RX72M Group

Encoder BLDC motor control using EtherCAT Communications

R01AN6295EJ0110
Rev.1.10
Nov.30.2023

Outline

This application note describes a sample program for the RX72M. The program has an encoder vector control function for a permanent magnet synchronous motor (hereinafter referred to as a PMSM) and works with the EtherCAT communications controller of the RX72M. The module provides an interface via the EtherCAT Slave Stack Code (SSC) of Beckhoff, which is used in the RX family products that incorporate an EtherCAT slave controller (ESC) for industrial Ethernet communications.
The FIT module itself does not include the SSC. Therefore, generate the executable code after obtaining the sample SSC from the EtherCAT Technology Group (ETG Association).
This FIT module is hereinafter referred to as the EtherCAT FIT module.

Target Devices

- RX72M group devices

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation and testing of the modified program.

Contents

Description

1. Overview

1.1 This Application Note

This application note describes a sample program for the RX72M. The program has an encoder vector control function for a permanent magnet synchronous motor (hereinafter referred to as a PMSM) and works with the EtherCAT communications controller of the RX72M.
The sample program is intended to run on a combination of a board with an RX72M CPU and inverter board.

1.2 Operation Environment

Table 1-1 Operation Environment

Target MCU	RX72M Group
Evaluation board	Manufactured by Renesas RX72M CPU card + inverter board *
Integrated development environment (IDE)	Renesas e2 studio, 2023-10
C compiler	Renesas C/C++ compiler package for RX Family V3.05.00 or later
Motor	BLDC Motor : BLY171D-24V-4000 (Anaheim Automation) Encoder : AMT102-V (CUI DEVICES)
Emulator	Renesas e2 Lite
Communication protocol	EtherCAT
SSC tool	Provided by the EtherCAT Technology Group (ETG) Slave Stack Code (SSC) tool Version 5.13
Software PLC	TwinCAT® 3 (download this from the Beckhoff web site) of Beckhoff Automation

Note: * It is included in the "Evaluation System for BLDC Motor (RTK0EMX270S00020BJ)" manufactured by Renesas Electronics.

1.3 Projects

The sample program realizes single-chip motor control via EtherCAT communications. It was prepared by modifying other projects for motor control and EtherCAT communications.

Table 1-2 Base Projects and Changes that were Required

Function/Project Name (Application note)	Changes
Motor control RX72M_ESS_STM_RSLV_FOC_E2S_RV100	● API functions were added for control of the motor by the EtherCAT communications program. ● The units of position and velocity were converted to conform with the CiA402 object specifications.
EtherCAT communications rx72m_com_cia402 (r01an4672ej0104-rx72m-ecat)	● Objects were added to fit the CiA402 drive profile. ● Calls of API functions to control the motor

The project included for the sample program is shown below.
In the following sections, the RX72M CPU card plus inverter board project is used as an example. When using a different project, read the project name in this application note as that of the given project.

Table 1-3 List of Projects

MCU	Evaluation Board	Project Name
RX72M	RX72M CPU card + inverter board	rx72m_ecat_cia402_bldc_encd

2. System Overview

2.1 Hardware Configuration

The following figure shows the hardware configuration of the environment where the sample program runs.

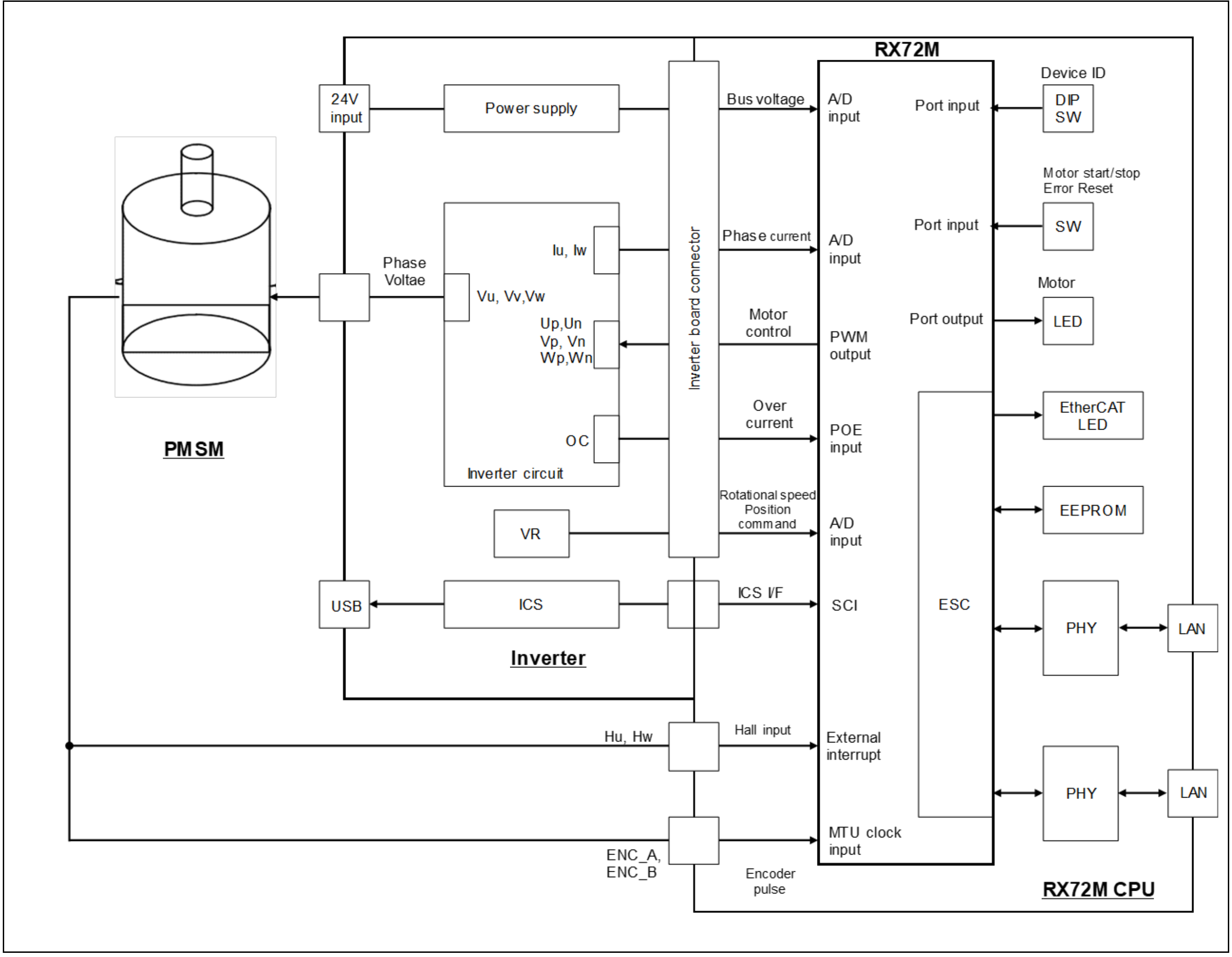


Figure 2-1 Hardware Configuration

2.2 Hardware Specifications

Table 2-1 to Table 2-4 list the pin interfaces for use in the sample program.

Table 2-1 Motor Control Related Pin Interface

Pin Name	Description
P40 /AN000	U-phase current measurement
P42 /AN002	W phase current measurement
P23/GTIOC0A	PWM output (Up)
P17/GTIOC0B	PWM output (Un)
P22/GTIOC1A	PWM output (Vp)
P87/GTIOC1B	PWM output (Vn)
P21/GTIOC2A	PWM output (Wp)
P86/GTIOC2B	PWM output (Wn)
P00/AN118	Inverter bus voltage measurement
P33 / IRQ3	Hall U phase input
P32 / IRQ2	Hall V phase input
P31 / IRQ1	Hall W phase input
P24 /MTCLKA	Encoder A phase input
P25 /MTCLKB	Encoder B phase input
PC4/GTETRGC	PWM emergency stop input when overcurrent is detected
PC5	SW1 input
PC3	SW2 input
P01/AN119	VR input
P80	LED1 control
PK2	LED2 control
P76	LED3 control

Table 2-2 EtherCAT Communications Related Pin Interface (1)

Pin Name	Description
PK6/CATLINKACT0	Link / Activity LED control output
PK7/CATLINKACT1	Link / Activity LED control output
PH1/CATI2CCLK	EEPROM I2C clock output
P15/CATLEDRUN	RUN LED (green LED) control output
PH3/CATLEDERR	ERR LED (red LED) control output
PL3/CAT0_RX_CLK	Receive clock input
PM4/CAT0_ETXD2	4-bit transmission data output (bit2)
PM5/CAT0_ETXD3	4-bit transmission data output (bit3)
PL4/CAT0_ETXD0	4-bit transmission data output (bit0)
PL5/CAT0_ETXD1	4-bit transmission data output (bit1)
PK5/CAT0_ERXD3	4-bit transmission data input (bit3)
PK4/CAT0_ERXD2	4-bit transmission data input (bit2)
P74/CAT0_ERXD1	4-bit transmission data input (bit1)
P75/CAT0_ERXD0	4-bit transmission data input (bit0)
PL7/CAT0_MDIO	Management data I / O input / output
PN3/CAT1_RX_ER	Receive error input
P84/CAT1_LINKSTA	Link status input from PHY-LSI
PQ2/CAT1_RX_DV	Received data valid input
PL6/CAT0_TX_EN	Send enable output
PN2/CAT1_TX_CLK	Transmission clock input
PH4/CATLEDSTER	RUN LED control (off at ERR) output for STATE LED (2-color LED)
PH5/CATLATCH0	LATCH signal input
PH6/CATLATCH1	LATCH signal input
PA4/CATIRQ	IRQ output
PQ7/CAT1_TX_EN	Send enable output
PC2/CAT0_RX_DV	Received data valid input
PM1/CAT1_ERXD1	4-bit received data input (bit1)
PM2/CAT1_ERXD2	4-bit received data input (bit2)
PM3/CAT1_ERXD3	4-bit received data input (bit3)
PL2/CAT0_RX_ER	Receive error input
PM0/CAT1_ERXD0	4-bit received data input (bit0)
PQ4/CAT1_RX_CLK	Receive clock input
PJ5/CATSYNC0	SYNC0 signal output
PA6/CATRESTOUT	PHY reset signal output

Table 2-3 EtherCAT Communications Related Pin Interface (2)

Pin Name	Description
PN1/CAT1_ETXD3	4-bit transmission data output (bit3)
PQ5/CAT1_ETXD0	4-bit transmission data output (bit0)
PN0/CAT1_ETXD2	4-bit transmission data output (bit2)
PQ6/CAT1_ETXD1	4-bit transmission data output (bit1)
P11/CATSYNC1	SYNC1 signal output
PM6/CAT0_TX_CLK	Transmission clock input
PK0/CAT0_MDC	Management data clock output
P34/CAT0_LINKSTA	Link status input from PHY-LSI
PH2/CATI2CDATA	EEPROM I2C data input / output

Table 2-4 Other Pin Interface

Pin Name	Description
P72	Device ID DIP SW (bit3)
PC1	Device ID DIP SW (bit4)
PN5	Device ID DIP SW (bit5)

2.3 Software Configuration

2.3.1 Software File Configuration

Folders and files configured for the sample program are listed in Table 2-5.
The files in gray-shaded cells are those which required changes from the base project to implement the functionality of the sample program. The files listed in bold letters are those that have been added.

Table 2-5 Software file structure

Directory motor/			File	Description
ecat/	application/	renesas	cia402sample.h cia402sample.c	CiA402 application definition
		Beckhoff/ Src	SSC source file	Stored after applying the batch file
	interface/		**r_mtr_driver_ecat_access.h** **r_mtr_driver_ecat_access.c**	Common definition
app/	cfg/		r_app_control_cfg.h	Definitions of units
	main/		r_app_main.h r_app_main.c	Definitions of various parameters

2.3.2 Software Module Configuration

Modules that perform motor control via EtherCAT communication are located in the Application Layer and Middle Layer.
This section describes the structure and roles of the modules in each layer.

Table 2-6 Changes According to the Module Layer

Layer / Module	Related File	Description of File
Application layer / EtherCAT application	cia402sample.h cia402sample.c	Application layer for the EtherCAT communications program. This has the functions of passing commands from the EtherCAT master to the motor control program and passing indicators of state from the motor control program to the EtherCAT master.
Middle layer / EtherCAT interface module	r_mtr_dirver_ecat_acces.h r_mtr_dirver_ecat_acces.c	API functions for interfacing the motor control program and the EtherCAT communication program, providing access to EtherCAT communication-specific motor control program variables as APIs.

Figure 2-3 shows the module configuration of the EtherCAT communications program.

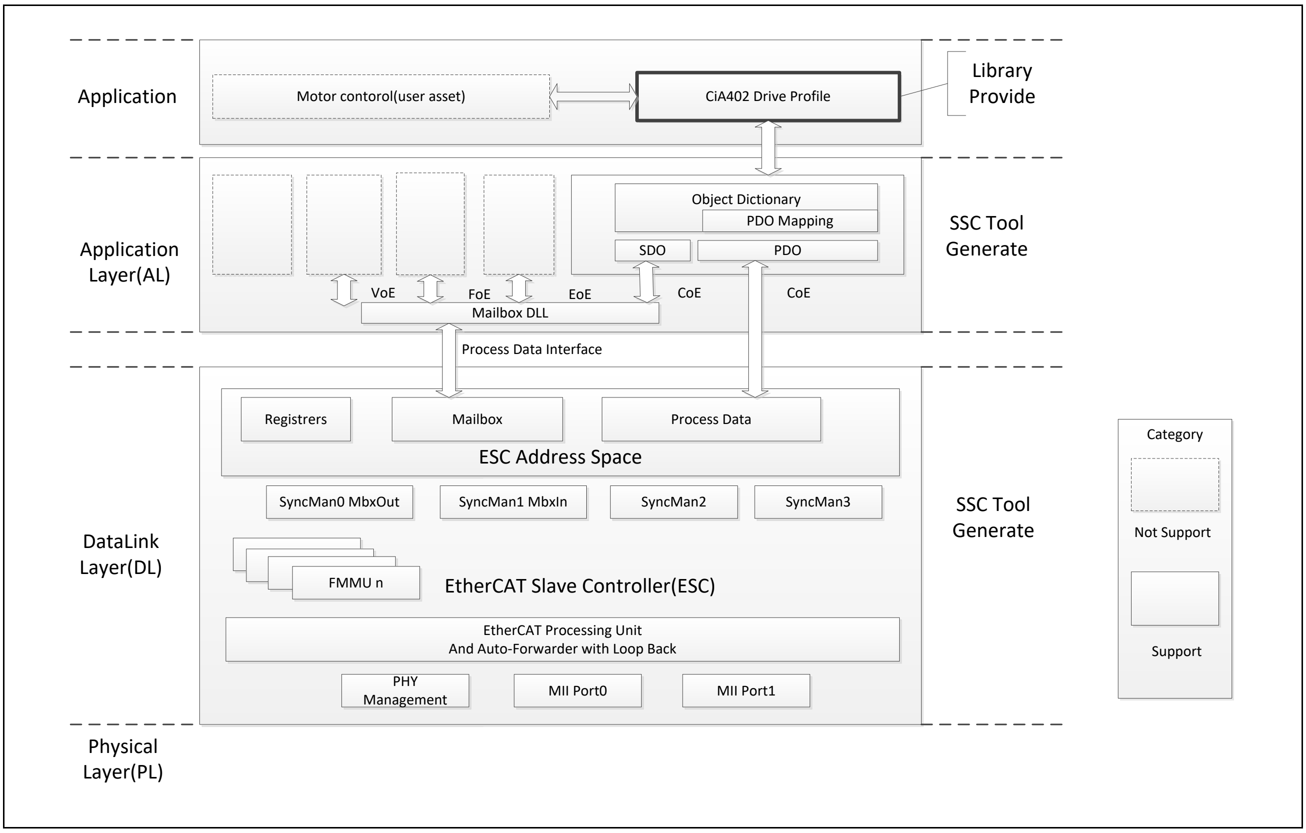


Figure 2-2 Module Configuration of the EtherCAT Communications Program

2.4 Software Specifications

The basic software specifications of the sample program are listed below.

Table 2-7 Basic Specifications of the Motor Control Program

Item	Description
Control method	Vector control
Rotor pole position detection	Incremental encoder (A phase, B phase), Hall sensor (UVW phase)
Input voltage	DC 24 V
Carrier frequency (PWM)	20 [kHz] (carrier period: 50 [μs])
Dead time	2 [μs]
Control period (current)	50 [μs]
Control period (velocity, position)	500 [μs]
Range of position command values	Creation of position command value: Position profile by velocity trapezoidal wave method (Input range) -32768°～32767° (Speed limit) CW / CCW：-4000~4000 [rpm]
Range of velocity command values	CW: 0 [rpm] to 4000 [rpm] CCW: 0 [rpm] to 4000 [rpm]
Position resolution	0.09 ° (encoder pulse: 1000 [p/r], 4000 [cpr] when multiplied by 4)
Positional dead zone *1	±1 count
Frequencies specific to the control systems	Current control system: 300 Hz Velocity control system: 30 Hz Position control system: 10 Hz
Compiler optimization settings	Optimization level：2 (-optimize = 2) (default setting) Optimization method：Code size-focused optimization (-size) (default setting)
Protection stop processing	Deactivate the motor control signal output (6 lines) under any of the following conditions. 1. Current in each phase exceeds 3.82 [A] (monitored every 50 [μs]) 2. Inverter bus voltage exceeds 28 [V] (monitored every 50 [μs]) 3. Inverter bus voltage less than 14 [V] (monitored every 50 [μs]) 4. Rotation speed exceeds 4500 [rpm] (monitored every 50 [μs]) 5. Hall sensor pattern error (at startup) When an external overcurrent detection signal (POE/POEG) and output short circuit are detected, set the PWM output terminal to high impedance.

Note 1. The dead zone is provided to prevent hunting when deciding the position.

Table 2-8 Basic Specifications of the EtherCAT Communications Program

Item	Description
Physical layer	100 BASE-TX (IEEE802.3)
Baud rate	100 [Mbps] (full duplex)
Number of communications ports	2
EtherCAT LED	RUN, ERR, STAT, L/A IN, or L/A OUT
Station ID	Specified by the device ID DIP switch block (6 bits)
Explicit device ID	Supported
Device profile	CiA402 device profile
Sync manager	4
FMMU	3
Communications objects	SDO (service data object) PDO (process data object)
Synchronous mode	SM2 event synchronous mode DC mode
Form of providing the protocol stack	The SSC tool project files for the sample program are provided. A patch for the CiA402 application is also provided. The EtherCAT communications program can be created by applying the patch after the protocol stack code has been generated by using the SSC tool.

3. CiA402 Drive Profile

The CiA402 drive profile is the device profile for drivers and motion controllers and mainly defines functional operations for servo drives, sine wave inverter, and stepping motor controller. In this profile, the multiple operating modes and corresponding parameters are defined as an object dictionary. Moreover, the finite state automaton (FSA) to define the internal and external behavior in every state is included. To change the status, set the control word object, then status word which shows the current status reflects the result after transition. The control word and various command values (such as for velocity) are assigned to RxPDO, and the status word and various actual values (such as for position) are assigned to TxPDO. For details, refer to the CiA402 Specifications.

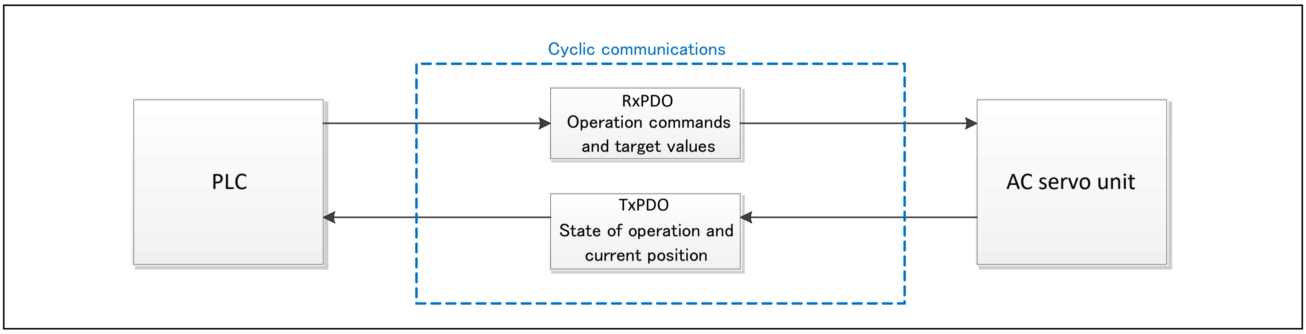


Figure 3-1 Flow of CiA402 Communications

3.1 Operating Mode

Among the operating modes specified by the CiA402 standard, the sample program supports the following modes.

Table 3-1 List of Supported Operating Modes

Operating Mode	Support
Profile position mode	Available
Velocity mode (frequency converter)	Not available
Profile velocity mode	Not available
Profile torque mode	Not available
Homing mode	Available
Interpolated position mode	Not available
Cyclic synchronous position mode	Available
Cyclic synchronous velocity mode	Available
Cyclic synchronous torque mode	Not available
Cyclic synchronous torque mode with commutation angle	Not available
Manufacturer specific mode	Not available

3.2 State Transitions

Among the finite state automata (FSAs) defined in the CiA402 standard, the sample program supports the following modes.
In Figure 3-2, the state where torque is being applied through the motor is "Operation enabled". The motor is activated at the times of transitions from "Switched on" to "Operation enabled" (transition 4). The motor is deactivated at the times of transitions from "Operation enabled" to several other states (transitions 5, 8, 9). However, at the times of transitions from "Operation enabled" to "Quick stop active" (transition 11) or the timing of transition to "Fault reaction active" (transition 13), the application of torque is maintained.

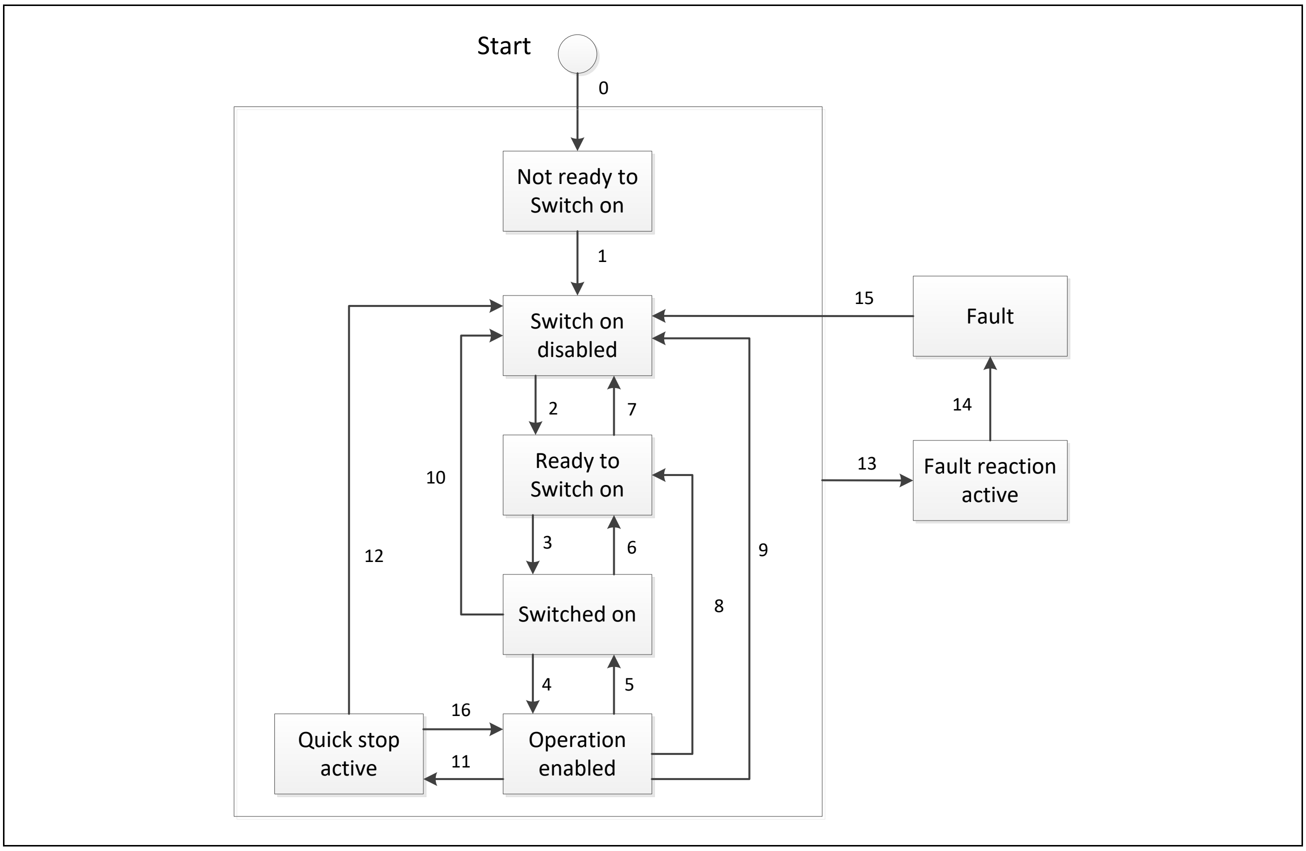


Figure 3-2 CiA402 State Transition Diagram

3.3 State Transition Functions

Table 3-2 shows the CiA402 state transition function list. Each function is linked to the number of each state transition of CiA402 FSA shown in Figure 3-2, and the corresponding function is called when the state transition occurs.

Table 3-2 List of CiA402 State Transition Functions

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

The specifications of the CiA402 state transition functions are described below.

CiA402_StateTransition(N)

These functions are called when a state transition (N), where N = 1 to 12 and 14 to 16 as specified for a CiA402 FSA, occurs.
Write code for the processing to be executed when the given state transitions occur.

Format

UINT16 CiA402_StateTransition(N)(TCiA402Axis *pCiA402Axis)

Parameters

TCiA402Axis *pCiA402Axis

Return Values

0: Normal end
1: The state transition did not proceed

Properties

The prototypes are declared in cia402appl.h.

Description

If a fault occurs during processing, set the objects to appropriate values and end function calling in accord with the CiA402 standard. When 1 is set as the return value, the state transition has not proceeded.

Example

```
TCiA402Axis *pCiA402Axis;
UINT16 retval ;

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

CiA402_LocalError

This function is called in response to the detection of an error specified in the CiA402 drive profile. After this function is executed, state transition 13 will proceed as is specified for CiA402 FSAs.
Write code for the processing to be executed when an error is detected.

Format

void CiA402_LocalError(UINT16 ErrorCode)

Parameters

UINT16 ErrorCode : CiA402 drive profile error code

Return Values

None

Properties

The prototypes are declared in cia402appl.h.

Description

The error code specified as an argument is stored in object 0x603F and sent to the EtherCAT master.

Example

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

3.4 Object Dictionary

The portion of the object dictionary supported by the sample program is listed below.

Table 3-3 Object Dictionary Supported by the Sample Program

INDEX	Sub	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	Mandatory(all)	rw	INT8	No
0x6061		Modes of operation display	Mandatory(all)	ro	INT8	TxPDO
0x6064		Position actual value	Mandatory(csp,csv)	ro	INT32	TxPDO
0x6065		Following error window	Recommended(csp)	rw	UINT32	No
0x6066		Following error time out	Recommended(csp)	rw	UIN16	No
0x606C		Velocity actual value	Recommended(csv)	ro	INT32	TxPDO
0x6077		Torque actual value	Recommended(scp,csv)	ro	INT16	No
0x607A		Target position	Mandatory(csp)	rw	INT32	RxPDO
0x607B	0	Position range limit	Recommended(csp)	ro	UINT8	No
	1	Min position range limit	Recommended(csp)	rw	INT32	No
	2	Max position range limit	Recommended(csp)	rw	INT32	No
0x607C		Home Offset	Recommended(hm)	rw	INT32	RxPDO
0x607D	0	Software position limit	Recommended(csp)	ro	UINT8	No
	1	Min position limit	Recommended(csp)	rw	INT32	No
	2	Max position limit	Recommended(csp)	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	INT32	No
0x6098		Homing method	Mandatory(hm)	rw	INT8	RxPDO
0x6099	0	Homing speeds	Optional	ro	UINT8	No
	1	Speed during search for switch	Optional	rw	INT32	RxPDO
	2	Speed during search for zero	Optional	rw	INT32	RxPDO
0x609A		Homing acceleration	Optional	rw	UINT32	RxPDO
0x60B0		Position offset	Optional	rw	INT32	No

0x60B1		Velocity offset	Recommended(csp)	rw	INT32	No
0x60B2		Torque offset	Recommended(scp,csv)	rw	INT16	No
0x60C2	0	Interpolation time period	Recommended(csp,csv)	ro	UINT8	No
	1	Interpolation time period value	Recommended(csp,csv)	rw	UINT8	No
	2	Interpolation time index	Recommended(csp,csv)	rw	INT8	No
0x60F4		Following error actual value	Recommended(csp)	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. Motion Control Parameters

The definitions of the settings in the sample program for the motion control parameters set in the CiA402 objects are given below.

Table 4-1 List of motor control parameters

INDEX	OBJECT Name	Support Mode	Unit	Data	PDO Mapping
0x6064	Position actual value	pp,csp,csv	[deg]	INT32	TxPDO
0x606C	Velocity actual value	csv	[rpm]	INT32	TxPDO
0x607A	Target position	pp,csp	[deg]	INT32	RxPDO
0x6081	Profile velocity	pp	[deg]	UINT32	RxPDO
0x6083	Profile acceleration	pp	[rpm/s]	UINT32	RxPDO

Note : Profile velocity and Profile acceleration are not used because profile control is not performed in the position control of the stepping motor.

4.1 Data type

The parameter type is INT32 or UINT32, but it is in fixed-point format, including the first decimal place.
For example, if want to set the Target position to 178.9 °, set "1789". Similarly, if the value of Position actual value is "1789", it means that it is 178.9 °

4.2 Acceleration Parameters

Control parameters such as gain tuned by the motor control development support tool "Renesas Motor Workbench" are reflected in the motor control program source file, so that the same values are used for control by EtherCAT.

On the other hand, the command value of the acceleration used in RMW is different from the motor control parameter set in the (Profile acceleration) CiA402 object, so conversion is required.

RMW uses the acceleration time [s] to reach the target speed to define the acceleration.
Figure 4 1 shows that the acceleration changes from acc1 to acc2 by changing the acceleration time to reach the target speed speed from t1 to t2.
In addition, the acceleration at that time can be expressed by the formula of velocity ÷ acceleration time.

acc1 = speed÷t1
acc2 = speed÷t2

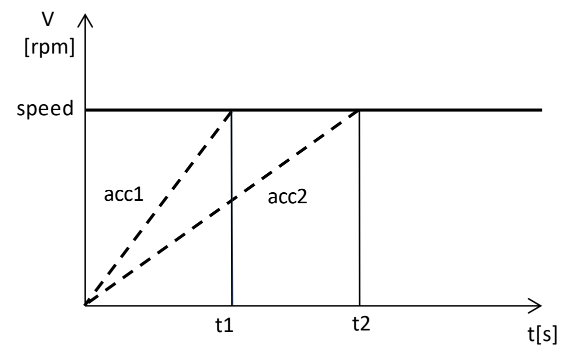


Figure 4-1 Acceleration Times and Acceleration

Conversion example

・ Acceleration command value

Convert TACC = 0.3 [s] to AC [rpm / s] when R = 2000 [rpm]

2000 rpm ÷ 0.3 s = 6666.7 rpm/s

Profile acceleration = Ac × 10 = 6666.7 × 10 = 66667

5. API Functions

5.1 Overview

The API functions for the motor control program interface are shown below.

Functions	Description
R_MTR_ECAT_Open	Make initial settings for the motor control program interface.
R_MTR_ECAT_GetPositionPFStatus	Gets the profile status of the encoder position control.
R_MTR_ECAT_SetActualPosition	Set the current position [deg].
R_MTR_ECAT_SetProfileSpeed	Set the profile speed command value [rpm].
R_MTR_EACAT_SetAcceleration	Set the acceleration command value [rpm / s].
R_MTR_ECAT_SetDeceleration	Set the deceleration command value [rpm / s].
p_api	API pointer for motor control※

API pointer for motor control※
This motor control program differs for each motor. The Motor control API is named diffrently in BLDC motor and stepper motor. The structure members of API pointer for motor control are named as a operation names imdependent of type of motor. That allows EtherCAT communications to control the motor without being aware of type of motor.

5.2 R_MTR_ECAT_Open

This function initializes the motor control program. This function must be executed before any other API function is called.

Format

e_mtr_ecat_ret_t R_MTR_ ECAT_Open (st_encoder_vector_control_t *p_st_srfoc);

Parameters

st_encoder_vector_control_t *p_st_srfoc
Sets a pointer to the encoder vector control variable to be controlled

Return Values

MTR_ECAT_SUCCESS: Initialization successful
MTR_ECAT_ERR_NOT_SUPPORT: Variable pointer is empty

Properties

Prototype is declared in "r_mtr_driver_ecat_acces.h".

Description

Initializes the system operating status and sets the default values for motor control parameters.

Example

```
/* Setup EtherCAT motor interface */
R_MTR_ECAT_Open(&g_st_encoder_vector);
```

5.3 R_MTR_ ECAT_GetPositionPFStatus

This function gets the profile status of the encoder position control.

Format

e_mtr_ecat_ret_t R_MTR_ECAT_GetPositionPFStatus(uint8_t *u1_status)

Parameters

u1_status: Profile status

MTR_POS_STEADY_STATE (0): Steady state (current position has not changed)
MTR_POS_TRANSITION_STATE (1): Transition state (current position is changing)

MTR_ECAT_SUCCESS: Successful acquisition
MTR_ECAT_ERR_NOT_OPEN: Acquisition failure because the driver is not opened

Return Values

None

Properties

Prototype is declared in "r_mtr_driver_ecat_acces.h".

Description

This function can be used to check whether the motor is stationary during position control.

Example

```
  uint8_t    u1_pos_state;

/* Get position profile status */
R_MTR_ECAT_GetPositionPFStatus(&u1_pos_status);
```

5.4 R_MTR_ ECAT_SetActualPosition

This function sets the current position [deg].

Format

e_mtr_ecat_ret_t R_MTR_SetActualPositionUnits (float f4_actual_position)

Parameters

Position command value [deg]

Return Values

MTR_ECAT_SUCCESS: Successful setting
MTR_ECAT_ERR_NOT_OPEN: Setting failure because the driver is not open

Properties

Prototype is declared in "r_mtr_driver_ecat_acces.h".

Description

Position command value is signed and the unit is [deg].
This function is a function that sets only the value of the current position without controlling the motor.
It can be used when you want to have an offset at the initial position in homing mode.

Example

```
/* Set current position 180.1[deg]*/
R_MTR_SetActualPosition(180.1f);
```

5.5 R_MTR_ ECAT_SetProfileSpeed

This function sets the maximum speed command value for profile control.

Format

e_mtr_ecat_ret_t R_MTR_ECAT_SetProfileSpeed(float f4_profile_speed)

Parameters

Speed command value [rpm]

Return Values

MTR_ECAT_SUCCESS: Successful setting
MTR_ECAT_ERR_NOT_OPEN: Setting failure because the driver is not open

Properties

Prototype is declared in "r_mtr_driver_ecat_acces.h".

Description

The speed command value is signed and the unit is [rpm].

Example

```
/* Set Profile Speed 2000.5 rpm */
R_MTR_SetSpeedUinits(2000.5f);
```

5.6 R_MTR_ECAT_SetAcceleration

This function sets the acceleration command value [rpm / s].

Format

e_mtr_ecat_ret_t R_MTR_ECAT_SetAcceleration(float f4_ref_acceleration)

Parameters

Acceleration command value [rpm / s]

Return Values

MTR_ECAT_SUCCESS: Successful setting
MTR_ECAT_ERR_NOT_OPEN: Setting failure because the driver is not open

Properties

Prototype is declared in "r_mtr_driver_ecat_acces.h".

Description

The acceleration command value is signed and the unit is [rpm / s].

Example

```
/* Set acceleration 4678.9[rpm/s] */
R_MTR_ECAT_SetAcceleration (4678.9f);
```

5.7 R_MTR_ ECAT_SetDeceleration

This function sets the deceleration command value.

Format

e_mtr_ecat_ret_t R_MTR_ECAT_SetDecceleration(float f4_ref_deceleration)

Parameters

This function sets the deceleration command value

Return Values

MTR_ECAT_SUCCESS: Successful setting
MTR_ECAT_ERR_NOT_OPEN: Setting failure because the driver is not open

Properties

Prototype is declared in "r_mtr_driver_ecat_acces.h".

Description

The deceleration command value is signed and the unit is [rpm / s].
[Note] The sample program does not use the deceleration command value for motor control.

Example

```
/* Set deceleration 4678.9 [rpm/s] */
R_MTR_ECAT_SetDeceleration (4678.9f);
```

5.8 p_api

This is variables which store function pointer of motor control API.

Format

st_ecat_motor_api_t const * p_api;

Struct

The members of motor control API structures "st_ecat_motor_api_t" and the stored motor control API are shown below.

Member name	PositionCommandModeSet
Type	void (* PositionCommandModeSet) (st_encoder_vector_control_t *p_st_encoder_vector, uint8_t u1_position_command_mode);
Motor control API	R_MOTOR_ENCODER_VECTOR_PositionCommandModeSet
Member name	CtrlTypeSet
Type	void (* CtrlTypeSet) (st_encoder_vector_control_t * p_st_encoder_vector, uint8_t ctrl_type);
Motor control API	R_MOTOR_ENCODER_VECTOR_CtrlTypeSet
Member name	MotorStart
Type	void (* MotorStart)(st_encoder_vector_control_t *p_st_encoder_vector);
Motor control API	R_MOTOR_ENCODER_VECTOR_MotorStart
Member name	MotorStop
Type	void (* MotorStop)(st_encoder_vector_control_t *p_st_encoder_vector);
Motor control API	R_MOTOR_ENCODER_VECTOR_MotorStop
Member name	MotorErrorCancel
Type	void (* MotorErrorCancel)(st_encoder_vector_control_t *p_st_encoder_vector);
Motor control API	R_MOTOR_ENCODER_VECTOR_MotorReset
Member name	MotorReset
Type	void (* MotorReset)(st_encoder_vector_control_t *p_st_encoder_vector);
Motor control API	R_MOTOR_ENCODER_VECTOR_MotorReset
Member name	PositionGet
Type	float (* PositionGet)(st_encoder_vector_control_t *p_st_encoder_vector);
Motor control API	R_MOTOR_ENCODER_VECTOR_PositionGet
Member name	PositionSet
Type	void (* PositionSet) (st_encoder_vector_control_t *p_st_encoder_vector, float f4_ref_position);
Motor control API	R_MOTOR_ENCODER_VECTOR_PositionSet
Member name	SpeedGet
Type	float (* SpeedGet)(st_encoder_vector_control_t *p_st_encoder_vector);
Motor control API	R_MOTOR_ENCODER_VECTOR_SpeedGet
Member name	SpeedSet
Type	void (* SpeedSet) (st_encoder_vector_control_t *p_st_encoder_vector, float f4_ref_speed);
Motor control API	R_MOTOR_ENCODER_VECTOR_SpeedSet
Member name	StatusGet
Type	uint8_t (* StatusGet)(st_encoder_vector_control_t *p_st_encoder_vector);
Motor control API	R_MOTOR_ENCODER_VECTOR_StatusGet
Member name	ErrorStatusGet
Type	uint16_t (* ErrorStatusGet)(st_encoder_vector_control_t * p_st_encoder_vector);
Motor control API	R_MOTOR_ENCODER_VECTOR_ErrorStatusGet

Properties

Prototype is declared in “r_mtr_driver_ecat_acces.h”

Description

The motor control API stored in the pointar “p_api” is defined in ”r_mtr_driver_ecat_acces.c” as a table.
const st_ecat_motor_api_t ecat_motor_api

Example

```
/* Motor API settings */
st_ecat_motor_api_t const * p_api;
p_api = &ecat_motor_api;
st_ecat_motor_api_t const * p_motor_api = p_api;
/* Motor Start */
(p_motor_api->MotorStart)(ecat_param_buffer.p_st_foc);
```

6. Checking Operation of the Application on the Solution Kit

This section describes the operation of the sample application that controls a motor via EtherCAT communications with the use of the motor solution kit.

6.1 Operating Environment

The sample program covered in these manual runs in the environment below.

Table 6-1 Operating Environment

Item	Description
Boards to be used	RX72M CPU Card with RDC-IC Renesas Electronics: RTK0EMXDE0C00000BJ Inverter board for driving stepping motor Renesas Electronics: RTK0EM0000B11020BJ
CPU	RX CPU (RXv3)
Operating voltage	24 V
Communication protocol	EtherCAT
Integrated development environment	CCRX compiler (V3.05.00 or later) + e2studio (2023-10 or later)
FIT Module	r_ecat_rx: 1.30 or later
Emulator	Renesas E2 Lite
SSC tool	Provided by the EtherCAT Technology Group (ETG) Slave Stack Code (SSC) tool Version 5.13 or later
Software PLC	Beckhoff Automation TwinCAT® 3 (download this from the Beckhoff web site)

In addition, installation of the SSC tool and software programmable logic controller (PLC) is required before starting the settings.

Software components used in the sample programs in this manual and their versions are assumed to be as follows.

Table 6-2 Operation confirmed component

Component	Version	Setting
Borad Support Package (r_bsp)	7.41	r_bsp
EtherCAT Slave Controller (ESC) Software (r_ecat_rx)	1.30	r_ecat_rx
Event link controller	1.9.0	Config_ELC
Watchdog timer	1.10.0	Config_IWDT
Compare match timer	2.5.0	Config_CMT0, Config_CMT2
Single scan mode S12AD	2.4.0	Config_S12AD0, Config_S12AD1
Normal mode timer	1.12.0	Config_MTU0
Port	2.4.1	Config_PORT
Port output enable	1.11.0	Config_POEG
Phase Counting Mode Timer	2.4.0	Config_MTU1
Interrupt Controller	2.3.0	Config_ICU
General purpose PWM timer	1.5.2	Config_GPT0, Config_GPT1, Config_GPT2, Config_GPT3,

6.2 Operating Environment Settings and Connection

Connect the required wiring between the power supply, motor, and inverter board.

(1) Connect the power line of the motor to the inverter board output section, the encoder output line of the motor to the input section of the CPU card as shown below.

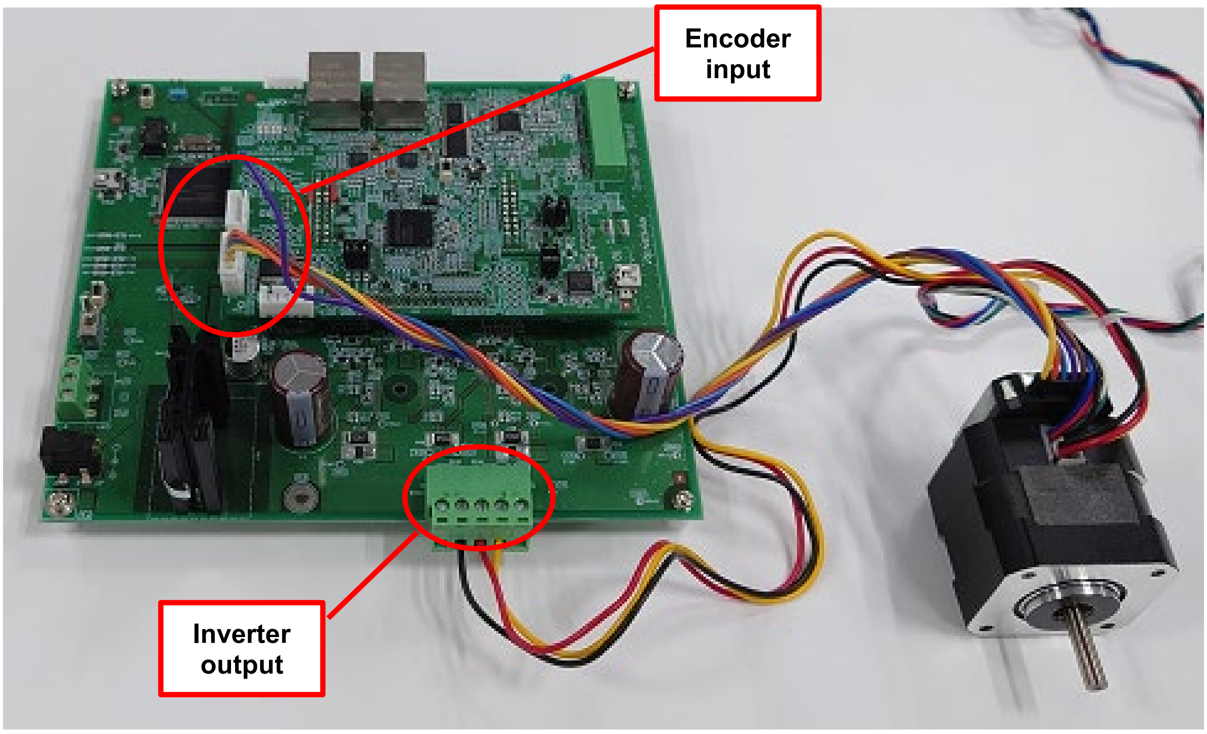


Figure 6-1 Connection of the Three-Phase Power Lines

(2) Connect the power supply to the inverter board as follows

Figure 6-2 Connection of the Power Supply for the Inverter

(3) The connection configuration is as follows.

(4) The details of the inverter board are shown below.

-- Use the AC adapter to turn on the power.

-- Connect the RMW connector when using the RMW (Renesas Motor Workbench).

-- SW is not used in control as covered by this manual.

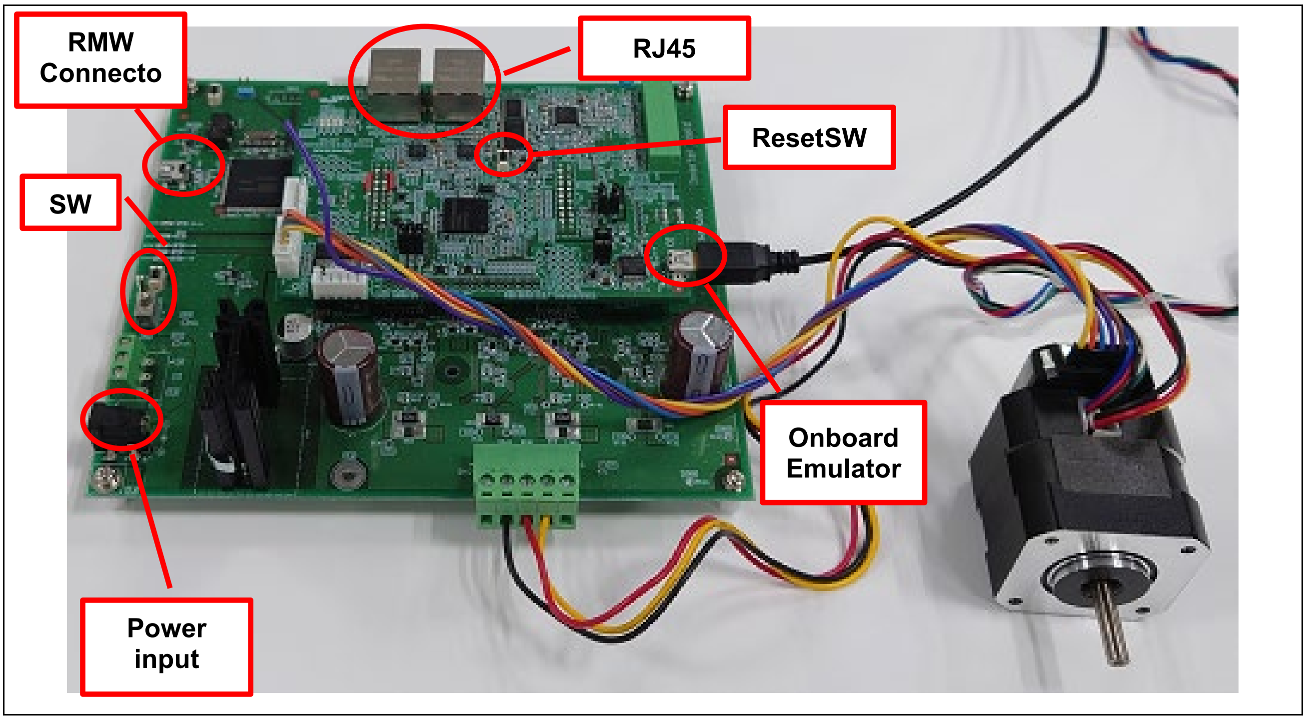


Figure 6-3 Inverter board connection diagram

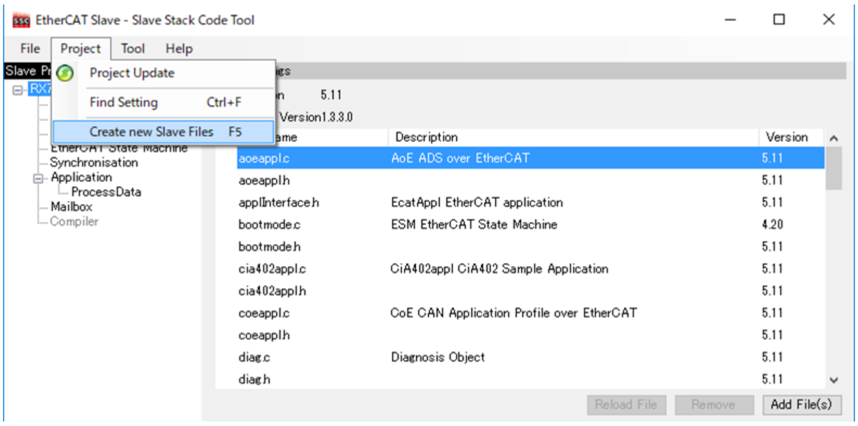
6.3 Building the Sample Program

This sample project does not include the EtherCAT Slave Stack Code.
However, the project requires the EtherCAT Slave Stack Code and to generate and use this you must obtain the EtherCAT Slave Stack Code (SSC) tool.
The EtherCAT Technology Group (ETG) provides the SSC package.

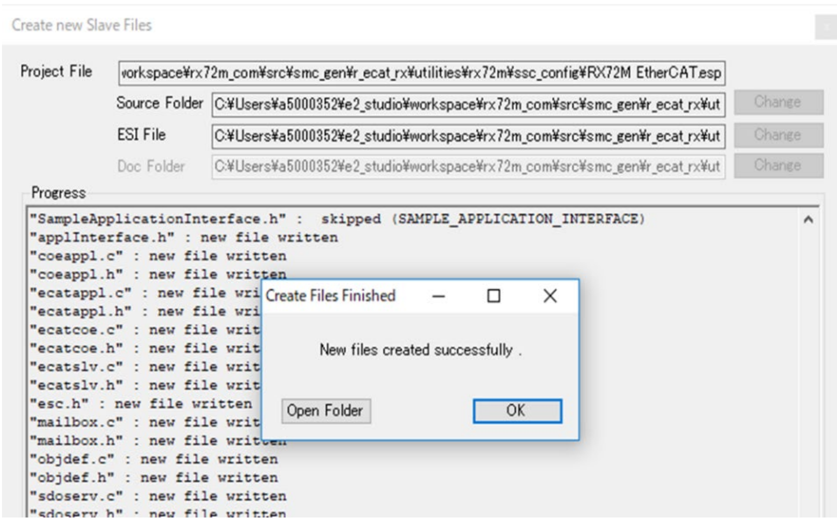
(1) Double-click on the SSC project file of the sample program to activate the SSC tool.

rx72m_ecat_cia402_bldc_encd\utilities\ssc_config\RX72M EtherCAT CiA402.esp

(2) Click on [Project] → [Create New Slave Files], and then click on [Current new Slave Files] → [Start].



(3) When the source code has been generated normally, "New files created successfully" will be displayed. Click on [OK].



(4) If you have not installed the patch file, GNU Patch Ver2.5.9 or later is required. If it has been installed, skip this step.

Download the patch file (Ver2.5.9) from the following Web page and store "patch.exe" in a folder that has a path executable from the command prompt.

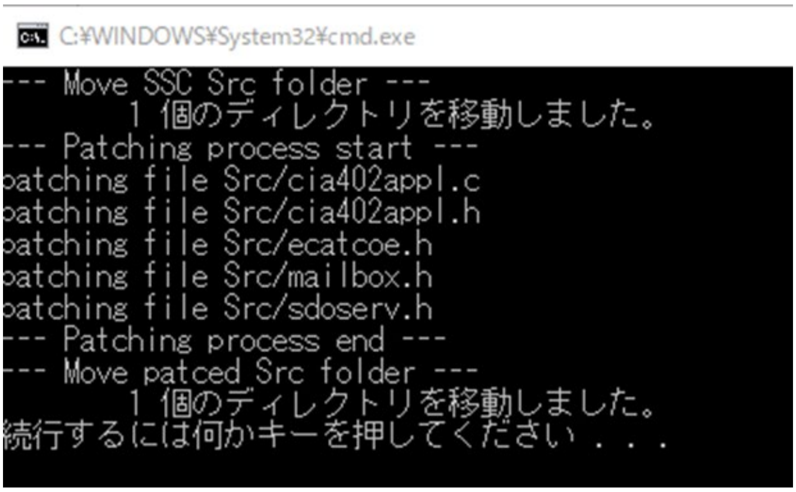
http://gnuwin32.sourceforge.net/packages/patch.htm

(5) Right-click on the apply_patch.bat file and select [Run as administrator] → [Yes]. The patch file contains the RX-specific modifications to the SSC source files.

rx72m_ecat_cia402_bldc_encd\utilities\batch_files\apply_patch.bat

(6) After the patch has been applied, the modified source file will be stored in the following folder.

rx72m_ecat_cia402_bldc_encd\project\ecat\application\beckhoff\Src

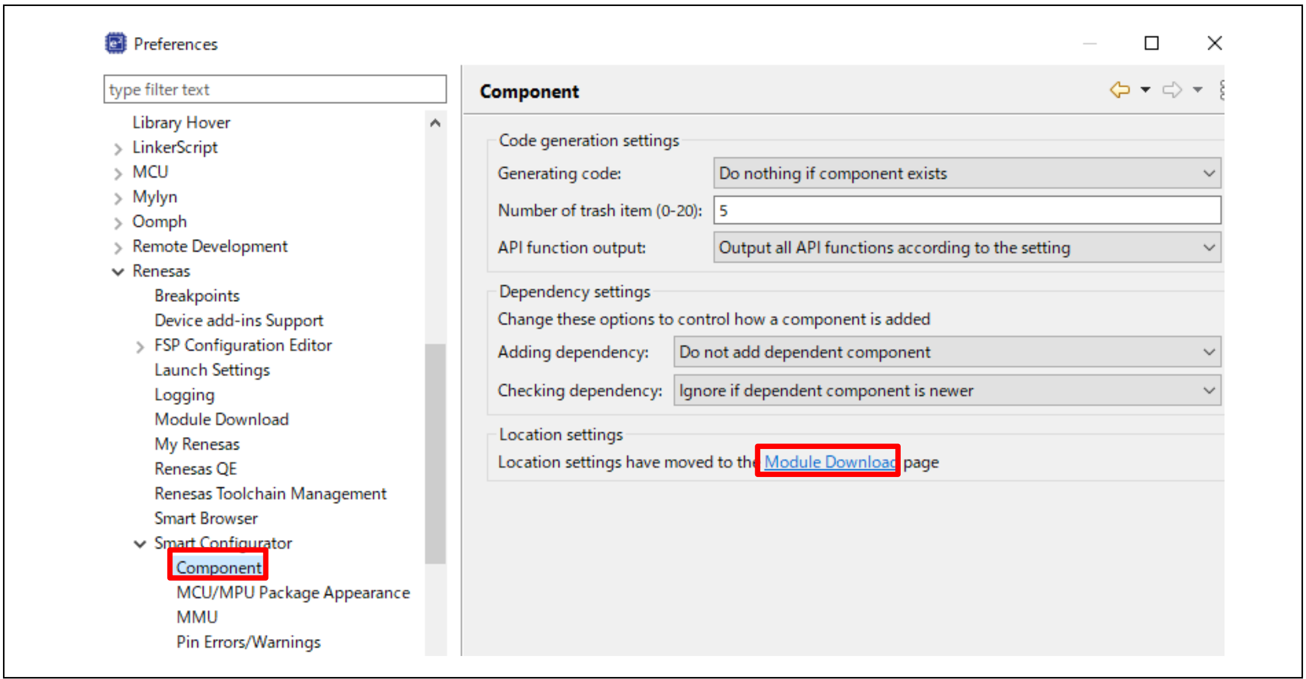


6.4 Adding the FIT Module to your Project

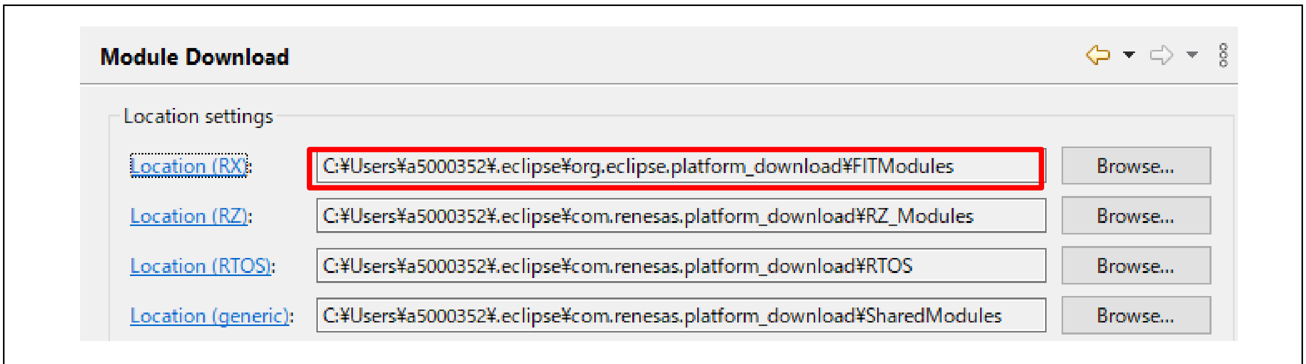
To enable the use of EtherCAT FIT modules in the Smart Configurator, they must be added manually to e2 studio.

(1) Copy the EtherCAT FIT module to the folder where the FIT module of e2 studio is saved.
Check the location of the FIT module in e2 studio

- [Window] → [Preferences] →The Preferences window opens.
- [C/C++] → [Renesas] → [Smart Configurator] → [Component]
- [Folder settings] → [Module Download] open the page



- Location (RX):" is the folder where the FIT module of e2 studio is saved.



EtherCAT FIT module is stored in the FITModules folder of the sample program.

- Copy the files in the an-r01an4881xx*NNNN*-rx-ecat\FITModules folder to the location where the FIT modules are saved.

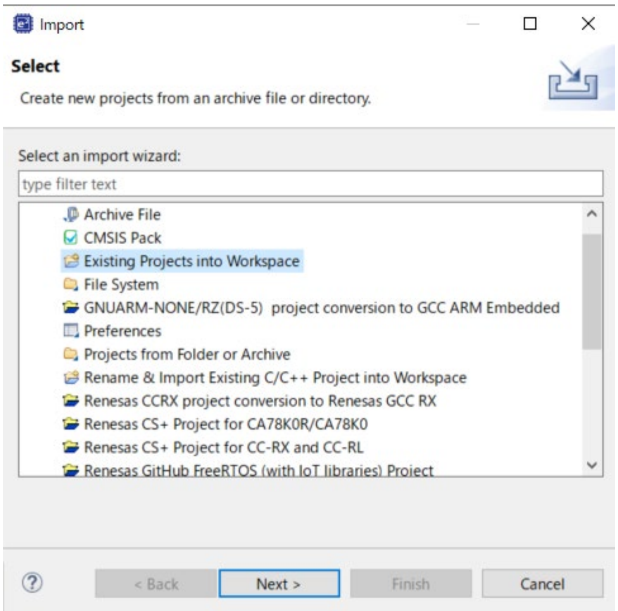
  r_ecat_rx_v*N.NN*.xml
  r_ecat_rx_v*N.NN*.zip
  r_ecat_rx_v*N.NN*_extend.mdf

  NNNN and *N.NN* are numerical values that represent the version.

6.5 Importing the Sample Project into the e2 studio

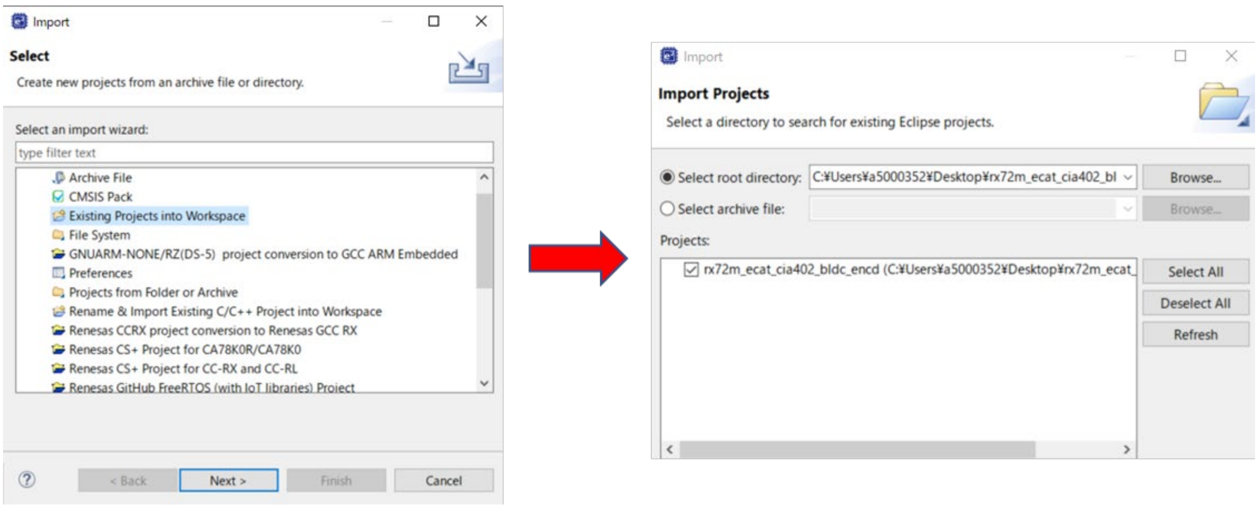
(1) Click on [File] → [Import].

(2) In the [Select an import wizard] dialog box, select [General] → [Existing Project to Workspace] and click on [Next].



(3) Select the [Select root directory] checkbox in the [Import Project] dialog box and click on [Browse].

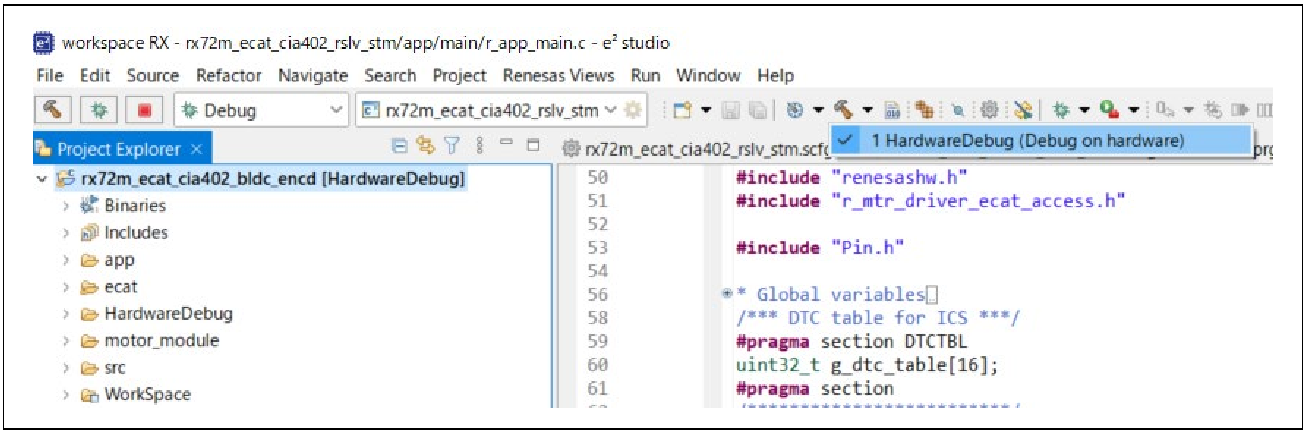
(4) Select "rx72m_ecat_cia402_bldc_encd", which is the sample project for the communications board, and click on [Open]. Check "rx72m_ecat_cia402_bldc_encd" indicated under [Project] and click on [Next] to import the project.



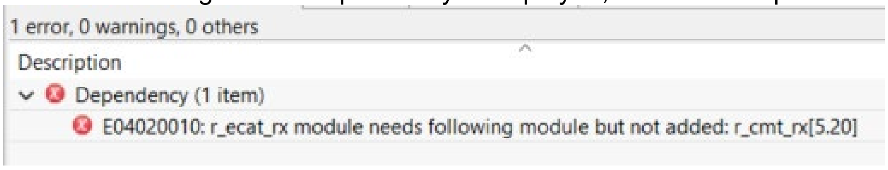
6.6 Programming and Debugging

(1) Left-click on the "rx72m_ecat_cia402_bldc_encd" project name in the project explorer, click on the arrow next to the [Build] button (hammer icon), and select [Hardware Debug] from the drop-down menu.

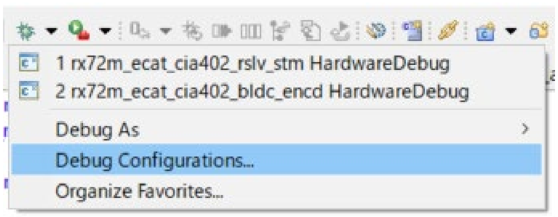
The project is built by e2studio. Check that the console does not indicate any errors in building.



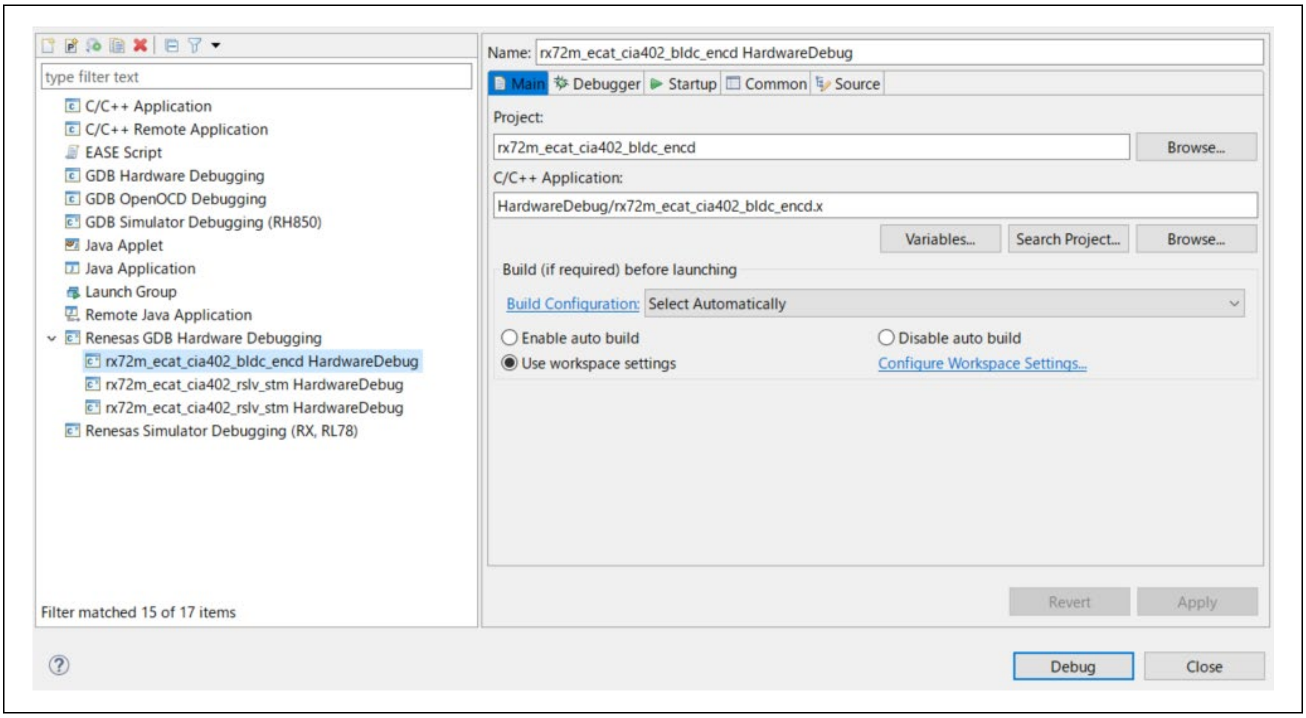
【Note】
Code generation is implemented when you build.
The error message about dependency is displayed, but this is no problem.



(2) Once building is completed, start debugging by clicking on the arrow next to the [Debug] button (insect icon) and selecting [Debug Configurations].
The project is built by e2studio. Once the build is completed, start debugging by clicking the arrow next to the [Debug] button (bug icon) and selecting [Debug Configurations].



(3) Click on "rx72m_ecat_cia402_bldc_encd Hardware Debug" to download the program to the target and press the debugging button to start it.



(4) A firewall warning for "e2-server-gdb.exe" may be displayed. Select the checkbox for [Private networks such as home and work networks] and click on <Allow access>.

(5) The user account control (UAC) dialog box may be displayed. Enter your administrator password and click on [Yes].

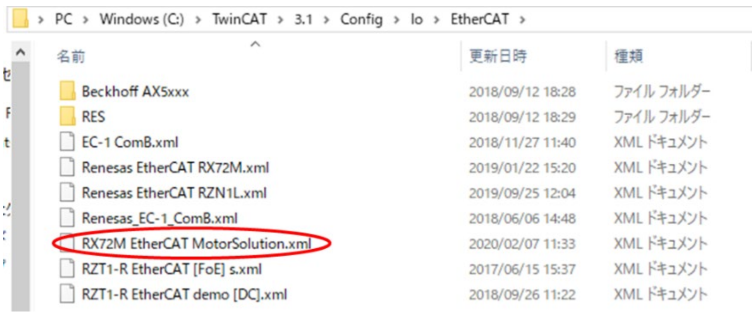
(6) If a dialog box recommending a change of the perspective is displayed in the confirmation dialog box for switching perspectives, select the "Always use this setting" checkbox and click [Yes].

(7) The green "ACT" LED on the E2 Lite debugger is always on.

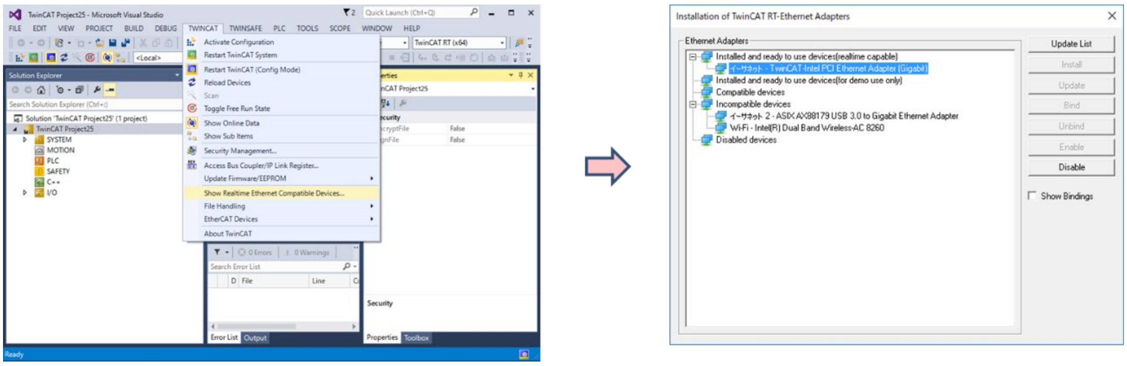
After downloading the code, click on the [Resume] button to run the code up to the first line of the function main. Click on the [Resume] button again to run the rest of the code on the target.

6.7 Connection with TwinCAT (Writing the ESI File)

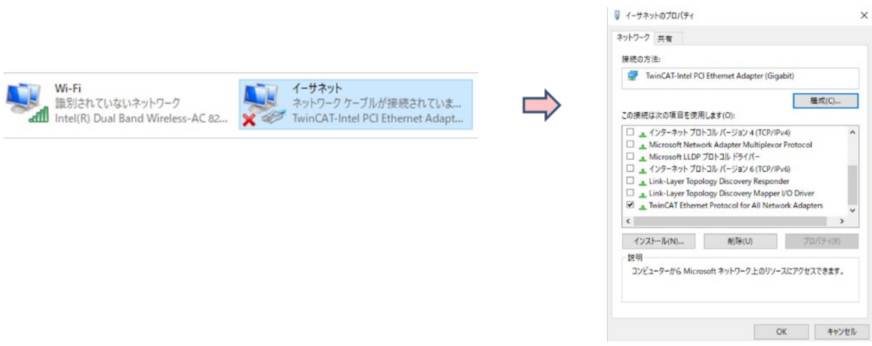
(1) Before starting TwinCAT, copy the ESI file included in the release folder to "/TwinCAT / 3.x / Config / IO / EtherCAT".
"rx72m_ecat_cia402_bldc_encd\utilities\esi\ RX72M EtherCAT MotorSolution.xml"



(2) Add the TwinCAT driver through the following procedure. This is only required the first time.
From the Start menu, select [TWINCAT] → [Show Realtime Ethernet Compatible Devices...].
Select the connected Ethernet port from among the communications ports and press [Install].



(3) Select the connected Ethernet port from among the communications ports to display its properties.
Only enable [TwinCAT Ethernet Protocol for All Network Adapters] from among the properties and close the dialog box.

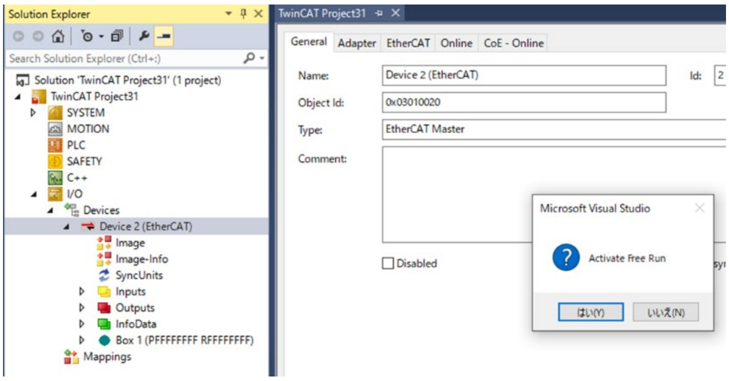


(4) Connect the LAN cable to the evaluation board. As the In/Out direction of EtherCAT has been decided, connect it to CN2 IN.

(5) Select [Beckhoff] → [TwinCAT3] → [TwinCAT XAE (VS2013)] from the start menu. After starting the program, select [FILE] → [New] → [Project] and then select [TwinCAT XAE Project] in Templates to create a new project.

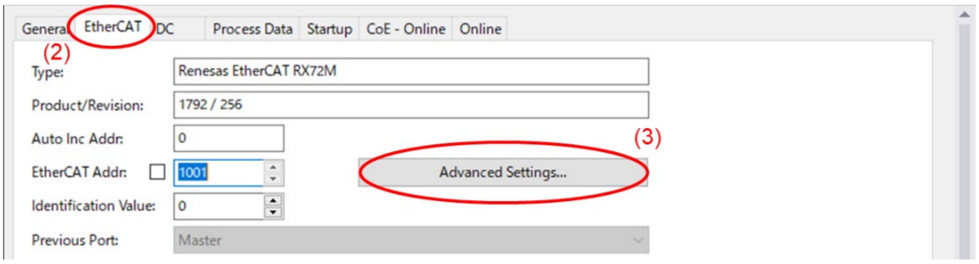
(6) Select Solution Explorer → I / O → Device → [Scan].

(7) When [Scan for boxes] is executed, the slave of Box1 will be detected and appear in Solution Explorer. If the ESI file is not recognized, it will be displayed as Box 1 (PFFFFF). In such cases, select [No] for [Activate Free Run] to download the ESI file.



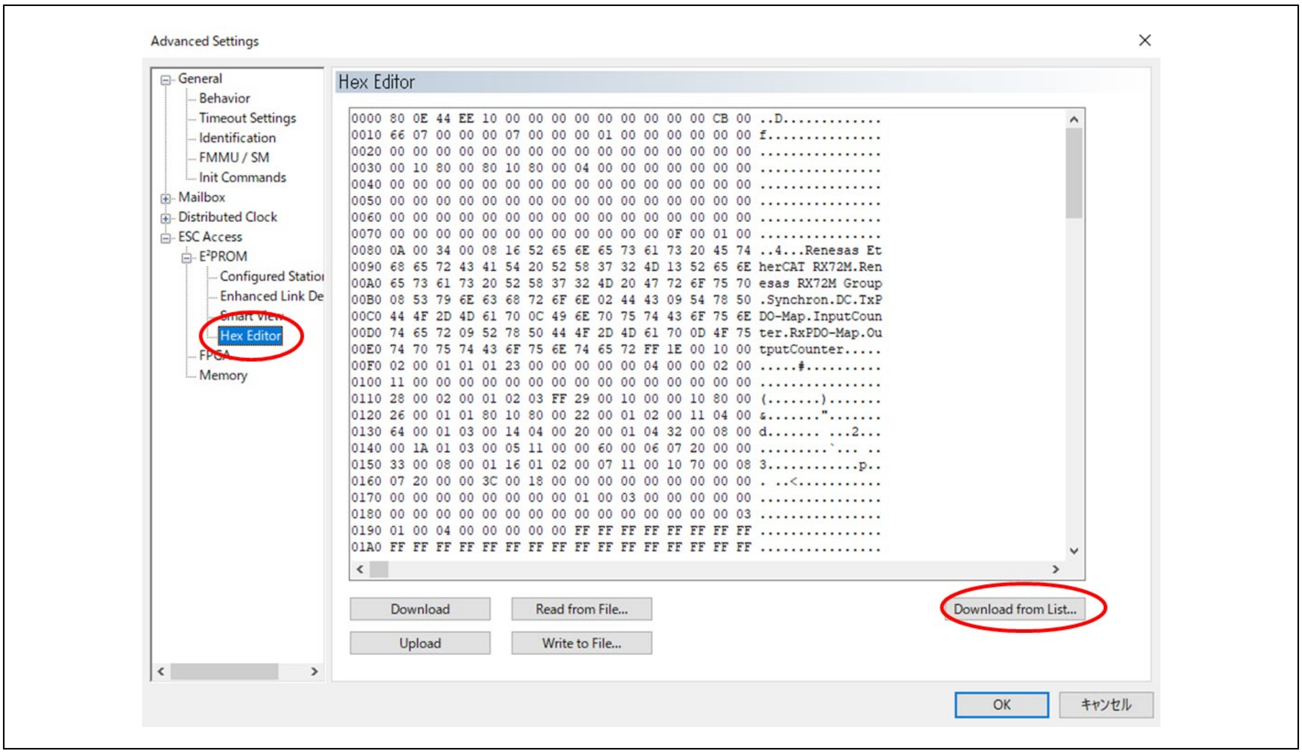
(8) If the data of another application has been written to the EEPROM, replace it. The procedure for replacing the data in the EEPROM is as follows.

- Double-click on [Box 1]. The settings screen will appear.
- Select the [EtherCAT] tab.
- Click the [Advanced Setting] button.

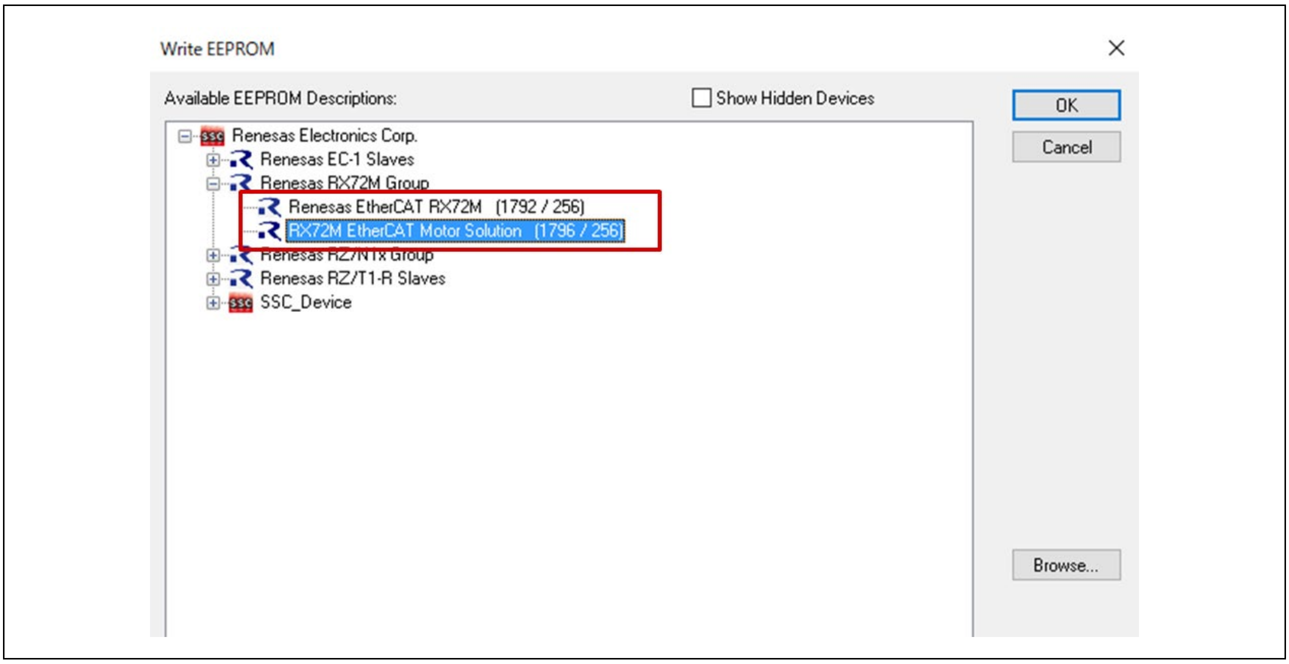


(9) Select [ESC Access] → [EEPROM] → [Hex Editor].

Select [Download from List].

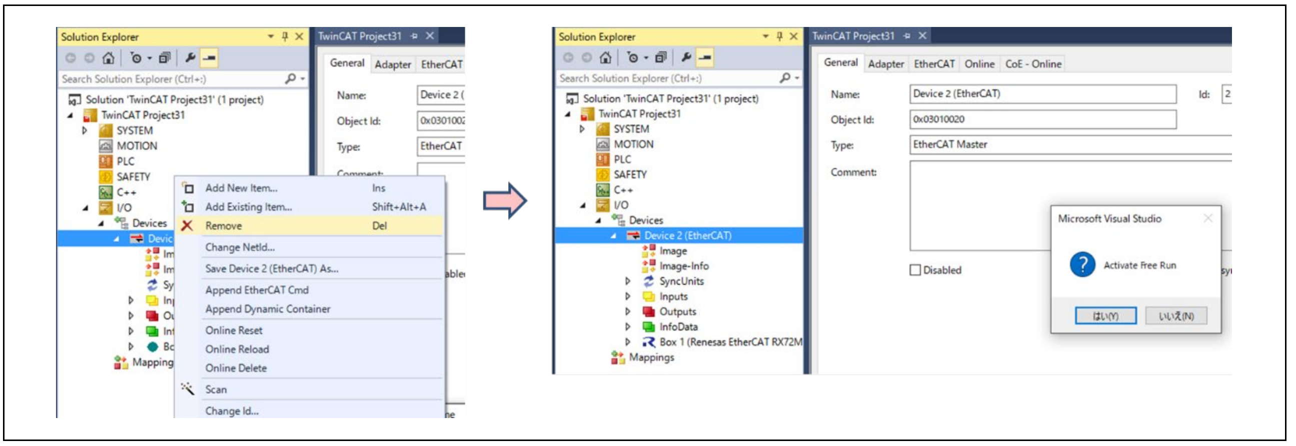


(10) A list of ESI files registered with TwinCAT3 will appear. Select the relevant file. For the motor board, select [RX72M EtherCAT MotorSolution.xml]. For the I/O board, select [Renesas EtherCAT RX72M.xml].



(11) Reflect the settings of the downloaded ESI file. Since this requires resetting the slave, temporarily delete the slave from the TwinCAT network.

After the slave has been reset, the ESI file will be read by scanning it again. Execute this with "Activate Free Run".

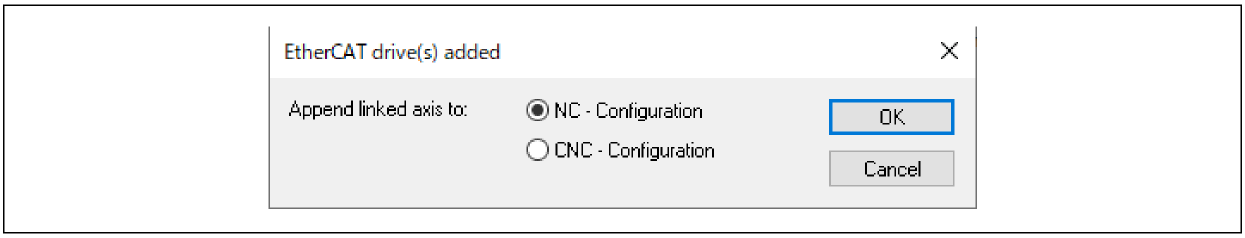


6.8 Confirmation of connection with TwinCAT3

This chapter describes the procedure for connecting and operating the evaluation environment in which the sample program is installed with TwinCAT3.

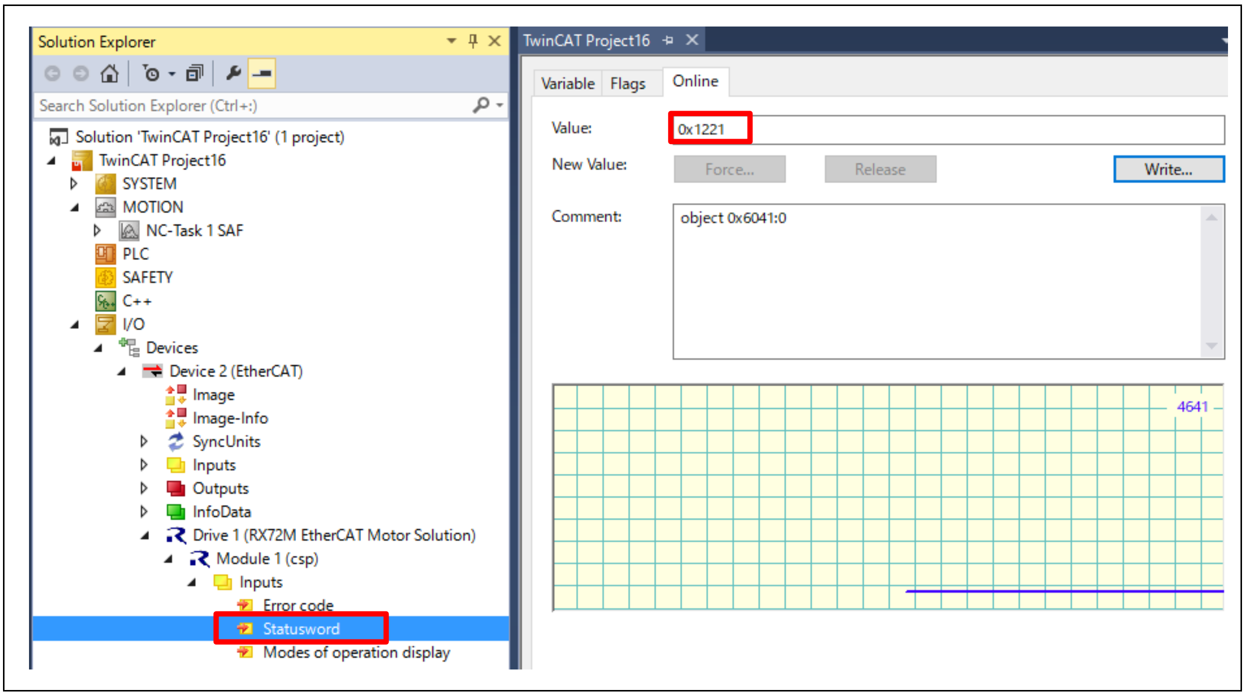
6.8.1 Confirmation of motor operation with TwinCAT3

(1) After updating the ESI file, restart TwinCAT. When you scan the Device, the ESI for Motor Solution is used, so the Axis Configuration settings will appear. Select [NC-Configuration].

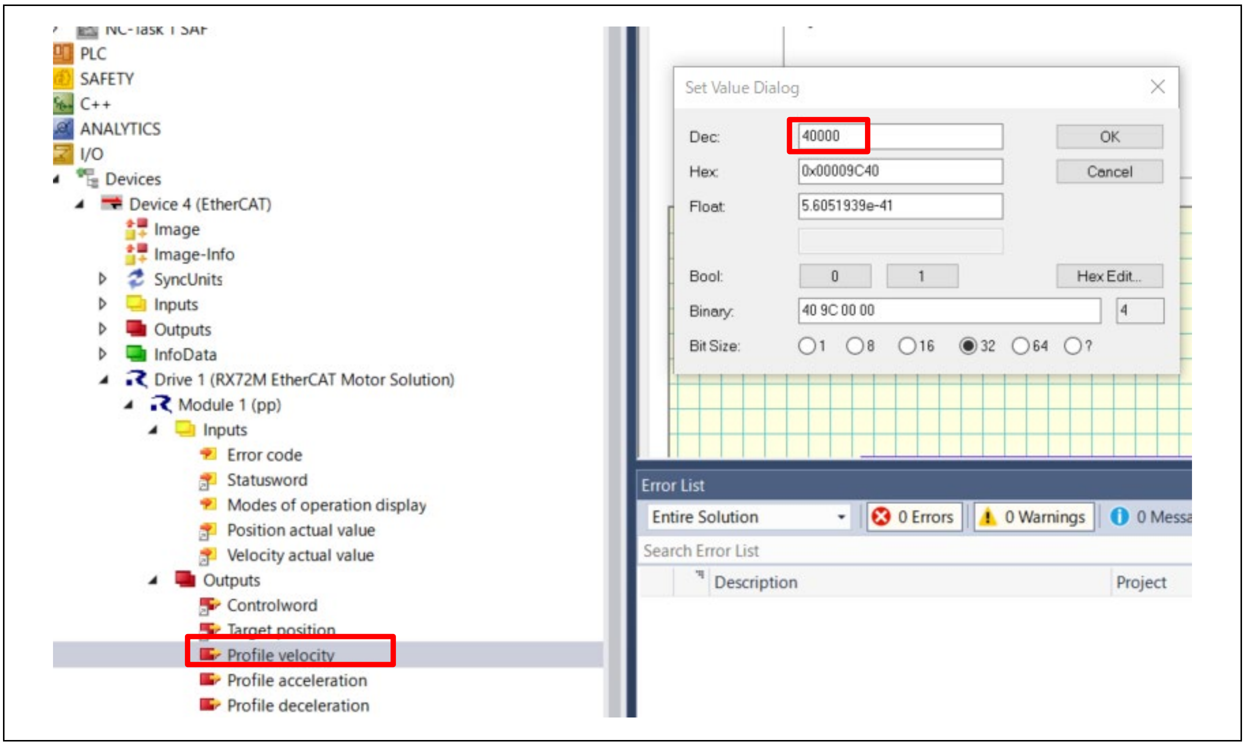


(2) The Confirm the operation of pp profile.
Set the Control word to 128 (Dec).

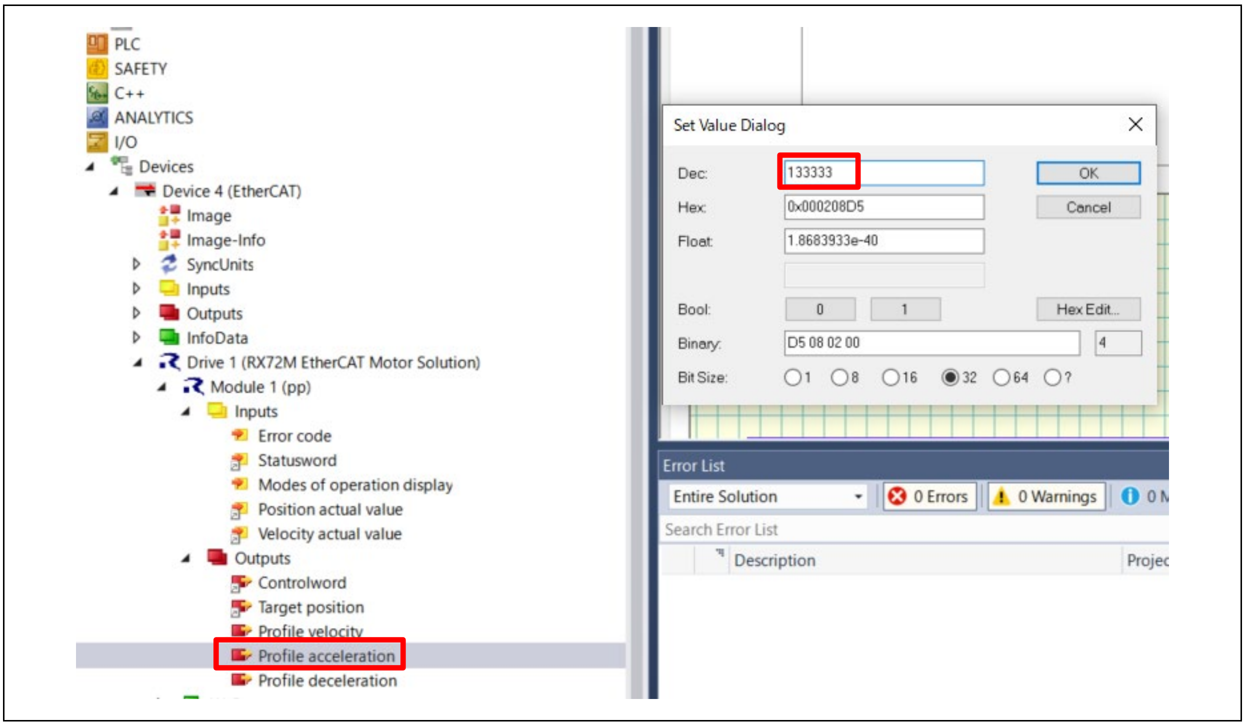
(3) Confirm that the Status word is 0xX221 (Hex)



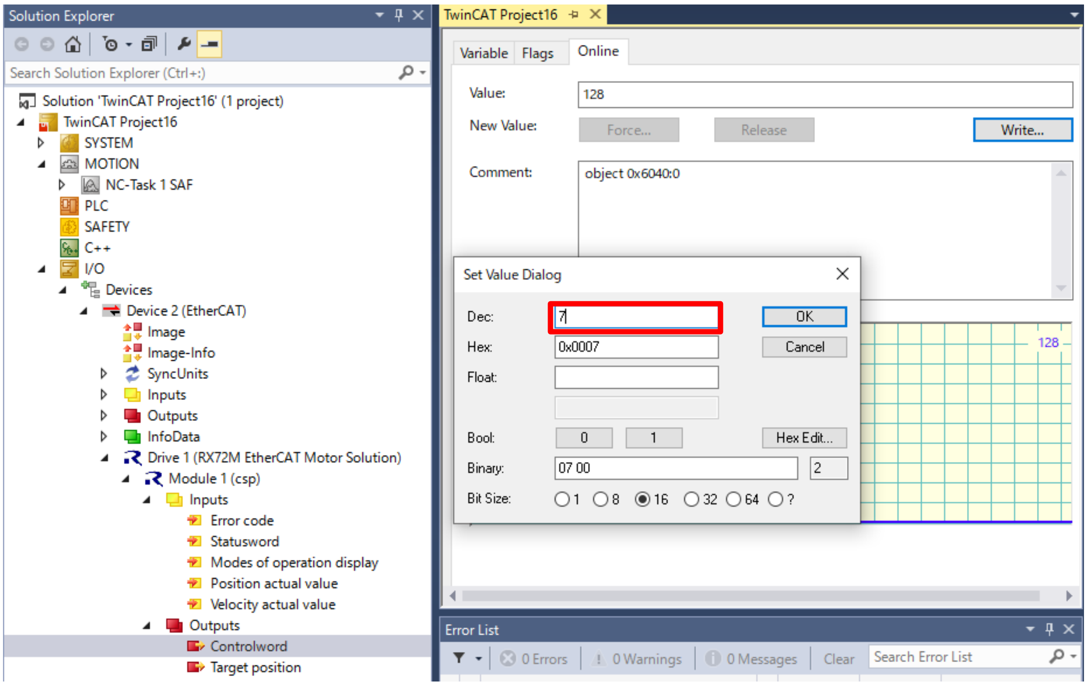
(4) Set the Profile velocity.
Set Profile velocity = 40,000 (Dec).



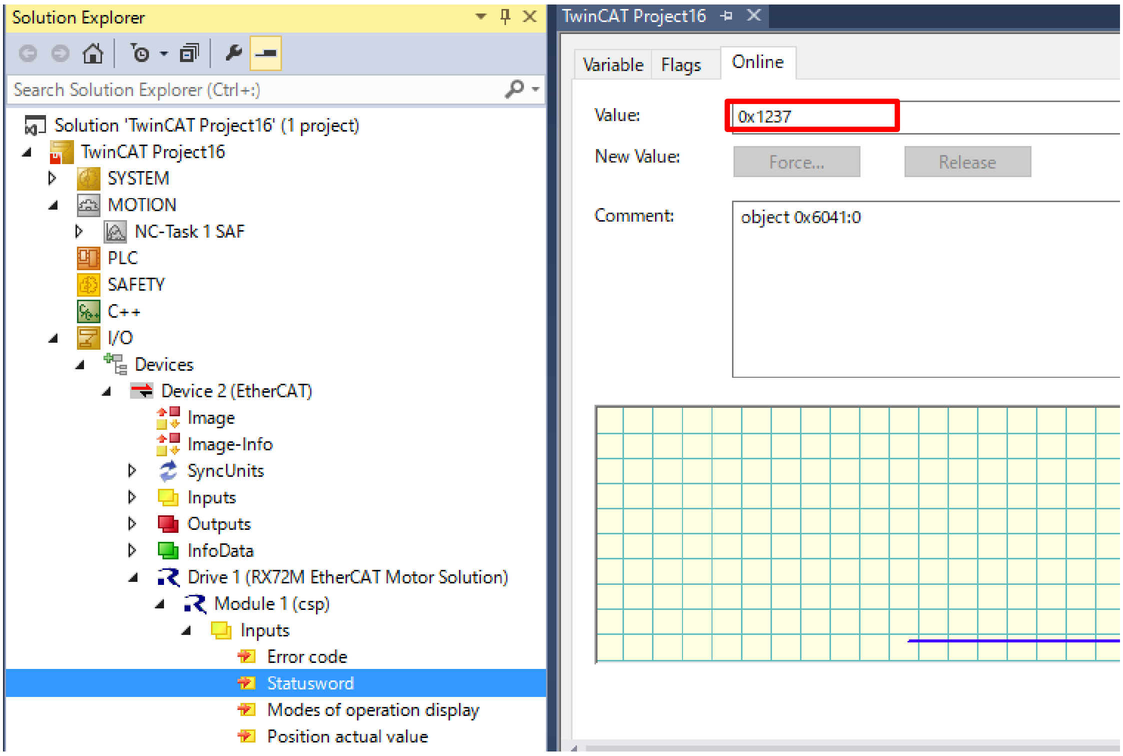
(5) Profile Set Profile accelaeration.
Set Profile accelaeration = 133333 (Dec).



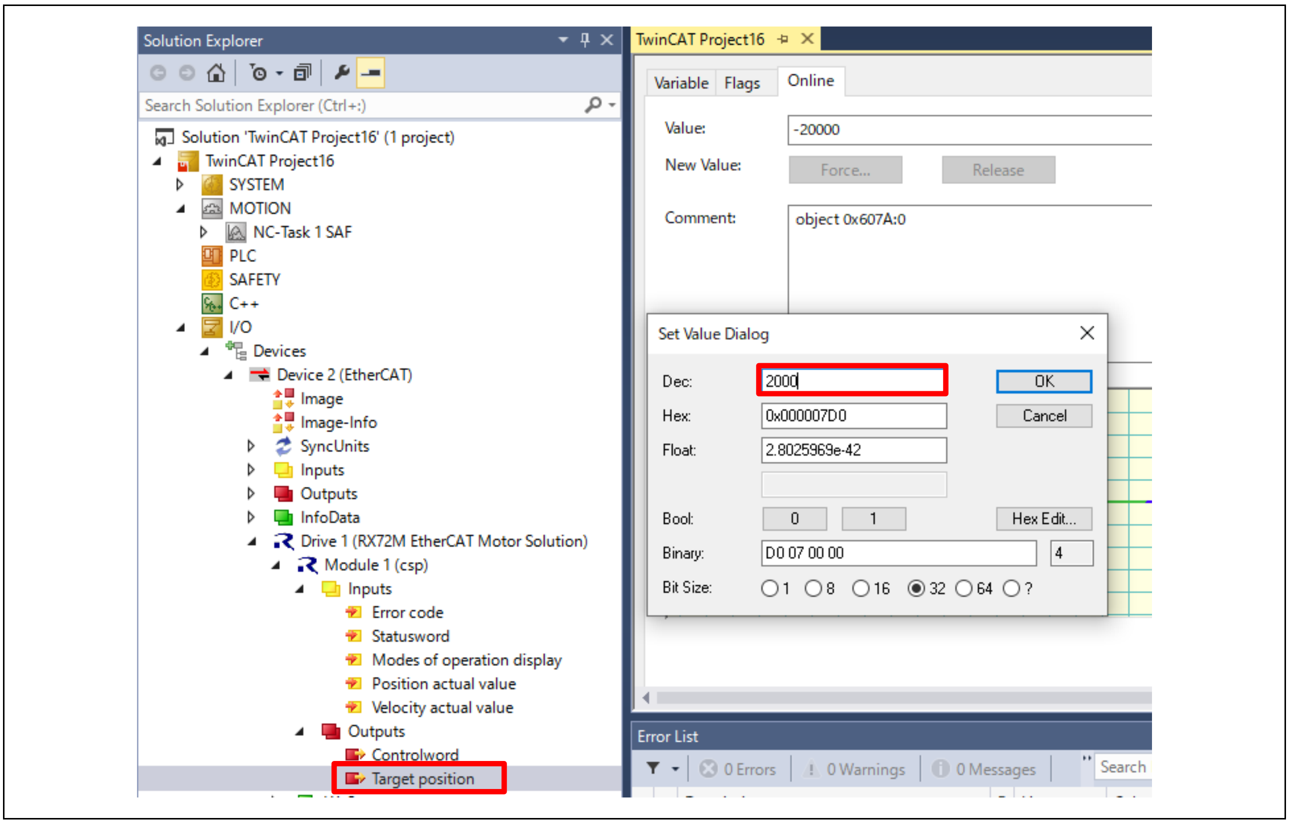
(6) Set the Controlword from 7 to 15. [Controlword] → [Online] → [7], [15]



(7) Confirm that the Status word is 0x1237 (Hex) and LED1 is lit.

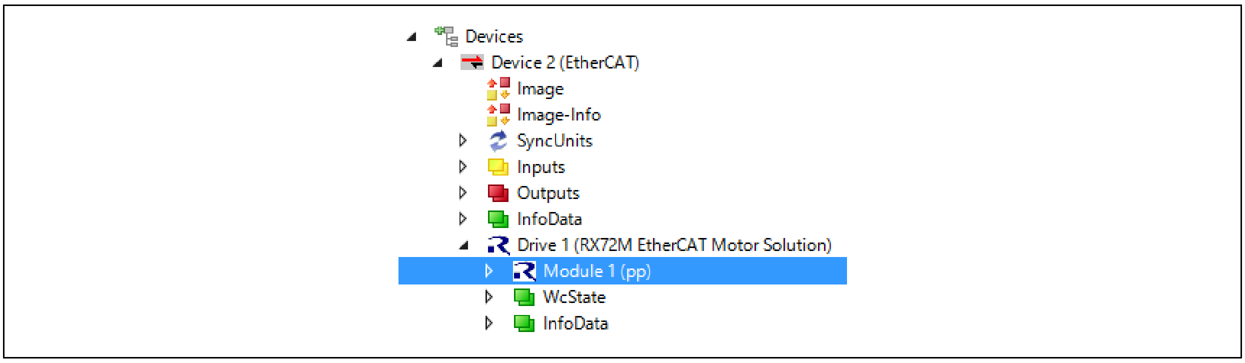


(8) Position control is possible by entering a value in Target position.

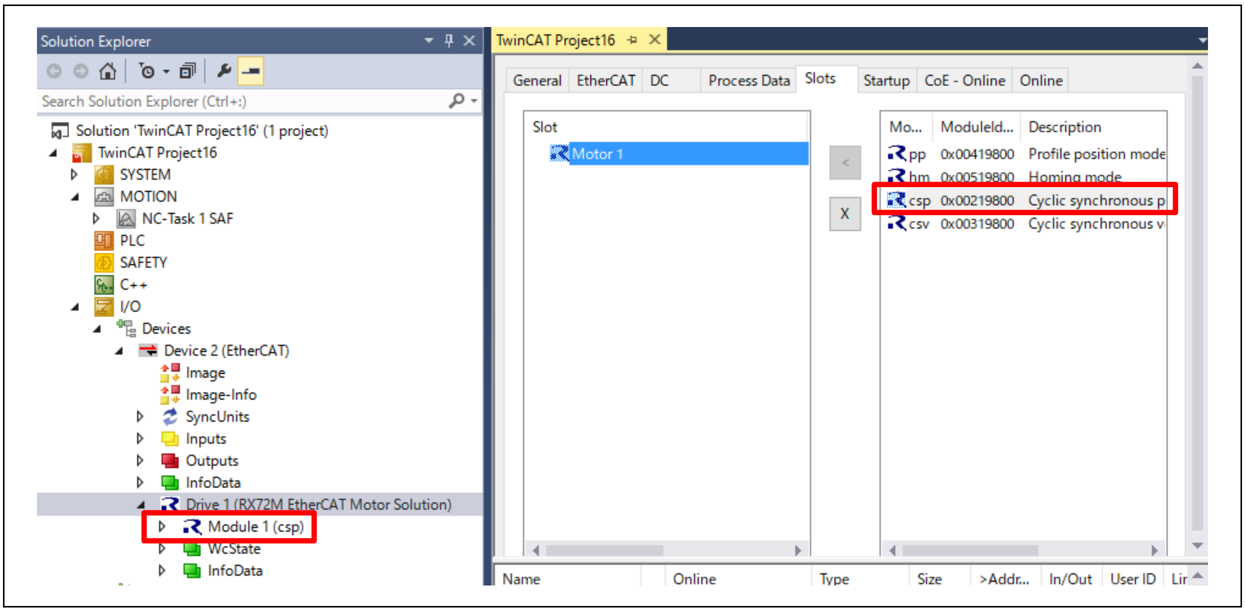


(9) Check the operation of the csp, (csv) profile.

(10) Remove the pp profile from the device's Slots and add the csp, (csv) profile.

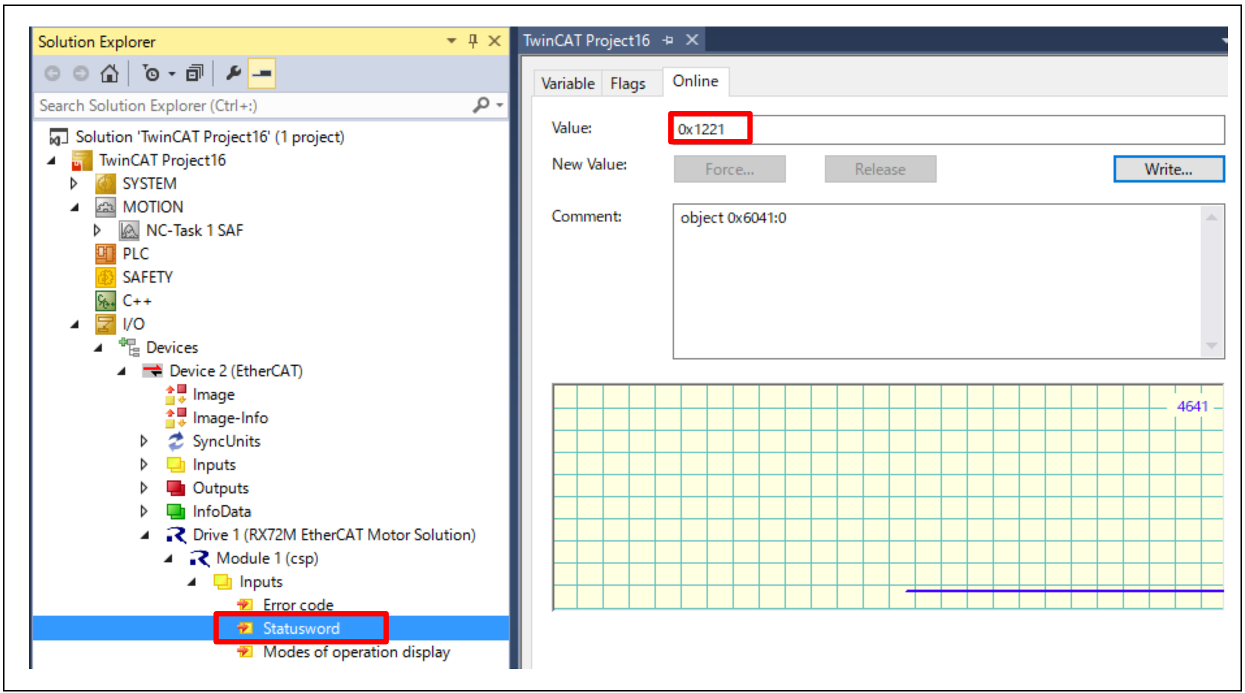


(11) Reload the Device.

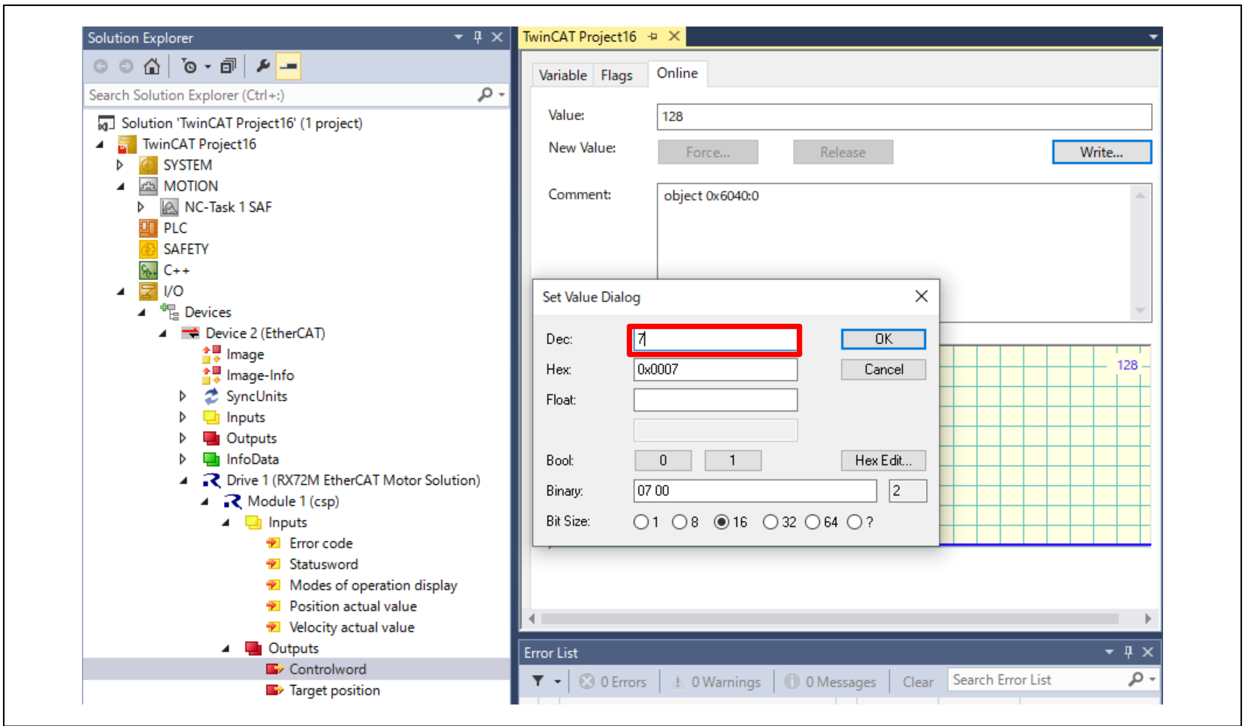


(12) Set the Control word to 128 (Dec).

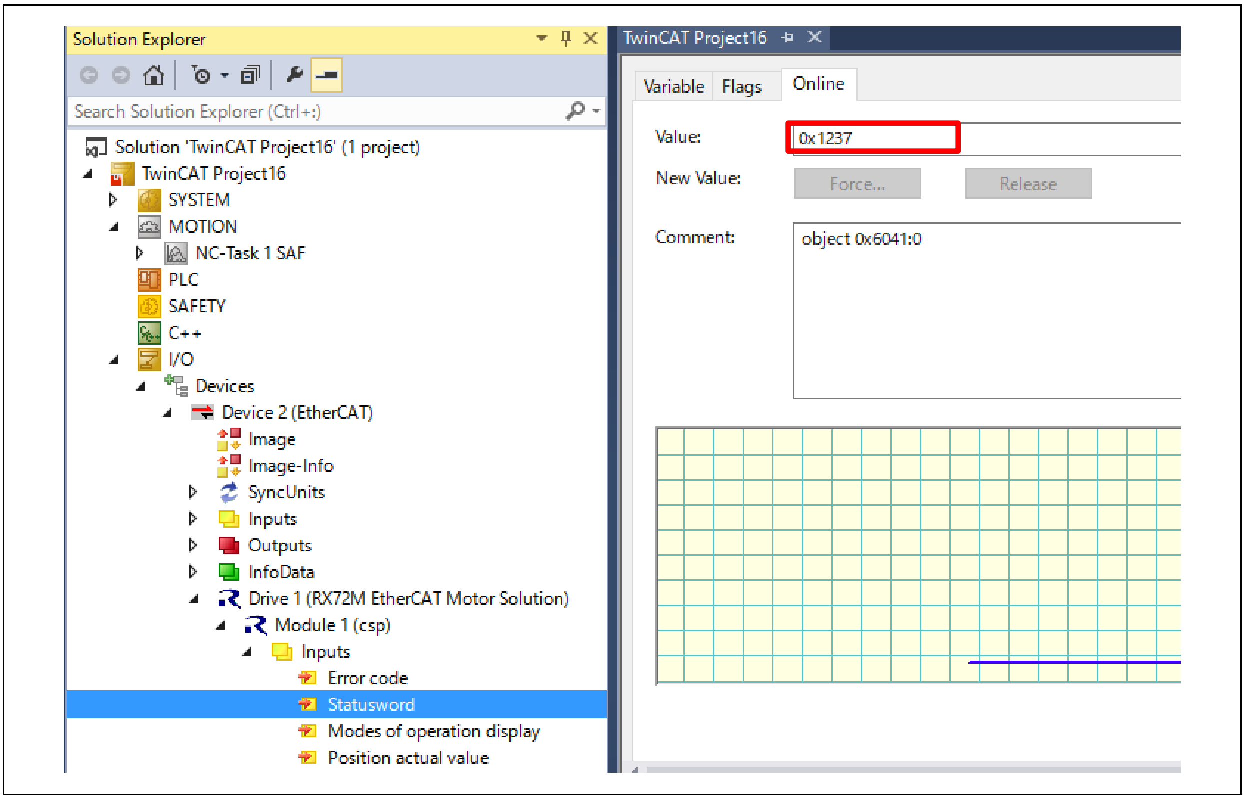
(13) Confirm that the Status word is 0xX221 (Hex).



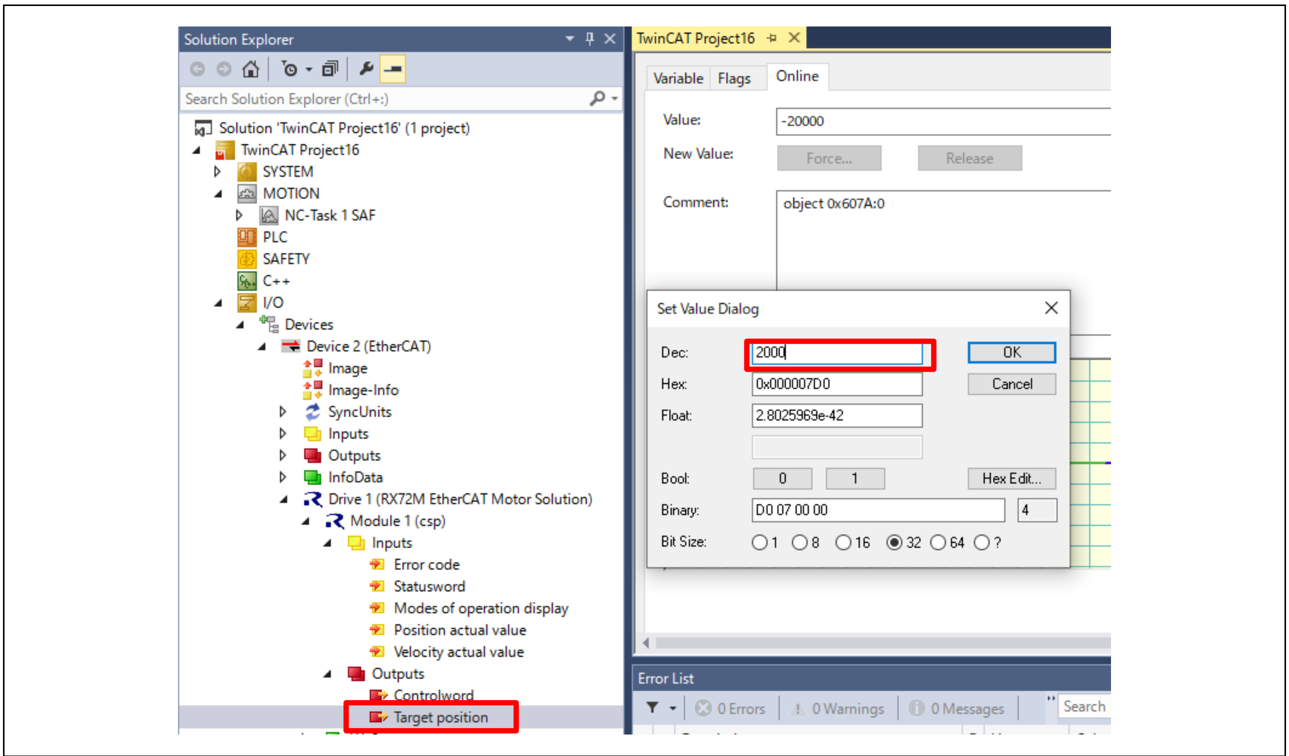
(14) Set the Controlword from 7 to 15. [Controlword] → [Online] → [7], [15]



(15) Confirm that the Status word is 0x1237 (Hex) and LED1 is lit.



(16) Position control is possible by entering a value in Target position.



7. Documents for Reference

User's Manual: Hardware

RX72M Group User's Manual: Hardware (Document No. R01UH0804)
Renesas Starter Kit+ for RX72M User's Manual (Document No. R20UT4383)
RX72M Group Communications Board Hardware Manual (Document No. R01AN4661)
(Download the latest version from the Renesas Electronics website.)

Startup Manual

RX72M Group RSK Board EtherCAT Startup Manual (Document No. R01AN4689)
RX72M Group Communications Board EtherCAT Startup Manual (Document No. R01AN4672)
(Download the latest version from the Renesas Electronics website.)

Technical Updates/Technical News

(Download the latest version from the Renesas Electronics website.)

User's Manual: Development Environment

RX Family C/C++ Compiler, Assembler, Optimizing Linkage Editor Compiler Package (R20UT0570)
(Download the latest version from the Renesas Electronics website.)

8. APPENDIX

8.1 About EtherCAT FIT module

Section describes the points to note regarding the introduction of the EtherCAT FIT module when creating a new project such as porting to a custom board.

EtherCAT FIT module (r_ecat_rx V.1.30) used in this sample program has been changed from the normal version to avoid contention between the motor control program and resources.

Normal version	This sample program	Setting
Use FIT CMT driver (r_cmt_rx) as timer	Use Config_TMR2 as a timer	Use Config_TMR2 to avoid timer channel conflicts

When creating a new project If you add the EtherCAT FIT module, the normal version will be added.

After adding the EtherCAT FIT module (r_ecat_rx), it is necessary to add the configurator.

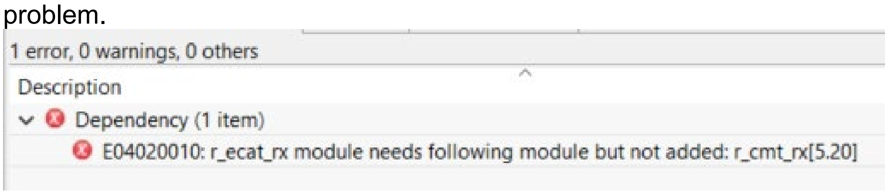
Component	Additional operation	Description
CMT driver r_cmt_rx	Remove from component	Configuration error is displayed but can be ignored
Compare match timer Config_CMT2	Add to component	Interval time: 1000us Check "Allow compare match interrupts (CMI2)"

After generating the code, copy the file from the target folder of this sample program.

Folder	File	Description
src/smc_gen/Config_CMT2	Config_CMT2_user.c	Add EtherCAT timer processing to interrupt callback function
src/smc_gen/r_ecat_rx	Files under the src folder.	Change to module source code to avoid resource contention

Please add “ECAT_CMT_CG_USE” in the definition of Preprocessor Macro of Compiler.

【Note】
The error message about dependency is displayed by deleting “r_cmt_rx” from component, but this is no problem.

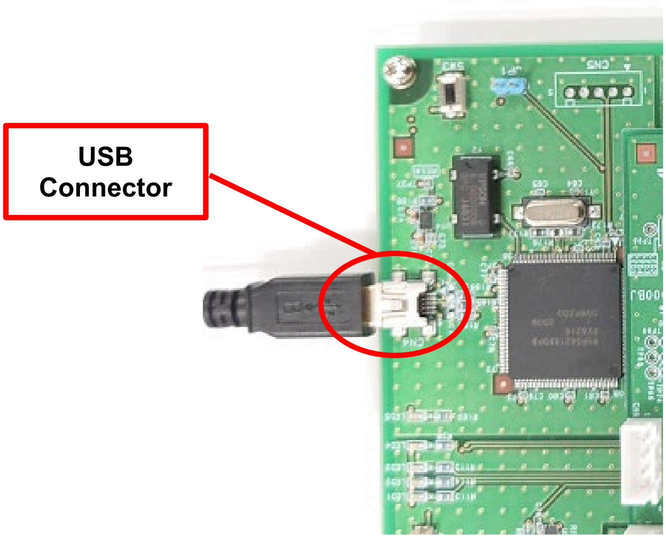


8.2 This motor board can be used with the RMW (Renesas Motor Workbench).

RMW (Renesas Motor Workbench) related procedure
Download RMW from the following Web page
https://www.renesas.com/jp/ja/solutions/proposal/motor-control.html#kits
Proceed through the preparations according to Table 2.1 in the RMW user's manual (R21UZ0004), which will be in the RMW folder.

Note: The authentication file can be downloaded from [Authentication file download] at the same link as the RMW.

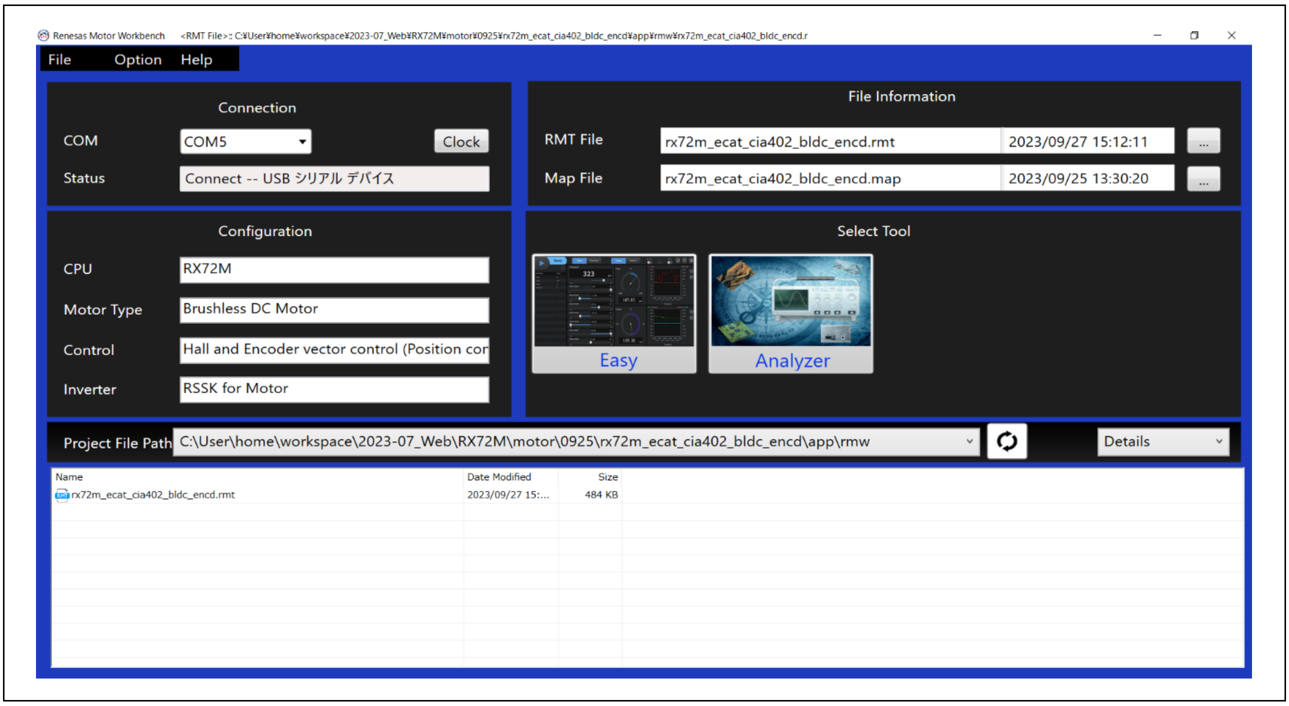
Use USB1 on the inverter board for the connection, and connect it to the USB port of the PC.



Start RMW and specify the following files.

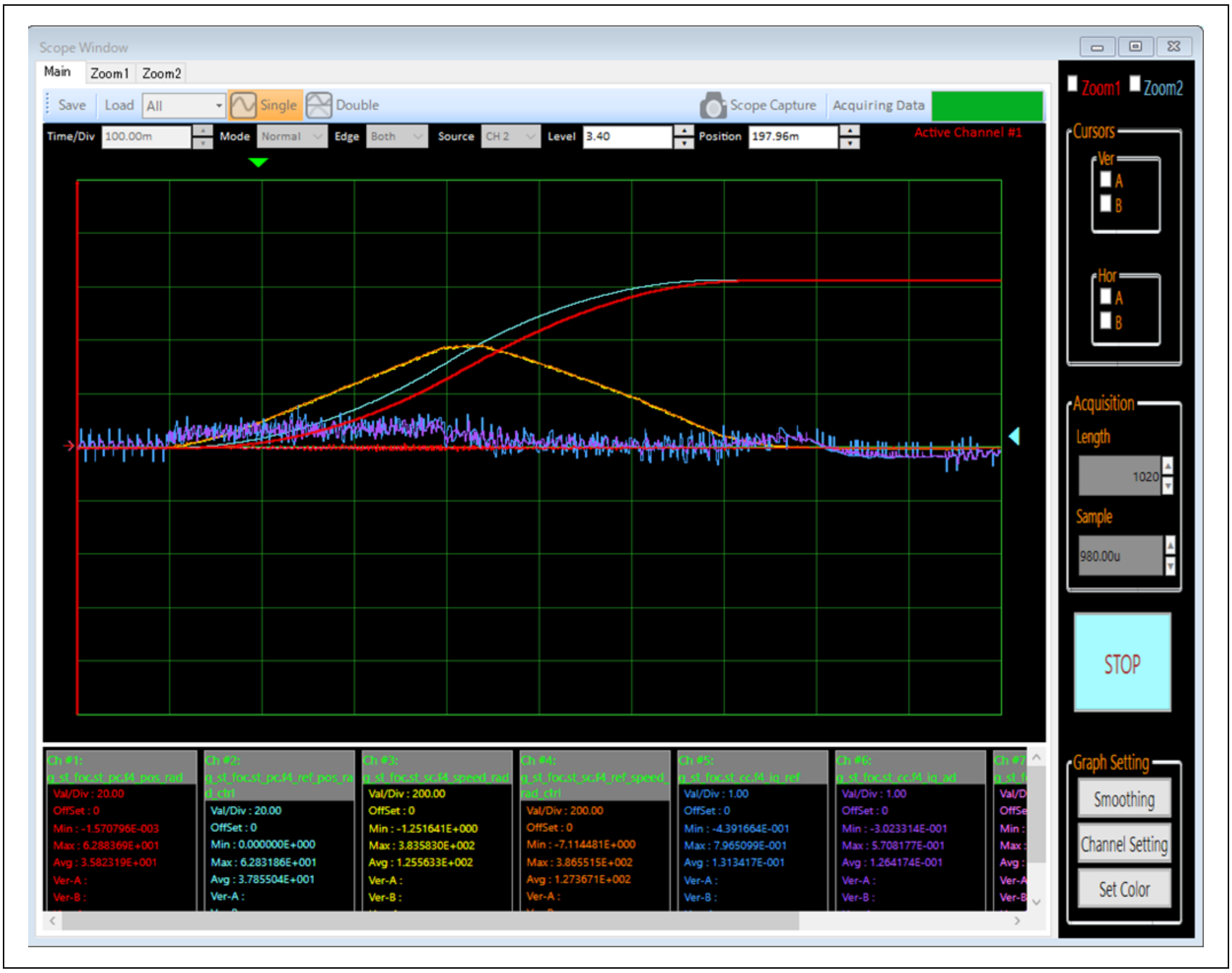
- Environment file
  rx72m_ecat_cia402_bldc_encd\project\app\rmw
  \rx72m_ecat_cia402_bldc_encd.rmt
-map file
  rx72m_ecat_cia402_bldc_encd\HardwareDebug
  \rx72m_ecat_cia402_bldc_encd.map

Connect RMW and the motor board
When Connection → COM is specified, the following is displayed if the connection was made correctly.



Get the motor drive waveforms.
[Analyzer] → Press the [RUN] button in the [Scope Window]

Note: The example below shows the waveforms when Target Position is changed from 0 to 40000.



Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Apr 28, 2022	—	First edition issued
1.10	Nov 30, 2023	Program	Supported EtherCAT FIT Module version 1.30
			Added utilities and project folders to separate SSC-related files and project-related files
			Supported e2 studio 64-bit version
			Supported SSC 5.13
			Supported CTT V2.4.0
		4	Changed the version of the operation environments in "Table 1-1 Operation Environment"
		5	Changed the version of the base project in "Table 1-2 Base Projects and Changes that were Required"
		10	Changed the folder name and Added new files in "Table 2 5 Software file structure" due to changing folder sturucture
		18	Changed a part of Object and Added Sub column in "Table 3 3 Object Dictionary Supported by the Sample Program"
		22	Added motor control API "p_api" in "5.1 Overview"
		29	Added "5.8 p_api"
		31	Changed the version of the operationg environments in "Table 6-1 Operating Environment"
		32	Added and deleted a part of component, changed the component to the latest version in "Table 6-2 Operation confirmed component"
		31, 32, 35, 36, 41	Changed the folder name due to changing folder sturucture
		39	Added the notes regarding building the programs
		53	Changed the description due to using EtherCAT FIT module version 1.30 in "8.1 About EtherCAT FIT module"
		55	Changed the folder name due to changing the folder sturucture and replace the screen capture of RMW

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.