RX64M グループ

R01AN2153JJ0100
Rev.1.00
2014.09.01

TCP/IP プロトコルスタックを用いた産業向けネットワークソリューション

RX Driver Package Application

要旨

本資料は、TCP/IP プロトコルスタック（M3S-T4-Tiny）を用いた産業向けネットワークソリューションのアプリケーションノートです。本アプリケーションノートには、Web サーバとモジュールの初期化や駆動処理を行うメインプログラムの各サンプルコードが含まれており、RX64M グループ用 RX Driver Package と組み合わせることで、Web サーバシステムを構築することができます。RX Driver Package と組み合わせて動作するサンプルアプリケーションを総じて、RX Driver Package Application と呼びます。

Web サーバは、TCP/IP 上で動作するアプリケーションプログラムであり、一般的には Web ブラウザからアクセスされ、Web サーバ上に保存されているコンテンツを TCP/IP を用いて Web ブラウザに転送する機能を提供します。

本アプリケーションノートでは、メインプログラムと Web サーバを、RX64M グループ用 RX Driver Package に入っている TCP/IP プロトコルスタック（M3S-T4-Tiny）、Ethernet ドライバ、FAT ファイルシステム（M3S-TFAT-Tiny）、USB ドライバ（ホストマスストレージ）等と組み合わせて評価するまでの手順について説明します。

動作確認デバイス

RX64M グループ（Renesas Starter Kit+ RX64M）

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。

目次

1. 概要

1.1 本アプリケーションノートについて

本資料は、TCP/IP プロトコルスタック（M3S-T4-Tiny）を用いた産業向けネットワークソリューションのアプリケーションノートです。本アプリケーションノートには、Web サーバとモジュールの初期化や駆動処理を行うメインプログラムの各サンプルコードが含まれており、RX64M グループ用 RX Driver Package と組み合わせることで、Web サーバシステムを構築することができます。RX Driver Package と組み合わせて動作するサンプルアプリケーションを総じて、RX Driver Package Application と呼びます。

Web サーバは、TCP/IP 上で動作するアプリケーションプログラムであり、一般的には Web ブラウザからアクセスされ、Web サーバ上に保存されているコンテンツを TCP/IP を用いて Web ブラウザに転送する機能を提供します。

本アプリケーションノートでは、メインプログラムと Web サーバを、RX64M グループ用 RX Driver Package に入っている TCP/IP プロトコルスタック（M3S-T4-Tiny）、Ethernet ドライバ、FAT ファイルシステム（M3S-TFAT-Tiny）、USB ドライバ（ホストマスストレージ）等と組み合わせて評価するまでの手順について説明します。

本アプリケーションノートは、Renesas Starter Kit+ for RX64M（以降、RSK と表記）上で動作します。

1.2　動作環境

本アプリケーションノートの動作環境を以下に示します。

表1.2.1　動作環境

対応 MCU	RX64M グループ
評価ボード	Renesas Starter Kit+ RX64M http://japan.renesas.com/products/tools/introductory_tools/renesas_starter_kits/rsk_plus_rx64m/index.jsp
統合開発環境（IDE）	e² studio V3.0.1.09 以降 または、 CubeSuite+ V2.02.00 以降
クロスツール	RX ファミリ用 C/C++コンパイラパッケージ V2.02.00 以降
エミュレータ	E1（Renesas Starter Kit+ RX64M に同梱）, E20
RX Driver Package	RX64M 用 RX Driver Package Ver1.00（R01AN2144JJ0100）※注

【注】本アプリケーションノートは、上記の RX Driver Package に入っているモジュールと組みわせて動作を確認しています。本アプリケーションノートで使用するモジュールを、別のモジュールと入れかえた場合については、各自で動作を確認してください。

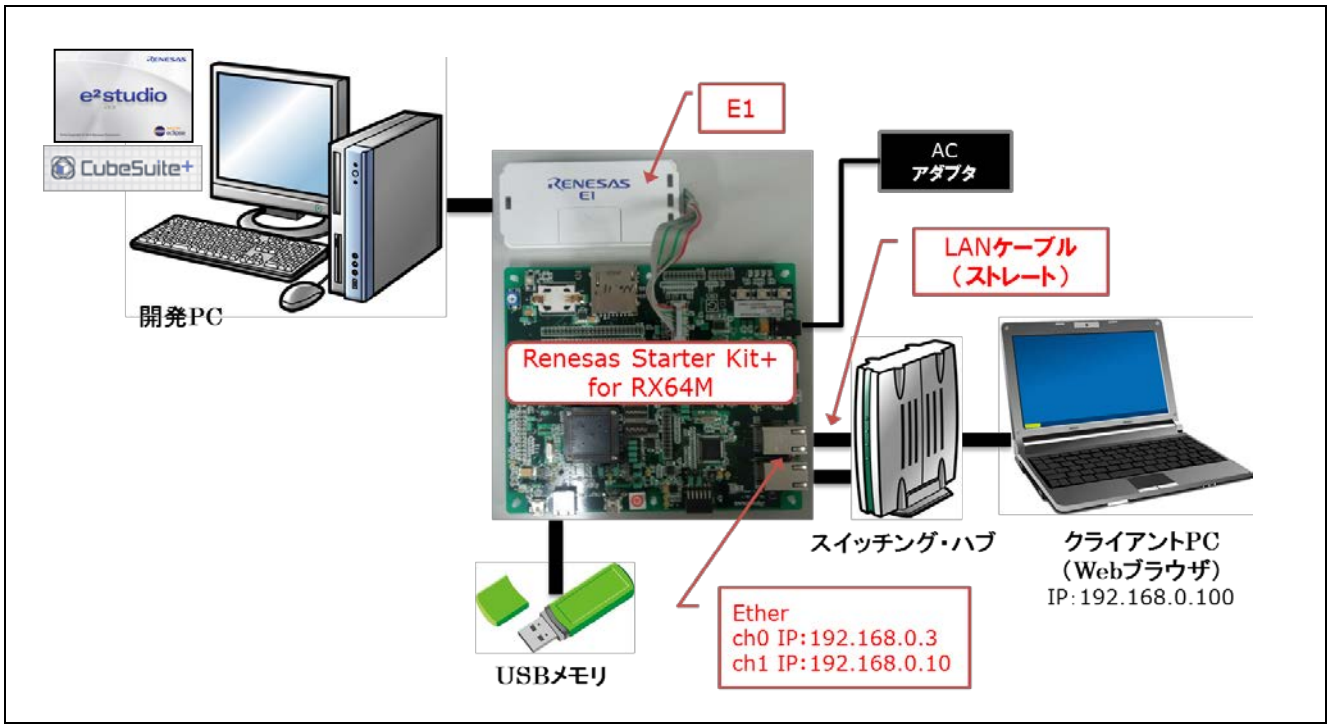


図1.2.1　動作環境の例

1.3　モジュール構成

本アプリケーションノートのモジュール構成とモジュール一覧を以下に示します。

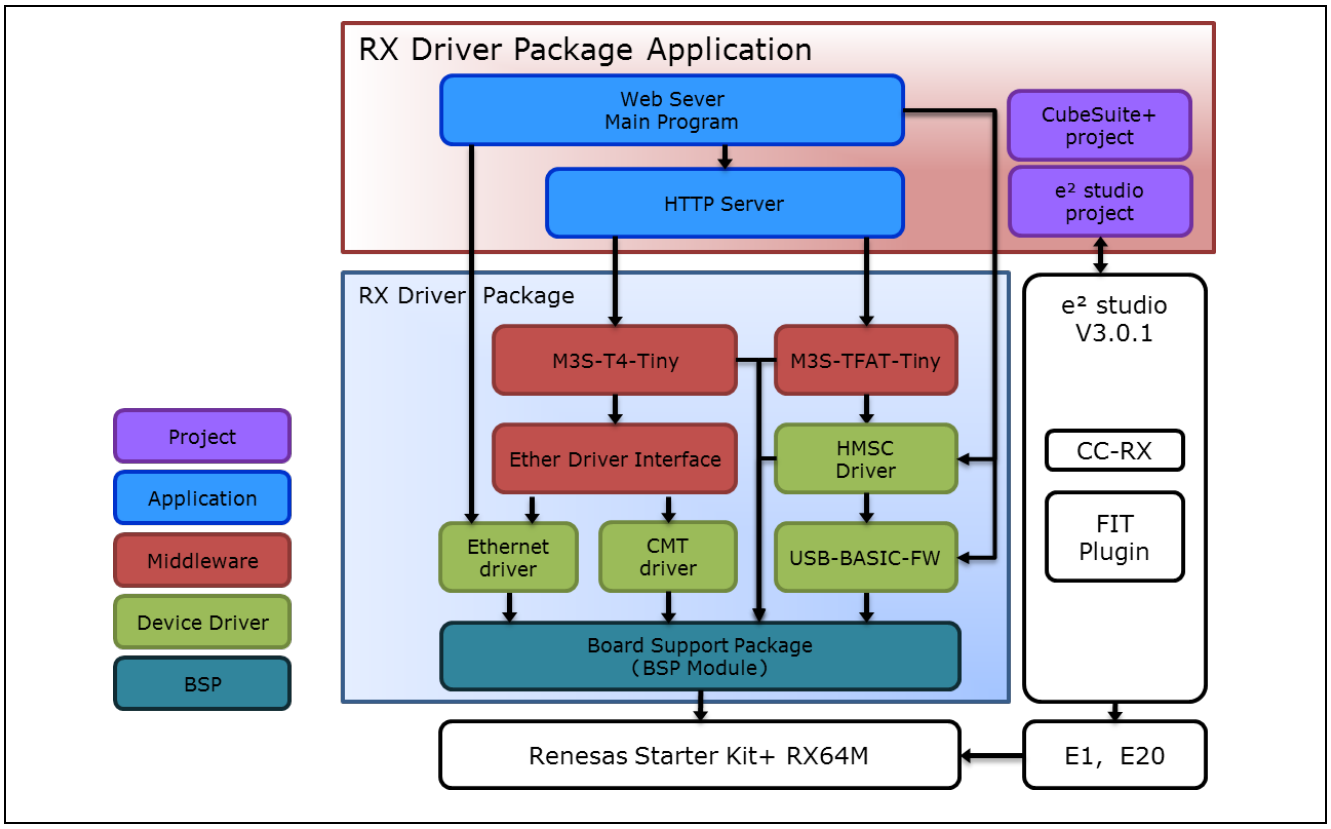


図1.3.1　モジュール構成

表1.3.1　モジュール一覧

種類	モジュール名	FIT モジュール名	バージョン
Board Support Package	ボードサポートパッケージ（BSP モジュール）	r_bsp	2.60
Device Driver	コンペアマッチタイマ (CMT)	r_cmt_rx	2.30
Device Driver	イーサネットコントローラ (ETHERC)	r_ether_rx	1.00
Middleware	M3S-T4-Tiny インタフェース変換モジュール	r_t4_driver_rx64m	1.00
Middleware	TCP/IP プロトコルスタックライブラリ（M3S-T4-Tiny）	r_t4_rx	2.00
Middleware	FAT ファイルシステム（M3S-TFAT-Tiny）	r_tfat_rx	3.00
Device Driver	USB Basic Firmware	r_usb_basic	1.00
Device Driver	USB ホストマスストレージクラス	r_usb_hmsc	1.00
Application	HTTP サーバ	r_t4_http_server_rx	1.03
Application	Web サーバシステムメインプログラム	r_httpd_main_rx64m	1.00

1.4　ファイル構成

本アプリケーションノートのファイル構成を以下に示します。

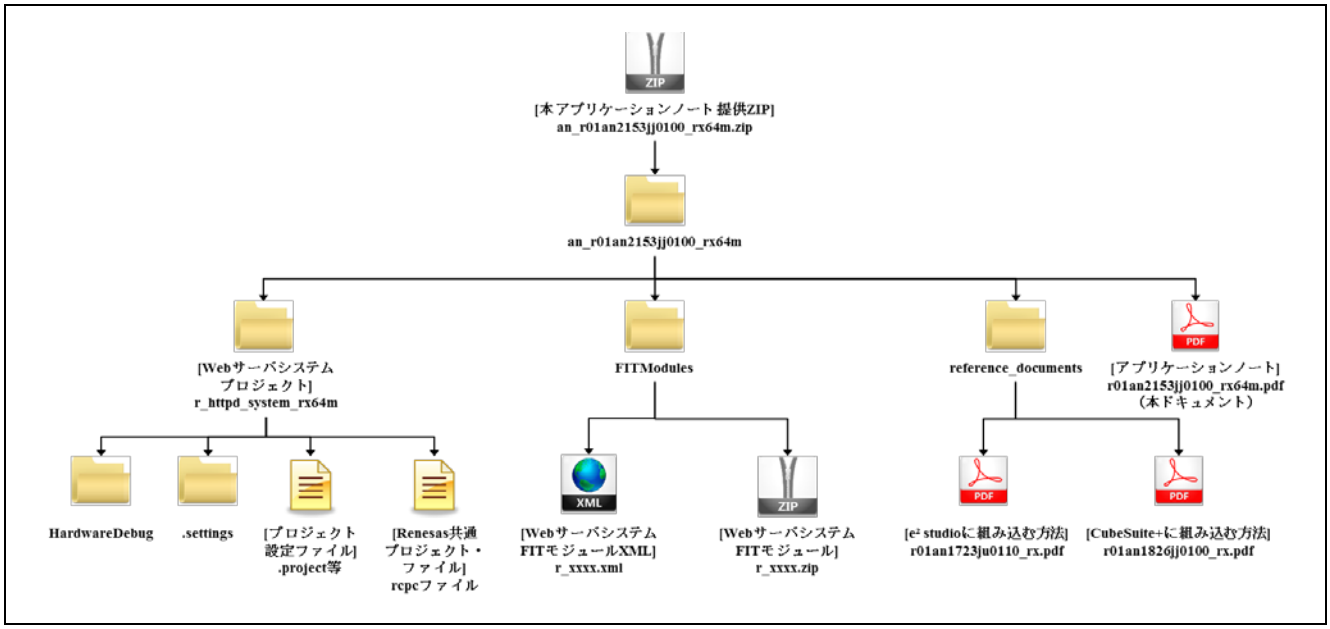


図1.4.1　ファイル構成

本アプリケーションノート提供 ZIP ファイルを解凍すると、同名のフォルダが作成され、その中に各フォルダやファイルが入っています。

「プロジェクト（r_httpd_rx64m）」は、Web サーバシステムを構築するための専用プロジェクトです。e² studio のワークスペースにインポートして使用します。また、CubeSuite+でプロジェクトを読み込むための「Renesas 共通プロジェクト・ファイル」が入っています。

「FITModules」フォルダの中には、Web サーバシステムの FIT モジュールが入っています。

「reference_documents」フォルダの中には、FIT モジュールを各開発環境で使用するための説明書が入っています。「e² studio に取り込む方法（r01an1723ju0110_rx.pdf）」は、FIT プラグインを使用して、FIT モジュールを e² studio のプロジェクトに組み込む方法について説明したドキュメントです。「CubeSuite+に取り込む方法（r01an1826jj0100_rx.pdf）」は、FIT モジュールを CubeSuite+のプロジェクトに組み込む方法について説明したドキュメントです。

「アプリケーションノート（r01an2153jj0100_rx64m.pdf）」は、本ドキュメントです。

1.5　プロジェクトについて

本アプリケーションノートには、Web サーバシステムをビルド及び評価するための、e² studio 及び CubeSuite+用のプロジェクトが付属しています。プロジェクトには、ビルド設定を保存した「ビルド構成（CubeSuite+ではビルド・モード）」と、デバッグ設定を保存した「デバッグ構成（CubeSuite+ではデバッグ・ツール）」を登録しています。

以下に、プロジェクトに登録してあるビルド構成とデバッグ構成の一覧を示します。

表1.5.1　プロジェクト設定

	構成名	説明
ビルド構成 ※CubeSuite+では「ビルド・モード」	HardwareDebug (Debug on hardware)	デバッグ情報付きのロードモジュールを生成するための構成です。 ■主な設定 • デバッグ情報あり • 最適化なし（-optimize=0）
デバッグ構成 ※CubeSuite+では「デバッグ・ツール」	HardwareDebug (E1) ※CubeSuite+では「RX E1(JTAG)」	「HardwareDebug (Debug on hardware)」で生成したロードモジュールを使用して、E1 エミュレータ経由でのハードウェアデバッグを行います。

2.　開発環境の入手

2.1　e² studio の入手とインストール方法

e² studio は、ルネサスのホームページからダウンロードできます。

1. 以下の URL にアクセスし、e² studio のダウンロードページを表示します。
   http://japan.renesas.com/e2studio_download

2. 表示された項目の中から、「統合開発環境 e² studio 3.0.0.22 インストーラ」をクリックします。（分割ダウンロード版と一括ダウンロード版がありますが、内容の違いはありません。）
   その後表示されたページの指示に従い、e² studio のインストーラをダウンロードします。

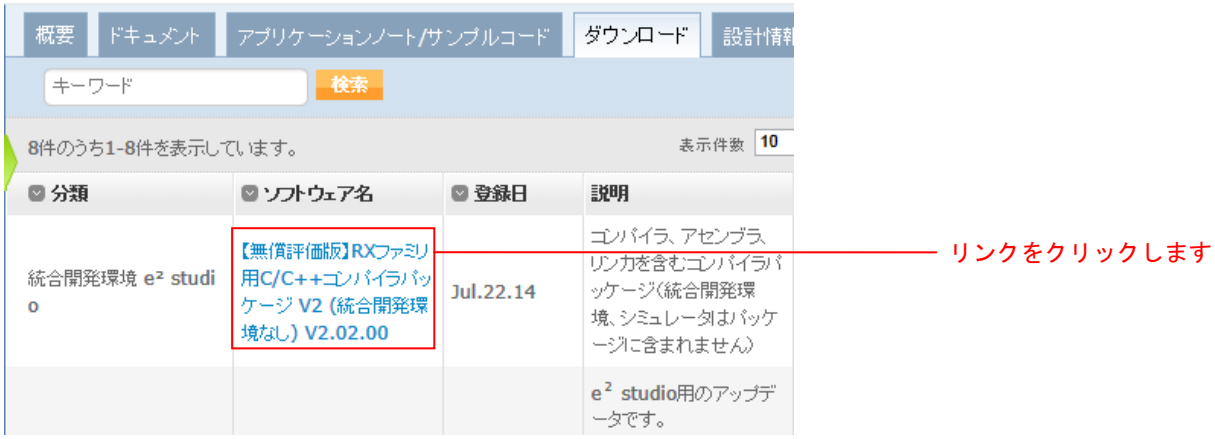
3. ダウンロードした e² studio のインストーラを実行し、e² studio を PC にインストールします。
   インストール方法は「e² studio　統合開発環境　ユーザーズマニュアル入門ガイド」を参照してください。
   http://documentation.renesas.com/doc/products/tool/doc/r20ut2858jj0100_e2_start_s.pdf

2.2　コンパイラパッケージの入手方法

Web サーバシステムをビルドするには、RX ファミリ用 C/C++コンパイラパッケージ V2.02.00 以降が必要です。ここでは、まだ製品版をお持ちでない場合を想定して、無償評価版を利用する例をご紹介します。

1. 以下の URL にアクセスし、e² studio のダウンロードページを表示します。
   http://japan.renesas.com/e2studio_download

2. 表示された項目の中から、「【無償評価版】RX ファミリ用 C/C++コンパイラパッケージ V2（統合開発環境なし）V2.02.00」をクリックします。
   その後表示されたページの指示に従い、コンパイラのインストーラをダウンロードします。



3. ダウンロードしたコンパイラのインストーラを実行し、コンパイラを PC にインストールします。

2.3　Ver.3.0.1.09 へのアップデート

PC にインストールした e² studio を、最新版（現時点では Ver.3.0.1.09）にアップデートします。

1. 以下の URL にアクセスし、e² studio のダウンロードページを表示します。
   http://japan.renesas.com/e2studio_download

2. 表示されたページの右側にあるバージョン情報のリンクをクリックします。

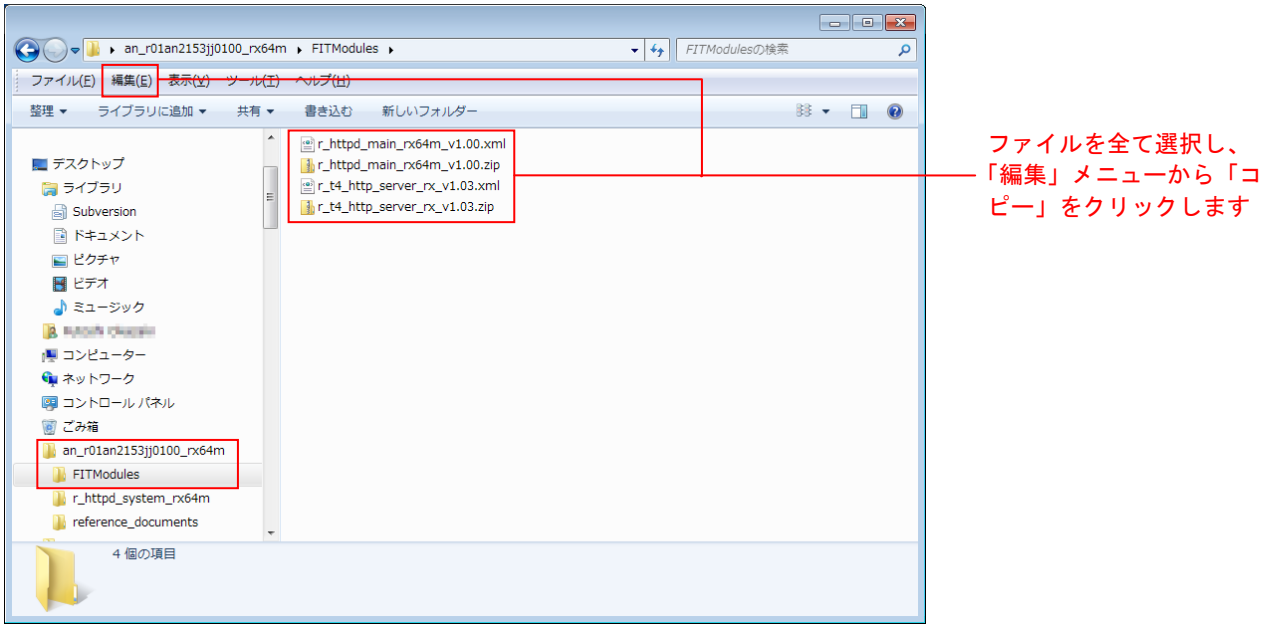
3. 表示された項目の中から、「Eclipse ベース 統合開発環境 e2 studio V3.0.1.09 へのリビジョンアップのお知らせ」をクリックします。
   その後表示されたページの指示に従い、e² studio のアップデートを実施してください。

3.　環境の準備

3.1　FIT モジュールのイントール

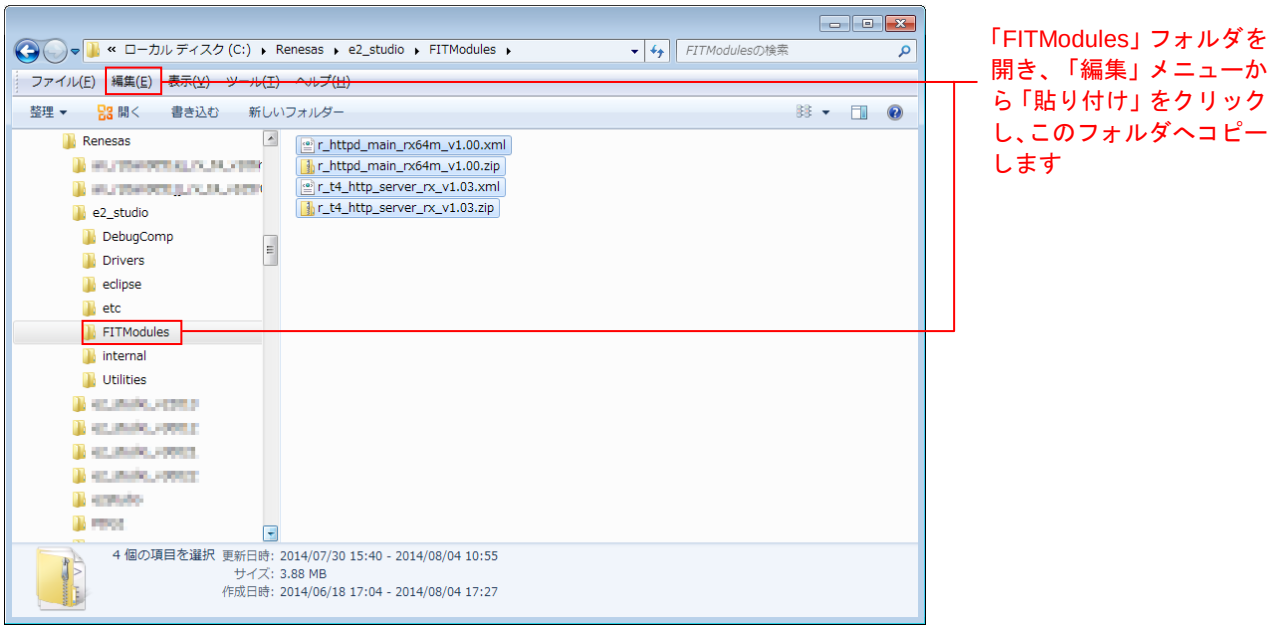
本アプリケーションノートに入っている Web サーバシステムの FIT モジュールを、e² studio にインストールします。

1. 本アプリケーションノート提供 ZIP ファイルを任意のフォルダに解凍します。
2. 解凍してできた同名のフォルダを開き、その中にある「FITModules」フォルダを開きます。
3. 「FITModules」フォルダ内ある全てのファイルを選択し、「編集」メニューから「コピー」をクリックします。



4. e² studio のインストールフォルダ（通常は、C:¥Renesas¥e2_studio です）を開き、その中にある「FITModules」フォルダを開きます。
5. 「編集」メニューから「貼り付け」をクリックします。

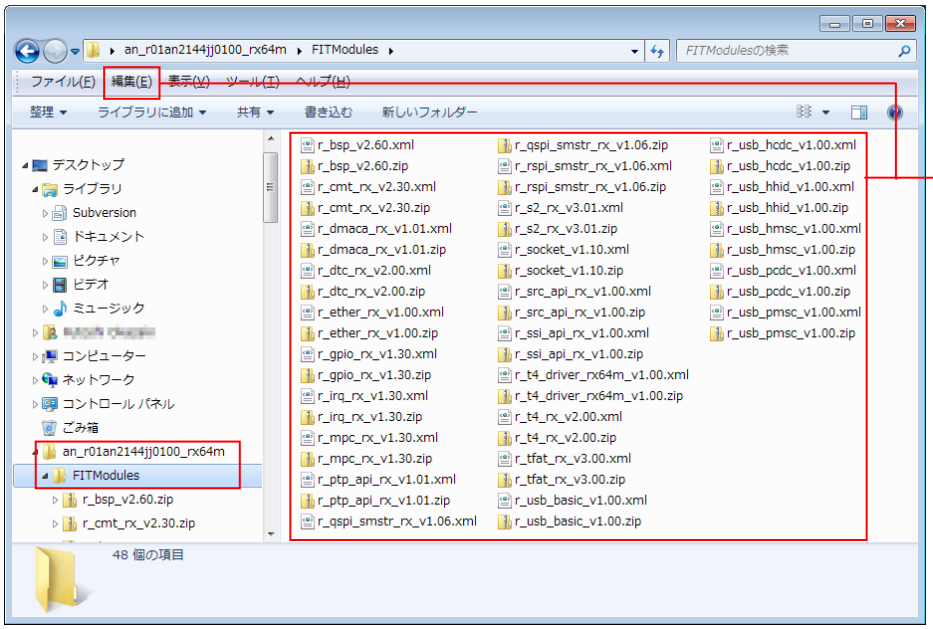
e² studio の「FITModules」フォルダに、FIT モジュールがコピーされます。



3.2　RX Driver Package のイントール

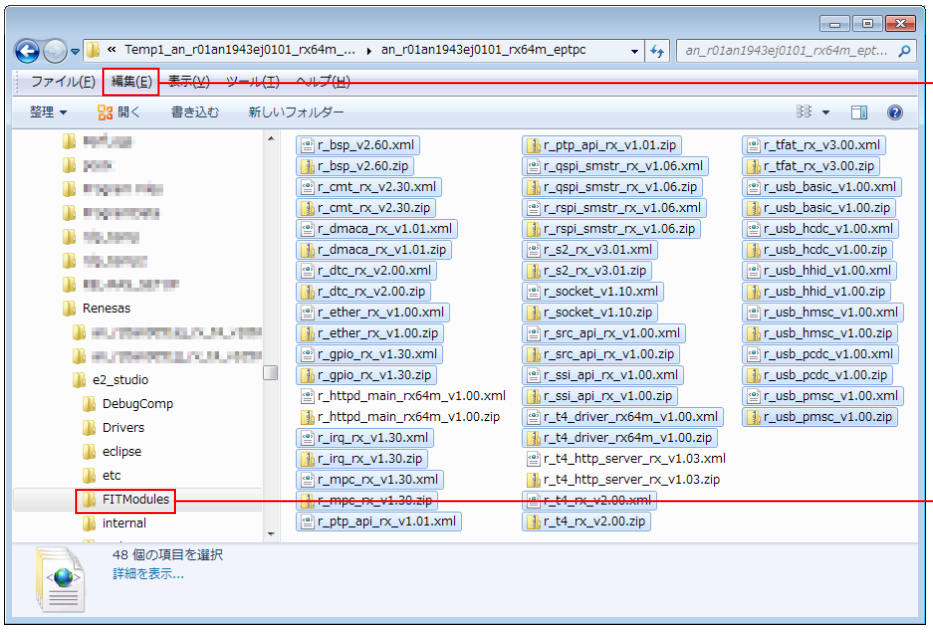
RX64M グループ用 RX Driver Package に入っている FIT モジュールを、e² studio にインストールします。

1. RX64M グループ用 RX Driver Package をダウンロードし、「an_r01an2144jj0100_rx64m.zip」ファイルを、任意のフォルダに解凍します。
2. 解凍してできたフォルダを開き、その中にある「FITModules」フォルダを開きます。
3. 「FITModules」フォルダ内ある全てのファイルを選択し、「編集」メニューから「コピー」をクリックします。



4. e² studio のインストールフォルダ（通常は、C:¥Renesas¥e2_studio です）を開き、その中にある「FITModules」フォルダを開きます。
5. 「編集」メニューから「貼り付け」をクリックします。

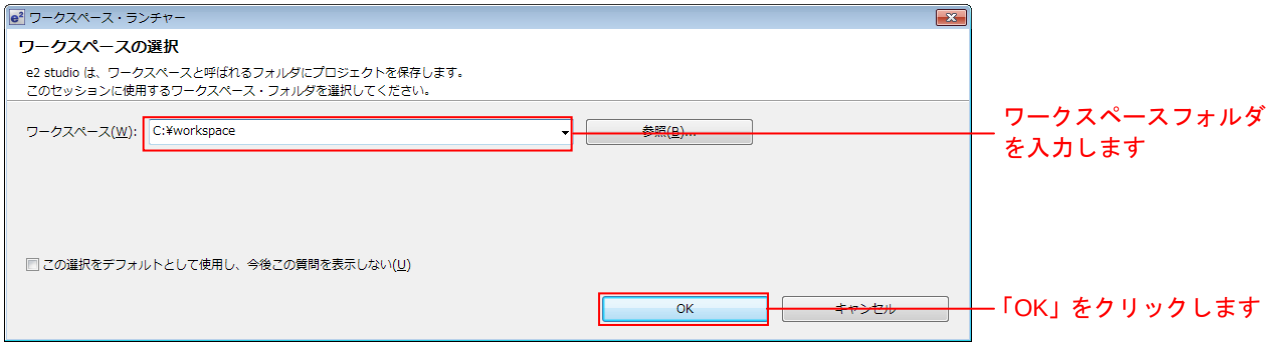
e² studio の「FITModules」フォルダに、FIT モジュールがコピーされます。



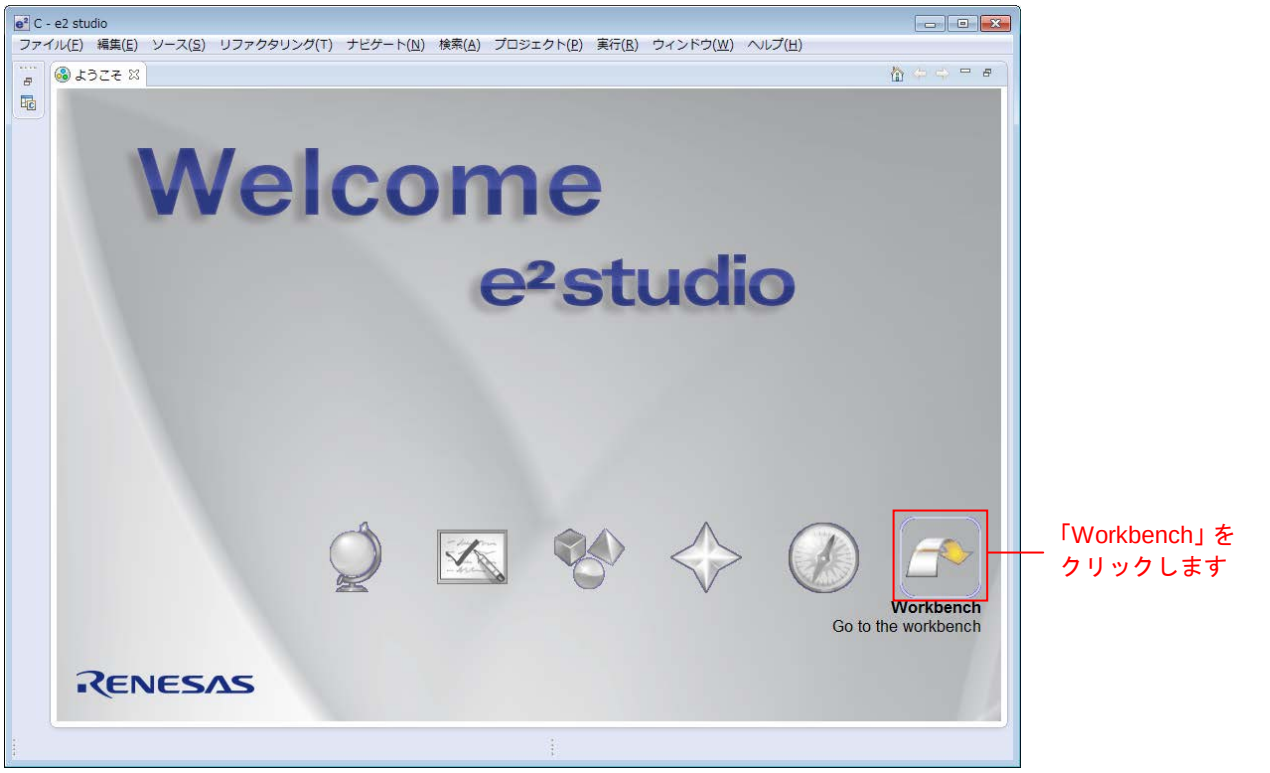
4.　プロジェクトの構築

4.1　ワークスペースの作成

1. e² studio を起動します。
2. 表示されたダイアログに、任意のワークスペースフォルダを入力し、「OK」をクリックします。



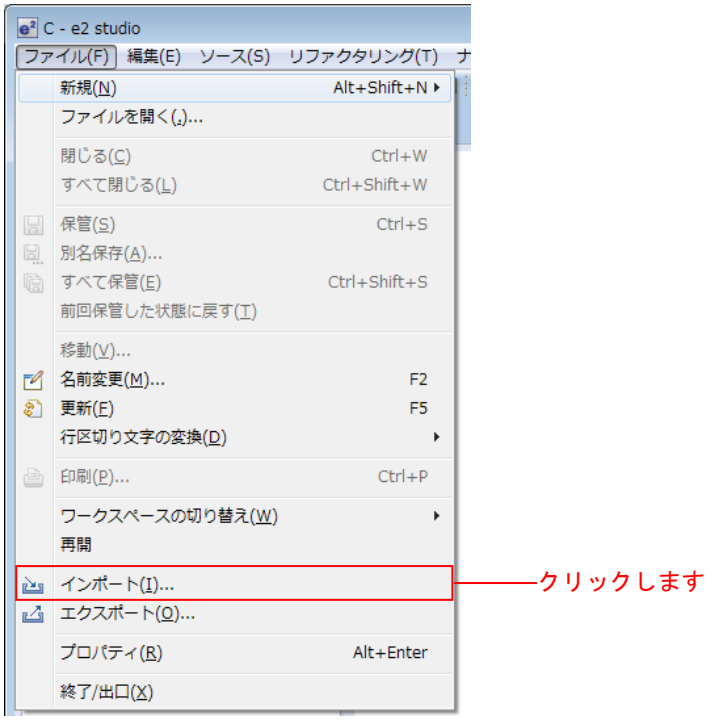
3. 以下の画面が表示されたら、「Workbench」をクリックします。



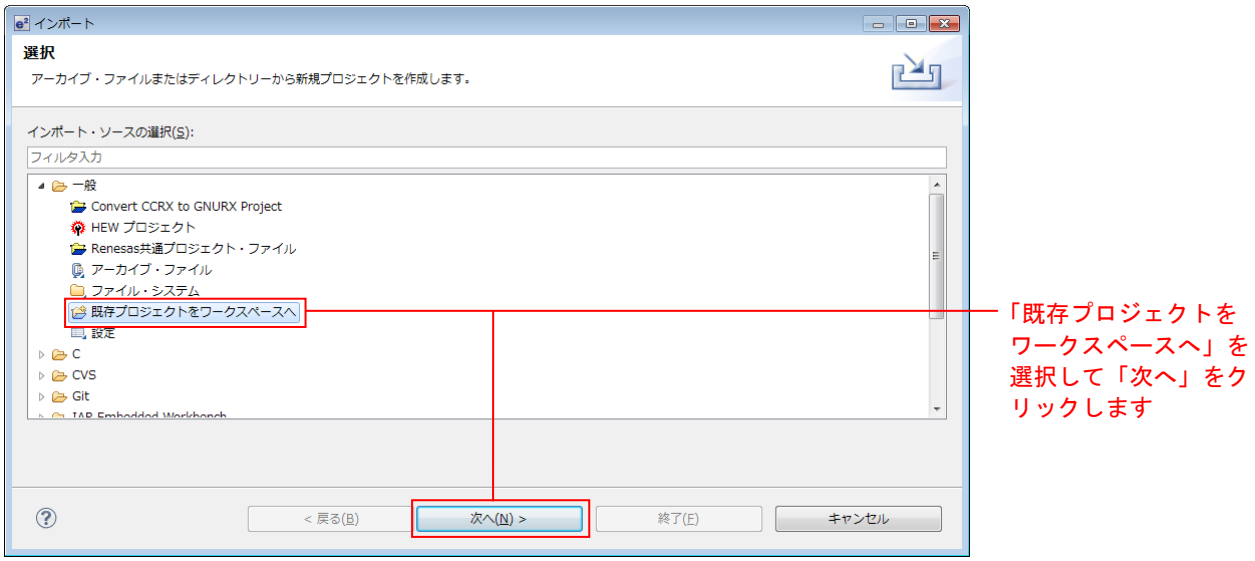
4.2　プロジェクトのインポート

本アプリケーションノートに付属しているプロジェクトを、作成したワークスペースにインポートします。

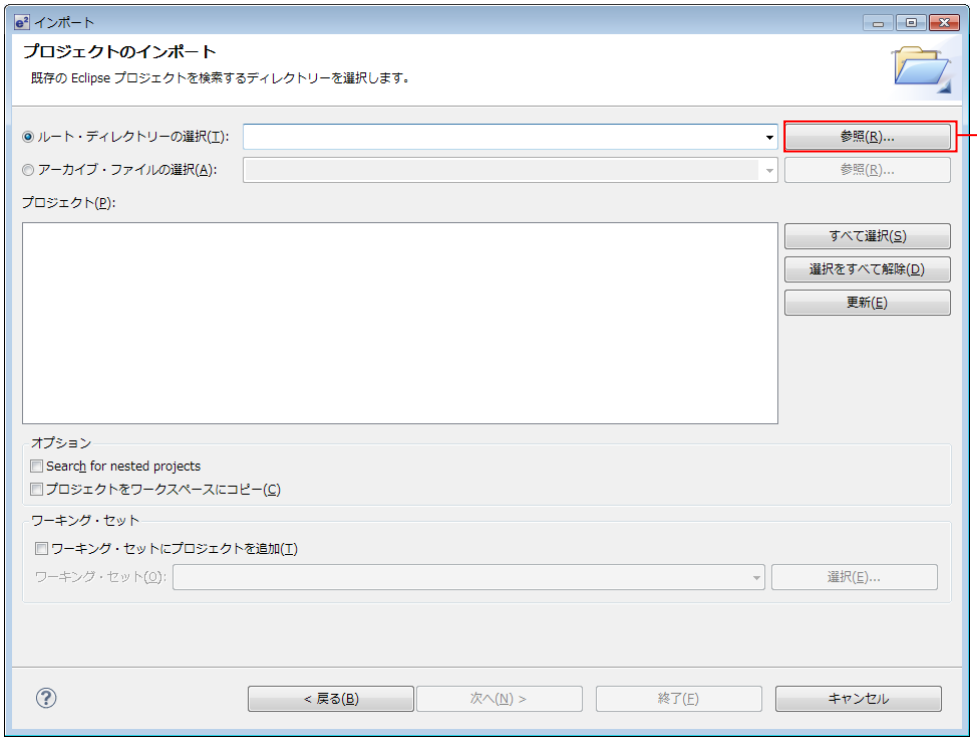
1. e² studio の「ファイル」メニューの「インポート」をクリックします。



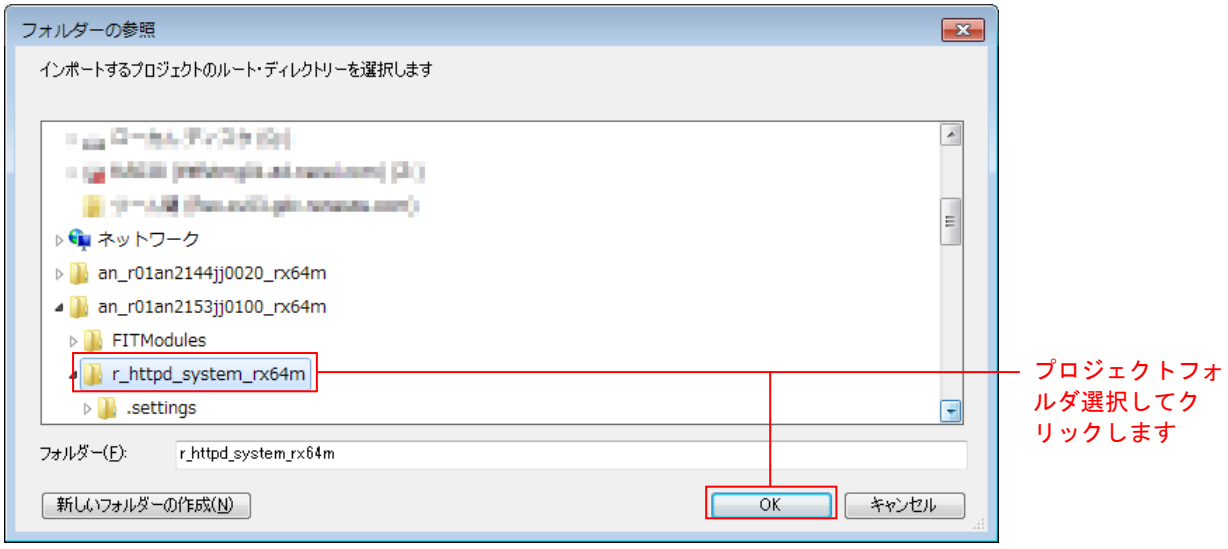
2. 「一般」から「既存プロジェクトをワークスペースへ」を選択し、「次へ」をクリックします。



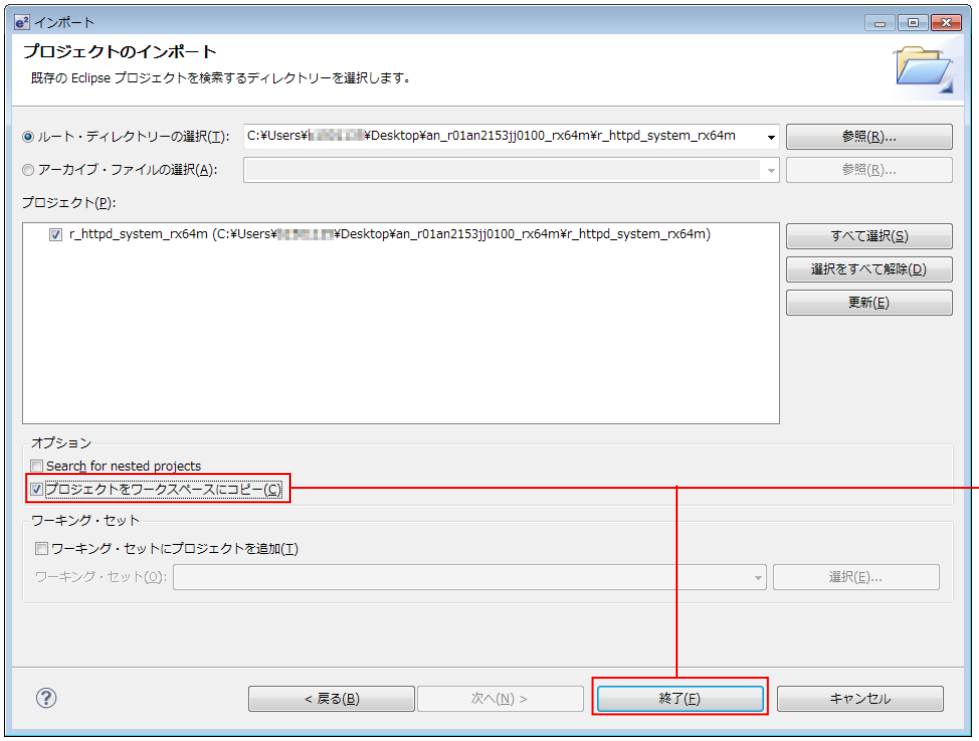
3. 「参照」をクリックします。



4. 本アプリケーションノートに付属しているプロジェクトフォルダ（r_httpd_system_rx64m）を選択し、「OK」をクリックします。



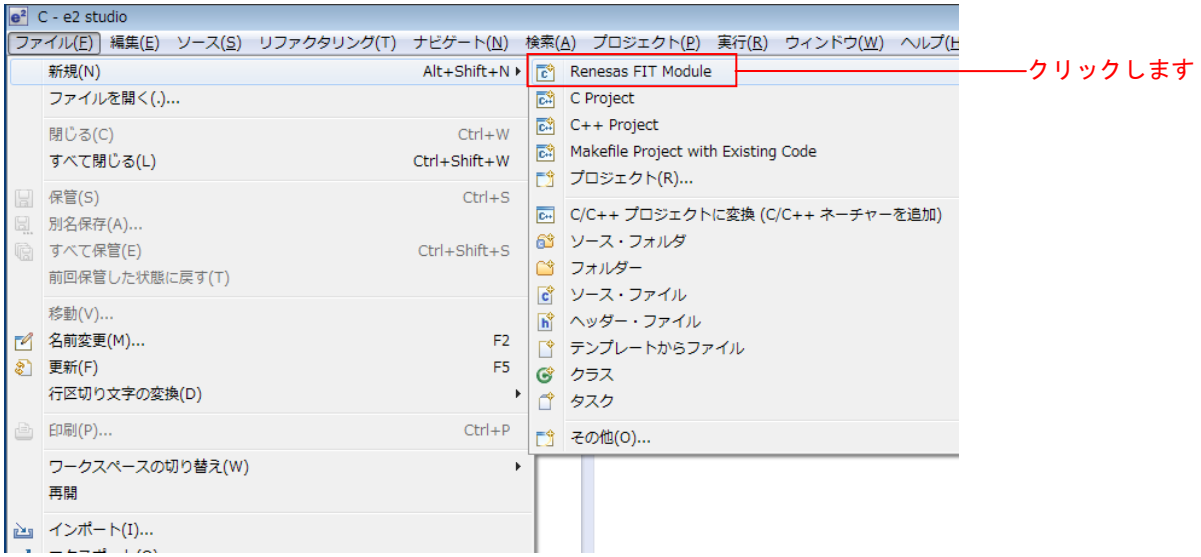
5. 「プロジェクトをワークスペースにコピー」にチェックを入れ、「終了」をクリックします。



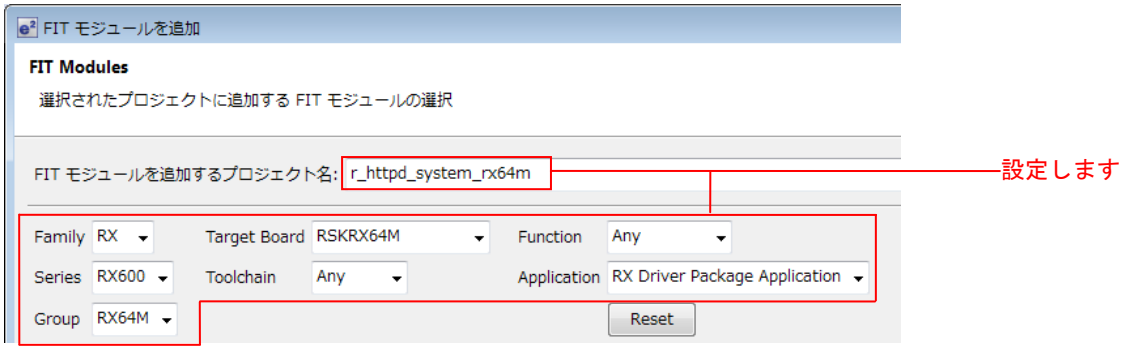
4.3　プロジェクトにWebサーバシステムのFITモジュールを追加する

e² studio の FIT プラグインを使用して、プロジェクトに Web サーバシステムで使用する FIT モジュールを追加します。

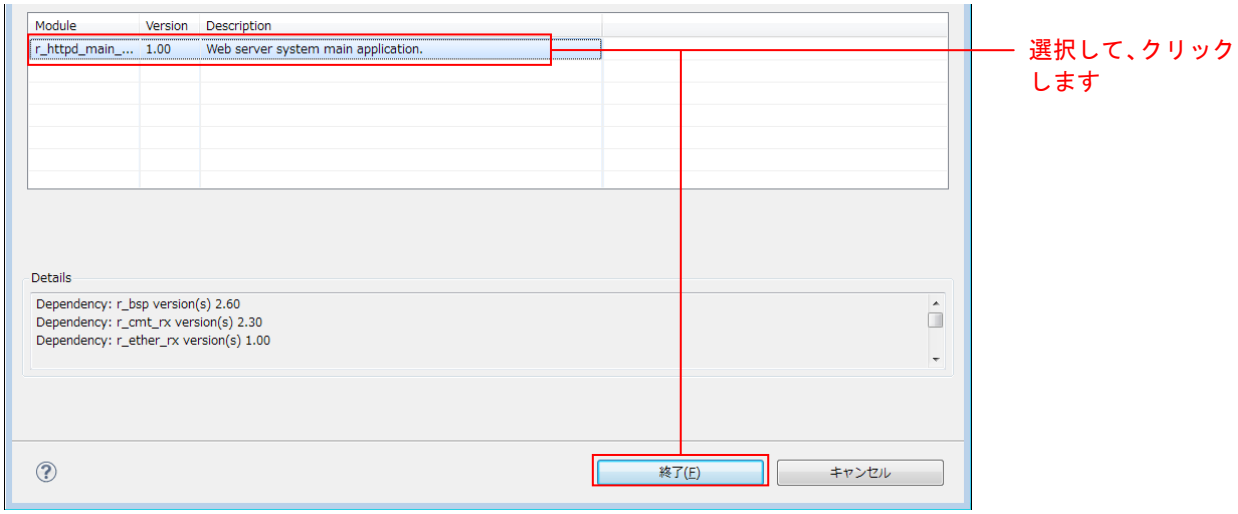
1. e² studio の「ファイル」メニューの「新規」の「Renesas FIT Module」をクリックして FIT プラグインを起動します。



2. FIT プラグインの各項目を以下の様に設定します。



3. FIT プラグインのモジュールリストから「r_httpd_main_rx64m」を選択し、「終了」をクリックします。



4. いくつかのメッセージダイアログ等が表示されますが、全て「OK」をクリックします。
以上により、プロジェクトに必要な全ての FIT モジュールがインストールされます。インストール後のプロジェクト構成は以下のようになります。

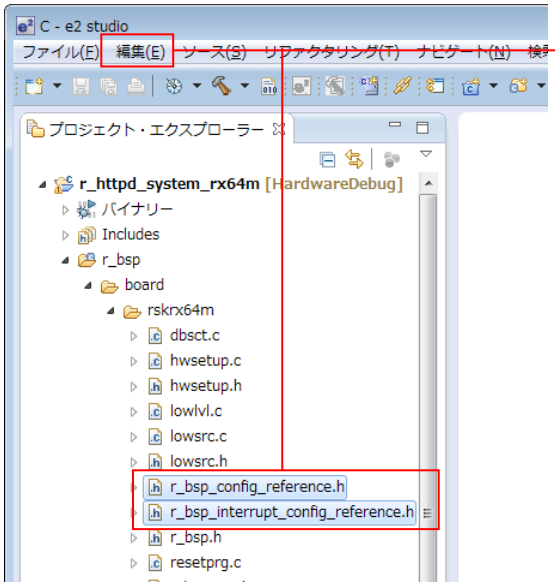
```
r_httpd_system_rx64m          // Webサーバシステム プロジェクトフォルダ
 ├ r_bsp                      // BSPモジュールフォルダ
 ├ r_cmt_rx                   // コンペアマッチタイマFITモジュールフォルダ
 ├ r_ether_rx                 // RX用Ethernetドライバ FITモジュールフォルダ
 ├ r_t4_driver_rx64m          // T4用イーサネットドライバインターフェース変換モジュールフォルダ
 ├ r_t4_http_server_rx        // HTTPサーバ FITモジュールフォルダ
 ├ r_t4_rx                    // T4 FITモジュールフォルダ
 ├ r_tfat_rx                  // TFAT FITモジュールフォルダ
 ├ r_usb_basic                // USB Basic Host and Peripheral firmware FITモジュールフォルダ
 ├ r_usb_hmsc                 // USB Host Mass Storage Class driver FITモジュールフォルダ
 ├ src                        // メインソースフォルダ
 ├ HardwareDebug              // ビルド構成フォルダ（デバッグ用）
 ├ doc                        // Webサーバシステム ドキュメントフォルダ
 └ r_config                   // FIT コンフィギュレーションファイルフォルダ
```

4.4　ボードサポートパッケージ（BSP モジュール）の設定

4.4.1　コンフィギュレーションファイルのコピー

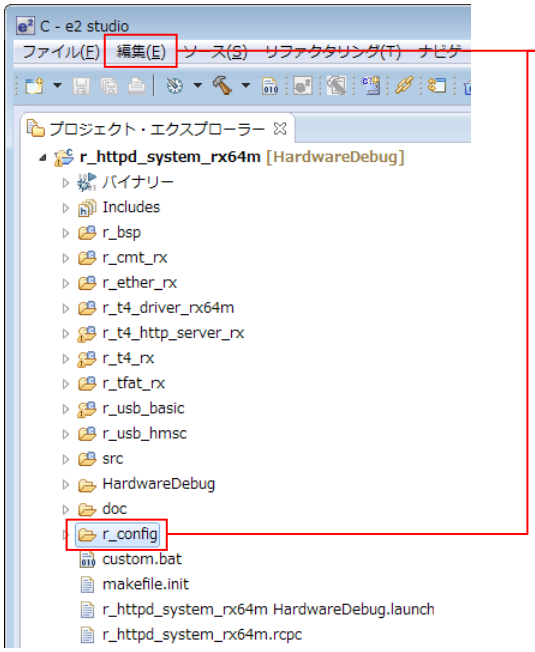
使用する MCU のコンフィギュレーションファイルを r_config フォルダにコピーします。

1. e² studio のプロジェクト・エクスプローラから「r_bsp/board/rskrx64m」を開き、「r_bsp_config_reference.h」と「r_bsp_interrupt_config_reference.h」の 2 つのファイルを選択し、「編集」メニューの「コピー」をクリックします。



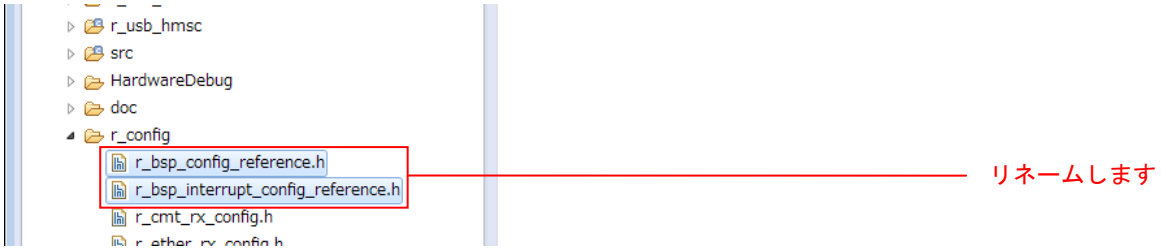
２つのファイルを選択し、「編集」メニューの「コピー」をクリックします

2. e² studio のプロジェクト・エクスプローラから「r_config」を選択し、「編集」メニューの「貼り付け」をクリックします。



「r_config」フォルダを選択し、「編集」メニューの「貼り付け」をクリックします

3. コピーしたファイルのファイル名を、それぞれ「r_bsp_config.h」と「r_bsp_interrupt_config.h」にリネームします（「_reference」を削除する）。



4.4.2　platform.h の修正

使用するターゲットボードに合わせて platform.h を修正します。

r_bsp/platform.h を開き、RSKRX64M 用の r_bsp.h のインクルード行のコメントを解除してください。

【r_bsp/platform.h】

```
/* RDKRX63N */
//#include "./board/rdkrx63n/r_bsp.h"

/* RDKRX631 */
//#include "./board/rdkrx631/r_bsp.h"

/* RSKRX64M */
#include "./board/rskrx64m/r_bsp.h"

/* RSKRX210 */
//#include "./board/rskrx210/r_bsp.h"

/* HSBRX21AP */
//#include "./board/hsbrx21ap/r_bsp.h"
```

4.5　コンフィギュレーションの変更

Web サーバシステムを構成する各 FIT モジュールのコンフィギュレーションファイルを変更します。

コンフィギュレーションファイルの項目と設定内容については、各 FIT モジュールの doc フォルダに入っているマニュアル等を参照してください。

以下に、Web サーバシステムを動作させるためのコンフィギュレーションファイルの変更箇所を示します。

4.5.1　割り込みスタックサイズの変更

Web サーバシステムでは、Ethernet コントローラの割り込みハンドラから、Web サーバの主な処理を行っており、約 2.5kbyte の割り込みスタックを必要とします。

r_bsp のコンフィギュレーションファイルで定義されている割り込みスタックサイズを、以下の様に変更してください。

【r_config/r_bsp_config.h】

```
/* Interrupt Stack size in bytes. The Renesas RX toolchain sets the stack size
using the #pragma stacksize directive.
 * If the interrupt stack is the only stack being used then the user will likely
want to increase the default size
 * below.
 */
#pragma stacksize si=0x1000
```

4.5.2　コンペアマッチタイマドライバの設定変更

コンペアマッチタイマの割り込みレベルを、USB ドライバの割り込みレベル（IPR=3）より低く設定します。

【r_config/r_cmt_rx_config.h】

```
/* The interrupt priority level to be used for CMT interrupts. */
#define CMT_RX_CFG_IPR        (2)
```

4.5.3　USB ドライバの設定変更

チャネル 0 を未使用（USB_NOUSE_PP）にしてください。

【r_config/r_usb_config.h】

```
//  #define USB_FUNCSEL_USBIP0_PP      USB_HOST_PP     /* Host Mode */
//  #define USB_FUNCSEL_USBIP0_PP      USB_PERI_PP     /* Peripheral Mode */
    #define USB_FUNCSEL_USBIP0_PP      USB_NOUSE_PP
```

4.5.4　　T4 の設定変更

T4 の設定を以下のように変更します。

t4_callback 関数の外部参照宣言をコメントアウトし、新たに http_callback 関数の外部参照宣言を追加してください。

【r_t4_rx/src/config_tcpudp.c】

```
#include "r_t4_itcpip.h"
//extern ER t4_callback(ID cepid, FN fncd , VP p_parblk);
extern ER http_callback(ID cepid, FN fncd , VP p_parblk);
```

TCP 受付口の設定を 6 個に増やし、各ローカルポートを変更してください。

【r_t4_rx/src/config_tcpudp.c】

```
/***  Definition of TCP reception point (only port number needs to be set) ***/
T_TCP_CREP tcp_crep[6] =
{
    /* { attribute of reception point, {local IP address, local port number}} */
    { 0x0000, { 0, 80 }},
    { 0x0000, { 0, 80 }},
    { 0x0000, { 0, 80 }},
    { 0x0000, { 0, 80 }},
    { 0x0000, { 0, 80 }},
    { 0x0000, { 0, 80 }},
};
```

TCP 通信端点の設定を、以下のように変更してください。

【r_t4_rx/src/config_tcpudp.c】

```
/***  Definition of TCP communication end point
     (only receive window size needs to be set) ***/
T_TCP_CCEP tcp_ccep[6] =
{
    /* { attribute of TCP communication end point,
         top address of transmit window buffer, size of transmit window buffer,
         top address of receive window buffer, size of receive window buffer,
         address of callback routine }
    */
    { 0, 0, 0, 0, 1460, http_callback },
    { 0, 0, 0, 0, 1460, http_callback },
    { 0, 0, 0, 0, 1460, http_callback },
    { 1, 0, 0, 0, 1460, http_callback },
    { 1, 0, 0, 0, 1460, http_callback },
    { 1, 0, 0, 0, 1460, http_callback },
```

2-MSL 待ち時間を、1 分から 10ms に変更してください。

【r_t4_rx/src/config_tcpudp.c】

```
/***  2MSL wait time (unit:10ms)  ***/
const UH    _tcp_2msl[] =
{
    (1),      /* 10 ms */
    (1),      /* 10 ms */
};
```

4.5.5　HTTP サーバの設定変更

CGI_FILE_NAME_TABLE_LIST を以下の様に設定します。

【r_config/r_t4_http_server_rx_config.h】

```
/*#define CGI_FILE_NAME_TABLE_LIST ¥*/
/*  {"cgi_smpl.cgi", NULL}, ¥*/
extern ER cgi_sample_function(ID cepid, void *res_info);
#define CGI_FILE_NAME_TABLE_LIST ¥
    {"cgi_smpl.cgi", cgi_sample_function, NULL}, ¥
```

HTTP サーバで使用する通信端点の数を、r_t4_rx/src/config_tcpudp.c の tcp_ccep テーブル数に合わせて 6 個に変更します。

【r_config/r_t4_http_server_rx_config.h】

```
// set same value number of CEPID in config_tcpudp.c
#define  HTTP_TCP_CEP_NUM  6
```

4.6　ソースコードの変更

以下に、Web サーバシステムを動作させるためのコードの変更箇所を示します。

4.6.1　多重割り込みの許可

本システムでは多重割り込みを使用しています。

t4_driver.c の timer_interrupt ハンドラ関数及び lan_inthdr ハンドラ関数内でコールしている、_process_tcpip 関数を呼びだす前に、割り込みを許可します。

【r_t4_driver_rx64m/src/t4_driver.c】

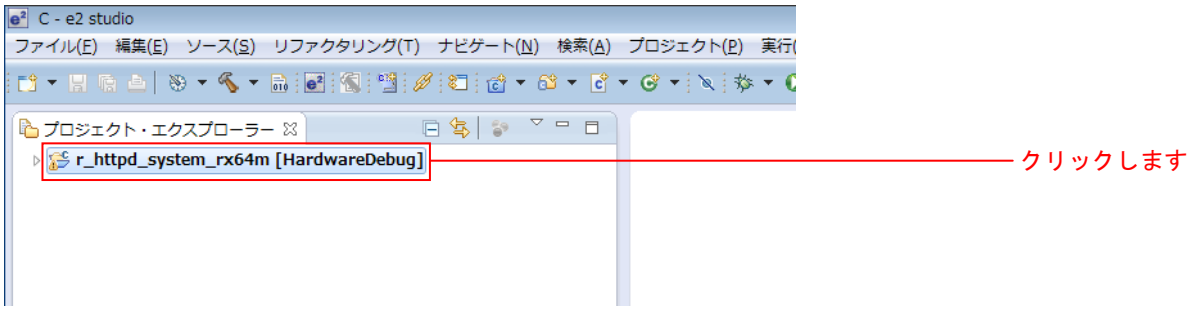
```
/***********************************************************************
Functions (Interrput handler)
***********************************************************************/

void timer_interrupt(void *pdata)
{
  R_BSP_InterruptsEnable();

  if (tcpip_flag == 1)
  {
    _process_tcpip();
    tcpudp_time_cnt++;
  }

  /* for wait function */
  if (wait_timer < 0xFFFF)
  {
    wait_timer++;
  }
}

void lan_inthdr(void *ppram)  // callback from r_ether.c
{
  R_BSP_InterruptsEnable();

  if (tcpip_flag == 1)
  {
    _process_tcpip();
  }
}
```

5.　動作確認

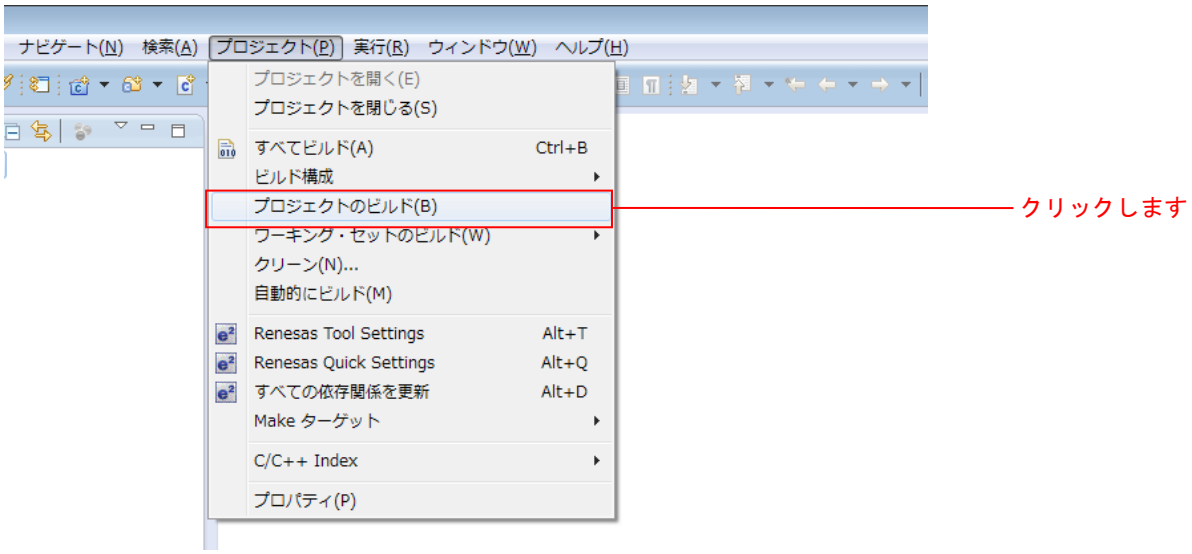
5.1　プロジェクトのビルド

以下の手順に従い、プロジェクトをビルドしてロードモジュールを生成します。

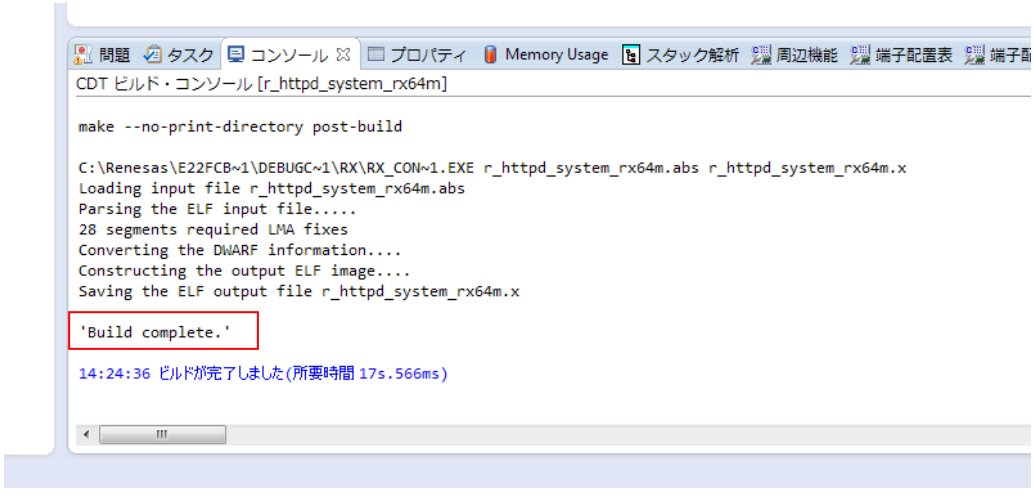
1.　「プロジェクト・エクスプローラ」からビルドするプロジェクトをクリックします。



2.　「プロジェクト」メニューの「プロジェクトのビルド」をクリックします。



3.　「コンソールパネル」に「'Build complete.'」と表示されたらビルド完了です。



5.2　デバッグの準備

5.2.1　機器の構成

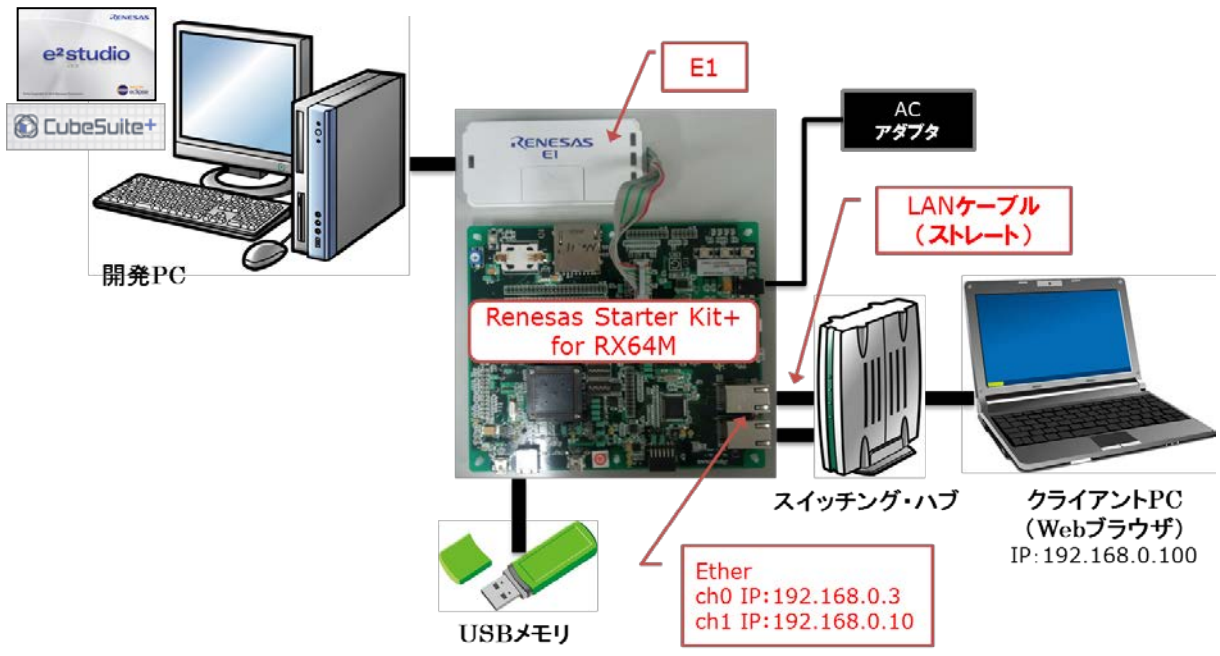
デバッグを開始する前に、評価ボードを準備します。

必要な機器の一覧と構成を以下に示します。

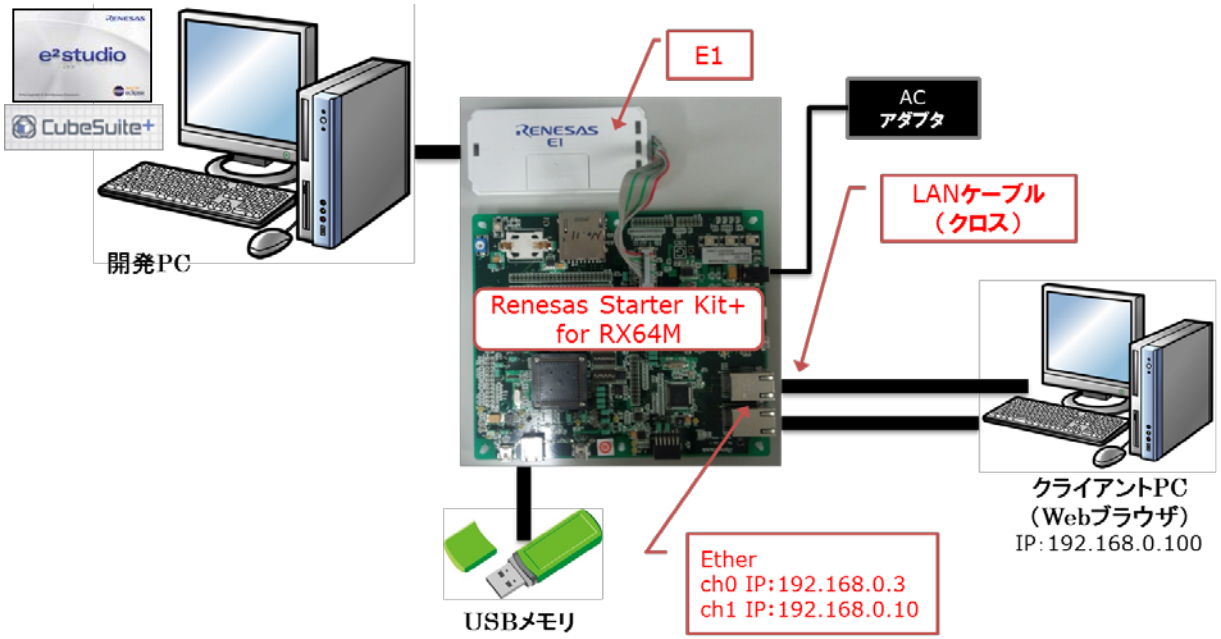
表5.2.1.1　機器構成

No.	機器	補足
1	開発 PC	開発を行う PC です。
2	評価ボード（Renesas Starter Kit+ for RX64M）	
3	USB メモリ	FAT、または FAT32 でフォーマットしたもの。
4	クライアント PC（Web ブラウザ）	開発 PC で代用可能です。
5	クライアント PC と RSK（Web サーバ）を接続するためのネットワーク環境として以下のいずれか 1. スイッチング・ハブを使用する場合 a. スイッチング・ハブ b. LAN ケーブル（ストレート）　3 本 2. クロスケーブルを使用する場合 a. LAN ケーブル（クロス）　2 本	クロスケーブルを使用し、Ethernet を 2ch 使用する場合は、クライアント PC に LAN ポートが 2 個必要です。 Ethernet を 1ch しか使用しない場合の LAN ケーブルの本数は、以下の様になります。 1. スイッチング・ハブを使用する場合 LAN ケーブル（ストレート）　2 本 2. クロスケーブルを使用する場合 LAN ケーブル（クロス）　1 本

スイッチング・ハブ構成（Ethernet 2ch 使用）



クロスケーブル構成（Ethernet 2ch 使用）



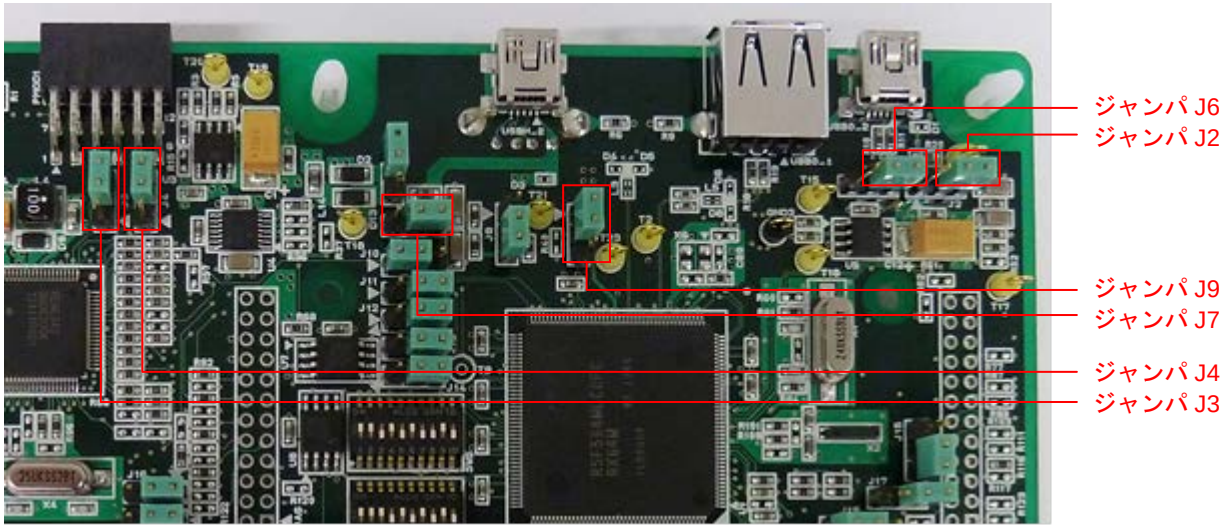
5.2.2　評価ボードの設定

Web サーバシステムを動作させるための、評価ボードの設定を以下に示します。

1. USB ch0 のモード（ホスト／ペリフェラル）を設定します。r_usb_config.h の「USB_FUNCSEL_USBIP0_PP」の設定に合わせて、ジャンパ J2 及び J6 を設定します。
2. USB ch1 のモード（ホスト／ペリフェラル）を設定します。r_usb_config.h の「USB_FUNCSEL_USBIP1_PP」の設定に合わせて、ジャンパ J7 及び J9 を設定します。
3. Ethernet コントローラから PHY IC を制御するために使用する PHY IC のチャネルを指定します。r_ether_rx_config.h の「ETHER_CFG_CH0_PHY_ACCESS」と「ETHER_CFG_CH1_PHY_ACCESS」の設定に合わせて、ジャンパ J3 及び J4 を設定します。

表5.2.2.1　ジャンパ設定一覧

No.	設定内容	ジャンパ	設定内容
1	USB0 をホストモードで使用する場合 （USB_FUNCSEL_USBIP0_PP = USB_HOST_PP）	J2	1-2 をショート
		J6	2-3 をショート
	USB0 をペリフェラルモードで使用する場合 （USB_FUNCSEL_USBIP0_PP = USB_PERI_PP）	J2	2-3 をショート
		J6	1-2 をショート
2	USB1 をホストモードで使用する場合 （USB_FUNCSEL_USBIP1_PP = USB_HOST_PP）	J7	1-2 をショート
		J9	2-3 をショート
	USB1 をペリフェラルモードで使用する場合 （USB_FUNCSEL_USBIP1_PP = USB_PERI_PP）	J7	2-3 をショート
		J9	1-2 をショート
3	PHY IC を ch1 で制御する	J3	2-3 をショート
		J4	2-3 をショート



【注】　画像と実際の設定は異なります。

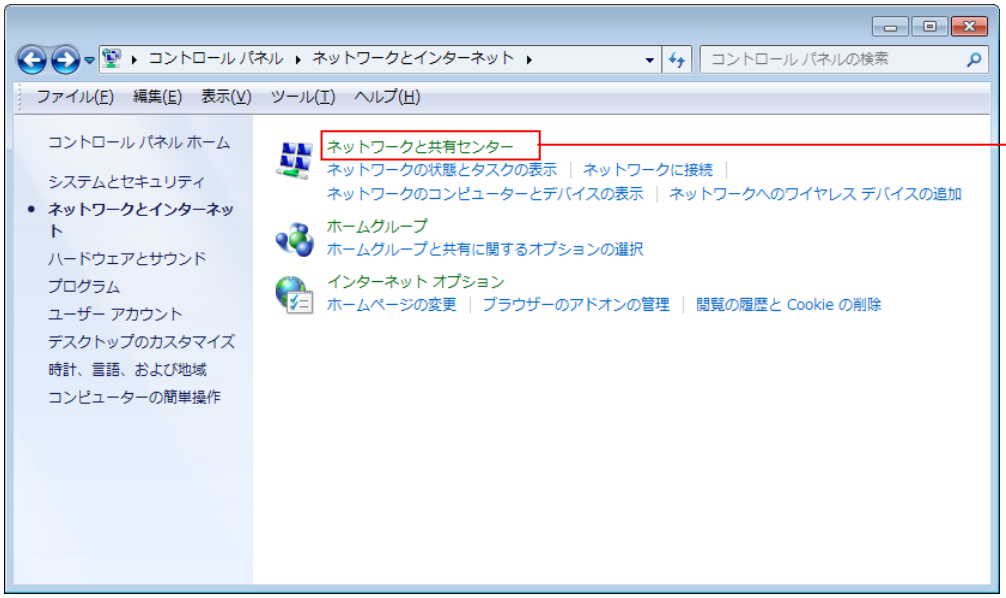
図5.2.2.1　Renesas Starter Kit+ for RX64M のジャンパ位置

5.2.3　クライアント PC の設定

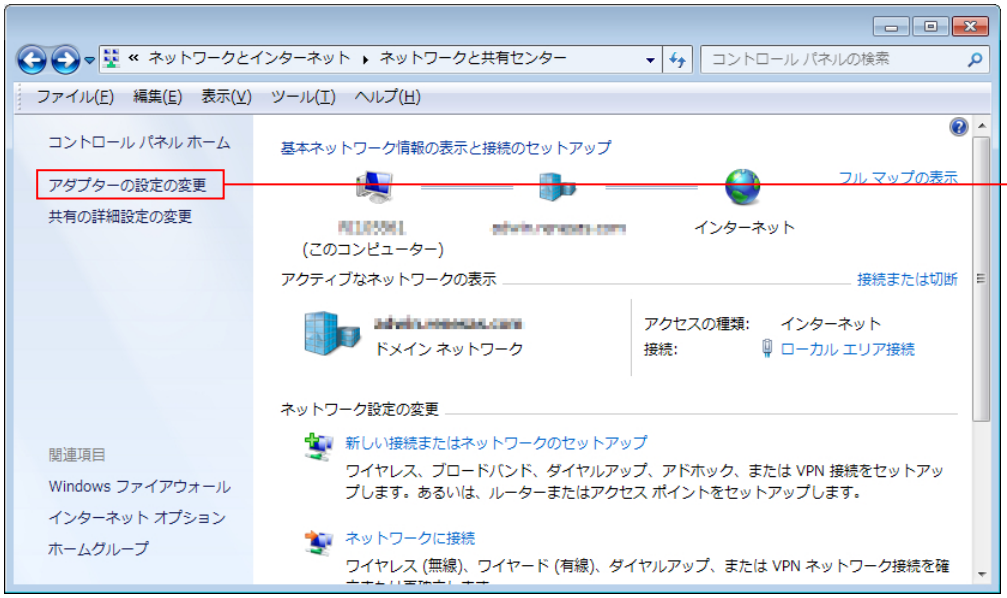
クライアント PC のネットワークを設定します。ここでは Windows7 の場合の例を示します。

1. クライアント PC の「コントロールパネル」を開き、「ネットワークとインターネット」をクリックします。

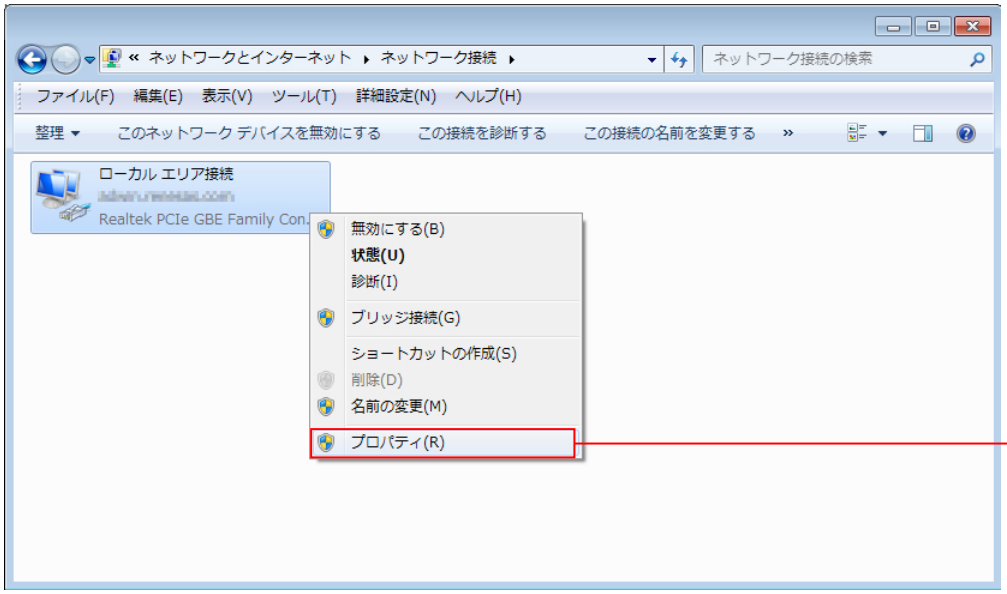
2.「ネットワークと共有センター」をクリックします。



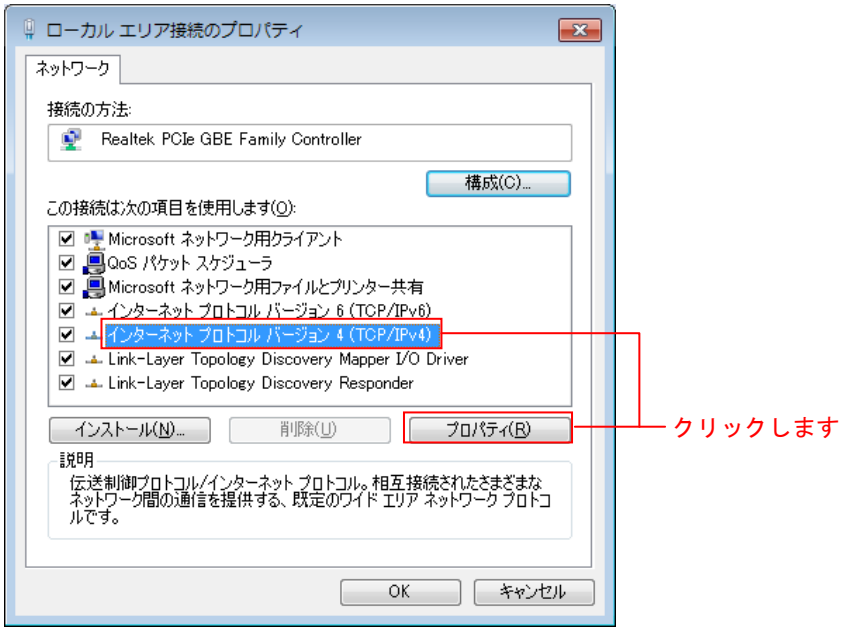
3. 「アダプターの設定の変更」をクリックします。



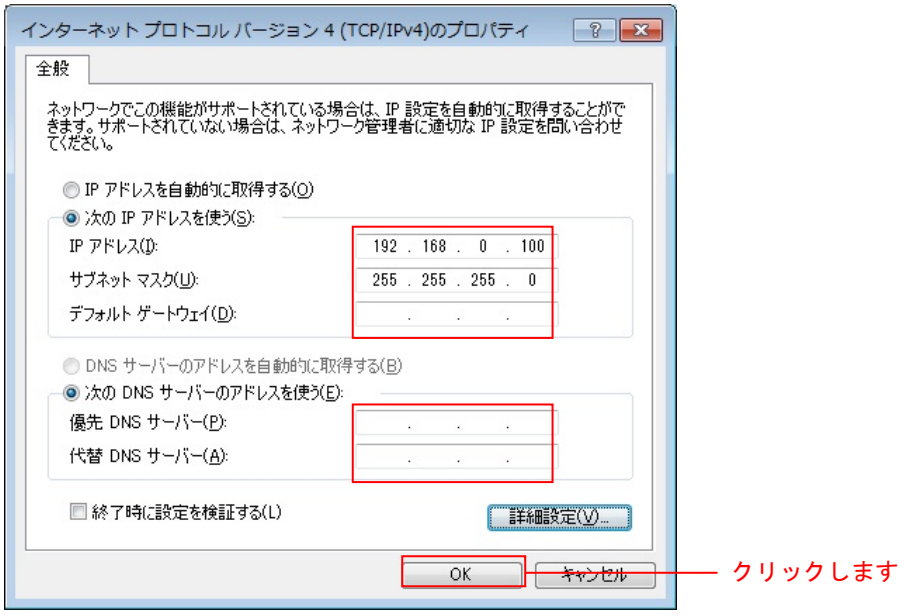
4. 「ローカル エリア接続」を右クリックして、プロパティをクリックします。



5. 「インターネットプロトコルバージョン 4 (TCP/IPv4)」を選択して「プロパティ」をクリックします。



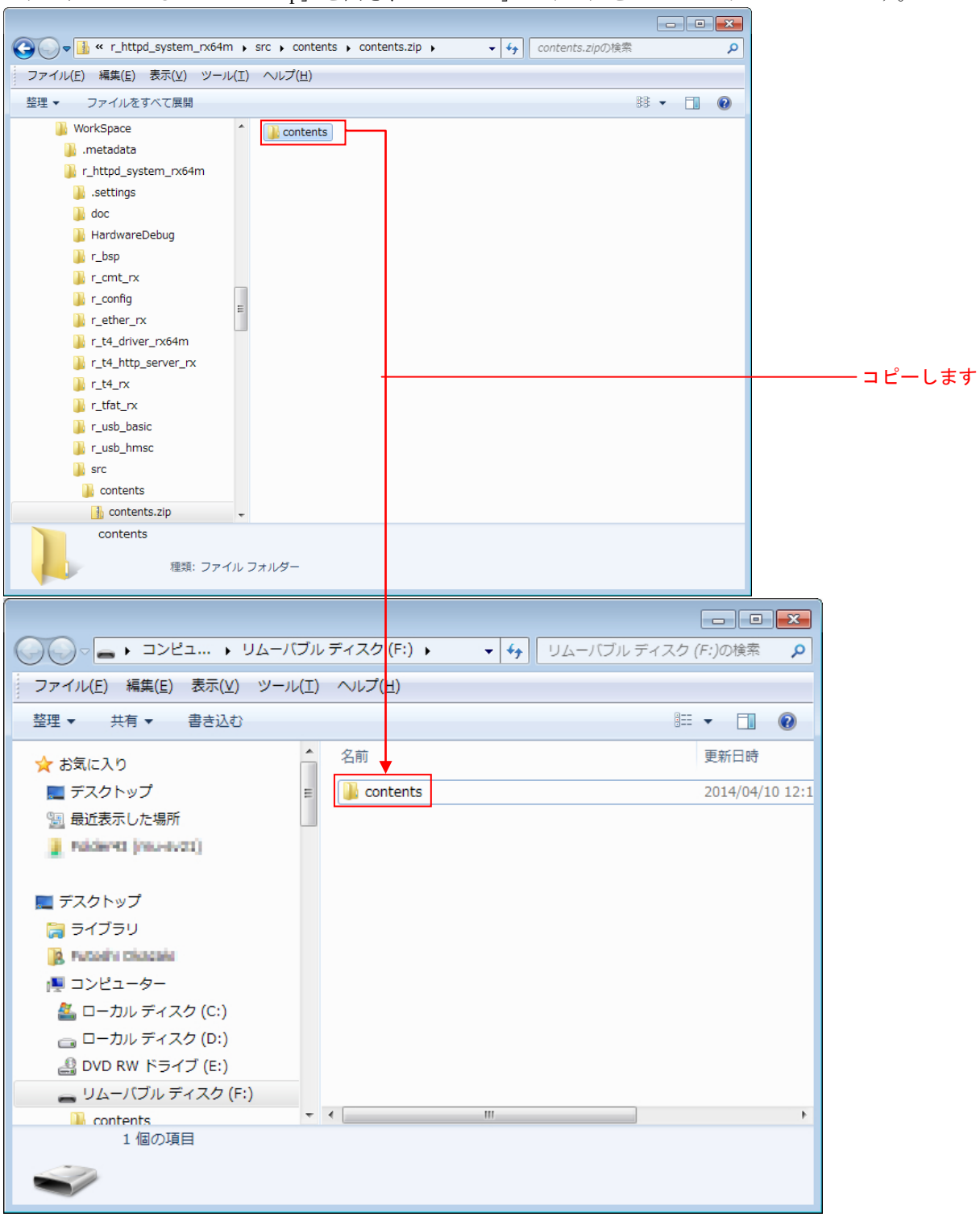
6. IP アドレス等の設定情報が表示されます。以下のように設定し「OK」をクリックします。



5.2.4　　USB メモリの準備

USB メモリに HTML コンテンツを格納しておきます。

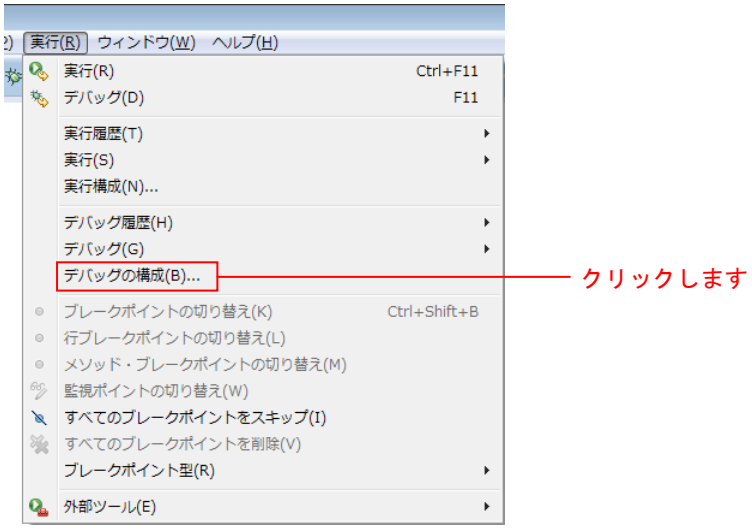
1. プロジェクト内の「src」フォルダを開き、その中にある「contents」フォルダを開きます。「demo」フォルダに入っている「contents.zip」を開き、「contents」フォルダを USB メモリにコピーします。



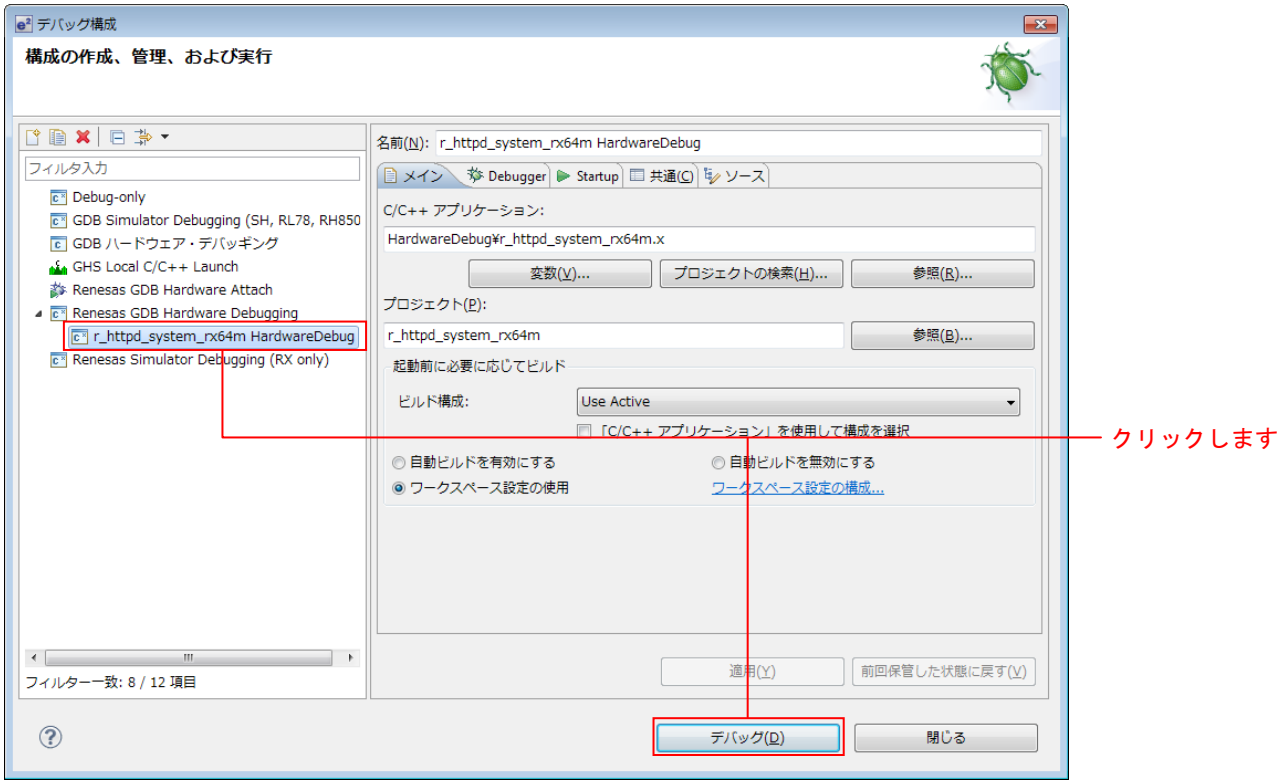
5.3　プロジェクトのデバッグ

以下の手順に従い、プロジェクトのデバッグを開始します。

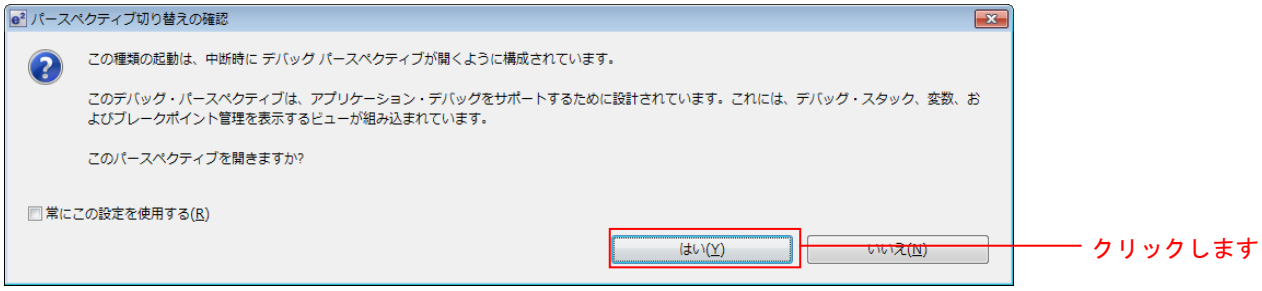
1. 開発 PC と E1 エミュレータを USB ケーブルで接続します。
2. 評価ボード（Renesas Starter Kit+ for RX64M）にアダプタを接続し、電源を入れます。
3. e² studio の「実行」メニューの「デバッグ構成」をクリックします。



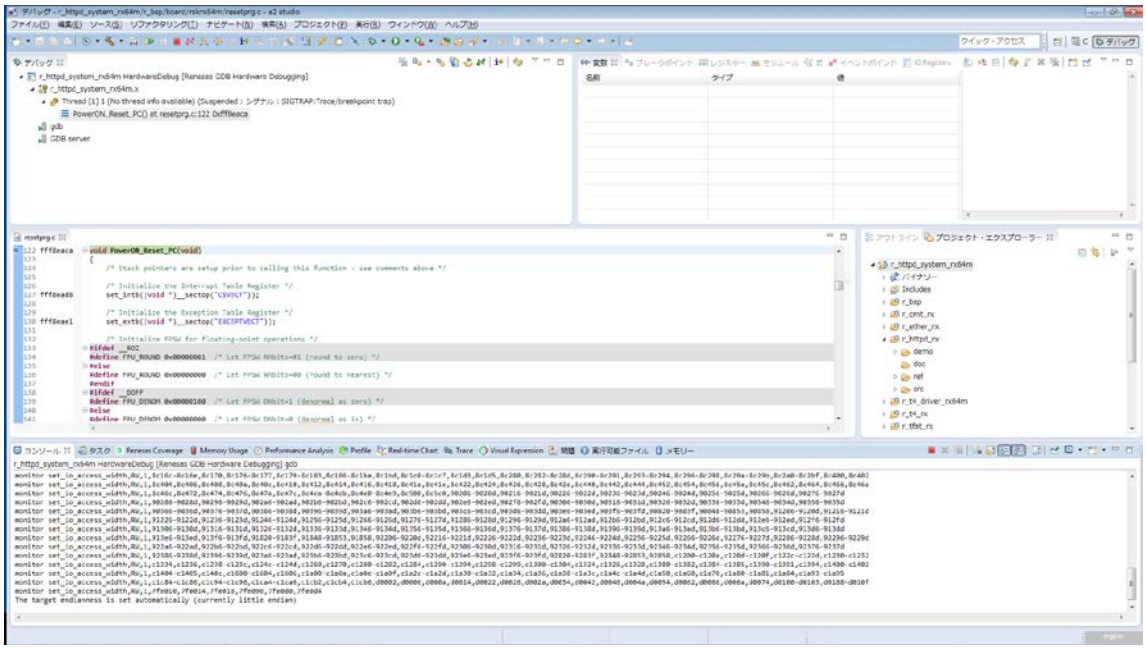
4. 「Renesas GDB Hardware Debugging」の「r_httpd_system_rx64m HardwareDebug」をクリックし、「デバッグ」をクリックします。



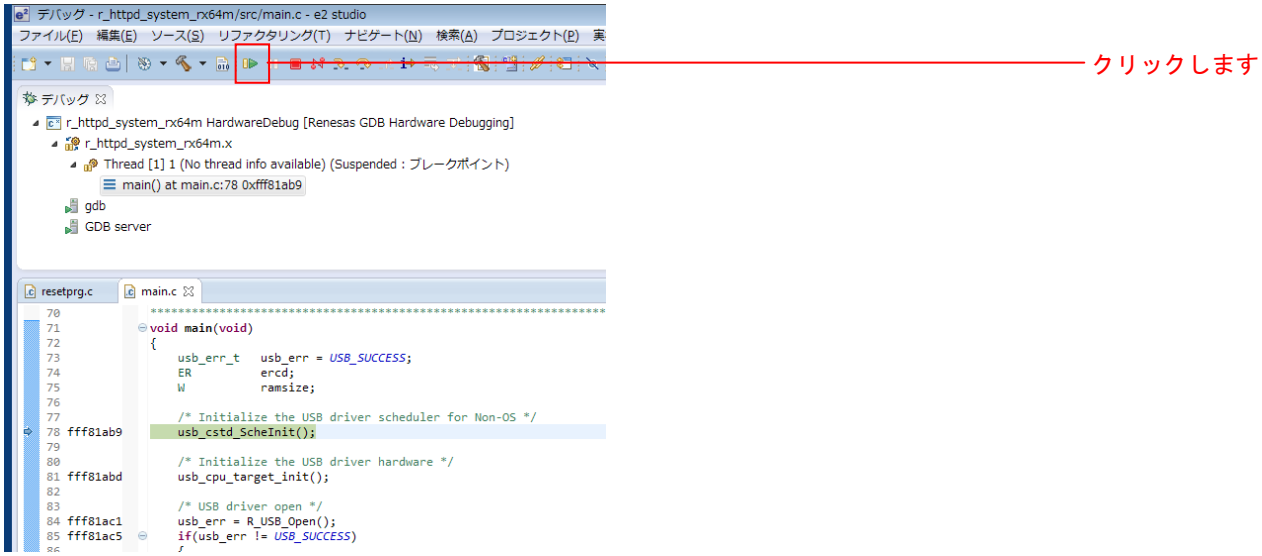
以下のメッセージが表示されたら、「はい」をクリックします。



ロードモジュールのダウンロードが完了すると、「デバッグ」パースペクティブが開きます。



5. ツールバーの「再開」をクリックします。プログラムが実行され、main 関数の先頭でブレークします。



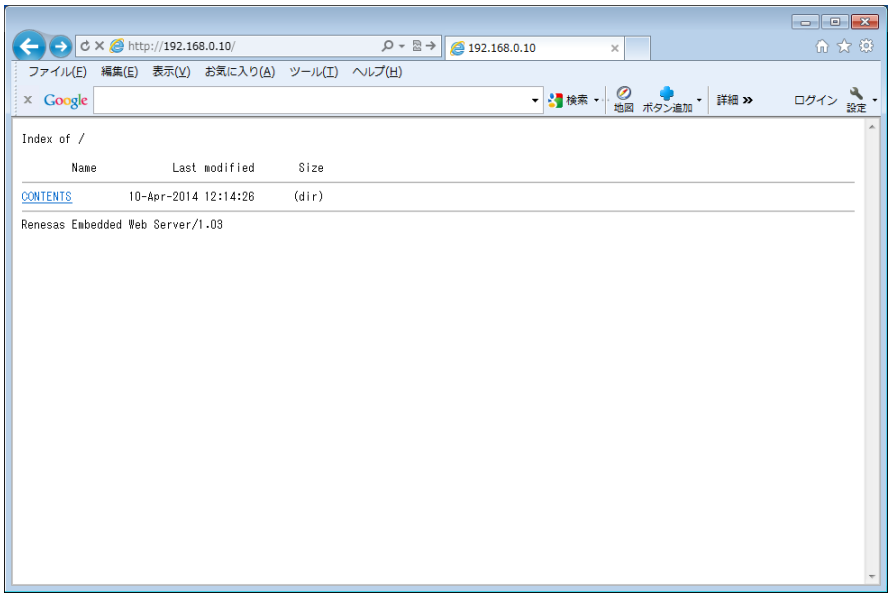
main 関数の先頭でブレークした後に、もう一度ツールバーの「再開」をクリックします。

6. クライアント PC で Web ブラウザを起動し、LAN ケーブルを接続したポートに合わせて、以下のアドレスを入力します。

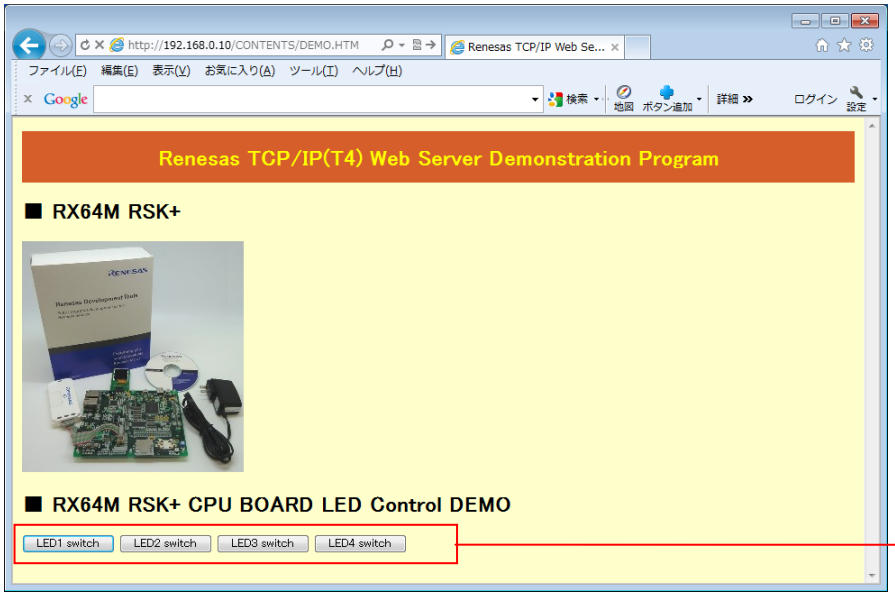
Ethernet ポート番号	Web サーバアドレス
0	http://192.168.0.3
1	http://192.168.0.10

【注】Web サーバアドレスは、コンフィギュレーションで変更可能です。

Web ブラウザ上に、USB メモリ内のルートディレクトリにあるファイルの一覧が表示されます。
Name はファイル名を表し、Last modified は最終更新日を表し、Size はディレクトリである場合には(dir)と表し、Byte でファイルサイズを表します。Parent Directory をクリックすれば一つ上のディレクトリに移動します。



7. 「CONTENTS」をクリックし、その中にある「DEMO.HTM」をクリックすると以下のような画面が表示されます。「LEDx switch」ボタンを押すと、ボード上の LED を制御(点灯/消灯)することが出来ます。



6.　Webサーバ仕様

6.1　性能概要

本 Web サーバは、簡易的な Web サーバを HTTP/1.0 の仕様に基づき実現しています。本 Web サーバは、ユーザが組み込み製品向けに独自の Web サーバプログラムを M3S-T4-Tiny (以下 T4 と称す)上で開発する際のベースとなることを目的としています。本 Web サーバは、SYN-FLOOD 攻撃等のアタックに対する対策等が盛り込まれておらず、またセキュリティ機能を搭載していないため、インターネットに接続して www ポート(80 番)を待ち受けてサーバ動作させるような用途には適しません。サンプルプログラムは事務所/工場内ネットワーク等の悪意のあるアタッカーが存在しないローカルのネットワークにおいて限定して使用することを想定しています。また扱えるファイル名はショートファイル名に限られます。

また、本 Web サーバは、ファイル I/O を除くと特別なメモリを必要とせずマイコン内蔵メモリのみで動作します。処理性能は RAM 容量に左右されますが、これを柔軟に設定出来るようにプログラム上で定義しています。本 Web サーバでは、RX64M の ROM/RAM 容量に合わせ適切なメモリ設定を行っています。

本 Web サーバの性能は以下の通りです。

表6.1.1　Web サーバ性能

項目	性能
ROM size	約 6.6 Kbyte
RAM size	約 36 Kbyte （約 5 Kbyte × 同時接続クライアント数 + α）
同時接続数	5 クライアント　(設定可能)
CGI 機能	Web ブラウザからマイコンをリモートコントロール出来る機能。

6.2　動作概要

本 Web サーバは、インターネットで広く使われている一般的な Web サーバ(Apache 等)と比べ、実装機能を最小限にとどめています。また組み込み用途で使いやすいようにノンブロッキングコールを用いて実装されており、アプリケーション側は定期的に R_httpd() 関数を呼び出すのみで Web サーバの処理が可能です。R_httpd() 関数では通信に使用する全通信端点(一般的にはソケットと呼ばれます)を監視し、切断状態になっていたら接続待ちに遷移させます。通信処理は T4 の API、_process_tcpip()で実行されていて、本 Web サーバではこの API をタイマ割り込みと Ethernet 割り込みの中で呼び出しています。_process_tcpip()は処理完了通知のためコールバック関数を呼び出します。HTTP のデータ解析処理、データ生成処理はこのコールバックルーチンの中にあります。_process_tcpip()の起動を含むこれらの割り込み処理は送受信ドライバの性能やコールバックルーチンの実装次第で処理時間が大幅に変動するため、必要に応じてこれら割り込み処理の優先度を下げたり、割り込み禁止にしたりしてアプリケーションの動作を優先させても構いません。

また、コンフィグファイル(r_t4_http_server_rx_config.h)のマクロ定義を変更することで Web サーバの挙動をカスタマイズすることができます。

6.3　CGI 機能

本 Web サーバは簡易的な CGI(Common Gateway Interface)機能を持っています。CGI とは Web ブラウザからの要求に従い、Web サーバ上でユーザプログラムを実行する仕掛けです。本 Web サーバでは CGI ファイルとして予め設定された URL が要求されると、対応する内部関数を呼び出します。

6.4　コンフィグレーション

コンフィグファイル(r_t4_http_server_rx_config.h)のマクロ定義を変更することで Web サーバの挙動をカスタマイズすることができます。

- Server ヘッダフィールド：HTTPD_VERSION_CODE
  Web ブラウザとの通信時に、Web ブラウザに送信する Server ヘッダフィールドに格納するデータを指定することが出来ます。

- ルートディレクトリ：ROOT_DIR
  外部メモリ上のどのディレクトリをルートディレクトリにするかを指定することが出来ます。
  例： #define ROOT_DIR “”
  #define ROOT_DIR “user”
  #define ROOT_DIR “user/root_dir”

- インデックスページの表示/非表示：INDEXES
  Web ブラウザからディレクトリ指定された場合の挙動を指定することが出来ます。
  1 を指定した場合、ディレクトリの内容をレスポンスします。
  0 を指定した場合、DEFAULT_FILE_NAME で指定されているファイルをレスポンスします

- インデックスページ非表示の場合にレスポンスするファイル：DEFAULT_FILE_NAME
  INDEXES に 0 を指定した場合にレスポンスするファイルです。
  このファイルが見つからない場合は 404 Not Found レスポンスを返します。

- 対応する Content-Type の数：MAX_EXTENSION
  外部メモリに格納するファイルの拡張子リストの定義数です。

- 対応する Content-Type：EXTENSION_TYPE_TABLE_LIST
  外部メモリに格納するファイルの拡張子リストです。
  ここに定義していない拡張子のファイルを転送する場合、リストの先頭に定義されている拡張子の設定でファイルをレスポンスします。

- 登録された CGI ファイルの個数：MAX_CGI_FILE

- CGI ファイル名と対応する内部関数のテーブル：CGI_FILE_NAME_TABLE_LIST

- インデックスページの生成に用いられる改行コード：LF_CODE

- 同時に受け付け可能な最大クライアント数：TCP_CEP_NUM
  T4 の config_tcpudp.c で定義されている通信端点の個数と合わせてください。

- インデックスページに表示可能な最大ファイル数：MAX_FILE_LIST
  BODY_BUF_SIZE を超えないように設定してください。

- 受信バッファサイズ：RCV_BUF_SIZE

- ヘッダフィールド用の送信バッファサイズ：HDR_BUF_SIZE

- ボディフィールド用の送信バッファサイズ：BODY_BUF_SIZE

6.5　ファイル一覧

Web サーバのファイル一覧を以下に示します。

表6.5.1　Web サーバファイル一覧

フォルダ名	ファイル名	内容
r_t4_http_server_rx/src	r_http_server.c	Web サーバソースファイル
	r_http_server_config.c	Web サーバコンフィギュレーションソースファイル
	r_http_server_config.h	Web サーバコンフィギュレーションヘッダファイル

6.6　API リファレンス

6.6.1　R_httpd

Description

アプリケーションは本関数を定期的に呼び出します。R_httpd()は、HTTP の通信に必要な通信端点を管理します。本関数は通信端点の管理のみを行い、通信自体は T4 が割り込み駆動により自動的に行います。

Usage

#include “r_t4_http_server_rx_if.h”

void R_httpd (void);

Parameters

無し

Return Value

無し

Remark

無し

6.6.2　R_httpd_pending_release_request

Description

アプリケーションは CGI 応答保留を解除する時に本関数を呼び出します。

使用方法は、6.9.1cgi_sample_functionを参照してください。

Usage

#include “r_t4_http_server_rx_if.h”

void R_httpd_pending_release_request(ID cepid);

Parameters

cepid　　入力　　通信端点 ID

Return Value

無し

Remark

無し

6.6.3　R_T4_HTTP_SERVER_GetVersion

Description

本関数は、現在インストールされているモジュールのバージョンを返します。バージョン番号はコード化されています。最初の 2 バイトがメジャーバージョン番号で、後の 2 バイトがマイナーバージョン番号です。例えば、バージョンが 4.25 の場合、戻り値は‘0x00040019’ となります。

Usage

```
#include “r_t4_http_server_rx_if.h”

uint32_t R_T4_HTTP_SERVER_GetVersion(void);
```

Parameters

無し

Return Value

Web サーバのバージョン

Remark

無し

6.7　ユーザ定義関数リファレンス　（ファイル関連）

本 Web サーバは本関数群を呼び出します。ユーザはファイルシステムに応じて適切に本関数の処理内容を定義します。また、Web サーバは本データ構造体を使用し、外部メモリの情報を取得することが出来ます。本 Web サーバでは TFAT を用いた例を定義しています。

表6.7.1　ファイル関連ユーザ定義関数一覧

関数名	機能概要	関数名	機能概要
change_dir()	作業ディレクトリの変更	file_write()	ファイルの書き込み
file_close()	ファイルのクローズ	get_file_info()	ファイル情報の取得
file_delete()	ファイルの削除	get_file_list_info()	ファイルリストの取得
file_open()	ファイルのオープン	get_file_size()	ファイルサイズの取得
file_read()	ファイルの読み込み	make_dir()	ディレクトリの作成
file_rename()	ファイル名の変更	remove_dir()	ディレクトリの削除
file_exist()	ファイルの有無を確認		

【注】上記関数群のうち本 Web サーバで使用しない関数はグレーアウト表記にしています。

6.7.1　　データ構造体

【日付情報構造体】

```
typedef struct date_info_
{
    uint16_t    year;                   // 2011, 2012, …
    uint8_t     month[4];               // Jan, Feb, Mar, …
    uint8_t     day;                    // 1-31
    uint8_t     day_of_the_week[4];     // Sun, Mon, Tus, …
    uint16_t    hour;                   // 0-23
    uint16_t    min;                    // 0-59
    uint16_t    sec;                    // 0-59
}DATE_INFO;
```

【ファイルリスト構造体】

```
typedef struct file_list_
{
    uint8_t     file_name[13];
    uint32_t    file_size;
    uint32_t    file_attr;
    DATE_INFO   date_info;
}FILE_LIST;
```

【マクロ定義】

```
#define FILE_WRITE          (0x10)
#define FILE_READ           (0x01)
#define FILE_ATTR_RDO       0x01    /* Read only */
#define FILE_ATTR_HID       0x02    /* Hidden */
#define FILE_ATTR_SYS       0x04    /* System */
#define FILE_ATTR_VOL       0x08    /* Volume label */
#define FILE_ATTR_DIR       0x10    /* Directory */
#define FILE_ATTR_ARC       0x20    /* Archive */
```

6.7.2 change_dir

Description

本関数は引数で指定されたディレクトリパスを作業ディレクトリに設定します。ディレクトリパスはフルパスで指定します。作業ディレクトリの情報は、通信端点毎に管理されます。

Usage

```
#include <stdint.h>
#include “r_file_driver.h”
int32_t change_dir(uint8_t *dir_path);
```

Parameters

dir_path	入力	指定されたディレクトリパスの格納先

Return Value

-1	ディレクトリが存在しない
0	ディレクトリが存在する

Remark

dir_path の終端には’/’が付く場合と付かない場合があります。使用するファイルシステムにあわせて’/’の有無を調整してください。

6.7.3 file_close

Description

本関数は引数で指定された ID 値に対応するファイルをクローズし、管理情報を破棄します。

Usage

```
#include <stdint.h>
#include “r_file_driver.h”
int32_t file_close(int32_t file_id);
```

Parameters

file_id	入力	クローズするファイルの ID 値

Return Value

-1	エラー
0	正常終了

Remark

無し

6.7.4 file_delete

Description

本関数は引数で指定されたファイルを削除します。ファイルの指定はルートディレクトリからのフルパスで指定します。

Usage

```
#include <stdint.h>
#include “r_file_driver.h”
int32_t file_delete(uint8_t *file_path);
```

Parameters

file_path	入力	ファイルのフルパスの格納先

Return Value

-1	エラー
0	正常終了

Remark

無し

6.7.5 file_open

Description

本関数は第 1 引数で指定されたファイルを第 2 引数で指定されたモードでオープンし、管理情報を独自で保存します。また、保存した管理情報を Web サーバが ID 参照できるように、戻り値として管理情報の ID 値を指定します。保存した管理情報はファイルのクローズ関数で ID 値が指定されるまで保持しなければなりません。

Usage

```
#include <stdint.h>
#include “r_file_driver.h”
int32_t file_open(uint8_t *file_path, uint8_t mode_flag);
```

Parameters

file_path	入力	ファイルのフルパスの格納先
mode_flag	入力	ファイルオープンのモード (FILE_WRITE または FILE_READ)

Return Value

-1	エラー
0 以上	オープンしたファイルの ID 値

Remark

ファイルオープンの状態はファイルのクローズ関数で対応する ID 値が指定されるまで保持しなければなりません。

6.7.6 file_read

Description

本関数は第 1 引数で指定された ID 値に対応するファイルデータを、第 2 引数が示すアドレスに、最大第 3 引数の値で示すサイズ分読み込まれます。第 1 引数の ID 値に対応する管理情報内のファイルポインタは読み込んだ分だけ更新され、ファイルのクローズ関数が呼び出されるまで保持します。

Usage

```
#include <stdint.h>
#include “r_file_driver.h”
int32_t file_read(int32_t file_id, uint8_t *buf, int32_t read_size);
```

Parameters

file_id	入力	読み込むファイルの ID 値
buf	出力	読み込んだファイルデータの格納先
read_size	入力	読み込むファイルサイズ

Return Value

-1	エラー
0 以上	読み込んだデータサイズ

Remark

無し

6.7.7 file_rename

Description

本関数は第1引数で指定されたファイルまたはディレクトリを第2引数で指定された名前に変更します。第 1 引数、第 2 引数ともにルートディレクトリからのフルパスで指定します。

Usage

```
#include <stdint.h>
#include “r_file_driver.h”
int32_t file_rename(uint8_t *old_name, uint8_t *new_name);
```

Parameters

old_name	入力	変更対象のファイルまたはディレクトリ
new_name	入力	変更後の名前

Return Value

-1	エラー
0	正常終了

Remark

無し

6.7.8 file_exist

Description

本関数は引数で指定されたファイルまたはディレクトリの有無を確認します。引数はルートディレクトリからのフルパスで指定します。

Usage

```
#include <stdint.h>
#include “r_file_driver.h”
int32_t file_exist(uint8_t *file_path);
```

Parameters

file_path	入力	有無を確認するファイルまたはディレクトリ

Return Value

-1	存在しない
0	存在する

Remark

無し

6.7.9 file_write

Description

本関数は第 1 引数で指定された ID 値に対応するファイルに対し、第 2 引数で指定されたアドレスから第 3 引数で指定されたサイズ分のデータを書き込みます。第 1 引数の ID 値に対応する管理情報内のファイルポインタは書き込んだ分だけ更新され、ファイルのクローズ関数が呼び出されるまで保持します。

Usage

```
#include <stdint.h>
#include “r_file_driver.h”
int32_t file_write(int32_t file_id, uint8_t *buf, int32_t write_size);
```

Parameters

file_id	入力	書き込むファイルの ID 値
buf	入力	書き込むデータの先頭アドレス
write_size	入力	書き込むサイズ

Return Value

-1	エラー
0	正常終了

Remark

無し

6.7.10　get_file_info

Description

本関数は第 1 引数で指定された ID 値に対応するファイルの管理情報を読み込み、ファイルの日付情報を第 2 引数で示す日付情報構造体に書き出します。

Usage

```
#include <stdint.h>
#include “r_file_driver.h”
int32_t get_file_info(int32_t file_id, DATE_INFO *date_info);
```

Parameters

file_id	入力	読み込むファイルの ID 値
date_info	出力	日付情報の格納先

Return Value

-1	エラー
0	正常終了

Remark

無し

6.7.11　get_file_list_info

Description

本関数は第 1 引数で指定されたディレクトリパスに格納されているファイルまたはディレクトリの情報を第 2 引数で指定されたファイルリスト構造体に書き出します。一度に書き出す最大情報個数は第 3 引数で指定し、第 4 引数でファイルリストの読み出し開始位置を指定します。

Usage

```
#include <stdint.h>
#include “r_file_driver.h”
int32_t get_file_list_info(uint8_t *dir_path, FILE_LIST *file_list, uint32_t num_file_list, int32_t read_index);
```

Parameters

dir_path	入力	読み出すディレクトリパスの格納先
file_list	出力	読み出したファイルリストの格納先 リストの最後にはファイル名格納領域の先頭に’¥0’を格納します。
num_file_list	入力	一度に読み出すファイルリスト情報の最大個数
read_index	入力	ファイルリストの読み出し開始位置

Return Value

-1	エラー
0 以上	読み出したファイルの個数

Remark

戻り値が num_file_list より小さい値を返した場合は、ファイルリスト情報の読み出しが終了したことを示し、num_file_list と同じ値を返した場合はファイルリスト情報に続きがあることを示します。本関数はファイルリストの続きを読み出す際に、read_index にファイルリストの読み出し開始位置を指定して呼び出します。

dir_path の終端には’/’が付く場合と付かない場合があります。使用するファイルシステムにあわせて’/’の有無を調整してください。

6.7.12　get_file_size

Description

本関数は引数で指定された ID 値に対応するファイルの管理情報を読み込み、ファイルサイズを返します。

Usage

```
#include <stdint.h>
#include “r_file_driver.h”
int32_t get_file_size(int32_t file_id);
```

Parameters

file_id	入力	読み込むファイルの ID 値

Return Value

-1	エラー
0 以上	ファイルサイズ

Remark

無し

6.7.13　make_dir

Description

本関数は引数で指定されたディレクトリを作成します。ディレクトリパスはフルパスで指定します。

Usage

```
#include <stdint.h>
#include “r_file_driver.h”
int32_t make_dir(uint8_t *dir_path);
```

Parameters

dir_path	入力	作成するディレクトリ名

Return Value

-1	エラー
0	正常終了

Remark

dir_path の終端には’/’が付く場合と付かない場合があります。使用するファイルシステムにあわせて’/’の有無を調整してください。

6.7.14　remove_dir

Description

本関数は引数で指定されたディレクトリを削除します。ディレクトリパスはフルパスで指定します。

Usage

```
#include <stdint.h>
#include “r_file_driver.h”
int32_t remove_dir(uint8_t *dir_path);
```

Parameters

dir_path	入力	削除するディレクトリ名

Return Value

-1	エラー
0	正常終了

Remark

dir_path の終端には’/’が付く場合と付かない場合があります。使用するファイルシステムにあわせて’/’の有無を調整してください。

6.8　ユーザ定義関数リファレンス（システムタイマ関連 ）

Web サーバは本関数群を呼び出します。ユーザはシステムタイマを定義します。

表6.8.1　システムタイマ関連ユーザ定義関数一覧

関数名	機能概要
get_sys_time()	システムタイマの先頭アドレス取得

6.8.1　データ構造体

【システムタイマ構造体】

```
typedef struct sys_time_
{
    uint32_t    sec;
    uint32_t    min;
    uint32_t    hour;
    uint32_t    day;
    uint32_t    month;
    uint32_t    year;
}SYS_TIME;
```

6.8.2　get_sys_time

Description

本関数はシステムタイマの先頭アドレスを取得します。

Usage

```
#include <stdint.h>
#include “r_t4_http_server_rx_config.h”
SYS_TIME *get_sys_time( void );
```

Parameters

無し

Return Value

システムタイマの先頭アドレス

Remark

システムタイマを管理する変数はユーザが定義してください。

6.9　サンプル CGI 関数

6.9.1　cgi_sample_function

Description

本関数は、”R_t4_http_server_config.h”の CGI_FILE_NAME_TABLE_LIST マクロで定義されている CGI 関数です。CGI_FILE_NAME_TABLE_LIST の第 2 要素(CGI 関数ポインタ) は、Web サーバで定義した cgi ファイルの URL を要求したときに呼び出されます。次に、HTTPd が CGI 関数を呼び出します。

HTTPd の動作は、戻り値によって変わります。

ケース：正常終了

CGI 処理が本関数で正常終了

ケース：内部エラー

CGI 処理エラーが本関数で発生

CGI process errors occur in this function.

ケース: CGI 応答保留

CGI 処理が本関数で終了しない（応答保留）、CGI_FILE_NAME_TABLE_LIST の第 3 要素（CGI 関数ポインタ）は、CGI 処理終了するとき、ユーザが R_httpd_pending_release_request()を呼び出したとき、呼び出されます。

Usage

```
#include "r_t4_itcpip.h"
#include "r_http_server_config.h"
#include "r_t4_http_server_rx_if.h"
ER cgi_sample_function(ID cepid, void *res_info);
```

Parameters

cepid	入力	CGI 関数実行の要求が有った通信端点 ID
res_info	入力	(HTTPD_RESOURCE_INFO*)res_info->param
		Web ブラウザから要求のあった URL に付属するパラメータ
	出力	(HTTPD_RESOURCE_INFO*)res_info->res.body
		応答として返す HTML 文字列
	出力	(HTTPD_RESOURCE_INFO*)res_info->res.body_size
		応答として返す HTML 文字列の長さ

Return Value

-1	内部エラー
-2	CGI 応答保留
0	正常終了

Remark

なし

7.　メインプログラム仕様

7.1　　ファイル一覧

メインプログラムのファイル一覧を以下に示します。

表7.1.1　メインプログラムファイル一覧

フォルダ名	ファイル名	内容
src	main.c	メインソースファイル
	led.c	LED 初期化処理ソースファイル
	led.h	LED 初期化処理ヘッダファイル
	r_file_driver.c	Web サーバ用ファイルシステムインタエースソースファイル
	r_file_driver.h	Web サーバ用ファイルシステムインタエースヘッダファイル
	r_http_server_cgi_sample.c	CGI サンプルソースファイル
	r_sys_time.c	Web サーバ用システムタイマソースファイル
	r_sys_time.h	Web サーバ用システムタイマヘッダファイル
	r_usb_hmsc_api.c	USB ドライバ呼び出し処理ソースファイル
	r_usb_hmsc_api.h	USB ドライバ呼び出し処理ヘッダファイル

7.2　モジュール一覧

メインプログラムのモジュール一覧を以下に示します。

表7.2.1　メインプログラムモジュール一覧

ファイル名	モジュール名	内容
main.c	main	メインプログラムのメイン処理。 各 FIT モジュールの初期化処理を呼び出し、Web サーバと USB ドライバと Ethernet ドライバのメイン処理を駆動する（無限ループによる周期起動）。
r_usb_hmsc_api.c	usb_cstd_IdleTaskStart	Low Power Mode で使用するアイドルタスクを起動する。
	usb_cstd_IdleTask	Low Power Mode で使用するアイドルタスク ホスト動作では処理なし。
	usb_hmsc_task_start	HMSC ドライバ起動処理 USB IP の初期化やクラスドライバの登録を行う。
	usb_apl_task_switch	非 OS 環境用の各 USB ドライバタスクのスケジューリング処理
	usb_hapl_task_start	HMSC ドライバアプリケーションタスクを起動する。
	usb_hmsc_DummyFunction	HMSC ドライバ用ダミー関数
	usb_hmsc_DriveOpen	HMSC ドライバオープン処理。
	usb_hapl_registration	HMSC ドライバを登録する。
	usb_hmsc_apl_init	HMSC ドライバアプリケーションタスクの内部変数の初期化を行う。
	usb_hmsc_StrgCommandResult	R_usb_hmsc_StrgDriveSearch()のコールバック処理
	usb_hmsc_SampleAplTask	HMSC ドライバアプリケーションタスク処理 USB メモリを検出し、ファイルシステムをマウントする。
led.c	led_init	LED 初期化
r_file_driver.c	ー	「6.7ユーザ定義関数リファレンス （ファイル関連）」参照
r_http_server_cgi_sample.c	ー	「6.9サンプル CGI 関数」参照
r_sys_time.c	ー	「6.8ユーザ定義関数リファレンス（システムタイマ関連 ）」参照

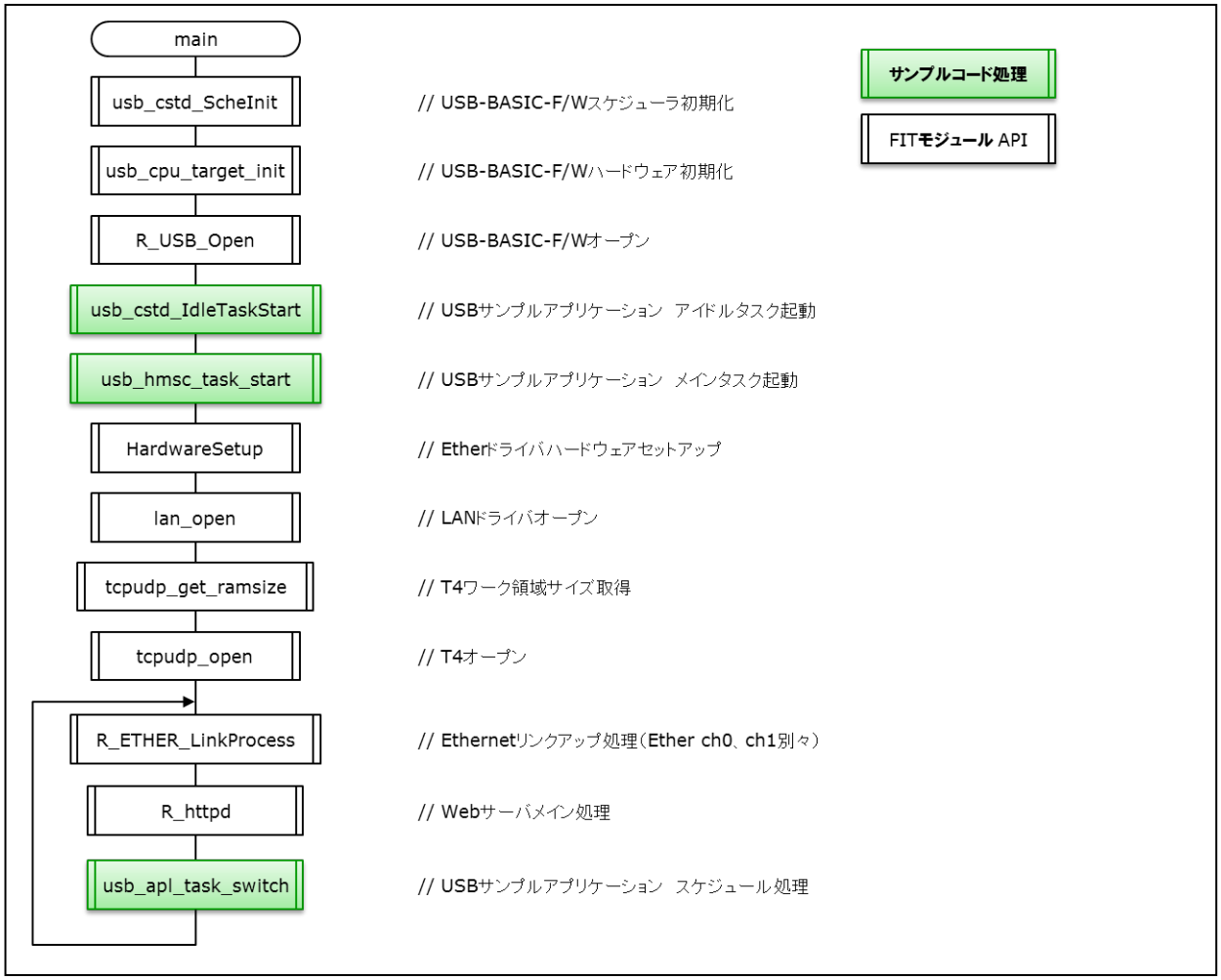
7.3　処理フロー

メインプログラムの各モジュールの処理フローを以下に示します。

1.　main()

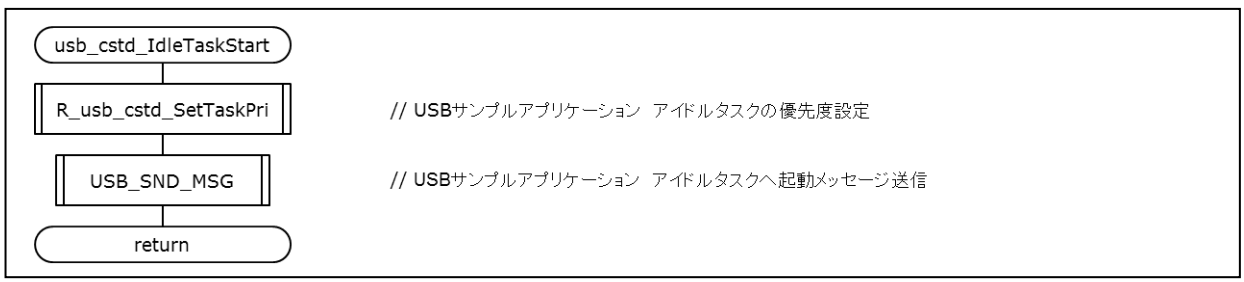
ボードサポートパッケージ（BSP モジュール）のスタートアップルーチンから最初に呼ばれるメイン関数です。

各ドライバと T4 の初期化を行い、無限ループ処理内で Ethernet ドライバのリンクアップと、Web サーバのメイン処理と、USB ドライバのスケジュール処理を定期的に呼び出します。



2.　usb_cstd_IdleTaskStart

USB ドライバ処理のアイドルタスクを起動します。

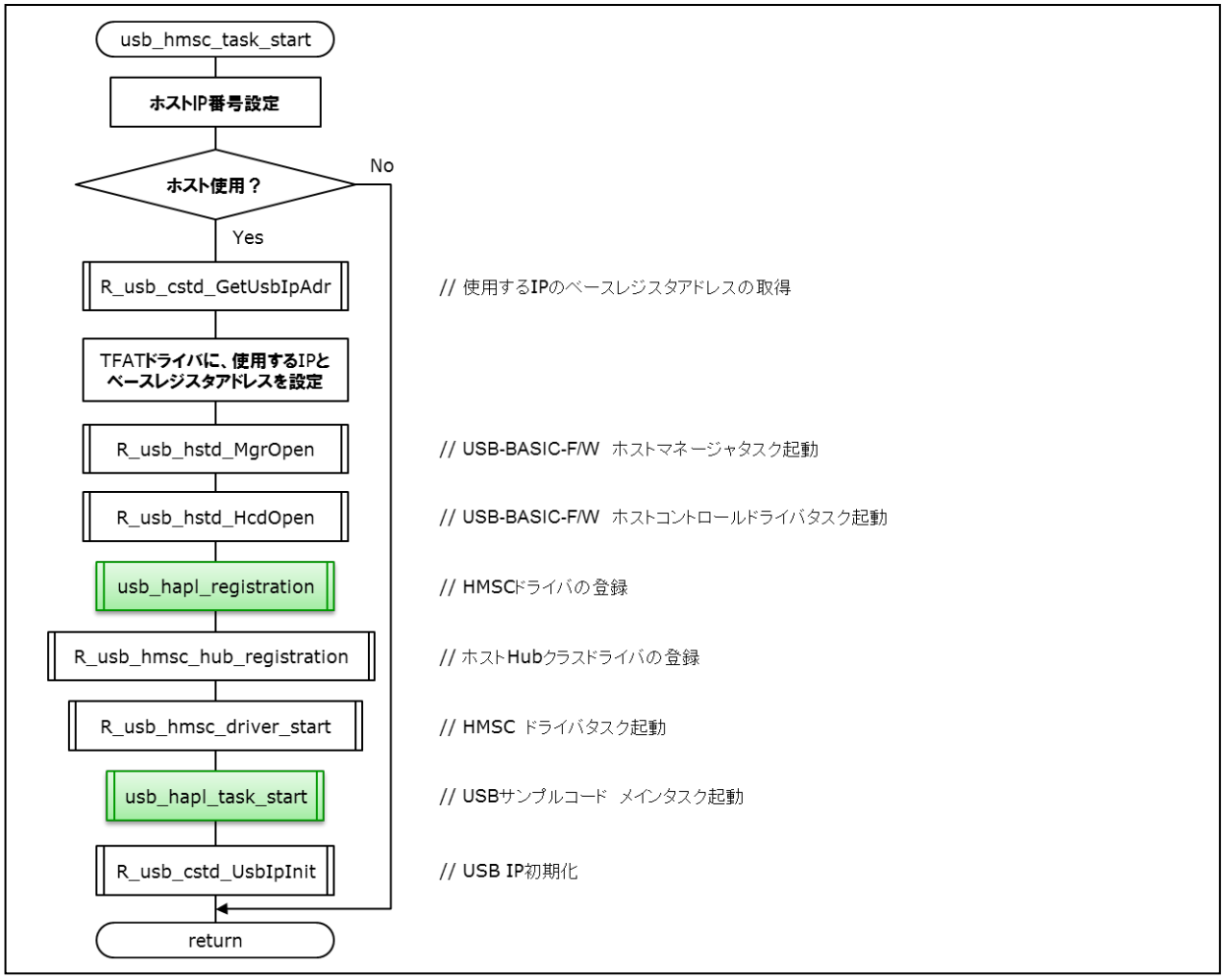


3. usb_cstd_IdleTask

USB ドライバ処理のアイドルタスクです。

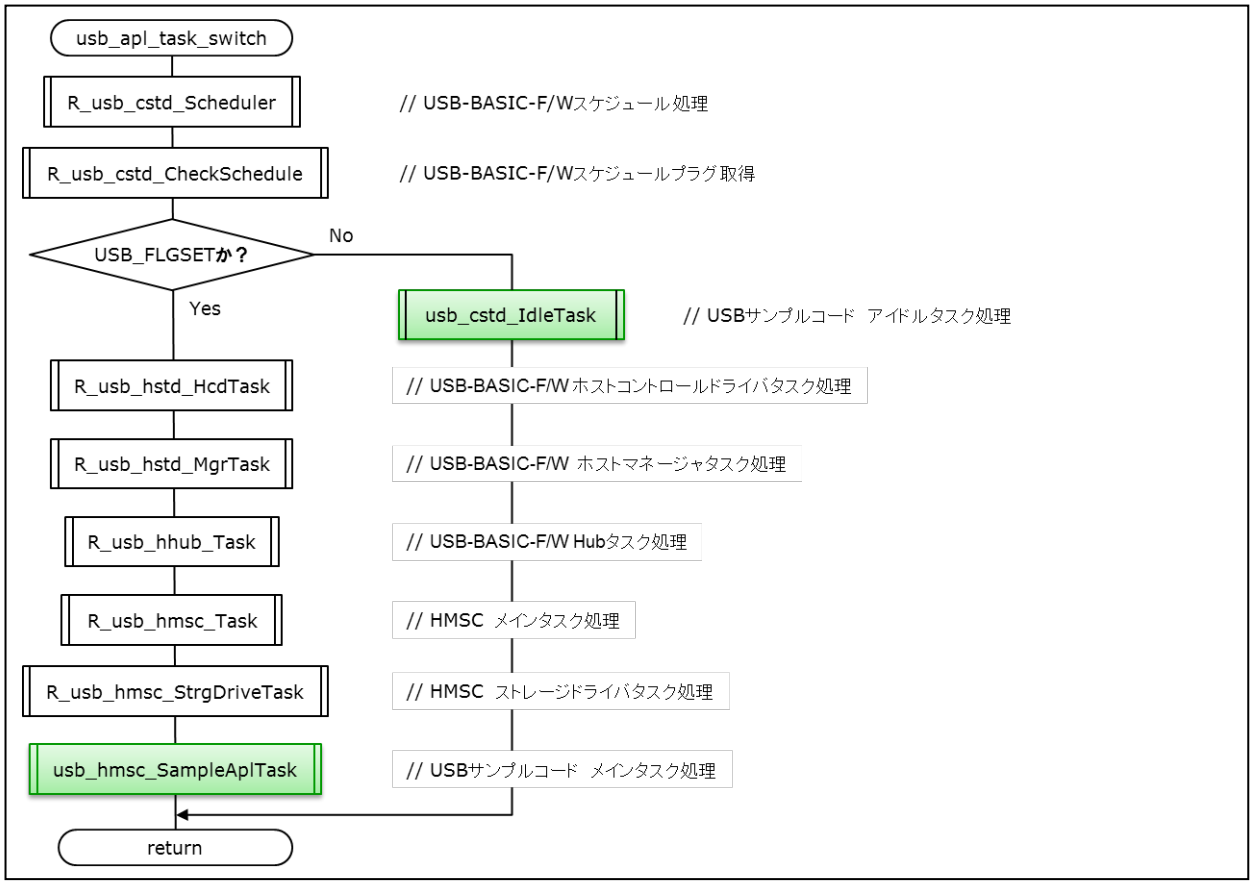
4. usb_hmsc_task_start

USB ドライバ内の各タスクの起動と、クラスドライバの登録を行い、USB メモリのマウント処理用タスクを起動します。



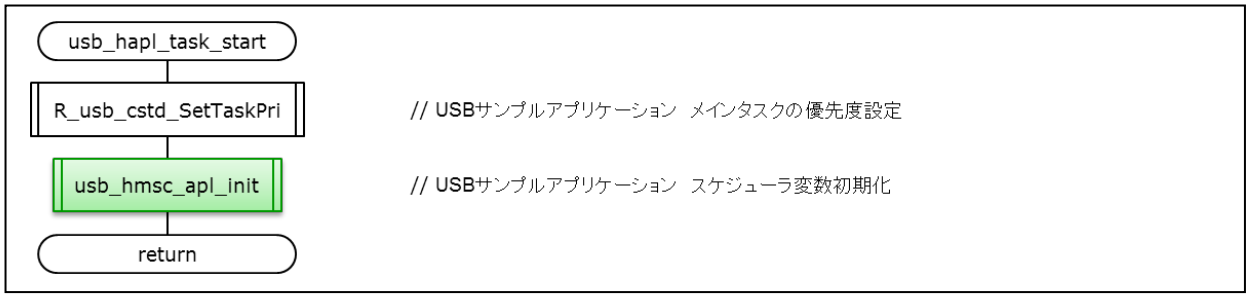
5. usb_apl_task_switch

USB ドライバのスケジュール処理です。



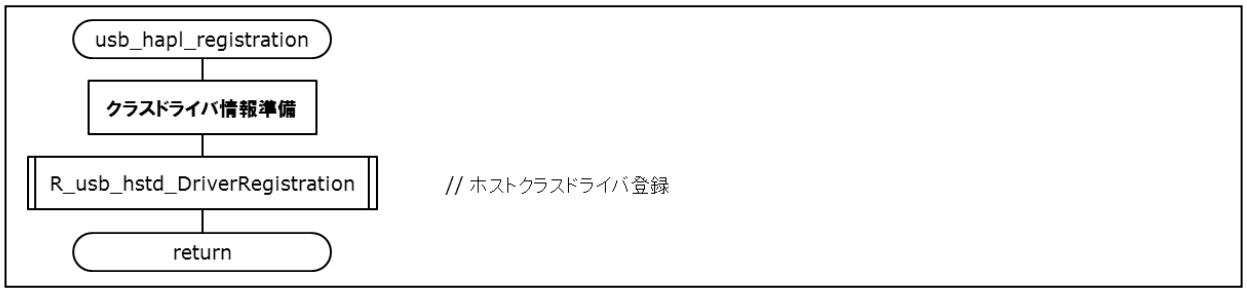
6. usb_hapl_task_start

USB メモリのマウント処理用タスクの初期化を行います。



7. usb_hapl_registration

クラスドライバの登録処理です。



8. usb_hmsc_apl_init

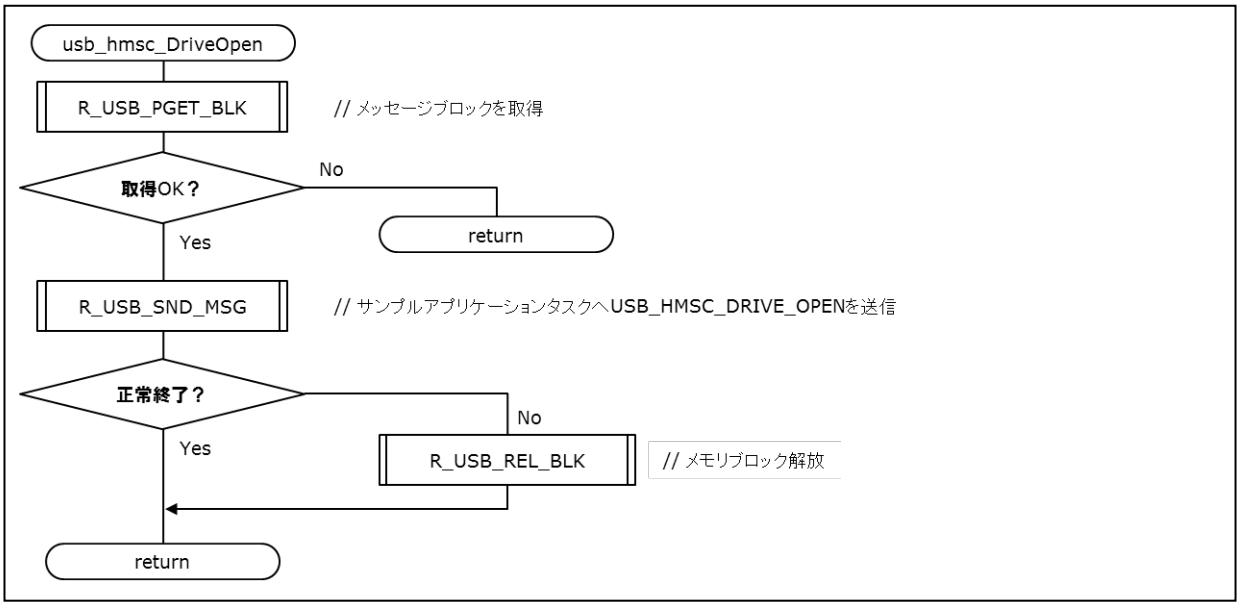
USB メモリのマウント処理用タスクのシーケンス処理変数を初期化します。

9. usb_hmsc_DummyFunction

クラスドライバ登録時に指定する suspend と resume 用のダミー関数です。

10. usb_hmsc_DriveOpen

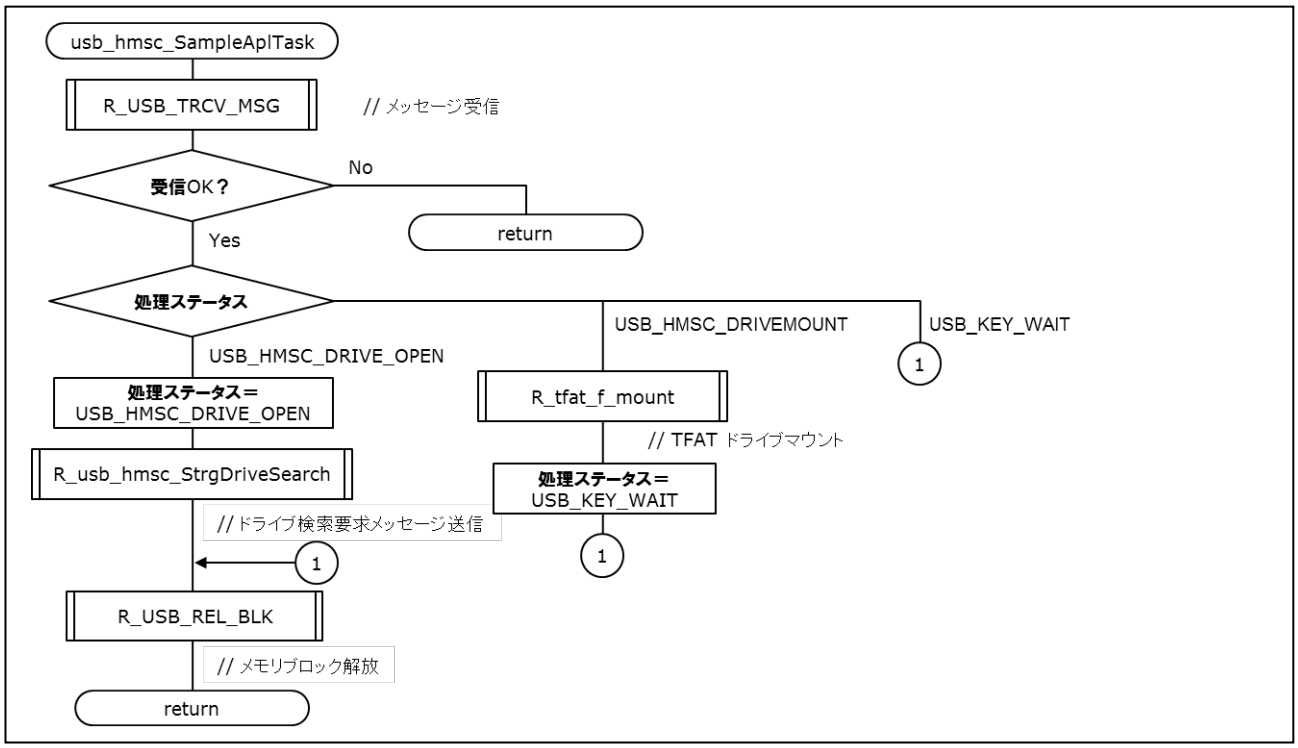
USB メモリが挿入されたときに、USB ドライバから呼び出されるコールバック関数です。サンプルアプリケーションタスクに対し、USB_HMSC_DRIVE_OPEN メッセージを送信します。



11. usb_hmsc_SampleAplTask

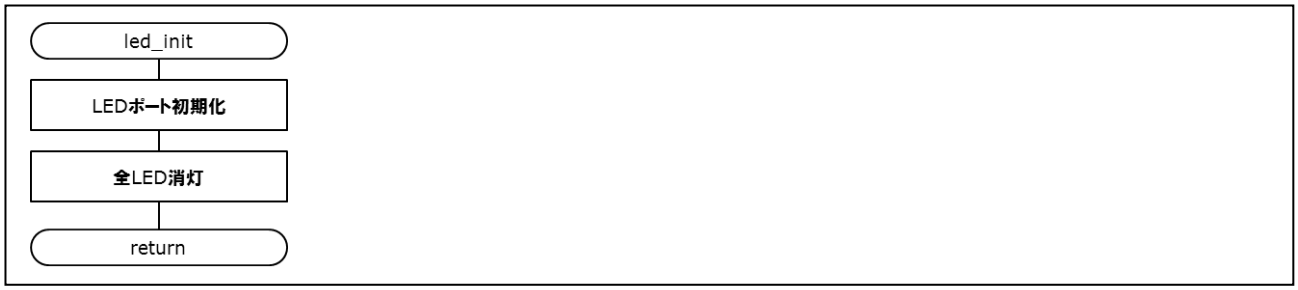
サンプルアプリケーションタスク処理です。usb_hmsc_DriveOpen 関数から送信された USB_HMSC_DRIVE_OPEN メッセージを受信し、マウント可能なドライブの検索を行います。

また、usb_hmsc_StrgCommandResult 関数から送信された USB_HMSC_DRIVEMOUNT メッセージを受信し、ファイルシステムへのマウントを行います。



12. led_init

Renesas Starter Kit+ for RX64M 上の LED を使用するための初期化処理を行います。



8.　ユーザ定義関数について

ユーザ定義関数は、ユーザのシステム環境に合わせて、ユーザが記述する必要があるコードです。FIT モジュールの中には、ユーザ定義関数を必要とするものがあります。

本パッケージでは、以下のユーザ定義関数のサンプルを収録しています。ユーザ定義関数の仕様については、対応する FIT モジュールのマニュアル等を参照してください。

表8.1　ユーザ定義関数一覧

ユーザ定義関数	ファイル名	FIT モジュール名	ドキュメント名／型番
ファイルシステムインタフェース	r_file_driver.c	r_t4_http_server_rx	6.7 ユーザ定義関数リファレンス
システムタイマインタフェース	r_sys_time.c	r_t4_http_server_rx	6.8 ユーザ定義関数リファレンス（システムタイマ関連 ）

9.　CubeSuite+を使用する場合

本アプリケーションノートは、CubeSuite+でも評価することができます。なお、ビルドを行うには「RX ファミリ用 C/C++コンパイラパッケージ V2.02.00」以降が必要です。ここでは、まだ製品版をお持ちでない場合を想定して、無償評価版を利用する例をご紹介します。

9.1　CubeSuite+の入手とインストール方法

ルネサスのホームページから CubeSuite+をダウンロードします。

1. 以下の URL にアクセスし、CubeSuite+のダウンロードページを表示します。
   http://japan.renesas.com/cubesuite+_download

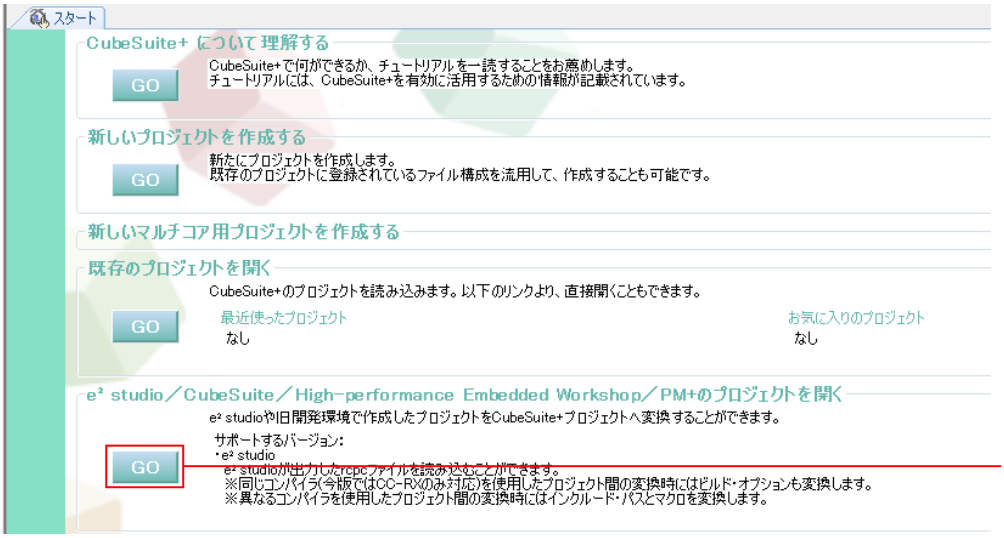
2. 表示された項目の中から、「【無償評価版】統合開発環境 CubeSuite+ V2.02.00」をクリックします。（分割ダウンロード版と一括ダウンロード版がありますが、内容の違いはありません。）
   その後表示されたページの指示に従い、CubeSuite+のインストーラをダウンロードします。

3. ダウンロードした CubeSuite+のインストーラを実行し、CubeSuite+ を PC にインストールします。
   インストール方法は「CubeSuite+ V2.02.00 統合開発環境 ユーザーズマニュアル 起動編」を参照してください。
   http://documentation.renesas.com/doc/products/tool/doc/r20ut2865jj0100_qsst.pdf

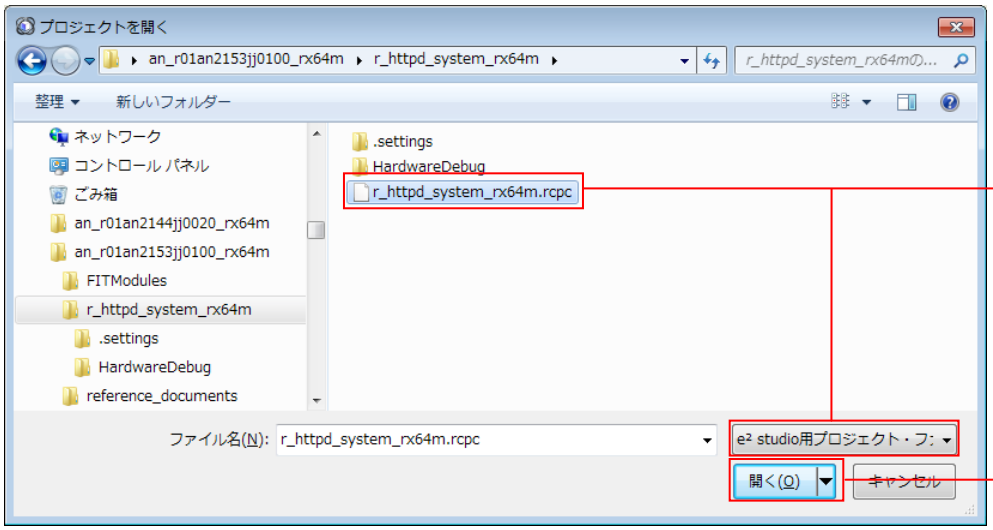
9.2　プロジェクトのインポート

CubeSuite+に、本アプリケーションノートに付属している「Renesas 共通プロジェクト・ファイル」をインポートします。

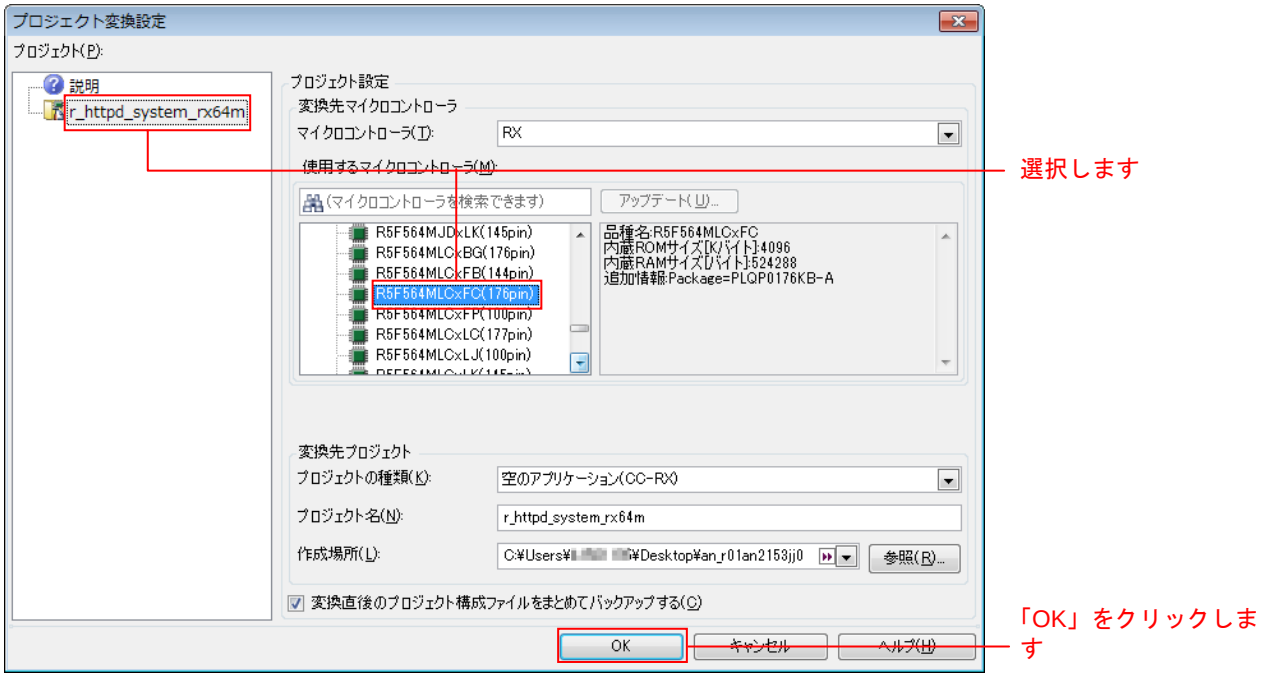
1. 「本アプリケーションノート提供 ZIP」ファイルを、任意のフォルダに解凍します。
2. CubeSuite+を起動し、スタート画面から「e² studio／CubeSuite／High-performance Embedded Workshop／PM+のプロジェクトを開く」の「GO」をクリックします。



3. 1.で解凍したフォルダを開き、その中にある「Web サーバシステムプロジェクト（r_httpd_system_rx64m フォルダ）」を開き、その中にある「Renesas 共通プロジェクト・ファイル（r_httpd_system_rx64m.rcpc）」を選択し、「開く」をクリックします。



4. 「プロジェクトツリー」からプロジェクトを選択した後、以下に示す様に各項目を選択し、「OK」をクリックします。尚、「使用するマイクロコントローラ」は、使用する評価ボードに実装されているデバイスに合わせて選択してください。



5. プロジェクトの変換が行われ、変換されたプロジェクトが開かれます。また、e2 studio のプロジェクトは、バックアップが作成されます。

9.3　プロジェクトにFITモジュールを追加する

本アプリケーションノートと RX64M グループ用 RX Driver Package に入っている FIT モジュールを、プロジェクトに追加します。

追加する FIT モジュールを以下に示します。

表9.3.1　追加 FIT モジュール一覧

種類	モジュール名	FIT モジュール名	Version
Board Support Package	ボードサポートパッケージ（BSP モジュール）	r_bsp	2.60
Device Driver	コンペアマッチタイマ（CMT）	r_cmt_rx	2.30
Device Driver	イーサネットコントローラ（ETHERC）	r_ether_rx	1.00
Middleware	M3S-T4-Tiny インタフェース変換モジュール	r_t4_driver_rx64m	1.00
Middleware	TCP/IP プロトコルスタックライブラリ（M3S-T4-Tiny）	r_t4_rx	2.00
Middleware	FAT ファイルシステム（M3S-TFAT-Tiny）	r_tfat_rx	3.00
Device Driver	USB Basic Firmware	r_usb_basic	1.00
Device Driver	USB ホストマスストレージクラス	r_usb_hmsc	1.00
Application	HTTP サーバ	r_t4_http_server_rx	1.03
Application	Web サーバシステムメインプログラム	r_httpd_main_rx64m	1.00

FIT モジュールを、プロジェクトに追加する方法は、「RX ファミリ　CubeSuite+に組み込む方法」を参照してください。

http://documentation.renesas.com/doc/products/mpumcu/apn/rx/r01an1826jj0100_rx.pdf

10. 補足

10.1　USB ドライバの制限事項

USB の ch0 と ch1 の両方をホストモードに設定した場合、ch0 側でしか USB メモリを認識しません。ch1 側をホストモードで使用したい場合は、ch0 側を未使用かペリフェラルモードに設定してください。

【r_config/r_usb_config.h】

```
/* Select USB mode(Host or Periphera) per each USB IP */
//  #define USB_FUNCSEL_USBIP0_PP      USB_HOST_PP    /* Host Mode */
//  #define USB_FUNCSEL_USBIP0_PP      USB_PERI_PP    /* Peripheral Mode */
    #define USB_FUNCSEL_USBIP0_PP      USB_NOUSE_PP

    #define USB_FUNCSEL_USBIP1_PP      USB_HOST_PP    /* Host Mode */
//  #define USB_FUNCSEL_USBIP1_PP      USB_PERI_PP    /* Peripheral Mode */
//  #define USB_FUNCSEL_USBIP1_PP      USB_NOUSE_PP
```

図10.1.1　ch1 側をホストモードで使用する場合

10.2　Web サーバシステムの制限事項

プログラム動作後に USB メモリを抜いて、再度挿入した場合、USB メモリが認識されません。プログラムを再起動してください。

10.3　無償評価版の「RX ファミリ用 C/C++コンパイラパッケージ」を利用する場合の注意事項

無償評価版の「RX ファミリ用 C/C++コンパイラパッケージ」には、使用期限と使用制限があります。使用期限が過ぎた場合、使用制限によりロードモジュールが正しく生成されなくなる場合があります。

詳しくは、ルネサスのホームページにある、評価版ソフトウェアツールのページを参照してください。

URL：http://japan.renesas.com/products/tools/evaluation_software/index.jsp

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2014.09.01	ー	初版発行

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. 未使用端子の処理
   【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】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. リザーブアドレス（予約領域）のアクセス禁止
   【注意】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、将来の機能拡張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。
4. クロックについて
   【注意】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。
5. 製品間の相違について
   【注意】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、内部ROM、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、ノイズ輻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、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 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に関し、当社は、何らの責任を負うものではありません。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. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製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）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、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（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、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。お客様が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. 本資料に記載されている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。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
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。

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その総株主の議決権の過半数を直接または間接に保有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。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/

Colophon 3.0