

## RX72M グループ

R01AN5434JJ0110

Rev.1.10

## EtherCAT 通信を用いたシングルチップモータ制御

2020/8/31

### 要旨

本アプリケーションノートでは、永久磁石同期モータ(以降、PMSM と表記)のエンコーダベクトル制御機能と EtherCAT 通信機能を RX72M に実装したサンプルプログラムについて説明します。本モジュールは産業イーサネット通信用 EtherCAT スレーブコントローラ (EtherCAT Slave Controller : ESC) を内蔵した RX ファミリで Beckhoff 社製 EtherCAT スレーブスタックコード(Slave Stack Code : SSC)を使用するためのインタフェースを提供します。

本モジュールには SSC は含まれておりません。EtherCAT Technology Group(ETG 協会)より SSC ツールを入手の上、コードを生成してください。

以降、本モジュールを EtherCAT FIT モジュールと称します。

### 対象デバイス

- RX72M グループ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

## 目次

## 内容

1. 概要	4
1.1 本アプリケーションノートについて	4
1.2 動作環境	4
1.3 プロジェクトについて	5
2. システム概要	6
2.1 ハードウェア構成	6
2.2 ハードウェア仕様	7
2.3 ソフトウェア構成	10
2.3.1 ソフトウェア・ファイル構成	10
2.3.2 ソフトウェア・モジュール構成	13
2.4 ソフトウェア仕様	16
3. CiA402 ドライブプロファイル	18
3.1 動作モード	19
3.2 状態遷移	20
3.3 状態遷移関数	21
3.4 オブジェクトディクショナリ	23
4. モーション制御パラメータ	25
4.1 速度パラメータ	25
4.2 加速度パラメータ	25
4.3 RMW からの単位変換	25
5. API 関数	27
5.1 概要	27
5.2 R_MTR_InitControl	28
5.3 R_MTR_SetUserifMode	29
5.4 R_MTR_ExecEvent	30
5.5 R_MTR_ChargeCapacitor	31
5.6 R_MTR_GetLoopModeStatus	32
5.7 R_MTR_SetPositionStatus	33
5.8 R_MTR_SetPosition	34
5.9 R_MTR_GetPosition	35
5.10 R_MTR_GetPositioningFlag	36
5.11 R_MTR_SetSpeed	37
5.12 R_MTR_GetSpeed	38
5.13 R_MTR_SetDir	39
5.14 R_MTR_GetDir	40
5.15 R_MTR_GetStatus	41
5.16 R_MTR_InputBuffParamReset	42
5.17 R_MTR_CtrlInput	43
5.18 R_MTR_SetVariables	44

5.19	R_MTR_AutoSetVariables .....	45
5.20	R_MTR_CtrlGainCalc.....	46
5.21	R_MTR_UpdatePolling.....	47
5.22	R_MTR_GetErrorStatus .....	48
5.23	R_MTR_GetPositionPFStatus.....	49
5.24	R_MTR_SetPositionUnits.....	50
5.25	R_MTR_SetActualPositionUnits.....	51
5.26	R_MTR_GetPositionUnits .....	52
5.27	R_MTR_GetSpeedUnits.....	53
5.28	R_MTR_SetSpeedUnits .....	54
5.29	R_MTR_SetAccelerationUnits.....	55
5.30	R_MTR_SetDecelerationUnits .....	56
6.	ソリューションキットでの動作確認.....	57
6.1	動作環境 .....	57
6.2	動作環境の設定、接続.....	58
6.3	サンプルプログラムの構築 .....	62
6.4	サンプル・プロジェクトを e <sup>2</sup> studio にインポート.....	64
6.5	プログラミングとデバッグ .....	65
6.6	TwinCAT との接続 (ESI ファイルの書き込み).....	67
6.7	CODESYS との接続確認 .....	71
6.7.1	デバイスネットワークの設定.....	71
6.7.2	CODESYS の起動 .....	71
6.7.3	PLC の起動.....	72
6.7.4	スレーブデバイスの更新 .....	73
6.7.5	PLC との接続設定 .....	74
6.8	CODESYS を使った動作確認 .....	80
7.	参考ドキュメント .....	81
8.	APPENDIX .....	82

## 1. 概要

### 1.1 本アプリケーションノートについて

本アプリケーションノートでは、永久磁石同期モータ(以降、PMSM と表記)のエンコーダベクトル制御機能と EtherCAT 通信機能を RX72M に実装したサンプルプログラムについて説明します。

サンプルプログラムは、RX72M CPU カードと 24V 系インバータボードの組み合わせで動作します。

### 1.2 動作環境

表 1-1 動作環境

対応 MCU	RX72M グループ
評価ボード	ルネサスエレクトロニクス製 RX72M CPU カード+ 24V 系インバータボード※
統合開発環境 (IDE)	ルネサスエレクトロニクス製 e2 studio V.7.5.0 以降 IAR Embedded Workbench for Renesas RX 4.13.1 以降
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.01.00 以降
	GCC for Renesas RX 4.8.4.201803 以降
	IAR C/C++ Compiler for Renesas RX version 4.13.1 以降
モータ	Leadshine 製 インクリメンタルエンコーダ付き永久磁石同期モータ BLM57050-1000
エミュレータ	ルネサスエレクトロニクス製 e2 Lite
通信プロトコル	EtherCAT
SSC Tool	EtherCAT Technology Group (ETG) 提供 Slave Stack Code (SSC) Tool Version 5.12
ソフトウェア PLC	Beckhoff Automation 製 TwinCAT® 3 (Beckhoff web サイトからダウンロード)
	3S-Smart Software Solutions 製
	CODESYS

※ルネサスエレクトロニクス製「24V モータ制御システム」に同梱されています。

24V Motor Control Evaluation System for RX23T (RTK0EM0006S01212BJ)

### 1.3 プロジェクトについて

サンプルプログラムはモータ制御または EtherCAT 通信、個別のプロジェクトをベースに幾つかの変更点を加えることで 2 つの機能をマージした EtherCAT 通信によるシングルチップモータ制御を実現しています。

表 1-2 ベースプロジェクトと変更点

機能/プロジェクト名 (アプリケーションノート)	変更点
モータ制御 RX72M_MRSSK_SPM_ENCD_FOC_E2S_RV100 (r01an5386jj0100-rx72m-motor)	<ul style="list-style-type: none"> <li>● EtherCAT 通信プログラムからモータ制御を行うための API 関数の追加</li> <li>● CiA402 のオブジェクトの仕様に合わせた位置や速度の単位変換</li> </ul>
EtherCAT 通信 rx72m_com_cia402 (r01an4672jj0100-rx72m-ecat)	<ul style="list-style-type: none"> <li>● CiA402 ドライブプロファイルに合わせたオブジェクトの追加</li> <li>● モータ制御を行うための API 関数の呼び出し</li> </ul>

サンプルプログラムに含まれるプロジェクトを示します。

以降の章では RX72M CPU カード+ 24V 系インバータボードのプロジェクトを例にして説明します。異なるプロジェクトを使用する場合は適宜、プロジェクト名を置き換えて、お読みください。

表 1-3 プロジェクト一覧

MCU	評価ボード名	プロジェクト名
RX72M	RX72M CPUカード+ 24V系インバータボード	ecat_cia402_motor_rsskrx72m

2. システム概要

2.1 ハードウェア構成

サンプルプログラムのハードウェア構成を次に示します。

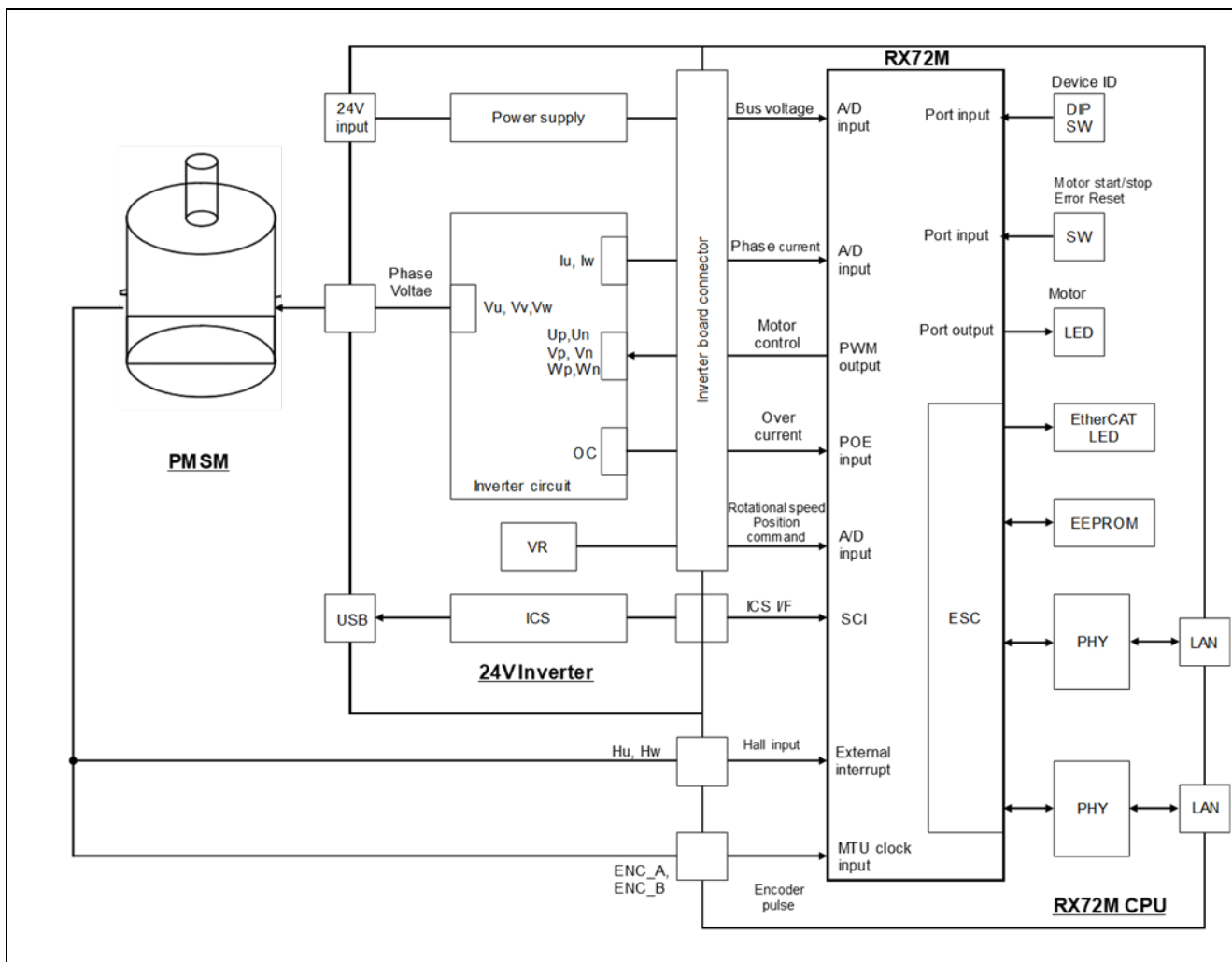


図 2-1 ハードウェア構成図

## 2.2 ハードウェア仕様

サンプルプログラムで使用する端子インタフェースを表 2-1 から表 2-4 に示します。

表 2-1 モータ制御関連端子インタフェース

端子名	機能
P43 / AN003	インバータ母線電圧測定
P47 / AN007	回転位置/速度指令値入力用(アナログ値)
P30	START/STOP トグルスイッチ
P02	ERROR RESET トグルスイッチ
P71	LD1 点灯/消灯制御
PN4	LD2 点灯/消灯制御
PH0	LD3 点灯/消灯制御
P40 / AN000	U 相電流測定
P42 / AN002	W 相電流測定
PE1 / MTIOC3B	PWM 出力(U <sub>p</sub> )
PE2 / MTIOC4A	PWM 出力(V <sub>p</sub> )
PE3 / MTIOC4B	PWM 出力(W <sub>p</sub> )
PE0 / MTIOC3D	PWM 出力(U <sub>n</sub> )
PE5 / MTIOC4C	PWM 出力(V <sub>n</sub> )
PE4 / MTIOC4D	PWM 出力(W <sub>n</sub> )
P31 / IRQ1	ホール U 相入力
PD3 / IRQ3	ホール V 相入力
PB0 / IRQ12	ホール W 相入力
PA4 / MTCLKA	エンコーダ A 相入力
P25 / MTCLKB	エンコーダ B 相入力
PC4 / POE0#	過電流検出時の PWM 緊急停止入力

表 2-2 EtherCAT 通信関連端子インタフェース(1)

端子名	機能
PK6/CATLINKACT0	Link/Activity LED 制御出力
PK7/CATLINKACT1	Link/Activity LED 制御出力
PH1/CATI2CCLK	EEPROM I2C クロック出力
P15/CATLEDRUN	RUN LED (緑色 LED)制御出力
PH3/CATLEDERR	ERR LED (赤色 LED)制御出力
PL3/CAT0_RX_CLK	受信クロック入力
PM4/CAT0_ETXD2	4 ビットの送信データ出力(bit2)
PM5/CAT0_ETXD3	4 ビットの送信データ出力(bit3)
PL4/CAT0_ETXD0	4 ビットの送信データ出力(bit0)
PL5/CAT0_ETXD1	4 ビットの送信データ出力(bit1)
PK5/CAT0_ERXD3	4 ビットの受信データ入力(bit3)
PK4/CAT0_ERXD2	4 ビットの受信データ入力(bit2)
P74/CAT0_ERXD1	4 ビットの受信データ入力(bit1)
P75/CAT0_ERXD0	4 ビットの受信データ入力(bit0)
PL7/CAT0_MDIO	マネジメントデータ I/O 入出力
PN3/CAT1_RX_ER	受信エラー入力
P84/CAT1_LINKSTA	PHY-LSI からのリンクステータス入力
PQ2/CAT1_RX_DV	受信データ有効入力
PL6/CAT0_TX_EN	送信イネーブル出力
PN2/CAT1_TX_CLK	送信クロック入力
PH4/CATLEDSTER	STATE LED (2色 LED)用の RUN LED 制御(ERR 時消灯)出力
PH5/CATLATCH0	LATCH 信号入力
PH6/CATLATCH1	LATCH 信号入力
P27/CATIRQ	IRQ 出力
PQ7/CAT1_TX_EN	送信イネーブル出力
PK2/CAT0_RX_DV	受信データ有効入力
PM1/CAT1_ERXD1	4 ビットの受信データ入力(bit1)
PM2/CAT1_ERXD2	4 ビットの受信データ入力(bit2)
PM3/CAT1_ERXD3	4 ビットの受信データ入力(bit3)
PL2/CAT0_RX_ER	受信エラー入力
PM0/CAT1_ERXD0	4 ビットの受信データ入力(bit0)
PQ4/CAT1_RX_CLK	受信クロック入力
PJ5/CATSYNC0	SYNC0 信号出力
PA6/CATRESTOUT	PHY リセット信号出力



表 2-3 EtherCAT 通信関連端子インタフェース(2)

端子名	機能
PN1/CAT1_ETXD3	4 ビットの送信データ出力(bit3)
PQ5/CAT1_ETXD0	4 ビットの送信データ出力(bit0)
PN0/CAT1_ETXD2	4 ビットの送信データ出力(bit2)
PQ6/CAT1_ETXD1	4 ビットの送信データ出力(bit1)
P11/CATSYNC1	SYNC1 信号出力
PM6/CAT0_TX_CLK	送信クロック入力
PK0/CAT0_MDC	マネジメントデータクロック出力
P34/CAT0_LINKSTA	PHY-LSI からのリンクステータス入力
P82/CATI2CDATA	EEPROM I2C データ入出力

表 2-4 その他端子インタフェース

端子名	機能
P12/RXD2	SCI2 の受信データ入力端子
P13/TXD2	SCI2 の送信データ出力端子
PH2	デバイス ID DIP SW(bit0)
PQ3	デバイス ID DIP SW(bit1)
P05	デバイス ID DIP SW(bit2)
P72	デバイス ID DIP SW(bit3)
PC1	デバイス ID DIP SW(bit4)
PN5	デバイス ID DIP SW(bit5)

## 2.3 ソフトウェア構成

### 2.3.1 ソフトウェア・ファイル構成

サンプルプログラムのフォルダとファイル構成を表 2-5～表 2-8 に示します。

網掛けされたファイルはサンプルプログラムの機能を実現するためにベースプロジェクトから変更があったことを示し、太字はファイルが追加されたことを示します。

表 2-5 モータ制御プログラムファイル構成(1)

motor ディレクトリ		ファイル	内容	
application/	main/	main.h, main.c	メイン関数	
	user_interface/	ics/	r_mtr_ics.h, r_mtr_ics.c	ICS 関連関数定義
			ICS_RX72M.h ICS_RX72M.lib	ツール用通信関連定義 ツール用通信ライブラリ
		board/	r_mtr_board.h, r_mtr_board.h	board ユーザ関数定義
middle/	interface/	r_mtr_driver_access.h r_mtr_driver_access.c	ユーザアクセス関数定義	
		<b>r_mtr_driver_ecat_access.h</b> <b>r_mtr_driver_ecat_access.c</b>	EtherCAT 通信プログラム アクセス関数定義	
	common/	r_mtr_common.h	共通定義	
		<b>r_mtr_units.h</b>	単位系定義	
		r_mtr_filter.h, r_mtr_filter.c	汎用フィルタ関数定義	
		r_mtr_fluxwkn.h r_mtr_fluxwkn.obj	弱め磁束制御関連関数定義	
		r_mtr_pi_control.h r_mtr_pi_control.c	PI 制御関数定義	
		r_mtr_transform.h r_mtr_transform.c	座標変換関数定義	
		r_mtr_mod.h, r_mtr_mod.c	変調関数定義	
		r_mtr_volt_err_comp.h r_mtr_volt_err_comp.obj	電圧誤差補償関数定義	
		r_mtr_statemachine.h r_mtr_statemachine.c	ステートマシン関数定義	
		control/	r_mtr_parameter.h	各種パラメータ定義
			r_mtr_ctrl_gain_calc.obj	制御ゲイン算出関数定義
			r_mtr_foc_action.c	アクション関数定義
			r_mtr_interrupt_carrier.c	キャリア割込み関数定義
	r_mtr_interrupt_timer.c		周期割込み関数定義	
	r_mtr_interrupt_sensor.c		センサ入力割込み関数定義	
	r_mtr_foc_control_encd_position.h r_mtr_foc_control_encd_position.c		FOC 関数定義	

表 2-6 モータ制御プログラムファイル構成(2)

motor ディレクトリ		ファイル	内容
middle/	control/	r_mtr_foc_current.h r_mtr_foc_current.c	電流制御関数定義
		r_mtr_foc_speed.h r_mtr_foc_speed.c	速度制御関数定義
		r_mtr_foc_position.h r_mtr_foc_position.c	位置制御関数定義
		r_mtr_position_profiling.h r_mtr_position_profiling.c	位置指令値作成関数定義
		r_mtr_ipd.h r_mtr_ipd.obj	IPD 制御関数定義
		r_mtr_speed_observer.h r_mtr_speed_observer.obj	速度オブザーバ関数定義
driver/	inverter/	r_mtr_ctrl_mrsk.h r_mtr_ctrl_mrsk.c	インバータボード依存関数定義
	mcu/	r_mtr_interrupt.c	割り込み関数定義
		r_mtr_ctrl_rx72m.h r_mtr_ctrl_rx72m.c	MCU 固有関数定義
		r_mtr_ctrl_mcu.h	MCU 共通定義
	sensor/	r_mtr_ctrl_encoder.h r_mtr_ctrl_encoder.c	エンコーダ関数定義
		r_mtr_ctrl_hall.h r_mtr_ctrl_hall.c	ホール関数定義
	config/		r_mtr_config.h
		r_mtr_motor_parameter.h	モータパラメータコンフィグレーション定義
		r_mtr_inverter_parameter.h	インバータパラメータコンフィグレーション定義
		r_mtr_control_parameter.h	制御パラメータコンフィグレーション定義
		r_mtr_encoder_parameter.h	エンコーダパラメータコンフィグレーション定義

表 2-7 EtherCAT 通信プログラムファイル構成(1)

smc_gen/r_ecat_rx ディレクトリ		ファイル	内容	
./		r_ecat_rx_if.h	FIT モジュール API 定義	
		readme.txt	FIT モジュール添付文書	
./doc/	en/	r01an4881ejxxxx-rx-ecat.pdf	アプリケーションノート (英語版)	
	ja/	r01an4881jjxxxx-rx-ecat.pdf	アプリケーションノート (日本語版)	
./ref/		r_ecat_rx_config_reference.h	FIT モジュールデフォルト オプション定義	
./src/	hal/		renesashw.h renesashw.c	ESC アクセス関数定義
	phy/		phy.h, phy.c	PHY 制御関数定義
	targets/	rx72m/	r_ecat_setting_rx72m.c	MCU 固有関数定義
./utilities/	rx72m/	batch_files/	apply_patch.bat	SSC ソースファイル修正 用バッチファイル
			RX72M_Motor_YYMMDD.patch	シングルチップモータ制御 向け修正パッチ
	esi/		RX72M EtherCAT MotorSolution.xml	ESI ファイル
	ssc_config/		Renesas_RX72M_config.xml	SSC コンフィグレーション ファイル
			RX72M EtherCAT CiA402.esp	SSC ツールプロジェクト ファイル

表 2-8 EtherCAT 通信プログラムファイル構成(2)

application ディレクトリ	ファイル	内容
./ecat/	cia402sample.h cia402sample.c	CiA402 アプリケーション 定義
	SSC ソースファイル	修正用バッチファイルを適 用すると格納される

### 2.3.2 ソフトウェア・モジュール構成

モータ制御プログラムのモジュール構成を図 2-2 に示します。

モータ制御のベースプロジェクトに対して EtherCAT 通信による制御を行うために追加、または変更したファイルを赤枠で囲んでいます。

なお、赤枠実線はファイルの追加を、赤枠点線はファイルの変更を表します。

モジュール階層別に主な変更点を示します。

表 2-9 モジュール階層別変更点

階層 / モジュール	関連ファイル	ファイルの説明
Application Layer / EtherCAT Application	cia402sample.c	EtherCAT 通信プログラムのアプリケーション層。EtherCAT マスタからの指令をモータ制御プログラムに、モータ制御プログラムからのステータスを EtherCAT マスタに受け渡す役割を担う。
Middle Layer / EtherCAT Interface module	r_ecat_dirver_acces.c	モータ制御プログラムと EtherCAT 通信プログラムのインタフェース用 API 関数。
Middle Layer / Control Module	r_mtr_foc_control_encd_position.c r_mtr_foc_speed.c r_mtr_foc_position.c	CiA402 で使用する速度[count/s]や位置[count]の単位をモータ制御プログラムで使用する速度[rad/s]や位置[rad]の単位に変換する。

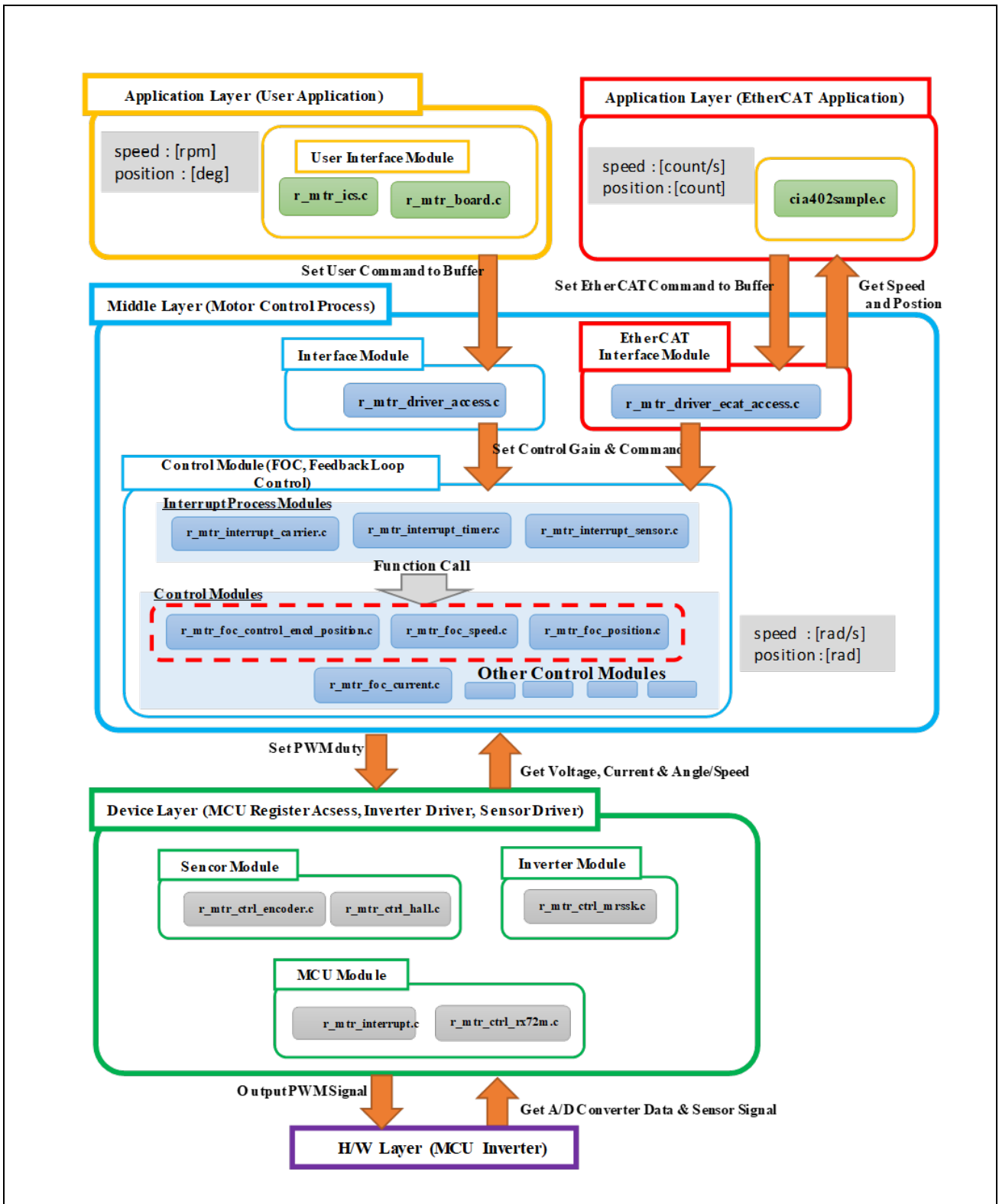


図 2-2 モータ制御プログラムモジュール構成

EtherCAT 通信プログラムのモジュール構成を図 2-3 に示します。

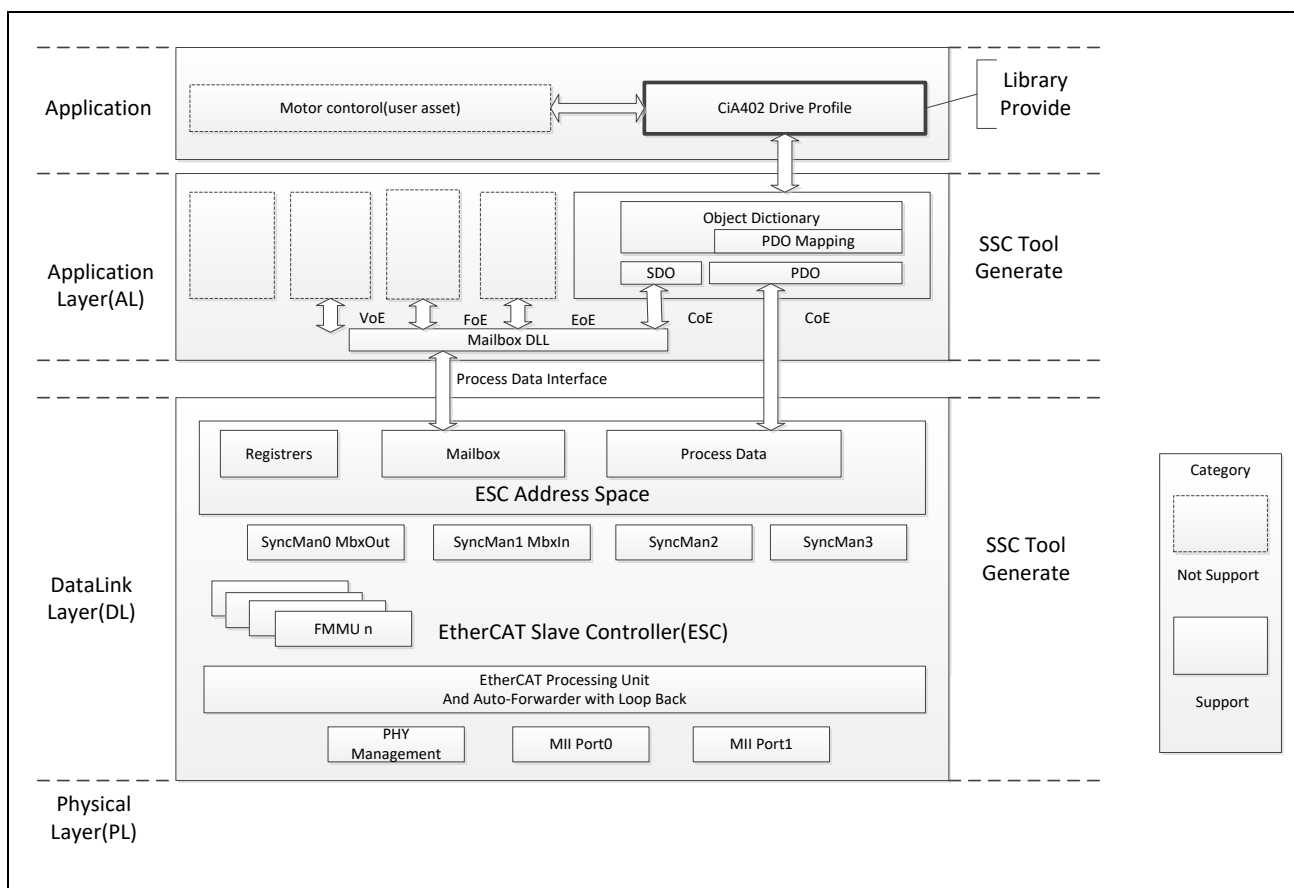


図 2-3 EtherCAT 通信プログラムモジュール構成

## 2.4 ソフトウェア仕様

サンプルプログラムのソフトウェアの基本仕様を下記に示します。

表 2-10 モータ制御プログラム基本仕様

項目	内容	
制御方式	ベクトル制御	
回転子磁極位置検出	インクリメンタルエンコーダ(A相、B相)、ホールセンサ(UVW相)	
入力電圧	DC 24V	
キャリア周波数(PWM)	20 [kHz](キャリア周期 : 50 [μs])	
デッドタイム	2 [μs]	
制御周期(電流)	50 [μs]	
制御周期(速度・位置)	500 [μs]	
位置指令値範囲	board UI	-180° ~ 180°
	ICS UI	-32768° ~ 32767°
	ETH UI	-2 147 483 648[count] ~ 2 147 483 647[count] <sup>注2</sup>
速度指令範囲	CW : 0[rpm] ~ 2000[rpm] CCW : 0[rpm] ~ 2000[rpm]	
位置分解能	0.3° (エンコーダパルス : 1000[ppr]、4通倍時4000[cpr])	
位置の不感帯 <sup>注1</sup>	エンコーダ±1カウント (±0.09°)	
各制御系固有周波数	電流制御系 : 300Hz 速度制御系 : 30Hz 位置制御系 : 10Hz	
保護停止処理	以下のいずれかの条件の時、モータ制御信号出力(6本)を非アクティブにする 各相の電流が3.82 [A]を超過(50 [μs]毎に監視) インバータ母線電圧が28 [V]を超過(50 [μs]毎に監視) インバータ母線電圧が 14[V]未満(50 [μs]毎に監視) 回転速度が3000 [rpm]を超過(50 [μs]毎に監視)  外部からの過電流検出信号(POE0#端子に立ち下りエッジを検出)及び出力短絡を検出した場合、PWM出力端子をハイインピーダンスにする	

【注1】位置決め時のハンチング等を防ぐため、不感帯を設けています。

【注2】[unit]はエンコーダカウント数です。



表 2-11 EtherCAT 通信プログラムの基本仕様

項目	内容
物理層	100BASE-TX (IEEE802.3)
ボーレート	100[Mbps] (Full duplex)
通信ポート数	2 ポート
EtherCAT LED	RUN、ERR、STAT、L/A IN、L/A OUT
ステーション ID	デバイス ID DIP SW (6bit)により設定
Explicit Device ID	対応
デバイスプロファイル	CiA402 デバイスプロファイル
Sync Manager	4
FMMU	3
通信オブジェクト	SDO (Service Data Object) PDO (Process Data Object)
同期モード	SM2 イベント同期モード DC モード
プロトコルスタックの提供形態	サンプルプログラム向けの SSC Tool プロジェクト ファイルを提供。また CiA402 アプリケーションな どの追加分のパッチを提供。SSC Tool にてプロト コルスタックコードを生成後、パッチを適用する ことで EtherCAT 通信プログラムが出来る。

### 3. CiA402 ドライブプロファイル

CiA402 ドライブプロファイルはドライブおよびモーションコントロール用のデバイスプロファイルであり、主にサーボドライブ、正弦波インバータ、およびステッピングモータ用コントローラの機能動作を定義します。このプロファイルでは、複数の動作モードと対応する設定パラメータがオブジェクトディクショナリとして規定されます。また、状態ごとの内部および外部動作を規定する有限状態オートマトン (Finite State Automaton: FSA) も含まれます。状態を変更する場合はコントロールワードオブジェクトを通じて指定することで、現在の状態を示すステータスワードオブジェクトに遷移後の結果が反映されます。コントロールワードと各種コマンド値 (速度など) は RxPDO に割り当てられ、ステータスワードと各種実査値 (位置など) は TxPDO に割り当てられます。詳細については CiA402 規格書の内容を確認してください (リファレンス(1))。

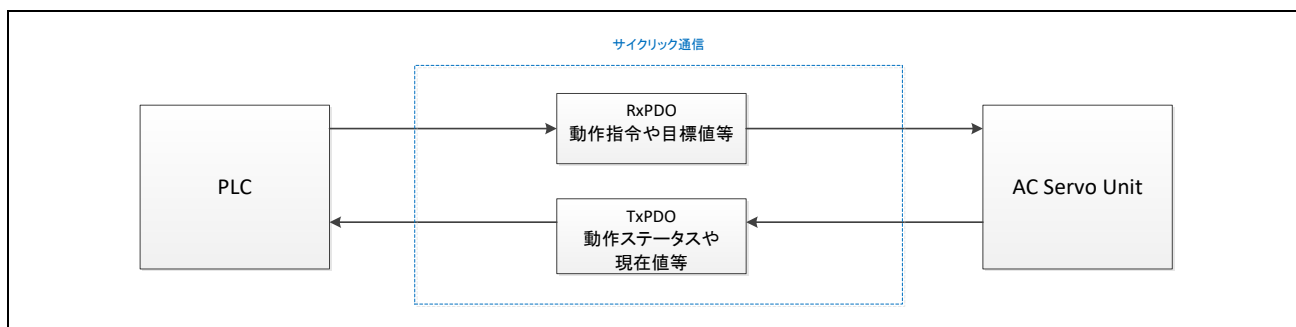


図 3-1 CiA402 通信の流れ

### 3.1 動作モード

CiA402 として規定されている動作モードのうち、サンプルプログラムでは以下をサポートします。

表 3-1 サポート動作モード一覧

Operation Mode	Support
Profile position mode	○
Velocity mode (frequency converter)	×
Profile velocity mode	×
Profile torque mode	×
Homing mode	○
Interpolated position mode	×
Cyclic synchronous position mode	○
Cyclic synchronous velocity mode	○
Cyclic synchronous torque mode	×
Cyclic synchronous torque mode with commutation angle	×
Manufacturer specific mode	×

### 3.2 状態遷移

CiA402 として規定されている FSA として、サンプルプログラムでは以下をサポートします。

図 3-2 のうちモータにトルクがかかっている状態は"Operation enabled"になります。"Switched on"から"Operation enabled"に遷移する契機(遷移 4)でモータをアクティブにします。また、"Operation enabled"から他の状態に遷移する契機(遷移 5、8、9)ではモータを非アクティブにします。ただし、"Operation enabled"から"Quick stop active"に遷移する契機(遷移 11)、または"Fault reaction active"に遷移する契機(遷移 13)ではトルクがかかっている状態を維持します。

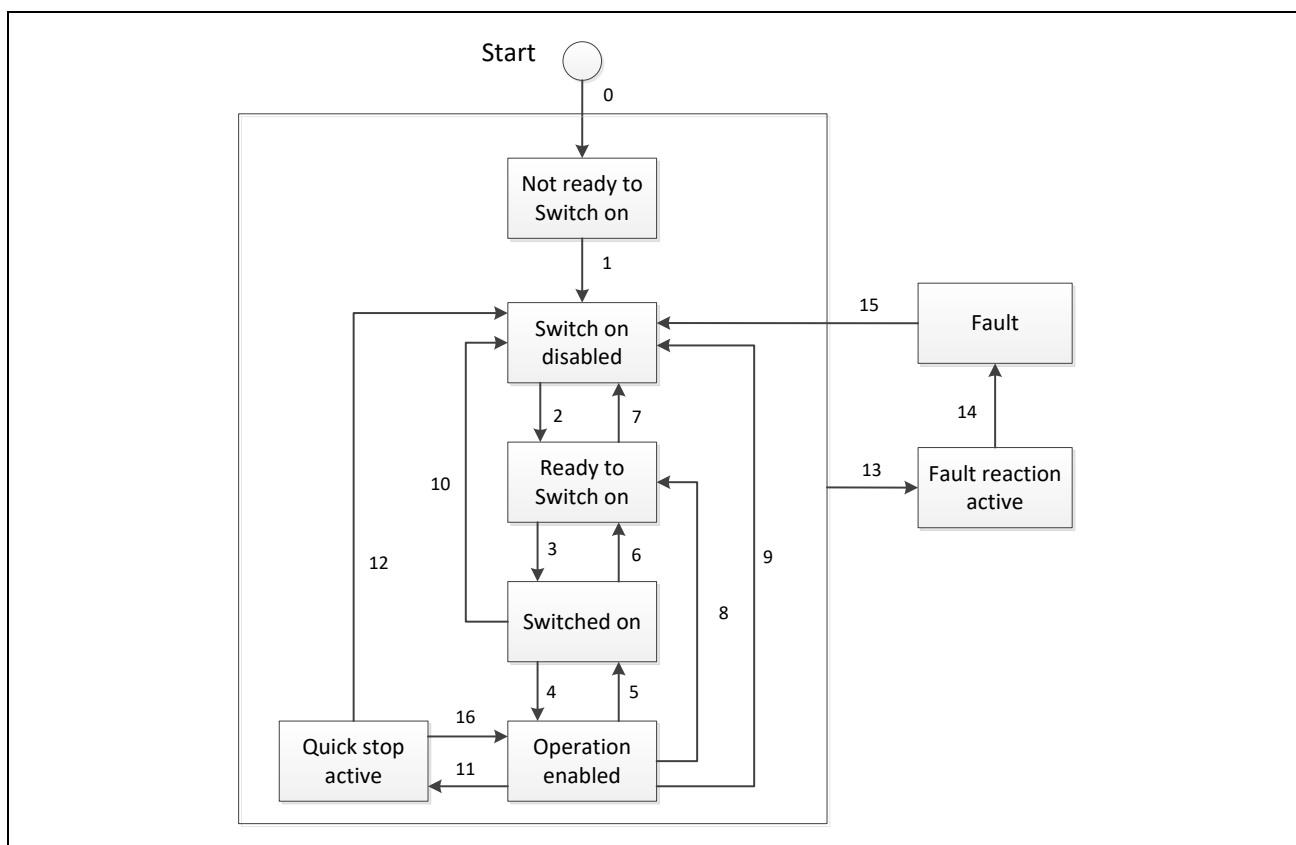


図 3-2 CiA402 状態遷移図

### 3.3 状態遷移関数

CiA402 状態遷移関数一覧を表 3-2 に示します。各関数は図 3-2 に示す CiA402 FSA の各状態遷移の番号とリンクしており、状態遷移が発生した際に対応する関数が呼び出されます。

表 3-2 CiA402 状態遷移関数一覧

Transition No.	function name
1	CiA402_StateTransition1
2	CiA402_StateTransition2
3	CiA402_StateTransition3
4	CiA402_StateTransition4
5	CiA402_StateTransition5
6	CiA402_StateTransition6
7	CiA402_StateTransition7
8	CiA402_StateTransition8
9	CiA402_StateTransition9
10	CiA402_StateTransition10
11	CiA402_StateTransition11
12	CiA402_StateTransition12
13	CiA402_LocalError
14	CiA402_StateTransition14
15	CiA402_StateTransition15
16	CiA402_StateTransition16

CiA402 状態遷移関数の関数仕様を示します。

---

### CiA402\_StateTransition(N)

---

CiA402FSA にて規定されている状態遷移(N)=1~12、14~16 が発生した際に使用します。  
状態遷移が発生した際に実行する処理を記述してください。

#### Format

UINT16 CiA402\_StateTransition(N)(TCiA402Axis \*pCiA402Axis)

#### Parameters

TCiA402Axis \*pCiA402Axis

#### Return Values

0 : 正常終了  
1 : 状態遷移しない

#### Properties

cia402appl.h にプロトタイプ宣言されています。

#### Description

処理中に異常が発生した場合は CiA402 規格に従って、各オブジェクトに適切な値を設定して関数を終了してください。戻り値に 1 を設定した場合には状態遷移は行われません。

#### Example

```
TCiA402Axis *pCiA402Axis;  
UINT16 retval ;  
  
/* Transition1 */  
retval = CiA402_StateTransition1 (pCiA402Axis);
```

---

### CiA402\_LocalError

---

CiA402 ドライブプロファイルで規定されたエラーを検出したときに使用します。本関数を実行後に CiA402FSA にて規定されている状態遷移 13 が発生します。  
エラーを検出した際に実行する処理を記述してください。

#### Format

void CiA402\_LocalError(UINT16 ErrorCode)

#### Parameters

UINT16 ErrorCode : CiA402 ドライブプロファイルエラーコード

#### Return Values

なし

#### Properties

cia402appl.h にプロトタイプ宣言されています。

#### Description

引数として指定したエラーコードはオブジェクト 0x603F に格納され EtherCAT マスタに通知されます。

#### Example

```
/* Over speed error is detected */  
CiA402_LocalError (ERROR_SPEED);
```

### 3.4 オブジェクトディクショナリ

サンプルプログラムでサポートしているオブジェクトディクショナリの一覧を以下に示します。

表 3-3 サポートオブジェクトディクショナリ一覧

INDEX	OBJECT Name	Category	Access	DataType	PDO Mapping
0x603F	Error code	Optional	ro	UINT16	TxPDO
0x6040	Controlword	Mandatory(all)	rw	UINT16	RxPDO
0x6041	Statusword	Mandatory(all)	ro	UINT16	TxPDO
0x605A	Quick stop option code	Optional	rw	INT16	No
0x605B	Shutdown option code	Optional	rw	INT16	No
0x605C	Disable operation option code	Optional	rw	INT16	No
0x605E	Fault reaction option code	Optional	rw	INT16	No
0x6060	Modes of operation	Optional	rw	INT8	No
0x6061	Modes of operation display	Optional	ro	INT8	TxPDO
0x6064	Position actual value	Mandatory(pp,csp,csv)	ro	INT32	TxPDO
0x6065	Following error window	Optional	rw	UINT32	No
0x6066	Following error time out	Optional	rw	UIN16	No
0x606C	Velocity actual value	Mandatory(csv)	ro	INT32	TxPDO
0x6077	Torque actual value	Optional	ro	INT32	No
0x607A	Target position	Mandatory(pp,csp)	rw	INT32	RxPDO
0x607B	Position rage limit	Optional	rw	INT32	No
0x607C	Home Offset	Optional	rw	INT32	RxPDO
0x607D	Software position limit	Optional	c,rw	INT32	No
0x607F	Max profile velocity	Optional	rw	UINT32	No
0x6080	Max motor speed	Optional	rw	UINT32	No
0x6081	Profile velocity	Mandatory(pp)	rw	UINT32	RxPDO
0x6083	Profile acceleration	Mandatory(pp)	rw	UINT32	RxPDO
0x6084	Profile deceleration	Optional	rw	UINT32	RxPDO
0x6085	Quick stop deceleration	Optional	rw	UINT32	No
0x6098	Homing method	Mandatory(hm)	rw	INT8	RxPDO
0x6099	Homing speeds	Mandatory(hm)	rw	UINT32	RxPDO
0x609A	Homing acceleration	Optional	rw	UINT32	RxPDO
0x60B0	Position offset	Optional	rw	INT32	No
0x60B1	Velocity offset	Optional	rw	INT32	No
0x60B2	Torque offset	Optional	rw	INT16	No
0x60C2	Interpolation time period	Mandatory(csp,csv)	rw	Record	No
0x60F4	Following error actual value	Optional	ro	INT32	No

---

0x60FF	Target velocity	Mandatory(csv)	rw	INT32	RxPDO
0x6402	Motor type	Optional	rw	UINT16	No
0x6502	Supported drive modes	Mandatory(all)	ro	UINT32	No



## 4. モーション制御パラメータ

CiA402 のオブジェクトに設定するモーション制御パラメータの設定値のサンプルプログラムにおける定義について説明します。

### 4.1 速度パラメータ

サンプルプログラムでは速度パラメータの単位は「制御周期ごとのエンコーダカウント」と定義しています。制御周期の周期は非常に短いため、速度値は  $2^{16}=65536$  を乗算した 16.16 ビット固定小数点の数値として処理されます。

1 秒あたりのエンコーダカウントから制御パラメータへの単位の変換では、数値に制御周期のタイムスライスを乗算した後、さらに 65536 を乗算します。

サンプルプログラムでは制御周期が 500us で動作するので、たとえば 1 秒あたり 5000 のエンコーダカウントの速度は、

$$5000 \times 0.0005 \times 65536 = 163840$$

に変換されます。

1 秒あたり 5000 のエンコーダカウントに相当する速度値は、163840 となります。

### 4.2 加速度パラメータ

サンプルプログラムでは加速度パラメータの単位は「制御周期の二乗ごとのエンコーダカウント」と定義しています。加速度と減速度の値は速度と同じく  $2^{16}=65536$  を乗算した 16.16 ビット固定小数点の数値として処理されます。

1 秒あたりのエンコーダカウントから制御パラメータへの単位の変換では、数値に位置周期のタイムスライスの二乗を乗算した後、さらに 65536 を乗算します。

制御周期が 500us で動作するので、たとえば 1 秒あたり 5000 のエンコーダカウントの加速度は、

$$5000 \times 0.0005 \times 0.0005 \times 65536 = 81$$

に変換されます。

1 秒あたり 5000 のエンコーダカウントに相当する加速度は、81 となります。

### 4.3 RMW からの単位変換

モータ制御開発支援ツール「Renesas Motor Workbench」でチューニングしたゲイン等の制御パラメータはモータ制御プログラムのソースファイルに反映することで、EtherCAT による制御の際も同じ値が使用されます。

一方、RMW で使用した位置や速度の指令値は CiA402 のオブジェクトに設定するモーション制御パラメータとは単位が異なるため、変換が必要となります。

RMW で使用した指令値を CiA402 オブジェクト指令値に変換する変換式を示します。

表 4-1 RMW→CiA402 指令値変換式

項目	RMW 指令値	CiA402 指令値	変換式	データ型
位置	$\Theta$ [deg]	$P_c$ [count]	$P_c = \Theta \div 360 \times \text{CPR}$	INT32
速度	$R$ [rpm]	$S_c$ [count/s]	$S_c = R \div 60 \times \text{CPR} \times T_s \times K_0$	INT32 (16.16)
加速度	$t_{\text{Acc}}$ [s] <sup>*1</sup>	$A_c$ [count/s <sup>2</sup> ]	$A_c = R \div t_{\text{Acc}} \div 60 \times \text{CPR} \times T_s \times T_s \times K_0$	UINT32 (16.16)

ただし、式中の記号は表 4-2 となります。

表 4-2 変換式記号

記号	意味	値
CPR	1 回転あたりのエンコーダカウント数[count]	4000
T <sub>s</sub>	制御周期[s]	0.0005
K <sub>Q</sub>	16.16 固定小数点変換用係数	2 <sup>16</sup> =65536

※1. RMW における加速度について

RMW では加速度を定義するために、目標速度に到達するまでの加速時間[s]を使用します。

図 4-1 は目標速度 speed に到達するまでの加速時間を t<sub>1</sub>→t<sub>2</sub> に変更することにより、加速度は acc<sub>1</sub>→acc<sub>2</sub> に変化することを示しています。

また、そのときの加速度は速度÷加速時間の式で表すことができます。

$$\text{acc1} = \text{speed} \div t_1$$

$$\text{acc2} = \text{speed} \div t_2$$

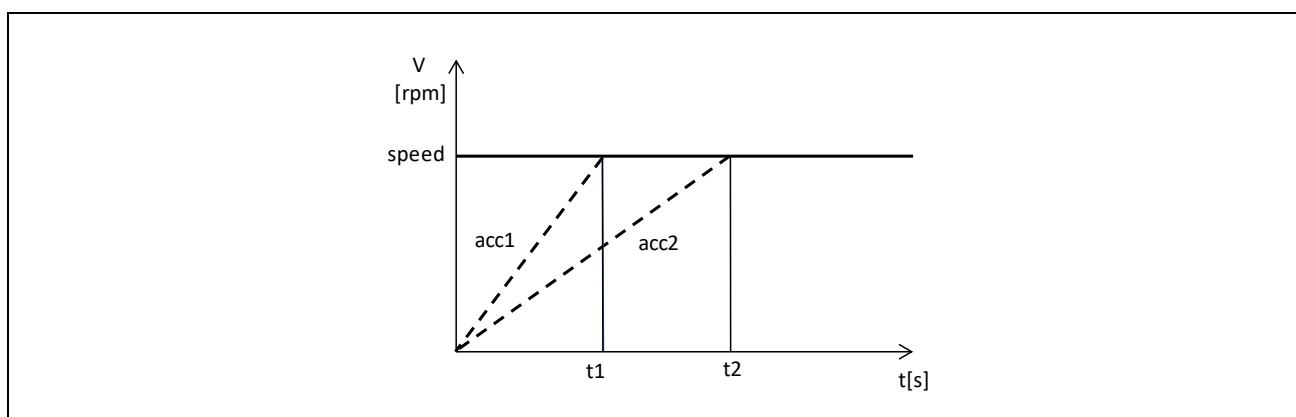


図 4-1 加速時間と加速度

#### 変換例

- ・ 位置指令値

Θ=180[deg]を P<sub>c</sub>[count]に変換する

$$180 \div 360 \times 4000 = 2000$$

- ・ 速度指令値

R=2000[rpm]を S<sub>c</sub>[count/s]に変換する

$$2000 \div 60 \times 4000 \times 0.0005 \times 65536 = 4369066$$

- ・ 加速度指令値

R=2000[rpm]のとき T<sub>ACC</sub>=0.3[s]を A<sub>c</sub>[count/s<sup>2</sup>]に変換する

$$2000 \div 0.3 \div 60 \times 4000 \times 0.0005 \times 0.0005 \times 65526 = 7281$$

## 5. API 関数

## 5.1 概要

モータ制御プログラムインタフェース用 API 関数を示します。

関数	説明
R_MTR_InitControl	モータ制御プログラムの初期設定を行います。
R_MTR_SetUserifMode	指令値（位置、速度）の自動更新に関するモードを設定します。
R_MTR_ExecEvent	システムの動作状態に対するイベントを発行します。
R_MTR_ChargeCapacitor	インバータ母線電圧が安定するまで待ちます。
R_MTR_GetLoopModeStatus	設定されている制御ループモードを取得します。
R_MTR_SetPositionStatus	位置指令値の入力方式を設定します。
R_MTR_SetPosition	位置指令値[deg]を設定します。
R_MTR_GetPosition	現在位置の値[deg]を取得します。
R_MTR_GetPositioningFLag	位置決め完了フラグの値を取得します。
R_MTR_SetSpeed	速度指令値[rpm]を設定します。
R_MTR_GetSpeed	現在速度の値[rpm]を取得します。
R_MTR_SetDir	モータの回転方向を設定します。
R_MTR_GetDir	モータの回転方向を取得します。
R_MTR_GetStatus	モータ制御プログラムのシステム動作状態を取得します。
R_MTR_InputBuffParamReset	ICS 入力用変数バッファをデフォルト値に設定します。
R_MTR_CtrlInput	ICS 入力用変数を ICS 入力用変数バッファにコピーします。
R_MTR_SetVariables	ICS 入力用変数バッファの内容をモータ制御パラメータに設定します。
R_MTR_AutoSetVariables	ICS 入力用変数バッファの位置指令値、速度指令値をモータ制御パラメータに設定します。
R_MTR_CtrlGainCalc	モータ制御パラメータのゲインを算出します。
R_MTR_UpdatePolling	モータ制御パラメータに対する書き込み許可フラグをポーリングします。
R_MTR_GetErrorStatus	エンコーダ位置/速度制御のエラーステータスを取得します。
R_MTR_GetPositionPFStatus	エンコーダ位置制御のプロファイルステータスを取得します。
R_MTR_SetPositionUnits	位置指令値[count]を設定します。
R_MTR_SetActualPositionUnits	現在位置[count]を設定します。
R_MTR_GetPositionUnits	現在位置の値[count]を取得します。
R_MTR_GetSpeedUnits	現在速度の値[count/s]を取得します。
R_MTR_SetAccelerationUnits	加速度指令値[count/s <sup>2</sup> ]を設定します。
R_MTR_SetDecelerationUnits	減速度指令値[count/s <sup>2</sup> ]を設定します。

---

## 5.2 R\_MTR\_InitControl

---

モータ制御プログラムの初期設定を行います。この関数は他の API 関数を使用する前に実行される必要があります。

### Format

```
void R_MTR_InitControl (uint8_t u1_id)
```

### Parameters

u1\_id

制御対象のモータの ID を指定します。

(注)サンプルプログラムで制御できるのは 1 個のモータです。

```
MTR_ID_A /* モータ A*/
```

### Return Values

なし

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

システム動作状態の初期化やモータ制御パラメータのデフォルト値の設定を行います。

### Example

```
/* Initialize motor FOC control by ID "MTR_ID_A" */  
R_MTR_InitControl(MTR_ID_A);
```

---

### 5.3 R\_MTR\_SetUserifMode

---

ユーザインタフェースモジュールで設定される指令値（位置、速度）の自動更新に関するモードを設定する関数です。

#### Format

```
void R_MTR_SetUserifMode (uint8_t u1_user_mode)
```

#### Parameters

u1\_user\_mode

自動更新の許可・禁止を設定します。

MTR\_DISABLE\_AUTO\_SET(0)

MTR\_ENABLE\_AUTO\_SET(1)

#### Return Values

なし

#### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

#### Description

タイマーによる定期的な自動更新を許可するか否かを設定します。

ボードユーザインタフェース使用時のときのみ、許可します。

サンプルプログラムのデフォルト値は自動更新禁止です。

#### Example

```
/**  
if (BOARD_UI == g_u1_sw_userif)  
{  
    R_MTR_SetUserifMode(MTR_ENABLE_AUTO_SET);  
}
```

## 5.4 R\_MTR\_ExecEvent

この関数は、モータ制御プログラムのシステムの動作状態に対するイベントを発行します。

### Format

```
void R_MTR_ExecEvent (uint8_t u1_event, uint8_t u1_id)
```

### Parameters

u1\_event

イベント名	値	発生要因
MTR_EVENT_INACTIVE	0x00	モータのトルクを ON にするとき
MTR_EVENT_ACTIVE	0x01	モータのトルクを OFF にするとき
MTR_EVENT_ERROR	0x02	システムがエラーを検出したとき
MTR_EVENT_RESET	0x03	初期化、またはエラーから復帰するとき

u1\_id

制御対象のモータの ID を指定します。

MTR\_ID\_A /\* モータ A\*/

### Return Values

なし

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

引数として指定するイベントを発生させることにより、システムの動作状態を遷移させます。

### Example

```
/* Execution ACTIVE event */  
R_MTR_ExecEvent(MTR_EVENT_ACTIVE, MTR_ID_A);
```

---

## 5.5 R\_MTR\_ChargeCapacitor

---

この関数は、インバータ母線電圧が安定するまで待ちます。モータ制御を行う前に必ず本関数を実行する必要があります。

### Format

```
void R_MTR_ChargeCapacitor(uint8_t u1_id)
```

### Parameters

u1\_id  
制御対象のモータの ID を指定します。  
MTR\_ID\_A /\* モータ A\*/

### Return Values

なし

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

インバータ母線電圧(VDC) が DC24V の 80%を超えているかチェックします。  
超えるまでタイムアウト無しで待ちます。

### Example

```
/* Wait for charging capacitor */  
R_MTR_ChargeCapacitor(MTR_ID_A);
```

---

## 5.6 R\_MTR\_GetLoopModeStatus

---

この関数は、現在設定されている制御ループモードを取得します。

### Format

```
uint8_t R_MTR_GetLoopModeStatus(uint8_t u1_id)
```

### Parameters

u1\_id

制御対象のモータの ID を指定します。

MTR\_ID\_A /\* モータ A\*/

### Return Values

制御ループモード

MTR\_LOOP\_SPEED(0): 速度制御

MTR\_LOOP\_POSITION(1): 位置制御

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

サンプルプログラムでの制御ループモードのデフォルト値は位置制御です。

### Example

```
uint8_t u1_status
```

```
u1_status = R_MTR_GetLoopModeStatus(MTR_ID_A);
```



---

## 5.7 R\_MTR\_SetPositionStatus

---

この関数は、位置指令値の入力方式を設定します。

### Format

```
void R_MTR_SetPositionStatus(uint8_t u1_pos_status)
```

### Parameters

u1\_pos\_status

入力方式

MTR\_POS\_CONST(0) : 0 指令

MTR\_POS\_STEP(1) : 直接入力(ステップ入力)

MTR\_POS\_TRAPEZOID(2) : 指令値作成

### Return Values

なし

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

サンプルプログラムの入力方式のデフォルト値は MTR\_POS\_TRAPEZOID です。

### Example

```
/* set position state "STEP"*/  
R_MTR_SetPositionStatus(MTR_POS_STEP);
```

---

## 5.8 R\_MTR\_SetPosition

---

この関数は、位置指令値[deg]を設定します。

### Format

```
void R_MTR_SetPosition(int16_t s2_ref_position)
```

### Parameters

s2\_ref\_position  
位置指令値[deg]

### Return Values

なし

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

位置指令値は符号付で単位は[deg]です。  
なおモータ始動時の磁石位置の初期化後の位置を 0[deg]とします。

### Example

```
/* Set reference position 180[deg] */  
R_MTR_SetPosition(180);
```

---

## 5.9 R\_MTR\_GetPosition

---

この関数は、現在位置の値[deg]を取得します。

### Format

```
int16_t R_MTR_GetPosition(uint8_t u1_id)
```

### Parameters

u1\_id

制御対象のモータの ID を指定します。

MTR\_ID\_A /\* モータ A\*/

### Return Values

現在位置[deg]

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

現在位置の値は符号付きで単位は[deg]です。

なおモータ始動時の磁石位置の初期化後の位置を 0[deg]とします

### Example

```
int16_t s2_pos;
```

```
/* Get current position */
```

```
s2_pos = R_MTR_GetPosition(MTR_ID_A);
```

---

## 5.10 R\_MTR\_GetPositioningFlag

---

この関数は、位置決め完了フラグの値を取得します。

### Format

uint8\_t R\_MTR\_GetPositioningFlag (uint8\_t u1\_id)

### Parameters

u1\_id

制御対象のモータの ID を指定します。

MTR\_ID\_A /\* モータ A\*/

### Return Values

MTR\_FLG\_CLR(0) : 位置決め未完了

MTR\_FLG\_SET(1) : 位置決め完了

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

位置決め完了は、目標位置と現在位置との差分がある範囲内に収まっているかどうかで判断します。

### Example

```
/* If the positioning flag is set, then turn on the led */
if (MTR_FLG_SET == R_MTR_GetPositioningFlag(MTR_ID_A))
{
    led_on();
}
```

---

## 5.11 R\_MTR\_SetSpeed

---

この関数は、速度指令値[rpm]を設定します。

### Format

```
void R_MTR_SetSpeed(int16_t ref_speed)
```

### Parameters

ref\_speed  
速度指令値[rpm]

### Return Values

なし

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

速度指令値の単位は[rpm]です。負の値の場合、正回転と逆方向に回転します。  
サンプルプログラムでの設定有効範囲は-2000[rpm] ~ 2000[rpm]です。

### Example

```
/* Set reference position 2000[rpm] */  
R_MTR_SetSpeed(2000);
```

---

## 5.12 R\_MTR\_GetSpeed

---

この関数は、現在速度の値[rpm]を取得します。

### Format

```
int16_t R_MTR_GetSpeed (uint8_t u1_id)
```

### Parameters

u1\_id

制御対象のモータの ID を指定します。

MTR\_ID\_A /\* モータ A\*/

### Return Values

現在速度[rpm]

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

関数実行時の速度の値を取得します。単位は[rpm]です。

値が負のときは正回転と逆方向に回転していることを示します。

### Example

```
int16_t s2_speed;
```

```
/* Get current speed */
```

```
s2_speed = R_MTR_GetSpeed(MTR_ID_A);
```

---

### 5.13 R\_MTR\_SetDir

---

この関数は、モータの回転方向を設定します。

#### Format

```
void R_MTR_SetDir (uint8_t dir, uint8_t u1_id)
```

#### Parameters

dir

回転方向

MTR\_CW(0) : 時計回り (右回り)

MTR\_CCW(1) : 左周り

u1\_id

制御対象のモータの ID を指定します。

MTR\_ID\_A /\* モータ A\*/

#### Return Values

なし

#### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

#### Description

位置や速度は設定された回転方向を正方向として処理されます。  
サンプルプログラムのデフォルト値は CW です。

#### Example

```
/* Set the direction to CW */  
R_MTR_SetDir(MTR_CW, MTR_ID_A);
```

---

## 5.14 R\_MTR\_GetDir

---

この関数は、モータ回転方向を取得します。

### Format

```
uint8_t R_MTR_GetDir(uint8_t u1_id)
```

### Parameters

u1\_id

制御対象のモータの ID を指定します。

MTR\_ID\_A /\* モータ A\*/

### Return Values

#### 回転方向

MTR\_CW(0) : 右回り

MTR\_CCW(1) : 左周り

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

関数実行時の回転方向を取得します。

### Example

```
uint8_t u1_dir;
```

```
/* Get direction */
```

```
u1_dir = R_MTR_GetDir(MTR_ID_A);
```



---

## 5.15 R\_MTR\_GetStatus

---

この関数は、モータ制御プログラムのシステム動作状態を取得します。

### Format

```
uint8_t R_MTR_GetStatus(uint8_t u1_id)
```

### Parameters

u1\_id

制御対象のモータの ID を指定します。

MTR\_ID\_A /\* モータ A\*/

### Return Values

システム動作状態

MTR\_MODE\_INACTIVE (0x00)

MTR\_MODE\_ACTIVE (0x01)

MTR\_MODE\_ERROR (0x02)

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

システムの動作状態は、モータ駆動停止 (INACTIVE)、モータ駆動 (ACTIVE)、異常状態 (ERROR) があります。

### Example

```
uint8_t u1_motor_status
```

```
/* Get status of motor control system */
```

```
u1_motor_status = R_MTR_GetStatus(MTR_ID_A);
```

---

## 5.16 R\_MTR\_InputBuffParamReset

---

この関数は、ICS 入力用変数バッファをデフォルト値に設定します。

### Format

```
void R_MTR_InputBuffParamReset(void)
```

### Parameters

なし

### Return Values

なし

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

ICS 入力用変数バッファはユーザインタフェースモジュール(User Interface Module)により入力されたモータ制御パラメータ等をインタフェースモジュール(Interface Module)で使用するためのものです。

変数名	型	変数シンボル	使用モジュール/レイヤ
ICS 入力用変数	mtr_ctrl_input_t 型	st_ctrl_input	ユーザインタフェース /アプリケーション
ICS 入力用変数バッファ	mtr_ctrl_input_t 型	st_ctrl_input_buff	インタフェース/ミドル

### Example

```
/* Initialize st_ctrl_input_buff parameters */  
R_MTR_InputBuffParamReset();
```

---

## 5.17 R\_MTR\_CtrlInput

---

この関数は、ICS 入力用変数を ICS 入力用変数バッファにコピーします。

### Format

```
void R_MTR_CtrlInput(mtr_ctrl_input_t *st_ctrl_input)
```

### Parameters

mtr\_ctrl\_input\_t 型構造体へのポインタ

### Return Values

なし

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

ポインタが指す mtr\_ctrl\_input\_t 型変数から、ICS 入力用変数バッファ st\_ctrl\_input\_buff にコピーします。更にモータ制御パラメータへの書き込み許可フラグ(u1\_trig\_enable\_write)をセットします。

### Example

```
/* Structure for ICS input */
mtr_ctrl_input_t st_ctrl_input;

/* copy variables */
R_MTR_CtrlInput(&st_ctrl_input);
```

---

## 5.18 R\_MTR\_SetVariables

---

この関数は、ICS 入力用変数バッファの内容をモータ制御パラメータに設定します。

### Format

```
void R_MTR_SetVariables(void)
```

### Parameters

なし

### Return Values

なし

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

ICS 入力用変数バッファ st\_ctrl\_input\_buff に設定された各種パラメータの値を対応するモータ制御パラメータに設定します。

### Example

```
/* Set control input buffer to motor control structure members */  
R_MTR_SetVariables();
```

---

## 5.19 R\_MTR\_AutoSetVariables

---

この関数は、ICS 入力用変数バッファの位置指令値、速度指令値をモータ制御パラメータに設定します。

### Format

```
void R_MTR_AutoSetVariables(void)
```

### Parameters

なし

### Return Values

なし

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

500us 周期割り込みで実行されます。

パラメータの自動更新モードが許可されているときのみ、設定が有効です。

### Example

```
/* Set control input buffer to motor control structure members */  
R_MTR_AutoSetVariables ();
```

---

## 5.20 R\_MTR\_CtrlGainCalc

---

この関数は、モータ制御パラメータのゲインを算出します。

### Format

```
void R_MTR_CtrlGainCalc(void)
```

### Parameters

なし

### Return Values

なし

### Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

### Description

PI 制御、IPD 制御、速度オブザーバ制御のゲインを算出します。

### Example

```
/* Motor gain calculation*/  
R_MTR_CtrlGainCalc();
```

---

## 5.21 R\_MTR\_UpdatePolling

---

この関数は、モータ制御パラメータに対する書き込み許可フラグをポーリングします。

Format

```
void R_MTR_UpdatePolling(void)
```

Parameters

なし

Return Values

なし

Properties

r\_mtr\_driver\_acces.h にプロトタイプ宣言されています。

Description

モータ制御パラメータに対する書き込み許可フラグ(u1\_trig\_enable\_write)をポーリングし、フラグがセットされているとき ICS 入力用変数バッファの値をモータ制御パラメータに設定 (R\_MTR\_SetVariables 関数)、またモータ制御パラメータのゲインを算出(R\_MTR\_CtrlGainCalc 関数)します。実行後、モータ制御パラメータへの書き込み許可フラグ(u1\_trig\_enable\_write)をクリアします。

Example

```
/* Update commands and configurations when trigger flag is set */  
R_MTR_UpdatePolling();
```

## 5.22 R\_MTR\_GetErrorStatus

この関数は、エンコーダ位置/速度制御のエラーステータスを取得します。

## Format

```
uint16_t R_MTR_GetErrorStatus(uint8_t u1_id)
```

## Parameters

u1\_id

制御対象のモータの ID を指定します。

MTR\_ID\_A /\* モータ A\*/

## Return Values

エラーステータス

## Properties

r\_mtr\_driver\_ecat\_acces.h にプロトタイプ宣言されています。

## Description

エラーステータスは表のとおりです。

マクロ名	値	エラー種別
MTR_ERROR_NONE	0x0000	エラーなし
MTR_ERROR_OVER_CURRENT_HW	0x0001	過電流エラー (H/W 検出)
MTR_ERROR_OVER_VOLTAGE	0x0002	インバータ母線電圧過電圧エラー
MTR_ERROR_OVER_SPEED	0x0004	回転速度エラー
MTR_ERROR_HALL_TIMEOUT	0x0008	未使用
MTR_ERROR_BEMF_TIMEOUT	0x0010	未使用
MTR_ERROR_HALL_PATTERN	0x0020	ホール検出角度エラー
MTR_ERROR_BEMF_PATTERN	0x0040	未使用
MTR_ERROR_UNDER_VOLTAGE	0x0080	インバータ母線電圧低電圧エラー
MTR_ERROR_OVER_CURRENT_SW	0x0100	過電流エラー(S/W 検出)
MTR_ERROR_UNKNOWN	0xffff	未使用

## Example

```
uint16_t u2_error_status;
```

```
/* Get FOC error status */
```

```
u2_error_status = R_MTR_GetErrorStatus(MTR_ID_A);
```



---

### 5.23 R\_MTR\_GetPositionPFStatus

---

この関数は、エンコーダ位置制御のプロファイルステータスを取得します。

#### Format

```
uint8_t R_MTR_GetPositionPFStatus(uint8_t u1_id)
```

#### Parameters

u1\_id

制御対象のモータの ID を指定します。

MTR\_ID\_A /\* モータ A\*/

#### Return Values

プロファイルステータス

MTR\_POS\_STEADY\_STATE (0) : 安定状態(現在位置を変更していない)

MTR\_POS\_TRANSITION\_STATE (1) : 遷移状態(現在位置を変更している)

#### Properties

r\_mtr\_driver\_ecat\_acces.h にプロトタイプ宣言されています。

#### Description

位置制御時にモータが静止状態かどうかを調べるときに本関数を使用できます。

#### Example

```
uint8_t u1_pos_state;
```

```
/* Get position profile status */
```

```
u1_pos_state = R_MTR_GetPositionPFStatus(MTR_ID_A);
```

---

## 5.24 R\_MTR\_SetPositionUnits

---

この関数は、位置指令値[count]を設定します。

### Format

```
void R_MTR_SetPositionUnits (int32_t s4_position_units)
```

### Parameters

位置指令値[count]

### Return Values

なし

### Properties

r\_mtr\_driver\_ecat\_acces.h にプロトタイプ宣言されています。

### Description

位置指令値は符号付で単位は[count]です。

なおモータ始動時の磁石位置の初期化後の位置を 0[count]とします。

### Example

```
/* 2000 count is equivalent to 180 degrees */  
R_MTR_SetPositionUnits (2000);
```

---

## 5.25 R\_MTR\_SetActualPositionUnits

---

この関数は、現在位置[count]を設定します。

### Format

```
void R_MTR_SetActualPositionUnits (int32_t s4_position_units)
```

### Parameters

位置指令値[count]

### Return Values

なし

### Properties

r\_mtr\_driver\_ecat\_acces.h にプロトタイプ宣言されています。

### Description

位置指令値は符号付で単位は[count]です。

本関数はモータ制御を行わずに現在位置の値のみを設定する関数です。

ホーミングモードで初期位置にオフセットを持たせたいとき等に使用できます。

### Example

```
/* Set current position */  
R_MTR_SetActualPosition(2000);
```

---

## 5.26 R\_MTR\_GetPositionUnits

---

この関数は、現在位置の値[count]を取得します。

### Format

```
int32_t R_MTR_GetPositionUnits(uint8_t u1_id)
```

### Parameters

u1\_id

制御対象のモータの ID を指定します。

MTR\_ID\_A /\* モータ A\*/

### Return Values

現在位置[count]

### Properties

r\_mtr\_driver\_ecat\_acces.h にプロトタイプ宣言されています。

### Description

現在位置の値は符号付で単位は[count]です。

### Example

```
int32_t s4_current_pos_units;
```

```
/* Get position units */
```

```
s4_current_pos_units = R_MTR_GetPositionUnits(MTR_ID_A);
```

---

## 5.27 R\_MTR\_GetSpeedUnits

---

この関数は、現在速度の値[count/s]を取得します。

### Format

```
int32_t R_MTR_GetSpeedUnits(uint8_t u1_id)
```

### Parameters

u1\_id

制御対象のモータの ID を指定します。

MTR\_ID\_A /\* モータ A\*/

### Return Values

現在速度[count/s]を取得します

### Properties

r\_mtr\_driver\_ecat\_acces.h にプロトタイプ宣言されています。

### Description

現在速度の値は符号付で単位は[count/s]です。

16.16 ビット固定小数点形式に変換した値となります。

### Example

```
int32_t s4_speed_units;
```

```
/* Get speed units */
```

```
s4_speed_units = R_MTR_GetSpeedUnits(MTR_ID_A);
```

---

## 5.28 R\_MTR\_SetSpeedUnits

---

この関数は、速度指令値を設定します。

### Format

```
void R_MTR_SetSpeedUnits(int32_t s4_speed_units)
```

### Parameters

速度指令値[count/s]

### Return Values

なし

### Properties

r\_mtr\_driver\_ecat\_access.h にプロトタイプ宣言されています。

### Description

速度指令値は符号付で単位は[count/s]です。

16.16 ビット固定小数点形式に変換した値となります。

### Example

```
/* 4 369 066 count/s is equivalent to 2000 rpm */  
R_MTR_SetSpeedUnits(4369066);
```

---

## 5.29 R\_MTR\_SetAccelerationUnits

---

この関数は、加速度指令値[count/s<sup>2</sup>]を設定します。

### Format

```
void R_MTR_SetAccelerationUnits(uint32_t u4_acceleration_units)
```

### Parameters

加速度指令値[count/s<sup>2</sup>]

### Return Values

なし

### Properties

r\_mtr\_driver\_ecat\_acces.h にプロトタイプ宣言されています。

### Description

加速度指令値は符号付で単位は[count/s<sup>2</sup>]です。  
16.16 ビット固定小数点形式に変換した値となります。

### Example

```
/* Set acceleration 10[cout/s2] */  
R_MTR_SetAccelerationUnits (10*65536);
```

---

### 5.30 R\_MTR\_SetDecelerationUnits

---

この関数は、減速度指令値を設定します。

#### Format

```
void R_MTR_SetDecelerationUnits(uint32_t u4_deceleration_units)
```

#### Parameters

減速度指令値[count/s<sup>2</sup>]

#### Return Values

なし

#### Properties

r\_mtr\_driver\_ecat\_acces.h にプロトタイプ宣言されています。

#### Description

減速度指令値は符号付で単位は[count/s<sup>2</sup>]です。

16.16 ビット固定小数点形式に変換した値となります。

【注】 サンプルプログラムでは減速度指令値をモータ制御に使用していません。

#### Example

```
/* Set deceleration 10[cout/s2] */  
R_MTR_SetDecelerationUnits (10*65536);
```



## 6. ソリューションキットでの動作確認

本章ではモータソリューションキットを用いて EtherCAT 通信でモータを制御するサンプルアプリケーションの動作について説明します。

### 6.1 動作環境

本マニュアルのサンプルプログラムは、下記の環境を想定しています。

表 6.6-1 動作環境

項目	内容
使用ボード	RX72M CPU ボード Tessera Technology 製 TS-TCS02796  RX23T インバータボード : RTK0EM0006S01212BJ  モータエンコーダーI/F ボード
CPU	RX CPU (RXv3)
動作電圧	24V
通信プロトコル	EtherCAT
統合開発環境	CCRX コンパイラ(V3.01.00 以降) + e2studio(V7.5.0 以降)
エミュレータ	ルネサスエレクトロニクス 製 e2 Lite
SSC Tool	EtherCAT Technology Group (ETG) 提供 Slave Stack Code (SSC) Tool Version 5.12 以降
ソフトウェア PLC	Beckhoff Automation 製 TwinCAT® 3 (Beckhoff web サイトからダウンロード)  CODESYS

なお、SSC Tool、ソフトウェア PLC のインストールは完了しているものとします。

## 6.2 動作環境の設定、接続

電源、モータ、インバータボードを配線します。

- (1) モータの三相電源線とインバータボードの U、V、W 相の出力部を以下のように配線します。
  - 1-A) モータの茶色の線をインバータ U 相
  - 1-B) モータの青色の線をインバータ V 相
  - 1-C) モータの黒色の線をインバータ W 相

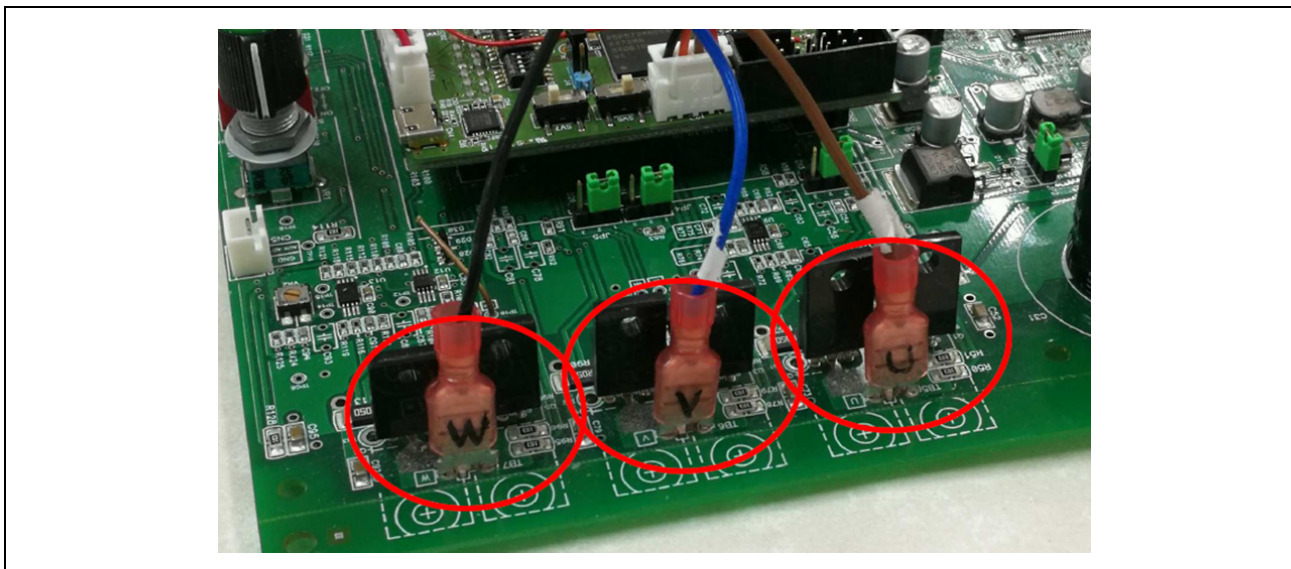


図 6-1 三相電源線の接続

- (2) モータエンコーダーとエンコーダ I/F 基板を以下のように配線します。

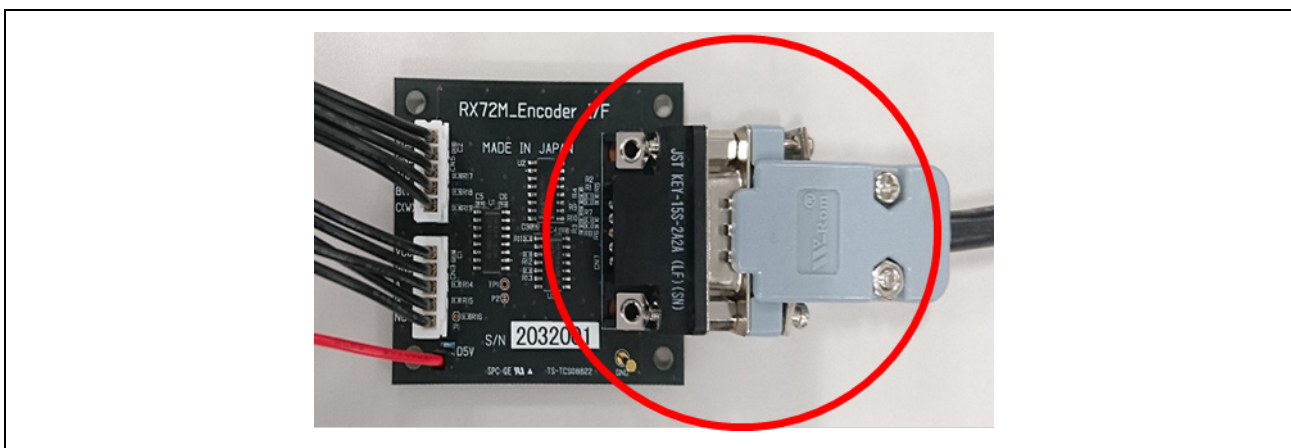


図 6-2 エンコーダ I/F 基板の接続

- (3) エンコーダーI/F基板とCPUボードの5Vを以下のように配線します。

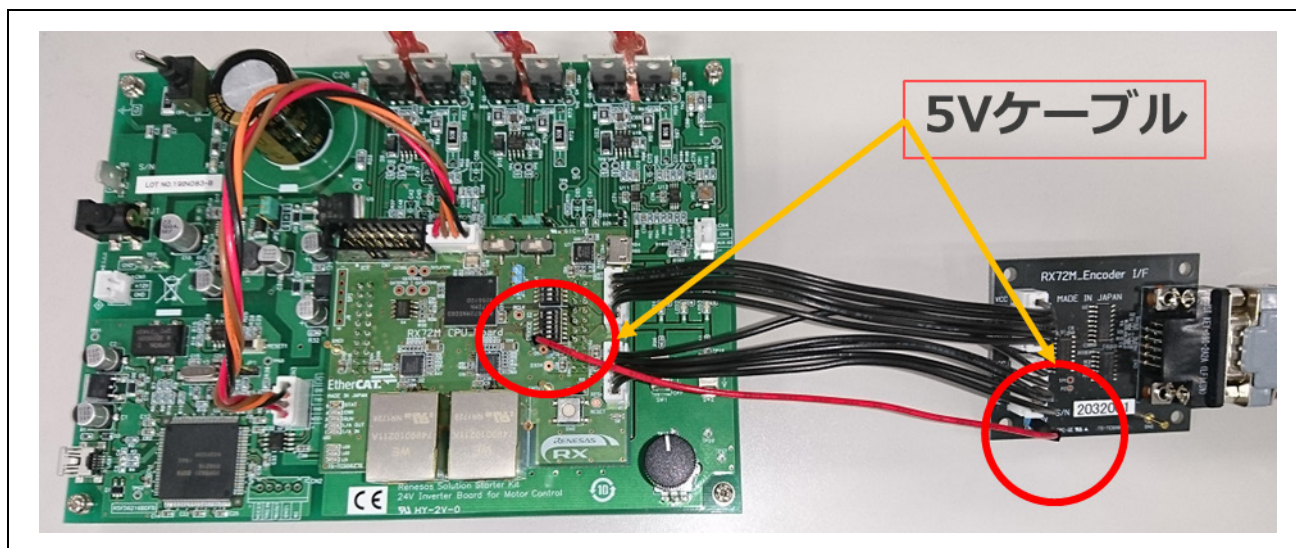


図 6-3 5V 電源の接続

- (4) インバータボードとCPUボードを以下のように配線します。
- CPUボードをインバータボードに取り付けてください。
  - ICSケーブルをCPUボードとインバータボードに取り付けてください。

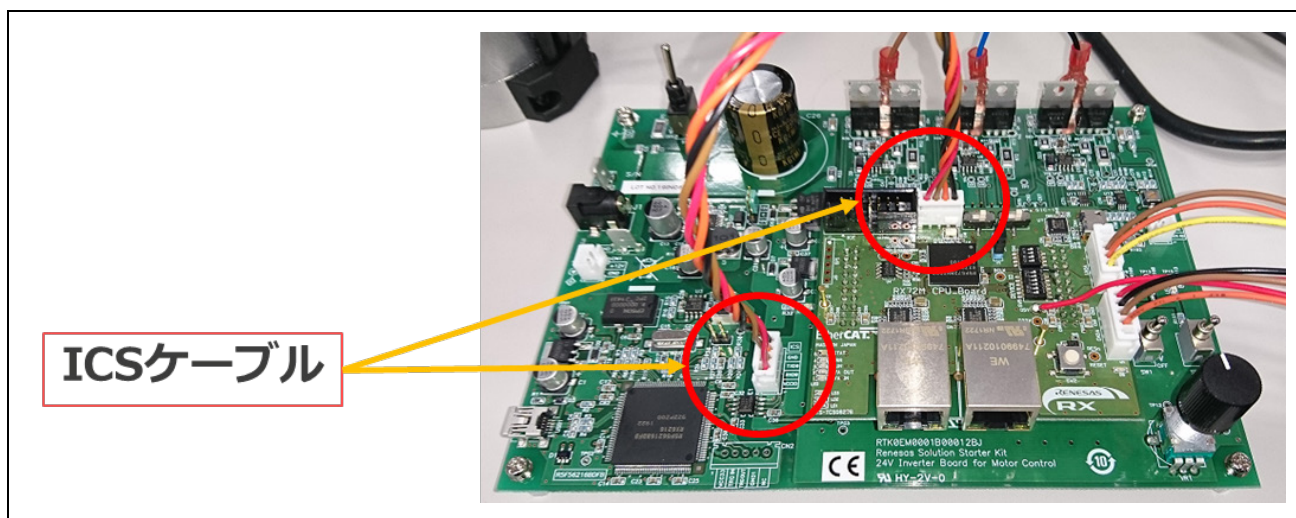


図 6-4 CPUボードとインバータボードの接続

(5) インバータボードに電源を以下のように接続します。

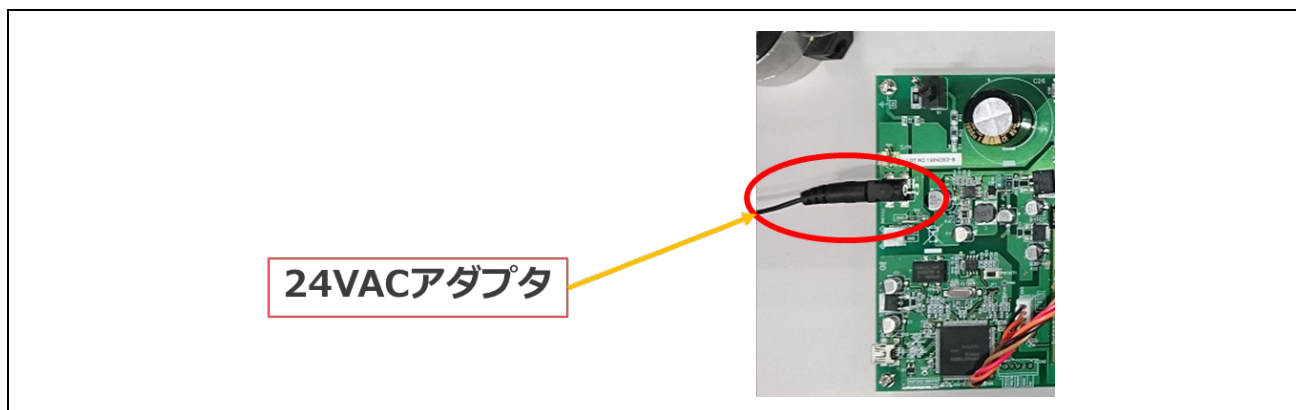


図 6-5 電源の接続

(6) 以下のような接続構成となります。

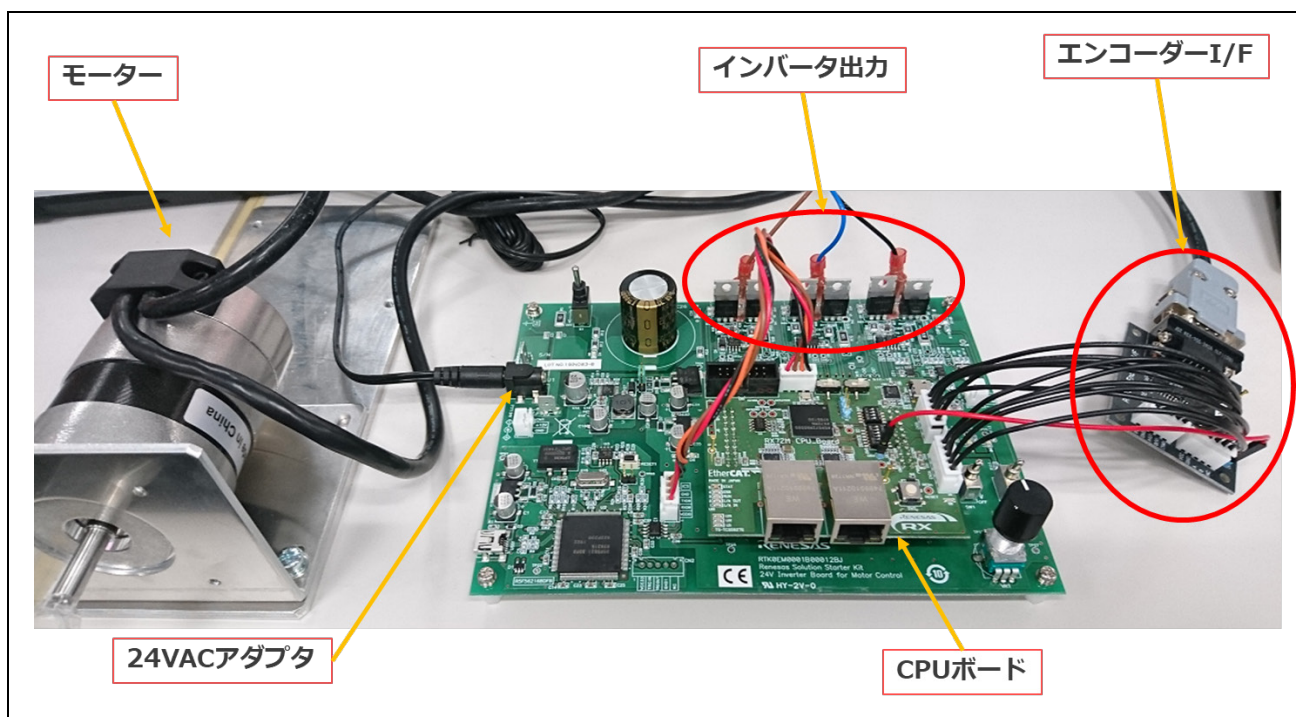


図 6-6 全体構成

(7) インバータボードの詳細を以下に示します。

- 電源投入は主 SW で行います。
- ICS I/F は RMW (Renesas Motor Workbench) 使用時に接続してください。
- SW1、2 は本マニュアルでは使用しません。

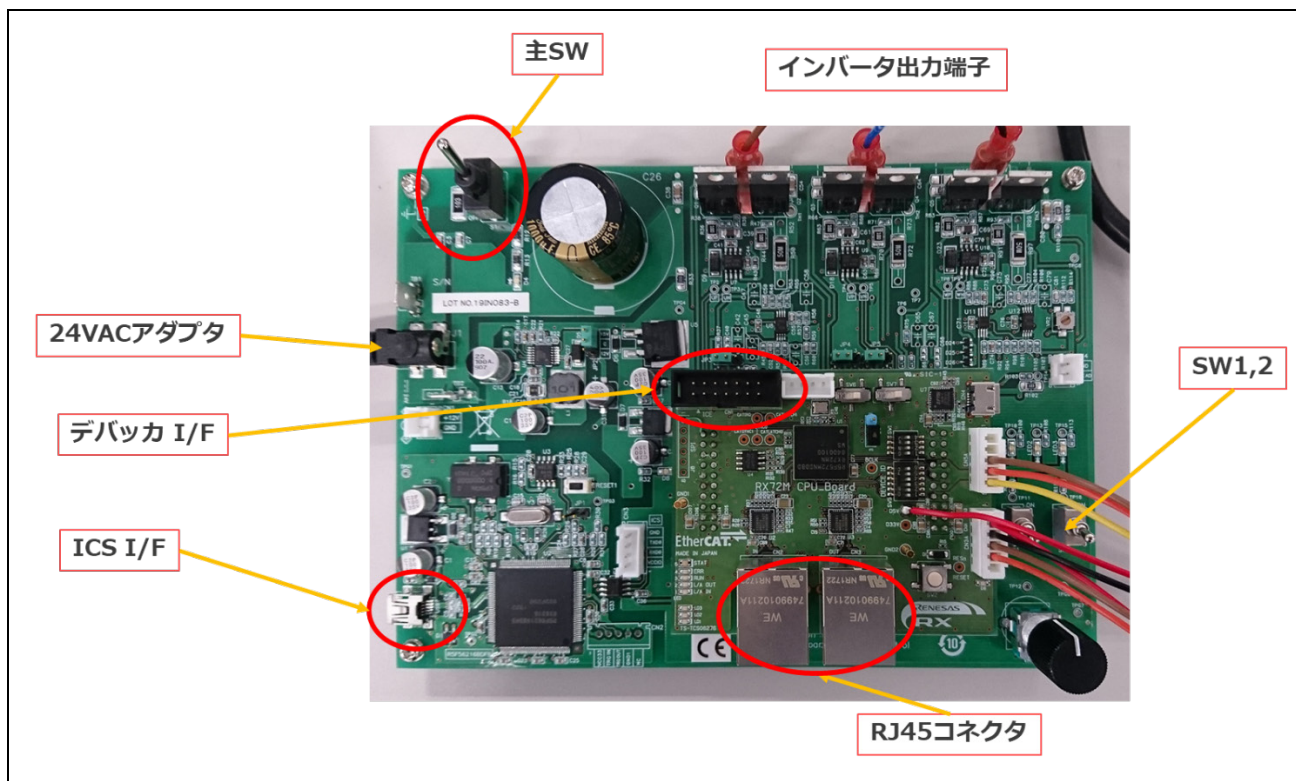


図 6-7 インバータボード詳細

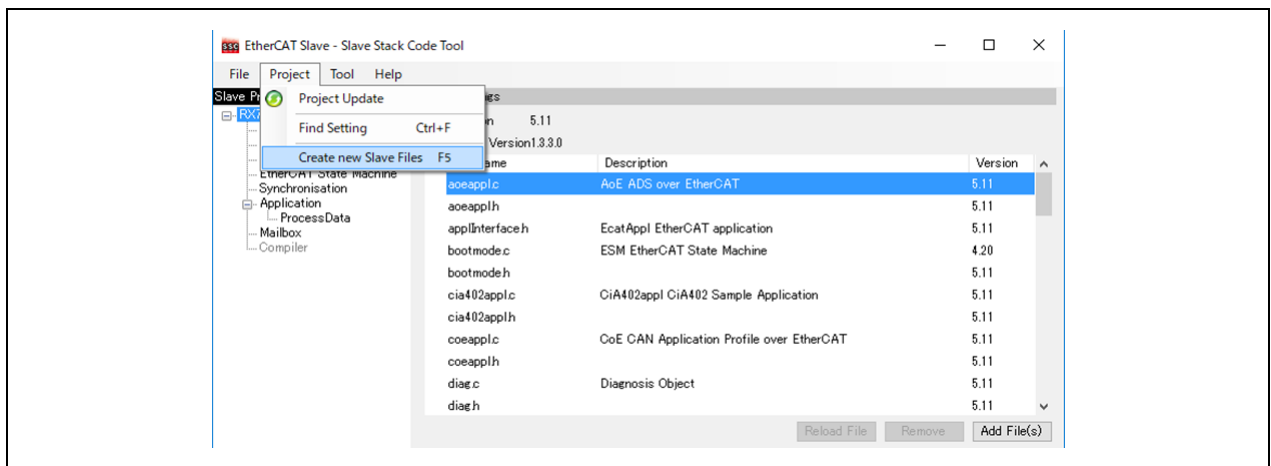
### 6.3 サンプルプログラムの構築

本サンプル・プロジェクトには EtherCAT スレーブスタックコードは同梱されていません。  
 EtherCAT スレーブスタックコードの生成には”EtherCAT Slave Stack Code(SSC) Tool”が必要です。  
 SSC Tool は ETG 協会から入手可能です。

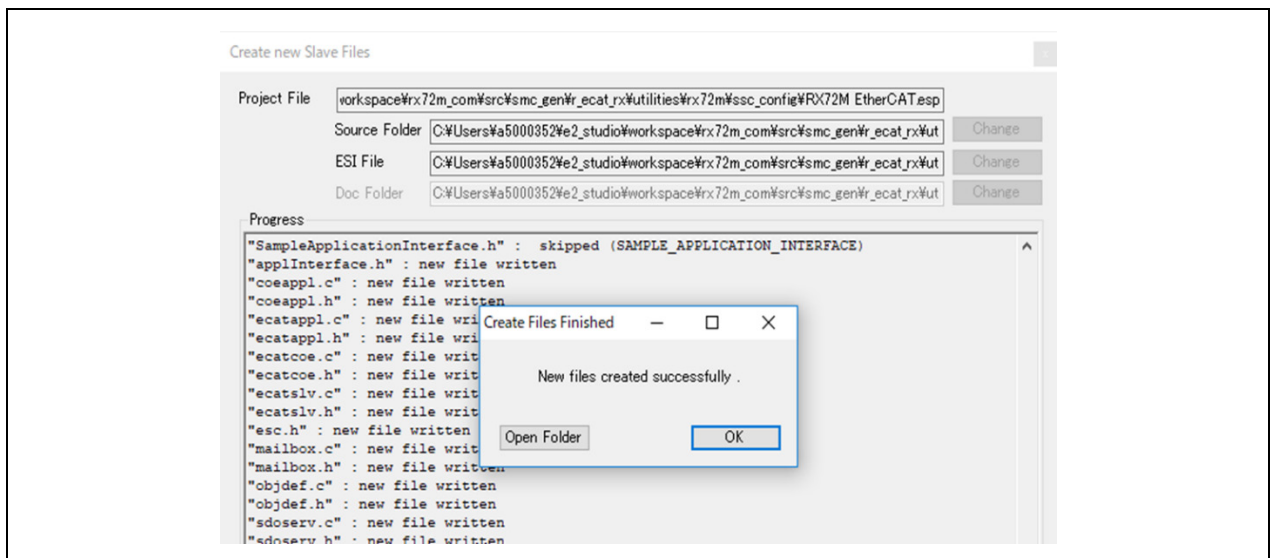
- (1) サンプルプログラムの SSC プロジェクトファイルをダブルクリックして SSC ツールを起動します。

```
ecat_cia402_motor_rsskrx72m\src\smc_gen\r_ecat_rx\utilities\rx72m\ssc_config
\rx72m EtherCAT CiA402.esp
```

- (2) [Project]→[Create New Slave Files]をクリック[Current new Slave Files] →[Start]をクリックします。



- (3) ソースコードが生成され、成功すると” New Files created successfully” と表示されるので[OK]をクリックします。



- (4) パッチコマンドをインストールしていない場合、GNU Patch Ver2.5.9 以後が必要です。インストール済みの場合は本手順をスキップしてください。

下記の Web サイトからパッチコマンド(Ver2.5.9)をダウンロードし” patch.exe” をコマンドプロンプトから実行可能なパスの通ったフォルダに格納します。

<http://gnuwin32.sourceforge.net/packages/patch.htm>

- (5) apply\_patch.bat ファイルを右クリックして[管理者として実行] ⇒ [はい]を選択します。パッチファイルは SSC ソースファイルに対する RX 向けの修正を含んでいます。

```
ecat_cia402_motor_rsskrx72m\src\smc_gen\r_ecat_rx\utilities\rx72m\batch_files\apply_patch.bat
```

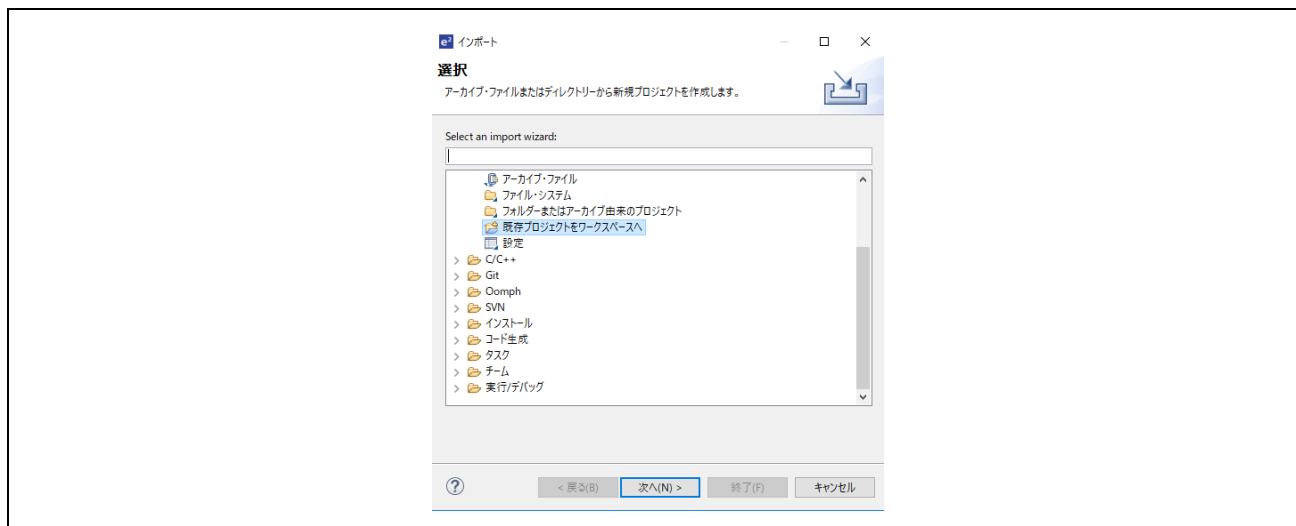
- (6) パッチ実行後、修正されたソースファイルは下記のフォルダに格納されます。

```
ecat_cia402_motor_rsskrx72m\src\application\ecat
```

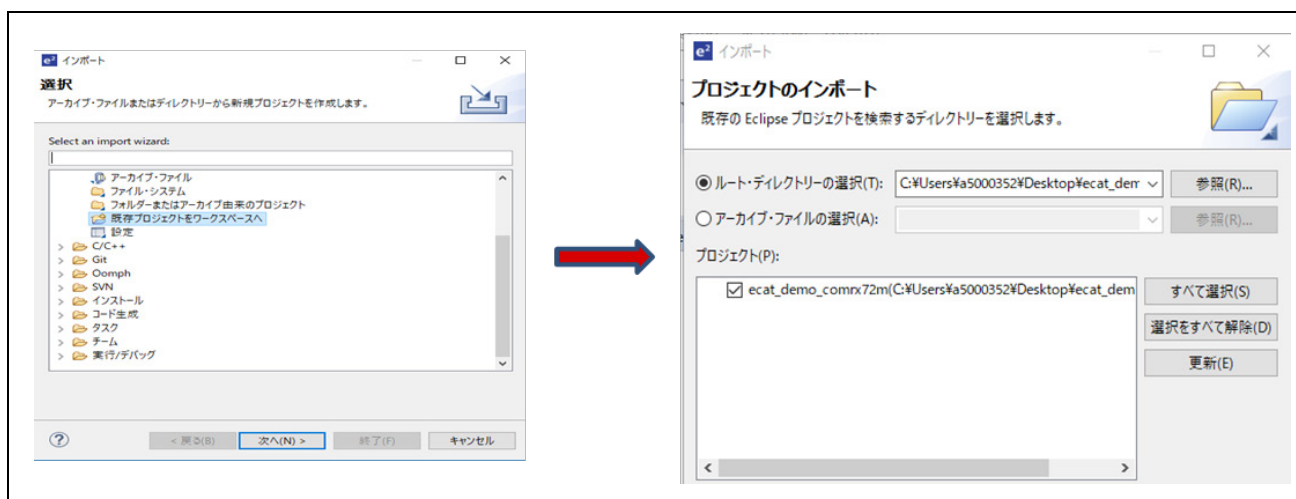
```
cmd.exe C:\WINDOWS\System32\cmd.exe
--- Move SSC Src folder ---
1 個のディレクトリを移動しました。
--- Patching process start ---
patching file Src/cia402appl.c
patching file Src/cia402appl.h
patching file Src/ecatcoe.h
patching file Src/mailbox.h
patching file Src/sdoserv.h
--- Patching process end ---
--- Move patched Src folder ---
1 個のディレクトリを移動しました。
続行するには何かキーを押してください . . .
```

6.4 サンプル・プロジェクトを e<sup>2</sup>studio にインポート

- (1) [ファイル]→[インポート]をクリックします。
- (2) [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[次へ]をクリックします。



- (3) [プロジェクトのインポート]ダイアログの[ルート・ディレクトリの選択]チェックボックスを選択し、[参照]をクリックします。
- (4) 通信ボード用サンプル・プロジェクトである"ecat\_cia402\_motor\_rsskrx72m"を選択して[開く]をクリックします。[プロジェクト]の"ecat\_cia402\_motor\_rsskrx72m"をチェックし[次へ]をクリックするとプロジェクトがインポートされます。

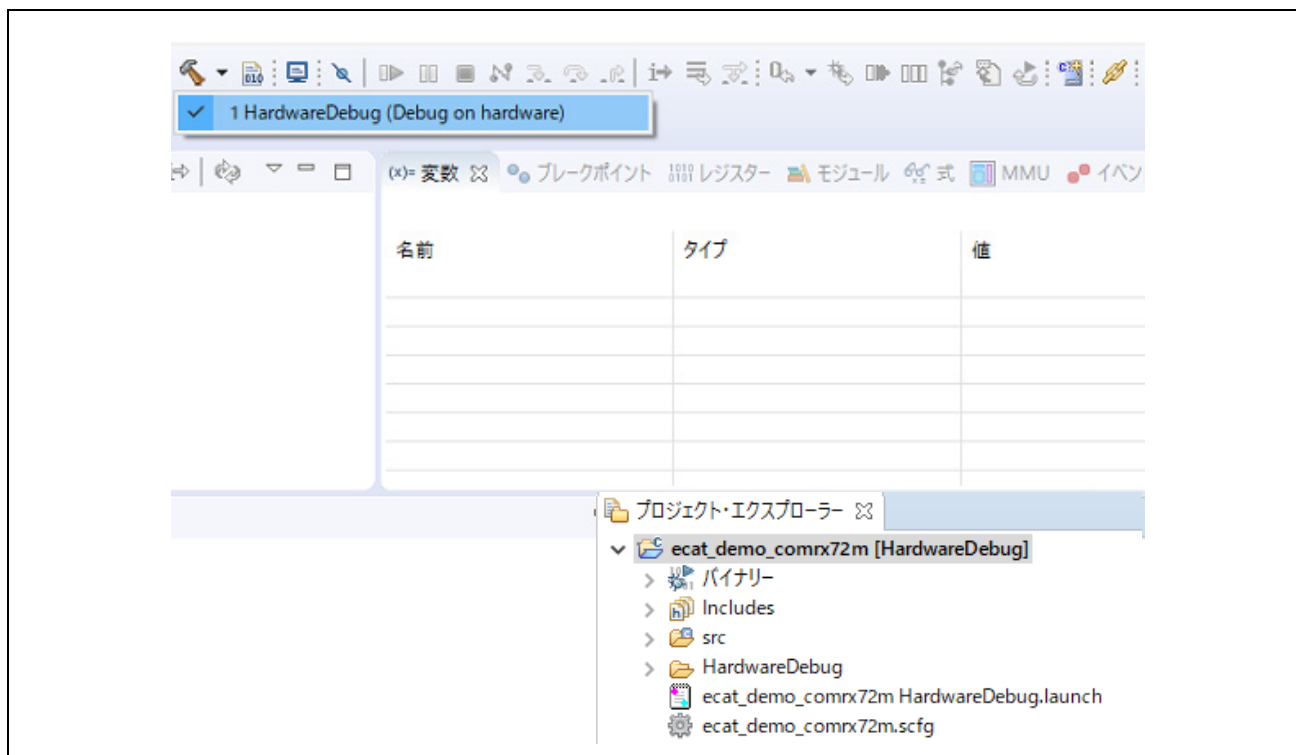




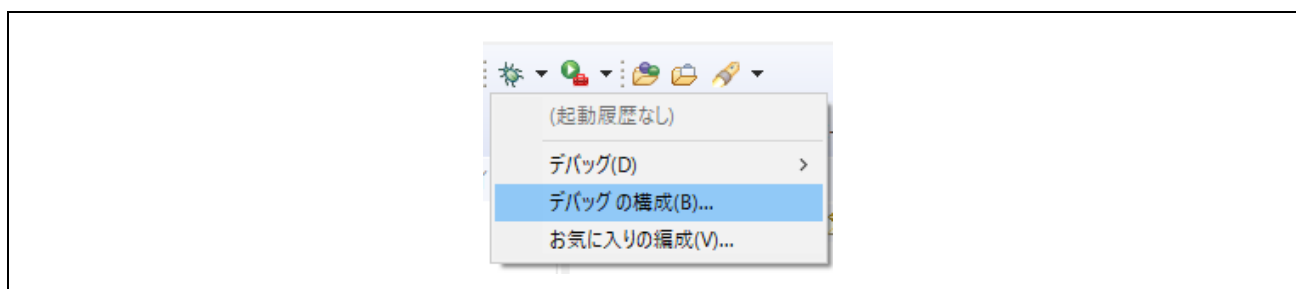
## 6.5 プログラミングとデバッグ

- (1) プロジェクト・エクスプローラーで“ecat\_cia402\_motor\_rsskrx72m”プロジェクトを左クリックし、[ビルド]ボタン（ハンマーアイコン）の横にある矢印をクリックし、ドロップダウンメニューから[Hardware Debug]を選択します。

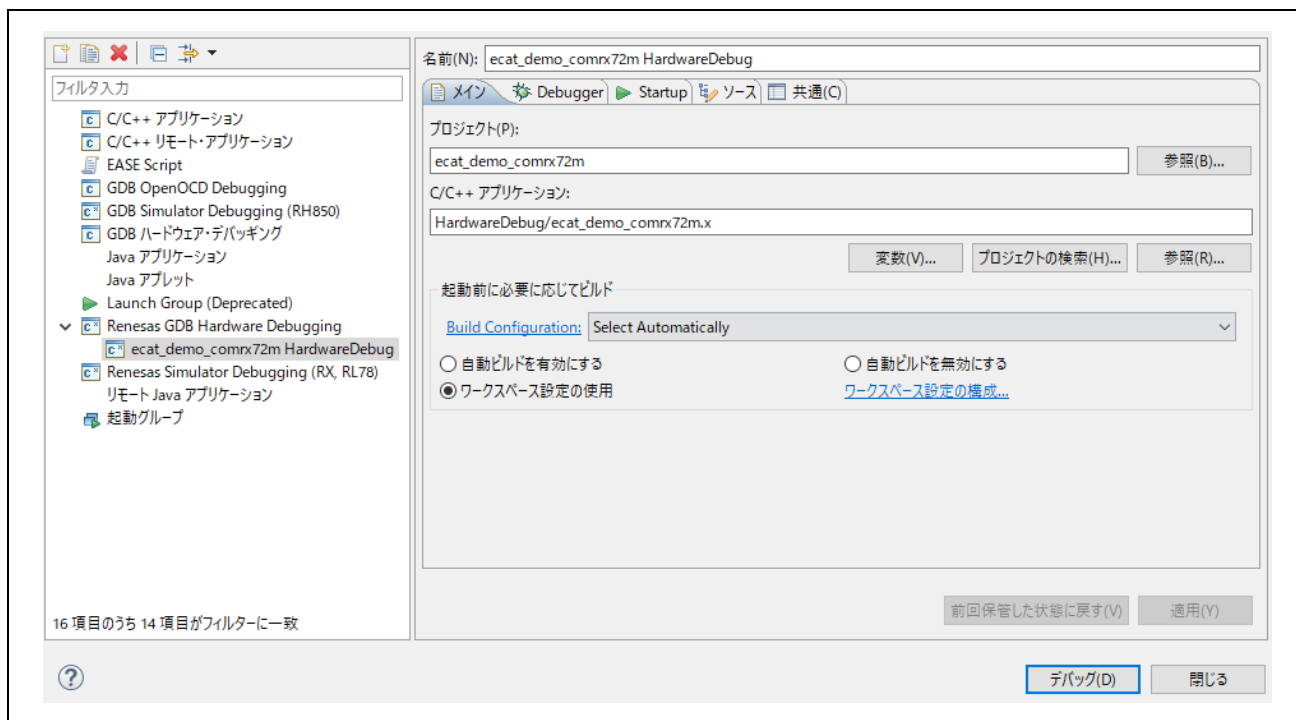
e2studio を使用してプロジェクトをビルドします。コンソール上でビルドエラーがないことを確認してください。



- (2) ビルドが完了したら、[デバッグ]ボタン（バグアイコン）の横にある矢印をクリックし、「デバッグ構成」を選択することでデバッグを開始できます。



- (3) “ecat\_cia402\_motor\_rsskrx72m Hardware Debug”をクリックしてターゲットにプログラムをダウンロードし、デバッグボタンを押して開始します。



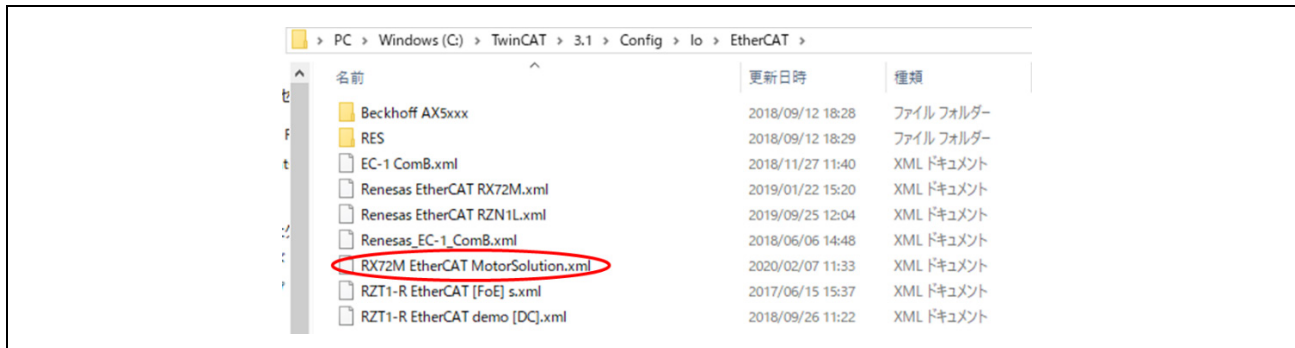
- (4) 'e2-server-gdb.exe'のファイアウォール警告が表示されることがあります。[自宅や職場のネットワークなどのプライベートネットワーク]のチェックボックスをチェックにして、<アクセスを許可>をクリックします。
- (5) ユーザーアカウント制御 (UAC) ダイアログが表示されることがあります。管理者パスワードを入力して、[はい]をクリックします。
- (6) パースペクティブ切り替えの確認ダイアログにてパースペクティブの変更を勧めるダイアログが表示される場合は「常にこの設定を使用する」チェックボックスにチェックし、[はい]をクリックします。
- (7) E2 Lite デバッガの緑色の「ACT」LED が常に点灯します。

コードをダウンロードしたら、<再開>ボタンをクリックして、メイン関数 main 最初の行までコードを実行します。もう一度<再開>ボタンをクリックすると、残りのコードでターゲットが実行されます。

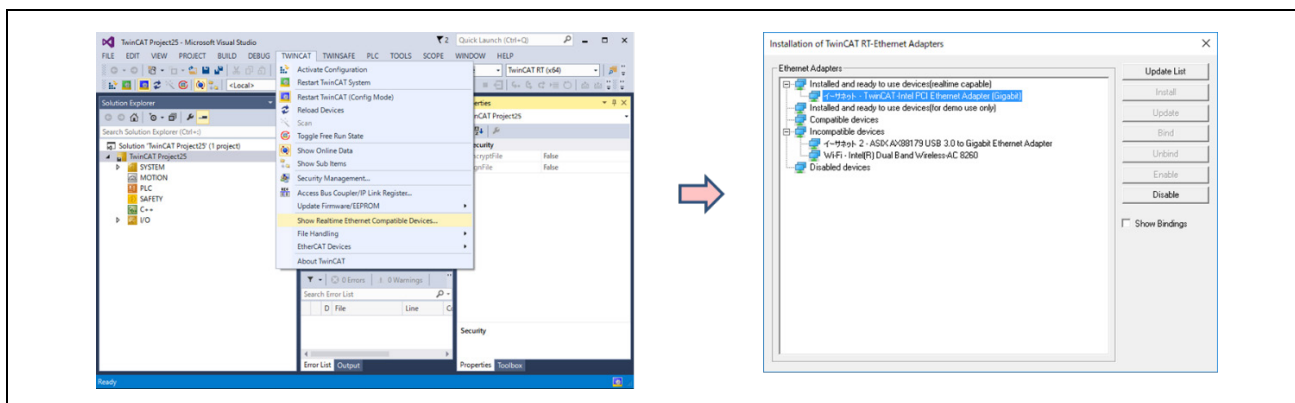
## 6.6 TwinCAT との接続 (ESI ファイルの書き込み)

- (1) TwinCAT を開始する前に、リリースフォルダに含まれている ESI ファイルを、  
“/TwinCAT / 3.x / Config / IO / EtherCAT” にコピーしてください

“ecat\_cia402\_motor\_rsskrx72m¥src¥smc\_gen¥r\_ecat\_rx¥utilities¥rx72m¥esi¥ RX72M EtherCAT MotorSolution.xml”



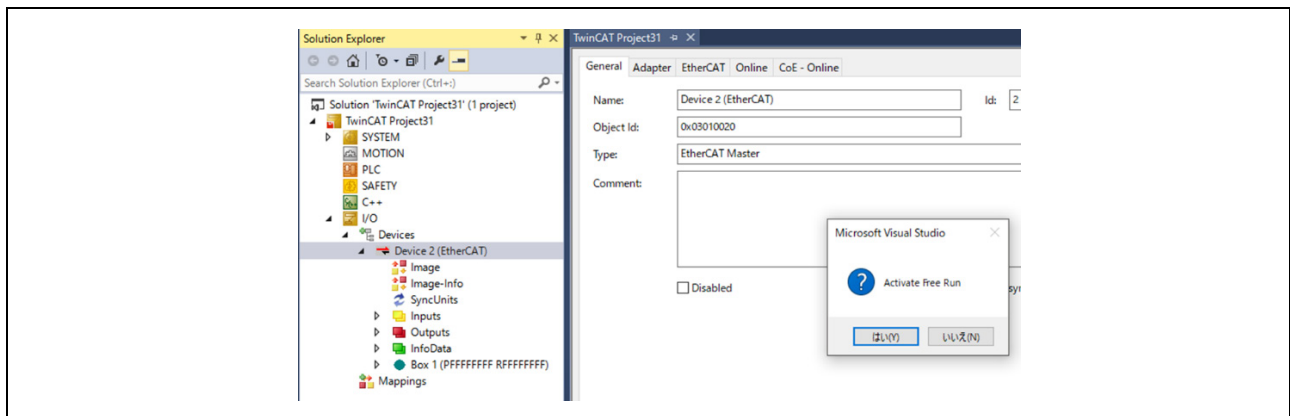
- (2) 次の手順で TwinCAT 用のドライバを追加します。(初回のみ)  
スタートメニューから[TWINCAT]→[Show Realtime Ethernet Compatible Device]を選択します。  
通信ポートの中から接続している Ether ポートを選択してインストールを押します。



- (3) 通信ポートの中から接続している Ether ポートを選択して、プロパティを表示させます。  
プロパティから[TwinCAT Ethernet Protocol for All Network Adapters]のみ有効にして閉じます。

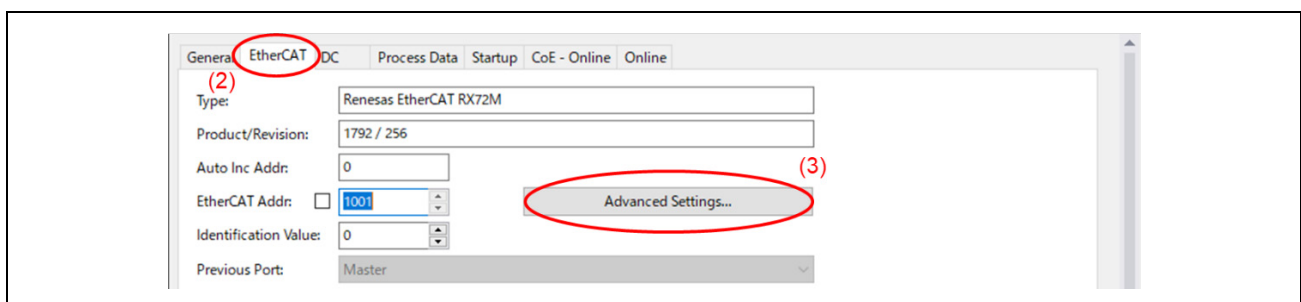


- (4) 評価ボードに LAN ケーブルを接続します。EtherCAT は In/Out が決められていますので、CN2 IN に接続してください。
- (5) スタートメニューから [Beckhoff] → [TwinCAT3] → [TwinCAT XAE (VS2013)] を選択、プログラムの起動後、[FILE] → [New] → [Project] を選択して、Templates の中の [TwinCAT XAE Project] を選択して新規プロジェクトを作成します。
- (6) ソリューションエクスプローラ → I/O → デバイス → 「スキャン」 を選択します。
- (7) [Scan for boxes] を実行すると、検出された Box1 のスレーブが Solution Explorer に現れます。ESI ファイルを認識できていない状態では、Box 1 (PFFFFFF RFFFFFF) と等で表示されます。ESI をダウンロードする必要がありますので、[Activate free Run] はいいえとしてください。

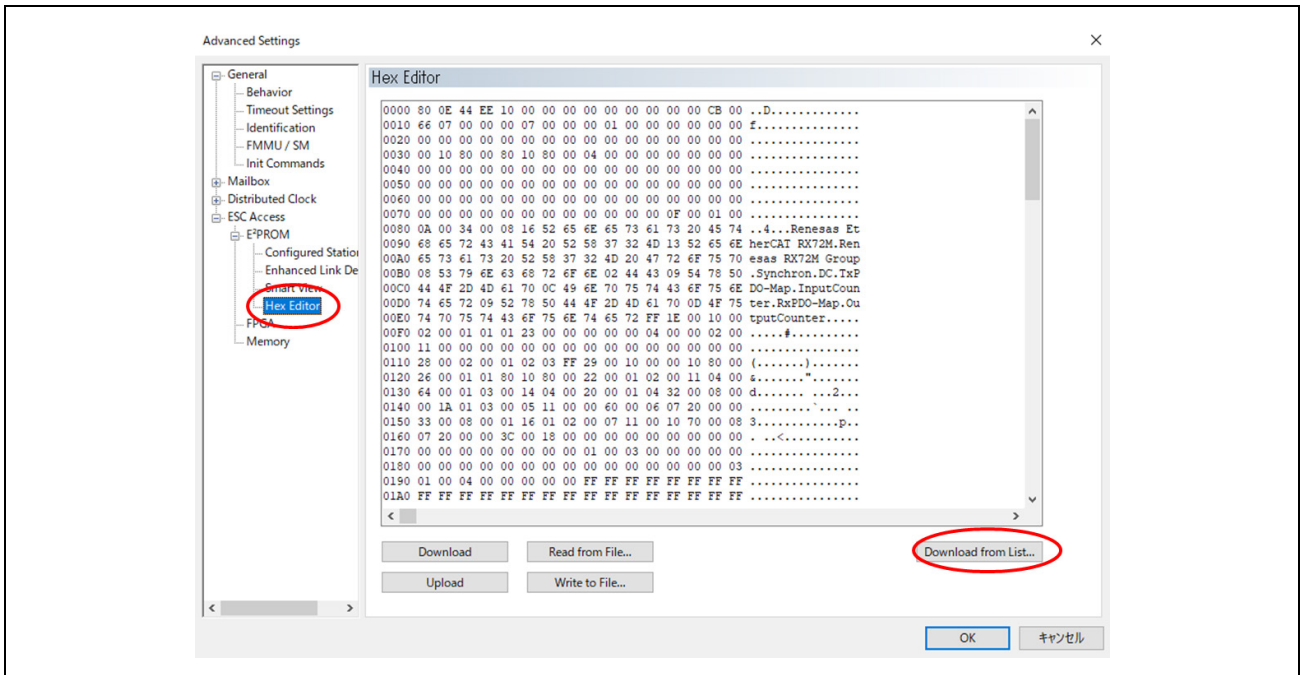


- (8) 別のアプリケーションのデータがすでに EEPROM に書き込まれている場合は、データを置き換えます。EEPROM 上のデータを置換する手順は以下の通りです。

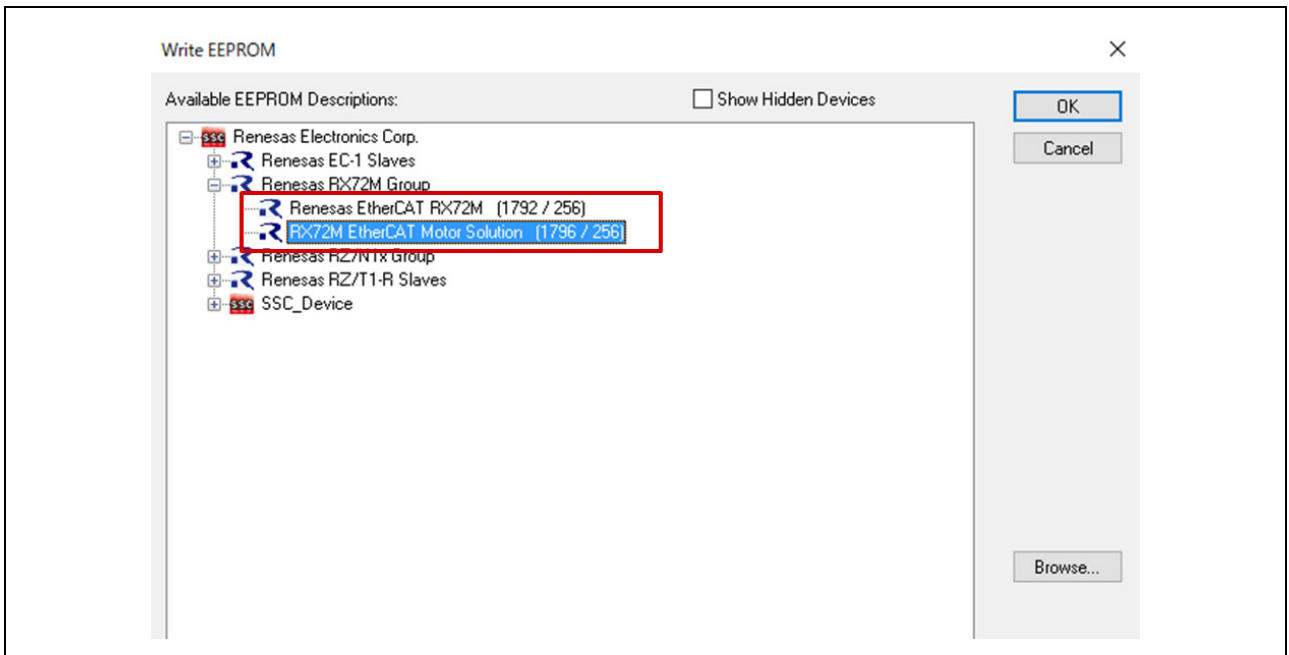
- [Box 1] をダブルクリックしてください。設定画面が表示されます。
- [EtherCAT] tab を選択してください。
- [Advanced Setting] ボタンを実行してください。



- (9) [ESC Access] → [EEPROM] → [Hex Editor]を選択してください  
 [Download from List] を選択してください。

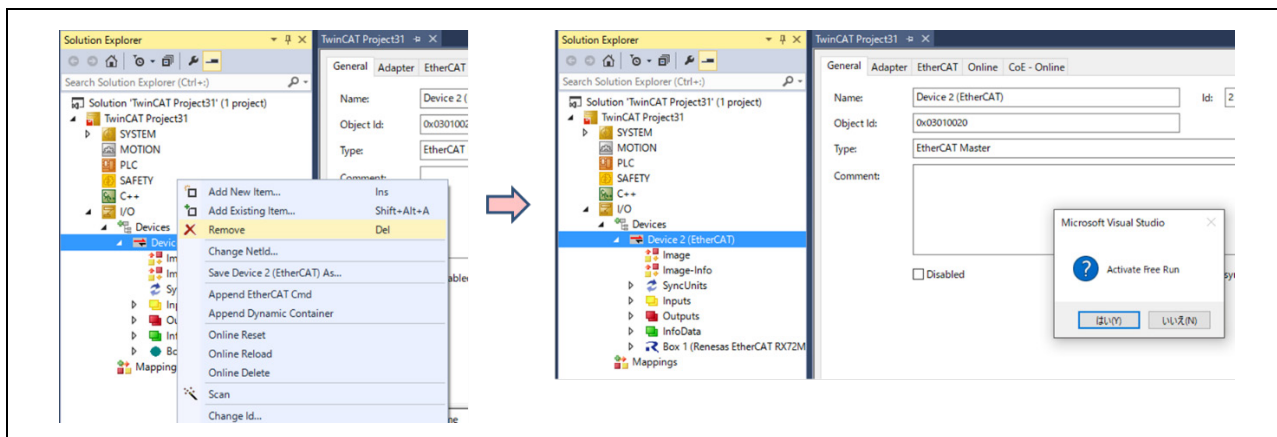


- (10) TwinCAT3 に登録してある ESI ファイルの一覧が現れますので、該当するファイルを選択してください。モータボードの場合は、[RX72M EtherCAT MotorSolution.xml]です。I/O ボードの場合は、[Renesas EtherCAT RX72M.xml]です。



- (11) DL した ESI ファイルの設定を反映させます。スレーブをリセットする必要がありますので、TwinCAT のネットワークから一旦スレーブを削除します。

スレーブをリセット後、再度スキャンをすると ESI ファイルが読み込まれていますので、Activate Free Run で実行してください。

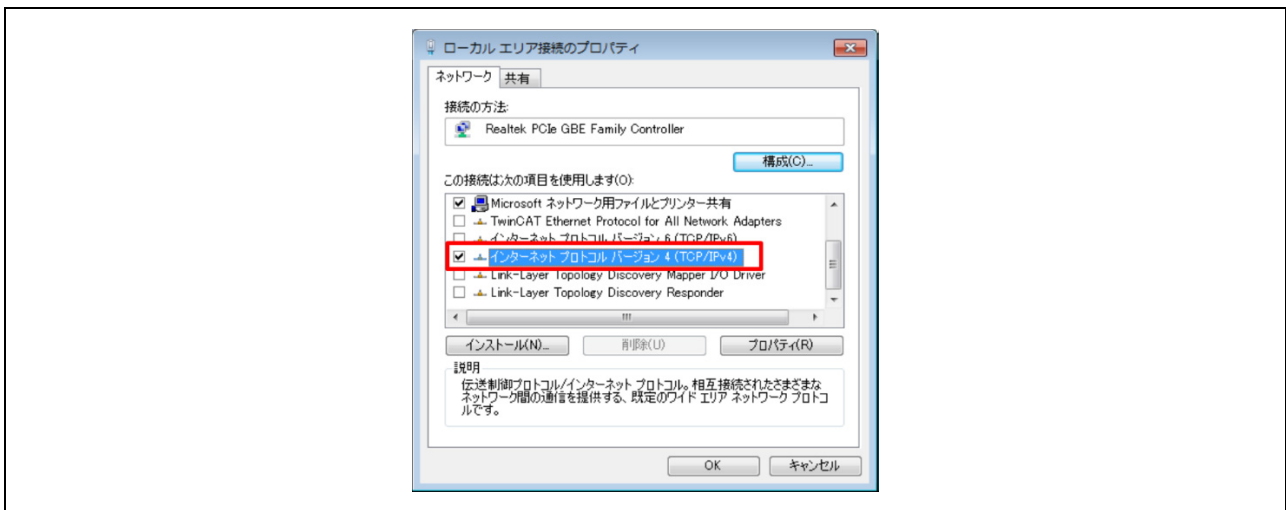


## 6.7 CODESYS との接続確認

本章ではサンプルプログラムをインストールした評価環境を、CODESYS Software PLC にて接続、動作させるための手順について説明します。

### 6.7.1 デバイスネットワークの設定

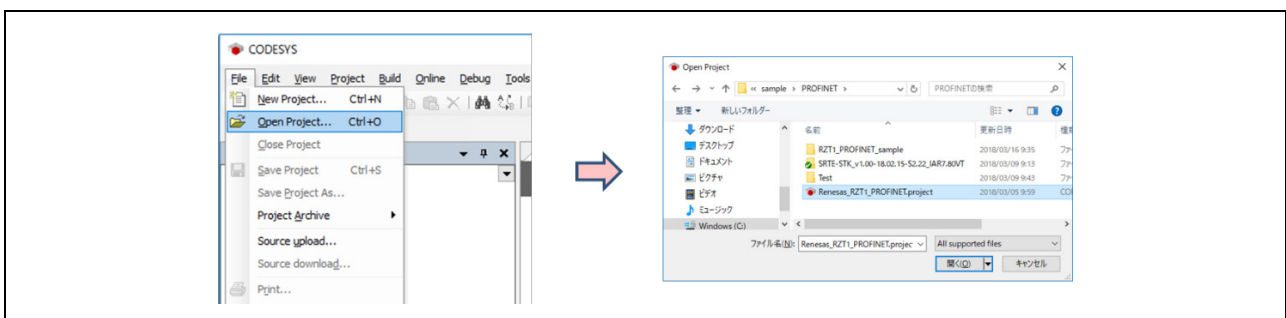
- (1) デバイス設定を行う前にホストの IP アドレス設定を行います。「ネットワーク設定」を開きます。
- (2) ローカルエリア接続をダブルクリック（もしくは右クリック）しプロパティを選択。
- (3) TCP/IPv4 を選択し、プロパティボタンをクリック。IP アドレス、サブネットマスクを設定。



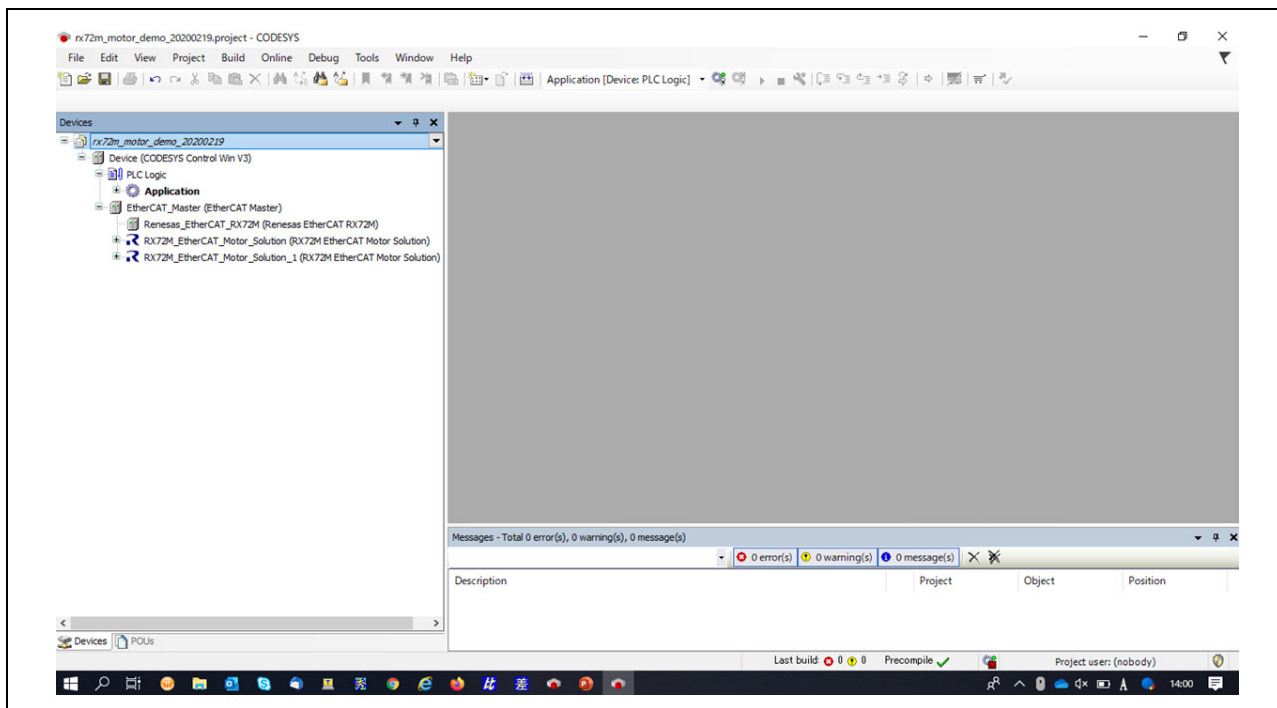
### 6.7.2 CODESYS の起動

- (1) Windows のスタートメニューからすべてのプログラム > CODESYS > CODESYS Gateway V3 またはインストール後にデスクトップに作成される CODESYS アイコンからも起動できます。
- (2) [File]→[Open Project ...]をクリックし” rx72m\_motor\_demo.project” ファイルを選択してプロジェクトを開きます。

\*CODESYS の新規プロジェクト構築についての手順及び UI 作成・確認手順については、「R-IN, RZ/T1, EC-1, TPS-1 グループ Software PLC Guide プロジェクト構築・UI 作成編」を参照してください

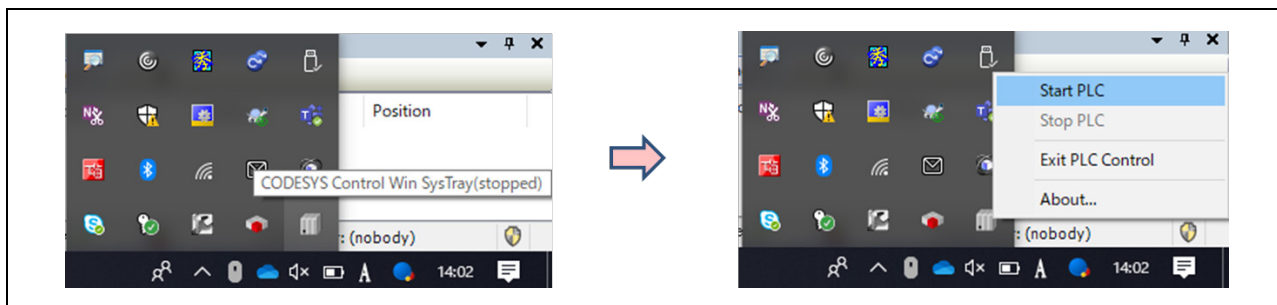


(3) プロジェクトが起動されると、デバイスツリーが表示されます。



### 6.7.3 PLC の起動

ソフト PLC の動作状況をシステムトレイで確認し、停止している場合はクリックし「Start PLC」を選択し起動操作を行なってください。ソフト PLC は通常 Windows 起動時にサービスとして自動起動します。デスクトップ右下のシステムトレイにあるアイコンが動作状況を示しています。



※システムトレイにアイコンがない場合

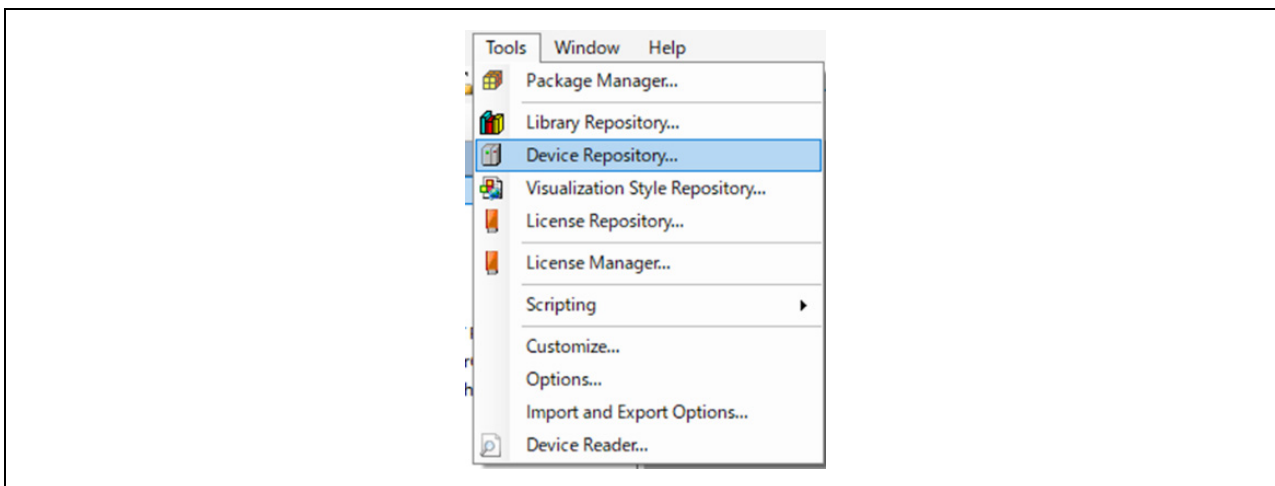
スタートメニューよりすべてのプログラム>CODESYS>CODESYS Gateway V3> CODESYS Gateway V3 を選択しゲートウェイサーバを起動してください。ゲートウェイサーバを起動しても、システムトレイにアイコンが表示されない場合はお使いの端末の再起動をお試しください。



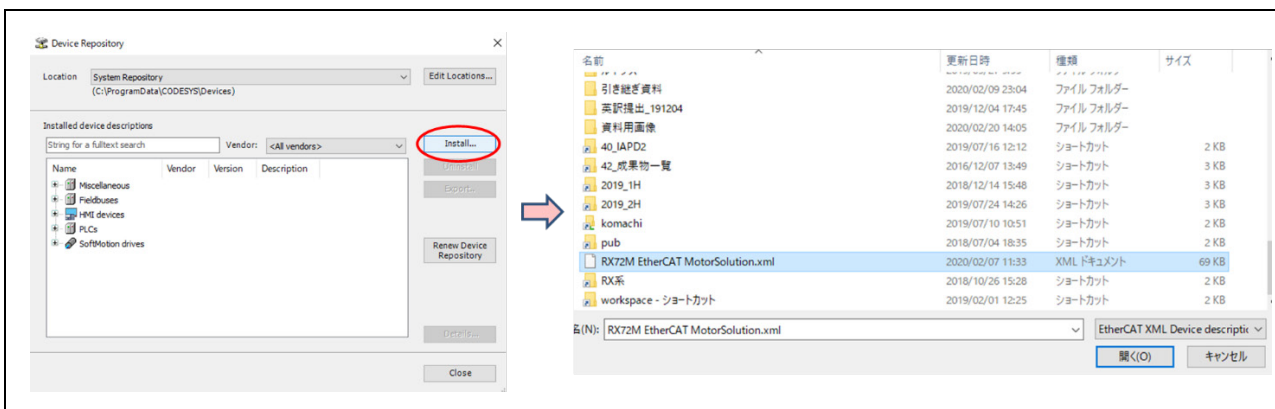
## 6.7.4 スレーブデバイスの更新

本章は、「rx72m\_motor\_demo.project」を初めて起動する時のみ行います。

- (1) EtherCAT スレーブデバイスを使用するには、デバイス情報が記述されている ESI ファイルのインストールが必要となります。ESI ファイルは、「Renesas EtherCAT RX72M.xml」と「RX72M EtherCAT MotorSolution.xml」をご利用ください。  
CODESYS 上の「ツール」メニューから「デバイスリポジトリ」を選択します。

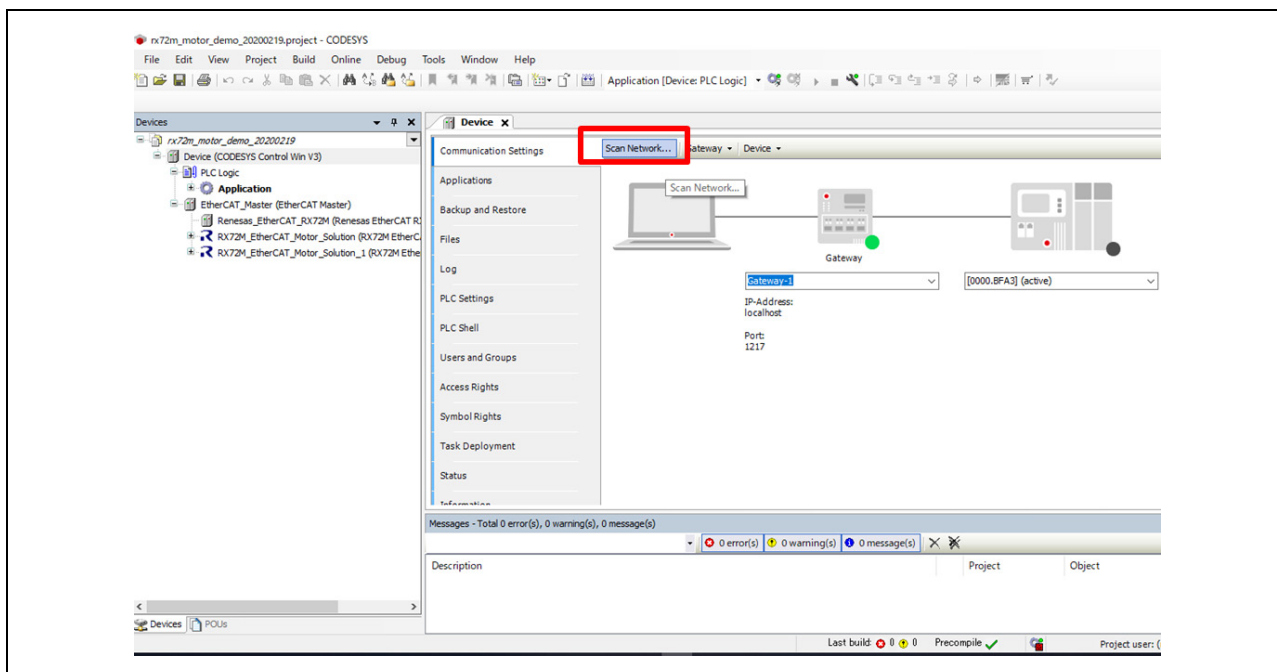


- (2) デバイスリポジトリダイアログにて、「インストール」をクリック、ファイルダイアログが表示されますので、ESI ファイル「RX72M EtherCAT MotorSolution.xml」を指定してください。

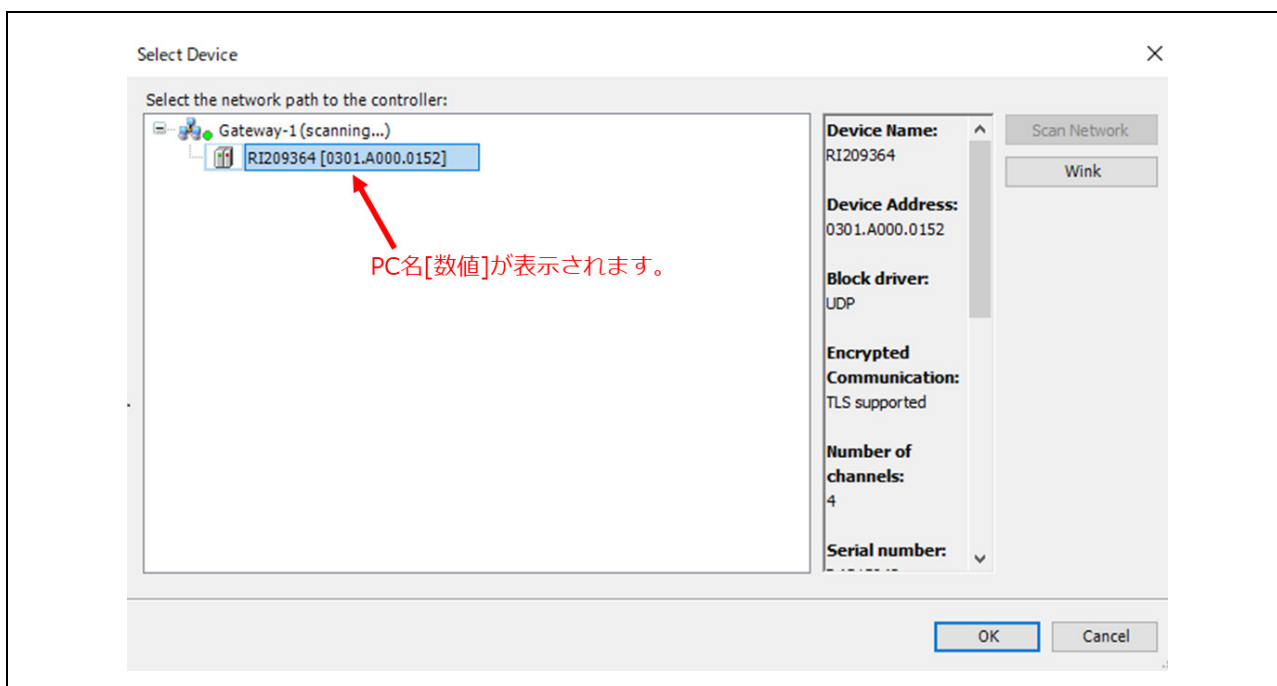


## 6.7.5 PLC との接続設定

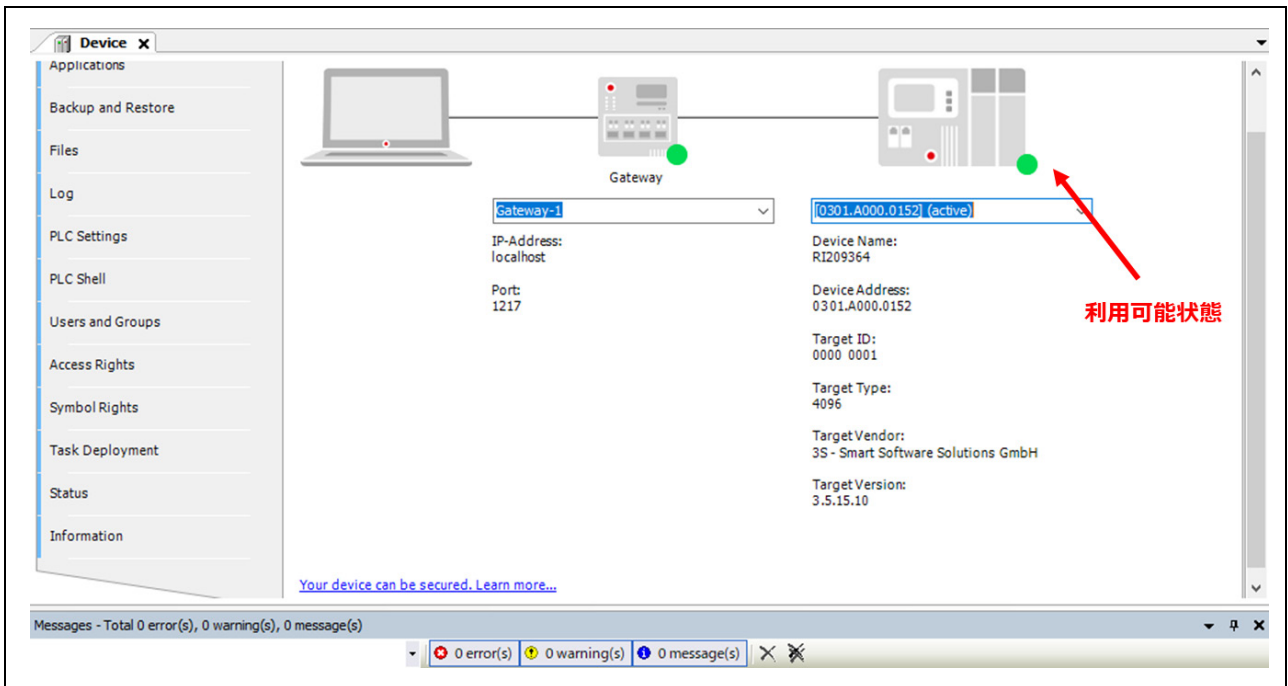
- (1) 「デバイス」ウィンドウのツリーから「Device (CODESYS Control Win V3)」をダブルクリックし、「通信設定」画面を開きます。この画面で開発環境からソフト PLC サービスへ接続するための通信設定をすることができます。  
「通信設定」タブの「Scan network…」ボタンをクリックしてください。



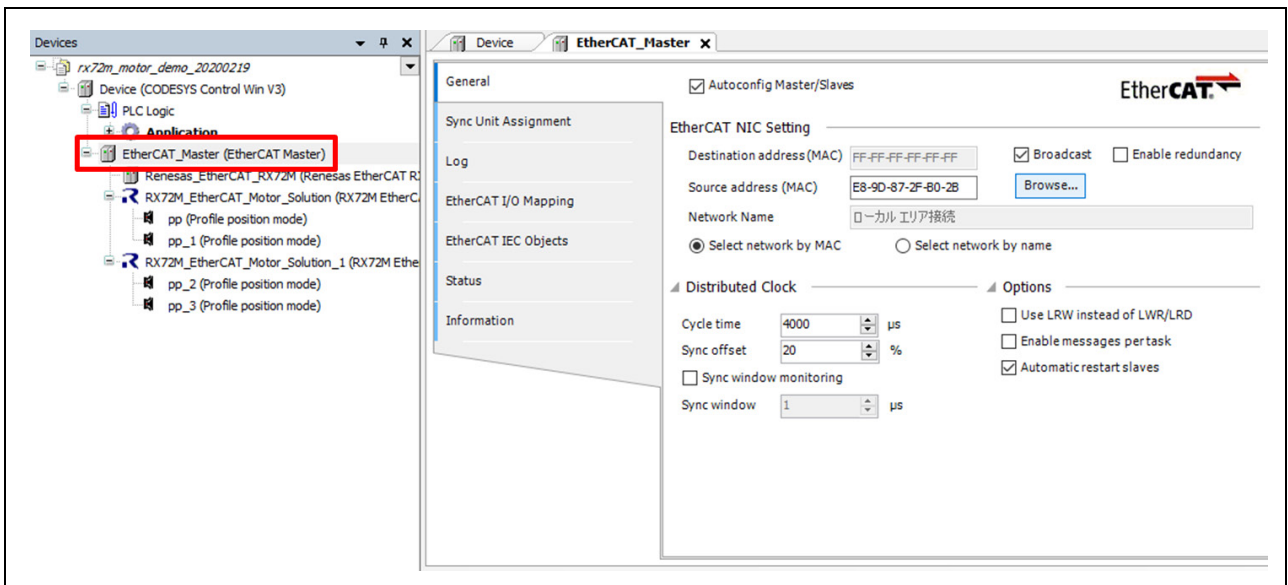
- (2) 「デバイスの選択」ウィンドウが表示され自動的にローカルネットワークの利用可能なデバイスが検索されます。ソフト PLC サービスが見つければ成功です。表示された PC 名をダブルクリックしてください。



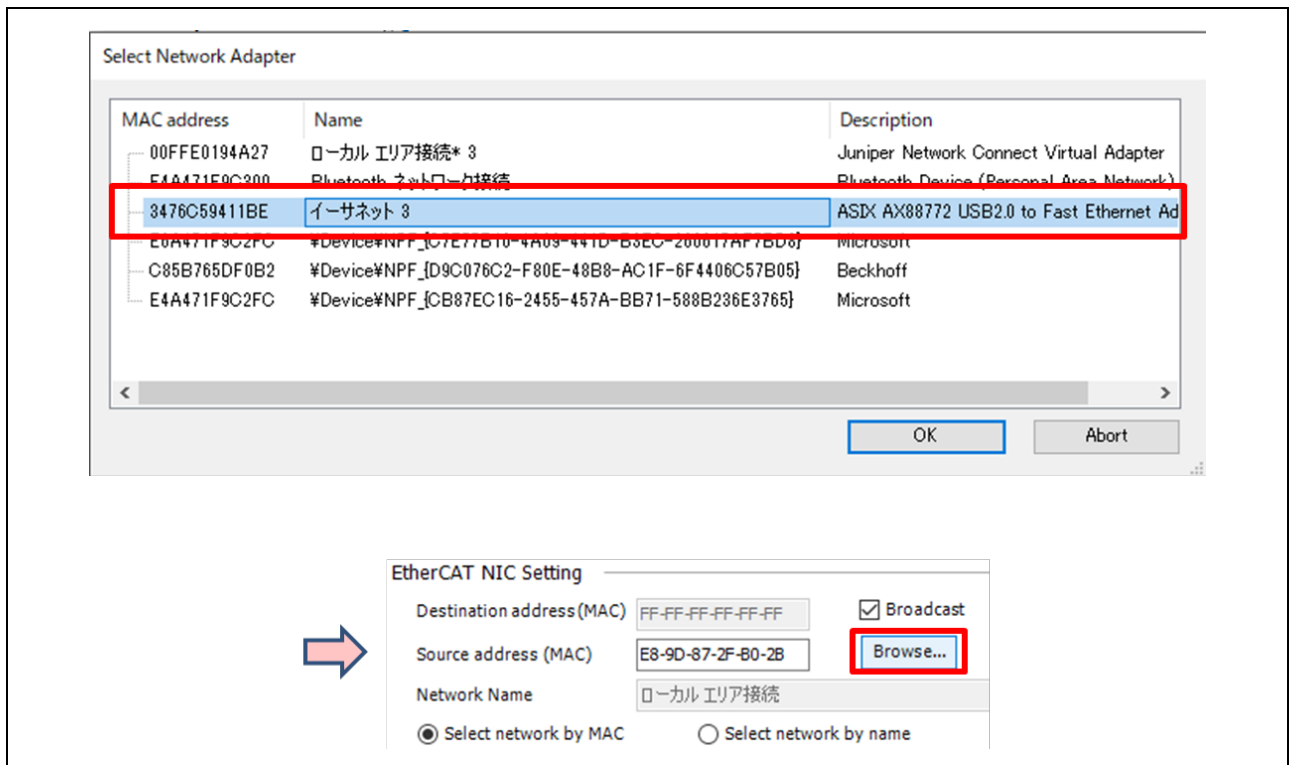
(3) スキャンが成功すると GateWay に PC が登録されます。



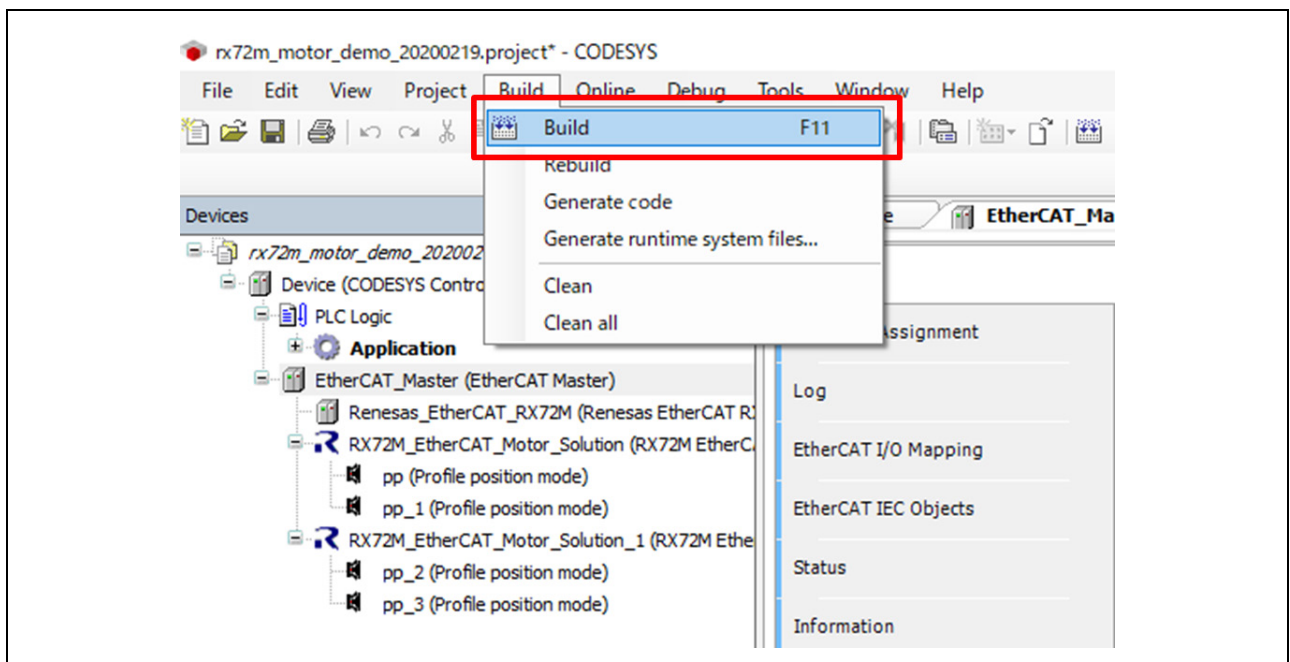
(4) 利用するネットワークの設定を行います。「EtherCAT\_Master」をダブルクリックして設定の General 画面を開いてください。



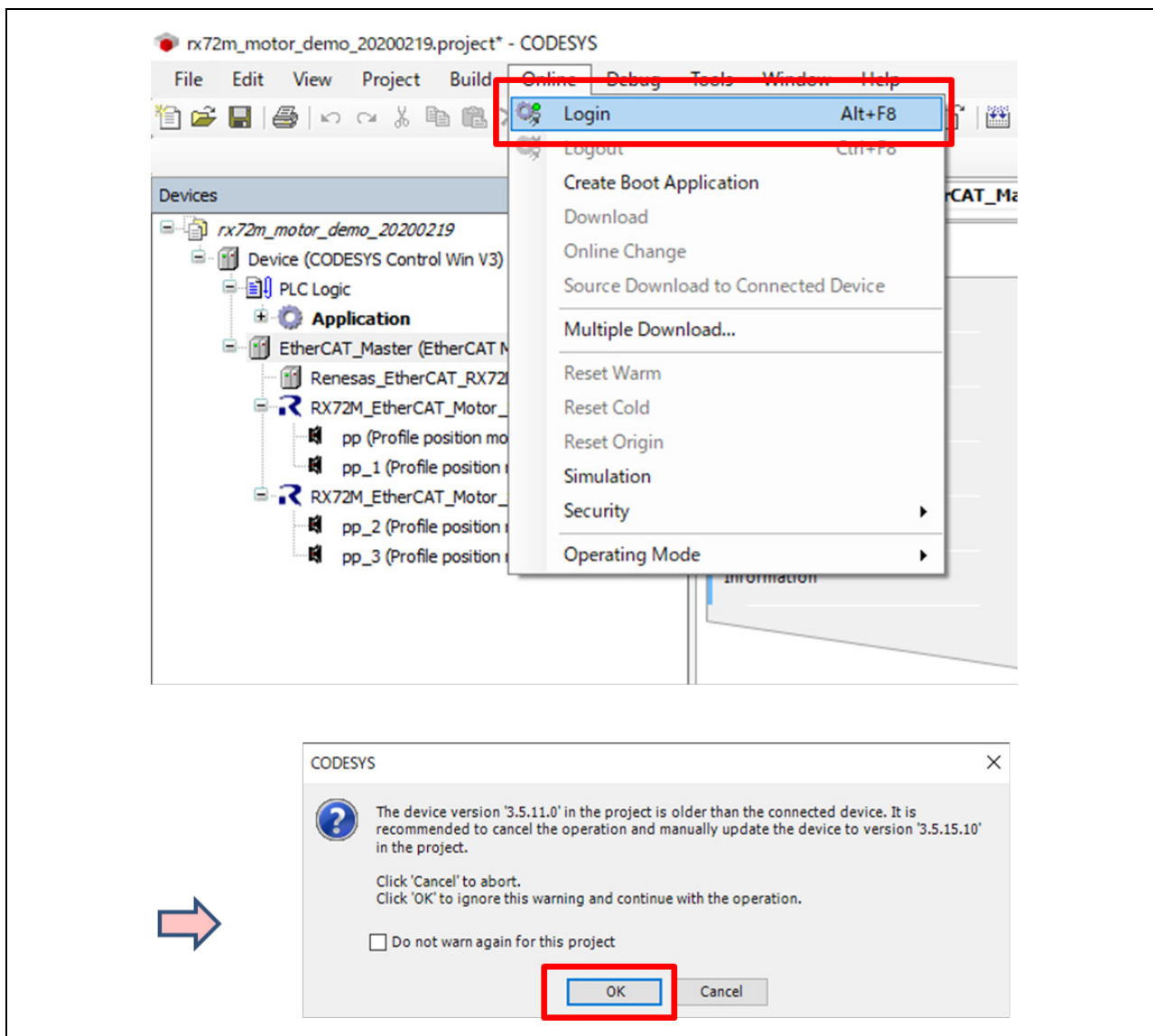
- (5) 利用するネットワークを選択してください。  
 選択したら、「EtherCAT\_Master」画面の「Browse」を押して、MACアドレスを確定してください。



- (6) プログラムのビルドを行います。メニューからビルドを選択します

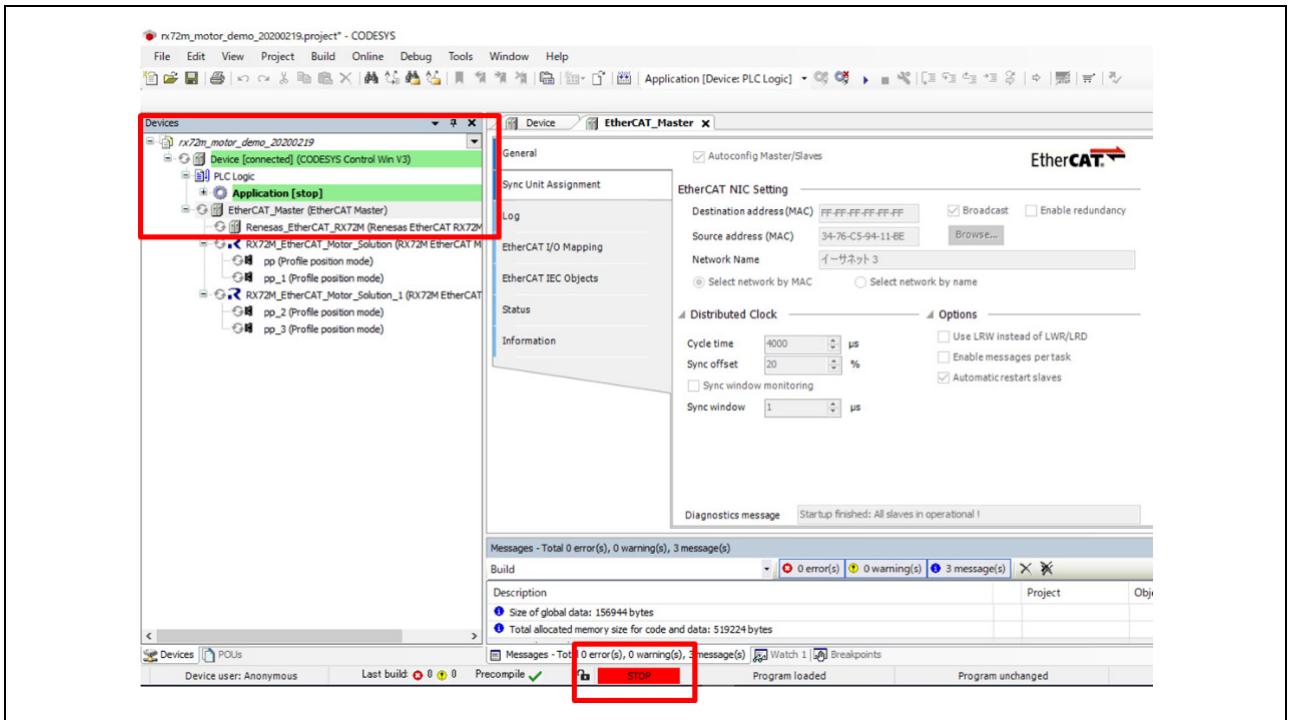


(7) ビルドが完了したら、ログインをします。メニューからログインを選択します。

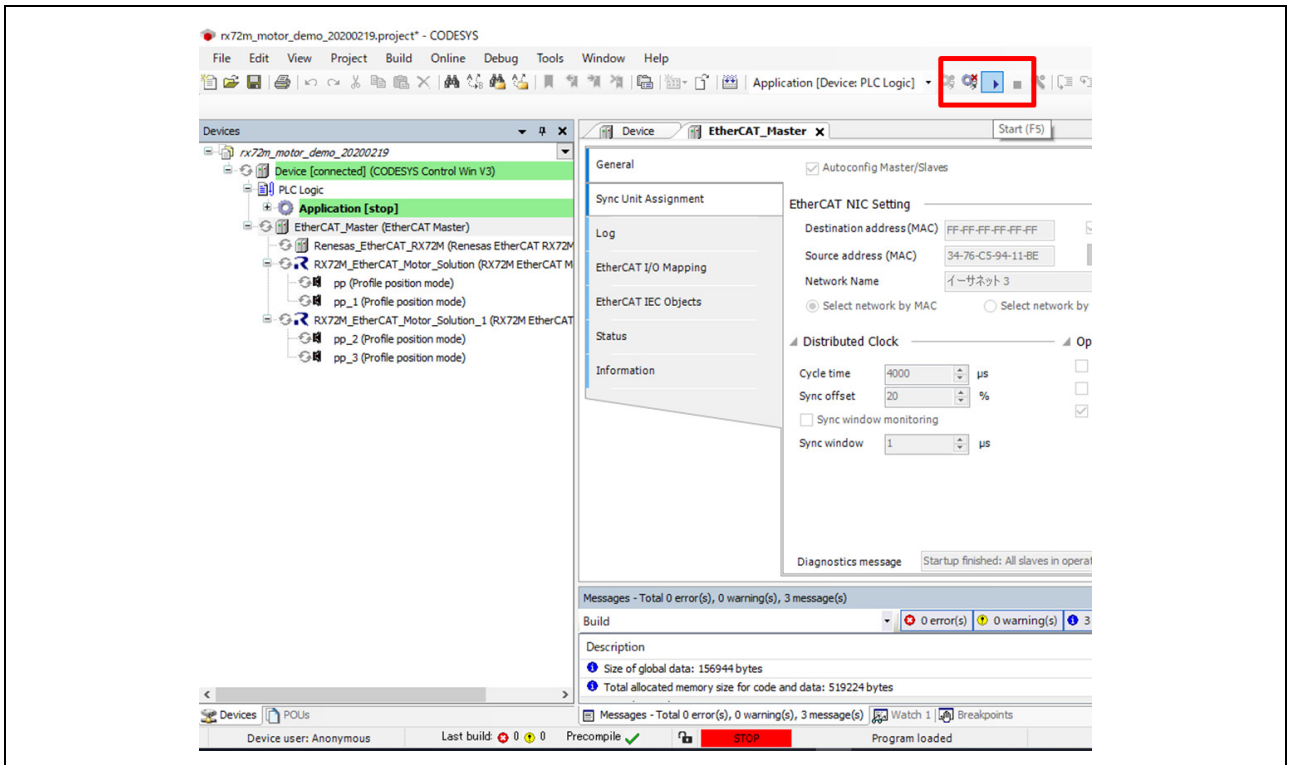


(8) ログインが正常に終了したら、STOP 状態で停止します。

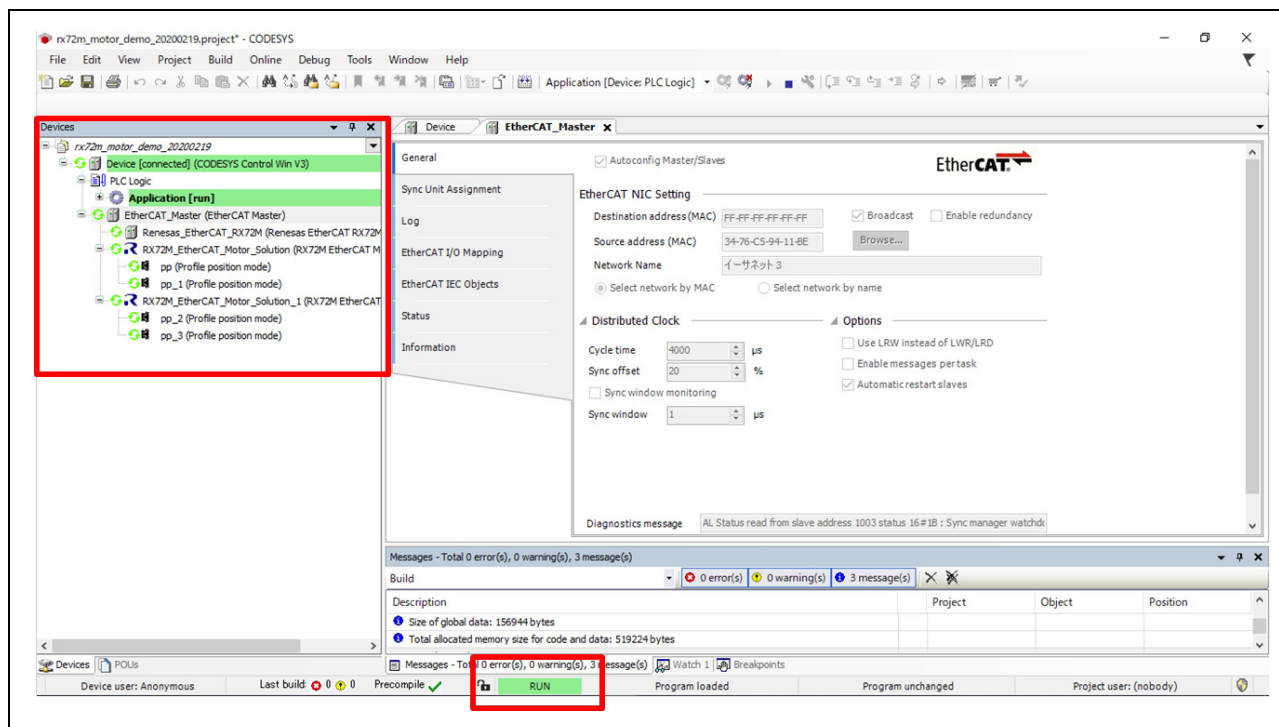
\*停止せずに RUN まで移行する場合があります。







(9) スタートします。ボタンを押して開始するか、ツールバーから運転を選択します。



(10) 正常に接続ができれば、下記の様に、ネットワークの接続と運転が開始されます。



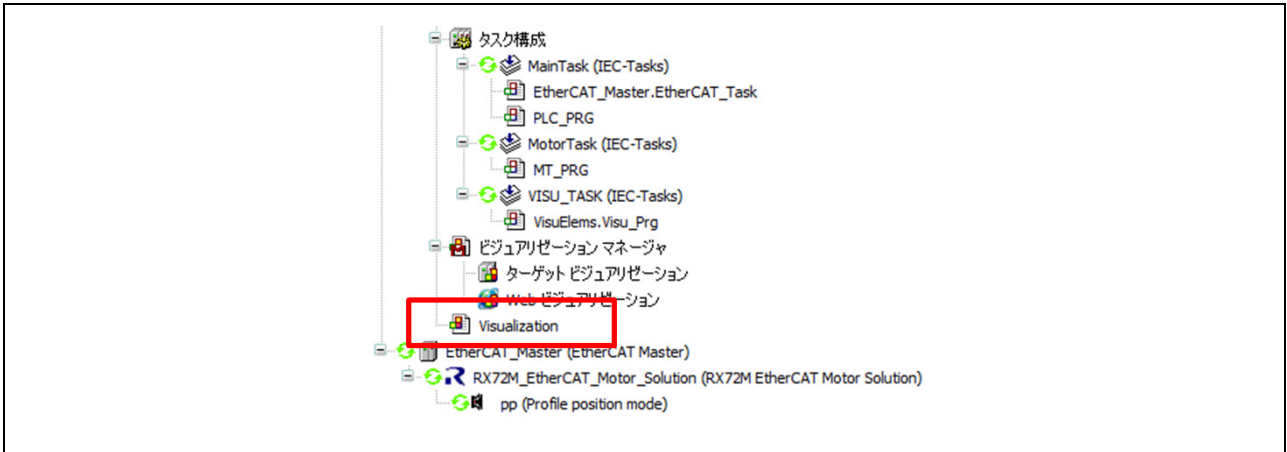
### デバイスステータス

-  : PLC が接続されていてアプリケーションが実行中の状態です。
-  : PLC が接続されていてアプリケーションが停止中の状態です。
-  : エラーとなっています。エラー内容及びデバイス設定内容をご確認ください。
-  : デバイス情報がデバイスリポジトリにありません。デバイス情報ファイルを見直した上で、インストールし直して下さい。

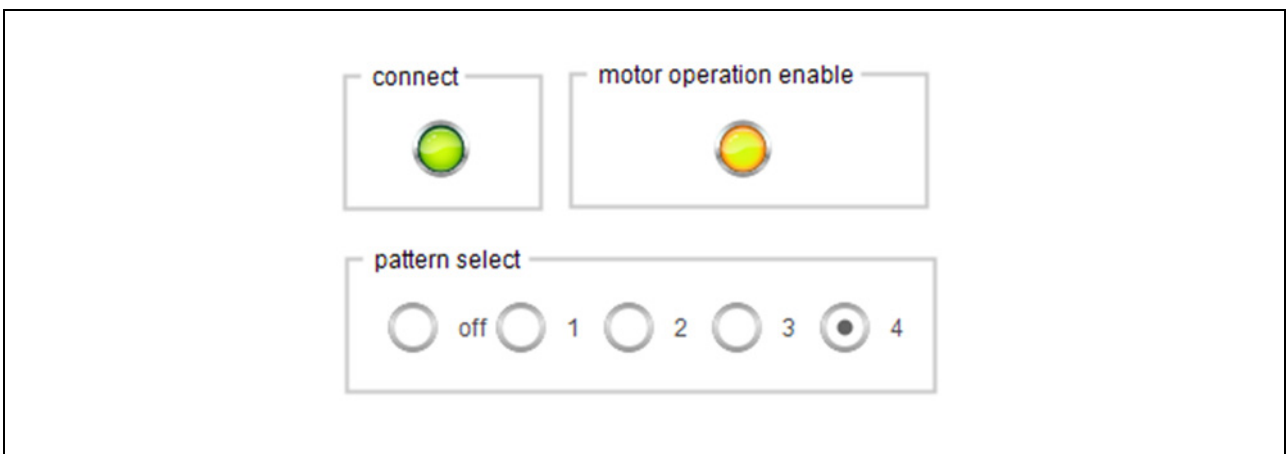
## 6.8 CODESYS を使った動作確認

“rx72m\_motor\_control.project”には付属のモータを駆動させるためのプログラムが構築されています。ネットワーク接続が完了の状態から、CODESYS に構築された GUI によりモータの制御を行います。

(1) デバイスツリーから“Visualization”を選択しダブルクリックします。



(2) モータ制御用の GUI 画面が起動します。



Pattern select によりモータの回転動作状態が変化します

- 1 : 1 回転を 90°、180°、360° に分割して繰り返し回転します。
- 2 : 1 の逆回転
- 3 : 10 回転、-10 回転を 1 セットで繰り返し回転します。
- 4 : -10 回転、10 回転を 1 セットで繰り返し回転します。

connect : 通信確立を表示します。  
 motor operation enable : command 遷移状態を表示します。



## 7. 参考ドキュメント

### ユーザーズマニュアル:ハードウェア

RX72M グループ ユーザーズマニュアル ハードウェア編 (ドキュメント No. R01UH0804)

Renesas Starter Kit+ for RX72M ユーザーズマニュアル (ドキュメント No. R20UT4383)

RX72M グループ 通信ボードハードウェアマニュアル (ドキュメント No. R01AN4661)

(最新版をルネサスエレクトロニクスホームページから入手してください。)

### スタートアップマニュアル

RX72M グループ RSK ボード EtherCAT スタートアップマニュアル (ドキュメント No. R01AN4689)

RX72M グループ通信ボード EtherCAT スタートアップマニュアル (ドキュメント No. R01AN4672)

(最新の情報をルネサスエレクトロニクスホームページから入手してください。)

### テクニカルアップデート/テクニカルニュース

(最新の情報をルネサスエレクトロニクスホームページから入手してください。)

### ユーザーズマニュアル:開発環境

RX ファミリー C/C++コンパイラ、アセンブラ、最適化リンカージェディタコンパイラパッケージ (R20UT0570)

(最新版をルネサスエレクトロニクスホームページから入手してください。)

## 8. APPENDIX

本モータボードは RMW(Renesas Motor Workbench)が使用できます。

**RMW(Renesas Motor Workbench)関連の手順**

RMW V2.0 を下記リンクからダウンロードする

<https://www.renesas.com/jp/ja/solutions/proposal/motor-control.html#kits>

RMW フォルダにある、RMW UM(r21uz0004jj0201-motor.pdf)表 3.1 に従い準備作業を実行する  
No 3.1, 3.5, 3.6, 3.7, 3.8

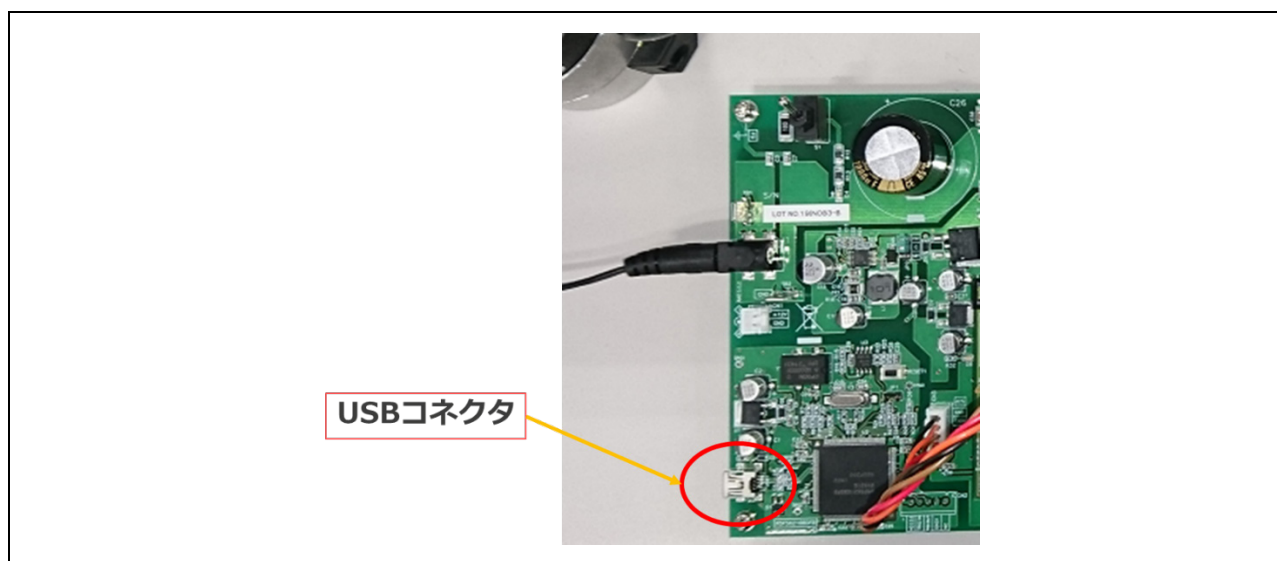
※認証ファイルは RMW のリンク場所にある「認証ファイルダウンロード」から行う。

表 3-1 準備作業の項目と利用方法

準備作業の項目		利用方法 (注 1, 注 2)		
No	章のタイトル	(a)	(b)	(c)
3.1	Renesas Motor Workbench をインストールする	○	○	○
3.2	ユーザプログラムへ通信ライブラリを組み込む	○	×	△
3.3	Map ファイルを生成する	○	×	△
3.4	ユーザプログラムをマイコンへ書き込む	○	○	○
3.5	Renesas Motor Workbench を起動する	○	○	○
3.6	Renesas Motor Workbench の認証の確認	○	○	○
3.7	Renesas Motor Workbench から通信するための準備	○	○	○
3.8	ボードとパソコンを接続	○	○	○
3.9	RMT ファイル (環境ファイル) に保存	○	○	○

(注 1) 記号説明 (○ : 作業が必要、△ : 作業は状況により必要、× : 作業は不要)

接続はインバータボードの USB1 を使用し、PC の USB ポートと接続してください

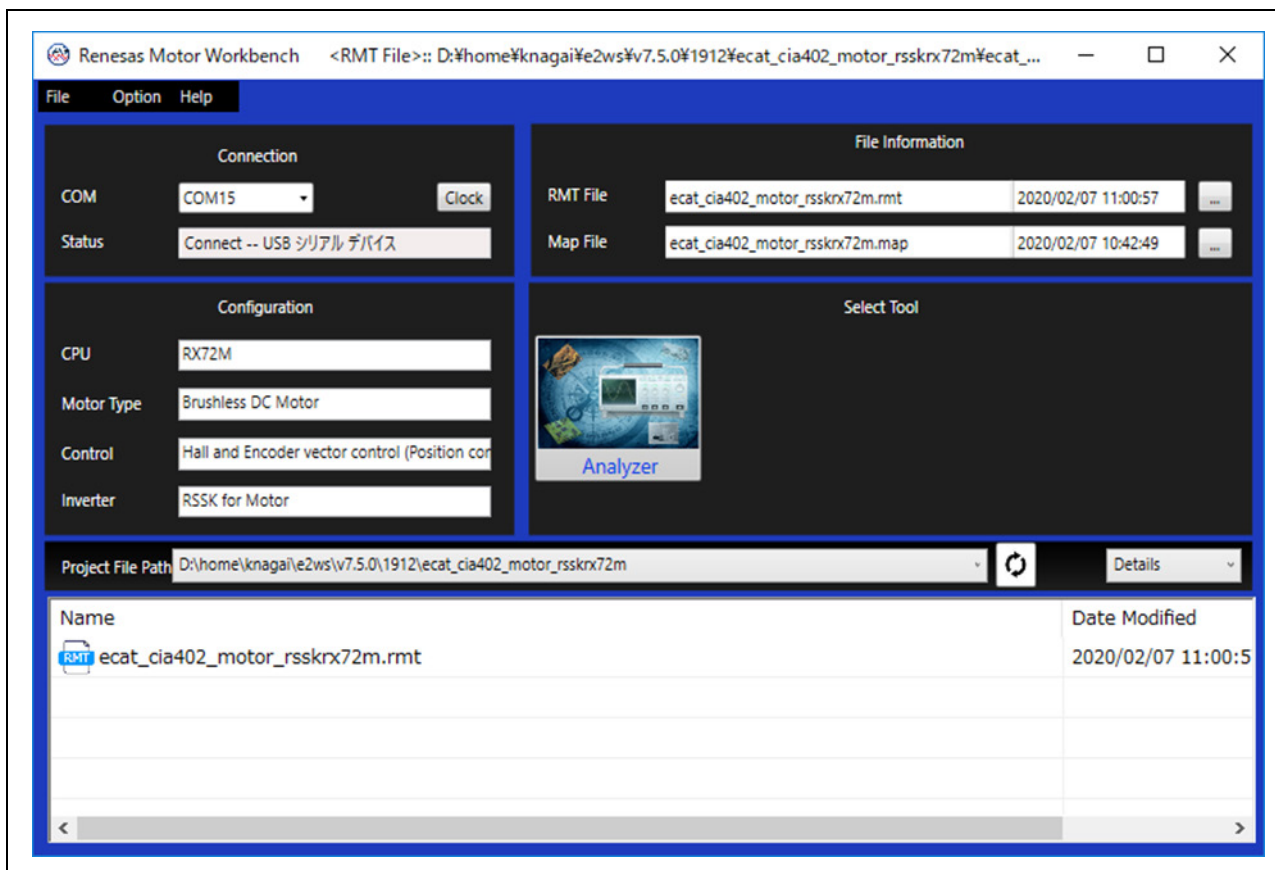


RMW を起動して次のファイルを指定します。

- 環境ファイル
  - ecat\_cia402\_motor\_rsskrx72m
  - ¥ecat\_cia402\_motor\_rsskrx72m.rmt
- map ファイル
  - ecat\_cia402\_motor\_rsskrx72m¥HardwareDebug
  - ¥ecat\_cia402\_motor\_rsskrx72m.map

RMW とモータ基板を接続する

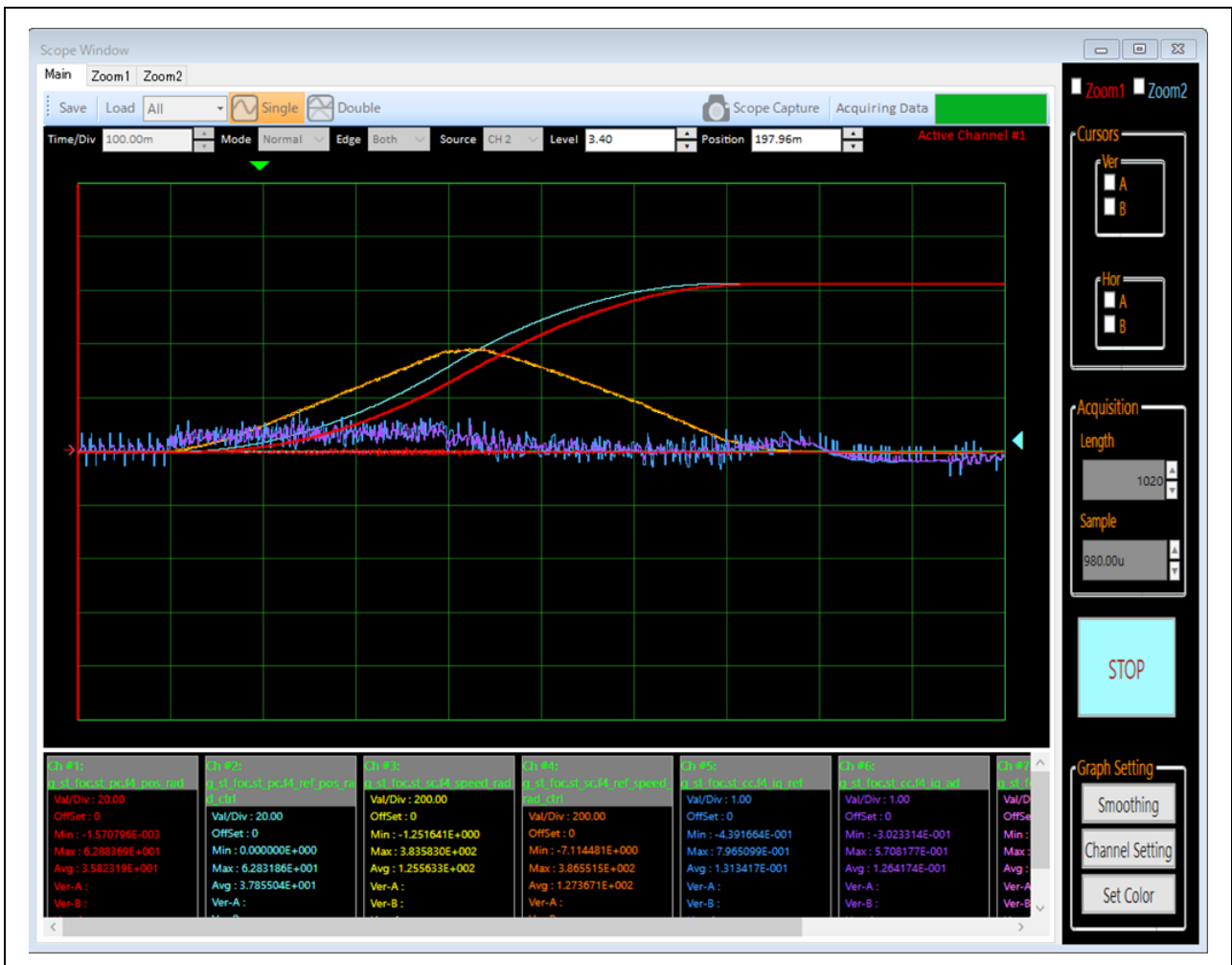
Connection→COM を指定し、正しく接続すると下記の様になる。



モータ駆動波形を取得する。

「Analyzer」 → 「Scope Window」 の「RUN」 ボタンを押す

\* 下記の例は Target Position を  
「0」 → 「40000」 に変更したときの波形。



## 改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2020/06/10	-	初版発行
1.10	2020/08/31	12	EtherCAT FIT モジュール Rev.1.10 対応によるフォルダ構成変更のため、2.3.1 表 2-7 を変更、表 2-8 を追加
		63	同じく、6.3 (6)のフォルダ名を変更

## 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しないでください。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えば、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、ノイズ輻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。

○Arm<sup>®</sup> およびCortex<sup>®</sup> は、Arm Limited（またはその子会社）のEUまたはその他の国における登録商標です。 All rights reserved.

○Ethernetおよびイーサネットは、富士ゼロックス株式会社の登録商標です。

○IEEEは、the Institute of Electrical and Electronics Engineers, Inc. の登録商標です。

○TRONは” The Real-time Operation system Nucleus” の略称です。

○ITRONは” Industrial TRON” の略称です。

○ $\mu$ ITRONは” Micro Industrial TRON” の略称です。

○TRON、ITRON、および $\mu$ ITRONは、特定の商品ないし商品群を指す名称ではありません。

○EtherCAT<sup>®</sup>、およびTwinCAT<sup>®</sup>は、ドイツBeckhoff Automation GmbHによりライセンスされた特許取得済み技術であり登録商標です。

○その他、本資料中の製品名やサービス名は全てそれぞれの所有者に属する商標または登録商標です。

## 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まれます。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）、交通管制（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、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なります。
  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](http://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/](http://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。