

AP-RX72N-0A (RX72N CPU BOARD)

wolfSSL サンプルプログラム解説

2.1版 2023年10月02日

1. 概要

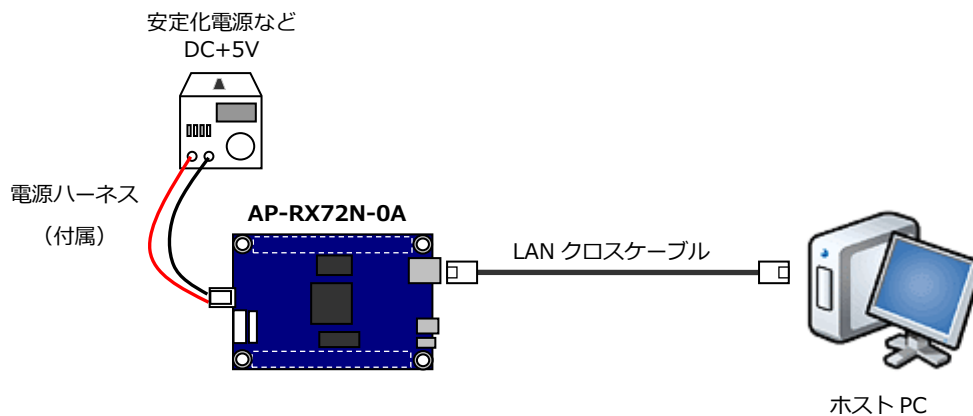
1.1 概要

本アプリケーションノートでは、弊社製 CPU ボード AP-RX72N-0A を用いて、wolfSSL を使用したネットワーク通信を動作させるサンプルプログラムについて説明します。

サンプルプログラム	動作内容
AP-RX72N-0A wolfSSL サンプルプログラム	<ul style="list-style-type: none">・ネットワーク通信・セキュリティ (wolfSSL)

1.2 接続概要

「wolfSSL サンプルプログラム」の動作を確認する上で必要な CPU ボードとホスト PC 間の接続例を以下に示します。詳細な接続に関しては後述の「3. 動作説明」を参照してください。



1.3 本サンプルプログラムについて

本サンプルプログラムは、ルネサス エレクトロニクス株式会社提供のミドルウェアおよびドライバを AP-RX72N-0A に移植しています。

各ミドルウェアおよびドライバの詳細については、以下の資料を参照してください。

入手につきましては、ルネサス エレクトロニクス株式会社ウェブサイトの下記のページにて、検索を行ってください。

FIT モジュールにつきましては、Smart Configurator から入手することも可能です。

(RX Driver Package は、ver 1.26 を使用しています。)

ルネサス エレクトロニクス社 RX72N サンプルコード

<https://www.renesas.com/jp/ja/products/microcontrollers-microprocessors/rx-32-bit-performance-efficiency-mcus/rx72n-rx-family-flagship-32-bit-microcontroller-highest-performance#documents>

● BSP
・資料名 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 機能名称: BSP <R01AN1685 Rev 5.52>
● BYTEQ
・資料名 RX ファミリ バイト型キューバッファ (BYTEQ) モジュール Firmware Integration Technology 機能名称: その他 <R01AN1683 Rev 1.80>
● CMT
・資料名 RX ファミリ CMT モジュール Firmware Integration Technology 機能名称: タイマ <R01AN1856 Rev 4.40>
● ネットワーク通信
・資料名 RX ファミリ イーサネットモジュール Firmware Integration Technology 機能名称: Ethernet <R01AN2009 Rev 1.20> RX ファミリ Ethernet ドライバと組み込み用 TCP/IP M3S-T4-Tiny のインタフェース変換モジュール Firmware Integration Technology 機能名称: 組み込み用 TCP/IP M3S-T4-Tiny <R01AN0311 Rev 1.08>
● TCP/IP サンプルプログラム
・資料名 RX ファミリ 組み込み用 TCP/IP M3S-T4-Tiny を用いたサンプルプログラム 機能名称: TCP/IP (サンプルプログラム) <R01AN0312 Rev 1.06>
● タイマ
・資料名 RX ファミリ システムタイマモジュール Firmware Integration Technology 機能名称: DMAC <R20AN0431 Rev 1.01>

(※) 資料をダウンロードするにはルネサス エレクトロニクス株式会社の My Renesas への登録が必要となります。

1.4 開発環境について

本サンプルプログラムは、統合開発環境「CS+」と「Smart Configurator」を用いて開発されています。

本サンプルプログラムに対応する開発環境、コンパイラのバージョンは次の通りです。

ソフトウェア	バージョン	備考
CS+	v8.03.00	–
RX用コンパイラ CC-RX	V3.02.00	–
Smart Configurator	v2.5.0	RX Driver Package は、ver 1.26 を使用。
Visual Studio 2019 Professional	16.7.4	ツールセット Visual Studio 2017 (v142) ※動作確認用 PC アプリ作成に使用

1.5 ワークスペースについて

本サンプルプログラムのプロジェクトファイルは次のフォルダに格納されています。

サンプルプログラム	フォルダ
wolfSSL サンプルプログラム プロジェクトフォルダ	¥Sample¥ap_rx72n_0a_ether_sample

1.6 wolfSSL について

wolfSSL とは、wolfSSL 社製のマイコン組込みシステム向けの軽量 SSL/TLS ライブラリです。最新プロトコル標準、暗号アルゴリズムにも対応し、世界中の組込み製品で利用されています。wolfSSL の製品は、オープンソース版と商用版の 2 種類があります。デバイスや商用ソフトウェアに wolfSSL 製品の採用を希望される場合、商用版のご契約が必要です。詳細や使用条件などにつきましては、wolfSSL 社 Web サイトをご覧ください。

wolfSSL 日本語サイト : <https://www.wolfssl.jp/>

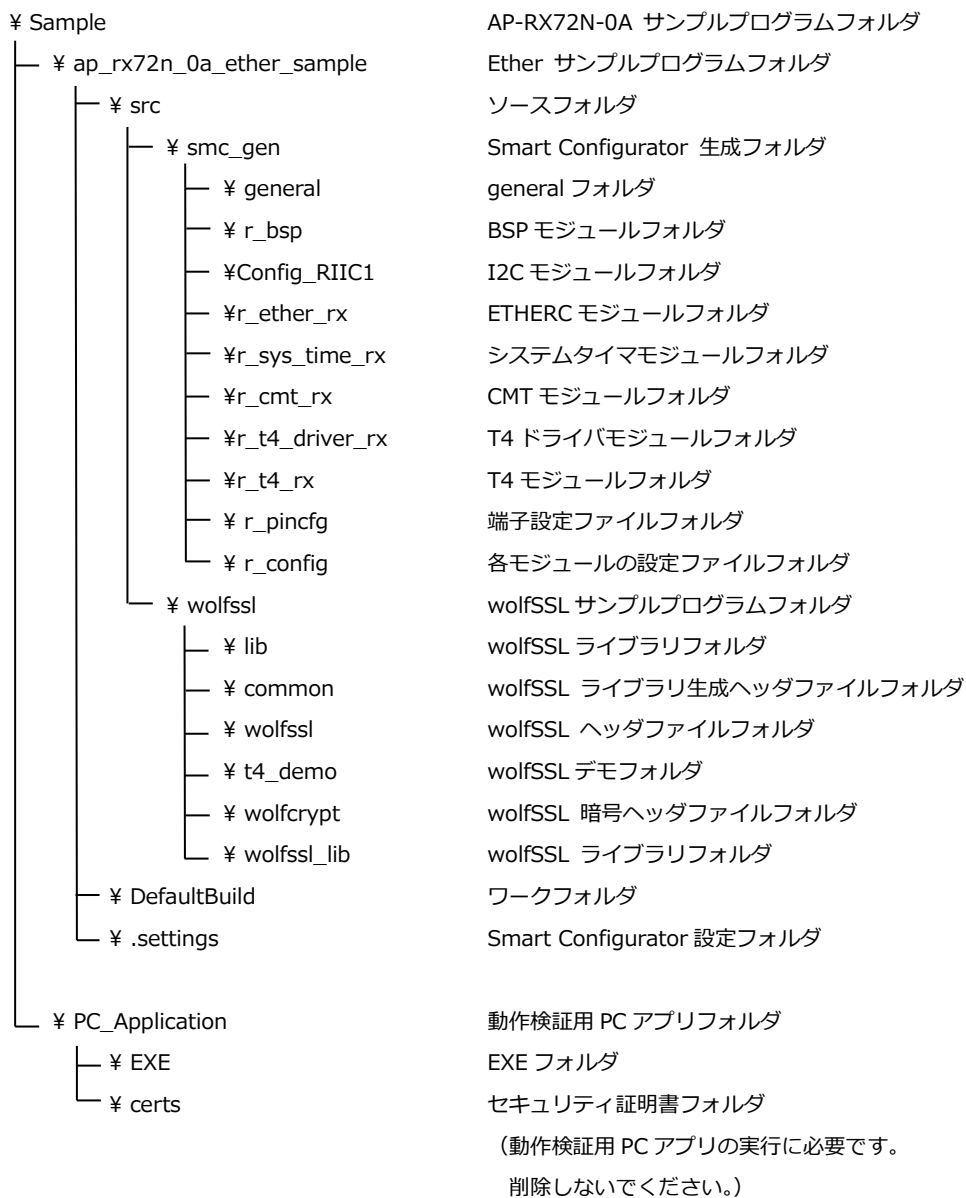
本サンプルプログラムでは、「オープンソース版 v4.5.0」を使用しています。オープンソース版の使用条件につきましては、wolfSSL 社 Web サイト 組込み SSL ライブラリ オープンソース版ダウンロードページにて、「使用許諾契約」をご確認ください。

wolfSSL 社 組込み SSL ライブラリページ : <https://www.wolfssl.jp/products/wolfssl/>

2. サンプルプログラムの構成

2.1 フォルダ構成

サンプルプログラムは下記のようなフォルダ構成になっています。



2.2 ファイルの構成

本サンプルプログラムは以下のファイルで構成されています。

本章では、ミドルウェア・ドライバ等の既存のファイルに関しては説明を省略してあります。

<¥Sample¥PC_Application フォルダ内>

client.bat	...	client.exe 起動用バッチファイル
server.bat	...	server.exe 起動用バッチファイル

<¥Sample¥PC_Application¥EXE フォルダ内>

client.exe	...	クライアント動作 PC アプリケーション
server.exe	...	サーバ動作 PC アプリケーション

<¥Sample¥ap_rx72n_0a_ether_sample フォルダ内>

ap_rx72n_0a_ether_sample.mtpj	...	CS+用プロジェクトファイル
ap_rx72n_0a_ether_sample.cfg	...	Smart Configurator 用ファイル (CS+上から Smart Configurator を起動できます。)
ap_rx72n_0a_ether_sample.rcpe	...	e2studio 用プロジェクトファイル
ap_rx72n_0a_ether_sample_	...	Board Description File
V1.0.bdf		(本プログラムのクロック周波数、端子設定を Smart Configurator にインポートできます。)

<¥Sample¥ap_rx72n_0a_ether_sample¥DefaultBuild フォルダ内>

ap_rx72n_0a_ether_sample.abs	...	elf 形式オブジェクトファイル
ap_rx72n_0a_ether_sample.mot	...	モトローラ S フォーマット形式ファイル
ap_rx72n_0a_ether_sample.map	...	マップファイル

<¥Sample¥ap_rx72n_0a_ether_sample¥src フォルダ内>

smc_gen	...	Smart Configurator により生成されたモジュールフォルダ
main.c	...	メイン処理ソースファイル
echo_srv_tcp_blocking.c	...	TCP エコーバック処理ソースファイル
cmt_dev.c	...	タイマドライバソースファイル
i2c_dev.c	...	I2C ドライバソースファイル
eprom.c	...	EEPROM ドライバソースファイル
echo_srv.h	...	TCP エコーバック処理ヘッダファイル
cmt_dev.h	...	タイマドライバヘッダファイル
i2c_dev.h	...	I2C ドライバヘッダファイル
eprom.h	...	EEPROM ドライバヘッダファイル

<¥Sample¥ap_rx72n_0a_ether_sample¥src¥wolfssl フォルダ内>

wolf_client.c	...	wolfSSL Client 処理ソースファイル
wolf_server.c	...	wolfSSL Server 処理ソースファイル
wolfssl_console.c	...	標準入出力用関数ソースファイル
wolf_demo.h	...	wolfSSL サンプルプログラムヘッダファイル

<¥Sample¥ap_rx72n_0a_ether_sample¥src¥wolfssl¥lib フォルダ内>

wolfssl_lib.lib	...	wolfSSL ライブラリファイル (v4.5.0)
-----------------	-----	----------------------------

3. プログラム作成方法

本章では、AP-RX72N-0A の Ethernet のサンプルプログラムに wolfSSL を追加する方法を説明します。
なお、本サンプルプログラムを動作させる場合には、本章の手順は必要ありません。
動作方法に関しては、「4. 動作説明」をご覧ください。

3.1 プロジェクトの準備

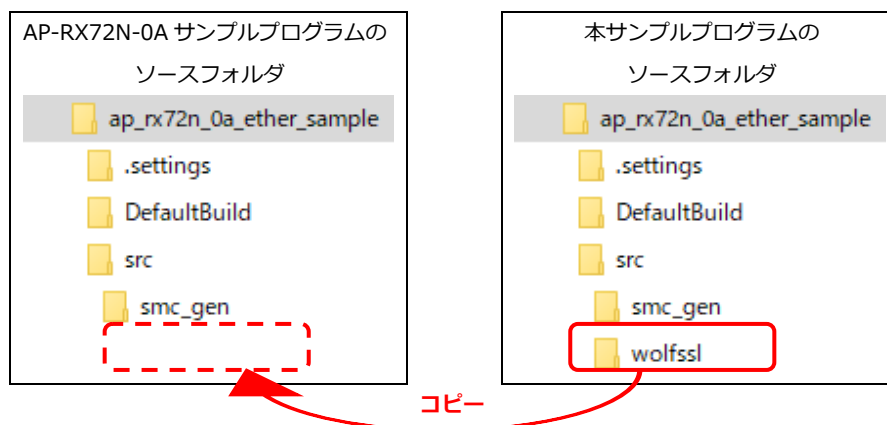
AP-RX72N-0A のサンプルプログラムに wolfSSL を追加するために、以下のプログラムを準備してください。

- ① AP-RX72N-0A サンプルプログラムの、Ethernet サンプルプロジェクト
(弊社 Web サイト AP-RX72N-0A 製品ページよりダウンロード可能なサンプルプログラム
『AP-RX72N-0A サンプルプログラム』に含まれています。)
 - ・ ap_rx72n_0a_ether_sample
- ② 本サンプルプログラム
 - ・ ap_rx72n_0a_ether_sample

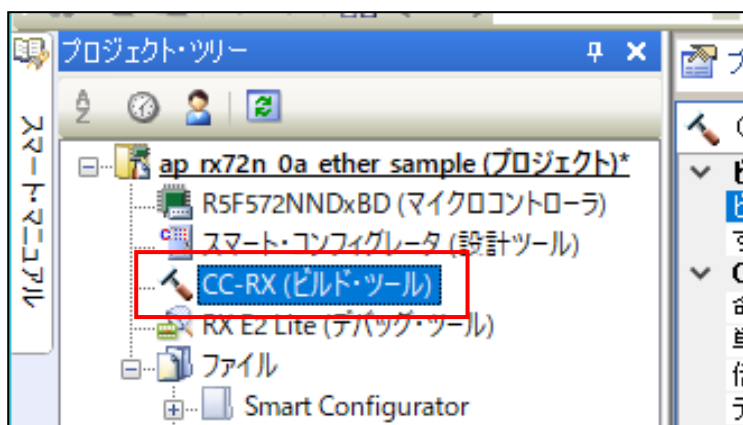
3.2 プロジェクト設定

「3.1 プロジェクトの準備」で用意した、元とする AP-RX72N-0A サンプルプログラムの mtpj ファイルを CS+ で読み込み、以下の手順でプロジェクトの設定を行ってください。

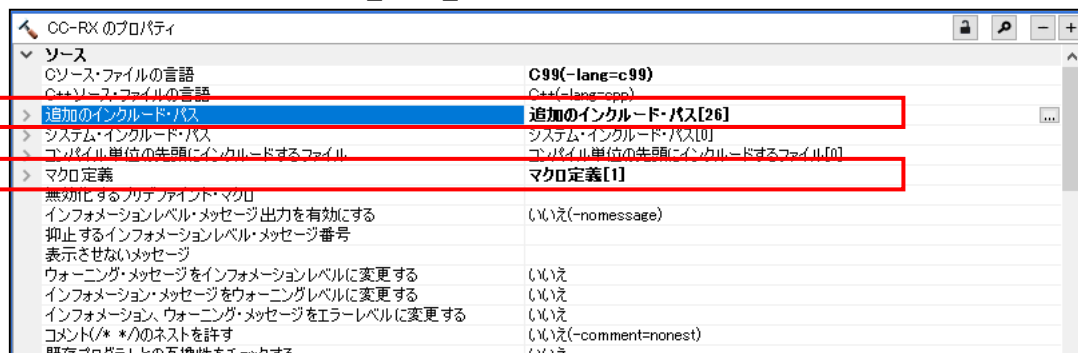
- ① エクスプローラ上で、元とするサンプルプログラムのソースフォルダに、本サンプルプログラムの¥src フォルダ内のフォルダ「¥wolfssl」をコピーします。



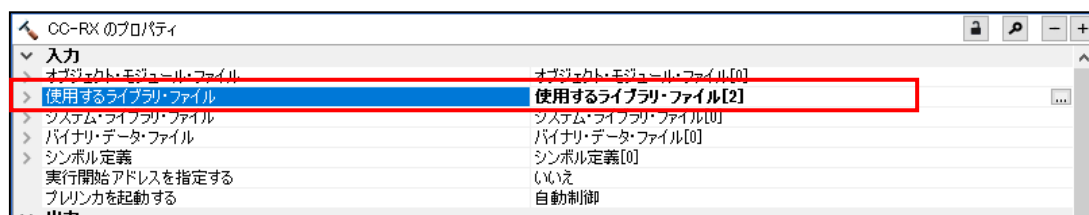
- ② CS+上で、「プロジェクト・ツリー」から、「CC-RX (ビルド・ツール)」のプロパティを開きます。



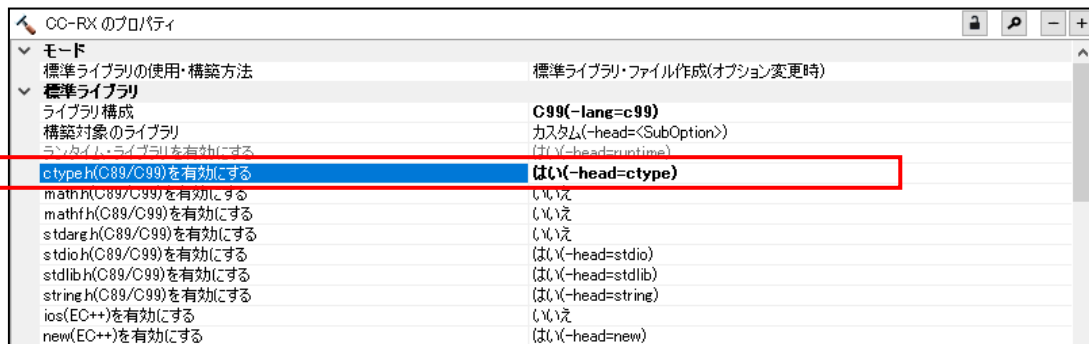
- ③ プロパティの「コンパイル・オプション」タブを開き、
「追加のインクルード・パス」に「src¥wolfssl」と「src¥wolfssl¥common」を追加します。
また、「マクロ定義」に「WOLFSSL_USER_SETTINGS」を追加します。



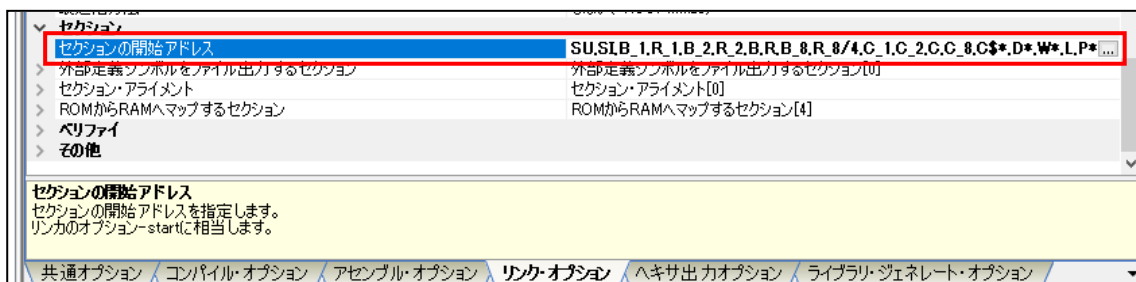
- ④ プロパティの「リンク・オプション」タブを開き、
「使用するライブラリ・ファイル」に「src¥wolfssl¥lib¥wolfssl_lib.lib」を追加します。



- ⑤ プロパティの「ライブラリ・ジェネレート・オプション」タブを開き、「ctype.h(C89/C99)を有効にする」を「はい(-head=ctype)」に設定します。



- ⑥ プロパティの「リンク・オプション」タブを開き、「セクションの開始アドレス」を選択して「B_ETHERNET_BUFFERS_1」「B_RX_DESC_1」「B_TX_DESC_1」の3セクションの配置アドレスを「0x00022000」に変更します。



⑦ プロジェクトに、①でコピーしたフォルダ「wolfssl」内の次のソースファイルを追加します。

- ・ %wolfssl%\wolf_client.c
- ・ %wolfssl%\wolf_server.c
- ・ %wolfssl%\wolf_console.h
- ・ %wolfssl%\wolf_demo.h
- ・ %wolfssl%\t4_demo\benchmark.c
- ・ %wolfssl%\t4_demo\test.c
- ・ %wolfssl%\common\wolfssl_dummy.c
- ・ %wolfssl%\common\user_settings.h

ファイルの追加は、「プロジェクト」 - 「追加」 - 「既存のファイルを追加」から登録するか、

「プロジェクト・ツリー」にファイルをドロップすることで行うことができます。

なお、%wolfssl%\lib内のライブラリファイル「**wolfssl_lib.lib**」については、④でプロジェクトに登録しているため、プロジェクトから外すか、ビルド対象から除外してください。

※ ここでは、本サンプルプログラムから AP-RX72N-0A サンプルプログラムに wolfSSL 処理を移植する方法を説明しています。

最新の wolfSSL などの追加方法に関しては、「6. wolfSSL の入手方法」をご覧ください。

3.3 ソースファイルの変更

元となる AP-RX72N-0A サンプルプログラムのソースを変更します。

一部のファイルの変更は Smart Configurator を用いて行いますが、ファイルを直接変更することもできます。

- ① ソースファイル「main.c」を編集します。

各モジュールの初期化終了後、メインのサーバ処理より前に、「wolfSSL_main();」を追加します。

<ソース例>

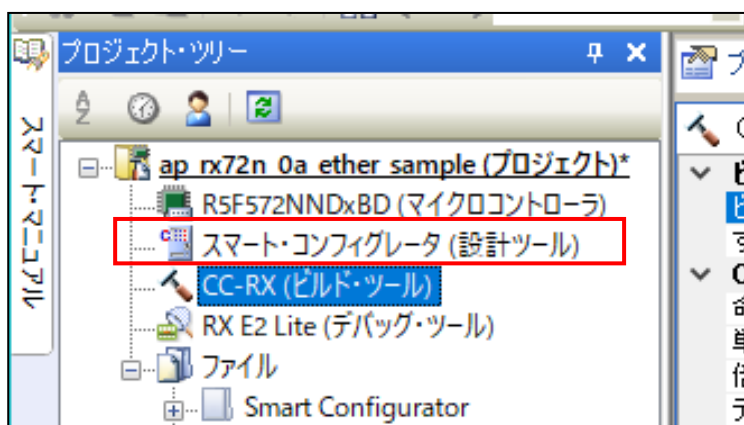
```
ercd = tcpudp_open(tcpudp_work);
if (ercd != E_OK)
{
    /* Specifies that the integer is of type bool. */
    /* WAIT_LOOP */
    while ((bool)1)
    {
        /* Do Nothing. */
    };
}

wolfSSL_init();      ← 追加

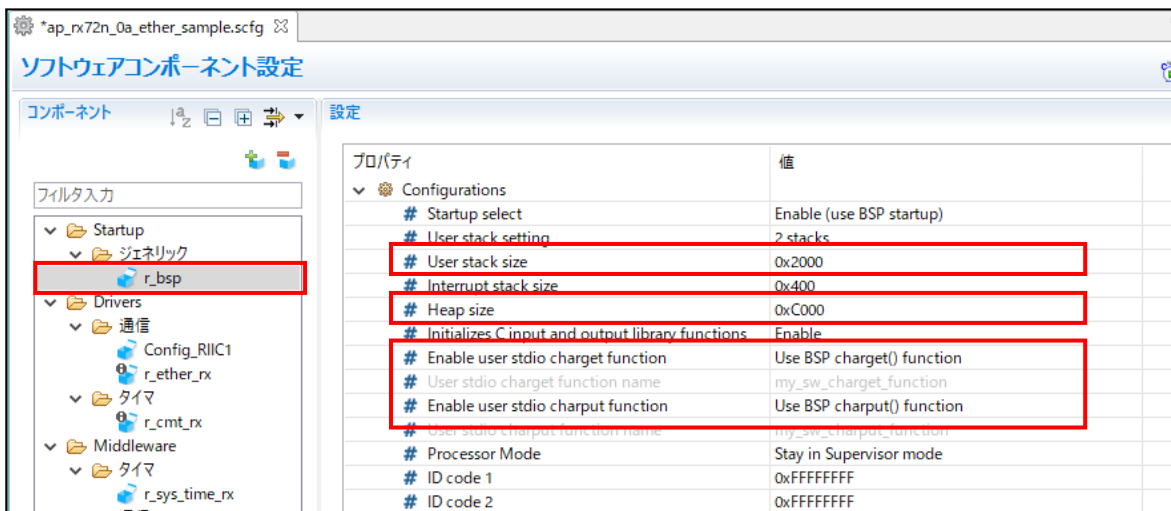
/* start echo server */
echo_srv();
```

- ② CS+の「プロジェクト・ツリー」から、Smart Configurator を起動します。

Smart Configurator を使用しない場合は、次の手順へ進んでください。

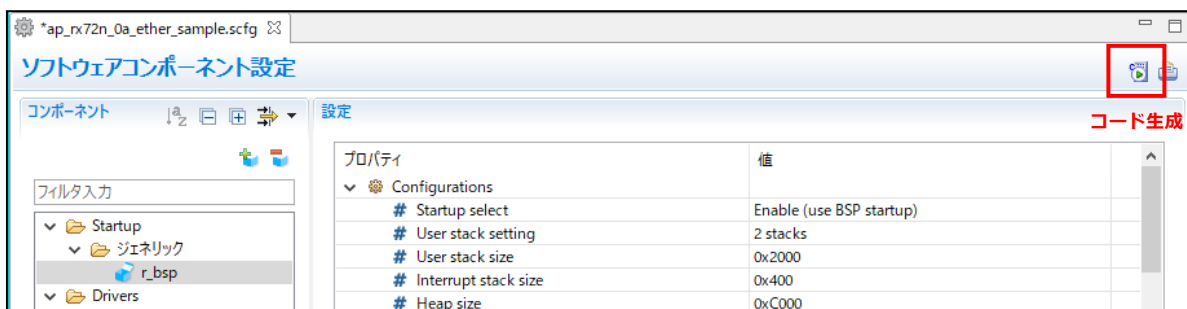


- ③ Smart Configurator 上で、「コンポーネント」タブを開き、「r_bsp」を選択します。
 「User stack size」を「0x2000」、「Heap size」を「0xC000」に設定します。
 また、「Enable user stdio charget function」を「Use BSP charget() function」、
 「Enable user stdio charput function」を「Use BSP charput() function」に設定します。
 ※ Smart Configurator を使用しない場合は、「¥src¥smc_gen¥r_config¥r_bsp_config.h」を変更してください。



変更する項目	設定値	Smart Configurator (コンポーネントタブ内 r_bsp)	対応するソースファイルの箇所 (¥src¥smc_gen¥r_config ¥r_bsp_config.h)
スタックサイズ	0x2000	User stack size	#pragma stack size su = 0x2000
ヒープサイズ	0xC000	Heap size	#define BSP_CFG_HEAP_BYTES
user charget() 関数の使用	Use BSP charget() function (0)	Enable user stdio charget function	#define BSP_CFG_USER_CHARGET_ENABLED
user charput() 関数の使用	Use BSP charput() function (0)	Enable user stdio charput function	#define BSP_CFG_USER_CHARPUT_ENABLED

- ④ Smart Configurator の「コードの生成」を実行します。



- ⑤ ソースファイル「echo_srv.h」を編集します。
「#define DEBUG_PRINT」をコメントアウトします。

<ソース例>

```
#ifndef ECHO_SRV_H
#define ECHO_SRV_H

/*****
Macro definitions
*****/
/* Please define the same price as _t4_channel_num variable in config_tcpudp.c. */
/* 0 = BLOCKING 1 = NON_BLOCKING */
#define APPLICATION_T4_BLOCKING_TYPE (0)

// #define DEBUG_PRINT ← コメントアウト
```

以上で、wolfSSL サンプルプログラムの追加は完了です。
ビルドを行い、動作を確認してください。

4. 動作説明

4.1 サンプルプログラムの動作

4.1.1 サンプルプログラム動作説明

本サンプルプログラムは、下記の動作を行います。

- ネットワーク通信

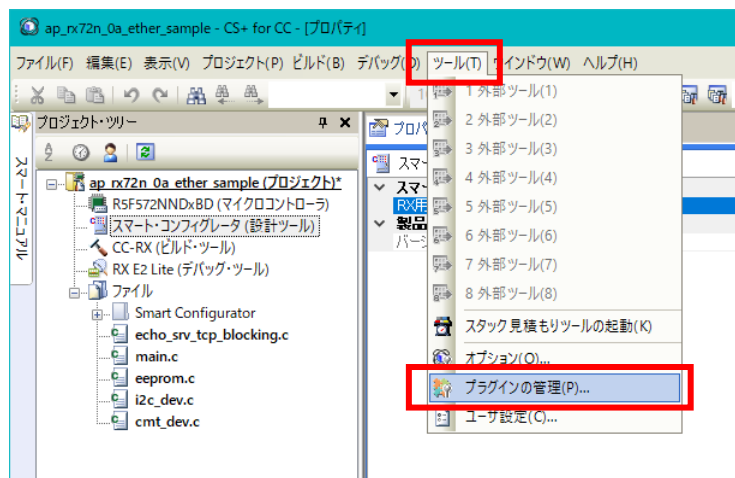
Ethernet で wolfSSL を利用した通信を行います。

- ※ 「サーバ」の動作に関しては「4.1.2 サンプルプログラム サーバ動作」を、「クライアント」の動作に関しては「4.1.3 サンプルプログラム クライアント動作」を参照してください。

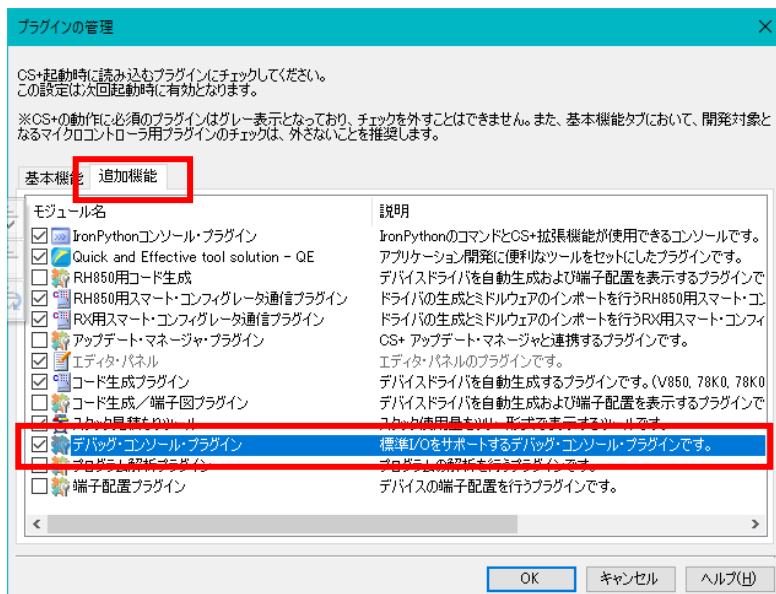
※本サンプルプログラムは CS+上のデバッグ・コンソール機能を使用して動作します。

デバッグ・コンソール機能を有効にするためには、あらかじめ以下の設定を行う必要があります。

- ① CS+を起動し、メニューバー「ツール」>「プラグインの管理」を選択してください。

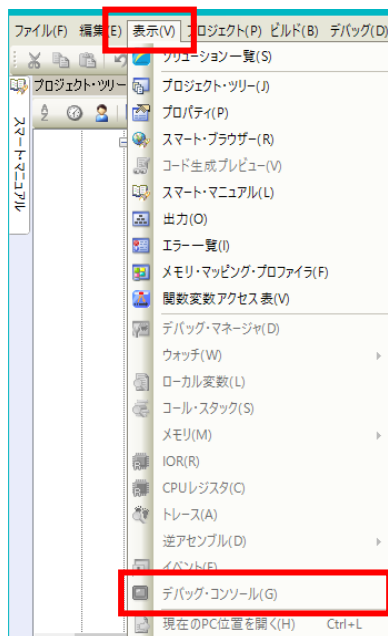


- ② プラグインの管理ウィンドウの追加機能タブにある「デバッグ・コンソール・プラグイン」にチェックをつけて「OK」ボタンを押してください。



- ③ CS+を再起動します。

再起動後、メニューバーの「表示」の中に「デバッグ・コンソール」の表示があれば正しく設定が行なえています。
なお、「デバッグ・コンソール」を選択できるのはデバッグ時のみとなりますのでご注意ください。



4.1.2 サンプルプログラム サーバ動作

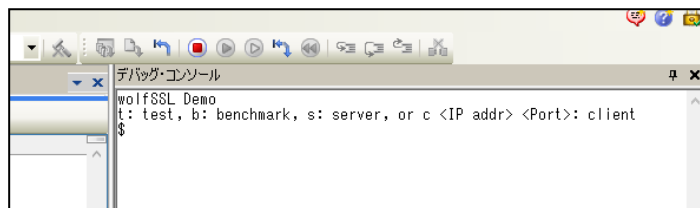
wolfSSL を利用したネットワーク動作（サーバ）の確認は、以下の手順に従って行ってください。

IP アドレス等を変更する場合には、「4.1.4 ネットワーク設定」を参照してください。

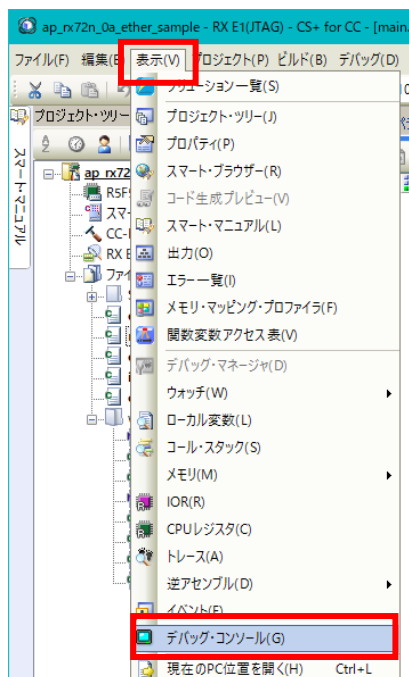
- ① 「1.2 接続概要」を参考に CPU ボードとホスト PC を接続します。
LAN クロスケーブルは、LAN コネクタ（CN4）と接続してください。
- ② ホスト PC 上でネットワークの設定を行います。
CPU ボードの設定に合わせるため、ホスト PC のネットワーク設定を下記の内容に変更してください。

IP アドレス	192.168.1.202
サブネットマスク	255.255.255.0
ゲートウェイ	192.168.1.254

- ③ CPU ボードに電源を投入し、サンプルプログラムを動作させます。
CPU ボードはサーバ設定で起動し、クライアントの接続を待機します。
デバッグ・コンソールには以下のように表示されます。

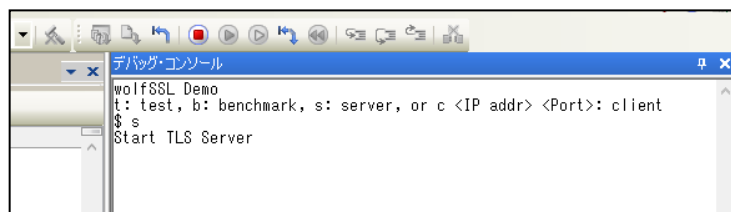


（※デバッグ・コンソールは、CS+上の「表示」>「デバッグ・コンソール」を選択することで表示できます）



- ④ デバッグ・コンソール上で「s」と入力します。

※ 最後に改行コードを送信してください。



- ④ ホスト PC 上で、フォルダ「¥Sample¥PC_Application」内の「client.bat」を実行します。

「client.bat」の実行により「¥Sample¥PC_Application¥EXE¥client.exe」が起動し、ホスト PC がクライアントとして動作します。

※ 「client.bat」「client.exe」の実行には、「certs」フォルダが必要です。「certs」フォルダは削除しないでください。

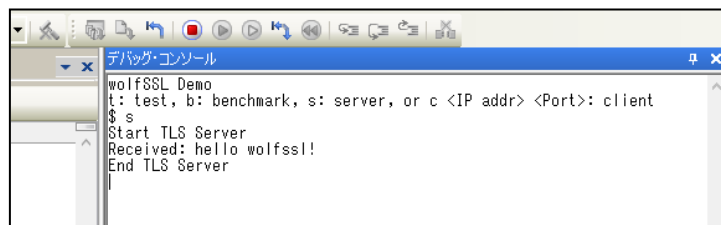
※ 「client.bat」は、CPU ボードが以下の設定であることを前提に作成されています。

IP アドレス	192.168.1.200
ポート番号	1024

- ⑤ CPU ボード（サーバ）とホスト PC（クライアント）の接続が完了すると、ホスト PC（client.exe）は CPU ボードに

「hello wolfssl!」を送信します。

CPU ボードが受信した文字列はデバッグ・コンソールに表示されます。



以上で、サーバ動作の確認は終了です。

SSL 通信の確認に関しては、「4.1.5 通信プロトコルの確認方法」を参照してください。

※ CPU ボードの IP アドレスやポート番号を初期値から変更している場合、「client.bat」の変更が必要です。

バッチファイルの書き換えについては、「6.3 PC アプリの作成」を参考にしてください。

4.1.3 サンプルプログラム クライアント動作

wolfSSL を利用したネットワーク動作（クライアント）の確認は、以下の手順に従って行ってください。

IP アドレス等を変更する場合には、「4.1.4 ネットワーク設定」を参照してください。

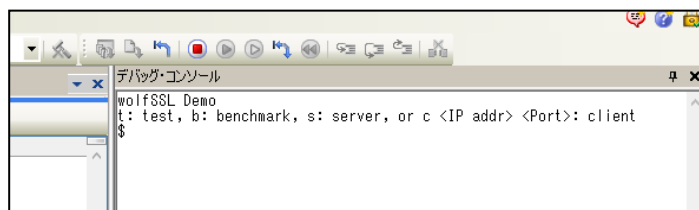
- ① 「1.2 接続概要」を参考に CPU ボードとホスト PC を接続します。
LAN クロスケーブルは、LAN コネクタ（CN4）と接続してください。
- ② ホスト PC 上でネットワークの設定を行います。
CPU ボードの設定に合わせるため、ホスト PC のネットワーク設定を下記の内容に変更してください。

IP アドレス	192.168.1.202
サブネットマスク	255.255.255.0
ゲートウェイ	192.168.1.254

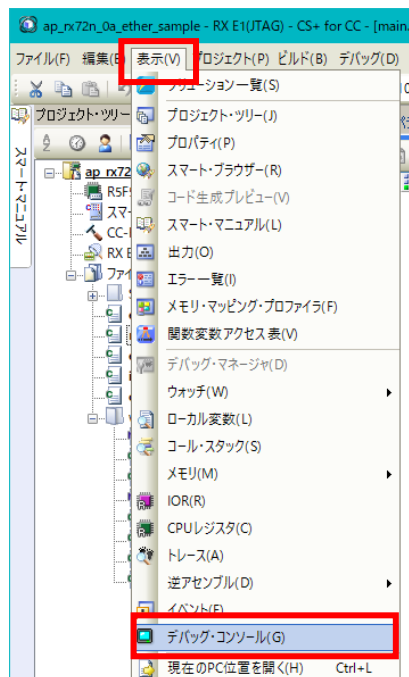
- ③ ホスト PC 上で、フォルダ「¥Sample¥PC_Application」内の「server.bat」を実行します。
「server.bat」の実行により「¥Sample¥PC_Application¥EXE¥server.exe」が起動し、ホスト PC がサーバとして動作します。
※「server.bat」「server.exe」の実行には、「certs」フォルダが必要です。
「certs」フォルダは削除しないでください。
※CPU ボードは以下の設定であるものとします。

IP アドレス	192.168.1.200
---------	---------------

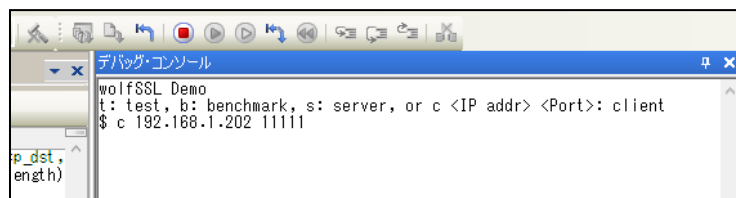
- ④ CPU ボードに電源を投入し、サンプルプログラムを動作させます。
CPU ボードはクライアント設定で起動し、デバッグ・コンソールに以下のように表示されます。



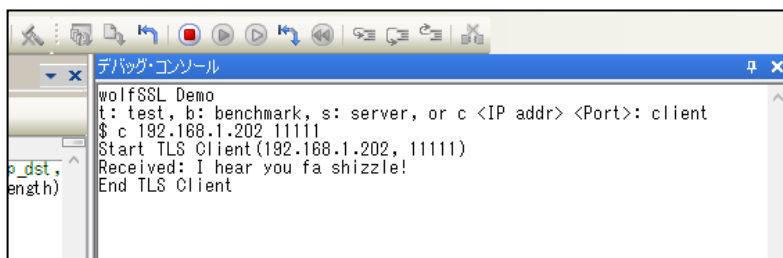
(※デバッグ・コンソールは、CS+上の「表示」>「デバッグ・コンソール」を選択することで表示できます)



- ⑤ ターミナルソフトから、接続するサーバの IP アドレスとポート番号を指定します。
IP アドレスはホスト PC の IP アドレス、ポート番号は「11111」(server.exe により固定)です。
表示されている「\$」に続いて、以下のように入力してください。
- ※ 最後に改行コードを送信してください。



- ⑥ CPU ボード (クライアント) は指定されたサーバに接続します。
接続が完了すると、ホスト PC (server.exe) は CPU ボードに「I hear you fa shizzle!」を送信します。
デバッグ・コンソールには以下のように表示されます。



以上で、クライアント動作の確認は終了です。

SSL 通信の確認に関しては、「4.1.5 通信プロトコルの確認方法」を参照してください。

4.1.4 ネットワーク設定

● 推奨環境

本サンプルプログラムに実装されたネットワーク通信の確認に必要な推奨環境は以下の通りです。

ホスト PC	PC/AT 互換機
OS	Windows 10/11
LAN ポート	10/100BASE-TX 以上対応の LAN ポート
LAN ケーブル	クロスケーブル

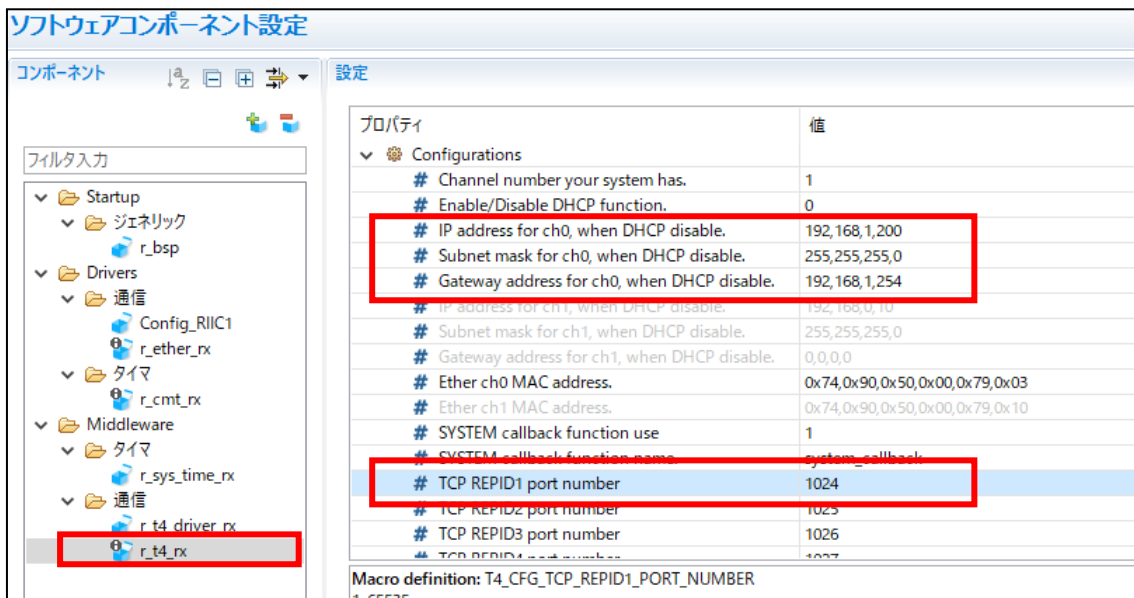
● ネットワーク設定

本 CPU ボードのネットワーク設定は以下の通りです。

	CH0
IP アドレス	192.168.1.200
サブネットマスク	255.255.255.0
ゲートウェイ	192.168.1.254
ポート番号	1024
MAC アドレス	00-0C-7B-4E-XX-XX ※ XX-XX の値は製品ごとに異なります。

IPアドレス・サブネットマスク・ゲートウェイ・ポート番号の設定は、Smart Configurator 上のコンポーネント設定>r_t4_rx で定義しています。

(対応するソースファイルとしては "%src%smc_gen%r_config%r_t4_rx_config.h")



項目	設定値	Smart Configurator (コンポーネントタブ内 r_t4_rx)	対応するソースファイルの箇所 (%src%smc_gen%r_config %r_t4_rx_config.h)
IPアドレス	192.168.1.200	IP address for ch0, when DHCP disable.	#define T4_CFG_FIXED_IP_ADDRESS_CH0
サブネットマスク	255.255.255.0	Subnet mask for ch0, when DHCP disable.	#define T4_CFG_FIXED_SABNET_MASK_CH0
ゲートウェイ	192.168.1.254	Gateway address for ch0, when DHCP disable	#define T4_CFG_FIXED_GATEWAY_ADDRESS_CH0
ポート番号	1024	TCP REPID1 port number	#define T4_CFG_TCP_REPID1_PORT_NUMBER

また、MAC アドレスは EEPROM の先頭 6Byte に格納されています。

アドレス	格納値
先頭アドレス + 0x00	0x00
+ 0x01	0x0C
+ 0x02	0x7B
+ 0x03	0x4E
+ 0x04	0xXX
+ 0x05	0xXX

※ 0xXX の値は製品ごとに異なります

本製品の MAC アドレスは、弊社が米国電気電子学会（IEEE）より取得したアドレスとなります。
MAC アドレスを変更される際は、お客様にて IEEE より MAC アドレスを取得し、設定してください。

4.1.5 通信プロトコルの確認方法

- 使用ソフトウェア

通信プロトコルの確認には、ネットワークアナライザを使用します。

ここでは、フリーソフトウェア「Wireshark」を使用した確認方法を説明します。

「Wireshark」は下記の Web サイトからダウンロードできます。

Wireshark 公式サイト： <https://www.wireshark.org/>

- 確認方法

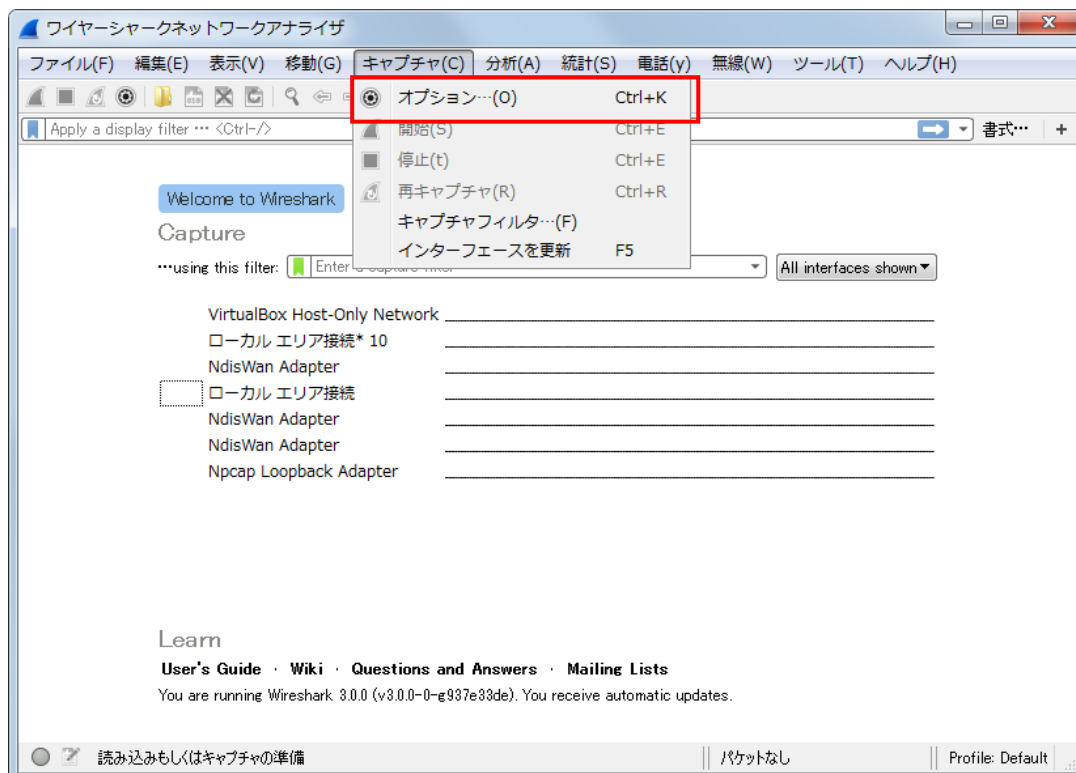
本節の説明では、「Wireshark v3.0.0」を使用しています。

その他のバージョンをご利用の場合、読み替えて行ってください。

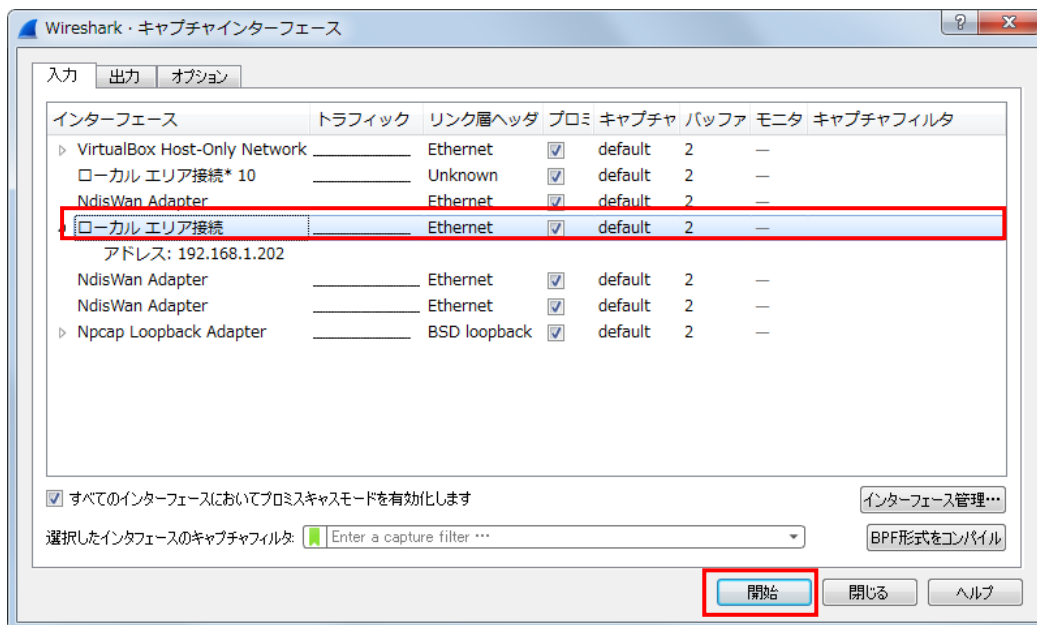
※Wireshark のバージョンにより、詳細なプロトコルが表示されないことがあります。

その場合、Wireshark のバージョンアップを行ってください。

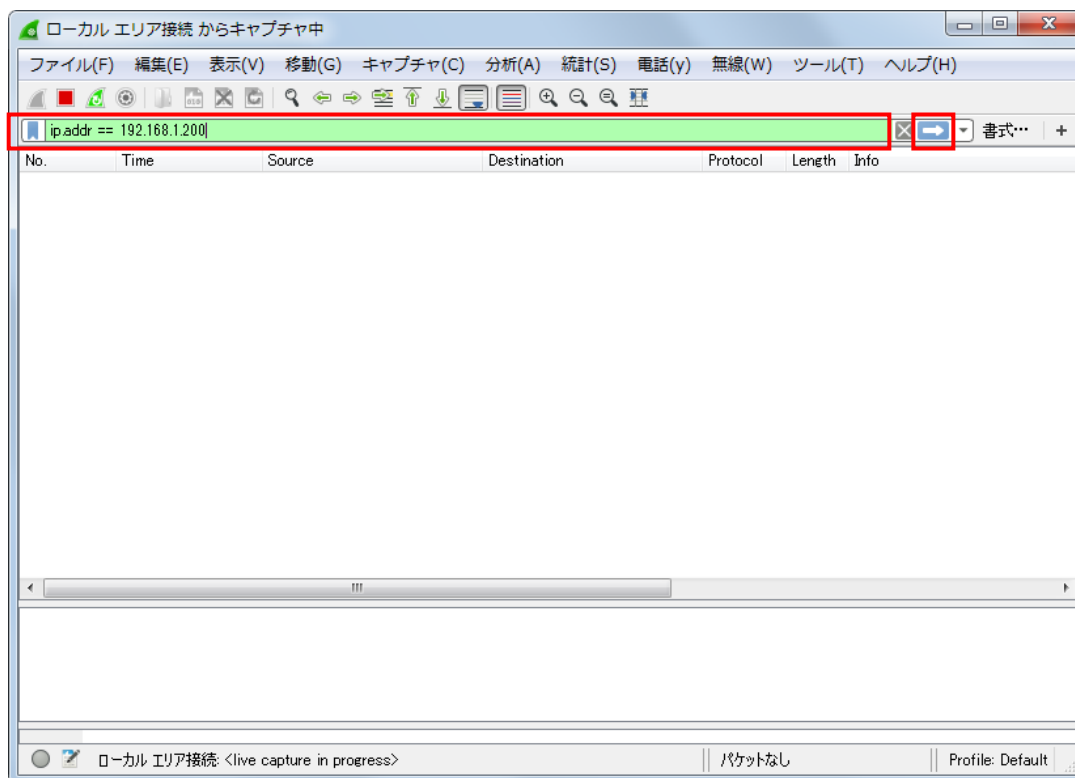
- ① Wireshark を起動し、メニューの「キャプチャ」 - 「オプション」を選択します。



- ② ホスト PC で使用するインタフェースをクリックし、「開始」を押します。
ここでは、「ローカル エリア接続」（アドレス：192.168.1.202）を選択します。

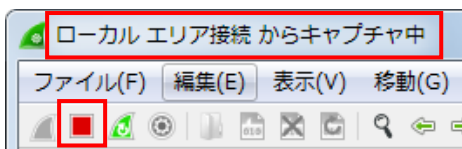


- ③ CPU ボードの IP アドレスを含むパケットのみキャプチャするように、フィルタを設定します。
フィルタ入力欄に「**ip.addr == 192.168.1.200**」を入力します。
入力後は Enter キーを押すか、Apply ボタンを押して確定してください。



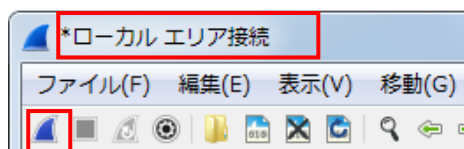
- ④ 接続を選択すると自動でキャプチャが開始します。
 手動でキャプチャを開始する場合、開始ボタンを押してキャプチャを開始してください。
 キャプチャを開始した後、サンプルプログラムを動作させます。
 サンプルプログラムの動作方法に関しては、「4.1.2 サンプルプログラム サーバ動作」または
 「4.1.3 サンプルプログラム クライアント動作」を参照してください。

<キャプチャ中の表示>



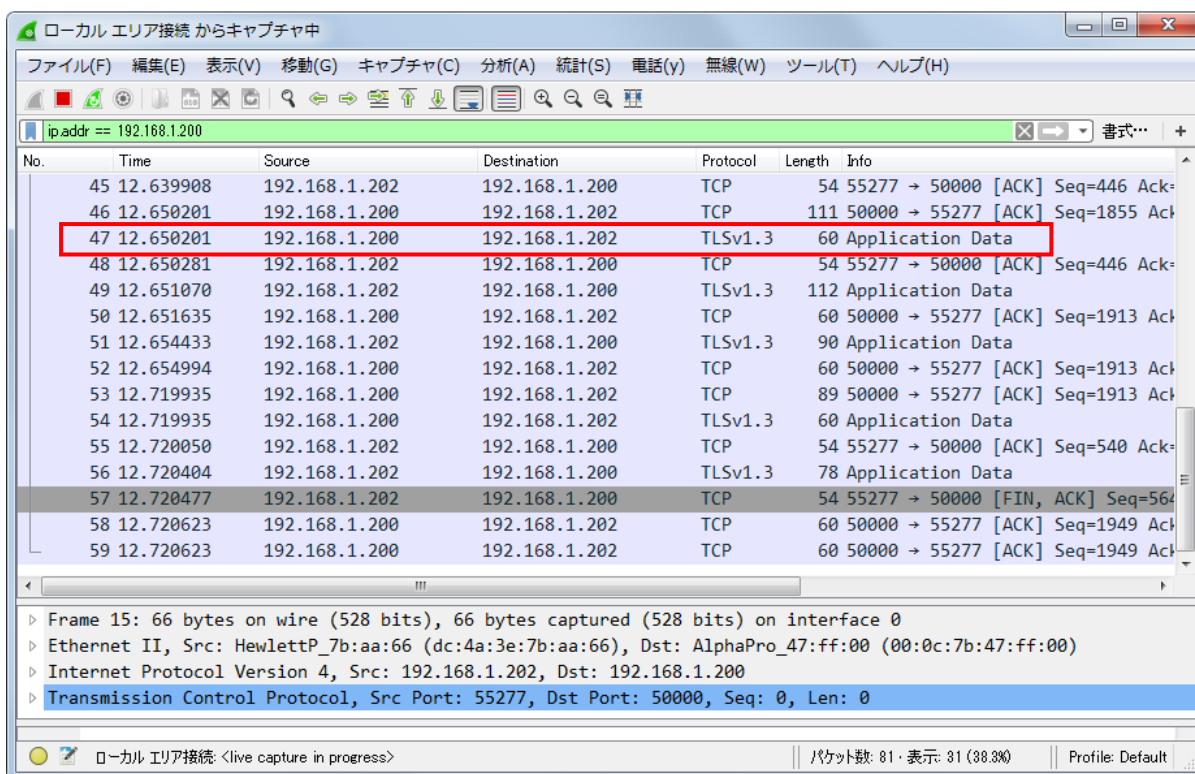
キャプチャ停止
ボタン

<停止中の表示>



キャプチャ開始
ボタン

- ⑤ 取得したデータを確認します。
 「Protocol」に「**TLSvX.X**」(X.X はバージョン)の記載があれば、その通信は暗号化通信をしています。



以上で通信プロトコルの確認は終了です。
 Wireshark の詳細な使用方法については、Wireshark のマニュアル等をご覧ください。

4.2 メモリマップ

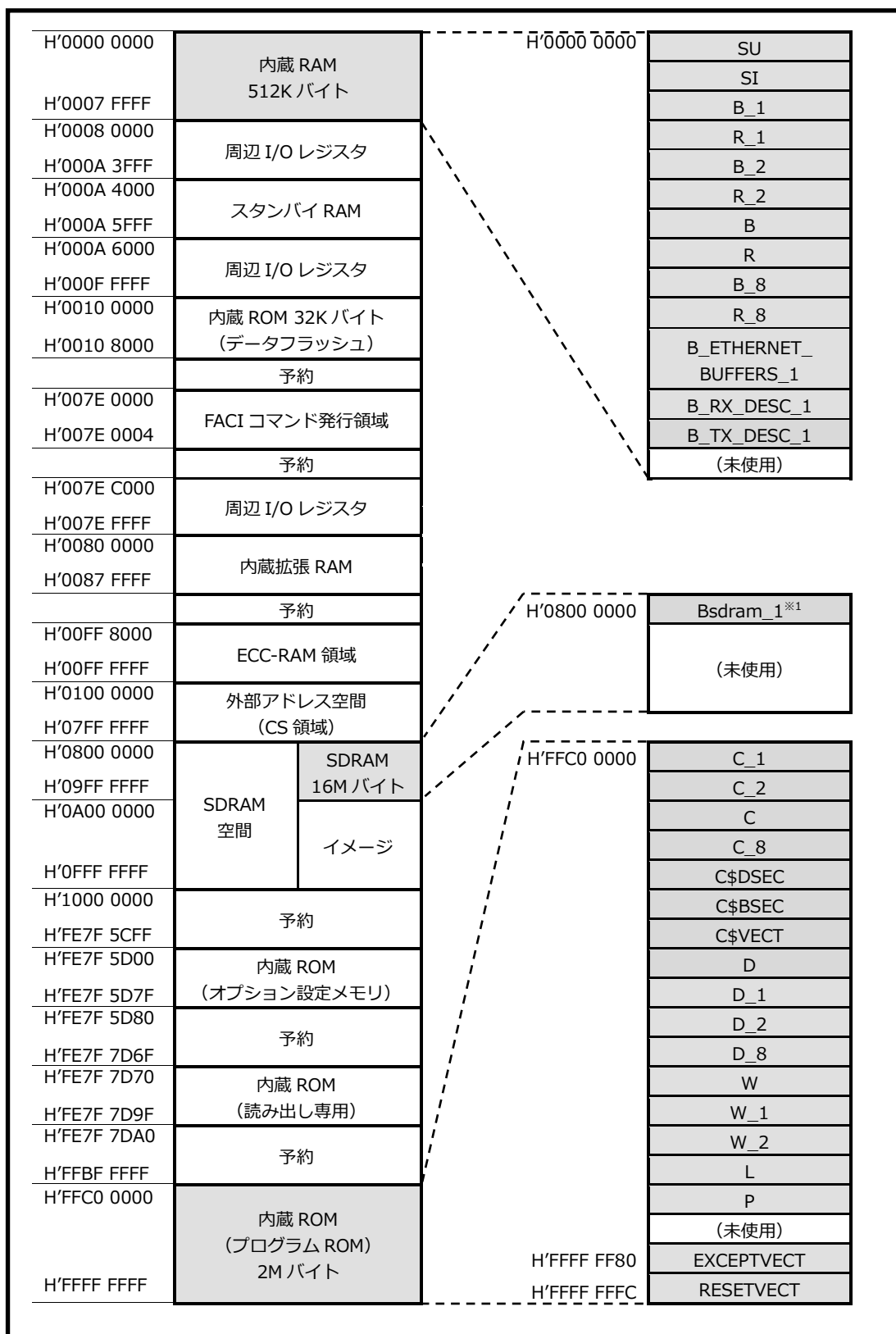


Fig 4.2-1 サンプルプログラム メモリマップ

4.3 サンプルプログラムのダウンロード

サンプルプログラムを CPU ボード上で実行するためには、ビルドしたサンプルプログラムの実行ファイルを CPU ボードにダウンロードする必要があります。

サンプルプログラムのビルド方法および CPU ボードにサンプルプログラムをダウンロードする方法については、アプリケーションノート「**AN1526 RX 開発環境の使用方法(CS+、Renesas Flash Programmer)**」に詳細な手順が記されていますので、参照してください。

5. 開発環境使用時の各設定値

開発環境を使用する際の、AP-RX72N-0A 固有の設定を以下に示します。

表内の「項目番号」はアプリケーションノート

「AN1526 RX 開発環境の使用方法(CS+, Renesas Flash Programmer)」内で示されている

項目番号を示していますので、対応したそれぞれの設定値を参照してください。

ビルド・動作確認方法		
項目名	項目番号	設定値
出力フォルダ	2-2	ap_rx72n_0a_ether_sample¥DefaultBuild
モトローラファイル名	2-3	ap_rx72n_0a_ether_sample ¥DefaultBuild¥ap_rx72n_0a_ether_sample.mot
アブソリュートファイル名	2-4	ap_rx72n_0a_ether_sample ¥DefaultBuild¥ap_rx72n_0a_ether_sample.abs
マップファイル	2-5	ap_rx72n_0a_ether_sample ¥DefaultBuild¥ap_rx72n_0a_ether_sample.map

Renesas Flash Programmer を使用した Flash 書き込み方法 (シリアルポート(SCI)を使用する方法)		
項目名	項目番号	設定値
ボード設定 (Flash 書き込み)	3-1	ボード : Fig 5-1 を参照 ケーブル接続 : CN7
Flash に書き込むファイル	3-3	ap_rx72n_0a_ether_sample ¥DefaultBuild¥ap_rx72n_0a_ether_sample.mot
ボード設定 (動作)	3-4	Fig 5-3 を参照

Renesas Flash Programmer を使用した Flash 書き込み方法 (USB ブートモードを使用する方法)		
項目名	項目番号	設定値
ボード設定 (Flash 書き込み)	3-5	ボード : Fig 5-2 を参照 ケーブル接続 : CN5 (USB microB)
ツール選択	3-6	[USB Direct]
Flash に書き込むファイル	3-7	ap_rx72n_0a_ether_sample ¥DefaultBuild¥ap_rx72n_0a_ether_sample.mot
ボード設定 (動作)	3-8	Fig 5-3 を参照

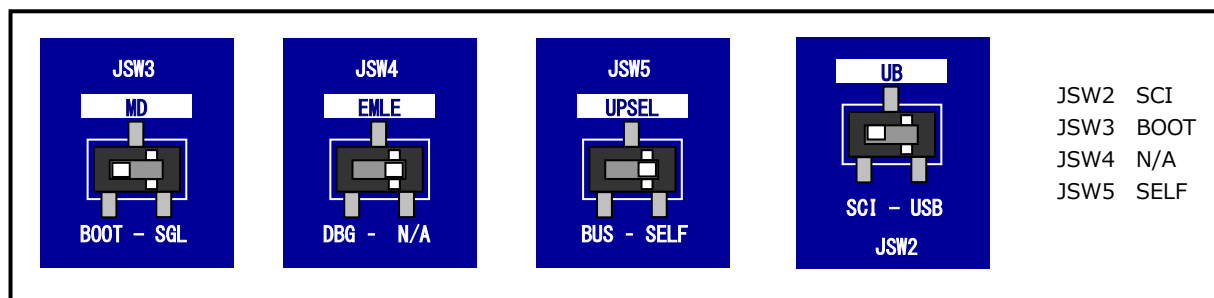


Fig 5-1 Flash 書き込み(シリアルポート使用)時のボード設定

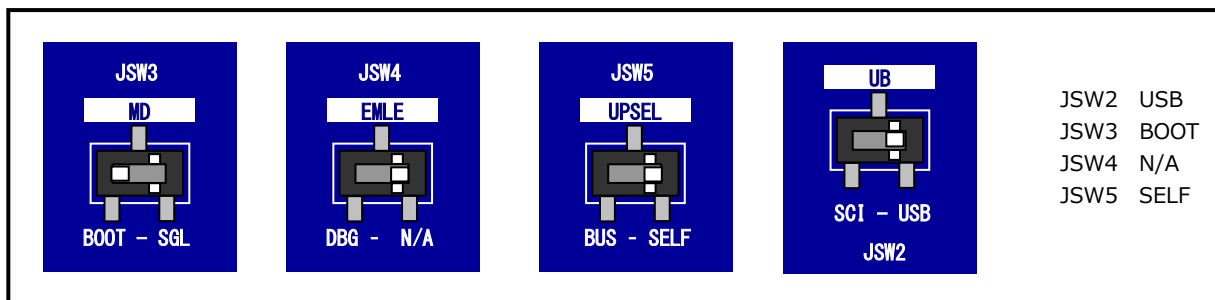


Fig 5-2 Flash書き込み(USBブートモード)時のボード設定

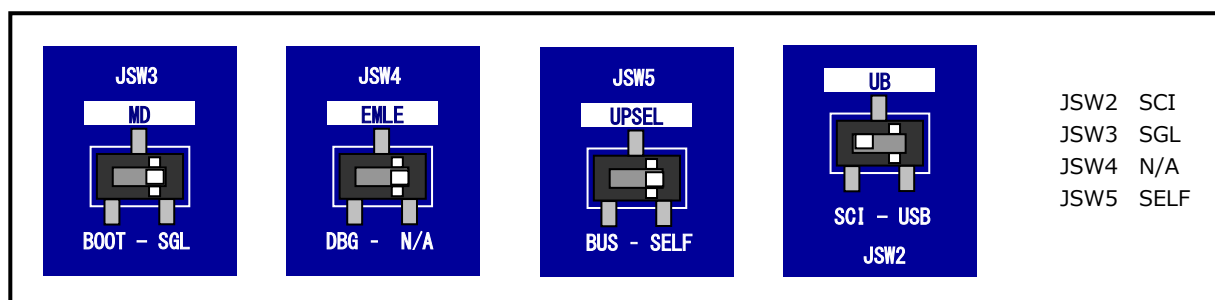


Fig 5-3 サンプルプログラム動作時のボード設定

E1 エミュレータ/E2 エミュレータ Lite を使用したデバッグ方法		
項目名	項目番号	設定値
ボード設定	4-1	Fig 5-4 を参照
JTAG クロック	4-10	E1 エミュレータを使用する場合 : 16.5(MHz) E2 エミュレータ Lite を使用する場合 : 6.00(MHz)
EXTAL クロック	4-11	24(MHz)

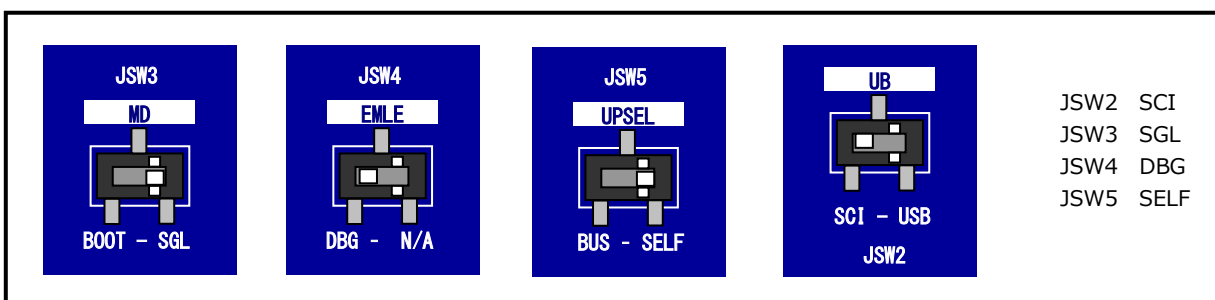


Fig 5-4 E1 エミュレータ/E2 エミュレータ Lite デバッグ時のボード設定

6. wolfSSL の入手方法

本章では、wolfSSL 組込み向け軽量 SSL/TLS ライブラリの最新版を入手し、サンプルプログラムに組み込む方法を説明します。

6.1 wolfSSL の入手

wolfSSL は、wolfSSL 社 Web サイトから入手できます。必要なバージョンをダウンロードしてください。ライセンスや商用利用につきましては、「1.6 wolfSSL について」および wolfSSL 社 Web サイトをご覧ください。

wolfSSL 社 組込み SSL ライブラリページ：<https://www.wolfssl.jp/products/wolfssl/>
(wolfSSL 日本語サイト：<https://www.wolfssl.jp/>)

6.2 ライブラリの作成、差し替え

SSL/TLS ライブラリのビルドを行います。

- ① ダウンロードしたファイルを展開し、フォルダ「¥IDE¥Renesas¥cs+¥Projects¥wolfssl_lib」内のプロジェクトファイル「wolfssl_lib.mtpj」を、CS+で開きます。
- ② メニューの「ビルド」-「ビルド・プロジェクト」を実行し、ビルドを行います。
※プロジェクトは、提供時には「RX72N」に設定されています。RX72N 以外の CPU を使用する場合、「マイクロコントローラ」と「命令セット・アーキテクチャ」を使用する CPU に合わせてから、ビルドを行います。
- ③ ビルドに成功すると、フォルダ「¥IDE¥Renesas¥cs+¥Projects¥wolfssl_lib¥DefaultBuild」にライブラリファイル「wolfssl_lib.lib」が生成します。

作成したライブラリファイルを、サンプルプログラム内のライブラリファイルと差し替えます。
 差し替えるファイルは、フォルダ「¥src¥wolfssl¥lib」の「wolfssl_lib.lib」です。
 また、必要なヘッダファイルなども差し替えます。(バージョンにより内容が異なる場合があります。)



6.3 PC アプリの作成

サンプルプログラムの動作確認に使用する PC アプリを作成します。

ここでの説明は、「Microsoft Visual Studio 2019 Professional」を元に行います。

Visual Studio の使用方法については、Visual Studio のマニュアル等をご覧ください。

- ① Microsoft Visual Studio を用いて、ダウンロードしたファイルに含まれる「wolfssl64.sln」を読み込みます。
「wolfssl64.sln」は、「wolfssl-4.5.0.zip」をダウンロードした場合、「¥wolfssl-4.5.0」に存在します。
- ② 各プロジェクトのプロパティを開き、ソースファイルの文字コードを設定します。
「wolfssl-4.5.0.zip」をダウンロードした場合、ソースファイルの文字コードは「UTF-8」のため、「プロパティ」-「C/C++」-「コマンドライン」から、「/source-charset:utf-8」を入力します。
- ③ メニューの「ビルド」-「ソリューションのリビルド」を実行し、ソリューション内の全てのプロジェクトをビルドします。
ビルドが完了すると、フォルダ「¥Debug」に実行ファイル（拡張子.exe）が生成されます。

- ④ 実行ファイルを使用するためのバッチファイルを作成します。

例（サンプルプログラム内のバッチファイル）

<client.exe 用バッチファイル>

```
cd ./EXE
client -h 192.168.1.200 -p 50000 -v 4
pause
```

※接続するサーバの IP アドレス、ポート番号を指定します。

※「-v 4」は、TLSv1.3 を指定しています。

<server.exe 用バッチファイル>

```
cd ./EXE
server -b -d -v 4
pause
```

※「-v 4」は、TLSv1.3 を指定しています。

6.4 wolfSSL の設定変更

wolfSSL のライブラリ、PC アプリのビルド時に、環境や用途に合わせて設定を変更することができます。

詳細は、wolfSSL のマニュアル等をご覧ください。

- wolfSSL ライブラリの設定
フォルダ「¥IDE¥Renesas¥cs+¥Projects¥common」内のヘッダファイル「user_settings.h」を変更できます。
- PC アプリの設定
フォルダ「¥IDE¥WIN」内のヘッダファイル「user_settings.h」を変更できます。
- 本サンプルプログラムの変更箇所
ライブラリの「user_settings.h」に以下の設定を追加しています。

```
#define WOLFSSL_TLS13
#define HAVE_FFDHE_2048
#define WC_RSA_PSS
#define HAVE_HKDF
#define USE_WOLF_TIME_T
#define HAVE_TLS_EXTENSIONS
#define HAVE_SUPPORTED_CURVES
```

また、PC アプリのプロジェクト設定に以下の設定を追加しています。

```
#define WOLFSSL_TLS13
#define HAVE_TLS_EXTENSIONS
#define HAVE_SUPPORTED_CURVES
#define HAVE_HKDF
#define WC_RSA_PSS
#define NO_DH
```

ご注意

- ・本文書の著作権は株式会社アルファプロジェクトが保有します。
- ・本文書の内容を無断で転載することは一切禁止します。
- ・本文書に記載されているサンプルプログラムの著作権は株式会社アルファプロジェクトが保有します。
- ・本サンプルプログラムで使用されているミドルウェアおよびドライバの著作権はルネサス エレクトロニクス株式会社が保有します。
- ・本文書に記載されている内容およびサンプルプログラムについてのサポートは一切受け付けておりません。
- ・本文書の内容およびサンプルプログラムに基づき、アプリケーションを運用した結果、万一損害が発生しても、弊社では一切責任を負いませんのでご了承ください。
- ・本文書の内容については、万全を期して作成いたしました。万が一不審な点、誤りなどお気づきの点がありましたら弊社までご連絡ください。
- ・本文書の内容は、将来予告なしに変更されることがあります。

商標について

- ・RX はルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。
- ・CS+はルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。
- ・E1 エミュレータ はルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。
- ・E2 エミュレータ Lite はルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。
- ・Renesas Flash Programmer はルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。
- ・Smart Configurator はルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。

- ・wolfSSL は、wolfSSL Inc.の登録商標、商標または商品名称です。

- ・Windows®の正式名称は Microsoft®Windows®Operating System です。
Microsoft、Windows、Visual Studio は、米国 Microsoft Corporation.の米国およびその他の国における商標または登録商標です。
Windows®10、Windows®11 は、米国 Microsoft Corporation.の商品名称です。
Microsoft Visual Studio 2019 Professional は、米国 Microsoft Corporation.の商品名称です。
本文書では下記のように省略して記載している場合がございます。ご了承ください。
Windows®10 は Windows 10 もしくは Win10
Windows®11 は Windows 11 もしくは Win11

- ・その他の会社名、製品名は、各社の登録商標または商標です。



株式会社アルファプロジェクト
〒431-3114
静岡県浜松市中央区積志町8 3 4
<https://www.apnet.co.jp>
E-Mail: query@apnet.co.jp