AP-RX65N-0A (RX65N 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

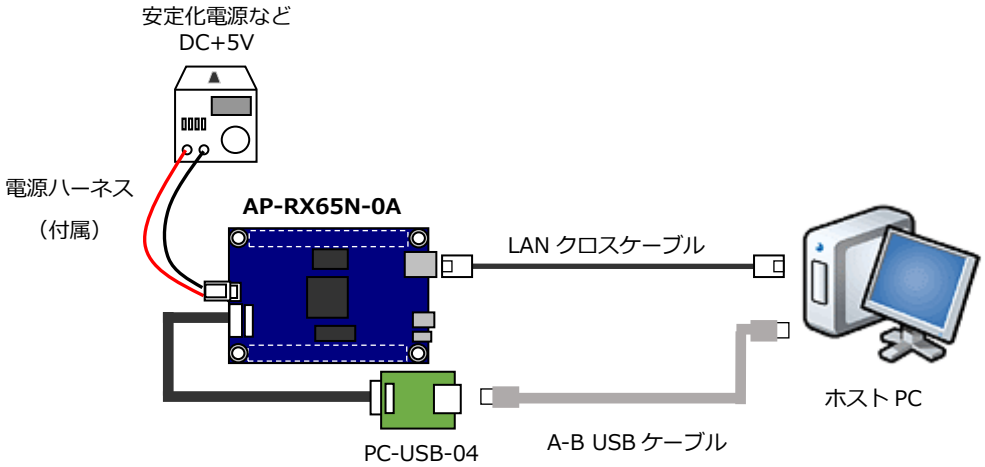
1.1　概要

本アプリケーションノートでは、弊社製 CPU ボード AP-RX65N-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。
wolfSSL サンプルプログラムは、弊社 Web サイトで公開中の AP-RX65N-0A の「USB HOST サンプルプログラム」および「USB FUNCTION サンプルプログラム」に、wolfSSL を組み込んでいます。
これらのサンプルプログラムについては、アプリケーションノート「**AN1535 USB HOST サンプルプログラム解説**」「**AN1536 USB FUNCTION サンプルプログラム解説**」を参照してください。

サンプルプログラム	動作内容
AP-RX65N-0A wolfSSL サンプルプログラム	・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「４．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-RX65N-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 1.20 を使用しています。)

ルネサス エレクトロニクス社 RX65N サンプルコード
https://www.renesas.com/jp/ja/products/microcontrollers-microprocessors/rx-32-bit-performance-efficiency-mcus/rx65n-32-bit-microcontrollers-rxv2-core-large-capacity-ram-and-enhanced-security-connectivity-and-hmi#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 5.20>
● 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>
● CAN
・資料名 RX64M, RX71M, RX65N Group CAN API Using Firmware Integration Technology 機能名称：CAN　　<R01AN2472　Rev 3.00>
● 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 4.00>
● GPIO
・資料名 RX ファミリ GPIO モジュール Firmware Integration Technology 機能名称：I/O 設定　　<R01AN1721　Rev 3.00>
● I2C
・資料名 RX ファミリ I2C バスインタフェース(RIIC)モジュール Firmware Integration Technology 機能名称：I2C バス　　<R01AN1692　Rev 2.41>
● SCI
・資料名 RX ファミリ SCI モジュール Firmware Integration Technology 機能名称：SCI　　<R01AN1815　Rev 3.00>

（※）資料をダウンロードする際にはルネサス 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。

● 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 1.16> RX ファミリ システムタイマモジュール Firmware Integration Technology 機能名称：組み込み用 TCP/IP M3S-T4-Tiny　<R01AN0431　Rev 1.00> 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 1.07> RX ファミリ 組み込み用 TCP/IP M3S-T4-Tiny モジュール Firmware Integration Technology 機能名称：組み込み用 TCP/IP M3S-T4-Tiny　<R01AN0051　Rev 2.08>
● USB HMSC
・資料名 USB Basic Host and Peripheral Driver Firmware Integration Technology 機能名称：USB　<R01AN2025　Rev 1.26> RX ファミリ USB Host Mass Storage Class Driver (HMSC) Firmware Integration Technology 機能名称：USB　<R01AN2026　Rev 1.26> RX ファミリ M3S-TFAT-Tiny メモリドライバインタフェースモジュール 機能名称：オープンソース FAT ファイルシステム　<R01AN0335　Rev 1.05> RX ファミリ オープンソース FAT ファイルシステム M3S-TFAT-Tiny モジュール Firmware Integration Technology 機能名称：オープンソース FAT ファイルシステム　<R01AN0038　Rev 3.04>
● USB PCDC
・資料名 USB Basic Host and Peripheral Driver Firmware Integration Technology 機能名称：USB　<R01AN2025　Rev 1.26> RX ファミリ USB Peripheral Communications Device Class Driver (PCDC) Firmware Integration Technology 機能名称：USB　<R01AN2030　Rev 1.26>
● 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>
● USB HMSC サンプルプログラム
・資料名 RX ファミリ USB オープンソース FAT ファイルシステム M3S-TFAT-Tiny モジュールを用いたサンプルプログラム Firmware Integration Technology 機能名称：ファイルシステム（サンプルプログラム）　<R01AN0293　Rev 1.01>
● USB PCDC サンプルプログラム
・資料名 USB ペリフェラルコミュニケーションデバイスクラスドライバ(PCDC)による USB ホストとの USB 通信を行うサンプルプログラム Firmware Integration Technology 機能名称：USB（サンプルプログラム）　<R01AN2238　Rev 1.23>

（※）資料をダウンロードする際にはルネサス 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	v2.08.01	–
Smart Configurator	v2.2.0	–
Microsoft Visual Studio Express 2017 for Windows Desktop	v15.9.3	ツールセット　Visual Studio 2017 (v141) ※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。
なお、各プロジェクトは、元にしたサンプルプログラムのプロジェクトの違いであり、
wolfSSL を利用したネットワーク通信動作は共通です。

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(usbhost)	¥Sample¥ap_rx65n_0a_usbhost_sample_cs
wolfSSL サンプルプログラム プロジェクトフォルダ(usbfunc)	¥Sample¥ap_rx65n_0a_usbfunc_sample_cs

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　v4.0.0」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。

¥ Sample	AP-RX65N-0A サンプルプログラムフォルダ
¥ ap_rx65n_0a_usbhost_sample_cs	wolfSSL サンプルプログラムフォルダ（usbhost）
¥ src	ソースフォルダ
¥ smc_gen	Smart Configurator 生成フォルダ
¥ general	general フォルダ
¥ r_bsp	BSP モジュールフォルダ
¥ r_byteq	BYTEQ モジュールフォルダ
¥ r_can_rx	CAN モジュールフォルダ
¥ r_cmt_rx	CMT モジュールフォルダ
¥ r_ether_rx	ETHERC モジュールフォルダ
¥ r_gpio_rx	GPIO モジュールフォルダ
¥ r_riic_rx	I2C モジュールフォルダ
¥ r_sci_rx	SCI モジュールフォルダ
¥ r_sys_time_rx	システムタイマモジュールフォルダ
¥ r_t4_driver_rx	T4 ドライバモジュールフォルダ
¥ r_t4_rx	T4 モジュールフォルダ
¥ r_tfat_driver_rx	M3S-TFAT-Tiny ドライバモジュールフォルダ
¥ r_tfat_rx	M3S-TFAT-Tiny モジュールフォルダ
¥ r_usb_basic	USB BASIC モジュールフォルダ
¥ r_usb_hmsc	USB HMSC モジュールフォルダ
¥ r_pincfg	端子設定ファイルフォルダ
¥ r_config	各モジュールの設定ファイルフォルダ
¥ tcp_sample	TCP/IP ソースフォルダ
¥ usb_dev	USB ドライバソースフォルダ
¥ wolfssl	wolfSSL サンプルプログラムフォルダ
¥ lib	wolfSSL ライブラリフォルダ
¥ common	wolfSSL ライブラリ生成ヘッダファイルフォルダ
¥ wolfssl	wolfSSL ヘッダファイルフォルダ
¥ DefaultBuild	ワークフォルダ
¥ .settings	Smart Configurator 設定フォルダ

<次のページへ>

¥ ap_rx65n_0a_usbfunc_sample_cs	wolfSSL サンプルプログラムフォルダ（usbfunc）
¥ src	ソースフォルダ
¥ smc_gen	Smart Configurator 生成フォルダ
¥ general	general フォルダ
¥ r_bsp	BSP モジュールフォルダ
¥ r_byteq	BYTEQ モジュールフォルダ
¥ r_can_rx	CAN モジュールフォルダ
¥ r_cmt_rx	CMT モジュールフォルダ
¥ r_ether_rx	ETHERC モジュールフォルダ
¥ r_gpio_rx	GPIO モジュールフォルダ
¥ r_riic_rx	I2C モジュールフォルダ
¥ r_sci_rx	SCI モジュールフォルダ
¥ r_sys_time_rx	システムタイマモジュールフォルダ
¥ r_t4_driver_rx	T4 ドライバモジュールフォルダ
¥ r_t4_rx	T4 モジュールフォルダ
¥ r_usb_basic	USB BASIC モジュールフォルダ
¥ r_usb_pcdc	USB PCDC モジュールフォルダ
¥ r_pincfg	端子設定ファイルフォルダ
¥ r_config	各モジュールの設定ファイルフォルダ
¥ tcp_sample	TCP/IP ソースフォルダ
¥ usb_dev	USB ドライバソースフォルダ
¥ inc	USB ドライバヘッダファイルフォルダ
¥ wolfssl	wolfSSL サンプルプログラムフォルダ
¥ lib	wolfSSL ライブラリフォルダ
¥ common	wolfSSL ライブラリ生成ヘッダファイルフォルダ
¥ wolfssl	wolfSSL ヘッダファイルフォルダ
¥ DefaultBuild	ワークフォルダ
¥ .settings	Smart Configurator 設定フォルダ
¥ PC_Application	動作検証用 PC アプリフォルダ
¥ certs	セキュリティ証明書フォルダ （動作検証用 PC アプリの実行に必要です。 削除しないでください。）

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.exe	･･･	クライアント動作 PC アプリケーション
client.bat	･･･	client.exe 起動用バッチファイル
echoserver.exe	･･･	エコーサーバ動作 PC アプリケーション
echoserver.bat	･･･	echoserver.exe 起動用バッチファイル

＜¥Sample¥ap_rx65n_0a_usbhost_sample_cs フォルダ内＞

ap_rx65n_0a_usbhost_sample_cs.mtpj	･･･	CS+用プロジェクトファイル
ap_rx65n_0a_usbhost_sample_cs.rcpe	･･･	e2studio 用プロジェクトファイル
ap_rx65n_0a_usbhost_sample_cs.scfg	･･･	Smart Configurator 用ファイル （CS+上から Smart Configurator を起動できます。）
AP-RX65N-0A_usbhost_sample_cs_V2.1.bdf	･･･	Board Description File （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_rx65n_0a_usbhost_sample_cs¥DefaultBuild フォルダ内＞

ap_rx65n_0a_usbhost_sample_cs.abs	･･･	elf 形式オブジェクトファイル
ap_rx65n_0a_usbhost_sample_cs.mot	･･･	モトローラ S フォーマット形式ファイル
ap_rx65n_0a_usbhost_sample_cs.map	･･･	マップファイル

＜¥Sample¥ap_rx65n_0a_usbfunc_sample_cs フォルダ内＞

ap_rx65n_0a_usbfunc_sample_cs.mtpj	･･･	CS+用プロジェクトファイル
ap_rx65n_0a_usbhost_sample_cs.rcpe	･･･	e2studio 用プロジェクトファイル
ap_rx65n_0a_usbfunc_sample_cs.scfg	･･･	Smart Configurator 用ファイル （CS+上から Smart Configurator を起動できます。）
AP-RX65N-0A_usbfunc_sample_cs_V2.1.bdf	･･･	Board Description File （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_rx65n_0a_usbfunc_sample_cs¥DefaultBuild フォルダ内>

ap_rx65n_0a_usbfunc_sample_cs.abs	・・・	elf 形式オブジェクトファイル
ap_rx65n_0a_usbfunc_sample_cs.mot	・・・	モトローラ S フォーマット形式ファイル
ap_rx65n_0a_usbfunc_sample_cs.map	・・・	マップファイル

<¥Sample¥ap_rx65n_0a_usbhost_sample_cs¥src フォルダ内>
<¥Sample¥ap_rx65n_0a_usbfunc_sample_cs¥src フォルダ内>　（共通）

smc_gen	・・・	Smart Configurator により生成されたモジュールフォルダ
ap_rx65n_0a.c	・・・	メイン処理ソースファイル
buffer_rw.c	・・・	バッファ処理ソースファイル
can_dev.c	・・・	CAN ドライバソースファイル
cmd_proc_app.c	・・・	コマンド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
echoback_app.c	・・・	エコーバック処理ソースファイル
eeprom.c	・・・	EEPROM 処理ソースファイル
ether_app.c	・・・	Ethernet アプリケーションソースファイル
i2c_dev.c	・・・	I2C ドライバソースファイル
ioport.c	・・・	方形波出力処理ソースファイル
sci_dev.c	・・・	SCI ドライバソースファイル
sdram_dev.c	・・・	SDRAM ドライバソースファイル
buffer_rw.h	・・・	バッファ処理ヘッダファイル
can_dev.h	・・・	CAN ドライバヘッダファイル
cmd_proc_app.h	・・・	コマンド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
eeprom.h	・・・	EEPROM 処理ヘッダファイル
ether_app.h	・・・	Ethernet アプリケーションヘッダファイル
i2c_dev.h	・・・	I2C ドライバヘッダファイル
ioport.h	・・・	方形波出力処理ヘッダファイル
sci_dev.h	・・・	SCI ドライバヘッダファイル

<¥Sample¥ap_rx65n_0a_usbhost_sample_cs¥src¥tcp_sample フォルダ内>
<¥Sample¥ap_rx65n_0a_usbfunc_sample_cs¥src¥tcp_sample フォルダ内>　（共通）

config_tcpudp.c	・・・	TCP/IP 定義ファイル
echo_srv.c	・・・	TCP/IP 処理ソースファイル
echo_srv_sample.h	・・・	TCP/IP 処理ヘッダファイル
ether_dev.h	・・・	Ethernet ドライバヘッダファイル

<¥Sample¥ap_rx65n_0a_usbhost_sample_cs¥src¥usb_dev フォルダ内>

r_data_file.c	・・・	USB Host MSC 保存データファイル
r_usb_hmsc_apl.c	・・・	USB Host MSC 処理ソースファイル
r_data_file.h	・・・	USB Host MSC 保存データヘッダファイル
r_usb_hmsc_apl.h	・・・	USB Host MSC 処理ヘッダファイル
usbh_dev.h	・・・	USB Host ドライバヘッダファイル

<¥Sample¥ap_rx65n_0a_usbfunc_sample_cs¥src¥usb_dev フォルダ内>

r_usb_pcdc_descriptor.c	・・・	USB Func ディスクリプタ情報ファイル
r_usb_pcdc_echo_apl.c	・・・	USB Func 仮想 COM 処理ソースファイル
usbf_dev.h	・・・	USB Func ドライバヘッダファイル

<¥Sample¥ap_rx65n_0a_usbfunc_sample_cs¥src¥usb_dev¥inc フォルダ内>

r_usb_pcdc_apl.h	・・・	USB Func 仮想 COM エコーバック処理ヘッダファイル
r_usb_pcdc_apl_config.h	・・・	USB Func 設定ヘッダファイル

<¥Sample¥ap_rx65n_0a_usbhost_sample_cs¥src¥wolfssl フォルダ内>
<¥Sample¥ap_rx65n_0a_usbfunc_sample_cs¥src¥wolfssl フォルダ内>　（共通）

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.c	・・・	wolfSSL 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
wolfssl.h	・・・	wolfSSL Server/Client 選択ヘッダファイル

<¥Sample¥ap_rx65n_0a_usbhost_sample_cs¥src¥wolfssl¥lib フォルダ内>
<¥Sample¥ap_rx65n_0a_usbfunc_sample_cs¥src¥wolfssl¥lib フォルダ内>　（共通）

wolfssl_lib.lib	・・・	wolfSSL ライブラリファイル（v4.0.0）

3.　プログラム作成方法

本章では、AP-RX65N-0A 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-RX65N-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-RX65N-0A サンプルプログラムのうち、どちらか一方のプロジェクト
（弊社 Web サイト AP-RX65N-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-RX65N-0A サンプルプログラム(CS+版)』に含まれています。）
・ap_rx65n_0a_usbhost_sample_cs
・ap_rx65n_0a_usbfunc_sample_cs

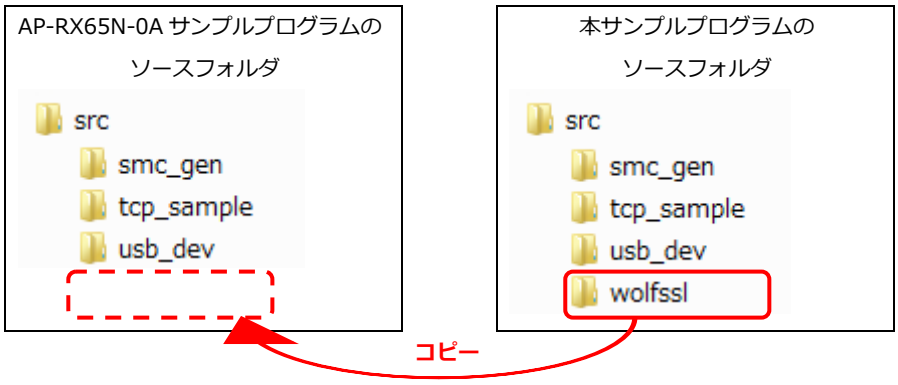
②　本サンプルプログラムのうち、どちらか一方のプロジェクト
・ap_rx65n_0a_usbhost_sample_cs
・ap_rx65n_0a_usbfunc_sample_cs

①、②ともに、どちらのプロジェクトを選択しても、問題ありません。
本章では、「ap_rx65n_0a_usbfunc_sample_cs」を元に説明しますので、異なるプロジェクトを使用する場合は、
読み替えて行ってください。

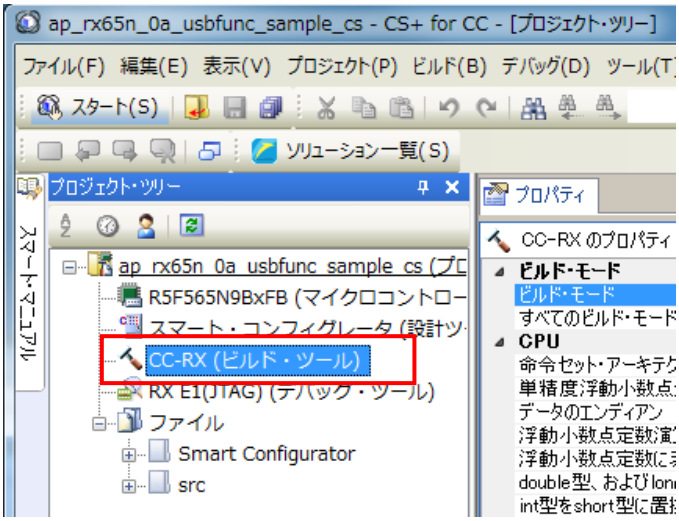
3.2　プロジェクト設定

「3.1　プロジェクトの準備」で用意した、元とする AP-RX65N-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。



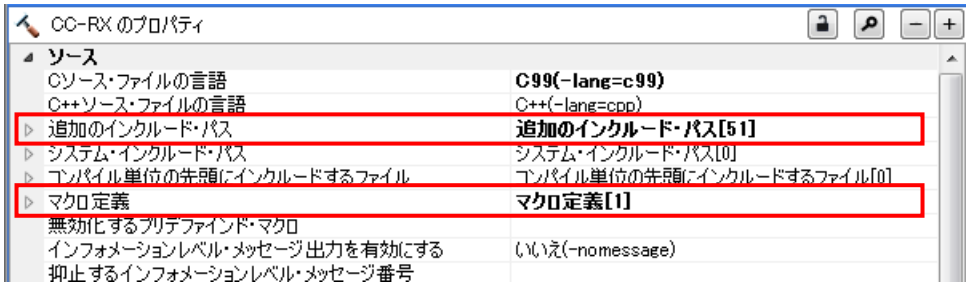
② CS+上で、「プロジェクト・ツリー」から、「CC-RX（ビルド・ツール）」のプロパティを開きます。



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。

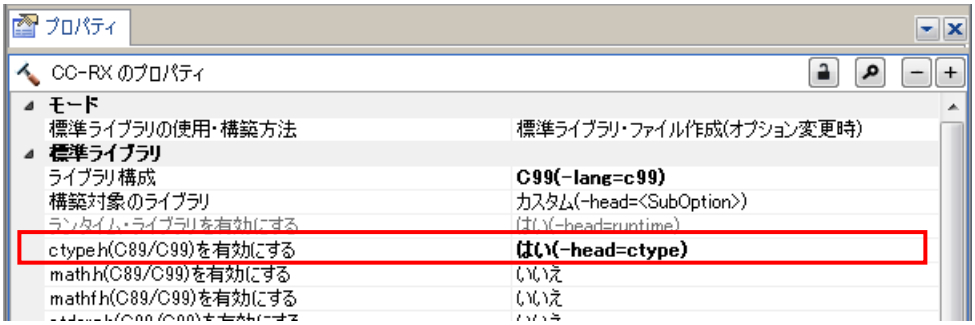
また、「マクロ定義」に「**WOLFSSL_USER_SETTINGS**」を追加します。



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　プロジェクトに、①でコピーしたフォルダ「wolfssl」内のソースファイルを追加します。
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」にファイルやフォルダ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-RX65N-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-RX65N-0A サンプルプログラムのソースを変更します。

① ソースファイル「ap_rx65n_0a.c」を編集します。
各モジュールの初期化終了後、メインループの while 文より前に、「wolfSSL_main();」を追加します。

<ソース例>

```
/* Initialize */
SdramInit();
CmtInit();
CanInit();
SciInit();
EthernetAppInit();
UsbfInit();
SqwPortInit();

wolfSSL_main();          ← 追加
```

このとき、不要な処理は削除、またはコメントアウトします。
wolfSSL を利用したネットワーク通信に必要な処理を下に記載しますので、それ以外の処理は削除してかまいません。

- CmtInit();
- SciInit();
- EthernetAppInit();
- wolfSSL_main();　　←追加した処理

② 「¥src¥smc_gen¥r_config¥**r_bsp_config.h**」を変更します。

変更する項目	設定値	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	#pragma stack size su = 0x2000
ヒープサイズ	0xa000	#define BSP_CFG_HEAP_BYTES
user charget()関数の使用	Use user … (1)	#define BSP_CFG_USER_CHARGET_ENABLED
user charget()関数の関数名	my_sw_charget_function	#define BSP_CFG_USER_CHARGET_FUNCTION
user charput()関数の使用	Use user … (1)	#define BSP_CFG_USER_CHARPUT_ENABLED
user charput()関数の関数名	my_sw_charput_function	#define BSP_CFG_USER_CHARPUT_FUNCTION

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。

- シリアル通信

SCI6 でメッセージの送受信を行います。
シリアルの設定は、38400bps、ビット長 8、パリティなし、ストップビット 1、フロー制御なしです。
動作確認は、ホスト PC 上のターミナルソフト（ハイパーターミナル等）を使用してください。

- 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。

- タイマ割り込み

LD2（緑の LED）を通常時は 1000msec 周期、MAC アドレスの読み出しエラー発生時は 500msec 周期で点滅させます。（CMT 割り込み使用）

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、以下の手順に従い行ってください。
なお、事前にプログラムを「サーバ」設定でビルドする必要があります。「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。
ターミナルソフトには以下のように表示されます。

```
----------------------------
 wolfSSL version 4.0.0
----------------------------
Start TLS Server
```

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「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	50000

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、CPU ボードはクライアントからの受信を待機します。
ターミナルソフトには以下のように表示されます。

```
----------------------------
 wolfSSL version 4.0.0
----------------------------
Start TLS Server
Client IP add : 192.168.1.202 , port : 50002
Connection completed.
Protocol version : TLSv1.3
SSL cipher suite is TLS13-AES128-GCM-SHA256
----------------------------
```

※「port:」の数値は
環境により異なります。

⑥　接続完了後、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をエコーバックします。受信文字列はターミナルソフトに表示されます。

```
（略）
---------------------------
Received: hello wolfssl!
```

⑦　CPU ボードは受信待機状態に戻りますが、ホスト PC の client.exe の動作は終了です。
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、以下の手順に従い行ってください。
なお、事前にプログラムを「クライアント」設定でビルドする必要があります。
「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「echoserver.bat」を実行します。
「echoserver.bat」の実行により「echoserver.exe」が起動し、ホスト PC がサーバとして動作します。
※「echoserver.bat」「echoserver.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、ターミナルソフトに以下のように表示されます。

```
---------------------------
 wolfSSL version 4.0.0
---------------------------
wolfSSL client sample
  c <IP addr> <Port>: client
$
```

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」(echoserver.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。

```
$ c 192.168.1.202 11111
```

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、ターミナルソフトには以下のように表示されます。

```
---------------------------
 wolfSSL version 4.0.0
---------------------------
wolfSSL client sample
   c <IP addr> <Port>: client
$ c 192.168.1.202 11111
Start TLS Client (192.168.1.202 , 11111)
Connection completed.
Protocol version : TLSv1.3
SSL cipher suite is TLS13-AES128-GCM-SHA256
---------------------------
```

※「$ 」の行は⑤での入力箇所

⑦　CPU ボードから任意の文字列を送信します。
ターミナルソフトから任意の文字列を入力し、最後に改行コードを送信してください。

⑧　CPU ボードから送信した文字列は、ホスト PC（echoserver.exe）でエコーバックされます。
エコーバックされた文字列は、ターミナルソフトに表示されます。

```
（略）
---------------------------
ABC <改行>
Received: ABC
```

※「ABC」を入力した例

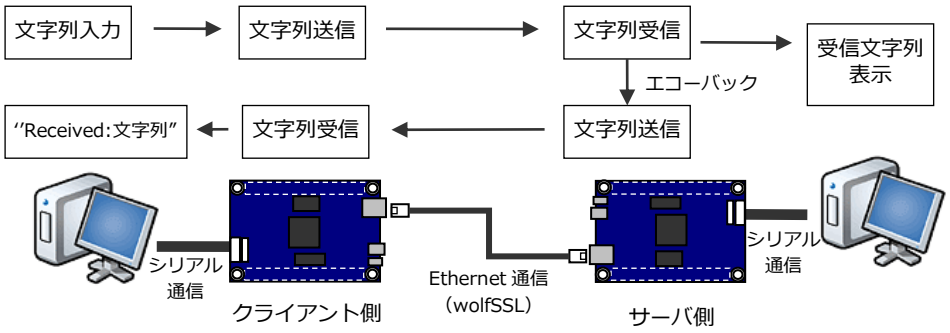
⑨　送受信の確認は、何度でも行うことができます。
以上で、クライアント動作の確認は終了です。
SSL 通信の確認に関しては、「4.1.5　通信プロトコルの確認方法」を参照してください。

CPU ボード 2 枚を用いた動作確認方法

ホスト PC（動作確認用 PC アプリ）と CPU ボードとの通信だけでなく、CPU ボード同士の通信を行うことができます。
CPU ボード 2 枚を用いて動作確認をする場合、以下の点に注意をしてください。

- 設定・接続
  - ・CPU ボードのうち、一方にサーバ用、もう一方にクライアント用のプログラムを書き込む必要があります。
  - ・サーバ用プログラム、クライアント用プログラムは、デフォルトでは同じ IP アドレスを設定しています。
    事前に通信が可能な IP アドレスに変更して、ビルドしてください。

- 動作確認
  - ・クライアント側 CPU ボードの電源を投入する前に、サーバ側 CPU ボードを起動してください。
  - ・動作内容は「4.1.2　サンプルプログラム　サーバ動作」「4.1.3　サンプルプログラム　クライアント動作」と同様です。各々の説明を参照してください。
    ただし、接続する IP アドレス、ポート番号などは、接続先の CPU ボードの設定を指定する必要があります。
  - ・サーバ側 CPU ボードは、一度文字列を受信した後も受信待機を継続しますので、クライアント側 CPU ボードから送信を行うたびに、ターミナルソフトに受信文字列を表示します。



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

- サーバ動作 / クライアント動作 選択

本サンプルプログラムは、デフォルトでは「サーバ動作」をします。
「クライアント動作」をさせたい場合、以下の定義の変更が必要です。
「サーバ動作」の場合は「USE_WOLFSSL_SERVER」を「1」に、
「クライアント動作」の場合は「USE_WOLFSSL_CLIENT」を「1」に設定し、必ずもう一方は「0」にしてください。

<¥wolfssl¥wolfssl.h>

```
14  /* Server か Client を選択 */
15  /* 使用する通信の定義は「1」、使用しないものは「0」 */
16
17  #define USE_WOLFSSL_SERVER          (1)
18  #define USE_WOLFSSL_CLIENT          (0)
19
```

※　本サンプルプログラムでは、上記の定義により、サーバ / クライアントの動作を選択しています。
「¥src¥ether_app.h」内の定義
「#define ETHERNET_TYPE　　(EtherType_TcpServer)」(32 行目)
の設定値は反映されませんので、ご注意ください。

● 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。

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	50000
MAC アドレス	00-0C-7B-47-XX-XX ※　XX-XX の値は製品ごとに異なります。

上記設定のうち、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、サンプルプログラム内の「¥src¥ether_app.h」で定義しています。
各設定の定義は以下の通りです。

	CPU ボードの設定
IP アドレス	ETHERNET0_MY_IPADDR
サブネットマスク	ETHERNET0_MY_SUBNET
ゲートウェイ	ETHERNET0_MY_GATEWAY
ポート番号	ETHERNET0_MY_PORT

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（CH0）	格納値
先頭アドレス　+　0x00	0x00
+　0x01	0x0C
+　0x02	0x7B
+　0x03	0x47
+　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

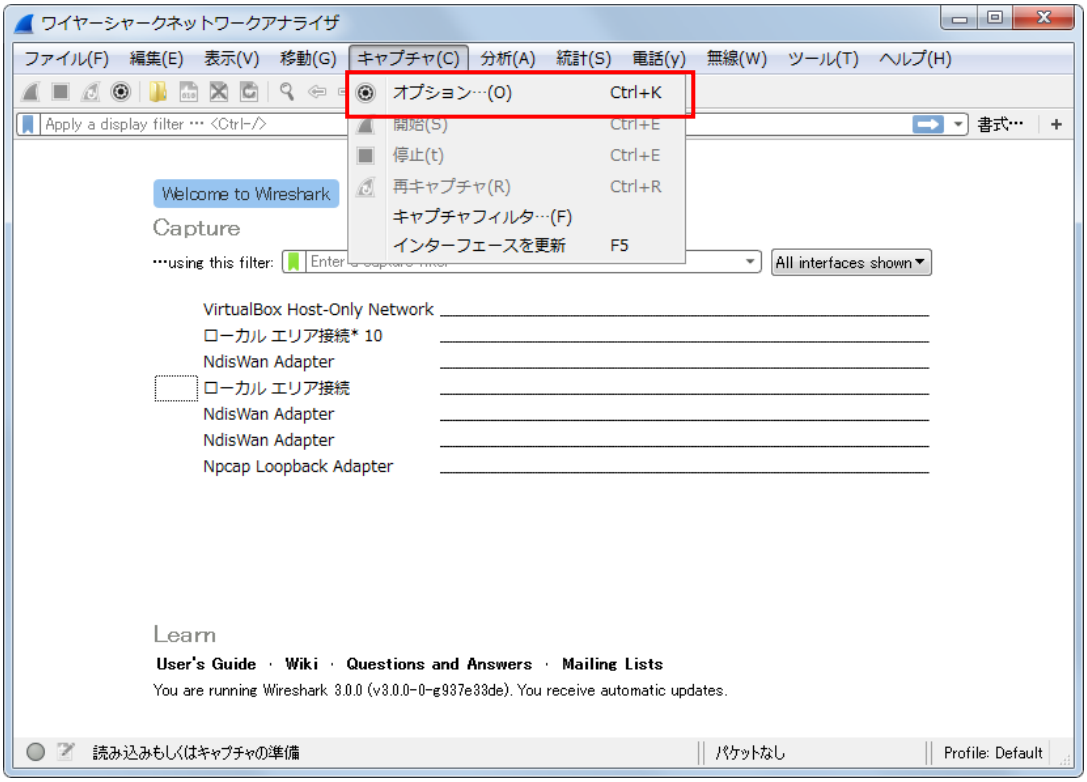
通信プロトコルの確認には、ネットワークアナライザを使用します。
ここでは、フリーソフトウェア「Wireshark」を使用した確認方法を説明します。
「Wireshark」は下記の Web サイトからダウンロードできます。

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

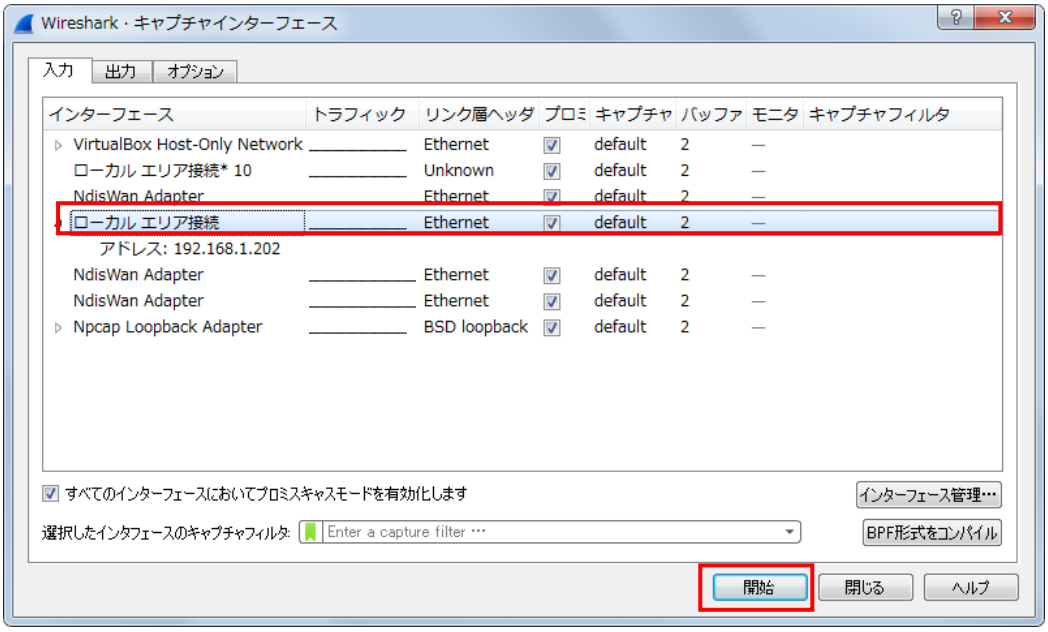
- 確認方法

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。

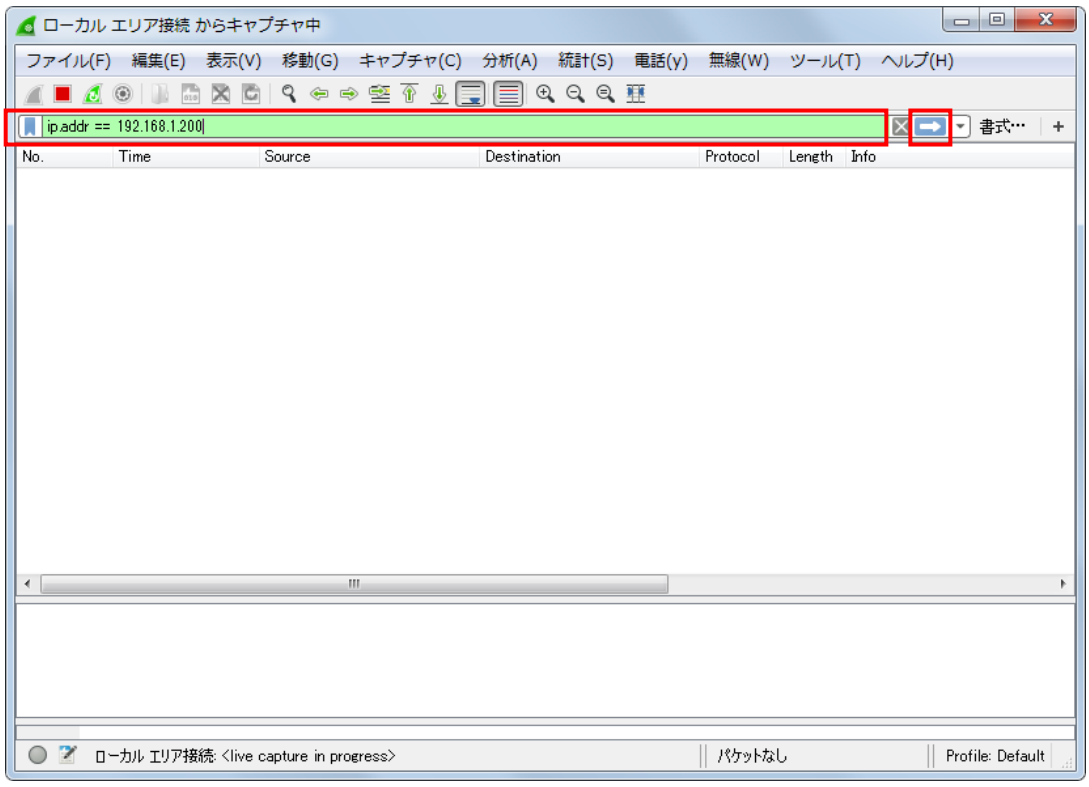
①　Wireshark を起動し、メニューの「キャプチャ」-「オプション」を選択します。



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。

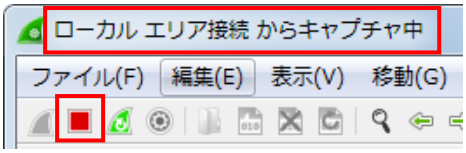


③ CPU ボードの IP アドレスを含むパケットのみキャプチャするように、フィルタを設定します。
フィルタ入力欄に「**ip.addr == 192.168.1.200**」を入力します。
入力後は Enter キーを押すか、Apply ボタンを押して確定してください。



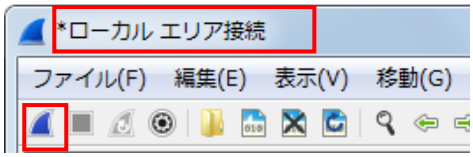
④　接続を選択すると自動でキャプチャが開始します。
手動でキャプチャを開始する場合、開始ボタンを押してキャプチャを開始してください。
キャプチャを開始した後、サンプルプログラムを動作させます。
サンプルプログラムの動作方法に関しては、「4.1.2　サンプルプログラム　サーバ動作」または「4.1.3　サンプルプログラム　クライアント動作」を参照してください。

<キャプチャ中の表示>



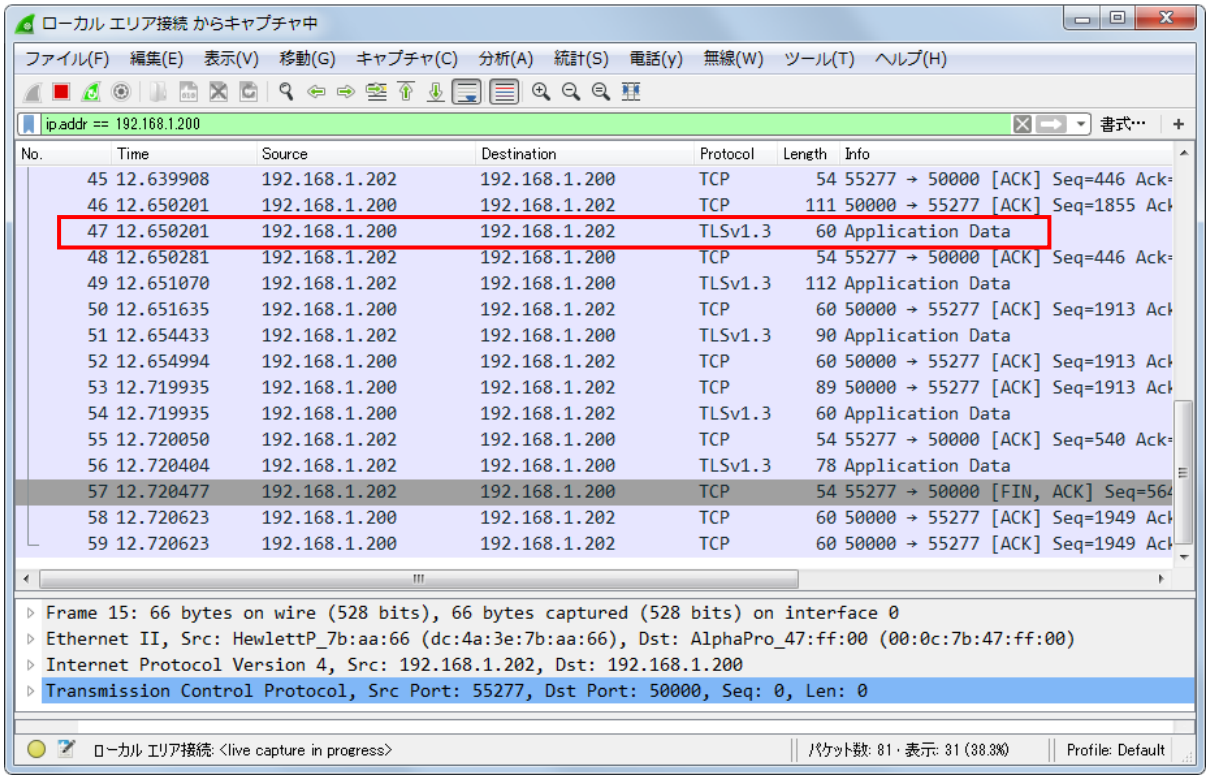
キャプチャ停止
ボタン

<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

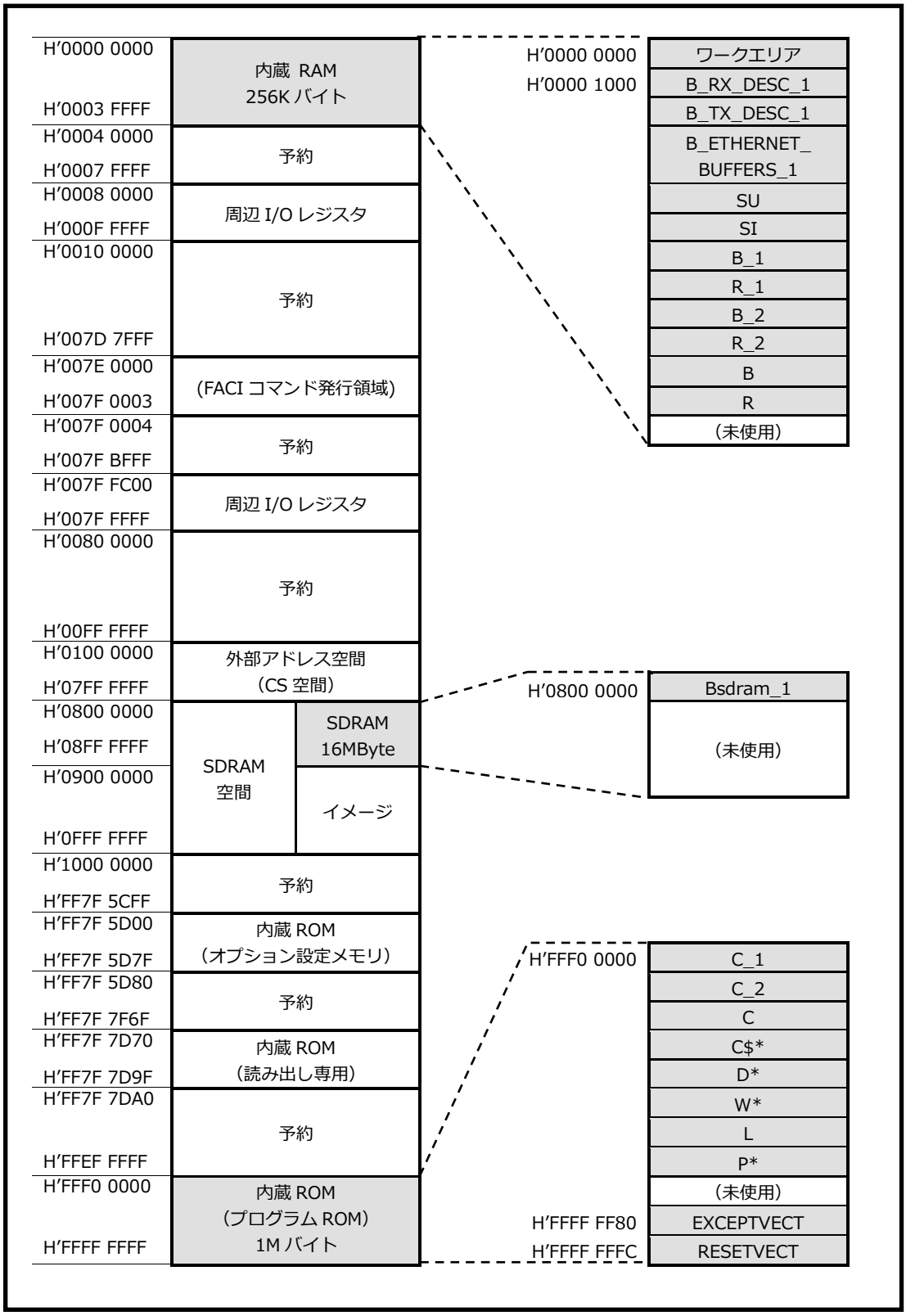


Fig 4.2-1　サンプルプログラム メモリマップ（host / func 共通）

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-RX65N-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。

※　本章では、「ap_rx65n_0a_usbfunc_sample_cs」を例に設定値を説明します。
「ap_rx65n_0a_usbhost_sample_cs」を使用する場合は、ファイル名を読み替えて行ってください。

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_rx65n_0a_usbfunc_sample_cs¥DefaultBuild
モトローラファイル名	2-3	ap_rx65n_0a_usbfunc_sample_cs ¥DefaultBuild¥ap_rx65n_0a_usbfunc_sample_cs.mot
アブソリュートファイル名	2-4	ap_rx65n_0a_usbfunc_sample_cs ¥DefaultBuild¥ap_rx65n_0a_usbfunc_sample_cs.abs
マップファイル	2-5	ap_rx65n_0a_usbfunc_sample_cs ¥DefaultBuild¥ap_rx65n_0a_usbfunc_sample_cs.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：CN5
Flash に書き込むファイル	3-3	ap_rx65n_0a_usbfunc_sample_cs ¥DefaultBuild¥ap_rx65n_0a_usbfunc_sample_cs.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：CN7 (USB microB)
ツール選択	3-6	[COM]　　詳細：[RX USB Boot(CDC)]
Flash に書き込むファイル	3-7	ap_rx65n_0a_usbfunc_sample_cs ¥DefaultBuild¥ap_rx65n_0a_usbfunc_sample_cs.mot
ボード設定（動作）	3-8	Fig 5-3 を参照

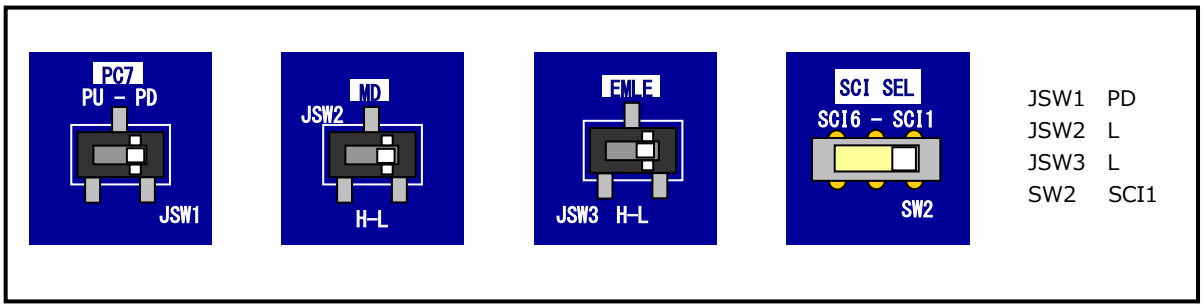


Fig 5-1　Flash 書き込み(シリアルポート使用)時のボード設定

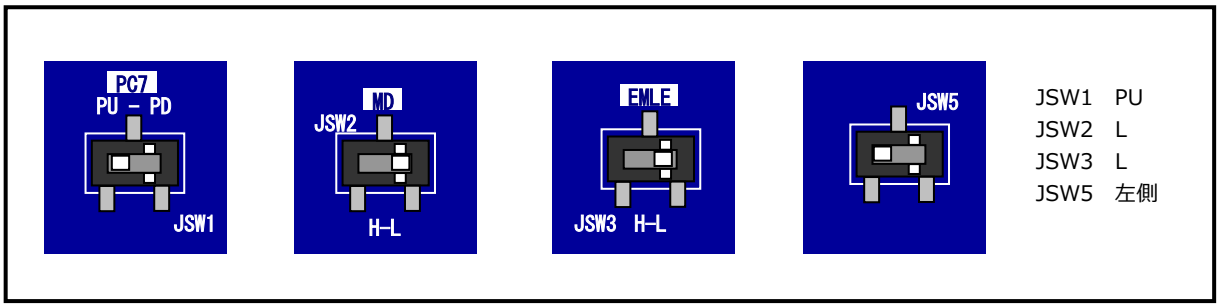


Fig 5-2　Flash 書き込み(USB ブートモード)時のボード設定

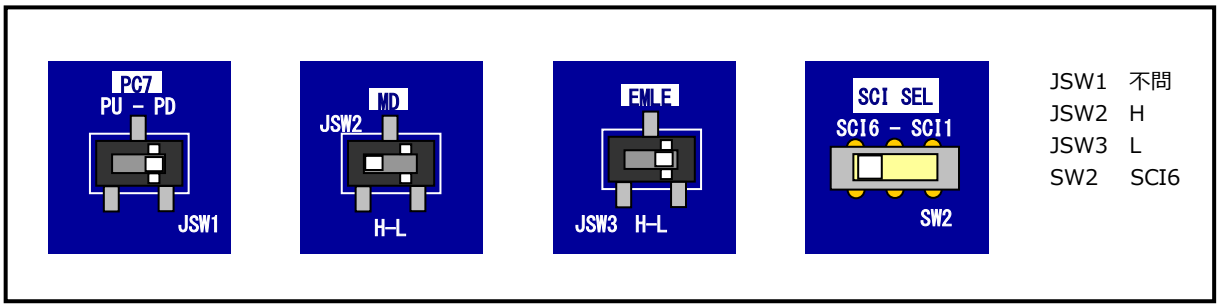


Fig 5-3　サンプルプログラム動作時のボード設定

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)

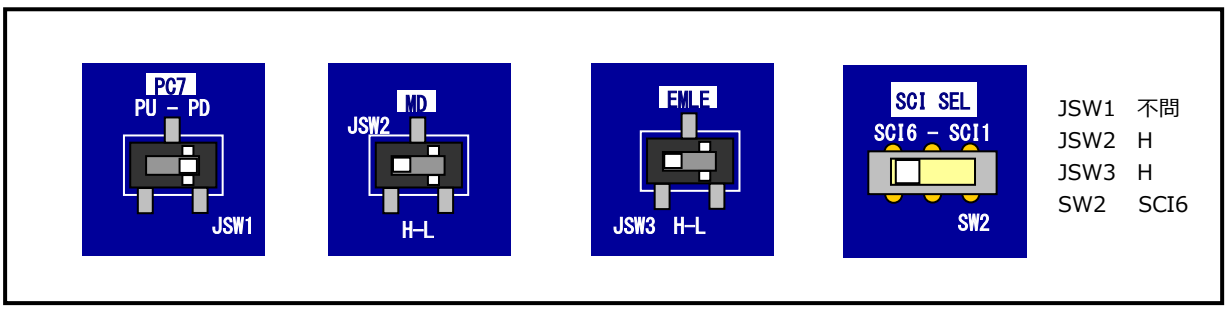


Fig 5-4　E1 エミュレータ/E2 エミュレータ Lite デバッグ時のボード設定

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。

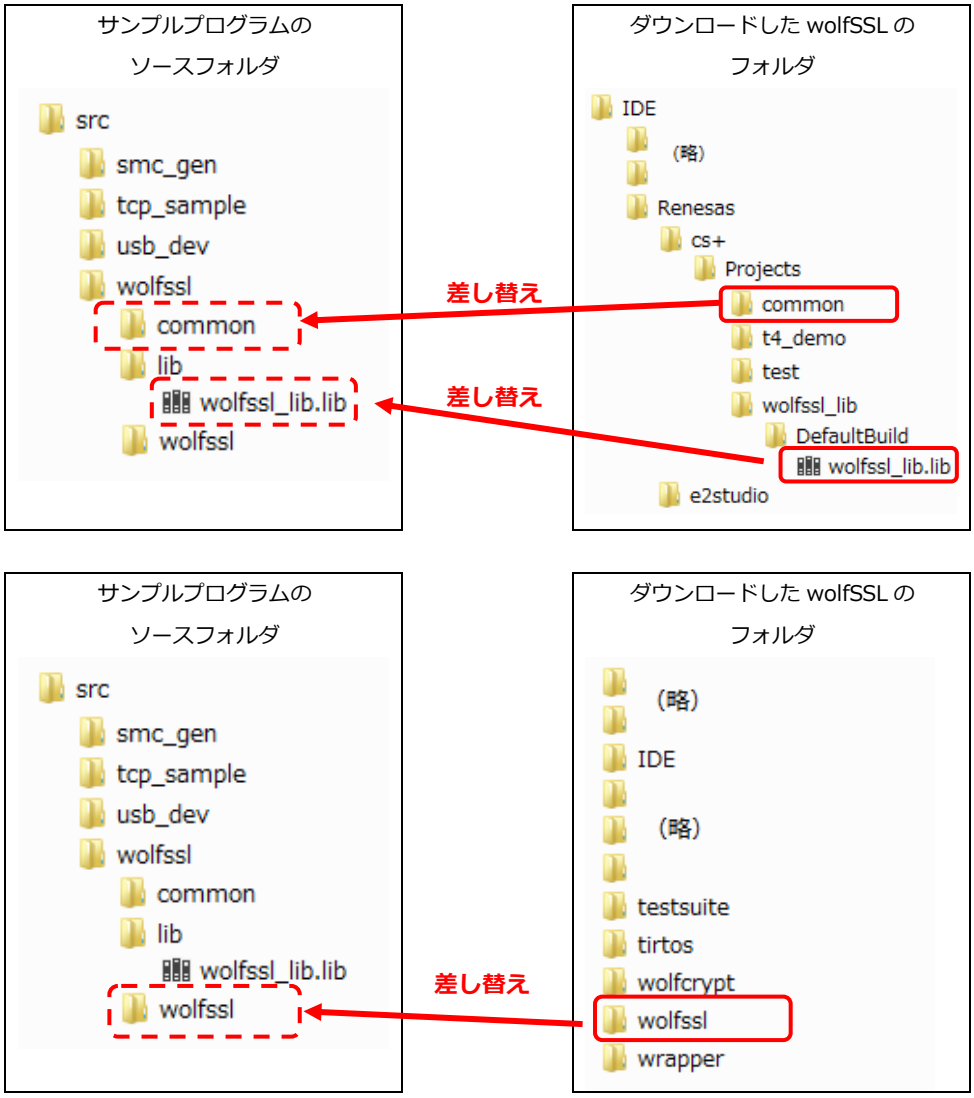
wolfSSL 社 組込み SSL ライブラリページ： https://www.wolfssl.jp/products/wolfssl/
（wolfSSL 日本語サイト：https://www.wolfssl.jp/）

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「RX71M」に設定されています。RX71M 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 Express 2017 for Windows Desktop」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-4.0.0.zip」をダウンロードした場合、「¥wolfssl-4.0.0」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-4.0.0.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。

<client.exe 用バッチファイル>

```
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。

<echoserver.exe 用バッチファイル>

```
echoserver – 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、PC アプリの両方の「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
```

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が、万一ご不審な点、誤りなどお気付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。
・Microsoft Visual Studio Express 2017 for Windows Desktop は、米国 Microsoft Corporation.の商品名称です。
・Windows®10、Windows®11 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。

ALPHA PROJECT
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８３４
https://www.apnet.co.jp
E-Mail: query@apnet.co.jp