RZ/N2L グループ

BACnet to OPC UA Gateway Sample Software

R01AN6963JJ0100
Rev.1.00
2023.07.31

要旨

本書は、ビルディングオートメーション(BA) 向け通信プロトコル BACnet を OPC UA に変換する Gateway を RZ/N2L で実行するためのサンプルソフトウェアについて説明します。

動作確認デバイス

RZ/N2L Group

目次

1. 概要 5
1.1 概要 5
1.2 動作環境 6
1.2.1 ソフトウェア環境 6
1.2.2 ハードウェア環境 7

2. ハードウェア構成 8
2.1 RSK ボード設定 8

3. サンプルソフト構成 11
3.1 フォルダ構成 11
3.2 ブートシーケンス 12
3.3 OPC UA Stack 15
3.3.1 OPC UA 15
3.3.2 情報モデル 15
3.3.3 Open62541 16
3.3.4 制限事項 16
3.4 BACnet Stack 16
3.4.1 BACnet Protocol Stack 17
3.4.2 ライセンス 17
3.4.3 仕様、サポート機能 18
3.4.3.1 制限事項 18
3.4.3.2 BACnet Revision 18
3.4.3.3 サービス 18
3.4.3.4 オブジェクト 20
3.4.3.5 BIBB 22
3.5 開発環境構築 24
3.5.1 統合開発環境 e2studio 24
3.5.1.1 インストール 24
3.5.1.2 プロジェクト立ち上げ 28
3.5.2 UaExpert 33
3.5.3 Wireshark 33

4. 動作確認 34
4.1 接続 34
4.2 IP アドレス設定 34
4.3 プロジェクト起動 36
4.3.1 ビルド設定 36
4.3.2 ビルド 38
4.3.3 Debug Configurations 設定 39
4.3.4 デバッグ 42
4.4 BACnet / OPC UA Gateway 通信確認 45
4.4.1 TimeSynchronization メソッド 46
4.4.2 NetworkScan メソッド 48
4.4.3 Write property メソッド 50

4.4.4 Read property メソッド 54
4.4.5 ADD/READ_OBJECT_TREE メソッド 57
4.5 B-GW ボード 1 枚での動作確認 61

5. Appendix 62
5.1 open62541 ファイル生成 62
5.1.1 Linux 環境構築 62
5.1.2 CMake インストール 64
5.1.3 Open62541 ファイル生成 65
5.1.4 生成ファイルの変更点 70
5.2 (参考) B-BC Device Profile 73

用語解説

本書で使用する用語は、以下に示すように定義して使用します。

用語	説明
FSP	Flexible Software Package
RSK	Renesas Starter Kit
BA	Building Automation
BACnet	Building Automation and Control Networking
B-SS	BACnet Smart Sensor
ASHRAE	American Society of Heating, Refrigerating and Air-Conditioning Engineers
ANSI	American National Standards Institute
BIBB	BACnet Interoperability Building Blocks
API	Application Program Interface
APDU	Application Layer Protocol Data Unit
OPC UA	Open Platform Communications Unified Architecture
XML	Extensible Markup Language

関連文書

分類	資料名	資料番号
Data Sheet	RZ/N2L データシート	R01DS0397JJ****
User's Manual	RZ/N2Lグループ ユーザーズマニュアル ハードウェア編	R01UH0955JJ****
User's Manual	Renesas Starter Kit+ for RZ/N2L ユーザーズマニュアル	R20UT4984JG****
Application Note	RZ/N2L Group TCP/IP lwIP Sample Program Package	R01AN6588EJ****
Application Note	RZ/N2L BACnet Sample Software	R01AN6789JJ****

1. 概要

1.1 概要

産業用アプリケーションの相互運用を実現する OPC UA はファクトリーオートメーション(FA)だけでなく広く様々な産業分野にも使われるようになってきています。ビルディングオートメーション(BA)用の主要な通信プロトコルである BACnet と OPC UA とのコンパニオンスペック OPC 30030 が策定され、業界を超えた相互運用が始まっています。

本書は、産業ネットワーク用 RZ プロセッサ RZ/N2L において、OPC UA と BACnet の Gateway を実現するためのサンプルソフトの構成、およびその使い方について説明します。



Fig. 1-1 RSK+ for RZ/N2L

本書で説明するサンプルソフトは、Gateway という機器同士を繋ぐインターフェースであることから、本書での動作確認は Fig. 1-2 のように OPC Client として PC 上のアプリケーションを用い、対向の BACnet サーバーとして、RZ/N2L BACnet Sample Software (r01an6789xx0101-rzn2l-bacnet) を使用します。

説明の都合上、本書で説明する Gateway を B-GW、対向の BACnet サーバーを B-SS と呼びます。

また、サンプルソフトには RZ/N2L RSK ボード 1 枚でも評価できるように、BACnet の疑似データを生成するコードも含みます。

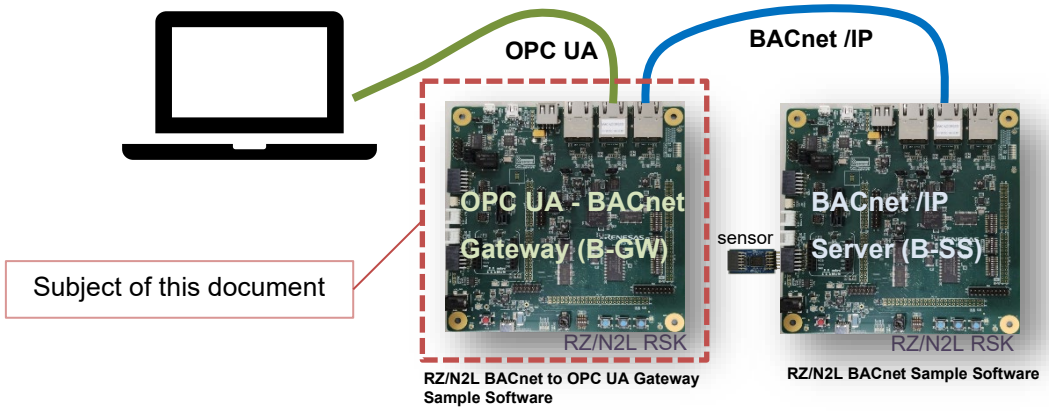


Fig. 1-2 Subject of this document and test setup

1.2 動作環境

1.2.1 ソフトウェア環境

本サンプルソフトの動作環境を Table 1-1 に示します。

Table 1-1 Operating Environment

Category	Name	Version	Link	備考
RZ/N2L BACnet to OPC UA GW サンプルソフト	サンプル パッケージ			
統合開発環境	e2studio	22.10.0	https://www.renesas.com/document/sws/e-studio-and-rzn2l-fsp-installer	e2studio インストーラーに同梱
Flexible Software Package	FSP	1.1.0		e2studio インストーラーに同梱
GNU Arm Embedded Toolchain	GCC Toolchain	V9.3.1.20200408 (注 1)		e2studio インストーラーに同梱
OPC UA クライアントツール	UaExpert	1.6.3	OPC UA Clients - Unified Automation (unified-automation.com)	
Packet analyzer	Wireshark	4.0.7	Wireshark · Download	Packet analyzer

注1. e2studio のインストール時に V10.3.1.20210824 もインストールされますが、V9.3.1.20200408 の使用を推奨しております。

1.2.2 ハードウェア環境

本サンプルソフトは Table 1-2 のハードウェア環境にて、動作確認を行っております。

Table 1-2 ハードウェア環境

Name	Type Name	Maker	Link	Note
Renesas Starter Kit+ for RZ/N2L	RTK9RZN2L0S00000BE	Renesas Electronics	www.renesas.com/rskrzn2l	RSK ボード 2pcs
空気速度センサ Pmod™ボード	US082-FS3000EVZ	Renesas Electronics	US082-FS3000EVZ - 空気速度センサ Pmod™ボード（ルネサス クイックコネクト IoT） | Renesas	ルネサス クイックコネクト IoT

2. ハードウェア構成

本サンプルソフトを実行するハードウェア構成について説明します。

2.1 RSK ボード設定

本サンプルソフトウェアを実行するときの、RSK ボード設定を Fig. 2-1 に示します。

- ブートモードは NOR Flash ROM ブートモードを使用
- 外部メモリとして、SD-RAM を使用
- RSK ボードでは、外部バス使用時イーサネットポート 2 (ETH2) は使用不可

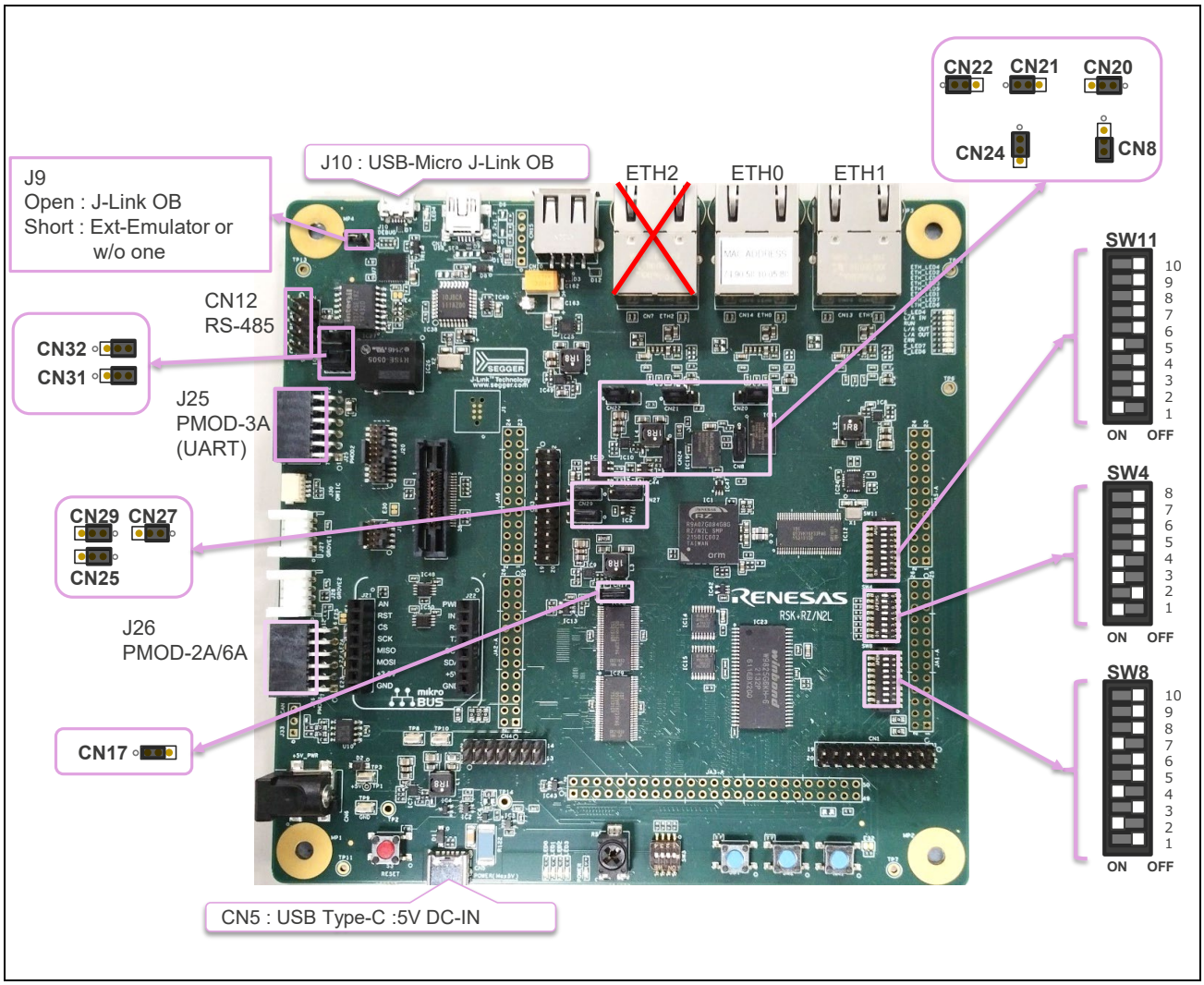


Fig. 2-1 Board Configuration

各スイッチ、ジャンパ設定を Table 2-1, Table 2-2 に示します。赤字は、BACnet サンプルソフト (r01an6789xx0101-rzn2l-bacnet) の RSK ボード設定との差異を示しています。

Table 2-1 DIPSW Settings

DIPSW		Setting	Description
SW11	1	ON	Enable LED_RED2 signal
	2	OFF	
	3	OFF	
	4	OFF	Enable RS485_RX signal
	5	ON	
	6	OFF	Disable P21_5、M2_VP、CAN_RX、ADTRG、P01_7
	7	OFF	
	8	OFF	
	9	OFF	
	10	OFF	
SW4	1	ON	16bit Bus boot mode (NOR Flash ROM Boot)
	2	OFF	
	3	ON	
	4	ON	JTAG Authentication by Hash is disabled
	5	OFF	-
	6	OFF	Enables signals other the trace. (Motor, RS485, etc.) (TRACE_OPTION_SEL=H)
	7	OFF	Enables external bus. (BSC_OPTION_SW=H)
	8	OFF	Enable SW3 (general purpose DIPSW)
SW8	1	OFF	Enable LED_GREEN
	2	ON	
	3	OFF	
	4	ON	Enable LED5
	5	OFF	
	6	OFF	Enable RS485_DE
	7	ON	
	8	OFF	Disable P02_2, IRQ4, CAN_TX
	9	OFF	
	10	OFF	

Table 2-2 Jumper Settings

Jumper	Setting	Description
J9	open	When using the J-Link® OB
	short	When using the external emulator or not using the emulator
CN31	2-3short	RS485 Half Duplex
CN32	2-3short	RS485 Half Duplex
CN20	1-2short	When using 3 ports in the same PHY mode
CN21	1-2short	When using 3 ports in the same PHY mode
CN22	1-2short	When using 3 ports in the same PHY mode
CN24	1-2short	Connect 3.3V Power rail to VCC1833_3. (Using External Bus)
CN8	2-3short	Select QSPI Serial Flash (QSPI_CS)

CN29	1-2short	USB Serial (UART_USB_RX)
CN27	1-2short	HyperRAM (IC41)
CN25	1-2short	Other than the SHOST interface(Trace, SPI, external bus)
CN17	1-2short	Use 3.3V for VCC1833_2 (disable ETH2)

3. サンプルソフト構成

本章では、サンプルソフトについて説明します。

3.1 フォルダ構成

本サンプルソフトのフォルダ構成を以下に示します。太字は目安として、本サンプルソフトをカスタマイズするときに必要なファイルが含まれるフォルダを示しています。

RZN2L_OPC_BGW_V1.0.0

├──BACNETOSS	BACnet Open Source Software
│ └──bacnet	**BACnet Protocol Stack**
├──OPC_UA_SERVER	**Open62541 , BACnet related OPC UA files**
├──rzn	RZ
│ ├──arm	ARM
│ │ └──CMSIS_5	CMSIS
│ ├──aws	AWS
│ │ └──amazon-freertos	FreeRTOS
│ ├──board	Board
│ │ └──rzn2l_rsk	RZ/N2L Renesas Starter Kit
│ └──fsp	Flexible Software Package
├──rzn_cfg	Configuration
│ ├──aws	AWS
│ └──fsp_cfg	FSP
│ └──bsp	Board Support Package
├──rzn_gen	Generated files
├──script	Linker Script
├──src	**User Thread Entry**
└──user	User files
├──oss	Other OSS
│ ├──amazon-freertos	AWS_OSS
│ └──lwip	lwIP_OSS
└──renesas	Renesas common files
├──application	**User lwIP application**, OPCfeature
├──module	User module
└──oss_deps	lwIP OSS dependencies

Fig.3-1 Folder Structure

3.2 ブートシーケンス

ブート手順とメモリ配置について説明します。

本サンプルソフトは(16-bit bus NOR flash boot mode)です。以下の図はスマート・コンフィグレータの BSP タブを表示したものです。

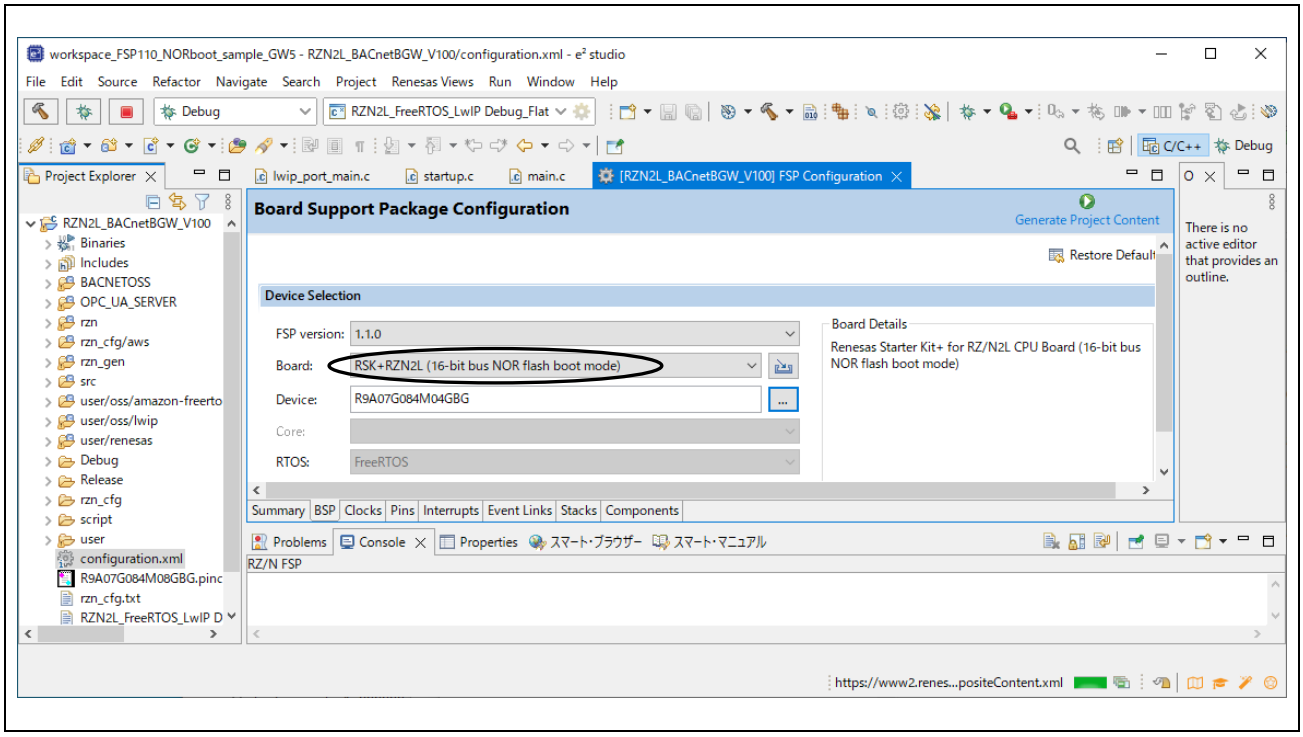


Fig.3-2 Boot mode

フラッシュメモリデバイスにダウンロード後はデバッガ接続なしの状態で RSK ボード上の RESET ボタン押下または電源 OFF/ON を行うとボード単独で動作します。引き続きデバッガを接続して評価を行うこともできます。ただし、RSK ボードの J9 をショートするとデバッガ(J-Link OB)は接続できません。

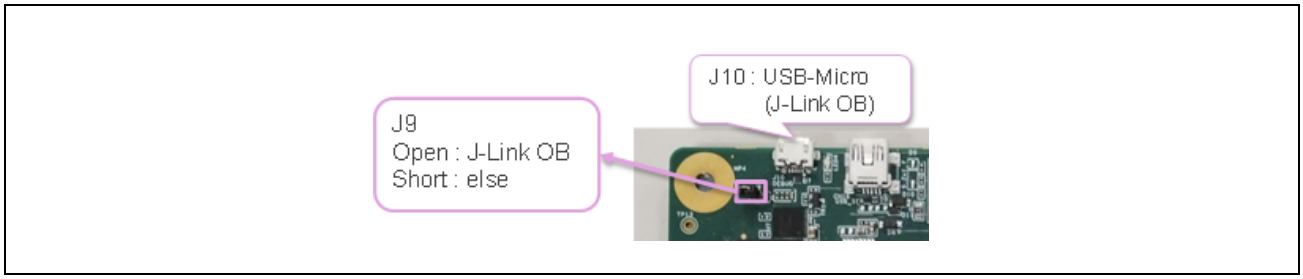


Fig.3-3 J9

NOR フラッシュメモリデバイスと接続する BSC(BusStateControler)の端子設定(Pins タブ)を示すスマート・コンフィグレータ画面です。設定済みのため変更は不要です。

16-bit bus NOR flash boot mode を選択すると、アドレスバス A0-A20 の端子設定は自動で行われますが、プログラムサイズが 2MB 以上の場合、A21-A25 については BSC で個別に端子設定行う必要があります。本サンプルプロジェクトでは設定済のため変更は不要です。

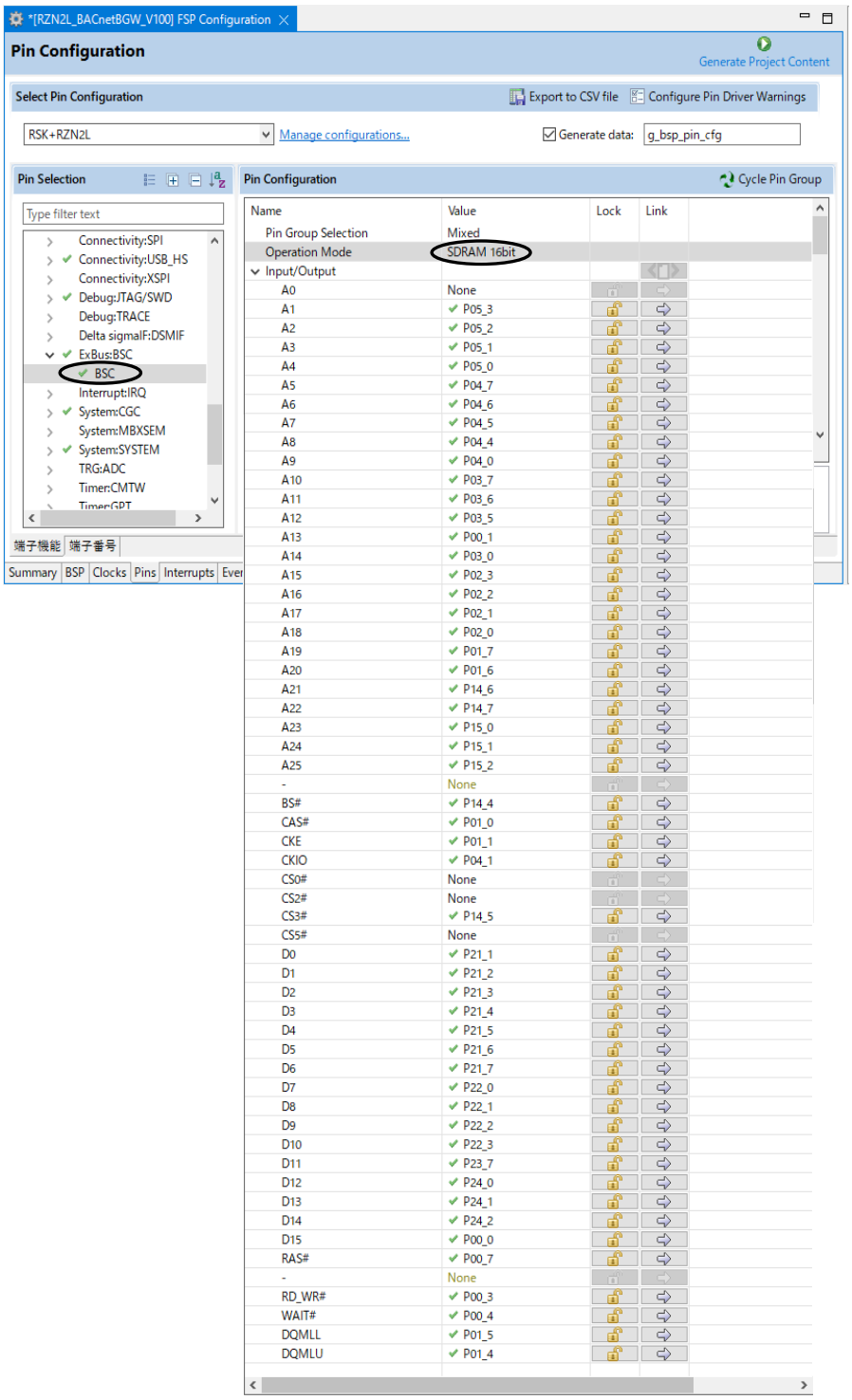


Fig.3-4 BSC Pin Configuration

Fig.3-5 のメモリ配置表の writing order 欄にブートシーケンスにおけるメモリ書き込み順番を示しています。(1)ら(5)の順番で書き込みを行います。

Address	Memory	Content	Length		writing order	remarks
0x00000000	ATCM	intvec(64B)	0x00020000	128KB	(3)	Internal tightly coupled memory
0x00000040		Unused				
0x00000100		hal_entry,ROMdata				
0x00020000	Reserved area	-	-			
0x00100000	BTCM	Unused	0x00020000	128KB	(2)	Internal tightly coupled memory
0x00102000		Loader program(24KB)				
0x00108000		stack(60KB)				
0x00120000	Reserved area	-	-			
0x10000000	SYSTEM_RAM	Body of program and data	0x00180000	1.5MB	(4)	Cached system RAM
0x10180000	Reserved area	-	-			
0x30000000	SYSTEM_RAM_MIRROR	Unused	0x00180000	1.5MB		
0x30180000	Reserved area	-	-			
0x40000000	xSPI0_CS0_SPACE_MIRROR	Unused	0x04000000	64MB		
0x44000000	xSPI0_CS1_SPACE_MIRROR	Unused	0x04000000	64MB		
0x48000000	xSPI1_CS0_SPACE_MIRROR	Unused	0x04000000	64MB		
0x4C000000	xSPI1_CS1_SPACE_MIRROR	Unused	0x04000000	64MB		
0x50000000	CS0_SPACE_MIRROR	Unused	0x04000000	64MB		
0x54000000	CS2_SPACE_MIRROR	Unused	0x04000000	64MB		
0x58000000	CS3_SPACE_MIRROR	Unused	0x04000000	64MB		
0x5C000000	CS5_SPACE_MIRROR	Unused	0x04000000	64MB		
0x60000000	xSPI0_CS0_SPACE	Unused	0x04000000	64MB		
0x64000000	xSPI0_CS1_SPACE	Unused	0x04000000	64MB		
0x68000000	xSPI1_CS0_SPACE	Unused	0x04000000	64MB		
0x6C000000	xSPI1_CS1_SPACE	Unused	0x04000000	64MB		
0x70000000	CS0_SPACE	Parameters for the loader(76B)	0x02000000	32MB	(1)	256M bits NOR Flash
0x7000004C		Loader program(24KB)				
0x7000604C		Body of program and data				
0x72000000		Unused	0x02000000	32MB		
0x74000000	CS2_SPACE	Unused	0x04000000	64MB		
0x78000000	CS3_SPACE	Body of program and data	0x02000000	32MB	(5)	256M bits SDRAM
0x7A000000		Unused	0x02000000	32MB		
0x7C000000	CS5_SPACE	Unused	0x04000000	64MB		

Fig.3-5 Memory layout

3.3 OPC UA Stack

3.3.1 OPC UA

OPC UA は、産業オートメーション分野をはじめとする様々な業界に対して、安全で信頼性のあるデータ交換を実現するためのオープンな通信規格として OPC Foundation により開発されました。

OPC UA の前身となる OPC Classic では Windows ベースでしたが、OPC UA ではマルチプラットフォームとなり、クラウドを含む Windows システムからフィールド機器の RTOS まで様々なプラットフォームで実行可能となりました。本サンプルソフトでは、RZ/N2L のサンプルソフトとして提供している FreeRTOS 上にオープンソースの open62541 プロトコルスタックを実装しています。

3.3.2 情報モデル

ベンダ間、業界間の相互運用を実現するため、OPC UA では情報モデル(Information Model) と呼ばれる統一されたデータモデルが xml ファイル形式で用意されています。OPC UA で共通に利用するビルトインモデルと、業界・団体毎に利用するコンパニオンモデル、更にベンダ毎に利用することが可能なベンダ固有のモデルが含まれます。(Fig. 3-6)

本サンプルソフトでは、情報モデルの .xml ファイルを C 言語コードに変換して使用しております。詳細は、5.1 章をご参照ください。

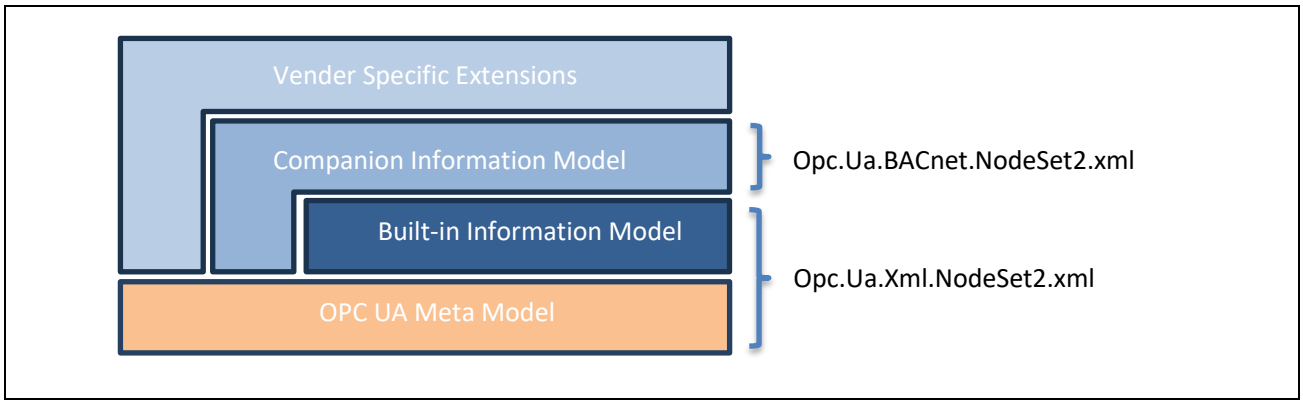


Fig. 3-6 Information Model

(1) Opc.Ua.Xml.NodeSet2.xml

OPC UA 情報モデルを記述する為のルール(メタモデル) とメタモデルによって定義されている OPC UA の基本的な情報モデル(Built-in Information Model) が記述されています。

本サンプルソフトでは、下記バージョンを使用しています。

The OPC UA XML version 1.05.01 (UA-Nodeset/XML/Opc.Ua.Xml.NodeSet2.xml at d1bb6a22125bd7cd986272b1ee98a18a91d76fff · OPCFoundation/UA-Nodeset · GitHub)

(2) Opc.Ua.BACnet.NodeSet2.xml

業界毎に定義されるコンパニオンスペックによる情報モデルの一つで、ビルオートメーション用通信プロトコル BACnet のコンパニオン情報モデルです。

(OPC UA Companion Specification : OPC 30030: BACnet)

本サンプルソフトでは、下記バージョンを使用しています。

bacnet XML version 2.0 (UA-Nodeset/BACnet/Opc.Ua.BACnet.NodeSet2.xml at d1bb6a22125bd7cd986272b1ee98a18a91d76fff · OPCFoundation/UA-Nodeset · GitHub)

3.3.3 Open62541

本サンプルソフトでは、OPC UA サーバーのプロトコルスタックとして、オープンソースの open62541 を使用しています。Open62541 の詳細は下記 Link をご参照ください。

open62541

(1) バージョン

本サンプルソフトで使用している open62541 の Base Version は以下です。

Base Version ：v1.3.4-564-gb7e5e49f3

(commit b7e5e49f32d00490be74c2eacef892c7fbd0be60)

(2) ライセンス

Open62541 のライセンス条件は、MPL v2.0 です。

詳細は https://www.mozilla.org/en-US/MPL/2.0/ をご確認の上、ライセンス条件を遵守してご利用ください。

(3) open62541 ファイル

Open62541 を freeRTOS + LwIP の環境で実行するため、下記 Link 先で紹介されている CMake を利用して open62541.c と open62541.h ファイルを生成する方法を用いています。詳細は、Appendix の 5.1 章 をご参照ください。

Building open62541 — open62541 1.3.0-dirty documentation

3.3.4 制限事項

本サンプルソフトのリリースバージョン V1.0.0 は主に以下の制限事項がございます。

- ✓ セキュリティ証明書には対応しておりません。
- ✓ NTP クライアントには対応しておりません。(Time Synchronization Method により UTC 時刻を取得することは可能です。)

3.4 BACnet Stack

BACnet (Building Automation and Control Network) は、ASHRAE/ANSI Standard 135 で規格化されている主要な BA (Building Automation)通信プロトコルです。空調、照明、防災、アクセス制御など統合してビルの制御、監視を行うことができます。

BACnet デバイスはオペレータやコントローラなど、機能・用途に応じて様々なプロファイルに分類されます。主要なプロファイルとしては、中央監視プロファイル B-OWS (BACnet Operator Workstation)、コントローラプロファイル B-BC (BACnet Building Controller)、各種センサ用プロファイル B-SS (BACnet Smart Sensor) 、さらに前述のコントローラプロファイルと併せ持つことのできる、その他(Miscellaneous)プロファイルとして異なる通信プロトコルデバイス間を中継するための B-RTR(BACnet Router)や B-GW(BACnet Gateway)などがあります。

本サンプルソフトは、BACnet と OPC UA の Gateway (B-GW)を実現しており、B-GW と B-BC の 2 種類のデバイスプロファイルから構成されます。B-GW として OPC UA の Companion Specification OPC 30030: BACnet に従い BACnet が定義するオブジェクトに属するプロパティと OPC UA が定義するノード変数とをマッピングします。OPC UA クライアントのアクセス要求を BACnet/IP ネットワークに接続する BACnet サーバー機器(ここでは、B-SS サンプルソフト)へと中継し、サーバーからの応答を OPC UA クライアントに転送します。

B-SS サンプルソフトのビルドや起動方法などの詳細はアプリケーションノート(R01AN6789JJ****)に記載しています。同アプリケーションノートの 5. BACnet I/P 接続を参照ください。

3.4.1 BACnet Protocol Stack

BACnet Protocol Stack (bacnet-stack) は BACnet 通信プロトコルのオープンソーススタックです。本サンプルソフトは、BACnet Protocol Stack を RZ/N2L に移植しています。

Base Version : eb36033f (Commits on Jan 18, 2023)

GitHub - bacnet-stack/bacnet-stack: BACnet Protocol Stack library provides a BACnet application layer, network layer and media access (MAC) layer communications services.

3.4.2 ライセンス

BACnet Protocol Stack ライセンス条件は GPL with exception license となっています。

参考として原文を以下に転記いたします。詳細は、BACnet Protocol Stack download | SourceForge.net をご確認の上、ライセンス条件を遵守してご利用ください。

> *This BACnet protocol stack implementation is specifically designed for the embedded BACnet appliance, using a GPL with exception license (like eCos), which means that any changes to the core code that are distributed are shared, but the BACnet library can be linked to proprietary code without the proprietary code becoming GPL. Note that some of the source files are designed as skeleton or example or template files, and are not copyrighted as GPL.*
>
> *The text of the GPL exception included in each source file is as follows:*
>
> *"As a special exception, if other files instantiate templates or use macros or inline functions from this file, or you compile this file and link it with other works to produce a work based on this file, this file does not by itself cause the resulting work to be covered by the GNU General Public License. However the source code for this file must still be made available in accordance with section (3) of the GNU General Public License."*

3.4.3 仕様、サポート機能

3.4.3.1 制限事項

本サンプルソフトは BACnet 規格で定義するゲートウェイ(B-GW)デバイスプロファイルをサポートします。B-BC コントローラデバイスプロファイルの機能も一部含んでおりますが、BACnet 規格の標準 B-BC デバイスプロファイル要件は満たしておらず、本バージョンでは非対応となります。

B-BC は次版以降にサポート予定ですが、本バージョンでの対応状況は 5.2 章をご参照ください。

3.4.3.2 BACnet Revision

本サンプルソフトで使用する BACnet スタックの BACnet 規格プロトコルバージョン、およびリビジョンは以下です。

・BACnet standard Protocol Version：1

・BACnet standard Protocol Revision：22

BACnet 規格書(ANSI/ASHRAE Standard 135-2020)が示すバージョンは 1、リビジョンは 22 です。

移植した BACnet スタックのリビジョンのデファイン値は 24 ですが、リビジョン 23 および 24 で追加されたオブジェクトは非対応とし、以下のとおり 22 に変更しています。

BACNETOSS\bacnet\bacdef.h

#define BACNET_PROTOCOL_REVISION 22

3.4.3.3 サービス

本サンプルソフトが実装する BACnet スタックはサービス主導型です。BACnet デバイスの相互運用(Interoperability)は、サービス(WhoIs、I-Am、ReadProperty など)を介してユーザーとプロバイダー間が接続されることで行われます。

サービスは確認なし(Unconfirmed)型と確認あり(Confirmed)型があります。確認なし型はユーザーが要求したサービスに対してプロバイダーは Ack を返しません。一方の確認あり型は Ack を返します。

本サンプルソフトのユーザーとは以下を指します。

BACnet IP プロトコルを使って BACnet サーバーと接続する**クライアント**に該当します。

プロバイダーとは以下を指します。

BACnet IP プロトコルを使って BACnet クライアントと接続する**サーバー**に該当します。

本サンプルソフトで動作する B-GW は BACnet インターネットワークにおける他のプロバイダー(B-SS)に対してはユーザーです。しかし、BACnet インターネットワークにおける他のユーザーに対してはプロバイダーでもあります。

本サンプルソフトが実装するサービスを **Table 3-1** に示します。

Table 3-1 実装サービス

BACnet service	Initiate[1]	Execute[2]
Who-Is	✓	✓
I-Am	✓	✓

BACnet service	Initiate[1]	Execute[2]
Who-Has	✓	✓
I-Have	✓	✓
ReadProperty	✓	✓
WriteProperty	✓	✓
DeviceCommunicationControl		
ReinitializeDevice		
AtomicReadFile		
AtomicWriteFile		
TimeSynchronization		
UTCTimeSynchronization		
SubscribeCOV		
ConfirmedCOVNotification		
UnconfirmedCOVNotification		
ReadPropertyMultiple		
ReadPropertyConditional		
ReadRange		
WritePropertyMultiple		
GetAlarmSummary		
GetEventInformation		
GetEnrollmentSummary		
AcknowledgeAlarm		
ConfirmedEventNotification		
UnconfirmedEventNotification		
UnconfirmedTextMessage		
ConfirmedTextMessage		
AddListElement		
RemoveListElement		
CreateObject		
DeleteObject		
UnconfirmedPrivateTransfer		
ConfirmedPrivateTransfer		
VTOpen		
VTData		
VTClose		

✓は該当、空欄は非該当

1. BACnet サービス要求または通知を送信します。

2. BACnet サービスを実行し、応答(ただし確認あり型サービスを要求された場合)を送信します。

実装サービスの概要は次のとおりです。

Table 3-2 実装サービス概要

BACnet サービス	概要
Who-Is	Who-Is サービスは、ネットワークを共有する他の BACnet デバイスを知るために、BACnet ユーザーによって使用されます。Who-Is サービスはブロードキャスト送信され、確認なし型(Ack を要求しない)サービスです。
I-Am	I-Am サービスは、Who-Is サービス要求に応答するために使用されます。ただし、I-Am サービス要求はいつでも発行可能なブロードキャスト送信です。Who-Is サービス要求の受信が先行する必要はありません。
Who-Has	Who-Has サービスは、BACnet ユーザーが、特定のオブジェクトを持つ BACnet デバイスを識別するために使用されます。Who-Has サービスはブロードキャスト送信され、確認なし型のサービスです。
I-Have	I-Have サービスは、Who-Has サービス要求に応答するために使用されます。ただし、I-Have サービス要求はいつでも発行できます。Who-Has サービス要求の受信が先行する必要はありません。I-Have サービスはブロードキャスト送信され、確認なし型のサービスです。
ReadProperty	ReadProperty サービスは、BACnet ユーザーが 1 つの BACnet オブジェクトの 1 つのプロパティの値を要求するために使用されます。BACnet プロバイダーは Ack を応答して結果を返します。
WriteProperty	WriteProperty サービスは、BACnet オブジェクトの 1 つの指定されたプロパティの値を変更するために、BACnet ユーザーによって使用されます。BACnet プロバイダーは Ack を応答します。指定されたプロパティへの書き込みアクセスを制限したい場合、「エラークラス」PROPERTY および「エラーコード」WRITE_ACCESS_DENIED のエラーが返されます。

3.4.3.4 オブジェクト

BACnet デバイスはオブジェクトの集合で構成されます。

オブジェクトをオブジェクトタイプと 0～4194303 のインスタンス番号で表したものをオブジェクト ID と呼びます。ただし、4194303 番は無効を意味し、使用されていないことを表します。

デバイス自身もオブジェクトであり、Device オブジェクトに定義します。Device のオブジェクト ID をデバイス ID と呼びます。BACnet デバイスは、必ず Device オブジェクトを持つように定められています。さらにオブジェクトは様々なデータ型のプロパティの集合で構成され、このプロパティを読み書きすることで BACnet デバイスはハードウェアにアクセスします。

本サンプルソフトの実装オブジェクトを Table 3-3 に示します。

Table 3-3 実装オブジェクト

BACnet object type	Object ID	Implementation
Accumulator		
Analog Input	Analog Input, 0	✓
	Analog Input, 1	✓
Analog Output		
Analog Value		
Averaging		
Binary Input		

BACnet object type	Object ID	Implementation
Binary Output		
Binary Value		
Calendar		
Command		
Device	Device, 12	✓
Event Enrollment		
File		
Group		
Life Safety Point		
Life Safety Zone		
Loop		
Multi state Input		
Multi state Output		
Multi state Value		
Notification Class		
Program		
Pulse Converter		
Schedule		
Trend Log		
Access Door		
Event Log		
Load Control		
Structured View		
Trend Log Multiple		
Access Point		
Access Zone		
Access User		
Access Rights		
Access Credential		
Credential Data Input		
CharacterString Value		
DateTime Value		
Large Analog Value		
BitString Value		
OctetString Value		
Time Value		
Integer Value		

BACnet object type	Object ID	Implementation
Positive Integer Value		
Date Value		
DateTime Pattern Value		
Time Pattern Value		
Date Pattern Value		
Network Security		
Global Group		
Notification Forwarder		
Alert Enrollment		
Channel		
Lighting Output		
Network Port	NetworkPort,1	✓
Binary Lighting Output		

✓は該当、空欄は非該当

実装オブジェクトタイプの概要は次のとおりです。

Table 3-4 実装オブジェクトタイプ概要

BACnet オブジェクトタイプ	概要
Analog Input	Analog Input オブジェクトはハードウェアからのアナログ入力を表すプロパティを持ちます。
Device	BACnet デバイスには、確実に 1 つの Device オブジェクトが必要です。BACnet デバイス固有の Object_Identifier プロパティを持ちます。これはネットワーク全体でも固有です。
Network Port	Network Port オブジェクトは BACnet デバイスのネットワーク構成を表すプロパティを持ち、少なくとも 1 つのネットワークポートオブジェクトが含まれている必要があります。

3.4.3.5 BIBB

BIBB(BACnet Interoperability Building Blocks)は相互運用する BACnet デバイスに適用するサービスの集合を定義しています。「A」と「B」のデバイスに分かれており、「A」デバイスは BACnet ユーザー、「B」デバイスは BACnet プロバイダーを表します。

BACnet 規格(Annex L)では B-SS(BACnet Smart Sensor)や B-BC(BACnet Building Controller)、B-OWS(BACnet Operator WorkStation)など、各デバイスの性格を表す様々なデバイスプロファイルが定義されています。本サンプルソフトの B-GW は「B」の性格を持つデバイスです。

本サンプルソフトの実装 BIBB を Table 3-5 に示します。

Table 3-5 実装 BIBB(B-GW デバイスプロファイルの場合)

BIBB Class	BIBB	BACnet Service	Initiate[1]	Execute[2]	B-GW Standardized[3]
DataSharing	DS-RP-B	ReadProperty		✓	✓
	DS-WP-B	WriteProperty		✓	✓
	DM-DDB-B	Who-Is		✓	✓

BIBB Class	BIBB	BACnet Service	Initiate[1]	Execute[2]	B-GW Standardized[3]
Device & Network Management		I-Am	✓		✓
	DM-DOB-B	Who-Has		✓	✓
		I-Have	✓		✓
	GW-EO-B	The B device provides access to data and functionality in non-BACnet devices.			✓

✓は該当、空欄は非該当

1. BACnet サービス要求または通知を送信します。
2. BACnet サービスを実行し、応答(ただし確認あり型サービスを要求された場合)を送信します。
3. BACnet 規格で B-GW プロファイルの標準と定めている BIBB です(ANNEX L)。

実装 BIBB の概要は次のとおりです。

Table 3-6 実装 BIBB 概要

BIBB	概要
DS-RP-B	B デバイスは、A デバイスへ、1 つのプロパティ値を返します。
DS-WP-B	B デバイスは、A デバイスからの値を 1 つのプロパティに書き込みます。
DM-DDB-B	B デバイスは、A デバイスからの識別要求に応答します。
DM-DOB-B	B デバイスは、A デバイスから指定されたオブジェクトを持つ識別要求に応答します。
GW-EO-B	B デバイスは、非 BACnet デバイスのデータと機能へのアクセスを提供します。B デバイスには、BACnet オブジェクトおよびサービスを通じて、他のデバイスのデータと機能が含まれます。

3.5 開発環境構築

3.5.1 統合開発環境 e2studio

3.5.1.1 インストール

Table 1-1 に記載のバージョンをダウンロードして、お使いの PC にインストールしてください。

FSP 最新バージョンでは、FSP、e2studio、GCC ツールチェーンが 1 つのパッケージとして同梱されたインストーラーがダウンロードできます。

・ダウンロードした setup_rznfsp_v1_1_0_e2s_v2022-10.exe をダブルクリックします。

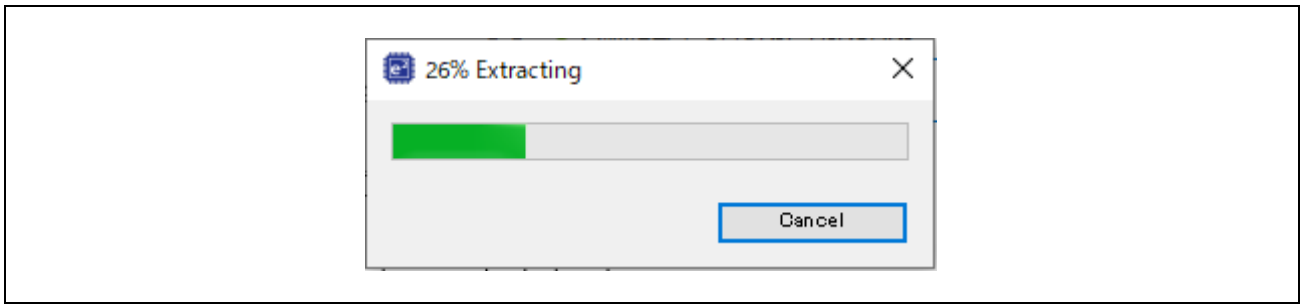


Fig.3-7 e2studio Install (1)

・以下を選択します。

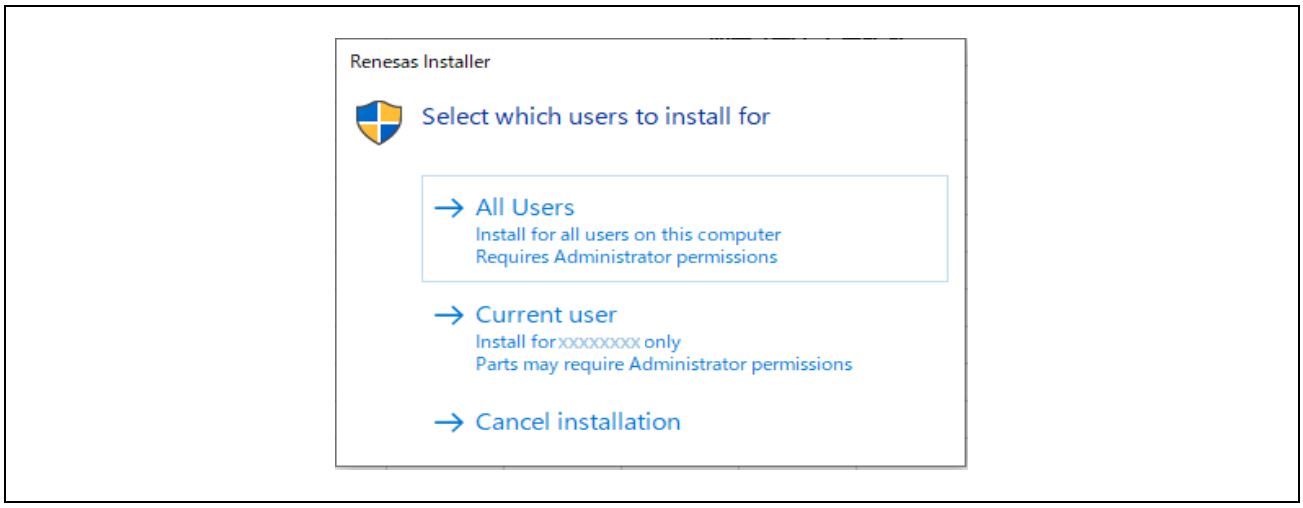


Fig.3-8 e2studio Install (2)

・以下を選択します。

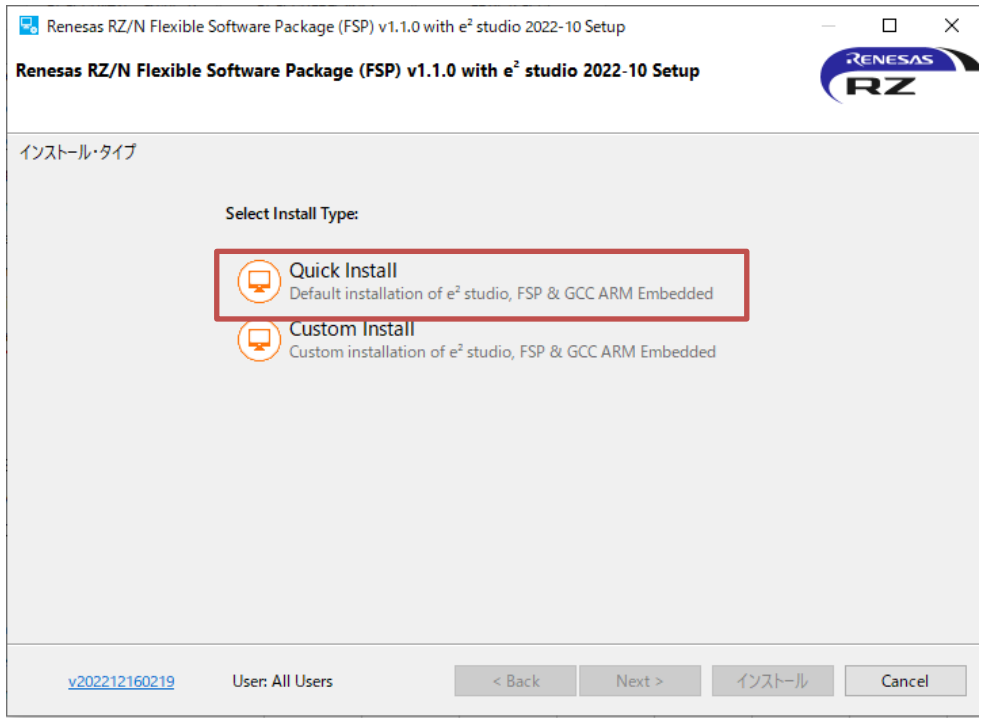


Fig.3-9 e2studio Install (3)

・インストールフォルダを設定します。

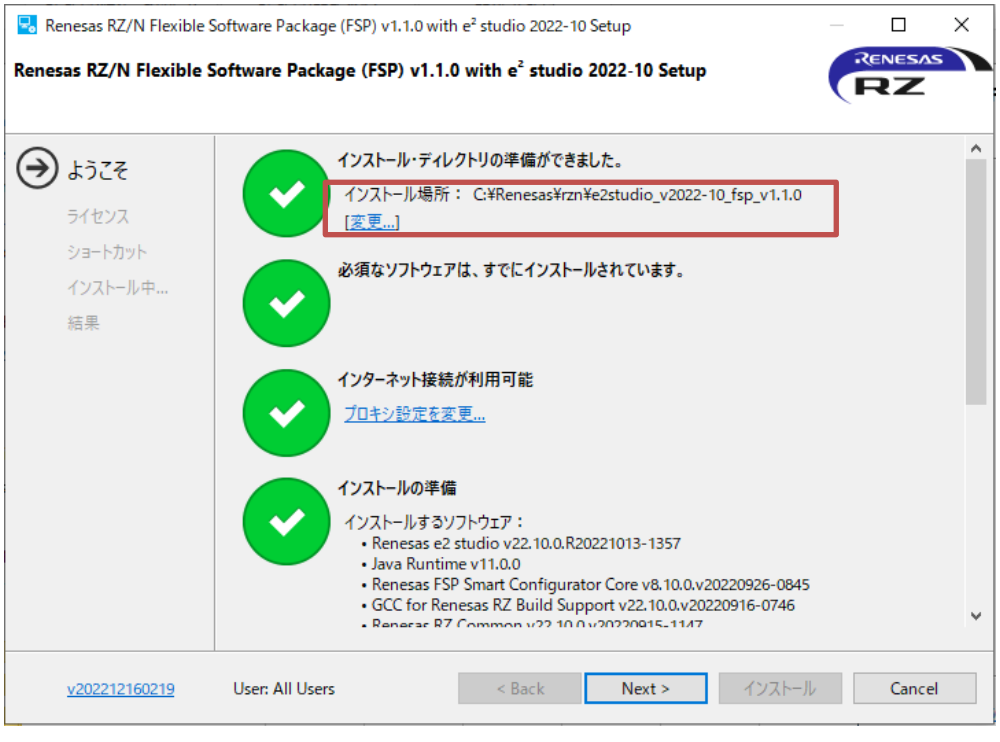


Fig.3-10 e2studio Install (4)

・同意をチェックした後、Next をクリックします。

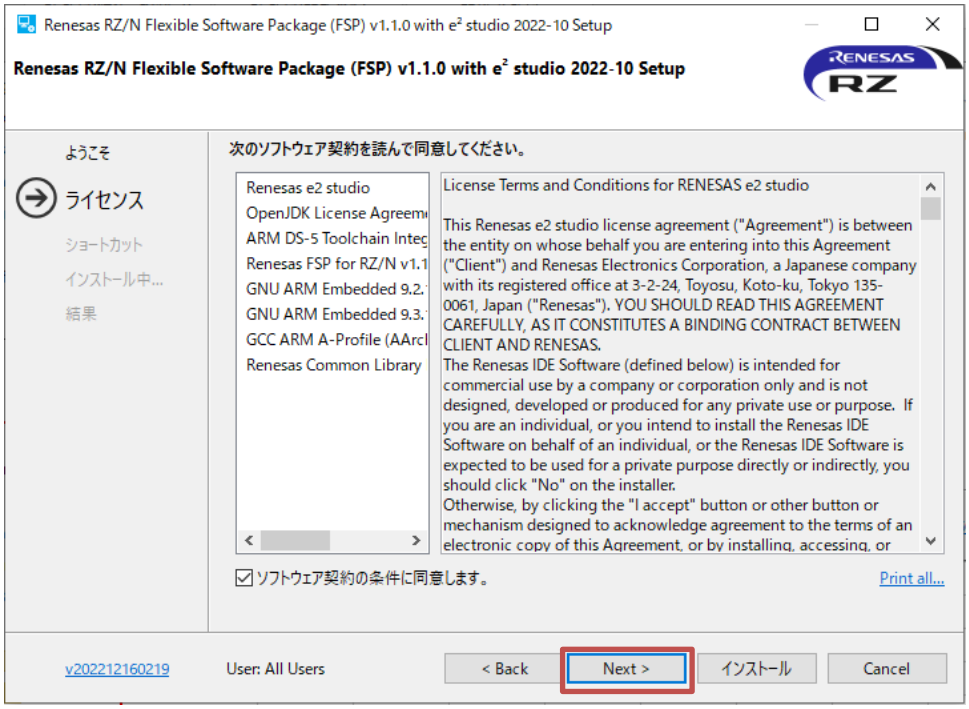


Fig.3-11 e2studio Install (5)

・インストールをクリックします。

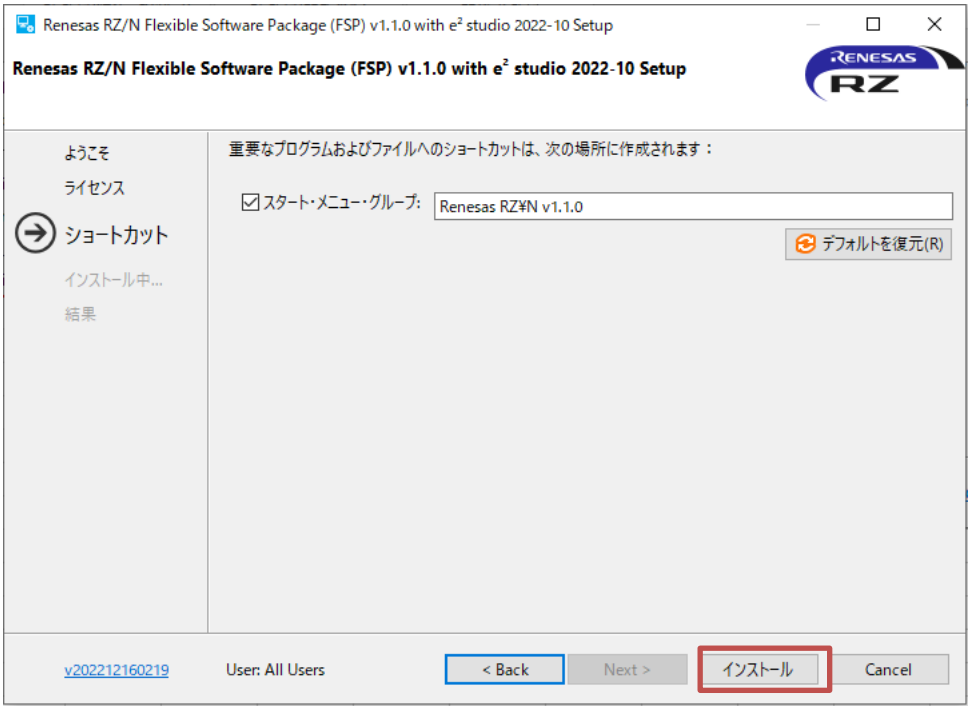


Fig.3-12 e2studio Install (6)

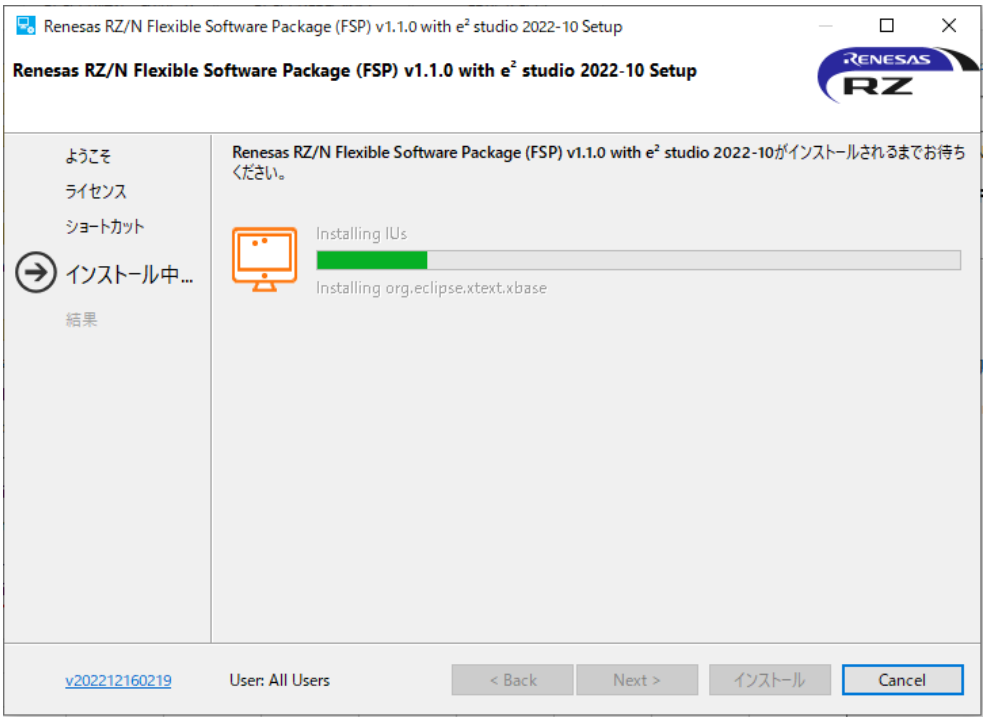


Fig.3-13 e2studio Install (7)

・OK をクリックします。

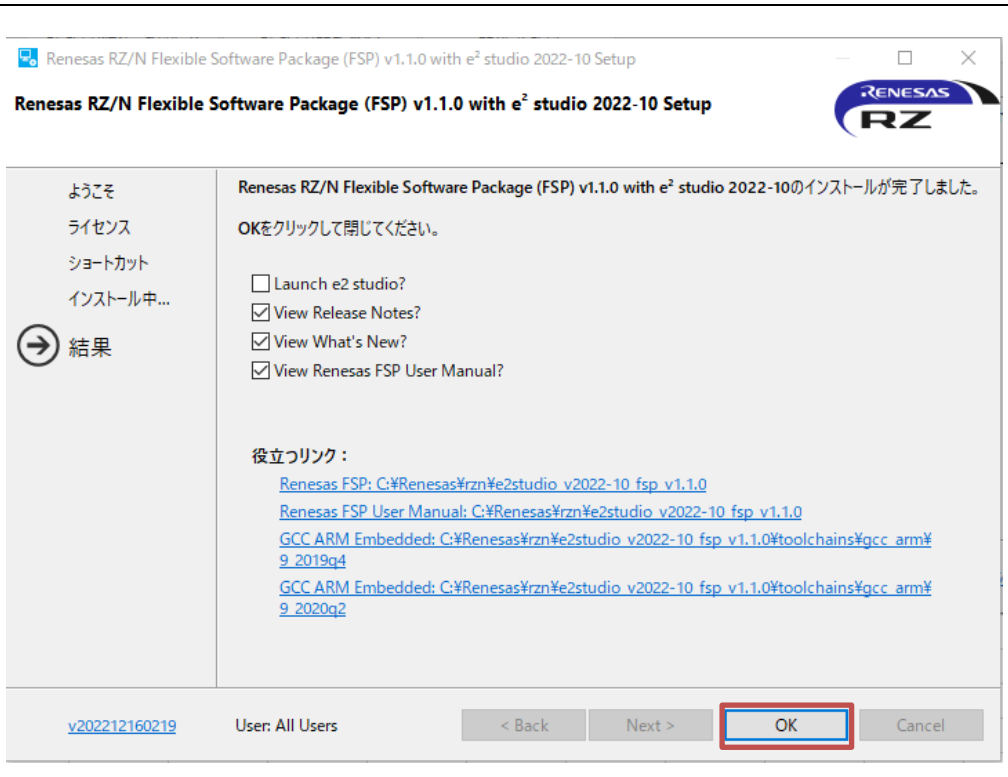


Fig.3-14 e2studio Install (8)

3.5.1.2 プロジェクト立ち上げ

(1) zip ファイル解凍

まず、アーカイブされた本サンプルソフトのパッケージ (RZN2L_OPC_BGW_V***.zip) を解凍し、任意のフォルダに格納します。e2studio はフォルダ階層が深くてフルパスが長すぎると認識できませんので、フルパスが短くなるよう配置してください。また、日本語のパスも使用しないでください。

(2) e2studio 起動

次に、e2studio を起動します。インストールされたフォルダの”e2studio.exe”を実行してください。デフォルトのインストール先は下記になります。

\Renesas\rzn\e2studio_v2022-10_fsp_v1.1.0\eclipse¥e2studio.exe

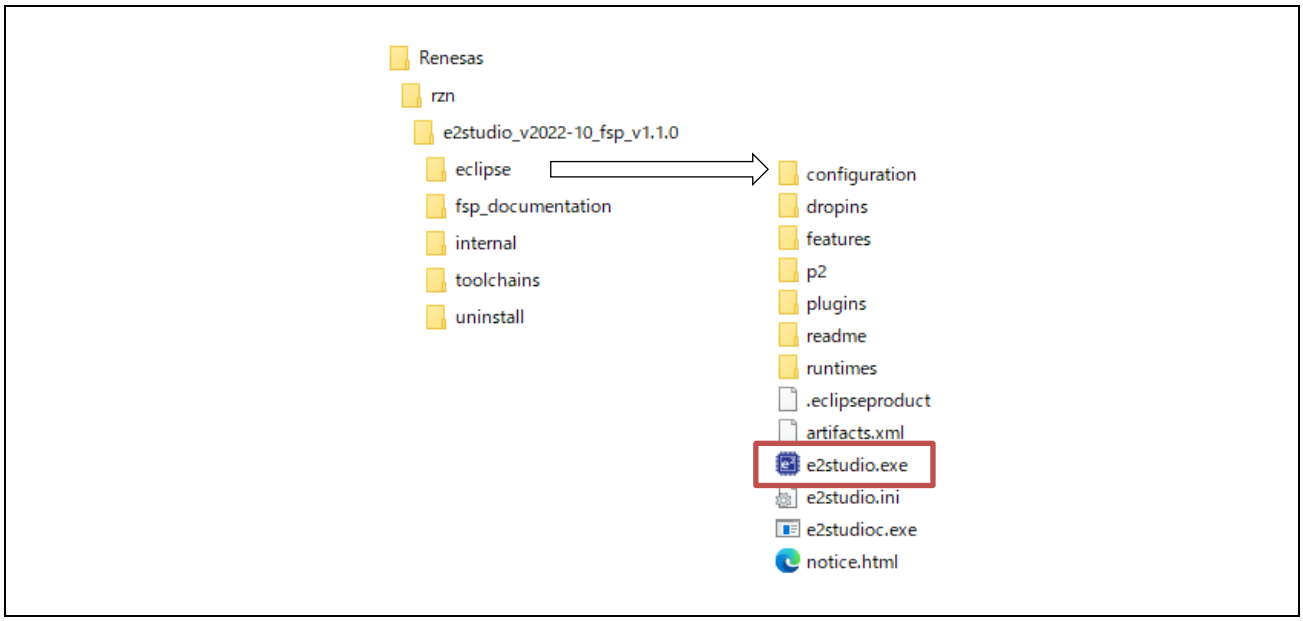


Fig.3-15 Launch project (1)

(3) プロジェクトのインポート

・任意のワークスペースディレクトリを入力して、Launch をクリックします。

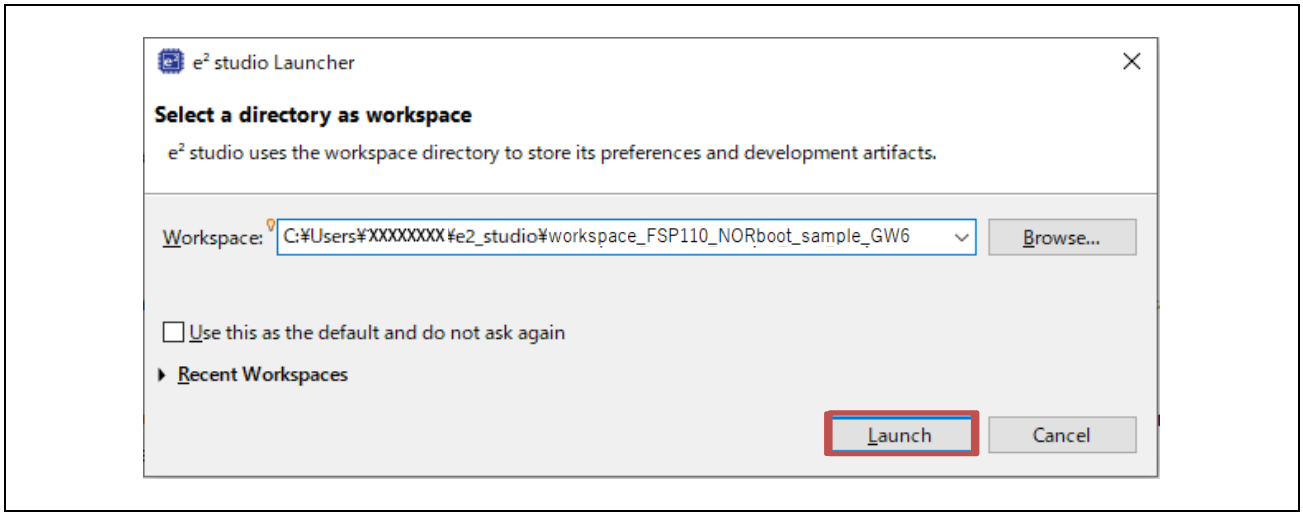


Fig.3-16 Launch project (2)

・GNU ARM Embeded – 9.3.1.20200408 ツールチェーンを選択して、登録をクリックします。

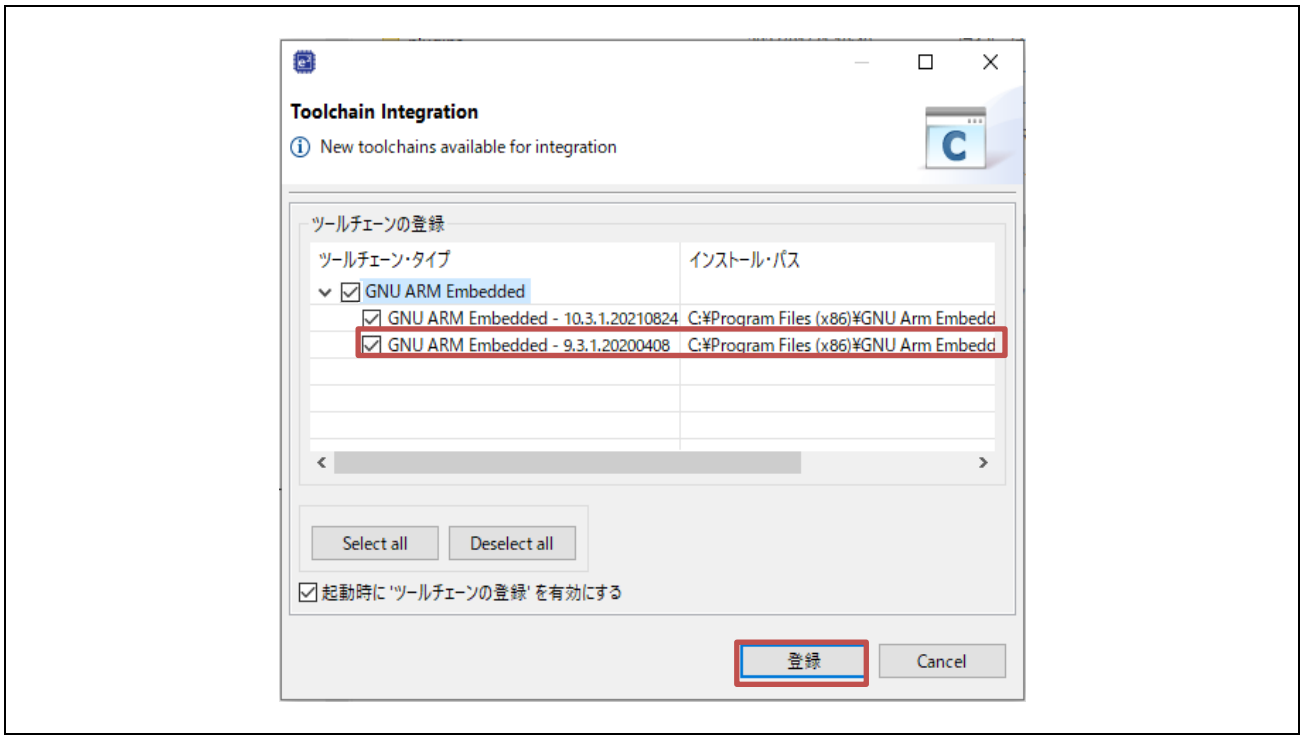


Fig.3-17 Launch project (3)

・Import existing projects をクリックします。

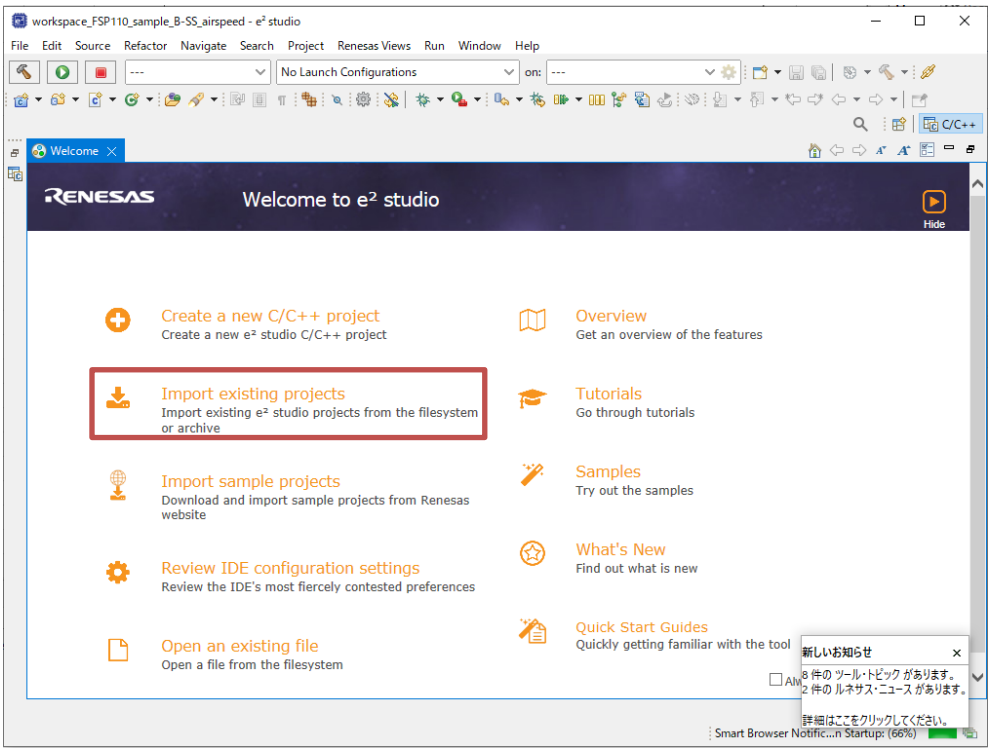


Fig.3-18 Launch project (4)

・Select root directory:の Browse をクリックして、インポートするプロジェクトフォルダを入力します。
Copy projects into workspace にチェックを入れるとインポートプロジェクトがコピーされます。

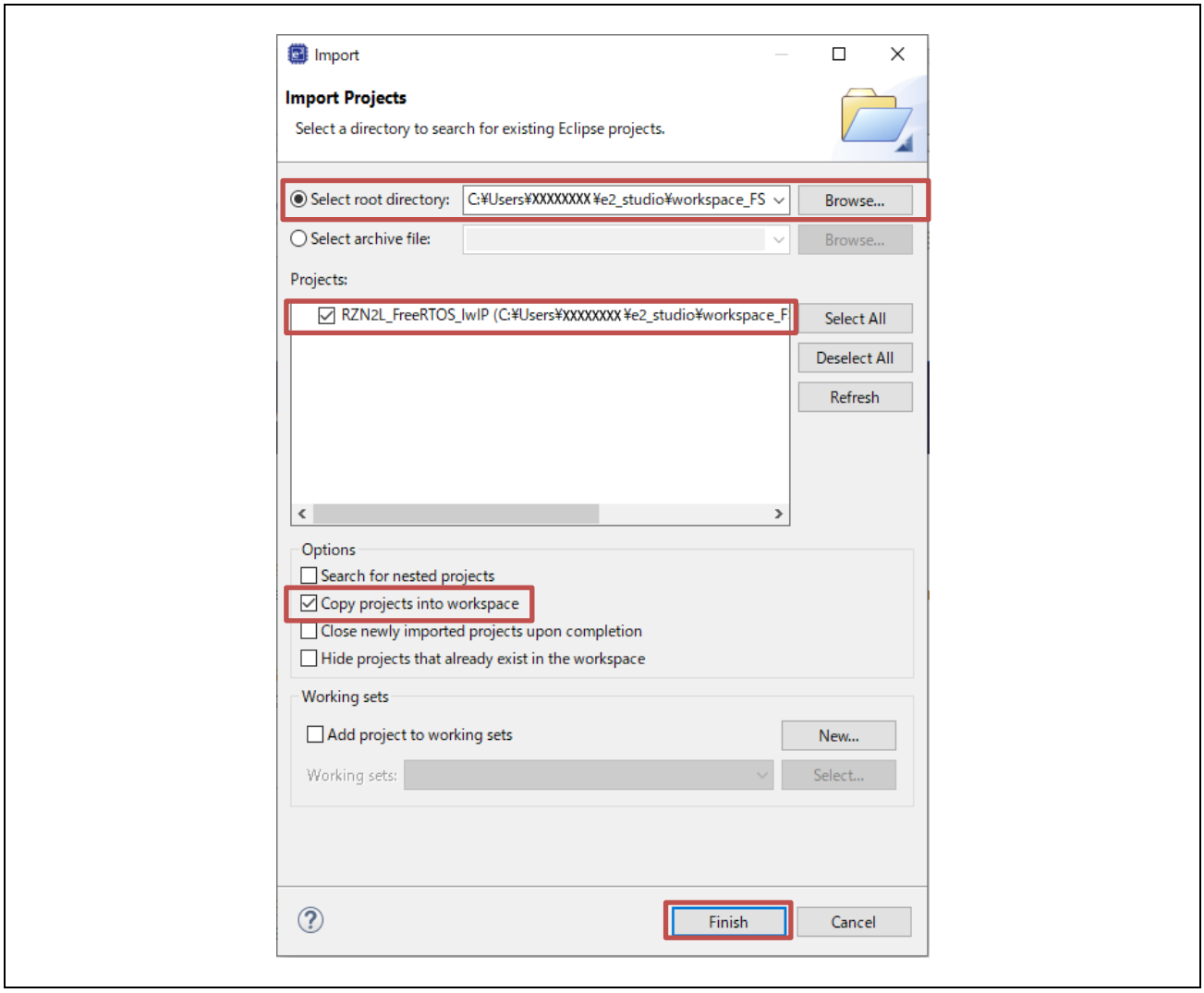


Fig.3-19 Launch project (5)

・Fig.3-19 の Finish をクリックすると以下が表示されますので Yes To All をクリックします。

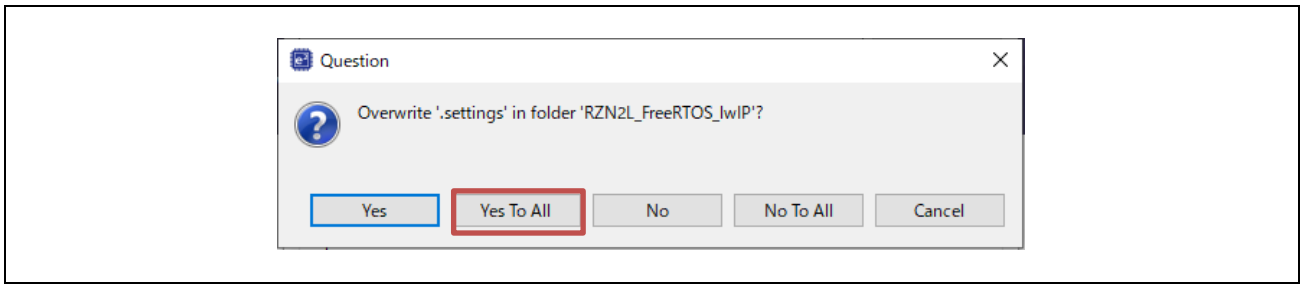


Fig.3-20 Launch project (6)

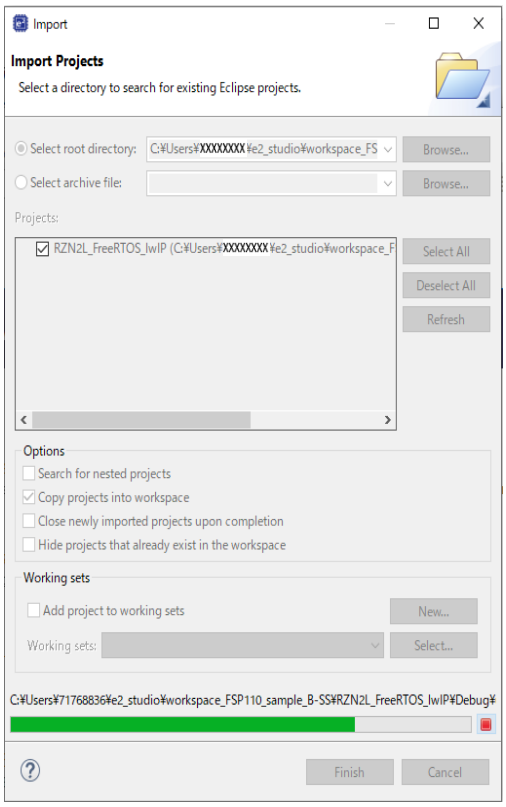


Fig.3-21 Launch project (7)

・プロジェクトのインポートが終了すると以下が表示されます。これ以降は 4.3.1 章にて説明します。

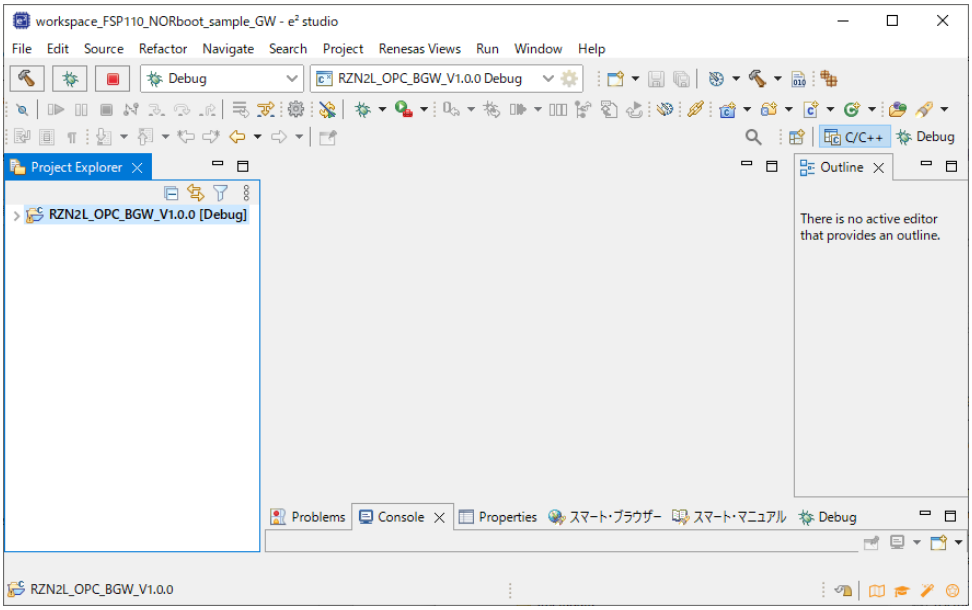


Fig.3-22 Launch project (8)

3.5.2 UaExpert

UaExpert は OPC UA クライアントツールです。ここでは、OPC UA サーバー(B-GW)と接続して BACnet /IP ネットワーク上の B-SS のオブジェクトノードにアクセスするために使います。

Table 1-1 に記載のバージョンを web サイトからダウンロードして、お使いの PC にインストールしてください。ダウンロード前に Unified Automation Web サイトに登録してアカウントをアクティブ化する必要があります。すべてのコンテンツは無料で提供されますが、このページからソフトウェアをダウンロードまたはインストールすると、Unified Automation ソフトウェア使用許諾契約 (SLA) に自動的に同意したことになります。ソフトウェアおよび情報のライセンス条件は、次のリンクを参照ください。

https://www.unified-automation.com/products/sdk-overview/licenses.html#c341

上記使用条件を確認の上、お使いください。



Fig.3-23 UaExpert

3.5.3 Wireshark

Wireshark は無償で使えるネットワークプロトコルアナライザーです。Table 1-1 のリンクから Wireshark をダウンロードしてインストールします。

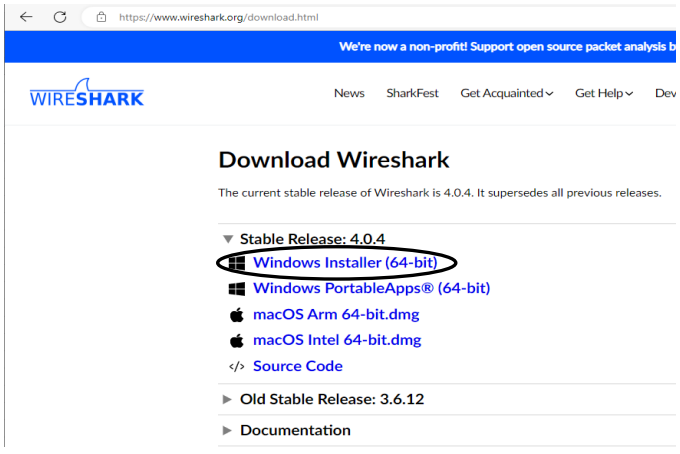


Fig.3-24 download Wireshark

4. 動作確認

4.1 接続

Fig.4-1 にサンプルソフト実行時の接続図を示します。RZ/N2L RSK ボードに Ethernet ケーブル、J-Link OB デバッガ、5V DC の各ケーブルを接続してください。図に示すとおり、B-SS 側ボードを接続する場合は同ボードの J26 コネクタに空気速度センサを接続してください。

B-GW 側ボードと B-SS 側ボードのボード設定は異なりますのでご注意ください。詳細は 2.1 章をご参照ください。B-GW 側ボードの ETH2(Ethernet2)コネクタは使用できません。RSK ボード上のデバッガ J-Link OB を使用する場合は J9 をオープンとし、USB Micro のケーブルを接続します。

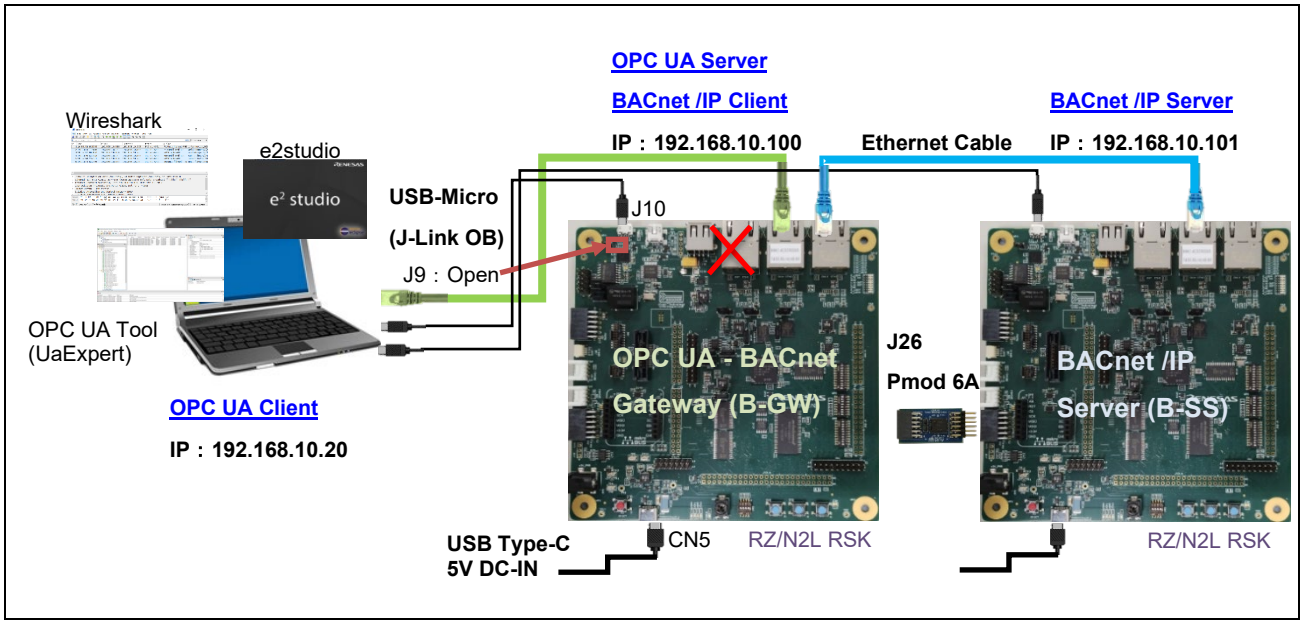


Fig.4-1 OPC UA - BACnet /IP Hardware Diagram

4.2 IP アドレス設定

OPC UA Client となる PC 側イーサネットのアドレス設定を行います。

Windows のスタートの設定をクリックします。

以下のとおりに進み、IP アドレスを設定してください。

設定＞ネットワークとインターネット＞アダプターのオプションを変更する＞イーサネット＞プロパティ＞インターネットプロトコルバージョン 4(TCP/IPv4)＞プロパティ

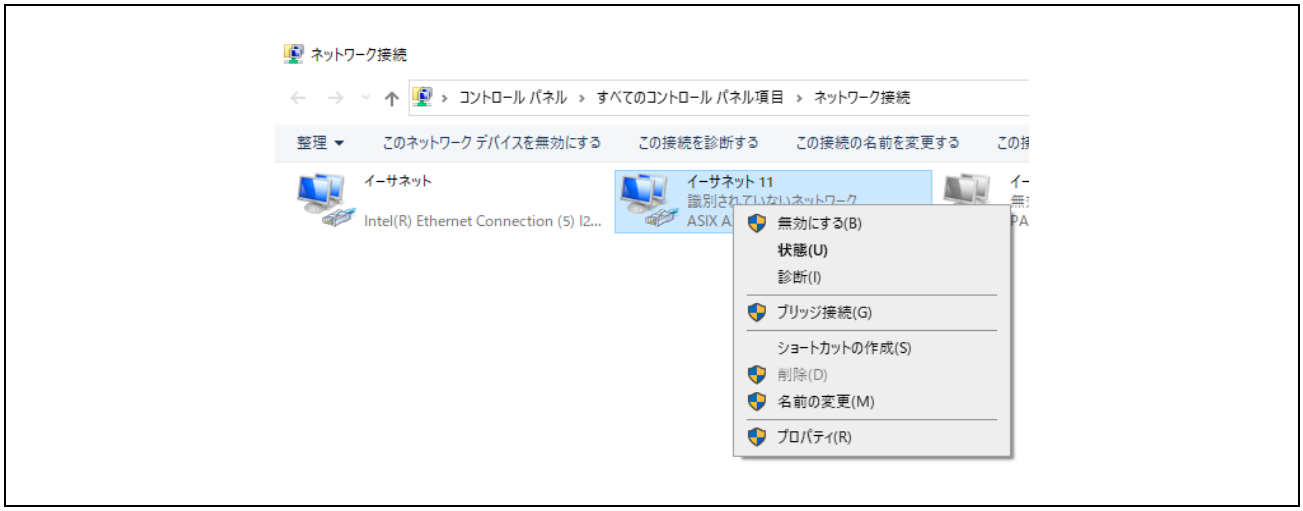


Fig.4-2 network connection

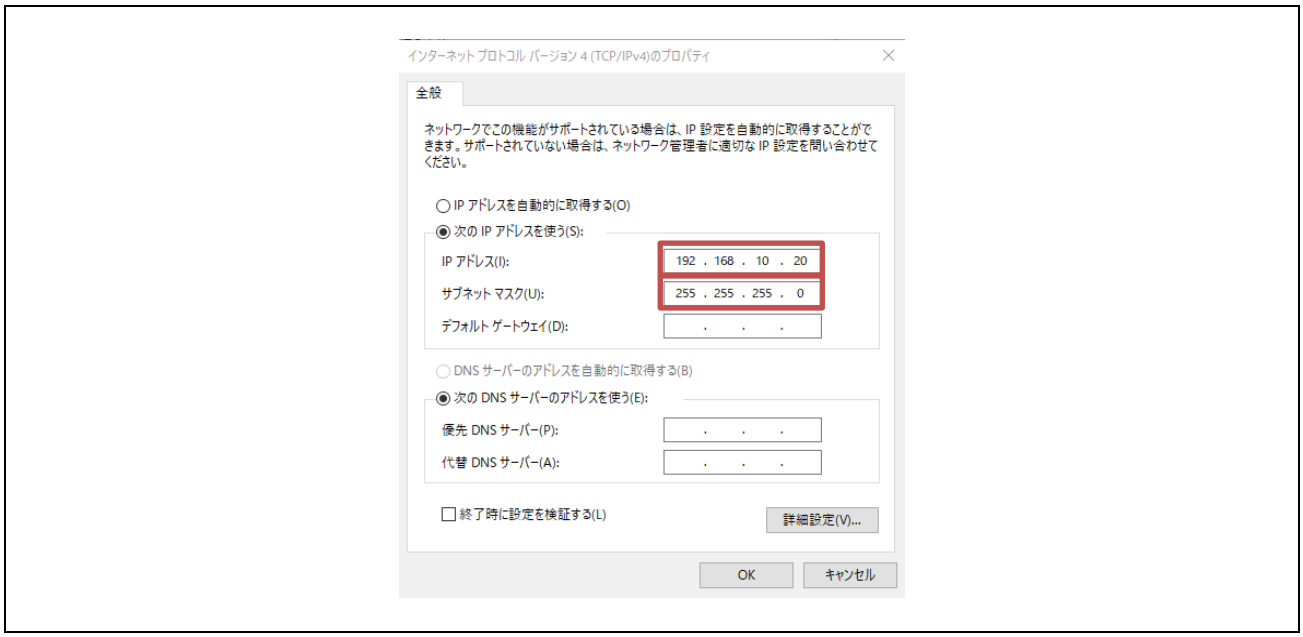


Fig.4-3 TCP/IPv4 properties

B-GW サンプルソフトで使用する RSK ボードの IP アドレスは 192.168.10.100 です。PC 側の IP アドレスを 192.168.10.XXX に設定する必要があります。本ドキュメントでは 192.168.10.20 を設定しています。

4.3 プロジェクト起動

まず、3.5.1.2 章の手順で、プロジェクトをインポートします。

4.3.1 ビルド設定

Project Explorer ウィンドウのプロジェクト名を選択したうえで、Project メニューの Properties を開きます。

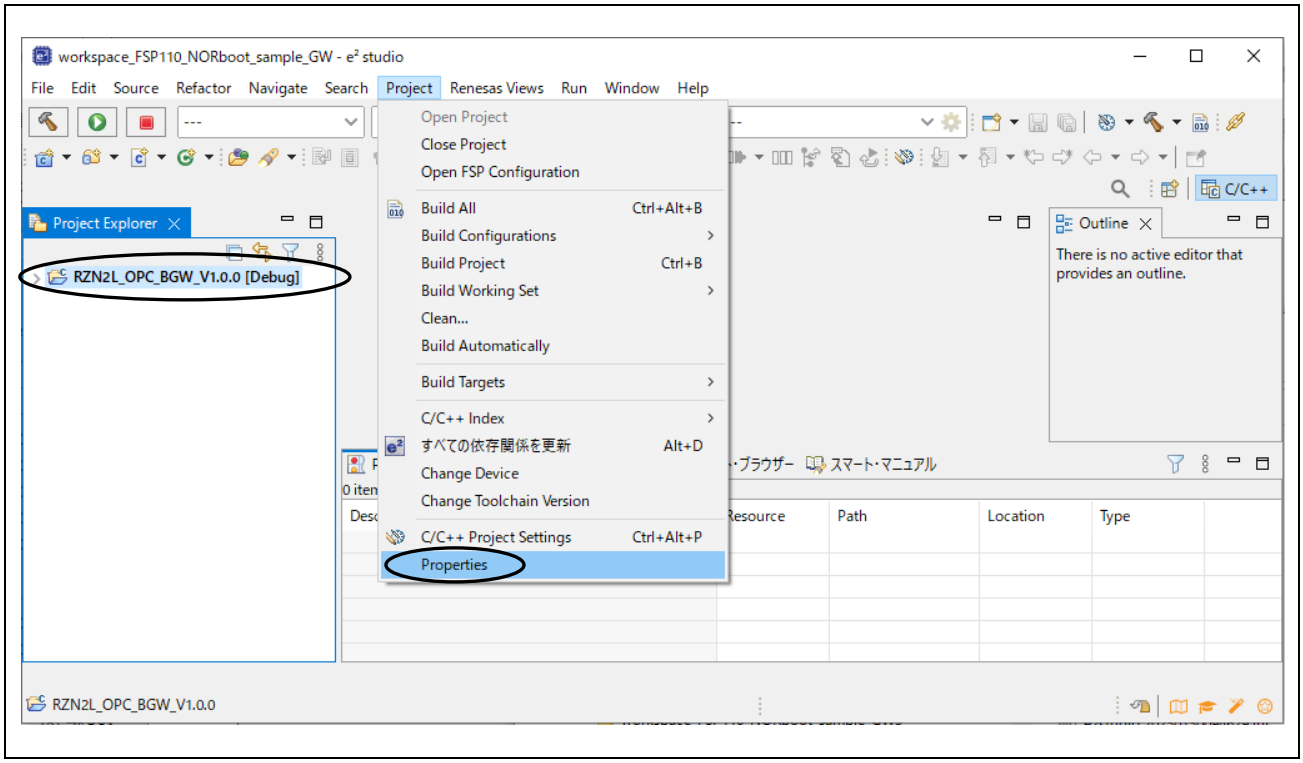


Fig.4-4 Open project properties

C/C++General > Paths and Symbols の#Symbols タグから Languages の GNU C を選択します。

(1) RSK ボード 1 枚での評価

B-SS の RSK ボードを接続せず B-GW の RSK ボード単独で評価する場合は、Symbol の **#WITHOUT_B_SS_BOARD** を **1** に変更します。変更は“Edit…” をクリックすると可能になります。

これにより本来 B-SS から読み出すセンサ入力値を B-GW 内部で疑似的に生成して、B-GW の AnalogInput,0 オブジェクトの PresentValue ノードから読み出すことができます。読み出す手順は 4.5 章で説明します。

(2) RSK ボード 2 枚での評価

B-SS の RSK ボードを接続する場合は Fig.4-5 に示す通り、Symbol の**#WITHOUT_B_SS_BOARD** を **0** に変更します。

Apply and Close をクリックし設定を適用します。ポップアップダイアログの Yes をクリックします。

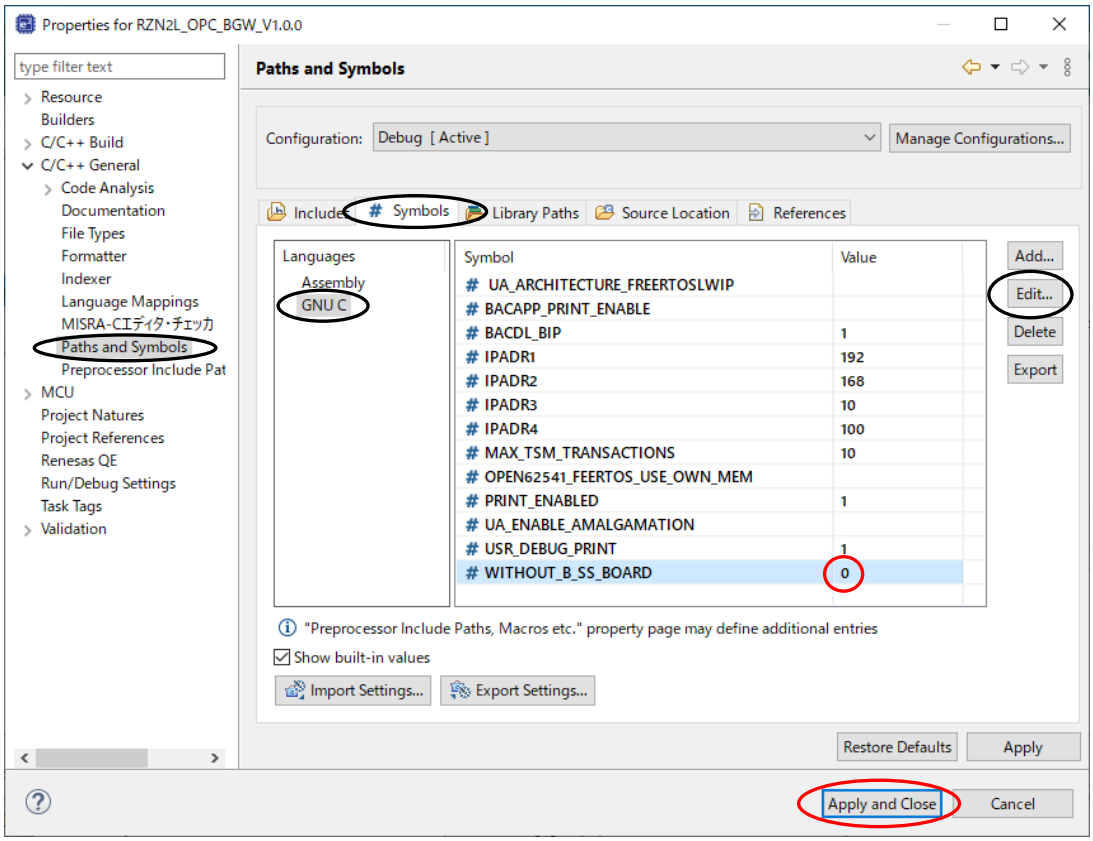


Fig.4-5 Change #WITHOUT_B_SS_BOARD

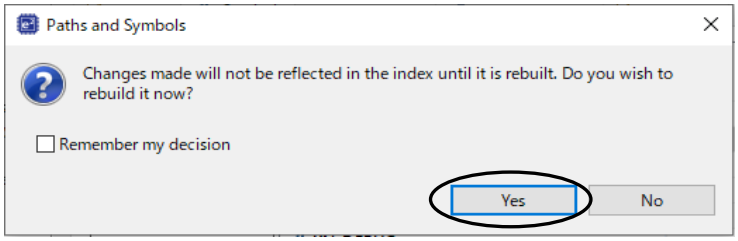


Fig.4-6 Click YES

4.3.2 ビルド

Project Explorer ウィンドウのプロジェクト名を選択したうえで、Project メニューの Clean…をクリックします。

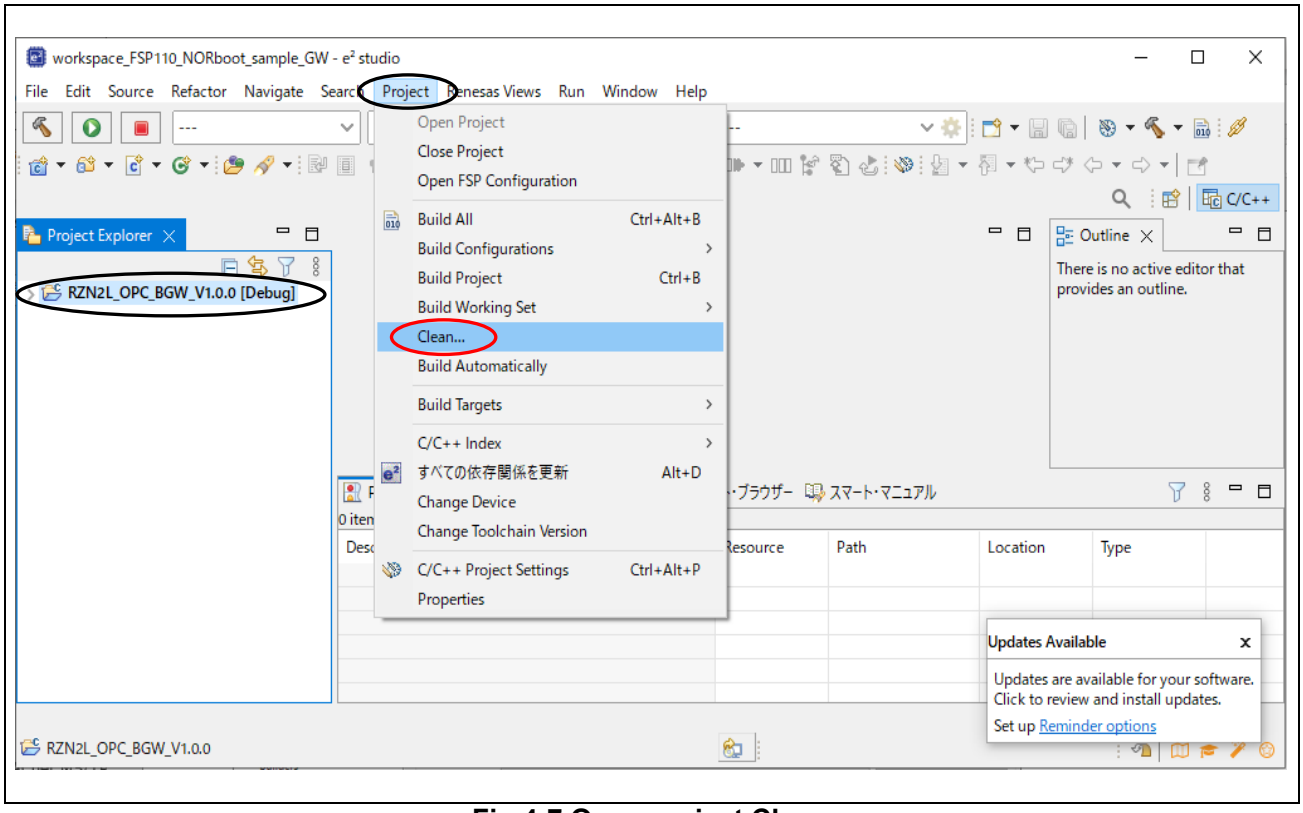


Fig.4-7 Open project Clean…

ポップアップダイアログの以下を有効にして Clean をクリックすると全ビルドを開始します。

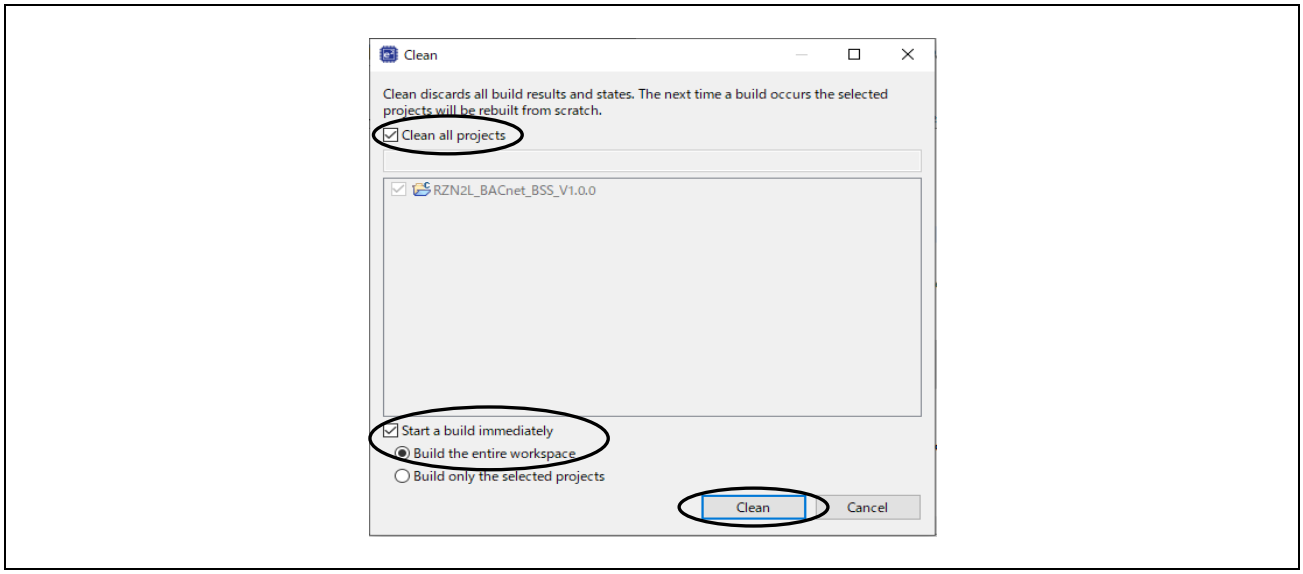


Fig.4-8 clean and rebuild

4.3.3 Debug Configurations 設定

全ビルド結果が 0 errors であることを確認後、Project Explorer ウィンドウのプロジェクト名を選択したうえで、Run メニューの Debug Configurations…をクリックします。Warning が発生しますが、無視してください。

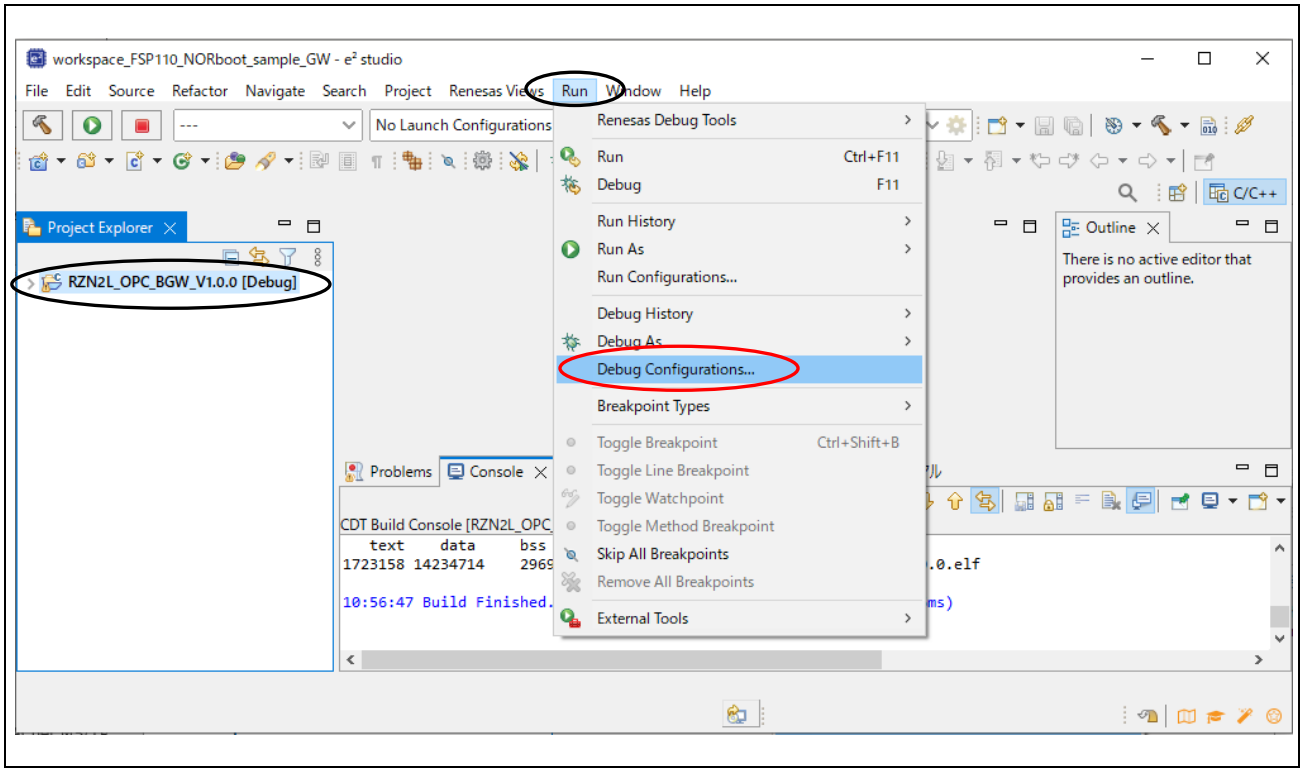


Fig.4-9 Open Debug Configurations…

- プロジェクトインポート後、初回デバッガ起動するときの操作

  プロジェクトをインポートして、初回にデバッガ起動を行うときだけ、次の操作を行ってください。

  a. RZN2L_OPC_BGW_V*** Debug[local]を生成

  b. Target Device を選択

  c. デバッグ・ツール設定

  上記は次に続く説明を参照ください。

a. RZN2L_OPC_BGW_V*** Debug[local]を生成

Renesas GDB Hardware Debugging をダブルクリックして、RZN2L_OPC_BGW_V*** Debug[local]を生成します。

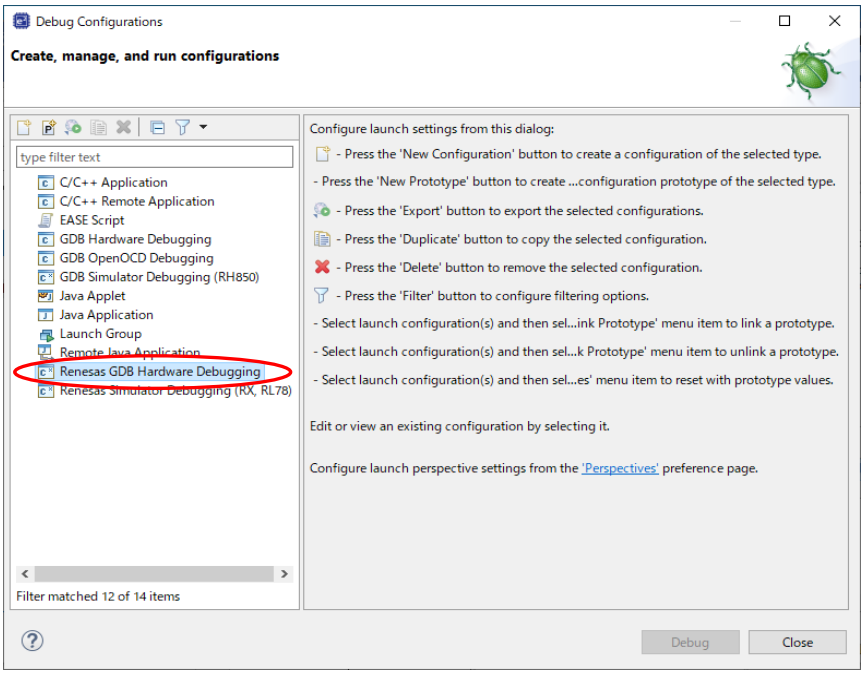


Fig.4-10 Debug Configurations(1)

b. Target Device を選択

表示されたダイアログの Debugger タグをクリックして、Target Device を選択します。

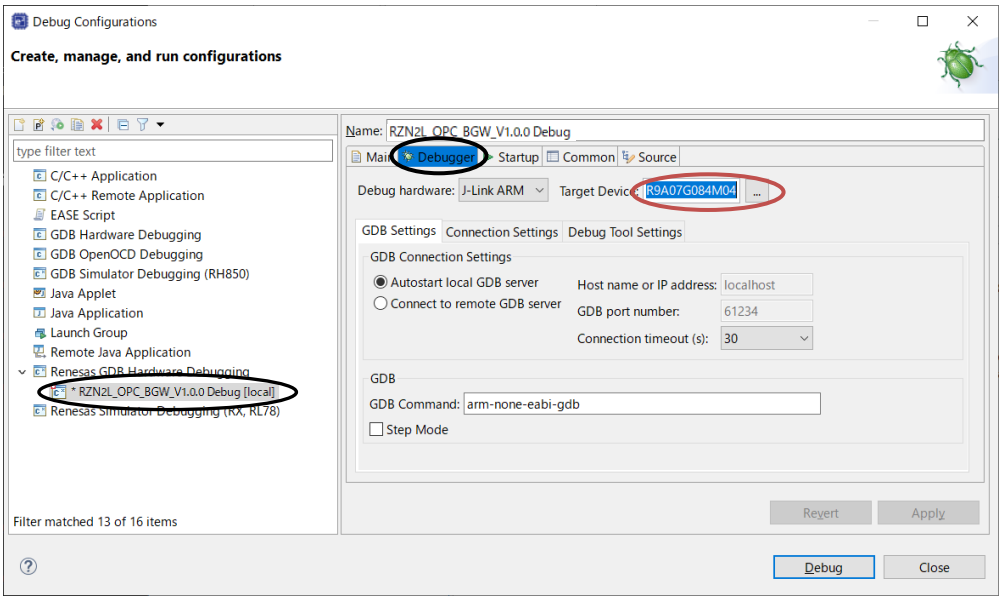


Fig.4-11 Debug Configurations(2)

R9A07G084M04 を選択して OK をクリックします。

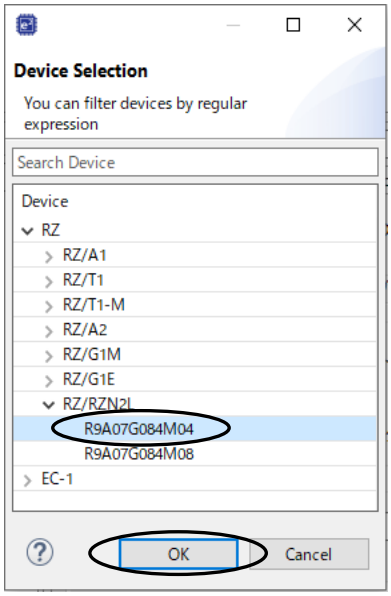


Fig.4-12 Debug Configurations(3)

c. デバッグ・ツール設定

デバッグ・ツール設定タグをクリックして、Operating Frequency [MHz]に 400 を入力します。

Debug をクリックするとダウンロードを開始します。続けて Fig.4-16 の手順を参照してください。

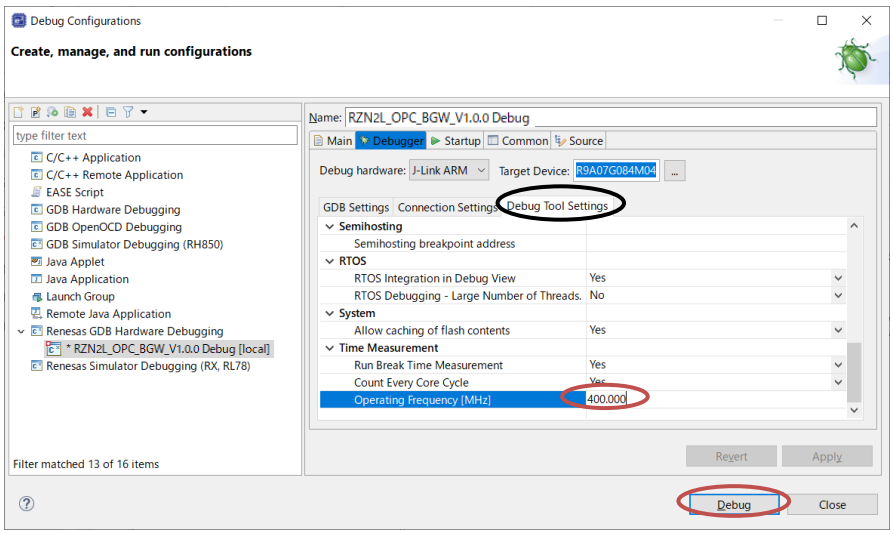


Fig.4-13 Debug Configurations(4)

4.3.4 デバッグ

ビルド終了後のダウンロード手順を以下に示します。

2 回目以降のデバッガ起動時は、C/C++ビューのプロジェクト名を選択した状態で、Run メニューをクリックします。Debug As にカーソルを置いて、Renesas GDB Hardware Debugging をクリックします。

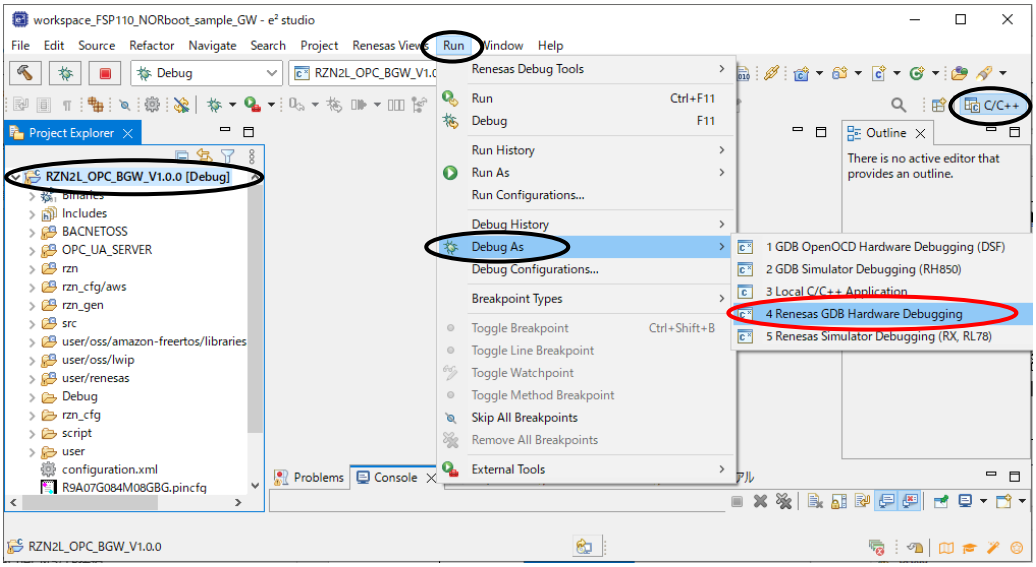


Fig.4-14 Run menu Debug As

NOR フラッシュメモリにプログラムをダウンロードします。(少し時間がかかります)

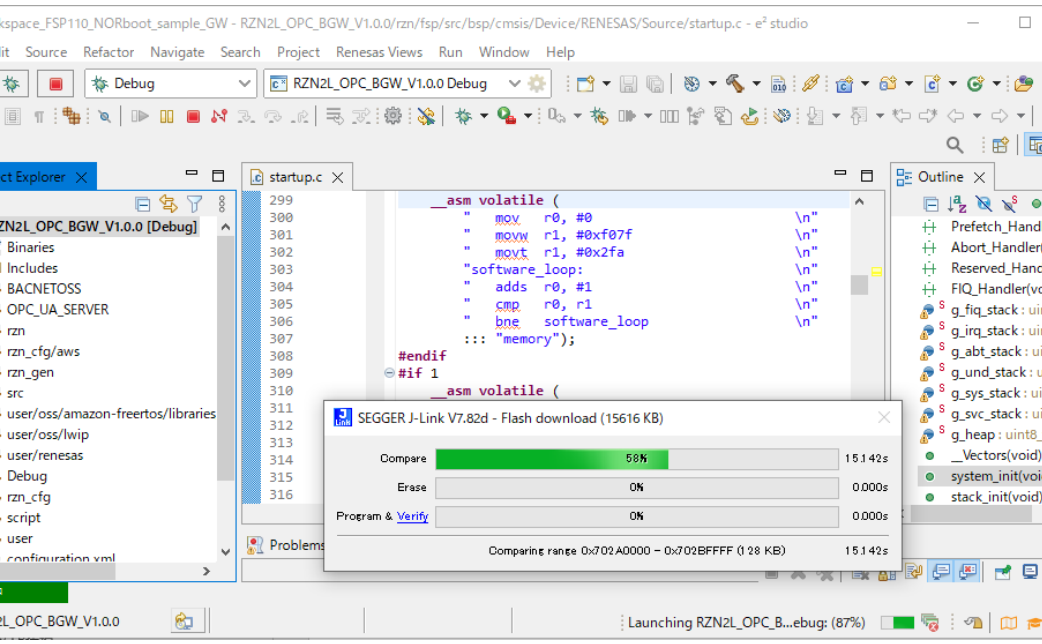


Fig.4-15 Download

・デバッグビューへ切り替えるために Switch をクリックします。

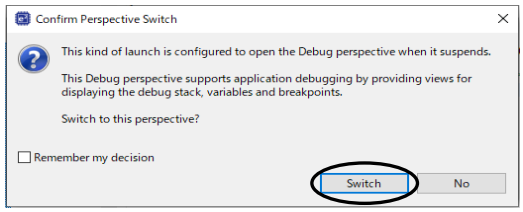


Fig.4-16 Perspective Switch

CPU はダウンロードデータに含まれるローダープログラムを自動的に BTCM メモリに展開します。展開後、ローダープログラム上の初期設定プログラム先頭にある system_init() で Break します。

※ デバッガを使用せず RSK ボード単独で動作させる場合は、ここでボードの電源を OFF し、デバッガケーブルを外してから電源を再度 ON してください。

デバッガを使用する場合は、Debug 画面に切り替わったら、まず reset アイコン をクリックしてから resume をクリックしてください。

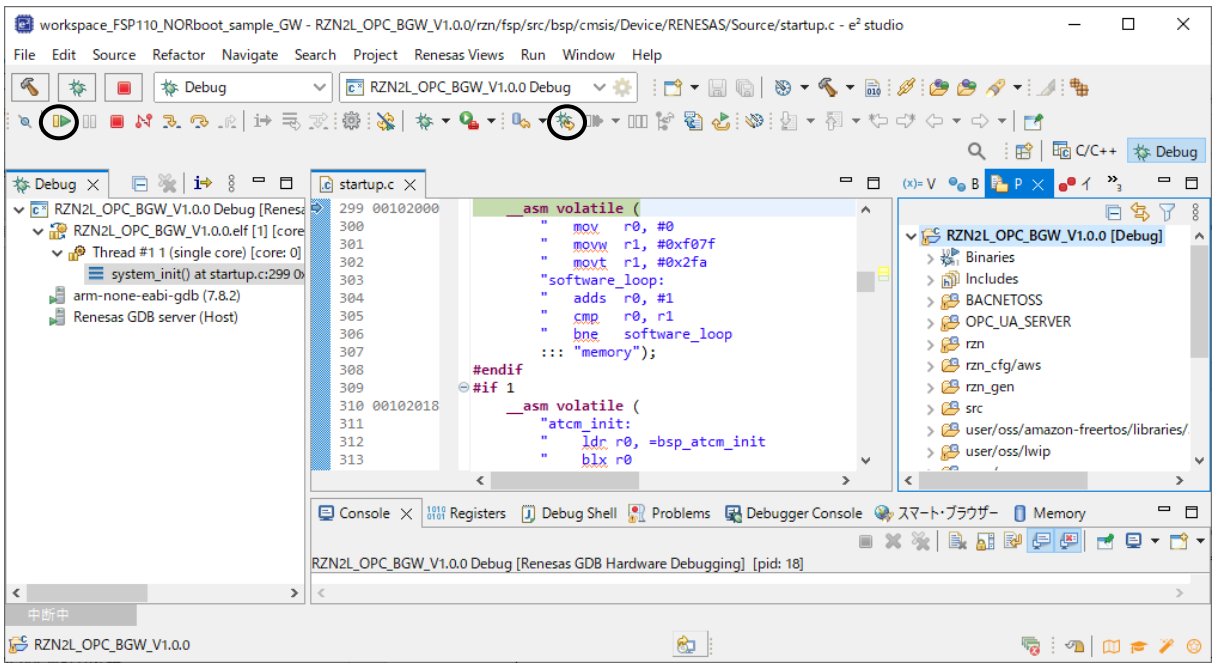


Fig.4-17 Break at system_init()

ローダープログラムは初期設定を終了すると main() の先頭で Break します。続けて resume をクリックしてプログラムを実行します。

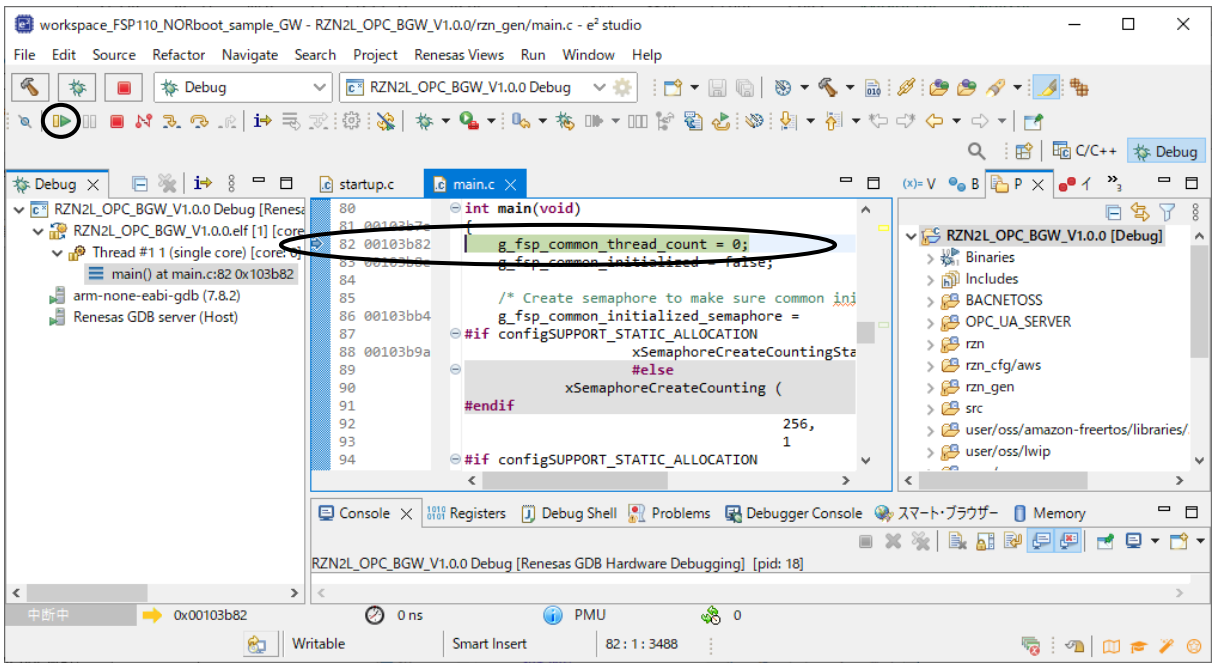


Fig.4-18 Break at main()

4.4 BACnet / OPC UA Gateway 通信確認

・UaExpert の立ち上げ

Windows のスタート を開いて UaExpert をクリックします。

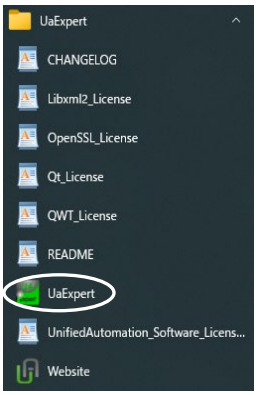


Fig.4-19 Launch UaExpert

・OPC UA サーバーの追加

UaExpert が起動したら、ツールバーの をクリックします。

Advanced タブの Endpoint Url に” opc.tcp://192.168.10.100:4840”を設定し、Anonymous を選択します。”Connect Automatically” にチェックを入れて、最後に OK をクリックします。

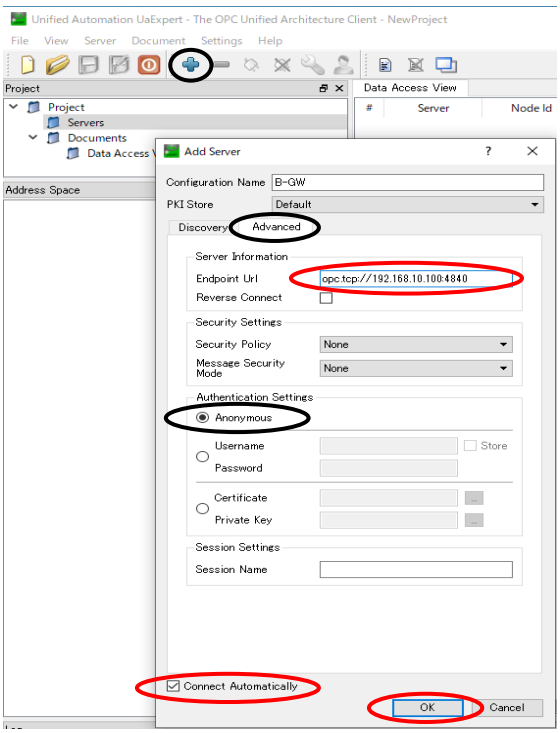


Fig.4-20 UaExpert Add server

OPC UA サーバーすなわち B-GW が接続されると Project ウィンドウの B-GW が接続されたことを示すアイコンが表示されます。Address Space ウィンドウの Object ツリーに表示された BACnet-Client-Mapping は B-GW のオブジェクトです。また、その下に BACnet-Server-Mapping というオブジェクトが現れ、B-GW に繋がっている BACnet デバイスのオブジェクトノードにアクセスします。

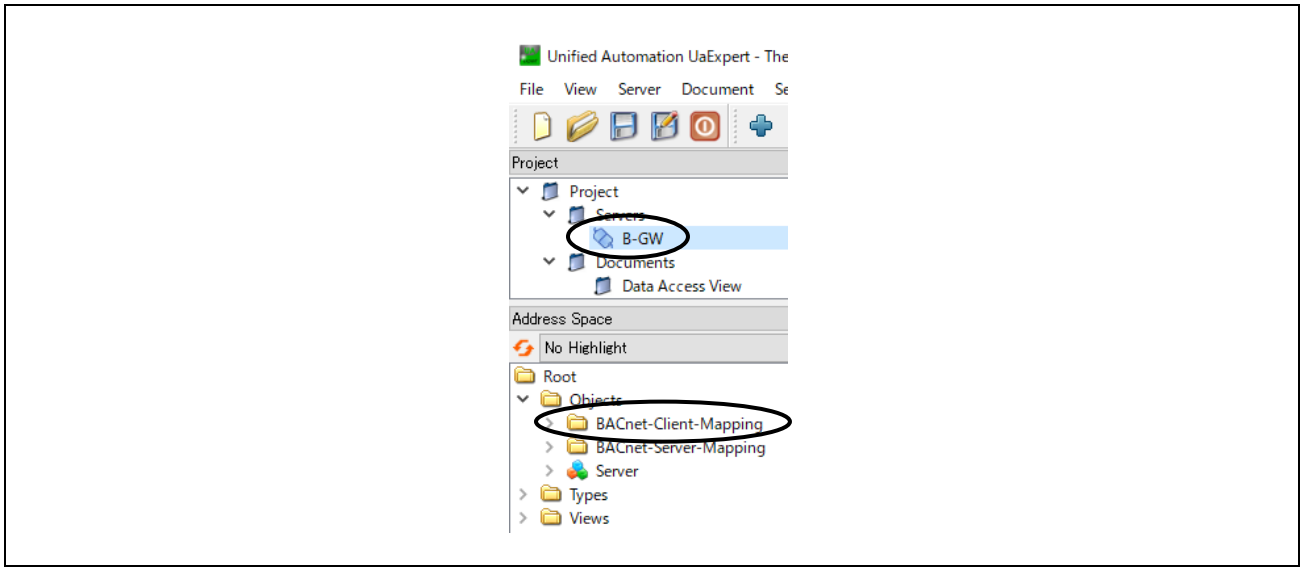


Fig.4-21 UaExpert OPC UA server connection

4.4.1 TimeSynchronization メソッド

TimeSynchronization メソッドは B-GW に対して UTC 時刻を設定します。設定時刻は B-GW 内部でタイムスタンプに反映されます。取得した UTC 時刻をローカル時刻に補正後、BACnet サーバー機器へローカルブロードキャストで転送します。

Address Space ウィンドウの

Root>Objects>BACnet-Client-Mapping>OBJECT_INTERNETWORKTYPE>TimeSynchronization を右クリックし、Call…を選択クリックします。

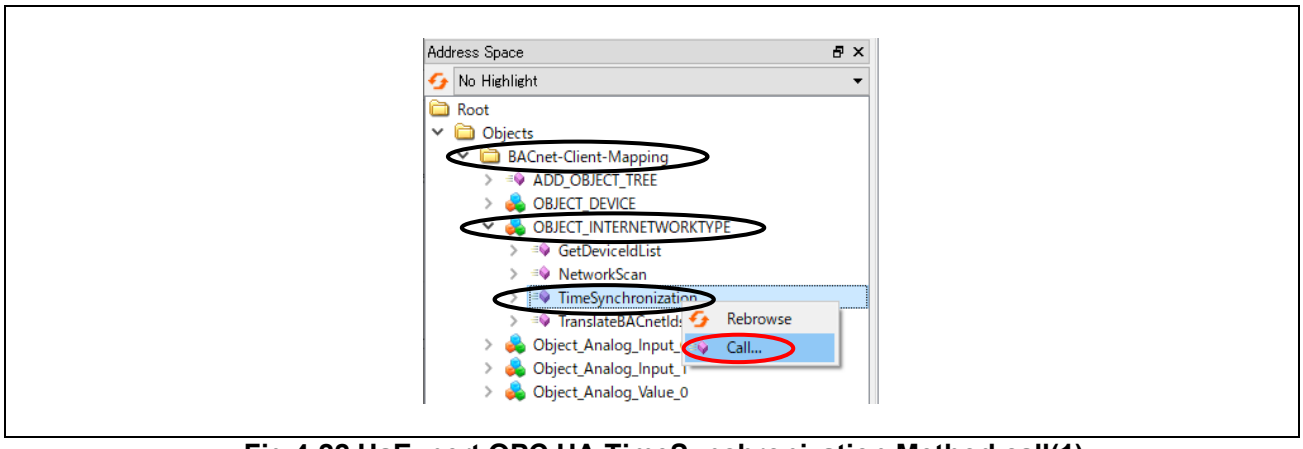


Fig.4-22 UaExpert OPC UA TimeSynchronization Method call(1)

表示されるダイアログに UTC 時刻を設定し、Call をクリックします。

UTC 時刻はローカル時刻にタイムゾーン時間を補正してください。例として TOKYO JAPAN の場合はローカル時刻から 9 時間を減算した結果が UTC 時刻です。

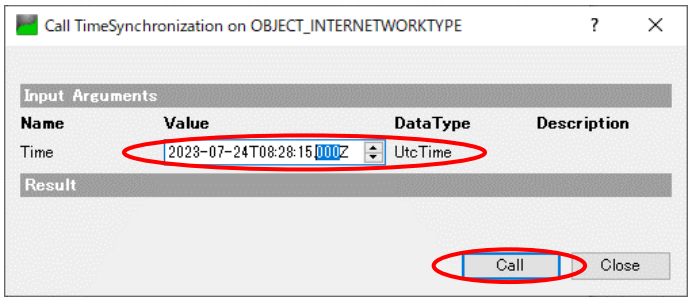


Fig.4-23 UaExpert OPC UA TimeSynchronization Method call(2)

メソッドが正常終了することを確認して Close をクリックします。

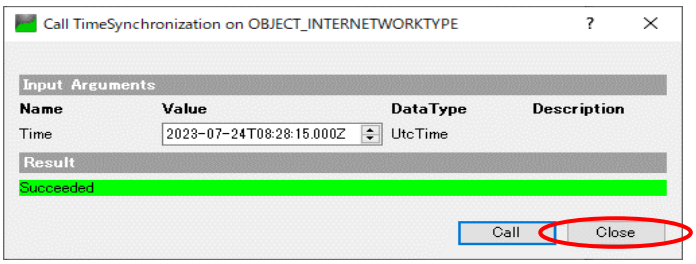


Fig.4-24 UaExpert OPC UA TimeSynchronization Method call(3)

次の wireshark ログは上記のメソッド CallRequest と CallResResponse を示しています。

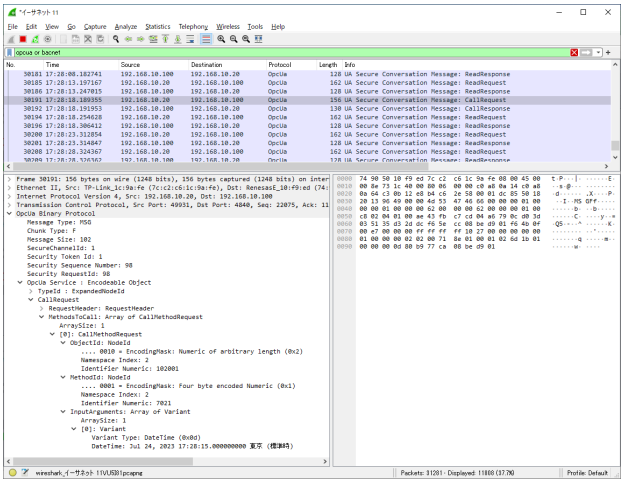


Fig.4-25 UaExpert OPC UA TimeSynchronization Method call(4)

4.4.2 NetworkScan メソッド

NetworkScan メソッドは B-GW の属するネットワークに接続する他のデバイスについて IP アドレスとデバイスインスタンス番号を取得します。

Address Space ウィンドウの

Root>Objects>BACnet-Client-Mapping>OBJECT_INTERNETWORKTYPE>NetworkScan を右クリックし、Call…を選択クリックします。

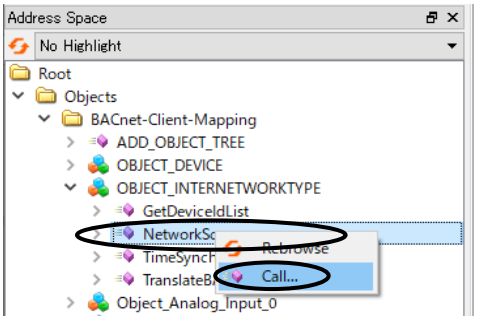


Fig.4-26 UaExpert OPC UA NetworkScan Method call(1)

表示されるダイアログに次を設定します。

- ・Wait TimeInSeconds： 他のデバイスからの I-Am 応答待ち時間を秒単位で設定します。
- ・ApplyRange： 接続デバイスの探索範囲を有効☑/無効☐にします。
  無効の場合は全デバイスインスタンス範囲 0~ 4194303 を探索します。
- ・DeviceRangeLow： 探索範囲を有効にした場合、接続デバイスの最小インスタンス番号を設定します。
- ・DeviceRangeHigh： 探索範囲を有効にした場合、接続デバイスの最大インスタンス番号を設定します。

上記を設定後、Call をクリックします。

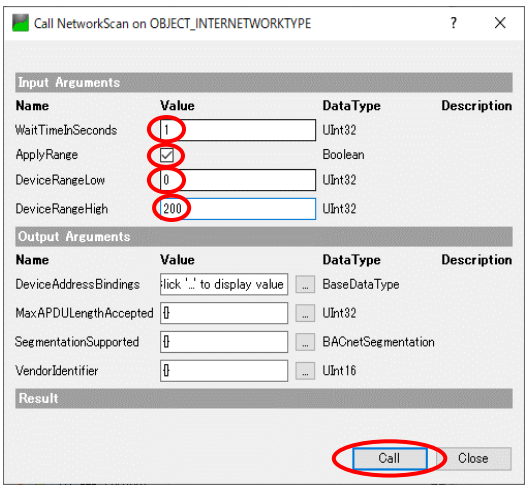


Fig.4-27 UaExpert OPC UA NetworkScan Method call(2)

メソッドが正常終了することを確認して Output Arguments の DeviceAddressBindings [...] をクリックします。例では B-SS を検出し、接続デバイス IP：192.168.10.101、デバイスインスタンス番号：100 を示しています。

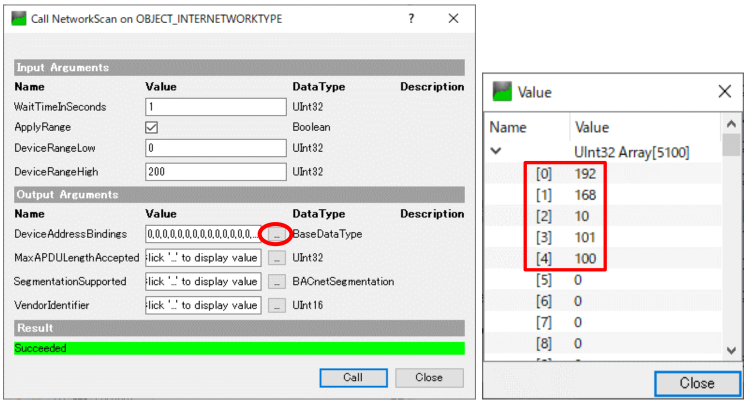


Fig.4-28 UaExpert OPC UA NetworkScan Method call(3)

次の wireshark ログは上記のメソッド CallRequest と CallResResponse を示しています。

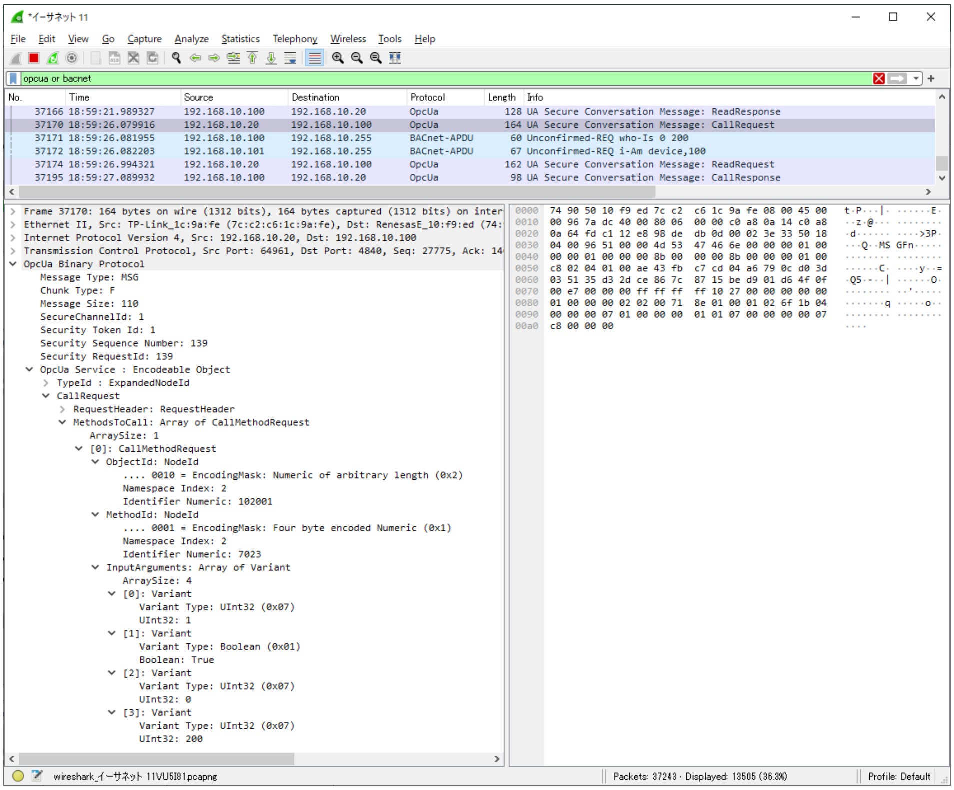


Fig.4-29 UaExpert OPC UA TimeSynchronization Method call(4)

4.4.3 Write property メソッド

Write property メソッドは B-SS デバイスオブジェクトのプロパティ値を変更します。

Address Space ウィンドウの *Root>Objects>BACnet-Server-Mapping>Write property* を右クリックし、Call…を選択クリックします。

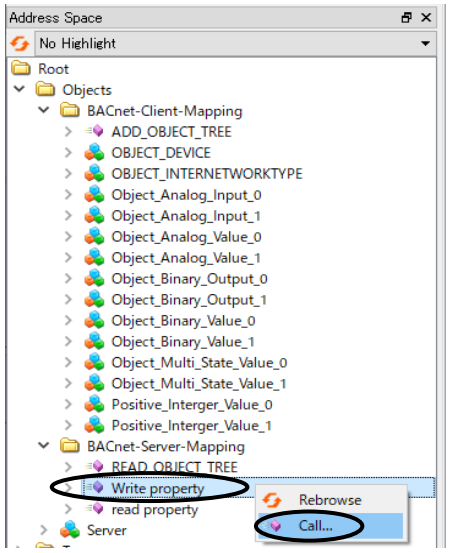


Fig.4-30 UaExpert OPC UA Write property Method call(1)

表示されるダイアログに次を設定します。

Table 4-1 Write property Method Input Arguments(1)

Input Arguments	Property	Object_Type					
		AnalogInput	AnalogValue	BinaryOutput	BinaryValue	MultiStateValue	PositiveIntegerValue
DEVICE_ID		100					
OBJECT_TYPE		0	2	4	5	19	48
OBJECT_INSTANCE		0 or 1	0 or 1	0 or 1 or 2 or 3	0 or 1	0 or 1	0 or 1
PROPERTY_ID	Present_Value	85	85	85	85	85	85
PRIORITY		1~16					
TAG		4	4	9	9	2	2
OBJECT_VALUE		0.0~	0.0~	0 or 1	0 or 1	1 or 2 or 3	0~4294967295

Table 4-2 Write property Method Input Arguments(2)

Input Arguments	Property	Object_Type	
		Device	
DEVICE_ID		100	
OBJECT_TYPE		8	
OBJECT_INSTANCE		100	
PROPERTY_ID	Apdu_Timeout	11	
	Number_Of_Apdu_Retries		73
PRIORITY		1~16	
TAG		2	
OBJECT_VALUE (Recommended value)		1~ (6000)	0~ (3)

DEVICE_ID：B-SS のデバイスインスタンス番号です。

OBJECT_TYPE：入力値は BACNETOSS\bacnet\bacenum.h に定義された値です。

OBJECT_INSTANCE：各オブジェクトのインスタンス番号です。

PROPERTY_ID：入力値は BACNETOSS\bacnet\bacenum.h に定義された値です。

PRIORITY：複数のクライアント間で同一プロパティを書き込むときの優先度です。16 は最低優先度で、1 が最高優先度です。

TAG：プロパティ値のデータ型で BACNETOSS\bacnet\bacenum.h に定義された値です。

OBJECT_VALUE：プロパティへの設定値です。

以下の例では B-SS デバイス 100、AnalogOutput,0 オブジェクト、Present_Value プロパティ、優先度 16、データ型 Enumerated、設定値 Active(1)を書き込んでいます。

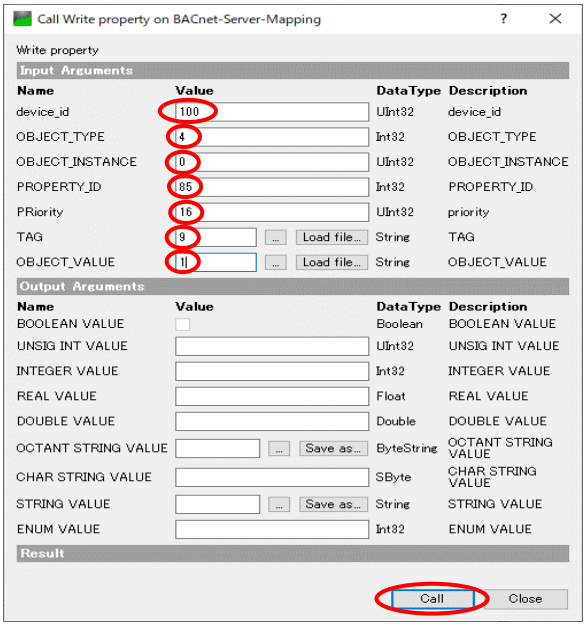


Fig.4-31 UaExpert OPC UA Write property Method call(2)

次のとおり正常終了と Output Arguments を確認します。

Readback 値が設定値と同じデータ型 Enumerated の、Active(1)を表しています。

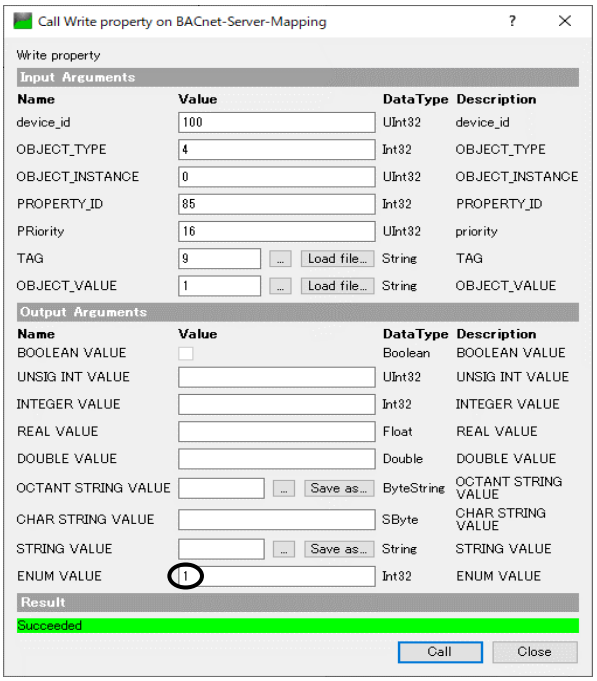


Fig.4-32 UaExpert OPC UA Write property Method call(3)

次の wireshark ログは上記のメソッド CallRequest と CallResResponse を示しています。

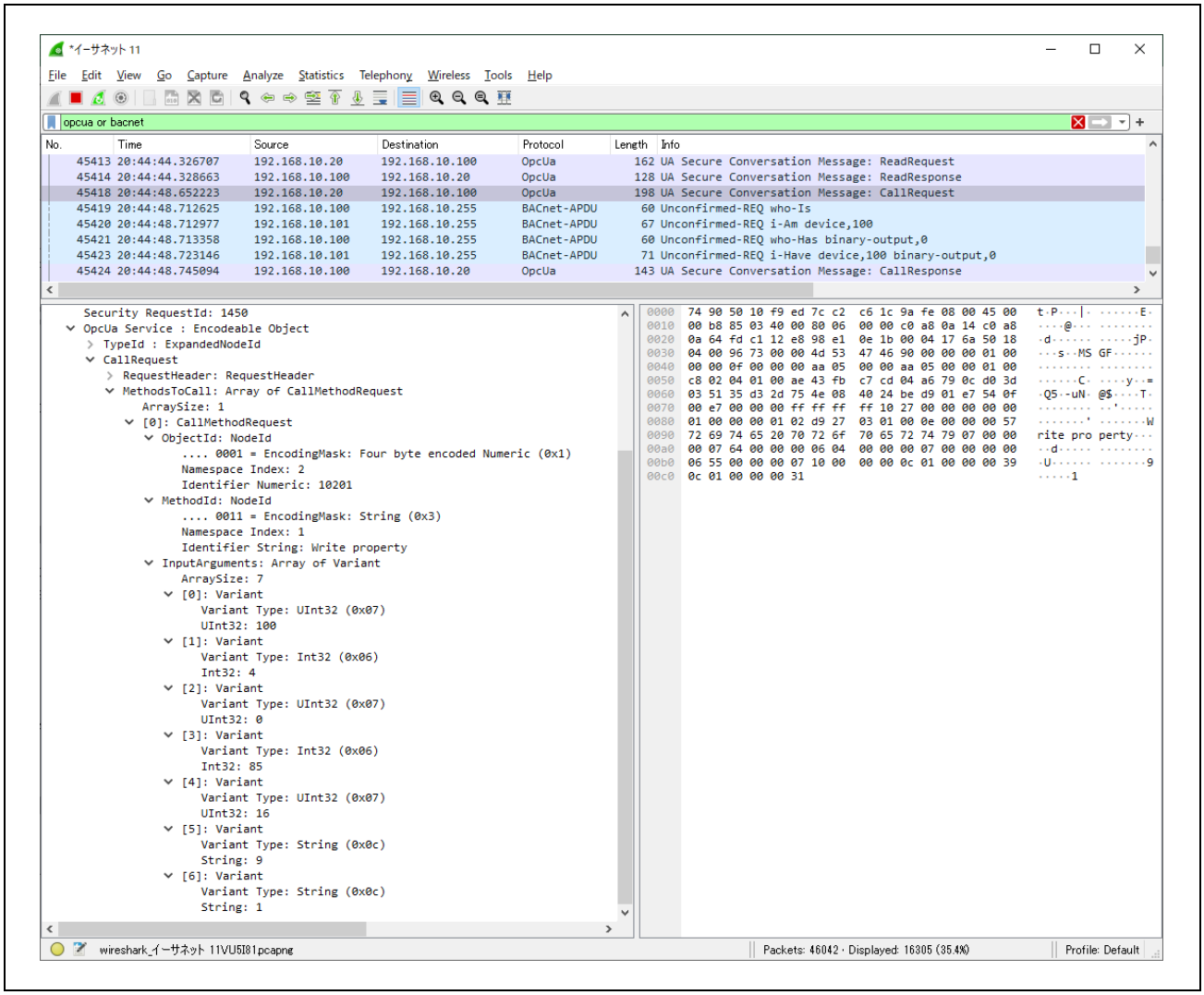


Fig.4-33 UaExpert OPC UA Write property Method call(4)

4.4.4 Read property メソッド

Read property メソッドは B-SS デバイスオブジェクトのプロパティ値を読み出します。

Address Space ウィンドウの

Root>Objects>BACnet-Server-Mapping>Read property を右クリックし、Call…を選択クリックします。

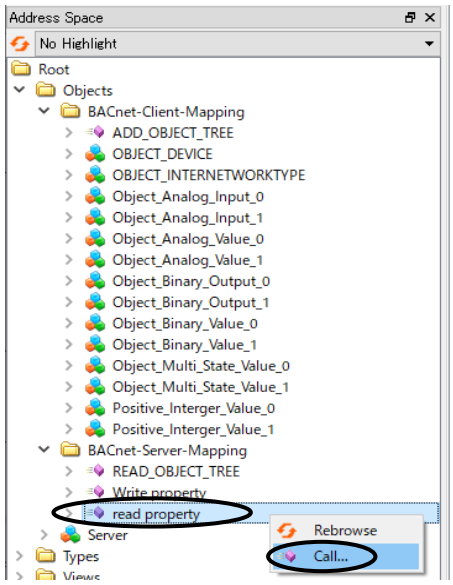


Fig.4-34 UaExpert OPC UA Read property Method call(1)

表示されるダイアログに次を設定します。

Table 4-3 Reaed property Method Input Arguments(1)

Input Arguments	Property	Object_Type					
		AnalogInput	AnalogValue	BinaryOutput	BinaryValue	MultiStateValue	PositiveIntegerValue
DEVICE_ID		100					
OBJECT_TYPE		0	2	4	5	19	48
OBJECT_INSTANCE		0 or 1	0 or 1	0 or 1 or 2 or 3	0 or 1	0 or 1	0 or 1
PROPERTY_ID	Present_Value	85	85	85	85	85	85

Table 4-4 Reaed property Method Input Arguments(2)

Input Arguments	Property	Object_Type	
		Device	
DEVICE_ID		100	
OBJECT_TYPE		8	
OBJECT_INSTANCE		100	
PROPERTY_ID	Apdu_Timeout	11	
	Number_Of_Apdu_Retries		73

DEVICE_ID：B-SS のデバイスインスタンス番号です。

OBJECT_TYPE：入力値は BACNETOSS\bacnet\bacenum.h に定義された値です。

OBJECT_INSTANCE：各オブジェクトのインスタンス番号です。

PROPERTY_ID：入力値は BACNETOSS\bacnet\bacenum.h に定義された値です。

以下の例では B-SS デバイス 100、AnalogInput,0 オブジェクト、Present_Value プロパティを読み出しています。

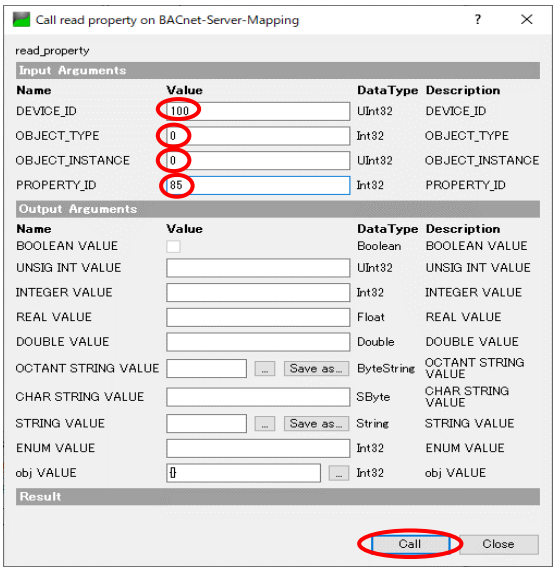


Fig.4-35 UaExpert OPC UA Read property Method call(2)

次のとおり正常終了と Output Arguments を確認します。

Readback 値が設定値と同じデータ型 REAL_VALUE(float)の、読み出し値を表しています。

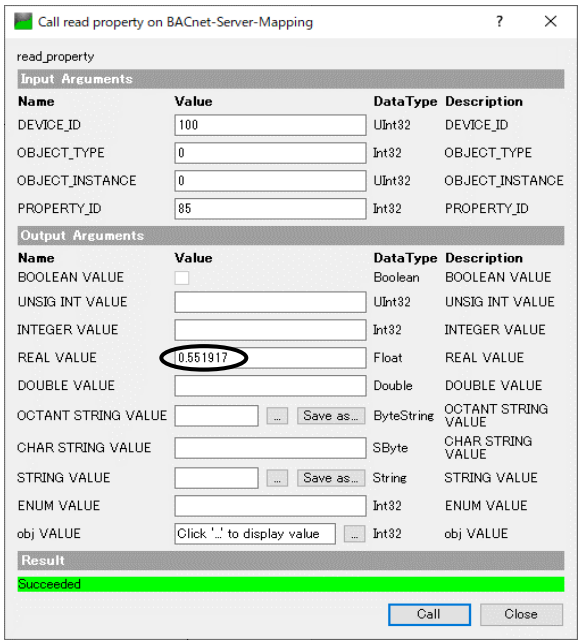


Fig.4-36 UaExpert OPC UA Read property Method call(3)

次の wireshark ログは上記のメソッド CallRequest と CallResResponse を示しています。

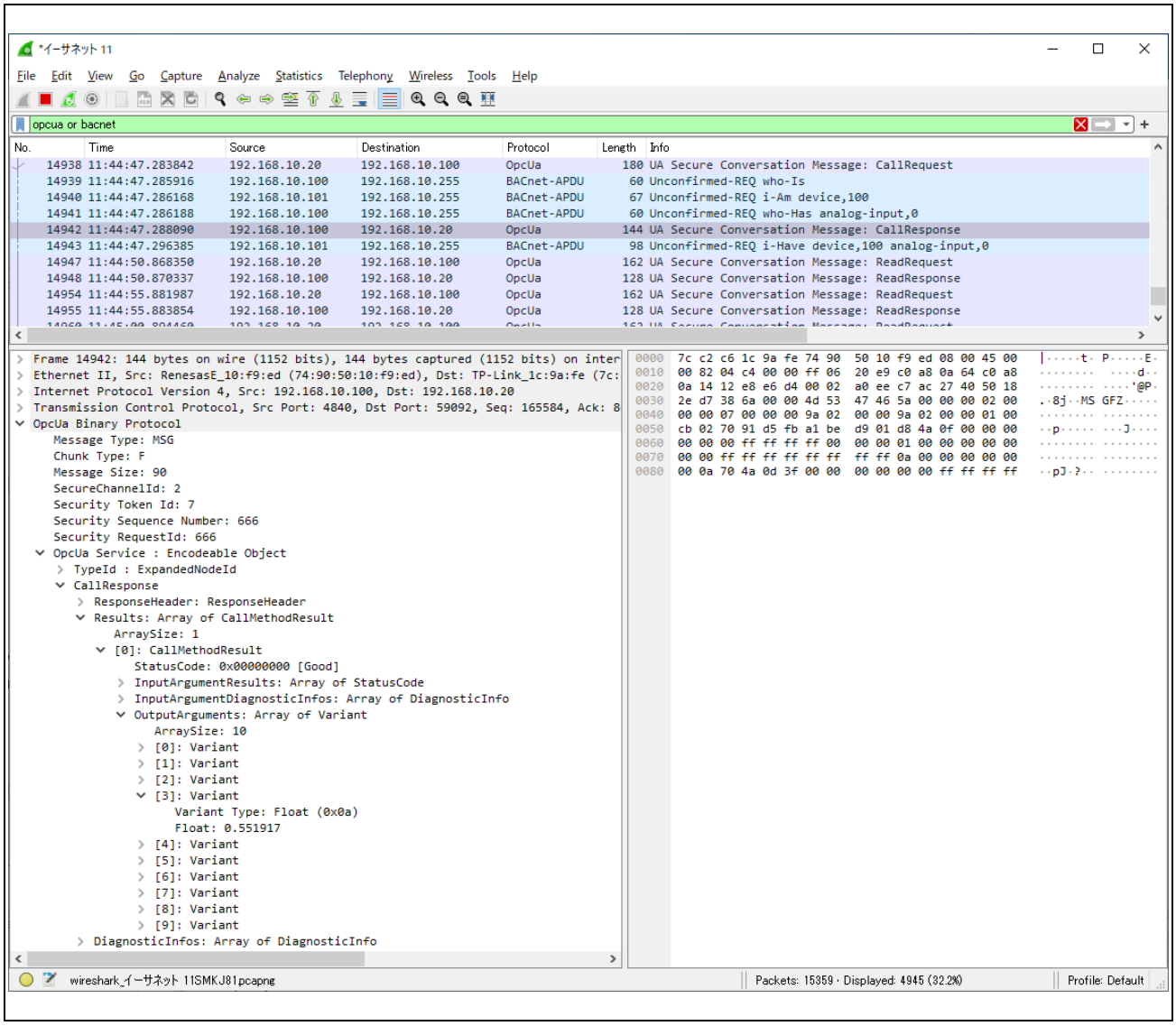


Fig.4-37 UaExpert OPC UA Read property Method call(4)

4.4.5 ADD/READ_OBJECT_TREE メソッド

ADD_OBJECT_TREE メソッドは B-SS デバイスのオブジェクトツリーを生成します。

Read property メソッドのようにオブジェクトごとの読み出しではなく複数オブジェクトを一括読み出しするために対象となるオブジェクトをツリーに追加します。

一括読み出し対象のオブジェクトタイプは次のとおりです。

AnalogInput、BinaryOutput、MultiStateValue、AnalogValue、BinaryValue、PositiveIntegerValue

・制限事項

次のオブジェクトタイプは一括読み出しの対象に含まれません。

Device、AnalogOutput、BinaryInput

Address Space ウィンドウの

Root>Objects>BACnet-Client-Mapping>ADD_OBJECT_TREE を右クリックし、Call…を選択クリックします。

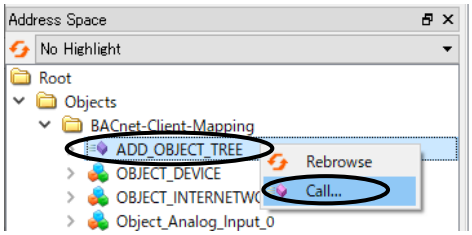


Fig.4-38 UaExpert OPC UA ADD_OBJECT_TREE Method call(1)

表示されるダイアログに次を設定します。最後に Call をクリックします。

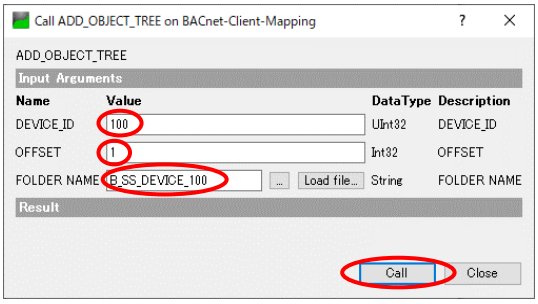


Fig.4-39 UaExpert OPC UA ADD_OBJECT_TREE Method call(2)

DEVICE_ID：B-SS のデバイスインスタンス番号です。例では 100 を設定しています。

OFFSET：B-GW 内部用途の値です。1~255 の中から選択します。例では 1 を設定しています。

FOLDER_NAME：B-SS オブジェクトツリーの名称です。例では B_SS_DEVICE_100 としています。

Address Space ウインドウの

Root>Objects>BACnet-Server-Mapping を右クリックして、Rebrowse をクリックします。

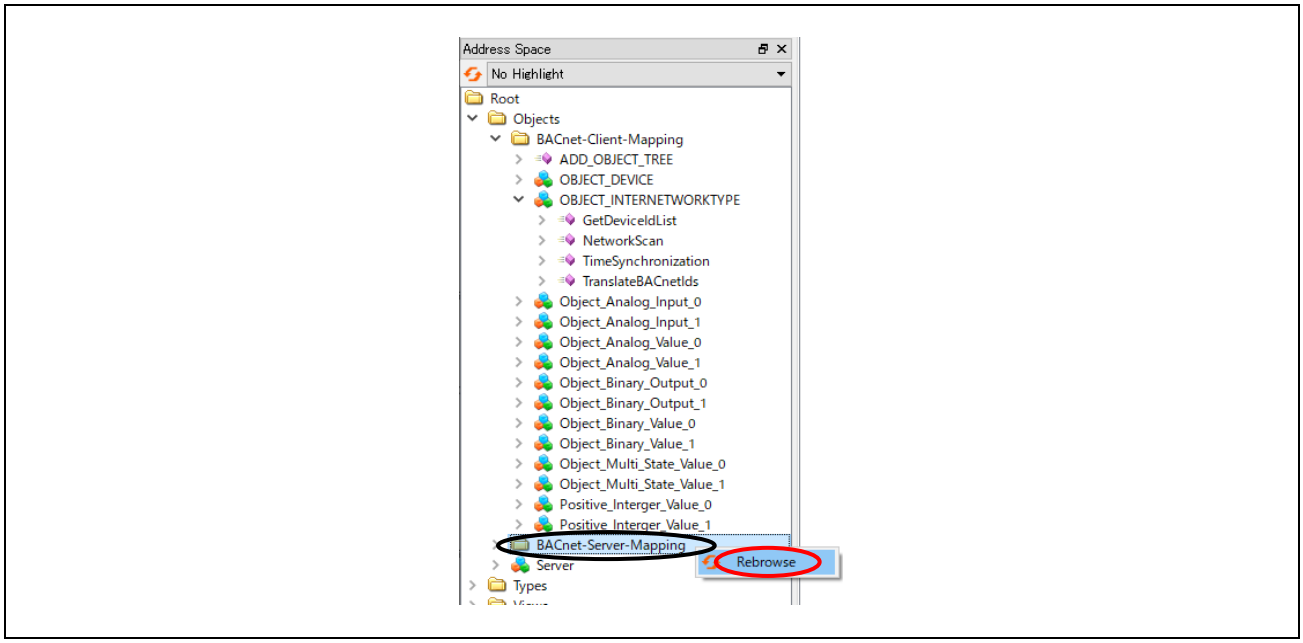


Fig.4-40 UaExpert OPC UA ADD_OBJECT_TREE Method call(3)

Address Space ウインドウの

Root>Objects>BACnet-Server-Mapping を展開すると、追加したオブジェクトツリーが表示されます。

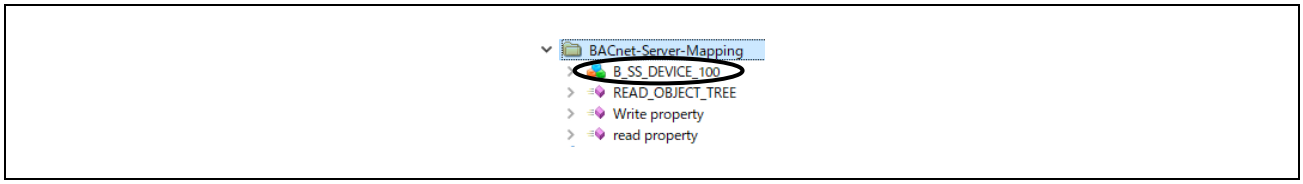


Fig.4-41 UaExpert OPC UA ADD_OBJECT_TREE Method call(4)

追加されたオブジェクトツリーを展開します。

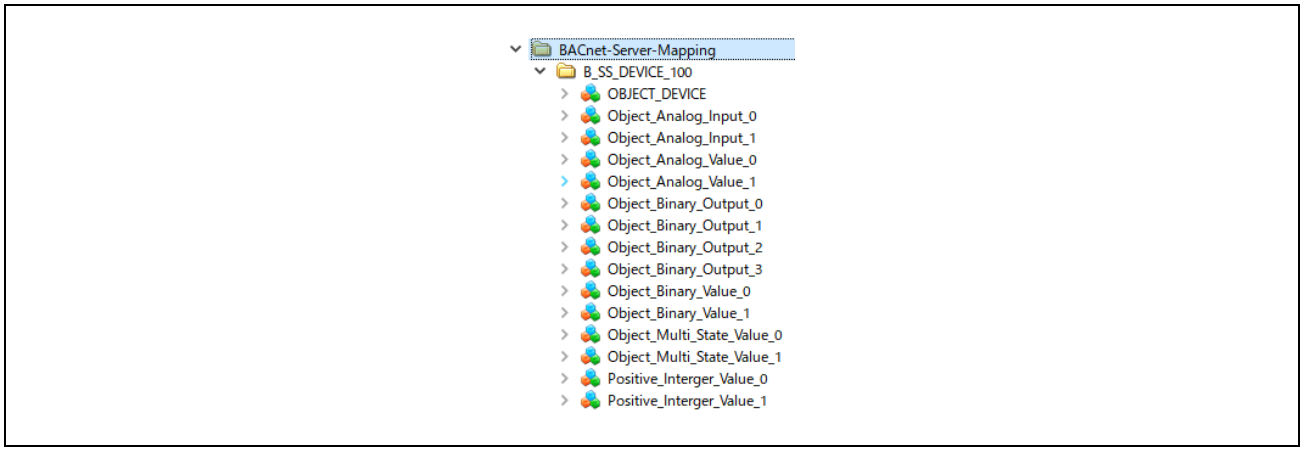


Fig.4-42 UaExpert OPC UA ADD_OBJECT_TREE Method call(5)

オブジェクトツリーの各オブジェクトを展開すると Present_Value ノードが存在しますので、この Present_Value ノードを Data Access View ウィンドウにドラッグアンドドロップします。図では Object_Analog_Input_0 のみ展開して、以降のオブジェクトの展開を省略していますが、すべてのオブジェクトを展開して Present_Value をドラッグアンドドロップします。ただし OBJECT_DEVICE と表記された Device オブジェクトは対象外としています。

Data Access View ウィンドウの Value 欄に各オブジェクトの Present_Value 初期値が表示されます。

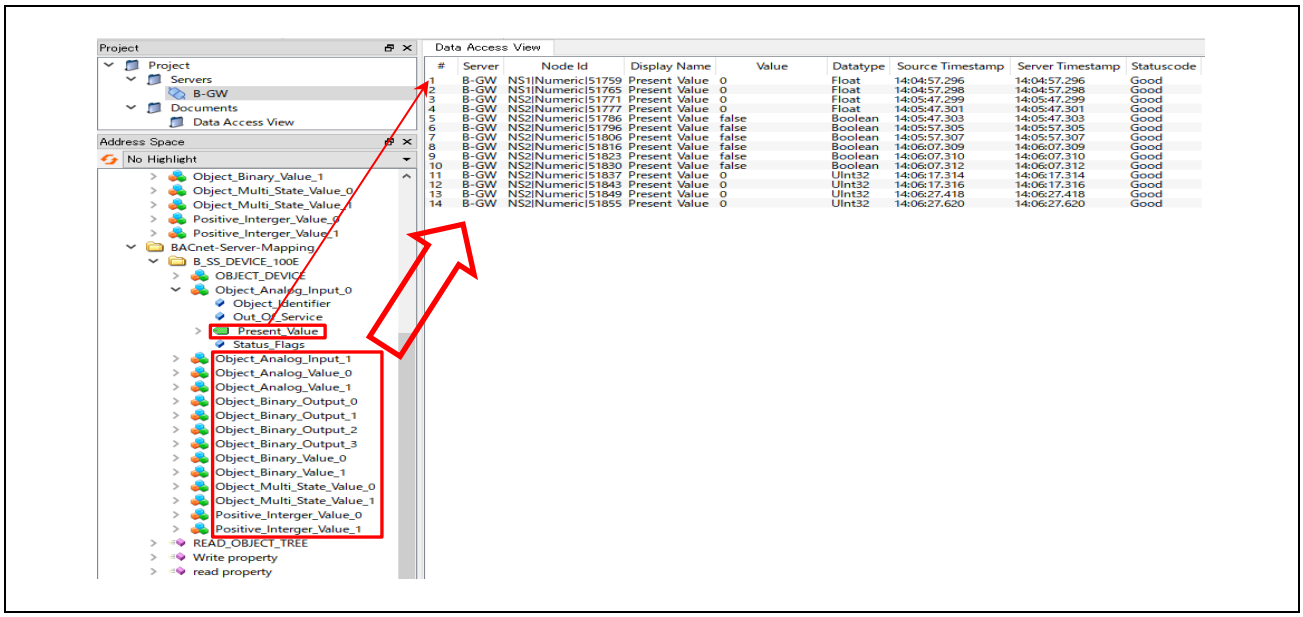


Fig.4-43 UaExpert OPC UA ADD_OBJECT_TREE Method call(6)

READ_OBJECT_TREE メソッドを call します。

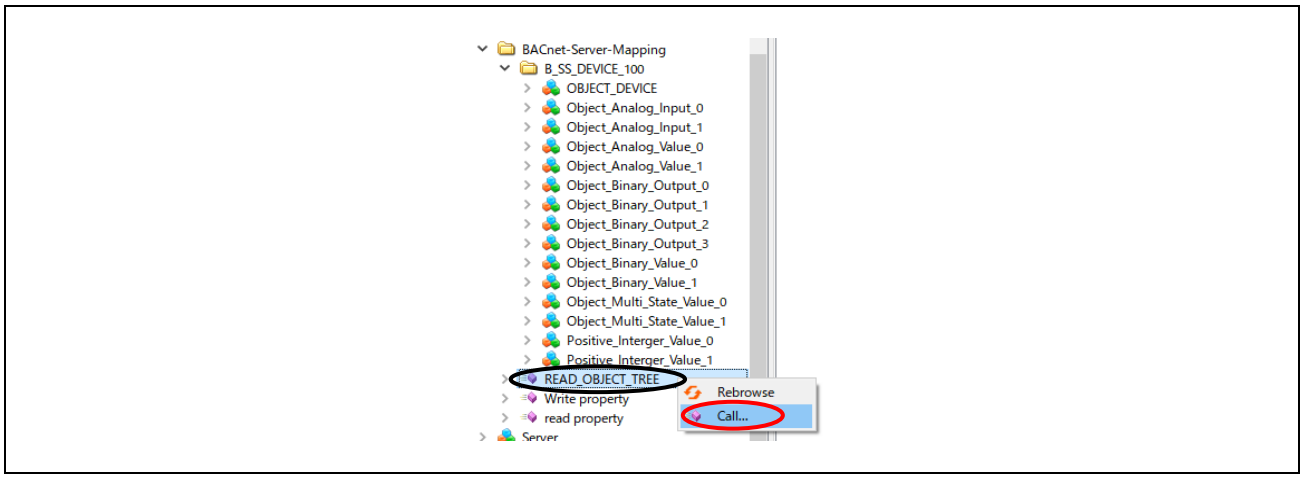


Fig.4-44 UaExpert OPC UA READ_OBJECT_TREE Method call(1)

次のダイアログに 1 を設定して Call をクリックします。この操作により B-GW 内部では B-SS あてに ReadProperty サービス要求を行い B-SS からの応答結果が Value 欄の値に反映されます。このように read_object_tree=1 設定で READ_OBJECT_TREE メソッドを Call する度に値が更新されます。自動的に連続して読み出すことは行いませんので、読み出し前に必ず READ_OBJECT_TREE メソッドを Call してください。

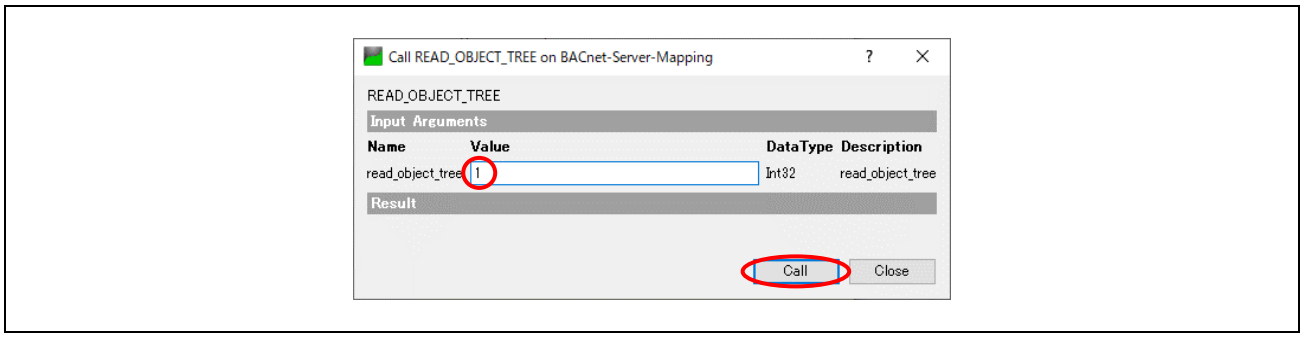


Fig.4-45 UaExpert OPC UA READ_OBJECT_TREE Method call(2)

B-SS オブジェクトの読み出しが正常終了してから、Data Access View ウィンドウの Value 欄を確認します。

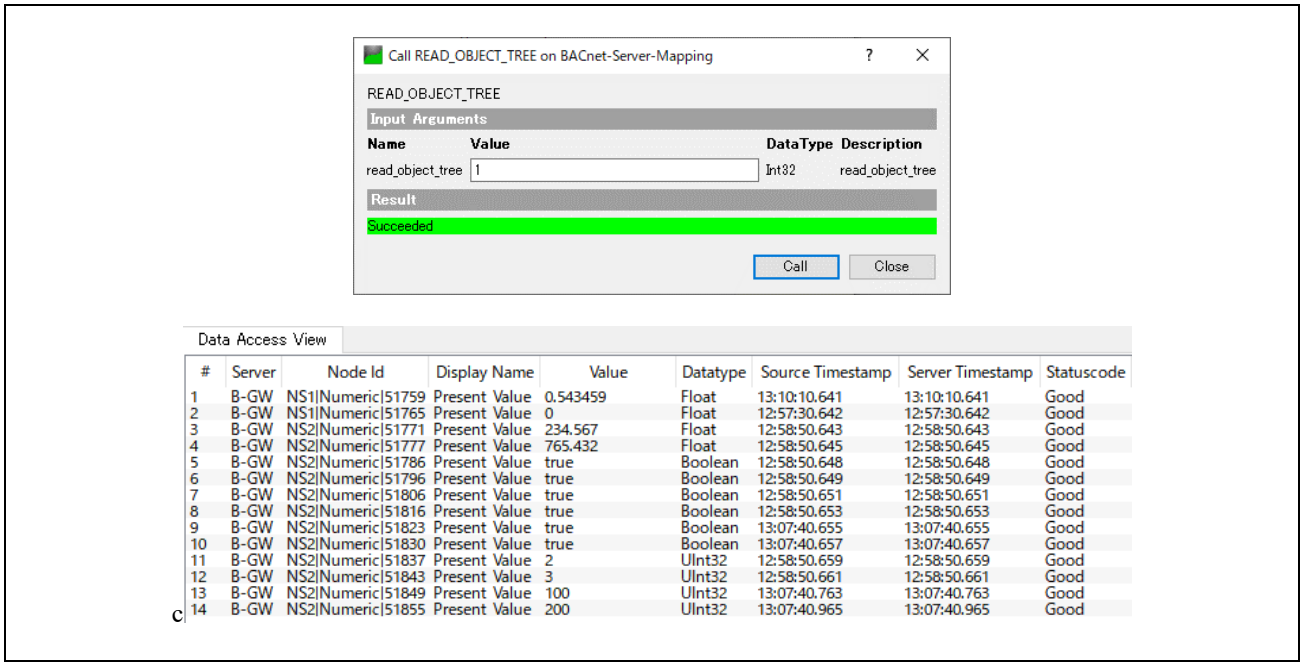


Fig.4-46 UaExpert OPC UA READ_OBJECT_TREE Method call(3)

4.5 B-GW ボード 1 枚での動作確認

B-SS ボードを接続しない場合の OPC UA サーバー動作確認手段を説明します。ビルド手順は 4.3.1(1)章に記載していますので参照ください。

ビルド後は最初に 4.4.1 章の TimeSynchronization メソッドを実行してください。

本来 B-SS から読み出す空気速度センサ入力値を B-GW 内部で疑似的に生成して、B-GW の AnalogInput,0 オブジェクトの PresentValue ノードから読み出すことができます。

次の図のとおり、Address Space ウィンドウの

Root>Objects>BACnet-Client-Mapping>Object_Analog_Input_0>Present_Value ノードを Data Access View ウィンドウにドラッグアンドドロップすると Value 欄の値が変化します。値は空気速度センサ仕様に合わせて 0.0[m/sec]~7.23[m/sec]の範囲で変化を繰り返します。

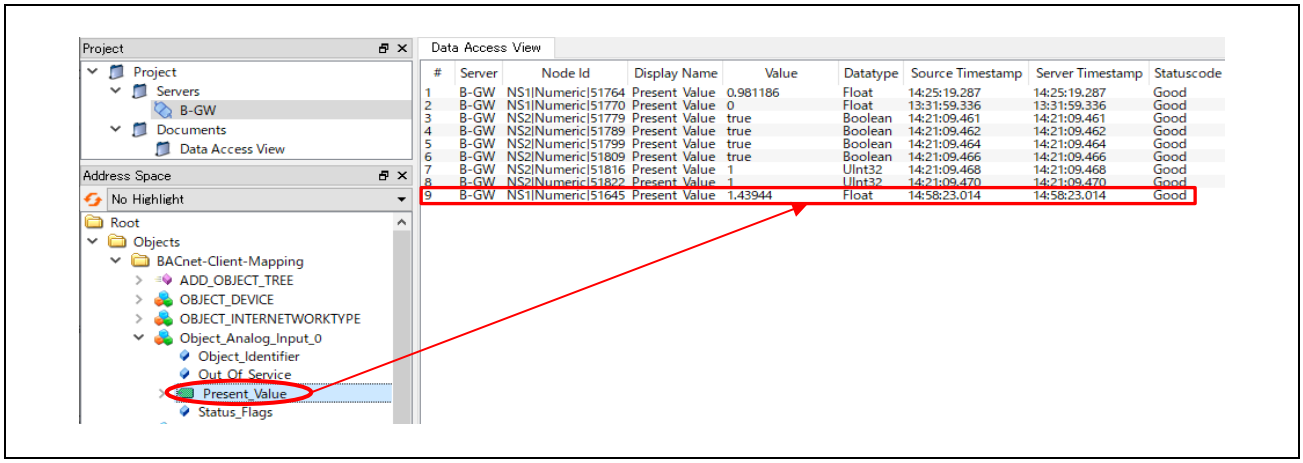


Fig.4-47 B-SS pseudo input value reading

5. Appendix

5.1 open62541 ファイル生成

本サンプルソフトの OPC UA スタックはオープンソースの open62541 を使用しています。Open62541 を freeRTOS + LwIP の環境で実行するため、下記 Link 先では CMake を使って open62541.c と open62541.h を生成する方法が推奨されており、本サンプルソフトでもこの方法を用いています。

Building open62541 — open62541 1.3.0-dirty documentation

本章では、Windows10 環境で open62541 および BACnet 情報モデルを e2studio で実行するためのファイルとして生成する手順について説明します。ここでは、WSL2 が実行可能な Window10 バージョン 1903 以降(OS Build 19044.2965) を使用しています。

5.1.1 Linux 環境構築

CMake を実行するための Linux 環境を構築します。本書では、下記 Link 先ページを参考に WSL2 を使用して Linux (Ubuntu 18.04) をインストールした環境で CMake を実行します。

(参考) 以前のバージョンの WSL の手動インストール手順 | Microsoft Learn

1) PowerShell を [管理者として実行]します。 PowerShell を検索し、右クリックで[管理者として実行]を選択します。
2) 次のコマンドを入力し Linux 用 Windows サブシステムを有効にします。

dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart

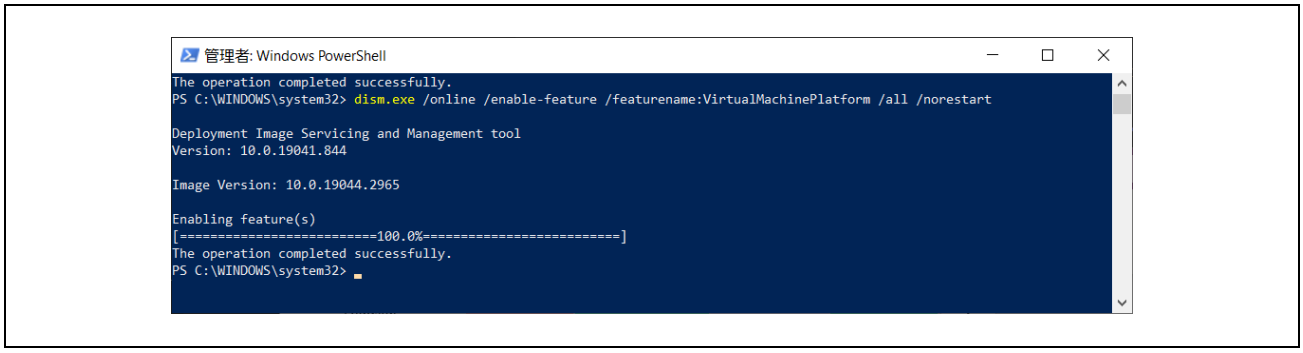


Fig.5-1 Microsoft-Windows-Subsystem-Linux

3) 次のコマンドを入力し仮想マシン プラットフォーム機能を有効にします。

dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

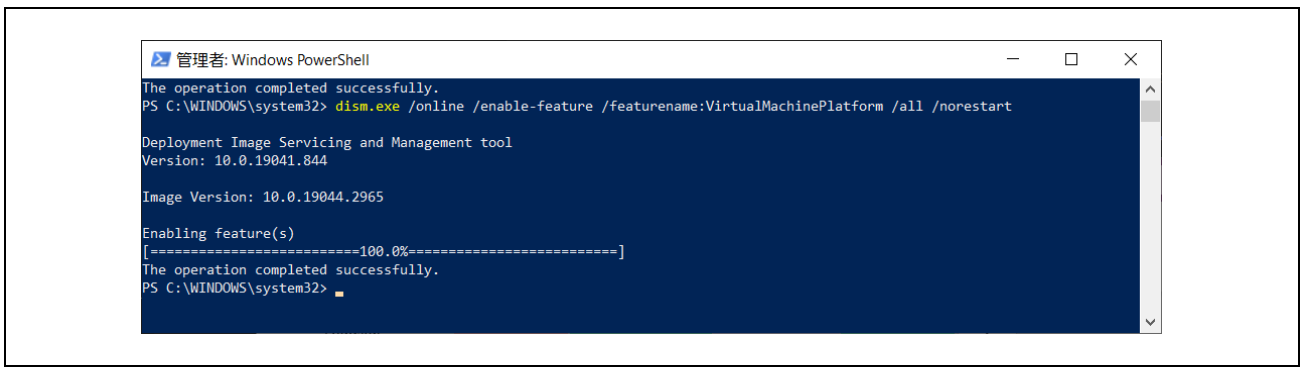


Fig.5-2 VirtualMachinePlatform

4) PC を再起動し、WSL のインストールを完了します。
5) 下記より x64 マシン用 WSL2 Linux カーネル更新プログラム パッケージをダウンロードして実行します。

x64 マシン用 WSL2 Linux カーネル更新プログラム パッケージ

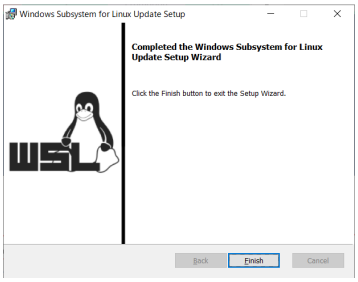


Fig.5-3 wsl_update_x64.msi

6) 下記コマンドを実行し、WSL 2 を既定のバージョンとして設定します。

wsl --set-default-version 2

7) Linux ディストリビューションをダウンロードします。ここでは以下より、Ubuntu 18.04 をダウンロードします。

Ubuntu 18.04

8) ダウンロードしたファイルを含むフォルダに移動し、下記コマンドを実行します。

Add-AppxPackage . \Ubuntu_1804.2019.522.0_x64.appx

9) Ubuntu_1804.2019.522.0_x64.appx をダブルクリックしてインストールします。



Fig.5-4 Ubuntu Install

10) Linux のユーザ名とパスワードを設定します。

(参考) WSL 開発環境を設定する | Microsoft Learn

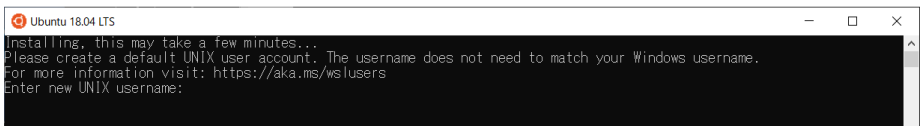


Fig.5-5 UNIX username

5.1.2 CMake インストール

11) 下記 Linux コマンドを実行します。

sudo apt-get update

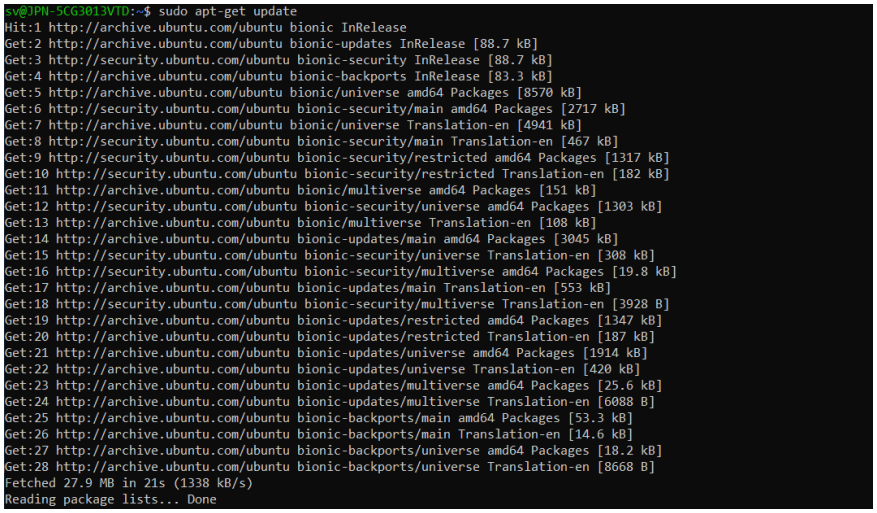


Fig.5-6 apt-get update

12) 下記コマンドを実行します。

sudo apt-get install git build-essential gcc pkg-config cmake python

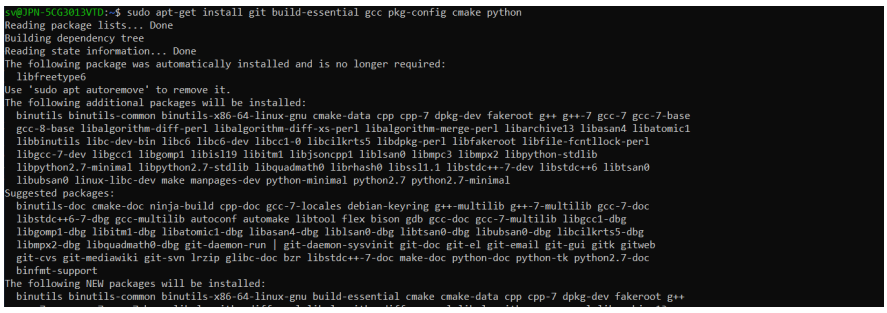


Fig.5-7 install

途中で下記画面がでたら OK を選択します。

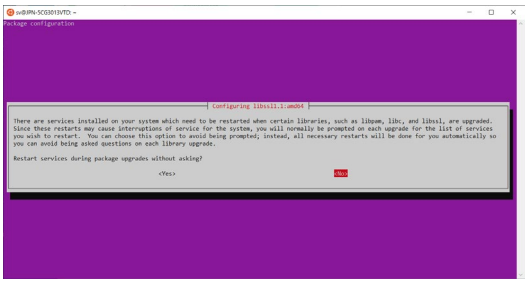


Fig.5-8 restart

13) 下記コマンドをそれぞれ実行します。

```
sudo apt-get install cmake-curses-gui                # Needed for CMAKE GUI
sudo apt-get install libmbedtls-dev                  # For encryption
sudo apt-get install liburcu-dev                     # For multithreading
sudo apt-get install check                           # For unit tests
sudo apt-get install python-sphinx graphviz          # For doc generation
sudo apt-get install python-sphinx-rtd-theme         # For doc's style
```

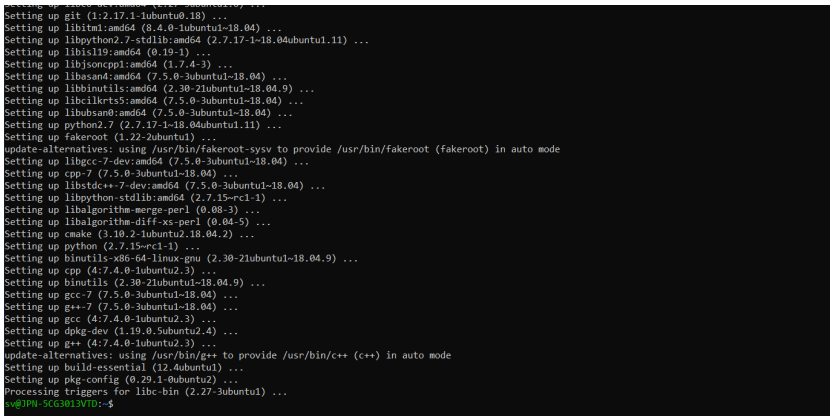


Fig.5-9 install

5.1.3 Open62541 ファイル生成

14) open62541 を任意のフォルダにクローンを生成します。

```
git clone https://github.com/open62541/open62541.git
```

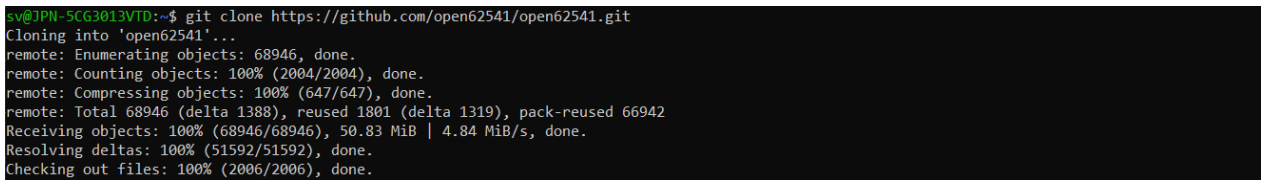


Fig.5-10 git clone

15) /open61541 ディレクトリに移動し、Git submodule で特定のコミットバージョン(ここでは、v1.3.4-564-gb7e5e49f3) を参照します。

```
cd open62541/
git log -1
git checkout b7e5e49f32d00490be74c2eacef892c7fbd0be60
git submodule init
git submodule update
```

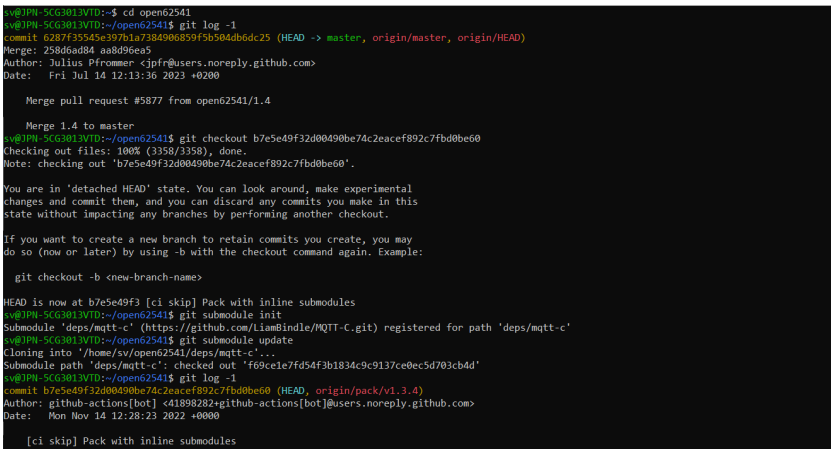


Fig.5-11 git submodule

16) File Explorer から Linux フォルダを開きます。/home/(username)/open62541 ディレクトリに CMakeLists.txt があることを確認します。サンプルソフトに添付の patch_open62541.zip を解凍すると得られる下記 4 つのパッチファイルをここにコピーします。

include-bacnet-xmls.patch

CMakeLists.txt.patch

Opc.Ua.NodeSet2.Reduced.xml.patch

datatypes_dataaccess.txt.patch

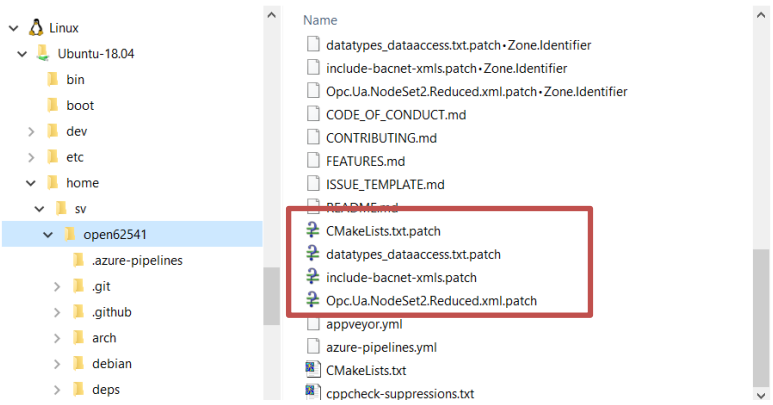


Fig.5-12 Copy patch files

17) /open62541 ディレクトリで下記 Patch コマンドを実行します。

patch -p1 < include-bacnet-xmls.patch

patch -p1 < CMakeLists.txt.patch

patch -p1 < Opc.Ua.NodeSet2.Reduced.xml.patch

patch -p1 < datatypes_dataaccess.txt.patch

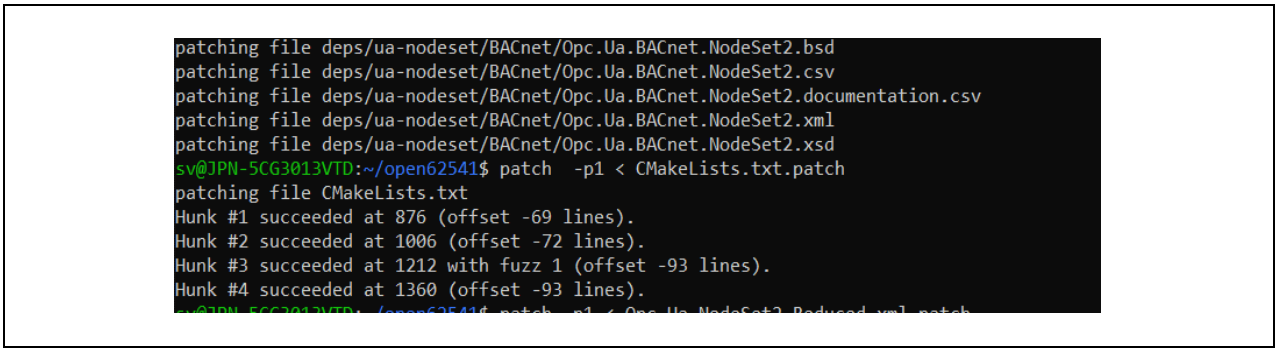


Fig.5-13 patch command

18) cmake プロジェクトの標準的な手順に従いライブラリをコンパイルします。/open62541/build ディレクトリを作成し、cmake .. を実行します。(Failed になる項目もありますが問題ありません)

mkdir build && cd build

cmake ..



Fig.5-14 cmake

19) 下記コマンドを実行し ccmake の設定 Window を起動します。

ccmake ..

20) 下記の通り設定を変更し、[c] to configure を実行後、 [q] to quit without generating で閉じます。

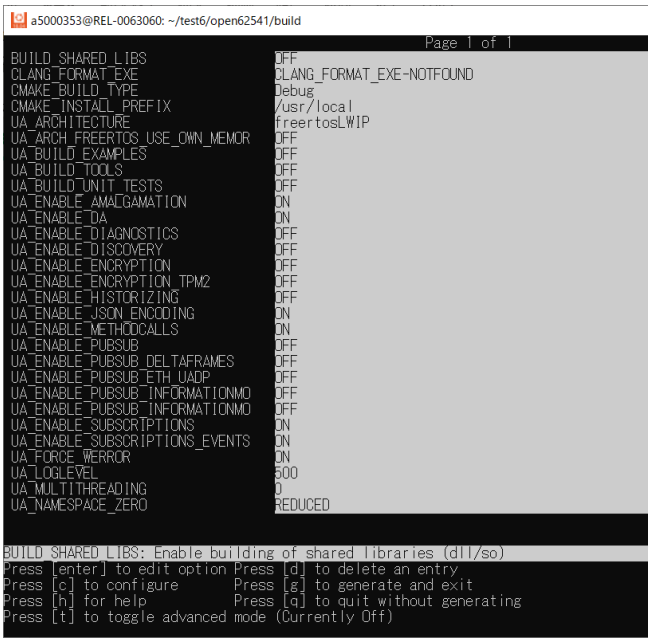


Fig.5-15 ccmake

21) 下記コマンドで make します。make 自体はエラー終了しますが、問題ありません。

make -j

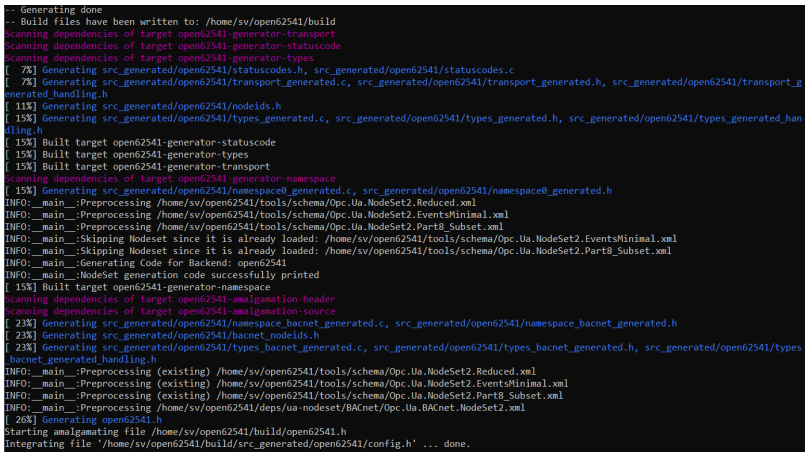


Fig.5-16 make

22) ”make -j” が完了したら、下記コマンドをそれぞれ実行します。

find ./ -type f -exec sed -i -e 's/fields\.unsigned\;/fields\.unsignedValue\;/g' {} \;

find ./ -type f -exec sed -i -e 's/fields\.boolean\;/fields\.booleanValue\;/g' {} \;

find ./ -type f -exec sed -i -e 's/fields\.enum\;/fields\.enumValue\;/g' {} \;

find ./ -type f -exec sed -i -e 's/fields\.enum)/fields\.enumValue)/g' {} \;

find ./ -type f -exec sed -i -e 's/fields\.boolean)/fields\.booleanValue)/g' {} \;

find ./ -type f -exec sed -i -e 's/fields\.unsigned)/fields\.unsignedValue)/g' {} \;

```
find ./ -type f -exec sed -i -e 's/fields\.signed)/fields\.signedValue)/g' {} \;
find ./ -type f -exec sed -i -e 's/enum\;/enumValue\;/g' {} \;
find ./ -type f -exec sed -i -e 's/boolean\;/booleanValue\;/g' {} \;
find ./ -type f -exec sed -i -e 's/unsigned\;/unsignedValue\;/g' {} \;
find ./ -type f -exec sed -i -e 's/signed\;/signedValue\;/g' {} \;
```



Fig.5-17 substitution command

23) *ls* コマンドで /open62541 ディレクトリおよび /src_generated/open62541 ディレクトリ に下記ファイル が生成されていることを確認します。

- *open62541.c*
- *open62541.h*
- *types_bacnet_generated.c*
- *types_bacnet_generated.h*
- *types_bacnet_generated_handling.h*
- *Namespace_bacnet_generated.h*
- *Namespace_bacnet_generated.c*
- *Bacnet_nodeids.h*

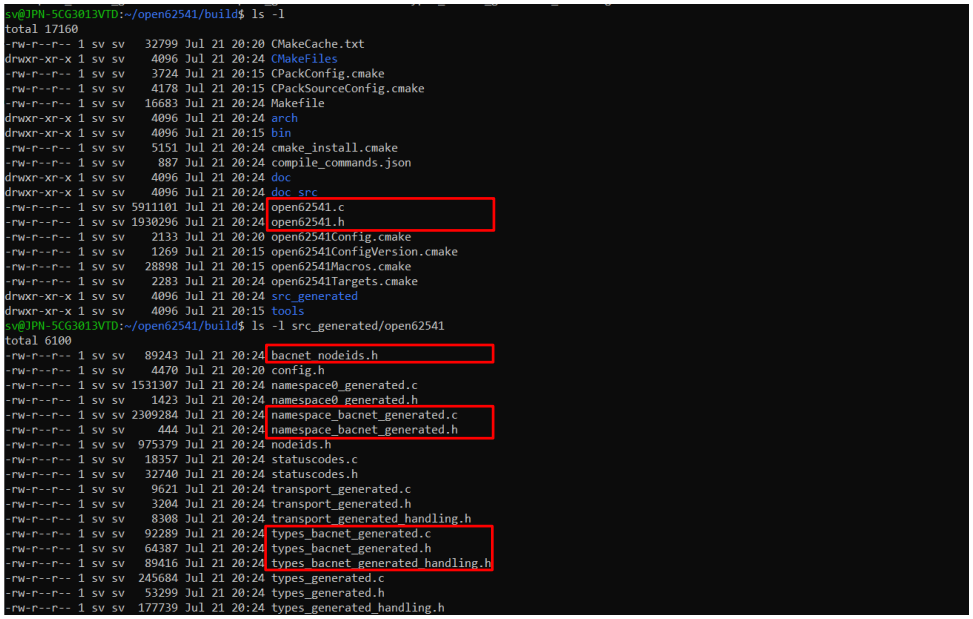


Fig.5-18 Generated Files

24) IDE プロジェクト内にフォルダを作成し、生成されたファイルを図に示すようにプロジェクトにインポートします。ここで OPC_UA_SERVER は e2studio のプロジェクト下に作成されたファイルです。

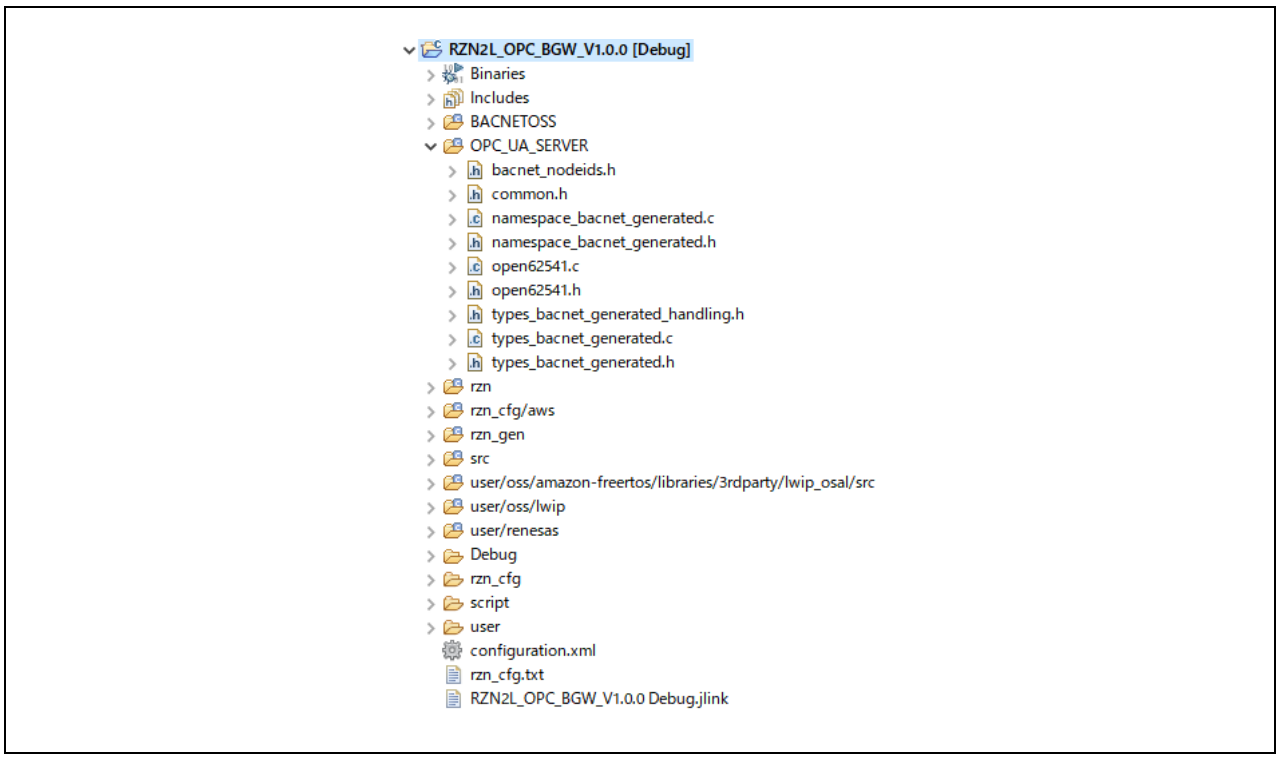


Fig.5-19 Import

5.1.4 生成ファイルの変更点

本手順で生成した open62541 のファイルに対して幾つか変更を行っています。以下に概要を示します。

1. Open62541.c

open62541.c は、namespace の不整合を回避し、BACnet namespace とデフォルトの application namespace をマージするために Fig.5-20 のように変更します。以下のコードをインクルードして、application に BACnet namespace をマージします。

```
for(size_t i = 0; i < UA_TYPES_BACNET_COUNT; ++i) {
if(UA_NodeId_equal(&UA_TYPES_BACNET[i].typeId, typeId))
        return &UA_TYPES_BACNET[i];
  }
```

以下を置換します。

```
value->value.data = booleanValue; to   value->value.data = boolean;
```

application namespace と bacnet namespace の間の namespace 不整合を避けるために、以下のコードをコメントします。

```
//if(n1->namespaceIndex != n2->namespaceIndex)
//    return (n1->namespaceIndex < n2->namespaceIndex) ? UA_ORDER_LESS : UA_ORDER_MORE;
```

nodeset コンパイラーは python スクリプトを使用するため、ノードの作成順序に指定はありません。

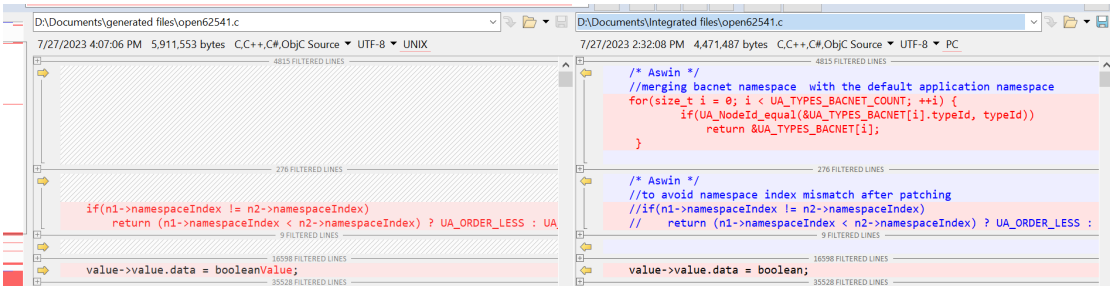


Fig.5-20 difference in open62541.c

2. Open62541.h
変更なし

3. bacnet_nodeids
変更なし

4. types_bacnet_generated_handling.h
5. 変更なし

6. Namespace_bacnet_generated.c
UA_TYPES->UA_TYPES_BACNET

&UA_TYPES->&UA_TYPES_BACNET

78LU->80LU

78LU はオプションノードのマクロ, 80LU は必須ノードのマクロです。

ネットワークスキャン、再初期化、時刻同期方法など、オプションノードを必須ノードに有効化し、オブジェクトフォルダ内のアドレス空間に含めるために、上記の変更が行われました。

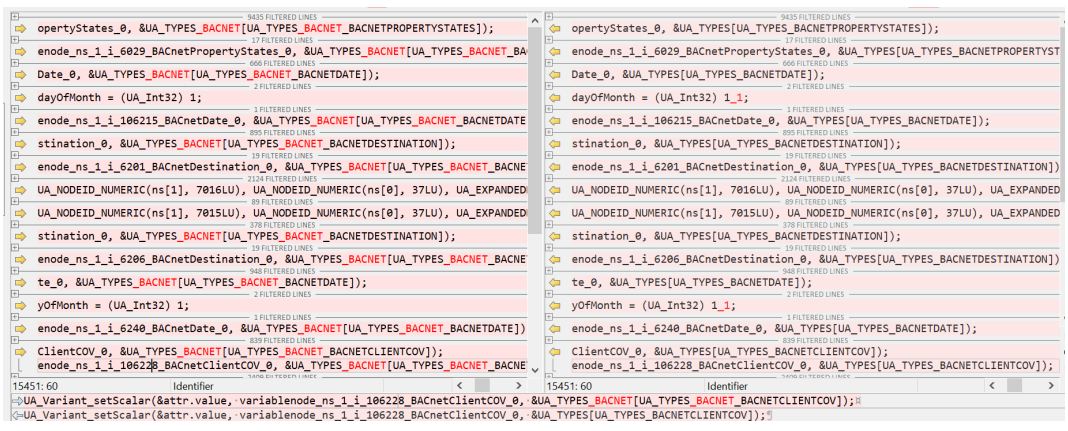


Fig.5-21 difference in namespace_bacnet_generated.c

7. Namespace_bacnet_generated.h

open62541.h がインクルードされているので、types_bacnet_generated.h を削除。



Fig.5-22 difference in namespace_bacnet_generated.h

8. **types_bacnet_generated.h**

#include "types_generated.h" を削除



Fig.5-23 difference in types_bacnet_generated.h

9. types_bacnet_generated.c

インクルードされていた types_generated .h を削除し、open62541.h をインクルード。

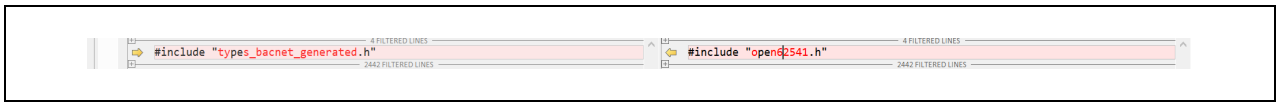


Fig.5-24 difference in types_bacnet_generated.c

5.2 (参考) B-BC Device Profile

本サンプルソフトバージョンで B-BC プロファイルのサポート状況を示します。以降のテーブルに ”Not available” と記載のある箇所は要件を満たしておらず、次版以降の対応を予定しています。

Table 5-1 BACnet Service implementation status required for B-BC

BACnet Service	Initiate[1]	Execute[2]
Who-Is	✓	✓
I-Am	✓	✓
Who-Has	✓	✓
I-Have	✓	✓
ReadProperty	✓	✓
WriteProperty	✓	✓
DeviceCommunicationControl		Not available
ReinitializeDevice		Not available
AtomicReadFile		Not available
AtomicWriteFile		Not available
TimeSynchronization		Not available
UTCTimeSynchronization		
SubscribeCOV		
ConfirmedCOVNotification		
UnconfirmedCOVNotification		
ReadPropertyMultiple	Not available	Not available
ReadPropertyConditional		
ReadRange		Not available
WritePropertyMultiple	Not available	Not available
GetAlarmSummary		
GetEventInformation		Not available
GetEnrollmentSummary		
AcknowledgeAlarm		Not available
ConfirmedEventNotification	Not available	
UnconfirmedEventNotification	Not available	
UnconfirmedTextMessage		
ConfirmedTextMessage		
AddListElement		
RemoveListElement		
CreateObject		
DeleteObject		

BACnet Service	Initiate[1]	Execute[2]
UnconfirmedPrivateTransfer		
ConfirmedPrivateTransfer		
VTOpen		
VTData		
VTClose		

✓は該当、空欄は非該当

1. BACnet サービス要求または通知を送信します。
2. BACnet サービスを実行し、応答(ただし確認あり型サービスを要求された場合)を送信します。

Table 5-2 BACnet Object implementation status required for B-BC

BACnet Object Type	Object ID	Implementation
Accumulator		
Analog Input	Analog Input, 0	✓
	Analog Input, 1	Not available
Analog Output		
Analog Value	Analog Value, 0	Not available
	Analog Value, 1	Not available
Averaging		
Binary Input		
Binary Output	Binary Output, 0	Not available
	Binary Output, 1	Not available
Binary Value	Binary Value, 0	Not available
	Binary Value, 1	Not available
Calendar		
Command		
Device	Device, 12	Not available
Event Enrollment		
File		
Group		
Life Safety Point		
Life Safety Zone		
Loop		
Multi state Input		
Multi state Output		
Multi state Value	Multi state Value, 0	Not available
	Multi state Value, 1	Not available
Notification Class		Not available

BACnet Object Type	Object ID	Implementation
Program		
Pulse Converter		
Schedule		Not available
Trend Log		Not available
Access Door		
Event Log		
Load Control		
Structured View		
Trend Log Multiple		
Access Point		
Access Zone		
Access User		
Access Rights		
Access Credential		
Credential Data Input		
CharacterString Value		
DateTime Value		
Large Analog Value		
BitString Value		
OctetString Value		
Time Value		
Integer Value		
Positive Integer Value	Positive Integer Value, 0	Not available
	Positive Integer Value, 1	Not available
Date Value		
DateTime Pattern Value		
Time Pattern Value		
Date Pattern Value		
Network Security		
Global Group		
Notification Forwarder		
Alert Enrollment		
Channel		
Lighting Output		
Network Port		Not available
Binary Lighting Output		

✓は該当、空欄は非該当

Table 5-3 BIBB implementation status required for B-BC

BIBB Class	BIBB	BACnet Service	Initiate[1]	Execute[2]	B-BC Standardized[3]
DataSharing	DS-RP-A,B	ReadProperty	✓	✓	✓
	DS-WP-A,B	WriteProperty	✓	✓	✓
	DS-RPM-A,B	ReadPropertyMultiple	Not available	Not available	✓
	DS-WPM-A,B	WritePropertyMultiple	Not available	Not available	✓
Alarm & Event Management	AE-N-I-B	ConfirmedEventNotification	Not available		✓
		UnconfirmedEventNotification	Not available		✓
	AE-ACK-B	AcknowledgeAlarm		Not available	✓
	AE-INFO-B	GetEventInformation		Not available	✓
Scheduling	SCHED-E-B	WriteProperty	Not available	Not available	✓
		ReadProperty		Not available	✓
Trending	T-VMT-I-B	ReadRange		Not available	✓
	T-ATR-B	ConfirmedEventNotification	Not available		✓
		UnconfirmedEventNotification	Not available		✓
		ReadRange		Not available	✓
Device & Network Management	DM-DDB-A,B	Who-Is	✓	✓	✓
		I-Am	✓	✓	✓
	DM-DOB-A,B	Who-Has	✓	✓	✓
		I-Have	✓	✓	✓
	DM-DCC-B	DeviceCommunicationControl		Not available	✓
	DM-TS-B	TimeSynchronization		Not available	✓
	DM-RD-B	ReinitializeDevice		Not available	✓
	DM-BR-B	AtomicReadFile		Not available	✓
		AtomicWriteFile		Not available	✓
		ReinitializeDevice		Not available	✓

改定記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2023/07/31	—	新規作成

商標

* Arm および Cortex は、Arm Limited（またはその子会社）の EU またはその他の国における登録商標です。
* Ethernet およびイーサネットは、 富士ゼロックス株式会社の登録商標です。
* その他、本資料中の製品名やサービス名は全てそれぞれの所有者に属する商標または登録商標です

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

© 2023 Renesas Electronics Corporation. All rights reserved.

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1 2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の 商標です。すべての商標および登録商標は、それぞれの所有者に帰属し ます。

© 2023 Renesas Electronics Corporation. All rights reserved.

© 2023 Renesas Electronics Corporation. All rights reserved.