

32

RX130 Group

Renesas Starter Kit Smart Configurator Tutorial Manual For CS+

RENESAS 32-Bit MCU RX Family / RX100 Series

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics www.renesas.com

Rev. 1.00 Jun 2017

Notice

- Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
- 2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
- 3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
- 4. You shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
- 5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.

- 6. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond such specified ranges.
- 7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or systems manufactured by you.
- 8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
- 9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
- 10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
- 11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
- 12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.
- (Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.
- (Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

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

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

1. Handling of Unused Pins

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

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

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

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

Access to reserved addresses is prohibited.

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

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

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

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

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

Disclaimer

By using this Renesas Starter Kit (RSK), the user accepts the following terms:

The RSK is not guaranteed to be error free, and the entire risk as to the results and performance of the RSK is assumed by the User. The RSK is provided by Renesas on an "as is" basis without warranty of any kind whether express or implied, including but not limited to the implied warranties of satisfactory quality, fitness for a particular purpose, title and non-infringement of intellectual property rights with regard to the RSK. Renesas expressly disclaims all such warranties. Renesas or its affiliates shall in no event be liable for any loss of profit, loss of data, loss of contract, loss of business, damage to reputation or goodwill, any economic loss, any reprogramming or recall costs (whether the foregoing losses are direct or indirect) nor shall Renesas or its affiliates be liable for any other direct or indirect special, incidental or consequential damages arising out of or in relation to the use of this RSK, even if Renesas or its affiliates have been advised of the possibility of such damages.

Precautions

The following precautions should be observed when operating any RSK product:

This Renesas Starter Kit is only intended for use in a laboratory environment under ambient temperature and humidity conditions. A safe separation distance should be used between this and any sensitive equipment. Its use outside the laboratory, classroom, study area or similar such area invalidates conformity with the protection requirements of the Electromagnetic Compatibility Directive and could lead to prosecution.

The product generates, uses, and can radiate radio frequency energy and may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment causes harmful interference to radio or television reception, which can be determined by turning the equipment off or on, you are encouraged to try to correct the interference by one or more of the following measures;

- ensure attached cables do not lie across the equipment
- reorient the receiving antenna
- increase the distance between the equipment and the receiver
- connect the equipment into an outlet on a circuit different from that which the receiver is connected
- power down the equipment when not in use
- consult the dealer or an experienced radio/TV technician for help NOTE: It is recommended that wherever
 possible shielded interface cables are used.

The product is potentially susceptible to certain EMC phenomena. To mitigate against them it is recommended that the following measures be undertaken;

- The user is advised that mobile phones should not be used within 10m of the product when in use.
- The user is advised to take ESD precautions when handling the equipment.

The Renesas Starter Kit does not represent an ideal reference design for an end product and does not fulfil the regulatory standards for an end product.

How to Use This Manual

1. Purpose and Target Readers

This manual is designed to provide the user with an understanding of how to use Application Leading Tool (Smart Configurator) for RX together with the CS+ 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 CS+, but does not intend to be a complete guide to software development on the RSK platform. Further details regarding operating the RX130 microcontroller may be found in the Hardware Manual and within the provided sample code.

Particular attention should be paid to the precautionary notes when using the manual. These notes occur within the body of the text, at the end of each section, and in the Usage Notes section.

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

The revision history summarizes the locations of revisions and additions. It does not list all revisions. Refer to the text of the manual for details.

The following documents apply to the RX130 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.	RSKRX130-512KB User's Manual	R20UT3921EG
Tutorial Manual	Provides a guide to setting up RSK environment, running sample code and debugging programs.	RSKRX130-512KB Tutorial Manual	R20UT3922EG
Quick Start Guide	Provides simple instructions to setup the RSK and run the first sample.	RSKRX130-512KB Quick Start Guide	R20UT3923EG
Smart Configurator Tutorial Manual	Provides a guide to code generation and importing into the CS+ IDE.	RSKRX130-512KB Smart Configurator Tutorial Manual	R20UT3924EG
Schematics	Full detail circuit schematics of the RSK.	RSKRX130-512KB Schematics	R20UT3920EG
Hardware Manual	Provides technical details of the RX130 microcontroller.	RX130 Group Hardware Manual	R01UH0560EJ

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
DVD	Digital Versatile Disc
E1/E2 Lite	Renesas On-chip Debugging Emulator
GUI	Graphical User Interface
IDE	Integrated Development Environment
IRQ	Interrupt Request
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LSB	Least Significant Bit
LVD	Low Voltage Detect
MCU	Micro-controller Unit
MSB	Most Significant Bit
PC	Personal Computer
PLL	Phase-locked Loop
Pmod™	This is a Digilent Pmod [™] Compatible connector. Pmod [™] is registered to <u>Digilent Inc.</u> Digilent-Pmod_Interface_Specification
RAM	Random Access Memory
ROM	Read Only Memory
RSK	Renesas Starter Kit
RTC	Real Time Clock
SAU	Serial Array Unit
SCI	Serial Communications Interface
SPI	Serial Peripheral Interface
TAU	Timer Array Unit
TFT	Thin Film Transistor
TPU	Timer Pulse Unit
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
WDT	Watchdog Timer

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

Table of Contents

1. Overview	8
1.1 Purpose	
1.2 Features	
2. Introduction	9
3. Project Creation with CS+	
3.1 Introduction3.2 Creating the Project	
	10
4. Smart Configurator Using the CS+	
4.1 Introduction	
4.2 Project Configuration using Smart Configurator - Ove	
4.3 Clock configuration page	
4.4 Components page	
•	
•	
	21
4.5 Pins configuration page	
4.5.1 Change pin assignment of a software compone	ent34
5. Completing the Tutorial Project	
5.1 Project Settings	
5.2 Additional Folders	
5.3 Precautions on using Smart Configurator	
5.4 LCD Code Integration	
	45
	46
5.5 Switch Code Integration	
5.5.1 Interrupt Code	
5.6 Debug Code Integration	
5.7 UART Code Integration	
5.8 LED Code Integration	
6. Debugging the Project	62
7. Running the Smart Configurator Tutorial	63
7.1 Running the Tutorial	
8. Additional Information	
	• • • • • • • • • • • • • • • • • • • •

RENESAS

RSKRX130-512KB

RENESAS STARTER KIT

1. Overview

1.1 Purpose

This RSK is an evaluation tool for Renesas microcontrollers. This manual describes how to use the CS+ IDE Smart Configurator to create a working project for the RSK platform.

1.2 Features

This RSK provides an evaluation of the following features:

- Project Creation with CS+
- Code Generation using the Smart Configurator.
- 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 for the RX family together with the CS+ IDE to create a working project for the RSK platform. The tutorials help explain the following:

- Project generation using the CS+
- Detailed use of the Smart Configurator in for CS+
- Integration with custom code
- Building the project CS+

The project generator will create a tutorial project with three selectable build configurations:

- 'DefaultBuild' is a project with debug support and optimisation level set to two.
- 'Debug' 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' options 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 CS+ debugger, compiler toolchains or the E2 emulator Lite. Please refer to the relevant user manuals for more indepth information.



3. Project Creation with CS+

3.1 Introduction

In this section the user will be guided through the steps required to create a new C project for the RX130 MCU, ready to generate peripheral driver code using Smart Configurator. This project generation step is necessary to create the MCU-specific project and debug files.

3.2 Creating the Project

To use the program, start CS+:

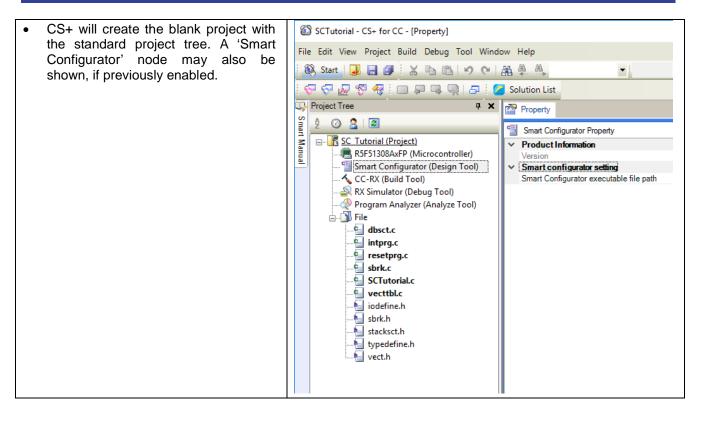
Windows[™] 7: Start Menu > All Programs > Renesas Electronics CS+ > CS+ for CC (RL78,RX,RH850)

Windows[™] 8.1 & 8: From Apps View , click 'CS+ for CC (RL78,RX,RH850)' icon

Windows[™] 10: Start Menu > All Apps > Renesas Electronics CS+ > CS+ for CC (RL78,RX,RH850)

•	CS+ will show the Start Page. Use the 'GO' button to Create a New Project.	Create New Project A new project can be created. A new project can also be created by reusing the file configuration registered to an existing project.
•	In the 'Create Project' dialog, select 'RX' from the 'Microcontroller' pull- down.	Create Project X Microcontroller: RX
•	In the 'Using Microcontroller' list control, scroll down to 'RX130' and expand the tree control by clicking '+'. Select 'R5F51308AxFP(100pin)'. Ensure that in the 'Kind of project' pull-	Using microcontroller: Update Update Product Name:R5F51308AxFP On-chip ROM size[Kbytes]:512 On-chip RAM size[Kbytes]:49152 Additional Information:Package=PLQP0100KB-B
•	down, 'Application(CC-RX)' is selected. Choose an appropriate name and location for the project, then click 'Create'.	R5F51308AxFL(48pin) R5F51308AxFN(80pin) R5F51308AxFP(100pin) R5F51308AxNE(48pin)
	Note: this tutorial assumes the project is named and located at the place shown opposite.	Kind of project: Application(CC-RX) Project name: SC_Tutorial
•	If the folder entered cannot be found a 'Question' dialog will be displayed; click 'Yes'.	Place: C:\Workspace Browse Make the project folder C:\Workspace\SCTutorial.SCTutorial.mtpj
		Pass the file composition of an existing project to the new project Project to be passed: (Input project file to be diverted.) Browse Copy composition files in the diverted project folder to a new project folder.
		Create Cancel Help







4. Smart Configurator Using the CS+

4.1 Introduction

The Smart Configurator for the RX130 has been used to generate the sample code discussed in this document. Smart Configurator for CS+ is a tool for generating template 'C' source code and project settings for the RX130. When using Smart Configurator, it supports user with a visual way of configuring the target device, clocks, software components, hardware resources and interrupts for the project. Thereby bypassing the need in most cases to refer to sections of the Hardware Manual.

By following the steps detailed in this tutorial, the user will generate a CS+ project called SC_Tutorial. A fully completed Tutorial project is contained on the DVD and may be imported into CS+ 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 CS+.

Once the user has configured the project, the 'Smart Configrator' function is used to generate code modules for each specific MCU feature selected, general folder, r_bsp folder, r_config folder and r_pincfg folder. These code modules are name 'Config_xxx.h', 'Config_xxx.c', and 'Config_xxx_user.c', where 'xxx' is an acronym for the relevant MCU feature, for example 'S12AD'. Within these code modules, the user is then free to add custom code to meet their specific requirement. Custom code should be added, whenever possible, in between the following comment delimiters:

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

Smart Configurator will locate these comment delimiters, and preserve any custom code inside the delimiters on subsequent code generation operations. This is useful if, after adding custom code, the user needs to revisit Smart Configurator to change any MCU operating parameters.

The SC_Tutorial project uses interrupts for switch inputs, the ADC module, the 8-Bit Timer, the Compare Match Timer (CMT), the Serial Communications Interface (SCI) and uses these modules to perform A/D conversion and display the results via the Virtual COM port to a terminal program and also on the LCD display on the RSK.

Following a tour of the key user interface features of Smart Configurator in and the reader is guided through each of the peripheral function configuration dialogs in §4.2. In §5, the reader is familiarised with the structure of the template code, as well as how to add their own code to the user code areas provided by the code generator.



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

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

Smart Configurator will start up by double clicking on "Smart Configurator (Design Tool)" on the project tree. The Smart Configurator initial view is displayed as illustrated in **Figure 4-1**.

Te Smart Configurator					- 6	×
File Window Help						
					-	🗈 📓
∰ SCTutorial.scfg ⊠		c	- 8	MCU Package 🛛		- 0
Overview information		۲		Type pin function		
▼ General Information		0				
This editor allows you to modify the settings stored in configuration file (.scfg)						
Board						
Allow board and device selection						
	Application under]]			91 20	
Clocks	development				n n 37	
Allow clock configuration		-Components			5x0 8x1	
	Middleware	J			54 24	
Components Allow software component selection and configuration	Device driver RTOS			RENESAS	314	
Allow software component selection and configuration		Pins			a a	
Pins				RX130	100 N1	
Allow general pin configuration and pin configuration for selected software component				KSFS1308AXFP	81 21	
				107 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	31 34 37	
Interrupt					10 72	
Allow general interrupt configuration and interrupt configuration for selected software component						
▼ Current Configuration						
Selected board/device: R5F51308AxFP						
Selected components:			~			
Overview Board Clocks Components Pins Interrupts				▶ Legend		
	🗄 🕶 📑 🖝 🗖	Sconfiguration Problems	3			~
Smart Configurator Output		0 items				
M05000001: Pin 13 is assigned to EXTAL M05000001: Pin 11 is assigned to XTAL	^	Description		Туре		
	~					
<	>					

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 CS+ project that builds and runs without error.



4.3 Clock configuration page

Clocks configuration page configures clocks of the device selected. Clock source, frequency, PLL settings and clock divider settings can be configured for the output clocks.

4.3.1 Clocks configuration

Figure 4-2 shows a screenshot of Smart Configurator with the Clocks tab. Click on the 'Clocks' tab. Configure the system clocks as shown in the figure. In this tutorial, we are using the on-board 8 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 4-2**.

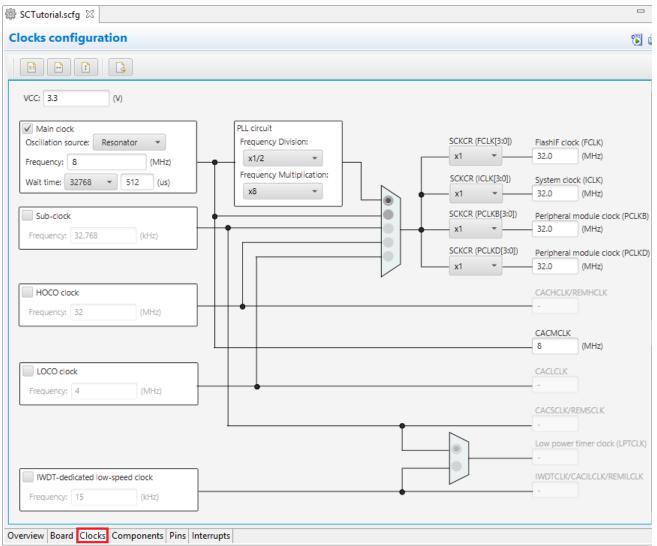


Figure 4-2 Clocks Configuration page



4.4 Components page

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

omponents	ļª _z ⊑ ⊞ ‡
	ت 😜
type filter text	
🗸 🗁 Startup	
🗸 🗁 Generic	
💣 r_bsp	
🗁 Drivers	
Middleware	
Application	

Figure 4-3 Components page

4.4.1 Add a software component into the project

Smart Configurator supports two types of software components: Code Generator and Firmware Integration Technology. In the following sub-sections, the reader is guided through the steps to configure the MCU for a simple project containing interrupts for switch inputs, timers, ADC and a SCI by component of Code Generator.

Intersection in the second sector is a second sector in the second sector is a second sector in the second sector is a second secon	
Software component	t configuration
Components	
	1
type filter text	
🗸 🗁 Startup	
✓	
	Software component Components type filter text > E Startup > E Generic

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

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

Rew Component			
Software Component Selection Select component from those available	le in list		
Function All			~
Type All			~
Filter Firmware Integration Techr	nology		
Code Generator			
Components	Туре	Version	^
🖶 8-Bit Timer	Code Generator	1.0.0	
🖶 Buses	Code Generator	1.0.0	
H Clock Frequency Accuracy Me	Code Generator	1.0.0	
# Comparator	Code Generator	1.0.0	

Figure 4-5 Add a Code Generator component (2)



4.4.2 8-Bit Timer

TMR0 will be used as an interval timer for generation of accurate delays. Select '8-Bit Timer' as shown in **Figure 4-6** below then click 'Next'.

New Component		— C	x נ		
Software Component Selection Select component from those available in list					
Function All Type Code Generator Filter			~		
Components	Туре	Version	^		
🖶 8-Bit Timer	Code Generator	1.0.0			
🖶 Buses	Code Generator	1.0.0	_		
H Clock Frequency Accuracy Me	Code Generator	1.0.0			
# Comparator	Code Generator	1.0.0			
🖶 Compare Match Timer	Code Generator	1.2.0			
🖶 Complementary PWM Mode Ti	Code Generator	1.1.0	~		
<			>		
✓ Show only last version Description					
This software component generates t timer (TMR) module that comprise tw channels.			~		
Download more software component: Configure general settings	2				
? < Back	Next > Finis	h C	ancel		

Figure 4-6 Select 8-Bit Timer

In 'Add new configuration for selected component' dialog -> Resource, select 'TMR0' as shown in **Figure 4-7** below then click 'Finish'.

New Component			
Add new configuration	for selected component	#	
8-Bit Timer			
Configuration name:	Config_TMR0]
Count mode:	8 bit	~	
Resource:	TMR0	~	
	TMR0 TMR1 TMR2 TMR3		
?	< Back Next > Finish	Cancel	

Figure 4-7 Select Resource - TMR0



In the 'Config_TMR0' configure TMR0 as shown in **Figure 4-8**. This timer is configured to generate a high priority interrupt every 1ms. We will use this interrupt later in the tutorial to provide an API for generating high accuracy delays required in our application.

ser scrutonaliscig a				
Software component configuration				
Components $\downarrow^a_{\mathbb{Z}} \boxdot \textcircled{\blacksquare} \xrightarrow{\Rightarrow} \checkmark$	Configure			
type filter text <th>Count setting Clock source Counter clear Compare match A value (TCORA)</th> <th>PCLK/1024 Cleared by compare match A</th> <th>~</th> <th>31.25 (kHz) (Actual value: 0.992000)</th>	Count setting Clock source Counter clear Compare match A value (TCORA)	PCLK/1024 Cleared by compare match A	~	31.25 (kHz) (Actual value: 0.992000)
 ✓ c_bsp ✓ brivers ✓ brivers ✓ brivers ✓ Config_TMR0 	Compare match B value (TCORB) TMO0 output setting Enable TMO0 output	1 ms		(Actual value: 0.992000)
i Middleware i Application	Output at compare match A Output at compare match B	No change No change	~	
	Interrupt setting Enable TCORA compare match interrupt (CMIA0) Enable TCORB compare match interrupt (CMIB0)			
	Enable TCNT overflow interrupt (OVI0) Priority	Level 10	~	

Figure 4-8 Config_TMR0 setting

4.4.3 Compare Match Timer

CMT0 and CMT1 will be used as timers in de-bouncing of switch interrupts.

Click 'Add component' ^t icon. In 'Software Component Selection' dialog -> Type, select 'Code Generator'. Select 'Compare Match Timer' as shown in **Figure 4-9** then click 'Next'.

😵 New Component				×
Software Component Selection Select component from those available	e in list		-	8 8 8 8
Function All				~
Type Code Generator				\sim
Filter				
Components	Туре	Version		^
Clock Frequency Accuracy Me	Code Generator	1.0.0		
+ Comparator	Code Generator	1.0.0		
H Compare Match Timer	Code Generator	1.2.0		
🖶 Complementary PWM Mode Ti	Code Generator	1.1.0		
🖶 Continuous Scan Mode S12AD	Code Generator	1.0.0		
ECRC Calculator	Code Generator	1.0.0		~
<				>
Show only last version Description This software component generates t timer (TMR) module that comprise tv channels.			bit	< >
Download more software components				
Configure general settings				
? < Back Figure 4-9 Select	Next > Finis		Canc	_

R20UT3924EG0100 Rev. 1.00 Jun 30, 2017



In 'Add new configuration for selected component' dialog -> Resource, select 'CMT0' as shown in **Figure 4-10** below then click 'Finish'.

Rew Component		—		×
Add new configuration	on for selected component			
Compare Match Time	ır —			
Configuration name:	Config_CMT0			
Resource:	CMT0			\sim
	CMT0 CMT1			
	CMIT			_
?	< Back Next > Fir	nish	Can	cel
		-		

Figure 4-10 Select Resource – CMT0

In the 'Config_CMT0' configures CMT0 as shown in **Figure 4-11**. This timer is configured to generate a high priority interrupt after 20ms. This timer is used as our short switch de-bounce timer later in this tutorial.

SCIUtorial.scrg 23						
Software component configurat	on					
Components	+i} ▼	Configure				
type filter text	•	Count clock setting PCLK/8 Compare match setting Interval value Register value (CMCOR) Enable compare match interrupt (CMI0) Priority	O PCLK/128 20 19999 Level 10	○ PCLK/512	ms	(Actual value: 20.00000)

Figure 4-11 Config_CMT0 setting

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

Rew Component		-		×
Add new configuration	for selected component			
Compare Match Timer				
Configuration name:	Config_CMT01			
Resource:	СМТ0			\sim
	CMT0 CMT1			
?	< Back Next > Finish		Car	ncel

Figure 4-12 Select Resource – CMT1



Navigate to the 'Config_CMT1' and configure CMT1 as shown in **Figure 4-13**. This timer is configured to generate a high priority interrupt after 200ms. This timer is used as our short switch de-bounce timer later in this tutorial.

🔅 *SCTutorial.scfg 🛛	
Software component configuration	
$\begin{array}{ c c c c } \hline \textbf{Components} & \downarrow^a_{\mathcal{Z}} & \boxdot & \textcircled{\blacksquare} & \textcircled{\Rightarrow} & \checkmark \\ \hline \end{array}$	Configure
type filter text ✓ ➢ Startup ✓ ➢ Generic ☞ r_bsp ✓ ➢ Drivers ✓ ➢ Timers ☞ Config_CMT1 ☞ Config_CMT1	Count clock setting PCLK/32 PCLK/128 PCLK/512 Compare match setting Interval value 200 ms (Actual value: 200.000000) Register value (CMCOR) 12499 Interval value Interval value PCLK/128 PCLK/512 Priority Level 10 V V V V V

Figure 4-13 Config_CMT1 setting

4.4.4 Interrupt Controller

Referring to the RSK schematic, SW1 is connected to IRQ1(P31) and SW2 is connected to IRQ2 (P32). SW3 is connected IRQ6(P16) and the ADTRG0n. Tutorial used ADTRG0n and will be configured later in §4.4.8.

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

elect component from those avail	able in list		
unction All			```
ype Code Generator			×
ilter			
Components	Туре	Version	^
HI2C Master Mode	Code Generator	1.0.0	
I2C Slave Mode	Code Generator	1.0.0	
🖶 Interrupt Controller	Code Generator	1.2.0	
🖶 Low Power Consumption	Code Generator	1.2.0	
🖶 Low Power Timer	Code Generator	1.0.0	
🖶 Normal Mode Timer	Code Generator	1.0.0	~
<			>
Show only last version escription This software component generat timer (TMR) module that compris	es two units (unit 0, unit 1) e two 8-bit counter channe		t A
channels.			~

Figure 4-14 Select Interrupt Controller



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

📧 New Component		- D X
Add new configuration fo	or selected component	
Interrupt Controller		
Configuration name:	Config_ICU	
Resource:	ICU	~
? < 6	Back Next > Finis	h Cancel
Figure 4	-15 Select resource	e – ICU

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

oftware component configurat							
• • 2 • • •	-+r	ng					
type filter text	Software interrupt	Priority	Level 15 (highest)	\sim			
✓ ➢ Startup	NMI pin interrupt settin	g					
✓	NMI pin interrupt	Detection type	Falling edge	✓ Digital filter	No filter $~~\vee~$	0	(MHz)
 ✓ ➢ Drivers ✓ ➢ Interrupt 	IRQ0 setting	-					
💣 Config_ICU	IRQ0	Detection type	Low level	✓ Digital filter	No filter $~~\vee~$	0	(MHz)
 Timers Config_CMT0 		Priority	Level 15 (highest)	\sim			
Config_CMT1	IRQ1 setting						
💣 Config_TMR0 🗁 Middleware	IRQ1	Detection type	Falling edge	 Digital filter 	No filter \sim	0	(MHz)
🗁 Application		Priority	Level 15 (highest)	\sim			
	IRQ2 setting						
	IRQ2	Detection type	Falling edge	 Digital filter 	No filter $\ \ \lor$	0	(MHz)
		Priority	Level 15 (highest)	\sim			
	IRQ3 setting						
	IRQ3	Detection type	Low level	✓ Digital filter	No filter $~~\vee~$	0	(MHz)
		Priority	Level 15 (highest)	\sim			
	IRQ4 setting						
	IRQ4	Detection type	Low level	✓ Digital filter	No filter $~~\vee~$	0	(MHz)
		Priority	Level 15 (highest)	\sim			
	IRQ5 setting						
	IRQ5	Detection type	Low level	✓ Digital filter	No filter $~~\vee~$	0	(MHz)
		Priority	Level 15 (highest)	\sim			
	IRQ6 setting						
	IRQ6	Detection type	Low level	✓ Digital filter	No filter $~~\vee~$	0	(MHz)
		Priority	Level 15 (highest)	\sim			
	IRQ7 setting			_			
	IRQ7	Detection type		✓ Digital filter	No filter \sim	0	(MHz)
		Priority	Level 15 (highest)	\sim			

Figure 4-16 Config_ICU setting



4.4.5 Ports

Referring to the RSK schematic, LED0 is connected to PD3, LED1 is connected to PD4, LED2 is connected to PE6 and LED3 is connected to PE7. P17 is used as one of the LCD control lines, together with PB2, PC2 and PC3.

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

New Component				<
Software Component Selection Select component from those availabl	e in list		₿	
Function All			~	1
Type Code Generator			~	-
Filter				
Components	Туре	Version	^	
Honor Phase Counting Mode Timer	Code Generator	1.2.0		
H Port Output Enable	Code Generator	1.0.0		
+ Ports	Code Generator	1.2.0		
🖶 PWM Mode Timer	Code Generator	1.0.0		I.
🖶 Real Time Clock	Code Generator	1.0.0		
🖶 Remote Control Signal Receiver	Code Generator	1.0.0	~	
<			>	
Show only last version Description	6 .: (C		0	_
This software component provides c Common features such as reading, w pins can be configured. Enabling fea pull-ups are also supported.	riting, and setting the direct	ion of ports	and	
Download more software component	s			
Configure general settings				
? < Back	Next > Finis	h	Cancel	

Figure 4-17 Select Ports

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

🐻 New Component			C	x i
Add new configuration	for selected component			
Ports				
Configuration name:	Config_PORT			
Resource:	PORT			\sim
?	< Back Next > Finis	h	С	ancel

Figure 4-18 Select resource – PORT



'PORT1', 'PORTB', 'PORTC', 'PORTD' and 'PORTE' tick box is checked as shown in **Figure 4-19** below.

omponents	.ª. E 🗄 🛟 ▼	Port selection	PORT1	PORTR	PORTC	PORTD	PORTE
	5 5						
type filter text		PORT0		∠ PO	RT1		
 Generic 		PORT2		D PO	RT3		
✓ r_bsp✓ ➢ Drivers		PORT4		Про	RT5		
🗸 🗁 Interrupt							
♂ Config_ICU ✓ (⇒ I/O Ports)		DORTA		∠ PO	RTB		
Config_PORT		PORTC		∠ PO	RTD		
Config_CMT		PORTE		D PO	RTH		
Config_CMT		D PORTJ					
Middleware Application							

Figure 4-19 Select resource – PORT

Navigate to the 'Ports' configure these four I/O lines and LCD control lines as shown in, **Figure 4-20**, **Figure 4-21**, **Figure 4-22**, **Figure 4-23** and **Figure 4-24** below. Ensure that the 'Output 1' tick box is checked, except PC3. Select 'PORT1' tab.

🌼 *SCTutorial.scfg 🛛	
Software component configuration	
Components	
	Port selection PORT1 PORTB PORTC PORTD PORTE
🐮 😇	
type filter text	Apply to all
🗸 🗁 Startup	Output □ Pull-up CMOS output □ Output 1 □ High-drive output Output 1 □ High-drive outp
✓ Generic	P12
 ✓ ➢ Drivers ✓ ➢ Interrupt 	Unused O In O Out Pull-up CMOS output Output 1 High-drive output
Config_ICU	P13
✓ ➢ I/O Ports ✓ Config_PORT	● Unused ○ In ○ Out □ Pull-up CMOS output ∨ □ Output 1 □ High-drive output
✓ ➢ Timers ➢ Config_CMT0	P14
Config_CMT1	Unused O In O Out Pull-up CMOS output Output 1 High-drive output
➢ Middleware	~ P15
🔁 Application	● Unused ○ In ○ Out □ Pull-up CMOS output ∨ □ Output 1 □ High-drive output
	P16
	● Unused ○ In ○ Out □ Pull-up CMOS output ∨ □ Output 1 □ High-drive output
	P17
	O Unused O In Out □ Pull-up CMOS output V Output 1 □ High-drive output

Figure 4-20 Select PORT1 tab



Select 'PORTB' tab.

omponents 👌 🖓 🕞 🕀 1	Port selection	PORT1	PORTB P	ORTC PORTE	PORTE			
type filter text	Apply to a Onused		Out	Pull-up	CMOS output	~	Output 1	High-drive output
 ✓ (⇒ Generic i _ bsp ✓ (⇒ Drivers 	PB0 © Unused	Oln	() Out	Pull-up	CMOS output	~	Output 1	High-drive outpu
 ✓ Enterrupt Config_ICU ✓ I/O Ports Config_PORT	PB1 Unused	Oln		Pull-up	CMOS output	~	Output 1	High-drive outpu
 Emers Config_CMT0 Config_CMT1 	PB2 O Unused	⊖In	Out	Pull-up	CMOS output	~	Output 1	High-drive outpu
 Config_TMR0 Middleware Application 	PB3 Unused	◯In	() Out	Pull-up	CMOS output	\sim	Output 1	High-drive outpu
	PB4	◯In	() Out	Pull-up	CMOS output	\sim	Output 1	High-drive outpu
	PB5	◯In	() Out	Pull-up	CMOS output	\sim	Output 1	High-drive outpu
	PB6 Unused	◯In	() Out	Pull-up	CMOS output	~	Output 1	High-drive outpu
	PB7 Unused	⊖ In	() Out	Pull-up	CMOS output	~	Output 1	High-drive outpu

Select 'PORTC' tab.

🌼 *SCTutorial.scfg 🛛 Software component configuration Components $\downarrow^{a}_{Z} \models \mp \Rightarrow$ Configure ت 😰 Port selection PORT1 PORTB PORTC PORTD PORTE type filter text 🗸 🗁 Startup Apply to all 🗸 🗁 Generic ● Unused ○ In ○ Out □ Pull-up CMOS output Output 1 High-drive output 💣 r_bsp ✓ → Drivers PC0-🗸 🗁 Interrupt ● Unused ○ In ○ Out □ Pull-up CMOS output ∨ Output 1 High-drive output Config_ICU ✓ ≥ I/O Ports Config_PORT PC1 🗸 🗁 Timers ● Unused ◯ In ◯ Out □ Pull-up Output 1 High-drive output CMOS output Config_CMT0 Config_CMT1 Config_TMR0 PC2 ◯ Unused ◯ In 💿 Out 🗌 Pull-up CMOS output Output 1 High-drive output 🗁 Middleware \sim Application PC3 CMOS output Output 1 High-drive output \sim PC4 ◯ Unused ◯ In ◯ Out □ Pull-up CMOS output \sim Output 1 High-drive output PC5 ● Unused ○ In ○ Out □ Pull-up Output 1 High-drive output CMOS output PC6 Output 1 High-drive output ● Unused ○ In ○ Out □ Pull-up CMOS output ∨ PC7 ● Unused ○ In ○ Out □ Pull-up CMOS output Output 1 High-drive output

Figure 4-22 Select PORTC tab



Select 'PORTD' tab.

omponents 👌 🖓 🕞 🕀 🗄	🛶 🗸 Configure							
*	Port selection	PORT1	PORTB P	ORTC PORTE	PORTE			
type filter text								
🗸 🗁 Startup	Apply to	all						
✓	Unused	\bigcirc In	Out	Pull-up			Output 1	High-drive outpu
✓	PDO							
✓ ➢ Interrupt	Unused	◯In	⊖ 0ut	Pull-up	CMOS output	\sim	Output 1	High-drive outpu
Config_PORT	PD1							
 Config_CMT0 	Unused	\bigcirc In	Out	Pull-up	CMOS output	\sim	Output 1	High-drive outpu
Config_CMT1	PD2							
Config_TMR0 🗁 Middleware	Unused	\bigcirc In	⊖ Out	Pull-up	CMOS output	\sim	Output 1	High-drive outpu
🗁 Application	PD3							
	⊖ Unused	◯In	Out	Pull-up			🗹 Output 1	High-drive outpu
	PD4							
	⊖ Unused	◯In	Out	Pull-up			🗹 Output 1	High-drive output
	PD5							
	Unused	◯In	⊖ 0ut	Pull-up			Output 1	High-drive output
	PD6							
	Unused	◯In	⊖ 0ut	Pull-up			Output 1	High-drive output
	PD7							
	Unused		Out	Pull-up			Output 1	High-drive output

Select 'PORTE' tab.

Configure						
Port selection	PORT1	PORTB P	ORTC PORTE	PORTE		
		Out	Pull-up		Output 1	High-drive output
PE0 Unused	◯In	() Out	Pull-up	CMOS output	✓ Output 1	High-drive output
PE1 Unused	◯In	() Out	Pull-up	CMOS output	 ✓ Output 1 	High-drive output
PE2 Unused	◯In	() Out	Pull-up	CMOS output	✓ Output 1	High-drive output
PE3 Unused	◯In	() Out	Pull-up	CMOS output	 ✓ Output 1 	High-drive output
PE4 Unused	◯In	() Out	Pull-up		Output 1	High-drive output
PE5 Unused	◯In	() Out	Pull-up		Output 1	High-drive output
PE6 O Unused	◯In	Out	Pull-up		✓ Output 1	High-drive output
PE7	⊖ln	Out	Pull-up		Output 1	High-drive output
	Port selection Pet Unused PE0 Unused PE1 Unused PE2 Unused PE3 Unused PE4 Unused PE5 Unused PE5 Unused PE6 Unused	Port selection PORT1 Apply to all Unused Unused In PE0 Unused Unused In PE1 Unused Unused In PE2 In PE3 Unused Unused In PE4 Unused Unused In PE5 Unused Unused In PE6 Unused Unused In	Port selection PORT1 PORTB P □ Apply to all □ □ 0ut □ Unused □ In ○ Uut □ E0 □ □ 0ut □ Unused □ In ○ Out □ Unused ○ In ○ Out □ E1 □ ○ • Unused ○ In ○ Out □ PE2 □ ○ • Unused ○ In ○ Out □ PE3 □ ○ Uut □ PE4 □ ○ • Unused ○ In ○ Out □ PE5 □ □ out □ PE6 ○ ○ Out □ PE7 □ □	Port selection PORT1 PORTB PORTC PORTI Apply to all Unused In Out Pull-up PE0 In Out Pull-up PE1 In Out Pull-up PE1 In Out Pull-up PE2 In Out Pull-up PE3 In Out Pull-up PE4 In Out Pull-up PE5 In Out Pull-up PE5 In Out Pull-up PE6 Unused In Out Pull-up PE6 Unused In Pull-up PE6 VInused In In Pull-up PE7 In In Pull-up	Port selection PORT1 PORTB PORTC PORTD PORTE Apply to all Unused In Out Pull-up PE0 Quarticle Quarticle	Port selection PORT1 PORTB PORTC PORTD PORTE Apply to all Unused In Out Pull-up Output 1 PE0 Unused In Out Pull-up CMOS output Output 1 PE1 Out Pull-up CMOS output Output 1 PE1 Out Pull-up CMOS output Output 1 PE2 Ounused In Out Pull-up CMOS output Output 1 PE3 Ounused In Out Pull-up CMOS output Output 1 PE4 Out Pull-up Output 1 PE5 Out Pull-up Output 1 PE6 Out Pull-up Output 1 PE7 <

Figure 4-24 Select PORTE tab



4.4.6 SCI/SCIF Asynchronous Mode

In the RSKRX130-512KB SCI1 is connected via a Renesas RL78/G1C to provide a USB virtual COM port as shown in the schematic.

Click 'Add component' icon. In 'Software Component Selection' dialog -> Type, select 'Code Generator'. Select 'SCI/SCIF Asynchronous Mode' as shown in **Figure 4-25** then click 'Next'.

New Component				×
Software Component Selection Select component from those available	e in list			
Function All				\sim
Type Code Generator				\sim
Filter				
Components	Туре	Version		^
# Real Time Clock	Code Generator	1.0.0		
Remote Control Signal Receiver	Code Generator	1.0.0		
H SCI/SCIF Asynchronous Mode	Code Generator	1.0.0		1
SCI/SCIF Clock Synchronous M	Code Generator	1.0.0		
🖶 Single Scan Mode S12AD	Code Generator	1.2.0		
H Smart Card Interface Mode	Code Generator	1.0.0		¥
<			>	
Show only last version Description This software component provides co processor) asynchronous mode.	onfigurations for SCI(SCIF) si	ingle(multi-		^
				~
Download more software components Configure general settings	i			
? Kack	Next > Finis	h	Cancel	

Figure 4-25 Select SCI/SCIF Asynchronous Mode

In 'Add new configuration for selected component' dialog -> Work mode, select 'Transmission/Reception' as shown in **Figure 4-26** below.

🐻 New Component			—		- X
Add new configuration	for selected co	omponent			8 8 8 8
SCI/SCIF Asynchronous I	lode				
Configuration name:	Config_SCI0				
Work mode:	Transmission	I.			~
Resource:	Transmission Reception	1			
	Transmission Multi-process Multi-process Multi-process	sor Transmis: sor Receptior			
	india proces	sor mansmis.	sion, neception		
?	Back	Next >	Finish	(Cancel

Figure 4-26 Select Work mode – Transmission/Reception



RSKRX130-512KB

In 'Resource', select 'SCI1' as shown in Figure 4-27 below.

New Component			×
Add new configuration	n for selected component		
SCI/SCIF Asynchronous	Mode		
Configuration name:	Config_SCI0		
Work mode:	Transmission/Reception		\sim
Resource:	SCI0		\sim
	SCI0 SCI1 SCI5 SCI6 SCI8 SCI9 SCI12		
?	< Back Next > Finish	Can	icel

Figure 4-27 Select Resource – SCI1

Ensure that the 'Configuration name' is set to 'Config_SCI1' as shown in Figure 4-28 below then click 'Finish'.

SCI/SCIF Asynchrono	us Mode		
Configuration name:	Config_SCI1		
Work mode:	Transmission/Reception		~
Resource:	SCI1		`

Figure 4-28 SCI1 Setting tab



Configure SCI1 as shown in **Figure 4-29**. Ensure the 'Start bit edge detection' is set as 'Falling edge on RXD1 pin' and the 'Bit rate' is set to 19200 bps. All other settings remain at their defaults.

In SCTutorial.scfg ⋈			
Software component configuration			
Components $\downarrow^a_{\mathbb{Z}} \square \blacksquare \rightrightarrows \checkmark$	Configure		
1 T	Start bit edge detection setting		
type filter text	O Low level on RXD1 pin	Falling edge on RXD1 pin	
V 🔁 Startup	Data length setting		
 ✓ → Generic 	○ 9 bits	8 bits	○ 7 bits
💣 r_bsp	Parity setting		
 ✓ ➢ Drivers ✓ ➢ Interrupt 	None	⊖ Even	Odd
Config_ICU	Stop bit length setting		
✓ ▷ I/O Ports Config_PORT	I bit	◯ 2 bits	
Communications	Transfer direction setting		
Config_SCI1	LSB-first	⊖ MSB-first	
✓ ➢ Timers ➢ Config_CMT1	Transfer rate setting		
Config_CMT0	Transfer clock	Internal clock	~
Config_TMR0	Base clock	16 cycles for 1-bit period	~
Application	Bit rate	19200	 (bps) (Actual value: 19230.769, Error: 0.160%)
	Enable modulation duty correction		_
	SCK1 pin function	SCK is not used	~
	Noise filter setting		
	Enable noise filter		
	Noise filter clock	Clock signal divided by 1	32000000 (Hz)
	Hardware flow control setting		
	None	○ CTS1#	○ RTS1#
	Data handling setting		
	Transmit data handling	Data handled in interrupt service routine	~
	Receive data handling	Data handled in interrupt service routine	~
	Interrupt setting		
	Enable reception error interrupt (ERI1)		
	TXI1, RXI1, TEI1, ERI1 priority	Level 15 (highest)	~
	Callback function setting		
	Transmission end	Reception end	Reception error

Figure 4-29 Config_SCI1 setting



4.4.7 SPI Clock Synchronous Mode

In the RSKRX130-512KB SCI6 is used as an SPI master for the Pmod LCD on the PMOD1 connector as

shown in the schematic. Click 'Add component' icon. In 'Software Component Selection' dialog -> Type, select 'Code Generator'. Select 'SPI Clock Synchronous Mode' as shown in **Figure 4-30** then click 'Next'.

Rew Component			×
Software Component Selection Select component from those available	e in list		
Function All Type [©] Code Generator Filter			~
Components Single Scan Mode S12AD Smart Card Interface Mode SPI Clock Synchronous Mode SPI Operation Mode Voltage Detection Circuit Watchdog Timer Show only last version Description	Type Code Generator Code Generator Code Generator Code Generator Code Generator Code Generator	Version 1.2.0 1.0.0 1.0.0 1.0.0 1.0.0 1.0.0	^ ~ >
This component provides clock syncl bus). It includes 4 transfer modes: Sla transmit/receive and Master transmit <u>Download more software components</u> <u>Configure general settings</u>	ve transmit/receive, Slave tra		~
? < Back	Next > Finis	h Ca	ncel

Figure 4-30 Select SPI Clock Synchronous Mode

Ensure 'Operation' is set to 'Master transmit only' as shown in Figure 4-31 below.

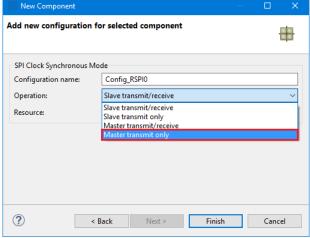


Figure 4-31 Select Operation – Master Transmit



RSKRX130-512KB

In 'Resource', select 'SCI6' as shown in Figure 4-32 below.

Rew Component			×
Add new configura	ation for selected component		
-SPI Clock Synchror	ious Mode		
Configuration nam	e: Config_RSPI0		
Operation:	Master transmit only		\sim
Resource:	RSPIO		\sim
	RSPI0 SCI0 SCI1 SCI5 SCI6 SCI6 SCI9 SCI9 SCI12		
?	< Back Next > Finish	Can	cel

Figure 4-32 Select Resource – SCI6

Ensure that the 'Configuration name' update to 'Config_SCI6' as shown in **Figure 4-33** below then click 'Finish'.

Rew Component				×
Add new configuration	n for selected compon	ent		
SPI Clock Synchronous	Mode			
Configuration name:	Config_SCl6			
Operation:	Master transmit only			\sim
Resource:	SCI6			\sim
	50.0			
?	< Back Next >	Finish	Canc	el

Figure 4-33 Ensure Configuration name - Config_SCI6



Configure SCI6 as shown in **Figure 4-34**. Ensure the 'Transfer direction' is set as 'MSB-first' and the 'Bit rate' is set to 8000 kbps. All other settings remain at their defaults.

mponents	!ª₂ 🕞 🕀 🛟 ▼	Configure	
ype filter text Startup Generic Full Config_ICU Config_ICU Config_PORT	÷ 5	Transfer direction setting C LSB-first Data inversion setting Normal Transfer speed setting Transfer clock Bit rate	MSB-first Inverted Internal clock (SCK6 pin functions as clock output pin) 8000 (kbps) (Actual value: 8000, Error: 0%
Communications Config_SCI6 Config_SCI1 Config_SCI1 Config_CMT1 Config_CMT0		Clock setting Clock delay	tion
Config_TMR0 Middleware Application		Data handling setting Transmit data handling Interrupt setting	Data handled in interrupt service routine $\qquad \lor$
		TXI6, TEI6 priority	Level 15 (highest) \vee

Figure 4-34 Config_SCI6 setting



4.4.8 Single Scan Mode S12AD

We will be using the S12AD on Single Scan Mode on the AN000 input, which is connected to the RV1 potentiometer output on the RSK. The conversion start trigger will be via the pin connected to SW3. Click

'Add component' icon. In 'Software Component Selection' dialog -> Type, select 'Code Generator'. Select 'Single Scan Mode S12AD' as shown in **Figure 4-35** then click 'Next'.

Function All			~
Type Code Generator			· · ·
ilter			
Components	Туре	Version	^
SCI/SCIF Asynchronous Mode	Code Generator	1.0.0	
SCI/SCIF Asynchronous Mode		1.0.0	
Single Scan Mode S12AD	Code Generator	1.2.0	_
Smart Card Interface Mode	Code Generator	1.0.0	_
SPI Clock Synchronous Mode	Code Generator	1.0.0	
SPI Operation Mode	Code Generator	1.0.0	
<			>
Show only last version Description This software component generates t timer (TMR) module that comprise tv channels.			it o
Download more software components Configure general settings	2		

Figure 4-35 Select Single Scan Mode S12AD

In 'Add new configuration for selected component' dialog -> Resource, select 'S12AD0' as shown in **Figure 4-36** below then click 'Finish'.

💿 New Component			
dd new configuration	for selected component		
Single Scan Mode S12AD			
Configuration name:	Config_S12AD0		
Resource:	\$12AD0		~
?	Back Next >	Finish	Cancel

Figure 4-36 Select resource – S12AD0



Configure S12AD0 as shown in **Figure 4-37** and **Figure 4-38**. 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.

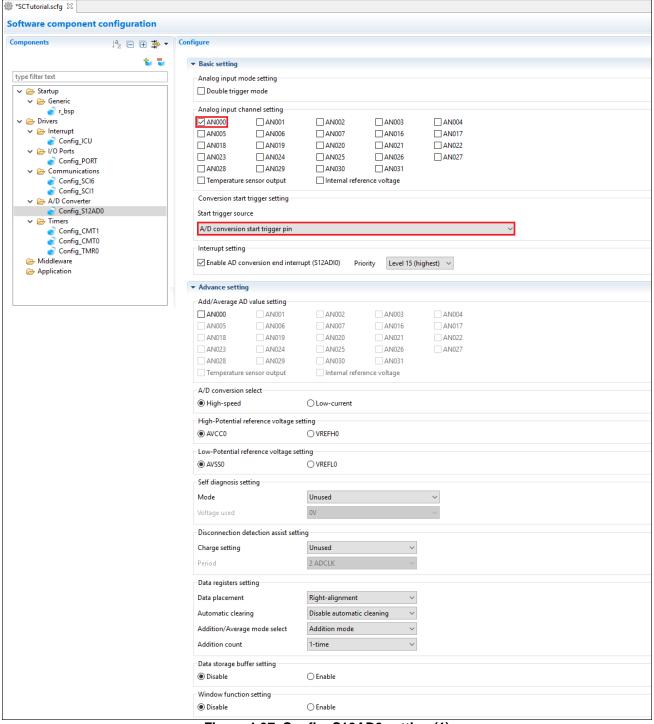


Figure 4-37 Config_S12AD0 setting (1)



4. Smart Configurator Using the CS+

Window A/B operation setting	
Enable comparison window A	Enable comparison window B
Window A/B complex condition	S12ADWMELC output when window A comparison conditions OR window B comparison conditions are met
	(S12ADWUMELC is output in other cases)
A/D comparison A setting	
Reference data 0 for comparison	0
Reference data 1 for comparison	0
Use comparator for AN000	Reference data 0 > A/D-converted value
Use comparator for AN001	Reference data 0 > A/D-converted value
Use comparator for AN002	Reference data 0 > A/D-converted value
Use comparator for AN003	Reference data 0 > A/D-converted value
Use comparator for AN004	Reference data 0 > A/D-converted value
Use comparator for AN005	Reference data 0 > A/D-converted value
Use comparator for AN006	Reference data 0 > A/D-converted value
Use comparator for AN007	Reference data 0 > A/D-converted value <
Use comparator for AN016	Reference data 0 > A/D-converted value
Use comparator for AN017	Reference data 0 > A/D-converted value
Use comparator for AN018	Reference data 0 > A/D-converted value
Use comparator for AN019	Reference data 0 > A/D-converted value
Use comparator for AN020	Reference data 0 > A/D-converted value
Use comparator for AN021	Reference data 0 > A/D-converted value
Use comparator for AN022	Reference data 0 > A/D-converted value
Use comparator for AN023	Reference data 0 > A/D-converted value v
Use comparator for AN024	Reference data 0 > A/D-converted value
Use comparator for AN025	Reference data 0 > A/D-converted value
Use comparator for AN026	Reference data 0 > A/D-converted value
Use comparator for AN027	Reference data 0 > A/D-converted value
Use comparator for AN028	Reference data 0 > A/D-converted value
Use comparator for AN029	Reference data 0 > A/D-converted value
Use comparator for AN030	Reference data 0 > A/D-converted value V
Use comparator for AN031	Reference data 0 > A/D-converted value \lor
Use comparator for Temperature	e sensor output Reference data 0 > A/D-converted value ~
Use comparator for Internal refer	rence voltage Reference data 0 > A/D-converted value ~
A/D comparison B setting	
Reference data 0 for comparison	0
Reference data 1 for comparison	0
Comparison B channel	Unused \checkmark
	Reference data 0 > A/D-converted value <
Input sampling time setting	
AN000/Self-diagnosis	0.183 (us) (Actual value: 0.188)
AN001	0.183 (us) (Actual value: 0.188)
AN002	0.183 (us) (Actual value: 0.188)
AN003	0.183 (us) (Actual value: 0.188)
AN004	0.183 (us) (Actual value: 0.188)
AN005	0.183 (us) (Actual value: 0.188)
AN006	0.183 (us) (Actual value: 0.188)
AN007	0.183 (us) (Actual value: 0.188)
AN016-AN031	0.183 (us) (Actual value: 0.188)
Temperature sensor output	0.183 (us) (Actual value: 0.188)
Internal reference voltage	0.183 (us) (Actual value: 0.188)
	(Total conversion time: 1.562us)
Event link control setting	
ELC scan end event generation con	dition On completion of all scans \vee

Figure 4-38 Config_S12AD0 setting (2)



4.5 Pins configuration page

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

🐡 *SCTutorial.scfg 🛛	
Pin configuration	
rinconiguration	
Hardware Resource	
The second se	
Type filter text	
🚣 All	^
🗱 Clock generator	
Voltage detection circuit	
€ LVD2	
Clock frequency accuracy measurement circul	it
Interrupt controller unit	
Multi-function timer pulse unit 2	
MTU0	
MTU1	
MTU2	
MTU3	
 MTU4 MTU5 	
MIOS Port output enable 2	
V (2. 8-bit timer	
TIMR0	
TMR1	
TIMR2	
TMR3	
✓ ## Serial communications interface	
SCI0	
SCI1	~
Pin Function Pin Number	
Overview Board Clocks Components Pins Interrupts	

Figure 4-39 Pin configuration page

4.5.1 Change pin assignment of a software component

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



Figure 4-40 Change view to show by Software Components



Select the Config_ICU of software component. In the Pin Function list -> Assignment column, change the pin assignment IRQ1 to P31, IRQ2 to P32. Ensure the 'Enable' tick box of IRQ1 and IRQ2 are checked, as shown in **Figure 4-41**.

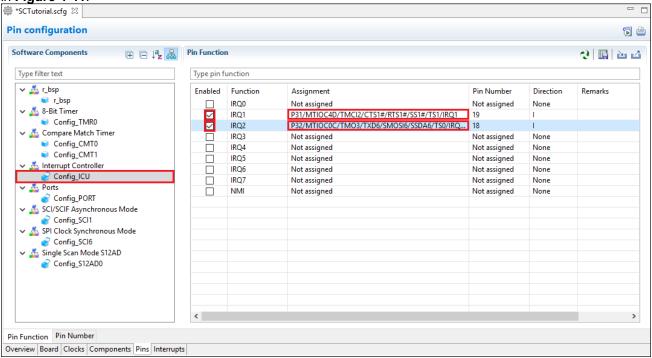


Figure 4-41 Configure pin assignment - Config_ICU

Select the Config_SCI1 of software component. In the Pin Function list -> Assignment column, Ensure the 'Enable' tick box of RXD1 and TXD1 are checked and Assignment column of RXD1 is P30 and TXD1 is P26 as shown in **Figure 4-42**.

ftware Components $\oplus = \downarrow_z^a$	Rin Function	n				- 🔁 🖬 🔁
ype filter text	Type pin f	unction				
 isp isp isp isp isp isp ispin icompare Match Timer iconfig_CMT0 iconfig_CMT1 iconfig_ICU iconfig_ICU iconfig_PORT iconfig_SCI1 iconfig_SCI1 iconfig_SCI4 iconfig_SCI6 iconfig_SCI6 iconfig_S12AD0 iconfig_S12AD0	Enabled	Function CTS1# RTS1# RXD1 SCK1 TXD1	Assignment Not assigned P30/MTIOC48/POE8#/TMRI3/RXD1/SMISO1/SSCL1/T Not assigned P26/MTIOC2A/TMO1/TXD1/SMOSI1/SSDA1/TS4	Pin Number Not assigned Not assigned 20 Not assigned 22	Direction None None I None O	Remarks

Figure 4-42 Configure pin assignment - Config_SCI1

Select the Config_SCI6 of software component. In the Pin Function list -> Assignment column, Ensure the 'Enable' tick box of SCK6 and SMOSI6 are checked and Assignment column of SCK6 is PB3, SMOSI6 is PB1 as shown in **Figure 4-43**.

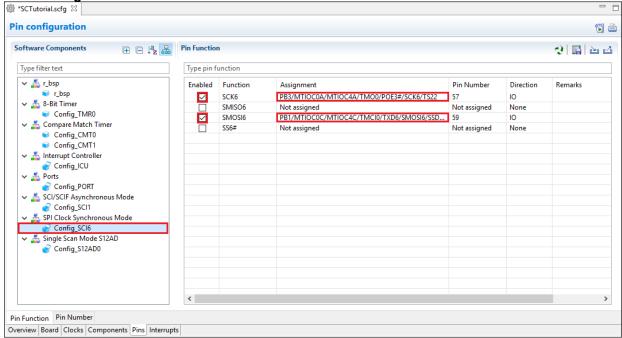


Figure 4-43 Configure pin assignment - Config_SCI6

Select the Config_S12AD0 of software component. In the Pin Function list -> Assignment column, Ensure the 'Enable' tick box of AN000, AVCC0, AVSS0 and ADTRG0# are checked and Assignment column of AN000 is P40, ADTRG0# is P16 as shown in **Figure 4-44**.

ftware Components 🛛 🕀 🖃 🚜	Pin Functio	n				-2 🖪 2
ype filter text	Type pin f	unction				
🗸 🚣 r_bsp	Enabled	Function	Assignment	Pin Number	Direction	Remarks
📦 r_bsp		ADTRG0#	P16/MTIOC3C/MTIOC3D/TMO2/TXD1/SMOSI1/SSDA1	30	1	
🗸 🚣 8-Bit Timer		AN000	P40/AN000	95	1	
Config_TMR0		AN001	Not assigned	Not assigned	None	
🗸 💑 Compare Match Timer		AN002	Not assigned	Not assigned	None	
Config_CMT0		AN003	Not assigned	Not assigned	None	
Config_CMT1		AN004	Not assigned	Not assigned	None	
🗸 🚣 Interrupt Controller		AN005	Not assigned	Not assigned	None	
Config_ICU		AN006	Not assigned	Not assigned	None	
🗸 🚣 Ports		AN007	Not assigned	Not assigned	None	
Config_PORT		AN016	Not assigned	Not assigned	None	
SCI/SCIF Asynchronous Mode		AN017	Not assigned	Not assigned	None	
Config_SCI1		AN018	Not assigned	Not assigned	None	
SPI Clock Synchronous Mode		AN019	Not assigned	Not assigned	None	
Config_SCI6		AN020	Not assigned	Not assigned	None	
🗸 🚣 Single Scan Mode S12AD		AN021	Not assigned	Not assigned	None	
Config_S12AD0		AN022	Not assigned	Not assigned	None	
		AN023	Not assigned	Not assigned	None	
		AN024	Not assigned	Not assigned	None	
		AN025	Not assigned	Not assigned	None	
		AN026	Not assigned	Not assigned	None	
		AN027	Not assigned	Not assigned	None	
		AN028	Not assigned	Not assigned	None	
		AN029	Not assigned	Not assigned	None	
		AN030	Not assigned	Not assigned	None	
		AN031	Not assigned	Not assigned	None	
		AVCC0	AVCC0	97	1	
		AVSS0	AVSS0	99	1	
		VREFH0	Not assigned	Not assigned	None	
		VREFL0	Not assigned	Not assigned	None	
	<			-		

Figure 4-44 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 4-45**.



The Console pane should report 'Code generation is successful', as shown **Figure 4-46** below.

ne echecie parle chedia report ecae generation le caececerat, ac chemi			
E Console 🛛	🖹 🛃 🔛 🚽 🖃 י	- 📬 - 🗖	
Smart Configurator Output			
M04000001: File generated:src\smc_gen\general\r_cg_cmt.h			~
M04000001: File generated:src\smc_gen\general\r_cg_riic.h			
M04000001: File generated:src\smc_gen\general\r_cg_doc.h			
M04000001: File generated:src\smc_gen\general\r_cg_tmr.h			
M04000001: File generated:src\smc_gen\general\r_cg_crc.h			
M04000001: File generated:src\smc_gen\general\r_cg_lvd.h			
M04000001: File generated:src\smc_gen\general\r_cg_cmpb.h			
M04000001: File generated:src\smc_gen\general\r_cg_elc.h			
M04000001: File generated:src\smc_gen\general\r_cg_lpt.h			
M04000001: File generated:src\smc gen\general\r_cg bsc.h			
M05000012: File generated:src\smc_gen\r_pincfg\Pin.h			
M05000012: File generated:src\smc_gen\r_pincfg\Pin.c			
M06000002: File generated:src\smc_gen\general\r_smc_interrupt.c			
M06000002: File generated:src\smc_gen\general\r_smc_interrupt.h			
M0000002: Code generation is successful			
M03000004: File modified:src\smc_gen\r_config\r_bsp_config.h			
			\sim
<		3	>

Figure 4-46 Smart Configurator console

When code generation is executed, the startup files generated at the time of CS+ project creation are replaced with those generated by Smart Configurator. **Figure 4-47** the project tree after code generation. In the next chapter, user code is added to these files, and SC_Tutorial is completed by adding a new source file to the project.

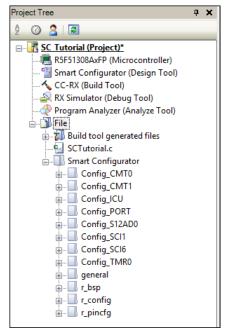


Figure 4-47 Smart Configurator folder structure

5. Completing the Tutorial Project

5.1 Project Settings

•	In the 'Project Tree' pane, select 'CC-RX (Build Tool)'. The build properties will appear in the main window. CS+ creates a single build configuration called 'Default Build' for the project. This has standard code optimisation turned on by default.	CC-RX Property ■ Build Mode DefaultBuild Change property value for all build modes at once No ■ Call mode DefaultBuild Change property value for all build modes at once No ■ Call mode RXv2 architecture(isa=nv2) Uses floating-point operation instructions Yes(fpu) Endian type for data Little endian data(endian=little) Rounding method for floating-point constant operations round to nearest(round-nearest) Handling of denormalized numbers in floating-point constants Handles as seros(-denomalize-off) Precision of the double type and long double type Handles as unsigned char(unsigned_char) Sign of the char type No Sign of the char type Handles as unsigned (unsigned_char) Sign of the char type Allocates from night(bit_order=right) Assumes the boundary alignment value for structure members is 1 No(unpack) Enables the C++ exceptional handling function (tyru, catch and throw) No(nexception) Enables the C++ exceptional handling function (tyru catch and typeid) No(fitt=off) General registers used only in fast interrupt functions None(first_register=0) Base register for ROM None Base register fo
•	Select the 'Compile Options' tab at the bottom of the properties window pane. Under 'Language of the C source file' select 'C99(-lang=c99)' as shown opposite.	CC-RX Property CC-RX Property Source Language of the C source file C39(Lang=c39) Language of the C++ source file C++(lang=c39) Language of the C++ source file System include paths[0] lnvalidates the predefined macro Enables information-level messages output No(nomessage) Suppresses the number of information-level messages Changes the information-level messages to information-level messages No Changes the information-level and warning-level messages No Changes the information-level and warning-level messages to information-level message No Changes the information-level and warning-level messages to information-level messages No Changes the information-level and warning-level messages to error-level mess. No Permits comment (/**) nesting No(comment=nonest)
•	Select the 'Link Options' tab at the bottom of the properties window pane. Under 'Section -> ROM to RAM mapped section', add the three mappings as shown opposite.	Section start address B_1.R_1.B_2.R_2.B.R.SU.SI/04.PResetPRG/0FFE00000.C_1.C_2.C.CS The specified section that outputs externally defined symbols to the file Section alignment ROM to RAM mapped section ROM to RAM mapped section[3] Output
	These settings are easily added by clicking the button '' and pasting the following text into the dialog: R 1=R_1 2=R_2 This ensures that the linker assigns RAM rather than ROM addresses to C variables. Click 'OK'	Text Edit × Iext: D=R D_1=R_1 ////////////////////////////////////



RSKRX130-512KB

5. Completing the Tutorial Project

•	From the 'Build' menu, select 'Build Mode Settings'. Click 'Duplicate' and in the resulting 'Character String Input' dialog, enter 'Debug' for the name of the duplicate Build Mode.	SCTutorial - CS+ for CC - [Property] File Edit View Project Build Project Start Start <td< th=""></td<>
•	The new 'Debug' Build Mode will be added to the Build Mode list. Click 'Close'. Now, in the main CC-RX Property window, under the 'Common Options' tab, click on the line containing 'Build Mode', click the pull-down arrow and select 'Debug' from the pull-down'.	
•	In the 'Frequently Used Options (for Compile)' group, select the 'Optimization Level' option and select '0' from the pull-down. We have now created a 'Debug' Build Mode with no code optimisation and will be using the Build Mode to create and debug the project.	



RSKRX130-512KB

- CC-RX Property All of the sample code projects Build Mode contained in this RSK are configured Build mode Rele Change property value for all build modes at once No with three Build Modes: CPU 'DefaultBuild', 'Debug' and 'Release'. PIC/PID Output File Type and Path 'Release' is created in the same way Output file type Intermediate file outp Execute Module(Load Module File) %BuildModeName% as above; by duplicating 'Default Frequently Used Options(for Compile) Additional include paths Build'. 'Release' Build Mode leaves Additional include paths[16] System include paths System include paths[0] code optimisation turned on and Macro definition Macro definition[0] Outputs debugging information Yes(-debug) removes debug information from the Optimization level 2(-optimize=2) Outputs additional information for inter-module optimization output file. Optimizes with emphasis on code size(-size) Optimization type Outputs a source list file No(-nolistfile) To remove debug information from • Frequently Used Options(for As Frequently Used Options(for Link) the 'Release' Build Mode, in the 'CC-Using libraries Using libraries[0] RX Property' window, select the Yes (Outputs to the output file)(-DEBug) Yes (Outputs to the output file)(-DEBug) Yes (Outputs to <output file name>.dbg file)(-SDeb Optimization type 'Common Options' tab at the bottom ection start addre Frequently Used Options(for Hex Output) of the window pane. For the 'Outputs debugging information' option, select 'No(-nodebug). Reset the Build Mode back to • 'Debug' using the 'Build Mode' pull-

All' to save all project settings.

From the menus, select 'File -> Save

down control.

•

5.2 **Additional Folders** Before new source files are added × to the project, we will create two 🕜 🙎 🗃 additional folders in the CS+ 📕 R5F51308 🛍 Build SC_Tutorial Project Tree. 🖫 Smart Co 👔 Rebuild SC_Tutorial CC-RX (B 👔 Clean SC_Tutorial In the Project Tree pane, right-click 🚔 RX Simula the SC_Tutorial project and select 🕀 Program IJ Open Folder with Explorer 'Add -> Add New Category'. 🗄 \iint File Windows Explorer Menu E Add Add Subproject... Add New Subproject... í. Set SC Tutorial as Active Project Ē. Add File... 1 Save Project and Development Tools as Package. Add New File. ß Paste Ctrl+V Add New Category aje Rename F2 Property Rename the newly-created 'New • SC Tutorial (Project)* Category' folder to 'C Source Files'. R5F51308AxFP (Microcontroller) Repeat these steps to create a new Smart Configurator (Design Tool) category folder for 'Dependencies'. 🔨 CC-RX (Build Tool) RX Simulator (Debug Tool) 🕀 Program Analyzer (Analyze Tool) 🖮 j File - 📶 Build tool generated files 🗄 🔄 Smart Configurator Source Files



5. Completing the Tutorial Project

5.3 Precautions on using Smart Configurator

When executing the build using Smart Configurator, the warning message shown in Figure 5-1 may be displayed.

	Output
	لر"W0561100:Cannot find "PResetPRG" specified in option "start
	لي"W0561100:Cannot find "C\$INIT" specified in option "start"
	W0561100:Cannot find "C\$YTBL" specified in option "start" لله
	W0561100:Cannot find "PIntPRG" specified in option "start", الـ "W0561100:Cannot find "PIntPRG" specified in option
l	Renesas Optimizing Linker Completed,

Figure 5-1 : CS+ output window

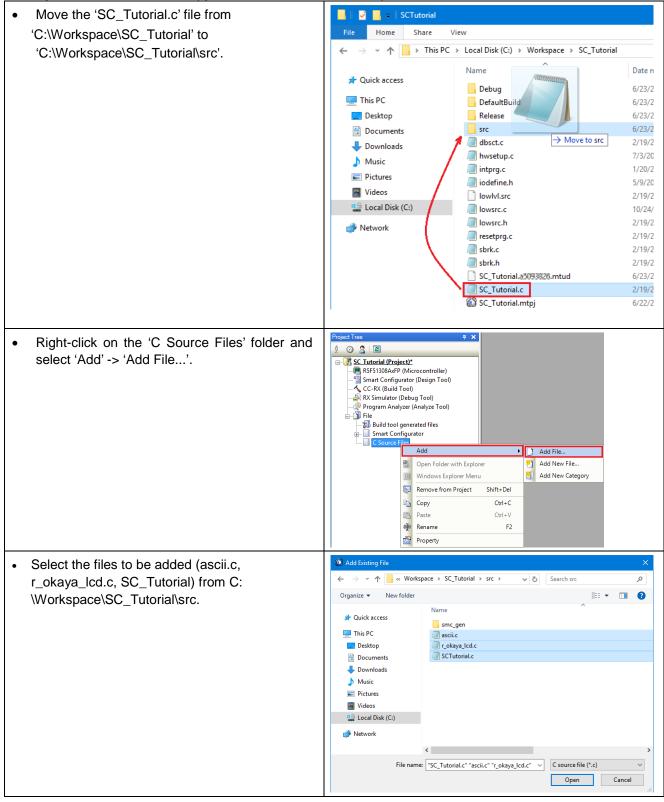
It is a warning sentence that is displayed because there are unused sections, and there is no problem in operation in SC_Tutorial created in this manual. If, do not want to display a warning message, follow the procedure below to make the setting.

•	Double-click 'CC-RX(Build Tool)' to display the build tool properties.	Project Tree 4 X 2 2 2 2
•	Select the 'Link Options' tab at the bottom of the Properties screen.	Property
•	Section is set to the start address' of the section.	Input Output Output List Optimization Section Section start address B_I.R_I.B_2.R_2.B_RSUSI/04,PResetPRG/0FFE00 The specified section that outputs externally defined symbols to the file The specified section that outputs externally defined symbols to the file Section alignment Section alignment ROM to RAM mapped section[3] Verify Others
•	Overwrite the start address of the section below.	
	B_1,R_1,B_2,R_2,B,R,SU,SI/04,C_1,C_2 ,C,C\$*,D*,W*,L,P*/0FFF80000,FIXEDVE CT/0FFFFF80	Input Output Output Utput List Optimization Section start address B_1.R_1.B_2.R_2.B.R.SU.SI./04.C_1.C_2.C The specified section that outputs externally defined symbols to the file Section alignment Section alignment Section alignment ROM to RAM mapped section[3] Verify Others



5.4 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. Locate the files ascii.h, r_okaya_lcd.h, ascii.c, and r_okaya_lcd.c in this folder. Copy these files into the C:\Workspace\SC_Tutorial\src folder.





RSKRX130-512KB

5. Completing the Tutorial Project

• Similarly, add 'ascii.h' and 'r_okaya_lcd.h' to the	Add Existing File	×
'Dependencies' folder.	$\leftarrow \rightarrow \checkmark \uparrow$ \sim \land Workspace \rightarrow SC_Tutorial \rightarrow src \rightarrow \checkmark \circlearrowright Search src	م
	Organize New folder	
Note: Choose 'Header file (*.h; *.hpp; *.inc)'	Name	
	This PC	
	Desktop	
	Documents	
	Downloads Music	
	E Pictures	
	Videos	
	Local Disk (C:)	
	💣 Network	
	<	>
	File name: "r_okaya_lcd.h" "ascii.h"	*.h; *.hpp; *.inc) 🛛 🗸
	Open	Cancel
Make ourse the project tree is the same of the	Project Tree 7 ×	
Make sure the project tree is the same as the screen shot.	9 0 3 8	
	□ <mark></mark>	
	Smart Configurator (Design Tool)	
	🔨 CC-RX (Build Tool)	
	Build tool generated files	
	C Source Files	
	… 🤤 r_okaya_lcd.c 	
	E	
	ascii.h	
		1



Code must be inserted in to the user code area in many files in this project, in the areas delimited by comments as follows:

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

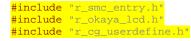
Where _xxxx_ depends on the particular area of code, i.e. 'function' for insertion of user functions and prototypes, 'global' for insertion of user global variable declarations, or 'include' for insertion of pre-processor include directives. User code inserted inside these comment delimiters is protected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

In the CS+ Project Tree, expand the 'Smart Configurator/general' folder and open the file 'r_cg_userdefine.h' by double-clicking on it. Insert the following #defines in between the user code delimiter comments as shown below.

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

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

In the CS+ Project Tree, expand the 'C Source Files' folder and open the file 'SC_Tutorial.c' by double-clicking on it. Add header files above the 'main' function as shown below.



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

```
void main(void)
{
    /* Initialize the debug LCD */
    R_LCD_Init();
    /* Displays the application name on the debug LCD */
    R_LCD_Display(0, (uint8_t *)" RSKRX130-512KB ");
    R_LCD_Display(1, (uint8_t *)" Tutorial ");
    R_LCD_Display(2, (uint8_t *)" Press Any Switch ");
    while (1U)
    {
        ;
        }
}
```



5.4.1 SPI Code

The Okaya LCD display is driven by the SPI Master that was configured using Smart Configurator in §4.4.7 In the CS+ Project Tree, expand the 'Smart Configurator/Config_SCI6' and open the file 'Config_SCI6.h' by double-clicking on it. Insert the following code in the user code area at the end of the file:

/* Start user code for function. Do not edit comment generated here */ /* Exported functions used to transmit a number of bytes and wait for completion */ MD_STATUS R_SCI6_SPIMasterTransmit(uint8_t * const tx_buf, const uint16_t tx_num);

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

Now, open the Config_SCI6_user.c file and insert the following code in the user area for global:

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

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

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

```
static void r_Config_SCI6_callback_transmitend(void)
{
    /* Start user code for r_Config_SCI6_callback_transmitend. Do not edit comment generated here */
    sci6_txdone = TRUE;
    /* End user code. Do not edit comment generated here */
}
```

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

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

```
Function Name: R_SCI6_SPIMasterTransmit
 Description : This function sends SPI6 data to slave device.
* Arguments : tx_buf -
              transfer buffer pointer
          tx_num -
              buffer size
* Return Value : status -
             MD_OK or MD_ARGERROR
MD_STATUS R_SCI6_SPIMasterTransmit (uint8_t * const tx_buf,
                         const uint16_t tx_num)
{
  MD_STATUS status = MD_OK;
  /* Clear the flag before initiating a new transmission */
  sci6_txdone = FALSE;
  /* Send the data using the API */
  status = R_Config_SCI6_SPI_Master_Send(tx_buf, tx_num);
  /* Wait for the transmit end flag */
  while (FALSE == sci6 txdone)
  {
     /* Wait */
  }
  return (status);
}
* End of function R_SCI6_SPIMasterTransmit
****************
```

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

5.4.2 TMR Code

The LCD code needs to insert delays to meet the timing requirements of the display module. This is achieved using the dedicated timer which was configured using Smart Configurator in §4.4.2. In the CS+ Project Tree, expand the 'Smart Configurator\Config_TMR0\Config_TMR0.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_TMR_MsDelay(const uint16_t millisec);
/* End user code. Do not edit comment generated here */
```

Open the file 'Config_TMR0_user.c' and insert the following code in the user area for global at the beginning of the file:

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

static volatile uint8_t one_ms_delay_complete = FALSE;

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

Scroll down to the r_Config_TMR0_cmia0_interrupt function and insert the following line in the user code area:

```
static void r_Config_TMR0_cmia0_interrupt(void)
{
    /* Start user code for r_Config_TMR0_cmia0_interrupt. Do not edit comment generated here */
    one_ms_delay_complete = TRUE;
    /* End user code. Do not edit comment generated here */
}
```

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

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

```
* Function Name: R_TMR_MsDelay
* Description : Uses TMR0 to wait for a specified number of milliseconds
* Arguments
        : uint16_t millisecs, number of milliseconds to wait
* Return Value : None
          void R_TMR_MsDelay (const uint16_t millisec)
{
  uint16_t ms_count = 0;
  do
  {
    R_Config_TMR0_Start();
    while (FALSE == one_ms_delay_complete)
     {
       /* Wait */
    R_Config_TMR0_Stop ();
    one_ms_delay_complete = FALSE;
    ms count++;
  } while (ms_count < millisec);</pre>
End of function R_TMR_MsDelay
```

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

The project may now be run using the debugger as described in §6. The program will display 'RSKRX130-512KB Tutorial Press Any Switch' on three lines in the LCD display.

5.5 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. Locate the files rskrx130_512kbdef.h, r_rsk_switch.h and r_rsk_switch.c in this folder. Copy these files into the C:\Workspace\SC_Tutorial\src folder. Add these three files into the project in the same way as the LCD files.

The switch code uses interrupt code in the files Config_ICU.c, Config_ICU_user.c and Config_ICU.h and timer code in the files Config_ICU.c, Config_ICU_user.c, Config_CMT0.h, Config_CMT0.c, Config_CMT0_user.c, Config_CMT1.h, Config_CMT1.c, and Config_CMT1_user.c, as described in §4.4.3 and §4.4.4. 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.

5.5.1 Interrupt Code

In the CS+ Project Tree, expand the 'Smart Configurator/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 IROIsFallingEdge (const uint8 t irg no)
  uint8_t falling_edge_trig = 0x0;
  if (ICU.IRQCR[irq_no].BYTE & _04_ICU_IRQ_EDGE_FALLING)
  {
     falling_edge_trig = 1;
  }
  return (falling_edge_trig);
}
End of function R_ICU_IRQIsFallingEdge
```



RSKRX130-512KB

```
* Function Name: R_ICU_IRQSetFallingEdge
* Description : This function sets/clears the falling edge trigger for the
          specified ICU_IRQ.
        : uint8_t irq_no
*
 Arguments
          uint8_t set_f_edge, 1 if setting falling edge triggered, 0 if
          clearing
* Return Value : None
     void R_ICU_IRQSetFallingEdge (const uint8_t irq_no, const uint8_t set_f_edge)
  if (1 == set_f_edge)
  {
     ICU.IRQCR[irq_no].BYTE | = _04_ICU_IRQ_EDGE_FALLING;
  }
  else
  {
     ICU.IRQCR[irq_no].BYTE &= (uint8_t) ~_04_ICU_IRQ_EDGE_FALLING;
  }
}
         * End of function R_ICU_IRQSetFallingEdge
      * Function Name: R_ICU_IRQSetRisingEdge
* Description : This function sets/clear the rising edge trigger for the
          specified ICU_IRQ.
*
        : uint8_t irq_no
Arguments
          uint8_t set_r_edge, 1 if setting rising edge triggered, 0 if
          clearing
* Return Value : None
             void R_ICU_IRQSetRisingEdge (const uint8_t irq_no, const uint8_t set_r_edge)
ł
  if (1 == set_r_edge)
  {
    ICU.IRQCR[irq_no].BYTE | = _08_ICU_IRQ_EDGE_RISING;
  }
  else
  {
     ICU.IRQCR[irq_no].BYTE &= (uint8_t) ~_08_ICU_IRQ_EDGE_RISING;
  }
}
  * End of function R_ICU_IRQSetRisingEdge
```

/* 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_irq1_interrupt:

/* Start user code for r_Config_ICU_irq1_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_irq2_interrupt:

/* Start user code for r_Config_ICU_irq2_interrupt. Do not edit comment generated here */
/* Switch 2 callback handler */
R_SWITCH_IsrCallback2();

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

5.5.2 De-bounce Timer Code

In the Project Tree, expand the 'Smart Configurator\Config_CMT0' folder and open the 'Config_CMT0_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_CMT0_user.c' file, insert the following code in the user code area inside the function r_Config_CMT0_cmi0_interrupt:

/* Start user code for r_Config_CMT0_cmi0_interrupt. Do not edit comment generated here */
/* Stop this timer - we start it again in the de-bounce routines */
R_Config_CMT0_Stop();
/* Call the de-bounce call back routine */
R_SWITCH_DebounceIsrCallback();
/* End user code. Do not edit comment generated here */

In the Project Tree, expand the 'Smart Configurator\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 */

Open the 'Config_CMT1_user.c' file and 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 */



5.5.3 Main Switch and ADC Code

In this part of the tutorial we add the code to act on the switch presses to activate A/D conversions and display the result on the LCD. In §4.4.8, 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 CS+ Project Tree, expand the 'Smart Configurator\general' folder and open the file 'r_cg_userdefine.h' by double-clicking on it. Insert the following code the user code area, resulting in the code shown below

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

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

extern volatile uint8_t g_adc_trigger;

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

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

<pre>#include "r_smc_entry.h" #include "r okaya lcd.h"</pre>
<pre>#include "r_rsk_switch.h" #include "Config S12AD0.h"</pre>
#include "r_cg_userdefine.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);
<pre>/* Prototype declaration for lcd_display_adc */ static void lcd_display_adc (const uint16_t adc_result);</pre>



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

```
void main(void)
{
    /* Initialize the switch module */
    R_SWITCH_Init();
    /* Set the call back function when SW1 or SW2 is pressed */
    R_SWITCH_SetPressCallback(cb_switch_press);
    /* Initialize the debug LCD */
    R LCD Init();
    /* Displays the application name on the debug LCD */
   R_LCD_Display(0, (uint8_t *)" RSKRX130-512KB ");
R_LCD_Display(1, (uint8_t *)" Tutorial ");
    R_LCD_Display(2, (uint8_t *)" Press Any Switch ");
      * Start the A/D converter */
    R_Config_S12AD0_Start();
    while (1U)
    {
        uint16_t adc_result;
        /* Wait for user requested A/D conversion flag to be set (SW1 or SW2) */
        if (TRUE == g_adc_trigger)
        {
             /* Call the function to perform an A/D conversion */
            adc_result = get_adc();
              * Display the result on the LCD */
            lcd_display_adc(adc_result);
            /* Reset the flag */
            g_adc_trigger = FALS<mark>E;</mark>
        /* SW3 is directly wired into the ADTRGOn pin so will
           cause the interrupt to fire */
        else if (TRUE == g_adc_complete)
        {
             /* Get the result of the A/D conversion */
            R_Config_S12AD0_Get_ValueResult(ADCHANNEL0, &adc_result);
            /* Display the result on the LCD */
            lcd_display_adc(adc_result);
             /* Reset the flag */
            g_adc_complete = FALSE;
        }
        else
        {
             /* do nothing */
        }
    }
}
```

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



```
* End of function cb_switch_press
                    ***************
* Function Name : get_adc
* Description    : Reads the ADC result, converts it to a string and displays
            it on the LCD panel.
        : none
* Argument
* Return value : uint16_t adc value
                           static uint16_t get_adc (void)
ł
   /* A variable to retrieve the adc result */
  uint16_t adc_result;
   /* Stop the A/D converter being triggered from the pin ADTRGOn */
  R_Config_S12AD0_Stop();
   /* Start a conversion */
  R_S12AD0_SWTriggerStart();
   /* Wait for the A/D conversion to complete */
  while (FALSE == g_adc_complete)
  {
     /* Wait */
   }
   /* Stop conversion */
  R_S12AD0_SWTriggerStop();
   /* Clear ADC flag */
  g_adc_complete = FALSE;
  R_Config_S12AD0_Get_ValueResult(ADCHANNEL0, &adc_result);
   /* Set AD conversion start trigger source back to ADTRGOn pin */
  R_Config_S12AD0_Start();
  return (adc_result);
}
* End of function get_adc
                    *****
* Function Name : lcd_display_adc
* Description : Converts adc result to a string and displays
* it on the LCD panel
* Argument : uint16_t adc result
            it on the LCD panel.
 Return value : none
          *****
+++++
static void lcd_display_adc (const uint16_t adc_result)
{
  /* Declare a temporary variable */
  uint8 t a;
   /* Declare temporary character string */
       lcd_buffer[11] = " ADC: XXXH";
  char
  /* Convert ADC result into a character string, and store in the local.
    Casting to ensure use of correct data type. */
  a = (uint8_t)((adc_result & 0x0F00) >> 8);
  lcd_buffer[6] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));</pre>
  a = (uint8_t)((adc_result & 0x00F0) >> 4);
  lcd_buffer[7] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
  a = (uint8_t)(adc_result & 0x000F);
  lcd_buffer[8] = (char)((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 Project Tree, expand the 'Smart Configurator\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 */

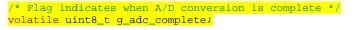
Open the file 'Config_S12AD0.c' by double-clicking on it. Insert the following code in the user code area for adding at the end of the file, resulting in the code shown below:

```
/* Start user code for adding. Do not edit comment generated here */ % f(x) = 0
* Function Name: R_S12AD0_SWTriggerStart
Description : This function starts the AD converter.
      : None
Arguments
* Return Value : None
void R_S12AD0_SWTriggerStart(void)
{
 IR(S12AD, S12ADI0) = 0U;
 IEN(S12AD, S12ADIO) = 1U;
 S12AD.ADCSR.BIT.ADST = 1U;
 Function Name: R_S12AD0_SWTriggerStop
Description : This function stops the AD converter.
      : None
Arguments
Return Value : None
****
void R_S12AD0_SWTriggerStop(void)
 S12AD, ADCSR, BIT, ADST = 0Ui
 IEN(S12AD, S12ADIO) = OU;
 IR(S12AD, S12ADIO) = 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 */



/* 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. Do not edit comment generated here */ g add complete = TRUE; /* End user code. Do not edit comment generated here */ R20UT3924EG0100 Rev. 1.00

RENESAS

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

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



5.6 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. Locate the files r_rsk_debug.h and r_rsk_debug.c in this folder. Copy these files into the C:\Workspace\SC_Tutorial\src folder. Add these two files into the project in the same way as the LCD files.

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

```
/* Macro for definition of serial debug transmit function - user edits this */
#define SERIAL_DEBUG_WRITE (R_SCI1_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.

5.7 UART Code Integration

5.7.1 SCI Code

In the CS+ Project Tree, expand the 'Smart Configurator\Config_SCI1' folder and open the file 'Config_SCI1.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_SCI1_AsyncTransmit(uint8_t * const tx_buf, const uint16_t tx_num);
/* Character is used to receive key presses from PC terminal */
extern uint8_t g_rx_char;
/* End user code. Do not edit comment generated here */
```

Open the file 'Config_SCI1_user.c'. Insert the following code in the user area for global near the beginning of the file:

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

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

/* Flag used locally to detect transmission complete */
static volatile uint8_t sci1_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_SCI1_callback_transmittend function:

```
static void r_scil_callback_transmitend(void)
{
    /* Start user code for r_Config_SCI1_callback_transmitend. Do not edit comment generated here */
    scil_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_SCI1_callback_receiveend function:

```
void r_Config_SCI1_callback_receiveend(void)
{
    /* Start user code for r_Config_SCI1_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 SCI1 receive buffer and callback function again */
    R_Config_SCI1_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_SCI1_AsyncTransmit
* Description : This function sends SCI1 data and waits for the transmit end flag.
* Arguments
        : tx_buf -
              transfer buffer pointer
           tx_num -
              buffer size
* Return Value : status -
             MD_OK or MD_ARGERROR
MD_STATUS R_SCI1_AsyncTransmit (uint8_t * const tx_buf, const uint16_t tx_num)
{
  MD_STATUS status = MD_OK;
  /* clear the flag before initiating a new transmission */
  sci1_txdone = FALSE;
  /* Send the data using the API */
  status = R_Config_SCI1_Serial_Send(tx_buf, tx_num);
  /* Wait for the transmit end flag */
  while (FALSE == sci1_txdone)
  {
     /* Wait */
  return (status);
}
* End of function R_SCI1_AsyncTransmit
 *****
```



5.7.2 Main UART code

In the Project Tree, expand the 'C Source Files' folder and open the file 'SC_Tutorial.c'. Add the following declaration to above the 'main' function:

```
#include "r_smc_entry.h"
#include "r_okaya_lcd.h"
#include "r_rsk_switch.h'
#include "r_rsk_debug.h"
#include "Config_S12AD0.h"
#include "Config_SCI1.h"
#include "r_cg_userdefine.h"
/* Variable for flagging user requested ADC conversion */
volatile uint8_t g_adc_trigger = FALSE;
/* Prototype declaration for cb_switch_press */
static void cb_switch_press (void);
/* Prototype declaration for get_adc */
static uint16_t get_adc(void);
/* Prototype declaration for lcd_display_adc */
static void lcd_display_adc (const uint16_t adc_result);
/* Prototype declaration for uart_display_adc */
static void uart_display_adc(const uint8_t adc_count, const uint16_t adc_result);
/* Variable to store the A/D conversion count for user display */
static uint8_t adc_count = 0;
```

Add the following highlighted code to the 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 *)" RSKRX130-512KB ");
    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 SCI1 receive buffer and callback function */
R_Config_SCI1_Serial_Receive((uint8_t *)&g_rx_char, 1);
    /* Enable SCI1 operations */
    R_Config_SCI1_Start();
    while (1U)
    {
        uint16_t adc_result;
        /* Wait for user requested A/D conversion flag to be set (SW1 or SW2) */
        if (TRUE == g_adc_trigger)
        {
             /* Call the function to perform an A/D conversion */
            adc_result = get_adc();
             /* Display the result on the LCD */
             lcd_display_adc(adc_result);
             /* Increment the adc_count */
             if (16 == (++adc_count))
             {
                 adc_count = 0;
             }
```



}

```
/* Send the result to the UART */
        uart_display_adc(adc_count, adc_result);
        /* 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)
    ł
        /* Get the result of the A/D conversion */
        R_Config_S12AD0_Get_ValueResult(ADCHANNEL0, &adc_result);
        /* Display the result on the LCD */
        lcd_display_adc(adc_result);
        /* Increment the adc_count */
        if (16 == (++adc_count))
            adc_count = 0;
        }
        /* Send the result to the UART */
        uart_display_adc(adc_count, adc_result);
        /* Reset the flag */
        g_adc_complete = FALSE;
    }
    else
    {
        /* do nothing */
    }
}
```

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

```
* Function Name : uart_display_adc
* Description : Converts adc result to a string and sends it to the UART1.
* Argument
           : uint8_t : adc_count
             uint16_t: adc result
* Return value : none
static void uart_display_adc (const uint8_t adc_count, const uint16_t adc_result)
{
   /* Declare a temporary variable */
  char a;
   /* Declare temporary character string */
  static char uart_buffer[] = "ADC xH Value: xxxH\r\n";
   /\ast Convert ADC result into a character string, and store in the local.
     Casting to ensure use of correct data type. */
   a = (char)(adc_count \& 0x000F);
  uart_buffer[4] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
   a = (char)((adc_result & 0x0F00) >> 8);
  uart_buffer[14] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));</pre>
  a = (char)((adc_result & 0x00F0) >> 4);
  uart_buffer[15] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));</pre>
   a = (char)(adc_result \& 0x000F);
   uart_buffer[16] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));</pre>
   /* Send the string to the UART */
  R_DEBUG_Print(uart_buffer);
}
* End of function uart_display_adc
```

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

The project may now be run using the debugger as described in §6. Connect the RSK G1CUSB0 port to a USB port on a PC. If this is the first time the RSK has been connected to the PC then a device driver will be installed automatically. Open Device Manager, the virtual COM port will now appears 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 SCI1 (see §4.4.6). When any switch is pressed, or when 'c' is sent via the COM port, the program will perform an A/D conversion of the voltage level on the ADPOT line and display the result on the LCD panel and send the result to the PC terminal program via the SCI1. Return to this point in the SC_Tutorial to add the LED user code.

5.8 LED Code Integration

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

```
#include "r_smc_entry.h"
#include "r_okaya_lcd.h"
#include "r_rsk_switch.h"
#include "r_rsk_debug.h"
#include "rskrx130_512kbdef.h"
#include "Config_S12AD0.h"
#include "Config_SCI1.h"
#include "r_cg_userdefine.h"
/* Variable for flagging user requested ADC conversion */
volatile uint8_t g_adc_trigger = FALSE;
/* Prototype declaration for cb_switch_press */
static void cb_switch_press (void);
/* Prototype declaration for get_adc */
static uint16_t get_adc(void);
/* Prototype declaration for lcd_display_adc */
static void lcd_display_adc (const uint16_t adc_result);
/* Prototype declaration for uart_display_adc */
static void uart_display_adc(const uint8_t adc_count, const uint16_t adc_result);
/* Variable to store the A/D conversion count for user display */
static uint8_t adc_count = 0;
/* Prototype declaration for led_display_count */
static void led_display_count(const uint8_t count);
```

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

```
void main(void)
ł
    /* 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 *)" RSKRX130-512KB ");
   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 SCI1 receive buffer and callback function */
   R_Config_SCI1_Serial_Receive((uint8_t *)&g_rx_char, 1);
    /* Enable SCI1 operations */
   R_Config_SCI1_Start();
   while (1U)
    {
       uint16_t adc_result;
```



RSKRX130-512KB

```
/* Wait for user requested A/D conversion flag to be set (SW1 or SW2) */
if (TRUE == g_adc_trigger)
{
    /* Call the function to perform an A/D conversion */
    adc_result = get_adc();
    /* Display the result on the LCD */
    lcd_display_adc(adc_result);
    /* Increment the adc_count and display using the LEDs */
    if (16 == (++adc_count))
    {
        adc count = 0;
    }
    led_display_count(adc_count);
    /* Send the result to the UART */
    uart_display_adc(adc_count, adc_result);
    /* Reset the flag */
    g_adc_trigger = FALSE;
}
/* SW3 is directly wired into the ADTRGOn pin so will
  cause the interrupt to fire */
else if (TRUE == g_adc_complete)
{
    /* Get the result of the A/D conversion */
   R_Config_S12AD0_Get_ValueResult(ADCHANNEL0, &adc_result);
    /* Display the result on the LCD */
    lcd_display_adc(adc_result);
    /* Increment the adc_count and display using the LEDs */
    if (16 == (++adc_count))
    {
        adc count = 0;
    led_display_count(adc_count);
    /* Send the result to the UART */
    uart_display_adc(adc_count, adc_result);
    /* Reset the flag */
    g_adc_complete = FALSE;
}
else
{
    /* do nothing */
}
```

}

}



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

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

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

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



6. Debugging the Project

 In the 'Project Tree' pane, right-click the 'RX Simulator (Debug Tool)'. Select 'Using Debug Tool -> RX E2 Lite'. 	Project Tree P × Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint of the system Image: Constraint o
 Double-click 'RX E2 Lite (Debug Tool)' to display the debugger tool properties. Under 'Clock', change the main clock frequency to 8 MHz and operating frequency to 32MHz.' Under 'Connection with Target Board', change 'Power target from the emulator. (MAX 200mA) to 'Yes'. All other settings can remain at their defaults. 	RX E2 Lite Property Internal ROM/RAM Size of internal ROM[KBytes] 512 Size of internal RAM[KBytes] 48 Size of DataFlash memory[KBytes] 8 V Clock EXTAL Main clock source EXTAL Main clock frequency[MHz] 8.0000 Operating frequency[MHz] 8.0000 Allow changing of the clock source on writing internal flash memory No V Connection with Emulator Emulator serial No. Emulator serial No. Power target from the emulator.(MAX 200mA) Yes Supply voltage 3.3V Communications method FINE FINE baud rate[bps] 1500000 1500000
 Connect the E2 Lite to the PC and the RSK E1/E2 Lite connector. Connect the Pmod LCD to the PMOD1 connector. From the 'Debug' menu select 'Download' to start the debug session and download code to the target. 	



7. Running the Smart Configurator Tutorial

7.1 Running the Tutorial

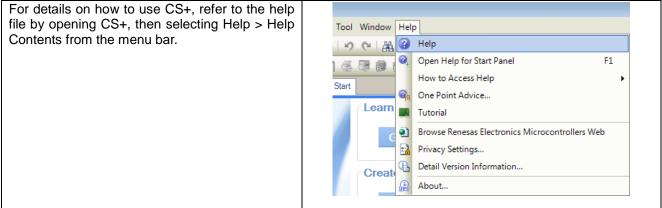
Once the program has been downloaded onto the RSK device, the program can be executed. Click the 'Go' button or press F5 to begin the program from the current program counter position. It is recommended that you run through the program once first, and then continue to the Tutorial manual to review the code.





8. Additional Information

Technical Support



For information about the RX130 group microcontroller refer to the RX130 Group Hardware Manual.

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

Technical Contact Details

Please refer to the contact details listed in section 9 of the "Quick Start Guide".

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

Trademarks

All brand or product names used in this manual are trademarks or registered trademarks of their respective companies or organisations.

Copyright

This document may be, wholly or partially, subject to change without notice. All rights reserved. Duplication of this document, either in whole or part is prohibited without the written permission of Renesas Electronics Europe Limited.

© 2017 Renesas Electronics Europe Limited. All rights reserved.

© 2017 Renesas Electronics Corporation. All rights reserved.

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



REVISION HISTORY

RSKRX130-512KB Smart Configurator Tutorial Manual

Rev.	Date	Description	
		Page	Summary
1.00	Jun 30, 2017	—	First Edition issued

Renesas Starter Kit Manual: Smart Configurator Tutorial Manual

Publication Date: Rev. 1.00 Jun 30, 2017

Published by: Renesas Electronics Corporation



SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information. Renesas Electronics America Inc. 2001 Scott Boulevard Santa Clara, CA 90505-2549, U.S.A. Tel: +1408-588-6100, Fax: +14-08-588-6130 Renesas Electronics Canada Limited 2051 Yong Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3 Tel: +1905-237-2004 Renesas Electronics Canado Limited Dukes Meadow, Milboad Road, Bourne End, Buckinghamshire, SL8 5FH, U.K Tel: +44-1628-585-100, Fax: +44-1628-585-900 Renesas Electronics Canado Limited Dukes Meadow, Milboad Road, Bourne End, Buckinghamshire, SL8 5FH, U.K Tel: +49-211-8503-0, Fax: +44-1628-585-900 Renesas Electronics Change End, Buckinghamshire, SL8 5FH, U.K Tel: +49-211-8503-0, Fax: +49-211-5503-1327 Renesas Electronics (Shanghai) Co., Ltd. Room 1709, Quantum Plaza, No.27 ZhiChunLu Haldian District, Beijing 100191, P.R.China Tel: +89-10-825-1155, Fax: +86-10-825-7679 Renesas Electronics (Shanghai) Co., Ltd. Niti 301, Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333 Tel: +86-21-2226-0888, Fax: +86-21-2226-0989 Renesas Electronics Taiwan Co., Ltd. Diff. No. 303, Tower 2, Garand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong Tel: +852-1852-6688, Fax: +862-218-2020 Renesas Electronics Taiwan Co., Ltd. Diff. No. 303, Tower 2, Garand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong Tel: +852-266-6688, Fax: +862-218-0020 Renesas Electronics Malwing Kong Limited Diff. No. 303, Towir +65-6213-0300 Renesas Electronics Malwing Kong, Imited Diff. No. 303, Towir +65-213-0300 Renesas Electronics India Pvt. Ltd. No.777C, 100 Ke B, Meara Anncorp, Anncorp Trade Centre, No. 18, Jin Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia Tel: +603-735-9390, Fax: +03-735-7670 Renesas Electronics India Pvt. Ltd. No.777C, 100 Ke B, Meara Anncorp, Jincorp Trade Centre, No. 18, Jin Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia Tel: +603-735-9390, Fax: +043-73570, Fax: Habe-

RX130 Group



R20UT3924EG0100