



# RX72N Group

Renesas Starter Kit+ for RX72N Smart Configurator Tutorial Manual For e<sup>2</sup> studio

## RENESAS 32-Bit MCU RX Family / RX700 Series

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics 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).

Rev. 1.00 Nov 2019

#### 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.
- Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims 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, drawings, charts, programs, algorithms, and 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 reverse engineer 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, copying or reverse engineering.
- 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; 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.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; 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 any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

- 6. When using 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 dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of 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. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating 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. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using 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 comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
- 10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
- This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
   Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas
- Electronics products.
- (Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
- (Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

#### **Corporate Headquarters**

TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

#### Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

#### Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: <a href="http://www.renesas.com/contact/">www.renesas.com/contact/</a>.

## 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. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

#### 2. Processing at power-on

The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time 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 time 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 time when power is supplied until the power is supplied until the power reaches the level at which reseting is specified.

3. Input of signal during power-off state

Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

Handle unused pins in accordance with the directions given under handling of unused pins in the manual. 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 the 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.

#### 5. Clock signals

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.

6. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between  $V_{IL}$  (Max.) and  $V_{IH}$  (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between  $V_{IL}$  (Max.) and  $V_{IH}$  (Min.).

7. Prohibition of access to reserved addresses

Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of 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 RX72N microcontroller may be found in 'RX72N Group User's Manual: Hardware' and within the provided sample code. The setup procedure for the RSK+ Web installer is described in the Quick Start Guide.

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 RX72N 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.	Renesas Starter Kit+ for RX72N User's Manual	R20UT4443EG
Tutorial Manual	Provides a guide to setting up RSK+ environment, running sample code and debugging programs.	Renesas Starter Kit+ for RX72N Tutorial Manual	R20UT4440EG
Quick Start Guide	Provides simple instructions to setup the RSK+ and run the first sample.	Renesas Starter Kit+ for RX72N Quick Start Guide	R20UT4441EG
Smart Configurator Tutorial	Provides a guide to code generation and importing into the e <sup>2</sup> studio IDE.	Renesas Starter Kit+ for RX72N Smart Configurator Tutorial Manual	R20UT4442EG
Schematics	Full detail circuit schematics of the RSK+.	Renesas Starter Kit+ for RX72N Schematics	R20UT4435EG
Hardware Manual	Provides technical details of the RX72N microcontroller.	RX72N Group User's Manual: Hardware	R01UH0824EJ

### 2. List of Abbreviations and Acronyms

Abbreviation	Full Form
ADC	Analog-to-Digital Converter
API	Application Programming Interface
bps	bits per second
CMT	Compare Match Timer
СОМ	COMmunications port referring to PC serial port
CPU	Central Processing Unit
E1 / E2 Lite	Renesas On-chip Debugging Emulator
GUI	Graphical User Interface
IDE	Integrated Development Environment
IRQ	Interrupt Request
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LSB	Least Significant Bit
LVD	Low Voltage Detect
MCU	Micro-controller Unit
MSB	Most Significant Bit
PC	Personal Computer
PLL	Phase-locked Loop
This is a Digilent Pmod™ Compatible connector. Pmod™ is registered	
Pmod™	Digilent-Pmod_Interface_Specification
PSU	Power Supply Unit
RAM	Random Access Memory
ROM	Read Only Memory
RSK+	Renesas Starter Kit+
RTC	Real Time Clock
SCI	Serial Communications Interface
SPI	Serial Peripheral Interface
TFT	Thin Film Transistor
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
WDT	Watchdog Timer

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

## Table of Contents

11       Purpose       8         1.2       Features       8         1.2       Features       8         2.       Introduction       9         3.       Project Creation with e <sup>2</sup> studio       10         3.1       Introduction       10         3.2       Creating the Project       10         4.       Smart Configurator Using the e <sup>2</sup> studio       13         4.1       Introduction       13         4.2       Tre/Board tabbed page       15         4.3       The Board tabbed page       15         4.4       The Configuration using Smart Configurator       14         4.4       The Components' tabbed page       16         4.4.1       Clocks' tabbed page       17         4.5.2       Compare Match Timer       18         4.5.3       Interrupt Controller       22         4.5.4       Ports       24         4.5.5       SCI/SCIF Asynchronous Mode       28         4.5.6       SPI Clock Synchronous Mode       28         4.5.6       SPI Clock Synchronous Mode       24         4.5.7       Single Scan Mode S12AD       34         4.6       The Pins' tabbed page       37     <	1. Overview	8
2. Introduction       9         3. Project Creation with e <sup>2</sup> studio       10         3.1 Introduction       10         3.2 Creating the Project       10         4. Smart Configurator Using the e <sup>2</sup> studio       13         4.1 Introduction       13         4.2 Project Configuration using Smart Configurator       14         4.3 The Board tabbed page       15         4.3.1 Board configuration page       15         4.3.1 Board configuration page       16         4.3.1 Clocks configuration       16         4.4.1 Clocks configuration       16         4.5.1 Add a software component into the project       17         4.5.2 Compare Match Timer       18         4.5.3 Interrupt Controller       22         4.5.4 Ports       22         4.5.5 SCI/SCIF Asynchronous Mode       28         4.5.6 SPI Clock Synchronous Mode       31         4.5.7 Single Scan Mode S12AD       34         4.5.8 Cl/SCIF Asynchronous Mode       37         4.5.1 Charge pin assignment of a software component       37         4.5.2 Clock Synchronous Mode       34         4.5.1 Charge pin assignment of a software component       37         4.5.2 Clock Synchronous Mode       34         4.5.1 Charge pin		
3. Project Creation with e <sup>2</sup> studio.       10         3.1 Introduction       10         3.2 Creating the Project       10         4. Smart Configurator Using the e <sup>2</sup> studio       13         4.1 Introduction       13         4.2 Project Configuration using Smart Configurator       14         4.3 The 'Board' tabbed page       15         4.3.1 Board configuration page       15         4.3.1 Board configuration page       15         4.4.1 Clocks configuration       16         4.5.1 Add a software component into the project       17         4.5.1 Add a software component into the project       17         4.5.2 Compare Match Timer       18         4.5.3 Interrupt Controller       22         4.5.4 Ports       24         4.5.5 SCI/SCIF Asynchronous Mode       31         4.5.7 Single Scan Mode S12AD       34         4.6 The 'Pins' tabbed page       37         4.6.1 Change pin assignment of a software component       37         4.7 Building the Project       40         5.1 Set Code Integration       41         5.2 LCD Code Integration       42         5.2 LCD Code Integration       42         5.3 Additional include paths       45         5.4 Switch Code Integration	1.2 Features	8
3.1       Introduction       10         3.2       Creating the Project       10         4.       Smart Configurator Using the e <sup>2</sup> studio       13         4.1       Introduction       13         4.2       Project Configuration using Smart Configurator       14         4.3       The Board tabbed page       15         4.3.1       Board configuration page       15         4.3.1       Board configuration page       16         4.4.1       Clocks' tabbed page       16         4.4.1       Clocks configuration       16         4.5.1       The' Components' tabbed page       17         4.5.2       Compare Match Timer       18         4.5.3       Interrupt Controller       22         4.5.4       Ports       24         4.5.5       SCI/SCIF Asynchronous Mode       31         4.5.6       SPI Clock Synchronous Mode       31         4.5.7       Single Scan Mode S12AD       34         4.6       The'Pins' tabbed page       37         4.6.1       Change pin assignment of a software component.       37         4.5       SUber Code Integration       41         5.1       SPI Code       42 <td< td=""><td>2. Introduction</td><td>9</td></td<>	2. Introduction	9
3.1       Introduction       10         3.2       Creating the Project       10         4.       Smart Configurator Using the e <sup>2</sup> studio       13         4.1       Introduction       13         4.2       Project Configuration using Smart Configurator       14         4.3       The Board tabbed page       15         4.3.1       Board configuration page       15         4.3.1       Board configuration page       16         4.4.1       Clocks' tabbed page       16         4.4.1       Clocks configuration       16         4.5.1       The' Components' tabbed page       17         4.5.2       Compare Match Timer       18         4.5.3       Interrupt Controller       22         4.5.4       Ports       24         4.5.5       SCI/SCIF Asynchronous Mode       31         4.5.6       SPI Clock Synchronous Mode       31         4.5.7       Single Scan Mode S12AD       34         4.6       The'Pins' tabbed page       37         4.6.1       Change pin assignment of a software component.       37         4.5       SUber Code Integration       41         5.1       SPI Code       42 <td< td=""><td>2. Drainet Creation with <math>a^2</math> studie</td><td>10</td></td<>	2. Drainet Creation with $a^2$ studie	10
3.2       Creating the Project       10         4. Smart Configurator Using the e <sup>2</sup> studio       13         4.1       Introduction       13         4.2       Project Configuration using Smart Configurator       14         4.3       The 'Board' tabbed page       15         4.4.1       The 'Clocks' tabbed page       16         4.4.1       The 'Components' tabbed page       16         4.4.1       Clocks configuration       16         4.5.1       Add a software component into the project       17         4.5.1       Add a software component into the project       17         4.5.2       Compare Match Timer       18         4.5.3       Interrupt Controller       22         4.5.4       Ports       24         4.5.5       SCI/SCIF Asynchronous Mode       28         4.5.6       SPI Clock Synchronous Mode       31         4.5.7       Single Scan Mode S12AD       34         4.6       The 'Pins' tabbed page       37         4.6.1       Change pin assignment of a software component       37         4.6.1       Change pin assignment of a software component       37         5.2       COM Integration       41         5.2       Code I		
4. Smart Configurator Using the e <sup>2</sup> studio       13         11 Introduction       13         4.2 Project Configuration using Smart Configurator       14         4.3 The 'Board' tabbed page       15         4.4 The 'Clocks' tabbed page       16         4.4.1 Clocks configuration page       16         4.4.1 Clocks configuration age       17         4.5 The 'Component' tabbed page       17         4.5.1 Add a software component into the project       17         4.5.2 Compare Match Timer       18         4.5.3 Interrupt Controller       22         4.5.4 Ports       224         4.5.5 SCI/SCIF Asynchronous Mode       28         4.5.6 SPI Clock Synchronous Mode       31         4.5.7 Single Scan Mode S12AD       34         4.6 The 'Pins' tabbed page       37         4.6.1 Change pin assignment of a software component.       37         4.7 Building the Project       40         5.1 Veer Code Integration       41         5.1 Project Settings       41         5.2 CMT Code       43         5.4.3 Main Switch and ADC Code       49         5.4.3 Main Switch and ADC Code       49         5.4.3 Main Switch and ADC Code       50         5.6.1 SCI Code       55		
4.1       Introduction       13         4.2       Project Configuration using Smart Configurator       14         4.3       The Board' tabbed page       15         4.3.1       Board configuration page       15         4.4.1       The Clocks' tabbed page       16         4.4.1       Clocks configuration       16         4.4.1       Clocks configuration       16         4.4.1       Clocks configuration       16         4.5.1       Add a software component into the project.       17         4.5.2       Compare Match Timer       18         4.5.3       Interrupt Controller       22         4.5.4       Ports       24         4.5.5       SCI/SCIF Asynchronous Mode       28         4.5.6       SPI Clock Synchronous Mode       31         4.5.7       Single Scan Mode S12AD       34         4.6       The 'Pins' tabbed page       37         4.6.1       Change pin assignment of a software component       37         4.7       Building the Project       40         5.2       LCD Code Integration       41         5.2.1       SPI Code       43         5.2.2       CMT Code       44         5.3		
4.2       Project Configuration using Smart Configurator       14         4.3       The 'Board' tabbed page.       15         4.4       The 'Clocks' tabbed page.       16         4.4.1       Clocks configuration page       16         4.4.1       Clocks configuration       16         4.5       The 'Components' tabbed page       17         4.5.1       Add a software component into the project.       17         4.5.2       Compare Match Timer.       18         4.5.3       Interrupt Controller       22         4.5.4       Ports       24         4.5.5       SCI/SCIF Asynchronous Mode       31         4.5.7       Single Scan Mode S12AD       34         4.6       The 'Pins' tabbed page       37         4.6.1       Charge pin assignment of a software component.       37         4.7       Building the Project.       40         5.1       SPI Code       44         5.2       CMT Code       43         5.2.1       SPI Code       44         5.2       CMT Code       43         5.2.2       CMT Code       43         5.3       Additional include paths       45         5.4.1       Inte		
4.3       The 'Board' tabbed page.       15         4.3.1       Board configuration page       15         4.4       The 'Clocks' tabbed page       16         4.4.1       Clocks configuration       16         4.5       The 'Components' tabbed page       17         4.5.1       Add a software component into the project       17         4.5.2       Compare Match Timer       18         4.5.3       Interrupt Controller       22         4.5.4       Ports       24         4.5.5       SCI/SCIF Asynchronous Mode       31         4.5.6       SPI Clock Synchronous Mode       31         4.5.7       Single Scan Mode S12AD       34         4.6       The 'Pins' tabbed page       37         4.6.1       Change pin assignment of a software component.       37         4.6.1       Change pin assignment of a software component.       37         4.7       Building the Project.       40         5.2       CMT Code       43         5.2.1       SPI Code       43         5.2.2       CMT Code       43         5.4       Switch Code Integration       46         5.4       Switch Code Integration       46		
4.3.1       Board configuration page       15         4.4       The 'Clocks' tabbed page       16         4.4.1       Clocks configuration       16         4.5       The 'Components' tabbed page       17         4.5.1       Add a software component into the project       17         4.5.2       Compare Match Timer       18         4.5.3       Interrupt Controller       22         4.5.4       Ports       24         4.5.5       SCI/SCIF Asynchronous Mode       28         4.5.6       SPI Clock Synchronous Mode       31         4.5.7       Single Scan Mode S12AD       34         4.6       The 'Pins' tabbed page       37         4.7       Building the Project       40         5.       User Code Integration       41         5.1       Project Settings       41         5.2       CMT Code       43         5.2.1       SPI Code       44         5.3       Additional include paths       45         5.4       Switch Code Integration       42         5.4       Switch Code Integration       46         5.4.1       Interrupt Code       40         5.4       Main Switch and ADC Code		
4.4       The 'Clocks' tabbed page       16         4.4.1       Clocks configuration       16         4.5       The 'Components' tabbed page       17         4.5.1       Add a software component into the project       17         4.5.2       Compare Match Timer       18         4.5.3       Interrupt Controller       22         4.5.4       Ports       22         4.5.5       SCI/SCIF Asynchronous Mode       28         4.5.6       SPI Clock Synchronous Mode       31         4.5.7       Single Scan Mode S12AD       34         4.6       The 'Pins' tabbed page       37         4.6.1       Change pin assignment of a software component       37         4.6.1       Change pin assignment of a software component       37         4.7       Building the Project       40         5.       User Code Integration       41         5.1       Project Settings       41         5.2.1       SPI Code       43         5.2.2       CMT Code       44         5.3       Additional include paths       45         5.4       Switch Code Integration       46         5.4.1       Interrupt Code       50         5	- 10	
4.4.1       Clocks configuration       16         4.5       The 'Components' tabbed page       17         4.5.1       Add a software component into the project       17         4.5.2       Compare Match Timer       18         4.5.3       Interrupt Controller       22         4.5.4       Ports       22         4.5.5       SCI/SCIF Asynchronous Mode       28         4.5.6       SPI Clock Synchronous Mode       34         4.6       The 'Pins' tabbed page       37         4.6.1       Change pin assignment of a software component       37         4.6.1       Change pin assignment of a software component       37         4.7       Building the Project       40         5.       User Code Integration       41         5.1       Project Settings       41         5.2       CMT Code       43         5.2.1       SPI Code       44         5.3       Additional include paths       45         5.4       Switch Code Integration       46         5.4.1       Interrupt Code       46         5.4.2       De-bounce Timer Code       46         5.4.1       Interrupt Code       55         5.6	5 1 5	
4.5       The 'Components' tabbed page       17         4.5.1       Add a software component into the project       17         4.5.2       Compare Match Timer       18         4.5.3       Interrupt Controller       22         4.5.4       Ports       24         4.5.5       SCI/SCIF Asynchronous Mode       28         4.5.6       SPI Clock Synchronous Mode       31         4.5.7       Single Scan Mode S12AD       34         4.6       The 'Pins' tabbed page       37         4.6.1       Change pin assignment of a software component       37         4.6.1       Change pin assignment of a software component       37         4.7       Building the Project       40         5. User Code Integration       41         5.2       LCD Code Integration       42         5.2.1       SPI Code       43         5.2.2       CMT Code       44         5.3       Additional include paths       45         5.4       Switch Code Integration       46         5.4.1       Interrupt Code       46         5.4.2       De-bounce Timer Code       49         5.4.3       Main Switch and ADC Code       55         5.6		
4.5.1       Add a software component into the project.       17         4.5.2       Compare Match Timer.       18         4.5.3       Interrupt Controller       22         4.5.4       Ports.       24         4.5.5       SCI/SCIF Asynchronous Mode       28         4.5.6       SPI Clock Synchronous Mode       31         4.5.7       Single Scan Mode S12AD       34         4.6       The 'Pins' tabbed page       37         4.6.1       Change pin assignment of a software component       37         4.6.1       Change pin assignment of a software component       37         4.6.1       Change pin assignment of a software component       37         4.6.1       Project Settings       41         5.1       Project Settings       41         5.2       CMT Code       42         5.2.1       SPI Code       43         5.2.2       CMT Code       44         5.3       Additional include paths       45         5.4       Switch Code Integration       46         5.4.1       Interrupt Code       44         5.3       Main Switch and ADC Code       55         5.4       Waith Code Integration       55	- 5	
4.5.2       Compare Match Timer	1 - 1 - 5	
4.5.3       Interrupt Controller       22         4.5.4       Ports       24         4.5.5       SCI/SCIF Asynchronous Mode       28         4.5.6       SPI Clock Synchronous Mode       31         4.5.7       Single Scan Mode S12AD       34         4.6       The 'Pins' tabbed page       37         4.6.1       Change pin assignment of a software component       37         4.7       Building the Project       40         5. User Code Integration       41         5.1       Project Settings       41         5.2       CMT Code       42         5.2.1       SPI Code       43         5.2.2       CMT Code       44         5.3       Additional include paths       45         5.4       Switch Code Integration       46         5.4.1       Interrupt Code       46         5.4.2       De-bounce Timer Code       49         5.4.3       Main Switch and ADC Code       50         5.5       Debug Code Integration       55         5.6.1       SCI Code       55         5.6.2       Main UART code       55         5.7       LED Code Integration       56         5.7 </td <td></td> <td></td>		
4.5.4       Ports       24         4.5.5       SCI/SCIF Asynchronous Mode       28         4.5.6       SPI Clock Synchronous Mode       31         4.5.7       Single Scan Mode S12AD.       34         4.6       The 'Pins' tabbed page       37         4.6.1       Change pin assignment of a software component.       37         4.6.1       Change pin assignment of a software component.       37         4.7       Building the Project.       40         5.       User Code Integration.       41         5.1       Project Settings.       41         5.2       LCD Code Integration.       42         5.2.1       SPI Code       43         5.2.2       CMT Code       44         5.3       Additional include paths       45         5.4       Switch Code Integration       46         5.4.1       Interrupt Code       46         5.4.2       De-bounce Timer Code       49         5.4.3       Main Switch and ADC Code       50         5.5       Debug Code Integration       55         5.6.1       SCI Code       55         5.6.2       Main UART code       56         5.7       LED Code Integratio		
4.5.5       SCI/SCIF Asynchronous Mode       28         4.5.6       SPI Clock Synchronous Mode       31         4.5.7       Single Scan Mode S12AD       34         4.6       The 'Pins' tabbed page       37         4.6.1       Change pin assignment of a software component       37         4.7       Building the Project       40         5.       User Code Integration       41         5.       User Code Integration       41         5.2       LCD Code Integration       42         5.2.1       SPI Code       43         5.2.2       CMT Code       44         5.3       Additional include paths       45         5.4       Switch Code Integration       46         5.4.1       Interrupt Code       49         5.4.3       Main Switch and ADC Code       50         5.5       Debug Code Integration       55         5.6       UART Code       55         5.6.1       SCI Code       55         5.6.2       Main UART code       56         5.7       LED Code Integration       59	•	
4.5.6       SPI Clock Synchronous Mode       31         4.5.7       Single Scan Mode S12AD       34         4.6       The 'Pins' tabbed page       37         4.6.1       Change pin assignment of a software component       37         4.6.1       Change pin assignment of a software component       37         4.7       Building the Project       40         5.       User Code Integration       41         5.1       Project Settings       41         5.2       LCD Code Integration       42         5.2.1       SPI Code       43         5.2.2       CMT Code       43         5.2.3       Additional include paths       45         5.4       Switch Code Integration       46         5.4.1       Interrupt Code       44         5.3       Additional include paths       45         5.4.3       Main Switch and ADC Code       50         5.4       UART Code Integration       55         5.6.1       SCI Code       55         5.6.2       Main UART code       56         5.7       LED Code Integration       59		
4.5.7Single Scan Mode S12AD.344.6The 'Pins' tabbed page374.6.1Change pin assignment of a software component.374.7Building the Project.405. User Code Integration.415.1Project Settings.415.2LCD Code Integration425.2.1SPI Code435.2.2CMT Code445.3Additional include paths455.4Switch Code Integration465.4.1Interrupt Code465.4.2De-bounce Timer Code495.4.3Main Switch and ADC Code505.5Debug Code Integration555.6.1SCI Code555.6.1SCI Code555.6.2Main UART code565.7LED Code Integration59		
4.6       The 'Pins' tabbed page       37         4.6.1       Change pin assignment of a software component       37         4.7       Building the Project       40         5.       User Code Integration       41         5.1       Project Settings       41         5.2       LCD Code Integration       42         5.2.1       SPI Code       43         5.2.2       CMT Code       44         5.3       Additional include paths       45         5.4       Switch Code Integration       46         5.4.1       Interrupt Code       46         5.4.2       De-bounce Timer Code       49         5.4.3       Main Switch and ADC Code       50         5.5       Debug Code Integration       55         5.6.1       SCI Code       55         5.6.2       Main UART code       56         5.7       LED Code Integration       59	•	
4.7       Building the Project       40         5. User Code Integration       41         5.1       Project Settings       41         5.2       LCD Code Integration       42         5.2.1       SPI Code       43         5.2.2       CMT Code       44         5.3       Additional include paths       45         5.4       Switch Code Integration       46         5.4.1       Interrupt Code       46         5.4.2       De-bounce Timer Code       49         5.4.3       Main Switch and ADC Code       50         5.5       Debug Code Integration       55         5.6       UART Code Integration       55         5.6.1       SCI Code       55         5.6.2       Main UART code       56         5.7       LED Code Integration       59		
5. User Code Integration       41         5.1 Project Settings       41         5.2 LCD Code Integration       42         5.2.1 SPI Code       43         5.2.2 CMT Code       44         5.3 Additional include paths       45         5.4 Switch Code Integration       46         5.4.1 Interrupt Code       46         5.4.2 De-bounce Timer Code       49         5.4.3 Main Switch and ADC Code       50         5.5 Debug Code Integration       55         5.6 UART Code Integration       55         5.6.1 SCI Code       55         5.6.2 Main UART code       56         5.7 LED Code Integration       59	4.6.1 Change pin assignment of a software component	
5.1Project Settings415.2LCD Code Integration425.2.1SPI Code435.2.2CMT Code445.3Additional include paths455.4Switch Code Integration465.4.1Interrupt Code465.4.2De-bounce Timer Code495.4.3Main Switch and ADC Code505.5Debug Code Integration555.6UART Code Integration555.6.1SCI Code555.7LED Code Integration55	4.7 Building the Project	40
5.1Project Settings415.2LCD Code Integration425.2.1SPI Code435.2.2CMT Code445.3Additional include paths455.4Switch Code Integration465.4.1Interrupt Code465.4.2De-bounce Timer Code495.4.3Main Switch and ADC Code505.5Debug Code Integration555.6UART Code Integration555.6.1SCI Code555.7LED Code Integration55	5. User Code Integration	
5.2       LCD Code Integration       42         5.2.1       SPI Code       43         5.2.2       CMT Code       44         5.3       Additional include paths       45         5.4       Switch Code Integration       46         5.4.1       Interrupt Code       46         5.4.2       De-bounce Timer Code       49         5.4.3       Main Switch and ADC Code       50         5.5       Debug Code Integration       55         5.6       UART Code       50         5.6.1       SCI Code       55         5.6.2       Main UART code       56         5.7       LED Code Integration       59		
5.2.2       CMT Code       44         5.3       Additional include paths       45         5.4       Switch Code Integration       46         5.4.1       Interrupt Code       46         5.4.2       De-bounce Timer Code       49         5.4.3       Main Switch and ADC Code       50         5.5       Debug Code Integration       55         5.6       UART Code Integration       55         5.6.1       SCI Code       55         5.6.2       Main UART code       56         5.7       LED Code Integration       59		
5.3Additional include paths455.4Switch Code Integration465.4.1Interrupt Code465.4.2De-bounce Timer Code495.4.3Main Switch and ADC Code505.5Debug Code Integration555.6UART Code Integration555.6.1SCI Code555.6.2Main UART code565.7LED Code Integration59	5.2.1 SPI Code	43
5.4Switch Code Integration465.4.1Interrupt Code465.4.2De-bounce Timer Code495.4.3Main Switch and ADC Code505.5Debug Code Integration555.6UART Code Integration555.6.1SCI Code555.6.2Main UART code565.7LED Code Integration59		
5.4.1       Interrupt Code       .46         5.4.2       De-bounce Timer Code       .49         5.4.3       Main Switch and ADC Code       .50         5.5       Debug Code Integration       .55         5.6       UART Code Integration       .55         5.6.1       SCI Code       .55         5.6.2       Main UART code       .56         5.7       LED Code Integration       .59		
5.4.2       De-bounce Timer Code       .49         5.4.3       Main Switch and ADC Code       .50         5.5       Debug Code Integration       .55         5.6       UART Code Integration       .55         5.6.1       SCI Code       .55         5.6.2       Main UART code       .56         5.7       LED Code Integration       .59		
5.4.3Main Switch and ADC Code505.5Debug Code Integration555.6UART Code Integration555.6.1SCI Code555.6.2Main UART code565.7LED Code Integration59	•	
5.5         Debug Code Integration         .55           5.6         UART Code Integration         .55           5.6.1         SCI Code         .55           5.6.2         Main UART code         .56           5.7         LED Code Integration         .59		
5.6       UART Code Integration		
5.6.1       SCI Code	0 0	
5.6.2       Main UART code	0	
5.7 LED Code Integration		
6. Debugging the Project		
	6. Debugging the Project	61
7. Additional Information	7. Additional Information	63

## RENESAS

### 1. Overview

#### 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.

The tutorial examples in this manual assume that installation procedures described in the RSK+ Quick Start Guide have been completed. Please refer to the Quick Start Guide for details of preparing the configuration.

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 RX72N 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

•	Start e <sup>2</sup> studio and select a suitable location	e <sup>2</sup> Eclipse Launcher X		
for the project workspace.		Select a directory as workspace		
		e <sup>2</sup> studio uses the workspace directory to store its preferences and development artifacts.		
		Workspace: C:\Workspace V Browse		
		Use this as the default and do not ask again		
		Launch Cancel		
		2 Induse-Ando - D X		
•	In the Welcome page, click 'Create a new C/C++ project'.	Be (At linguists formation and the state of		
		Contraction     Contracti		
	(The Welcome page can also be opened from 'Help'-> 'Welcome'.)			
		Create a new C(C++ project     Create a new C(C++ servicet     Create a new C(C++ service		
		Import existing projects Import existing eritation the filesystem or archive Projects Go through tutorials		
		Review IDE configuration settings Review the IDE's most fiercely contasted preferences         Samples Try out the samples		
		Copen an existing file Open a file from the Resystem Find out what's New Find out what is new		
		Z August show Voicone at start op		
		M. Aways show Welcome at start up		
•	In the 'Templates for New C/C++ Project'	e² New C/C++ Project — □ X		
	dialog, selecting 'Renesas RX' -> 'Renesas	Templates for New C/C++ Project		
	CC-RX C/C++ Executable Project'.			
•	Click 'Next'.			
		All GCC for Renesas RX C/C++ Executable Project Make A C/C++ Executable Project for Renesas RX using		
		Renesas Debug the GCC for Renesas RX Toolchain.		
		GCC for Renesas RX C/C++ Library Project		
		the GCC for Renesas RX Toolchain.		
		Renesas CC-RX C/C++ Executable Project		
		Renesas CCRX toolchain.		
		Renesas CC-RX C/C++ Library Project		
		the Renesas CCRX toolchain.		
		<u>Back</u> <u>Cancel</u>		
		A Back Next > Einish Cancel     Canc		



#### 3. Project Creation with e<sup>2</sup> studio

•	Enter the project name 'SC_Tutorial'. Click	e – – ×
	'Next'.	New Renesas CC-RX Executable Project
		New Renesas CC-RX Executable Project
		Project name: SC_Tutorial
		Use default location
		Location: C:\Workspace\SC_Tutorial Browse Create Directory for Project
		Choose file system: default ~
		Working sets
		Add project to working sets
		Working sets: V Select
		< Back
	In the 'Select teelphein device ? debur	e – 🗆 X
•	In the 'Select toolchain, device & debug	
	settings' dialog, select the options as shown	New Renesas CC-RX Executable Project Select toolchain, device & debug settings
	in the screenshot opposite.	
٠	In 'Toolchains' choose 'Renesas CCRX'.	Toolchain Settings
•	The R5F572NNDxBD MCU is found under	Language: O C O C++
	RX700 -> RX72N ->	Toolchain: Renesas CCRX V
		Toolchain Version: v3.02.00 V
	RX72N - 224 pin.	Manage Toolchains RTOS: None 🗸
٠	Select 'E2 Lite (RX)' from the pulldown and	RTOS Version:
	check 'Create Release Configuration' check	
	box.	Device Settings Configurations
•	Click 'Next'.	Target Device: R5F572NNDxBD
•	CIER NEXT.	Unlock Devices E2 Lite (RX)
		Endian: Little
		Project Type: Default   RX Simulator   V
		Create Release Configuration
		(?) < <u>Back</u> <u>N</u> ext > <u>F</u> inish Cancel
•	In the 'Select Coding Assistant settings'	e – – ×
	dialog, select 'Smart Configurator'.	New Renesas CC-RX Executable Project
	Click 'Next'.	Select Coding Assistant settings
	UIUN INEXL.	Smart Configurator
		Use Peripheral Code Generator
		Use FIT Module Download FIT Modules 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 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
		Configured in GUI Selected in GUI and Generated and Imported T G
		User Application Driver and Middleware Driver Code Configured in GUI and Generated MCU Hardware
		MCU Hardware ਲੈ
		? < Back Next > Einish Cancel
		Lance Lance Conce



•	Click 'Next'.	Image: Constants of Files to be Generated       Image: Constants of Files to be Generated         What kind of initialization routine would you like to create?       Image: Constants of Files to be Generated         Image: Use Renesas Debug Virtual Console       Image: Constants of Files to be Generated         Image: State of I/O Stream Buffer:       Image: Console         Image: The Constant of Files to be Generated       Image: Constant of Files to be Generated         Image: The Constant of Files to be Generated       Image: Constant of Files to be Generated         Image: The Constant of Files to be Generated       Image: Constant of Files to be Generated         Image: The Constant of Files to be Generated       Image: Constant of Files to be Generated         Image: The Constant of Files to be Generated       Image: Constant of Files to be Generated         Image: The Constant of Files to be Generated       Image: Constant of Files to be Generated         Image: The Constant of Files to be Generated       Image: Constant of Files to be Generated         Image: The Constant of Files to be Generated       Image: Constant of Files to be Generated         Image: The Constant of Files to be Generated       Image: Constant of Files to be Generated         Image: The Constant of Files to be Generated       Image: Constant of Files to be Generated         Image: The Constant of Files to be Generated       Image: Constant of Files to be Generated         Image: The Co
•	A summary dialog will appear, click 'Finish' to complete the project generation.	Image: SC-RX Executable Project         Summary of project "SC_Tutorial"         TOOLCHAIN VERSION :       v3.02.00         GENERATION FILES :
•	Wait for file generation to start.	Progress Information         Image: Smart Configurator operation in progress         Preparing startup code         Cancel
•	In future, to skip the pop-up message on the right, check the 'Remember my decision' check box and click on 'Open Perspective'. The perspective changes automatically when the Smart Configurator starts up.	e <sup>2</sup> Open Associated Perspective?       X         Image: This kind of project is associated with the Smart Configurator perspective. Do you want to open this perspective now?         Image: The state of the state



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

#### 4.1 Introduction

The Smart Configurator plug-in for the RX72N 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 RX72N. When using Smart Configurator, it provides the 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 'Generate Code' function is used to generate three code modules for each specific MCU feature selected. These code modules are named 'Config\_xxx.h', 'Config\_xxx.c', and 'Config\_xxx\_user.c', where 'xxx' is an acronym for the relevant MCU feature, for example 'S12AD'. Within these code modules, the user is then free to add custom code to meet their specific requirement. However, these files require custom code to be added between the following comment delimiters:

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

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.

Note: If code is added outside the above user code area, it will be lost if code generation is executed again with Smart Configurator.

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 in the RSK+ Web Installer (<u>https://www.renesas.com/rskrx72n/install/e2</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. Results are displayed via the virtual COM port in a terminal program and also on the PMOD display connected to the RSK+.

Following a tour of the key user interface features of Smart Configurator in the tabbed pages (board, clocks, components and pins), as well as a demonstration of building a project, the reader is guided through each of the peripheral function configuration pages and familiarised with the structure of the template code, including the process of adding their own code to the user code areas provided by the Smart Configurator



#### 4.2 **Project Configuration using Smart Configurator**

In this section, a brief tour of Smart Configurator is presented. For further details of the Smart Configurator paradigm and reference, refer to the RX Smart Configurator User's Guide: e<sup>2</sup> studio. You can download the latest document from: <u>https://www.renesas.com/smart-configurator</u>.

The Smart Configurator initial view is displayed as illustrated in Figure 4-1.

e <sup>2</sup> Workspace - SC_Tutorial/SC_Tutori	alscfg-e <sup>*</sup> studio — 🗆 X
<u>F</u> ile <u>E</u> dit <u>N</u> avigate Se <u>a</u> rch <u>P</u> rojec	t Renesas <u>Views Run Wi</u> ndow <u>H</u> elp
🔦 🐐 🔳 🔅 Debug	✓ E SC_Tutorial HardwareDebug
	Quick Access 🕴 🖽 C/C++ 📓 Smart Configurator 🕸 Debug
🎦 Project Explorer 🛛 👘 🗖	🏟 SC_Tutorialsefg 🛛 👘 🖬 👘 MCU Package 🖾 👘 🗖
E SC_Tutorial	Overview information
>  Includes	← General Information ⑦ ^
> 🗁 smc_gen	This editor allows you to modify the settings stored in configuration file (scfg)
> 💽 SC_Tutorial.c 📄 SC_Tutorial HardwareDebuدِ إن SC_Tutorial.scfg	Board Allow board and device selection
	Cocks Application under development Components Components
	Components Allow software component selection and configuration
	Pins Allow general pin configuration and pin configuration for selected software component
	Interrupt Allow general interrupt configuration and interrupt configuration for selected software component
< >	Overview Board Clocks Components Pins Interrupts    Legend
🕒 Console 🔀	🐘 🚂 🕪 🖻 🖛 🗖 🕶 🗖 🐂 🖳 Configuration Problems 🔣 Debugger Console 🕪 Smart Browser 😒 👘 🗖
Smart Configurator Output	smc_gen\general\r_cg_macrodriver.h A a · ·
M04000001: File generated:src	smc_gen\general\r_cg_userdefine.h
	smc gen/general/r cg hardware setup.c
M05000012: File generated: <u>src\</u> M05000012: File generated: <u>src\</u>	sac genty pintfg\Pin.c
M06000002: File generated:src	Sac gen/general/r sac interrupt.d
M00000002: Code generation is	smc_gen\r config\r bsp interrupt config.h successful
M03000004: File modified:src\s	mc gen\r config\r bsp config.h
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	

Figure 4-1 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 The 'Board' tabbed page

On the 'Board' tabbed page, set the board type and device type. Click the 'Board' tab and it will be displayed as shown in **Figure 4-2**.

∰ SC_Tutorial.scfg ⊠	
Device selection	🕤 🇅
Device selection	24
Board: Custom User Board Device: R5F572NNDxBD Download more boards	
Overview Board Clocks Components Pins Interrupts	

Figure 4-2 Board configuration page

#### 4.3.1 Board configuration page

Make sure that 'Custom User Board' is selected for the 'board:'.

鬱 SC_Tutor	ial.scfg 🛛				
Device s	election				
Device selection					
Board:	Board: Custom User Board				
Device:	R5F572NNDxBD				
	Download more boards				

Figure 4-3 Select board



#### 4.4 The 'Clocks' tabbed page

The 'Clocks' tabbed 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 in the r\_bsp\_config.h file in \src\smc\_gen\r\_config.

#### 4.4.1 Clocks configuration

**Figure 4-4** shows a screenshot of Smart Configurator with the Clocks configurations. 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 4-4**.



Figure 4-4 The 'Clocks' tabbed page



#### 4.5 The 'Components' tabbed page

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

∰ *SC_Tutorial.scfg 🔀					
Software component configuration					
Components					
	😜 🗟				
type filter text					
<ul> <li>✓</li></ul>					
Overview Board Clocks Components Pins Interrupts					

Figure 4-5 Components page

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

Smart Configurator supports five types of software components: Startup, Drivers, Middleware, Application and RTOS. 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 Drivers.

Click the 'Add component' 💼 icon.

🏟 SC_Tutorial.scfg 🛛				
Software component configuration				
Components $\downarrow^{a}_{Z} \models \ddagger \qquad \downarrow^{a}_{\rightarrow} \checkmark$				
	<b>1</b>			
type filter text				
🗸 🗁 Startup				
🗸 🗁 Generic				
💣 r_bsp				

Figure 4-6 Add a Software component (1)

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

omponent			×
Software Component Selection Select component from those available in list			88
All			~
All Startup			
Drivers Middleware Application RTOS			
	Component Selection nponent from those available in list All All Startup Drivers Middleware Application	Component Selection nponent from those available in list All All Startup Drivers Middleware Application	Component Selection nponent from those available in list All All Startup Drivers Middleware Application

Figure 4-7 Add a Software component (2)



#### 4.5.2 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 Figure 4-8 below then click 'Next'.

e <sup>2</sup> New C	omponent		— C	x I
	Component Selection nponent from those available	e in list		
Туре	Drivers			~
Function	All			~
Filter				
Compor	nents	Туре	Version	^
🖶 8-Bit	Timer	Code Generator	1.7.0	
🖶 Buses	s	Code Generator	1.8.0	
H Clock	Frequency Accuracy Me	Code Generator	1.8.0	
	pare Match Timer	Code Generator	2.0.0	
	plementary PWM Mode Ti		1.8.0	
	inuous Scan Mode S12AD	Code Generator	1.8.0	
	Calculator	Code Generator	1.8.0	
	Converter	Code Generator	1.8.0	
	Operation Circuit	Code Generator	1.8.0	~
<	T ( C ) "	C 1 C 1	100	>
Descriptio	only latest version on ware component provides cc CMT/CMTW and can genera		it timer with	< ~ ~
Download	d more software components	i		
<u>Configure</u>	<u>e general settings</u>			
?	< <u>B</u> ack	<u>N</u> ext > <u>F</u> inisl	h C	ancel

Figure 4-8 Select Compare Match Timer



In 'Add new configuration for selected component' dialog -> Resource, select 'CMT0' as shown in **Figure 4-9** below.

1	e <sup>2</sup> New Component	-	-		×
,	Add new configurat	on for selected component			
	Compare Match Tim	er			
	Configuration name:	Config_CMTW0			
	Resource:	CMTW0			$\sim$
		CMTW0 CMTW1 CMT0 CMT1 CMT2 CMT3			
	?	< <u>B</u> ack <u>N</u> ext > <u>Finish</u>		Can	cel

Figure 4-9 Select Resource - CMT0

Ensure that the 'Configuration name' updates to 'Config\_CMT0' as shown in **Figure 4-10** below then click 'Finish'.

e <sup>2</sup> New Component					×
Add new configuration	on for selected c	omponent		4	
					-
Compare Match Timer					
Configuration name:	Config_CMT0				
Resource:	CMT0				$\sim$
?	< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	Cance	:I

Figure 4-10 Ensure Configuration name - CMT0

In 'Config\_CMT0', configure CMT0 as shown in **Figure 4-11**. 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.

Components		Configure				
type filter text	10 T	Ocunt clock setting PCLK/8	9 O PCLK/32	○ PCLK/128	○ PCLK/512	
<ul> <li>✓ Startup</li> <li>✓ Seneric</li> <li>✓ r_bsp</li> <li>✓ Drivers</li> <li>✓ Timers</li> <li>✓ Config_CMT0</li> </ul>	^	Compare match se Interval value Register value (CN I Compare matc Priority	-	1 7499 Level 10	ms V	✓ (Actual value: 1)

Figure 4-11 Config\_CMT0 setting



Click the 'Add component' <sup>to</sup> icon. In 'Software Component Selection' dialog -> Type, select 'Drivers'. Select 'Compare Match Timer' then click 'Next'. In 'Add new configuration for selected component' dialog -> Resource, select 'CMT1' as shown in **Figure 4-12** below.

e <sup>2</sup> New Component			×
Add new configuration	n for selected component		
Compare Match Time			
Configuration name:	Config_CMTW0		
Resource:	CMT1		$\sim$
	СМТW0 СМТW1 СМТО СМТО		
	CMT2 CMT3		
?	< <u>B</u> ack <u>N</u> ext > <u>Finish</u>	Can	cel

Figure 4-12 Select Resource – CMT1

Ensure that the 'Configuration name' updates to 'Config\_CMT1' as shown in **Figure 4-13** below then click 'Finish'.

e <sup>2</sup> New Component		—		×
Add new configuration	on for selected component		÷	
				-
Compare Match Timer				
Configuration name:	Config_CMT1			
Resource:	CMT1			$\sim$
?	< Back Next > Finis	sh	Cancel	

Figure 4-13 Ensure Configuration name – CMT1

Navigate to the 'Config\_CMT1' and configure CMT1 as shown in **Figure 4-14**. 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 $\downarrow^{a}_{\ Z}$	¥} ▼ (	Configure	
type filter text	5	Count clock setting O PCLK/8	○ PCLK/128 ○ PCLK/512
✓	^	Compare match setting Interval value Register value (CMCOR) I Compare match interrupt (CMI1) Priority	20 ms (Actual value: 20) 37499 Level 10 V

Figure 4-14 Config\_CMT1 setting



Click the 'Add component' <sup>to</sup> icon. In 'Software Component Selection' dialog -> Type, select 'Drivers'. Select 'Compare Match Timer' then click 'Next'. In 'Add new configuration for selected component' dialog -> Resource, select 'CMT2' as shown in **Figure 4-15** below.

e <sup>2</sup> New Component		C	x c
Add new configuration	n for selected component		
Compare Match Time			
Configuration name:	Config_CMTW0		
Resource:	CMTW0		~
	CMTW0 CMTW1 CMT0 CMT1 CMT2 CMT3		
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish	C	`ancel

Figure 4-15 Select Resource – CMT2

Ensure that the 'Configuration name' updates to 'Config\_CMT2' as shown in **Figure 4-16** below then click 'Finish'.

e <sup>2</sup> New Component		—		×
Add new configuration	on for selected component			8 8 8 8
Compare Match Timer				
Configuration name:	Config_CMT2			
Resource:	CMT2			$\sim$
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish	1	Can	cel

Figure 4-16 Ensure Configuration name – CMT2

Navigate to the 'Config\_CMT2' and configure CMT2 as shown in **Figure 4-17**. 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.

Components		Configure		
type filter text	10 T	Count clock setting OPCLK/8 OPCLK/32	○ PCLK/128	
<ul> <li>✓ Startup</li> <li>✓ Generic</li> <li>✓ r_bsp</li> <li>✓ Drivers</li> <li>✓ Timers</li> <li>✓ Config_CMT0</li> <li>✓ Config_CMT1</li> </ul>	^	Compare match setting Interval value Register value (CMCOR) I Compare match interrupt (CMI2) Priority	200 ms (Actual value: 200.004267) 23437 Level 10 ~	)
Config_CMT2				

Figure 4-17 Config\_CMT2 setting



#### 4.5.3 Interrupt Controller

Referring to the RSK+ schematic, SW1 is connected to IRQ13(P45) and SW2 is connected to IRQ12(P44). SW3 is connected to IRQ15(P07) and ADTRG0n. This tutorial uses ADTRG0n, which will be configured later in §4.5.7.

Click the 'Add component' <sup>t</sup> icon. In 'Software Component Selection' dialog -> Type, select 'Drivers'. Select 'Interrupt Controller' as shown in **Figure 4-18** then click 'Next'.

e <sup>2</sup> New Component		- 🗆	
Software Component Selection Select component from those availabl	e in list		
Tura			
Type Drivers			~
Function All			$\sim$
Filter			
Components	Туре	Version	^
Event Link Controller	Code Generator	1.6.0	
General PWM Timer	Code Generator	1.5.1	
Group Scan Mode S12AD	Code Generator	1.8.0	
I2C Master Mode	Code Generator	1.8.0	
I2C Slave Mode	Code Generator	1.8.0	
H Interrupt Controller	Code Generator	2.0.0	
Low Power Consumption	Code Generator	2.0.0	_
H Normal Mode Timer	Code Generator	1.8.0	
Hold Phase Counting Mode Timer	Code Generator	2.0.0	
************************************	- · - ·	100	>
Show only latest version Description Interrupt Controller configures the in interrupt, NMI pin interrupt and IRQ		y ICU: Software	^ ~
Download more software component	5		
Configure general settings			
? < <u>B</u> ack	<u>N</u> ext > <u>F</u> inisl	h Car	ncel

Figure 4-18 Select Interrupt Controller

In 'Add new configuration for selected component' dialog -> Resource, select 'ICU' as shown in **Figure 4-19** below then click 'Finish'.

2 New Component			×
dd new configuratio	on for selected component		
Interrupt Controller			
Configuration name:	Config_ICU		
Resource:	ICU		$\sim$

Figure 4-19 Select Resource – ICU



Navigate to the 'Config\_ICU', configure these two interrupts as falling edge triggered as shown in **Figure 4-20** below.

nents	Configure	
1	Software interrupt setting	
ïlter text	Software interrupt	
→ Startup	Software interrupt 2	
🖉 🗁 Generic	Priority Level 15 (highest)	
err r_bsp ⇒ Drivers	NMI pin interrupt setting	
<ul> <li>Drivers</li> <li>interrupt</li> </ul>	□ NMI pin interrupt Detection type Falling edge ~	Digital filter No filter 🗸 0 (MHz)
💣 Config_ICU	IRQ0 setting	
<ul> <li></li></ul>	IRQ0 Detection type Low level	Digital filter No filter V 0 (MHz)
Config_CMT1		orgital inter inte
Config_CMT2	Priority Level 15 (highest) 🗸	
	IRQ1 setting	
	□ IRQ1 Detection type Low level ~	Digital filter No filter V 0 (MHz)
	Priority Level 15 (highest) ~	
	IRQ2 setting	
	IRQ2 Detection type Low level ~	Digital filter No filter 🗸 0 (MHz)
	Priority Level 15 (highest) ~	
	IRQ3 setting	
	IRQ3 Detection type Low level ~	Digital filter No filter 0 (MHz)
	Priority Level 15 (highest) 🗸 🗸	
	IRQ4 setting	
	IRQ4 Detection type Low level ~	Digital filter No filter V 0 (MHz)
	Priority Level 15 (highest)	
	IRQ5 setting	
	□ IRQ5 Detection type Low level ∨	Digital filter No filter 0 (MHz)
	Priority Level 15 (highest) ~	
	✓ IRQ6 setting	
	□ IRQ6 Detection type Low level ~	Digital filter No filter 🗸 0 (MHz)
	Priority Level 15 (highest) ~	
	IRQ7 setting	Digital filter No filter V 0 (MHz)
	IRQ7 Detection type Low level	Digital filter No filter 0 (MHz)
	Priority Level 15 (highest) 🗸 🗸	
	IRQ8 setting	
	□ IRQ8 Detection type Low level ∨	Digital filter No filter V 0 (MHz)
	Priority Level 15 (highest) 🗸 🗸	
	IRQ9 setting	
	IRQ9 Detection type Low level	Digital filter No filter V 0 (MHz)
		(///////
	Priority Level 15 (highest) 🗸 🗸	
	IRQ10 setting	
	□ IRQ10 Detection type Low level ~	Digital filter No filter 0 (MHz)
	Priority $$ Level 15 (highest) $$ $$ $$ $$ $$	
	IRQ11 setting	
	□ IRQ11 Detection type Low level ✓	Digital filter No filter V 0 (MHz)
		erin ay
	Priority Level 15 (highest)	
	IRQ12 setting	
	RQ12 Detection type Falling edge ~	Digital filter No filter V 0 (MHz)
	Priority Level 15 (highest) $\vee$	
	IRQ13 setting	
	✓ IRQ13 Detection type Falling edge ✓	Digital filter No filter V 0 (MHz)
	Priority Level 15 (highest) ~	-
	Priority Level 13 (nignest)	
	IRQ14 setting	
	□ IRQ14 Detection type Low level ∨	Digital filter No filter V 0 (MHz)
	Priority Level 15 (highest) 🗸	
	IRO15 setting	
	IRQ15 setting IRQ15 Detection type Low level	Digital filter No filter V 0 (MHz)
		(VHZ)
	Priority Level 15 (highest) ~	

Figure 4-20 Config\_ICU setting



#### 4.5.4 Ports

Referring to the RSK+ schematic, LED0 is connected to P71, LED1 is connected to PH6, LED2 is connected to PL7 and LED3 is connected to PL6. PH3 is used as one of the LCD control lines, together with P02, PK7 and PL0.

Click the 'Add component' <sup>to</sup> icon. In 'Software Component Selection' dialog -> Type, select 'Drivers'. Select 'Ports' as shown in **Figure 4-21** then click 'Next'.

e <sup>2</sup> New C	omponent		— 🗖			
Software Component Selection						
Select cor	Select component from those available in list					
Туре	Drivers			$\sim$		
Function	All			~		
				-		
Filter						
Compo	nents	Туре	Version	^		
# 12C N	Aaster Mode	Code Generator	1.8.0			
# I2C S	lave Mode	Code Generator	1.8.0			
🖶 Inter	rupt Controller	Code Generator	2.0.0			
Low	Power Consumption	Code Generator	2.0.0			
H Norn	nal Mode Timer	Code Generator	1.8.0			
🖶 Phas	e Counting Mode Timer	Code Generator	2.0.0			
🖶 Port	Output Enable	Code Generator	1.8.0			
🖶 Ports	;	Code Generator	2.0.0			
	rammable Pulse Generator	Code Generator	1.4.0			
<	· · · · · ·		100	>		
Show	only latest version					
Descriptio	on					
Commo pins can	n features such as reading, v	onfigurations for General Pu vriting, and setting the direct tures such as open-drain out	ion of ports and			
Download	d more software component	<u>s</u>				
Configure	Configure general settings					
?	< <u>B</u> ack	<u>N</u> ext > <u>E</u> inis	h Car	ncel		

Figure 4-21 Select Ports

In 'Add new configuration for selected component' dialog -> Resource, select 'PORT' as shown in **Figure 4-22** below then click 'Finish'.

e <sup>2</sup> New Component		_	o ×
Add new configuratio	n for selected component		
Ports			
Configuration name:	Config_PORT		
Resource:	PORT		~
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish		Cancel

Figure 4-22 Select Resource – PORT



Tick the tickboxes for 'PORT0', 'PORT7', 'PORTH', PORTK' and 'PORTL' as shown in Figure 4-23 below.

Components	¦ª_Z 🕒 🕀 📫 ▼	Co	onfigure						
	ت 💕		Port selection	PORT0	PORT7	PORTH	PORTK	PORTL	
type filter text									
<ul> <li>✓</li></ul>	^		PORT0	I	D PC	RT1			
<ul> <li></li></ul>			PORT2		D PC	RT3			
✓			PORT4		D PC	RT5			
✓ ⇒ I/O Ports ♂ Config_PORT			PORT6		PC	RT7			
✓ → Timers Config_CMT0			PORT8		D PC	RT9			
Config_CMT1			PORTA		PC	RTB			
					D PC	RTD			
					PC	RTF			
					✓ PC	RTH			
					<b>∠</b> PC	RTK			
			PORTL		D PC	RTM			
					D PC	RTQ			

Figure 4-23 Select Port selection

Navigate through each of the 'PORTx' tabs, configuring these four I/O lines and LCD control lines as shown in **Figure 4-24**, **Figure 4-25**, **Figure 4-26**, **Figure 4-27** and **Figure 4-28** below. Tick the tickboxes for 'Out' and tick 'Output 1' the tickboxes except for PL0 under the 'PORTL' tab. Start with the 'PORT0' tab.

Components $\downarrow_Z^a \models \ddagger \clubsuit \bullet$	Configure
÷	Port selection PORTO PORTA PORTH PORTK PORTL
type filter text	
✓ 🔁 Startup ∧ ✓ 🗁 Generic	Apply to all       Unused GPIO     In     Out       Pull-up     CMOS output     Output 1
✓ ➢ Drivers ✓ ➢ Interrupt	P00
er Config_ICU	Output 1 Normal drive output
✓ ▷ I/O Ports Config_PORT	P01
✓ ▷ Timers Config_CMT0	Unused GPIO O In O Out Pull-up CMOS output O Output 1 Normal drive output
Config_CMT1	P02
comg_	O Unused GPIO O In O Out □ Pull-up CMOS output ✓ Output 1 Normal drive output ✓
	P03
	Unused GPIO O In O Out Pull-up CMOS output Output 1
	P05
	Unused GPI0 O In O Ut Pull-up CMOS output O Utput 1
	P07
	Unused GPIO O In O Out Pull-up CMOS output Output 1
	٢/٥
	Unused GPIO O In O Out Pull-up CMOS output O Output 1 Normal drive output
	P77
~	● Unused GPIO         O In         O Out         □ Pull-up         CMOS output         □ Output 1         Normal drive output         ✓

Figure 4-24 Select PORT0 tab



#### Select 'PORT7' tab.

Components $\downarrow_{Z}^{a} \boxdot \textcircled{P} \checkmark C$	onfigure	
10 T	Port selection PORTO PORTH PORTK PORTL	
type filter text		
✓	Apply to all     In ○ Out □ Pull-up CMOS output ∨ □ Output 1	
<ul> <li>✓ ➢ Drivers</li> <li>✓ ➢ Interrupt</li> </ul>	P70 ● Unused GPIO ◯ In ◯ Out □ Pull-up CMOS output ∨ □ Output 1 High-drive output	~
Config_ICU ✓ ▷ I/O Ports		
Config_PORT  Config_CORT  Config_CMT0	P71 O Unused GPIO O In Out Pull-up CMOS output V Output 1	
Config_CMT1	P72     Output 1 Normal drive output	~
	P73 ● Unused GPIO ○ In ○ Out □ Pull-up CMOS output ○ Output 1 High-drive output	~
	P74  Unused GPI0 O In O Out Pull-up CMOS output Output 1 Normal drive output	~
	P75	
	Output 1 Normal drive output	$\sim$
	Output 1 Normal drive output	$\sim$
	P77     Output 1 Normal drive output	~

Figure 4-25 Select PORT7 tab

#### Select 'PORTH' tab.

Components $\downarrow^a_{\mathbb{Z}} \boxdot \textcircled{H} \xrightarrow{\Rightarrow} \checkmark$	Configure
te te	Port selection PORTO PORTA PORTH PORTK PORTL
type filter text	
✓ ≥ Startup ✓ ≥ Generic ✓ r_bsp	Apply to all         Image: Unused GPIO       In       Out       Pull-up       CMOS output       Image: Output 1       Normal drive output
✓	PH0         Image: Construction of the state of the
<ul> <li>✓ Config_PORT</li> <li>✓ ⇐ Timers</li> <li>✓ Config_CMT0</li> </ul>	PH1            • Unused GPIO         Out         Ou
Config_CMT1	PH2 <ul> <li>Unused GPI0 O In O Out Pull-up CMOS output</li> <li>Output 1 Normal drive output</li> </ul>
	PH3         O Unused GPIO       In <ul> <li>Pull-up</li> <li>CMOS output</li> <li>Vormal drive output</li> <li>Normal drive output</li> </ul>
	PH4
	PH5         Image: Output GPIO       Image: Output Outp
	PH6         O Unused GPIO       In       Image: Output Outpu
	PH7         Image: Output GPIO O In O Out Pull-up CMOS output         Output 1         Normal drive output

Figure 4-26 Select PORTH tab



#### Select 'PORTK' tab.

Components $\downarrow^a_{\mathbb{Z}} \boxdot \boxdot \rightrightarrows \checkmark$	onfigure
10 T	Port selection PORT0 PORT7 PORTH PORTK PORTL
type filter text	
✓ 🦢 Startup ∧ ✓ 🍃 Generic	Apply to all         Image: Second state of the s
<ul> <li>✓ ➢ Drivers</li> <li>✓ ➢ Interrupt</li> <li>✓ Config_ICU</li> </ul>	PK0         Image: Second sec
<ul> <li>I/O Ports</li> <li>Config_PORT</li> </ul>	PK1
✓ ⇐ Timers Config_CMT0	Oursed GPIO O In O Out □ Pull-up CMOS output ∨ □ Output 1 Normal drive output ∨
<ul> <li>Config_CMT1</li> <li>Config_CMT2</li> </ul>	PK2
	PK3         Image: Construction of the second seco
	PK4  Unused GPIO O In O Out Pull-up CMOS output  V Output 1 Normal drive output
	PK5 © Unused GPIO O In O Out Pull-up CMOS output V Output 1 Normal drive output V
	PK6 © Unused GPIO O In O Out Pull-up CMOS output V Output 1 Normal drive output V
	PK7 ○ Unused GPI0 ○ In ● Out □ Pull-up CMOS output ✓ ☑ Output 1 Normal drive output ✓

Figure 4-27 Select PORTK tab

#### Select 'PORTL' tab.

mponents $\downarrow_Z^a \models \clubsuit \Rightarrow$	Configure
te te	Port selection PORT0 PORT7 PORTH PORTK PORTL
rpe filter text	
<ul> <li>✓ E Startup</li> <li>✓ E Generic</li> <li> <sup>●</sup> r_bsp      </li> </ul>	Apply to all         Image: Second conduction of the second conductine of the second conduction of the second cond
	PL0 O Unused GPIO O In Output Pull-up CMOS output Output 1 Normal drive output
✓ ▷ I/O Ports Onfig_PORT	PL1
✓ ▷ Timers 2 Config_CMT0 2 Config_CMT1	Unused GPIO O In O Out Pull-up CMOS output O Output Normal drive output
Config_CMT2	PL2
	PL3 © Unused GPIO O In O Out □ Pull-up CMOS output Output 1 Normal drive output
	PL4
	PL5 © Unused GPIO O In O Out Pull-up CMOS output Output 1 Normal drive output
	PL6 O Unused GPIO O In Out Pull-up CMOS output V Output 1 Normal drive output
	PL7
	O Unused GPIO O In  O Out □ Pull-up CMOS output ✓  Output 1 Normal drive output ✓

Figure 4-28 Select PORTL tab



#### 4.5.5 SCI/SCIF Asynchronous Mode

In the RSK+RX72N, SCI9 is connected via a Renesas RL78/G1C to provide a USB virtual COM port as shown in the schematic.

Click the 'Add component' <sup>to</sup> icon. In 'Software Component Selection' dialog -> Type, select 'Drivers'. Select 'SCI/SCIF Asynchronous Mode' as shown in **Figure 4-29** then click 'Next'.

e <sup>2</sup> New Component		— C	ı ×
Software Component Selection Select component from those availabl	a in list		-
Select component nom those available			-
Type Drivers			
Type Drivers			~
Function All			$\sim$
Filter			
Components	Туре	Version	^
🖶 Real Time Clock	Code Generator	1.5.0	
H SCI/SCIF Asynchronous Mode	Code Generator	1.8.0	
H SCI/SCIF Clock Synchronous M	Code Generator	1.8.0	
🖶 Single Scan Mode S12AD	Code Generator	2.0.0	
🖶 Smart Card Interface Mode	Code Generator	1.8.0	
🖶 SPI Clock Synchronous Mode	Code Generator	1.8.0	
H SPI Operation Mode	Code Generator	1.7.0	
🖶 Voltage Detection Circuit	Code Generator	1.8.0	
🖶 Watchdog Timer	Code Generator	1.8.0	
<			>
Show only latest version			
Description			
•			
This software component provides co processor) asynchronous mode.	onfigurations for SCI(SCIF) s	ingle(multi-	^
processory asynchronous mode.			
			~
Download more software components			
Configure general settings	•		
<u></u>			
? < Back	Next > Finis	h C	ancel
- gack	<u></u>		

Figure 4-29 Select SCI/SCIF Asynchronous Mode

In 'Add new configuration for selected component' dialog -> Work mode, select 'Transmission/Reception' as shown in **Figure 4-30** below.

uu new configuratio	on for selected component		
			-
SCI/SCIF Asynchronou	is Mode		
Configuration name:	Config_SCI0		
Work mode:	Transmission		~
Resource:	Transmission		
Resource:	Reception		
	Transmission/Reception		
	Multi-processor Transmission Multi-processor Reception		
	Multi-processor Transmission/Reception		
	Mala processor Hansmission/Acception	_	-

Figure 4-30 Select Work mode – Transmission/Reception



In 'Resource', select 'SCI9' as shown in Figure 4-31 below.

e <sup>2</sup> New Component	_		×
Add new configuratio	n for selected component		
SCI/SCIF Asynchronou	s Mode		
Configuration name:	Config_SCI0		
Work mode:	Transmission/Reception		$\sim$
Resource:	SCI0		$\sim$
	SCI0 SCI1 SCI2 SCI3 SCI4 SCI5 SCI6 SCI7 SCI8 SCI9 SCI9 SCI10 SCI10 SCI11 SCI12		
?	< Back Next > Finish	Can	cel

Figure 4-31 Select Resource – SCI9

Ensure that the 'Configuration name' updates to 'Config\_SCI9' as shown in **Figure 4-32** below then click 'Finish'.

2 New Component		– <b>D</b> X
dd new configuratio	n for selected component	
SCI/SCIF Asynchronou	s Mode	
Configuration name:	Config_SCI9	
Work mode:	Transmission/Reception	~
Resource:	SCI9	~
?	< Back Next > Fi	nish Cancel

Figure 4-32 Ensure Configuration name - Config\_SCI9



Configure SCI9 as shown in **Figure 4-33**. Ensure the 'Start bit edge detection' is set as 'Falling edge on RXD9 pin' and the 'Bit rate' is set to 19200 bps. All other settings remain at their defaults.

Components $ ^{a}_{Z} \boxdot  \Rightarrow $	Configure			
	FIFO mode setting			
🐌 🖥	Non-FIFO mode	◯ FIFO mode		
type filter text	Start bit edge detection setting			
✓ ➢ Startup ✓ ➢ Generic	O Low level on RXD9 pin	Falling edge on RXD9 pin		
er_bsp	Data length setting			
V 🗁 Drivers	○ 9 bits	8 bits	○ 7 bits	
✓	Parity setting			
✓	None	◯ Even	Odd	
Config_PORT	Stop bit length setting			
<ul> <li>Communications</li> <li>Config_SCI9</li> </ul>	<ul> <li>1 bit</li> </ul>	◯ 2 bits		
🗸 🧁 Timers	Transfer direction setting			
Config_CMT0	LSB-first	○ MSB-first		
Config_CMT2	Transfer rate setting			
	Transfer clock	Internal clock	~	
	Base clock	16 cycles for 1-bit period	v	
	Bit rate	19200	(bps) (Actual value: 19230.769, Error: 0.16%)	
	Enable modulation duty correction	19200	(bps) (Actual value: 19250.709, EITOI: 0.1078)	
	SCK9 pin function	SCK9 is not used	~	
		Sero is not used		
	Noise filter setting			
	Enable noise filter	Classic strengt divided by 1	<ul> <li>120000000 (Hz)</li> </ul>	
	Noise fliter clock	Clock signal divided by 1	(Hz)	
¥	Hardware flow control setting	0.000	0	
	None	O CTS9#	○ RT S9#	
	RTS9 output active trigger number	15	~	
	FIFO data setting			
	Transmit FIFO data trigger number	0	~	
	Receive FIFO data trigger number	8	~	
	Data match detection setting			
	Enable data match detection			
	Comparison data	0x00		
	Data handling setting			
	Transmit data handling	Data handled in interrupt service routine	~	
	Receive data handling	Data handled in interrupt service routine	~	
	Interrupt setting	1 1 45 (1-1		
	TXI9 priority		×	
	RXI9 priority	Level 15 (highest)	~	
	Enable reception error interrupt (ERI9)			
	TEI9, ERI9 priority (Group AL0)	Level 15 (highest)	~	
	Receive data ready interrupt	Receive data full interrupt (RXI)	×	
		Config SCI9 sotting		

Figure 4-33 Config\_SCI9 setting



#### 4.5.6 SPI Clock Synchronous Mode

In the RSK+RX72N, SCI7 is used as an SPI master for the Pmod LCD on the PMOD1 connector as shown in the schematic. Click the 'Add component' icon. In 'Software Component Selection' dialog -> Type, select 'Drivers'. Select 'SPI Clock Synchronous Mode' as shown in **Figure 4-34** then click 'Next'.

e <sup>2</sup> New Component		- 🗆	×		
Software Component Selection Select component from those available	e in list				
Type Drivers			$\sim$		
Function All			~		
Filter					
Components	Туре	Version	^		
Heal Time Clock	Code Generator	1.5.0			
SCI/SCIF Asynchronous Mode	Code Generator	1.8.0			
SCI/SCIF Clock Synchronous M	Code Generator	1.8.0			
H Single Scan Mode S12AD	Code Generator	2.0.0			
H Smart Card Interface Mode	Code Generator	1.8.0			
🖶 SPI Clock Synchronous Mode	Code Generator	1.8.0			
H SPI Operation Mode	Code Generator	1.7.0			
H Voltage Detection Circuit	Code Generator	1.8.0			
🖶 Watchdog Timer	Code Generator	1.8.0			
<			>		
Show only latest version Description This component provides clock syncl	hronous operation of RSPI o	r SCI (Simple SPI	^		
bus). It includes 4 transfer modes: Slave transmit/receive, Slave transmit, Master transmit/receive and Master transmit.					
Download more software components Configure general settings	5				
? < <u>B</u> ack	<u>N</u> ext > <u>F</u> inis	h Car	icel		

Figure 4-34 Select SPI Clock Synchronous Mode

In 'Add new configuration for selected component' dialog -> Operation, select 'Master transmit only' as shown in **Figure 4-35** below.

e <sup>2</sup> New Component		—		×
dd new configuratic	on for selected component			
SPI Clock Synchronou	s Mode			
Configuration name:	Config_RSPI0			
Operation:	Slave transmit/receive			~
Resource:	Slave transmit/receive Slave transmit only Master transmit/receive			
	Master transmit only			
?	< <u>B</u> ack <u>N</u> ext >	<u>F</u> inish	Ca	ncel

Figure 4-35 Select Operation – Master transmit only



In 'Resource', select 'SCI7' as shown in Figure 4-36 below.

e <sup>2</sup> New Component			—	□ ×
Add new configuratio				
SPI Clock Synchronous	Mode			
Configuration name:	Config_RSP	10		
Operation:	Master tran	smit only		$\sim$
Resource:	RSPI0			~
	RSPI0           RSPI1           RSPI2           SC10           SC11           SC12           SC13           SC14           SC15           SC16           SC17           SC18           SC19           SC110           SC111           SC112			
?	< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	Cancel

Figure 4-36 Select Resource – SCI7

Ensure that the 'Configuration name' updates to 'Config\_SCI7' as shown in **Figure 4-37** below then click 'Finish'.

<sup>2</sup> New Component		— 🗆 🗙
dd new configuratio	n for selected component	
SPI Clock Synchronous	: Mode	
Configuration name:	Config_SCI7	
Operation:	Master transmit only	~
Resource:	SCI7	~
?	< Back Next >	<u>Finish</u> Cancel

Figure 4-37 Ensure Configuration name - Config\_SCI7



Configure SCI7 as shown in **Figure 4-38**. 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.

Components $\downarrow^{a}_{Z} \models \ddagger^{*}$	Configure	
type filter text	Transfer direction setting       O LSB-first	MSB-first
<ul> <li>✓ E Startup</li> <li>✓ E Generic</li> <li> <sup>●</sup> r_bsp      </li> </ul>	Data inversion setting <ul> <li>Normal</li> </ul>	) Inverted
	Bit rate	Internal clock (SCK7 pin functions as clock output pin) v 15000 (kbps) (Actual value: 15000, Error: 0%)
✓ Communications     ✓ Config_SCI7     ✓ Config_SCI9     ✓ Config_SCI9     ✓ Config_CMT0     ✓ Config_CMT0	Enable modulation duty correction      Clock setting      Enable clock delay      []	Enable clock polarity inversion
Config_CMT1	Data handling setting Transmit data handling Interrupt setting	Data handled in interrupt service routine $\vee$
	TXI7 priority	Level 15 (highest) v Level 15 (highest) v
~	Callback function setting	

Figure 4-38 Config\_SCI7 setting



#### 4.5.7 Single Scan Mode S12AD

We will be using the S12AD in Single Scan Mode on the AN000 input, which is connected to the RV1 potentiometer output on the RSK+. The conversion start trigger will be via the pin connected to SW3. Click the 'Add component' icon. In 'Software Component Selection' dialog -> Type, select 'Drivers'. Select 'Single Scan Mode S12AD' as shown in **Figure 4-39** then click 'Next'.

e <sup>2</sup> New Component		- 🗆	×		
Software Component Selection Select component from those available in list					
Type Drivers			~		
			*		
Function All			$\sim$		
Filter					
Components	Туре	Version	^		
# Real Time Clock	Code Generator	1.5.0			
SCI/SCIF Asynchronous Mode	Code Generator	1.8.0			
SCI/SCIF Clock Synchronous M	Code Generator	1.8.0			
🖶 Single Scan Mode S12AD	Code Generator	2.0.0			
Smart Card Interface Mode	Code Generator	1.8.0	-		
SPI Clock Synchronous Mode	Code Generator	1.8.0			
BPI Operation Mode	Code Generator	1.7.0			
H Voltage Detection Circuit	Code Generator	1.8.0			
H Watchdog Timer	Code Generator	1.8.0			
<			>		
Show only latest version					
Description					
This software component provides si Converter which the analog inputs ar			^		
ascending channel order.	billing selected are convert	current entry entee in			
			$\sim$		
Download more software components					
Configure general settings					
? < <u>B</u> ack	<u>N</u> ext > <u>F</u> inisl	h Cance	el		

Figure 4-39 Select Single Scan Mode S12AD

Ensure that the 'Configuration name' is'Config\_S12AD0' as shown in Figure 4-40 below then click 'Finish'.

e <sup>2</sup> New Component		—	o x
Add new configuratio	on for selected component		
Single Scan Mode S12	AD		
Configuration name:	Config_S12AD0		
Resource:	S12AD0		~

Figure 4-40 Ensure Configuration name - S12AD0



Configure S12AD0 as shown in **Figure 4-41** and **Figure 4-42**. 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.

Components	Configure			
10 T	▼ Basic setting			
type filter text	Note			
		t use the P40 to P47, P03, P05, and P07 pins as output pins. ) to P93, PD0 to PD7, and PE0 to PE7 pins as output pins.		
	Analog input mode setting			
	Analog input channel setting AN000 AN001 AN002 AN005 AN006 AN007			
Config_PORT	Conversion start trigger setting			
Config_SCI7	Start trigger source A/D conversion start tr	igger pin v		
🗸 🗁 Timers	Interrupt setting			
Config_CMT0	Enable AD conversion end interrupt (S12ADI)	Priority Level 15 (highest) ~		
Config_CMT2	▼ Advance setting			
	Add/Average AD value setting			
	AN000 AN001 AN002 AN002 AN005 AN006 AN007			
	Self diagnosis setting			
	Mode	Unused 🗸		
~	Voltage used	0V ~		
	Disconnection detection assist setting			
	Charge setting	Unused ~		
	Period	2 ADCLK $\lor$		
	Dedicated sample hold circuit channel setting			
	AN000 AN001 AN002			
	Data registers setting			
	Data placement	Right-alignment ~		
	Automatic clearing	Disable automatic clearing $\checkmark$		
	Conversion resolution	12-bit accuracy 🗸		
	Addition/Average mode select	Addition mode ~		
	Addition count	1-time v		
	Window function setting			
	Disable	() Enable		
	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 condition matched $\qquad \lor$		

Figure 4-41 Config\_S12AD0 setting (1)



A/D comparison A setting						
Reference data 0 for comparison		0				
Reference data 1 for comparison		0				
Use comparator for AN000		Reference d	lata 0 > A/E	)-conve	rted value $\vee$	
Use comparator for AN001		Reference d	lata 0 > A/E	)-conve	rted value 🗸 🗸	
Use comparator for AN002		Reference d	lata 0 > A/E	)-conve	rted value 🗸 🗸	
Use comparator for AN003		Reference d	lata 0 > A/E	)-conve	rted value 🗸 🗸	
Use comparator for AN004		Reference d	lata 0 > A/E	)-conve	rted value 🗸	
Use comparator for AN005		Reference d	lata 0 > A/E	/D-converted value $\checkmark$		
Use comparator for AN006		Reference d	lata 0 > A/E	)-conve	rted value 🗸 🗸	
Use comparator for AN007		Reference d	lata 0 > A/E	)-conve	rted value $\vee$	
A/D comparison B setting						
Reference data 0 for comparison		0				
Reference data 1 for comparison		0				
Comparison B channel		Unused		~		
		Reference d	lata 0 > A/E	)-conve	rted value $\vee$	
Input sampling time setting						
Dedicated sample and hold circuit	0.4			(µs)	(Actual value: 0.400)	
AN000/Self-diagnosis	0.183			(µs)	(Actual value: 0.183)	
AN001	0.183			(µs)	(Actual value: 0.183)	
AN002	0.183			(µs)	(Actual value: 0.183)	
AN003	0.183			(µs)	(Actual value: 0.183)	
AN004	0.183			(µs)	(Actual value: 0.183)	
AN005	0.183			(µs)	(Actual value: 0.183)	
AN006 0.183				(µs)	(Actual value: 0.183)	
AN007 0.183				(µs)	(Actual value: 0.183)	
(Total conversion time: 0.567µs)						
Interrupt setting		Chapter -			· · · · · · · · · · · · · · · · · · ·	
Enable AD conversion compare in	terrupt A (S12				ersion compare interrupt B (S12CMPBI)	
Group BL1 priority			Level 15 (h	ignest)		

Figure 4-42 Config\_S12AD0 setting (2)


## 4.6 The 'Pins' tabbed 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.



Figure 4-43 The 'Pins' tabbed page

## 4.6.1 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 4-44 Change view to show by Hardware Resource



Select the Config\_ICU of Software Components. In the Pin Function list -> Assignment column, change the pin assignment IRQ12 to P44, IRQ13 to P45. Ensure the 'Enable' tick box of IRQ12 and IRQ13 are checked, as shown in **Figure 4-45**.

Type filter text	type filter	text (* = any stri	ng, ? = any character)			All	`
<ul> <li> </li> <li>r_bsp              </li> <li>Compare Match Timer             </li> <li>Config_CMT0             </li> <li>Config_CMT1             </li> <li>Config_ICU</li> <li>Config_ICU</li> <li>Config_ICU</li> <li>Config_PORT             </li> <li>SCU/SCIF Asynchronous Mode             </li> <li>Config_SCI9             </li> <li>Single Scan Mode S12AD             </li> <li>Config_S12AD0             </li> </ul>	Enabled	Function IRQ0 IRQ1 IRQ2 IRQ3 IRQ4 IRQ5 IRQ6 IRQ7 IRQ9 IRQ10 IRQ10 IRQ10 IRQ12 IRQ13 IRQ14 IRQ15 NMI	Assignment Not assigned	<ul> <li>Not assigned</li> <li>Vot assigned</li> </ul>	None None None None I I None None	Remarks	

Figure 4-45 Configure pin assignment - Config\_ICU

Select the Config\_SCI9 of Software Components. In the Pin Function list -> Assignment column, Ensure the 'Enable' tick box of RXD9 and TXD9 are checked and Assignment column of RXD9 is PL1 and TXD9 is PL2 as shown in **Figure 4-46**.

Type filter text	type filter f	text (* = any strin	ng, ? = any character)			All	~
<ul> <li>         r_bsp         r_bsp         Compare Match Timer         Config_CMT0         Config_CMT1         Config_CMT2         Config_ICU         Config_ICU         Config_PORT         Scl/SCIF Asynchronous Mode         Config_SCI9         SSI Clock Synchronous Mode         Config_SCI7         Single Scan Mode S12AD         Config_S12AD0         </li> </ul>	Enabled	Function CTS9# RTS9# RXD9 SCK9 TXD9	Assignment  Not assigned  Not assigned  PL1/TOC2/GTETRGB/SMISO9/SSCL9/RXD9/MOSIC  Not assigned  PL2/GTETRGC/SMOSI9/SSDA9/TXD9/MISOC/ET0	Direction None I None O	Remarks		>

Figure 4-46 Configure pin assignment - Config\_SCI9



Select the Config\_SCI7 of Software Components. In the Pin Function list -> Assignment column, Ensure the 'Enable' tick box of SCK7 and SMOSI7 are checked and Assignment column of SCK7 is PH0, SMOSI7 is PH2 as shown in **Figure 4-47**.

Software Components $ \blacksquare \Box \downarrow^{a}_{z} $	Pin Functio	n				2 🗉 🖬 è	5 Z
Type filter text	type filter	text (* = any strin	ng, ? = any character)			All	$\sim$
<ul> <li> </li> <li></li></ul>	Carlos Ca	Function SCK7 SMISO7 SMOSI7 SS7#	Assignment  PH0/TIC0/GTETRGA/SCK7/RSPCKA  Not assigned  PH2/GTETRGC/SMOSI7/SSDA7/TXD7/MISOA  Not assigned	Pin Number      N2      Not assigned      Not assigned      Not assigned	10	Remarks           - </td <td></td>	
Pin Function Pin Number Overview Board Clocks Components Pins Inter	rupts						

Figure 4-47 Configure pin assignment - Config\_SCI7

Select the Config\_S12AD0 of software components. In the Pin Function list -> Assignment column, Ensure the 'Enable' tick box of ADTRG0#, AN000, AVCC0, AVSS0, VREFH0 and VREFL0 are checked and Assignment column of AN000 is P40, ADTRG0# is P07 as shown in **Figure 4-48**.

/pe filter text	type filter t	ext (* = any stri	ng, ? = any character)			All
<ul> <li>✓ ▲ r_bsp</li> <li>● r_bsp</li> <li>✓ ▲ Compare Match Timer</li> </ul>	Enabled	Function ADTRG0# AN000	Assignment P07/IRQ15/ADTRG0# P40/IRQ8/AN000	Pin Number // E5 // D4	Direction	Remarks
Config_CMT0		AN001	Not assigned		None	
Config_CMT1		AN002	Not assigned	-	None	
Config_CMT2		AN003	Not assigned	-	None	
<ul> <li>✓ ▲ Interrupt Controller</li> <li>✓ Config_ICU</li> <li>✓ ▲ Ports</li> <li>✓ Config_PORT</li> <li>✓ ▲ SCU/SCIF Asynchronous Mode</li> </ul>		AN004	Not assigned		None	
		AN005	Not assigned	Not assigned	None	
		AN006	Not assigned	Not assigned	None	
		AN007	Not assigned	Not assigned	None	
Config_SCI9		AVCC0	AVCC0	/ B2	1	
<ul> <li>Config_SC(9)</li> <li>▲ SPI Clock Synchronous Mode</li> <li>Config_SCI7</li> <li>▲ Single Scan Mode S12AD</li> <li>Config_S12AD0</li> </ul>	$\checkmark$	AVSS0	AVSS0	/ B1	1	
		VREFH0	VREFH0	// A2	1	
		VREFLO	VREFL0	/ A3	1	
	<					

Figure 4-48 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 4-49**.

∰ *SC_Tutorial.scfg ⊠		- 8
Pin configuration		<b>()</b>

Figure	4-49	Generate	Code	Button
--------	------	----------	------	--------

The Console pane should report 'Code generation is successful', as shown **Figure 4-50** below.

Sonsole 🛛	🖳 📑 🔛 📑 🚍 🕶 🗖 🖛 🗖
Smart Configurator Output	
M04000001: File generated: <u>src\smc gen\Config S12AD0\Config S12AD0.h</u>	^
M04000001: File generated: <u>src\smc gen\Config S12AD0\Config S12AD0.c</u>	
M04000001: File generated: <u>src\smc gen\Config S12AD0\Config S12AD0 user.c</u>	
M04000001: File generated: <u>src\smc gen\general\r cg macrodriver.h</u>	
M04000001: File generated: <u>src\smc gen\general\r cg userdefine.h</u>	
M04000001: File generated: <u>src\smc gen\general\r smc entry.h</u>	
M04000001: File generated: <u>src\smc gen\general\r cg hardware setup.c</u>	
M04000001: File generated: <u>src\smc gen\general\r cg cmt.h</u>	
M04000001: File generated: <u>src\smc gen\general\r cg cmtw.h</u>	
M04000001: File generated: <u>src\smc gen\general\r cg icu.h</u>	
M04000001: File generated: <u>src\smc gen\general\r cg port.h</u>	
M04000001: File generated: <u>src\smc gen\general\r cg s12ad.h</u>	
M04000001: File generated: <u>src\smc gen\general\r cg sci.h</u>	
M04000001: File generated: <u>src\smc gen\general\r cg rspi.h</u>	
M05000012: File generated: <u>src\smc gen\r pincfg\Pin.h</u>	
M05000012: File generated: <u>src\smc gen\r pincfg\Pin.c</u>	
M06000002: File generated: <u>src\smc gen\general\r smc interrupt.c</u>	
M06000002: File generated: <u>src\smc gen\general\r smc interrupt.h</u>	
M06000002: File generated: <u>src\smc gen\r config\r bsp interrupt config.h</u>	
M0000002: Code generation is successful	
M03000004: File modified: <u>src\smc gen\r config\r bsp config.h</u>	
	~
<	>

Figure 4-50 Smart Configurator console

## 4.7 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.

✓
V 🕮 src
✓
> 🔁 Config_CMT0
> 🔁 Config_CMT1
> 🔁 Config_CMT2
> 🔁 Config_ICU
> 🔁 Config_PORT
> 🔁 Config_S12AD0
> 🔁 Config_SCI7
> 📂 Config_SCI9
> 🔁 general
> 🔁 r_bsp
> 🔁 r_config
> 🔁 r_pincfg
> C_Tutorial.c
> 🗁 trash
SC_Tutorial HardwareDebug.launch
SC_Tutorial.scfg

Figure 4-51 Generated folder structure

Switch back to the 'C/C++' perspective using the button on the top right of the e<sup>2</sup> studio workspace. Select SC\_Tutorial in the Project Explorer pane, then use 'Build Project' from the 'Project' menu or the button to build the tutorial. The project will build with no errors.



## 5. User Code Integration

In this section, the remaining application code is added to the project. Source files found in 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 into the user code area within many Smart Configurator-generated files in this project, these user code areas are 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 subsequently needs to use Smart Configurator to regenerate any of the Smart Configurator-generated code.

## 5.1 **Project Settings**

Change the optimization level of the build configuration 'HardwareDebug' before building the project. With the SC_Tutorial project selected, right- click and select [Properties], or use the shortcut keys [Alt] + [Enter] to open the Properties window.	Project Explorer       Image: Section of the section of
<ul> <li>Navigate to 'C/C++ Build -&gt; Settings -&gt;Compiler -&gt; Optimization.</li> </ul>	Settings 🗢 🔹 🔿 👻
<ul> <li>Select 'Level 0: Do not perform optimization' from the Optimization level pull-down.</li> </ul>	Configuration:       HardwareDebug [ Active ]       Manage Configurations         Tool Settings       Toolchain       Device       Build Steps       Devint       Binary Parsers       Error Parsers         Common       Optimization level       Optimization level       Level 0: Do not perform optimization       Level 1: Perform partial optimization         PIC/PID       Optimization type       Perform loop optimization       Level 1: Perform optimization       Level 1: Perform optimization         V       Source       Perform loop optimization       Depends on the optimization option       V         Solpict       Performs inline expansion automatically       Depends on the optimization option       V         Maximum increasing rate of function size       100       100
Press the 'Apply and Close' button to close Properties window.	Apply and Close Cancel



## 5.2 LCD Code Integration

API functions for the Okaya LCD display are provided with the RSK+. Refer to the Tutorial project folder created according to the Quick Start Guide procedure. Check that the following files are in the src folder:

- •ascii.c
- •ascii.h
- ·r\_okaya\_lcd.c
- ∙r\_okaya\_lcd.h

Copy these files in to the src folder below the workspace. These 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 macro define. Do not edit comment generated here \*/



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

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 declaration '#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 'main' function:

```
void main(void)
{
    /* Initialize the debug LCD */
    R_LCD_Init();
    /* Displays the application name on the debug LCD */
    R_LCD_Display(0, (uint8_t *)" RSK+RX72N ");
    R_LCD_Display(1, (uint8_t *)" Tutorial ");
    R_LCD_Display(2, (uint8_t *)" Press Any Switch ");
    while (1U)
    {
        ;
        }
}
```

Indentation is lost when the code described in this manual is pasted into the  $e^2$  studio source file. Also check that the pasted code is correct.



#### 5.2.1 SPI Code

}

The Okaya LCD display is driven by the SPI Master that was configured using Smart Configurator in §4.5.6. In the e<sup>2</sup> studio Project Tree, expand the 'src\smc\_gen\Config\_SCI7' folder and open the file 'Config\_SCI7.h' by double-clicking on it. Insert the following code in the user code area at the end of the file:

/\* Start user code for function. Do not edit comment generated here \*/ /\* Exported functions used to transmit a number of bytes and wait for completion \*/ MD\_STATUS R\_SCI7\_SPIMasterTransmit(uint8\_t \* const tx\_buf, const uint16\_t tx\_num);

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

Now, open the Config\_SCI7\_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 s\_sci7\_txdone;

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

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

```
static void r_Config_SCI7_callback_transmitend(void)
```

```
/* Start user code for r_Config_SCI7_callback_transmitend. Do not edit comment generated here */
s_sci7_txdone = TRUE;
/* End user code. Do not edit comment generated here */
```

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

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

```
* Function Name: R SCI7 SPIMasterTransmit
^{\star} Description \, : This function sends SPI7 data to slave device.
* Arguments : tx_buf -
               transfer buffer pointer
            tx num
               buffer size
* Return Value : status -
               MD OK or MD ARGERROR
 ******
MD STATUS R SCI7 SPIMasterTransmit (uint8 t * const tx buf,
                            const uint16 t tx num)
{
  MD STATUS status = MD OK;
  /* Clear the flag before initiating a new transmission */
  s sci7 txdone = FALSE;
  /* Send the data using the API */
  status = R_Config_SCI7_SPI_Master_Send(tx_buf, tx_num);
  /* Wait for the transmit end flag */
  while (FALSE == s sci7 txdone)
  {
     /* Wait */
  }
  return (status);
End of function R SCI7 SPIMasterTransmit
      *****
```

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



#### 5.2.2 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 §4.5.2. 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 gs\_one\_ms\_delay\_complete = FALSE;

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

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 \*/

gs one ms delay complete = TRUE;

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

}

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

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

```
* Function Name: R CMT MsDelay
* Description : Uses CMT0 to wait for a specified number of milliseconds
* Arguments : uint16 t millisecs, number of milliseconds to wait
* Return Value : None
           void R CMT MsDelay (const uint16 t millisec)
{
  uint16 t ms count = 0;
  do
  {
     R Config CMT0 Start();
     while (FALSE == gs_one_ms_delay_complete)
     {
       /* Wait */
     R Config CMT0 Stop();
     gs_one_ms_delay_complete = FALSE;
     ms count++;
  } while (ms count < millisec);</pre>
End of function R CMT MsDelay
                    ******
```



## 5.3 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**.

e <sup>2</sup> Properties for SC_Tutorial		– <b>D</b> X
type filter text           > Resource           Builders	Settings	\$ • • • •
✓ C/C++ Build Build Variables Environment Logging Settings	Configuration: HardwareDebug [A	ctive ]
Tool Chain Editor > C/C++ General Project References Renesas QE Run/Debug Settings	<ul> <li>Common</li> <li>CPU</li> <li>PIC/PID</li> <li>Miscellaneous</li> <li>Compiler</li> <li>Compiler</li> <li>Source</li> <li>Advanced</li> <li>Object</li> <li>List</li> <li>Optimization</li> <li>Advanced</li> <li>Output</li> <li>MISRA C Rule Check</li> </ul>	Include file directories
?	Miscellaneous	Apply and Close Cancel

Figure 5-2 Adding additional search paths

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

e <sup>2</sup> Add directory path	Κ.
Directory:	
\${workspace_loc:/\${ProjName}}	]
Add subdirectories	
OK Cancel Workspace File system	

Figure 5-3 Adding workspace search path

Close the property by clicking the 'Apply and Close' button shown in **Figure 5-2**, and when the 'Settings' dialog shown in **Figure 5-4** is appeared, click 'Yes' to finish the setting.



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+RX72N Tutorial Press Any Switch' on three lines in the LCD display.

## 5.4 Switch Code Integration

API functions for user switch control are provided with the RSK+. Refer to the Tutorial project folder created according to the Quick Start Guide procedure. Check that the following files are in the src folder:

rskrx72ndef.h

- ∙r\_rsk\_switch.c
- ∙r\_rsk\_switch.h

Copy these files in to the src folder below the workspace.

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 §4.5.2. and §4.5.3 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.

#### 5.4.1 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);

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



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
^{\star} Description \, : This function sets/clears the falling edge trigger for the
           specified ICU IRQ.
* Arguments
         : uint8 t irq_no
           uint8_t set_f_edge, 1 if setting falling edge triggered, 0 if
            clearing
* Return Value : None
                   void R_ICU_IRQSetFallingEdge (const uint8_t irq_no, const uint8_t set_f_edge)
  if (1 == set f edge)
  {
     ICU.IRQCR[irq no].BYTE |= 04 ICU IRQ EDGE FALLING;
  }
  else
  {
     ICU.IRQCR[irq no].BYTE &= (uint8 t) ~ 04 ICU IRQ EDGE FALLING;
  }
}
* End of function R_ICU_IRQSetFallingEdge
                             * * * * * * * * * * * * * * *
* Function Name: R ICU IRQSetRisingEdge
* Description : This function sets/clear the rising edge trigger for the
            specified ICU IRQ.
* Arguments
          : uint8_t irq_no
           uint8_t set_r_edge, 1 if setting rising edge triggered, 0 if
            clearing
* Return Value : None
                 void R_ICU_IRQSetRisingEdge (const uint8_t irq_no, const uint8_t set_r_edge)
  if (1 == set r edge)
  {
     ICU.IRQCR[irq no].BYTE |= 08 ICU IRQ EDGE RISING;
  }
  else
  {
     ICU.IRQCR[irq_no].BYTE &= (uint8_t) ~_08_ICU_IRQ_EDGE_RISING;
  }
* End of function R ICU IRQSetRisingEdge
               * * * *
                            *******
```

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

Open the 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\_irq13\_interrupt:

/\* Start user code for r\_Config\_ICU\_irq13\_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\_irq12\_interrupt:

/\* Start user code for r\_Config\_ICU\_irq12\_interrupt. Do not edit comment generated here \*/

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

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



#### 5.4.2 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"

/\* 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\_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();

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



## 5.4.3 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 §4.5.7 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 \*/
extern volatile uint8\_t g\_adc\_trigger;

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

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 "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 th<mark>e switch module */</mark>
    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+RX72N ");
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;
         <mark>else</mark>
         {
              /* 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:

```
*******
* Function Name : cb switch press
* Description : Switch press callback function. Sets g_adc_trigger flag.
* Argument : none
* Return value : none
        *****
static void cb_switch_press (void)
{
   /* Check if switch 1 or 2 was pressed */
  if (g_switch_flag & (SWITCHPRESS_1 | SWITCHPRESS_2))
   {
     /* set the flag indicating a user requested A/D conversion is required */
     g_adc_trigger = TRUE;
     /* Clear flag */
     g switch flag = 0 \times 0;
  }
}
* 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.
* it on
* Argument : none
* Return value : uint16_t adc value
                           static uint16 t get adc (void)
{
  /* A variable to retrieve the adc result */
  uint16 t adc result;
   /* Stop the A/D converter being triggered from the pin ADTRGOn */
  R_Config_S12AD0_Stop();
   /* Start a conversion */
  R S12AD0 SWTriggerStart();
   /* Wait for the A/D conversion to complete */
  while (FALSE == g_adc_complete)
  {
     /* Wait */
     nop();
  }
   /* 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
 ******************
```



Renesas Starter Kit+ for RX72N

```
* Function Name : lcd display adc
* Description : Converts add result to a string and displays
              it on the LCD panel.
* it on the LCD panel
* Argument : uint16_t adc result
* Return value : none
       ++++++
static void lcd_display_adc (const uint16_t adc_result)
   /* Declare a temporary variable */
   char t a;
   /\,\star\, Declare temporary character string \,\star/\,
   char t 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 = (char_t)((adc_result & 0x0F00) >> 8);
   lcd buffer[6] = (a < 0x0A) ? (a + 0x30) : (a + 0x37);
   a = (char t) ((adc result \& 0x00F0) >> 4);
   lcd_buffer[7] = (a < 0x0A) ? (a + 0x30) : (a + 0x37);
   a = (char_t) (adc_result \& 0x000F);
   lcd buffer[8] = (a < 0x0A) ? (a + 0x30) : (a + 0x37);
   /* 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\general' folder and open the file 'r\_cg\_userdefine.h' by double-clicking on it. Insert the following type define in between the user code delimiter comments as shown below.

/\* Start user code for type define. Do not edit comment generated here \*/
typedef char char\_t;
/\* End user code. Do not edit comment generated here \*/

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 ADO converter.
void R S12AD0 SWTriggerStart(void)
 IR(PERIB, INTB186) = OU;
 IEN(PERIB, INTB186) = 1U;
 S12AD.ADCSR.BIT.ADST = 1U;
}
/
End of function R_S12AD0_SWTriggerStart
*****
/****
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;
}
```

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

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 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 RV1 potentiometer line and display the result on the LCD panel. Return to this point in the Tutorial to add the UART user code.

## 5.5 Debug Code Integration

API functions for trace debugging via the RSK+ serial port are provided with the RSK+. Refer to the Tutorial project folder created according to the Quick Start Guide procedure. Check that the following files are in the src folder:

•r\_rsk\_debug.c
•r\_rsk\_debug.h

Copy these files in to the src folder below the workspace.

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_SCI9_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.

## 5.6 UART Code Integration

#### 5.6.1 SCI Code

In the e<sup>2</sup> studio Project Tree, expand the 'src\smc\_gen\Config\_SCI9' folder and open the file 'Config\_SCI9.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\_SCI9\_AsyncTransmit(uint8\_t \* const tx\_buf, const uint16\_t tx\_num);

/\* Character is used to receive key presses from PC terminal \*/
extern uint8 t g rx char;

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

Open the file 'Config\_SCI9\_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 \*/ % f(x)=0

/\* 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 s\_sci9\_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\_SCI9\_callback\_transmittend function:

static void r\_Config\_SCI9\_callback\_transmittend (void)

```
{
    /* Start user code for r_Config_SCI9_callback_transmitend. Do not edit comment generated here */
    s_sci9_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\_SCI9\_callback\_receiveend function:

```
static void r_Config_SCI9_callback_receiveend(void)
{
    /* Start user code for r_Config_SCI9_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 SCI9 receive buffer and callback function again */
    R_Config_SCI9_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 SCI9 AsyncTransmit
ш.
Description : This function sends SCI9 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 SCI9 AsyncTransmit(uint8 t * const tx buf, const uint16 t tx num)
  MD_STATUS status = MD_OK;
  /* Clear the flag before initiating a new transmission */
  s_sci9_txdone = FALSE;
  /* Send the data using the API */
  status = R Config SCI9 Serial Send(tx buf, tx num);
  /* Wait for the transmit end flag */
  while (FALSE == s sci9 txdone)
  {
     /* Wait */
  }
  return (status);
}
```

#### 5.6.2 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_witch.h"
#include "r_rsk_debug.h"
#include "Config_SCI9.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 s adc count = 0;

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+RX72N ");
   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 SCI9 receive buffer and callback function */
R_Config_SCI9_Serial_Receive((uint8_t *)&g_rx_char, 1);
    /* Enable SCI9 operations */
    R_Config_SCI9_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 s adc count */
            if (16 == (++s_adc_count))
            {
                 s adc count = 0;
            }
            /* Send the result to the UART */
            uart_display_adc(s_adc_count, adc_result);
            /* Reset the flag */
            g adc trigger = FALSE;
        }
        /\star 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 s_adc_count */
            if (16 == (++s_adc_count))
            {
                s adc count = 0;
            }
            /* Send the result to the UART */
            uart display adc(s adc count, adc result);
```

RENESAS

}

```
/* 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 UART.
 Argument : uint8_t : adc_count
            uint16_t: adc result
* Return value : none
static void uart display adc (const uint8 t adc count, const uint16 t adc result)
{
  /* Declare a temporary variable */
  char t a;
   /* Declare temporary character string */
  char_t uart_buffer[] = "ADC xH Value: xxxH\r\n";
   /\star Convert ADC result into a character string, and store in the local.
    Casting to ensure use of correct data type. */
  a = (char_t) (adc_count & 0x000F);
  uart_buffer[4] = (a < 0x0A) ? (a + 0x30) : (a + 0x37);
   a = (char t)((adc result & 0x0F00) >> 8);
  uart buffer[14] = (a < 0x0A) ? (a + 0x30) : (a + 0x37);
  a = (char_t) ((adc_result \& 0x00F0) >> 4);
  uart buffer[15] = (a < 0x0A) ? (a + 0x30) : (a + 0x37);
   a = (char_t) (adc_result \& 0x000F);
  uart buffer[16] = (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 'Project' 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 be appeared 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 SCI9 (Baudrate: 19200, Data Length: 8, Parity Bit: None, Stop Bit: 1, Flow Control: None).

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 RV1 potentiometer line and display the result on the LCD panel and send the result to the PC terminal program via the SCI9.



## 5.7 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 SCI9.h"
#include "rskrx72ndef.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 s_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+RX72N ");
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 SCI9 receive buffer and callback function */
    R_Config_SCI9_Serial_Receive((uint8_t *)&g_rx_char, 1);
    /* Enable SCI9 operations */
    R Config SCI9 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);
```

RENESAS

```
/* Increment the s adc count <mark>and display using the LEDs</mark> */
        if (16 == (++s_adc_count))
        {
            s adc count = 0;
        led display count(s adc count);
        /* Send the result to the UART */
        uart_display_adc(s_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 s adc count and display using the LEDs */
        if (16 == (++s_adc_count))
        {
            s adc count = 0;
        led display count(s adc count);
        /* Send the result to the UART */
        uart_display_adc(s_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)
{
  /* Set LEDs according to lower nibble of count parameter */
  LED0 = (uint8_t) ((count & 0x01) ? LED_ON : LED OFF);
  LED1 = (uint8_t) ((count & 0x02) ? LED_ON : LED_OFF);
  LED2 = (uint8 t) ((count \& 0x04) ? LED ON : LED OFF);
  LED3 = (uint8 t) ((count & 0x08) ? LED ON : LED OFF);
}
* End of function led_display_count
```

Select 'Build Project' from the 'Project' menu or use the <sup>5</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 s\_adc\_count in binary form.

}

#### **Debugging the Project** 6.

In the Project Explorer pane, ensure that the 'SC Tutorial' project is selected. To enter the 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		Ť.
Image: Second Secon	Name:       SC_Tutorial HardwareDebug         Main       Startup         Project:       SC_Tutorial         SC_Tutorial       C/C++ Application:         HardwareDebug/SC_Tutorial.x       Yariables         Build (if required) before launching       Build Configuration:         Build Configuration:       Select Automatically         © Enable auto build       © Disable auto build         @ Use workspace settings       Configure Workspace Settings	Browse
Filter matched 13 of 15 items	Revert	Apply
?	Debug	Close

Figure 6-1 Debug Configurations

In order to execute the project, it is necessary to change the following settings in 'Renesas GDB Hardware Debugging' -> 'SC\_Tutorial HardwareDebug' -> 'Debugger' -> 'Connection Settings'.

Set 'Power Target From The Emulator (MAX 200mA)' to 'No', set 'Extal Frequency [MHz]' and 'Operating Frequency [MHz]' to the correct frequency. (They should not use the 'Enter' key after typing in values.) These can be found from the device schematics (in the case of RSK+RX72N set the EXTAL Frequency:

24.0000, Operating Frequency: 240.000).

For more information on powering the RSK+RX72N please refer to the User's Manual.

eate, manage, and run configurations			45
eate, manage, and run comigurations			1
	[		
î 🗎 🗙   🖻 ‡⇒ ▼	Name: SC_Tutorial HardwareDebug		
/pe filter text	📄 Main 🟇 Debugger 🛛 🕨 Startup 🧤 Sou	urce) 🥅 Common)	
C C/C++ Application C C/C++ Remote Application EASE Script C GDB Hardware Debugging	Debug hardware: E2 Lite (RX) V Ta GDB Settings Connection Settings Debug	rget Device: R5F572NN	
C GDB Simulator Debugging (RH850)	✓ Clock		^
Java Applet Java Application	Main Clock Source	EXTAL	~
	Extal Frequency[MHz]	24.0000	
🚭 Launch Group	Operating Frequency [MHz]	240.000	
Launch Group (Deprecated)	Permit Clock Source Change On Writ	ing Interna Yes	~
Remote Java Application	✓ Connection with Target Board		
Renesas GDB Hardware Debugging	Emulator	(Auto)	
SC_Tutorial HardwareDebug	Connection Type	JTag	~
Renesas Simulator Debugging (RX, RL78)	JTag Clock Frequency[MHz]	6.00	~
	Fine Baud Rate[Mbps]	1.50	~
	Hot Plug	No	~
	✓ Power		
	Power Target From The Emulator (MA	AX 200mA) No	~
	Supply Voltage (V)	3.3	~
	✓ CPU Operating Mode		
	Register Setting	Single Chip	~ ~

Figure 6-2 Connection Settings

When the setting is complete, press the 'Apply' button followed by the "Close" button to close the debug configuration window.



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 like button. The debugger will stop again at the beginning of the main function. Press like again to run the code.

# 7. Additional Information

## **Technical Support**

For details on how to use $e^2$ studio, refer to the help file by opening $e^2$ studio, then		
selecting Help > Help Contents from the menu bar.	🔅 🗄 🚱 Welcome	
	🕞 🔹 🕐 Help Contents	
	Search	
	Show Contextual Help	

For information about the RX72N group microcontroller refer to 'RX72N Group User's Manual: Hardware'.

For information about the RX assembly language, refer to 'RX Family User's Manual: Software'.

## 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 GmbH.

© 2019 Renesas Electronics Europe GmbH. All rights reserved.

© 2019 Renesas Electronics Corporation. All rights reserved.



	RX72N Group
<b>REVISION HISTORY</b>	Renesas Starter Kit+ for RX72N
	Smart Configurator Tutorial Manual For e <sup>2</sup> studio

Rev.	Date		Description	
		Page	Summary	
1.00	Nov.29.19		First Edition issued	

RX72N Group Renesas Starter Kit+ for RX72N Smart Configurator Tutorial Manual For e<sup>2</sup> studio

Publication Date: Rev. 1.00 Nov.29.19

Published by: Renesas Electronics Corporation

RX72N Group



R20UT4442EG0100