User's Manual

32

RX113 Group

Renesas Starter Kit Code Generator Tutorial Manual
For e^2 studio

RENESAS MCU
RX Family / RX100 Series

Renesas Electronics
www.renesas.com

Rev. 1.00 Nov 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 Code Generator for RX together with the e^2 studio IDE to create a working project for the RSK platform. It is intended for users designing sample code on the RSK platform, using the many different incorporated peripheral devices.

The manual comprises of step-by-step instructions to generate code and import it into e^2 studio, but does not intend to be a complete guide to software development on the RSK platform. Further details regarding operating the RX113 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 RX113 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.	RSKRX113 User's Manual	R20UT2756EG
Tutorial	Provides a guide to setting up RSK environment, running sample code and debugging programs.	RSKRX113 Tutorial Manual	R20UT2760EG
Quick Start Guide	Provides simple instructions to setup the RSK and run the first sample.	RSKRX113 Quick Start Guide	R20UT2761EG
Code Generator Tutorial	Provides a guide to code generation in the e^2 studio IDE.	RSKRX113 Code Generator Tutorial Manual	R20UT3255EG
Schematics	Full detail circuit schematics of the RSK.	RSKRX113 Schematics	R20UT2755EG
Hardware Manual	Provides technical details of the RXRX113 microcontroller.	RX113 Group, User's Manual: Hardware	R01UH0448EJ

2. List of Abbreviations and Acronyms

Abbreviation	Full Form
ADC	Analog-to-Digital Converter
API	Application Programming Interface
COM	COMmunications port referring to PC serial port
CPU	Central Processing Unit
DVD	Digital Versatile Disc
E1	On-chip Debugger
GUI	Graphical User Interface
IDE	Integrated Development Environment
IRQ	Interrupt Request line
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MCU	Micro-controller Unit
PC	Personal Computer
Pmod™	Digilent Pmod™ Compatible connector. PmodTM is registered to Digilent Inc. Digilent-Pmod_Interface_Specification
PLL	Phase-locked Loop
RSK	Renesas Starter Kit
SCI	Serial Communications Interface
SPI	Serial Peripheral Interface

Table of Contents

1. Overview 7
1.1 Purpose 7
1.2 Features 7

2. Introduction 8

3. Project Creation with e^2 studio 9
3.1 Introduction 9
3.2 Creating the Project 9

4. Code Generation Using the e^2 studio plug in 15
4.1 Introduction 15
4.2 Code Generator Tour 15
4.3 Code Generation 18
4.3.1 Clock Generator 18
4.3.2 I/O Ports 20
4.3.3 Serial Communications Interface 21
4.3.4 12-bit A/D Converter 22
4.3.5 Generating the code 24
4.4 Building the Project 25

5. User Code Integration 26
5.1 Support file copying 26
5.2 LCD file copying 26
5.3 Adding Code to Generated Files 27
5.3.1 r_cg_userdefine.h Code Insertion 27
5.3.2 r_cg_s12ad.c Code Insertion 27
5.3.3 r_cg_s12ad.h Code Insertion 28
5.3.4 r_cg_s12ad_user.c Code Insertion 28
5.3.5 r_cg_sci_user.c Code Insertion 28
5.3.6 r_cg_sci.h Code Insertion 29
5.3.7 r_cg_main.c Code Insertion 30
5.4 Additional include paths 33
5.5 Linker Section Addresses for Release configuration 34

6. Debugging the Project 35

7. Additional Information 36

1. Overview

1.1 Purpose

This RSK is an evaluation tool for Renesas microcontrollers. This manual describes how to use the e^2 studio IDE code generator plug in to create a working project for the RSK platform.

1.2 Features

This RSK tutorial guides the user through creating a project to evaluate the following features:

- Project creation with e^2 studio,
- Code Generation using the Code Generator plug in,
- User circuitry such as switches, LEDs and a potentiometer.

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

2. Introduction

This manual is designed to answer, in tutorial form, how to use the Code Generator plug in 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:

- Project generation using the e^2 studio,
- Detailed use of the Code Generator plug in for e^2 studio,
- Integration with custom code,
- Building and running the project e^2 studio.

The project generator will create a tutorial project with two 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 RX113

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. Project Creation with e2 studio

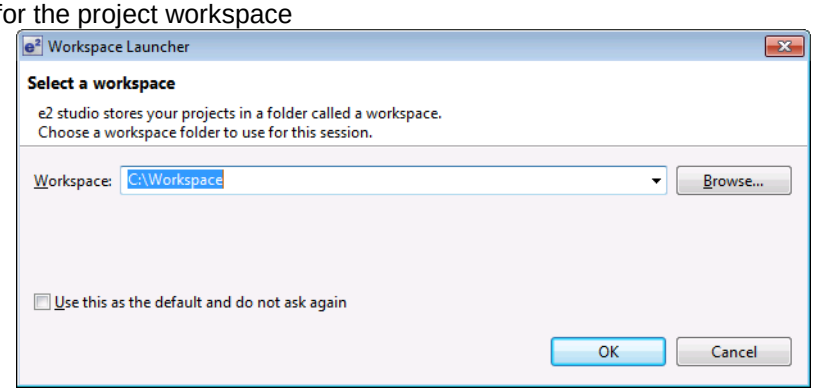
3.1 Introduction

In this section the user will be guided through the steps required to create a new 'C' project for the RX113 microcontroller, ready to generate peripheral driver code using Code Generator. This project generation step is necessary to create the MCU-specific source, project and debug files.

3.2 Creating the Project

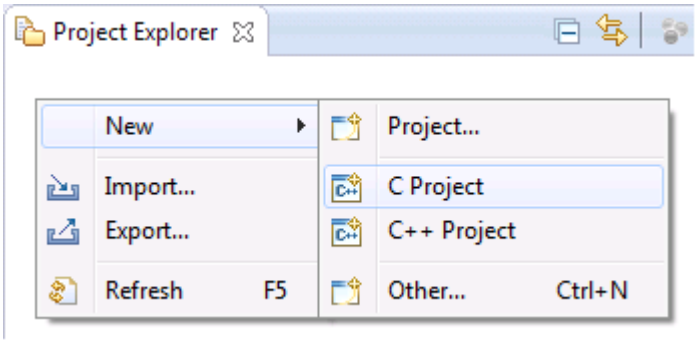
Start e2 studio and select a suitable location for the project workspace

- Start e2 studio and select a suitable location for the project workspace.

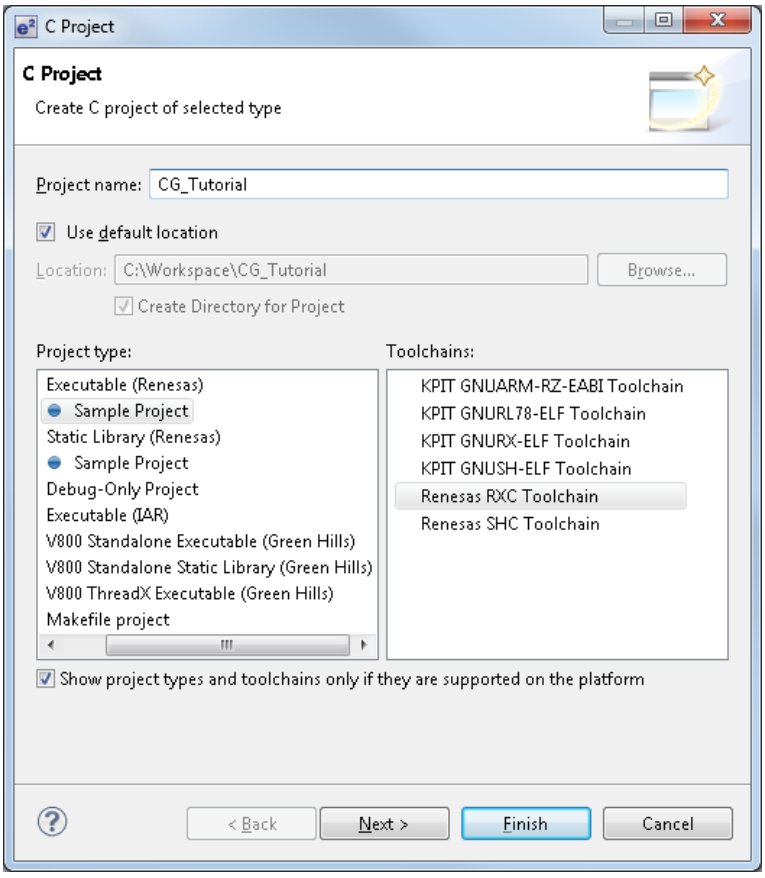


- In the Welcome page, click 'Go to the workbench'.

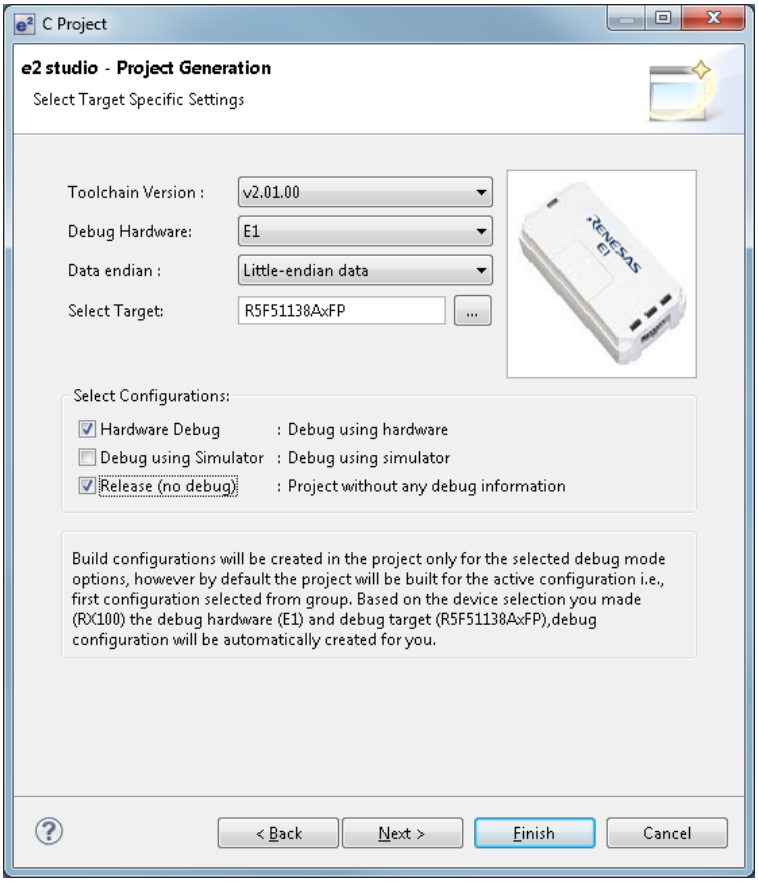
- Create a new C project by right-clicking in the Project Explorer pane and selecting ‘New -> C Project’ as shown. Alternatively, use the menu item ‘File -> New -> C Project’.



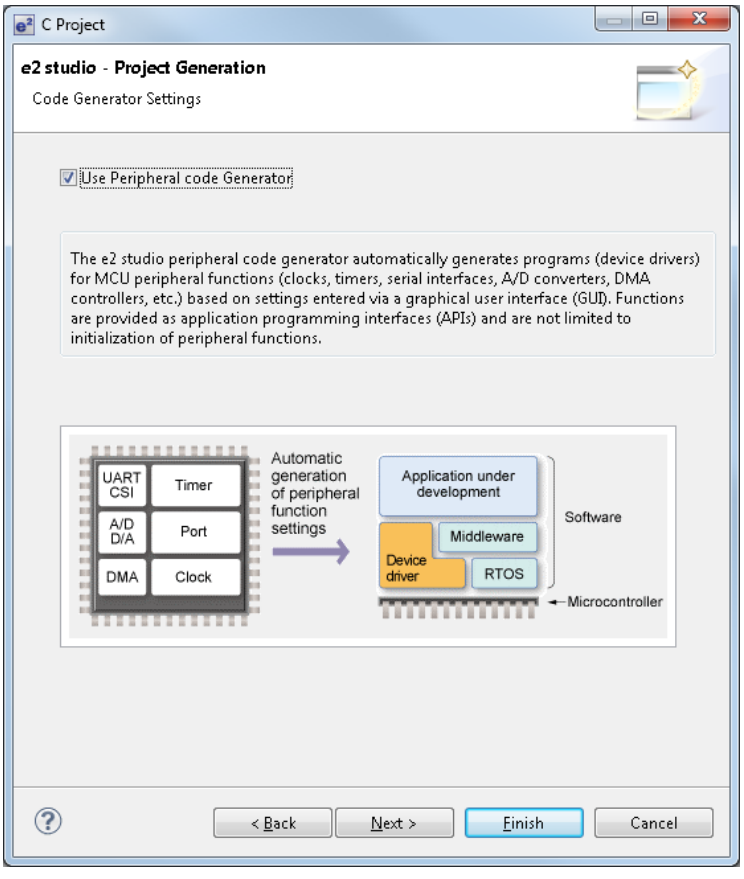
- Enter the project name ‘CG_Tutorial’. In ‘Project type:’ choose ‘Sample Project’. In ‘Toolchains’ choose ‘Renesas RXC Toolchain’. Click ‘Next’.



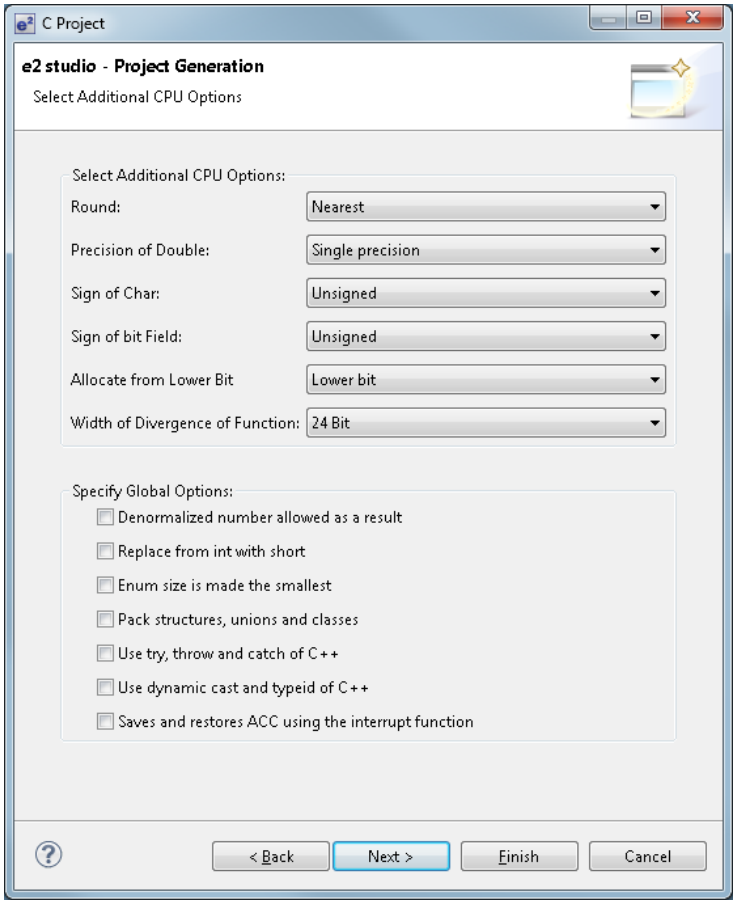
- In the ‘Target Specific Settings’ dialog, select the options as shown in the screenshot opposite.
- The R5F51138AxFP MCU is found under RX100 -> RX113 -> RX113 - 100 pin.
- Click ‘Next’.



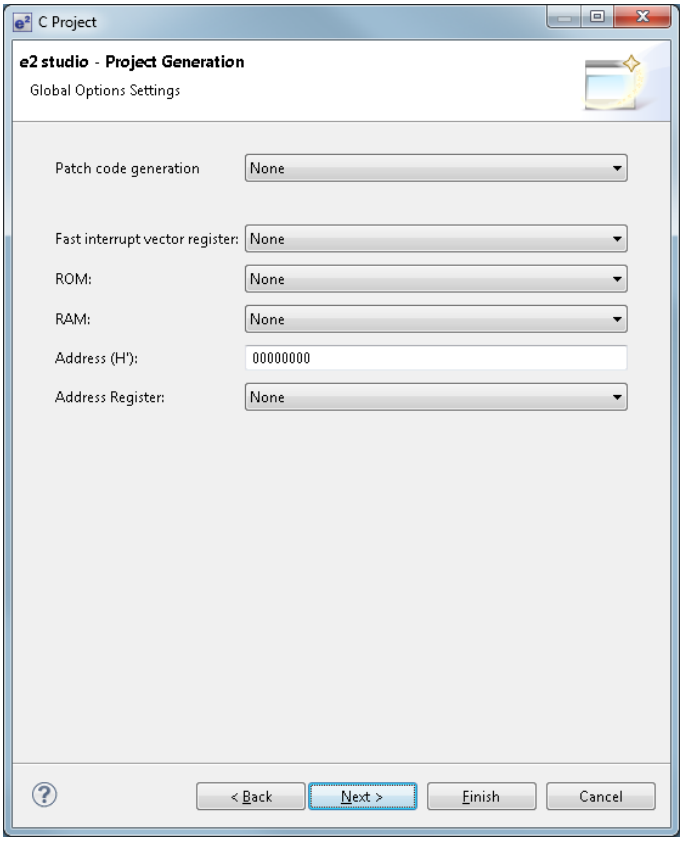
- In the ‘Code Generator Settings’ dialog, ensure the ‘Use Peripheral code Generator’ is checked.
- Click ‘Next’.



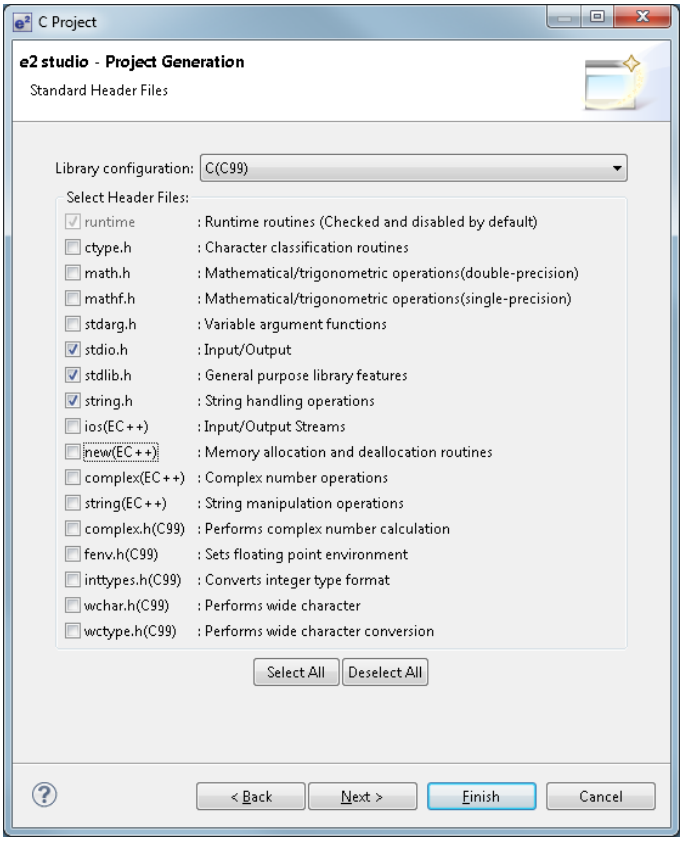
- In ‘Select Additional CPU Options’ leave everything at default values.
- Click ‘Next’.



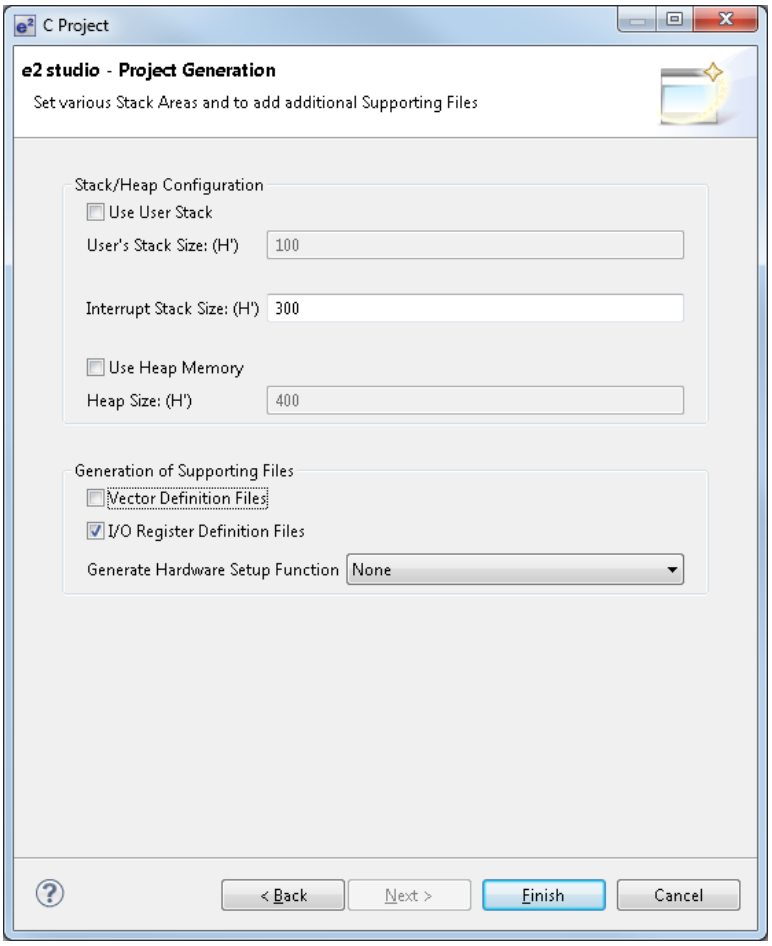
- In the ‘Global Options Settings’ leave everything at default values.
- Click ‘Next’.



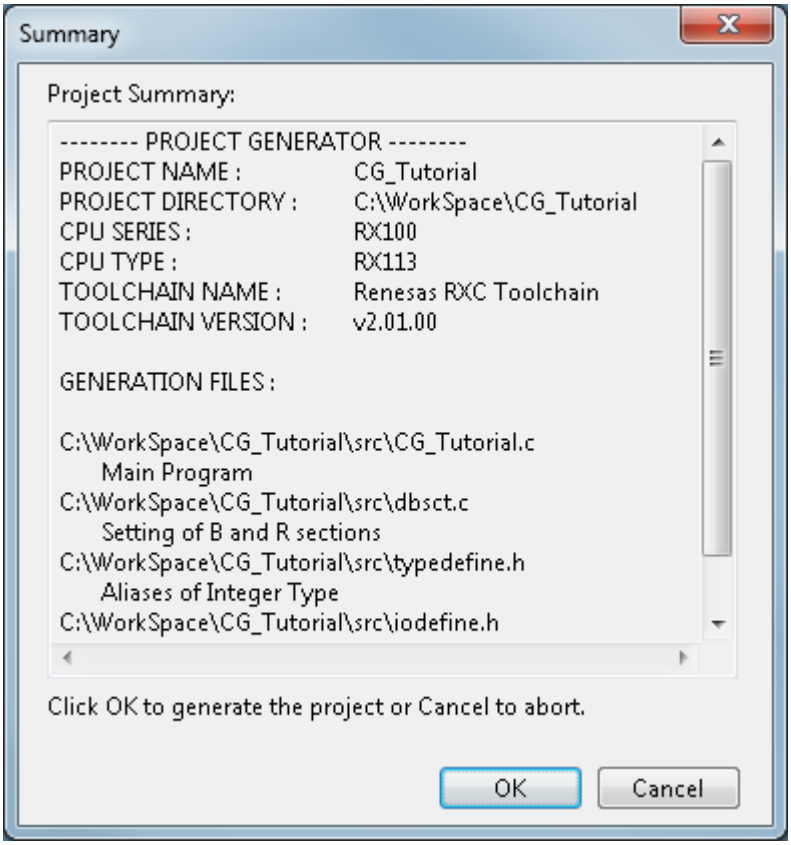
- In the ‘Standard Header Files’ dialog, select C99 for ‘Library Configuration’. Untick ‘new(EC++)’ and leave all others at defaults.
- Click ‘Next’.



- In the next dialog, untick all check boxes except ‘I/O Register Definition Files’ as shown opposite. Click ‘Finish’.



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



4. Code Generation Using the e2 studio plug in

4.1 Introduction

Code Generator is an e2 studio plug in GUI tool for generating template 'C' source code for the RX113. When using Code Generator, the user is able to configure various MCU features and operating parameters using intuitive GUI controls, bypassing the need, in most cases, to refer to sections of the Hardware Manual.

By following the steps detailed in this tutorial, the user will generate an e2 studio project called CG_Tutorial. A fully completed Tutorial project is contained on the DVD and may be imported into e2 studio by following the steps in the Quick Start Guide. This tutorial is intended as a learning exercise for users who wish to use the Code Generator to generate their own custom projects for e2 studio.

Once the user has configured the project, the 'Generate Code' function is used to generate three code modules for each specific MCU feature selected. These code modules are name 'r_cg_xxx.h', 'r_cg_xxx.c', and 'r_cg_xxx_user.c', where 'xxx' is a three letter acronym for the relevant MCU feature, for example 'adc'. Within these code modules, the user is free to add custom code to meet their specific requirement. Custom code should be added between the following comment delimiters:

```
/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

Code Generator will locate these comment delimiters, and preserve any custom code inside the delimiters on subsequent code generation operations. Any code outside of these comment delimiters will be overwritten on subsequent code generation sessions.

The CG_Tutorial project uses the ADC module with external trigger, Serial Communications Interface (SCI) and LCD Driver. These modules are used to perform an A/D conversion, display the results on a terminal program via the Virtual COM port and also on the LCD display attached to the RSK.

Following a tour of the key user interface features of Code Generator in §4.2, the reader is guided through each of the peripheral function configuration dialogs in §4.3. In §5, the reader is familiarised with the structure of the template code, as well as how to add custom code in the areas provided by the Code Generator.

4.2 Code Generator Tour

This section presents a brief tour of Code Generator. For further details of the Code Generator paradigm and reference, refer to the Application Leading Tool Common Operations manual (r20ut2663ej0100). Application Leading Tool is the stand-alone version of Code Generator and this manual is applicable to the Code Generator.

From the e2 studio menus, select 'Window -> Open Perspective -> Other. In the 'Open Perspective' dialog shown in Figure 4-1, select 'Code Generator' and click 'OK'.

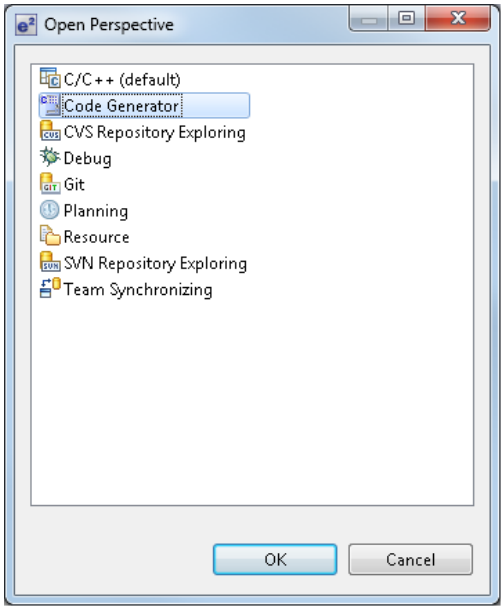


Figure 4-1 Open Perspective Dialog

In the Project Explorer pane, expand the 'Code Generator' and 'Peripheral Functions' node. The Code Generator initial view is displayed as illustrated in Figure 4-2.

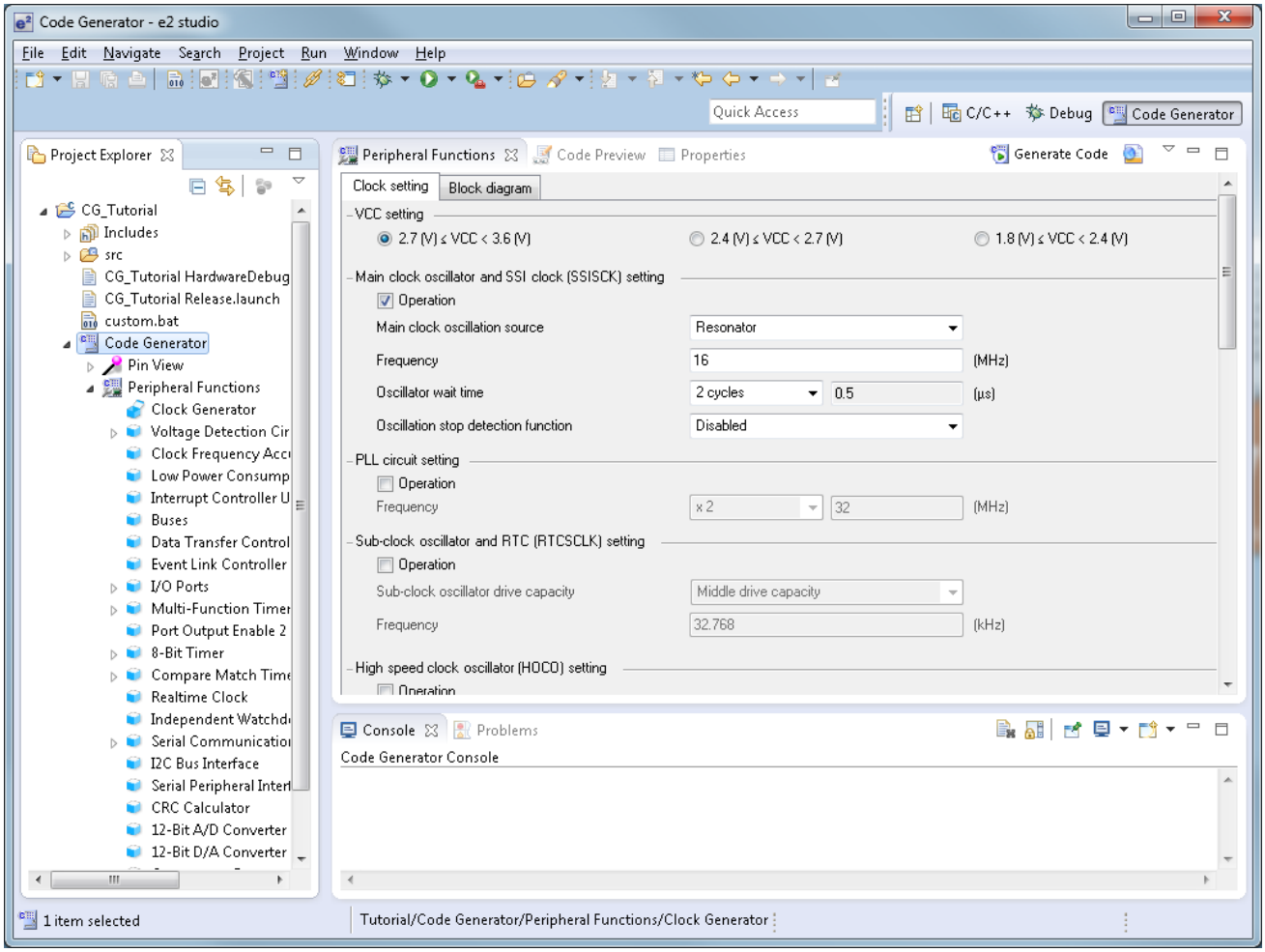


Figure 4-2 Initial View

Code Generator provides GUI features for configuration of MCU subsystems and peripherals. Once the user has configured all required MCU subsystems and peripherals, the user can click the 'Generate Code' button, resulting in a fully configured e^2 studio project.

Navigation to the MCU peripheral configuration screens may be performed by double-clicking the required function in the Code Generator -> Peripheral Function on the left.

It is also possible to see a preview of the code that will be generated for the current peripheral function settings by double-clicking the required function in the Code Generator -> Code Preview on the left.

4.3 Code Generation

In the following sections, the reader is guided through the steps to configure the MCU for a simple tutorial project containing ADC with external switch trigger, Serial Communications Interface (SCI) and LCD Output.

4.3.1 Clock Generator

Figure 4-3 shows a screenshot of Code Generator with the Clock Generator function open.

In this tutorial we are using the 16 MHz crystal resonator for the main clock source with the PLL circuit used as a multiplier. The sub-clock oscillator is used as a clock source for the LCD peripheral.

Double click on the 'Clock Generator' entry in the Code Generator -> Peripheral Functions list.
Configure the Clock Generator options as shown in Figure 4-3.

Proceed to the next section to configure the I/O Ports.

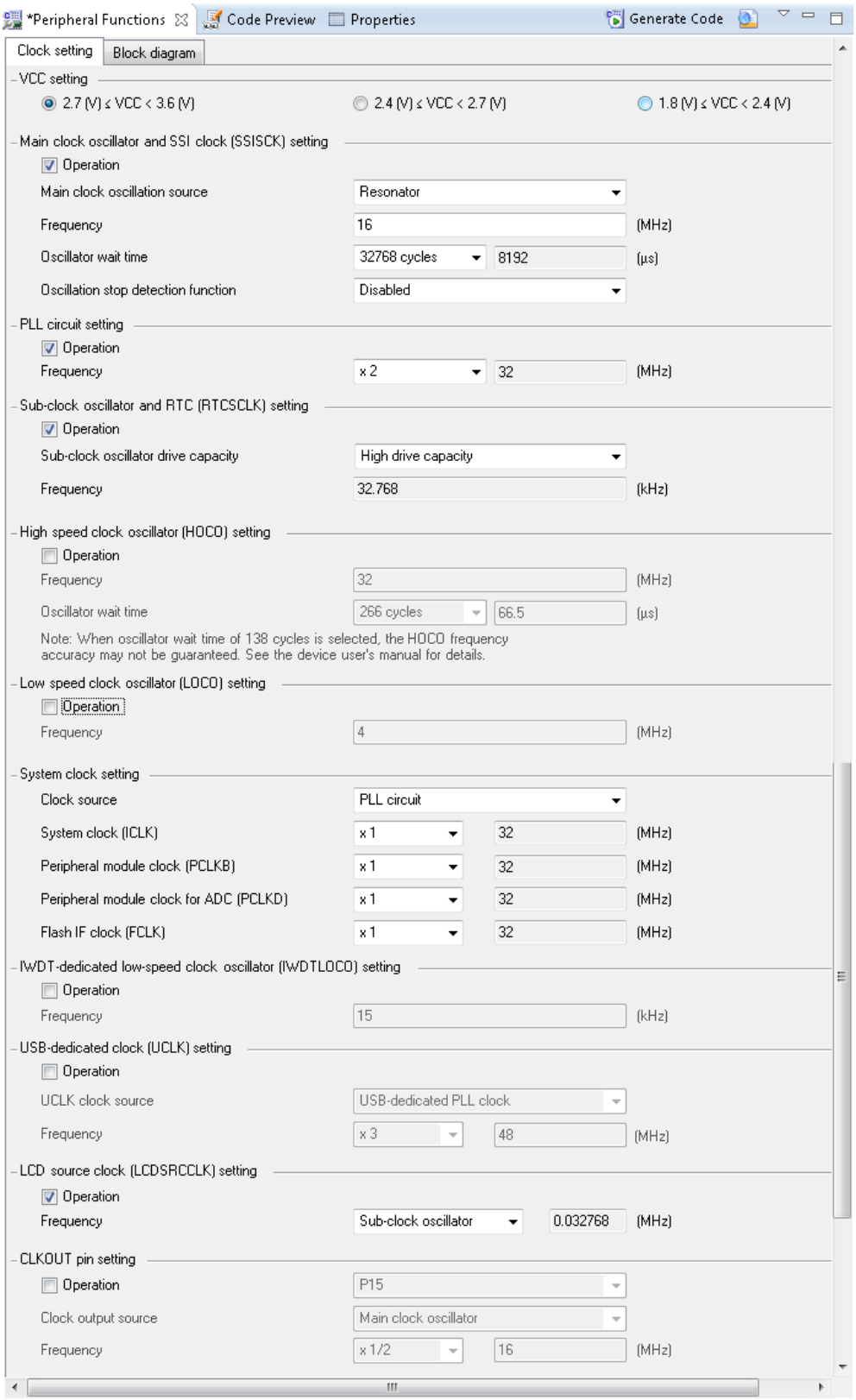


Figure 4-3 Clock setting tab

4.3.2 I/O Ports

This peripheral will be configured to assign output pins for user LEDs and input pins for user switches, with the exception of SW3 which is used as a trigger for the A/D Converter peripheral. Please refer to the RSK schematic for full details of the connectivity.

Double click on the 'I/O Ports' entry in the Code Generator -> Peripheral Functions list.
Configuration is required for Port2, Port3 and PortJ. The port is selected from the tabs at the top of the Peripheral Functions window.

Configure the ports as shown in Figure 4-4 Port 2 Configuration, Figure 4-5 Port 3 Configuration & Figure 4-6 Port J Configuration.

Proceed to the next section to configure the Serial Communications Interface.

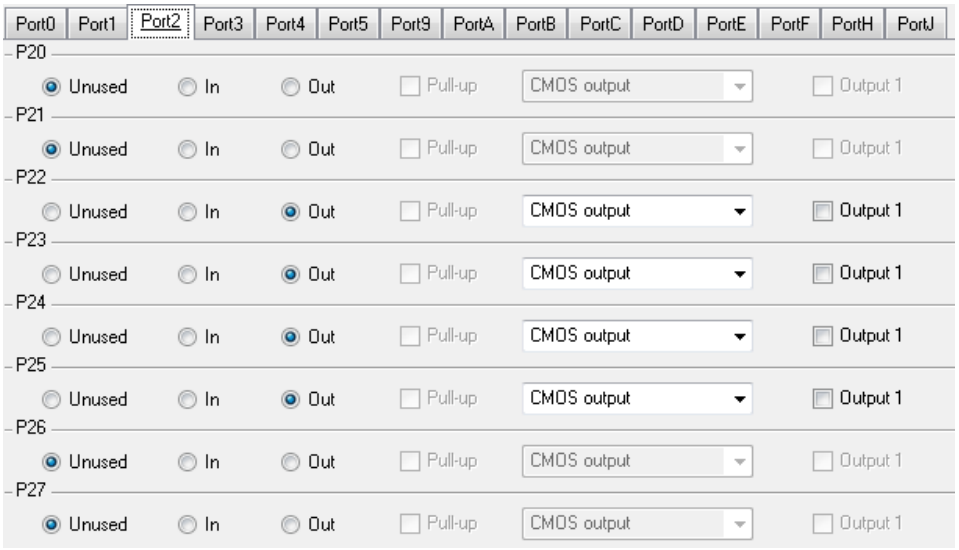


Figure 4-4 Port 2 Configuration

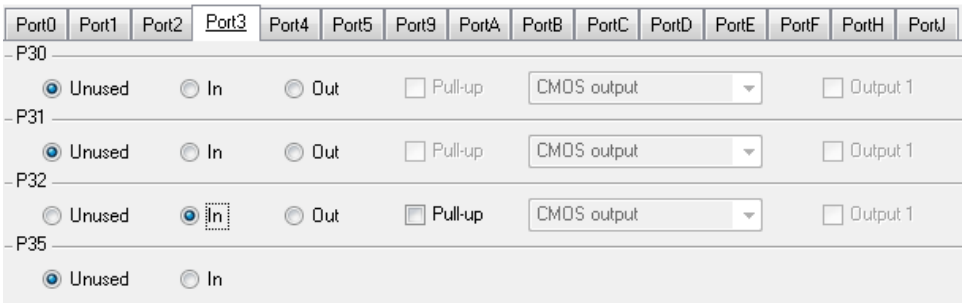


Figure 4-5 Port 3 Configuration

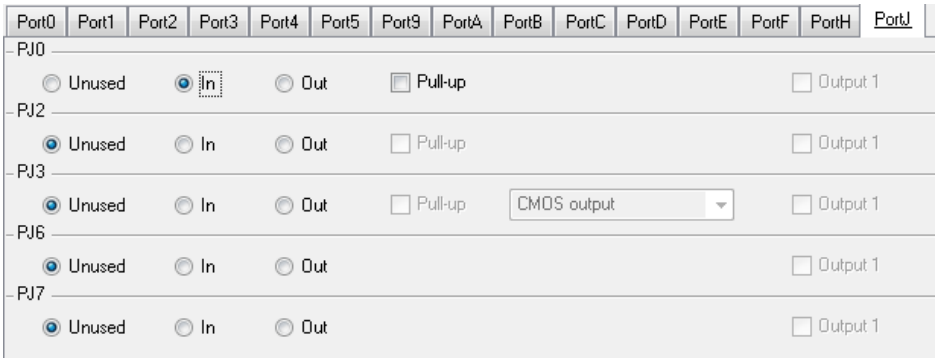


Figure 4-6 Port J Configuration

4.3.3 Serial Communications Interface

This peripheral is configured to use SCI1. This channel of the SCI is connected to the USB to serial converter and allows the application to send data to the terminal program running on the PC.

Double click on the 'Serial Communications Interface' entry in the Code Generator -> Peripheral Functions list. Configuration is required only SC1 which is selected from the tabs at the top of the Peripheral Functions window.

Configure the 'General setting' and 'Setting' sub-tabs as shown in Figure 4-7 SCI1 General Setting tab & Figure 4-8 SCI1 Setting tab.

This will configure the SCI1 channel to use asynchronous Tx/Rx using 8 data bits, No parity, 1 Stop bit at a rate of 19200 baud.

Proceed to the next section to configure the 12-Bit A/D Converter.

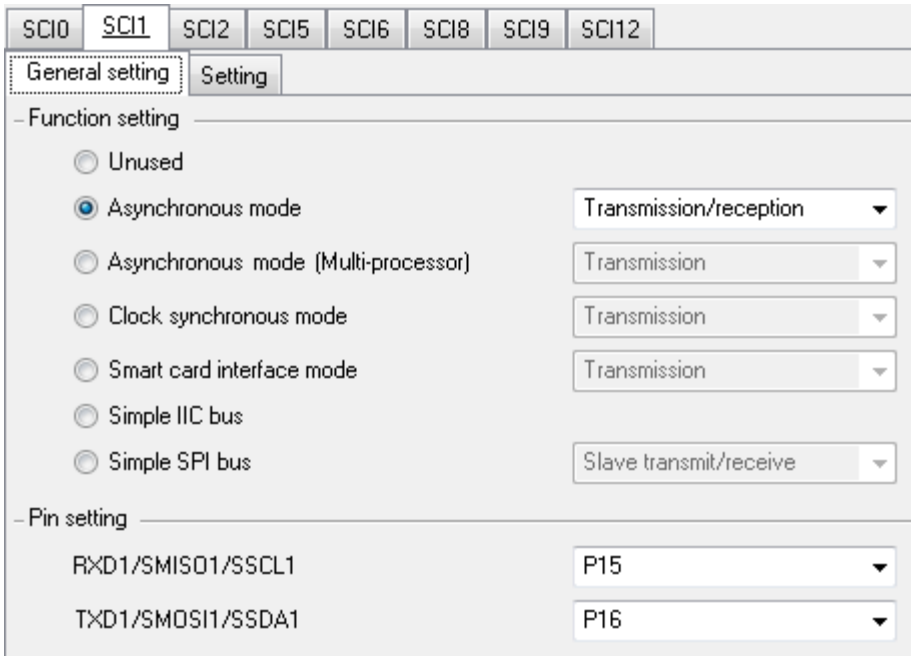


Figure 4-7 SCI1 General Setting tab

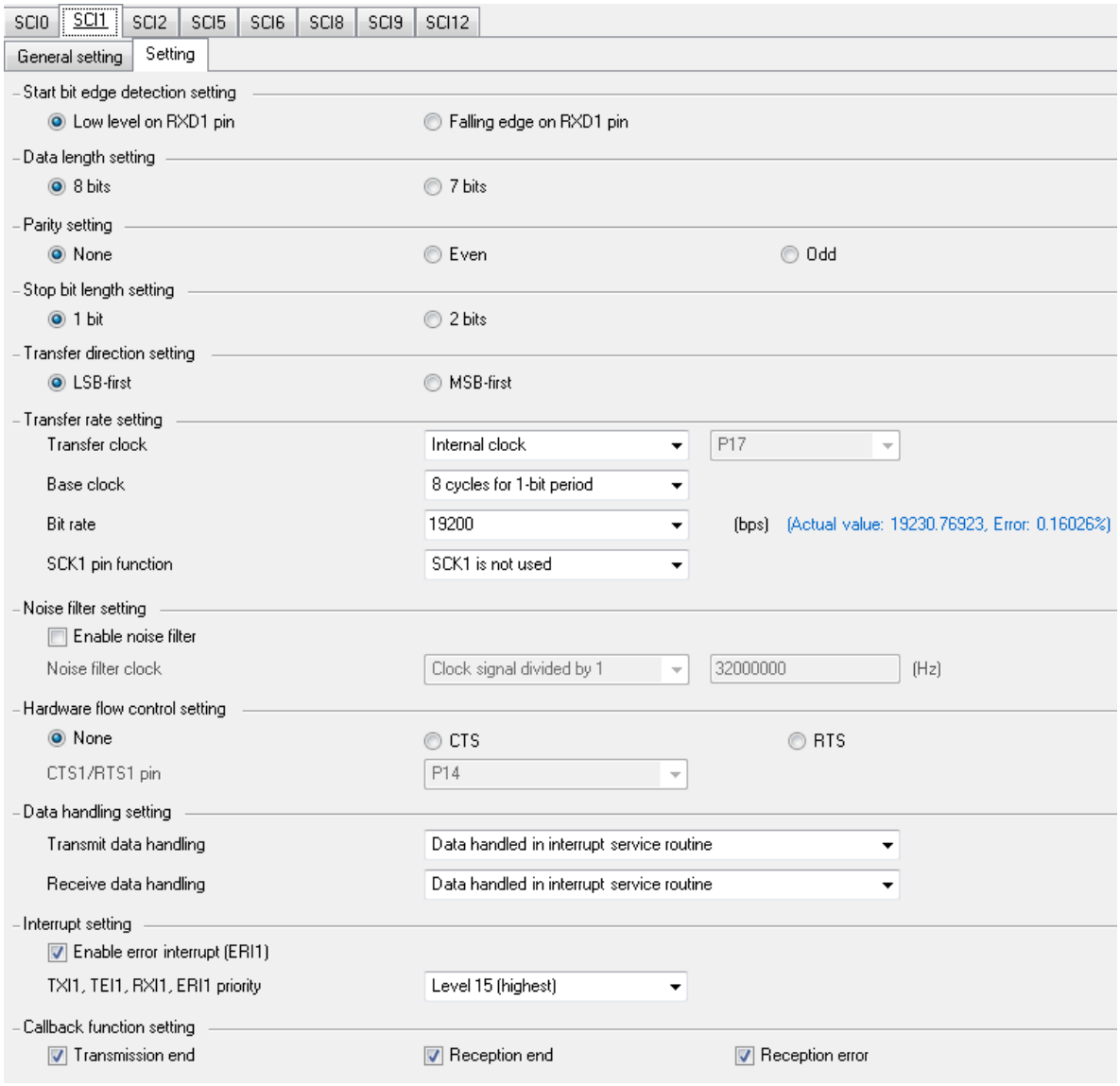


Figure 4-8 SCI1 Setting tab

4.3.4 12-bit A/D Converter

This peripheral is configured to sample the analogue output value of the RV1 potentiometer. The A/D Converter is set to perform a sample when the user presses SW3, which is connected to the ADTRG0 pin of the microcontroller.

Double click on the ’12-bit A/D Converter’ entry in the Code Generator -> Peripheral Functions list.

Configure the ‘General setting’ and ‘Setting’ sub-tabs as shown in Figure 4-9 A/D Converter General setting tab & Figure 4-10 A/D Converter Setting tab.

Code Generator configuration is now complete. Proceed to the next section to generate the code.

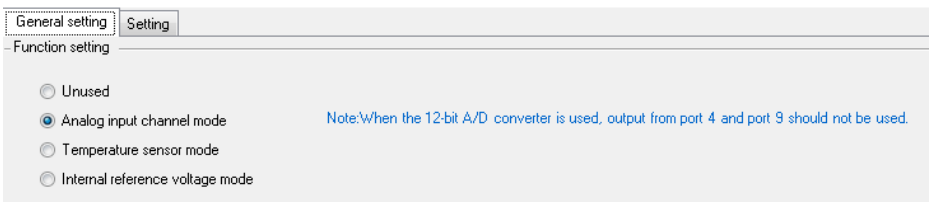


Figure 4-9 A/D Converter General setting tab

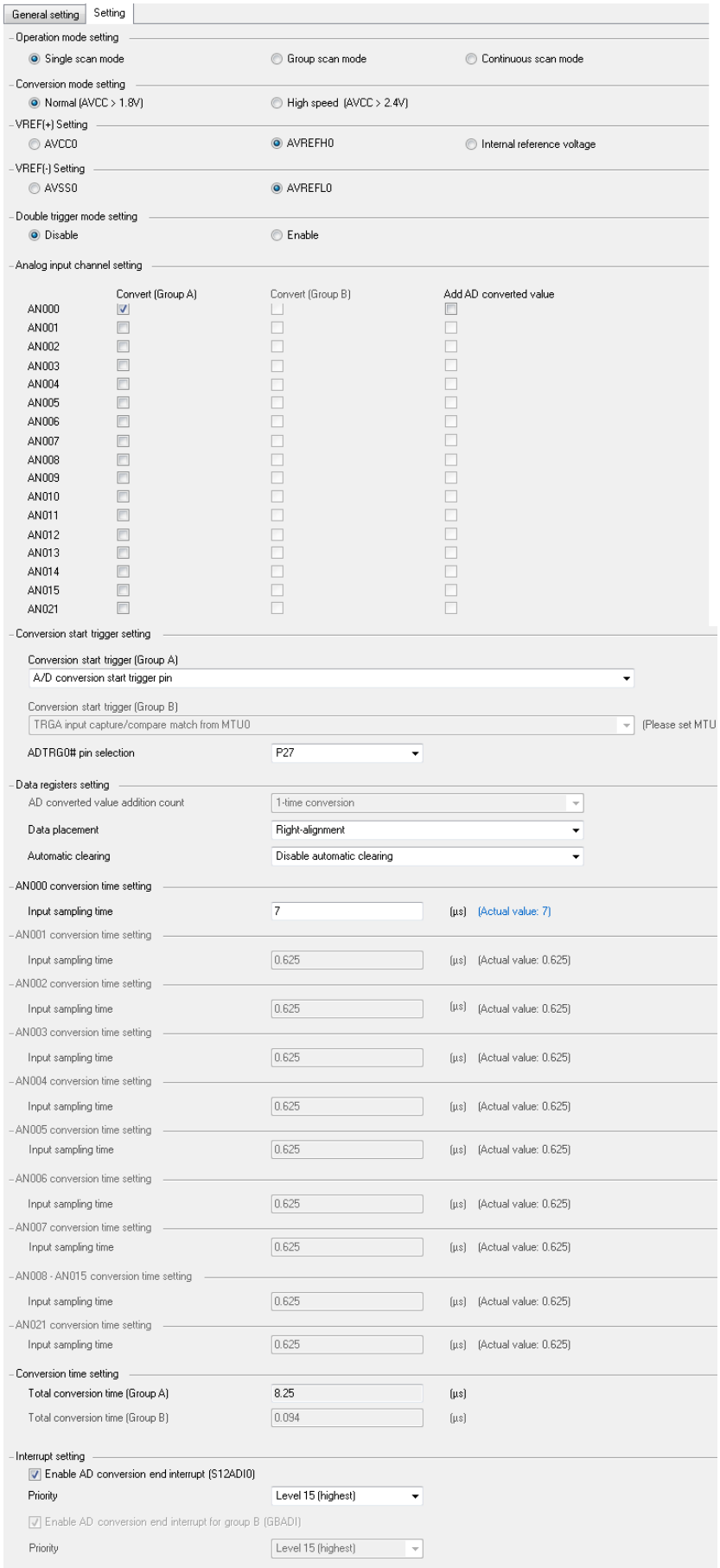


Figure 4-10 A/D Converter Setting tab

4.3.5 Generating the code

Peripheral function configuration is now complete. Click 'Generate Code' button located at the top right of the Peripheral Function tab. The Console pane should report 'The operation of generating file was successful', as shown Figure 4-11 below.

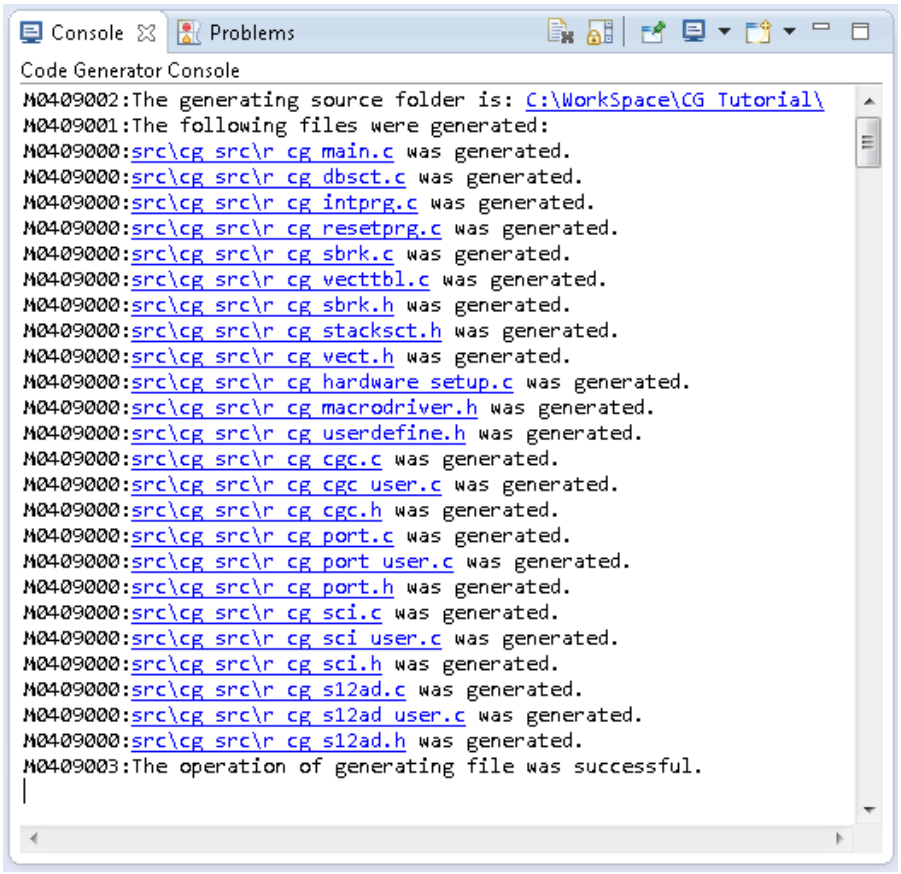


Figure 4-11 Code generator console

4.4 Building the Project

The project template created by Code Generator can now be built. In the Project Explorer pane expand the 'src' folder.

Three files created by the New Project Wizard in §3.2 have been excluded from the build automatically as part of the code generation procedure as shown in Figure 4-12. This is because the main() function now resides in r_cg_main.c in the cg_src folder, type definitions and setting of sections has been handled by the Code Generator.

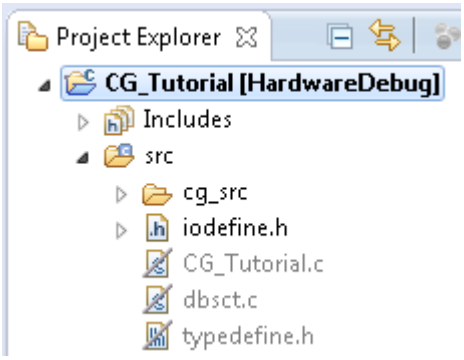


Figure 4-12 Files excluded from the build by Code Generator

Switch back to the 'C/C++' perspective using the 

button on the top right of the e2 studio workspace.

Use 'Build Project' from the 'Project' menu or the  button to build the tutorial. The project will build with no errors.

5. User Code Integration

At this stage of a typical project development the user would expand on the generated code to create the application required. As a demonstration this tutorial will include code lines and files from the complete 'Tutorial' project, supplied with the RSK.

The 'Tutorial' project is included as part of the RSK DVD installation process and can be found at the following location:

C:\Renesas\Workspace\RSK\RSKRX113\Tutorial

When inserting code in Code Generator created files, it must be placed in the areas delimited by comments as follows:

```
/* Start user code for _xxxxx_. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

Where _xxxx_ depends on the particular area of code, i.e. 'function' for insertion of user functions and prototypes, 'global' for insertion of user global variable declarations, or 'include' for insertion of pre-processor include directives. User code inserted inside these comment delimiters is protected from being overwritten by Code Generator, if the user refreshes the Code Generator-generated code.

5.1 Support file copying

RSK support and utility functions are provided in the following files:

r_rsk_utility.c,
r_rsk_utility.h,
rskrx113def.h.

Locate these files in the 'Tutorial' project and copy them in to the 'CG_Tutorial\src' folder. This will be located at the path specified during the project creation in section 3.2 Creating the Project.

5.2 LCD file copying

API functions for the RSK LCD App v2 display are included in the following files:

r_lcd_appv2.c
r_lcd_appv2.h

Locate these files in the 'Tutorial' project and copy them in to the 'CG_Tutorial\src' folder. This will be located at the path specified during the project creation in section 3.2 Creating the Project.
The newly copied files will automatically appear in e^2 studio's Project Explorer window, shown below.

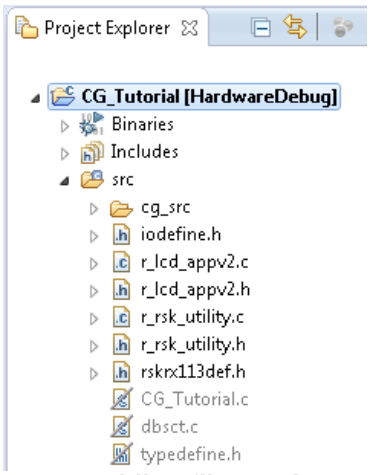


Figure 5-1 Adding files to the project

5.3 Adding Code to Generated Files

This section covers inserting code in to the newly created Code Generator files.

Each subsection is a Code Generated source file that needs to be opened by double clicking on the file name in e^2 studio's Project Tree window in the 'src -> cg_src' tree.

The code from each section should be copied from this document and pasted in to the relevant file at the location indicated.

5.3.1 r_cg_userdefine.h Code Insertion

Open this file by double clicking on the file name in e^2 studio's Project Tree window.

Insert the following at the end of the file between the user code delimiter comments as shown below.

```
/* Start user code for function. Do not edit comment generated here */

#define TRUE            (1)
#define FALSE           (0)

extern volatile uint8_t g_adc_trigger;

/* End user code. Do not edit comment generated here */
```

5.3.2 r_cg_s12ad.c Code Insertion

Open this file by double clicking on the file name in e^2 studio's Project Tree window.

Insert the following at the end of the file between the user code delimiter comments as shown below.

```
/* Start user code for adding. Do not edit comment generated here */

/***********************************************************************
 * Function Name: R_S12AD_SWTriggerStart
 * Description  : This function starts the AD0 converter.
 * Arguments    : None
 * Return Value : None
 ***********************************************************************/
void R_S12AD_SWTriggerStart (void)
{
    S12AD.ADCSR.BIT.ADST = 1U;
}

/***********************************************************************
 End of function R_S12AD_SWTriggerStart
 ***********************************************************************/

/***********************************************************************
 * Function Name: R_S12AD_SWTriggerStop
 * Description  : This function stops the AD0 converter.
 * Arguments    : None
 * Return Value : None
 ***********************************************************************/
void R_S12AD_SWTriggerStop (void)
{
    S12AD.ADCSR.BIT.ADST = 0U;
}

/***********************************************************************
 End of function R_S12AD_SWTriggerStop
 ***********************************************************************/

/* End user code. Do not edit comment generated here */
```

5.3.3 r_cg_s12ad.h Code Insertion

Open this file by double clicking on the file name in e^2 studio's Project Tree window.

Insert the following at the end of the file between the user code delimiter comments as shown below.

```
/* Start user code for function. Do not edit comment generated here */
/* Flag indicates when A/D conversion is complete */
extern volatile uint8_t g_adc_complete;

/* Functions for starting and stopping software triggered A/D conversion */
void R_S12AD_SWTriggerStart (void);
void R_S12AD_SWTriggerStop (void);
/* End user code. Do not edit comment generated here */
```

5.3.4 r_cg_s12ad_user.c Code Insertion

Open this file by double clicking on the file name in e^2 studio's Project Tree window.

Insert the following between the user code delimiter comments as shown below in the file section designated Global variables and functions:

```
/* Start user code for global. Do not edit comment generated here */
/* Flag indicates when A/D conversion is complete */
volatile uint8_t g_adc_complete;

/* End user code. Do not edit comment generated here */
```

Insert the following in to the function `static void r_s12ad_interrupt(void)`:

```
    /* Start user code. Do not edit comment generated here */

    /* Flag that the ADC had completed a sample */
    g_adc_complete = 1;

    /* End user code. Do not edit comment generated here */
```

5.3.5 r_cg_sci_user.c Code Insertion

Open this file by double clicking on the file name in e^2 studio's Project Tree window.

Insert the following between the user code delimiter comments as shown below in the file section designated Global variables and functions:

```
/* Start user code for global. Do not edit comment generated here */

/* Global used to receive a character from the PC terminal */
uint8_t g_rx_char;

/* Flag used to control transmission to PC terminal */
volatile uint8_t g_tx_flag = FALSE;

/* Flag used locally to detect transmission complete */
static volatile uint8_t sci1_txdone;

/* End user code. Do not edit comment generated here */
```

Insert the following in to the function `static void r_sci1_callback_transmitend(void)`

```
    /* Start user code. Do not edit comment generated here */
    sci1_txdone = TRUE;

    /* End user code. Do not edit comment generated here */
```

Insert the following in to the function `static void r_sci1_callback_receiveend(void)`

```
    /* Start user code. Do not edit comment generated here */
    /* Check the contents of g_rx_char */

        g_rx_char = g_rx_char & 0xDF;           /* Ensure ASCII char is in upper case */
```

```
        /* Check for the 'c' trigger command */
    if ('C' == g_rx_char)
    {
        g_adc_trigger = TRUE;
    }

    /* Set up SCI1 receive buffer and callback function again */
    R_SCI1_Serial_Receive((uint8_t *)&g_rx_char, 1);


    /* End user code. Do not edit comment generated here */
```

Insert the following between the user code delimiter comments at the end of the file:

```
/* Start user code for adding. Do not edit comment generated here */

/***********************************************************************
* Function Name: R_SCI1_AsyncTransmit
* Description  : This function sends SCI1 data and waits for the transmit end flag.
* Arguments    : tx_buf -
*                    transfer buffer pointer
*                tx_num -
*                    buffer size
* Return Value : status -
*                    MD_OK or MD_ARGERROR
***********************************************************************/
MD_STATUS R_SCI1_AsyncTransmit (uint8_t * const tx_buf, const uint16_t tx_num)
{
    MD_STATUS status = MD_OK;

    /* clear the flag before initiating a new transmission */
    sci1_txdone = FALSE;

    /* Send the data using the API */
    status = R_SCI1_Serial_Send(tx_buf, tx_num);

    /* Wait for the transmit end flag */
    while (FALSE == sci1_txdone)
    {
        /* Wait */
    }
    return (status);
}

/***********************************************************************
* End of function R_SCI1_AsyncTransmit
***********************************************************************/

/* End user code. Do not edit comment generated here */
```

5.3.6 r_cg_sci.h Code Insertion

Insert the following between the user code delimiter comments at the end of the file:

```
/* Start user code for function. Do not edit comment generated here */

/* Exported functions used to transmit a number of bytes and wait for completion */
MD_STATUS R_SCI1_AsyncTransmit (uint8_t * const tx_buf, const uint16_t tx_num);

/* Character is used to receive key presses from PC terminal */
extern uint8_t g_rx_char;

/* Flag used to control transmission to PC terminal */
extern volatile uint8_t g_tx_flag;

/* End user code. Do not edit comment generated here */
```

5.3.7 r_cg_main.c Code Insertion

Insert the following between the user code delimiter comments as shown below in the file section designated `Includes:`

```
/* Start user code for include. Do not edit comment generated here */
#include "r_cg_s12ad.h"
#include "r_lcd_appv2.h"
#include "r_rsk_utility.h"
#include "rskrx113def.h"

/* End user code. Do not edit comment generated here */
```

Insert the following between the user code delimiter comments as shown below in the file section designated Global Variables and functions:

```
/* Start user code for global. Do not edit comment generated here */

/* Welcome banner - displayed on serial port at startup*/
static uint8_t welcome_banner[] = "RSK RX113 - Tutorial - Press 'c' or SW3 for ADC Conversion\r\n\0";

/* Prototype declaration for get_adc */
static uint16_t get_adc (void);

/* Prototype declaration for lcd_display_adc */
static void lcd_display_adc (const uint16_t adc_result);

/* Prototype declaration for uart_display_adc */
static void uart_display_adc (const uint8_t adc_count, const uint16_t adc_result);

/* Variable to store the ADC conversion count for user display */
static uint8_t adc_count = 0;

/* Prototype declaration for led_display_count */
static void led_display_count (const uint8_t count);

/* Variable for flagging user requested ADC conversion */
volatile uint8_t g_adc_trigger = FALSE;

/* End user code. Do not edit comment generated here */
```

Insert the following in to the function **`void main (void)`**.
Note this overwrites the while(1U) loop included by Code Generator.

```
    /* Start user code. Do not edit comment generated here */

    /* Display Project Title on LCD*/
    R_LCD_DisplayPanelString( PANEL_LCD_LINE1, (uint8_t*) "TUTOR");

    /* Set up SCI1 receive buffer and callback function */
    R_SCI1_Serial_Receive((uint8_t *) &g_rx_char, 1);

    /* Enable SCI1 operations */
    R_SCI1_Start();

    /* Display Welcome Banner on Serial Port */
    R_SCI1_AsyncTransmit(welcome_banner, sizeof(welcome_banner));

    while (1U)
    {
        uint16_t adc_result;

        /* If the user has requested ADC sample via the serial port */
        if (TRUE == g_adc_trigger)
        {
            /* Call the function to perform an ADC conversion */
            adc_result = get_adc();

            /* Display the result on the LCD */
            lcd_display_adc(adc_result);
```

```
            /* Increment the adc_count and display using the LEDs */
            if (16 == (++adc_count))
            {
                adc_count = 0;
            }
            led_display_count(adc_count);

            /* Send the result to SCI1 UART */
            uart_display_adc(adc_count, adc_result);

            /* Reset the flag */
            g_adc_trigger = FALSE;
        }

        /* SW3 is directly wired into the ADTRG0n pin so will
         cause the conversion and interrupt */
        else if (TRUE == g_adc_complete)
        {
            /* Get the result of the ADC conversion */
            R_S12AD_Get_ValueResult(ADCHANNEL0, &adc_result);

            /* Display the result on the LCD */
            lcd_display_adc(adc_result);

            /* Increment the adc_count and display using the LEDs */
            if (16 == (++adc_count))
            {
                adc_count = 0;
            }
            led_display_count(adc_count);

            /* Send the result to the UART */
            uart_display_adc(adc_count, adc_result);

            /* Reset the flag */
            g_adc_complete = FALSE;
        }
        else
        {
            /* do nothing */
        }
    }

    /* End user code. Do not edit comment generated here */
```

Insert the following in to the function **void R_MAIN_UserInit (void):**

```
    /* Start user code. Do not edit comment generated here */

    /* Initialise the LCD for the RSK LCD APP V2 display board */
    R_LCD_Create();
    R_LCD_Start();

    /* Start the ADC */
    R_S12AD_Start();

    /* End user code. Do not edit comment generated here */
```

Insert the following between the user code delimiter comments at the end of the file:

```
/* Start user code for adding. Do not edit comment generated here */

/***********************************************************************
 * Function Name : get_adc
 * Description   : Creates a ADC12 Software trigger and returns the ADC result,
 *                           once the ADC conversion is complete.
 * Argument      : none
 * Return value  : uint16_t ADC sample value
 ***********************************************************************/
static uint16_t get_adc (void)
{
    /* A variable to retrieve the ADC result */
    uint16_t adc_result;

    /* Start a conversion */
```

```
    R_S12AD_SWTriggerStart();

    /* Wait for the ADC conversion to complete */
    while (FALSE == g_adc_complete)
    {
        /* Wait */
    }

    /* Stop conversion */
    R_S12AD_SWTriggerStop();

    /* Clear ADC flag */
    g_adc_complete = FALSE;

    R_S12AD_Get_ValueResult(ADCHANNEL0, &adc_result);

    /* Set AD conversion start trigger source back to ADTRG0n pin */
    R_S12AD_Start();

    return adc_result;
}

/***********************************************************************
 * End of function get_adc
 ***********************************************************************/

/***********************************************************************
 * Function Name : lcd_display_adc
 * Description   : Converts ADC result to a string and displays
 *                 it on the LCD panel.
 * Argument      : uint16_t adc result
 * Return value  : none
 ***********************************************************************/
static void lcd_display_adc (const uint16_t adc_result)
{
    /* Declare temporary character string */
    char lcd_buf[4];

    /* Convert ADC result into a character string, and store in the
     local string lcd_buffer */
    uint16_to_string(lcd_buf, 0u, adc_result);

    /* Display the ADC value - Line 3 provides three
     * characters, so skip the unused leading zero
     */
    R_LCD_DisplayPanelString( PANEL_LCD_LINE3, (uint8_t *) lcd_buf + 1);
}

/***********************************************************************
 * End of function lcd_display_adc
 ***********************************************************************/

/***********************************************************************
 * Function Name : uart_display_adc
 * Description   : Converts ADC result to a string and sends it to the UART1.
 * Argument      : uint8_t : adc_count
 *                 uint16_t: ADC result
 * Return value  : none
 ***********************************************************************/
static void uart_display_adc (const uint8_t adc_count, const uint16_t adc_result)
{
    /* Declare a temporary variable */
    char a;

    /* Declare temporary character string */
    static uint8_t uart_buffer[] = "ADC xH Value: xxxH\r\n";

    /* Convert ADC result into a character string, and store in the local.
     Casting to ensure use of correct data type. */
    a = (char) (adc_count & 0x000F);
    uart_buffer[4] = (char) ((a < 0x0A) ? (a + 0x30) : (a + 0x37));
    a = (char) ((adc_result & 0x0F00) >> 8);
    uart_buffer[14] = (char) ((a < 0x0A) ? (a + 0x30) : (a + 0x37));
    a = (char) ((adc_result & 0x00F0) >> 4);
    uart_buffer[15] = (char) ((a < 0x0A) ? (a + 0x30) : (a + 0x37));
    a = (char) (adc_result & 0x000F);
    uart_buffer[16] = (char) ((a < 0x0A)                    ? (a + 0x30) : (a + 0x37));
```

```
    /* Send the string to the UART */
    R_SCI1_AsyncTransmit(uart_buffer, sizeof(uart_buffer));
}

/***********************************************************************
 * End of function uart_display_adc
 ***********************************************************************/

/***********************************************************************
 * Function Name : led_display_count
 * Description   : Converts count to binary and displays on 4 LEDS0-3
 * Argument      : uint8_t count
 * Return value  : none
 ***********************************************************************/
static void led_display_count (const uint8_t count)
{
    /* Set LEDs according to lower nibble of count parameter */
    LED0 = (uint8_t) ((count & 0x01) ? LED_ON : LED_OFF);
    LED1 = (uint8_t) ((count & 0x02) ? LED_ON : LED_OFF);
    LED2 = (uint8_t) ((count & 0x04) ? LED_ON : LED_OFF);
    LED3 = (uint8_t) ((count & 0x08) ? LED_ON : LED_OFF);
}

/***********************************************************************
 * End of function led_display_count
 ***********************************************************************/

/* End user code. Do not edit comment generated here */
```

5.4 Additional include paths

Before the project can be built the compiler needs some additional include paths added. Select the CG_Tutorial project in the Project Explorer pane. Use the button in the toolbar to open the project settings. Navigate to 'C/C++ Build -> Settings ->Compiler -> Source and click the button as shown in below in Figure 5-2.

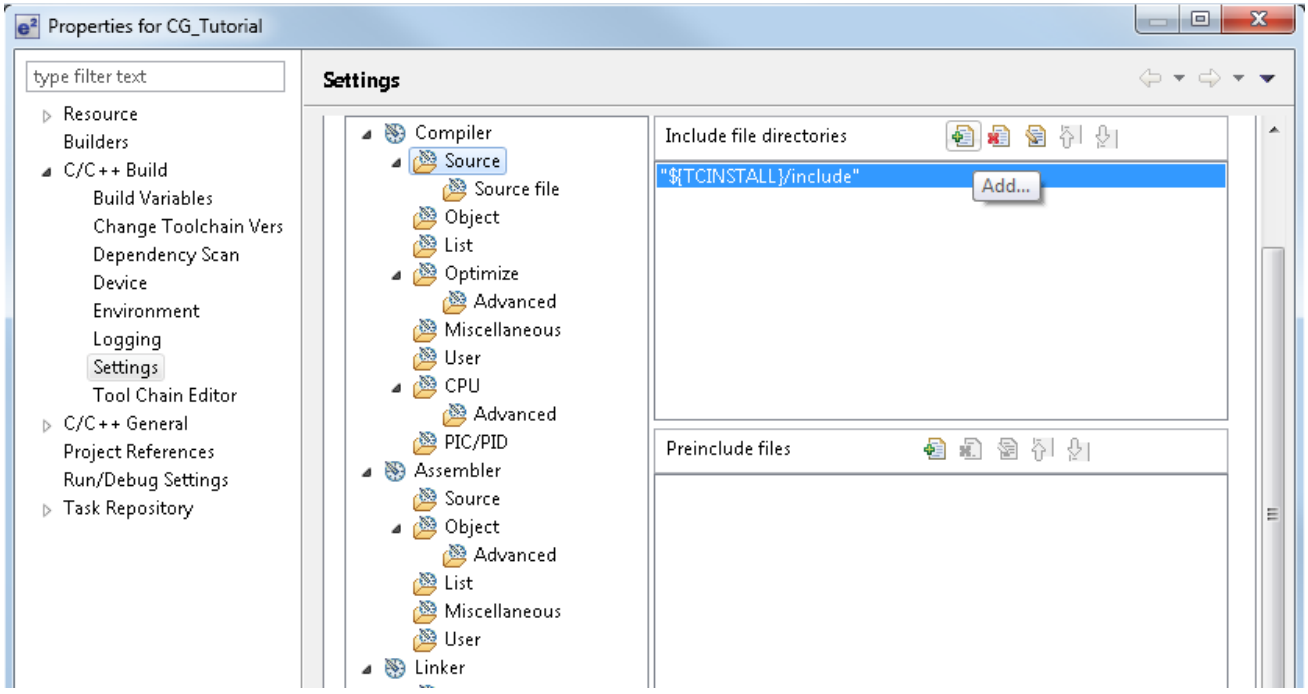


Figure 5-2 Adding additional search paths

In the ‘Add directory path’ dialog, click the ‘Workspace’ button and in the ‘Folder selection’ dialog browse to the ‘CG_Tutorial/src’ folder and click ‘OK’. e^2 studio formats the path as show in Figure 5-3 below.

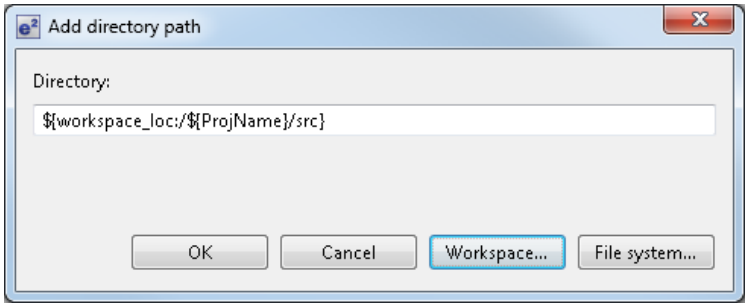


Figure 5-3 Adding workspace search path

Repeat the above steps to add the ‘src/cg_src’ workspace search path and press OK to exit the Properties dialogue.

Select ‘Build Project’ from the ‘Project’ menu, or use the  button. e^2 studio will build the project with no errors.

The project may now be run using the debugger as described in §6.

5.5 Linker Section Addresses for Release configuration

Code Generator makes changes to Linker Section addresses while generating code. These changes are only performed on the build configuration currently selected.

The steps followed above will create a working ‘HardwareDebug’ build configuration. Follow the steps below to create a working ‘Release’ build configuration. For details of the differences in these build configurations please see Section 2.

Select the ‘Release’ Build configuration by clicking:

Project > Build Configurations > Set Active > Release (Release – No Debug).

In the Project Explorer tree, right click the entry ‘Code Generator’ and select ‘Generate Code’. This will run an update for the generated code and make the required changes to Linker Section addresses.

6. Debugging the Project

In the Project Explorer pane, ensure that the ‘CG_Tutorial’ project is selected. To debug the project, click the  button. The dialog shown in Figure 6-1 will be displayed.

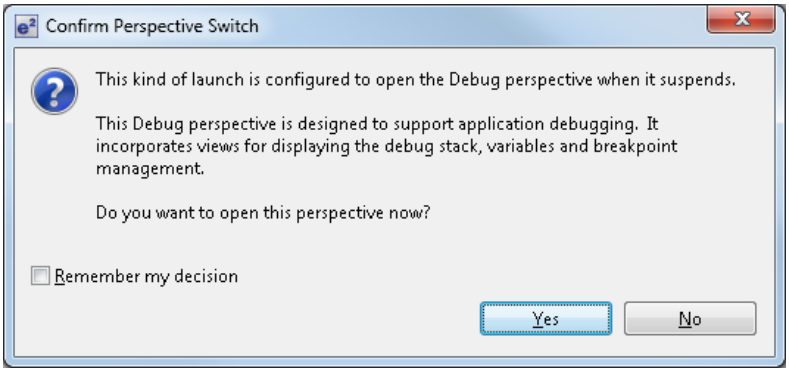


Figure 6-1 Perspective Switch Dialog

Click ‘OK’ to confirm that the debug window perspective will be used.

The debugger will start up and the e^2 studio will show the Code Generator function ‘PowerOn_Reset’.

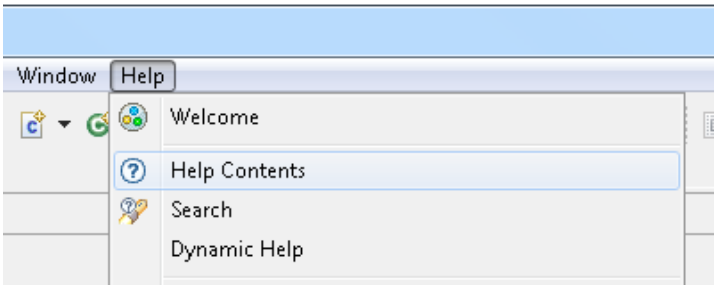
Click the ‘Resume’  button. The debugger will stop again at the beginning of the main() function.
Press  again to run the code.

The program will display ’RSK RX113 - Tutorial - Press 'c' or SW3 for ADC Conversion’ on the serial terminal and ‘TUTOR’ on the bottom line of the LCD. Pressing SW3 or entering the character ‘C’ in the serial terminal window will trigger an ADC conversion and display the resulting value on the terminal window and the LCD

For more information on the e^2 studio debugger refer to the Tutorial manual.

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 RX113 group microcontroller refer to the RX113 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 8 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

REVISION HISTORY	RSK RX113 Code Generator Tutorial Manual (e2 studio)

Rev.	Date	Description	
		Page	Summary
1.00	Nov 30, 2014	—	First Edition issued

Renesas Starter Kit Manual: Code Generator Tutorial Manual

Publication Date: Rev. 1.00 Nov 30, 2014

Published by: Renesas Electronics Corporation

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.
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

RX113 Group

Renesas Electronics Corporation

R20UT3255EG0100