

# 32

## RX65N Group

Renesas Starter Kit+ Smart Configurator Tutorial Manual For e<sup>2</sup> 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 Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics www.renesas.com

Rev. 1.00 May 2017

#### Notice

- 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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
- 2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving 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, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
- 3. 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 shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
- 5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended 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 (traffic lights); large-scale communication equipment; key financial terminal systems; safety control 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 (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability 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. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident 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 Renesas Electronics 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 ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, 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 by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, 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 investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
- 9. Renesas Electronics products and technologies shall 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 shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
- 10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
- 11. This document shall not be reprinted, 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.
- (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.

#### General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

- <sup>3</sup>⁄<sub>4</sub> The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
- 2. Processing at Power-on

The state of the product is undefined at the moment when power is supplied.

- <sup>3</sup>⁄<sub>4</sub> The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
- 3. Prohibition of Access to Reserved Addresses

Access to reserved addresses is prohibited.

- <sup>3</sup>⁄<sub>4</sub> The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
- 4. Clock Signals

After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

- When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
- 5. Differences between Products

Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

<sup>3</sup>⁄<sub>4</sub> The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

### 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 Smart Configurator for RX together with the e<sup>2</sup> 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<sup>2</sup> 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.

In this manual, the display may differ slightly from screen shots. There is no problem in reading this manual.

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-2MB User's Manual	R20UT3888EG
Tutorial Manual	Provides a guide to setting up RSK+ environment, running sample code and debugging programs.	RSK+RX65N-2MB Tutorial Manual	R20UT3892EG
Quick Start Guide	Provides simple instructions to setup the RSK+ and run the first sample.	RSK+RX65N-2MB Quick Start Guide	R20UT3893EG
Smart Configurator Tutorial	Provides a guide to code generation and importing into the e <sup>2</sup> studio IDE.	RSK+RX65N-2MB Smart Configurator Tutorial Manual	R20UT3894EG
Schematics	Full detail circuit schematics of the RSK+.	RSK+RX65N-2MB Schematics	R20UT3887EG
Hardware Manual	Provides technical details of the RX65N microcontroller.	RX65N Group, RX651 Group Hardware Manual	R01UH0590EJ

## 2. List of Abbreviations and Acronyms

Full Form
Analog-to-Digital Converter
Application Programming Interface
bits per second
Compare Match Timer
COMmunications port referring to PC serial port
Central Processing Unit
Digital Versatile Disc
Renesas On-chip Debugging Emulator
Graphical User Interface
Integrated Development Environment
Interrupt Request
Liquid Crystal Display
Light Emitting Diode
Least Significant Bit
Low Voltage Detect
Micro-controller Unit
Most Significant Bit
Personal Computer
Phase-locked Loop
This is a Digilent Pmod <sup>™</sup> Compatible connector. Pmod <sup>™</sup> is registered to Digilent Inc.
Digilent-Pmod_Interface_Specification
Power Supply Unit
Random Access Memory
Read Only Memory
Renesas Starter Kit+
Real Time Clock
Serial Array Unit
Serial Communications Interface
Serial Peripheral Interface
Timer Array Unit
Thin Film Transistor
Timer Pulse Unit
Universal Asynchronous Receiver/Transmitter
Universal Serial Bus
Watchdog Timer

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

## Table of Contents

1. Overview	8
1.1 Purpose	8
1.2 Features	8
2. Introduction	9
	4.0
3. Project Creation with e <sup>2</sup> studio	
3.1 Introduction	
3.2 Creating the Project	10
4. Smart Configurator Using the e <sup>2</sup> studio plug-in	13
4.1 Introduction	
4.2 Project Configuration using Smart Configurator - Overview page	
4.3 Clocks configuration page	15
4.3.1 Clocks configuration	
4.4 Components page	16
4.4.1 Add a software component into the project	
4.4.2 Compare Match Timer	17
4.4.3 Interrupt Controller	
4.4.4 Ports	
4.4.5 SCI/SCIF Asynchronous Mode	
4.4.6 SPI Clock Synchronous Mode	
4.4.7 Single Scan Mode S12AD	
4.5 Pins configuration page	
4.5.1 Change pin assignment of a software component	
4.6 Building the Project	
5. User Code Integration	39
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	53
5.5.2 Main UART code	
5.6 LED Code Integration	
6. Debugging the Project	60
7. Additional Information	60
	02

## RENESAS

### RSK+RX65N-2MB

RENESAS STARTER KIT+

#### 1.1 Purpose

This RSK+ is an evaluation tool for Renesas microcontrollers. This manual describes how to use the e<sup>2</sup> studio IDE Smart Configurator 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 e<sup>2</sup> studio.
- Code generation using the Smart Configurator 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 Smart Configurator plug-in for the RX family together with the e<sup>2</sup> studio IDE to create a working project for the RSK+ platform. The tutorials help explain the following:

- Project generation using e<sup>2</sup> studio
- Detailed use of the Smart Configurator plug-in for e<sup>2</sup> studio
- Integration with custom code
- Building the project in e<sup>2</sup> 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 'Outputs debugging information' option not 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 e<sup>2</sup> studio debugger, compiler toolchains or the E2 emulator Lite. Please refer to the relevant user manuals for more in-depth information.



## 3. Project Creation with e<sup>2</sup> 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 Smart Configurator. This project generation step is necessary to create the MCU-specific source, project and debug files.

#### 3.2 Creating the Project

location for th	udio and select a suitable ne project workspace.	Workspace: CAWorkspace	ace directory to store its preference	es and development artifacts.	Rrowse
<ul> <li>In the Welco C/C++ project</li> </ul>	me page, click 'Create a new ct'.	Create a new et  Create a new et  Import existi Import existi Import existi Review IDE c	e <sup>2</sup> studio projects from the filesystem or configuration settings is most fiercely contested preferences sting file	Lieb         Overview         Get an overview of the features         Controls         Go through tutorials         You the samples         Try out the samples         Find out what is new	-
dialog, sele	lates for New C/C++ Project' ecting 'Renesas RX' -> CC-RX C/C++ Executable	enesas KX	C++ Project GCC for Renesas A C/C++ Executabl using the GCC for R GCC for Renesas A C/C++ Library Pr the GCC for Renesas Renesas CC-RX C Renesas CC-RX coole Renesas CC-RX coole	Is RX Toolchain. C/C++ Executable Proj r Renesas RX using the chain. C/C++ Library Project roject for Renesas RX using toolchain.	oject 7 ect



#### RSK+RX65N-2MB

Click 'Next'. Here near CM Encode Preas Performer (Lond Performer (Lond) Performer (Lond) Perform			
<ul> <li>In the 'Select Coolina in, device &amp; debug settings' dialog, select the options as shown in the screenshot opposite.</li> <li>In 'Toolchains': choose 'Renesa CCRX, Toolchains'.</li> <li>The R5F65NEDXFC MCU is found under RX600 &gt; RX65N -&gt; RX65N - 176 pin.</li> <li>Click 'Next'.</li> <li>In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> </ul>	٠	Enter the project name 'SC_Tutorial'.	e ×
<ul> <li>In the 'Select toolchain, device &amp; debug stream 's and 's a</li></ul>		Click 'Next'.	
<ul> <li>In the 'Select toolchain, device &amp; debugs shown in the screenshow toopsoite.</li> <li>In 'Toolchains' choose 'Renessa CCRX, Toolchain'.</li> <li>Click Next'.</li> <li>In the 'Select tooling Assistant Tool' dialog, select the options' as four in the screenshow to provide the selection of the screensho</li></ul>			New Kenesas CC-RX Executable Project
<ul> <li>In the 'Select toolchain, device &amp; debugs shown in the screenshow toopsoite.</li> <li>In 'Toolchains' choose 'Renessa CCRX, Toolchain'.</li> <li>Click Next'.</li> <li>In the 'Select tooling Assistant Tool' dialog, select the options' as four in the screenshow to provide the selection of the screensho</li></ul>			Project name: SC_Tutoria
<ul> <li>In the 'Select toolchain, device &amp; debug settings' dialog, select the options as shown in the screenshot opposite.</li> <li>In Toolchains' choose 'Renesas CCRX Toolchain'.</li> <li>The R5F565NEDXFC MCU is found under RX660 -&gt; RX65N -&gt; RX65N - 176 pin.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>In the 'Select toolchain, device &amp; debugs stims' dialog, select the options as shown in the screenshot opposite.</li> <li>In 'Toolchains' choose 'Renessas CCRX. To Toolchain'.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>In the 'Select toolchain, device &amp; debug settings' dialog, select the options as shown in the screenshot opposite.</li> <li>In 'Toolchains' choose 'Renesas CCRX Toolchains'.</li> <li>The R5F565NEDXFC MCU is found under RX6600 -&gt; RX655N -&gt; RX655N - 176 pin.</li> <li>Click 'Next'.</li> <li>In the 'Select Coding Assistant Tool' dialog. select 'Smart Configurator'.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>In the 'Select toolchain, device &amp; debugs status in the screenshot opposite.</li> <li>In 'Toolchains' choose 'Renessa CCRX, Toolchain'.</li> <li>In 'Toolchains' choose 'Renessa CCRX, RX65N - 176 pin.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>In the 'Select toolchain, device &amp; debug as shown in the screenshot opposite.</li> <li>In 'Toolchains' choose 'Renesas CCRX. Toolchain'.</li> <li>The R5565NEDxFC MCU is found under RX600 -&gt; RX65N -&gt; Click 'Next'.</li> <li>Click 'Next'.</li> <li>In the 'Select Coding Assistant Tool' dialog. select 'Smart Configurator'.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>In the 'Select toolchain, device &amp; debug stings' dialog, select the options as shown in the screenshot opposite.</li> <li>In 'Toolchains' choose 'Reness CCRX Toolchain'.</li> <li>The R5F565NEDxFC MCU is found under RX600 - RX65N -&gt; RX65N - 176 pin.</li> <li>Click 'Next'.</li> <li>In the 'Select Coding Assistant Tool' dialog. select 'Smart Configurator'.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>In the 'Select toolchain, device &amp; debug settings' dialog, select the options as shown in the screenshot opposite.</li> <li>In 'Toolchains' choose 'Renessas CCRX Toolchains'.</li> <li>The R5F565NEDxFC MCU is found under RX600 -&gt; RX65N -&gt; RX65N - 176 pin.</li> <li>Click 'Next'.</li> </ul>			Working sets: Sglect
<ul> <li>In the 'Select toolchain, device &amp; debug settings' dialog, select the options as shown in the screenshot opposite.</li> <li>In 'Toolchains' choose 'Renessas CCRX Toolchains'.</li> <li>The R5F565NEDxFC MCU is found under RX600 -&gt; RX65N -&gt; RX65N - 176 pin.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>In the 'Select toolchain, device &amp; debug settings' dialog, select the options as shown in the screenshot opposite.</li> <li>In 'Toolchains' choose 'Renessas CCRX Toolchains'.</li> <li>The R5F565NEDxFC MCU is found under RX600 -&gt; RX65N -&gt; RX65N - 176 pin.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>In the 'Select toolchain, device &amp; debug settings' dialog, select the options as shown in the screenshot opposite.</li> <li>In 'Toolchains' choose 'Renessas CCRX Toolchains'.</li> <li>The R5F565NEDxFC MCU is found under RX600 -&gt; RX65N -&gt; RX65N - 176 pin.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>In the 'Select toolchain, device &amp; debug settings' dialog, select the options as shown in the screenshot opposite.</li> <li>In 'Toolchains' choose 'Renessas CCRX Toolchains'.</li> <li>The R5F565NEDxFC MCU is found under RX600 -&gt; RX65N -&gt; RX65N - 176 pin.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>In the 'Select toolchain, device &amp; debug settings' dialog, select the options as shown in the screenshot opposite.</li> <li>In 'Toolchains' choose 'Renessas CCRX Toolchains'.</li> <li>The R5F565NEDxFC MCU is found under RX600 -&gt; RX65N -&gt; RX65N - 176 pin.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> </ul>			(?) < <u>Back Next&gt;</u> Einish Cancel
<ul> <li>In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> </ul>	•	In the 'Select toolchain, device & debug	ei X
<ul> <li>shown in the screenshot opposite.</li> <li>In 'Toolchains' choose 'Renesas CCRX Toolchain'.</li> <li>The RSF565NEDXFC MCU is found under RX600 -&gt; RX65N -&gt; RX65N - 176 pin.</li> <li>Click 'Next'.</li> <li>Click 'Next'.</li> <li>In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> </ul>	•		
<ul> <li>In 'Toolchains' choose 'Renessas CCRX Toolchain'.</li> <li>The RSF56SNEDxFC MCU is found under RX600 -&gt; RX65N -&gt; RX65N -&gt; RX65N -176 pin.</li> <li>Click 'Next'.</li> <li>Click 'Next'.</li> <li>Click 'Next'.</li> </ul>		<b>a b</b>	
<ul> <li>In Toolchain'.</li> <li>The R5F565NEDxFC MCU is found under RX600 -&gt; RX65N -&gt; RX65N -&gt; RX65N -&gt; RX65N -&gt; RX65N -&gt; Click 'Next'.</li> <li>Click 'Next'.</li> <li>Click 'Next'.</li> <li>In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>The R5F565NEDXFC MCU is found under RX600 -&gt; RX65N -&gt;</li></ul>	٠		
<ul> <li>The RSF-565NEDXFC MCU is found under RX600 -&gt; RX65N -&gt; RX65N - 176 pin.</li> <li>Click 'Next'.</li> <li>Click 'Next'.</li> <li>In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> </ul>		Toolchain'.	Toolchain: Renesas CCRX 🗸
under RX660 -> RX65N -> RX65N - 176 pin. Click 'Next'. • Click 'Next'. • In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'. • Click 'Next'. • Click 'Next'.	•	The R5F565NEDxFC MCU is found	
RX65N - 176 pin. Click 'Next'. In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'. Click 'Next'. In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'. Click 'Next'. Click 'Next'.			
<ul> <li>Click 'Next'.</li> <li>Click 'Next'.</li> <li>In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>Project Type Default</li> <li>Project Type Default</li> <li>The Number of Cases Release Configuration</li> <li>The Number of Cases Release Configuration</li> <li>Select 'Smart Configurator'.</li> <li>Click 'Next'.</li> </ul>		RX65N - 176 pin.	
<ul> <li>In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> </ul>	•	Click 'Next'.	Create Debug Configuration
<ul> <li>In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> <li>Click 'Next'.</li> </ul>			Project Type: Default V RX Simulator V
<ul> <li>In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> <li>Click 'Next'.</li> </ul>			Create Release Configuration
<ul> <li>In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> <li>Click 'Next'.</li> </ul>			
<ul> <li>In this concert counting relation in roots a datalog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> </ul> New Reness CC XX Executable Project Select Coding Assistant settings       Image: Configurator of the confi			< Back
<ul> <li>In this concert counting relation in roots a datalog, select 'Smart Configurator'.</li> <li>Click 'Next'.</li> </ul> New Reness CC XX Executable Project Select Coding Assistant settings       Image: Configurator of the confi			
set Click 'Next'.	٠		
Click 'Next'.      Click 'Next'.      Click 'Next'.      Click 'Next'.      Click 'Next'.      Contract Configuration     Configurati		select 'Smart Configurator'.	
Use Peripheral Code Generator         Due FIT Module         Due FIT Module         Smart Configurator is a single User Interface that combines the functionalities of Code Generator and FIT Configurator which imports, configurer and generate different types of drivers and middleware modules.         Smart Configurator rencompasses unified clock configuration view, interrupt configuration view and pin configurator with modules, interrupts and pin according and middleware modules will be notified.         Hardware resources configurator is available only for the supported devices)         User Application         User Application         Orfigurator is available only for the supported devices)         Smart Configurator is available only for the supported devices         User Application         Mitted and imported         User Application         Mitted and imported         Selected in GUI         Selected in GUI         Mitted and Imported         User Hardware         Mitted and Imported         Mitted and Imported         Selected in GUI         Mitted and Imported         Mitted and Imported         Mitted and Imported         Mitted and Imported	•	Click 'Next'.	
In the FIT Module  In the F			
Smart Configurator is a single User Interface that combines the functionalities of Code Generator and FIT Configurator which imports, configures and generates different types of drivers and middleware modules. Smart Configurator encompasses unified tock configuration view and pin configuration view. Hardware resources confifici n peripheral modules, interrupts and pins occurred in different types of drivers and middleware modules. (Smart Configurator is available only for the supported devices) User Application Driver and Middleware Driver Code Configured in GU and Generated in GU and Imported			
Smart Configurator encompasses unified clock configuration view, interrupt configuration view, and pin configuration view. Hardware resources conflict in peripheral modules, interrupts and pins occurred in different types of drivers and middleware modules will be notified. (Smart Configurator is available only for the supported devices) User Application Driver and Middleware Driver Code Configured in GUI Selected in GUI and Generated MCU Hardware			Smart Configurator is a single User Interface that combines the functionalities of Code Generator and FIT Configurator which imports,
will be notified. (Smart Configurator is available only for the supported devices) User Application Driver and Middleware Driver Code Configured in GU Selected in GU MCU Hardware			Smart Configurator encompasses unified clock configuration view, interrupt configuration view and pin configuration view.
User Application Driver and Middleware Driver Code Configured in GU Selected in GU and Imported MCU Hardware			will be notified.
Driver Code Configured in GU and Generated MCU Hardware			(Smart Configurator is available only for the supported devices)
Driver Code Configured in GU and Generated MCU Hardware			User Application
Driver Code Configured in GU and Generated MCU Hardware			Driver and Middleware
			Driver Code FIT Modules + O
			Configured in GUI Selected in GUI and Generated
			MCU Hardware
② < <u>Back</u> <u>Next&gt;</u> <u>Finish</u> Cancel			
② < <u>Back</u> <u>Next</u> > <u>Finish</u> Cancel			
? < Back Next > Einish Cancel			
? < Back Next > Einish Cancel			
			(?) < <u>Back</u> <u>Next</u> > <u>Finish</u> Cancel



#### RSK+RX65N-2MB

		e <sup>2</sup> - • ×
•	Click 'Next'.	New Renesas CC-RX Executable Project
		New Kenesas CC- KX Executable Project Settings The Contents of Files to be Generated
		What kind of initialization routine would you like to create?
		Use I/O Library
		Number of I/O Streams:
		20 🗘
		< Back
•	A summary dialog will appear, click	ē – 🗆 X
	'Finish' to complete the project	New Renesas CC-RX Executable Project
	generation.	Summary of project "SC_Tutorial"
		TOOLCHAIN NAME : Renesas CCRX
		TOOLCHAIN VERSION : v2.07.00
		GENERATION FILES :
		(?) <u>&lt; Back</u> <u>Next</u> <u>Einish</u> Cancel
•	You may be prompted to open the Smart	■ Open Associated Perspective? ×
	Configurator perspective. Click 'Yes' to	This kind of project is associated with the C/C++ perspective. Do you want to open
	open the Smart Configurator perspective.	this perspective now?
		Remember my decision
		Yes No
		<u> </u>
•	Wait for file generation to start.	Progress Information
		Smart Configurator operation in progress
		Preparing startup code
		Cancel



## 4. Smart Configurator Using the e<sup>2</sup> studio plug-in

#### 4.1 Introduction

The Smart Configurator plug-in for the RX65N has been used to generate the sample code discussed in this document. Smart Configurator for e<sup>2</sup> studio is a plug-in tool for generating template 'C' source code and project settings for the RX65N. When using Smart Configurator, it supports user with a visual way of configuring the target device, clocks, software components, hardware resources and interrupts for the project; Thereby bypassing the need, in most cases, to refer to sections of the Hardware Manual.

Once the user has configured the project, the 'Smart Configurator' function is used to generate three code modules for each specific MCU feature selected, general folder, r\_bsp folder, r\_config folder and r\_pincfg folder. These code modules are name 'Config\_xxx.h', 'Config\_xxx.c', and 'Config\_xxx\_user.c', where 'xxx' is an acronym for the relevant MCU feature, for example 'CMT'. 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 \*/

Smart Configurator 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 revisit Smart Configurator to change any MCU operating parameters.

By following the steps detailed in this Tutorial, the user will generate an e<sup>2</sup> studio project called SC\_Tutorial. The fully completed Tutorial project is contained on the RSK Web Installer (<u>https://www.renesas.com/rskrx65n2mb/install</u>) and may be imported into e<sup>2</sup> 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 Smart Configurator to generate their own custom projects for e<sup>2</sup> studio.

The SC\_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 Smart Configurator in 'Clocks configuration page', 'Components page', 'Pins configuration page' and 'Building the Project', the reader is guided through each of the peripheral function configuration pages, familiarised with the structure of the template code, and adding their own code to the user code areas provided by the Smart Configurator.

The Smart Configurator installer is contained on the RSK Web Installer. This installer must be run before proceeding to the next section.



#### 4.2 **Project Configuration using Smart Configurator - Overview page**

In this section, a brief tour of Smart Configurator is presented. For further details of the Smart Configurator paradigm and reference, refer to the Smart Configurator User Guide. You can download the latest document from: https://www.renesas.com/smart-configurator.

From the e<sup>2</sup> studio menus, select 'Window -> Perspective -> Open Perspective -> Other. In the 'Open Perspective' dialog shown in **Figure 4-1**, select 'Smart Configurator' and click 'OK'.

e <sup>2</sup> Open Perspective	_	-		×
電 C/C++ (default) 옐 Code Generator 茶 Debug				
載 Java 家 Java Browsing は Java Type Hierarchy 電 Remote System Explorer				
Cripting				
E <sup>O</sup> Team Synchronizing				
	ОК		Cancel	

Figure 4-1 Open Perspective Dialog

The Smart Configurator initial view is displayed as illustrated in Figure 4-2.

<u>File Edit Navigate Search Project Renesas Views R</u>	un <u>W</u> indow <u>H</u> elp						
🔦 🎋 🔳 🎋 Debug 🗸 🖻 SC_Tu	utorial HardwareDebug 🛛 🗸 🔅	📑 🕶 🔛 🕼   🕲 🕶	🍕 • 🛞 🐘 • 🦠 🗰 💷 😭 🍪	🏟 📲 🖋 😋 🐐 • 💁 • 🖋 • 🖢 •	{{ <b>v</b> \$\$ \$\$ \$\$ \$\$ \$\$ <b>•</b> \$\$ \$\$	*	Quick Access 😰 😼 🎄 🛣
🕒 Project Explorer 😒 🛛 🖃 😫 🔍 📟 🗖	Interview SC_Tutorial.scfg ≥					MCU Package 😫	
SC_Tutonal	Overview information				۵ 🖏	👪 🔺 🔎 🔊 🖻 🕒 Type pin	function
v 🚑 src	➡ General Information				0		
> 🦢 smc_gen > 🗟 SC_Tutorial.c 🗑 custom.bat	This editor allows you to modify	the settings stored in configu	uration file (.scfg)		Ų		
SC_Tutorial HardwareDebug.launch SC_Tutorial.scfg	Board						
tipp SC_lutional.scrg	Allow board and device selection	n					
	Clocks Allow clock configuration		Applic	ation under lopment Middleware			
	Components Allow software component sele	stion and configuration	Device	RTOS			
		contaile configuration	1111	+ Pins			
	Pins Allow general pin configuration	and ain configuration for cal	erted coftware component				
	Allow general pin conliguration	and princonniguration for sei	ected software component				ESAS
	Interrupt						
	Allow general interrupt configu	ration and interrupt configura	tion for selected software component				
	✓ Current Configuration						(65N
	Selected board/device: R5F5651	VEDxFC					
	Selected components:					R5F56	5NEDxFC
	Component	Version	Configuration				
	♥ r_bsp	Generic(v=3.60)	r_bsp(used)				
							*****
< >	Overview Board Clocks Compo	nent Pins Interrupt				▶ Legend	
Console 13			🔍 🔐 📴 🚽 🖬 🕶 🗂	S Configuration Problems			~
Smart Configurator Output				0 items			
M06000002: File generated:src\smc_gen/_config M06000002: File generated:src\smc_gen/_config M06000002: File generated:src\smc_gen/general/ M060000002: File generated:src\smc_gen/general/ M060000002: File generated:src\smc_gen/general/	<pre>\r_bsp_irq_priority_cfg.h r_smc_interrupt.c r smc_interrupt.h</pre>			Description	Туре		
M00000002: Code generation is successful							
<			>				
0 items selected							·a 🔰 🐼 😌 🔶 🖉

Figure 4-2 Overview page

Smart Configurator 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 e<sup>2</sup> studio project that builds and runs without error.

### 4.3 Clocks configuration page

Clocks configuration page configures clocks of the device selected. Clock source, frequency, PLL settings and clock divider settings can be configured for the output clocks. Clock configurations will be reflected to r\_bsp\_config.h file in \src\smc\_gen\r\_config.

#### 4.3.1 Clocks configuration

**Figure 4-3** shows a screenshot of Smart Configurator with the Clocks tab. Click on the 'Clocks' 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**.

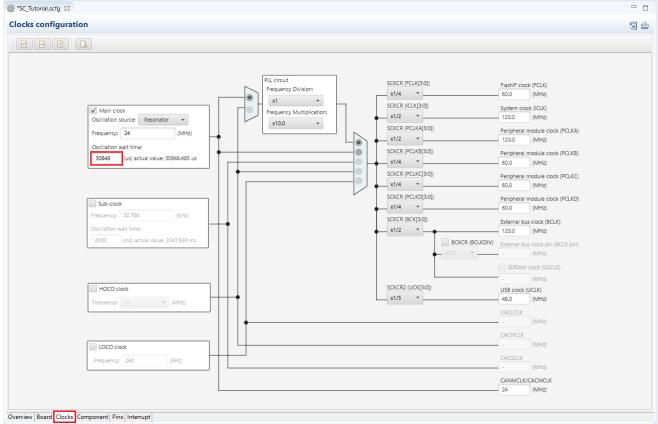


Figure 4-3 Clocks Configuration page



#### 4.4 Components page

Drivers and middleware are handled as software components in Smart Configurator. The Components page allows user to select and configure software components.

omponents	↓ª 🗖 🕀 🛱
	ت 💕
type filter text	
🗸 🗁 Startup	
🗸 🗁 Generic	
💣 r_bsp	
🗁 Drivers	
🗁 Middleware	
Application	

#### Figure 4-4 Components page

#### 4.4.1 Add a software component into the project

Smart Configurator supports two types of software components: Code Generator and Firmware Integration Technology. 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 by component of Code Generator.

lick 'Add component' 눟 icon.	🔅 SC_Tutorial.scfg ⊠		
	Software compon	ent configuration	
	Components	.¦ª_ E ⊞ 🛟 ▼	Configure
	type filter text Startup Generic r_bsp Drivers Middleware Application		

Figure 4-5 Add a Code Generator component (1)

In 'Software Component Selection' dialog -> Type, select 'Code Generator'.

e <sup>2</sup> New C	omponent				$\times$
	Component Selection mponent from those available	in list		ŧ	
Function	All				$\sim$
Туре	All				$\sim$
Filter	All Firmware Integration Techno Code Generator	blogy			
Compor	nent	Туре	Version		^
🖶 8-Bit	Timer	Code Generator	1.0.0		
Cloci	k Frequency Accuracy Mea	Code Generator	1.0.0		
🖶 Com	pare Match Timer	Code Generator	1.1.1		

Figure 4-6 Add a Code Generator component (2)



#### 4.4.2 **Compare Match Timer**

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. Select 'Compare Match Timer' as shown in Fig ro 4-7 below then click 'Next'

Software Component Selection Select component from those available in list  Function All Type Code Generator Filter Component Type Version Besit Timer Code Generator Code	Type Code Generator Code Generator	1.0.0	-
Function       All         Type       Code Generator         Filter       Code Generator         100       Code Generator </td <td>Type Code Generator Mea Code Generator Code Generator</td> <td>1.0.0</td> <td></td>	Type Code Generator Mea Code Generator Code Generator	1.0.0	
Type       Code Generator         Filter       Type       Version         @ Selit Timer       Code Generator       1.0.0         @ Clock Frequency Accuracy Mea       Code Generator       1.0.0         @ Compare Match Timer       Code Generator       1.1.1         @ Compare Match Timer       Code Generator       1.1.1         @ Compare Match Timer       Code Generator       1.1.0         @ Compare Match Timer       Code Generator       1.0.0         @ D/A Converter       Code Generator       1.0.0         @ Data Operation Circuit       Code Generator       1.0.0         @ Data Operation Circuit       Code Generator       1.0.0         @ Data Operation Circuit       Code Generator       1.0.0         @ Data Transfer Controller       Code Generator       1.0.0         @ Data Transfer Controller       Code Generator       1.0.0         @ Data Transfer Controller       Code Generator       1.0.0         @ Court Sout Match Stable       Code Generator       1.0.0         @ Show only last version       Code Generator       1.0.0         @ Show only last version       Description       This software component provides configurations for 16-bit/32-bit timer with module	Code Generator Code Generator Code Generator	1.0.0	
Type       Code Generator         Filter       Component       Type       Version         # 8-Bit Timer       Code Generator       1.0.0         # Clock Frequency Accuracy Mea       Code Generator       1.0.0         # Compare Match Timer       Code Generator       1.1.1         # Compare Match Timer       Code Generator       1.1.1         # Compare Match Timer       Code Generator       1.1.0         # CRC Calculator       Code Generator       1.0.0         # D/A Converter       Code Generator       1.0.0         # Data Operation Circuit       Code Generator       1.0.0         # Data Transfer Controller       Code Generator       1.0.0         # DMA Controller       Code Generator       1.0.0         # DMA Controller       Code Generator       1.0.0         # Show only last version       Description       >         This software component provides configurations for 16-bit/32-bit timer with module       >	Code Generator Code Generator Code Generator	1.0.0	
Filter     Type     Version       Gomponent     Type     Version       # 8-Bit Timer     Code Generator     1.0.0       # Clock Frequency Accuracy Mea     Code Generator     1.0.0       # Clock Frequency Accuracy Mea     Code Generator     1.1.1       # Compare Match Timer     Code Generator     1.1.1       # Compare Match Timer     Code Generator     1.1.0       # CRC Calculator     Code Generator     1.0.0       # D/A Converter     Code Generator     1.0.0       # Data Operation Circuit     Code Generator     1.0.0       # Data Transfer Controller     Code Generator     1.0.0       # DMA Controller     Code Generator     1.0.0       # DMA Controller     Code Generator     1.0.0       # Show only last version     Description     >       This software component provides configurations for 16-bit/32-bit timer with module	Code Generator Code Generator Code Generator	1.0.0	
Component     Type     Version       # 8-Bit Timer     Code Generator     1.0.0       # Clock Frequency Accuracy Mea     Code Generator     1.0.0       # Clock Frequency Accuracy Mea     Code Generator     1.0.0       # Clock Frequency Accuracy Mea     Code Generator     1.1.1       # Complementary PWIM Mode Ti     Code Generator     1.1.0       # CRC Calculator     Code Generator     1.0.0       # D/A Converter     Code Generator     1.0.0       # DI/A Converter     Code Generator     1.0.0       # DA Controller     Code Generator     1.0.0       # DAA Controller     Code Generator     1.0.0       # DMA Controller     Code Generator     1.0.0       # Controller     Code Generator     1.0.0       # State State State     1.0.0     Image: State State State       # Sono only last version     Description     Description       This software component provides configurations for 16-bit/32-bit timer with module     Image: State	Code Generator Code Generator Code Generator	1.0.0	
Component     Type     Version       # 8-Bit Timer     Code Generator     1.0.0       # Clock Frequency Accuracy Mea     Code Generator     1.0.0       # Clock Frequency Accuracy Mea     Code Generator     1.0.0       # Compare Match Timer     Code Generator     1.1.1       # Complementary PWM Mode Ti     Code Generator     1.1.0       # CRC Calculator     Code Generator     1.0.0       # DI/A Converter     Code Generator     1.0.0       # Data Operation Circuit     Code Generator     1.0.0       # DMA Controller     Code Generator     1.0.0       # DMA Controller     Code Generator     1.0.0       # DMA Controller     Code Generator     1.0.0       # Show only last version     Description     >       This software component provides configurations for 16-bit/32-bit timer with module     >	Code Generator Code Generator Code Generator	1.0.0	
# 8-Bit Timer       Code Generator       1.0.0         # Clock Frequency Accuracy Mea       Code Generator       1.0.0         # Compare Match Timer       Code Generator       1.0.0         # Compare Match Timer       Code Generator       1.1.1         # Compare Match Timer       Code Generator       1.1.1         # Compare Match Timer       Code Generator       1.0.0         # Complementary PVM Mode Ti       Code Generator       1.0.0         # DA Converter       Code Generator       1.0.0         # Data Operation Circuit       Code Generator       1.0.0         # Data Transfer Controller       Code Generator       1.0.0         # DMA Controller       Code Generator       1.0.0         # DMA Controller       Code Generator       1.0.0         # Sent Link Controller       Code Generator       1.0.0         Show only last version       Description       Description         This software component provides configurations for 16-bit/32-bit timer with module       1600 <td>Code Generator Code Generator Code Generator</td> <td>1.0.0</td> <td></td>	Code Generator Code Generator Code Generator	1.0.0	
Clock Frequency Accuracy Mea     Code Generator     1.0.0     Code Generator     1.1.1     Complementary PWM Mode Ti     Code Generator     1.1.0     Code Generator     1.0.0     Code Generator     1.0.0     DAta Operation Circuit     Code Generator     1.0.0     Data Transfer Controller     Code Generator     1.0.0     DMA Controller     Code Generator     1.0.0     DMA Controller     Code Generator     1.0.0     Code Generator     1.0.	lea Code Generator Code Generator		
Compare Match Timer     Code Generator     1.1.1     Complementary PWIM Mode Ti     Code Generator     1.1.0     Code Generator     1.0.0     DAta CRC Calculator     Code Generator     1.0.0     Data Transfer Controller     Code Generator     1.0.0     DAta Controller     Code Generator     1.0.0     DMA Controller     Code Generator     1.0.0     DMA Controller     Code Generator     1.0.0     DMA Controller     Code Generator     1.0.0     Code Generator     1.0.0     DMA Controller     Code Generator     1.0.0     DMA Controller     Code Generator     1.0.0     Code Generator     1.0.0     To softwom only last version     Description      This software component provides configurations for 16-bit/32-bit timer with module	Code Generator	1.0.0	
Complementary PWM Mode Ti     Code Generator     1.1.0     CRC Calculator     Code Generator     1.0.0     DAA Converter     Code Generator     1.0.0     Data Operation Circuit     Code Generator     1.0.0     DAta Operation Circuit     Code Generator     1.0.0     DAta Controller     Code Generator     1.0.0     DAMA Controller     Code Generator     1.0.0     DMA Controller     Code Generator     1.0.0     Show only last version     Description     This software component provides configurations for 16-bit/32-bit timer with module			
# CRC Calculator       Code Generator       1.0.0         # D/A Converter       Code Generator       1.0.0         # Data Operation Circuit       Code Generator       1.0.0         # Data Transfer Controller       Code Generator       1.0.0         # DMA Controller       Code Generator       1.0.0         # DMA Controller       Code Generator       1.0.0         # Event Link Controller       Code Generator       1.0.0         Core Construct Stable       Code Generator       1.0.0         Show only last version       Description       This software component provides configurations for 16-bit/32-bit timer with module			
D/A Converter     Code Generator     1.0.0     Data Operation Circuit     Code Generator     1.0.0     Data Transfer Controller     Code Generator     1.0.0     DMA Controller     Code Generator     1.0.0     Description     This software component provides configurations for 16-bit/32-bit timer with module			
Data Operation Circuit Code Generator 1.0.0     Data Transfer Controller Code Generator 1.0.0     DMA Controller Code Generator 1.1.0     DMA Controller Code Generator 1.0.0     Courd Controller Code Generator 1.0.0     Show only last version     Description     This software component provides configurations for 16-bit/32-bit timer with module			
DAta Transfer Controller     Code Generator     1.0.0     DMA Controller     Code Generator     1.1.0     DMA Controller     Code Generator     1.0.0     Code Generator     1.0.0     Show only last version     Description     This software component provides configurations for 16-bit/32-bit timer with module			
DMA Controller Code Generator 1.1.0     Event Link Controller Code Generator 1.0.0     Code Generator 1.0.0     Show only last version     Description     This software component provides configurations for 16-bit/32-bit timer with module			
Event Link Controller Code Generator 1.0.0     Code Generator 1.0.0     Show only last version Description This software component provides configurations for 16-bit/32-bit timer with module			
Construction			
Show only last version Description This software component provides configurations for 16-bit/32-bit timer with module			
Description This software component provides configurations for 16-bit/32-bit timer with module	C-1-C	100	>
	Code Generator Code Generator Code Generator Code Generator Code Generator	1.0.0 1.0.0 1.1.0 1.0.0	
Download more software compor	n	Code Generator Code Generator Code Generator Code Generator Code Generator es configurations for 16-bit/32 terrupts at set intervals.	Code Generator 1.0.0 Code Generator 1.1.0 Code Generator 1.0.0 Code Generator 1.0.0 code Generator 1.0.0 es configurations for 16-bit/32-bit timer with m terrupts at set intervals.

In 'Add new configuration for selected component' dialog -> Resource, select 'CMT0' as shown in Figure 4-8 below then click 'Finish'.

e <sup>2</sup> New Component			□ ×
Add new configuration	for selected component		
Compare Match Timer			
Configuration name:	Config_CMT0		
Resource:	CMT0		~
	CMT0		
	CMT1 CMT2		
	CMT3		
	CMTW0 CMTW1		
?	< Back Next >	<u>F</u> inish	Cancel
	S DOCK	Lunsh	Cancel

Figure 4-8 Select Resource - CMT0



In the 'Config\_CMT0' configures CMT0 as shown in **Figure 4-9**. 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.

🐞 *SC_Tutorial.scfg 🛛				
Software component configuration				
Components $\downarrow^{a}_{Z} \boxdot \boxplus \clubsuit$	Configure			
type filter text	Count clock setting PCLK/8 OPCLK/32	○ PCLK/128 (	O PCLK/512	
v	Compare match setting Interval value Register value (CMCOR)	1 7499	ms	<ul> <li>(Actual value: 1.000000)</li> </ul>
Config_CMT0     Middleware	Enable compare match interrupt (CMI0)     Priority	Level 10	<b>~</b>	

Figure 4-9 Config\_CMT0 setting

Click 'Add component' to icon. In 'Software Component Selection' dialog -> Type, select 'Code Generator'. Select 'Compare Match Timer' then click 'Next'. In 'Add new configuration for selected component' dialog -> Resource, select 'CMT1' as shown in **Figure 4-10** below then click 'Finish'.

e <sup>2</sup> New Component		—	o x
Add new configuration	for selected component		
Compare Match Timer			
Configuration name:	Config_CMT1		
Resource:	CMT1		~
?	< Back Next >	<u>F</u> inish	Cancel
	10 Select Res		

Navigate to the 'Config\_CMT1' and configure CMT1 as shown in **Figure 4-11**. This timer is configured to generate a high priority interrupt after 20ms. This timer is used as our short switch de-bounce timer later in this tutorial.

Components $\downarrow^{a}_{\ \ \ }$ $\Box$ $\boxdot$ $$	Configure	
type filter text	Count clock setting O PCLK/8  PCLK/32	○ PCLK/128 ○ PCLK/512
✓	Compare match setting Interval value Register value (CMCOR)	20 (Actual value: 20.00000) 37499
Config_CMT0     Config_CMT1	<ul> <li>Enable compare match interrupt (CMI1)</li> <li>Priority</li> </ul>	Level 10

Figure 4-11 Config\_CMT1 setting



Click 'Add component' <sup>1</sup> icon. In 'Software Component Selection' dialog -> Type, select 'Code Generator'. Select 'Compare Match Timer' then click 'Next'. In 'Add new configuration for selected component' dialog -> Resource, select 'CMT2' as shown in **Figure 4-12** below then click 'Finish'.

e <sup>2</sup> New Component		—	D X
Add new configuration	for selected compon	ent	-
Compare Match Timer Configuration name: Resource:	Config_CMT2		·
(?) < <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	Cancel

Figure 4-12 Select Resource – CMT2

Navigate to the 'Config\_CMT2' and configure CMT2 as shown in **Figure 4-13**. 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.

∰ *SC_Tutorial.scfg ⊠			
Software component configuration	I		
Components 👌 🔁 🕀 🕈 🔻	Configure Count clock setting O PCLK/32	○ PCLK/128  ● PCLK/5	12
type filter text Startup Generic Config_CMT2	Compare match setting Interval value Register value (CMCOR) Compare match interrupt (CMI2) Priority	200 23437	ms v (Actual value: 200.004267)
Config_CMT0	Phoney	Lever to	

Figure 4-13 Config\_CMT2 setting



#### 4.4.3 Interrupt Controller

Referring to the RSK+ schematic, SW1 is connected to IRQ11(P03) and SW2 is connected to IRQ13 (P05). SW3 is connected IRQ15(P07) and the ADTRG0n. Tutorial used ADTRG0n and will be configured later in §4.4.7.

Click 'Add component' <sup>to</sup> icon. In 'Software Component Selection' dialog -> Type, select 'Code Generator'. Select 'Interrupt Controller' as shown in **Figure 4-14** then click 'Next'.

e <sup>2</sup> New Component			□ ×
Software Component Selection Select component from those availabl	e in list		
Function All Type <sup>E</sup> Code Generator			~
Filter			
Component I2C Master Mode I2C Slave Mode	Type Code Generator Code Generator	Version 1.0.0 1.0.0	^
Interrupt Controller Low Power Consumption	Code Generator Code Generator	1.2.0	
H Normal Mode Timer Phase Counting Mode Timer	Code Generator Code Generator	1.0.0 1.2.0	
Port Output Enable     Ports     Programmable Pulse Generator	Code Generator Code Generator Code Generator	1.0.0 1.2.0 1.0.0	
PVM Mode Timer	Code Generator	1.1.0	~
Show only last version Description			
Interrupt Controller configures the in interrupt, NMI pin interrupt and IRQ		ed by ICU: Softwa	re ^
Download more software component	S		
? < <u>B</u> ack	<u>N</u> ext >	Einish	Cancel

Figure 4-14 Select Interrupt Controller

In 'Add new configuration for selected component' dialog -> Resource, select 'ICU' as shown in **Figure 4-15** below then click 'Finish'.

e <sup>2</sup> New Component			□ ×
Add new configuration	for selected component		
Interrupt Controller			
Configuration name:	Config_ICU		
Resource:	(CU		~
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> in	ish	Cancel

Figure 4-15 Select Resource – ICU



Navigate to the 'Config\_ICU', configure these two interrupts as falling edge triggered as shown in **Figure 4-16** below.

component configuration								
ts ↓ª 🕞 🕀 井 🔻	Configure							
ت ن	Software interrupt setting							
r text	Software interrupt	Priority	Level 15 (highest)	$\sim$				
tartup	Software interrupt 2	Priority	Level 15 (highest)	$\sim$				
→ Generic	NMI pin interrupt setting							
Drivers	NMI pin interrupt	Detection type	Falling edge	$\sim$	Digital filter	No filter $\sim$ 0	(MH <i>z</i> )	
Interrupt Config_ICU	IRQ0 setting							
Timers	IRQ0	Detection type	Low level	$\sim$	Digital filter	No filter 🗸 0	(MHz)	
Config_CMT2		Priority	Level 15 (highest)	$\sim$				
Config_CMT1	IRQ1 setting							
dleware lication		Detection type	Low level	$\sim$	Digital filter	No filter 🗸 0	(MHz)	
			Level 15 (highest)					
		Phoney	Level 15 (highest)					
	IRQ2 setting	-		_				
	IRQ2	Detection type		~	Digital filter	No filter $\lor$ 0	(MHz)	
		Priority	Level 15 (highest)	$\sim$				
	IRQ3 setting							
	IRQ3	Detection type	Low level	$\sim$	Digital filter	No filter \vee 0	(MHz)	
		Priority	Level 15 (highest)	$\sim$				
	IRQ4 setting							
	IRQ4	Detection type	Low level	$\sim$	Digital filter	No filter $\sim$ 0	(MHz)	
		Priority	Level 15 (highest)	$\sim$				
	IBO5 setting							
	IRQ5 setting	Detection type	Low level	$\sim$	Digital filter	No filter 🗸 0	(MHz)	
			Level 15 (highest)		rear (1000)		,	
		Priority	Level 15 (highest)	- V				
	IRQ6 setting					AL (1)		
	IRQ6	Detection type		$\sim$	Digital filter	No filter $\lor$ 0	(MHz)	
		Priority	Level 15 (highest)	$\sim$				
	IRQ7 setting							
	IRQ7	Detection type	Low level	$\sim$	Digital filter	No filter $\lor$ 0	(MHz)	
		Priority	Level 15 (highest)	$\sim$				
	IRQ8 setting							
	IRQ8	Detection type	Low level	$\sim$	Digital filter	No filter \vee 0	(MHz)	
		Priority	Level 15 (highest)	$\sim$				
	IRO9 cetting							
	IRQ9 setting	Detection type	Low level	~	Digital filter	No filter 🗸 0	(MHz)	
					angreat three		(11112)	
		Priority	Level 15 (highest)					
	IRQ10 setting	_						
	IRQ10	Detection type		$\sim$	Digital filter	No filter $\lor$ 0	(MHz)	
		Priority	Level 15 (highest)	$\sim$				
	IRQ11 setting							
	✓ IRQ11	Detection type	Falling edge	~	Digital filter	No filter 🗸 0	(MHz)	
		Priority	Level 15 (highest)	~				
	IRQ12 setting							
	IRQ12	Detection type	Low level	$\sim$	Digital filter	No filter \vee 🛛 0	(MHz)	
			Level 15 (highest)	$\sim$				
	19012							
	IRQ13 setting	Detection type	Falling od		Digital file	No filter 🗸 0	(MHz)	
	IRQ13			~	Digital filter	No litter V	(ivirtz)	
		Priority	Level 15 (highest)	~				
	IRQ14 setting							
	IRQ14	Detection type	Low level	~	Digital filter	No filter $\sim$ 0	(MHz)	
		Priority	Level 15 (highest)	$\sim$				
	IRQ15 setting							
	IRQ15	Detection type	Low level	$\sim$	Digital filter	No filter \vee 0	(MHz)	
			Level 15 (highest)	~				

Figure 4-16 Config\_ICU setting



#### 4.4.4 Ports

Referring to the RSK+ schematic, LED0 is connected to P73, LED1 is connected to PG7, LED2 is connected to PG6 and LED3 is connected to PG5. PJ3 is used as one of the LCD control lines, together with PF5, PG3 and PG4.

Click 'Add component' <sup>to</sup> icon. In 'Software Component Selection' dialog -> Type, select 'Code Generator'. Select 'Ports' as shown in **Figure 4-17** then click 'Next'.

e <sup>2</sup> New Component						
oftware Component Selection Select component from those available in list						
Function All			~			
Type Code Generator			~			
Filter						
Component	Туре	Version	*			
H Low Power Consumption	Code Generator	1.1.0				
H Normal Mode Timer	Code Generator	1.0.0				
H Phase Counting Mode Timer	Code Generator	1.2.0				
H Port Output Enable	Code Generator	1.0.0				
# Ports	Code Generator	1.2.0				
Programmable Pulse Generator	Code Generator	1.0.0				
PWM Mode Timer	Code Generator	1.1.0				
Heal Time Clock	Code Generator	1.0.0				
H SCI/SCIF Asynchronous Mode	Code Generator	1.0.0	~			
<			>			
Show only last version Description						
This software component provides or Common features such as reading, w can be configured. Enabling features ups are also supported.	riting, and setting the dir	ection of ports	and pins			
Download more software components	5					
? < <u>B</u> ack	<u>N</u> ext > <u>F</u>	inish	Cancel			

Figure 4-17 Select Ports

In 'Add new configuration for selected component' dialog -> Resource, select 'PORT' as shown in **Figure 4-18** below then click 'Finish'.

e <sup>2</sup> New Component		
Add new configuratior	for selected component	
Ports		
Configuration name:	Config_PORT	
Resource:	PORT	~
?	< <u>Back</u> <u>N</u> ext > <u>Finish</u>	Cancel

Figure 4-18 Select Resource – PORT



#### 'PORT7', 'PORT<u>F'</u>, 'PORTG', 'PORTJ' tick box is checked as shown in **Figure 4-19** below.

🏟 *SC_Tutorial.scfg 🙁			
Software component configuration	ı		
Components $\downarrow^a_Z \models \oplus \Rightarrow$	Configure		
10 T	Port selection	PORT7 PORTF PORTG PO	ORTJ
type filter text			
<ul> <li>✓</li></ul>	DORT0	DORT1	
<ul> <li>✓ r_bsp</li> <li>✓ ➢ Drivers</li> </ul>	PORT2	PORT3	
✓	DORT4	PORT5	
✓ ▷ I/O Ports Config_PORT	DORT6	PORT7	
✓	PORT8	PORT9	
Config_CMT2	DORTA	PORTB	
Middleware Application			
	PORTG	PORTJ	

Figure 4-19 Select Port selection

Navigate to the 'Ports' configure these four I/O lines and LCD control lines as shown in **Figure 4-20**, **Figure 4-21**, **Figure 4-22** and **Figure 4-23** below. Ensure that the 'Output 1' tick box is checked, except PG3. Select 'PORT7' tab.

Interview SC_Tutorial.scfg ≥		- 8
Software component configuration		🐻 🖨
Components la 🖂 🕀 🔹 🕻	Port selection PORT7 PORTF PORTG PORTJ	^
type filter text		
✓ ➢ Startup ✓ ➢ Generic	Apply to all     Unused In Out Pull-up CMOS output      Output 1	
<ul> <li>✓ r_bsp</li> <li>✓ ➢ Drivers</li> <li>✓ ➢ Interrupt</li> </ul>	P70 Output 1 High-drive output	
<ul> <li>✓ Config_ICU</li> <li>✓ (&gt;&gt; I/O Ports</li> <li>✓ Config_PORT</li> </ul>		
Config_CMT1	Unused O In Out Pull-up CMOS output Output 1	
Config_CMT0	P72  Unused O In O Out Pull-up CMOS output Output 1 Normal drive output	~
🧀 Application	P73 ○ Unused ○ In Out □ Pull-up CMOS output ✓ ☑Output 1 High-drive output	~
	P74 ● Unused ○ In ○ Out □ Pull-up CMOS output ∨ □ Output 1 Normal drive output	~
	P75 © Unused O In O Out Pull-up CMOS output Output 1 Normal drive output	~
	P76 © Unused O In O Out Pull-up CMOS output Output 1 Normal drive output	~
	P77 ● Unused ◯ In ◯ Out □ Pull-up CMOS output ─ □ Output 1 Normal drive output	

Figure 4-20 Select PORT7 tab



#### Select 'PORTF' tab.

🗱 *SC_Tutorial.scfg 🙁		- 8
Software component configuration		🐻 🖆
Components $\downarrow^{a}_{\mathbb{Z}} \models \boxplus \clubsuit$	Configure	^
tu tu	Port selection PORT7 PORTF PORTG PORTJ	
type filter text	Apply to all	
✓ ➢ Startup ✓ ➢ Generic ✓ ♂ r_bsp	Output 1	
✓	PF0 <ul> <li>Unused</li> <li>In</li> <li>Out</li> <li>Pull-up</li> <li>CMOS output</li> <li>Output 1</li> </ul>	
✓	PF1     Out Pull-up CMOS output Output 1	
Config_CMT2 Config_CMT0 Config_CMT0	PF2         Image: Output Out	
➢ Application	PF3	
	PF4     Out Out Out Out Out Out Out Out Output	
	PF5 ○ Unused ○ In	

#### Figure 4-21 Select PORTF tab

#### Select 'PORTG' tab.

*SC_Tutorial.scfg 🔀		
oftware component configuration		۵ 💽
Components 🕴 📮 🖬 🔹	Configure	
5.5	Port selection PORT7 PORTF PORTG PORTJ	
type filter text		
🗸 🧁 Startup	Apply to all	
✓ → Generic → r bsp	Unused O In Out Pull-up CMOS output Output Normal drive output	$\sim$
✓ ➢ Drivers	PG0	
✓ ➢ Interrupt ♂ Config_ICU	● Unused         O In         O Out         Pull-up         CMOS output         □ Output 1         Normal drive output	~
✓ → I/O Ports ✓ Config_PORT	_ PG1	
Config_Point     Config_CMT1	● Unused         O In         O Out         Pull-up         CMOS output         □ Output 1         Normal drive output	~
Config_CMT2	_ PG2	
Config_CMT0	● Unused         O In         O Out         Pull-up         CMOS output         □ Output 1         High-drive output	$\sim$
🔁 Application	PG3	
	O Unused O In Out □ Pull-up CMOS output ✓ □ Output 1 High-drive output	~
	PG4	
	O Unused O In Out □ Pull-up CMOS output ✓ Output 1 High-drive output	~
	PG5	
	O Unused O In  O Out □ Pull-up CMOS output ✓ ☑ Output 1 High-drive output	~
	~ PG6	
	O Unused O In Out □ Pull-up CMOS output ✓ Output 1 High-drive output	~
	_ PG7	
	O Unused O In Out □ Pull-up CMOS output ✓ ☑ Output 1 High-drive output	~

Figure 4-22 Select PORTG tab

#### Select 'PORTJ' tab.

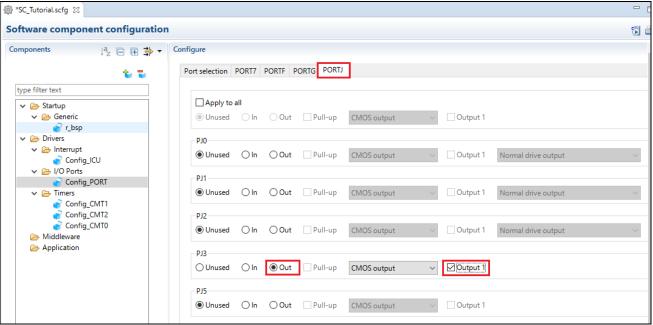


Figure 4-23 Select PORTJ tab



#### 4.4.5 SCI/SCIF Asynchronous Mode

In the RSK+RX65N-2MB SCI8 is connected via a Renesas RL78/G1C to provide a USB virtual COM port as shown in the schematic.

Click 'Add component' is icon. In 'Software Component Selection' dialog -> Type, select 'Code Generator'. Select 'SCI/SCIF Asynchronous Mode' as shown in **Figure 4-24** then click 'Next'.

e <sup>2</sup> New Component			
oftware Component Selection	- 10 A		
Select component from those available	e in list		
Function All			$\sim$
Type Code Generator			~
Filter			
riter			
Component	Туре	Version	^
PWM Mode Timer	Code Generator	1.1.0	
Real Time Clock	Code Generator	1.0.0	
H SCI/SCIF Asynchronous Mode	Code Generator	1.0.0	
SCI/SCIF Clock Synchronous M	Code Generator	1.0.0	
H Single Scan Mode S12AD	Code Generator	1.1.0	
SPI Clock Synchronous Mode	Code Generator	1.0.0	
H SPI Operation Mode	Code Generator	1.0.0	
Voltage Detection Circuit	Code Generator	100	×
			-
Show only last version			
Description			
This software component provides co processor) asynchronous mode.	onfigurations for SC	l(SCIF) single(multi-	· ^
processor) asynchronous mode.			
			~
Download more software components			
(?) < Back	Next >	Finish	Cancel
- Juck	<u></u> exc	<u>-</u>	2011CC

Figure 4-24 Select SCI/SCIF Asynchronous Mode

In 'Add new configuration for selected component' dialog -> Work mode, select 'Transmission/Reception' as shown in **Figure 4-25** below.

e <sup>2</sup> New Component	_		×
Add new configuration	for selected component		
SCI/SCIF Asynchronous	Mode		
Configuration name:	Config_SCI0		
Work mode:	Multi-processor Reception		~
Resource:	Multi-processor Reception Multi-processor Transmission Multi-processor Transmission/Reception Reception Transmission Transmission/Reception		
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Can	cel

Figure 4-25 Select Work mode – Transmission/Reception



#### In 'Resource', select 'SCI8' as shown in **Figure 4-26** below.

e <sup>2</sup> New Component			×
Add new configuration	for selected component	ł	
SCI/SCIF Asynchronous	Node		
Configuration name:	Config_SCI0		
Work mode:	Transmission/Reception		$\sim$
Resource:	SCI8	 	$\sim$
	SCI0           SC11           SC110           SC111           SC12           SC12           SC13           SC14           SC15           SC16           SC17           SC18           SC19		
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Cance	el

Figure 4-26 Select Resource – SCI8

Ensure that the 'Configuration name' updates to 'Config\_SCI8' as shown in **Figure 4-27** below then click 'Finish'

e <sup>2</sup> New Component						×
Add new configuration	n for selecto	ed componen	ıt		ł	
SCI/SCIF Asynchronous	Mode					
Configuration name:	Config_9	SCI8				
Work mode:	Transmi	ssion/Receptior	n			$\sim$
Resource:	ISCI8					$\sim$
				_		

Figure 4-27 Ensure Configuration name - Config\_SCI8



Configure SCI8 as shown in **Figure 4-28**. Ensure the 'Start bit edge detection' is set as 'Falling edge on RXD8 pin' and the 'Bit rate' is set to 19200 bps. All other settings remain at their defaults.

∰ *SC_Tutorial.scfg 🔀			
Software component configuration			
Components	Configure Start bit edge detection setting		
type filter text	O Low level on RXD8 pin	Falling edge on RXD8 pin	
ype mer text ✓ ➢ Startup ✓ ➢ Generic @ r_bsp ✓ ➢ Drivers	Data length setting	● 8 bits	○ 7 bits
<ul> <li>✓ (⇒ Interrupt</li> <li>✓ Config_ICU</li> <li>✓ (&gt; I/O Ports</li> <li>✓ Config_PORT</li> </ul>	Stop bit length setting	O 2 bits	
Communications	Transfer direction setting	⊖ MSB-first	
Config_CMT1 Config_CMT2 Config_CMT0 Confi	Transfer rate setting Transfer clock Base clock Bit rate	Internal clock 16 cycles for 1-bit period 19200	<ul> <li>(bps) (Actual value: 19230.769, Error: 0.160%)</li> </ul>
	Enable modulation duty correction SCK8 pin function	SCK is not used	~
	Noise filter setting Enable noise filter Noise filter clock	Clock signal divided by 1	<ul> <li>60000000 (Hz)</li> </ul>
	Hardware flow control setting None	⊖ CTS8#	O RTS8#
	Data handling setting Transmit data handling Receive data handling	Data handled in interrupt service routine Data handled in interrupt service routine	<ul><li>∨</li><li>∨</li></ul>
	Interrupt setting TXI8 priority RXI8 priority	Level 15 (highest) Level 15 (highest)	v v
	Enable reception error interrupt (ERI8) TEI8, ERI8 priority (Group BL1)	Level 15 (highest)	~
	Callback function setting ☑ Transmission end	Reception end	Reception error

Figure 4-28 Config\_SCI8 setting



#### 4.4.6 SPI Clock Synchronous Mode

In the RSK+RX65N-2MB SCI6 is used as an SPI master for the Pmod LCD on the PMOD1 connector as

shown in the schematic. Click 'Add component' <sup>1</sup>/<sub>1</sub> icon. In 'Software Component Selection' dialog -> Type, select 'Code Generator'. Select 'SPI Clock Synchronous Mode' as shown in **Figure 4-29** then click 'Next'.

Mew Component		— L	×
oftware Component Selection			
Select component from those available	e in list		
Function All			~
Type Code Generator			
Type Code Generator			Ň
Filter			
^	_		
Component	Туре	Version	^
Programmable Pulse Generator	Code Generator	1.0.0	
PWM Mode Timer	Code Generator	1.1.0	
🖶 Real Time Clock	Code Generator	1.0.0	
H SCI/SCIF Asynchronous Mode	Code Generator	1.0.0	
H SCI/SCIF Clock Synchronous M	Code Generator	1.0.0	
🖶 Single Scan Mode S12AD	Code Generator	1.1.0	_
H SPI Clock Synchronous Mode	Code Generator	1.0.0	
🖶 SPI Operation Mode	Code Generator	1.0.0	
Voltage Detection Circuit	Code Generator	1.0.0	~
<			>
Show only last version			
Description			
This component provides clock synch	hronous operation of RSPI of	or SCI (Simple SPI	~
bus). It includes 4 transfer modes: Sla			
transmit/receive and Master transmit			
			~
Download more software components			
? < Back	Next > Fini	sh Ca	ncel
. Buck			

Figure 4-29 Select SPI Clock Synchronous Mode

Ensure Operation, select 'Master transmit only' as shown in Figure 4-30 below.

d new configuration	n for selected component		
SPI Clock Synchronous	Mode		
Configuration name:	Config_RSPI0		
Operation:	Master transmit only		~
Resource:	Slave transmit/receive Slave transmit only Master transmit/receive		
	Master transmit only		
۰	< Back Next >	Finish	Cancel

Figure 4-30 Select Operation – Master transmit only



#### In 'Resource', select 'SCI6' as shown in **Figure 4-31** below.

e <sup>2</sup> New Component			×
Add new configuration	for selected component		
- SPI Clock Synchronous N	lode		
Configuration name:	Config_RSPI0		
Operation:	Master transmit only		$\sim$
Resource:	RSPIO		$\sim$
	RSPI0           RSPI1           RSP12           SCI0           SCI1           SCI10           SCI11           SCI12           SCI2           SCI3           SCI4           SCI5           SCI6		
?	SCI8 < <u>B</u> SCI9		

Figure 4-31 Select Resource – SCI6

Ensure that the 'Configuration name' updates to 'Config\_SCI6' as shown in **Figure 4-32** below then click 'Finish'

SPI Clock Synchronous I Configuration name:	Config_SCI6	
Operation:	Master transmit only	~
Resource:	SCI6	~

Figure 4-32 Ensure Configuration name - Config\_SCI6

Configure SCI6 as shown in **Figure 4-33**. Ensure the 'Transfer direction' is set as 'MSB-first' and the 'Bit rate' is set to 15000 kbps. All other settings remain at their defaults.

∰ *SC_Tutorial.scfg ⊠		
Software component configuration		
Components	Configure	
Components ↓ <sup>2</sup> ↓ <sup>2</sup>	Configure Transfer direction setting LSB-first Data inversion setting Normal Transfer speed setting Transfer clock Bit rate Enable modulation duty correction Clock setting Transmit data handling Interrupt setting TXI6 priority TEl6 priority (Group BL0) Callback function setting	MSB-first   Inverted   Internal clock (SCK6 pin functions as clock output pin)   15000   (kbps)   (Actual value: 15000, Error: 0%)     Data handled in interrupt service routine   Level 15 (highest)   Level 15 (highest)
	Transmission end	

Figure 4-33 Config\_SCI6 setting



#### 4.4.7 Single Scan Mode S12AD

We will be using the S12AD on 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. Click

'Add component' icon. In 'Software Component Selection' dialog -> Type, select 'Code Generator'. Select 'Single Scan Mode S12AD' as shown in **Figure 4-34** then click 'Next'.

Software Component Selection Select component from those available in list  Function All Type Code Generator Filter  Component Type Version Programmable Pulse Generator PWM Mode Timer Code Generator 1.0.0 FReal Time Clock Code Generator 1.0.0 SCI/SCIF Asynchronous Mode Code Generator 1.0.0 SSI/SCIF Clock Synchronous M Code Generator 1.0.0 SOURCE Clock Synchronous Mode Code Generator 1.0.0 SOURCE Clock Synchron SOURCE Clock Synchronous Mode Code Generator Cod	~ ~
Function       All         Type       Code Generator         Filter       Component       Type         Version       Code Generator       1.0.0         P Programmable Pulse Generator       Code Generator       1.0.0         PWM Mode Timer       Code Generator       1.0.0         Real Time Clock       Code Generator       1.0.0         SCI/SCIF Asynchronous Mode       Code Generator       1.0.0         Single Scan Mode S12AD       Code Generator       1.0.0         SIPI Clock Synchronous Mode       Code Generator       1.0.0         SPI Operation Mode       Code Generator       1.0.0         SPI Operation Mode       Code Generator       1.0.0         SPI Show only last version       Description       Description         This software component provides single scan mode configurations for 12-Bit A Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	
Type Code Generator Filter Component Type Version Programmable Pulse Generator PWM Mode Timer Code Generator 1.0.0 Code Generator 1.1.0 Real Time Clock Code Generator 1.0.0 SCI/SCIF Asynchronous Mode Code Generator 1.0.0 Single Scan Mode S12AD Code Generator 1.0.0 Single Scan Mode S12AD Code Generator 1.0.0 SPI Clock Synchronous Mode Code Generator 1.0.0 SPI Clock Synchronous Mode Code Generator 1.0.0 SPI Operation Mode SPI Operation Mode Code Generator 1.0.0 SPI Operation Mode Cop	~
Type Code Generator Filter Component Type Version Programmable Pulse Generator PWM Mode Timer Real Time Clock Code Generator 1.0.0 SolyClF Clock Synchronous Mode Code Generator 1.0.0 Single Scan Mode S12AD Code Generator 1.0.0 SolyClF Clock Synchronous Mode Code Generator 1.0.0 This software component provides single scan mode configurations for 12-Bit A Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	~
Filter       Type       Version         Programmable Pulse Generator       Code Generator       1.0.0         PVMM Mode Timer       Code Generator       1.1.0         Real Time Clock       Code Generator       1.0.0         Sci/SCIF Asynchronous Mode       Code Generator       1.0.0         Sigle Scan Mode S12AD       Code Generator       1.0.0         Spl Clock Synchronous Mode       Code Generator       1.0.0         Spl Clock Synchronous Mode       Code Generator       1.0.0         Spl Operation Mode       Code Generator       1.0.0         Spl Operation Mode       Code Generator       1.0.0         Show only last version       Description       Description         This software component provides single scan mode configurations for 12-Bit A Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	~
Filter       Type       Version         Programmable Pulse Generator       Code Generator       1.0.0         PVM Mode Timer       Code Generator       1.1.0         Real Time Clock       Code Generator       1.0.0         SCI/SCIF Asynchronous Mode       Code Generator       1.0.0         Sigle Scan Mode S12AD       Code Generator       1.0.0         SPI Clock Synchronous Mode       Code Generator       1.0.0         SPI Clock Synchronous Mode       Code Generator       1.0.0         SPI Operation Mode       Code Generator       1.0.0         SPI Operation Mode       Code Generator       1.0.0         Show only last version       Description       Description         This software component provides single scan mode configurations for 12-Bit A Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	
Component       Type       Version         Programmable Pulse Generator       Code Generator       1.0.0         PWM Mode Timer       Code Generator       1.1.0         Real Time Clock       Code Generator       1.0.0         SCI/SCIF Asynchronous Mode       Code Generator       1.0.0         SCI/SCIF Clock Synchronous M       Code Generator       1.0.0         Single Scan Mode S12AD       Code Generator       1.0.0         SPI Clock Synchronous Mode       Code Generator       1.0.0         SPI Operation Mode       Code Generator       1.0.0         Show only last version       Description       This software component provides single scan mode configurations for 12-Bit A Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	
Component     Type     Version       Programmable Pulse Generator     Code Generator     1.0.0       PWM Mode Timer     Code Generator     1.1.0       Real Time Clock     Code Generator     1.0.0       SCI/SCIF Asynchronous Mode     Code Generator     1.0.0       SCI/SCIF Clock Synchronous M     Code Generator     1.0.0       Single Scan Mode S12AD     Code Generator     1.0.0       SPI Clock Synchronous Mode     Code Generator     1.0.0       SPI Clock Synchronous Mode     Code Generator     1.0.0       SPI Clock Synchronous Mode     Code Generator     1.0.0       Soft Single Scan Mode S12AD     Code Generator     1.0.0       Show only last version     Description     Description       This software component provides single scan mode configurations for 12-Bit A Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	
Programmable Pulse Generator     PWM Mode Timer     Code Generator     PWM Mode Timer     Code Generator     Code Generator     1.0.0     Real Time Clock     Code Generator     1.0.0     SCI/SCIF Asynchronous Mode     SCI/SCIF Clock Synchronous M     Code Generator     1.0.0     SSI Clock Synchronous Mode     Sol Generator     1.0.0     Sol Sol Clock Synchronous Mode     Sol Generator     1.0.0     Sol Clock Synchronous Mode     Sol Generator     1.0.0     Sol Generator     1.0.0     Sol Clock Synchronous Mode     Sol Generator     1.0.0     Sol Clock Synchronous Mode     Sol Generator     1.0.0     Sol Clock Synchronous Mode     Sol Generator     1.0.0     Sol Generator     Sol G	^
PWM Mode Timer     Code Generator     1.1.0     Real Time Clock     Code Generator     1.0.0     SCI/SCIF Asynchronous Mode     Code Generator     1.0.0     SCI/SCIF Clock Synchronous M     Code Generator     1.0.0     SPI Clock Synchronous Mode     Code Generator     1.0.0     SPI Operation Mode     Code Generator     1.0.0     Secription This software component provides single scan mode configurations for 12-Bit A     Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	
Real Time Clock       Code Generator       1.0.0         SCI/SCIF Asynchronous Mode       Code Generator       1.0.0         SCI/SCIF Clock Synchronous M       Code Generator       1.0.0         Single Scan Mode S12AD       Code Generator       1.0.0         SPI Clock Synchronous Mode       Code Generator       1.0.0         SPI Clock Synchronous Mode       Code Generator       1.0.0         SPI Operation Mode       Code Generator       1.0.0         Show only last version       Description       Secription         This software component provides single scan mode configurations for 12-Bit A Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	
SCI/SCIF Asynchronous Mode       Code Generator       1.0.0         SCI/SCIF Clock Synchronous M       Code Generator       1.0.0         Single Scan Mode S12AD       Code Generator       1.0.0         SPI Clock Synchronous Mode       Code Generator       1.0.0         SPI Clock Synchronous Mode       Code Generator       1.0.0         SPI Operation Mode       Code Generator       1.0.0         Solve only last version       Description       Software component provides single scan mode configurations for 12-Bit A Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	
SCI/SCIF Clock Synchronous M       Code Generator       1.0.0         Single Scan Mode S12AD       Code Generator       1.1.0         SPI Clock Synchronous Mode       Code Generator       1.0.0         SPI Operation Mode       Code Generator       1.0.0         Solution       Solution       Code Generator       1.0.0         Show only last version       Description       Software component provides single scan mode configurations for 12-Bit A Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	
Image: Scan Mode S12AD       Code Generator       1.1.0         Image: SPI Clock Synchronous Mode       Code Generator       1.0.0         Image: SPI Operation Mode       Code Generator       1.0.0         Image: SPI O	
Show only last version Description This software component provides single scan mode configurations for 12-Bit A Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	
Show only last version Description This software component provides single scan mode configurations for 12-Bit A Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	
Description This software component provides single scan mode configurations for 12-Bit A Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	1
This software component provides single scan mode configurations for 12-Bit A Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	
Converter which the analog inputs of up to 8 (unit 0) and 21 (unit 1) channels	
	/D ^
arbitrarily selected are converted for only once in ascending channel order.	
	~
Download more software components	
(?) < <u>Back</u> <u>Next</u> > <u>Finish</u>	

Figure 4-34 Select Single Scan Mode S12AD

In 'Add new configuration for selected component' dialog -> Resource, select 'S12AD0' as shown in **Figure 4-35** below then click 'Finish'.

e <sup>2</sup> New Component				×
Add new configuratior	for selected component		-	88
Single Scan Mode S12AI	)			
Configuration name:	Config_S12AD0			
Resource:	S12AD0			$\sim$
	S12AD0 S12AD1			
	JIZADI			_
0	< <u>B</u> ack <u>N</u> ext > <u>F</u> i	nish	Cance	el

Figure 4-35 Select Resource – S12AD0



Configure S12AD0 as shown in **Figure 4-36** and **Figure 4-37**. Ensure the 'Analog input channel' tick box for AN000 is checked and the 'Start trigger source' is set to 'A/D conversion start trigger pin'. All other settings remain at their defaults.

∰ *SC_Tutorial.scfg ⊠		
Software component configuration		
Components $\downarrow^a_Z \models \blacksquare \rightrightarrows \checkmark$ Con	figure	
	Basic setting	
type filter text	Analog input mode setting Double trigger mode	
✓ ➢ Startup ✓ ➢ Generic ✓ ☑ r_bsp	Analog input channel setting	AN002 AN003 AN004
<ul> <li>✓ (⇒) Drivers</li> <li>✓ (⇒) Interrupt</li> <li>✓ Config_ICU</li> </ul>	AN005 AN006	
Config_PORT     Config_PORT     Communications	Conversion start trigger setting Start trigger source	
Config_SCI6	A/D conversion start trigger pin	×
Config_SCI8 Converter Config_S12AD0	Interrupt setting	
<ul> <li>Timers</li> </ul>	Enable AD conversion end interru	pt (S12ADI) Priority Level 15 (highest) V
Config_CMT1	<ul> <li>Advance setting</li> </ul>	
🧉 Config_CMT0 🍋 Middleware	Add/Average AD value setting	
Application	AN000 AN001 AN001 AN005	AN002 AN003 AN004
	Self diagnosis setting	
	Mode	Unused ~
	Voltage used	ov 🗸
	Disconnection detection assist settin	9
	Charge setting	Unused ~
	Period	2 ADCLK 🗸
	Dedicated sample hold circuit chann	
	AN000	AN002
	Data registers setting	
	Data placement	Right-alignment V
	Automatic clearing Conversion resolution	Disable automatic cleaning v 12-bit accuracy v
	Addition/Average mode select	Addition mode
	Addition count	1-time V
	Window function setting	
	Oisable	○ Enable

Figure 4-36 Config\_S12AD0 setting (1)



#### RSK+RX65N-2MB

#### 4. Smart Configurator Using the e2 studio plug-in

Window A/B operation setting			
Enable comparison window A	Enable comparison window B		
Window A/B complex condition	Window A comparison condition matched OR	window B comparison of	condition matched $\sim$
A/D comparison A setting			
Reference data 0 for comparison	0		
Reference data 1 for comparison	0		
Use comparator for AN000	Reference data 0 > A/D-conve	erted value	$\sim$
Use comparator for AN001	Reference data 0 > A/D-conve	erted value	$\sim$
Use comparator for AN002	Reference data 0 > A/D-conve	erted value	$\sim$
Use comparator for AN003	Reference data 0 > A/D-conve	erted value	$\sim$
Use comparator for AN004	Reference data 0 > A/D-conve	erted value	$\sim$
Use comparator for AN005	Reference data 0 > A/D-conve	erted value	$\sim$
Use comparator for AN006	Reference data 0 > A/D-conve	erted value	$\sim$
Use comparator for AN007	Reference data 0 > A/D-conve	erted value	$\sim$
A/D comparison B setting			
Reference data 0 for comparison	0		
Reference data 1 for comparison	0		
Comparison B channel	Unused		$\sim$
	Reference data 0 > A/D-conve	erted value	$\sim$
Input sampling time setting			
Dedicated sample and hold circuit	4	(us) (Actual v	value: 4.0)
AN000/Self-diagnosis	0.183	(us) (Actual v	value: 0.183)
AN001	0.183	(us) (Actual v	value: 0.183)
AN002	0.183	(us) (Actual v	value: 0.183)
AN003	0.183	(us) (Actual v	value: 0.183)
AN004	0.183	(us) (Actual v	value: 0.183)
AN005	0.183	(us) (Actual v	value: 0.183)
AN006	0.183	(us) (Actual v	value: 0.183)
AN007	0.183		value: 0.183)
	(Total conversion time: 0.767us	s)	
Interrupt setting			
Enable AD conversion compare		ion compare interrupt B	(S12CMPBI)
Group BL1 priority	Level 15 (highest) 🛛 🗸	r	

Figure 4-37 Config\_S12AD0 setting (2)



### 4.5 Pins configuration page

Smart Configurator assigns pins to the software components that are added to the project. Assignment of the pins can be changed using the Pins page.

🐡 *SC_Tutorial.scfg 🛛
Pins Configuration
This configuration
Hardware Resource
Type filter text
👗 All
🗱 Clock generator
🦓 Clock frequency accuracy measurement circu
* 🛱 Buses
EXDMA controller
Interrupt controller unit
> Multi-function timer pulse unit 3
Port output enable 3
16-bit timer pulse unit
TPU0
C TPU1
C TPU2
TPU3
<ul> <li>TPU4</li> <li>TPU5</li> </ul>
<ul> <li>IPUS</li> <li>Programmable pulse generator</li> </ul>
PPG0
PPG0
v (3. 8-bit timer
< >>
Pie Numbra
Pin Function Pin Number
Overview Board Clocks Component Pins Interrupt

Figure 4-38 Pin configuration page

#### 4.5.1 Change pin assignment of a software component

To change the pin assignment of a software component in the Pin Function list, click to change view to show by Software Components.

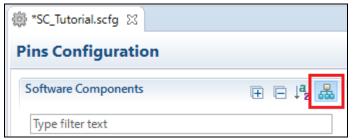


Figure 4-39 Change view to show by Software Components



Select the Config\_ICU of Software Components. In the Pin Function list -> Assignment column, change the pin assignment IRQ13 to P05, IRQ11 to P03. Ensure the 'Enable' tick box of IRQ11 and IRQ13 are checked, as shown in **Figure 4-40**.

oftware Components 🛛 🕀 🕞 📲 🚜	Pin Function	1				- 🤣 📑 🖬 🗠 🛛
ype filter text	Type pin fu	unction				
🗸 🚣 r_bsp	Enabled	Function	Assignment	Pin Number	Direction	Remarks
📦 r_bsp		IRQ0	Not assigned	Not assigned	None	
🗸 🚣 Compare Match Timer		IRQ1	Not assigned	Not assigned	None	
Config_CMT0		IRQ2	Not assigned	Not assigned	None	
Config_CMT1		IRQ3	Not assigned	Not assigned	None	
Config_CMT2		IRQ4	Not assigned	Not assigned	None	
Interrupt Controller     Config_ICU     Ports     Config_PORT     Config_PORT		IRQ5	Not assigned	Not assigned	None	
		IRQ6	Not assigned	Not assigned	None	
		IRQ7	Not assigned	Not assigned	None	
		IRQ8	Not assigned	Not assigned	None	
SCI/SCIF Asynchronous Mode		IRQ9	Not assigned	Not assigned	None	
🚽 矿 Config_SCl8		IRQ10	Not assigned	Not assigned	None	
SPI Clock Synchronous Mode		IRQ11	P03/IRQ11/DA0	4	1	
<ul> <li>Config_SCI6</li> <li>Single Scan Mode S12AD</li> <li>Config_S12AD0</li> </ul>		IRQ12	Not assigned	Not assigned	None	
		IRQ13	P05/IRQ13/DA1	2	1	
		IRQ14	Not assigned	Not assigned	None	
		IRQ15	Not assigned	Not assigned	None	
		NMI	Not assigned	Not assigned	None	
	<					3

Figure 4-40 Configure pin assignment - Config\_ICU

Select the Config\_SCI8 of Software Components. In the Pin Function list -> Assignment column, Ensure the 'Enable' tick box of RXD8 and TXD8 are checked and Assignment column of RXD8 is PJ1 and TXD8 is PJ2 as shown in **Figure 4-41**.

oftware Components 🛛 🕀 🕞 🕌	Pin Functior	1				- 🔁 🔚 🛅 🖻
ype filter text	Type pin fu	inction				
<ul> <li>         r_bsp         r_bsp     </li> <li>         Compare Match Timer         Config_CMT0         Config_CMT1         Config_CMT2         Interrupt Controller         Config_ICU         Solver Science         Config_PORT         Sci/SCIF Asynchronous Mode         Config_SCI8         SPI Clock Synchronous Mode         Config_SCI6         Single Scan Mode S12AD         Config_S12AD0         </li> </ul>	Enabled	Function CTS8# RTS8# RXD8 SCK8 TXD8	Assignment Not assigned PJ1/MTIOC6A/RXD8/SMISO8/S Not assigned PJ2/TXD8/SMOSI8/SSDA8/SSLC	Not assigned	Direction None I None O	Remarks

Figure 4-41 Configure pin assignment - Config\_SCI8



Select the Config\_SCI6 of Software Components. In the Pin Function list -> Assignment column, Ensure the 'Enable' tick box of SCK6 and SMOSI6 are checked and Assignment column of SCK6 is P02, SMOSI6 is P00 as shown in **Figure 4-42**.

SC_Tutorial.scfg 🔀						
Software Components $\square \square \downarrow^a_{\mathbf{Z}}$	Pin Function	n				🤣 🖪 🖻 🖆
Type filter text	Type pin f	unction				
<ul> <li>Image: state of the state of th</li></ul>	Enabled	Function SS6# SCK6 SMISO6 SMOSI6	Assignment Not assigned P02/TMCI1/SCK6/IRQ10/AN120 Not assigned P00/TMRI0/TXD6/SMOSI6/SSD	Pin Number Not assigned 8	Direction None IO IO	Remarks
Pin Function Pin Number	-					,
Overview Board Clocks Component Pins Ir	nterrupt					

Figure 4-42 Configure pin assignment - Config\_SCI6

Select the Config\_S12AD0 of Software Components. In the Pin Function list -> Assignment column, Ensure the 'Enable' tick box of AN000, AVCC0, AVSS0, ADTRG0#, VREFH0 and VREFL0 are checked and Assignment column of AN000 is P40, ADTRG0# is P07 as shown in **Figure 4-43**.

oftware Components 🛛 🕀 📮 🕌	Pin Function	1				- 🤣 🖪 🖻 e
Type filter text	Type pin fu	unction				
∽ 📥 r_bsp	Enabled	Function	Assignment	Pin Number	Direction	Remarks
📦 r_bsp		AN000	P40/IRQ8/AN000	173	1	
🗸 🚣 Compare Match Timer		AN001	Not assigned	Not assigned	None	
Config_CMT0		AN002	Not assigned	Not assigned	None	
Config_CMT1		AN003	Not assigned	Not assigned	None	
Config_CMT2 Linterrupt Controller Config_ICU		AN004	Not assigned	Not assigned	None	
		AN005	Not assigned	Not assigned	None	
		AN006	Not assigned	Not assigned	None	
V 🚣 Ports		AN007	Not assigned	Not assigned	None	
Config_PORT	$\checkmark$	AVCC0	AVCC0	175	1	
V 🚣 SCI/SCIF Asynchronous Mode		AVSS0	AVSS0	1	1	
✓ Config_SCl8 ✓ <sup>4</sup> / <sub>2</sub> SPI Clock Synchronous Mode		ADTRG0#	P07/IRQ15/ADTRG0#	176	1	
		VREFH0	VREFH0	174	1	
Config_SCI6		VREFLO	VREFLO	172	1	
✓ ▲ Single Scan Mode S12AD						
Config_S12AD0						
	<					

Figure 4-43 Configure pin assignment - Config\_S12AD0

Peripheral function configuration is now complete. Save the project using the File -> Save, then click ©Generate Code' at location of **Figure 4-44**.

		_
Pin configuration	1	Ð
∰ SC_Tutorial.scfg ⊠		

## Figure 4-44 Generate Code Button

Peripheral function configuration is now complete. Save the project using the File -> Save, then click 'Generate Code'. The Console pane should report 'Code generation is successful', as shown **Figure 4-45** below.

📮 Console 🐹	🖹 🔝 🔛 📑 🖃 🛨 📑 🛨 🗖
mart Configurator Output	
M04000001: File generated:src\smc_gen\general\r_cg_dmac.h	^
M04000001: File generated:src\smc_gen\general\r_cg_sci.h	
M04000001: File generated:src\smc_gen\general\r_cg_rspi.h	
M04000001: File generated:src\smc_gen\general\r_cg_ppg.h	
M04000001: File generated:src\smc_gen\general\r_cg_tmr.h	
M04000001: File generated:src\smc_gen\general\r_cg_lvd.h	
M04000001: File generated:src\smc_gen\general\r_cg_dmac_user.c	
M04000001: File generated:src\smc_gen\general\r_smc_entry.h	
M04000001: File generated:src\smc_gen\general\r_cg_cmt.h	
M04000001: File generated:src\smc_gen\general\r_cg_crc.h	
M04000001: File generated:src\smc_gen\general\r_cg_doc.h	
M04000001: File generated:src\smc_gen\general\r_cg_poe.h	
M04000001: File generated:src\smc_gen\general\r_cg_riic.h	
M04000001: File generated:src\smc_gen\general\r_cg_rtc.h	
M04000001: File generated:src\smc_gen\general\r_cg_elc.h	
M05000012: File generated:src\smc_gen\r_pincfg\Pin.h	
M05000012: File generated:src\smc_gen\r_pincfg\Pin.c	
M06000002: File generated:src\smc_gen\r_config\r_bsp_interrupt_config.h	
M06000002: File generated:src\smc_gen\r_config\r_bsp_irq_priority_cfg.h	
M06000002: File generated:src\smc_gen\general\r_smc_interrupt.c	
M06000002: File generated:src\smc_gen\general\r_smc_interrupt.h	
M06000002: File generated:src\smc_gen\general\r_bsp_icu_userdefine.h	
M0000002: Code generation is successful	
	*
<	> )

Figure 4-45 Smart Configurator console

# 4.6 Building the Project

The project template created by Smart Configurator can now be built. In the Project Explorer pane expand the 'src' folder then smc\_gen folder.

🎦 Project Explorer 🛛 📄 🔄 🌄 🗆 🗖
✓ SC_Tutorial
> 🔊 Includes
✓ 📇 src
∽ 🗁 smc_gen
> Config_CMT0
> 🔁 Config_CMT1
> 🦕 Config_CMT2
> 🦕 Config_ICU
> Config_PORT
> 🧽 Config_S12AD0 > 🍋 Config_SCI6
> Config_SCI8
> 🔁 general
> > r_bsp
> 🤂 r_config
> 👝 r_pincfg
> C_Tutorial.c
> 🗁 trash
🗟 custom.bat
SC_Tutorial HardwareDebug.launch
SC_Tutorial.scfg

Figure 4-46 Generated folder structure

Switch back to the 'C/C++' perspective using the to button on the top right of the e<sup>2</sup> studio workspace. Use 'Build Project' from the 'Project' menu or the button to build the tutorial. The project will build with no errors.

RENESAS

# 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 Smart Configurator, if the user needs to subsequently change any of the Smart Configurator-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 e<sup>2</sup> studio. Copy these files into the C:\Workspace\SC\_Tutorial\src directory. The files will be automatically added to the project as shown in **Figure 5-1**.

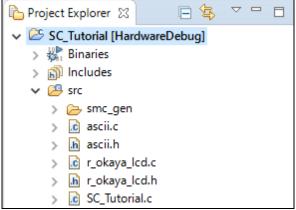


Figure 5-1 Adding files to the project



In the e<sup>2</sup> studio Project Tree, expand the 'src\smc\_gen\general' folder and open the file 'r\_cg\_userdefine.h' by double-clicking on it. Insert the following #defines in between the user code delimiter comments as shown below.

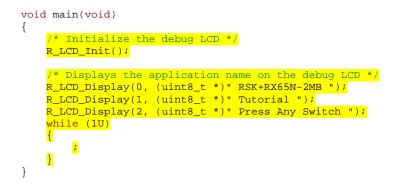
/\* Start user code for function. Do not edit comment generated here \*/
#define TRUE (1)
#define FALSE (0)

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

In the e<sup>2</sup> studio Project Tree, expand the 'src' folder and open the file 'SC\_Tutorial.c' by double-clicking on it. Add header files near the declarations '#include r\_smc\_entry.h'.

#include "r\_smc\_entry.h" #include "r\_okaya\_lcd.h" #include "r\_cg\_userdefine.h"

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:





## 5.1.1 SPI Code

The Okaya LCD display is driven by the SPI Master that was configured using Smart Configurator in §4.4.6. In the e2 studio Project Tree, expand the 'src\smc\_gen\Config\_SCI6' folder and open the file 'Config\_SCI6.h' by double-clicking on it. Insert the following code in the user code area at the end of the file:

/\* Start user code for function. Do not edit comment generated here \*/
/\* Exported functions used to transmit a number of bytes and wait for completion \*/
MD\_STATUS R\_SCI6\_SPIMasterTransmit(uint8\_t \* const tx\_buf, const uint16\_t tx\_num);

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

Now, open the Config\_SCI6\_user.c file and insert the following code in the user area for global:

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

/\* Flag used locally to detect transmission complete \*/ static volatile uint8\_t sci6\_txdone;

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

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

```
static void r_Config_SCI6_callback_transmitend(void)
```

```
/* Start user code for r_Config_SCI6_callback_transmitend. Do not edit comment generated here */
```

```
sci6_txdone = TRUE;
```

{

}

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

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

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

```
* Function Name: R_SCI6_SPIMasterTransmit
* Description : This function sends SPI6 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_SCI6_SPIMasterTransmit (uint8_t * const tx_buf,
                            const uint16_t tx_num)
{
  MD_STATUS status = MD_OK;
  /* Clear the flag before initiating a new transmission */
  sci6_txdone = FALSE;
   /* Send the data using the API */
  status = R_Config_SCI6_SPI_Master_Send(tx_buf, tx_num);
  /* Wait for the transmit end flag */
  while (FALSE == sci6_txdone)
  {
      /* Wait */
  }
  return (status);
}
End of function R_SCI6_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 Smart Configurator in §4.4.2. Open the file 'src\smc\_gen\Config\_CMT0\Config\_CMT0.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 'Config\_CMT0\_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\_Config\_CMT0\_cmi0\_interrupt function and insert the following line in the user code area:

```
static void r_Config_CMT0_cmi0_interrupt(void)
```

/\* Start user code for r\_Config\_CMT0\_cmi0\_interrupt. 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)
ł
  uint16 t ms count = 0;
  do
  {
     R_Config_CMT0_Start();
     while (FALSE == one_ms_delay_complete)
     {
       /* Wait */
     R_Config_CMT0_Stop();
     one_ms_delay_complete = FALSE;
     ms_count++;
  } while (ms_count < millisec);</pre>
}
  End of function R_CMT_MsDelay
                    *****
```



# 5.2 Additional include paths

Before the project can be built the compiler needs some additional include paths added. Select the SC\_Tutorial project in the Project Explorer pane. Right click in the Project Explorer window, and select 'Properties'. Navigate to 'C/C++ Build -> Settings ->Compiler -> Source and click the button as shown in **Figure 5-2**.

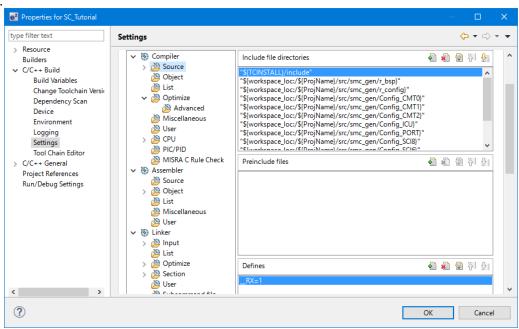


Figure 5-2 Adding additional search paths

In the 'Add directory path' dialog, click the 'Workspace' button and in the 'Folder selection' dialog browse to the 'SC\_Tutorial/src' folder and click 'OK'. e<sup>2</sup> studio formats the path as show in **Figure 5-3** below.

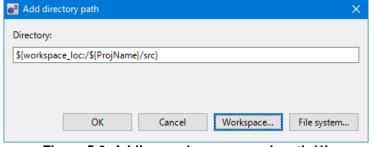


Figure 5-3 Adding workspace search path (1)

'Settings' dialog will appear, click 'Yes' to complete the include file directories.

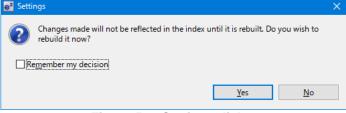


Figure 5-4 Settings dialog

Select 'Build Project' from the 'Project' menu, or use the button. e<sup>2</sup> 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-2MB Tutorial Press Any Switch' on three lines in the LCD display.

RENESAS

# 5.3 Switch Code Integration

API functions for user switch control are provided with the RSK+. Locate the files rskrx65n2mbdef.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 e<sup>2</sup> studio. Copy these files into the C:\Workspace\SC\_Tutorial\src directory.

The switch code uses interrupt code in the files Config\_ICU.h, Config\_ICU.c and Config\_ICU\_user.c and timer code in the files Config\_CMT1.h, Config\_CMT1.c, Config\_CMT1\_user.c, Config\_CMT2.h, Config\_CMT2.c and Config\_CMT2\_user.c as described in §4.4.2. and §4.4.3 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 e<sup>2</sup> studio Project Tree, expand the 'src\smc\_gen\Config\_ICU' folder and open the file 'Config\_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);



Now, open the Config\_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 IROIsFallingEdge
                           ************
     * Function Name: R_ICU_IRQSetFallingEdge
* 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
* Description : This function sets/clear the rising edge trigger for the
          specified ICU_IRQ.
* 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)
ł
  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
```

Open the Config\_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\_Config\_ICU\_irq11\_interrupt:

/\* Start user code for r\_Config\_ICU\_irq11\_interrupt. 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\_Config\_ICU\_irq13\_interrupt:

/\* Start user code for r\_Config\_ICU\_irq13\_interrupt. Do not edit comment generated here \*/

/\* Switch 2 callback handler \*/ R\_SWITCH\_IsrCallback2();



### 5.3.2 De-bounce Timer Code

In the e<sup>2</sup> studio Project Tree, expand the 'src\smc\_gen\Config\_CMT1' folder and open the 'Config\_CMT1\_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 Config\_CMT1\_user.c' file, insert the following code in the user code area inside the function r\_Config\_CMT1\_cmi1\_interrupt:

/\* Start user code for r\_Config\_CMT1\_cmi1\_interrupt. Do not edit comment generated here \*/

/\* Stop this timer - we start it again in the de-bounce routines \*/ R\_Config\_CMT1\_Stop(); /\* Call the de-bounce call back routine \*/

R\_SWITCH\_DebounceIsrCallback();

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

In the e<sup>2</sup> studio Project Tree, expand the 'src\smc\_gen\Config\_CMT2' folder and open the file 'Config\_CMT2\_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  ${}^{\star}/$ 

Open the Config\_CMT2\_user.c file and insert the following code in the user code area inside the function r\_Config\_CMT2\_cmi2\_interrupt:

/\* Start user code for r\_Config\_CMT2\_cmi2\_interrupt. Do not edit comment generated here \*/

/\* Stop this timer - we start it again in the de-bounce routines \*/ R\_Config\_CMT2\_Stop();

/\* Call the de-bounce call back routine \*/
R\_SWITCH\_DebounceIsrCallback();



## 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 §4.4.7 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, expand the 'src\smc\_gen\general' folder and 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 \*/ % f(x)=0

#define TRUE (1)
#define FALSE (0)
extern volatile uint8\_t g\_adc\_trigger;

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

In the e2 studio Project Tree, expand the 'src' folder and Open the file 'SC\_Tutorial.c' and add the highlighted code, resulting in the code shown below:

#include "r\_smc\_entry.h"
#include "r\_okaya\_lcd.h"
#include "r\_cg\_userdefine.h"
#include "Config\_S12AD0.h"
#include "config\_S12AD0.h"
#include "r\_rsk\_switch.h"
/\* 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);



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

```
void main(void)
{
    /* 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-2MB "
R_LCD_Display(1, (uint8_t *)" Tutorial ");
                                                    ");
    R_LCD_Display(2, (uint8_t *)" Press Any Switch ");
    /* Start the A/D converter */
    R_Config_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 = FALS<mark>E;</mark>
        /* 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_Config_S12AD0_Get_ValueResult(ADCHANNEL0, &adc_result);
             /* Display the result on the LCD */
            lcd_display_adc(adc_result);
            /* Reset the flag *,
            g_adc_complete = FALSE;
        else
        {
             /* do nothing */
        }
    }
}
```

Then add the definition for the switch call-back, get\_adc and lcd\_display\_adc functions below the main function, as shown below:

```
/* 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)
ł
  /* A variable to retrieve the adc result */
  uint16_t adc_result;
   /* Stop the A/D converter being triggered from the pin ADTRGOn */
  R_Config_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_Config_S12AD0_Get_ValueResult(ADCHANNEL0, &adc_result);
   /* Set AD conversion start trigger source back to ADTRGOn pin */
  R_Config_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));</pre>
  a = (uint8_t)((adc_result & 0x00F0) >> 4);
```



}

```
lcd_buffer[7] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));</pre>
  a = (uint8_t)(adc_result & 0x000F);
  lcd_buffer[8] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));</pre>
  /* Display the contents of the local string lcd_buffer */
  R_LCD_Display(3, (uint8_t *)lcd_buffer);
* End of function lcd_display_adc
*****
```

In the e<sup>2</sup> studio Project Tree, expand the 'src\smc\_gen\Config\_S12AD0' folder and open the file 'Config\_S12AD0.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 'Config\_S12AD0.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:

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

```
Function Name: R_S12AD0_SWTriggerStart
* Description : This function starts the AD0 converter.
* Arguments : None
* Return Value : None
void R_S12AD0_SWTriggerStart(void)
 IR(PERIB, INTB186) = OU;
 IEN(PERIB, INTB186) = 1U;
 S12AD.ADCSR.BIT.ADST = 10;
}
Function Name: R_S12AD0_SWTriggerStop
Description : This function stops the ADO converter.
Arguments : None
Return Value : None
*******
void R_S12AD0_SWTriggerStop(void)
 S12AD.ADCSR.BIT.ADST = 0U;
 IEN(PERIB, INTB186) = OU;
 IR(PERIB, INTB186) = OU;
}
```



Open the file Config\_S12AD0\_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\_Config\_S12AD0\_interrupt function, resulting in the code shown below:

static void r\_Config\_S12AD0\_interrupt(void)
{
 /\* Start user code for r\_Config\_S12AD0\_interrupt. 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. e<sup>2</sup> studio will build the project with no

Select 'Build Project' from the 'Project' menu, or use the button. e<sup>2</sup> 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-2MB\_Tutorial project for e<sup>2</sup> studio. Copy these files into the C:\Workspace\SC\_Tutorial\src directory.

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_SCI8_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 e<sup>2</sup> studio Project Tree, expand the 'src\smc\_gen\Config\_SCI8' folder and open the file 'Config\_SCI8.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_SCI8_AsyncTransmit(uint8_t * const tx_buf, const uint16_t tx_num);
/* Character is used to receive key presses from PC terminal */
```

<mark>extern uint8\_t g\_rx\_char;</mark>

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

Open the file 'Config\_SCI8\_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 locally to detect transmission complete \*/
static volatile uint8\_t sci8\_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\_Config\_SCI8\_callback\_transmittend function:

```
static void r_Config_SCI8_callback_transmitend (void)
{
    /* Start user code for r_Config_SCI8_callback_transmitend. Do not edit comment generated here */
    sci8_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\_Config\_SCI8\_callback\_receiveend function:

```
static void r_Config_SCI8_callback_receiveend(void)
{
    /* Start user code for r_Config_SCI8_callback_receiveend. 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 SCI8 receive buffer and callback function again */
    R_Config_SCI8_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_SCI8_AsyncTransmit
^{st} Description \, : This function sends SCI8 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_SCI8_AsyncTransmit(uint8_t * const tx buf, const uint16_t tx num)
{
  MD_STATUS status = MD_OK;
  /* Clear the flag before initiating a new transmission */
  sci8_txdone = FALSE;
  /* Send the data using the API */
  status = R_Config_SCI8_Serial_Send(tx_buf, tx_num);
  /* Wait for the transmit end flag */
  while (FALSE == sci8_txdone)
  {
     /* Wait */
  }
  return (status);
}
  * End of function R_SCI8_AsyncTransmit
    *****
```



{

#### 5.5.2 Main UART code

Open the file 'SC Tutorial.c'. Add the following declaration to near the top of the file:

```
#include "r_smc_entry.h"
#include "r_okaya_lcd.h"
#include "r_cg_userdefine.h"
#include "Config_S12AD0.h"
""
#include "r_rsk_switch.h"
#include "r_rsk_debug.h"
#include "Config_SCI8.h"
/* 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;
```

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

```
void main(void)
    /* 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-2MB ");
R_LCD_Display(1, (uint8_t *)" Tutorial ");
    R_LCD_Display(2, (uint8_t *) " Press Any Switch ");
    /* Start the A/D converter */
    R_Config_S12AD0_Start();
    /* Set up SCI8 receive buffer and callback function *,
    R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);
    /* Enable SCI8 operations */
    R_Config_SCI8_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 */
            if (16 == (++adc_count))
            {
                 adc_count = 0;
            }
             /* Send the result to the UART */
            uart_display_adc(adc_count, adc_result);
```

RENESAS

}

```
/* 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_Config_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 */
    }
}
```

Then, add the following function definition in the end of the file:

```
* Function Name : uart_display_adc
* Description : Converts adc result to a string and sends it to the UART1.
            : uint8_t : adc_count
* Argument
              uint16_t: adc result
* Return value : none
*****
static void uart_display_adc (const uint8_t adc_count, const uint16_t adc_result)
   /* Declare a temporary variable */
   char a;
   /* Declare temporary character string */
   static char uart_buffer[] = "ADC xH Value: xxxH\r\n";
   /\ast 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));
   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. e<sup>2</sup> 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 SCI8 (see §4.4.5). 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 SCI8. Return to this point in the Tutorial to add the LED user code.

# 5.6 LED Code Integration

Open the file 'SC\_Tutorial.c'. Add the following declaration to the near the top of the file:

```
#include "r_smc_entry.h"
#include "r_okaya_lcd.h"
#include "r_cg_userdefine.h"
#include "Config_S12AD0.h"
#include "r_rsk_switch.h"
#include "r_rsk_debug.h"
#include "Config_SCI8.h"
#include "rskrx65n2mbdef.h"
/* 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;
/* Prototype declaration for led_display_count */
static void led_display_count(const uint8_t count);
```

Add the following highlighted code in the main function:

```
void main(void)
    /* 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-2MB '
R_LCD_Display(1, (uint8_t *)" Tutorial ");
                                                     ");
    R_LCD_Display(2, (uint8_t *)" Press Any Switch ");
    /* Start the A/D converter */
    R_Config_S12AD0_Start();
    /* Set up SCI8 receive buffer and callback function */
    R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);
    /* Enable SCI8 operations */
    R_Config_SCI8_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;
        led_display_count(adc_count);
        /* Send the result to the UART */
        uart_display_adc(adc_count, adc_result);
        /* Reset the flag */
        g_adc_trigger = FALSE;
    }
    /* SW3 is directly wired into the ADTRGOn pin so will
       cause the interrupt to fire */
    else if (TRUE == g_adc_complete)
    {
        /* Get the result of the A/D conversion */
        R_Config_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;
        ied_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 */
    }
}
```

}



Then, add the following function definition at the end of the file:

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

Select 'Build Project' from the 'Build' menu, or use the <sup>st</sup> button. e<sup>2</sup> 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 'SC\_Tutorial' project is selected. To enter the debug configurations, click upon the arrow next to the debug button and select 'Debug Configuration'.

e <sup>2</sup> Debug Configurations		×
Create, manage, and run configurations		
Image: Second Secon	Name: SC_Tutorial HardwareDebug	<u>B</u> rowse <u>Y</u> ariables Searc <u>h</u> Project B <u>r</u> owse
(?)		<u>D</u> ebug Close

Figure 6-1 Debug Configurations

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-2MB the setting should be 24.0000).

For more information on powering the RSK+RX65N-2MB please refer to the User Manual.

ebug hardware: E2 Lite (RX) V Target	Device: R5F565NE		
GDB Settings Connection Settings Debug Too	ol Settings		
✓ Clock			^
Main Clock Source	EXTAL	$\checkmark$	
Extal Frequency[MHz]	24.0000		
Permit Clock Source Change On Writing	l Yes	$\sim$	
<ul> <li>Connection with Target Board</li> </ul>			
Emulator	(Auto)		
Connection Type	JTag	$\checkmark$	
JTag Clock Frequency[MHz]	6.00	~	
Fine Baud Rate[Mbps]	1.50	$\sim$	
Hot Plug	No	$\checkmark$	
✓ Power			
Power Target From The Emulator (MAX 2		$\checkmark$	
Supply Voltage	3.3V	$\sim$	
<ul> <li>CPU Operating Mode</li> </ul>			
Register Setting	Single Chip	$\checkmark$	
Mode pin	Single-chip mode	$\sim$	
Change startup bank	No	$\sim$	
Startup bank	Bank 0	$\sim$	

Figure 6-2 Connection Settings



Connect the E2 Lite to the PC and the RSK+ E1/E2 Lite 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 'SC\_Tutorial' project is selected. To debug the project, click the button. The dialog shown in **Figure 6-3** will be displayed.

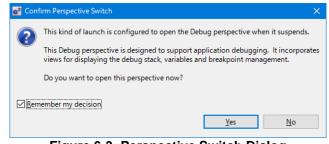


Figure 6-3 Perspective Switch Dialog

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 Smart Configurator function 'PowerOn\_Reset\_PC' as shown in **Figure 6-4**.

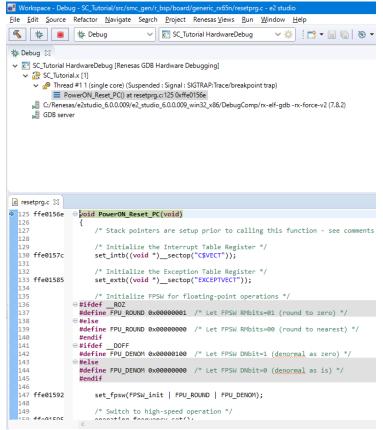


Figure 6-4 Debugger start up screen

For more information on the e<sup>2</sup> studio debugger refer to the Tutorial manual. To run the code click the button. The debugger will stop again at the beginning of the main function. Press **l** 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	Window	Help		
menu bar.	<u>c</u> - G	3	Welcome	
		0	Help Contents	
		<u>?</u> ?	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.

© 2017 Renesas Electronics Europe Limited. All rights reserved.

© 2017 Renesas Electronics Corporation. All rights reserved.

© 2017 Renesas System Design Co., Ltd. All rights reserved.



# **REVISION HISTORY**

# RSK+RX65N-2MB Smart Configurator Tutorial Manual

Rev.	Date	Description			
		Page	Summary		
1.00	May 31, 2017	_	First Edition issued		

Renesas StarterKit+ Manual:Smart Configurator Tutorial<br/>ManualPublication Date:Rev. 1.00May 31, 2017Published by:Renesas Electronics Corporation



### SALES OFFICES

**Renesas Electronics Corporation** 

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information. Renesas Electronics America Inc. 2001 Scott Boulevard Santa Clara, CA 90505-2549, U.S.A. Tel: +1408-588-6100, Fax: +14-08-588-6130 Renesas Electronics Canada Limited 2051 Yong Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3 Tel: +1905-237-2004 Renesas Electronics Canado Limited Dukes Meadow, Milboad Road, Bourne End, Buckinghamshire, SL8 5FH, U.K Tel: +44-1628-585-100, Fax: +44-1628-585-900 Renesas Electronics Canado Limited Dukes Meadow, Milboad Road, Bourne End, Buckinghamshire, SL8 5FH, U.K Tel: +49-211-8503-0, Fax: +44-1628-585-900 Renesas Electronics Change End, Buckinghamshire, SL8 5FH, U.K Tel: +49-211-8503-0, Fax: +49-211-5503-1327 Renesas Electronics (Shanghai) Co., Ltd. Room 1709, Quantum Plaza, No.27 ZhiChunLu Haldian District, Beijing 100191, P.R.China Tel: +89-10-825-1155, Fax: +86-10-825-7679 Renesas Electronics (Shanghai) Co., Ltd. Niti 301, Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333 Tel: +86-21-2226-0888, Fax: +86-21-2226-0989 Renesas Electronics Taiwan Co., Ltd. Diff. No. 303, Tower 2, Garand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong Tel: +852-1852-6688, Fax: +862-218-2020 Renesas Electronics Taiwan Co., Ltd. Diff. No. 303, Tower 2, Garand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong Tel: +852-266-6688, Fax: +862-218-0020 Renesas Electronics Malwing Kong Limited Diff. No. 303, Towir +65-6213-0300 Renesas Electronics Malwing Kong, Imited Diff. No. 303, Towir +65-213-0300 Renesas Electronics India Pvt. Ltd. No.777C, 100 Ke B, Meara Anncorp, Anncorp Trade Centre, No. 18, Jin Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia Tel: +603-735-9390, Fax: +03-735-7670 Renesas Electronics India Pvt. Ltd. No.777C, 100 Ke B, Meara Anncorp, Jincorp Trade Centre, No. 18, Jin Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia Tel: +603-735-9390, Fax: +043-730570 Renesas E

RX65N Group



R20UT3894EG0100