本ドキュメントは Cypress (サイプレス) 製品に関する情報が記載されております。本ドキュメントには、「MB」から始まるシリーズ名，品名およびオーダ型格が記載されておりますが、これらはすべて「CY」から始まるシリーズ名，品名およびオーダ型格として、新規および既存のお客様に引き続き提供してまいります。

オーダ型格の調べ方について

1. www.cypress.com/pcnにアクセスしてください。
2. SEARCH PCNS フィールドに、オーダ型格などのキーワードを入力し、「Apply」をクリックしてください。
3. 該当するタイトル(Title)をクリックしてください。
4. 「Affected Parts List」ファイルを開いてください。
   当該ファイルに記載されている各種変更情報をご利用ください。

詳しいお問い合わせ先

Cypress 製品およびそのソリューションの詳細につきましては、お近くの営業所へお問い合わせください。

サイプレスについて

サイプレスは、世界で最も革新的な車載や産業機器、スマート家電、民生機器および医療機器製品向けに、最先端の組み込みシステム ソリューションを提供するリーディング カンパニーです。サイプレスのマイクロコントローラーや、アナログ IC、ワイヤレスおよび USB ベースのコネクティビティ ソリューション、高い信頼性と高性能を提供するメモリ製品は、各種機器メーカーの差異化製品の開発と早期市場参入を支援します。サイプレスは、ベストクラスのサポートと開発リソースをグローバルに提供することで、彼らが従来市場を破壊しまったく新しい製品カテゴリを歴史的なスピードで市場投入できるよう支援します。詳細はサイプレスのウェブサイト (japan.cypress.com) をご覧ください。

AN204417

MB9BFXXX, S6E2CC, S6E2G イーサネットドライバマニュアル

本イーサネットドライバは FM3 や FM4 デバイスなどの FM ファミリ向けの低レベルドライバです。ドライバ単体もしくはサイプレス PDL(Peripheral Drivers Library)として統合したものを利用できます。

Contents

1 イントロダクション 1
2 イーサネットドライバ構造.....1
2.1 ドライバのファイル構成.....1
2.2 サンプルファイル(ドライバ単体の場合).....2
2.3 ドライバシステム 2
2.4 割込みフロー 3
3 イーサネットドライバの使用方法.....4
3.1 emac_user.h でのユーザ設定 4
3.2 *emac.c* での追加設定 12
3.3 サンプルの使用方法.....13
4 イーサネットドライバのコンフィグレーションとデータ型 16
4.1 はじめに 16
4.2 コンフィグレーションの構造体.....16
4.3 リンクステータスの enum 型 16
4.4 リンクモードの enum 型.....16
5 イーサネットドライバ API.....17
5.1 Ethernet MAC functions.....17
5.2 Ethernet PHY functions.....24
6 追加情報.....25
7 改訂履歴.....26
ワールドワイドな販売と設計サポート 27

1 イントロダクション

本イーサネットドライバは FM3 や FM4 デバイスなどの FM ファミリ向けの低レベルドライバです。ドライバ単体もしくはサイプレス PDL(Peripheral Drivers Library)として統合したものを利用できます。

2 イーサネットドライバ構造

ここではイーサネットドライバの構造を説明します。

2.1 ドライバのファイル構成

イーサネットドライバは、*emac* フォルダに下記のファイルを格納しています。

emac.c	イーサネット API
emac.h	イーサネット API のヘッダファイル
emac_user.h	イーサネットドライバのユーザ設定ファイル
emac_user.c	特定 PHY のためのユーザ提供によるコールバック関数
EmacDescription.h	イーサネットドライバの詳細な変更ログ。ドキュメントのみ

2.2 サンプルファイル(ドライバ単体の場合)

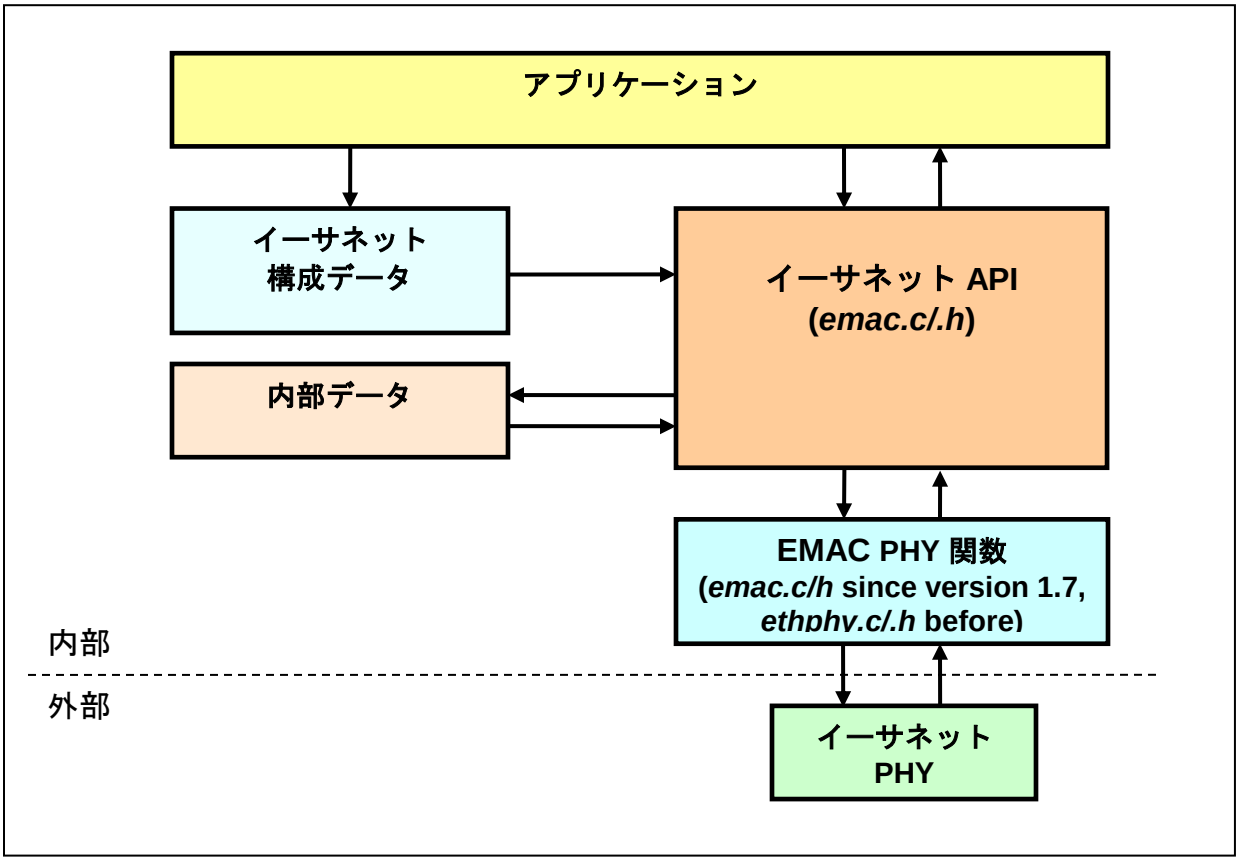
イーサネットドライバの使用例を示すために、ソフトウェアパッケージには *main.c* モジュール以外に評価ボードの動作のためのファイルがあります。

ライブラリ内のファイル	ハードウェアドライバ
scheduler.c ,	循環機能用のシンプルなスケジューラ関数
scheduler.h	スケジューラ関数のヘッダファイル
stacklessudp.c	デモ用にローカルネットワークでデータを送信するためのフル機能の TCP / IP スタックなしで UDP パケットを作成する関数
stacklessudp.h	UDP 関数のヘッダファイル
tasks.c	スケジューラによって実行するタスク関数
tasks.h	タスク関数のプロトタイプ

2.3 ドライバシステム

下記の図はイーサネット MAC を 1 つ使用した場合のイーサネットドライバの動作を表しています。

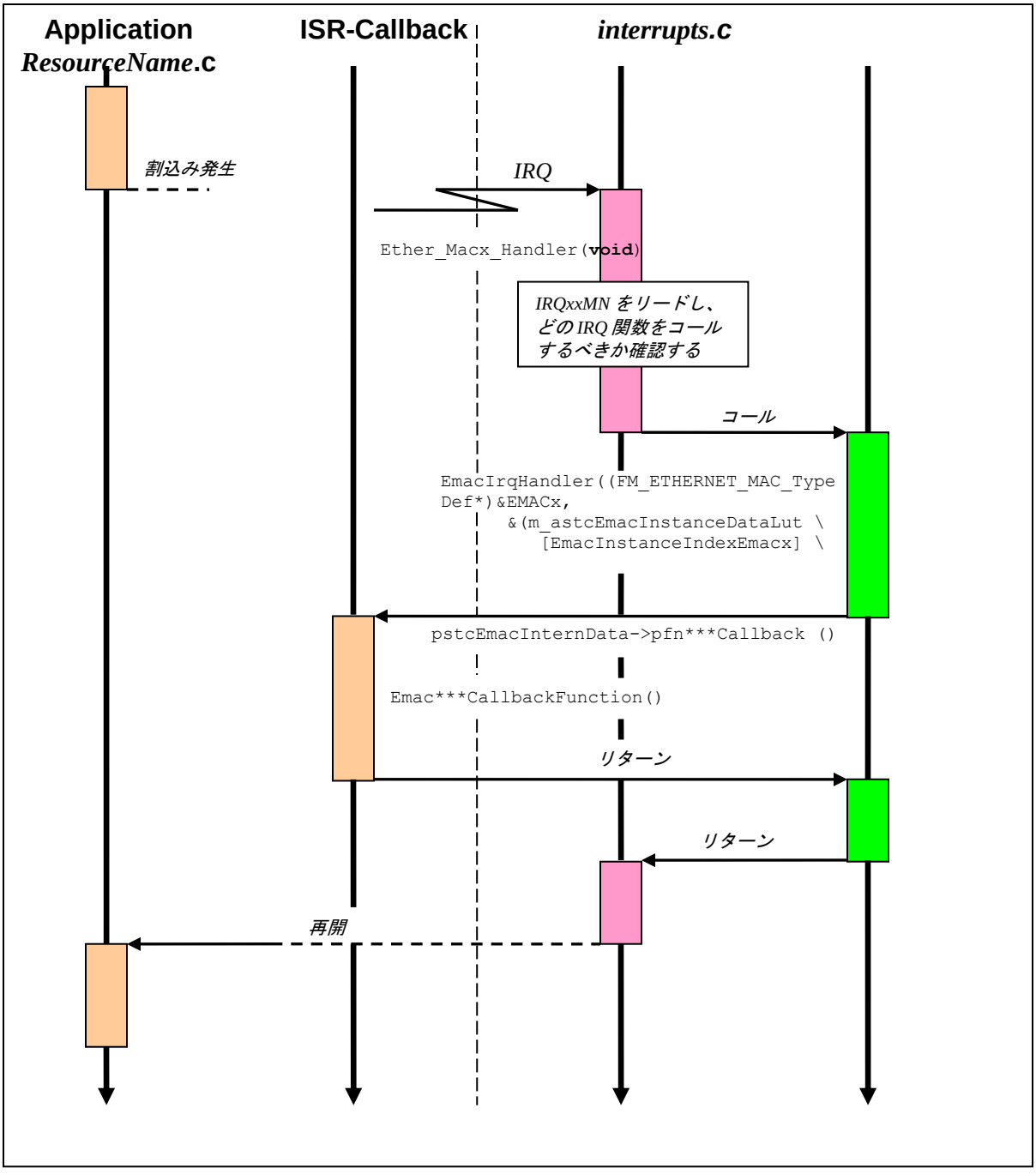
Figure 1. ドライバ構造の例



2.4 割込みフロー

下記のフローは L3 ハンドラがどのように割込まれるかを表しています。

Figure 2. 割込みとコールバックのフロー



3 イーサネットドライバの使用方法

イーサネットドライバのセットアップ方法を説明します。

3.1 emac_user.h でのユーザ設定

emac_user.h ファイルは、コンパイルする際に必要となるイーサネットドライバの構成を設定できます。システム実行後にはコールバック関数を設定できます。これは 4.2 節に記載しています。

3.1.1 PDL integration

サイプレスが提供する PDL(Peripheral Drivers Library)と併用または独立してイーサネットドライバを使用できます。これは PDL_ON や PDLOFF といったパラメータを使用するようドライバが設計されているためです。スタンドアローンでドライバを使用する場合は、必ずこれらのパラメータを設定してください。これは最初のオプション設定で定義します。

```
/**
 ******************************************************************************
 ** \brief Should Cypress Peripheral Drivers Library be used or not
 **
 ** Possible definitions are 1 or 0
 **
 ** 0: Use driver stand-alone
 ** 1: Use driver together with PDL
 **
 ******************************************************************************/
#define EMAC_USE_PDL        0
```

3.1.2 イーサネットの起動

PDL_ON を設定することによりイーサネット MAC を有効にします。使用する FM ファミリに 2 つのイーサネット MAC がついている場合は、別々に設定します。

```
/**
 ******************************************************************************
 ** \brief User Defines for EMAC resource enable
 **
 ** Possible definitions are PDL_ON and PDL_OFF.
 ******************************************************************************/
#define PDL_PERIPHERAL_ENABLE_EMAC0  PDL_ON
#define PDL_PERIPHERAL_ENABLE_EMAC1  PDL_ON
```

3.1.3 割込みモード

イーサネット IRQ サポートを使用する場合は、PDL_ON を設定してください。割込み発生後は ETHER_MACx_IRQHandler()がコールされます。

```
/**
 ******************************************************************************
 ** \brief Activate IRQ support for Ethernet driver
 **
 ** Possible definitions are PDL_ON and PDL_OFF.
 **
 ******************************************************************************/
#define EMAC_INTERRUPT_MODE     PDL_OFF
```

3.1.4 割込み優先度

割込みの優先度はイーサネット MAC それぞれに設定できます。

```
/**
 ******************************************************************************
 ** \brief User Emac Interrupt level settings
 **
 ** Possible values are 0 (high priority) to 15 (low priority)
 ******************************************************************************/
#define PDL_IRQ_LEVEL_EMAC0             2
#define PDL_IRQ_LEVEL_EMAC1             4
```

3.1.5 バッファの位置(EMAC0 の例)

ユーザはバッファを、ドライバ内部もしくはアプリ側で持つかを選択できます。PDL_ON を設定することで、ドライバのメモリ空間内部にバッファを持ちます。

```
/**
 ******************************************************************************
 ** \brief EMAC0 User/Driver Buffer location
 **
 ** PDL_ON:  Driver buffer are used
 ** PDL_OFF: User has to provide buffer memory
 ******************************************************************************/
#define EMAC0_BUFFERS_IN_DRIVERSPACE PDL_ON
```

3.1.6 送受信リングサイズ(EMAC0 の例)

受信側および送信側のイーサネットフレームが、リングバッファに保存されます。この定義は、リングバッファが包含する要素の数を、送信方向と受信方向のそれぞれについて決定します。

```
/**

******************************************************************************
*
 ** \brief EMAC0 Transmission/Reception Ring size

******************************************************************************
*/
#define EMAC0_TX_RING_SIZE     2
#define EMAC0_RX_RING_SIZE     4
```

3.1.7 送受信バッファサイズ(EMAC0 の例)

送信および受信バッファの各サイズはここで設定されます。この定義は、処理可能なイーサネットフレームの最大サイズを決定します。定義値は 4 の倍数である必要があります。またデフォルト値はアプリケーションに適した値であることが必要です。

```
/**
 ******************************************************************************
 ** \brief EMAC0 Transmission/Reception Buffer size
 **
 ** Only used if EMAC0_BUFFERS_IN_DRIVERSPACE == PDL_ON
 ** Must be multiple of 4, 13 Bit value, page 159
 ******************************************************************************/
#define EMAC0_TX_BUF_SIZE     1536
#define EMAC0_RX_BUF_SIZE     1536
```

3.1.8 EMAC1 のバッファ設定

EMAC1 のバッファの位置、リングサイズ、バッファサイズに対しても上記と同様に設定できます。

3.1.9 スターターキットの設定

サイプレスのスターターキットを使用する際の設定を説明します。これによって、ボード固有の正しい PHY アドレスが自動的に選択されます。

```
/**
 ******************************************************************************
 ** \brief Support for Cypress starter kits
 **
 ** Activate board specific settings by defining exactly one of these symbols:
 **
 ** SK_FM3_176PMC_ETHERNET from Fujitsu Semiconductor Europe:
 ** STARTERKIT_SK_FM3_176PMC_ETHERNET
 **
 ** If version 1.0 is used, please use additionally:
 ** SK_FM3_176PMC_ETHERNET_BOARDVERSION10
 **
 ** FSSDC-9B618-EVB from Fujitsu Semiconductor Limited Asia:
 ** STARTERKIT_FSSDC9B618EVB
 **
 ******************************************************************************/
#define STARTERKIT_SK_FM3_176PMC_ETHERNET
//#define SK_FM3_176PMC_ETHERNET_BOARDVERSION10
//#define STARTERKIT_FSSDC9B618EVB
//#define STARTERKIT_SK_FM4_216_ETHERNET
```

3.1.10 外付け PHY のアドレス

MII/RMII 準拠のイーサネット PHY には、通常プルアップ抵抗またはプルダウン抵抗を使用して設定できるハードウェアアドレスがあります。各インスタンス用に使用される外部 PHY のハードウェア固有のアドレスは下記のように調整できます。この設定は使用するボードによって異なります。

EMAC1_PHY_ADDRESS は、マイクロコントローラに Ethernet MAC が 1 つしか含まれていない場合でも定義する必要があります。値の定義は必要ですが、下記のケースでは何も影響しません。

```
/**
 ******************************************************************************
 ** \brief Addresses of the used external PHYs
 ******************************************************************************/
#define EMAC0_PHY_ADDRESS     0
#define EMAC1_PHY_ADDRESS     1
```

3.1.11 EMAC0/EMAC1 の MAC アドレス

各インスタンスに応じて MAC アドレスやハードウェアアドレスを定義できます。なお、ドライババージョン 1.4 以降では、この定義はデモ用途だけです。実際のアドレス設定は、実行時に行われます。

```
/**
 ******************************************************************************
 ** \brief EMAC0 MAC address
 ******************************************************************************/
#define EMAC0_MAC_ADDRESS0  0x00
#define EMAC0_MAC_ADDRESS1  0x01
#define EMAC0_MAC_ADDRESS2  0x01
#define EMAC0_MAC_ADDRESS3  0x66
#define EMAC0_MAC_ADDRESS4  0x73
#define EMAC0_MAC_ADDRESS5  0x42

/**
 ******************************************************************************
 ** \brief EMAC1 MAC address
 ******************************************************************************/
#define EMAC1_MAC_ADDRESS0  0x00
#define EMAC1_MAC_ADDRESS1  0x01
#define EMAC1_MAC_ADDRESS2  0x01
#define EMAC1_MAC_ADDRESS3  0x66
#define EMAC1_MAC_ADDRESS4  0x73
#define EMAC1_MAC_ADDRESS5  0x38
```

3.1.12 マルチキャストアドレス

マルチキャストアドレスの設定例を示します。

```
/**
 ******************************************************************************
 ** \brief Multicast address
 ******************************************************************************/
#define MULTICAST_ADDRESS0  0x00
#define MULTICAST_ADDRESS1  0x00
#define MULTICAST_ADDRESS2  0x00
#define MULTICAST_ADDRESS3  0x00
#define MULTICAST_ADDRESS4  0x00
#define MULTICAST_ADDRESS5  0x00
```

3.1.13 外付け PHY のリセットピン

GPIO 端子を使用したマイクロコントローラによって、外付け PHY がリセットされます。外付け PHY のリセットピンの設定は、PHY のリセット入力ラインを制御するポート端子を定義します。デバイスのヘッダファイルで定義されている GPIO ビットバンドエイリアスアドレスへのアドレスポインタを指定してください。

設定方法の例を下記に示します。ここでは、P45 が PHY0、P44 が PHY1 のリセットピンとして設定しています。

```
/**
 ******************************************************************************
 ** \brief External PHY reset pins
 **
 ** Use GPIO addresses of the bit-band alias definitions of the device header
 ** file.
 ******************************************************************************/
#define EMAC0_PHY_RESET_PIN ((uint32_t*) &bFM3_GPIO_PDOR4_P5)
#define EMAC1_PHY_RESET_PIN ((uint32_t*) &bFM3_GPIO_PDOR4_P4)
```

3.1.14 PLL クロックモード

イーサネット対応のマイクロコントローラには、実行可能な 25MHz と 50MHz の出力があります。ジッタと周波数許容値がこの設定に対応可能かどうかは、PHY の製造元のデータシートを確認してください。

```
/**
 ******************************************************************************
 ** \brief PLL for External PHY
 **
 ** PDL_ON:  PLL is used for PHY clock
 ** PDL_OFF: PLL is not used for PHY clock
 ******************************************************************************/
#define EMAC_ECOUT  PDL_OFF
```

3.1.15 PHY のマネジメントバス設定

MII/RMII にはマネジメントバスがあります。接続された外部 PHY に対するマネジメントバスを、どの EMAC インスタンスが使用するかを設定するために、以下の定義を使用します。本定義は、両方の PHY が共通のマネジメントバスにつながっている場合に設定してください。

```
/**
 ******************************************************************************
 ** \brief Management Bus Definition
 ******************************************************************************/
#define EMAC0_MANAGEMENTBUS   EMAC0
#define EMAC1_MANAGEMENTBUS   EMAC1
```

3.1.16 RMII インタフェースの使用

下記の定義に PDL_ON を使用することで、PHY のインタフェースを RMII に設定できます。PDL_OFF の場合は MII が設定されます。2 つのイーサネット MAC を同時に使用するためには RMII に設定する必要があります。

```
/**
 ******************************************************************************
 ** \brief RMII PHY interface usage
 ******************************************************************************/
#define EMAC_PHYINTERFACE_RMII   PDL_ON
```

3.1.17 バッファのフラグメンテーション

PDL_ON に設定することで、全てのバッファをフラグメントしないように設定できます。これは PDL_ON に設定することを推奨します。

```
/**
 ******************************************************************************
 ** \brief EMAC Buffer fragmentation
 **
 ** All buffers are large enough to contain a whole Ethernet frame (MTU size)
 ******************************************************************************/
#define EMAC_BUFFERS_NOT_FRAGMENTED   PDL_ON
```

3.1.18 チェックサム挿入制御

下記のとおり設定することで本機能を使用できます。

0: チェックサム挿入がディセーブルされる

1: IP ヘッダチェックサムの計算と挿入のみがイネーブルされる

2: IP ヘッダチェックサムとペイロードチェックサムの計算と挿入がイネーブルされるが、
擬似ヘッダチェックサムはハードウェアでは計算されない

3: IP ヘッダチェックサムとペイロードチェックサムの計算と挿入がイネーブルされ、
擬似ヘッダチェックサムがハードウェアで計算される

```
/**
 ******************************************************************************
 ** \brief CIC (Checksum Insertion Control)
 **
 ** These bits control the checksum calculation and insertion. Bit encodings
 ** are as shown below.
 ** - 0  Checksum Insertion Disabled.
 ** - 1  Only IP header checksum calculation and insertion are enabled.
 ** - 2  IP header checksum and payload checksum calculation and insertion are
 **      enabled, but pseudo-header checksum is not calculated in hardware.
 ** - 3  IP Header checksum and payload checksum calculation and insertion are
 **      enabled, and pseudo-header checksum is calculated in hardware.
 ******************************************************************************/
#define EMAC_COE_MODE   3
```

3.1.19 ICMP チェックサムバグの回避方法

EMAC_ENABLE_ICMP_CHECKSUM_BUG_WORKAROUND に PDL_ON を設定することで、ドライバはチェックサムフィールドの ping 応答(echo 応答)に 0 を上書きします。本オプションは、EMAC_COE_MODE を 3 に設定していて、さらにシステムに対して ping コマンドを送っている間にチェックサムエラーを検出する場合に設定してください。

```
/**
 ******************************************************************************
 ** \brief Activate work-around for bug in uIP and LwIP TCP/IP stacks
 **
 ** For the Checksum Offload Engine to work correctly, the checksum fields in
 ** the IP header, as well as in the TCP, UDP or ICMP header must be 0.
 **
 ** LwIP up to the recent stable 1.4.0 version, and current uIP have an option
 ** to disable software checksum calculation for IP, UDP and TCP.
 ** A similar option ICMP was (apparently by accident) omitted.
 **
 ** If EMAC_ENABLE_ICMP_CHECKSUM_BUG_WORKAROUND is set to PDL_ON,
 ** every frame to be sent will be checked if it is an ICMP echo reply
 ** and if so, its checksum field cleared.
 **
 ** This slows down every packet transmission and should be fixed in the
 ** original code. This is a known bug and is expected to be fixed
 ** in future releases of LwIP.
 **
 ** For other TCP/IP stacks, this work-around is usually not necessary
 ** and should be disabled.
 **
 ** Cypress tries to retain third-party software in software examples in its
 ** original state as far as possible in order to retain compatibility to the
 ** maximum possible extend.
 **
 ******************************************************************************/
      #define EMAC_ENABLE_ICMP_CHECKSUM_BUG_WORKAROUND   PDL_ON
```

3.1.20 マルチキャストアドレスフィルタ

EMAC_MULTICAST_FILTER に PDL_ON を設定することで、ユーザはマルチキャストアドレスでフィルタされたフレームを受信できます。

```
/**
 ******************************************************************************
 ** \brief Activate multicast filtering
 **
 ** PDL_ON:  Multicast filtering is used
 ** PDL_OFF: Multicast filtering is not used
 **
 ******************************************************************************/
#define EMAC_MULTICAST_FILTER   PDL_OFF
```

3.1.21 PHY オートネゴシエーション コールバック関数名

IEEE802.3 標準では、オートネゴシエーションプロセスを開始する方法は定義されていますが、その結果の処理方法は定義されていません。そのため、このような特定のコールバック関数は、使用される特定の PHY タイプごとに提供されなければなりません。PHY 固有のコールバック関数名をここで指定する必要があります。

```
/**
 ******************************************************************************
 ** \brief Select callback function to read out autonegotiation result from PHY
 **
 ** Provided as a reference are callback functions for PHYs that are assembled
 ** on eval boards SK-FM3-176PMC-ETHERNET and SK-FM4-216-ETHERNET
 ******************************************************************************/
#define EMAC_AUTONEG_FUNCTION  EmacUser_AutoNegotiatePhy_SMSC_LAN8710A_CB
```

3.1.22 PHY オートネゴシエーション コールバック関数定義

コールバック関数の定義は *emac_user.c* ファイルに実装してください。サイプレスの評価ボードで使用されている PHY に適した、2 つの関数をリファレンスとして用意しています。

SK-FM3-176PMC-ETHERNET 用の EmacUser_AutoNegotiatePhy_SMSC_LAN8710A_CB()と
SK-FM4-216-ETHERNET 用の EmacUser_AutoNegotiatePhy_Micrel_KSZ8091_CB()です。

```
#if (EMAC_AUTONEG_FUNCTION == EmacUser_AutoNegotiatePhy_Micrel_KSZ8091_CB)
en_emac_link_mode_t EmacUser_AutoNegotiatePhy_Micrel_KSZ8091_CB(volatile
FM_ETHERNET_MAC_TypeDef* pstcEmac)
{
    en_emac_link_mode_t enLinkMode = EMAC_LinkModeAutonegotiation;
    uint16_t u16PhyRegVal = 0;

// According to data sheet, three LSB bits of register 0x1E indicate
// auto-negotiation result
    // Read out auto-negotiation result from PHY
    u16PhyRegVal = Ethphy_Read(pstcEmac, 0x1E);
    u16PhyRegVal &= 0x7;

    switch (u16PhyRegVal)
    {
        case(1):
            enLinkMode = EMAC_LinkModeHalfDuplex10M;
            break;
        case(2):
            enLinkMode = EMAC_LinkModeHalfDuplex100M;
            break;
        case(5):
            enLinkMode = EMAC_LinkModeFullDuplex10M;
            break;
         case(6):
            enLinkMode = EMAC_LinkModeFullDuplex100M;
            break;
         default:
            //printf("Autonegotiation error!");
            // Setting 100M full duplex, as it is a common setting
            enLinkMode = EMAC_LinkModeFullDuplex100M;
            break;
    }
    return enLinkMode;
}
#endif // (EMAC_AUTONEG_FUNCTION == EmacUser_AutoNegotiatePhy_Micrel_KSZ8091_CB)
```

3.2 *emac.c* での追加設定

ユーザは下記のサンプルコードを加えることで様々な機能を使用できます。

3.2.1 マルチキャスト MAC アドレスのフィルタ設定

`Emac_Autonegotiate` 関数内に下記のサンプルコードを加えることで、マルチキャストアドレスでフィルタしたフレームを受信できます。本サンプルコードは、マルチキャストアドレスのビット 0～ビット 7 をマスクした値でフィルタします。

```
      pstcEmac->MAR1H = (uint32_t)(((au8GenericMulticast[5]) <<  8)
                                   |((au8GenericMulticast[4]) <<  0)
                         );
      pstcEmac->MAR1L = (uint32_t)(((au8GenericMulticast[3]) << 24)
                                   |((au8GenericMulticast[2]) << 16)
                                   |((au8GenericMulticast[1]) <<  8)
                                   |((au8GenericMulticast[0]) <<  0));

      pstcEmac->MAR1H |= (uint32_t)0x20000000;
      pstcEmac->MAR1H |= (uint32_t)0x80000000;
```

3.3 サンプルの使用方法

ここではサンプルコードの具体的な使い方を説明します。

3.3.1 イーサネットインタフェースのセットアップ方法

EMAC0 と EMAC1 を両方使用する場合、下記のように初期化します。MCU の端子機能を初期化後にこの設定を行ってください。必要であれば、あらかじめコールバック関数も設定してください。

```
// Define configuration structure
stc_emac_config_t stcEmacConfig;
// Initialize configuration structure with zeros
PDL_ZERO_STRUCT(stcEmacConfig);

// Board pins setting for using Ethernet
ConfigureEthernetPins();

// Callback functions setting for EMAC0
stcEmacConfig.pfnRxCallback = EMAC0RxCallbackFunc;
stcEmacConfig.pfnTxCallback = EMAC0TxCallbackFunc;

// Set MAC address for EMAC0 (You can provide your own method here)
stcEmacConfig.au8MacAddress[0] = EMAC0_MAC_ADDRESS0;
stcEmacConfig.au8MacAddress[1] = EMAC0_MAC_ADDRESS1;
stcEmacConfig.au8MacAddress[2] = EMAC0_MAC_ADDRESS2;
stcEmacConfig.au8MacAddress[3] = EMAC0_MAC_ADDRESS3;
stcEmacConfig.au8MacAddress[4] = EMAC0_MAC_ADDRESS4;
stcEmacConfig.au8MacAddress[5] = EMAC0_MAC_ADDRESS5;

// Ethernet MAC 0 initialization
Emac_Init(&EMAC0, &stcEmacConfig);
Emac_Autonegotiate(&EMAC0);

// Define Ethernet MAC 1 Configuration
PDL_ZERO_STRUCT(stcEmacConfig); // Reuse config structure but reinitialize it
stcEmacConfig.au8MacAddress[0] = EMAC1_MAC_ADDRESS0;
stcEmacConfig.au8MacAddress[1] = EMAC1_MAC_ADDRESS1;
stcEmacConfig.au8MacAddress[2] = EMAC1_MAC_ADDRESS2;
stcEmacConfig.au8MacAddress[3] = EMAC1_MAC_ADDRESS3;
stcEmacConfig.au8MacAddress[4] = EMAC1_MAC_ADDRESS4;
stcEmacConfig.au8MacAddress[5] = EMAC1_MAC_ADDRESS5;

// callback functions setting for EMAC1
stcEmacConfig.pfnRxCallback = EMAC1RxCallbackFunc;
stcEmacConfig.pfnTxCallback = EMAC1TxCallbackFunc;

// Ethernet MAC 1 initialization
Emac_Init(&EMAC1, &stcEmacConfig);
Emac_Autonegotiate(&EMAC1);
```

定期的に `Emac_Autonegotiate`()を呼び出してイーサネットリンクを確立し、変更された接続状態に適応させることを推奨します。リンクがすでに確立している状態でこの関数を呼び出しても、何も問題ありません。

3.3.2 フレームの送信方法

初期化後に Emac_TxFrame 関数をコールすることでフレームを送信できます。本関数を使用するには、下記 3 つのパラメータを設定する必要があります。

1. EMAC のポインタ: EMAC (0 or 1)の選択
2. バッファのアドレス
3. バッファの長さ(バイト数)

あらかじめバッファにプロトコルヘッダとペイロードを設定してください。下記のサンプルはセットアップ(TxBufferUDPFill())後に、UDP フレームを送信(Emac_TxFrame())しています。

```
uint8_t ua8EthBuf[1500];
uint8_t ua8Payload[] = "Hello World!”;

// creating an Ethernet frame with IP and UDP header
TxBufferUDPFill(ua8EthBuf, ua8Payload, sizeof(ua8Payload));
ua8EthBuf[42 + sizeof(ua8Payload) - 3] = (u8NumberOfCalls + '0');

// transmit the frame
Emac_TxFrame(&EMAC0, ua8EthBuf, 42 + sizeof(ua8Payload));
```

3.3.3 フレームの受信方法

割込みモードを有効にしている場合、Emac_RxFrame_GetBufPtr 関数をコール後に memcpy 関数を使用することでフレームを受信できます。フレーム受信後、Emac_RxFrame_ReleaseBuf 関数をコールして次のディスクリプタを設定する必要があります。割込みモードを無効にしている場合、Emac_Rx 関数をコールすることでフレームを受信できます。

```
#if (EMAC_INTERRUPT_MODE == PDL_ON)
        u8pbuffer =(uint8_t *) Emac_RxFrame_GetBufPtr(pstcEmac);
        if (NULL != u8pbuffer)
        {
          memcpy((uint8_t*)p->payload, buffer, p->len);
        }
        Emac_RxFrame_ReleaseBuf(pstcEmac);
#else
        p->len =  Emac_RxFrame(pstcEmac,p->payload);
#endif
```

3.3.4 割込み処理

割込みを使用する場合、EMAC_INTERRUPT_MODE を PDL_ON に設定してください。

```
#define EMAC_INTERRUPT_MODE     PDL_ON
```

Emac_InitIrq 関数で必要な割込みを追加してください。デフォルトの設定ではアブノーマル割込みは許可されていません。

```
pstcEmac->stcIER.NIE = 1;
pstcEmac->stcIER.RIE = 1;
pstcEmac->stcIER.TIE = 1;
```

割込み発生後、Emac_IrqHandler 関数がコールされます。必要に応じて Emac_IrqHandler 関数内の下記のコードを変更してください。デフォルトの設定では、送受信完了後にコールバック関数がコールされます。

```
if(pstcEmac->stcSR.RI != 0)
{
    pstcEmac->stcSR.RI = 1;
    if(pstcEmacInternData->pfnRxCallback != NULL)
    {
        pstcEmacInternData->pfnRxCallback();
    }
}
if(pstcEmac->stcSR.TI != 0)
{
    pstcEmac->stcSR.TI = 1;
    if(pstcEmacInternData->pfnTxCallback != NULL)
    {
        pstcEmacInternData->pfnTxCallback();
    }
}
```

4 イーサネットドライバのコンフィグレーションとデータ型

ここではイーサネットデータ構造を説明します。

4.1 はじめに

静的なコンパイラ定義だけでなく、ユーザは個々のコンフィグレーションを使用してイーサネットインスタンスを動的に構成することもできます。本コンフィグレーションは、初期化関数(ハードウェアと内部データを設定)で使用されます。 初期化関数で使用された後、このコンフィグレーションは使用されません。

4.2 コンフィグレーションの構造体

下記のデータは、イーサネットドライバのコンフィグレーションの構造体である `stc_emac_config_t` で使用されます。

型	フィールド	値	説明
boolean_t	bPromiscuousMode	PDL_ON, PDL_OFF	有効な場合、インタフェースはすべてのフレーム(アドレス設定がないものも含めて)を受信します。
func_ptr_t	pfnRxCallback	-	受信コールバック関数の関数ポインタ
func_ptr_t	pfnTxCallback	-	送信コールバック関数の関数ポインタ
func_ptr_t	pfnErrorCallback	-	エラーコールバック関数の関数ポインタ
uint8_t array	au8MacAddress[6]	-	イーサネットハードウェアアドレス(MAC アドレスとも呼ぶ)

4.3 リンクステータスの enum 型

イーサネットドライバはリンクステータスに enum 型を使用します。これは、下記のとおり `en_emac_link_t` で定義されています。

設定値	説明
EMAC_LinkStatusLinkDown	コネクションが確立されていない
EMAC_LinkStatusLinkUp	コネクションが確立され、データ交換が可能なように MAC が設定されています
EMAC_LinkStatusAutonegotiationInProgress	オートネゴシエーション開始
EMAC_LinkStatusAutonegotiationSuccessful	ノードとのオートネゴシエーションが完了
EMAC_LinkStatusAutonegotiationNotSupported	ノードがオートネゴシエーションに非対応
EMAC_LinkStatusInvalidParameter	異常な値が設定されている
EMAC_LinkStatusUnknownError	不明なエラーが発生

4.4 リンクモードの enum 型

リンクモードについても、`en_enum_link_mode_t` という名前の enum 型が定義されています。内容は下記のとおりです。

設定値	説明
EMAC_LinkModeHalfDuplex10M	半二重モード 10Mbit/s
EMAC_LinkModeFullDuplex10M	全二重モード 10Mbit/s
EMAC_LinkModeHalfDuplex100M	半二重モード 100Mbit/s
EMAC_LinkModeFullDuplex100M	全二重モード 100Mbit/s

5 イーサネットドライバ API

イーサネットドライバ API の詳細とサンプルコードを説明します。

5.1 Ethernet MAC functions

ここではイーサネットドライバの API 関数の詳細とサンプルコードについて説明します。

5.1.1 Emac_Init()

本関数はイーサネットを初期化します。`pstcEmac` は `(FM_ETHERNET_MAC_TypeDef*)&EMAC0` か `(FM_ETHERNET_MAC_TypeDef*)&EMAC1` のどちらかでなければいけません。`pstcConfig` はユーザ設定用の変数のアドレスです。例えば、`&stcEmacConfig` のように設定します。

プロトタイプ	

`en_result_t Emac_Init( volatile FM_ETHERNET_MAC_TypeDef* pstcEmac,`
`stc_emac_config_t* pstcConfig`
`)`	

パラメータ名	**説明**
`[in] pstcEmac`	イーサネット MAC のポインタ
`[in] pstcConfig`	イーサネット構成のパラメータ
戻り値	**説明**
Ok	指定チャネルを有効に設定

ErrorInvalidParameter	下記のどちらかが設定されています。
	`pstcEmac == NULL`
	`pstcEmacInternData == NULL`
	`(invalid or disabled Ethernet unit)`

5.1.2 Emac_DeInit()

本関数はイーサネットの初期化を解除します。

プロトタイプ	
`en_result_t Emac_DeInit( volatile FM_ETHERNET_MAC_TypeDef* pstcEmac )`	
パラメータ名	**説明**
`[in] pstcEmac`	イーサネット MAC のポインタ
戻り値	**説明**
`Ok`	指定チャネルを無効に設定
`ErrorInvalidParameter`	下記が設定されています。 `pstcEmac == NULL`

この関数はイーサネット IRQ の初期化を解除し、EMAC をリセット状態にし、内部データを消去します。

MAC1 は、MAC0 にクロック信号が供給されている場合にのみ機能します。したがって、Emac_DeInit(EMAC0)は EMAC0 をリセット状態にしますが、EMAC1 がアクティブでない場合にのみクロック供給をオフにします。Emac_DeInit(EMAC1)を先に呼び出すか、あるいは Emac_Init(EMAC1)を先に呼び出さない場合に起こります。両方のイーサネットチャネルの初期化を解除する場合は、Emac_DeInit(EMAC1)を先に呼び出し、Emac_DeInit(EMAC0)をそのあとに呼び出してください。

5.1.3 Emac_TxFrame()

本関数はフレーム送信に使用されます。

プロトタイプ	
`en_result_t Emac_TxFrame( volatile FM_ETHERNET_MAC_TypeDef* pstcEmac, uint8_t* pu8Buf, uint16_t u16Len )`	
パラメータ名	**説明**
`[in] pstcEmac`	イーサネット MAC のポインタ
`[in] pu8Buf`	送信されるバイト配列バッファへのポインタ
`[in] u16Len`	バイト配列バッファ長、送信されるバイト数
戻り値	**説明**
`Ok`	送信完了
`ErrorInvalidParameter`	下記が設定されています。 `pstcEmac == NULL`

5.1.4 Emac_GetRxFrameLength()

本関数は受信フレームの長さを返します。

プロトタイプ	
`uint16_t Emac_GetRxFrameLength( volatile FM_ETHERNET_MAC_TypeDef* pstcEmac )`	
パラメータ名	**説明**
`[in] pstcEmac`	イーサネット MAC のポインタ
戻り値	**説明**
`uint16_t`	受信したフレームの長さ(何も受信していない、もしくは `pstcEmac == NULL`

5.1.5 Emac_RxFrame()

Copies a received Ethernet frame into a buffer provided by the user application.

プロトタイプ	
`uint16_t Emac_RxFrame( volatile FM_ETHERNET_MAC_TypeDef* pstcEmac, uint8_t* pu8Buf )`	
パラメータ名	**説明**
`[in] pstcEmac`	イーサネット MAC のポインタ
`[in] pu8Buf`	受信したイーサネットフレームの内容で上書きされるバイト配列バッファへのポインタ
戻り値	**説明**
`uint16_t`	受信したフレームの長さ あるいは 0 (何も受信していない)、もしくは `pstcEmac == NULL`

5.1.6 Emac_GetLinkStatus()

本関数は最新のリンクステータスを返します。型には 4.3 章に記載されている `en_emac_link_status_t` を使用します。

プロトタイプ	
`en_emac_link_status_t Emac_GetLinkStatus( volatile FM_ETHERNET_MAC_TypeDef* pstcEmac )`	
パラメータ名	**説明**
`[in] pstcEmac`	イーサネット MAC のポインタ
戻り値	**説明**

`en_emac_link_status_t`	下記の enum 値を返します。
	`EMAC_LinkStatusLinkDown`
	`EMAC_LinkStatusLinkUp`
	`EMAC_LinkStatusAutonegotiationInProgress`
	`EMAC_LinkStatusAutonegotiationSuccessful`
	`EMAC_LinkStatusAutonegotiationNotSupported`
	`EMAC_LinkStatusInvalidParameter`
	`EMAC_LinkStatusUnknownError`
	
	詳細は 4.3 章を参照してください。

5.1.7 Emac_GetLinkMode()

本関数は最新のリンクモードを返します。型には 4.4 節に記載されている en_emac_link_mode_t を使用します。

プロトタイプ	
en_emac_link_mode_t Emac_GetLinkMode(volatile FM_ETHERNET_MAC_TypeDef* pstcEmac)	
パラメータ名	**説明**
[in] pstcEmac	イーサネット MAC のポインタ
戻り値	**説明**
en_emac_link_mode_t	下記の enum 値を返します。 EMAC_LinkModeAutonegotiation EMAC_LinkModeHalfDuplex10M EMAC_LinkModeFullDuplex10M EMAC_LinkModeHalfDuplex100M EMAC_LinkModeFullDuplex100M 詳細は 4.4 節を参照してください。

5.1.8 Emac_SetLinkMode()

本関数はリンクモードを設定します。引数と戻り値には 4.4 節に記載されている en_emac_link_mode_t を使用します。

プロトタイプ	
en_emac_link_mode_t Emac_SetLinkMode(volatile FM_ETHERNET_MAC_TypeDef* pstcEmac, en_emac_link_mode_t enLinkMode)	
パラメータ名	**説明**
[in] pstcEmac	イーサネット MAC のポインタ
[in] enLinkMode	リンクモード。enum の値は下記の戻り値を参照してください。
戻り値	**説明**
en_emac_link_mode_t	下記の enum 値を返します。 EMAC_LinkModeAutonegotiation EMAC_LinkModeHalfDuplex10M EMAC_LinkModeFullDuplex10M EMAC_LinkModeHalfDuplex100M EMAC_LinkModeFullDuplex100M 詳細は 4.4 節を参照してください。

5.1.9 Emac_Autonegotiate()

本関数はイーサネットマクロのオートネゴシエーションを実行します。定期的にコールしてください。

プロトタイプ	
en_emac_link_status_t Emac_Autonegotiate(volatile FM_ETHERNET_MAC_TypeDef* pstcEmac)	
パラメータ名	**説明**
[in] pstcEmac	イーサネット MAC のポインタ
戻り値	**説明**
en_emac_link_status_t	下記の enum 値を返します。 EMAC_LinkStatusLinkDown EMAC_LinkStatusLinkUp EMAC_LinkStatusAutonegotiationInProgress EMAC_LinkStatusAutonegotiationSuccessful EMAC_LinkStatusAutonegotiationNotSupported EMAC_LinkStatusInvalidParameter EMAC_LinkStatusUnknownError 詳細は 4.3 章を参照してください。

5.1.10 Emac_SetDescToTxBuf()

本関数はバッファの位置がイーサネットドライバ側に定義されている場合にのみ利用できます。バッファの位置は、EMAC0_BUFFERS_IN_DRIVERSPACE もしくは EMAC1_BUFFERS_IN_DRIVERSPACE で定義できます。

プロトタイプ	
en_result_t Emac_SetDescToTxBuf(volatile FM_ETHERNET_MAC_TypeDef* pstcEmac, uint16_t u16DescNumber, uint8_t * pu8UserBuf, uint16_t u16BufLength)	
パラメータ名	**説明**
[in] pstcEmac	イーサネット MAC のポインタ
[in] u16DescNumber	DMA ディスクリプタの番号
[in] pu8UserBuf	バッファのポインタ
[in] u16BufLength	バッファの長さ
戻り値	**説明**
Ok	ディスクリプタ設定完了
ErrorInvalidParameter	下記が設定されています。 pstcEmac == NULL

5.1.11 Emac_SetDescToRxBuf()

本関数はバッファの位置がイーサネットドライバ側に定義されている場合にのみ利用できます。バッファの位置は、EMAC0_BUFFERS_IN_DRIVERSPACE と EMAC1_BUFFERS_IN_DRIVERSPACE で定義できます。

プロトタイプ
en_result_t Emac_SetDescToRxBuf(volatile FM_ETHERNET_MAC_TypeDef* pstcEmac, uint16_t u16DescNumber, uint8_t * pu8UserBuf, uint16_t u16BufLength)

パラメータ名	説明
[in] pstcEmac	イーサネット MAC のポインタ
[in] u16DescNumber	DMA ディスクリプタの番号
[in] pu8UserBuf	バッファのポインタ
[in] u16BufLength	バッファの長さ
戻り値	**説明**
Ok	ディスクリプタ設定完了
ErrorInvalidParameter	下記が設定されています。 pstcEmac == NULL

5.1.12 Emac_SetEcout()

本関数は PLL クロックの使用を許可します。EMAC_ECOUT が PDL_ON の場合、E_COUT ピンは出力に設定され USB/Ethernet クロックが有効になります。

プロトタイプ
en_result_t Emac_SetEcout(void)

パラメータ名	説明
void	-
戻り値	**説明**
Ok	設定完了

5.1.13 Emac_RxFrame_GetBufPtr()

受信フレームのバッファをポインタで返します。受信処理でのデータコピー処理に使用できます。割込みモードを有効にしている場合に使用してください。(3.3.3 項を参照)

プロトタイプ	
`void* Emac_RxFrame_GetBufPtr( volatile FM_ETHERNET_MAC_TypeDef* pstcEmac )`	
パラメータ名	**説明**
`[in] pstcEmac`	イーサネット MAC のポインタ
戻り値	**説明**
`void *`	受信フレームのバッファのポインタまたは 0(何も受信していない)、もしくは `pstcEmac == NULL`

5.1.14 Emac_RxFrame_ReleaseBuf()

受信ディスクリプタを次のディスクリプタへ更新します。受信処理によってデータを所持したバッファに対し、解放処理をします。割込みモードを有効にしている場合に使用してください。(3.3.3 項を参照)

プロトタイプ	
`void Emac_RxFrame_ReleaseBuf( volatile FM_ETHERNET_MAC_TypeDef* pstcEmac )`	
パラメータ名	**説明**
`[in] pstcEmac`	イーサネット MAC のポインタ
戻り値	**説明**
`void`	-

5.2 Ethernet PHY functions

イーサネットドライバには、接続されたイーサネット PHY をリセットし、MII/RMII のステーションマネジメントインタフェース経由でデータを書き込み、そこから読み取る関数があります。これらの関数は内部処理と見なされますが、5.1 節で説明した Emac_で始まる関数がそれらを処理するため、安全に無視されます。

5.2.1 Ethphy_Read()

この関数は PHY から値を読み出します。

プロトタイプ	
uint16_t Ethphy_Read(volatile FM_ETHERNET_MAC_TypeDef* pstcEmac, uint8_t u8PhyReg);	
パラメータ名	**説明**
[in] pstcEmac	イーサネットマクロインスタンスのポインタ
[in] u8PhyReg	読み出されるレジスタアドレス
戻り値	**説明**
uint16_t	指定されたレジスタの値

5.2.2 Ethphy_Write()

この関数は PHY へ値を書き込みます。

プロトタイプ	
en_result_t Ethphy_Write(volatile FM_ETHERNET_MAC_TypeDef* pstcEmac, uint8_t u8PhyReg, uint16_t u16Val)	
パラメータ名	**説明**
[in] pstcEmac	イーサネットマクロインスタンスのポインタ
[in] u8PhyReg	上書きするレジスタアドレス
[in] u16Val	PHY に書き込まれる値
戻り値	**説明**
OK	値は PHY に書き込まれました。
ErrorInvalidParameter	pstcEmac は無効です。

5.2.3 Ethphy_Reset()

この関数は、*emac_user.h* の EMAC0_PHY_RESET_PIN または EMAC1_PHY_RESET_PIN で定義された GPIO ピンを介して PHY をリセットします。

プロトタイプ	
`uint16_t Ethphy_Read( volatile FM_ETHERNET_MAC_TypeDef* pstcEmac );`	
パラメータ名	**説明**
`[in] pstcEmac`	イーサネットマクロインスタンスのポインタ
戻り値	**説明**
`OK`	PHY リセットシーケンスが実行されました
`ErrorInvalidParameter`	pstcEmac は無効です

6 追加情報

サイプレスのマイクロコントローラに関する詳細な情報は、下記サイトをご利用ください。

http://www.cypress.com/cypress-microcontrollers

このアプリケーションノートに関するソフトウェアサンプルとして下記を用意しています。

mb9bfxxx_ethernet_driver

s6e2cc_ethernet_driver

7 改訂履歴

文書名: AN204417 - MB9BFXXX, S6E2CC, S6E2G イーサネットドライバマニュアル

文書番号: 002-04418

版	ECN	変更者	発行日	変更内容
**	-	NNAK	01/31/2014	スパンションアプリケーションノート AN706-00059-1v1-J をサイプレスとして登録しました。本版の変更はありません
*A	5783957	SSAS	06/23/2017	これは英語版 002-04417 Rev. *A を翻訳した日本語版です。

ワールドワイドな販売と設計サポート

サイプレスは、事業所、ソリューション センター、メーカー代理店、および販売代理店の世界的なネットワークを保持しています。お客様の最寄りのオフィスについては、サイプレスのロケーション ページをご覧ください。

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
車載用	cypress.com/automotive
クロック&バッファ	cypress.com/clocks
インターフェース	cypress.com/interface
IoT (モノのインターネット)	cypress.com/iot
メモリ	cypress.com/memory
マイクロコントローラ	cypress.com/mcu
PSoC	cypress.com/psoc
電源用 IC	cypress.com/pmic
タッチ センシング	cypress.com/touch
USB コントローラー	cypress.com/usb
ワイヤレス／RF	cypress.com/wireless

PSoC® ソリューション

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

サイプレス開発者コミュニティ

フォーラム | WICED IOT Forums | Projects | ビデオ | ブログ | トレーニング | Components

テクニカルサポート

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2013-2017. 本書面は, Cypress Semiconductor Corporation 及び Spansion LLC を含むその子会社 (以下「Cypress」という。) に帰属する財産である。本書面 (本書面に含まれ又は言及されているあらゆるソフトウェア若しくはファームウェア (以下「本ソフトウェア」という。) を含む) は, アメリカ合衆国及び世界のその他の国における知的財産法令及び条約に基づき Cypress が所有する。Cypress はこれらの法令及び条約に基づく全ての権利を留保し, 本段落で特に記載されているものを除き, その特許権, 著作権, 商標権又はその他の知的財産権のライセンスを一切許諾しない。本ソフトウェアにライセンス契約書が伴っておらず, かつ Cypress との間で別途本ソフトウェアの使用方法を定める書面による合意がない場合, Cypress は, (1) 本ソフトウェアの著作権に基づき, (a) ソースコード形式で提供されている本ソフトウェアについて, Cypress ハードウェア製品と共に用いるためにのみ, かつ組織内部でのみ, 本ソフトウェアの修正及び複製を行うこと, 並びに (b) Cypress のハードウェア製品ユニットに用いるためにのみ, (直接又は再販売者及び販売代理店を介して間接のいずれかで) 本ソフトウェアをバイナリーコード形式で外部エンドユーザーに配布すること, 並びに (2) 本ソフトウェア (Cypress により提供され, 修正がなされていないもの) が抵触する Cypress の特許権のクレームに基づき, Cypress ハードウェア製品と共に用いるためにのみ, 本ソフトウェアの作成, 利用, 配布及び輸入を行うことについての非独占的で譲渡不能な一身専属的ライセンス (サブライセンスの権利を除く) を付与する。本ソフトウェアのその他の使用, 複製, 修正, 変換又はコンパイルを禁止する。

適用される法律により許される範囲内で, Cypress は, 本書面又はいかなる本ソフトウェア若しくはこれに伴うハードウェアに関しても, 明示又は黙示をとわず, いかなる保証 (商品性及び特定の目的への適合性の黙示の保証を含むがこれらに限られない) も行わない。適用される法律により許される範囲内で, Cypress は, 別途通知することなく, 本書面を変更する権利を留保する。Cypress は, 本書面に記載のある, いかなる製品若しくは回路の適用又は使用から生じる一切の責任を負わない。本書面で提供されたあらゆる情報 (あらゆるサンプルデザイン情報又はプログラムコードを含む) は, 参照目的のためのみに提供されたものである。この情報で構成するあらゆるアプリケーション及びその結果としてのあらゆる製品の機能性及び安全性を適切に設計, プログラム, かつテストすることは, 本書面のユーザーの責任において行われるものとする。Cypress 製品は, 兵器, 兵器システム, 原子力施設, 生命維持装置若しくは生命維持システム, 蘇生用の設備及び外科的移植を含むその他の医療機器若しくは医療システム, 汚染管理若しくは有害物質管理の運用のために設計され若しくは意図されたシステムの重要な構成部分としての使用, 又は装置若しくはシステムの不具合が人身傷害, 死亡若しくは物的損害を生じさせるようなその他の使用 (以下「本目的外使用」という。) のためには設計, 意図又は承認されていない。重要な構成部分とは, それの不具合が装置若しくはシステムの不具合を生じさせるか又はその安全性若しくは実効性に影響すると合理的に予想できるような装置若しくはシステムのあらゆる構成部分をいう。Cypress 製品のあらゆる本目的外使用から生じ, 若しくは本目的外使用に関連するいかなる請求, 損害又はその他の責任についても, Cypress はその全部又は一部をとわず一切の責任を負わず, かつ Cypress はそれら一切から本書により免除される。Cypress は Cypress 製品の本目的外使用から生じ又は本目的外使用に関連するあらゆる請求, 費用, 損害及びその他の責任 (人身傷害又は死亡に基づく請求を含む) から免責補償される。

Cypress, Cypress のロゴ, Spansion, Spansion のロゴ及びこれらの組み合わせ, WICED, PSoC, CapsSense, EZ-USB, F-RAM, 及び Traveo は, 米国及びその他の国における Cypress の商標又は登録商標である。Cypress のより完全な商標のリストは, cypress.com を参照すること。その他の名称及びブランドは, それぞれの権利者の財産として権利主張がなされている可能性がある。