R01AN8173EJ0100
Rev.1.00
Dec.19, 2025

RZ/T2M, RZ/T2L Group

Safety Motor Control Reference Kit Startup Manual(EtherCAT/FSoE)

Introduction

This application note is a quick start manual for RZ/T2M&RZ/T2L Safety Motor Control Reference Kit equipped with MPU of RZ/T2M group by Renesas Electronics Corporation.

Target Device

RZ/T2M, RZ/T2L Group

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 of the modified program.

Contents

1. Overview

1.1 RZ/T2M&RZ/T2L Safety Motor Control Reference Kit Overview

RZ/T2M&RZ/T2L Safety Motor Control Reference Kit (Safety Motor Control Kit) consists of the following boards and software.

- Boards equipped with RZT2M and RZ/T2Lx2 (Safety Motor Control Board)
- Firmware featuring motor control, EtherCAT communication, and FSoE-compliant safety drive functionality
- User's manual & circuit diagram of Safety Motor Control Board
-

By connecting Renesas' RZ/T series Inverter Board Kit (Inverter Board + BLDC), single-axis motor control can be achieved. Furthermore, using this kit enables motor stop functions compliant with EtherCAT (CiA402), FSoE communication, and IEC61800-5-2 standards: STO and SS1-t.

The software features are as follows:

- Vector control of permanent magnet synchronous motors using a serial encoder
- EtherCAT (CiA402) and FSoE communication
- Motor stop functions compliant with IEC61800-5-2: STO and SS1-t

1.2 Connection configuration

Figure 1-1 is shown the Safety Motor Control Kit connection configuration.

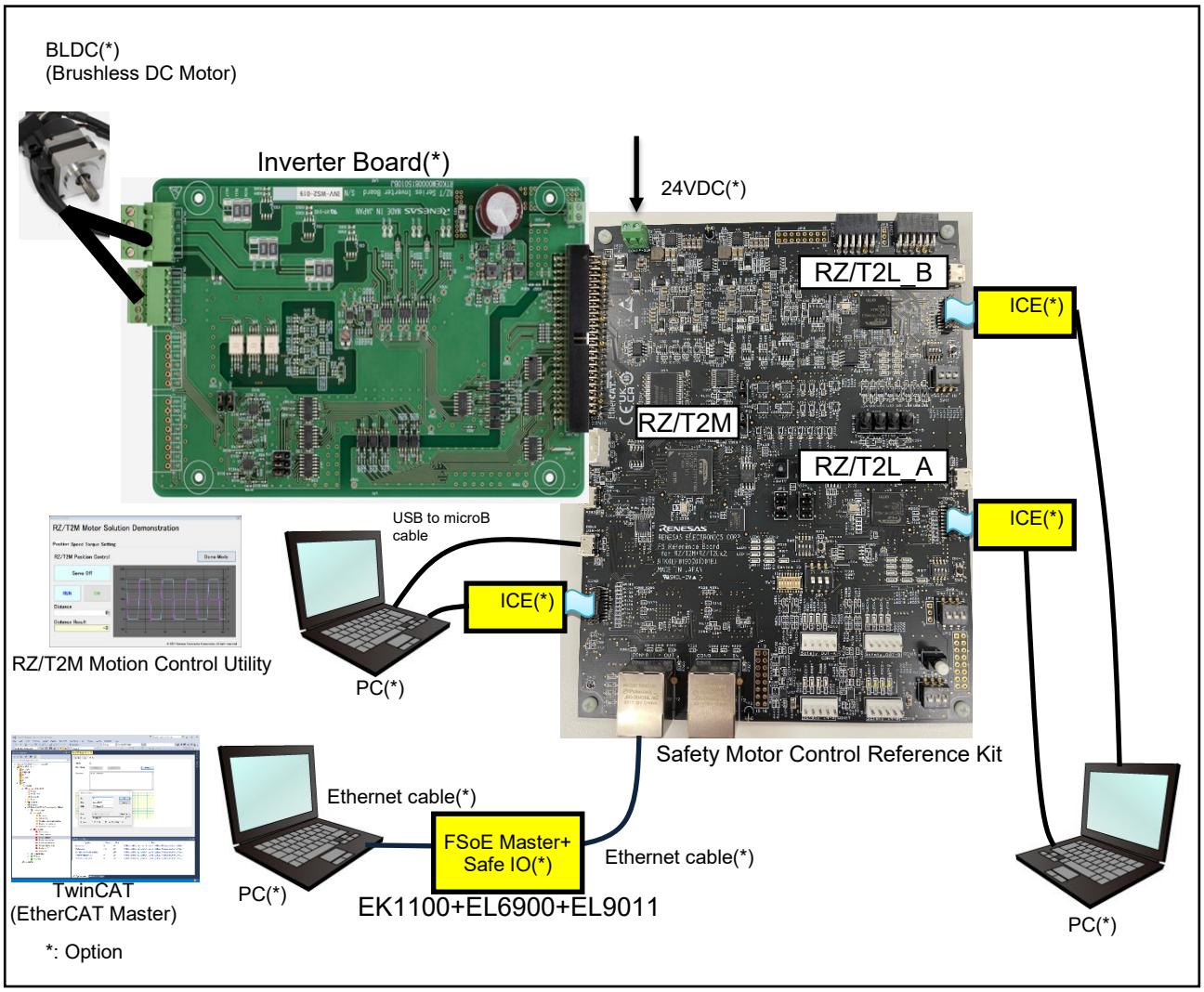


Figure 1-1 Safety Motor Control Kit connection configuration

1.3 Connection Equipment

Table 1.1 is shown the connection equipment list.

Table 1.1 connection equipment list

Item	Model	Remarks
RZ/T Series Inverter Board Kit	RTK0EM0000S05010BJ (Renesas) (*) ・Inverter Board ・TSM3101N2001E020 (Tamagawa Seiki)	-
ICE	・I-jet (IAR Systems) ・J-Link Base Ver.11.0 or later (SEGGER)	-
FSoE Master	・EK1100 (Beckhoff Automation) ・EL6900 (Beckhoff Automation) ・EL9011 (Beckhoff Automation)	-

(*) RTK0EM0000S05010BJ can be purchased from the following URL.
https://www.renesas.com/invb-lv-rzt-b

1.3.1 Safety Motor Control Board and Inverter Board Connection

Safety Motor Control Board and Inverter Board connect directly.

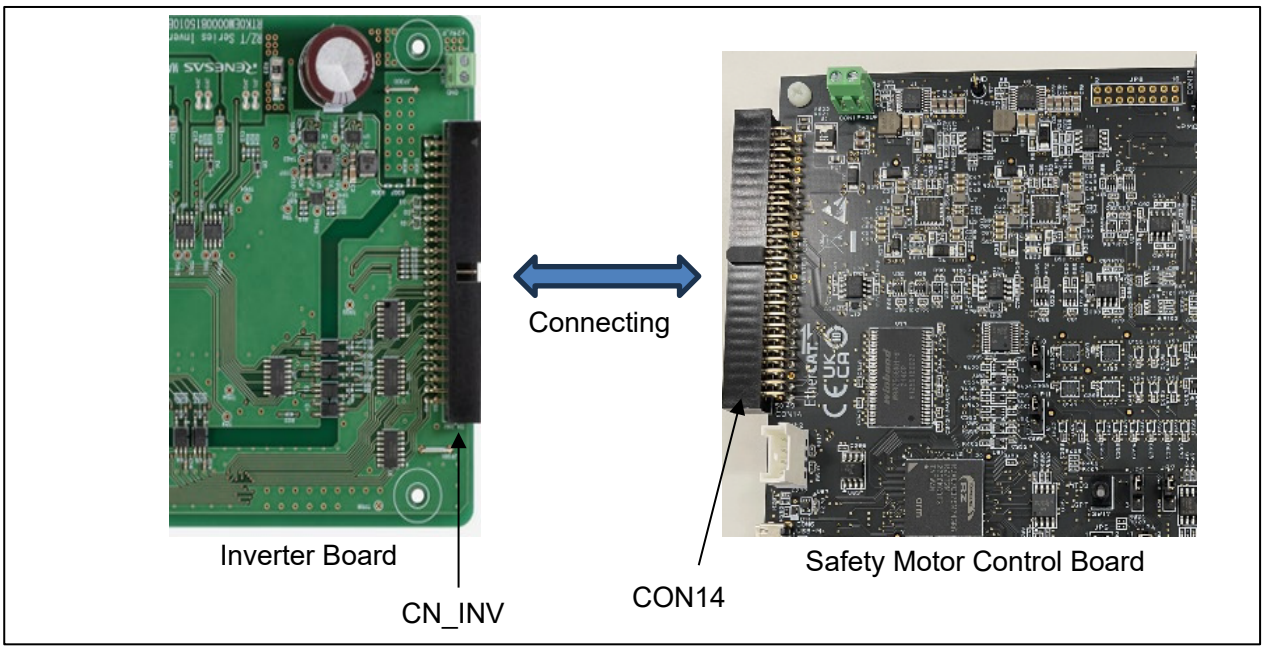


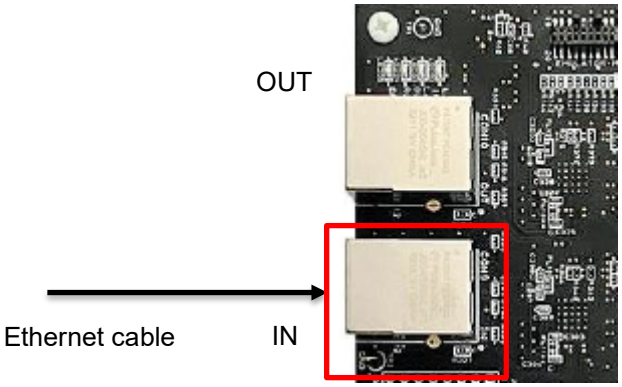
Figure 1-2 Safety Motor Control Board and Inverter Board connection

1.3.2 Motor/ Encoder and Inverter Board Connection

For Tamagawa Seiki: TSM3101N2001E020 (Motor/ Encoder (FA-CODER)) and Inverter Board Connection, please see “RZ/T Series Inverter Board/Kit User’s Manual (R12UZ0155)”.

1.3.3 PC Connection

Connect the Ethernet cable to the OUT side of the FSoE master and the IN side of the RJ45 connector on the safety motor control board.



1.3.4 ICE

1.3.4.1 RZ/T2M / RZ/T2L_A /RZ/T2L_B

For how to connect ICE of RZ/T2M, RZ/T2L_A and RZ/T2L_B, refer to “RTK0EF0190D01001BJ User’s Manual (R30UZ0214)”.

1.3.5 Power Supply

DC24V is supplied to Safety Motor Control Board.
For how to power supply, refer to “RTK0EF0190D01001BJ User’s Manual (R30UZ0214)”.

If Inverter Board is connected, 24 VDC can be supply from Safety Motor Control Board to Inverter Board. In this case, don’t supply 24 VDC to Inverter Board.

Just in case, current of Inverter Board is not enough, power of Inverter Board and Safety Motor Control Board can be separated. If you want to power them separately, refer to “RTK0EF0190D01001BJ User's Manual (R30UZ0214)”.

2. Operating Environment

2.1 RZ/T2M

Table 2.1 Operating environment (RZ/T2M)

Item	Content
Board	Safety Motor Control Board
MPU	RZ/T2M Group R9A07G075M28GBG: 320pinFBGA
Operating frequency	CPU Core0：800MHz（Arm® Cortex®-R52） CPU Core1：800MHz（Arm® Cortex®-R52）
Operating voltage	3.3V/1.8V/1.1V
Operating mode	xSPI0 boot mode(x1 boot serial flash)
Device	Serial Flash ROM (64Mbyte) RENESAS AT25SF128A
Communication protocol	EtherCAT®
Integrated development environment	RENESAS e² studio 2025-04-1 (25.4.1)
Emulator	SEGGER J-Link Base Ver.11.0 or later
SSC Tool	Provided by EtherCAT Technology Group (ETG) Slave Stack Code (SSC) Tool Version 5.13
Software PLC	Beckhoff Automation TwinCAT® 3
Flexible Software Package (FSP)	Version.2.3.0

2.2 RZ/T2L_A, RZ/T2L_B

Table 2.2 Operating environment (RZ/T2L_A, RZ/T2L_B)

Item	Content
Board	Safety Motor Control Board
MPU	RZ/T2N GroupRZ/T2L Group R9A07G074M04GBG: 196pin LFBGA
Operating frequency	800MHz（Arm® Cortex®-R52）
Operating voltage	3.3V/1.8V/1.1V
Memory	Serial Flash ROM (64Mbyte) RENESAS AT25SF128A
Integrated development environment	IAR Systems Embedded Workbench® for Arm Version 9.20.3 Functional Safety
Emulator	IAR Systems I-jet

For this software, Please download the following URL.

https://info.renesas.com/rel-ia-fusa-software-info-request

3. Related Application Notes

The application notes related to this application note are listed below for reference.

- RTK0EF0190D01001BJ User's Manual (R30UZ0214)
- RZ/T Series Inverter Board/Kit User's Manual (R12UZ0155)
- Encoder Vector Control of Permanent Magnet Synchronous Motor - Serial Encoder (R01AN8003)

- CiA402 Documents:

- IEC 61800-7-201 Edition 1.0

Adjustable speed electrical power drive systems Part 7-201: Generic interface and use of profiles for power drive systems Profile type 1 specification

- IEC 61800-7-301 Edition 1.0

Adjustable speed electrical power drive systems Part 7-301: Generic interface and use of profiles for power drive systems Mapping of profile type 1 to network technologies

4. Board Setting

4.1 Safety Motor Control Board Setting

4.1.1 Switches

SW1

Pin	1	2	3
	OFF	OFF	OFF

SW3 - 5

Pin	1	2	3	4
	ON	ON	ON	OFF

SW7

Pin	1	2	3
	OFF	OFF	OFF

SW9

Pin	1	2	3
	OFF	OFF	OFF

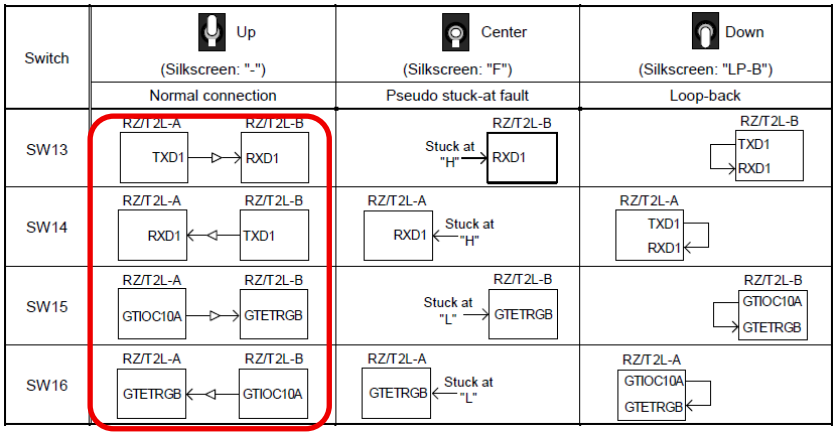
SW11

Pin	1	2	3
	OFF	OFF	OFF

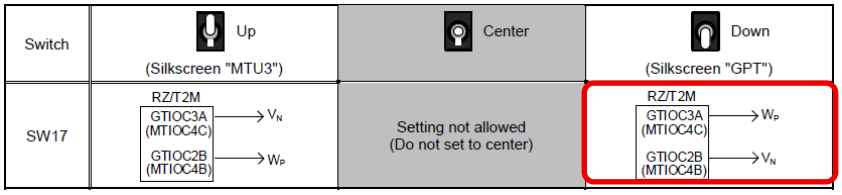
SW12

Pin	1	2	3	4	5	6	7	8
	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

SW13 -16



SW17



For switch details of Safety Motor Control Board, refer to the “RTK0EF0190D01001BJ User’s Manual (R30UZ0214)”.

4.1.2 Jumpers

No	JP	Setting
1	JP4	1-2 short
2	JP5	1-2 short
3	JP6	5-6 short
4	JP7	1-2 short
5	JP10	1-2 short
6	JP11	1-2 short

For Jumper details of Safety Motor Control Board, refer to the “RTK0EF0190D01001BJ User’s Manual (R30UZ0214)”.

4.2 Inverter Board

4.2.1 Jumpers

No	JP	Setting
1	JP7,8,9	1-2 short
2	JP10	2-3 short
3	JP11	2-3 short

For Jumper details of Inverter Board, refer to the “RZ/T Series Inverter Board/Kit User’s Manual (R12UZ0155)”.

5. How to Write Program

5.1 RZ/T2M

5.1.1 Generating the EtherCAT SubDevice Stack

Note:

This sample project does not include the EtherCAT SubDevice stack code.

To generate EtherCAT slave stack code, you need the EtherCAT SubDevice Stack Code (SSC) Tool.

The SSC Tool can be obtained from the ETG (EtherCAT Technology Group).

Before launching the SSC Tool, please check “Appendix : A point of caution when using SSC Tool”.

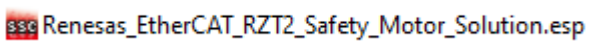
To execute this guide, patch.exe must be installed on your PC.

Refer to ”Appendix : How to install patch” and install patch.exe.

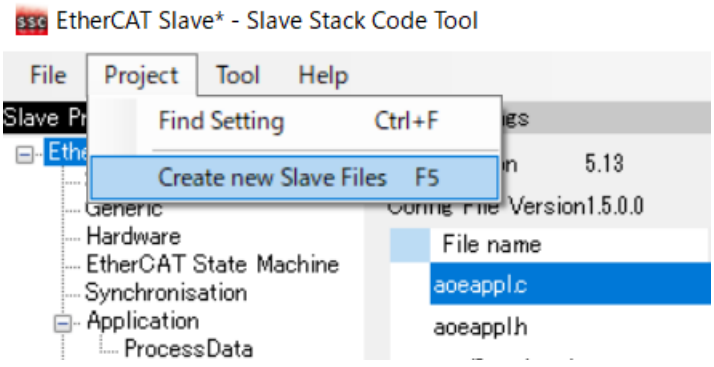
1. Extract RZT2M\RZT2M_SMCRK.zip so that the folder structure looks like this:

```
r01an8173ej0100-rzt2m-rzt2l-smcr-package
├──Common
└──RZT2M
    └──RZT2M_SMCRK
        ├──RZT2M_SMCRK_ESC_E2S_V100
        └──RZT2M_SMCRK_INVBLB_SPM_ENCD_FOC_E2S_V100
```

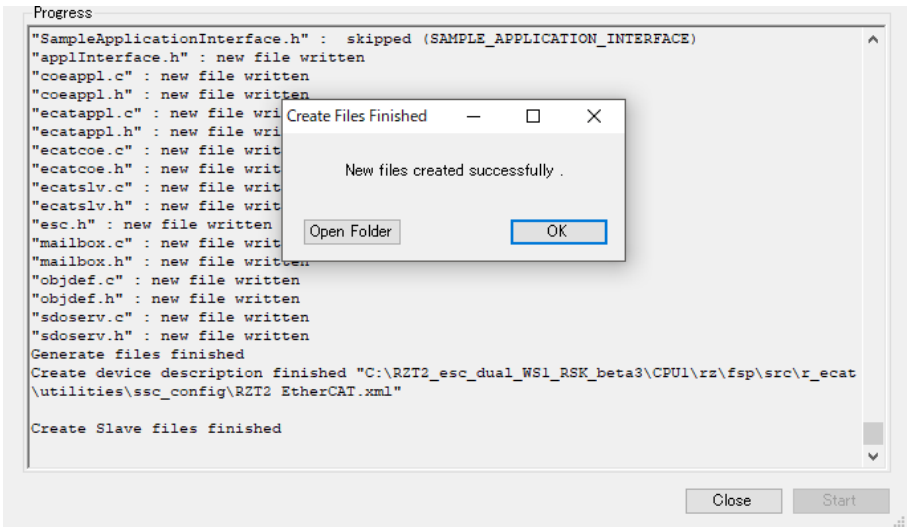
2. Double-click the .esp file located in the “Common\SSCConfig” folder to launch the SSC Tool.



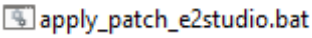
3. Click [Project] → [Create New Slave Files], then click [Current new Slave Files] → [Start].



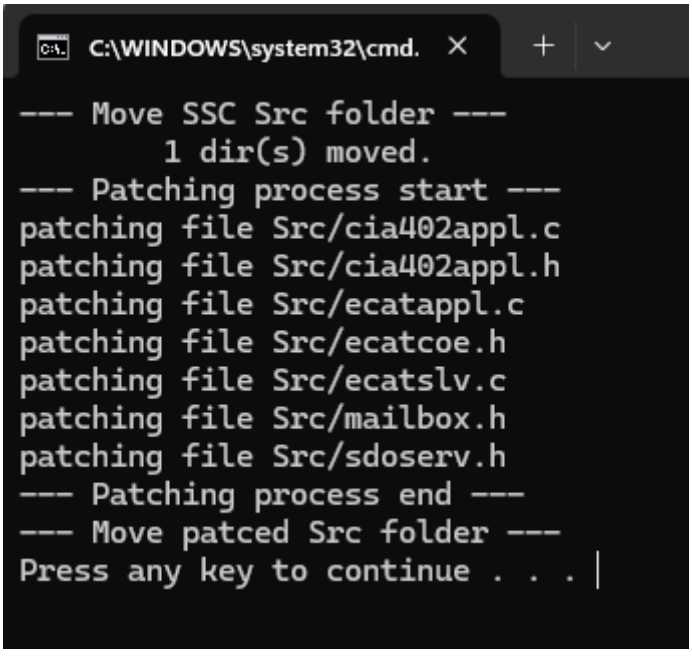
4. When the source code is successfully generated, the message “New Files created successfully” will appear. Click [OK].



5. Double-click [apply_patch_e2studio.bat] located in the “Common\Patch” folder to execute the batch file.

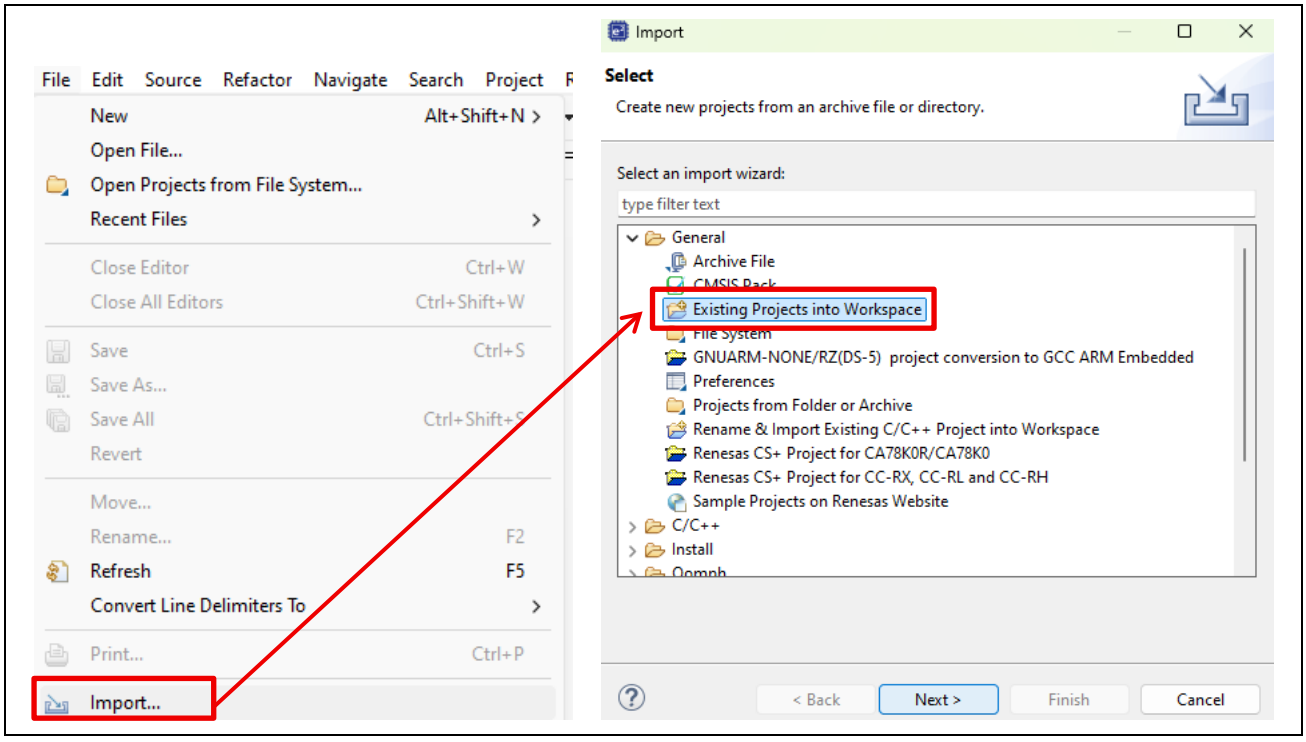


6. After executing the batch file, the Src folder will be moved to: RZT2M\RZT2M_SMCRK\RZT2M_SMCRK_ESC_E2S_V100\src\ethercat\beckhoff.



5.1.2 Building the Project

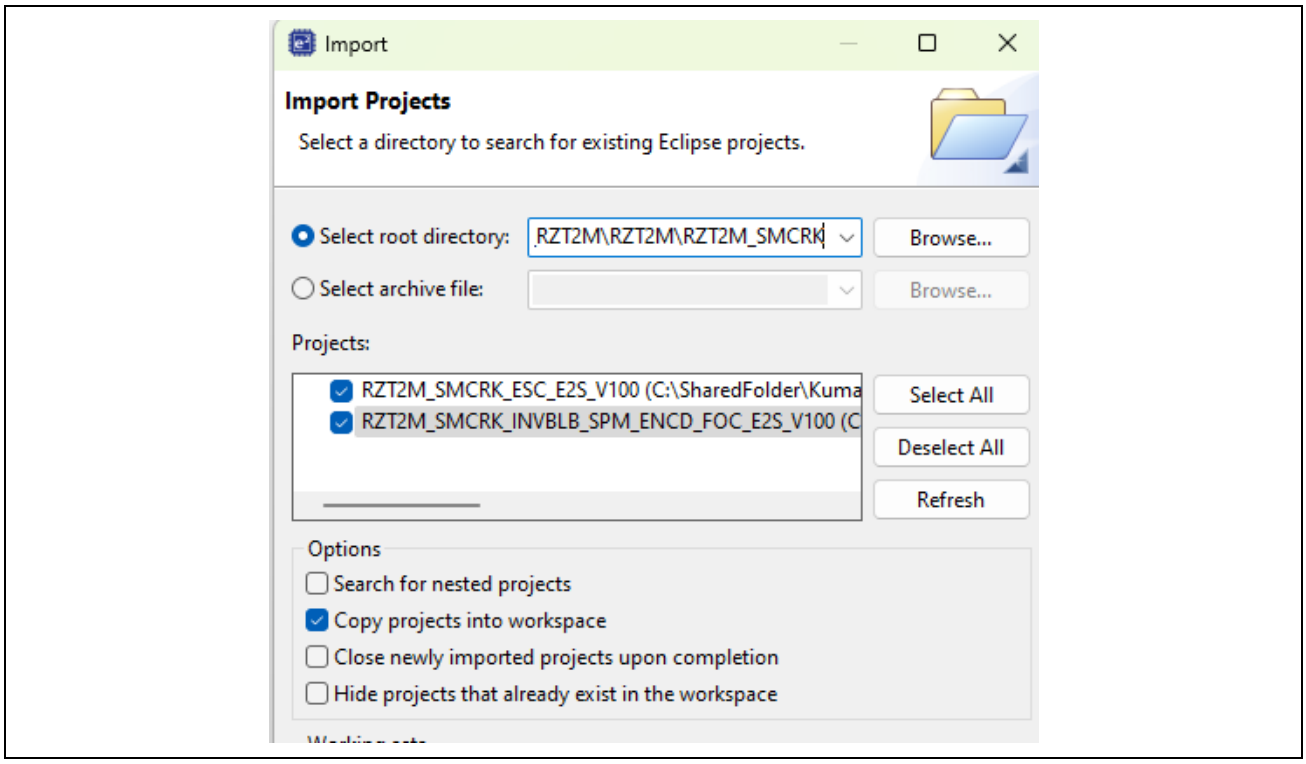
(1) Launch e² studio, then from the top menu select: [File] > [Import] > [Existing Projects into Workspace], and click Next.



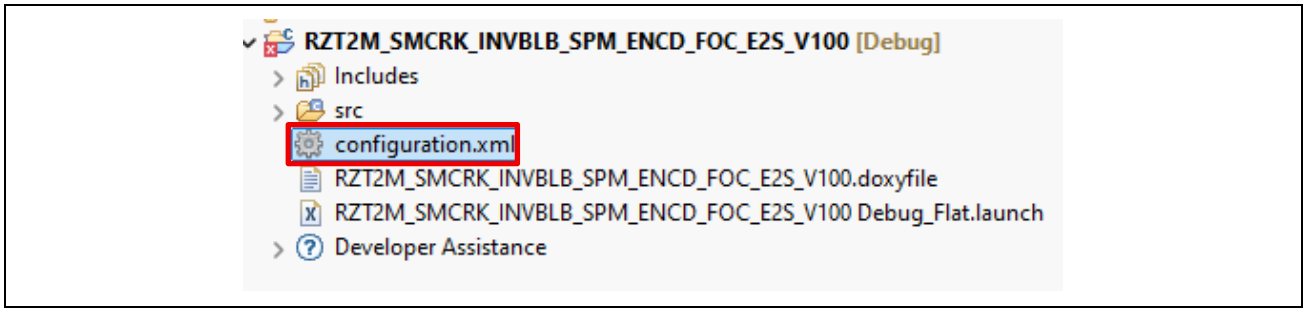
(2) Select “Select root directory”, then choose the “RZT2M\RZT2M_SMCRK” folder.

(3) Check both : “RZT2M_SMCRK_INVBLB_SPM_ENCD_FOC_E2S_V100” and “RZT2M_SMCRK_ESC_E2S_V100”

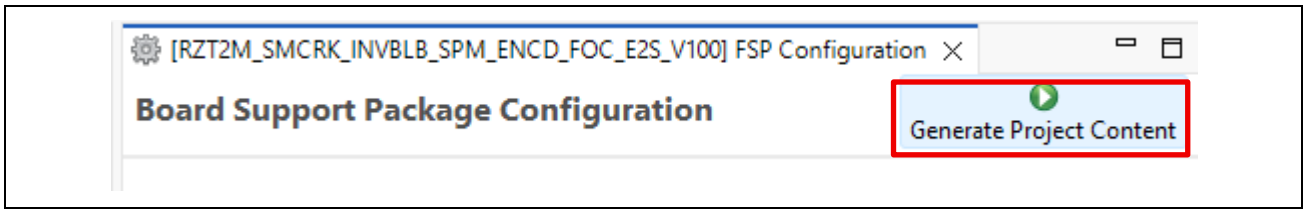
Also, under Options, check “Copy projects into workspace” to import while copying into the workspace.



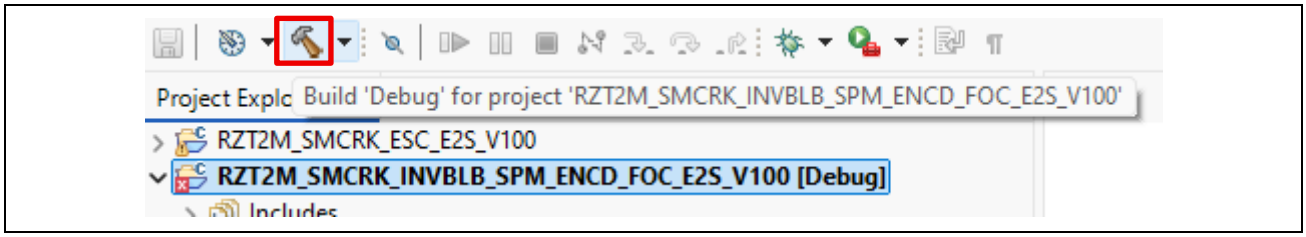
(4) Open configuration.xml in the RZT2M_SMCRK_INVBLB_SPM_ENCD_FOC_E2S_V100 project.



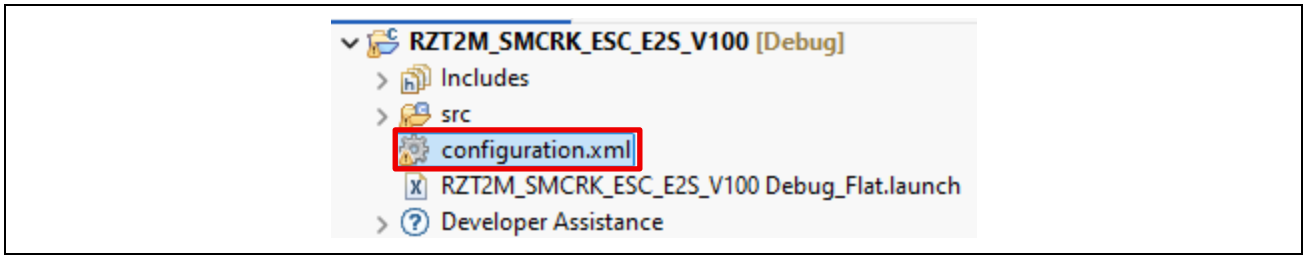
(5) Perform code generation.



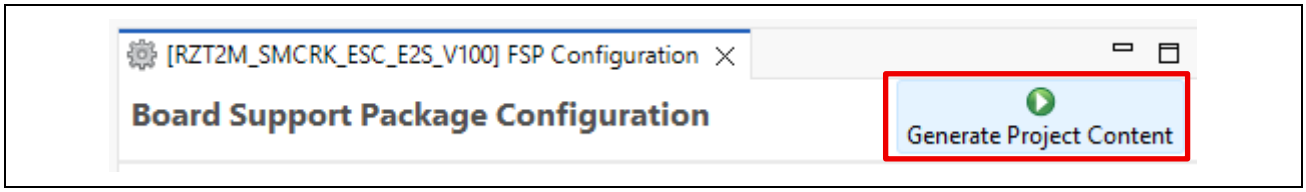
(6) Build the RZT2M_SMCRK_INVBLB_SPM_ENCD_FOC_E2S_V100 project.



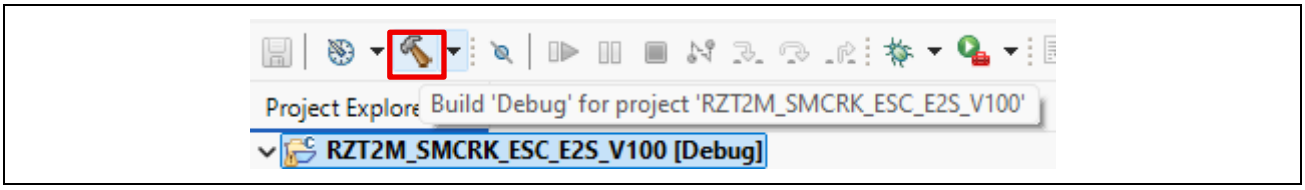
(7) Open configuration.xml in the RZT2M_SMCRK_ESC_E2S_V100 project.



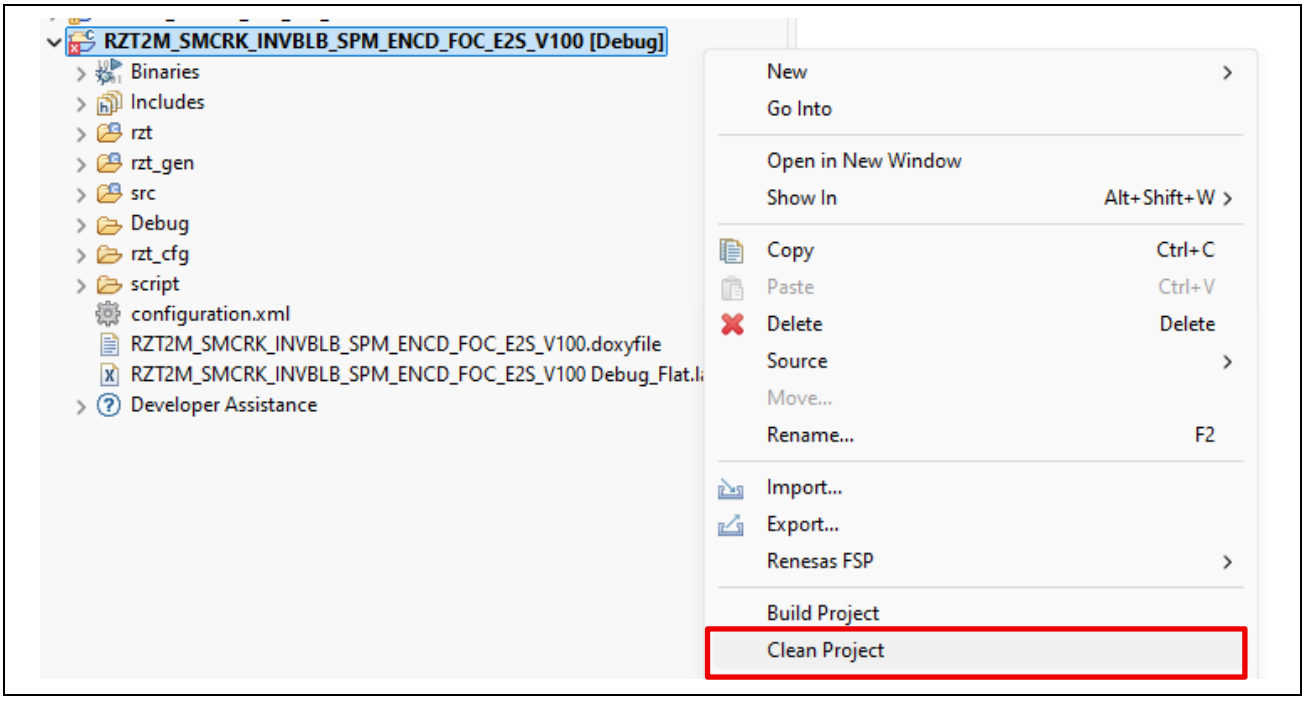
(8) Perform code generation.



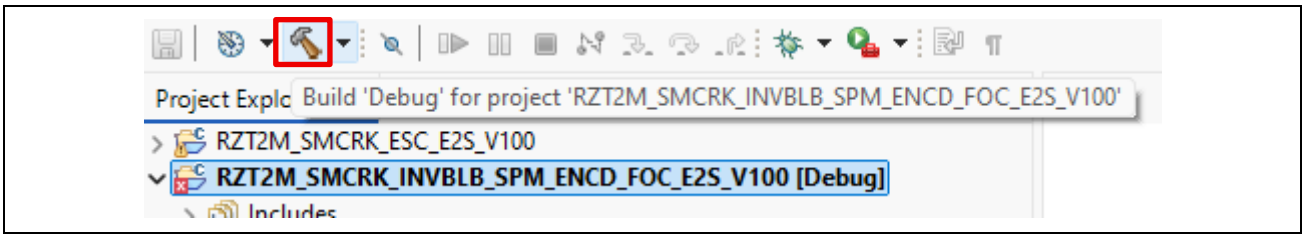
(9) Build the RZT2M_SMCRK_ESC_E2S_V100 project.



(10) Clean the RZT2M_SMCRK_INVBLB_SPM_ENCD_FOC_E2S_V100 project.



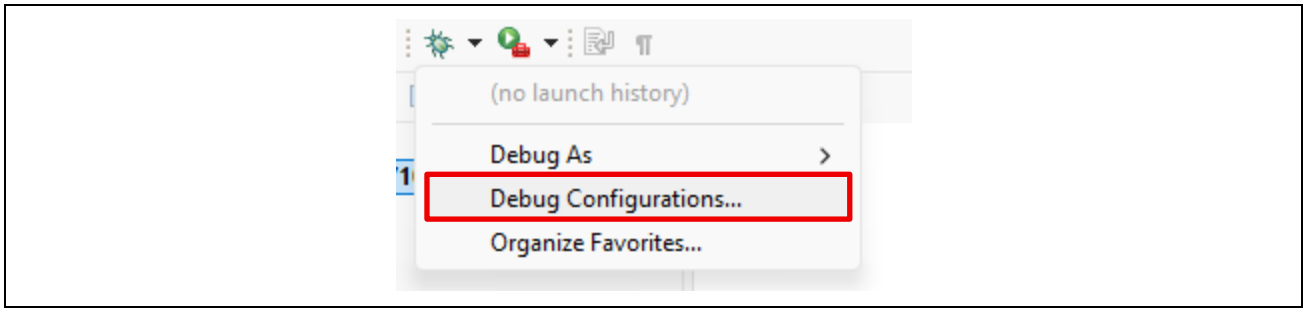
(11) After cleaning, build the RZT2M_SMCRK_INVBLB_SPM_ENCD_FOC_E2S_V100 project again.



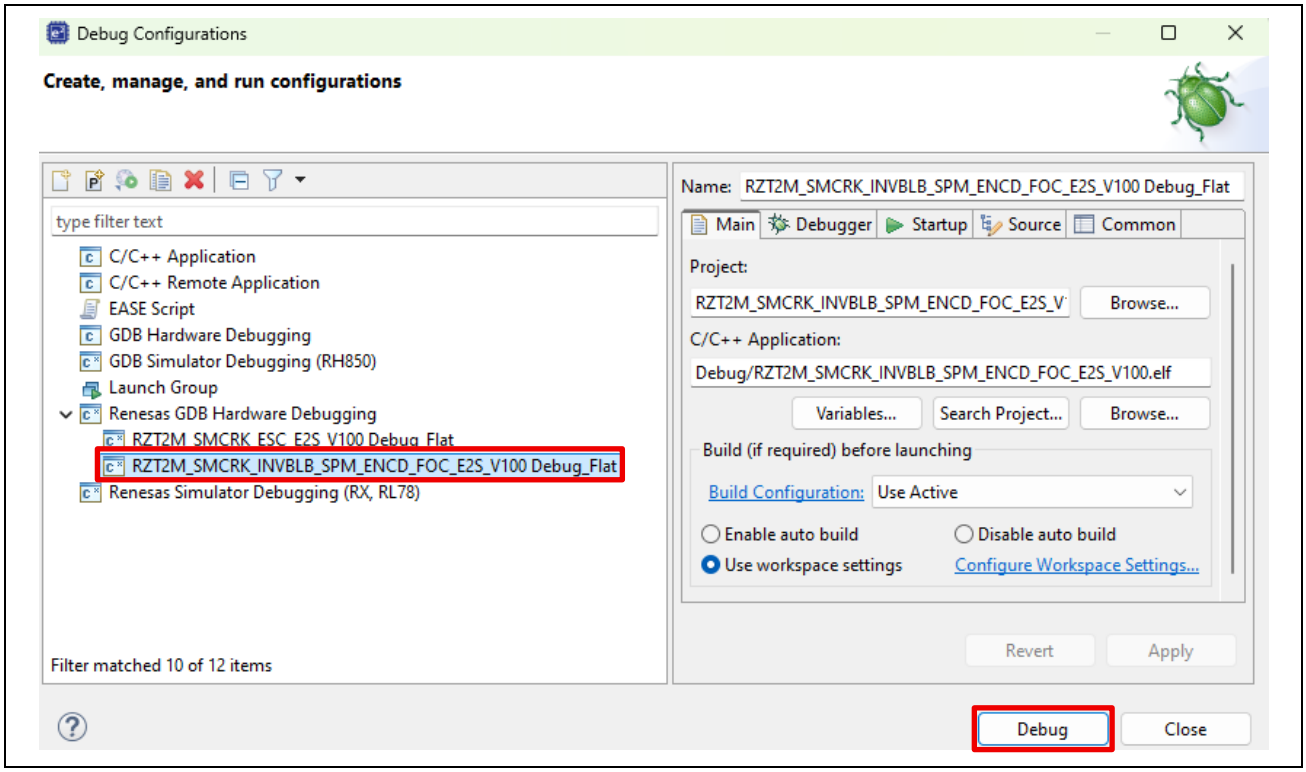
5.1.3 Writing the Project

(1) Apply 24V DC to the Safety Motor Control Board and turn on the power.

(2) Connect the J-Link debugger to the RZ/T2M, then open the Debug Configuration.



(3) Select “RZT2M_SMCRK_INVBLB_SPM_ENCD_FOC_E2S_V100 Debug_Flat” and start debugging.



(4) After the writing process completes and debugging starts, terminate the debug session and press the reset button (SW2) on the board.

This sample runs in standalone.

5.2 RZ/T2L

This chapter describes the procedure for writing the RZ/T2L functional safety control software to the flash ROM of the reference board "RTK0EF0190D01001BJ". The diagrams in this chapter are examples on the RZ/T2L-A side, but the following steps should be performed separately for both the RZ/T2L-A and RZ/T2L-B.

5.2.1 Development Tools

The following tool is required to use the RZ/T2L Functional Safety Control Software (Table 7.66, #4 and #8).

- IAR Systems Embedded Workbench for ARM, Functional Safety edition（EWARM-FS）v9.20.3 *1 *2

The following tool is required to use the project (Table 7.66, #5 and #9) that writes the RZ/T2L functional safety control software to the flash ROM of the reference board "RTK0EF0190D01001BJ".

- IAR Systems Embedded Workbench for ARM v9.20.3 or later *3

For specifications and usage of the EWARM-FS, I-jet emulator, and reference board, refer to the respective manuals.

*1: EWARM-FS 9.20.3 SP1 is not supported.
*2: Before using RZ/T2M on EWARM-FS 9.20.3, apply the dedicated device patch to EWARM-FS 9.20.3. Device patch is provided by IAR Systems.
*3: If you use a version other than EWARM-FS v9.20.3, please convert it to support the target tool version when opening the project.

5.2.2 Copying Workspace Folder

The following describes how to copy a workspace folder.

1) Copy the two RZ/T2L workspace folders for the redundant system (named: for RZ/T2L-A "FSoE_SW_M", for RZ/T2L-B "FSoE_SW_S") to any location on your PC.

（Example）

For RZ/T2L-A : C:\FSoE_SW_M ← (Copy Table 7.66, #2)

For RZ/T2L-B : C:\FSoE_SW_S ← (Copy Table 7.66, #6)

This chapter is explained under the following assumptions. Please replace data paths and names as necessary.

[RZ/T2L-A]

- Workspace path : "C:\FSoE_SW_M"
- Workspace name : "RZT2L_main"
- Path for project file : "C:\FSoE_SW_M\RZT2L\PL-SW"
- Project name : "RZT2L"
- Hex file output address : "C:\FSoE_SW_M\RZT2L\PL-SW\Debug\Exe"

[RZ/T2L-B]

- Workspace path : "C:\FSoE_SW_S"
- Workspace name : "RZT2L_sub"
- Path for project file : "C:\FSoE_SW_S\RZT2L\PL-SW"
- Project name : "RZT2L"
- Hex file output address : "C:\FSoE_SW_S\RZT2L\PL-SW\Debug\Exe"

5.2.3 Opening Workspace

The following describes how to open workspace for this project.

1) Start EWARMFS 9.20.3 from Windows [Start] menu.
2) On the [File] menu, select [Open Workspace]. (Figure 5.1, #1 #2)
3) Specify the workspace files in the PL-SW folder (RZT2L_main.eww / RZT2L_sub.eww). (Figure 5.1, #3)

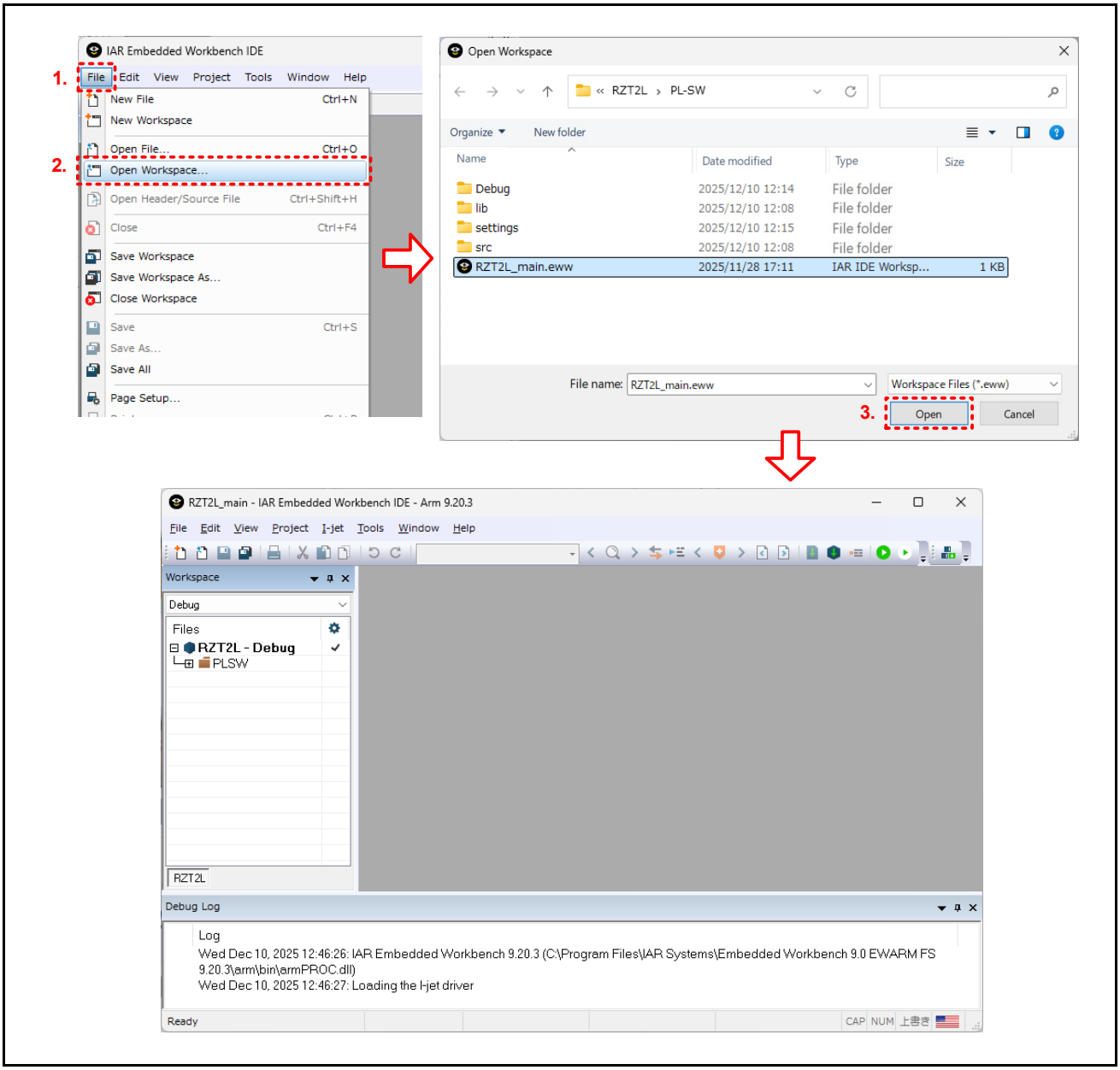


Figure 5.1 Opening Workspace

Procedures of opening workspace are complete now.

5.2.4 Building the Project

The following describes the steps to build the project. In this projects, build options are optimized for application by default.

1) Select [Project] > [Rebuild All] to run a build. (Figure 5.2, #1 #2)
2) When the build is complete, error statistics are displayed in the [Build] window.

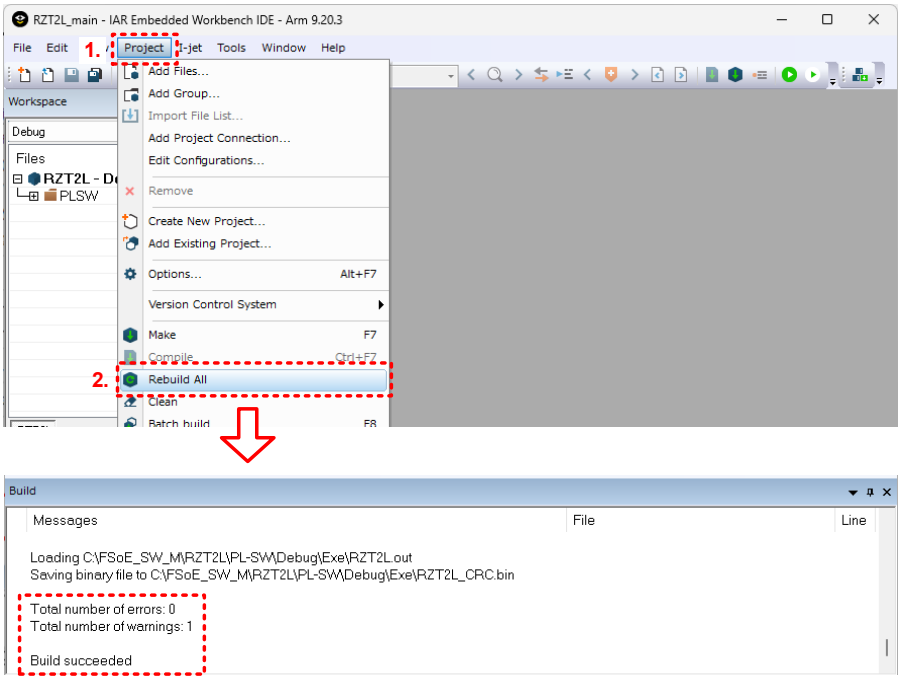


Figure 5.2 Building the Project

Note: Due to the software specifications, one warning ([Pe177]) will occur, but please ignore it.

Build operation is complete now.

5.2.5 Writing to ROM

Figure 5.3 shows the area addresses of flash memory and RZ/T2L internal RAM where project codes are written.

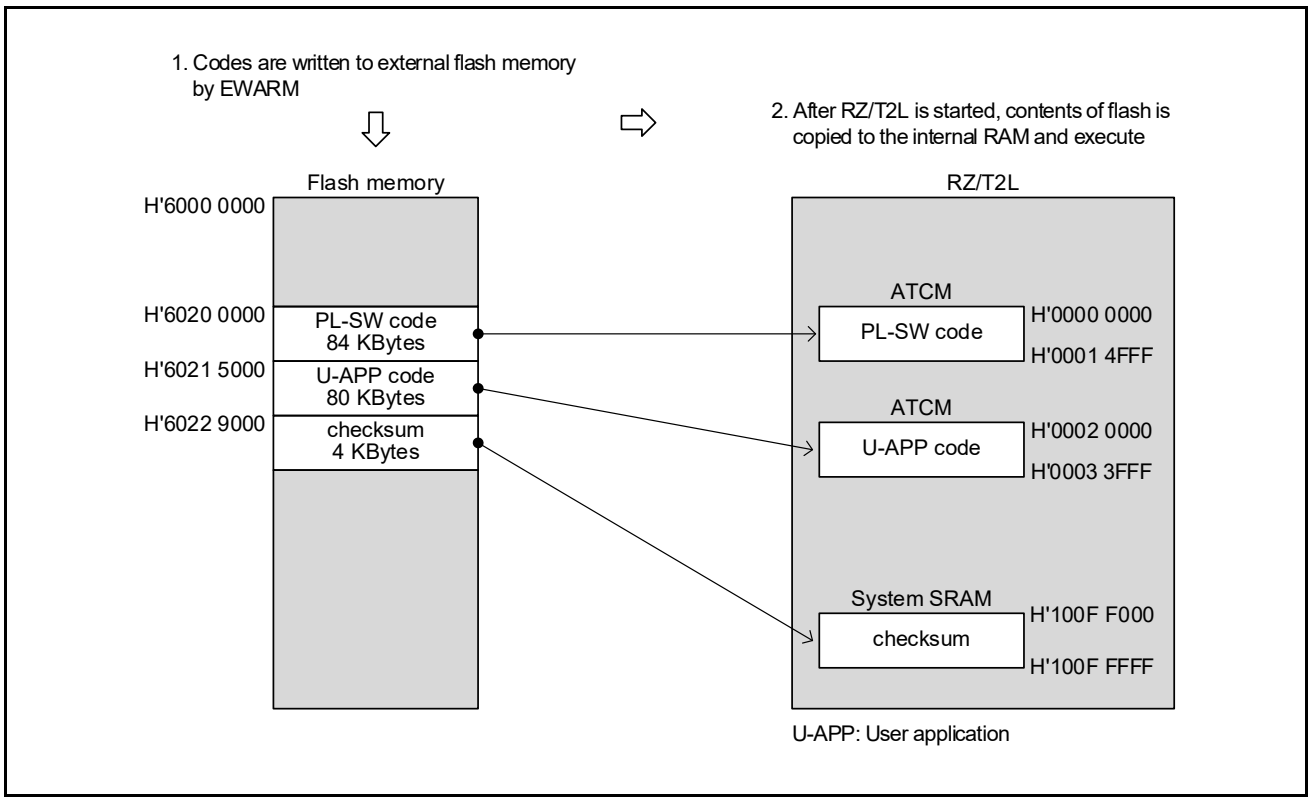


Figure 5.3 Code Area Address

The following describes the procedure for writing a project to flash memory.

1) Overwrite the arm folder (Table 7.66, #11) in the flashloader folder (Table 7.66, #10) with the arm folder in the EWARM install folder.

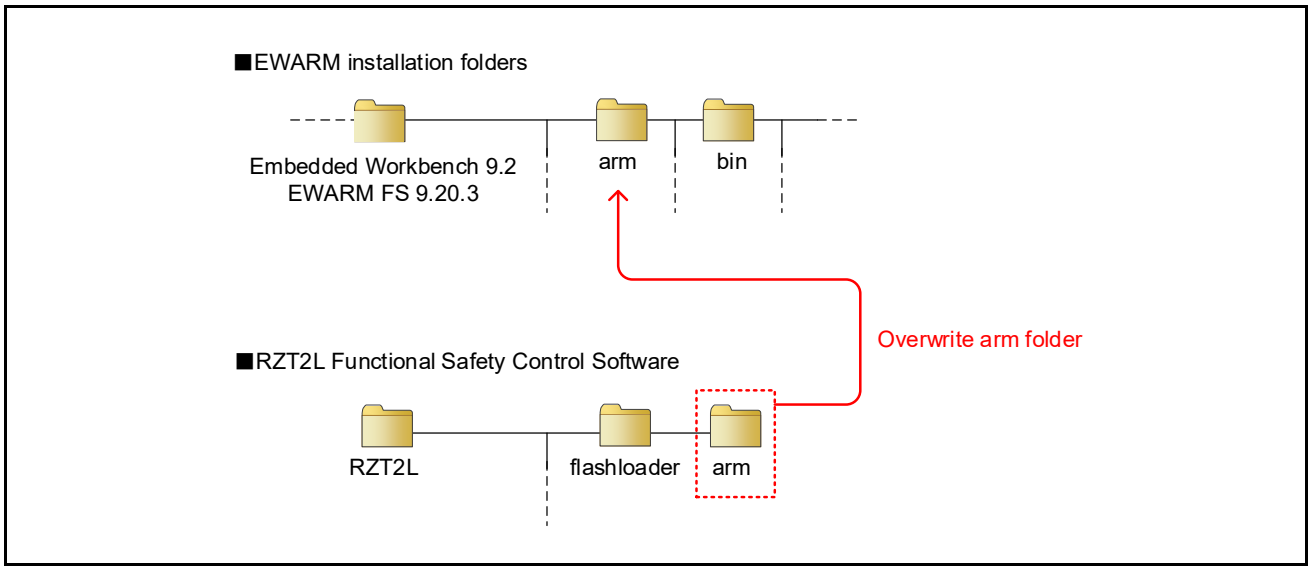


Figure 5.4 Adding Loader File to EWARM

2) Build the RZ/T2L functional safety control software project first.
3) Close the workspace for RZ/T2L functional safety control software, and open the workspace for loader program.

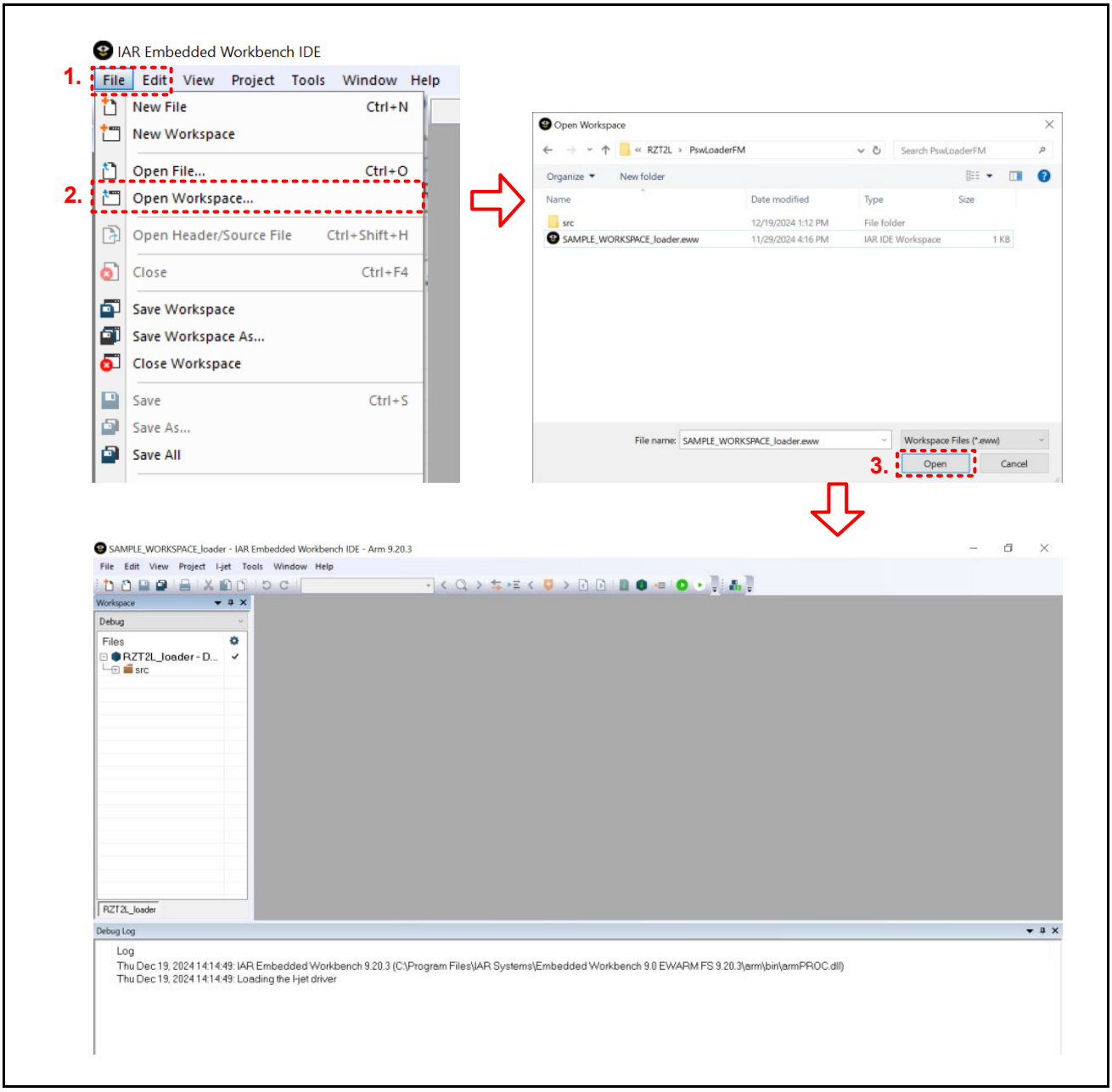


Figure 5.5 Opening Workspace for Loader Program

4) Build the loader program.

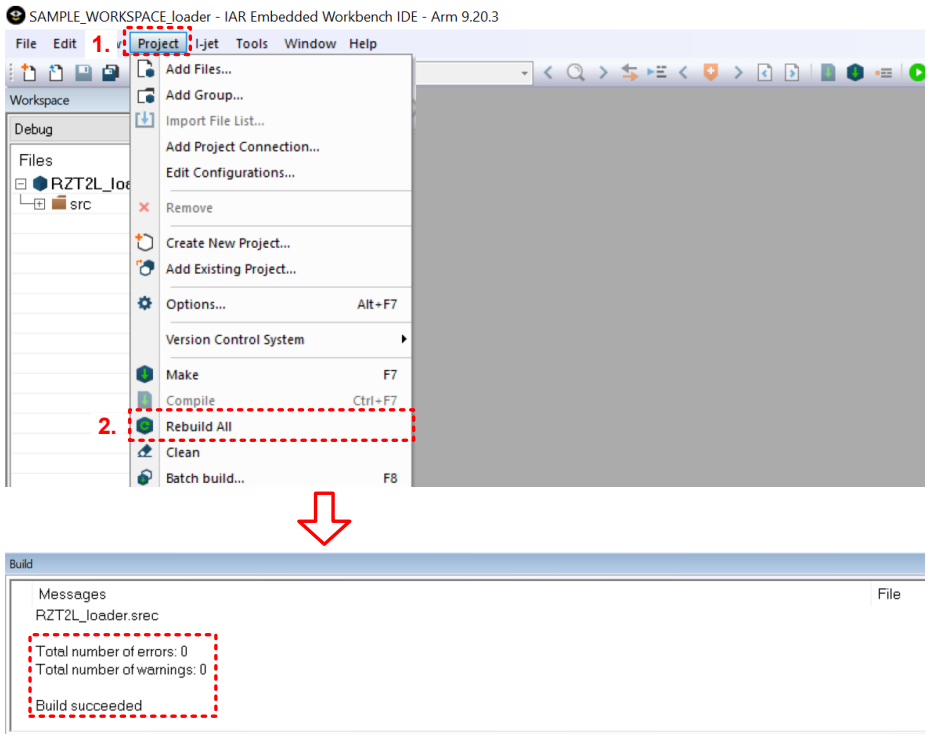


Figure 5.6 Building Loader Program

5) Start downloading program to the flash memory.

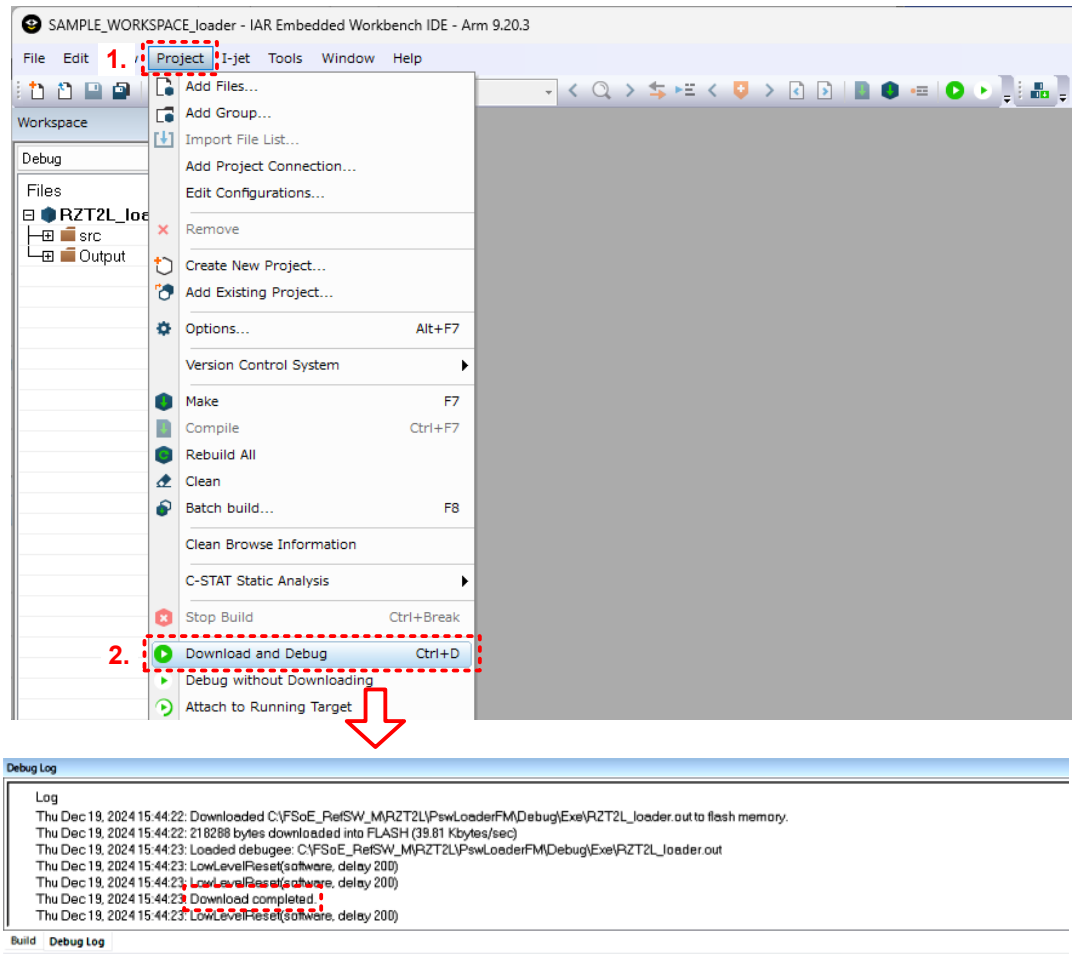


Figure 5.7 Downloading to Flash Memory

ROM writing operation is complete now.

6. How to Operate Program

6.1 Connecting to TwinCAT

6.1.1 Preparing the ESI File (First Time Only)

(1) Before starting TwinCAT, you need to copy the ESI file for this sample to a specific folder.

Source ESI file (located in Common\ESI folder):

- “Renesas EtherCAT RZT2 SafetyMotorSolution.xml”

Destination folder (may vary depending on your environment):

- C:\TwinCAT\3.1\Config\Io\EtherCAT

6.1.2 Installing the TwinCAT Driver (First Time Only)

(1) Launch TwinCAT 3.

(2) From the Start Menu, select : [TwinCAT 3] → [Show Realtime Ethernet Compatible Devices…]

(3) Select the Ethernet adapter you will use and install the driver.

(4) If the selected Ethernet adapter moves to [Installed and ready to use devices (realtime capable)], the installation is successful.

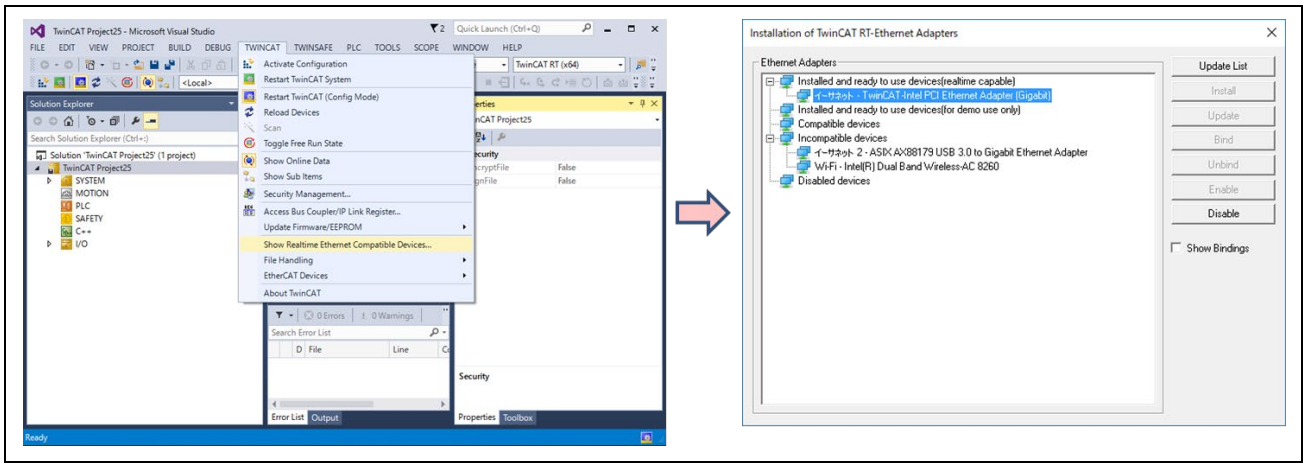


Figure 6.1. : TwinCAT Driver Installation

6.1.3 Scanning Devices and Writing to EEPROM

(1) Create a new project in TwinCAT.

(2) In Solution Explorer → I/O → Devices, select “Scan”.

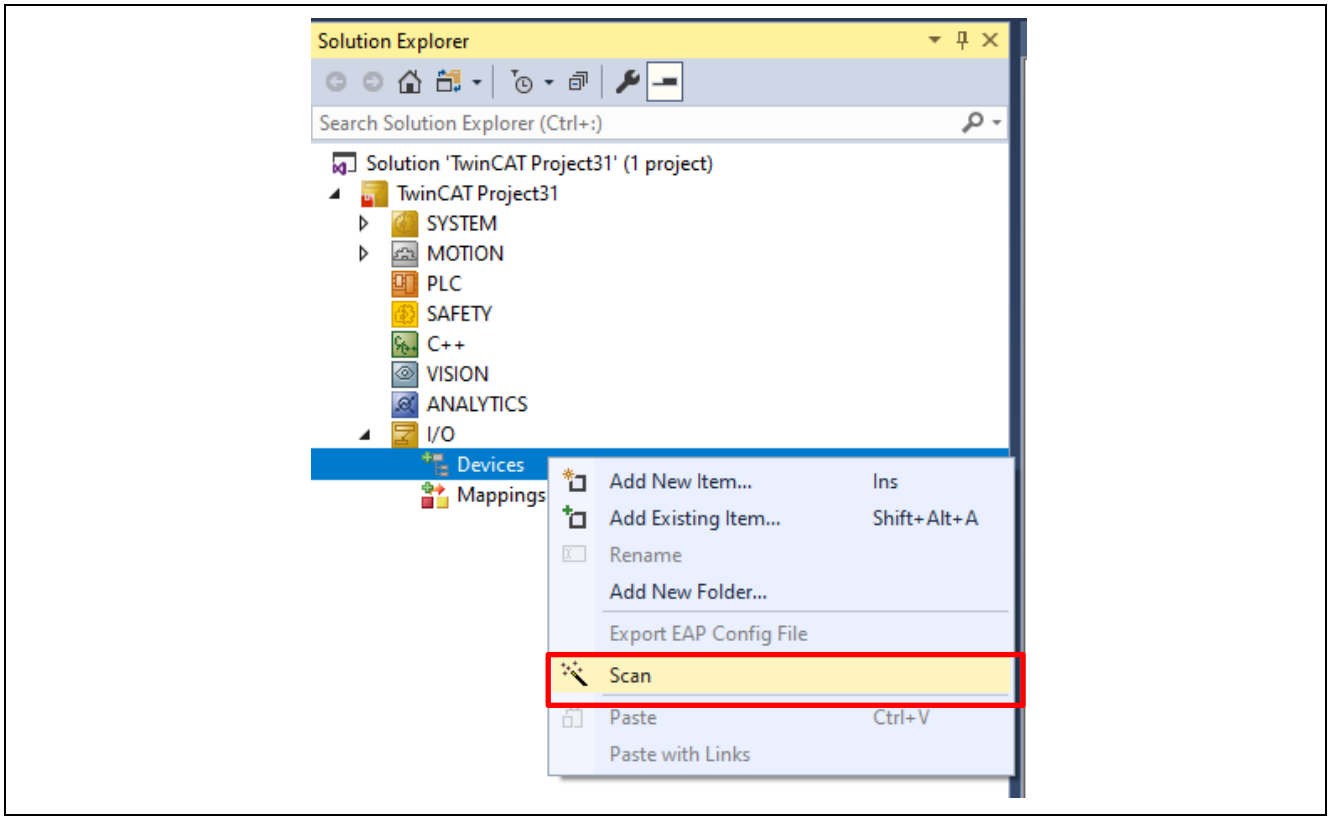


Figure 6.2. Device Scan

(3) If the RZ/T2M is recognized as an EtherCAT device, the EtherCAT port will be displayed as shown below.

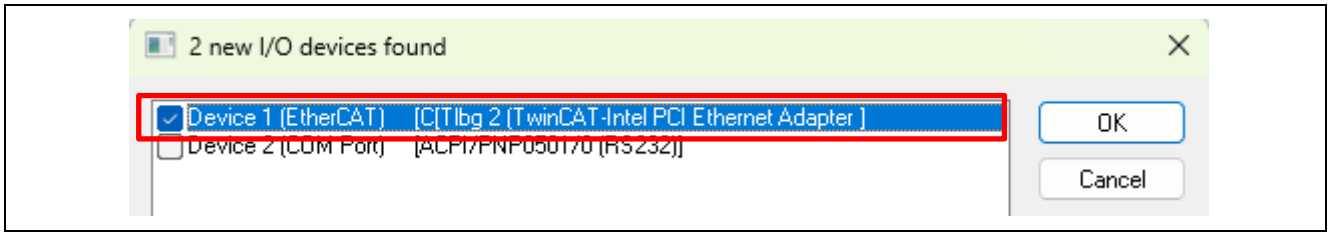


Figure 6.3. Scan Result

(4) Execute [Scan for boxes].

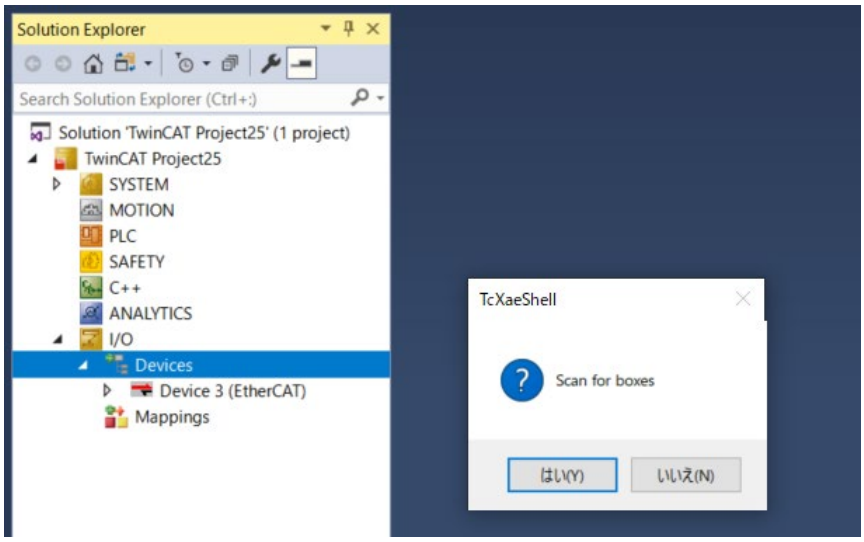


Figure 6.4. Scan for boxes Window

(5) When prompted for [Activate free run], click OK.

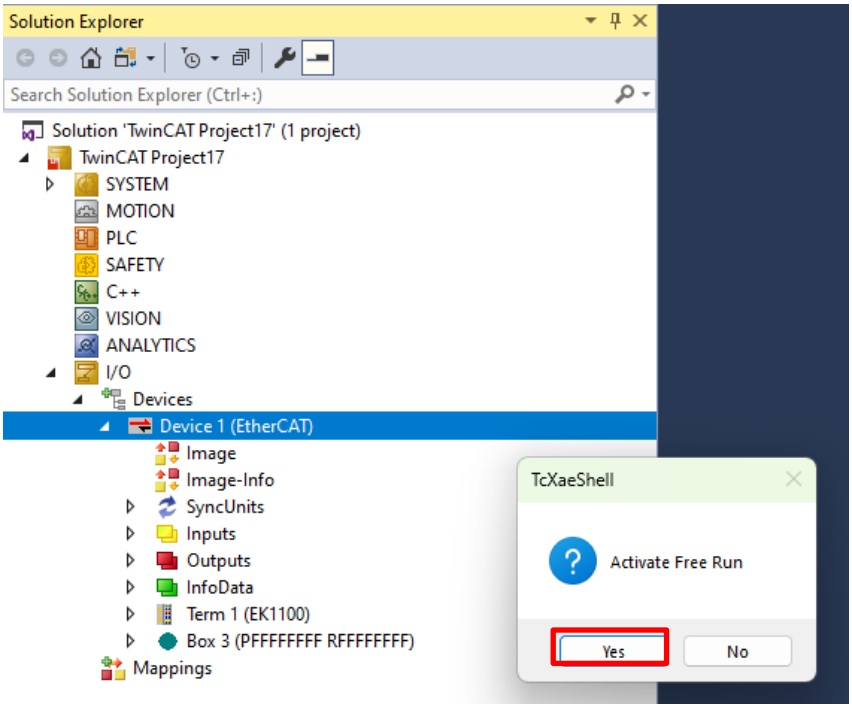


Figure 6.5. Activate Free Run

(6) If Box x (PFFFFF..) is displayed, the Safety Control Board's EEPROM is empty. You need to write the EtherCAT SubDevice information for this sample to the EEPROM.

(7) Steps to write to EEPROM:

- Double-click [Box X] to open the settings screen.
- Select the [EtherCAT] tab.
- Click [Advanced Setting].

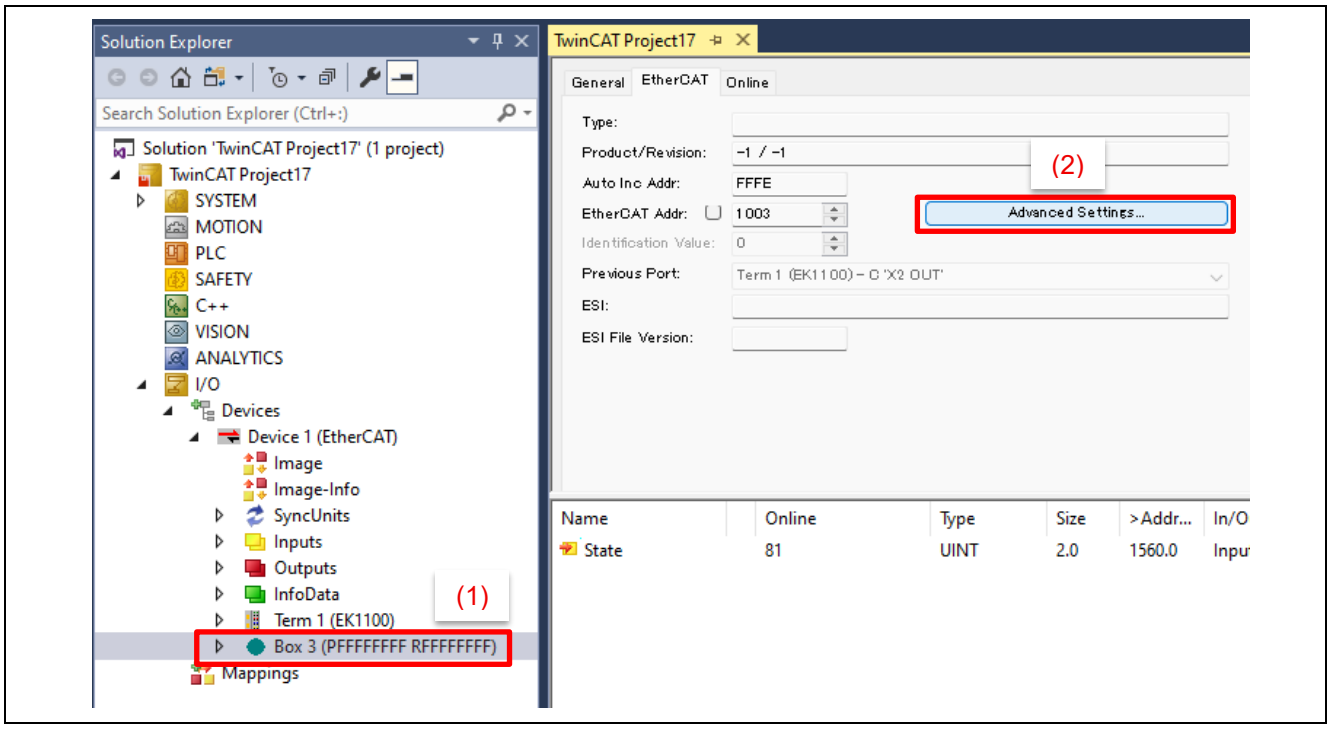


Figure 6.6. EEPROM Write Method 1

- Navigate to [ESC Access] → [EEPROM] → [Hex Editor].
- Select [Download from List].

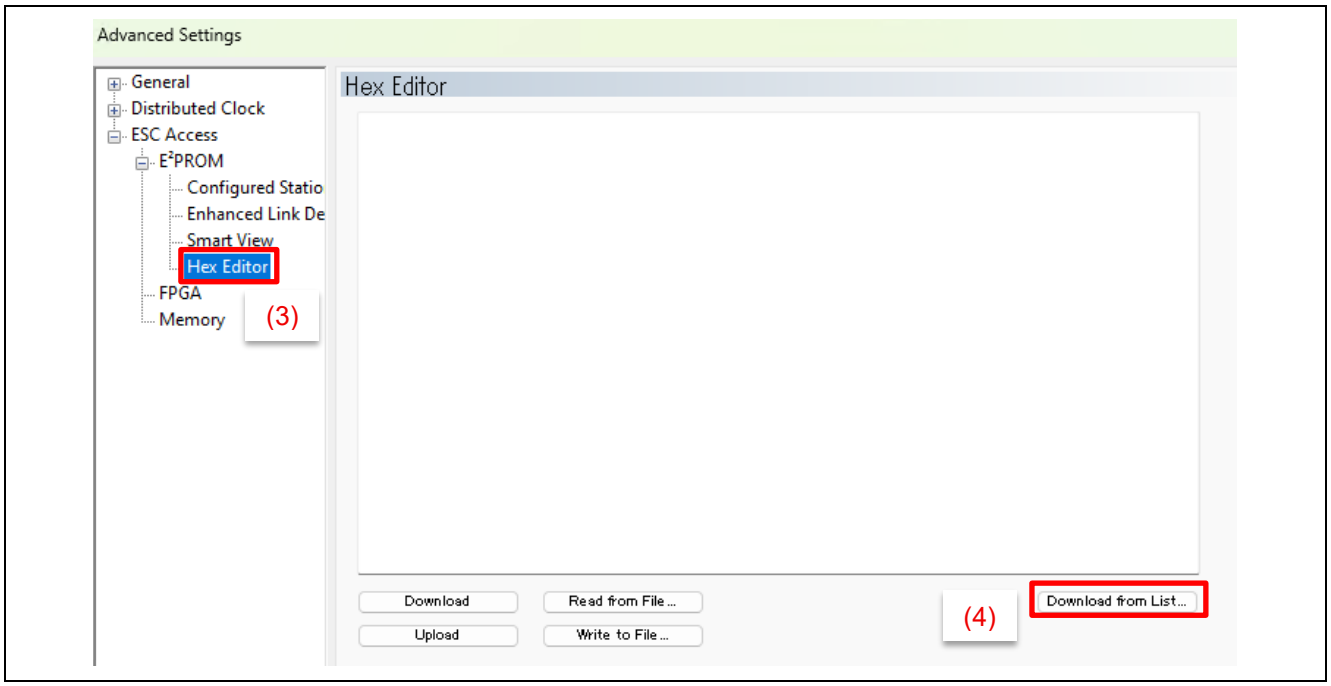


Figure 6.7. EEPROM Write Method 2

- A list of ESI files registered in TwinCAT 3 will appear. Select the appropriate file. For this sample, choose : [Renesas EtherCAT RZ/T2 SafetyMotorSolution].

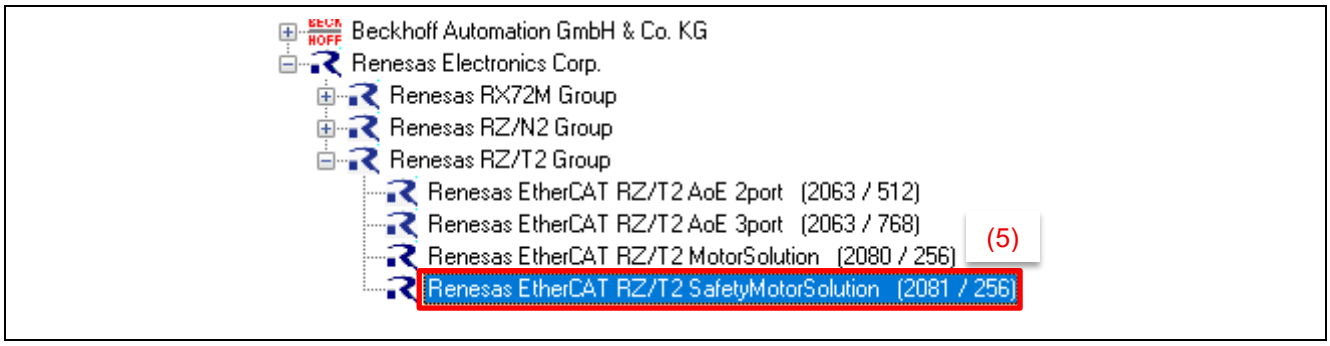


Figure 6.8. EEPROM Write Method 3

(8) After writing to EEPROM, press SW2 on the board to restart the RZ/T2M and reload the EEPROM.

(9) In TwinCAT, delete [Device x (EtherCAT)] and perform a rescan.

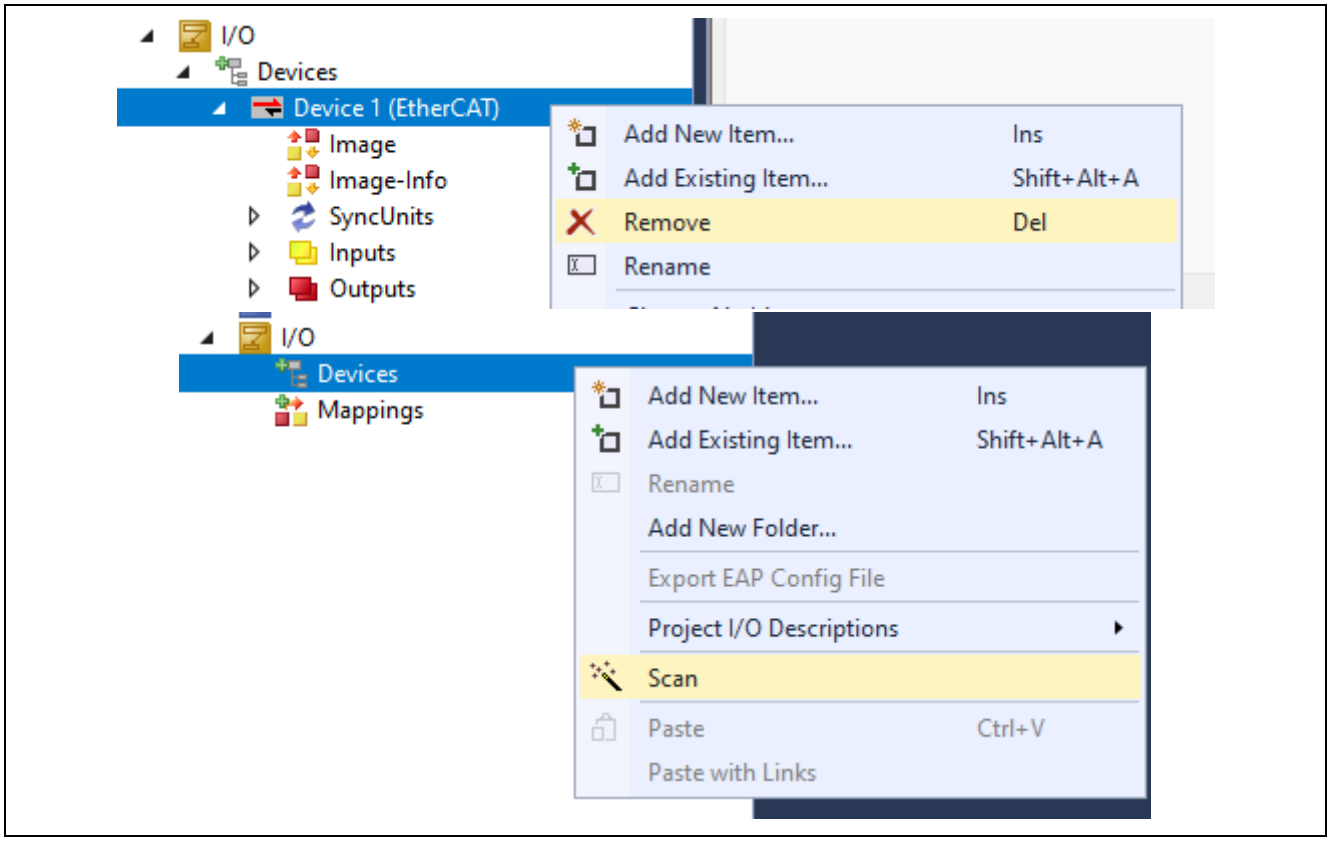


Figure 6.9. Rescanning Devices

(10) If the scan is successful, the following display will appear.

Since the EtherCAT drive is not used for operation verification, select [Cancel].

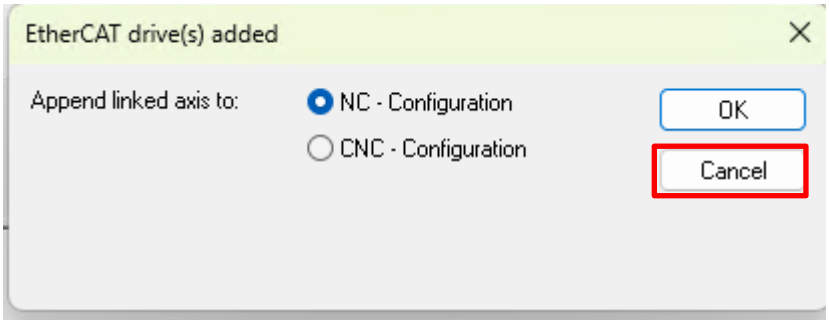


Figure 6.10. EtherCAT Drive Selection

(11) Execute [Reload Devices] to restart the network.

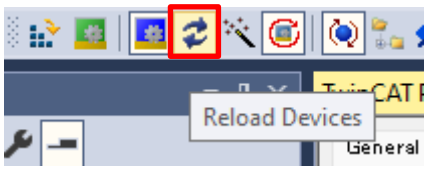


Figure 6.11. Network Restart

(12) Confirm that the device appears in the tree as shown below and that the Current State in the [Online] tab is “OP”.

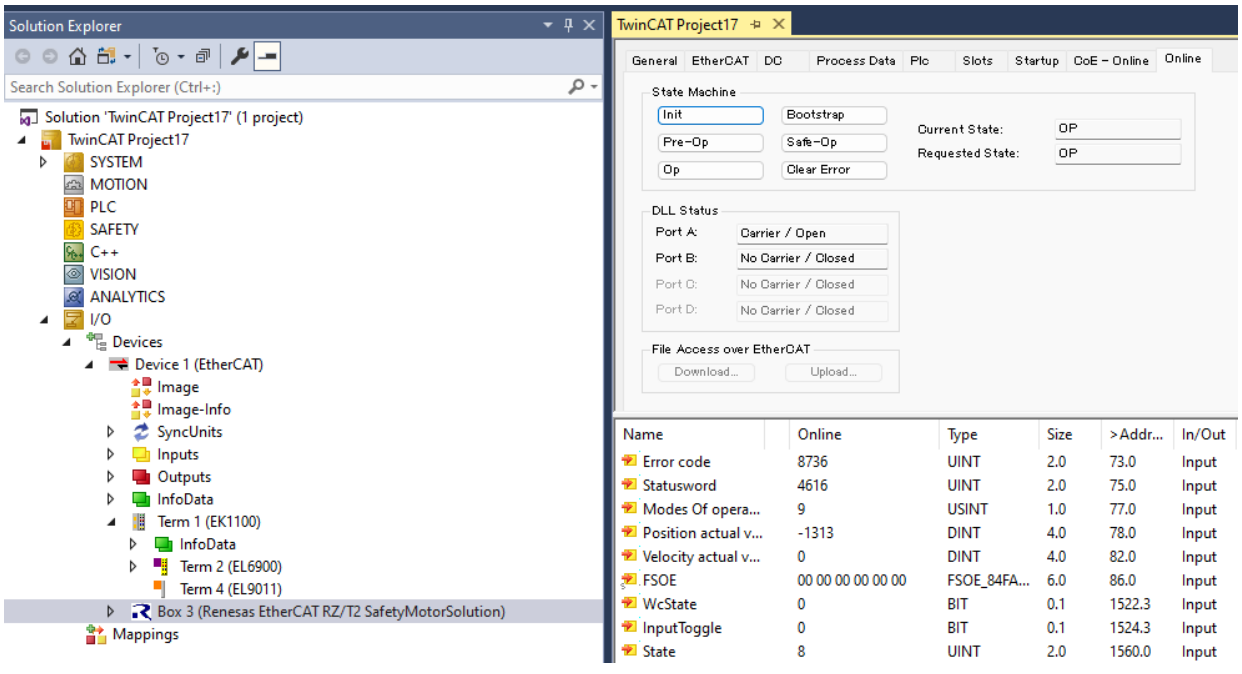


Figure 6.12. Confirming “OP” State

6.2 Preparing the FSoE Master

This section explains how to create the FSoE master program. There is no specific device required for the FSoE master, but this document uses the following products from Beckhoff Automation as an example:

- EK1100 : EtherCAT Coupler
- EL6900 : EtherCAT Terminal communication interface, TwinSAFE Logic
- EL9011 : Bus end cover for E-bus contacts

Note:

Perform this procedure after completing “6.1.3 Scanning Devices and Writing to EEPROM” and ensuring that the Box has been added to the System Manager tree.

In Section 6.2, add the Safety project and PLC project created with TwinCAT version 4026.19.
If your TwinCAT version is older, the projects may not be added successfully.
If you cannot add them, refer to the Appendix: Procedure for Creating the FSoE Master Program.

6.2.1 Adding a Safety Project

The following describes the steps to add a Safety project.

1) In TwinCAT3, select [Solution Explorer] > [Safety_Drive] > [SAFETY] > [Add Existing Item...]
2) Select Safety_Drive.tfzip file (Table 7.66, #13)
3) Click [Open] to add the Safety project.

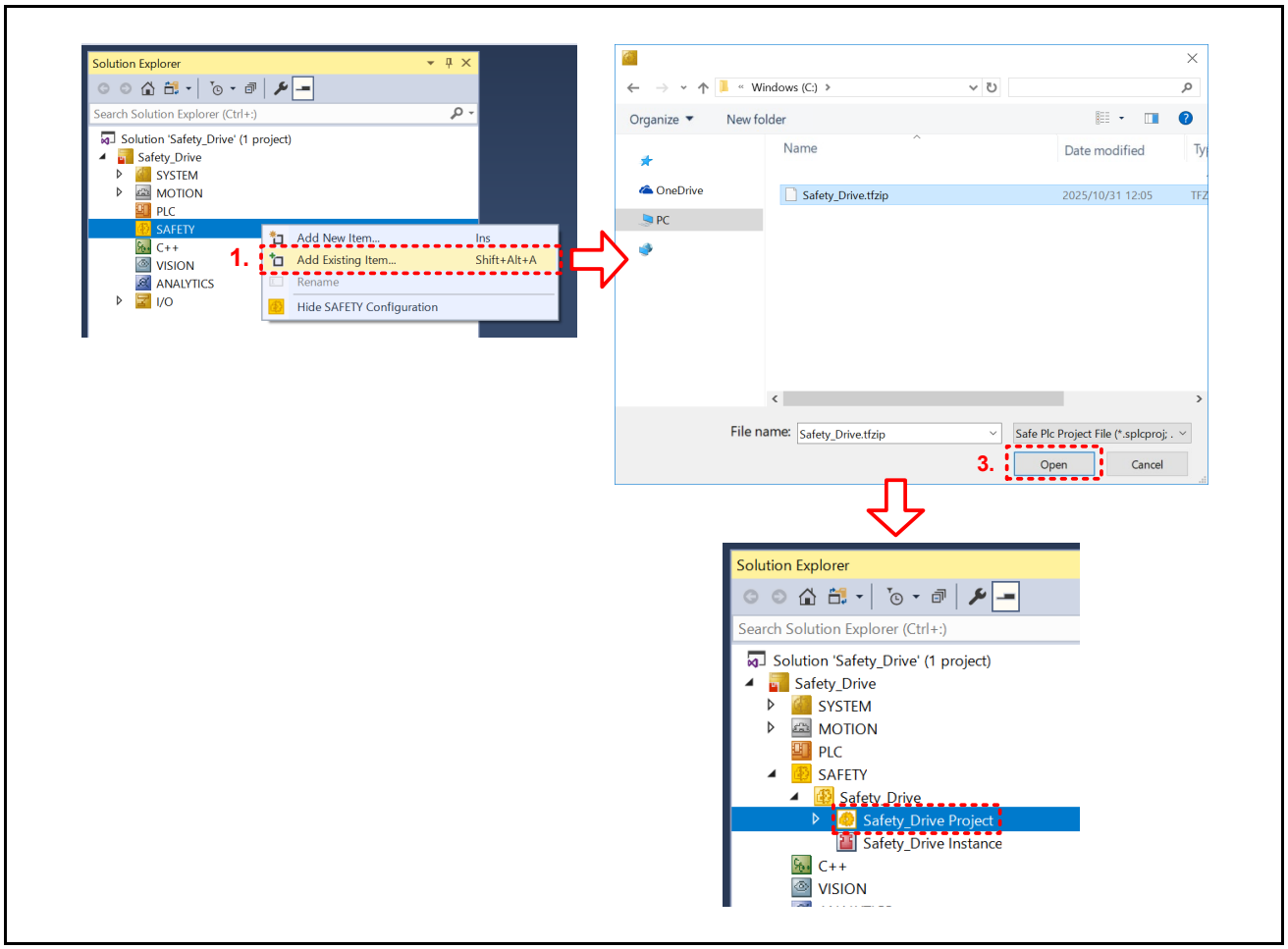


Figure 6.13. Adding Safety Project

6.2.2 Adding a PLC Project

The following describes the procedure for adding a PLC project.

1) In TwinCAT3, select [Solution Explorer] > [Safety_Drive] > [PLC] > [Add Existing Item...]
2) Select Safety_Drive.tpzip file (Table 7.66, #14)
3) Click [Open] to load the PLC project.

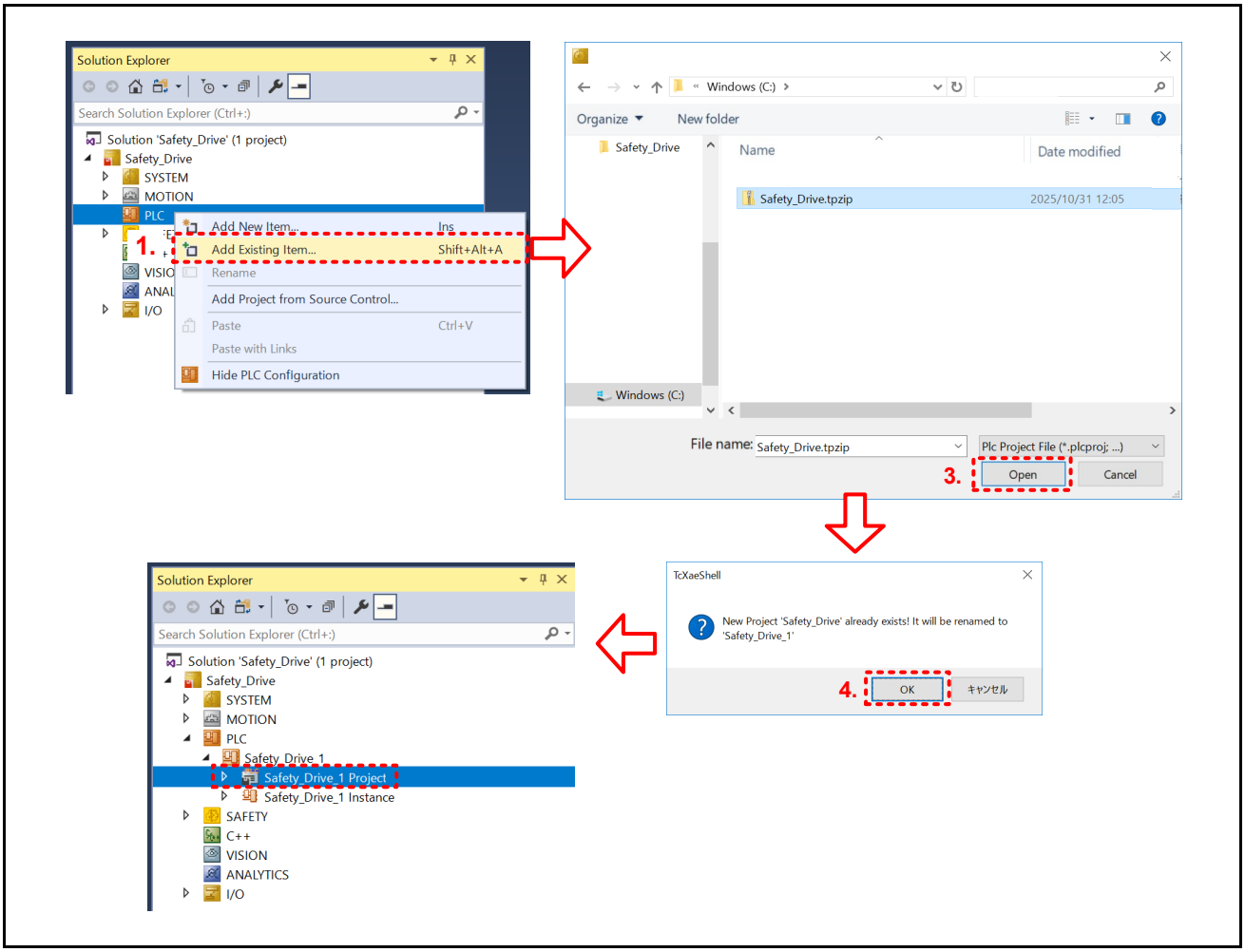


Figure 6.14. Adding PLC Project

Adding PLC project is complete now

6.2.3 Selecting the ErrAck Signal

1. In the TwinCAT 3 System Manager tree, navigate to:[SAFETY] → [safety_project_1] → [safety_project_1 project] → [TwinsafeGroup1] → [Alias Devices], then double-click [ErrorAcknowledgement.sds].
2. Click the icon next to [Full Name].
3. Select MAIN.bErrAck and click [OK].

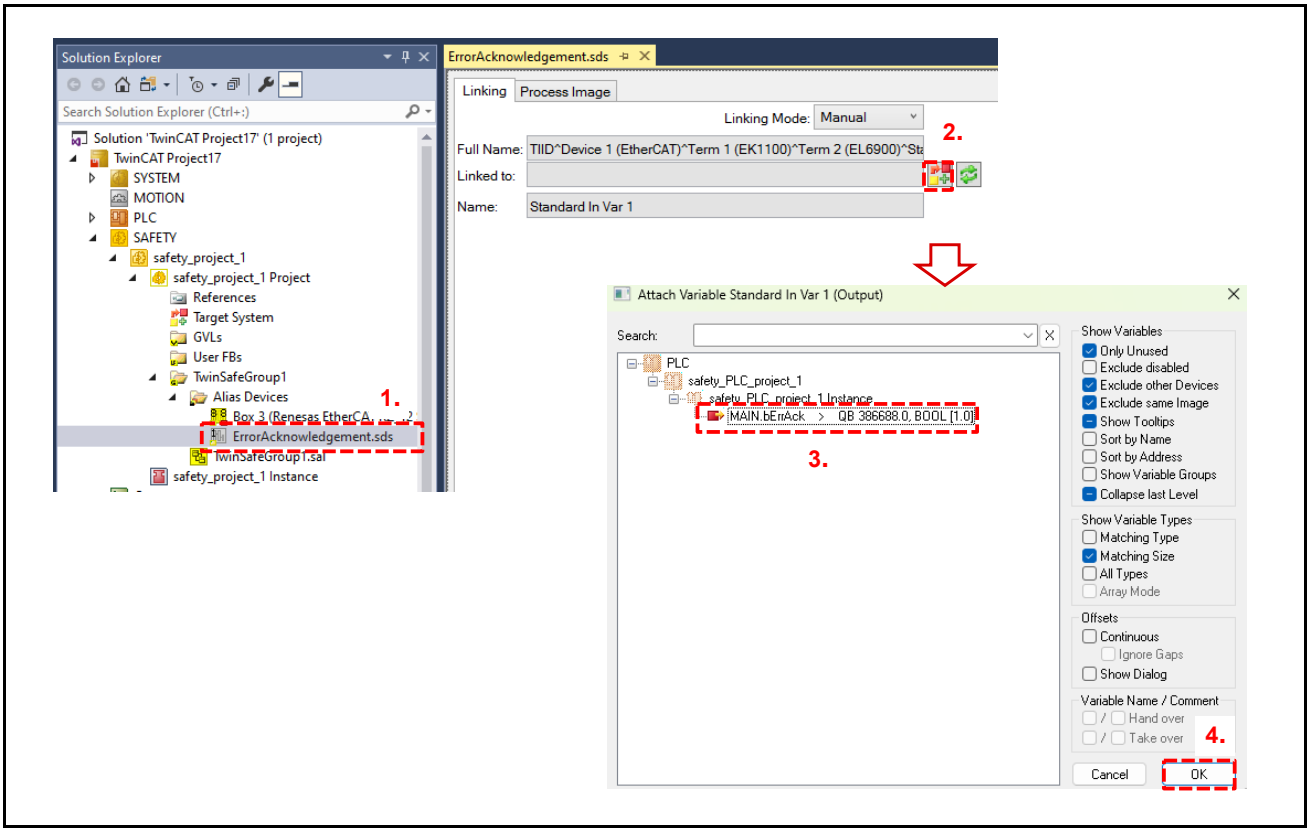


Figure 6.15. Selecting ErrAck Signal

6.2.4 Checking the FSoE Master Device

1. In the TwinCAT 3 System Manager tree, navigate to [SAFETY] → [Safety_Drive], then right-click [Safety_Drive Project].
2. Select [Properties].
3. Click the [Target System] tab and confirm that the correct FSoE master device is selected (in the example, “EL6900”).
4. The device’s Serial Number will be required in section 6.3.
5. Confirm that [Safe Address] and [Hardware Address] match. If they differ, click the icons in order (a) → (b) to synchronize the addresses.

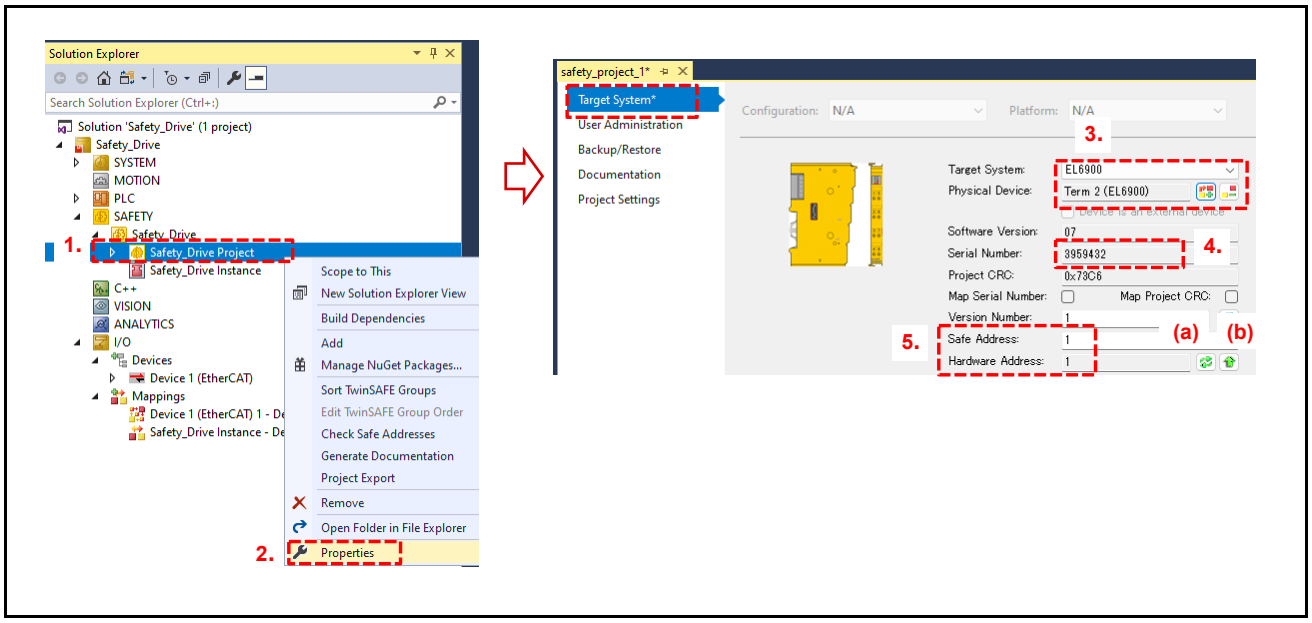


Figure 6.16. Checking FSoE Master Device

6.3 Downloading the FSoE Master Project

(1) Verify the Program Before Downloading Before downloading, verify that the created program has no issues:

1. From the [TWINSAFE] tab, select [Verify Safety Project].
2. If there are no issues, the message [Verification Process succeeded] will appear.
3. Next, select [Verify Complete Safety Project].
4. If there are no issues, the message [Verification Process succeeded] will appear again.

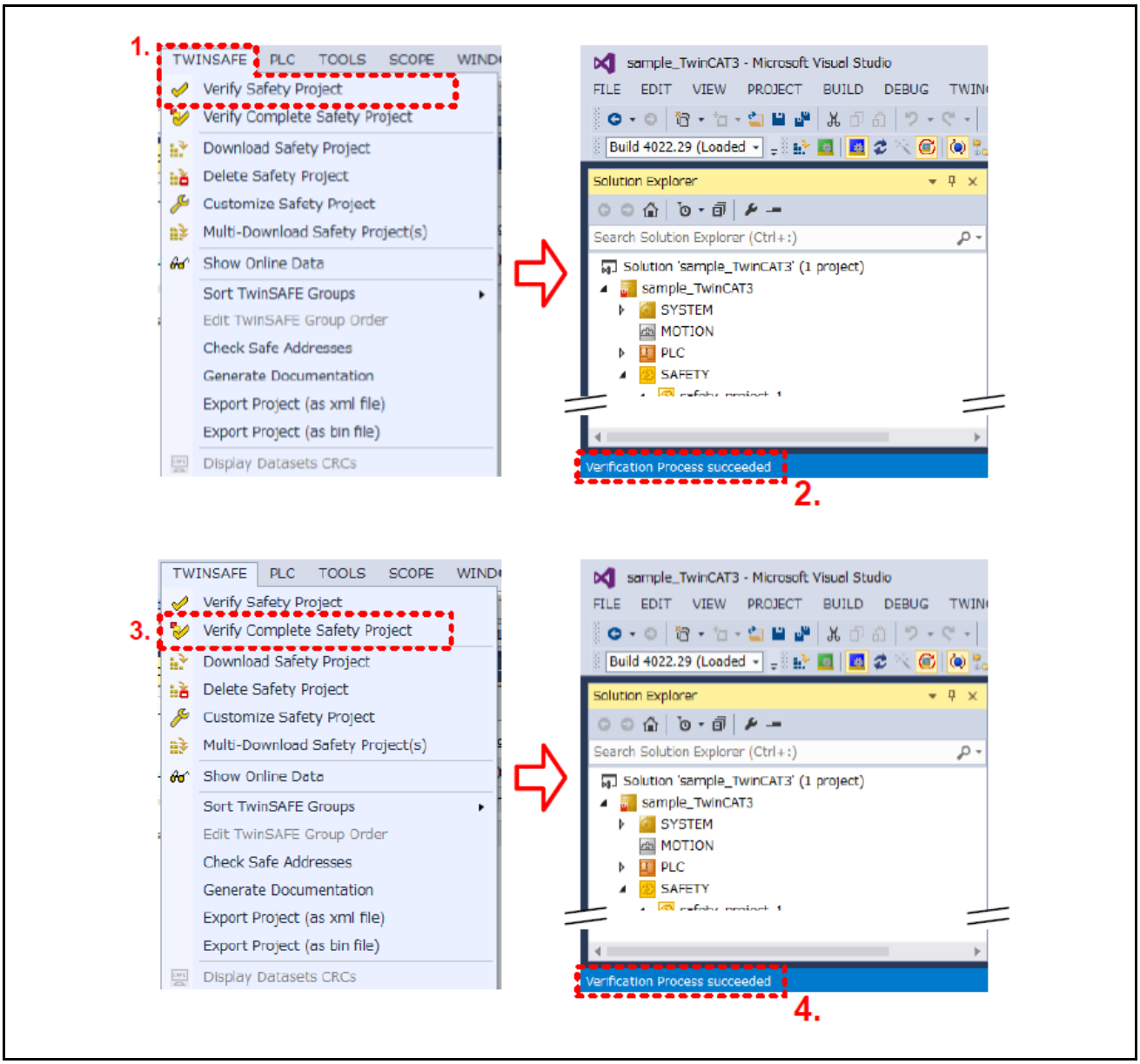


Figure 6.17. Verifying the FSoE Master Program

Note:

If [Verification Process failed] appears, check that there are no duplicate FSoE slave addresses on the same network and review the settings in Section 6.2.

(2) Download the Program to the FSoE Master Device

1. From the [TWINSAFE] tab, click [Download Safety Project].
2. When the confirmation screen appears, click [Yes] to start the download.

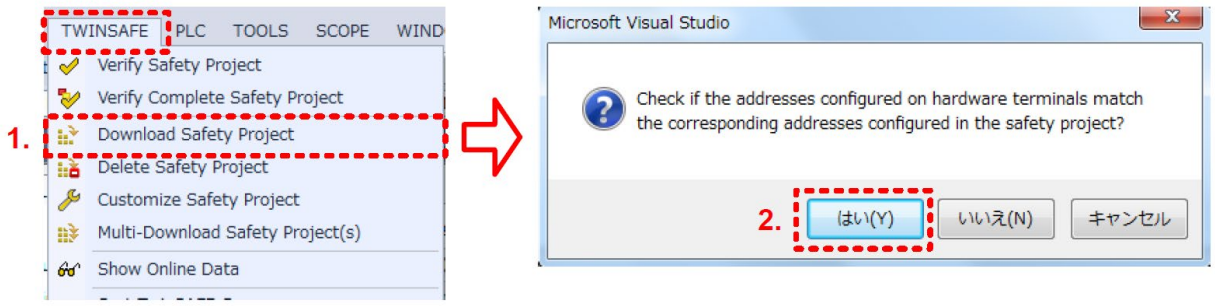


Figure 6.18. Starting FSoE Master Program Download

(3) Confirm the Download Result

1. In the [Login] tab, enter the following values and click [Next]:
   - [Username]: Administrator
   - [Serial Number]: The serial number confirmed in Section 6.2
   - [Password]: TwinSAFE
2. In the [Download Result] tab, confirm that all items have succeeded, then click [Next].
3. In the [Final Verification] tab, confirm that there are no issues with the final data, check the box for “I have manually ~”, and click [Next].
4. In the [Activation] tab, enter the [Password] again and click [Finish].

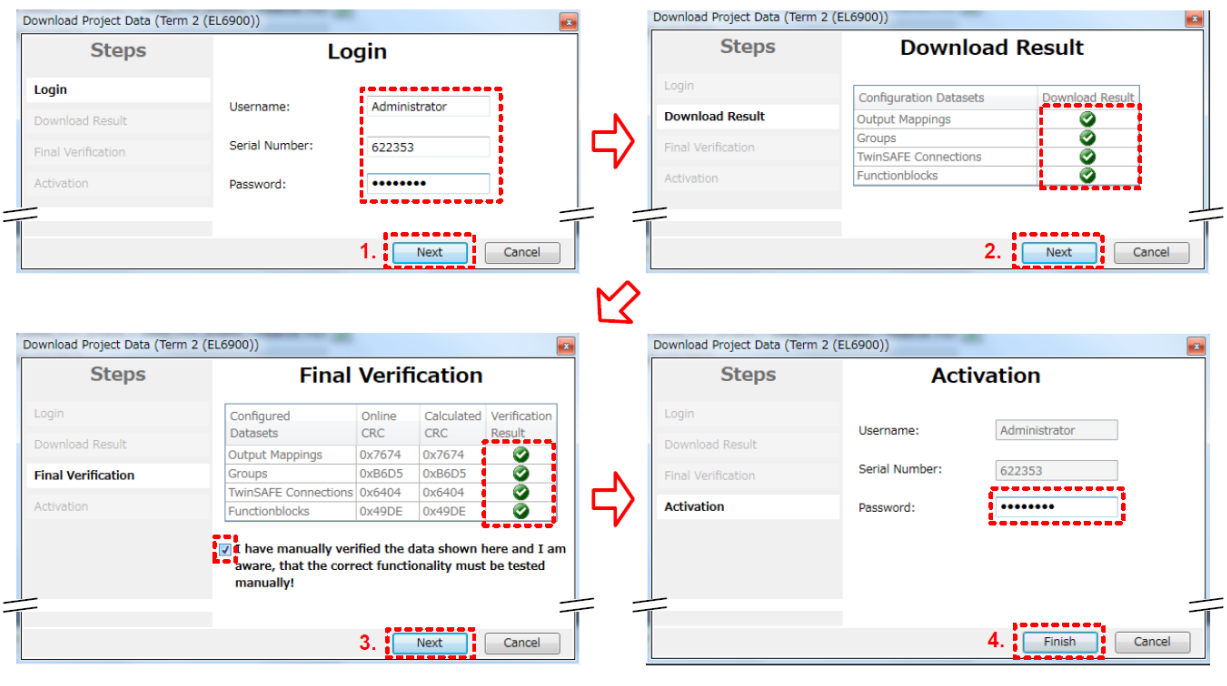


Figure 6.19. Downloading the FSoE Master Program

6.4 How to Run the TwinCAT Project

First, power on the Safety Motor Control Reference Kit and ensure that the program written to the microcontroller on the board is running.

The following steps describe how to execute the TwinCAT project.

(1) Set the EtherCAT Communication Cycle. This example shows how to set the EtherCAT communication cycle to 250 µs:

1. In the TwinCAT 3 System Manager tree, click [SYSTEM] → [Real-Time] to open the properties window.
2. Click the [Settings] tab.
3. From the [Base Time] dropdown menu, select “250 µs”.
4. In the System Manager tree, click [SYSTEM] → [Tasks] → [PlcTask] to open the properties window.
5. Click the [Task] tab.
6. In the left-side setting menu for [Cycle ticks], set the value to “1” using the spin button.
7. Confirm that the right-side setting menu for [Cycle ticks] shows “0.250 ms”.

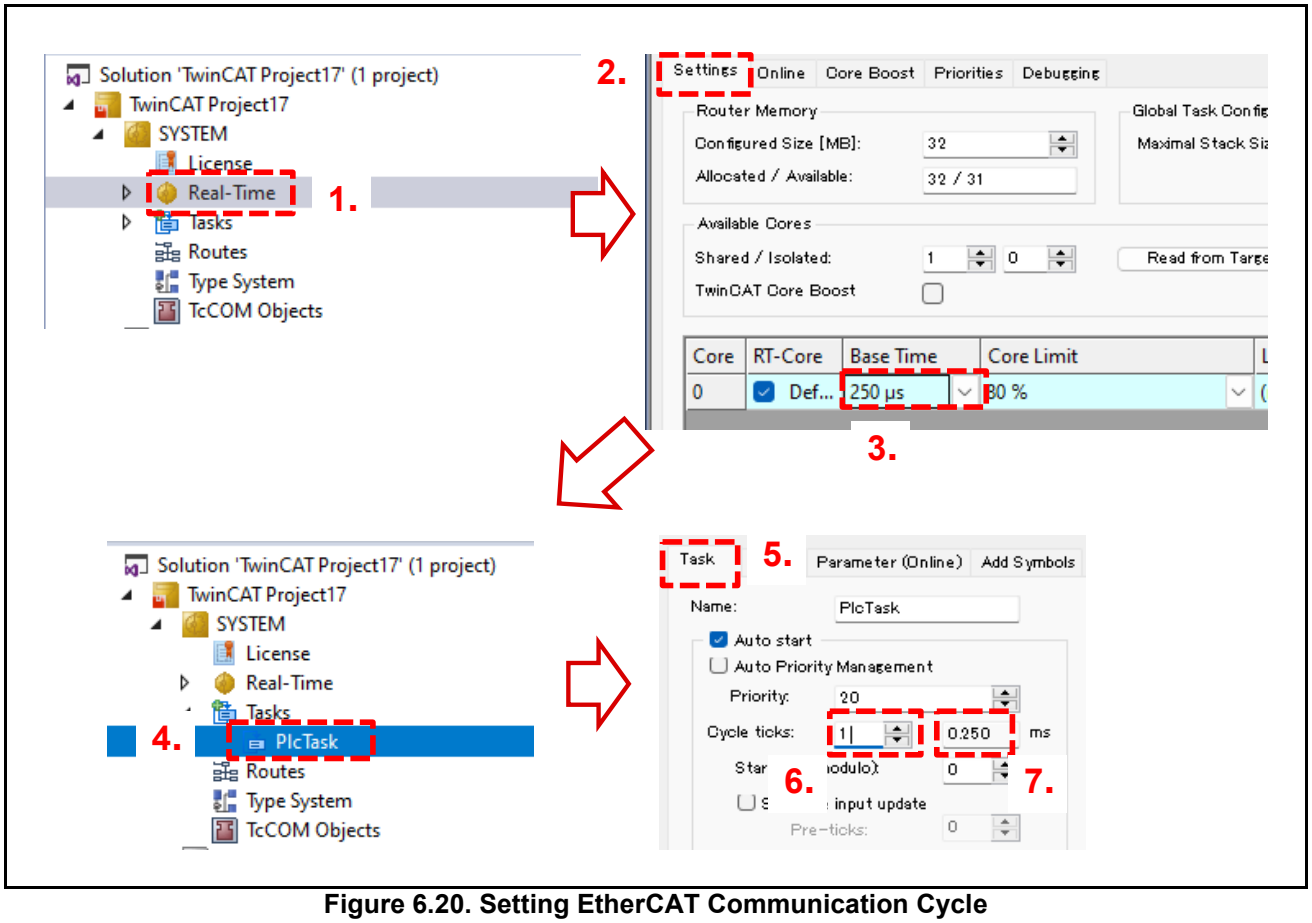


Figure 6.20. Setting EtherCAT Communication Cycle

(2) Set EtherCAT Synchronization Mode to DC Synchronization

1. In the System Manager tree, click [Box X (Renesas EtherCAT RZ/T2 SafetyMotorSolution)] to open the properties window.
2. Click the [DC] tab.
3. Under Operation Mode, select [DC-Synchron].
4. Click [Advanced Settings...].
5. Check [Enable SYNC1].
6. Set the timing for SYNC1 interrupt in [Shift Time (µs):].
7. Click [OK].

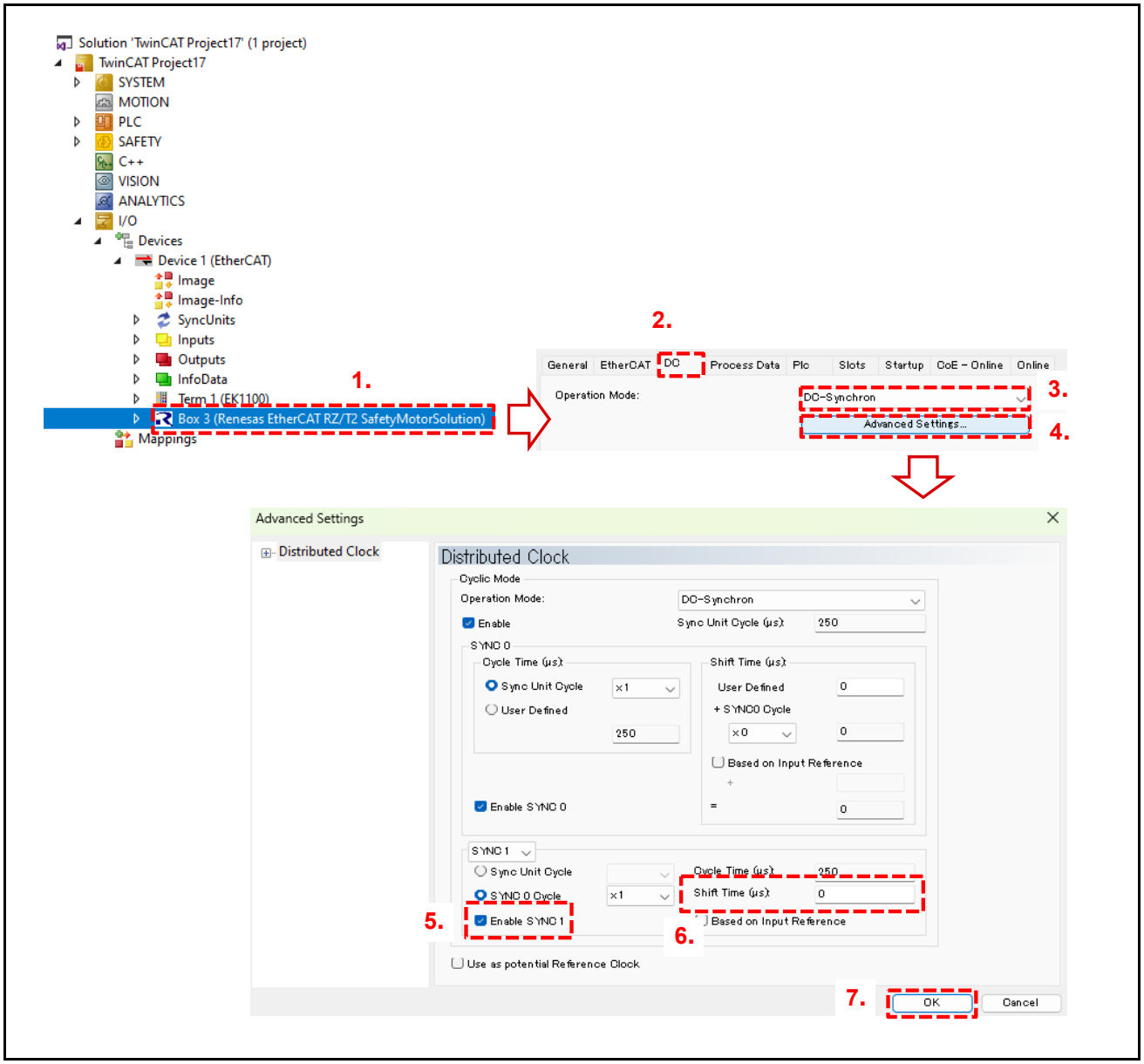


Figure 6.21. DC Synchronization Settings Part 1

8. In the System Manager tree, click [I/O] → [Devices] → [Device X (EtherCAT)] to open the properties window.
9. Click the [EtherCAT] tab.
10. Click [Advanced Settings...] to open the Advanced Settings window.
11. In the Advanced Settings window, click [Distributed Clocks].
12. Set the timing for SYNC0 interrupt under Percent of cycle time.
13. Click [OK].

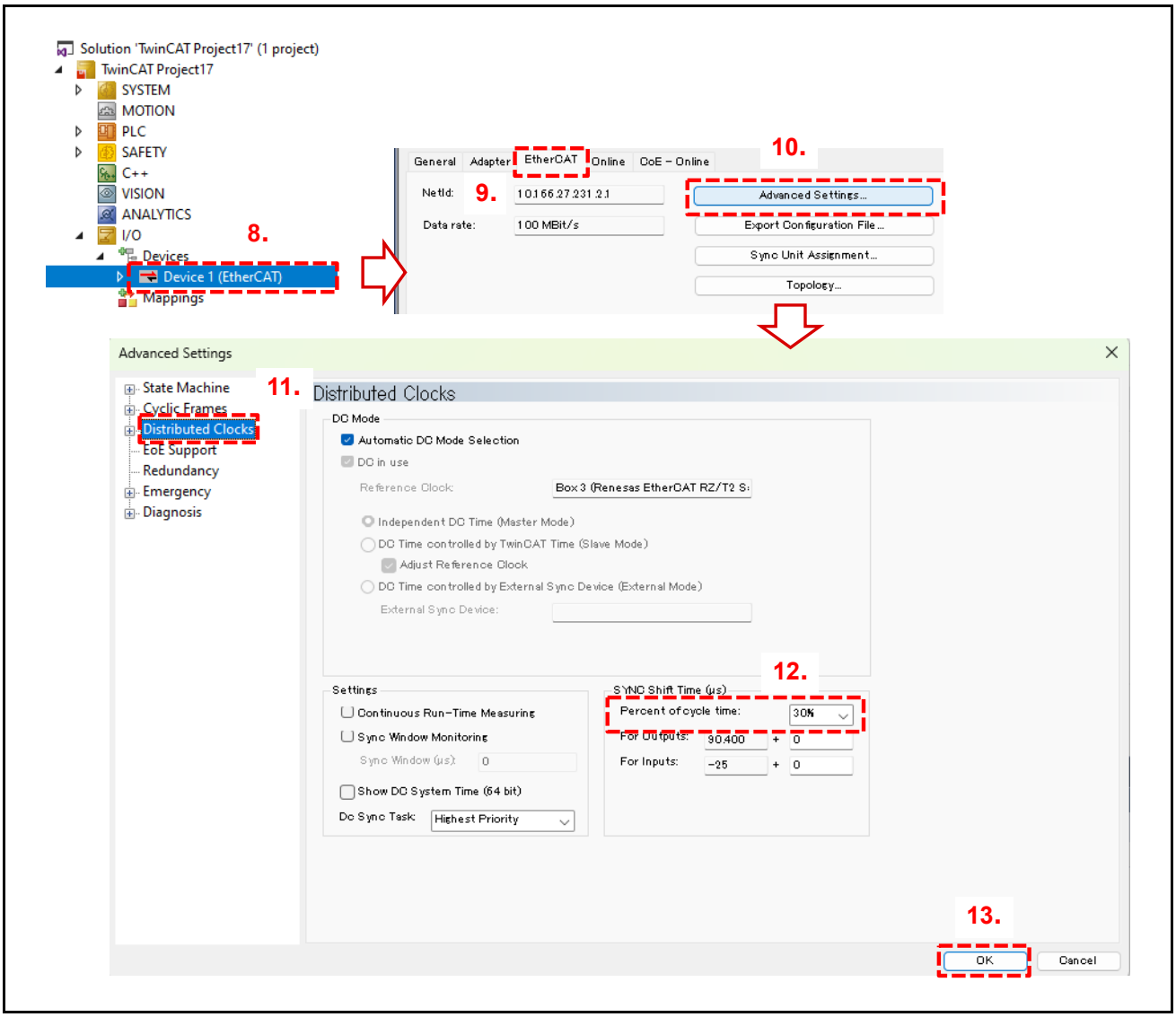


Figure 6.22. DC Synchronization Settings Part 2

(3) Enable EtherCAT Network Settings

1. In the System Manager tree, click [I/O] → [Devices] → [Device X (EtherCAT)] to open the properties window.
2. Click the [EtherCAT] tab.
3. Click [Sync Unit Assignment...] to open the Sync Unit Assignment window.
4. If Task is set to <default>, change it to PlcTask.
5. Click [Apply], then [OK].

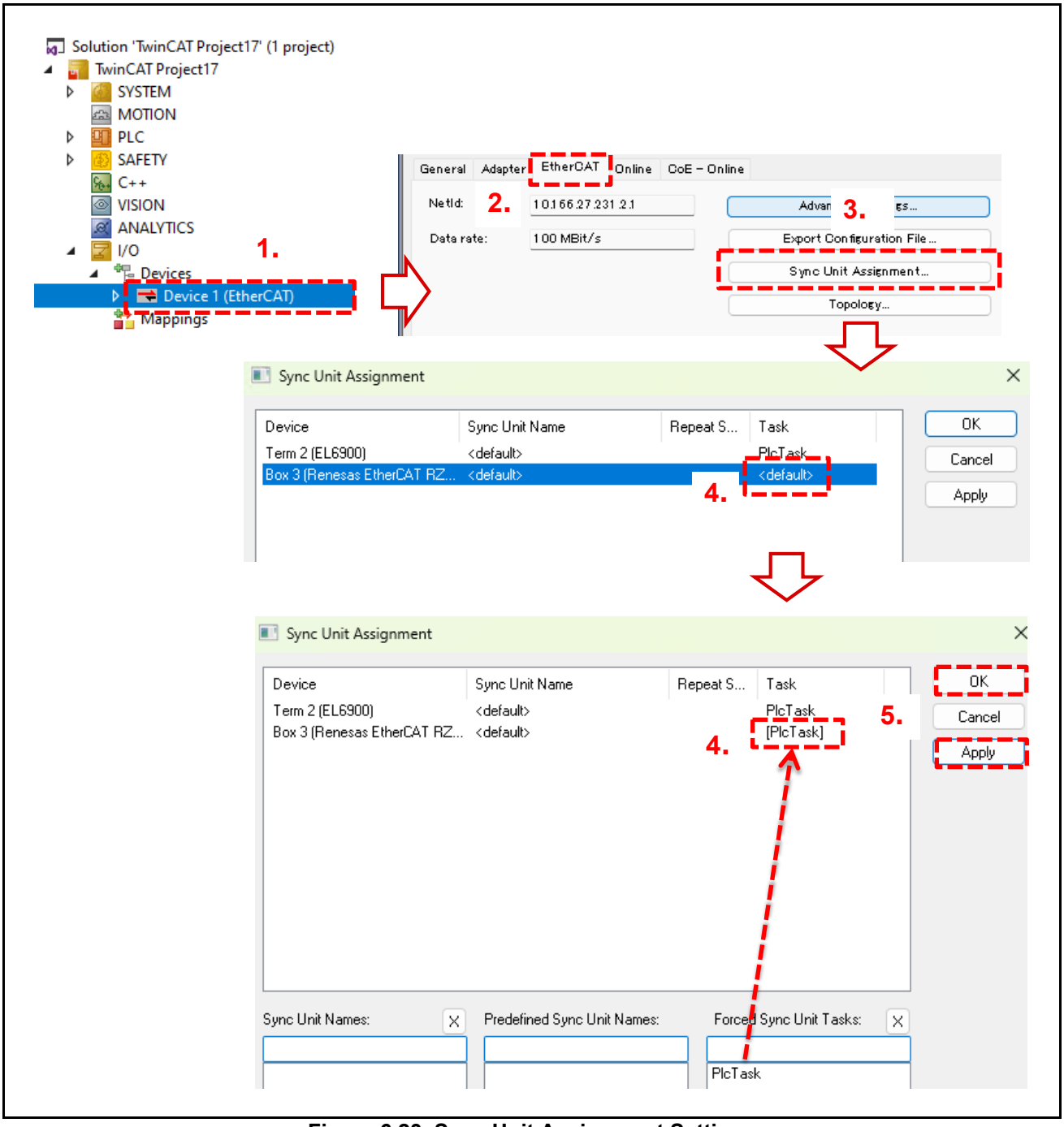


Figure 6.23. Sync Unit Assignment Settings

6. Click the [Activate Configuration] icon in TwinCAT 3.
7. When the Activate Configuration window appears, click [OK].
8. When the Restart TwinCAT System in Run Mode window appears, click [OK] to switch the TwinCAT 3 project to Run Mode.

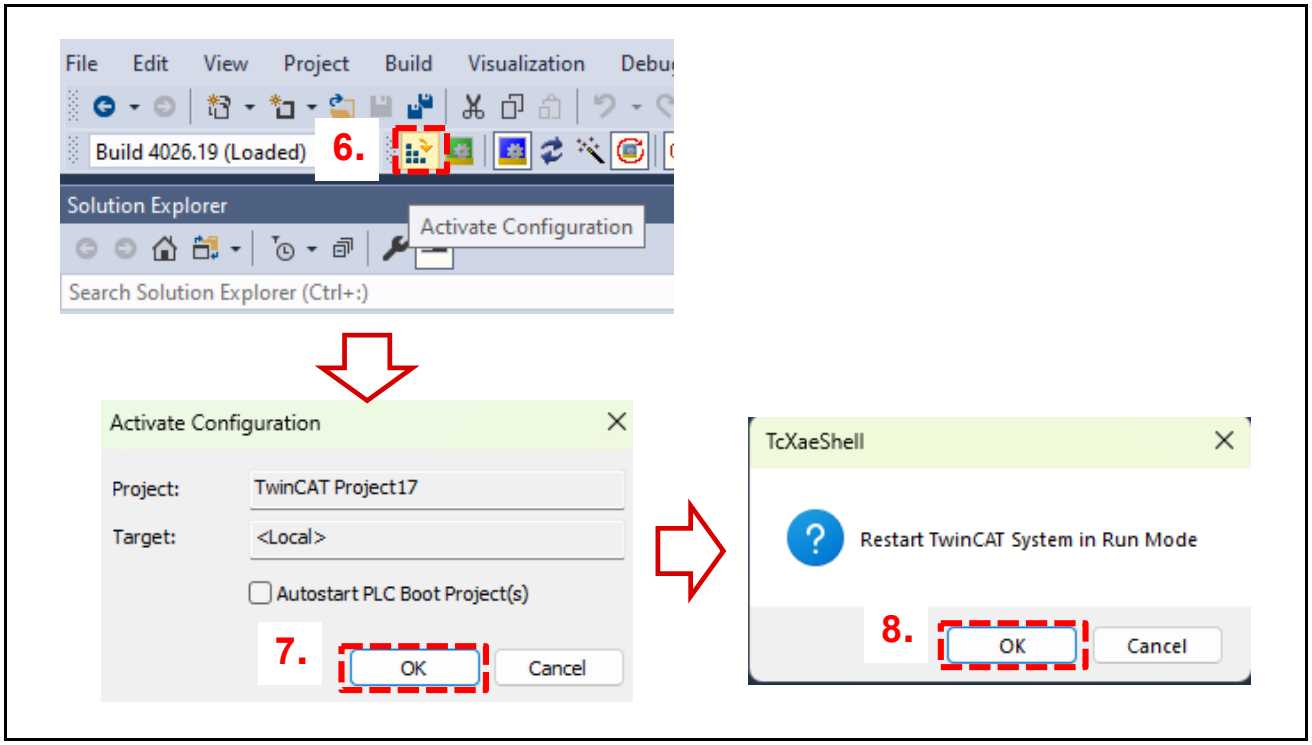


Figure 6.24. Starting Run Mode

(4) Confirm Connection Between TwinCAT 3 and Safety Motor Control Reference Kit

1. In the System Manager tree, double-click [Box X (Renesas EtherCAT RZ/T2 SafetyMotorSolution)] to open the properties window.
2. Click the [Online] tab.
3. If “Current Status” shows [OP], TwinCAT 3 is successfully connected to the Safety Motor Control Reference Kit.

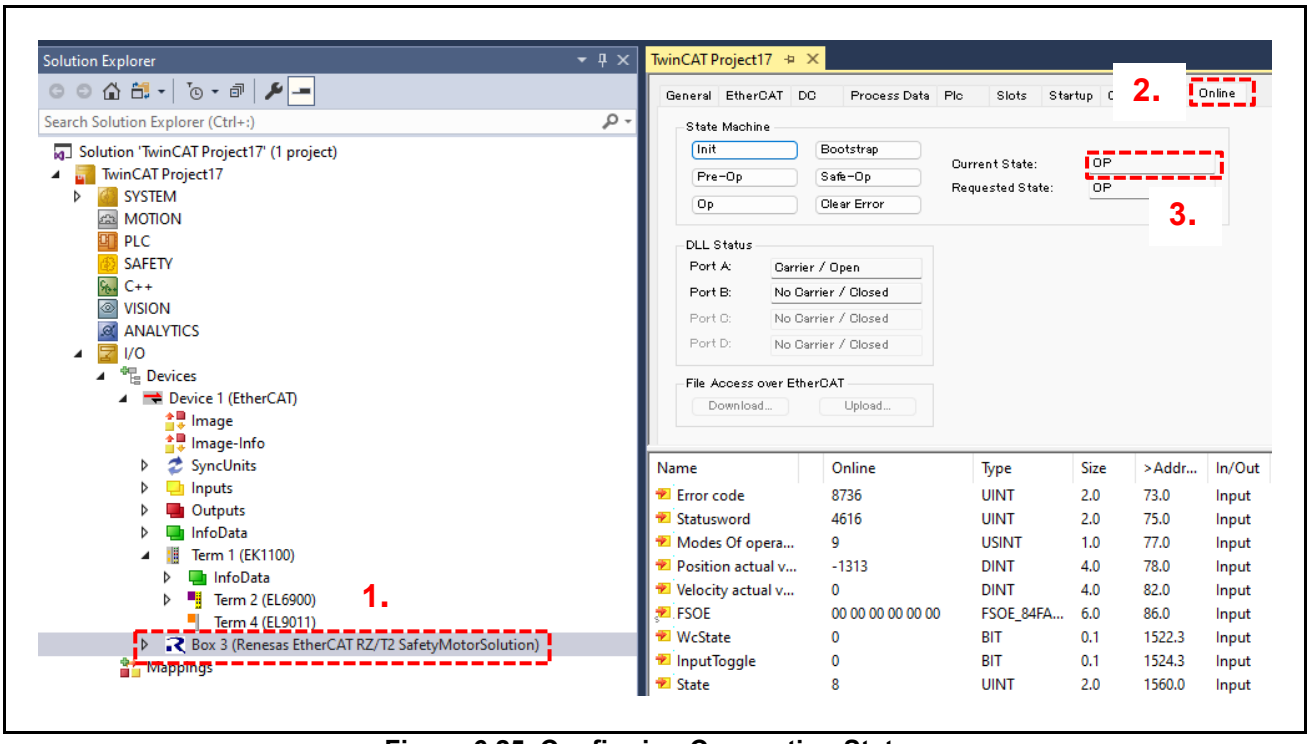


Figure 6.25. Confirming Connection Status

(5) Execute the TwinCAT Project

1. In TwinCAT 3, click the [Login] button to log in to the PLC.
2. When the TwinCAT PLC Control window appears, select [Yes].
3. Click the [Start] button to run the TwinCAT project.

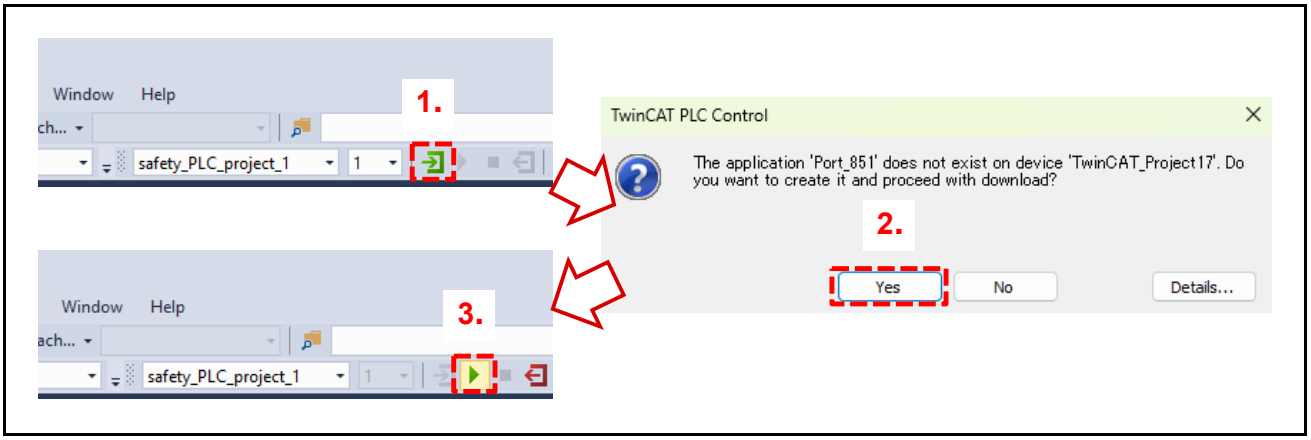


Figure 6.26. Executing the TwinCAT Project

6.5 How to Verify Operation

This section explains the procedure for verifying the operation of the Safety Motor Control Reference Kit.

6.5.1 Selecting Safety Drive Function

(1) When selecting STO function

Start the Safety Motor Control Reference Kit with SW9 and SW11 all set to OFF.

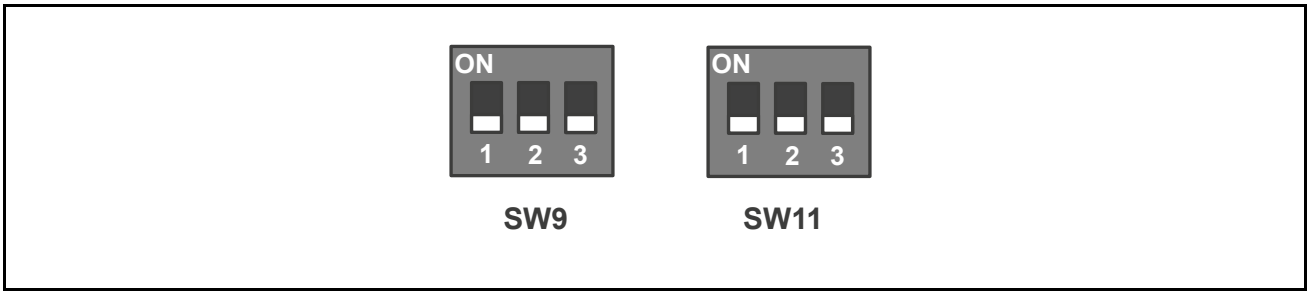


Figure 6.27. SW Settings for STO Function

(2) When selecting SS1-t function

Start the Safety Motor Control Reference Kit with SW9-1 and SW11-1 set to ON.

Figure 6.28. SW Settings for SS1-t Function

6.5.2 Selecting CiA402 Operating Mode

(1) In TwinCAT 3 System Manager tree, double-click [Box X (Renesas EtherCAT RZ/T2 SafetyMotorSolution)] to open the properties window.

(2) Click the [Slots] tab.

(3) In the left pane of the [Slots] tab, click the [Motor1] row.

(4) In the right pane, select the desired CiA402 operating mode.

(5) Click the [<] button to apply the change.

(6) Confirm that the [Motor1] row now shows the selected CiA402 operating mode.

(7) Click [Activate Configuration] in TwinCAT 3 to restart in Run Mode.

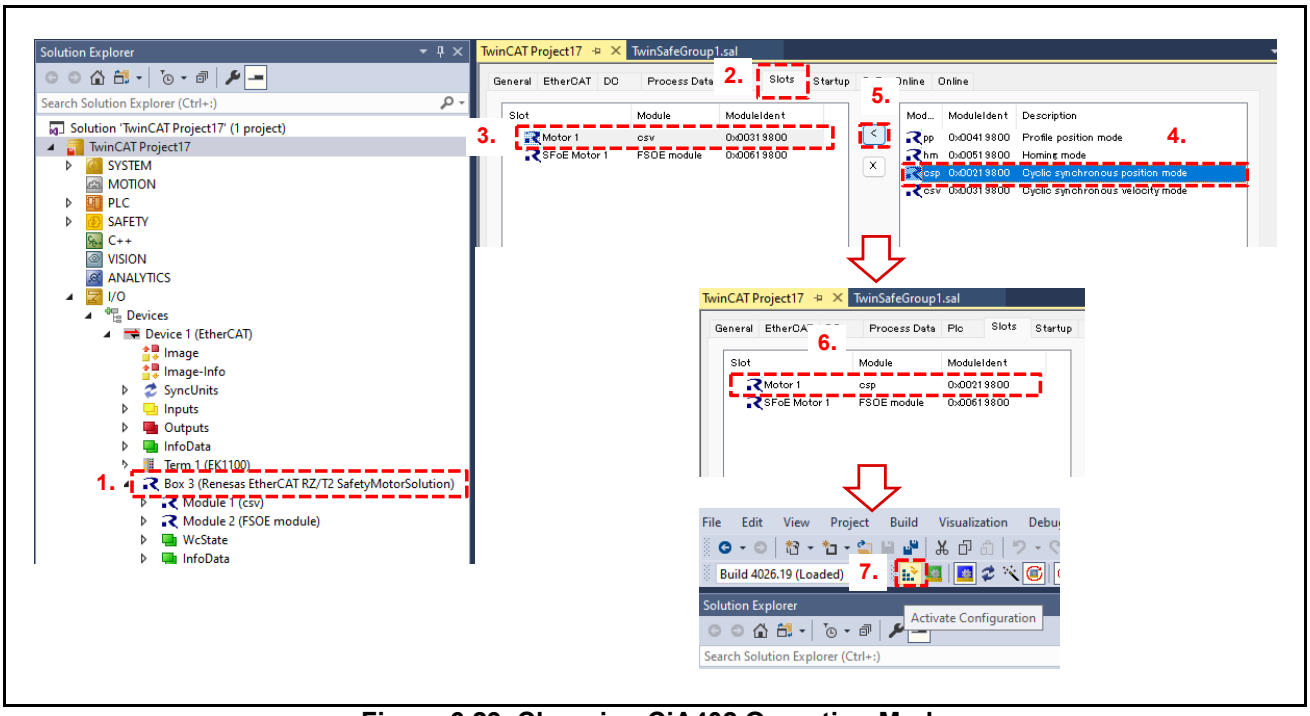


Figure 6.29. Changing CiA402 Operating Mode

6.5.3 Establishing FSoE Communication and Releasing Motor Stop Request

(1) Complete the steps in Section 6.4 and start the TwinCAT PLC program.

(2) Check communication status:

1. Navigate to [SAFETY] → [Safety_Drive] → [Safety_Drive Project] → [TwinSafeGroup1].
2. Right-click [TwinSafeGroup1.sal] and select [Open] to launch the TwinSAFE graphical editor.
3. In TwinCAT 3, click the [TWINSAFE] menu and select [Show Online Data].
4. Confirm that TwinSafeGroup is in Error state.

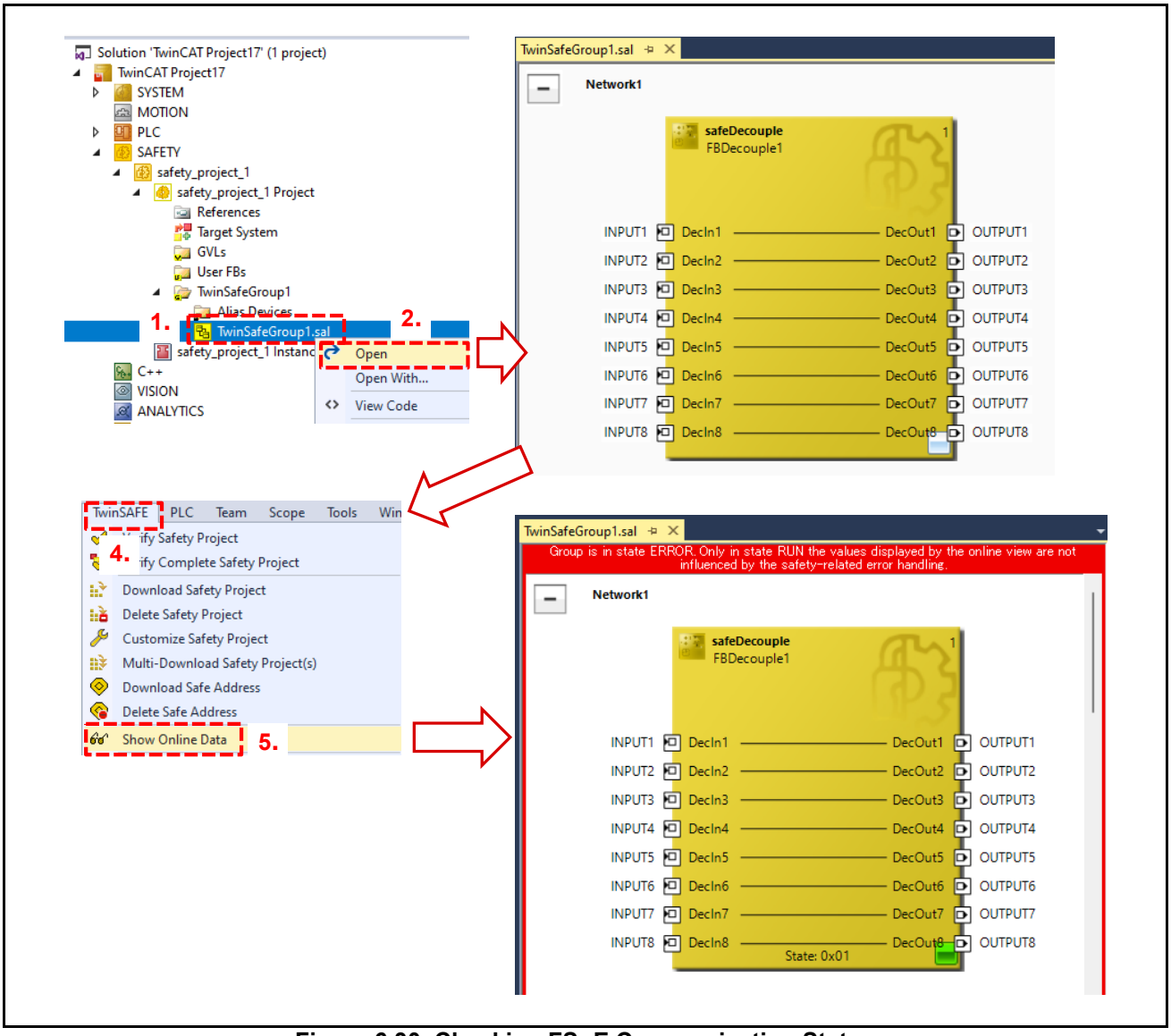


Figure 6.30. Checking FSoE Communication Status

7. Communication status can also be checked via LEDs LED13–LED31 on the board. When FSoE communication is in Error, the LEDs behave as shown in the figure.

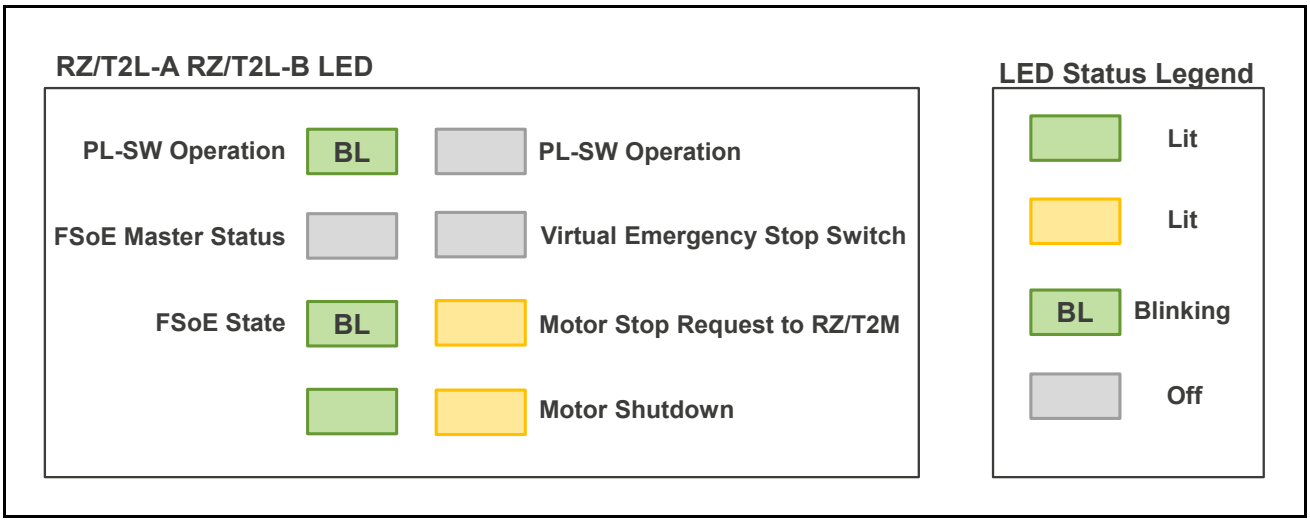


Figure 6.31. LED Status in Error State

(3) Next, write a value to the input variable ErrAckIn:

1. In the System Manager tree, click [MAIN.bErrAckIn].
2. Open the [Online] tab.
3. Click [Write] to open the Set Value Dialog window.
4. Enter “1” in [Dec].
5. click [OK].
6. Open the dialog again, enter “0”, and click [OK].

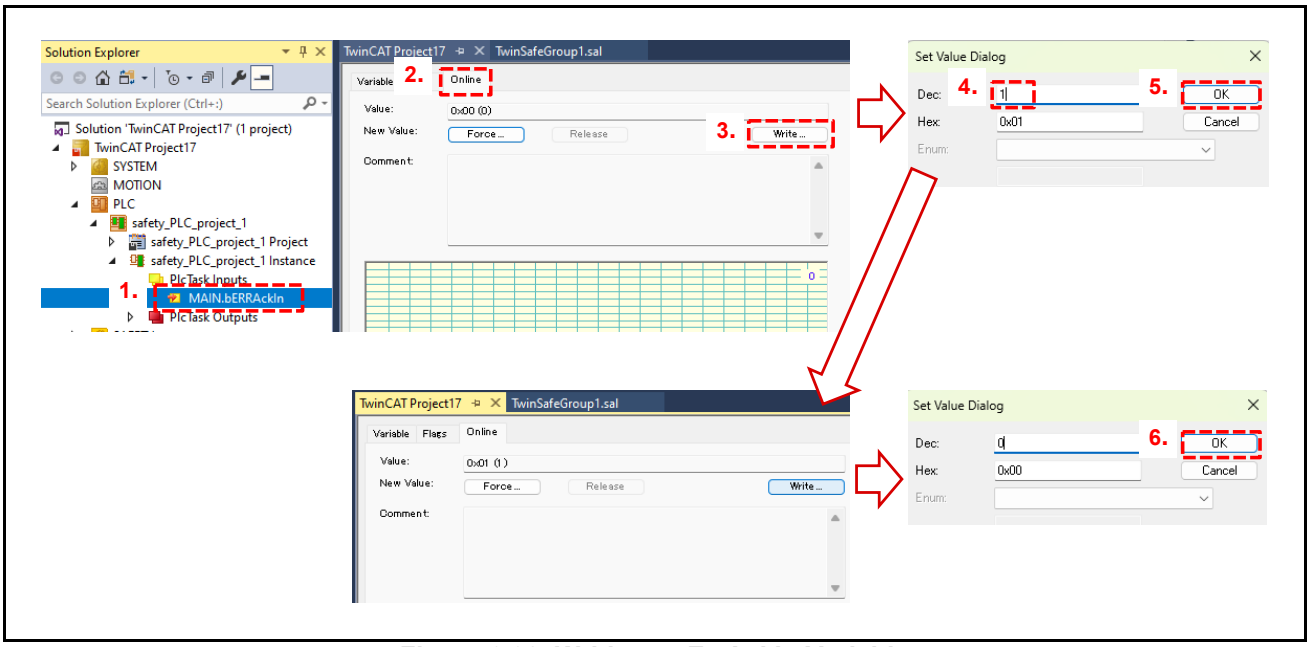


Figure 6.32. Writing to ErrAckIn Variable

7. After this, confirm communication status in TwinCAT. If successful, TwinSafeGroup will change to RUN state.
8. LEDs on the board will also indicate RUN state as shown below.

Figure 6.33. LED Status in RUN State

(4) Perform Safety Signal Input and Output

1. Press SW8 and SW10 on the Safety Motor Control Reference Kit.
2. When the board outputs the safety signal, LED44 to LED51 on the board will all turn ON.
3. At this time, the signal lines in the function block will be displayed in green.
4. Additionally, the LEDs on the board will appear as shown below: the motor stop request and motor cutoff LEDs for RZ/T2M will turn OFF.

Figure 6.34. TwinSAFEGroup and RZ/T2L-A, RZ/T2L-B LEDs during Safety Signal I/O

6.5.4 Controlling CiA402 State Machine

6.5.4.1 How to Transition to Operation Enable

1. Complete the steps in Section 6.5.3 to release the motor stop request for RZ/T2M.
2. In TwinCAT 3 System Manager tree, navigate to:[Box X (Renesas EtherCAT RZ/T2 SafetyMotorSolution)] → [Module 1] → [Inputs] → [Statusword], and double-click.
3. Open the [Flags] tab.
4. Change the Display mode of [Statusword] to Hex.
5. Open the [Online] tab.
6. Confirm that bit 3 of Statusword is 0. If it is 1, the system is in Fault state and recovery is required. Perform the steps in Section 6.5.4.2 to recover from Fault.

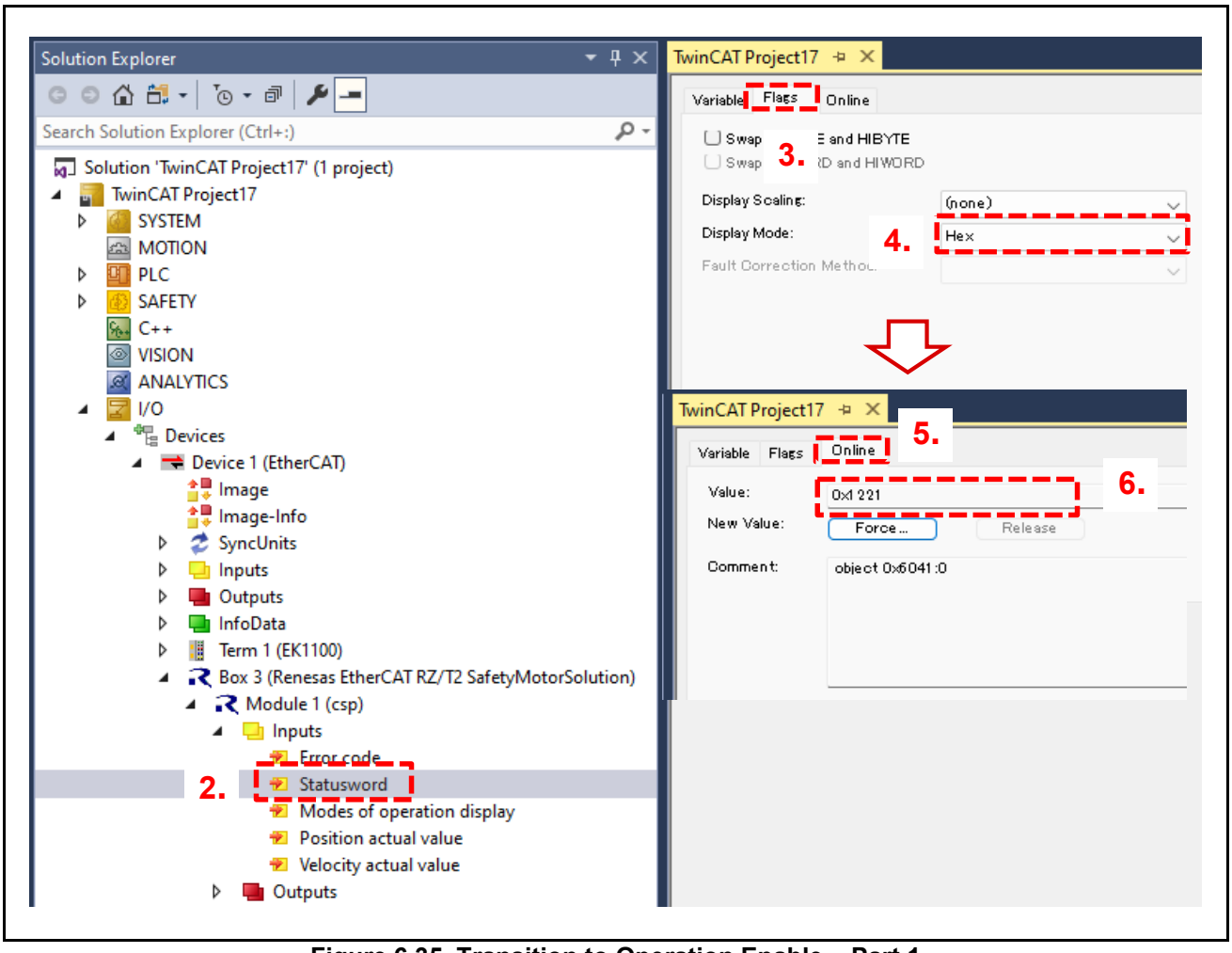


Figure 6.35. Transition to Operation Enable – Part 1

7. Navigate to [Outputs] → [Controlword], click it.
8. open the [Online] tab.
9. Click [Write].
10. Enter 15 and click OK.

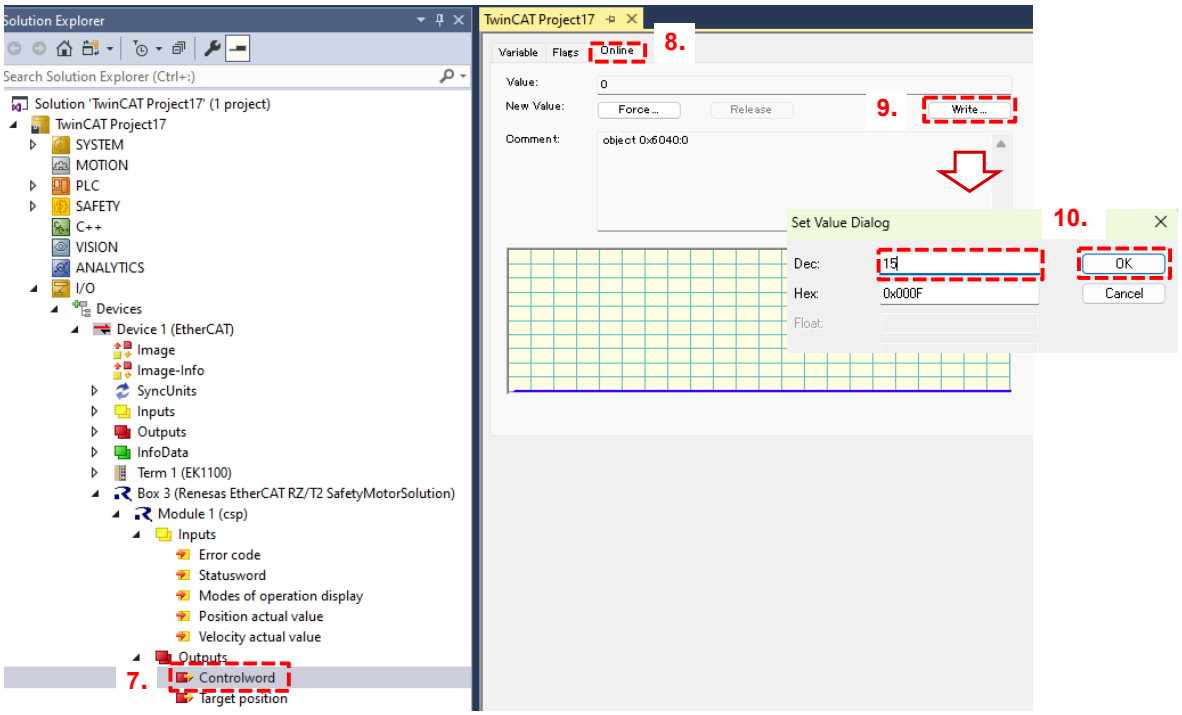


Figure 6.36. Transition to Operation Enable – Part 2

11. Confirm the value of [Statusword]. If bit 2 is 1, the transition to Operation Enable was successful.
12. You can also confirm this by checking LEDs LED8–LED15 on the board.

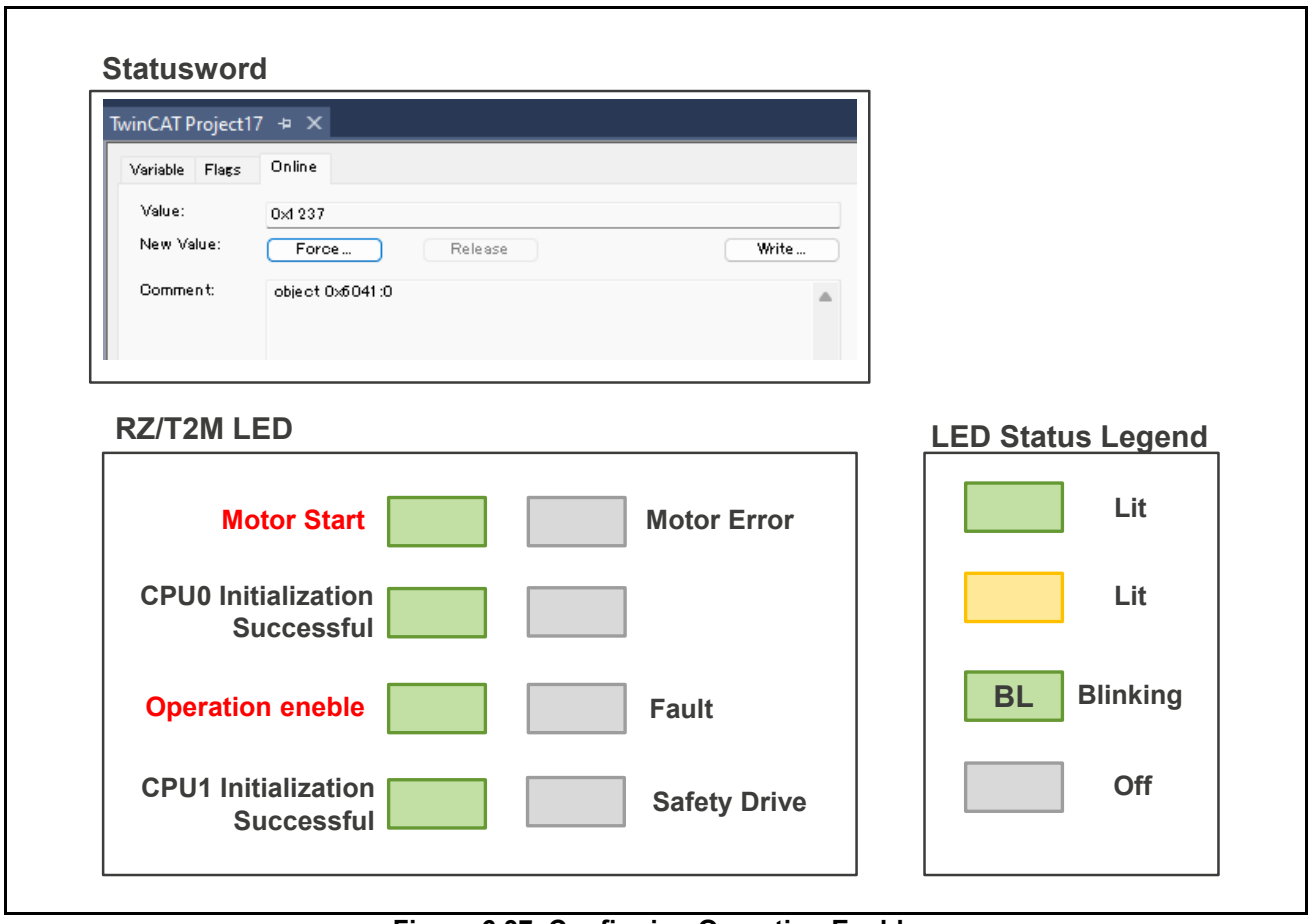


Figure 6.37. Confirming Operation Enable

Note:

If the motor stop request for RZ/T2M is not released, the transition to Operation Enable will fail and the system will enter Fault state.

6.5.4.2 How to Recover from Fault

1. If bit 3 of [Statusword] is 1, the system is in Fault state. You can also check LED14 on the board to confirm Fault state.

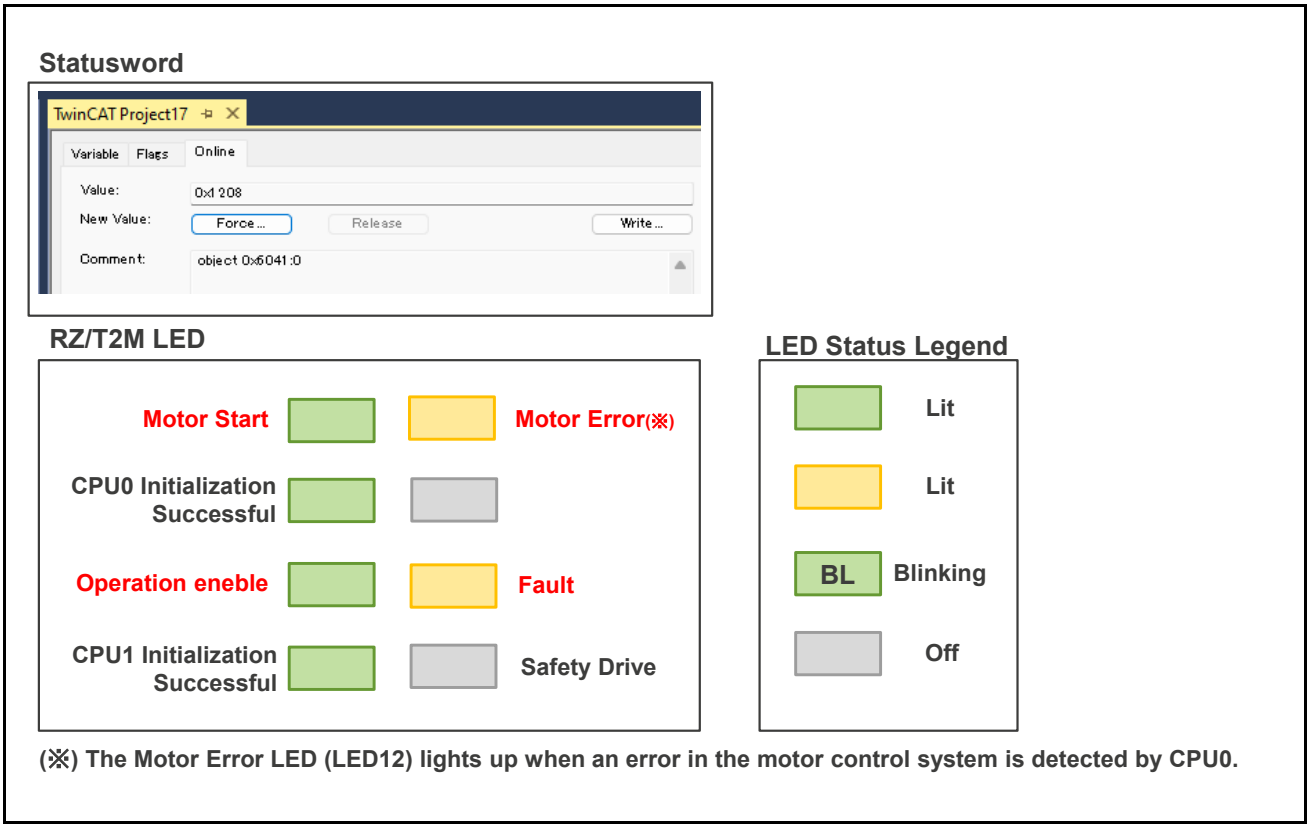


Figure 6.38. Identifying Fault State

2. In TwinCAT 3 System Manager tree, navigate to:[Box X (Renesas EtherCAT RZ/T2 SafetyMotorSolution)] → [Module 1] → [Outputs] → [Controlword].
3. Open the [Online] tab.
4. Click [Write].
5. Enter 128 and click OK.
6. Confirm that bit 3 of Statusword is now 0. Fault recovery is complete.

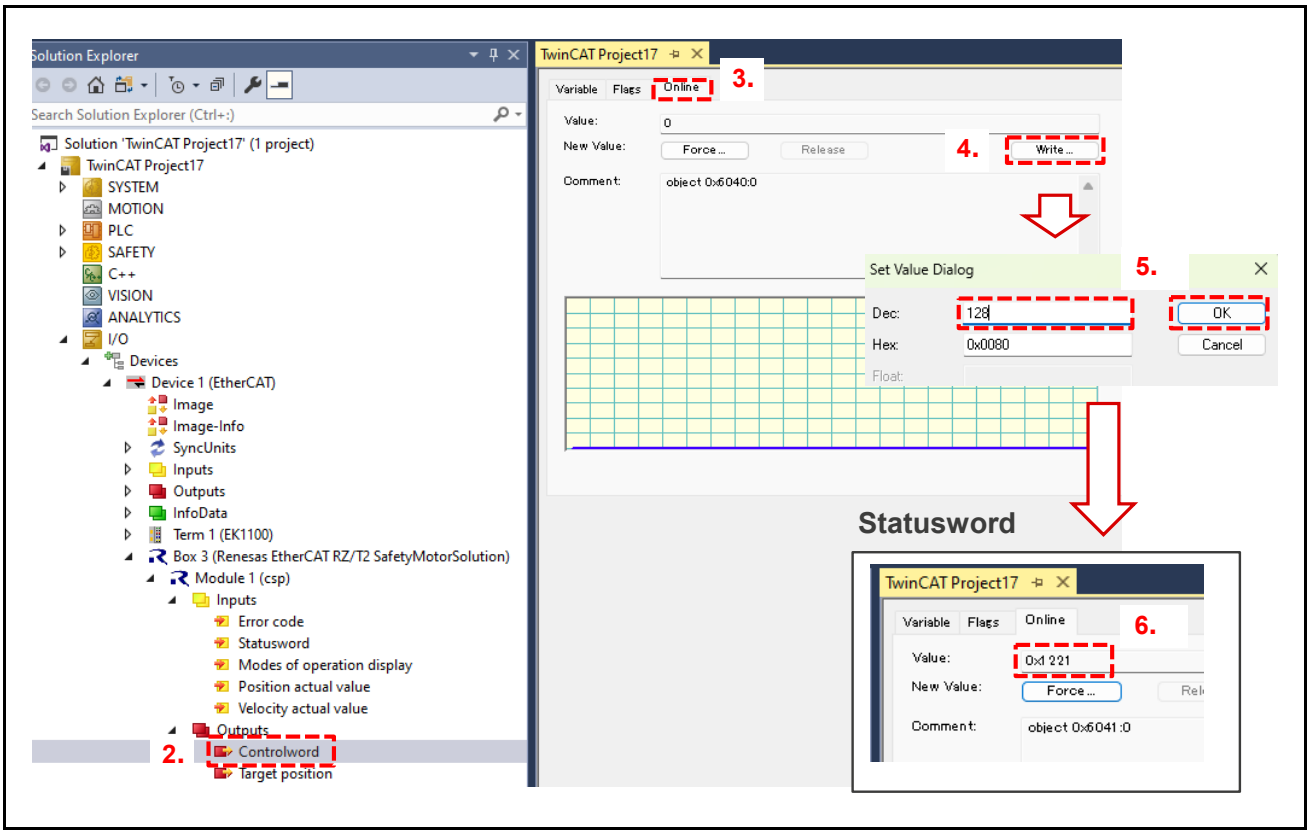


Figure 6.39. Fault Recovery

6.5.5 Motor Operation Methods

6.5.5.1 CSP Mode (Cyclic Synchronous Position)

1. Follow the steps in Section 6.5.2 to set the motor's CiA402 operating mode to CSP mode.
2. Follow the steps in Section 6.5.3 to release the motor stop request for RZ/T2M.
3. Follow the steps in Section 6.5.4 to transition the CiA402 state machine to Operation Enable.
4. In TwinCAT 3 System Manager tree, click [Target position].
5. Open the [Online] tab and click [Write].
6. Enter 36000 to issue a position command for the motor to move to 3600.0 deg.
7. The motor rotates and moves to 3600.0 deg.
8. In the System Manager tree, click [Position actual value].
9. Open the [Online] tab and confirm that the value is 36000.

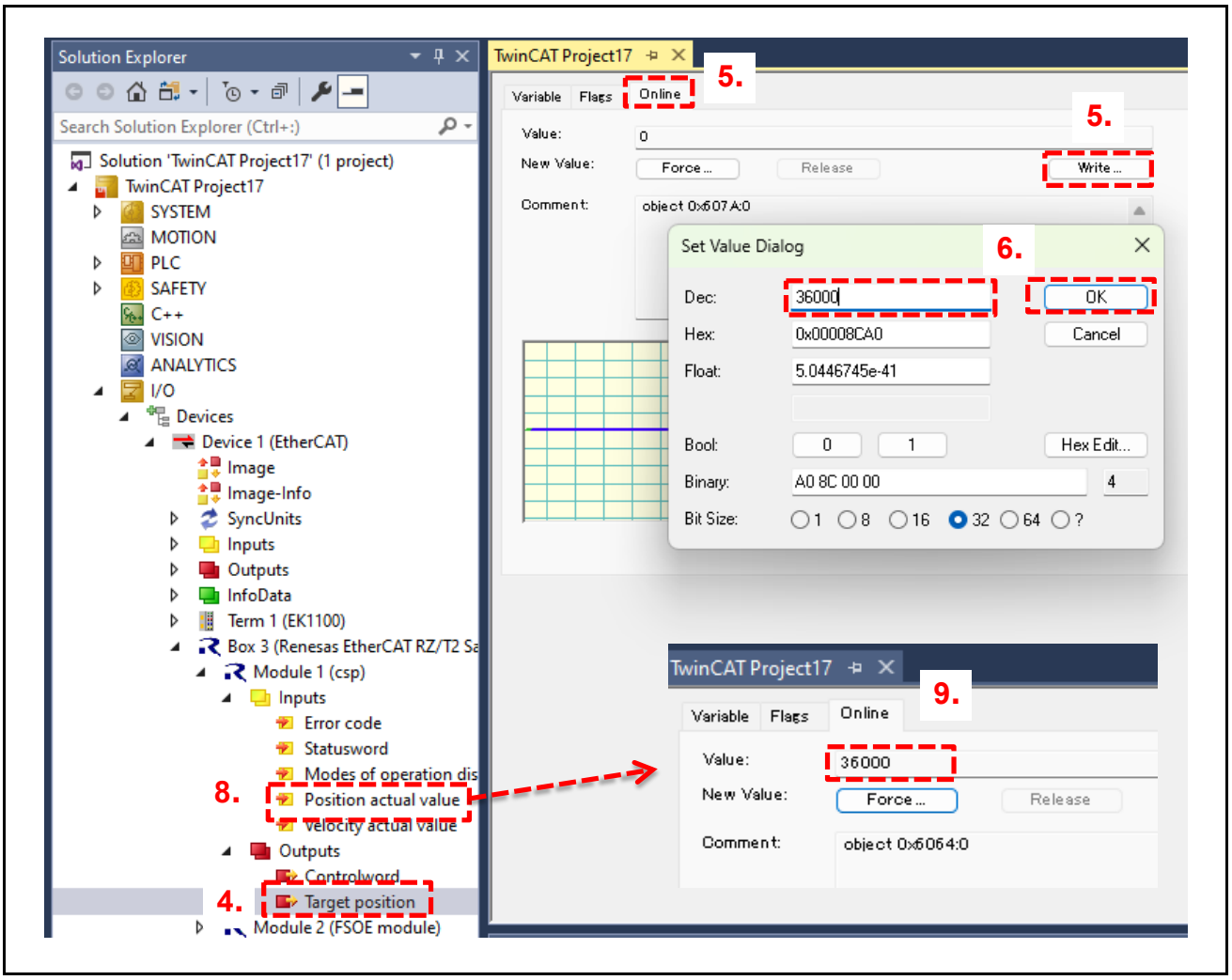


Figure 6.40. CSP Mode Operation Check

6.5.5.2 PP Mode (Profile Position)

1. Follow the steps in Section 6.5.2 to set the motor's CiA402 operating mode to PP mode.
2. Follow the steps in Section 6.5.3 to release the motor stop request for RZ/T2M.
3. Follow the steps in Section 6.5.4 to transition the CiA402 state machine to Operation Enable.
4. In the System Manager tree, click [Profile velocity].
5. Open the [Online] tab and click [Write].
6. Enter 30000 to set the velocity profile to 3000.0 rpm.
7. In the System Manager tree, click [Profile acceleration].
8. Open the [Online] tab, click [Write].
9. Enter 150000 to set the acceleration profile to 15000.0 rpm/s.

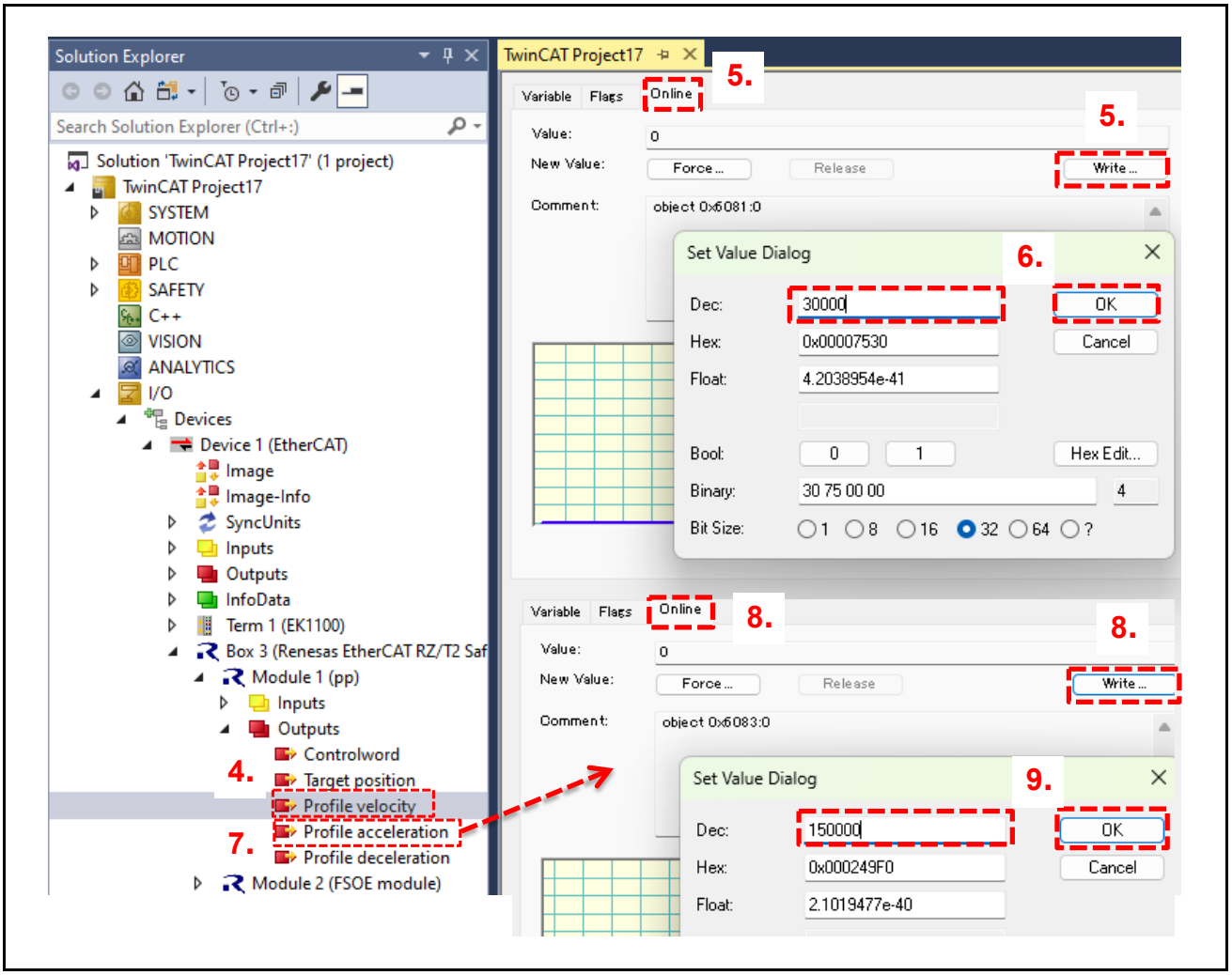


Figure 6.41. PP Mode Operation Check – Part 1

10. In the System Manager tree, click [Target position].
11. Open the [Online] tab, click [Write].
12. Enter 36000 to issue a position command for the motor to move to 3600.0°.
13. The motor rotates according to the set profile and moves to 3600.0°.
14. In the System Manager tree, click [Position actual value].
15. Open the [Online] tab and confirm that the value is 36000.

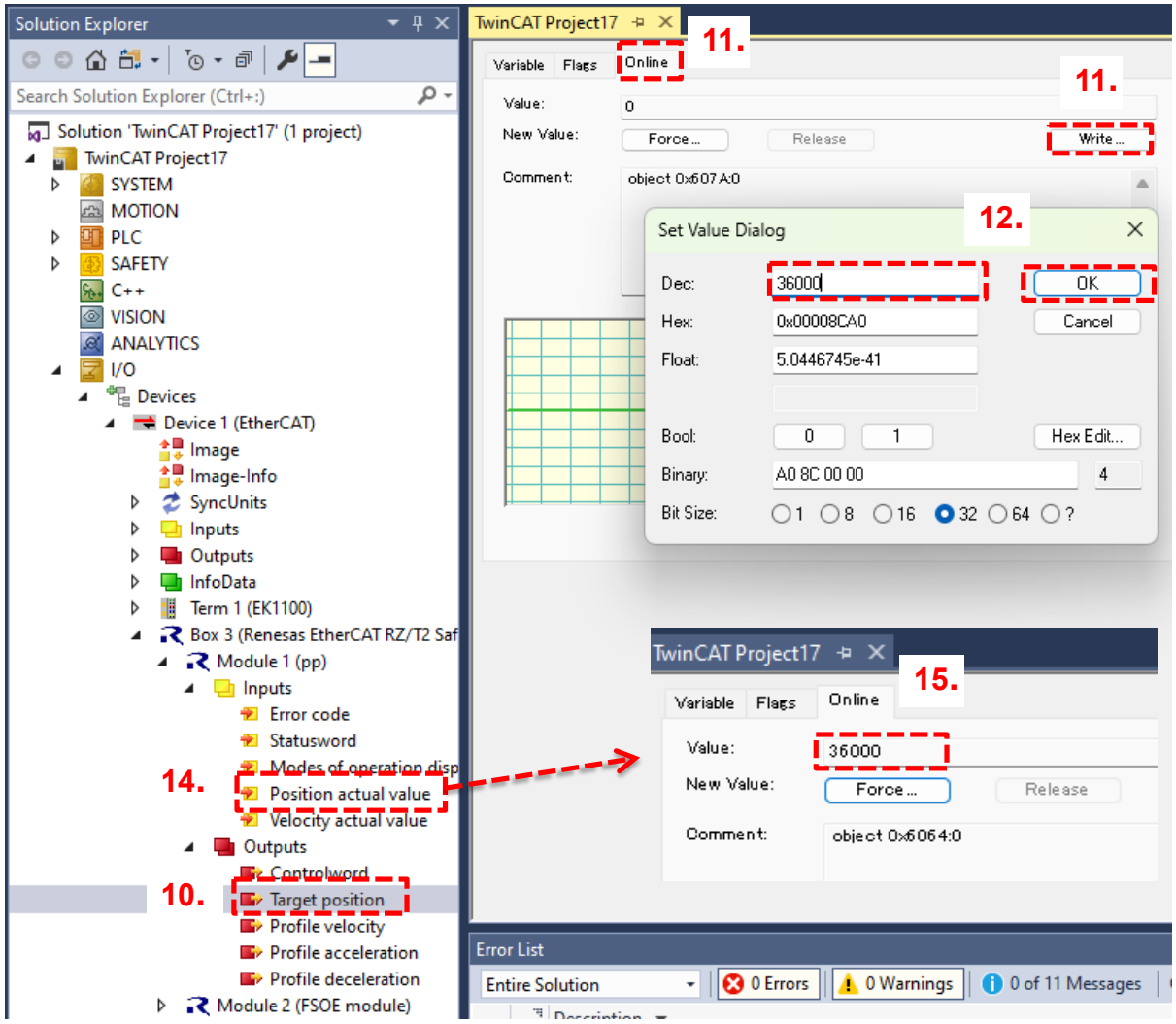


Figure 6.42. PP Mode Operation Check – Part 2

6.5.5.3 CSV Mode (Cyclic Synchronous Velocity)

1. Follow the steps in Section 6.5.2 to set the motor's CiA402 operating mode to CSV mode.
2. Follow the steps in Section 6.5.3 to release the motor stop request for RZ/T2M.
3. Follow the steps in Section 6.5.4 to transition the CiA402 state machine to Operation Enable.
4. In the System Manager tree, click [Target velocity].
5. Open the [Online] tab and click [Write].
6. Enter 20000 to issue a speed command for the motor to rotate at 2000 rpm.
7. The motor rotates at 2000 rpm.
8. Click [Velocity actual value], open the [Online] tab, and confirm that the value is around 20000.

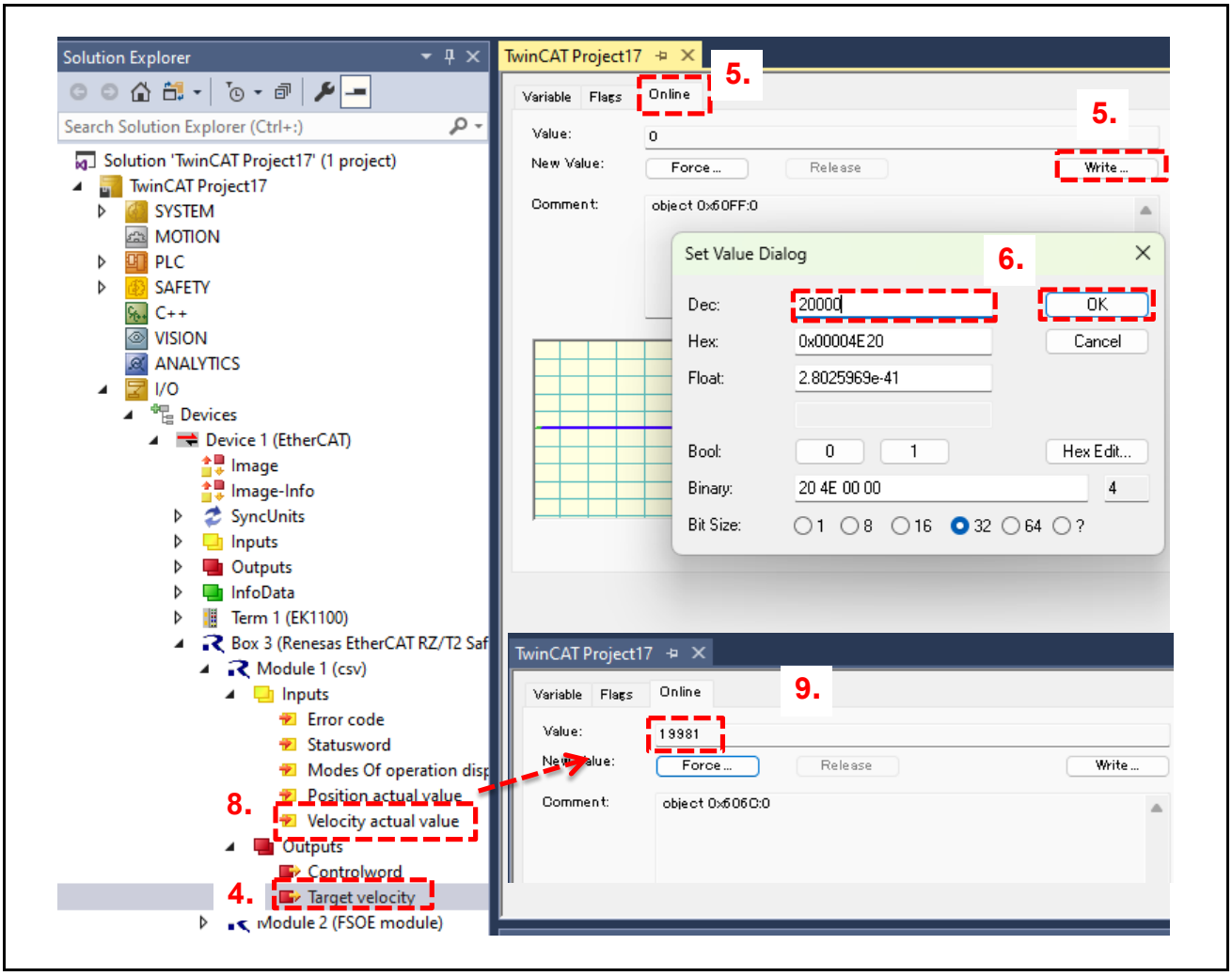


Figure 6.43. CSV Mode Operation Check

6.5.5.4 HM Mode (Homing)

1. Follow the steps in Section 6.5.2 to set the motor's CiA402 operating mode to HM mode.
2. Follow the steps in Section 6.5.3 to release the motor stop request for RZ/T2M.
3. Follow the steps in Section 6.5.4 to transition the CiA402 state machine to Operation Enable.
4. In the System Manager tree, click [Position actual value].
5. Open the [Online] tab and check the current value.
6. In the System Manager tree, click [Home offset].
7. Confirm that it is 0.

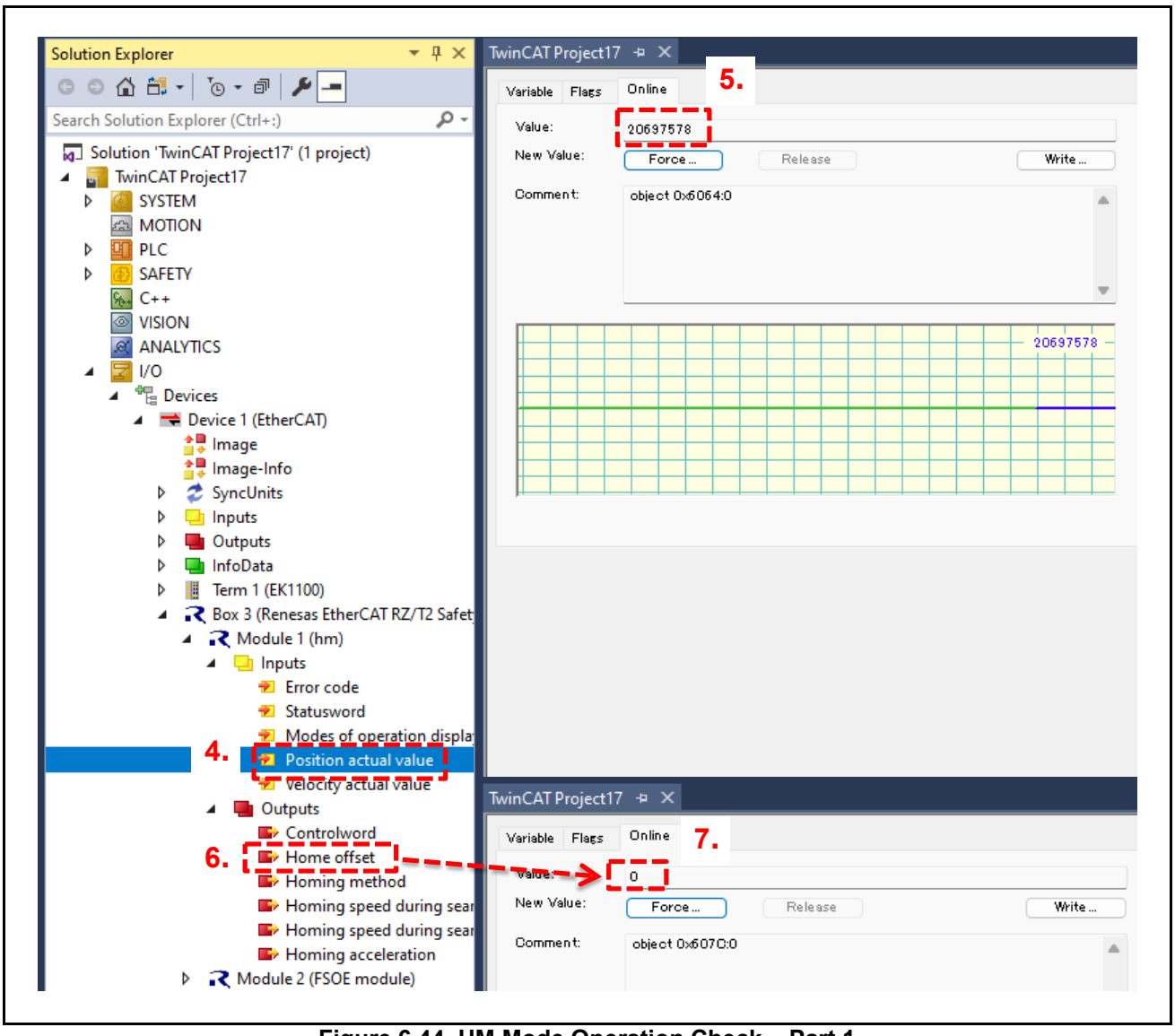


Figure 6.44. HM Mode Operation Check – Part 1

8. In the System Manager tree, click [Home method].
9. Open the [Online] tab and click [Write].
10. Enter 37 to specify Home Method 37.

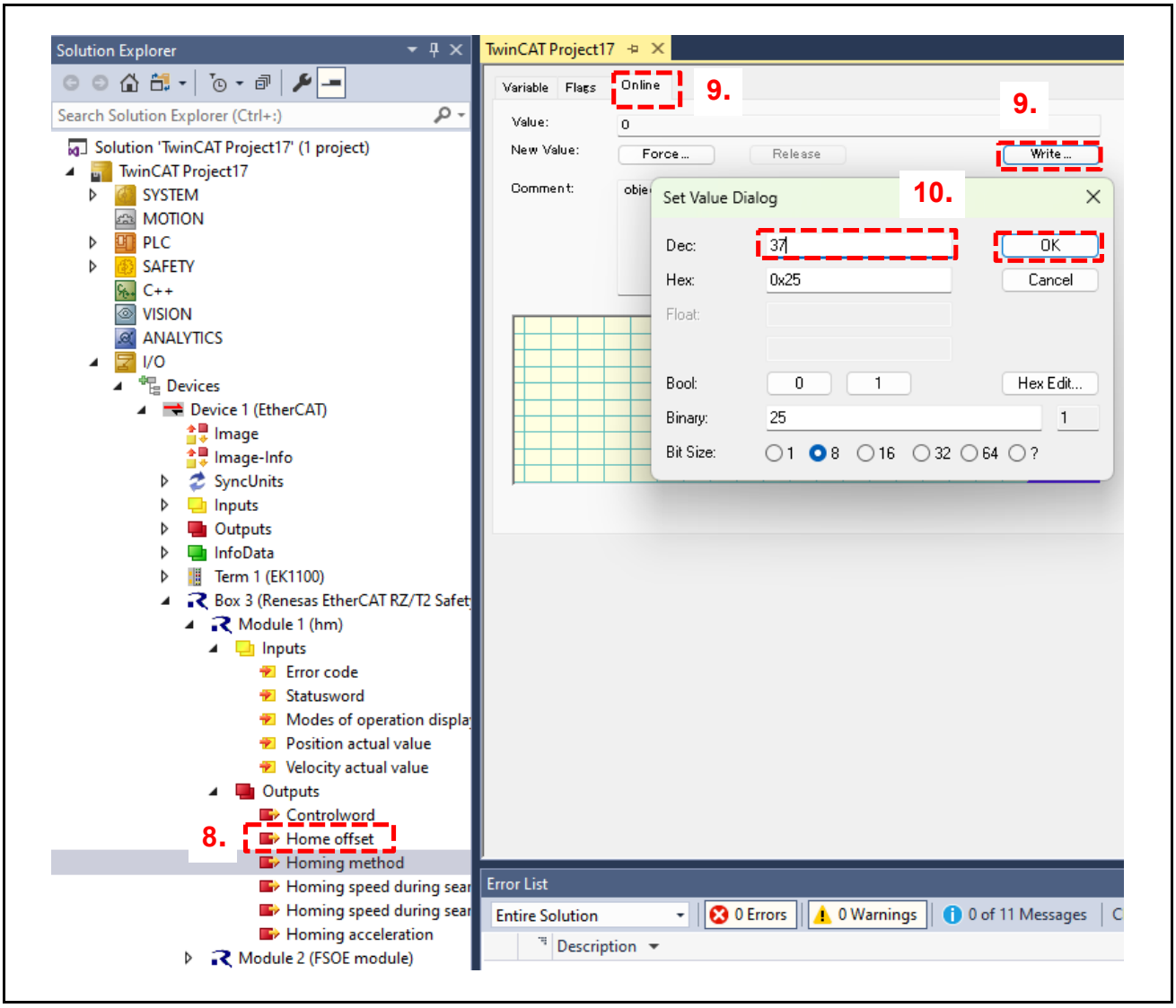


Figure 6.45. HM Mode Operation Check – Part 2

11. In the System Manager tree, click [Controlword].
12. Open the [Online] tab and click [Write].
13. Enter 31 to execute Home Method 37.
14. In the System Manager tree, click [Position actual value].
15. Open the [Online] tab and confirm that the value has changed to 0.

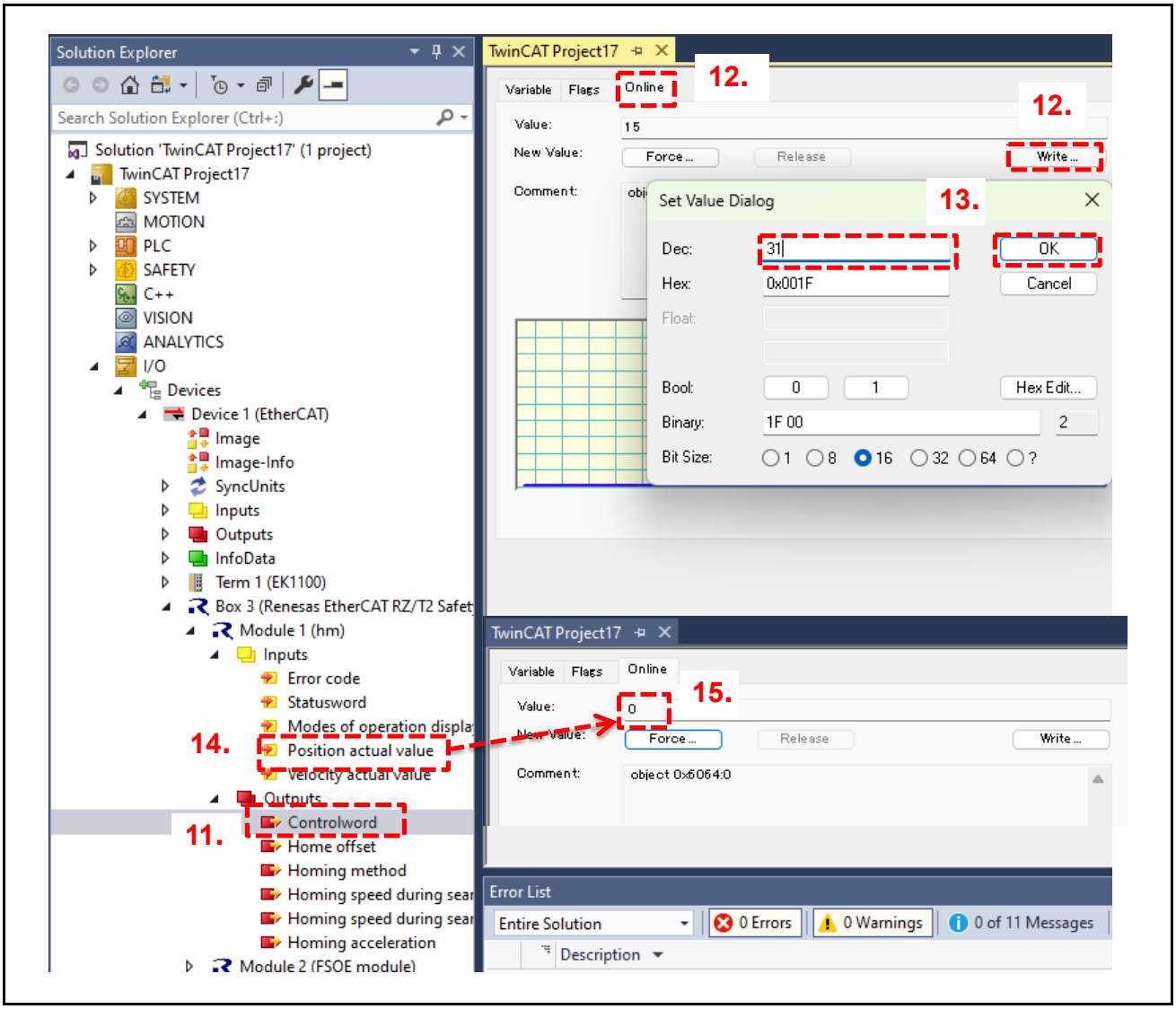


Figure 6.46. HM Mode Operation Check – Part 3

6.5.6 Verifying Safety Drive Functions

6.5.6.1 STO (Safe Torque Off)

1. Set the Safety Drive function to STO according to Section 6.5.1.
2. Rotate the motor in CSV mode following Section 6.5.5.3.
3. Turn SW9-2 ON on the board to execute STO.
4. When STO is executed, the motor stops immediately.
5. After turning SW9-2 OFF, you can restore the system by following the steps in Sections 6.5.3 and 6.5.4.2.

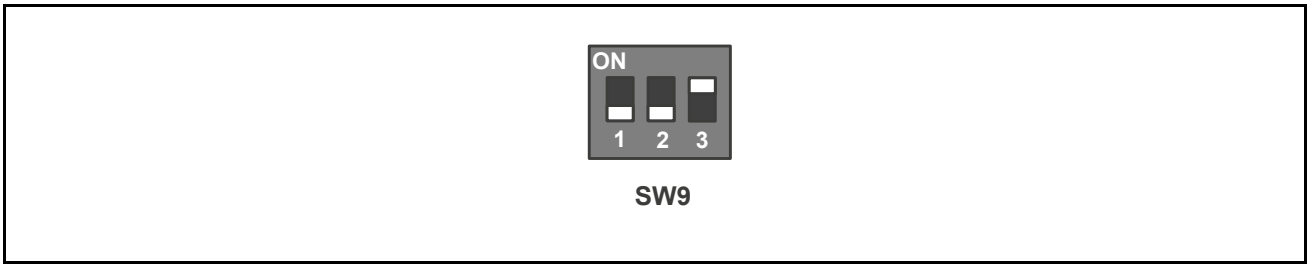


Figure 6.47. SW Settings for STO Execution

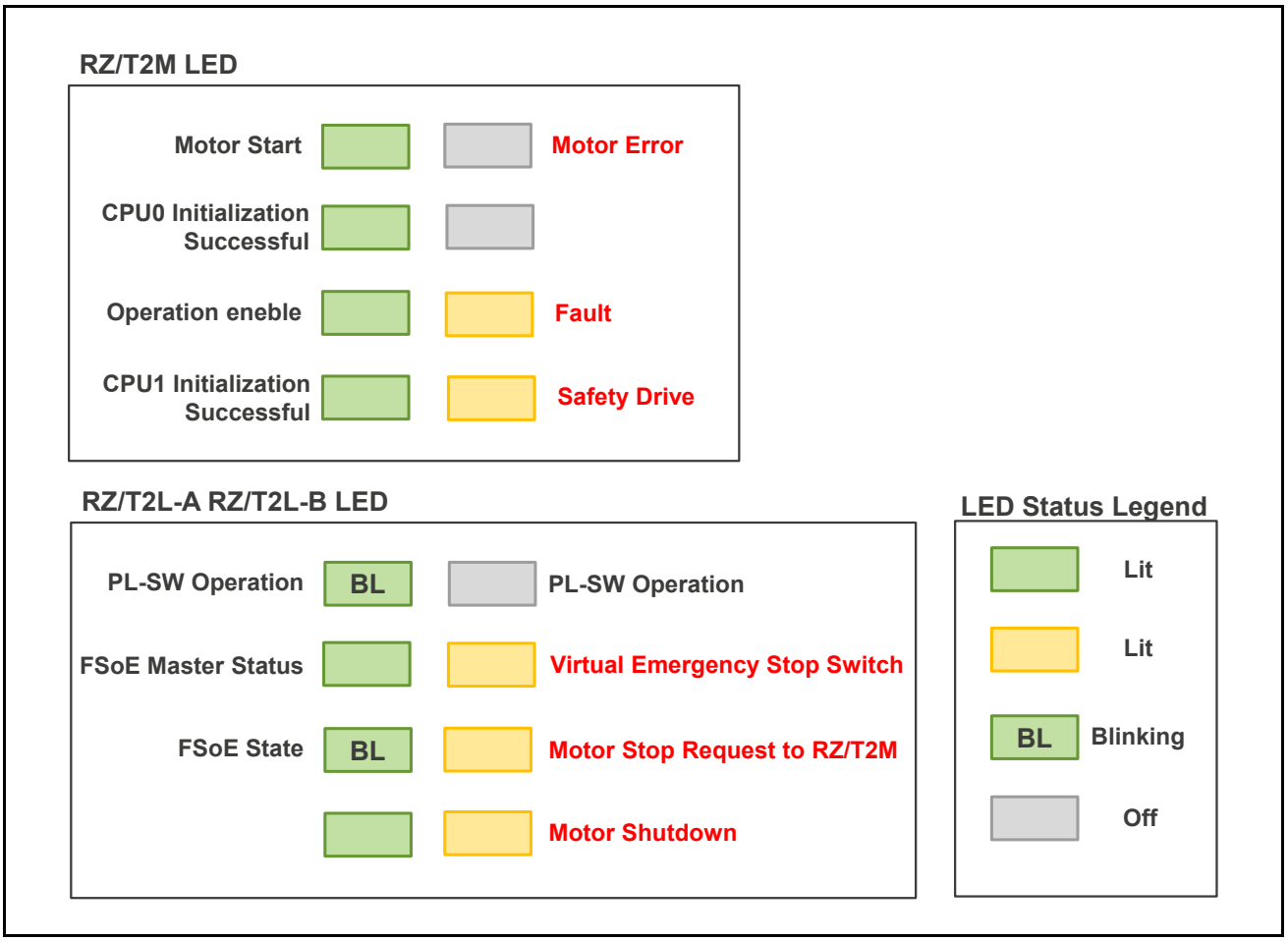


Figure 6.48. LED Status After STO Execution

6.5.6.2 SS1-t (Safe Stop 1 – Time Controlled)

1. Set the Safety Drive function to SS1-t according to Section 6.5.1.
2. Rotate the motor in CSV mode following Section 6.5.5.3.
3. Turn SW9-2 ON on the board to execute SS1-t.
4. When SS1-t is executed, the motor speed gradually decreases and reaches 0 rpm after 3 seconds.
5. After turning SW9-2 OFF, you can restore the system by following the steps in Sections 6.5.3 and 6.5.4.2.

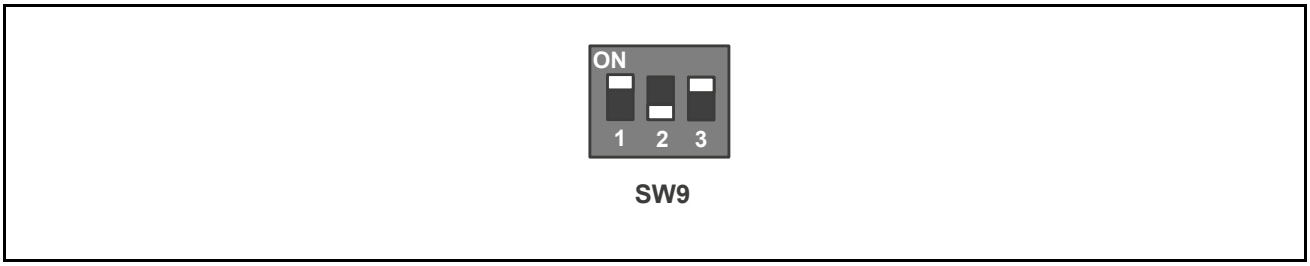


Figure 6.49. SW Settings for SS1-t Execution

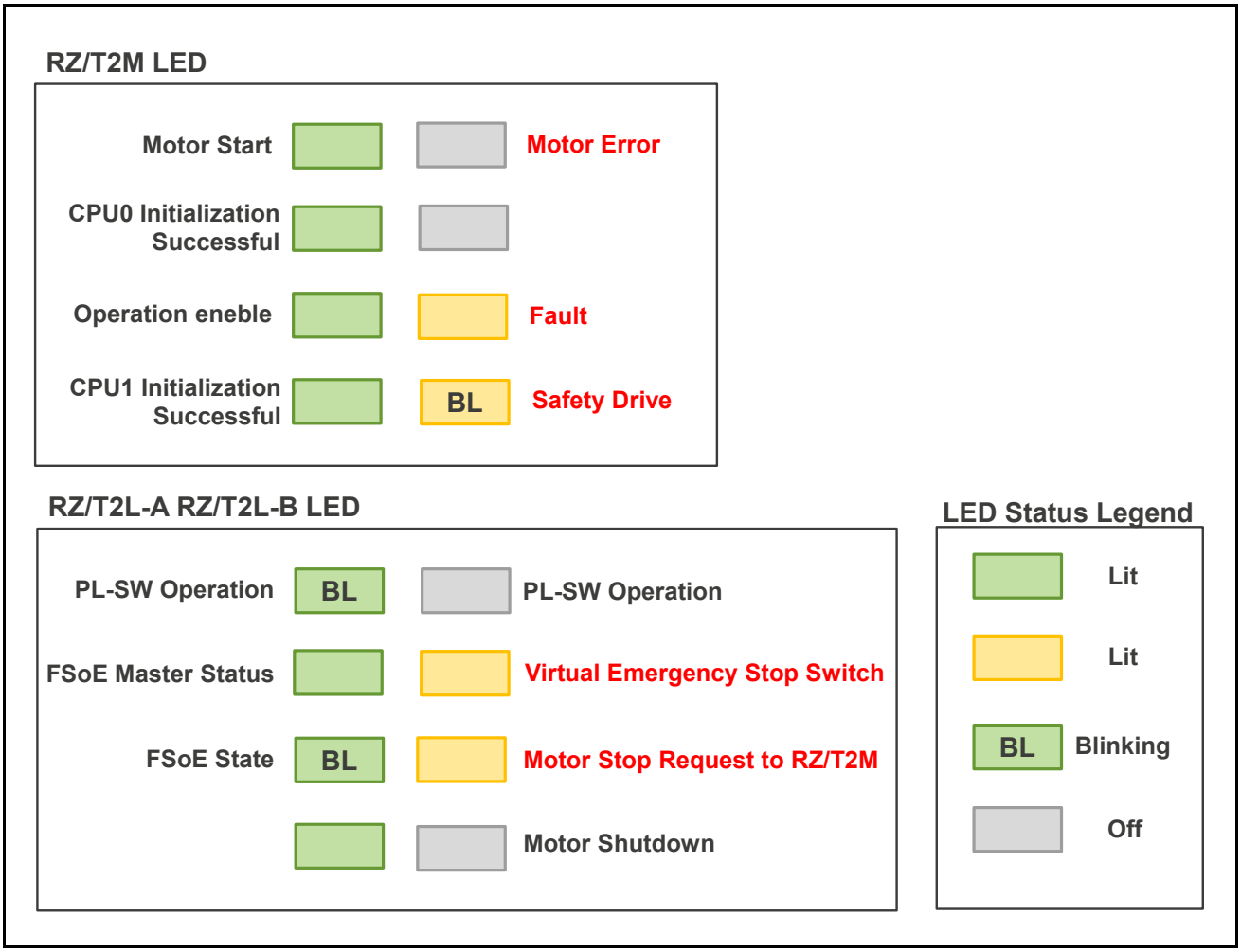


Figure 6.50. LED Status During SS1-t Execution

6.5.6.3 Safety Control on RZ/T2L Fault

1. Rotate the motor in CSV mode following Section 6.5.5.3.
2. Operate any of SW13–SW16 on the board to trigger a fault in the RZ/T2L system.
3. The motor stops immediately.
4. Restore the SW13–SW16 settings, press SW2 to restart the system, and then follow the steps in Sections 6.5.3 and 6.5.4.2 to recover.

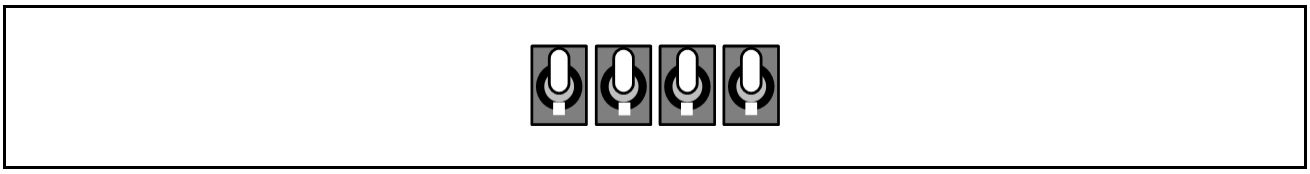


Figure 6.51. SW13–SW16 Settings

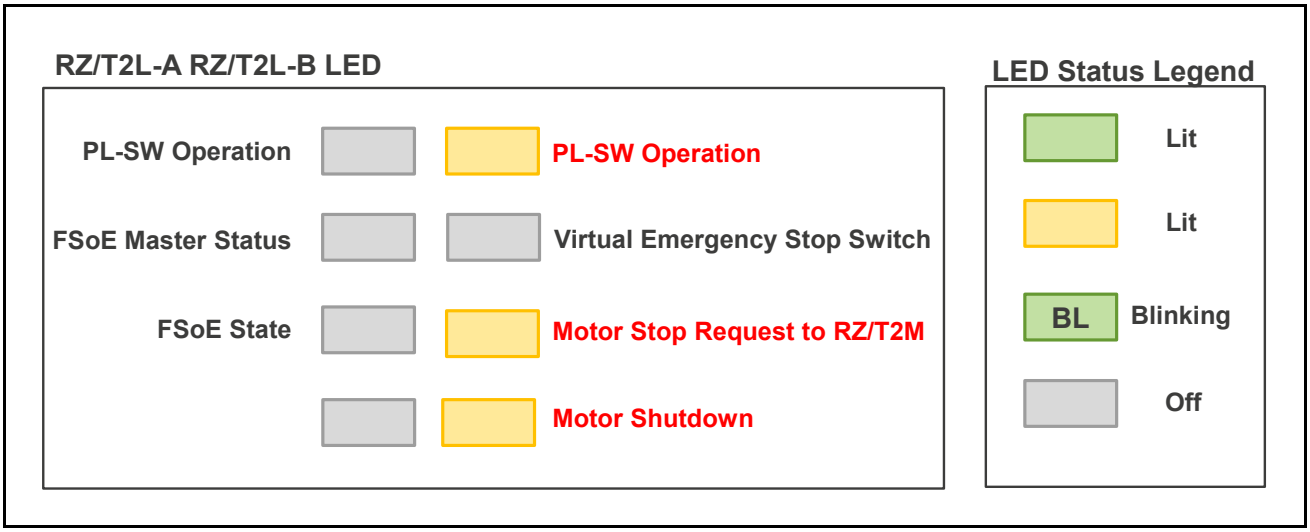


Figure 6.52. LED Status During RZ/T2L Fault

6.5.6.4 Safety Control on FSoE Master Fault

1. Rotate the motor in CSV mode following Section 6.5.5.3.
2. Disconnect the LAN cable connected to the In Port of the Safety Motor Control Reference Kit.
3. The motor stops immediately.
4. Reconnect the LAN cable and follow the steps in Sections 6.5.3 and 6.5.4.2 to recover.

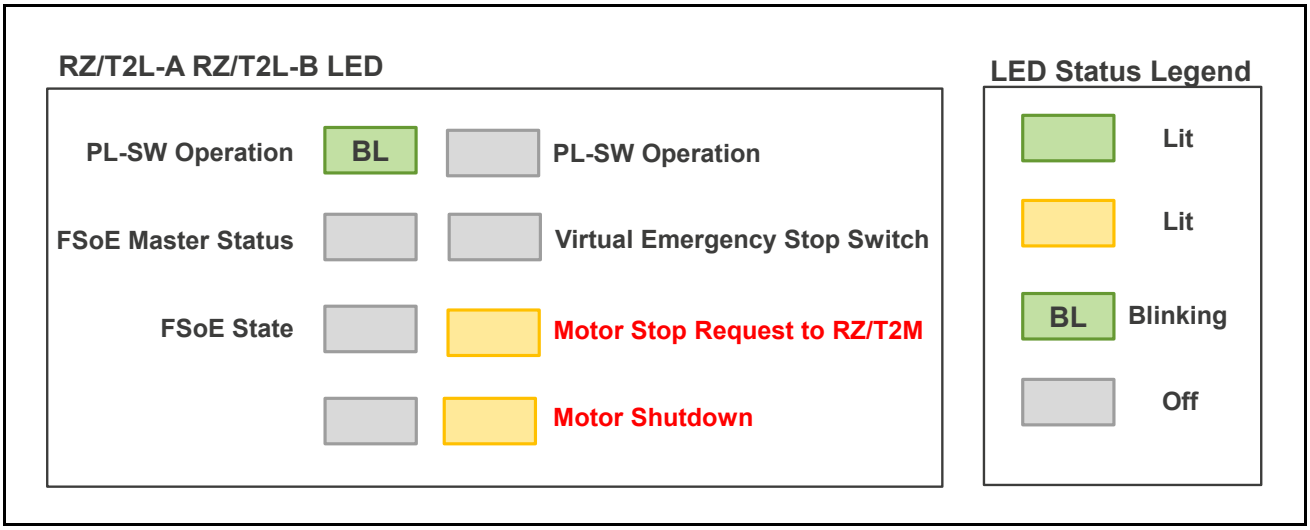


Figure 6.53. LED Status During FSoE Master Fault

7. Software Specification

7.1 Software Configuration

Figure 7.1 shows software configuration of safety motor control kit.

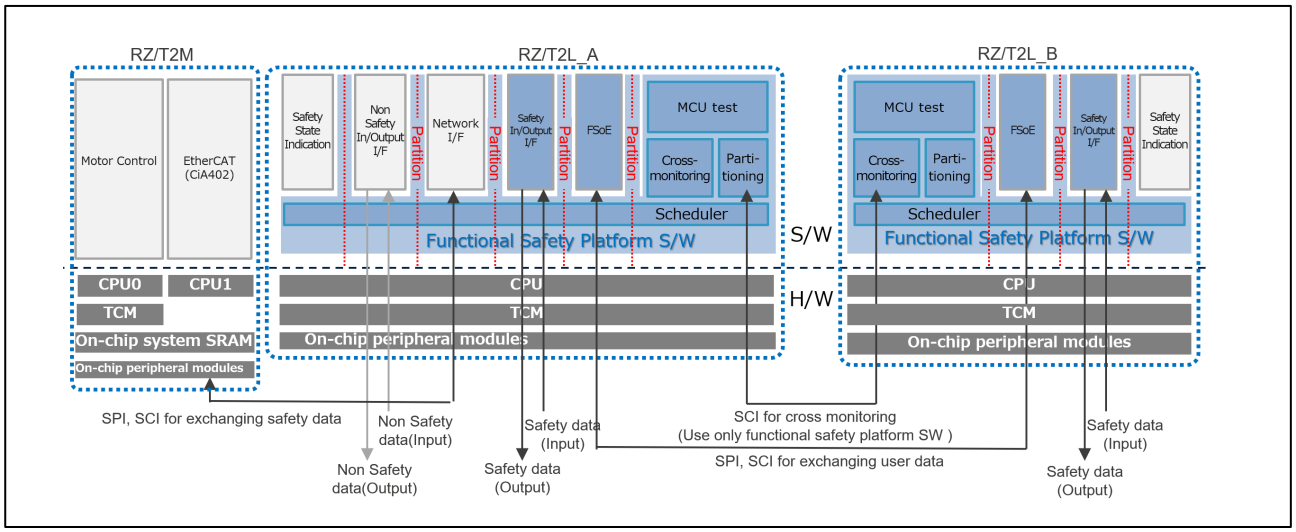


Figure 7.1. Software Configuration of Safety Motor Control Kit

7.2 Motor Control (RZ/T2M CPU0)

7.2.1 Overall Software Structure

The CPU0 software of the RZ/T2M in this sample program is based on “RZ/T2M Encoder Vector Control of Permanent Magnet Synchronous Motor – Serial Encoder (R01AN8003)”.

For details of the base project, refer to the application note R01AN8003.

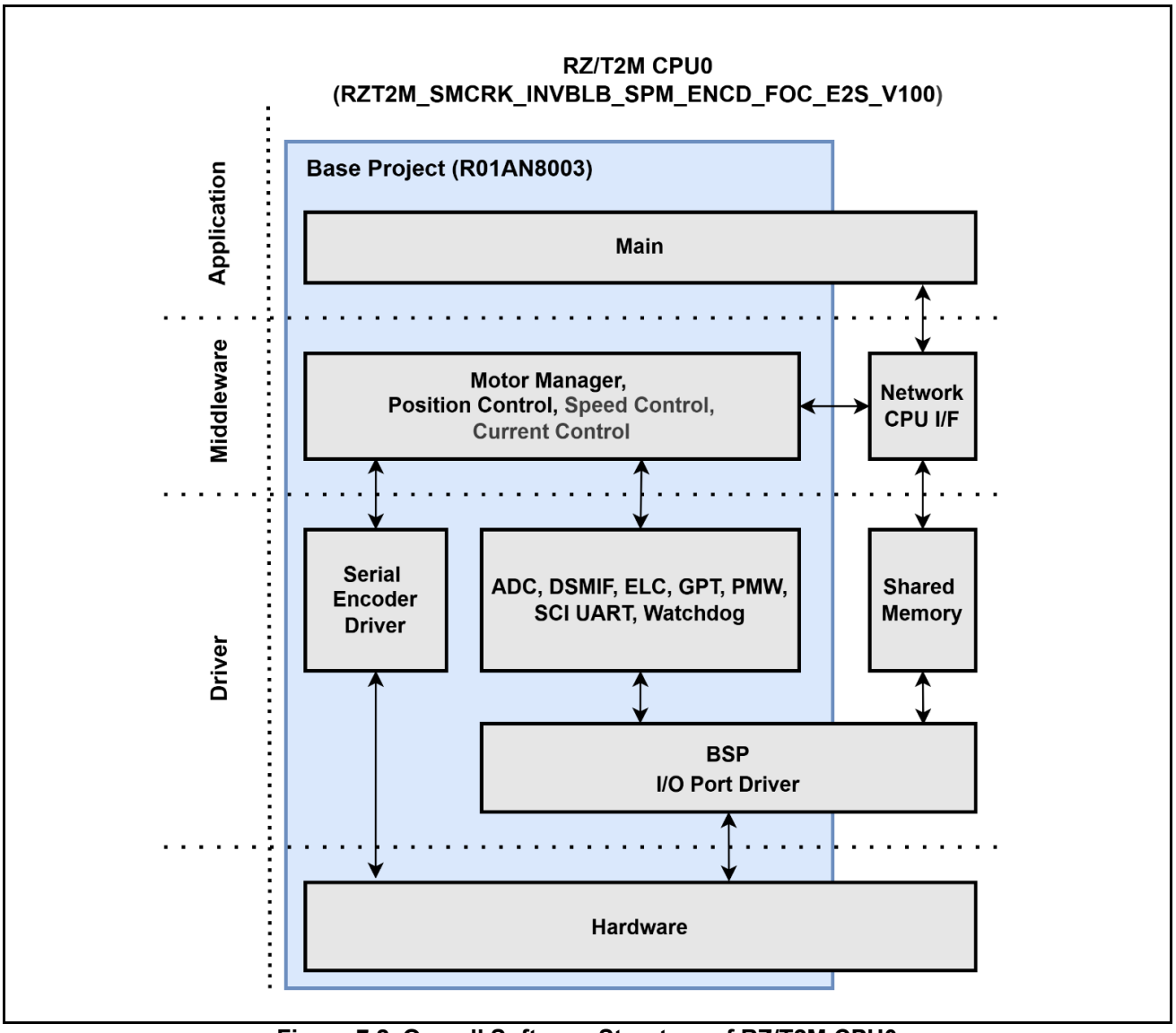


Figure 7.2. Overall Software Structure of RZ/T2M CPU0

7.2.2 Changes from the Base Project

This section describes the changes made from the base project.

Table 7.1. Changes from Base Project

Folder / File Changed			Description of Change
rzt			Files and folders modified or added due to changes in Configuration.xml
rzt_gen			
rzt_cfg			
src	network_cpu_if	r_network_cpu_if.c	Added Network CPU I/F module (see Section 7.2.3)
		r_network_cpu_if.h	
	serial_encoder_driver\mcu\rzt2m\lib\ecl	r_ecl_rzt2_if.h	Updated EC library version from 2.00 to 3.00
		r_ecl_rzt2_gcc.a	
	hal_entry.c		Modified processing inside hal_entry function (see Figure Figure 7.2)
configuration.xml			Added Shared Memory Driver. (settings in Table 7.2, Table 7.3, Table 7.4) Enabled I/O Port P01_6. (LED9) Set CPU1 clock to 800 MHz.
.cproject			Added include path for “src/network_cpu_if” . Added RZ/T2M CPU1 object file.

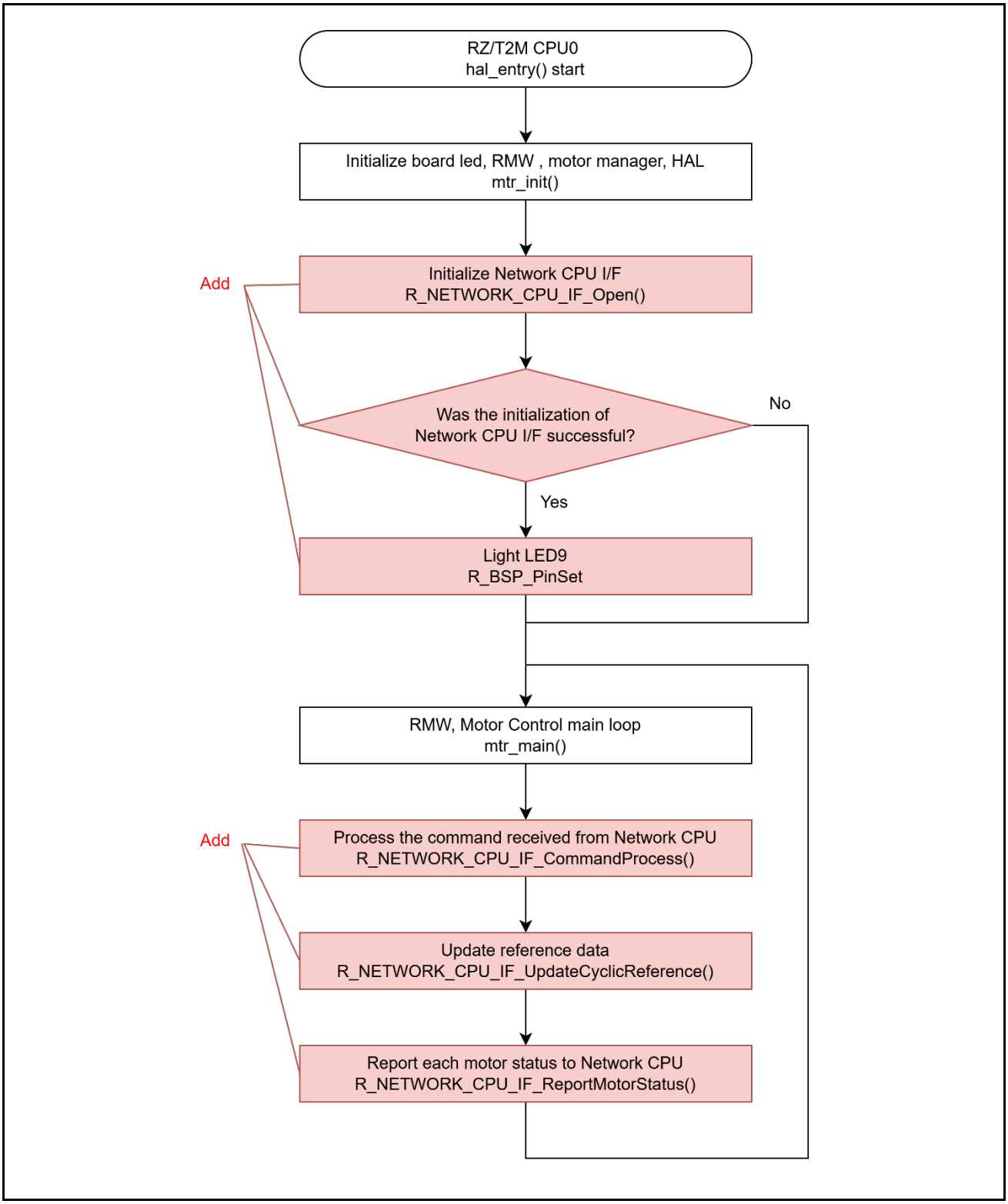


Figure 7.3. Flowchart of hal_entry Function

Table 7.2. Added Shared Memory Driver Settings (r_shared_memory)

Property	Setting
Common - Parameter Checking	Default (BSP)
Module - Name	g_shared_memory0
Module - Shared Memory Address	0x301A0000
Module - Shared Memory Size	1024
Module - Semaphore	0
Module - Callback	shared_memory_callback

Table 7.3. Added Shared Memory Driver Settings (r_icu_inter_cpu_irq, ch0)

Property	Setting
Common - Parameter Checking	Default (BSP)
Common - Multiplex Interrupt	Disabled
Module - Name	g_icu_inter_cpu_irq0
Module - Channel	0
Module - Software Interrupt Priority	Priority6
Module - Callback	r_shared_memory_callback

Table 7.4. Added Shared Memory Driver Settings (r_icu_inter_cpu_irq, ch1)

Property	Setting
Common - Parameter Checking	Default (BSP)
Common - Multiplex Interrupt	Disabled
Module - Name	g_icu_inter_cpu_irq1
Module - Channel	1
Module - Software Interrupt Priority	Disabled
Module - Callback	r_shared_memory_callback

7.2.3 Network CPU I/F Module

The Network CPU I/F module uses the Shared Memory Driver to exchange motor control commands, reference values, and motor status between the Network CPU (RZ/T2M CPU1) and the motor control system.

It also calls the Motor Manager API to execute control operations.

7.2.3.1 Module Structure

The module structure diagram is shown in Figure 7.4.

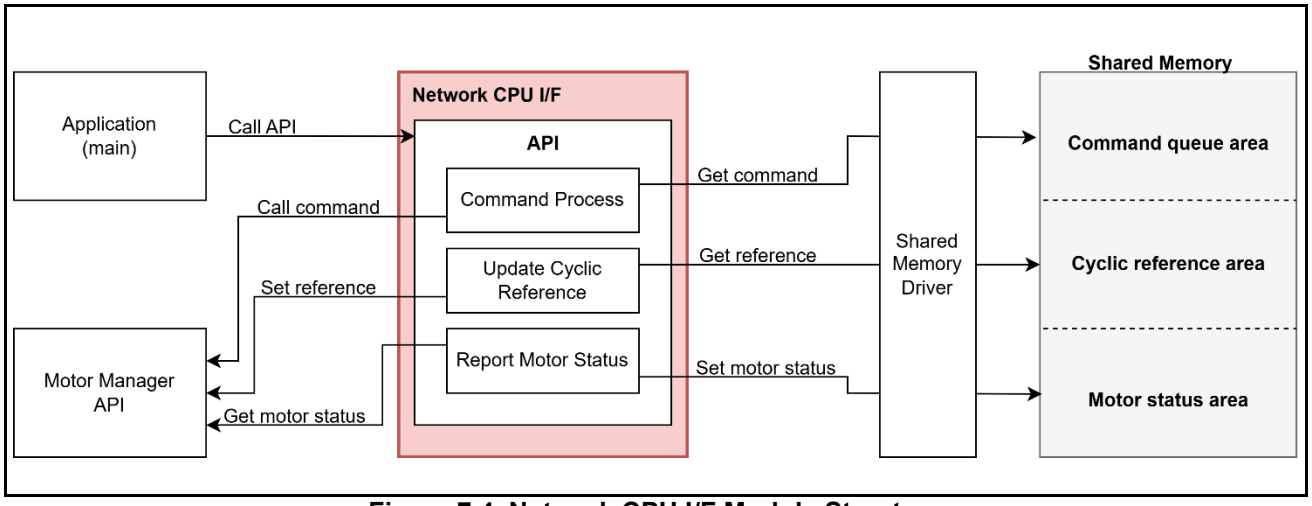


Figure 7.4. Network CPU I/F Module Structure

This module mainly provides three types of APIs:

(1) Command Process

Checks whether motor control commands are stored in the Command Queue Area of Shared Memory.

If commands exist, the corresponding Motor Manager API is called to execute the process.

(2) Update Cyclic Reference

Checks whether various reference values in the Cyclic Reference Area of Shared Memory have been updated.

If updated by the Network CPU, the corresponding Motor Manager API is called to set the new reference values.

(3) Report Motor Status

Calls the Motor Manager API to obtain various motor statuses and checks for updates.

If updated, reflects the status in the Motor Status Area of Shared Memory.

7.2.3.2 Shared Memory Allocation

Table 7.5. Shared Memory Allocation (Command queue area)

Start Address	Name	Size [bytes]	Usage
0x301A_0000	Command queue n (n = 0 … 7)	16 × 8 (=128)	Command queue
0x301A_0080	head	4	Queue head
0x301A_0084	tail	4	Queue tail

Table 7.6. Shared Memory Allocation (Cyclic reference area)

Start Address	Name	Size [bytes]	Usage
0x301A_0100	f4_ref_position	4	Reference position
0x301A_0104	f4_ref_speed	4	Reference speed
0x301A_0108	f4_ref_torque	4	Reference torque

Table 7.7. Shared Memory Allocation (Motor status area)

Start Address	Name	Size [bytes]	Usage
0x301A_0180	f4_pos_deg	4	Current position
0x301A_0184	f4_speed	4	Current speed
0x301A_0188	u1_status	1	State machine: 0x00: STOP 0x01: RUN 0x02: ERROR
0x301A_0189	reserved1	3	Unused
0x301A_018C	u2_error_status	2	Error status: 0x0000: No error 0x0001: HW overcurrent error 0x0002: Overvoltage error 0x0004: Overspeed error 0x0080: Undervoltage error 0x0100: SW overcurrent error 0x0200: Inverter overheat error 0xFFFF: Unknown error
0x301A_018E	reserved2	2	Unused
0x301A_0190	u1_ctrl_loop_mode	1	LOOP MODE: 0: d-axis current control 1: q-axis current control 2: Speed control 3: Position control
0x301A_0191	reserved3	3	Unused
0x301A_0194	u1_in_position	1	Position control completion: 0: Not completed 1: Completed
0x301A_0195	reserved4	3	Unused
0x301A_0198	u1_offset_cal_finished_flag	1	Offset measurement status: 0: Measuring 1: Completed

0x301A_0199	reserved5	3	Unused

7.2.3.3 API

Table 7.8. Network CPU I/F API List

API	Description
R_NETWORK_CPU_IF_Open	Initializes the Network CPU I/F and the Shared Memory Driver. Flowchart: Figure 7.5
R_NETWORK_CPU_IF_Close	Terminates the Network CPU I/F and the Shared Memory Driver. Flowchart: Figure 7.6
R_NETWORK_CPU_IF_CommandProcess	Checks if motor control commands are stored in the Command Queue Area of Shared Memory. If commands exist, calls the corresponding Motor Manager API to execute the process. Flowchart: Figure 7.7
R_NETWORK_CPU_IF_UpdateCyclicReference	Checks if reference values in the Cyclic Reference Area of Shared Memory have been updated. If updated by the Network CPU, calls the Motor Manager API to set new reference values. Flowchart: Figure 7.8
R_NETWORK_CPU_IF_ReportMotorStatus	Calls the Motor Manager API to obtain motor status and checks for updates. If updated, reflects the status in the Motor Status Area of Shared Memory. Flowchart: Figure 7.9

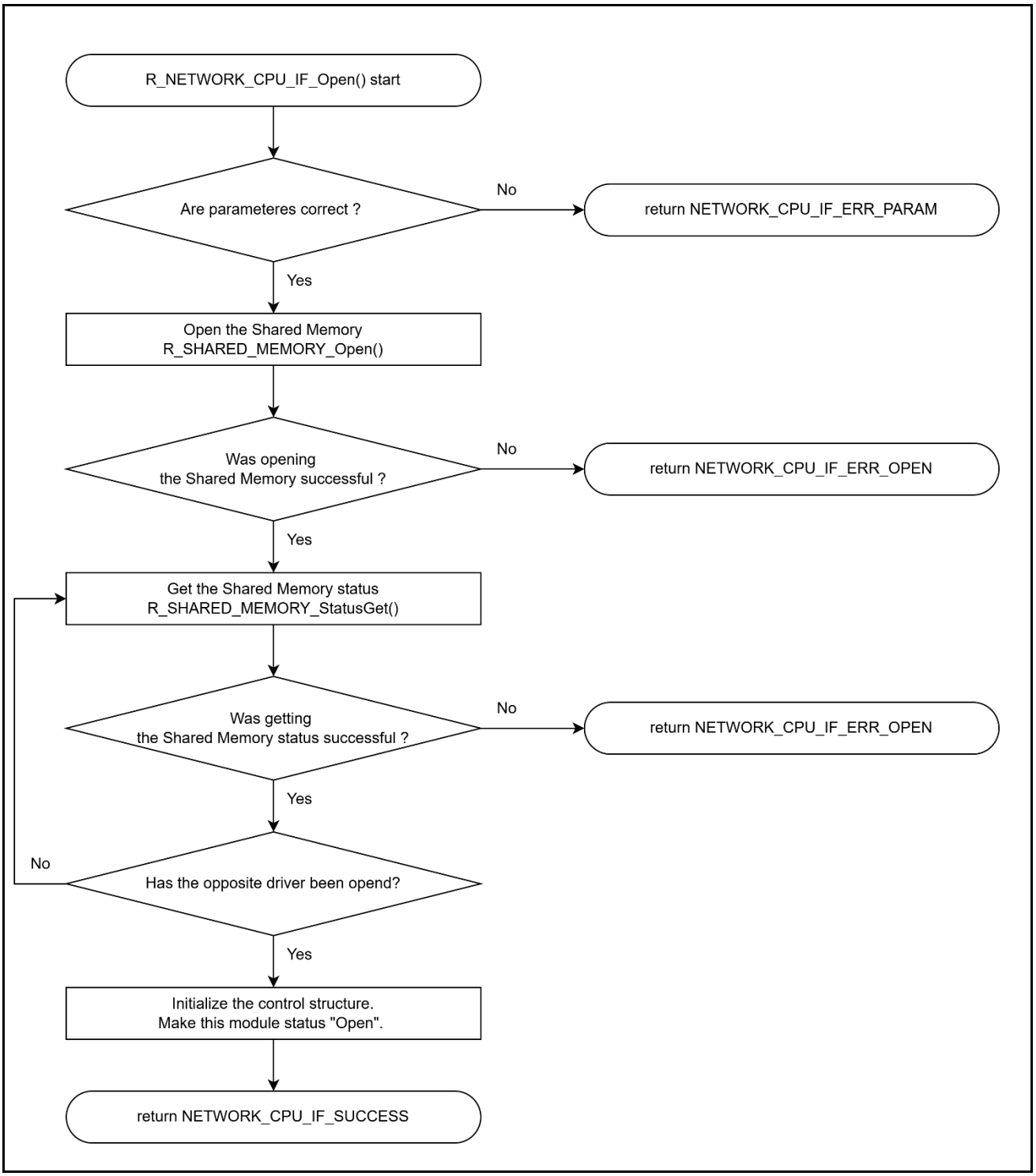


Figure 7.5. R_NETWORK_CPU_IF_Open function flowchart

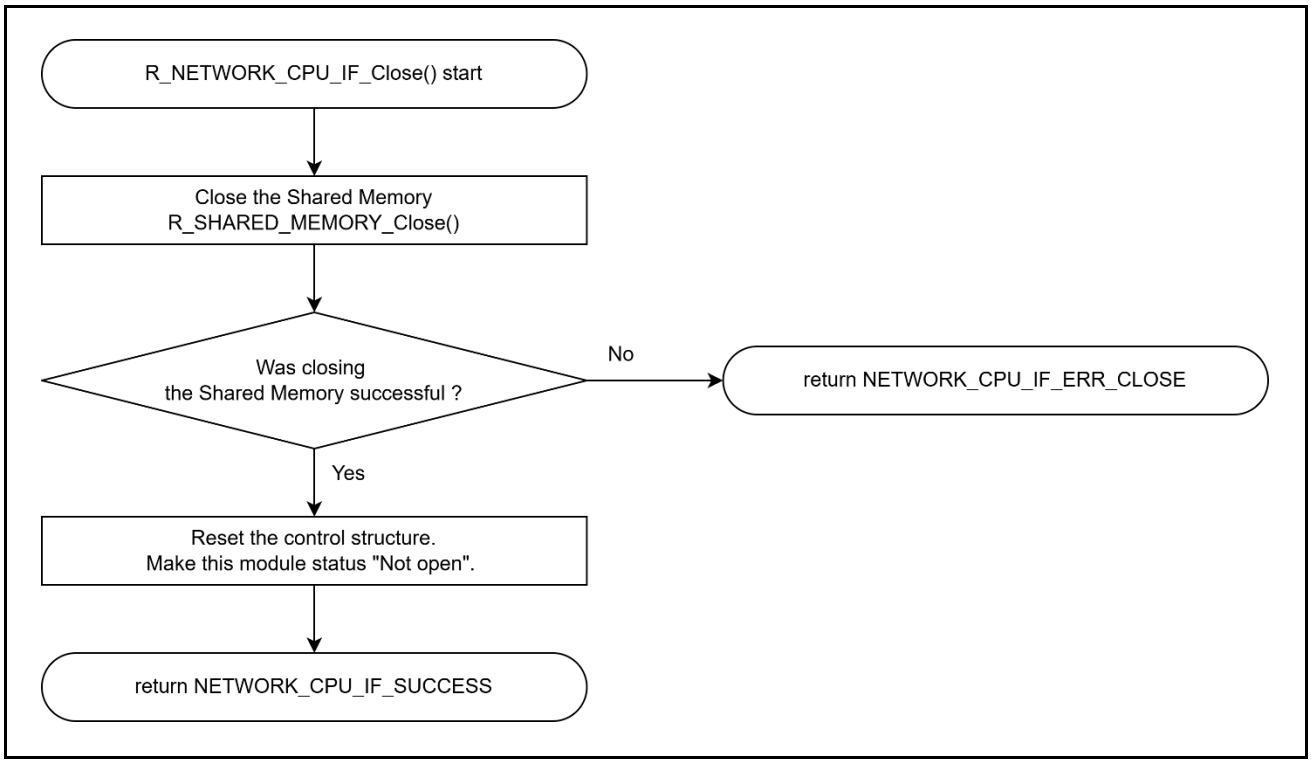


Figure 7.6. R_NETWORK_CPU_IF_Close function flowchart

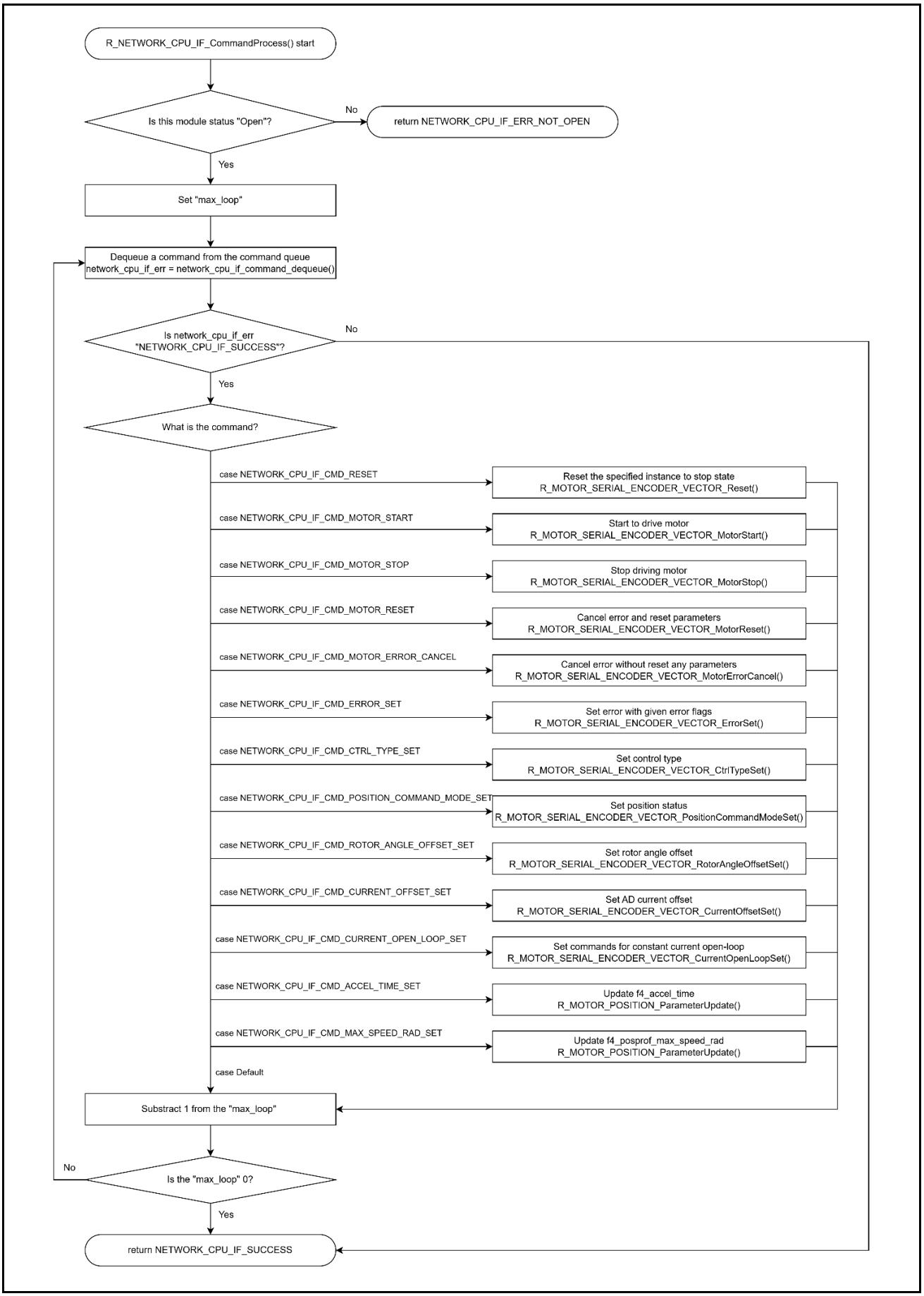


Figure 7.7. R_NETWORK_CPU_IF_CommandProcess function flowchart

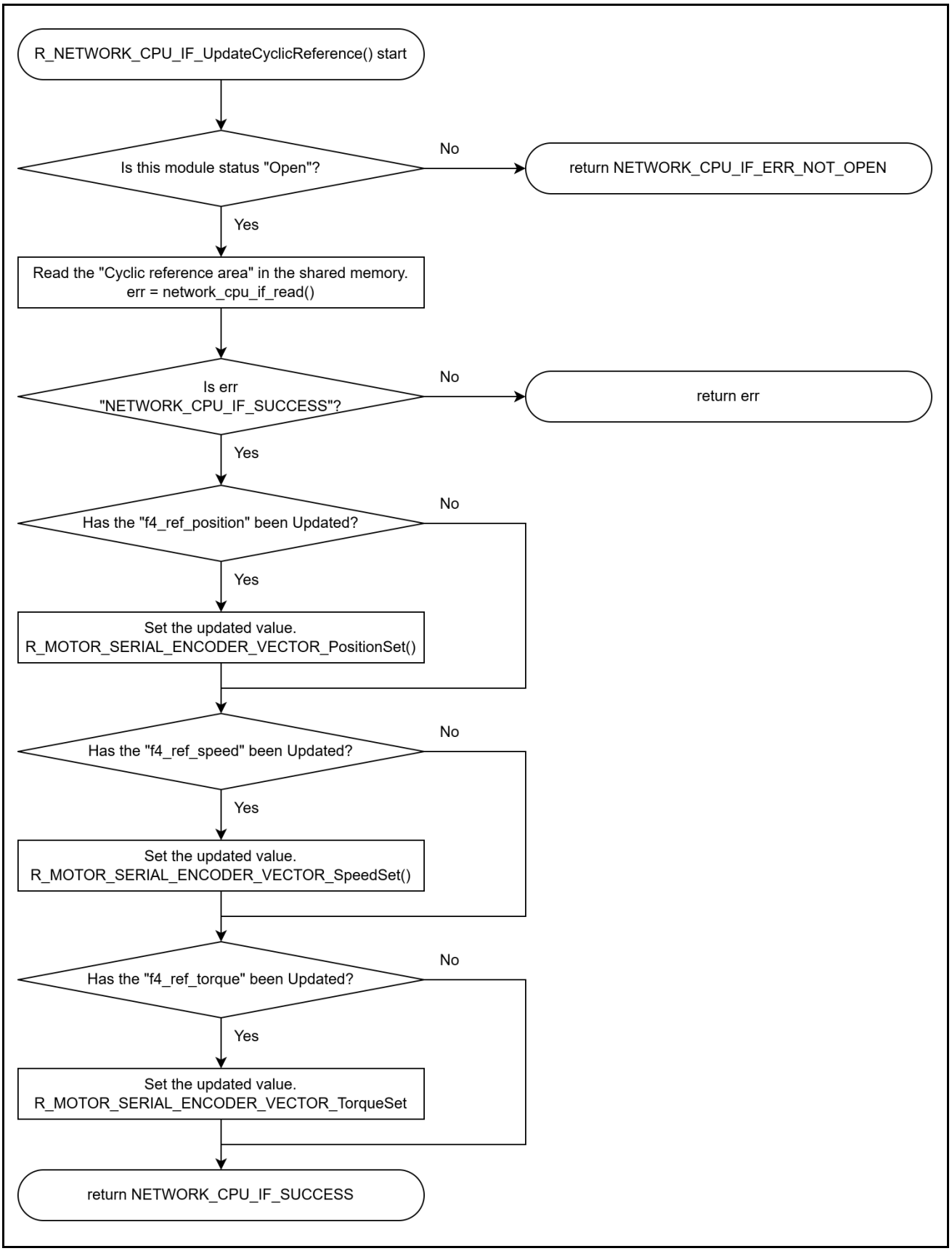


Figure 7.8. R_NETWORK_CPU_IF_UpdateCyclicReference function flowchart

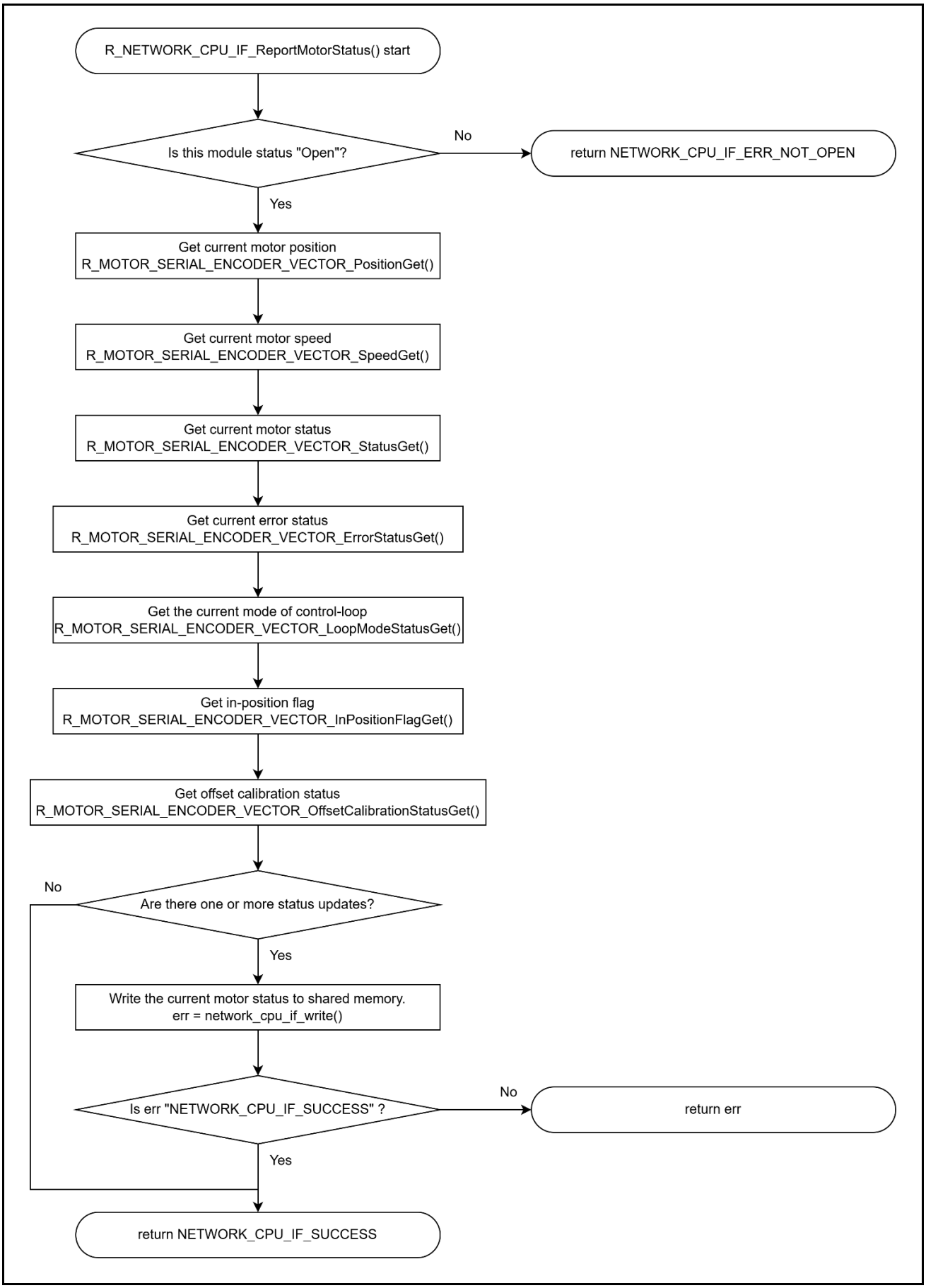


Figure 7.9. R_NETWORK_CPU_IF_ReportMotorStatus function flowchart

7.2.3.4 Private Functions

Table 7.9. Network CPU I/F Private Functions List

Private Functions	Description
network_cpu_if_write	Writes data to Shared Memory. Flowchart: Figure 7.10
network_cpu_if_read	Reads data from Shared Memory. Flowchart: Figure 7.11
network_cpu_if_command_dequeue	Reads commands from the Command Queue. Flowchart: Figure 7.12

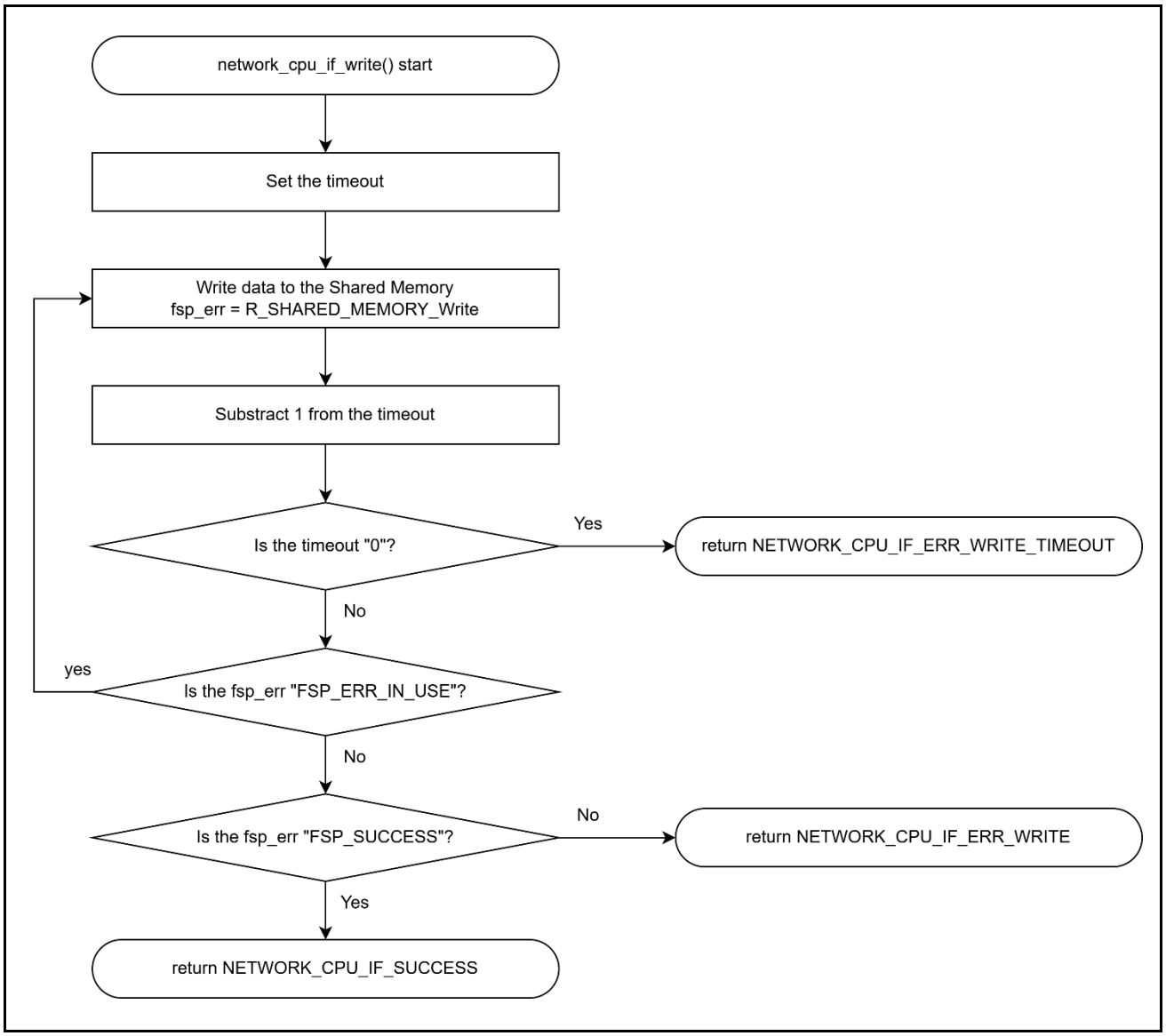


Figure 7.10. network_cpu_if_write function flowchart

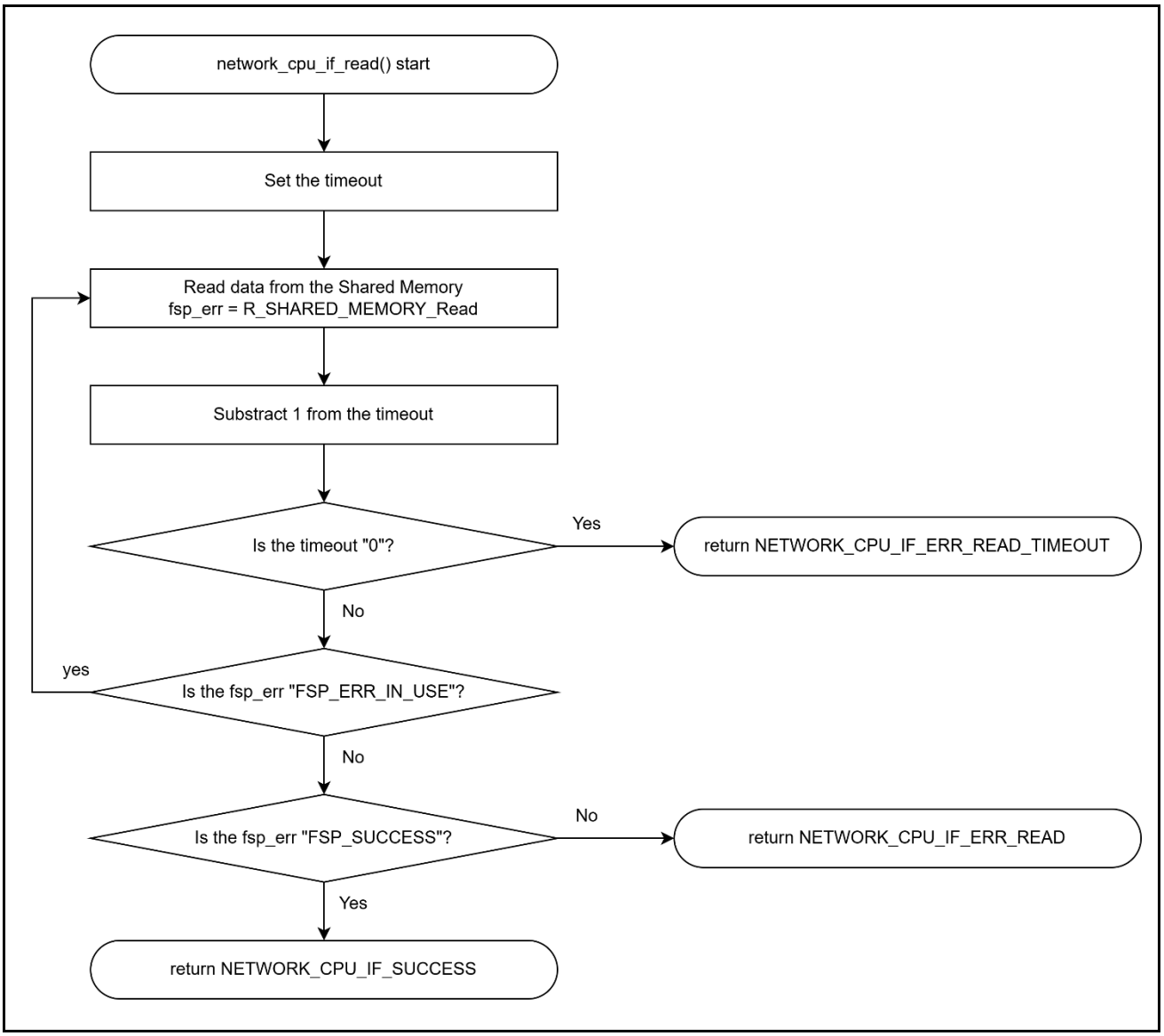


Figure 7.11. network_cpu_if_read function flowchart

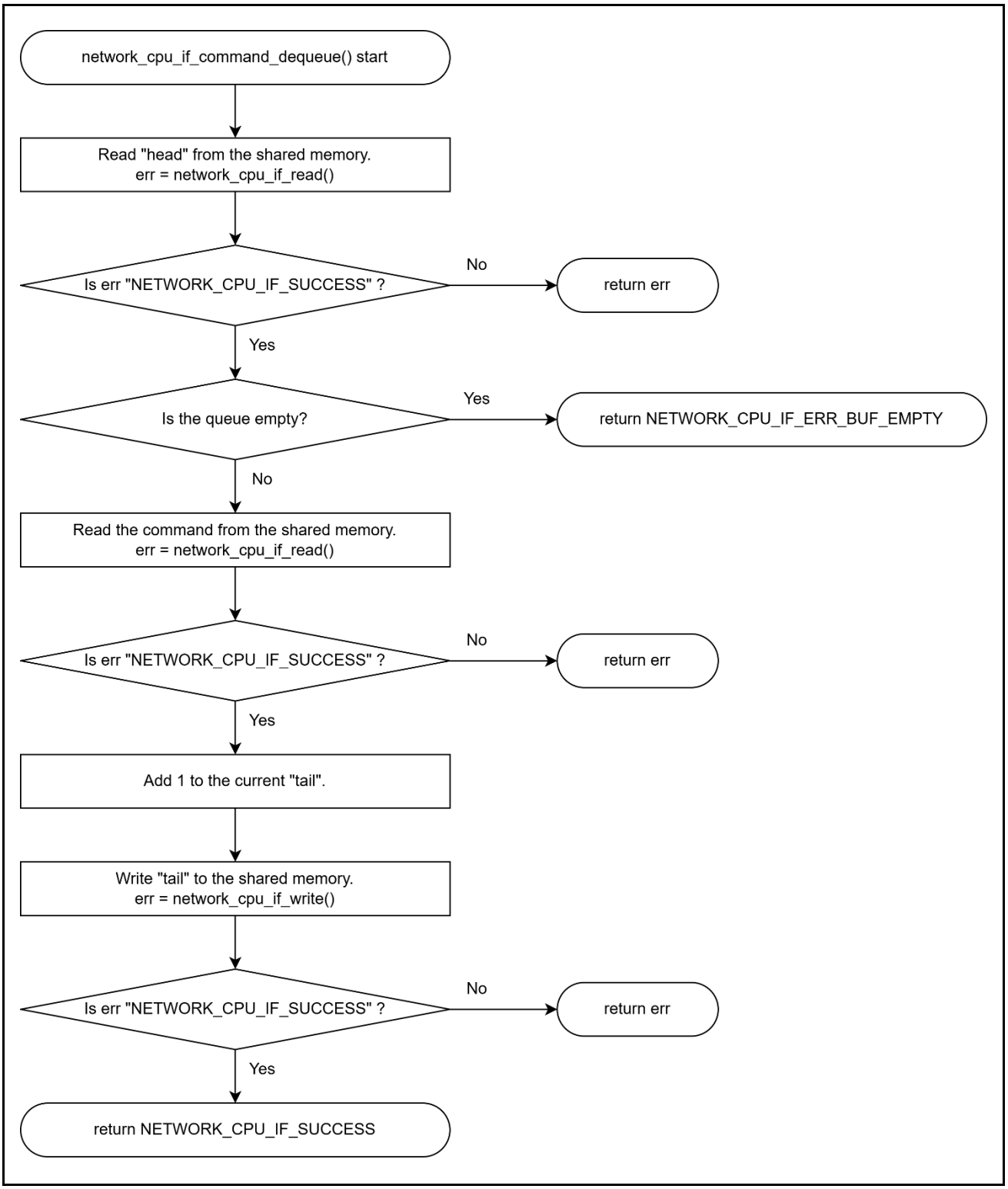


Figure 7.12. network_cpu_if_ command_dequeue function flowchart

7.2.3.5 Callback Function

Table 7.10. Network CPU I/F Callback Functions List (r_network_cpu_if.c)

Private Functions	Description
shared_memory_callback	Callback function for the Shared Memory Driver. This interrupt occurs when RZT2M CPU1 writes to Shared Memory. No processing is performed inside the callback function. Consider adding processing if necessary.

7.2.3.6 Macro Definitions

Table 7.11. Network CPU I/F Macro Definitions List（r_network_cpu_if.h）

Macro Name	Value	Remarks
CFG_CMD_Q_NUM	8	Number of command queues (must be a power of 2)
CFG_CMD_Q_OFFSET	0x0000	Offset address of Command Queue Area
CFG_CYCLIC_REF_AREA_OFFSET	0x0100	Offset address of Cyclic Reference Area
CFG_MTR_STATUS_AREA_OFFSET	0x0180	Offset address of Motor Status Area
CFG_NETWORK_CPU_IF_TIMEOUT	100	Timeout value for Shared Memory read/write
CFG_SHARED_MEMORY_INSTANCE_ADDR	&g_shared_memory0	Pointer to Shared Memory Driver instance
CFG_MOTOR_CTRL_INSTANCE_ADDR	&g_st_serial_encoder_vector	Pointer to Motor Control instance
NETWORK_CPU_IF_CMD_DATA_MAX_BYTE	12	Maximum data size for command queue

7.2.3.7 Enumerations

Table 7.12. Network CPU I/F Enumerations List（r_network_cpu_if.h）

Enumerations	Definitions	Remarks
network_cpu_if_err_t Network CPU I/F Error Codes	NETWORK_CPU_IF_SUCCESS	No error
	NETWORK_CPU_IF_ERR_OPEN	Open error
	NETWORK_CPU_IF_ERR_CLOSE	Close error
	NETWORK_CPU_IF_ERR_PARAM	Parameter error
	NETWORK_CPU_IF_ERR_NOT_OPEN	Not open error
	NETWORK_CPU_IF_ERR_BUF_FULL	Command queue full
	NETWORK_CPU_IF_ERR_BUF_EMPTY	Command queue empty
	NETWORK_CPU_IF_ERR_WRITE	Write error
	NETWORK_CPU_IF_ERR_WRITE_TIMEOUT	Write timeout
	NETWORK_CPU_IF_ERR_READ	Read error
	NETWORK_CPU_IF_ERR_READ_TIMEOUT	Read timeout
	NETWORK_CPU_IF_ERR_EXCEED_PAYLOAD	Data exceeded

Enumerations	Definitions	Remarks
network_cpu_if_status_t Network CPU I/F Status	NETWORK_CPU_IF_STATUS_NOT_OPEN	Not open
	NETWORK_CPU_IF_STATUS_OPEN	Open
network_cpu_if_cmd_t Network CPU I/F Command Definitions	NETWORK_CPU_IF_CMD_NONE	No command
	NETWORK_CPU_IF_CMD_RESET	Commands used in CommandProcess API (See Figure 7.6)
	NETWORK_CPU_IF_CMD_MOTOR_START	
	NETWORK_CPU_IF_CMD_MOTOR_STOP	
	NETWORK_CPU_IF_CMD_MOTOR_RESET	
	NETWORK_CPU_IF_CMD_MOTOR_ERROR_CANCEL	
	NETWORK_CPU_IF_CMD_ERROR_SET	
	NETWORK_CPU_IF_CMD_CTRL_TYPE_SET	
	NETWORK_CPU_IF_CMD_POSITION_COMMAND_MODE_SET	
	NETWORK_CPU_IF_CMD_ROTOR_ANGLE_OFFSET_SET	
	NETWORK_CPU_IF_CMD_CURRENT_OFFSET_SET	
	NETWORK_CPU_IF_CMD_CURRENT_OPEN_LOOP_SET	
	NETWORK_CPU_IF_CMD_ACCEL_TIME_SET	
	NETWORK_CPU_IF_CMD_MAX_SPEED_RAD_SET	

7.2.3.8 Structure and Variable Information

Table 7.13. Network CPU I/F Structure Definitions (r_network_cpu_if.h)

Structure	Variable	Description
st_current_offset_cfg_t Current offset configuration structure	f4_iu_offset	U-phase current offset
	f4_iv_offset	V-phase current offset
	f4_iw_offset	W-phase current offset
st_current_open_loop_cfg_t Open loop current configuration structure	f4_open_loop_current	Open loop current value
	f4_ref_speed_rpm	Reference speed (RPM)
network_cpu_if_cmd_hdr_t command queue head/tail structure	head	Head of command queue
	tail	Tail of command queue
network_cpu_if_cmd_q_t command queue structure	cmd	Command
	size	Data size
	data[NETWORK_CPU_IF_CMD_DATA_MAX_BYTE]	Data array
network_cpu_if_cyclic_ref_t	f4_ref_position	Reference position

Cyclic Reference structure	f4_ref_speed	Reference speed
	f4_ref_torque	Reference torque
network_cpu_if_mtr_status_t Motor Status structure	f4_pos_deg	Current position
	f4_speed	Current speed
	u1_status	State machine
	reserved1	Unused area
	u2_error_status	Error status
	reserved2	Unused area
	u1_ctrl_loop_mode	LOOP MODE
	reserved3	Unused area
	u1_in_position	Position control completion status
	reserved4	Unused area
	u1_offset_cal_finished_flag	Offset measurement status
	reserved5	Unused area
network_cpu_if_cfg_t Network CPU I/F Configuration structure	p_shared_memory_instance	Pointer to Shared Memory instance
	timeout	Timeout value
	shr_mem_cmd_data_max_size	Maximum command data size
	shr_mem_cmd_q_num	Number of command queues
	shr_mem_cmd_q_mask	Command queue mask
	shr_mem_cmd_q_offset	Offset of Command Queue Area in Shared Memory
	shr_mem_cmd_q_size	Size of Command Queue Area in Shared Memory
	shr_mem_cmd_head_offset	Offset of command queue head area
	shr_mem_cmd_head_size	Size of command queue head area
	shr_mem_cmd_tail_offset	Offset of command queue tail area
	shr_mem_cmd_tail_size	Size of command queue tail area
	shr_mem_cyclic_ref_area_offset	Offset of Cyclic Reference Area
	shr_mem_cyclic_ref_area_size	Size of Cyclic Reference Area
	shr_mem_ref_pos_offset	Offset of reference position area
	shr_mem_ref_vel_offset	Offset of reference speed area
	shr_mem_ref_tor_offset	Offset of reference torque area
	shr_mem_mtr_status_area_offset	Offset of Motor Status Area
	shr_mem_mtr_status_area_size	Size of Motor Status Area
	shr_mem_mtr_pos_offset	Offset of current position area
	shr_mem_mtr_vel_offset	Offset of current speed area
	shr_mem_mtr_status_offset	Offset of state machine area
	shr_mem_mtr_err_status_offset	Offset of error status area

	shr_mem_mtr_ctrl_loop_mode_offset	Offset of LOOP MODE area
	shr_mem_mtr_in_pos_offset	Offset of position control completion area
	shr_mem_mtr_cal_finished_flag_offset	Offset of offset measurement status area
	max_loop	Maximum loop count used by CommandProcess API
	p_st_serial_encoder_vector_control	Pointer to serial encoder vector control structure
network_cpu_if_ctrl_t Network CPU I/F Control structure	status	Network CPU I/F status
	cmd_hdr	Instance of command queue head/tail structure
	timeout_count	Timeout occurrence count
	position_offset	Position offset (unused)
	cyclic_ref_buf	Buffer for Cyclic Reference area
	mtr_status_buf	Buffer for Motor Status area
	p_cfg	Pointer to Network CPU I/F configuration structure
network_cpu_if_api_t Network CPU I/F API structure	Open	Open function
	Close	Close function
	CommandProcess	Command processing function
	UpdateCyclicReference	Cyclic reference update function
	ReportMotorStatus	Motor status reporting function

7.2.3.9 Global Variable Information

Table 7.14. Network CPU I/F Global Variables (r_network_cpu_if.c)

Type	Instance Name	Description
network_cpu_if_ctrl_t	g_network_cpu_if_ctrl	Network CPU I/F control structure instance. Initialized by the Open function.
network_cpu_if_cfg_t	g_network_cpu_if_cfg	Network CPU I/F configuration structure instance. Each member is initialized at instance creation. Values cannot be changed after initialization.
network_cpu_if_api_t	g_network_cpu_if_api	Network CPU I/F API structure instance. Each member is initialized at instance creation. Values cannot be changed after initialization.

7.3 EtherCAT (CiA402) (RZ/T2M CPU1)

7.3.1 Basic Specifications

Table 7.15. RZ/T2M CPU1 Basic Specifications

Function	Item	Description
EtherCAT SubDevice Stack	Interface with PHY	MII
	MDIO type	GMAC
	Number of Communication Ports	2 ports (In, Out)
	EtherCAT LED	4 types: RUN, ERR, L/A IN, L/A OUT
	EtherCAT Device ID	Configurable via SW12 (8-bit)
	Device Profile	CiA402 Drive Profile, FSoE Safety Drive Connection
	Mailbox	Supports CoE
	Synchronization Mode	Supports 2 types: - DC Mode (DC-Synchronous with SM2 event) - SM Mode (SM-synchronous)
	DC Mode Cycle Time	Minimum 50 µs
	Interrupt Priority (PDI, Sync0, Sync1)	Priority 4
	Interrupt Priority (Timer)	Priority 6
CiA402 Drive Profile	Mode	Supports 4 types: - Profile Position Mode - Homing Mode - Cyclic Synchronous Position Mode - Cyclic Synchronous Velocity Mode
	Number of Motor Axes	1 axis
Fusa CPU Communication	Role	Sends Safety Master PDU received from FSoE Master to Functional Safety CPU (RZ/T2L-A) Sends Safety Slave PDU received from Functional Safety CPU to FSoE Master
Safety Drive	Stop Function	Compliant with IEC61800-5-2: STO, SS1-t (*1)
Motor CPU I/F	Role	Provides communication interface with Motor Control CPU
	Communication Method	Shared Memory
	Interrupt Priority (INTCPU)	Priority 6
Fusa I/F	Role	Provides communication interface with Functional Safety CPU
	Communication Method	Asynchronous Serial (UART)
	Interrupt Priority (ERI, RXI, RXI, TEI)	Priority 2

(*1) In this software, the SS1-t stop function is supported only in Cyclic Synchronous Velocity Mode.

7.3.2 Overall Software Structure

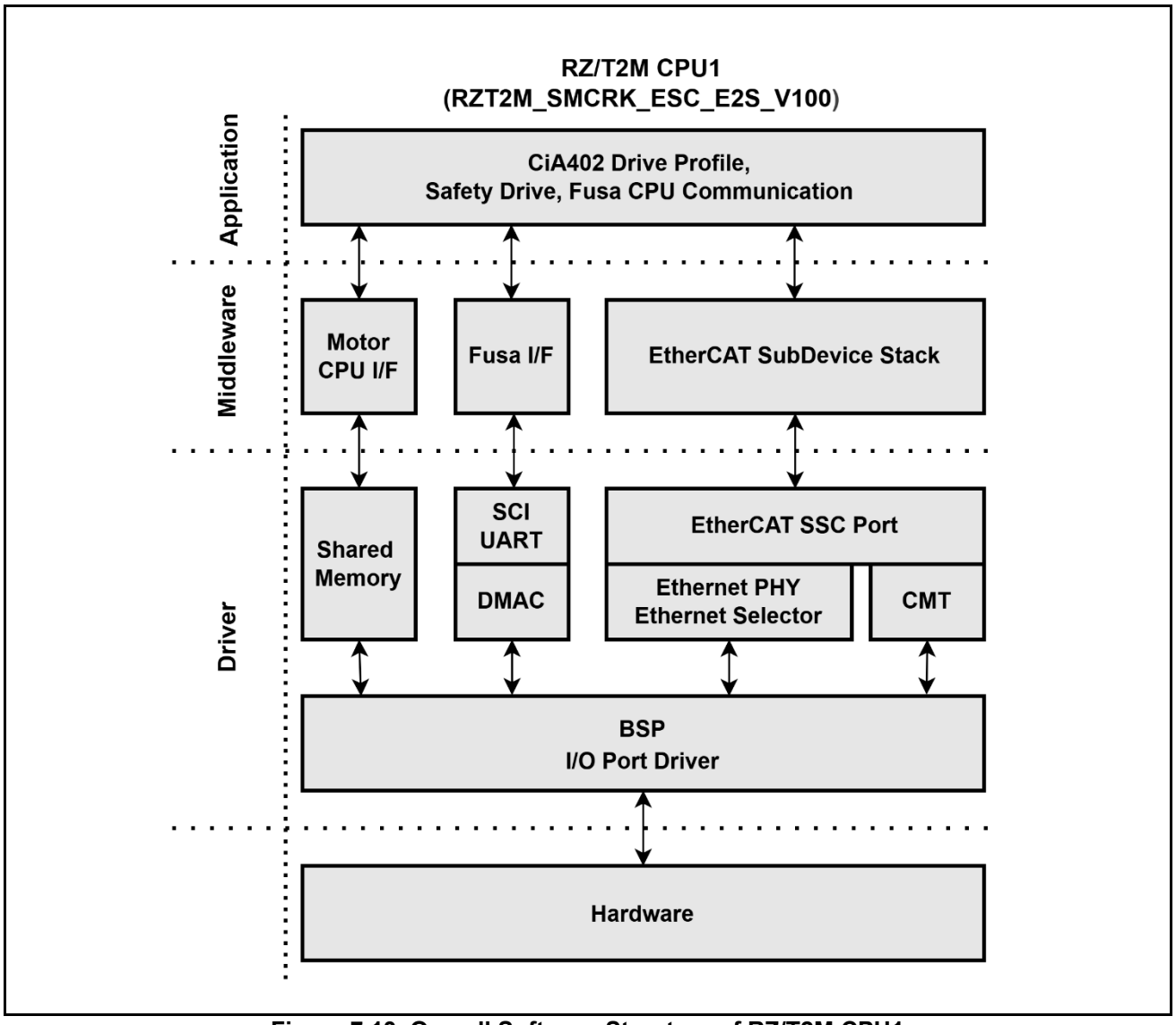


Figure 7.13. Overall Software Structure of RZ/T2M CPU1

7.3.3 src Folder Structure

Table 7.16. RZ/T2M CPU1 src Folder Structure

Folder / File				Description
src	ethercat	Beckhoff/Src	applInterface.h	EtherCAT SubDevice Stack (SSC Tool generated code)
			coeappl.c	
			coeappl.h	
			ecatappl.c	
			ecatappl.h	
			ecatcoe.c	
			ecatcoe.h	
			ecatslv.c	
			ecatslv.h	
			ecat_def.h	
			esc.h	
			mailbox.c	
			mailbox.h	
			objdef.c	
			objdef.h	
			sdoserv.c	
			sdoserv.h	
			cia402appl.c	CiA402 Drive Profile FSoE Safety Drive Connection (SSC Tool generated code)
			cia402appl.h	
		renesas	samplecia402.c	CiA402 Drive Profile Safety Drive Fusa CPU Communication (Renesas developed code)
			samplecia402.h	
	fusa_if			Fusa I/F Module
	motor_cpu_if			Motor CPU I/F Module
	hal_entry.c			hal_entry function

7.3.4 hal_entry and Main Processing

The flowchart of the hal_entry function is shown in Figure 7-13.

Additionally, the functions called within hal_entry are listed in Table 7-17.

Table 7.17. Functions Called by hal_entry

Function Name	Description
R_MOTOR_CPU_IF_Open	Open function for Motor CPU I/F. See Section 7.3.8 for details.
R_FUSA_IF_Open	Open function for FuSa I/F. See Section 7.3.9 for details.
RM_ETHERCAT_SSC_PORT_Open	Open function for EtherCAT SSC Port provided by FSP. Refer to *RZT Flexible Software Package Documentation* for details.
change_phy_reg	Function to modify PHY registers. Changes the value of “GPIO Control 2 register” for each PHY (VSC8541) to **0x2000**.
ecat_ssc_init	Function that performs initialization of EtherCAT SubDevice Stack Code and CiA402 Drive Profile. The flowchart is shown in Figure 7.14. For details of functions and variables in the flowchart, refer to *Application Note ET9300 (EtherCAT SubDevice Stack Code)*.
R_BSP_PinSet	Function provided by FSP. Turns on LED11. Refer to *RZT Flexible Software Package Documentation* for details.
MainLoop	Main processing function for EtherCAT SubDevice Stack Code and CiA402 Drive Profile. Refer to *Application Note ET9300 (EtherCAT SubDevice Stack Code)* for details.

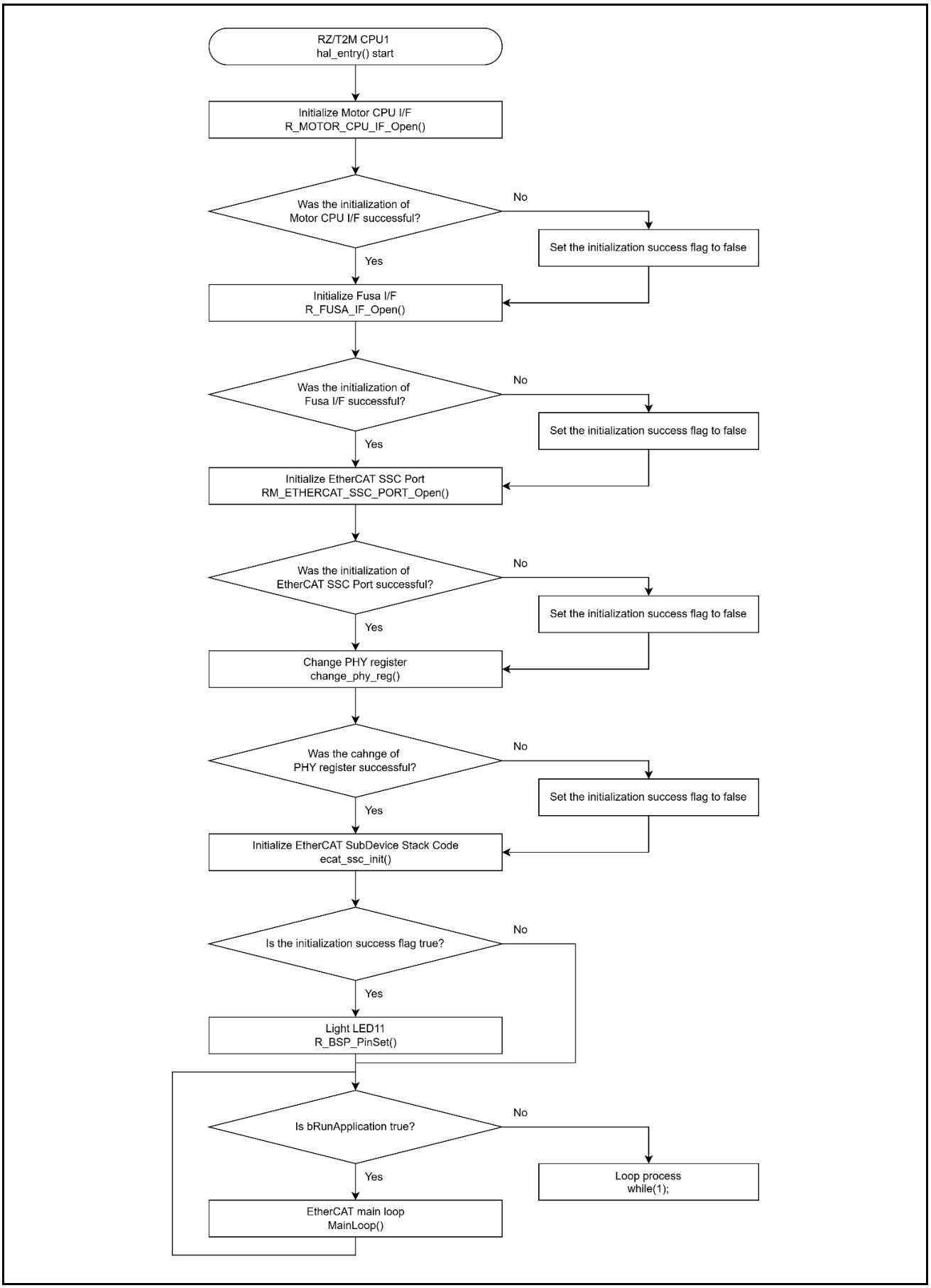


Figure 7.14. RZ/T2M CPU1 hal_entry function flowchart

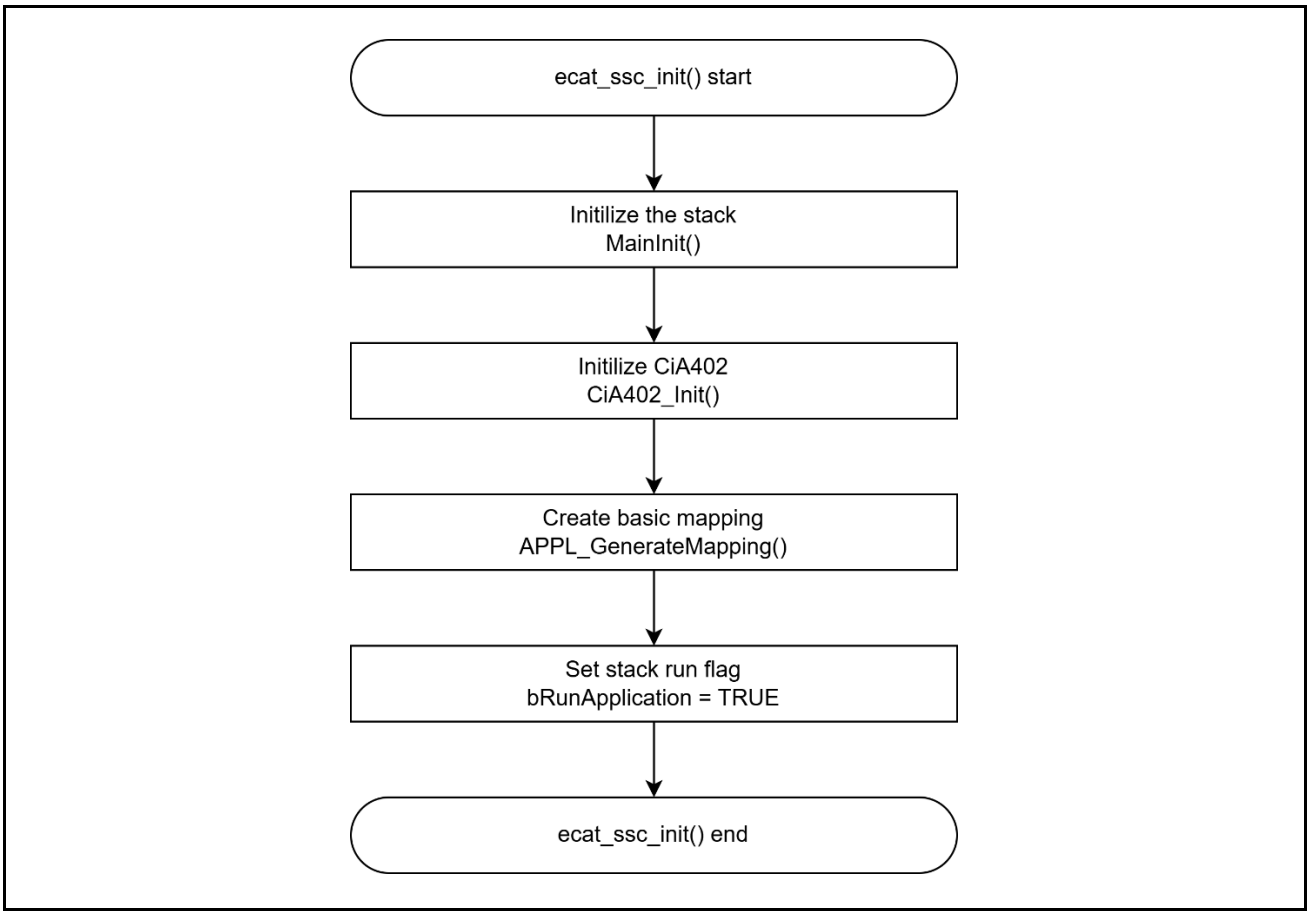


Figure 7.15. ecat_ssc_init function flowchart

7.3.5 CiA402 Drive Profile

The CiA402 Drive Profile is a device profile for drives and motion control, primarily defining functional behavior for servo drives, sinusoidal inverters, and stepper motor controllers.

This profile specifies multiple operating modes and corresponding configuration parameters as an Object Dictionary.

It also includes a Finite State Automaton (FSA) that defines internal and external behavior for each state.

To change states, the Control Word object is used, and the result after transition is reflected in the Status Word object.

Control Word and various command values (e.g., speed) are assigned to RxPDO, while Status Word and actual values (e.g., position) are assigned to TxPDO.

For details, refer to the CiA402 specification document.

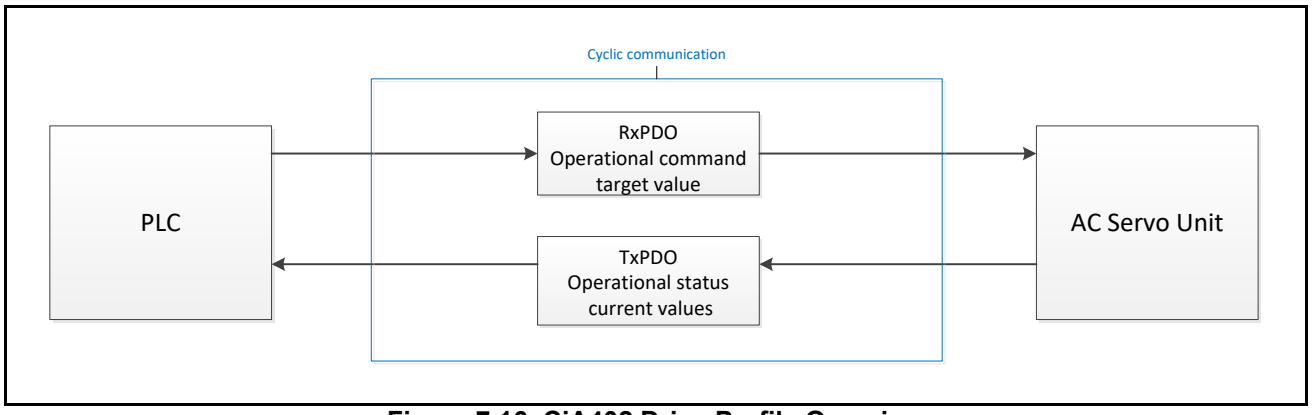


Figure 7.16. CiA402 Drive Profile Overview

7.3.5.1 CiA402 Operating Modes

Among the operating modes defined by CiA402, this program supports the following:

Table 7.18. Supported Operating Modes

Operation Mode	Support	Refer for details
Profile position mode	Available	7.3.5.4
Velocity mode (frequency converter)	Not available	-
Profile velocity mode	Not available	-
Profile torque mode	Not available	-
Homing mode	Available	7.3.5.5
Interpolated position mode	Not available	-
Cyclic synchronous position mode	Available	7.3.5.2
Cyclic synchronous velocity mode	Available	7.3.5.3
Cyclic synchronous torque mode	Not available	-
Cyclic synchronous torque mode with commutation angle	Not available	-
Manufacturer specific mode	Not available	-

7.3.5.2 Cyclic Synchronous Position Mode (CSP)

In this mode, the target position is periodically transmitted for motor position control.

Table 7.19. CSP Mode RxPDO List

RxPDO	Index	Description
Control Word	0x6040	CiA402 control command
Target Position	0x607A	Position command value [deg] (*1)

(*1) The command value sent to the motor is converted to **1/10** of the input value.
Example: Input 1234 → Motor command value = 123.4.

Table 7.20. CSP Mode TxPDO List

TxPDO	Index	Description
Error Code	0x603F	CiA402 error code
Status Word	0x6041	CiA402 state
Modes of Operation Display	0x6061	Mode type (=8)
Position Actual Value	0x6064	Motor current position [deg] (*2)
Velocity Actual Value	0x606C	Motor current speed [rpm] (*2)

(*2) Actual values are converted by multiplying by **10**. Example: Output 5678 → Actual value = 567.8.

Table 7.21. CSP Mode Motor Settings

Item	Setting
OOP MODE	Position control
Position command generation mode	Trapezoidal drive motion

7.3.5.3 Cyclic synchronous velocity mode (CSV)

In this mode, the target velocity is periodically transmitted for motor speed control.

Table 7.22. CSV Mode RxPDO List

RxPDO	Index	Description
Control Word	0x6040	CiA402 control command
Target Velocity	0x60FF	Velocity command value [rpm] (*1)

(*1) The command value sent to the motor is converted to **1/10** of the input value.
Example: Input 1234 → Motor command value = 123.4.

Table 7.23. CSV Mode TxPDO List

TxPDO	Index	Description
Error Code	0x603F	CiA402 error code
Status Word	0x6041	CiA402 state
Modes of Operation Display	0x6061	Mode type (=9)
Position Actual Value	0x6064	Motor current position [deg] (*2)
Velocity Actual Value	0x606C	Motor current speed [rpm] (*2)

(*2) Actual values are converted by multiplying by **10**. Example: Output 5678 → Actual value = 567.8.

Table 7.24. CSV Mode Motor Settings

Item	Setting
LOOP MODE	Speed control

7.3.5.4 Profile position mode (PP)

In this mode, speed and acceleration are set for motor position control.

Table 7.25. PP Mode RxPDO List

RxPDO	Index	Description
Control Word	0x6040	CiA402 control command
Target Position	0x607A	Position command value [deg] (*1)
Profile Velocity	0x6081	Profile velocity [rpm] (*1)
Profile Acceleration	0x6083	Profile acceleration [rpm/s] (*1)
Profile Deceleration	0x6084	Profile deceleration [rpm/s] (*3)

(*1) The command value sent to the motor is converted to **1/10** of the input value.
Example: Input 1234 → Motor command value = 123.4.

(*3) Not used in this software.

Table 7.26. PP Mode TxPDO List

TxPDO	Index	Description
Error Code	0x603F	CiA402 error code
Status Word	0x6041	CiA402 state
Modes of Operation Display	0x6061	Mode type (=1)
Position Actual Value	0x6064	Motor current position [deg] (*2)
Velocity Actual Value	0x606C	Motor current speed [rpm] (*2)

(*2) Actual values are converted by multiplying by **10**. Example: Output 5678 → Actual value = 567.8.

Table 7.27. PP Mode Motor Settings

Item	Setting
LOOP MODE	Position control
Position command generation mode	Trapezoidal drive motion
Accel/Decel time [s]	(Profile Velocity) / (Profile Acceleration)
Max trapezoidal speed [rpm]	(Profile Velocity) / 10

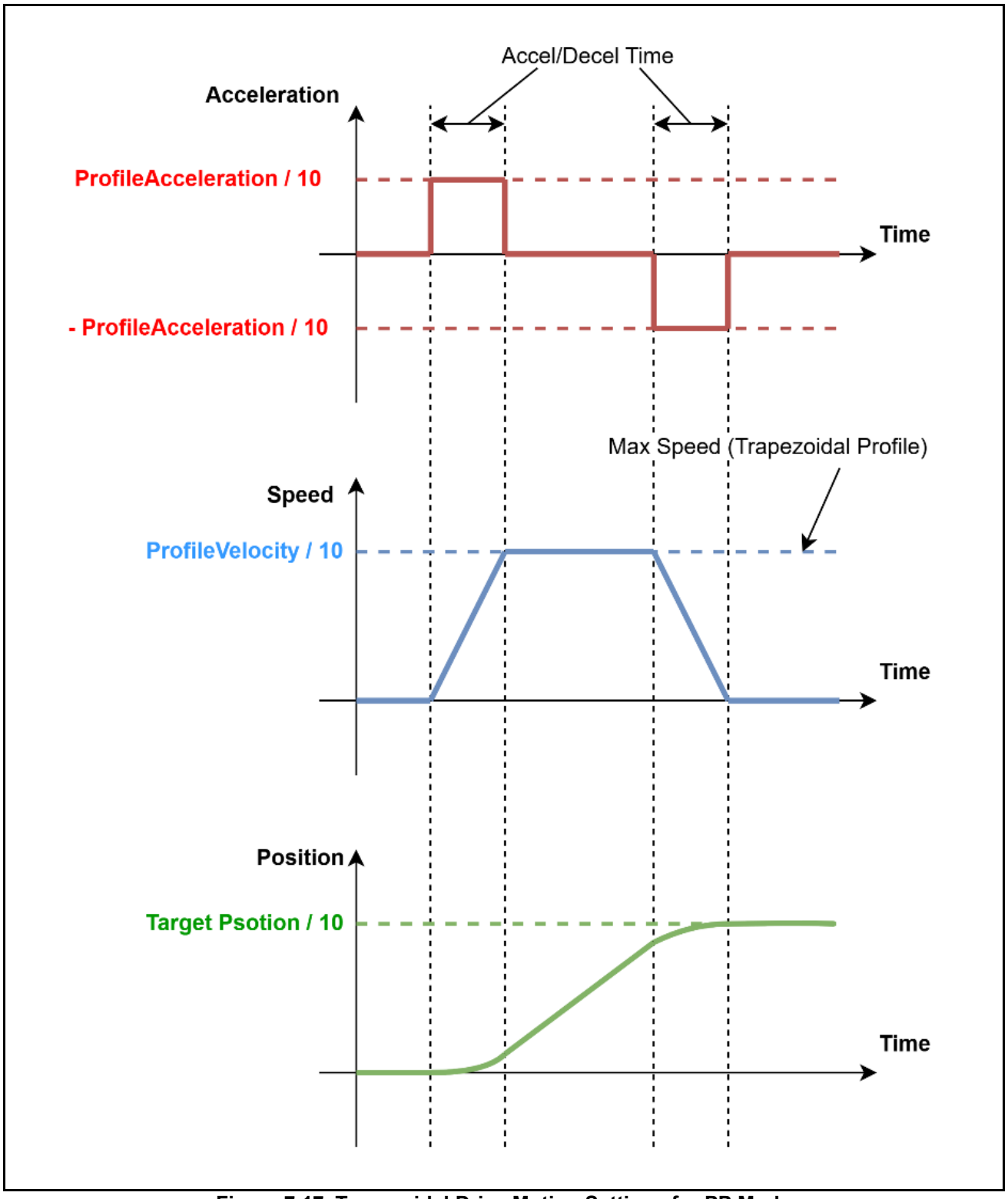


Figure 7.17. Trapezoidal Drive Motion Settings for PP Mode

7.3.5.5 Homing Mode (HM)

In this mode, Homing Method 37 is used to change the motor's current position.

Table 7.28.: Homing Mode RxPDO List

RxPDO	Index	Description
Control Word	0x6040	CiA402 control command
Home Offset	0x607C	Commanded current position [deg] (*1)
Home Method	0x6098	Homing method
Homing Speed For Switch	0x6099	Switch homing speed (*3)
Homing Speed For Zero	0x6099	Zero homing speed (*3)
Homing Acceleration	0x609A	Overall homing accel/decel (*3)

(*1) The command value sent to the motor is converted to **1/10** of the input value.
Example: Input 1234 → Motor command value = 123.4.

(*3) Not used in this software.

Table 7.29.: Homing Mode TxPDO List

xPDO	Index	Description
Error Code	0x603F	CiA402 error code
Status Word	0x6041	CiA402 state
Modes of Operation Display	0x6061	Mode type (=6)
Position Actual Value	0x6064	Motor current position [deg] (*2)
Velocity Actual Value	0x606C	Motor current speed [rpm] (*2)

(*2) Actual values are converted by multiplying by **10**. Example: Output 5678 → Actual value = 567.8.

Table 7.30.: Homing Mode Motor Settings

Item	Setting
LOOP MODE	Position control
Position command generation mode	Trapezoidal drive motion

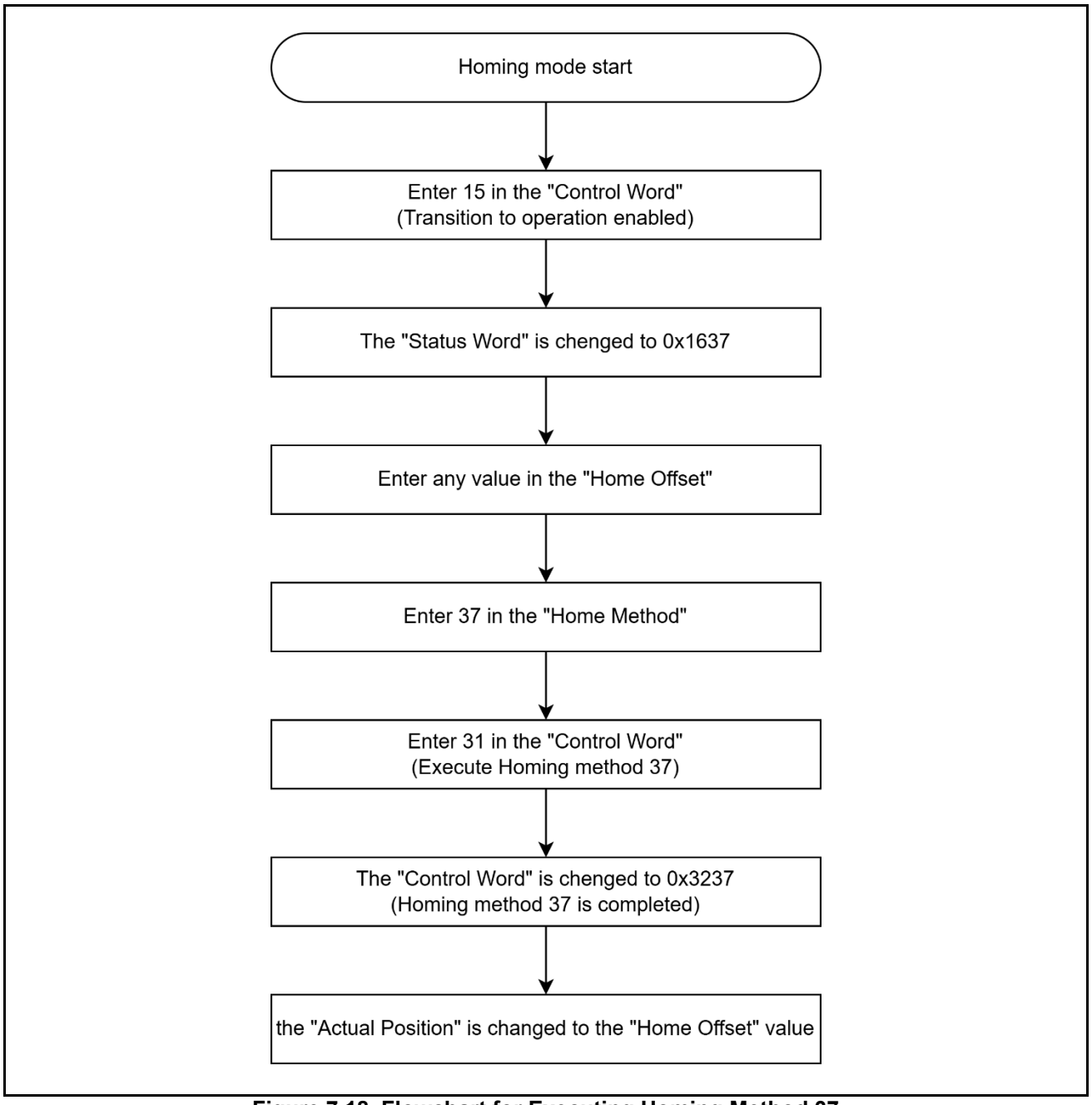


Figure 7.18. Flowchart for Executing Homing Method 37

7.3.5.6 CiA402 State Transitions

As defined by CiA402 FSA (Finite State Automaton), the sample program supports the following transitions. Table 7.31 shows the corresponding transition numbers, their triggers, and behavior.

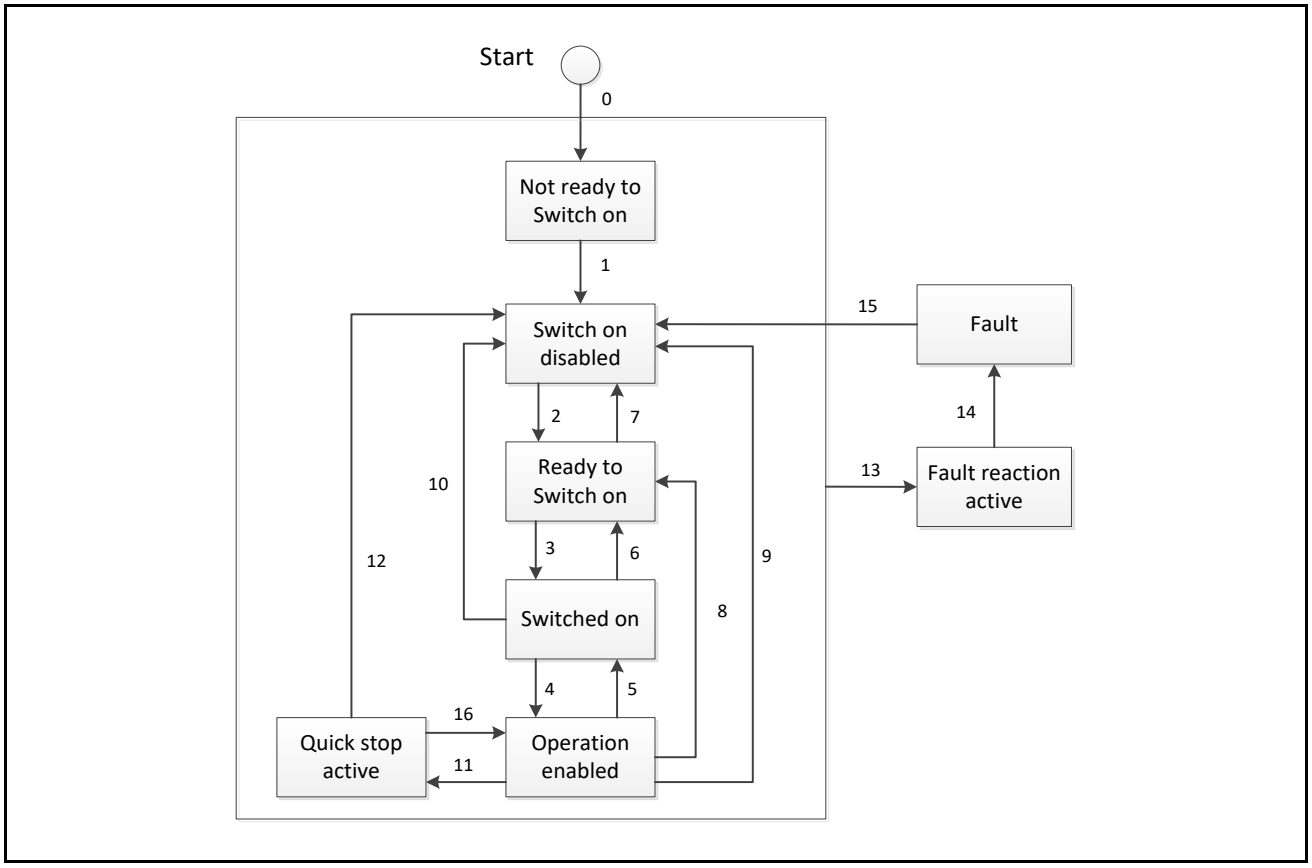


Figure 7.19. CiA402 State Transition Diagram

Table 7.31. CiA402 State Transitions

ransition No.	Trigger	Behavior
0, 1	Automatically triggered if EtherCAT State Machine (ESM) is in OP	No action
2		See Figure 7.20
3	Control Word = 7 or Control Word = 15	See Figure 7.21
4	Control Word = 15	See Figure 7.22
5	Control Word = 7	See Figure 7.23
6	Control Word = 6	No action
7	Control Word = 0	No action
8	Control Word = 6	See Figure 7.23
9	Control Word = 0	See Figure 7.23
10	Control Word = 0	No action
11	Control Word = 2	See Figure 7.23
12	Control Word = 0	No action

13	A) When executing transition 2 and a motor error occurs B) When executing transitions 3 or 16 and motor operation is prohibited C) When Operation Enabled and a motor error occurs D) When Operation Enabled and STO / SS1-t request occurs	See Figure 7.24
14	Automatically triggered when Fault Reaction Active	See Figure 7.25
15	Control Word = 128	See Figure 7.26
16	Control Word = 15	See Figure 7.27

7.3.5.7 CiA402 Error Code

When transitioning to Fault Reaction Active, the error code (0x603F) is set according to the error that occurred.

Table 7.32 lists the error codes.

Table 7.32. Error Code (0x603F) List

Error Type	Error Code	Description
Motor Error	0x2214	HW overcurrent error (0x0001) occurred
	0x3211	Overvoltage error (0x0002) occurred
	0x7310	Overspeed error (0x0004) occurred
	0x3221	Undervoltage error (0x0080) occurred
	0x2220	SW overcurrent error (0x0100) occurred
	0x4310	Overtemperature error (0x0200) occurred
	0x7120	Undefined error (0xFFFF) occurred
Motor Stop Request	0xFF00	STO request occurred
	0xFF01	SS1-t request occurred
Motor Operation Prohibited	0x2250	Motor operation is prohibited

7.3.5.8 Functions for CiA402 Drive Profile

This section describes the functions implemented in this sample software.

For functions implemented in the EtherCAT SubDevice Stack Code generated by SSC Tool, refer to:

"Application Note ET9300 (EtherCAT SubDevice Stack Code)".

Table 7.33. CiA402 State Transition Functions (samplecia402.c)

Function Name	Description
CiA402_StateTransition1	No processing in this software (*1)
CiA402_StateTransition2	Flowchart shown in Figure 7.20 (*1)
CiA402_StateTransition3	Flowchart shown in Figure 7.21 (*1)
CiA402_StateTransition4	Flowchart shown in Figure 7.22 (*1)
CiA402_StateTransition5	Flowchart shown in Figure 7.23 (*1)
CiA402_StateTransition6	No processing in this software (*1)
CiA402_StateTransition7	No processing in this software (*1)
CiA402_StateTransition8	Flowchart shown in Figure 7.23 (*1)
CiA402_StateTransition9	Flowchart shown in Figure 7.23 (*1)
CiA402_StateTransition10	No processing in this software (*1)
CiA402_StateTransition11	Flowchart shown in Figure 7.23 (*1)
CiA402_StateTransition12	No processing in this software (*1)
CiA402_LocalError	Flowchart shown in Figure 7.24 (*1)
CiA402_StateTransition14	Flowchart shown in Figure 7.25 (*1)
CiA402_StateTransition15	Flowchart shown in Figure 7.26 (*1)
CiA402_StateTransition16	Flowchart shown in Figure 7.27 (*1)

(*1) Called when state transitions 1–16 defined in CiA402 FSA (Figure 7.19) occur.

Table 7.34. Functions for CiA402 (samplecia402.c)

Function Name	Description
APPL_MOTOR_MotionControl_Main	Motor control function called periodically from CiA402_Application function. Flowchart shown in Figure 7.28
check_position_limit	Checks whether the current position exceeds the limit. Flowchart shown in Figure 7.29
int32_to_real32	Converts int32 to real32 (float)
uint32_to_real32	Converts uint32 to real32 (float)
rpm_to_rads	Converts rpm to rad
update_actual_value	Retrieves motor's current position and speed. Flowchart shown in Figure 7.30
process_error_state	Called when a motor error occurs. Flowchart shown in Figure 7.31

process_profile_position_mode	Called in PP mode. Flowchart shown in Figure 7.32
process_homing_mode	Called in HM mode. Flowchart shown in Figure 7.33
process_cyclic_position_mode	Called in CSP mode. Flowchart shown in Figure 7.34
process_cyclic_velocity_mode	Called in CSV mode. Flowchart shown in Figure 7.35
APPL_GetDeviceID	Function to retrieve EtherCAT ID

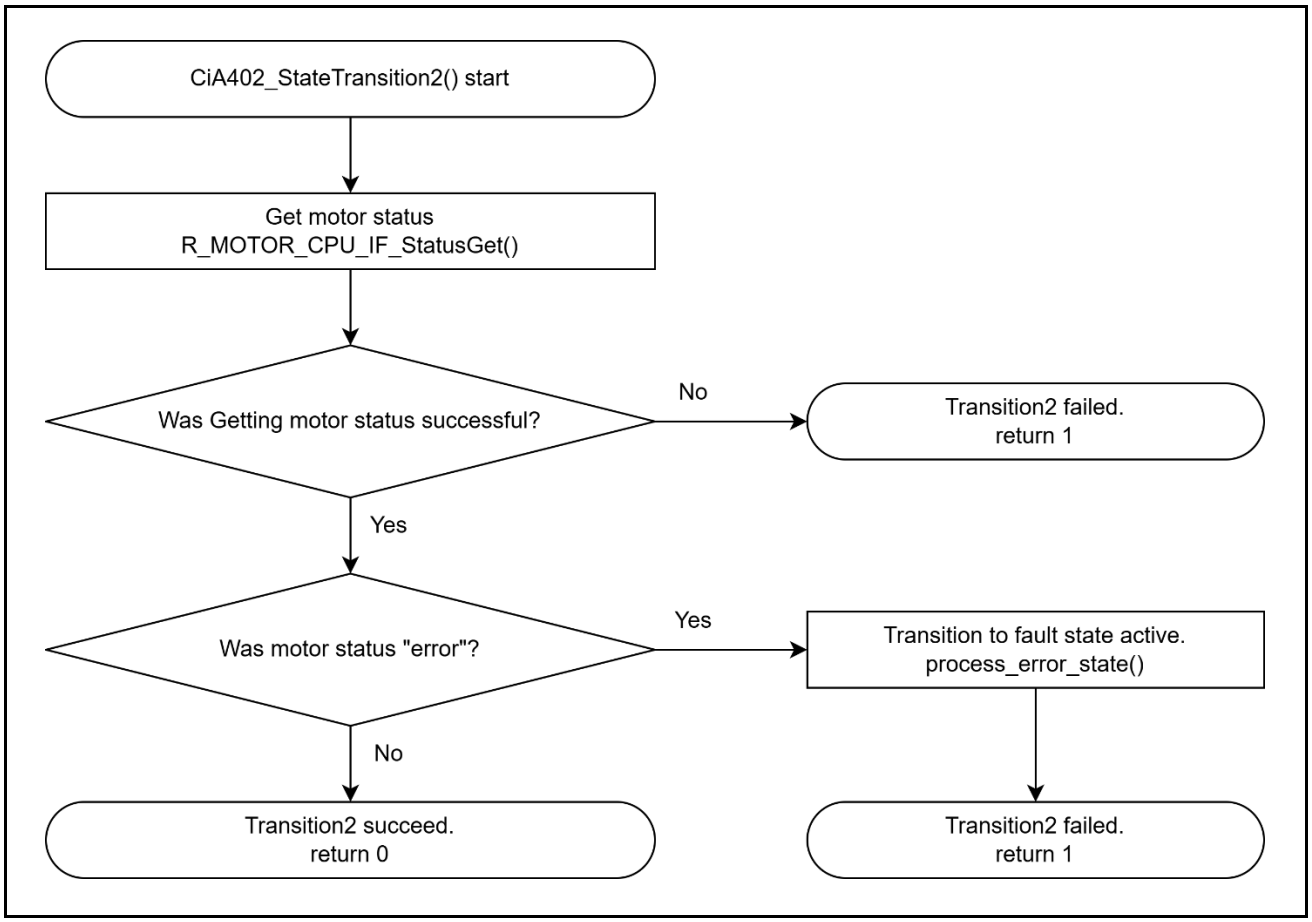


Figure 7.20. CiA402_StateTransition2 function flowchart

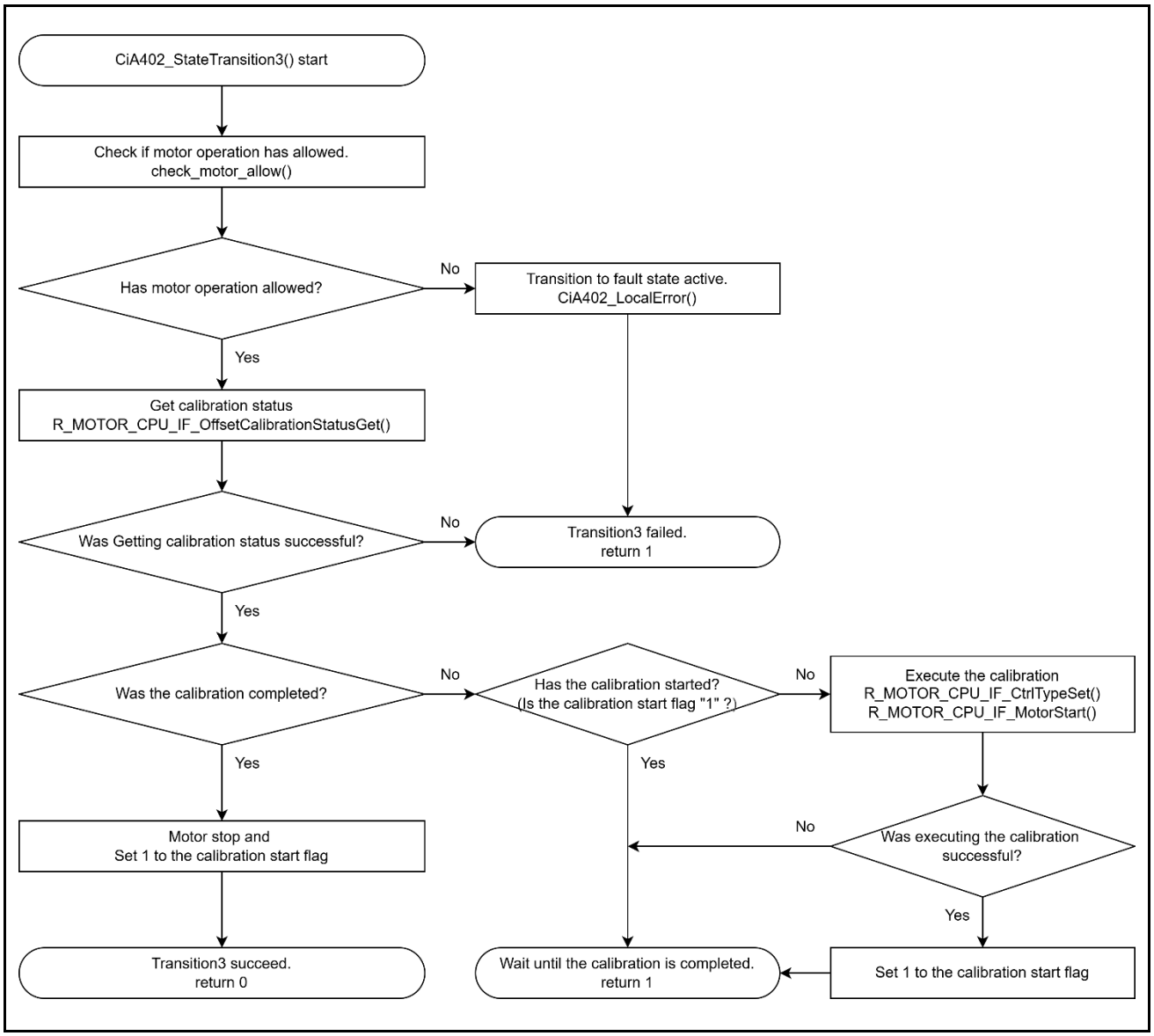


Figure 7.21. CiA402_StateTransition3 function flowchart

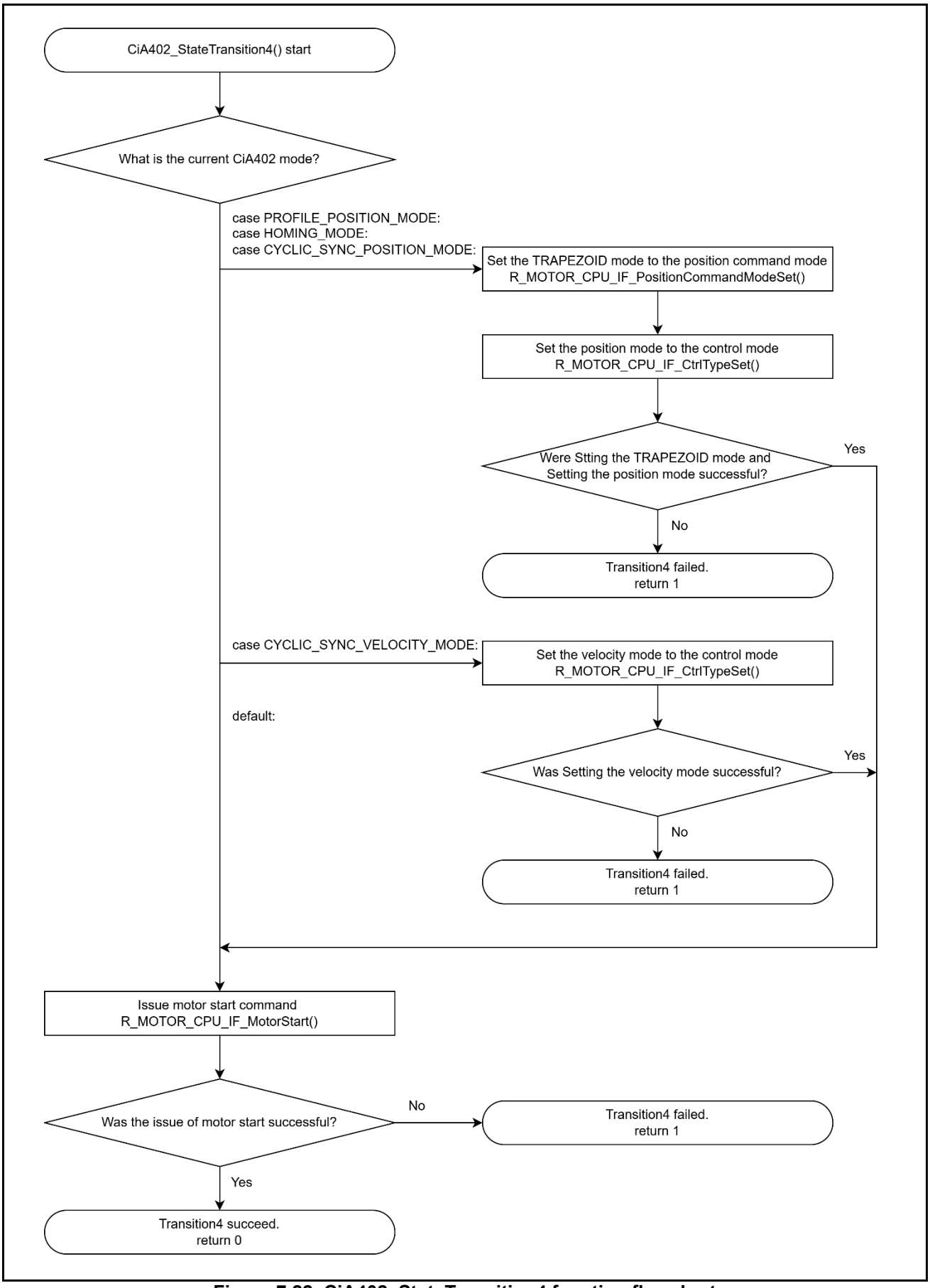


Figure 7.22. CiA402_StateTransition4 function flowchart

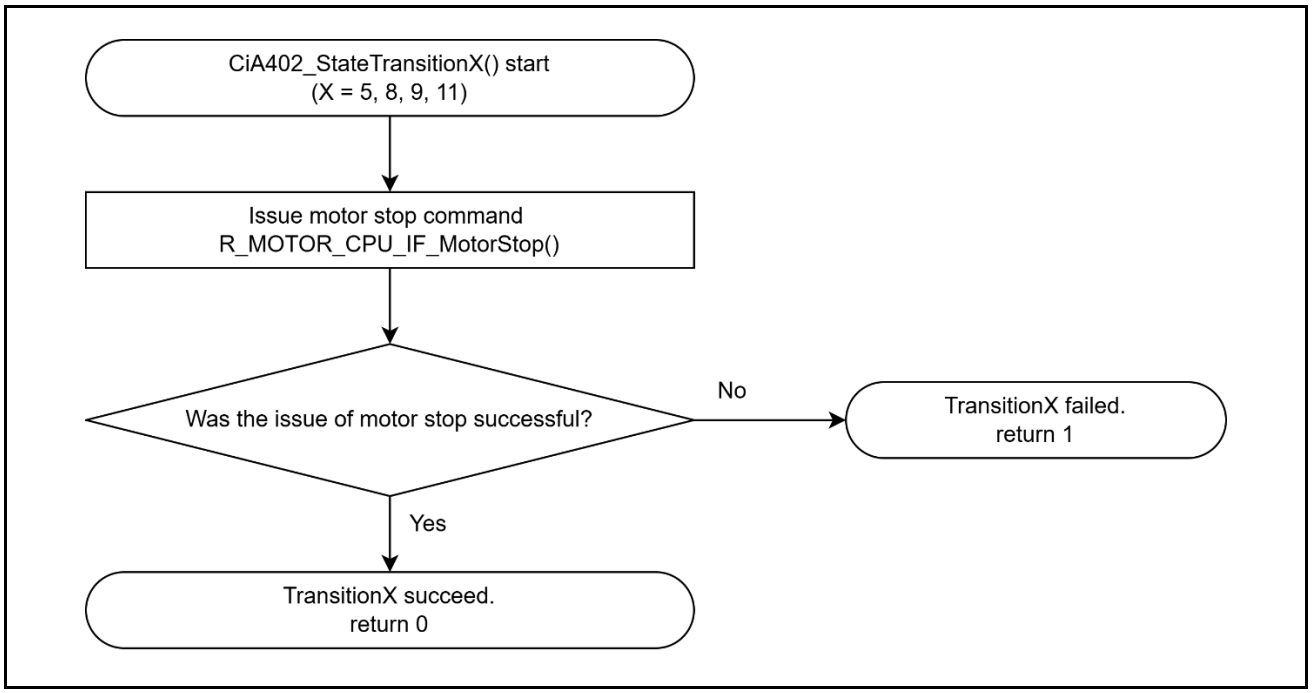


Figure 7.23. CiA402_StateTransition5, 8, 9, 11 function flowcharts

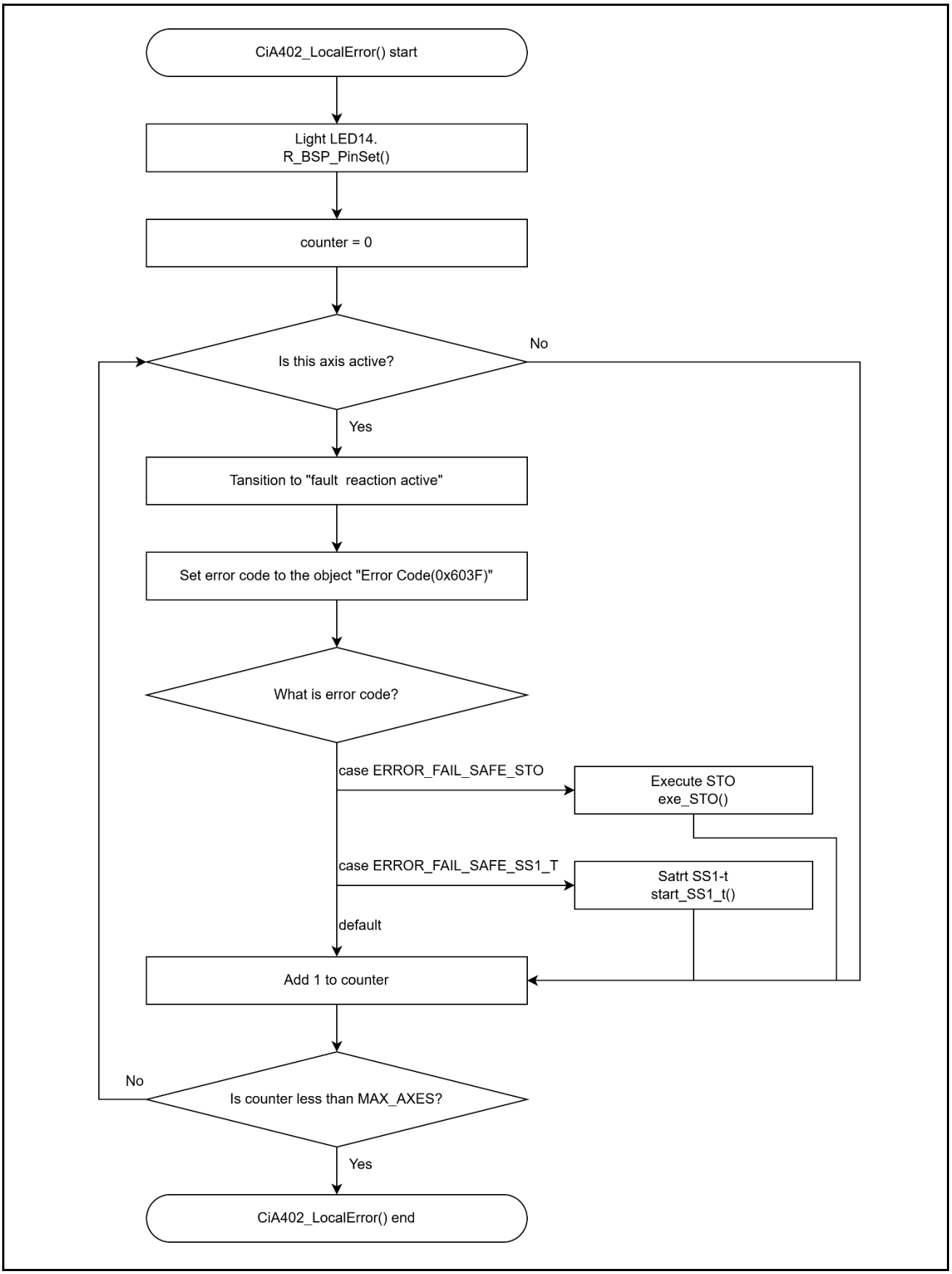


Figure 7.24. CiA402_LocalError function flowchart

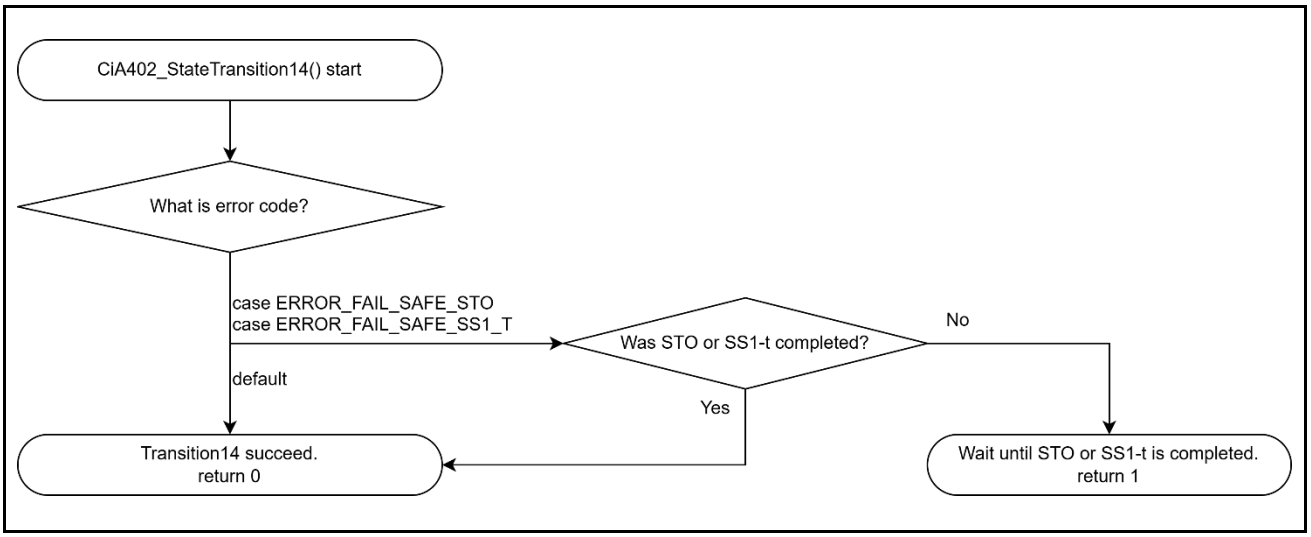


Figure 7.25. CiA402_StateTransition14 function flowchart

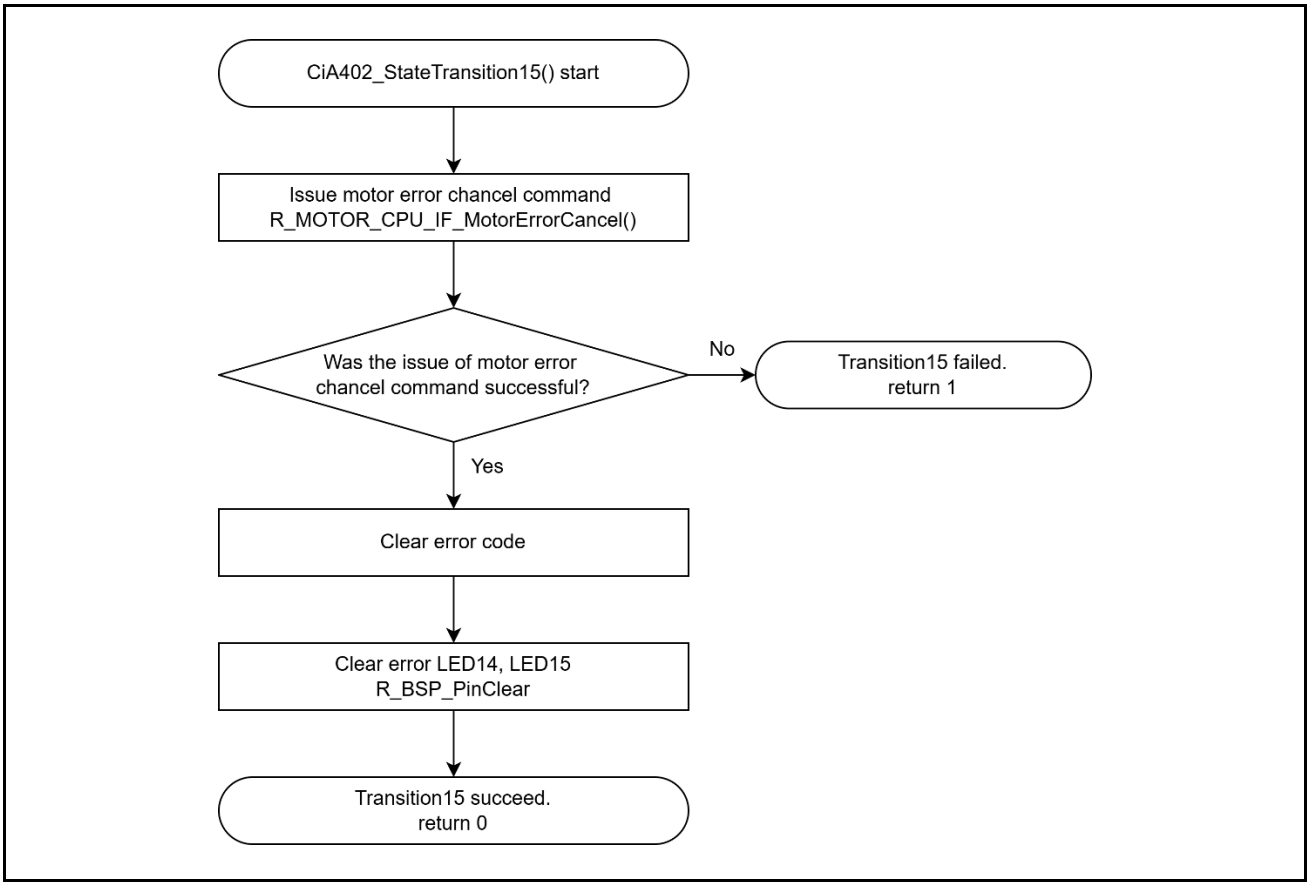


Figure 7.26. CiA402_StateTransition15 function flowchart

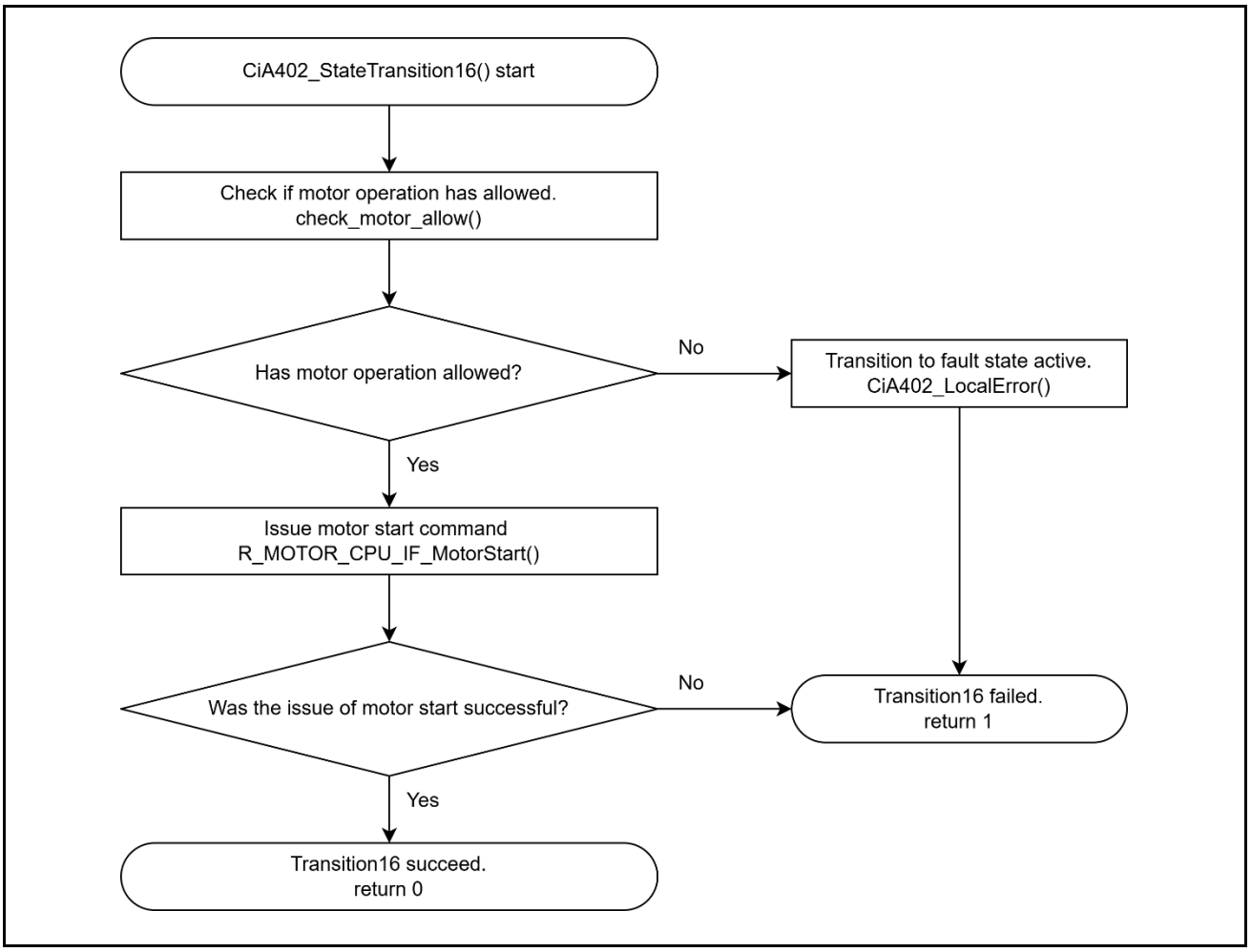


Figure 7.27. CiA402_StateTransition16 function flowchart

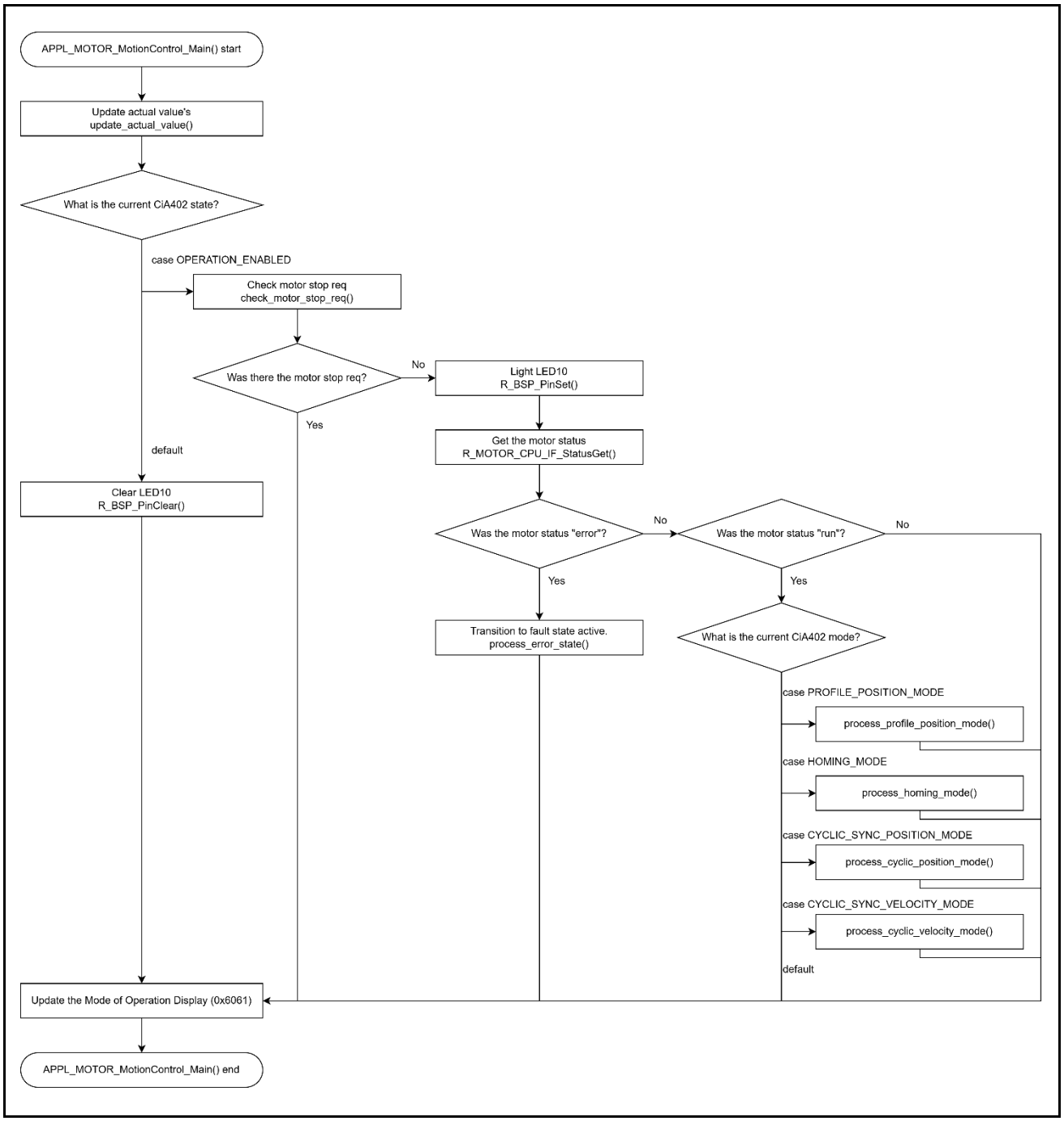


Figure 7.28. APPL_MOTOR_MotionControl_Main function flowchart

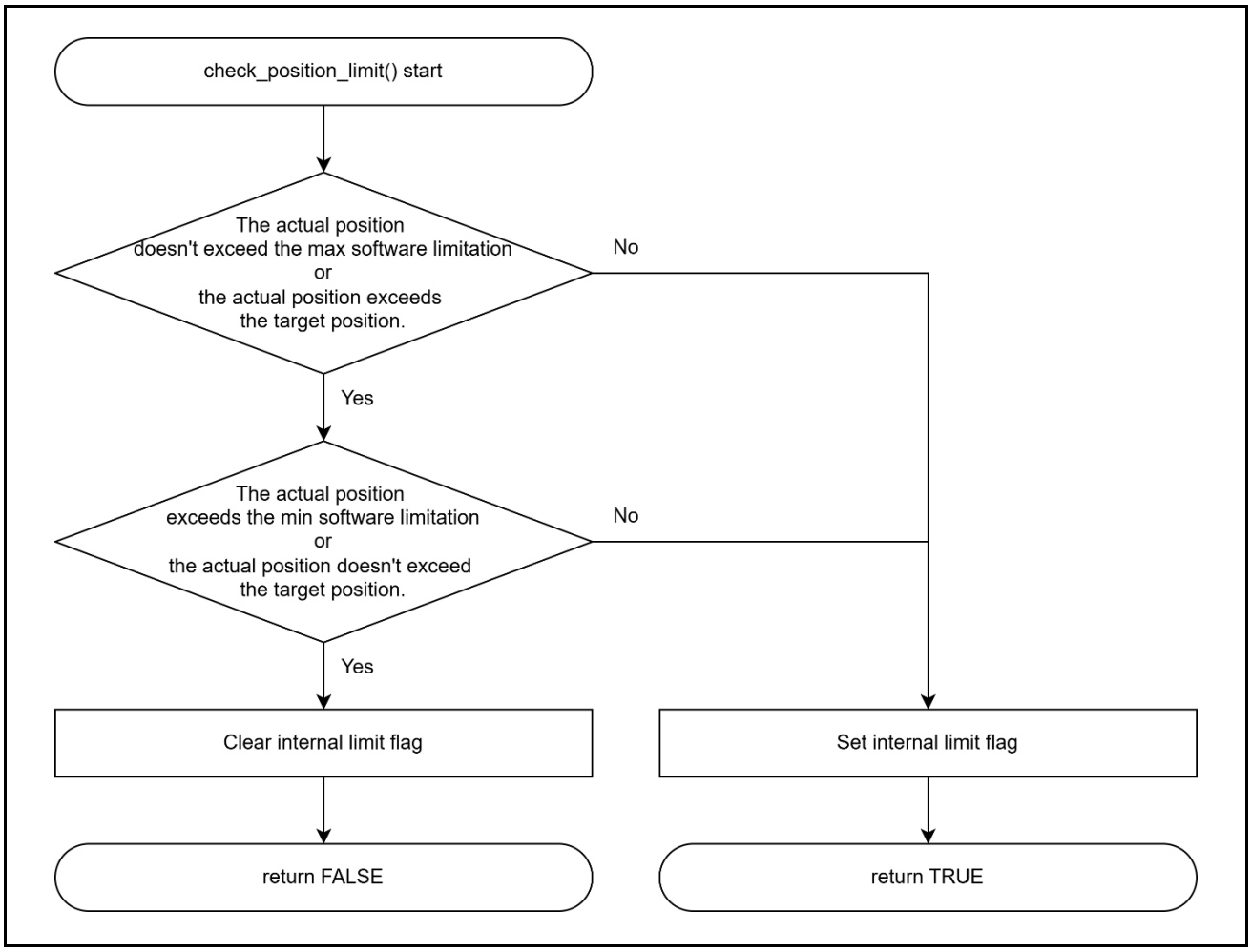


Figure 7.29. check_position_limit function flowchart

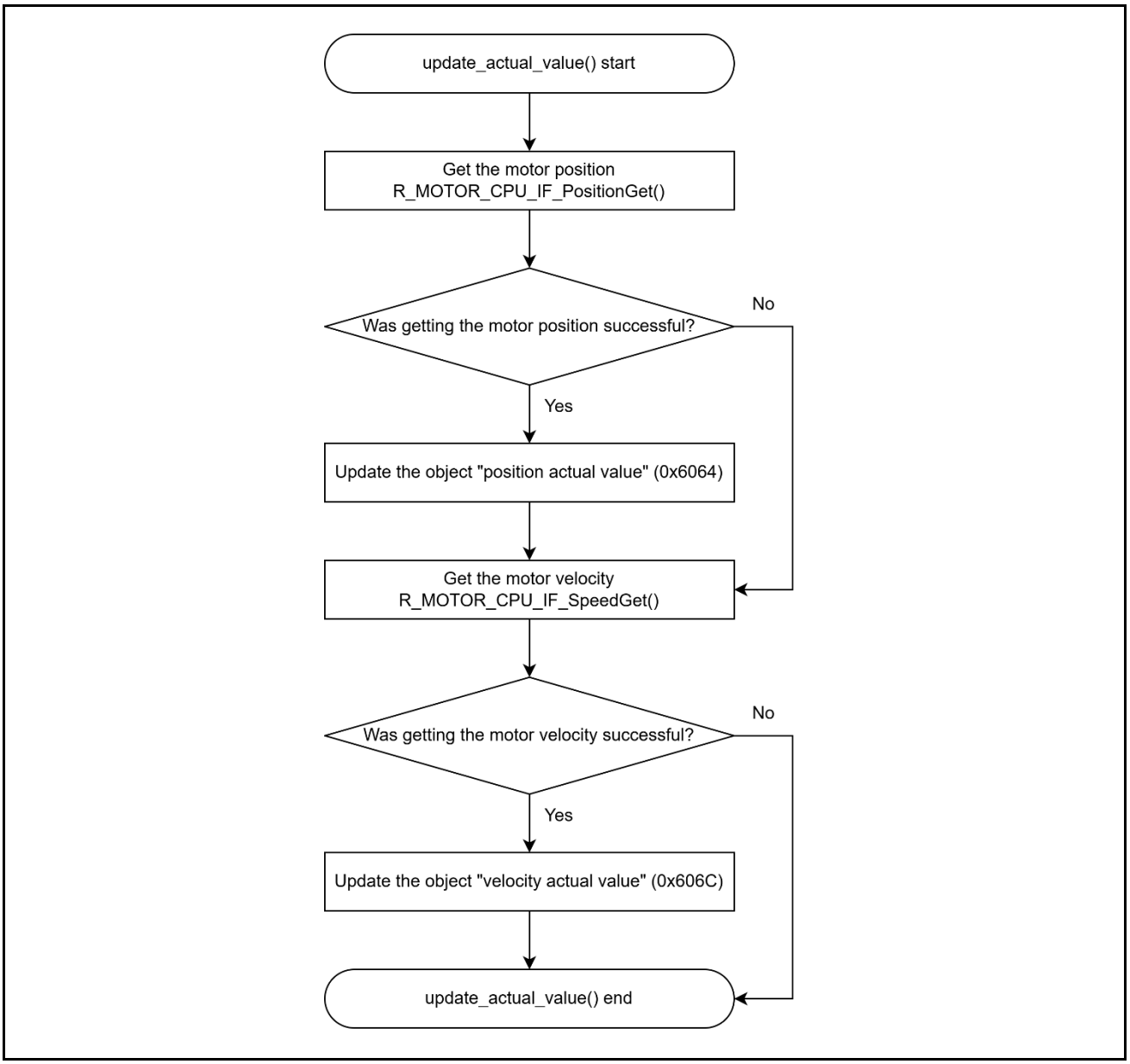


Figure 7.30. update_actual_value function flowchart

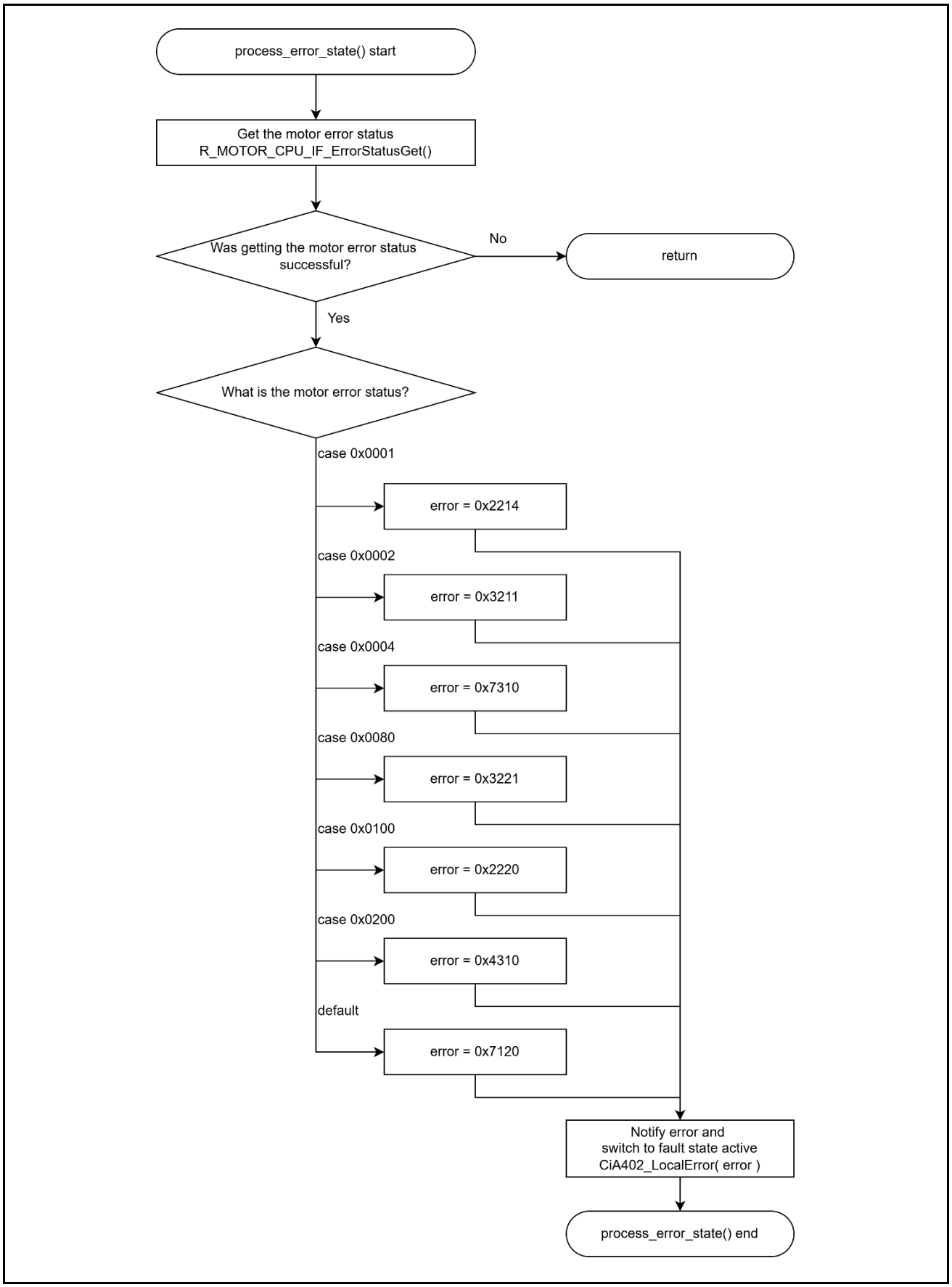


Figure 7.31. process_error_state function flowchart

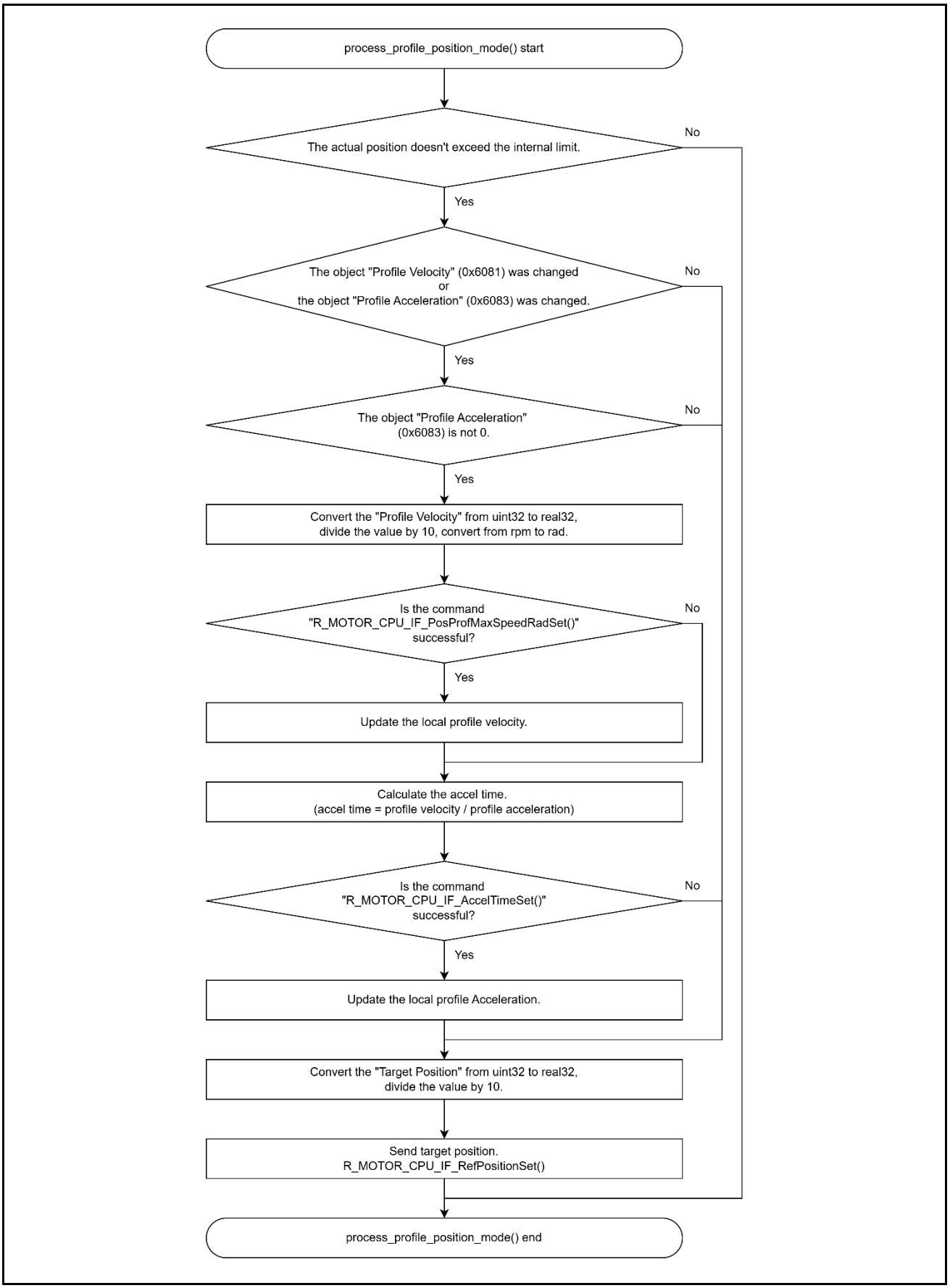


Figure 7.32. process_profile_position_mode function flowchart

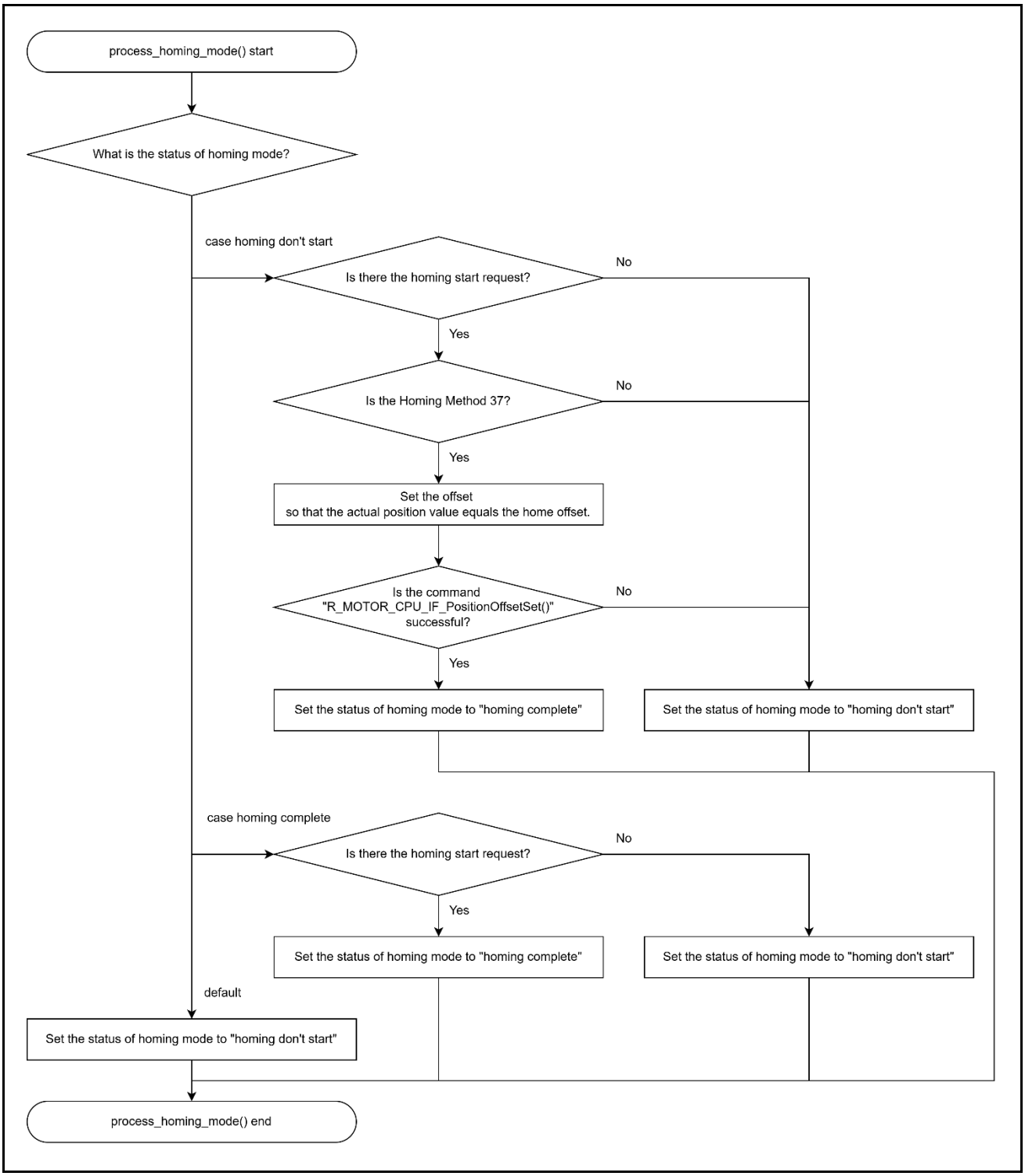


Figure 7.33. process_homing_mode function flowchart

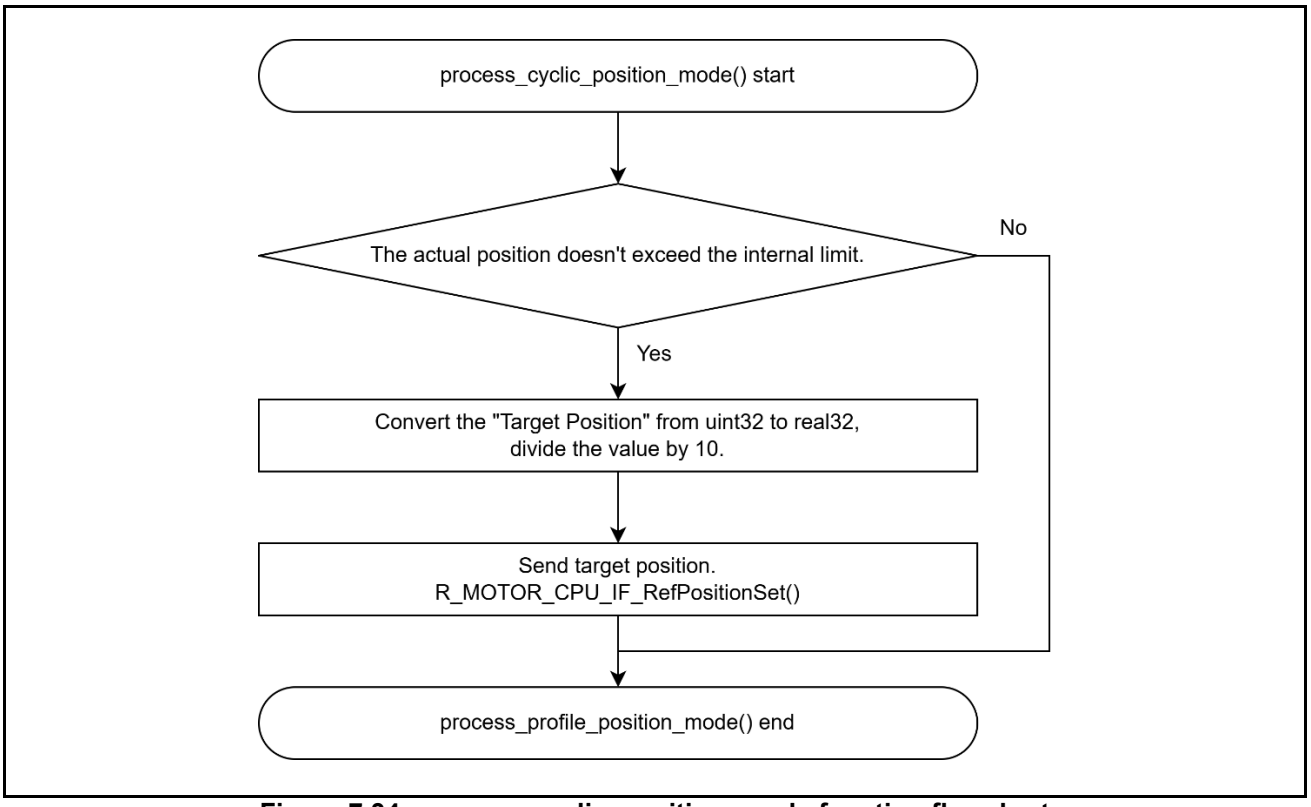


Figure 7.34. process_cyclic_position_mode function flowchart

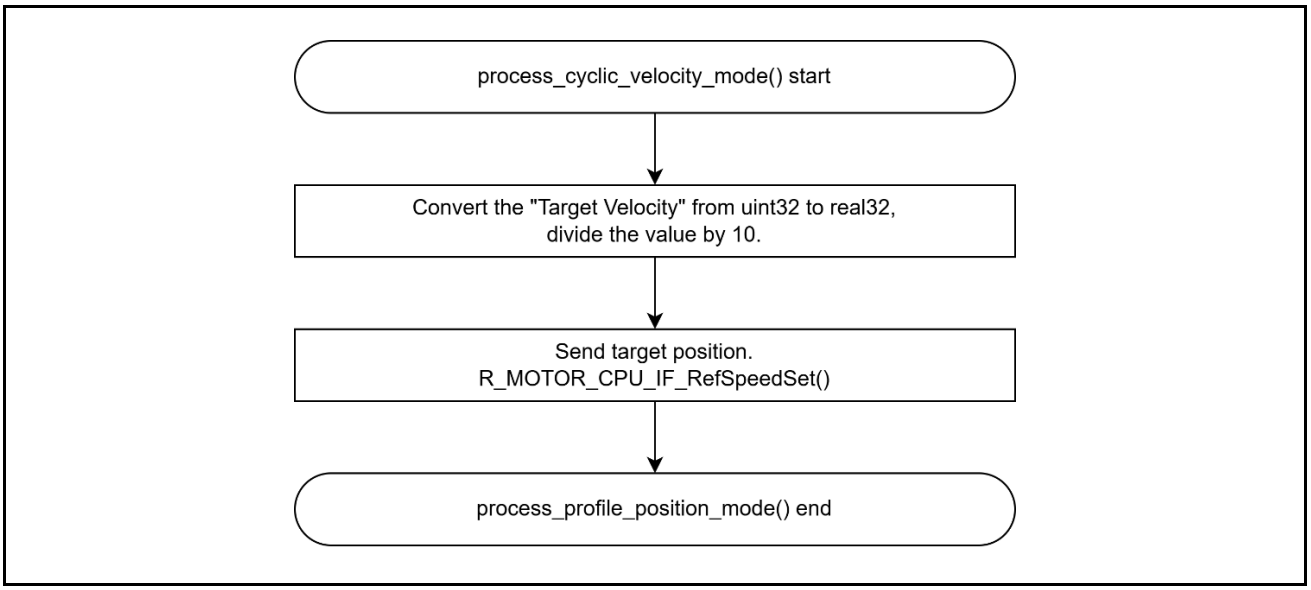


Figure 7.35. process_cyclic_velocity_mode function flowchart

7.3.5.9 Macro Definitions

This section describes the macro definitions implemented in this sample software.

For macro definitions implemented in the EtherCAT SubDevice Stack Code generated by the SSC Tool, refer to:

"Application Note ET9300 (EtherCAT SubDevice Stack Code)".

Table 7.35. Macro Definitions for CiA402 Drive Profile (samplecia402.c)

Macro Name	Definition Value	Remarks
CFG_POS_MODE	MOTOR_CPU_IF_POS_MODE_TRAPEZOID	Motor position control mode setting

7.3.5.10 Changes After SSC Tool Generated Files

This section describes the modifications made to the EtherCAT SubDevice Stack Code generated by the SSC Tool.

For details of the original source code before modification, refer to:

"Application Note ET9300 (EtherCAT SubDevice Stack Code)".

Table 7.36. List of Changes After SSC Tool Generation

File Name	Change	Details
cia402appl.c	Added include paths	Added hal_data.h, samplecia402.h, r_fusa_if.h
	Modified CiA402_Init function	Added initialization for new members in TCiA402Axis structure and initialization for added objects
	Modified CiA402_StateMachine function	Changed to call each CiA402 transition function listed in Table 7.33 during state transitions
	Deleted CiA402_LocalError and CiA402_DummyMotionControl functions	CiA402_LocalError is defined in samplecia402.c
	Modified CiA402_Application function	Changed to call CiA402 transition functions listed in Table 7.33 during state transitions. Enabled motor operation in SM mode as well as DC mode. Replaced CiA402_DummyMotionControl with APPL_MOTOR_MotionControl_Main
	Modified Write0xF030 function	Updated processing to match changes in PDO Assign and PDO mapping
	Modified APPL_GenerateMapping function	Updated to correctly calculate process data size
	Modified APPL_InputMapping function	Updated to match changes in PDO Assign and PDO mapping
	Modified APPL_OutputMapping function	
	Modified APPL_Application function	Added call to FSoE_Application function

	Deleted APPL_GetDeviceID function	APPL_Application is defined in samplecia402.c
cia402appl.h	Added macro definitions	See Table 7.38
	Changed macro definition values	Changed MAX_AXES to 1
	Added/modified Object Dictionary definitions	See Table 7.39
	Added members to TCiA402Axis structure	See Table 7.37
ecatappl.c	Added include path	Added samplecia402.h
	Added extern declaration	Added declaration for LocalAxes structure
	Modified MainLoop function	Added processing to call exe_SS1_t function every 1 ms
ecatcoe.h	Fixed structure alignment to 1	—
ecatslv.c	Modified StartInputHandler function	Removed processing that sets bits 2 and 3 of AL Event Mask Register to 1
mailbox.h	Fixed structure alignment to 1	—
	Changed macro definition value	Changed MBX_HEADER_SIZE to SIZEOF(TMBXHEADER)
sdoserv.h	Fixed structure alignment to 1	—

Table 7.37. Added Members in TCiA402Axis Structure (cia402appl.h)

Member Name	Role
bFSoEIsActive	FSoE module active flag
bCalibration	Calibration execution flag
bSTOCompleted	STO completion flag
bSS1tStarted	SS1-t completion flag
u32LocalProfVel	Stores previous profile velocity value
u32LocalProfAcc	Stores previous profile acceleration value
i32SS1tStartVel	Stores motor speed at SS1-t start
i32SS1tTime	Stores time from SS1-t start to completion
i32SS1tcnt	Counter used in SS1-t processing

Table 7.38. Added Macro Definitions (cia402appl.h)

Macro Name	Definition Value	Remarks
STATUSWORD_HOMING_MASK	0x2400	Statusword homing bit mask
STATUSWORD_HOMING_NOT_STARTED	0x0400	Homing not started
STATUSWORD_HOMING_COMPLETED	0x2000	Homing completed
CONTROLWORD_HOMING_START	0x0010	Homing start bit
HOMING_METHOD_37	0x0025	Homing Method 37
ERROR_FAIL_SAFE_STO	0xFF00	Error code for STO
ERROR_FAIL_SAFE_SS1_T	0xFF01	Error code for SS1-t
MAX_FSOE_MODULES	MAX_AXES	Number of FSoE modules
MAX_MODULE	MAX_AXES + MAX_FSOE_MODULES	Total number of modules
PP_MODULE_ID	0x00419800	ID for PP mode
HM_MODULE_ID	0x00519800	ID for HM mode
FSOE_MODULE_ID	0x00619800	ID for FSoE module
ENTRIES_1600	5	Number of mappings for PDO 0x1600
ENTRIES_1601	6	Number of mappings for PDO 0x1601
ENTRIES_1602	2	Number of mappings for PDO 0x1602
ENTRIES_1603	2	Number of mappings for PDO 0x1603
ENTRIES_1604	11	Number of mappings for PDO 0x1604
ENTRIES_1A00	5	Number of mappings for PDO 0x1A00
ENTRIES_1A01	5	Number of mappings for PDO 0x1A01
ENTRIES_1A02	5	Number of mappings for PDO 0x1A02
ENTRIES_1A03	5	Number of mappings for PDO 0x1A03
ENTRIES_1A04	11	Number of mappings for PDO 0x1A04

7.3.5.11 Object Dictionary

Table 7.39. Object Dictionary List

Index	ObjectCode	SI	DataType	Access	Object Name
0x1nnn Communication Area					
0x1000	VAR	-	UDINT	RO	Device type
0x1001	VAR	-	USINT	RO	Error Register
0x1008	VAR	-	STRING(43)	RO	Device Name
0x1009	VAR	-	STRING(4)	RO	Manufacturer Hardware Version
0x100A	VAR	-	STRING(4)	RO	Manufacturer Software Version
0x1018	RECORD	0	USINT	RO	Identity Object
		1	UDINT	RO	Vendor ID
		2	UDINT	RO	Product Code
		3	UDINT	RO	Revision Number
		4	UDINT	RO	Serial Number
0x10F1	RECORD	0	USINT	RO	Error Settings
		1	UDINT	RO	Local Error Reaction
		2	UINT	RW	Sync Error Counter Limit
0x10F8	VAR	-	ULINT	RO	Timestamp Object
0x1600	RECORD	0	USINT	RO	pp RxPDO
		1-5	UDINT	RO	SubIndex 00x
0x1601	RECORD	0	USINT	RO	hm RxPDO
		1-6	UDINT	RO	SubIndex 00x
0x1602	RECORD	0	USINT	RO	csp RxPDO
		1-2	UDINT	RO	SubIndex 00x
0x1603	RECORD	0	USINT	RO	csv RxPDO
		1-2	UDINT	RO	SubIndex 00x
0x1604	RECORD	0	USINT	RO	FSOE RxPDO
		1-11	UDINT	RO	SubIndex 0xx
0x1A00	RECORD	0	USINT	RO	pp TxPDO
		1-5	UDINT	RO	SubIndex 00x
0x1A01	RECORD	0	USINT	RO	hm TxPDO
		1-5	UDINT	RO	SubIndex 00x
0x1A02	RECORD	0	USINT	RO	csp TxPDO
		1-5	UDINT	RO	SubIndex 00x
0x1A03	RECORD	0	USINT	RO	csv TxPDO
		1-5	UDINT	RO	SubIndex 00x
0x1A04	RECORD	0	USINT	RO	FSOE TxPDO

		1-11	UDINT	RO	SubIndex 0xx
0x1C00	ARRAY	0	USINT	RO	Sync Manager Type
		1-4	USINT	RO	SubIndex 00x
0x1C12	ARRAY	0	USINT	RO *	RxPDO Assign
		1-2	UINT	RO *	SubIndex 00x
0x1C13	ARRAY	0	USINT	RO *	TxPDO Assign
		1-2	UINT	RO *	SubIndex 00x
0x1C32	RECORD	0	USINT	RO	SM output parameter
		1	UINT	RO *	Synchronization Type
		2	UDINT	RO	Cycle Time
		4	UINT	RO	Synchronization Types supported
		5	UDINT	RO	Minimum Cycle Time
		6	UDINT	RO	Calc and Copy Time
		8	UINT	RW	Get Cycle Time
		9	UDINT	RO	Delay Time
		10	UDINT	RW	Sync0 Cycle Time
		11	UINT	RO	SM-Event Missed
		12	UINT	RO	Cycle Time Too Small
		13	UINT	RO	Shift Time Too Short Counter
		32	BOOL	RO	Sync Error
0x1C33	RECORD	0	USINT	RO	SM input parameter
		1	UINT	RO *	Synchronization Type
		2	UDINT	RO	Cycle Time
		4	UINT	RO	Synchronization Types supported
		5	UDINT	RO	Minimum Cycle Time
		6	UDINT	RO	Calc and Copy Time
		8	UINT	RW	Get Cycle Time
		9	UDINT	RO	Delay Time
		10	UDINT	RW	Sync0 Cycle Time
		11	UINT	RO	SM-Event Missed
		12	UINT	RO	Cycle Time Too Small
		13	UINT	RO	Shift Time Too Short Counter
		32	BOOL	RO	Sync Error
0x6nnn CiA402 Drive Profile Area (ETG.6010)					
0x603F	VAR	-	UINT	RO	Error Code
0x6040	VAR	-	UINT	RW	Control Word
0x6041	VAR	-	UINT	RO	Status Word
0x605A	VAR	-	INT	RW	Quick stop option code

0x605B	VAR	-	INT	RW	Shutdown option code
0x605C	VAR	-	INT	RW	Disable operation option code
0x605D	VAR	-	INT	RW	Halt option code
0x605E	VAR	-	INT	RW	Fault reaction option code
0x6060	VAR	-	SINT	RW	Modes of Operation
0x6061	VAR	-	SINT	RO	Modes of operation display
0x6062	VAR	-	DINT	RO	Position demand value
0x6063	VAR	-	DINT	RO	Position actual internal value
0x6064	VAR	-	DINT	RO	Position actual value
0x6065	VAR	-	UDINT	RW	Following error window
0x6066	VAR	-	UINT	RW	Following error time out
0x6067	VAR	-	UDINT	RW	Position window
0x606C	VAR	-	DINT	RO	Velocity actual value
0x6072	VAR	-	UINT	RW	Max Torque
0x6077	VAR	-	INT	RO	Torque actual value
0x607A	VAR	-	DINT	RW	Target position
0x607B	RECORD	0	USINT	RO	Position range limit
		1	DINT	RW	Min position range limit
		2	DINT	RW	Max position range limit
0x607C	VAR	-	DINT	RW	Home Offset
0x607D	RECORD	0	USINT	RO	Software position limit
		1	DINT	RW	Min position limit
		2	DINT	RW	Max position limit
0x607F	VAR	-	UDINT	RW	Max profile velocity
0x6080	VAR	-	UDINT	RW	Max motor speed
0x6081	VAR	-	UDINT	RW	Profile velocity
0x6083	VAR	-	UDINT	RW	Profile acceleration
0x6084	VAR	-	UDINT	RW	Profile deceleration
0x6085	VAR	-	UDINT	RW	Quick stop deceleration
0x6091	RECORD	0	UDINT	RW	Gear ratio
		1	UDINT	RW	Motor revolutions
		2	UDINT	RW	Shaft revolutions
0x6099	RECORD	0	USINT	RO	Homing speeds
		1	UDINT	RW	Speed during search for switch
		2	UDINT	RW	Speed during search for zero
0x609A	VAR	-	UDINT	RW	Homing acceleration
0x60B0	VAR	-	DINT	RW	Position offset
0x60B1	VAR	-	DINT	RW	Velocity offset

0x60B2	VAR	-	INT	RW	Torque offset
0x60B8	VAR	-	UINT	RW	Touch probe function
0x60B9	VAR	-	UINT	RO	Touch probe status
0x60BA	VAR	-	DINT	RO	Touch probe position 1 positive value
0x60BB	VAR	-	DINT	RO	Touch probe position 1 negative value
0x60C2	RECORD	0	USINT	RO	Interpolation time period
		1	USINT	RW	Interpolation time period value
		2	SINT	RW	Interpolation time index
0x60D0	RECORD	0	USINT	RO	Touch probe source
		1	INT	RW	Touch probe 1 source
0x60E0	VAR	-	UINT	RW	Positive torque limit value
0x60E1	VAR	-	UINT	RW	Negative torque limit value
0x60F4	VAR	-	DINT	RO	Following error actual value
0x60FD	VAR	-	UDINT	RO	Digital inputs
0x60FE	RECORD	0	USINT	RO	Digital outputs
		1	UDINT	RW	Physical outputs
		2	UDINT	RW	Bit mask
0x60FF	VAR	-	DINT	RW	Target velocity
0x6402	VAR	-	UINT	RW	Motor Type
0x6502	VAR	-	UDINT	RO	Supported drive modes
0x67nn FSoE Safety Drive Connection (ETG.6100)					
0x6760	RECOED	0	USINT	RO	FSOE Slave Frame Elements
		1	USINT	RO	Command
		2	GAP	None	SubIndex 002
		3	UINT	RO	Connection ID
		4	UINT	RO	CRC0
0x6761	RECOED	0	USINT	RO	FSOE Inputs
		1-8	BOOL	RO	SubIndex 00x
0x6770	RECOED	0	USINT	RO	FSOE Main Frame Elements
		1	USINT	RO	Command
		2	GAP	None	SubIndex 002
		3	UINT	RO	Connection ID
		4	UINT	RO	CRC0
0x6771	RECOED	0	USINT	RO	FSOE Outputs
		1-8	BOOL	RW	SubIndex 00x
0x6780	RECOED	0	USINT	RO	Safety Parameter
		1	USINT	RW	Sensor Test Channel 1 active
		2	USINT	RW	Sensor Test Channel 2 active

0x6790	RECOED	0	USINT	RO	FSOE Info Data
		1	UINT	RO	FSOE Connection ID
		2	UINT	RO	Device Type
		3	UDINT	RO	Vendor ID
		4	UDINT	RO	Product Code
		5	UDINT	RO	Revision Number
		6	UDINT	RO	Serial Number
		7	UINT	RO	Network Flags
		8	UINT	RO	Network Port
		9	UINT	RO	Network Segment Address
0x6791	RECOED	0	USINT	RO	FSOE Communication Parameter
		1	STRING(2)	RO	Version
		2	UINT	RO	Safe Address
		3	UINT	RO	Connection
		4	UINT	RO	Watchdog Time
		5	ARRAY [0..5] OF BYTE	RO	Unique ID
		6	UINT	RO	Connection Type
		7	UINT	RO	Communication Parameter Length
		8	UINT	RO	Application Paramter Length
0x67A0	RECOED	0	USINT	RO	FSOE Connection Diagnosis
		1	UINT	RO	Connection State
		2	UINT	RO	Connection Diagnosis
0xF000	RECOED	0	USINT	RO	Modular Device Profile
		1	UINT	RO	Index distance
		2	UINT	RO	Maximum number of modules
0xF010	ARRAY	0	USINT	RO	Module List
		1-2	UDINT	RO	SubIndex 00x
0xF030	ARRAY	0	USINT	RO	Configured Module Ident List
		1-2	UDINT	RO	SubIndex 00x
0xF050	ARRAY	0	USINT	RO	Detected modules
		1-2	UDINT	RO	SubIndex 00x
0xF980	RECOED	0	USINT	RO	Safe Address
		1	UINT	RO	FSoE Address

* This is RW only if its status is PreOP.

7.3.6 Safety Drive

This software implements a function to stop the motor controlled by RZ/T2M when a system abnormality is detected.

The motor stop function complies with IEC61800-5-2 STO and SS1-t.

The I/O ports used for this function are shown in Table 7.40.

Table 7.40. I/O Ports for Safety Drive Function

I/O Port	Description
P14_2	This port is controlled by RZ/T2L-A. When “L”, motor operation is prohibited. When “H”, motor operation is allowed. If the level changes from “H” to “L”, it is considered a motor stop request, and STO or SS1-t is executed depending on the level of P14_3.
P14_3	I/O port for selecting the motor stop function. This port is controlled by RZ/T2L-A. When “L”, the motor stop function is STO. When “H”, the motor stop function is SS1-t.

The flow from motor operation start to STO/SS1-t occurrence and recovery is shown in Figure 7.36 and Figure 7.37.

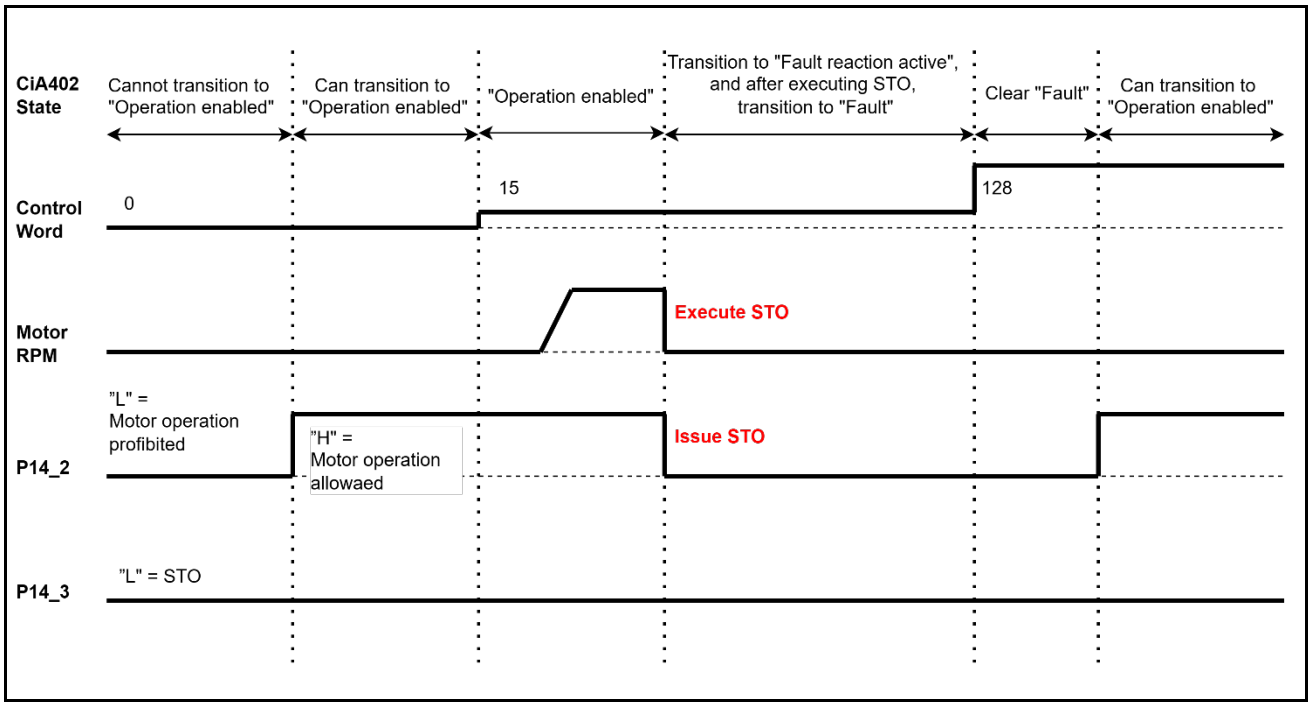


Figure 7.36. Flow of Motor Operation, STO, and Recovery

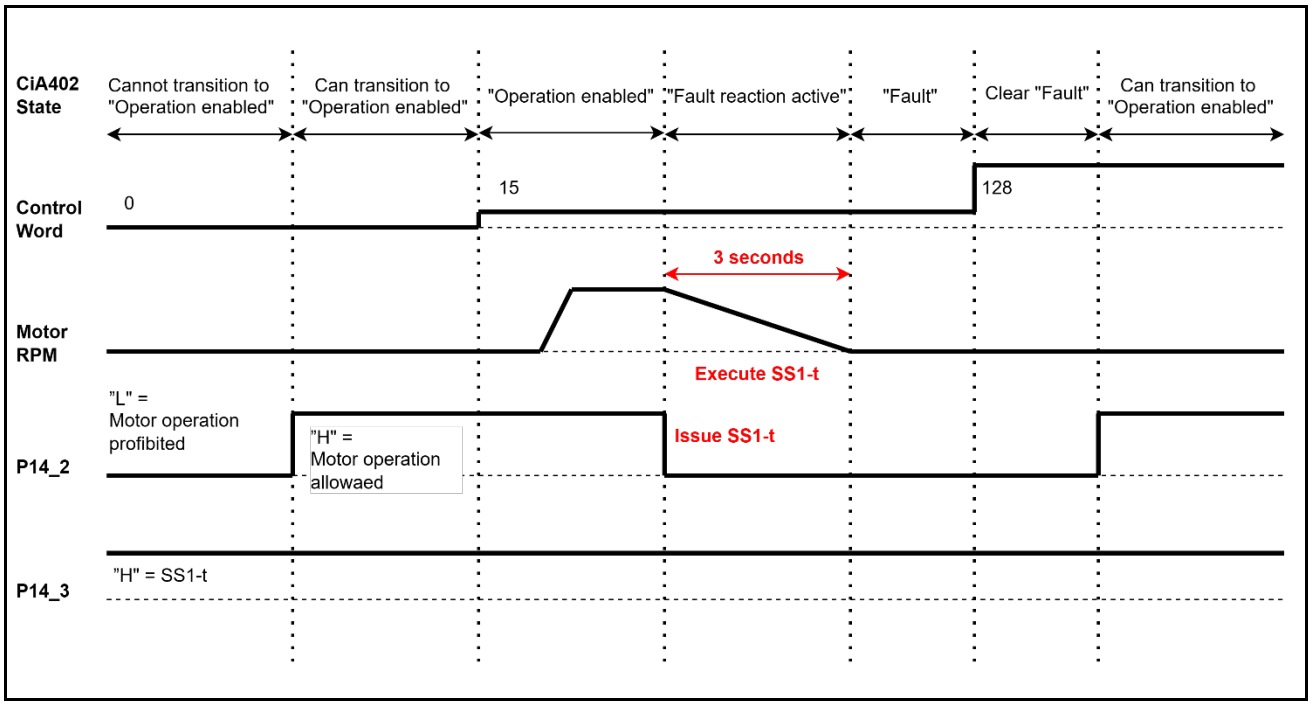


Figure 7.37. Flow of Motor Operation, SS1-t, and Recovery

7.3.6.1 Functions for Safety Drive

Table 7.41. Safety Drive Functions (samplecia402.c)

Function Name	Description
check_motor_allow	Reads the level of P14_2 and checks whether motor operation is allowed.
check_motor_stop_req	Checks whether a motor stop request occurs during motor operation. If a stop request occurs, calls CiA402_LocalError function to transition to “Fault Reaction Active” and starts STO or SS1-t processing.
exe_STO	Executes STO. Sends motor stop command to the motor control CPU and stops the motor. Also turns on LED15 to indicate STO completion and sets the STO completion flag in the TCiA402Axis structure to TRUE.
start_SS1_t	Starts SS1-t processing. Sets SS1-t related members in the TCiA402Axis structure and sets the SS1-t start flag to TRUE.
exe_SS1_t	Executes SS1-t processing. If the SS1-t start flag in TCiA402Axis is TRUE, this function is called every 1 ms from the MainLoop function. Blinks LED15 every 100 ms to indicate SS1-t execution. Calculates speed every 1 ms and updates the speed command sent to the motor control CPU. When SS1-t processing ends, calls exe_STO function.

7.3.6.2 Macro Definitions

Table 7.42. Safety Drive Macro Definitions (samplecia402.c)

Macro Name	Definition Value	Remarks
CFG_SS1T_TIME	3000	Sets the time [ms] from SS1-t start to completion.

7.3.7 Fusa CPU Communication

This module sends the Safety Master PDU received from the FSoE Master to the Functional Safety CPU (RZ/T2L-A).

It also sends the Safety Slave PDU received from the Functional Safety CPU to the FSoE Master.

Communication with the Functional Safety CPU uses the FuSa I/F module.

For details of the FuSa I/F module, see Section 7.3.9 Fusa I/F.

For specifications of the Functional Safety CPU and communication, refer to:

"RZ Family FSoE Application Software Development Handbook (R30UZ0176JJ0110)".

7.3.7.1 FSoE PDO Settings

Table 7.43. FSoE RxPDO List

RxPDO	Index	Size[bit]
FSoE output Command	0x6770:1	8
FSoE output data 1	0x6771:1	1
FSoE output data 2	0x6771:2	1
FSoE output data 3	0x6771:3	1
FSoE output data 4	0x6771:4	1
FSoE output data 5	0x6771:5	1
FSoE output data 6	0x6771:6	1
FSoE output data 7	0x6771:7	1
FSoE output data 8	0x6771:8	1
FSoE output CRC0	0x6770:4	16
FSoE output ConnectionID	0x6770:3	16

Table 7.44. FSoE TxPDO List

TxPDO	Index	Size[bit]
FSoE input Command	0x6760:1	8
FSoE input data 1	0x6761:1	1
FSoE input data 2	0x6761:2	1
FSoE input data 3	0x6761:3	1
FSoE input data 4	0x6761:4	1
FSoE input data 5	0x6761:5	1
FSoE input data 6	0x6761:6	1
FSoE input data 7	0x6761:7	1
FSoE input data 8	0x6761:8	1
FSoE input CRC0	0x6760:4	16
FSoE input ConnectionID	0x6760:3	16

7.3.7.2 Objects Used for Communication with Functional Safety CPU

Table 7.45. Objects Used for Sending to Functional Safety CPU

Packet Name	Object	Index	Size[byte]
Safety Master PDU	FSoE output Command	0x6770:1	1
	FSoE output data 1 ~ 8	0x6771:1 ~ 8	1
	FSoE output CRC0	0x6770:4	2
	FSoE output ConnectionID	0x6770:3	2

Table 7.46. Objects Used for Receiving from Functional Safety CPU

Packet Name	Object	Index	Size[byte]
Safety Slave PDU	FSoE input Command	0x6760:1	1
	FSoE input data 1 ~ 8	0x6761:1 ~ 8	1
	FSoE input CRC0	0x6760:4	2
	FSoE input ConnectionID	0x6760:3	2
Communication parameter	Connection ID	0x6791:3	2
	Connection State	0x67A0:1	2
	Connection Diagnosis	0x67A0:2	2
	Watchdog Time	0x6791:4	2
	Communication Parameter Length	0x6791:7	2
	Application Parameter Length	0x6791:8	2

7.3.7.3 Functions

Table 7.47. Function List (samplecia402.c)

Function Name	Description
FSoE_Application	Called periodically from APPL_Application function. Flowchart shown in Figure 7.38

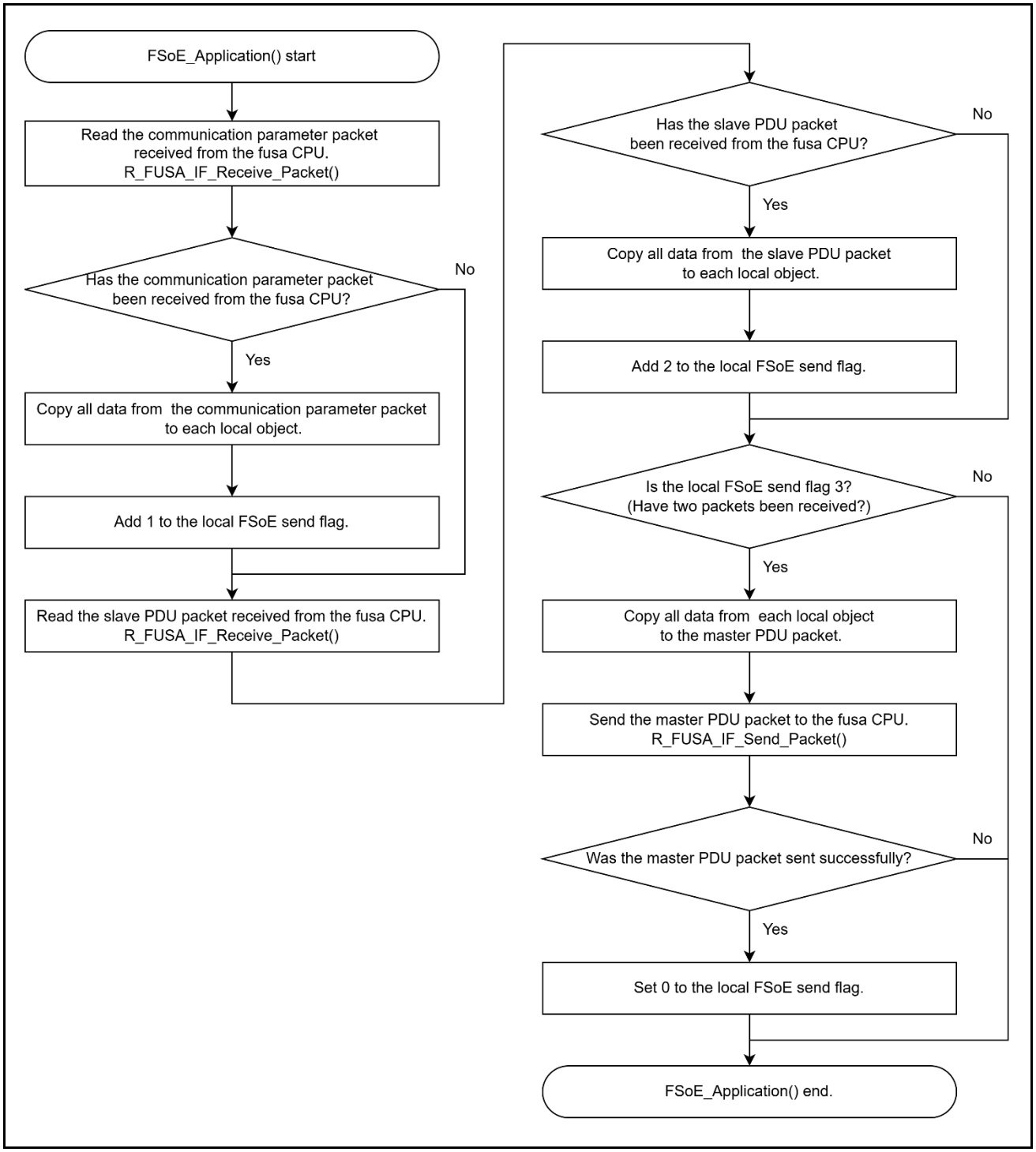


Figure 7.38. Flowchart of FSoE_Application Function

7.3.7.4 Structures and Global Variables

Table 7.48. Structure List (r_samplecia402.h)

Structure	Variable	Remarks
fsoe_params_t Communication parameter packet structure	connection_id	Used in FSoE_Application function
	connection_state	
	connection_diagnosis	
	watchdog_time	
	com_parameter_length	
	appl_parameter_length	

Table 7.49. Global Variables (r_samplecia402.c)

Type	Instance Name	Description
UINT8	u8FSoESendFlag	FSoE send flag. Used in FSoE_Application function. Initial value is 3.

7.3.8 Motor CPU I/F

The Motor CPU I/F module uses the Shared Memory Driver to send motor control commands, update reference values, and retrieve motor status from the Motor CPU (RZ/T2M CPU0).

7.3.8.1 Module Structure

The module structure diagram is shown in Figure 7-39.

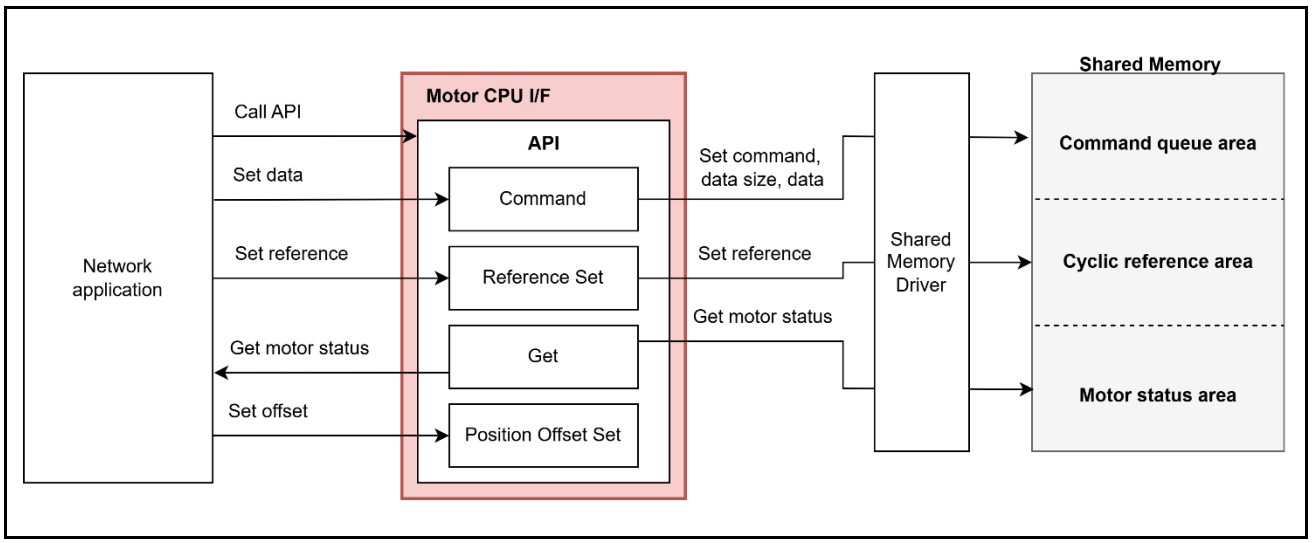


Figure 7.39. Motor CPU I/F Module Structure

This module mainly provides **four types of APIs**:

(1) **Command Set**
Stores motor control commands, data size, and data in the **Command Queue Area** of Shared Memory.

(2) **Reference Set**
Updates various reference values set in the **Cyclic Reference Area** of Shared Memory.

(3) **Get Motor Status**
Retrieves various motor status values stored in the **Motor Status Area** of Shared Memory.

(4) **Position Offset Set**
API prepared for setting the Home position. Sets an offset to position information.

7.3.8.2 Shared Memory Allocation

Same as the tables described in Section 7.2.3.2.

7.3.8.3 API

Table 7.50. Motor CPU I/F API List

API	Description
R_MOTOR_CPU_IF_Open	Initializes Motor CPU I/F and Shared Memory Driver. Flowchart: Figure 7.40
R_MOTOR_CPU_IF_Close	Terminates Motor CPU I/F and Shared Memory Driver. Flowchart: Figure 7.41
R_MOTOR_CPU_IF_Reset R_MOTOR_CPU_IF_MotorStart R_MOTOR_CPU_IF_MotorStop R_MOTOR_CPU_IF_MotorReset R_MOTOR_CPU_IF_MotorErrorCancel R_MOTOR_CPU_IF_ErrorSet R_MOTOR_CPU_IF_CtrlTypeSet R_MOTOR_CPU_IF_PositionCommandModeSet R_MOTOR_CPU_IF_RotorAngleOffsetSet R_MOTOR_CPU_IF_CurrentOffsetSet R_MOTOR_CPU_IF_CurrentOpenLoopSet R_MOTOR_CPU_IF_AccelTimeSet R_MOTOR_CPU_IF_PosProfMaxSpeedRadSet	Command Set API: Stores motor control commands, data size, and data in the Command Queue Area of Shared Memory. Flowchart: Figure 7.42
R_MOTOR_CPU_IF_RefPositionSet R_MOTOR_CPU_IF_RefSpeedSet R_MOTOR_CPU_IF_RefTorqueSet	Reference Set API: Updates various reference values in the Cyclic Reference Area of Shared Memory. Flowchart: Figure 7.43
R_MOTOR_CPU_IF_PositionGet R_MOTOR_CPU_IF_SpeedGet R_MOTOR_CPU_IF_StatusGet R_MOTOR_CPU_IF_ErrorStatusGet R_MOTOR_CPU_IF_LoopModeStatusGet R_MOTOR_CPU_IF_InPositionFlagGet R_MOTOR_CPU_IF_OffsetCalibrationStatusGet	Get Motor Status API: Retrieves various motor status values from the Motor Status Area of Shared Memory. Flowchart: Figure 7.44
R_MOTOR_CPU_IF_PositionOffsetSet	Position Offset Set API: Sets an offset to position information. Flowchart: Figure 7.45

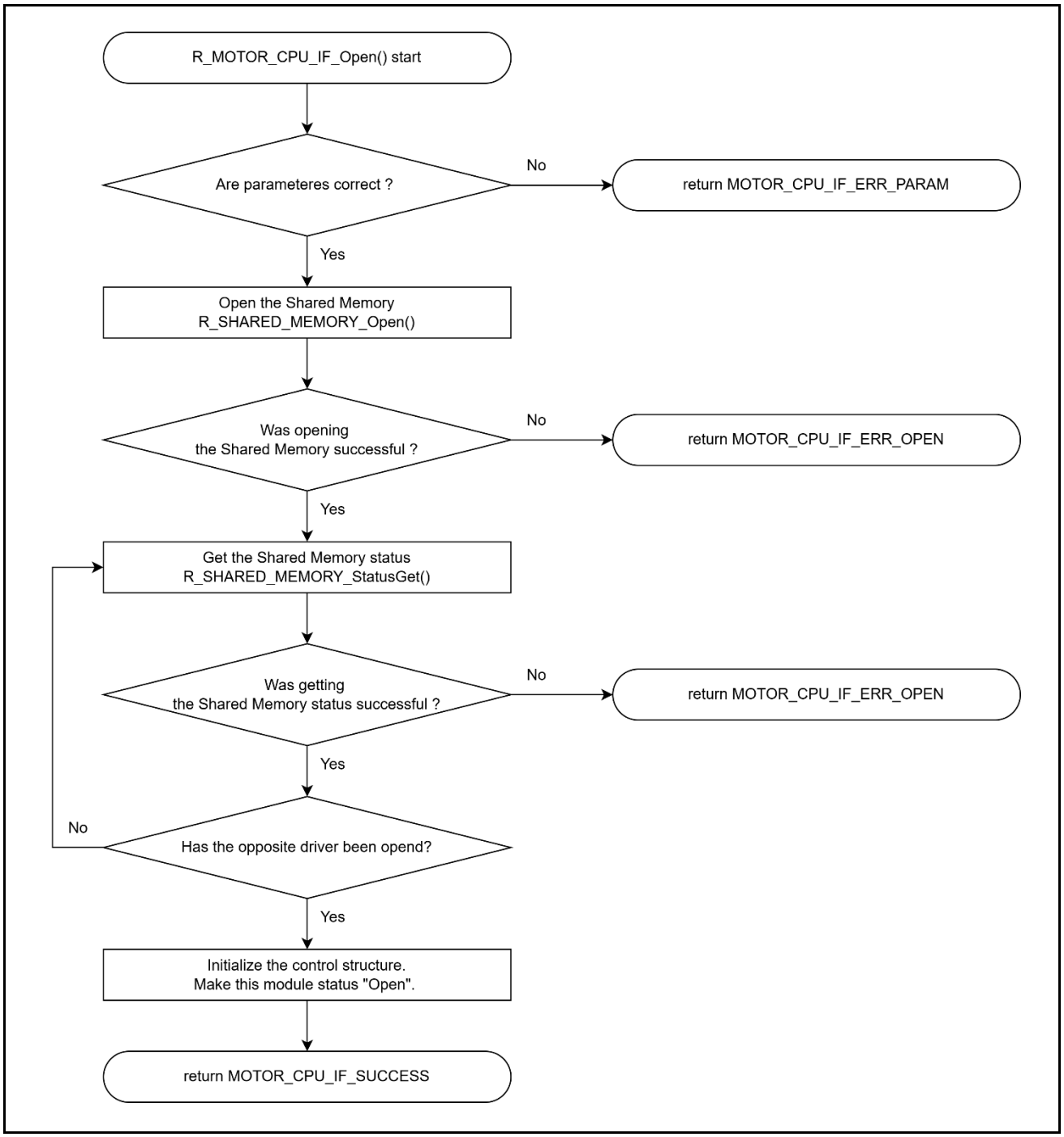


Figure 7.40. Flowchart of R_MOTOR_CPU_IF_Open Function

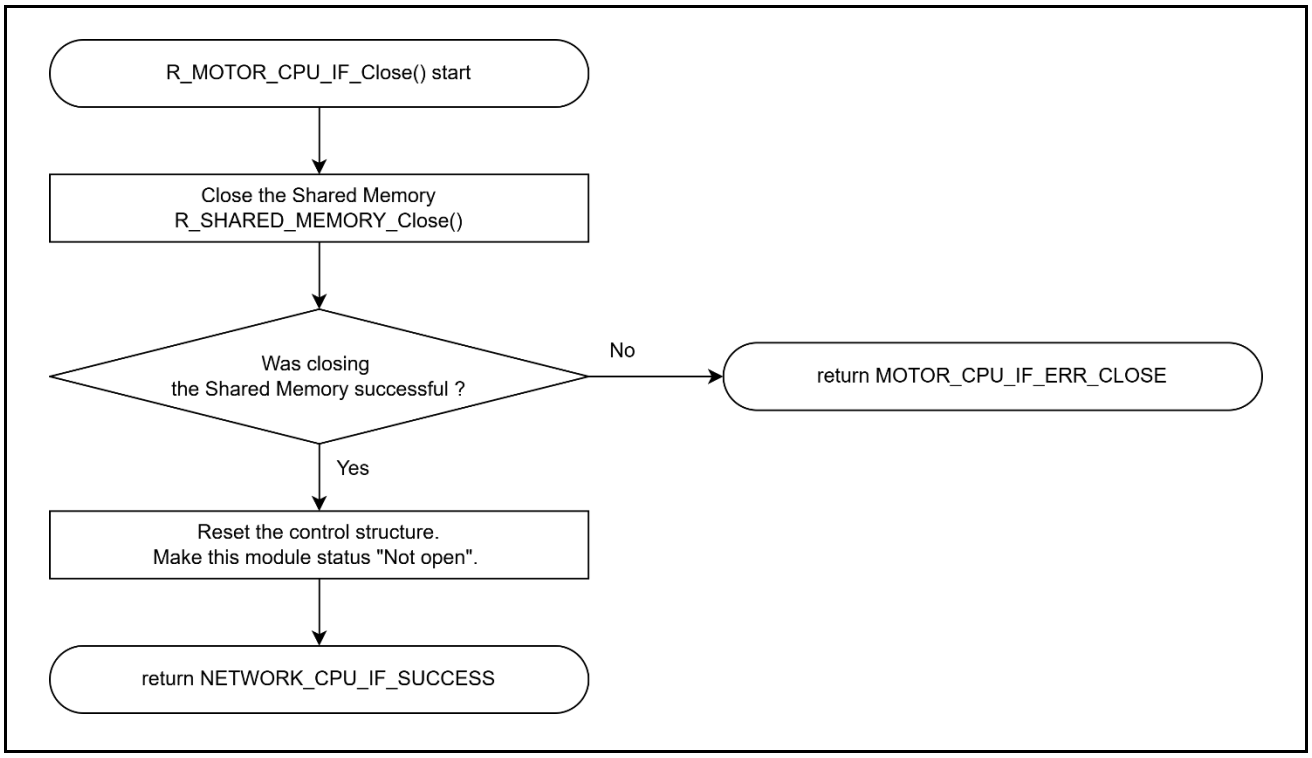


Figure 7.41. Flowchart of R_MOTOR_CPU_IF_Close Function

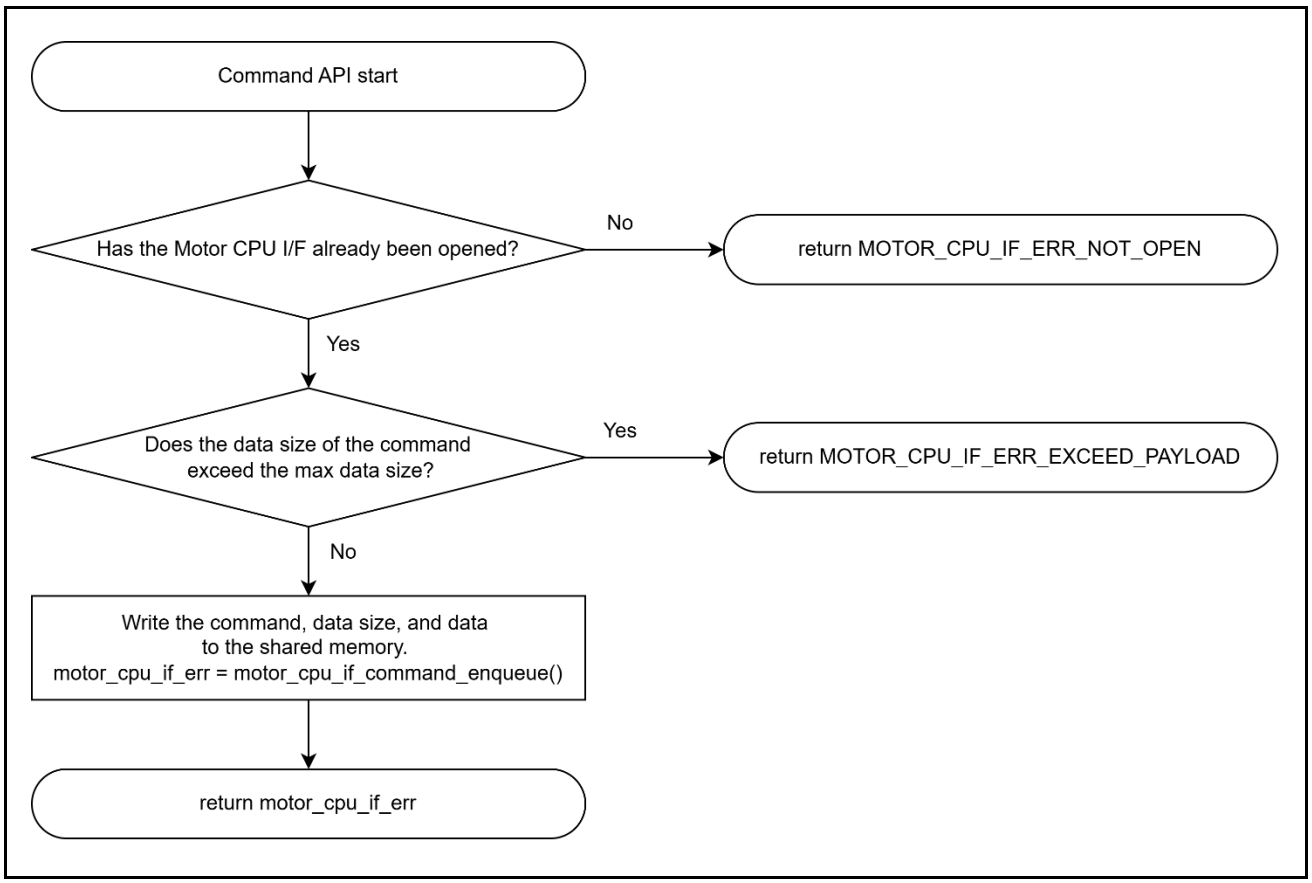


Figure 7.42. Flowchart of R_MOTOR_CPU_IF Command Set API

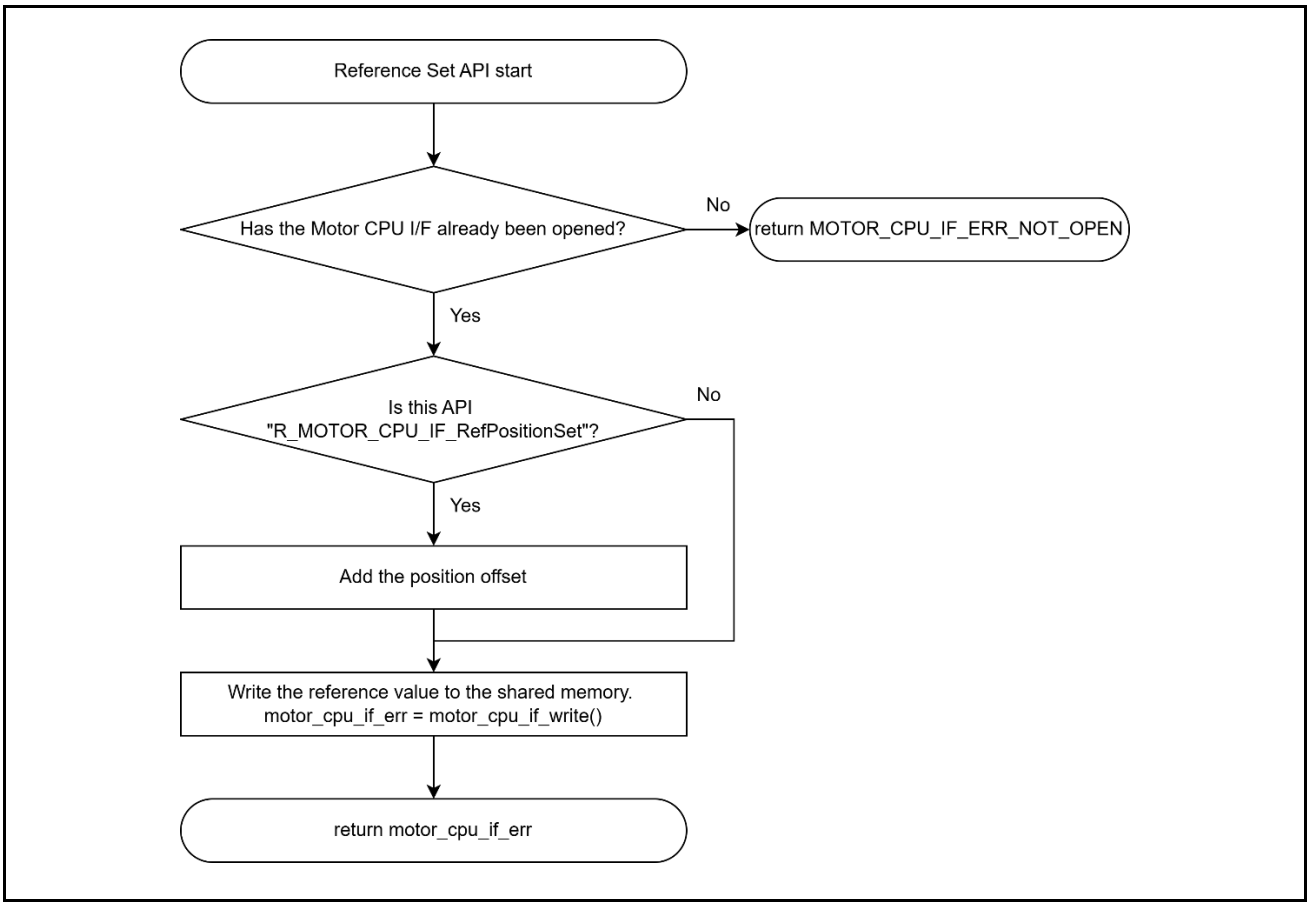


Figure 7.43. Flowchart of R_NETWORK_CPU_IF Reference Set API

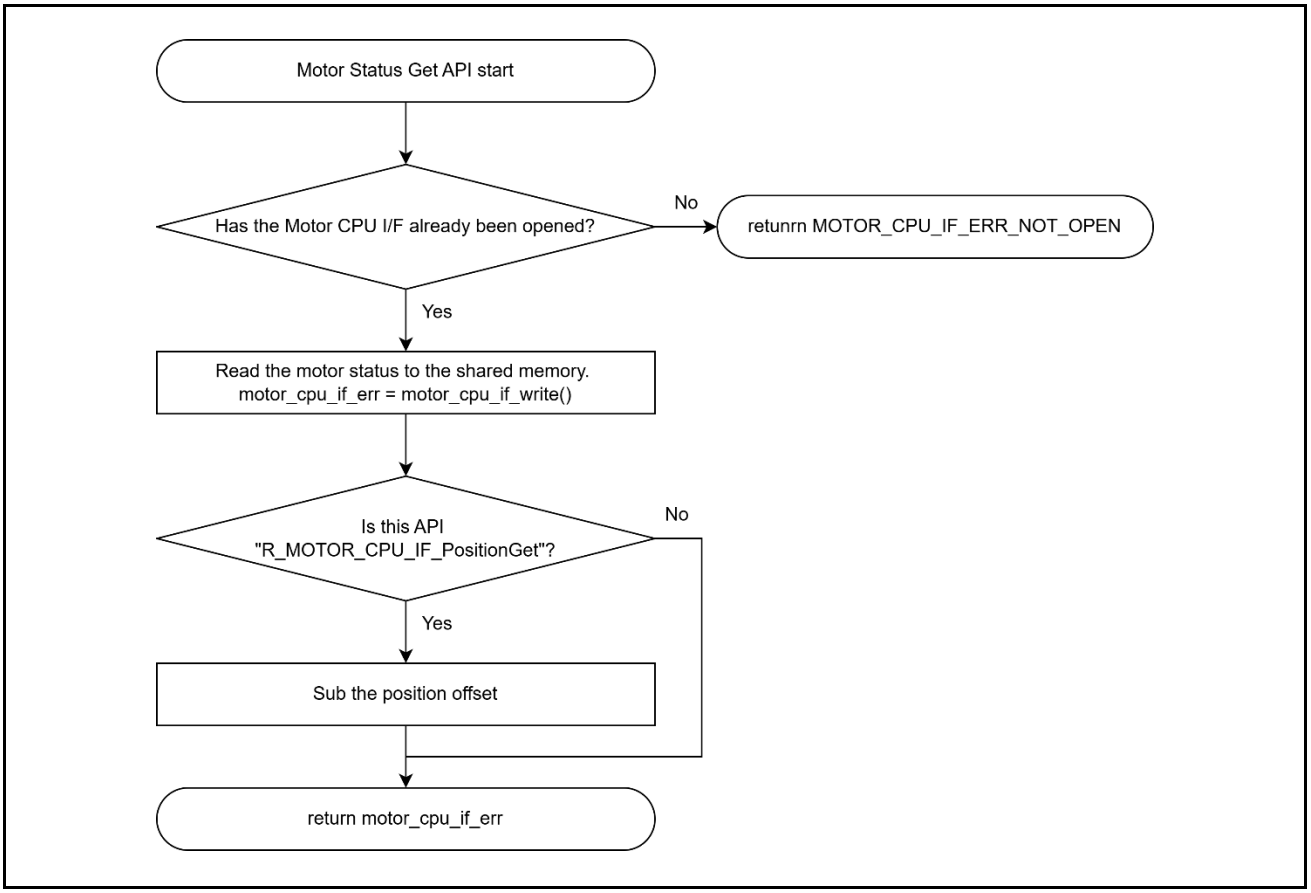


Figure 7.44. Flowchart of R_NETWORK_CPU_IF Motor Status Get API

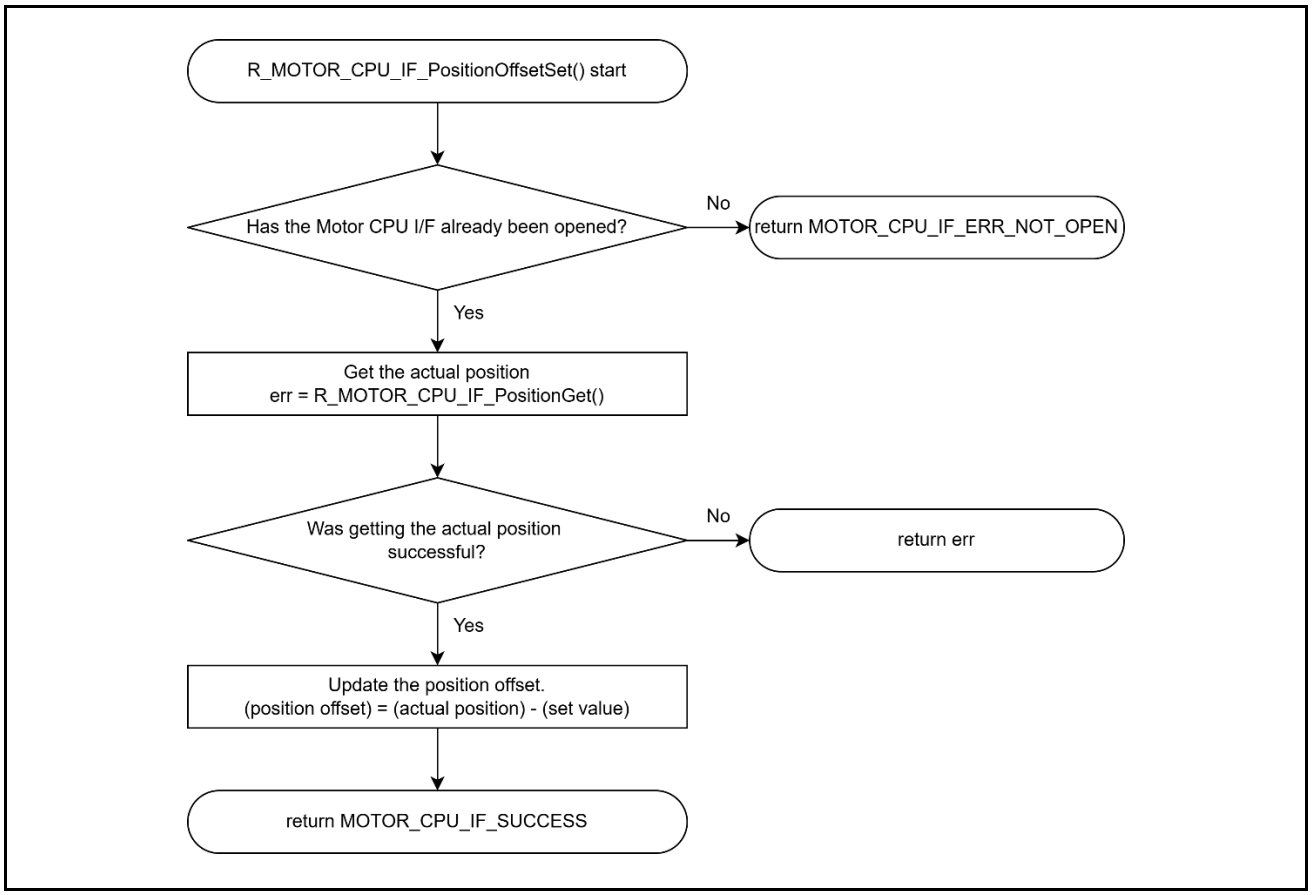


Figure 7.45. Flowchart of R_MOTOR_CPU_IF_PositionOffsetSet Function

7.3.8.4 Private Functions

Table 7.51. Motor CPU I/F Private Functions

Private Function	Description
motor_cpu_if_write	Writes data to Shared Memory. Flowchart: Figure 7.46
motor_cpu_if_read	Reads data from Shared Memory. Flowchart: Figure 7.47
motor_cpu_if_command_enqueue	Writes commands to the Command Queue. Flowchart: Figure 7.48

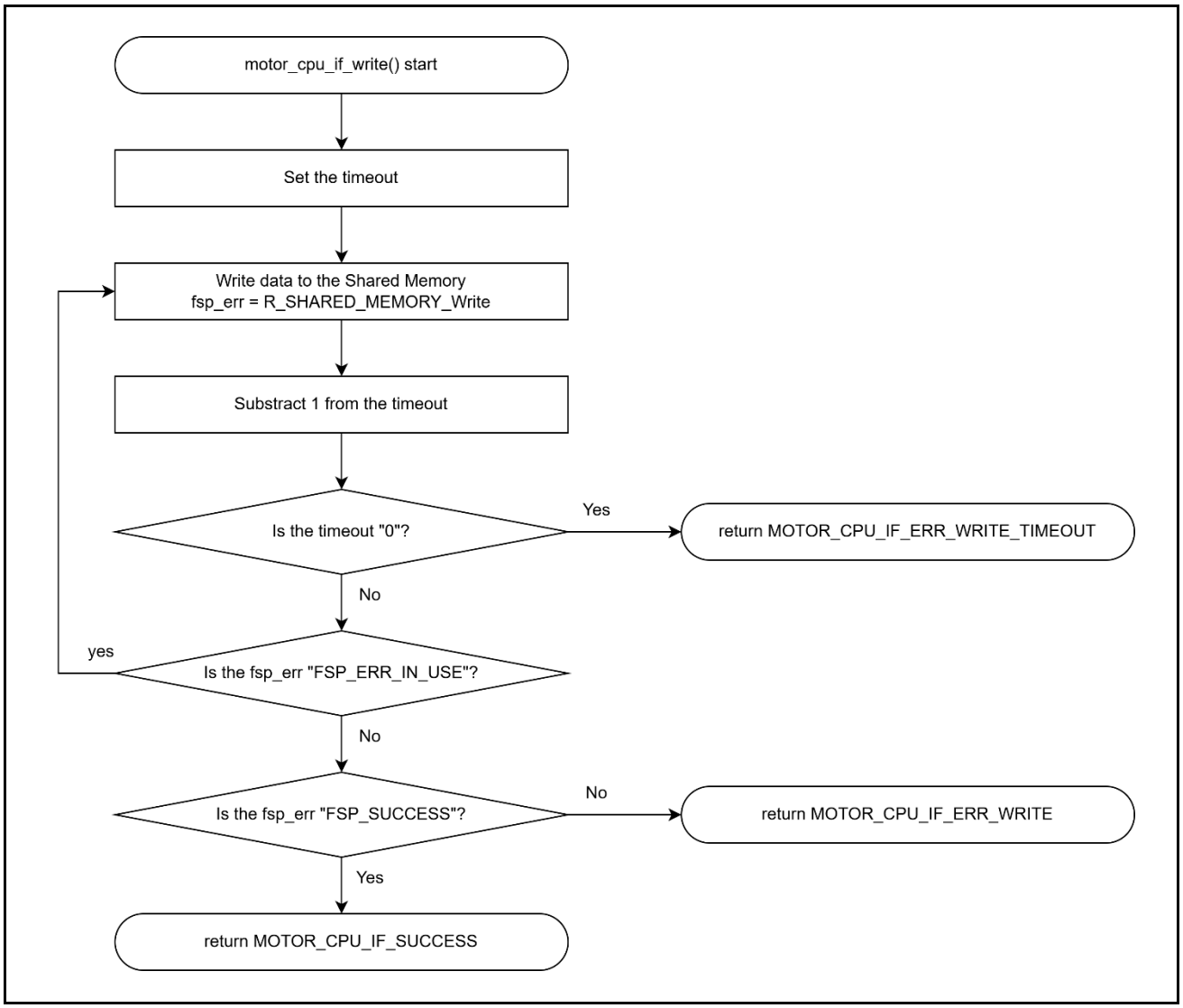


Figure 7.46. Flowchart of motor_cpu_if_write Function

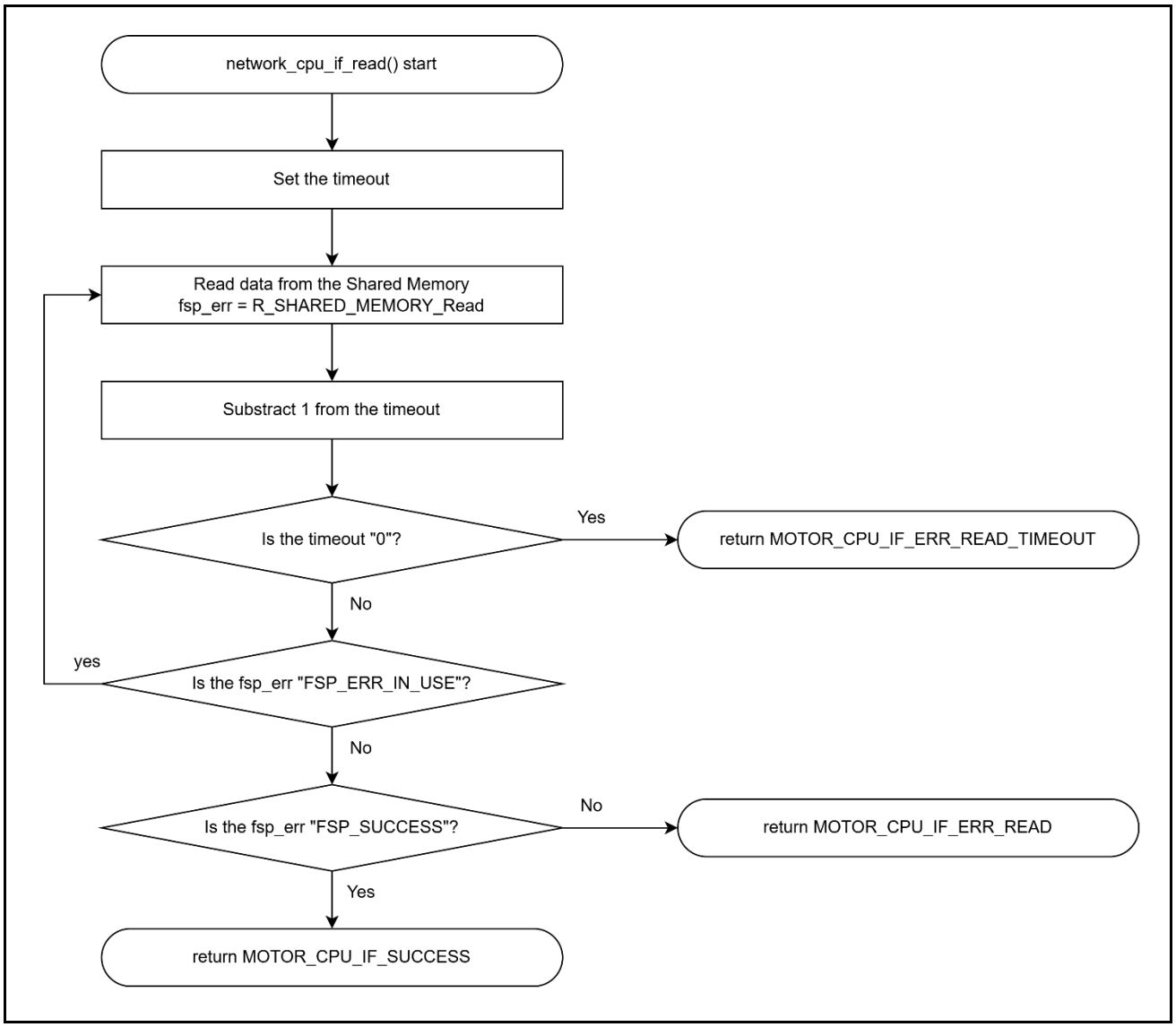


Figure 7.47. Flowchart of motor_cpu_if_read Function

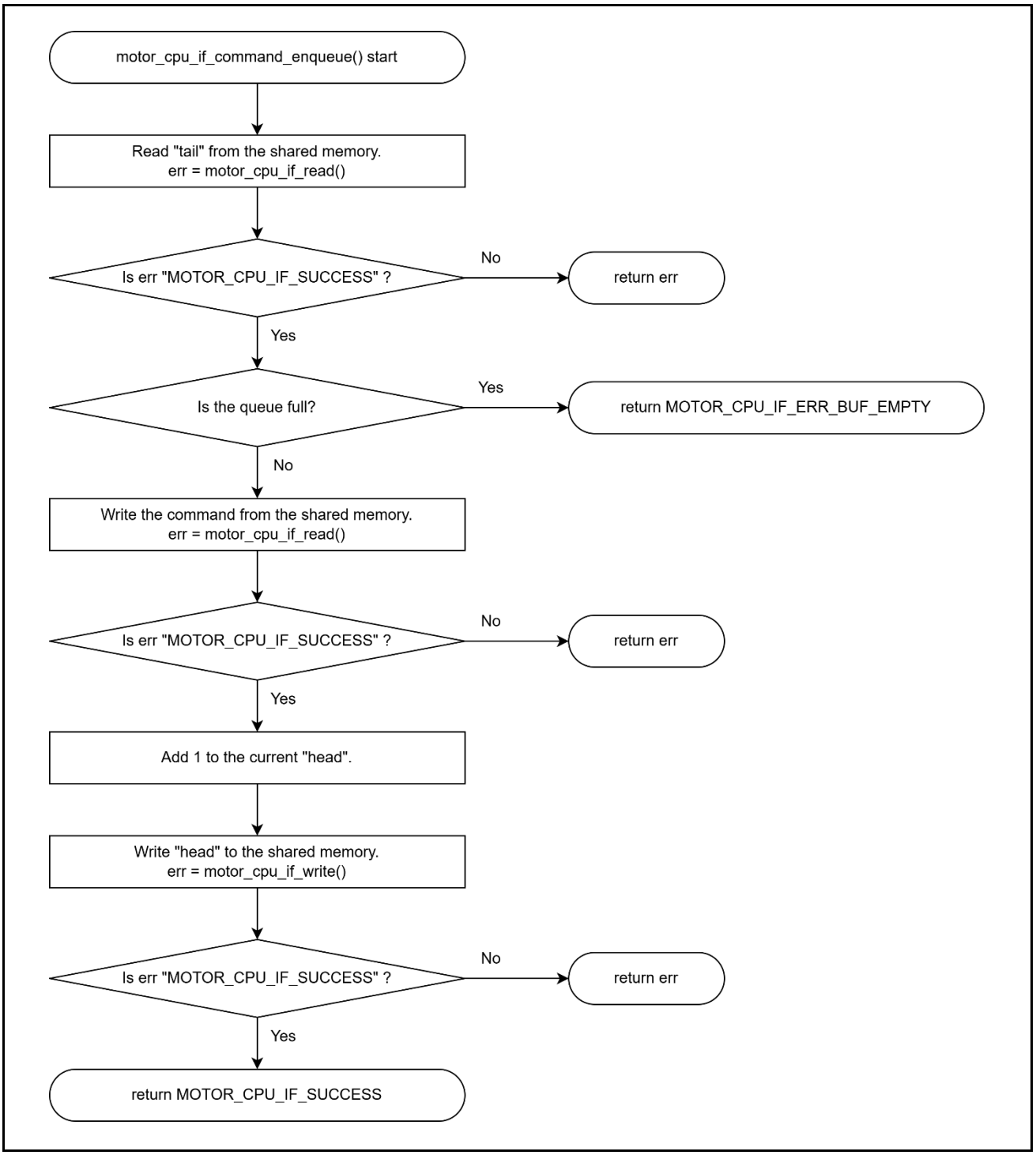


Figure 7.48. Flowchart of motor_cpu_if_command_enqueue Function

7.3.8.5 Callback Function

Table 7.52. Motor CPU I/F Callback Functions (r_motor_cpu_if.c)

Function Name	Description
shared_memory_callback	Callback function for Shared Memory Driver. This interrupt occurs when RZT2M CPU0 writes to Shared Memory. No processing is performed inside the callback function. Consider adding processing if necessary.

7.3.8.6 Macro Definitions

Table 7.53. Motor CPU I/F Macro Definitions (r_motor_cpu_if.h)

Macro Name	Definition Value	Remarks
CFG_CMD_Q_NUM	8	Number of command queues (must be a power of 2)
CFG_CMD_Q_OFFSET	0x0000	Offset address of Command Queue Area
CFG_CYCLIC_REF_AREA_OFFSET	0x0100	Offset address of Cyclic Reference Area
CFG_MTR_STATUS_AREA_OFFSET	0x0180	Offset address of Motor Status Area
CFG_MOTOR_CPU_IF_TIMEOUT	100	Timeout value used for Shared Memory read/write
CFG_SHARED_MEMORY_INSTANCE_ADDR	&g_shared_memory0	Pointer to Shared Memory driver instance
MOTOR_CPU_IF_CMD_DATA_MAX_BYTE	12	Maximum data size for the command queue

7.3.8.7 Enumerations

Table 7.54. Motor CPU I/F Enumerations (r_motor_cpu_if.h)

Enumerations	Definitions	Remarks
motor_cpu_if_err_t Motor CPU I/F Error Codes	MOTOR_CPU_IF_SUCCESS	No error
	MOTOR_CPU_IF_ERR_OPEN	Open error
	MOTOR_CPU_IF_ERR_CLOSE	Close error
	MOTOR_CPU_IF_ERR_PARAM	Parameter error
	MOTOR_CPU_IF_ERR_NOT_OPEN	Not open error
	MOTOR_CPU_IF_ERR_BUF_FULL	Command queue full
	MOTOR_CPU_IF_ERR_BUF_EMPTY	Command queue empty
	MOTOR_CPU_IF_ERR_WRITE	Write error
	MOTOR_CPU_IF_ERR_WRITE_TIMEOUT	Write timeout
	MOTOR_CPU_IF_ERR_READ	Read error
	MOTOR_CPU_IF_ERR_READ_TIMEOUT	Read timeout
	MOTOR_CPU_IF_ERR_EXCEED_PAYLOAD	Data exceeded

Enumerations	Definitions	Remarks
motor_cpu_if_status_t Motor CPU I/F Status	MOTOR_CPU_IF_STATUS_NOT_OPEN	Not open
	MOTOR_CPU_IF_STATUS_OPEN	Open
motor_cpu_if_cmd_t Motor CPU I/F Command Definitions	MOTOR_CPU_IF_CMD_NONE	No command
	MOTOR_CPU_IF_CMD_RESET	Used by Command Set API
	MOTOR_CPU_IF_CMD_MOTOR_START	
	MOTOR_CPU_IF_CMD_MOTOR_STOP	
	MOTOR_CPU_IF_CMD_MOTOR_RESET	
	MOTOR_CPU_IF_CMD_MOTOR_ERROR_CANCEL	
	MOTOR_CPU_IF_CMD_ERROR_SET	
	MOTOR_CPU_IF_CMD_CTRL_TYPE_SET	
	MOTOR_CPU_IF_CMD_POSITION_COMMAND_MODE_SET	
	MOTOR_CPU_IF_CMD_ROTOR_ANGLE_OFFSET_SET	
	MOTOR_CPU_IF_CMD_CURRENT_OFFSET_SET	
	MOTOR_CPU_IF_CMD_CURRENT_OPEN_LOOP_SET	
	MOTOR_CPU_IF_CMD_ACCEL_TIME_SET	
	MOTOR_CPU_IF_CMD_MAX_SPEED_RAD_SET	
motor_cpu_if_pos_mode_t Motor CPU I/F Position Control Mode	MOTOR_CPU_IF_POS_MODE_STEP	Step mode
	MOTOR_CPU_IF_POS_MODE_TRAPEZOID	Trapezoid mode
motor_cpu_if_ctrl_type_t Motor CPU I/F Motor Control Type	MOTOR_CPU_IF_CTRL_TYPE_POS	Position control
	MOTOR_CPU_IF_CTRL_TYPE_SPEED	Speed control
	MOTOR_CPU_IF_CTRL_TYPE_TORQUE	Torque control
	MOTOR_CPU_IF_CTRL_TYPE_VOLTAGE	Open-loop control
	MOTOR_CPU_IF_CTRL_TYPE_CURRENT	Current control
	MOTOR_CPU_IF_CTRL_TYPE_ADJ_CRNT_OFFSET	Adjust current offset
	MOTOR_CPU_IF_CTRL_TYPE_ADJ_POS_OFFSET	Adjust position offset
motor_cpu_if_motor_err_code_t Motor CPU I/F Motor Error Codes	MOTOR_CPU_IF_MOTOR_ERROR_NONE	No error
	MOTOR_CPU_IF_MOTOR_ERROR_OVER_CURRENT_HW	HW overcurrent error
	MOTOR_CPU_IF_MOTOR_ERROR_OVER_VOLTAGE	Overvoltage error
	MOTOR_CPU_IF_MOTOR_ERROR_OVER_SPEED	Overspeed error
	MOTOR_CPU_IF_MOTOR_ERROR_UNDER_VOLTAGE	Undervoltage error

	MOTOR_CPU_IF_MOTOR_ERROR_OVER_CURRENT_SW	SW overcurrent error
	MOTOR_CPU_IF_MOTOR_ERROR_OVER_TEMPERATURE	Inverter overheat error
	MOTOR_CPU_IF_MOTOR_ERROR_UNKNOWN	Unknown error
motor_cpu_if_motor_state_t Motor CPU I/F Motor Status	MOTOR_CPU_IF_STATEMACHINE_STATE_STOP	STOP mode
	MOTOR_CPU_IF_STATEMACHINE_STATE_RUN	RUN mode
	MOTOR_CPU_IF_STATEMACHINE_STATE_ERROR	ERROR mode

7.3.8.8 Structures & Variable Information

Table 7.55. Motor CPU I/F Structure Definitions (r_motor_cpu_if.h)

Structure	Variable	Description
st_current_offset_cfg_t Current offset configuration structure	f4_iu_offset	U-phase current offset
	f4_iv_offset	V-phase current offset
	f4_iw_offset	W-phase current offset
st_current_open_loop_cfg_t Open loop current configuration structure	f4_open_loop_current	Open loop current value
	f4_ref_speed_rpm	Reference speed (RPM)
motor_cpu_if_cmd_hdr_t command queue head/tail structure	head	Head of command queue
	tail	Tail of command queue
motor_cpu_if_cmd_q_t command queue structure	cmd	Command
	size	Data size
	data[MOTOR_CPU_IF_ CMD_DATA_MAX_BYTE]	Data array
motor_cpu_if_cyclic_ref_t Cyclic Reference structure	f4_ref_position	Reference position
	f4_ref_speed	Reference speed
	f4_ref_torque	Reference torque
motor_cpu_if_mtr_status_t Motor Status structure	f4_pos_deg	Current position
	f4_speed	Current speed
	u1_status	State machine
	reserved1	Unused area
	u2_error_status	Error status
	reserved2	Unused area
	u1_ctrl_loop_mode	LOOP MODE
	reserved3	Unused area
	u1_in_position	Position control completion status
	reserved4	Unused area

	u1_offset_cal_finished_flag	Offset measurement status
	reserved5	Unused area
motor_cpu_if_cfg_t Motor CPU I/F Configuration structure	p_shared_memory_instance	Pointer to Shared Memory instance
	timeout	Timeout value
	shr_mem_cmd_data_max_size	Maximum command data size
	shr_mem_cmd_q_num	Number of command queues
	shr_mem_cmd_q_mask	Command queue mask
	shr_mem_cmd_q_offset	Offset of Command Queue Area in Shared Memory
	shr_mem_cmd_q_size	Size of Command Queue Area in Shared Memory
	shr_mem_cmd_head_offset	Offset of command queue head area
	shr_mem_cmd_head_size	Size of command queue head area
	shr_mem_cmd_tail_offset	Offset of command queue tail area
	shr_mem_cmd_tail_size	Size of command queue tail area
	shr_mem_cyclic_ref_area_offset	Offset of Cyclic Reference Area
	shr_mem_cyclic_ref_area_size	Size of Cyclic Reference Area
	shr_mem_ref_pos_offset	Offset of reference position area
	shr_mem_ref_vel_offset	Offset of reference speed area
	shr_mem_ref_tor_offset	Offset of reference torque area
	shr_mem_mtr_status_area_offset	Offset of Motor Status Area
	shr_mem_mtr_status_area_size	Size of Motor Status Area
	shr_mem_mtr_pos_offset	Offset of current position area
	shr_mem_mtr_vel_offset	Offset of current speed area
	shr_mem_mtr_status_offset	Offset of state machine area
	shr_mem_mtr_err_status_offset	Offset of error status area
	shr_mem_mtr_ctrl_loop_mode_offset	Offset of LOOP MODE area
	shr_mem_mtr_in_pos_offset	Offset of position control completion area
	shr_mem_mtr_cal_finished_flag_offset	Offset of offset measurement status area
motor_cpu_if_ctrl_t Motor CPU I/F Control structure	status	Network CPU I/F status
	cmd_hdr	Instance of command queue head/tail structure
	timeout_count	Timeout occurrence count
	position_offset	Position
	p_cfg	Pointer to Network CPU I/F configuration structure

motor_cpu_if_api_t Motor CPU I/F API structure	Open	Open function
	Close	Close function
	Reset	Command send function
	MotorStart	
	MotorStop	
	MotorReset	
	MotorErrorCancel	
	ErrorSet	
	CtrlTypeSet	
	PositionCommandModeSet	
	RotorAngleOffsetSet	
	CurrentOffsetSet	
	CurrentOpenLoopSet	
	AccelTimeSet	
	PosProfMaxSpeedRadSet	
	PositionOffsetSet	Position offset setting function
	RefPositionSet	Reference value set function
	RefSpeedSet	
	RefTorqueSet	
	PositionGet	Motor status get function
	SpeedGet	
	StatusGet	
	ErrorStatusGet	
	LoopModeStatusGet	
	InPositionFlagGet	
	OffsetCalibrationStatusGet	

7.3.8.9 Global Variables

Table 7.56. Motor CPU I/F Global Variables (r_motor_cpu_if.c)

Type	Instance Name	Description
motor_cpu_if_ctrl_t	g_motor_cpu_if_ctrl	Motor CPU I/F control structure instance. Initialized by the **Open** function.
motor_cpu_if_cfg_t	g_motor_cpu_if_cfg	Motor CPU I/F configuration structure instance. Each member is initialized at instance creation. Values cannot be changed after initialization.
motor_cpu_if_api_t	g_motor_cpu_if_api	Motor CPU I/F API structure instance. Each member is initialized at instance creation. Values cannot be changed after initialization.

7.3.9 Fusa I/F

The FuSa I/F module communicates with the Functional Safety CPU (RZ/T2L-A) via UART.

For the specifications of the Functional Safety CPU and its communication, refer to:

"RZ Family FSoE Application Software Development Handbook (R30UZ0176JJ0110)".

7.3.9.1 Module Structure

The module structure diagram is shown in Figure 7.49.

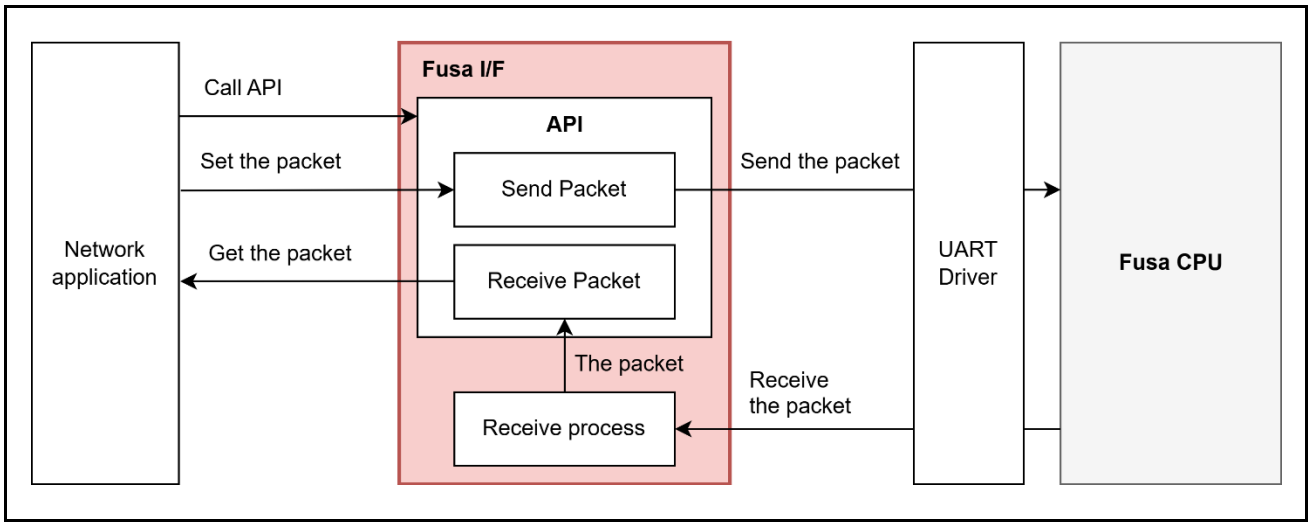


Figure 7.49. FuSa I/F Module Structure

7.3.9.2 Transmit/Receive Packets

In the FuSa I/F module, the command codes and size information of packets to be transmitted/received are registered in the FuSa I/F configuration structure instance (g_fusa_if_cfg).

The registered packets are listed below.

Table 7.57. List of Registered Receive Packets

#	Packet Name	Command Code	Size Info	Data Received
1	Start request	0x0101	0x0000	None
2	Safety Slave PDU	0x0202	0x0006	Objects listed in the Safety Slave PDU row of Table 7.46
3	Communication parameter	0x0203	0x000C	Objects listed in the Communication parameter row of Table 7.46

Table 7.58. List of Registered Transmit Packets

#	Packet Name	Command Code	Size Info	Data Transmitted
1	Acknowledgement of start request	0x0102	0x0000	None
2	Safety Master PDU	0x0201	0x0006	Objects listed in the Safety Master PDU row of Table 7.45

7.3.9.3 API

Table 7.59. FuSa I/F API List

API	Description
R_FUSA_IF_Open	Initializes the UART Driver. Also waits to receive the start request packet. when reception is complete, it transmits the acknowledgement of start request packet. Flowchart: Figure 7.50
R_FUSA_IF_Close	Terminates the FuSa I/F and the UART Driver. Flowchart: Figure 7.51
R_FUSA_IF_Send_Packet	Sends a packet to the FuSa CPU. Takes a pointer to a structure with the command code of the packet to be sent set in advance. If that command code is not registered in this module, the packet is not sent. Also, if the previous packet transmission is not completed, the packet is not sent. Flowchart: Figure 7.52
R_FUSA_IF_Receive_Packet	Retrieves a packet received from the FuSa CPU. Takes a pointer to a structure with the command code of the packet to be retrieved set in advance. If that command code is not registered in this module, the packet is not retrieved. Also, if reception of that packet has not completed, it cannot be retrieved. Flowchart: Figure 7.53

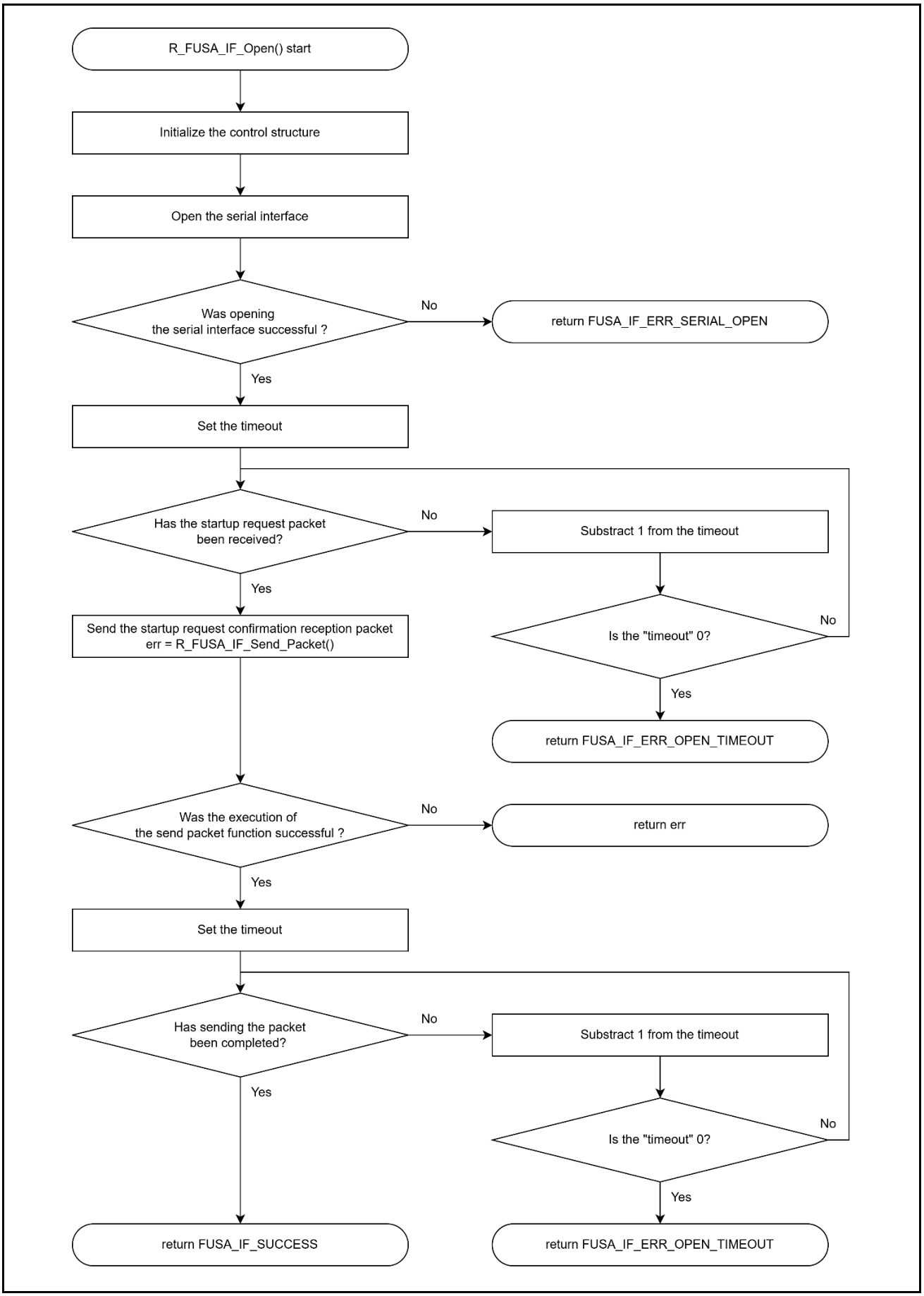


Figure 7.50. Flowchart of R_FUSA_IF_Open Function

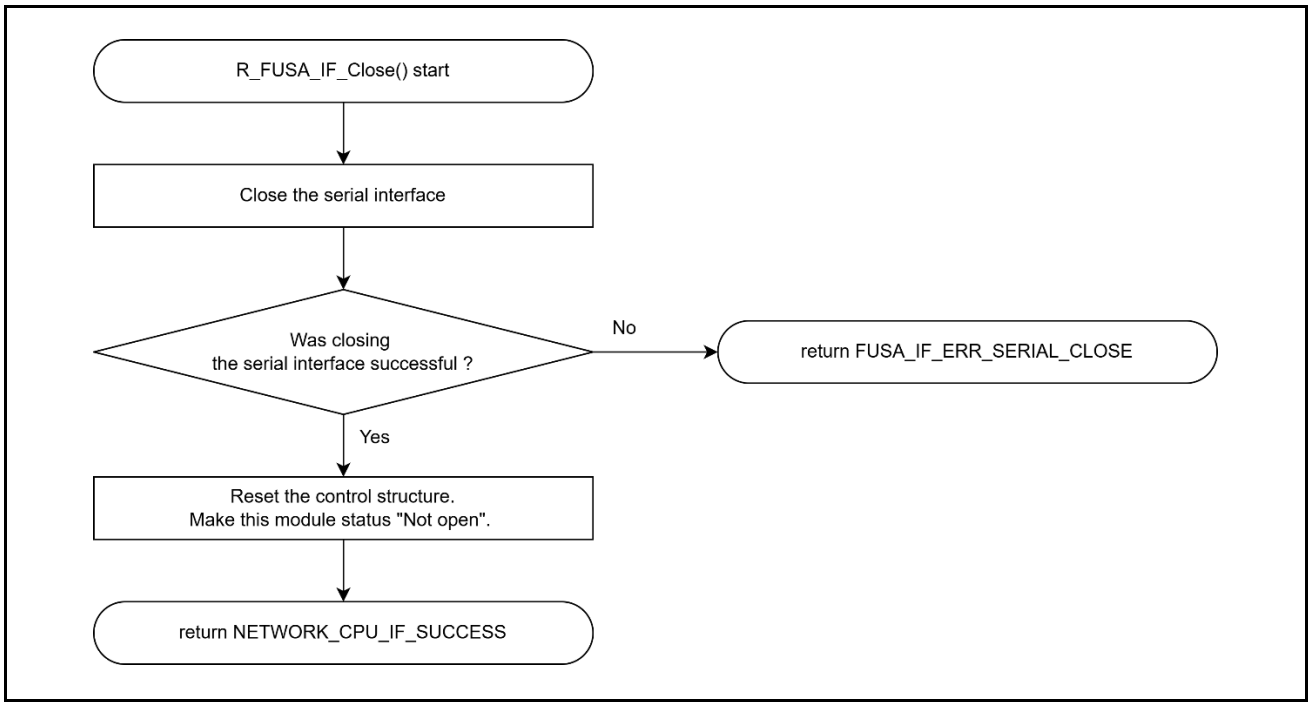


Figure 7.51. Flowchart of R_FUSA_IF_Close Function

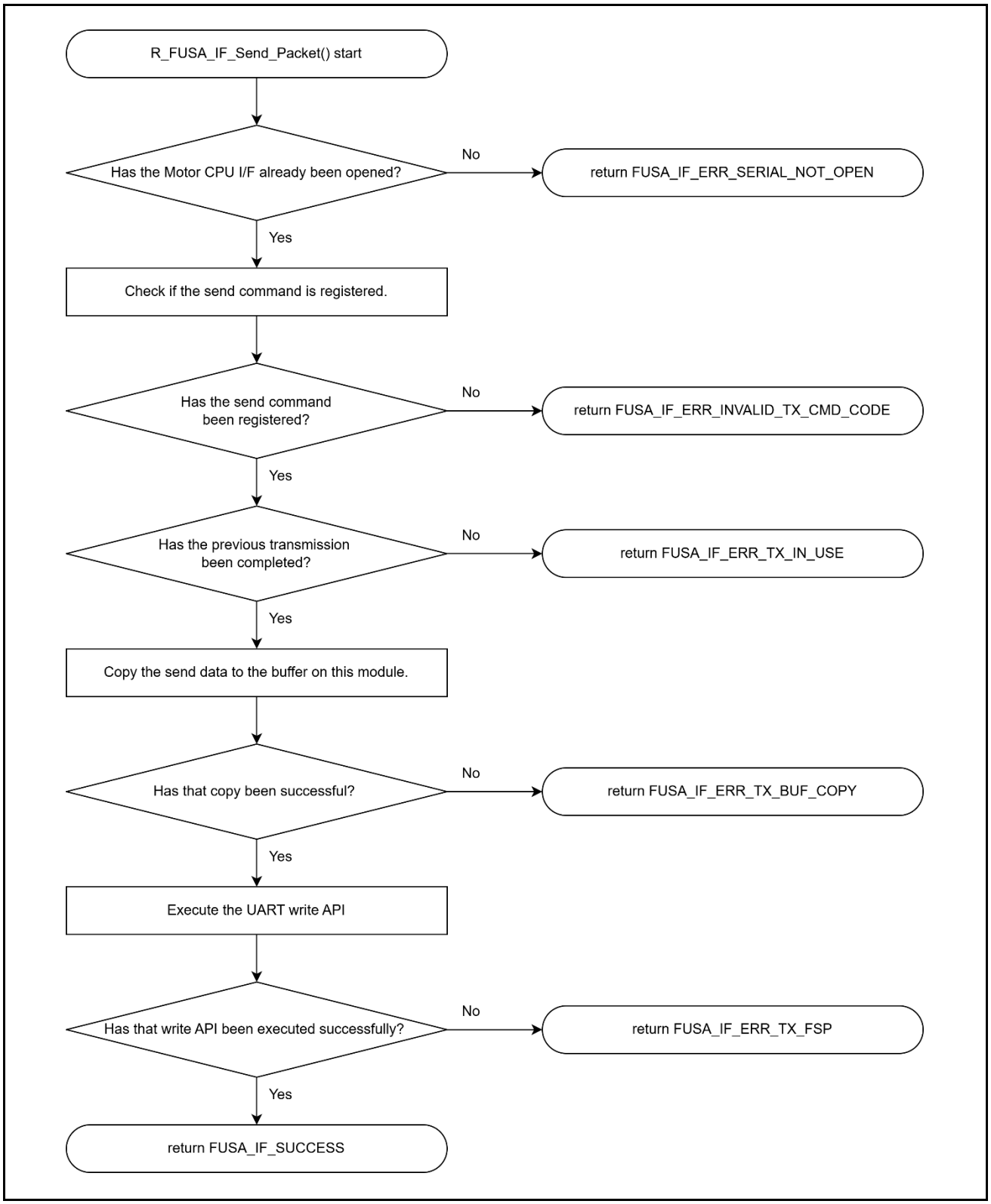


Figure 7.52. Flowchart of R_FUSA_IF_Send_Packet Function

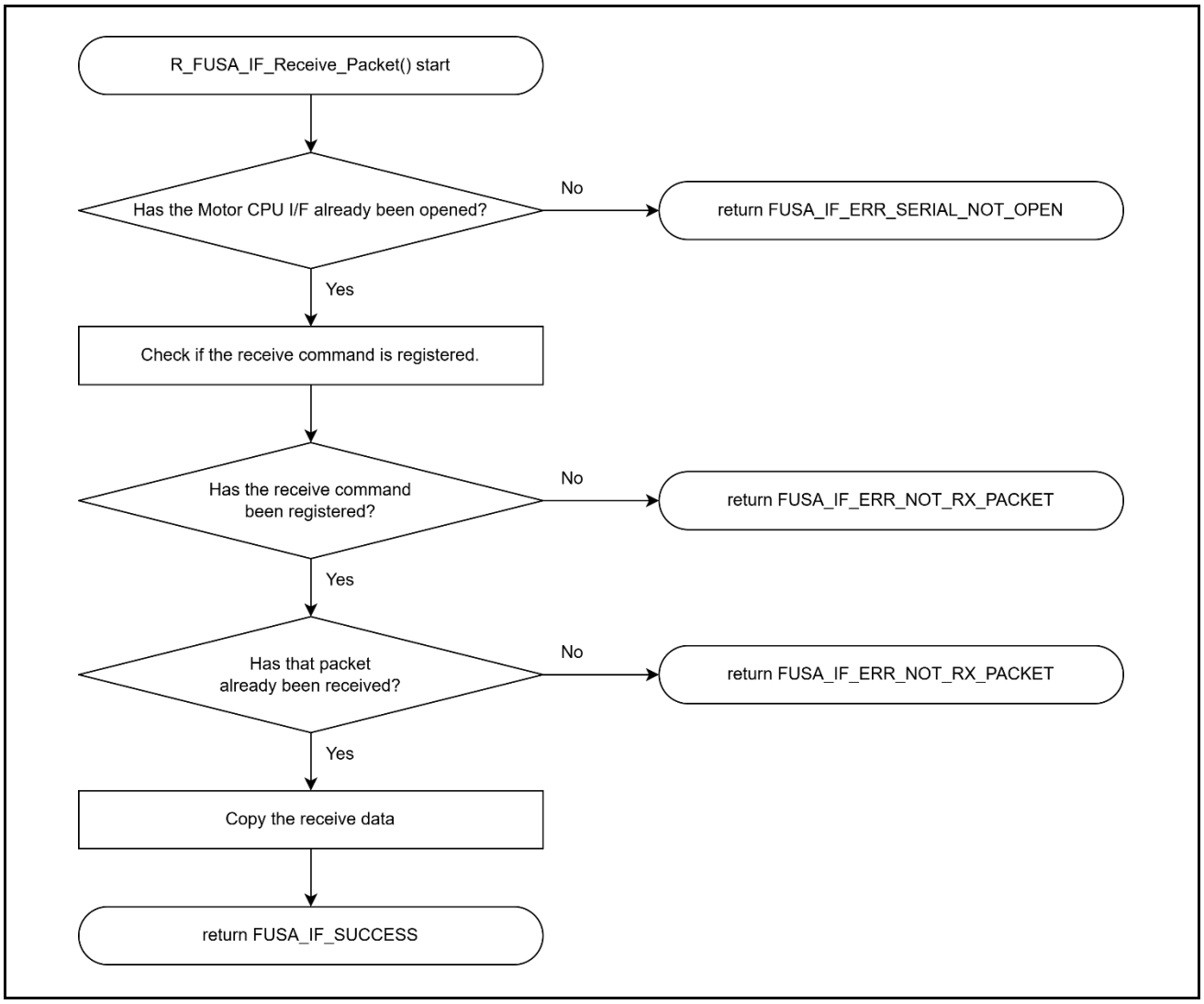


Figure 7.53. Flowchart of R_FUSA_IF_Receive_Packet Function

7.3.9.4 Private Function

Table 7.60. FuSa I/F Private Function List

Private Function	Description
fusa_if_receive_process	Parses the data received from the FuSa CPU and, if there are no issues, stores it as a packet.

7.3.9.5 Callback Function

Table 7.61. FuSa I/F Callback Function List (r_network_cpu_if.c)

Function Name	Description
fusa_if_callback	Callback function of the UART Driver. The following processing is performed inside the callback: (1) When one byte of data is received ● Calls fusa_if_receive_process function. (2) When transmission is completed ● Notifies FuSa I/F that transmission has been completed. Consider adding processing as needed.

7.3.9.6 Macro Definitions

Table 7.62. FuSa I/F Macro Definitions (r_fusa_if.h)

Macro Name	Definition Value	Remarks
CFG_FUSA_IF_UART_INSTANCE_ADDR	&g_uart0	Pointer to the UART driver instance
CFG_FUSA_IF_CMD_RX_NUM	3	Number of registered receive packets
CFG_FUSA_IF_CMD_TX_NUM	2	Number of registered transmit packets
CFG_FUSA_IF_CMDCODE_RX_START_REQ	0x0101	Command code for start request
CFG_FUSA_IF_CMDCODE_RX_PDU	0x0202	Command code for Safety Slave PDU
CFG_FUSA_IF_CMDCODE_RX_PARAM	0x0102	Command code for communication parameter
CFG_FUSA_IF_CMDCODE_TX_START_REQ	0x0102	Command code for start request acknowledgement
CFG_FUSA_IF_CMDCODE_TX_PDU	0x0201	Command code for Safety Master PDU
CFG_FUSA_IF_DATASIZE_RX_START_REQ	0	Data size of start request
CFG_FUSA_IF_DATASIZE_RX_PDU	6	Data size of Safety Slave PDU
CFG_FUSA_IF_DATASIZE_RX_PARAM	12	Data size of communication parameter
CFG_FUSA_IF_DATASIZE_TX_START_REQ	0	Data size of start request acknowledgement
CFG_FUSA_IF_DATASIZE_TX_PDU	6	Data size of Safety Master PDU
CFG_FUSA_IF_DATASIZE_MAX	CFG_FUSA_IF_DATASIZE_RX_PARAM	Maximum data size
CFG_FUSA_IF_RXCMD0_CMDCODE	CFG_FUSA_IF_CMDCODE_RX_START_REQ	Command code for receive packet 0
CFG_FUSA_IF_RXCMD0_DATASIZE	CFG_FUSA_IF_DATASIZE_RX_START_REQ	Data size for receive packet 0
CFG_FUSA_IF_RXCMD1_CMDCODE	CFG_FUSA_IF_CMDCODE_RX_PDU	Command code for receive packet 1
CFG_FUSA_IF_RXCMD1_DATASIZE	CFG_FUSA_IF_DATASIZE_RX_PDU	Data size for receive packet 1
CFG_FUSA_IF_RXCMD2_CMDCODE	CFG_FUSA_IF_CMDCODE_RX_PARAM	Command code for receive packet 2
CFG_FUSA_IF_RXCMD2_DATASIZE	CFG_FUSA_IF_DATASIZE_RX_PARAM	Data size for receive packet 2
CFG_FUSA_IF_TXCMD0_CMDCODE	CFG_FUSA_IF_CMDCODE_TX_START_REQ	Command code for transmit packet 0

CFG_FUSA_IF_TXCMD0_DATASIZE	CFG_FUSA_IF_DATASIZE_TX_START_REQ	Data size for transmit packet 0
CFG_FUSA_IF_TXCMD1_CMDCODE	CFG_FUSA_IF_CMDCODE_TX_PDU	Command code for transmit packet 1
CFG_FUSA_IF_TXCMD1_DATASIZE	CFG_FUSA_IF_DATASIZE_TX_PDU	Data size for transmit packet 1
CFG_OPEN_TIMEOUT	0x0FFFFFFF	Timeout value

7.3.9.7 Enumerations

Table 7.63. FuSa I/F Enumerations (r_fusa_if.h)

Enumeration	Definition	Remarks
fusa_if_err_t (FuSa I/F Error Codes)	FUSA_IF_SUCCESS	No error
	FUSA_IF_ERR_INVALID_PARAMETER	Parameter error
	FUSA_IF_ERR_SERIAL_OPEN	Open error
	FUSA_IF_ERR_SERIAL_CLOSE	Close error
	FUSA_IF_ERR_SERIAL_NOT_OPEN	Not open error
	FUSA_IF_ERR_INVALID_RX_CMD_CODE	Invalid receive command code
	FUSA_IF_ERR_NOT_RX_PACKET	No packet received
	FUSA_IF_ERR_INVALID_TX_CMD_CODE	Invalid transmit command code
	FUSA_IF_ERR_TX_IN_USE	Transmission in progress
	FUSA_IF_ERR_TX_BUF_COPY	Failed to copy to transmit buffer
	FUSA_IF_ERR_TX_FSP	Error occurred in FSP function
	FUSA_IF_ERR_OPEN_TIMEOUT	Open timeout
fusa_if_status_t (FuSa I/F Status)	FUSA_IF_STATUS_NOT_OPEN	Not open
	FUSA_IF_STATUS_OPEN	Open
fusa_if_rx_status_t (FuSa I/F Receive Status)	FUSA_IF_RX_NO_DATA	No data received
	FUSA_IF_RX_RECEIVED_CMD1	First byte of command code received
	FUSA_IF_RX_RECEIVED_CMD2	Command code received
	FUSA_IF_RX_RECEIVED_SIZE1	First byte of data size received
	FUSA_IF_RX_RECEIVED_SIZE2	Data size received

7.3.9.8 Structures & Variables

Table 7.64. FuSa I/F Structures (r_fusa_if.h)

Structure	Variable	Remarks
fusa_if_cmd_list_t (Command list structure)	command_code	Command code information
	data_size	Data size information
fusa_if_cfg_t (FuSa I/F Configuration structure)	p_uart_instance	Pointer to UART instance
	open_timeout	Timeout value
	rx_cmd_num	Number of receive packets
	tx_cmd_num	Number of transmit packets
	rx_cmd_list[CFG_FUSA_IF_CMD_RX_NUM]	Receive packet list
	tx_cmd_list[CFG_FUSA_IF_CMD_TX_NUM]	Transmit packet list
fusa_if_packet_t (FuSa I/F Packet structure)	command_code	Command code
	data_size	Data size
	data[CFG_FUSA_IF_DATASIZE_MAX]	Data
fusa_if_ctrl_t (FuSa I/F Control structure)	fusa_if_status	FuSa I/F status
	transfer_complete	Transmission complete flag
	tx_tmp	Transmit buffer
	rx_status	Receive status
	rx_data_pos	Receive data position
	rx_tmp	Receive buffer
	rx_buf[CFG_FUSA_IF_CMD_RX_NUM]	Storage buffers for each receive packet
	rx_complete_flag[CFG_FUSA_IF_CMD_RX_NUM]	Receive flags for each receive packet
fusa_if_api_t (FuSa I/F API structure)	Open	Open function
	Close	Close function
	send_packet	Send packet
	receive_packet	Receive packet

7.3.9.9 Global Variables

Table 7.65. FuSa I/F Global Variables (r_fusa_if.c)

Type	Instance Name	Description
fusa_if_ctrl_t	g_fusa_if_ctrl	FuSa I/F control structure instance. Initialized by the Open function.
fusa_if_cfg_t	g_fusa_if_cfg	FuSa I/F configuration structure instance. Each member is initialized at instance creation. Values cannot be changed after initialization.
fusa_if_api_t	g_fusa_if_api	FuSa I/F API structure instance. Each member is initialized at instance creation. Values cannot be changed after initialization.

7.4 Functional Safety Control (for RZ/T2L-A, RZ/T2L-B)

7.4.1 Overview

The functional overview of the RZ/T2L Functional Safety Control Software is shown below.

- Realization of HFT = 1 functional safety system featuring RZ/T2L
- Safety control operation for the motor controlled by RZ/T2M on the reference board (RTK0EF0190D01001BJ) in compliance with IEC61800-5-2 : STO, SS1-t.
- FSoE protocol processing compliant with ETG.5100ES (R) V1.2.0

This software uses the following functional safety solution products to realize safety unit with RZ/T2L:

1. RZ(CR52 core) Family Self-Test Software Kit : RTK0EF0102F01001SJ Ver.1.30
   including Diagnostic Software : Ver.1.01
2. RZ/T2L Group SIL3 system software kit : RTK0EF0154F01001SJ Ver.1.03
   including Functional Safety Platform Software : Ver.1.00
3. RZ Family FSoE Application Software Kit : RTK0EF0129F01001SJ Ver.1.12
   including FSoE Application Software : Ver.1.10

For details on these software products, please refer to the documentation included with each product.
Table 7.66 shows the folder structure of the RZ/T2L functional safety control software.

Table 7.66 Folder Structure of RZ/T2L Functional Safety Control Software71

#	Folder / file	Description
1	\RZT2L	RZ/T2L project folder
2	\FSoE_SW_M	RZ/T2L-A (Main) project folder
3	\RZT2L	
4	\PL-SW	
5	\PswLoaderFM	
6	\FSoE_SW_S	RZ/T2L-B (Sub) project folder
7	\RZT2L	
8	\PL-SW	
9	\PswLoaderFS	
10	\flashloader	Folder for EWARM files for writing ROM on RTK0EF0190D01001BJ
11	\arm	
12	\TwinCAT3	FSoE Master Project *
13	Safety_Drive.tfzip	
14	Safety_Drive.tpzip	

*: This project is available only on the system comprised of TwinCAT3 (Build 4026.19) as EtherCAT master and the following FSoE Masters manufactured by Beckhoff Automation. If this is not the case, the user must add a function described in section 7.4.7 to the FSoE Master.

- EK1100：EtherCAT Coupler
- EL6900：EtherCAT Terminal communication interface, TwinSAFE Logic
- EL9011：Bus end cover for E-bus contacts

7.4.2 Functional Overview

Table 7.67 shows functional overview of RZ/T2L functional safety control software.

Table 7.67 Functional Overview of RZ/T2L Functional Safety Control Software 72

73Item	Specification	Refer to
Motor safety controller unit:	Status check of motor stop mode selector switch (SW9_1 and SW11_1) and notification to the motor controller MPU (RZ/T2M)	7.4.4
	Release of motor safe state and notification to the RZ/T2M [Conditions for releasing motor from safety state] By meeting the following conditions, the user can release the motor from the safe state by pressing down two system boot switches (SW8 and SW10). The order of pressing switches does not matter. - FSoE communication is operated normally by the RZ/T2L and is in the FailSafeData state. - SafeData received from the FSoE Master shows that both pseudo emergency stop switches on the RTK0EF0190D01001BJ (SW9_3, SW11_3) are OFF.	7.4.5
	Motor safety control operation and notification to the RZ/T2M [Conditions for motor safety control operation] - SafeData received from the FSoE Master shows that pseudo emergency stop switches (SW9_3, SW11_3) are ON. - Transition of RZ/T2L FSoE to a state other than ProcessData. - An error has been detected by PL-SW diagnostics in the safety controller RZ/T2L.	7.4.6
FSoE processing unit:	Protocol processing compliant with ETG.5100ES (R) V1.2.0	-
	FSoE response performance: 339 µs	Figure 7.55
	FSoE slave address: 0x0010 (Fixed)	-
	SafeData size: 1 byte (PDU size: 6 bytes)	7.4.7
	SafeData specifications [Safety Slave PDU side] SafeData is formed from status of safety input connectors (CON17, CON18) and pseudo emergency switches (SW9_3, SW11_3). SafeData[7] = ! CON17_5 SafeData[6] = ! CON17_4 SafeData[5] = ! CON17_3 SafeData[4] = ! CON17_2 & SW9_3 SafeData[3] = ! CON18_5 SafeData[2] = ! CON18_4 SafeData[1] = ! CON18_3 SafeData[0] = ! CON18_2 & SW11_3 [Safety Master PDU side] Outputs the received SafeData directly to safety output connectors (CON15, CON16). SafeData[7] = CON15_5 SafeData[6] = CON15_4 SafeData[5] = CON15_3 SafeData[4] = CON15_2 SafeData[3] = CON16_5 SafeData[2] = CON16_4 SafeData[1] = CON16_3 SafeData[0] = CON16_2	
Others	Operational status of the safety unit is displayed with LEDs	7.4.8

7.4.3 Software Configuration

Figure 7.54 shows configuration of RZ/T2L functional safety control software. Hereinafter, Functional Safety Platform Software and FSoE Application Software are referred to as "PL-SW" and "FSoE Application" respectively.

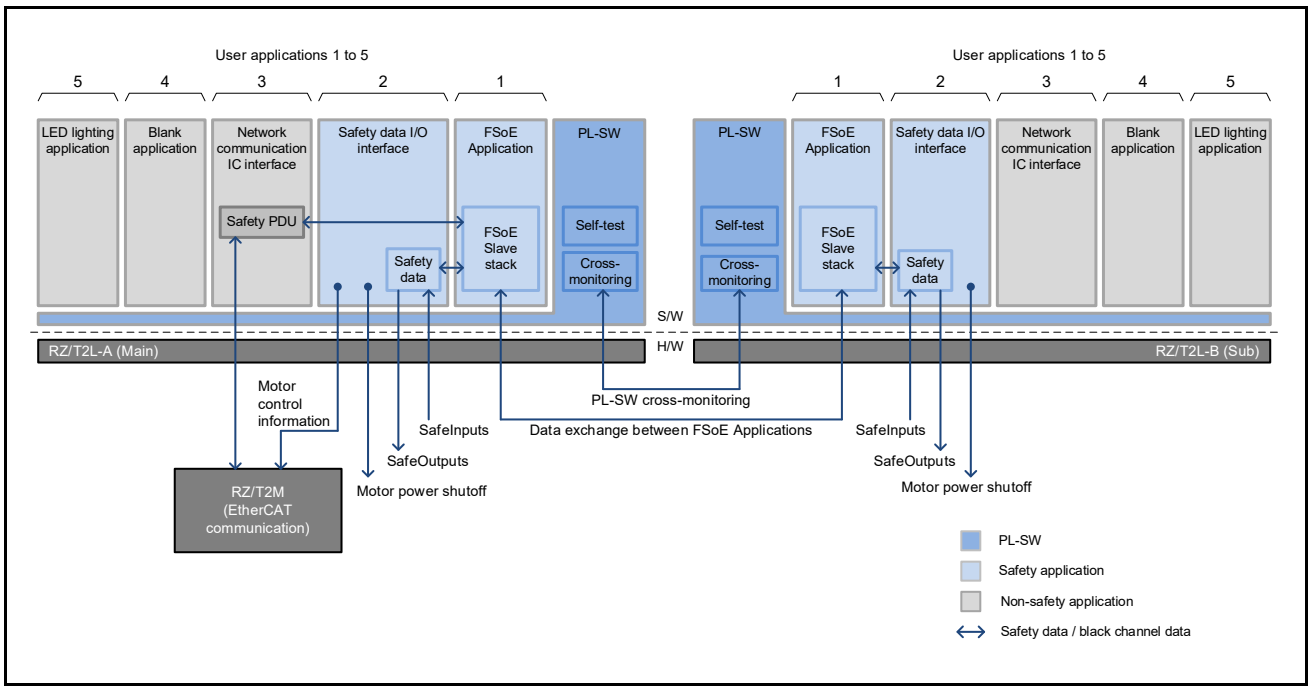


Figure 7.54 RZ/T2L Functional Safety Control Software Configuration71

Table 7.68 shows descriptions of user applications.

Table 7.68 User Application Description74

App No	Application	概要
1	FSoE Application	An application for protocol stack processing, which serially exchanges data with FSoE Application running on the counterpart RZ/T2L.
2	Safety data I/O interface	Software that controls safety input/output ports and exchanges safety data with FSoE Application. It also notifies of motor control information to RZ/T2M.
3	Network communication IC interface	[On RZ/T2L-A (Main)] Transmits and receives Safety PDUs to/from RZ/T2M that performs EtherCAT communication. Also exchanges Safety PDUs and communication parameters with FSoE Application. [On RZ/T2L-B (Sub)] Exchanges communication parameters with FSoE Application.
4	Blank application	An empty task reserved for user processing.
5	LED lighting application	An application for evaluation purpose that controls LEDs to display operation status of this software.

Figure 7.55 shows operation sequence of RZ/T2L functional safety control software. Operation sequence of user applications is controlled by PL-SW. In this sequence, FSoE data for one data communication is processed per PL-SW cycle.

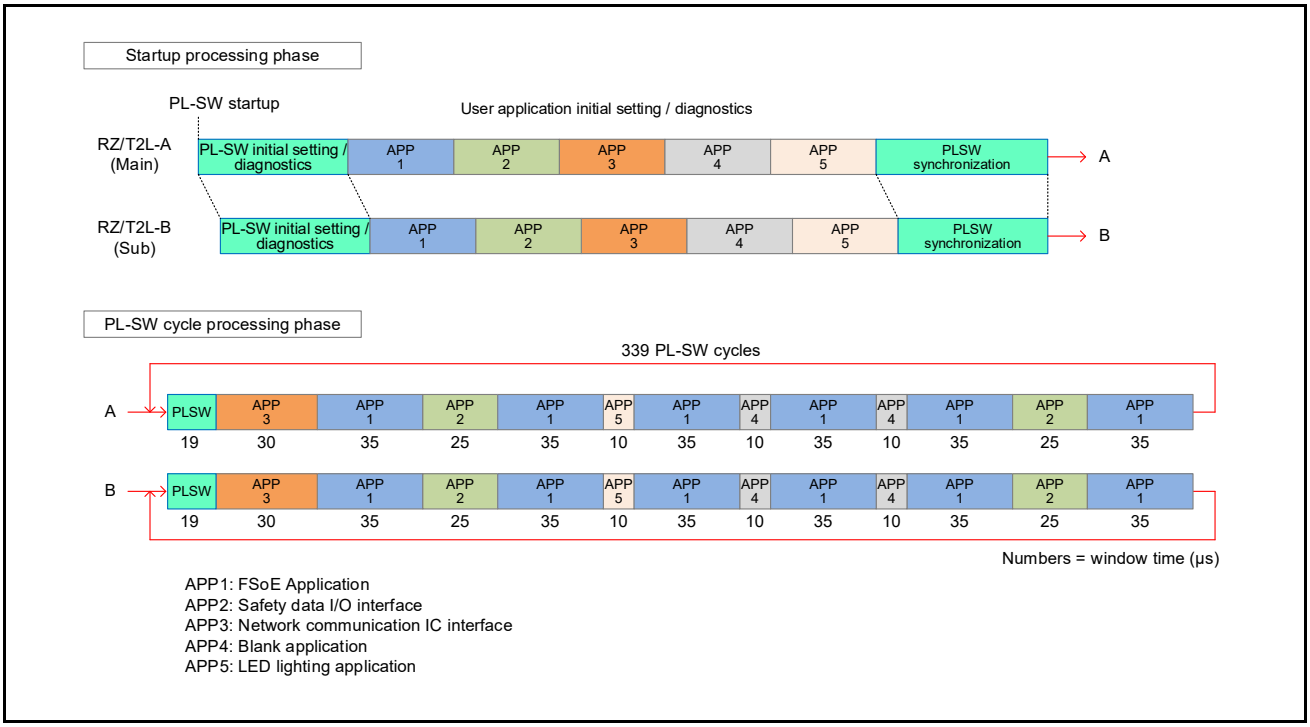


Figure 7.55 RZ/T2L Functional Safety Control Software Operation Sequence72

Table 7.69 shows addressing of PL-SW and user applications in the memory.

Table 7.69 PL-SW and User Applications Addressing75

Memory	Address	Software description	Subject to permanent fault test	Subject to ROM writing
ATCM	H'0000 0000 - H'0001 4FFF	PL-SW code	✓	✓
	H'0001 5000 - H'0001 AFFF	PL-SW data	✓	
	: :			
	H'0002 0000 - H'0002 4FFF	APP 1 code	✓	✓
	H'0002 5000 - H'0002 5FFF	APP 2 code	✓	✓
	H'0002 6000 - H'0002 6FFF	APP 3 code		✓
	H'0002 7000 - H'0002 7FFF	APP 4 code		✓
	H'0002 8000 - H'0002 8FFF	APP 5 code		✓
	H'0002 9000 - H'0003 3FFF	APP 6~16 code (Not used)		✓
	H'0003 4000 - H'0007 FFFF	Not used		
System SRAM	H'1000 0000 - H'1000 0FFF	APP 1 data	✓	
	H'1000 1000 - H'1000 1FFF	APP 2 data	✓	
	H'1000 2000 - H'1000 2FFF	APP 3 data		
	H'1000 3000 - H'1000 3FFF	APP 4 data		
	H'1000 4000 - H'1000 4FFF	APP 5 data		
	H'1000 5000 - H'1000 FFFF	APP 6~16 data (Not used)		
	: :	Not used		
	H'100F F000 - H'100F FFFF	PL-SW checksum		✓

Table 7.70 shows resources used by RZ/T2L functional safety control software. See user manual of the reference board (RTK0EF0190D01001BJ) for details about connectors (CONxx) and switches (SWxx).

Table 7.70 Resources used by RZ/T2L Safety Drive Software76

Item	Module used	Pin used	Application	
PL-SW	SCI1	TXD1 : P02_2 RXD1 : P02_0	Cross-monitoring (RZ/T2L-A ⇔ RZ/T2L-B)	
	GPT7, 9, 10	GTIOC10A : P13_2 GTETRGB : P24_0		
	CRC1	None	Code data test	
FSoE Application	SCI0	TXD0 : P16_0 RXD0 : P16_1	FSoE-related data exchange (RZ/T2L-A ⇔ RZ/T2L-B)	
	DMAC0 ch0-1	None		
	CRC0	None		
Data I/O interface	PORT	P01_3~0	SafeInputs	RZ/T2L-A: CON17_5~2 pins RZ/T2L-B: CON18_5~2 pins
		P13_7~4	SafeOutputs	RZ/T2L-A: CON15_5~2 pins RZ/T2L-B: CON16_5~2 pins
		P10_4	Motor stop mode (STO / SS1-t)	RZ/T2L-A: SW9_1 *1 RZ/T2L-B: SW11_1 *1
		P10_3	Not used	RZ/T2L-A: SW9_2 *2 RZ/T2L-B: SW11_2 *2
		P10_2	Pseudo emergency stop switch	RZ/T2L-A: SW9_3 *3 RZ/T2L-B: SW11_3 *3
		P10_0	System boot switch	RZ/T2L-A: SW8 RZ/T2L-B: SW10
		P14_3	Notification of motor stop mode (STO / SS1-t) to RZ/T2M	
		P14_2	Notification of motor operation permission to RZ/T2M	
		P07_2-1	Motor power shutoff	
Network communication IC interface	SCI4	TXD4 : P18_4 RXD4 : P18_5	Safety PDU exchange (RZ/T2M ⇔ RZ/T2L-A)	
	DMAC0 ch2-3	None		
LED lighting application	PORT	P17_7~5, P17_3 P21_7~6, P21_2~1	Control of status indicator LED	

*1: Operation modes are referenced on RZ/T2L startup. Any mode changes during operation will not be accepted.

*2: These switches must always be "OFF".

*3: Switches SW9_3 and SW11_3 must be "OFF" after the RZ/T2L has been released from reset.

7.4.4 Selecting Motor Stop Mode

Switches on the reference board (RTK0EF0190D01001BJ) allows users to select motor stop mode. On startup processing phase of PL-SW, the state of the mode selector switches is identified and reported to the RZ/T2M. Once reported, any further change made to switch setting will not be accepted.
Figure 7.56 shows signal control for motor stop mode.

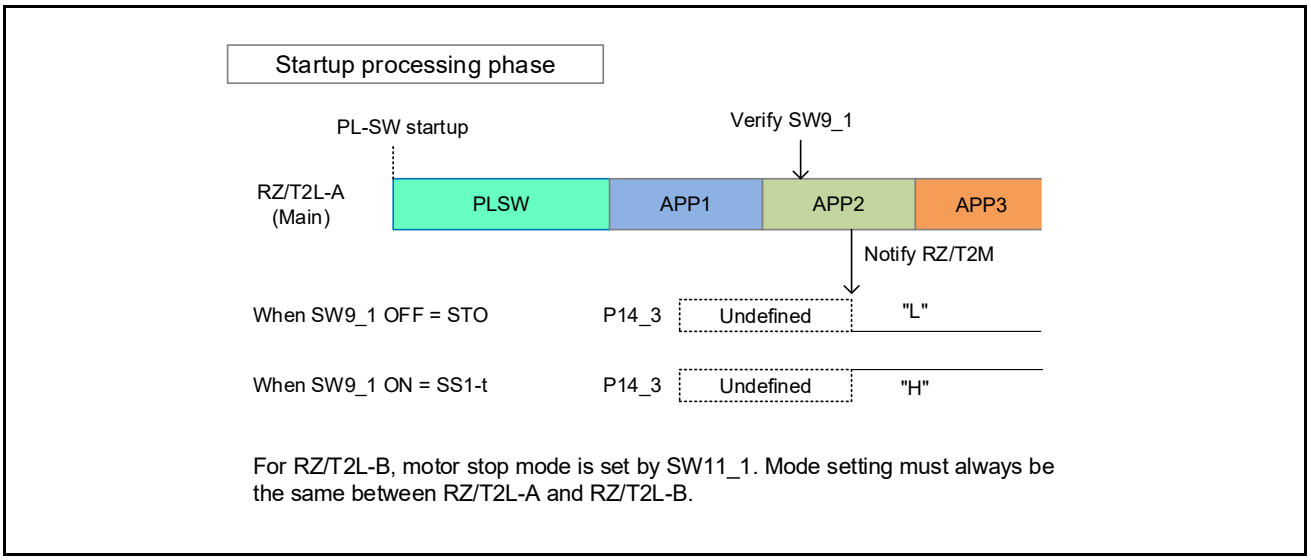


Figure 7.56 Signal Control for Motor Stop Mode73

7.4.5 Releasing Motor from Safe State

After the safety stop performed by RZ/T2L, motor can be released from the safe state when the following conditions are satisfied:

[Conditions]

1. FSoE communication is operated normally by the RZ/T2L and is in the FailSafeData state.
2. SafeData received from the FSoE Master shows that both pseudo emergency stop switches on the RTK0EF0190D01001BJ (SW9_3, SW11_3) are OFF (SafeData[4] = SafeData[0] = "1").
3. The above conditions are satisfied and two system boot switches (SW8 and SW10) have been pressed down. The order of pressing switches does not matter.

System boot switches are only sampled while above conditions 1 and 2 are satisfied. These conditions can be identified via status LEDs. See "7.4.8 Status LED Specifications" for details.
Figure 7.57 shows flow of releasing motor from the safety state.

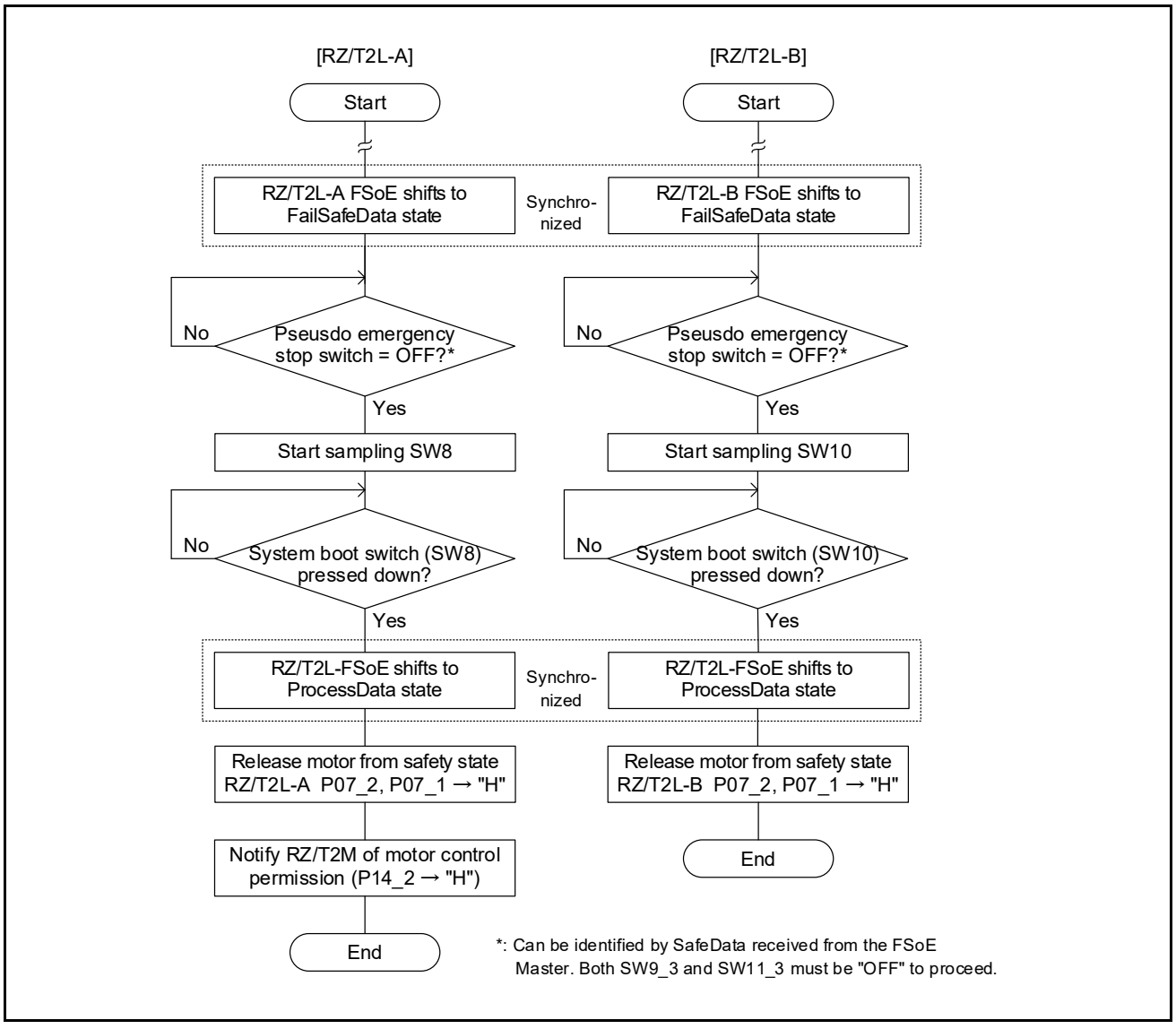


Figure 7.57 Safety State Release Flow74

7.4.6 Safety control operation

RZ/T2L stops the motor and starts safety control operation when the following conditions are satisfied:

[Conditions]

1. SafeData received from the FSoE Master shows that one or both pseudo emergency stop switch are ON (SafeData[4] = "0" or SafeData[0] = "0").
2. Transition of RZ/T2L FSoE to a state other than ProcessData due to FSoE communication error
3. An error has been detected by PL-SW diagnostics in the safety controller RZ/T2L.

Figure 7.58 shows sequence of motor safety control operation.

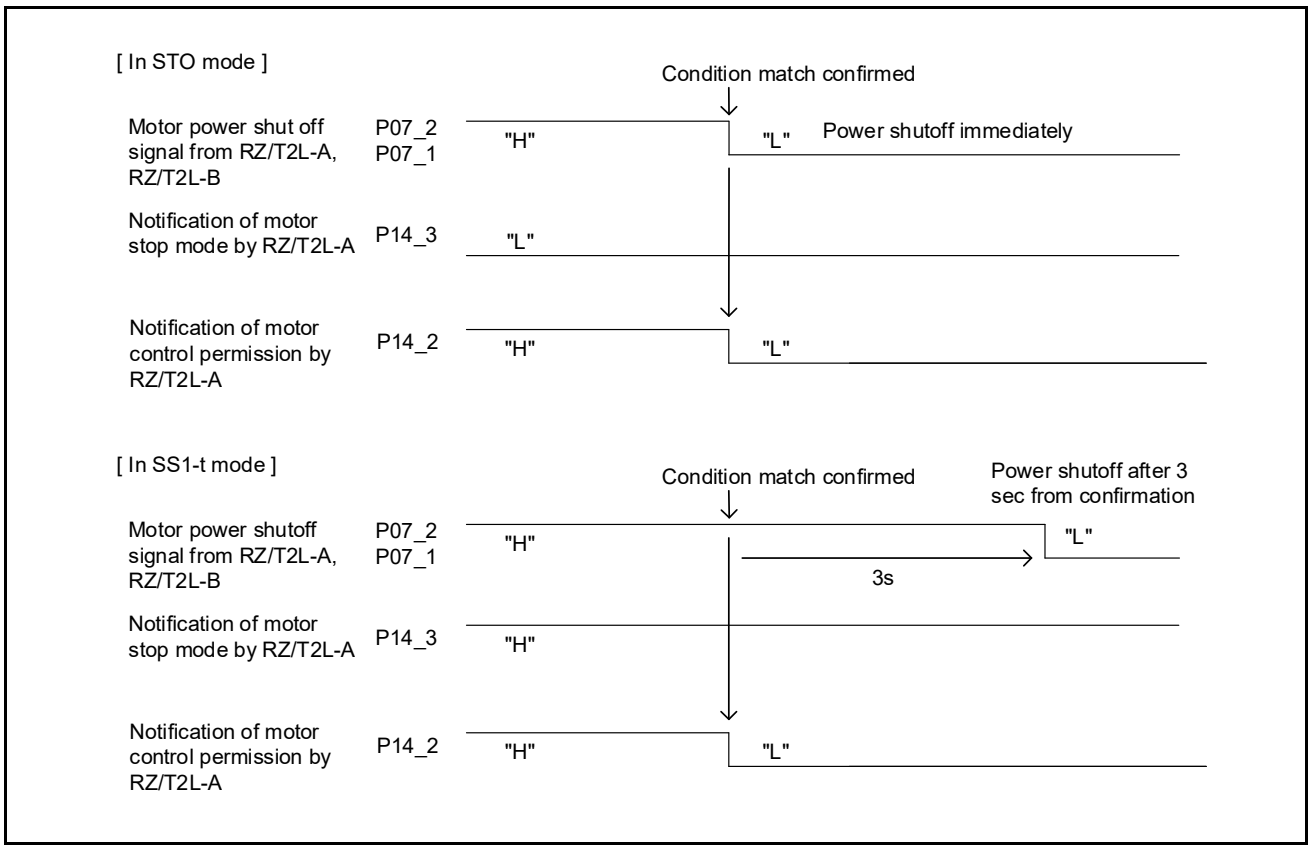
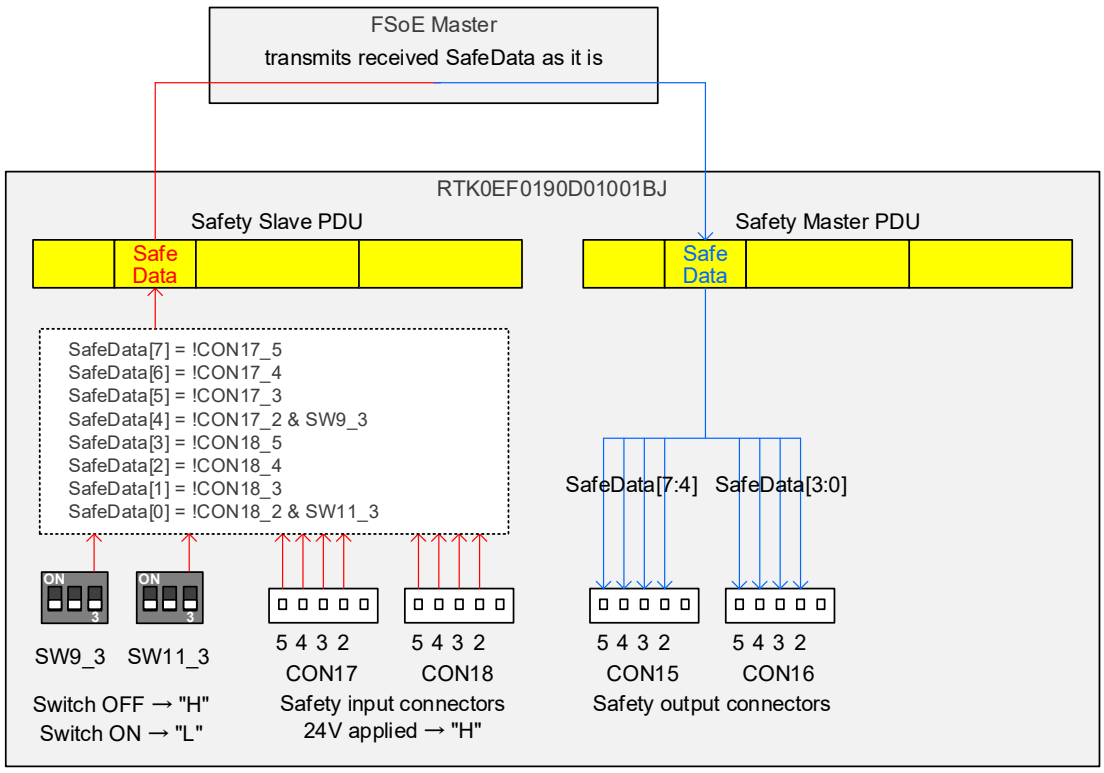


Figure 7.58 Safety Control Sequence75

7.4.7 SafeData Specifications

RZ/T2L functional safety control software generates SafeData in Safety Slave PDU from the input value of the safety input connectors (CON17, CON18) on the reference board (RTK0EF0190D01001BJ) and the value of on-board pseudo emergency stop switch (SW9_3, SW11_3). It also outputs the SafeData value in Safety Master PDU to safety output connectors (CON15, CON16) on the reference board. Note that this software is designed on the premise that SafeData received by the FSoE Master is output without changes. Figure 7.59 shows specifications of SafeData.



SafeData[7:0] generation example

Example 1: Both pseudo emergency stop switches = OFF, no voltage applied to safety input connectors
- SW9_3 = "H"
- SW11_3 = "H" → SafeData[7:0] = H'FF
- CON17_5~2 = "LLLL"
- CON18_5~2 = "LLLL"

Example 2: Pseudo emergency stop switch SW9_3 = ON, SW11_3 = OFF, no voltage applied to safety input connectors
- SW9_3 = "L"
- SW11_3 = "H" → SafeData[7:0] = H'EF
- CON17_5~2 = "LLLL"
- CON18_5~2 = "LLLL"

Example 3: Both pseudo emergency stop switches = OFF, 24 V applied to safety input connector CON18_5
- SW9_3 = "H"
- SW11_3 = "H" → SafeData[7:0] = H'F7
- CON17_5~2 = "LLLL"
- CON18_5~2 = "HLLL"

Figure 7.59 SafeData Specifications76

7.4.8 Status LED Specifications

Operation status of RZ/T2L functional safety control software can be identified through LED on the reference board (RTK0EF0190D01001BJ).

Figure 7.60 shows specifications of status LED.

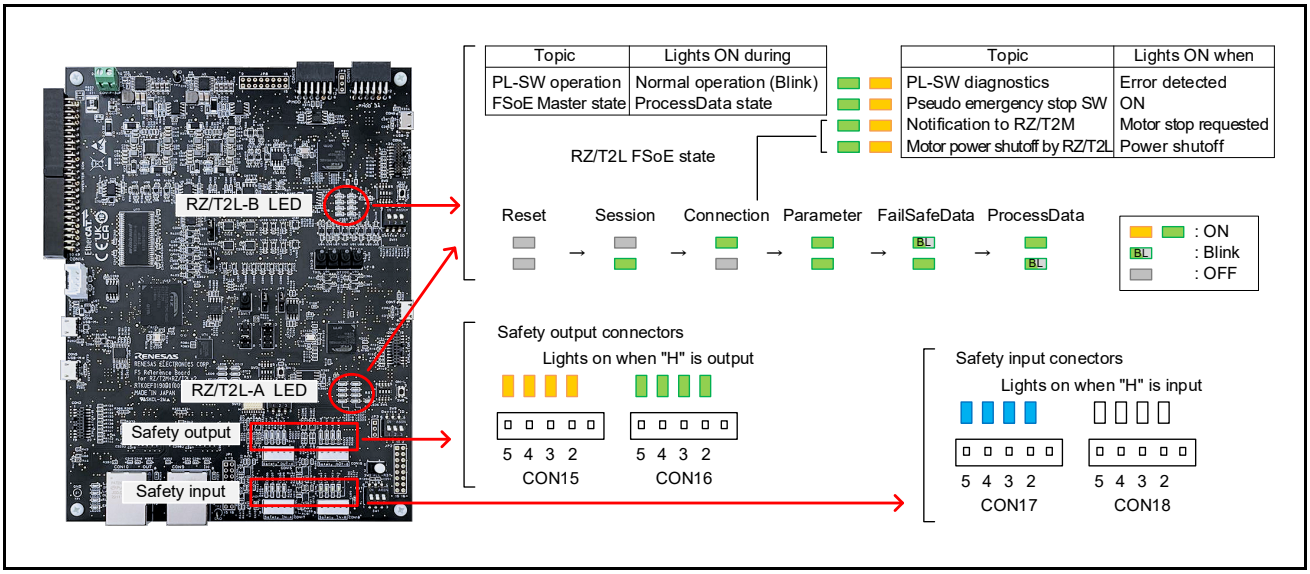


Figure 7.60 Status LED Specifications77

The conditions of pressing down the system boot switch can be identified through status LEDs.

Figure 7.61 shows LED indication when the motor is released from the safety state.

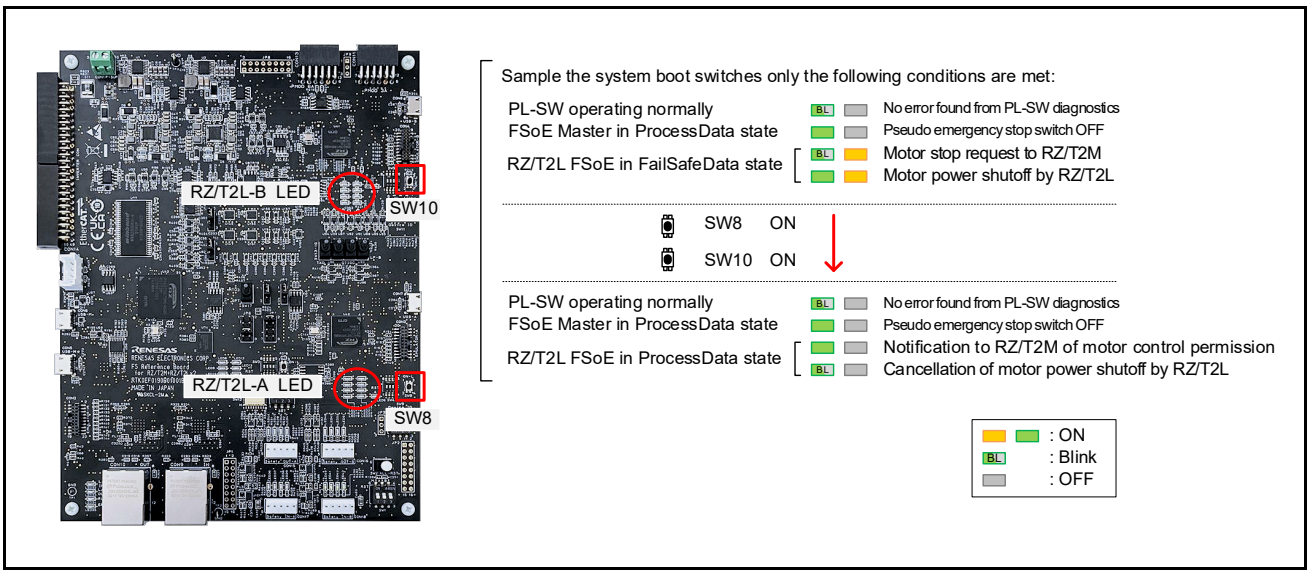


Figure 7.61 LED Indication on Releasing Safety State78

8. Appendix

8.1 Integrated development environment Installation

8.1.1 EWARM

Please download from the website of IAR Systems.
https://www.iar.com/products#/search?architecture=Arm

8.1.2 e^2 studio

Please download and install the latest e^2 studio from the following URL.
However, if you have already installed the latest e^2 studio, specify a different folder to install.

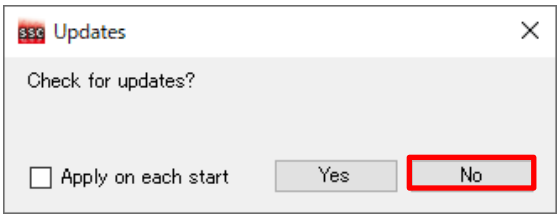
https://github.com/renesas/rzt-fsp

8.2 Appendix : A point of caution when using SSC Tool

※ When opening SSC Tool

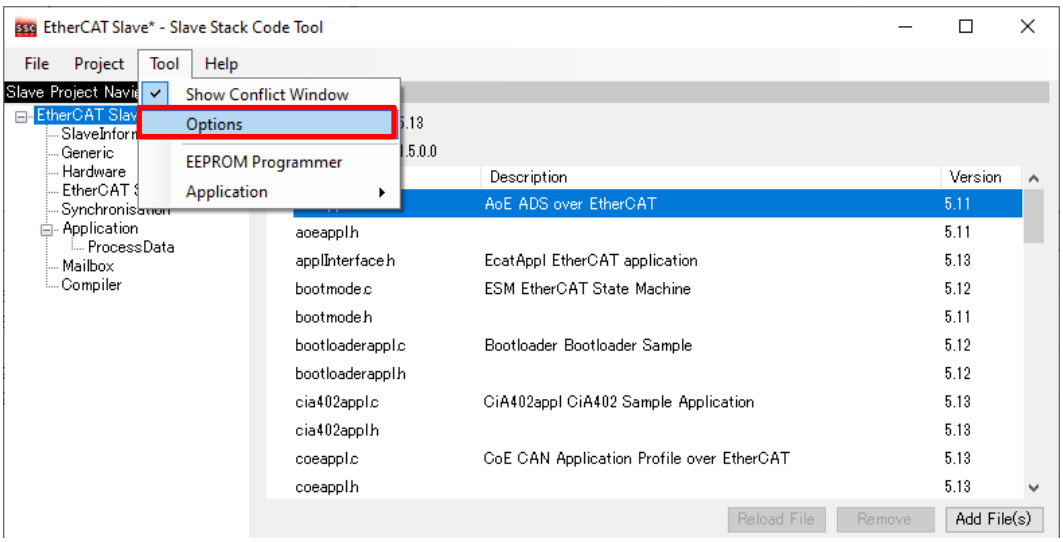
1. Please open the SSC Tool as an Administrator.
   Otherwise, generating SSC code may fail.

2. When opening the SSC Tool for the first time, the following window may be displayed.
   Please select [No] to not check for updates.

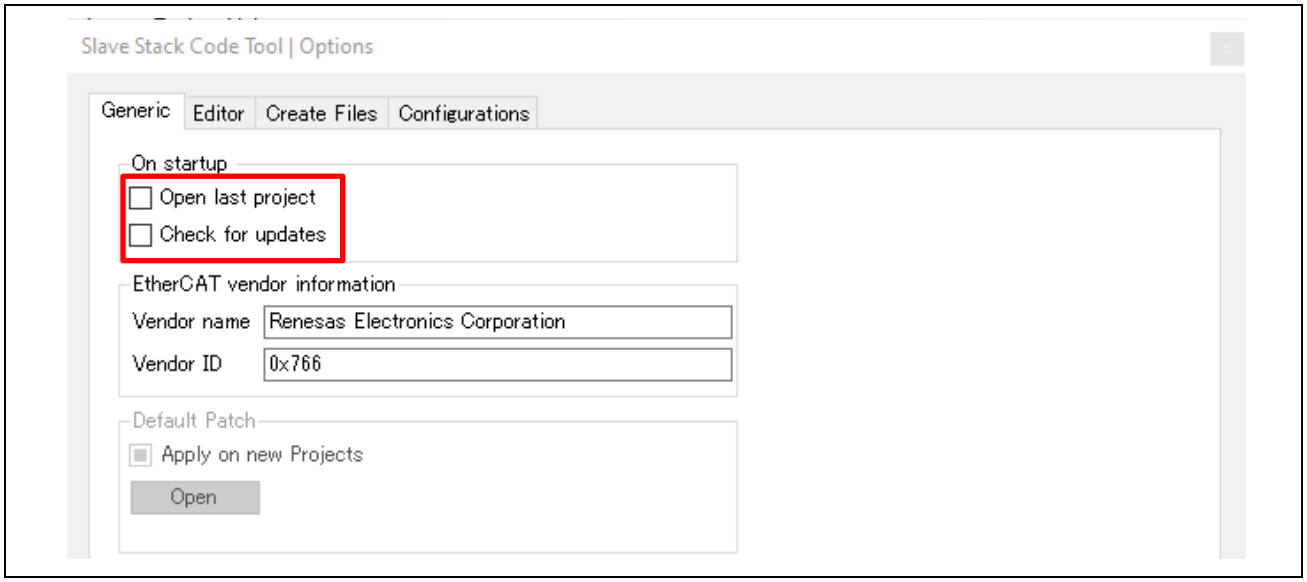


※ Regarding SSC Tool settings

3. Please go to [Tool] > [Options] to confirm SSC Tool settings.



4. Under the [Generic] tab, uncheck [Open last project] and [Check for updates].



5. Under the [Create Files] tab, uncheck [Create device description (ESI)].

6. Click [OK] to apply the updated settings.
   This concludes with the SSC Tool setup for this sample program.

8.3 Appendix : How to install patch

There are two methods as follows:

1. Via Git for Windows (64bit)
2. Via MinGW Installation Manager

8.3.1 Via Git for Windows (64bit)

This section describes how to install patch via Git for Windows.

1. Download the installer (e.g., Git-x.xx.x-64-bit.exe) from the official Git for Windows website.

   Git for Windows

2. Run the downloaded installer and follow the setup instructions. Use the default settings unless you have specific requirements.

3. After installation, add the path to patch.exe to your system's environment variables.
   For a default installation, the path is typically:

   "C:\Users\<your-username>\AppData\Local\Programs\Git\usr\bin"

   To apply the changes, restart your computer after updating the environment variables.

4. Start Command Prompt, enter "where patch".
   If the path to patch.exe is displayed, the installation was successful.

8.3.2 Via MinGW Installation Manager

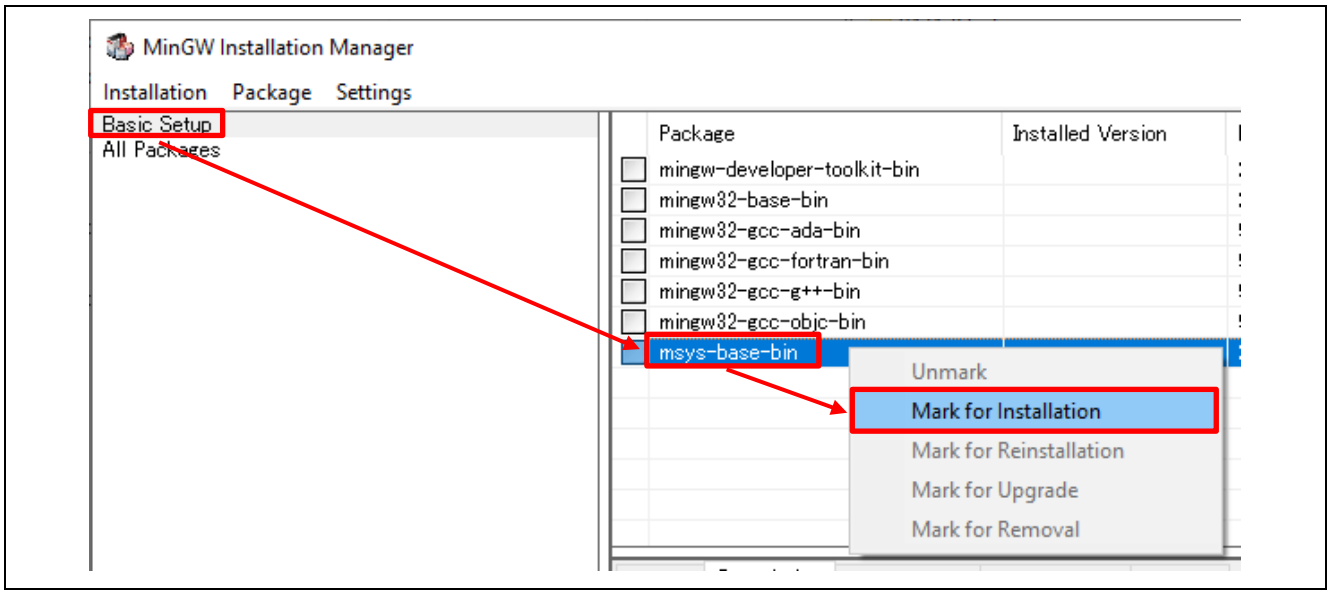
This section describes how to install patch via MinGW Installation Manager.

1. Download “mingw-get-setup.exe” from the following URL.

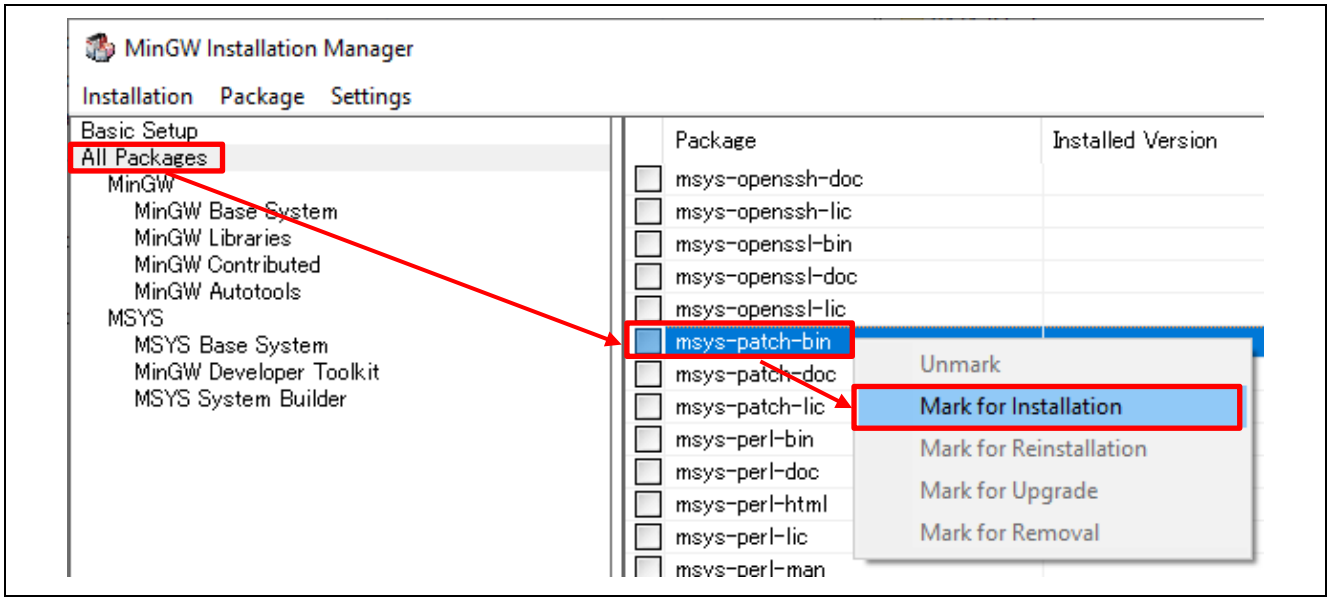
   https://sourceforge.net/projects/mingw/

2. Execute “mingw-get-setup.exe”, install “Mingw-installation-manager” according to the dialog.

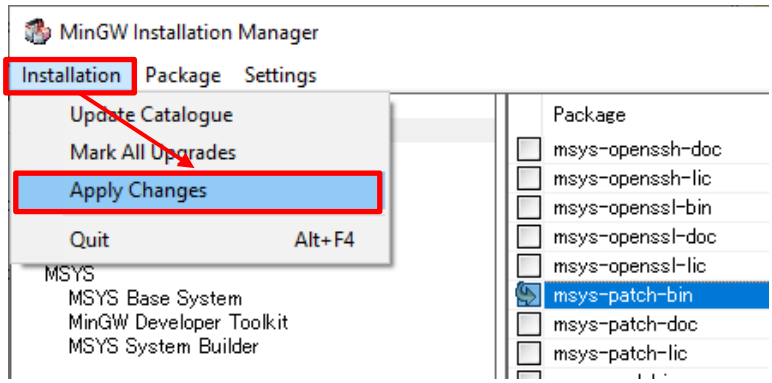
3. If It’s completed and the Mingw-installation-manager window is displayed, select “Basic Setup” in the left window, right-click on “msys-base-bin” in the right window, and select “Mark for Installation”.



4. Select “All Packages” in the left window, right-click on “msys-patch-bin” in the right window, and select “Mark for Installation”.



5. Select “Apply Changes” in “Installation” in the above menubar.



6. “Schedule of Pending Actions” window is displayed, click “Apply” button.

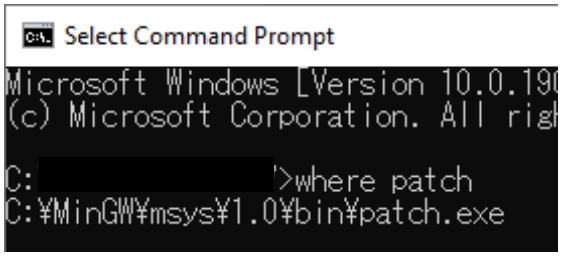
7. If “All changes were applied successfully; you may now clone this dialogue.” is displayed, Installing patch.exe is succeeded.

8. Add the path to the installed patch.exe to the system environment variables.
   For example, add the following path in the case of default path.

   “C:\MinGW\msys\1.0\bin”

   After updating the system environment variables, restart your computer to apply the changes.

9. Start Command Prompt, enter “where patch”.
   If the path to the installed patch.exe is displayed, there are no problem.



8.4 Appendix: Procedure for Creating the FSoE Master Program

This section describes how to create an FSoE master program equivalent to the one prepared in Section 6.2.

(1) Creating the TwinSAFE Project

1. In the TwinCAT 3 System Manager tree, right-click [SAFETY].
2. Select [Add New Item] to create a new project.
3. Choose [TwinCAT Safety Project Preconfigured ErrAck].
4. Edit the project name as desired. In this document, the project name will be “safety_project_1” for explanation purposes.
5. Click [Add].
6. Set Target System to [Hardware Safety PLC] and Programming Language to [Graphical Editor].
7. Click [OK].

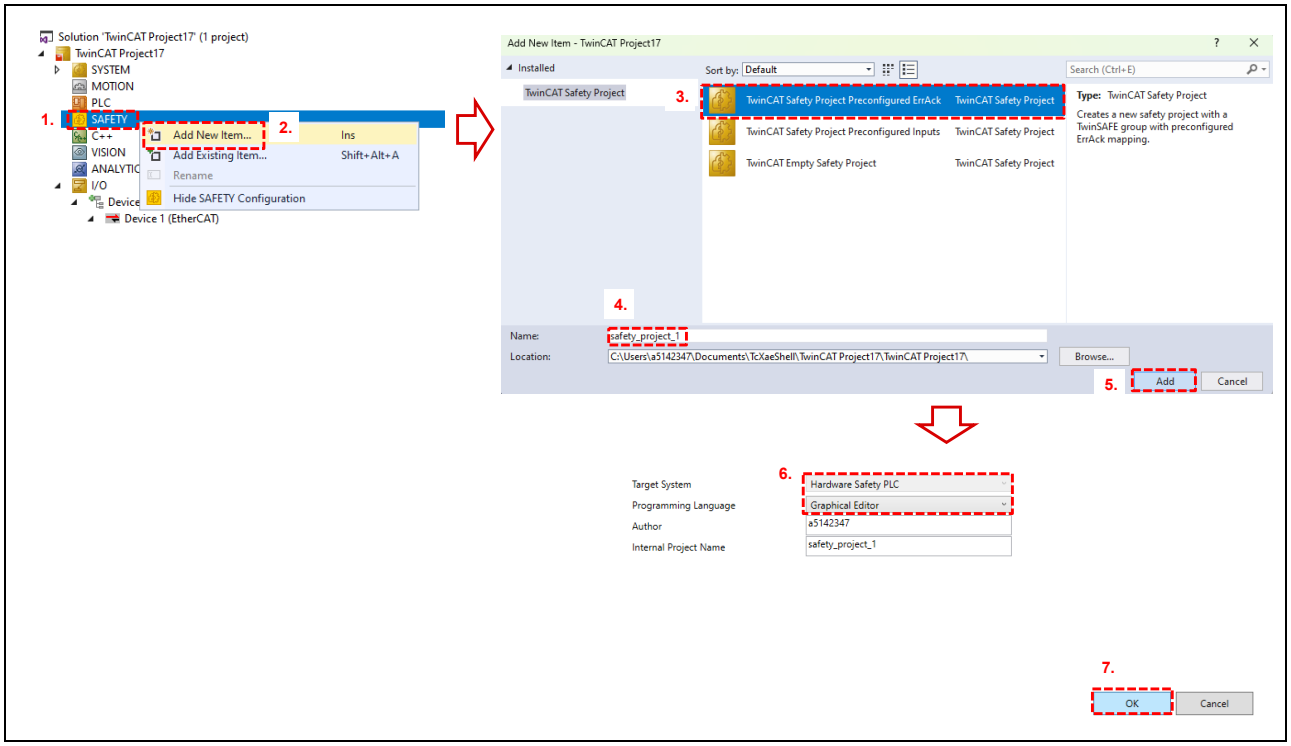


Figure 8.1. TwinSAFE Project Creation Procedure

(2) Importing the Device

1. In the System Manager tree, navigate to [SAFETY] → [safety_project_1] → [safety_project_1 Project] → [TwinSafeGroup1], then right-click [Alias Devices].
2. Select [Import Alias-Device(s) from I/O-configuration]. The Select from I/O tree window will appear.
3. Check the box for [Renesas EtherCAT RZ/T2 SafetyMotorSolution].
4. Click [OK].
5. If the import is successful, the message [Importing of 1 Alias Devices finished] will appear. Click [OK].

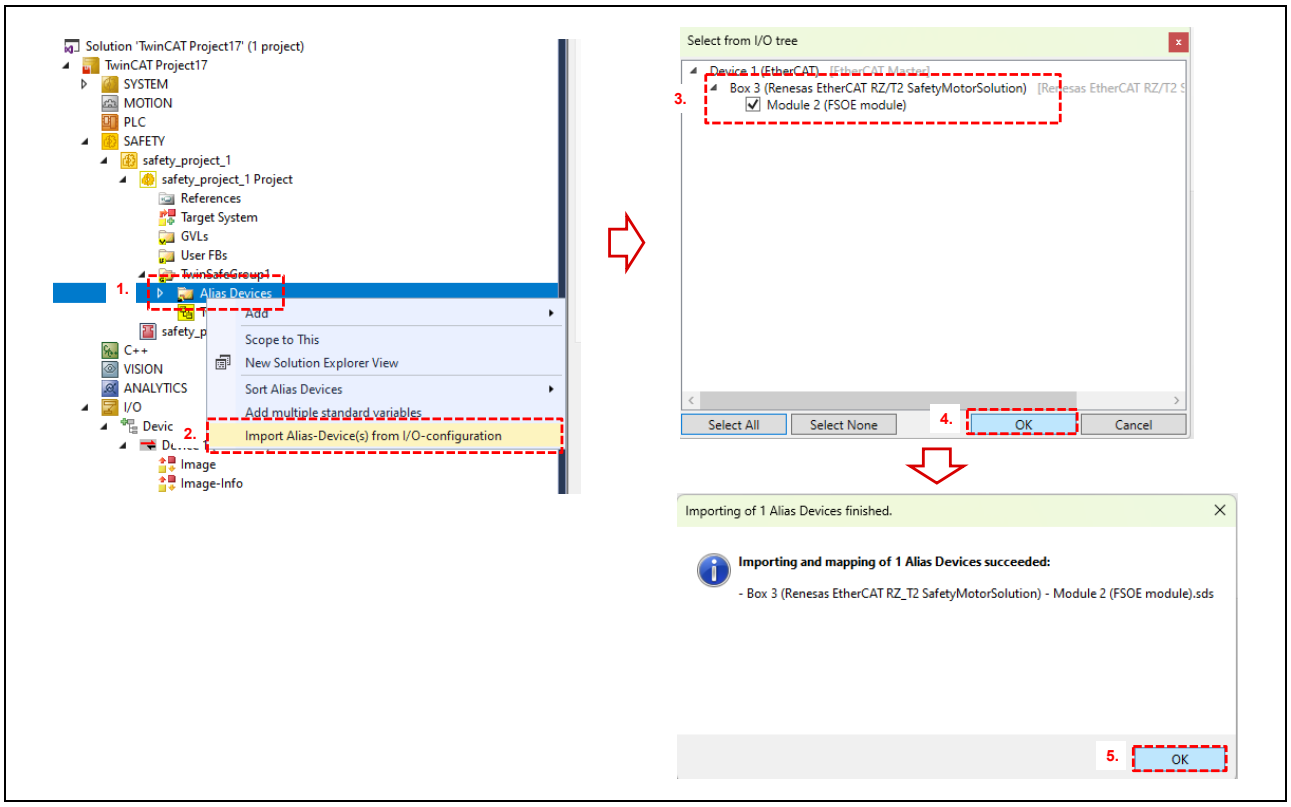


Figure 8.2. Importing the Device

(3) Configuring the TwinSAFE Graphical Editor

1. In the System Manager tree, navigate to [SAFETY] → [safety_project_1] → [safety_project_1 Project] → [TwinSafeGroup1].
2. Right-click [TwinSafeGroup1.sal].
3. Select [Open] to launch the TwinSAFE Graphical Editor.
4. Click the [Properties Window] button to open the Properties window.
5. In the Properties window, set Map Diag and Map State under Info Data to True.

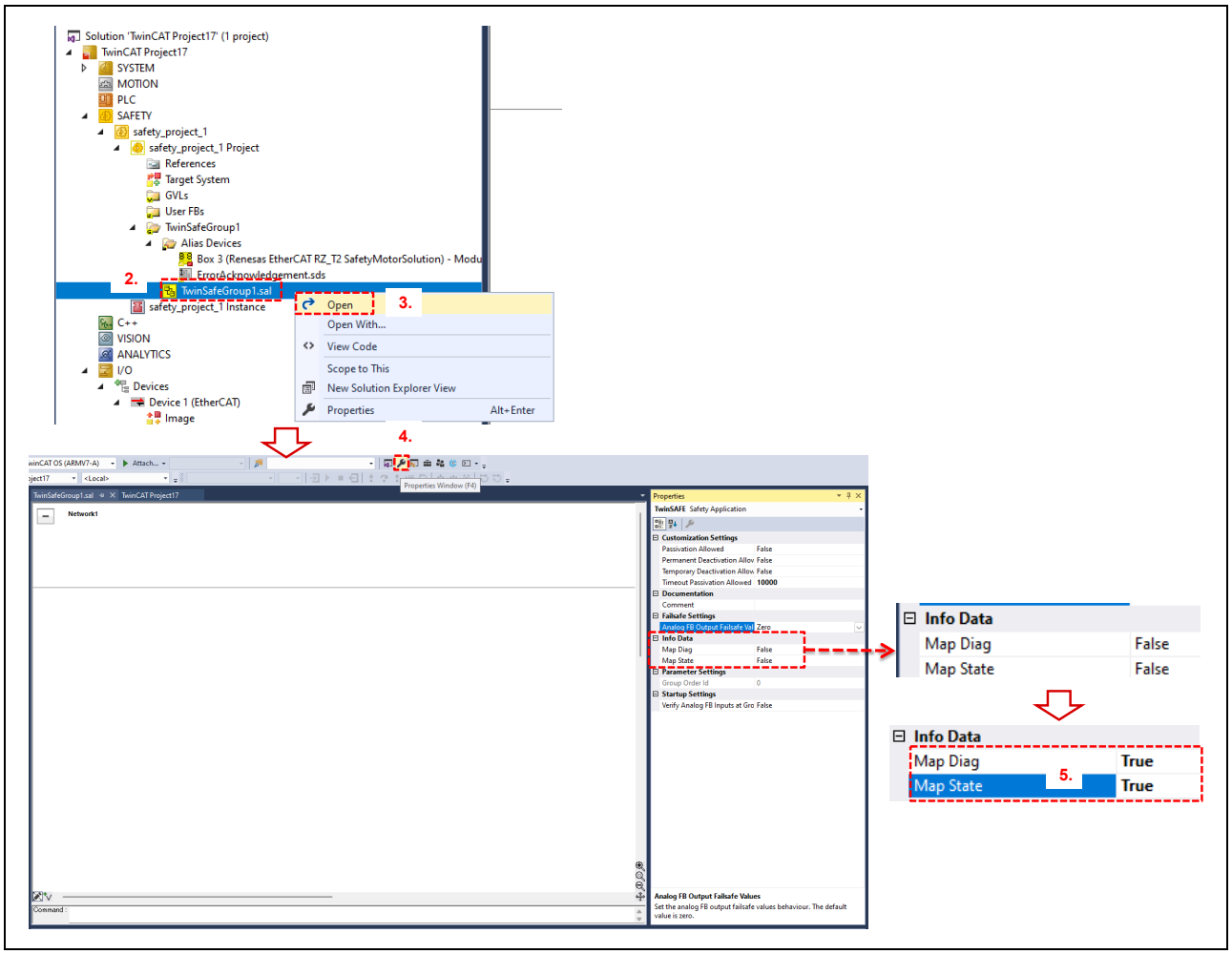


Figure 8.3. Configuring the TwinSAFE Graphical Editor

(4) Creating Function Blocks

1. Select [VIEW].
2. Select [Toolbox] to display the Toolbox window.

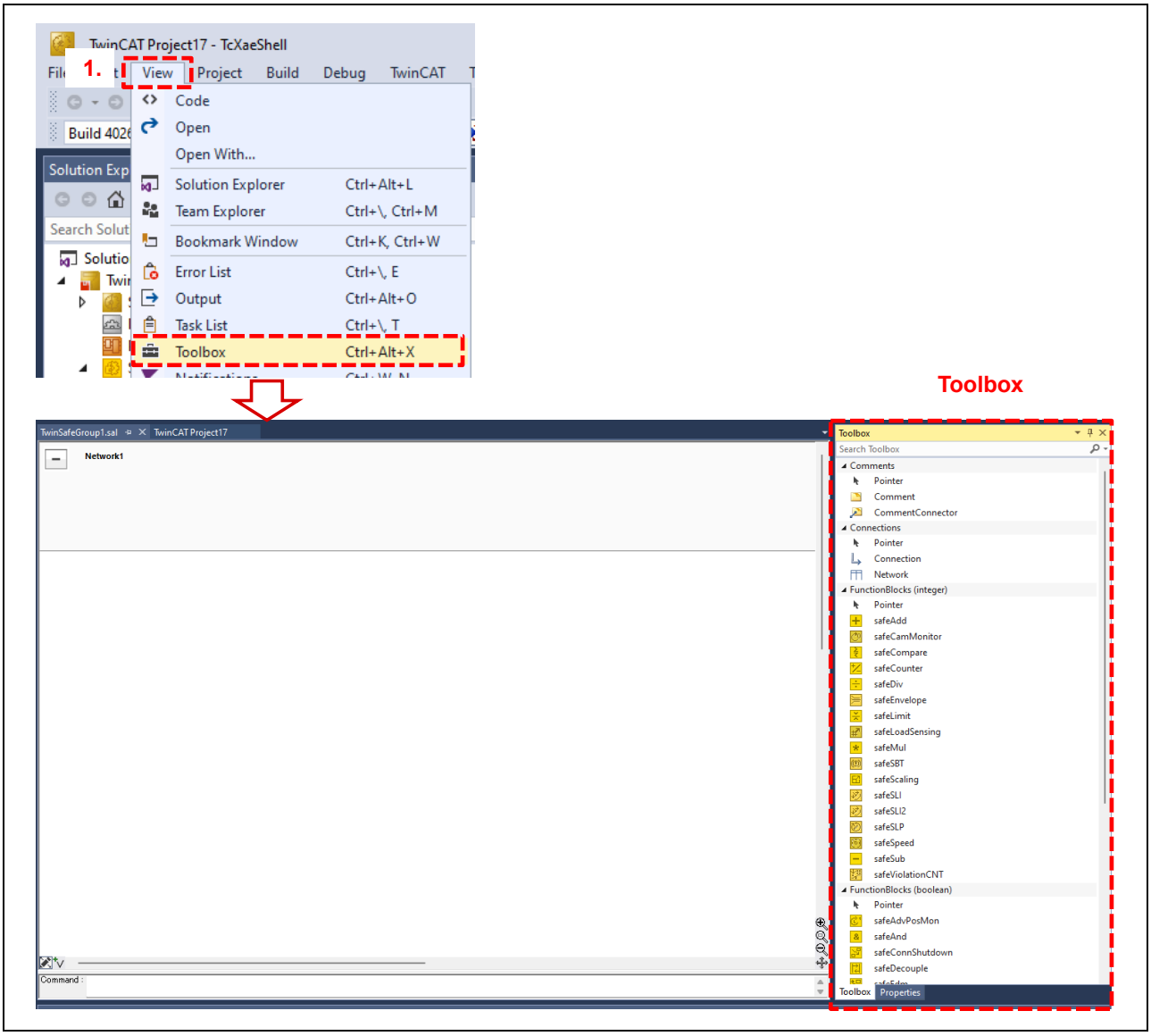


Figure 8.4. Creating Function Blocks 1

3. From [FunctionBlocks(boolean)], select [safeDecouple].
4. Drag and drop [safeDecouple] into the TwinSafeGroup1.sal window to create the function block.

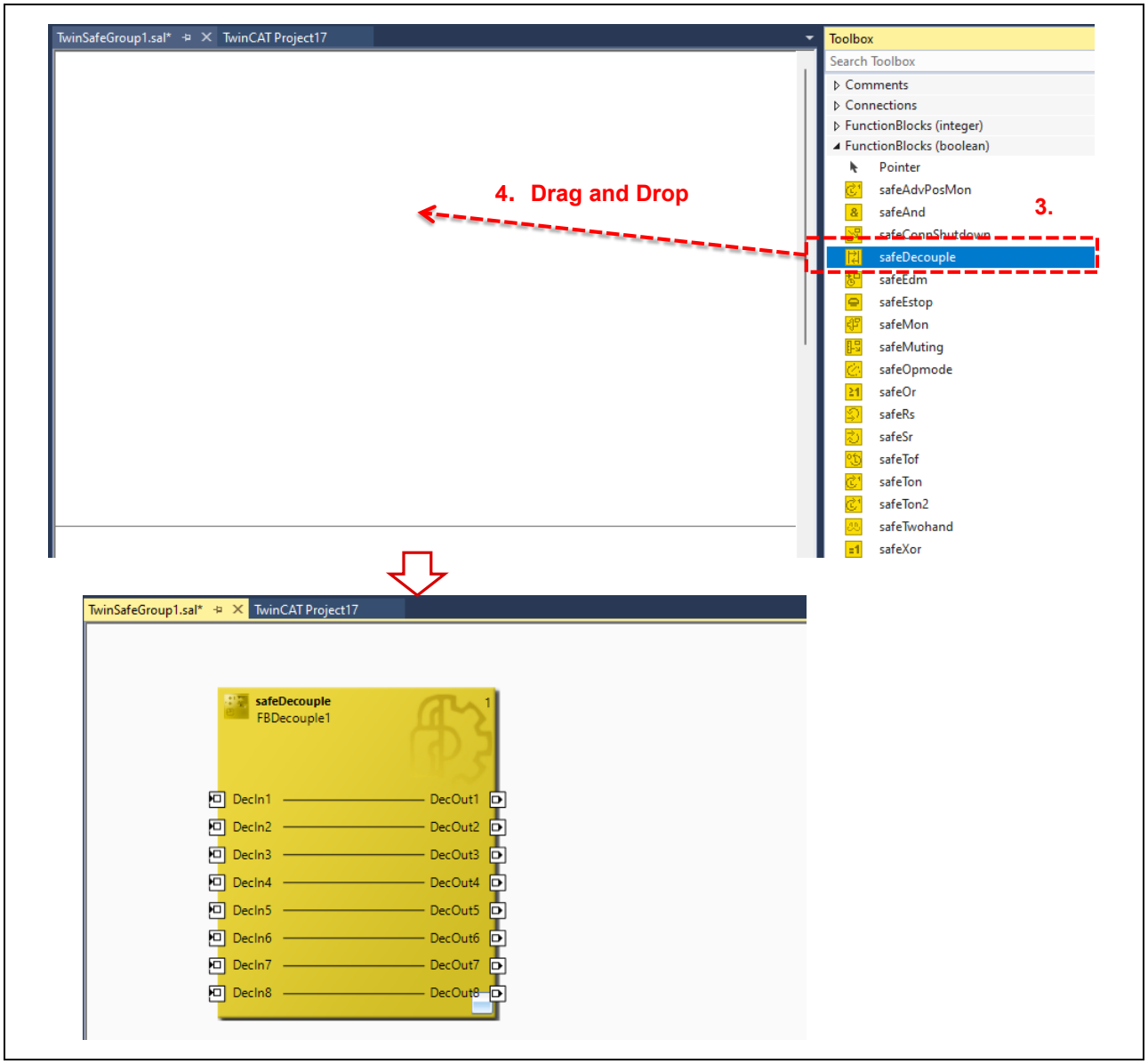


Figure 8.5. Creating Function Blocks 2

(5) Assigning Variable Names to Function Block I/O

1. Right-click [DecIn1] on the function block [FBDecouple1].
2. Select [Properties].
3. Under [Parameter Settings], set [Assigned Variable Name] to INPUT1.
4. Similarly, set [DecIn2] to INPUT2.

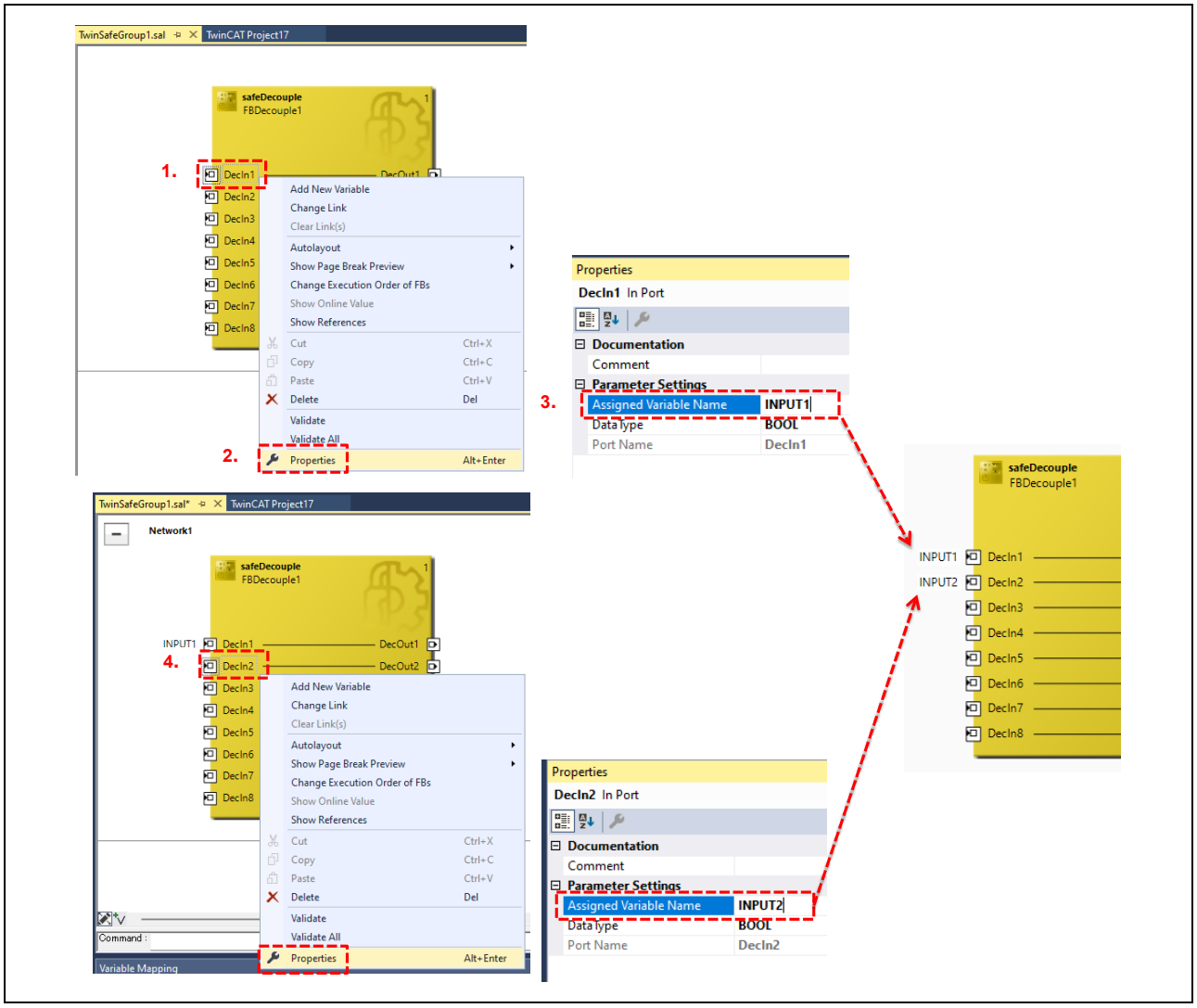


Figure 8.6. Assigning Variable Names 1

5. Repeat steps 1–4 for all terminals.

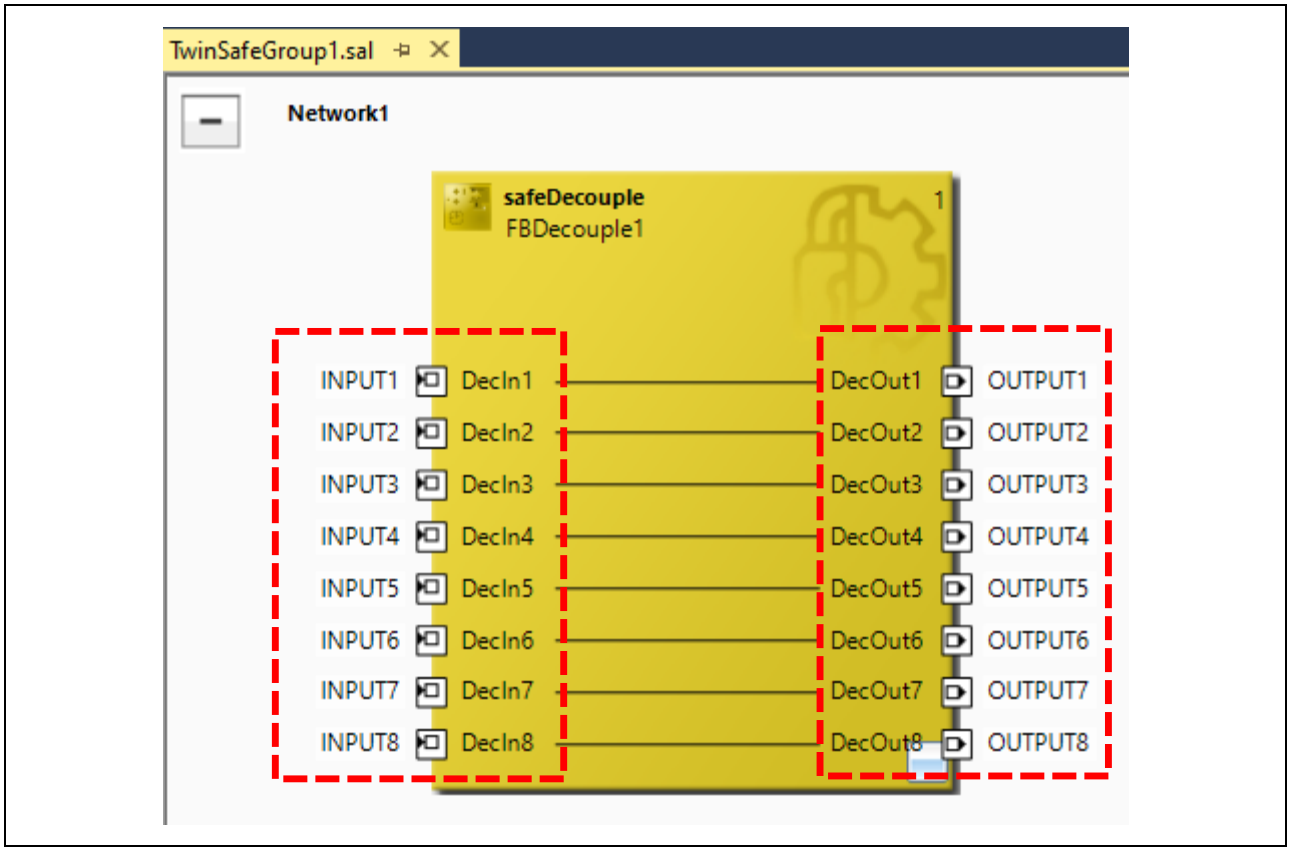


Figure 8.7. Assigning Variable Names 2

(6) Mapping Variables

1. In the TwinSAFE Graphical Editor, select the [Variable Mapping] tab at the bottom.
2. Select the [Variables] tab.
3. For INPUT1, click […] to open the Map to window.
4. Select [SubIndex 001].
5. Click [OK].
6. Repeat for other variables.

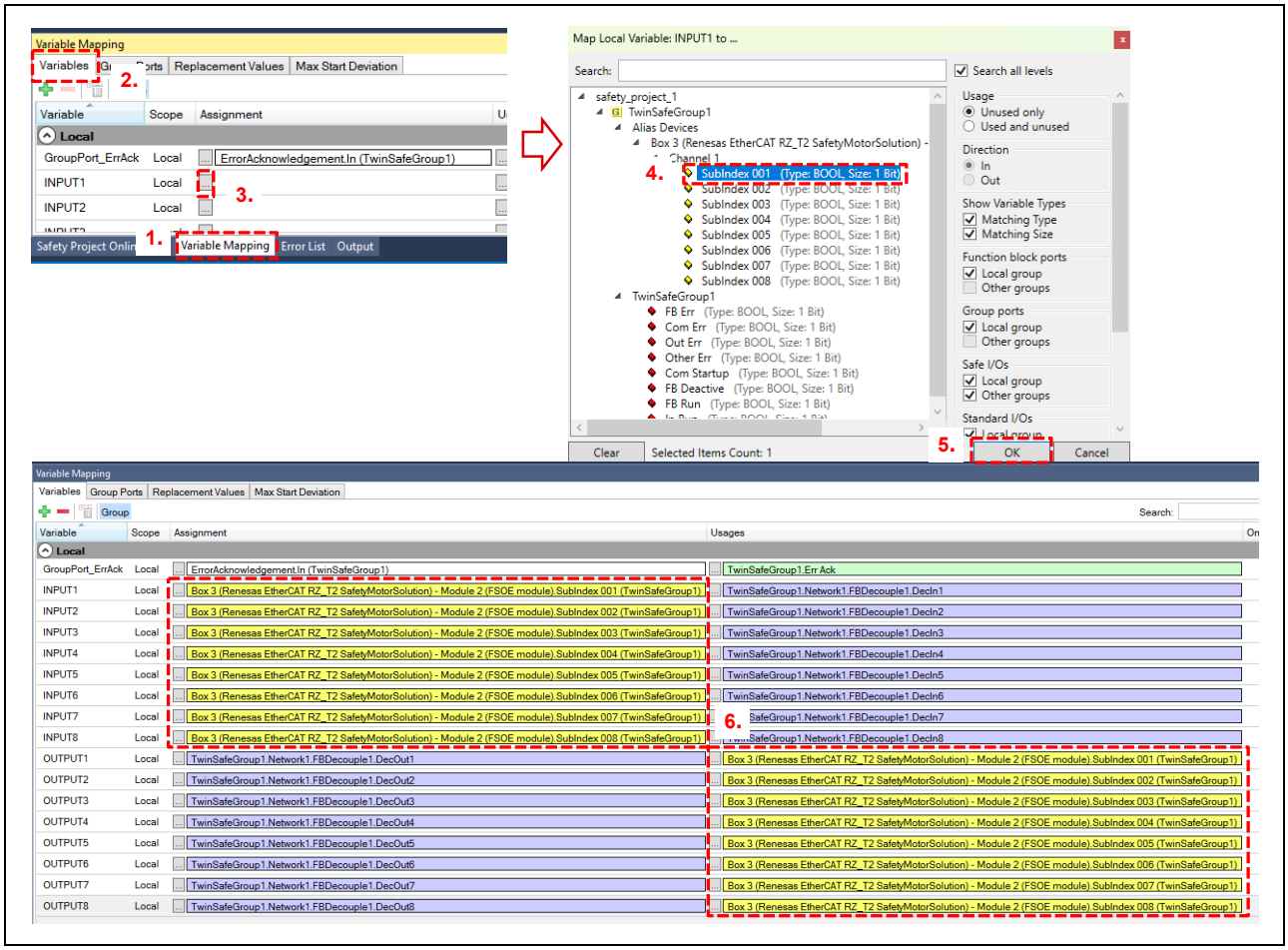


Figure 8.8. Mapping Variables

(7) Creating the PLC Project

1. In the System Manager tree, right-click [PLC].
2. Select [Add New Item…] to create a new project.
3. Choose [Standard PLC Project].
4. Enter a project name. In this document, the name safety_PLC_project_1 will be used.
5. Click [Add].

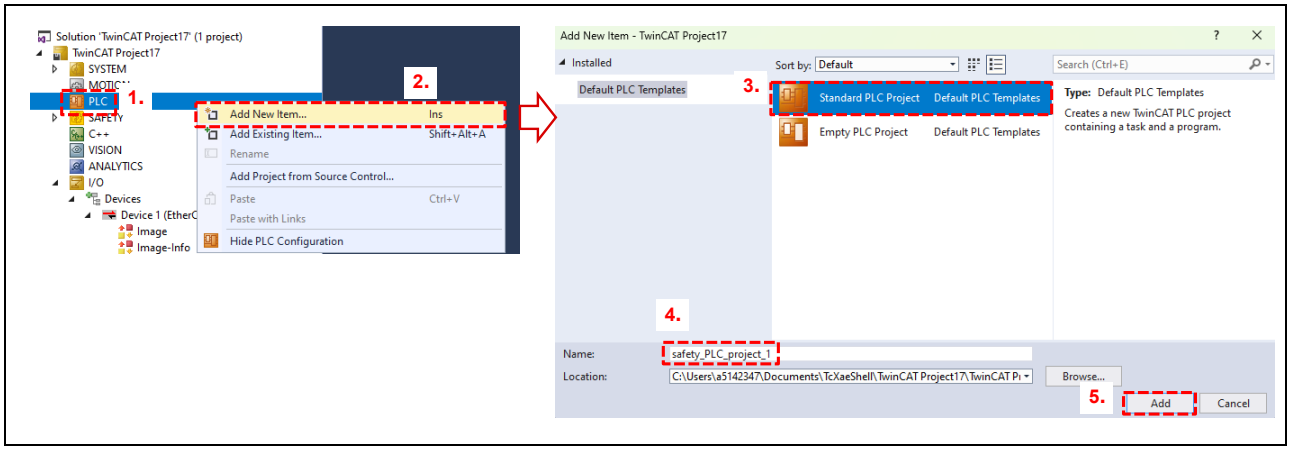


Figure 8.9. Creating the PLC Project

(8) Writing the PLC Program

1. In the System Manager tree, navigate to [PLC] → [safety_PLC_project_1] → [safety_PLC_project_1 project] → [POUs], then right-click [MAIN(PRG)].
2. Select [Open] to open the PLC project.
3. Add the program shown in the figure.

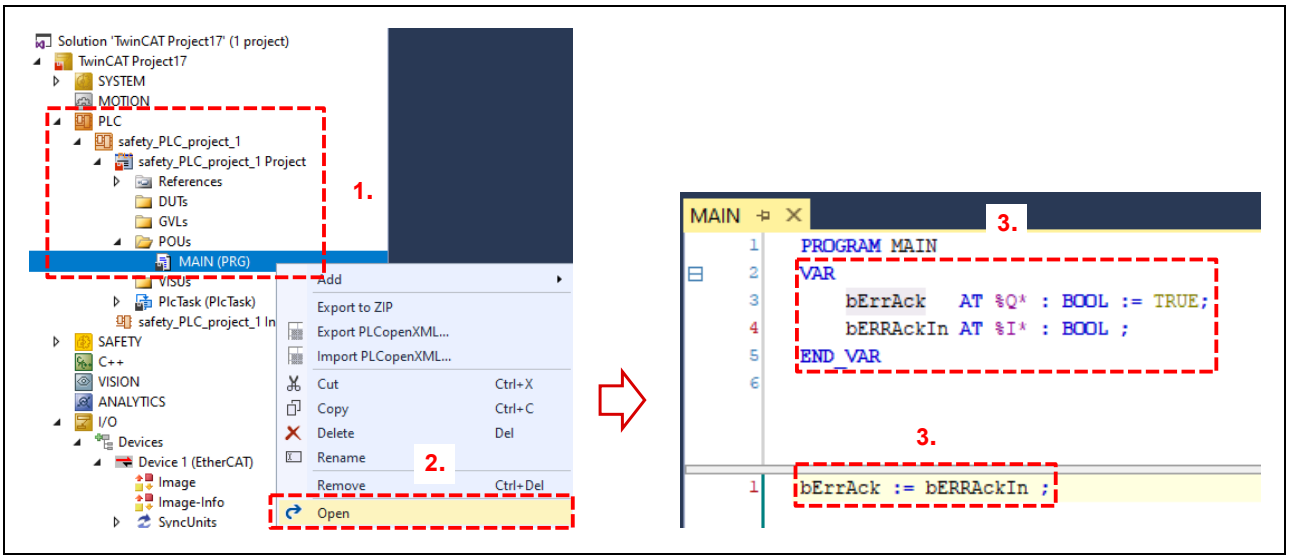


Figure 8.10. Writing the PLC Program

(9) Building the PLC Program

1. In the System Manager tree, right-click [safety_PLC_project_1 project].
2. Select [Rebuild] to build the project.
3. If successful, [Rebuild All succeeded] will appear, and the two variables declared with VAR will be generated in the Instance.

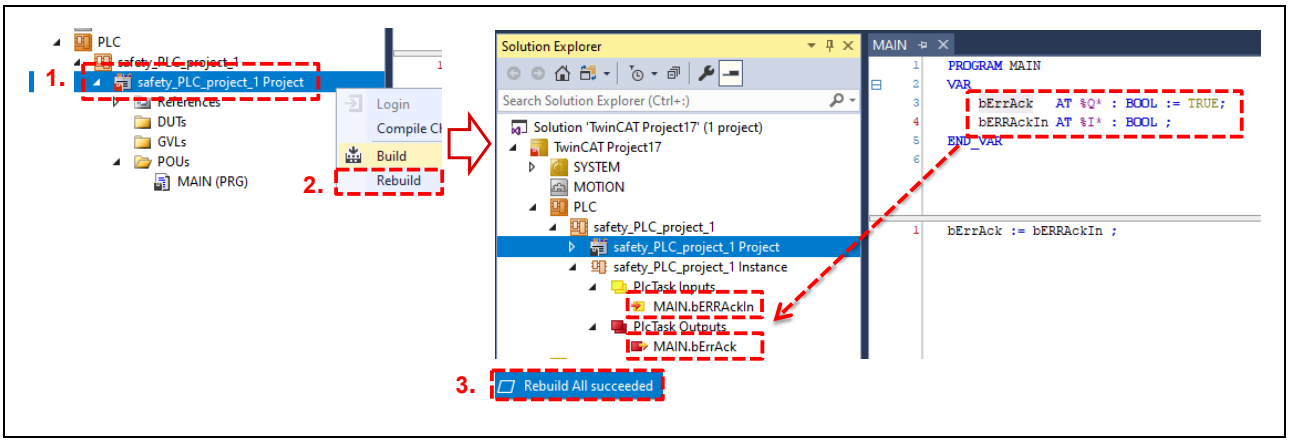


Figure 8.11. Building the PLC Program

(10) Registering the FSoE Master Device

1. In the System Manager tree, navigate to [SAFETY] → [safety_project_1], then right-click [safety_project_1 Project].
2. Select [Properties].
3. In the [Target System] tab, click the icon for [Physical Device] to open the Choose physical terminal for mapping window.
4. Select the connected FSoE master device (e.g., EL6900).
5. Click [OK].
6. After configuration, device information will update. The Serial Number will be required later.
7. Confirm that [Safe Address] and [Hardware Address] match. If they differ, click the icons in order (a) → (b) to synchronize.

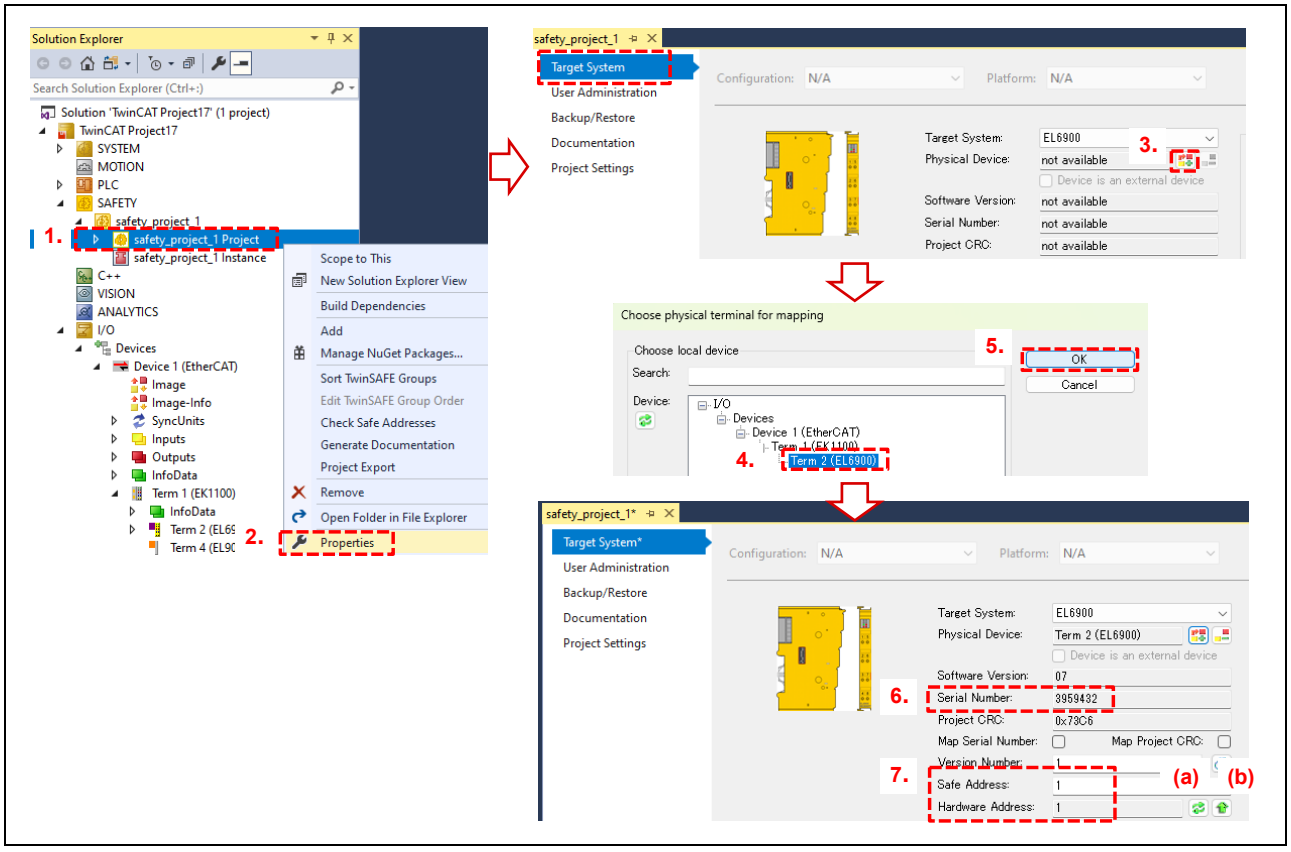


Figure 8.12. Registering the FSoE Master Device

(11) Checking the FSoE Slave Device (Safety Motor Control Reference Kit)

1. In the System Manager tree, navigate to [SAFETY] → [safety_project_1] → [safety_project_1 Project] → [TwinSafeGroup1] → [Alias Devices], then double-click [Box X (Renesas EtherCAT RZ_T2 SafetyMotorSolution) - Module X (FSOE module)].sds.
2. Confirm that [FSoE Address] and [Dip Switch] match the slave device's FSoE address. If not, click the update icon.
3. Confirm that [Full Name] and [Linked to] display device information. If they show "not available," click the update icon.

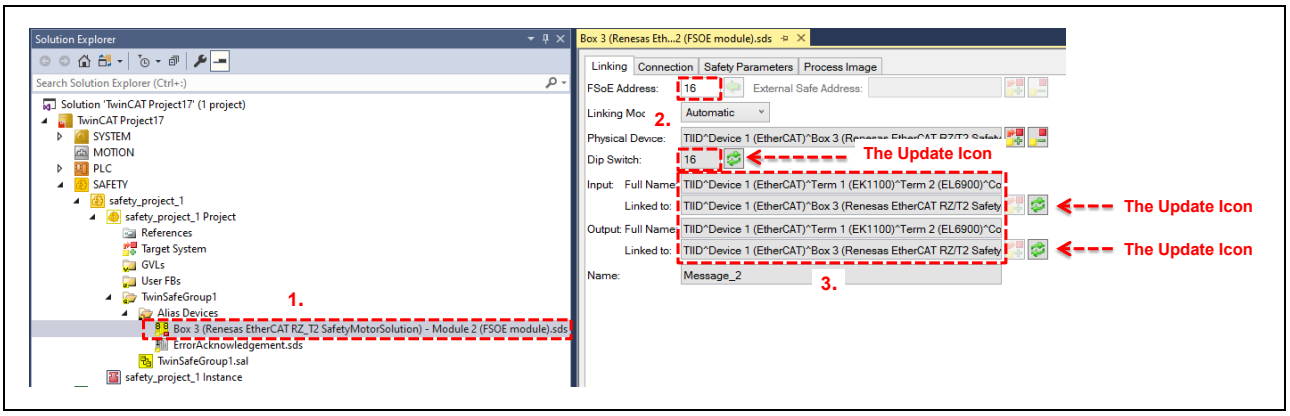


Figure 8.13. Checking the FSoE Slave Device

Note 1: If [Dip Switch] shows "n.a" and does not update, communication with the slave device may have been lost. Reconfigure starting from Section 6.1.

Note 2: If [Full Name] and [Linked to] remain "not available," reconfigure starting from Section 6.1.

(12) Selecting the ErrAck Signal

1. In the System Manager tree, navigate to [SAFETY] → [safety_project_1] → [safety_project_1 Project] → [TwinSafeGroup1] → [Alias Devices], then double-click [ErrorAcknowledgement.sds].
2. Click the icon next to [Full Name].
3. Select MAIN.bErrAck and click [OK].

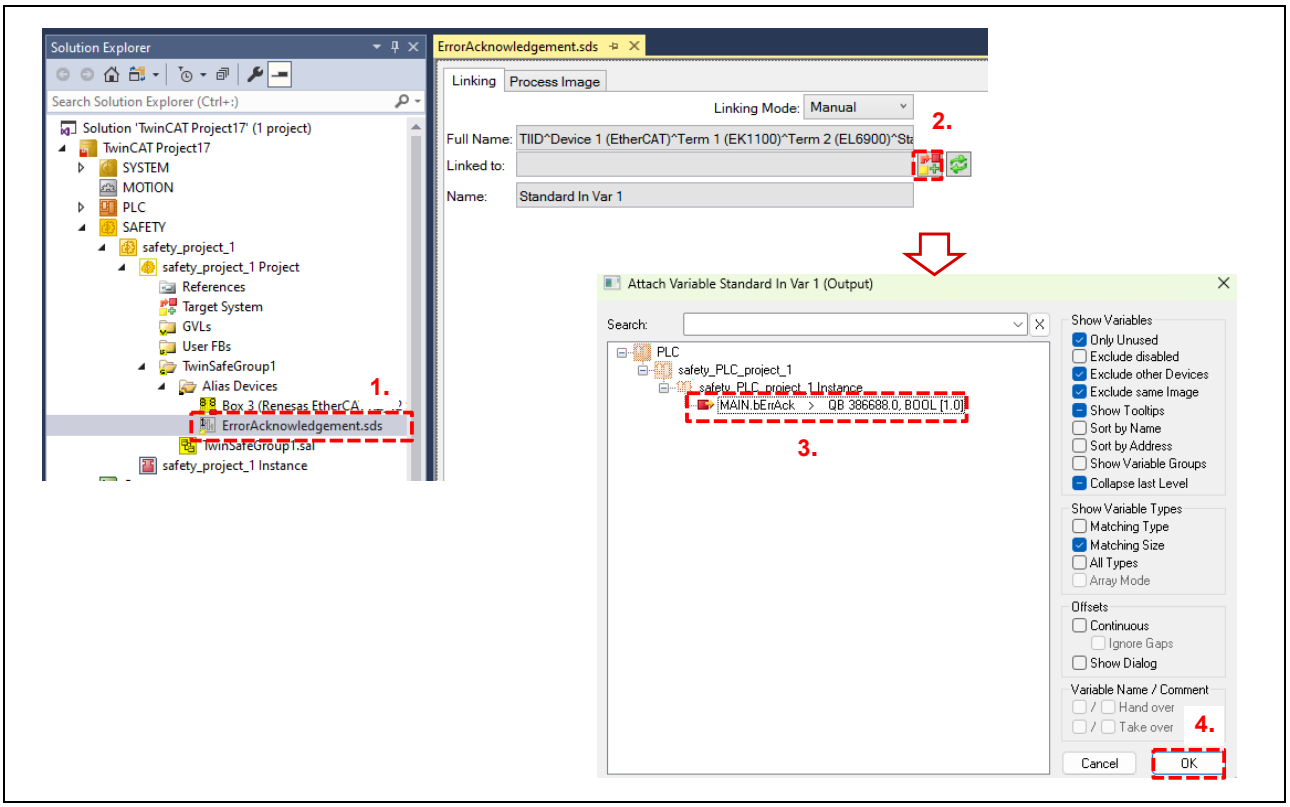


Figure 8.14. Selecting the ErrAck Signal

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Dec.19, 2025	-	First Edition Issued

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.

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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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.5.0-1 October 2020)

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/.