RX72M Group

EtherCAT ETG.5003 Sample Program Firmware Information Technology

Introduction

This application note describes a sample program of a slave device applied to semiconductor manufacturing equipment in EtherCAT® communication, which is one of the industrial Ethernet communication protocols.

The profile applied to semiconductor manufacturing equipment in EtherCAT is specified in etg.5003 Semiconductor Device Profile.

Sample program conforms to the following specifications among the ETG.5003 Semiconductor Device Profiles.

- Common Device Profile (CDP) [ETG.5003.1]
- Firmware update functionality [ETG.5003.2]

The Specific Device Profile (SDP) [ETG.5003.2xxx] is not supported in this application note.

Target Device

- RX72M Group

When using this product with your product, please thoroughly evaluate it according to your environment. In addition, when applying this application note to other microcomputers, modify it according to the specifications of the other microcomputers and fully evaluate it.

Content

1. Overview 4
1.1 About this application note 4
1.2 About bank numbers 4
1.3 Linear mode and dual mode comparison 5
1.4 Operating Environment 5
1.5 FIT Module Structure 6
1.6 Project 6

2. **Obtaining a Development Environment** 7
2.1 How to Obtain e² studio 7
2.2 How to Obtain the Compiler Package 7

3. Building the Sample Project 8
3.1 Import EtherCAT slave stack code into the sample program 8
3.2 Importing the Sample Project into the e² studio 10
3.3 How to install EtherCAT FIT module to e² studio 11

4. **Configuration of project** 12
4.1 Configuration of FIT module 12
4.1.1 Configuration of BSP FIT module 12
4.1.2 Configuration of Flash FIT module 13
4.1.3 SCI FIT module 14
4.1.4 Configuration of EtherCAT FIT module 14
4.2 Pin settings 15
4.3 build configuration 16
4.3.1 Linear mode build configuration 16
4.3.2 Build configuration of Dual mode 22
4.4 Debug configuration 28
4.4.1 Debug configuration of Linear mode 28
4.4.2 Debug configuration of Dual mode 29

5. Building and debugging projects 31
5.1 Generate Code 31
5.2 Build on Linear mode 31
5.3 Build on Dual mode 31
5.4 Preparation for Debug 32
5.5 Debug on Linear mode 33
5.6 Debug on Dual mode 35

6. Connection with TwinCAT 36
6.1 Preparation of ESI file 36
6.2 Starting TwinCAT 36
6.3 Add Ether driver 36
6.4 Scanning the network 37

6.5 Writing of SII EEPROM 38
6.6 Rescan of the device 39
6.7 I/O operation confirmation 40

7. Operation check by TwinCAT 42
7.1 Firmware writing 42
7.2 Firmware readout 47

8. Sample program overview 50
8.1 Overview of linear mode operation 50
8.2 Overview of dual mode operation 53
8.3 Bank Info 55
8.4 Download file 56
8.5 SII EEPROM 59
8.6 Firmware update program details 60
8.6.1 File structure 60
8.6.2 List of constants 60
8.6.3 Type definition list 62
8.6.4 Variable list 64
8.6.5 Function list 65
8.7 FoE Program Details 66
8.7.1 File structure 66
8.7.2 List of constants 66
8.7.3 Type definition list 67
8.7.4 Variable list 67
8.7.5 Function list 68
8.8 Common Device Profile [ETG.5003.1] 69
8.9 Object Dictionary 70
8.10 Semi Test Record [ETG.7000.2-Annex5003-0001] 76
8.10.1 Device Reset Command (Standard reset) 76
8.10.2 Dynamic PDO 77
8.10.3 Store Parameters 81

9. Appendices To support code flash memory capacity of 2MB 84

10. Reference documents 86

1. Overview

1.1 About this application note

This application note mainly describes the firmware update function (ETG.5003.2) using the File Access over EtherCAT (FoE) protocol.

Refer to 8.8 for the support contents of Common Device Profile (ETG.5003.1).

Sample program works in conjunction with FIT modules such as EtherCAT, Flash, boardsupport packages (abbreviated asBSPs).

Sample program does not include SSCs. Obtain the SSC tool from EtherCAT Technology Group (ETG Association) and generate the SSC.

1.2 About bank numbers

Sample program is stored in code flash memory and can be updated while the program is running.

Sample program supports two banks, one for executing the user program and one for writing the update program.

In this application note, the bank or BANK number is fixed because it is shared between the linear mode and dual mode.

In dual mode, the start-up bank switching bit(BANKSEL. Note that the numbers are not swapped by the value of BANKSWP[2:0]).

From the one close to the reset vector(FFFF FFFCH-FFFF FFFFH), it is (0 to 1)

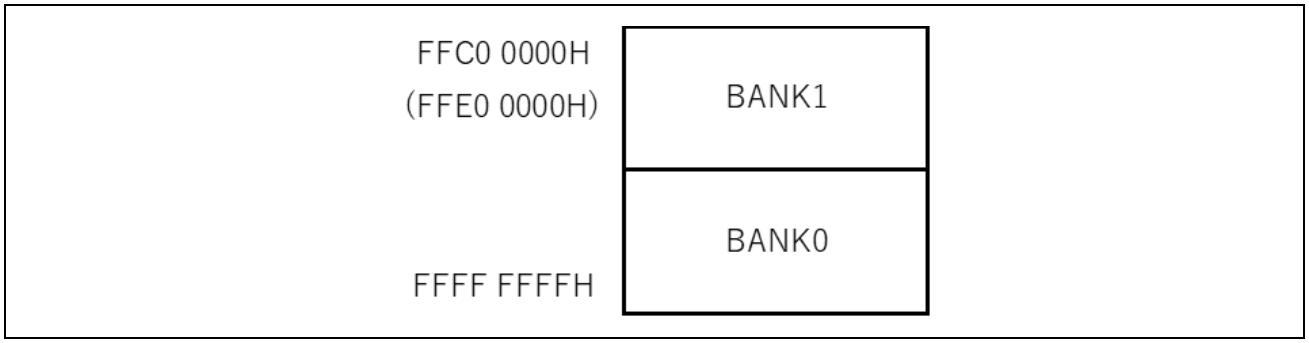


Figure 1.1 **Bank number**

1.3 Linear mode and dual mode comparison

This application note supports both linear mode and dual mode, but there are differences and limitations of functions depending on the mode.

The following is a summary, use it as a reference when choosing a mode.

Table 1.1 Comparing the features of the sample program

Item	Linear mode	Dual mode
Switching user program at startup	Switching which depend on Bootloader program	Switching which depend on dual bank function of hardware
Code flash block configuration	The configurations are different for two banks of 8Kbyte and 32Kbyte	The configuration of two banks is same
Trusted Memory Target area	Blocks 8, 9	Blocks 8, 9 and blocks 78, 79
User program size can be used in one blank	Excluding capacity for 32KB that allocated to boot loader from 1/2 of the installed code flash memory capacity	Entire of 1/2 of the installed code flash memory capacity
Reboot when EtherCAT link is broken※	without broken link	With broken link (for using software reset)
Operation of update firmware	For two banks case, user need to determine which one is available for writing. Then download the appropriate file	User can download the download file without being aware of the banks

※When reboot is occurred, it is possible to reconnect even though EtherCAT link is broken

.

1.4 Operating Environment

Sample program run on RX72M evaluation board made by TESSERA Technology (hereafter, inscribe with communication board).

Table 1.2 Confirmation of operating environment

MCU Support	RX72M Group R5F572MNxDBD The capacity of Code flash memory: 4Mbyte※
Evaluation board	RX72M evaluation board from TESSERA technology is TS-RX72M-COM RX72M CPU Card （RTK0EMXDE0C00000BJ） Renesas Starter Kit+ for RX72M（RTK5572MNxCxxxxxBJ）
Integrated development environment (IDE)	e^2 studio 2024-01 from Renesas electronic Corporation
Cross Tool	C/C++ Compiler Package for RX Family V3.06.00 from Renesas electronic Corporation
Emulator	E2 Lite
SSC Tool	Slave Stack Code (SSC) Tool Version 5.13 form EtherCAT Technology Group (ETG)
Software PLC	TwinCAT® 3 from Beckhoff Automation

※To support code flash memory capacity product please refer to [9 Appendix Supporting for code flash memory capacity of 2MB]

1.5 FIT Module Structure

This application note consists FIT module as shown below.

Table 1.3 FIT Module Structure

Type	Module name	FIT module name	Rev.
Board Support Package	Board Support Package (BSP)	r_bsp	7.42
Device Driver	Compare Match Timer (CMT)	r_cmt_rx	5.60
Device Driver	Serial Communication Interface (SCI)	r_sci_rx	4.90
Middleware	Byte Type Queue buffer (BYTEQ)	r_byteq	2.10
Device Driver	EtherCAT	r_ecat_rx	1.31
Device Driver	Flash	r_flash_rx	5.10

1.6 Project

The projects included in this application note are shown below.

In the following chapters, describe the serial mode project of RX72M communication board. For the different project please read after replacing the project name with suitability.

Table 1.4 Project List

MCU	Evaluation board name	Dual bank function	Project name
RX72M	Communication board	Linear mode	ecat_foe_linear_demo_comrx72m
		Dual mode	ecat_foe_dual_demo_comrx72m
	CPU card	Linear mode	ecat_foe_linear_demo_cpurx72m
		Dual mode	ecat_foe_dual_demo_cpurx72m
	RSK board	Linear mode	ecat_foe_linear_demo_rskrx72m
		Dual mode	ecat_foe_dual_demo_rskrx72m

2. Obtaining a Development Environment

2.1 How to Obtain e² studio

Access the following URL and download the e² studio.

https://www.renesas.com/us/en/products/software-tools/tools/ide/e2studio.html#downloads

This application note assumes that you will be using 2024-01 or a later version of the e² studio. If you are using a version earlier than 2024-01, some functions of the e² studio may not be available.

When downloading the e² studio, obtain the latest version on the website.

2.2 How to Obtain the Compiler Package

Access the following URL and download the RX family C/C++ Compiler Package.

https://www.renesas.com/us/en/products/software-tools/tools/ide/e2studio.html#downloads

3. Building the Sample Project

3.1 Import EtherCAT slave stack code into the sample program

This sample project does not include the EtherCAT Slave Stack Code.

However, the project requires the EtherCAT Slave Stack Code and to generate and use this you must obtain the EtherCAT Slave Stack Code (SSC) tool.

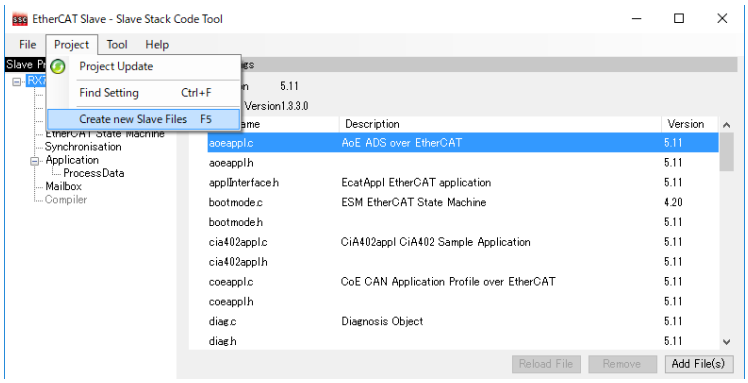
The EtherCAT Technology Group (ETG) provides the SSC package.

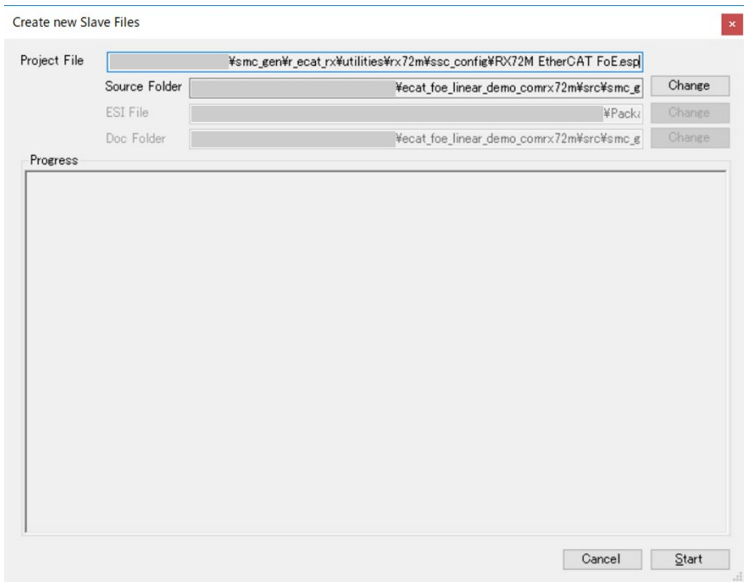
The sample program is provided in the from "ecat_foe_linear_demo_comrx72m.zip", extract it to any folder in advance.

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

ecat_foe_linear_demo_comrx72m¥utilities¥ssc_config¥RX72M EtherCAT FoE.esp

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





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

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

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

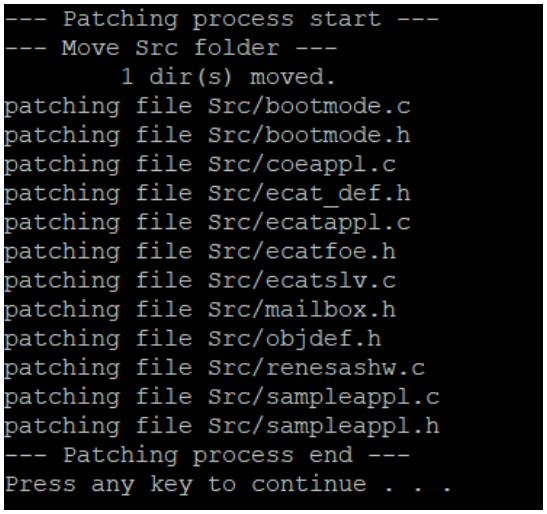
http://gnuwin32.sourceforge.net/packages/patch.htm

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

ecat_foe_linear_demo_comrx72m¥utilities¥rx72m¥patch¥apply_patch.bat

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

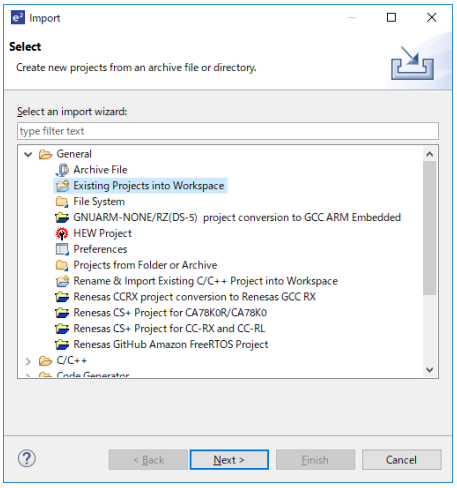
ecat_foe_linear_demo_comrx72m¥project¥src¥application¥ecat



3.2 Importing the Sample Project into the e² studio

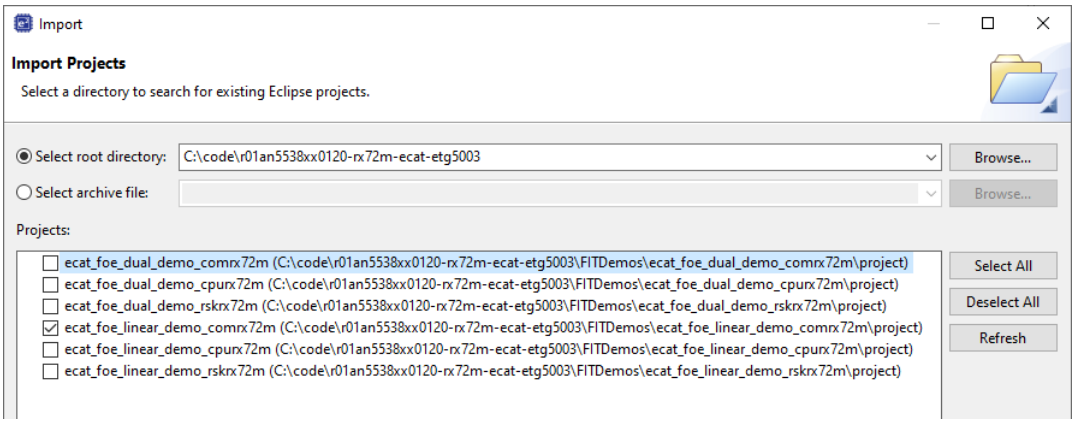
(1) Click on [File] → [Import].

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



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

(4) Select "r01an5538xx0120-rx72m-ecat-etg5003" and click [Open].



(5) Check [ecat_foe_linear_demo_comrx72m] in [Project] and click [Next] to import the project.

3.3 How to install EtherCAT FIT module to e² studio

To be able to use the EtherCAT FIT module in the Smart Configurator, you need to add it to e² studio.

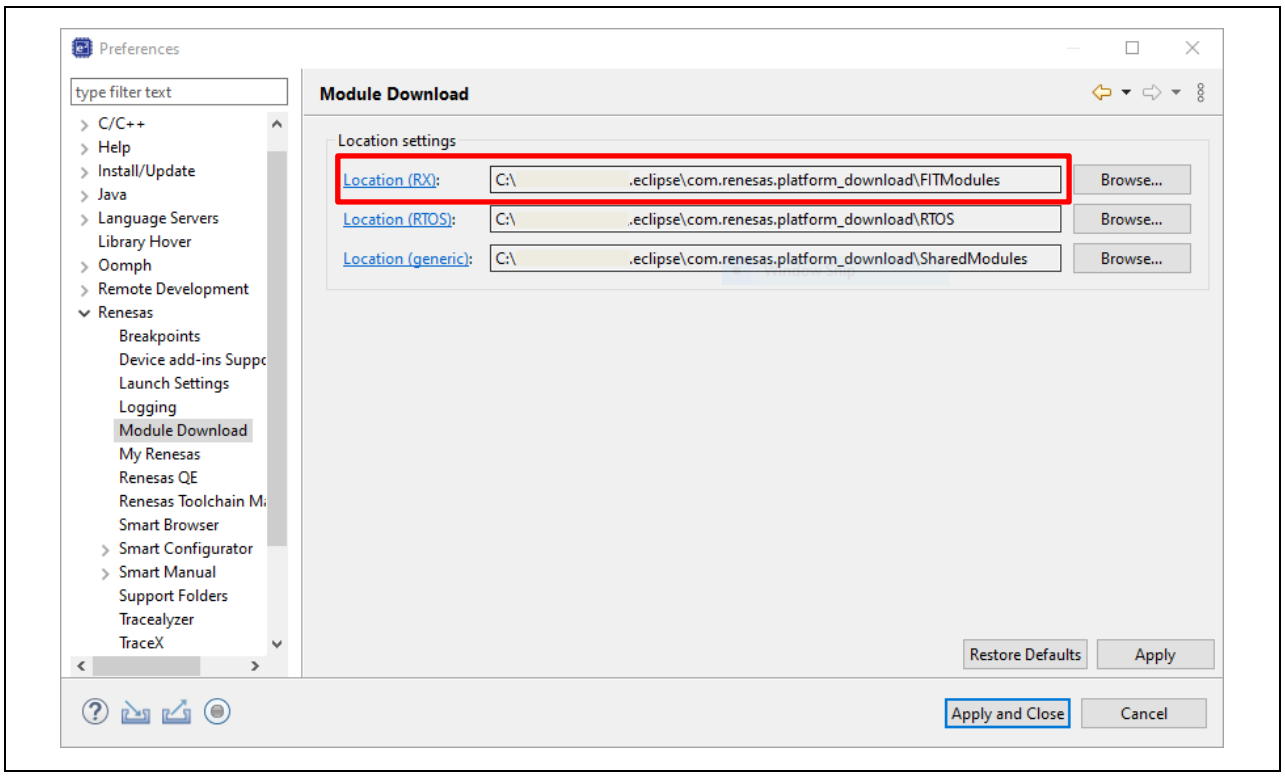
Shows how to add it manually.

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

In e² studio, check the location of the FIT module.

1. “Window” → “Preferences” → Open setting window.
2. Select "Renesas" → "Module Download".

The path shown as Location (RX): is the destination folder of the FIT module.



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

Copy the file in the r01an5538xx0120-rx72m-ecat-etg5003¥FITModules folder to the folder where the FIT module is saved.

r_ecat_rx_vN.NN.xml

r_ecat_rx_vN.NN.zip

r_ecat_rx_vN.NN_extend.mdf

Note that N.NN is a numerical value that represents the version.

4. Configuration of project

This chapter describes the various settings for this project.

If you want to check the operation in the project where the sample program is imported, it has already been set. Go to “5. Building and Debugging Projects”.

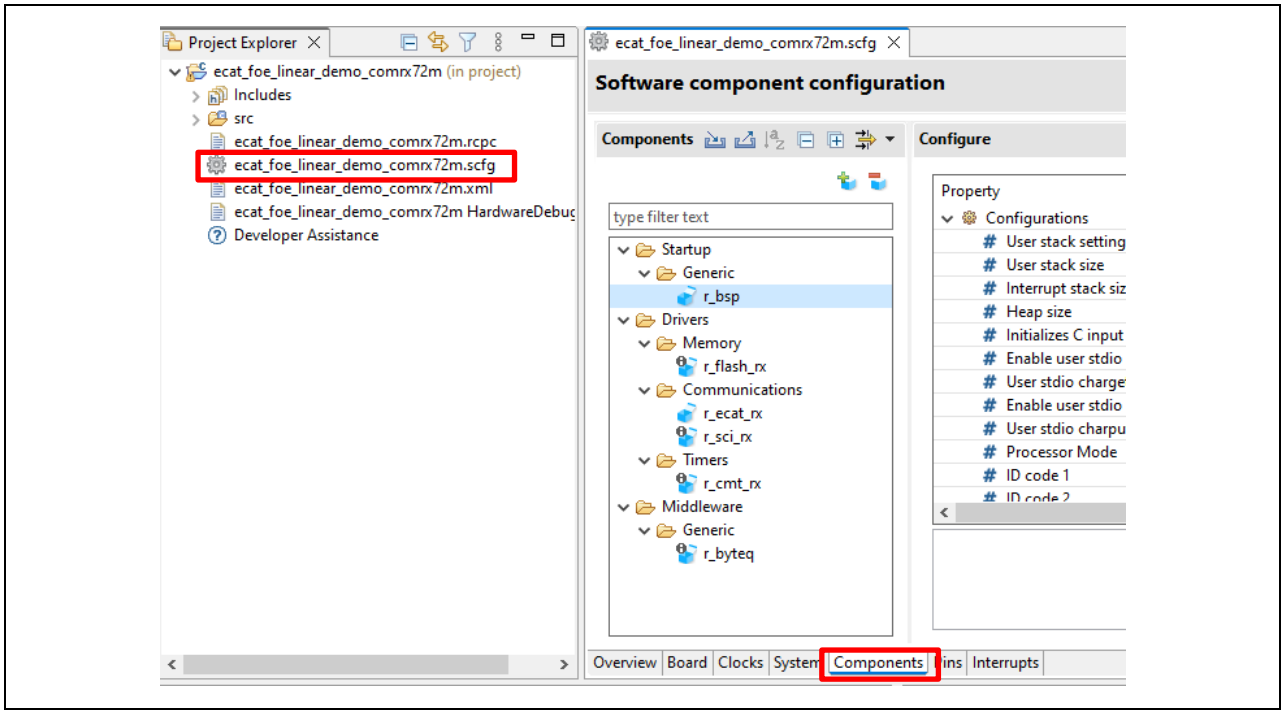
4.1 Configuration of FIT module

In this project, the configuration settings of the FIT module are changed from the default in order to configure this sample program.

You can open the Smart Configuration file "ecat_foe_linear_demo_comrx72m.scfg" and check and change the configuration of the FIT module on the Components tab.

For the items and settings in the configuration files, refer to the manuals, etc. in the doc folder for each FIT module.

The table below lists the changes in configurations.

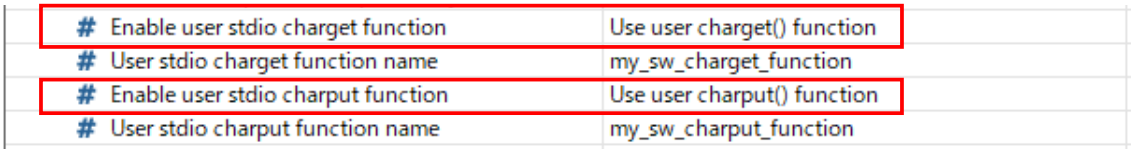


4.1.1 Configuration of BSP FIT module

4.1.1.1 Configuration of charget / charput function

Set by Using user charget() function.

Configurations	Value
Enable user stdio charget function	Use user charget() function
Enable user stdio charpur function	Use user charput() function



4.1.1.2 Configuration of bank mode

Set according to the mode in which you want to use the bank switching function.

Macro Name	Setting
BSP_CFG_CODE_FLASH_BANK	0 ; Dual mode 1 : Linear mode (default value))

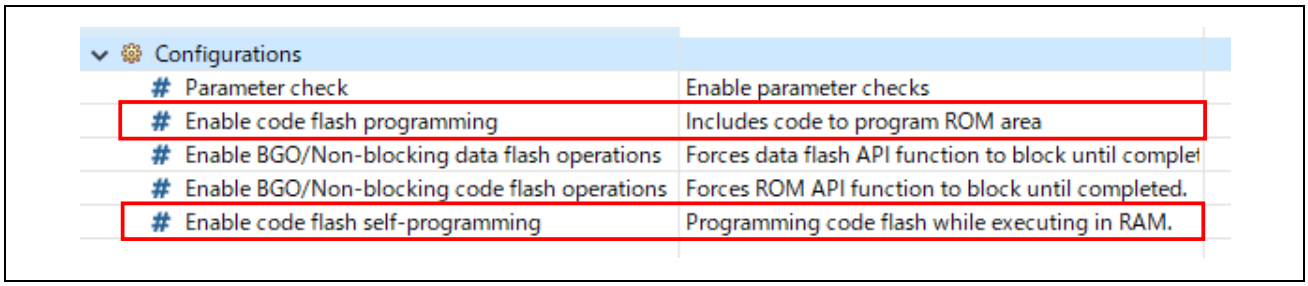
* This setting cannot be checked or changed on the Components tab. You can check and change it in the file "src¥smc_gen¥r_config¥r_bsp_config.h" that is generated after code generation.

4.1.2 Configuration of Flash FIT module

- linear mode

The code that rewrites the code flush is set to run in RAM。

Configurations	Value
Enable code flash programming	Include code to program ROM area
Enable code flash self-programming	Programming code flash while executing on RAM. (default)

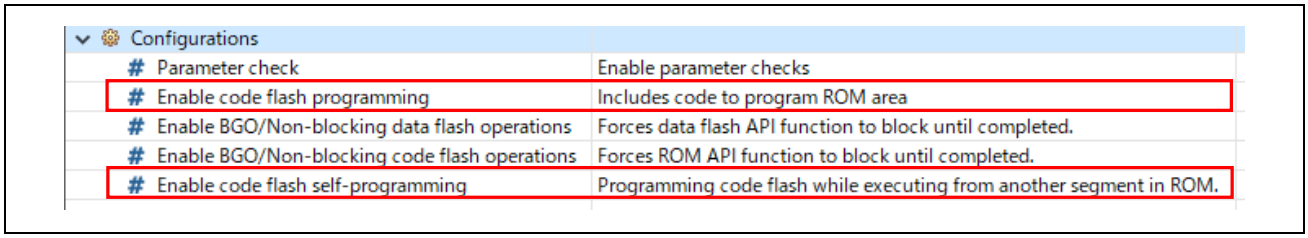


- Dual mode

Run the code from the code flash and set it to rewrite the code flash.

Configurations	Value
Enable code flash programming	Include code to program ROM area
Enable code flash self-programming	Programming code flash while executing from another segment in ROM.

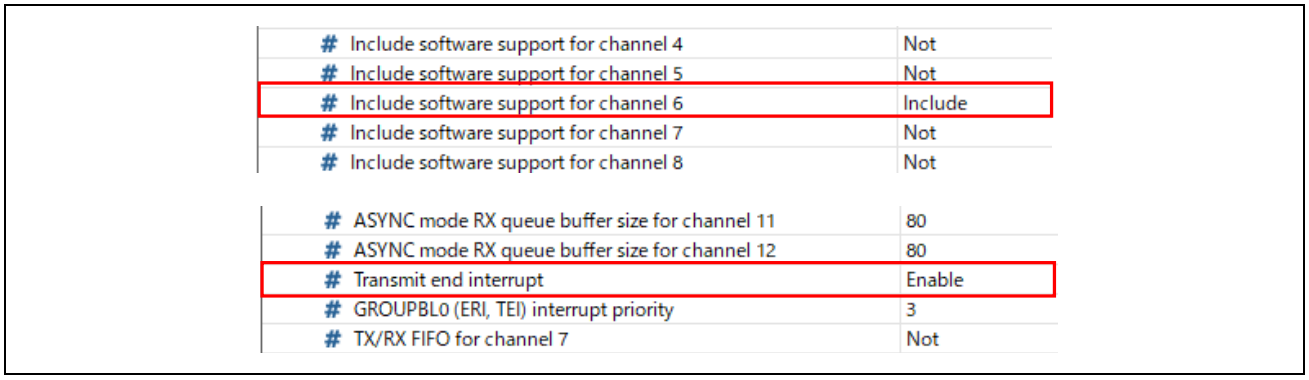
※ BGO mode is not used in this sample program.



4.1.3 Configuration of SCI FIT module

Set CH6 and send completion interrupts to enable。

Configurations	Value
Include software support for channel 6	Include
Transmit end interrupt	Enable

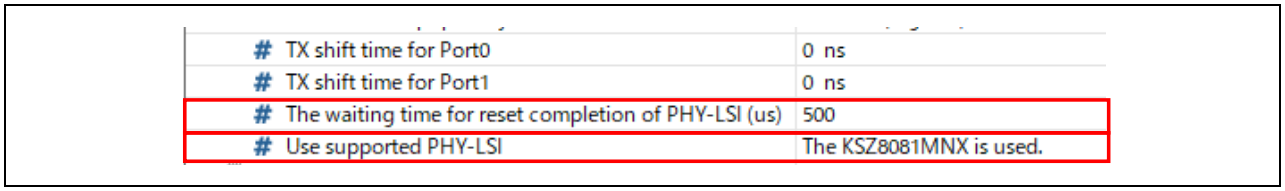


4.1.4 Configuration of EtherCAT FIT module

Set the reset wait time for the PHY LSI and PHY according to the evaluation board.

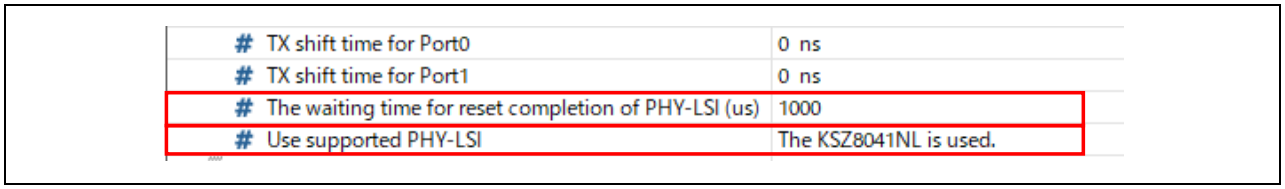
1. In case of using COM board or CPU card

Configurations	Value
The waiting time for reset completion of PHY-LSI (us)	500
Use supported PHY-LSI	The KSZ8081MNX is used.



2. In case of using RSK board

Configurations	Value
The waiting time for reset completion of PHY-LSI (us)	1000
Use supported PHY-LSI	The KSZ8041NL is used.

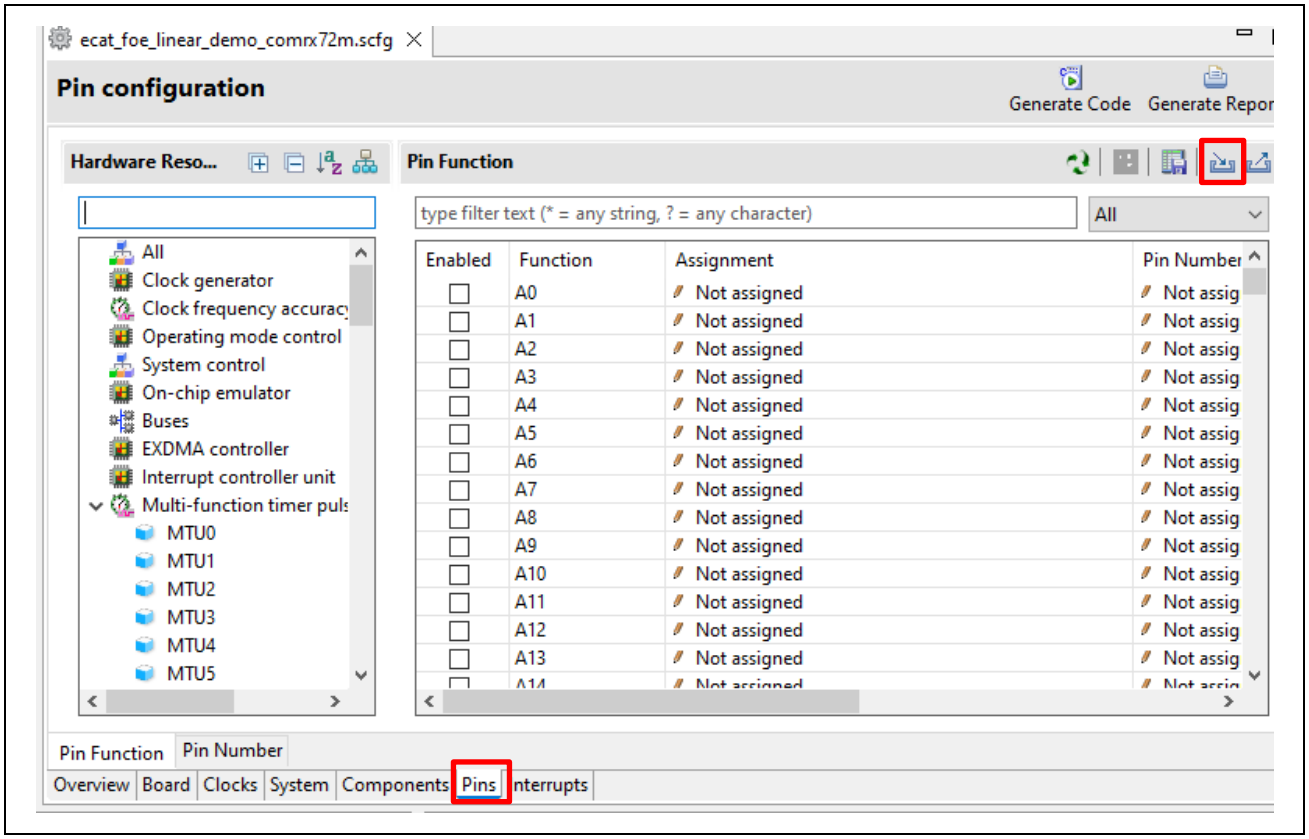


4.2 Pin settings

You can import the terminal settings to run the sample program.

On the [Pins] tab of Smart Configurator, click the [Import Terminal Function Arrangement] button and select "ecat_demo_comrx72m_board.xml".

The xml file is included in the sample program.



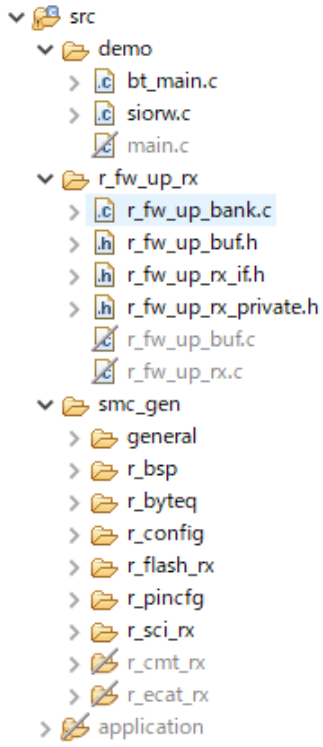
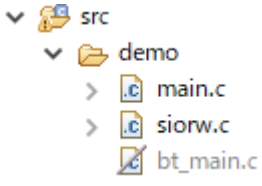
For the terminals used in the FIT module, the following items must be enabled in [Resources] on the [Component] tab of the Smart Configurator.

Component	Resources to enable	Pins to enable
r_ecat_rx	ESC ESC_MII0 ESC_MII1	All pins
r_sci_rx	SCI6	RXD6/SMISO6/SSCL6 pin TXD6/SMOSI6/SSDA6 pin

4.3 build configuration

4.3.1 Linear mode build configuration

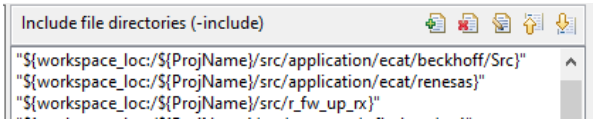
Table 4-1 Linear mode build configuration

Composition name.	Description
BootLoader	Build the boot loader program. Files other than the functionality required by the boot loader are excluded from the build. Right-click the corresponding file or folder and exclusion of build is possible by selecting [Resource Configuration] → [Exclude from Build]. ＜Settings＞ 
BANK0	Build the EtherCAT slave program that you want to download to BANK0. Build artifacts are Motorola S-formatted files. Excludes only demo/bt_main.c from build. ＜Settings＞ 
BANK1	Build the EtherCAT slave program that you want to download to BANK1. Bank0 and bank1 have the same settings except that the code flash mapping and the download file name that is the build artifacts is different.

4.3.1.1 BootLoader

You can check the settings in [Project] → [Properties] → [C/C++ Build] → [Settings].

Table 4-2 BootLoader – tools Settings tab

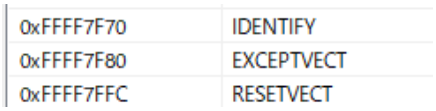
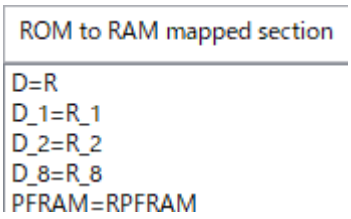
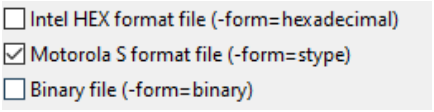
<table>
<tr><th>Item</th><th>Changing Contents</th><th>Description</th></tr>
<tr><td rowspan="2">Compiler -
Source</td><td rowspan="2">Add Include path to
[Include・File・Directory]</td><td>In each FIT module, please add the required include path for the setting.
The code-generated folders are automatically configured only if you use the FIT Configurator to include each FIT module.</td></tr>
<tr><td>About program files that you do not want to generate, you must manually add the include path.
In this project, the following three items have been added.
 
</td></tr>
<tr><td>Compiler -
Optimization</td><td>Change optimization
Level 0 : No performing optimization</td><td>The optimization level is set to 0 due to prevent the const table referenced by bootloader from being built by optimization.</td></tr>
<tr><td>Linker
- Section</td><td>Change the starting position of the PResetPRG placed in the ROM area to 0xFFFF8000.
<Setting>
0xFFFF8000 PResetPRG
C_1
C_2
C
C_8
C$*
D*
W*
L
P
PFRAM
0xFFFFFF80 EXCEPTVECT
0xFFFFFFFC RESETVECT</td><td>The mapping of boot loader is from 0xFFF8000 to 0xFFFFFFFF. Size is 32KB.
The exception vector table of 128 bytes including reset vector is placed at the end of the line.</td></tr>
</table>

4.3.1.2 BANK0

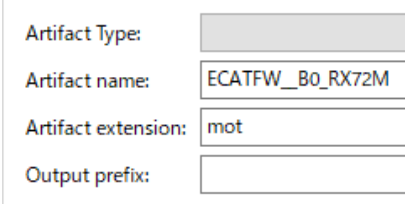
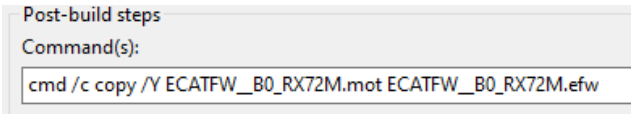
You can check the settings in Project→Properties→C/C++ Build →Settings.

Table 4-3 BANK0 – Setting Tool Tab

Item	Changing Contents	Description
Compiler - Source	Add Include path to [Include・File・Directory]	The settings are the same as those of BootLoader.
Compiler - Source	Add the definition of processor macro <Setting> Macro definition FLASH_APPL_BANK=0 _DISABLE_REVNO_CHECK REVISION_NUMBER=0x00000100	If set “FLASH_APPL_BANK=0”, the code for BANK0 will be built. If define “_DISABLE_REVNO_CHECK” even the firmware and EEPROM revision No are different, the error will not occur. Define “REVISION_NUMBER” as 1.00
Compiler – Optimization	Change optimization level Level 1: Implement a part of the optimization	The optimization level is set to 1 due to prevent some of the code generated by SSC from being built by optimization.
Linker – Section	Add RPFRAM section into RAM area < Setting > Address: 0x00000004 Section Name: SU, SI, B_1, R_1, B_2, R_2, B, R, B_8, R_8, RPFRAM	The rewrite code of the code flash memory is added to the RAM area.
	Change the starting position of the PResetPRG placed in the ROM area to 0xFFE00000. Added PFRAM section. < Setting > 0xFFE00000: PResetPRG, C_1, C_2, C, C_8, C$*, D*, W*, L, P, PFRAM	The mapping of BANK0 is from 0xFFE00000 to 0xFFFF7FFF. Size is 2016KB. Add code to the ROM area to rewrite the code flash memory.

	Add IDENTIFY section in ROM, set The start position to 0xFFFF7F70. < Setting > 	The 16 bytes of table data including in firmware version is placed in the IDENTIFY section. The IDENTIFY section and the exception vector table (128 bytes) including reset vector is placed at the end of BANK0.
Linker – Section – Symbol file	Add the section to map from ROM to RAM < Setting > 	The sample program performs the code flash memory rewrite in RAM. Therefore, add the setting “PFRAM=RPFRAM” that maps form ROM to RAM.
Converter – Output	Change the setting to output Motorola S format file. < Setting > 	The download file is a Motorola S format file.

able 4-4 BANK0 – the other tab

Tab Item	Changing Contents	Description
Build result	Change the output file to ECATFW__B0_RX72.mot < Setting > 	The prefix of the download file name is “ECATFW__B0”.
Build・Step	Create a copy of output file as ECATFW__B0_RX72.efw after completed building. < Setting > 	The reason is that the extension of file can be downloaded by TwinCAT is “efw”.

4.3.1.3 BANK1

You can check the settings in [Project] → [Properties] → [C/C++ Build] → [Settings].

Table 4-5 BANK1 – Setting Tool tab

Item	Changing Contents	Description
Compiler - Source	Add Include path to [Include・File・Directory]	The settings are the same as those of BootLoader.
Compiler - Source	Add the definition of processor macro <Setting> Macro definition FLASH_APPL_BANK=1 _DISABLE_REVNO_CHECK REVISION_NUMBER=0x00000110	If set “FLASH_APPL_BANK=1”, the code for BANK1 will be built. If define “_DISABLE_REVNO_CHECK” even the firmware and EEPROM revision No are different, the error will not occur. Define “REVISION_NUMBER” as 1.10
Compiler – Optimization	Change optimization level Level 1: Implement a part of the optimization	The optimization level is set to 1 due to prevent some of the code generated by SSC from being built by optimization.
Linker – Section	Add RPFRAM section into RAM area < Setting > Address / Section Name 0x00000004 SU SI B_1 R_1 B_2 R_2 B R B_8 R_8 RPFRAM	The rewrite code of the code flash memory is added to the RAM area.
	Change the starting position of the PResetPRG placed in the ROM area to 0xFFC00000. Added PFRAM section. < Setting > 0xFFC00000 PResetPRG C_1 C_2 C C_8 C$* D* W* L P PFRAM	The mapping of BANK1 is from 0xFFC00000 to 0xFFDF7FFF. Size is 2016KB. Add code to the ROM area to rewrite the code flash memory.

	Add IDENTIFY section in ROM, set The start position to 0xFFDF7F70. < Setting > 0xFFDF7F70 IDENTIFY 0xFFDF7F80 EXCEPTVECT 0xFFDF7FFC RESETVECT	The 16 bytes of table data including in firmware version is placed in the IDENTIFY section. The IDENTIFY section and the exception vector table (128 bytes) including reset vector is placed at the end of BANK1.
Linker – Section – Symbol file	Add the section to map from ROM to RAM < Setting > ROM to RAM mapped section D=R D_1=R_1 D_2=R_2 D_8=R_8 PFRAM=RPFRAM	The sample program performs the code flash memory rewrite in RAM. Therefore, add the setting “PFRAM=RPFRAM” that maps form ROM to RAM.
Converter – Output	Change the setting to output Motorola S format file. < Setting > ☐ Intel HEX format file (-form=hexadecimal) ☑ Motorola S format file (-form=stype) ☐ Binary file (-form=binary)	The download file is a Motorola S format file.

Table 4-6 BANK1 - the other tab

Tab Item	Changing Contents	Description
Build result	Change the output file to ECATFW__B1_RX72.mot. < Setting > Artifact Type: Artifact name: ECATFW__B1_RX72M Artifact extension: mot Output prefix:	The prefix of the download file name is “ECATFW__B1”.
Build ・Step	Create a copy of output file as ECATFW__B1_RX72.efw after completed building. < Setting > Post-build steps Command(s): cmd /c copy /Y ECATFW__B0_RX72M.mot ECATFW__B0_RX72M.efw	The reason is that the extension of file can be downloaded by TwinCAT is “efw”.

4.3.2 Build configuration of Dual mode

This chapter describes the build configuration of project on using dual mode.

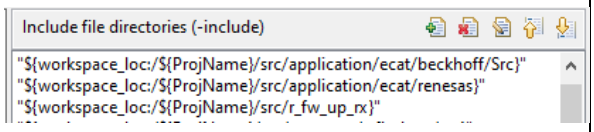
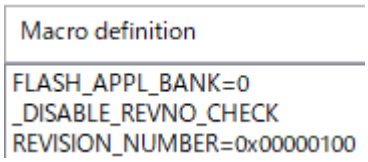
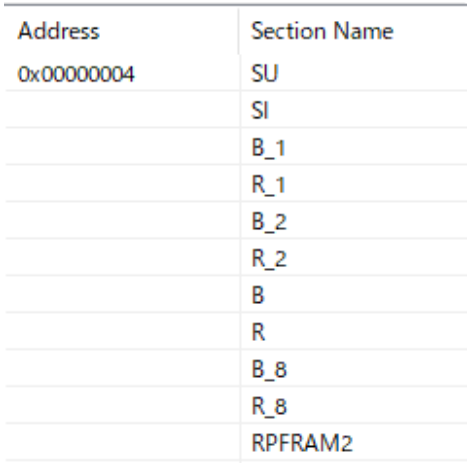
Table 4-7 Build configuration of dual mode

Configuration name	Description
Hardware Debug	Generate a load module with debug information of EtherCAT slave program. There are no files to exclude from the build.
Download	Build download EtherCAT slave program. Result of building is Motorola S format file. There are no files to exclude from the build.

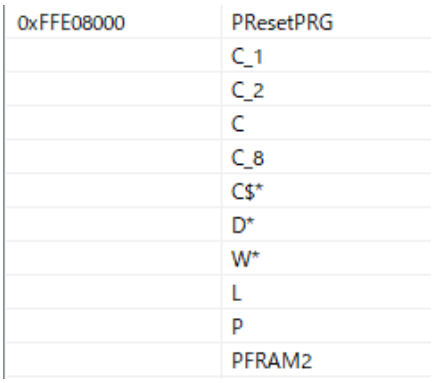
4.3.2.1 HardwareDebug

You can check the settings in [Project] → [Properties] → [C/C++ Build] → [Settings].

Table 4-8 HardwareDebug – Setting Tool tab

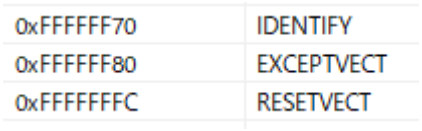
<table>
<tr><th>Item</th><th>Changing Contents</th><th>Description</th></tr>
<tr><td rowspan="2">Compiler
- Source</td><td rowspan="2">Add Include path to
[Include・File・Directory]</td><td>In each FIT module, please add the required include path for the setting.
The code-generated folders are automatically configured only if you use the FIT Configurator to include each FIT module.</td></tr>
<tr><td>About program files that you do not want to generate, you must manually add the include path.
In this project, the following three items have been added.
  </td></tr>
<tr><td>Compiler
- Source</td><td>Add the definition of processor macro.
<Setting>
  </td><td>If set “FLASH_APPL_BANK=0”, the code for BANK0 will be built.
If define “_DISABLE_REVNO_CHECK” even the firmware and EEPROM revision No are different, the error will not occur.
Define “REVISION_NUMBER” as 1.00</td></tr>
<tr><td>Compiler
– Optimization</td><td>Change optimization level.
Level 1: Implement a part of the optimization</td><td>The optimization level is set to 1 due to prevent some of the code generated by SSC from being built by optimization.</td></tr>
<tr><td>Linker
– Section</td><td>Add RPFRAM2 section into RAM area.
< Setting >
  </td><td>Add code to switch bank in RAM area.</td></tr>
</table>

<table>
<tr><td></td><td>Changed the starting position of PResetPRG to 0xFFE08000 in the ROM area.

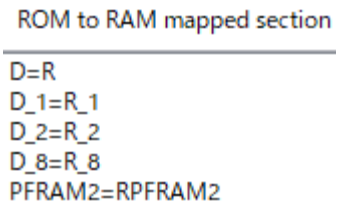
Added PFRAM2 section.
< Setting >
 
<table>
<tr><td>0xFFE08000</td><td>PResetPRG</td></tr>
<tr><td></td><td>C_1</td></tr>
<tr><td></td><td>C_2</td></tr>
<tr><td></td><td>C</td></tr>
<tr><td></td><td>C_8</td></tr>
<tr><td></td><td>C$*</td></tr>
<tr><td></td><td>D*</td></tr>
<tr><td></td><td>W*</td></tr>
<tr><td></td><td>L</td></tr>
<tr><td></td><td>P</td></tr>
<tr><td></td><td>PFRAM2</td></tr>
</table></td><td>The mapping for BANK0 is from 0xFFE00000 to 0xFFFFFFFF, but the starting position of PResetPRG is set to 0xFFE08000.

The size of BANK0 is 2048 KB.

The code for rewriting the code flash memory is added to the ROM area.</td></tr>
<tr><td></td><td>Add IDENTIFY section in ROM, set the start position at 0xFFFFFF70.

< Setting >
 
<table>
<tr><td>0xFFFFFF70</td><td>IDENTIFY</td></tr>
<tr><td>0xFFFFFF80</td><td>EXCEPTVECT</td></tr>
<tr><td>0xFFFFFFFC</td><td>RESETVECT</td></tr>
</table></td><td>The 16 bytes of table data including in firmware version is placed in the IDENTIFY section.

The IDENTIFY section and the exception vector table (128 bytes) including reset vector is placed at the end of BANK0.</td></tr>
<tr><td>Linker
– Section
– Symbol file</td><td>Add the section to map from ROM to RAM

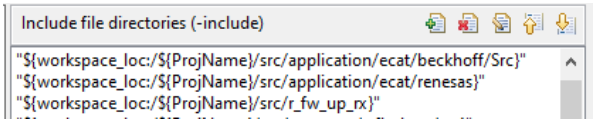
< Setting >
 
<table>
<tr><td>ROM to RAM mapped section</td></tr>
<tr><td>D=R
D_1=R_1
D_2=R_2
D_8=R_8
PFRAM2=RPFRAM2</td></tr>
</table></td><td>The code to switch bank is executed in RAM.

Therefore, add the setting that maps from ROM to RAM.</td></tr>
</table>

4.3.2.2 Download

You can check the settings in [Project] → [Properties] → [C/C++ Build] → [Settings].

Table 4-9 Download – Setting Tool tab

Item	Changing Contents	Description
Compiler - Source	Add Include path to [Include・File・Directory]	In each FIT module, please add the required include path for the setting. The code-generated folders are automatically configured only if you use the FIT Configurator to include each FIT module.
		About program files that you do not want to generate, you must manually add the include path. In this project, the following three items have been added. 
Compiler - Source	Add the definition of processor macro < Setting > Macro definition FLASH_APPL_BANK=1 _DISABLE_REVNO_CHECK REVISION_NUMBER=0x00000110	If set “FLASH_APPL_BANK=1”, the code for BANK1 will be built. If define “_DISABLE_REVNO_CHECK” even the firmware and EEPROM revision No are different, the error will not occur. Define “REVISION_NUMBER” as 1.10
Compiler – Optimization	Change optimization level Level 1: Implement a part of the optimization.	The optimization level is set to 1 due to prevent some of the code generated by SSC from being built by optimization.
Linker – Section	Add RPFRAM2 section into RAM area < Setting > Address: 0x00000004 Section Name: SU, SI, B_1, R_1, B_2, R_2, B, R, B_8, R_8, RPFRAM2	Add the rewrite code of code flash memory in RAM area.

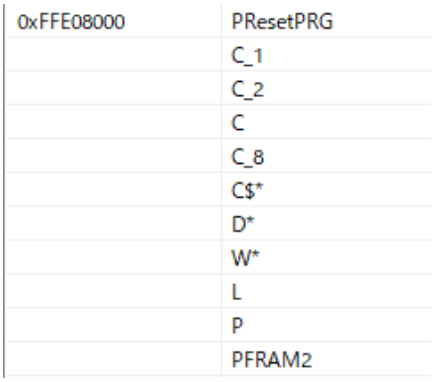
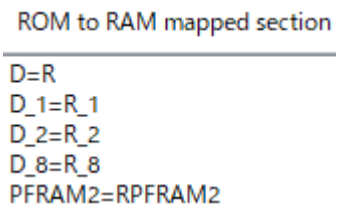
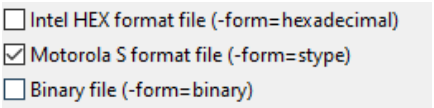
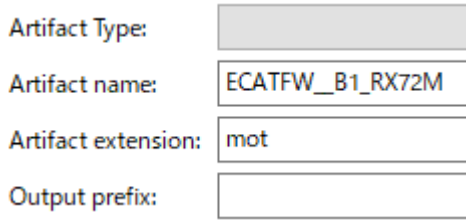
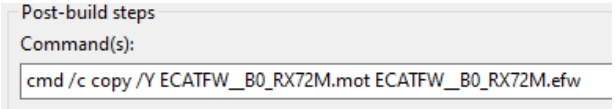
<table>
<tr><td></td><td>Changed the starting position of PResetPRG to 0xFFE08000 in the ROM area.
< Setting >
 
</td><td>The mapping for BANK0 is from 0xFFE00000 to 0xFFFFFFFF, but the starting position of PResetPRG is set to 0xFFE08000.
The size of BANK0 is 2048 KB.
Code for rewriting the code flash memory is added to the ROM area.</td></tr>
<tr><td></td><td>Add IDENTIFY section in ROM, set the start position at 0xFFFFFF70.
< Setting >
 
</td><td>The 16 bytes of table data including in firmware version is placed in the IDENTIFY section.
The IDENTIFY section and the exception vector table (128 bytes) including reset vector is placed at the end of BANK0.</td></tr>
<tr><td>Linker
– Section
– Symbol file</td><td>Add the section to map from ROM to RAM
< Setting >
 
</td><td>The code to switch bank is executed in RAM.
Therefore, add the setting to map form ROM to RAM.</td></tr>
<tr><td>Converter
– Output</td><td>Change the output setting for Motorola S format file.
< Setting >
 
</td><td>The download file is a Motorola S format file.</td></tr>
</table>

Table 4-10 Download - the other tab

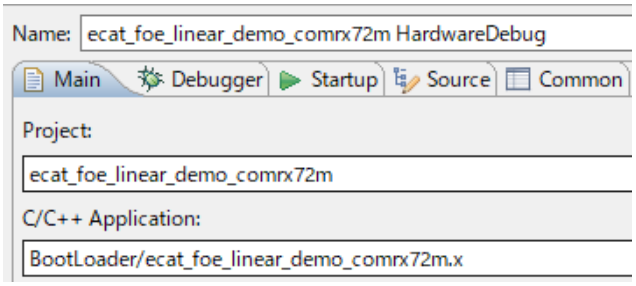
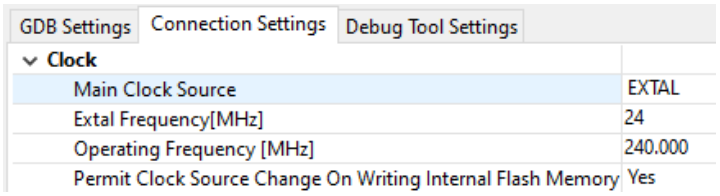
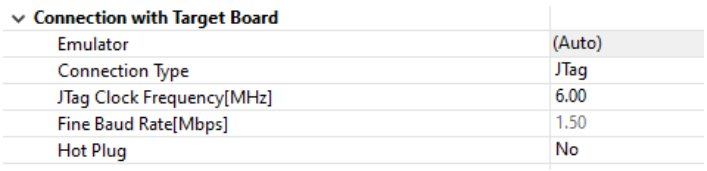
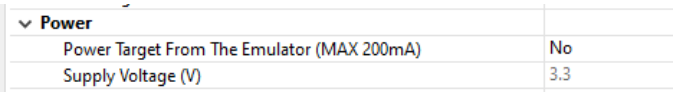
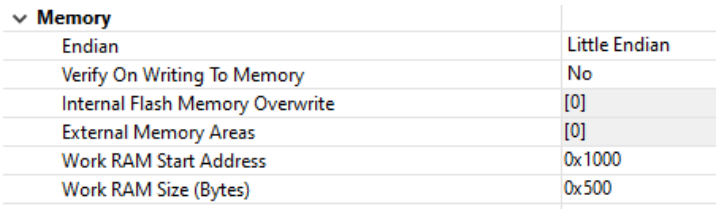
Tab Item	Changing Contents	Description
Build result	Change the output file to ECATFW__B1_RX72.mot. < Setting > 	The prefix of the download file name is “ECATFW__B1”.
Build ・Step	Create a copy of output file as ECATFW__B1_RX72.efw after completed building. < Setting > 	The reason is that the extension of file can be downloaded by TwinCAT is “efw”.

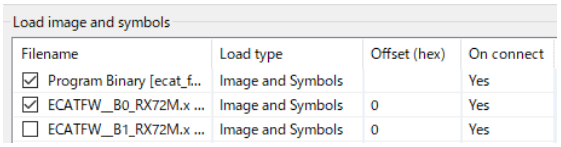
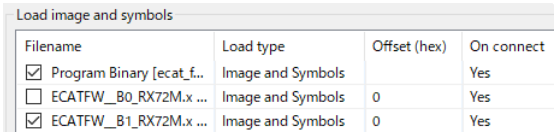
4.4 Debug configuration

4.4.1 Debug configuration of Linear mode

You can check the settings in [Run] → [the debug configuration] → [ecat_foe_linear_demo_comrx72m].

Table 4-11 Debug configuration of linear mode

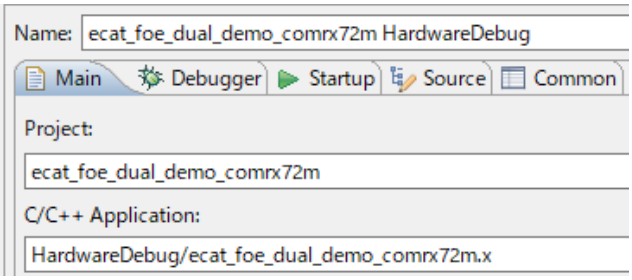
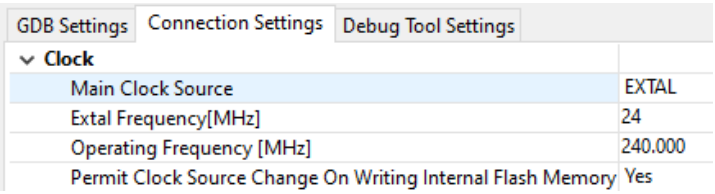
Tab Item	Changing Contents	Description
main	Change C/C++ application to boot loader. < Setting > 	
Debugger – Connection Settings – Clock	Set the main clock source to EXTAL, the EXTAL frequency to 24 MHz, and the operating frequency to 240 MHz < Setting > 	
Debugger – Connection Settings – Connection with Target Board	Set the connection type to "JTAG". In case of using a CPU card, set it to "FINE". < configuration example> 	
Debugger – Connection Settings – Power	Change the setting to not supply power from the emulator. < configuration example> 	
Debugger – Setting Tool Debug	In all blocks change the overwrite of the internal flash memory. < Setting > 	Click on the button at the right of " Overwrite internal flash memory ", click on [Deselect all] → [OK], than all blocks will be deleted and overwritten, and [0] will be displayed.

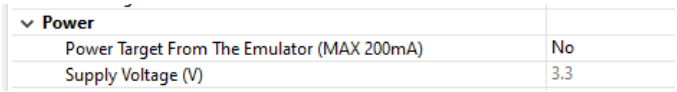
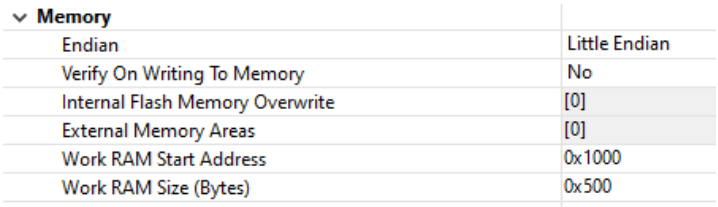
Startup	Set the image to write to code flash along with boot loader. < Setting > Debug BANK0 and boot loader  Load image and symbols Filename / Load type / Offset (hex) / On connect Program Binary [ecat_f... / Image and Symbols / / Yes ECATFW__B0_RX72M.x ... / Image and Symbols / 0 / Yes ECATFW__B1_RX72M.x ... / Image and Symbols / 0 / Yes < Setting > Debug BANK1 and boot loader  Load image and symbols Filename / Load type / Offset (hex) / On connect Program Binary [ecat_f... / Image and Symbols / / Yes ECATFW__B0_RX72M.x ... / Image and Symbols / 0 / Yes ECATFW__B1_RX72M.x ... / Image and Symbols / 0 / Yes	Set the setting to load the images and symbols of the bank you want to debug. Specify the download module from [Add] → [search project].

4.4.2 Debug configuration of Dual mode

You can check the settings in [Run] → [the debug configuration] → [ecat_foe_linear_demo_comrx72m].

Table 4-12 Debug configuration of Dual mode

Tab Item	Changing Contents	Description
main	Set C/C++application. < Setting >  Name: ecat_foe_dual_demo_comrx72m HardwareDebug Main / Debugger / Startup / Source / Common Project: ecat_foe_dual_demo_comrx72m C/C++ Application: HardwareDebug/ecat_foe_dual_demo_comrx72m.x	
Debugger – Connection Settings – Clock	Set the main clock source to EXTAL, the EXTAL frequency to 24 MHz, and the operating frequency to 240 MHz. < Setting >  GDB Settings / Connection Settings / Debug Tool Settings Clock Main Clock Source: EXTAL Extal Frequency[MHz]: 24 Operating Frequency [MHz]: 240.000 Permit Clock Source Change On Writing Internal Flash Memory: Yes	
Debugger – Connection Settings – Connection with Target Board	Set the connection type to "JTAG". In case of using a CPU card, set it to "FINE". < configuration example>  Connection with Target Board Emulator: (Auto) Connection Type: JTag JTag Clock Frequency[MHz]: 6.00 Fine Baud Rate[Mbps]: 1.50 Hot Plug: No	

Debugger – Connection Settings – Power	Change the setting to not supply power from the emulator. < configuration example>  	
Debugger – Connection Settings – Communication Mode	[CPU operation mode] – [Change startup bank] is set to "No" to start the user program of BANK0.	When the debugger is starting, the BANKSWP[2:0] bits are set to ”111b”.
Debugger – Debug Tool Setting	In all blocks change the overwrite of the internal flash memory. < Setting >  	Click on the button at the right of “ Overwrite internal flash memory ”, click on [Deselect all] → [OK], than all blocks will be deleted and overwritten, and [0] will be displayed.

5. Building and debugging projects

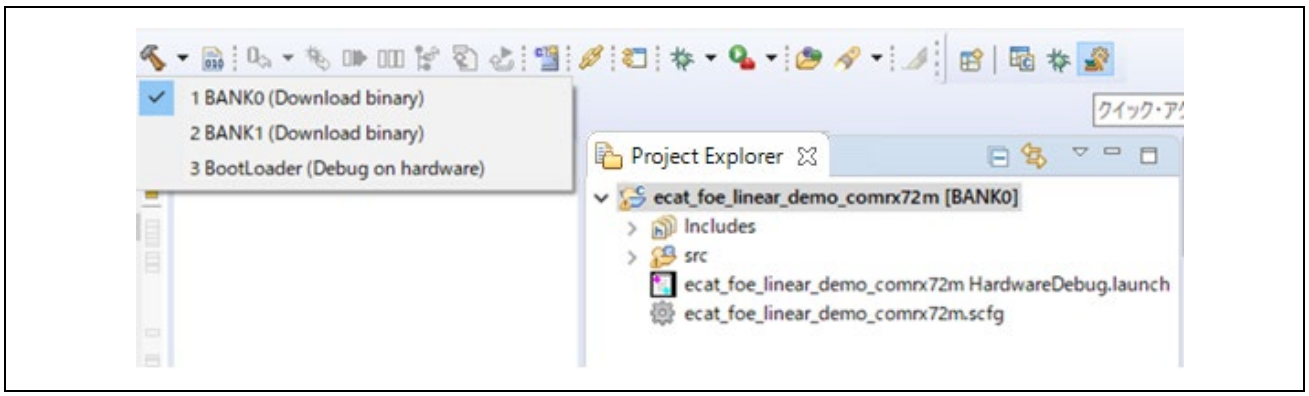
5.1 Generate Code

(1) Open the smart configuration file [ecat_foe_linear_demo_comrx72m.scfg] and press [Code Generation] to perform code generation.

(2) The generated code is stored under “ecat_foe_linear_demo_comrx72m¥src¥smc_gen folder”.

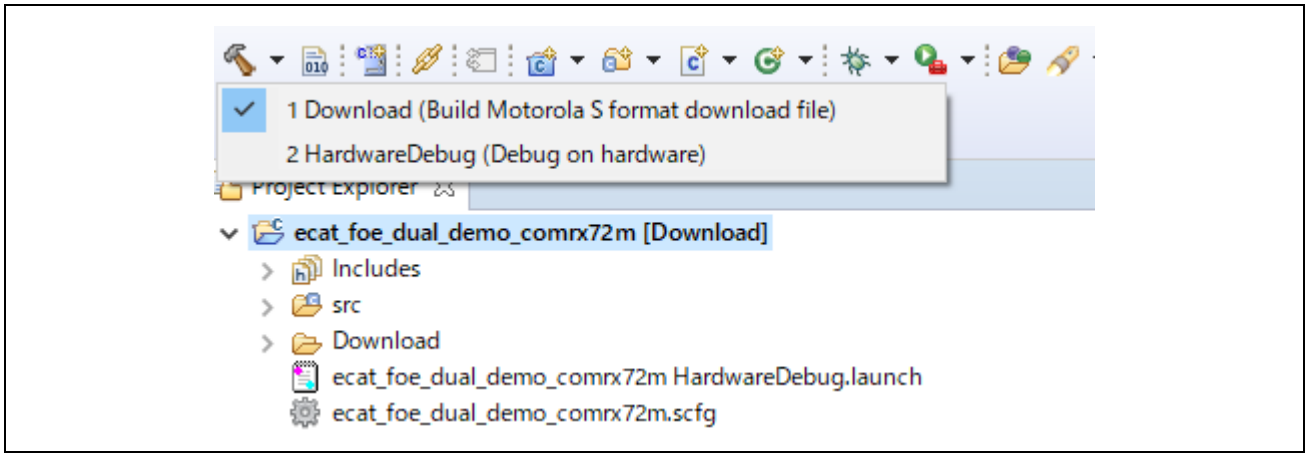
5.2 Build on Linear mode

(3) Click the arrow next to the Build button (hammer icon) in the toolbar and select [BANK0], [BANK1], [BootLoader] from the drop-down menu.



5.3 Build on Dual mode

(1) Click the arrow next to the Build button (hammer icon) in the toolbar and select [Download], [HardwareDebug] from the drop-down menu.



5.4 Preparation for Debug

The settings of the evaluation board for operating this sample program are shown below.

Table 5-1 the list of evaluation board setting

Setting Item	MCU	Evaluation board name	Setting contents
LAN Cable	RX72M	COM board	Connected to the "ECAT IN" side
		CPU card	Connected to the "CN8" side
		RSK board	Connected to the "ECAT IN" side
Debugger	RX72M	COM board	Connecting the E2 Lite to the JTAG Connector
		CPU card	Connect the USB cable to the USB connector
		RSK board	Connecting the E2 Lite to the JTAG Connector

(1) Follow Table 5-1 to connect the LAN cable to the PC.

(2) Follow Table 5-1 to connect the debugger to the PC When using the E2 Lite, the "ACT" LED flashes.

(3) The "Found New Hardware" wizard will appear, follow the steps below to install the driver. For Windows™ 7/8/8.1, administrator privileges are required.

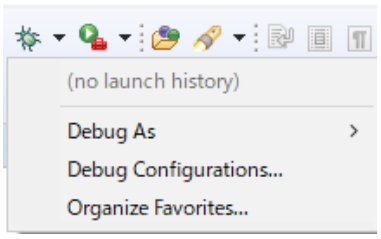
Windows™ 7/8/8.1: When the installation is complete, the Windows taskbar will notify you of the completion of the installation.

Windows™ 10: The device settings button appears on the Windows taskbar and is auto installed.

(4) Provide power to the evaluation board.

5.5 Debug on Linear mode

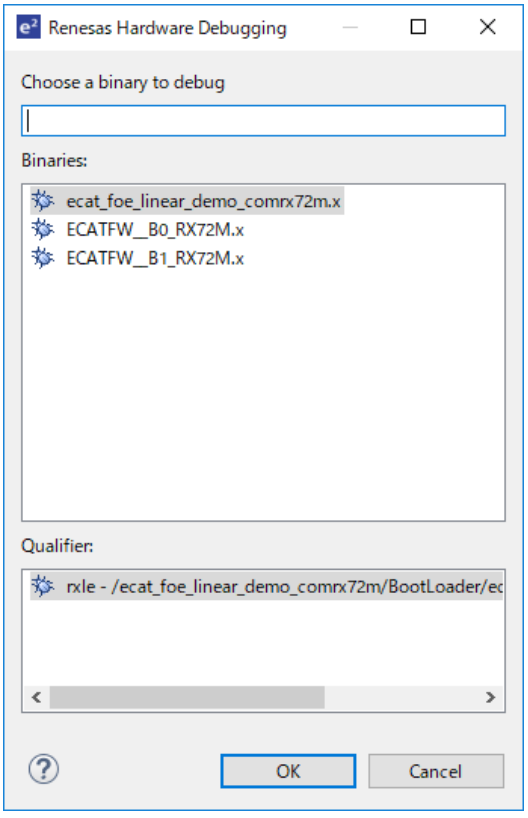
(1) Click on the arrow next to the [Debug] button (Bug icon), select [Debug configurations], than it will start debugging.



(2) Click on “ecat_foe_linear_demo_comrx72m HardwareDebug” to download the program into the target and press debug button to start.

About the combination of module that you want to debug refer to [4.3.1 Build configuration of Linear mode]

(3) When the following window show up, select “ecat_foe_linear_demo_comrx72m.x”.



(4) The firewall warning 'e2-server-gdb.exe' may be shown. Check [Private Network such as home and office network] checkbox and click on < Allow Access >.

(5) The User Account Control (UAC) dialog is shown. Enter your administration password and click [YES].

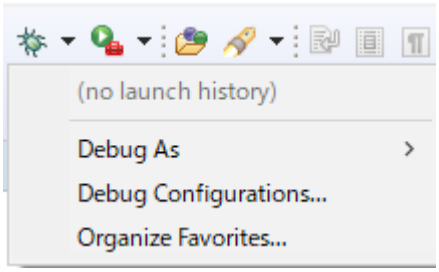
(6) If there is recommended dialog to change the perspective display in the confirmation dialog for switching perspective. Check “Always use this setting” checkbox and click on [Yes].

(7) The green [ACT] LED light of E2 Lite debugger is always on.

(8) After download the code, click on < Resume > button and executes the code until the first line of main () function. Click on < Resume > button again to run the target with the rest of the code.

5.6 Debug on Dual mode

(1) Click on the arrow next to the [Debug] button (Bug icon), by select [Debug configurations], it will start debugging.



(2) Click on “ecat_foe_linear_demo_comrx72m HardwareDebug” to download the program into the target and press debug button to start.

(3) The firewall warning 'e2-server-gdb.exe' may be shown. Check [Private Network such as home and office network] checkbox and click on < Allow Access >.

(4) The User Account Control (UAC) dialog is shown. Enter your administration password and click [YES].

(5) If there is recommended dialog to change the perspective display in the confirmation dialog for switching perspective. Check “Always use this setting” checkbox and click on [Yes].

(6) The green [ACT] LED light of E2 Lite debugger is always on.

(7) After download the code, click on < Resume > button and executes the code until the first line of main () function. Click on < Resume > button again to run the target with the rest of the code.

6. Connection with TwinCAT

This section explains how to operate the sample program using TwinCAT3.

6.1 Preparation of ESI file

Before starting TwinCAT please copy the ESI file included in the sample program to the prescribed location of TwinCAT (¥TwinCAT¥3.x¥Config¥IO¥EtherCAT).

ecat_foe_linear_demo_comrx72m¥utilities¥esi¥RX72M EtherCAT FoE.xml

6.2 Starting TwinCAT

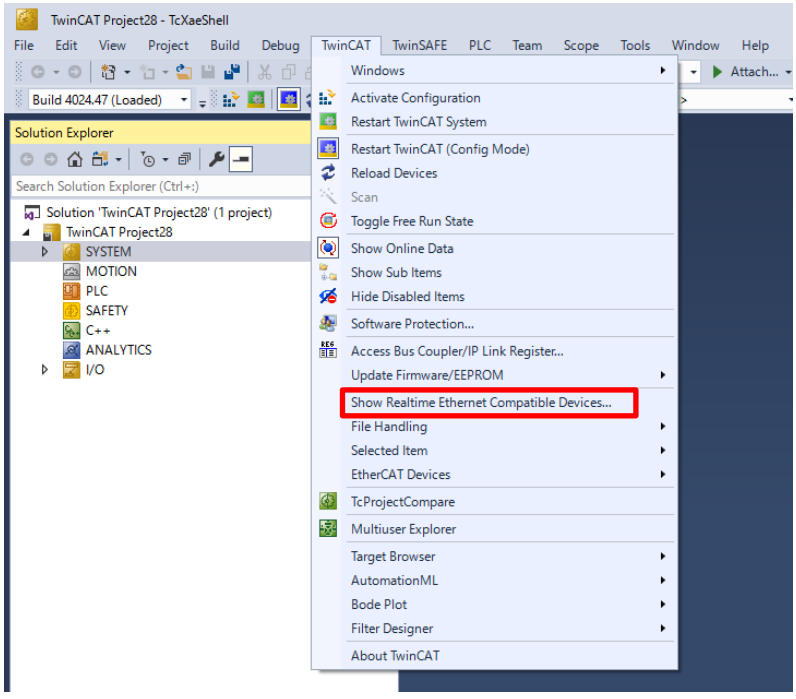
(1) From the start menu, select [Beckhoff] → [TwinCAT 3] → [TwinCAT XAE (VS 20 xx)].

(2) After starting the program, create a new project of type TwinCAT XAE Project as [File] → [New] → [Project].

6.3 Add Ether driver

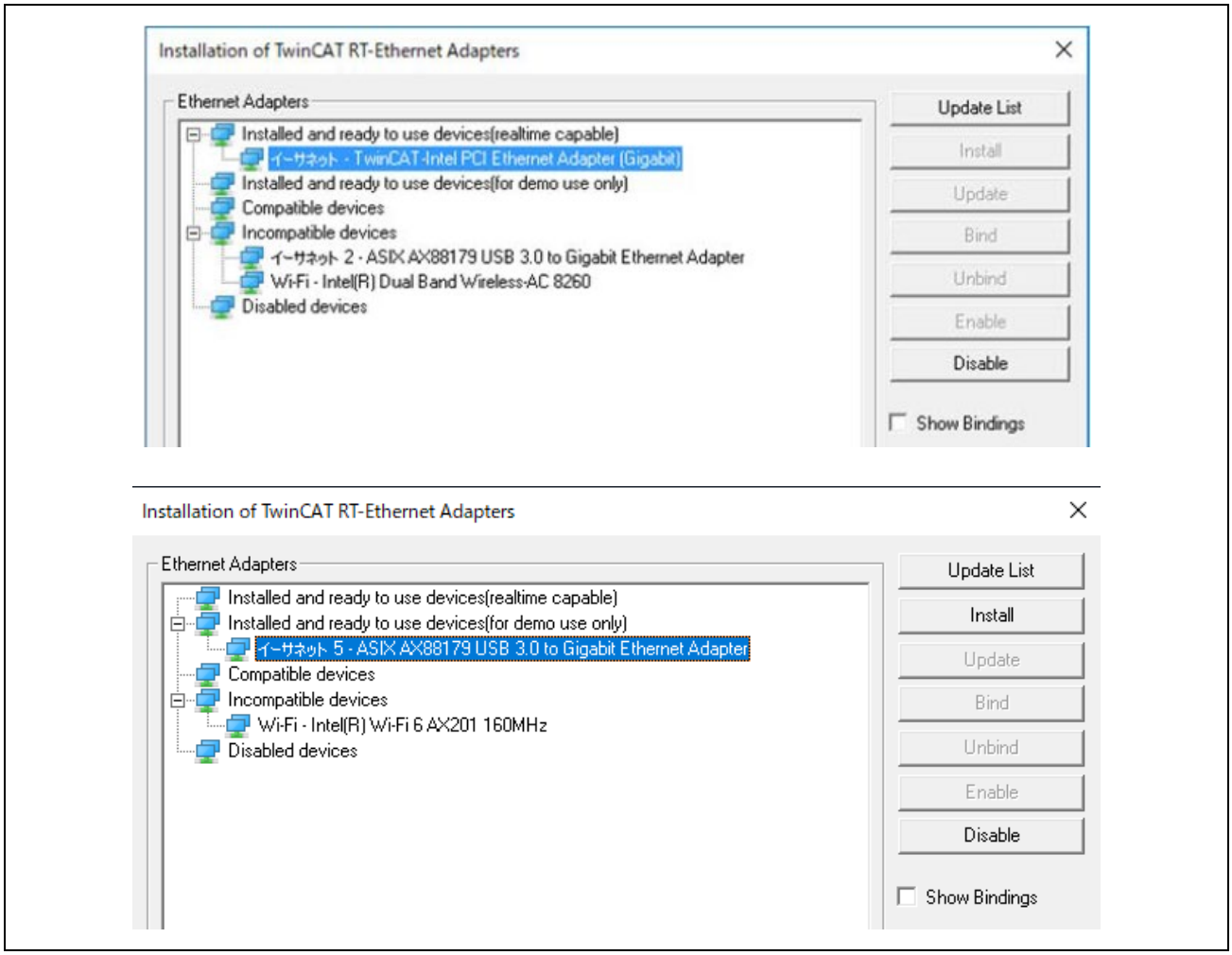
*If you have already processed this section, you do not need to process this section.

From the top menu bar, select [TwinCAT] → Select the [Show Realtime Ethernet Compatible Devices…]



After selecting the Ethernet adapter connected to the PC, press [Install] to install it.

Make sure that you add the installed driver to Installed and ready to use devices (realtime capable) or Installed and ready to use devices (for demo use only).



6.4 Scanning the network

(1) In the System Manager tree, right-click [I / O] → [Devices] and select [Scan].

(2) Click [OK] on the [HINT: Not all types of devices can be found automatically] dialog

(3) In the [new I / O devices found] dialog box, select the check box of the Ethernet adapter to be scanned and click [OK].

(4) Clicking [Yes] in the [Scan for Boxes] dialog starts scanning and the devices in the EtherCAT segment are automatically recognized.

(5) "Active Free Run" dialog is displayed and click [Yes].

In the system manager tree, if Box is added as "Device1" → "Box1" under "I/O" → "Devices", it is normal.

6.5 Writing of SII EEPROM

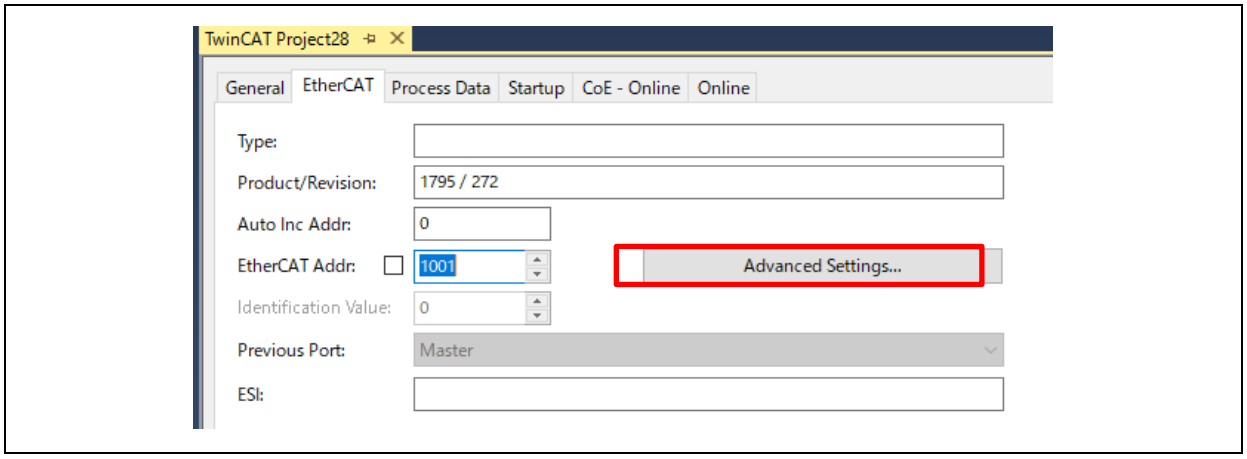
* Because the EEPROM is blank at the time of shipment of the communication board, be sure to perform writing.

* If the EEPROM has been written, the processing in this section is unnecessary.

If the EEPROM is blank, it is displayed as "Box 1 (PFFFFFFFF RFFFFFFFF)" in the system manager tree.

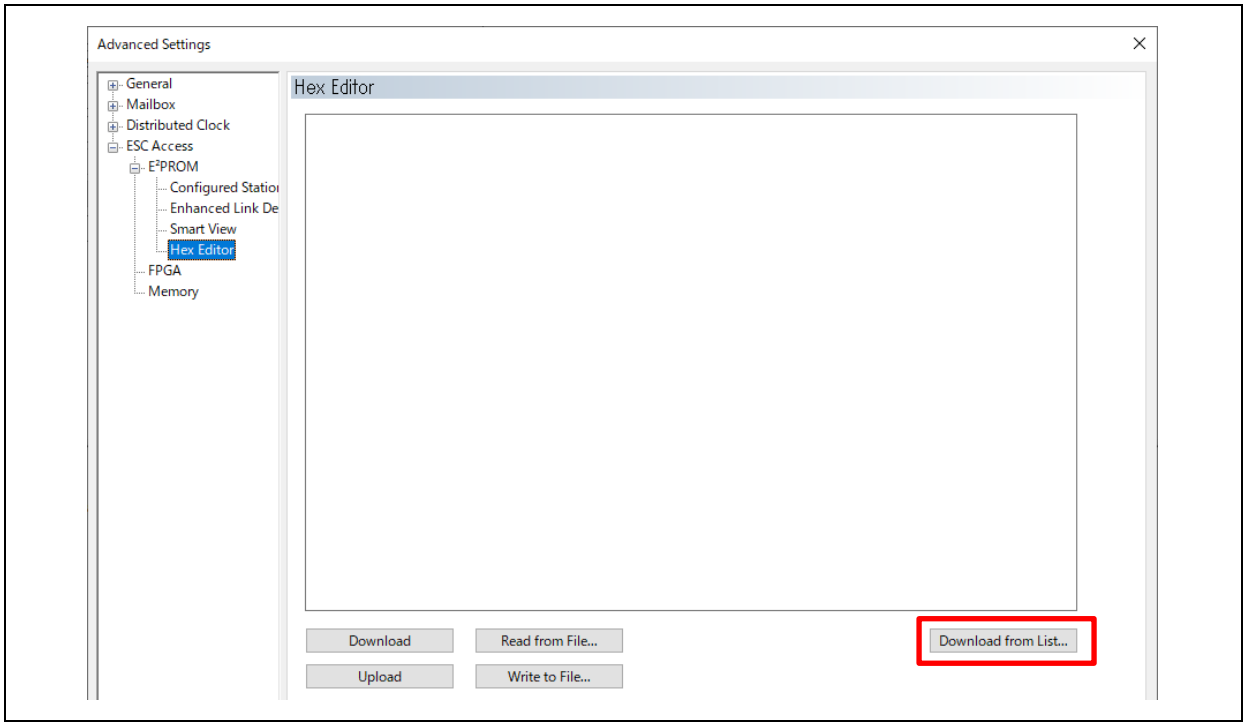
(1) Double-click [Box 1] in the System Manager tree, the panel will be displayed on the right side.

(2) Select the [EtherCAT] tab and click the [Advanced Settings] button.

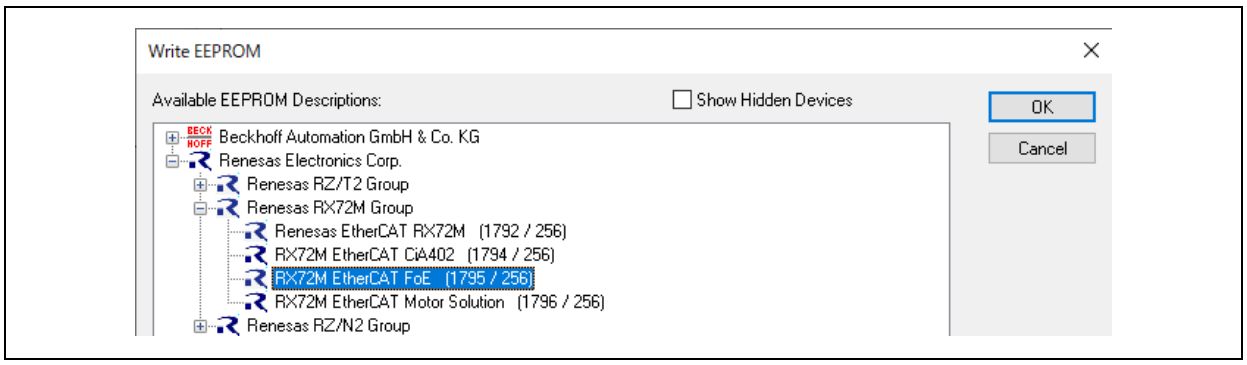


(3) Select [ESC Access] -> [EEPROM] -> [Hex Editor] in the left tree of the Advanced Settings dialog.

(4) In the [Hex Editor] dialog, select "Download from list".



(5) In the [Write EEPROM] dialog box, select [Renesas Electronics Corp.] → [Renesas RX72M Group] → [RX72M EtherCAT FoE] and click [OK]. The EEPROM is written.



(6) After writing, restart the communication board (turn the power supply on again or reset), so that the rewritten data is reflected in the operation of the microcomputer.

6.6 Rescan of the device

(1) Delete [Device x] under [devices] in [I/O].

(2) In the System Manager tree again, right-click [I/O] → [Devices] and select [Scan].

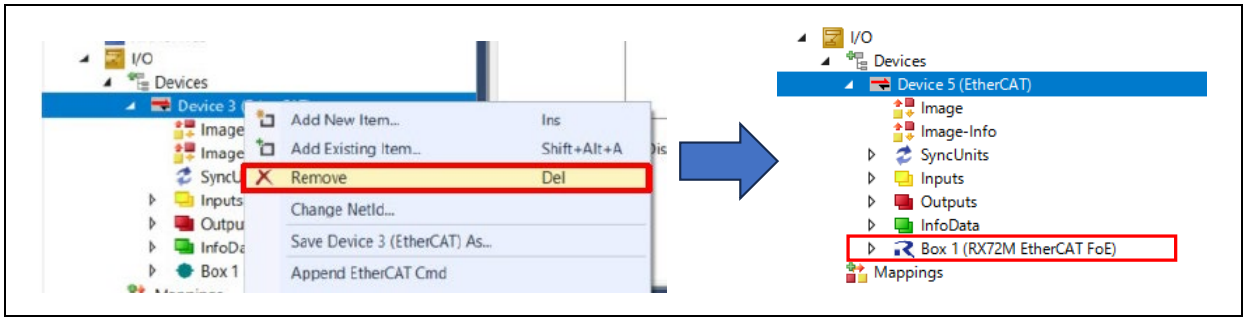
(3) In the [HINT: Not all types of devices can be found automatically] dialog, click [OK].

(4) In the [new I/O devices found] dialog, select the check boxes of the Ethernet adapters you want to scan, and then click [OK].

(5) In the [Scan for Boxes] dialog, click [Yes].

(6) In the [Active Free Run] dialog, click [Yes].

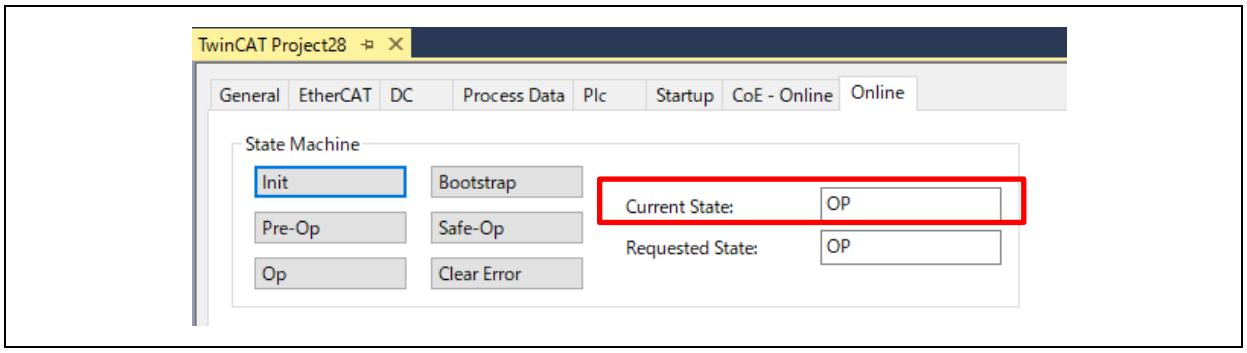
If "Box1" in the System Manager tree is "Box1 (RX72M EtherCAT FoE)", it is normal.



6.7 I/O operation confirmation

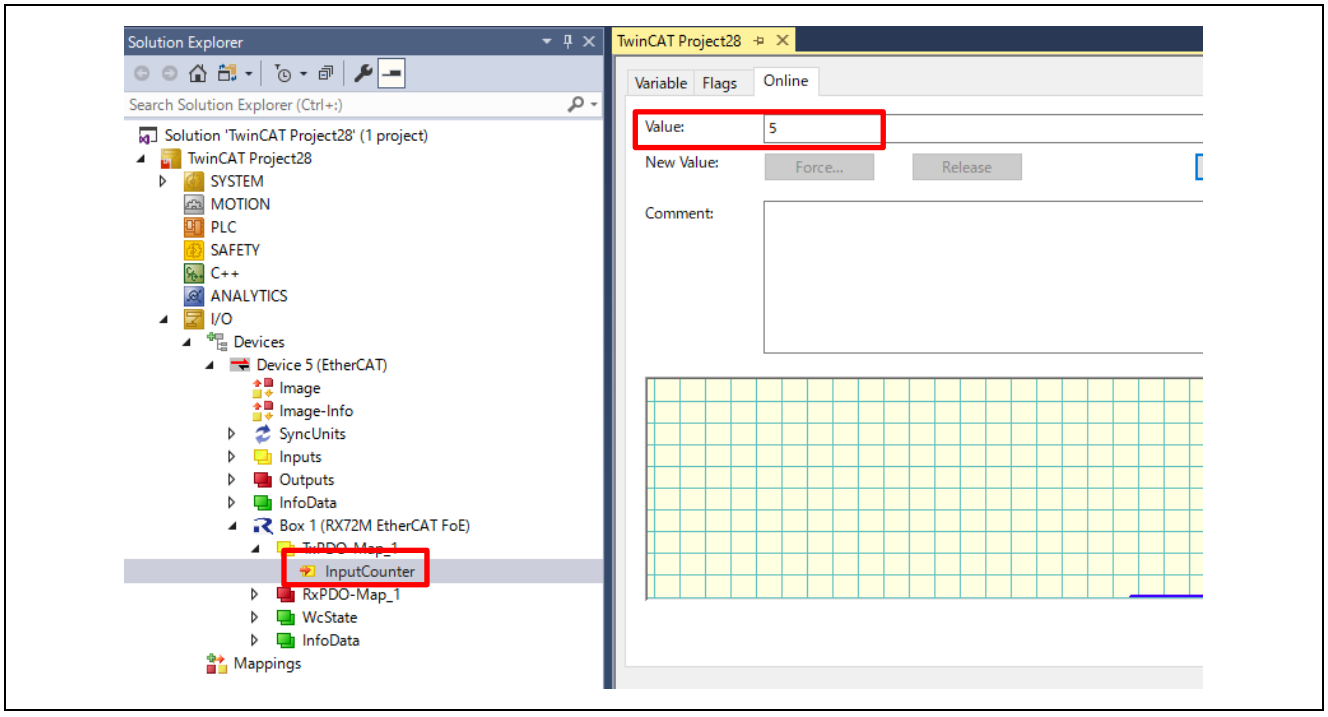
(1) Double-click "Box 1" in the System Manager tree, the panel will be displayed on the right side.

(2) Select the "Online" tab and make sure "Current Status" is "OP".



(3) Expand the + next to "Box 1" in the System Manager tree.

(4) Select "TxPDO-Map_1" → "Input Counter" and select the "Online" tab on the right-side panel to display "Value".



*The value of "InputCounter" is determined by the DIP SW or jumper pins on the evaluation board. Table 6-1 shows the DIP SW or jumper pins used as the value of "InputCounter" in this sample program.

Table 6-1 DIP SW or jumper pin (JP) used as the value of "InputCounter"

Evaluation Board	Used DIPSW or JP	Upper and lower limits of numbers
COM board	DIP SW 5	0 ~ 255
CPU card	JP 5	0 ~ 7
RSK board	DIP SW 5	0 ~ 255

(5) Select "RxPDO-Map_1" -> "Output Counter" in the system manager tree and "Value" is displayed when you select the "Online" tab on the right panel.

Make sure that "0" is displayed.

(6) Click [Write] and enter an arbitrary numerical value in the "Set Value Dialog" dialog.

(7) Click "OK" and confirm that the value "Value" is the entered value.

Depending on the value you enter, the LED on the evaluation board will light up.
(LED lights up when bit = 1)

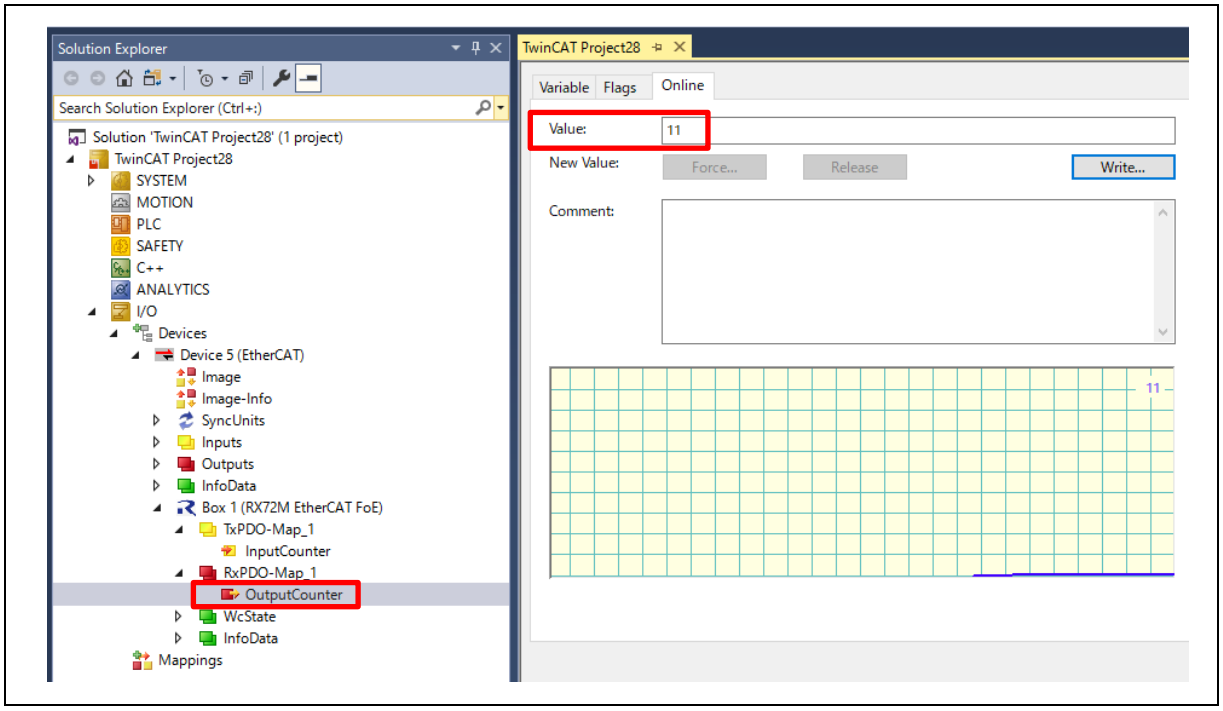


*Table 6-2 shows the correspondence between the "OutputCounter" defined in this program and the LEDs on each evaluation board.

Table 6-2 Mapping of "OutputCounter" to the LEDs on Each Evaluation Board

Evaluation Board	Value of ”OutputCounter”	LEDs that light up
COM board	Bit 0 is 1	LED 1
	Bit 1 is 1	LED 2
	Bit 2 is 1	LED 3
	Bit 3 is 1	LED 4
CPU card	Bit 0 is 1	LED 6
	Bit 1 is 1	LED 7
RSK board	Bit 0 is 1	LED 1
	Bit 1 is 1	LED 2
	Bit 2 is 1	LED 3
	Bit 3 is 1	LED 4

7. Operation check by TwinCAT

7.1 Firmware writing

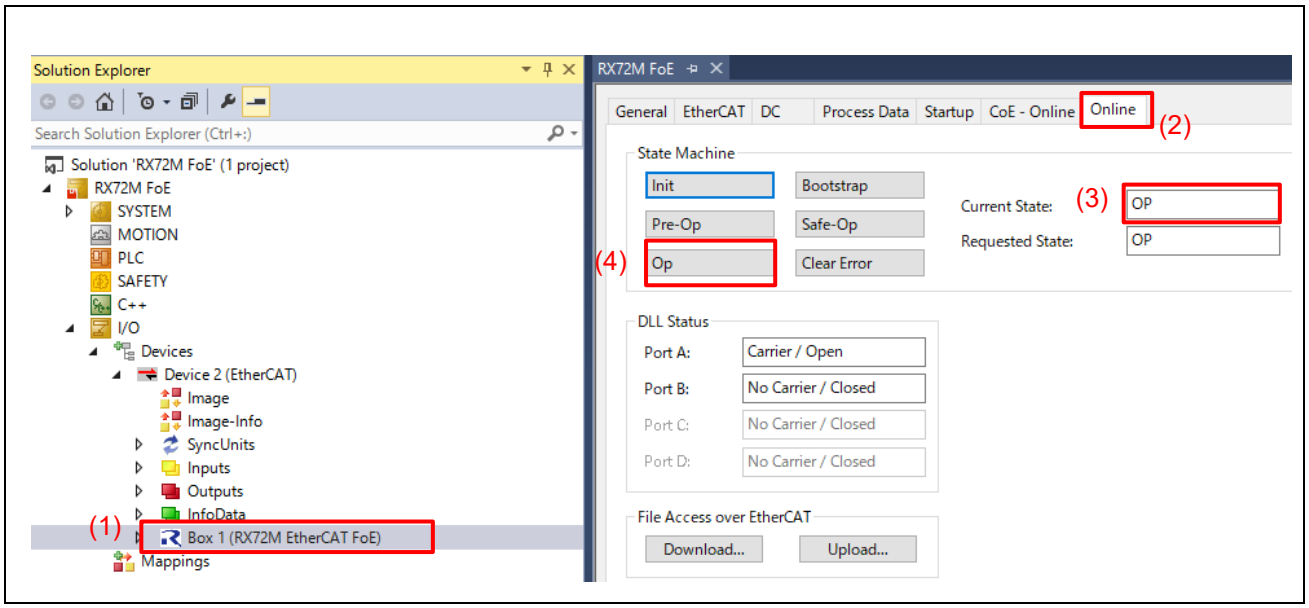
This chapter describes the procedure to write the new revision firmware to BANK1 while the program is running on BANK0.

(1) Select "Box 1 (RX72M EtherCAT FoE)" in "Solution Explorer"

(2) Click the "Online" tab

(3) Make sure the Current State is “OP”.

(4) If it is not "OP", press the "Op" button to transition to "OP"



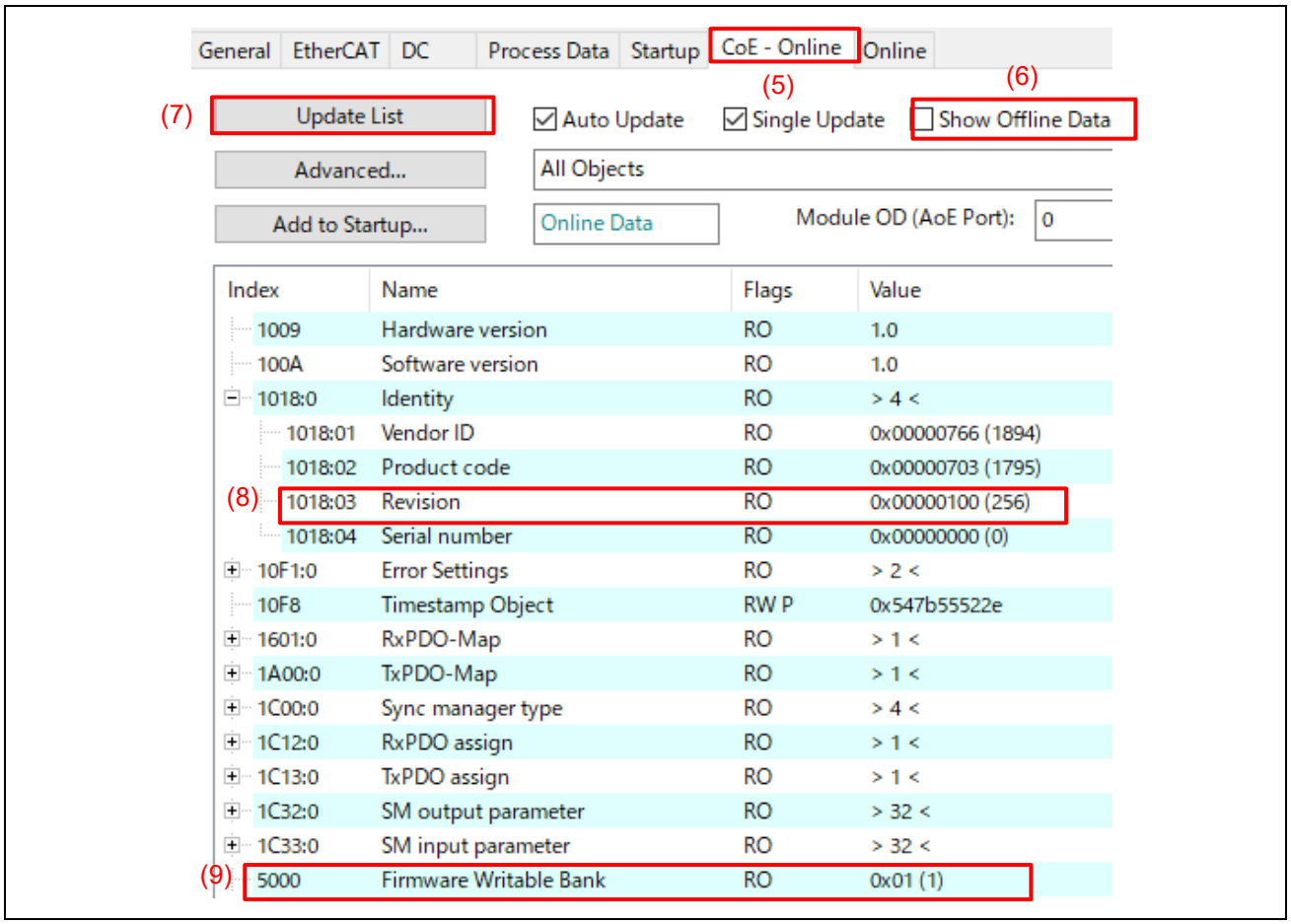
(5) Click the “CoE – Online”.

(6) Confirm "Show Offline Data" is not checked. If it is checked, please clear it.

(7) Press the "Update List" button to update the CoE list.

(8) Check the value of Index 1018:03 Revision. In the example, 0x00000100(256), which means Rev 1.00, is displayed.

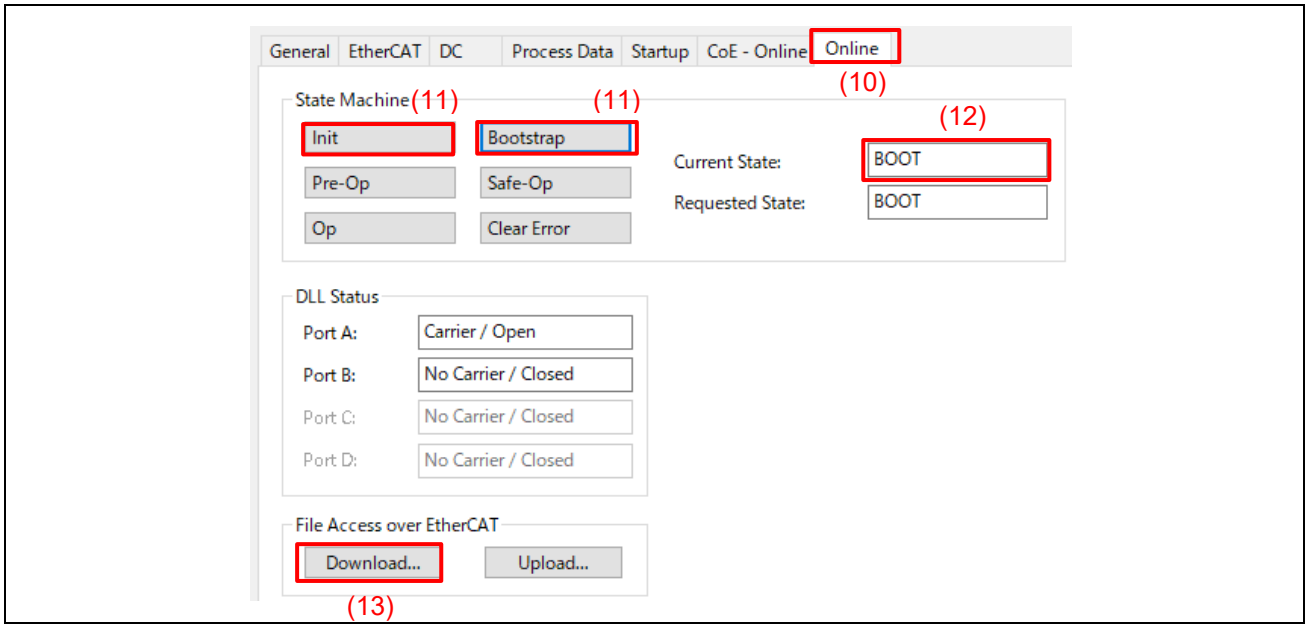
(9) Check the value of Index 5000 Firmware Writable Bank. In the example, the writable bank is BANK1, so 0x01(1) is displayed.



(10) Click on the “Online”” tab

(11) Press “Init” button -> “Bootstrap” button in sequence.

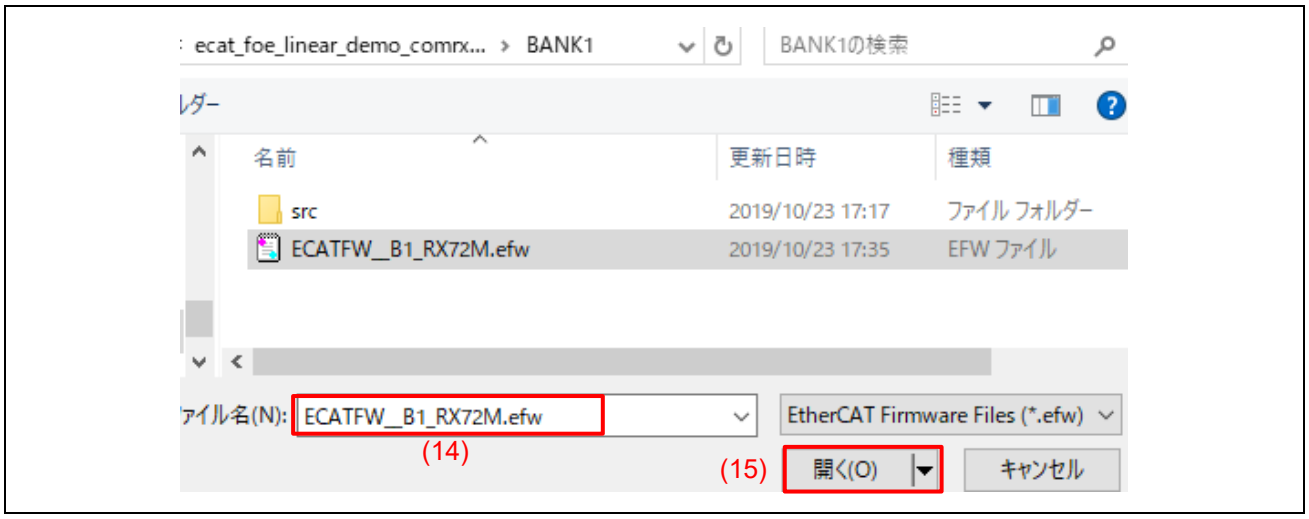
(12) Confirm that the Current State changes to "BOOT"



(13) Click the "Download" button of File Access over EtherCAT to open the window for selecting the download file.

(14) Select "ECAT__B1_RX72M.efw" which is the download file for BANK1

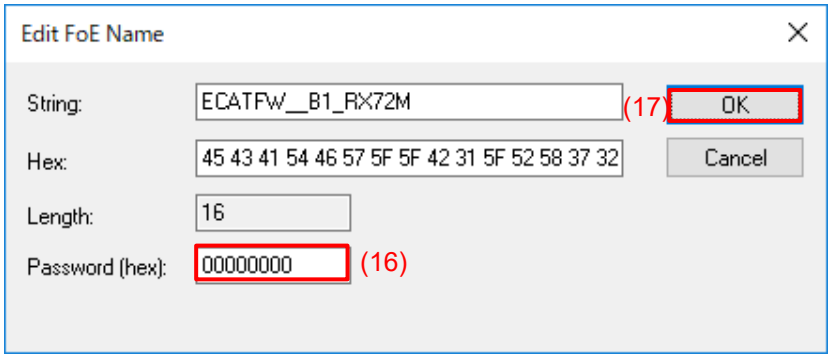
(15) Pressing the "Open".



The file name editing window will open.

(16) Password remains of "00000000"

(17) Press "OK"



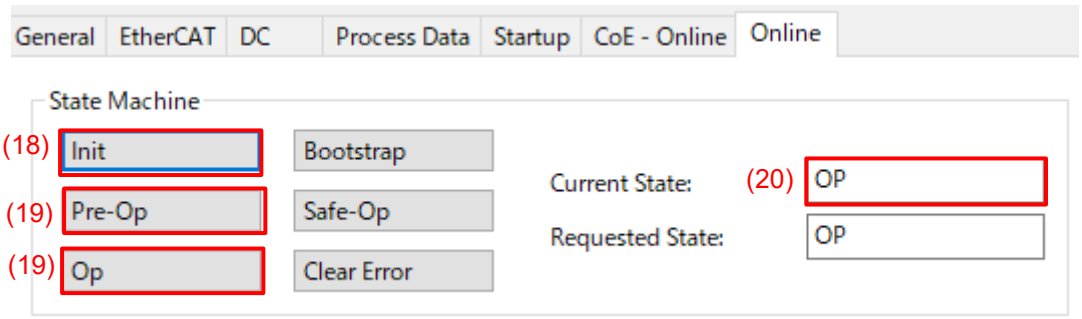
The download status is displayed along with the message "Downloading" on the bottom left of TwinCAT System Manager screen. If no error message is displayed and the above window disappears and the status is "Ready", the firmware update was successful.

In the "Online"" tab

(18) Press the "Init" button to restart with the updated firmware.

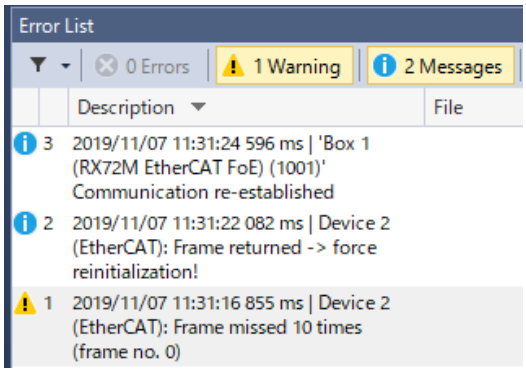
(19) Press "Preop" button -> "Op" button.

(20) Current State transitions to "OP" and can see the updated firmware action.



In case of linear mode, can verify that EtherCAT does not leak links during reboots.

In case of dual mode, the software is reset when the system is restarted, so the link will be broken and the Warning message shown below will be displayed, but this is not a problem.



Click the “CoE – Online” tab again and click the “Update List” button to update the CoE list.

(21) Check the value of Index 1018:03 Revision. You can see that it changed to Rev 1.10 of BANK 1 revision 0x00000110(272)

(22) Check the value of Index 5000 Firmware Writable Bank. The writable bank has changed from BANK1 to BANK0, so 0x00(0) is displayed.

⊟	1018:0	Identity	RO	> 4 <
	1018:01	Vendor ID	RO	0x00000766 (1894)
	1018:02	Product code	RO	0x00000703 (1795)
(21)	1018:03	Revision	RO	0x00000110 (272)
	1018:04	Serial number	RO	0x00000000 (0)
⊞	10F1:0	Error Settings	RO	> 2 <
	10F8	Timestamp Object	RW P	0x244348b266
⊞	1601:0	RxPDO-Map	RO	> 1 <
⊞	1A00:0	TxPDO-Map	RO	> 1 <
⊞	1C00:0	Sync manager type	RO	> 4 <
⊞	1C12:0	RxPDO assign	RO	> 1 <
⊞	1C13:0	TxPDO assign	RO	> 1 <
⊞	1C32:0	SM output parameter	RO	> 32 <
⊞	1C33:0	SM input parameter	RO	> 32 <
(22)	5000	Firmware Writable Bank	RO	0x00 (0)

7.2 Firmware readout

It is possible to read the binary data of the firmware stored in the BANK area where the program is executed.

The binary data is saved in the EtherCAT master as an upload file.

The format of the upload file is shown in Table 7-1.

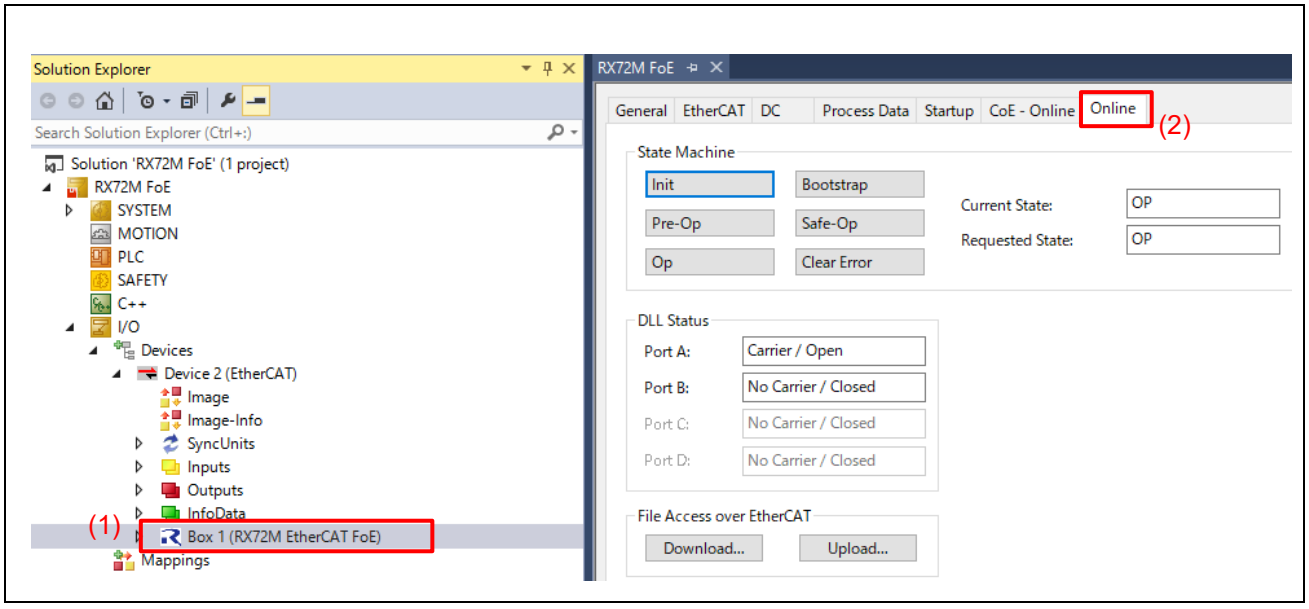
Table 7-1 Format of the upload file

Item	Contents	
File name prefix	”ECATFW__”	
File extension	bin	
File Formats	Binary format * Note that this is different from the Motorola S format of the download file.	
File size	Linear mode	4MB Product : 2016KB 2MB Product : 992KB
	Dual mode	4MB Product : 2048KB 2MB Product : 1024KB
Read target BANK	Linear mode	BANK1 when Firmware Writable Bank = 0 BANK0 when Firmware Writable Bank = 1
	Dual mode	Always BANK0

This section describes the procedure for reading the binary data of the firmware.

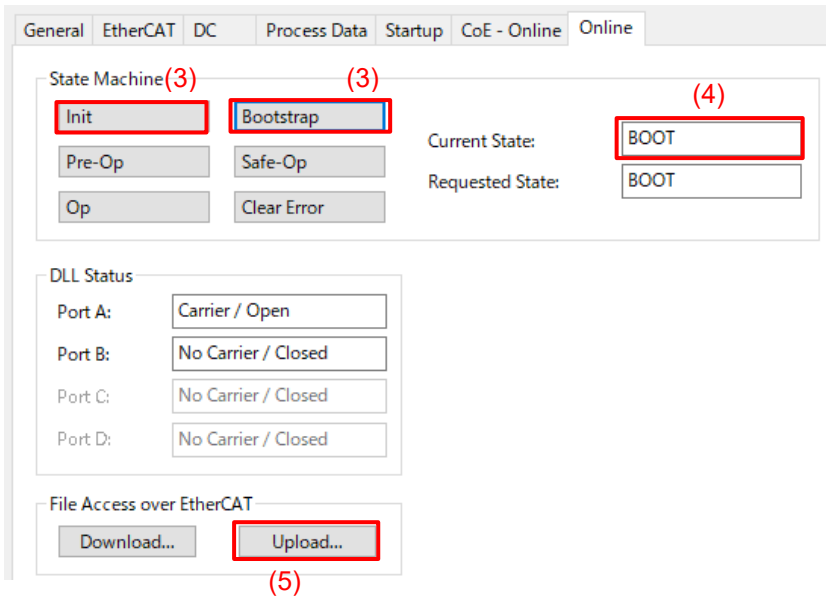
(1) After selecting "Box 1 (RX72M EtherCAT FoE)" in "Solution Explorer"

(2) Click the “Online” tab.



(3) Press the “Init” button-> “Bootstrap” button in order,

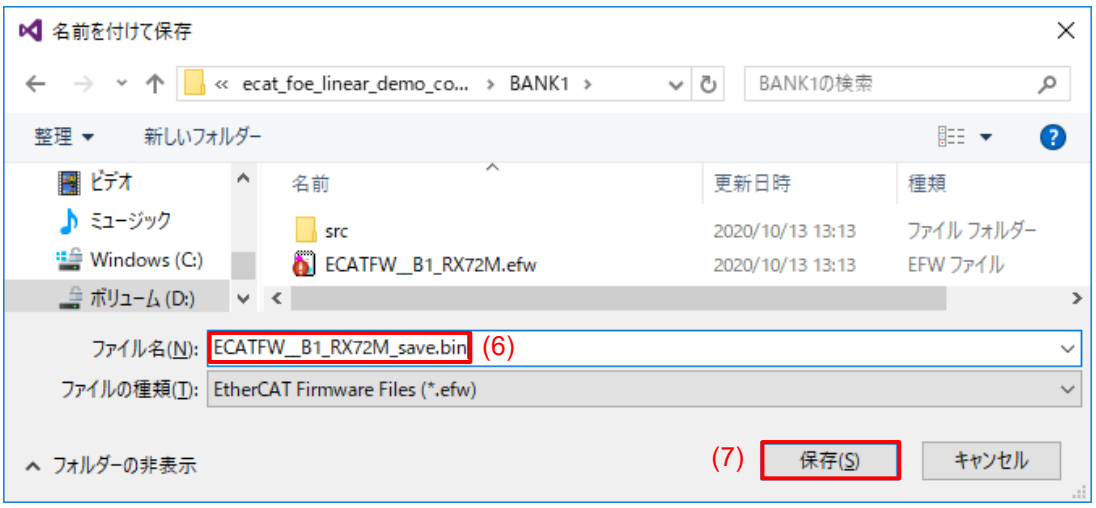
(4) Make sure that the Current State transitions to "BOOT".



(5) Click the "Upload" button of File Access over EtherCAT to open the save file window.

(6) Enter "ECAT__B1_RX72M_save.bin" as the upload file name.

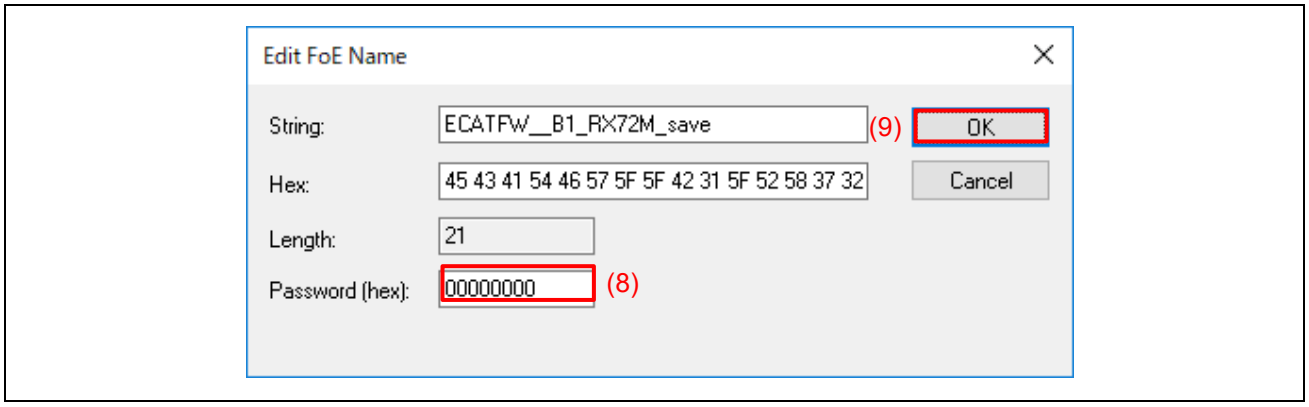
(7) Press "Save".



The file name editing window will open.

(8) Password remains "00000000"

(9) Press "OK".



The upload status is displayed with the message "Uploading" at the bottom left of the screen of TwinCAT System Manager. If no error message is displayed and the upper window disappears and becomes "Ready", the upload is successful.

8. Sample program overview

8.1 Overview of linear mode operation

Linear mode uses code flash in three areas.

Table 8.1 Linear mode code flash area segment

Defined	Features
BANK1	Area to store user program：BANK1 Can be rewritten while running a user program on BANK0
BANK0	Area to store user program：BANK0 Can be rewritten while running a user program on BANK1
Boot Loader	Run the user program by selecting and branching the new BANK user program by comparing Rev No. stored in BANK0 and BANK1 at startup. Not included in user-programmed rewrites

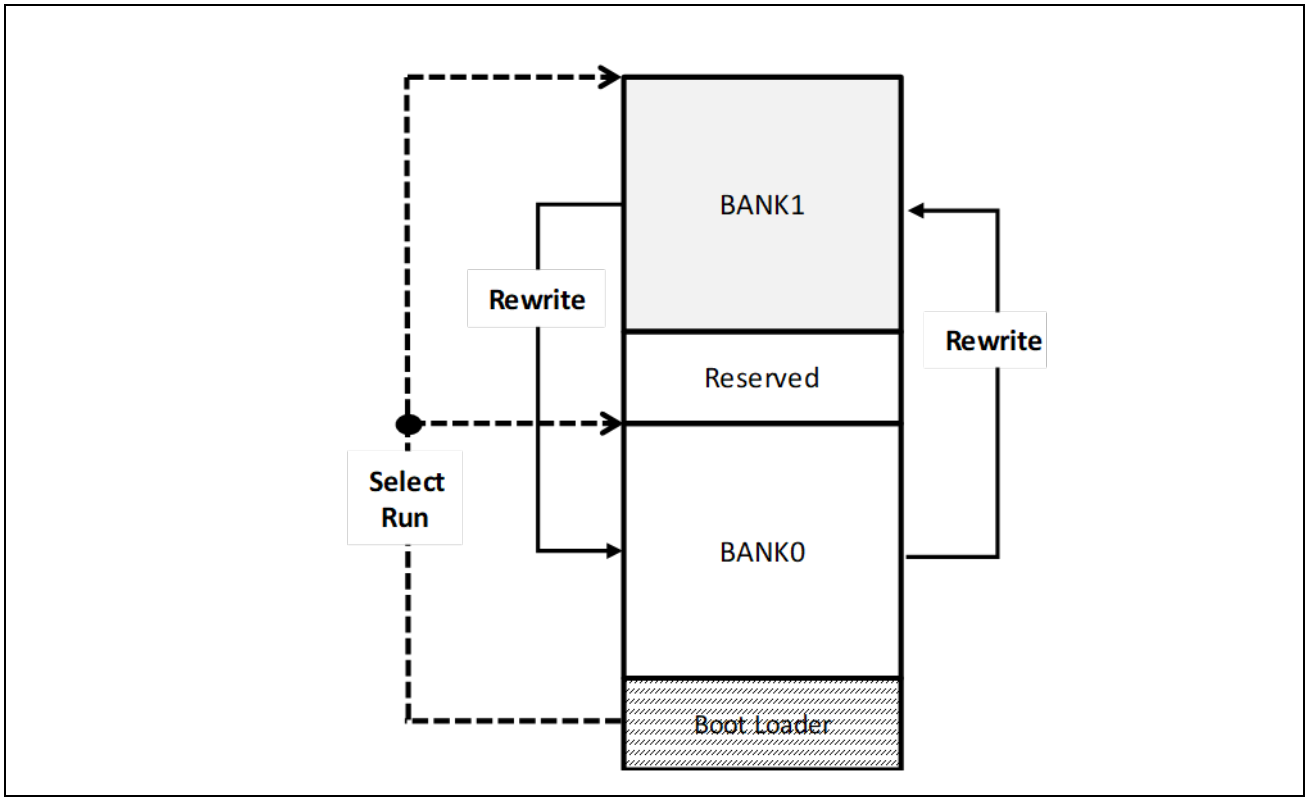


Figure 8.1 Linear mode code flash use image

The relationship between the address and capacity of each region and the block No belonging to each region is shown in the table below.

For 4MB products:

Table 8.2 Address range and capacity of each area and block No (linear 4MB)

Defined	First address	Last address	Capacity	Block No.
BANK1	FFC0_0000H	FFDF_7FFFH	2016KB	133～71 (32KB)
Reserved	FFDF_8000H	FFDF_FFFFH	32KB	70 (32KB)
BANK0	FFE0_0000H	FFFF_7FFFH	2016KB	69～8(32KB) 7～4(8KB)
Boot Loader	FFFF_8000H	FFFF_FFFFH	32KB	3～0(8KB)

For 2MB product

Table 8.3 Address range and capacity of each area and block No (linear 2MB)

Defined	First address	Last address	Capacity	Block No.
BANK1	FFE0_0000H	FFEF_7FFFH	992KB	69～39(32KB)
Reserved	FFEF_8000H	FFEF_FFFFH	32KB	38(32KB)
BANK0	FFF0_0000H	FFFF_7FFFH	992KB	37～8 (32KB) 7～4(8KB)
Boot Loader	FFFF_8000H	FFFF_FFFFH	32KB	3～0(8KB)

Here is a memory block diagram.

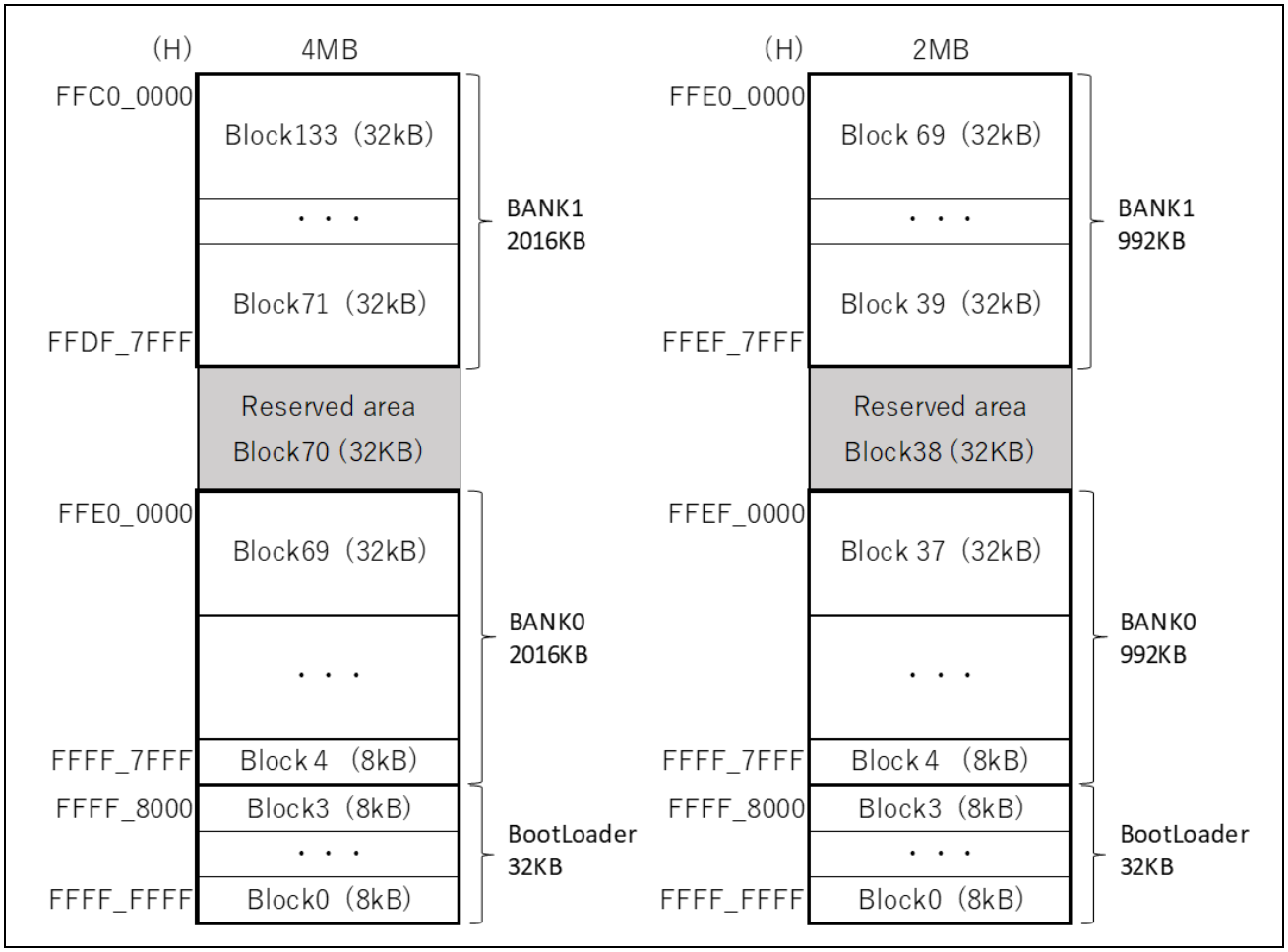


Figure 8.2 Block Assignments in Linear Mode

8.2 Overview of dual mode operation

This section describes the dual mode operation by using the sequence of rewriting a user program from Rev 1.0 to Rev 1.1 and restarting it as an example.

(1) After booting, the Rev1.0 user program stored in the second 2 MB of the code flash area starts. At this time, the value of the startup bank switching bits (BANKSEL.BANKSWP[2:0]) is 111b.

(2) While the Rev1.0 user program is running, erase the first 2MB and rewrite to the Rev1.1 user program.

(3) After flipping the startup bank switching bit (BANKSEL.BANKSWP[2:0]=000b), execute a software reset and restart.

(4) Rev1.1 is replaced by 2MB in the latter half and Rev1.0 is replaced by 2MB in the first half by the startup bank selection function. Rev1.1 user program starts.

Updates from Rev1.1 to Rev1.2 are a similar sequence.

Therefore, dual mode firmware update has the following features.

- Rewriting the code flash is always for the first half 2MB (1MB)
- Banks will be swapped after rebooting, so rewrite with the code that runs in the second half 2MB (1MB)

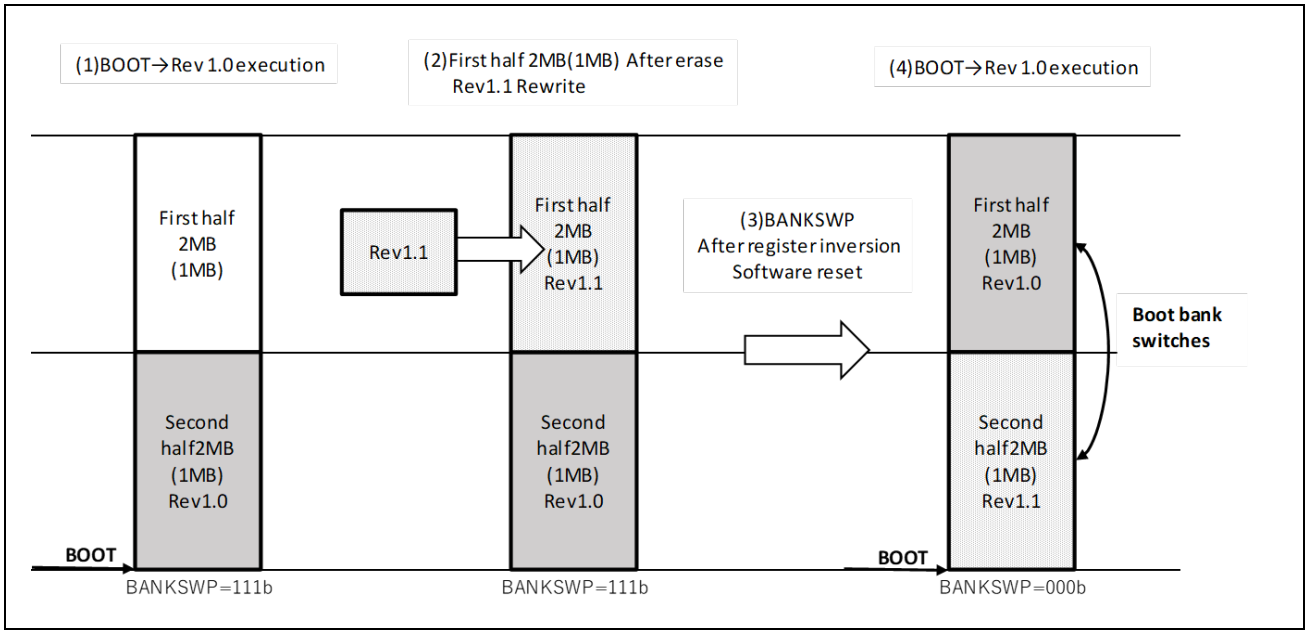


Figure 8.3 Dual-mode sequence of operations

As described in "1.2 Bank Numbers", in this application note, the first half 2MB (1MB) is BANK1 and the second half 2MB (1MB) is BANK0.

The table below shows the relationship between the address and capacity of each area and the block numbers belonging to each area.

In case of 4MB products:

Table 8.4 Address range and capacity for each area and block No (dual 4MB)

Defined	First address	Last address	Capacity	Block No.
BANK1	FFC0_0000H	FFDF_FFFFH	2048KB	139～78 (32KB) 77～70(8KB)
BANK0	FFE0_0000H	FFFF_FFFFH	2048KB	69～8(32KB) 7～0(8KB)

In case of 2MB product

Table 8.5 Address range and capacity for each area and block No (dual 2MB)

Defined	First address	Last address	Capacity	Block No.
BANK1	FFD0_0000H	FFDF_FFFFH	1024KB	107～78(32KB) 77～70(8KB)
BANK0	FFF0_0000H	FFFF_FFFFH	1024KB	37～8 (32KB) 7～0(8KB)

8.3 Bank Info

(1) Bank Erasure

Erase before writing the user program to the bank.

In the linear mode, erase the bank on the opposite side to the bank currently executing the program.

（BANK0→BANK1、BANK1→BANK0）

In dual mode, the eraser will always be BANK1.

(2) Writable Bank Objects

"Firmware Writable Bank" has been added as an object that indicates a writable bank so that the EtherCAT master can confirm which bank the firmware can be written.

If BANK1 is rewritable, the object value is "1", and if BANK0 is rewritable, the object value is "0".

Shows the details of the object.

Table 8.6 Writable Bank Objects

OBJECT Name	INDEX	Category	Access	Data Type	PDO Mapping
Firmware Writable Bank	0x5000	Optional	RO	UINT8	No

(3) Selecting reboots and boot banks

After writing the new firmware to the bank, reboot and execute the new firmware.

The table below shows the reboot method and boot bank selection method.

Table 8.7 Rebooting and Selecting Banks

Item	Linear mode	Dual mode
Reboot method	Branch from the user program to the boot loader	Perform a software reset in the user program.
Start-up bank selection method	The boot loader compares the Rev No. stored in BANK0 and BANK1 and selects the newer BANK user program.	Start-up bank switch bit Selected by the value of (BANKSEL.BANKSWP[2:0]). The bank is switched at the next reset by inverting the value of the startup bank switch bit before rebooting.

8.4 Download file

(1) Download file format

The sample program outputs download files that can be transferred by TwinCAT as build artifacts. Indicates the format of the download file.

Table 8.8 Download file format

Item	Description
File name prefix	Bank 0：”ECATFW__B0” Banks 1：”ECATFW__B1”
File extension	efw
File Formats	Motorola S format

(2) Relationship between download files and banks

The download file is for each bank only and cannot be used for other banks or modes. Shows the relationship between the download file and the bank.

The dual mode download file is for BANK1 but note that the address range is BANK0. This is because it is executed as BANK0 by changing the startup bank after rewriting.

Table 8.9 Relationship between download files and banks

	Linear mode		Dual mode
Download file	ECATFW__B1_xxx.efw	ECATFW__B0_xxx.efw	ECATFW__B1_xxx.efw
Banks to be rewritten	BANK1	BANK0	BANK1
Address range 4MB product	FFC0_0000H~ FFDF_7FFFH	FFE0_0000H~ FFFF_7FFFH	FFE0_0000H~ FFFF_FFFFH
Address range 2MB product	FFE0_0000H~ FFEF_7FFFH	FFF0_0000H~ FFFF_7FFFH	FFF0_0000H~ FFFF_FFFFH
Firmware Writable Bank values to check before downloading	1	0	1
Bank running program at download	BANK0	BANK1	BANK0

(3) Check items when downloading files

Since the firmware is rewritten with the data of the download file, using an invalid file may cause the malfunction.

For this reason, some check items are provided.

It shows the check item and the contents of the check, etc.

Table 8.10 Check items for download files

Item	Check contents	Check timing
File name prefix	String In case of BANK0 : ”ECATFW__B0” In case of BANK1 : ”ECATFW__B1”	Check that the prefix is correct at the start of the download
File password	8 digits Default value：00000000	Check to see if the password matches at the start of the download
Address range	Motorola S Record Format Address Field	Check if the address is within the address range of the bank to be rewritten during the download
Checksum	Checksum field in Motorola S record format	Check if the checksum calculated from the received record matches during the download

(4) Download file writing in linear mode

In linear mode, write the download file to the bank on the side opposite to the one running the program.

The download file for BANK0 cannot be written while the BANK0 user program is running. Similarly, the download file for BANK1 cannot be written while the BANK1 user program is running.

Therefore, the user needs to find out which bank is to be written in advance and prepare an appropriate download file.

The writable bank can be confirmed from the EtherCAT master as the value of the object "Firmware Writable Bank".

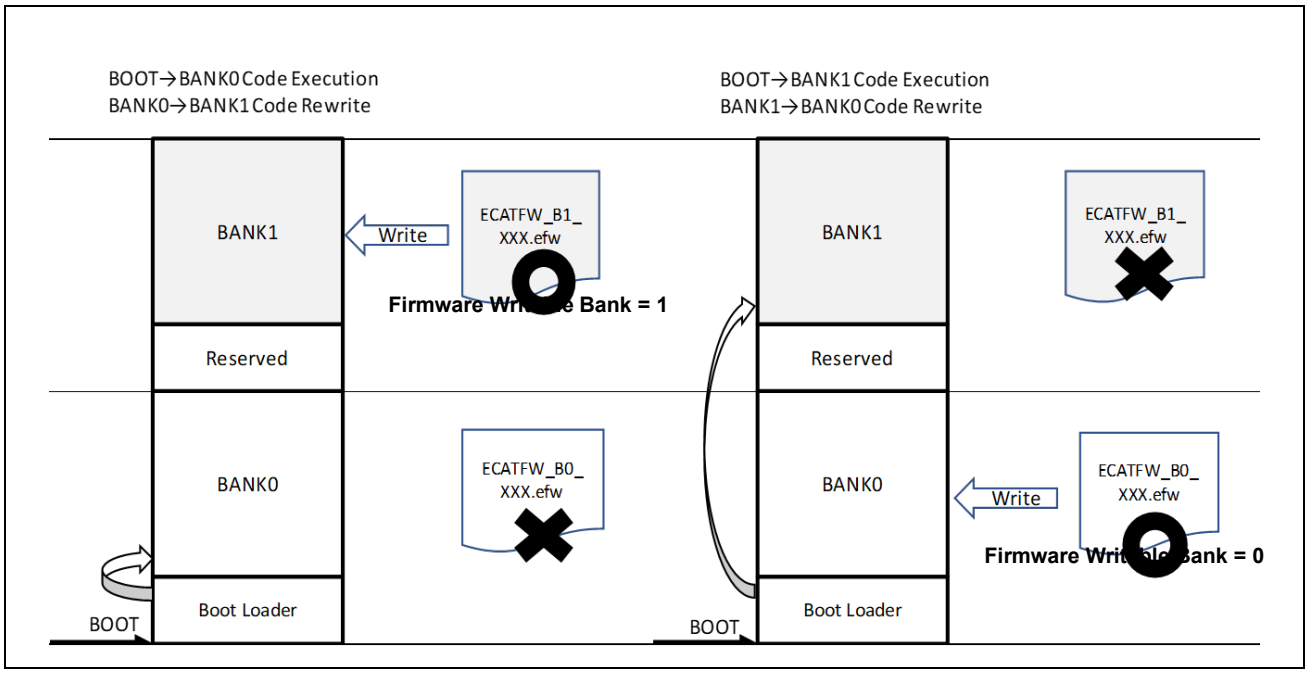


Figure 8.4 Relationship between Banks and Download Files in Linear Mode

(5) Download files in dual mode

In dual mode, the startup bank is swapped after rebooting, so the specifications are as follows.

- Always set the write target bank to BANK1
- Download file uses the one mapped to BANK0

The address included in the Motorola S record format, which is the file format of the download file, indicates BANK0, so when writing to BANK1, write the address as -2MB or -1MB.

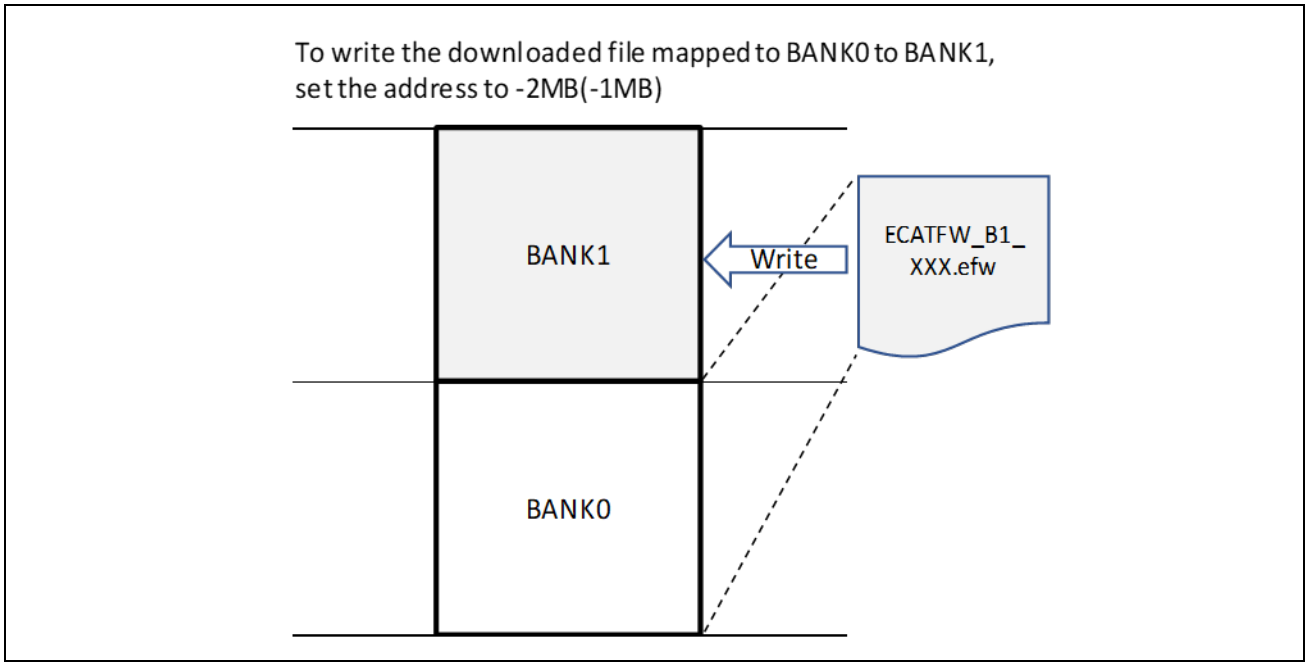


Figure 8.5 Writing Download Files in Dual Mode

8.5 SII EEPROM

(1) SII EEPROM UPDATE

The sample program includes a function to update Revision of SII EEPROM when the firmware is updated.

The contents of the Identify object (Index: 1018) are stored in addresses 16 to 31 bytes of the SII EEPROM.

Shows the structure of the Identify object.

Table 8.11 Configuring the Identity Object

Defined	Contents	Index	Offset address(Byte)
Vendor ID	Vendor ID Defined as “VENDOR_ID” in ecat_def.h	1018:01	0-3
Product code	Product Codes Defined as "PRODUCT_CODE" in ecat_def.h	1018:02	4-7
Revision	Firmware Revision No. Defined as ecat_def.h or the preprocessor macro "REVESION_NUMBER".	1018:03	8-11
Serial number	Serial No. It is defined as "SERAIL_NUMBER" in ecat_def.h.	1018:04	12-15

The contents of the Identify object are assigned to the fixed address of the code flash as an IDENTIFY section.

Indicates the base address of the IDENTIFY section.

Table 8.12 Base address of the IDENTIFY section

Mode	Bank	Base address (H)
Linear mode	BANK0	FFFF_7F80
	BANK1	FFDF_7F80
Dual mode	BANK1	FFFF_7F80

Update the revision of the SII EEPROM to the value of the firmware revision written to the flash after updating the firmware.

(2) Firmware revision check

At the time of INIT->PREOP transition, check whether the revision of firmware being executed and the revision of SII EEPROM match.

If they do not match, an error code will be returned to the EtherCAT master.

AL stratus code (0x0006) AL Status (ERROR INIT)

The issue of error code can be disabled by defining the preprocessor macro "_DISABLE_REVNO_CHECK".

8.6 Firmware update program details

8.6.1 File structure

Table 8.13 Files to be used in the firmware update program

File name	Overview
r_fw_up_rx.c	Firmware Update Source Files
r_fw_up_rx_if.h	Firmware Update Interface File
r_fw_up_rx_private.h	Firmware Update Header File
r_fw_up_buf.c	Buffering source files for firmware data
r_fw_up_buf.h	Buffer data header file for firmware data
r_fw_up_bank.c	Source file of processing that handles information about banks

Table 8.14 Standard include files used in the firmware update program

File name	Overview
stdbool.h	Defines a macro for logical type and logical value
stdint.h	Declare an integer type with a specified width to define a macro
stdlib.h	This library performs standard processing with C programs such as storage area management
string.h	It is a library that compares and copies character strings

8.6.2 List of constants

Table 8.15 Constants to be used in the firmware update program(r_fw_up_rx_if.h)

Constant name	Setting value	Contents
FW_UP_BANK0	(0)	Defines BANK0
FW_UP_BANK1	(1)	Defines BANK1
FW_UP_CFG_PART_MEMORY_SIZE	BSP_CFG_MCU_PART_MEMORY_SIZE	Refer to the value defined in BSP FIT module and define the memory size of CPU
FW_UP_DUAL_BANK_MODE	FLASH_IN_DUAL_BANK_MODE	Refer to the value defined in the flash FIT module to indicate whether it is dual mode. When in dual mode (1)

Table 8.16 Constants to be used in the firmware update program(r_fw_up_rx_private.h)

Constant name	Setting value	Contents
FW_UP_BINARY_BUF_SIZE	(256u)	Buffer size of data for writing to code flash memory
FW_UP_BINARY_BUF_NUM	(2u)	Number of buffers for writing data to code flash memory
FW_UP_BUF_NUM	(60u)	Number of arrays that store the contents of the analyzed Motorola S record format data
FW_UP_BLANK_VALUE	(0xFFFFFFFFu)	Read value when the code flash memory is blank

Table 8.17 Constants to be used in the firmware update program(r_fw_up_buf.h)

Constant name	Setting value	Contents
MOT_S_CHECK_SUM_FIELD	(0x02)	Number of characters in checksum field of Motorola S record format
ADDRESS_LENGTH_S1	(0x04)	Motorola S Record Format Address Field Characters (S1 Type)
ADDRESS_LENGTH_S2	(0x06)	Number of characters in the address field of the Motorola S record format (S2 type)
ADDRESS_LENGTH_S3	(0x08)	Number of characters in the address field of the Motorola S record format (S3 type)
BUF_LOCK	(1)	The specified Motorola S record format buffer is locked.
BUF_UNLOCK	(0)	The specified Motorola S record format buffer is free.

8.6.3 Type definition list

```
typedef enum e_fw_up_return_t
{
    FW_UP_SUCCESS,
    FW_UP_ERR_OPENED,
    FW_UP_ERR_NOT_OPEN,
    FW_UP_ERR_NULL_PTR,
    FW_UP_ERR_INVALID_RECORD,
    FW_UP_ERR_BUF_FULL,
    FW_UP_ERR_BUF_EMPTY,
    FW_UP_ERR_INITIALIZE,
    FW_UP_ERR_ERASE,
    FW_UP_ERR_WRITE,
    FW_UP_ERR_INTERNAL,
} fw_up_return_t;

typedef struct st_fw_up_fl_data_t
{
    uint32_t src_addr;
    uint32_t dst_addr;
    uint32_t len;
    uint16_t count;
} fw_up_fl_data_t;

typedef struct st_fw_up_bank_t
{
    uint32_t low_addr;
    uint32_t high_addr;
    uint32_t start_block;
    uint32_t revno;
    uint32_t blockno;
} fw_up_bank_t;

typedef struct st_fw_up_bankinfo_t
{
    uint16_t active;
    uint16_t writable;
} fw_up_bankinfo_t;
```

Figure 8.6 Type definitions used in the firmware update program(r_fw_up_rx_if.h)

```
typedef enum fw_up_mot_s_cnt_t
{
    STATE_MOT_S_RECORD_MARK = 0,
    STATE_MOT_S_RECORD_TYPE,
    STATE_MOT_S_LENGTH_1,
    STATE_MOT_S_LENGTH_2,
    STATE_MOT_S_ADDRESS,
    STATE_MOT_S_DATA,
    STATE_MOT_S_CHKSUM_1,
    STATE_MOT_S_CHKSUM_2
} fw_up_mot_s_cnt_t;

typedef struct MotSBufS
{
    uint8_t addr_length;
    uint8_t data_length;
    uint8_t *paddress;
    uint8_t *pdata;
    uint8_t type;
    uint8_t act;
    struct MotSBufS *pnext;
} fw_up_mot_s_buf_t;

typedef struct WriteDataS
{
    uint32_t addr;
    uint32_t len;
    uint8_t data[FW_UP_BINARY_BUF_SIZE];
    struct WriteDataS *pnext;
    struct WriteDataS *pprev;
} fw_up_write_data_t;
```

Figure 8.7 Type definitions used in the firmware update program(r_fw_up_buf.h)

8.6.4 Variable list

Table 8.18 Static type variables used in the firmware update program(r_fw_up_rx.c)

Type	Variable name	Contents	Use function
static bool	is_opend	Firmware update initial setting completion flag	fw_up_open fw_up_close write_firmware fw_up_put_data fw_up_get_data

Table 8.19 Static type variables used in the firmware update program(r_fw_up_buf.c)

Type	Variable name	Contents	Use function
static fw_up_mot_s_buf_t	*papp_put_mot_s_buf	Pointer to the Motorola S record data buffer currently used in Motorola S format analysis processing	fw_up_buf_init fw_up_put_mot_s
static fw_up_mot_s_buf_t	*papp_get_mot_s_buf	Pointer to the Motorola S record data buffer currently used in the code flash memory write data creation process	fw_up_buf_init fw_up_get_binary
static fw_up_mot_s_buf_t	mot_s_buf[FW_UP_BUF_NUM]	Buffer to store the contents of Motorola S record format data	fw_up_buf_init fw_up_memory_init
static fw_up_write_data_t	*papp_write_buf	Pointer to the currently used code flash memory write data buffer	fw_up_buf_init fw_up_get_binary
static fw_up_write_data_t	write_buf[FW_UP_BINARY_BUF_NUM]	Buffer that stores data for writing code flash memory	fw_up_buf_init
static fw_up_mot_s_cnt_t	mot_s_data_state	Motorola S record format data analysis status	fw_up_buf_init fw_up_put_mot_s
static uint32_t	write_current_address	Current code flash memory write destination address	fw_up_buf_init fw_up_get_binary
static bool	detect_terminal_flag	End record detection flag	fw_up_buf_init fw_up_put_mot_s fw_up_get_binary

Table 8.20 Variables for bank information used in the firmware update program(r_fw_up_bank.c)

Type	Variable name	Contents	Use function
fw_up_bank_t	Bank[2]	For BANK0 and BANK1, the upper and lower limits of address, starting block address, firmware revision, and total number of blocks are shown	main(Boot loader) fw_up_bank_initial fw_up_check_addr_value fw_up_bank_revno_update BL_Data BL_Reboot
fw_up_bank_t	BankInfo	The bank executing the program and the writable bank are shown	main(User program) erase_another_bank analyze_and_write_data BL_Data

8.6.5 Function list

Table 8.21 Functions used in the firmware update program

Function name	Overview	Listed files
fw_up_open_flash	Flash FIT module initialization	r_fw_up_rx.c
fw_up_open	Firmware update initialization	r_fw_up_rx.c
fw_up_close	Firmware update end process	r_fw_up_rx.c
erase_another_bank	Bank code flash memory erase	r_fw_up_rx.c
analyze_and_write_data	Received data analysis and code flash memory write processing	r_fw_up_rx.c
bank_toggle	Switch startup bank	r_fw_up_rx.c
fw_up_soft_reset	Software reset execution	r_fw_up_rx.c
write_firmware	Code flash memory writing	r_fw_up_rx.c
fw_up_put_data	Received data analysis	r_fw_up_rx.c
fw_up_get_data	Acquisition of code flash memory write data	r_fw_up_rx.c
fw_up_buf_init	Initialization of buffer used for firmware update	r_fw_up_buf.c
fw_up_memory_init	Initialize pointer to buffer	r_fw_up_buf.c
fw_up_put_mot_s	Motorola S record format data analysis	r_fw_up_buf.c
fw_up_get_binary	Acquisition of code flash memory write data	r_fw_up_buf.c
fw_up_ascii_to_hexbyte	Conversion from ASCII format data to binary format data	r_fw_up_buf.c
fw_up_bank_initial	Initial setting of bank information	r_fw_up_bank.c
fw_up_check_addr_value	Check if specified address is within bank range	r_fw_up_bank.c
fw_up_bank_revno_update	Update bank info revision to current value	r_fw_up_bank.c

8.7 FoE Program Details

8.7.1 File structure

Table 8.22 Files used by the FoE program

File Name	Overview
sampleappl.c	Source file of application layer processing including FoE service
sampleappl.h	Header file of application layer processing including FoE service
renesashw.c	Source file for linear mode reboot process
renesashw.h	Header file for linear mode reboot processing
bootmode.c	Source files for firmware update and dual mode reboot process
bootmode.h	Header file for firmware update and dual mode reboot process

Table 8.23 Standard include files used in FoE programs

File Name	Overview
stdio.h	Define functions and macros related to input/output.
stdint.h	Defines a macro by declaring an integer type with the specified width.

8.7.2 List of constants

Table 8.24 Constants used in FoE programs(sampleappl.h)

Constant name	Setting value	Contents
SII_EEP_IDENTIFY_OFFSET	0x08	Start word address of Identify stored in SII EEPROM
SII_EEP_VENDORID	0x00	Offset word address of vendor ID stored in SII EEPROM
SII_EEP_PRODUCTCODE	0x02	Offset word address of product code stored in SII EEPROM
SII_EEP_REVESIONNO	0x04	Offset word address of revision stored in SII EEPROM
SII_EEP_SERIALNO	0x04	Offset word address of serial number stored in SII EEPROM

Table 8.25 Constants used in FoE programs(bootmode.c)

Constant name	Setting value	Contents
BL_DATA_STATUS_IDLE	(0)	Indicates that the file data reception processing status is IDLE
BL_DATA_STATUS_ERASE_START	(1)	Indicates that the file data reception processing status is the code flash erase start status.
BL_DATA_STATUS_ERASE	(2)	Indicates that the file data reception processing status is the code flash erase completion status.
BL_DATA_STATUS_WRITE	(3)	File data reception processing status indicates that code flash is being written

8.7.3 Type definition list

```
typedef union
{
    UINT32      dword[4];
    UINT16      word[8];
    UINT8       byte[16];
}EEPBUFFER;
```

Figure 8.8 Type definitions used in FoE programs(sampleappl.h)

8.7.4 Variable list

Table 8.26 Static variables used in FoE programs(bootmode.c)

Type	Variable name	Contents	Use function
static UINT8	DataStatus	File data reception processing status	BL_StartDownload BL_Data
static BOOL	bReboot	Reboot flag. Set to 1 when doing a reboot	BL_SetRebootFlag BL_CheckRebootFlag

Table 8.27 Cosnt variables used in FoE programs(bootmode.c)

Type	Variable name	Contents	Use function
const UINT32	tbl_identify[4]	(VENDOR_ID), (PRODUCT_CODE), (REVISION_NUMBER), (SERIAL_NUMBER)	AL_ControlInd

Table 8.28 Cosnt variables used in FoE programs(sampleappl.c)

Type	Variable name	Contents	Use function
const UINT16	aFileDownloadHeader [5]	Download file prefix string Bank 0: "ECATFW__B0" Bank 1: "ECATFW__B1"	FoE_Write FoE_Read
const UINT32	aFilePassword	Download file password 8 digits Initial value: 0x00000000	FoE_Write FoE_Read

8.7.5 Function list

Table 8.29 Functions used in FoE programs

Function Name	Overview	Listed files
BL_Start	Processing executed at INIT→BOOT transition	bootmode.c
BL_Stop	Processing executed at BOOT→INIT transition	bootmode.c
BL_StartDownload	File download start processing	bootmode.c
BL_Data	File data reception processing Erase and write code flash	bootmode.c
BL_SetRebootFlag	Reboot flag set	bootmode.c
BL_CheckRebootFlag	Reboot flag check	bootmode.c
BL_Reboot	Dual-mode reboot process	bootmode.c
FoE_Read	FoE read request reception process	sampleapl.c
FoE_ReadData	FoE file data reception processing	sampleapl.c
FoE_WriteData	FoE light request reception processing	sampleapl.c
FoE_Write	FoE file data transmission processing	sampleapl.c
HW_Reboot	Linear mode reboot process	renesashw.c

8.8 Common Device Profile [ETG.5003.1]

In case of handling semiconductor devices with EtherCAT, it is necessary to support the device profile specified in the ETG5003 specifications.

The structure of ETG.5003 is as follows.

1. Common Device Profile (CDP) [ETG.5003.1]
2. Firmware update functionality [ETG.5003.2]
3. Specific Device Profile (SDP) [ETG.5003.2xxx]

Common Device Profile (CDP) specifies requirements that apply to all devices described in Specific Device Profile (SDP)

The sample program provides the object dictionary definition equivalent to CDP [ETG.5003.1 Ver1.1.0] Appendix A. For individual addresses defined in CDP, consider the necessity according to the SDP used.

Also, the sample program provides only the framework for defining the object dictionary. Please separately consider and implement the settings and required processing.

The CDP definition is added below.

Table **8.30** CDP changes

File Name	Additions/changes
sampleappl.h	Add CDP definition to ApplicationObjDic[] Add various CDP address definitions and setting values
RX72M EtherCAT FoE.xml	Add various DataType definition and Object definition of CDP

For Common Device Profile Ver1.1.0, please refer to the following ETG.5003.1 standard.

If you have any questions regarding CDP, please contact the ETG Association.

ETG5003.1 standard

ETG5003-1 S (R) V1.1.0

EtherCAT Semiconductor Device Profile

Part1 Common Device Profile

8.9 Object Dictionary

Table 8-31 shows the object dictionaries defined in this sample program.

Objects after Index 0xF000 are object definitions equivalent to CDP [ETG.5003.1 Ver1.1.0] Appendix A.

Table 8-31 Object Dictionaries Defined in this Sample Program

Index	ObjectCode	SI	DataType	Access	PDO	Name
0x1000	VAR	-	UDINT	RO	-	Device Type
0x1001	VAR	-	USINT	RO	-	Error Register
0x1008	VAR	-	STRING (18)	RO	-	Manufacturer Device name
0x1009	VAR	-	STRING (3)	RO	-	Manufacturer Hardware version
0x100A	VAR	-	STRING (3)	RO	-	Manufacturer Software version
0x100B	VAR	-	STRING (3)	RO	-	Manufacturer Bootloader Version
0x1010	ARRAY	0	USINT	RO	-	Store parameters
		1	UDINT	RW	-	SubIndex 001
0x1011	ARRAY	0	USINT	RO	-	Restore default parameters
		1	UDINT	RW	-	SubIndex 001
0x1018	RECORD	0	USINT	RO	-	Identity Object
		1	UDINT	RO	-	Vender ID
		2	UDINT	RO	-	Product Code
		3	UDINT	RO	-	Revision Number
		4	UDINT	RO	-	Serial Number
0x10F0	RECORD	0	USINT	RO	-	Backup parameter handling
		1	UDINT	RO	-	Checksum
		2	BOOL	RW	T	Backup Parameter Changed
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	-	RxPDO-Map
		1	UDINT	RO	-	SubIndex 001
0x17FF	RECORD	0	USINT	RO	-	Device User RxPDO-Map
		1	UDINT	RO	-	SubIndex 001
0x1A00	RECORD	0	USINT	RO	-	TxPDO-Map
		1	UDINT	RO	-	SubIndex 001
0x1BFF	RECORD	0	USINT	RO	-	Device User TxPDO-Map
		1	UDINT	RO	-	SubIndex 001

Index	ObjectCode	SI	DataType	Access	PDO	Name
0x1C00	ARRAY	0	USINT	RO	-	Sync manager type
		1	USINT	RO	-	SubIndex 001
		2	USINT	RO	-	SubIndex 002
		3	USINT	RO	-	SubIndex 003
		4	USINT	RO	-	SubIndex 004
0x1C12	ARRAY	0	USINT	RO *	-	RxPDO assign
		1	UINT	RO *	-	SubIndex 001
		2	UINT	RO *	-	SubIndex 002
0x1C13	ARRAY	0	USINT	RO *	-	TxPDO assign
		1	UINT	RO *	-	SubIndex 001
		2	UINT	RO *	-	SubIndex 002
0x1C32	RECORD	0	USINT	RO	-	Output SuncManager Paramerter
		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	-	Input SuncManager Paramerter
		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

* It will be RW only in the PreOP state.

Index	ObjectCode	SI	DataType	Access	PDO	Name
0x5000	VAR	-	USINT	RO	-	Firmware Writable Bank
0x6000	VAR	-	UDINT	RO	T	InputCounter
0x7000	VAR	-	UDINT	RW	R	OutputCounter
0XF000	RECORD	0	USINT	RO	-	Semiconductor Device Profile
		1	UINT	RO	-	Index Distance
		2	UINT	RO	-	Maximum Number of Modules
0xF010	ARRAY	0	USINT	RO	-	Module Profile List
		1	UDINT	RO	-	SubIndex 001
0xF020	ARRAY	0	USINT	RO	-	Configured Address List
		1	UDINT	RO	-	SubIndex 001
0xF030	ARRAY	0	USINT	RO	-	Configured Module Ident List
		1	UDINT	RO	-	SubIndex 001
0xF050	ARRAY	0	USINT	RO	-	Detected Module Ident List
		1	UDINT	RO	-	SubIndex 001
0xF380	VAR	-	USINT	RO	T	Active Exception Status
0xF381	ARRAY	0	USINT	RO	-	Active Device Warming Details
		1	UDINT	RO	T	SubIndex 001
0xF382	ARRAY	0	USINT	RO	-	Active Manufacture Warming Details
		1	UDINT	RO	T	SubIndex 001
0xF383	ARRAY	0	USINT	RO	-	Active Device Error Details
		1	UDINT	RO	T	SubIndex 001
0xF384	ARRAY	0	USINT	RO	-	Active Manufacture Error Details
		1	UDINT	RO	T	SubIndex 001
0xF385	RECORD	0	USINT	RO	-	Active Global Device Warming Details
		1	UDINT	RO	T	SubIndex 001
0xF386	RECORD	0	USINT	RO	-	Active Global Manufacture Warming Details
		1	UDINT	RO	T	SubIndex 001
0xF387	RECORD	0	USINT	RO	-	Active Global Device Error Details
		1	UDINT	RO	T	SubIndex 001
0xF388	RECORD	0	USINT	RO	-	Active Global Manufacture Error Details
		1	UDINT	RO	T	SubIndex 001
0xF390	VAR	-	USINT	RO	T	Latched Exception Status
0xF391	ARRAY	0	USINT	RO	-	Latched Device Warming Details
		1	UDINT	RO	T	SubIndex 001
0xF392	ARRAY	0	USINT	RO	-	Latched Manufacture Warming Details
		1	UDINT	RO	T	SubIndex 001

Index	ObjectCode	SI	DataType	Access	PDO	Name
0xF393	ARRAY	0	USINT	RO	-	Latched Device Error Details
		1	UDINT	RO	T	SubIndex 001
0xF394	ARRAY	0	USINT	RO	-	Latched Manufacture Error Details
		1	UDINT	RO	T	SubIndex 001
0xF395	RECORD	0	USINT	RO	-	Latched Global Device Warming Details
		1	UDINT	RO	T	SubIndex 001
0xF396	RECORD	0	USINT	RO	-	Latched Global Manufacture Warming Details
		1	UDINT	RO	T	SubIndex 001
0xF397	RECORD	0	USINT	RO	-	Latched Global Device Error Details
		1	UDINT	RO	T	SubIndex 001
0xF398	RECORD	0	USINT	RO	-	Latched Global Manufacture Error Details
		1	UDINT	RO	T	SubIndex 001
0xF3A1	ARRAY	0	USINT	RO	-	Device Warming Mask
		1	UDINT	RW	-	SubIndex 001
0xF3A2	ARRAY	0	USINT	RO	-	Manufacture Warming Mask
		1	UDINT	RW	-	SubIndex 001
0xF3A3	ARRAY	0	USINT	RO	-	Device Error Mask
		1	UDINT	RW	-	SubIndex 001
0xF3A4	ARRAY	0	USINT	RO	-	Manufacture Error Mask
		1	UDINT	RW	-	SubIndex 001
0xF3A5	RECORD	0	USINT	RO	-	Global Device Warming Mask
		1	UDINT	RW	-	SubIndex 001
0xF3A6	RECORD	0	USINT	RO	-	Global Manufacture Warming Mask
		1	UDINT	RW	-	SubIndex 001
0xF3A7	RECORD	0	USINT	RO	-	Global Device Error Mask
		1	UDINT	RW	-	SubIndex 001
0xF3A8	RECORD	0	USINT	RO	-	Global Manufacture Error Mask
		1	UDINT	RW	-	SubIndex 001
0xF6F0	ARRAY	0	USINT	RO	-	Input Latch Local Timestamp
		1	UDINT	RO	T	SubIndex 001
0xF6F1	ARRAY	0	USINT	RO	-	Input Latch ESC Timestamp (32-bit)
		1	UDINT	RO	T	SubIndex 001
0xF6F2	ARRAY	0	USINT	RO	-	Input Latch ESC Timestamp (64-bit)
		1	ULINT	RO	T	SubIndex 001
0xF9F0	VAR	-	STRING(8)	RO	-	Manufacturer Serial Number
0xF9F1	ARRAY	0	USINT	RO	-	CDP Function Generation Number
		1	UDINT	RO	-	SubIndex 001

Index	ObjectCode	SI	DataType	Access	PDO	Name
0xF9F2	ARRAY	0	USINT	RO	-	SDP Function Generation Number
		1	UDINT	RO	-	SubIndex 001
0xF9F3	VAR	-	STRING(7)	RO	-	Vendor Name
0xF9F4	ARRAY	0	USINT	RO	-	Semiconductor SDP Device Name
		1	STRING(8)	RO	-	SubIndex 001
0xF9F5	ARRAY	0	USINT	RO	-	Output Identifier
		1	USINT	RW	RT	SubIndex 001
0xF9F6	VAR	-	UDINT	RO	-	Time since power on
0xF9F7	VAR	-	UDINT	RO	-	Total time powered
0xF9F8	VAR	-	UDINT	RO	-	Firmware Update Functional Generation Number
0xF9F9	ARRAY	0	USINT	RO	-	Module Manufacturer Hardware Version
		1	STRING (8)	RO	-	SubIndex 001
0xF9FA	ARRAY	0	USINT	RO	-	Module Manufacturer Software Version
		1	STRING (8)	RO	-	SubIndex 001
0xF9FB	ARRAY	0	USINT	RO	-	Module Manufacturer Serial Number
		1	STRING (8)	RO	-	SubIndex 001
0xFBF0	RECORD	0	USINT	RO	-	Device Reset Command
		1	ARRAY [0..5] OF BYTE	RW	-	Command
		2	USINT	RO	-	Status
		3	ARRAY [0..1] OF BYTE	RO	-	Response
0xFBF1	RECORD	0	USINT	RO	-	Exception Reset Command
		1	ARRAY [0..4] OF BYTE	RW	-	Command
		2	USINT	RO	-	Status
		3	ARRAY [0..1] OF BYTE	RO	-	Response
0xFBF2	RECORD	0	USINT	RO	-	Store Parameters Command
		1	ARRAY [0..3] OF BYTE	RW	-	Command
		2	USINT	RO	-	Status
		3	ARRAY [0..1] OF BYTE	RO	-	Response
0xFBF3	RECORD	0	USINT	RO	-	Calculate Checksum Command
		1	ARRAY [0..3] OF BYTE	RW	-	Command
		2	USINT	RO	-	Status
		3	ARRAY [0..7] OF BYTE	RO	-	Response

Index	ObjectCode	SI	DataType	Access	PDO	Name
0xFBF4	RECORD	0	USINT	RO	-	Load Parameters Command
		1	ARRAY [0..3] OF BYTE	RW	-	Command
		2	USINT	RO	-	Status
		3	ARRAY [0..1] OF BYTE	RO	-	Response

8.10 Semi Test Record [ETG.7000.2-Annex5003-0001]

In this sample program, the program is implemented assuming that it corresponds to the following test items of Semi Test Record ETG.7000.2-Annex5003-0001.

1. Device Reset Command (Standard reset)
2. Dynamic PDO
3. Store Parameters

8.10.1 Device Reset Command (Standard reset)

When a specific value is entered in subindex 1 of the object 0xFBF0, the evaluation board restarts.

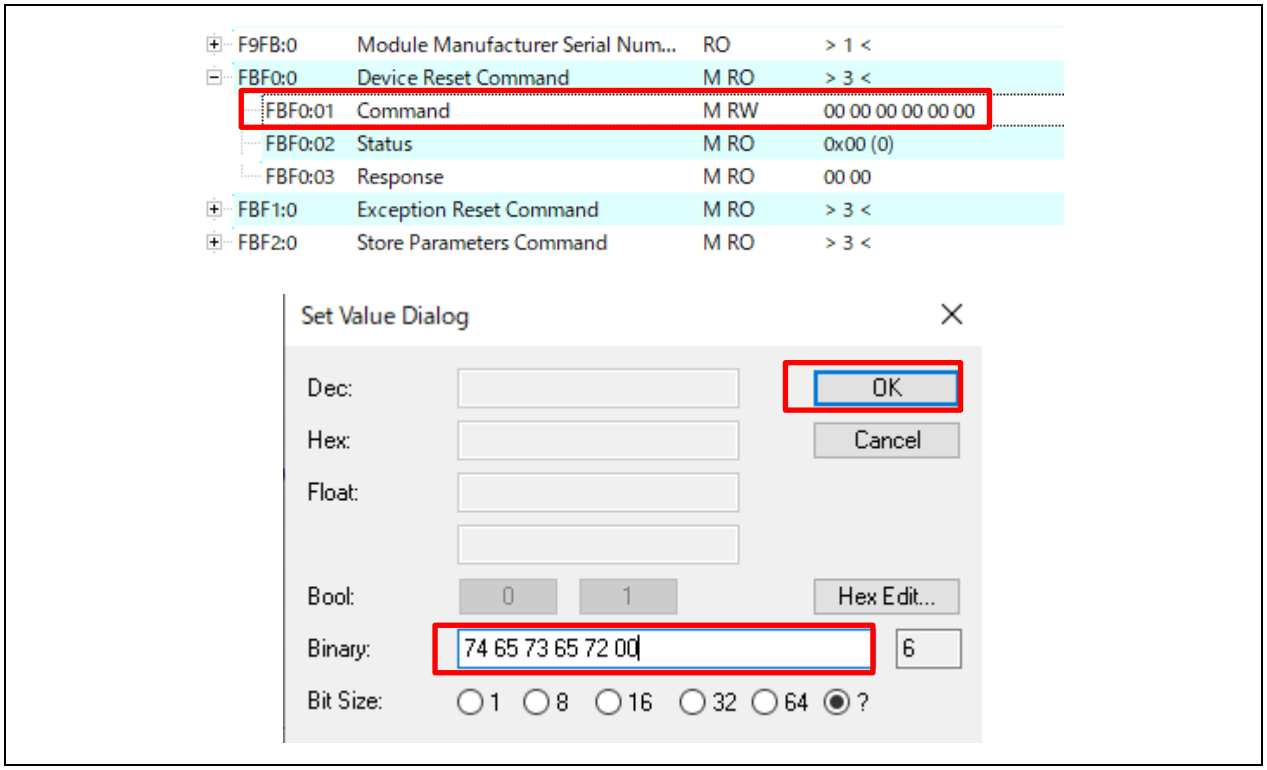
- How to check

(1) Connect the serial port of the evaluation board to the serial port of the PC and start terminal software such as Tera Term on the PC. In the serial settings of the terminal, set 115200 bps, 8-bit data, no parity, 1 stop bit, no flow control.

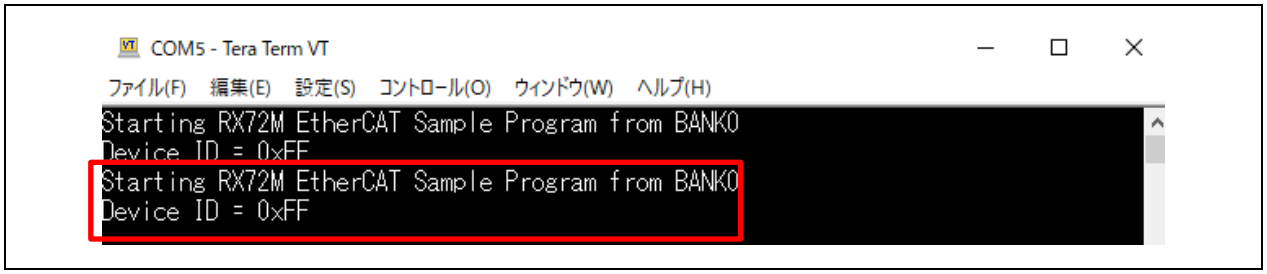
(2) Perform From 6. "Connecting to TwinCAT" to 6.6 "Rescan the device" , Connect the evaluation board to TwinCAT.

(3) Select the "Online" tab and make sure that the "Current Status" is set to "OP".

(4) Select the "CoE - Online" tab, double-click on Index FBF0:01 "Command" and write "74 65 73 65 72 00" in "binary".



(5) If the "Starting RX72M EtherCAT Sample Program from BANKx" output by serial communication is displayed in terminal software such as Tera Term, the evaluation board has been successfully restarted.



8.10.2 Dynamic PDO

In this sample program, the settings related to PDO in the ESI file are shown in Table 8-32.

Table 8-32 PDO Settings for this Sample Program

PDO Items	setting
PdoAssign	true
PdoConfig	true

In addition, the PDO assignment/mapping object settings are shown in Table 8-33.

Table 8-33 PDO Assignment Mapping Object Settings

Object Name	Index	Access
RxPDO assign	0x1C12	RW only in the PreOP state
TxPDO assign	0x1C13	RW only in the PreOP state
Device User RxPDO-Map	0x17FF	RW only in the PreOP state
Device User TxPDO-Map	0x1BFF	RW only in the PreOP state

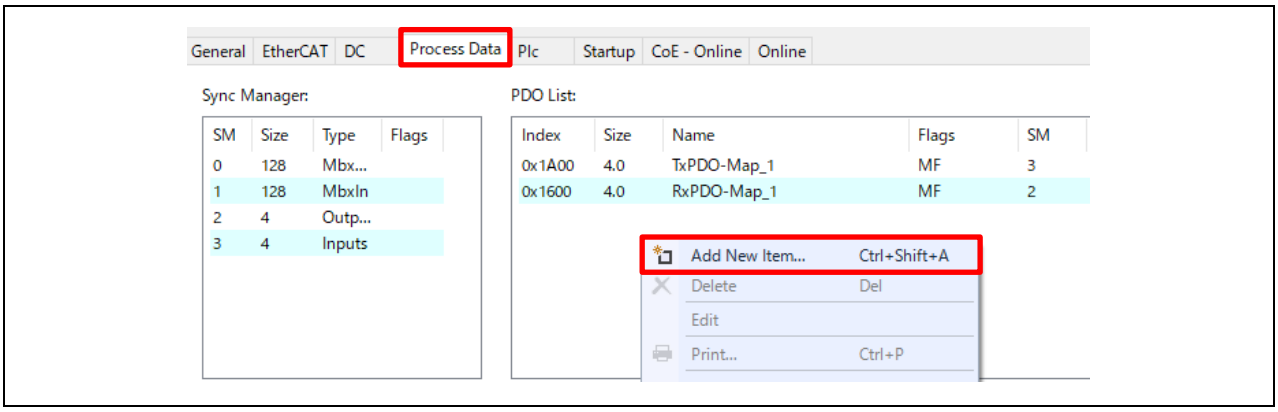
Therefore, objects 0x17FF "Device User RxPDO-Map" and 0x1BFF "Device User TxPDO-Map" that are not assigned by default can be assigned and mapped on the EtherCAT setting tool.

- How to set up

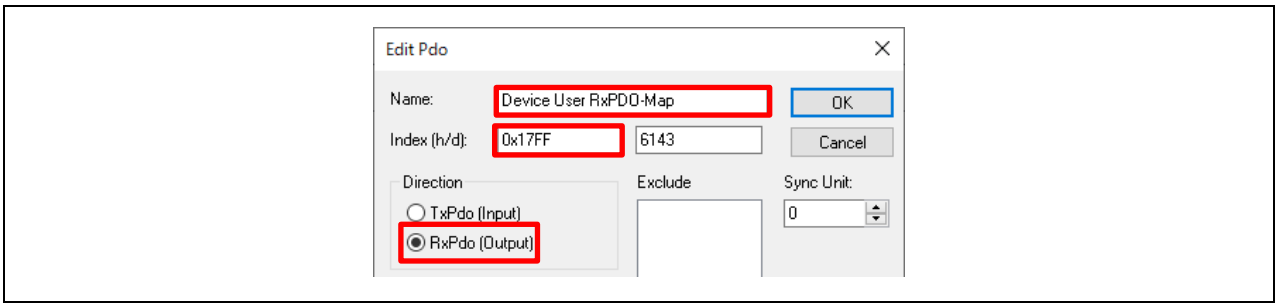
(1) Perform From 6. "Connecting to TwinCAT" to 6.6 "Rescan the device" , Connect the evaluation board to TwinCAT.

(2) Select the "Online" tab and make sure that the "Current Status" is set to "OP".

(3) Select the "Process Data" tab and right-click in the "PDO List" area, select "Add New Item...".



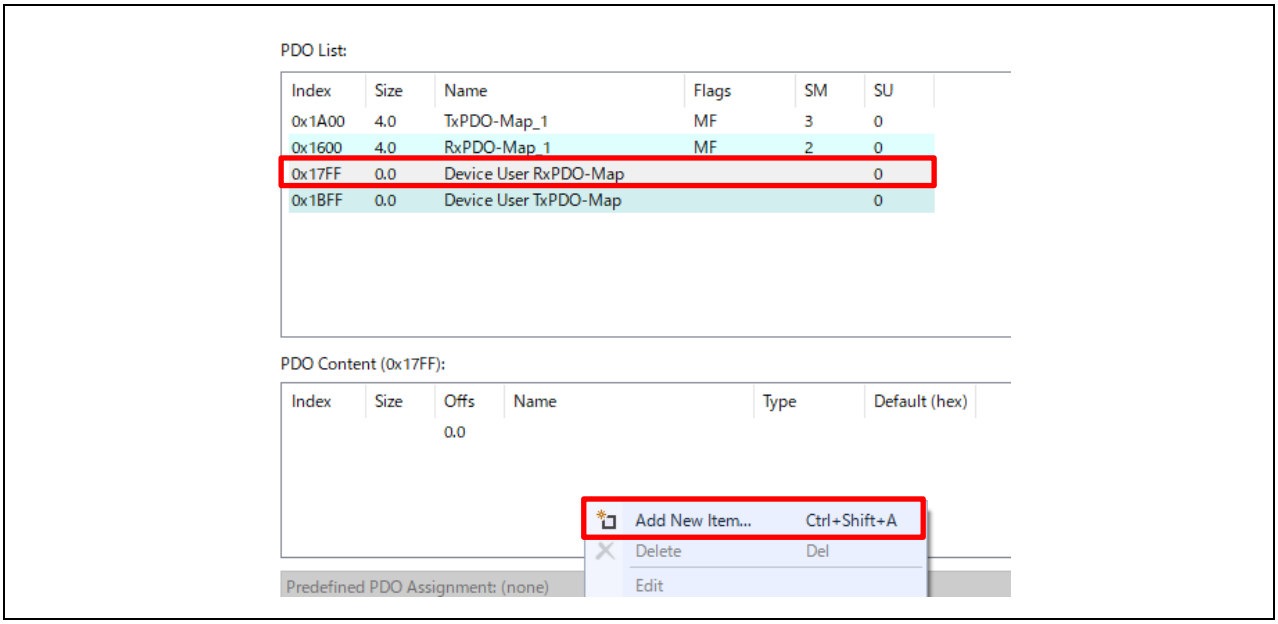
(4) In the "Edit Pdo" window, set the name to "Device User RxPDO-Map", the Index to "0x17FF", the Direction to "RxPdo" and press "OK".



(5) Right-click in the "PDO List" area again, select "Add New Item...", in the "Edit Pdo" window set the name to "Device User TxPDO-Map", the Index to "0x1BFF", the Direction to "TxPdo" and press "OK".

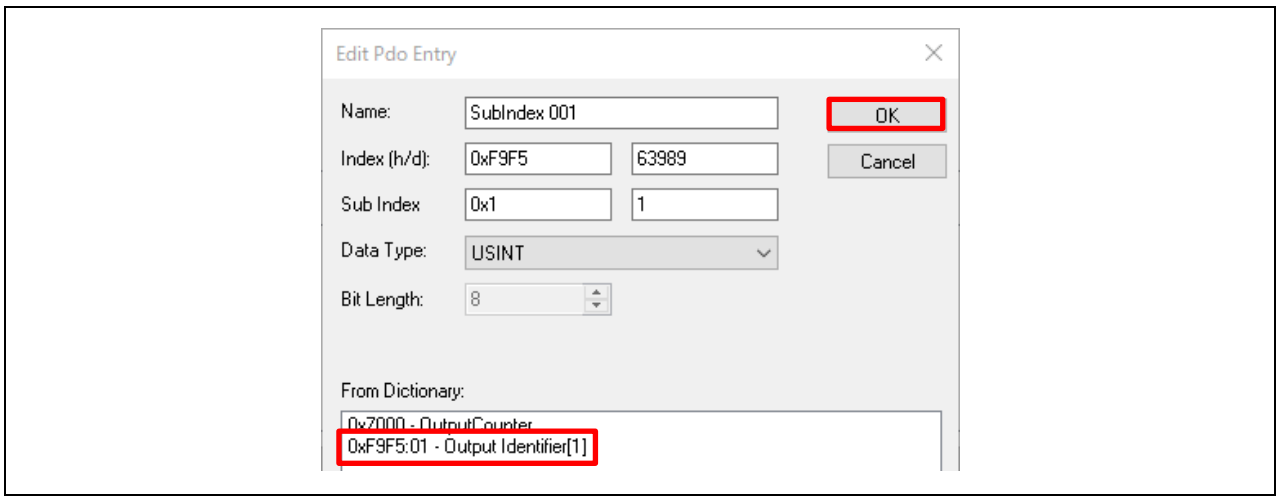
"Device User RxPDO-Map" and "Device User TxPDO-Map" have now been added to the "PDO List".

(6) Click "Device User RxPDO-Map" in the "PDO List" and right-click in the "PDO Content (0x17FF):" area, select "Add New Item...".



(7) In the "Edit Pdo Entry" window, click "0xF9F5:01 – Output Identifier[1]" from "From Dictionary" and press OK.

The Index 0xF9F5 is now mapped to the Index 0x17FF.

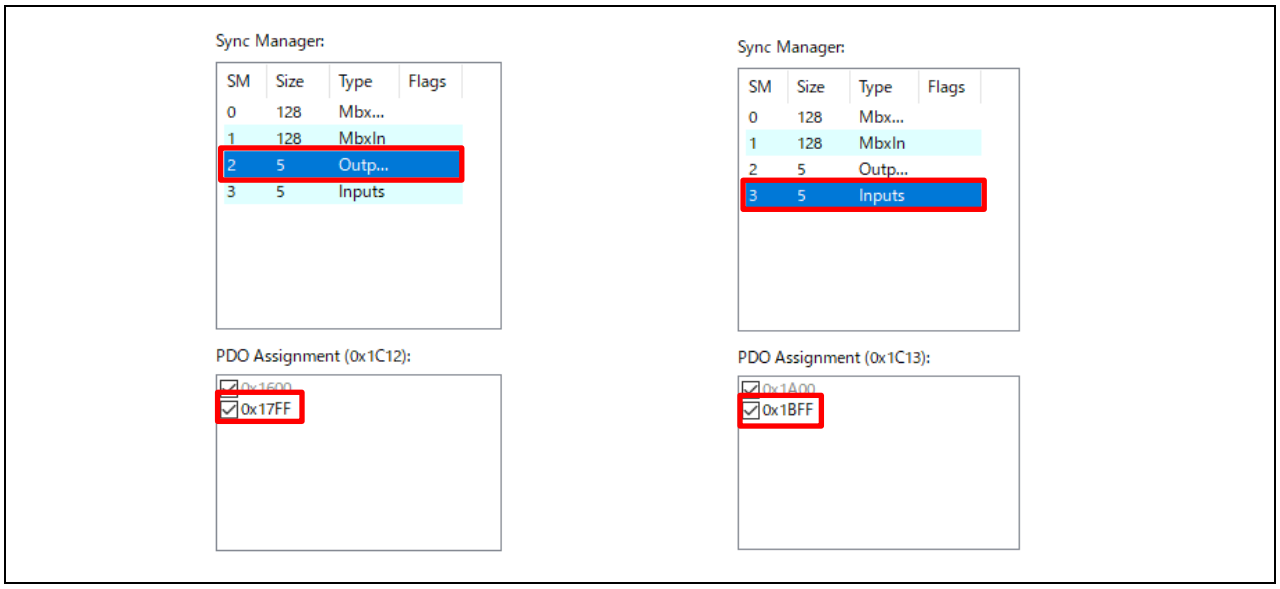


(8) Similarly, click on "Device User TxPDO-Map" in the "PDO List", right-click on the "PDO Content (0x1BFF):" area, select "Add New Item...", and in the "Edit Pdo Entry" window, select "From Dictionary" to "0xF9F5:01 - Output Identifier[1]" and press OK.

The Index 0xF9F5 is now mapped to the Index 0x1BFF.

(9) In the "Sync Manager" area, click on the row with the "SM" column of 2 and check the 0x17FF in the "PDO Assignment (0x1C12)" area.

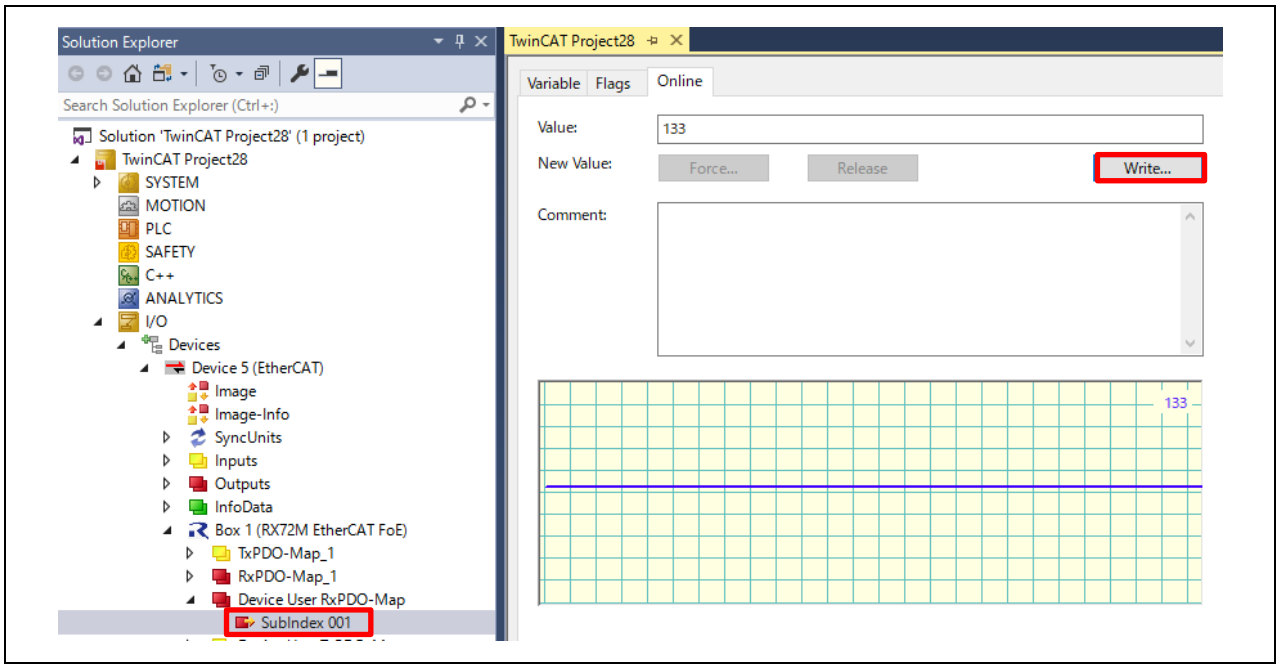
(10) Similarly, in the "Sync Manager" area, click on the row with the "SM" column of 3 and check the 0x1BFF in the "PDO Assignment (0x1C13)" area. You have now assigned an additional 0x17FF to the Index 0x1C12 and an additional 0x1BFF to the 0x1C13.



(11) Restart TwinCAT by pressing [TwinCAT] → [Restart TwinCAT (Config Mode)] in the upper menu bar.

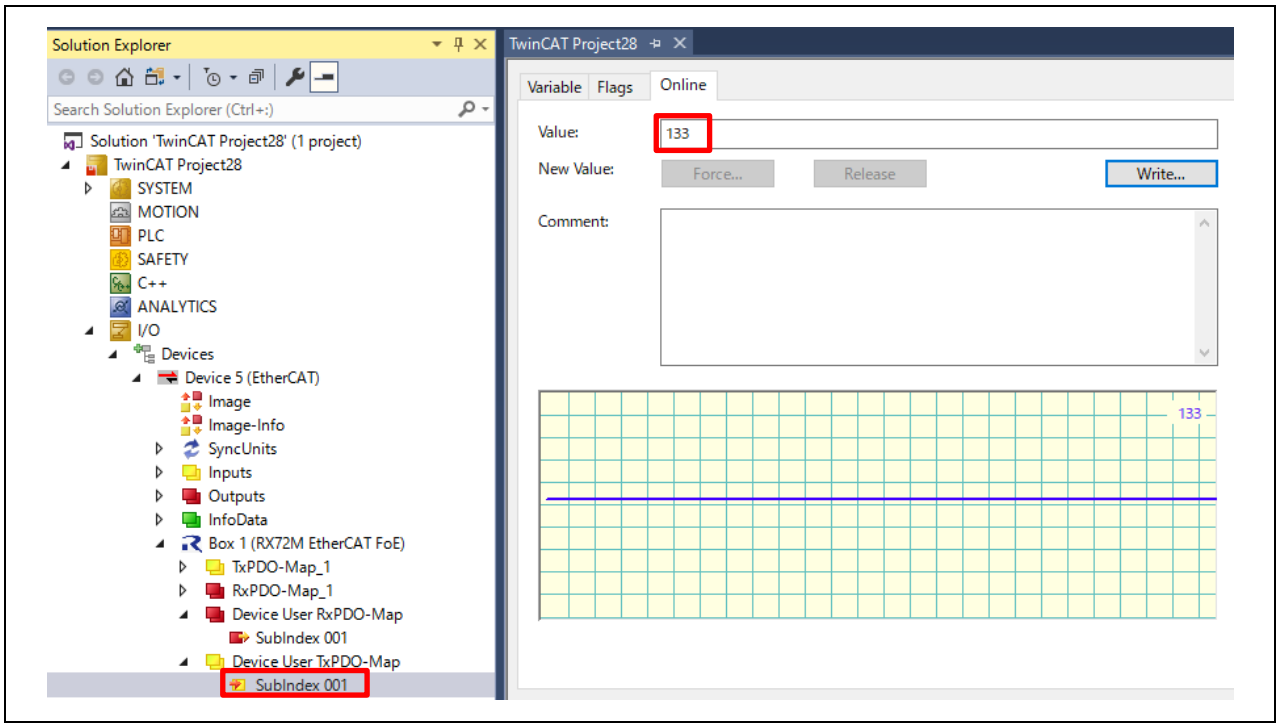
(12) Expand [Device User RxPDO-Map] in [Box 1] and click [SubIndex 001].

(13) Write an arbitrary value of 1~255 in Value.



(14) Expand Device User TxPDO-Map and click SubIndex 001.

(15) If the value of Value is the same as the value written in (13), it is normal.



8.10.3 Store Parameters

In this sample program, when a 0x65766173 is entered in subindex 1 of the object 0xFBF2, the value of the object with the Backup flag is saved in the non-volatile memory of the evaluation board.

Table 8-34 shows the objects with the Backup flag in this sample program.

Table 8-34 Objects with Backup Flags

Index	Name
0x10F0	Backup parameter handling
0xF3A1	Device Warming Mask
0xF3A2	Manufacture Warming Mask
0xF3A3	Device Error Mask
0xF3A4	Manufacture Error Mask
0xF3A5	Global Device Warming Mask
0xF3A6	Global Manufacture Warming Mask
0xF3A7	Global Device Error Mask
0xF3A8	Global Manufacture Error Mask

Table 8-35 shows the code flash memory area used in this sample program as a non-volatile memory area in which the values of objects with the Backup flag are stored.

Table 8-35 Nonvolatile Memory Areas for Backup Objects

Bank mode	First address	Last address	Capacity	Block No.
Linear（4MB）	FFDF_8000H	FFDF_FFFFH	32KB	70 (32KB)
Linear（2MB）	FFEF_8000H	FFEF_FFFFH	32KB	38 (32KB)
Dual（4MB）	FFC0_0000H	FFC0_8000H	32KB	139 (32KB)
Dual（2MB）	FFD0_0000H	FFD0_8000H	32KB	107 (32KB)

Table 8-36 lists the constants used by programs that store in nonvolatile memory.

Table 8-36 Constants Used in Programs Storing Nonvolatile Memory (sampleapple.h)

Constant Name	Setting Values	Details
BACKUP_MEMORY_START_ADDRESS	Table 8-34 First Addresses	The beginning address of the non-volatile memory area for the Backup objects.
BACKUP_BYTESIZE	FLASH_CF_MIN_PGM_SIZE	Amount of non-volatile memory area for Backup objects.
Default_Data_Is_Not_Initialized	(0xFFFF)	The non-volatile memory area for the Backup object has not been initialized.
Default_Data_Is_Initialized	(0x5555)	The non-volatile memory area for the Backup object has been initialized.
NonVolatileWordOffset_0x10F0	(0x0001)	Number of offset words from BACKUP_MEMORY_START_ADDRESS.

NonVolatileWordOffset_0xF3A1	(0x0008)	Number of offset words from BACKUP_MEMORY_START_ADDRESS.
NonVolatileWordOffset_0xF3A2	(0x000c)	Number of offset words from BACKUP_MEMORY_START_ADDRESS.
NonVolatileWordOffset_0xF3A3	(0x0010)	Number of offset words from BACKUP_MEMORY_START_ADDRESS.
NonVolatileWordOffset_0xF3A4	(0x0014)	Number of offset words from BACKUP_MEMORY_START_ADDRESS.
NonVolatileWordOffset_0xF3A5	(0x0018)	Number of offset words from BACKUP_MEMORY_START_ADDRESS.
NonVolatileWordOffset_0xF3A6	(0x001c)	Number of offset words from BACKUP_MEMORY_START_ADDRESS.
NonVolatileWordOffset_0xF3A7	(0x0020)	Number of offset words from BACKUP_MEMORY_START_ADDRESS.
NonVolatileWordOffset_0xF3A8	(0x0024)	Number of offset words from BACKUP_MEMORY_START_ADDRESS.

This sample program uses the functions defined in 6.2.4.1 "Backup Parameter Support" of Application Note ET9300 (EtherCAT Slave Stack Code).

For more information, see Application Note ET9300 (EtherCAT Slave Stack Code) and sampleapple.c.

- How to check

(1) Connect the serial port of the evaluation board to the serial port of the PC and start terminal software such as Tera Term on the PC. In the serial settings of the terminal, set 115200 bps, 8-bit data, no parity, 1 stop bit, no flow control.

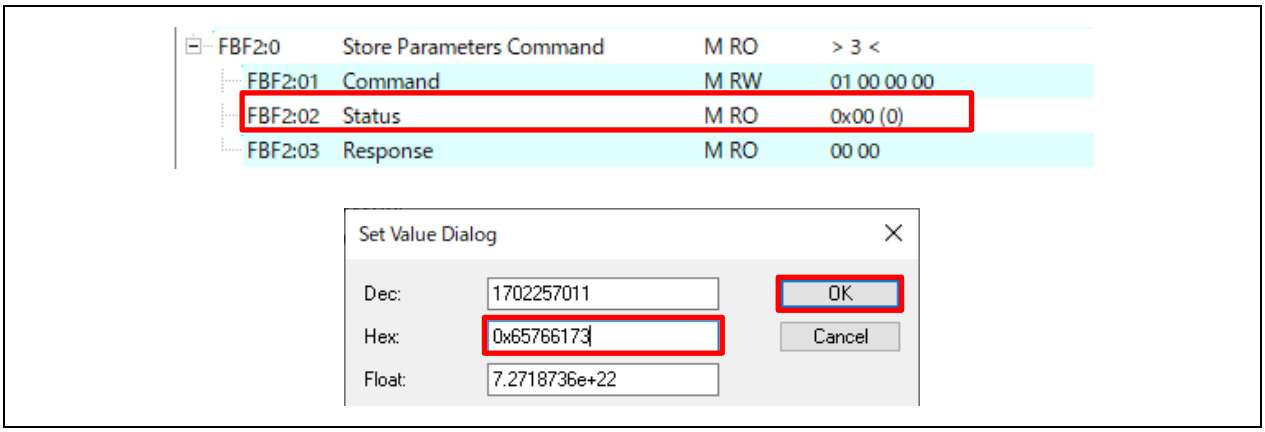
(2) Connect the evaluation board to TwinCAT by completing 6.6 "Rescan the device" in 6. "Connecting to TwinCAT".

(3) Select the "Online" tab and make sure that "Current Status" is set to "OP".

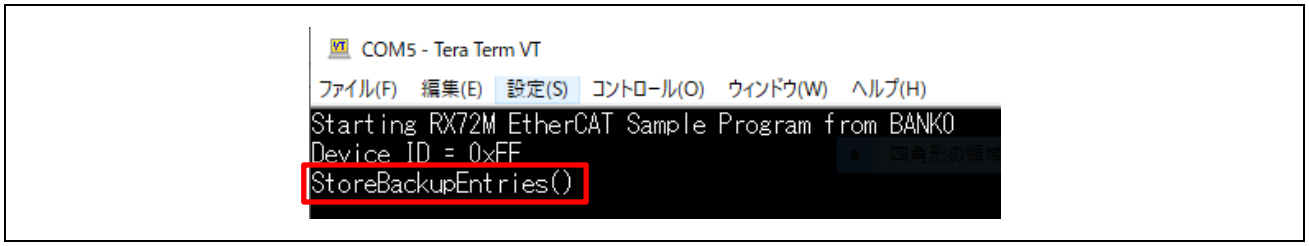
(4) Write the value to the Backup object. Here, we write "0xAAAAAAAA" at Index 0xF3A1:01.

(5) Writes a value to an object that does not have the Backup flag. Here, we write "0x000A" at Index 0x10F1:02.

(6) Select the "CoE - Online" tab, double-click "Command" at Index FBF2:01 and write "0x65766173" in "Hex".



(6) At this time, "StoreBackupEntries()" will be displayed in terminal software such as Tera Term.



(7) Run the 8.10.1 'Device Reset Command' to reboot the evaluation board.

(8) It is normal if the Index 0xF3A1:01 written in (4) is "0xAAAAAAAA".

Objects with the Backup flag are stored in nonvolatile memory and are read at startup.

(9) It is normal if the Index 0x10F1:02 written in (5) is the default value of "0x0004".

Objects that do not have the Backup flag are not stored in non-volatile memory, and the default values are read at startup.

9. Appendices To support code flash memory capacity of 2MB

The project included in this application note is a product with a code flash memory capacity of 4MB.

To support products with 2MB code flash memory, it is necessary to change the configuration file of BSP and some of the section settings of build configuration.

The changes in the BSP configuration file are as follows:

【 src/smc_gen/r_config/r_bsp_config.h 】

Change the value of BSP_CFG_MCU_PART_MEMORY_SIZE to 0xD

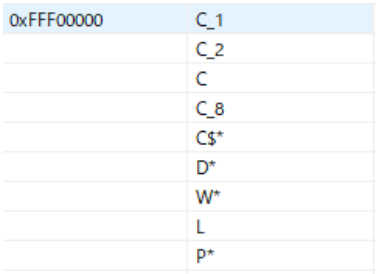
```
#define BSP_CFG_MCU_PART_MEMORY_SIZE        (0xD)
```

Changes to the build configuration section settings are as follows:

To change the settings, go to Project → Properties → C/C++ Build → Settings.

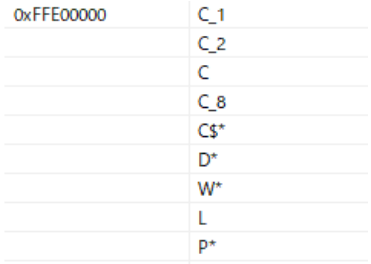
- Linear mode BANK0

Table 9.1 BANK0 Changes - Tools Settings tab

Item	Changes	Description
Linker-section	Change the ROM start position to 0xFFF00000 ＜Settings＞ 	BANK0 mapping is from 0xFFF00000 to 0xFFFF7FFF. The size is 992KB.

- Linear mode BANK1

Table 9.2 BANK1 Changes - Tools Settings tab

Item	Changes	Description
Linker-section	Change the ROM start position to 0xFFF00000 ＜Settings＞ 	BANK1 mapping is from 0xFFE00000 to 0xFFEF7FFF. The size is 992KB.

	Add an IDENTIFY section to the ROM area and set the start position to 0xFFEF7F70. ＜Settings＞ 0xFFEF7F70 IDENTIFY 0xFFEF7F80 EXCEPTVECT 0xFFEF7FFC RESETVECT	

- Dual mode HardwareDebug

Table 9.3 HardwareDebug changes-Tool settings tab

Item	Changes	Description
Linker-section	Changed the starting position of the PResetPRG placed in the ROM area to 0xFFF08000. ＜Settings＞ 0xFFF08000 PResetPRG C_1 C_2 C C_8 C$* D* W* L P PFRAM2	The mapping for BANK0 is 0xFFF00000 to 0xFFFFFFFF but sets the starting position of PResetPRG to 0xFFF08000. The size of BANK0 is 1024 KB.

- Dual mode Download

Table 9.4 Download changes-Tool settings tab

Item	Changes	Description
Linker-section	Changed the starting position of the PResetPRG placed in the ROM area to 0xFFF08000. ＜Settings＞ 0xFFF08000 PResetPRG C_1 C_2 C C_8 C$* D* W* L P PFRAM2	The mapping for BANK0 is 0xFFF00000 to 0xFFFFFFFF but sets the starting position of PResetPRG to 0xFFF08000. The size of BANK0 is 1024 KB.

10. Reference documents

User's Manual: Hardware

RX72M Group User's Manual Hardware Edition（Document No. R01UH0804）
RX72M Group Communication Board Hardware Manual（Document No. R01AN4661）
RX72M CPU Card with RDC-IC User's Manual (Document No. R12UZ0098）
RX72M Renesas Starter Kit+ for RX72M User's Manual (Document No. R20UT4391）
（Please obtain the latest version from the Renesas Electronics website.）

Startup Manual

RX72M Group Communication Board EtherCAT Startup Manual（Document No. R01AN4672）
（Please obtain the latest version from the Renesas Electronics website.）

Application Note：

RX Family EtherCAT Module Firmware Integration Technology（Document No. R01AN4881）

Technical Updates/Technical News

（Please obtain the latest version from the Renesas Electronics website.）

User's Manual：Development environment

RX Family C/C++ Compilers, Assemblers, and Optimized Linkage Editor Compiler Packages
（R20UT0570）
（Please obtain the latest version from the Renesas Electronics website.）

EtherCAT Slave Stack Code Reference material：

Application Note ET9300 (EtherCAT Slave Stack Code)

Version 1.8

ETG5003.1 Standards：

ETG5003-1 S (R) V1.1.0

EtherCAT Semiconductor Device Profile

Part1 Common Device Profile

ETG5003.2 Standards：

ETG5003.2 S(R) V1.0.0

EtherCAT Semiconductor Device Profile

Part2 Firmware Update

ETG7000.-Annex5003-0001 Standards：

ETG7000.-Annex5003-0001 V1.0.1

Semi Test Record

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Aug. 31, 2020	—	First edition issued
1.10	Nov. 20. 2020	34	Changed the title of Chapter 7 from "Firmware update by TwinCAT" to "Operation check by TwinCAT"
		34-38	Changed the description of "7. Firmware update by TwinCAT" to "7.1 Firmware writing"
		39-41	Added "7.2 Firmware readout"
		59	Renamed file prefix string variables in Table 8-28 and added FoE_Read to functions used
1.20	Feb.15.2024	Program	EtherCAT FIT Module Version 1.31
			Add utilities and project folders to separate SSC-related files from project-related files.
			Compatible with e^2 studio 2024-01
			Compatible with SSC 5.13
			Modify the program so that the value of the InputCounter corresponds to the DIP SW of the evaluation board.
			Modified program to support Device Reset Command (Standard reset), Dynamic PDO, and Store Parameters
			Compatible with CPU cards and RSK boards
		5	Updated Table 1-2 "Operating Environments"
		6	Updated Table 1-3 “FIT Module Configurations”
		6	Add CPU Card and RSK Board Project Names to Table 1-4 "Project List"
		10	Updated descriptions and diagrams when importing projects
		11	Added 3.3 "How to install EtherCAT FIT module to e^{2}studio"
		12	Chapter 4 has been renamed to "Project Configuration"
		12	4.1 "EtherCAT FIT module file structure change" has been removed because there is no change in the file structure of the EtherCAT FIT module.
		12-14	4.1 has been renamed to "FIT Module Configuration" and the content has been changed to support the current Smart Configurator.
		15	Added 4.2 "Pin Settings"
		16-30	the descriptions of build and debug configurations moved to 4.3 "Build Configurations" and 4.4 "Debug Configurations"
		16-30	4.3 "Build Configuration" and 4.4 "Debug Configuration" have been modified to match the display of e^{2}studio 2024-01 version.
		31	Added 5.1 "Code Generation"
		32	5.4 "Preparing for Debugging" Added description of RSK board and CPU board
		36	Added 6.2 "Add Ether driver"
		40-41	6.7 "I/O Operation Check" Changed the description so that the value of InputCounter corresponds to the DIP SW of the evaluation board.
		40-41	6.7 "I/O Operation Check" Added Table 6-1 and Table 6-2 to support RSK and CPU boards
		70-79	Added sections 8.9 and 8.10.

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

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

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

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

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

Trademarks

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

Contact information

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