R-IN32M3 Module (RY9012A0)

RL78/G14 サンプルアプリケーション (uGOAL 版)

R30AN0400JJ0104
Rev.1.04
2023.12.15

要旨

本書は、RL78/G14 を R-IN32M3 Module (RY9012A0)のホスト CPU として産業イーサネット通信を行う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

RL78/G14

R-IN32M3 Module (RY9012A0)

目次

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
本ボード	本書で解説するサンプルプログラムのターゲットボードである RL78/G14 Fast Prototype Board と R-IN32M3 Module 搭載アダプタボード(YCONNECT-IT-I-RJ4501)
本サンプルソフト	本書で解説するホストマイコン (RL78/G14) 用サンプルプログラム全体 (サンプルパッケージに同梱)
API	Application Programming Interface
GOAL/uGOAL	Generic Open Abstraction Layer 詳細は、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照

関連文書

資料名	資料番号
R-IN32M3 Module (RY9012A0) データシート	R19DS0109JJ****
R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ハードウェア編	R19UH0122JJ****
R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編	R17US0002JJ****
R-IN32M3 Module (RY9012A0) Management Tool 操作ガイド	R30AN0390JJ****
R-IN32M3 Module (RY9012A0) Modbus TCPスタートアップマニュアル	R30AN0406JJ****
R-IN32M3 Module 搭載アダプタボード YCONNECT-IT-I-RJ4501 ユーザーズマニュアル	R12UZ0094JJ****
R-IN32M3 Module (RY9012A0) ユーザ実装ガイド (uGOAL版)	R30AN0402JJ****
RL78/G14 Fast Prototyping Board Quick Start Guide	R20UT4571EJ****
RL78/G14 Fast Prototyping Board ユーザーズマニュアル	R20UT4573JJ****
R-IN32M3 Module (RY9012A0)ソフトウェア PLC接続ガイド: TwinCAT	R30AN0380JJ****

1. 概要

1.1 概要

本書は、RL78/G14 Fast Prototyping Board 用の R-IN32M3 モジュールサンプルソフトウェアについて説明します。

本サンプルソフトは RL78/G14 評価ボードである RL78/G14 Fast Prototyping Board に R-IN32M3 Module 搭載アダプタボード(YCONNECT-IT-I-RJ4501) を Arduino™ コネクタで接続した評価環境で実行することで、PROFINET, EtherNet/IP, EtherCAT といった主要な産業イーサネットプロトコルで通信を行うことができます。

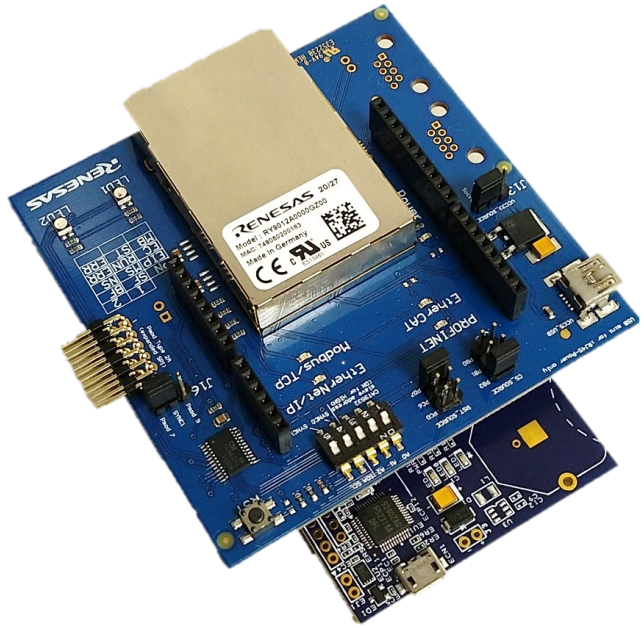

図 1.1 R-IN32M3 モジュール + RL78/G14 Fast Prototyping Board

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表 1-1 に示します。

表 1-1 動作環境

Category	Name	Version	Link	備考
R-IN32M3 モジュール サンプルパッケージ	サンプル パッケージ	Rev.1.04	Renesas R-IN32M3 Module Sample Package	https://www.renesas.com/
統合開発環境	e2studio	2023-04	e² studio 2023-04 Windows| Renesas	
RL ファミリ用 GNU Toolchain	GCC for Renesas RL78	V4.9.2.202201	-	e2studio インストーラーに同梱
Management Tool・簡易 PLC	ICE	V1.5.1	-	port industrial automation GmbH 社製 サンプルパッケージに同梱
ソフトウェア PLC	TwinCAT	V3.1	https://www.beckhoff.com/	Beckhoff Automation 社製

1.2.2　ハードウェア環境

本サンプルソフトは、RL78/G14 MCU グループ評価キット(RL78/G14 Fast Prototyping Board)に R-IN32M3 モジュール搭載アダプタボードを接続したハードウェア環境で動作確認を行っております。

RL78/G14 Fast Prototyping Board をお使いの場合には、E2 エミュレータ Lite 相当のエミュレータサーキットがボードに内蔵されているために、本サンプルソフトの実行のために個別にエミュレータを準備する必要はありません。

また、本サンプルソフトには複数のアプリケーションが含まれておりますが、マルチプロトコルアプリケーション、Remote I/O アプリケーションは、表 1-2 の Digilent 製 Pmod ボードを接続することで実行することが可能です。詳細は、3.3.2 章をご参照下さい。

表 1-2 ハードウェア環境

Name	Type Name	Maker	Link	Note
RL78/G14 Fast Prototyping Board	RTK5RLG140C00000BJ	Renesas Electronics Corporation	RL78/G14 Fast Prototyping Board	
Adapter Board with R-IN32M3 Module	YCONNECT-IT-I-RJ4501	Renesas Electronics Corporation	R-IN32M3-Module-Solution-Kit	
6-pin Pmod with 4-ch Switch	Pmod SWT (410-083)	Digilent, Inc.	https://reference.digilentinc.com/reference/pmod/pmodswt/start	Multi-protocol application, EtherCAT ID, remote I/O application
6-pin Pmod with 4ch LED	Pmod LED (410-076)	Digilent, Inc.	https://reference.digilentinc.com/reference/pmod/pmodled/start	remote I/O application

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 アダプタボード設定

本サンプルソフトウェアご使用時は、R-IN32M3 モジュール搭載アダプタボード(YCONNECT-IT-I-RJ4501)の J13、J8、および J7 ジャンパーブロックを以下のように設定してください。

- J13：Socket 端子と iRJ45 端子を接続
- J8　：CS 信号は、PB2 を選択
- J7　：RST 信号は、PD7 を選択

また、EtherCAT の DC モードを使用する際は、J10 の 3pin と 6pin 及び 4pin と 7pin をショートして下さい。

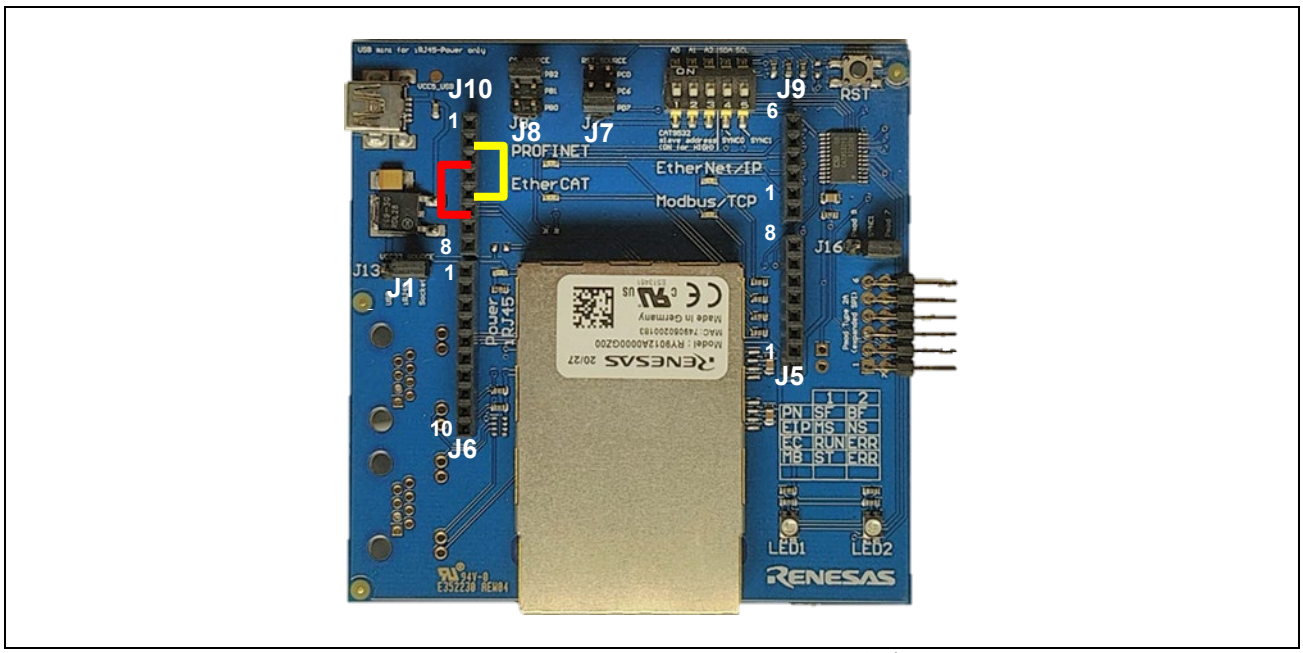

図 2.1　　R-IN32M3 モジュール搭載アダプタボード

R-IN32M3 モジュール搭載アダプタボード裏面にあるオスの Arduino コネクタを、RL78/G14 Fast Prototyping Board のソケットに差し込みます。

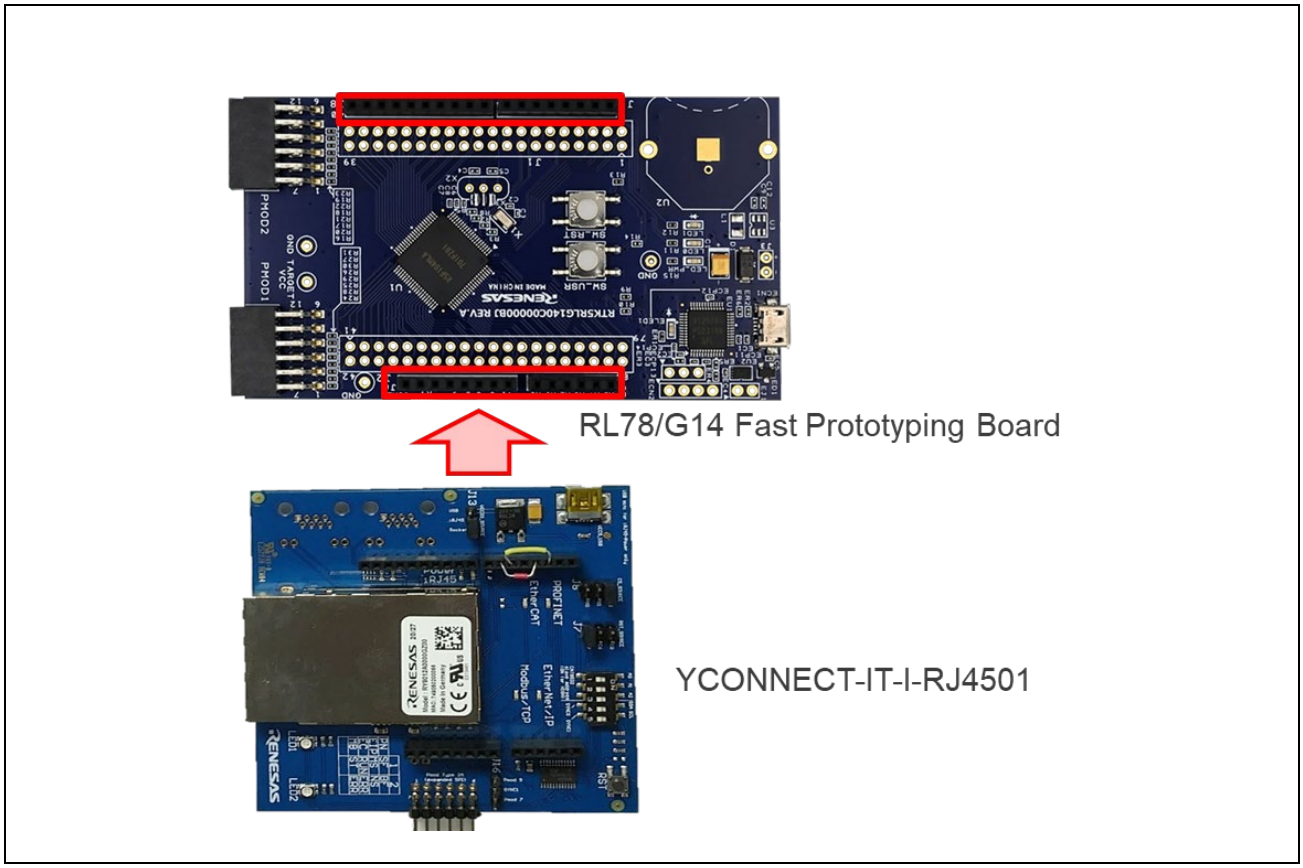

図 2.2 Arduino 接続

2.2　Multi-protocol 対応

本サンプルソフトの multi-protocol サンプルアプリケーションでは、multi-protocol(PROFINET, EtherNet/IP, EtherCAT, Modbus TCP)のセレクタ入力は、RL78/G14 Fast Prototyping Board の Pmod2 コネクタ上段(1-6pin)に Pmod SWT を接続して設定されることを想定しています。

Remote I/O 用サンプルアプリケーションのスイッチ入力、及び、EtherCAT Explicit Device ID と兼用で使用します。

表 2-1 Pmod SWT (multi-protocol セレクタ)接続

PMOD2 Upper	RL78/G14 Fast Prototyping Board	Pmod SWT
1	P16	Selector-ID1
2	P13	Selector-ID2
3	P14	Selector-ID3
4	P15	Selector-ID4
5	GND	GND
6	+3.3V	VCC

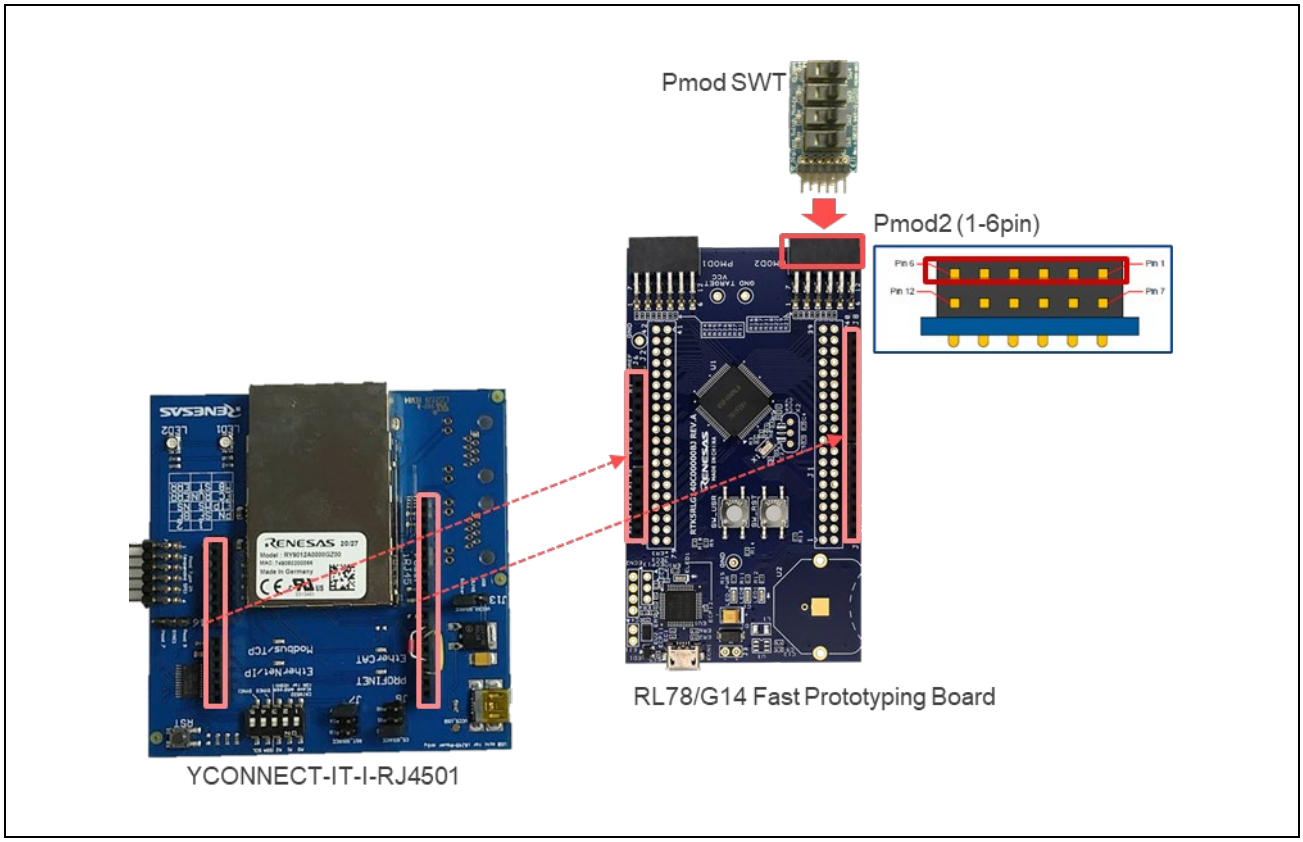

図 2.3　multi-protocol セレクタ接続

2.3　Remote I/O 対応

Remote I/O 用サンプルアプリケーションは、Pmod2 コネクタの上段(1-6pin)にスイッチ入力(Pmod SWT)、Pmod1 コネクタの下段(7-12pin) に LED 出力(Pmod LED) を接続することを想定しています。(図 2.4)

multi-protocol サンプルアプリケーションのセレクタ入力、及び、EtherCAT Explicit Device ID と兼用で使用します。

表 2-2 Pmod SWT 接続

PMOD2 Upper	RL78/G14 Fast Prototyping Board	Pmod SWT
1	P16	SW1
2	P13	SW2
3	P14	SW3
4	P15	SW4
5	GND	GND
6	+3.3V	VCC

表 2-3 Pmod LED 接続

PMOD1 Lower	RL78/G14 Fast Prototyping Board	Pmod LED
7	P140	LED1
8	P130	LED2
9	P147	LED3
10	P146	LED4
11	GND	GND
12	+3.3V	VCC

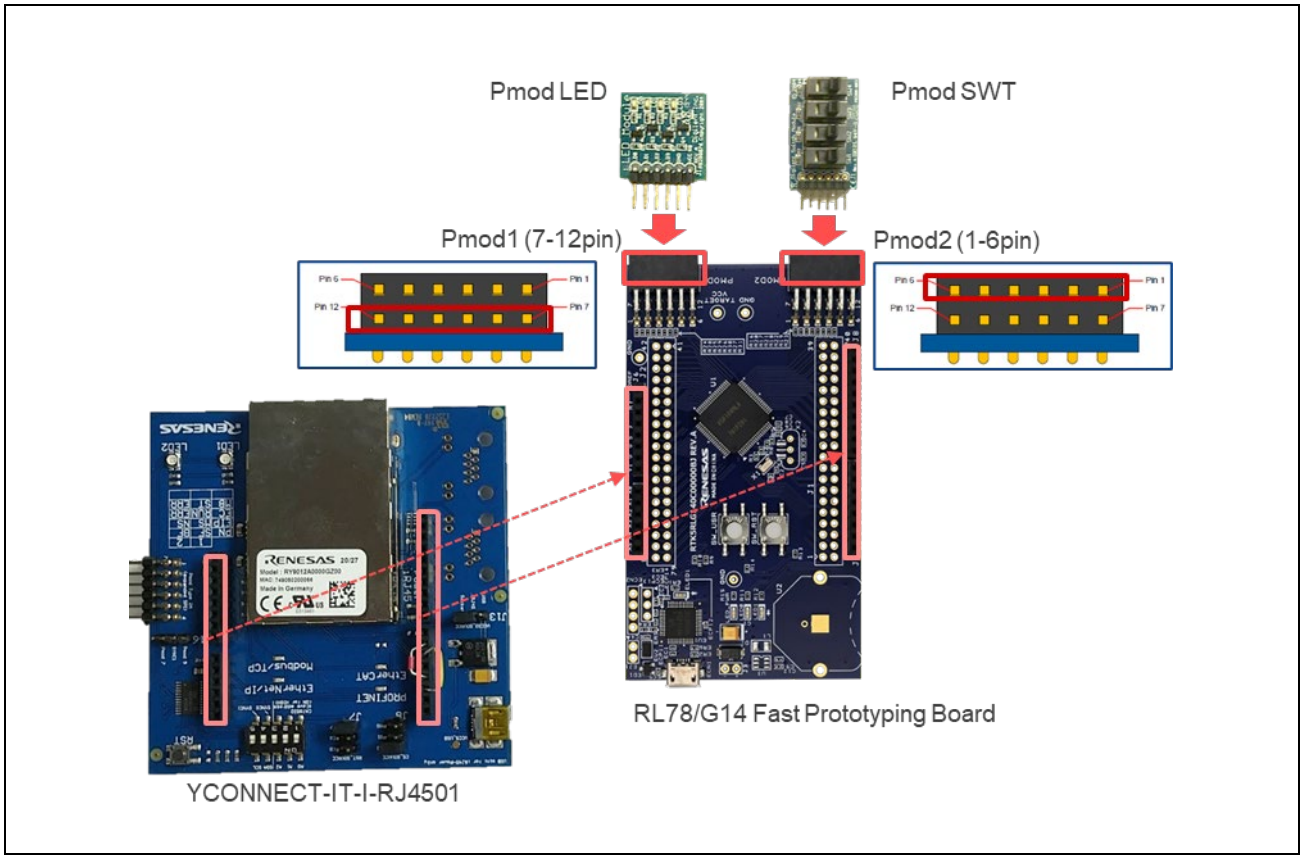

図 2.4 Remote I/O 接続

2.4　EtherCAT Explicit Device ID 入力対応

本サンプルソフトの EtherCAT では、Explicit Device ID のセレクタ入力を RL78/G14 Fast Prototyping Board の Pmod2 コネクタ上段(1-6pin)に Pmod SWT を接続して設定することを想定しています。

multi-protocol サンプルアプリケーションのセレクタ入力、及び、Remote I/O 用サンプルアプリケーションのスイッチ入力と兼用で使用します。

表 2-4 Pmod SWT (EtherCAT ID)接続

PMOD2 Upper	RL78/G14 Fast Prototyping Board	Pmod SWT
1	P16	ECAT-ID1
2	P13	ECAT-ID2
3	P14	ECAT-ID3
4	P15	ECAT-ID4
5	GND	GND
6	+3.3V	VCC

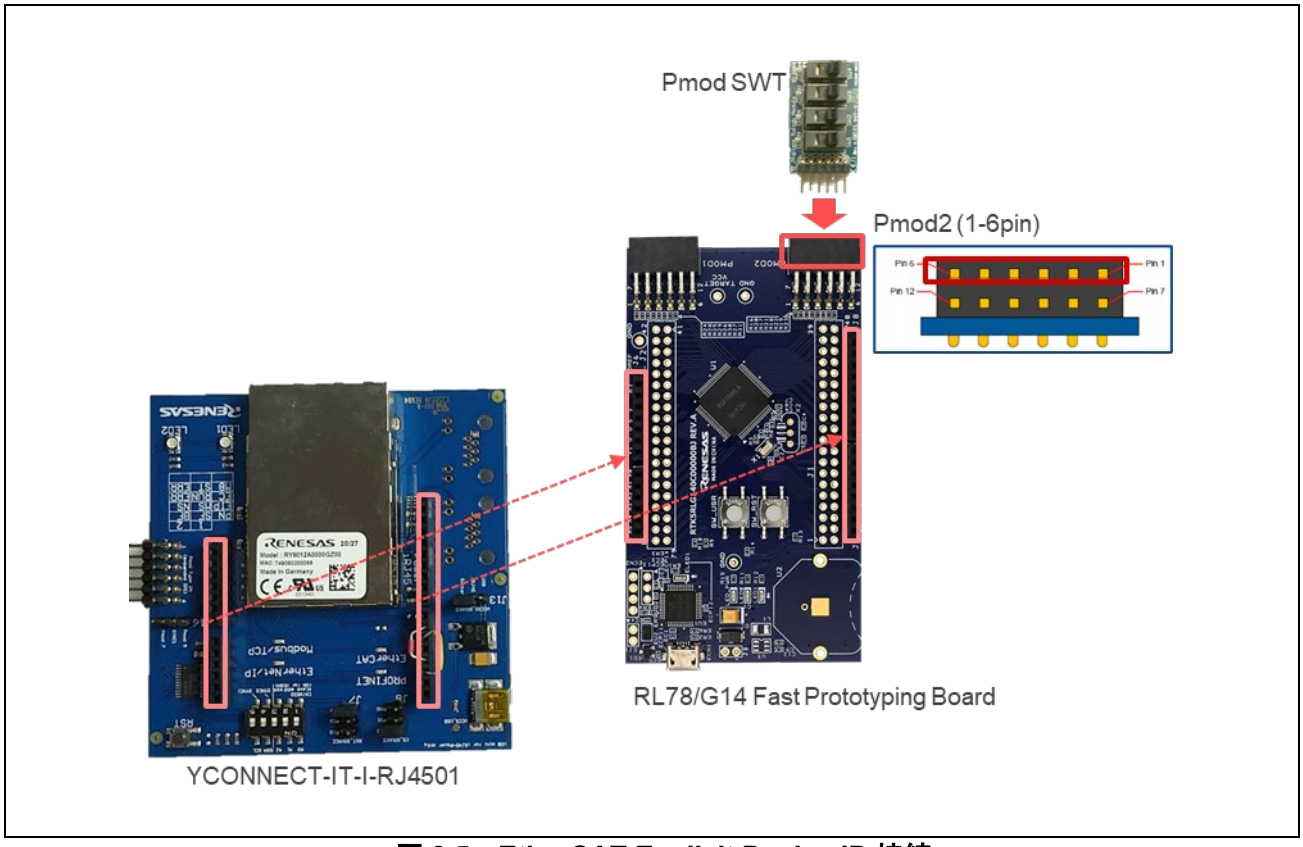

図 2.5　EtherCAT Explicit Device ID 接続

なお、本環境で EtherCAT Conformance Test Tool を実行する場合は Pmod SWT を [5] に設定してください。テストケース TF-1201 ESM - Explicit Device Identification では Explicit Device ID [5] を期待値としています。

3. サンプルソフト構成

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。

```
RL78_uCCM_V***
 ├──appl                    ユーザアプリケーション
 │    ├──01_pnio            PROFINET サンプルアプリケーション
 │    ├──02_eip             EtherNet/IP サンプルアプリケーション
 │    ├──03_ecat            EtherCAT サンプルアプリケーション
 │    ├──04_pnio_largesize  PROFINET Large Size サンプルアプリケーション
 │    ├──05_eip_largesize   EtherNet/IP Large Size サンプルアプリケーション
 │    ├──06_ecat_largesize  EtherCAT Large Size サンプルアプリケーション
 │    ├──07_modbus_tcp_slave Modbus TCP サンプルアプリケーション
 │    └──10_multi_protocol  Multi [01_pnio, 02_eip, 03_ecat, 07_modbus] サンプルアプリケーション
 │
 ├──plat                    デバイス依存コンポーネント(OS 依存部、ボード仕様、ドライバ群)
 ├──projects                ユーザアプリケーションと対になるプロジェクトファイル一式
 │
 ├──ugoal                   uGOAL(Generic Open Abstraction Layer *)のメイン部
 ├──rpc                     NW プロトコルや MCTC を含む RPC(Remote Procedure Call)に関連した機能部
 ├──sapi                    Simple API
 └──ext                     外部ソフトウェアコンポーネント
```

* uGOAL の詳細については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照ください。

3.2　サンプルソフト概要

本サンプルソフトに搭載されているプロトコル (PROFINET、EtherNet/IP、EtherCAT) は、以下の機能をサポートします。

表 3-1 Protocol and feature

プロトコル	機能
PROFINET	・Conformance：CC-B（RT） ・Netload： I Min Interval：1ms ・I&M：1-4
EtherNet/IP	・DLR：Support
EtherCAT	・DC：Support ・Mailbox：CoE / FoE / EoE ・Profile：MDP

サンプルソフトでは、アプリケーション例として 2 種類のデータ送受信アプリケーションを実装しています。

- Remote-IO (LED/Switch)：評価ボードの LED 点灯制御および Switch 状態を送受信
- Mirror：マスタから受信したデータをミラーバック送信

プロジェクト名	プロトコル	参照先
01_pnio	PROFINET	3.4.1 PROFINET
02_eip	EtherNet/IP	3.4.2 EtherNet/IP
03_ecat	EtherCAT	3.4.3 EtherCAT
04_pnio_largesize	PROFINET	3.4.1 PROFINET
05_eip_largesize	EtherNet/IP	3.4.2 EtherNet/IP
06_ecat_largesize	EtherCAT	3.4.3 EtherCAT
07_mbus_tcp_sever	ModbusTCP	3.4.4 Modbus TCP
10_multi_protocol	PROFINET / EtherNet/IP / EtherCAT / Modbus TCP	3.4.5 multi-protocol

04_pnio_largesize, 05_eip_largesize, 06_ecat_largesize プロジェクトでは Cyclic 通信に加え RPC 通信を使った 大容量データ通信が可能です。RPC 通信に関する詳細は『R-IN32M3 Module (RY9012A0) ユーザ実装ガイド (uGOAL 編) (R30AN0402JJ****)』を参照ください。

3.3　開発環境構築

本サンプルソフトの動作環境については、1.2 章をご参照ください。

3.3.1　インストール

(1)　統合開発環境 e2studio, GCC for Renesas RL78

以下の 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。e2studio と合わせて RL ファミリ用 GNU Toolchain である GCC for Renesas RL78 がインストールされます。

インストールに関して以下 4 点の注意事項があります。

①インストールの途中で[デバイス・ファミリー]を選択する画面になりましたら、”RL78”へのチェックを忘れずに行ってください (他と合わせて複数選択可)。

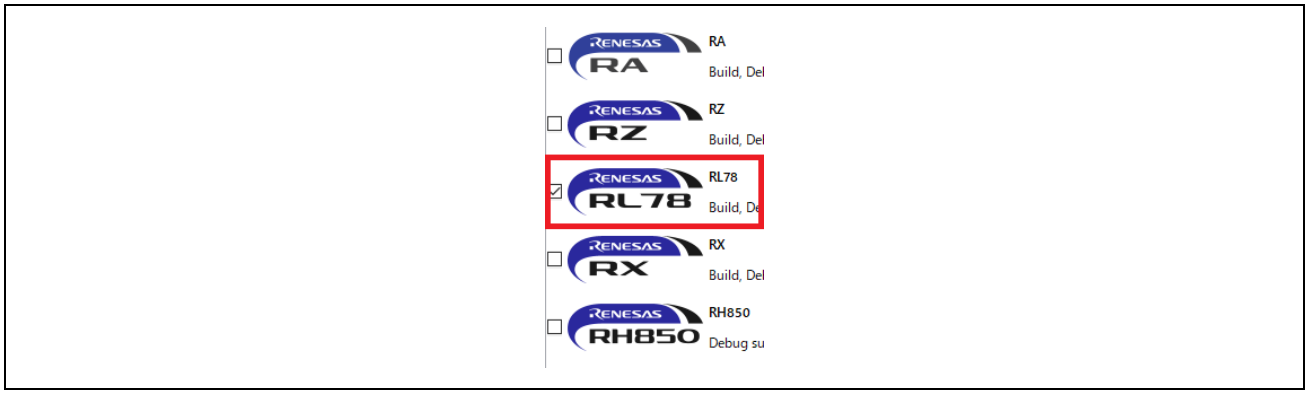

図 3-1 デバイス・ファミリー選択

②インストールの途中で[追加ソフトウェア]を選択する画面になりましたら、”GCC Toolchains & Utilities”タブを選択します。”GCC for Renesas RL78 4.9.2.202201”を有効にして GCC をインストールしてください。

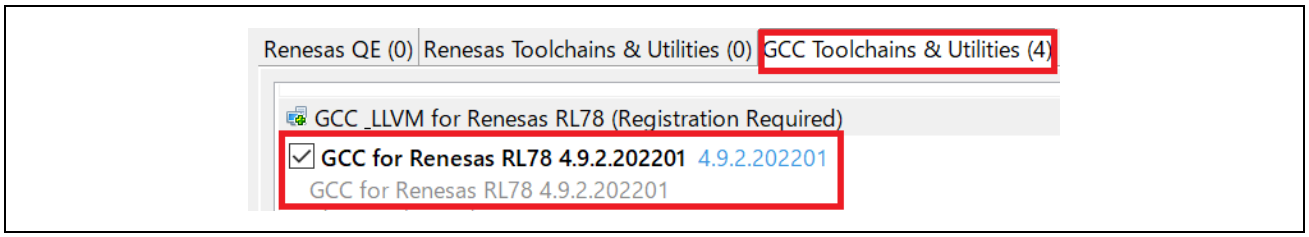

図 3-2 追加ソフトウェア選択

③GCC コンパイラのインストールの途中で、CyberTHOR Studios Limited 社のユーザ登録が求められる場合があります。

アカウントを所持していない場合は、”Register Now”から登録してください。もしくは、Open Source Tools for Renesas (llvm-gcc-renesas.com) からも登録可能です。

図 3-3 GCC コンパイラのインストール(アカウント未所持)

登録後もしくはアカウントを所持している場合は、registered use にチェックして、 [Next >] を押します。その後、登録した e-mail, および Authentication Code を入力して GCC のインストールを進めてください。

④[Change PATH environment variable automatically]へのチェックが付いている事を確認しインストールを進めてください。

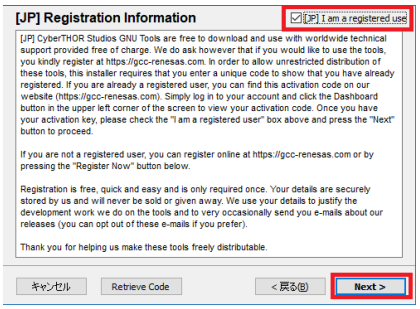

図 3-4 GCC コンパイラのインストール(アカウント所持)

3.3.2　開発環境接続

(1)　P-mod による追加接続なしで評価する場合

RL78/G14 Fast Prototyping Board に R-IN32M3 Module 搭載アダプタボードを接続した後(2.1 参照)、お使いの PC を以下のように接続します。ボードに USB micro B ケーブルを接続する事によって、ボードに電源が供給されます。

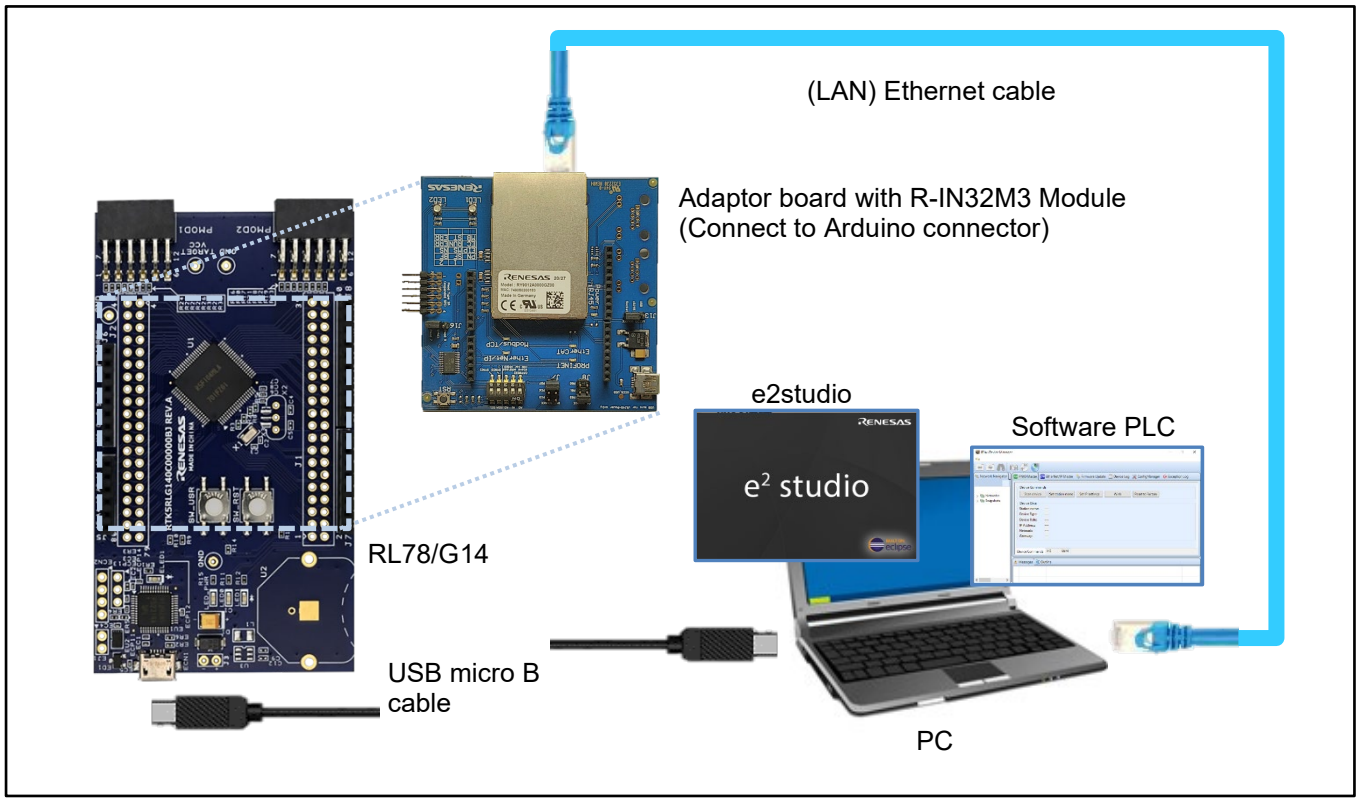

図 3-5　　接続構成

(2)　Pmod SWT, Pmod LED を接続する場合

RL78/G14 Fast Prototyping Board 上にある Pmod2 端子の上段(1-6pin)に Pmod SWT(表 1-2 参照)、Pmod1 端子の下段(7-12pin)に Pmod LED(表 1-2 参照)をそれぞれ接続します(2.3 参照)。ボードに USB micro B ケーブルを接続する事によって、ボードに電源が供給されます。

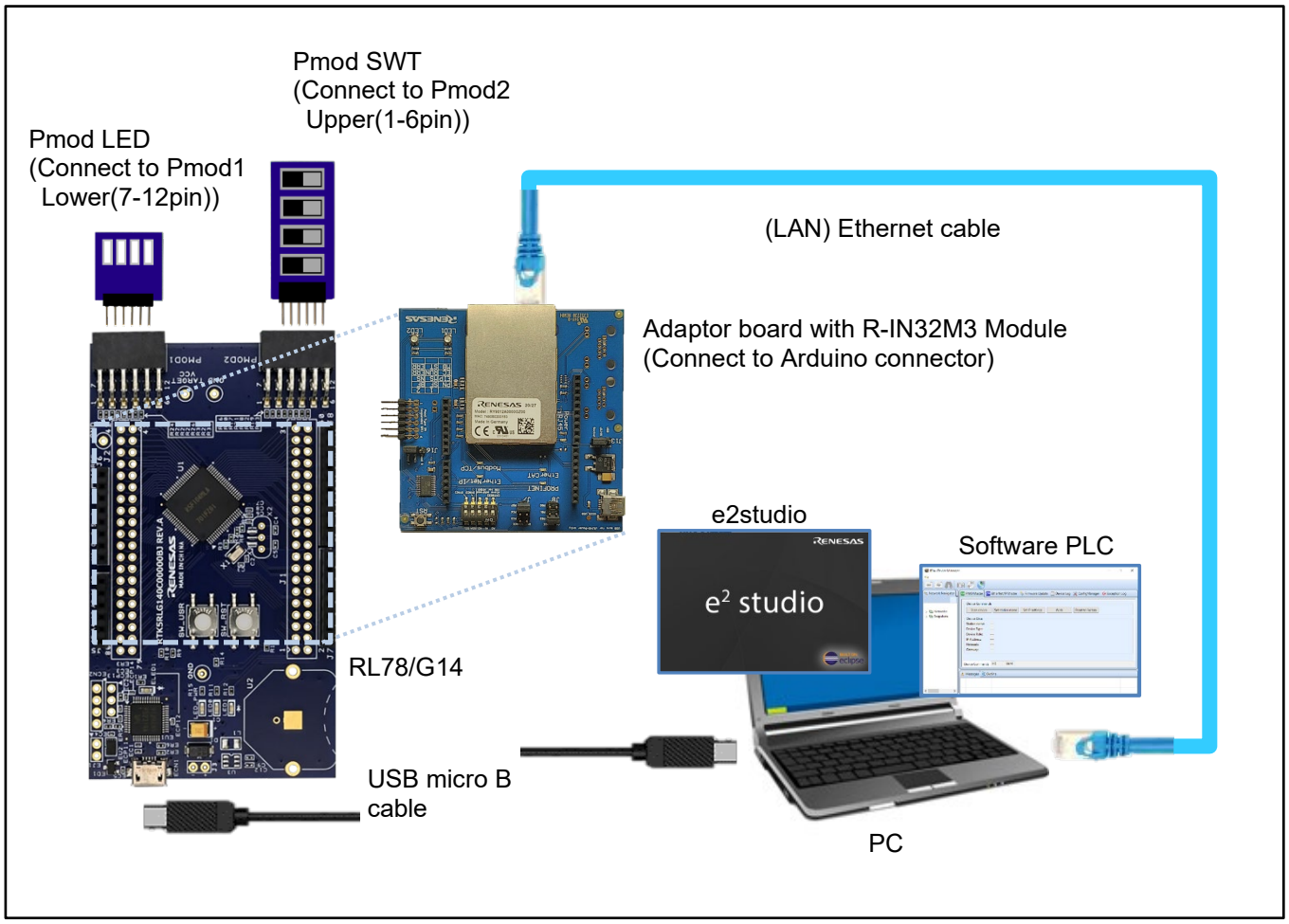

図 3-6　接続構成(Pmod SWT と Pmod LED を接続)

3.3.3　プロジェクト立上げ

(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 (RL78_uCCM_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。

なお、上記でインストールされたコンパイラを確認する場合は、[ウィンドウ]→[設定]を選択し、[設定]ダイアログで[Renesas]→[Renesas ツールチェーン管理]を選択します。[Renesas ツールチェーン管理]ダイアログで、”GCC for Renesas RL78”に該当コンパイラが追加されているかを確認できます。

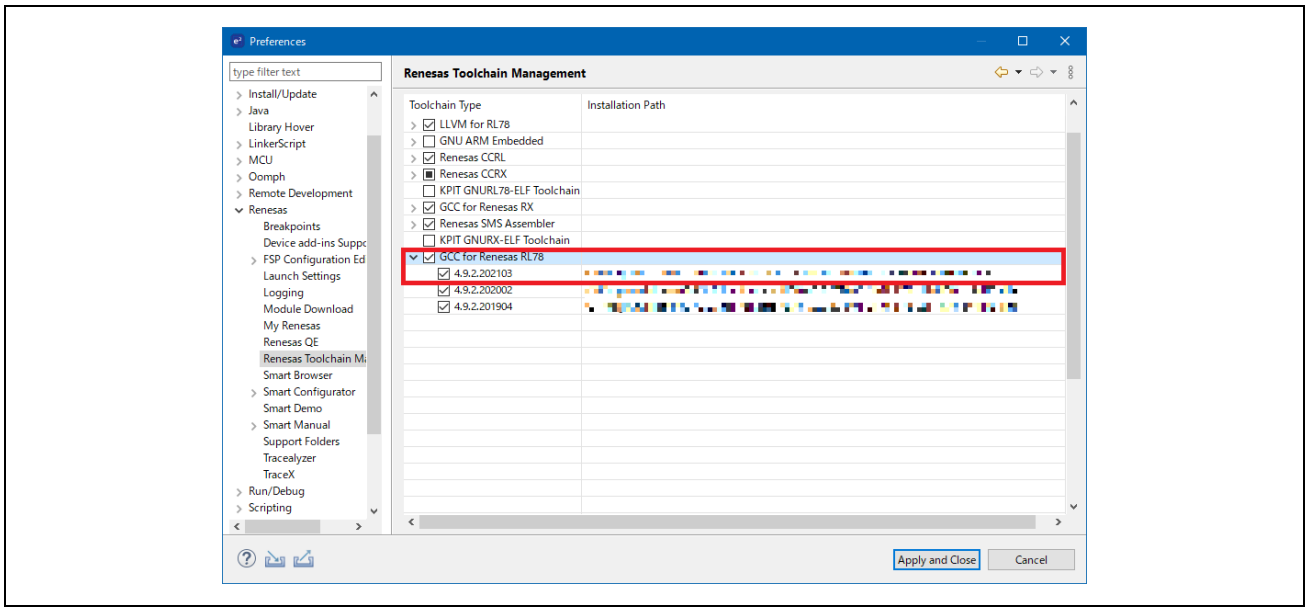

図 3-7 Renesas ツールチェーン管理

(3)　プロジェクトのインポート

以下の手順に沿って、サンプルプロジェクトを e2studio にインポートします。

[ファイル]→[インポート]を選択します。

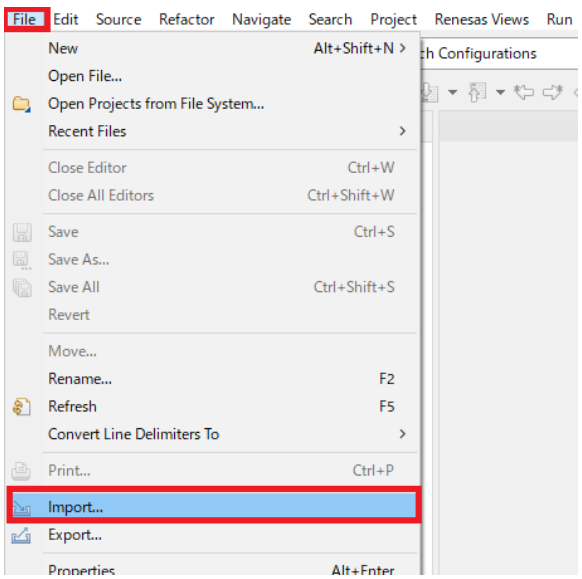

図 3-8 インポート

[選択]ダイアログで[一般]→[既存プロジェクトをワークスペースへ]を選択した後、[次へ]を選択します。

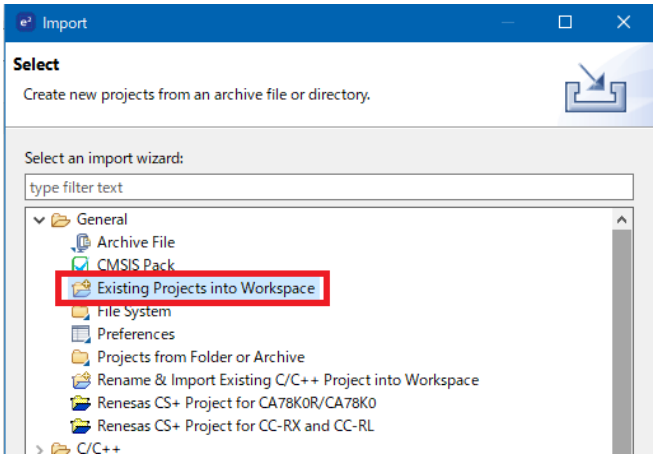

図 3-9 既存プロジェクトをワークスペースへ を選択

[プロジェクトのインポート]ダイアログの[ルート・ディレクトリーの選択]チェックボックスを選択した後、[参照]を選択します。任意のフォルダに格納した本サンプルソフトのパッケージ (RL78_uCCM_V***)を選択して[OK]を選択します。

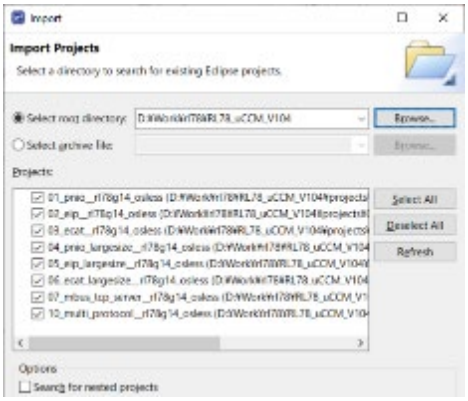

図 3-10 プロジェクトのインポート

[プロジェクト]にリスト化された各サンプルプロジェクトの中から使用するサンプルプロジェクトにチェックを付けた後、[終了]を選択すると、プロジェクトがインポートされます。

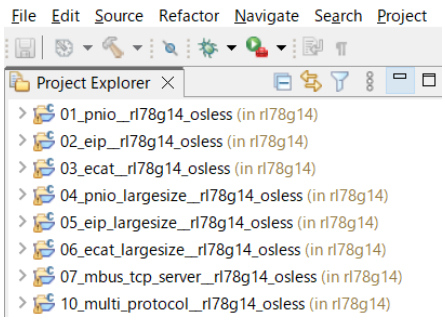

図 3-11 インポート完了

3.3.4　ビルド

e2studio 上の[プロジェクト・エクスプローラー]にて、サンプルプロジェクトを選択した後、[ビルド]ボタン (ハンマーアイコン) の横にある矢印を選択し、ドロップダウンメニューから[HardwareDebug]を選択します。

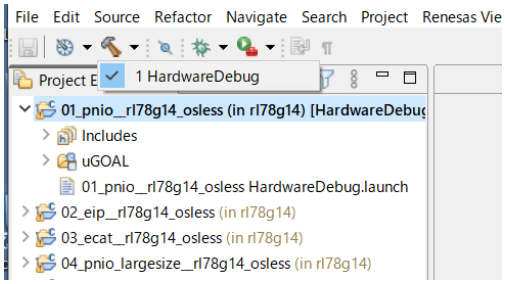

図 3-12 ビルド

e2studio が選択されたプロジェクトをビルドします。ビルドが完了したら、画面下部の[コンソール]に”Build Finished”のメッセージが出力されます。

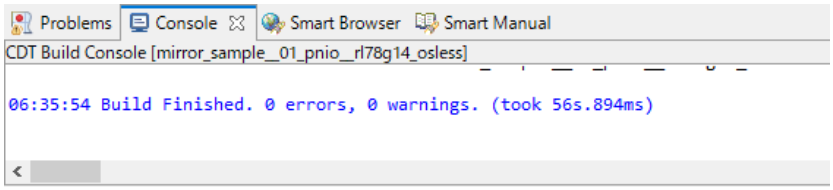

図 3-13 ビルド完了

3.3.5　デバッグ

ビルドが完了したら、デバッグをすぐに開始することが出来ます。 [デバッグ]ボタン (バグアイコン) の横にある矢印を選択し、[デバッグ構成]を選択します。

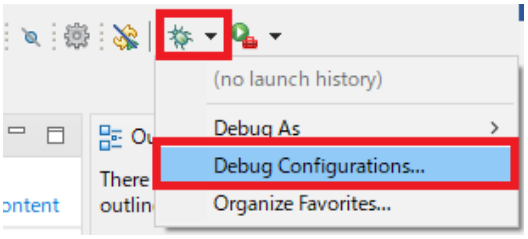

図 3-14 デバッグ構成

[デバッグ構成]ダイアログの[Renesas GDB Hardware Debugging]から該当する”xxxx HardwareDebug”を選択して、[デバッグ]ボタンを選択することで、デバッグ画面が立ち上がります。

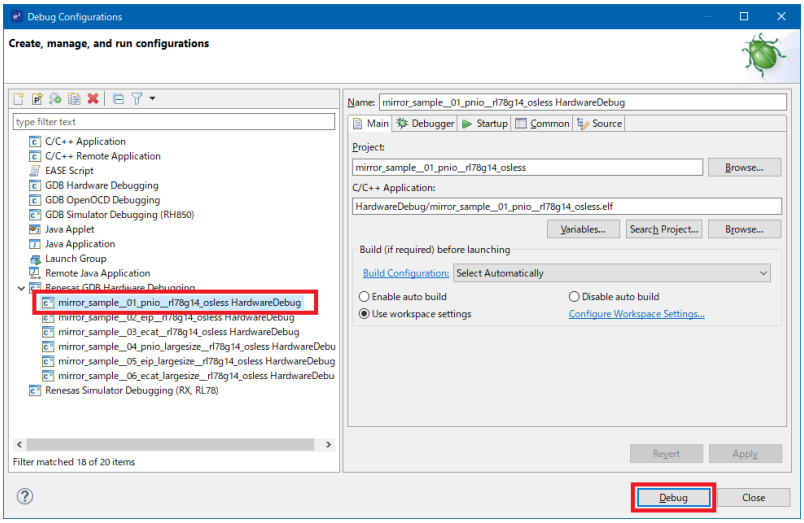

図 3-15 デバッグ開始

“e2-server-gdb.exe”のファイアウォール警告が表示されることがあります。 [自宅や職場のネットワークなどのプライベートネットワーク]のチェックボックスをチェックにして、<アクセスを許可>を選択します。

パースペクティブ切り替えの確認ダイアログにてパースペクティブの変更を勧めるダイアログが表示される場合は、「常にこの設定を使用する」チェックボックスにチェックし、[はい]を選択します。

デバッガ画面が立ち上がり、プログラムのダウンロードが完了したら、[再開]ボタンを選択することで、プログラムが実行されます。

3.4　プロトコル接続とアプリケーション制御

簡易マスターツール Management Tool (PROFINET, EtherNet/IP 接続) または TwinCAT (EtherCAT 接続) をつかったプロトコル接続と、各サンプルアプリケーションの制御方法について説明します。

3.4.1　PROFINET

PROFINET サンプルアプリケーションの評価手順について説明します。
対象サンプルを表 3-2 に示します。

表 3-2 PROFINET Sample software

Sample software	Overview
01_pnio	サイクリック通信 サンプル
04_pnio_largesize	サイクリック通信、RPC 通信(Large Size データ通信) サンプル
10_multi_protocol	01_pnio, 02_eip, 03_ecat, 07_modbus 統合サンプル

本サンプルアプリケーションを使用する際は、R-IN32M3 Module のファームウェアのバージョンを 2.1.0.0 以上にする必要があります。ファームウェアのアップデート方法は、『R-IN32M3 Module (RY9012A0) Management Tool 操作ガイド (R30AN0390JJ****)』を参照して下さい。

1.　評価環境セットアップ

-1.　評価ボード準備

開発環境を構築し (参照: 3.3 章)、サンプルプロジェクトのビルド、およびプログラムダウンロードを実行します (3.3.4～3.3.5 章)。正しくプログラムが実行されると、R-IN32M3 Module 搭載アダプタボード上のプロトコル表示 LED (PROFINET)が点灯します。

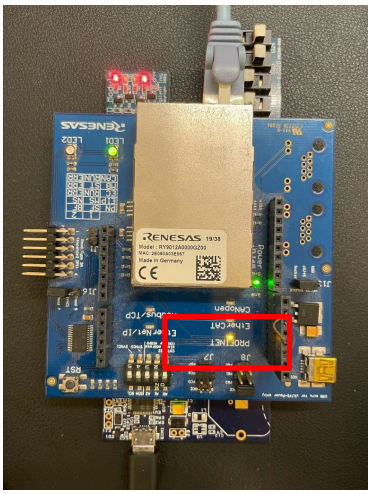

図 3-16 プロトコル LED: PROFINET

-2.　IP 設定

評価で使用する PC の IP 設定をします。ネットワークアダプタの[ネットワークプロパティ]を開き、固定 IP を設定します (例として 192.168.0.1 を用います)。

IP アドレス	192.168.0.1
サブネットマスク	255.255.255.0

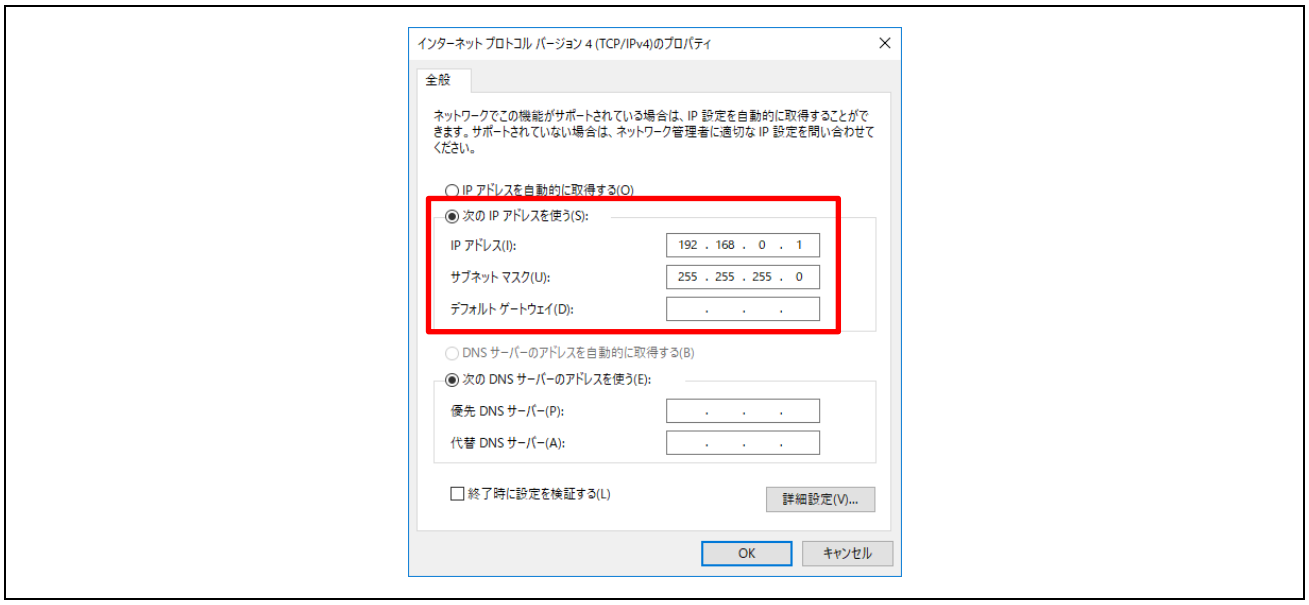

図 3-17 IP アドレス設定

2. マスタ接続

PROFINET マスタには、サンプルプログラムパッケージに同梱されている Management tool を使用します。Management tool の詳細は、『R-IN32M3 Module (RY9012A0) Management Tool 操作ガイド (R30AN0390JJ****)』を参照下さい。

以下の手順に沿って Management tool を操作して、本サンプルアプリケーションとの接続、データ送受信の確認を行います。

-1. [Network Navigator]パネルにて、使用するネットワークを選択した後、[Scan Network]ボタンを選択します。

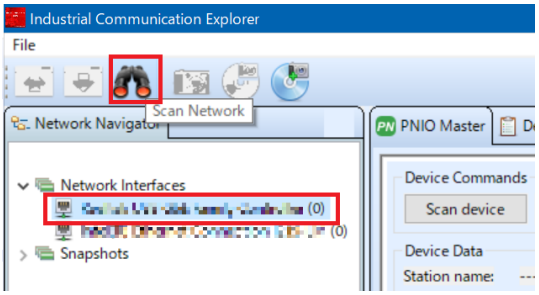

図 3-18 ネットワークスキャン

-2. [ネットワークスキャン]ダイアログが表示され、”Scan complete. found 1 device”が表示されれば検出完了ですので、[OK]を選択します。

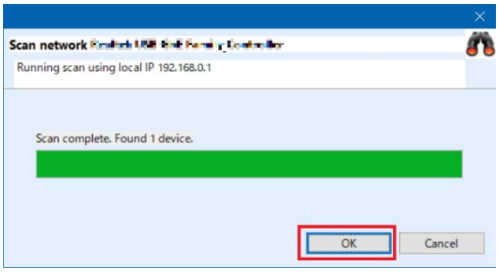

図 3-19 スキャン完了

-3. スキャンされたネットワーク内の[Network Navigator]パネルに、新しいデバイスとして”R-IN32M3_Module”が表示されますので、[R-IN32M3_Module]を選択します。

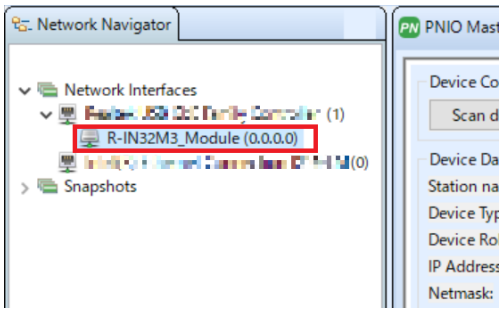

図 3-20 R-IN32M3 Module を選択

-4. R-IN32M3 Module の IP アドレスが、PC の IP アドレスと同じ IP ネットワーク内にある必要があります。そのため、R-IN32M3 Module の構成マネージャー変数 (揮発性メモリおよび不揮発性メモリに保存された構成変数) にアクセスして、IP アドレスと Netmask を設定します。
[R-IN32M3_Module]を選択したまま、[ConfigManager]パネルを表示した状態で[Read configuration]ボタンを選択します。

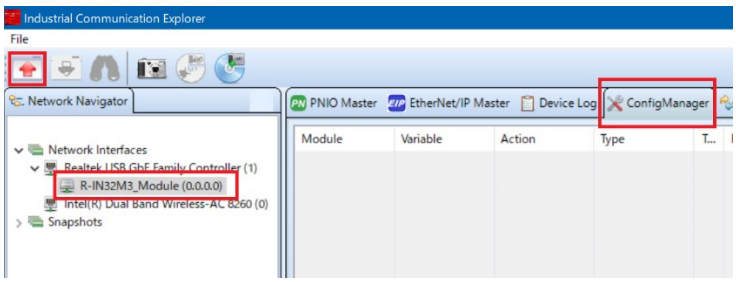

図 3-21 ConfigManager

-5.　[ConfigManager]パネルに表示された Configuration のうち、以下の項目を変更します。なお、IP アドレスと Netmask を有効にするために、VALID に 1 を設定する必要があります。変更された Value は黄色にハイライトされます。

Module	Variable	Value 設定例
GOAL_ID_NET	**IP**	192.168.0.100
GOAL_ID_NET	**NETMASK**	255.255.255.0
GOAL_ID_NET	**VALID**	0x01

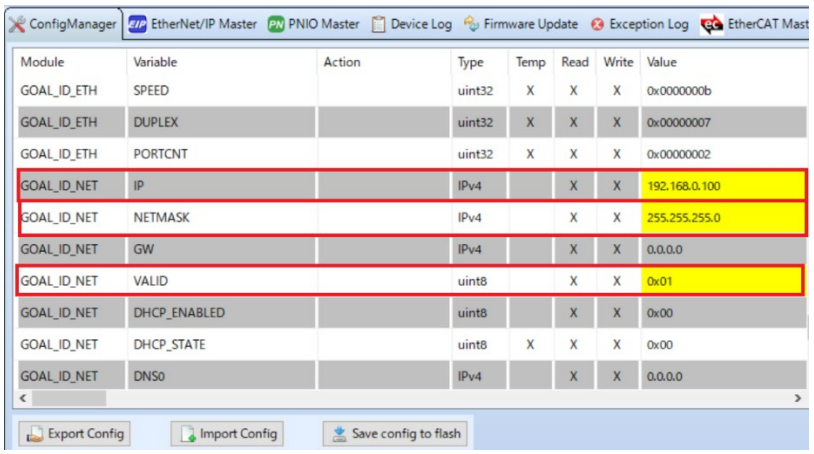

図 3-22 IP アドレス設定

-6.　[Write configuration]ボタンを選択して、変更された構成マネージャー変数が R-IN32M3 Module にダウンロードされます。

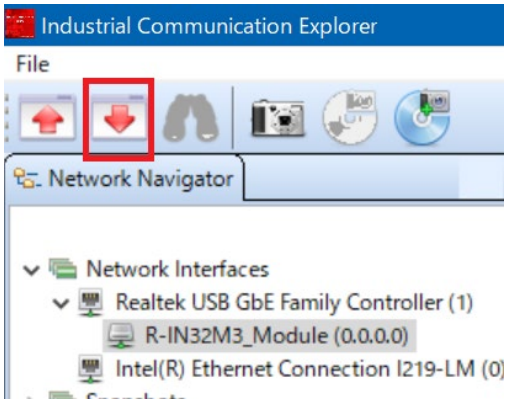

図 3-23 ダウンロード

-7. 変更確認のダイアログが表示された場合は[はい]を選択します。
変更された値は R-IN32M3 Module に転送され、RAM でのみ変更されます。R-IN32M3 Module に搭載された Flash の値を変更する際には、[Save config to flash]ボタンを使用します。

IP アドレス設定の詳細については 4.3 章を参照ください。

-8. [PNIO Master]パネルを選択して、[Scan device]を選択します。

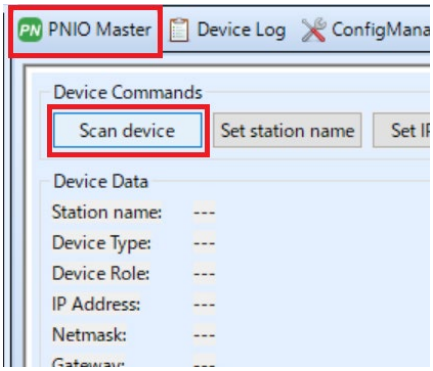

図 3-24 PNIO Master

-9. PROFINET デバイスが検出されると、[Device Data]に R-IN32M3 Module のデバイス情報が表示されます。

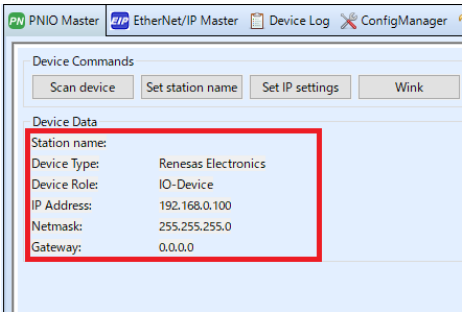

図 3-25 Device Data

-10. [I/O]パネルを開き、[Load GSDML file]ボタンを選択して、GSDML ファイルを読み込みます。

GSDML ファイルはサンプルプログラムの gsdml フォルダに格納されています。

Sample software	GSDML file
01_pnio	01_pnio¥gsdml¥GSDML-V2.4-Renesas-irj45-20211014.xml
10_multi_protocol	
11_pnio_http	
04_pnio_largesize	04_pnio_largesize ¥gsdml¥GSDML-V2.4-Renesas-irj45-20211014.xml

GSDML ファイルには、Slot、Module の割り当てが既に定義されています。
[Slots:]や[Modules]に GSDML に沿った内容が表示されることを確認し、[Device Interval]のプルダウンから[32]を選択した後、[Connect]ボタンを選択します。
接続に接続成功すると[Disconnect]ボタンに切り替わります。また、本ボードのプロトコルステータス LED が点灯します。

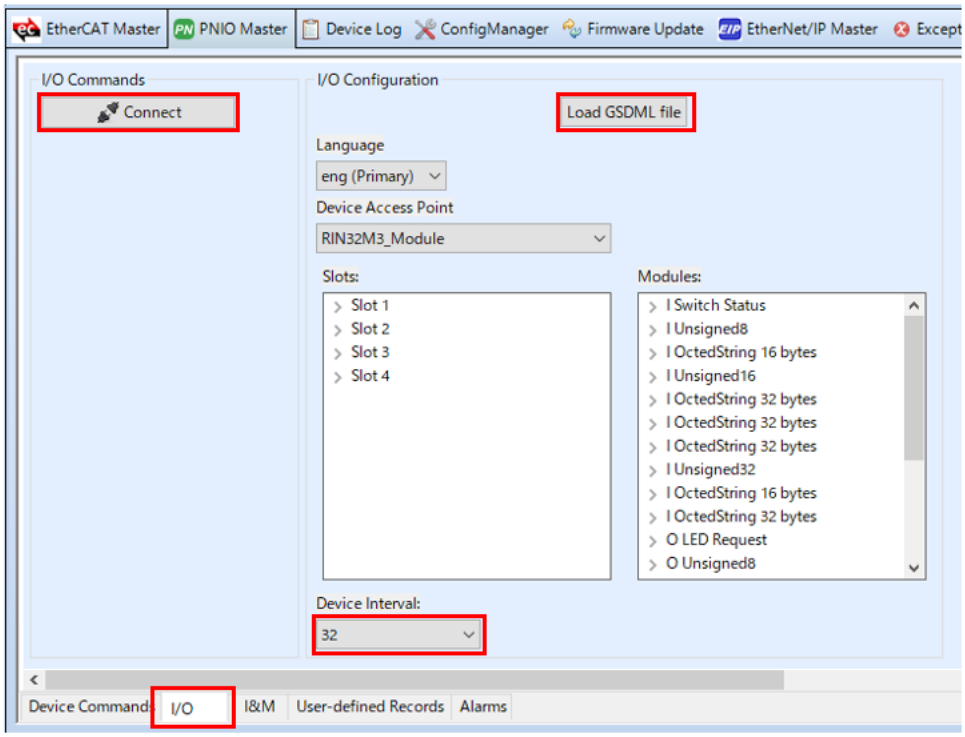

図 3-26 GSDML

-11. サンプルアプリケーションのデータ通信を行います。
サンプルソフトでは、アプリケーション例として 2 種類のデータ送受信アプリケーションを実装しています。

- **Remote-IO (LED/Switch)制御**：P-mod の LED 点灯制御および Switch 状態を送受信
  対象プロジェクト：01_pnio, 04_pnio_largesize, (10_multi_protocol)
- **Mirror 制御**：マスタから受信したデータをミラーバック送信
  対象プロジェクト：01_pnio, 04_pnio_largesize, (10_multi_protocol)
- **Mirror 制御(RPC)**：スキャナから受信したデータをミラーバック送信
  対象プロジェクト：04_pnio_largesize

これらのアプリケーションとして以下のように定義しています。

表 3-3 入出力アプリケーション

sample		Sample app.	Slot	Size
04_pnio_large	01_pnio	LED Data Reception	Slot 2	1
		Mirror Data Reception	Slot 4	16
		Switch Data Transmission	Slot 1	1
		Mirror Data Transmission	Slot 3	16
	-	Mirror Data Reception_1 (rpc)	Slot 6	32
		Mirror Data Reception_2 (rpc)	Slot 8	32
		Mirror Data Reception_3 (rpc)	Slot 10	32
		Mirror Data Transmission_1 (rpc)	Slot 5	32
		Mirror Data Transmission_2 (rpc)	Slot 7	32
		Mirror Data Transmission_3 (rpc)	Slot 9	32

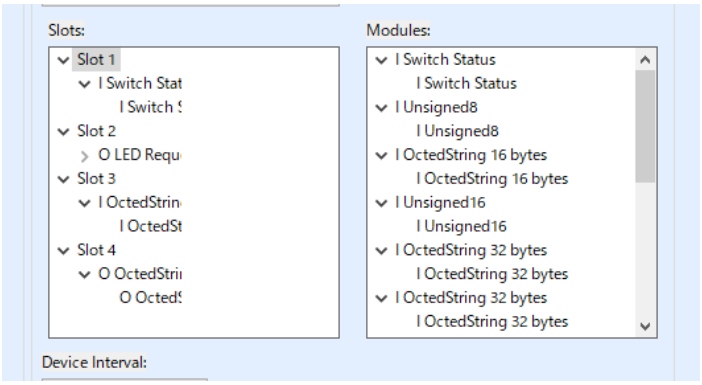

図 3-27 アプリケーション登録 (例 01_pnio)

Remote-IO (LED/Switch)制御

Switch には P-mod 接続したスイッチ状態に対応した Input Data、LED には P-mod 接続した LED に対応した Output Data が 1byte データとして登録されています。

I/O app.		Remote I/O 操作
Switch (Slot 1)	P-mod スイッチ	スイッチを操作することで Input Data の値が変化
LED (Slot 2)	P-mod LED	Output Data に値を登録することで LED が変化

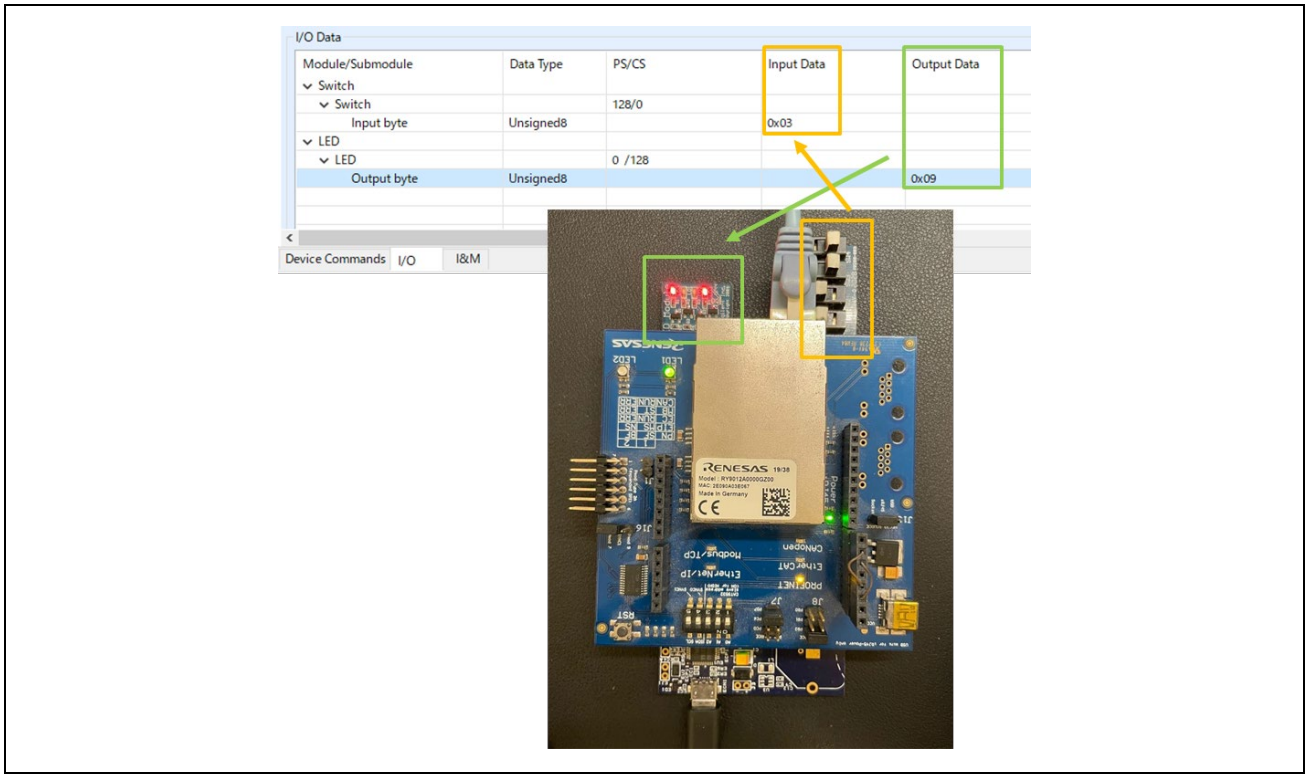

図 3-28 Remote-IO (LED/Switch)制御 [PROFINET]

Mirror 制御

Output Data に登録された値をマスタから受信したモジュールが、マスタへミラーバックし Input Data へ反映されます。

Mirror app.	Mirror 操作
Mirror Data Transmission (Slot 3: Input 16Byte)	モジュールからミラー制御で送信されてきた値が Input Data へ反映
Mirror Data Reception (Slot 4: Output 16Byte)	Output Data に登録した値をモジュールが受信

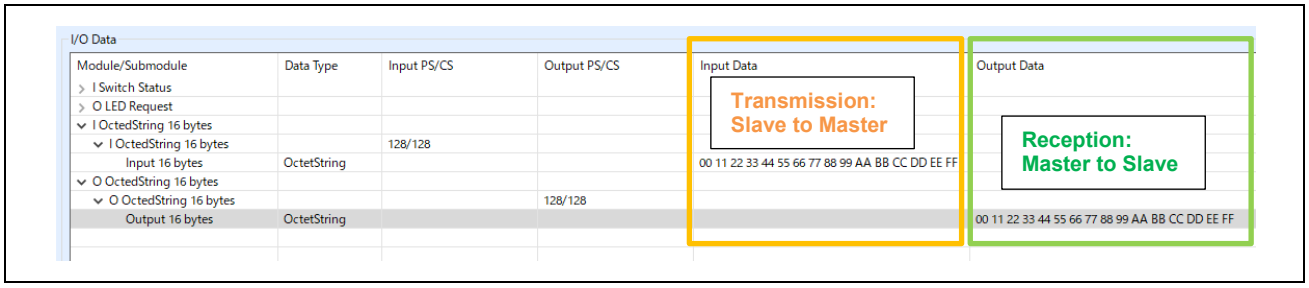

図 3-29 Mirror 制御 [PROFINET]

-12. [Disconnect]で通信を終了します。

3.4.2　EtherNet/IP

EtherNet/IP サンプルアプリケーションについて説明します。
対象サンプルを表 3-4 に示します。

表 3-4 EtherNet/IP Sample software

Sample software	Overview
02_eip	サイクリック通信 サンプル
05_eip_largesize	サイクリック通信、RPC 通信(Large Size データ通信) サンプル
10_multi_protocol	01_pnio, 02_eip, 03_ecat, 07_modbus 統合サンプル

本サンプルアプリケーションを使用する際は、R-IN32M3 Module のファームウェアのバージョンを 2.1.0.0 以上にする必要があります。ファームウェアのアップデート方法は、『R-IN32M3 Module (RY9012A0) Management Tool 操作ガイド (R30AN0390JJ****)』を参照して下さい。

1.　評価環境セットアップ

-1.　評価ボード準備

開発環境を構築し (参照: 3.3 章)、サンプルプロジェクトのビルド、およびプログラムダウンロードを実行します (3.3.4～3.3.5 章)。正しくプログラムが実行されると、R-IN32M3 Module 搭載アダプタボード上のプロトコル表示 LED (EtherNet/IP)が点灯します。

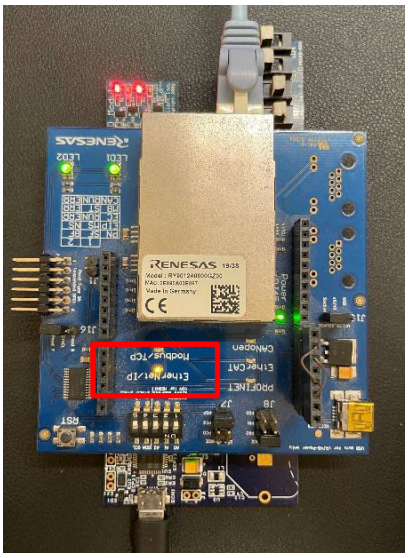

図 3-30 プロトコル LED: EtherNet/IP

-2.　IP 設定

評価で使用する PC の IP 設定をします。ネットワークアダプタの[ネットワークプロパティ]を開き、固定 IP を設定します (例として 192.168.0.1 を用います)。

IP アドレス	192.168.0.1
サブネットマスク	255.255.255.0

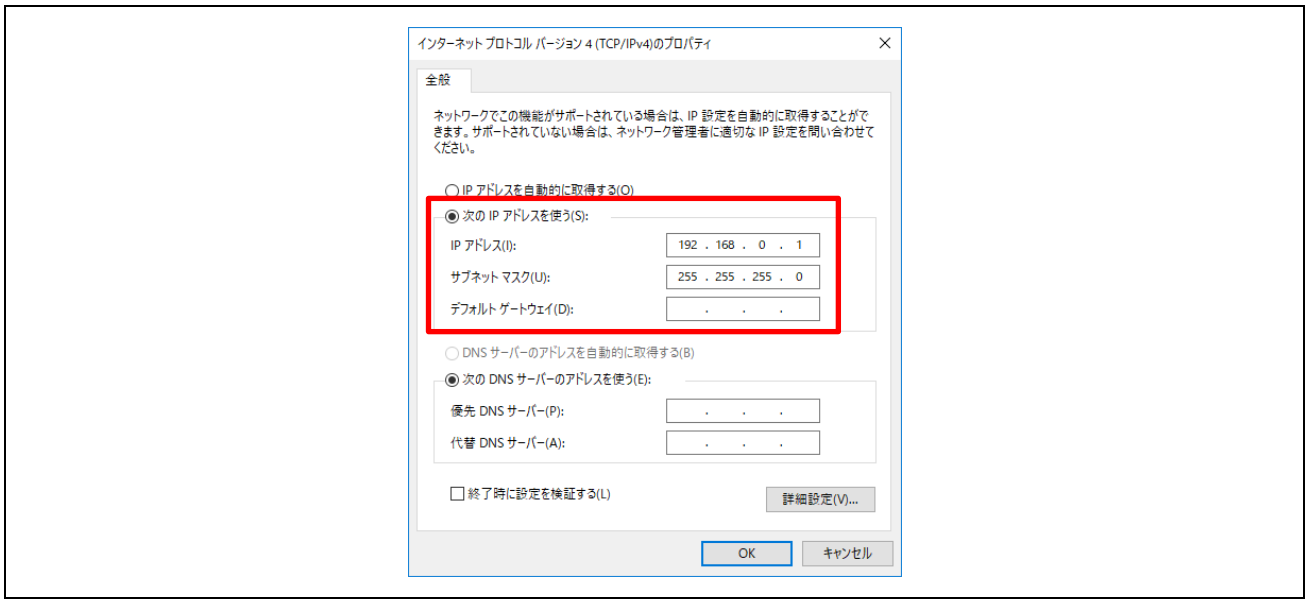

図 3-31 プロトコル LED: PROFINET

2. マスタ接続

PROFINET マスタには、サンプルプログラムパッケージに同梱されている Management tool を使用します。Management tool の詳細は、『R-IN32M3 Module (RY9012A0) Management Tool 操作ガイド (R30AN0390JJ****)』を参照下さい。

以下の手順に沿って Management tool を操作して、本サンプルアプリケーションとの接続、データ送受信の確認を行います。

-1. [Network Navigator]パネルにて、使用するネットワークを選択した後、[Scan Network]ボタンを選択します。

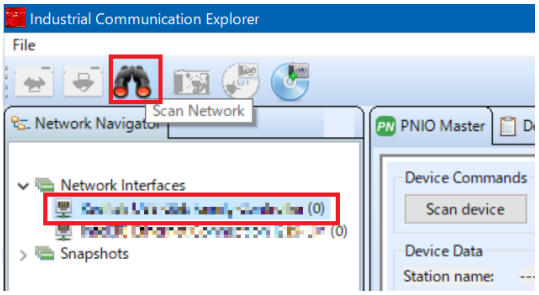

図 3-32 ネットワークスキャン

-2. [ネットワークスキャン]ダイアログが表示され、”Scan complete. found 1 device”が表示されれば検出完了ですので、[OK]を選択します。

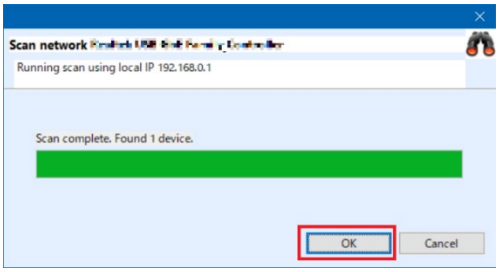

図 3-33 スキャン完了

-3. スキャンされたネットワーク内の[Network Navigator]パネルに、新しいデバイスとして”R-IN32M3_Module”が表示されますので、[R-IN32M3_Module]を選択します。

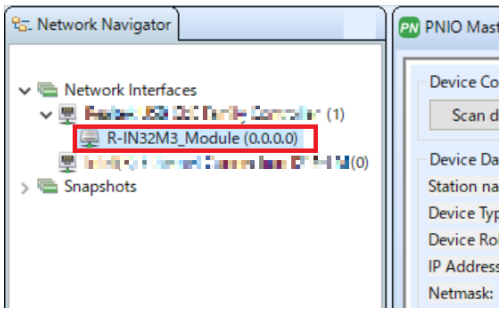

図 3-34 R-IN32M3 Module を選択

-4. R-IN32M3 Module の IP アドレスが、PC の IP アドレスと同じ IP ネットワーク内にある必要があります。そのため、R-IN32M3 Module の構成マネージャー変数 (揮発性メモリおよび不揮発性メモリに保存された構成変数) にアクセスして、IP アドレスと Netmask を設定します。
[R-IN32M3_Module]を選択したまま、[ConfigManager]パネルを表示した状態で[Read configuration]ボタンを選択します。

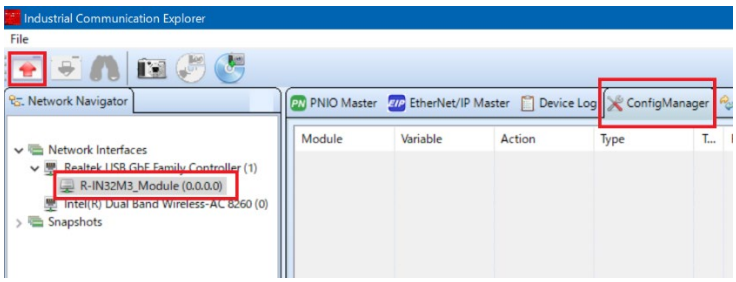

図 3-35 ConfigManager

-5. [ConfigManager]パネルに表示された Configuration のうち、以下の項目を変更します。なお、IP アドレスと Netmask を有効にするために、VALID に 1 を設定する必要があります。変更された Value は黄色にハイライトされます。

Module	Variable	Value 設定例
GOAL_ID_NET	**IP**	192.168.0.100
GOAL_ID_NET	**NETMASK**	255.255.255.0
GOAL_ID_NET	**VALID**	0x01

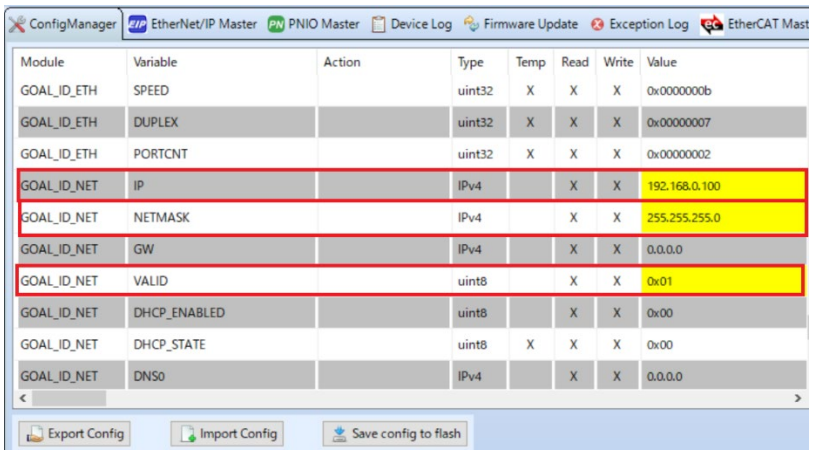

図 3-36 IP アドレス設定

-6. [Write configuration]ボタンを選択して、変更された構成マネージャー変数が R-IN32M3 Module にダウンロードされます。

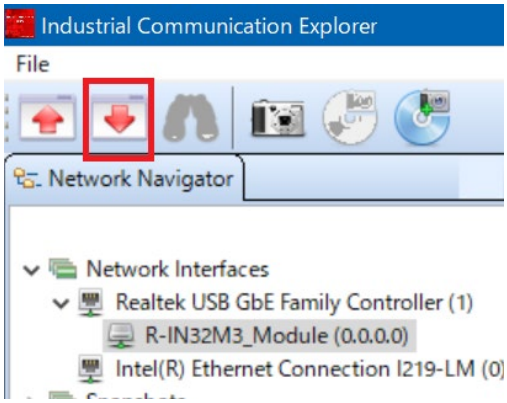

図 3-37 ダウンロード

-7. 変更確認のダイアログが表示された場合は[はい]を選択します。
変更された値は R-IN32M3 Module に転送され、RAM でのみ変更されます。R-IN32M3 Module に搭載された Flash の値を変更する際には、[Save config to flash]ボタンを使用します。

IP アドレス設定の詳細については 4.3 章を参照ください。

-8. [EtherNet/IP Master]パネルを選択して、[Scan device]を選択します。

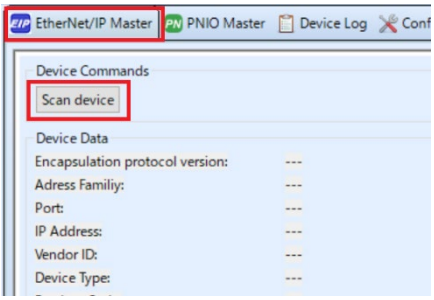

図 3-38 EtherNet/IP Scanner

-9. EtherNet/IP デバイスが検出されると、[Device Data]に R-IN32M3 Module のデバイス情報が表示されます。

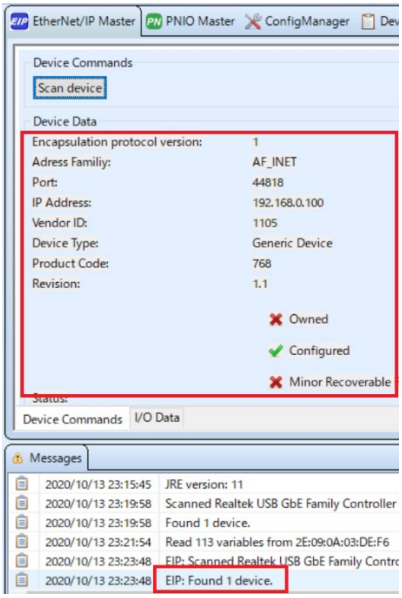

図 3-39 Device Data

-10. [I/O Data]パネルを開き スキャナ(マスタ)、アダプタ(スレーブ)間のデータ通信を定義します。
サンプルプログラムでは、アプリケーション例として 2 種類のデータ送受信アプリケーションを実装しています。

- **Remote-IO (LED/Switch)制御**：P-mod の LED 点灯制御および Switch 状態を送受信
  対象プロジェクト：02_eip, 05_eip_largesize, (10_multi_protocol)
- **Mirror 制御**：スキャナから受信したデータをミラーバック送信
  対象プロジェクト：02_eip, 05_eip_largesize, (10_multi_protocol)
- **Mirror 制御(RPC)**：スキャナから受信したデータをミラーバック送信
  対象プロジェクト：05_eip_largesize

これらのアプリケーションとして以下のように定義しています。

表 3-5 入出力アプリケーション

sample		Sample app.	Assembly ID	size
05_eip_large	02_eip	LED Data Reception	150	1
		Mirror Data Reception	151	16
		Switch Data Transmission	100	1
		Mirror Data Transmission	101	16
	-	Mirror Data Reception_1 (rpc)	152	32
		Mirror Data Reception_2 (rpc)	153	32
		Mirror Data Reception_3 (rpc)	154	32
		Mirror Data Transmission_1 (rpc)	102	32
		Mirror Data Transmission_2 (rpc)	103	32
		Mirror Data Transmission_3 (rpc)	104	32

表 3-6 コンフィグレーション

sample		Sample app.	Assembly ID	size
05_eip_large	02_eip	Config Data	200	10

Remote-IO (LED/Switch)制御

表 3-5 および 表 3-6 を参照し、接続パラメータを設定します。

Packet interval in ms はデフォルト値のままとします。

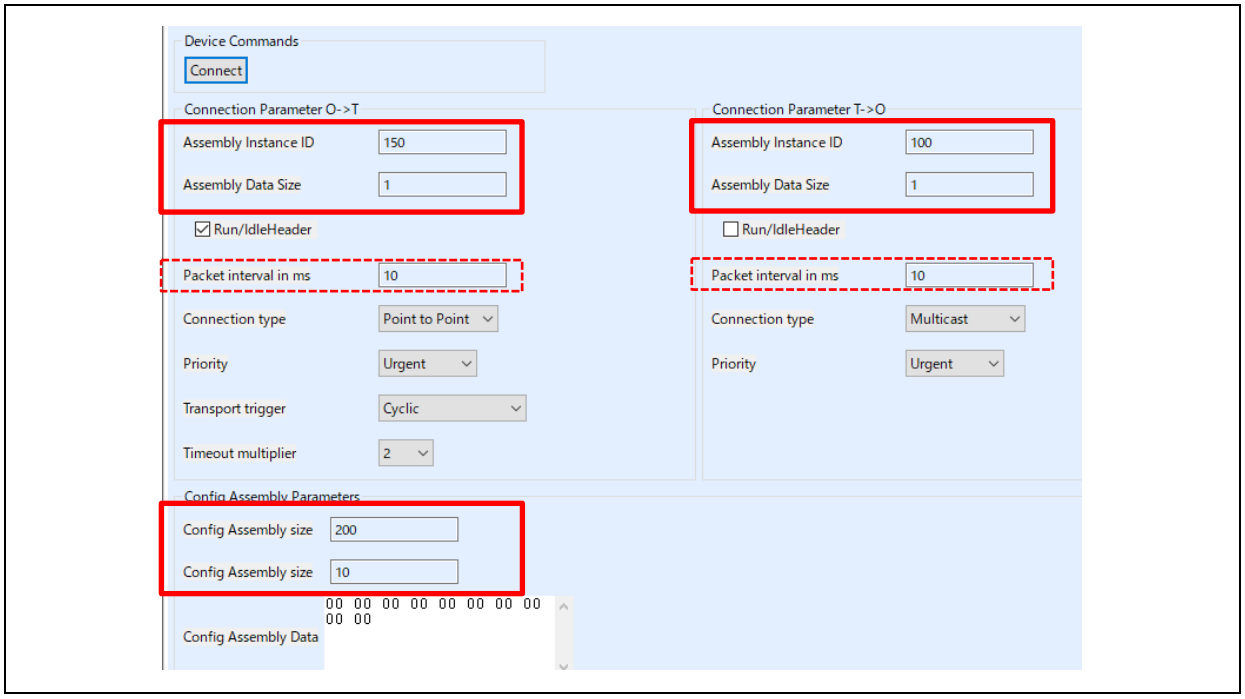

図 3-40 Remote-IO アプリケーションパラメータ設定

Mirror 制御

表 3-5 および 表 3-6 を参照し、ミラー制御用の接続パラメータを設定します。

Packet interval in ms はデフォルト値のままとします。

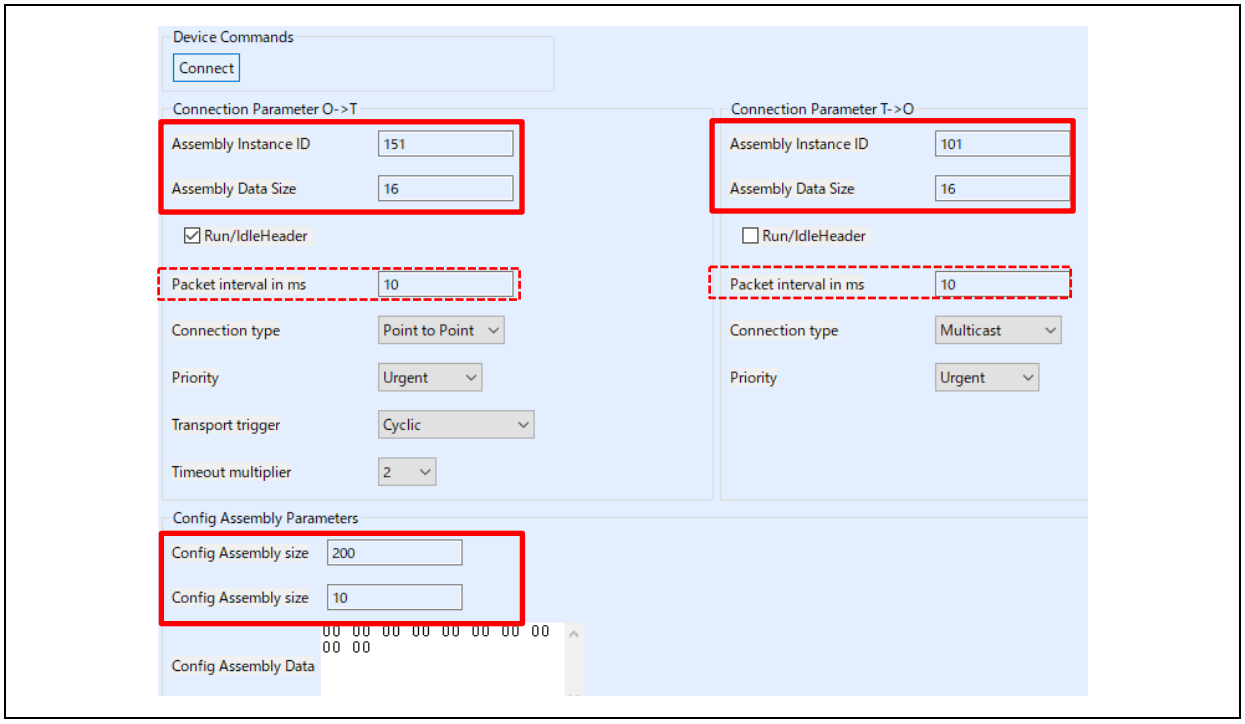

図 3-41 Mirror アプリケーションパラメータ設定

Mirror 制御(RPC)

05_eip_largesize では RPC を利用したプロセスデータ通信を行うこともできます。

これは、R-IN32M3 Module とホストマイコン間の SPI 通信フレーム(128byte)における RPC データフレームを使ってプロセスデータ通信を行う方式です。本来、非同期データ通信に用いることを目的とした RPC フレームを使用するため、通常の Cyclic データフレームを使用した方式に比べ大きいサイズのデータを送ることができますが(69byte 以上のプロセスデータを送受信可能)、アプリケーションの更新周期は制限があります。詳細は『R-IN32M3 Module (RY9012A0) ユーザ実装ガイド (R30AN0402JJ****)』をご参照ください。

表 3-5 および 表 3-6 を参照し、ミラー制御用の接続パラメータを設定します。図 3-42 では、Mirror Data Reception_2 (153)、Mirror Data Transmission_2 (103) の通信設定例を示します。

設定可能な Packet interval in ms 設定 (いわゆる RPI 設定) はデータサイズ、コネクション数に影響します。RPC を使用した設定値については十分ご評価の上、決定していただきますようお願いいたします。

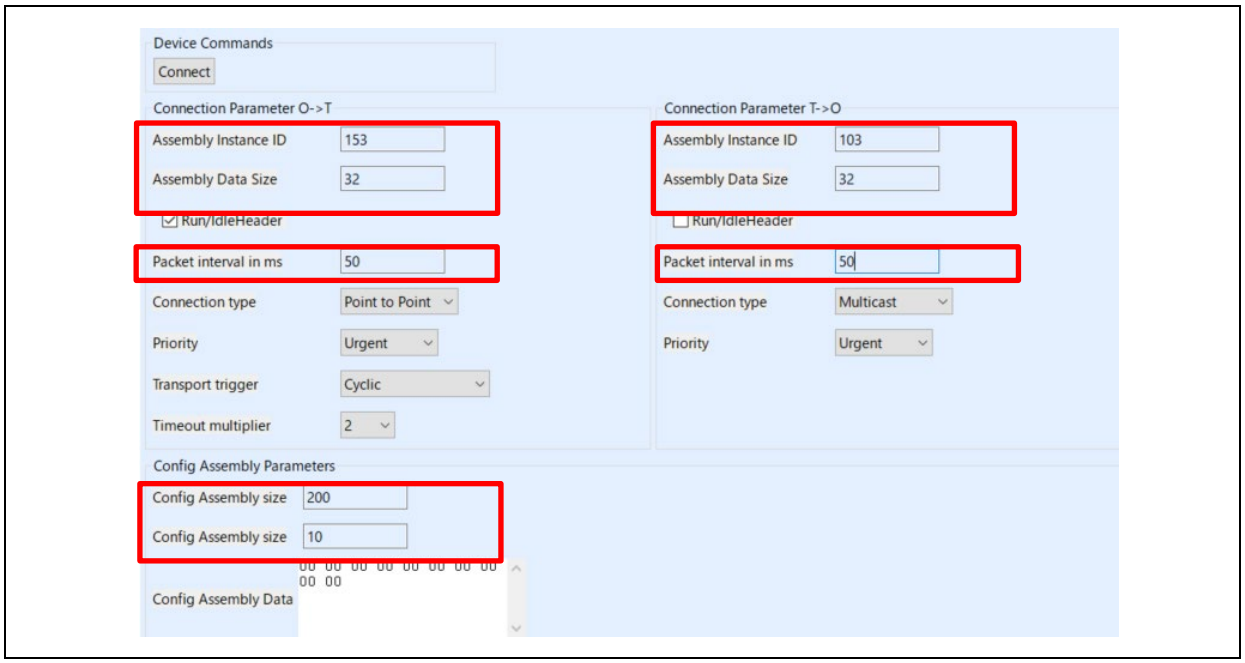

図 3-42 Mirror アプリケーション(RPC)パラメータ設定

-11. [Connect]ボタンを選択し、接続に接続成功すると[Disconnect]ボタンに切り替わります。また、本ボードのプロトコルステータス LED が点灯します。

-12. アプリケーションの入出力を確認します。

Remote-IO (LED/Switch)制御

Switch には P-mod 接続したスイッチ状態に対応した Input Data、LED には P-mod 接続した LED に対応した Output Data が 1byte データとして登録されています。

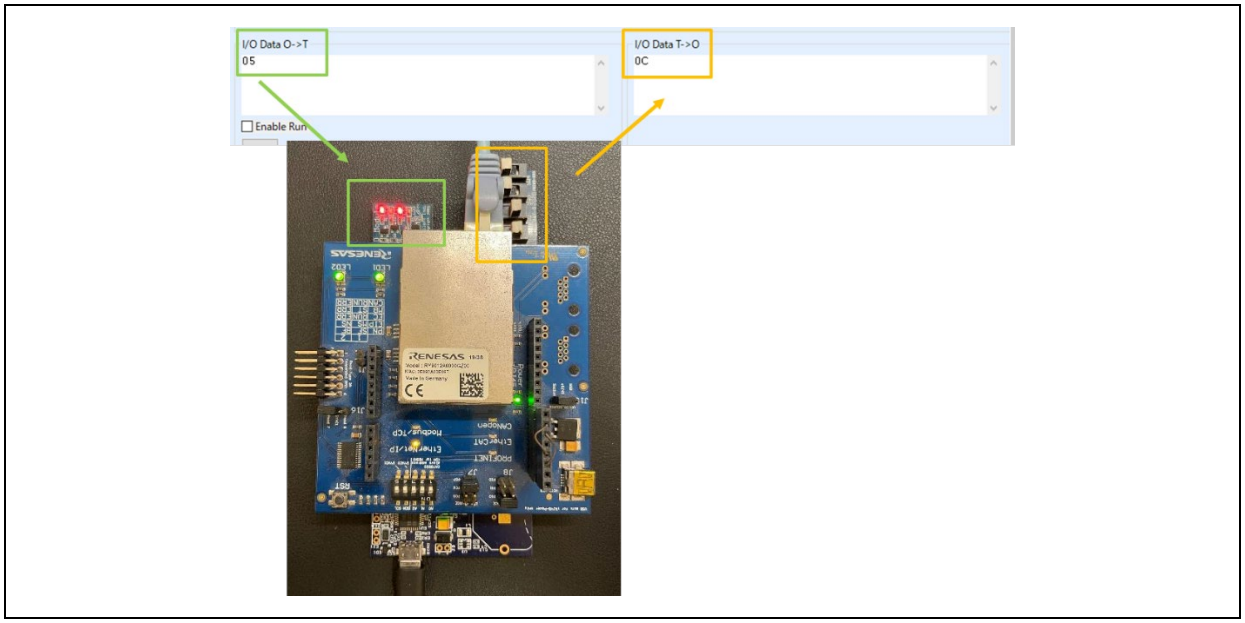

図 3-43 Remote-IO (LED/Switch)制御 [EtherNet/IP]

Mirror 制御

I/O Data O->T に入力した値をスキャナから受信したモジュールが、スキャナへミラーバックし I/O Data T->O へ反映されます。

ここでは、02_eip サンプルのミラー制御を例に示します。

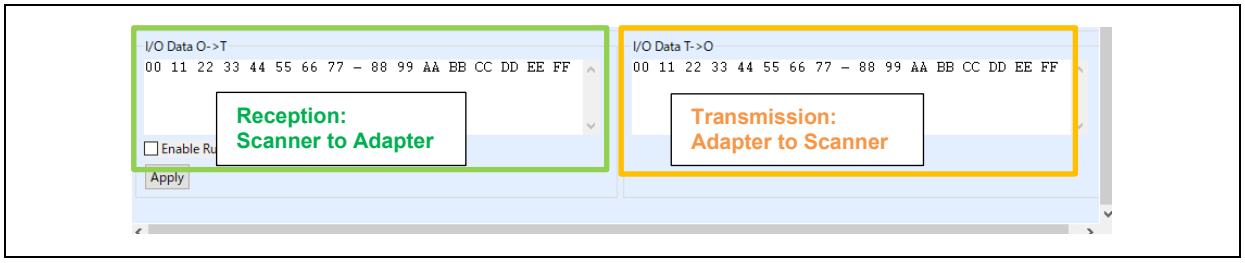

図 3-44 Mirror 制御 [EtherNet/IP]

-13. [Disconnect]で通信を終了します。

3.4.3　EtherCAT

EtherCAT サンプルアプリケーションについて説明します。
対象サンプルを表 3-7 に示します。

表 3-7 EtherCAT Sample software

Sample software	Overview
03_ecat	サイクリック通信 サンプル
06_ecat_largesize	サイクリック通信、RPC 通信(Large Size データ通信) サンプル
10_multi_protocol	01_pnio, 02_eip, 03_ecat, 07_modbus 統合サンプル

本サンプルアプリケーションを使用する際は、R-IN32M3 Module のファームウェアのバージョンを 2.1.0.0 以上にする必要があります。ファームウェアのアップデート方法は、『R-IN32M3 Module (RY9012A0) Management Tool 操作ガイド (R30AN0390JJ****)』を参照して下さい。

1.　評価環境セットアップ

-1.　評価ボード準備

開発環境を構築し (参照: 3.3 章)、サンプルプロジェクトのビルド、およびプログラムダウンロードを実行します (3.3.4～3.3.5 章)。正しくプログラムが実行されると、R-IN32M3 Module 搭載アダプタボード上のプロトコル表示 LED (EtherCAT)が点灯します。

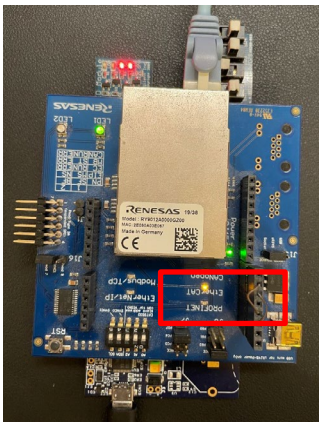

図 3-45 プロトコル LED: EtherCAT

-2. ネットワークアダプタ設定

TwinCAT 3 による EtherCAT フレームの送受信のリアルタイム性のために、使用するイーサネットカードで専用のドライバを有効に必要があります。TwinCAT むけドライバのインストールは『ソフトウェア PLC 接続ガイド TwinCAT (R30AN0380JJ****)』を参照して下さい。

専用ドライバ

・TwinCAT RT-Ethernet Filter Driver

・TwinCAT Ethernet Protocol for All Network Adapters

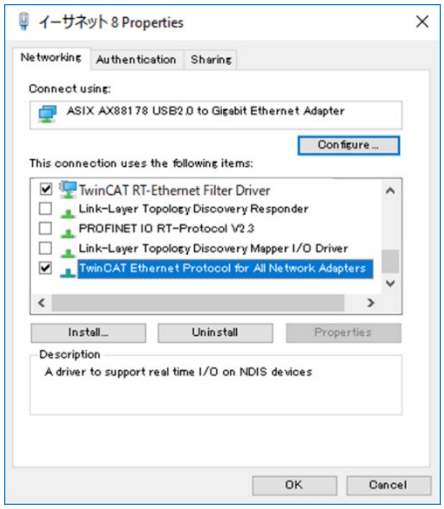

図 3-46 ネットワークアダプタ設定

[注意] ネットワークドライバの種類によっては、TwinCAT RT-Ethernet Filter Driver がインストールされない場合があります。この時は、**TwinCAT Ethernet Protocol for All Network Adapters** のみ有効にしてください。

-3. ESI ファイル

TwinCAT 3 を起動する前に ESI (EtherCAT Slave Information)ファイルを TwinCAT フォルダへ格納する必要があります。

ESI ファイルはサンプルプログラムの esi フォルダに格納されています。

Sample software	ESI file
03_ecat	03_ecat¥esi¥Renesas_RINmodule_03ecat.xml
10_multi_protocol	
13_ecat_http	
06_ecat_largesize	06_ecat_largesize ¥esi¥Renesas_RINmodule_06ecat.xml

[ESI 格納先フォルダ]

C:¥TwinCAT¥3.1¥Config¥Io¥EtherCAT

2. マスタ接続

EtherCAT マスタには、Beckhoff Automation 社製の TwinCAT を使用します。TwinCAT 接続の詳細は『ソフトウェア PLC 接続ガイド TwinCAT (R30AN0380JJ****)』を参照して下さい。

以下の手順に沿って TwinCAT を操作して、本サンプルアプリケーションとの接続、データ送受信の確認を行います。

-1. タートメニューから [Beckhoff]→[TwinCAT3]→[TwinCAT XAE Shell を選択して、TwinCAT を起動

-2. [File]→[New]→[Project]を選択して [TwinCAT XAE Project]タイプの新規プロジェクトを作成

-3. TwinCAT プロジェクトツリーにて [I/O]→[Devices]を右クリックして、[Scan]を選択

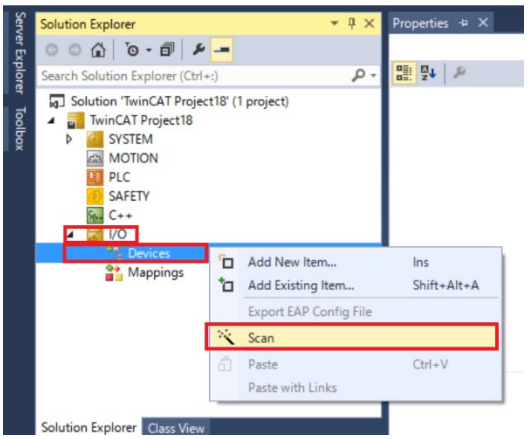

図 3-47 ネットワークスキャン

-4. [HINT：Not all types of devices can be found automatically]ダイアログで[OK]を選択
[Init12¥IO:Set State…]ダイアログで[OK]を選択

-5. EtherCAT モジュールが検出されると接続しているネットワークアダプタにチェック(☑) が付いた状態で表示されます。

-6. [Scan for Boxes]ダイアログで[Yes]を選択
[Active Free Run]ダイアログで[Yes]を選択

-7.　[I/O]→[Devices]の下に[Device x]→[Box 1]が追加されていれば接続は完了です。

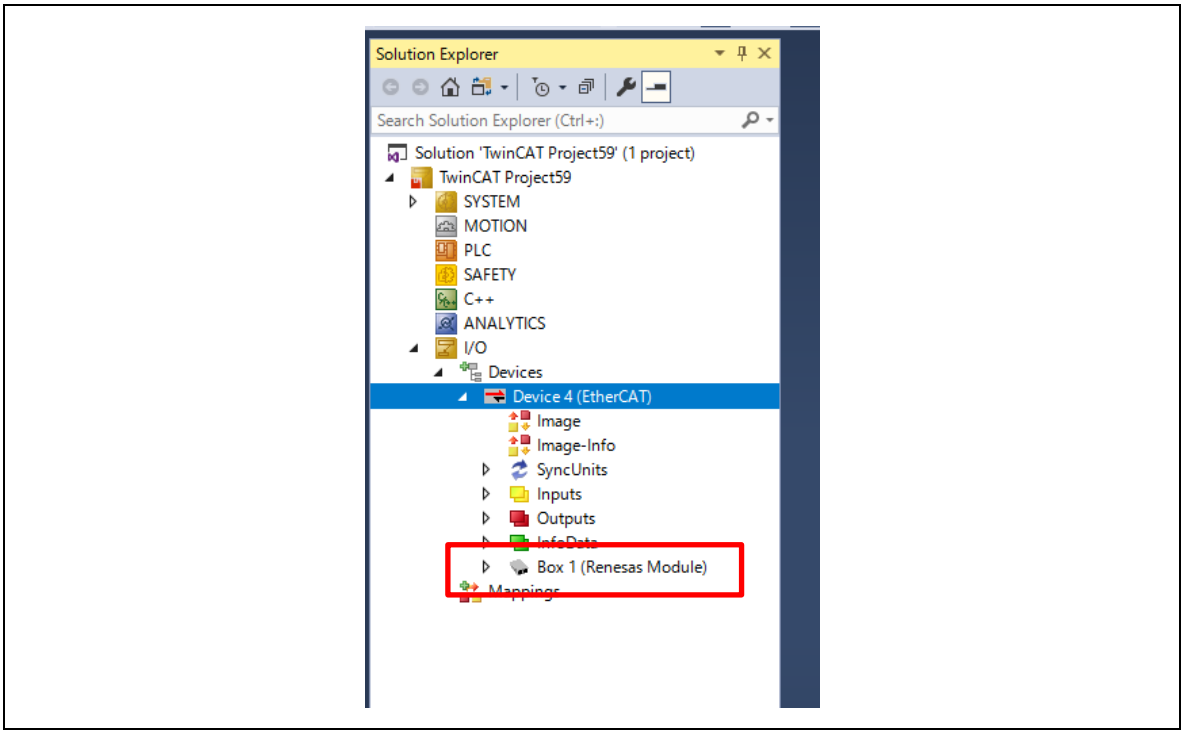

図 3-48 TwinCAT 接続

R-IN32M3 モジュールは、工場出荷状態では EEPROM に SII(Slave Information Interface)が書き込まれていないため、Box には "Box　(PFFFFFFFF RFFFFFFFF)" が表示されます。この場合は『ソフトウェア PLC 接続ガイド TwinCAT (R30AN0380JJ****)』を参照し EEPROM に SII をプログラムさせてください。

-8. サンプルアプリケーションのデータ通信を行います。
サンプルソフトでは、アプリケーション例として 2 種類のデータ送受信アプリケーションを実装しています。

- **Remote-IO (LED/Switch)制御**：P-mod の LED 点灯制御および Switch 状態を送受信
  対象プロジェクト：03_ecat, 06_ecat_largesize , (10_multi_protocol)
- **Mirror 制御**：マスタから受信したデータをミラーバック送信
  対象プロジェクト：03_ecat, 06_ecat_largesize, (10_multi_protocol)
- **Mirror 制御(RPC)**：スキャナから受信したデータをミラーバック送信
  対象プロジェクト：06_ecat_largesize

これらのアプリケーションとして以下のように定義しています。

表 3-8 入出力アプリケーション

sample		Sample app.	Index [sub]	Size
06_ecat_large	03_ecat	LED Output	0x6200 [1]	1
		Mirror Data out 1-16	0x6201 [1]	16
		Switch Data Transmission	0x6000 [1]	1
		Mirror Data in 1-16	0x6001 [1]	16
	-	Mirror Data out (rpc) 1-31	0x6210 [1]	31
		Mirror Data out (rpc) 32-62	0x6210 [2]	31
		Mirror Data out (rpc) 63-93	0x6210 [3]	31
		Mirror Data in (rpc) 1-31	0x6010 [1]	31
		Mirror Data in (rpc) 32-62	0x6010 [2]	31
		Mirror Data in (rpc) 63-93	0x6010 [3]	31

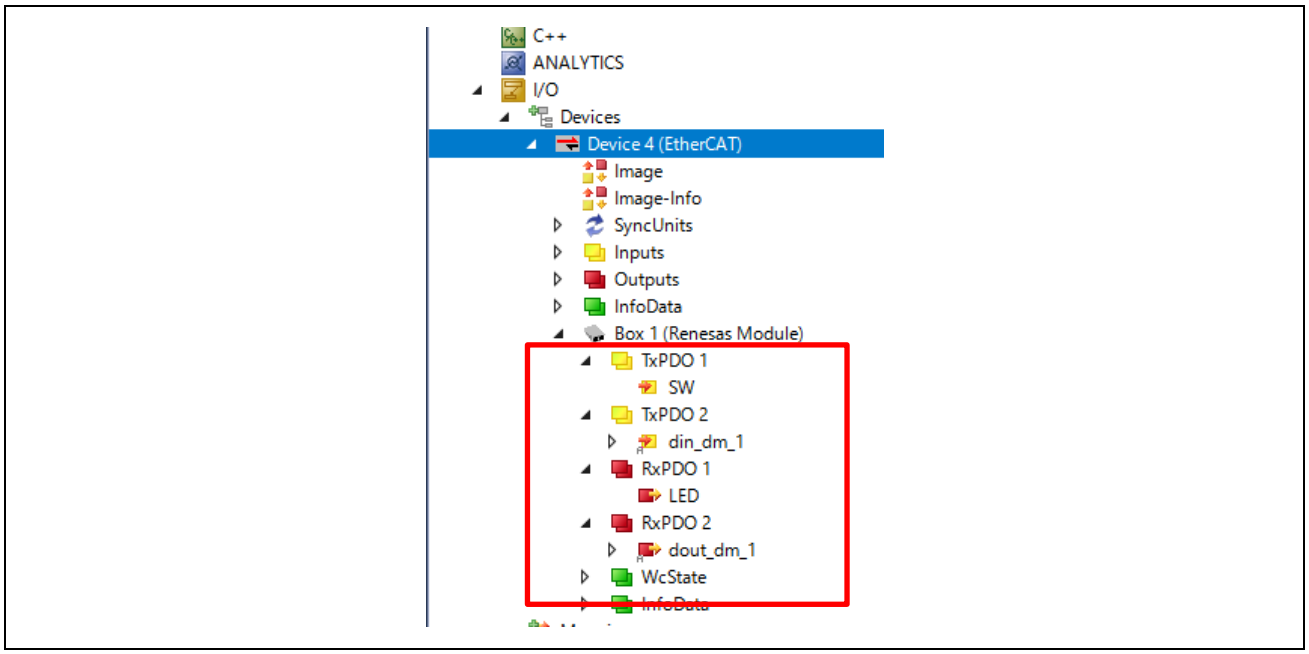

図 3-49 アプリケーション登録 (例 03_ecat)

Remote-IO (LED/Switch)制御

Switch には P-mod 接続したスイッチ状態に対応した Input Data、LED には P-mod 接続した LED に対応した Output Data が 1byte データとして登録されています。

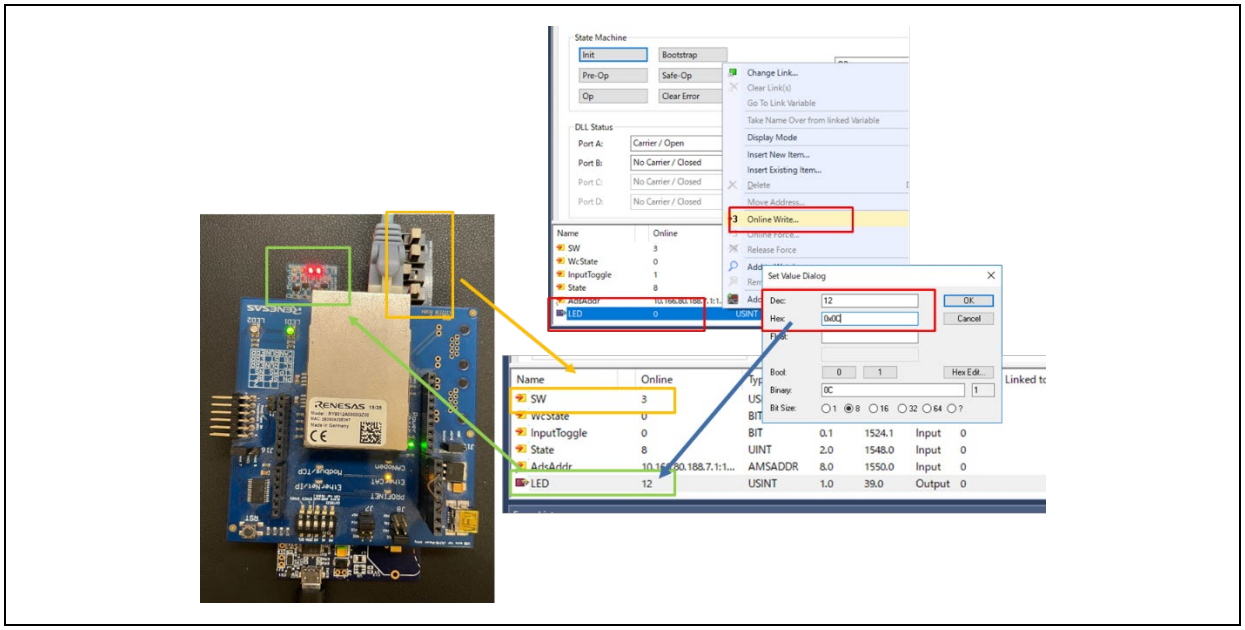

図 3-50 Remote-IO 制御 [EtherCAT]

Mirror 制御

Dout_dm_1 に入力した値をマスタから受信したモジュールが、マスタへミラーバックし din_dm_1 へ反映されます。

ここでは、03_ecat サンプルのミラー制御を例に示します。

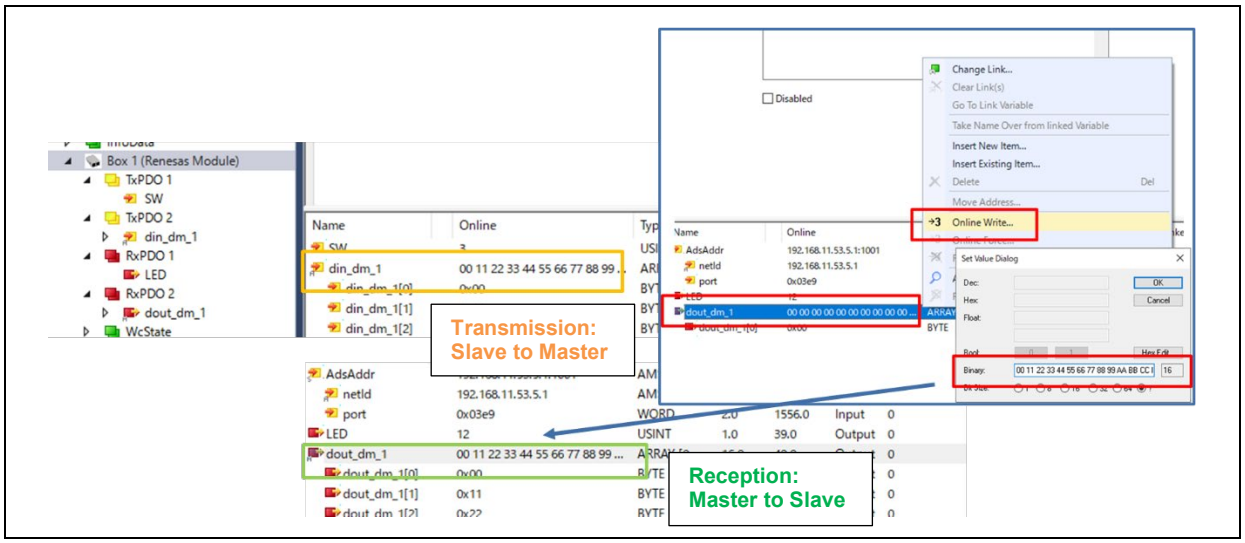

図 3-51 Mirror 制御 [EtherCAT]

3.4.4 Modbus TCP

Modbus TCP による Remote I/O サンプルアプリケーションに関しては、『R-IN32M3 Module (RY9012A0) Modbus TCP スタートアップマニュアル (R30AN0406JJ****)』を参照ください。

対象サンプルを表 3-9 に示します。

表 3-9 Modbus Sample software

Sample software	Overview
07_mbus_tcp_server	Modbus TCP サンプルアプリケーション

本サンプルアプリケーションを使用する際は、R-IN32M3 Module のファームウェアのバージョンを 2.1.0.0 以上にする必要があります。ファームウェアのアップデート方法は、『R-IN32M3 Module (RY9012A0) Management Tool 操作ガイド (R30AN0390JJ****)』を参照して下さい。

3.4.5　multi-protocol

multi-protocol(PROFINET, EtherNet/IP, EtherCAT, ModbusTCP)サンプルアプリケーションについて説明します。
対象サンプルを表 3-10 に示します。

表 3-10 multi-protocol Sample software

Sample software	Overview
10_multi_protocol	01_pnio, 02_eip, 03_ecat, 07_modbus 統合サンプル

本サンプルアプリケーションを使用する際は、R-IN32M3 Module のファームウェアのバージョンを 2.1.0.0 以上にする必要があります。ファームウェアのアップデート方法は、『R-IN32M3 Module (RY9012A0) Management Tool 操作ガイド (R30AN0390JJ****)』を参照して下さい。

1.　評価環境セットアップ

-1.　評価ボード準備

P-mode スイッチの値に応じて使用するプロトコルが実行されます。(参照: 2.2 章)

開発環境を構築し (参照: 2.2, 3.3 章)、サンプルプロジェクトのビルド、およびプログラムダウンロードを実行します (3.3.4～3.3.5 章)。

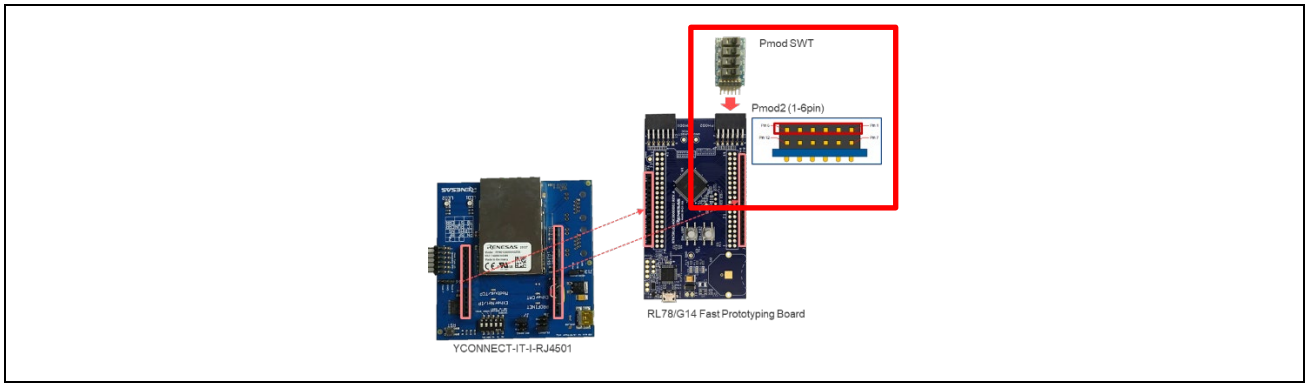

図 3-52 P-mod スイッチ multi-protocol selector

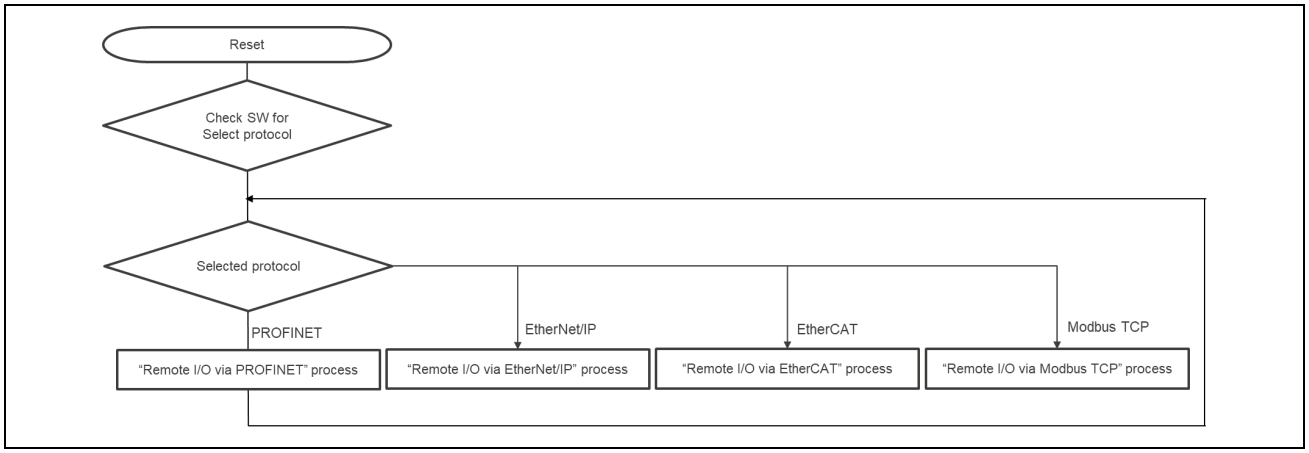

図 3-53 multi-protocol サンプルアプリケーションの流れ

Pmod SWT	プロトコル
on-off-off-off	PROFINET
off-on-off-off	EtherNet/IP
off-off-on-off	EtherCAT
off-off-off-on	Modbus TCP Server
others	PROFINET

PROFINET：Pmod SWT 設定 [on-off-off-off] など

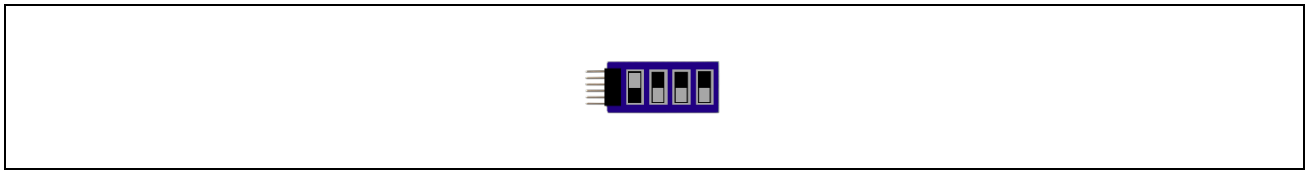

図 3-54 PROFINET

EtherNet/IP：Pmod SWT 設定 [off-on-off-off]

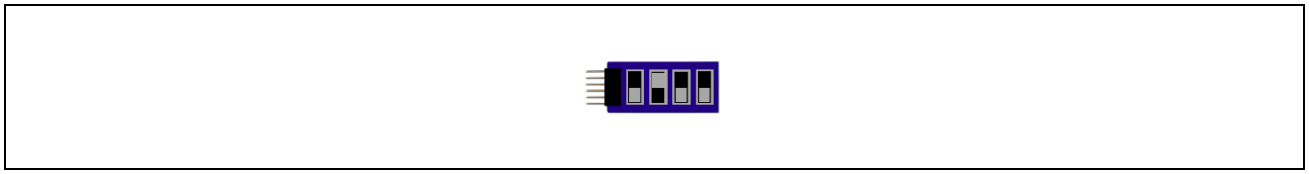

図 3-55 EtherNet/IP

EtherCAT：Pmod SWT 設定 [off-off-on-off]

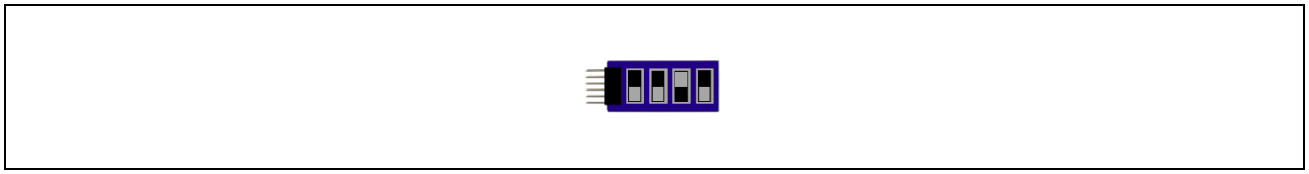

図 3-56 EtherCAT

Modbus TCP Server：Pmod SWT 設定 [off-off-off-on]

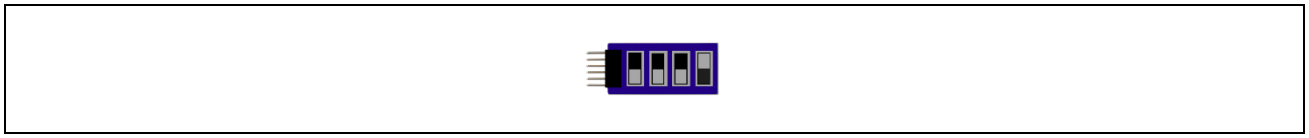

図 3-57 Modbus TCP Server

-2.　ネットワークアダプタ設定

選択したプロトコルにあわせて各章のネットワークアダプタ設定の手順を参照してください。

Protocol	Refer
PROFINET	3.4.1 PROFINET
EtherNet/IP	3.4.2 EtherNet/IP
EtherCAT	3.4.3 EtherCAT
ModbusTCP	3.4.4 Modbus TCP

2.　マスタ接続

選択したプロトコルにあわせて各章のマスタ接続の手順を参照してください。

Protocol	Refer
PROFINET	3.4.1 PROFINET
EtherNet/IP	3.4.2 EtherNet/IP
EtherCAT	3.4.3 EtherCAT
ModbusTCP	3.4.4 Modbus TCP

3.5　アプリケーションインプリガイド

本章では、ユーザアプリケーションとして固有の処理を実装する際の手順について説明します。

本サンプルソフトは、uGOAL ミドルウェアを備えており、その設計思想に基づいた構成となっています。uGOAL ではユーザアプリケーション固有の処理のために、appl_init()、appl_setup()、appl_loop()の関数が用意されており、uGOAL の初期フェーズで appl_init()と appl_setup()が実行され、その後のループフェーズで周期的に appl_loop()が実行される構成となっています。

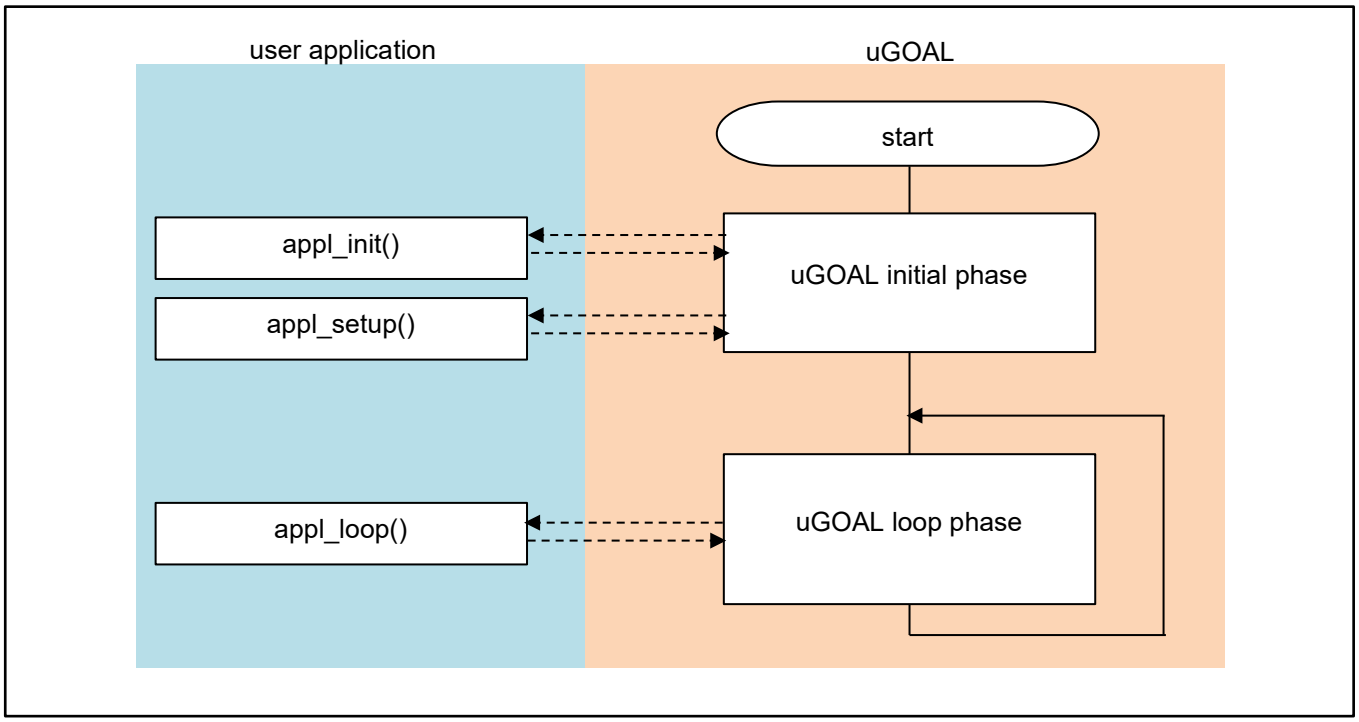

図 3-58　プログラム全体の流れ

ユーザアプリケーションの関数で行う固有処理の概要を以下に示します。また、これらは各サンプルアプリケーションのメインソースコードである goal_appl.c に定義されています。

表 3-11 ユーザアプリケーションと固有処理

ユーザアプリケーション	固有処理の概要
appl_init()	各プロトコルスタックの初期化、ボード依存のハードウェア初期化、といった uGOAL コア部分が初期化される前に行う初期化ステップを行います。
appl_setup()	ベンダ ID 設定等プロトコルスタック毎のプロファイル設定を行います。また、コールバック関数登録を行い、各プロトコルを介して R-IN32M3 Module からデータを受信します。
appl_loop()	ループ制御機能の実行を含む通常の操作を行います。

3.5.1 PROFINET

PROFINET による I/O ミラー応答サンプルアプリケーションにおけるユーザアプリケーション部分の実装について説明します。なお、各 API の詳細については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照ください。

(1) appl_init

本関数には、uGOAL コアモジュール等が初期化される前に、アプリケーション固有の初期化ステップを含めます。uGOAL で PROFINET のサポートを有効にするには、最初に goal_pnioInit を呼び出して uGOAL の PROFINET スタックを uGOAL に登録する必要があるため、goal_pnioInit を含む各モジュールの初期化ルーチンを呼び出します。

```
GOAL_STATUS_T appl_init(
    void
)
{
    GOAL_STATUS_T res;                          /**< result */

    /* initialize ccm RPC interface */
    res = appl_ccmRpcInit();
    if (GOAL_RES_ERR(res)) {
        goal_logErr("Initialization of ccm RPC failed");
    }

    res = goal_snmpInit();
    if (GOAL_RES_ERR(res)) {
        goal_logErr("Initialization of SNMP failed");
    }

    /* initialize PROFINET */
    res = goal_pnioInit();
    if (GOAL_RES_ERR(res)) {
        goal_logErr("Initialization of PROFINET failed");
    }

・・・

    return res;
}
```

①

①uGOAL の各モジュールを初期化します。goal_pnioInit は appl_init から呼び出される必要があります。

(2) appl_setup

PROFINET のインスタンス生成などプロトコルにおける静的な設定を定義します。PROFINET のインスタンスは、goal_pnioNew で生成して使用可能な状態にします。スロットメモリの予約量や、どのベンダ ID を使用するかといった設定は、goal_pnioInit と goal_pnioNew の間に行う必要があります。これらの設定は、goal_pnioCfg から始まる API 群で設定します。goal_pnioNew の後は、スロットやモジュールの作成など、他の全ての API を使用することができます。

```
GOAL_STATUS_T appl_setup(
    void
)
{
・・・

    res = goal_snmpNew(&pInstanceSnmp, APPL_SNMP_ID);                                    ①
    if (GOAL_RES_ERR(res)) {
        goal_logErr(“failed to create SNMP instance”);
        return res;
    }

    /* set SNMP instance id for new PNIO instance */
    res = goal_pnioCfgSnmpIdSet(APPL_SNMP_ID);
    if (GOAL_RES_ERR(res)) {
        goal_logErr(“failed to set SNMP instance id”);
        return res;
    }

・・                                                                                      ②

    /* set identification of the slave (vendor name) */
    res = goal_pnioCfgVendorNameSet(APPL_PNIO_VENDOR_NAME);
    if (GOAL_RES_ERR(res)) {
        goal_logErr(“failed to set vendor name”);
        return res;
    }

・・・

    /* create new PROFINET instance */
    res = goal_pnioNew(&pPnio, APPL_PNIO_ID, appl_pnioCb);
    if (GOAL_RES_ERR(res)) {
        goal_logErr(“failed to create a new PROFINET instance”);                         ③
        return res;
    }

・・・
```

①SNMP のインスタンスを生成。

②プロトコルにおける静的な設定を定義。本サンプルでは、ベンダ ID やデバイス ID 等を設定します。

③PRFINET のインスタンスを生成およびメインコールバック (appl_pnioCb) の登録。メインコールバック関数には、プロトコルスタックから報告される状態に応じた処理を記述します。報告される状態については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照してください。

```
    goal_logInfo(“Initializing device structure”);

    /* create subslots */
    res = goal_pnioSubslotNew(pPnio, APPL_API, APPL_SLOT_1, APPL_SLOT_1_SUB_1, GOAL_PNIO_FLG_AUTO_GEN);
    if (GOAL_RES_ERR(res)) {
        goal_logErr(“failed to add subslot”);
        return res;
    }

・・・

    /* create submodules */
    res = goal_pnioSubmodNew(pPnio, APPL_MOD_1, APPL_MOD_1_SUB_1, GOAL_PNIO_MOD_TYPE_INPUT,
                             APPL_SIZE_1_SUB_1_IN, 0, GOAL_PNIO_FLG_AUTO_GEN);
    if (GOAL_RES_ERR(res)) {
        goal_logErr(“failed to add submodule”);
        return res;
    }

・・・

    /* plug modules into slots */
    res = goal_pnioSubmodPlug(pPnio, APPL_API, APPL_SLOT_1, APPL_SLOT_1_SUB_1,
                              APPL_MOD_1, APPL_MOD_1_SUB_1);
    if (GOAL_RES_ERR(res)) {
        goal_logErr(“failed to plug submodule”);
        return res;
    }

・・・

    /* PROFINET configuration succesful */
    goal_logInfo(“PROFINET ready”);

・・・

    return res;
}
```

④

⑤

④サブスロットのインスタンスを生成。

⑤サブモジュールのインスタンスを生成し、サブスロットと関連付けします。

(3) appl_loop

uGOAL の初期化が終わった後のデータの処理を行います。

```
void appl_loop(
    void
)
{
    GOAL_STATUS_T res;                          /* result */
    uint8_t iops;                               /* IO producer status */

・・・

    if ((GOAL_TRUE == flgAppReady) && (plat_getElapseTime(tsTout) >= APPL_TIMEOUT_TRIGGER_VAL)) {
        /* read data from output module */
        res = goal_pnioDataOutputGet(pPnio, APPL_API, APPL_SLOT_4, APPL_SLOT_4_SUB_1, dataDm,
                                     APPL_SIZE_13_SUB_1_OUT, &iops);
        if (GOAL_RES_ERR(res)) {
            return;
        }
        /* copy data to input module */
        res = goal_pnioDataInputSet(pPnio, APPL_API, APPL_SLOT_3, APPL_SLOT_3_SUB_1, dataDm,
                                    APPL_SIZE_3_SUB_1_IN , GOAL_PNIO_IOXS_GOOD);
        if (GOAL_RES_ERR(res)) {
            return;
        }

        /* read data from output module */
        res = goal_pnioDataOutputGet(pPnio, APPL_API, APPL_SLOT_2, APPL_SLOT_2_SUB_1, dataDm,
                                     APPL_SIZE_11_SUB_1_OUT, &iops);
        if (GOAL_RES_ERR(res)) {
            return;
        }
        /* copy data to input module */
        res = goal_pnioDataInputSet(pPnio, APPL_API, APPL_SLOT_1, APPL_SLOT_1_SUB_1, dataDm,
                                    APPL_SIZE_1_SUB_1_IN, GOAL_PNIO_IOXS_GOOD);
        if (GOAL_RES_ERR(res)) {
            return;
        }

        /* update base timestamp */
        tsTout = goal_timerTsGet();
    }

・・・

}
```

①

①受信データの格納とミラー応答による送信データの設定を一定間隔で行います。

3.5.2 EtherNet/IP

EtherNet/IP による I/O ミラー応答サンプルアプリケーションにおけるユーザアプリケーション部分の実装について説明します。なお、各 API の詳細については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照ください。

(1) appl_init

uGOAL コアモジュール等が初期化される前に、アプリケーション固有の初期化ステップを含めます。uGOAL で EtherNet/IP のサポートを有効にするには、最初に goal_eipInit を呼び出して uGOAL の EtherNet/IP スタックを uGOAL に登録する必要があるため、goal_eipInit を含む各モジュールの初期化ルーチンを呼び出します。

```
GOAL_STATUS_T appl_init(
    void
)
{
    GOAL_STATUS_T res;                          /**< result */

    /* initialize rpc wrappers */
    res = appl_ccmRpcInit();
    if (GOAL_RES_ERR(res)) {
        goal_logErr("Initialization of ccm RPC failed");
    }

    /* initialize EtherNet/IP */
    res = goal_eipInit();
    if (GOAL_RES_ERR(res)) {
        goal_logErr("Initialization of EtherNet/IP failed");
    }

・・・

    return res;
}
```

①

①uGOAL の各モジュールを初期化します。goal_eipInit は appl_init から呼び出される必要があります。

(2) appl_setup

本関数では、EtherNet/IP のインスタンス生成などプロトコルにおける静的な設定を定義します。EtherNet/IP のインスタンスは、goal_eipNew で生成して使用可能な状態にします。ベンダ ID 等の設定は、goal_eipInit と goal_eipNew の間に行う必要があります。これらの設定は、goal_eipCfg から始まる API 群で設定します。goal_eipNew の後は、各種データにアクセス可能な状態となります。

```
GOAL_STATUS_T appl_setup(
    void
)
{
・・・

    /* for a real device the serial number should be unique per device */
    res = goal_eipCfgSerialNumSet(123456789);
    if (GOAL_RES_ERR(res)) {
        goal_logErr(“failed to set Serial Number”);
        return res;
    }
・・・

    res = goal_eipNew(&pHdlEip, 0, main_eipCallback);
    if (GOAL_RES_ERR(res)) {
        goal_logErr("failed to create eip instance %"FMT_x32, res);
        return res;
    }

    res = main_eipApplInit(pHdlEip);
    if (GOAL_RES_ERR(res)) {
        goal_logErr(“failed to initialize assembly and attribute configuration”);
        return res;
    }

・・・

}
```

①
②
③

①プロトコルにおける静的な設定を定義。本サンプルでは、ベンダ ID やプロダクトコード等を設定します。

②EtherNet/IP のインスタンスを生成。メインコールバック (main_eipCallback) の登録。コールバック関数では、プロトコルスタックから報告される状態に応じた処理を記述します。報告される状態については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照してください。

③生成した EtherNet/IP インスタンスを CIP オブジェクトに設定。

(3) appl_loop

uGOAL の初期化が終わった後のデータの処理を行います。

```
void appl_loop(
    void
)
{
    GOAL_STATUS_T res;                          /* result */

・・・

    if ((GOAL_TRUE == flgAppReady) && (plat_getElapseTime(tsTout) >= APPL_TIMEOUT_TRIGGER_VAL)) {
        /* get output data */
        res = goal_eipAssemblyObjectRead(pHdlEip, GOAL_APP_ASM_ID_OUTPUT, &outputData[0],
                                         GOAL_APP_ASM_SIZE_OUTPUT);

        /* mirror output data to input data */
        if (GOAL_RES_OK(res)) {
            GOAL_MEMCPY(&inputData[0], &outputData[0], GOAL_APP_ASM_SIZE_INPUT);

            /* store input data */
            res = goal_eipAssemblyObjectWrite(pHdlEip, GOAL_APP_ASM_ID_INPUT, &inputData[0],
                                              GOAL_APP_ASM_SIZE_INPUT);
        }

        /* update base timestamp */
        tsTout = goal_timerTsGet();
    }
}
```

①

①受信データの格納とミラー応答による送信データの設定を一定間隔で行います。

3.5.3 EtherCAT

EtherCAT による I/O ミラー応答サンプルアプリケーションにおけるユーザアプリケーション部分の実装について説明します。なお、各 API の詳細については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照ください。

(1) appl_init

uGOAL コアモジュール等が初期化される前に、アプリケーション固有の初期化ステップを含めます。uGOAL で EtherCAT のサポートを有効にするには、最初に goal_ecatInit を呼び出して uGOAL の EtherCAT スタックを uGOAL に登録する必要があるため、goal_ecatInit を含む各モジュールの初期化ルーチンを呼び出します。

```
GOAL_STATUS_T appl_init(
    void
)
{
    GOAL_STATUS_T res;                          /**< result */

    /* initialize ccm RPC interface */
    res = appl_ccmRpcInit();
    if (GOAL_RES_ERR(res)) {
        goal_logErr("Initialization of ccm RPC failed");
    }

    /* initialize EtherCAT */
    res = goal_ecatInit();
    if (GOAL_RES_ERR(res)) {
        goal_logErr("Initialization of EtherCAT failed");
    }

    return res;
}
```

①

①uGOAL の各モジュールを初期化します。goal_ecatInit は appl_init から呼び出される必要があります。

(2) appl_setup

EtherCAT のインスタンス生成などプロトコルにおける設定を定義します。EtherCAT のインスタンスは、goal_ecatNew で生成して使用可能な状態にします。また、必要に応じて goal_ecatCfg から始まる API 群で、インスタンス生成前に EtherCAT プロトコルの設定を行う必要があります。インスタンス生成後、必要なオブジェクトディクショナリを生成し、初期値を設定します。

```
GOAL_STATUS_T appl_setup(
    void
)
{

・・・

    /* enable CoE emergency */
    res = goal_ecatCfgEmergencyOn(GOAL_TRUE);
    if (GOAL_RES_ERR(res)) {
        goal_logErr(“failed to enable CoE Emergency support”);
        return res;
    }

・・・

#if APPL_ECAT_SII_INIT == 1
    goal_logInfo(“initializing EtherCAT SSI data”);

    res = appl_ccmCfgSsiVendorId(
        &__03_ecat_slave_eeprom_bin[0],         /* data buffer */
        __03_ecat_slave_eeprom_bin_len,         /* data buffer length */
        APPL_ECAT_VENDOR_ID);
    if (GOAL_RES_ERR(res)) {
        goal_logErr(“failed to configure EEPROM ssi vendor id”);
    }

・・・

    /* configure SII in EEPROM before creating the EtherCAT instance */
    res = appl_ccmEcatSsiUpdate(
        &__03_ecat_slave_eeprom_bin[0],         /* data buffer */
        __03_ecat_slave_eeprom_bin_len,         /* data buffer length */
        GOAL_FALSE);                            /* always overwrite ssi data */
    if (GOAL_RES_ERR(res)) {
        goal_logErr(“failed to configure EEPROM ssi data”);
    }
#endif
```

①

②

①EtherCAT プロトコルの設定を行っています。goal_ecatNew でインスタンスを生成する前に実施する必要があります。

②SII の初期化を実施します。（デフォルトでは無効）

```
    res = goal_ecatNew(&pHdlEcat, GOAL_ECAT_INSTANCE_DEFAULT, appl_ecatCallback);
    if (GOAL_RES_ERR(res)) {
        goal_logErr("failed to create a new EtherCAT instance");
        return res;
    }

    res = appl_ecatCreateObjects(pHdlEcat);
    if (GOAL_RES_ERR(res)) {
        goal_logErr("failed to initialize object dictionary");
        return res;
    }

    /* set settings for ccm firmware update via FoE */
    res = appl_ccmFoeUpdateSettings(
         "ccm.efw",                              /* filename beginning */
        0,                                      /* 0 -> match all characters */
        0,                                      /* password */
        GOAL_TRUE);                             /* only update in ESM state bootstrap */
    if (GOAL_RES_ERR(res)) {
        goal_logErr("failed to configure FoE firmware update of CC");
        return res;
    }

・・・

#if GOAL_CONFIG_MEDIA_MA_EVENT == 1
    /* open GPIO ma */
    if (GOAL_RES_OK(res)) {
        res = goal_maEventOpen(GOAL_ID_DEFAULT, &pHdlMaEvent, GOAL_TRUE, appl_gpioDcEvent);
        if (GOAL_RES_OK(res)) {
            goal_logInfo("event generation enabled");
        }
    }
#endif

・・・

    return res;
}
```

③
④
⑤
⑥

③EtherCAT のインスタンスを生成し、メインコールバック (main_ecatCallback) の登録をします。コールバック関数では、プロトコルスタックから報告される状態に応じた処理を記述します。報告される状態については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ソフトウェア編 (R17US0002JJ****)』を参照してください。

④各オブジェクトディクショナリ (OD) を生成します。goal_ecatdynOdObjAdd 等を使用して OD を追加しますが、最後に goal_ecatdynOdFinish で OD 生成の終了をします。

⑤FoE を使用したファームウェアアップデートの設定を行っています。

⑥EtherCAT Explicit Device ID 設定用のモジュールの初期化します。EtherCAT Explicit Device ID を設定する際は、別途 Pmod SWT が必要です。詳細は 2.4 を参照してください。

(3) appl_loop

uGOAL の初期化が終わった後のデータの処理を行います。

```
void appl_loop(
    void
)
{

・・・

    if ((GOAL_TRUE == flgAppReady) && (plat_getElapseTime(tsTout) >= APPL_TIMEOUT_TRIGGER_VAL)) {
        /* map process data */
        read_state8_input1 = write_state8_output1;
        read_state8_input2 = write_state8_output2;

        read_analog16_input1 = write_analog16_output1;
        read_analog16_input2 = write_analog16_output2;

        /* process cyclic process data */
        appl_obj_200d = cntDC0Event;
        appl_obj_200e = cntDC1Event;

        /* update base timestamp */
        tsTout = goal_timerTsGet();
    }

・・・

}
```

①

①受信データの格納とミラー応答による送信データの設定を一定間隔で行います。

4. Appendix

4.1 uGOAL API

ホストマイコンは、uGOAL が提供する R-IN32M3 Module を制御するための API 関数を介して、R-IN32M3 Module と通信します。API はプロトコル毎に分類されており、詳細は、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアルソフトウェア編 (R17US0002JJ****)』を参照ください。

4.2　ロギング

本サンプルソフトでは、RL78/G14 のメモリ配置制限を理由に、デバッグ用途を目的としたログメッセージを出力する機能は使用できません。

4.3　IP アドレス設定

本章では、R-IN32M3 Module の IP アドレス設定について説明します。

R-IN32M3 Module の IP アドレスは、起動時に内部の不揮発性メモリに保存された GOAL_ID_NET (12)設定に従い設定されますが、ホスト CPU から IP アドレスを設定する *goal_maNetIpSet()* を呼び出して設定することも可能です。
本サンプルの”01_pnio”、”02_eip”、”04_pnio_large”、”05_eip_large”のサンプルアプリケーションでは、デフォルト設定では、内部に保存された設定を元に IP アドレスが設定されるようになっており(Configured IP)、プログラム内に“GOAL_CONFIG_STATIC_IP”マクロを 1 に定義することでホストマイコンから任意の IP アドレス(Static IP)を設定することができます。

表 4-1　IP Configuration (GOAL_ID_NET)

Variable Name	Variable ID	Type	Max. Size	Description
IP	0	GOAL_CM_IPV4	4	IP address of first interface
NETMASK	1	GOAL_CM_IPV4	4	NETMASK of first interface
GW	2	GOAL_CM_IPV4	4	GATEWAY of first interface
VALID	3	GOAL_CM_UINT8	1	Validity of IP address: 0, Stored IP address is not valid, interface settings originate from network stack of system 1, Stored IP address is valid, will be applied to interface at start of device
DHCP_ENABLED	4	GOAL_CM_UINT8	1	DHCP enable: 0, DHCP disabled 1, DHCP enabled

不揮発性メモリに保存された IP アドレス設定を有効とするためには、VALID = 1 となっている必要がありますのでご注意ください。*goal_maNetIpSet()* を実行すると IP、NETMASK および GW 設定は不揮発性メモリにも保存されますが、VALID 設定については最後の引数 flgTemp で保存するかどうか指定することができます。(GOAL_FALSE：VALID 設定を更新、GOAL_TRUE：VALID 設定は更新せず)

```
1.  GOAL_STATUS_T goal_maNetIpSet(
2.      GOAL_MA_NET_T *pNetHdl,          /**< pointer to store NET handler */
3.      uint32_t addrIp,                 /**< IP address */
4.      uint32_t addrMask,               /**< subnet mask */
5.      uint32_t addrGw,                  /**< gateway */
6.      GOAL_BOOL_T flgTemp              /**< temporary IP config flag */
7.  );
```

また、DHCP を有効とする場合は、GOAL_ID_NET (12) の DHCP_ENABLED を 1 に設定するか、EtherNet/IP の場合は、*goal_eipCfgDhcpOn()* を呼び出します。02_eip サンプルでは、プログラム内に“GOAL_CONFIG_ENABLE_DHCP”マクロを 1 で定義することで、DHCP が有効になります。

表 4-2 に IP アドレスの設定方法の一覧を示します。

表 4-2　IP address setting list

Methods	Descriptions
Configured IP	・R-IN32M3 Module 内の不揮発性メモリに保持された値を使用します。 ・Management Tool を使って値の変更が可能です。詳細は、『R-IN32M3 Module (RY9012A0) Management Tool 操作ガイド (R30AN0390JJ****)』を参照してください。 ・本サンプルの”01_pnio”、”02_eip”、”04_pnio_large”、”05_eip_large”のサンプルアプリケーションのデフォルト設定は、この方法になります。
Static IP	・主に評価用に用いられます。 ・変更した値は R-IN32M3 Module 内の不揮発性メモリに保持されます。 ・本サンプルの”01_pnio”、”02_eip”、”04_pnio_large”、”05_eip_large”のサンプルアプリケーションで値の変更が可能です。プログラム内に“GOAL_CONFIG_STATIC_IP”マクロを 1 で定義することで、任意の IP アドレス設定が可能になります。
DHCP	・Management Tool を使って DHCP の有効無効の変更が可能です。 ・本サンプルの”02_eip”と”05_eip_large”サンプルでも DHCP の変更が可能で、デフォルト設定は無効です。プログラム内に“GOAL_CONFIG_ENABLE_DHCP”マクロを 1 で定義することで、DHCP が有効になります。 ・DHCP が有効、且つ、DHCP サーバがネットワーク上に無い場合は、R-IN32M3 Module 内の不揮発性メモリに保持された値を使用します。

4.4　ボード単体動作

RL78/G14 Fast Prototyping Board 上の EJ1 ヘッダ(エミュレータリセットヘッダ)を短絡させることにより エミュレータが強制リセット状態になります。エミュレータ強制リセット中は、RL78/G14 を e2studio 等の IDE から制御せずに単体での動作が可能です(ヘッダ部品は未搭載です)。
詳細は、『RL78/G14 Fast Prototyping Board ユーザーズマニュアル(R20UT4573JJ****)』を参照

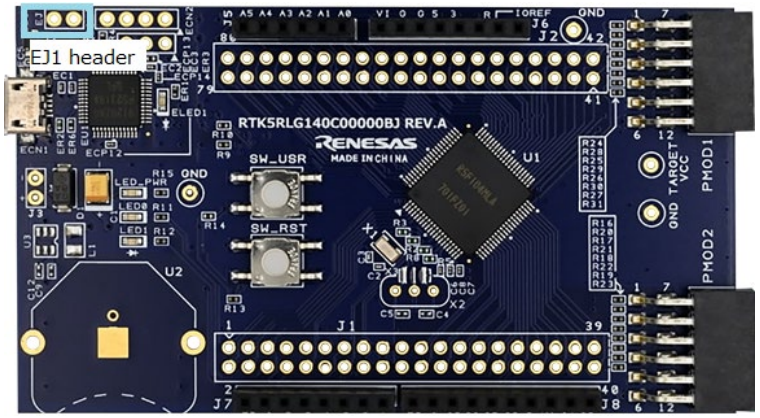

図 4-1 EJ1 ヘッダ

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2021/10/15	-	新規作成
1.01	2022/1/11	23	Remote I/O サンプルアプリケーションに関する説明を追加
		51	Modbus TCP サンプルアプリケーションに関する説明を追加
1.02	2022/8/5	-	誤記訂正などの軽微な修正
1.03	2023/5/31	23	サンプルプログラム更新に伴う説明見直し
1.04	2023/12/15	39	3.4.2 章 RPC 通信の記載追加
		29	図 3-26 差し替え、説明を追記
		5	表 1-1 更新
		3	関連文書更新
		-	誤記訂正などの軽微な修正

商標

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。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（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、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、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。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（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。
6. 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。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、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。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 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。
10. 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。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 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.4.0-1　2017.11)

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。