. ether_int_hnd = ecbt;
   R_ETHER_Control(CONTROL_SET_INT_HANDLER, ectrl);

 /* ==== Open standard Ether ==== */
 #if (1 == LINK_CH)
   for (ch = LINK_CH;ch > (LINK_CH - NUM_CH);ch--)
 #else  /* (0 == LINK_CH) */
   for (ch = 0;ch < NUM_CH;ch++)
 #endif
   { /* Power on ether channel */
      ectrl.channel = ether_ch[ch];
      R_ETHER_Control(CONTROL_POWER_ON, ectrl);
```

```
      /* Initialize EDMAC interface and peripheral modules */
      ether_ret = R_ETHER_Open_ZC2(ch, (const uint8_t*)ptplc.port[ch].macAddr,
ETHER_FLAG_OFF);

      if (ETHER_SUCCESS != ether_ret)
      {
           goto Err_End;
      }
   }

   /* ==== Set PTP configuration ==== */
   /* Reset EPTPC */
   R_PTPL_Reset();

   /* Initialize EPTPC */
   ptpl_ret = R_PTPL_Init(&ptplc);
   if (PTPL_OK != ptpl_ret)
   {
      goto Err_End;
   }

 #if (LINK_CH == 1)
   for (ch = LINK_CH;ch > (LINK_CH - NUM_CH);ch--)
 #else  /* (LINK_CH == 0) */
   for (ch = 0;ch < NUM_CH;ch++)
 #endif
   {
      /* Set promiscuous mode */
      pcuous.channel = ch;
      pcuous.bit = ETHER_PROMISCUOUS_ON;
      ectrl.p_ether_promiscuous = &pcuous;
      R_ETHER_Control(CONTROL_SET_PROMISCUOUS_MODE, ectrl);

      /* Clear extended promiscuous mode */
      R_PTPL_SetExtPromiscuous(ch, false);

      /* Set frame filter */
      R_PTPL_SetSyncConfig(ch, &fil1, &fil2, NULL, NULL);
   }

   /* ==== Link standard Ether ==== */
 #if (LINK_CH == 1)
   for (ch = LINK_CH;ch > (LINK_CH - NUM_CH);ch--)
 #else  /* (LINK_CH == 0) */
   for (ch = 0;ch < NUM_CH;ch++)
 #endif
   {
      while (1)
      { /* Check EDMAC Host interface status */
          ether_ret = R_ETHER_CheckLink_ZC(ch);
          if (ETHER_SUCCESS == ether_ret)
          {
              break;
          }
      }

      /* Set EDMAC interface to transfer standard Ethernet frame */
      R_ETHER_LinkProcess(ch);
   }

   /* Start user operation */
```

注意

この関数は通常、初期化時、またはエラーからの復帰時に実行します。

標準イーサネット MAC のチャネル 0 に最初に（チャネル 1 より以前）アクセスする場合、関連ドキュメント[3]「RX ファミリ イーサネットモジュール Firmware Integration Technology」（= r_ether_rx）の r_ether_rx_config.h で定義されている設定を、デフォルト設定から変更する必要があります。

#define ETHER_CFG_CH0_PHY_ACCESS (0) /* default (1) */

#define ETHER_CFG_CH1_PHY_ACCESS (0) /* default (1) */

3.3 R_PTPL_SetTran ()

ポート間の転送モードを設定します。

フォーマット

ptp_return_t R_PTPL_SetTran(TranMode *mode, RelEnabDir *dir);

パラメータ

mode – ポート間の転送モード

dir – 中継が行われる方向（ビットマップ形式）

戻り値

PTPL_OK: 処理は正常に終了しました。

PTPL_ERR_PARAM: パラメータに誤りがありました。

プロパティ

r_ptp_light_rx_if.h にプロトタイプ宣言しています

説明

ポート間の転送モードを設定します。以下の動作を実行します。

- 中継が行われる方向の有効/無効を設定します。
  CH0 から CH1 への転送の有効/無効の設定
  CH1 から CH0 への転送の有効/無効の設定
- 転送モードを設定します。
  ストア&フォワード方式またはカットスルー方式を選択します。

リエントラント

この関数はリエントラントではありません。

使用例

この関数の使用例を以下に示します。

```
#include "r_ptp_light_rx_if.h"
#include "r_ether_if.h"

/* reception frame MAC address: 01:00:5E:00:01:02 */
#define MAC_ADDR_H (0x00000100)
#define MAC_ADDR_L (0x5E000102)

   ptpl_return_t ret; /* PTP light driver return code */
   TranMode mode; /* Inter ports transfer mode */
   RelEnabDir dir; /* Relay direction */
   uint32_t fmac[2];

   /* Reset EPTPC */
   R_PTPL_Reset();

   /* ==== Clear extended promiscuous mode: CH0 ==== */
   R_PTPL_SetExtPromiscuous(0, false);

   /* Set transfer mode to cut-through mode */
   mode = CT_THR;

   /* Set relay enable both directions */
   dir = ENAB_BT;

   /* ==== Set inter ports transfer mode ==== */
```

```
ret = R_PTPL_SetTran(&mode, &dir);
if (PTPL_OK != ret)
{
   goto Err_end; /* error */
}

/* Set reception frame MAC address */
fmac[0] = ((MAC_ADDR_H << 8u) | (MAC_ADDR_L >> 24u));
fmac[1] = (MAC_ADDR_L & 0x00FFFFFF);

/* ==== Set Multicast (MC) frames filter (FFLTR): CH0 ==== */
/* SYNFP CH0, only receive FMAC1R(U/L) and update FMAC1R(U/L) */
ret = R_PTPL_SetMCFilter(0, MC_REC_REG1, fmac);
if (PTPL_OK != ret)
{
   goto Err_end; /* error */
}

/* Thereafter, frame propagates both directions and cut-through method */
/* Only receives FMAC1R(U/L) registered address frame */
```

注意

この関数による設定は標準イーサネットフレームにのみ有効です。（PTP フレームには有効ではありません。）

引数（mode または dir）が NULL ポインタのときには、その値は設定されません。

3.4 R_PTPL_SetMCFilter ()

この関数はマルチキャスト（MC）フレームのフィルタ（FFLTR）の設定を行います。

フォーマット

ptp_return_t R_PTPL_SetMCFilter(uint8_t ch, MCRecFil fil, uint32_t *fmac);

パラメータ

ch – 同期フレーム処理部のチャネル番号（SYNFP0 または SYNFP1）

fil – マルチキャスト（MC）フレームのフィルタ設定

fmac – 受信フレームの MAC アドレス（FMAC0R(U/L)または FMAC1R(U/L)レジスタ）

戻り値

PTPL_OK: 処理は正常に終了しました。

PTPL_ERR: エラーが発生しました。

プロパティ

r_ptp_light_rx_if.h にプロトタイプ宣言しています

説明

MC フィルタ（FFLTR）の設定を行います。以下の動作を実行します。

- 受信モードを設定します。
  全てのフレームを受信する、フレームを受信しない、または登録したフレームのみ受信のいずれかを設定します。
- 受信 MAC アドレスの更新
  引数 fmac を指定している場合、引数 fil に応じ、FMAC0R (U/L)レジスタまたは FMAC1R (U/L)レジスタを更新します。

リエントラント

この関数はリエントラントではありません。

使用例

この関数の使用例は「3.3 R_PTPL_SetTran ()」を参照ください。

注意

この関数による設定は標準イーサネットフレームにのみ有効です。PTP フレームには「3.9 R_PTPL_SetSyncConfig ()」を使用ください。

マルチキャストフィルタ機能は拡張プロミスキャスモードでは無効です。マルチキャストフィルタ機能を使用する場合、「3.3 R_PTPL_SetTran ()」の使用例に示すように拡張プロミスキャスモードをクリアしてください。

3 番目の引数（fmac）が NULL ポインタのときには、FMAC0R (U/L)レジスタと FMAC1R (U/L)レジスタはともに更新されません。

3.5 R_PTPL_SetExtPromiscuous ()

拡張プロミスキャスモードを設定/解除します。

フォーマット

ptp_return_t R_PTPL_SetExtPromiscuous(uint8_t ch, bool is_set);

パラメータ

ch – 同期フレーム処理部のチャネル番号（SYNFP0 または SYNFP1）

is_set - (true): 設定、(false): クリア

戻り値

PTPL_OK: 処理は正常に終了しました。

PTPL_ERR_PARAM: パラメータに誤りがありました。

プロパティ

r_ptp_light_rx_if.h にプロトタイプ宣言しています

説明

拡張プロミスキャスモードの設定/クリアを行います。

リエントラント

この関数はリエントラントではありません。

使用例

この関数の使用例を以下に示します。

```
#include "r_ptp_light_rx_if.h"
#include "r_ether_if.h"

   /* Set full functionality used case */
   int32_t ch; /* Ethernet and SYNC unit channel */
   ether_promiscuous_t pcuous; /* promiscuous control */
   ether_param_t ectrl; /* ETHERC control */

   for (ch = 0; ch < 2; ch++)
   {
      /* ==== Set promiscuous mode (CH0 and CH1) ==== */
      pcuous.channel = ch;
      pcuous.bit = ETHER_PROMISCUOUS_ON;
      ectrl. ether_promiscuous_t = &pcuous;
      R_ETHER_Contorl(CONTROL_SET_PROMISCUOUS_MODE, ectrl);

      /* ==== Clear extended promiscuous mode (CH0 and CH1) ==== */
      R_PTPL_SetExtPromiscuous(ch, false);
   }

   /* Thereafter, frame operations are followed */
   /* - Unicast
        coincidence address is EDMAC and PRC-TC,
        and other frames is only PRC-TC
      - Multicast
        depends on multicast frame filter
      - Broadcast
        EDMAC and Packet relation control */
```

注意

この関数による拡張プロミスキャスモード設定の結果に関しては、4.4 節を参照ください。

3.6 R_PTPL_Init ()

この関数はデバイスの構成に従い、EPTPC の初期設定を行います。

フォーマット

ptp_return_t R_PTPL_Init(PTPConfig *tbl);

パラメータ

tbl – PTP 構成設定テーブル

戻り値

PTPL_OK: 処理は正常に終了しました。

PTPL_ERR: エラーが発生しました。

プロパティ

r_ptp_light_rx_if.h にプロトタイプ宣言しています

説明

デバイスの構成に従い、EPTPC の初期設定を行います。以下の動作を実行します。

- RX72M の場合、バイパス機能を解除します。
- MAC アドレスと IP アドレスを設定します。
- PTP フレーム受信フィルタ（SYRFL1R と SYRFL2R）を設定します。
- 同期フレーム処理部（SYNFP0 と SYNFP1）の設定値を有効にします。
- パケット中継部（PRC-TC）を初期化します。
- EPTPC の割り込み状態と割り込みマスクをクリアします。
- EPTPC 割り込みの設定を行います。
  ptp_dev_start 関数を呼び出します。

リエントラント

この関数はリエントラントではありません。

使用例

この関数の使用例は「3.2 R_PTPL_Reset ()」に説明しています。

注意

この関数は ETHERC と標準イーサネットの EDMAC を起動後、1 度だけ実行する必要があります。

チャネル毎のパラメータ設定で、for 文（ループ処理）を使用しています。

3.7 R_PTPL_RegMINTHndr ()

この関数は EPTPC の MINT 割り込みを設定し、MINT 割り込みハンドラにユーザ関数を登録します。

フォーマット

void R_PTPL_RegMINTHndr(MINT_Reg reg, unit32_t event, MINT_HNDLR func);

パラメータ

reg – MINT 割り込みレジスタ
MINT_FUNC_PRC: PRC-TC からの割り込み
MINT_FUNC_SYN0: SYNFP0 からの割り込み
MINT_FUNC_SYN1: SYNFP1 からの割り込み
event – 割り込み要因
func – 登録されるユーザ関数

戻り値

なし

プロパティ

r_ptp_light_rx_if.h にプロトタイプ宣言しています

説明

EPTPC の MINT 割り込みを設定し、MINT 割り込みハンドラにユーザ関数を登録します。次の動作が実行されます。

- 割り込み要因が PRC-TC の場合、PRC-TC から呼ばれる割り込みハンドラにユーザ関数を登録します。
- 割り込み要因が SYNFP0/1 の場合、SYNFP0/1 から呼ばれる割り込みハンドラにユーザ関数を登録します。
- MINT 割り込みの設定または解除をします。

リエントラント

この関数はリエントラントではありません。

使用例

この関数の使用例を以下に示します。

```
#include "r_ptp_light_rx_if.h"

    /* user MINT handler function */
    void user_mint_func(uint32_t reg);

    /* register user function called by INTCHG (logMessageInterval updated)
event of SYNFP0 */
    R_PTPL_RegMINTHndr(MINT_FUNC_SYN0, 0x00000002, (MINT_HNDLR)user_mint_func);

    /* wait interrupt from SYNFP0 */

    /* interrupt occurred (call user_mint_func) */

    /* Update transmission interval of Delay_Req message to suitable interval */

    /* release registered user read PTP message function */
    R_PTPL_RegMINTHndr(MINT_FUNC_SYN0, 0x00000002, (MINT_HNDLR)NULL);

    return;
```

注意

同じ要因の MINT 割り込みハンドラに関数を登録する場合、既に登録されているユーザ関数は更新されます。

引数（func）が NULL ポインタのときには、既に登録されているユーザ関数は解除され、同じ要因の MINT 割り込みは無効になります。

3.8 R_PTPL_GetSyncConfig ()

PTP フレーム制御の構成（SYRFL1R、SYRFL2R、SYTRENR および SYCONFR）取得を行います。

フォーマット

ptp_return_t R_PTPL_GetSyncConfig(uint8_t ch, uint32_t *fil1, uint32_t *fil2, uint32_t *tren, uint32_t *conf);

パラメータ

ch – 同期フレーム処理部のチャネル番号（SYNFP0 または SYNFP1）

fil1 –SYRFL1R

fil2 –SYRFL2R

tren[1] – SYTRENR.

conf[1] –SYCONFR

[1] これらの引数は標準イーサネットフレームでは無効です。

戻り値

PTPL_OK: 処理は正常に終了しました。

PTPL_ERR: エラーが発生しました。

プロパティ

r_ptp_light_rx_if.h にプロトタイプ宣言しています

説明

PTP フレーム制御構成（SYRFL1R、SYRFL2R、SYTRENR および SYCONFR）を取得します。以下の動作を実行します。

- SYNFP 受信フィルタレジスタ 1（SYRFL1R）の値を取得します。
  Announce、Sync、Follow_Up、Delay_Req、Delay_Resp、Pdelay_Req、Pdelay_Resp および Pdelay_Resp_Follow_Up の各メッセージに対するフィルタ設定
- SYNFP 受信フィルタレジスタ 2（SYRFL2R）の値を取得します。
  Management、Signaling、および ilegal（定義外）の各メッセージに対するフィルタ設定
- SYNFP 送信許可レジスタ（SYTRENR）の値を取得します。
  Announce、Sync、Delay_Req および Pdelay_Req の各メッセージに対する送信設定
- SYNFP 動作設定レジスタ（SYCONFR）の値を取得します。
  TC モード（E2E TC または P2P TC）と送信間隔の設定

リエントラント

関数はリエントラントです。

使用例

この関数の使用例を以下に示します。

```
#include <stdio.h>
#include "r_ptp_light_rx_if.h"

    ptpl_return_t ret; /* PTP light driver return code */
    uint32_t fil1; /* Current SYRFL1R (CH0) value */
    uint32_t fil2; /* Current SYRFL2R (CH0) value */

    /* Get SYRFL1R, SYRFL2R and SYTRENR from SYNFP0. (Not get the SYCONFR) */
    ret = R_PTPL_GetSyncConfig(0, &fil1, &fil2, NULL, NULL);
    if (PTPL_OK != ret)
```

```
    {
        goto Err_end; /* error */
    }

    printf("SYRFL1R (CH0) value = %8x¥n", fil1);
    printf("SYRFL2R (CH0) value = %8x¥n", fil2);
```

注意

引数（fil1、fil2、tren または conf）が NULL ポインタのときには、その引数に対応する値は返しません。

3.9 R_PTPL_SetSyncConfig ()

PTP フレーム制御の構成（SYRFL1R、SYRFL2R、SYTRENR および SYCONFR）を設定します。

フォーマット

ptp_return_t R_PTPL_SetSyncConfig(uint8_t ch, uint32_t *fil1, uint32_t *fil2, uint32_t *tren, uint32_t *conf);

パラメータ

ch – 同期フレーム処理部のチャネル番号（SYNFP0 または SYNFP1）

fil1 –SYRFL1R

fil2 –SYRFL2R

tren[1] – SYTRENR.

conf[1] –SYCONFR

[1] これらの引数は標準イーサネットフレームでは無効です。

戻り値

PTPL_OK: 処理は正常に終了しました。

PTPL_ERR: エラーが発生しました。

プロパティ

r_ptp_light_rx_if.h にプロトタイプ宣言しています

説明

PTP フレーム制御構成の設定（SYRFL1R、SYRFL2R、SYTRENR および SYCONFR）を行います。以下の動作を実行します。

- SYNFP 受信フィルタレジスタ 1（SYRFL1R）の値を設定します。
  Announce、Sync、Follow_Up、Delay_Req、Delay_Resp、Pdelay_Req、Pdelay_Resp および Pdelay_Resp_Follow_Up の各メッセージに対するフィルタ設定
- SYNFP 受信フィルタレジスタ 2（SYRFL2R）の値を設定します。
  Management、Signaling、および illegal（定義外）の各メッセージに対するフィルタ設定
- SYNFP 送信許可レジスタ（SYTRENR）の値を設定します。
  Announce、Sync、Delay_Req および Pdelay_Req の各メッセージに対する送信設定
- SYNFP 動作設定レジスタ（SYCONFR）の値を設定します。
  TC モード（E2E TC または P2P TC）と送信間隔の設定

リエントラント

関数はリエントラントではありません。

使用例

この関数の使用例を以下に示します。

```
#include "r_ptp_light_rx_if.h"

   ptpl_return_t ret; /* PTP light driver return code */
   uint32_t fil1; /* SYRFL1R setting value, 0:SYNFP0 1:SYNFP1 */
   uint32_t fil2; /* SYRFL2R setting value, 0:SYNFP0 1:SYNFP1 */

   /* Transfer all PTP messages to the other channel */
   fil1 = 0x22222222;

   fil2 = 0x00000022;

   /* Set fil1_val and fil2_val to SYNFP1. (Neither set the SYTRENR nor
SYCONFR) */
```

```
ret = R_PTPL_SetSyncConfig(1, &fil1, &fil2, NULL, NULL);
if (PTPL_OK != ret)
{
    goto Err_end; /* error */
}

/* Complete set synchronous configuration */
```

注意

中継有効ビット（bit[4N+1] (N=1, 2,,, 7)）のみが設定可能です[1]。他のビットは 0 にクリアしてください。

引数（fil1、fil2、tren または conf）が NULL ポインタのときには、その引数に対応する値は設定しません。

[1] このドライバでは PTP フレームの処理は中継のみ可能です。

3.10 R_PTPL_SetInterrupt ()

EPTPC の INFABT（制御情報異常検出フラグ）要因割り込みを有効にします。

フォーマット

ptp_return_t R_PTPL_SetInterrupt(uint8_t ch);

パラメータ

ch – 同期フレーム処理部のチャネル番号（SYNFP0 または SYNFP1）

戻り値

PTPL_OK: 処理は正常に終了しました。

PTPL_ERR_PARAM: パラメータに誤りがありました。

プロパティ

r_ptp_light_rx_if.h にプロトタイプ宣言しています

説明

EPTPC の INFABT（制御情報異常検出フラグ）要因割り込みを有効にします。以下の動作を実行します。

- SYNFP0/1 からの割り込みを EPTPC の MIEIPR レジスタの設定で有効にする。
- SYNFP0/1.SYSR の INFABT 要因割り込みを EPTPC0/1 の SYIPR レジスタの設定で有効にする。

リエントラント

この関数はリエントラントではありません。

使用例

この関数の使用例を以下に示します。

```
#include "r_ptp_light_rx_if.h"
#include "r_ether_if.h"

   ptpl_return_t ret;   /* PTP light driver return code */
   bool is_det;         /* INFABT interrupt detection flag */

   /* Standard Ethernet open and link were completed */

   /* PTP open was completed */

   /* Enable EPTPC INFABT interrupt CH0 */
   ret = R_PTPL_SetInterrupt(0);
   if (PTPL_OK != ret)
   {
      goto Err_end; /* error */
   }

   while(1)
   {
      ret = R_PTPL_ChkInterrupt(0, &is_det);
      if (PTPL_OK != ret)
      {
         goto Err_end; /* error */
      }

      /* Check INFABT error */
      if (true == is_det)
      { /* INFABT error detected */
          /* stop standard Ether */
         R_ETHER_Close_ZC2(0);
```

```
            /* Reset EPTPC */
            R_PTPL_Reset();

            /* Clear INFABT interrupt flag */
            R_PTPL_ClrInterrupt(0);

            /* Thereafter, please execute retrieve operation */
        }
    }
```

注意

なし

3.11 R_PTPL_ChkInterrupt ()

EPTPC の INFABT 割り込み要因の確認を行います。

フォーマット

ptp_return_t R_PTPL_ChkInterrupt(uint8_t ch, bool *is_det);

パラメータ

ch – 同期フレーム処理部のチャネル番号（SYNFP0 または SYNFP1）

is_det – INFABT 割り込み要因発生フラグ

戻り値

PTPL_OK: 処理は正常に終了しました。

PTPL_ERR_PARAM: パラメータに誤りがありました。

プロパティ

r_ptp_light_rx_if.h にプロトタイプ宣言しています

説明

EPTPC の INFABT 割り込み要因の確認を行います。

リエントラント

この関数はリエントラントです。

使用例

この関数の使用例は「3.10 R_PTPL_SetInterrupt ()」を参照ください。

注意

なし

3.12 R_PTPL_ClrInterrupt ()

EPTPC の INFABT 割り込み要因検出フラグのクリアを行います。

フォーマット

ptp_return_t R_PTPL_ClrInterrupt(uint8_t ch);

パラメータ

ch – 同期フレーム処理部のチャネル番号（SYNFP0 または SYNFP1）

戻り値

PTPL_OK: 処理は正常に終了しました。

PTPL_ERR_PARAM: パラメータに誤りがありました。

プロパティ

r_ptp_light_rx_if.h にプロトタイプ宣言しています

説明

EPTPC の INFABT 割り込み要因検出フラグのクリアを行います。

リエントラント

この関数はリエントラントではありません。

使用例

この関数の使用例は「3.10 R_PTPL_SetInterrupt ()」を参照ください。

注意

なし

4. 付録

4.1 内部関数

PTP 簡易ドライバは動作時に下記の内部関数を呼び出しています。表 4.1 に内部関数の一覧を示します。

表 4.1 内部関数

関数	内容
_R_PTPL_Int_Syn0()	SYNFP0 の INFABT 割り込みフラグを設定
_R_PTPL_Int_Syn1()	SYNFP1 の INFABT 割り込みフラグを設定
_R_PTPL_Init_SYNFP()	SYNFP のパラメータを初期値で設定
_R_PTPL_Init_PRC()	PRC-TC のパラメータを初期値で設定

4.2 関連する Ether ドライバの API

PTP 簡易ドライバは Ether ドライバと組み合わせて使用してください。PTP 簡易ドライバが使用する Ether ドライバの API の情報に関しては、「RX ファミリ イーサネットモジュール Firmware Integration Technology」（関連ドキュメント[3]）を参照してください。これらの API の典型的な使い方は、実例として 3 章に記載しています。

4.3 標準イーサネットの拡張機能

PTP 簡易ドライバは PTP による時刻同期は持ちませんが、簡易スイッチ機能[1]とマルチキャストフレームフィルタ機能に対応しています。「3.3 R_PTPL_SetTran ()」は簡易スイッチ機能を、「3.4 R_PTPL_SetMCFilter ()」はマルチキャストフレームフィルタ機能の設定を行うものです。これら機能の概要を以下に説明します。

[1] これはハードウェアによるポート間の転送を意味します。

4.3.1 簡易スイッチ（PRC-TC に実装）

ストア&フォワードまたはカットスルー転送方式を選択し使用できます。特に産業用ネットワークで一般的なディジーチェーン接続にカットスルー転送方式を適用した場合、ポート間の転送における遅延を短縮でき、有効な機能となります。

また、ネットワークをチャネル毎に分離することで、独立した 2 個のネットワークとして使用することもできます。

4.3.2 マルチキャストフレームフィルタ（SYNFP0 および SYNFP1 に実装）

不要なマルチキャストフレームの受信を省略することで、システムとしての性能を向上させることができます。また、フィルタを有効にした場合でも、特定の 2 種類のフレームを受信することができます。

PTP フレームには、固有のフィルタ（SYRFL1R/2R）を別に実装しています。このフィルタの設定に関しては「3.9 R_PTPL_SetSyncConfig ()」を参照ください。

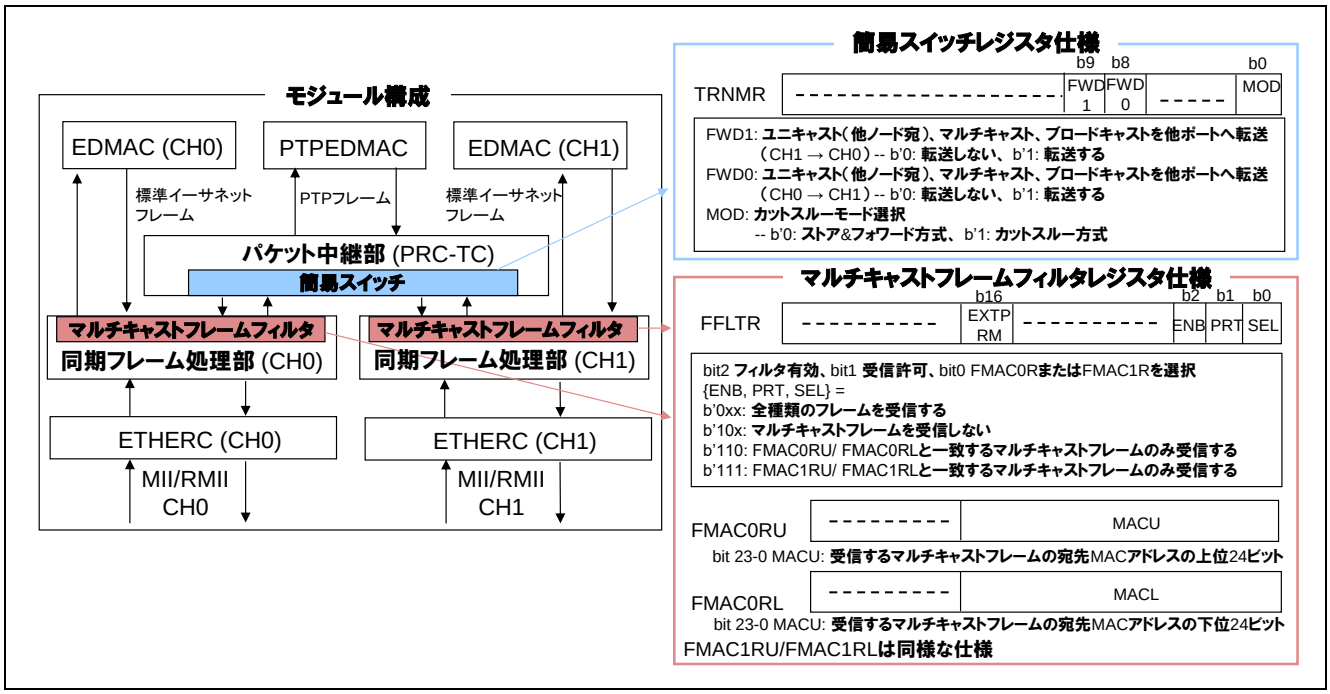
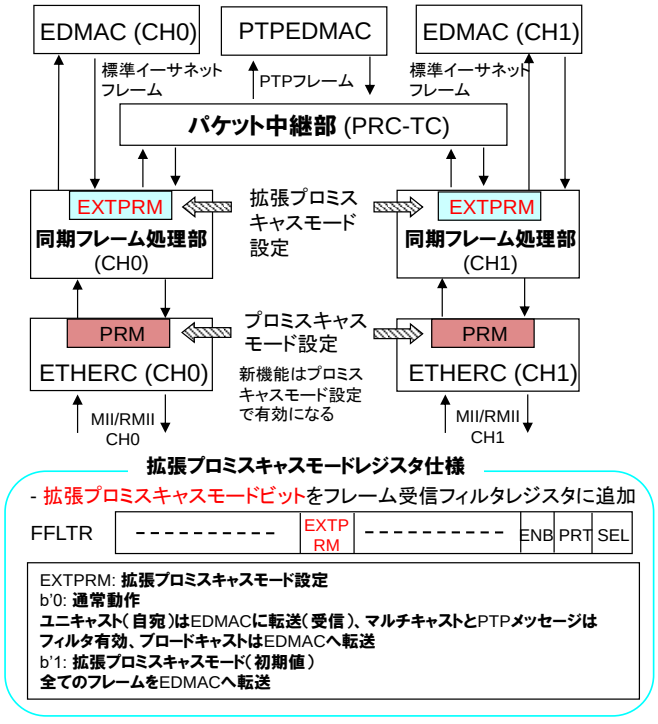


図 4.1 標準イーサネットの拡張機能

4.4　既存のデバイスとの互換性

RX64M/71M のイーサネットモジュールは SH7216 や RX63N 等の既存のルネサス製品と互換性があります。互換性はプロミスキャスモード（PRM）と拡張プロミスキャスモード（EXTPRM）の組み合わせで対応しています。プロミスキャスモードは R_ETHER_Control 関数のコントロールコード CONTROL_SET_PROMISCUOUS_MODE で設定することができます。詳細は「3.5 R_PTPL_SetExtPromiscuous ()」を参照してください。これらモードの設定に関しては、「RX ファミリ イーサネットモジュール Firmware Integration Technology」（関連ドキュメント[3]）を参照ください。拡張プロミスキャスモードは「3.5 R_PTPL_SetExtPromiscuous ()」で設定します。図 4.2 に設定の組み合わせを示します。



モードによるフレーム転送先

b'1: 設定、b'0: 解除　　PTPフレームの場合は省略

PRM	EXTPRM	フレーム種別	転送先
b'0	b'0	ユニキャスト	自宛: EDMAC 他宛: 受信しない
		マルチキャスト	マルチキャストフレームフィルタに従う
		ブロードキャスト	EDMACとPRC-TC
b'0	b'1	ユニキャスト	自宛: EDMAC 他宛: 受信しない
		マルチキャスト	EDMAC
		ブロードキャスト	EDMAC
b'1	b'0	ユニキャスト	自宛: EDMAC 他宛: PRC-TC
		マルチキャスト	マルチキャストフレームフィルタに従う
		ブロードキャスト	EDMACとPRC-TC
b'1	b'1	ユニキャスト	自宛: EDMAC 他宛: EDMAC
		マルチキャスト	EDMAC
		ブロードキャスト	EDMAC

既存デバイスと完全互換（SH7216、RX63N等）

新機能の全てを使用

図 4.2　プロミスキャスモードと拡張プロミスキャスモードの設定

4.5 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 4.2 動作確認環境

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio V7.2.0 IAR Embedded Workbench for Renesas RX 4.12.1
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 4.08.04.201801 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 4.12.1 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	Rev.1.13
使用ボード	Renesas Starter Kit+ for RX64M Renesas Starter Kit+ for RX71M Renesas Starter Kit+ for RX72M

4.6 トラブルシューティング

本 FIT モジュールの動作確認環境を以下に示します。

(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)」
- e² 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 e² 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_ptp_light_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、ビルド実行するとコンフィグ設定が間違っているエラーが発生します。

A：“r_ptp_light_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_ptp_light_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「2.9 コンパイル時の設定」を参照してください。

5. 提供されているモジュール

提供されているモジュールは、ルネサス エレクトロニクスのウェブサイトからダウンロードすることができます。

6. 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 Rev.1.10（R01UH0377JJ）
RX71M グループユーザーズマニュアル ハードウェア編 Rev.1.10（R01UH0493JJ）
RX72M グループユーザーズマニュアル ハードウェア編 Rev.1.00（R01UH0804JJ）
最新版はルネサス エレクトロニクスのウェブサイトからダウンロードできます。

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 CC-RX コンパイラ ユーザーズマニュアル（R20UT3248）
（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の対応について

本モジュールは以下のテクニカルアップデートの内容を反映しています。

- TN-RX*-A125A/J

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2015.10.30	—	初版発行
1.10	2016.03.31	—	データ構造変更
1.11	2016.11.11	10	バージョン取得関数の内部処理を修正
		30	モードによるフレーム転送先を修正（図 4.2 の記述）
1.12	2019.07.31	—	GNUC と ICCRX に対応
		—	MINT 割り込みハンドラ処理を変更
		4, 21	MINT 割り込みハンドラにユーザ関数を登録する関数を追加
		5	使用する割り込みベクタの説明を追加
		7	INT_Reg 定数を追加
		8	コールバック関数の説明を追加
		8	コードサイズの更新と GNUX と ICCRX の場合を追加
		8	FIT モジュールの追加方法を更新
		9	ループ処理のコメントに関する説明を追加
		11	リセット完了待ちで R_BSP_SoftwareDelay 関数を使用するように変更
		11	R_PTPL_Reset 関数にリセット解除待ち動作を追加
		20	チャネル毎のパラメータ設定で「WAIT_LOOP」コメントを追加
		34	「動作確認環境」の節を追加
		34	「トラブルシューティング」の節を追加
1.13	2019.08.31	—	RX72M に対応
		—	バイパス設定を追加

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。