

RX65N Group

Renesas Starter Kit+ Code Generator Tutorial Manual For e² studio

RENESAS 32-Bit MCU RX Family / RX600 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 Jun 2016

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 RX65N 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 RX65N 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+RX65N User's Manual	R20UT3558EG	
Tutorial Manual	Provides a guide to setting up RSK+ environment, running sample code and debugging programs.	RSK+RX65N Tutorial Manual	R20UT3562EG	
Quick Start Guide	Provides simple instructions to setup the RSK+ and run the first sample.	RSK+RX65N Quick Start Guide	R20UT3563EG	
Code Generator Tutorial	····· ···· ···· ···· ···· ···· ····		R20UT3564EG	
Schematics	Full detail circuit schematics of the RSK+.	RSK+RX65N Schematics	R20UT3557EG	
Hardware Manual	Provides technical details of the RX65N microcontroller.	RX65N Group, RX651 Group Hardware Manual	R01UH0590EJ	

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 / E2 Lite	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
PLL	Phase-locked Loop
Pmod™	This is a Digilent Pmod [™] Compatible connector. Pmod [™] is registered to <u>Digilent Inc.</u> Digilent-Pmod_Interface_Specification
PSU	Power Supply Unit
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

 Overview Purpose Features 	7
2. Introduction	8
3. Project Creation with e ² studio	0
3.1 Introduction	
3.2 Creating the Project	
4. Code Generation Using the e ² studio plug in	1/
4.1 Introduction	
4.2 Code Generator Tour	
4.3 Code Generation	
4.3.1 Clock Generator	
4.3.2 Interrupt Controller Unit	
4.3.3 Compare Match Timer	
4.3.4 12-bit A/D Converter	
4.3.5 Serial Communications Interface	
4.3.6 I/O Ports	
4.4 Building the Project	
5. User Code Integration	
5.1 LCD Code Integration	
5.1.1 SPI Code	
5.1.2 CMT Code	
5.2 Additional include paths	
5.3 Switch Code Integration	
5.3.1 Interrupt Code	
5.3.2 De-bounce Timer Code	
5.3.3 Main Switch and ADC Code	
5.4 Debug Code Integration	
5.5 UART Code Integration	
5.5.1 SCI Code	
5.5.2 Main UART code	
5.6 LED Code Integration	
6. Debugging the Project	51
7. Additional Information	53

RENESAS

RSK+RX65N

RENESAS STARTER KIT

1. Overview

1.1 Purpose

This RSK+ is an evaluation tool for Renesas microcontrollers. This manual describes how to use 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 (level two) and no outputs debugging information options selected, producing code suitable for release in a product.

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 E2 Lite emulator. Please refer to the relevant user manuals for more in-depth information.



×

3. Project Creation with e² studio

3.1 Introduction

In this section the user will be guided through the steps required to create a new C project for the RX65N 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.

In the Welcome page, click 'Go to

the e2 studio workbench'.



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





- Enter the project name 'CG_Tutorial'. In 'Project type:' choose 'Sample Project'. In 'Toolchains' choose 'Renesas RXC Toolchain'. Click 'Next'.
- e² C Project - -C Project - 0 Create C project of selected type Project name: CG_Tutorial Use default location Location: C:\workspace\CG_Tutorial Browse... Create Directory for Project Project type: Toolchains: a 🗁 Executable (Renesas) KPIT GNUARM-NONE-EABI Toolchain Sample Project KPIT GNURL78-ELF Toolchain 🔺 👝 Static Library (Renesas) KPIT GNURX-ELF Toolchain Sample Project KPIT GNUSH-ELF Toolchain Debug-Only Project Renesas CCRL Toolchain Executable Renesas GCC for RL78 Shared Library
 Static Library Renesas GCC for RX Renesas RXC Toolchain Others Renesas SHC Toolchain 👂 📂 Makefile project $\overline{\ensuremath{\mathbb V}}$ Show project types and toolchains only if they are supported on the platform ? < Back Next > Finish Cancel
- In the 'Target Specific Settings' dialog, select the options as shown in the screenshot opposite.
- The R5F565N9AxFB MCU is found under RX600 -> RX65N -> RX65N - 144 pin.
- Click 'Next'.





- In the 'Select Coding Assistant Tool' dialog, select 'Peripheral Code Generator or Firmware Integration Technology (FIT)' then ensure the 'Use Peripheral code Generator' is checked.
- Click 'Next'.



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

e ² C Project		
e2 studio - Project Generation Select Additional CPU Options		
Select Additional CPU Options: Round: Precision of Double: Sign of Char: Sign of bit Field: Allocate from Lower Bit Width of Divergence of Function: Specify Global Options: Denormalized number allow Replace from int with short Enum size is made the smal Pack structures, unions and Use try, throw and catch of Use dynamic cast and typei Saves and restores ACC usin	wed as a result llest l classes C++ id of C++	
?	< Back Next > Finish	Cancel



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

- - e² C Project e2 studio - Project Generation \diamond Global Options Settings Patch code generation None • Fast interrupt vector register: None • ROM: None • RAM: None • 00000000 Address (H'): Address Register: None • ? < Back Finish Next > Cancel
- In the 'Standard Header Files' dialog, select C99 for 'Library Configuration'. Untick 'new(EC++)' and leave all others at defaults.
- Click 'Next'.





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

.

² C Project	
2 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	
VO Register Definition Files	
Generate Hardware Setup Function None	_
Comparison of the sector	Cancel

• A summary dialog will appear, click 'OK' to complete the project generation.

e ² Project generator summary				
Project summary for CG_Tutorial				
 The following target de generated. 	vice settings and files will be			
PROJECT NAME :	CC Transiel	<u> </u>		
PROJECT DIRECTORY :	CG_Tutorial C:\workspace			
CPU SERIES : CPU TYPE :	RX600 RX65N			
TOOLCHAIN NAME : TOOLCHAIN VERSION :	Renesas_RXC v2.05.00			
	V2.03.00	E		
GENERATION FILES :				
Stack File \src\stacksct.h				
Custom Batch file \custom.bat				
Main Program \src\CG_Tutorial.c				
Setting of B and R sections	;	-		
?	ОК	Cancel		



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 RX65N. 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 an e² studio project called CG_Tutorial. The fully completed Tutorial project is contained on the RSK Web Installer (<u>https://www.renesas.com/rskrx65n/install</u>) and may be imported into e² studio by following the steps in the Quick Start Guide. This Tutorial is intended as a learning exercise for users who wish to use the Code Generator to generate their own custom projects for e² studio.

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 RSK Web Installer. 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.

You can download the latest document from: https://www.renesas.com/applilet

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 -> Perspective -> Open Perspective -> Other. In the 'Open Perspective' dialog shown in **Figure 4-1**, select 'Code Generator' and click 'OK'.



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**.

Code Generator - e2 studio				- 6 💌	
File Edit Navigate Search Project Renesas Views	Run Window Help				
	• • • • <i>• ∞</i> • 2 • 2 • • • • • •	> •	Quick Access 📔 🖬 C/C+	+ 🎄 Debug 🔛 Code Generator	
Project Explorer 🛛 🕞 🔄 🔽 🗖	🕎 Peripheral Functions 🙁 🛒 Code Preview 🔲	Properties 💯 FIT Configurator			
CG_Tutorial [HardwareDebug]	Clock setting Block diagram On-chip debug setting	Clock setting Block diagram On-chip debug setting			
Includes	- FIT setting				
⊳ 😕 src	Use clock configuration in "r. bsp. config.h"				
CG_Tutorial HardwareDebug.launch CG Tutorial Release.launch	Clock settings in this view will overwrite "r, bsp. config.h" on [Generate Code]				
custom.bat	Clock settings in this view will overwrite "r_bsp_c	contig.n on [Generate Code]		E	
Code Generator	- Main clock oscillator and RTCMCLK setting				
	Operation				
	Main clock oscillator forced oscillation (or	nly for RTC, software standby and dee	ep software standby mode)		
	Main clock oscillation source	Resonator	•		
	Frequency	24	(MHz)		
	Oscillator wait time	11000 (µs) (Act	tual value: 11090.909 µs)		
	Oscillation stop detection function	Disabled	Priority Level 15 (highest)		
	- Low speed clock oscillator (LOCO) setting				
	✓ Operation				
	Frequency	240	(kHz)		
	- High speed clock oscillator (HOCO) setting				
	Operation				
	Frequency	16	- (MHz)		
	- PLL circuit setting				
	Constantion			*	
	🔄 Console 🛿 🖹 Problems 🛛 🗎	31 🕅 🛃 🖶 🕶 🗂 🖛 🗖	📌 Conflicts View 🖂	~	
	Code Generator Console		0 items		
			Description	Resource Type	
	4		*		
)		E E E E E E E E E E E E E E E E E E E	<	•	
😂 CG_Tutorial					

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 SCI.

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 📲 Device Top View 🚆 Device List View				
Clock setting Block diagram On-chip debug setting				
-FIT setting				
Use clock configuration in "r_bsp_config.h"	Load			
Clock settings in this view will overwrite "r_bsp_co	nfig.h" on [Generate Code]			
-Main clock oscillator and RTCMCLK setting				
Operation				
Main clock oscillator forced oscillation (only Main clock oscillation source	r for RTC, software standby and deep s Resonator			
	24	(MHz)		
Frequency Oscillator wait time		(MHZ) I value: 11090.909 μs)		
	Disabled			
Oscillation stop detection function	Disabled			
-Low speed clock oscillato: (LOCO) setting Operation				
Frequency	240	(kHz)		
-High speed clock oscillator (HOCO) setting				
Operation				
Frequency	16	- (MHz)		
-PLL circuit setting				
V Operation				
PLL clock source	Main clock oscillator	•		
Input frequency division ratio	x1 👻			
Frequency multiplication factor	x 10.0 🗸			
Frequency	240 (MHz)			
-Sub-clock oscillator and RTC (RTCSCLK) setting				
Operation				
Sub-clock oscillator drive capacity	Drive capacity for low CL	· · · · · · · · · · · · · · · · · · ·		
Frequency	32.768	(kHz)		
Oscillator wait time	2252.73 (ms) (Actual v	value: 2296.182 ms)		
-RTC clock setting				
Operation Clock source	Sub-clock oscillator			
 -IWDT-dedicated low-speed clock oscillator (IWDTLOC Operation 	O) setting			
Frequency	120	(kHz)		
-System clock setting				
Clock source	PLL circuit	•		
System clock (ICLK)	x 1/2 👻 120	(MHz)		
Peripheral module clock (PCLKA)	x 1/2	(MHz)		
Peripheral module clock (PCLKB)	× 1/4 60	(MHz)		
Peripheral module clock (PCLKC)	x 1/4 - 60	(MHz)		
Peripheral module clock (PCLKD)	x 1/4 - 60	(MHz)		
External bus clock (BCLK)	x 1/2 - 120	(MHz)		
Flash IF clock (FCLK)	x 1/4 - 60	(MHz)		
USB clock (JCLK)	x 1/5 - 48	(MHz)		
ROM wait cycle	2 wait cycles 🗸	(1112)		
-BCLK pin output setting Enable BCLK forced output				
Clock output source				
	DOLIT			
-SDCLK pin output setting				

Figure 4-3 Clock setting tab



Click the arrow next to the Generate Report icon. Select 'Interrupt Controller Unit' as shown in **Figure 4-4** below. Proceed to the next section on the Interrupt Controller Unit.

<u>R</u> un <u>W</u> indow <u>H</u> elp			
/ 2 - 2 - 4 4 - + -			Quick Access
🚆 Peripheral Functions 🙁 🔙 Code Previe	w 🔲 Properties 💯 FIT Configurator		🔁 Generate Code 🛛 🔯 📼
Clock setting Block diagram On-chip debug	setting		📩 Clock Generator
- FIT setting			Voltage Detection Circuit
Use clock configuration in "r_bsp_config.h"	Load		Clock Frequency Accuracy Measurement Circuit
Clock settings in this view will overwrite "r_	bsp_config.h" on [Generate Code]		Low Power Consumption
			🔐 Interrupt Controller Unit
 Main clock oscillator and RTCMCLK setting — Operation 			🗱 Buses
Main clock oscillator forced oscillation	(only for RTC, software standby and deep software star	ndhu mode)	DMA Controller
		idby mode)	🛱 Data Transfer Controller
Main clock oscillation source	Resonator		Event Link Controller
Frequency	24	(MHz)	\$1〕 I/O Ports
Oscillator wait time	11000 (µs) (Actual value: 110)	90.909 μs)	Multi-Function Timer Pulse Unit 3
Oscillation stop detection function	Disabled	Priority Level 15 (highest) v	Ort Output Enable 3
			16-Bit Timer Pulse Unit
 Low speed clock oscillator (LOCO) setting — Operation 			Ch. Programmable Pulse Generator
	240 (kH	I=\	8-Bit Timer
Frequency	240 (КП	2	Ocmpare Match Timer
- High speed clock oscillator (HOCO) setting			Compare Match Timer W
Operation			Realtime Clock
Frequency	16 V (Mł	Hz)	🔏 Watchdog Timer
- PLL circuit setting			independent Watchdog Timer
 Operation 			Serial Communications Interface
PLL clock source	Main clock oscillator		₽ 🗱 I2C Bus Interface
Input frequency division ratio	x1 ¥		Interface
			Star CRC Calculator
Frequency multiplication factor	x 10.0 🗸		G. 12-Bit A/D Converter
Frequency	240 (MHz)		4 12-Bit D/A Converter
- Sub-clock oscillator and RTC (RTCSCLK) setti	ng		🤯 Data Operation Circuit
Operation			
Sub-clock oscillator drive capacity	Drive capacity for low CL		

Figure 4-4 Select Interrupt Controller Unit



4.3.2 Interrupt Controller Unit

Referring to the RSK+ schematic, SW1 is connected to IRQ8 (P00) and SW2 is connected to IRQ9 (P01). SW3 is connected 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 4-5** below.

💹 *Peripheral Functions 🛛	🥳 Code Preview 🚂	Device Top View 🦉 Dev	vice l	.ist View				🐻 Generate C
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)	Ŧ					
Software interrupt 2	Priority	Level 15 (highest)	-					
- NMI setting		(=	_					
NMI pin interrupt	Valid edge	Falling	-	Digital filter	No filter	-	0	(MHz)
- IRQ0 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	P32	-	Digital filter	No filter	Ŧ	0	(MHz)
	Valid edge	Low level	-	Priority	Level 15 (highest)	-		
- IRQ3 setting	~	[D22	_	Di la LEI	NL Ch	_		
IRQ3	Pin	P33	-	Digital filter	No filter	-	0	(MHz)
	Valid edge	Low level	-	Priority	Level 15 (highest)	~		
- IRQ4 setting			_	District films	N- Ch	_	0	(11)
IRQ4	Pin	PB1	-	Digital filter	No filter		0	(MHz)
	Valid edge	Low level	*	Priority	Level 15 (highest)	Ŧ		
- IRQ5 setting		[_	D 1 (1) (1)		_		
IRQ5	Pin	PA4	-	Digital filter	No filter	-	0	(MHz)
	Valid edge	Low level	Ŧ	Priority	Level 15 (highest)	Ŧ		
- IRQ6 setting			_					
IRQ6	Pin	PA3	-	Digital filter	No filter	-	0	(MHz)
	Valid edge	Low level		Priority	Level 15 (highest)	-		
- IRQ7 setting		[DE2	_	Di la Lon		_		
IRQ7	Pin	PE2	-	Digital filter	No filter	-	0	(MHz)
	Valid edge	Low level	Ŧ	Priority	Level 15 (highest)	Ŧ		
- IRQ8 setting		D 00	-	Di la LEI	NL Ch	-		
V IRQ8	Pin	P00	•	Digital filter	No filter	•	0	(MHz)
	Valid edge	Falling	•	Priority	Level 15 (highest)	•		
IRQ9 setting		D 01	_	District Eller	No Ches	_	0	
V IRQ9	Pin	P01	•	Digital filter	No filter	•	0	(MHz)
	Valid edge	Falling	•	Priority	Level 15 (highest)	•		
- IRQ10 setting IRQ10	Pi-	P42	-	Digital filter	No filter	-	0	(MU=)
E INGRIO	Pin		=	-			0	(MHz)
	Valid edge	Low level		Priority	Level 15 (highest)	-		
- IRQ11 setting	Di-	P42		Digital filter	No filter		0	(MUL=)
IRQ11		P43	-	Digital filter			0	(MHz)
	Valid edge	Low level	-	Priority	Level 15 (highest)	Ŧ		
-IRQ12 setting	~	Du	_	Di la LEI	NL Ch	_		
IRQ12	Pin	P44	-	Digital filter	No filter	-	0	(MHz)
19012	Valid edge	Low level	-	Priority	Level 15 (highest)	-		
- IRQ13 setting IRQ13	Pin	P45	-	Digital filter	No filter	-	0	(MHz)
			-	Priority	Level 15 (highest)			(
10014	Valid edge	Low level	4	r nonty	Lever 15 (highest)	T		
- IRQ14 setting	Pin	P46	-	Digital filter	No filter	~	0	(MHz)
			=			_		(1112)
	Valid edge	Low level	-	Priority	Level 15 (highest)	-		
- IRQ15 setting	D.	P47		Digital filter	No filter	-	0	(MU=)
	Pin		-				J	(MHz)
	Valid edge	Low level	-	Priority	Level 15 (highest)	Ŧ		

Figure 4-5 Interrupt Functions tab



Navigate to the 'Group Interrupts' sub tab and ensure that the 'Group BL0' interrupt is selected as shown in **Figure 4-6**. 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 🚆 Device Top View 📃 Device List View
General <u>Group Interrupts</u>	Interrupt B/A selection
- Group BE0 setting Group BE0	Priority Level 15 (highest)
Group BL0 setting	
Group BL0	Priority Level 15 (highest) -
- Group BL1 setting	
Group BL1	Priority Level 15 (highest)
-Group BL2 setting	
Group BL2	Priority Level 15 (highest)
- Group AL0 setting	
Group AL0	Priority Level 15 (highest)
-Group AL1 setting	
Group AL1	Priority Level 15 (highest)

Figure 4-6 Group Interrupt Functions tab

Click the arrow next to the Generate Report icon. Select 'Compare Match Timer' as shown in **Figure 4-7** below. Proceed to the next section on the Compare Match Timer.

:/C+	+ 🖳 Code Generator 🔅 Debug Quick Access					
	🕃 Generate Code 🔯 🔻 🗖 🗍 🍟					
÷.	Clock Generator					
	Voltage Detection Circuit					
8	Clock Frequency Accuracy Measurement Circuit					
Ċ.	Low Power Consumption					
W.	Interrupt Controller Unit					
	Buses					
, and a second	DMA Controller					
ű,	Data Transfer Controller					
1	Event Link Controller					
\$00	I/O Ports					
\odot	Multi-Function Timer Pulse Unit 3					
Q.	Port Output Enable 3					
3	16-Bit Timer Pulse Unit					
Ø.	Programmable Pulse Generator					
٢	8-Bit Timer					
٩	Compare Match Timer					
٢	Compare Match Timer W					
	Realtime Clock					
æ	Watchdog Timer					
æ	Independent Watchdog Timer					
Ţ	Serial Communications Interface					
₽ <mark>₩</mark>	I2C Bus Interface					
7	Serial Peripheral Interface					
1 <mark>2</mark>	CRC Calculator					
<u>.</u>	12-Bit A/D Converter					
dh,	12-Bit D/A Converter					
÷	Data Operation Circuit					

Figure 4-7 Select Compare Match Timer



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 configures CMT0 as shown in **Figure 4-8**. 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 🖇	🕎 Device Top View 💈	📱 Device List View	
CMT0 CMT1 CMT2 C	MT3			
 Compare match timer opera O Unused 	ation setting	Used		
- Count clock setting PCLK/8	PCLK/32	© PCLK/128	O PCLK/512	
 Interval value setting Interval value 		1	ms 🔻	(Actual value: 1)
-Interrupt setting Priority	tch interrupt (CMI0)	Level 10	•	

Figure 4-8 CMT0 tab

Navigate to the 'CMT1' sub-tab and configure CMT1 as shown in **Figure 4-9**. This timer is configured to generate a high priority interrupt after 20ms. This timer is used as hour short switch de-bounce timer later in this tutorial.

💯 *Peripheral Functions 🙁 🧕 Code Preview	🕎 Device Top View	🕎 Device List View							
CMT0 CMT1 CMT2 CMT3									
- Compare match timer operation setting									
⊘ Unused	Used								
- Count clock setting									
PCLK/8 PCLK/32	PCLK/128	PCLK/512							
- Interval value setting									
Interval value	20	ms (Actual value: 20) 							
- Interrupt setting									
Enable compare match interrupt (CMI1)									
Priority	Level 10	-							

Figure 4-9 CMT1 tab



Navigate to the 'CMT2' sub-tab and configure CMT2 as shown in **Figure 4-10**. 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.

ĺ	🕎 *Peripheral Functions	s 🐹 🛃 Code Preview	🕎 Device Top View	Device List View
	CMT0 CMT1 CMT2	CMT3		
	- Compare match timer op	eration setting		
	O Unused	-	Osed	
	- Count clock setting			
	PCLK/8	PCLK/32	PCLK/128	@ PCLK/512
	-Interval value setting —			
	Interval value		200	ms
	- Interrupt setting			
	Enable compare n	match interrupt (CMI2)		
		,	1	
	Priority		Level 10	-

Figure 4-10 CMT2 tab

Click the arrow next to the Generate Report icon. Select '12-Bit A/D Converter' as shown in **Figure 4-11** below. Proceed to the next section on the 12-Bit A/D Converter.

C/C+	+ 🖳 Code Generator 🕸 Debug Quick Access
	🐻 Generate Code 🗕 🔽 P 🗉 🏢
đ.	Clock Generator
	Voltage Detection Circuit
8	Clock Frequency Accuracy Measurement Circuit
Ċ.	Low Power Consumption
۲	Interrupt Controller Unit
	Buses
	DMA Controller
and a	Data Transfer Controller
12	Event Link Controller
\$@	I/O Ports
\odot	Multi-Function Timer Pulse Unit 3
Q.	Port Output Enable 3
\odot	16-Bit Timer Pulse Unit
Ø.	Programmable Pulse Generator
٢	8-Bit Timer
٨	Compare Match Timer
٢	Compare Match Timer W
	Realtime Clock
2	Watchdog Timer
8	Independent Watchdog Timer
Ţ	Serial Communications Interface
8 <mark>-8</mark>	I2C Bus Interface
Ţ	Serial Peripheral Interface
12	CRC Calculator
<u>4</u> 1.	12-Bit A/D Converter
ц <mark>к</mark>	12-Bit D/A Converter
÷-	Data Operation Circuit

Figure 4-11 Select 12-Bit A/D Converter



4.3.4 12-Bit A/D Converter

Navigate to the '12-Bit A/D Converter' node in Code Generator. In the 'S12AD0' sub-tab configures S12AD0 as shown in **Figure 4-12**, **Figure 4-13** and configure the S12AD0 as shown. We will be using the S12AD0 in Single scan 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.

💯 *Peripheral Functions 🙁 🛒 Code Pro	eview 📃 Device Top View 💈	Device List View	V	🐻 G
S12AD0 S12AD1				
General setting Window A setting Windo	ow B setting			
-S12AD0 operation setting				
Unused	Osed			
- Operation mode setting				
Single scan mode	🔘 Group scan mode	•	Continuo	us scan mode
- Group scan select				
Two groups (A,B)	Three groups (A,I)	3,C)		
- Double trigger mode setting				
Oisable	Enable			
- Self diagnosis setting				
Mode	Unused	•		
Voltage used	Use VREFH0 x 0	-		
- Disconnection detection assist setting				
Charge setting	Unused	•		
Period	2 ADCLK	-		
- Group scan priority setting				
Group priority	Group A without price	ority		-
Group action	Not restarted or con	tinued due to Grou	p A priority	•
Restart channel selection	Restarted from the f	irst scan channel		-
– A/D converted value count setting —				
 Addition mode 	Average mode			
-Window function setting				
Oisable	Enable			
-Window A operation setting				
Oisable	Enable			
- Window B operation setting				
 Disable 	Enable			
- Window A/B complex condition setting				
Window A/B complex condition	Window A comparis	on condition match	ed OR Window B compa	rison condition matched 📃 👻
- Analog input channel setting				
Convert	Convert	Convert	Add/Average	Dedicated sample
(Group A)		(Group C)	AD value	and hold
AN000				
AN001				
AN002 AN003				
AN003				
AN005				
AN006				
AN007				
	Figure 4-12 S ²	12AD0 tab	(1)	



- Conversion start trigger setting			
Conversion start trigger (Group A)			
A/D conversion start trigger pin			*
Conversion start trigger (Group B)			
Compare match/input capture from MTU0.TG	RA		· · · · · · · · · · · · · · · · · · ·
Conversion start trigger (Group C)			
Compare match/input capture from MTU1.TG	RA		· · · · · · · · · · · · · · · · · · ·
ADTRG0# pin selection	P07 -	•	
- Data registers setting			
AD converted value addition count	1-time conversion		v
Data placement	Right-alignment -	•	
Automatic clearing	Disable automatic clearing		•
Conversion resolution	12-bit resolution	•	
- Dedicated sample and hold circuit setting			
Input sampling time	8	(µs)	(Actual value: 0)
-AN000 / Self-diagnosis conversion time setting -			
Input sampling time	3.667	(µs)	(Actual value: 3.667)
-AN001 conversion time setting			
Input sampling time	3.667	(µs)	(Actual value: 3.667)
-AN002 conversion time setting			
Input sampling time	3.667	(µs)	(Actual value: 3.667)
-AN003 conversion time setting			
Input sampling time	3.667	(µs)	(Actual value: 3.667)
-AN004 conversion time setting			
Input sampling time	3.667	(µs)	(Actual value: 3.667)
-AN005 conversion time setting			
Input sampling time	3.667	(µs)	(Actual value: 3.667)
-AN006 conversion time setting			
Input sampling time	3.667	(µs)	(Actual value: 3.667)
-AN007 conversion time setting			
Input sampling time	3.667	(µs)	(Actual value: 3.667)
- Conversion time setting			
Total conversion time (Group A)	4.05	(µs)	
Total conversion time (Group B)		(µs)	(Note: Continuous sampling is disabled)
Total conversion time (Group C)		(µs)	
- Interrupt setting			
Enable AD conversion end interrupt (S12A)	DI)		
Priority	Level 15 (highest)	•	
☑ Enable AD conversion end interrupt for gro	up B (S12GBADI)		
Priority	Level 15 (highest)	-	
✓ Enable AD conversion end interrupt for gro	up C (S12GCADI)		
Priority	Level 15 (highest)	-	
	E: 4.40.040		4-1- (Q)

Figure 4-13 S12AD0 tab (2)



Click the arrow next to the Generate Report icon. Select 'Serial Communications Interface' as shown in **Figure 4-14** below. Proceed to the next section on the Serial Communications Interface.



Figure 4-14 Select Serial Communications Interface



4.3.5 Serial Communications Interface

Navigate to the 'Serial Communications Interface' tab in Code Generator, select the SCI7 sub-tab and apply the settings shown in **Figure 4-15**. In the RSK+RX65N SCI7 is used as an SPI master for the Pmod LCD on the PMOD1 connector as shown in the schematic.

🕎 *Peripheral Fur	nctions 🛛	🧾 Cod	le Previ	ew 归	Device	Top Vie	ew 🔛	Device L	ist View	
SCI0 SCI1 S	CI2 SCI3	SCI4	SCI5	SCI6	<u>SCI7</u>	SCI8	SCI9	SCI10	SCI11	SCI12
General setting	Setting									
- Function setting										
O Unused										
Asynchro	Asynchronous mode									
Asynchro	nous mode (Multi-pro	cessor)		Transn	nission			-	
Clock syn	nchronous m	ode			Transmission -					
Smart car	rd interface m	node			Transmission 💌					
Simple II	C bus									
Simple Simple Si	PI bus				Master	transm	it only		•	
- Pin setting										
TXD7	P90		T		RXD7	P9	2		-	
SSDA7	P90		Ŧ		SSCL7	P9	2		-	
SMOSI7	P90		•		SMISO7	P9	2		Ŧ	

Figure 4-15 SCI7 General Setting tab

Select the SCI7 'Setting' sub-tab and configure the SPI Master as illustrated in **Figure 4-16**. Make sure the 'Transfer direction setting' is set to 'MSB-first' and the 'Bit rate' is set to 15000000. All other settings remain at their defaults.

💯 *Peripheral Functions 🛛 😹 Code Preview 🧏	Device Top View 📲 Device List View
SCI0 SCI1 SCI2 SCI3 SCI4 SCI5 SCI6	SCI7 SCI8 SCI9 SCI10 SCI11 SCI12
General setting Setting	
- Transfer direction setting	
CLSB-first	MSB-first
- Data inversion setting	
Normal	Inverted
- Transfer rate setting	
Transfer clock	Internal clock P91
Bit rate	15000000 • (bps) (Actual value: 15000000, Error : 0%)
Enable modulation duty correction	
SCK7 pin function	Clock output
Charlesewise	
- Clock setting Clock delay	Clock is not delayed
Enable clock polarity inversion	
- Data handling setting	
Transmit data handling	Data handled in interrupt service routine
- Interrupt setting	
TXI7 priority	Level 15 (highest)
TEI7 priority (Group BL0)	(Please set priority setting in ICU)
	(······ ·····························
- Callback function setting	
Transmission end	

Figure 4-16 SCI7 SPI Master Setting

Staying in the 'Serial Communications Interface' tab in Code Generator, select the SCI2 sub-tab and apply the settings shown in **Figure 4-17**. In the RSK+RX65N SCI2 is connected via a Renesas RL78/G1C to provide a USB virtual COM port as shown in the schematic.

💯 *Peripheral Functions 🛛 🛃 Code Preview 🚆	Device Top View 🚆 Device List View				
SCI0 SCI1 SCI2 SCI3 SCI4 SCI5 SCI6	SCI7 SCI8 SCI9 SCI10 SCI11 SCI12				
General setting Setting					
- Function setting					
O Unused					
Asynchronous mode	Transmission/reception -				
Asynchronous mode (Multi-processor)	Transmission				
Clock synchronous mode	Transmission 👻				
Smart card interface mode	Transmission -				
⊘ Simple IIC bus					
Simple SPI bus	Slave transmit/receive 💌				
- Pin setting					
TXD2 P50 🗸	RXD2 P52 -				
SSDA2 P13 👻	SSCL2 P12 -				
SMOSI2 P13 👻	SMISO2 P12 -				

Figure 4-17 SCI2 General Setting tab

Select the SCI2 'Setting' sub-tab and configure SCI2 as illustrated in **Figure 4-18**. Make sure the 'Start bit edge detection' is set as 'Falling edge on RXD2 pin' and the 'Bit rate' is set to 19200 bps. All other settings remain at their defaults.

👮 *Peripheral Functions 🙁 🛒 Code Preview 🙎	📱 Device Top View 🛛 🖳 Device List V	iew
SCI0 SCI1 SCI2 SCI3 SCI4 SCI5 SCI6	SCI7 SCI8 SCI9 SCI10 SC	111 SCI12
General setting Setting		
-Start bit edge detection setting		
Low level on RXD2 pin	Falling edge on RXD2 pin	
- Data length setting		
◎ 9 bits	8 bits	⑦ 7 bits
- Parity setting		
None	⊘ Even	© Odd
Stop bit length setting 1 bit	2 bits	
- Transfer direction setting		
SB-first	MSB-first	
-Transfer rate setting		
Transfer clock	Internal clock -	
Base clock	16 cycles for 1-bit period 👻	
Bit rate	19200 -	(bps) (Actual value: 19230.769, Error: 0.16%)
Enable modulation duty correction		
SCK2 pin function	SCK2 is not used -	P51 -
-Noise filter setting		
Enable noise filter		
Noise filter clock	Clock signal divided by 1	60000000 (Hz)
- Hardware flow control setting		
None		_
© CTS2#	P54	
© RTS2#	P54 -]
- Data handling setting		
Transmit data handling	Data handled in interrupt service ro	utine 👻
Receive data handling	Data handled in interrupt service ro	utine 👻
- Interrupt setting		
TXI2 priority	Level 15 (highest)	
RXI2 priority	Level 15 (highest) -	
✓ Enable error interrupt (ERI2)		
TEI2, ERI2 priority (Group BL0)		(Please set priority setting in ICU)
- Callback function setting		
☑ Transmission end	Reception end	Reception error

Figure 4-18 SCI2 Asynchronous Setting



Click the arrow next to the Generate Report icon. Select 'I/O Ports' as shown in **Figure 4-19** below. Proceed to the next section on the I/O Ports.



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 P73 and LED3 is connected to PJ5. Navigate to the 'I/O Ports' tab in Code Generator and configure these four I/O lines as shown in **Figure 4-20**, **Figure 4-21** and **Figure 4-22** below. Ensure that the 'Output 1' tick box is checked. This ensures that the code is generated to set LEDs initially off.

🕎 *Peripher	al Functio	ns 🛛 🛃 (Code Preview	/ 👮 Device Top Vie	w 📓 Device I	List View		🐻 Gen
Port0 Port	1 Port2	Port3 Port	4 Port5 P	Port6 Port7 Port8	Port9 PortA	PortB PortC	PortD PortE PortF PortJ	
- P00	🔘 ln 👎	🔘 Out 😲	Pull-up	CMOS output	-	Output 1	Normal drive output	▼
	🔘 ln 👎	🔘 Out 😲	Pull-up	CMOS output	T	Output 1	Normal drive output	*
Our Unused - P03	⊚ In	⊚ Out	Pull-up	CMOS output	•	Output 1	Normal drive output	*
O Unused	⊚ In	Out	Pull-up	CMOS output	•	Vutput 1		
O Unused	⊚ In	Out	Pull-up	CMOS output	•	Output 1		
Onused	🔘 ln 🥊	🔘 Out 😲	Pull-up	CMOS output	*	Output 1		

Figure 4-20 I/O ports – Port0



🕎 *Peri	iphera	al Functio	ons 🖾 🖁	🧾 Co	de Previ	ew 💯	Device	Top Vi	ew 🔛	Device	List Viev	v					🐻 Gener
	Port1	Port2	Port3	Port4	Port5	Port6	Port7	Port8	Port9	PortA	PortB	PortC	PortD	PortE	PortF	PortJ	
– P70 —	sed	⊚ In	⊚ Out	[_ Pull-u	P CN	10S outp	out		Ŧ)utput 1	Norm	al drive	output		▼
 Unu P72 — 	sed	© In	⊚ Out		Pull-u	P CN	10S outp	out		Ŧ)utput 1					
O Unu - P73	sed	⊚ In	⊚ Out	[_ Pull-u	P CN	10S outp	out		Ŧ)utput 1					
© Unu - P74	sed	⊚ In	Out	[Pull-u	P CI	10S outp	out		•)utput 1	High	drive ou	itput		•
◉ Unu - P75 —	sed	⊚ In	⊚ Out		Pull-u	P CN	10S outp	out		Ŧ)utput 1	Norm	al drive	output		-
Onu - P76 —	sed	© In	⊚ Out	[Pull-u	P CN	10S outp	out		Ŧ)utput 1	Norm	al drive	output		Ŧ
◉ Unu - P77	sed	⊚ In	⊚ Out		Pull-u	P CN	10S outp	out		Y)utput 1	Norm	al drive	output		
Onu	sed	⊚ In	⊚ Out	[Pull-u	PC	10S outp	out		Ŧ)utput 1	Norm	al drive	output		-

Figure 4-21 I/O ports – Port7

💯 *Peripheral Functions 🙁 📓 Code Preview 📃 Device Top View 📃 Device List View															
Port0 Port	1 Port2	Port3	Port4	Port5	Port6	Port7	Port8	Port9	PortA	PortB	PortC	PortD	PortE	PortF	PortJ
- PJ3															
Onused	🔘 In	🔘 Ou	ıt [Pull-u	P CN	IOS outp	ut		-)utput 1				
- PJ5															
O Unused	🔘 In	Ou	it [Pull-u	P CN	IOS outp	ut		-	V)utput 1				

Figure 4-22 I/O ports - PortJ

P02 is used as one of the LCD control lines, together with P55, P56 and P93. Configure these lines as shown in **Figure 4-23**, **Figure 4-24** and **Figure 4-25**.

归 *Peripher	al Functio	ns 🖾	🧾 Co	de Previ	ew 归	Device	: Top Vi	ew 归	Device	List Viev	v					🐻 Gene
Port0 Port	1 Port2	Port3	Port4	Port5	Port6	Port7	Port8	Port9	PortA	PortB	PortC	PortD	PortE	PortF	PortJ	
- P00	🔘 ln 🤫) (Out	•	Pull-u	P CM	OS out	put		Ŧ)utput 1	Norm	nal drive	output		v
O Unused P02	🔊 In 🥊) (Out	•	Pull-u	PCM	OS out	put		Y)utput 1	Norm	nal drive	output		▼
O Unused - P03	© In	 Out 		Pull-u	P CM	OS out	put		•)utput 1	Norm	nal drive	output		•
O Unused	⊚ In	Out		Pull-u	P CM	OS out	put		•	V ()utput 1					
O Unused	⊚ In	Out		Pull-u	P CM	OS out	put		•	V)utput 1					
O Unused	🔘 In 🥊	Out 🔘	•	Pull-u	PCM	OS out	put		¥)utput 1					

Figure 4-23 I/O ports – Port0



4. Code Generation Using the e² studio plug in

归 *Perip	heral	Functio	ons 🖂	2	Code Previ	ew ይ	Device	e Top Vi	ew 🔛	Device	List View	/						🐻 Gene
	Port1	Port2	Port3	B Por	t4 <u>Port5</u>	Port6	Port7	Port8	Port9	PortA	PortB	PortC	PortD	PortE	PortF	P	ortJ	
- P50	ed (🖱 In 🥊) © C	Dut 👎	Pull-u	PCN	10S out	put		T		utput 1	Norm	nal drive	output			-
O Unuse	ed () In	© 0	Out	Pull-u	PCN	10S out	put		T		utput 1	Norm	nal drive	output			-
- P52	ed (🖱 In 🥊) © C	Dut 👎	Pull-u	PCN	10S out	put		T		utput 1	Norm	nal drive	output			-
 Unuse P54 	ed () In	© 0	Out	Pull-u	PCN	10S out	put		¥		utput 1	Norm	nal drive	output			-
 P54 Onuse P55 	ed () In	© 0	Out	Pull-u	P CN	10S out	put		Ŧ		utput 1	Norm	nal drive	output			Ŧ
 P35 Unuse P56 	ed () In	0	Out	🗌 Pull-u	P CN	/IOS out	put		•	0	utput 1	High	drive ou	Itput			•
O Unuse	ed () In	0	Out	Pull-u	P CN	/IOS out	put		•		utput 1	Norm	nal drive	output			•

Figure 4-24 I/O ports – Port5

🕎 *Peripheral Functions 🛞 😹 Code Preview 👮 Device Top View 👮 Device List View											🐻 Genera
	Port1	Port2	Port3	Port4	Port5	Port6 Port7	Port8	ort9 PortA	PortB PortC	PortD PortE PortF PortJ	
	ed) In 🥊	Out 🔘	•	Pull-u	P CMOS outp	out	v	Output 1	Normal drive output	T
– P91 –––	ed) In 🥊) 🔘 Out	•	_ Pull-u	P CMOS outp	out	Ţ	Output 1	Normal drive output	v
 P32 Onuse P93 	ed 🤅) In	⊚ Out	[Pull-u	P CMOS outp	ut	-	Output 1	Normal drive output	
O Unuse	ed 🤅) In	Out	[Pull-u	P CMOS outp	ut	•	Output 1	Normal drive output	•

Figure 4-25 I/O ports - Port9

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 4-26** below.

E Console 🛛 📳 Problems	
Code Generator Console	
===== Start generate code (2016/7/29 18:19:34) =====	
M0409002:The generating source folder is: C:\workspace\CG Tutorial\	
M0409001:The following files were generated:	
M0409000: <u>src\cg src\r cg main.c</u> was generated.	
M0409000: <u>src\cg src\r cg dbsct.c</u> was generated.	
M0409000: <u>src\cg src\r cg intprg.c</u> was generated.	
M0409000: <u>src\cg src\r cg resetprg.c</u> was generated.	
M0409000: <u>src\cg src\r cg sbrk.c</u> was generated.	
M0409000: <u>src\cg src\r cg vecttbl.c</u> was generated.	
M0409000: <u>src\cg src\r cg sbrk.h</u> 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.	
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:src\cg src\r cg cgc.h 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:src\cg src\r cg sci.c was generated.	
M0409000:src\cg src\r cg sci user.c was generated.	
M0409000:src\cg src\r cg sci.h was generated.	
M0409000:src\cg src\r cg s12ad.c was generated.	
M0409000:src\cg src\r cg s12ad user.c was generated.	
M0409000:src\cg src\r cg s12ad.h was generated.	
M0409003: The operation of generating file was successful.	
===== Generate code ended (2016/7/29 18:19:35) =====	

Figure 4-26 Code generator console



4.4 Building the Project

The project template created by Code Generator can now be built. In the Project Explorer pane expand the 'src' folder. The four 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 4-27**. 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 4-27 Files excluded from the build by Code Generator

Switch back to the 'C/C++' perspective using the C/C++ button on the top right of the e² studio workspace. Use 'Build Project' from the 'Project' menu or the V 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 Web Installer 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 Web Installer. 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**.





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)
/* used to stop warnings being generated in r cg_intprg.c */
extern void r_sci2_transmitend_interrupt(void);
extern void r_sci2_receiveerror_interrupt(void);
extern void r_sci7_transmitend_interrupt(void);
/* 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+RX65N ");
    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 */
}
```



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_SCI7_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 sci7_txdone;
/* End user code. Do not edit comment generated here */
```

Insert the following code in the transmit end call-back function for SCI7:

```
static void r_sci7_callback_transmittend(void)
{
    /* Start user code. Do not edit comment generated here */
    sci7_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 SCI7 SPIMasterTransmit
* Description : This function sends SPI7 data to slave device.
* Arguments : tx_buf -
              transfer buffer pointer
           tx num -
              buffer size
* Return Value : status -
              MD OK or MD ARGERROR
MD STATUS R SCI7 SPIMasterTransmit (uint8 t * const tx buf,
                          const uint16 t tx num)
  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 SPI_Master_Send(tx_buf, tx_num);
  /* Wait for the transmit end flag */
  while (FALSE == sci7 txdone)
  {
     /* Wait */
  }
  return (status);
}
* End of function R SCI7 SPIMasterTransmit
```

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
* Arguments
          : uint16 t millisecs, number of milliseconds to wait
* Return Value : None
         void R CMT MsDelay (const uint16 t millisec)
  uint16_t ms_count = 0;
  do
  {
     R CMTO Start();
     while (FALSE == one ms delay complete)
     {
        /* Wait */
     R CMTO 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 e button in the toolbar to open the project settings. Navigate to 'C/C++ Build -> Settings -> Compiler -> Source and click the e button as shown in **Figure 5-2**.

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 	 ▲ Source ⊗ Object ⊗ Optimize ⊗ Miscellaneous ⊗ User > ⊗ CPU ⊗ PIC/PID ⊗ MISRA C Rule Check ▲ Source > ⊗ Object ⊗ Diject ⊗ Miscellaneous ⊗ User 	Include file directories	
4	🖉 🖉 Linker	Defines	🗐 🗐 🛜 🖓 🖓 🗸 🗸
?			OK Cancel

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**. e^2 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+RX65N 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 rskrx65ndef.h, r_rsk_switch.h and r_rsk_switch.c on the RSK Web Installer. 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 IRQIsFallingEdge (const uint8 t irq 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
^{\star} Description \, : This function sets/clears the falling edge trigger for the
           specified ICU IRQ.
         : uint8_t irq_no
* Arguments
           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
^{\star} Description \, : This function sets/clear the rising edge trigger for the
           specified ICU IRO.
* Arguments
          : uint8 t irq no
           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"

 $/\star$ End user code. Do not edit comment generated here $\star/$

In the same file insert the following code in the user code area inside the function r_icu_irq8_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 */
```

In the same file insert the following code in the user code area inside the function r icu irg9 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 */
```

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, SW3. 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;
/* used to stop warnings being generated in r_cg_intprg.c */
extern void r_sci2_transmitend_interrupt(void);
extern void r_sci2_receiveerror_interrupt(void);
extern void r_sci7_transmitend_interrupt(void);
/* 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+RX65N ");
    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 */
/* Variable for flagging user requested ADC conversion */
volatile uint8_t g_adc_trigger = FALSE;
/* 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);
/* 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 */
    /\,\star\, Initialize the switch module \,\star/\,
    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+RX65N ");
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;
        <mark>else</mark>
             /* do nothing */
     * 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:

```
/* Start user code for adding. Do not edit comment generated here *,
/*****
                              * * * * * * * * * * * * * * * * * *
                                                        ****
                          * * * *
* Function Name : cb_switch_press
* Description : Switch press callback function. Sets g_adc_trigger flag.
* Argument
             : none
* 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 adc value
                            static uint16 t get adc (void)
  /* 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.
         : uint16_t adc result
* Argument
* 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";
   /\star Convert ADC result into a character string, and store in the local.
    Casting to ensure use of correct data type. */
   a = (uint8_t)((adc_result & 0x0F00) >> 8);
  lcd_buffer[6] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
a = (uint8_t)((adc_result & 0x00F0) >> 4);
```



Open the file 'r_cg_s12ad.h' by double-clicking on it. Insert the following code 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);
```

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

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

Insert the following code 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 Web Installer. These files can be found in the RSK+RX65N_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_SCI2_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 SCI7 SPIMasterTransmit(uint8 t * const tx buf, const uint16 t tx num);
MD_STATUS R_SCI2_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 sci7_txdone;
static volatile uint8_t sci2_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_sci2_callback_transmitend function:

```
static void r_sci2_callback_transmitend(void)
{
    /* Start user code. Do not edit comment generated here */
    sci2_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_sci2_callback_receiveend function:

```
static void r_sci2_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 SCI2 receive buffer and callback function again */
    R_SCI2_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 SCI2 AsyncTransmit
 Description : This function sends SCI2 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 SCI2 AsyncTransmit (uint8 t * const tx buf, const uint16 t tx num)
{
  MD STATUS status = MD OK;
  /* clear the flag before initiating a new transmission */
  sci2 txdone = FALSE;
  /* Send the data using the API */
  status = R_SCI2_Serial_Send(tx_buf, tx_num);
  /* Wait for the transmit end flag */
  while (FALSE == sci2_txdone)
  {
     /* Wait */
  }
  return (status);
}
* End of function R SCI2 AsyncTransmit
```



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:

```
/* Start user code for include. Do not edit comment generated here */
#include "r_okaya_lcd.h"
#include "r rsk switch.h"
#include "r_rsk_debug.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:

```
/* Start user code for global. Do not edit comment generated here */
/* Variable for flagging user requested ADC conversion */
volatile uint8_t g_adc_trigger = FALSE;
/* 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);
/* 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;
```

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

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+RX65N ");
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 SCI2 receive buffer and callback function */
    R_SCI2_Serial_Receive((uint8_t *)&g_rx_char, 1);
    /* Enable SCI2 operations */
    R_SCI2_Start();
    while (1U)
    {
        uint16 t adc result;
         /* Wait for user requested A/D conversion flag to be set (SW1 or SW2) ^{\prime}
        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 */
             if (16 == (++adc_count))
```

RENESAS

```
{
            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)
    {
        /* 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;
   }
   else
   {
        /* do nothing */
    }
/* 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.
* Argument : uint8_t : adc_count
              uint16 t: adc result
* Return value : none
                    static void uart display adc (const uint8 t adc count, const uint16 t adc result)
{
   /* Declare a temporary variable */
   char a;
   /* Declare temporary character string */
   static char uart buffer[] = "ADC xH Value: xxxH\r\n";
   /\star 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));
   /* 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 SCI2 (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 SCI2. 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:

/* Start user code for include. Do not edit comment generated here */
#include "r_okaya_lcd.h"
#include "r_rsk_switch.h"
#include "r rsk debug.h"
#include "rskrx65ndef.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:

```
/* 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;
/* 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;
/* Prototype declaration for led display count */
static void led display count(const uint8 t count);
```

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

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+RX65N ");
    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 SCI2 receive buffer and callback function */
R SCI2 Serial Receive((uint8 t *)&g rx char, 1);
/* Enable SCI2 operations */
R SCI2 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);
        /* Increment the adc_count and display using the LEDs */
        if (16 == (++adc_count))
        {
            adc count = 0;
        ied_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;
    }
    else
    {
        /* do nothing */
    }
/* End user code. Do not edit comment generated here */
```

1



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 LEDSO-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 */
  LEDO = (uint8 t) ((count & 0x01) ? LED ON : LED OFF);
  LED1 = (uint8_t)((count & 0x02) ? LED_ON : LED_OFF);
LED2 = (uint8_t)((count & 0x04) ? LED_ON : LED_OFF);
  LED3 = (uint8 t) ((count & 0x08) ? LED ON : LED OFF);
* End of function led_display_count
                           *****
 *****
/* End user code. Do not edit comment generated here */
```

Select 'Build Project' from the 'Build' 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 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' -> 'Debugger' -> '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+RX65N the setting should be 24.0000).

For more information on powering the RSK+RX65N please refer to the Usermanual.

Debug hardware: E2 Lite (RX) Target Device: R5F565N9 GDB Settings Connection Settings Debug Tool Settings		ш
Clock		
Main Clock Source	EXTAL	_
Extal Frequency[MHz]	24.0000	
Permit Clock Source Change On Writing Internal Flash Memor	Yes	
✓ Connection with Target Board	, 	
Emulator	(Auto)	
Connection Type	Fine	
JTag Clock Frequency[MHz]	6.00	
Fine Baud Rate[Mbps]	1.50	
Hot Plug	No	
⊿ Power		
Power Target From The Emulator (MAX 200mA)	No	
Supply Voltage	3.3V	
▲ CPU Operating Mode		
Register Setting	Single Chip	Ŧ
III	4	
	Revert Apply	

Figure 6-1 Debug Configurations



Connect the E2 Lite to the PC and the RSK+ E1 connector. Connect the Pmod LCD to the PMOD1 connector. Connect the center positive +5V PSU to the PWR connector on the RSK+ and apply power.

In the Project Explorer pane, ensure that the 'CG_Tutorial' project is selected. To debug the project, click the button. The dialog shown in **Figure 6-2** will be displayed.

e ² Conf	irm Perspective Switch
\bigcirc	This kind of launch is configured to open the Debug perspective when it suspends.
	This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.
	Do you want to open this perspective now?
<mark>▼ R</mark> em	nember my decision
	<u>Y</u> es <u>N</u> o

Figure 6-2 Perspective Switch Dialog

Click Remember my decision to skip this dialog later. Click 'YES' 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**.

Image: strap force-64bit-double (7.8.2)
: trap) force-64bit-double (7.8.2)
force-64bit-double (7.8.2)
c r_cg_resetprg.c ⋈ [№] ₄
to regrescipique as 4
e FPSW */
W RMbits=01 (round to zero
W RMbits=00 (round to near
W DNbit=1 (denormal as zer
W DNbit=1 (denormal as zer
W DNbit=1 (denormal as zer
WF

For more information on the e^2 studio debugger refer to the Tutorial manual. To run the code click the lease button. The debugger will stop again at the beginning of the main function. Press lease 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	
	?	Help Contents	
	? ?	Search	
		Dynamic Help	

For information about the RX65N group microcontroller refer to the RX65N Group, RX651 Group Hardware Manual.

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

Technical Contact Details

Please refer to the contact details listed in section 8 of the "Quick Start Guide".

General information on Renesas microcontrollers can be found on the Renesas website at: <u>https://www.renesas.com/</u>

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.

© 2016 Renesas Electronics Europe Limited. All rights reserved.

- © 2016 Renesas Electronics Corporation. All rights reserved.
- © 2016 Renesas System Design Co., Ltd. All rights reserved.



REVISION HISTORY	RSK+RX65N Code Generator Tutorial Manual

Rev.	Date	Description	
		Page	Summary
1.00	Jun 30, 2016	_	First Edition issued

Renesas Starter Kit+ Manual: Code Generator Tutorial Manual

Publication Date: Rev. 1.00 Jun 30, 2016

Published by: Renesas Electronics Corporation



Renesas Electronics Corporation

http://www.renesas.com

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

SALES OFFICES

Renesas Electronics America Inc. 2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A. Tel: +1-408-588-6000, Fax: +1-408-588-6130 Renesas Electronics Canada Limited 9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3 Tel: +1-905-237-2004 Renesas Electronics Europe Limited Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K Tel: +44-1628-585-100, Fax: +44-1628-585-900 Renesas Electronics Europe GmbH Arcadiastrasse 10, 40472 Düsseldorf, Germany Tel: +49-211-6503-0, Fax: +49-211-6503-1327 Renesas Electronics (China) Co., Ltd. Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China Tel: +86-10-8235-1155, Fax: +86-10-8235-7679 Renesas Electronics (Shanghai) Co., Ltd. Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333 Tel: +86-21-2226-0888, Fax: +86-21-2226-0999 Renesas Electronics Hong Kong Limited Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong Tel: +852-2265-6688, Fax: +852 2886-9022 Renesas Electronics Taiwan Co., Ltd. 13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan Tel: +886-2-8175-9600, Fax: +886 2-8175-9670 Renesas Electronics Singapore Pte. Ltd. 80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949 Tel: +65-6213-0200, Fax: +65-6213-0300 Renesas Electronics Malaysia Sdn.Bhd. Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd. No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India Tel: +91-80-67208700, Fax: +91-80-67208777 Renesas Electronics Korea Co., Ltd. 12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea Tel: +82-2-558-3737, Fax: +82-2-558-5141

> © 2016 Renesas Electronics Corporation. All rights reserved. Colophon 4.0

RX65N Group



R20UT3564EG0100