RX64M Group

RX Driver Package User’s Manual

R01AN2144EJ0100
Rev.1.00
Sep 1, 2014

Introduction

This application note documents the RX64M Group RX Driver Package User’s Manual, version 1.00.

The RX Driver Package is a software platform that collects middleware and drivers that support the Firmware Integration Technology (FIT). This package provides an environment for conveniently evaluating the rich set of peripheral functions provided by RX microcontrollers.

The RX Driver Package includes device drivers for the peripheral modules included in RX microcontrollers, middleware developed for RX microcontrollers, a variety of interface modules, and the Board Support Package (BSP). Users can freely combine the modules included in the RX Driver Package construct systems simply by creating applications that using those modules, and thus quickly start the evaluation process.

This document describes the RX Driver Package, the basic structures and features of RX Driver Package applications, and the FIT modules included in this application note.

See the related documents for detailed information on using the RX Driver Package and procedures for evaluating applications that use the RX Driver Package.

In the use to the product, to fit your environment, please careful evaluated.

Target Device

RX64M Group (Renesas Starter Kit+ RX64M)

Related Documents

- RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685EU)
- Firmware Integration Technology User’s Manual (R01AN1833EU)
- RX Family Adding Firmware Integration Technology Modules to Projects (R01AN1723EU)
- RX Family Adding Firmware Integration Technology Modules to CubeSuite+ Projects (R01AN1826EJ)
- The User’s Manual provided with the RX Driver Package Application.

Contents

1. Overview

1.1 Applicability

This document applies to the RX64M Group RX Driver Package, version 1.00.

1.2 RX64M Group RX Driver Package

The RX64M Group RX Driver Package (referred to as “this package” below) is a collection of a large number of RX64M device drivers, middleware, and other software. Application programs in a wide range of areas can be implemented using the rich set of peripheral modules provided by the RX64M microcontroller by using this package.

1.3 Operating Environment

This package runs under the operating environment described below.

Table 1.3.1 Operating Environment

Microcontroller	RX64M Group
Evaluation board	Renesas Starter Kit+ RX64M
Integrated development environment (IDE)	e^2 studio, V3.0.1.09 or later Or: CubeSuite+ V2.02.00 or later
Cross tools	RX Family C/C++ Compiler Package V2.01.00 or later
Emulator	E1, E20

2. RX Driver Package

2.1 RX Driver Package Structure

The RX Driver Package is a software platform that collects the lower level software components required to develop application programs. A wide range of applications can be developed using the RX Driver Package.

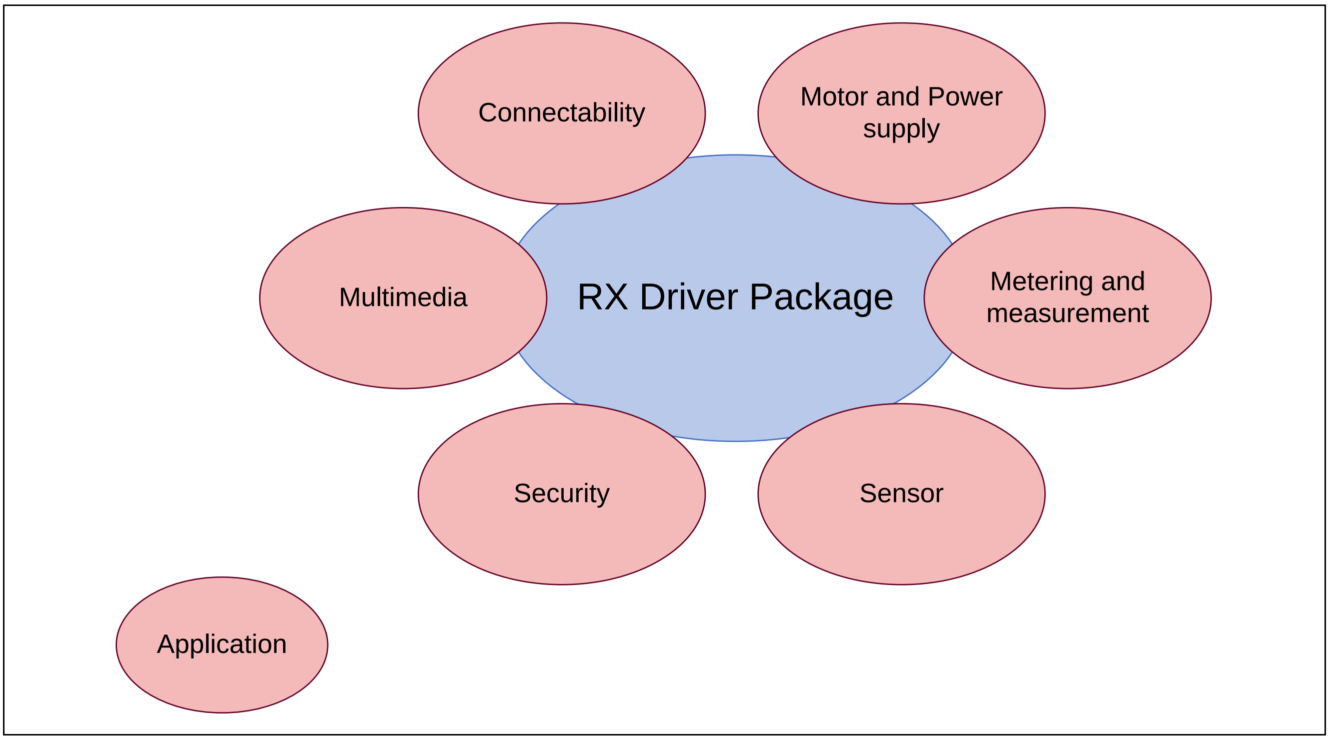
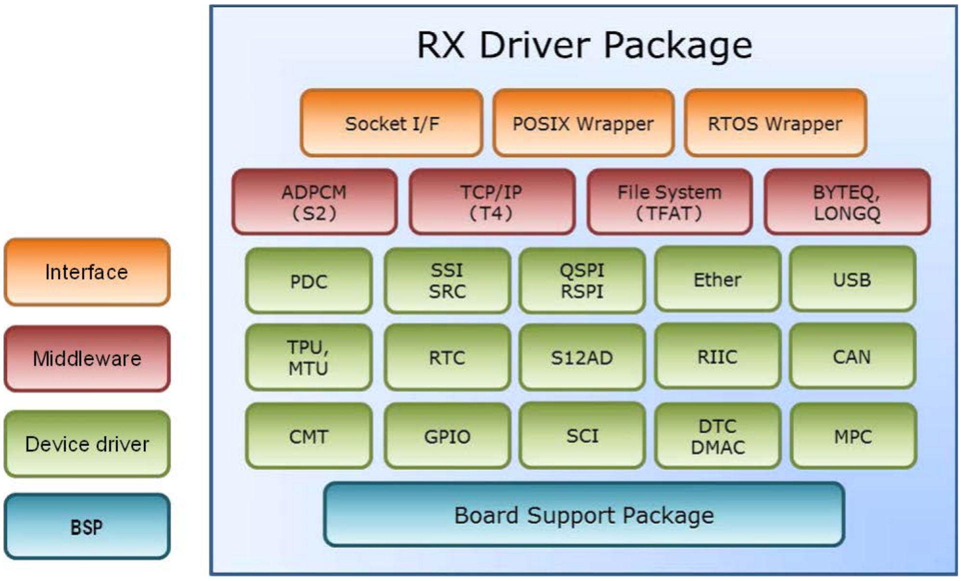


Figure 2.1.1 Types of RX Driver Package Applications

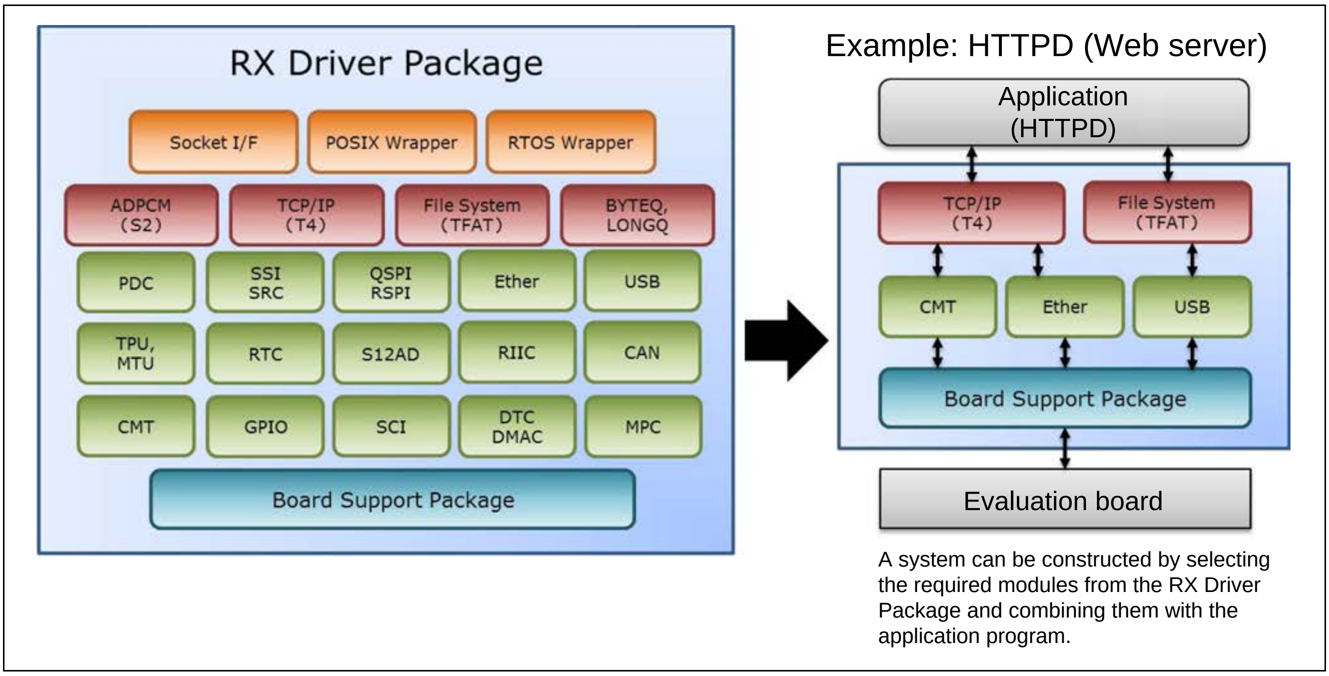
The RX Driver Package consists of the BSP module, microcontroller peripheral module device drivers, middleware (TCP/IP, file systems, and other items), and various interfaces.



Note: Items marked with an asterisk are examples.

Figure 2.1.2 RX Driver Package Structure

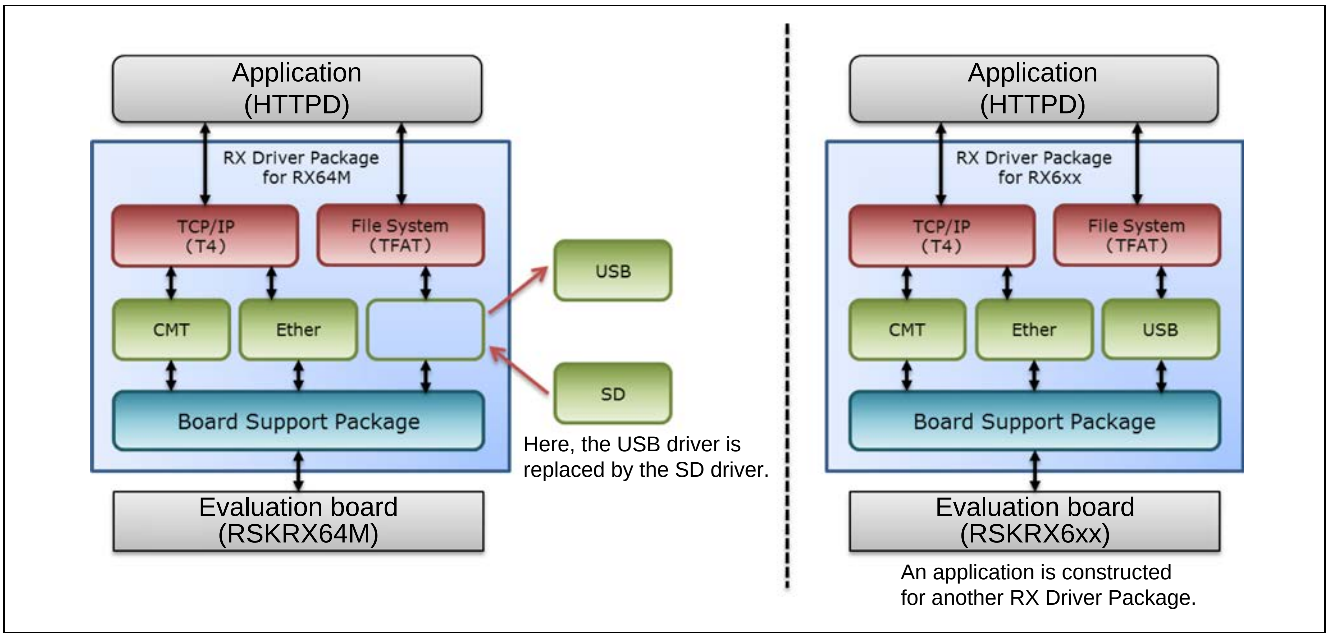
For example, as shown in the figure below, a system can be constructed by collecting the required components from those in the RX Driver Package and providing an application program.



A system can be constructed by selecting the required modules from the RX Driver Package and combining them with the application program.

Figure 2.1.3 System Construction with the RX Driver Package

Furthermore, all of these modules meet the FIT specifications and since the folder structure and API are standardized (with a few exceptions), this structure makes it easy to swap modules in and out or to port an application to another microcontroller.



Here, the USB driver is replaced by the SD driver.

An application is constructed for another RX Driver Package.

Figure 2.1.4 RX Driver Package Application Examples

2.2 RX Driver Package Features

The RX Driver Package has the following features.

- The RX Driver Package is a software platform that provides, in a single package, the device drivers and middleware required for application program development. Renesas will proved RX Driver Packages in a timely as microcontrollers are released.
- Application programs developed on one RX Driver Package can easily be ported to the RX Driver Package for a different microcontroller.
- All the device drivers and middleware included in the RX Driver Package are either compliant with or compatible with the FIT specifications. Thus applications that take advantage of the FIT features can be developed easily.
- The RX Driver Package provides a common interface to middleware and real-time OSes. This makes it easy to reuse software assets and to port applications between real-time OSes.
- Renesas supplies "RX Driver Package Application" sample applications that, when combined with the corresponding RX Driver Package, allow evaluation to be started immediately.
- The RX Driver Package is provided without charge. All the modules in the RX Driver Package can be used without charge, even in commercial applications.

2.3 RX Driver Package Roadmap

While the RX64M is at the head of the list for RX Driver Package provision, Renesas will deploy versions for other microcontrollers as they are developed. Renesas furthermore plans to add new device drives and middleware in the future.

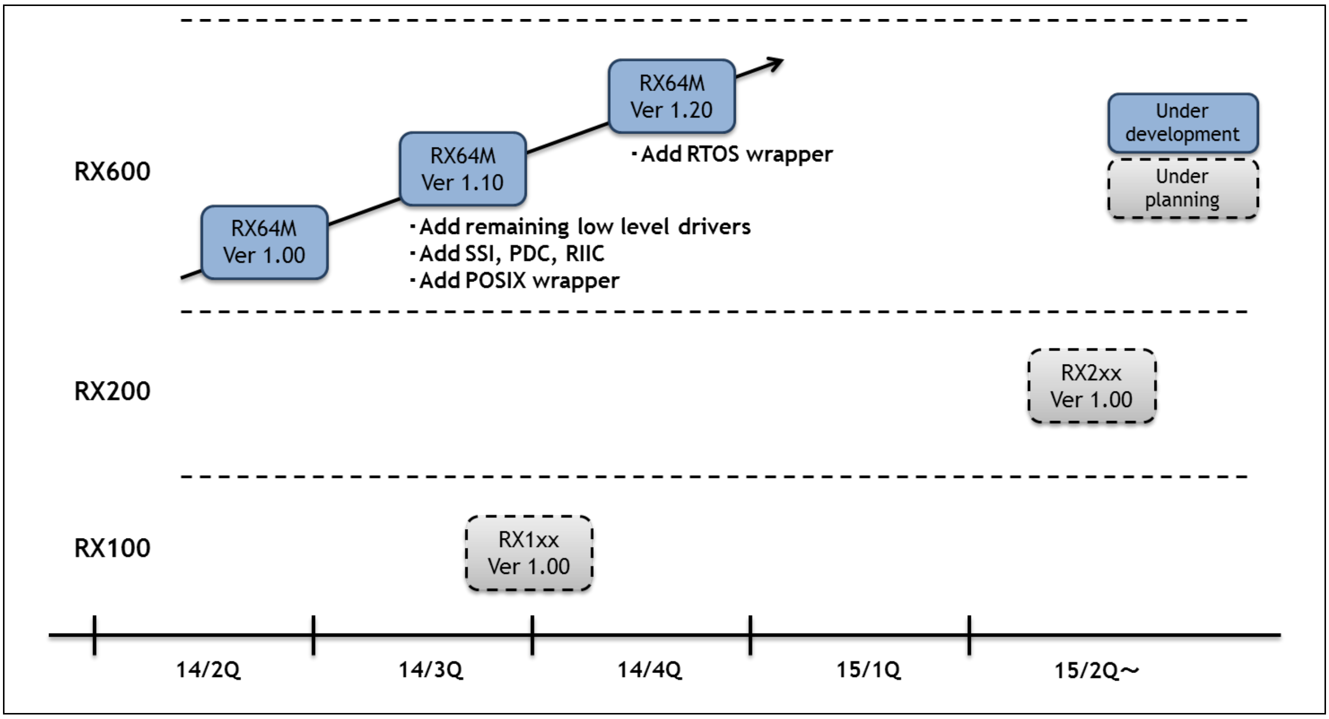


Figure 2.3.1 RX Driver Package Roadmap

3. Structure of the RX64M Group RX Driver Package

3.1 Folder Structure

The folder structure used in this package is shown below.

When the ZIP file for this package is downloaded from the Renesas web site and decompressed, a folder of the same name will be present and it will contain a FITModules folder, a reference_documents folder, and this document.

The FITModules folder contains the FIT modules for the modules shown in table 3.3.1 (as ZIP files and XML files).

The reference_documents folder contains the documentation for using this package in various development environments.

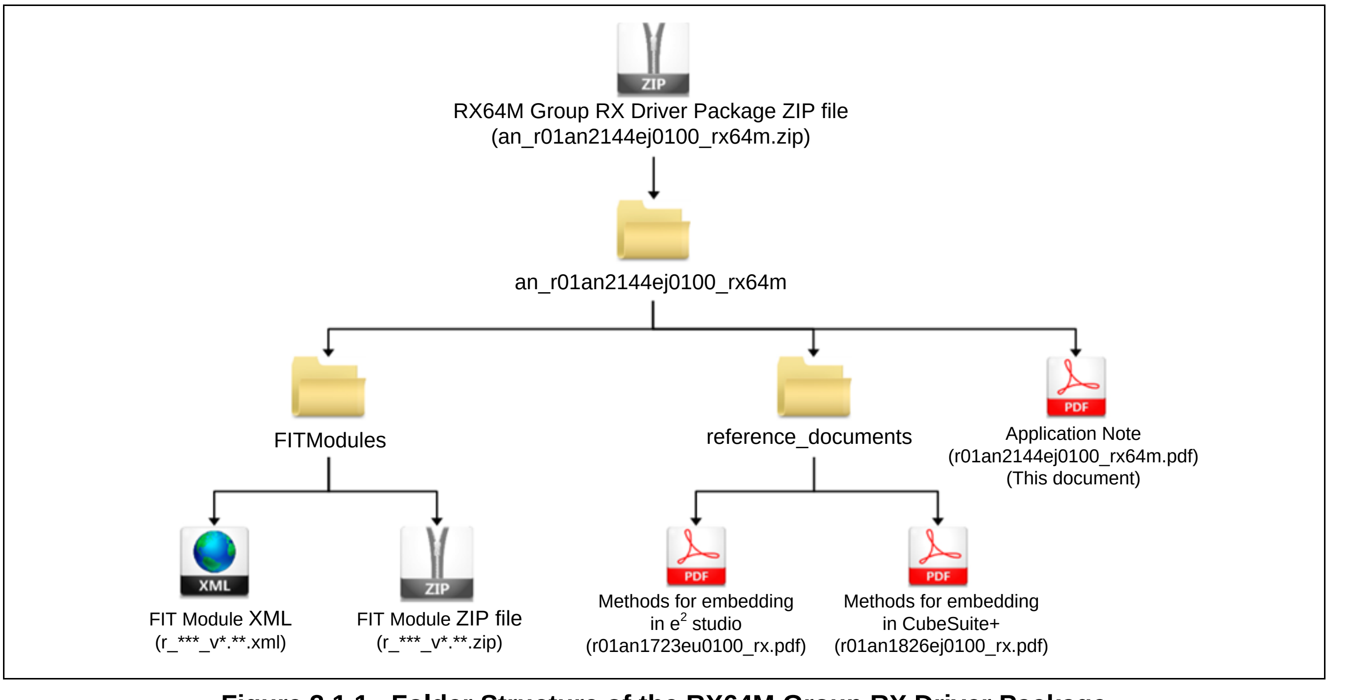


Figure 3.1.1 Folder Structure of the RX64M Group RX Driver Package

3.2 Module Structure

The figure below shows the types and structure of the FIT modules included in this package.

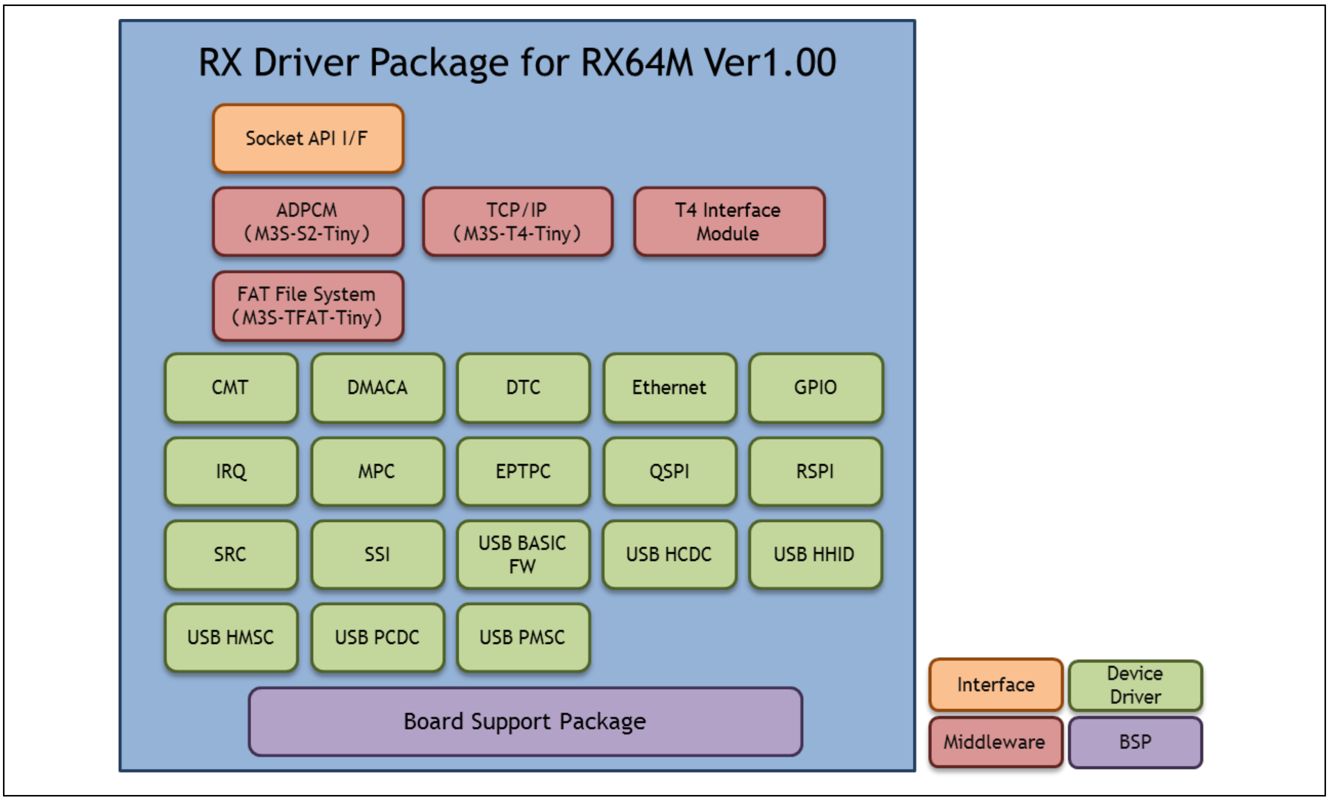


Figure 3.2.1 RX64M Group RX Driver Package FIT Module Structure

3.3 FIT Modules

The table below lists the FIT modules included in this package.

Table 3.3.1 RX64M Group RX Driver Package FIT Modules

Type	Module	FIT Module Name	Version
Board Support Package	Board support package	r_bsp	2.60
Device Driver	Compare match timer (CMT)	r_cmt_rx	2.30
Device Driver	DMA controller (DMACA)	r_dmaca_rx	1.01
Device Driver	Data transfer controller (DTC)	r_dtc_rx	2.00
Device Driver	Ethernet controller (ETHERC)	r_ether_rx	1.00
Device Driver	General-purpose I/O	r_gpio_rx	1.30
Device Driver	Interrupt Controller (IRQ)	r_irq_rx	1.30
Device Driver	Multi-function pin controller (MPC)	r_mpc_rx	1.30
Device Driver	Ethernet controller PTP controller (EPTPC)	r_ptp_api_rx	1.01
Device Driver	Quad serial peripheral interface (QSPI)	r_qspi_smstr_rx	1.06
Device Driver	Serial peripheral interface (RSPI)	r_rspi_smstr_rx	1.06
Middleware	M3S-S2-Tiny (ADPCM encoding/decoding library)	r_s2_rx	3.01
Interface	M3S-T4-Tiny socket API module	r_socket	1.10
Device Driver	Sampling rate converter (SRC)	r_src_api_rx	1.00
Device Driver	Serial Sound Interface (SSI)	r_ssi_api_rx	1.00
Middleware	M3S-T4-Tiny interface conversion module	r_t4_driver_rx64m	1.00
Middleware	M3S-T4-Tiny (TCP/IP protocol stack library)	r_t4_rx	2.00
Middleware	M3S-TFAT-Tiny (FAT file system)	r_tfat_rx	3.00
Device Driver	USB basic firmware	r_usb_basic	1.00
Device Driver	USB host communication device class	r_usb_hcdc	1.00
Device Driver	USB host human interface device class	r_usb_hhid	1.00
Device Driver	USB host mass storage class	r_usb_hmsc	1.00
Device Driver	USB peripheral communication device class	r_usb_pcdc	1.00
Device Driver	USB peripheral mass storage class	r_usb_pmsc	1.00

Note: This package include the M3S-T4-Tiny (TCP/IP protocol stack library) of evaluation version. For the commercial version, please go to the below URL.
http://www.renesas.com/mw/t4

4. Usage Procedures

The RX Driver Package allows programs to be easily constructed by using the FIT plugin included in e^2 studio. The remainder of this section presents a simple usage example using e^2 studio. To use CubeSuite+, see the document “RX Family Adding Firmware Integration Technology Modules to CubeSuite+ Projects (R01AN1826EJ)” included in this package.

4.1 Environment Used

The RX64M is used as the target microcontroller and the Renesas Starter Kit+ RX64M is used as the target board. If a different environment is used, replace the specifics used in the example with the ones for that environment as you read.

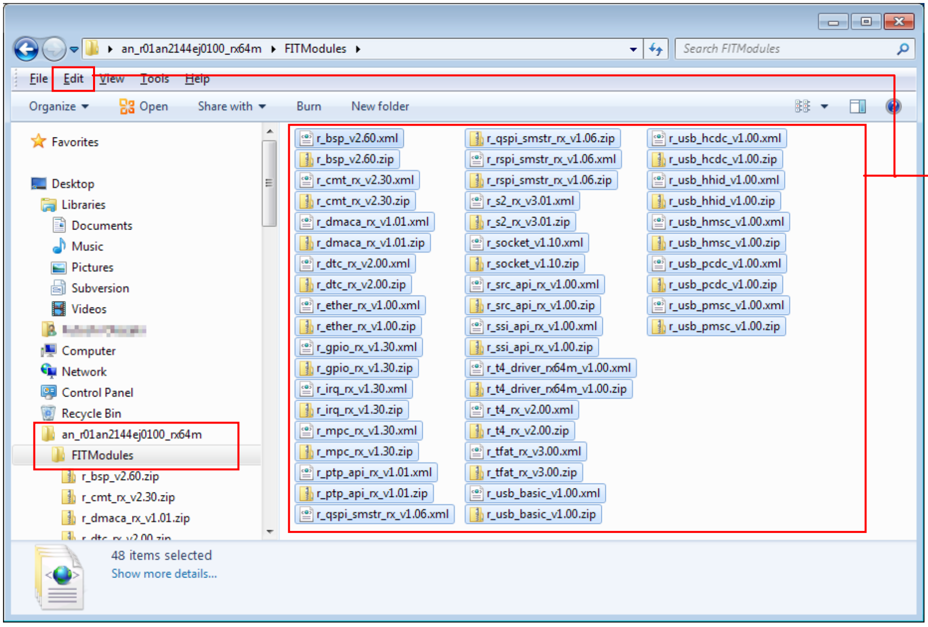
4.2 Install RX Driver Package in e^2 studio

Install the FIT modules in the RX Driver Package into e^2 studio.

1. Decompress the downloaded file **an_r01an2144ej0100_rx64m.zip** into an arbitrary directory.

2. Open the folder that was decompressed and open the **FITModules** folder in that folder.

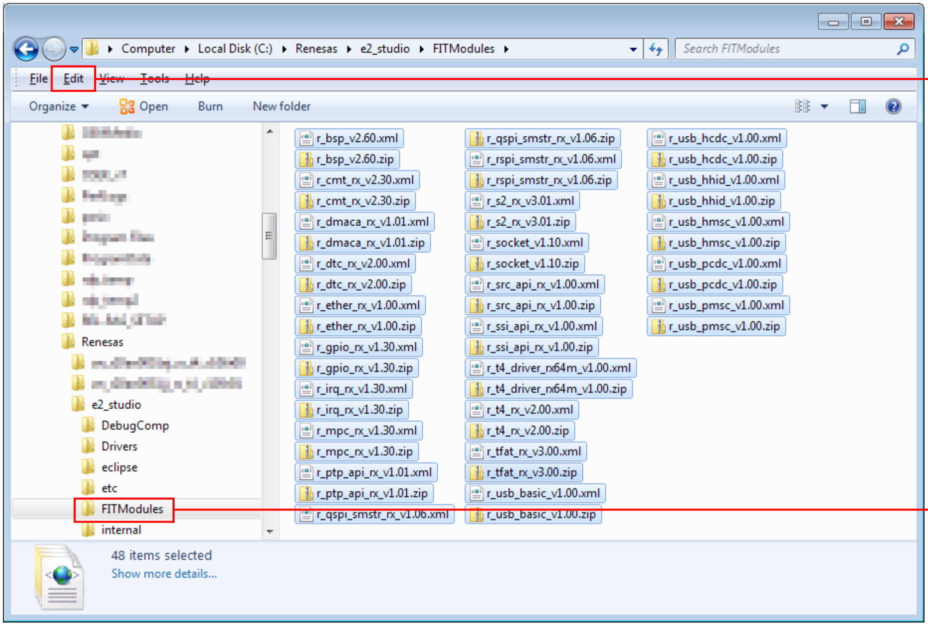
3. Select all the files in the **FITModules** folder and click **Copy** in the **Edit** menu.



Select all files and click **Copy** in the **Edit** menu.

4. Open the e^2 studio install folder (Usually, this will be c:/Renesas/e2_studio.) and open the **FITModules** folder in that folder.

5. Click **Paste** on the **Edit** menu.
   The e^2 studio **FITModules** folder will be copied to the FIT modules.



Open the **FITModules** folder and click **Paste** on the **Edit** menu.
The folder will be copied.

4.3 Application Creation

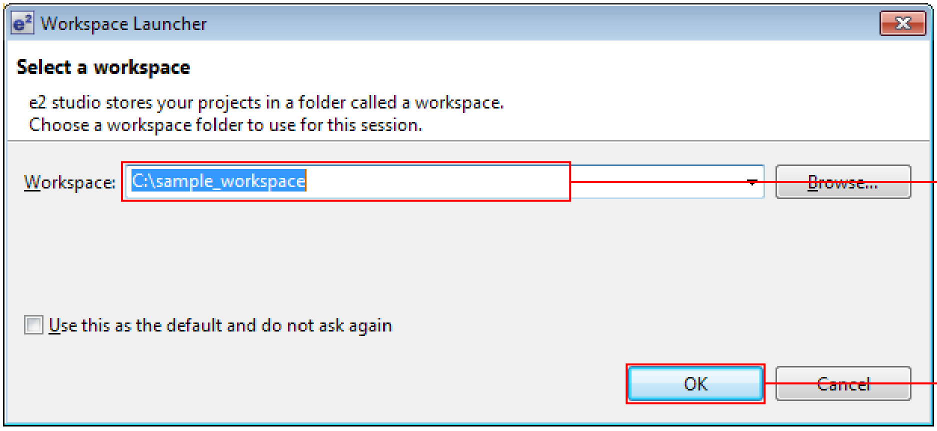
In this section, create a simple application that drives an LED.

4.3.1 Create a Workspace and a Project

First, create a new workspace and a new project.

1. Start e^2 studio.

2. Enter an arbitrary workspace folder in the displayed dialog box and click **OK**.



Select a workspace

e2 studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

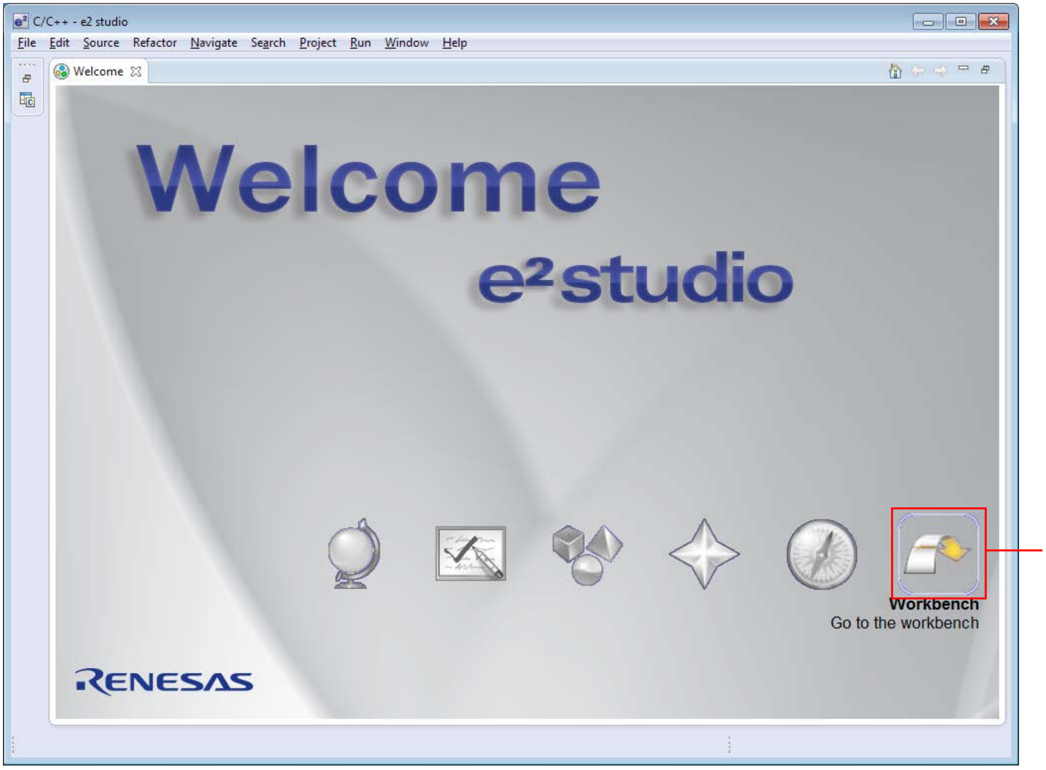
Workspace: C:\sample_workspace

□ Use this as the default and do not ask again

Enter a workspace folder.

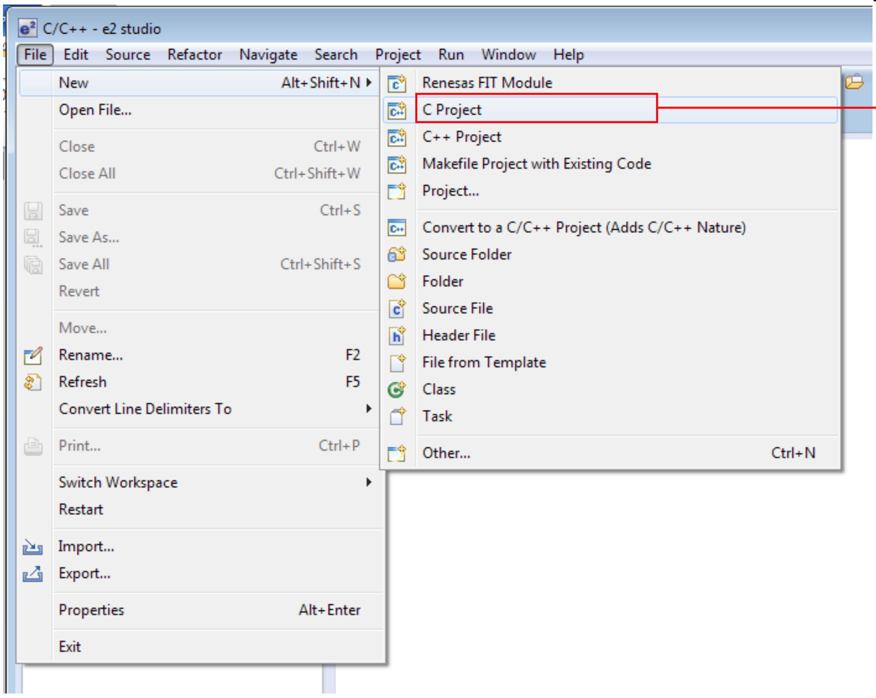
Click **OK**.

3. When the following window is displayed, click **Workbench**.



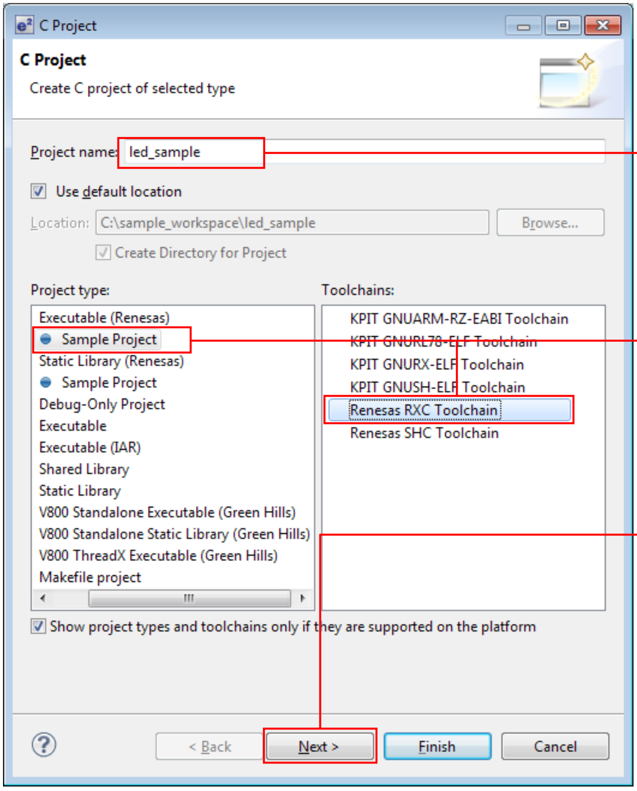
Click **Workbench**.

4. When the workbench has started, select **New** from the **File** menu and click **C Project**.



Click here.

5. Enter the project name. For the project type, click **Sample Project** under **Executable (Renesas)**. For the tool chain, click **Renesas RXC Toolchain**. After making these settings, click **Next**.



C Project

C Project

Create C project of selected type

Project name: led_sample

☑ Use default location

Location: C:\sample_workspace\led_sample

☑ Create Directory for Project

Project type:

- Executable (Renesas)
  - ● Sample Project
- Static Library (Renesas)
  - ● Sample Project
- Debug-Only Project
- Executable
- Executable (IAR)
- Shared Library
- Static Library
- V800 Standalone Executable (Green Hills)
- V800 Standalone Static Library (Green Hills)
- V800 ThreadX Executable (Green Hills)
- Makefile project

Toolchains:

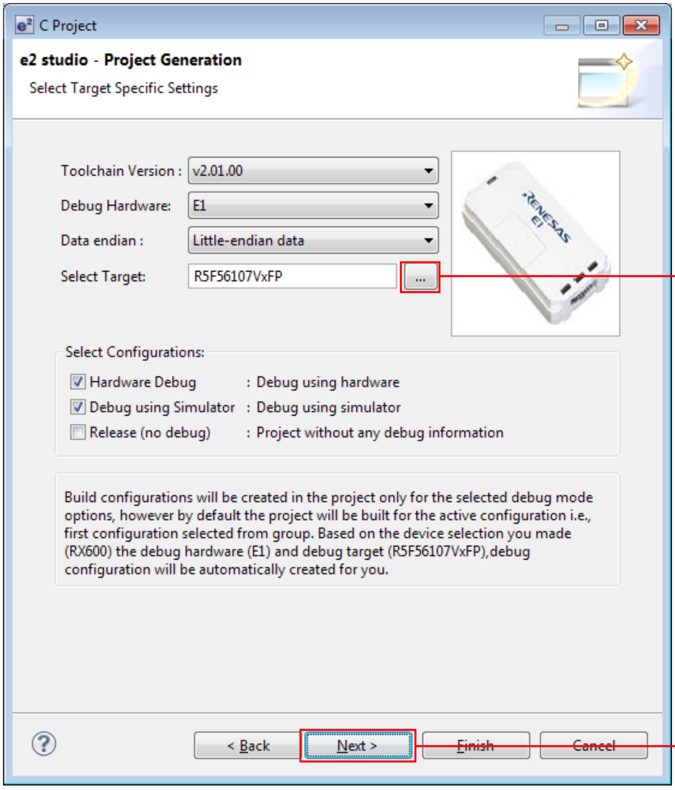
- KPIT GNUARM-RZ-EABI Toolchain
- KPIT GNURL78-ELF Toolchain
- KPIT GNURX-ELF Toolchain
- KPIT GNUSH-ELF Toolchain
- Renesas RXC Toolchain
- Renesas SHC Toolchain

☑ Show project types and toolchains only if they are supported on the platform

Enter the project name.

Click here.

6. Select the target. Click the “...” button under **Target Selection** and select **R5F564MLCxFC**. After making these settings, click **Next**.



C Project

e2 studio - Project Generation

Select Target Specific Settings

Toolchain Version : v2.01.00

Debug Hardware: E1

Data endian : Little-endian data

Select Target: R5F56107VxFP

Select Configurations:

☑ Hardware Debug : Debug using hardware

☑ Debug using Simulator : Debug using simulator

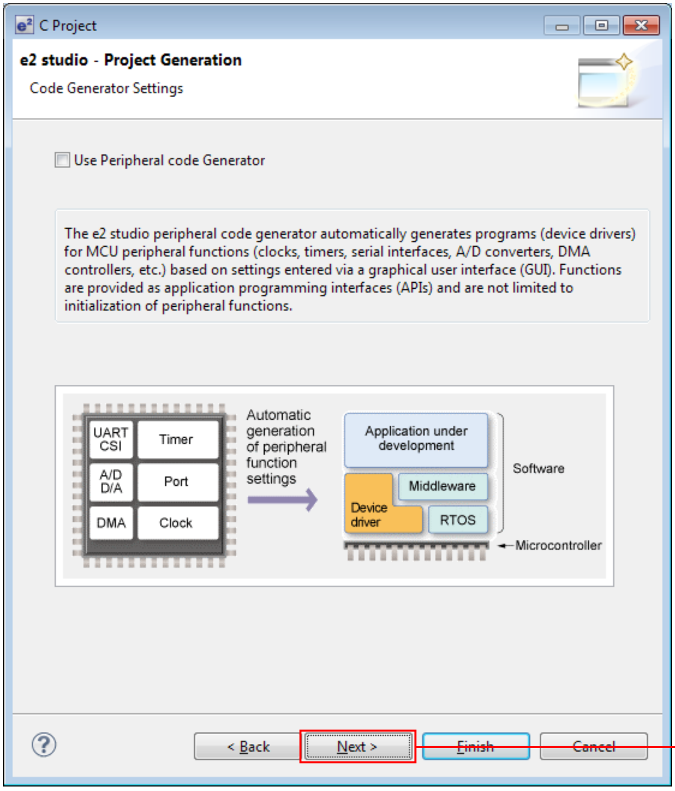
□ Release (no debug) : Project without any debug information

Build configurations will be created in the project only for the selected debug mode options, however by default the project will be built for the active configuration i.e., first configuration selected from group. Based on the device selection you made (RX600) the debug hardware (E1) and debug target (R5F56107VxFP),debug configuration will be automatically created for you.

Click here and select **R5F564MLCxFC**.

Click here.

7. Simply click **Next** here.



C Project

e2 studio - Project Generation

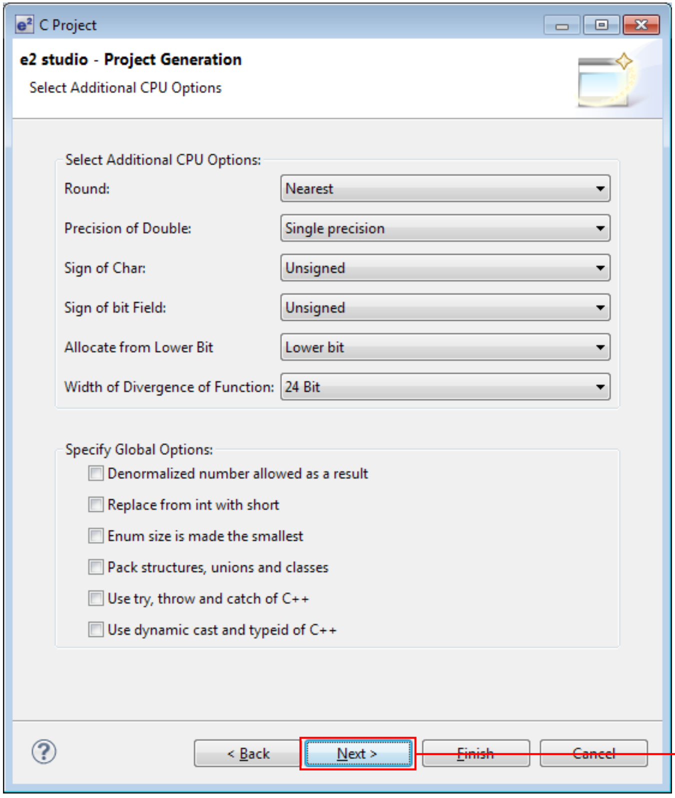
Code Generator Settings

□ Use Peripheral code Generator

The e2 studio peripheral code generator automatically generates programs (device drivers) for MCU peripheral functions (clocks, timers, serial interfaces, A/D converters, DMA controllers, etc.) based on settings entered via a graphical user interface (GUI). Functions are provided as application programming interfaces (APIs) and are not limited to initialization of peripheral functions.

Click here.

8. Simply click **Next** here.



C Project

e2 studio - Project Generation

Select Additional CPU Options

Select Additional CPU Options:

Round:	Nearest
Precision of Double:	Single precision
Sign of Char:	Unsigned
Sign of bit Field:	Unsigned
Allocate from Lower Bit	Lower bit
Width of Divergence of Function:	24 Bit

Specify Global Options:

□ Denormalized number allowed as a result

□ Replace from int with short

□ Enum size is made the smallest

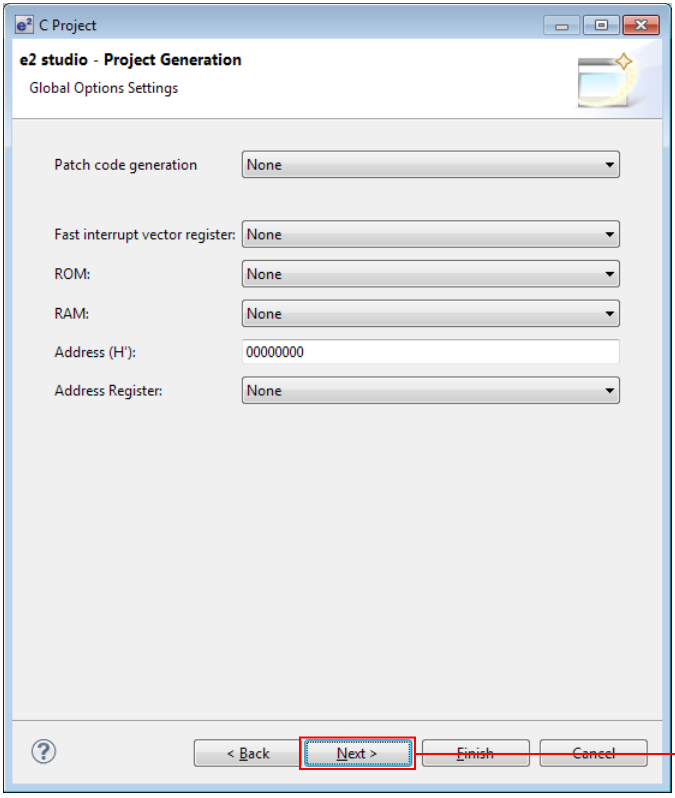
□ Pack structures, unions and classes

□ Use try, throw and catch of C++

□ Use dynamic cast and typeid of C++

Click here.

9. Simply click **Next** here.



C Project

e2 studio - Project Generation

Global Options Settings

Patch code generation: None

Fast interrupt vector register: None

ROM: None

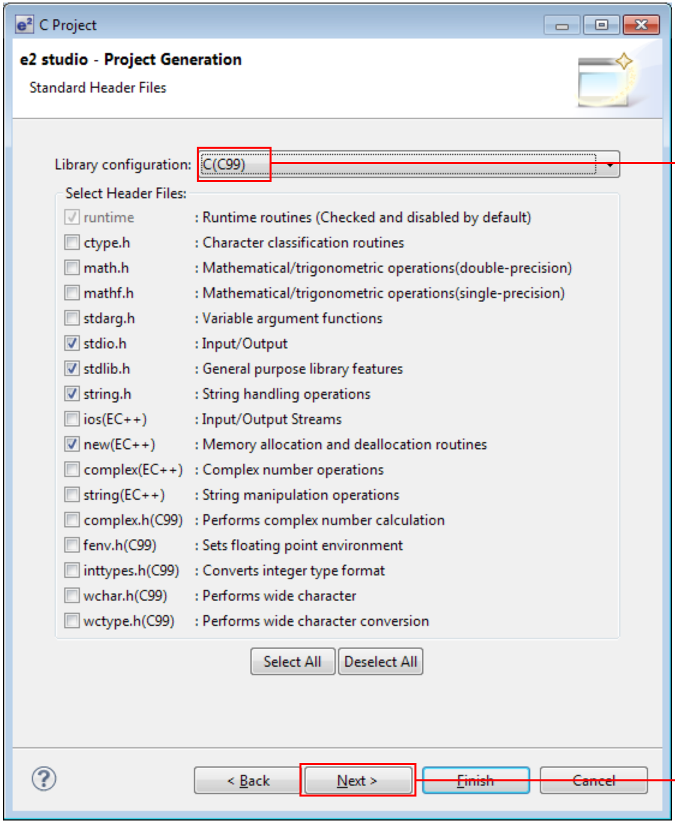
RAM: None

Address (H'): 00000000

Address Register: None

Click here.

10. Select **C(C99)** under **Library Structure** and click **Next**.



C Project

e2 studio - Project Generation

Standard Header Files

Library configuration: C(C99)

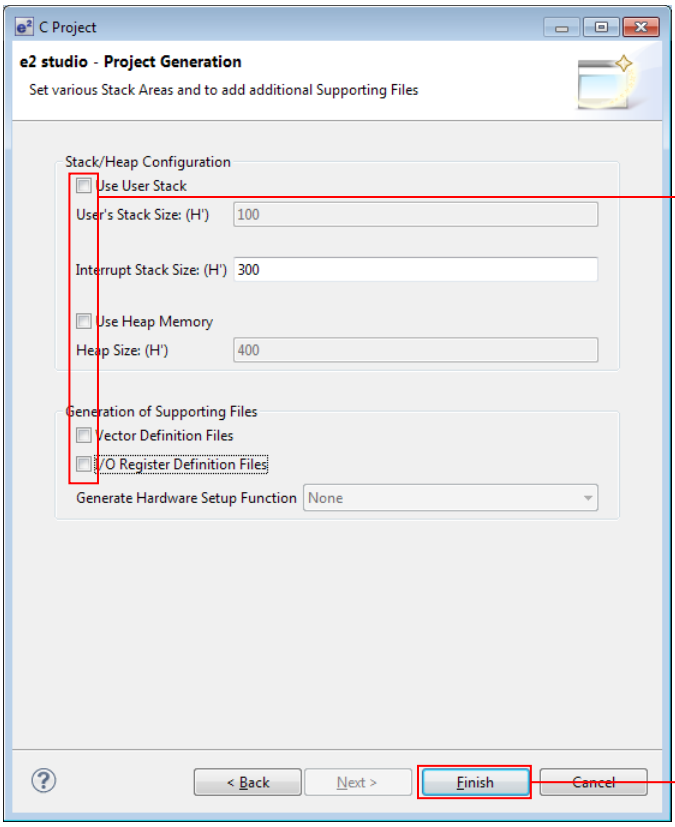
Select Header Files:

☑ runtime : Runtime routines (Checked and disabled by default)
☐ ctype.h : Character classification routines
☐ math.h : Mathematical/trigonometric operations(double-precision)
☐ mathf.h : Mathematical/trigonometric operations(single-precision)
☐ stdarg.h : Variable argument functions
☑ stdio.h : Input/Output
☑ stdlib.h : General purpose library features
☑ string.h : String handling operations
☐ ios(EC++) : Input/Output Streams
☑ new(EC++) : Memory allocation and deallocation routines
☐ complex(EC++) : Complex number operations
☐ string(EC++) : String manipulation operations
☐ complex.h(C99) : Performs complex number calculation
☐ fenv.h(C99) : Sets floating point environment
☐ inttypes.h(C99) : Converts integer type format
☐ wchar.h(C99) : Performs wide character
☐ wctype.h(C99) : Performs wide character conversion

Select **C(C99)**.

Click here.

11. Clear all four check boxes and click **Finish**.



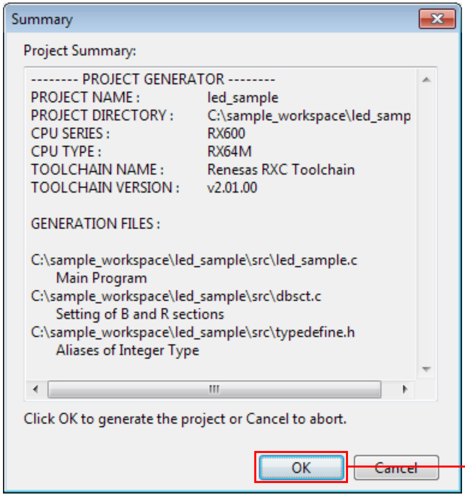
C Project
e2 studio - Project Generation
Set various Stack Areas and to add additional Supporting Files
Stack/Heap Configuration
Use User Stack
User's Stack Size: (H')
Interrupt Stack Size: (H')
Use Heap Memory
Heap Size: (H')
Generation of Supporting Files
Vector Definition Files
I/O Register Definition Files
Generate Hardware Setup Function
< Back
Next >
Finish
Cancel

□ Use User Stack
User's Stack Size: (H') 100
Interrupt Stack Size: (H') 300
□ Use Heap Memory
Heap Size: (H') 400
□ Vector Definition Files
□ I/O Register Definition Files
Generate Hardware Setup Function None

Clear all the check boxes.

Click here.

12. Click **OK**. The project will be generated.



Project Summary:

```
-------- PROJECT GENERATOR --------
PROJECT NAME :          led_sample
PROJECT DIRECTORY :     C:\sample_workspace\led_samp
CPU SERIES :            RX600
CPU TYPE :              RX64M
TOOLCHAIN NAME :        Renesas RXC Toolchain
TOOLCHAIN VERSION :     v2.01.00

GENERATION FILES :

C:\sample_workspace\led_sample\src\led_sample.c
    Main Program
C:\sample_workspace\led_sample\src\dbsct.c
    Setting of B and R sections
C:\sample_workspace\led_sample\src\typedefine.h
    Aliases of Integer Type
```

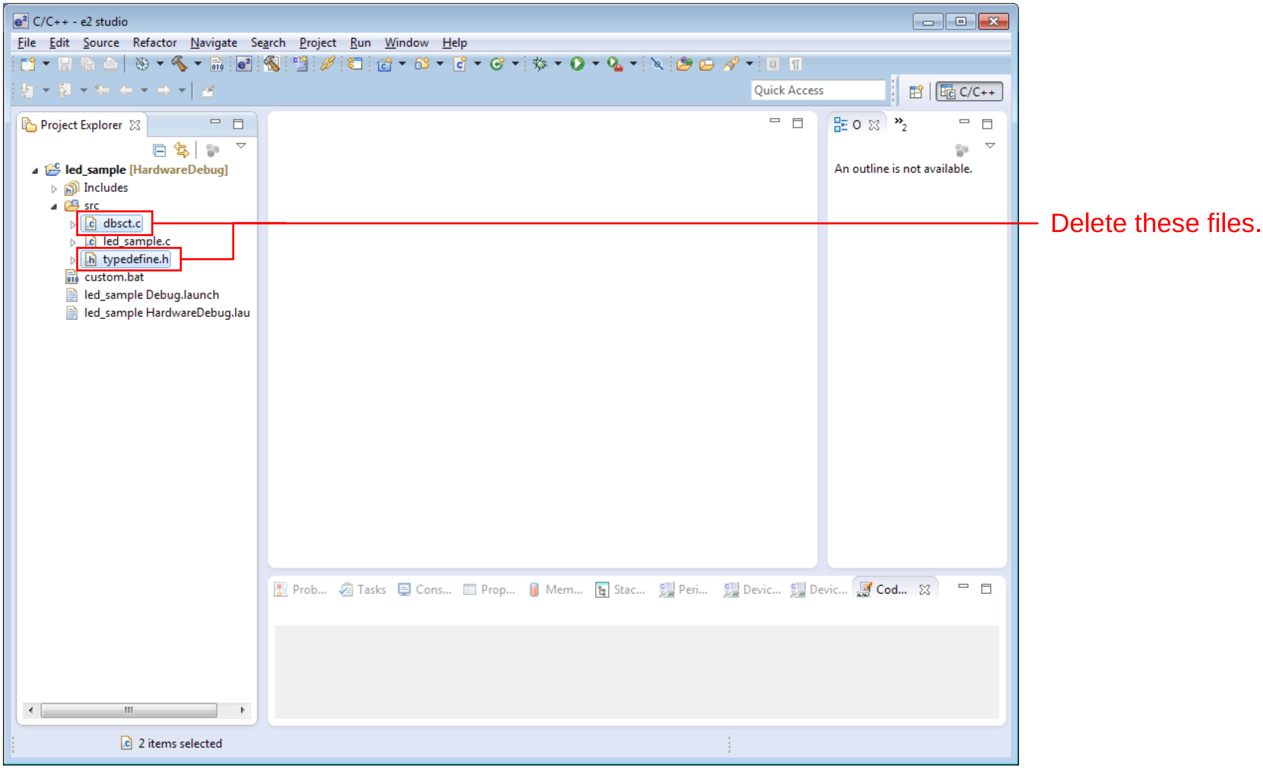
Click OK to generate the project or Cancel to abort.

Click here.

4.3.2 Prepare to Embed the FIT Modules

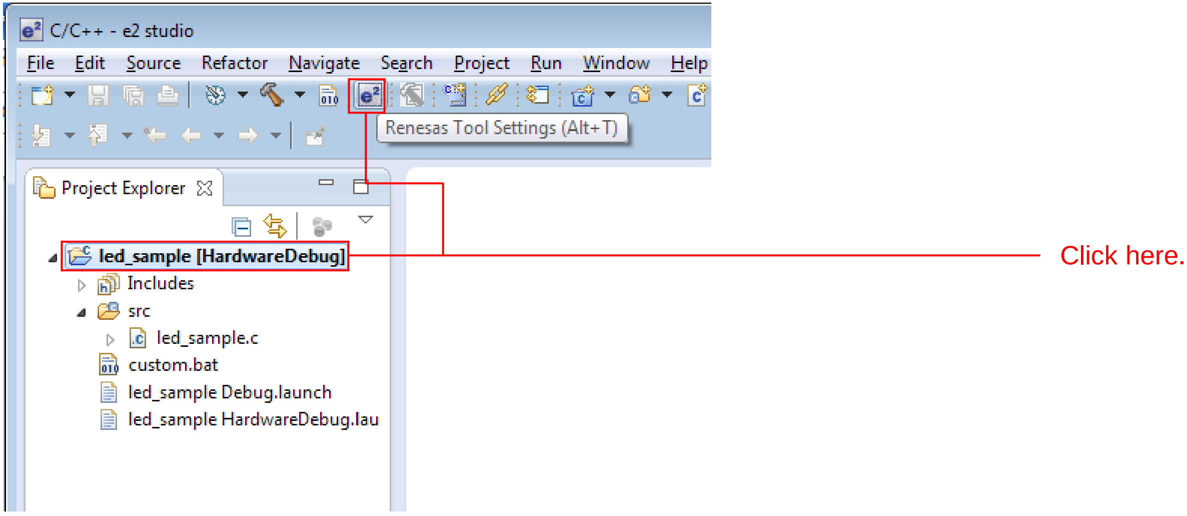
Since the FIT modules are independently initialized in BSP, it is necessary to modify certain aspects of the project generated by e^2 studio.

1. In **Project Explorer**, delete the **dbsct.c** and **typedefine.h** files in the **src** folder.


C/C++ - e2 studio
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
C/C++
Project Explorer
led_sample [HardwareDebug]
Includes
src
dbsct.c
led_sample.c
typedefine.h
custom.bat
led_sample Debug.launch
led_sample HardwareDebug.lau
An outline is not available.
Prob... Tasks Cons... Prop... Mem... Stac... Peri... Devic... Devic... Cod...
2 items selected

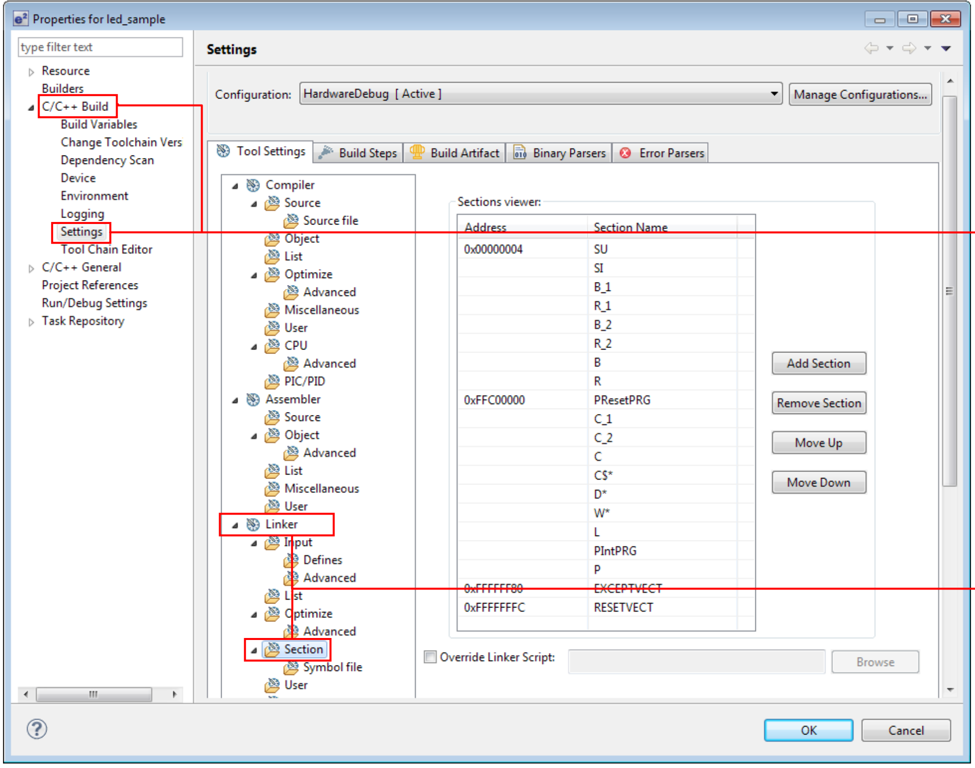
Delete these files.

2. In **Project Explorer**, click Project and open the properties by clicking **Renesas Tool Settings** in the toolbar.


C/C++ - e2 studio
File Edit Source Refactor Navigate Search Project Run Window Help
Renesas Tool Settings (Alt+T)
Project Explorer
led_sample [HardwareDebug]
Includes
src
led_sample.c
custom.bat
led_sample Debug.launch
led_sample HardwareDebug.lau

Click here.

3. Click **Settings** under **C/C++ Build** in the properties and click **Section** under **Linker**.



Configuration: HardwareDebug [Active] Manage Configurations...

Sections viewer:

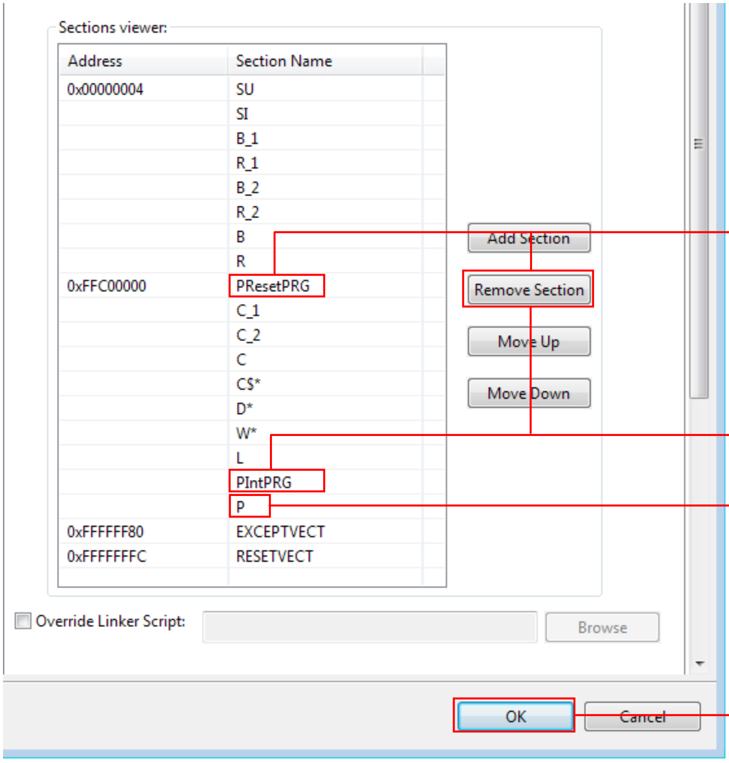
Address	Section Name
0x00000004	SU
	SI
	B_1
	R_1
	B_2
	R_2
	B
	R
0xFFC00000	PResetPRG
	C_1
	C_2
	C
	C$*
	D*
	W*
	L
	PIntPRG
	P
0xFFFFFF80	EXCEPTVECT
0xFFFFFFFC	RESETVECT

□ Override Linker Script:

Click here.

Click here.

4. In **Section View**, delete **PResetPRG** and **PIntPRG**. Click each section and click **Delete Section**. Also, the **P** section must be changed to **P***. Click the **P** section and modify it to be **P***.



Sections viewer:

Address	Section Name
0x00000004	SU
	SI
	B_1
	R_1
	B_2
	R_2
	B
	R
0xFFC00000	PResetPRG
	C_1
	C_2
	C
	C$*
	D*
	W*
	L
	PIntPRG
	P
0xFFFFFF80	EXCEPTVECT
0xFFFFFFFC	RESETVECT

□ Override Linker Script:

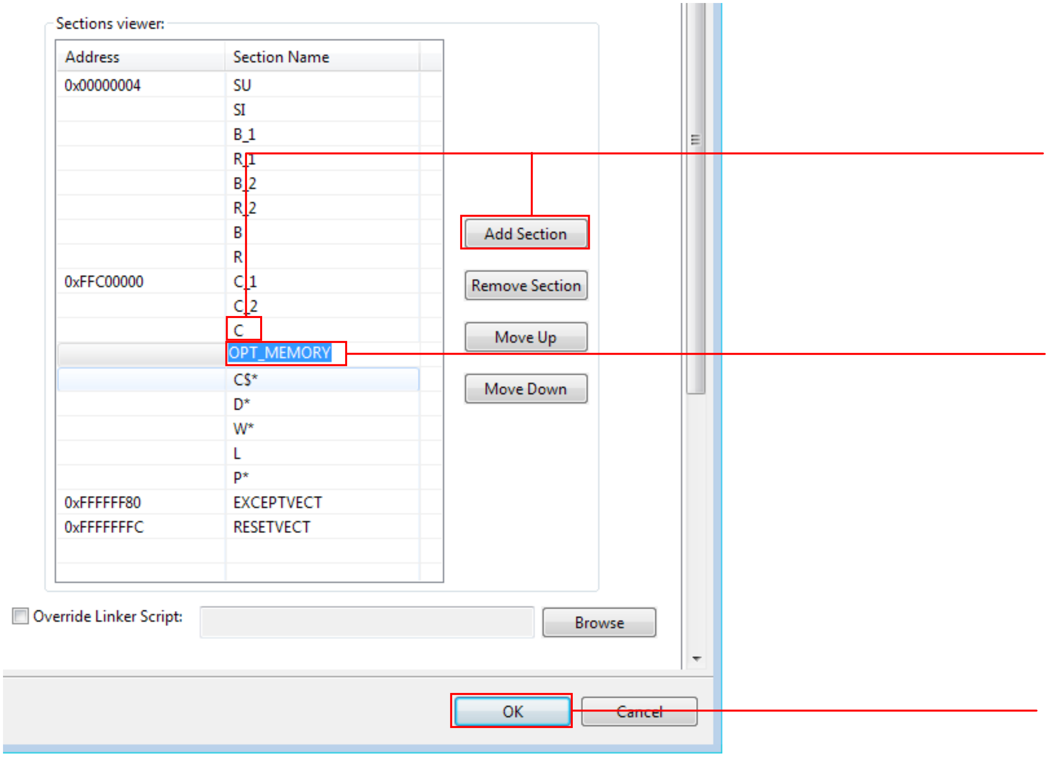
Click here.

Click here.

Modify this to be **P***.

Click here.

5. Add an **OPT_MEMORY** section after the **C** section. Click the **C** section and click **Add Section**. Click the added section, **NEW_SECTION_1**, and modify the name to be **OPT_MEMORY**.
   After completing the modification, click **OK**.



Sections viewer:

Address	Section Name
0x00000004	SU
	SI
	B_1
	R_1
	B_2
	R_2
	B
	R
0xFFC00000	C_1
	C_2
	C
	OPT_MEMORY
	C$*
	D*
	W*
	L
	P*
0xFFFFFF80	EXCEPTVECT
0xFFFFFFFC	RESETVECT

□ Override Linker Script:

Click here.

Modify this to be **OPT_MEMORY**.

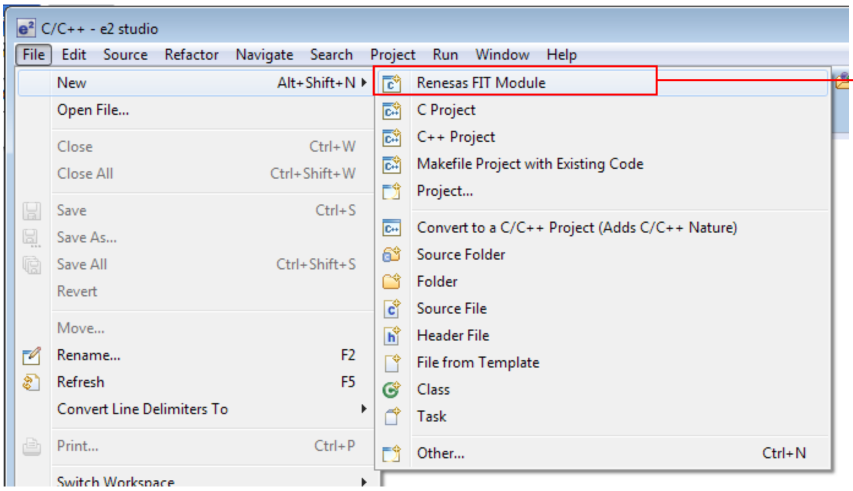
Click here.

4.3.3 Install the FIT Modules with the FIT Plugin.

Install the required modules with the FIT plugin into the created project.

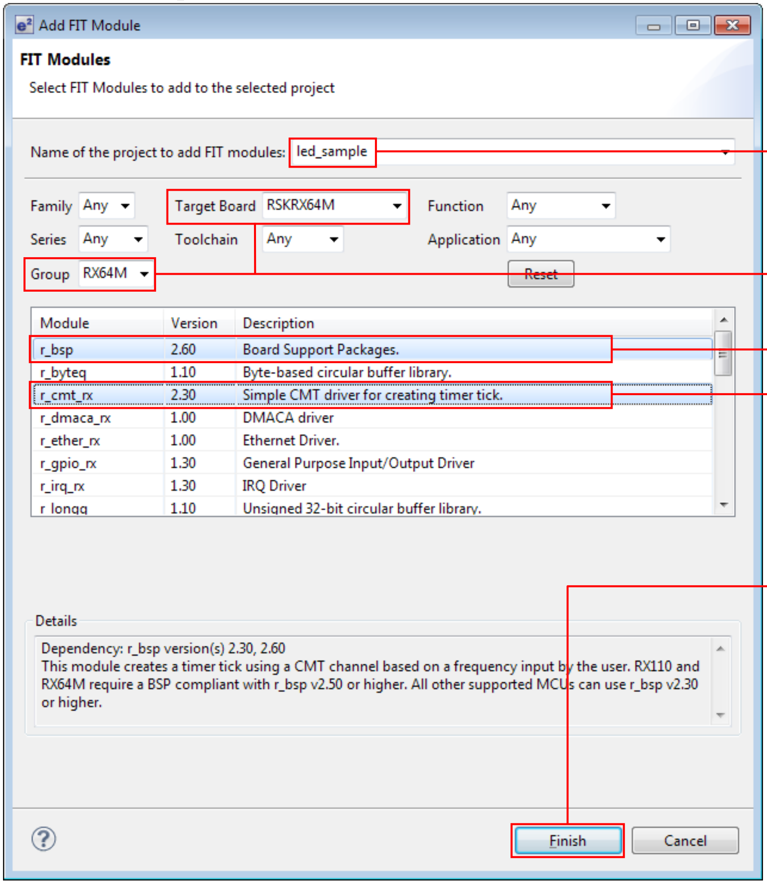
Here, install the BSP module (r_bsp) and the compare match timer driver (r_cmt_rx).

1. In the **File** menu, select **New** and click **Renesas FIT Module**.



Click here.

2. Select the created project with **Project to Add FIT Module** to.
Next, select **RX64M** under **Group** and select **RSKRX64M** from **Target Board**.
Next, click **r_bsp** and **r_cmt_rx** in the module list and click **Finish**.



Name of the project to add FIT modules: led_sample

Family: Any
Series: Any
Group: RX64M
Target Board: RSKRX64M
Toolchain: Any
Function: Any
Application: Any

Module	Version	Description
r_bsp	2.60	Board Support Packages.
r_byteq	1.10	Byte-based circular buffer library.
r_cmt_rx	2.30	Simple CMT driver for creating timer tick.
r_dmaca_rx	1.00	DMACA driver
r_ether_rx	1.00	Ethernet Driver.
r_gpio_rx	1.30	General Purpose Input/Output Driver
r_irq_rx	1.30	IRQ Driver
r_longq	1.10	Unsigned 32-bit circular buffer library.

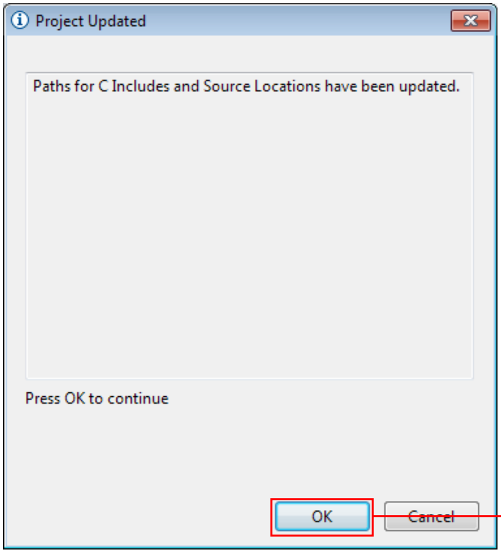
Dependency: r_bsp version(s) 2.30, 2.60
This module creates a timer tick using a CMT channel based on a frequency input by the user. RX110 and RX64M require a BSP compliant with r_bsp v2.50 or higher. All other supported MCUs can use r_bsp v2.30 or higher.

Select the project.

Select this item.

Click here.

3. Click **OK** with changing anything.



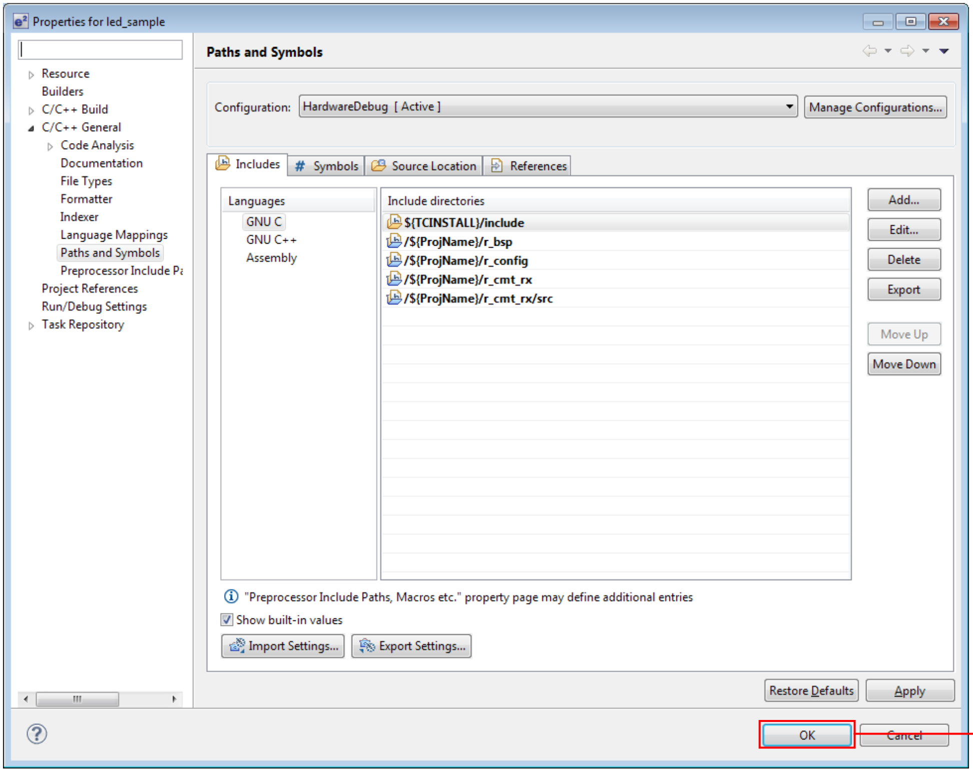
Project Updated

Paths for C Includes and Source Locations have been updated.

Press OK to continue

Click here.

4. Click **OK** with changing anything.



Properties for led_sample

Paths and Symbols

Configuration: HardwareDebug [Active]

Languages: GNU C, GNU C++, Assembly

Include directories:

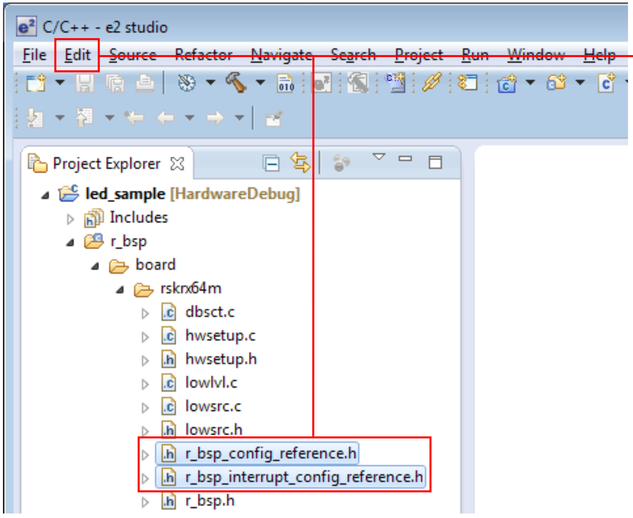
```
${TCINSTALL}/include
/${ProjName}/r_bsp
/${ProjName}/r_config
/${ProjName}/r_cmt_rx
/${ProjName}/r_cmt_rx/src
```

"Preprocessor Include Paths, Macros etc." property page may define additional entries

☑ Show built-in values

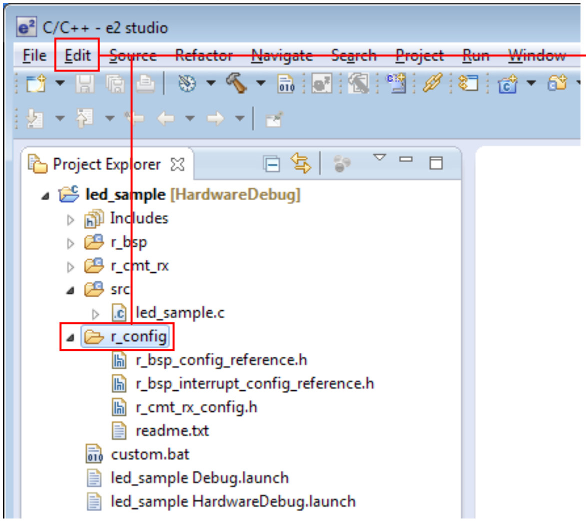
Click here.

5. Open **r_bsp/board/rskrx64m** from the e^2 studio Project Explorer, select the two files **r_bsp_config_reference.h** and **r_bsp_interrupt_config_reference.h**, and click **Copy** on the **Edit** menu.



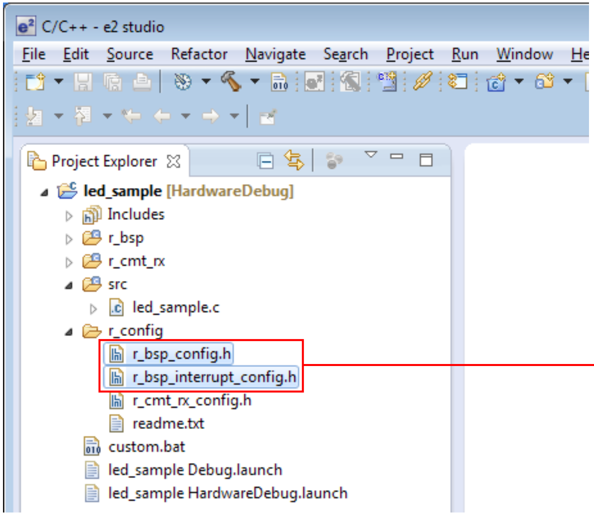
Select these two files and click **Copy** on the **Edit** menu.

6. Select r_config from the e^2 studio **Project Explorer** and click **Paste** on the **Edit** menu.



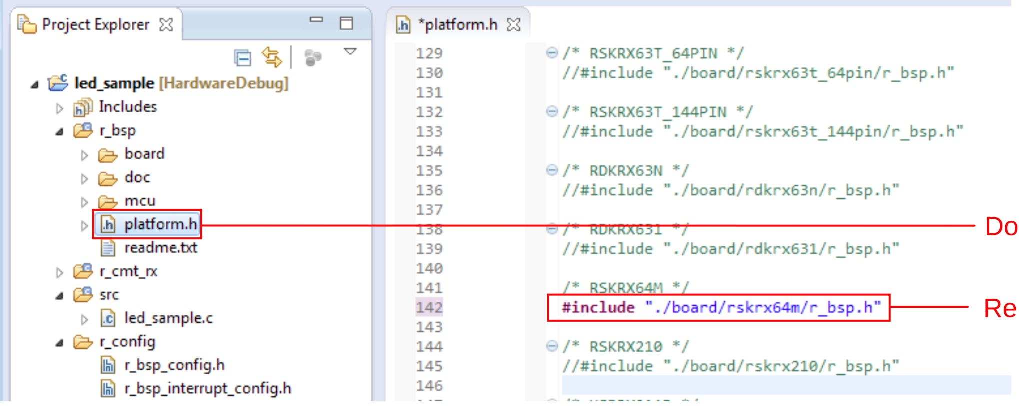
Select the **r_config** folder and click **Paste** on the **Edit** menu.

7. Change the names of the two copied files to **r_bsp_config.h** and **r_bsp_interrupt_config.h**. That is, delete the "_reference_" part of the file names.



Rename these files.

8. Modify platform.h to correspond to the target board used.
   Double click **r_bsp/platform.h** from the e^2 studio **Project Explorer** and, in the editor, remove the comment from the include line for the r_bsp.h file for the RSKRX64M.



```
/* RSKRX63T_64PIN */
//#include "./board/rskrx63t_64pin/r_bsp.h"

/* RSKRX63T_144PIN */
//#include "./board/rskrx63t_144pin/r_bsp.h"

/* RDKRX63N */
//#include "./board/rdkrx63n/r_bsp.h"

/* RDKRX631 */
//#include "./board/rdkrx631/r_bsp.h"

/* RSKRX64M */
#include "./board/rskrx64m/r_bsp.h"

/* RSKRX210 */
//#include "./board/rskrx210/r_bsp.h"
```

Double click.

Remove the comment.

4.3.4 Create an LED Driving Program

Create a program that toggles the LED0 on/off state every 0.5 seconds using the compare match timer.

Open the file src/led_sample.c and modify it as shown below.

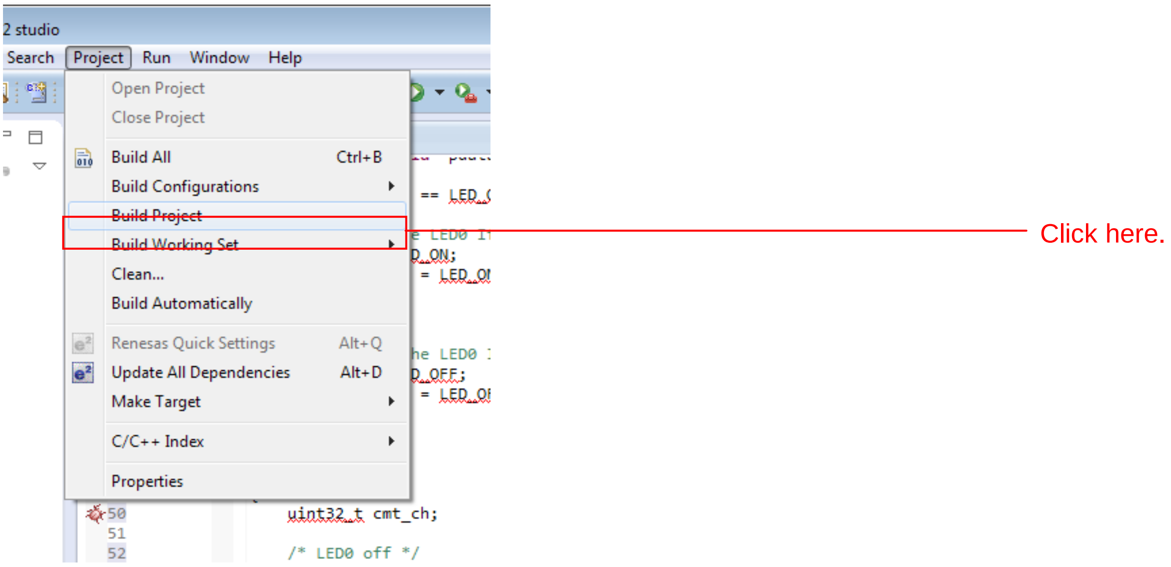
src/led_sample.c

```
/***********************************************************************/
/*                                                                     */
/*  FILE        :Main.c or Main.cpp                                    */
/*  DATE        :Tue, Oct 31, 2006                                     */
/*  DESCRIPTION :Main Program                                          */
/*  CPU TYPE    :                                                      */
/*                                                                     */
/*  NOTE:THIS IS A TYPICAL EXAMPLE.                                    */
/*                                                                     */
/***********************************************************************/
#include "platform.h"
#include "r_cmt_rx_if.h"

/* LED Currently status */
uint32_t ledstatus = LED_OFF;

void call_back(void *pdata)
{
 if (ledstatus == LED_OFF)
 {
    /* Turn ON the LED0 If the status is LED_OFF */
   LED0 = LED_ON;
   ledstatus = LED_ON;
 }
 else
 {
    /* Turn OFF the LED0 If the status is LED_ON */
   LED0 = LED_OFF;
   ledstatus = LED_OFF;
 }
}

void main(void)
{
 uint32_t cmt_ch;

 /* LED0 off */
 LED0 =LED_OFF;
 /* Create of 0.5 second(2Hz) cyclic timer. */
 R_CMT_CreatePeriodic(2, &call_back, &cmt_ch);

 while(1);
}
```

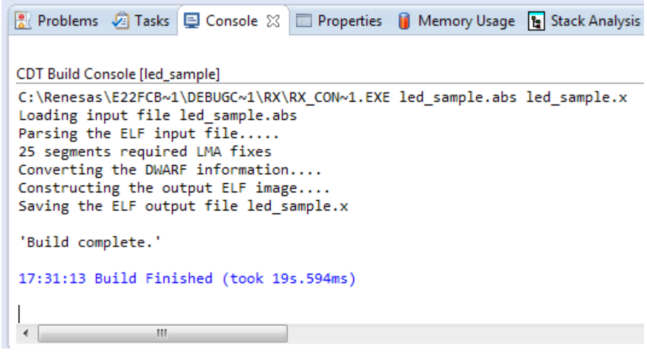
4.3.5 Build and Try Running the Program

Build the program just created and verify that it runs.

1. Click **Build Project** on the **Project** menu.



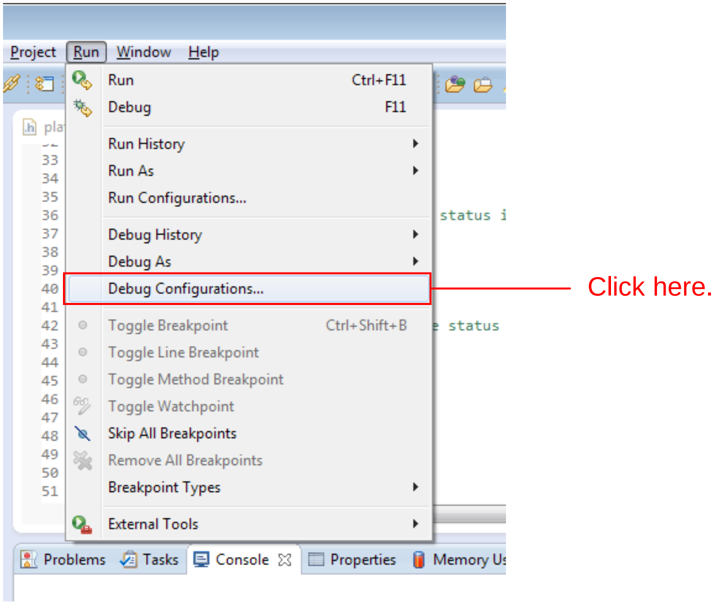
2. When the build completes, the following will be displayed in **Console** view.



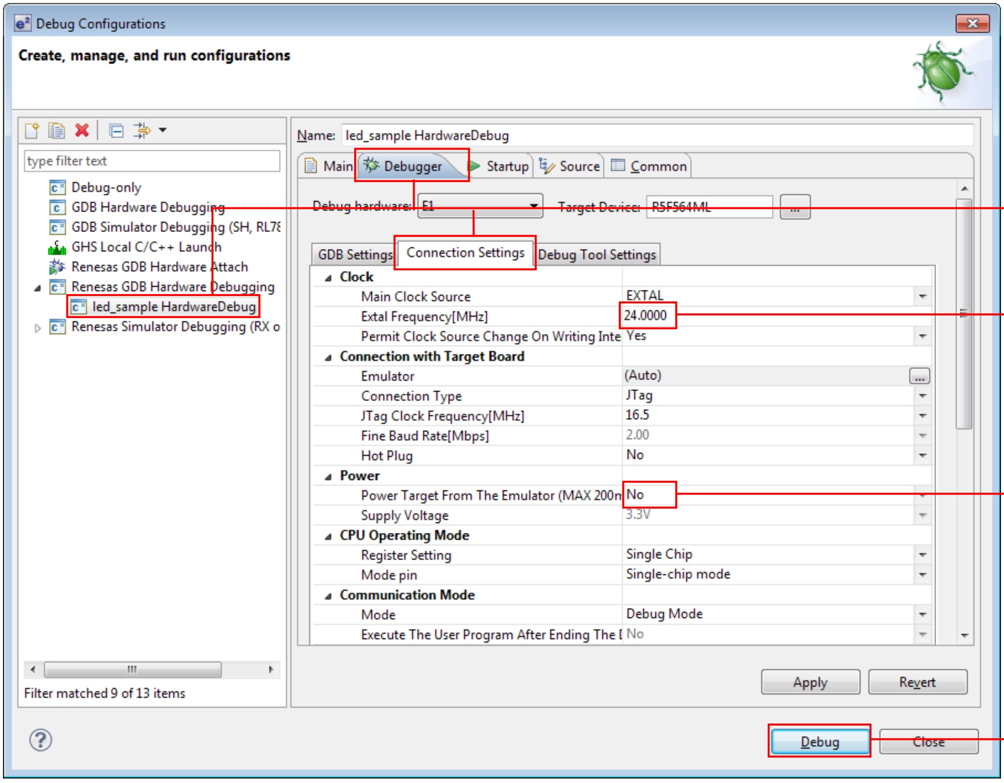
CDT Build Console [led_sample]

```
C:\Renesas\E22FCB~1\DEBUGC~1\RX\RX_CON~1.EXE led_sample.abs led_sample.x
Loading input file led_sample.abs
Parsing the ELF input file.....
25 segments required LMA fixes
Converting the DWARF information....
Constructing the output ELF image....
Saving the ELF output file led_sample.x

'Build complete.'

17:31:13 Build Finished (took 19s.594ms)
```

3. Click **Debug Build** on the **Run** menu.



4. Click **led_sample HardwareDebug** under the **Renesas GDB Hardware Debugging**. Click the **Debugger** tab and click **Connection Setting**.
   Modify **EXTAL Frequency** to be **24.0000** and change **Provide Power from Emulator** to **No**.
   When these changes have been made, click **Debug**.



Debug Configurations

Create, manage, and run configurations

type filter text

- Debug-only
- GDB Hardware Debugging
- GDB Simulator Debugging (SH, RL78
- GHS Local C/C++ Launch
- Renesas GDB Hardware Attach
- Renesas GDB Hardware Debugging
  - led_sample HardwareDebug
- Renesas Simulator Debugging (RX o

Filter matched 9 of 13 items

Name: led_sample HardwareDebug

Main | Debugger | Startup | Source | Common

Debug hardware: E1 Target Device: R5F564ML

GDB Settings | Connection Settings | Debug Tool Settings

Clock	
Main Clock Source	EXTAL
Extal Frequency[MHz]	24.0000
Permit Clock Source Change On Writing Inte	Yes
Connection with Target Board	
Emulator	(Auto)
Connection Type	JTag
JTag Clock Frequency[MHz]	16.5
Fine Baud Rate[Mbps]	2.00
Hot Plug	No
Power	
Power Target From The Emulator (MAX 200n	No
Supply Voltage	3.3V
CPU Operating Mode	
Register Setting	Single Chip
Mode pin	Single-chip mode
Communication Mode	
Mode	Debug Mode
Execute The User Program After Ending The I	No

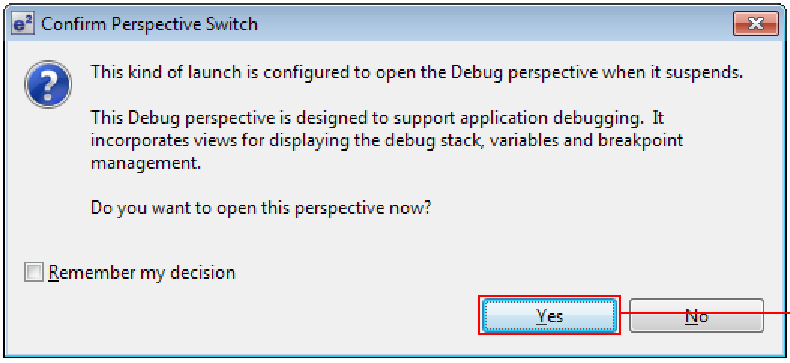
Click here.

Modify to be **24.0000**.

Modify to be **No**.

Click here.

5. When the following message is displayed, click **Yes**.



Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

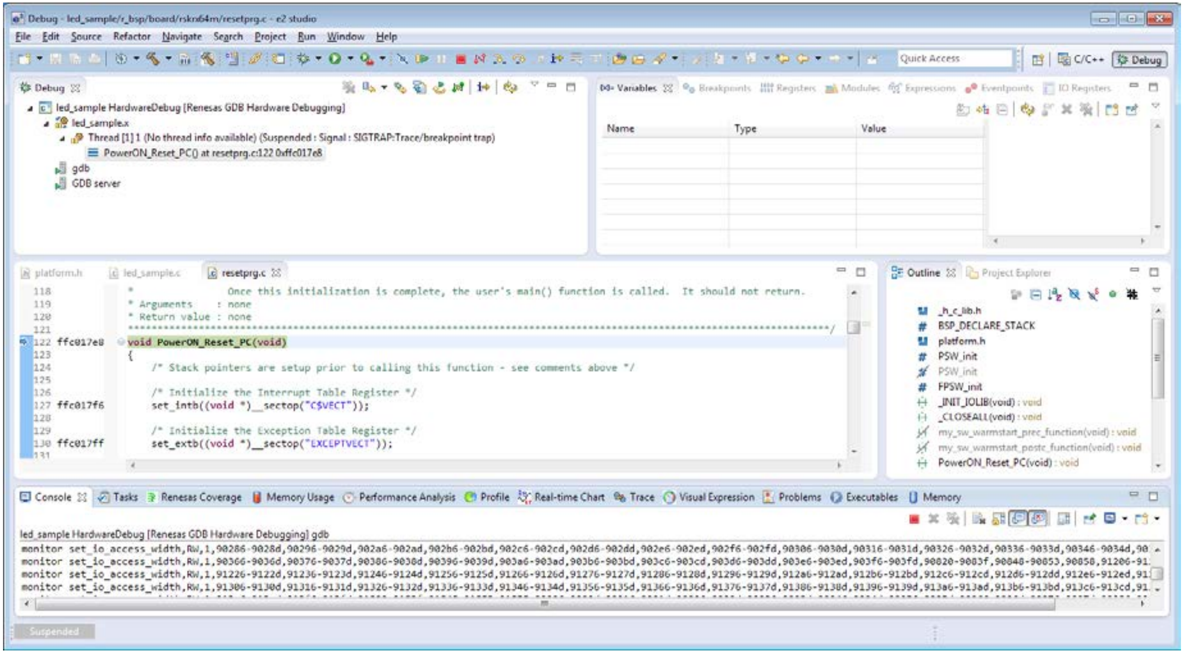
This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

□ Remember my decision

Click here.

6. When the load module download completes, a **Debug** perspective opens.



Debug - led_sample/r_bsp/board/rskrx64m/resetprg.c - e2 studio

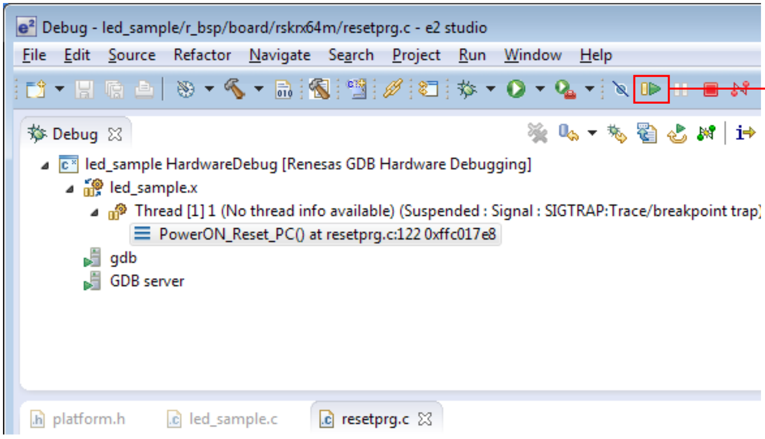
led_sample HardwareDebug [Renesas GDB Hardware Debugging]
led_sample.x
Thread [1] 1 (No thread info available) (Suspended : Signal : SIGTRAP:Trace/breakpoint trap)
PowerON_Reset_PC() at resetprg.c:122 0xffc017e8
gdb
GDB server

```
              *           Once this initialization is complete, the user's main() function is called.  It should not return.
              * Arguments    : none
              * Return value : none
              ***********************************************************************************************/
     ffc017e8 void PowerON_Reset_PC(void)
              {
                  /* Stack pointers are setup prior to calling this function - see comments above */

                  /* Initialize the Interrupt Table Register */
     ffc017f6     set_intb((void *)__sectop("C$VECT"));

                  /* Initialize the Exception Table Register */
     ffc017ff     set_extb((void *)__sectop("EXCEPTVECT"));
```

led_sample HardwareDebug [Renesas GDB Hardware Debugging] gdb

```
monitor set_io_access_width,RW,1,90286-9028d,90296-9029d,902a6-902ad,902b6-902bd,902c6-902cd,902d6-902dd,902e6-902ed,902f6-902fd,90306-9030d,90316-9031d,90326-9032d,90336-9033d,90346-9034d,90
monitor set_io_access_width,RW,1,90366-9036d,90376-9037d,90386-9038d,90396-9039d,903a6-903ad,903b6-903bd,903c6-903cd,903d6-903dd,903e6-903ed,903f6-903fd,90820-9083f,90848-90853,90858,91206-91
monitor set_io_access_width,RW,1,91226-9122d,91236-9123d,91246-9124d,91256-9125d,91266-9126d,91276-9127d,91286-9128d,91296-9129d,912a6-912ad,912b6-912bd,912c6-912cd,912d6-912dd,912e6-912ed,91
monitor set_io_access_width,RW,1,91306-9130d,91316-9131d,91326-9132d,91336-9133d,91346-9134d,91356-9135d,91366-9136d,91376-9137d,91386-9138d,91396-9139d,913a6-913ad,913b6-913bd,913c6-913cd,91
```

7. Click **Restart** on the toolbar. The program will be executed and a break will occur at the start of the main function.



Debug - led_sample/r_bsp/board/rskrx64m/resetprg.c - e2 studio

led_sample HardwareDebug [Renesas GDB Hardware Debugging]
led_sample.x
Thread [1] 1 (No thread info available) (Suspended : Signal : SIGTRAP:Trace/breakpoint trap)
PowerON_Reset_PC() at resetprg.c:122 0xffc017e8
gdb
GDB server

File Edit Source Refactor Navigate Search Project Run Window Help
Debug
platform.h
led_sample.c
resetprg.c

Click here.

8. After the break at the start of the main function, click **Restart** on the tool bar again.
The project will be run and the program will iterate toggling LED0 with a period of 0.5 seconds.

5. RX Driver Package Application

5.1 RX Driver Package Application Structure

The RX Driver Package Application is a sample application program provided so that users can use the RX Driver Package easily. The RX Driver Package Application consists of an application program that operates using device drivers and middleware included in the RX Driver Package and a project file for building that application. This allows users to start evaluation quickly.

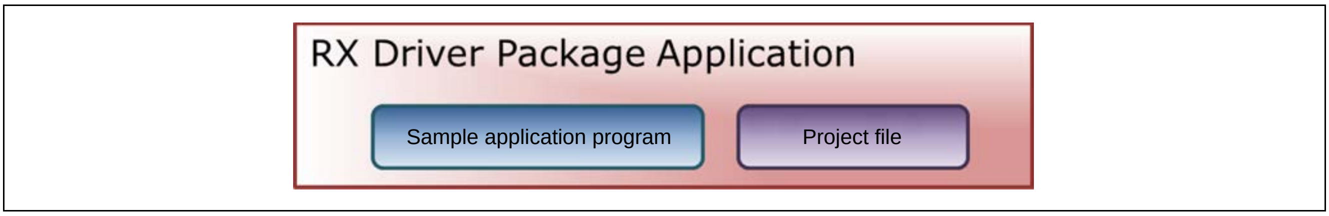


Figure 5.1.1 RX Driver Package Application Structure

Renesas plans to release a variety of types of this RX Driver Package Application in the future, such as system programs that operate using a combination of multiple drivers and middleware and evaluation programs for independent modules from the RX Driver Package.

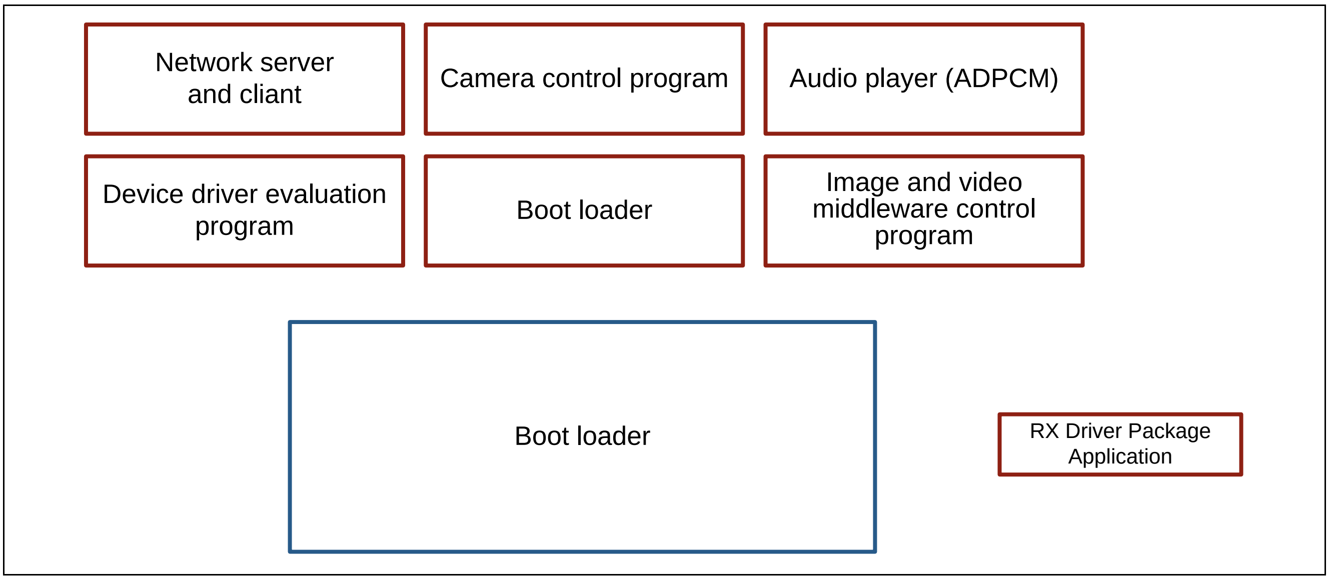

Network server and cliant
Camera control program
Audio player (ADPCM)
Device driver evaluation program
Boot loader
Image and video middleware control program
Boot loader
RX Driver Package Application

Figure 5.1.2 Types of RX Driver Package Application

5.2 RX Driver Package Application Features

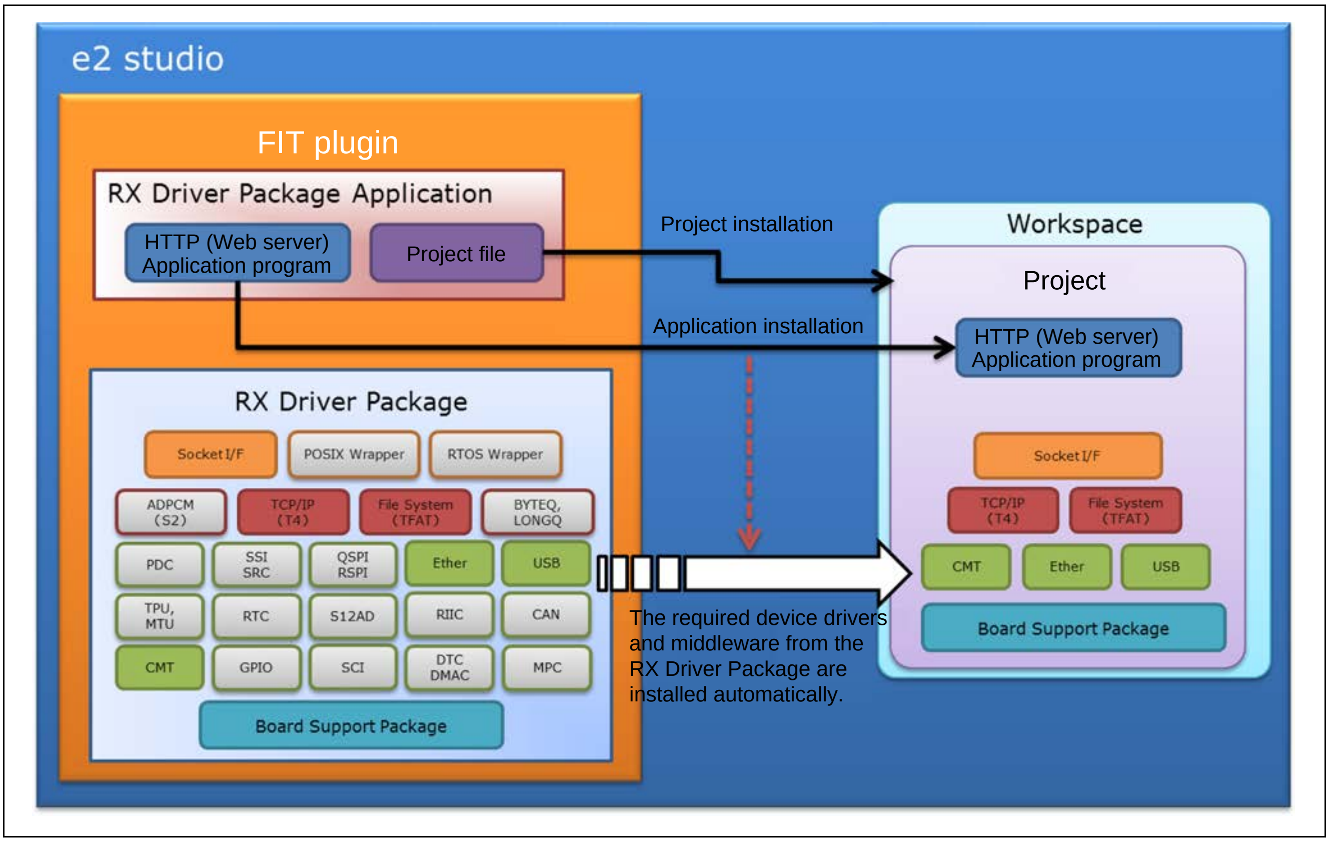
The RX Driver Package Application has the following features.

- The RX Driver Package Application is evaluated in combination with the RX Driver Package.
- Project files are included in the RX Driver Package Application. Since both build and debug configurations for the application project are already set up in the provided project file, the user can quickly begin building and evaluating simply by importing the project into a workspace.
- If e^2 studio is used as the integrated development environment (IDE), the device drivers and middleware used in the application project can be automatically added to the project by using the FIT plugin provided with e^2 studio.
- Renesas provides the RX Driver Package Application without charge.

5.3 RX Driver Package Application Usage Example (when e^2 studio is used)

The device drivers and middleware required by the RX Driver Package Application are automatically added to the project by the FIT plugin, which comes with the e^2 studio.

After the project provided with the RX Driver Package Application has been installed in an e^2 studio workspace, the required device drivers and middleware from the RX Driver Package are also installed simply by installing in the project by selecting the RX Driver Package Application with the FIT plugin. Therefore all that remains is to build the project and start evaluation.



The required device drivers and middleware from the RX Driver Package are installed automatically.

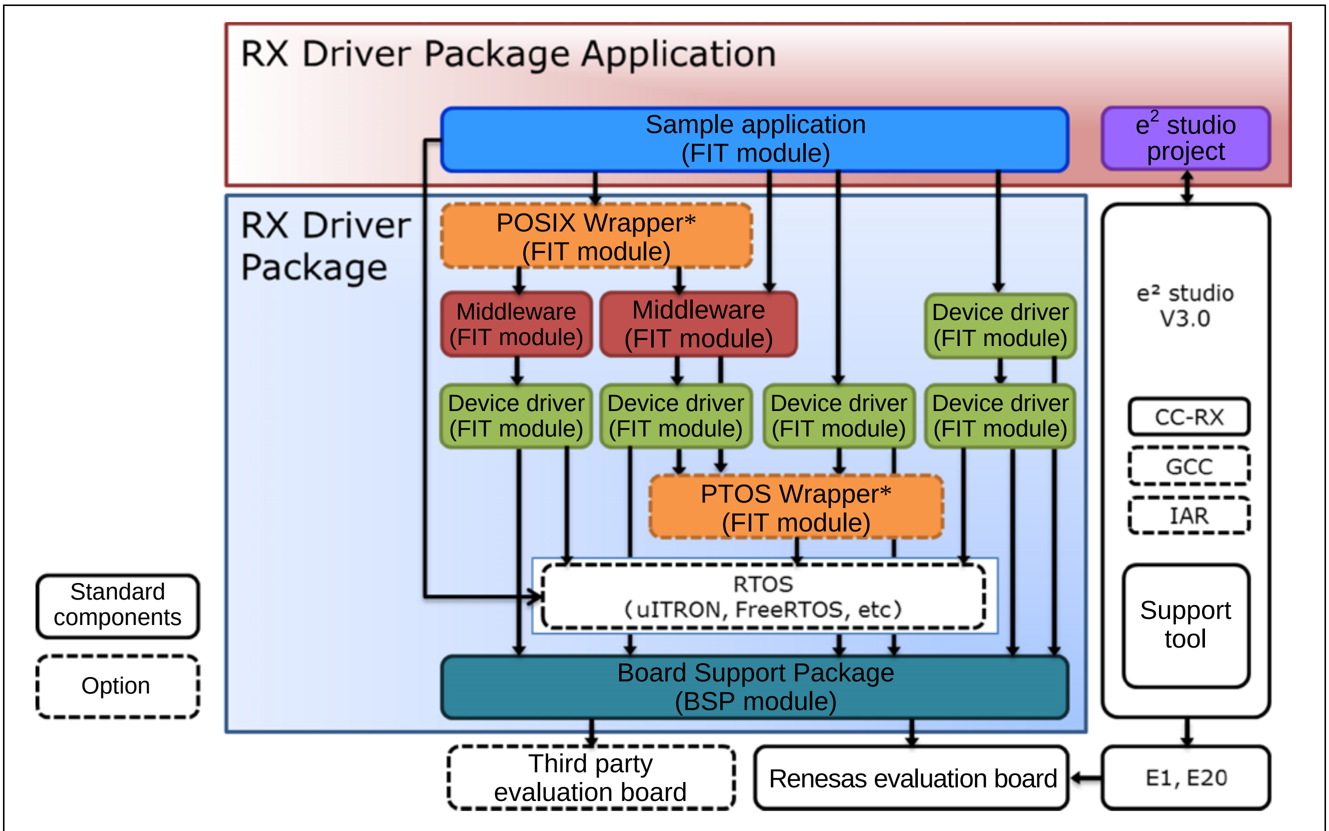
Figure 5.3.1 FIT Plugin Automatic Installation

5.4 When Using in Combination with an RX Driver Package Application

See the document provided with each RX Driver Package Application for detailed usage methods for that RX Driver Package Application.

6. System Structure

The figure below shows the structure of a system that uses the RX Driver Package.


RX Driver Package Application
Sample application (FIT module)
e^2 studio project
RX Driver Package
POSIX Wrapper* (FIT module)
Middleware (FIT module)
Middleware (FIT module)
Device driver (FIT module)
Device driver (FIT module)
Device driver (FIT module)
Device driver (FIT module)
Device driver (FIT module)
PTOS Wrapper* (FIT module)
RTOS (uITRON, FreeRTOS, etc)
Board Support Package (BSP module)
e² studio V3.0
CC-RX
GCC
IAR
Support tool
Standard components
Option
Third party evaluation board
Renesas evaluation board
E1, E20

Note: * Items marked with an asterisk are under development.

Figure 6.1 System Structure

7. Supplement

7.1 M3S-T4-Tiny（TCP/IP protocol stack ）

This package include the M3S-T4-Tiny (TCP/IP protocol stack library) of evaluation version. For the commercial version, please go to the below URL.

http://www.renesas.com/mw/t4

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

All trademarks and registered trademarks are the property of their respective owners.

Revision History

		Description	
Rev.	Date	Page	Summary
1.00	Sep 1, 2014	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU 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. Handling of Unused Pins
   Handle unused pins in accord 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.
5. Differences between Products
   Before changing from one product to another, i.e. to a product with a different type number, confirm that the change will not lead to problems.
   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended 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; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor 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 damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be 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, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

© 2014 Renesas Electronics Corporation. All rights reserved.
Colophon 4.0