

## RX Family

### EtherCAT Module Using Firmware Integration Technology

---

#### Introduction

This application note describes the EtherCAT® module which uses Firmware Integration Technology (FIT). This module provides an interface for using Beckhoff's EtherCAT slave stack code (SSC) in the RX family with built-in EtherCAT slave controller (ESC) for industrial Ethernet communication.

This module does not include SSC. Obtain SSC tool from EtherCAT Technology Group (ETG Association) and generate SSC.

After this, this module is called EtherCAT FIT module.

#### Target Device

The following is a list of devices that are currently supported by this API:

- RX72M Group

In case of applying this application note to other microcomputers, change them according to the specifications of the microcomputer and fully evaluate them.

**Contents**

1. Overview.....	3
1.1 EtherCAT FIT Module .....	3
2. API Information.....	4
2.1 Hardware Requirements .....	4
2.2 Software Requirements.....	4
2.3 Supported Toolchain .....	4
2.4 Usage of Interrupt Vector .....	4
2.5 Header Files .....	4
2.6 Integer Types.....	5
2.7 Configuration Overview .....	5
2.8 Code size.....	7
2.9 Arguments .....	7
2.10 Return value .....	7
2.11 Callback function .....	8
2.12 Adding the FIT Module to Your Project .....	10
2.13 About for, while, and do while statements.....	11
3. API Functions .....	12
3.1 R_ECAT_Initial.....	12
3.2 R_ECAT_Open.....	13
3.3 R_ECAT_Close .....	14
3.4 R_ECAT_Process .....	15
4. Pin Setting .....	16
5. Demo Projects .....	17
5.1 ecat_demo_rskrx72m.....	17
5.1.1 Board Settings and its Connection.....	17
5.1.2 Demo project operation procedure.....	17
5.2 ecat_demo_comrx72m.....	17
5.2.1 Board Settings and its Connection.....	17
6. Appendix.....	18
6.1 Confirmed Operation environment .....	18
7. Reference Documents.....	22
Revision History .....	23

## 1. Overview

The EtherCAT FIT module provides a means to perform EtherCAT communication using SSC

This module supports the following features:

- Interface for controlling hardware access from SSC to ESC
- PHY initialization
- Start, execution and end processing of EtherCAT slave stack
- ESC interrupt

### 1.1 EtherCAT FIT Module

The EtherCAT FIT module is implemented in a project and used as the API. Refer to 2.12 Adding the FIT Module for details on implementing the module to the project.

## 2. API Information

The API functions of the EtherCAT FIT module adhere to the Renesas API naming standards.

### 2.1 Hardware Requirements

This driver requires your MCU supports the following feature:

- EtherCAT slave controller (ESC)

### 2.2 Software Requirements

FIT modules depend on the following FIT modules.

- Board support package module (r\_bsp) v5.20 or later
- CMT Module (r\_cmt) v3.40 or higher

### 2.3 Supported Toolchain

This FIT module has been confirmed for operation with the tool chain shown in “6.1 Confirmed Operation environment”.

### 2.4 Usage of Interrupt Vector

The ESC interrupts are enabled by calling R\_ECAT\_Open function.

Table 2.1 shows the interrupt vectors used by this FIT module.

**Table 2.1 List of Usage of Interrupt Vectors**

Device	Contents
RX72M	GROUPBL1 interrupt (Vector number: 113) ESCI (EtherCAT interrupt) : bit13 ESC SYNC0 interrupt (Vector number: 252*) ESC SYNC1 interrupt (Vector number: 253*)

※ESC SYNC0 interrupt and ESC SYNC1 interrupt are selective interrupts A and can be assigned to vector numbers 208 to 255. When changing the vector number, modify the definition in r\_ecat\_rx\_private.h accordingly.

### 2.5 Header Files

All API calls and their supporting interface definitions are located in “r\_ecat\_rx\_if.h”.

Build-time configuration options are selected or defined in the file “r\_ecat\_rx\_config.h”.

To reference EtherCAT API elements in this FIT Module from your code include the following:

```
#include “r_ecat_rx_if.h”
```

## 2.6 Integer Types

This software uses ANSI C99. These types are defined in `stdint.h`.

## 2.7 Configuration Overview

The configuration options in the EtherCAT FIT module are specified in `r_ecat_rx_config.h`.

The option names and setting values are listed in the table below.

Configuration option in <code>r_ecat_rx_config.h</code>	
ECAT_CFG_MODE_SEL Note: Default value = 0	This option specifies the connection interface between ESC and PHY. <ul style="list-style-type: none"> <li>0: Specify the MII interface.</li> </ul> Note: Currently fixed at 0 and cannot be changed
ECAT_PHY_OFFSET_ADDRESS Note: Default value = 1	This option defines the value to be set in the PHY address offset setting register (PHYOFF). Set an appropriate value according to your system configuration.
ECAT_CFG_CH0_PHY_ADDRESS Note: Default value = 1 ECAT_CFG_CH1_PHY_ADDRESS Note: Default value = 2	This option defines the PHY-LSI address. Set an appropriate value according to the configuration of your system.
ECAT_CFG_PHYLINK0 ECAT_CFG_PHYLINK1 Note: Default value = 1	This option defines the value to be set for the CATn_LINKSTA pin polarity bit (LINKPOLn). (n = 0,1) <ul style="list-style-type: none"> <li>0: Active high</li> <li>1: Active low</li> </ul> Set an appropriate value according to the configuration of your system.
ECAT_CFG_EEPROM_SIZE Note: Default value = 0	This option defines the value to be set for the EEPROM size setting bit (PROMSIZE). <ul style="list-style-type: none"> <li>0: 16K bits or less</li> <li>1: 32K bits to 4M bits</li> </ul> Set an appropriate value according to the configuration of your system.
ECAT_CFG_ESC_INT_COND Note: Default value = 0	This option defines the value to be set in the ESCI interrupt generation condition setting bit (ESCIC). <ul style="list-style-type: none"> <li>0: An interrupt occurs when PDI_IRQ is "1".</li> <li>1: Interrupt occurs when PDI_IRQ is "0".</li> </ul>
ECAT_CFG_SYNC0_INT_COND ECAT_CFG_SYNC1_INT_COND Note: Default value = 0	This option defines the value to be set in the SYNCn interrupt generation condition setting bit (SYNCnC). (n = 0,1) <ul style="list-style-type: none"> <li>0: An interrupt is generated at the rising edge of the SYNCn signal.</li> <li>1: An interrupt is generated at the falling edge of the SYNCn signal.</li> </ul>

ECAT_CFG_AL1_INT_PRIORITY ECAT_CFG_SYNC0_INT_PRIORITY ECAT_CFG_SYNC1_INT_PRIORITY Note: Default value = 15	This option sets the priority level of the ESCI interrupt or SYNCn interrupt (n = 0,1). The lowest priority level is 1 and the highest is 15.
ECAT_CFG_TX_SHIFT0 ECAT_CFG_TX_SHIFT1 Note: Default value = 0	This option defines the value to be set in the port n transmit signal delay setting bits (TXSFTn [1: 0]). (n = 0,1) <ul style="list-style-type: none"> <li>• 00: 0ns</li> <li>• 01: 10ns</li> <li>• 10: 20ns</li> <li>• 11: 30ns</li> </ul> Set an appropriate value according to the configuration of your system.
ECAT_CFG_PHY_DLAY_RESET Note: Default value = 1000	This option defines the time between PHY reset and access to the PHY registers. (Unit: $\mu$ s) Set an appropriate value according to your system configuration.

#### Configuration option in *r\_ecat\_rx\_config.h*

ECAT_CFG_USE_SUPPORTED_PHY Note: Default value = 0	This option defines whether to use the PHY supported by the FIT module. If a supported PHY is set, the initial setting of the PHY register is not required. <ul style="list-style-type: none"> <li>• 0: do not use supported PHY</li> <li>• 1: use the KSZ8041NL</li> <li>• 2: use the KSZ8081MNX</li> </ul> When using with Renesas Starter Kit + for RX72M, set "1". Set "2" when using the RX72M communication board manufactured by Tessera Technology.
---	---

## 2.8 Code size

Typical code sizes associated with this module are listed below.

The table lists reference values when the C compiler's compile options are set to their default values, as described in 2.3, Supported Toolchain. The compile option default values are optimization level: 2, optimization type: for size, and data endianness: little-endian. The code size varies depending on the C compiler version and compile options.

ROM and RAM Sizes		
Device	Category	Size
RX72M	ROM	1417 bytes
	RAM	92 bytes

## 2.9 Arguments

This section documents the enumerations, unions, and structures used as arguments to API functions. These are included in the `r_ecat_rx_if.h` header file along with the API function prototype declarations.

```
/* EtherCAT When to start the slave stack */
typedef enum
{
    OPEN_BOOT, /* boot */
    OPEN_REBOOT /* reboot */
} ecat_open_t;
```

## 2.10 Return value

This section documents the API function return values. This enumeration type is described in `r_ecat_rx_if.h` along with API function prototype declarations.

```
/* ECAT API error code */
typedef enum
{
    ECAT_SUCCESS = 0, /* Indicates that the function ended normally. */
    ECAT_ERR_INVALID_PTR = -1, /* Pointer argument is NULL or FIT_NO_PTR. */
    ECAT_ERR_INVALID_DATA = -2, /* The argument value is out of range. */
    ECAT_ERR_INVALID_ARG = -3, /* The argument is invalid. */

    ECAT_ERR_EEPROM = -4, /* Indicates that EEPROM loading has failed. */
    ECAT_ERR_OTHER = -5 /* Other error. */
} ecat_return_t;
```

## 2.11 Callback function

In this module, the callback function set by the user is called at the following timing.

- R\_ECAT\_Open function end timing
- R\_ECAT\_Close function end timing
- ESCI interrupt generation timing
- SYNC0 interrupt generation timing
- SYNC1 interrupt generation timing

Indicates the member of the pointer structure for each callback function. The callback function is set by storing the address of the user's function in the described structure member.

```
/* Callback function pointer structure */  
typedef struct  
{  
    void (* pcb_open_func) (void *); /* R_ECAT_Open function end timing */  
    void (* pcb_close_func) (void *); /* R_ECAT_Close function end timing */  
    void (* pcb_intesci_hnd) (void *); /* ESCI interrupt generation timing */  
    void (* pcb_intsync0_hnd) (void *); /* SYNC0 interrupt generation timing */  
    void (* pcb_intsync1_hnd) (void *); /* SYNC1 interrupt generation timing */  
} ecat_cb_t;
```

Indicates the member of the argument structure for each callback function.

When the callback function is called, the variable stored in the described structure member is passed as an argument.

Note: Currently, the arguments are dummy because they are not used. Please correct it if necessary.

```
/* Callback function argument structure */  
typedef struct  
{  
    uint32_t arg1; /* First argument */  
    uint32_t arg2; /* second argument */  
} ecat_cb_arg_t;
```

Since the argument type is passed as a void pointer type, the callback function argument is implemented as follows.

```
/* R_ECAT_Open function end timing callback function example */  
if ((NULL != cb_func.pcb_open_func) && (FIT_NO_FUNC != cb_func.pcb_open_func))  
{  
    cb_arg.arg1 = 0;  
    cb_arg.arg2 = 1;  
    (* cb_func.pcb_open_func) ((void *) & cb_arg);  
}
```



In this module, the callback function is registered in the R\_ECAT\_Initial function without using the API, and is implemented as follows.

```
// --- Initialize the callback function pointer
cb_func.pcb_open_func = NULL;
cb_func.pcb_close_func = NULL;

// --- Initialize the interrupt handler pointer
cb_func.pcb_intesci_hnd = &ecat_esci_inthdr;
cb_func.pcb_intsync0_hnd = &ecat_sync0_inthdr;
cb_func.pcb_intsync1_hnd = &ecat_sync1_inthdr;
```

The R\_ECAT\_Open and R\_ECAT\_Close function callback functions are not registered.  
Add as necessary.

For interrupts, a callback function is implemented to execute the SSC interrupt handler. When adding processing, modify the callback function accordingly.

Call timing	Callback function	SSC function to execute
ESCI interrupt	ecat_esci_inthdr	PDI_Isr function
SYNC0 interrupt	ecat_sync0_inthdr	Sync0_Isr function
SYNC1 interrupt	ecat_sync1_inthdr	Sync1_Isr function

## 2.12 Adding the FIT Module to Your Project

By adding the EtherCAT FIT module to e2 studio, it can be used with Smart Configurator.

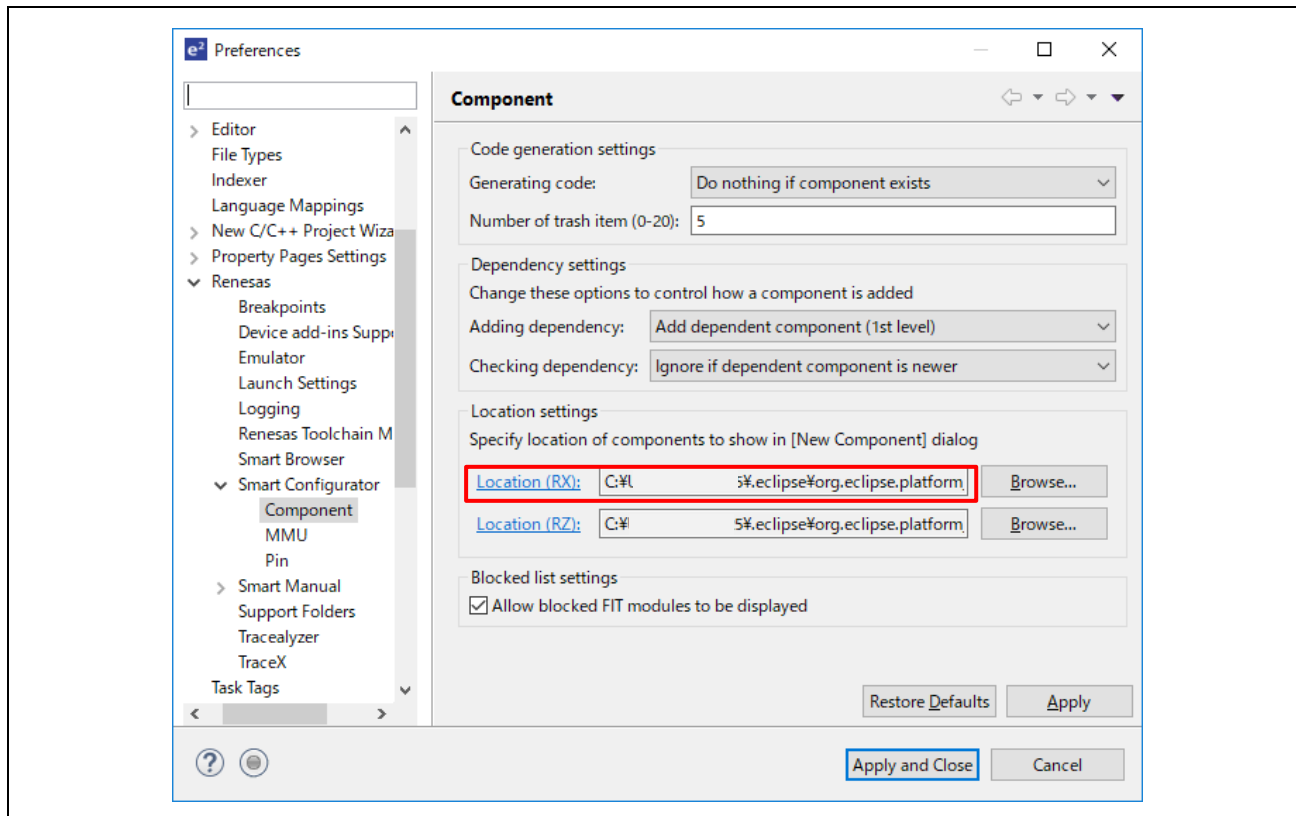
Here is how to add it manually.

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

Check the location of the FIT module in e2 studio.

- [Window]→[Preferences]→The Preferences window opens.
- [C/C++]→[Renesas] →[Smart Configurator]→[Component]

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



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

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

r\_ecat\_rx\_vN.NN.xml

r\_ecat\_rx\_vN.NN.zip

r\_ecat\_rx\_vN.NN\_extend.mdf

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

## 2.13 About for, while, and do while statements

In this module, for statement, while statement, and do while statement (loop processing) are used in the process of waiting for register reflection. In these loop processes, comments with the keyword "WAIT\_LOOP" are described. Therefore, when the user incorporates fail-safe processing into the loop processing, the corresponding processing can be searched with "WAIT\_LOOP".

Description example is shown below.

Example of while statement:

```
/* WAIT_LOOP */  
while (0 == SYSTEM.OSCOVFSR.BIT.PLOVF)  
{  
    /* The delay period needed is to make sure that the PLL has stabilized. */  
}
```

Example for statement:

```
/* Initialize reference counters to 0. */  
/* WAIT_LOOP */  
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)  
{  
    g_protect_counters [i] = 0;  
}
```

Example of do while statement:

```
/* Reset completion waiting */  
do  
{  
    reg = phy_read (ether_channel, PHY_REG_CONTROL);  
    count++;  
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /*  
WAIT_LOOP */
```

### 3. API Functions

---

#### 3.1 R\_ECAT\_Initial

---

This function initializes the EtherCAT module. This function must be executed before using other API functions.

**Format**

void R\_ECAT\_Initial(void)

**Parameters**

None

**Return Values**

None

**Properties**

Prototyped in file "r\_ecat\_rx\_if.h "

**Description**

This function registers the following callback functions.

- R\_ECAT\_Open function end timing
- R\_ECAT\_Close function end timing
- ESCI interrupt generation timing
- SYNC0 interrupt generation timing
- SYNC1 interrupt generation timing

**Example**

```
/* Initialize EtherCAT module */  
R_ECAT_Initial();
```

---

## 3.2 R\_ECAT\_Open

---

This function performs the settings required to start the EtherCAT slave stack. It also enables ESC interrupts. This function must be executed before using the R\_ECAT\_Process function.

### Format

ecat\_return\_t R\_ECAT\_Open(uint32\_t mode)

### Parameters

uint32\_t mode

Specify when to start the EtherCAT slave stack.

OPEN\_BOOT / \* boot \*/  
OPEN\_REBOOT / \* Reboot \*/

### Return Values

[ECAT_SUCCESS]	/ * Indicates that the function ended normally. */
[ECAT_ERR_EEPROM]	/ * Failed to load EEPROM. */
[ECAT_ERR_OTHER]	/ * PHY-LSI initialization failed. */

### Properties

Prototyped in file "r\_ecat\_rx\_if.h"

### Description

This function makes the following settings necessary to start EtherCAT slave stack.

- ESC initialization
- Load EEPROM
- PHY-LSI reset release and PHY register initialization \*
- Enable ESC interrupt
- EtherCAT slave stack initialization

Note: Modify the initial settings of the PHY register according to the PHY-LSI used.

### Example

```
/* Open EtherCAT module */  
R_ECAT_Open(OPEN_BOOT);
```

### 3.3 R\_ECAT\_Close

---

This function performs the settings required to close the EtherCAT slave stack.

**Format**

ecat\_return\_t R\_ECAT\_Close(void)

**Parameters**

None

**Return Values**

[ECAT\_SUCCESS]                    / \* Indicates that the function ended normally. \* /

[ECAT\_ERR\_OTHER]                / \* Other error. \* /

**Properties**

Prototyped in file "r\_ecat\_rx\_if.h"

**Description**

This function makes the following settings necessary to close EtherCAT slave stack.

**Example**

```
/* Close EtherCAT module */
```

```
R_ECAT_Close();
```

### 3.4 R\_ECAT\_Process

---

This function executes the EtherCAT slave stack.

**Format**

ecat\_return\_t R\_ECAT\_Process(void)

**Parameters**

None

**Return Values**

None

**Properties**

[ECAT_SUCCESS]	/ * Indicates that the slave stack has finished normally. * /
[ECAT_ERR_OTHER]	/ * Indicates that the slave stack needs to be terminated. * /

**Description**

The EtherCAT slave stack is executed by SSC's MainLoop function.

**Example**

```
/* Execute EtherCAT Slave Stack */
```

```
R_ECAT_Process();
```

## 4. Pin Setting

To use the EtherCAT FIT module, assign input/output signals of the peripheral function to pins with the multi-function pin controller (MPC). The pin assignment is referred to as the “Pin Setting” in this document. Please perform the pin settings before calling the R\_ECOT\_Start function.

When performing the Pin Setting in the e<sup>2</sup> studio, the Pin Setting feature of the FIT configurator or the Smart Configurator can be used. When using the Pin Setting feature, a source file is generated according to the option selected in the Pin Setting window in the FIT configurator or the Smart Configurator. Pins are configured by calling the function defined in the source file. Refer to Table 4.1 for details.

**Table 4.1 Function Output by the FIT Configurator**

MCU Used	Function to be Output	Remarks
RX72M	R_ECOT_PinSet_ESC	
	R_ECOT_PinSet_ESC_MII0	
	R_ECOT_PinSet_ESC_MII1	



## 5. Demo Projects

EtherCAT demo projects are complete stand-alone programs. They include function main() that utilizes the module and its dependent modules.

---

### 5.1 ecat\_demo\_rskrx72m

---

ecat\_demo\_rskrx72m is a demonstration for performing EtherCAT communication with the Renesas Starter Kit + for RX72M board (hereinafter RSKRX72M board). The LED on the RSKRX72M board can be lit from the EtherCAT master.

The setup method and operation procedure of the demo program are described below.

#### 5.1.1 Board Settings and its Connection

For the details on the board, refer to the Renesas Starter Kit+ for RX72M Board User's Manual (hereinafter RSKRX72M board manual).

For the board connection, refer to “RX72M Group RSK Board EtherCAT Startup Manual” (hereinafter RSKRX72M Startup Manual) for board connection.

#### 5.1.2 Demo project operation procedure

For the board operation, refer to “RSKRX72M Startup Manual” for the operation procedure of the demo project.

---

### 5.2 ecat\_demo\_comrx72m

---

ecat\_demo\_comrx72m is a demonstration for performing EtherCAT communication with the Renesas Starter Kit + for RX72M board (hereinafter RSKRX72M board). The LED on the RSKRX72M board can be lit from the EtherCAT master.

The setup method and operation procedure of the demo program are described below.

#### 5.2.1 Board Settings and its Connection

For the details on the board, refer to the RX72M Group Communication Board Hardware Manual (hereinafter COMRX72M board manual).

For the board connection, refer to “RX72M Group Communication Board EtherCAT Startup Manual” (hereinafter COMRX72M Startup Manual) for board connection.

## 6. Appendix

### 6.1 Confirmed Operation environment

This section describes confirmed operation environment for the EtherCAT FIT module.

**Table 6.1 Confirmed Operation Environment (Rev.1.00)**

Item	Contents
<b>Integrated development environment</b>	Renesas Electronics e2 studio (V7.5.0 or later) IAR Embedded Workbench for Renesas RX (V4.13.1 or later)
<b>C compiler</b>	Renesas Electronics C/C++ Compiler Package for RX Family (V3.01.00 or later) Compiler option: The following option is added to the default settings of the integrated development environment. -lang=C99 GCC for Renesas RX (4.8.4.201803 or later) Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99
<b>Endian</b>	Little endian
<b>Revision of the module</b>	Rev.1.00
<b>Board used</b>	<ul style="list-style-type: none"> <li>• Renesas Starter Kit+ for RX72M (product number: RTK5572MNDCxxxxxBJ)</li> <li>• RX72M communication board for industrial network evaluation by Tessera Technology (Model name: TS-RX72M-COM)</li> </ul>

**Table 6.2 Confirmed Operation Environment (Rev.1.20)**

Item	Contents
<b>Integrated development environment</b>	Renesas Electronics e2 studio (V7.8.0) IAR Embedded Workbench for Renesas RX (V4.20.1)
<b>C compiler</b>	Renesas Electronics C/C++ Compiler Package for RX Family (V3.02.00) Compiler option: The following option is added to the default settings of the integrated development environment. -lang=C99 GCC for Renesas RX (8.3.0.202004) Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99
<b>Endian</b>	Little endian
<b>Revision of the module</b>	Rev.1.20
<b>Board used</b>	<ul style="list-style-type: none"> <li>• Renesas Starter Kit+ for RX72M (product number: RTK5572MNDCxxxxxBJ)</li> <li>• RX72M communication board for industrial network evaluation by Tessera Technology (Model name: TS-RX72M-COM)</li> </ul>

**Table 6.3 Confirmed Operation Environment (Rev.1.30)**

Item	Contents
<b>Integrated development environment</b>	Renesas Electronics e2 studio 2022-10 IAR Embedded Workbench for Renesas RX (V4.20.1)
<b>C compiler</b>	Renesas Electronics C/C++ Compiler Package for RX Family (V3.04.00) Compiler option: The following option is added to the default settings of the integrated development environment. -lang=C99
	GCC for Renesas RX (8.3.0.202202) Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99
<b>Endian</b>	Little endian
<b>Revision of the module</b>	Rev.1.30
<b>Board used</b>	<ul style="list-style-type: none"> <li>• Renesas Starter Kit+ for RX72M (product number: RTK5572MNDCxxxxxBJ)</li> <li>• RX72M communication board for industrial network evaluation by Tesseract Technology (Model name: TS-RX72M-COM)</li> <li>• RX72M CPU Card with RDC-IC (product number: RTK0EMXDE0C00000BJ)</li> </ul>

## 6.2 How to install EtherCAT FIT module in e2 studio

The EtherCAT FIT module is not included in the RX Driver package, so if you want to use it as a software component of the smart configurator other than the sample project, you need to manually install it in e2 studio.

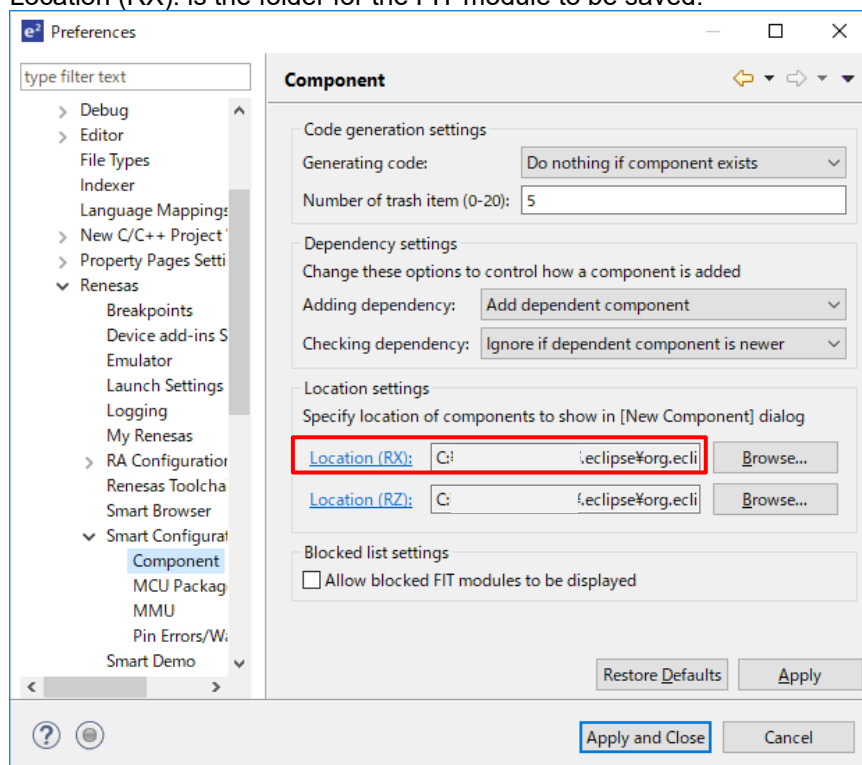
This section describes how to copy the EtherCAT FIT module to the location where the FIT modules downloaded in e2studio.

- The location of FIT modules to be saved

[Window] -> [Preferences] -> The preferences window will open.

[C/C++] -> [Renesas] -> [Smart Configurator] -> [Component]

Location (RX): is the folder for the FIT module to be saved.



- EtherCAT FIT modules

Copy the following three files stored in the FIT Modules folder of the EtherCAT Slave sample program package.

ECAT > Package > an-r01an4881xx0101-rx-ecat > FITModules

名前	更新日時
r_ecat_rx_v1.00.xml	2019/08/26 13:44
r_ecat_rx_v1.00.zip	2019/11/01 13:25
r_ecat_rx_v1.00_extend.mdf	2019/08/22 16:53

Note: Please use the latest version of EtherCAT FIT module.

- Restart e2 studio and make sure the EtherCAT FIT module is registered as a software component of the smart configurator.

The EtherCAT FIT module is included in the EtherCAT Slave sample program package in the RX72M communication board sample program package.

- RX72M communication board sample program package  
r01an4882xxNNNN-rx72m-sample-package.zip
- EtherCAT Slave sample program package  
an-r01an4881xxNNNN-rx-ecat.zip

Note: NNNN is a 4-digit number that represents the revision of each package.

- Precautions when using the smart configurator

Generating the FIT module code in the smart configurator will overwrite the "smc\_gen" folder. Please note that if you save the EtherCAT slave stack code generated by the SSC tool under the "r\_ecat\_rx" folder, it will be overwritten and erased when the codes of the FIT modules are generated. You can avoid overwriting by saving in a folder at the same level as the "smc\_gen" folder like the "src \ application" folder.

## 7. Reference Documents

### User's Manual: Hardware

RX72M Group User's Manual: Hardware (Doc No. R01UH0804)

PS9352AL2 Optically Isolated Delta-Sigma Modulator Data Sheet (Doc No. R08DS0129)

(The latest version can be downloaded from the Renesas Electronics website.)

### Startup manual

RX72M Group RSK Board EtherCAT Startup Manual (Document No. R01AN4689)

RX72M Group Communication Board EtherCAT Startup Manual (Document No. R01AN4672)

(The latest version can be downloaded from the Renesas Electronics website.)

### Technical Update/Technical News

(The latest information can be downloaded from the Renesas Electronics website.)

### User's Manual: Development Tools

RX Family C/C++ Compiler, Assembler, Optimizing Linkage Editor Compiler Package (R20UT0570)

(The latest version can be downloaded from the Renesas Electronics website.)

**Revision History**

Rev.	Date	Description	
		Page	Summary
1.00	Aug.31.19	—	First edition issued
1.01	Nov.01.19	demo project	Add RSK board sample project (ecat_demo_rskrx72m.zip) to FITDemos folder
1.10	Apr.30.20	17	Table 6.1 Confirmed Operation Environment is revised.
		program	Delete “src / ssc” and “src / appl” folders to make EtherCAT slave stack out of FIT module management.
			Changed interrupt handler definition of CATSYNC0 and CATSYNC1 to use BSP module macro. Also corrected that there was an error in the interrupt vector of CATSYNC1.
1.11	Aug.31.20	10	2.12 Adding the FIT Module to Your Project is revised.
		program	Added setting of CLKOUT25M pin by smart configurator.
1.20	Aug.31.21	6	Add a description of ECAT_CFG_USE_SUPOORTED_PHY to 2.7 Configuration option in r_ecat_rx_config.h
		program	Add ECAT_CFG_USE_SUPOORTED_PHY to configuration option.
			Moved PHY register initialization process to PHY.C.
			Moved DIP SW loading process to sampleapp.c.
			Fixed an issue where ECAT_CFG_TX_SHIFT1 was not set to TXSFT1.
1.21	Feb.01.2022	19-20	Add 6.2 How to install EtherCAT FIT module in e2 studio
		program	Changed the xml file of FIT module
1.30	Jan.31.2023	4	Changed the ESC SYNC interrupt vector number in "Table 2.1 List of interrupt vectors to be used" ESC SYNC0: 210→252 ESC SYNC1: 211→253
		18	Delete big endian from the operation confirmation environment
		program	Supports SSC 5.13
			Supports e2 studio 64-bit version
			Delete Utilities folder and below. The files contained in the same folder are provided by the sample program.

# 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](http://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/](http://www.renesas.com/contact/).