R-IN32M3 Module (RY9012A0)

RX66T サンプルアプリケーション (uGOAL 版)

R30AN0399JJ0105
Rev.1.05
2024.5.31

要旨

本書は、R-IN32M3 Module Evaluation Board (シマフジ電機製 / 型番：SEMB1320) 上に実装されたホストマイコン (RX66T) 用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

RX66T (R5F566TKADFP)

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 R-IN32M3 Module Evaluation Board：SEMB1320 (シマフジ電機製)
本サンプルソフト	本書で解説するホストマイコン (RX66T) 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) ユーザ実装ガイド (uGOAL編)	R30AN0402JJ****
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****
永久磁石同期モータのセンサレスベクトル制御 RX66T 実装編	R01AN4244JJ****
Renesas Solution Starter Kit 24V Motor Control Evaluation System for RX23T (Motor RSSK) 取扱説明書	R20UT3697JJ****
RX スマート・コンフィグレータ ユーザーガイド: e² studio 編	R20AN0451JS****
ソフトウェアPLC接続ガイド TwinCAT	R30AN0380JJ****

1. 概要

1.1 概要

R-IN32M3 Module Evaluation Board (型番：SEMB1320) 上に実装されたホストマイコン (RX66T) 用のサンプルソフトウェアについて説明します。

本ボードには R-IN32M3 Module と SPI 接続されたホストマイコン RX66T が実装されています。

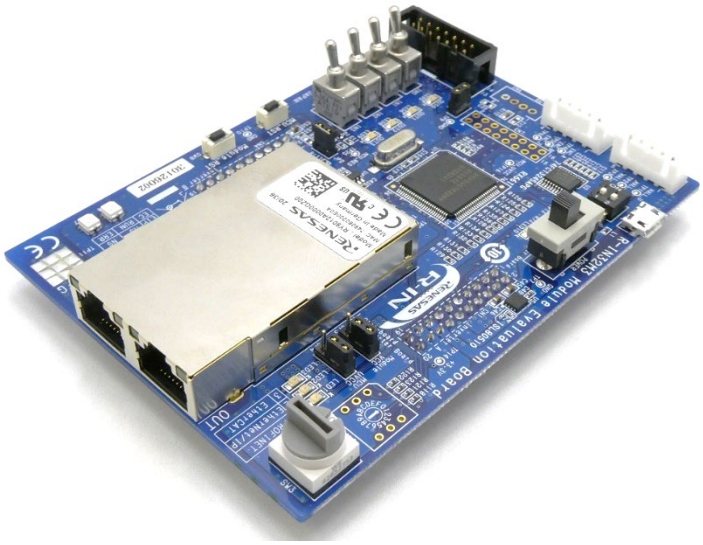

図 1-1　R-IN32M3 Module Evaluation Board

別売りのインバータボード (24V Motor Control Evaluation System for RX23T 同梱) と接続することで、産業イーサネットプロトコル通信によるモータ制御の評価を行うことが可能です。

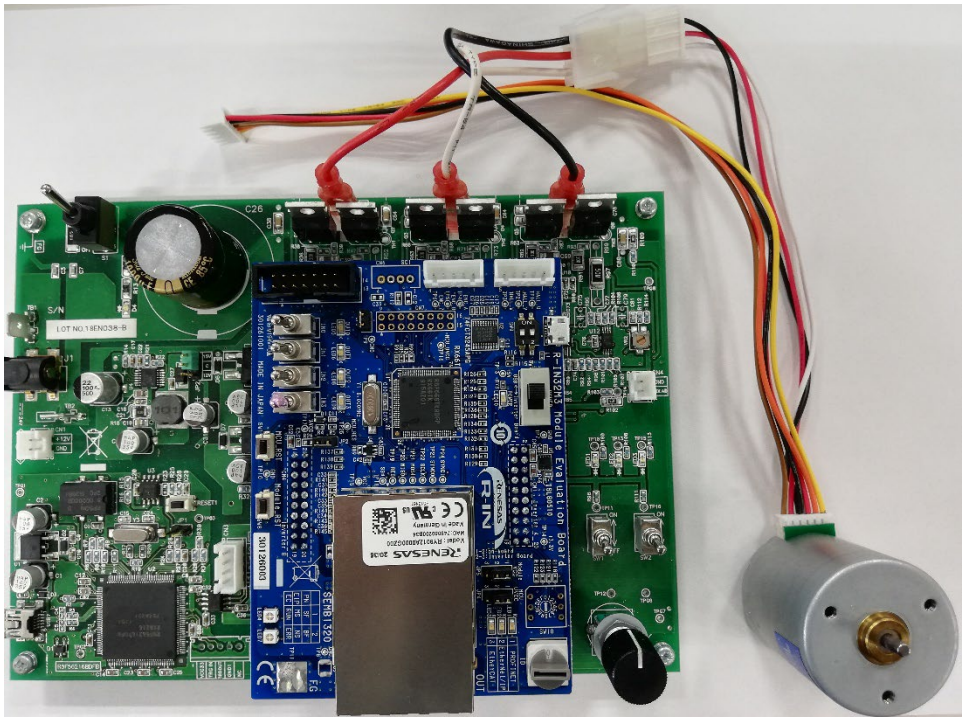

図 1-2　Inverter Board に本ボードを接続した写真

1.2　動作環境

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 Module サンプルパッケージ	サンプルパッケージ	Rev.1.05	Renesas R-IN32M3 Module Sample Package	https://www.renesas.com/
統合開発環境	e2studio	2024-04	e² studio 2024-04 Windows| Renesas	
RX ファミリ用 GNU Toolchain	GCC for Renesas RX	V8.3.0.202311	-	e2studio インストーラに同梱、必要に応じて個別にインストール (4.5 参照)
RX ファミリ用 C/C++コンパイラパッケージ	CC-RX	V3.06.00	RX ファミリ用 C/C++コンパイラパッケージ | Renesas	e2studio インストーラに同梱
FIT モジュール	RX Driver Package	V1.42	RX ファミリ RX Driver Package | Renesas	必要に応じて個別にインストール、必要に応じて個別にインストール (4.6 参照)
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

サンプルソフトの実行には下記いずれかのエミュレータが別途必要となります。

表 1-2 対応エミュレータ

略名	正式名称	詳細
E1	E1 エミュレータ	オンチップデバッギングエミュレータ兼フラッシュプログラマ 型名：R0E000010KCE00
E2 Lite	E2 エミュレータLite	オンチップデバッギングエミュレータ兼フラッシュプログラマ 型名：RTE0T0002LKCE00000R

2. ハードウェア構成

2.1 仕様一覧

ボードの仕様一覧を表 2-1 に示します。

表 2-1 ボード仕様

項目		機能・仕様
入力電源	入力電圧	5V (USB Micro B またはインバータボードから供給) または 3.3V (インバータボードから供給)
MCU	型名	R5F566TKADFP
	フラッシュメモリ容量	1MB
	RAM 容量	128KB
	データフラッシュメモリ容量	32KB
MCU 入力クロック	発振子	8MHz
コネクタ	USB Micro B	電源入力のみ データ線は未接続
	JTAG	E1 / E2 Lite エミュレータ 2.54mm ピッチ 14 ピン
	Inverter Board	CNA：2.54mm ピッチ 20 ピン CNB：2.54mm ピッチ 20 ピン
	Hall センサ	B5B-XH-A
	Encoder	B5B-XH-A
	SCI	B4B-XH-A (未実装)
スイッチ	電源入力切替	スライドスイッチ SPDT
	汎用	DIP スイッチ 2bit トグルスイッチ 4bit ロータリスイッチ 16 進 (未実装)
	EtherCAT ID	ロータリスイッチ 16 進
	CPU Reset	プッシュスイッチ 1bit
	R-IN32M3 Module Reset	プッシュスイッチ 1bit
LED	5V 電源	Red 1bit
	汎用	Green 4bit
	プロトコル表示	Green 3bit
	プロトコルステータス	Bi-Color(2bit) x 2
外部拡張用ピンヘッダ		2.54 ピッチ 16 ピン (未実装)
消費電流	モータ制御プログラム実行時	60mA typ.
	イーサネット通信プログラム実行時	260mA typ.
動作温度		0 ~ 45 ℃
基板寸法		80mm × 110mm t = 1.6mm

2.2　ボード外観

ボードの外観図を図 2-1 に、主要部品及び外部インタフェースを表 2-2 に示します。

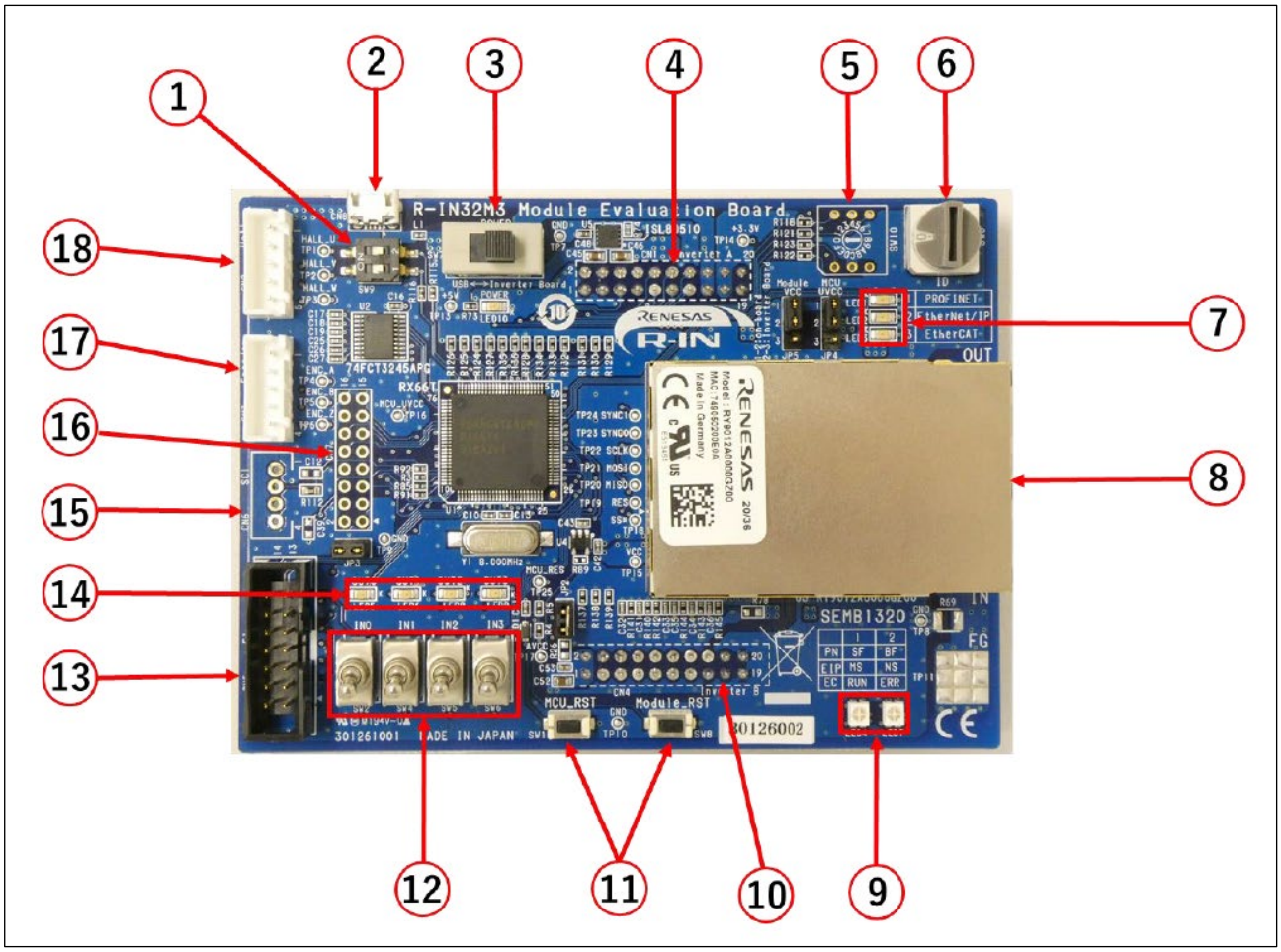

図 2-1　外観図

表 2-2　主要部品及び外部インタフェース

No.	Component Description	No.	Component Description
1	汎用 DIP スイッチ	10	Inverter Board コネクタ B
2	USB Micro B	11	Reset スイッチ
3	電源スイッチ	12	汎用入力スイッチ
4	Inverter Board コネクタ A	13	JTAG コネクタ
5	汎用 ロータリスイッチ (未実装)	14	汎用出力 LED
6	EtherCAT ID 設定スイッチ	15	SCI コネクタ (未実装)
7	プロトコル表示 LED	16	外部拡張用ピンヘッダ (未実装)
8	R-IN32M3 Module	17	Encoder コネクタ
9	プロトコルステータス LED	18	Hall センサコネクタ

2.3 ブロック図

ボードのブロック図を以下に示します。

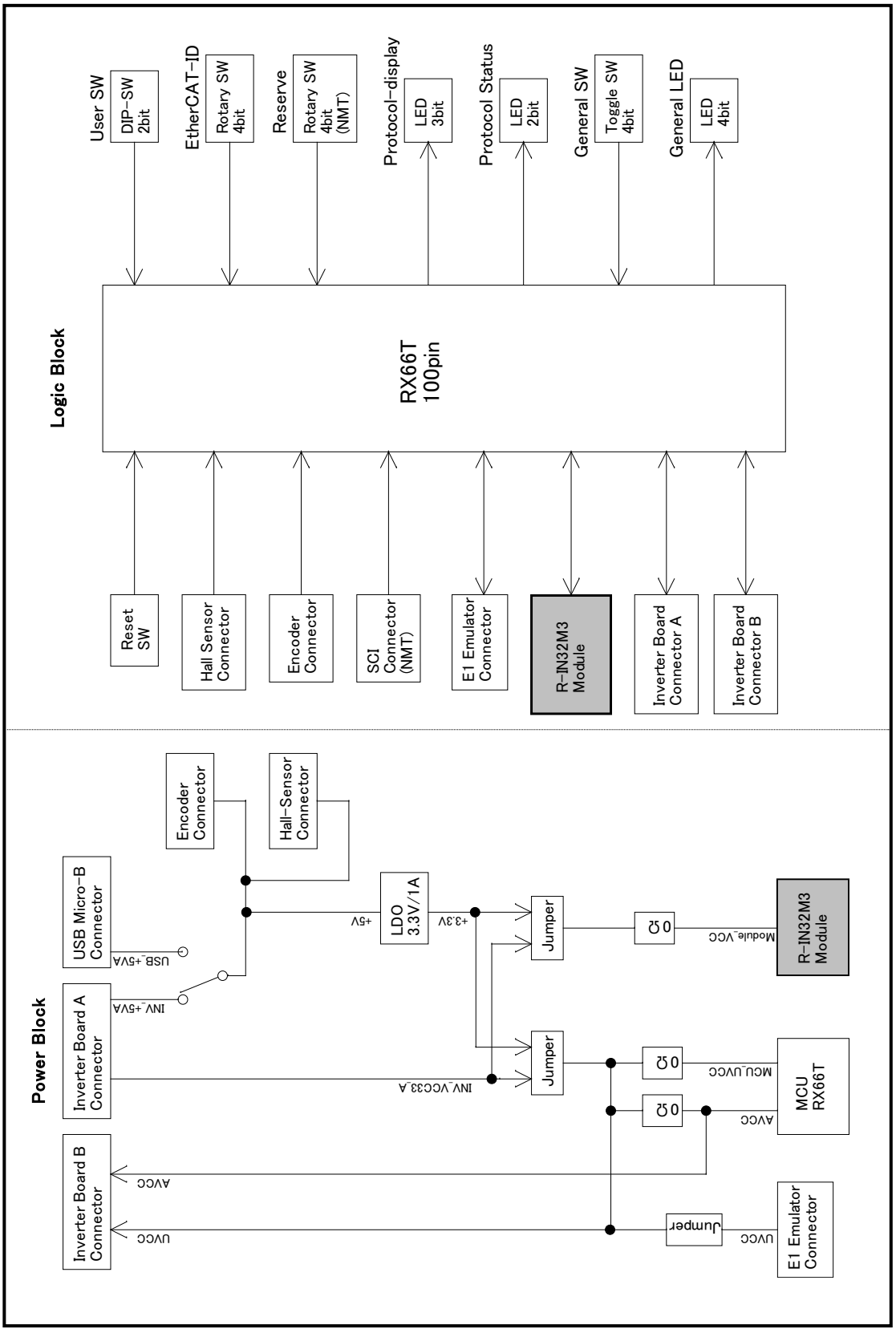

図 2-2 ブロック図

2.4　機能

2.4.1　電源

電源供給は USB Micro B コネクタ 又は Inverter Board コネクタから行います。

ボードを単体で使用する場合は、電源スイッチ(SW7) を USB Micro B コネクタ側として下さい。

MCU 及び Module への 3.3V の電源供給方法の選択はジャンパで行うことができます。詳細は、2.5.5(3) JP4、2.5.5(4) JP5 を参照してください。

電源構成図を図 2-3 に示します。

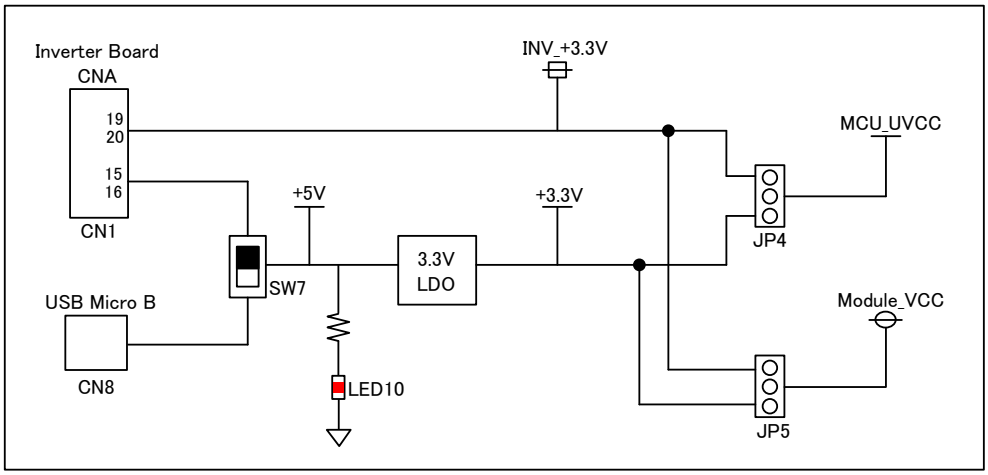

図 2-3　電源構成図

2.4.2　GND

ボードの GND と AVSS は CN4 付近で R39 によって一点接続されています。

出荷状態では、JP2 をショートしており、MCU の PGAVSS0 端子と AVSS を接続しています。

R-IN32M3 Module の FG 端子は TP11 に接続されており、R69 を実装することにより GND と接続できます。

GND の接続図を以下に示します。

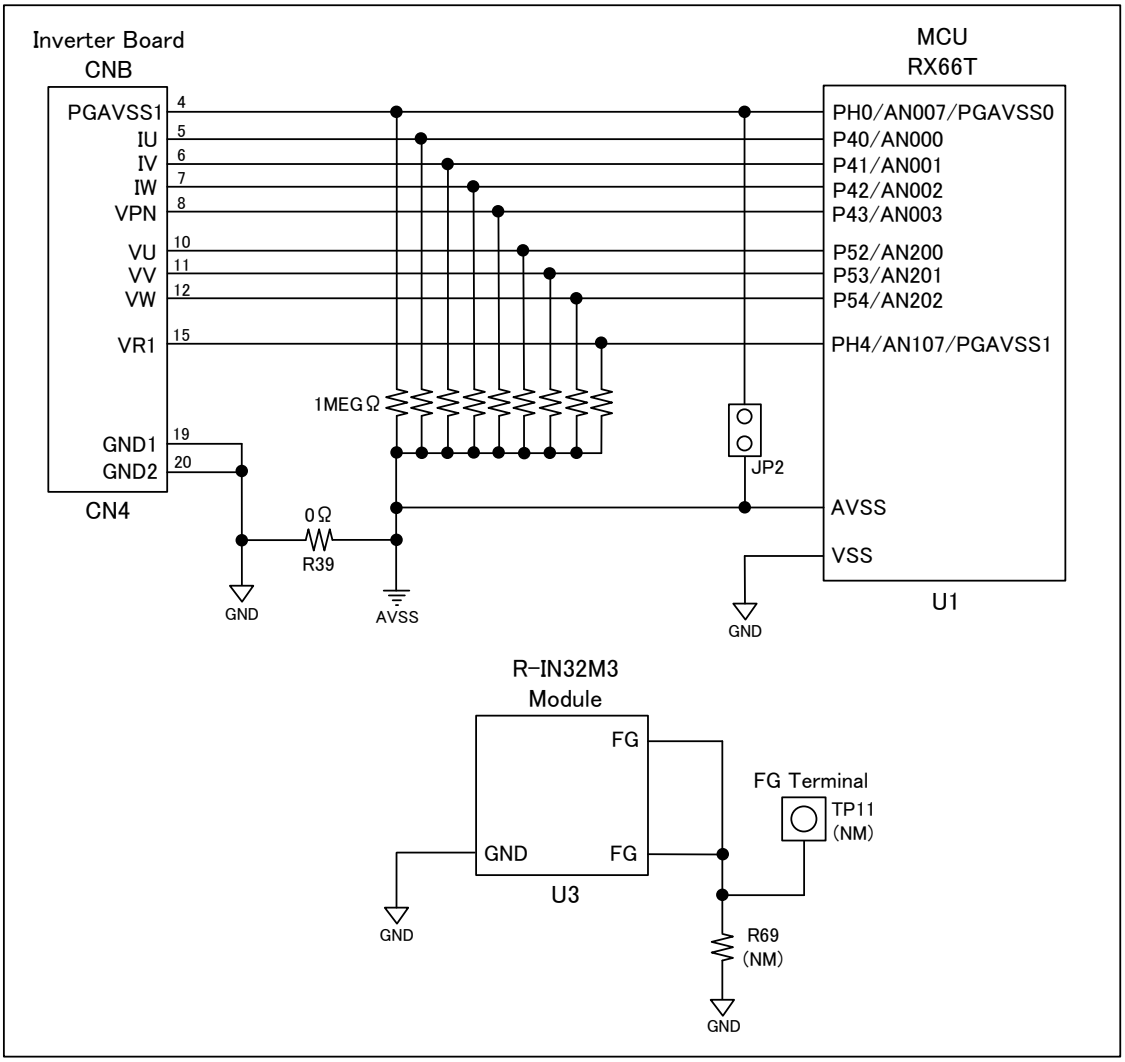

図 2-4　GND

2.4.3　RESET 及び JTAG

リセットは「Power ON Reset」、「JTAG エミュレータによるリセット」、「外部スイッチによるリセット」があります。　RESET 及び JTAG の構成図を図 2-5 に示します。

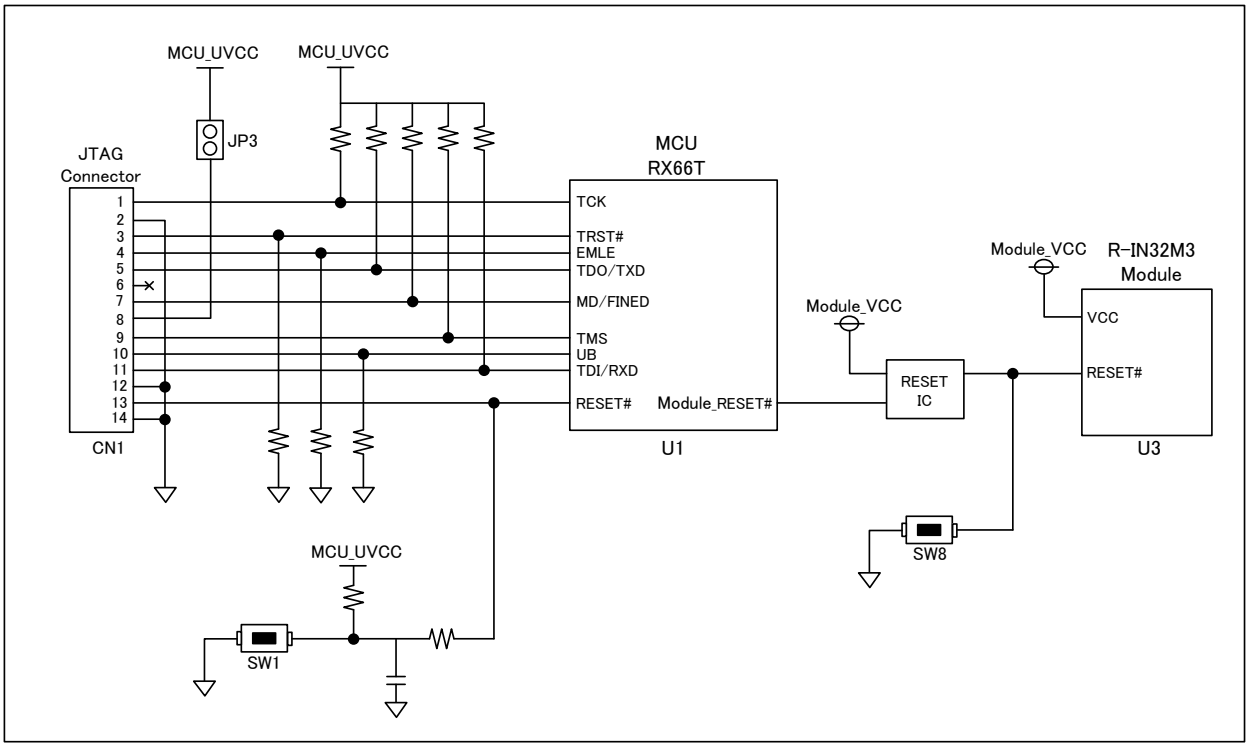

図 2-5　RESET 構成図

2.4.4　R-IN32M3 Module

R-IN32M3 Module の詳細については、『R-IN32M3 Module (RY9012A0) ユーザーズマニュアル ハードウェア編 (R19UH0122JJ****)』を参照してください。

R-IN32M3 Module と MCU の通信は 4 線式 SPI にて行います。

構成図を**図 2-6** に示します。SPI の各信号は R-IN32M3 Module 内で Pull-Up 抵抗または Pull-Down 抵抗が付与されているため、本ボード上では信号線の処理を行っていません。

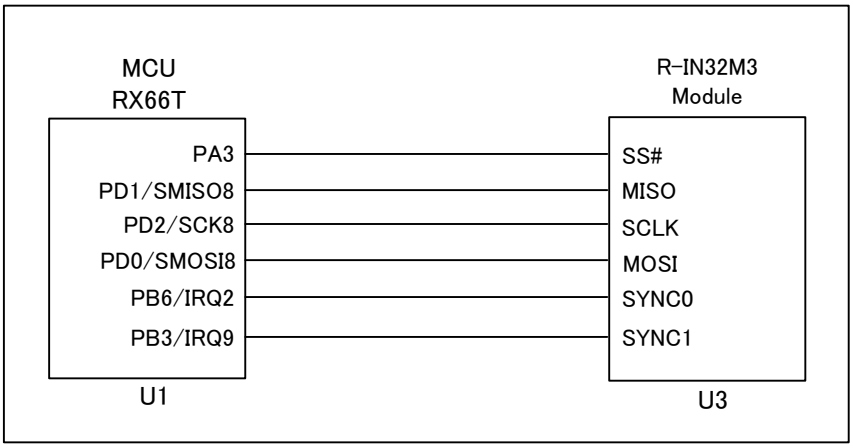

図 2-6 SPI

2.4.5　エミュレータ接続

RX66T のプログラムの書き換えは、ルネサスエレクトロニクス製オンチップデバッギングエミュレータである E1 または E2 Lite を用います。E1 または E2 Lite を本製品の Emulator コネクタと PC の USB に接続することでプログラムの書き換えが可能です。

このとき、統合開発環境における E1, E2 Lite からの電源供給は行わないでください。

2.5　外部インタフェース

2.5.1　Ethernet コネクタ

R-IN32M3 Module は RJ45 ネットワークコネクタを 2 個備えています。

R-IN32M3 の Ethernet スイッチ機能により、ディジーチェーン接続など何種類かのネットワークトポロジーで外部接続が可能です。R-IN32M3-EC の内部 PHY 層は、さまざまな産業通信プロトコルを処理することができ、10BASE-T および 100BASE-TX/FX をサポートします。

2.5.2　LED

本ボードには 5V 電源表示 LED、プロトコル表示 LED、各プロトコルのステータスを表すプロトコルステータス LED、汎用出力 LED が搭載されています。

(1)　5V 電源表示 (LED10)

USB Micro B コネクタ 又は Inverter Board コネクタからの+5V 電源供給により LED10 (Red) が点灯します。構成は、図 2-3 を参照して下さい。

(2)　プロトコル表示 (LED1~3)

使用する産業通信プロトコルに応じてサンプルソフト内のプロジェクトを RX66T で実行します。

実行する産業通信プロトコルに応じて LED1～3 (Green) が点灯します。

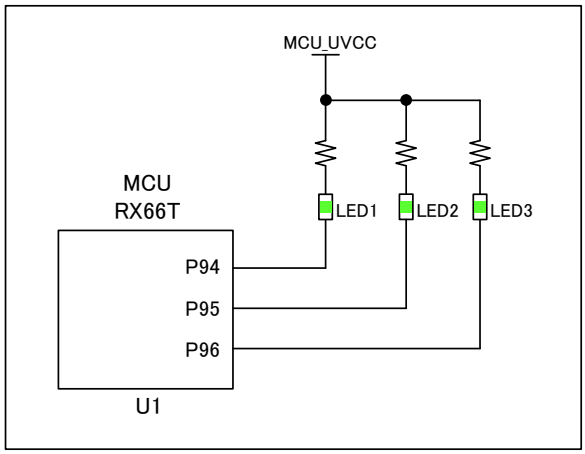

図 2-7 プロトコル表示 LED

イーサネットプロトコルの LED 表示を表 2-3 に示します。

表 2-3 プロトコル表示 LED

プロトコル	LED1 (P94)	LED2 (P95)	LED3 (P96)
PROFINET	ON	OFF	OFF
EtherNet/IP	OFF	ON	OFF
EtherCAT	OFF	OFF	ON

(3)　プロトコルステータス表示 (LED4,7)

LED4 と LED7 は Green と Red の Bi-Color LED で、各プロトコルに規定されたステータス LED を表示します。　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****)』を参照してください。

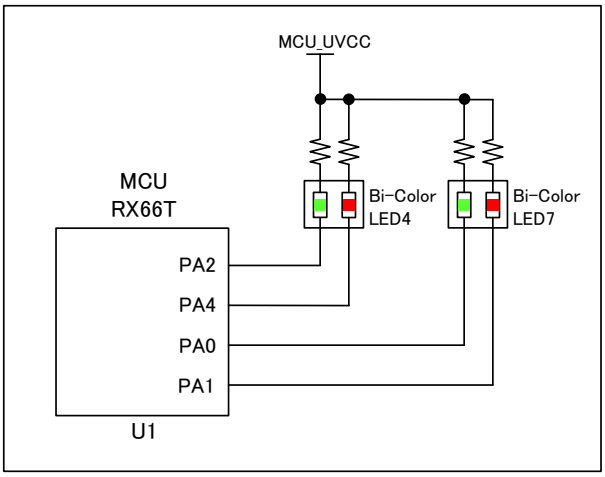

図 2-8 プロトコルステータス LED

表 2-4 Protocol Status LED

Mode		LED7		LED4	
		GREEN	RED	GREEN	RED
		PA0	PA1	PA2	PA4
1	PROFINET	Connection	BF	DCP indicator (Blink)	SF
2	EtherNet/IP	NS	NS	MS	MS
3	EtherCAT	OFF	ERR	RUN	OFF

SF: system failure, BF: bus failure, DCP: discovery and configuration protocol

MS: module status indicator, NS: network status indicator,

(4)　汎用出力 LED (LED5,6,8,9)

汎用 I/O 用途を想定した 4bit の出力 LED LED5,6,8,9 (GREEN)を用意しています。

Remote I/O 用サンプルアプリケーションでは、LED 出力として用いられます。

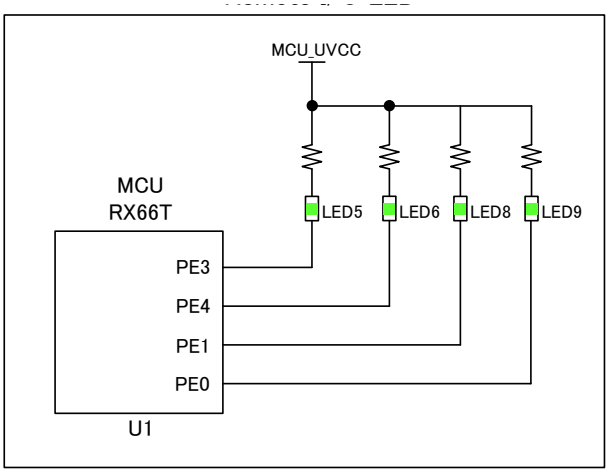

図 2-9　汎用 I/O 出力 LED 出力

2.5.3　スイッチ

本ボードには EtherCAT Explicit Device ID スイッチ、汎用 I/O 用途を想定した入力スイッチ、汎用 DIP スイッチ、電源スイッチ、リセットスイッチが搭載されています。

(1)　EtherCAT Explicit Device ID スイッチ (SW3)

プロトコル設定として EtherCAT を選択しているとき、SW3 で Explicit Device ID を設定します。

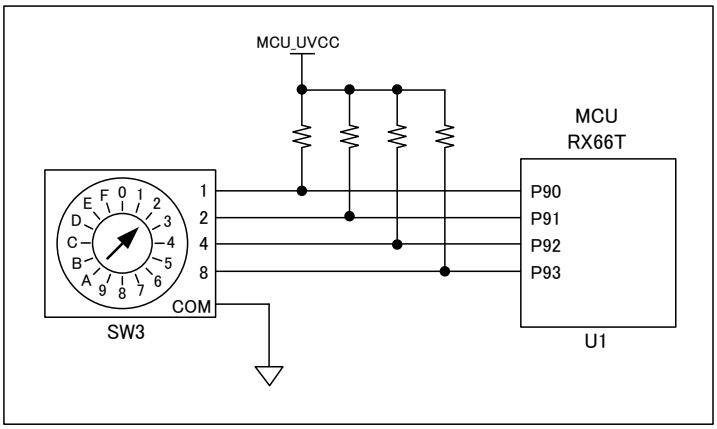

図 2-10　EtherCAT Explicit Device ID Switch

表 2-5 EtherCAT ID 設定 論理表

EtherCAT Device ID	SW3	P93	P92	P91	P90
0	0	0	0	0	0
1	1	0	0	0	1
2	2	0	0	1	0
3	3	0	0	1	1
4	4	0	1	0	0
-	-				
15	F	1	1	1	1

(2)　汎用入力スイッチ (SW2,4,5,6)

汎用 I/O 用途を想定した各 4bit の入力スイッチとして、SW2,4,5,6 (トグルスイッチ)を用意しています。

Remote I/O 用サンプルアプリケーションでは、スイッチ入力として用いられます。

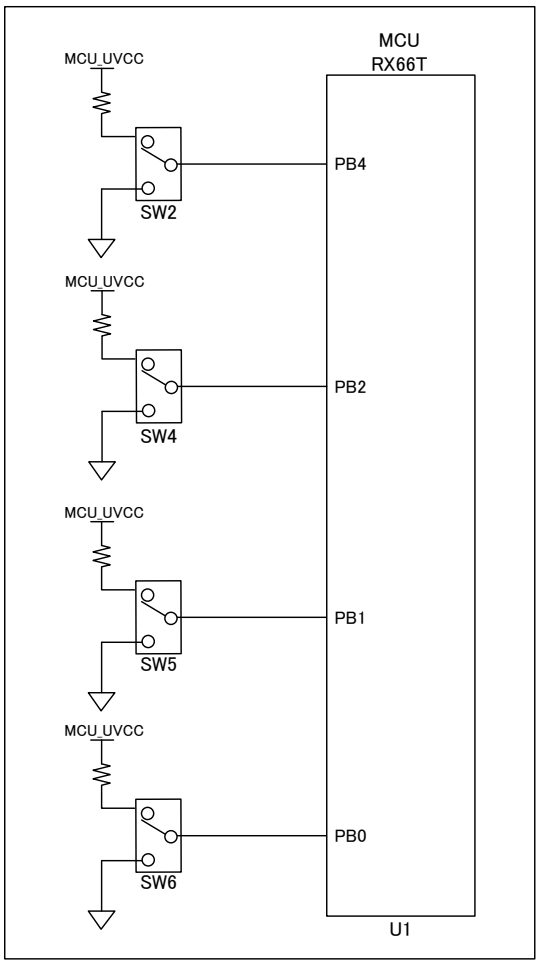

図 2-11 汎用 I/O 入力 SW

(3)　汎用スイッチ (SW9)

図 2-12 に SW9 の汎用 DIP スイッチを示します。

multi-protocol サンプルアプリケーションでは、プロトコル(PROFINET, EtherNet/IP, EtherCAT, Modbus TCP)を設定するためのセレクタ入力として用いられます。

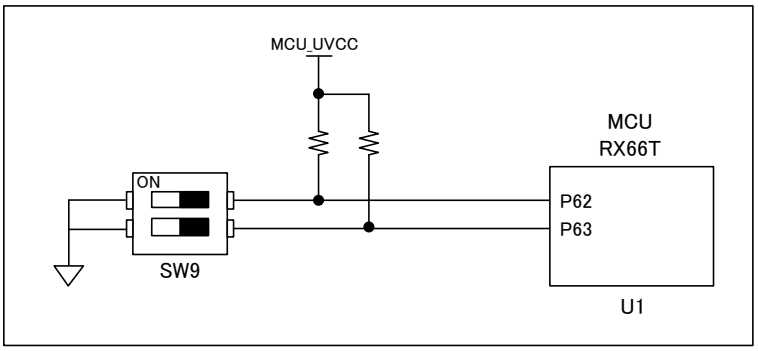

図 2-12 汎用 SW

(4)　電源スイッチ(SW7)

SW7 は本ボードへの供給する 5V 電源を選択するスイッチです。USB Micro B コネクタ 又は Inverter Board コネクタから選択して電源を供給します。

構成は、図 2-3 を参照して下さい。

表 2-6 SW7

SW7 選択	動作
1-2 (シルク記載 "USB")	USB コネクタ (CN8) から本ボードに 5V 電源が供給されます
2-3 (シルク記載 "Inverter Board")	Inverter Board コネクタ A (CN1) から 5V 電源が供給されます。

(5)　リセットスイッチ (SW1, SW8)

SW1 は RX66T マイコン、SW8 は R-IN32M3 Module をリセットするプッシュスイッチです。

リセット構成については図 2-5 を参照してください。

2.5.4　コネクタ

搭載しているコネクタを表 2-7 に示します (R-IN32M3 Module 搭載の RJ-45 コネクタを除く)。

表 2-7 コネクター覧

CN 番号	コネクタ型番	備考
CN1	SFH11-PBPC-D10-ST-BK	Inverter Board コネクタ A
CN2	B5B-XH-A	Hall センサコネクタ
CN3	B5B-XH-A	Encoder コネクタ
CN4	SFH11-PBPC-D10-ST-BK	Inverter Board コネクタ B
CN5	XG4C-1431	JTAG コネクタ
CN6	B4B-XH-A(未実装)	SCI コネクタ(未実装)
CN7	A1-16PA-2.54DSA(未実装)	外部拡張用ピンヘッダ(未実装)
CN8	10118194-0001LF	USB Micro B

(1)　Inverter Board コネクタ (CN1, CN4)

24V Motor Control Evaluation System の CPU カードとして使用するための Inverter Board コネクタが搭載されています。以下に、Inverter Board コネクタのピンアサインを示します。

表 2-8 Inverter Board コネクタ A (CN1)

ピン番号	信号名	MCU ポート	備考	ピン番号	信号名	MCU ポート	備考
1	LED1#	P22		2	LED2#	P21	
3	LED3#	P20		4	VRL	P24	
5	FO#	P70		6	NC	-	NC
7	WN	P76/ MTIOC4D		8	VN	P75/ MTIOC4C	
9	UN	P74/ MTIOC3D		10	WP	P73/ MTIOC4B	
11	VP	P72/ MTIOC4A		12	UP	P71/ MTIOC3B	
13	SW1#	P23		14	SW2#	P27	
15	+5VA1	-		16	+5VA2	-	
17	GND	-		18	GND	-	
19	VCC33_A1	-		20	VCC33_A2	-	

表 2-9　Inverter Board コネクタ B (CN4)

ピン番号	信号名	MCU ポート	備考	ピン番号	信号名	MCU ポート	備考
1	AVCC1	-		2	AVCC2	-	
3	NC	-	NC	4	PGAVSS1	PH0/ PGAVSS1	
5	IU	P40/AN000		6	IV	P41/AN001	
7	IW	P42/AN002		8	VPN	P43/AN003	
9	TEMP(VOT)	-	NC	10	VU	P52 /AN200	
11	VV	P53/AN201		12	VW	P54/ AN202	
13	VAC	-	NC	14	IPFC	-	NC
15	VR1	PH4/AN107		16	VN	-	NC
17	VCCIO1	-		18	VCCIO2	-	
19	GND1	-		20	GND2	-	

(2)　Hall センサコネクタ (CN2)

Hall センサコネクタは 5V 動作の Hall センサにのみ対応しています。

VCC ピンに 5V が接続されているため、3.3V 動作の Hall センサを接続すると故障する恐れがあります。

信号線はボード上のレベル変換 IC によって 5V の入力信号を 3.3V へ変換して MCU へ接続しています。

Hall センサコネクタのピンアサインを以下に示します。

表 2-10 Hall センサコネクタ

ピン番号	信号名	MCU ポート	備考
1	VCC	-	5V 出力
2	GND	-	
3	HALL_U	P61/ IRQ5	本ボード上で 3.3V に変換
4	HALL_V	P60/IRQ4	本ボード上で 3.3V に変換
5	HALL_W	P55/IRQ3	本ボード上で 3.3V に変換

(3) Encoder コネクタ (CN3)

Encoder コネクタは 5V 動作の Encoder にのみ対応しています。

VCC ピンに 5V が接続されているため、3.3V 動作の Hall センサを接続すると故障する恐れがあります。

信号線はボード上のレベル変換 IC によって 5V の入力信号を 3.3V へ変換して MCU へ接続しています。

Encoder コネクタのピンアサインを以下に示します。

表 2-11 Encoder コネクタ (CN3)

ピン番号	信号名	MCU ポート	備考
1	VCC	-	5V 出力
2	GND	-	
3	ENC_A	P33/MTCLKA	本ボード上で 3.3V に変換
4	ENC_B	P32/MTCLKB	本ボード上で 3.3V に変換
5	ENC_Z	PA5/MTIOC1A	本ボード上で 3.3V に変換

(4) JTAG コネクタ (CN5)

JTAG コネクタのピンアサインを以下に示します。

表 2-12 JTAG コネクタ (CN5)

ピン番号	信号名	MCU ポート	備考	ピン番号	信号名	MCU ポート	備考
1	TCK/FINEC	PD4/TCK		2	GND	-	
3	TRST#	PD7/TRST#		4	EMLE	EMLE	
5	TDO /TXD1	PD3/TXD1		6	NC	-	NC
7	MD/FINED	MD/FINED		8	VCC	-	
9	TMS	PD6_TMS		10	UB	P00_UB	
11	TDI/RXD1	PD5_RXD1		12	GND	-	
13	RESET#	RESET#		14	GND	-	

(5) SCI コネクタ (CN6：未実装)

SCI コネクタのピンアサインを以下に示します。CN6 は未実装です。

表 2-13 SCI コネクタ (CN6)

ピン番号	信号名	MCU ポート
1	VCC	-
2	TXD	P81/TXD6
3	RXD	P80/RXD6
4	GND	-

(6) 外部拡張用コネクタ (CN7：未実装)

外部拡張用コネクタ (CN7) には、MCU の未使用ピンが接続されています。

外部拡張用コネクタのピンアサインを以下に示します。CN7 のコネクタ (ピンヘッダ) は未実装です。

表 2-14 外部拡張用コネクタ (CN7)

ピン番号	信号名	MCU ポート	備考	ピン番号	信号名	MCU ポート	備考
1	MCU_UVCC	-		2	MCU_UVCC	-	
3	MCU_PE5	PE5		4	MCU_P01	P01	
5	MCU_PE2	PE2		6	MCU_P10	P10	
7	MCU_P11	P11		8	MCU_P82	P82	
9	MCU_P44	P44		10	MCU_P45	P45	
11	MCU_P46	P46		12	MCU_P47	P47	
13	MCU_PB7	PB7		14	GND	-	
15	GND	-		16	GND	-	

(7) USB micro B (CN8)

ボードへの 5V 電源供給用 USB Micro B コネクタです。電源構成は図 2-3 を参照してください。

SW7 を "USB" とシルク記載している方に切り替えることで、CN8 から電源が供給されます。

2.5.5　ジャンパピン

ジャンパピンの一覧を以下に示します。通常は、出荷状態のままご使用下さい。

表 2-15 ジャンパピン一覧

JP 番号	型番	備考
JP2	XJ8C-0211	PGAVSS0 と AVSS の接続選択
JP3	XJ8C-0211	JTAG_VCC と MCU_UVCC の接続選択
JP4	XJ8D-0311	MCU_UVCC の入力先選択
JP5	XJ8D-0311	Module_VCC の入力先選択

各ジャンパピンの設定を以下に示します。

(1)　JP2

JP2 は MCU の PGAVSS0 ピンと AVSS を接続するためのジャンパピンです。

JP2 の設定表を以下に示します。

インバータボードを使用する際は、本ボードの JP2 をショート (出荷状態) したまま、ご使用下さい。

表 2-16　JP2 設定

JP 選択	説明	出荷時
1-2	PGAVSS0 ピンと AVSS を接続します。 ※ 通常はこちらを設定してください。	○
未選択	PGAVSS0 ピンと AVSS を接続しません。	

(2)　JP3

JP3 は JTAG コネクタ (CN5) の VCC ピンを MCU_UVCC と接続するためのジャンパピンです。JP3 の設定表を以下に示します。

表 2-17　JP3 設定

JP 選択	説明	出荷時
1-2	JTAG コネクタの VCC ピンと MCU_UVCC を接続します。 ※ 通常はこちらを設定してください。	○
未選択	JTAG コネクタの VCC ピンと MCU_UVCC を接続しません。	

(3) JP4

JP4 は MCU_UVCC の入力先を設定するためのジャンパピンです。JP4 の設定表を以下に示します。

表 2-18 JP4 設定

JP 選択	説明	出荷時
1-2	3.3V LDO 出力を MCU_UVCC として使用します。 ※ 通常はこちらを設定してください。	○
2-3	Inverter Board コネクタ A の VCC33_A を MCU_UVCC として使用します。	
未選択	MCU に電源が供給されません。 ※ JTAG の電源供給機能を使用する場合に設定してください。	

(4) JP5

JP5 は Module_VCC の入力先を設定するためのジャンパピンです。JP4 の設定表を以下に示します。

表 2-19 JP5 設定

JP 選択	説明	出荷時
1-2	3.3V LDO 出力を Module_VCC として使用します。 ※ 通常はこちらを設定してください。	○
2-3	Inverter Board コネクタ A の VCC33_A を Module_VCC として使用します。	
未選択	R-IN32M3 Module に電源が供給されません。	

2.6　RX66T CPU カードとの差異

本ボードは RX66T CPU カード (RTK0EMX870C00000BJ) をベースに、産業イーサネット通信モジュールを追加する構成となっておりますが、一部 MCU および周辺回路に差異があります。このため、RX66T CPU カード用のサンプルソフトを動作させるためには、以下のハードウェアの差異を考慮した変更が必要となります。

RX66T CPU カードと本ボードについて、R-IN32M3 Module 関連回路、および CPU カード上の MPU 周辺回路の主な差異を表 2-20 に示します。

表 2-20　RX66T CPU カードと本ボードの主な差異

#	項目		本ボード	RX66T CPU カード RTK0EMX870C00000BJ	接続	備考
1	搭載 MPU	型名	R5F566TKADFP	R5F566TEADFP		
2		ROM	1024KByte	512KByte		
3		RAM	128KByte	64KByte		
4		Package	LFQFP / 100 pin	同左		
5		動作電源電圧	3.3V (MPU 仕様: 2.7~5.5V)	5V (MPU 仕様: 2.7~5.5V)		
6	エンコーダ入力	ENCA	58pin (P33 / MTCLKA)	同左	CN3	
7		ENCB	59pin (P32 / MTCLKB)	同左	CN3	
8		ENCZ	36pin (PA5 / MTIOC1A)	同左	CN3	
9	ホール素子入力	HU	76pin (P61 / IRQ5)	17pin (PE0 / IRQ7)	CN2	
10		HV	77pin (P60 / IRQ4)	16pin (PE1 / IRQ15)	CN2	
11		HW	78pin (P55 / IRQ3)	1pin (PE5 / IEQ0)	CN2	
12	ボリューム入力	VR_1	86pin (PH4 / AN107)	68pin (P21 / AN217)	CN4	電圧変換必要[注 1]
13	DC リンク電圧検出	VPN	87pin (P43 / AN003)	75pin (P62 / AN208)	CN4	
14	波形モニタツール通信	TXD6	97pin (P81 / TXD6)	35pin (PB0 / TXD6)	CN6	使用不可
15		RXD6	98pin (P80 / RXD6)	34pin (PB1 / RXD6)	CN6	使用不可
16	インバータボードインタフェース	SW2#	64pin (P27)	97pin (P81)	CN1	エラー解除
17		SW1#	66pin (P23)	98pin (P80)	CN1	モータ回転許可
18		LED1#	67pin (P22)	9pin (PE3)	CN1	通常動作
19		LED2#	68pin (P21)	26pin (PB7)	CN1	エラー発生
20		LED3#	69pin (P20)	32pin (PB3)	CN1	

注1.　入力電源電圧を 5V から 3.3V に変更したことにより、ソフトウェア電圧読み込み処理を補正する必要があります。尚、本サンプルソフトウェアは対応済みです。

3. サンプルソフト

R-IN32M3 Module サンプルソフトは **1.2 動作環境** より入手できます。

サンプルソフトは リトルエンディアンとなっています。 ビックエンディアン対応は **4.4 Big-endian 対応** を参照しプロジェクト作成してください。

3.1 フォルダ構成

サンプルソフトのフォルダ構成を以下に示します。

```
RX66T_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
│   ├──11_pnio_http        01_pnio サンプルに web サーバ機能，ホストマイコン FW 更新機能 拡張
│   ├──12_eip_http         02_eip に web サーバ機能，ホストマイコン FW 更新機能 拡張
│   ├──13_ecat_http        03_ecat に web サーバ機能，ホストマイコン FW 更新機能 拡張
│   └──17_fwup_bootloader  ホストマイコン FW 更新機能用 bootloader
│
├──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

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。

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 Server	3.4.4 Modbus TCP
10_multi_protocol	PROFINET / EtherNet/IP / EtherCAT / ModbusTCP Server	3.4.5 multi-protocol
11_pnio_http	PROFINET	3.4.6 Web サーバ機能
12_eip_http	EtherNet/IP	
13_ecat_http	EtherCAT	

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。

本書では、データ通信機能に関してのみ解説しています。ファームウェア更新機能に関する詳細は『R-IN32M3 Module (RY9012A0) ファームウェア更新ガイド(R30AN0401JJ****)』を参照ください。

3.3　開発環境構築

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

e2studio をダウンロードし、インストールしてください。

Download Link: 1.2 動作環境

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、”RX”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)。

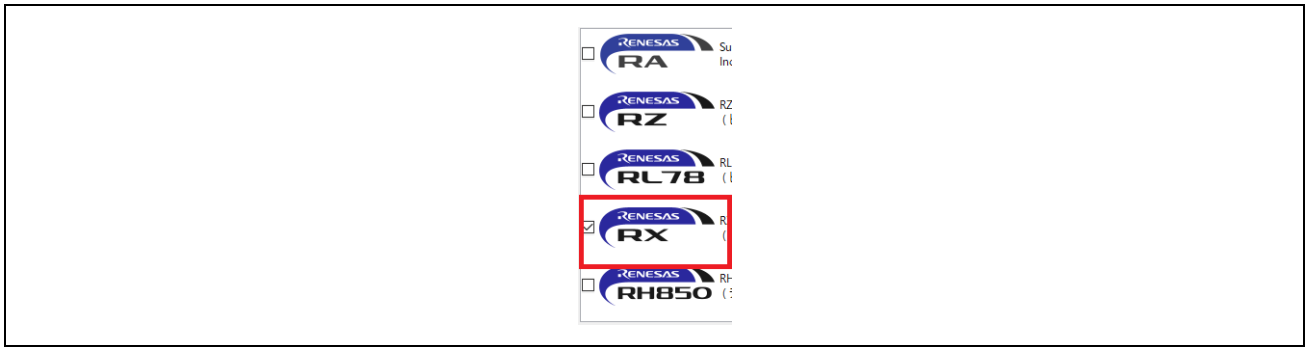

図 3-1 デバイス・ファミリー選択

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” カテゴリで対象のコンパイラを選択します。

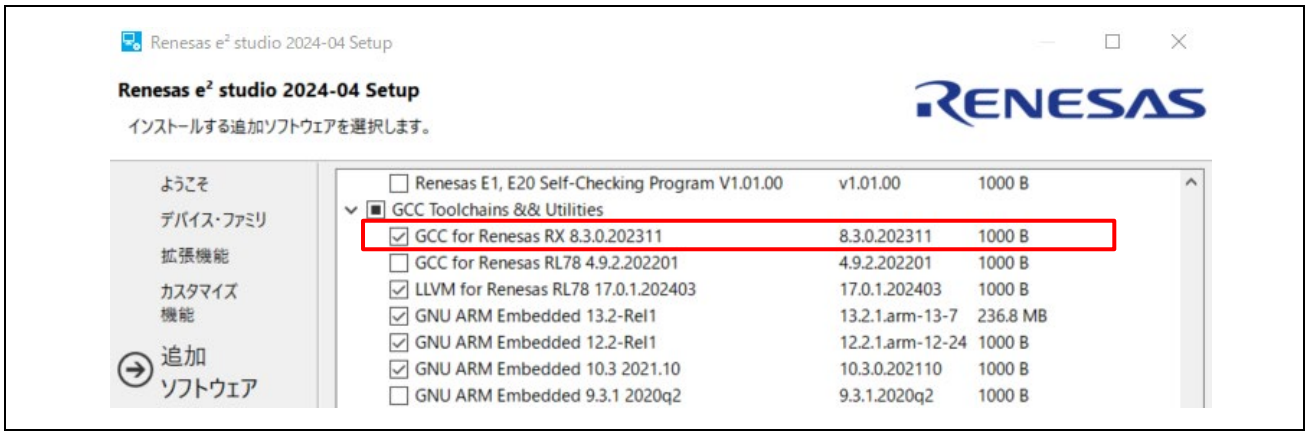

図 3-2 追加ソフトウェア選択

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。

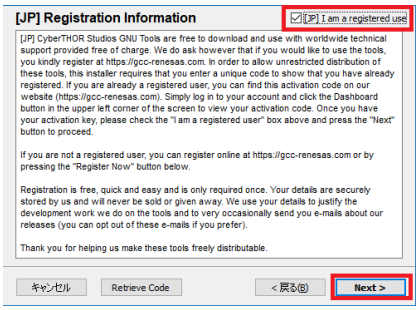

図 3-4 GCC コンパイラのインストール(アカウント所持)

(2)　CC-RX

CC-RX を利用の場合は、CC-RX のインストールが必要です。

e2studio インストール時に表 1-1 に記載のバージョンを選択してインストールしてください。該当するバージョンが含まれない場合は、4.7 章を参照して個別インストールをしてください。

e2studio インストール:

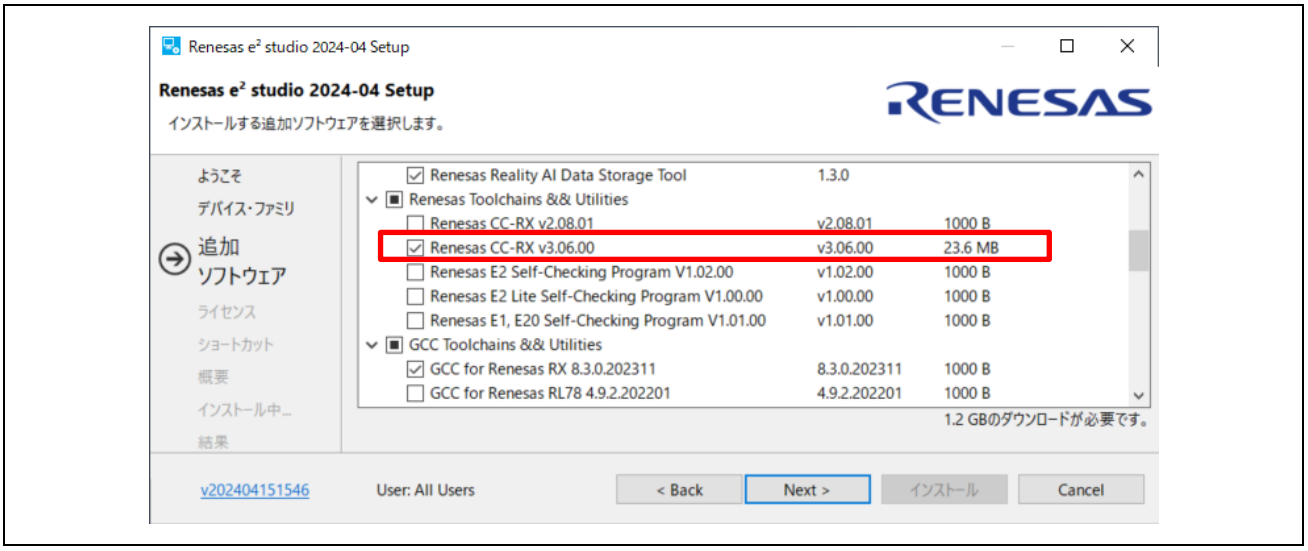

図 3-5 CC-RX インストール

なお、無償評価版であるために、ライセンスには有効期限があり 60 日試用版となります。60 日以降は ROM サイズが 128Kbyte に制限されます。詳細は、「CC-RX コンパイラ ユーザーズマニュアル」をご参照ください。

3.3.2　開発環境接続

SEMB1320 と E1 エミュレータ及び、PC を以下のように接続します。

電源入力スイッチ SW7 を”USB”側に設定した後、USB micro B ケーブルを接続する事によって、本ボードに電源が供給されます。

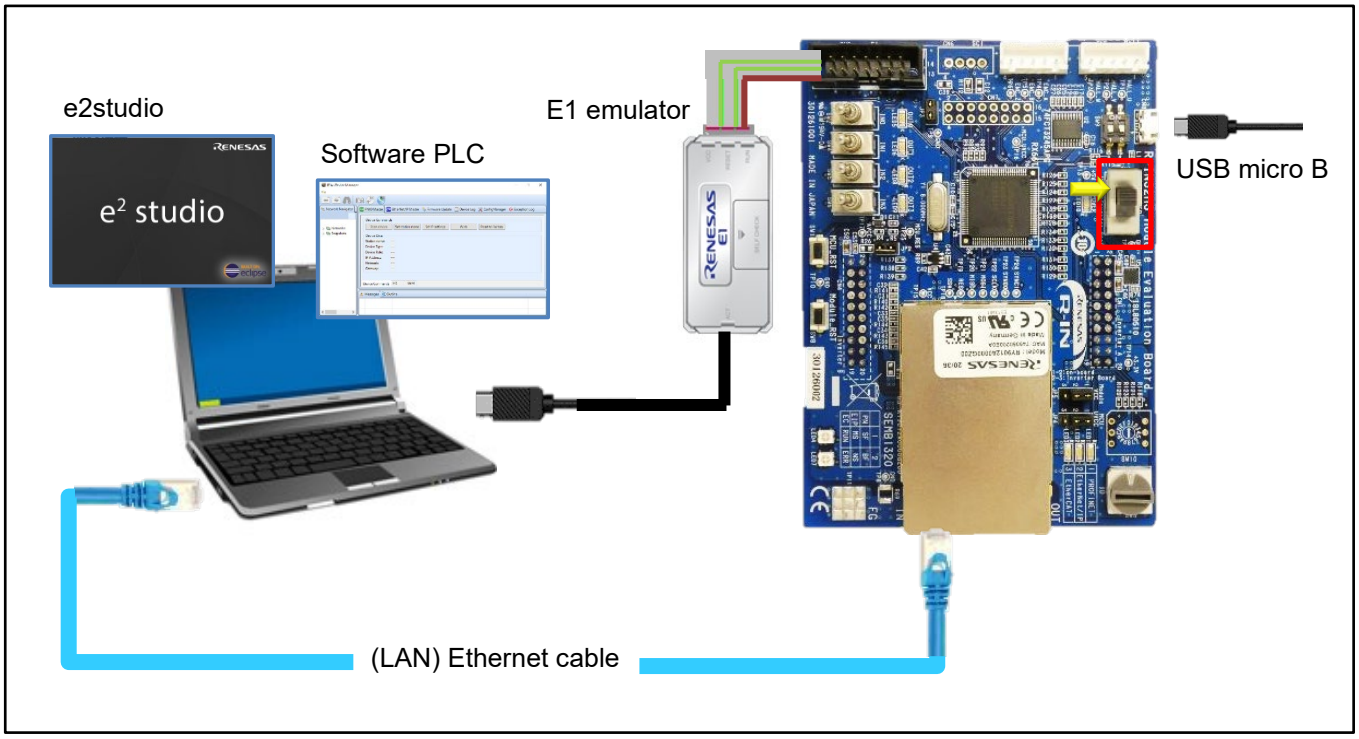

図 3-6　接続構成

3.3.3　プロジェクト立上げ

(1)　zip ファイル解凍

アーカイブされた本サンプルソフトのパッケージ (RX66T_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下記フォルダ(デフォルトの場合)の”e2studio.exe”を実行してください。

¥Renesas¥e2_studio¥eclipse¥e2studio.exe

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、使用するビルド環境 ”Renesas CCRX”もしくは ”GCC for Renesas RX”に該当コンパイラが追加されているかを確認できます。

本ガイドでは GCC 8.3.0.202311 の使用を前提にしています。ダイアログ上に GCC 8.3.0.202311 が見つからない場合は、個別にインストールが必要です。詳細は 4.5 GCC ツールチェーンの個別インストール をご覧ください。

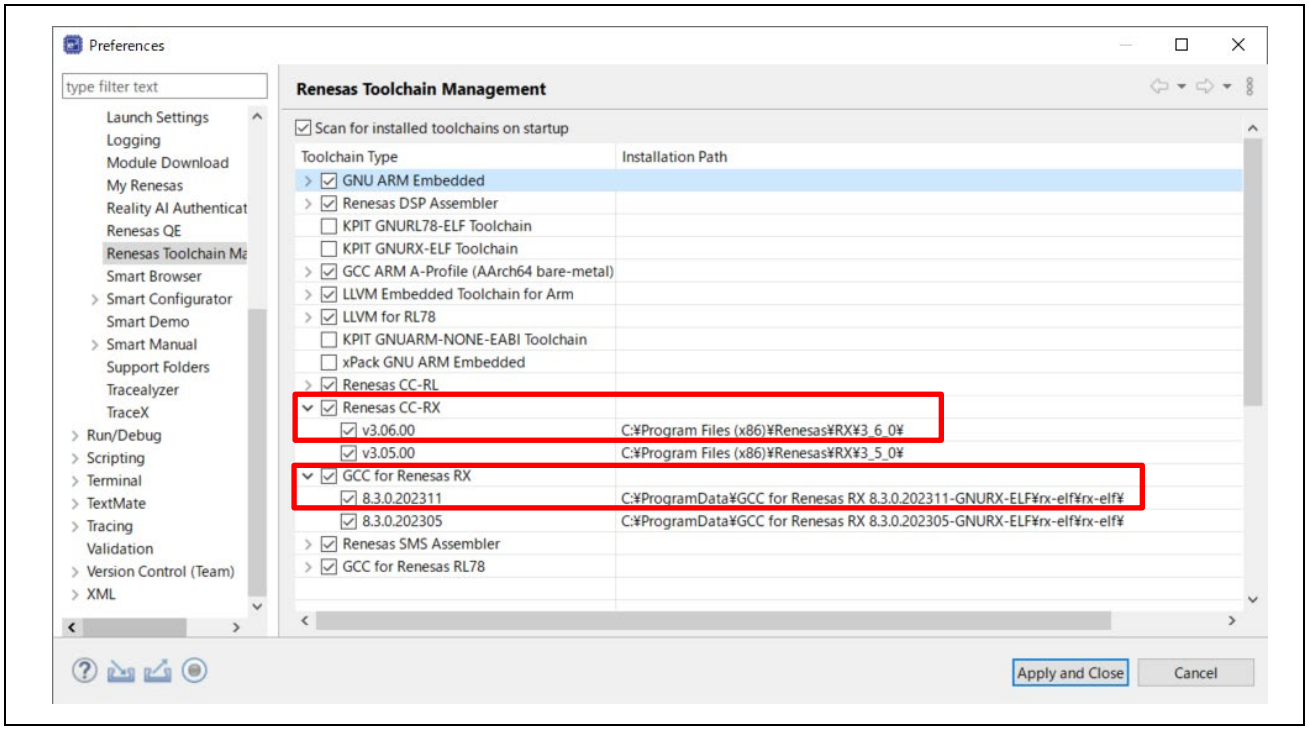

図 3-7 Renesas ツールチェーン管理

(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。

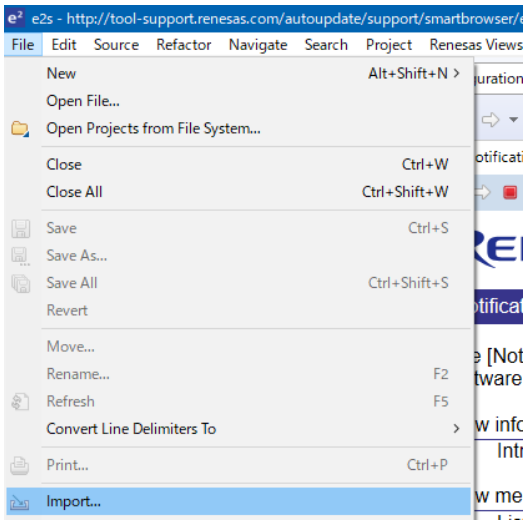

図 3-8 インポート

[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。

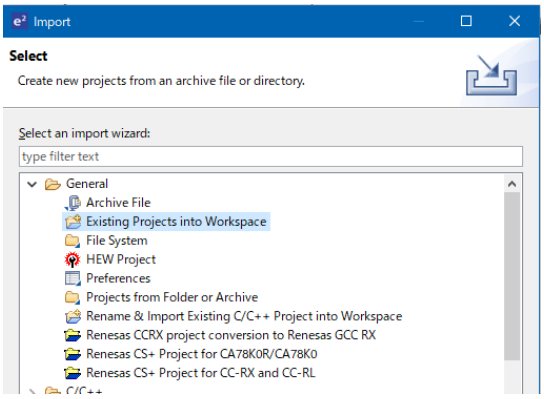

図 3-9 既存プロジェクトをワークスペースへ を選択

[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 (RX66T_uCCM_V***) を選択して[OK]を選択します。

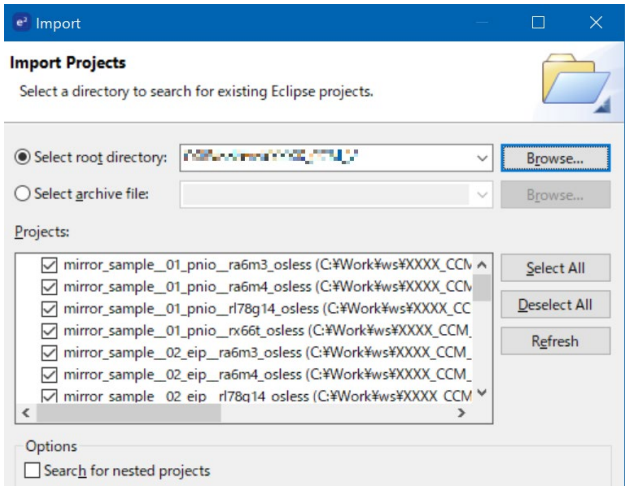

図 3-10 プロジェクトのインポート

[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。

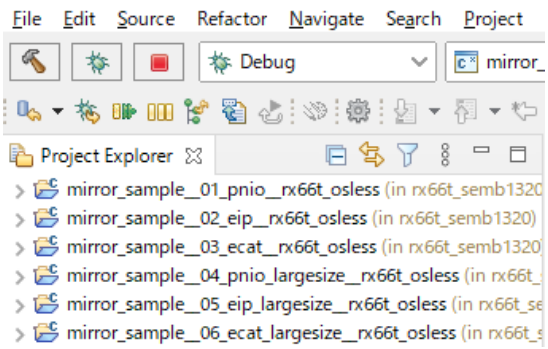

図 3-11 インポート完了

3.3.4　FIT モジュール

(1)　FIT モジュールのダウンロード

本サンプルソフト内の各サンプルプロジェクトは、FIT モジュールを使用していますので、FIT モジュールを e2studio のスマート・コンフィグレータを使ってダウンロードします。ダウンロードに関しては、同一バージョンであれば、1 度実施すれば、以降プロジェクト毎に実施する必要はありません。すでにダウンロード済みの場合は次項へ進んでください。詳細については『RX スマート・コンフィグレータ ユーザーガイド: e2studio 編 (R20AN0451JS****)』を参照ください。

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展開して、コンフィグレーションファイルを選択します。

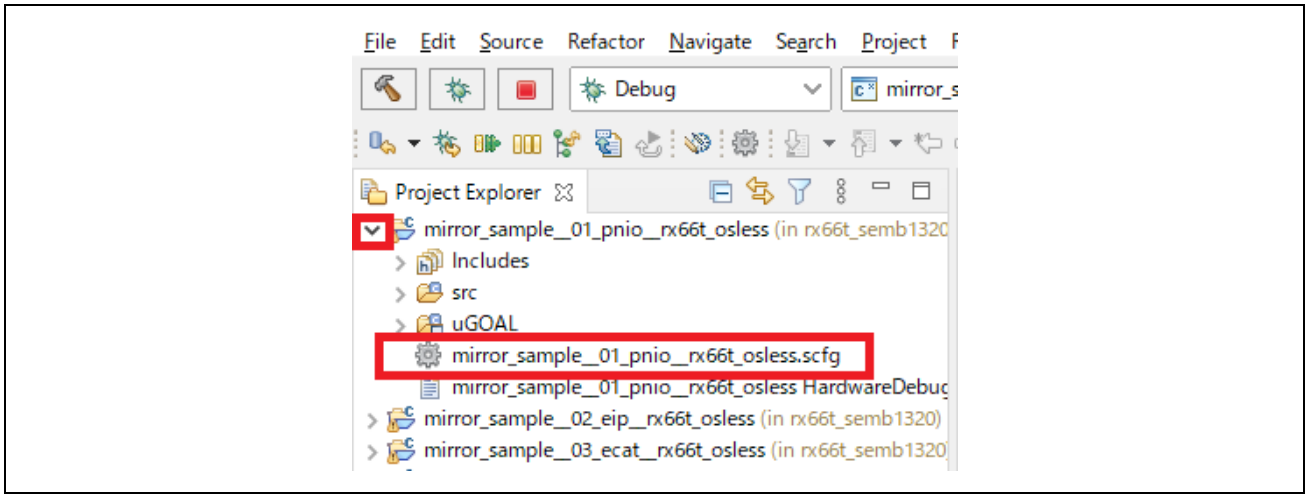

図 3-12 コンフィグレーションファイルを選択

下記ダイアログが表示された場合は、[パースペクティブを開く]を選択します。

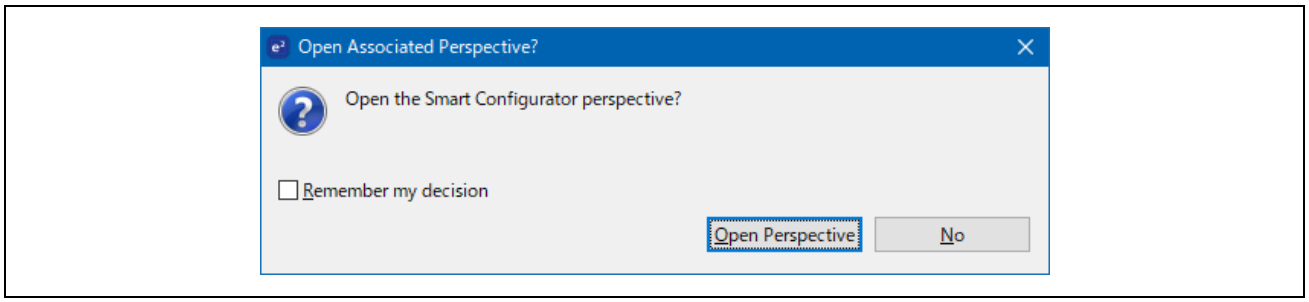

図 3-13 パースペクティブを開く

スマート・コンフィグレータ・パースペクティブ画面の[コンポーネント]タブに移動し、[コンポーネントの追加]ボタンを選択します。

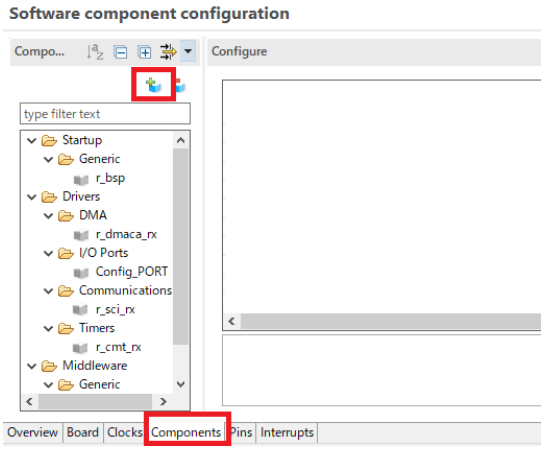

図 3-14 コンポーネントの追加

[ソフトウェアコンポーネントの選択]ダイアログの[他のソフトウェアコンポーネントをダウンロードする]を選択します。地域選択ダイアログが表示された場合は、自身の地域を選択します。

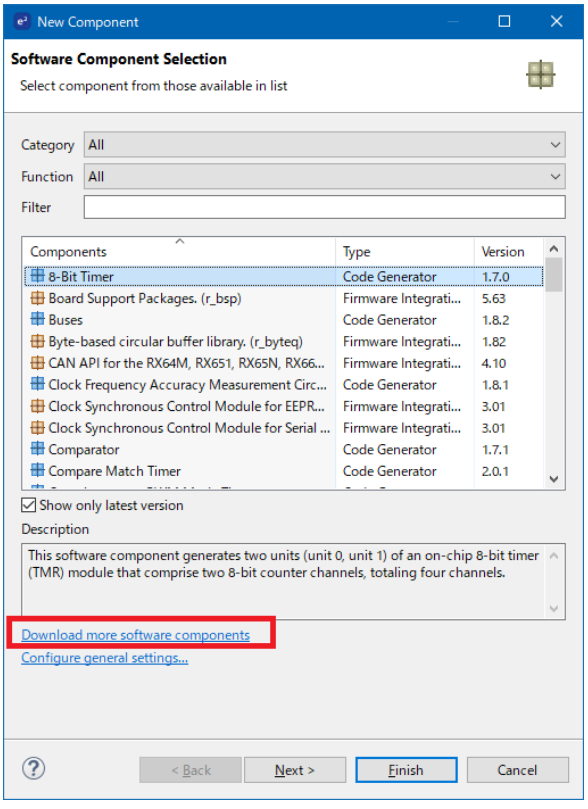

図 3-15 他のソフトウェアコンポーネントのダウンロード選択

RX Driver Package にチェックを付けた後、[ダウンロード]を選択します。ライセンスのダイアログが表示された場合は、[Accept]ボタンを選択します。本ガイドでは RX Driver Package Ver.1.42 の適用を前提としています。Ver.1.42 が選択できない場合には別途パッケージのダウンロードと適用作業が必要になります。詳細は 4.6 FIT モジュールの個別インストールと適用方法 を参照のこと。

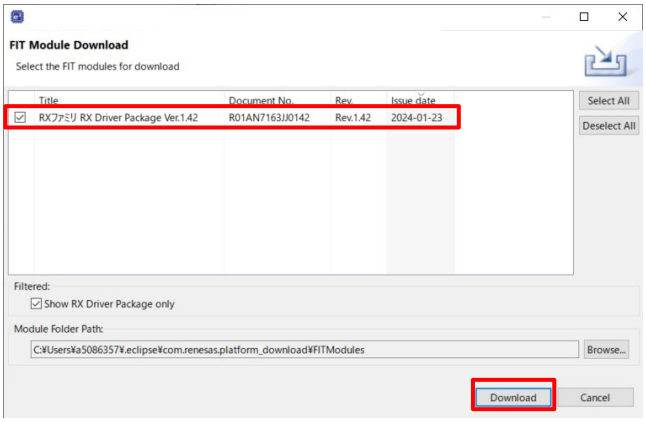

図 3-16 ダウンロード

ダウンロード完了後、[ソフトウェアコンポーネントの選択]ダイアログを閉じます。正常にダウンロードされると、FIT モジュールの各要素が活性化状態になります。

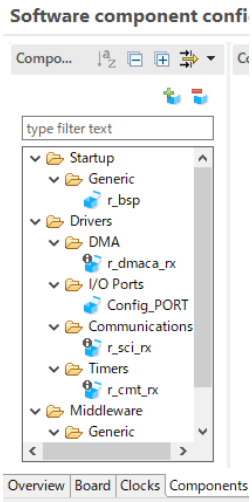

図 3-17 FIT モジュール

各コンポーネントが活性化しない場合、又は、各コンポーネントを個別に最新バージョンへ更新したい場合は、スマート・コンフィグレータ・パースペクティブ画面の[コンポーネント]タブに移動し、[コンポーネントの追加]ボタンを選択します(図 3-14)。その後、[ソフトウェアコンポーネントの選択]ダイアログの[基本設定]を選択します。

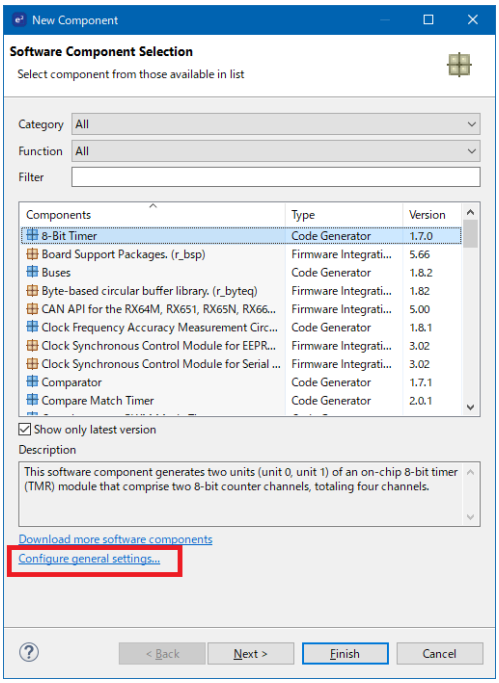

図 3-18 ソフトウェアコンポーネントの基本設定

コンポーネントの設定で[全ての FIT モジュールを表示する]にチェックを付けて、[適用]を選択します。

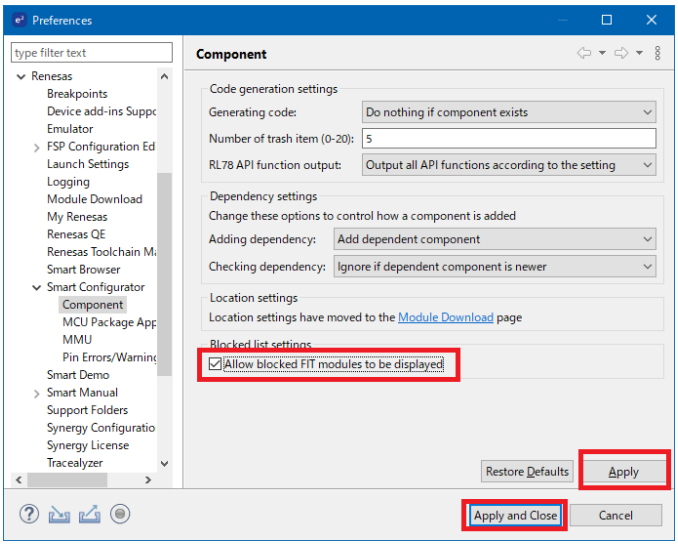

図 3-19 全ての FIT モジュールを表示するにチェック

それぞれのコンポーネントを右クリックして、[バージョンの変更]を選択します。

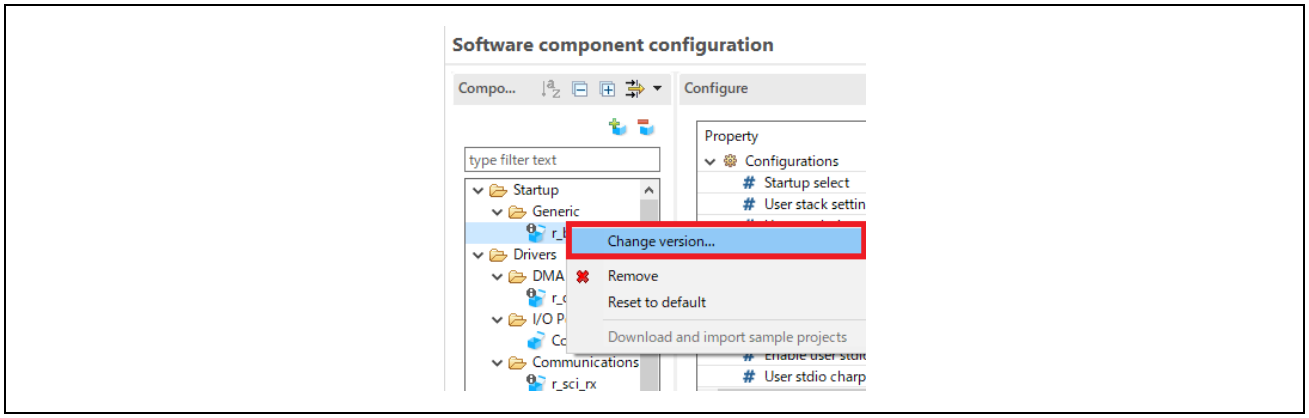

図 3-20 コンポーネントのバージョン変更

[バージョン変更] ダイアログにて[次へ]→[終了]を選択します。

利用可能なバージョンの中に最新のバージョンがある場合は、そのバージョンを選択した後、[次へ]→[終了]を選択します。

(2) FIT モジュールのコード生成

使用する FIT モジュールのソースコードを生成します。コード生成に関しては、プロジェクト毎に行う必要があります。スマート・コンフィグレータ・パースペクティブ画面の[概説]タブに移動し、[コードの生成]ボタンを選択して、必要なソースコードを生成します。

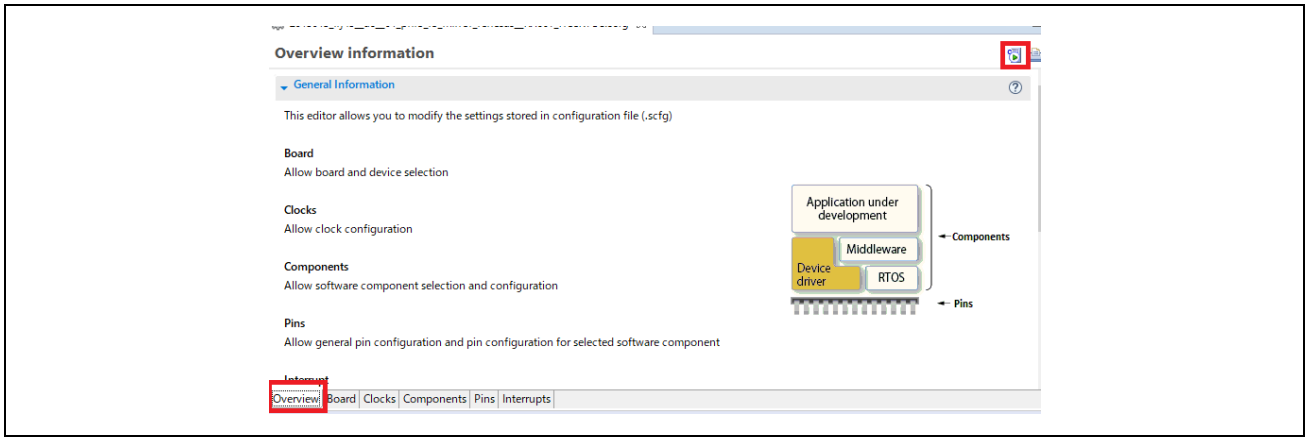

図 3-21 コード生成

以上で、プロジェクトのビルドができる状態になります。

3.3.5　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。

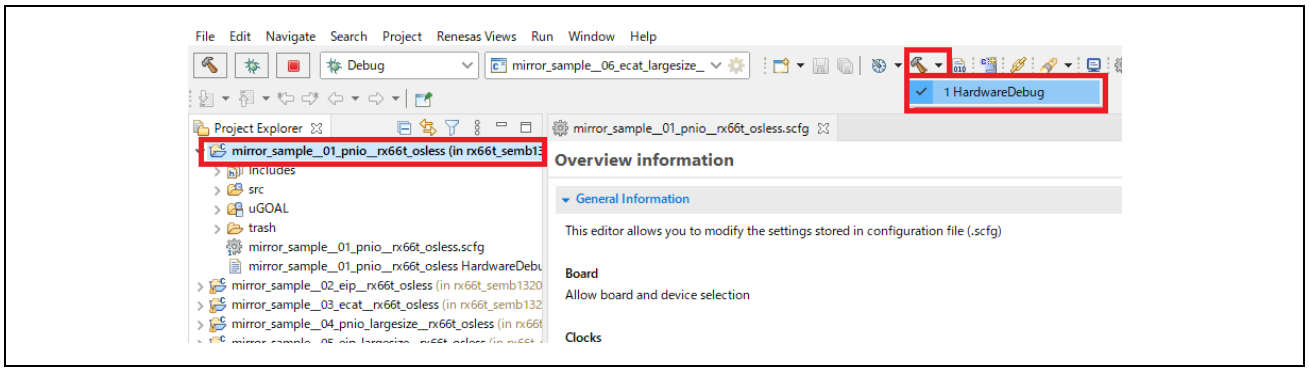

図 3-22 ビルド

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。

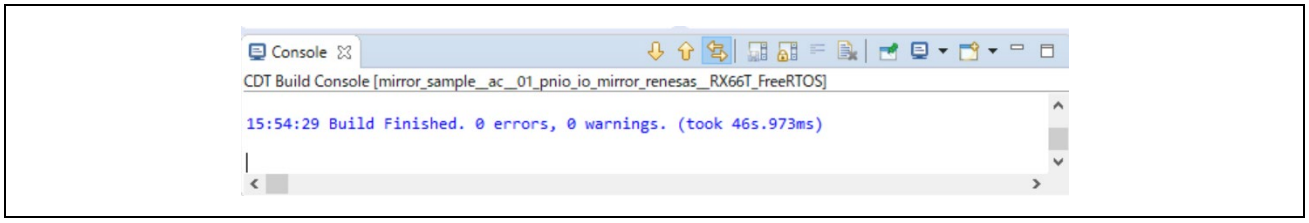

図 3-23 ビルド完了

3.3.6　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。

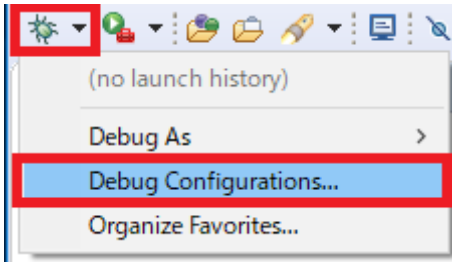

図 3-24 デバッグ構成

[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。

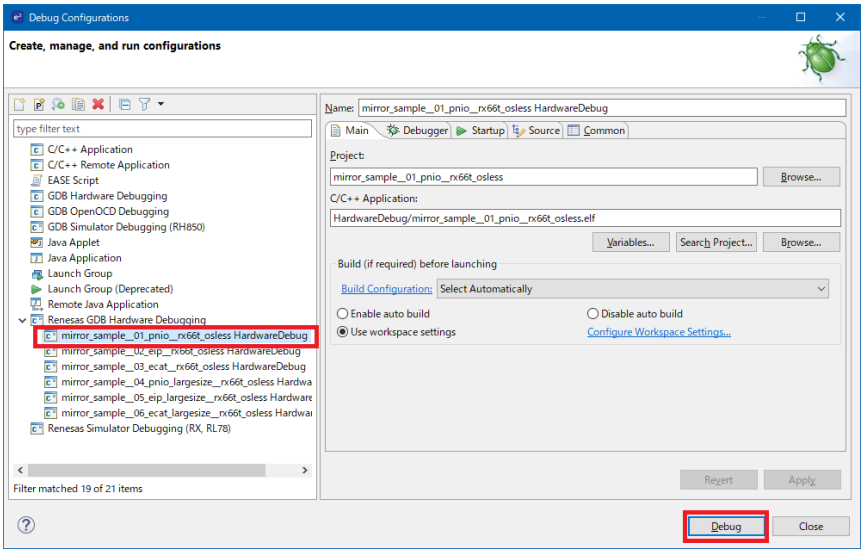

図 3-25 デバッグ開始

E2 Lite エミュレータを使用するときは、[Debugger] を開き、[Debug hardware] から使用するエミュレータを選択します。このとき、[エミュレータから電源を供給する(MAX 200mA)]設定を "いいえ" にしてください。

[Debugger] 設定を変更したら、[適用(Y)] および [デバッグ(D)]を押してください。

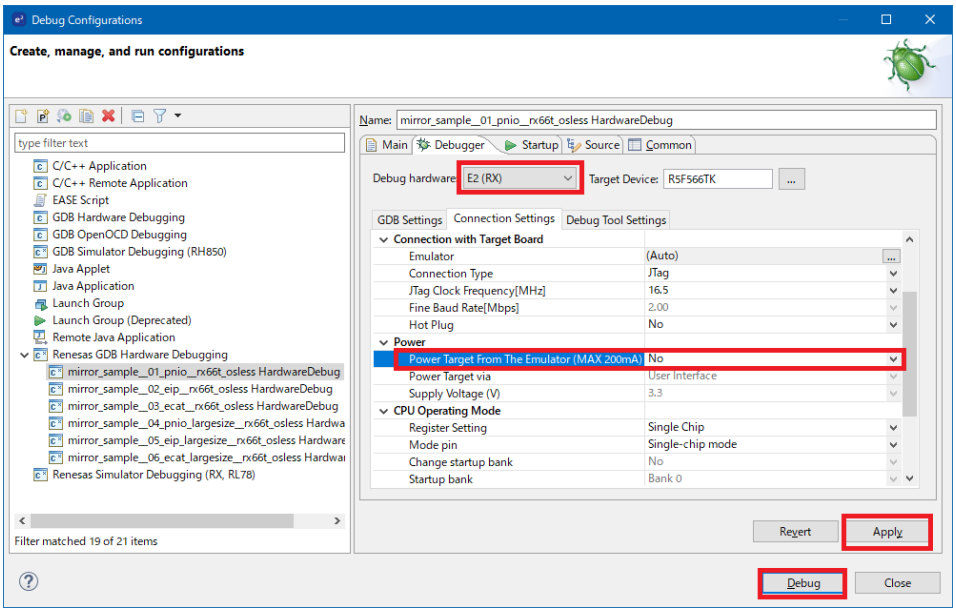

図 3-26 Debug hardware 変更

“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 3-1 に示します。

表 3-1 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
11_pnio_http	01_pnio に web ブラウザ機能、ホストマイコン Firmware 更新機能の拡張サンプル

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 (参照: 3.3.4～3.3.6)。正しくプログラムが実行されると、SEMB12320 上のプロトコル表示 LED (LED1: PROFINET)が点灯します。

図 3-27 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

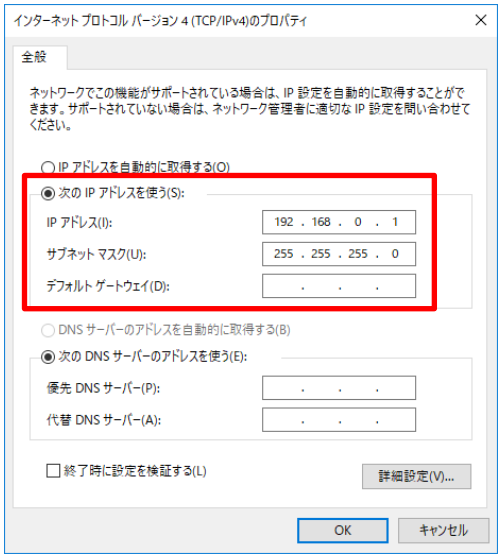

図 3-28 IP アドレス設定

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。

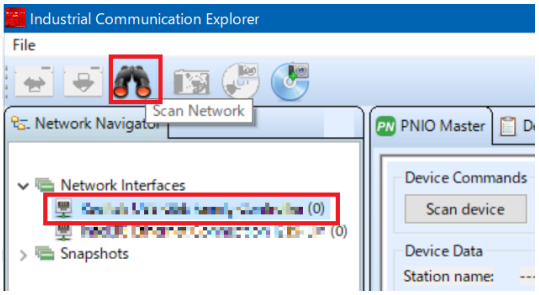

図 3-29 ネットワークスキャン

-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。

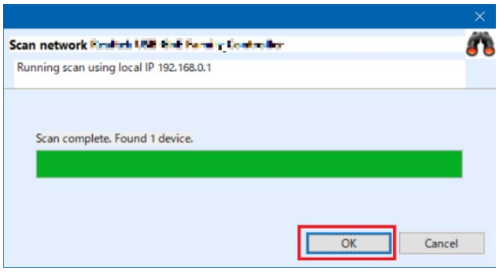

図 3-30 スキャン完了

-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。

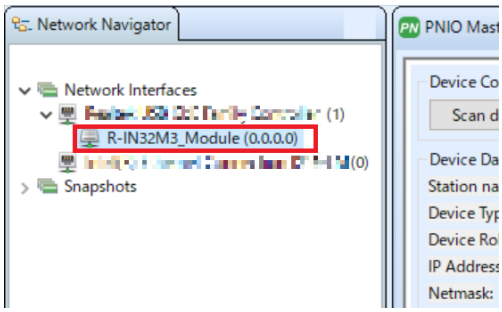

図 3-31 R-IN32M3 Module を選択

-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。

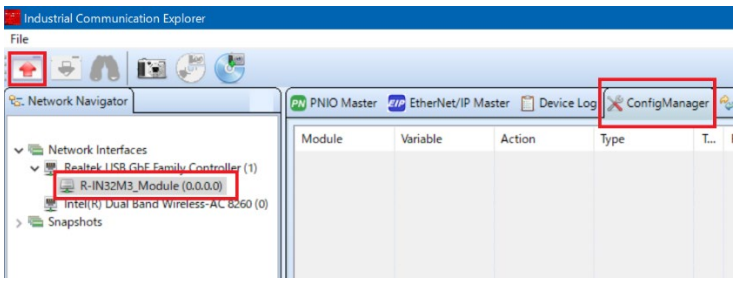

図 3-32 ConfigManager

-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

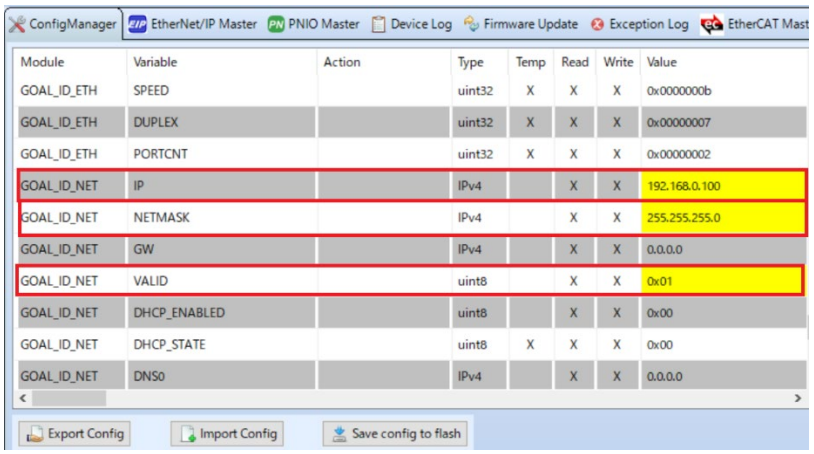

図 3-33 IP アドレス設定

-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。

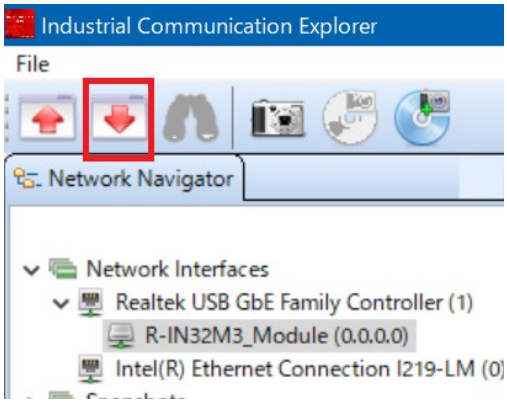

図 3-34 ダウンロード

-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。

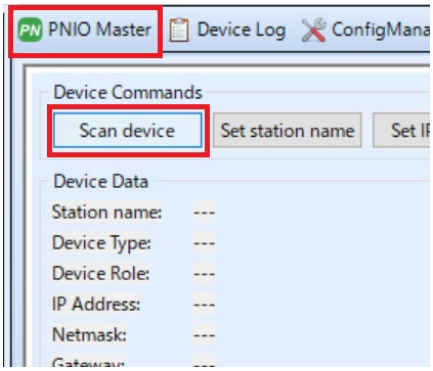

図 3-35 PNIO Master

-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。

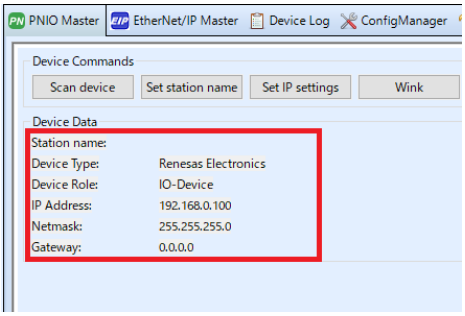

図 3-36 Device Data

-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。

表 3-2 GSDML ファイル

Sample software	GSDML file
01_pnio	01_pnio¥gsdml¥GSDML-V2.43-Renesas-irj45-20240130_01_pnio.xml
10_multi_protocol	
11_pnio_http	
04_pnio_largesize	04_pnio_largesize ¥gsdml¥GSDML-V2.43-Renesas-irj45-20240130_04_pnio.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 (LED4) が点灯します。

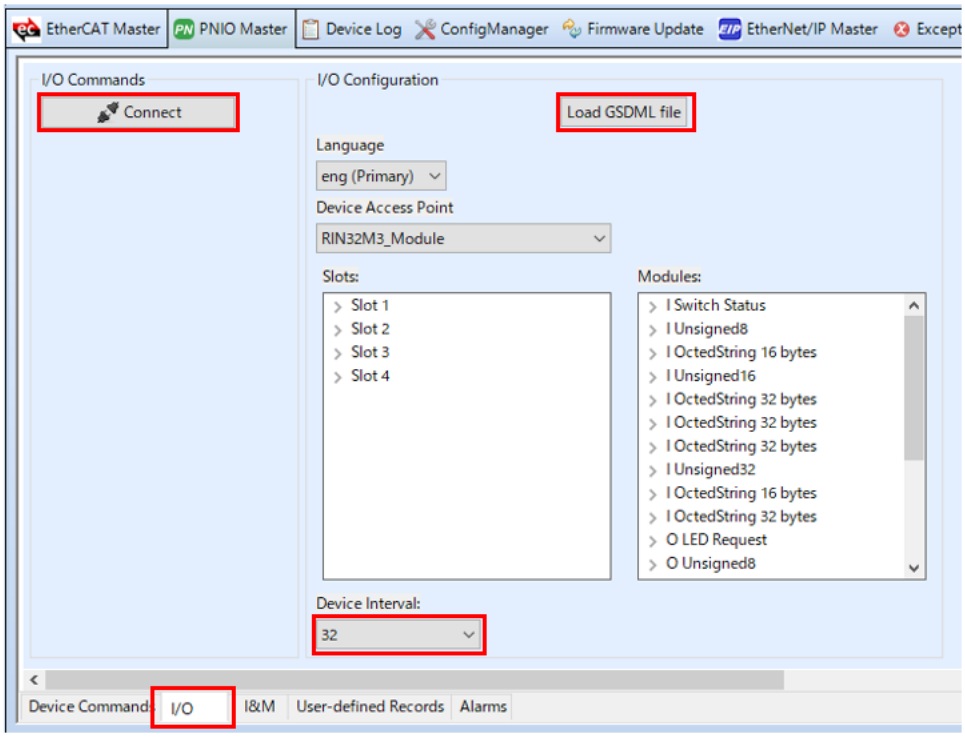

図 3-37 GSDML

-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**：評価ボードの 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), 11_pnio_http
- **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), 11_pnio_http
- **Mirror 制御(RPC)**：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：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

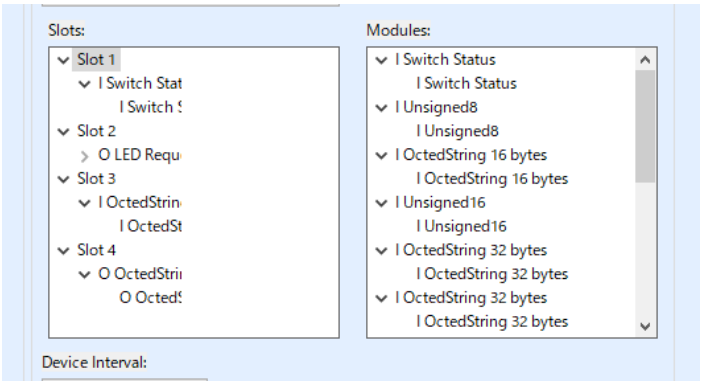

図 3-38 アプリケーション登録 (例 01_pnio)

Remote-IO (LED/Switch)制御

Switch には SEMB1320 上の汎用入力スイッチに対応した Input Data、LED には SEMB1320 上の汎用出力 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)	汎用入力スイッチ SW2, 4, 5, 6	汎用スイッチを操作することで Input Data の値が変化 SW2：bit0 SW4：bit1 SW5：bit2 SW6：bit3
LED (Slot 2)	汎用出力 LED LED5, 6, 8, 9	Output Data に値を登録することで 汎用出力 LED が変化 bit0：LD5 bit1：LD6 bit2：LD8 bit3：LD9

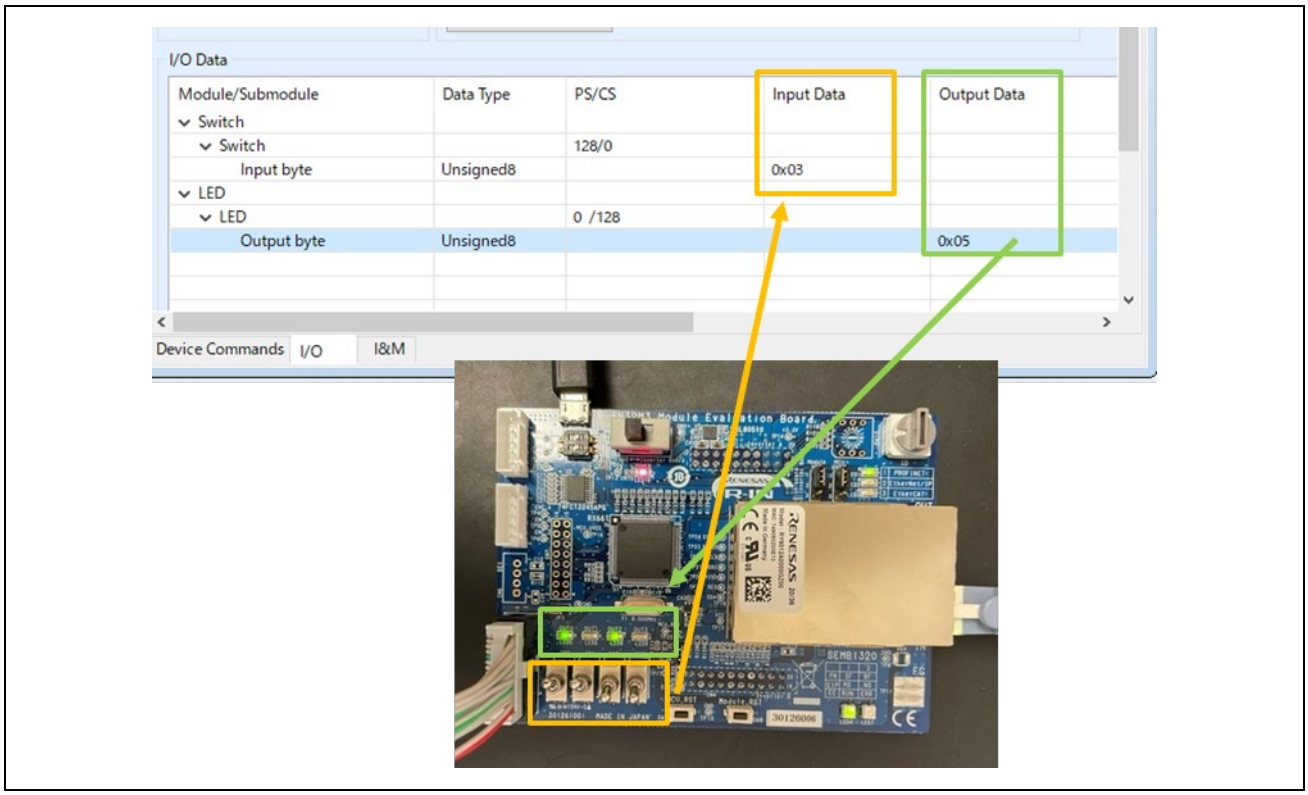

図 3-39 Remote-IO (LED/Switch)制御 [PROFINET]

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

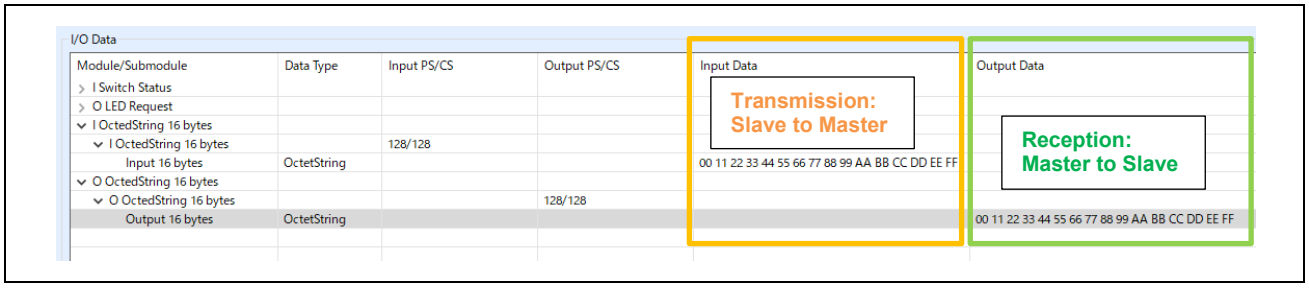

図 3-40 Mirror 制御 [PROFINET]

-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
12_eip_http	02_eip に web ブラウザ機能、ホストマイコン Firmware 更新機能の拡張サンプル

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 (参照: 3.3.4～3.3.6)。正しくプログラムが実行されると、SEMB12320 上のプロトコル表示 LED (LED2: EtherNet/IP)が点灯します。

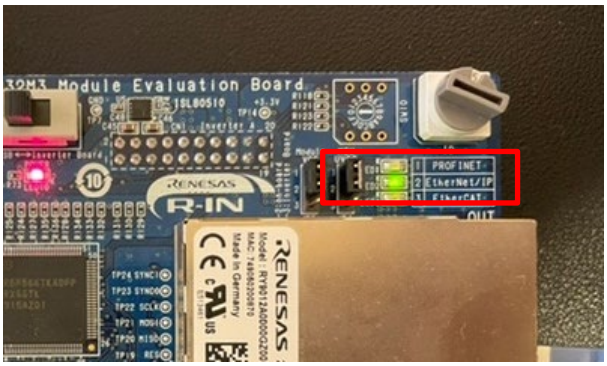

図 3-41 プロトコル LED: EtherNet/IP

-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

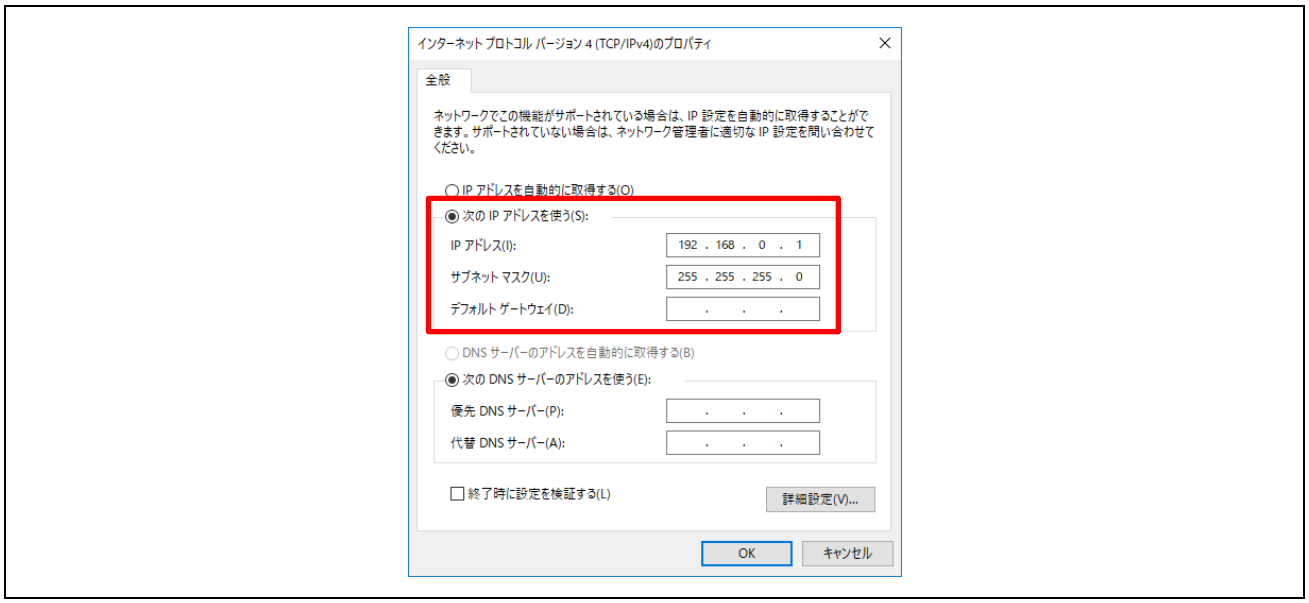

図 3-42 プロトコル LED: PROFINET

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。

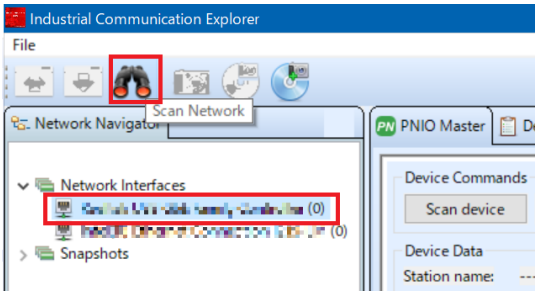

図 3-43 ネットワークスキャン

-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。

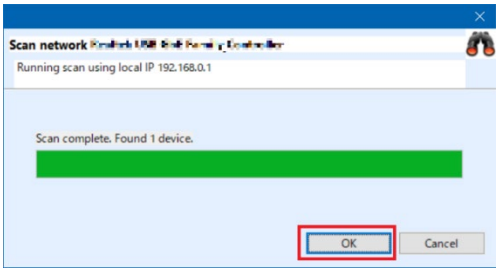

図 3-44 スキャン完了

-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。

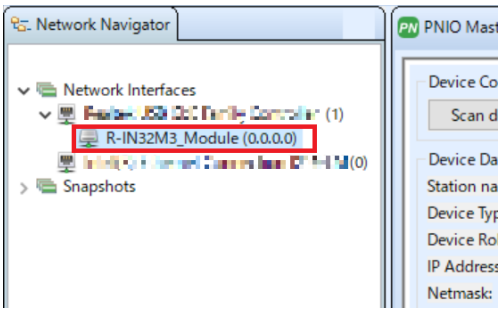

図 3-45 R-IN32M3 Module を選択

-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。

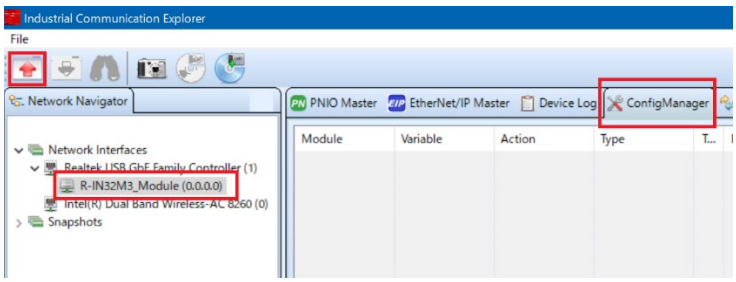

図 3-46 ConfigManager

-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

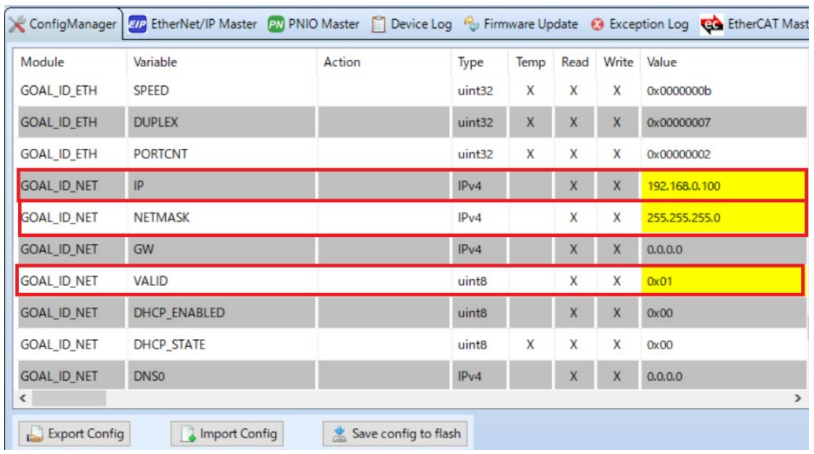

図 3-47 IP アドレス設定

-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。

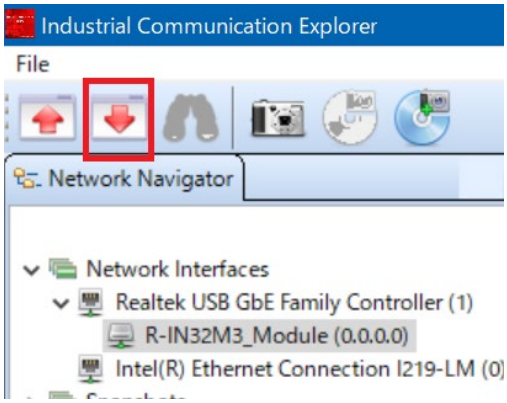

図 3-48 ダウンロード

-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。

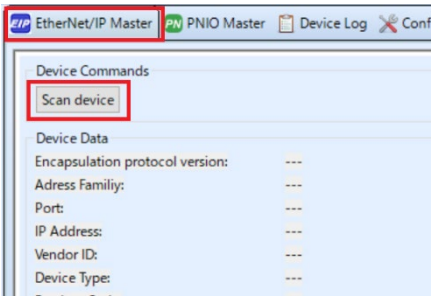

図 3-49 EtherNet/IP Scanner

-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。

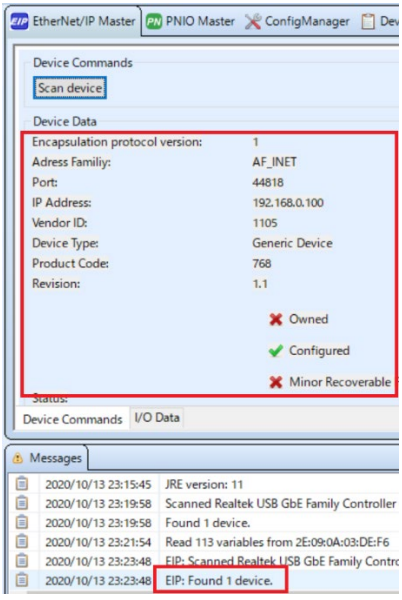

図 3-50 Device Data

-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**：評価ボードの 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), 12_eip_http
- **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), 12_eip_http
- **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。

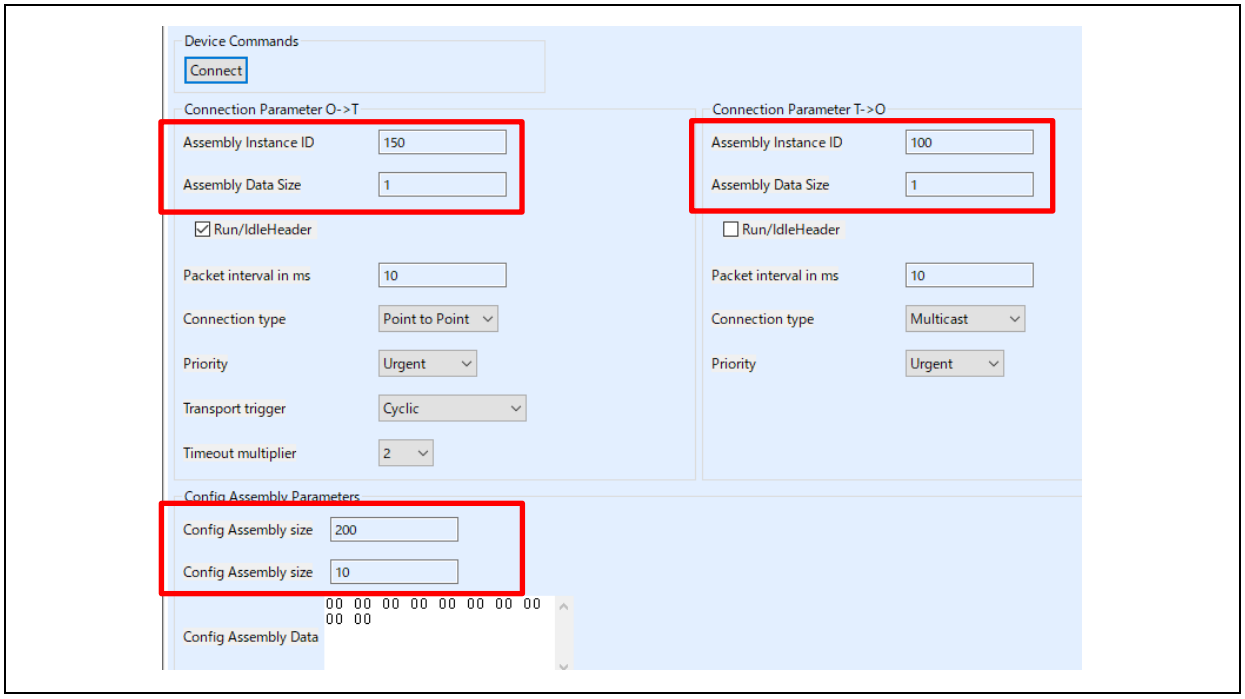

図 3-51 Remote-IO アプリケーションパラメータ設定

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。

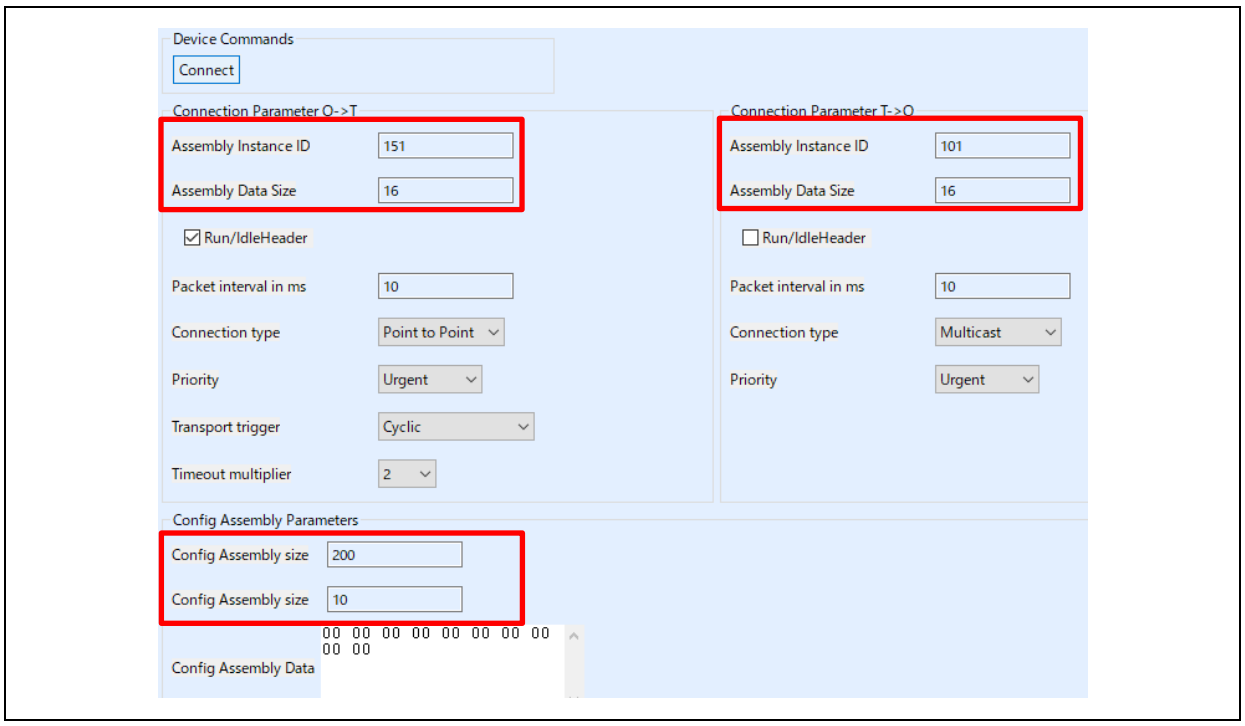

図 3-52 Mirror アプリケーションパラメータ設定

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 3-53 では Mirror Data Reception_1(152)、Mirror Data- Transmission_1(102) 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。

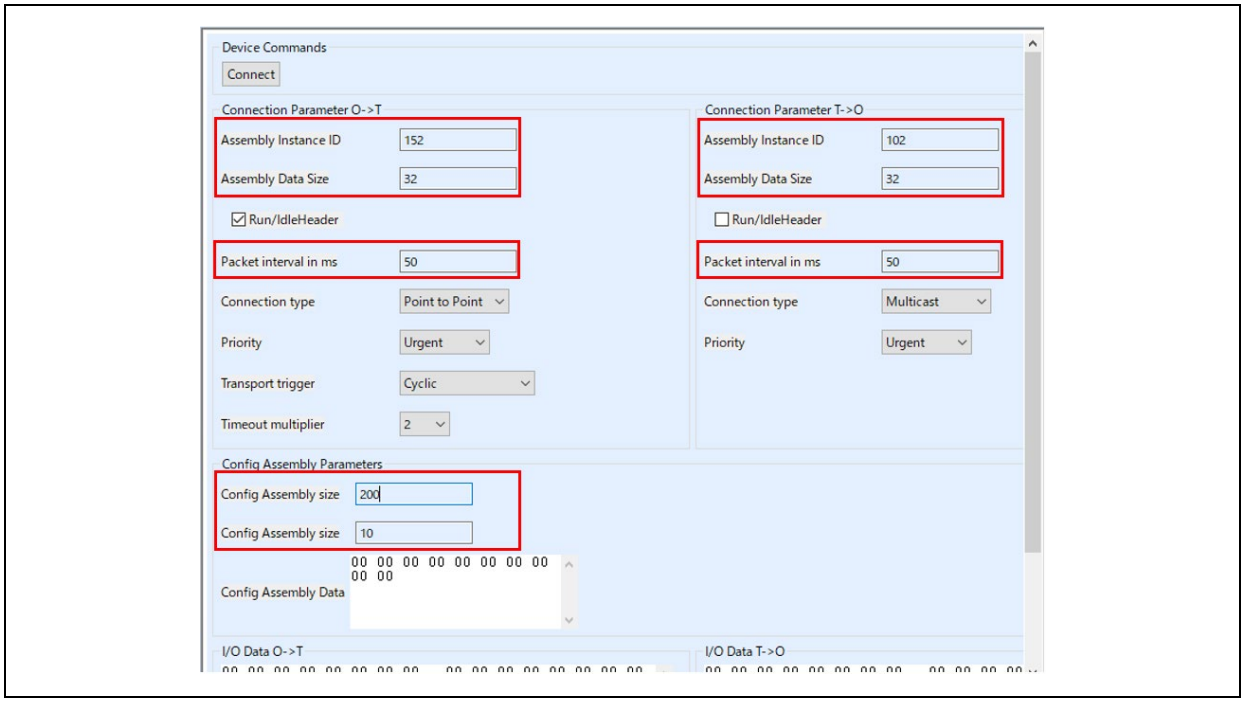

図 3-53 Mirror(rpc)アプリケーションパラメータ設定

-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 (LED4) 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 SEMB1320 上の汎用入力スイッチに対応した Input Data、LED には SEMB1320 上の汎用出力 LED に対応した Output Data が 1byte データとして登録されています。

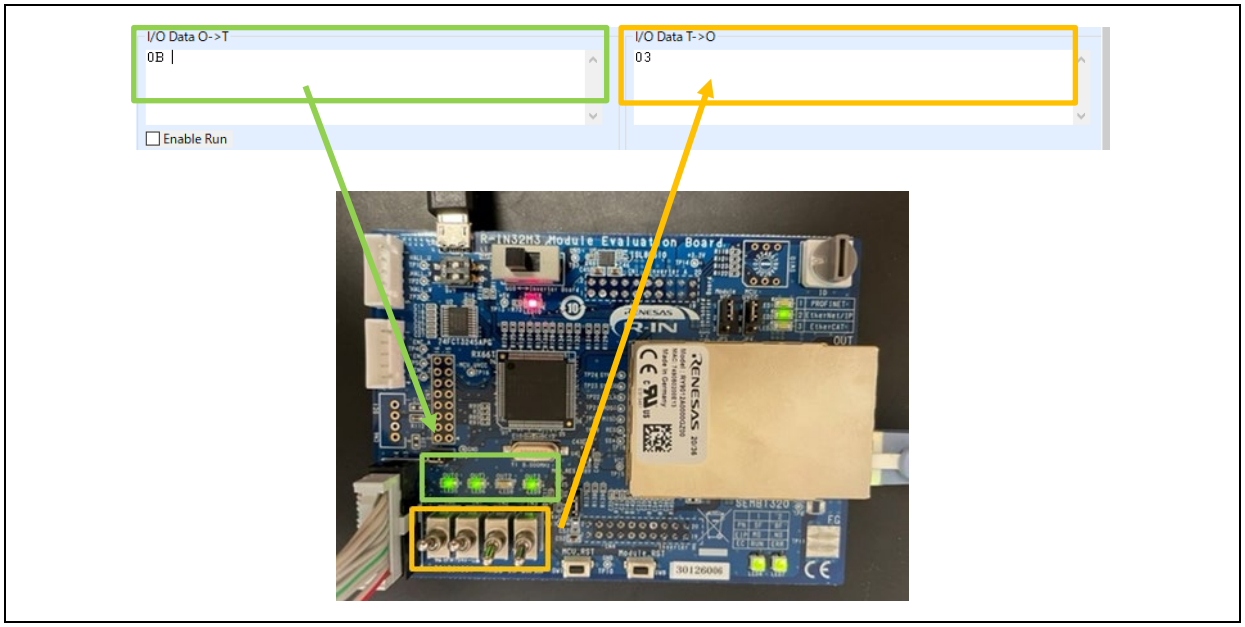

図 3-54 Remote-IO (LED/Switch)制御 [EtherNet/IP]

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。

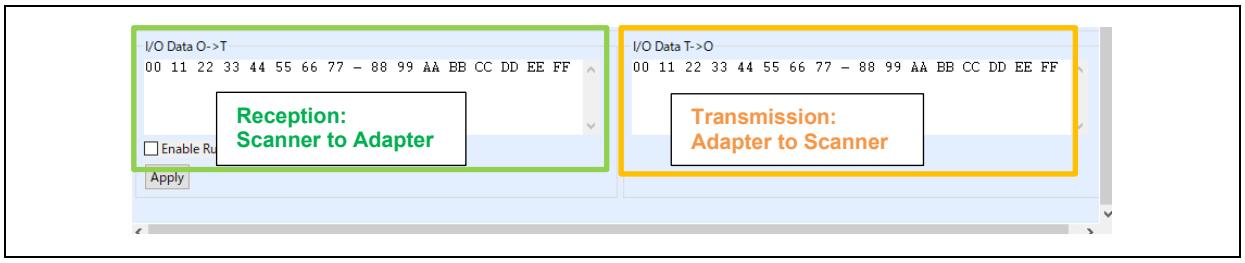

図 3-55 Mirror 制御 [EtherNet/IP]

-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
13_ecat_http	03_ecat に web ブラウザ機能、ホストマイコン Firmware 更新機能の拡張サンプル

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 (参照: 3.3.4～3.3.6)。正しくプログラムが実行されると、SEMB12320 上のプロトコル表示 LED (LED2: EtherCAT)が点灯します。

図 3-56 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 専用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

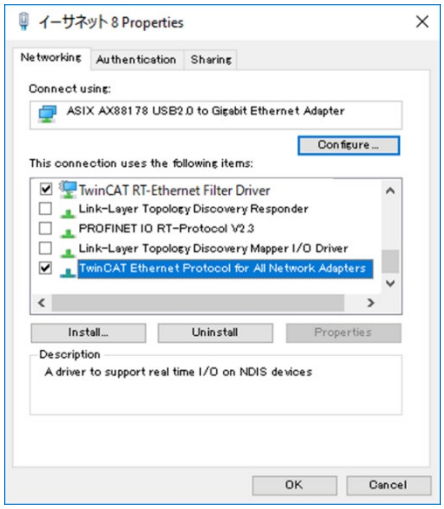

図 3-57 ネットワークアダプタ設定

[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. ス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

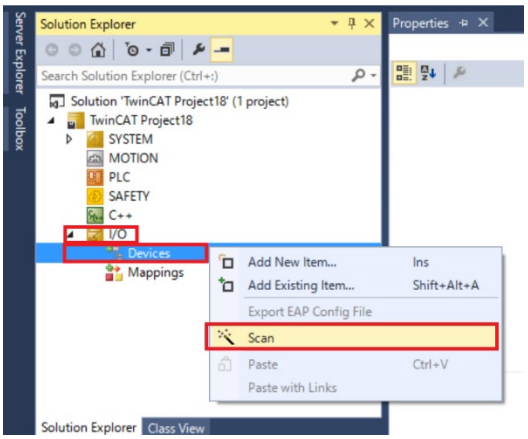

図 3-58 ネットワークスキャン

-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。

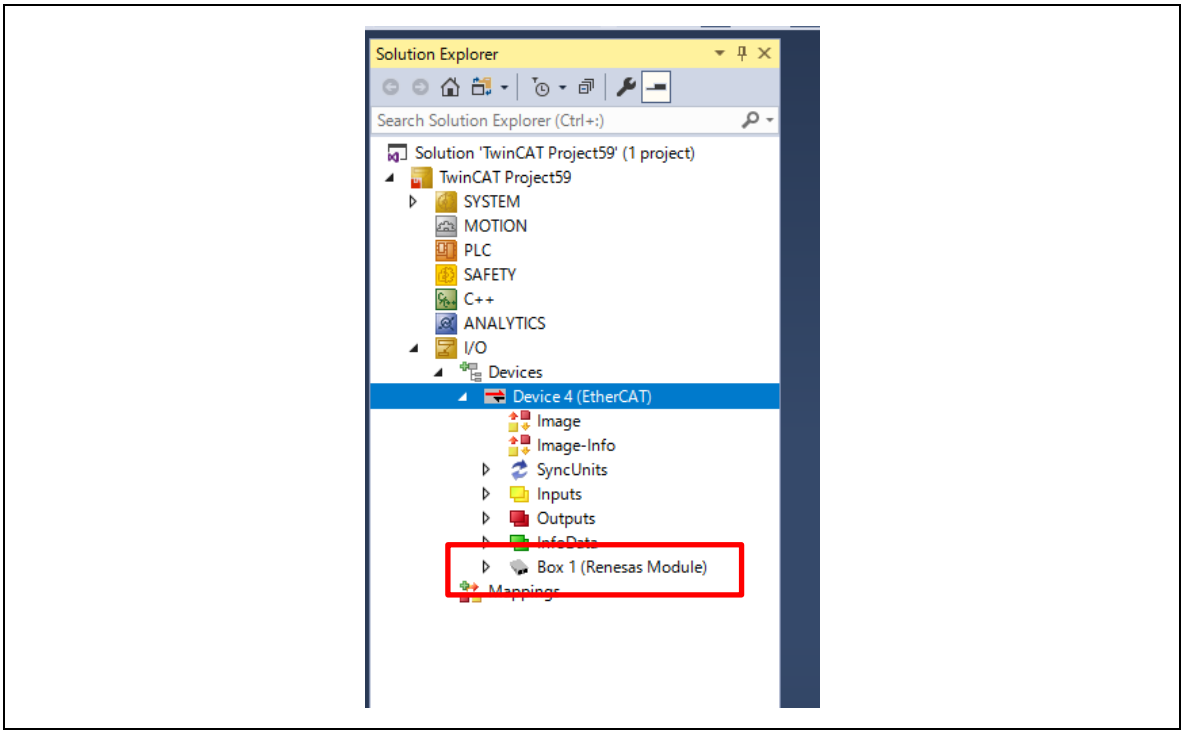

図 3-59 TwinCAT 接続

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**：評価ボードの 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), 13_ecat_http
- **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), 13_ecat_http
- Mirror 制御(RPC)：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：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

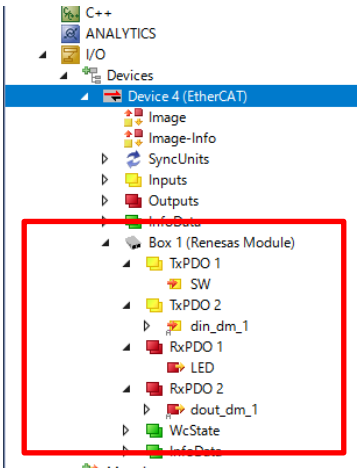

図 3-60 アプリケーション登録 (例 03_ecat)

Remote-IO (LED/Switch)制御

Switch には SEMB1320 上の汎用入力スイッチに対応した Input Data、LED には SEMB1320 上の汎用出力 LED に対応した Output Data が 1byte データとして登録されています。

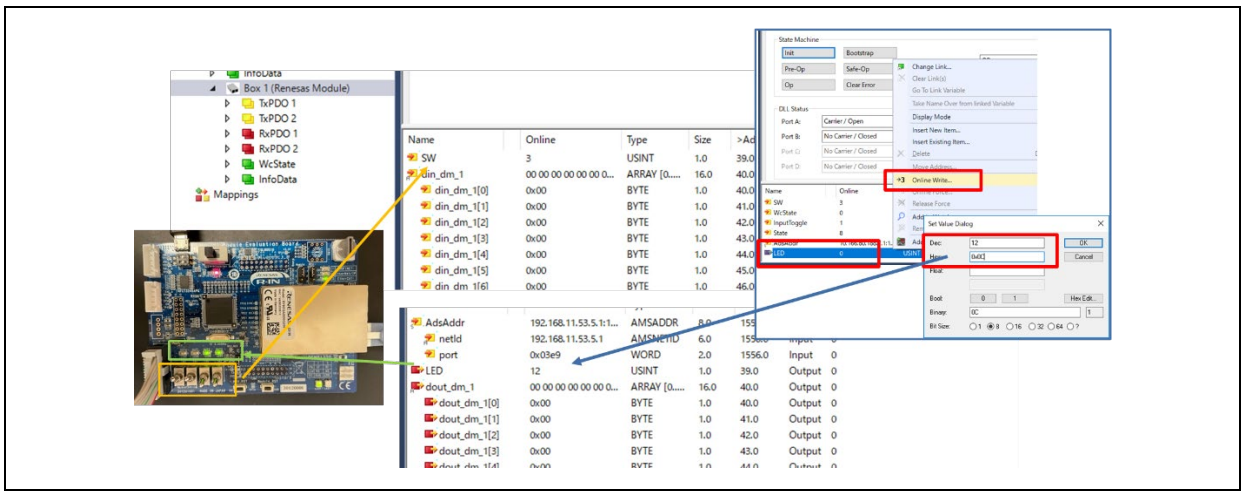

図 3-61 Remote-IO 制御 [EtherCAT]

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。

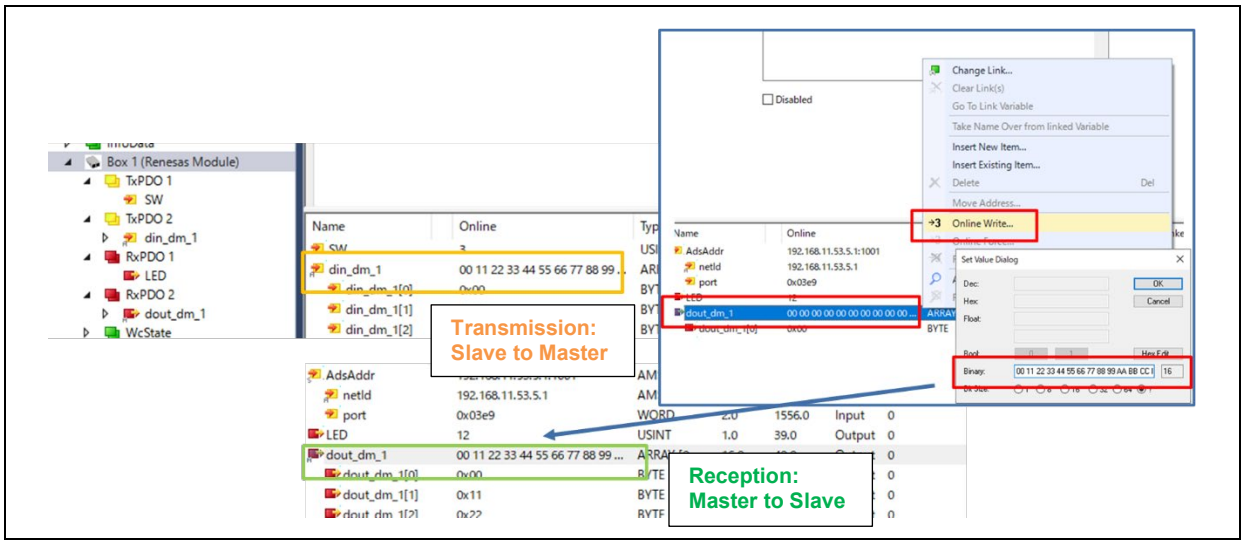

図 3-62 Mirror 制御 [EtherCAT]

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

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 (参照: 3.3.4～3.3.6)。汎用スイッチ (SW9)の値に応じて使用するプロトコルが実行されます。

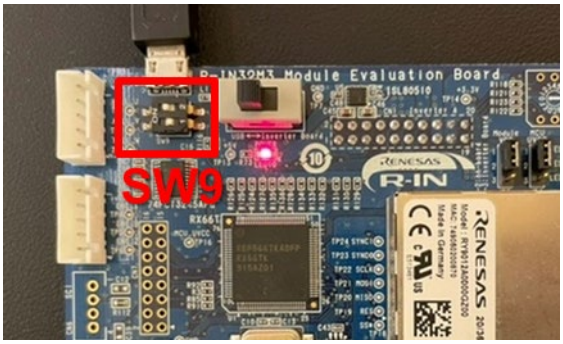

図 3-63 汎用スイッチ (SW9)

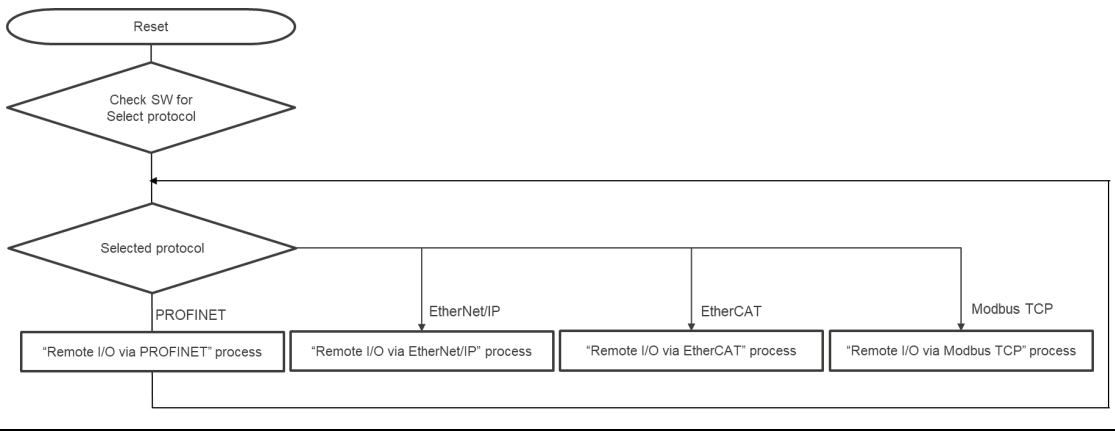

図 3-64 multi-protocol サンプルアプリケーションの流れ

SW9	Protocol	LED
0	ModbusTCP Server	全消灯
1	PROFINET	LED1 点灯
2	EtherNet/IP	LED2 点灯
3	EtherCAT	LED3 点灯

Modbus TCP Server：SW9 設定 [0]

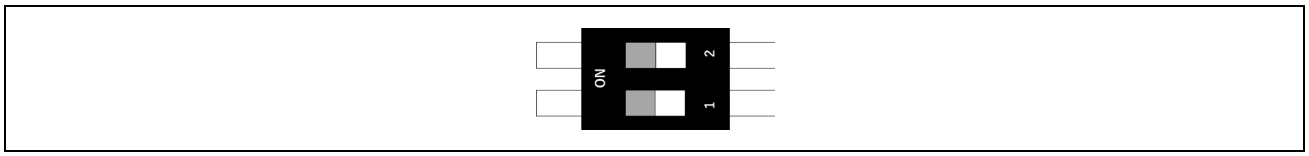

図 3-65 Modbus TCP 選択

PROFINET：SW9 設定 [1]

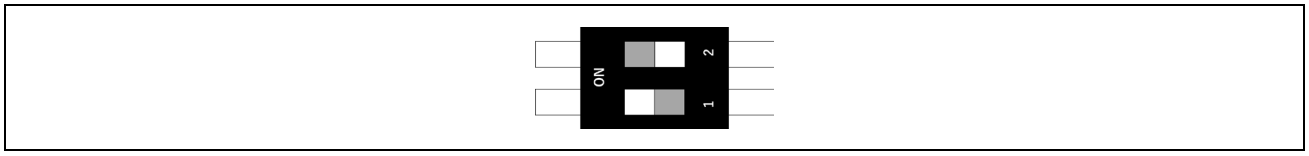

図 3-66 PROFINET 選択

EtherNet/IP：SW9 設定 [2]

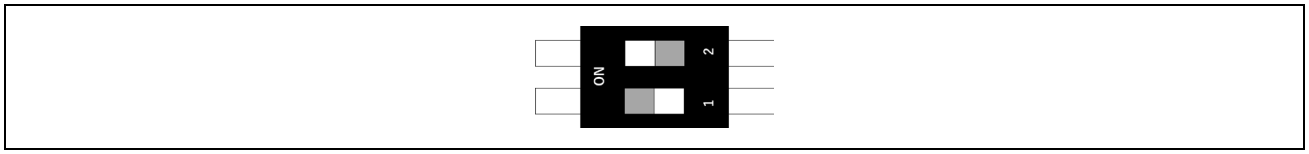

図 3-67 EtherNet/IP 選択

EtherCAT：SW9 設定 [3]

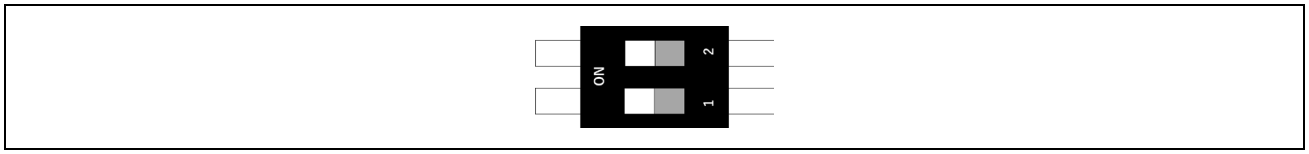

図 3-68 EtherCAT 選択

-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

3.4.6　Web サーバ機能

Web サーバ機能に対応したのサンプルアプリケーションについて説明します。
web コンテンツは *index.html* に 2.5.2(3) プロトコルステータス表示 (LED4,7)の状態を示すサンプルとなっています。これらの html データは *goal_http_fs.h* に記述されています。

対象サンプルを表 3-11 に示します。

表 3-11 Web Server Sample software

Sample software	Overview
11_pnio_http	01_pnio に web ブラウザ機能、ホストマイコン Firmware 更新機能の拡張サンプル
12_eip_http	02_eip に web ブラウザ機能、ホストマイコン Firmware 更新機能の拡張サンプル
13_ecat_http	03_ecat に web ブラウザ機能、ホストマイコン Firmware 更新機能の拡張サンプル

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。

評価環境セットアップ

選択したプロトコルにあわせて各章の評価環境セットアップ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

Web ブラウザのアクセス手法を説明します。

PROFINET, EtherNet/IP

ここでは、以下の条件を例に説明します。

PC ネットワーク設定	**[IP]** 192.168.0.1 **[MASK]** 255.255.255.0
R-IN32M3 Module	**[IP]** 192.168.0.100 **[MASK]** 255.255.255.0

プログラムを実行中に web ブラウザに R-IN32M3 Module に指定している IP アドレス(192.168.0.100)を入力しアクセスすることで、サンプルとして用意されている web サーバが読み込まれます。

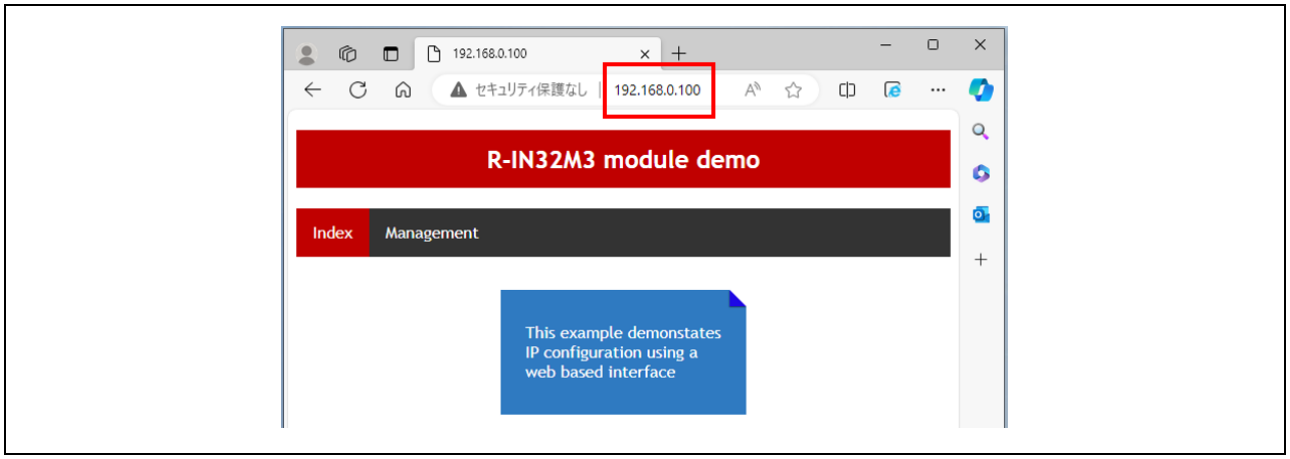

図 3-69 web ブラウザアクセス

EtherCAT

ここでは、以下の条件を例に説明します。

EtherCAT web サーバでは TwinCAT を利用し、以下の条件を例に説明します。

PC ネットワーク設定	**[IP]** 192.168.1.99 **[MASK]** 255.255.255.0
R-IN32M3 Module (EtherCAT EoE)	**[IP]** 192.168.1.100 **[MASK]** 255.255.255.0

-1. ネットワークアダプタ設定で TwinCAT Driver および 固定 IP を有効にします。

- **TwinCAT RT-Ethernet Filter Driver**
- **TwinCAT Ethernet Protocol for All Network Adapters**
- **インターネットプロトコルバージョン 4(TCP/IPv4)**

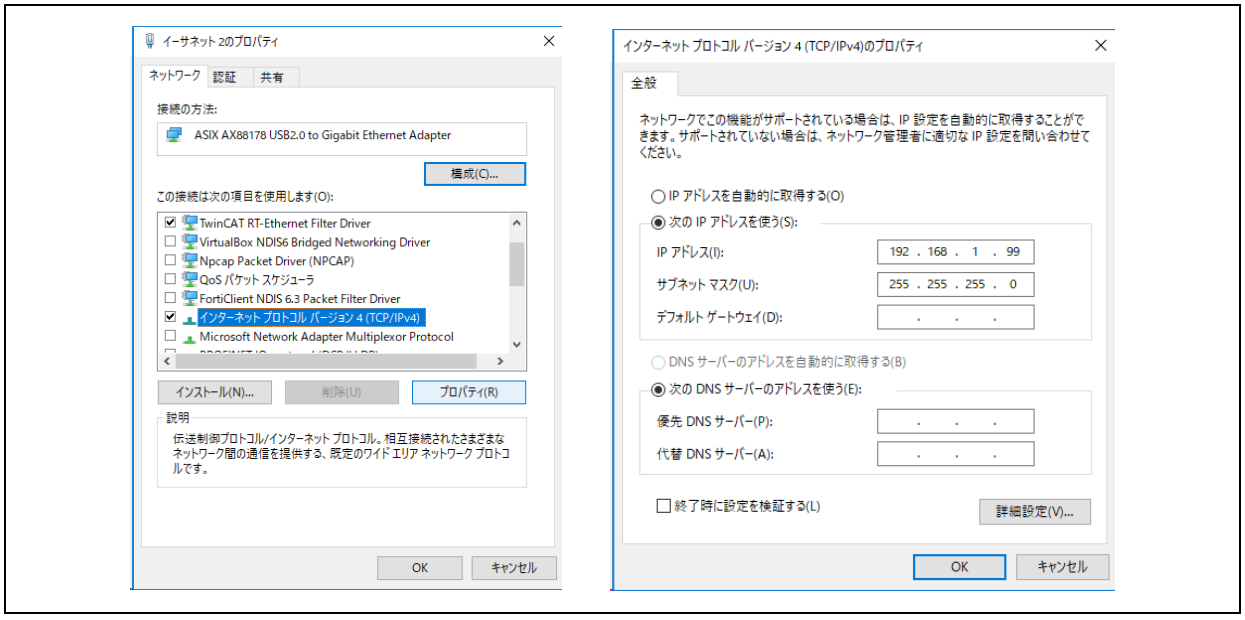

図 3-70 EtherCAT Web アクセスネットワーク設定

-2. TwinCAT 接続し、OP の状態で以下を設定します。その他は初期値設定としてください。
TwinCAT 接続手順は、3.4.3 EtherCAT を参照してください。

Slave 選択 > EtherCAT タブ > Advanced Settings...
Mailbox > “EoE” 選択し、**IP Port** および **IP Address** を設定

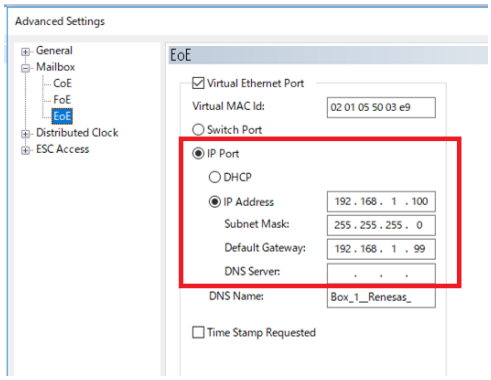

図 3-71 TwinCAT EoE 設定

-3. Restart TwinCAT (Config Mode) を実行し、TwinCAT を再接続します。

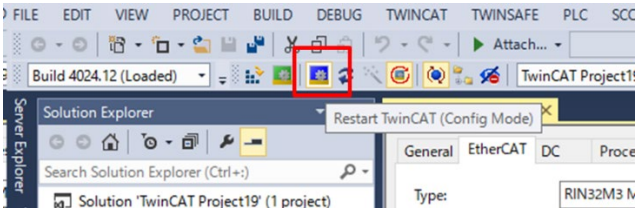

図 3-72 TwinCAT 再接続

-4. web ブラウザに IP アドレス(192.168.1.100)を入力しアクセスすることで、サンプルとして用意されている web サーバが読み込まれます。

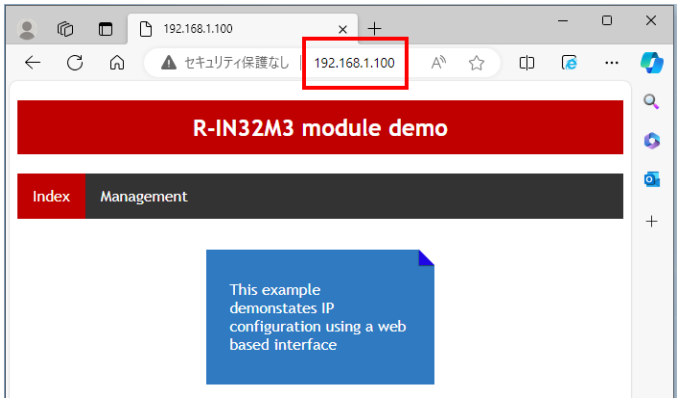

図 3-73 web ブラウザアクセス

3.5　アプリケーションインプリガイド

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。

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。

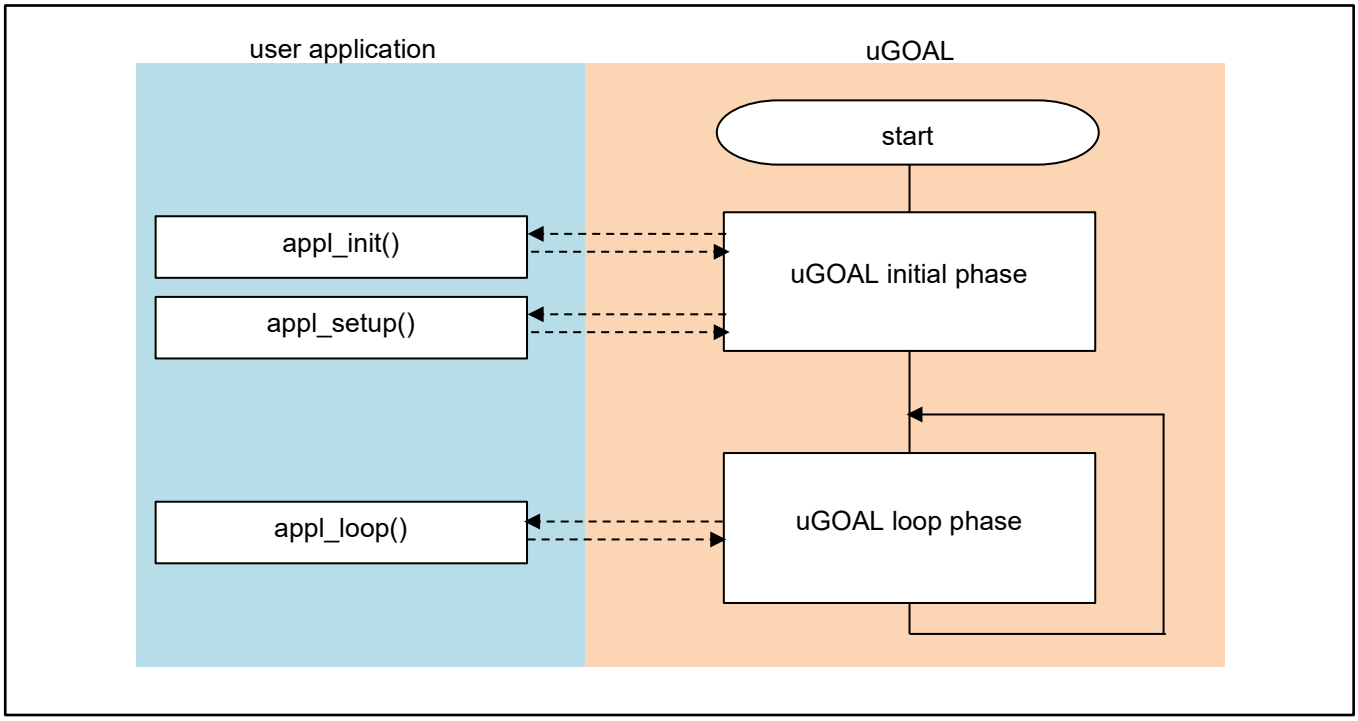

図 3-74　プログラム全体の流れ

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 3-12 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

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)) {                                                             ③
        goal_logErr(“failed to create a new PROFINET instance”);
        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

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、EtherCAT Explicit Device ID スイッチ (SW3)を用います。詳細は 2.5.3(1)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

デバッグ用途を目的としたログメッセージを出力することができます。

シリアル通信を介して、PC ターミナルソフト(TeraTerm 等)にログメッセージを出力する手順を以下に示します。

下記のコンパイルマクロを変更することで有効になります。サンプルアプリケーションでは、デフォルト無効となっています。

表 4-1 ログメッセージ出力方法とコンパイルマクロ

出力	コンパイルマクロ	デフォルト値
1	CONFIG_UGOAL_LOGGING	0

以降でそれぞれの出力方法について説明します。

4.2.1　TeraTerm での確認

サンプルソフトのログメッセージが、UART ドライバを介して本ボードの UART 通信ラインに送信されます。USB-UART コンバータケーブル(*)を用いて PC と本ボードを接続することで、TeraTerm 上でログメッセージを確認することができます。なお、R-IN32M3 Module のログメッセージは、TeraTerm に出力されません。

以下に手順を示します。

(*) “TTL-232R-RPI”などの USB-UART コンバータケーブルを別途ご用意ください。

(1) USB-UART コンバータケーブル(*)を用いて本ボードと PC を接続します。

- 本ボードの CN6 にて、Pin3 に USB-UART コンバータケーブルの TX を接続
- 本ボードの CN6 にて、Pin2 に USB-UART コンバータケーブルの RX を接続
- 本ボードの CN6 にて、Pin4 に USB-UART コンバータケーブルの GND を接続

CN6 の SCI コネクタ(B4B-XH-A)は未実装となります(2.5.4(5)参照)ので、個別に実装が必要です。

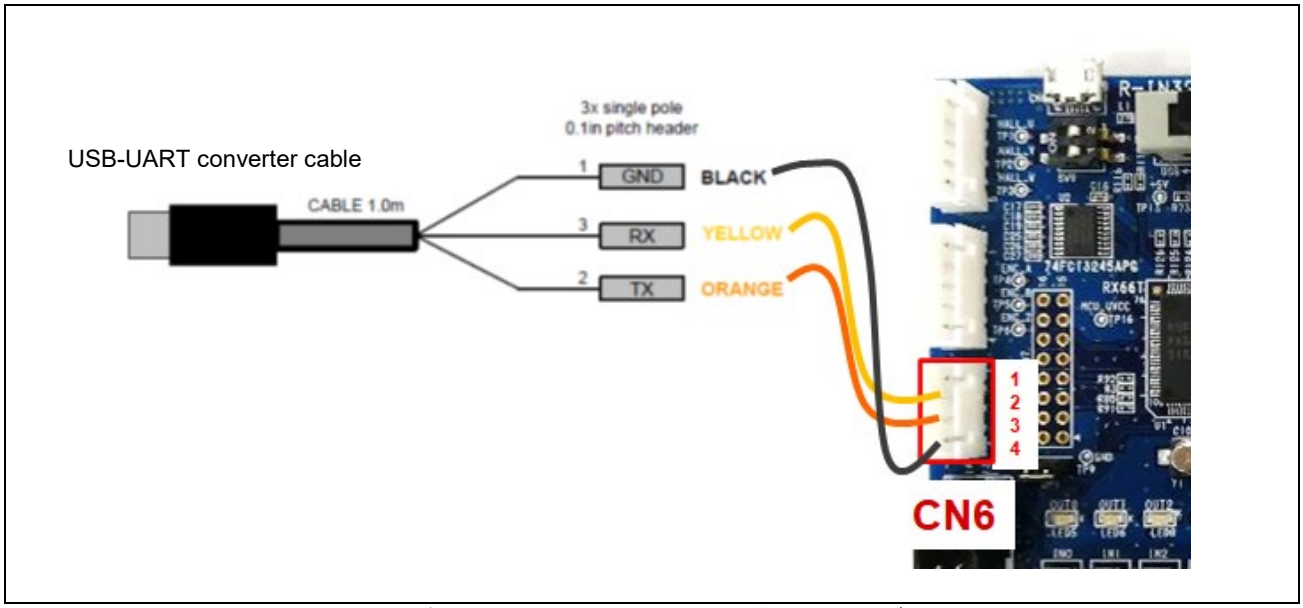

図 4-1 本ボードと USB-UART コンバータケーブルの接続図

(2) PC にて TeraTerm を起動して、シリアル設定を行います。

Speed	115200
Data	8 bits
Parity	none
Stop bit	1 bit
Flow control	none

(3) サンプルアプリケーション毎のフォルダの中の goal_config.h にて、”CONFIG_UGOAL_LOGGING”マクロの値を 0(デフォルト)→1 に変更します。

例)

ファイル: ¥appl¥ugoal¥01_pnio¥goal_config.h

```
#define CONFIG_UGOAL_LOGGING (0)
```

(4) 3.3.5～3.3.6 章を参考に、該当プロジェクトをビルドしてサンプルアプリケーションを実行します。

(5)3.4.1.1 章を例に Management tool を操作して、本サンプルアプリケーションとの接続、並びに、サイクリック通信を行います。

これにより、TeraTerm には以下の様なログメッセージが表示されます。

```
[INF] C:/Work/ws/XXXX_CCM_V/rpc/wrapper/pnio/goal_pnio_rpc_ac.c:286 auto data mapper for APDU is disabled
[INF] C:/Work/ws/XXXX_CCM_V/rpc/wrapper/pnio/goal_pnio_rpc_ac.c:360 PROFINET Application Core successfully started
[INF] C:/Work/ws/XXXX_CCM_V/appl/ugoal/01_pnio/goal_appl.c:281 Initializing device structure
[INF] C:/Work/ws/XXXX_CCM_V/rpc/goal_media/goal_mi_dm.c:296 in Write to CC part added at pos: 0, len: 1
[INF] C:/Work/ws/XXXX_CCM_V/rpc/goal_media/goal_mi_dm.c:296 in Read from CC part added at pos: 0, len: 1
[INF] C:/Work/ws/XXXX_CCM_V/rpc/goal_media/goal_mi_dm.c:296 in Write to CC part added at pos: 1, len: 1
[INF] C:/Work/ws/XXXX_CCM_V/rpc/goal_media/goal_mi_dm.c:296 in Read from CC part added at pos: 1, len: 1
[INF] C:/Work/ws/XXXX_CCM_V/rpc/goal_media/goal_mi_dm.c:296 in Write to CC part added at pos: 2, len: 1
[INF] C:/Work/ws/XXXX_CCM_V/rpc/goal_media/goal_mi_dm.c:296 in Read from CC part added at pos: 2, len: 1
[INF] C:/Work/ws/XXXX_CCM_V/rpc/goal_media/goal_mi_dm.c:296 in Write to CC part added at pos: 3, len: 2
[INF] C:/Work/ws/XXXX_CCM_V/rpc/goal_media/goal_mi_dm.c:296 in Read from CC part added at pos: 3, len: 1
[INF] C:/Work/ws/XXXX_CCM_V/rpc/goal_media/goal_mi_dm.c:296 in Write to CC part added at pos: 5, len: 1
[INF] C:/Work/ws/XXXX_CCM_V/rpc/goal_media/goal_mi_dm.c:296 in Read from CC part added at pos: 4, len: 2
[INF] C:/Work/ws/XXXX_CCM_V/rpc/goal_media/goal_mi_dm.c:296 in Write to CC part added at pos: 6, len: 1
[INF] C:/Work/ws/XXXX_CCM_V/rpc/goal_media/goal_mi_dm.c:296 in Read from CC part added at pos: 6, len: 1
[INF] C:/Work/ws/XXXX_CCM_V/appl/ugoal/01_pnio/goal_appl.c:498 PROFINET ready
[INF] C:/Work/ws/XXXX_CCM_V/appl/ugoal/01_pnio/goal_appl.c:500 Configuring DD
[INF] C:/Work/ws/XXXX_CCM_V/appl/ugoal/01_pnio/goal_appl.c:524 DD ready
[INF] C:/Work/ws/XXXX_CCM_V/rpc/goal_media/goal_mi_mctc.c:525 local setup done
[INF] C:/Work/ws/XXXX_CCM_V/ugoal/ugoal.c:307 HEAP utilization: 8720/11520 (75%).
```

図 4-2 TeraTerm 上でのログメッセージ

4.3　IP アドレス設定

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。
PROFINET、EtherNet/IP に関連する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 4-2　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 4-3 に IP アドレスの設定方法の一覧を示します。

表 4-3　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　Big-endian 対応

R-IN32M3 Module サンプルソフトは リトルエンディアンとなっています。ビックエンディアン対応するには以下の手順でプロジェクトを作成してください。

1. プロジェクト作成

1-1. 新規プロジェクトを作成

メニューから新規プロジェクトを作成します。

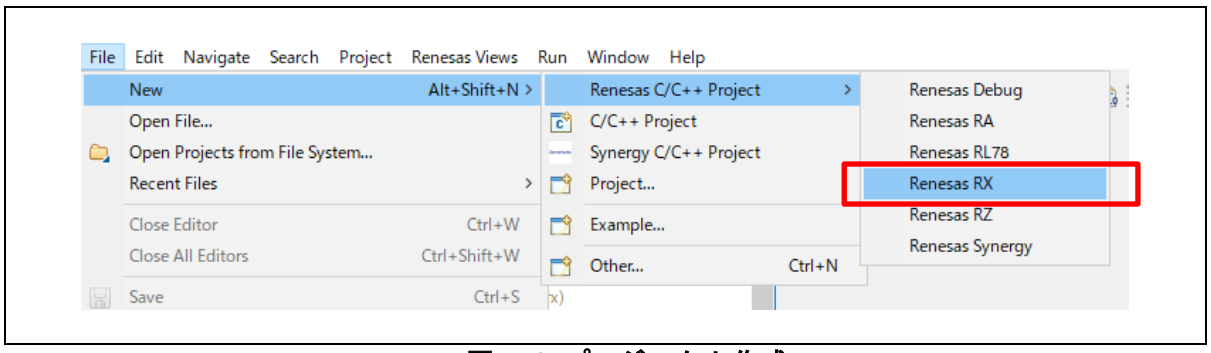

図 4-3 プロジェクト作成

1-2. 対応コンパイラの選択

作成したいコンパイラ環境 [GCC または CC-RX] を選択します。

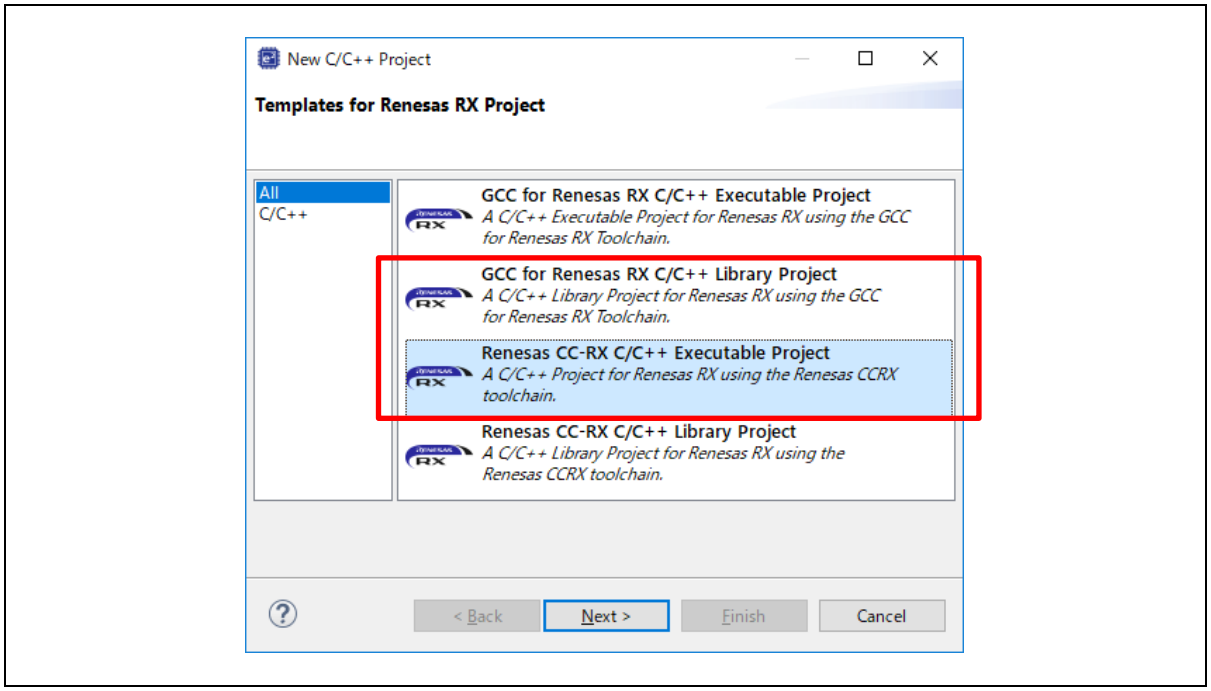

図 4-4 コンパイラ環境選択

1-3. プロジェクト名

プロジェクト名、フォルダを指定します。

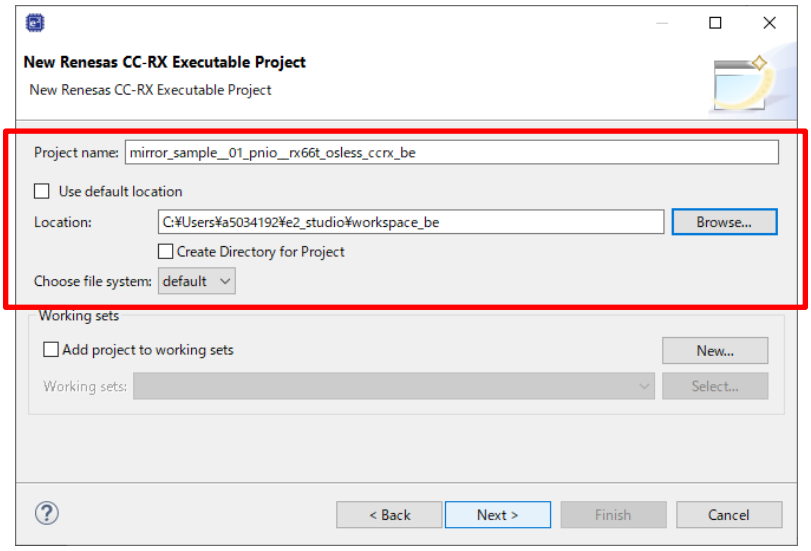

図 4-5 プロジェクト名設定

1-4. エンディアン設定とデバイス条件

エンディアンおよび 評価環境にあったデバイス条件を指定します。

Target Device：R5F566TKAxFP　* SEMB1320 の場合

Endian：Big

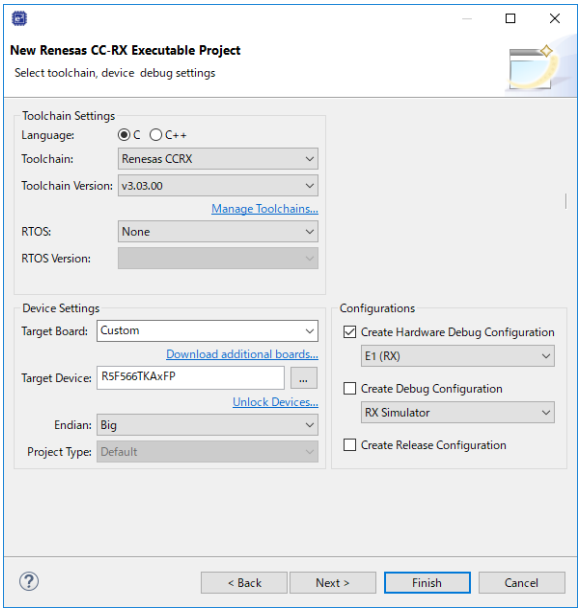

図 4-6 評価環境 設定

1-5. Smart Configurator 指定 * CC-RX コンパイラ環境のみ

“Use Smart Configurator” を有効にして Finish を押すことでプロジェクトが作成されます。

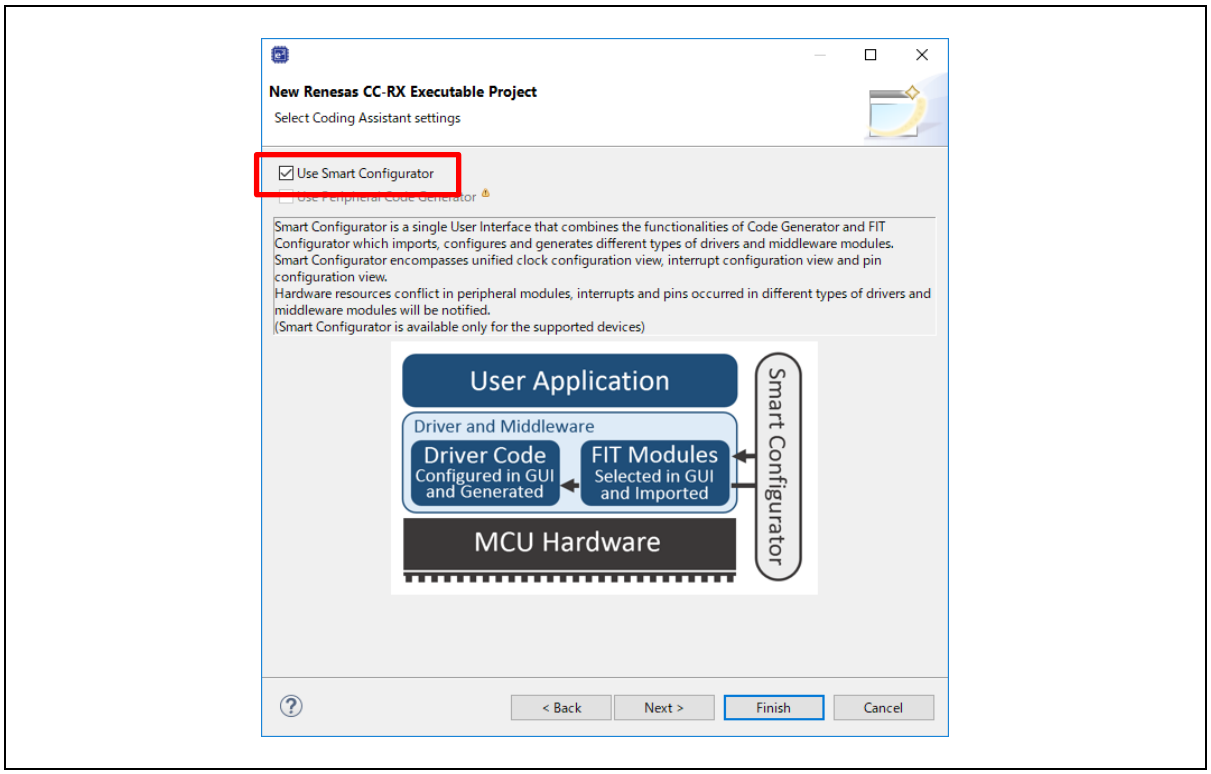

図 4-7 Smart Configurator

2. プロジェクト環境のコピー

2-1. サンプルプロジェクトからの移植

① プロジェクトが作成されると自動で表示されるコンフィグレータを閉じます。

② コンフィグレータ ファイルを削除します。

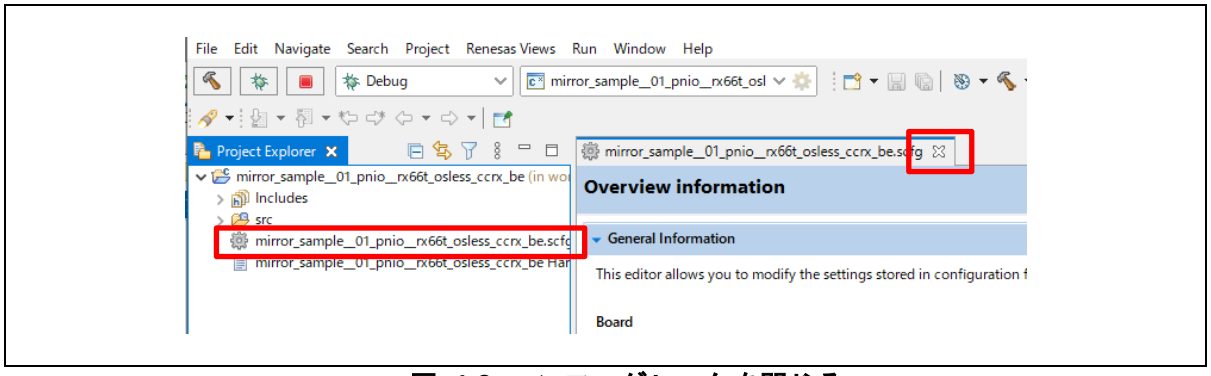

図 4-8 コンフィグレータを閉じる

③ 移植対象の R-IN32M3 Module サンプルプロジェクトをインポートします。
インポート手順は 3.3.3 プロジェクト立上げ を参照してください。

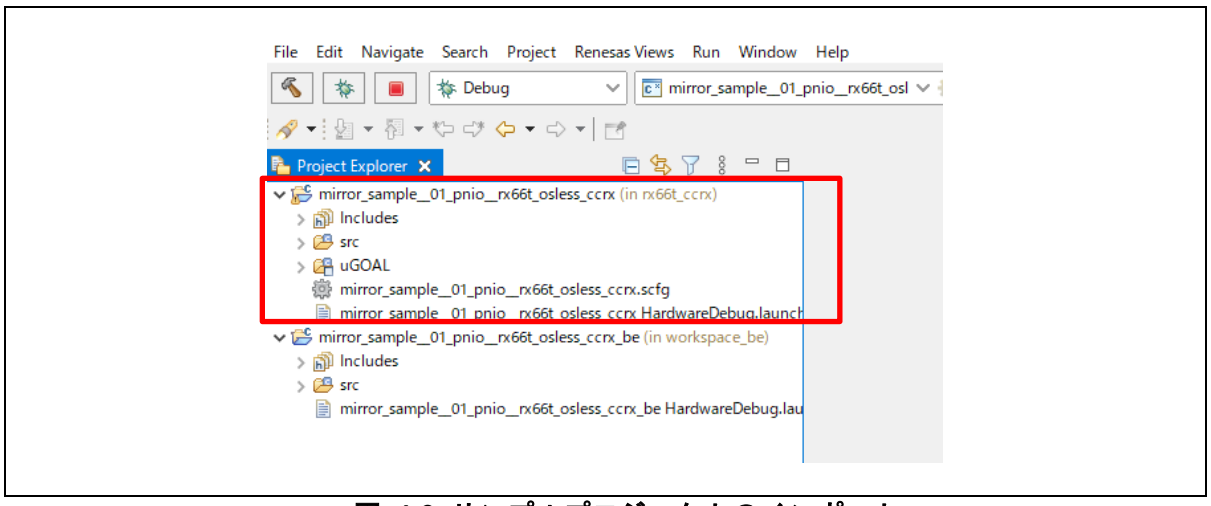

図 4-9 サンプルプロジェクトのインポート

④ 移植対象の R-IN32M3 Module サンプルプロジェクトから “uGOAL”フォルダ、
”コンフィグレータ” ファイル をビックエンディアン側のプロジェクトにコピーします。
⑤ コピーした ”コンフィグレータ” ファイル を移植元のファイルとの区別のため、
リネームします。

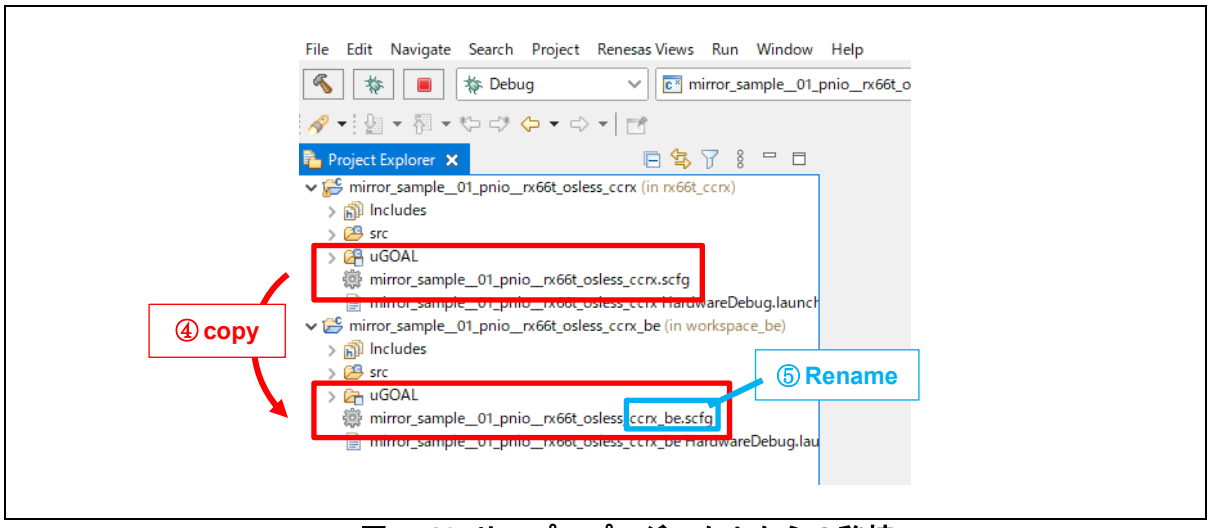

図 4-10 サンプルプロジェクトからの移植

2-2. プロジェクト設定

① “uGOAL” フォルダのプロパティを開き、Execute resource from build (/ビルドからリソースを除外)を無効にします。適用してプロパティを閉じます。

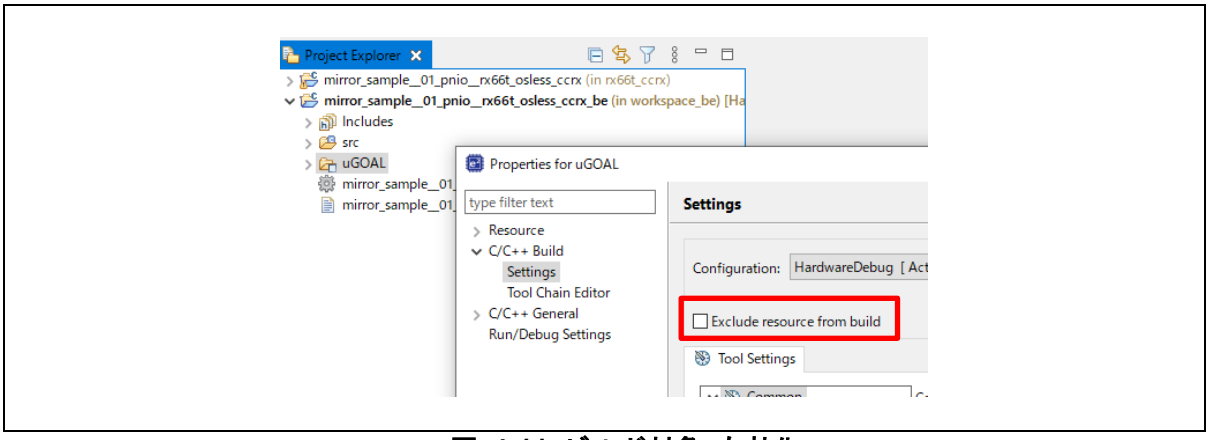

図 4-11 ビルド対象 有効化

② プロジェクト作成時に自動で生成された main 関数を含むファイル (mirror_sample__01_pnio__rx66t_osless_ccrx.c) のプロパティを開き、Execute resource from build (/ビルドからリソースを除外)を有効にします。適用してプロパティを閉じます。

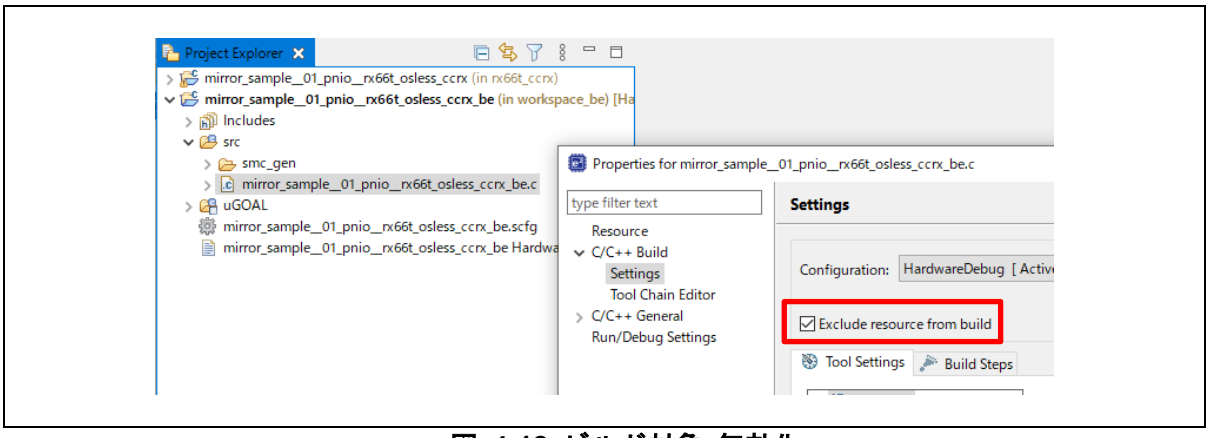

図 4-12 ビルド対象 無効化

2-3. ビルド設定

プロジェクトのプロパティを開きビルド環境を設定します。

① Build Variables (/ ビルド変数) 設定を移植元のプロジェクト設定からコピーし、適用させます。

変数名	GoalDirPath
型	String
値	${ProjDirPath}/../../../../..

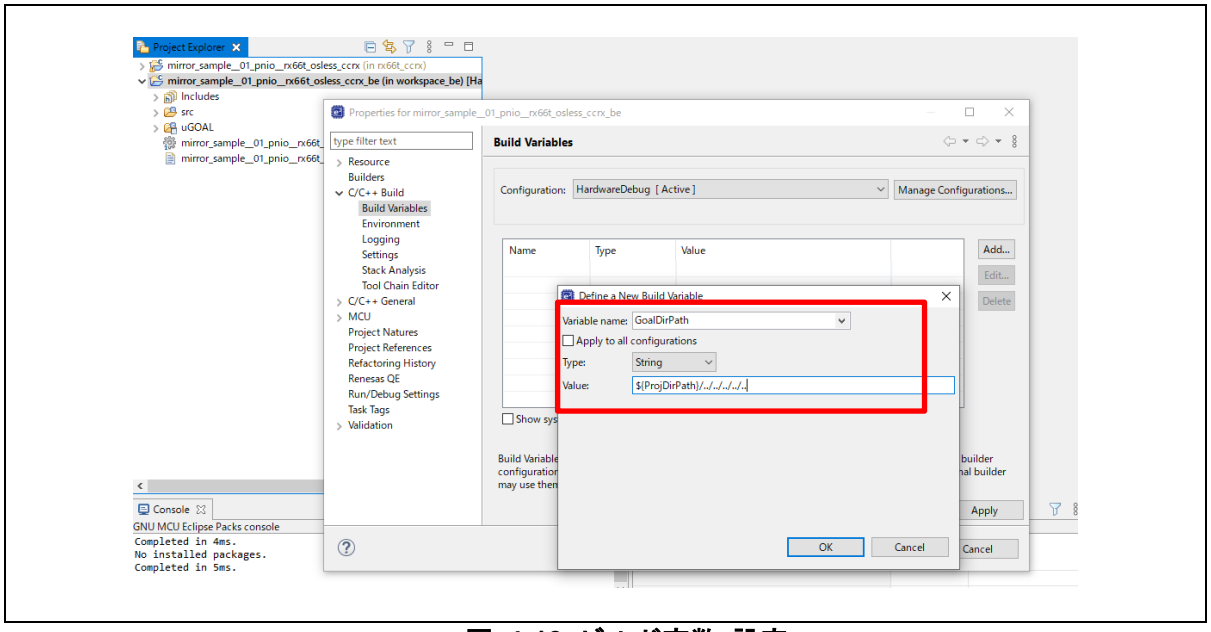

図 4-13 ビルド変数 設定

② インクルードファイルパス

[CCRX: **設定 > Compiler > Source**], [GCC: **設定 > Compiler > Includes**]

インクルードファイルパスを移植元のプロジェクト設定からコピーします。
コピー対象は先頭に ” ${GoalDirPath}” がついているパスです。

以下のインクルードファイル ディレクトリは “mirror_sample__01_pnio__rx66t_osless_ccrx” からのビックエンディアン移植時の例です。

"${GoalDirPath}/."	"${GoalDirPath}/rpc/wrapper"
"${GoalDirPath}/appl"	"${GoalDirPath}/rpc/wrapper/ccm"
"${GoalDirPath}/appl/mirror_sample/01_pnio"	"${GoalDirPath}/rpc/wrapper/dd"
"${GoalDirPath}/ext"	"${GoalDirPath}/rpc/wrapper/dd/protos"
"${GoalDirPath}/ext/uthash"	"${GoalDirPath}/rpc/wrapper/dd/protos/dd"
"${GoalDirPath}/plat/rx66t_semb1320"	"${GoalDirPath}/rpc/wrapper/dd/protos/dd/rpc"
"${GoalDirPath}/plat/rx66t_semb1320/Drivers/r_gpio_rx"	"${GoalDirPath}/rpc/wrapper/pnio"
"${GoalDirPath}/rpc"	"${GoalDirPath}/sapi"
"${GoalDirPath}/rpc/goal_mctc"	"${GoalDirPath}/ugoal"
"${GoalDirPath}/rpc/goal_media"	

③ マクロ定義

[CCRX: **設定 > Compiler > Source**], [GCC: **設定 > Compiler > Includes**]

ビックエンディアン対応マクロ定義の追加および、移植元のプロジェクト設定からマクロ定義をコピーします。

以下のマクロ定義は “mirror_sample__01_pnio__rx66t_osless_ccrx” からのビックエンディアン移植時の例です。

コピー	_RX66T_DEMO_
コピー	APPL_STANDALONE
コピー	CONFIG_UGOAL_HEAP_BUFFER_ALIGNMENT=4
追加	CONFIG_UGOAL_BIG_ENDIAN=1

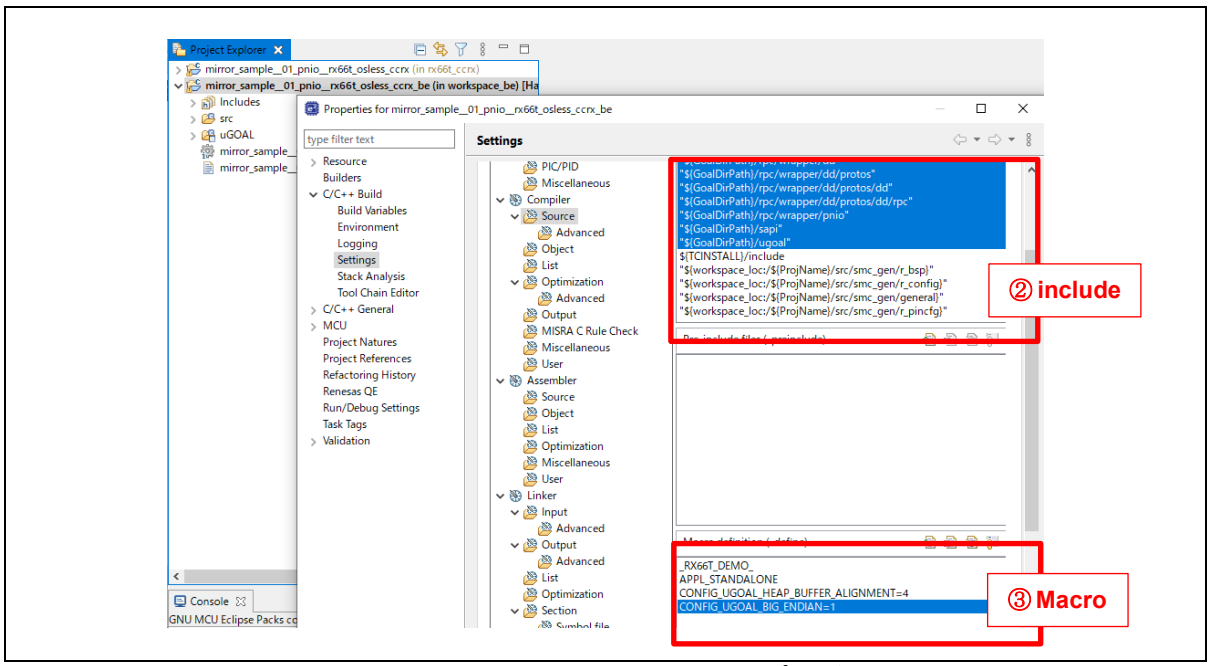

図 4-14 コンパイラ ソース 設定 (CCRX プロジェクトの場合)

④ オブジェクト設定

[CCRX: **設定 > Compiler > Source**]

下記 オブジェクト設定を有効にします。

初期値なし変数をアライメント数が 4 のセクションに配置する (-nostuff=B)
初期値あり変数をアライメント数が 4 のセクションに配置する (-nostuff=D)
const 修飾変数をアライメント数が 4 のセクションに配置する (-nostuff=C)

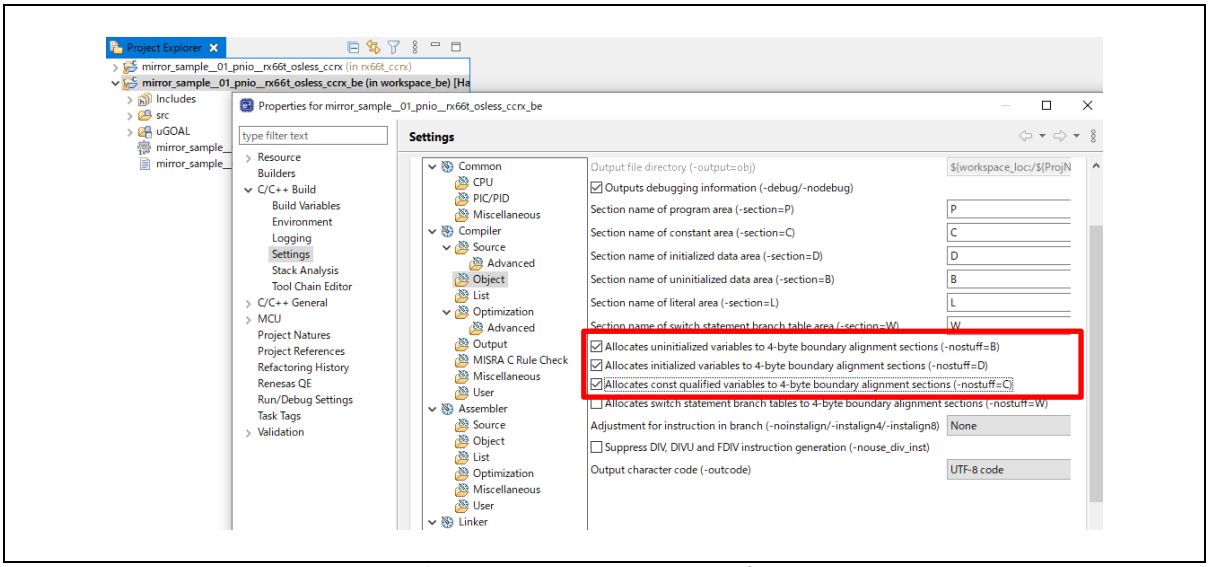

図 4-15 オブジェクト 設定 (CCRX プロジェクトの場合)

“適用して閉じる” でプロパティを閉じます。

以上でビックエンディアン対応したプロジェクトの作成は完了です。

以降は、3.3.4 FIT モジュール からの手順で開発環境構築を進めてください。

4.5　GCC ツールチェーンの個別インストール

必要なバージョンの GCC ツールチェーンが e2studio で表示されない場合は、以下の手順に従って GCC を個別に入手してください。ここでは GCC 8.3.0.202311 をダウンロードする手順を示します。

Open Source Tools https://llvm-gcc-renesas.com/ から、Products タブ → RX → Download Toolchains を選択し、ダウンロードページを表示します。

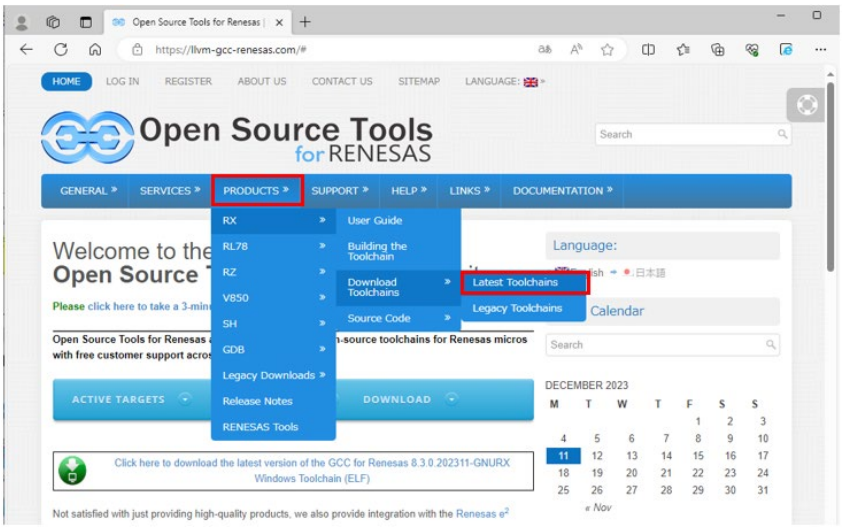

図 4-16 Open Source Tools ホームページ

GCC for Renesas 8.3.0.202311-GNURX Toolchain をダウンロードします。

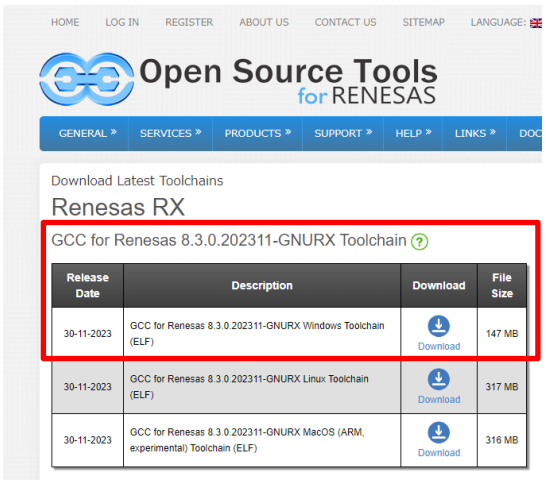

図 4-17 ダウンロードする GCC ツールチェーン

ダウンロードしたインストーラを実行し、GNURX-ELF Toolchain を含むコンポーネント一式をインストールしてください。インストール後、e2studio のツールチェーン管理ダイアログで該当コンパイラが追加されていることを確認します（3.3.3 プロジェクト立上げ を参照してください）。

4.6　FIT モジュールの個別インストールと適用方法

e2studio で RX Driver Package V1.42 を適用できない場合は、以下の手順に従って RX Driver Package を個別に入手し適用させてください。

Renesas ホームページ RX Driver Package https://www.renesas.com/jp/ja/software-tool/rx-driver-package にて入手可能です。当該バージョンが表示されていない場合は、ページ下部の [旧バージョン情報(Previous version)]へ進み、当該バージョンをダウンロードします。

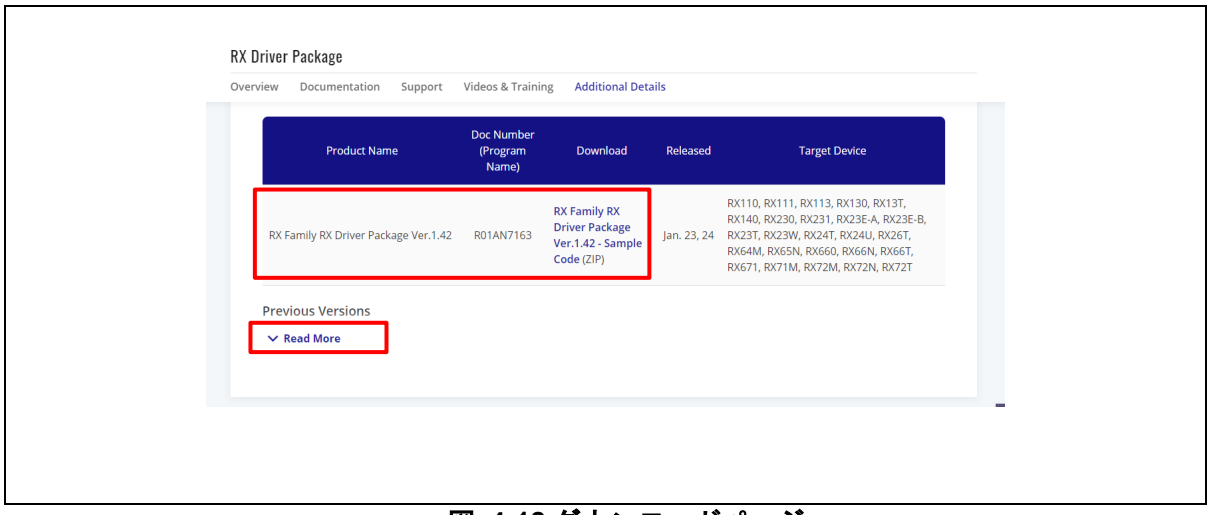

図 4-18 ダウンロードページ

ダウンロードしたパッケージ [r01an7163xx0142-rx-fit.zip] を、以下のフォルダに配置します。

C:¥Users¥**UserName**¥.eclipse¥com.renesas.platform_download¥FITModules¥Downloaded

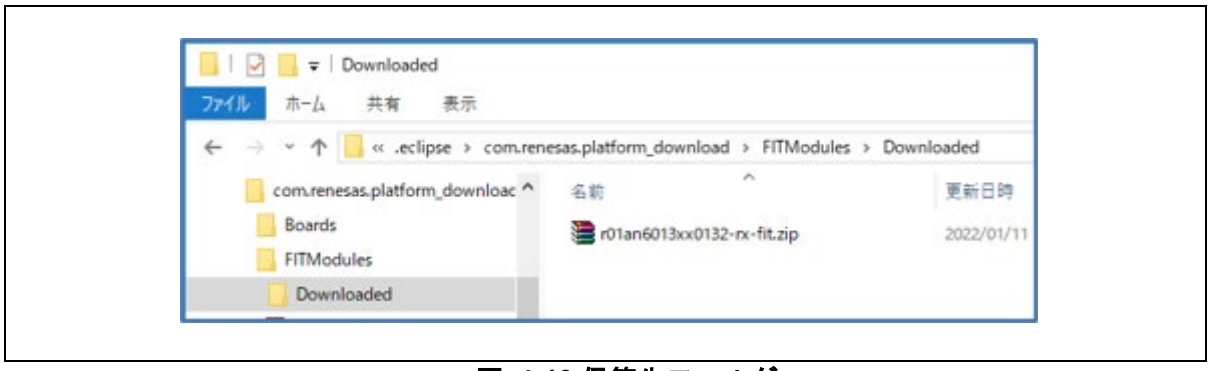

図 4-19 保管先フォルダ

同時に、ダウンロードした RX Driver Package Ver.1.42 の内容物の内、 [FITModules] のファイルに含まれるものを全て以下のフォルダに格納してください。

C:¥Users¥**UserName**¥.eclipse¥com.renesas.platform_download¥FITModules

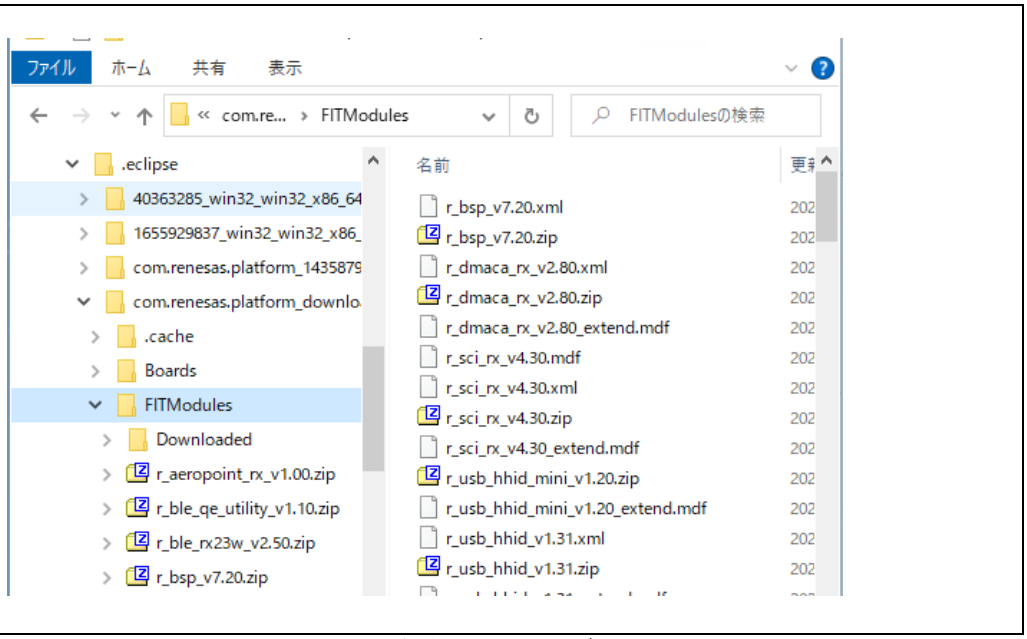

図 4-20 保管先フォルダ

データを配置できたら、e2studio を起動し FIT モジュールの活性化を行ってください（3.3.4 FIT モジュール を参照してください）。

4.7　RX ファミリ用 C コンパイラパッケージの個別インストール

必要なバージョンの RX コンパイラ（CC-RX）が e2studio で表示されない場合は、以下の手順に従って CC-RX を個別に入手してください。ここでは CC-RX V3.06.00 をダウンロードする手順を示します。

ルネサス web サイトの RX ファミリ用 C/C++コンパイラパッケージのページから、CC-RX V3.06.00 をダウンロードします。

図 4-21 CC-RX page

ダウンロードしたインストーラを実行し、コンポーネント一式をインストールしてください。インストール後、e2studio のツールチェーン管理ダイアログで該当コンパイラが追加されていることを確認します（3.3.3 プロジェクト立上げ を参照してください）。

改訂記録

Rev.	Date	Description	
		chapter	Summary
1.00	2021/10/15	—	新規作成
1.01	2022/1/11	3.4	Remote I/O サンプルアプリケーションに関する説明を追加
		3.4	Modbus TCP サンプルアプリケーションに関する説明を追加
		3.3.1	RX ファミリ用 C/C++コンパイラ(CC-RX)に関する説明を追加
1.02	2022/8/5	3.4.6	web サーバ機能サンプルに関する説明を追加
		4.4	Big-endian 対応手順の説明を追加
1.03	2023/5/31	3.4	サンプルプログラム更新に伴う説明見直し
1.04	2023/12/15	関連文書	接続ガイド TwinCAT を追記
		1.2	表 1-1 FIT モジュール等の指定を追記
		3.3.3	GCC の個別ダウンロード方法を追記
		3.3.4	FIT モジュールの個別ダウンロード方法を追記
		3.4.1	GSDML 図差し替え、説明を追記
		3.4.2	Mirror 制御(rpc)に関する説明を追加
		3.4.3	入出力アプリケーションに関する説明を追記
		3.4.6	図 3-69 差し替え
		4.5, 4.6	4.5 章, 4.6 章 を追加
1.05	2024/5/31	1.2	動作環境の各種バージョン更新
		3.3.1	図 3-2、図 3-5 差し替え
		3.3.3	図 3-7 差し替え、コンパイラバージョン更新
		3.3.4	FIT バージョン更新
		3.4.1	GSD ファイル名更新
		3.4.6	評価セットアップから Modbus TCP を削除
		4.5	図 4-17 差し替え、コンパイラバージョン更新
		4.6	図 4-18 差し替え、FIT バージョン更新
		4.7	4.7 章を追加

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。