RX72M Group

R01AN5434EJ0110
Rev.1.10
Aug.31.2020

Single-Chip Motor Control via EtherCAT Communications

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 a 24-V system inverter board.

1.2 Operation Environment

Table 1-1 Operation Environment

Target MCU	RX72M Group
Evaluation board	Manufactured by Renesas RX72M CPU card + 24-V system inverter board *
Integrated development environment (IDE)	Renesas e2 studio, V.7.5.0 or later IAR Embedded Workbench for Renesas RX 4.13.1 or later
C compiler	Renesas C/C++ compiler package for RX Family V3.01.00 or later
	GCC for Renesas RX 4.8.4.201803 or later
	IAR C/C++ compiler for Renesas RX version 4.13.1 or later
Motor	Permanent magnet synchronous motor with an incremental encoder from Leadshine Technology BLM57050-1000
Emulator	Renesas e2 Lite
Communication protocol	EtherCAT
SSC tool	Provided by the EtherCAT Technology Group (ETG) Slave Stack Code (SSC) tool Version 5.12
Software PLC	TwinCAT® 3 (download this from the Beckhoff web site) of Beckhoff Automation
	CODESYS of 3S-Smart Software Solutions

Note: * The 24-V motor control system manufactured by Renesas Electronics incorporates both items.

24V Motor Control Evaluation System for RX23T (RTK0EM0006S01212BJ)

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_MRSSK_SPM_ENCD_FOC_E2S_RV100 (r01an5386ej0100-rx72m-motor)	● 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 (r01an4672ej0100-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 24-V system 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 + 24-V system inverter board	ecat_cia402_motor_rsskrx72m

2. System Overview

2.1 Hardware Configuration

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

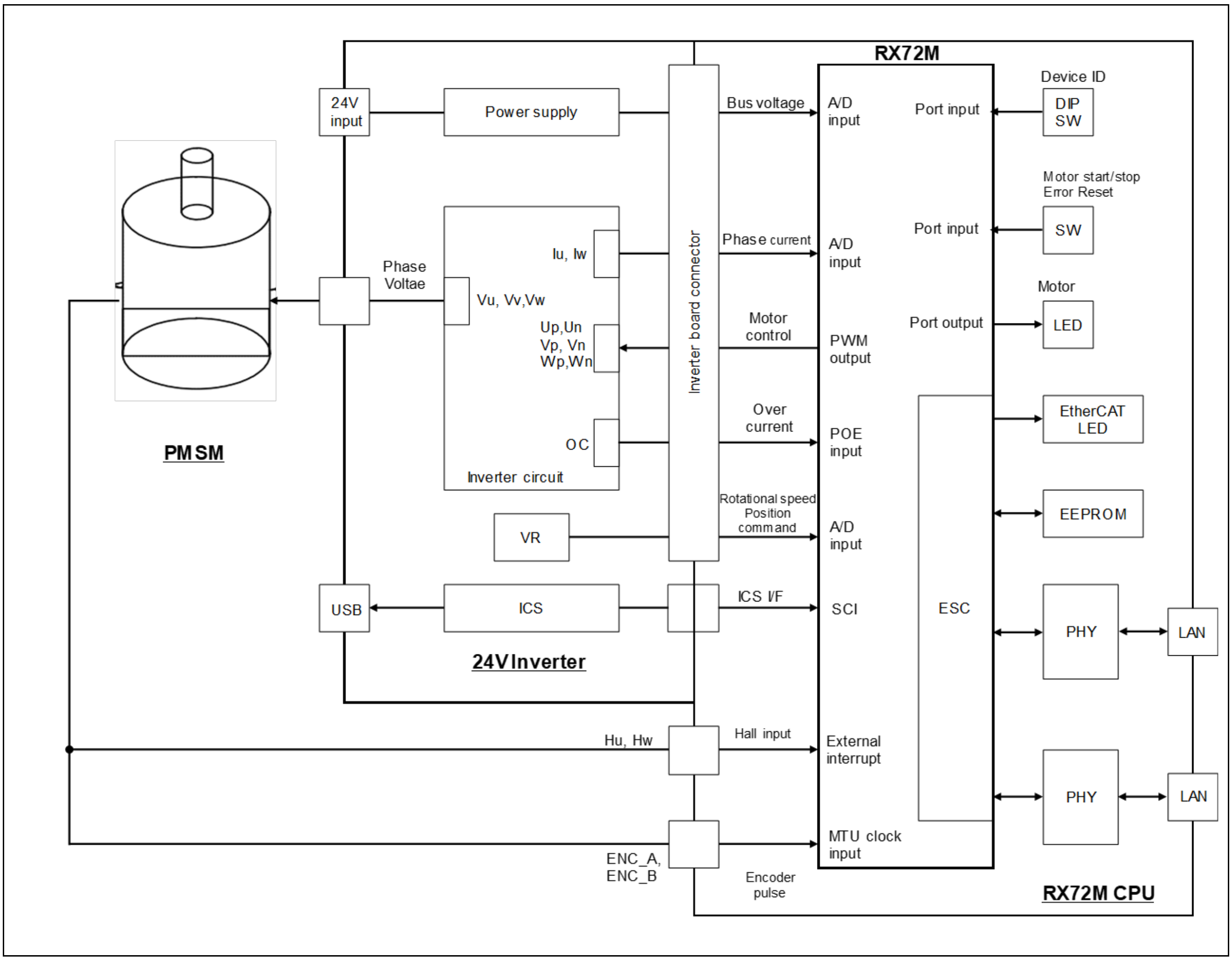


Figure 2-1 Hardware Configuration

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
P43 / AN003	Inverter's main line voltage measurement
P47 / AN007	For input of the rotational velocity and position command values (analog values)
P30	START/STOP toggle switch
P02	ERROR RESET toggle switch
P71	LD1 on/off control
PN4	LD2 on/off control
PH0	LD3 on/off control
P40 / AN000	U-phase current measurement
P42 / AN002	W-phase current measurement
PE1 / MTIOC3B	PWM output (U_p)
PE2 / MTIOC4A	PWM output (V_p)
PE3 / MTIOC4B	PWM output (W_p)
PE0 / MTIOC3D	PWM output (U_n)
PE5 / MTIOC4C	PWM output (V_n)
PE4 / MTIOC4D	PWM output (W_n)
P31 / IRQ1	Hall U-phase input
PD3 / IRQ3	Hall V-phase input
PB0 / IRQ12	Hall W-phase input
PA4 / MTCLKA	Encoder A-phase input
P25 / MTCLKB	Encoder B-phase input
PC4 / POE0#	Input for the emergency signal for stopping the PWM output on detection of an overcurrent

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 transmit data output (bit 2)
PM5/CAT0_ETXD3	4-bit transmit data output (bit 3)
PL4/CAT0_ETXD0	4-bit transmit data output (bit 0)
PL5/CAT0_ETXD1	4-bit transmit data output (bit1)
PK5/CAT0_ERXD3	4-bit receive data input (bit 3)
PK4/CAT0_ERXD2	4-bit receive data input (bit 2)
P74/CAT0_ERXD1	4-bit receive data input (bit 1)
P75/CAT0_ERXD0	4-bit receive data input (bit 0)
PL7/CAT0_MDIO	Management data I/O
PN3/CAT1_RX_ER	Receive error input
P84/CAT1_LINKSTA	Link status input from the PHY-LSI
PQ2/CAT1_RX_DV	Received data valid input
PL6/CAT0_TX_EN	Transmit enable output
PN2/CAT1_TX_CLK	Transmit clock input
PH4/CATLEDSTER	Output for RUN LED part of STATE LED (bicolor) (turned off while ERR)
PH5/CATLATCH0	LATCH signal input
PH6/CATLATCH1	LATCH signal input
P27/CATIRQ	IRQ output
PQ7/CAT1_TX_EN	Transmit enable output
PK2/CAT0_RX_DV	Received data valid input
PM1/CAT1_ERXD1	4-bit receive data input (bit 1)
PM2/CAT1_ERXD2	4-bit receive data input (bit 2)
PM3/CAT1_ERXD3	4-bit receive data input (bit 3)
PL2/CAT0_RX_ER	Receive error input
PM0/CAT1_ERXD0	4-bit receive data input (bit 0)
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 transmit data output (bit 3)
PQ5/CAT1_ETXD0	4-bit transmit data output (bit 0)
PN0/CAT1_ETXD2	4-bit transmit data output (bit 2)
PQ6/CAT1_ETXD1	4-bit transmit data output (bit 1)
P11/CATSYNC1	SYNC1 signal output
PM6/CAT0_TX_CLK	Transmit clock input
PK0/CAT0_MDC	Management data clock output
P34/CAT0_LINKSTA	Link status input from the PHY-LSI
P82/CATI2CDATA	EEPROM I2C data input/output

Table 2-4 Other Pin Interface

Pin Name	Description
P12/RXD2	SCI2 receive data input pin
P13/TXD2	SCI2 transmit data output pin
PH2	Device ID DIP SW (bit 0)
PQ3	Device ID DIP SW (bit 1)
P05	Device ID DIP SW (bit 2)
P72	Device ID DIP SW (bit 3)
PC1	Device ID DIP SW (bit 4)
PN5	Device ID DIP SW (bit 5)

2.3 Software Configuration

2.3.1 Software File Configuration

Folders and files configured for the sample program are listed in Table 2-5 to Table 2-7.
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 Configuration of Files for the Motor Control Program (1)

Directory motor/			File	Description
application/	main/		main.h, main.c	Main function
	user_interface/	ics/	r_mtr_ics.h, r_mtr_ics.c	ICS related function definition
			ICS_RX72M.h ICS_RX72M.lib	Communications-related definitions for tools Communications library for tools
		board/	r_mtr_board.h, r_mtr_board.h	Board user function definition
middle/	interface/		r_mtr_driver_access.h r_mtr_driver_access.c	User access function definition
			r_mtr_driver_ecat_access.h **r_mtr_driver_ecat_access.c**	Definitions of functions for access by the EtherCAT communications program
	common/		r_mtr_common.h	Common definition
			r_mtr_units.h	Definitions of units
			r_mtr_filter.h, r_mtr_filter.c	General-purpose filter function definition
			r_mtr_fluxwkn.h r_mtr_fluxwkn.obj	Definition of a function related to magnetic-flux weakening control
			r_mtr_pi_control.h r_mtr_pi_control.c	PI control function definition
			r_mtr_transform.h r_mtr_transform.c	Coordinate transformation function definition
			r_mtr_mod.h, r_mtr_mod.c	Modulation function definition
			r_mtr_volt_err_comp.h r_mtr_volt_err_comp.obj	Voltage error compensation function definition
			r_mtr_statemachine.h r_mtr_statemachine.c	State machine function definition
	control/		r_mtr_parameter.h	Definitions of various parameters
			r_mtr_ctrl_gain_calc.obj	Control gain calculation function definition
			r_mtr_foc_action.c	Action function definition
			r_mtr_interrupt_carrier.c	Carrier interrupt function definition
			r_mtr_interrupt_timer.c	Cycle interrupt function definition
			r_mtr_interrupt_sensor.c	Sensor input interrupt function definition
			r_mtr_foc_control_encd_position.h r_mtr_foc_control_encd_position.c	FOC function definition

Table 2-6 Configuration of Files for the Motor Control Program (2)

Directory motor/		File	Description
middle/	control/	r_mtr_foc_current.h r_mtr_foc_current.c	Current control function definition
		r_mtr_foc_speed.h r_mtr_foc_speed.c	Velocity control function definition
		r_mtr_foc_position.h r_mtr_foc_position.c	Position control function definition
		r_mtr_position_profiling.h r_mtr_position_profiling.c	Function for creating position command values
		r_mtr_ipd.h r_mtr_ipd.obj	IPD control function definition
		r_mtr_speed_observer.h r_mtr_speed_observer.obj	Velocity observer function definition
driver/	inverter/	r_mtr_ctrl_mrssk.h r_mtr_ctrl_mrssk.c	Inverter board dependent function definition
	mcu/	r_mtr_interrupt.c	Interrupt function definition
		r_mtr_ctrl_rx72m.h r_mtr_ctrl_rx72m.c	MCU specific function definition
		r_mtr_ctrl_mcu.h	MCU common definition
	sensor/	r_mtr_ctrl_encoder.h r_mtr_ctrl_encoder.c	Encoder function definition
		r_mtr_ctrl_hall.h r_mtr_ctrl_hall.c	Hall function definition
config/		r_mtr_config.h	Configuration common definition
		r_mtr_motor_parameter.h	Motor parameter configuration definition
		r_mtr_inverter_parameter.h	Inverter parameter configuration definition
		r_mtr_control_parameter.h	Control parameter configuration definition
		r_mtr_encoder_parameter.h	Encoder parameter configuration definition

Table 2-7 Configuration of Files for the EtherCAT Communications Program (1)

Directory smc_gen/r_ecat_rx/			File	Description
./			r_ecat_rx_if.h	FIT module API definition
			readme.txt	Attached document regarding the FIT module
./doc/	en/		r01an4881ejxxxx-rx-ecat.pdf	Application Note (English version)
	ja/		r01an4881jjxxxx-rx-ecat.pdf	Application Note (Japanese version)
./ref/			r_ecat_rx_config_reference.h	Default definitions of options for the FIT module
./src/	hal/		renesashw.h renesashw.c	ESC access function definition
	phy/		phy.h, phy.c	PHY control function definition
	targets/	rx72m/	r_ecat_setting_rx72m.c	MCU specific function definition
./utilities/	rx72m/	batch_files/	apply_patch.bat	Batch file for modifying the SSC source file
			RX72M_Motor_YYMMDD.patch	Modification patch for the single-chip motor control program
		esi/	RX72M EtherCAT MotorSolution.xml	ESI file
		ssc_config/	Renesas_RX72M_config.xml	SSC configuration file
			RX72M EtherCAT CiA402.esp	SSC tool project file

Table 2-8 Configuration of Files for the EtherCAT Communications Program (2)

Directory application/	File	Description
./ecat	cia402sample.h cia402sample.c	CiA402 application definition
	SSC source file	Stored after applying the batch file

2.3.2 Software Module Configuration

Figure 2-2 shows the module configuration of the motor control program.
The files added or modified to control the motor control base project by EtherCAT communication are enclosed in a red frame.
The frames with solid red lines indicate the files that have been added, and the frame with a broken red line indicates the files that have been changed.
The major changes are listed below according to the module layer.

Table 2-9 Changes According to the Module Layer

Layer / Module	Related File	Description of File
Application layer / EtherCAT application	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_ecat_dirver_acces.c	API functions for the interface between the motor control program and the EtherCAT communications program.
Middle layer / Control module	r_mtr_foc_control_encd_position.c r_mtr_foc_speed.c r_mtr_foc_position.c	These files convert the units of velocity [count/s] and position [count] used in the CiA402 to the units of velocity [rad/s] and position [rad] used in the motor control program.

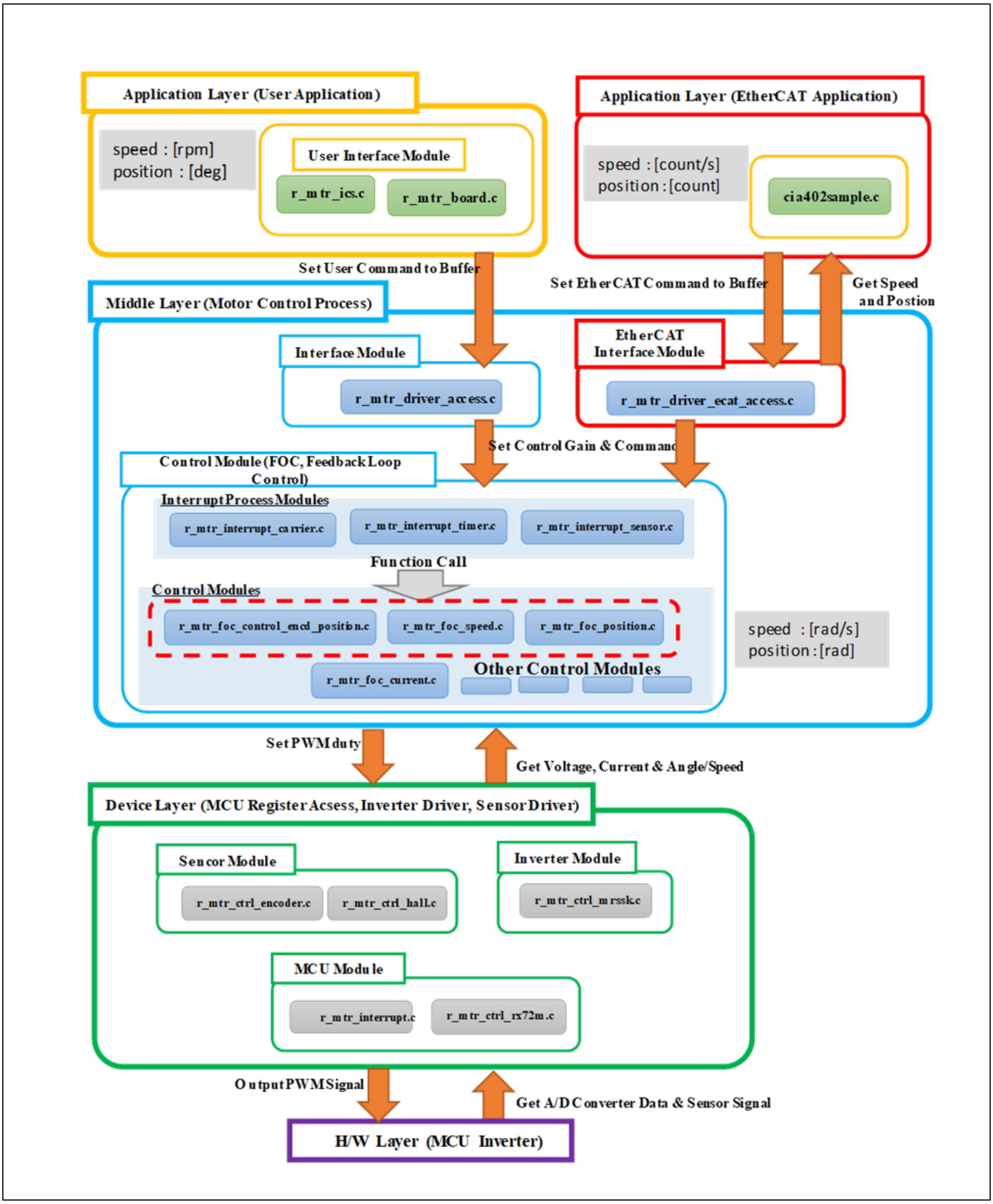


Figure 2-2 Module Configuration of the Motor Control Program

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

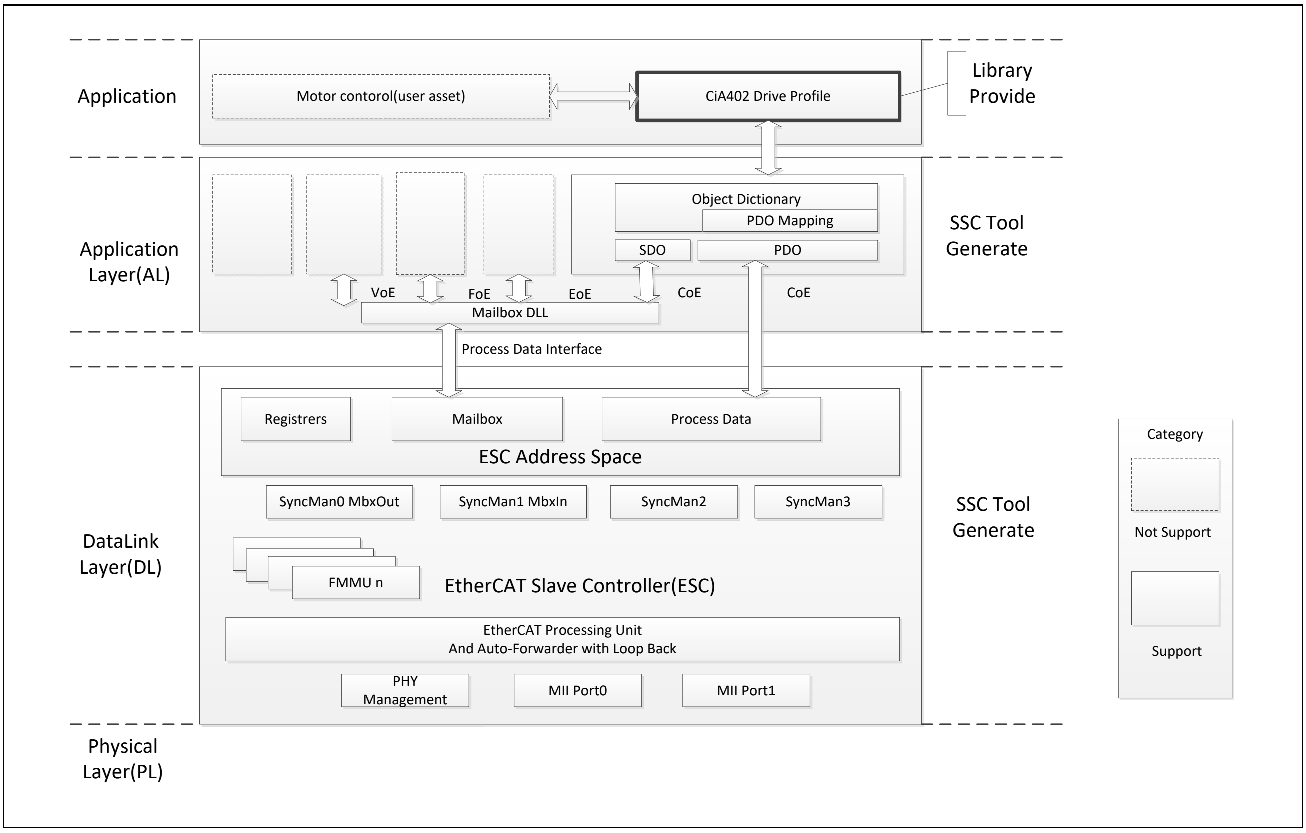


Figure 2-3 Module Configuration of the EtherCAT Communications Program

2.4 Software Specifications

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

Table 2-10 Basic Specifications of the Motor Control Program

<table>
<tr><th>Item</th><th colspan="2">Description</th></tr>
<tr><td>Control method</td><td colspan="2">Vector control</td></tr>
<tr><td>Detection of rotor's magnetic pole position</td><td colspan="2">Incremental encoder (A phase, B phase), hall sensor (UVW phase)</td></tr>
<tr><td>Input voltage</td><td colspan="2">DC 24 V</td></tr>
<tr><td>Carrier frequency (PWM)</td><td colspan="2">20 [kHz] (carrier period: 50 [μs])</td></tr>
<tr><td>Dead time</td><td colspan="2">2 [μs]</td></tr>
<tr><td>Control period (current)</td><td colspan="2">50 [μs]</td></tr>
<tr><td>Control period (velocity, position)</td><td colspan="2">500 [μs]</td></tr>
<tr><td rowspan="3">Range of position command values</td><td>Board UI</td><td>-180° to 180°</td></tr>
<tr><td>ICS UI</td><td>-32768° to 32767°</td></tr>
<tr><td>ETH UI</td><td>-2 147 483 648 [count] to 2 147 483 647 [count] *2</td></tr>
<tr><td>Range of velocity command values</td><td colspan="2">CW: 0 [rpm] to 2000 [rpm]
CCW: 0 [rpm] to 2000 [rpm]</td></tr>
<tr><td>Positional resolution</td><td colspan="2">0.3° (encoder pulse: 1000 [ppr], after multiplication by 4: 4000 [cpr])</td></tr>
<tr><td>Positional dead zone *1</td><td colspan="2">Encoder incrementing or decrementing by one t (±0.09°)</td></tr>
<tr><td>Frequencies specific to the control systems</td><td colspan="2">Current control system: 300 Hz
Velocity control system: 30 Hz
Position control system: 10 Hz</td></tr>
<tr><td>Protection stop processing</td><td colspan="2">The motor control signal outputs (6 lines) are set to the inactive level in response to any of the following four conditions.
The current in any phase exceeds 3.82 A (monitored once every 50 μs).
The inverter’s main line voltage exceeds 28 V (monitored once every 50 μs).
The inverter’s main line voltage falls below 14 V (monitored once every 50 μs).
The rotational speed exceeds 3000 rpm (monitored once every 50 μs).

The PWM output pins are placed in the high-impedance state, when external input of an overcurrent signal is detected (indicated by a falling edge on the POE0# pin) or when an output short circuit is detected.</td></tr>
</table>

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

Note 2. [unit] here indicates the numbers counted from the encoder.

Table 2-11 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.

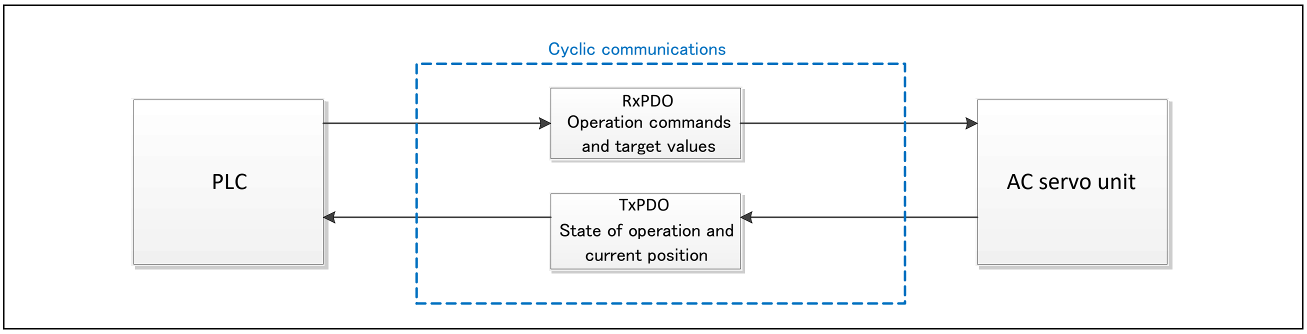


Figure 3-1 Flow of CiA402 Communications

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.

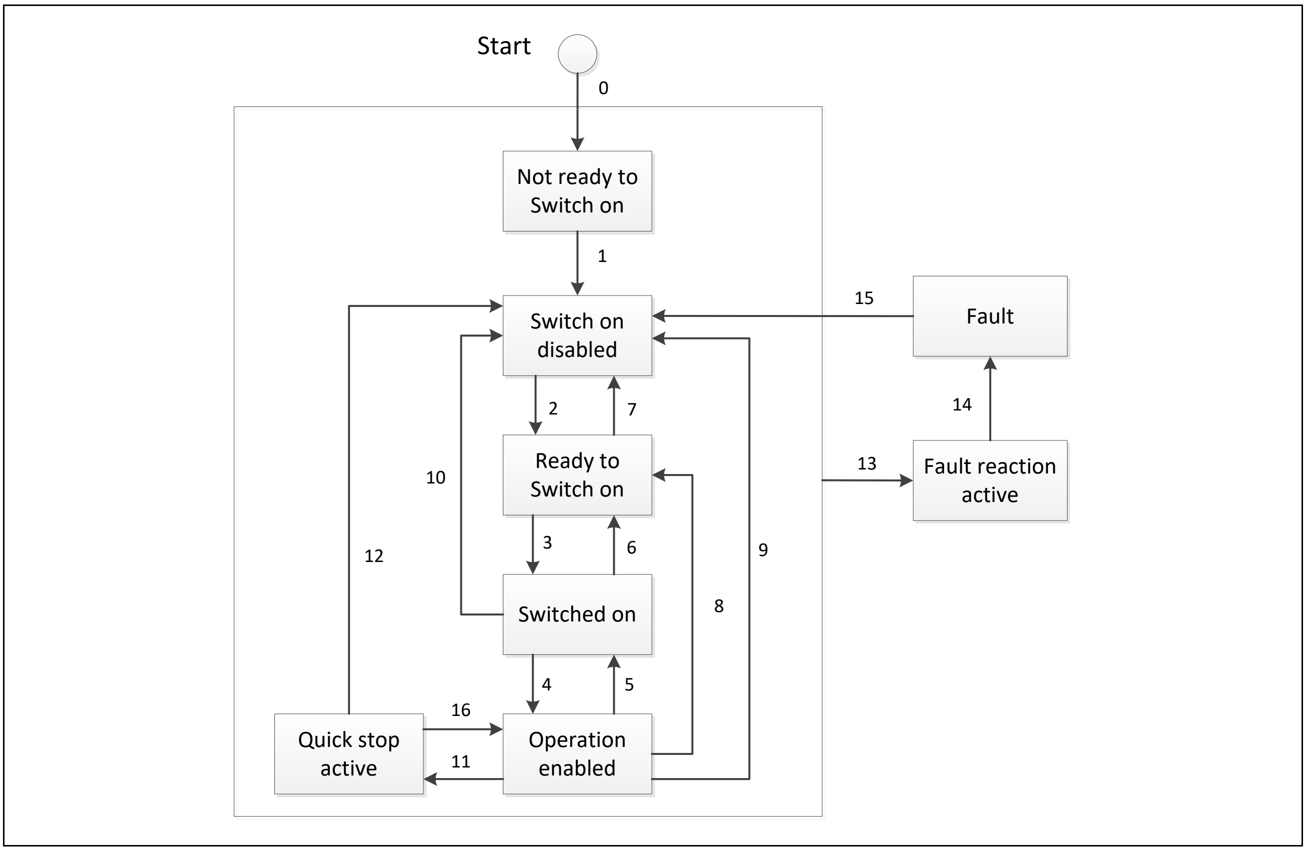


Figure 3-2 CiA402 State Transition Diagram

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	Object Name	Category	Access	Data Type	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

Index	Object Name	Category	Access	Data Type	PDO Mapping
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.

4.1 Velocity Parameters

In the sample program, the parameter serving as the unit of velocity is defined as the number counted from the encoder per control cycle. Since the control cycle is very short, the velocity values are handled as fixed-point numbers in 16.16 bit format after being multiplied by 2^{16}, i.e. 65536.
For the conversion of the unit of velocity produced by counting from the encoder per second to the corresponding control parameter, the number has to be multiplied by the number of control cycle time slices in a second and then by 65536.
Since the control cycle of the sample program is 500 us, the velocity derived from 5000 being counted by the encoder in a second is converted by the formula:
5000 × 0.0005 × 65536 = 163840
So, the corresponding velocity value is 163840.

4.2 Acceleration Parameters

In the sample program, the unit of acceleration parameter is defined as the number counted by the encoder per square of the control cycle. In a similar way to the calculation and handling of velocity, the acceleration and deceleration values are handled as fixed-point numbers in 16.16 bit format after being multiplied by 2^{16}, i.e. 65536.
For the conversion of unit from the number counted by the encoder count per second to the control parameter, the number has to be multiplied by the square of the position cycle time slice and then by 65536.
Since the control cycle is 500 us, the acceleration derived from 5000 being counted by the encoder in a second is converted by the formula:
5000 × 0.0005 × 0.0005 × 65536 = 81
So, the corresponding acceleration value is 81.

4.3 Conversion of Units by the RMW

The control parameters such as gain obtained through tuning by the motor control development support tool "Renesas Motor Workbench" are reflected in the source code of the motor control program, so the same values are also used in control via EtherCAT.

On the other hand, as the position and velocity command values that the RMW uses are in different units from those specified for use as motion control parameters set in the CiA402 objects, conversion is required. Table 4-1 lists the formulae used to convert command values used in the RMW into CiA402 object command values.

Table 4-1 Command Value Conversion Formulae: RMW to CiA402

Item	RMW Command Value	CiA402 Command Value	Formulae	Data Type
Position	Θ[deg]	P_C [count]	$P_C = \Theta \div 360 \times CPR$	INT32
Velocity	R[rpm]	S_C [count/s]	$SC = R \div 60 \times CPR \times T_S \times K_Q$	INT32 (16.16)
Acceleration	t_{ACC}[s] *1	A_C [count/s^2]	$AC = R \div t_{ACC} \div 60 \times CPR \times T_S \times T_S \times K_Q$	UINT32 (16.16)

The symbols used in the above formulae, other than standard ones and those with meanings stated in the table, are explained in Table 4-2.

Table 4-2 Symbols in the Conversion Formulae

Symbol	Description	Value
CPR	Number counted per rotation [count]	4000
T_S	Control cycle [s]	0.0005
K_Q	16.16 fixed point conversion coefficient	2^{16} = 65536

Note 1. Regarding acceleration in the RMW
RMW uses the time [s] to reach the target velocity as its definition of the acceleration.
Figure 4-1 shows how the acceleration changes from that for acc1 to that for acc2 by changing the acceleration time to reach the target velocity from t1 to t2.
The acceleration at those times can be expressed by dividing the velocity by the acceleration times.
  acc1 = speed ÷ t1
  acc2 = speed ÷ t2

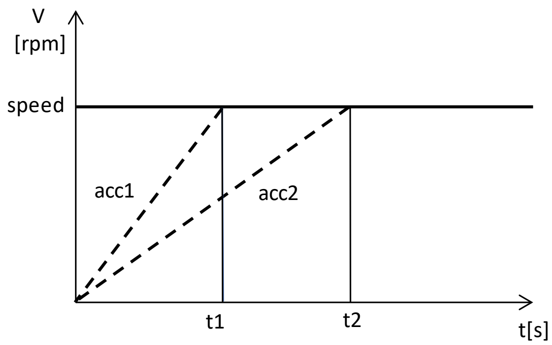


Figure 4-1 Acceleration Times and Acceleration

Examples of conversion:

・Position command value
Convert Θ = 180[deg] to P_C[count].
180 ÷ 360 × 4000 = 2000
・Velocity command value
Convert R = 2000[rpm] to S_C[count/s].
2000 ÷ 60 × 4000 × 0.0005 × 65536 = 4369066
・Acceleration command value
When R = 2000[rpm], convert T_{ACC} = 0.3[s] to A_C[count/s^2].
2000 ÷ 0.3 ÷ 60 × 4000 × 0.0005 × 0.0005 × 65526 = 7281

5. API Functions

5.1 Overview

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

Functions	Description
R_MTR_InitControl	Initializes the motor control program.
R_MTR_SetUserifMode	Enables or disables automatic updating of the command values for position and velocity.
R_MTR_ExecEvent	Issues indicators of events with regard to the state of system operation.
R_MTR_ChargeCapacitor	Waits for the inverter's main line voltage to become stable.
R_MTR_GetLoopModeStatus	Gets the setting for the control loop mode.
R_MTR_SetPositionStatus	Sets the input method of the position command value.
R_MTR_SetPosition	Sets the position command value in degrees.
R_MTR_GetPosition	Gets the value in degrees of the current position.
R_MTR_GetPositioningFLag	Gets the value of the positioning completed flag.
R_MTR_SetSpeed	Sets the velocity command value in rpm.
R_MTR_GetSpeed	Gets the current velocity value in rpm.
R_MTR_SetDir	Sets the direction of the motor's rotation.
R_MTR_GetDir	Gets the currently set direction of motor rotation.
R_MTR_GetStatus	Gets the state of system operation of the motor control program.
R_MTR_InputBuffParamReset	Sets the buffer of variables for ICS input to the default value.
R_MTR_CtrlInput	Copies an ICS input variable to the buffer of variables for ICS input.
R_MTR_SetVariables	Sets the contents of the buffer of variables for ICS input as the motor control parameters.
R_MTR_AutoSetVariables	Sets the position and velocity command values in the buffer of variables for ICS input as the given motor control parameters.
R_MTR_CtrlGainCalc	Calculates the gain for the motor control parameters.
R_MTR_UpdatePolling	Polls the write enable flag among the motor control parameters.
R_MTR_GetErrorStatus	Gets the error state of the encoder position and velocity control.
R_MTR_GetPositionPFStatus	Gets the profile state of encoder position control.
R_MTR_SetPositionUnits	Sets the position command value (number counted).
R_MTR_SetActualPositionUnits	Sets the current position (number counted).
R_MTR_GetPositionUnits	Gets the value (number counted) corresponding to the current position.
R_MTR_GetSpeedUnits	Gets the current velocity value (number counted per second).
R_MTR_SetAccelerationUnits	Sets the velocity command value in count/s^2.
R_MTR_SetDecelerationUnits	Sets the deceleration command value in count/s^2.

5.2 R_MTR_InitControl

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

Format

void R_MTR_InitControl (uint8_t u1_id)

Parameters

u1_id
Specifies the ID of the motor to be controlled.
Note: The sample program can only control a single motor.
MTR_ID_A /* Motor A*/

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

This function initializes the state of system operation and sets the default values of the motor control parameters.

Example

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

5.3 R_MTR_SetUserifMode

This function enables or disables automatic updating of the command values for position and velocity specified in the user interface module.

Format

void R_MTR_SetUserifMode (uint8_t u1_user_mode)

Parameters

u1_user_mode

Enables or disables automatic updating.
MTR_DISABLE_AUTO_SET(0)
MTR_ENABLE_AUTO_SET(1)

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

This function enables or disables periodic automatic updating in response to a signal from a timer.
Automatic updating is for the board user interface.
The default value in the sample program is for automatic updating to be disabled.

Example

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

5.4 R_MTR_ExecEvent

This function issues indicators of events with regard to the state of system operation of the motor control program.

Format

void R_MTR_ExecEvent (uint8_t u1_event, uint8_t u1_id)

Parameters

u1_event

Event Name	Value	Trigger Source
MTR_EVENT_INACTIVE	0x00	This indicator corresponds to torque through the motor being turned on.
MTR_EVENT_ACTIVE	0x01	This indicator corresponds to torque through the motor being turned off.
MTR_EVENT_ERROR	0x02	This indicator corresponds to the system detecting an error.
MTR_EVENT_RESET	0x03	This indicator corresponds to the operation being initialized or recovery from an error.

u1_id

Specifies the ID of the motor to be controlled.

MTR_ID_A /* Motor A*/

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

This function indicates changes to the state of system operation by issuing an event indicator as an argument.

Example

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

5.5 R_MTR_ChargeCapacitor

This function sets up a wait waits for the inverter's main line voltage to become stable. This function must be executed before starting motor control.

Format

void R_MTR_ChargeCapacitor(uint8_t u1_id)

Parameters

u1_id
Specifies the ID of the motor to be controlled.
MTR_ID_A /* Motor A*/

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

This function checks whether the inverter's main line voltage (VDC) has risen above 80% of 24-V DC, and waits until it has without timing out.

Example

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

5.6 R_MTR_GetLoopModeStatus

This function gets the current setting for the control loop mode.

Format

uint8_t R_MTR_GetLoopModeStatus(uint8_t u1_id)

Parameters

u1_id
  Specifies the ID of the motor to be controlled.
    MTR_ID_A /* Motor A*/

Return Values

  Control loop mode
    MTR_LOOP_SPEED(0): Velocity control
    MTR_LOOP_POSITION(1): Position control

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

The default value of the control loop mode in the sample program is MTR_LOOP_POSITION.

Example

```
uint8_t u1_status
u1_status = R_MTR_GetLoopModeStatus(MTR_ID_A);
```

5.7 R_MTR_SetPositionStatus

This function sets the input method of the position command value.

Format

void R_MTR_SetPositionStatus(uint8_t u1_pos_status)

Parameters

u1_pos_status
  Input method
    MTR_POS_CONST(0): 0 command
    MTR_POS_STEP(1): Direct input (step input)
    MTR_POS_TRAPEZOID(2): Command value creation

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

The default value for the input method of the sample program is MTR_POS_TRAPEZOID.

Example

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

5.8 R_MTR_SetPosition

This function sets the position command value in degrees.

Format

void R_MTR_SetPosition(int16_t s2_ref_position)

Parameters

s2_ref_position
    Position command value [deg]

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

The position command value is a signed value in units of degrees.
The position after initialization of the position of the motor's magnet is 0 degrees.

Example

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

5.9 R_MTR_GetPosition

This function gets the value in degrees of the current position.

Format

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

Parameters

u1_id
    Specifies the ID of the motor to be controlled.
        MTR_ID_A   /* Motor A*/

Return Values

Current position [deg]

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

The current position value is a signed value in units of degrees.
The position after initialization of the position of the motor’s magnet is 0 degrees.

Example

```
    int16_t s2_pos;

/* Get current position */
s2_pos = R_MTR_GetPosition(MTR_ID_A);
```

5.10 R_MTR_GetPositioningFlag

This function gets the value of the positioning completed flag.

Format

uint8_t R_MTR_GetPositioningFlag (uint8_t u1_id)

Parameters

u1_id
Specifies the ID of the motor to be controlled.
MTR_ID_A /* Motor A*/

Return Values

MTR_FLG_CLR(0): Positioning not completed
MTR_FLG_SET(1): Positioning completed

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

Checking of the completion of positioning is based on judging whether the difference between the target position and the current position is within a certain range.

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

This function sets the velocity command value in rpm.

Format

void R_MTR_SetSpeed(int16_t ref_speed)

Parameters

ref_speed
Velocity command value [rpm]

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

The unit of velocity command values is rpm. A negative value will cause rotation in the opposite direction to forward rotation.
The available range of settings in the sample program is -2000 rpm to 2000 rpm.

Example

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

5.12 R_MTR_GetSpeed

This function gets the current velocity value in rpm.

Format

int16_t R_MTR_GetSpeed (uint8_t u1_id)

Parameters

u1_id
Specifies the ID of the motor to be controlled.
MTR_ID_A /* Motor A*/

Return Values

Current velocity [rpm]

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

This function gets the velocity value at the time the function is executed. The unit is rpm.
A negative value indicates rotation being in the opposite direction to that of forward rotation.

Example

```
    int16_t    s2_speed;

/* Get current speed */
s2_speed = R_MTR_GetSpeed(MTR_ID_A);
```

5.13 R_MTR_SetDir

This function sets the direction of the motor's rotation.

Format

void R_MTR_SetDir (uint8_t dir, uint8_t u1_id)

Parameters

dir
    Rotation direction
        MTR_CW(0): Clockwise
        MTR_CCW(1): Counterclockwise
u1_id
    Specifies the ID of the motor to be controlled.
        MTR_ID_A   /* Motor A*/

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

Processing of the position and velocity are with the set direction of rotation as the positive direction.
The default value in the sample program is clockwise.

Example

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

5.14 R_MTR_GetDir

This function gets the currently set direction of motor rotation.

Format

uint8_t R_MTR_GetDir(uint8_t u1_id)

Parameters

u1_id
    Specifies the ID of the motor to be controlled.
        MTR_ID_A   /* Motor A*/

Return Values

Rotation direction
    MTR_CW(0): Clockwise
    MTR_CCW(1): Counterclockwise

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

When called, this function gets the direction of rotation.

Example

```
    uint8_t    u1_dir;


/* Get direction */
u1_dir = R_MTR_GetDir(MTR_ID_A);
```

5.15 R_MTR_GetStatus

This function gets the state of system operation of the motor control program.

Format

uint8_t R_MTR_GetStatus(uint8_t u1_id)

Parameters

u1_id
Specifies the ID of the motor to be controlled.
MTR_ID_A /* Motor A*/

Return Values

State of system operation

MTR_MODE_INACTIVE	(0x00)
MTR_MODE_ACTIVE	(0x01)
MTR_MODE_ERROR	(0x02)

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

The indicator of the state of system operation reflects motor driving being stopped (INACTIVE), the motor being driven (ACTIVE), or an abnormal state (ERROR).

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

This function sets the buffer of variables for ICS input to the default value.

Format

void R_MTR_InputBuffParamReset(void)

Parameters

None

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

The buffer of variables for ICS input is for use with the motor control parameters, input by the user interface module, and is for use by the interface module.

Variable Name	Type	Variable Symbol	Module/Layer in Use
ICS input variable	mtr_ctrl_input_t type	st_ctrl_input	User interface / Application
Buffer of variables for ICS input	mtr_ctrl_input_t type	st_ctrl_input_buff	Interface / Middle

Example

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

5.17 R_MTR_CtrlInput

This function copies an ICS input variable to the buffer of variables for ICS input.

Format

void R_MTR_CtrlInput(mtr_ctrl_input_t *st_ctrl_input)

Parameters

Pointer to an mtr_ctrl_input_t type structure

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

This function copies the mtr_ctrl_input_t type variable to which the pointer points to the buffer of variables for ICS input, st_ctrl_input_buff. It also sets the write enable flag (u1_trig_enable_write) among the motor control parameters.

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

This function sets the contents of the buffer of variables for ICS input as the motor control parameters.

Format

void R_MTR_SetVariables(void)

Parameters

None

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

This function sets the values of the various parameters set in the buffer of variables for ICS input, st_ctrl_input_buff, as the corresponding motor control parameters.

Example

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

5.19 R_MTR_AutoSetVariables

This function sets the position and velocity command values in the buffer of variables for ICS input as the given motor control parameters.

Format

void R_MTR_AutoSetVariables(void)

Parameters

None

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

This function is executed in response to the 500-us cycle interrupt.
This setting is only valid when the setting of the automatic update mode parameter is "enabled".

Example

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

5.20 R_MTR_CtrlGainCalc

This function calculates the gain for the motor control parameters.

Format

void R_MTR_CtrlGainCalc(void)

Parameters

None

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

This function calculates the gain for PI control, IPD control, and velocity observer control.

Example

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

5.21 R_MTR_UpdatePolling

This function polls the write enable flag among the motor control parameters.

Format

void R_MTR_UpdatePolling(void)

Parameters

None

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_acces.h.

Description

This function polls the write enable flag (u1_trig_enable_write) among the motor control parameters. If the flag is set, the values in the buffer of variables for ICS input are set as the motor control parameters (by the R_MTR_SetVariables function) and calculation of the motor control parameter gain (by the R_MTR_CtrlGainCalc function) proceeds.
After the execution of these functions is completed, the R_MTR_UpdatePolling function clears the write enable flag (u1_trig_enable_write) among the motor control parameters.

Example

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

5.22 R_MTR_GetErrorStatus

This function gets the error state of the encoder position and velocity control.

Format

uint16_t R_MTR_GetErrorStatus(uint8_t u1_id)

Parameters

u1_id
Specifies the ID of the motor to be controlled.
MTR_ID_A /* Motor A*/

Return Values

Error status

Properties

The prototypes are declared in r_mtr_driver_ecat_acces.h.

Description

The error states are described in the following list.

Macro Name	Value	Type of Error
MTR_ERROR_NONE	0x0000	No error
MTR_ERROR_OVER_CURRENT_HW	0x0001	Overcurrent error (H/W detection)
MTR_ERROR_OVER_VOLTAGE	0x0002	Inverter main line overvoltage error
MTR_ERROR_OVER_SPEED	0x0004	Rotation velocity error
MTR_ERROR_HALL_TIMEOUT	0x0008	Not in use
MTR_ERROR_BEMF_TIMEOUT	0x0010	Not in use
MTR_ERROR_HALL_PATTERN	0x0020	Hole detection angle error
MTR_ERROR_BEMF_PATTERN	0x0040	Not in use
MTR_ERROR_UNDER_VOLTAGE	0x0080	Inverter main line undervoltage error
MTR_ERROR_OVER_CURRENT_SW	0x0100	Overcurrent error (S/W detection)
MTR_ERROR_UNKNOWN	0xffff	Not in use

Example

```
uint16_t    u2_error_status;

/* Get FOC error status */
u2_error_status = R_MTR_GetErrorStatus(MTR_ID_A);
```

5.23 R_MTR_GetPositionPFStatus

This function gets the profile state of encoder position control.

Format

uint8_t R_MTR_GetPositionPFStatus(uint8_t u1_id)

Parameters

u1_id
Specifies the ID of the motor to be controlled.
MTR_ID_A /* Motor A*/

Return Values

Profile status
MTR_POS_STEADY_STATE (0): Stable state (the current position is not changing)
MTR_POS_TRANSITION_STATE (1): Transitional state (the current position is changing)

Properties

The prototypes are declared in r_mtr_driver_ecat_acces.h.

Description

This function can be used to check whether the motor is or is not running under positional control.

Example

```
uint8_t    u1_pos_state;

/* Get position profile status */
u1_pos_state = R_MTR_GetPositionPFStatus(MTR_ID_A);
```

5.24 R_MTR_SetPositionUnits

This function sets the position command value (number counted).

Format

void R_MTR_SetPositionUnits (int32_t s4_position_units)

Parameters

Position command value [count]

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_ecat_acces.h.

Description

The position command value is a signed value (number counted).
The position after initialization of the position of the motor's magnet is 0 degrees.

Example

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

5.25 R_MTR_SetActualPositionUnits

This function sets the current position (number counted).

Format

void R_MTR_SetActualPositionUnits (int32_t s4_position_units)

Parameters

Position command value [count]

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_ecat_acces.h.

Description

The position command value is a signed value (number counted).
This function is only for setting the value for the current position but does not control the motor.
It can be used for setting an offset in the initial position in homing mode.

Example

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

5.26 R_MTR_GetPositionUnits

This function gets the value (number counted) corresponding to the current position.

Format

int32_t R_MTR_GetPositionUnits(uint8_t u1_id)

Parameters

u1_id
Specifies the ID of the motor to be controlled.
MTR_ID_A /* Motor A*/

Return Values

Current position [count]

Properties

The prototypes are declared in r_mtr_driver_ecat_acces.h.

Description

The current position value is a signed value (number counted).

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

This function gets the current velocity value (number counted per second).

Format

int32_t R_MTR_GetSpeedUnits(uint8_t u1_id)

Parameters

u1_id
Specifies the ID of the motor to be controlled.
MTR_ID_A /* Motor A*/

Return Values

Get the Current velocity [count/s]

Properties

The prototypes are declared in r_mtr_driver_ecat_acces.h.

Description

The current velocity value is a signed value (number counted per second).
The value is converted to the 16.16-bit fixed point format.

Example

```
int32_t    s4_speed_units;

/* Get speed units */
s4_speed_units = R_MTR_GetSpeedUnits(MTR_ID_A);
```

5.28 R_MTR_SetSpeedUnits

This function sets the velocity command value.

Format

void R_MTR_SetSpeedUnits(int32_t s4_speed_units)

Parameters

Velocity command value [count/s]

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_ecat_acces.h.

Description

The velocity command value is a signed value (number counted per second).
The value is converted to the 16.16-bit fixed point format.

Example

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

5.29 R_MTR_SetAccelerationUnits

This function sets the acceleration command value in count/s^2.

Format

void R_MTR_SetAccelerationUnits(uint32_t u4_acceleration_units)

Parameters

Acceleration command value [count/s^2]

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_ecat_acces.h.

Description

The acceleration command value is a signed value in count/s^2.
The value is converted to the 16.16-bit fixed point format.

Example

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

5.30 R_MTR_SetDecelerationUnits

This function sets the deceleration command value in count/s^2.

Format

void R_MTR_SetDeccelerationUnits(uint32_t u4_deceleration_units)

Parameters

Deceleration command value [count/s^2]

Return Values

None

Properties

The prototypes are declared in r_mtr_driver_ecat_acces.h.

Description

The deceleration command value is a signed value in count/s^2.
The value is converted to the 16.16-bit fixed point format.
Note: The sample program does not use the deceleration command value in motor control.

Example

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

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 this manual runs in the environment below.

Table 6.6-1 Operating Environment

Item	Description
Boards to be used	RX72M CPU board TS-TCS02796 from Tessera Technology RX23T inverter board : RTK0EM0006S01212BJ) Motor encoder I/F board
CPU	RX CPU (RXv3)
Operating voltage	24 V
Communication protocol	EtherCAT
Integrated development environment	CCRX compiler (V3.01.00 or later) + e2studio (V7.5.0 or later)
Emulator	Renesas E2 Lite
SSC tool	Provided by the EtherCAT Technology Group (ETG) Slave Stack Code (SSC) tool Version 5.12 or later
Software PLC	Beckhoff Automation TwinCAT® 3 (download this from the Beckhoff web site) CODESYS

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

6.2 Operating Environment Settings and Connection

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

(1) Connect the three-phase power lines of the motor to the U, V, and W phase outputs of the inverter board as shown below.

1-A) Connect the brown line of the motor to the U phase inverter.

1-B) Connect the blue line of the motor to the V phase inverter.

1-C) Connect the black line of the motor to the W phase inverter.

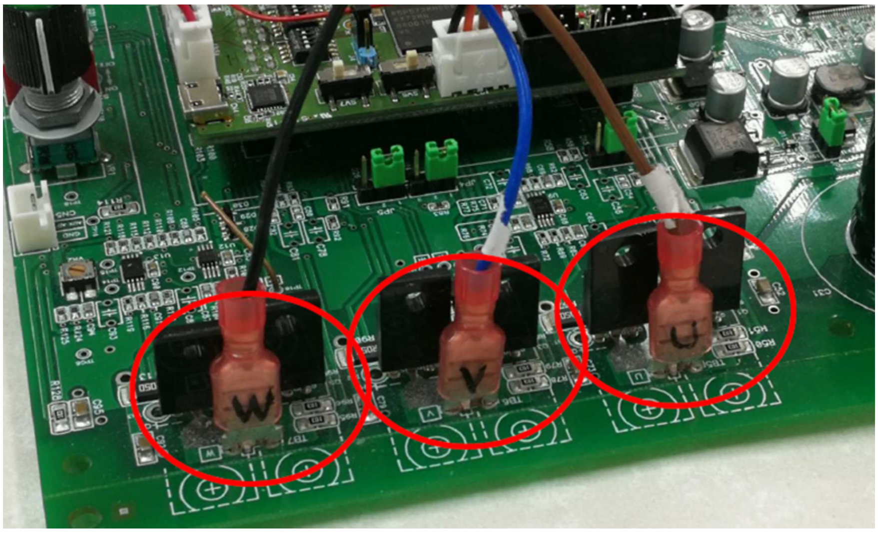


Figure 6-1 Connection of the Three-Phase Power Lines

(2) Connect the motor encoder to the encoder interface (I/F) board as shown below.

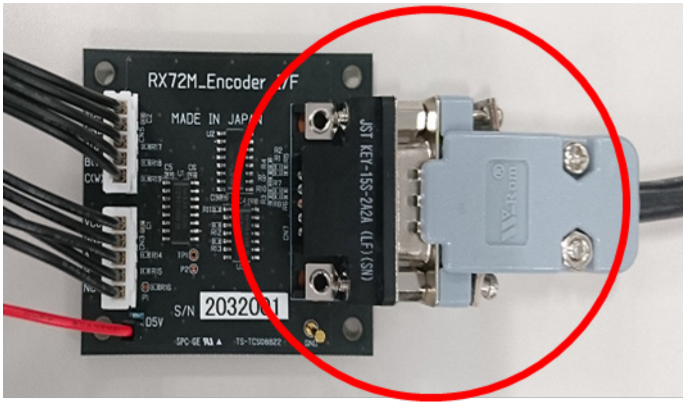


Figure 6-2 Connection of the Encoder I/F Board

(3) Connect the 5-V power supply points of the encoder I/F board and CPU board as shown below.

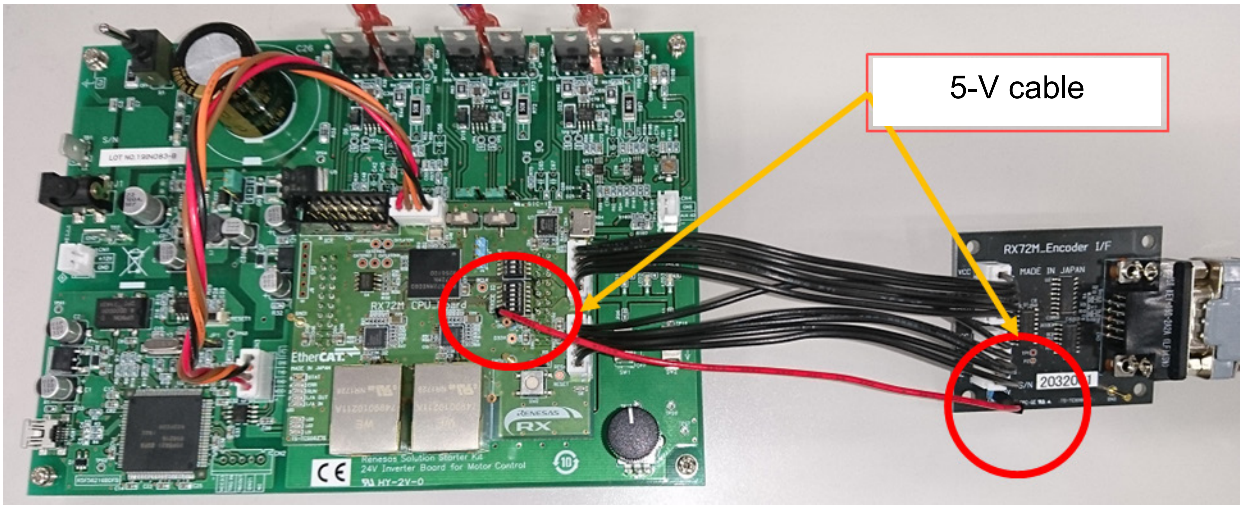


Figure 6-3 Connection of the 5-V Power Supply

(4) Connect the inverter board and CPU board as follows.

-- Attach the CPU board to the inverter board.

-- Connect the ICS cable between the CPU board and the inverter board.

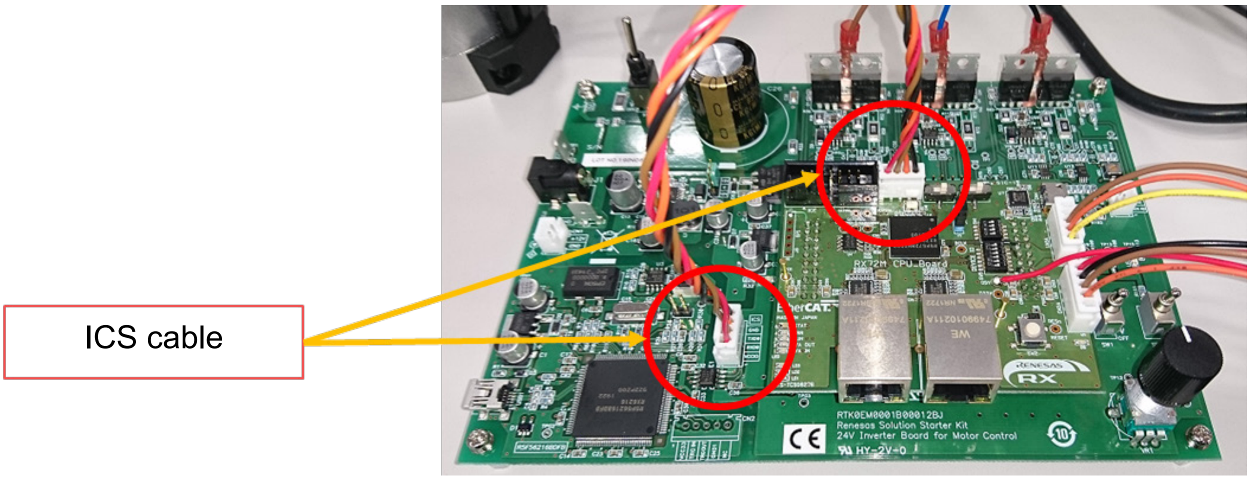


Figure 6-4 Connection of the CPU Board and Inverter Board

(5) Connect the power supply to the inverter board as follows.

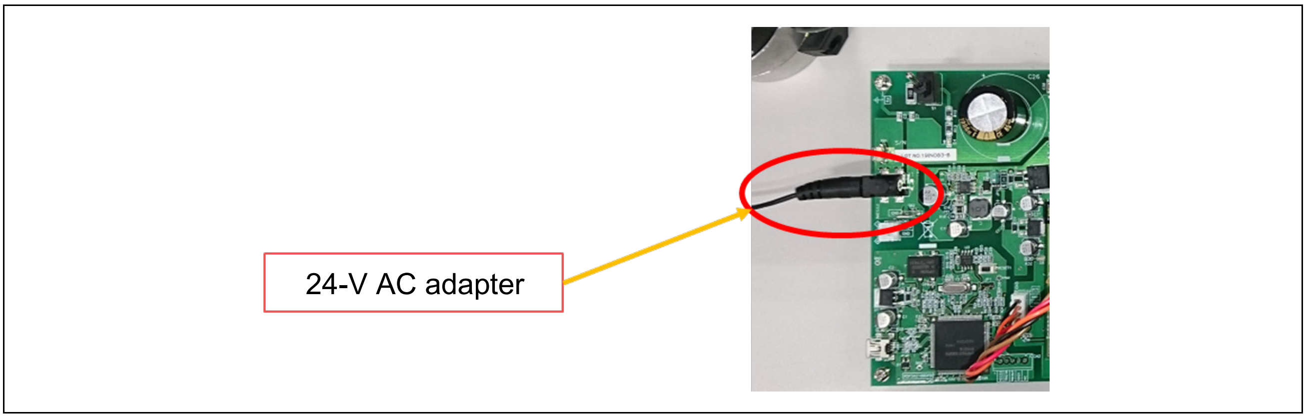


Figure 6-5 Connection of the Power Supply for the Inverter

(6) The overall configuration of the connected parts is shown below.

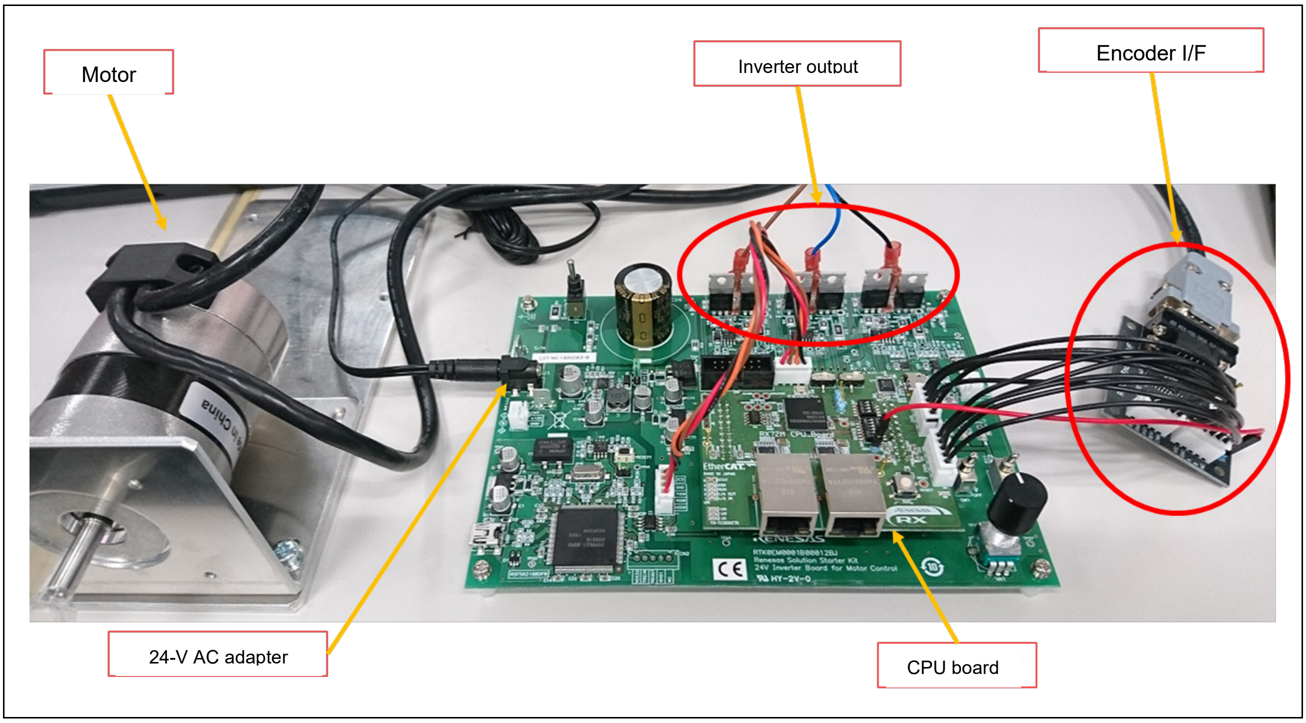


Figure 6-6 Overall Configuration

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

-- The power is supplied through the main switch.

-- Connect the ICS I/F when using the RMW (Renesas Motor Workbench).

-- SW1 and SW2 are not used in control as covered by this manual.

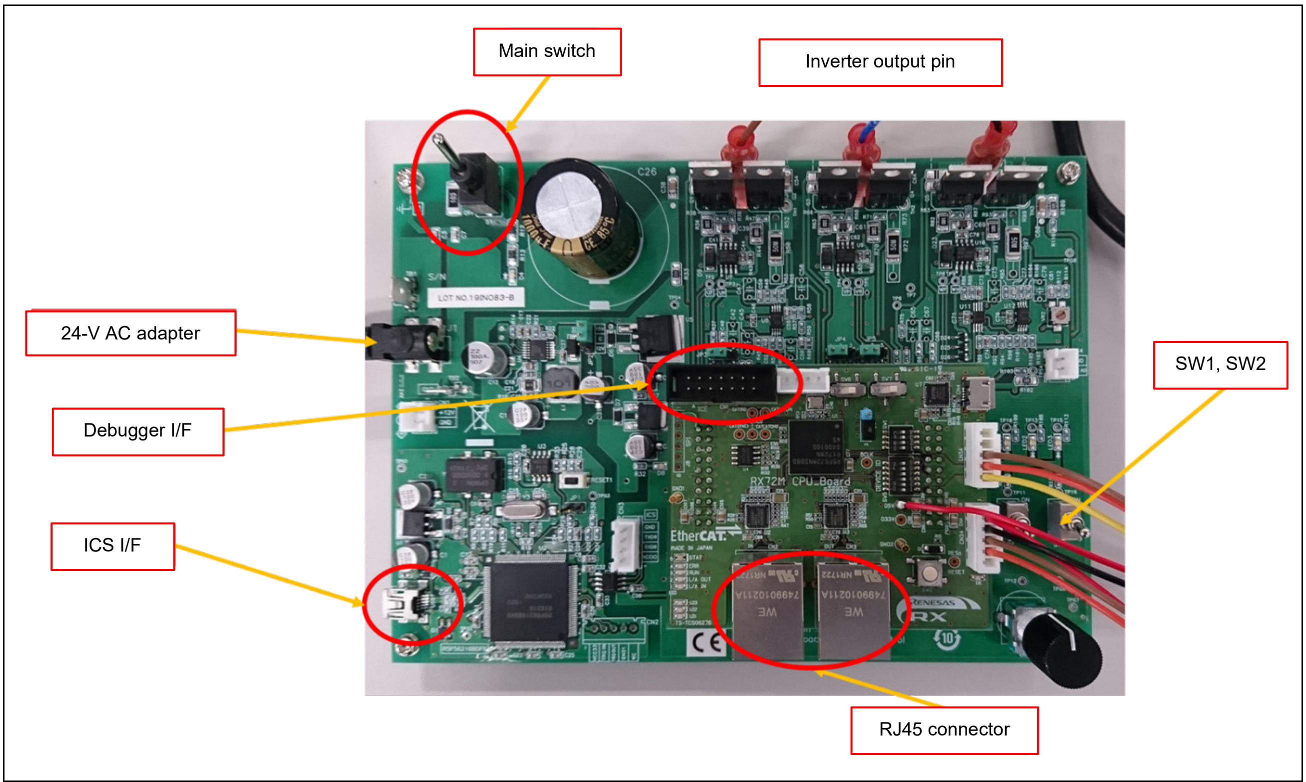


Figure 6-7 Details of the Inverter Board

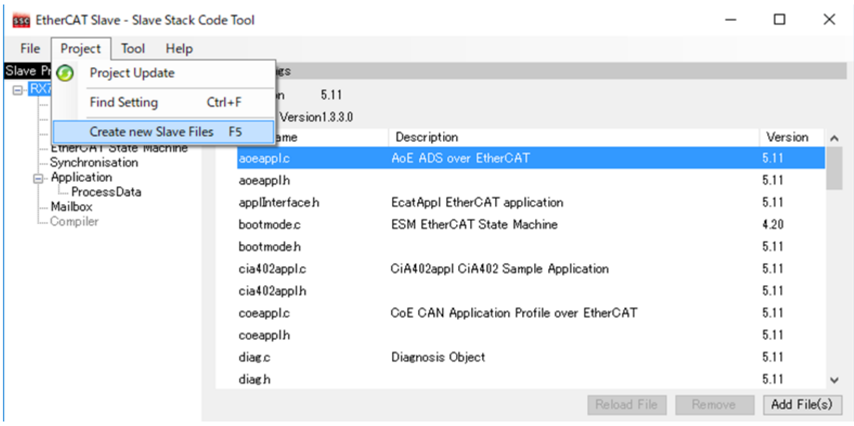
6.3 Building the Sample Program

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.

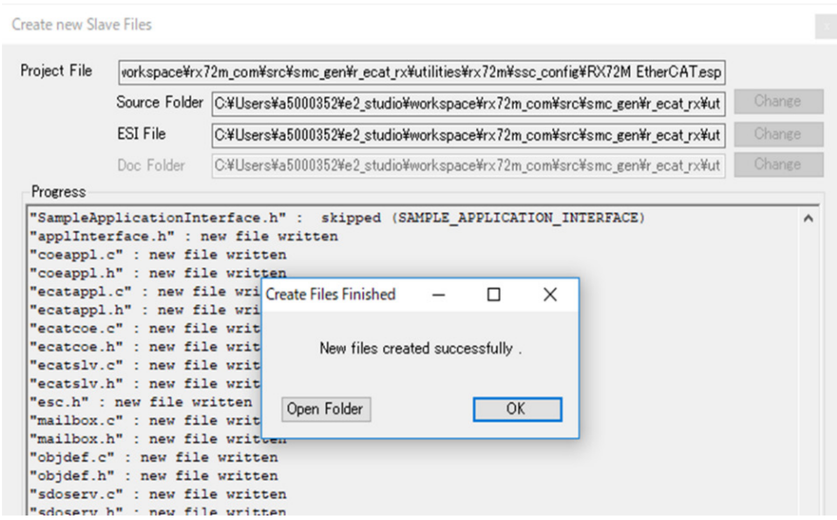
(1) Double-click on the SSC project file of the sample program to activate the SSC tool.

ecat_cia402_motor_rsskrx72m\src\smc_gen\r_ecat_rx\utilities\rx72m\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.

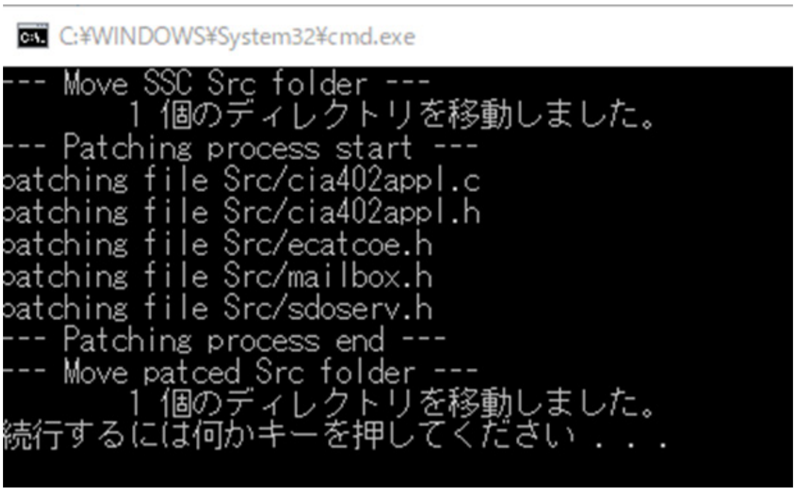
http://gnuwin32.sourceforge.net/packages/patch.htm

(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.

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

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

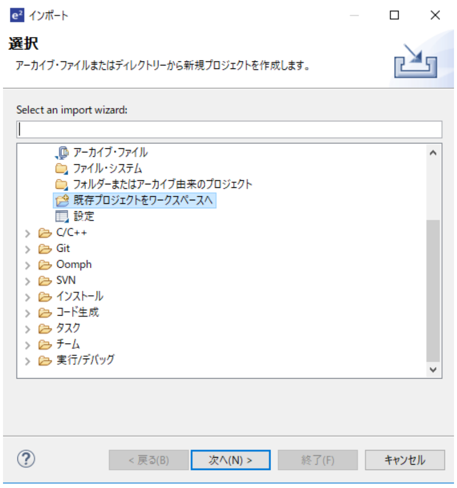
ecat_cia402_motor_rsskrx72m\src\application\ecat



6.4 Importing the Sample Project into the e2 studio

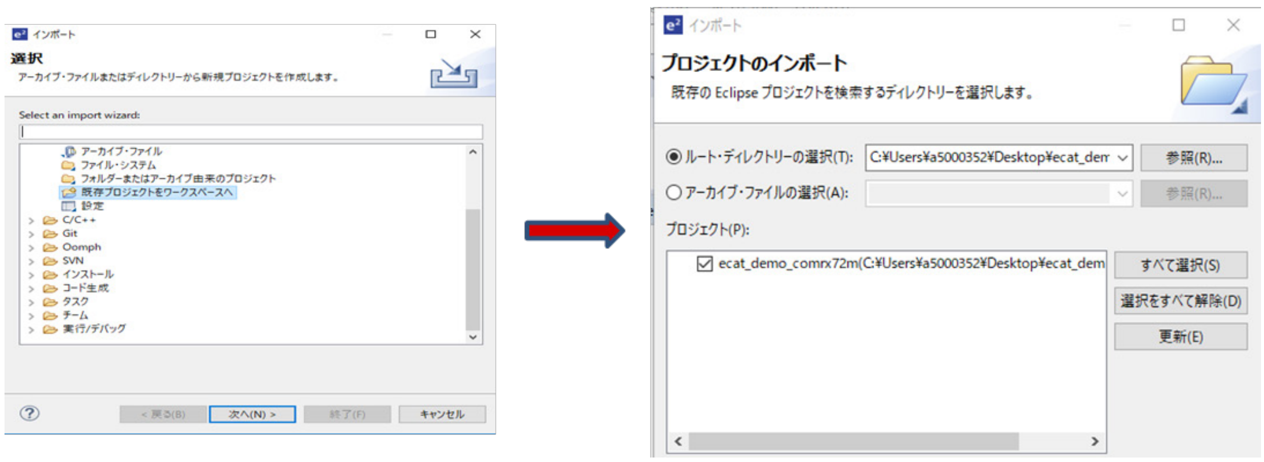
(1) Click on [File] → [Import].

(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].

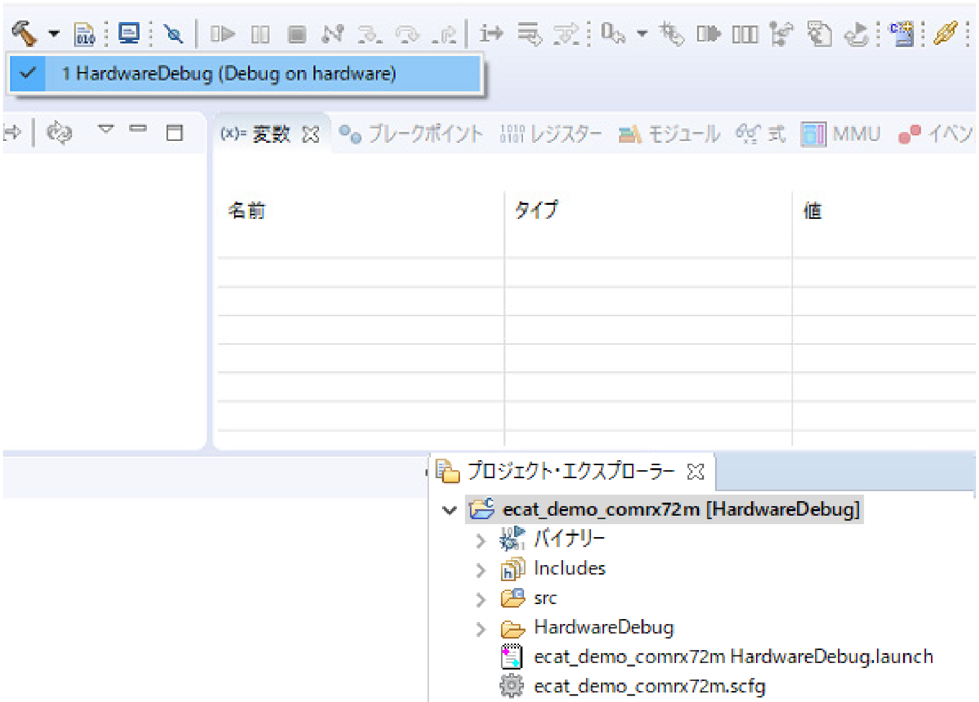
(4) Select "ecat_cia402_motor_rsskrx72m", which is the sample project for the communications board, and click on [Open]. Check "ecat_cia402_motor_rsskrx72m" indicated under [Project] and click on [Next] to import the project.



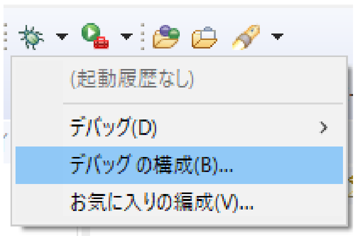
6.5 Programming and Debugging

(1) Left-click on the "ecat_cia402_motor_rsskrx72m" 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.

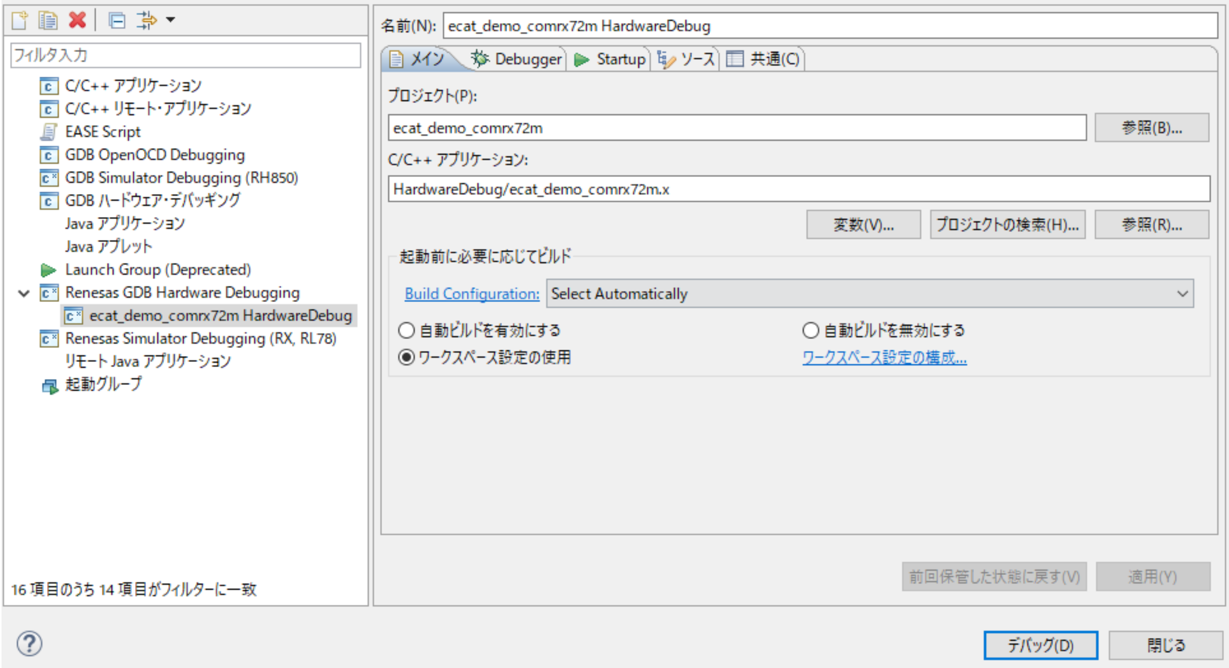
The project is built by e2studio. Check that the console does not indicate any errors in building.



(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 "ecat_cia402_motor_rsskrx72m 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].

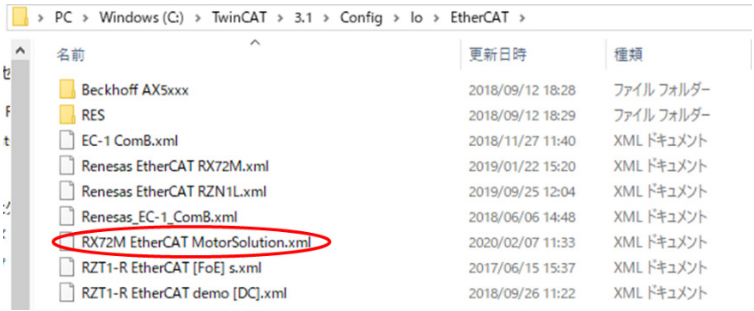
(6) If a dialog box recommending a change of the perspective is displayed in the confirmation dialog box for switching perspectives, select the "Always use this setting" checkbox and click [Yes].

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

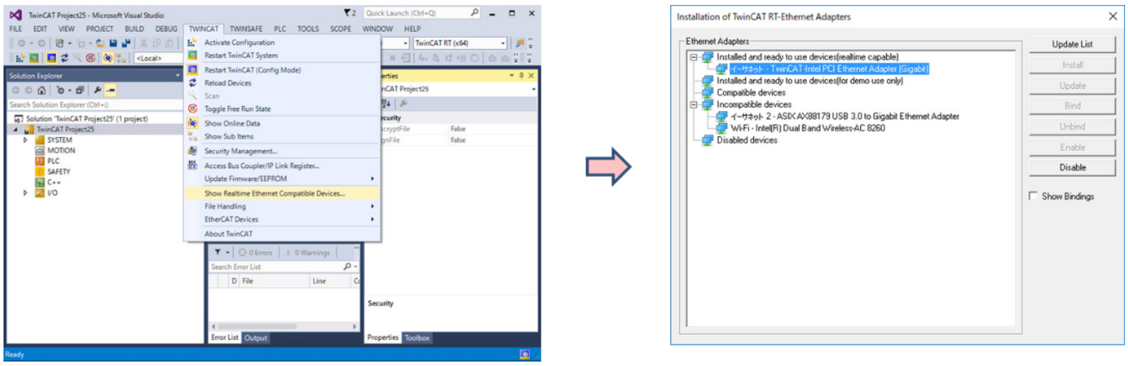
After downloading the code, click on the [Resume] button to run the code up to the first line of the function main. Click on the [Resume] button again to run the rest of the code on the target.

6.6 Connection with TwinCAT (Writing the ESI File)

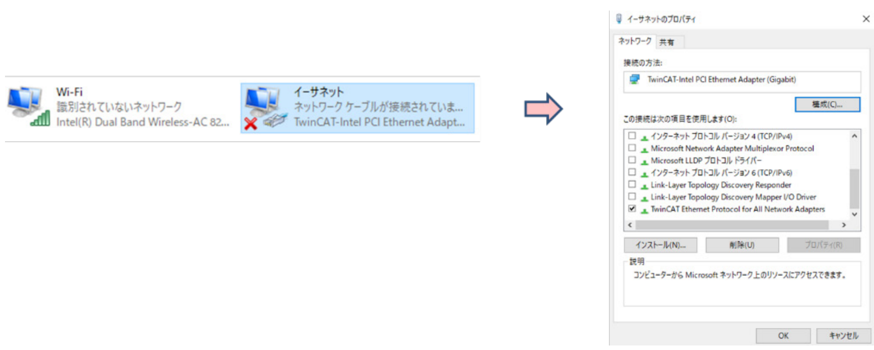
(1) Before starting TwinCAT, copy the ESI file included in the release folder to "/TwinCAT / 3.x / Config / IO / EtherCAT".
"ecat_cia402_motor_rsskrx72m\src\smc_gen\r_ecat_rx\utilities\rx72m\esi\ RX72M EtherCAT MotorSolution.xml"



(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.

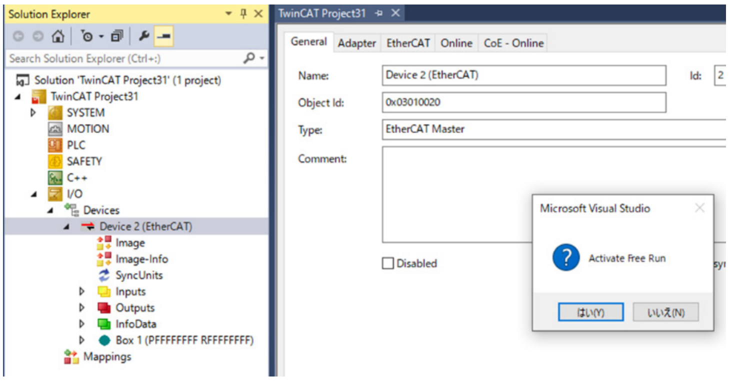


(4) Connect the LAN cable to the evaluation board. As the In/Out direction of EtherCAT has been decided, connect it to CN2 IN.

(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.

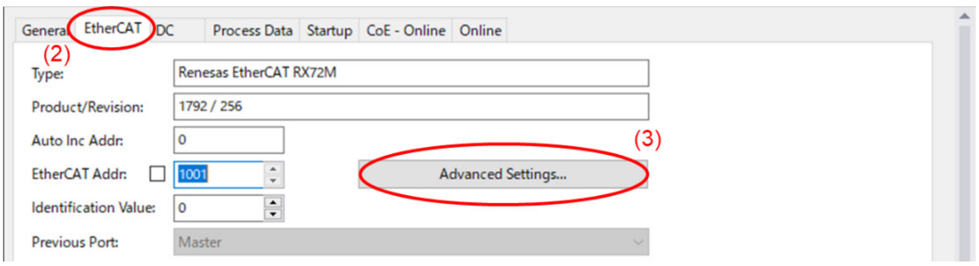
(6) Select Solution Explorer → I / O → Device → [Scan].

(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.



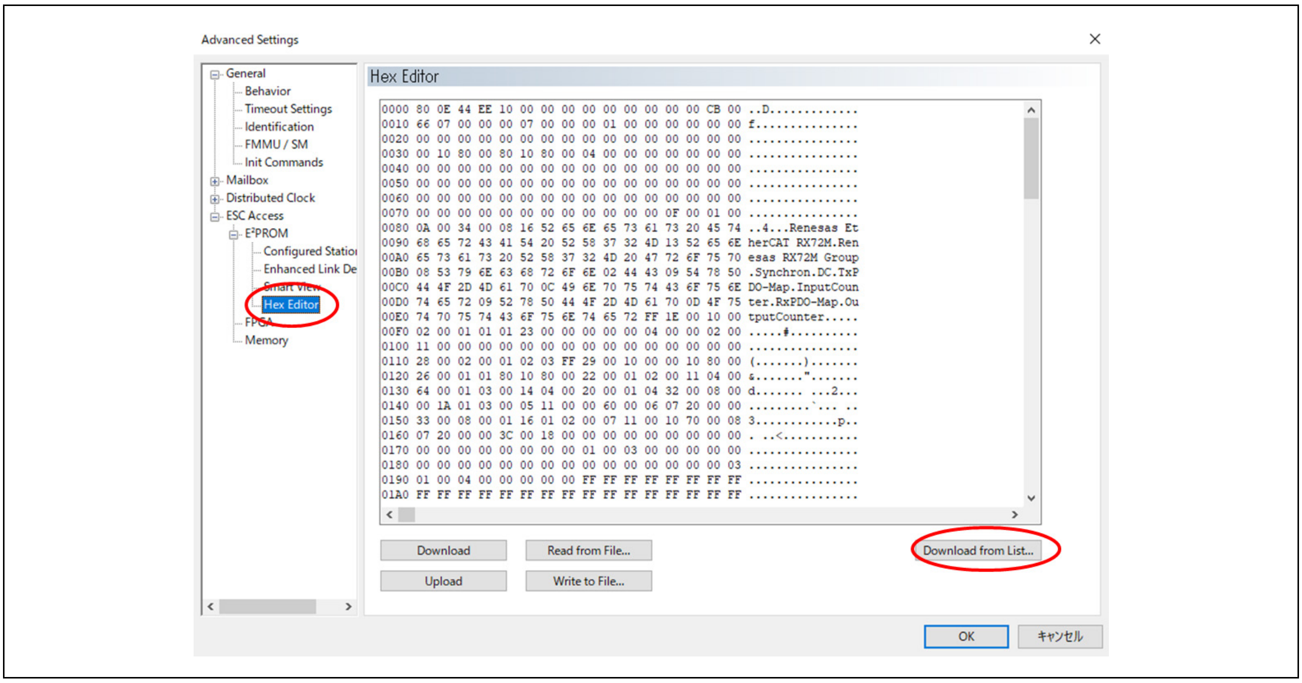
(8) If the data of another application has been written to the EEPROM, replace it. The procedure for replacing the data in the EEPROM is as follows.

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

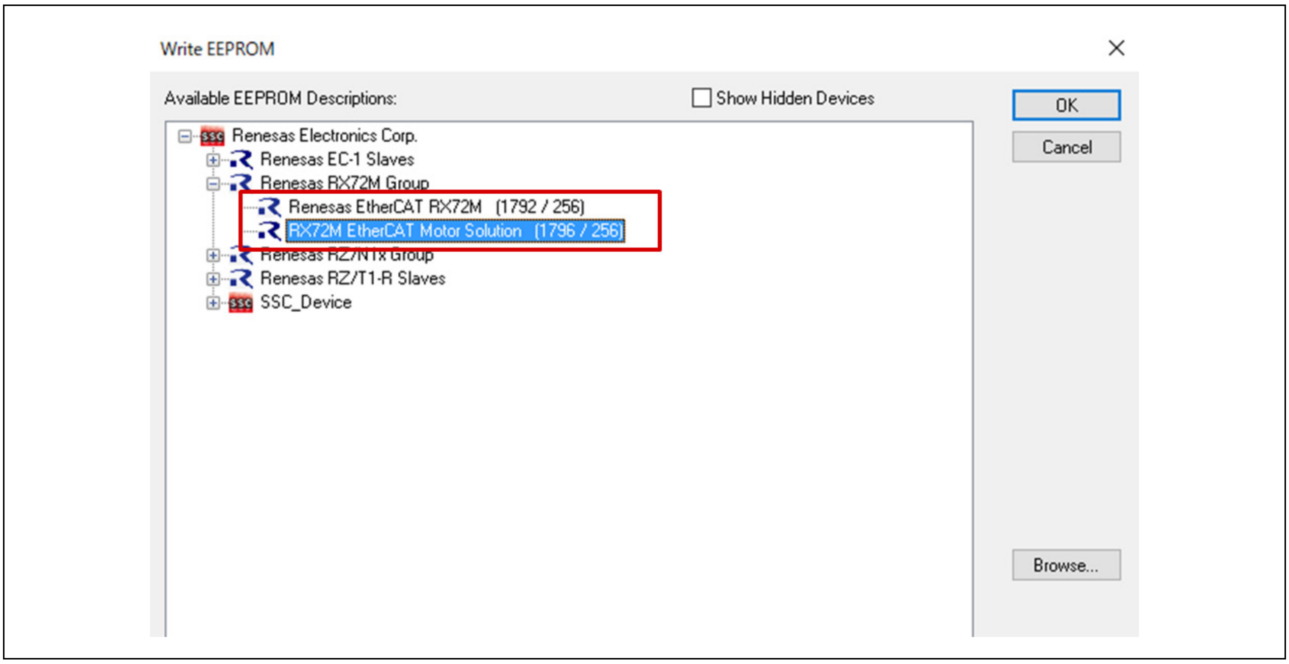


(9) Select [ESC Access] → [EEPROM] → [Hex Editor].

Select [Download from List].

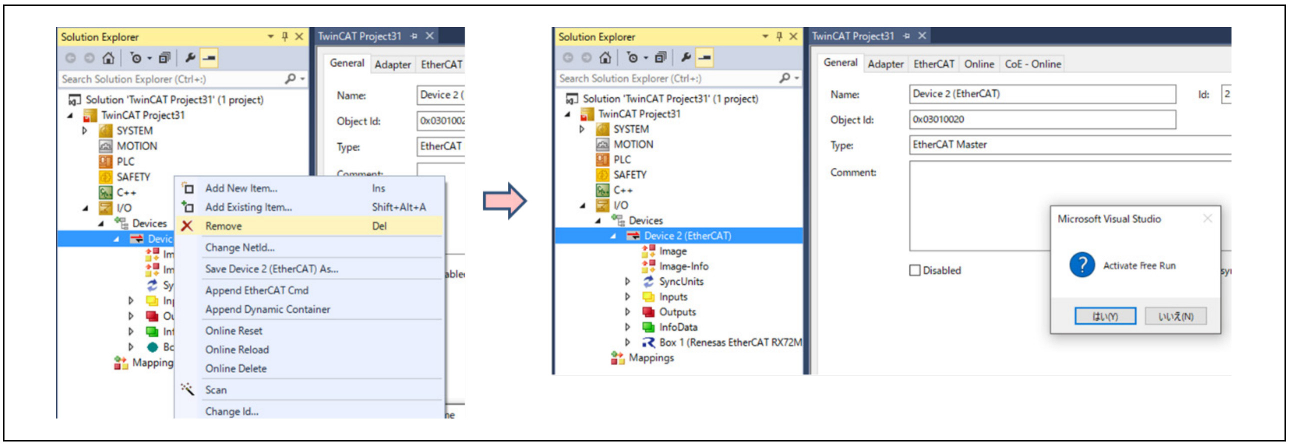


(10) A list of ESI files registered with TwinCAT3 will appear. Select the relevant file. For the motor board, select [RX72M EtherCAT MotorSolution.xml]. For the I/O board, select [Renesas EtherCAT RX72M.xml].



(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".



6.7 Checking the Connection with CODESYS

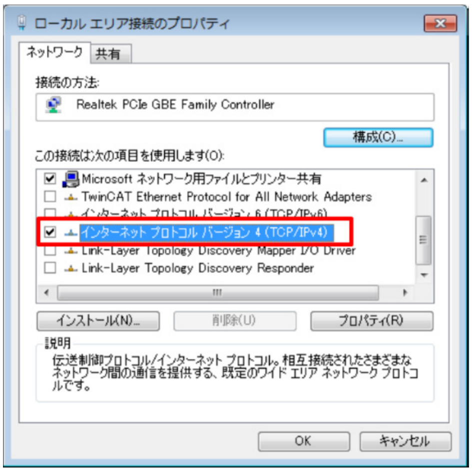
This section describes the procedure for connecting and operating an evaluation environment in which the sample program is installed along with the CODESYS software PLC.

6.7.1 Device Network Settings

(1) Set the IP address of the host before setting the device. Open [Network Settings].

(2) Double-click (or right-click) [Local Area Connection] and select [Properties].

(3) Select TCP/IPv4 and click on the [Properties] button. Set the IP address and subnet mask.

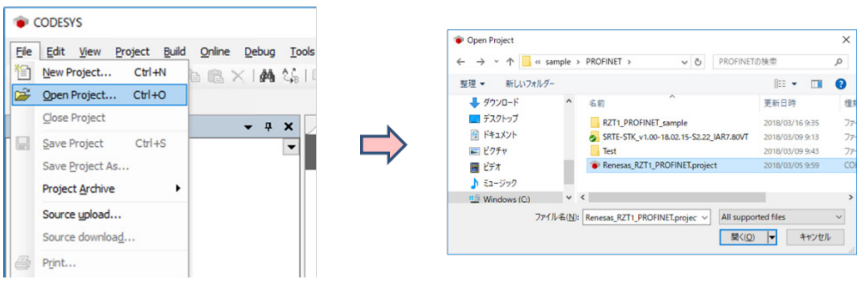


6.7.2 Starting CODESYS

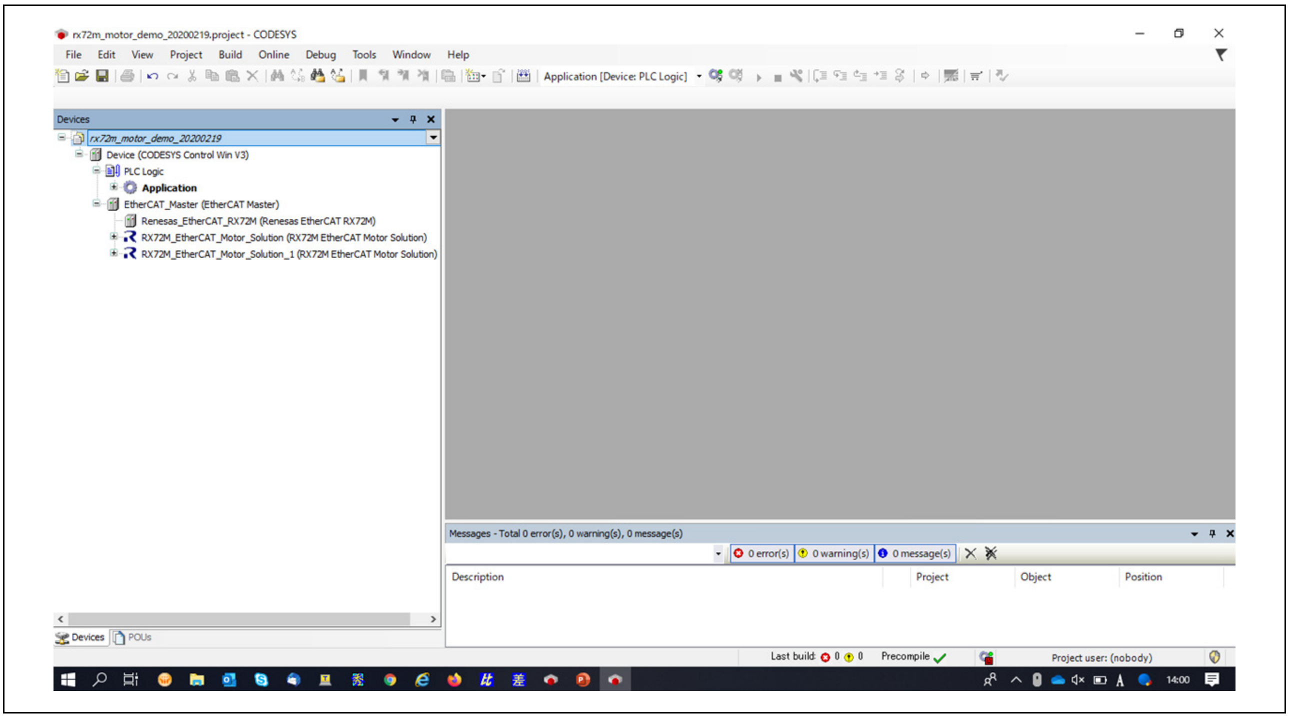
(1) Start this from the [Start] menu of Windows, selecting [All Programs] > [CODESYS] > [CODESYS Gateway V3] or from the CODESYS icon which will have been created on the desktop after installation.

(2) Click on [File] → [Open Project…] and select the "rx72m_motor_demo.project" file to open the project.

Note: Refer to *R-IN, RZ/T1, EC-1, TPS-1 Group Software PLC Guide Project Configuration/UI Creation* for the procedure for configuring a new project for CODESYS and the procedure for creating and confirming UIs.



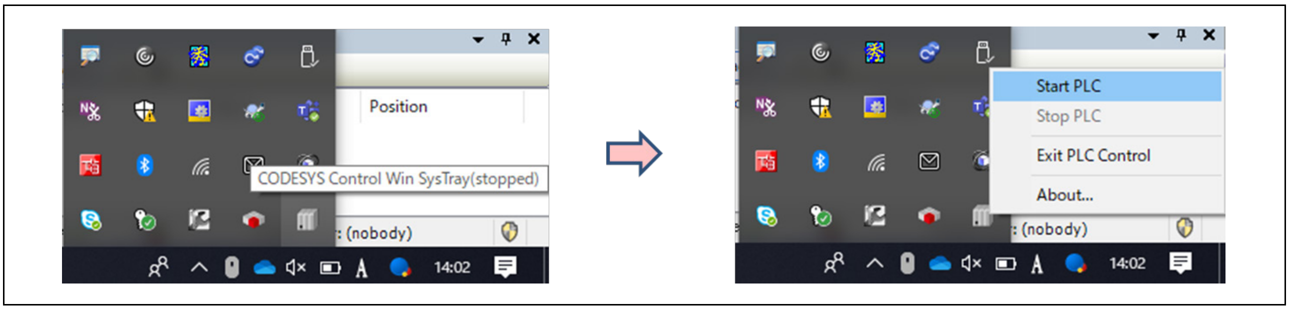
(3) After you start the project, the device tree is displayed.



6.7.3 Starting the PLC

Check the state of the software PLC operation from the system tray. If it is stopped, right-click on the PLC window and select "Start PLC" to start it. The software PLC usually starts automatically as a service when Windows is activated.

The icon in the system tray at the bottom right of the desktop shows the operating status.



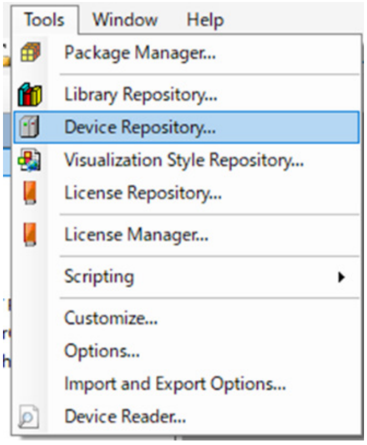
Note: When the icons is not in the system tray

Start the gateway server by selecting [All Programs] > [CODESYS] > [CODESYS Gateway V3] > [CODESYS Gateway V3] from the Start menu. If the icon is not displayed in the system tray after starting the gateway server, try restarting the device.

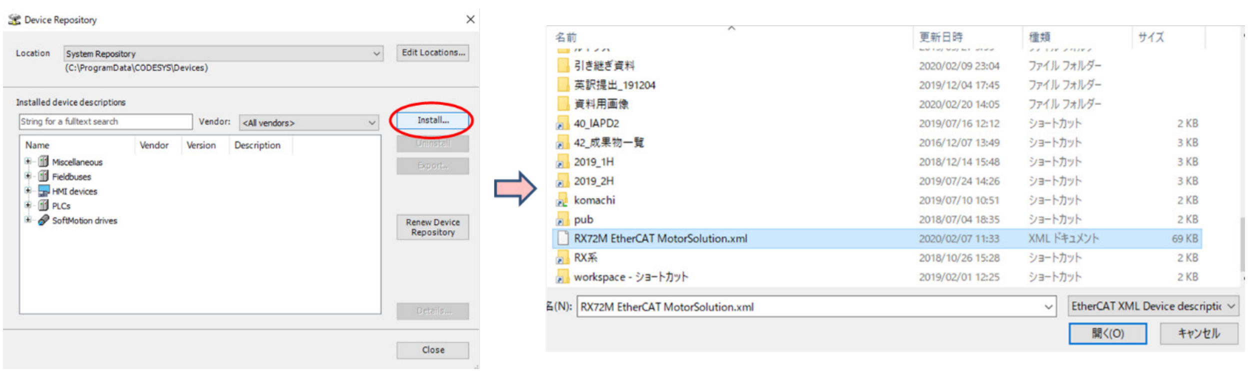
6.7.4 Updating the Slave Device

This section describes the operations to be performed to start the "rx72m_motor_demo.project" for the first time.

(1) Using the EtherCAT slave device requires installing the ESI file that contains the device information. Use "Renesas EtherCAT RX72M.xml" and "RX72M EtherCAT MotorSolution.xml" as the ESI files. From the [Tools] menu of CODESYS, select [Device Repository].

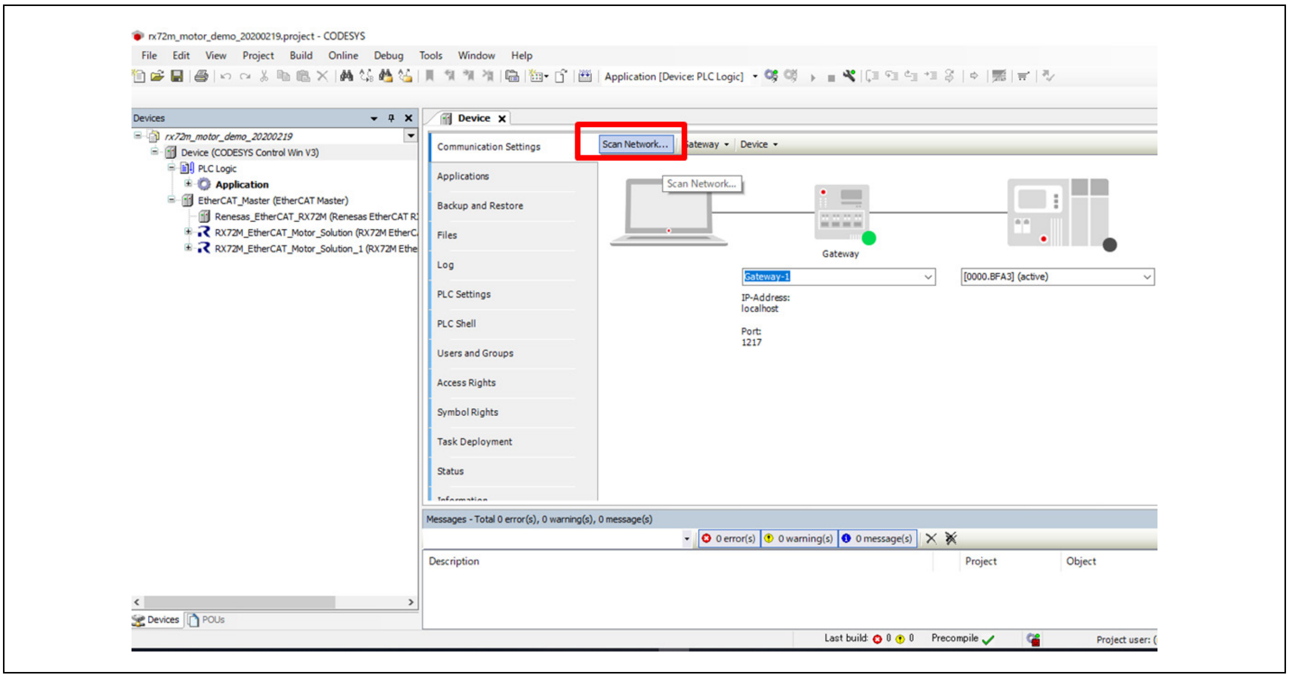


(2) In the Device Repository dialog box, click on [Install] to display the file dialog box. Specify the ESI file "RX72M EtherCAT MotorSolution.xml".

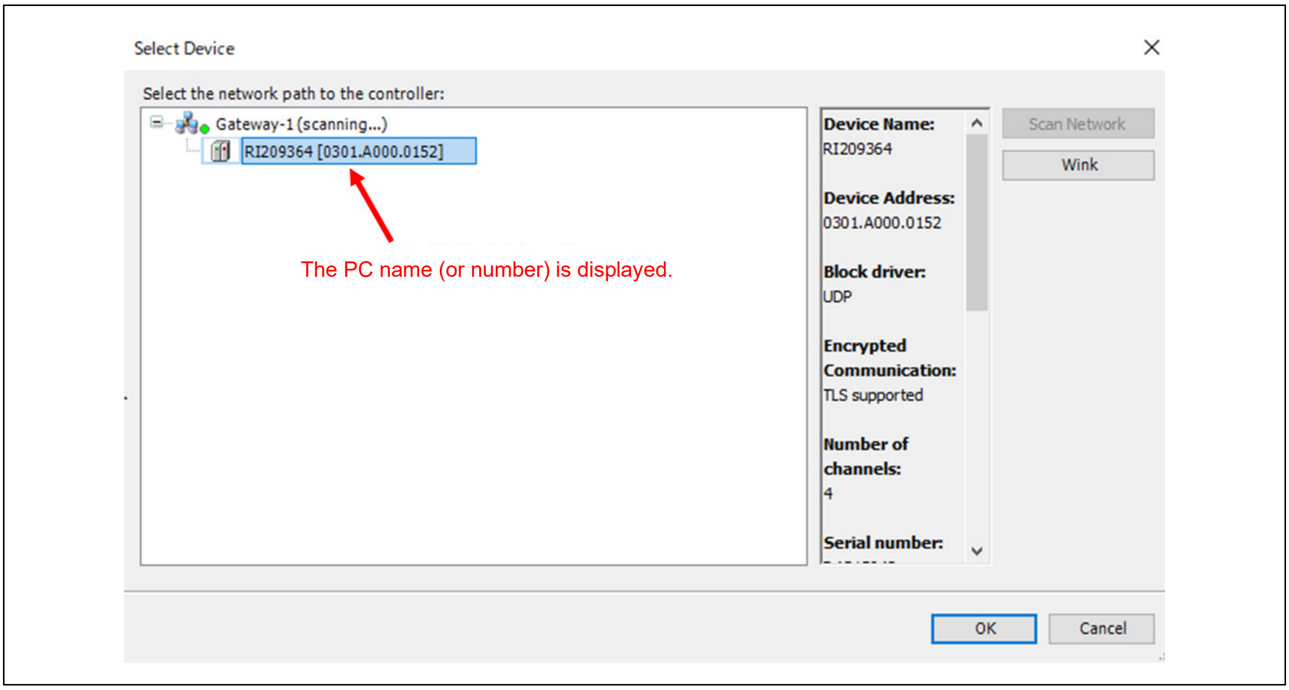


6.7.5 Setting up the Connection with PLC

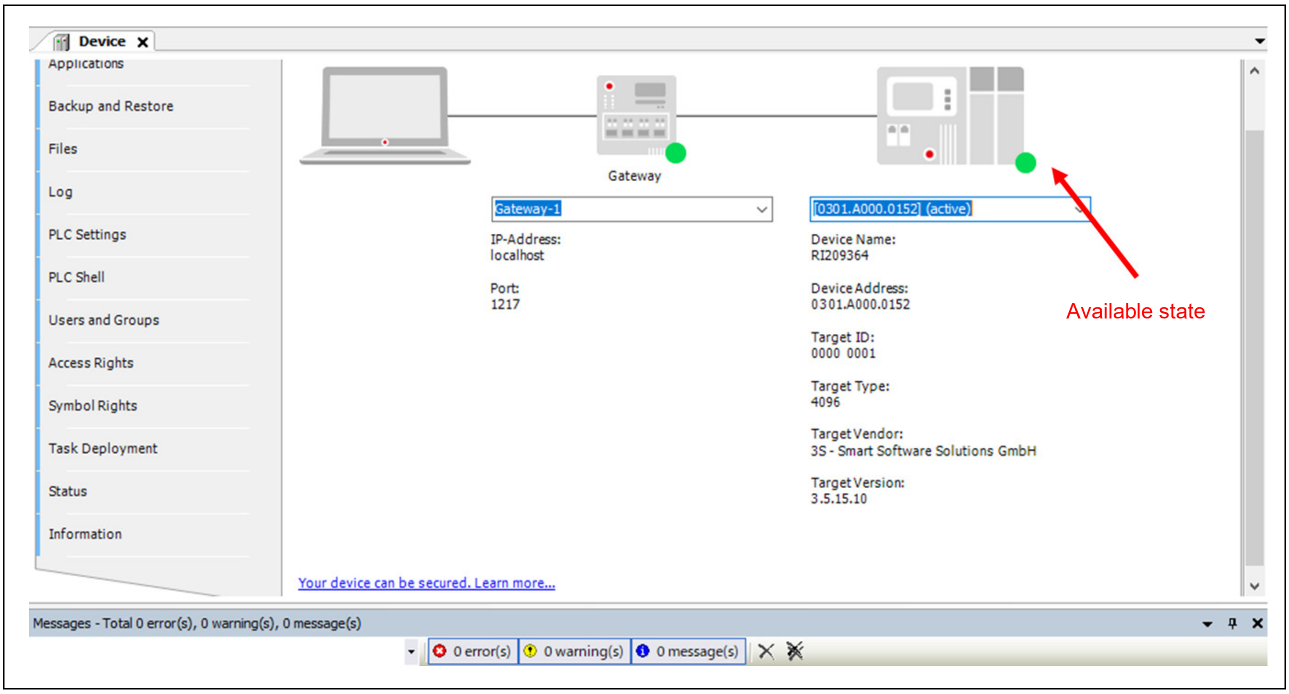
(1) Double-click on [Device (CODESYS Control Win V3)] in the tree of the [Devices] window to open the [Communication Settings] screen. You can set up communications to connect the development environment to the software PLC service on this screen.
Click on the [Scan network...] button on the [Communication Settings] tabbed page.



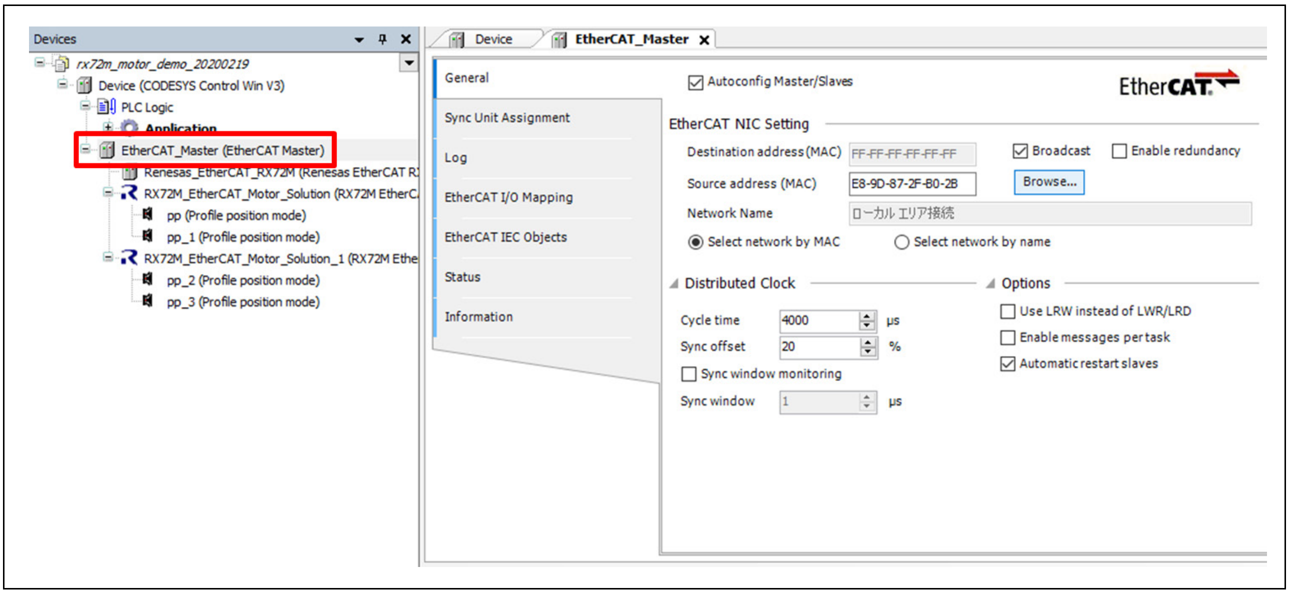
(2) The [Select Device] window will appear and an automatic search for the available devices on the local network will commence. The procedure is successful when the software PLC service is detected.
Double-click on the displayed PC name.



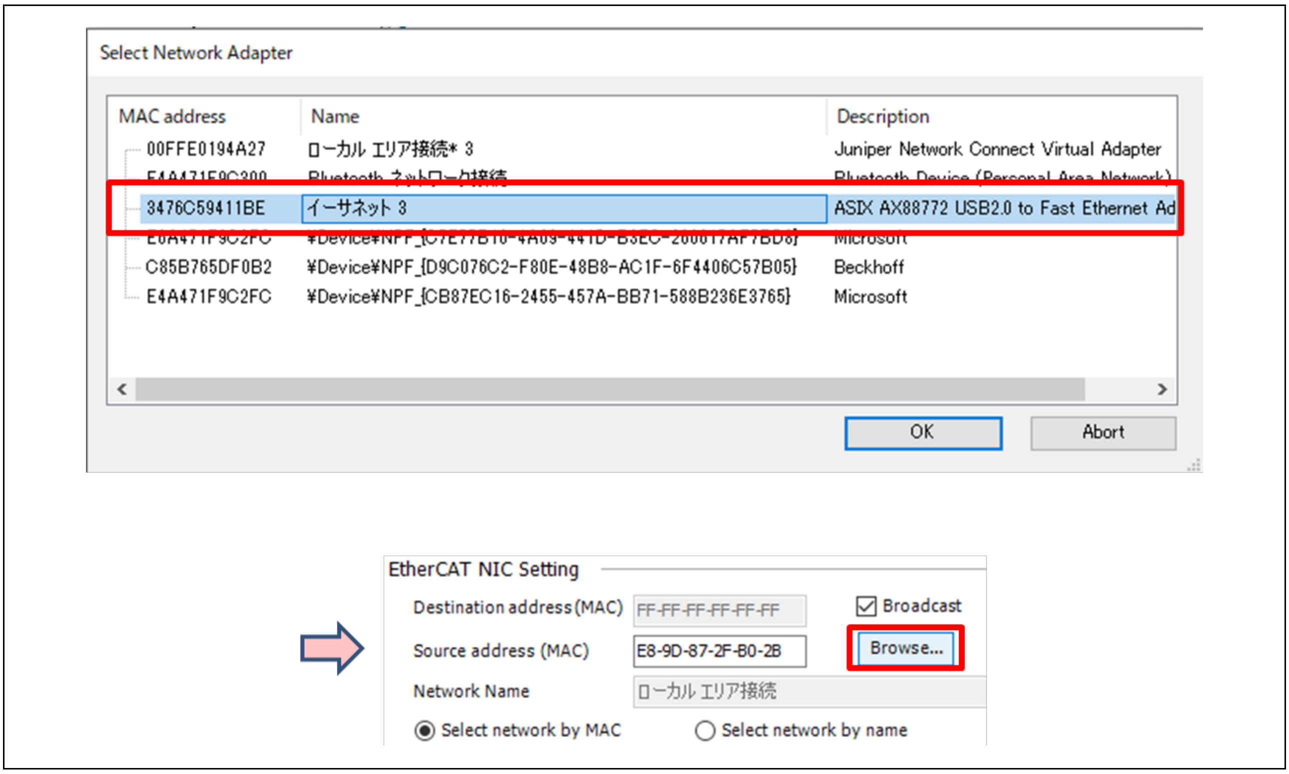
(3) If the scan was successful, the PC will be registered with GateWay.



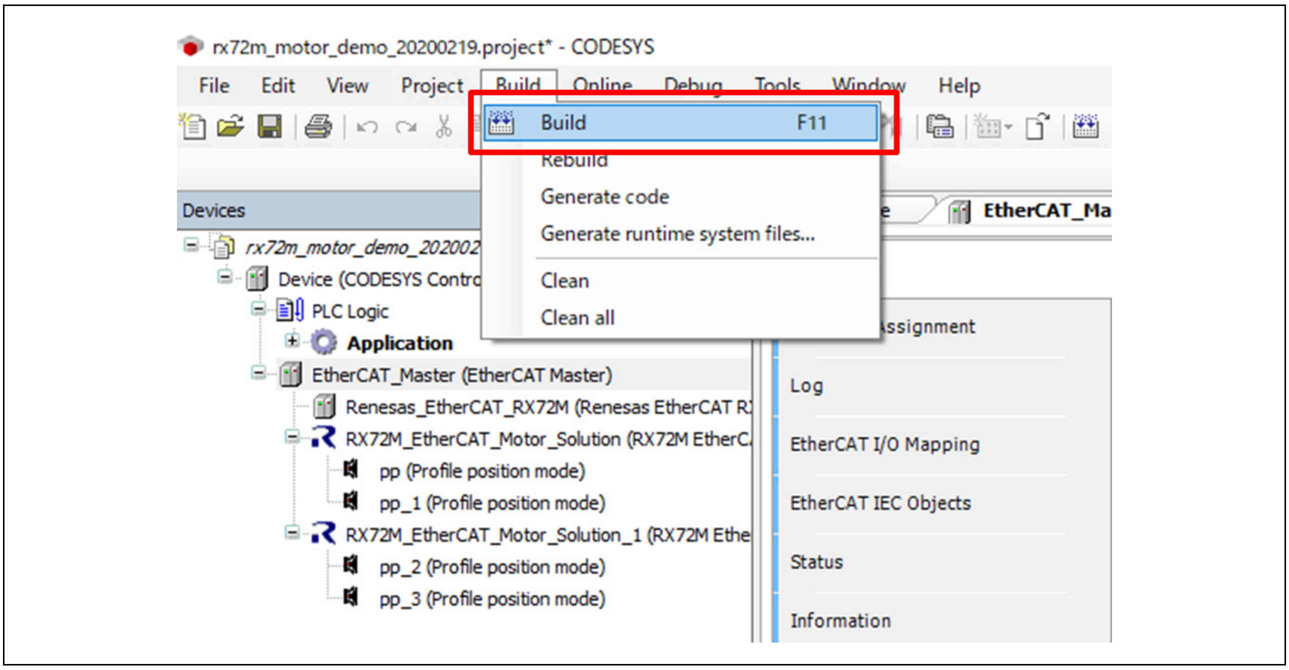
(4) Set the network to be used. Double-click on [EtherCAT_Master] to open the [General] screen for the corresponding settings.



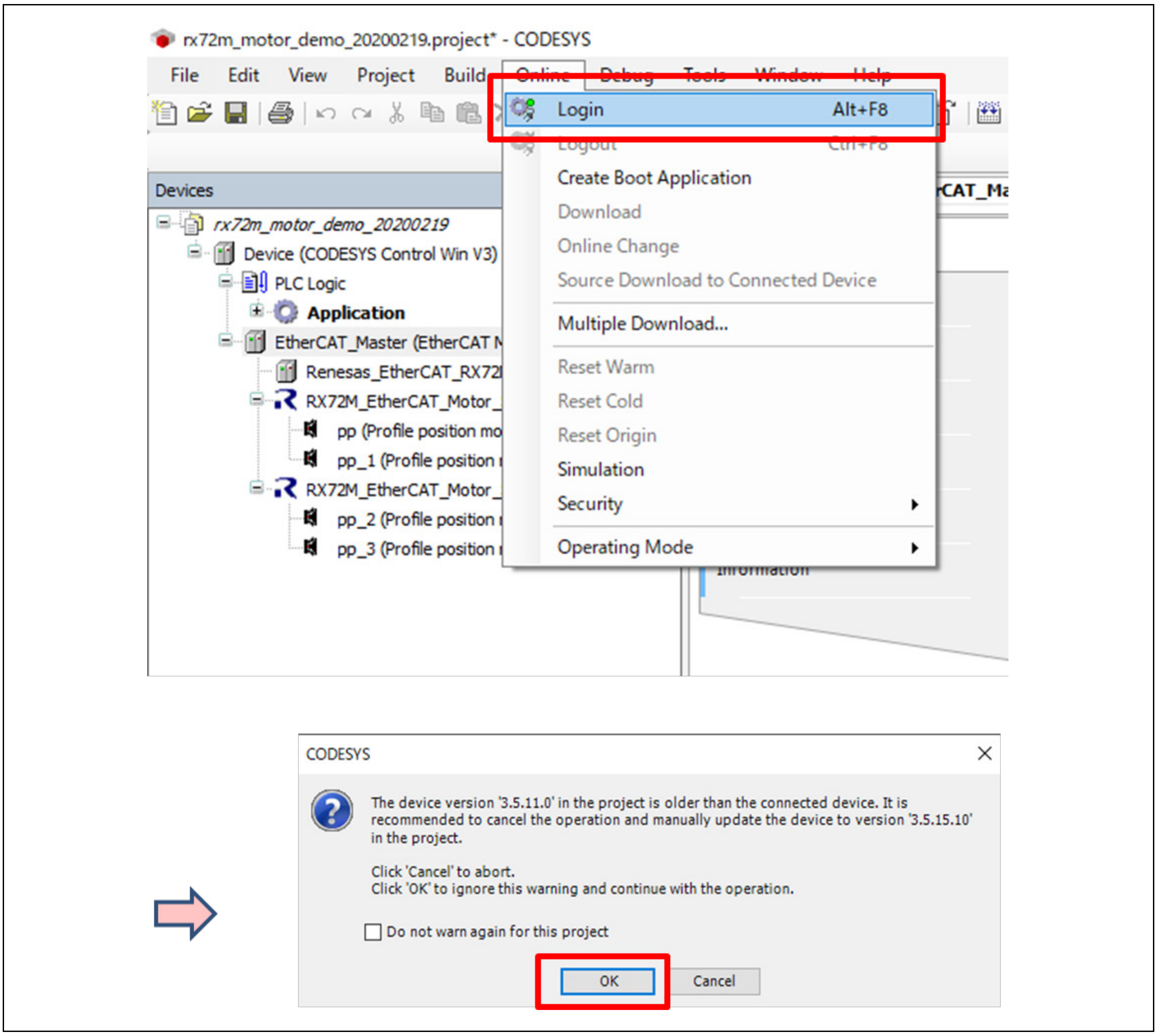
(5) Select the network to be used.
After having selected the network, press [Browse] on the [EtherCAT_Master] screen to confirm the MAC address.



(6) Build the program. Select [Build] from the [Build] menu.

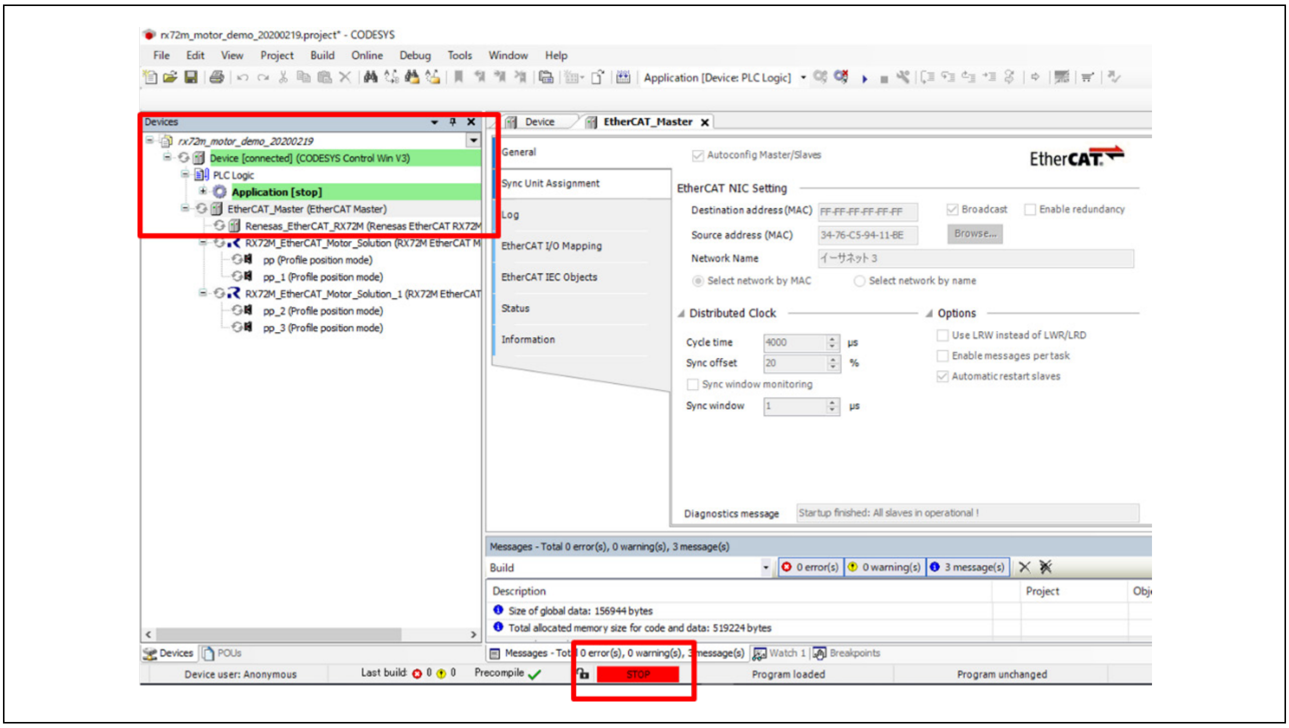


(7) After completing the build, log in. Select [Login] from the menu.

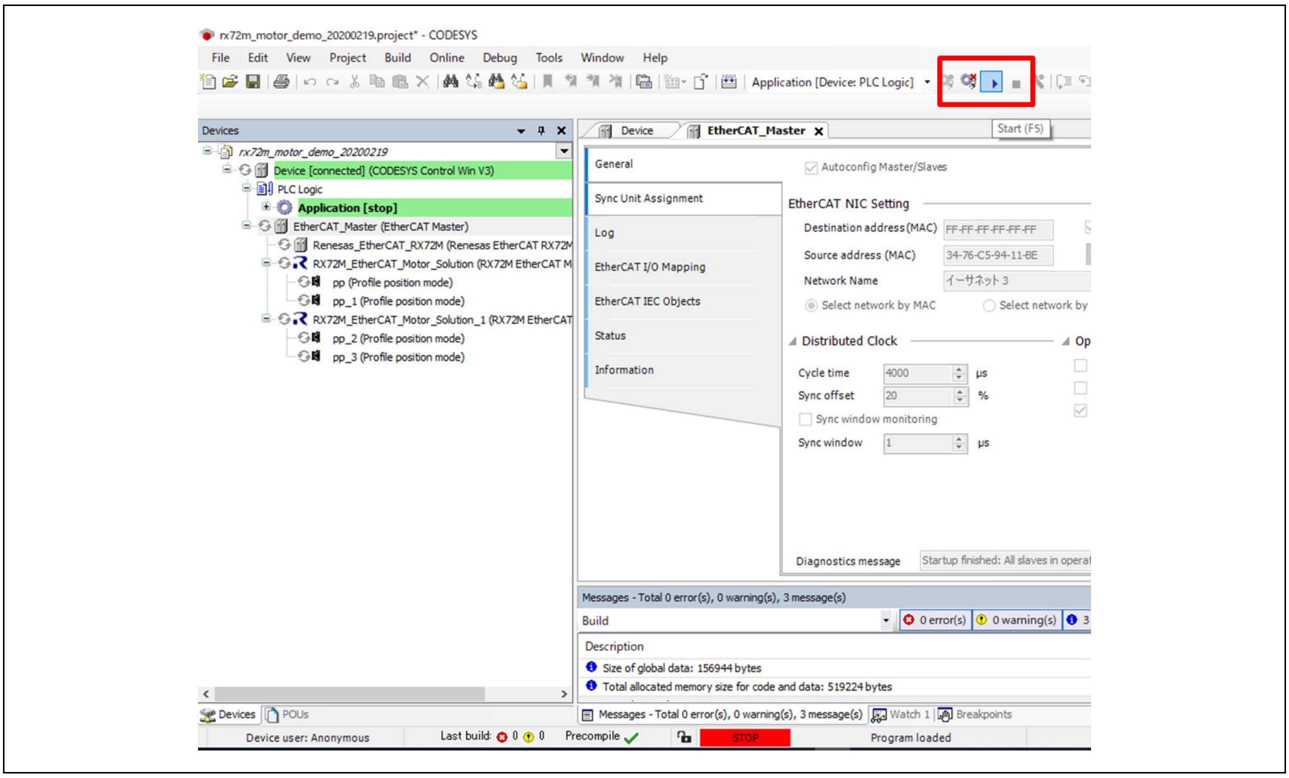


(8) After having logged in normally, the operation will be in the STOP state.

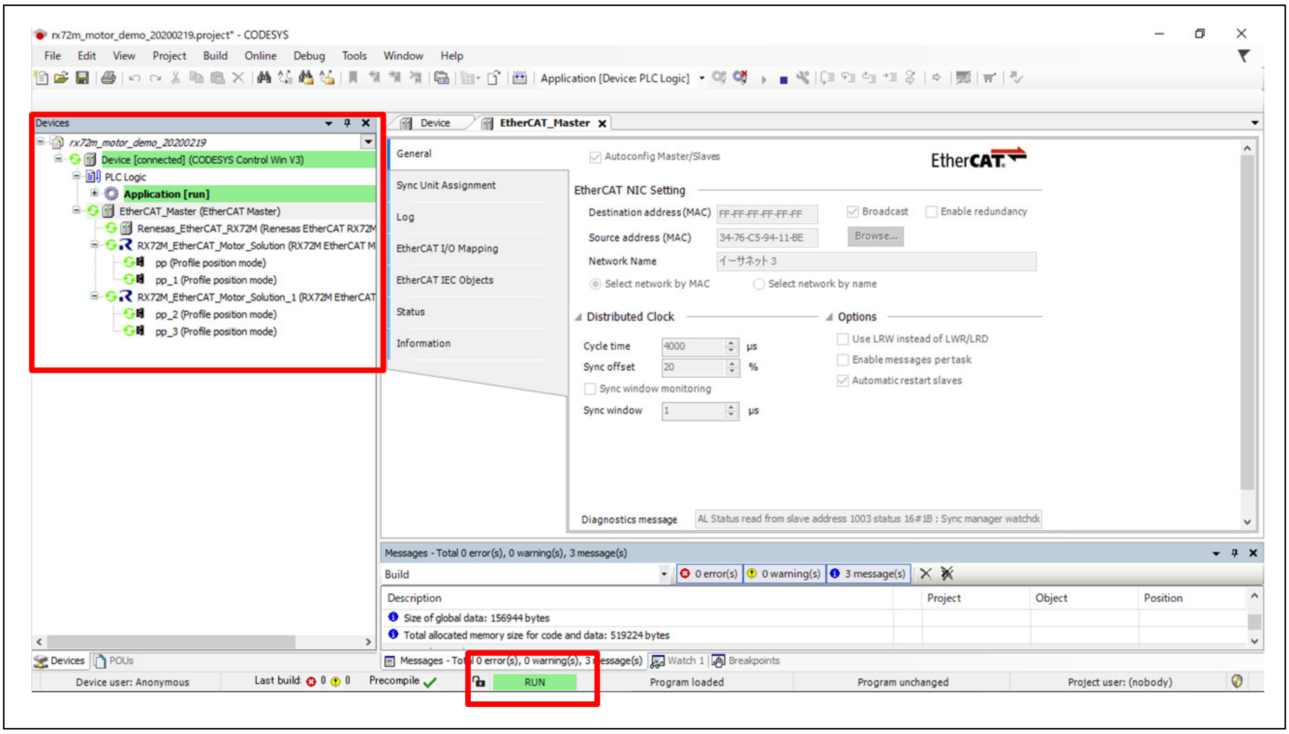
Note, however, that the state of operation may shift to RUN without having been in the STOP state.



(9) Start the operation. Press the button to start or select the run icon on the toolbar.



(10) If the connection was successful, the network connection will have been made and operation will start as shown below.



<u>Device status</u>

: The PLC is connected and the application is running.

: The PLC is connected but the application is stopped.

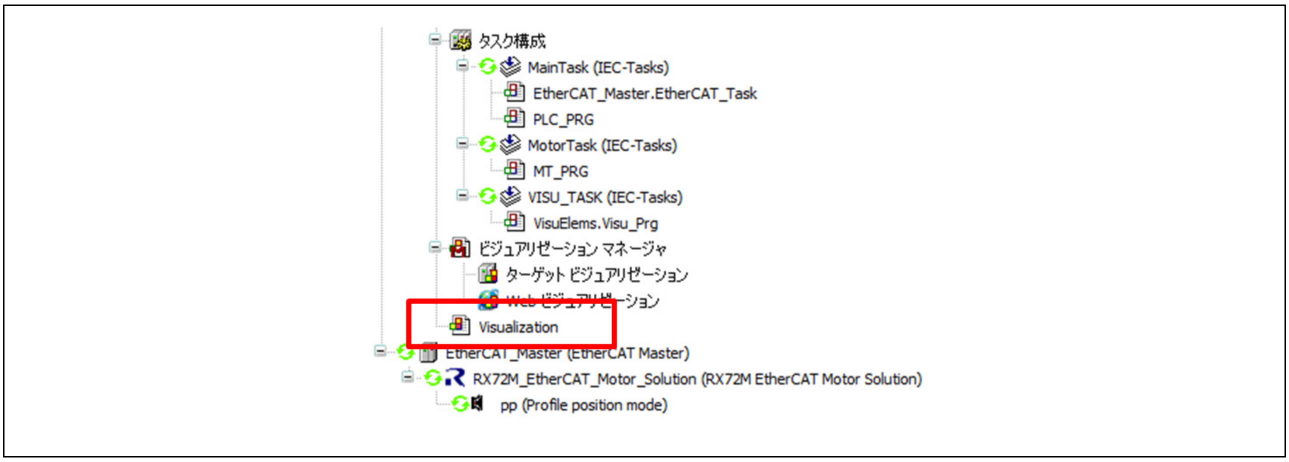
: There is an error. Check the details of the error and the device settings.

: The device information is not in the device repository. Check the device information file and install it correctly.

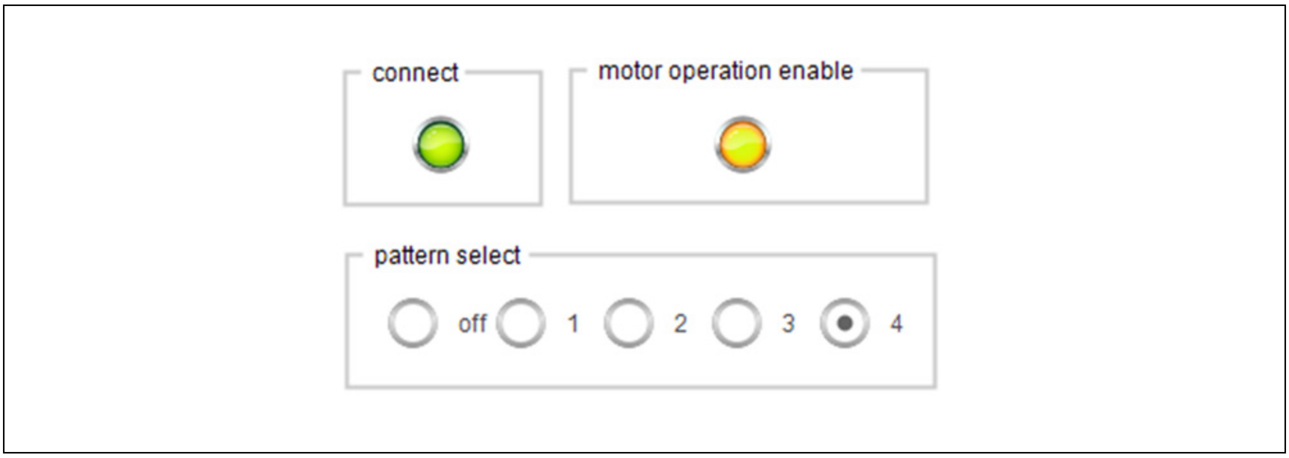
6.8 Using CODESY to Check Operation

The program for driving the attached motor is built from "rx72m_motor_control.project".
In the state where the network connection has been completed, the motor is controlled through the CODESYS GUI.

(1) Select "Visualization" from the device tree and double-click it.



(2) This starts up the GUI screen for motor control.



Pattern select is used to change the type of motor rotation operation.

1: Each set of rotations is divided into 90°, 180°, and 360° and this sequence of rotations is repeated.
2: One reverse rotation
3: A sequence of 10 rotations and then -10 rotations is repeated.
4: A sequence of -10 rotations and then 10 rotations is repeated.

connect: Indicates if communications have been established.
motor operation enable: Indicates the command transition state.

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

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

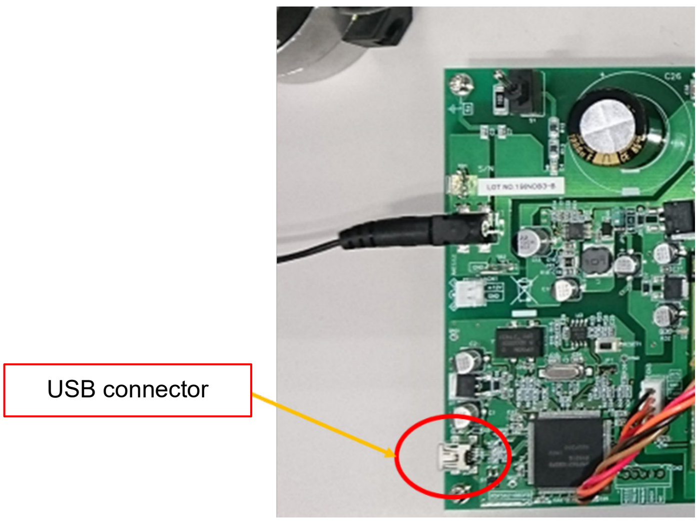
RMW (Renesas Motor Workbench) related procedure
Download RMW V2.0 from the following Web page
https://www.renesas.com/jp/ja/solutions/proposal/motor-control.html#kits
Proceed through the preparations according to Table 3.1 in the RMW user's manual (r21uz0004jj0201-motor.pdf), which will be in the RMW folder.
No 3.1, 3.5, 3.6, 3.7, 3.8
Note: The authentication file can be downloaded from [Authentication file download] at the same link as the RMW.

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）

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

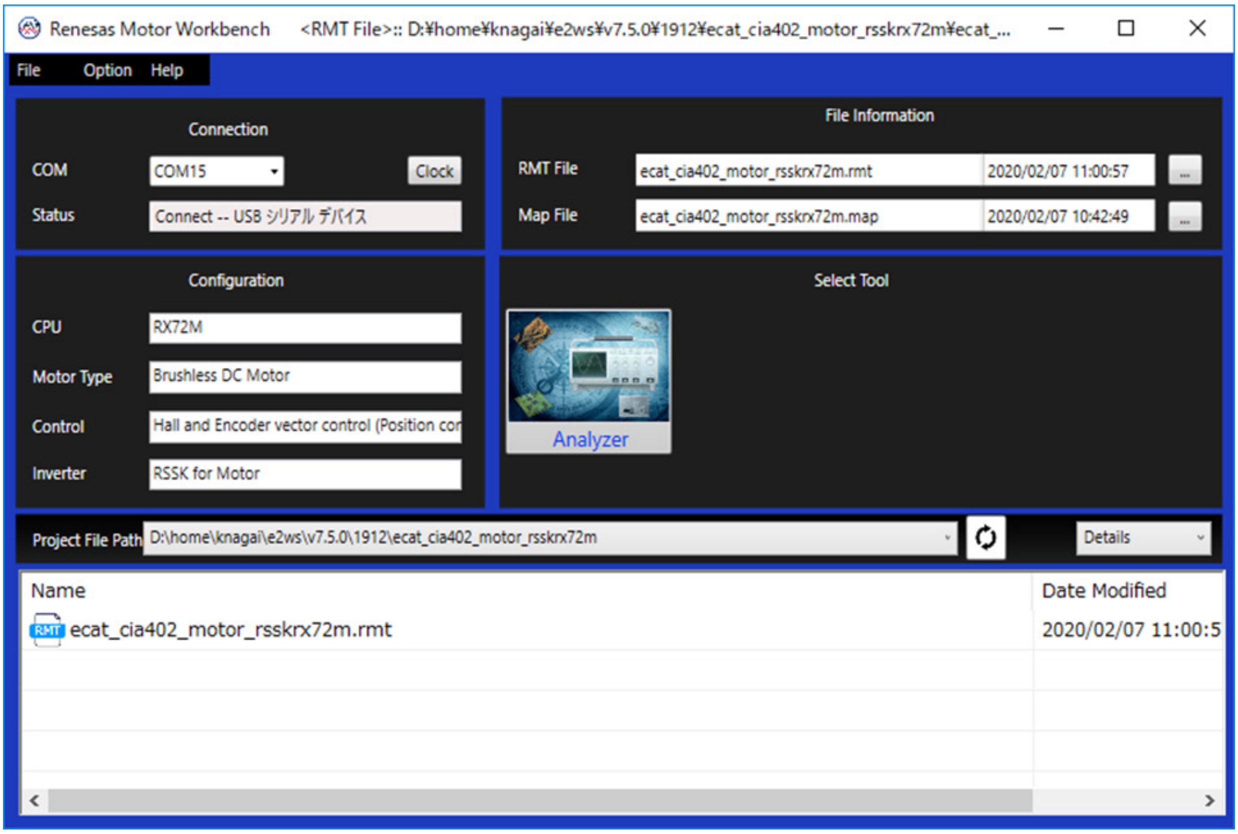


Start RMW and specify the following files.

- Environment file
  ecat_cia402_motor_rsskrx72m
  \ecat_cia402_motor_rsskrx72m.rmt
-map file
  ecat_cia402_motor_rsskrx72m\HardwareDebug
  \ecat_cia402_motor_rsskrx72m.map

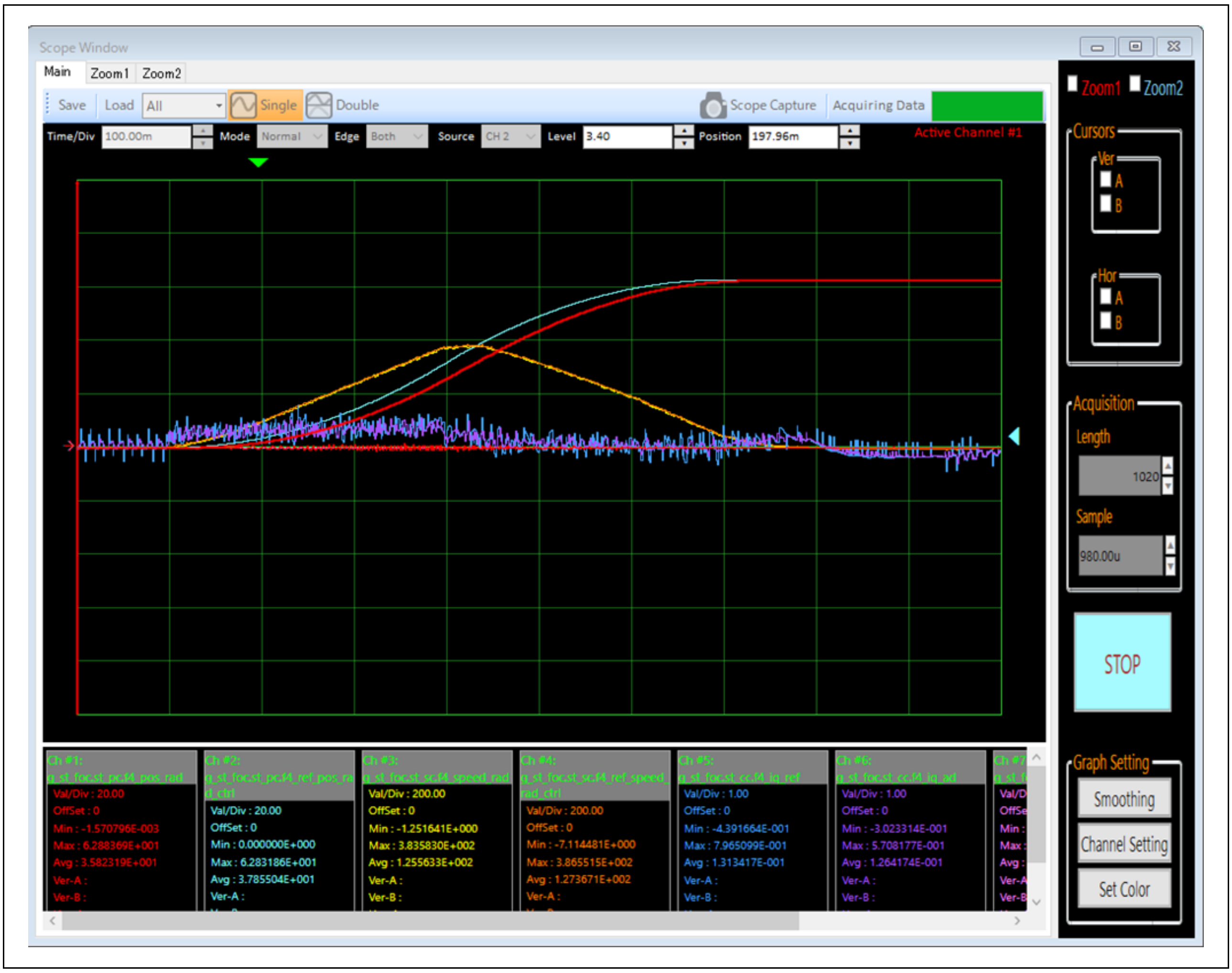
Connect RMW and the motor board

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	June 10, 2020	—	First edition issued
1.10	Aug. 31, 2020	12	To change the file configuration by supporting EtherCAT FIT module Rev.1.10 2.3.1 Table 2-7 is revised. Table 2-8 is added.
		63	The folder name of 6.3 (6) is changed.

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.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

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

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

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/.