AP-RA6M-0A
サンプルプログラム解説

3.1 版　2024 年 02 月 26 日

1. 概要

1.1 概要

本アプリケーションノートでは、AP-RA6M-0A(RA6M3 CPU)を用いて、Flexible Software Package を使用したサンプルプログラムについて解説します。
AP-RA6M-0A には、下記のサンプルプログラムが付属しています。
本サンプルプログラムで使用する主な機能を以下に記します。

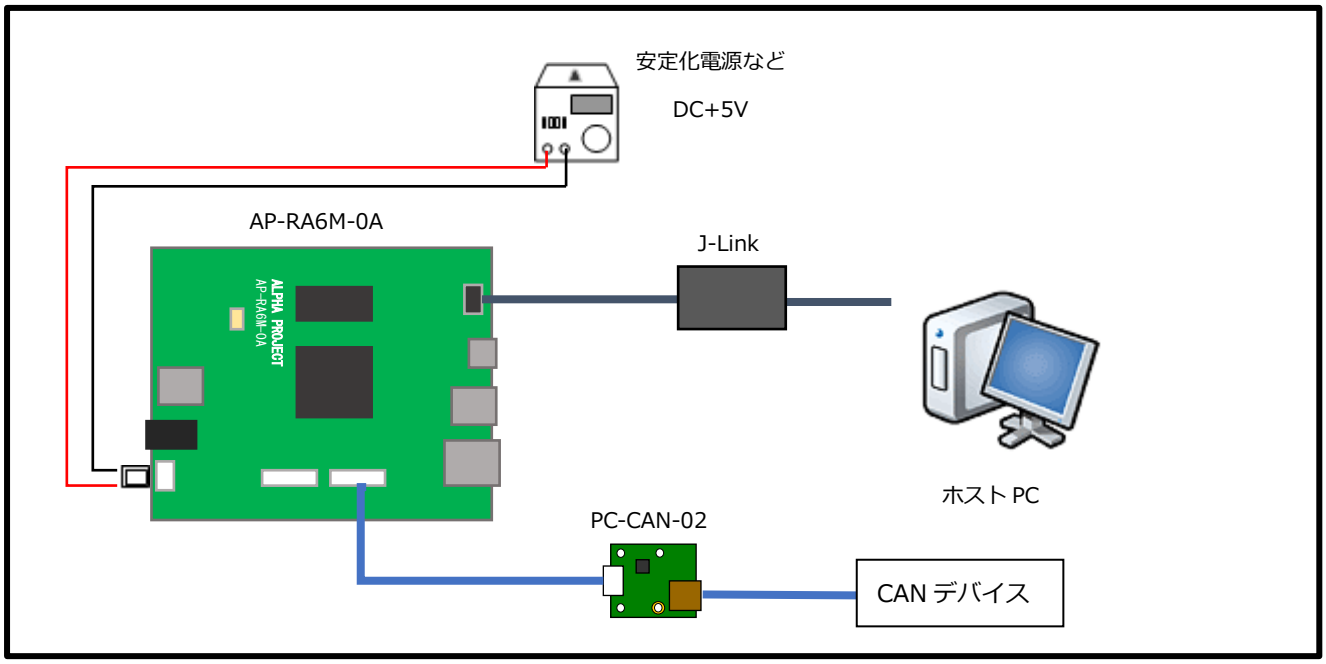
デバイス	機能
AP-RA6M-0A	・CAN 通信 ・ネットワーク通信 ・QSPI FlashROM 読み書き ・UART 通信 ・SD カード読み書き ・USB ホスト メモリ読み書き ・USB ファンクション 仮想 COM 通信

1.2　接続概要

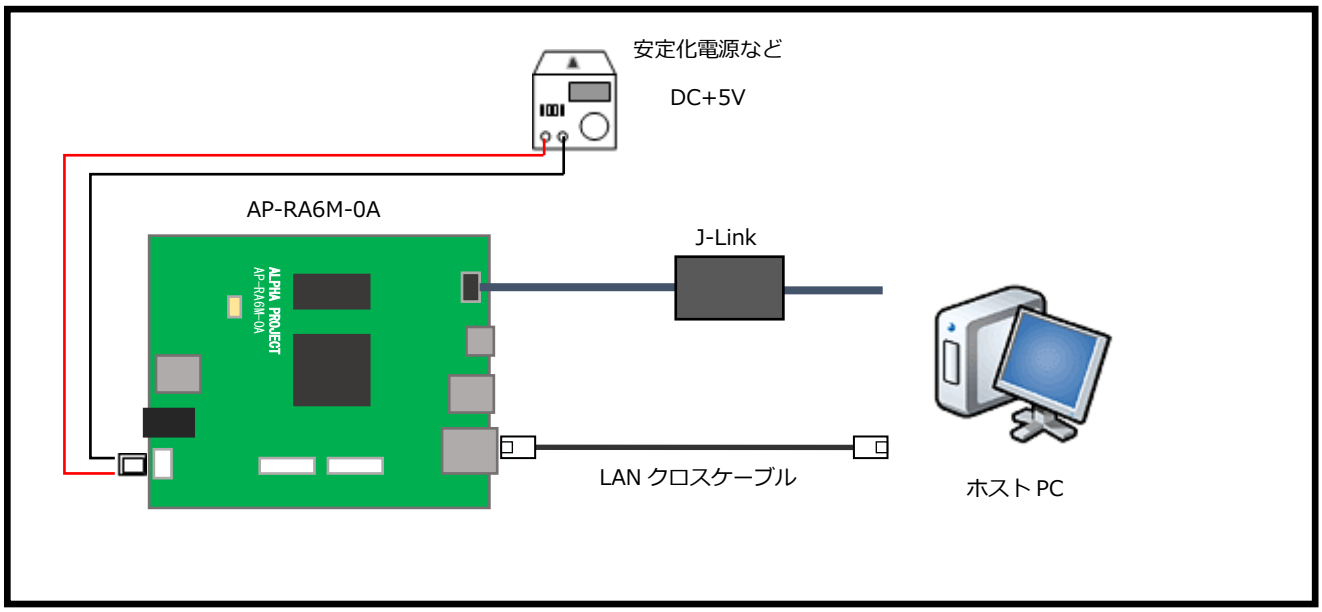
本サンプルプログラムの動作を確認する上で必要な CPU ボードの接続例を以下に示します。
詳細な接続に関しては後述の「3.2 動作説明」を参照してください。

※AP-RA6M-0A と J-Link を直接接続することはできません。
AP-RA6M-0A 側（ハーフピッチコネクタ）と J-Link 側（フルピッチコネクタ）を接続するための変換アダプタが必要となります。
変換アダプタについては、J-Link 取扱店へご確認ください。

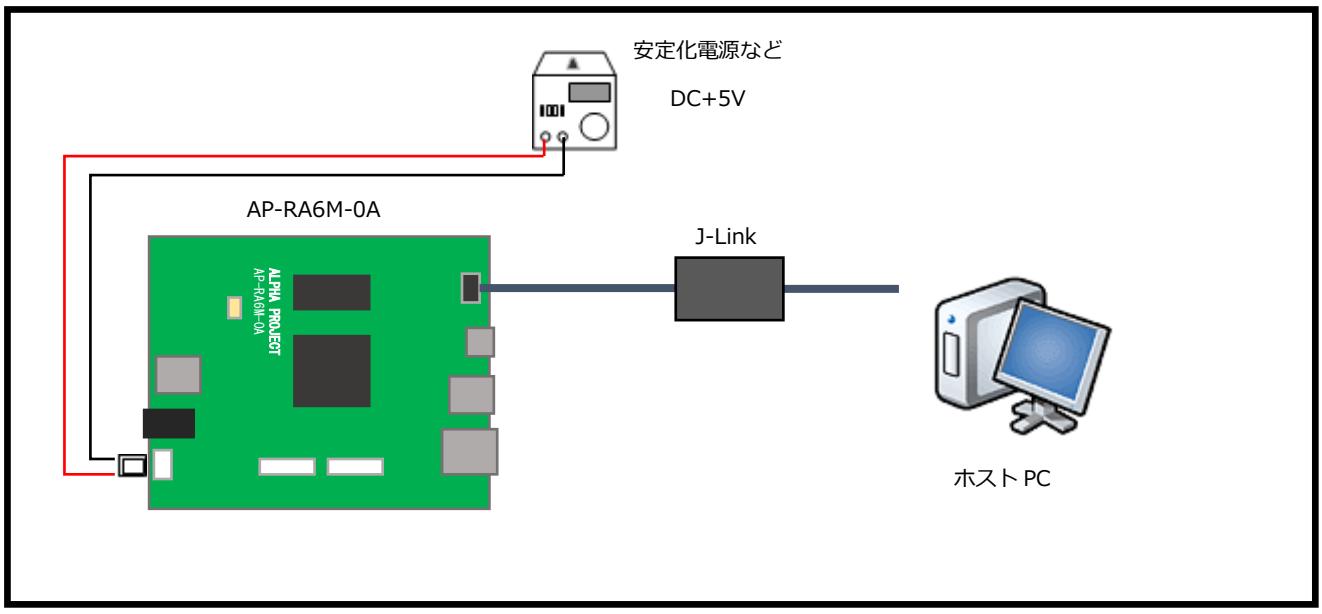
・CAN サンプルプログラム動作時の接続例



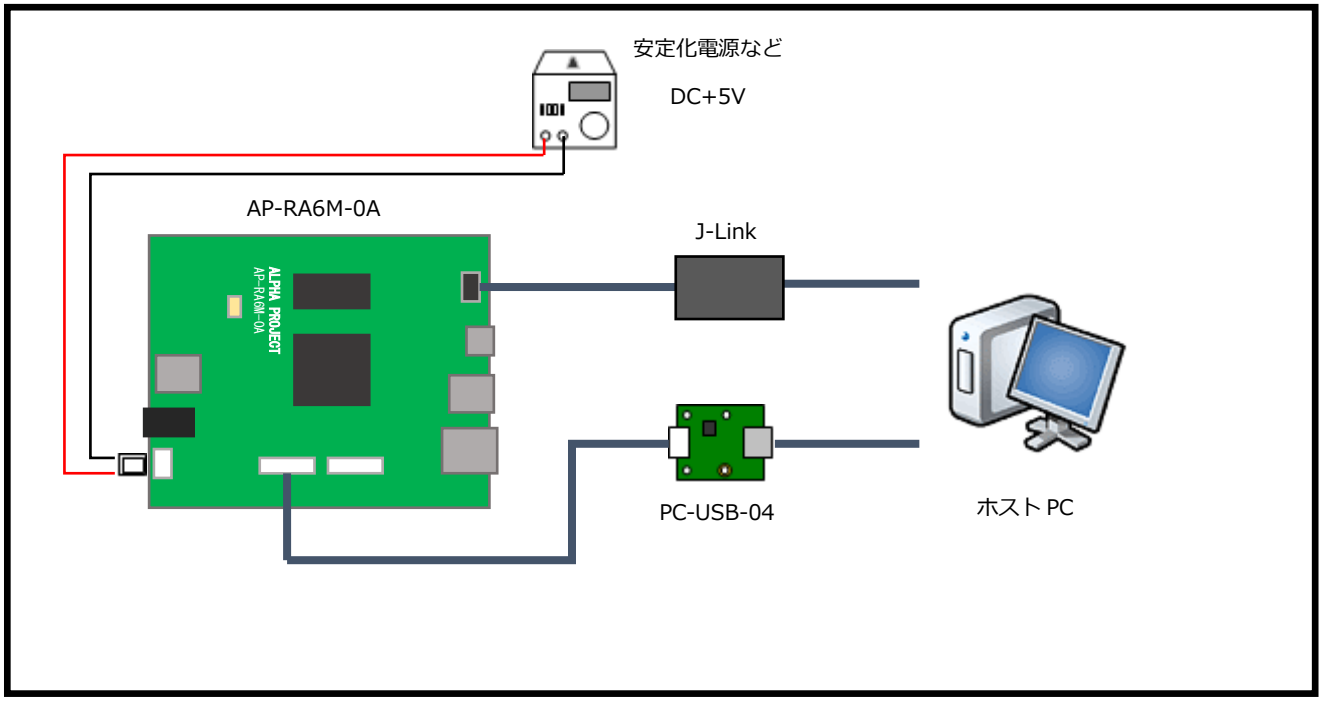
・Ethernet サンプルプログラム動作時の接続例



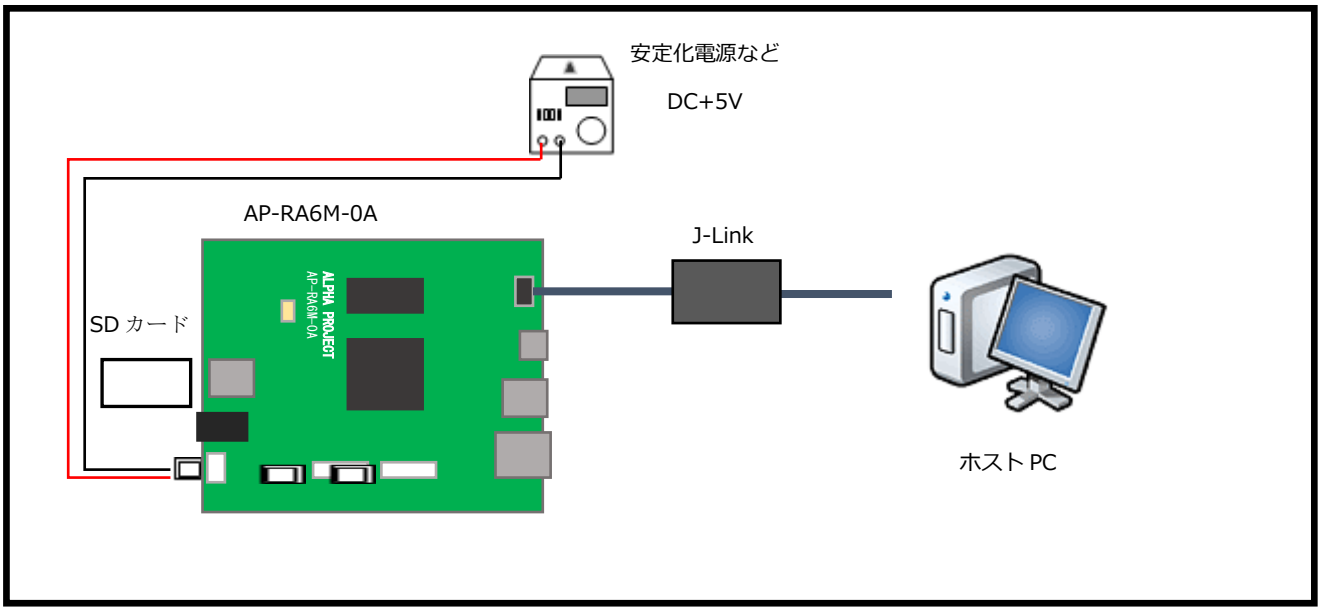
・QSPI サンプルプログラム動作時の接続例



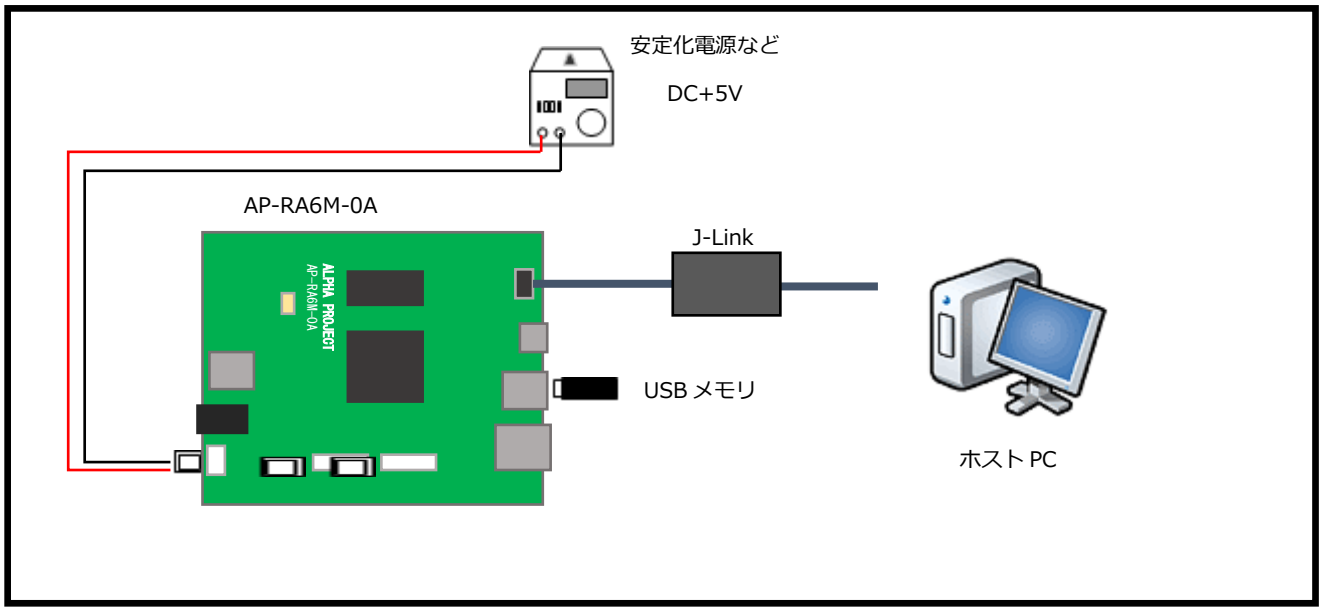
・UART サンプルプログラム動作時の接続例



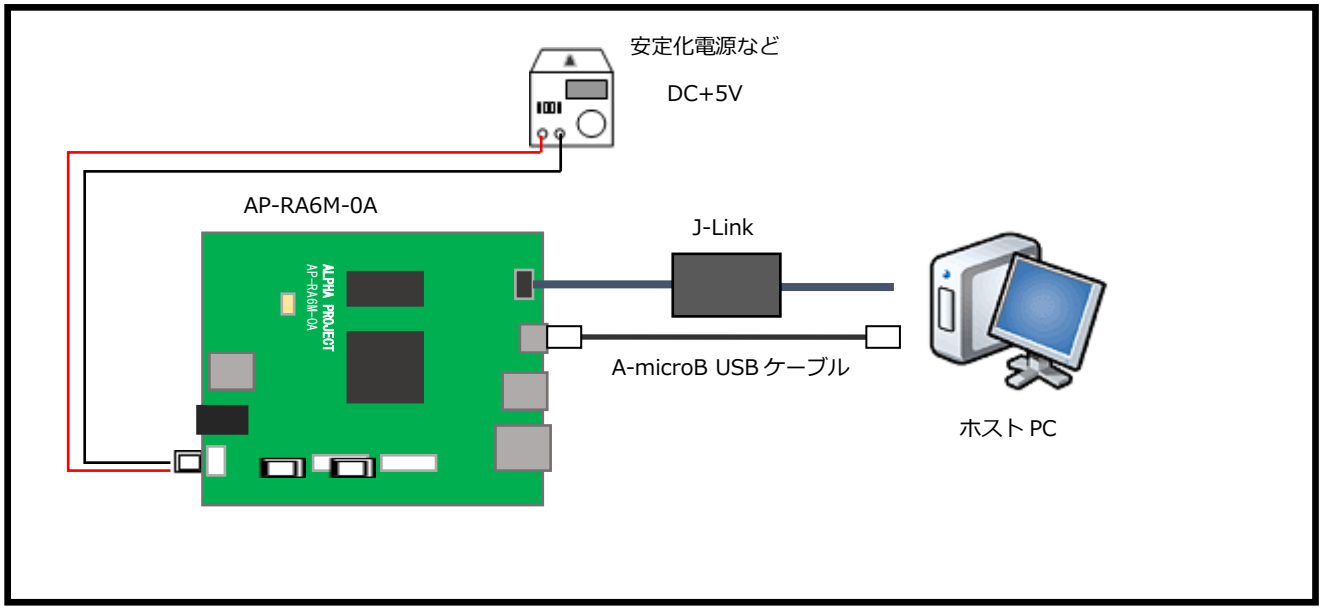
・SDHI サンプルプログラム動作時の接続例



・USB ホストサンプルプログラム動作時の接続例



・USB ファンクションサンプルプログラム動作時の接続例



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および本書含むアプリケーションノートは、弊社 Web サイトのボード紹介ページで公開されています。

株式会社アルファプロジェクト
AP-RA6M-0A 製品ページ　https://www.apnet.co.jp/product/ra/ap-ra6m-0a.html

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「e2 studio」と「Flexible Software Package（以下、FSP）」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、FSP、コンパイラ、デバッガ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
e2 studio	V2023-04	–
FSP	V4.5.0	–
GCC ARM Embedded	V10.3.1.20210824	–
RTTViewer	V7.92b	Segger Microcontroller Systems 社

デバッガ	ハードウェアバージョン	備考
J-Link	V10	Segger Microcontroller Systems 社

※AP-RA6M-0A と J-Link を直接接続することはできません。
AP-RA6M-0A 側(ハーフピッチコネクタ)と J-Link 側(フルピッチコネクタ)を接続するための変換アダプタが必要となります。
変換アダプタについては、J-Link 取扱店へご確認ください。

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
CAN サンプルプログラム	¥sample¥ap_ra6m_0a_can_sample
Ethernet サンプルプログラム	¥sample¥ap_ra6m_0a_ether_sample
QSPI サンプルプログラム	¥sample¥ap_ra6m_0a_qspi_sample
UART サンプルプログラム	¥sample¥ap_ra6m_0a_sci_uart_sample
SDHI サンプルプログラム	¥sample¥ap_ra6m_0a_sdhi_sample
USB ホストサンプルプログラム	¥sample¥ap_ra6m_0a_usb_hmsc_sample
USB ファンクションサンプルプログラム	¥sample¥ap_ra6m_0a_usb_pcdc_sample

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_ra6m_0a_can_sample	CAN サンプルプログラムフォルダ
│　├── .settings	設定ファイルフォルダ
│　├── script	リンカ設定ファイルフォルダ
│　├── src	ソースフォルダ
│　└── Debug	ワークフォルダ
├── ap_ra6m_0a_ether_sample	Ethernet サンプルプログラムフォルダ
│　├── .settings	設定ファイルフォルダ
│　├── script	リンカ設定ファイルフォルダ
│　├── src	ソースフォルダ
│　└── Debug	ワークフォルダ
├── ap_ra6m_0a_qspi_sample	QSPI サンプルプログラムフォルダ
│　├── .settings	設定ファイルフォルダ
│　├── script	リンカ設定ファイルフォルダ
│　├── src	ソースフォルダ
│　└── Debug	ワークフォルダ
├── ap_ra6m_0a_sci_uart_sample	UART サンプルプログラムフォルダ
│　├── .settings	設定ファイルフォルダ
│　├── script	リンカ設定ファイルフォルダ
│　├── src	ソースフォルダ
│　└── Debug	ワークフォルダ
├── ap_ra6m_0a_sdhi_sample	SDHI サンプルプログラムフォルダ
│　├── .settings	設定ファイルフォルダ
│　├── script	リンカ設定ファイルフォルダ
│　├── src	ソースフォルダ
│　└── Debug	ワークフォルダ
├── ap_ra6m_0a_usb_hmsc_sample	USB ホストサンプルプログラムフォルダ
│　├── .settings	設定ファイルフォルダ
│　├── script	リンカ設定ファイルフォルダ
│　├── src	ソースフォルダ
│　└── Debug	ワークフォルダ
├── ap_ra6m_0a_usb_pcdc_sample	USB ファンクションサンプルプログラムフォルダ
│　├── .settings	設定ファイルフォルダ
│　├── script	リンカ設定ファイルフォルダ
│　├── src	ソースフォルダ
│　└── Debug	ワークフォルダ

2.2　ファイルの構成

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。
本節では、サンプルプログラムの作成にあたって追加したファイルについてのみ記述し、自動生成ファイルなどに関しては説明を省略します。

・共通ファイル

<¥sample¥CustomBSP フォルダ内>

AlphaProject.ap_ra6m_0a.4.5.0.pack	・・・	AP-RA6M-0A パックファイル

・CAN サンプルプログラム

<¥sample¥ap_ra6m_0a_can_sample フォルダ内>

.cproject	・・・	CPROJECT ファイル
.project	・・・	PROJECT ファイル
configuration.xml	・・・	FSP コンフィギュレータファイル
ap_ra6m_0a.pincfg	・・・	AP-RA6M-0A ピンコンフィグファイル
ap_ra6m_0a_can_sample	・・・	AP-RA6M-0A CAN サンプルプログラム
Debug.launch		デバッグおよびランタイム設定ファイル

<¥sample¥ap_ra6m_0a_can_sample¥script フォルダ内>

fsp.ld	・・・	e2 studio 用 リンカスクリプトファイル

<¥sample¥ap_ra6m_0a_can_sample¥src フォルダ内>

SEGGER_RTT	・・・	RTTViewer ソース格納フォルダ
timer	・・・	タイマ処理ソース格納フォルダ
common_utils.h	・・・	共通ヘッダファイル
hal_entry.c	・・・	アプリケーションソースファイル

・Ethernet サンプルプログラム

<¥sample¥ap_ra6m_0a_ether_sample フォルダ内>

.cproject	・・・	CPROJECT ファイル
.project	・・・	PROJECT ファイル
configuration.xml	・・・	FSP コンフィギュレータファイル
ap_ra6m_0a.pincfg	・・・	AP-RA6M-0A ピンコンフィグファイル
ap_ra6m_0a_ether_sample	・・・	AP-RA6M-0A Ethernet サンプルプログラム
Debug.launch		デバッグおよびランタイム設定ファイル

<¥sample¥ap_ra6m_0a_ether_sample¥script フォルダ内>

fsp.ld	・・・	e2 studio 用 リンカスクリプトファイル

<¥sample¥ap_ra6m_0a_ether_sample¥src フォルダ内>

i2c	・・・	I2C 通信ソース格納フォルダ
SEGGER_RTT	・・・	RTTViewer ソース格納フォルダ
common_utils.h	・・・	共通ヘッダファイル
hal_entry.c	・・・	hal_entry 関数ソースファイル
net_thread_entry.c	・・・	ネットワークアプリケーションソースファイル
usr_app.h	・・・	ユーザーアプリケーションヘッダファイル

・QSPI サンプルプログラム

<¥sample¥ap_ra6m_0a_qspi_sample フォルダ内>

.cproject	・・・	CPROJECT ファイル
.project	・・・	PROJECT ファイル
configuration.xml	・・・	FSP コンフィギュレータファイル
APRA6M0A_QSPI.pincfg	・・・	AP-RA6M-0A ピンコンフィグファイル（QSPI サンプルプログラム専用）
ap_ra6m_0a_qspi_sample	・・・	AP-RA6M-0A QSPI サンプルプログラム
Debug.launch		デバッグおよびランタイム設定ファイル

<¥sample¥ap_ra6m_0a_qspi_sample¥script フォルダ内>

fsp.ld	・・・	e2 studio 用 リンカスクリプトファイル

<¥sample¥ap_ra6m_0a_qspi_sample¥src フォルダ内>

sdram	・・・	SDRAM 処理ソース格納フォルダ
SEGGER_RTT	・・・	RTTViewer ソース格納フォルダ
common_utils.h	・・・	共通ヘッダファイル
hal_entry.c	・・・	hal_entry 関数ソースファイル
qspi_ep.h	・・・	SPIFlashROM 情報ヘッダファイル

・UART サンプルプログラム

<¥sample¥ap_ra6m_0a_sci_uart_sample フォルダ内>

.cproject	・・・	CPROJECT ファイル
.project	・・・	PROJECT ファイル
configuration.xml	・・・	FSP コンフィギュレータファイル
ap_ra6m_0a.pincfg	・・・	AP-RA6M-0A ピンコンフィグファイル
ap_ra6m_0a_sci_uart_sample	・・・	AP-RA6M-0A UART サンプルプログラム
Debug.launch		デバッグおよびランタイム設定ファイル

<¥sample¥ap_ra6m_0a_sci_uart_sample¥script フォルダ内>

fsp.ld	・・・	e2 studio 用 リンカスクリプトファイル

<¥sample¥ap_ra6m_0a_sci_uart_sample¥src フォルダ内>

SEGGER_RTT	・・・	RTTViewer ソース格納フォルダ
common_utils.h	・・・	共通ヘッダファイル
hal_entry.c	・・・	hal_entry 関数ソースファイル
timer_pwm.c	・・・	PWM タイマ処理ソースファイル
timer_pwm.h	・・・	PWM タイマ処理ヘッダファイル
uart_ep.c	・・・	UART 通信ソースファイル
uart_ep.h	・・・	UART 通信ヘッダファイル

・SDHI サンプルプログラム

<¥sample¥ap_ra6m_0a_sdhi_sample フォルダ内>

.cproject	・・・	CPROJECT ファイル
.project	・・・	PROJECT ファイル
configuration.xml	・・・	FSP コンフィギュレータファイル
ap_ra6m_0a.pincfg	・・・	AP-RA6M-0A ピンコンフィグファイル
ap_ra6m_0a_sdhi_sample	・・・	AP-RA6M-0A SDHI サンプルプログラム
Debug.launch		デバッグおよびランタイム設定ファイル

<¥sample¥ap_ra6m_0a_sdhi_sample¥script フォルダ内>

fsp.ld	・・・	e2 studio 用 リンカスクリプトファイル

<¥sample¥ap_ra6m_0a_sdhi_sample¥src フォルダ内>

SEGGER_RTT	・・・	RTTViewer ソース格納フォルダ
common_utils.h	・・・	共通ヘッダファイル
hal_entry.c	・・・	hal_entry 関数ソースファイル
sdhi_ep.h	・・・	SDHI 情報ヘッダファイル
sdhi_thread_entry.c	・・・	sdhi_thread_entry 関数ソースファイル

・USB ホストサンプルプログラム

<¥sample¥ap_ra6m_0a_usb_hmsc_sample フォルダ内>

.cproject	・・・	CPROJECT ファイル
.project	・・・	PROJECT ファイル
configuration.xml	・・・	FSP コンフィギュレータファイル
ap_ra6m_0a.pincfg	・・・	AP-RA6M-0A ピンコンフィグファイル
ap_ra6m_0a_usb_hmsc_sample Debug.launch	・・・	AP-RA6M-0A USB ホストサンプルプログラムデバッグおよびランタイム設定ファイル

<¥sample¥ap_ra6m_0a_usb_hmsc_sample¥script フォルダ内>

fsp.ld	・・・	e2 studio 用 リンカスクリプトファイル

<¥sample¥ap_ra6m_0a_usb_hmsc_sample¥src フォルダ内>

SEGGER_RTT	・・・	RTTViewer ソース格納フォルダ
common_utils.h	・・・	共通ヘッダファイル
hal_entry.c	・・・	hal_entry 関数ソースファイル
usb_hmsc_ep.h	・・・	USB ホスト情報ヘッダファイル
usb_hmsc_thread_entry.c	・・・	usb_hmsc_thread_entry 関数ソースファイル

・USB ファンクションサンプルプログラム

<¥sample¥ap_ra6m_0a_usb_pcdc_sample フォルダ内>

.cproject	・・・	CPROJECT ファイル
.project	・・・	PROJECT ファイル
configuration.xml	・・・	FSP コンフィギュレータファイル
ap_ra6m_0a.pincfg	・・・	AP-RA6M-0A ピンコンフィグファイル
ap_ra6m_0a_usb_pcdc_sample Debug.launch	・・・	AP-RA6M-0A USB ファンクションサンプルプログラムデバッグおよびランタイム設定ファイル

<¥sample¥ap_ra6m_0a_usb_pcdc_sample¥script フォルダ内>

fsp.ld	・・・	e2 studio 用 リンカスクリプトファイル

<¥sample¥ap_ra6m_0a_usb_pcdc_sample¥src フォルダ内>

board_cfg.h	・・・	ボード情報ヘッダファイル
common_init.c	・・・	共通初期化ソースファイル
common_init.h	・・・	共通初期化ヘッダファイル
hal_entry.c	・・・	hal_entry 関数ソースファイル
hal_entry.h	・・・	hal_entry 関数ヘッダファイル
r_usb_pcdc_descriptor.c	・・・	USB ファンクション情報ヘッダファイル

3.　AP-RA6M-0A サンプルプログラム

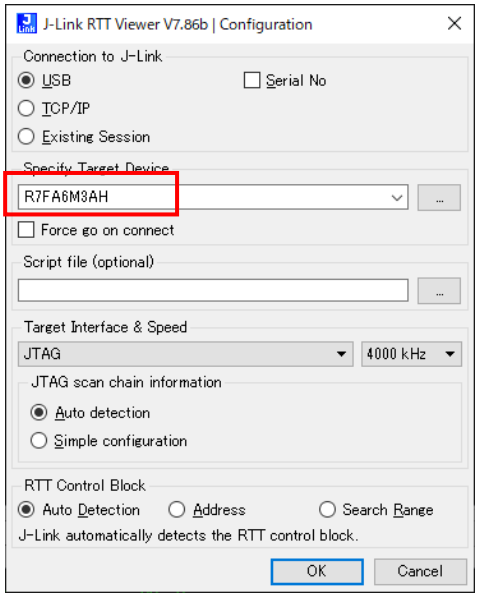
3.1　RTT Viewer 使用方法

サンプルプログラムは、SEGGER 社製ソフトウェア「RTT Viewer」を使用し、動作状況やプログラムの情報が表示される動作があります。
RTT Viewer は、J-Link Software and Documentation Pack に含まれており、下記の SEGGER 社 Web サイトより入手可能です。

SEGGER Microcontroller 社
J-Link / J-Trace Downloads ページ　https://www.segger.com/downloads/jlink/

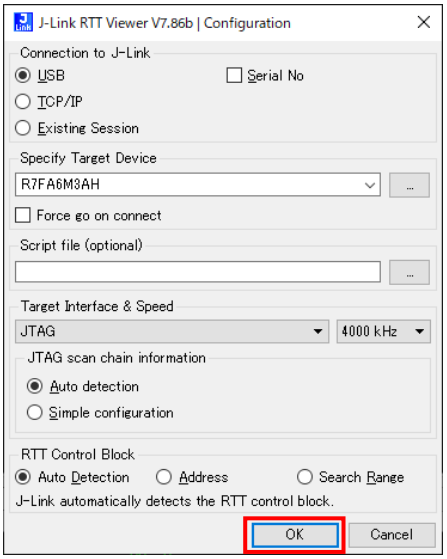
RTT Viewer を使用した接続は、以下の手順に従い行ってください。

① CPU ボードとホスト PC を、J-Link デバッガを使用して接続します。
② ホスト PC にて、「JLinkRTTViewer.exe」を起動します。
③ Configuration ウィンドウが表示されますので、Specify Target Device を「R7FA6M3AH」に設定します。
※.画面表示は、RTTViewer のバージョンにより異なる可能性があります。
以下の手順では、「V7.92b」を使用した場合の手順を記載します。

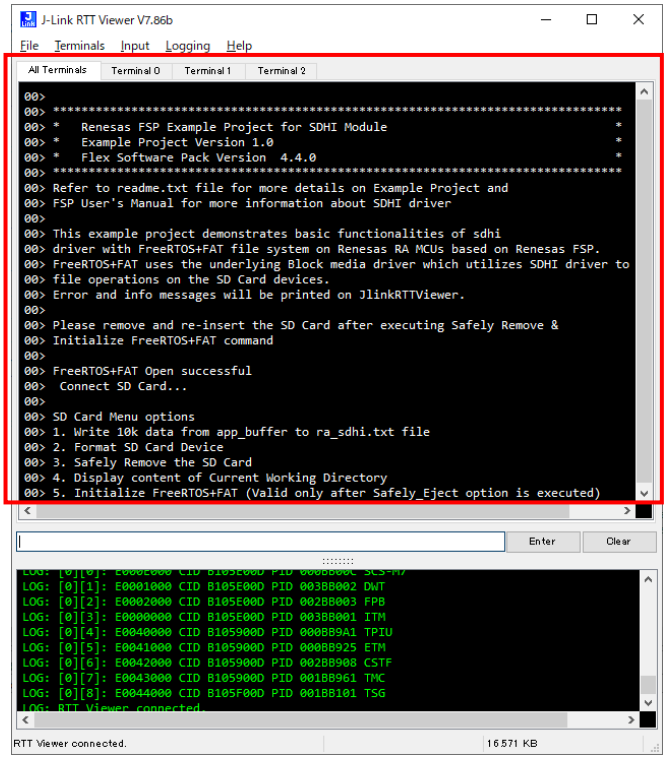


④　CPU ボードに電源を投入し、サンプルプログラムを動作させます。

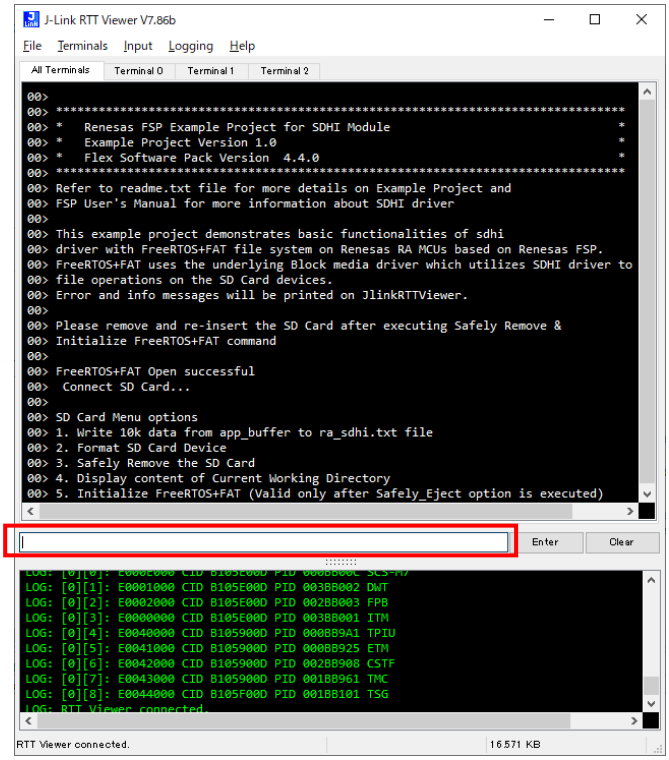
⑤　Configuration ウィンドウの[OK]ボタンを押下し、RTTViewer の接続を開始します。



⑥　接続が完了しますと、CPU ボードからの出力内容が、[Terminal]に表示されます。



⑦　CPU ボードに入力を行う場合は、[Terminal]の入力部にデータを入力します。



⑧　動作確認が完了したら、RTT Viewer を終了した後、CPU ボードの電源を落とします。

RTT Viewer の詳細解説は、「J-Link / J-Trace User Guide」を参照してください。

サンプルプログラムにて RTT Viewer 上に表示される情報は、Renesas サンプルプログラムがベースになっています。
一部表示に関しては、弊社サンプルプログラムの動作と異なる点もございますので、ご注意ください。

サンプルプログラムを実行しても RTT Viewer 上に情報が表示されない場合は、CPU ボードのプログラムをリセットした後、RTT Viewer のメニューバー [File] - [Disconnect] を選択して通信を切断してから改めて再接続を行い、プログラムを再実行してください。

3.2　動作説明

3.2.1　CAN サンプルプログラムの動作説明

本サンプルプログラムでは、プログラム開始後、5sec 間隔で固定データ（TX_MSG）の送信を行いながら、データの受信待ちを行います。
CAN 送信、受信動作の結果は、RTT Viewer に出力します。
※受信データは文字列として RTT Viewer に出力します。受信データに NULL 文字(0x00)が含まれていると、NULL 文字を文字列の終端とみなし、NULL 文字以前の文字までを RTT Viewer に出力します。

CAN の通信設定は、以下のように設定されています。

CPU ボードの設定		
ID	受信 Mailbox ID	B’00000000010　(0x002)
	送信 Mailbox ID	B’00000000011　(0x003)
フォーマット		スタンダードフォーマット、データフレーム、データ長 8 バイト
通信速度		500kbps

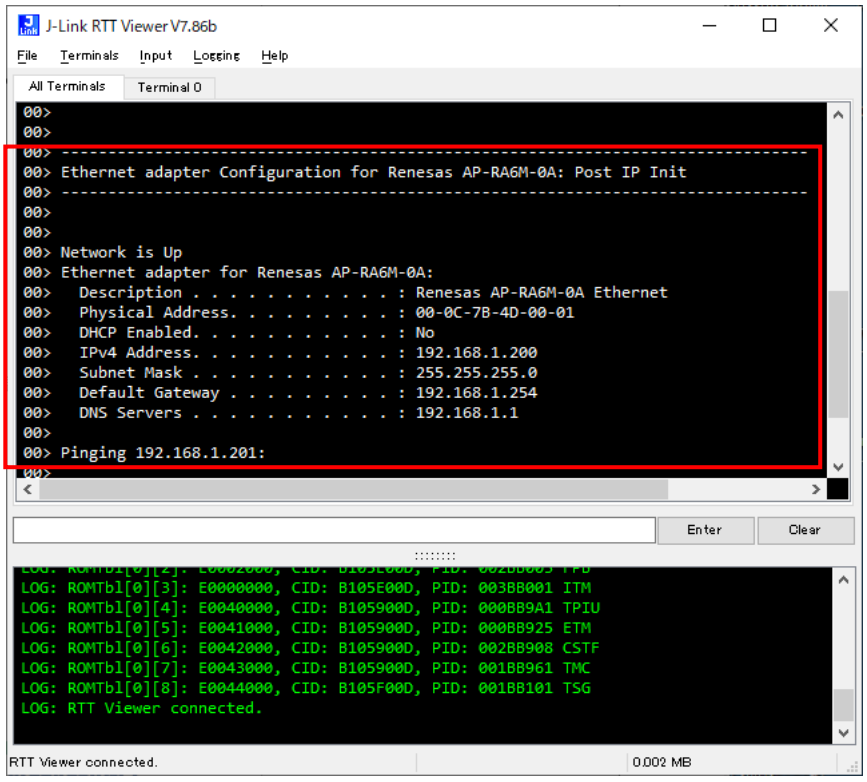
3.2.2　Ethernet サンプルプログラムの動作説明

本サンプルプログラムでは、CPU ボードから固定の IP アドレスに Ping 通信を行います。
ホスト PC を下記のネットワーク設定で動作させ、ボードへ接続した後、通信動作を行ってください。
ネットワーク動作の確認は、以下の手順に従い行ってください。

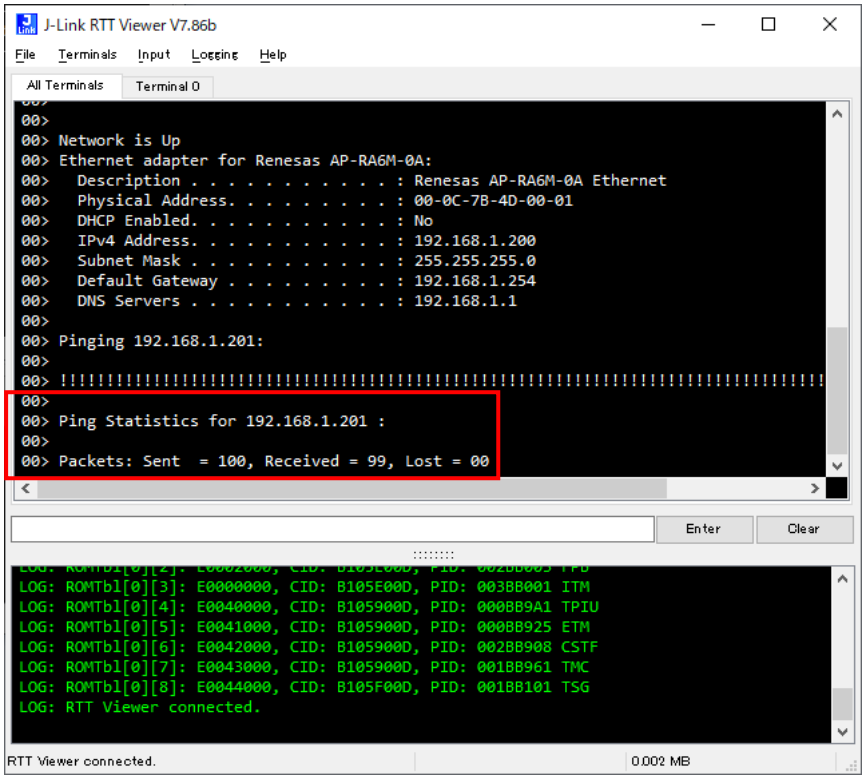
① LAN クロスケーブルを用い、CPU ボードの LAN コネクタ（CN5）とホスト PC を接続します。
② ホスト PC 上でネットワークの設定を行います。
CPU ボードの設定に合わせるため、ホスト PC のネットワーク設定を下記の内容に変更してください。

IP アドレス	192.168.1.201
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。
④ ホスト PC 上で RTTViewer を起動します。接続設定を行い、RTTViewer の接続を確認します。
接続完了が確認できた後に、RTTViewer のターミナル画面を開きます。
⑤ サンプルプログラムが正常に動作した場合は、以下のように、ターミナル画面にネットワーク設定が表示され、
その後、IP アドレス「192.168.1.201」に対し Ping 通信が開始されます。



⑥　正常に Ping 通信が行えた場合、通信結果が表示されます。
また、この状態で PC より CPU ボードに対して、Ping 応答を確認することも可能です。



3.2.2.1　ネットワーク推奨環境

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

3.2.2.2　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
MAC アドレス	00-0C-7B-4D-XX-XX ※　XX-XX の値は製品ごとに異なります。
Ping 通信先 IP アドレス	192.168.1.201

上記設定のうち、IP アドレス・サブネットマスク・ゲートウェイの設定は、サンプルプログラム内で定義しています。
各設定の定義は以下の通りです。

<¥sample¥ap_ra6m_0a_ether_sample¥src¥net_thread_entry.c>

設定	CPU ボードの設定
IP アドレス	ucIPAddress
サブネットマスク	ucNetMask
ゲートウェイ	ucGatewayAddress

<¥sample¥ap_ra6m_0a_ether_sample¥src¥usr_app.h>

設定	CPU ボードの設定
Ping 通信先 IP アドレス	USR_TEST_PING_IP

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	0x4D
+　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。

3.2.3　QSPI サンプルプログラムの動作説明

本サンプルプログラムでは、QSPI FlashROM へデータ書き込み、読出し、Verify チェックを行います。
Verify チェックの結果を、RTT Viewer に出力します。

また、「sample¥ap_ra6m_0a_qspi_sample¥src¥hal_entry.c」内の API「hal_entry(void)」61 行目にある #if を有効にすることで、SDRAM へデータ書き込み、読出し、Verify チェックも行えます。
SDRAM 内の Verify チェックで異常を検知すると、RTT Viewer に「SDRAM Check Error」と出力します。

3.2.4　UART サンプルプログラムの動作説明

本サンプルプログラムでは、シリアル通信(SCI4)を用いてコマンドを送信することで、LED の点灯/消灯を変更することができます。
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など）を使用して行ってください。
ターミナルソフトの COM ポート設定は、115200bps、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。
また、ターミナルソフトの改行コードの設定は、「CR」（0x0d）としてください。

コマンド送信の際は、コマンドの数値と改行コードを送信してください。

コマンド（バイナリ値）	説明
1（0x31, 0x0d）	モニタ LED1 を点灯します。 モニタ LED2 を消灯します。
2（0x32, 0x0d）	モニタ LED1 を消灯します。 モニタ LED2 を点灯します。
3（0x33, 0x0d）	モニタ LED1 を点灯します。 モニタ LED2 を点灯します。
4（0x34, 0x0d）	モニタ LED1 を消灯します。 モニタ LED2 を消灯します。

コマンド送信が正常に行われた場合、「Set next value」と表示され、モニタ LED が点灯/消灯します。
コマンド入力で誤った値が入力されたなど、コマンド送信で異常が発生した場合は、
「Invalid input. Input range is from 1 -4」と表示されます。

3.2.5　SDHI サンプルプログラムの動作説明

本サンプルプログラムでは、RTT Viewer を用いてコマンドを送信することで、SD カードへのデータ読み書きを行うことができます。

SDHI 動作の確認は、以下の手順に従い行ってください。

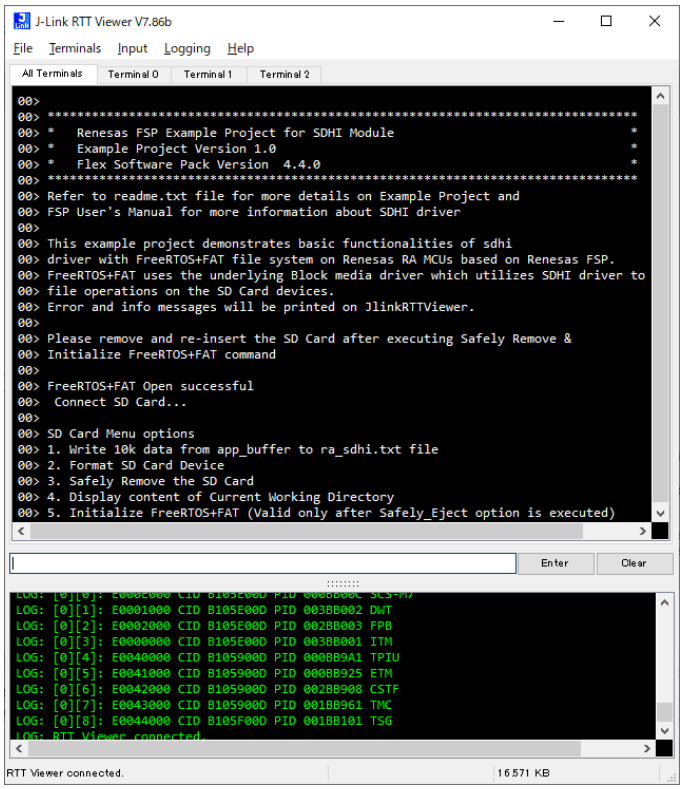
① J-Link を使用し、CPU ボードとホスト PC を接続します。

② ホスト PC 上で RTTViewer を起動します。

③ CPU ボードの SD カードを挿入した後、電源を投入し、サンプルプログラムを動作させます。

④ ホスト PC 上で RTTViewer の接続動作を行います。
接続完了が確認できた後に、RTTViewer のターミナル画面を開きます。

⑤ サンプルプログラムが正常に動作した場合は、以下のように、ターミナル画面に SD 動作ログが表示され、その後コマンド受信待ち状態が表示されます。



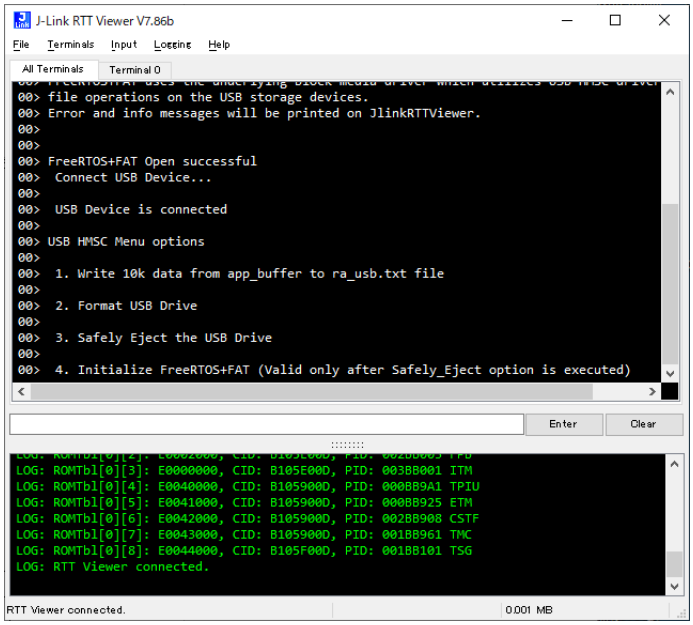
⑥ 下記のコマンド一覧を参考に、ターミナルにコマンドを入力します。
入力されたコマンドに合わせて、下記の動作が行われる事を確認してください。

コマンド	説明
1	SD カードにテスト用ファイルの書き込みと、書き込み後の読み出し確認を行います。
2	SD カードのフォーマットを行います。
3	SD カードの安全な抜出準備を行います。
4	SD カードのカレントディレクトリ情報を表示します。
5	RTOS、FAT の初期化を行います。

3.2.6　USB ホストサンプルプログラムの動作説明

本サンプルプログラムでは、RTTViewer を用いてコマンドを送信することで、USB メモリへのデータ読み書きを行うことができます。
ネットワーク動作の確認は、以下の手順に従い行ってください。

① J-Link を使用し、CPU ボードとホスト PC を接続します。
② ホスト PC 上で RTTViewer を起動します。
③ CPU ボードの USB ホストポート(CN4)に USB メモリを挿入して電源を投入し、サンプルプログラムを動作させます。
④ ホスト PC 上で RTTViewer の接続動作を行います。
接続完了が確認できた後に、RTTViewer のターミナル画面を確認します。
⑤ サンプルプログラムが正常に動作した場合は、以下のように、ターミナル画面に USB 動作ログが表示され、
その後コマンド受信待ち状態が表示されます。



⑥ 下記のコマンド一覧を参考に、ターミナルにコマンドを入力します。
入力されたコマンドに合わせて、下記の動作が行われる事を確認してください。

コマンド	説明
1	USB メモリにテスト用ファイルの書き込みと、書き込み後の読み出し確認を行います。
2	USB メモリのフォーマットを行います。
3	USB メモリの安全な抜出準備を行います。
4	RTOS、FAT の初期化を行います。

3.2.7　USB ファンクションサンプルプログラムの動作説明

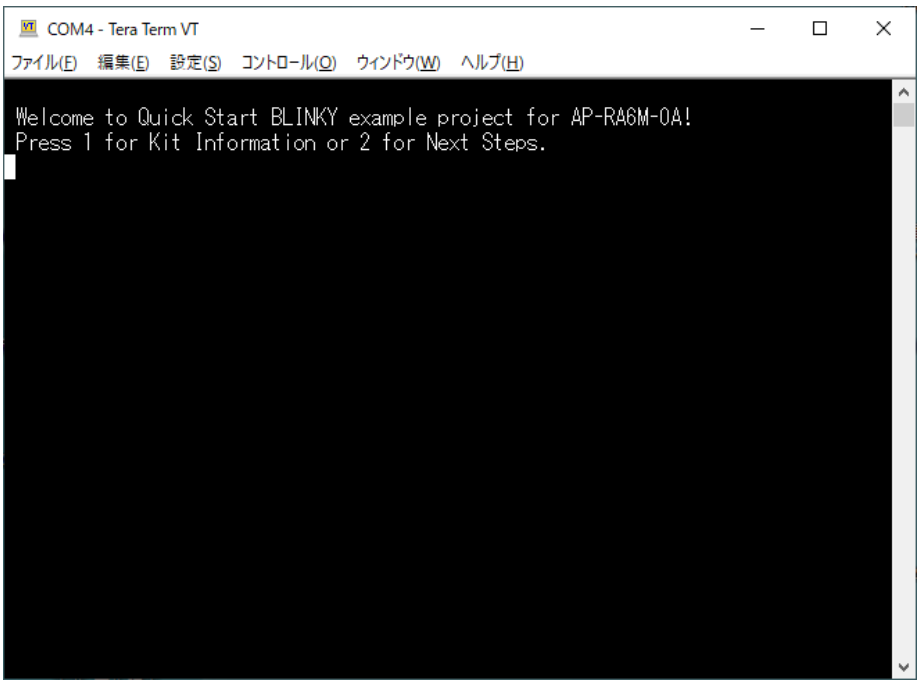
USB ファンクション動作の確認は、以下の手順に従い行ってください。

なお、Win10 よりも前の OS での USB ファンクションの動作確認は、あらかじめ USB 仮想シリアルドライバを PC にインストールしておく必要があります。
インストール方法につきましては、「AN178 USB 仮想シリアルドライバ インストールガイド」を参照してください。

① USB ケーブルを使い CPU ボードの USB ファンクションポート(CN3)とホスト PC の USB ポートを接続します。
② 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。
③ ホスト PC 上でターミナルソフト（ハイパーターミナルなど）を起動し、COM ポートの設定を行います。
その際使用する COM ポートは、
「AN178 USB 仮想シリアルドライバ インストールガイド」内で確認した仮想 COM ポートを選択してください。
COM ポートを以下の設定に変更します。

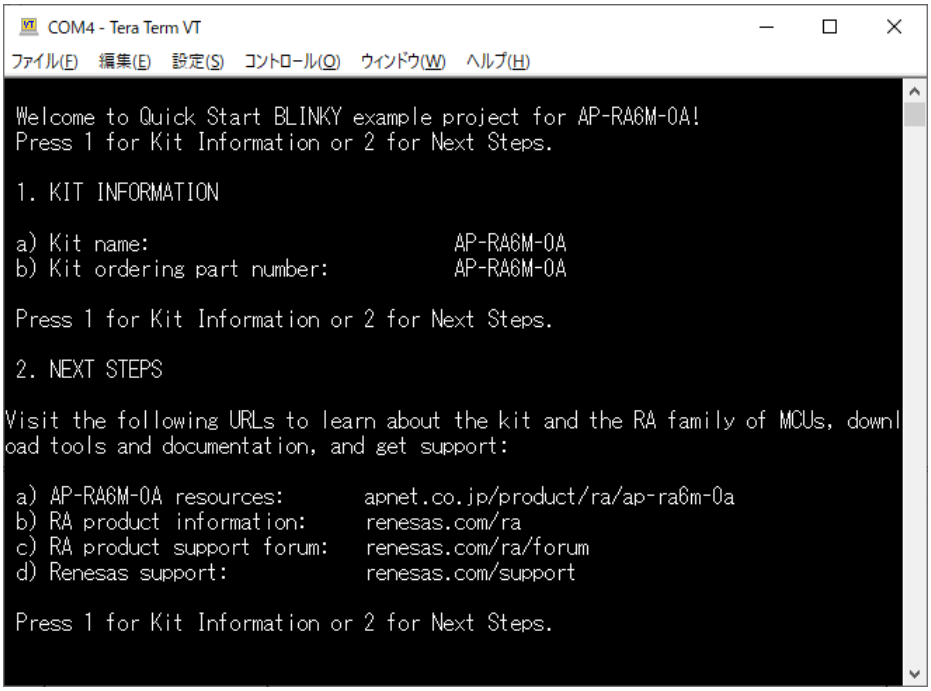
ボーレート	38400bps
ビット長	8bit
パリティ	なし
ストップビット	1bit
フロー制御	なし

④ ターミナルソフトで改行コード「CR」（0x0d）を送信します。
コマンド入力待ち状態が表示されます。



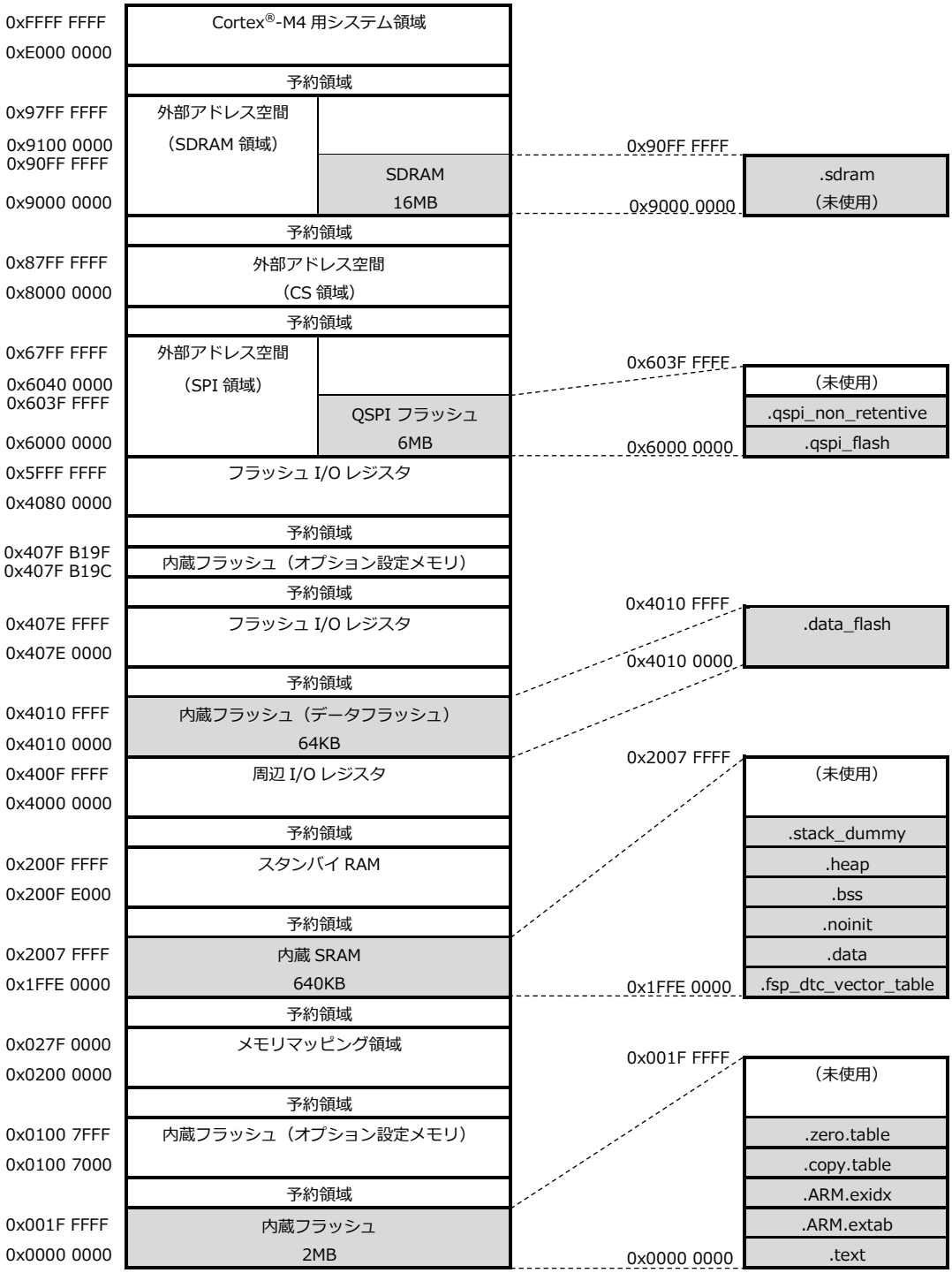
⑤　下記のコマンドを送信することで、情報の出力が行われます。
コマンド送信の際は、コマンドの数値を送信してください。

コマンド	説明
1	ボードの情報を表示します。
2	RA 関係の URL 情報を表示します。



3.3　メモリマップ

e2 studio のプロジェクトのメモリマップを以下に示します。
サンプルプログラムは、全て共通のメモリマップを使用しています。



3.4　e2 studio を用いたプロジェクトのビルド・デバッグ

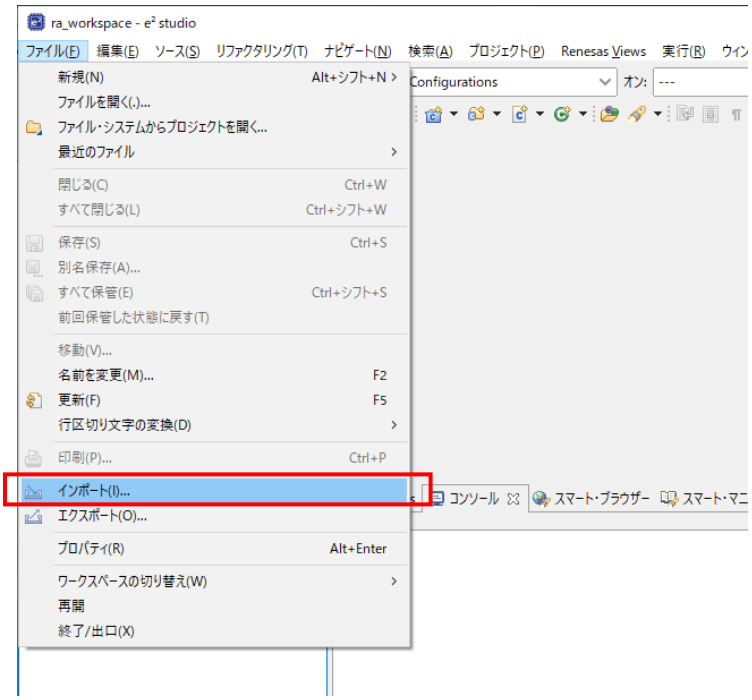
サンプルプログラムを CPU ボード上で実行するためには、e2 studio 上に一度サンプルプログラムをインポートし、ビルドを行う必要があります。

e2 studio 上へのサンプルプログラムのインポート方法、サンプルプログラムのビルド・デバッグ方法については本節で説明します。

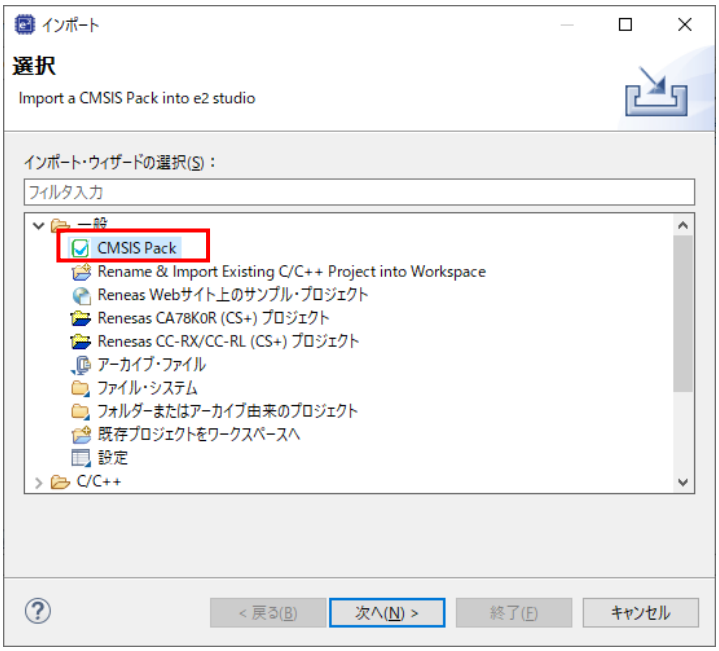
（下記で表示される図は「ap_ra6m_0a_ether_sample」をデバッグ・ビルドする際の例として表示しています。プロジェクト名等は、ビルド・デバッグを行うサンプルプログラムにより変化します。）

3.4.1　インポート方法

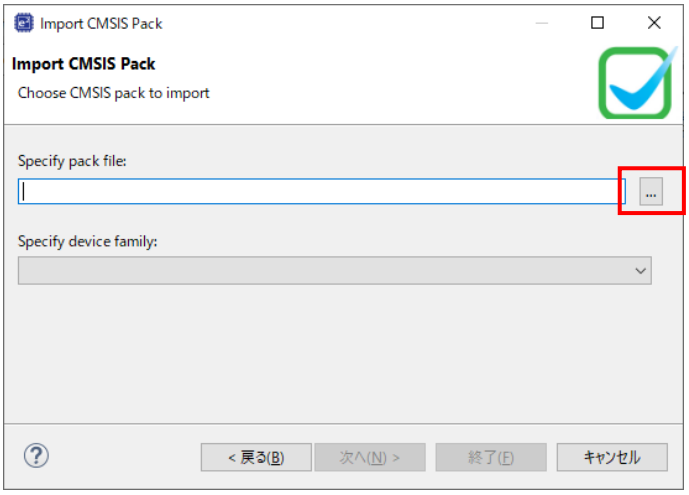
①　e2 studio を起動し、ツールバーの［ファイル］→［インポート］を選択します。



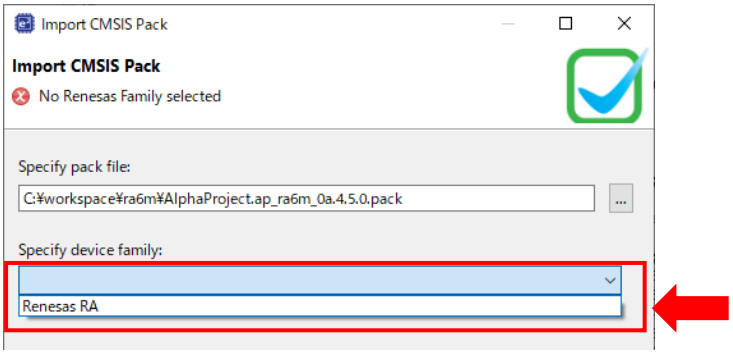
② ［CMSIS Pack］を選択し［次へ］を選択し、pack ファイル「AlphaProject.ap_ra6m_0a.4.5.0.pack」をインポートします。
すでに開発環境に pack ファイルをインポート済みである場合は、⑥へお進みください。



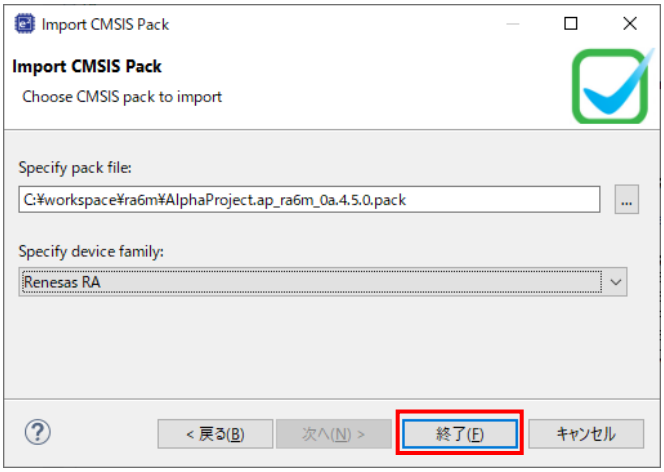
③ ［Import RA CMSIS Pack ウィンドウ］が表示されましたら、インポートする pack ファイル「sample¥ CustomBSP¥ AlphaProject.ap_ra6m_0a.4.5.0.pack」を選択してください。



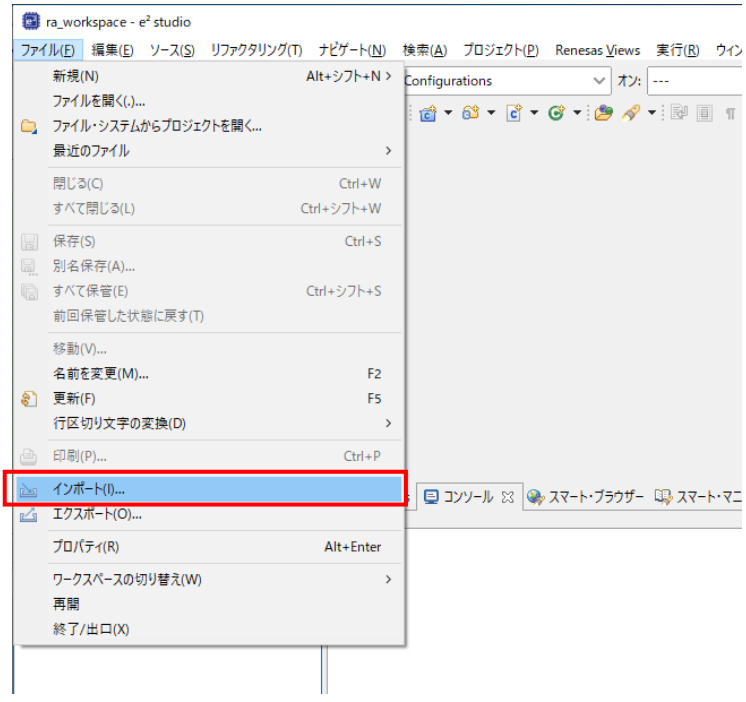
④　pack ファイルを選択後、メッセージ「No Renesas Family selected」が表示されるので、Specify device family から「Renesas RA」を選択してください。



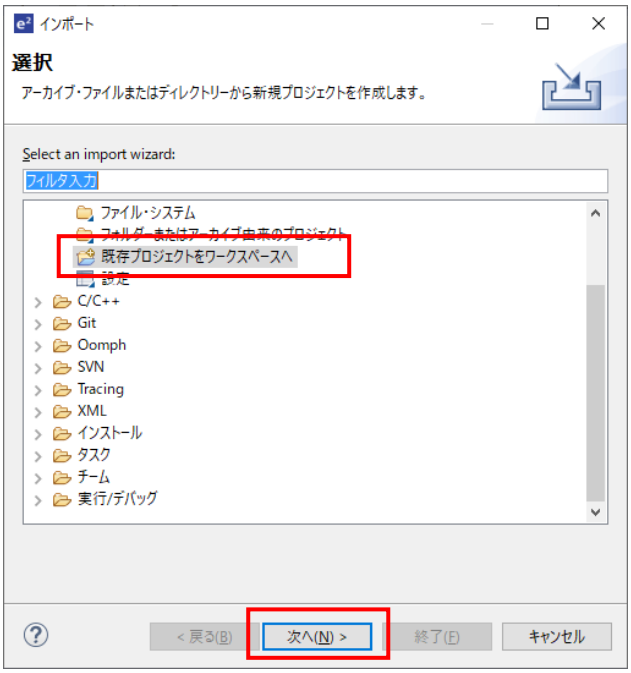
⑤　［終了］を選択してください。



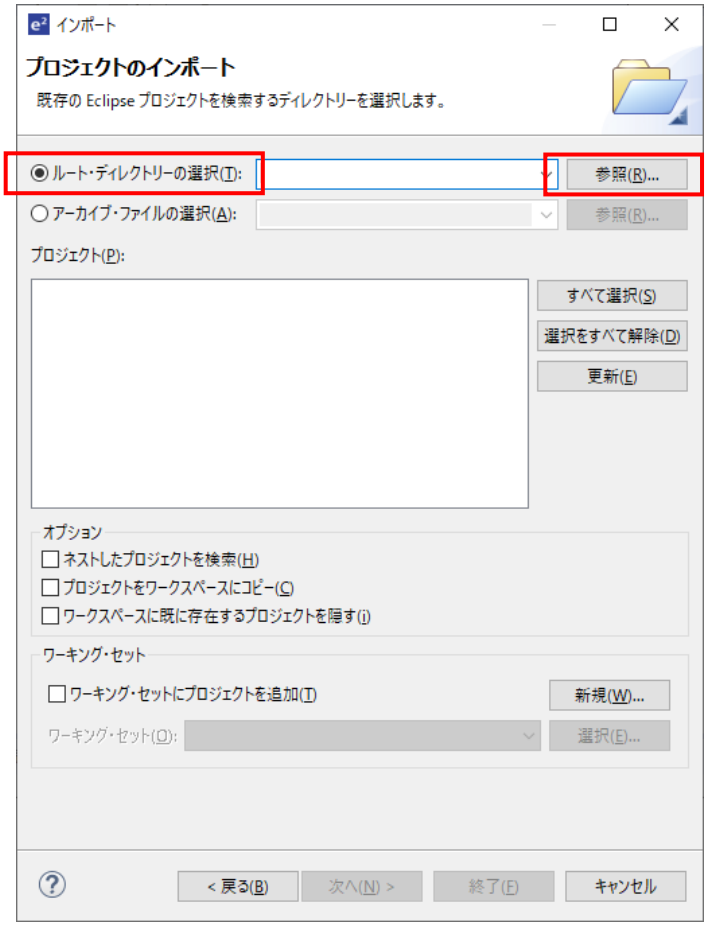
⑥　もう一度ツールバーの［ファイル］→［インポート］を選択します。



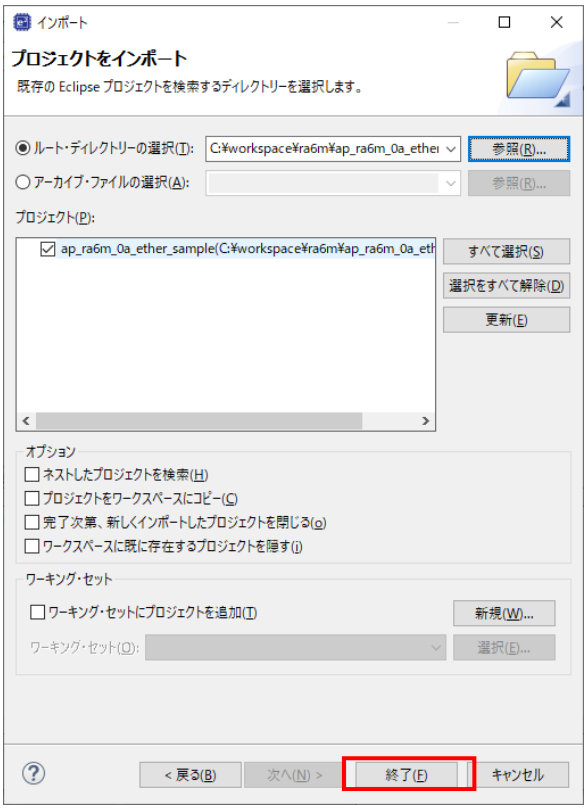
⑦　［既存のプロジェクトをワークスペースへ］を選択し［次へ］を選択します。



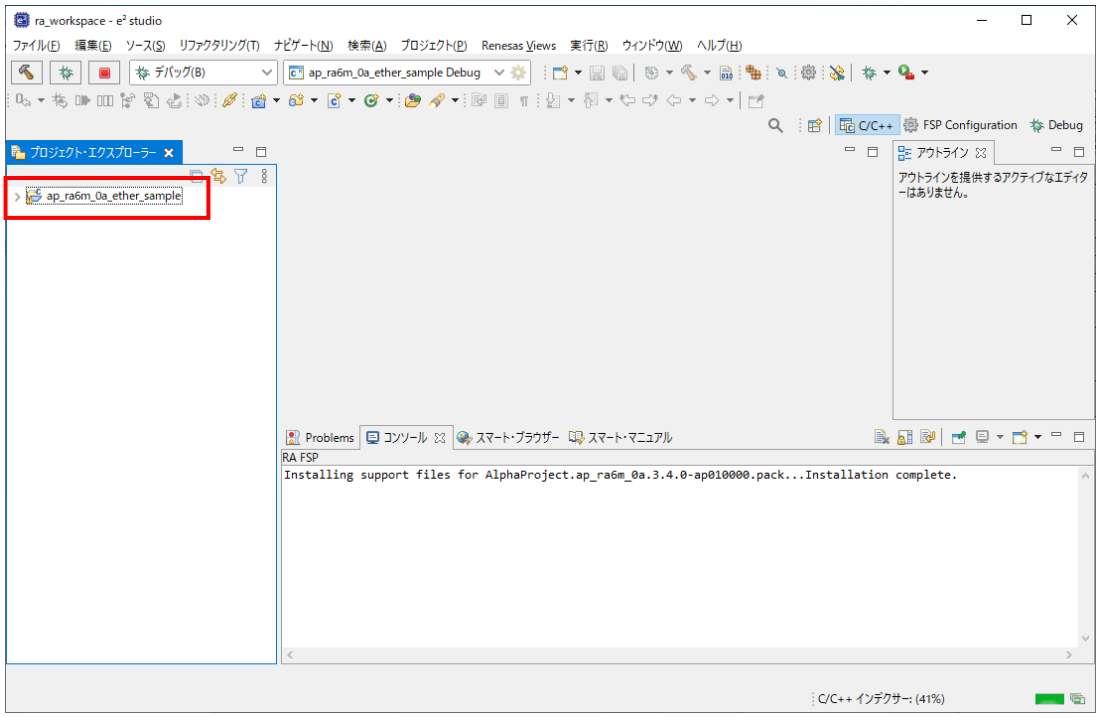
⑧　［ルート・ディレクトリーの選択］を選択し、［参照］からサンプルプログラムのフォルダを選択します。



⑨　[終了]を選択します。



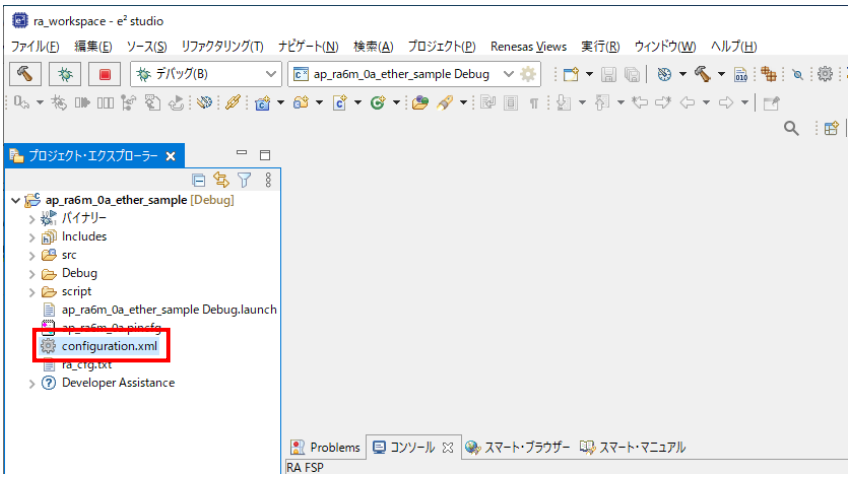
⑩　ナビゲーションウィンドウにサンプルプログラムのプロジェクトが追加されていることを確認します。



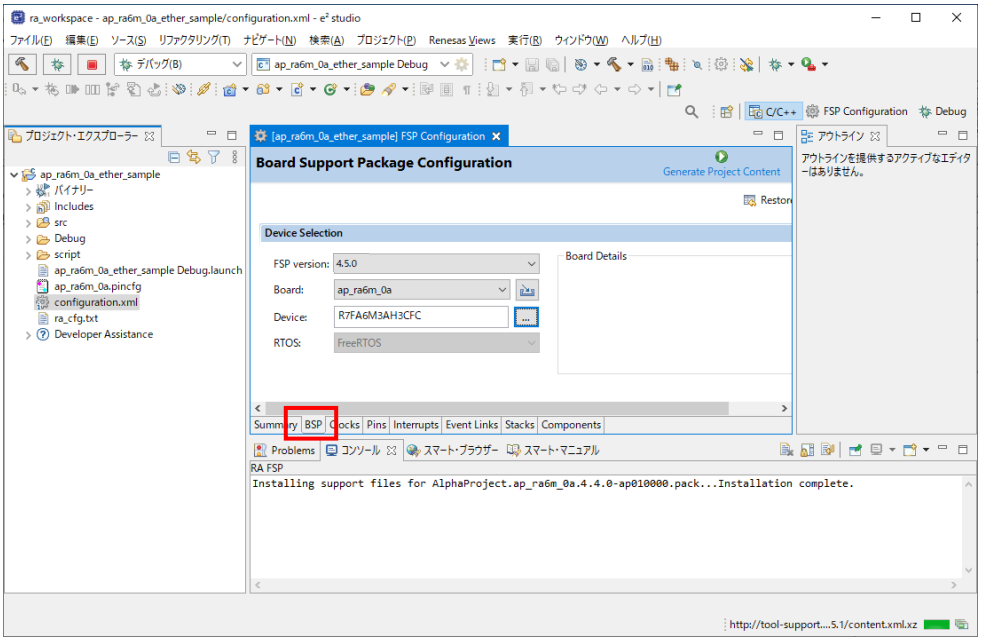
以上でプロジェクトのインポートは完了です。

3.4.2　ビルド方法

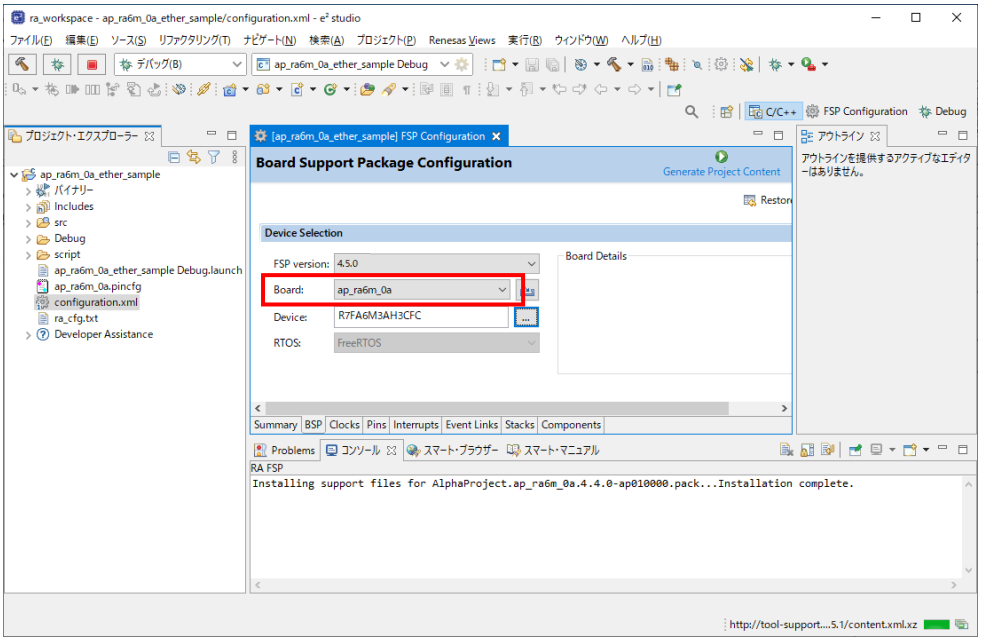
①　プロジェクトのコンフィギュレータファイルを開きます。



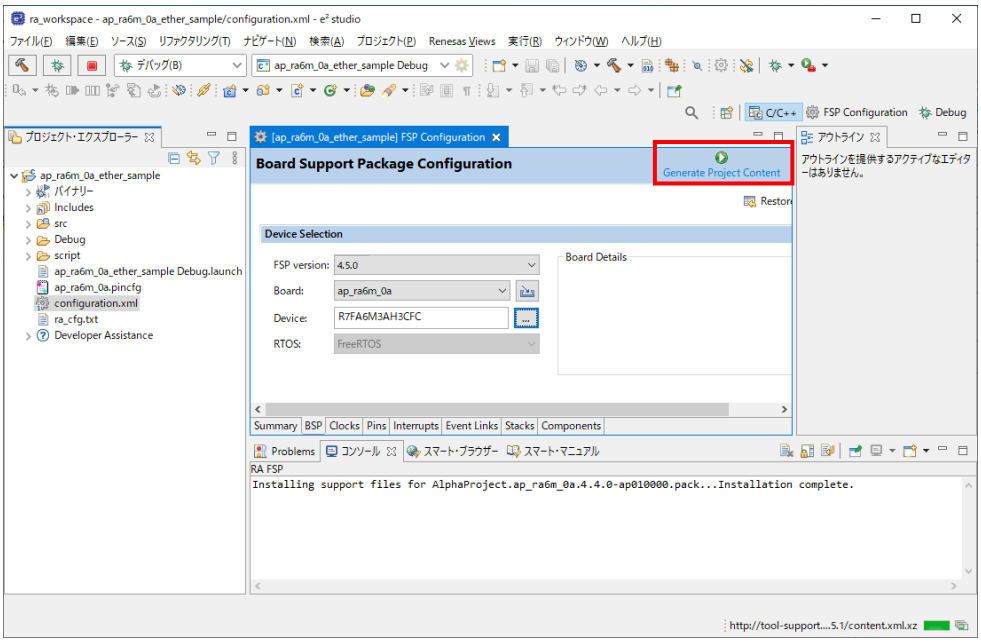
②　[BSP] タブを開きます。



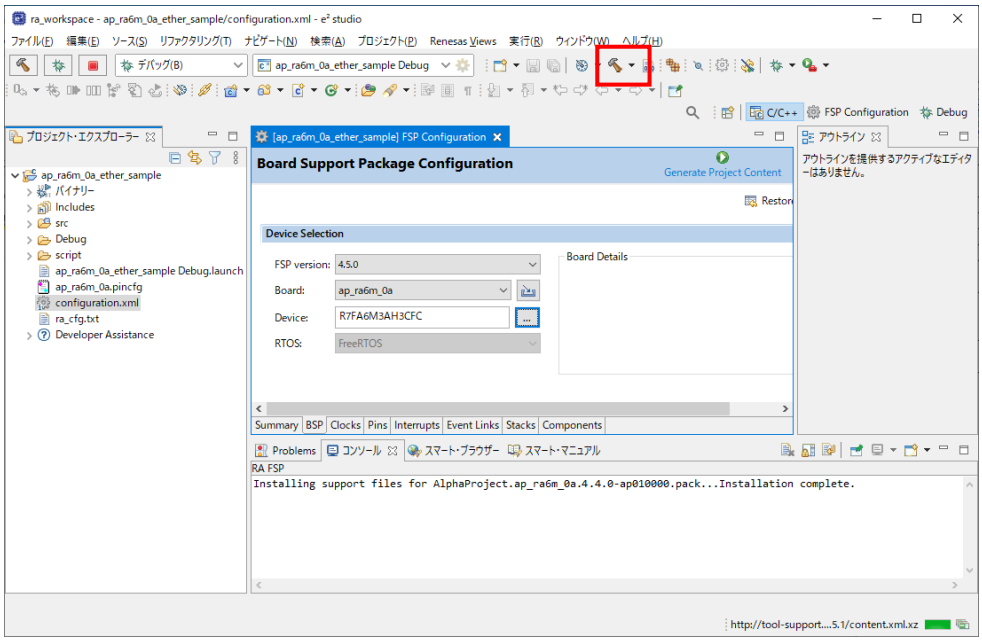
③　[BSP]タブで [Board] が「ap_ra6m_0a」であることを確認します。



④　[Generate Project Content] をクリックし、自動作成ファイルを出力して設定をプロジェクトに適用します。



⑤　ツールバーからビルドアイコンを選択します。
ビルドが成功すると、¥Debug ワークフォルダにオブジェクトファイルが生成されます。

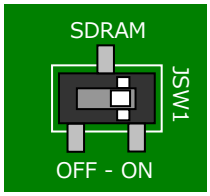


e2 studio の詳細な使用方法に関しては、 e2 studio のマニュアルを参照してください。

3.4.3　デバッグ、ダウンロード方法

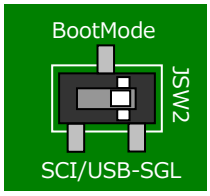
①　「3.3.2　ビルド方法」を参考に、プロジェクトをビルドしてください。

②　ボード上のディップスイッチを以下のように設定してください。



JSW1：ON

ボード上の SDRAM を使用する

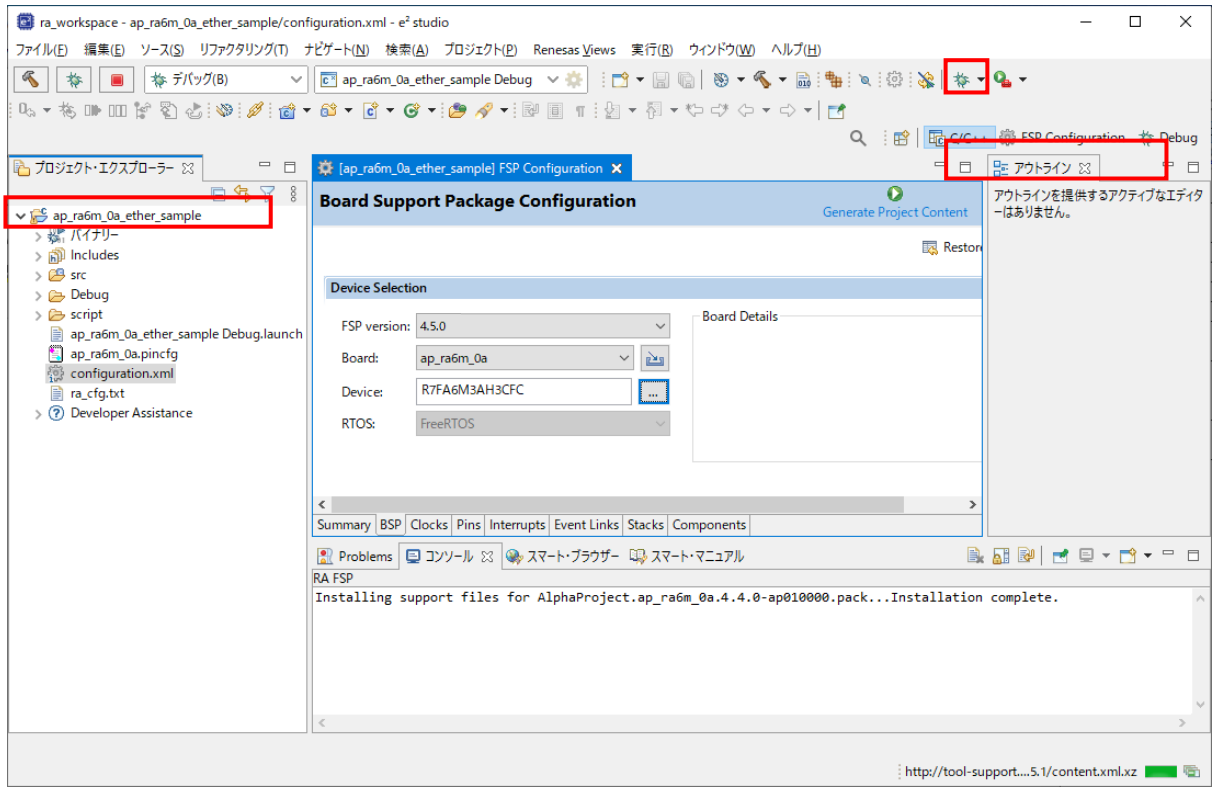


JSW2：SGL

シングルチップモード

③　ボードに電源を投入してください。

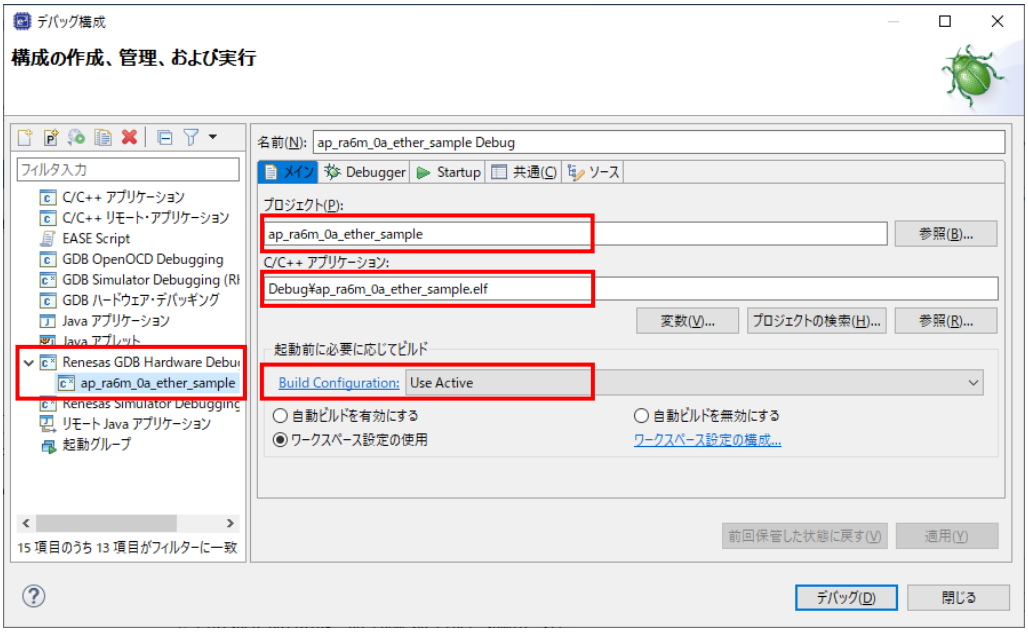
④　プロジェクトを選択し、メニューバーから [デバッグの構成] を開きます。



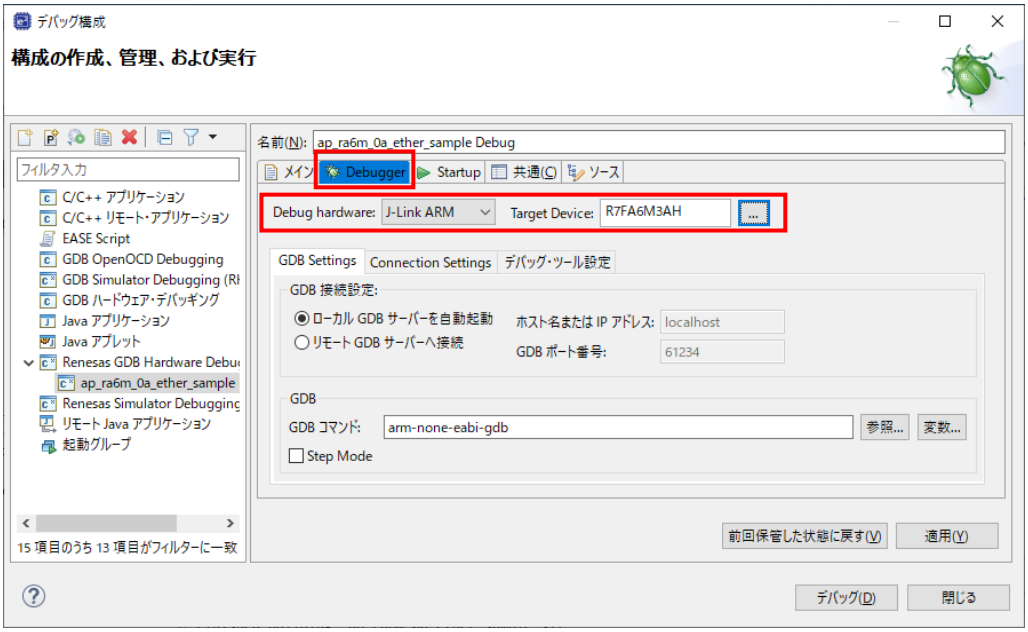
⑤　[Renesas GDB Hardware Debug] の [ap_ra6m_0a_XXXX Debug]を選択し、下記の内容になっていることを確認してください。

- [名前]：ap_ra6m_0a_XXXX Debug
- [プロジェクト]：ap_ra6m_0a_XXXX
- [C/C++アプリケーション]：Debug¥ ap_ra6m_0a_XXXX.elf

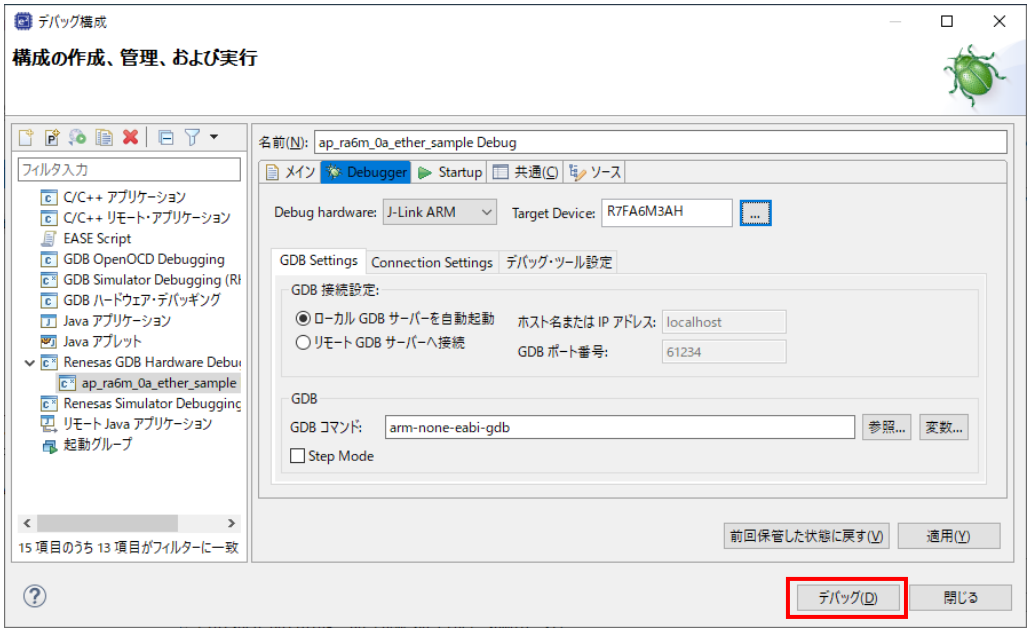
※.XXXX の個所は、デバッグ対象のサンプルプログラムにより名称が異なります。
「2.2 フォルダ構成」を参考に、デバッグ対象のサンプルプログラムに合わせたファイルを選択してください。



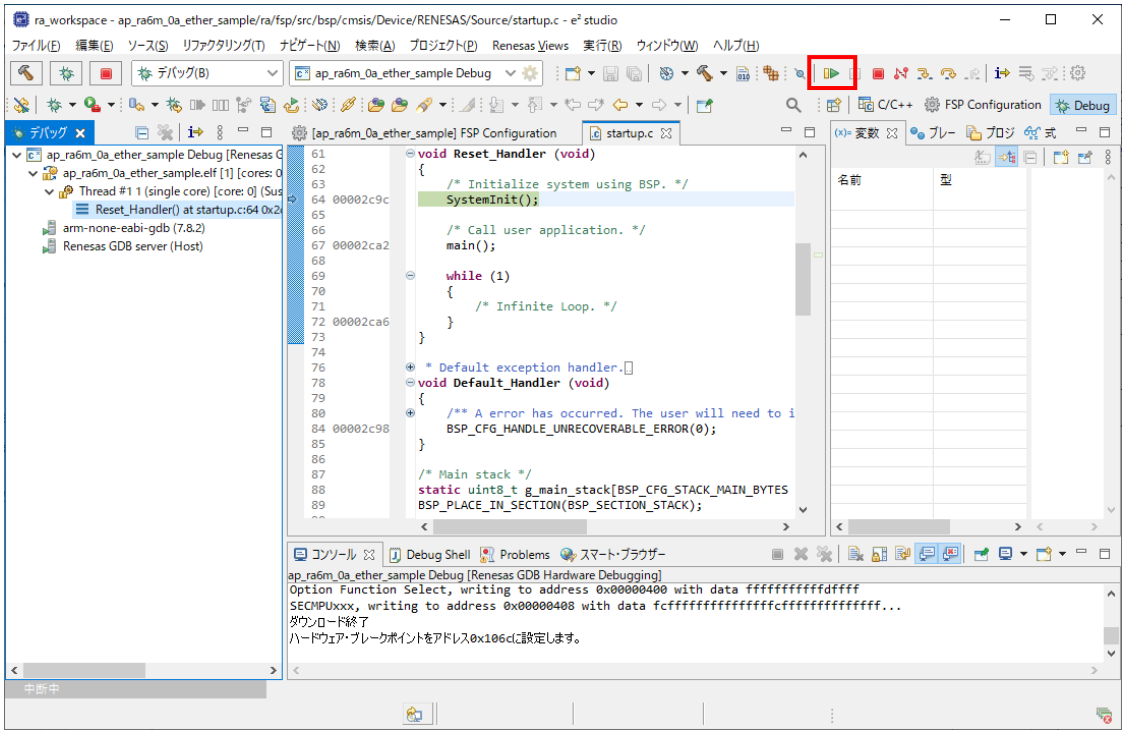
⑥　[Debugger] タブを選択し、 [Debug hardware] が [J-Link ARM] 、 [Target Device] が「R7FA6M3AH」に設定されていることを確認してください。



⑦　[デバッグ] を選択します。



⑧　ボードとの接続が完了したらプログラムを実行し、サンプルプログラムを動作させてください。



⑨　プログラムの動作が確認できましたら、CPU ボードへのプログラムのダウンロードも完了しています。
以降、電源投入によりダウンロードされたプログラムの動作が開始されます。

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

・RA ファミリおよび RA6M3 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 studio 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。
・Flexible Software Package 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。

・Arm®は Arm Ltd.の登録商標です。
・J-Link は、SEGGER Microcontroller GmbH & Co. KG の登録商標もしくは商標です。

・Windows®の正式名称は Microsoft®Windows®Operating System です。
・Microsoft、Windows 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。
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