

RX71M Group

Renesas Starter Kit+ Code Generator Tutorial Manual For e² studio

RENESAS MCU RX Family / RX700 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).

Rev. 1.00 Jan 2015

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.

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 Application Leading Tool (Code Generator) for RX together with the e² 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 RX71M 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 RX71M 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+RX71M User's Manual	R20UT3217EG
Tutorial	Provides a guide to setting up RSK+ environment, running sample code and debugging programs.	RSK+RX71M Tutorial Manual	R20UT3222EG
Quick Start Guide	Provides simple instructions to setup the RSK+ and run the first sample.	RSK+RX71M Quick Start Guide	R20UT3223EG
Code Generator Tutorial	Provides a guide to code generation in the e ² studio IDE.	RSK+RX71M Code Generator Tutorial Manual	R20UT3224EG
Schematics	Full detail circuit schematics of the RSK+.	RSK+RX71M Schematics	R20UT3216EG
Hardware Manual	Provides technical details of the RX71M microcontroller.	RX71M Group Hardware Manual	R01UH0493 EJ

2. List of Abbreviations and Acronyms

Abbreviation	Full Form
ADC	Analog-to-Digital Converter
API	Application Programming Interface
bps	Bits per second
CMT	Compare Match Timer
COM	COMmunications port referring to PC serial port
CPU	Central Processing Unit
DVD	Digital Versatile Disc
E1	Renesas On-chip Debugging Emulator
GUI	Graphical User Interface
IDE	Integrated Development Environment
IRQ	Interrupt Request
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LSB	Least Significant Bit
LVD	Low Voltage Detect
MCU	Micro-controller Unit
MSB	Most Significant Bit
PC	Personal Computer
Pmod [™]	This is a Digilent Pmod [™] Compatible connector. Pmod [™] is registered to <u>Digilent Inc.</u> Digilent-Pmod_Interface_Specification
PLL	Phase-locked Loop
RAM	Random Access Memory
ROM	Read Only Memory
RSK+	Renesas Starter Kit+
RTC	Realtime Clock
SAU	Serial Array Unit
SCI	Serial Communications Interface
SPI	Serial Peripheral Interface
TAU	Timer Array Unit
TFT	Thin Film Transistor
TPU	Timer Pulse Unit
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
WDT	Watchdog timer

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

Table of Contents

1. O 1.1 1.2	Verview Purpose Features	7
2. In	ntroduction	8
	roject Creation with e ² studio	
3.1 3.2	Introduction Creating the Project	
	ode Generation Using the e ² studio plug in	
4.1	Introduction	
4.2	Code Generator Tour	
4.3	Code Generation	
4.4	Building the Project	
5. U	ser Code Integration	
5.1	LCD Code Integration	
5.2	Additional include paths	
5.3	Switch Code Integration	
5.4	Debug Code Integration	41
5.5	UART Code Integration	41
5.6	LED Code Integration	
6. D	ebugging the Project	47
7. Ad	dditional Information	49

RENESAS

RSK+RX71M

RENESAS STARTER KIT+

1.1 Purpose

This RSK+ is an evaluation tool for Renesas microcontrollers. This manual describes how to the e² studio IDE code generator plug in to create a working project for the RSK+ platform.

1.2 Features

This RSK+ provides an evaluation of the following features:

- Project Creation with e² 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 e² studio IDE to create a working project for the RSK+ platform. The tutorials help explain the following:

- Project generation using the e² studio
- Detailed use of the code generator plug in for e² studio
- Integration with custom code
- Building the project e² 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 RX71M

These tutorials are designed to show you how to use the RSK+ and are not intended as a comprehensive introduction to the e² studio debugger, compiler toolchains or the E1 emulator. Please refer to the relevant user manuals for more indepth 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 RX71M MCU, 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'.

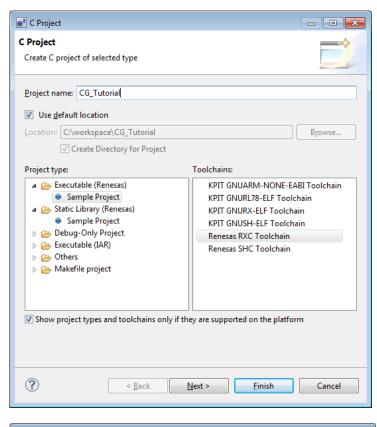
• Create a new C project by rightclicking in the Project Explorer pave and selecting 'New -> C Project' as shown. Alternatively, use the menu item 'File -> New -> C Project'.





 Enter the project name '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 R5F571MLCxFC MCU is found under RX700 -> RX71M -> RX71M -176 pin.
- Click 'Next'.

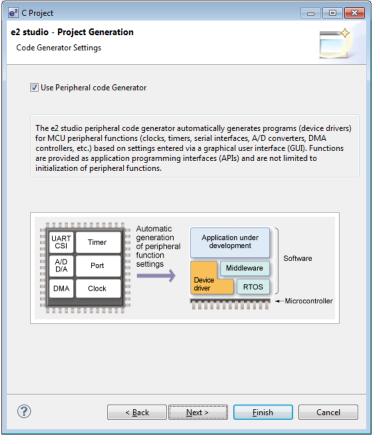


e ² C Project				
e2 studio - Project Gene Select Target Specific Setti				
Toolchain Version : Debug Hardware: Data endian : Select Target:	v2.03.00 E1 Little-endian data R5F571MLCxFC 	· · · · · · · · · · · · · · · · · · ·		
Select Configurations: Image: Image				
Build configurations will be created 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 (RX700) the debug hardware (E1) and debug target (RSF571MLCxFC), debug configuration will be automatically created for you.				
?	< Back Next >	<u>Finish</u> Cancel		



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

3. Project Creation with e² studio



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

e² C F	Project				
e2 st	udio - Project Generation				
Sele	ct 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 allow	wed as a result			
	Replace from int with short	:			
	Enum size is made the sma	llest			
	Pack structures, unions and	d classes			
	🔲 Use try, throw and catch of	C++			
	Use dynamic cast and typeid of C++				
	Saves and restores ACC using the interrupt function				
?	< <u>B</u> ack	<u>N</u> ext > <u>F</u> inish Cancel			

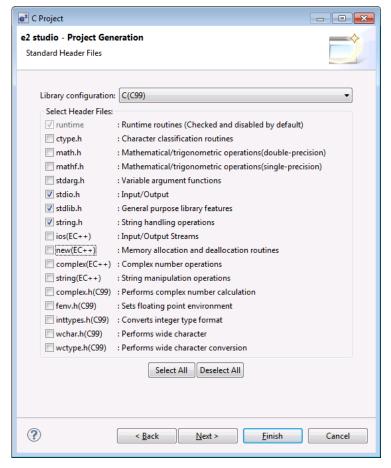


- In the 'Global Options Settings' leave everything at default values.
- Click 'Next'.

3. Project Creation with e² studio

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

e ² C Project		
e2 studio - Project Generation Global Options Settings		
Patch code generation	None	
Fast interrupt vector register:	None	-
ROM:	None	•
RAM:	None	•
Address (H'):	0000000	
Address Register:	None	-
?	Back Next > Einish	Cancel





- In the next dialog, untick all check boxes except 'I/O Register Definition Files' as shown opposite. Click 'Finish'.
- e² 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 V I/O Register Definition Files Generate Hardware Setup Function None -? < <u>B</u>ack <u>N</u>ext > <u>F</u>inish Cancel
- A summary dialog will appear, click 'OK' to complete the project generation.

Su	immary		×
	Project Summary:		
	PROJECT GENERA	TOR	
	PROJECT NAME :	CG_Tutorial	
	PROJECT DIRECTORY :	C:\workspace\CG_Tutorial	
	CPU SERIES : CPU TYPE :	RX700 RX71M	
	TOOLCHAIN NAME :	RATIM Renesas RXC Toolchain	
		v2.02.00	
	TOOLCHAIN VERSION.	¥2.02.00	=
	GENERATION FILES :		=
	C:\workspace\CG_Tutoria	l\src\CG_Tutorial.c	
	Main Program C:\workspace\CG_Tutoria	Nara) dhaat a	
	Setting of B and R sect		
	C:\workspace\CG_Tutoria		
	Aliases of Integer Type		
	C:\workspace\CG_Tutoria	l\src\iodefine.h	-
	*		Þ.
	Click OK to generate the pr	oject or Cancel to abort	
	circk on to generate the pr	oject of cancer to abort.	
		OKCan	cel

3. Project Creation with e² studio



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 RX71M. When using Code Generator, the user is able to configure various MCU features and operating parameters using intuitive GUI controls, thereby bypassing the need in most cases to refer to sections of the Hardware Manual.

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 then free to add custom code to meet their specific requirement. Custom code should be added, whenever possible, in 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. This is useful if, after adding custom code, the user needs to re-visit Code Generator to change any MCU operating parameters.

By following the steps detailed in this Tutorial, the user will generate a e^2 studio project called CG_Tutorial. The fully completed Tutorial project is contained on the DVD and may be imported into e^2 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 e^2 studio.

The CG_Tutorial project uses interrupts for switch inputs, the ADC module, the Compare Match Timer (CMT), the Serial Communications Interface (SCI) and uses these modules to perform A/D conversion and display the results via the Virtual COM port to a terminal program and also on the LCD display on 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 their own code to the user code areas provided by the code generator.

The Code Generator installer is contained on the DVD. This installer must be run before proceeding to the next section.

4.2 Code Generator Tour

In this section a brief tour of Code Generator is presented. For further details of the Code Generator paradigm and reference, refer to the Application Leading Tool Common Operations manual (r20ut2663ej0100_Code Generator.pdf). 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'.



e ² Open Perspective	- • •
C/C++ (default) Code Generator CVS Repository Exploring CVS Repository Exploring Code Git Planning Resource SVN Repository Exploring CVN Repository Exploring Team Synchronizing	
ОК	Cancel

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.

e ² Code Generator - e2 studio					
Eile Edit Navigate Search Project Run Window Help					
■ • 🖩 🦷 🔺 🖬 🖬 🕼 🗐 💋 💷 🔅 •	• • • • • • • • • • • • • • •	 	😭 📴 C/C++ 🖳 Code Generator		
🎦 Project Explorer 🛛 🕞 🔄 🌍 🍸 🗖 🗖	🔛 Peripheral Functions 🙁 🔙 Code Preview	/ 🔲 Properties	🐻 Generate Code 💿 🔽 🗖		
🖬 custom.bat	Clock setting On-chip debug setting Block diagram				
makefile.init	- Main clock oscillator and RTCMCLK setting				
Code Generator	Operation Operation				
Pin View	Main clock oscillator forced oscillation (only for RTC, software standby and deep software standby mode)				
Peripheral Functions	Main clock oscillator forced oscillation (c	only for RTC, software standby and deep software standby mode)			
Clock Generator	Main clock oscillation source	Resonator -	E		
 Voltage Detection Circuit Clock Frequency Accuracy Measur 	Frequency	24 (MHz)			
Low Power Consumption	Oscillator wait time	11000 (µs) (Actual value: 11090.909 µs)			
Interrupt Controller Unit	Oscillation stop detection function	Disabled			
Buses	Oscillation stop detection function	Uisabled			
DMA Controller	- PLL circuit setting				
 Data Transfer Controller Event Link Controller 	Operation				
VO Ports	PLL clock source	Main clock oscillator			
Multi-Function Timer Pulse Unit 3	Input frequency division ratio	x1 v			
Port Output Enable 3	Input frequency division ratio				
b Seneral PWM Timer	Frequency multiplication factor	x 10.0 👻			
b 16-Bit Timer Pulse Unit	Frequency	240 (MHz)			
Programmable Pulse Generator					
⊳ 📦 8-Bit Timer 🗧	- Sub-clock oscillator and RTC (RTCSCLK) settin	ng			
Compare Match Timer Compare Match Timer W	Operation				
Realtime Clock	Sub-clock oscillator drive capacity	Drive capacity for low CL 👻			
Watchdog Timer	Frequency	32.768 (kHz)			
Independent Watchdog Timer	Oscillator wait time	2252.73 (ms) (Actual value: 2296.182 ms)	-		
Serial Communications Interface					
Serial Communications Interface w	📮 Console 💥 🖹 Problems		▙ ૹ૾ૻ૽ ૻ		
12C Bus Interface	Code Generator Console				
 Serial Peripheral Interface CRC Calculator 	Code Generator Console				
▶ ■ 12-Bit A/D Converter					
■ 12-Bit D/A Converter					
Data Operation Circuit					
Code Preview			-		
×	*		•		
0 items selected	CG Tuto	rial/Code Generator/Peripheral Functions/Clock Generator			

Figure 4-2 Initial View



Code Generator provides GUI features for configuration of MCU sub systems. Once the user has configured all required MCU sub systems and peripherals, the user can click the 'Generate Code' button, resulting in a fully configured e² studio project that builds and runs without error.

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 sub-sections, the reader is guided through the steps to configure the MCU for a simple project containing interrupts for switch inputs, timers, ADC and a UART.

4.3.1 Clock Generator

Figure 4-3 shows a screenshot of Code Generator with the Clock Generator function open. Click on the 'Clock setting' sub tab. Configure the system clocks as shown in the figure. In this tutorial we are using the on board 24 MHz crystal resonator for our main clock oscillation source and the PLL circuit is in operation. The PLL output is used as the main system clock and the divisors should be set as shown in Figure 4-3.



💯 Peripheral Functions 🖾 🛃 Code Preview 📃	Properties	🐻 Gen	erate Code 🗕 🏾 🗖 🗖
Clock setting Block diagram			A
- Main clock oscillator and RTCMCLK setting			
	DIC 0		
Main clock oscillator forced oscillation (only for		andby and deep software s	xandby modej
Main clock oscillation source	Resonator		-
Frequency	24		(MHz)
Oscillator wait time	11000	(μs) (Actual valu	ie: 11090.909 μs)
Oscillation stop detection function	Disabled		•
- PLL circuit setting			
Operation	A. 1. 1. 1. 11		
PLL clock source	Main clock oscill	ator	•
Input frequency division ratio	×1	•	
Frequency multiplication factor	x 10.0	•	
Frequency	240	(MHz)	
- Sub-clock oscillator and RTC (RTCSCLK) setting			
Operation Sub-clock oscillator drive capacity	Drive capacity fo	or low CL	-
Frequency	32.768		(kHz)
Oscillator wait time	2252.73	(ms) (Actual value:	: 2296.182 ms)
	2202.10		2200.102 may
 High speed clock oscillator (HOCO) setting Operation 			
Frequency	16		▼ (MHz)
-Low speed clock oscillator (LOCO) setting			
Operation			=
Frequency	240		(kHz)
- IWDT-dedicated low-speed clock oscillator (IWDTLOC	0) setting		
Operation	-		
Frequency	120		(kHz)
 RTC clock setting Operation 			
Clock source	Sub-clock oscilla	ator	-
System clock setting Clock source	PLL circuit		-
System clock (ICLK)	x1 -	240	(MHz)
Peripheral module clock (PCLKA)			
	× 1/2 •	· 120	(MHz)
Peripheral module clock (PCLKB)	x 1/4 -	· 60	(MHz)
Peripheral module clock for ADC (PCLKC)	x 1/4 +	· 60	(MHz)
Peripheral module clock for ADC (PCLKD)	× 1/4 +		(MHz)
External bus clock (BCLK)	x 1/4 🗸	· 60	(MHz)
Flash IF clock (FCLK)	x 1/4 🗸		(MHz)
USB clock (UCLK)	x1/5 👻	48	(MHz)
- BCLK pin output setting			
Operation	Enable BCLK	forced output	
Clock output source	BCLK		
- SDCLK pin output setting			
Operation			
•			

Figure 4-3 Clock setting tab

Proceed to the next section on the Interrupt Controller Unit.

4.3.2 Interrupt Controller Unit

Referring to the RSK+ schematic, SW1 is connected to IRQ5 (P15) and SW2 is connected to IRQ2 (P12). SW3 is connected to directly to the ADTRG0n and will be configured later in §4.3.4. Navigate to the 'Interrupt Controller Unit' node in Code Generator and in the 'General' tab, configure these two interrupts as falling edge triggered as shown in Figure 4-4 below.

归 Peripheral Functions 🛛	🛃 Code Preview 🛛 🗉	Properties			🔞 Generate Coo	de 🙆	~ -	
General Group Interrupts	Interrupt B/A selection							
- Fast interrupt setting								
Fast interrupt	Interrupt source	BSC (BUSERR vect=16)	-					
– Software interrupt setting –								-
Software interrupt	Priority	Level 15 (highest)	•					
Software interrupt 2	Priority	Level 15 (highest)	•					
- NMI setting								-
NMI pin interrupt	Valid edge	Falling -	Digital filter	No filter	- 0	(MHz)		
-IRQO setting								
IRQ0	Pin	P30 -	Digital filter	No filter	- 0	(MHz)		
	Valid edge	Low level -	Priority	Level 15 (highest)	-			
-IRQ1 setting								
IRQ1	Pin	P31 -	Digital filter	No filter	- 0	(MHz)		
	Valid edge	Low level -	Priority	Level 15 (highest)	-			
-IRQ2 setting								
✓ IRQ2	Pin	P12 -	Digital filter	No filter	▼ 0	(MHz)		
	Valid edge	Falling 🗸	Priority	Level 15 (highest)	•			
-IRQ3 setting								
IRQ3	Pin	P33	Digital filter	No filter	- 0	(MHz)		
	Valid edge	Low level	Priority	Level 15 (highest)	-			
-IRQ4 setting								
IRQ4	Pin	PB1 -	Digital filter	No filter	- 0	(MHz)		
	Valid edge	Low level -	Priority	Level 15 (highest)	-			E
-IRQ5 setting								
₩ IRQ5	Pin	P15 -	Digital filter	No filter	• 0	(MHz)		
	Valid edge	Falling -	Priority	Level 15 (highest)	•			
IDOC III								

Figure 4-4 Interrupt Functions tab

Navigate to the 'Group Interrupts' sub tab and ensure that the 'Group BL0' interrupt is selected as shown in Figure 4-5. The Group BL0 interrupt is used for SCI Transmit End Interrupts (TEI) and Reception Error Interrupts (ERI) as described in §4.3.5.



🕎 *Peripheral Functions 🖾	🧾 Code Preview 🛛	Properties
General Group Interrupts	Interrupt B/A selection	
- Group BEO setting		
C Group BEO	Priority	Level 15 (highest) 🚽
- Group BLO setting		
📝 Group BLO	Priority	Level 15 (highest) 🚽
- Group BL1 setting		
Croup BL1	Priority	Level 15 (highest)
- Group ALO setting		
Croup AL0	Priority	Level 15 (highest) 👻
- Group AL1 setting		
C Group AL1	Priority	Level 15 (highest) 🚽

Figure 4-5 Group Interrupt Functions tab

4.3.3 Compare Match Timer

Navigate to the 'Compare Match Timer' node in Code Generator. CMT0 will be used as an interval timer for generation of accurate delays. CMT1 and CMT2 will be used as timers in de-bouncing of switch interrupts.

In the 'CMT0' sub-tab configure CMT0 as shown in Figure 4-6. This timer is configured to generate a High priority interrupt every 1ms. We will use this interrupt later in the tutorial to provide an API for generating high accuracy delays required in our application.

🕎 Peripheral Functions 🛛	🛃 Code Preview 🛛 🔲	Properties		뚢 Generate Code	0		
CMTO CMT1 CMT2 C	MT3						-
- Compare match timer opera	tion setting						_
🔘 Unused		Used					H
- 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 mat	ch interrupt (CMI0)						
Priority		Level 10	•				
							Ŧ
•	III					1	¢.

Figure 4-6 CMT0 tab

Navigate to the 'CMT1' sub-tab and configure CMT1 as shown in Figure 4-7. This timer is configured to generate a High priority interrupt after 20ms. This timer is used as our short switch de-bounce timer later in this tutorial.

RENESAS

归 Peripheral Functions 🛛	🧾 Code Preview 🛛 🔲	Properties		뚢 Generate Code	0	~	
CMTO CMT1 CMT2 CM	ИТЗ						
- Compare match timer operati	ion setting						-
🔘 Unused		Used					=
- Count clock setting							-
PCLK/8	PCLK/32	PCLK/128	PCLK/512				
- Interval value setting							_
Interval value		20	ms 🔻	(Actual value: 20)			
-Interrupt setting							_
📝 Enable compare mate	ch interrupt (CMI1)						
Priority		Level 10	•				
							-
•						- F	

Figure 4-7 CMT1 tab

Navigate to the 'CMT2' sub-tab and configure CMT2 as shown in Figure 4-8. This timer is configured to generate a High priority interrupt after 200ms. This timer is used as our long switch de-bounce timer later in this tutorial.

归 Peripheral Functions 🔅	🧾 🧾 Code Preview 📃	Properties		🐻 Generate Code 🛛 🙆		
CMT0 CMT1 CMT2 C	СМТЗ					
- Compare match timer opera	ition setting					-
🔘 Unused		Osed				=
- Count clock setting						-
PCLK/8	PCLK/32	PCLK/128	PCLK/512			
-Interval value setting						_
Interval value		200	ms 🔻	(Actual value: 200.004267)		
- Interrupt setting						_
👿 Enable compare ma	tch interrupt (CMI2)					
Priority		Level 10	-			
						Ŧ
•					+	

Figure 4-8 CMT2 tab

4.3.4 12-bit A/D Converter

Navigate to the '12-bit A/D Converter' tab in Code Generator. Refer to the screenshot shown in Figure 4-9 and configure the S12AD0 as shown. We will be using the S12AD0 in 12-bit one shot mode on the AN000 input, which is connected to the RV1 potentiometer output on the RSK. The conversion start trigger will be via the pin connected to SW3.

12AD0 \$12AD1									
Setting 1 Setting 2									
612AD0 operation setting									
 Unused 		O Used							
		0360							
Iperation mode setting									
Single scan mode		Group scan mode			Cont	tinuous scan mi	ode		
ouble trigger mode setting									
Oisable		💿 Enable							
elf diagnosis setting									
Mode		Unused	•						
Voltage used		Use VREFH0x0	-						
isconnection detection assist set	ting								
Charge setting		Unused							
Period		1 ADCLK	-						
iroup scan priority setting									
Group A priority		Group A without priority			-				
Group B action		Not restarted or continued d	ue to Gr	oup A	v priority 👻				
VD converted value count setting									
Addition mode		Average mode							
nalog input channel setting		~ -							
	Convert (Group A)	Convert (Group B)	٨٦	1/4	ware AD usive	Dediester	d a smala	and h	ald
		Convert (Group B)		JAVE	erage AD value	Dedicated	a sample	and n	ola
AN000									
AN001									
AN002									
AN003									
AN004									
AN005									
AN006									
AN007									
Conversion start trigger setting –									
Conversion start trigger (Group									
A/D conversion start trigger pi	in							•	
Conversion start trigger (Group	5 B)								
Conversion start trigger (Group Compare match with or input		RA						Ŧ	
Compare match with or input			-					Ŧ	
		RA P07	•					T	
Compare match with or input ADTRG0# pin selection			•					-	
Compare match with or input ADTRG0# pin selection	capture to MTU0.TG		•			·		-	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition o	capture to MTU0.TG	P07 1-time conversion				•		Ţ	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement	capture to MTU0.TG	P07 1-time conversion Right-alignment	•					-	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition o	capture to MTU0.TG	P07 1-time conversion						-	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement	capture to MTU0.TG	P07 1-time conversion Right-alignment						-	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing	•			▼		-	
Compare match with or input ADTRGO# pin selection Nata registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Nedicated sample and hold circuit	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy	•			•		-	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing	•	[μs]	(The input value i	•		-	
Compare match with or input ADTRGO# pin selection All registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy	•	(μs)	(The input value i	•		V	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy	•		(The input value i (Actual value: 3.6	• is invalid.)		Y	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time AN000 / Self-diagnosis conversion Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8	•			• is invalid.)		Y	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time N0000 / Self-diagnosis conversion Input sampling time N001 conversion time setting —	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667	· ·	(μs)	(Actual value: 3.6	s invalid.)			
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time NN000 / Self-diagnosis conversion Input sampling time NN001 conversion time setting Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8	· ·	(μs)		s invalid.)		•	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time NN000 / Self-diagnosis conversion Input sampling time NN001 conversion time setting Input sampling time NN002 conversion time setting	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667		(μs) (μs)	(Actual value: 3.6 (Actual value: 3.6	• is invalid.) 667)		•	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time NN000 / Self-diagnosis conversion Input sampling time NN001 conversion time setting Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667		(μs)	(Actual value: 3.6	• is invalid.) 667)		¥	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time NN000 / Self-diagnosis conversion Input sampling time NN001 conversion time setting Input sampling time NN002 conversion time setting Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667		(μs) (μs)	(Actual value: 3.6 (Actual value: 3.6	• is invalid.) 667)		•	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time NN000 / Self-diagnosis conversion Input sampling time NN001 conversion time setting Input sampling time NN002 conversion time setting Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667		(μs) (μs)	(Actual value: 3.6 (Actual value: 3.6	• s invalid.) 667) 667)		•	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time AN000 / Self-diagnosis conversion Input sampling time AN001 conversion time setting Input sampling time AN002 conversion time setting Input sampling time AN003 conversion time setting Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667		(μs) (μs) (μs)	(Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	• s invalid.) 667) 667)		•	
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time AN000 / Self-diagnosis conversion Input sampling time AN002 conversion time setting Input sampling time AN003 conversion time setting Input sampling time AN003 conversion time setting Input sampling time AN004 conversion time setting	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667		(μs) (μs) (μs) (μs)	(Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 s invalid.) 667) 667) 667) 		•	
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time AN000 / Self-diagnosis conversion Input sampling time AN002 conversion time setting Input sampling time AN003 conversion time setting Input sampling time AN004 conversion time setting Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667		(μs) (μs) (μs) (μs)	(Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 s invalid.) 667) 667) 667) 		•	
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time N000 / Self-diagnosis conversion Input sampling time N001 conversion time setting Input sampling time N003 conversion time setting Input sampling time N004 conversion time setting Input sampling time N004 conversion time setting Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667 3.667		(µs) (µs) (µs) (µs)	(Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	• invalid.) 677 677 677 677 677		•	
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time AN000 / Self-diagnosis conversion Input sampling time AN002 conversion time setting Input sampling time AN003 conversion time setting Input sampling time AN004 conversion time setting Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667		(μs) (μs) (μs) (μs)	(Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	• invalid.) 677 677 677 677 677		•	
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time N000 / Self-diagnosis conversion Input sampling time N001 conversion time setting Input sampling time N003 conversion time setting Input sampling time N004 conversion time setting Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667 3.667		(µs) (µs) (µs) (µs)	(Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	• invalid.) 677 677 677 677 677		•	
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time NN001 conversion time setting Input sampling time NN002 conversion time setting Input sampling time NN004 conversion time setting Input sampling time NN005 conversion time setting Input sampling time NN005 conversion time setting Input sampling time NN005 conversion time setting Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667 3.667		(µs) (µs) (µs) (µs)	(Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 sinvalid.) 67) 667) 667) 667) 667) 667) 667) 		v	
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time N0001 conversion time setting Input sampling time N0002 conversion time setting Input sampling time N0004 conversion time setting Input sampling time N0005 conversion time setting Input sampling time N0006 conversion time setting Input sampling time N0006 conversion time setting Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667 3.667 3.667		(µs) (µs) (µs) (µs)	(Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 sinvalid.) 67) 667) 667) 667) 667) 667) 667) 		v	
Compare match with or input ADTRQU# pin selection Aat registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Nedicated sample and hold circuit Input sampling time N000 / Self-diagnosis conversion Input sampling time N001 conversion time setting Input sampling time N003 conversion time setting Input sampling time N004 conversion time setting Input sampling time N005 conversion time setting Input sampling time N005 conversion time setting Input sampling time N006 conversion time setting Input sampling time N006 conversion time setting Input sampling time N006 conversion time setting N006 conversion time setting N006 conversion time setting Input sampling time N006 conversion time setting Input sampling time N007 conversion time setting	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667 3.667 3.667 3.667		(µs) (µs) (µs) (µs) (µs) (µs)	[Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 sinvalid.) 67) 667) 667) 667) 667) 670 671 673 674 675 675 675 676 676 677 <l< td=""><td></td><td>v</td><td></td></l<>		v	
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time N0001 conversion time setting Input sampling time N0002 conversion time setting Input sampling time N0004 conversion time setting Input sampling time N0005 conversion time setting Input sampling time N0006 conversion time setting Input sampling time N0007 conversion time setting Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667 3.667 3.667		(µs) (µs) (µs) (µs)	[Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 sinvalid.) 67) 667) 667) 667) 667) 670 671 673 674 675 675 675 676 676 677 <l< td=""><td></td><td>Y</td><td></td></l<>		Y	
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time NN001 conversion time setting Input sampling time NN002 conversion time setting Input sampling time NN004 conversion time setting Input sampling time NN005 conversion time setting Input sampling time NN006 conversion time setting Input sampling time NN007 conversion time setting Input sampling time	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667 3.667 3.667 3.667 3.667 3.667 3.667		(µs) (µs) (µs) (µs) (µs) (µs)	[Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 sinvalid.) 67) 667) 667) 667) 667) 670 671 673 674 675 675 675 676 676 677 <l< td=""><td></td><td>· · · · · · · · · · · · · · · · · · ·</td><td></td></l<>		· · · · · · · · · · · · · · · · · · ·	
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time AN000 / Self-diagnosis conversion Input sampling time AN001 conversion time setting Input sampling time AN003 conversion time setting Input sampling time AN004 conversion time setting Input sampling time AN005 conversion time setting Input sampling time AN005 conversion time setting Input sampling time AN006 conversion time setting Input sampling time AN006 conversion time setting	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667 3.667 3.667 3.667		(µs) (µs) (µs) (µs) (µs) (µs)	[Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 sinvalid.) 67) 667) 667) 667) 667) 670 671 673 674 675 675 675 676 676 677 <l< td=""><td></td><td>Y</td><td></td></l<>		Y	
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time NN000 / Self-diagnosis conversion Input sampling time NN001 conversion time setting Input sampling time NN002 conversion time setting Input sampling time NN005 conversion time setting Input sampling time NN005 conversion time setting Input sampling time NN006 conversion time setting Input sampling time NN007 conversion time setting Input sampling time Conversion time setting Total conversion time (Group A	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667 3.667 3.667 3.667 3.667 3.667 3.667		(µs) (µs) (µs) (µs) (µs) (µs) (µs)	[Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 sinvalid.) 67) 667) 667) 667) 667) 670 671 673 674 675 675 675 676 676 677 <l< td=""><td></td><td>Y</td><td></td></l<>		Y	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time AN000 / Self-diagnosis conversion Input sampling time AN002 conversion time setting Input sampling time AN003 conversion time setting Input sampling time AN004 conversion time setting Input sampling time AN005 conversion time setting Input sampling time AN006 conversion time setting Input sampling time AN006 conversion time setting Input sampling time AN006 conversion time setting Input sampling time AN007 conversion time setting Input	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667 3.667 3.667 3.667 3.667 3.667 3.667		(µs) (µs) (µs) (µs) (µs) (µs)	[Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 sinvalid.) 67) 667) 667) 667) 667) 670 671 671 673 674 675 <l< td=""><td></td><td>· · · · · · · · · · · · · · · · · · ·</td><td></td></l<>		· · · · · · · · · · · · · · · · · · ·	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time N000 / Self-diagnosis conversion Input sampling time N001 conversion time setting Input sampling time N002 conversion time setting Input sampling time N003 conversion time setting Input sampling time N004 conversion time setting Input sampling time N005 conversion time setting Input sampling time N005 conversion time setting Input sampling time N006 conversion time setting Input sampling time N006 conversion time setting Input sampling time N007 conversion time setting Input sampling time N007 conversion time setting Total conversion time (Group E (Note: Continuous sampling is	capture to MTU0.TG	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667 3.667 3.667 3.667 3.667 3.667 3.667		(µs) (µs) (µs) (µs) (µs) (µs) (µs)	[Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 sinvalid.) 67) 667) 667) 667) 667) 670 671 671 673 674 675 <l< td=""><td></td><td>· · ·</td><td></td></l<>		· · ·	
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time AN000 / Self-diagnosis conversion Input sampling time AN001 conversion time setting Input sampling time AN002 conversion time setting Input sampling time AN003 conversion time setting Input sampling time AN004 conversion time setting Input sampling time AN005 conversion time setting Input sampling time AN005 conversion time setting Input sampling time AN005 conversion time setting Input sampling time AN006 conversion time setting Input sampling time AN007 conversion time setting Total conversion time (Group A Total conversion time (Group A (Note: Continuous sampling is nterrupt setting	capture to MTU0.TG count setting n time setting	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667 3.667 3.667 3.667 3.667 3.667 3.667		(µs) (µs) (µs) (µs) (µs) (µs) (µs)	[Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 sinvalid.) 67) 667) 667) 667) 667) 670 671 671 673 674 675 <l< td=""><td></td><td>· · ·</td><td></td></l<>		· · ·	
Compare match with or input ADTRG0# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time N000 / Self-diagnosis conversion Input sampling time N001 conversion time setting Input sampling time N002 conversion time setting Input sampling time N003 conversion time setting Input sampling time N004 conversion time setting Input sampling time N005 conversion time setting Input sampling time N005 conversion time setting Input sampling time N006 conversion time setting Input sampling time N006 conversion time setting Input sampling time N007 conversion time setting Input sampling time N007 conversion time setting Total conversion time (Group E (Note: Continuous sampling is	capture to MTU0.TG count setting n time setting	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667 3.667 3.667 3.667 3.667 3.667 3.667		(µs) (µs) (µs) (µs) (µs) (µs) (µs)	[Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 sinvalid.) 67) 667) 667) 667) 667) 670 671 671 673 674 675 <l< td=""><td></td><td>· · · · · · · · · · · · · · · · · · ·</td><td></td></l<>		· · · · · · · · · · · · · · · · · · ·	
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time N000 / Self-diagnosis conversion Input sampling time N001 conversion time setting Input sampling time N002 conversion time setting Input sampling time N003 conversion time setting Input sampling time N004 conversion time setting Input sampling time N005 conversion time setting Input sampling time N006 conversion time setting Input sampling time N006 conversion time setting Input sampling time N006 conversion time setting Input sampling time N007 conversion time setting Total conversion time (Group A Total conversion time (Group A Note: Continuous sampling is Note: Continuous sampling is	capture to MTU0.TG count setting n time setting	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667 3.667 3.667 3.667 3.667 3.667 3.667 3.667 3.667		(µs) (µs) (µs) (µs) (µs) (µs) (µs)	[Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 sinvalid.) 67) 667) 667) 667) 667) 670 671 671 673 674 675 <l< td=""><td></td><td>· · · · · · · · · · · · · · · · · · ·</td><td></td></l<>		· · · · · · · · · · · · · · · · · · ·	
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time AN000 / Self-diagnosis conversion Input sampling time AN001 conversion time setting Input sampling time AN002 conversion time setting Input sampling time AN003 conversion time setting Input sampling time AN004 conversion time setting Input sampling time AN005 conversion time setting Input sampling time AN005 conversion time setting Input sampling time AN006 conversion time setting Input sampling time AN007 conversion time setting Input sampling time Conversion time setting Total conversion time (Group A (Note: Continuous sampling is interrupt setting ✓ Enable AD conversion end Priority	capture to MTU0.TG count setting n time setting h time setting k) i disabled) tinterrupt (\$124D10)	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667		(µs) (µs) (µs) (µs) (µs) (µs) (µs)	[Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 sinvalid.) 67) 667) 667) 667) 667) 670 671 671 673 674 675 <l< td=""><td></td><td></td><td></td></l<>			
Compare match with or input ADTRGO# pin selection Data registers setting AD converted value addition of Data placement Automatic clearing Data accuracy Dedicated sample and hold circuit Input sampling time AN000 / Self-diagnosis conversion Input sampling time AN001 conversion time setting Input sampling time AN002 conversion time setting Input sampling time AN003 conversion time setting Input sampling time AN004 conversion time setting Input sampling time AN005 conversion time setting Input sampling time AN006 conversion time setting Input sampling time AN006 conversion time setting Input sampling time AN007 conversion time setting Input sampling time Conversion time setting Total conversion time (Group A (Note: Continuous sampling is interrupt setting Imput setting Imput setting Imput setting	capture to MTU0.TG count setting n time setting h time setting k) i disabled) tinterrupt (\$124D10)	P07 1-time conversion Right-alignment Disable automatic clearing 12-bit accuracy 8 3.667		(µs) (µs) (µs) (µs) (µs) (µs) (µs)	[Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6 (Actual value: 3.6	 sinvalid.) 67) 667) 667) 667) 667) 670 671 671 673 674 675 <l< td=""><td></td><td></td><td></td></l<>			

Figure 4-9 A/D Converter tab



4.3.5 Serial Communications Interface

Navigate to the 'Serial Communications Interface' tab in Code Generator, select the SCI6 sub-tab and apply the settings shown in Figure 4-10. In the RSK+RX71M SCI6 is used as an SPI master for the Okaya Pmod[™] LCD on the PMOD1 connector as shown in the schematic.

🕎 *Peripheral Functions 🙁 🛒 Code Preview 🔲	Properties
SCIO SCI1 SCI2 SCI3 SCI4 SCI5 <u>SCI6</u>	SCI7 SCI12
General setting Setting	
- Function setting	
🔘 Unused	
Asynchronous mode	Transmission 👻
🔿 Asynchronous mode (Multi-processor)	Transmission 👻
Clock synchronous mode	Transmission 💌
Smart card interface mode	Transmission 📼
🔘 Simple IIC bus	
Simple SPI bus	Master transmit only 🔹
– Pin setting	
RXD6/SMISO6/SSCL6	P01 👻
TXD6/SMOSI6/SSDA6	P00 -

Figure 4-10 SCI6 General Setting tab

Select the SCI6 'Setting' sub-tab and configure the SPI Master as illustrated in Figure 4-11. Make sure the 'Transfer direction setting' is set to 'MSB-first' and the 'Bit rate' is set to 1500000. All other settings remain at their defaults.



4. Code Generation Using the e² studio plug in

🚆 Peripheral Functions 🕺 🛒 Code Preview 📲	Properties		🛐 Generate Code 🛛 🔍 🗁	
SCIO SCI1 SCI2 SCI3 SCI4 SCI5 SC	16 SCI7 SCI12			-
General setting Setting				
- Transfer direction setting				
C LSB-first	MSB-first			
- Data inversion setting				=
Normal	Inverted			_
- Transfer rate setting				_
Transfer clock	Internal clock	•	P02 -	
Bit rate	1500000	-	(bps) (Actual value: 1500000, Error : 0%)	
Enable modulation duty correction	255			
SCK6 pin function selection	Clock output	•		
- Clock setting				_
Clock delay	Clock is not delayed	-		
Enable clock polarity inversion				
– Data handling setting				_
Transmit data handling	Data handled in interrupt ser	rvice routi	ne 🔻	
- Interrupt setting				
TXI6 priority	Level 15 (highest)	•		
TEI6, ERI6 priority (Group BL0)	Level 15 (highest)			
– Callback function setting				_
📝 Transmission end				
				-
•				P

Figure 4-11 SCI6 SPI Master Setting

Staying in the 'Serial Communications Interface' tab in Code Generator, select the SCI7 sub-tab and apply the settings shown in Figure 4-12. In the RSK+RX71M SCI7 is connected via a Renesas RL78/G1C to provide a USB virtual COM port as shown in the schematic.



归 Perip	oheral Fu	unction	s 🖾 🕹	🔮 Code	e Previev	v 🔲 F	ropertie	25			
SCIO	SCI1	SCI2	SCI3	SCI4	SCI5	SCI6	<u>SCI7</u>	SCI12			
Gener	al setting	Setti	ng								
– Funct	ion setting	g									
(🔵 Unuse	d									
	Asyncl	hronous	mode				Transn	nission/recepti	ion	•	
0) Asyncl	hronous	mode (N	Multi-prod	cessor)		Transm	nission		-	
0) Clock	synchror	nous mo	de			Transn	nission		-	
0) Smart	card inte	rface mo	ode			Transmission			-	
0) Simple	IIC bus									
() Simple	SPI bus	;				Slave transmit/receive			-	
– Pinise	tting —										
F	XD7/SM	IISO7/S	SCL7				P92			•	
Т	XD7/SM	IOSI7/S	SDA7				P90			•	
•											

Figure 4-12 SCI7 General Setting tab

Select the SCI7 'Setting' sub-tab and configure SCI7 as illustrated in Figure 4-13. Make sure the 'Start bit edge detection' is set as 'Falling edge on RXD7 pin' and the 'Bit rate' is set to 19200 bps. All other settings remain at their defaults.



Peripheral Functions 🛛 📓 Code Preview	w 🔲 Properties	🐻 Generate Code 🛛 💆 💆
CIO SCI1 SCI2 SCI3 SCI4 SCI5	SCI6 SCI7 SCI12	
General setting Setting		
Start bit edge detection setting		
💿 Low level on RXD7 pin	Falling edge on RXD7 pin	
Data length setting		
🔘 9 bits	Ø 8 bits	🔘 7 bits
Parity setting		
None	🔘 Even	Odd 🔿
Stop bit length setting		
I bit	🔘 2 bits	
Transfer direction setting		
SB-first	MSB-first	
Transfer rate setting		
Transfer clock	Internal clock	▼ P91
Bit rate	19200	 (bps) (Actual value: 19230.769, Error: 0.16%)
Enable modulation duty correction	255	
SCK7 pin function	SCK7 is not used	▼
Noise filter setting		
Enable noise filter		
Noise filter clock	Clock signal divided by 1	▼ 60000000 (Hz)
Hardware flow control setting		
-	© CTS	■ DTC
None		🔘 RTS
None CTS7/RTS7 pin	P93	U HIS
CTS7/RTS7 pin		U HIS
CTS7/RTS7 pin Data handling setting	P93	•
CTS7/RTS7 pin Data handling setting Transmit data handling	P93 Data handled in interrupt service	e routine
CTS7/RTS7 pin Data handling setting Transmit data handling Receive data handling	P93	e routine
CTS7/RTS7 pin Data handling setting Transmit data handling Receive data handling Interrupt setting	P93 Data handled in interrupt servic Data handled in interrupt servic	e routine
CTS7/RTS7 pin Data handling setting Transmit data handling	P93 Data handled in interrupt service	e routine
CTS7/RTS7 pin Data handling setting Transmit data handling Receive data handling Interrupt setting	P93 Data handled in interrupt servic Data handled in interrupt servic	e routine
CTS7/RTS7 pin Data handling setting Transmit data handling Receive data handling Interrupt setting TXI7 priority RXI7 priority Q Enable error interrupt (ERI7)	P93 Data handled in interrupt servic Data handled in interrupt servic Level 15 (highest) Level 15 (highest)	e routine
CTS7/RTS7 pin Data handling setting Transmit data handling Receive data handling Interrupt setting TXI7 priority RXI7 priority	P93 Data handled in interrupt servic Data handled in interrupt servic Level 15 (highest)	e routine
CTS7/RTS7 pin Data handling setting Transmit data handling Receive data handling Interrupt setting TXI7 priority RXI7 priority Q Enable error interrupt (ERI7)	P93 Data handled in interrupt servic Data handled in interrupt servic Level 15 (highest) Level 15 (highest)	e routine

Figure 4-13 SCI7 Asynchronous Setting



4.3.6 I/O Ports

Referring to the RSK+ schematic, LED0 is connected to P03, LED1 is connected to P05, LED2 is connected to P26 and LED3 is connected to P27. Navigate to the 'I/O Ports' tab in Code Generator and configure these four I/O lines as shown in Figure 4-14 and Figure 4-15 below. Ensure that the 'Output 1' tick box is checked. This ensures that the code is generated to set LEDs initially off.

Port0 Port1	Port2 Port3	Port4 Port	t5 Port6	Port7 Port8	Port9 F	ortA PortE	B PortC	PortD	PortE	PortF	PortG	PortJ
P00												
💿 Unused	🔘 ln 🥊) 💿 Out 🤇	🌔 🗌 Pi	ull-up CMI	OS output		-	🗌 Outpu	it 1	- High-	drive outpu	ıt
P01												
O Unused Doc	🔘 In	🔘 Out	Pr	ull-up	OS output			📃 Outpu	it 1	- High-	drive outpu	It
P02 O Unused	🔘 In 🧃) 🔘 Out (P	ull-up CMI	DS output		7	🗌 Outpu	ıt 1	🗔 Hiah-	drive outpu	ıt
P03	<u> </u>	0.044	•									
🔘 Unused	🔘 In	💿 Out	P	all-up CMI	OS output	•	•	🔽 Outpu	ıt 1	🗸 High-	drive outpu	ıt
P05												
🔘 Unused	🔘 In) Out	Pt	ull-up CM	OS output	•	•	🔽 Outpu	at 1	🗸 High-	drive outpu	it
- P07	@ In . •	Out	P	dus CM	DS output			🗌 Outpu	4.1	- Liak	drive outpu	ıb
() Unused	🔘 ln 👎) 🔘 Out (an-up CM	utput				10-1		unve outpu	1.
4												•

Figure 4-14 I/O ports – Port0



🕎 Perip	oheral Fu	unctions 🛛 🛛	🕺 Code Previ	۵	Generate Code 🗕 🎽 🗖		
Port0	Port1	Port2 Port3	Port4 Port5	Port6 Port7	Port8 Port9 PortA	PortB PortC PortD	PortE PortF PortG PortJ
– P20 —							
۲	Unused	🔘 In	🔘 Out	🔄 Pull-up	CMOS output	🗾 📃 Output	1 High-drive output
– P21 —							
۲	Unused	🔘 In	🔘 Out	Pull-up	CMOS output	🚽 🗌 Output	1 High-drive output
– P22 —							E
۲	Unused	🔘 In	🔘 Out	Pull-up	CMOS output	- Output	1 High-drive output
– P23 —					·		
۲	Unused	🔘 In	🔘 Out	Pull-up	CMOS output	- Output	1 High-drive output
- P24 —							
۲	Unused	🔘 In	🔘 Out	Pull-up	CMOS output	- Output	1 High-drive output
- P25							
	Unused	🔘 In	🔘 Out	Pull-up	CMOS output		:1 High-drive output
- P26			0.00				
	Unused	🔘 In	💿 Out	Pull-up	CMOS output	- Vitput	1 J High-drive output
-P27	onasea	0	Jour		and a super		
	Unused	🔘 In) Out	Pull-up	CMOS output		1 🔽 High-drive output
	onuseu	0	S Out		Chron output	· Output	ngr anvo okpak
4							

Figure 4-15 I/O ports – Port2

P45 is used as one of the LCD control lines, together with P46 and P47. Configure these lines as shown in Figure 4-16.

🕎 Peripł	heral Fu	unctions 🛿 🚂	🕻 Code Previev	w 🔲 Propertie	es		🐻 Generate Code	0 ⁻
Port0 - P40	Port1	Port2 Port3	Port4 Port5	Port6 Port7	Port8 Port9 Port4	A PortB	PortC PortD PortE PortF F	ortG PortJ
	Jnused	🔘 In 😲	🔘 Out 😲	🗌 Pull-up	CMOS output	¥	🔲 Output 1	
	Jnused	🔘 In	🔘 Out	🔄 Pull-up	CMOS output	¥	🔲 Output 1	E
	Jnused	🔘 In	🔘 Out	🔄 Pull-up	CMOS output	•	🗌 Output 1	
٥ ل	Jnused	🔘 In	🔘 Out	Pull-up	CMOS output	v	🗌 Output 1	
P44 © L P45	Jnused	🔘 In	🔘 Out	Pull-up	CMOS output	v	🗌 Output 1	
	Jnused	🔘 In) Out	🔄 Pull-up	CMOS output	•	🔽 Output 1	
	Jnused	🔘 In) Out	🔄 Pull-up	CMOS output	•	🔽 Output 1	
	Jnused	🔘 In) Out	🔄 Pull-up	CMOS output	•	🔄 Output 1	
								-
•								4

Figure 4-16 I/O ports – Port4

Peripheral function configuration is now complete. Save the project using the File -> Save, then click 'Generate Code'. The Console pane should report 'The operation of generating file was successful', as shown Figure 4-17 below.

🗐 Console 🙁 🔝 Problems 🛛 🕞 🐨 🗉] • 📬 • 🗖 🗖
Code Generator Console	
M0409002:The generating source folder is: C:\Workspace\CG Tutorial\	
M0409001:The following files were generated:	
M0409000:src\cg src\r cg main.c was generated.	
M0409000: <u>src\cg src\r cg dbsct.c</u> was generated.	
M0409000:src\cg src\r cg intprg.c was generated.	
M0409000: <u>src\cg_src\r_cg_resetprg.c</u> _was_generated.	
M0409000:src\cg src\r cg sbrk.c was generated.	
M0409000: <u>src\cg src\r cg vecttbl.c</u> was generated.	
M0409000:src\cg src\r cg sbrk.h was generated.	
M0409000: <u>src\cg src\r cg stacksct.h</u> was generated.	
M0409000: <u>src\cg src\r cg vect.h</u> was generated.	E
M0409000: <u>src\cg src\r cg hardware setup.c</u> was generated.	
M0409000: <u>src\cg src\r cg macrodriver.h</u> was generated.	
M0409000: <u>src\cg src\r cg userdefine.h</u> was generated.	
M0409000: <u>src\cg src\r cg cgc.c</u> was generated.	
M0409000: <u>src\cg src\r cg cgc user.c</u> was generated.	
M0409000: <u>src\cg src\r cg cgc.h</u> was generated.	
M0409000: <u>src\cg src\r cg icu.c</u> was generated.	
M0409000: <u>src\cg src\r cg icu user.c</u> was generated.	
M0409000: <u>src\cg src\r cg icu.h</u> was generated.	
M0409000: <u>src\cg src\r cg port.c</u> was generated.	
M0409000: <u>src\cg src\r cg port user.c</u> was generated.	
M0409000: <u>src\cg src\r cg port.h</u> was generated.	
M0409000: <u>src\cg_src\r_cg_cmt.c</u> _was_generated.	
M0409000: <u>src\cg src\r cg cmt user.c</u> was generated.	
M0409000: <u>src\cg src\r cg cmt.h</u> was generated.	
M0409000: <u>src\cg src\r cg sci.c</u> was generated.	
M0409000: <u>src\cg src\r cg sci user.c</u> was generated.	
M0409000: <u>src\cg src\r cg sci.h</u> was generated.	
M0409000: <u>src\cg src\r cg s12ad.c</u> was generated.	
M0409000: <u>src\cg src\r cg s12ad user.c</u> was generated.	
M0409000: <u>src\cg src\r cg s12ad.h</u> was generated. M0409003:The operation of generating file was successful.	
Meteodestine operation of generating file was successful.	
	-
<	Þ

Figure 4-17 Code generator console



4.4 Building the Project

The project is template created by Code Generator can now be built. In the Project Explorer pane expand the 'src' folder. The 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 4-18. This is because the main() function now resides in r_cg_main.c in the cg_src folder and the type definitions and setting of sections has been handled by the Code Generator.

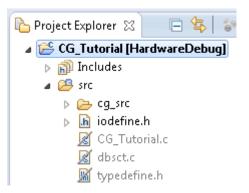


Figure 4-18 Files excluded from the build by Code Generator

Switch back to the 'C/C++' perspective using the $\boxed{\mathbb{C}/C^{++}}$ button on the top right of the e² studio workspace. Use 'Build Project' from the 'Project' menu or the $\boxed{\mathbb{C}}$ button to build the tutorial. The project will build with no errors.



5.User Code Integration

In this section the remaining application code is added to the project. Source files found on the RSK+ DVD are copied into the workspace and the user is directed to add code in the user areas of the code generator files.

Code must be inserted in to the user code area in many files in this project, 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 needs to subsequently change any of the Code Generator-generated code.

5.1 LCD Code Integration

API functions for the Okaya LCD display are provided with the RSK+. Locate the files ascii.h, r_okaya_lcd.h, ascii.c, and r_okaya_lcd.c on the RSK+ DVD. These files can be found in the Tutorial project for e² studio. Copy these files into the C:\Workspace\CG_Tutorial\src directory. The files will be automatically added to the project as shown in Figure 5-1.

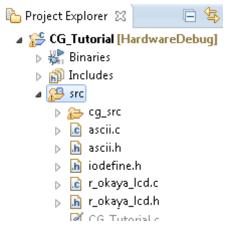


Figure 5-1 Adding files to the project

In the e² studio Project Tree, expand the 'src/cg_src' folder and open the file 'r_cg_userdefine.h' by doubleclicking on it. Insert the following #defines in 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)
/* End user code. Do not edit comment generated here */

In the same folder open the file 'r_cg_main.c' by double-clicking on it. Insert the following code in between the user code delimiter comments as shown below.

/* Start user code for include. Do not edit comment generated here */
#include "r_okaya_lcd.h"
/* End user code. Do not edit comment generated here */

Scroll down to the 'main()' function and insert the highlighted code as shown below into the beginning of the user code area of the main() function:

void main(void)
{



```
R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */
    /* Initialize the debug LCD */
    R_LCD_Init();
    /* Displays the application name on the debug LCD */
    R_LCD_Display(0, (uint8_t *)" RSK+RX71M ");
    R_LCD_Display(1, (uint8_t *)" Tutorial ");
    R_LCD_Display(2, (uint8_t *)" Tutorial ");
    while (lU)
    {
        ;
        }
        /* End user code. Do not edit comment generated here */
}
```

5.1.1 SPI Code

The Okaya LCD display is driven by the SPI Master that was configured using Code Generator in §4.3.5. In the e² studio Project Tree, open the file 'r_cg_sci.h' by double-clicking on it. Insert the following code in the user code area at the end of the file:

```
/* Start user code for function. Do not edit comment generated here */
MD_STATUS R_SCI6_SPIMasterTransmit(uint8_t * const tx_buf, const uint16_t tx_num);
/* End user code. Do not edit comment generated here */
```

Now, open the r_cg_sci_user.c file and insert the following code in the user area for global:

```
/* Start user code for global. Do not edit comment generated here */
/* Flag used locally to detect transmission complete */
static volatile uint8_t sci6_txdone;
/* End user code. Do not edit comment generated here */
```

Insert the following code in the transmittend call-back function for SCI6:

```
static void r_sci6_callback_transmitend(void)
{
    /* Start user code. Do not edit comment generated here */
    sci6_txdone = TRUE;
    /* End user code. Do not edit comment generated here */
}
```

Now insert the following function in the user code area at the end of the file:

```
/* Start user code for adding. Do not edit comment generated here */
/*
                                            ******
                                                   *******
* Function Name: R_SCI6_SPIMasterTransmit
 Description % \left( {{\mathcal{T}}_{{\rm{B}}}} \right) : This function sends SPI6 data to slave device.
              : tx_buf -
 Arguments
                    transfer buffer pointer
                tx_num
                    buffer size
* Return Value : status
                    MD_OK or MD_ARGERROR
*****
                                             MD_STATUS R_SCI6_SPIMasterTransmit (uint8_t * const tx_buf, const uint16_t tx_num)
ł
   MD STATUS status = MD OK;
    /* clear the flag before initiating a new transmission ^{\ast/}
   sci6_txdone = FALSE;
    /* Send the data using the API */
   status = R_SCI6_SPI_Master_Send(tx_buf, tx_num);
    /* Wait for the transmit end flag */
   while (FALSE == sci6_txdone)
    {
        /* Wait */
   }
   return (status);
```



This function uses the transmit end callback function to perform flow control on the SPI transmission to the LCD, and is used as the main API call in the LCD code module.

5.1.2 CMT Code

The LCD code needs to insert delays to meet the timing requirements of the display module. This is achieved using the dedicated timer which was configured using Code Generator in §4.3.3. Open the file r_cg_cmt.h and insert the following code in the user area for function at the end of the file:

```
/* Start user code for function. Do not edit comment generated here */
void R_CMT_MsDelay(const uint16_t millisec);
/* End user code. Do not edit comment generated here */
```

Open the file r_cg_cmt_user.c and insert the following code in the user area for global at the beginning of the file:

```
/* Start user code for global. Do not edit comment generated here */
static volatile uint8_t one_ms_delay_complete = FALSE;
/* End user code. Do not edit comment generated here */
```

Scroll down to the r_cmt_cmi0_interrupt() function and insert the following line in the user code area:

```
static void r_cmt_cmi0_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    one_ms_delay_complete = TRUE;
    /* End user code. Do not edit comment generated here */
}
```

Then insert the following function in the user code area at the end of the file:

```
/* Start user code for adding. Do not edit comment generated here */
* Function Name: R_CMT_MsDelay
* Description : Uses CMT0 to wait for a specified number of milliseconds
         : uint16_t millisecs, number of milliseconds to wait
* Arguments
* Return Value : None
void R_CMT_MsDelay (const uint16_t millisec)
{
  uint16_t ms_count = 0;
  do
  {
     R_CMT0_Start();
     while (FALSE == one_ms_delay_complete)
     {
       /* Wait */
     }
     R_CMT0_Stop();
     one_ms_delay_complete = FALSE;
     ms count++;
  } while (ms_count < millisec);</pre>
}
End of function R_CMT_MsDelay
```



5.2 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 ed 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.

e ² Properties for CG_Tutorial				×
type filter text	Settings		⇐ ◄ ⇔	•
 Resource Builders C/C++ Build Build Variables Change Toolchain Vers Dependency Scan Device Environment Logging Settings Tool Chain Editor C/C++ General Project References Run/Debug Settings Task Repository 	 Compiler Source Object List Optimize Miscellaneous User CPU PIC/PID Assembler Source Object List Miscellaneous User Source Object List Miscellaneous User List List List List List Dobject List List Dobject User List Dobject User Dobject Dobje	Include file directories "\${TCINSTALL}/include"	 ● ● ● ● ● 	
		Preinclude files	 € €	E
	Optimize	Defines	🛃 📾 🗟 🖓 🖓	

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'.e² studio formats the path as show in Figure 5-3 below.

e ² Add directory path	×
Directory:	
\${workspace_loc:/\${ProjName}/src}	
OK Cancel Workspace File system.	

Figure 5-3 Adding workspace search path

Repeat the above steps to add the 'src/cg_src' workspace search path. Select 'Build Project' from the 'Project' menu, or use the button.e² studio will build the project with no errors.

The project may now be run using the debugger as described in §6. The program will display 'RSK+RX71M Tutorial Press Any Switch' on 3 lines in the LCD display.

5.3 Switch Code Integration

API functions for user switch control are provided with the RSK. Locate the files rskrx71mdef.h, r_rsk_switch.h and r_rsk_switch.c on the RSK DVD. These files can be found in the Tutorial project for e² studio. Copy these files into the C:\Workspace\CG_Tutorial\src directory. Import these three files into the project in the same way as the lcd files.

The switch code uses interrupt code in the files $r_cg_icu.h$, $r_cg_icu.c$ and $r_cg_icu_user.c$ and timer code in the files $r_cg_cmt.h$, $r_cg_cmt.c$ and $r_cg_cmt_user.c$, as described in §4.3.2 and §4.3.3. It is necessary to provide additional user code in these files to implement the switch press/release detection and de-bouncing required by the API functions in $r_rsk_switch.c$.

5.3.1 Interrupt Code

In the e² studio Project Tree, expand the 'src/cg_src' folder and open the file 'r_cg_icu.h' by double-clicking on it. Insert the following code in the user code area at the end of the file:

/* Start user code for function. Do not edit comment generated here */
/* Function prototypes for detecting and setting the edge trigger of ICU_IRQ */
uint8_t R_ICU_IRQIsFallingEdge(const uint8_t irq_no);
void R_ICU_IRQSetFallingEdge(const uint8_t irq_no, const uint8_t set_f_edge);
void R_ICU_IRQSetRisingEdge(const uint8_t irq_no, const uint8_t set_r_edge);
/* End user code. Do not edit comment generated here */

Now, open the r_cg_icu.c file and insert the following code in the user code area at the end of the file:

```
/* Start user code for adding. Do not edit comment generated here */
/*
                                   ******
* Function Name: R_ICU_IRQIsFallingEdge
 Description : This function returns 1 if the specified ICU_IRQ is set to
             falling edge triggered, otherwise 0.
* Arguments
          : uint8_t irq_no
 Return Value : 1 if falling edge triggered, 0 if not
                                           * *
uint8 t R ICU IROIsFallingEdge (const uint8 t irg no)
   uint8_t falling_edge_trig = 0x0;
   if (ICU.IRQCR[irq_no].BYTE & _04_ICU_IRQ_EDGE_FALLING)
   {
      falling_edge_trig = 1;
   }
   return falling edge trig;
}
End of function R_ICU_IRQIsFallingEdge
                               * Function Name: R_ICU_IRQSetFallingEdge
 Description : This function sets/clears the falling edge trigger for the
             specified ICU IRO.
* Arguments
          : uint8_t irq_no
             uint8_t set_f_edge, 1 if setting falling edge triggered, 0 if
             clearing
 Return Value : None
                   *****
void R_ICU_IRQSetFallingEdge (const uint8_t irq_no, const uint8_t set_f_edge)
{
   if (1 == set_f_edge)
   {
      ICU.IRQCR[irq_no].BYTE = _04_ICU_IRQ_EDGE_FALLING;
   }
   else
   {
      ICU.IRQCR[irq_no].BYTE &= (uint8_t) ~_04_ICU_IRQ_EDGE_FALLING;
   }
}
```

```
* End of function R_ICU_IRQSetFallingEdge
               ************
* Function Name: R_ICU_IRQSetRisingEdge
* Description : This function sets/clear the rising edge trigger for the
          specified ICU_IRQ.
        : uint8_t irq_no
* Arguments
          uint8_t set_r_edge, 1 if setting rising edge triggered, 0 if
          clearing
* Return Value : None
*****
              void R_ICU_IRQSetRisingEdge (const uint8_t irq_no, const uint8_t set_r_edge)
{
  if (1 == set_r_edge)
  {
     ICU.IRQCR[irq_no].BYTE = _08_ICU_IRQ_EDGE_RISING;
  }
  else
  {
     ICU.IRQCR[irq_no].BYTE &= (uint8_t) ~_08_ICU_IRQ_EDGE_RISING;
  }
}
* End of function R_ICU_IRQSetRisingEdge
                         /* End user code. Do not edit comment generated here */
```

Open the r_cg_icu_user.c file and insert the following code in the user code area for include near the top of the file:

```
/* Start user code for include. Do not edit comment generated here */
/* Defines switch callback functions required by interrupt handlers */
#include "r_rsk_switch.h"
/* End user code. Do not edit comment generated here */
```

In the same file insert the following code in the user code area inside the function r_icu_irq2_interrupt ():

/* Start user code. Do not edit comment generated here */
/* Switch 2 callback handler */
R_SWITCH_IsrCallback2();
/* End user code. Do not edit comment generated here */

In the same file insert the following code in the user code area inside the function r_icu_irq5_interrupt ():

```
/* Start user code. Do not edit comment generated here */
/* Switch 1 callback handler */
R_SWITCH_IsrCallback1();
/* End user code. Do not edit comment generated here */
```

5.3.2 De-bounce Timer Code

Open the r_cg_cmt_user.c file and insert the following code in the user code area for include near the top of the file:

```
/* Start user code for include. Do not edit comment generated here */
/* Defines switch callback functions required by interrupt handlers */
#include "r_rsk_switch.h"
/* End user code. Do not edit comment generated here */
```

In the same file insert the following code in the user code area inside the function r_cmt_cmi1_interrupt ():

```
/* Start user code. Do not edit comment generated here */
/* Stop this timer - we start it again in the de-bounce routines */
R_CMT1_Stop();
/* Call the de-bounce call back routine */
R_SWITCH_DebounceIsrCallback();
/* End user code. Do not edit comment generated here */
```

In the same file insert the following code in the user code area inside the function r_cmt_cmi2_interrupt ():

```
/* Start user code. Do not edit comment generated here */
/* Stop this timer - we start it again in the de-bounce routines */
R_CMT2_Stop();
/* Call the de-bounce call back routine */
R_SWITCH_DebounceIsrCallback();
/* End user code. Do not edit comment generated here */
```

5.3.3 Main Switch and ADC Code

In this part of the tutorial we add the code to act on the switch presses to activate A/D conversions and display the result on the LCD. In §4.3.4 we configured the ADC to be triggered from the ADTRG0# pin. In this code, we also perform software triggered A/D conversion from the user switches SW1 and SW2, by reconfiguring the ADC trigger source on-the-fly once an SW1 or SW2 press is detected.

In the e² studio Project Tree open the file 'r_cg_userdefine.h'. Insert the following code the user code area, resulting in the code 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 */
```

Open the file 'r_cg_main.c' and insert #include "r_rsk_switch.h" in the user code area for include, resulting in the code shown below:

```
/* Start user code for include. Do not edit comment generated here */
#include "r_okaya_lcd.h"
#include "r_rsk_switch.h"
/* End user code. Do not edit comment generated here */
```

Next add the switch module initialization function call highlighted in the user code area inside the main() function, resulting in the code shown below:

```
void main(void)
{
   R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */
   /* Initialize the switch module */
 R_SWITCH_Init();
   /* Initialize the debug LCD */
   R_LCD_Init();
    /* Displays the application name on the debug LCD */
   R_LCD_Display(0, (uint8_t *)" RSK+RX71M ");
   R_LCD_Display(1, (uint8_t *)" Tutorial ");
   R_LCD_Display(2, (uint8_t *)" Press Any Switch ");
   while (1U)
    {
        ;
    }
    /* End user code. Do not edit comment generated here */
```

In the same file, insert the declarations in the user code area for global, resulting in the code shown below:

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

/* Prototype declaration for cb_switch_press */ static void cb_switch_press (void); /* 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);
/* Variable for flagging user requested ADC conversion */
volatile uint8_t g_adc_trigger = FALSE;
```

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

Next add the highlighted code below in the user code area inside the main() function and the code inside the while loop, resulting in the code shown below:

```
void main(void)
{
   R_MAIN_UserInit();
   /* Start user code. Do not edit comment generated here */
    /* Initialize the switch module */
   R_SWITCH_Init();
   /* Set the call back function when SW1 or SW2 is pressed */
  R_SWITCH_SetPressCallback(cb_switch_press);
    /* Initialize the debug LCD */
   R_LCD_Init ();
   /* Displays the application name on the debug LCD */
   R_LCD_Display(0, (uint8_t *)" RSK+RX71M ");
   R_LCD_Display(1, (uint8_t *)" Tutorial ");
   R_LCD_Display(2, (uint8_t *)" Press Any Switch ");
   /* Start the A/D converter */
 R_S12AD0_Start();
   while (1U)
    {
        uint16_t adc_result;
        /* Wait for user requested A/D conversion flag to be set (SW1 or SW2) */
        if (TRUE == g_adc_trigger)
        {
            /* Call the function to perform an A/D conversion */
           adc_result = get_adc();
            /* Display the result on the LCD */
           lcd_display_adc(adc_result);
            /* Reset the flag */
           g_adc_trigger = FALSE;
        }
          SW3 is directly wired into the ADTRGOn pin so will
          cause the interrupt to fire */
        else if (TRUE == g_adc_complete)
        {
            /* Get the result of the A/D conversion */
           R_S12AD0_Get_ValueResult(ADCHANNEL0, &adc_result);
            /* Display the result on the LCD */
            lcd_display_adc(adc_result);
            /* Reset the flag */
           g_adc_complete = FALSE;
        }
    }
    /* End user code. Do not edit comment generated here */
```

Then add the definition for the switch call-back, get_adc() and lcd_display_adc() functions in the user code area for adding at the end of the file, as shown below:



```
* Return value : none
     static void cb_switch_press (void)
{
   /* Check if switch 1 or 2 was pressed */
  if (g_switch_flag & (SWITCHPRESS_1 | SWITCHPRESS_2))
  {
      /* set the flag indicating a user requested A/D conversion is required */
     g_adc_trigger = TRUE;
     /* Clear flag */
     g_switch_flag = 0x0;
  }
}
* End of function cb_switch_press
             *****
* Function Name : get_adc
* Description : Reads the ADC result, converts it to a string and displays
            it on the LCD panel.
        : none
* Argument
* Return value : uint16_t <u>adc</u> value
          ++++
                          static uint16_t get_adc (void)
ł
  /* A variable to retrieve the adc result */
  uint16_t adc_result;
   /* Stop the A/D converter being triggered from the pin ADTRGOn */
  R_S12AD0_Stop();
  /* Start a conversion */
  R_S12AD0_SWTriggerStart();
  /* Wait for the A/D conversion to complete */
  while (FALSE == g_adc_complete)
  {
     /* Wait */
  }
   /* Stop conversion */
  R_S12AD0_SWTriggerStop();
  /* Clear ADC flag */
  g_adc_complete = FALSE;
  R_S12AD0_Get_ValueResult(ADCHANNEL0, &adc_result);
   /* Set AD conversion start trigger source back to ADTRGOn pin */
  R_S12AD0_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 a temporary variable */
  uint8_t a;
  /* Declare temporary character string */
  char lcd_buffer[11] = " ADC: XXXH";
   /* Convert ADC result into a character string, and store in the local.
    Casting to ensure use of correct
                                            data type. */
```

 $/\,{}^{\star}$ End user code. Do not edit comment generated here ${}^{\star}/$

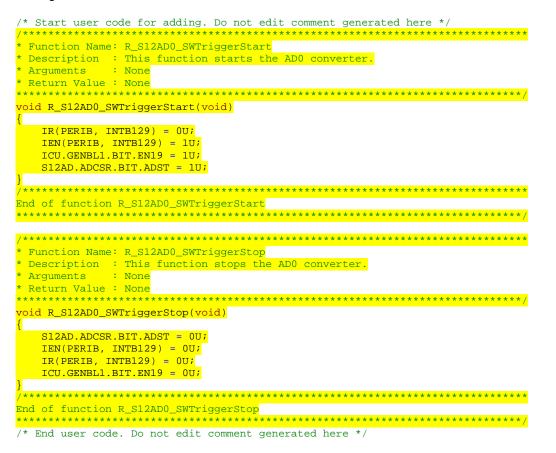
/* End user code. Do not edit comment generated here */

Open the file 'r_cg_s12ad.h' by double-clicking on it. Insert the following code in the in the user code area for function, resulting in the code 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_S12AD0_SWTriggerStart(void);
void R_S12AD0_SWTriggerStop(void);

Open the file 'r_cg_s12ad.c' by double-clicking on it. Insert the following code in the user code area for

adding at the end of the file, as shown below:



Open the file r_cg_s12ad_user.c and insert the following code in the in the user code area for global, resulting in the code shown below:

/* 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;

RENESAS

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

Insert the following code in the in the user code area of the r_s12ad0_interrupt () function, resulting in the code shown below:

```
static void r_s12ad0_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    g_adc_complete = TRUE;
    /* End user code. Do not edit comment generated here */
}
```

Select 'Build Project' from the 'Project' menu, or use the button. e² studio will build the project with no errors.

The project may now be run using the debugger as described in §6. When any switch is pressed, the program will perform an A/D conversion of the voltage level on the ADPOT line and display the result on the LCD panel. Return to this point in the Tutorial to add the UART user code.

5.4 Debug Code Integration

API functions for trace debugging via the RSK+ serial port are provided with the RSK+. Locate the files r_rsk_debug.h and r_rsk_debug.c on the RSK+ DVD. These files can be found in the RSKRX71M_Tutorial project for e² studio. Copy these files into the C:\Workspace\CG_Tutorial\src directory. Import these two files into the project in the same way as the LCD files.

In the r_rsk_debug.h file, ensure the following macro definition is included:

```
/* Macro for definition of serial debug transmit function - user edits this */
#define SERIAL_DEBUG_WRITE (R_SCI7_AsyncTransmit)
```

This macro is referenced in the r_rsk_debug.c file and allows easy re-direction of debug output if a different debug interface is used.

5.5 UART Code Integration

5.5.1 SCI Code

In the e² studio Project Tree, expand the 'src/cg_src' folder and open the file 'r_cg_sci.h' by double-clicking on it. Insert the following code in the user code area 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_SCI6_SPIMasterTransmit(uint8_t * const tx_buf, const uint16_t tx_num);
MD_STATUS R_SCI7_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 */

Open the file 'r_cg_sci_user.c. Insert the following code in the user area for global near the beginning of the file:

/* 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 sci6_txdone;



RSK+RX71M

static volatile uint8_t sci7_txdone;

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

In the same file, insert the following code in the user code area inside the r_sci7_callback_transmittend() function:

```
static void r_sci7_callback_transmitend(void)
{
    /* Start user code. Do not edit comment generated here */
    sci7_txdone = TRUE;
    /* End user code. Do not edit comment generated here */
}
```

In the same file, insert the following code in the user code area inside the r_sci7_callback_receiveend() function:

```
static void r_sci7_callback_receiveend(void)
{
    /* Start user code. Do not edit comment generated here */
    /* Check the contents of g_rx_char */
    if (('c' == g_rx_char) || ('C' == g_rx_char))
    {
        g_adc_trigger = TRUE;
    }
    /* Set up SCI7 receive buffer and callback function again */
    R_SCI7_Serial_Receive((uint8_t *)&g_rx_char, 1);
    /* End user code. Do not edit comment generated here */
}
```

At the end of the file, in the user code area for adding, add the following function definition:

```
* Function Name: R_SCI7_AsyncTransmit
 Description : This function sends SCI7 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_SCI7_AsyncTransmit (uint8_t * const tx_buf, const uint16_t tx_num)
ł
  MD_STATUS status = MD_OK;
  /* clear the flag before initiating a new transmission */
  sci7_txdone = FALSE;
  /* Send the data using the API */
  status = R_SCI7_Serial_Send(tx_buf, tx_num);
  /* Wait for the transmit end flag */
  while (FALSE == sci7_txdone)
  {
     /* Wait */
  }
  return (status);
}
* End of function R_SCI7_AsyncTransmit
                  ***************
/* End user code. Do not edit comment generated here */
```



5.5.2 Main UART code

Open the file 'r_cg_main.c'. Add the following declaration to the user code area for include near the top of the file:

#include "r_rsk_debug.h"

Add the following declaration to the user code area for global near the top of the file:

```
/* 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 A/D conversion count for user display */ static uint8_t adc_count = 0;
```

Add the following highlighted code to the user code area in the main function:

```
void main(void)
{
   R MAIN UserInit();
    /* Start user code. Do not edit comment generated here */
    /* Initialize the switch module */
   R_SWITCH_Init();
    /* Set the call back function when SW1 or SW2 is pressed */
   R_SWITCH_SetPressCallback(cb_switch_press);
    /* Initialize the debug LCD */
   R_LCD_Init ();
    /* Displays the application name on the debug LCD */
   R_LCD_Display(0, (uint8_t *)" RSK+RX71M ");
R_LCD_Display(1, (uint8_t *)" Tutorial ");
   R_LCD_Display(2, (uint8_t *)" Press Any Switch ");
    /* Start the A/D converter */
   R_S12AD0_Start();
    /* Set up SCI7 receive buffer and callback function */
   R_SCI7_Serial_Receive((uint8_t *)&g_rx_char, 1);
    /* Enable SCI7 operations */
   R_SCI7_Start();
    while (1U)
    {
        /* Wait for user requested A/D conversion flag to be set */
        if (TRUE == g_adc_trigger)
        {
            uint16_t adc_result;
            /* Call the function to perform an A/D conversion */
            adc_result = get_adc();
            /* Display the result on the LCD */
            lcd_display_adc(adc_result);
            /* Increment the adc_count */
            if (16 == ++adc_count)
            {
                adc_count = 0;
            }
            /* Send the result to the UART */
            uart_display_adc(adc_count, adc_result);
            /* Reset the flag */
            g_adc_trigger = FALSE;
         * SW3 is directly wired into the ADTRGOn pin so will
           cause the interrupt to fire */
        else if (TRUE == g_adc_complete)
```

RENESAS

```
/* Get the result of the A/D conversion */
          R_S12AD0_Get_ValueResult(ADCHANNEL0, &adc_result);
           /* Display the result on the LCD */
          lcd_display_adc(adc_result);
          /* Increment the adc_count */
          if (16 == ++adc_count)
          {
              adc_count = 0;
           /* Send the result to the UART */
          uart_display_adc(adc_count, adc_result);
          /* Reset the flag */
          g_adc_complete = FALSE;
       }
   /* End user code. Do not edit comment generated here */
}
Then, add the following function definition in the user code area at the end of the file:
 Function Name : uart_display_adc
 Description : Converts adc result to a string and sends it to the UART1.
             : uint8_t : adc_count
 Argument
               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 char 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));</pre>
   a = (char)((adc_result & 0x0F00) >> 8);
   uart_buffer[14] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));</pre>
   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));</pre>
   /* Send the string to the UART */
   R_DEBUG_Print(uart_buffer);
}
* End of function uart_display_adc
```

Select 'Build Project' from the 'Build' menu. e² studio will build the project with no errors.

The project may now be run using the debugger as described in §6. Connect the RSK+ G1CUSB0 port to a USB port on a PC. If this is the first time the RSK+ has been connected to the PC then a device driver will be installed automatically. Open Device Manager, the virtual COM port will now appear under 'Port (COM & LPT)' as 'RSK USB Serial Port (COMx)', where x is a number.

Open a terminal program, such as HyperTerminal, on the PC with the same settings as for SCI7 (see §4.3.5). When any switch is pressed, or when 'c' is sent via the COM port, the program will perform an A/D conversion of the voltage level on the ADPOT line and display the result on the LCD panel and send the result to the PC terminal program via the via SCI7. Return to this point in the Tutorial to add the LED user code.



5.6 LED Code Integration

Open the file 'r_cg_main.c'. Add the following declaration to the user code area for include near the top of the file:

```
#include "rskrx71mdef.h"
/* End user code. Do not edit comment generated here */
```

Add the following declaration to the user code area for global near the top of the file:

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

Add the following highlighted code to the user code area in the main function:

```
void main(void)
    R_MAIN_UserInit();
    /* Start user code. Do not edit comment generated here */
    /* Initialize the switch module */
   R_SWITCH_Init();
    /* Set the call back function when SW1 or SW2 is pressed */
   R_SWITCH_SetPressCallback(cb_switch_press);
    /* Initialize the debug LCD */
   R LCD Init ();
    /* Displays the application name on the debug LCD */
    R_LCD_Display(0, (uint8_t *)" RSK+RX71M ");
    R_LCD_Display(1, (uint8_t *)" Tutorial ");
   R_LCD_Display(2, (uint8_t *)" Press Any Switch ");
    /* Sart the A/D converter */
   R_S12AD0_Start();
    /* Set up SCI7 receive buffer and callback function */
   R_SCI7_Serial_Receive((uint8_t *)&g_rx_char, 1);
    /* Enable SCI7 operations */
    R_SCI7_Start();
    while (1U)
    ł
        uint16_t adc_result;
        /* Wait for user requested A/D conversion flag to be set(SW1 or SW2) */
        if (TRUE == g_adc_trigger)
        {
            /* Call the function to perform an A/D 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 the UART */
            uart_display_adc(adc_count, adc_result);
            /* Reset the flag */
            g_adc_trigger = FALSE;
        /* SW3 is directly wired into the ADTRGOn pin so will
           cause the interrupt to fire */
        else if (TRUE == g_adc_complete)
        {
            /* Get the result of the A/D
                                                              conversion */
```

}

R_S12AD0_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;
}
/* End user code. Do not edit comment generated here */

Then, add the following function definition in the user code area at the end of the file:

```
* 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 = (count & 0x01) ? LED_ON : LED_OFF;
  LED1 = (count & 0x02) ? LED_ON : LED_OFF;
  LED2 = (count \& 0x04) ? LED_ON : LED_OFF;
  LED3 = (count & 0x08) ? LED_ON : LED_OFF;
}
* End of function led_display_count
                         ******
* *
/* End user code. Do not edit comment generated here */
```

Select 'Build Project' from the 'Build' menu, or press F7. e² studio will build the project with no errors.

The project may now be run using the debugger as described in §6. The code will perform the same but now the LEDs will display the adc_count in binary form.



6. Debugging the Project

In the Project Explorer pane, ensure that the 'CG_Tutorial' project is selected. To enter the debug configurations click upon the arrow next to the debug button and select 'Debug Configuration'. In order to run the project there are two setting under 'Renesas GDB Hardware Debugging' -> 'Connection Settings' that need modifying.

Ensure that in debug configuration that the 'Power Target From The Emulator(MAX 200mA) is set to No , and the 'Extal Frequency' is set to the correct frequency, this can be found from the device schematics (in the case of RSK+RX71M the setting should be 24.00000).

For more information on powering the RSK+RX71M please refer to the Usermanual.

pe filter text	📄 Main 🧚 Debugger 🛛 🕨 Startup 🦆 Source	Common	
GDB Hardware Debugging GDB Simulator Debugging (GDB Simulator Chebugging (GHS Local C/C++ Launch Renesas GDB Hardware Atta	Debug hardware: E1 (RX) Target GDB Settings Connection Settings Debug Tool	Device: R5F571ML	
Renesas GDB Hardware Deb	A Clock		=
CG_Tutorial HardwareD	Main Clock Source	EXTAL	
CG_Tutorial Release	Extal Frequency[MHz]	24.0000	
💽 Renesas Simulator Debuggi	Permit Clock Source Change On Writing Ir	ntern Yes 👻 👻	
	Connection with Target Board		
	Emulator	(Auto)	
	Connection Type	JTag 👻	
	JTag Clock Frequency[MHz]	16.5 👻	-
	Fine Baud Rate[Mbps]	2.00 👻	
	Hot Plug	No	
	⊿ Power		
	Power Target From The Emulator (MAX 20		
	Supply Voltage	3.3V 👻	
	CPU Operating Mode		
	Register Setting	Single Chip 👻	
	Mode nin	Sinale-chin mode 🗾 👻	
ter matched 8 of 12 items		Apply Revert	
ter matched 8 of 12 items			

Figure 6-1 Debug Configurations

To debug the project, click the ⁵/₂ button. The dialog shown in Figure 6-2will be displayed.

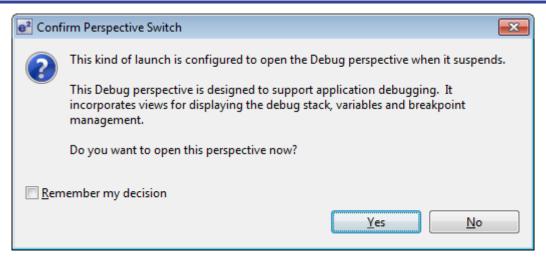


Figure 6-2 Perspective Switch Dialog

Click 'OK' to confirm that the debug window perspective will be used. The debugger will start up and the code will stop at the Code Generator function 'PowerOn_Reset_PC' as shown in Figure 6-3.

<pre>ile Edit Source Refactor Navigate Search Project Run Window Help</pre>
Debug S CG_Tutorial HardwareDebug [Renesas GDB Hardware Debugging] CG_Tutorial.x [1] Me CG_Tutorial.x [1] More CG_Tutorial.x
<pre> CG_Tutorial HardwareDebug [Renesas GDB Hardware Debugging] @ CG_Tutorial.x [1] @ Thread [1] 1 (single core) (Suspended : Signal : SIGINT:Interrupt) @ PowerON_Reset_PC() at r_cg_resetprg.c:66 0xffc00000 @ gdb @ GDB server . . GDB server . . GDB server . . Gob ffc00000 @ void PowerON_Reset_PC(void) {</pre>
<pre> Subtract State Sub</pre>
<pre>Name Name Name Name Name Name Name Name</pre>
<pre></pre>
<pre> gdb GDB server GDB server for r_cg_resetprg.c ☆ for ffc00000</pre>
<pre> r_cg_resetprg.c ☆ f66 ffc00000</pre>
<pre>66 ffc00000</pre>
<pre>66 ffc00000</pre>
<pre>66 ffc00000</pre>
<pre>66 ffc00000</pre>
<pre>66 ffc00000</pre>
<pre>67 { 68</pre>
<pre>69 ffc0000e set_extb(_sectop("EXCEPTVECT")); 70 #endif 71 ffc00017 set_intb(_sectop("C\$VECT"));</pre>
70 #endif /// /// ////////////////////////////
73 ⊖ #ifdefROZ /* Initialize FPSW */
74 #define _ROUND 0x00000001 /* Let FPSW RMbits=01 (round to zero) */ 75 ⊖ #else
76 #define _ROUND 0x00000000 /* Let FPSW RMbits=00 (round to nearest) */
77 #endif 78 ⊝#ifdef DOFF
📮 Console 🕱 🧔 Tasks 🔮 Renesas Coverage 🔋 Memory Usage 📀 Performance Analysis 🕚 Profile 👯 Real-time Chi
CG_Tutorial HardwareDebug [Renesas GDB Hardware Debugging] gdb
<pre>monitor set_io_access_width,RW,1,c1c84-c1c86,c1c94-c1c96,c1ca4-c1ca6,c1cb2,c1cb4,c1cb6,d0002,d000 monitor set io access width,RW,1,d0120-d0123,d0128-d012f,7fe010,7fe014,7fe018,7fe090,7fe0d0,7fe0d4</pre>

Figure 6-3 Debugger start up screen

To run the code click the button. The debugger will stop again at the beginning of the main() function. Press is again to run the code.

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.

Window	Help		
<u>c</u> - G	3	Welcome	
	0	Help Contents	
	% ?	Search	
		Dynamic Help	

For information about the RX71M group microcontroller refer to the RX71M Group Hardware Manual.

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

For more information about the Tutorial sample please refer to the Tutorial 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

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.

© 2015 Renesas Electronics Europe Limited. All rights reserved.

© 2015 Renesas Electronics Corporation. All rights reserved.

© 2015 Renesas System Design Co., Ltd. All rights reserved.



REVISION HISTORY

RSK+ RX71M Code Generator Tutorial Manual

Rev.	Date	Description	
		Page	Summary
1.00	Jan 23, 2015	—	First Edition issued

Renesas Starter
ManualKit+ Manual: Code Generator TutorialPublication Date:Rev. 1.00Jan 23, 2015Published by:Renesas Electronics Corporation



Renesas Electronics Corporation

SALES OFFICES

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. 281: +1-408-588-6000, Fax: +1-408-588-6100 2825 Yonge Street, H-408-588-6100 2825 Yonge Street, Stuite 8309 Richmond Hill, Ontario Canada L4C 9T3 Tei: +1-905-237-2004 Renesas Electronics Europe Limited Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K Tei: +44-1628-585-100, Fax: +44-1628-585-900 Renesas Electronics Europe GmbH Arcadiastrasse 10, 40472 Düsseldorf, Germany Tei: +49-211-6503-0, Fax: +44-9-211-6503-1727 Renesas Electronics (China) Co., Ltd. Room 1709, Quantum Plazz, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China Tei: +49-211-555, Fax: +68-10-8235-7679 Renesas Electronics (China) Co., Ltd. Room 1709, Quantum Plazz, No.27 ZhiChunLu Haidian District, Shanghai, P. R. China 200333 Tel: +96-12-825-1155, Fax: +68-210-82099 Renesas Electronics Nong Kong Limited Unit 1001-1611, 10/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong Tei: +82-226-5688, Fax: +862 2286-0022 Renesas Electronics Taiwan Co., Ltd. 80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949 Tei: +68-217-2000, Fax: +882 28175-9670 Renesas Electronics Malaysia Sch.Bhd. Unit 1207, Notes, R. Unit #06-02 Hyflux Innovation Centre, Singapore 339949 Tei: +68-217-2000, Fax: +60-27-955-9510 Renesas Electronics Malaysia Sch.Bhd. Unit 1207, Note, R. Menar Amoorp, Amoorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia Tei: +68-27-2000, Fax: +60-37-955-9510 Renesas Electronics Malaysia Sch.Bhd. Unit 1207, Note, R. Menar Amoorp, Amoorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia Tei: +68-27-2000, Fax: +80-67-20107 Renesas Electronics Malaysia Sch.Bhd. Unit 1207, Note, R. Menar Amoorp, Amoorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia Tei: +69-27-955-9300, Fax: +60-47-955-9510 Renesas Electro

http://www.renesas.com

RX71M Group

