Application Note

32

RX64M Group

Firmware Integration Technology (FIT) Tutorial For e^2 studio

RENESAS MCU
RX Family / 64M Series

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corporation without notice. Please review the latest information published by Renesas Electronics Corporation through various means, including the Renesas Electronics Corporation website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev. 1.00 May 2014

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; anticrime 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.

(2012.4)

Disclaimer

By using this Renesas Starter Kit+ (RSK+), the user accepts the following terms:

The RSK+ is not guaranteed to be error free, and the entire risk as to the results and performance of the RSK+ is assumed by the User. The RSK+ is provided by Renesas on an “as is” basis without warranty of any kind whether express or implied, including but not limited to the implied warranties of satisfactory quality, fitness for a particular purpose, title and non-infringement of intellectual property rights with regard to the RSK+. Renesas expressly disclaims all such warranties. Renesas or its affiliates shall in no event be liable for any loss of profit, loss of data, loss of contract, loss of business, damage to reputation or goodwill, any economic loss, any reprogramming or recall costs (whether the foregoing losses are direct or indirect) nor shall Renesas or its affiliates be liable for any other direct or indirect special, incidental or consequential damages arising out of or in relation to the use of this RSK+, even if Renesas or its affiliates have been advised of the possibility of such damages.

Precautions

The following precautions should be observed when operating any RSK+ product:

This Renesas Starter Kit+ is only intended for use in a laboratory environment under ambient temperature and humidity conditions. A safe separation distance should be used between this and any sensitive equipment. Its use outside the laboratory, classroom, study area or similar such area invalidates conformity with the protection requirements of the Electromagnetic Compatibility Directive and could lead to prosecution.

The product generates, uses, and can radiate radio frequency energy and may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment causes harmful interference to radio or television reception, which can be determined by turning the equipment off or on, you are encouraged to try to correct the interference by one or more of the following measures;

- ensure attached cables do not lie across the equipment
- reorient the receiving antenna
- increase the distance between the equipment and the receiver
- connect the equipment into an outlet on a circuit different from that which the receiver is connected
- power down the equipment when not in use
- consult the dealer or an experienced radio/TV technician for help NOTE: It is recommended that wherever possible shielded interface cables are used.

The product is potentially susceptible to certain EMC phenomena. To mitigate against them it is recommended that the following measures be undertaken;

- The user is advised that mobile phones should not be used within 10m of the product when in use.
- The user is advised to take ESD precautions when handling the equipment.

The Renesas Starter Kit does not represent an ideal reference design for an end product and does not fulfil the regulatory standards for an end product.

How to Use This Manual

1. Purpose and Target Readers

This manual is designed to provide the user with an understanding of how to use the e^2 studio IDE to create a working project using Renesas Firmware Integration Technology (FIT) modules. The document presents, in tutorial form, the steps required to create a working firmware program for the RSK+RX64M platform, starting from a FIT Board Support Package (BSP), adding a FIT module (Flash Library), then integrating code from the peripheral code generator built into e^2 studio.

The firmware described in this document is the RSK+ sample System_Input_Capture, which captures inputs from the RX64M temperature sensor, ADC and RTC and logs this data in non-volatile storage, for later retrieval over a serial port. The document is written for the RSK+RX64M specifically, but the steps required to create and import FIT modules and code generator files into a working project are applicable across the RX family.

The manual comprises of step-by-step instructions to generate code, but does not intend to be a complete guide to software development on the RSK+ platform. Further details regarding operating the RX64M microcontroller may be found in the Hardware Manual and within the provided sample code.

Particular attention should be paid to the precautionary notes when using the manual. These notes occur within the body of the text, at the end of each section, and in the Usage Notes section.

The revision history summarizes the locations of revisions and additions. It does not list all revisions. Refer to the text of the manual for details.

The following documents apply to the RX64M Group. Make sure to refer to the latest versions of these documents. The newest versions of the documents listed may be obtained from the Renesas Electronics Web site.

Document Type	Description	Document Title	Document No.
User's Manual	Describes the technical details of the RSK+ hardware.	RSK+RX64M User's Manual	R20UT2593EG
Tutorial	Provides a guide to setting up RSK+ environment, running sample code and debugging programs.	RSK+RX64M Tutorial Manual	R20UT2594EG
Quick Start Guide	Provides simple instructions to setup the RSK+ and run the first sample.	RSK+RX64M Quick Start Guide	R20UT2595EG
Code Generator Tutorial	Provides a guide to code generation and importing into the e^2 studio IDE.	RSK+RX64M Code Generator Tutorial Manual	R20UT2930EG
Schematics	Full detail circuit schematics of the RSK+.	RSK+RX64M Schematics	R20UT2589EG
Hardware Manual	Provides technical details of the RX64M microcontroller.	RX64M Group Hardware Manual	R01UH0377EJ
Application Note	Provides technical details of the FIT Flash API Module for RX64M.	RX64M Group Flash API for RX	R01AN0319EG
Application Note (this document)	Provides a guide to creating a project a FIT-based from scratch within the e^2 studio IDE.	RSK+RX64M FIT Tutorial Manual	R01AN0319EG

The following documents are applicable across the RX family and relate to the FIT architecture. Make sure to refer to the latest versions of these documents. The newest versions of the documents listed may be obtained from the Renesas Electronics Web site.

Document Type	Description	Document Title	Document No.
Application Note	Describes the technical details of the FIT Board Support Package for the RX family.	RX Family Board Support Package Module Using Firmware Integration Technology	R01AN1865EG
Application Note	Describes how to add and configure a FIT module to an e^2 studio project. .	RX Family Adding Firmware Integration Technology Modules to Projects	R01AN1723EG

2. List of Abbreviations and Acronyms

Abbreviation	Full Form
API	Application Programming Interface
BSP	Board Support Package
CMT	Compare Match Timer
CPU	Central Processing Unit
DVD	Digital Versatile Disc
E1	On-chip Debugger
FIT	Renesas Firmware Integration Technology
IDE	Integrated Development Environment
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MCU	Micro-controller Unit
NVM	Non Volatile Memory
RSK	Renesas Starter Kit
SCI	Serial Communications Interface

Table of Contents

1. Overview

1.1 Purpose

This RSK+ is an evaluation tool for Renesas microcontrollers. This manual describes how use FIT modules and peripheral code generator modules for RX together with the e^2 studio IDE to create a working project for the RSK+ platform.

1.2 Features

This RSK+ provides an evaluation of the following features:

- Creation of a base project using the RSK+RX64M Board Support Package (BSP) FIT module.
- Integration of the Flash library FIT module into the project.
- Peripheral Code Generation and integration using e^2 studio for RX64M.
- Project Building with e^2 studio.

The RSK+ board contains all the circuitry required for microcontroller operation.

2. Introduction

This application note is designed to answer, in tutorial form, how to use FIT modules for the RX family together with the e^2 studio IDE to create a working project for the RSK platform. The tutorials help explain the following:

- Creation of an empty e^2 studio project and importing and configuring the BSP for RSK+RX64M
- Importing and configuring an additional FIT Module
- Using the e^2 studio peripheral code generator alongside FIT modules
- Integration with custom code
- Building the project e^2 studio

The project generator will create a tutorial project with tow selectable build configurations:

- ‘HardwareDebug’ is a project built with the debugger support included. Optimisation is set to zero.
- ‘Release’ is a project with optimised compile options, producing code suitable for release in a product.

Some of the illustrative screenshots in this document will show text in the form RXxxx. These are general screenshots and are applicable across the whole RX family. In this case, simply substitute RXxxx for RX64M

These tutorials are designed to show you how to use the RSK+ and are not intended as a comprehensive introduction to the e^2 studio debugger, compiler toolchains or the E1 emulator. Please refer to the relevant user manuals for more in-depth information.

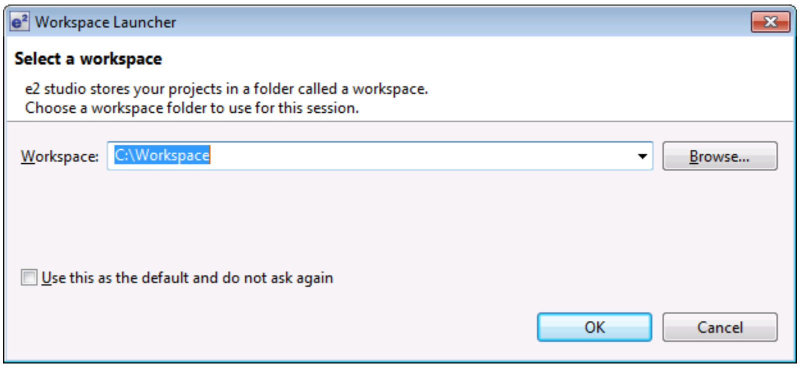
3. Creating an Empty Project and Importing the BSP

The following tutorial steps are taken from the two FIT reference documents r01an1723eg and r01an1685eg contained on the RSK+ DVD. Refer to these documents for further technical details on the BSP and FIT. The RSK+ installer will have already installed the FIT modules in the correct place in the e^2 studio install directory, so these steps have been skipped in this tutorial.

3.1 Creating the Empty Project

Start e^2 studio and select a suitable location for the project workspace

- Start e^2 studio and select a suitable location for the project workspace.



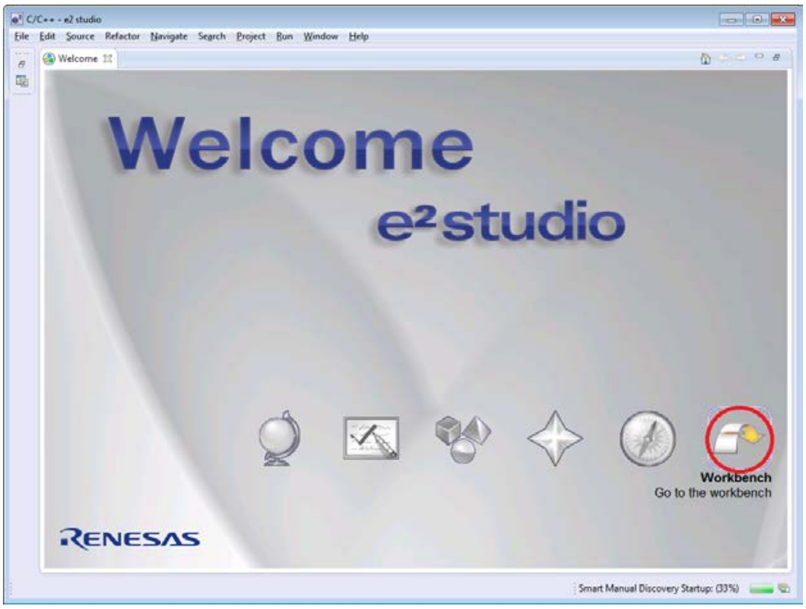
Select a workspace

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

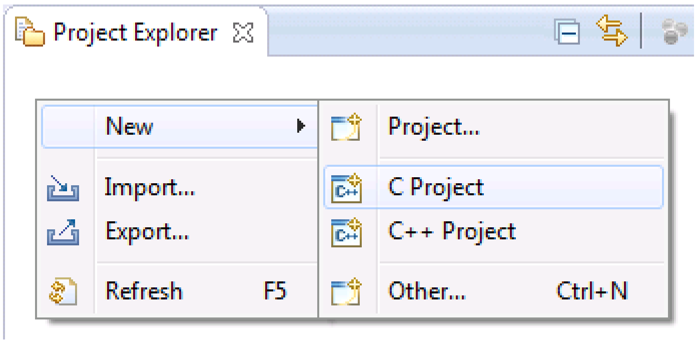
Workspace: C:\Workspace

□ Use this as the default and do not ask again

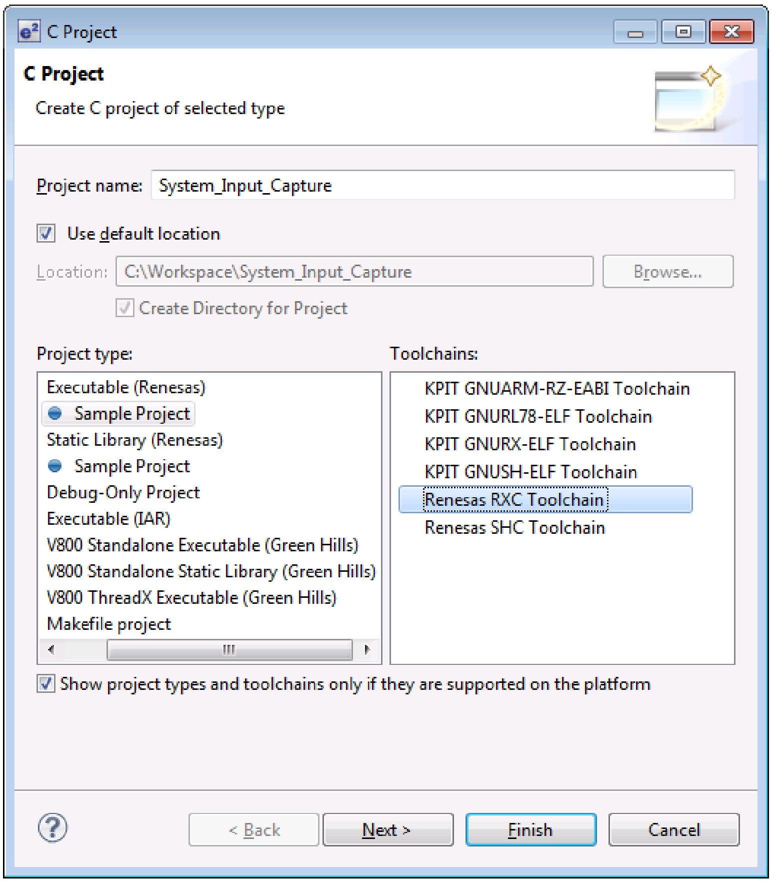
- In the Welcome page, click 'Go to the workbench'.


C/C++ - e2 studio
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome
Welcome
e²studio
Workbench
Go to the workbench
RENESAS
Smart Manual Discovery Startup: (33%)

- Create a new C project by right-clicking in the Project Explorer pave and selecting 'New -> C Project' as shown. Alternatively, use the menu item 'File -> New -> C Project'.



- Enter the project name ‘System_Input_Capture’. In ‘Project type:’ choose ‘Sample Project’. In ‘Toolchains’ choose ‘Renesas RXC Toolchain’. Click ‘Next’.



C Project

C Project

Create C project of selected type

Project name: System_Input_Capture

☑ Use default location

Location: C:\Workspace\System_Input_Capture Browse...

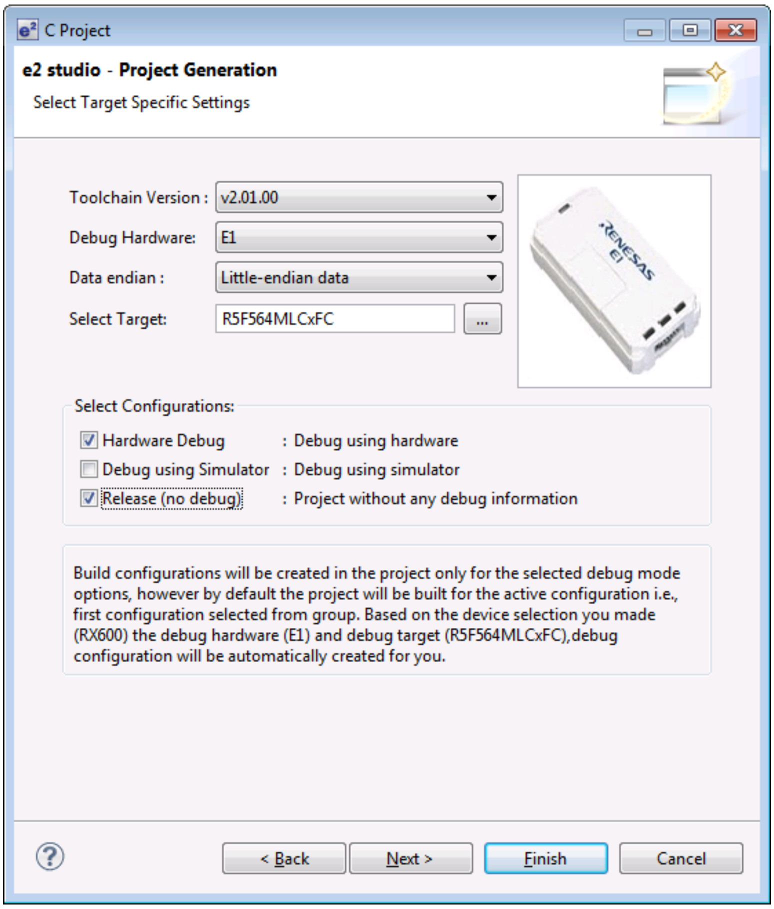
☑ Create Directory for Project

Project type:
- Executable (Renesas)
  - ● Sample Project
- Static Library (Renesas)
  - ● Sample Project
- Debug-Only Project
- Executable (IAR)
- 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

- In the ‘Target Specific Settings’ dialog, select the options as shown in the screenshot opposite.



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: R5F564MLCxFC

Select Configurations:

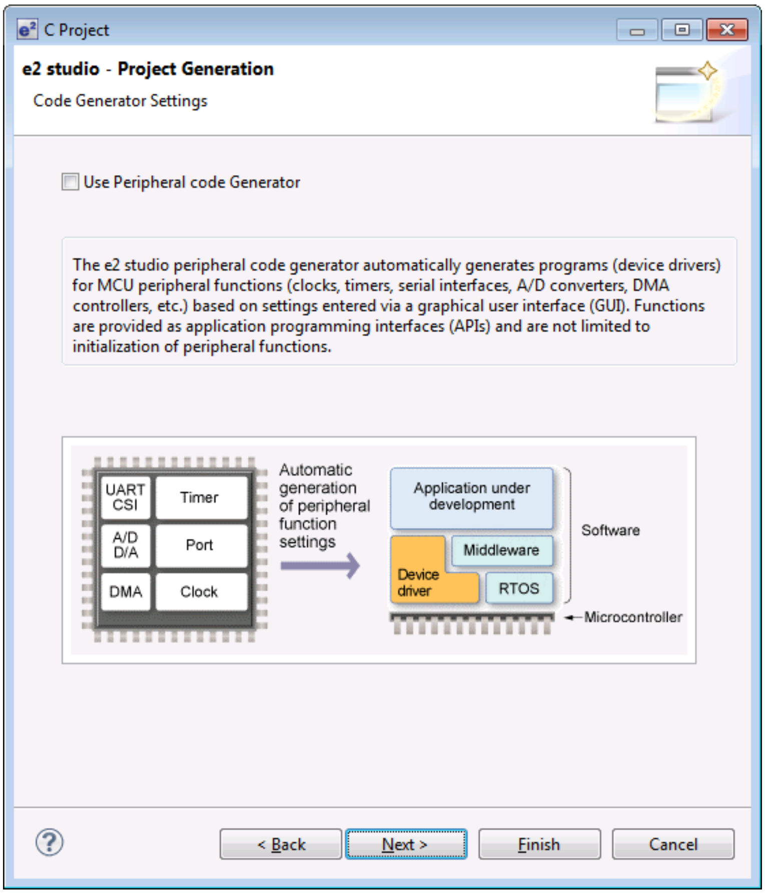
☑ Hardware Debug : Debug using hardware

☐ 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 (R5F564MLCxFC),debug configuration will be automatically created for you.

- In the ‘Code Generator Settings’ dialog, leave ‘Use Peripheral code Generator’ unchecked.



C Project

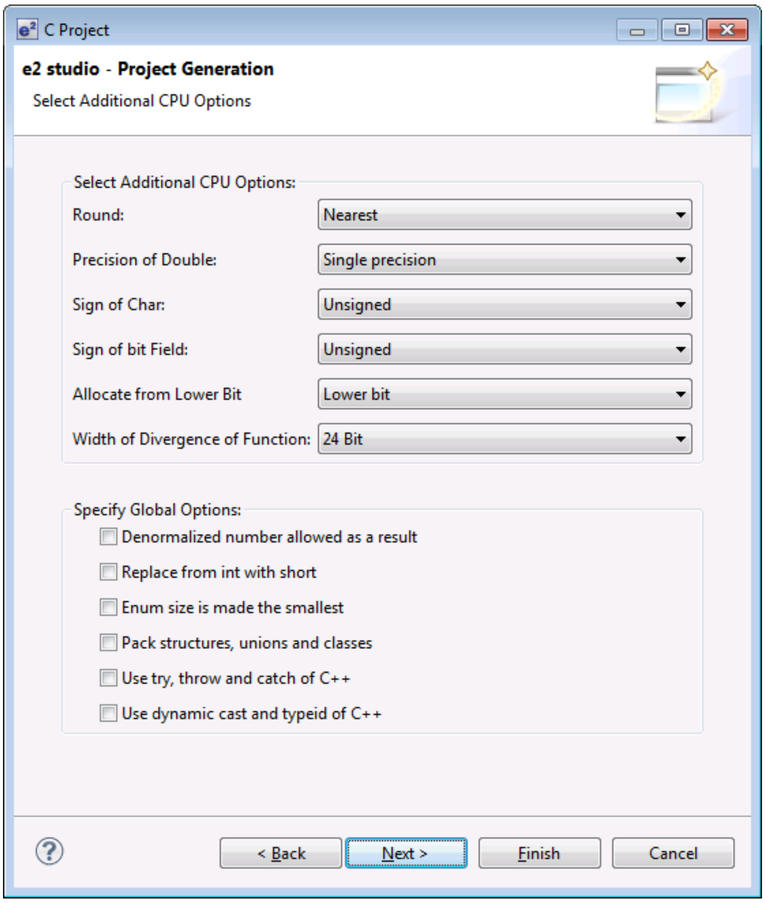
e2 studio - Project Generation

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.

- In ‘Select Additional CPU Options’ leave everything at default values.



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

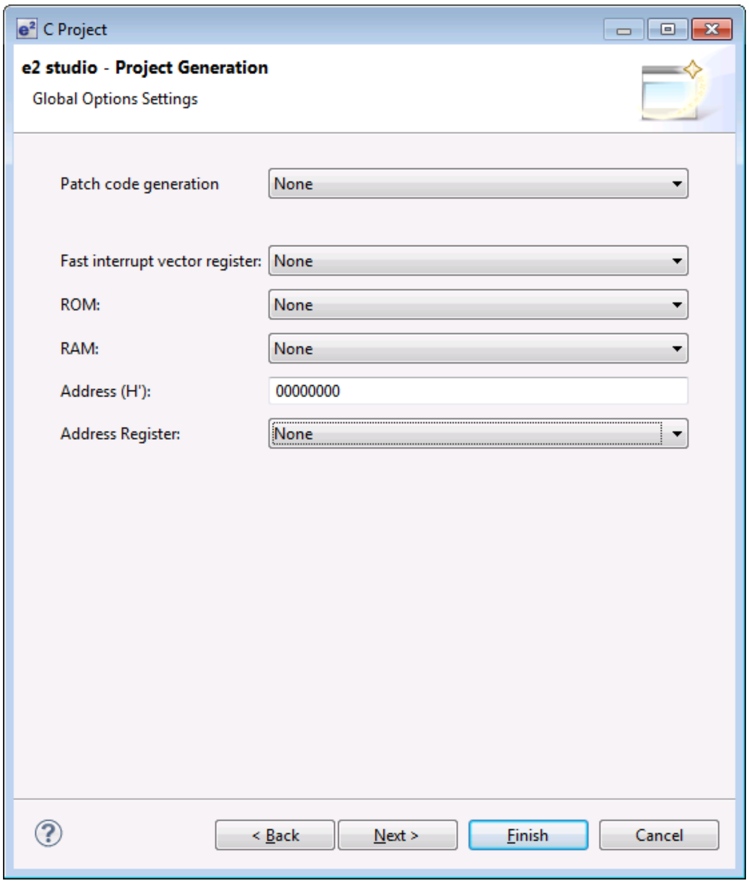
□ Enum size is made the smallest

□ Pack structures, unions and classes

□ Use try, throw and catch of C++

□ Use dynamic cast and typeid of C++

- In the ‘Global Options Settings’ leave everything at default values.



C Project

e2 studio - Project Generation

Global Options Settings

Patch code generation: None

Fast interrupt vector register: None

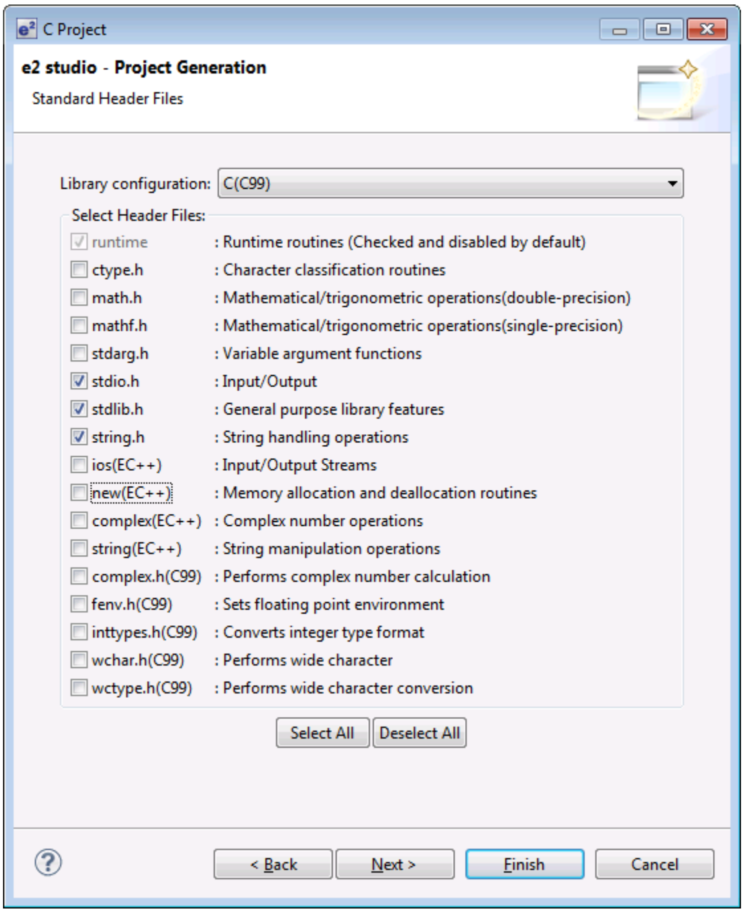
ROM: None

RAM: None

Address (H'): 00000000

Address Register: None

- In the ‘Standard Header Files’ dialog, select C99 for ‘Library Configuration’. Untick ‘new(EC++)’ and leave all others at defaults.
- ‘new(EC++) is the library for C++ style heap memory and is not used in this project.



C Project

e2 studio - Project Generation

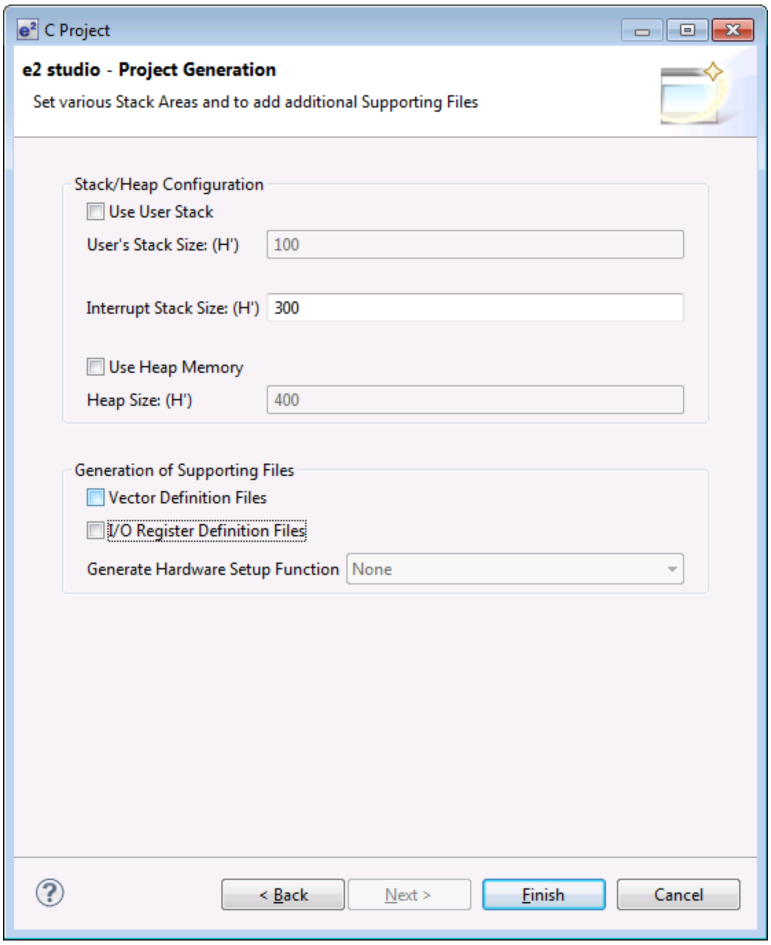
Standard Header Files

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

- In the next dialog, untick all check boxes as shown opposite. 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') 100

Interrupt Stack Size: (H') 300

□ Use Heap Memory

Heap Size: (H') 400

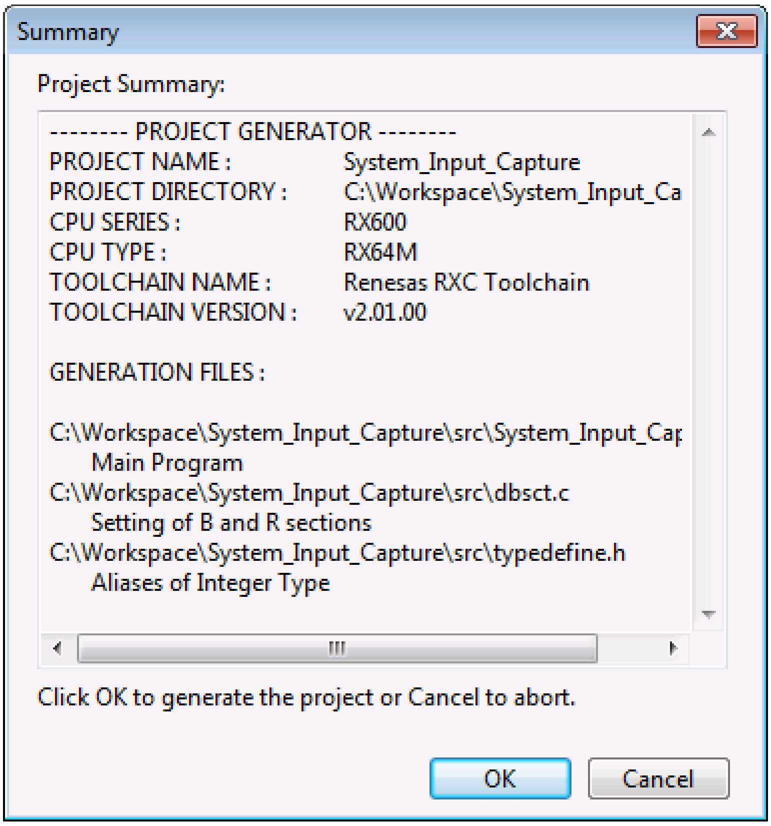
Generation of Supporting Files

□ Vector Definition Files

□ I/O Register Definition Files

Generate Hardware Setup Function None

- A summary dialog will appear, click ‘OK’ to complete the project generation.



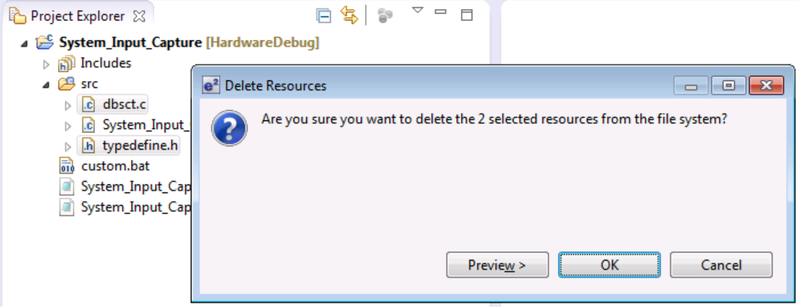
Summary

Project Summary:

```
-------- PROJECT GENERATOR --------
PROJECT NAME :          System_Input_Capture
PROJECT DIRECTORY :     C:\Workspace\System_Input_Ca
CPU SERIES :            RX600
CPU TYPE :              RX64M
TOOLCHAIN NAME :        Renesas RXC Toolchain
TOOLCHAIN VERSION :     v2.01.00

GENERATION FILES :

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

Click OK to generate the project or Cancel to abort.

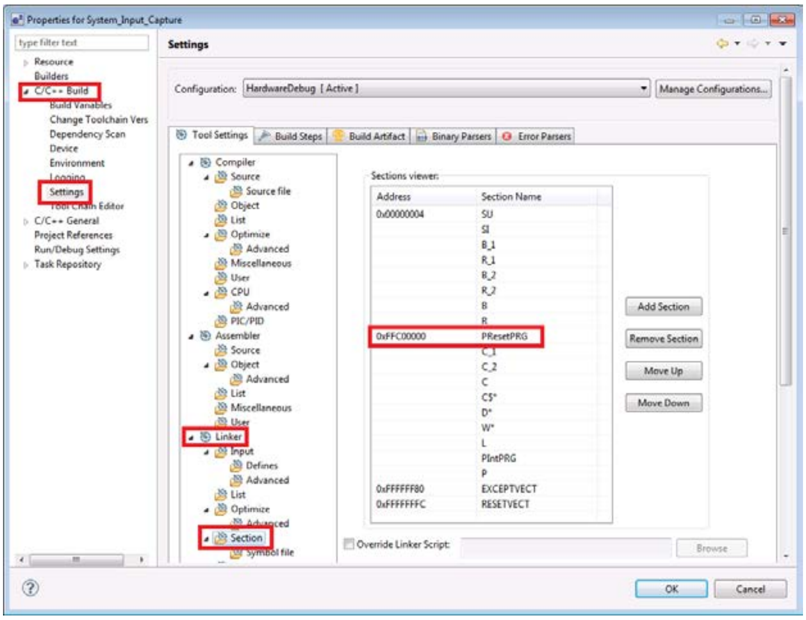
- In the Project Explorer pane, expand into the 'src' folder, select 'dbsct.c' and 'typedefine.h' and delete them from the project.



Delete Resources

Are you sure you want to delete the 2 selected resources from the file system?

- From the 'Project' menu, select 'Properties'. Expand 'C/C++ Build' and select 'Settings'. Under 'Tool Settings' select 'Linker -> Section'.
- Select the 'PResetPRG' section and click the 'Remove Section' button. The section C1 should now be assigned to address 0xFFC00000.
- Select the 'PIntPRG' section and click the 'Remove Section'
- Click on the 'P' section and change it to 'P*'. The use of the '*' character acts as a wildcard and will catch all 'P' sections used in the project.



Properties for System_Input_Capture

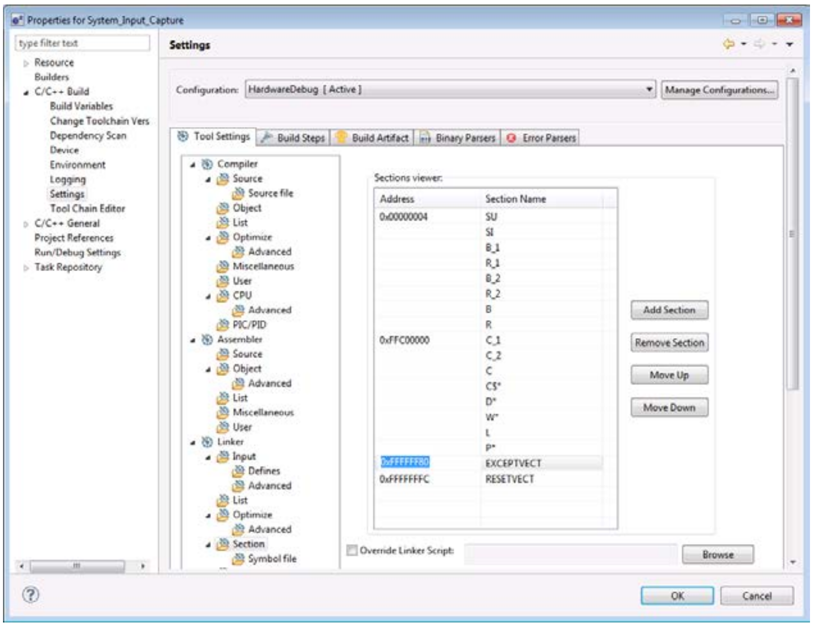
Configuration: HardwareDebug [Active]

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 'Apply'. The linker sections should be the same as shown in the screenshot opposite
- We will now setup the linker to fill in unused interrupt vectors with the address of the undefined_interrupt_source_isr() function.



Properties for System_Input_Capture

Configuration: HardwareDebug [Active]

Sections viewer:

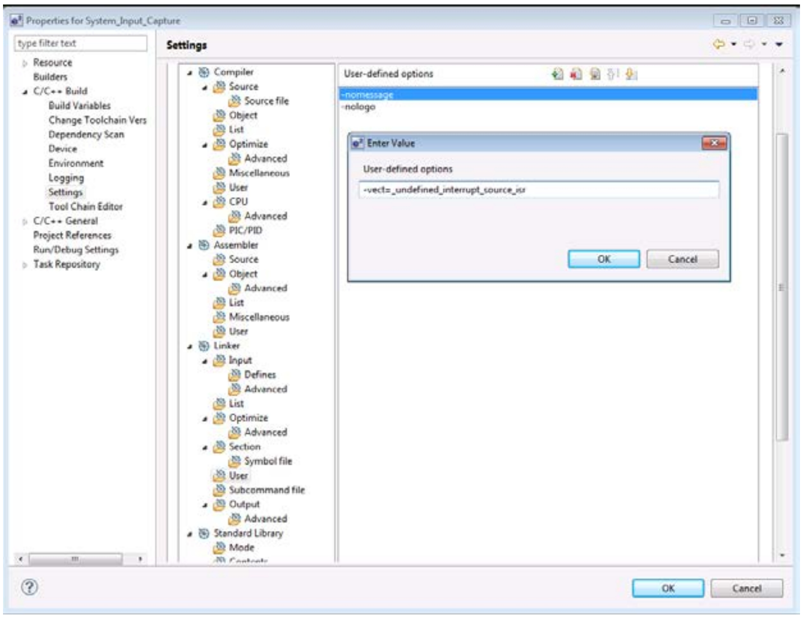
Address	Section Name
0x00000004	SU
	SI
	B_1
	R_1
	B_2
	R_2
	B
	R
0xFFC00000	C_1
	C_2
	C
	C$*
	D*
	W*
	L
	P*
0xFFFFFF80	EXCEPTVECT
0xFFFFFFFC	RESETVECT

□ Override Linker Script:

- Under 'Tool Settings' select 'Linker -> User'.
- Click the 'Add' button (with green '+' symbol) and in the window that pops up enter:

-vect=_undefined_interrupt_source_isr

- Click 'OK'. In the 'Properties' dialog, click 'OK'. The project is now ready to import the BSP.



User-defined options
-nomessage
-nologo

Enter Value

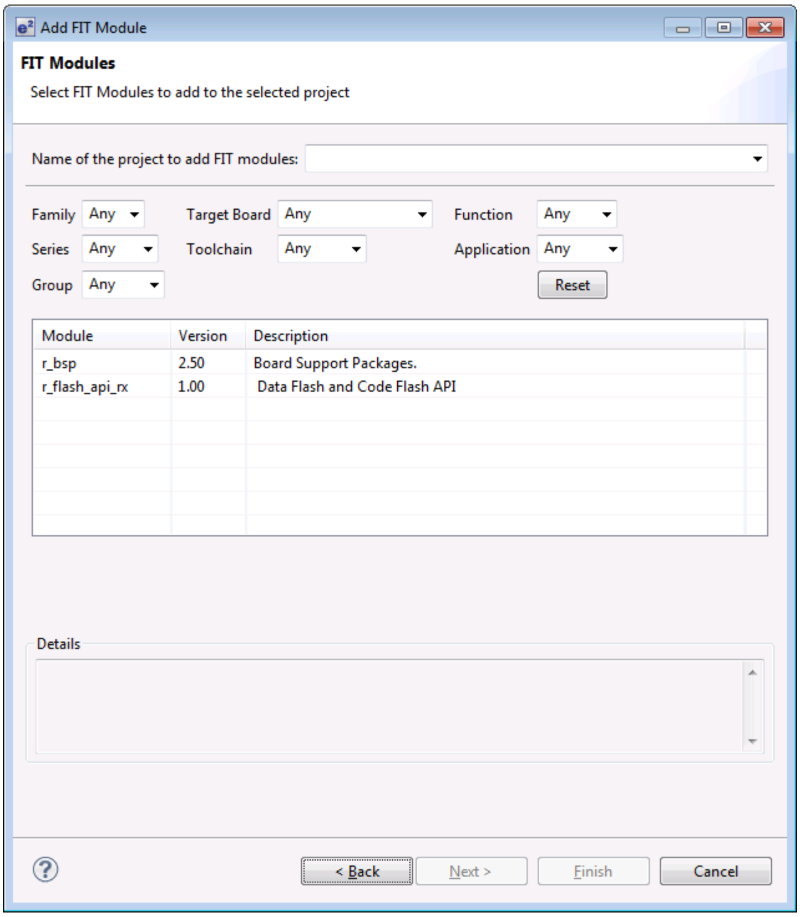
User-defined options: -vect=_undefined_interrupt_source_isr

OK Cancel

3.2 Importing the BSP into the Project

The two FIT modules required for this Tutorial have been installed with e^2 studio.

- From the 'File' menu, select 'New -> Renesas FIT Module'. The dialog opposite will be displayed.
- In the project name pull-down select 'System_Input_Capture'.
- In the 'Family' pull-down select 'RX'.
- In the 'Target Board' pull-down select 'RSKRX64M'.
- In the 'Series' pull-down select 'RX600'.
- In the 'Group' pull-down select 'RX64M'.



Add FIT Module

FIT Modules

Select FIT Modules to add to the selected project

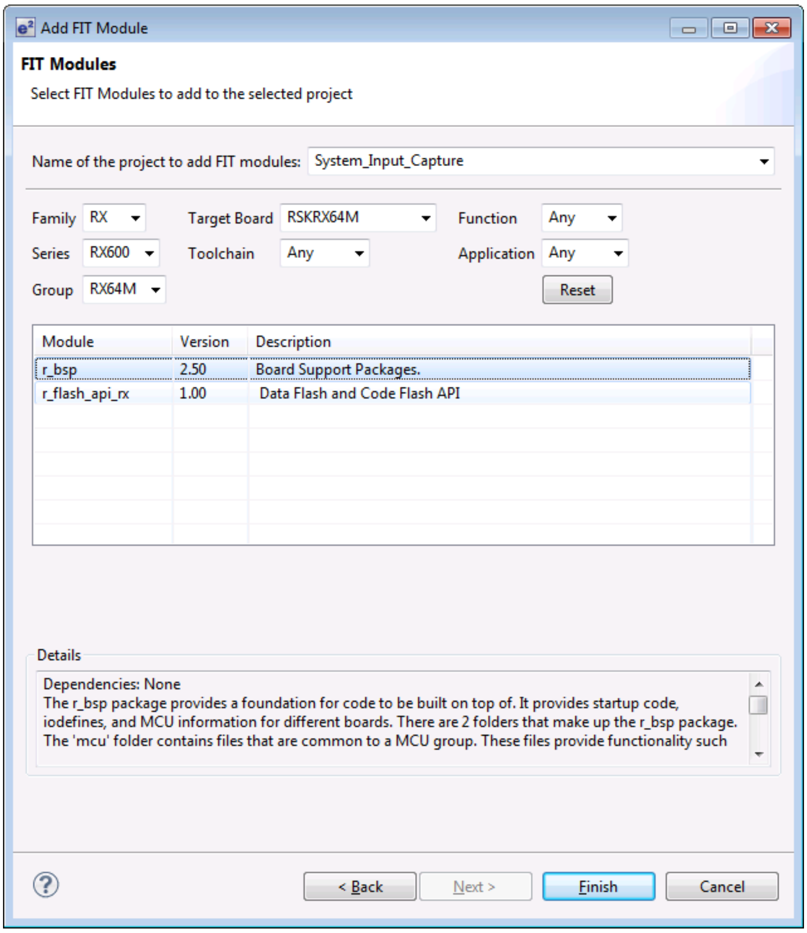
Name of the project to add FIT modules:

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

Module	Version	Description
r_bsp	2.50	Board Support Packages.
r_flash_api_rx	1.00	Data Flash and Code Flash API

Details

- In the main list view select the ‘r_bsp’ module, the ‘Finish’ button will then be enabled. Verify that the dialog is as shown opposite.
- Click ‘Finish’.



Add FIT Module

FIT Modules

Select FIT Modules to add to the selected project

Name of the project to add FIT modules: System_Input_Capture

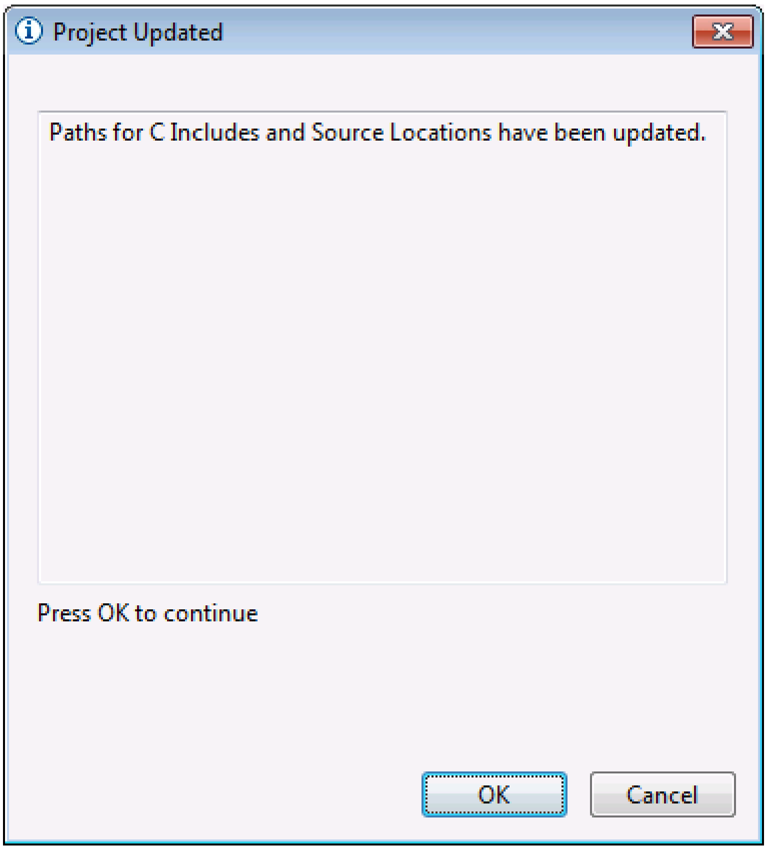
Family: RX
Series: RX600
Group: RX64M
Target Board: RSKRX64M
Toolchain: Any
Function: Any
Application: Any

Module	Version	Description
r_bsp	2.50	Board Support Packages.
r_flash_api_rx	1.00	Data Flash and Code Flash API

Details

Dependencies: None
The r_bsp package provides a foundation for code to be built on top of. It provides startup code, iodefines, and MCU information for different boards. There are 2 folders that make up the r_bsp package. The 'mcu' folder contains files that are common to a MCU group. These files provide functionality such

- A ‘Project Updated’ dialog will appear, indicating that include paths and source locations have been updated. Click ‘OK’.

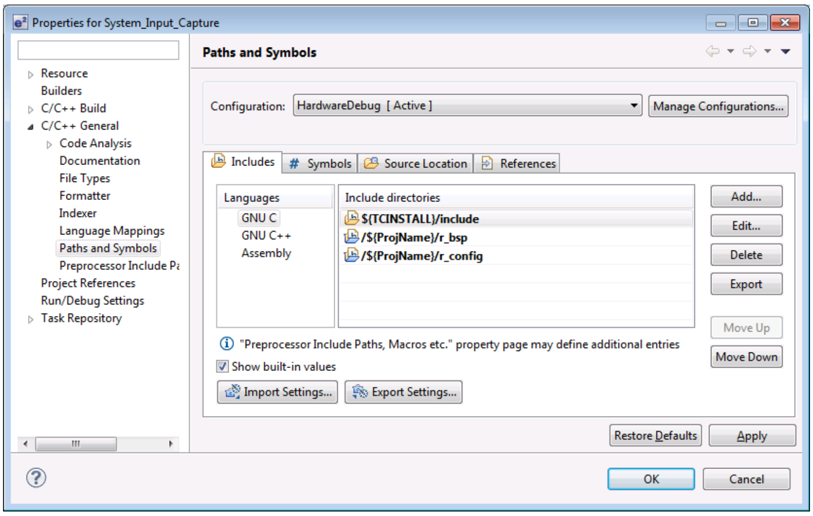


Project Updated

Paths for C Includes and Source Locations have been updated.

Press OK to continue

- The newly added paths and symbols will be displayed in a Properties dialog. Click 'OK'



Properties for System_Input_Capture

Paths and Symbols

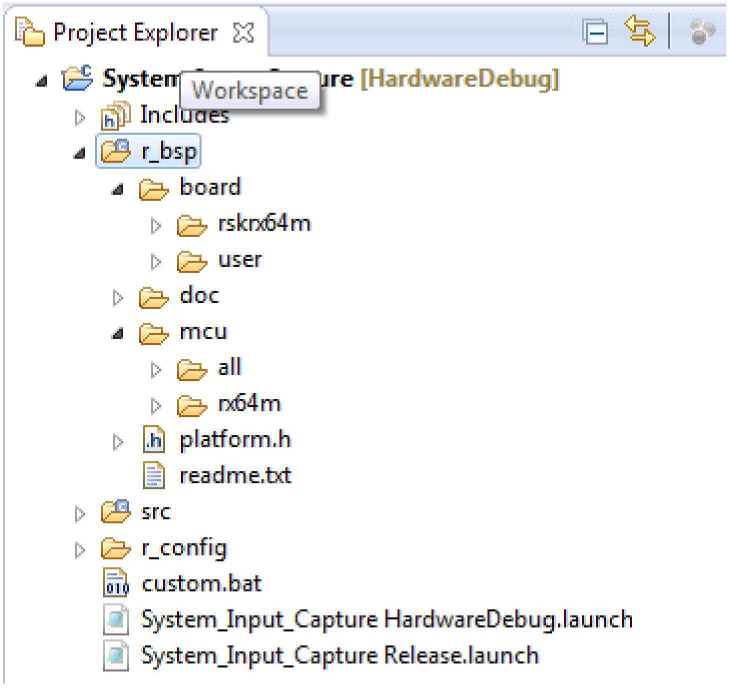
Configuration: HardwareDebug [Active]

Include directories:
- ${TCINSTALL}/include
- /${ProjName}/r_bsp
- /${ProjName}/r_config

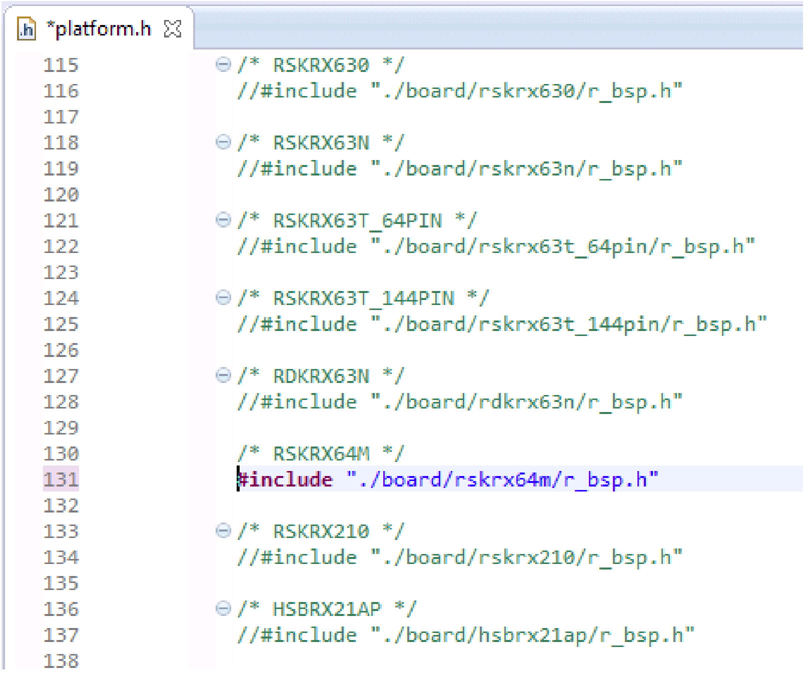
"Preprocessor Include Paths, Macros etc." property page may define additional entries

☑ Show built-in values

- In the Project Explorer pane, expand the 'r_bsp' folder and 'board' and 'mcu' folders.
- Verify that the 'rskrx64m' folder has been added to the 'board' folder.
- Verify that the 'rx64m' folder has been added to the 'mcu' folder.

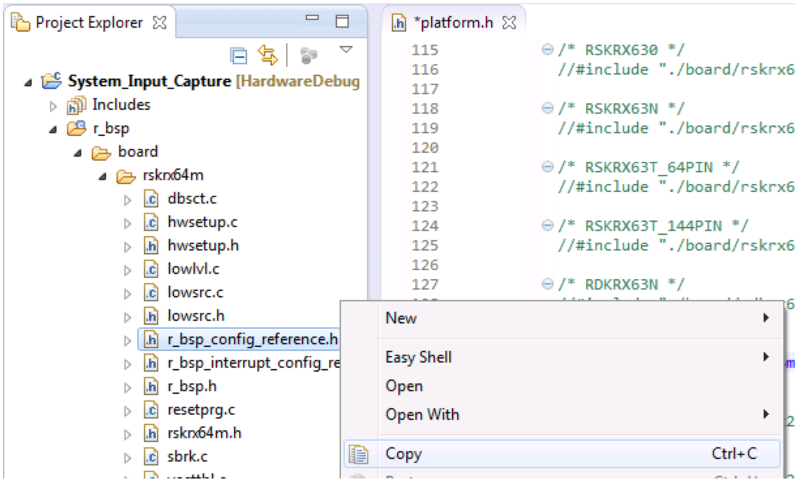


- In the Project Explorer pane, double-click the 'platform.h' header file icon to open the file in the edit pane
- Scroll down to the line shown opposite and uncomment the include directive for the RSK+RX64M board.



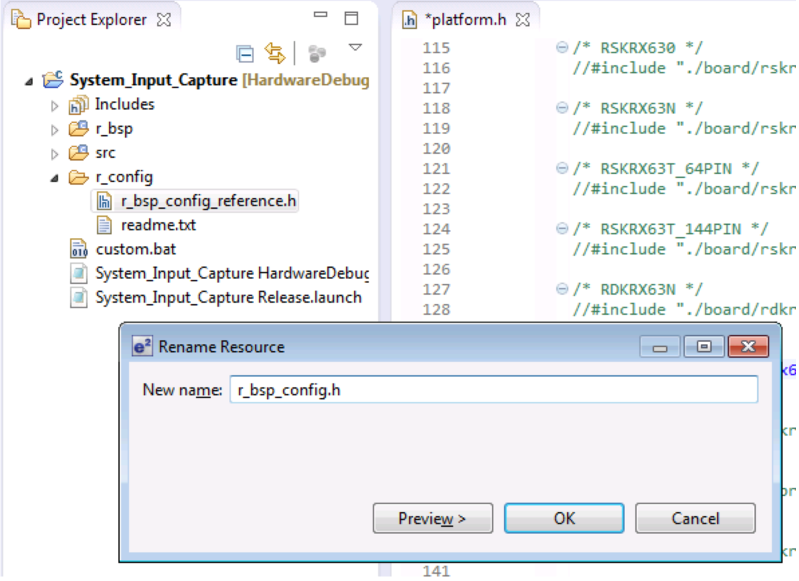
```
/* RSKRX630 */
//#include "./board/rskrx630/r_bsp.h"

/* RSKRX63N */
//#include "./board/rskrx63n/r_bsp.h"

/* 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"

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

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

/* HSBRX21AP */
//#include "./board/hsbrx21ap/r_bsp.h"
```

- The FIT project folder structure includes an 'r_config' folder, where each FIT module's configuration header file resides. The 'r_bsp' FIT module contains an 'r_bsp_config_reference.h' file which we will need to copy into the 'r_config' folder.
- Copy and paste this file into the 'r_config' folder as shown opposite.



```
/* RSKRX630 */
//#include "./board/rskrx6
/* RSKRX63N */
//#include "./board/rskrx6
/* RSKRX63T_64PIN */
//#include "./board/rskrx6
/* RSKRX63T_144PIN */
//#include "./board/rskrx6
/* RDKRX63N */
```

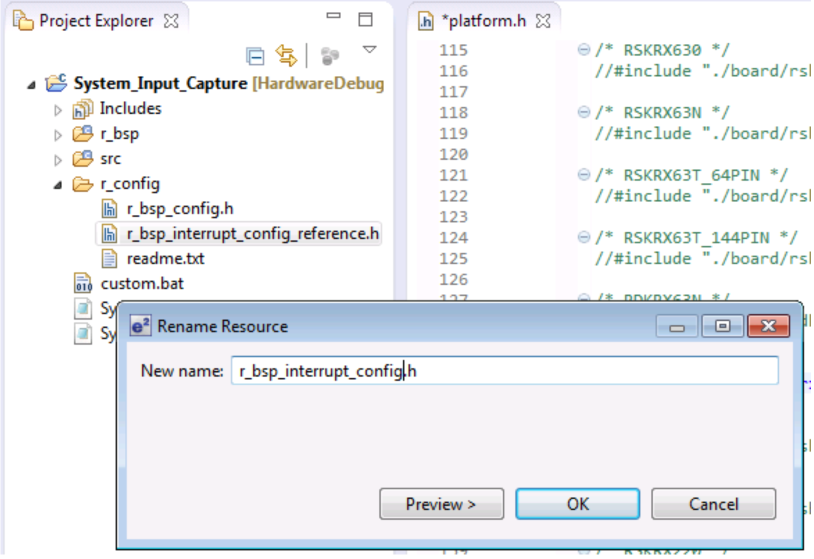
- Right-click the newly copied 'r_bsp_config_reference.h' file and select 'Rename..' from the context menu. Rename the resource to 'r_bsp_config.h' as shown opposite. Click 'OK'.
- The 'r_bsp_config.h' file is used to configure the BSP as described in Section 3.2 of r01an1685eu. For the purposes of this project, we will leave the file unchanged.



New name: r_bsp_config.h

```
/* RSKRX630 */
//#include "./board/rskr
/* RSKRX63N */
//#include "./board/rskr
/* RSKRX63T_64PIN */
//#include "./board/rskr
/* RSKRX63T_144PIN */
//#include "./board/rskr
/* RDKRX63N */
//#include "./board/rdkr
```

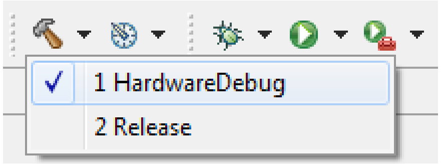
- Repeat the above steps for the 'r_bsp_interrupt_config_reference.h' file, found in the 'r_bsp -> board -> rskrx64m' folder
- The 'r_bsp_interrupt_config.h' file is used to configure the BSP interrupts. For the purposes of this project, we will leave the file unchanged.



```
/* RSKRX630 */
//#include "./board/rsl
/* RSKRX63N */
//#include "./board/rsl
/* RSKRX63T_64PIN */
//#include "./board/rsl
/* RSKRX63T_144PIN */
//#include "./board/rsl
```

New name: r_bsp_interrupt_config.h

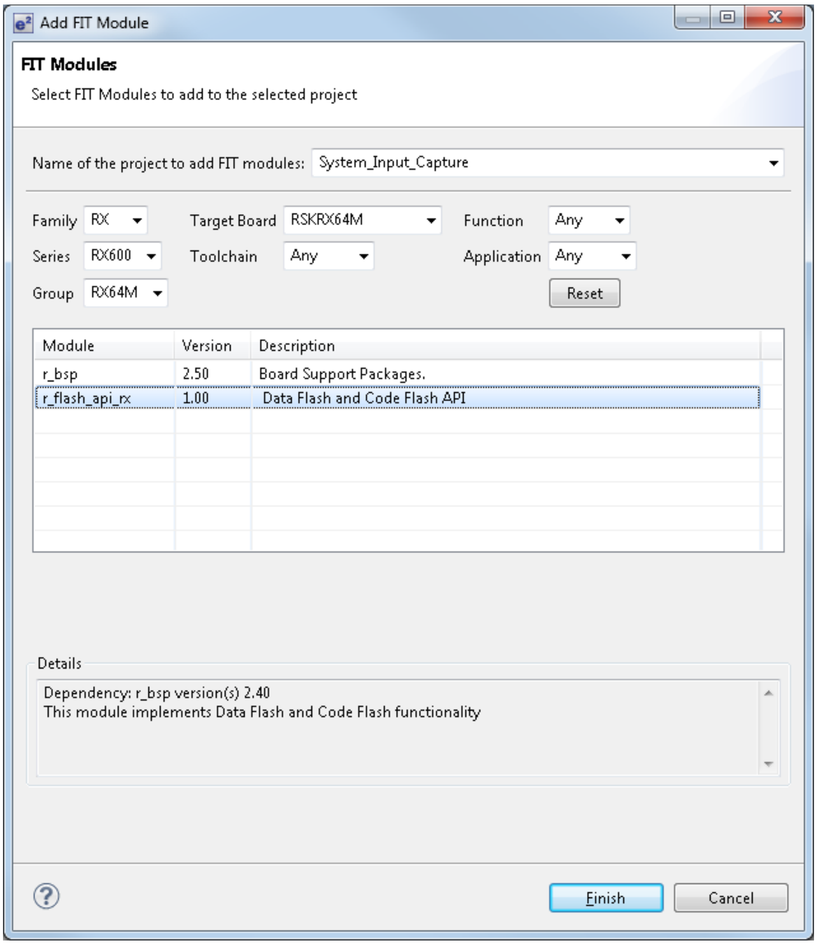
- Click the top level 'System_Input_Capture' folder again, and then the arrow next to the build button (hammer icon), and select the 'HardwareDebug' option.



- e^2 studio will now build the code.

4.Importing the Flash Library FIT Module

- From the 'File' menu, select 'New -> Renesas FIT Module'. The dialog opposite will be displayed.
- The dialog will be populated with the previous selections from §3.2.
- In the main list view select the 'r_flash_api_rx' module, the 'Finish' button will then be enabled. Verify that the dialog is as shown opposite.
- Click 'Finish'.



FIT Modules

Select FIT Modules to add to the selected project

Name of the project to add FIT modules: System_Input_Capture

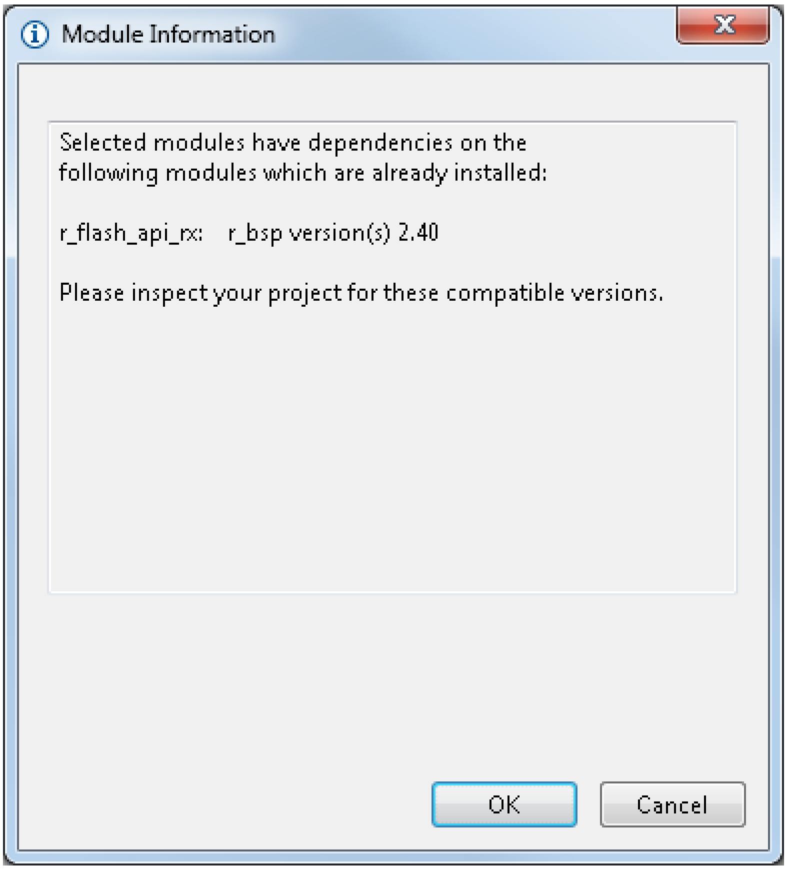
Family: RX
Series: RX600
Group: RX64M
Target Board: RSKRX64M
Toolchain: Any
Function: Any
Application: Any

Module	Version	Description
r_bsp	2.50	Board Support Packages.
r_flash_api_rx	1.00	Data Flash and Code Flash API

Details

Dependency: r_bsp version(s) 2.40
This module implements Data Flash and Code Flash functionality

- A 'Module Information' dialog will appear, indicating that the Flash API FIT module has a dependency on the BSP FIT module. Click 'OK'.

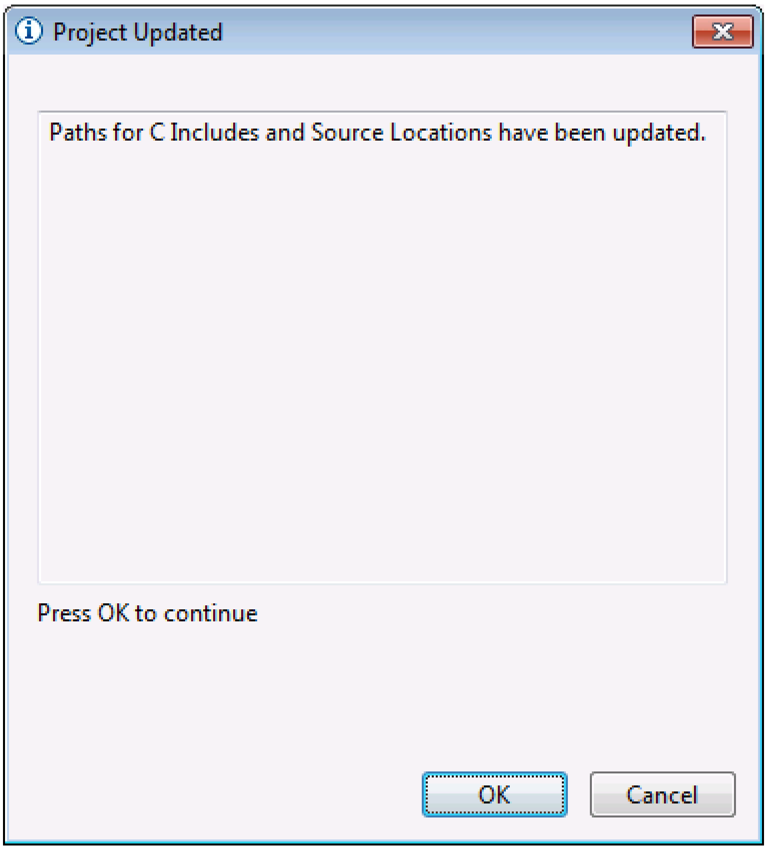


Selected modules have dependencies on the following modules which are already installed:

r_flash_api_rx: r_bsp version(s) 2.40

Please inspect your project for these compatible versions.

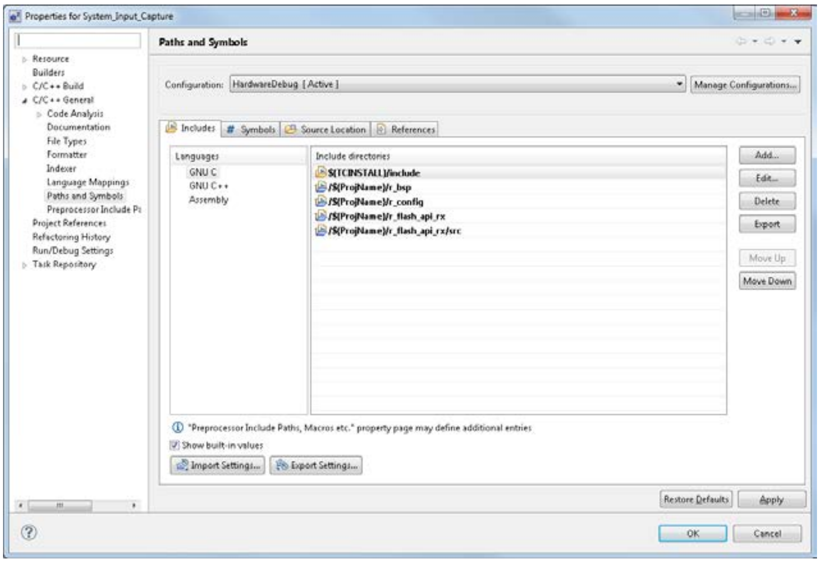
- A 'Project Updated' dialog will appear, indicating that include paths and source locations have been updated. Click 'OK'.



Paths for C Includes and Source Locations have been updated.

Press OK to continue

- The newly added paths and symbols will be displayed in a Properties dialog. Click ‘OK’



Properties for System_Input_Capture

Paths and Symbols

Configuration: HardwareDebug [Active]

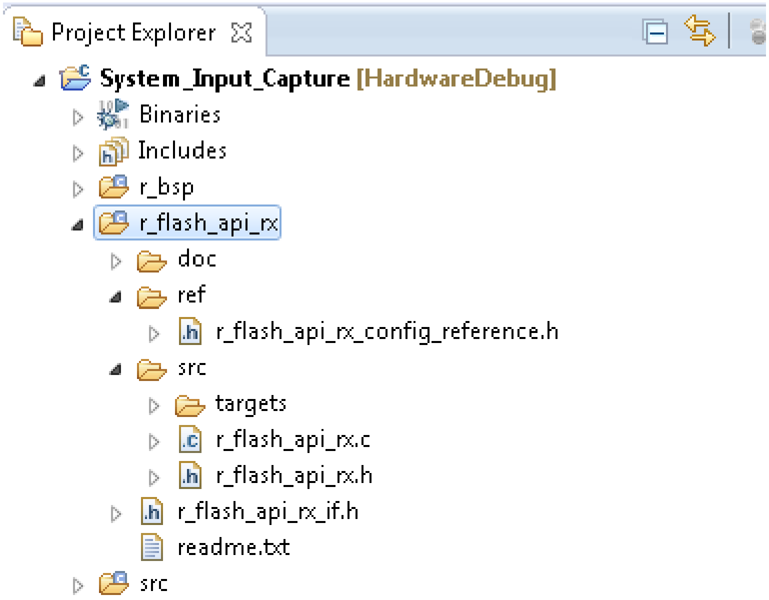
Languages: GNU C, GNU C++, Assembly

Include directories
$(TCINSTALL)/include
/${ProjName}/r_bsp
/${ProjName}/r_config
/${ProjName}/r_flash_api_rx
/${ProjName}/r_flash_api_rx/src

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

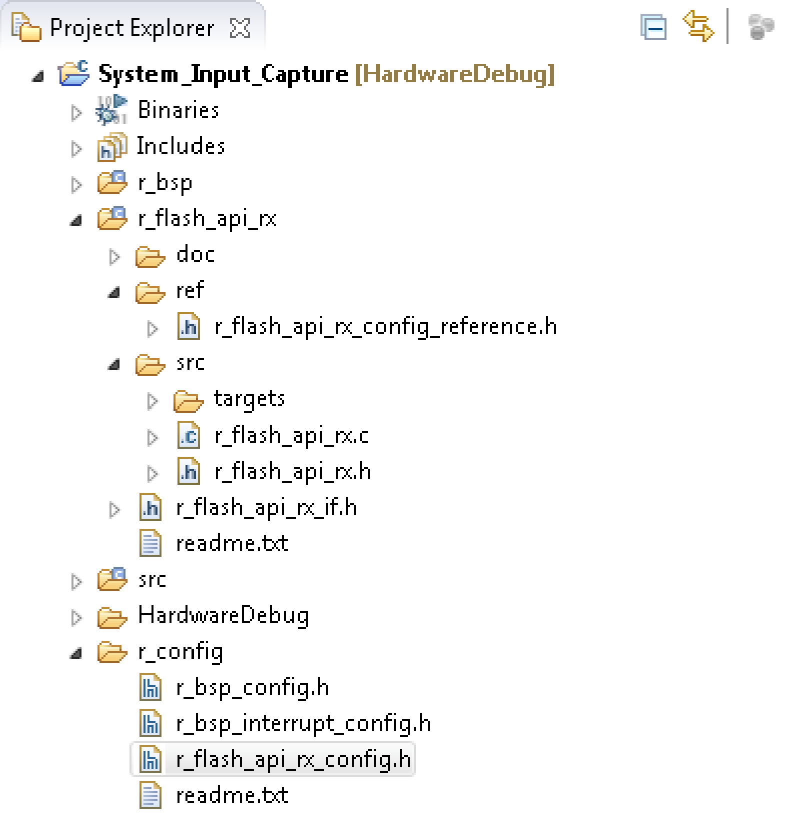
☑ Show built-in values

- In the Project Explorer pane, expand the ‘r_flash_api_rx’ folder and ‘ref’ and ‘src’ folders.
- The screenshot opposite shows the Flash API FIT module files have been installed correctly.



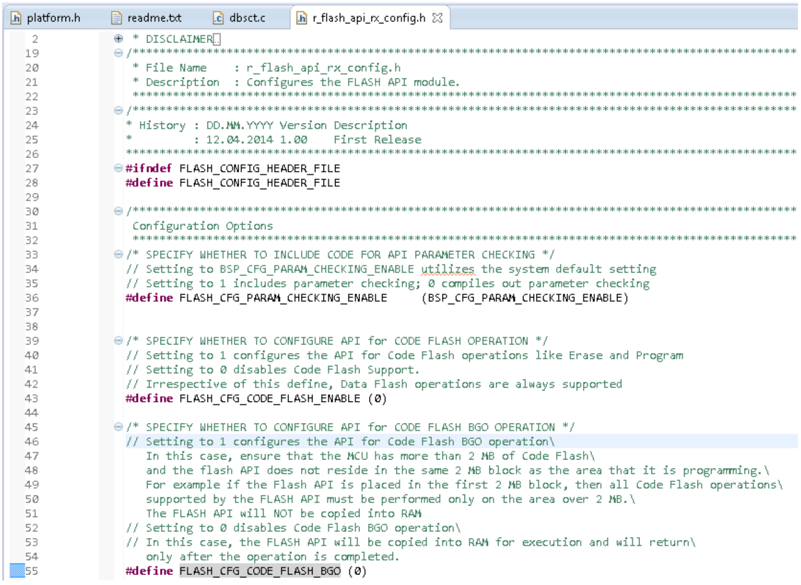
```
System_Input_Capture [HardwareDebug]
  Binaries
  Includes
  r_bsp
  r_flash_api_rx
    doc
    ref
      r_flash_api_rx_config_reference.h
    src
      targets
      r_flash_api_rx.c
      r_flash_api_rx.h
    r_flash_api_rx_if.h
    readme.txt
  src
```

- The ‘r_flash_api_rx’ FIT module installer has placed a ‘r_flash_api_rx_config.h’ file in the ‘r_config’ folder.



```
Project Explorer
System_Input_Capture [HardwareDebug]
    Binaries
    Includes
    r_bsp
    r_flash_api_rx
        doc
        ref
            r_flash_api_rx_config_reference.h
        src
            targets
            r_flash_api_rx.c
            r_flash_api_rx.h
        r_flash_api_rx_if.h
        readme.txt
    src
    HardwareDebug
    r_config
        r_bsp_config.h
        r_bsp_interrupt_config.h
        r_flash_api_rx_config.h
        readme.txt
```

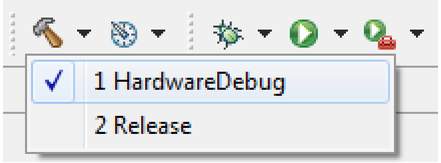
- Double-click this file and set the #defines for FLASH_CFG_CODE_FLASH_ENABLE and FLASH_CFG_CODE_FLASH_BGO to 0 as shown opposite.
- Refer to r01an2072 for details of these configuration options.



platform.h | readme.txt | dbsct.c | r_flash_api_rx_config.h

```
 * DISCLAIMER
/*********************************************************************************************
 * File Name    : r_flash_api_rx_config.h
 * Description  : Configures the FLASH API module.
 *********************************************************************************************
/*********************************************************************************************
* History : DD.MM.YYYY Version Description
*         : 12.04.2014 1.00    First Release
**********************************************************************************************
#ifndef FLASH_CONFIG_HEADER_FILE
#define FLASH_CONFIG_HEADER_FILE

/*********************************************************************************************
 Configuration Options
 *********************************************************************************************
/* SPECIFY WHETHER TO INCLUDE CODE FOR API PARAMETER CHECKING */
// Setting to BSP_CFG_PARAM_CHECKING_ENABLE utilizes the system default setting
// Setting to 1 includes parameter checking; 0 compiles out parameter checking
#define FLASH_CFG_PARAM_CHECKING_ENABLE     (BSP_CFG_PARAM_CHECKING_ENABLE)


/* SPECIFY WHETHER TO CONFIGURE API for CODE FLASH OPERATION */
// Setting to 1 configures the API for Code Flash operations like Erase and Program
// Setting to 0 disables Code Flash Support.
// Irrespective of this define, Data Flash operations are always supported
#define FLASH_CFG_CODE_FLASH_ENABLE (0)

/* SPECIFY WHETHER TO CONFIGURE API for CODE FLASH BGO OPERATION */
// Setting to 1 configures the API for Code Flash BGO operation\
   In this case, ensure that the MCU has more than 2 MB of Code Flash\
   and the flash API does not reside in the same 2 MB block as the area that it is programming.\
   For example if the Flash API is placed in the first 2 MB block, then all Code Flash operations\
   supported by the FLASH API must be performed only on the area over 2 MB.\
   The FLASH API will NOT be copied into RAM
// Setting to 0 disables Code Flash BGO operation\
// In this case, the FLASH API will be copied into RAM for execution and will return\
   only after the operation is completed.
#define FLASH_CFG_CODE_FLASH_BGO (0)
```

- Click the top level ‘System_Input_Capture’ folder again, and then the arrow next to the build button (hammer icon), and select the ‘HardwareDebug’ option.



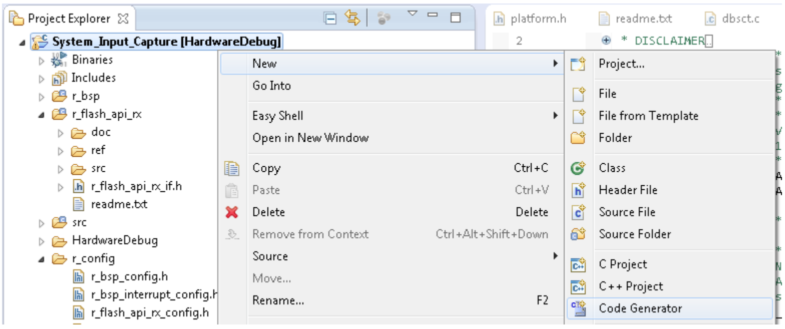
- e^2 studio will now build the code.

5.Using the Code Generator for Peripheral Functions

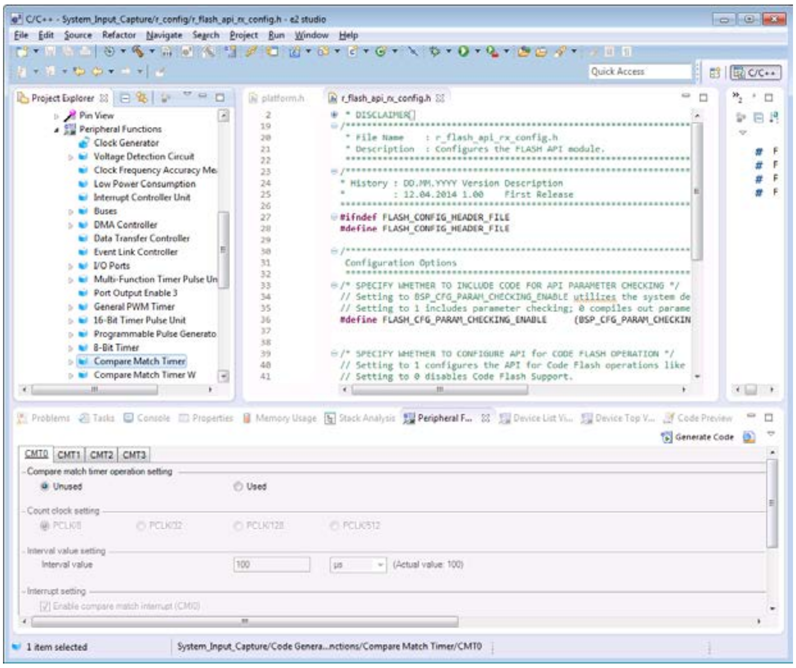
The System_Input_Capture sample applications uses a number of peripheral functions of the RX64M MCU. The drivers for each of these peripheral functions are created by the in-built Code Generator for e^2 studio. The Code Generator is normally used to create a stand-alone project from scratch. Since we have created this project using a FIT BSP, it is necessary to exclude some of the code generator start-up files from the build and manually add some include paths in order to properly integrate the code generator files within the FIT software architecture.

In this section of the Tutorial, the steps required to generate the peripheral drivers required for the sample are presented, together with the additional steps required to build the project.

- Right-click the top level 'System_Input_Capture' and select 'New -> Code Generator'.



- After a few seconds, a 'Code Generator' node will appear in the project explorer pane.
- Expand the 'Code Generator -> Peripheral Functions' node, and double-click the Compare Match Timer node.
- The 'Peripheral Functions' tab will be activated in the Build Pane as shown opposite.



```
* DISCLAIMER
/*****************************************************************
 * File Name    : r_flash_api_rx_config.h
 * Description  : Configures the FLASH API module.
 *****************************************************************
/*****************************************************************
* History : DD.MM.YYYY Version Description
*         : 12.04.2014 1.00    First Release
******************************************************************
#ifndef FLASH_CONFIG_HEADER_FILE
#define FLASH_CONFIG_HEADER_FILE

/*****************************************************************
 Configuration Options
 *****************************************************************
/* SPECIFY WHETHER TO INCLUDE CODE FOR API PARAMETER CHECKING */
// Setting to BSP_CFG_PARAM_CHECKING_ENABLE utilizes the system de
// Setting to 1 includes parameter checking; 0 compiles out parame
#define FLASH_CFG_PARAM_CHECKING_ENABLE     (BSP_CFG_PARAM_CHECKIN


/* SPECIFY WHETHER TO CONFIGURE API for CODE FLASH OPERATION */
// Setting to 1 configures the API for Code Flash operations like
// Setting to 0 disables Code Flash Support.
```

CMT0 CMT1 CMT2 CMT3

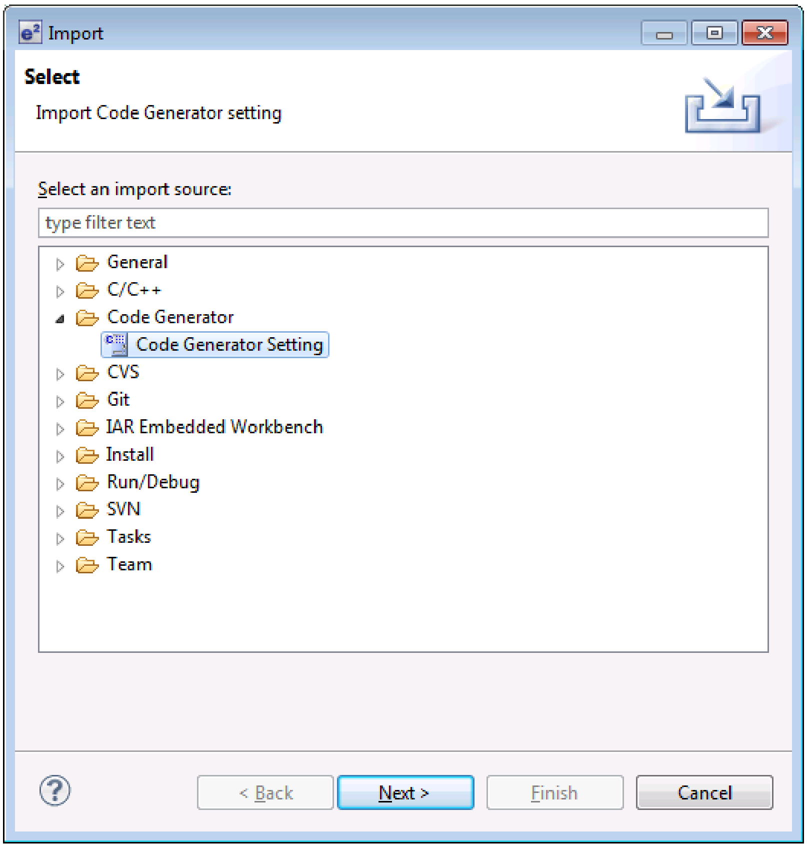
Compare match timer operation setting
● Unused ○ Used

Count clock setting
● PCLK/8 ○ PCLK/32 ○ PCLK/128 ○ PCLK/512

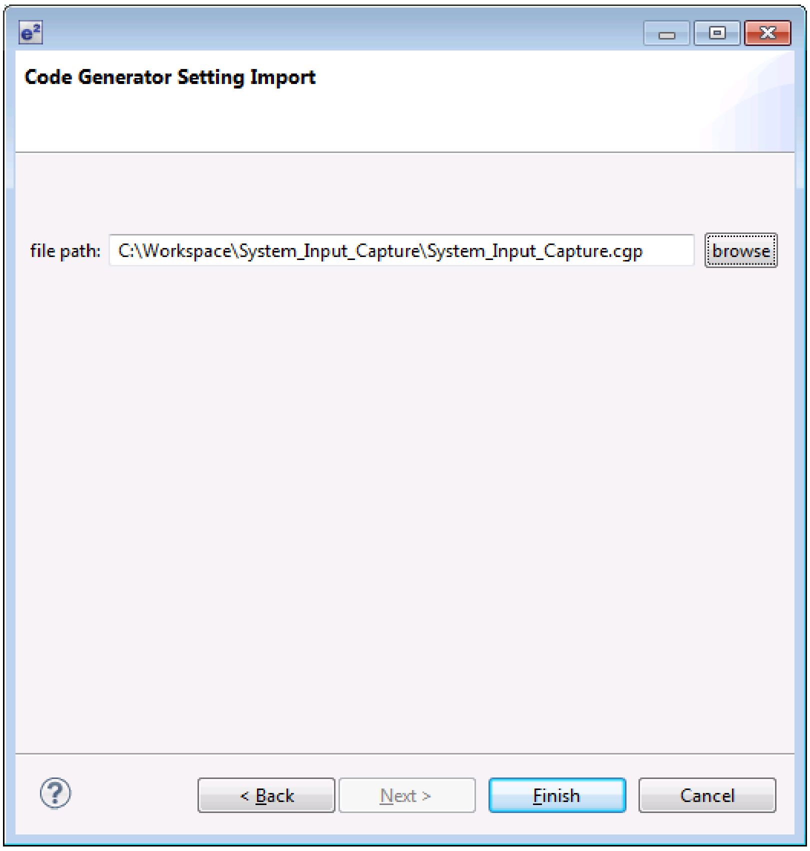
Interval value setting
Interval value 100 μs (Actual value: 100)

Interrupt setting
☑ Enable compare match interrupt (CMI0)

- Right-click the top level 'System_Input_Capture' and select 'Import'.
- In the dialog that appears, expand the 'Code Generator' node and select 'Code Generator Setting'. Click 'Next'.

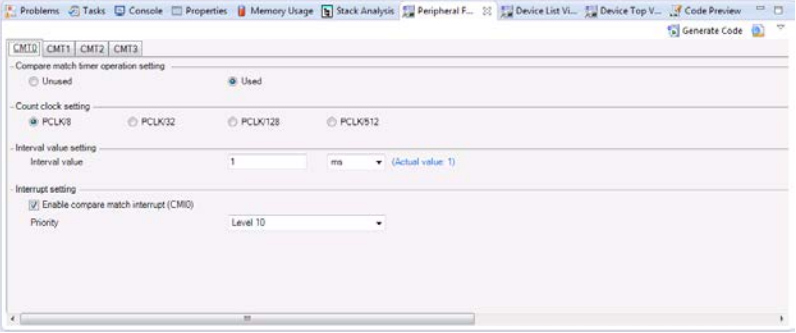


- The dialog shown opposite will be displayed.
- The file 'System_Input_Capture.cgp' is provided on the DVD and has been installed with the sample project in the directory 'System_Input_Capture'.
- Detailed use of the Code Generator GUI to configure MCU peripherals is outside the scope of this document, since it is covered in the Code Generator Tutorial r20ut2930.
- Locate this file and copy it into the local workspace for this project.
- Browse to this file using the 'browse' button, then click 'Finish'.
- In the warning dialog, click 'OK'.



file path: C:\Workspace\System_Input_Capture\System_Input_Capture.cgp

- Expand the ‘Code Generator -> Peripheral Functions’ node, and double-click the Compare Match Timer node again.
- Notice how the CMT settings have been imported for CMT0-3.
- Settings for the Compare Match Timer, Realtime Clock, Serial Communications Interface, I2C Bus Interface and 12-bit A/D Converter have been imported.



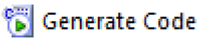
Compare match timer operation setting
○ Unused ● Used

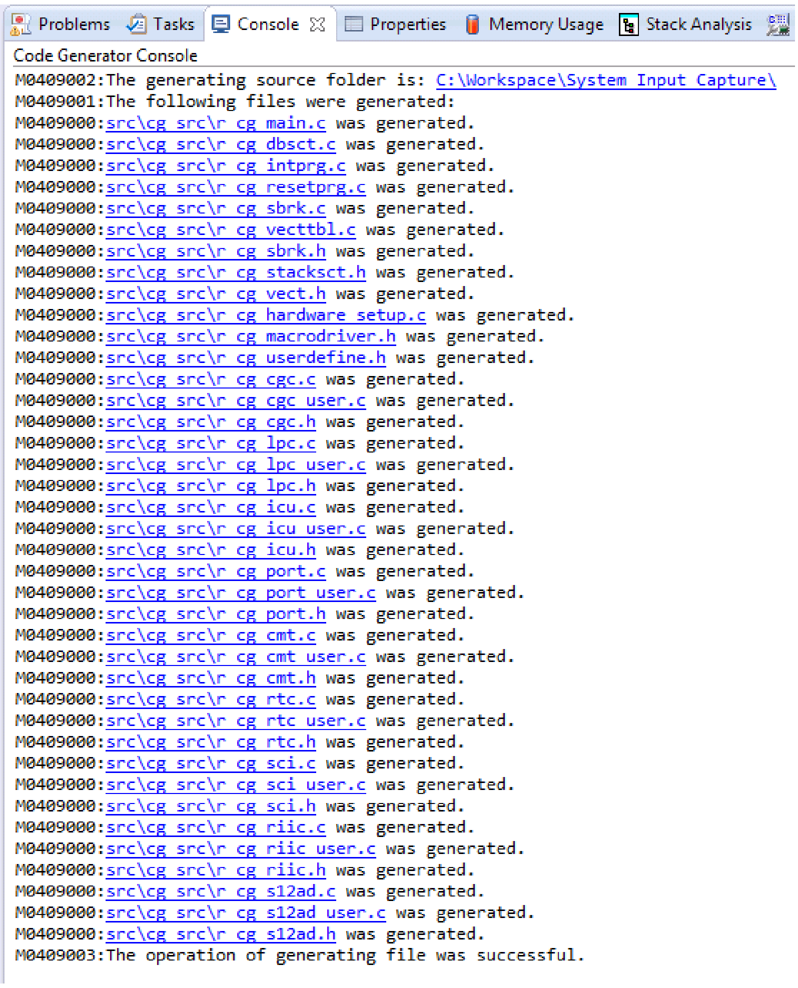
Count clock setting
● PCLK/8 ○ PCLK/32 ○ PCLK/128 ○ PCLK/512

Interval value setting
Interval value 1 ms (Actual value: 1)

Interrupt setting
☑ Enable compare match interrupt (CMI0)
Priority Level 10

- In the right of the Peripheral Functions dialog, click ‘Generate Code’
- The ‘Console’ tab in the Build Pane will be updated.

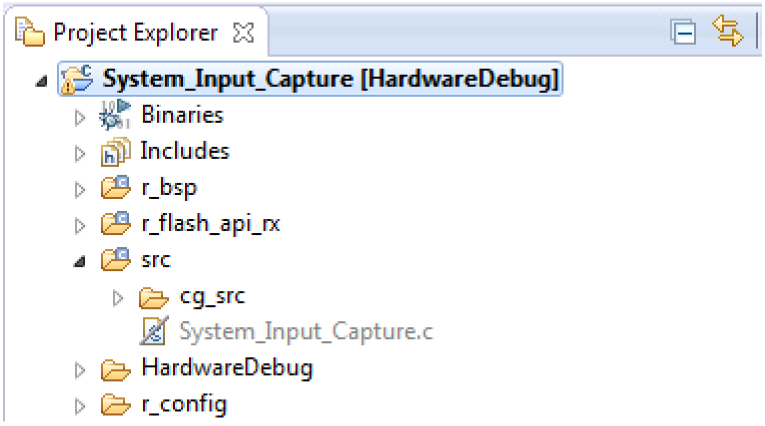




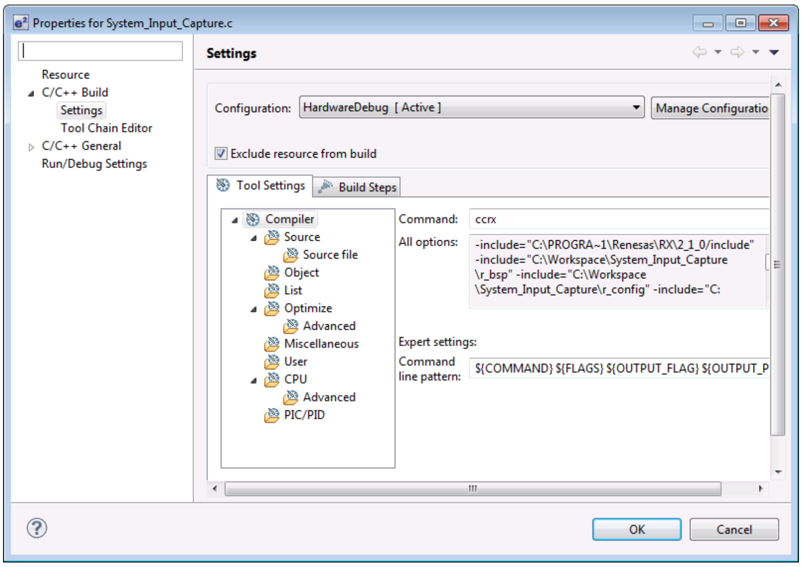
Code Generator Console

```
M0409002:The generating source folder is: C:\Workspace\System Input Capture\
M0409001:The following files were generated:
M0409000:src\cg src\r cg main.c was generated.
M0409000:src\cg src\r cg dbsct.c was generated.
M0409000:src\cg src\r cg intprg.c was generated.
M0409000:src\cg src\r cg resetprg.c was generated.
M0409000:src\cg src\r cg sbrk.c was generated.
M0409000:src\cg src\r cg vecttbl.c was generated.
M0409000:src\cg src\r cg sbrk.h was generated.
M0409000:src\cg src\r cg stacksct.h was generated.
M0409000:src\cg src\r cg vect.h was generated.
M0409000:src\cg src\r cg hardware setup.c was generated.
M0409000:src\cg src\r cg macrodriver.h was generated.
M0409000:src\cg src\r cg userdefine.h was generated.
M0409000:src\cg src\r cg cgc.c was generated.
M0409000:src\cg src\r cg cgc user.c was generated.
M0409000:src\cg src\r cg cgc.h was generated.
M0409000:src\cg src\r cg lpc.c was generated.
M0409000:src\cg src\r cg lpc user.c was generated.
M0409000:src\cg src\r cg lpc.h was generated.
M0409000:src\cg src\r cg icu.c was generated.
M0409000:src\cg src\r cg icu user.c was generated.
M0409000:src\cg src\r cg icu.h was generated.
M0409000:src\cg src\r cg port.c was generated.
M0409000:src\cg src\r cg port user.c was generated.
M0409000:src\cg src\r cg port.h was generated.
M0409000:src\cg src\r cg cmt.c was generated.
M0409000:src\cg src\r cg cmt user.c was generated.
M0409000:src\cg src\r cg cmt.h was generated.
M0409000:src\cg src\r cg rtc.c was generated.
M0409000:src\cg src\r cg rtc user.c was generated.
M0409000:src\cg src\r cg rtc.h was generated.
M0409000:src\cg src\r cg sci.c was generated.
M0409000:src\cg src\r cg sci user.c was generated.
M0409000:src\cg src\r cg sci.h was generated.
M0409000:src\cg src\r cg riic.c was generated.
M0409000:src\cg src\r cg riic user.c was generated.
M0409000:src\cg src\r cg riic.h was generated.
M0409000:src\cg src\r cg s12ad.c was generated.
M0409000:src\cg src\r cg s12ad user.c was generated.
M0409000:src\cg src\r cg s12ad.h was generated.
M0409003:The operation of generating file was successful.
```

- In the Project Explorer pane, expand the ‘src’ folder node.
- The System_Input_Capture.c’ file has been excluded from the build as part of the code generation operation.
- All of the Code Generator files have been placed inside a new sub-folder ‘cg_src’, under the ‘src’ folder.



- Right-click the greyed-out ‘System_Input_Capture.c’ node and select ‘Properties’.
- Select ‘C/C++ Build’ and untick the ‘Exclude resource from build’, check box. Click ‘OK’.


Properties for System_Input_Capture.c

Settings

Configuration: HardwareDebug [Active]

☑ Exclude resource from build

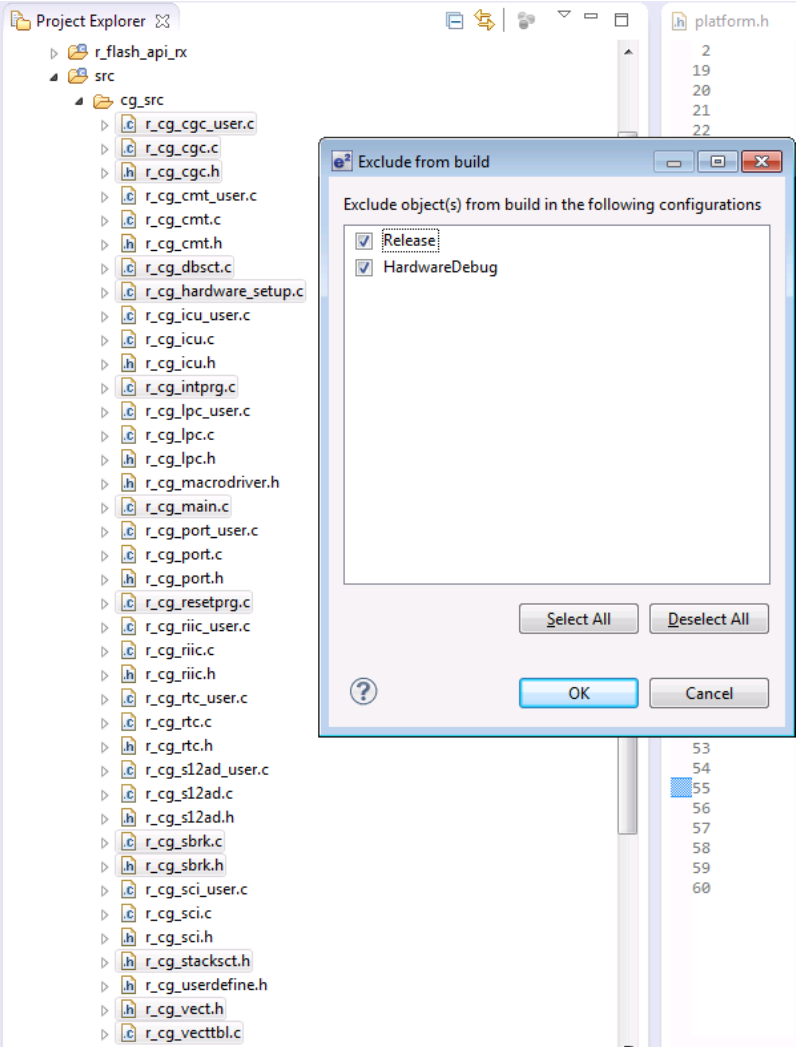
Command: ccrx

All options: -include="C:\PROGRA~1\Renesas\RX\2_1_0/include" -include="C:\Workspace\System_Input_Capture\r_bsp" -include="C:\Workspace\System_Input_Capture\r_config" -include="C:

Expert settings:

Command line pattern: ${COMMAND} ${FLAGS} ${OUTPUT_FLAG} ${OUTPUT_P

- Since we are using ‘System_Input_Capture.c’ for the main() C function, and FIT BSP source files for start-up and interrupt handling, we need to exclude some code generator source files from the build.
- In the Project Explorer pane, expand the ‘System_Input_Capture -> src -> cg_src’ folder node.
- Using the CTRL key, select the following files with the left mouse button:
  r_cg_cgc_user.c
  r_cg_cgc.c
  r_cg_cgc.h
  r_cg_dbsct.c
  r_cg_hardware_setup.c
  r_cg_intprg.c
  r_cg_main.c
  r_cg_resetprg.c
  r_cg_sbrk.c
  r_cg_sbrk.h
  r_cg_stacksct.h
  r_cg_vect.h
  r_cg_vecttbl.c
- Then right-click and select ‘Exclude from build…’. The dialog opposite will be displayed.
- Click the ‘Select All’ button to exclude the selected files from the build for Release and Hardware Debug build configurations.
- Click OK.

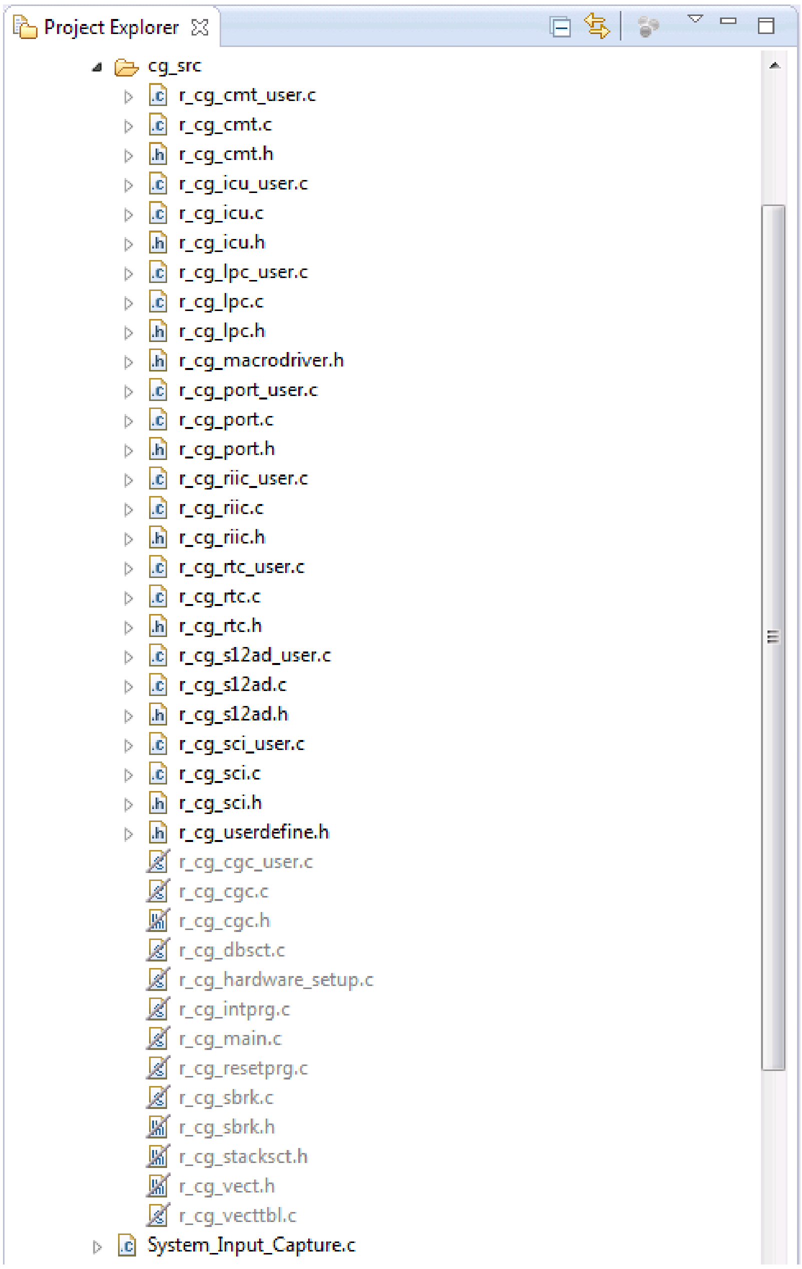


Exclude from build

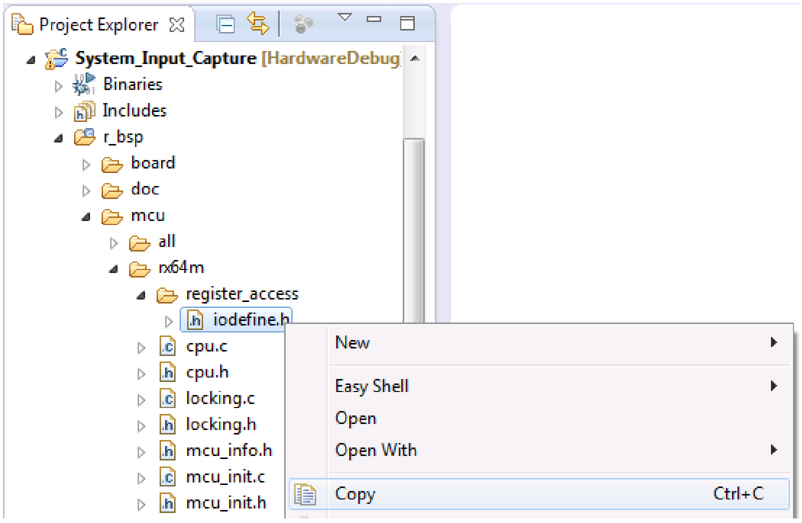
Exclude object(s) from build in the following configurations

☑ Release
☑ HardwareDebug

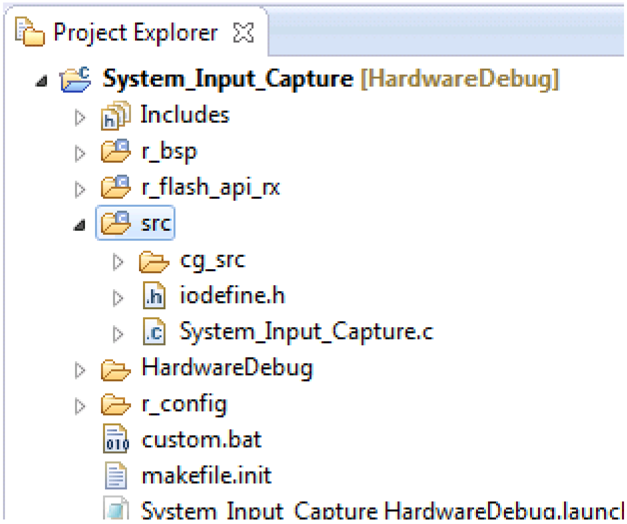
- Verify that the files shown opposite are excluded from the build (greyed-out with a line through the icon).



- In Project Explorer, locate the file ‘iodefine.h’ as shown opposite, right-click and select ‘Copy’.
- Paste the file into the ‘System_Input_Capture -> src’ folder, using the right-click context menu or the ‘Edit – Paste’ main menu item.



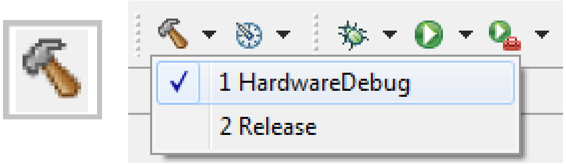
- Paste the file into the ‘System_Input_Capture -> src’ folder, using the right-click context menu or the ‘Edit – Paste’ main menu item.
- This is required because the code generator generates the #include “..\iodefine.h” directive in all of its source files.



Project Explorer

```
System_Input_Capture [HardwareDebug]
    Includes
    r_bsp
    r_flash_api_rx
    src
        cg_src
        iodefine.h
        System_Input_Capture.c
    HardwareDebug
    r_config
    custom.bat
    makefile.init
    System Input Capture HardwareDebug.launcl
```

- Click the top level ‘System_Input_Capture’ folder again, and then the arrow next to the build button (hammer icon), and select the ‘HardwareDebug’ option.



- e^2 studio will now build the code.

The project now contains a combined FIT/Code Generator base for adding the application code for System_Input_Capture. The steps described so far are also applicable to any user application that uses a FIT BSP, one or more FIT modules and code generator code in combination.

The steps required to complete the sample are described in the next Section. The next part of the Tutorial is more specific to the System_Input_Capture application, but is also applicable for customers who wish to build a FIT-based application while still retaining the power and flexibility of the Renesas Code Generator.

6.Completing the System_Input_Capture Application

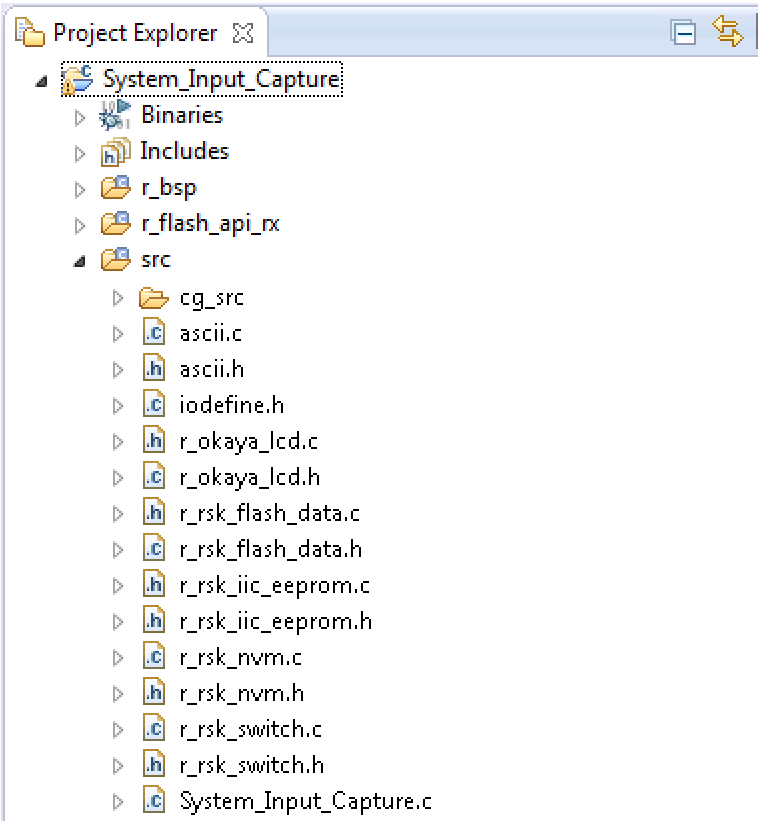
It is not the intention of this of this document to provide a detailed walkthrough of the System_Input_Capture sample, not to provide a Tutorial on the use of the code generator built into e^2 studio. Therefore, rather than inspect detailed application code in this document, it is sufficient to simply copy source files from the completed System_Input_Capture sample into the user's workspace. This applies to both top level application source files as well as manual edits that have been made in the user code areas of the code generator source files.

The reader is encouraged to inspect the application source files and the code added to the user area of the code generator files. In the latter case, this is easily achieved with any diff tool.

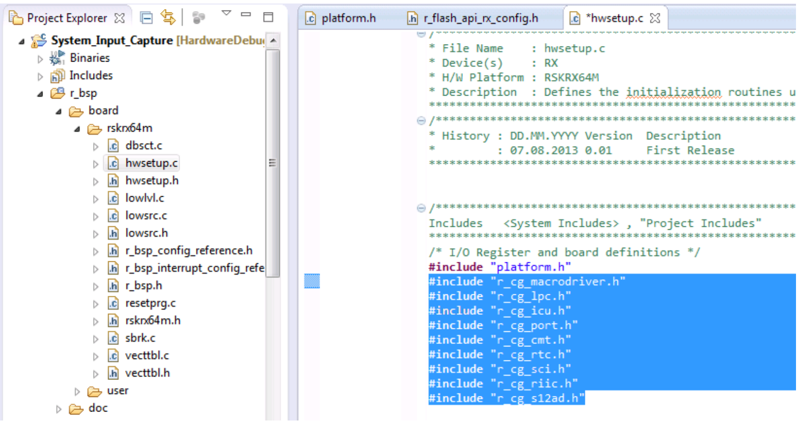
- Locate the following files in the completed System_Input_Capture sample (src folder) and copy them into the src folder of your workspace:

  ascii.c
  ascii.h
  r_okaya_lcd.c
  r_okaya_lcd.h
  r_rsk_data_flash.c
  r_rsk_data_flash.h
  r_rsk_iic_eeprom.c
  r_rsk_iic_eeprom.h
  r_rsk_nvm.c
  r_rsk_nvm.h
  r_rsk_switch.c
  r_rsk_switch.h
  System_Input_Capture.c
- Locate the following files in the completed System_Input_Capture sample (src/cg_src folder)and copy them into the src/cg_src folder of your workspace:

  r_cg_userdefine.h
  r_cg_sci_user.c
  r_cg_sci.h
  r_cg_s12ad_user.c
  r_cg_s12ad.h
  r_cg_rtc_user.c
  r_cg_rtc.h
  r_cg_riic_user.c
  r_cg_riic.h
  r_cg_riic.c
  r_cg_icu_user.c
  r_cg_icu.h
  r_cg_icu.c
  r_cg_cmt_user.c
  r_cg_cmt.h
  r_cg_cmt.c



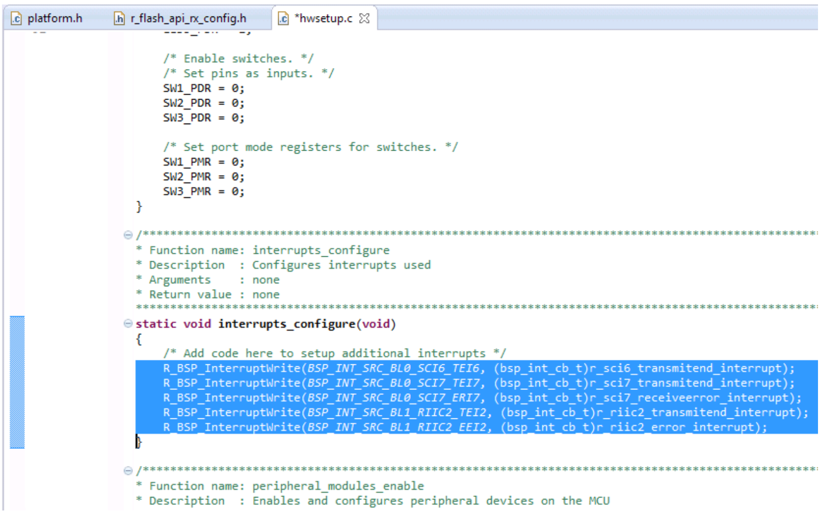
- It is necessary to edit the FIT BSP file hwsetup.c in order to initialise our code generator peripheral modules.
- There are stub functions in this file provided for this purpose, where we will add our initialisation code.
- Open the file 'r_bsp -> board -> rskrx64m -> hwsetup.c' by double-clicking it's node in the Project Explorer pane.
- Add the following code as shown in the screenshot opposite.



```
/*****************************************************
* File Name    : hwsetup.c
* Device(s)    : RX
* H/W Platform : RSKRX64M
* Description  : Defines the initialization routines u
******************************************************
/*****************************************************
* History : DD.MM.YYYY Version  Description
*         : 07.08.2013 0.01     First Release
******************************************************


/*****************************************************
Includes   <System Includes> , "Project Includes"
******************************************************
/* I/O Register and board definitions */
#include "platform.h"
#include "r_cg_macrodriver.h"
#include "r_cg_lpc.h"
#include "r_cg_icu.h"
#include "r_cg_port.h"
#include "r_cg_cmt.h"
#include "r_cg_rtc.h"
#include "r_cg_sci.h"
#include "r_cg_riic.h"
#include "r_cg_s12ad.h"
```

```
#include "r_cg_macrodriver.h"
#include "r_cg_lpc.h"
#include "r_cg_icu.h"
#include "r_cg_port.h"
#include "r_cg_cmt.h"
#include "r_cg_rtc.h"
#include "r_cg_sci.h"
#include "r_cg_riic.h"
#include "r_cg_s12ad.h"
```

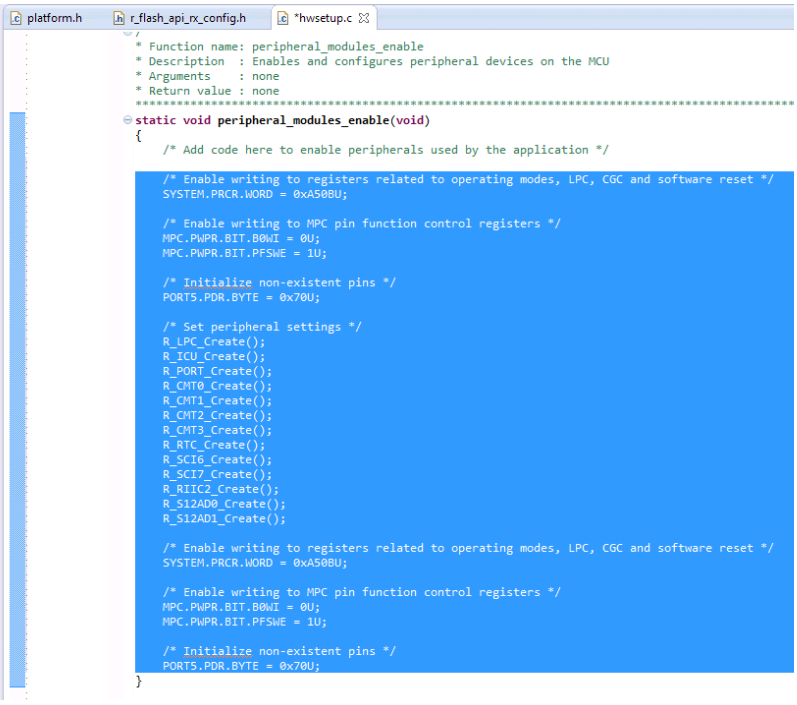
- Add the following code as shown in the screenshot opposite.
- These API calls connect the BSP interrupt architecture to the code-generator-supplied interrupt call back functions for the required BL0 and BL1 interrupts. For more information on the RX64M group interrupts refer to the hardware manual.



```
    /* Enable switches. */
    /* Set pins as inputs. */
    SW1_PDR = 0;
    SW2_PDR = 0;
    SW3_PDR = 0;

    /* Set port mode registers for switches. */
    SW1_PMR = 0;
    SW2_PMR = 0;
    SW3_PMR = 0;
}

/*******************************************************************************************
* Function name: interrupts_configure
* Description  : Configures interrupts used
* Arguments    : none
* Return value : none
*******************************************************************************************
static void interrupts_configure(void)
{
    /* Add code here to setup additional interrupts */
    R_BSP_InterruptWrite(BSP_INT_SRC_BL0_SCI6_TEI6, (bsp_int_cb_t)r_sci6_transmitend_interrupt);
    R_BSP_InterruptWrite(BSP_INT_SRC_BL0_SCI7_TEI7, (bsp_int_cb_t)r_sci7_transmitend_interrupt);
    R_BSP_InterruptWrite(BSP_INT_SRC_BL0_SCI7_ERI7, (bsp_int_cb_t)r_sci7_receiveerror_interrupt);
    R_BSP_InterruptWrite(BSP_INT_SRC_BL1_RIIC2_TEI2, (bsp_int_cb_t)r_riic2_transmitend_interrupt);
    R_BSP_InterruptWrite(BSP_INT_SRC_BL1_RIIC2_EEI2, (bsp_int_cb_t)r_riic2_error_interrupt);
}

/*******************************************************************************************
* Function name: peripheral_modules_enable
* Description  : Enables and configures peripheral devices on the MCU
```

```
R_BSP_InterruptWrite(BSP_INT_SRC_BL0_SCI6_TEI6, (bsp_int_cb_t)r_sci6_transmitend_interrupt);
R_BSP_InterruptWrite(BSP_INT_SRC_BL0_SCI7_TEI7, (bsp_int_cb_t)r_sci7_transmitend_interrupt);
R_BSP_InterruptWrite(BSP_INT_SRC_BL0_SCI7_ERI7, (bsp_int_cb_t)r_sci7_receiveerror_interrupt);
R_BSP_InterruptWrite(BSP_INT_SRC_BL1_RIIC2_TEI2, (bsp_int_cb_t)r_riic2_transmitend_interrupt);
R_BSP_InterruptWrite(BSP_INT_SRC_BL1_RIIC2_EEI2, (bsp_int_cb_t)r_riic2_error_interrupt);
```

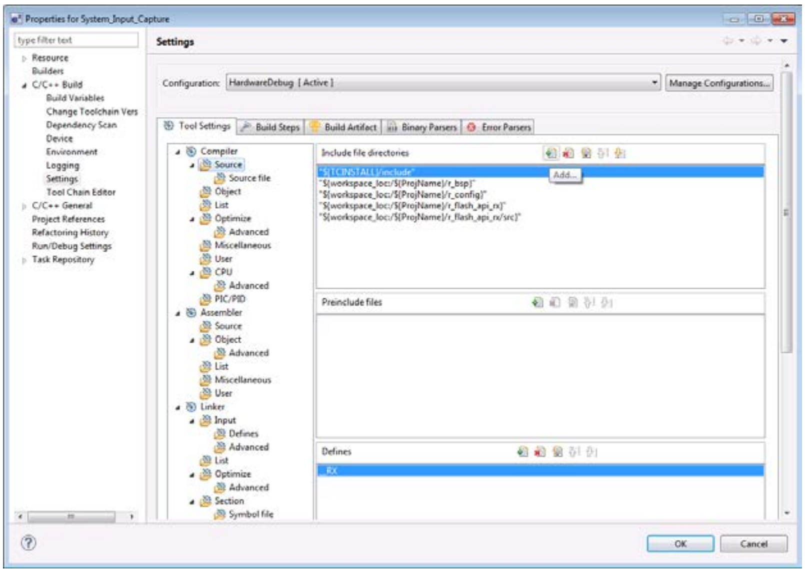
- Add the following code as shown in the screenshot opposite.
- Is code is taken from the code-generator-supplied r_cg_hardware_setup.c file and is necessary to initialise all of the code generator peripherals used in the project.
- Remember to save the edits (ctrl-s).



```
 * Function name: peripheral_modules_enable
 * Description  : Enables and configures peripheral devices on the MCU
 * Arguments    : none
 * Return value : none
 ******************************************************************************************
static void peripheral_modules_enable(void)
{
    /* Add code here to enable peripherals used by the application */

    /* Enable writing to registers related to operating modes, LPC, CGC and software reset */
    SYSTEM.PRCR.WORD = 0xA50BU;

    /* Enable writing to MPC pin function control registers */
    MPC.PWPR.BIT.B0WI = 0U;
    MPC.PWPR.BIT.PFSWE = 1U;

    /* Initialize non-existent pins */
    PORT5.PDR.BYTE = 0x70U;

    /* Set peripheral settings */
    R_LPC_Create();
    R_ICU_Create();
    R_PORT_Create();
    R_CMT0_Create();
    R_CMT1_Create();
    R_CMT2_Create();
    R_CMT3_Create();
    R_RTC_Create();
    R_SCI6_Create();
    R_SCI7_Create();
    R_RIIC2_Create();
    R_S12AD0_Create();
    R_S12AD1_Create();

    /* Enable writing to registers related to operating modes, LPC, CGC and software reset */
    SYSTEM.PRCR.WORD = 0xA50BU;

    /* Enable writing to MPC pin function control registers */
    MPC.PWPR.BIT.B0WI = 0U;
    MPC.PWPR.BIT.PFSWE = 1U;

    /* Initialize non-existent pins */
    PORT5.PDR.BYTE = 0x70U;
}
```

```
/* Enable writing to registers related to operating modes, LPC, CGC and software reset */
SYSTEM.PRCR.WORD = 0xA50BU;

/* Enable writing to MPC pin function control registers */
MPC.PWPR.BIT.B0WI = 0U;
MPC.PWPR.BIT.PFSWE = 1U;

/* Initialize non-existent pins */
PORT5.PDR.BYTE = 0x70U;

/* Set peripheral settings */
R_LPC_Create();
R_ICU_Create();
R_PORT_Create();
R_CMT0_Create();
R_CMT1_Create();
R_CMT2_Create();
R_CMT3_Create();
R_RTC_Create();
R_SCI6_Create();
R_SCI7_Create();
R_RIIC2_Create();
R_S12AD0_Create();
R_S12AD1_Create();

/* Enable writing to registers related to operating modes, LPC, CGC and software reset */
SYSTEM.PRCR.WORD = 0xA50BU;

/* Enable writing to MPC pin function control registers */
MPC.PWPR.BIT.B0WI = 0U;
MPC.PWPR.BIT.PFSWE = 1U;

/* Initialize non-existent pins */
PORT5.PDR.BYTE = 0x70U;
```

- Open the project properties from the 'Project -> Properties' menu item, or by selecting the project in the Project Explorer pane then clicking the button.
- In the 'Properties for System_Input_Capture' dialog, browse to 'C/C++ Build -> Settings Tool Settings -> Compiler -> Source'
- Click the button shown opposite with the tool-tip.



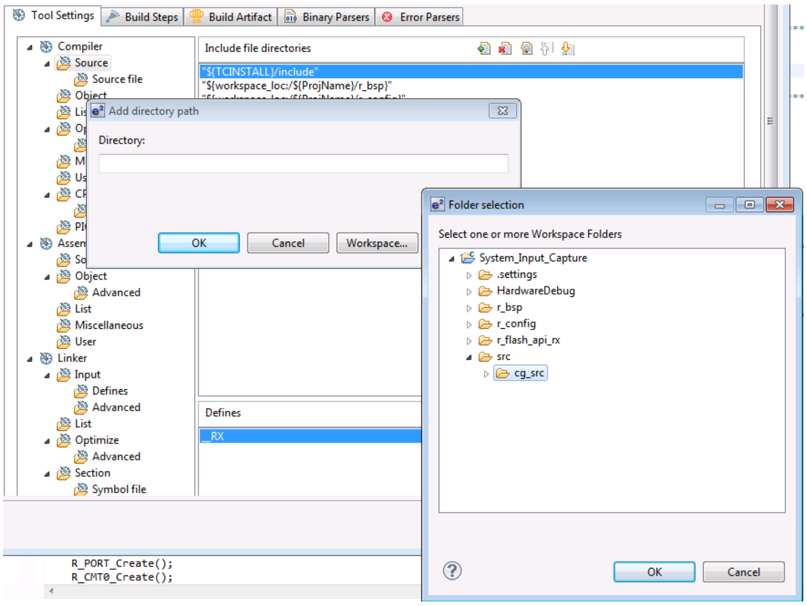
Configuration: HardwareDebug [Active]

Include file directories:

"${TCINSTALL}/include"
"${workspace_loc:/${ProjName}/r_bsp}"
"${workspace_loc:/${ProjName}/r_config}"
"${workspace_loc:/${ProjName}/r_flash_api_rx}"
"${workspace_loc:/${ProjName}/r_flash_api_rx/src}"

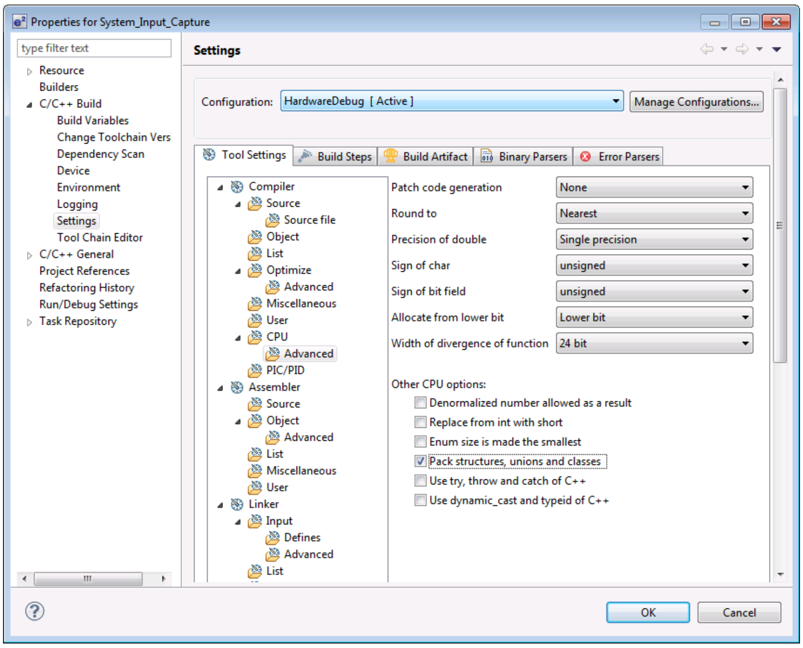
Defines: RX

- In the 'Add directory path' dialog, click the 'Workspace…' button.
- In the 'Folder selection' dialog', browse and select the 'System_Input_Capture -> src -> cg_src' folder. Click 'OK'.
- In the 'Add directory path' dialog, click 'OK'.
- Repeat the steps above to add the path 'System_Input_Capture -> src' to the list of Include file directories.
- In the 'Properties for System_Input_Capture' dialog, click 'Apply'.



```
R_PORT_Create();
R_CMT0_Create();
```

- Staying the ‘Tool Settings’ tab of the ‘Properties for System_Input_Capture’ dialog, navigate to ‘Compiler -> CPU -> Advanced’
- In ‘Other CPU options, ensure that ‘Pack structures, unions and classes’ is ticked.
- This is to ensure efficient reading, writing and use of NVM.
- In the ‘Properties for System_Input_Capture’ dialog, click ‘OK’.


Properties for System_Input_Capture

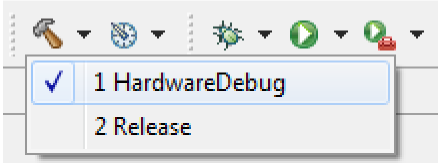
Settings

Configuration: HardwareDebug [Active]

Patch code generation: None
Round to: 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

Other CPU 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 the top level ‘System_Input_Capture’ folder again, and then the arrow next to the build button (hammer icon), and select the ‘HardwareDebug’ option.



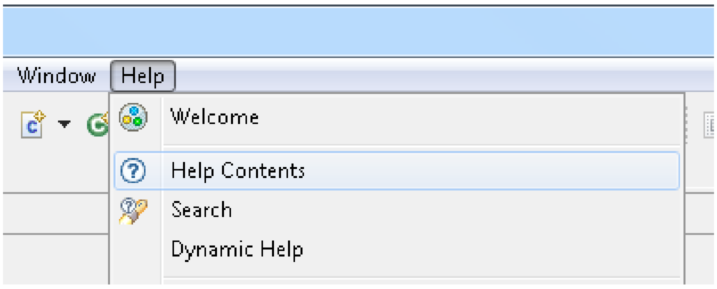
- e^2 studio will now build the code.

The application is now complete. For instructions on how to run the System_Input_Capture sample refer to the Description.txt file contained in the completed sample. To use the e2 studio debugger, refer to the Tutorial Manual r20ut2594eg.

7.Additional Information

Technical Support

For details on how to use e^2 studio, refer to the help file by opening e^2 studio, then selecting Help > Help Contents from the menu bar.



For information about the RX/64M series microcontrollers refer to the RX/64M Group Hardware Manual.

For information about the RX assembly language, refer to the RX Series Software Manual.

Technical Contact Details

Please refer to the contact details listed in section 9 of the "Quick Start Guide"

General information on Renesas microcontrollers can be found on the Renesas website at:
http://www.renesas.com/

Trademarks

All brand or product names used in this manual are trademarks or registered trademarks of their respective companies or organisations.

Copyright

This document may be, wholly or partially, subject to change without notice. All rights reserved. Duplication of this document, either in whole or part is prohibited without the written permission of Renesas Electronics Europe Limited.

© 2014 Renesas Electronics Europe Limited. All rights reserved.
© 2014 Renesas Electronics Corporation. All rights reserved.
© 2014 Renesas Solutions Corp. All rights reserved.

REVISION HISTORY	RSK RX64M Code Generator Tutorial Manual

Rev.	Date	Description	
		Page	Summary
1.00	May 31, 2014	—	First Edition issued

Renesas Starter Kit Manual: Code Generator Tutorial Manual

Publication Date: Rev. 1.00 May 31, 2014

Published by: Renesas Electronics Corporation

RENESAS

Renesas Electronics Corporation

SALES OFFICES

http://www.renesas.com

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

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, 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: +44-1628-651-700, Fax: +44-1628-651-804
Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327
Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, 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 Rd., Putuo District, Shanghai, China
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: +852-2886-9318, Fax: +852 2886-9022/9044
Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, 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-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

© 2014 Renesas Electronics Corporation. All rights reserved.
Colophon 2.0

RX64M Group

Renesas Electronics Corporation