RX ファミリ

lwIP (lightweight IP) Driver Firmware Integration Technology

要旨

本アプリケーションノートは、Firmware Integration Technology (FIT) に準拠した lwIP (lightweight IP) Driver FIT モジュール (r_lwip_driver_rx) について説明します。

以降、lwIP (lightweight IP) Driver FIT モジュールのソフトウェアを総じて“lwIP Driver FIT モジュール”、または”本 FIT モジュール”と称します。

動作確認デバイス

RX65N グループ
RX72M グループ
RX72N グループ
RX64M グループ
RX71M グループ
RX66N グループ

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。

- FreeRTOS™および FreeRTOS.org™は、Amazon Web Services, Inc.の商標です。

関連ドキュメント

[1] Firmware Integration Technology ユーザーズマニュアル (R01AN1833)
[2] RX ファミリ ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)
[3] e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)
[4] CS+に組み込む方法 Firmware Integration Technology (R01AN1826)
[5] Renesas e^2 studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)
[6] RX ファミリ イーサネットモジュール Firmware Integration Technology (R01AN2009)
[7] RX ファミリ CMT モジュール Firmware Integration Technology (R01AN1856)
[8] RX ファミリ TSIP (Trusted Secure IP) モジュール Firmware Integration Technology (R20AN0371)
[9] lwIP - A Lightweight TCP/IP stack (リンク)
[10] Lightweight IP stack (リンク)
[11] lwip-tcpip GitHub (リンク)
[12] RX ファミリ lwIP (lightweight IP) Firmware Integration Technology (R20AN0789)
[13] FreeRTOS.org (リンク)

目次

1. 概要

1.1 lwIP Driver FIT モジュールとは

本 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。本 FIT モジュールの組み込み方については、「2.11 FIT モジュールの追加方法」を参照してください。

1.2 lwIP Driver FIT モジュールの概要

本 FIT モジュールは、上位層に lwIP FIT モジュール (r_lwip_rx、lwIP) を使用し、TCP/IP プロトコルでのイーサネット通信をサポートします。本 FIT モジュールは、オープンソースソフトウェア (Open Source Software、OSS) である lwIP (lightweight IP) で構成される lwIP FIT モジュールの RX MCU 向けのポーティングレイヤを提供します。

1.2.1 ソフトウェア構成

ソフトウェア構成を、図 1 に示します。

本 FIT モジュール (lwIP Driver FIT モジュール) は、lwIP FIT モジュールと組み合わせて動作します。

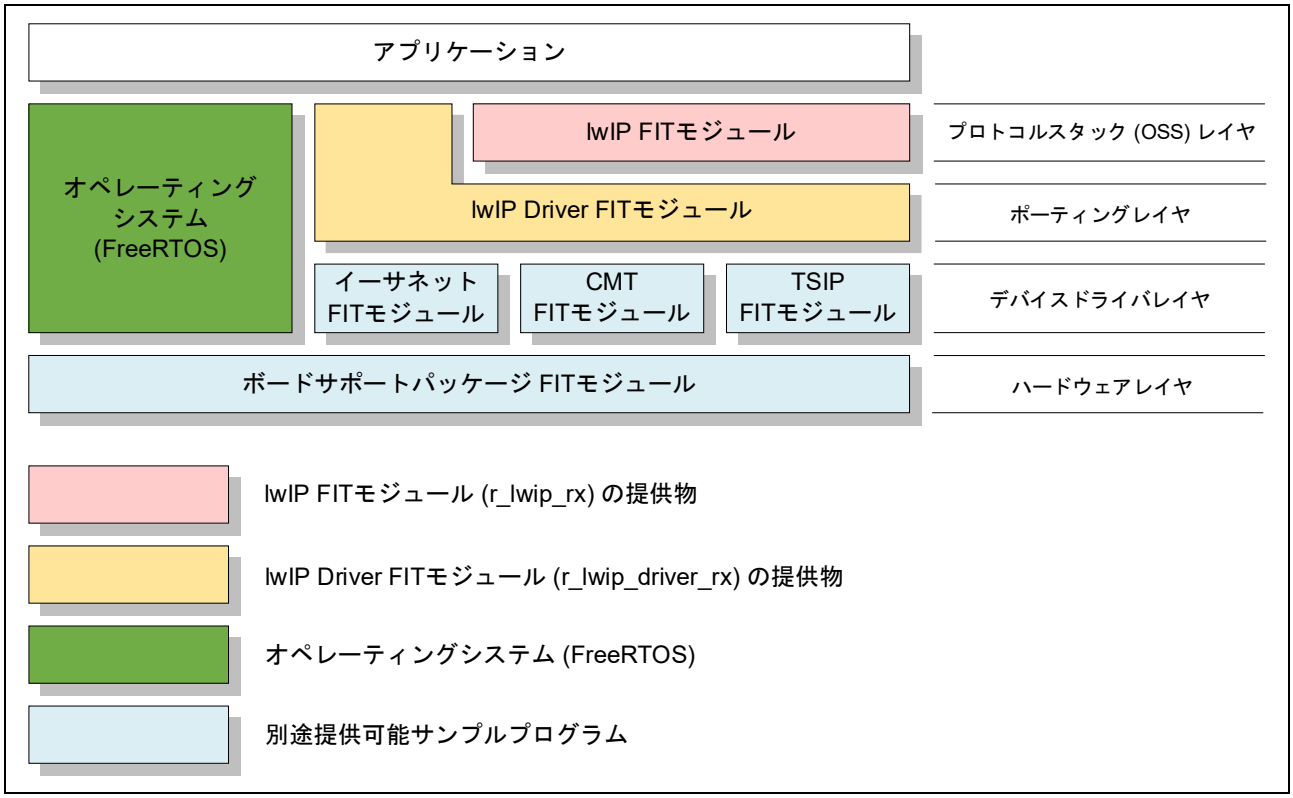

図 1 ソフトウェア構成図

(1) lwIP Driver FIT モジュール (r_lwip_driver_rx)
本 FIT モジュールです。lwIP FIT モジュール (lwIP) のポーティング層として使用するソフトウェアです。

(2) lwIP FIT モジュール (r_lwip_rx)
lwIP オープンソースソフトウェアを FIT モジュールとしてパッケージ化した、TCP/IP のプロトコルスタック実装モジュールです。

詳細については、先頭ページの「関連ドキュメント」を参照してください。

(3) イーサネット FIT モジュール (r_ether_rx)
イーサネットフレームの送受信を行います。サンプルプログラムが入手可能です。先頭ページの「関連ドキュメント」を参照し、入手してください。

(4) CMT FIT モジュール (r_cmt_rx)
lwIP FIT モジュール (lwIP) 内での時間計測に、コンペアマッチタイマ (CMT) を使用します。サンプルプログラムが入手可能です。先頭ページの「関連ドキュメント」を参照し、入手してください。
本モジュールは、OS (FreeRTOS) を使用しない場合のみ使用します。

(5) TSIP FIT モジュール (r_tsip_rx)
乱数生成処理を行います。サンプルプログラムが入手可能です。先頭ページの「関連ドキュメント」を参照し、入手してください。
本モジュールは、使用する MCU に TSIP が搭載されている場合のみ使用します。

(6) ボードサポートパッケージ FIT モジュール (r_bsp)
マイコンの設定を行います。サンプルプログラムが入手可能です。先頭ページの「関連ドキュメント」を参照し、入手してください。

(7) オペレーティングシステム (FreeRTOS)
オペレーティングシステム (OS) の API および機能を利用して lwIP FIT モジュール (lwIP) を使用する場合に使用します。本 FIT モジュールは、FreeRTOS に対応しています。
OS を使用する場合は、lwIP FIT モジュール (lwIP) の NO_SYS オプションに 0 を設定してください。
OS を使用する場合の設定およびユーザアプリケーションの実装については、関連ドキュメントおよび 5.2 節に示すサンプルプロジェクトを確認してください。

1.2.2　API の概要

本 FIT モジュールで定義する API 関数を、表 1.1 に示します。

表 1.1 lwIP Driver FIT API 関数

関数	説明
R_LWIP_DRIVER_Open	本 FIT モジュールの初期化を行います。
R_LWIP_DRIVER_Close	本 FIT モジュールで取得したリソースを解放します。
R_LWIP_DRIVER_EthernetClose	ETHERC の送受信機能を停止します。
R_LWIP_DRIVER_EthernetLinkCheck	イーサネットのリンク状態の確認を行います。
r_lwip_driver_ethernetif_init	ユーザが定義した netif 構造体の初期設定、およびイーサネット FIT モジュールの初期化処理を行います。 ユーザは、本関数を直接呼び出さないでください。
R_LWIP_DRIVER_Input	受信したイーサネットフレームの読み出しを行います。 OS を使用する場合、ユーザは本関数を直接呼び出さないでください。
r_lwip_driver_low_level_output	送信するイーサネットフレームの書き込みを行います。 ユーザは、本関数を直接呼び出さないでください。
r_lwip_driver_get_rand	乱数を生成します。
r_lwip_driver_input_thread	イーサネットフレームの受信処理を実行します。 OS を使用する場合に使用します。ユーザは、本関数を直接呼び出さないでください。

1.2.3　対応プロトコル

本 FIT モジュールが対応するプロトコルを、表 1.2 に示します。

表 1.2 lwIP Driver FIT 対応プロトコル

ネットワーク階層	プロトコル
インターネット層	IPv4, ICMP, IGMP, ARP, IPv6, ICMPv6
トランスポート層	TCP, UDP
アプリケーション層	DHCP, DHCPv6, HTTP (server)

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 ハードウェアの要求

ご使用になるマイコンが、以下の機能をサポートしている必要があります。

- イーサネットコントローラ (ETHERC)
- イーサネットコントローラ用 DMA コントローラ (EDMAC)
- コンペアマッチタイマ (CMT)
- Trusted Secure IP (TSIP) (注 1)

注1. TSIP を搭載している MCU を使用する場合に必要。

2.2 ソフトウェアの要求

本 FIT モジュールは、以下の FIT モジュールに依存します。

- r_bsp (ボードサポートパッケージ FIT モジュール[2])
- r_ether_rx (イーサネット FIT モジュール[6])
- r_cmt_rx (CMT FIT モジュール[7]) (注 1)
- r_lwip_rx (lwIP FIT モジュール[12])
- r_tsip_rx (TSIP FIT モジュール[8]) (注 2)

注1. OS を使用する場合、CMT FIT モジュールは使用しない。
注2. TSIP を搭載している MCU を使用する場合に必要。

2.3 サポートされているツールチェーン

本 FIT モジュールは、「5.1 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

なし

2.5 ヘッダファイル

本 FIT モジュールの API 呼び出しは、r_lwip_driver_rx_if.h で定義されています。

2.6 整数型

本 FIT モジュールは、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

本 FIT モジュールのコンフィグレーションオプションは、r_lwip_driver_rx_config.h で定義されています。

オプション名および設定値の説明を、表 2.1 に示します。

表 2.1 Configuration options (r_lwip_driver_rx_config.h)

Configuration options in r_lwip_driver_rx_config.h	
LWIP_DRIVER_CFG_ETH_MAC_ADDR0 ※デフォルトは “0x74”	MAC アドレスを設定します。 LWIP_DRIVER_CFG_ETH_MAC_ADDR0-5 で設定してください。 LWIP_DRIVER_CFG_ETH_MAC_ADDR0 が第 1 オクテットです。 それぞれ“0x00 ~ 0xFF”の範囲で設定してください。 MAC アドレスは、IEEE から取得したベンダーコード (Organizationally Unique Identifier、OUI) を含む、ネットワークインタフェースコントローラ (NIC) に割り当てられる一意な ID です。MAC アドレスは、各デバイスの本オプションにそれぞれ設定してください。マルチインタフェースを使用する場合の詳細は、5.3.1.3 節を参照してください。 本 FIT モジュールは、本オプションに設定された MAC アドレスを、r_lwip_driver_ethernetif_init() 関数から実行される r_lwip_driver_low_level_init() 関数で lwIP FIT モジュール (lwIP) へ設定します。 ソフトウェアのビルド後に不揮発メモリなどへ MAC アドレスを設定する必要がある場合は、本 FIT モジュールの MAC アドレス設定処理を変更してください。
LWIP_DRIVER_CFG_ETH_MAC_ADDR1 ※デフォルトは “0x90”	
LWIP_DRIVER_CFG_ETH_MAC_ADDR2 ※デフォルトは “0x50”	
LWIP_DRIVER_CFG_ETH_MAC_ADDR3 ※デフォルトは “0x10”	
LWIP_DRIVER_CFG_ETH_MAC_ADDR4 ※デフォルトは “0xFE”	
LWIP_DRIVER_CFG_ETH_MAC_ADDR5 ※デフォルトは “0x79”	
LWIP_DRIVER_CFG_ETH_DRV_CH ※デフォルトは “0”	lwIP で使用する ETHERC チャネル番号を設定します。 デフォルト値は ETHERC チャネル番号 0 を使用する場合です。制御する ETHERC チャネルに合わせて設定してください。 ETHERC が 2 チャネルあるデバイスの場合は 0 または 1 を、それ以外は 0 を設定してください。
LWIP_DRIVER_CFG_MTU ※デフォルトは “1500”	最大伝送単位を設定します。単位は byte です。 “576 ~ 1500”の範囲で設定してください。
LWIP_DRIVER_CFG_SEND_MAX_LOOP ※デフォルトは “20”	送信バッファの取得時にバッファフルだった場合、送信バッファが空くまでリトライする回数を設定します。
LWIP_DRIVER_CFG_SEND_DELAY_US ※デフォルトは “50”	送信バッファの取得時にバッファフルだった場合、送信バッファが空くまでのリトライ間隔を設定します。単位は、マイクロ秒です。

2.8　コードサイズ

本 FIT モジュールの ROM サイズ、RAM サイズ、最大使用スタックサイズを表 2.2 に示します。

ROM (コードおよび定数) と RAM (グローバルデータ) のサイズは、ビルド時の「2.7 コンパイル時の設定」のコンフィグレーションオプションによって決まります。

表 2.2 に示す値は、下記条件で確認しています。

FIT モジュールリビジョン: r_lwip_driver_rx rev1.11
コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.07.00
(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)
コンフィグレーションオプション: デフォルト設定

表 2.2 コードサイズ

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**	**備考**
RX65N	ROM	約 2k バイト	-
	RAM	12 バイト	-
	最大使用スタックサイズ	約 500 バイト	本 FIT モジュールのパッケージに同梱されているサンプルプロジェクトで測定した値です。(5.2 節を参照) この値は、本 FIT モジュールが呼び出す他の FIT モジュールのスタック使用量を含みます。

2.9 引数

本 FIT モジュールの API 関数の引数では、以下の構造体が使用されています。以下の構造体は、lwIP FIT モジュール (lwIP) の netif.h で定義されています。

- netif /* lwIP FIT モジュール (lwIP) 汎用データ構造体 */

2.10 戻り値

本 FIT モジュールの API 関数の戻り値では、以下の列挙型が使用されています。下記の列挙型は、r_lwip_driver_rx_if.h で定義されています。

```
typedef enum       /* 本 FIT モジュールの API 関数のエラーコード*/
{
    LWIP_DRV_SUCCESS            =  0, /* 正常終了 */
    LWIP_DRV_ERR_OPEN           = -1, /* 既に R_LWIP_DRIVER_Open() 関数を実行済み */
    LWIP_DRV_ERR_NOT_OPEN       = -2, /* R_LWIP_DRIVER_Open() 関数を実行していない */
    LWIP_DRV_ERR_ARG            = -3, /* 不正な引数 */
    LWIP_DRV_ERR_CMT            = -4, /* CMT FIT モジュールのエラー */
    LWIP_DRV_ERR_RTOS           = -5  /* RTOS のエラー */
} e_lwip_drv_return_t;
```

下記の列挙型は、lwIP FIT モジュール (lwIP) の err.h で定義されています。

- err_enum_t /* lwIP FIT モジュール (lwIP) の API 関数の戻り値 err_t の値の定義 */

2.11 FIT モジュールの追加方法

本 FIT 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)、(5)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 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
e^2 studio のスマート・コンフィグレータを使用して、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「RX スマート・コンフィグレータ ユーザーガイド: e^2 studio 編 (R20AN0451)」を参照してください。

(2) e^2 studio 上で FIT コンフィグレータを使用して FIT モジュールを追加する場合
e^2 studio の FIT コンフィグレータを使用して、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+上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
CS+上で、スタンドアロン版スマート・コンフィグレータを使用して、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「RX スマート・コンフィグレータ ユーザーガイド: CS+編 (R20AN0470)」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。

(5) IAREW 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
スタンドアロン版スマート・コンフィグレータを使用して、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「RX スマート・コンフィグレータ ユーザーガイド: IAREW 編 (R20AN0535)」を参照してください。

2.12 for 文、while 文、do while 文について

本 FIT モジュールでは、レジスタの反映待ち処理などで 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_LWIP_DRIVER_Open()

本 FIT モジュールの初期化を行います。

Format

```
e_lwip_drv_return_t R_LWIP_DRIVER_Open(
    uint32_t *p_cmt_ch,
    uint16_t ms
)
```

Parameters

p_cmt_ch (OUT)	CMT FIT が取得した CMT チャネルが格納されます
ms (IN)	CMT の割り込み周期を、ミリ秒単位で指定してください。1～1000 の範囲で、かつ 1000 を本引数への設定値で割る除算 (1000 / ms) で余りが出ない値を設定してください。

Return values

LWIP_DRV_SUCCESS	/* 正常終了 */
LWIP_DRV_ERR_OPEN	/* 既に本関数を実行済み */
LWIP_DRV_ERR_ARG	/* 不正な引数 */
LWIP_DRV_ERR_CMT	/* CMT FIT モジュールでエラー発生 */
LWIP_DRV_ERR_RTOS	/* RTOS でエラー発生 */

Properties

r_lwip_driver_if.h にプロトタイプ宣言されています。

Description

CMT FIT モジュールの R_CMT_CreatePeriodic() 関数、およびイーサネット FIT モジュールの R_ETHER_Initial() 関数を呼び出し、CMT および ETHERC の初期化を行います。CMT の初期化は、OS を使用しない場合のみ実行します。

CMT の割り込み周期は、1000 を第 2 引数への設定値で割った値 (1000 / ms) を整数型で設定します。カウント精度の誤差が出ないようにするため、第 2 引数へは割り切れる値を設定することを推奨します。

Reentrant

不可

Thread Safety

非対応

Examples

```
e_lwip_drv_return_t rx_lwip_drv_ret;
uint32_t cmt_ch;

#if R_LWIP_DRIVER_USE_TSIP
R_TSIP_Open(NULL, NULL);
#endif /* R_LWIP_DRIVER_USE_TSIP */

rx_lwip_drv_ret = R_LWIP_DRIVER_Open(&cmt_ch, 1);

/* Initialize the lwIP stack */
lwip_init();
```

Special Notes

lwIP FIT モジュールの API 関数、および本 FIT モジュールの他の API 関数を呼び出す前に、本関数を実行してください。

また、使用している MCU に Trusted Secure IP (TSIP) が搭載されている場合、本関数を実行する前に TSIP FIT モジュール (r_tsip_rx) の R_TSIP_Open() 関数をユーザアプリケーションで呼び出してください。

3.2 R_LWIP_DRIVER_Close()

lwIP Driver FIT モジュールで取得したリソースを解放します。

Format

```
e_lwip_drv_return_t R_LWIP_DRIVER_Close(
    uint32_t cmt_ch
)
```

Parameters

cmt_ch (IN) R_LWIP_DRIVER_Open() 関数で与えられた CMT のチャネルを指定してください

Return values

LWIP_DRV_SUCCESS	/* 正常終了 */
LWIP_DRV_ERR_NOT_OPEN	/* R_LWIP_DRIVER_Open() 関数を実行していない */
LWIP_DRV_ERR_CMT	/* CMT FIT モジュールでエラー発生 */

Properties

r_lwip_driver_if.h にプロトタイプ宣言されています。

Description

CMT FIT モジュールの R_CMT_Stop() 関数を呼び出し、リソースの解放を行います。本関数の処理は、OS を使用しない場合のみ実行されます。

Reentrant

不可

Thread Safety

非対応

Examples

```
e_lwip_drv_return_t rx_lwip_drv_ret;
uint32_t cmt_ch = 0;

rx_lwip_drv_ret = R_LWIP_DRIVER_Close(cmt_ch);
```

Special Notes

本関数を呼び出す前に R_LWIP_DRIVER_EthernetClose() 関数を実行して、使用した ETHERC の送受信機能を停止してください。

3.3 R_LWIP_DRIVER_EthernetClose()

ETHERC の送受信機能を停止します。

Format

```
e_lwip_drv_return_t R_LWIP_DRIVER_EthernetClose(
    struct netif *netif
)
```

Parameters

netif (IN)	ユーザが宣言した netif 変数

Return values

LWIP_DRV_SUCCESS	/* 正常終了 */
LWIP_DRV_ERR_ARG	/* 不正な引数 */

Properties

r_lwip_driver_if.h にプロトタイプ宣言されています。

Description

イーサネット FIT モジュールの R_ETHER_CheckWrite() 関数を呼び出して、本関数の引数へ設定された netif 構造体のメンバ name[1]が指定する Ethernet チャネルの送信完了を待った後、R_ETHER_Close_ZC2() 関数を呼び出してその Ethernet チャネルに対応するリソースの解放を行います。

R_LWIP_DRIVER_Close() 関数を呼び出す前に、必ず本関数を呼び出してください。

複数のネットワークインタフェースを使用する場合は、R_LWIP_DRIVER_Close() 関数を呼び出す前に本 API を実行して、使用した全ての Ethernet チャネルの送受信機能を停止してください。

Reentrant

不可

Thread Safety

非対応

Examples

```
e_lwip_drv_return_t rx_lwip_drv_ret;

rx_lwip_drv_ret = R_LWIP_DRIVER_EthernetClose(&gnetif);

rx_lwip_drv_ret = R_LWIP_DRIVER_Close(cmt_ch);
```

Special Notes

本 API は、イーサネットに異常が発生した時に、イーサネット FIT モジュール及び PHY の初期化を実行する場合に使用できます。

本関数を呼び出した後、lwIP FIT モジュールの API を使用してイーサネットインタフェースの初期化を実行してください。イーサネット FIT モジュール、ETHERC およびイーサネット PHY を一度クローズして、その後にイーサネットインタフェースの初期化を実行する場合の実装例を以下に示します。

```
R_LWIP_DRIVER_EthernetClose(&gnetif);
netif_remove(&gnetif);
netif_add(&gnetif, &ipaddr, &netmask, &gw, NULL, &r_lwip_driver_ethernetif_init,
&netif_input);

/* Check if the ethernet link up */
if(ETHER_SUCCESS == R_ETHER_CheckLink_ZC(LWIP_DRIVER_CFG_ETH_DRV_CH))
{
    /* Link is up */
    netif_set_default(&gnetif);
    netif_set_up(&gnetif);

}
```

3.4 R_LWIP_DRIVER_EthernetLinkCheck()

イーサネット PHY のリンク状態の確認を行います。

Format

```
void R_LWIP_DRIVER_EthernetLinkCheck(
    struct netif *netif
)
```

Parameters

netif (IN)　　ユーザが宣言した netif 変数

Return values

なし

Properties

r_lwip_driver_if.h にプロトタイプ宣言されています。

Description

イーサネット PHY の状態を読み出し、読み出されたイーサネットのリンク状態に応じて以下の処理を実行します。

読み出されたイーサネットのリンク状態がリンクアップ、かつ lwIP FIT モジュール (lwIP) の保持するリンク状態がリンクダウンだった場合は、lwIP の netif_set_link_up() 関数を呼び出して lwIP の保持するリンク状態をリンクアップ状態にします。一方、読み出されたイーサネットのリンク状態がリンクダウン、かつ lwIP の保持するリンク状態がリンクアップだった場合は、lwIP の netif_set_link_down() 関数を呼び出して lwIP の保持するリンク状態をリンクダウン状態にします。

本関数は、lwIP の保持するリンク状態を更新するため、メインループで繰り返し呼び出してください。

表 3.1 本関数の処理

	イーサネットのリンク状態	
lwIP のリンク状態	**リンクアップ**	**リンクダウン**
リンクアップ	処理なし	netif_set_link_down() 関数を呼び出す
リンクダウン	netif_set_link_up() 関数を呼び出す	処理なし

Reentrant

不可

Thread Safety

非対応

Examples

```
/* Global */
struct netif gnetif;
uint32_t EthernetLinkTimer = 0;

void Ethernet_Link_Periodic_Handle(struct netif *netif)
{
    /* Check the ethernet link every 100ms */
    if ((sys_now() - EthernetLinkTimer) >= 100)
    {
        EthernetLinkTimer = sys_now();
        R_LWIP_DRIVER_EthernetLinkCheck(netif);
    }
}

void main()
{
    uint32_t counts = 0;

    netif_set_default(&gnetif);
    netif_set_up(&gnetif);

    while(1)
    {
        Ethernet_Link_Periodic_Handle(&gnetif);

        sys_check_timeouts();

        if (netif_is_link_up(&gnetif) && (counts == 0))
        {
            counts++;
            tcp_echoclient_connect();
        }
    }
}
```

Special Notes

本 API でリンク状態を確認するイーサネット PHY のチャネルは、本関数の引数に設定された netif 構造体のメンバ name[1] (netif->name[1]) で指定されます。netif->name[1]は、r_lwip_driver_ethernetif_init() 関数で初期化されます。ユーザは、netif->name[1]の値を直接変更しないでください。

OS を使用してマルチスレッド環境で本 FIT モジュールを使用する場合、本関数で使用する netif_set_link_up() および netif_set_link_down() 関数と lwIP コアスレッドとの排他制御をするため、それぞれ代わりに netifapi_netif_set_link_up() および netifapi_netif_set_link_down() 関数を使用します。詳細は、5.4 節を参照してください。

3.5 r_lwip_driver_ethernetif_init()

netif 構造体の初期化、およびイーサネット FIT モジュールの初期化処理を行います。

Format

```
err_t r_lwip_driver_ethernetif_init(
    struct netif *netif
)
```

Parameters

netif (IN/OUT)　　ユーザが宣言した netif 構造体

Return values

ERR_OK	/* 正常終了 */
ERR_IF	/* イーサネット FIT モジュールでエラー発生 */

Properties

r_lwip_driver_if.h にプロトタイプ宣言されています。

Description

引数 netif へ設定された構造体の初期化を実行します。また、イーサネット FIT モジュールの R_ETHER_Open_ZC2() 関数を呼び出してその初期化を実行します。

本関数では、イーサネット FIT モジュールへ ETHERC-イーサネット PHY 間インタフェース規格の設定を実行します。使用するインタフェース規格は、イーサネット FIT モジュールの ETHER_CFG_MODE_SEL マクロで MII (Media Independent Interface) または RMII (Reduced Media Independent Interface) を選択します。

ユーザは、本関数を直接呼び出さないでください。lwIP FIT モジュール (lwIP) の netif_add() 関数を、その第 6 引数へ本関数の関数ポインタを設定して実行してください。本関数が、netif_add() 関数内で実行されます。

Reentrant

不可

Thread Safety

非対応

Examples

```
netif_add(&gnetif, &ipaddr, &netmask, &gw, NULL, &r_lwip_driver_ethernetif_init,
&netif_input);
```

Special Notes

ETHERC の端子設定は、Smart Configurator の端子設定ウィンドウで実施してください。

複数のネットワークインタフェース (マルチインタフェース) を使用する場合、本関数では 2 チャネル目のネットワークインタフェースの初期化処理を実行しません。ユーザは、本関数の実装を参考にして 2 チャネル目のネットワークインタフェースの初期化処理を実装してください。そのとき、netif 構造体のメンバ name[1] (netif->name[1]) へ 1 チャネル目と異なる値を設定してください。マルチインタフェースを使用する場合についての詳細は、5.3 節を参照してください。

3.6 R_LWIP_DRIVER_Input()

受信したイーサネットフレームの読み出しを行います。

Format

```
void R_LWIP_DRIVER_Input(
    struct netif *netif
)
```

Parameters

netif (IN)	ユーザが宣言した netif 構造体

Return values

なし

Properties

r_lwip_driver_if.h にプロトタイプ宣言されています。

Description

イーサネット FIT モジュールの R_ETHER_Read_ZC2() 関数を呼び出し、受信フレームの読み出しを行います。受信フレームがある場合、本関数の引数に設定された netif 構造体のメンバ input に設定された関数 (netif->input()) を呼び出します。

netif 構造体のメンバ input への関数の登録は、lwIP FIT モジュール (lwIP) の netif_add() 関数で行います。受信フレームの処理に使用する関数を lwIP に用意されているネットワークインタフェース関数 (netif_input_fn) から選択し、その関数ポインタを netif_add() 関数の第 7 引数へ設定して実行してください。詳細は、関連ドキュメント[10]の netif_add() を参照してください。

OS を使用しない場合は、メインループで本関数を繰り返し呼び出してください。一方、OS を使用する場合は、ユーザは本関数を直接呼び出さないでください。その場合、本関数は r_lwip_driver_input_thread() 関数を実行する受信処理タスクで実行されます。

Reentrant

不可

Thread Safety

非対応

Examples

```
/* Infinite loop */
while (1)
{
   /* Read a received packet from the Ethernet buffers and send it
      to the lwIP for handling */
   R_LWIP_DRIVER_Input(&gnetif);

   /* Handle timeouts */
   sys_check_timeouts();

   Ethernet_Link_Periodic_Handle(&gnetif);

#if LWIP_DHCP
   DHCP_Periodic_Handle(&gnetif);
#endif
}
```

Special Notes

なし

3.7 r_lwip_driver_low_level_output()

送信するイーサネットフレームの書き込みを行います。

Format

```
err_t r_lwip_driver_low_level_output(
    struct netif *netif,
    struct pbuf *p
)
```

Parameters

netif (IN/OUT)	ユーザが宣言した netif 構造体
p (IN)	送信データへのポインタ

Return values

ERR_OK	/* 正常終了 */
ERR_IF	/* イーサネット FIT モジュールでエラー発生 */
ERR_ARG	/* 引数に設定された Ethernet チャネル番号が不正 */

Properties

r_lwip_driver_if.h にプロトタイプ宣言されています。

Description

イーサネット FIT モジュールの R_ETHER_Write_ZC2_GetBuf() 関数、および R_ETHER_Write_ZC2_SetBuf() 関数を呼び出して、送信するイーサネットフレームをイーサネット FIT モジュールの送信バッファへ書き込み、そして送信開始処理を実行します。

ユーザは、本関数を直接呼び出さないでください。本関数は、r_lwip_driver_ethernetif_init() 関数で netif 構造体のメンバ linkoutput (netif->linkoutput) へ登録され、lwIP FIT モジュール (lwIP) の処理中に実行されます。

Reentrant

不可

Thread Safety

非対応

Examples

なし

Special Notes

複数のネットワークインタフェースを使用する場合は、2 チャネル目のネットワークインタフェースの初期化処理を実行する関数 (5.3.1.2 節の例では r_lwip_driver_ethernetif_init_2() 関数) で、本関数は netif 構造体のメンバ linkoutput へ登録されます。詳細は、5.3.1.2 節を参照してください。

3.8 r_lwip_driver_get_rand()

乱数生成処理を実行します。

Format

```
uint32_t r_lwip_driver_get_rand(
    void
)
```

Parameters

なし

Return values

32-bit 符号なし整数 /* 生成した乱数 */

Properties

r_lwip_driver_if.h にプロトタイプ宣言されています。

Description

使用する MCU に TSIP が搭載されている場合、本関数は Trusted Secure IP (TSIP) の真正乱数生成回路 (TRNG) を使用した乱数生成処理を実行します。

使用する MCU に TSIP が搭載されていない場合、本関数は標準 C ライブラリ関数の rand() を使用した疑似乱数生成処理を実行します。

Reentrant

不可

Thread Safety

非対応

Examples

なし

Special Notes

なし

3.9 r_lwip_driver_input_thread()

イーサネットフレームの受信処理を実行するタスク関数です。本関数は、OS を使用する場合に使用します。OS を使用しない場合は、本関数は使用しません。

Format

```
void r_lwip_driver_input_thread(
    void * arg
)
```

Parameters

arg (IN) ユーザが宣言した netif 構造体

Return values

なし

Properties

r_lwip_driver_if.h にプロトタイプ宣言されています。

Description

R_LWIP_DRIVER_Input() 関数を呼び出して、イーサネットフレームの受信処理を実行します。

ユーザは、本関数を直接呼び出さないでください。r_lwip_driver_ethernetif_init() 関数から呼び出される r_lwip_driver_low_level_init() 関数で作成されるタスクで実行されます。

Reentrant

不可

Thread Safety

非対応

Examples

なし

Special Notes

複数のネットワークインタフェース（マルチインタフェース）を使用する場合、本関数では 2 チャネル目のネットワークインタフェースの受信処理を実行しません。ユーザは r_lwip_driver_ethernetif_init() 関数および r_lwip_driver_low_level_init() 関数の実装を参考にして、2 チャネル目の受信処理を実行するタスクを作成する処理を実装してください。

4. 制限事項

4.1 制限事項

本 FIT モジュールの制限事項を、以下に示します。

- 本 FIT モジュールは netif 構造体の netif->name[1]を、イーサネット PHY のチャネル番号の管理に使用しています。ASCII 文字コードで、“0” (0x30) または”1” (0x31) を設定してください。複数のネットワークインタフェースを使用する場合の詳細は、5.3.1.2 節を参照してください。

5. 付録

5.1 動作確認環境

本 FIT モジュールの動作確認環境を、表 5.1 に示します。

表 5.1 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 e^2 studio 2025-12
コンパイラ	CC-RX	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.07.00 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
	GCC	GCC for Renesas RX 14.2.0.202505
エンディアン		リトルエンディアン／ビッグエンディアン*1
本 FIT モジュールのリビジョン		Rev1.11
使用ボード		Renesas CK-RX65N v2 (型名：RTK5CK65N0S08xxxBE) Renesas RX72N Envision Kit (型名：RTK5RX72N0C00000BJ) Renesas Starter Kit+ for RX72M (型名：RTK5572MNxSx0000BE)
RTOS	FreeRTOS	10.4.3-rx-1.0.10
FIT	BSP FIT	Ver 7.60
	Ethernet FIT	Ver 1.24
	CMT FIT	Ver 5.71
	TSIP FIT	Ver 1.23
	lwIP FIT	Ver 1.11
プロトコル	インターネット層	IPv4, ICMP, IGMP, ARP, IPv6, ICMPv6
	トランスポート層	TCP, UDP
	アプリケーション層	DHCP, DHCPv6, HTTP (server)

*1)　ベアメタルのみ、ビッグエンディアンをサポートします。

5.2　サンプルプロジェクト

本 FIT モジュールには、表 5.2 に示すサンプルプロジェクトが同梱されています。下記の URL より FIT モジュールのパッケージをダウンロードし、サンプルプロジェクトを入手してください。

URL: https://www.renesas.com/search?keywords=R20AN0788

表 5.2 サンプルプロジェクト (*1)

プロトコル／サービス	OS	プロジェクト名	対応キット
TCP client ICMP (PING) 受信	なし	rx65n_ck_v2_lwip_driver_tcp_client	CK-RX65N v2
		rx72n_evk_lwip_driver_tcp_client	RX72N Envision Kit
		rx72m_rsk_lwip_driver_tcp_client	RSK+ for RX72M*2 *3
	FreeRTOS	rx65n_ck_v2_lwip_driver_tcp_client_freertos	CK-RX65N v2
		rx72n_evk_lwip_driver_tcp_client_freertos	RX72N Envision Kit
		rx72m_rsk_lwip_driver_tcp_client_freertos	RSK+ for RX72M*2 *3
UDP client ICMP (PING) 受信	なし	rx65n_ck_v2_lwip_driver_udp_client	CK-RX65N v2
		rx72n_evk_lwip_driver_udp_client	RX72N Envision Kit
		rx72m_rsk_lwip_driver_udp_client	RSK+ for RX72M*2 *3
	FreeRTOS	rx65n_ck_v2_lwip_driver_udp_client_freertos	CK-RX65N v2
		rx72n_evk_lwip_driver_udp_client_freertos	RX72N Envision Kit
		rx72m_rsk_lwip_driver_udp_client_freertos	RSK+ for RX72M*2 *3
TCP server UDP server ICMP (PING) 受信	FreeRTOS	rx65n_ck_v2_lwip_driver_udptcp_server_freertos	CK-RX65N v2
		rx72n_evk_lwip_driver_udptcp_server_freertos	RX72N Envision Kit
		rx72m_rsk_lwip_driver_udptcp_server_freertos	RSK+ for RX72M*2 *3
TCP/IPv6 client ICMPv6 (PING) 受信	FreeRTOS	rx65n_ck_v2_lwip_driver_tcp_client_ipv6_freertos	CK-RX65N v2
HTTP server	FreeRTOS	rx65n_ck_v2_lwip_driver_http_server_freertos	CK-RX65N v2

*1) コンパイラは、ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family (CC-RX) を使用します。

*2) Renesas Starter Kit+ (RSK+) for RX72M は、イーサネット 2 チャネルのマルチインタフェースをサポートしています。

*3) 本 FIT モジュールは、複数のネットワークインタフェース（マルチインタフェース）を使用する場合の API は、1 チャネル目のみ対応しています。
lwIP FIT モジュールは、マルチインタフェースに対応しています。ただし、2 チャネル目の ETHERC の初期化、およびクローズ処理を実行する API 関数は、本 FIT モジュールでは提供されません。3.2 節および 3.5 節に記載の通り、ユーザがそれらを実行する処理を実装してください。本 FIT モジュールに同梱されている RSK+RX72M のサンプルコードを参考にすることができます。

5.3 マルチインタフェース対応

lwIP FIT モジュールは、複数のネットワークインタフェース (マルチインタフェース) を使用することができます。RX ファミリの MCU では、RX72M などの ETHERC およびイーサネット PHY を 2 チャネル持つ製品で、lwIP でイーサネットを 2 チャネル使用することができます。

ただし、本 FIT モジュールには lwIP でマルチインタフェースを使用する場合に必要となる、2 チャネル目のネットワークインタフェースを使用するための処理が実装されていません。5.3.1 節に示す実装方法のガイドおよび RSK+RX72M で動作するマルチインタフェースのサンプルプロジェクト (5.2 節を参照) を参考にして、ユーザが実装してください。

OS を使用する場合は、2 チャネル目のネットワークインタフェースの受信処理を実行するタスクを作成してください。詳細は、5.3.1.3 節を参照してください。

5.3.1 マルチインタフェース対応時に必要な実装

lwIP FIT モジュールで、マルチインタフェースを使用する場合に必要となる処理の実装方法を示します。本 FIT モジュールの一部の API 関数及びそれらから呼び出されている関数をコピーおよびリネームして、2 チャネル目のネットワークインタフェースを使用するための API 関数を作成します。

本節の例では、1 チャネル目のネットワークインタフェースのチャネル番号を 0、そして 2 チャネル目を 1 とします。(5.3.1.1 節を参照)

注意事項：

1) 本節のガイドで追加するマクロ定義および関数は、ユーザアプリケーションへ実装してください。それらを本 FIT モジュールのソースコードへ実装した場合、Smart Configurator の自動コード生成によりユーザが追加した実装が本 FIT モジュールパッケージのソースコードで上書きされて削除される場合があります。
2) ネットワークインタフェースを 2 チャネル使用する場合のチャネル番号は、0 および 1 を使用してください。1 チャネル目のチャネル番号が 0 の場合は、2 チャネル目は 1 を使用してください。
イーサネットを使用する場合は、イーサネット FIT モジュールの仕様により 2 チャネルまで使用できます。

5.3.1.1 2 チャネル目のネットワークインタフェースのチャネル番号のマクロを定義する

- ユーザアプリケーションで、ETHER_CH_2 マクロを定義する。
- 本節の例では、2 チャネル目のネットワークインタフェースのチャネル番号を 1 とする。

```
#define ETHER_CH_2      (1)
```

5.3.1.2 r_lwip_driver_ethernetif_init_2() 関数の作成

- r_lwip_driver_ethernetif_init() 関数をコピーし、コピーした関数の関数名をリネームする。本節の例では、r_lwip_driver_ethernetif_init_2 にリネームする。
- netif->name[1]への設定値を、” (ETHER_CH_2 + '0')” に変更する。(13 行目)

```
err_t r_lwip_driver_ethernetif_init_2(struct netif *netif)
{
    err_t err = ERR_OK;

    LWIP_ASSERT("netif != NULL", (netif != NULL));

#if LWIP_NETIF_HOSTNAME
    /* Initialize interface hostname */
    netif->hostname = "lwip_rx";
#endif /* LWIP_NETIF_HOSTNAME */

    netif->name[0] = 'c';
    netif->name[1] = (char) (ETHER_CH_2 + '0');
    /* We directly use etharp_output() here to save a function call.
     * You can instead declare your own function an call etharp_output()
     * from it if you have to do some checks before sending (e.g. if link
     * is available...) */
#if LWIP_IPV4
    netif->output     = etharp_output;
#endif /* LWIP_IPV4 */
#if LWIP_IPV6
    netif->output_ip6 = ethip6_output;
#endif /* LWIP_IPV6 */
    netif->linkoutput = r_lwip_driver_low_level_output;

    /* initialize the hardware */
    err = r_lwip_driver_low_level_init_2(netif);

    return err;
}
```

5.3.1.3 r_lwip_driver_low_level_init_2() 関数の作成

- r_lwip_driver_low_level_init() 関数をコピーし、コピーした関数の関数名をリネームする。本節の例では、r_lwip_driver_low_level_init_2 にリネームする。
- 変数 eth_ch に設定するチャネル番号を、ETHER_CH_2 に変更する。(5 行目)
- netif->hwaddr へ設定する MAC アドレスを、2 チャネル目のものに変更する。(11 - 16 行目)
- イーサネット FIT モジュールのコールバック関数 eth_param.ether_callback.pcb_func および eth_param.ether_int_hnd.pcb_int_hnd の設定をコメントアウトする。(25 – 42 行目)
  この処理は 1 チャネル目の処理の r_lwip_driver_low_level_init() 関数で実行されるため、2 チャネル目では不要。

```
static err_t r_lwip_driver_low_level_init_2(struct netif *netif)
{
    ether_param_t eth_param = { 0 };
    ether_return_t ether_ret;
    uint32_t eth_ch = ETHER_CH_2;

    /* Clear the link status flags */
    g_link_change[eth_ch] =  LWIP_DRIVER_LINK_CHANGE_FLAG_OFF;
    g_link_status[eth_ch] =  LWIP_DRIVER_LINK_STATUS_FLAG_OFF;

    /* set MAC hardware address length */
    netif->hwaddr_len = ETH_HWADDR_LEN;

    /* set MAC hardware address */
    netif->hwaddr[0] = XXXX_ADDR0;
    netif->hwaddr[1] = XXXX_ADDR1;
    netif->hwaddr[2] = XXXX_ADDR2;
    netif->hwaddr[3] = XXXX_ADDR3;
    netif->hwaddr[4] = XXXX_ADDR4;
    netif->hwaddr[5] = XXXX_ADDR5;

    /* maximum transfer unit */
    netif->mtu = LWIP_DRIVER_CFG_MTU;

    /* device capabilities */
    /* don't set NETIF_FLAG_ETHARP if this device is not an ethernet one */
    netif->flags |= NETIF_FLAG_BROADCAST | NETIF_FLAG_ETHARP;
#if LWIP_IGMP
    netif->flags |= NETIF_FLAG_IGMP;
#endif /* LWIP_IGMP */
#if LWIP_IPV6
    netif->flags |= NETIF_FLAG_MLD6;
#endif /* LWIP_IPV6 */

#if !NO_SYS
    /* Start input thread. */
    sys_thread_new("lwip_ether_input_thread_2", r_lwip_driver_input_thread,
                   netif, DEFAULT_THREAD_STACKSIZE, DEFAULT_THREAD_PRIO);
#endif

//    eth_param.ether_callback.pcb_func = r_lwip_driver_rx_ether_cb;
//
//    ether_ret = R_ETHER_Control(CONTROL_SET_CALLBACK, eth_param);
//
//    if (ETHER_SUCCESS != ether_ret)
//    {
//        return ERR_IF;
//    }
//
```

```
//    /* Set callback function called by EINT status interrupts. */
//    eth_param.ether_int_hnd.pcb_int_hnd = r_lwip_driver_rx_ether_int_cb;
//
//    ether_ret = R_ETHER_Control(CONTROL_SET_INT_HANDLER, eth_param);
//
//    if (ETHER_SUCCESS != ether_ret)
//    {
//        return ERR_IF;
//    }

    eth_param.channel = eth_ch;

    R_ETHER_Control(CONTROL_POWER_ON, eth_param);
    --- 省略 ---

    return ERR_OK;
}
```

5.4 OS 使用時の注意事項

lwIP の NO_SYS オプションを 0 に設定して OS を使用する場合、注意事項があります。

(1) FreeRTOS を使用する場合、イーサネット FIT モジュールが使用するグループ AL1 割り込みの優先レベルを設定する ETHER_CFG_AL1_INT_PRIORTY オプションの値は、FreeRTOS の configMAX_SYSCALL_INTERRUPT_PRIORITY オプションより小さい値に設定してください。本 FIT モジュールは、EDMAC からの割り込み (EINT) で呼び出される割り込みサービスルーチンで FreeRTOS の API 関数を実行します。詳細は、関連ドキュメント[13]を参照してください。

(2) lwIP の raw API などのスレッドセーフではない関数 (非スレッドセーフ関数) をマルチスレッド環境で実行する場合、それらの呼び出し処理が lwIP コアスレッド (tcpip_thread) の処理と競合しないようにする必要があります。tcpip_callback() などの OS mode の関数を使用して非スレッドセーフ関数を lwIP コアスレッドのコンテキストから呼び出す、または非スレッドセーフ関数を呼び出す処理と lwIP コアスレッド間の排他制御を実装するなどの方法により、競合が発生しないようにユーザアプリケーションを実装してください。詳細は、関連ドキュメント[10]の Multithreading を参照してください。

5.5 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：大きいサイズのデータを送信すると、netif 構造体のメンバ linkoutput (netif->linkoutput) に登録された関数の返り値で、lwIP の *ERR_IF* エラーが返される場合があります。 (3.7 節および 5.3.1.2 節を参照)

A：送信データサイズが大きいため、一定のリトライ期間中に送信バッファを確保できずエラーになっていると考えられます。その場合、以下のコンフィグ設定値を大きくすることで改善する場合があります。Smart Configurator を使用して、イーサネット FIT モジュールおよび本 FIT モジュールの以下のコンフィグ設定値を大きくすることで改善する場合があります。

- イーサネット FIT モジュール
  - ETHER_CFG_EMAC_TX_DESCRIPTORS
- lwIP Driver FIT モジュール (本 FIT モジュール)
  - LWIP_DRIVER_CFG_SEND_MAX_LOOP
  - LWIP_DRIVER_CFG_SEND_DELAY_US

ただし、LWIP_DRIVER_CFG_SEND_DELAY_US マクロへ大きな値を設定すると、通信スループットが低下する場合があります。そのため、できるだけ小さい値を設定することを推奨します。

(3) Q：プロジェクトの smc_gen / r_[ドライバ名]_rx フォルダ以下のファイルを編集したが、プロジェクトのビルドを行うと編集した内容が消えている。 (編集したファイルが元の状態に戻っている)

A：Smart Configurator は、デフォルト設定ではビルドなどの特定イベントの発生時に、自動でコード生成 (FIT モジュールパッケージの展開および smc_gen / r_config フォルダに格納されているコンフィグレーションヘッダへのコンフィグの実行) を行います。このコード生成により、ユーザによる編集内容が上書きされて削除されたと考られます。
以下のヘルプの手順により、Smart Configurator の自動コード生成を抑制することができます。

e2 stuido の[Help]メニューバーから[Help Contents]を選択、開いたヘルプ画面の Contents から、[e2 studio User Guide] > [Building Projects] > [Smart Configurator] > [Code Generation]を参照してください。

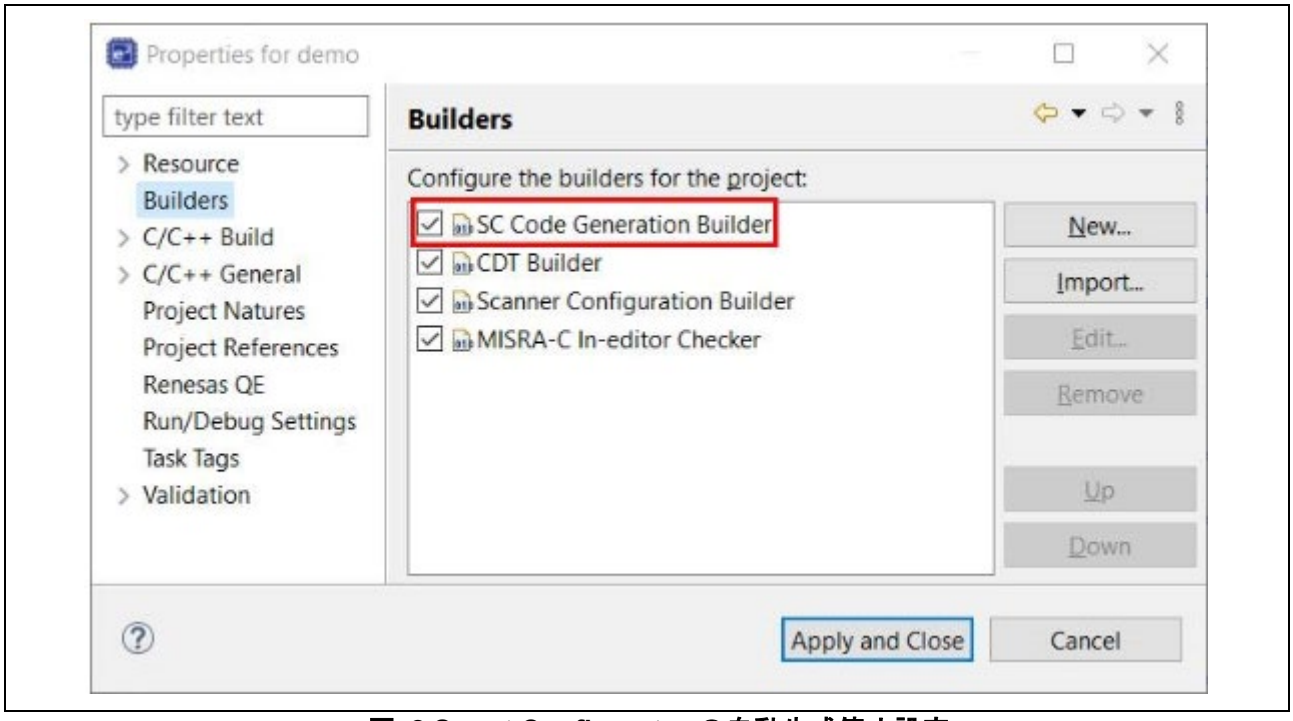

図 2 Smart Configurator の自動生成停止設定

(4) Q：lwIP の lwip_init() 関数を呼び出すと、その処理中に呼び出される Trusted Secure IP (TSIP) FIT モジュールの R_TSIP_GenerateRandomNumber() 関数で無限ループする。

A：TSIP FIT モジュールの R_TSIP_Open() 関数を実行していないことが考えられます。本 FIT モジュールの R_LWIP_DRIVER_Open() 関数を実行する前に、R_TSIP_Open() 関数をユーザアプリケーションで呼び出して実行してください。

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
（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 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。）

GNU-RX Compiler マニュアル
（最新版を下記のホームページから入手してください。）
https://llvm-gcc-renesas.com/ja/gnu-tools-manuals/gnu-compiler/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2025.06.20	-	新規作成
1.10	2025.12.19	P.4	1.2.1 ソフトウェア構成 FreeRTOS を追加
		P.6	OS を使用する場合の説明を追加 ・1.2.2 API の概要
		P.6	1.2.3 対応プロトコル 項目を追加
		P.13 P.15 P.22	OS を使用する場合の説明を追加 ・3.1 R_LWIP_DRIVER_Open() ・3.2 R_LWIP_DRIVER_Close() ・3.6 R_LWIP_DRIVER_Input()
		P.27	API 関数を追加 ・3.9 r_lwip_driver_input_thread()
		P.28	4.1 制限事項 ・一部の制限事項を削除
		P.29	5.1 動作確認環境 ・表 5.1 を更新
		P.30	5.2 サンプルプロジェクト ・表 5.2 にサンプルプロジェクトを追加 ・OS を使用する場合の説明を追加
1.11	2026.03.19	P.19	3.4 R_LWIP_DRIVER_EthernetLinkCheck() ・lwIP をマルチスレッドで使用する場合の注意事項を記載
		P.34	5.4 OS 使用時の注意事項 ・lwIP をマルチスレッドで使用する場合の注意事項を記載
		-	spdx ファイルをパッケージに同梱

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。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、お客様の責任において、お客様の機器・システムを設計ください。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. 当社製品または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. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 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。
6. 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。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 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。
9. 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。
10. 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。
11. 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。
12. 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。
13. 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。
14. 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.5.0-1　2020.10)

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。