



RX23W Group

Renesas Solution Starter Kit for RX23W Smart Configurator Tutorial Manual For e² studio

RENESAS 32-Bit MCU RX Family / RX200 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

Rev. 1.00 Aug 2019

Notice

- Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
- Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
- 3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
- 4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
- Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
 - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

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

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

- 6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
- 7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
- 8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
- 9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
- 10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
- This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
 Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas
- Electronics products.
- (Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
- (Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

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

www.renesas.com

Trademarks

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

Contact information

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

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

2. Processing at power-on

The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power is supplied until the power reaches the level at which reseting is specified.

3. Input of signal during power-off state

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

4. Handling of unused pins

Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin

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

7. Prohibition of access to reserved addresses

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

8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Disclaimer

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

The RSSK is not guaranteed to be error free, and the entire risk as to the results and performance of the RSSK is assumed by the User. The RSSK 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 RSSK. 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 RSSK, 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 RSSK product:

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

How to Use This Manual

1. Purpose and Target Readers

This manual is designed to provide the user with an understanding of how to use Smart Configurator for RX together with the e² studio IDE to create a working project for the RSSK platform. It is intended for users designing sample code on the RSSK platform, using the many different incorporated peripheral devices.

The manual comprises of step-by-step instructions to generate code and import it into e² studio, but does not intend to be a complete guide to software development on the RSSK platform. Further details regarding operating the RX23W microcontroller may be found in the RX23W Group Hardware Manual and within the provided sample code. The setup procedure for the RSSK Web installer is described in the Quick Start Guide.

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

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

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

The following documents apply to the RX23W 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 RSSK hardware.	Renesas Solution Starter Kit for RX23W User's Manual	R20UT4446EG
Tutorial Manual	Provides a guide to setting up RSSK environment, running sample code and debugging programs.	Renesas Solution Starter Kit for RX23W Tutorial Manual	R20UT4447EG
Quick Start Guide	Provides simple instructions to setup the RSSK and run the first sample.	Renesas Solution Starter Kit for RX23W Quick Start Guide	R20UT4448EG
Smart Configurator Tutorial	Provides a guide to code generation and importing into the e ² studio IDE.	Renesas Solution Starter Kit for RX23W Smart Configurator Tutorial Manual	R20UT4449EG
Schematics	Full detail circuit schematics of the RSSK.	Renesas Solution Starter Kit for RX23W Schematics	R20UT4445EG
Hardware Manual	Provides technical details of the RX23W microcontroller.	RX23W Group User's Manual: Hardware	R01UH0823EJ

2. List of Abbreviations and Acronyms

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

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

Table of Contents

1. Overview	8
1.1 Purpose	
1.2 Features	
2. Introduction	9
 3. Project Creation with e² studio 3.1 Introduction 	
3.2 Creating the Project	
4. Smart Configurator Using the e ² studio	13
4.1 Introduction	
4.2 Project Configuration using Smart Configurator	
4.3 The 'Board' tabbed page	
4.3.1 Board configuration page	
4.4 The 'Clocks' tabbed page 4.4.1 Clocks configuration	
4.4.1 Clocks configuration	
4.5.1 Add a software component into the project	
4.5.1 Add a software component into the project	
4.5.3 Interrupt Controller	
4.5.4 Ports	
4.5.5 SCI/SCIF Asynchronous Mode	
4.5.6 SPI Clock Synchronous Mode	
4.5.7 Single Scan Mode S12AD	
4.5.7 Single Scan Mode STZAD	
4.6.1 Change pin assignment of a software component	
4.0.1 Change pin assignment of a software component	
5. User Code Integration	40
5.1 Project Settings	40
5.2 LCD Code Integration	
5.2.1 SPI Code	
5.2.2 CMT Code	
5.3 Additional include paths	
5.4 Switch Code Integration	
5.4.1 Interrupt Code	
5.4.2 De-bounce Timer Code	
5.4.3 Main Switch and ADC Code	
5.5 Debug Code Integration	
5.6 UART Code Integration	
5.6.1 SCI Code	
5.6.2 Main UART code	
5.7 LED Code Integration	57
6. Debugging the Project	59
7. Additional Information	
	•••••••••••••••••••••••••••••••••••••••

RENESAS

1. Overview

1.1 Purpose

This RSSK is an evaluation tool for Renesas microcontrollers. This manual describes how to use the e² studio IDE Smart Configurator plug-in to create a working project for the RSSK platform.

1.2 Features

This RSSK provides an evaluation of the following features:

- Project Creation with e² studio.
- Code generation using the Smart Configurator plug-in.
- User circuitry such as switches, LEDs and a potentiometer.

The RSSK board contains all the circuitry required for microcontroller operation.



2. Introduction

This manual is designed to answer, in tutorial form, how to use the Smart Configurator plug-in for the RX family together with the e² studio IDE to create a working project for the RSSK platform. The tutorials help explain the following:

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

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

- 'HardwareDebug' is a project built with the debugger support included. Optimisation is set to zero.
- 'Release' is a project with optimised compile options (level two) and 'Outputs debugging information' option not selected, producing code suitable for release in a product.

These tutorials are designed to show you how to use the RSSK and are not intended as a comprehensive introduction to the e² studio debugger, compiler toolchains or the E2 emulator Lite. Please refer to the relevant user manuals for more in-depth information.



3. Project Creation with e² studio

3.1 Introduction

In this section, the user will be guided through the steps required to create a new C project for the RX23W MCU, ready to generate peripheral driver code using Smart Configurator. This project generation step is necessary to create the MCU-specific source, project and debug files.

3.2 Creating the Project

•	Start e ² studio and select a suitable location for the project workspace.	e ² Eclipse Launcher × Select a directory as workspace e ³ studio uses the workspace directory to store its preferences and development artifacts. Workspace: C:\Workspace > Use this as the default and do not ask again Launch Cancel
•	In the Welcome page, click 'Create a new C/C++ project'.	Image: state stat
•	In the 'Templates for New C/C++ Project' dialog, selecting 'Renesas RX' -> 'Renesas CC-RX C/C++ Executable Project'. Click 'Next'.	



3. Project Creation with e² studio

• Enter the project name 'SC_Tutorial'. (e² - X New Renesas CC-RX Executable Project Image: SC_Tutorial Project name: SC_Tutorial Image: SC_Tutorial Image: SC_Tutorial Image: SC_Teate Directory for Project Choose file system: Image: SC_Tutorial Image: SC_Tutorial Image: SC_Teate Directory for Project Image: SC_Tutorial Image: SC_Tutorial Image: SC_Tutorial Image: SC_Teate Directory for Project Image: SC_Tutorial Image:
		Working sets New Working sets: Sglect Sglect Sglect
•	In the 'Select toolchain, device & debug settings' dialog, select the options as shown in the screenshot opposite.	Image: Color of the second sector of the second
•	In 'Toolchains' choose 'Renesas CCRX'.	Toolchain Settings Language:
•	The R5F523W8AxBL MCU is found under RX200 -> RX23W ->	Language: C C C++ Toolchain: Renesas CCRX
	RX23W - 85 pin.	Toolchain Version: v3.01.00 v RTOS: None v
•	Select 'E2 Lite (RX)' from the pulldown and	Manage Toolchains
	check 'Create Release Configuration' check	Device Settings Configurations Target Device R5F523W8AxBL
•	box. Click 'Next'.	Unlock Devices E2 Lite (RX)
•		Project Type: Default Create Debug Configuration
		Create Release Configuration
		< Back
•	In the 'Select Coding Assistant Tool' dialog, select 'Smart Configurator'.	el - C X New Renesas CC-RX Executable Project
•	Click 'Next'.	Select Coding Assistant settings
		User Application Driver and Middleware Driver Code Configured in GUI Selected in GUI and imported MCU Hardware MCU Hardware Configured in GUI Back Linith Cancel



•	Click 'Next'.	New Renesas CC-RX Executable Project Settings The Contents of Files to be Generated What kind of initialization routine would you like to create? Use Renesas Debug Virtual Console Number of I/O Streams: 20
		< Back
•	A summary dialog will appear, click 'Finish' to complete the project generation.	Image: Second
•	Wait for file generation to start.	Progress Information Image: Smart Configurator operation in progress Preparing startup code Cancel
•	In future, to skip the pop-up message on the right, check the 'Always use this setting' check box and click on 'Open Perspective'.	e? Open Associated Perspective? × Image: Comparison of the project is associated with the Smart Configurator perspective. Do you want to open this perspective now?
•	The perspective changes automatically when the Smart Configurator starts up.	Remember my decision <u>Open Perspective</u> <u>No</u>



4. Smart Configurator Using the e² studio

4.1 Introduction

The Smart Configurator plug-in for the RX23W has been used to generate the sample code discussed in this document. Smart Configurator for e² studio is a plug-in tool for generating template 'C' source code and project settings for the RX23W. When using Smart Configurator, it provides the user with a visual way of configuring the target device, clocks, software components, hardware resources and interrupts for the project; thereby bypassing the need, in most cases, to refer to sections of the Hardware Manual.

Once the user has configured the project, the 'Generate Code' function is used to generate three code modules for each specific MCU feature selected. These code modules are named 'Config_xxx.h', 'Config_xxx.c', and 'Config_xxx_user.c', where 'xxx' is an acronym for the relevant MCU feature, for example 'S12AD'. Within these code modules, the user is then free to add custom code to meet their specific requirement. However, these files require custom code to be added between the following comment delimiters:

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

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

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

By following the steps detailed in this Tutorial, the user will generate an e² studio project called SC_Tutorial. The fully completed Tutorial project is contained in the RSSK Web Installer

(<u>https://www.renesas.com/rsskrx23w/install/e2</u>) and may be imported into e² studio by following the steps in the Quick Start Guide. This Tutorial is intended as a learning exercise for users who wish to use the Smart Configurator to generate their own custom projects for e² studio.

The SC_Tutorial project uses interrupts for switch inputs, the ADC module, the Compare Match Timer (CMT), the Serial Communications Interface (SCI) and uses these modules to perform A/D conversion and display the results via the virtual COM port in a terminal program and also on the PMOD display connected to the RSSK.

* USB serial communication requires a USB serial driver manufactured by FTDI. Please download from the following URL. Please contact FTDI for driver installation and questions. <u>https://www.ftdichip.com/Drivers/D2XX.htm</u>

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



4.2 **Project Configuration using Smart Configurator**

In this section, a brief tour of Smart Configurator is presented. For further details of the Smart Configurator paradigm and reference, refer to the RX Smart Configurator User's Guide: e² studio. You can download the latest document from: <u>https://www.renesas.com/smart-configurator</u>.

<u>File Edit Navigate Search Project Ren</u>	esas <u>V</u> iews <u>R</u> un <u>W</u> indow <u>H</u> elp		
🐔 🐐 🔳 🔅 Debug	V ESC_Tutorial HardwareDebug V 🏟 🗄 🐨 📰 🌚 V	% • 🗟 फि • ७ ाम 💷 😭 🖏 ८। 🖼 🖉 स्ट । 🖗 • 🢁 •	
		Quick Access	📅 C/C++ 📓 Smart Configurator 💠 Debug
Project Explorer 🛛 🗖 🗖	∰ SC_Tutorial.scfg ⊠		🔂 MCU Package 🙁 👘 🗖
E SC_Tutorial	Overview information	1	📓 🔺 🔎 🖻 🖻 🔺 »
> 🔊 Includes > 🔑 src	← General Information	0	
SC_Tutorial HardwareDebug.launch SC_Tutorial.scfg	This editor allows you to modify the settings stored in configuration file (.scf	g)	
	Board Allow board and device selection Clocks Allow clock configuration Components Allow software component selection and configuration Pins Allow general pin configuration and pin configuration for selected software of the sel	Application under development Device driver RTOS + Components + Pins	
< >	Overview Board Clocks Components Pins Interrupts		► Legend
E Console 🛛		🔝 Configuration Problems 🛛 🖳 Debugger Console 🔍 Smart Browser 🖇	3
Smart Configurator Output M05000012: File generated: <u>src\smc ge</u> M05000012: File generated: <u>src\smc ge</u>	en\r pincfg\Pin.c	Device: -	◇ ◆ ※ 激 囧 泰 非 参 ◆
<pre>M06000002: File generated:<u>src\smc_gu</u> M06000002: File generated:<u>src\smc_gu</u> M00000002: Code generation is succe: M03000004: File modified:<u>src\smc_gen</u></pre>	en\general\r smc interrupt.h ssful	Context Help User's Manual Technical Update Application Notes T Total: 1	ool News Notifications
	■ [1] = ¥ (0]	0 items selected	

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

Figure 4-1 Overview page

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



4.3 The 'Board' tabbed page

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

🗱 SC_Tutorial.scfg 🙁	
Device selection	🔁 🖆
Device selection	24
Board: Custom User Board Device: R5F523W8AxBL Download more boards	
Overview Board Clocks Components Pins Interrupts	

Figure 4-2 Board configuration page

4.3.1 Board configuration page

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

∰ SC_Tutorial.scfg ⊠				
Device s	Device selection			
Device selection				
Board:	Custom User Board 🛛 🗸 🗸			
Device:	R5F523W8AxBL			
	Download more boards			

Figure 4-3 Select board



4.4 The 'Clocks' tabbed page

The 'Clocks' tabbed page configures clocks of the device selected. Clock source, frequency, PLL settings and clock divider settings can be configured for the output clocks. Clock configurations will be reflected in the r_bsp_config.h file in \src\smc_gen\r_config.

4.4.1 Clocks configuration

Figure 4-4 shows a screenshot of Smart Configurator with the Clocks configurations. Click on the 'Clocks' tab. Configure the system clocks as shown in the figure. In the tutorial, the HOCO clock is used as the clock source. Select the route using the HOCO clock by the selector.

* An exclamation mark ¹ appears in the USB clock position, but ignore it because this project does not use the USB clock.

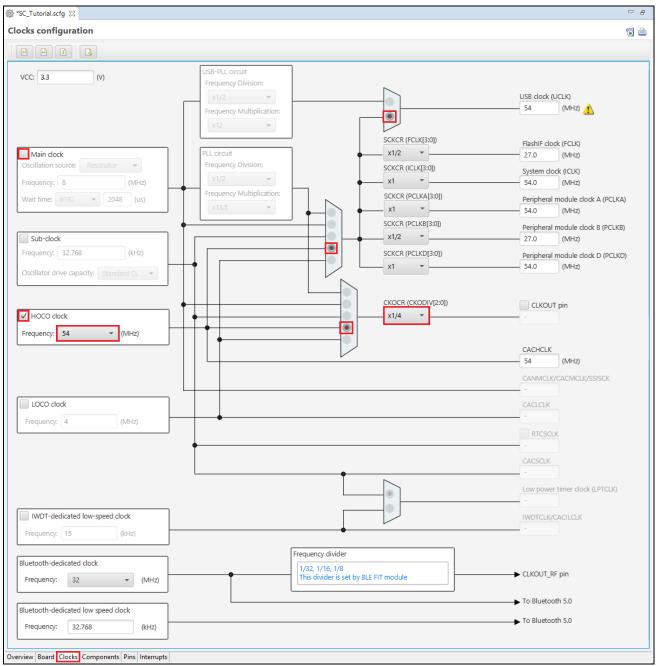


Figure 4-4 The 'Clocks' Tabbed page



4.5 The 'Components' tabbed page

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

Software compor	nent configuration
Components	
	ت 😺
type filter text	
✓ ➢ Startup ✓ ➢ Generic ✓ ➢ r_bsp	
- 1_03p	

Figure 4-5 The 'Components' tabbed page

4.5.1 Add a software component into the project

Smart Configurator supports five types of software components: Startup, Drivers, Middleware, Application and RTOS. In the following sub-sections, the reader is guided through the steps to configure the MCU for a simple project containing interrupts for switch inputs, timers, ADC and a SCI by component of Drivers.

lick the 'Add component' 笔 icon.	👼 SC_Tutorial.scfg ⊠
	Software component configuration
	Components $\downarrow_{Z}^{a} \boxdot \textcircled{H} \xrightarrow{\bullet}$ Configure
	1
	type filter text V >> Startup
	✓ (Generic
Fiç	gure 4-6 Add a Software component (

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

e ² New	e² New Component — 🗆 🗙				
Software Component Selection Select component from those available in list					
Type All				\sim	
Function All Startup Filter Drivers					
Comp	Middleware Application or RTOS				
🖶 8-E	lit Timer	Code Generator	1.6.0		
🖶 Bu	ses	Code Generator	1.6.0		
🖶 Clo	ck Frequency Accuracy Me	Code Generator	1.6.0		
C ;	Figure 4.7 Add a Software component (2)				

Figure 4-7 Add a Software component (2)



4.5.2 **Compare Match Timer**

CMT0 will be used as an interval timer for generation of accurate delays. CMT1 and CMT2 will be used as timers in de-bouncing of switch interrupts. Select 'Compare Match Timer' as shown in **Figure 4-8** below then click 'Next'.

Software	Component Selection			
Select co	mponent from those available	e in list		
Туре	Drivers			、 、
Function	All			,
Filter				
Filter				
Compo	nents	Туре	Version	^
- Com		Code Generator	1.6.0	- 1
	pare Match Timer	Code Generator	1.8.0	
- Com	plementary PWM Mode Ti	Code Generator	1.6.0	
🖶 Cont	inuous Scan Mode S12AD	Code Generator	1.6.0	
CRC	Calculator	Code Generator	1.6.0	
🖶 D/A	Converter	Code Generator	1.6.0	
🖶 Data	Operation Circuit	Code Generator	1.6.0	
🖶 Data	Transfer Controller	Code Generator	1.6.0	
	Controller	Code Generator	1.5.0	
<		C 1 C 1	150	>
Show	only latest version			
Descriptio				
· · · ·	tware component provides co	onfigurations for 16-hit	/32-bit timer with	
	CMT/CMTW and can genera			
				~
	d more software components			
C C	e general settings			
Configur				
Configur				
Contigur				
Contigur				
Contigur				

Figure 4-8 Select Compare Match Timer

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

e ² New Component			
Add new configuratio	n for selected component		
Compare Match Timer			
Configuration name:	Config_CMT0		
Resource:	CMT0		\sim
?	< <u>B</u> ack <u>N</u> ext >	<u>F</u> inish	Cancel

Figure 4-9 Select Resource - CMT0

In 'Config_CMT0' configure CMT0 as shown in **Figure 4-10**. 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.

Compare match interrupt (CMI0) Compare match interrupt (CMI0)	Components $\downarrow^{a}_{Z} \boxdot \Rightarrow \checkmark$	Configure	
∨ ▷ Startup Interval value 1 ms ∨ (Actual value: 1) ∨ ▷ Generic Interval value 1 3374 ∨ ▷ Drivers ▷ Compare match interrupt (CMI0) 3374			○ PCLK/128 ○ PCLK/512
Priority Level 10	 ✓ ⇐ Generic ir_bsp ✓ ⇐ Drivers 	Interval value Register value (CMCOR)	

Figure 4-10 Config_CMT0 setting

Click the 'Add component' ^{to} icon. In 'Software Component Selection' dialog -> Type, select 'Drivers'. Select 'Compare Match Timer' then click 'Next'. In 'Add new configuration for selected component' dialog -> Resource, select 'CMT1' as shown in **Figure 4-11** below then click 'Finish'.

e ² New Component			
Add new configuratio	n for selected component		
Compare Match Timer			
Configuration name:	Config_CMT01		
Resource:	CMT0		~
	CMT0		
	CMT1 CMT2		
	CMT3		
?	< Back Next >	Finish	Cancel
		-	

Figure 4-11 Select Resource – CMT1

Navigate to the 'Config_CMT1' and configure CMT1 as shown in **Figure 4-12**. This timer is configured to generate a high priority interrupt after 20ms. This timer is used as our short switch de-bounce timer later in this tutorial.

Components $ a_{Z} \models f \Rightarrow \bullet$	Configure	
type filter text	Count clock setting O PCLK/8 PCLK/32	○ PCLK/128 ○ PCLK/512
 ✓ ➢ Startup ✓ ➢ Generic ✓ ♂ r_bsp ✓ ➢ Drivers ✓ ➢ Timers ✓ Config_CMT0 ✓ Config_CMT1 	Compare match setting Interval value Register value (CMCOR) I Compare match interrupt (CMI1) Priority	20 (Actual value: 20) 16874

Figure 4-12 Config_CMT1 setting

Click the 'Add component' icon. In 'Software Component Selection' dialog -> Type, select 'Drivers'. Select 'Compare Match Timer' then click 'Next'. In 'Add new configuration for selected component' dialog -> Resource, select 'CMT2' as shown in **Figure 4-13** below then click 'Finish'.

e ² New Component			—		×
Add new configuratio	n for selecte	ed component	t		
Compare Match Timer					
Configuration name:	Config_CM	T01			
Resource:	CMT0				\sim
	CMT0 CMT1				
	CMT2				
	CMT3				-
?	< Back	Next >	Finish	Can	cel
F '	40.0				

Figure 4-13 Select Resource – CMT2

Navigate to the 'Config_CMT2' and configure CMT2 as shown in **Figure 4-14**. This timer is configured to generate a high priority interrupt after 200ms. This timer is used as our long switch de-bounce timer later in this tutorial.

Components $\downarrow^{a}_{Z} \boxdot \boxdot \Rightarrow \checkmark$	Configure	
type filter text	Count clock setting PCLK/8 PCLK/32 Compare match setting Interval value Register value (CMCOR) Compare match interrupt (CMI2) Priority	○ PCLK/128 ● PCLK/512 200 ms 〈 Actual value: 200.00237) 10546 ✓
Config_CMT2		

Figure 4-14 Config_CMT2 setting



4.5.3 Interrupt Controller

Referring to the RSSK schematic, SW1 is connected to IRQ1(P31) and SW2 is connected to IRQ0 (P30).

Click 'Add component' ^t icon. In 'Software Component Selection' dialog -> Type, select 'Drivers'. Select 'Interrupt Controller' as shown in **Figure 4-15** then click 'Next'.

e ² New C	omponent		— C	
Software	Component Selection			
Select cor	nponent from those availal	ole in list		
Туре	Drivers			\sim
Function	All			\sim
Filter				
Compo	nents	Туре	Version	^
H Event	t Link Controller	Code Generator	1.5.0	
🖶 Grou	p Scan Mode S12AD	Code Generator	1.6.0	
🖶 12C N	Aaster Mode	Code Generator	1.6.0	
🖶 I2C S	lave Mode	Code Generator	1.6.0	
🖶 Inter	rupt Controller	Code Generator	1.8.0	
H Low	Power Consumption	Code Generator	1.8.0	
H Low	Power Timer	Code Generator	1.3.0	
🖶 Norn	nal Mode Timer	Code Generator	1.6.0	
	e Counting Mode Timer	Code Generator	1.8.0	
	0 1 I F II	- 1	100	> [*]
Descriptio	ware component generate: MR) module that comprise			^ ~
	d more software componer e general settings	<u>ts</u>		
?	< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish C	ancel

Figure 4-15 Select Interrupt Controller

In 'Add new configuration for selected component' dialog -> Resource, select 'ICU' as shown in **Figure 4-16** below then click 'Finish'.

e ² New Component			_	
Add new configuratio	n for select	ed component		
Interrupt Controller				
Configuration name:	Config_ICU			
Resource:	ICU			~
?	< Back	Next >	Finish	Cancel
Figure 4-16 Select Resource – ICU				



Navigate to the 'Config_ICU', configure these two interrupts as falling edge triggered as shown in **Figure 4-17** below.

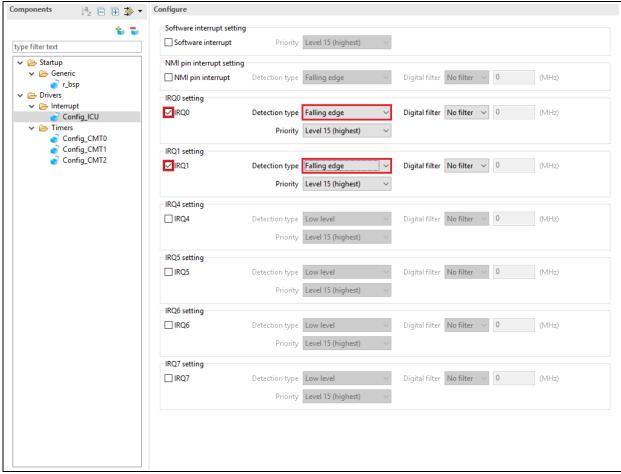


Figure 4-17 Config_ICU setting



4.5.4 Ports

Referring to the RSSK schematic, LED0 is connected to P41, LED1 is connected to P42, LED2 is connected to P43 and LED3 is connected to P44. PE3 is used as one of the LCD control lines, together with PB3, P03 and PJ3.

Click 'Add component' icon. In 'Software Component Selection' dialog -> Type, select 'Drivers'. Select 'Ports' as shown in **Figure 4-18** then click 'Next'.

e ² New Component						
Software Component Selection						
Select component from those availabl	e in list					
				_		
Type Drivers				\sim		
Function All				\sim		
Filter						
Components	Туре	Version		^		
H Normal Mode Timer	Code Generator	1.6.0				
🖶 Phase Counting Mode Timer	Code Generator	1.8.0				
🖶 Port Output Enable	Code Generator	1.6.0				
# Ports	Code Generator	1.8.0				
H PWM Mode Timer	Code Generator	1.6.0				
# Real Time Clock	Code Generator	1.4.0				
SCI/SCIF Asynchronous Mode	Code Generator	1.6.0				
SCI/SCIF Clock Synchronous M	Code Generator	1.6.0				
H Single Scan Mode S12AD	Code Generator	1.8.0		J		
C C C C C C C C C C C C C C C C C C C	C 1 C 1	100	>			
Show only latest version						
Description						
This software component provides configurations for General Purpose Input/Output. Common features such as reading, writing, and setting the direction of ports and pins can be configured. Inabling features such as open-drain outputs and internal pull-ups are also supported.						
Download more software components						
Configure general settings						
? < <u>B</u> ack	<u>N</u> ext > <u>F</u> inisl	h	Cancel			

Figure 4-18 Select Ports

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

e ² New Component				
Add new configuratio	n for selected component			
Ports				
Configuration name:	Config_PORT			
Resource:	PORT			\sim
?	< Back Next > Fin	nish	Can	cel

Figure 4-19 Select Resource – PORT



Tick the tickboxes for 'PORT0', 'PORT4', 'PORTB', PORTE' and 'PORTJ' as shown in Figure 4-20 below.

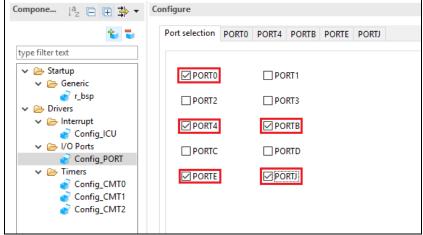


Figure 4-20 Select Port selection

Navigate through each of the 'PORTx' tabs, configuring these four I/O lines and LCD control lines as shown in **Figure 4-21**, **Figure 4-22**, **Figure 4-23**, **Figure 4-24** and **Figure 4-25** below. Ensure that the 'Output 1' tick box is checked, except for PJ3 under the 'PORTJ' tab. Start with the 'PORT0' tab.

Components $\downarrow_{\mathbb{Z}}^{a} \models \ddagger \stackrel{*}{\Rightarrow} \bullet$	Configure	
te te	Port selection PORT0 PORT4 PORTB PORTE PORTJ	
type filter text		
✓ ➢ Startup ✓ ➢ Generic ✓ ➢ r_bsp	Apply to all Unused GPIO In Out Pull-up	Output 1
✓	P03 O Unused GPIO O In Out Pull-up	Output 1
Config_PORT Config_PORT Config_CMT0	P05 ● Unused GPIO ○ In ○ Out □ Pull-up	Output 1
Config_CMT1	P07 ● Unused GPIO ○ In ○ Out □ Pull-up	Output 1

Figure 4-21 Select 'PORT0' tab



Select 'PORT4' tab.

Components $\downarrow^a_{\mathbb{Z}} \models \doteqdot \clubsuit \checkmark$	Configure	
10 T	Port selection PORT0 PORT4 PORTB PORTE PORTJ	
type filter text		
 ✓ ➢ Startup ✓ ➢ Generic 	Apply to all Unused GPIO In Out Pull-up	Output 1
 ✓ ➢ Drivers ✓ ➢ Interrupt ✓ Config_ICU 	P40	Output 1
 I/O Ports Config_PORT Timers 	P41 O Unused GPIO O In Out Pull-up	Output 1
Config_CMT0 Config_CMT1 Config_CMT2	P42	
	O Unused GPIO In Out Pull-up	☑ Output 1
	O Unused GPIO O In Out Pull-up	🗹 Output 1
	O Unused GPIO O In Out Pull-up	Output 1
	P45	Output 1
	P46 ● Unused GPIO	Output 1
	P47	Output 1

Figure 4-22 Select 'PORT4' tab

Salact	'PORTE	' tob
Select	PURIE	lap.

Components $\downarrow^{a}_{Z} \boxdot \Rightarrow \checkmark$	Configure
te te	Port selection PORT0 PORT4 PORTB PORTE PORTJ
type filter text	
 ✓ ➢ Startup ✓ ➢ Generic 	Apply to all Unused GPIO In Out Pull-up CMOS output Output High-drive output
✓ r_bsp ✓ ▷ Drivers	PBO
✓	Unused GPIO O In O Out Pull-up CMOS output Output 1 High-drive output
Config_PORT	PB1
✓ ⇐ Timers Config_CMT0	Output □ Pull-up CMOS output ∨ □ Output 1 □ High-drive output Output Output 1 □ High-drive output Ou
Config_CMT1	PB3
Config_CMT2	◯ Unused GPIO ◯ In Out □ Pull-up CMOS output ∨ ☑ Output 1 □ High-drive output
	PB5
	● Unused GPIO ○ In ○ Out □ Pull-up CMOS output ○ Output 1 □ High-drive output
	PB7 • Unused GPIO O In O Out Pull-up CMOS output v Output 1 High-drive output

Figure 4-23 Select 'PORTB' tab

Select 'PORTE' tab.

Components $\downarrow^a_{\mathbb{Z}} \boxdot \textcircled{H} \xrightarrow{\bullet} \checkmark$	Configure
10 N	Port selection PORT0 PORT4 PORTB PORTE PORTJ
type filter text	
🗸 🔁 Startup	Apply to all
✓ ⇐ Generic T_bsp	Output 1 □ High-drive output Output Output 1 □ High-drive output Ou
✓ ➢ Drivers	PEO
 ✓ (Interrupt ✓ Config_ICU ✓ (> I/O Ports 	● Unused GPIO O In O Out Pull-up CMOS output Output 1 High-drive output
Config_PORT	PE1
 Timers Config_CMT0 	● Unused GPIO ○ In ○ Out □ Pull-up CMOS output ○ Output 1 □ High-drive output
Config_CMT1	PE2
Config_CMT2	● Unused GPIO ○ In ○ Out □ Pull-up CMOS output ∨ □ Output 1 □ High-drive output
	PE3
	○ Unused GPIO ○ In O Out □ Pull-up CMOS output ✓ O Output 1 □ High-drive output
	PF4
	Unused GPIO O In O Out Pull-up CMOS output Output 1 High-drive output

Figure 4-24 Select 'PORTE' tab

Select 'PORTJ' tab.

Components $\downarrow^{a}_{\mathbb{Z}} \boxdot \textcircled{+} \xrightarrow{+} \checkmark$	Configure
1	Port selection PORT0 PORT4 PORTB PORTE PORTJ
type filter text	
V 🔁 Startup	Apply to all
✓ ⇐ Generic i _ bsp	Output 1 □ High-drive output Output 1 □ High-drive output
V 🗁 Drivers	PJ3
 Interrupt Config_ICU 	O Unused GPIO O In O Out □ Pull-up CMOS output ✓ □ Output 1 □ High-drive output
✓ ▷ I/O Ports Config_PORT	
 ✓ (⇒ Timers 	
Config_CMT0	
Config_CMT2	

Figure 4-25 Select 'PORTJ' tab



4.5.5 SCI/SCIF Asynchronous Mode

In the RSSKRX23W, SCI8 is connected via the FT234XD USB-UART converter to provide a USB virtual COM port as shown in the schematic.

Click 'Add component' ¹ icon. In 'Software Component Selection' dialog -> Type, select 'Drivers'. Select 'SCI/SCIF Asynchronous Mode' as shown in **Figure 4-26** then click 'Next'.

2 New Component		— C	
oftware Component Selection Select component from those availabl	e in list		
Type Drivers			~
Function All			
			*
Filter			
Components	Туре	Version	^
Real Time Clock	Code Generator	1.4.0	
SCI/SCIF Asynchronous Mode	Code Generator	1.6.0	
BCI/SCIF Clock Synchronous M	Code Generator	1.6.0	
H Single Scan Mode S12AD	Code Generator	1.8.0	
H Smart Card Interface Mode	Code Generator	1.6.0	
H SPI Clock Synchronous Mode	Code Generator	1.6.0	
🖶 SPI Operation Mode	Code Generator	1.6.0	
H Voltage Detection Circuit	Code Generator	1.6.0	
H Watchdog Timer	Code Generator	1.6.0	
<			>
Show only latest version			
Description			
		D aire al a (acculti	
This software component provides c processor) asynchronous mode.	onfigurations for SCI(SCI	r) single(multi-	<u>^</u>
			\sim
Download more software component	<u>s</u>		
Configure general settings			
? < <u>B</u> ack	<u>N</u> ext > <u>F</u>	inish C	ancel

Figure 4-26 Select SCI/SCIF Asynchronous Mode

In 'Add new configuration for selected component' dialog -> Work mode, select 'Transmission/Reception' as shown in **Figure 4-27** below.

e ² New Component	-		×
dd new configuratio	n for selected component		
SCI/SCIF Asynchronou	s Mode		
Configuration name:	Config_SCI1		
Work mode:	Transmission		~
Resource:	Transmission Reception		
	Transmission/Reception Multi-processor Transmission Multi-processor Reception Multi-processor Transmission/Reception		
?	< <u>B</u> ack <u>N</u> ext > <u>Finish</u>	Can	cel

Figure 4-27 Select Work mode – Transmission/Reception



In 'Resource', select 'SCI8' as shown in Figure 4-28 below.

e ² New Component			×
Add new configuratio	n for selected component		
SCI/SCIF Asynchronou	s Mode		
Configuration name:	Config_SCI1		
Work mode:	Transmission/Reception		\sim
Resource:	SCI1		~
	SCI1 SCI5 SCI8		
	SCI12		
?	< Back Next > Finish	Car	ncel

Figure 4-28 Select Resource – SCI8

Ensure that the 'Configuration name' updates to 'Config_SCI8' as shown in **Figure 4-29** below then click 'Finish'

² New Component		—	
dd new configuratio	n for selected component		8 8) 8 8)
SCI/SCIF Asynchronou	s Mode		
Configuration name:	Config_SCI8		
Work mode:	Transmission/Reception		~
Resource:	SCI8		~
?	< <u>B</u> ack Next >	<u>F</u> inish	Cancel

Figure 4-29 Ensure Configuration name - Config_SCI8



Configure SCI8 as shown in **Figure 4-30**. Ensure the 'Start bit edge detection' is set as 'Falling edge on RXD8 pin' and the 'Bit rate' is set to 19200 bps. All other settings remain at their defaults.

Components $\downarrow^a_Z \boxdot \blacksquare \rightrightarrows \checkmark$	Configure		
type filter text	Start bit edge detection setting O Low level on RXD8 pin Data length setting O 9 bits	 Falling edge on RXD8 pin 8 bits 	⊖ 7 bits
 ✓ Config_ICU 	Parity setting None Stop bit length setting) Even	⊖ Odd
✓ ➢ I/O Ports i Config_PORT	I bit Transfer direction setting	🔿 2 bits	
Communications Config_SCl8 Config_SCl8 Config_SCl8	States LSB-first	⊖ MSB-first	
Config_CMT0 Config_CMT1 Config_CMT2	Transfer rate setting Transfer clock Base clock Bit rate	16 cycles for 1-bit period	 (bps) (Actual value: 19176.136, Error: -0.124%)
	Enable modulation duty correction SCK8 pin function	SCK8 is not used	~
	Noise filter setting Enable noise filter Noise filter clock	Clock signal divided by 1	~ 27000000 (Hz)
	Hardware flow control setting None	○ CTS8#	O RTS8≠
	Data handling setting Transmit data handling Receive data handling		~ ~
	Interrupt setting Enable reception error interrupt (ERI8) TXI8, RXI8, TEI8, ERI8 priority	Level 15 (highest)	~
	Callback function setting	Reception end	Reception error

Figure 4-30 Config_SCI8 setting



4.5.6 SPI Clock Synchronous Mode

In the RSSKRX23W, SCI12 is used as an SPI master for the Pmod LCD on the PMOD2 connector as shown in the schematic. Click 'Add component' icon. In 'Software Component Selection' dialog -> Type, select 'Drivers'. Select 'SPI Clock Synchronous Mode' as shown in **Figure 4-31** then click 'Next'.

e ² New C	omponent		-	□ × □
Software	Component Selection			a la
Select cor	nponent from those available	e in list		-
Туре	Drivers			~
Function	All			~
Filter				
Compo	nents	Туре	Version	^
🖶 Real	Time Clock	Code Generator	1.4.0	
	CIF Asynchronous Mode	Code Generator	1.6.0	
🖶 SCI/S	CIF Clock Synchronous M	Code Generator	1.6.0	
🖶 Singl	e Scan Mode S12AD	Code Generator	1.8.0	
🖶 Smar	t Card Interface Mode	Code Generator	1.6.0	
🖶 SPI C	lock Synchronous Mode	Code Generator	1.6.0	
🖶 SPI O	peration Mode	Code Generator	1.6.0	
🖶 Volta	ge Detection Circuit	Code Generator	1.6.0	
🖶 Wato	hdog Timer	Code Generator	1.6.0	<u> </u>
<				>
Show (only latest version			
Descriptio	· ·			
	nponent provides clock syncl	transus aparation of	E PCDL or SCL (Simple	CDI 🔹
	ncludes 4 transfer modes: Sla			
	/receive and Master transmit		,	-
				\sim
Download	d more software components			
Configure	e general settings			
?	< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	Cancel

Figure 4-31 Select SPI Clock Synchronous Mode

In 'Add new configuration for selected component' dialog -> Operation, select 'Master transmit only' as shown in **Figure 4-32** below.

e ² New Component				×
Add new configuratio	n for selected component			
SPI Clock Synchronous	Mode			
Configuration name:	Config_RSPI0			
Operation:	Slave transmit/receive			\sim
Resource:	Slave transmit/receive Slave transmit only Master transmit/receive			
	Master transmit only			
	< Back Next >	Finish	Can	

Figure 4-32 Select Operation – Master transmit only



In 'Resource', select 'SCI12' as shown in **Figure 4-33** below.

e ² New Component			
Add new configuratio	n for selected component		
SPI Clock Synchronous	Mode		
Configuration name:	Config_RSPI0		
Operation:	Master transmit only		~
Resource:	RSPI0		~
	RSPI0 SCI1 SCI5 SCI8 <u>SCI12</u>		
?	< Back Next > Fini	sh	Cancel

Figure 4-33 Select Resource – SCI12

Ensure that the 'Configuration name' updates to 'Config_SCI12' as shown in **Figure 4-34** below then click 'Finish'

SPI Clock Synchronous	Mode		
Configuration name:	Config_SCI12		
Operation:	Master transmit only		~
Resource:	SCI12		~

Figure 4-34 Ensure Configuration name - Config_SCI12



Configure SCI12 as shown in **Figure 4-35**. Ensure the 'Transfer direction' is set as 'MSB-first' and the 'Bit rate' is set to 6000 kbps. All other settings remain at their defaults.

Components $\downarrow^a_{\mathbb{Z}} \boxdot \boxplus \clubsuit \checkmark$	Configure	
🐈 ᢏ	Transfer direction setting	MSB-first
✓ ⇐ Startup ✓ ⇐ Generic ✓ r_bsp	Data inversion setting Normal	◯ Inverted
✓ ➢ Drivers ✓ ➢ Interrupt ✓ Config_ICU	Transfer speed setting Transfer clock	Internal clock (SCK12 pin functions as clock output pin) $\qquad \lor$
 ✓ ▷ I/O Ports ✓ Config_PORT ✓ ▷ Communications 	Bit rate	6000 (kbps) (Actual value: 6750, Error: 12.5%)
Config_SCI12 Config_SCI8 Config_SCI8 Config_CMT0	Clock setting	Enable clock polarity inversion
Config_CMT1	Data handling setting Transmit data handling	Data handled in interrupt service routine \sim
	Interrupt setting TXI12, TEI12 priority	Level 15 (highest) \checkmark
	Callback function setting	

Figure 4-35 Config_SCI12 setting



4.5.7 Single Scan Mode S12AD

We will be using the S12AD in Single Scan Mode on the AN000 input, which is connected to the RV1 potentiometer output on the RSSK. Click 'Add component' icon. In 'Software Component Selection' dialog -> Type, select 'Drivers'. Select 'Single Scan Mode S12AD' as shown in **Figure 4-36** then click 'Next'.

e ² New Co	omponent			- 🗆	×
oftware (Component Selection				ala -
Select con	nponent from those available	e in list			
Туре	Drivers				\sim
Function	All				~
Filter					
Compor	nents	Туре	Vers	ion	^
🖶 Real 1	Time Clock	Code Generator	1.4.0)	
	CIF Asynchronous Mode	Code Generator	1.6.0)	
SCI/S	CIF Clock Synchronous M	Code Generator	1.6.0)	
🖶 Singl	e Scan Mode S12AD	Code Generator	1.8.0)	
🖶 Smar	t Card Interface Mode	Code Generator	1.6.0)	
🖶 SPI C	lock Synchronous Mode	Code Generator	1.6.0)	
🖶 SPI O	peration Mode	Code Generator	1.6.0)	
🖶 Volta	ge Detection Circuit	Code Generator	1.6.0)	
🖶 Watc	hdog Timer	Code Generator	1.6.0)	
<					>
Show	only latest version				
Descriptio					
<u> </u>				10.01 4/0	
	ware component provides si er which the analog inputs ar				
	ig channel order.				
v					\sim
Download	more software components				
Configure	general settings				
0		N. I.	F 11		
\mathbf{O}	< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	Cano	ei

Figure 4-36 Select Single Scan Mode S12AD

In 'Add new configuration for selected component' dialog -> Resource, select 'S12AD0' as shown in **Figure 4-37** below then click 'Finish'.

e ² New Component			×
Add new configuratio	n for selected component	ł	8 8 8 8
Single Scan Mode S124	AD		
Configuration name:	Config_S12AD0		
Resource:	S12AD0		\sim
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Canc	el

Figure 4-37 Select Resource – S12AD0



Configure S12AD0 as shown in **Figure 4-38** and **Figure 4-39**. Ensure the 'Analog input channel' tick box for AN000 is checked and the 'Start trigger source' is set to 'Software trigger'. All other settings remain at their defaults.

Components	nfigure	
🖕 😜	▼ Basic setting	
type filter text	Analog input mode setting	
🗸 🗁 Startup	Double trigger mode	
V 🗁 Generic	Analog input channel setting	
 ♂ r_bsp ✓ ➢ Drivers 		AN002 AN003 AN004
V 🗁 Interrupt	AN005 AN006	
Config_ICU		AN020AN027
 A/D Converter Config_S12AD0 	Temperature sensor output	Internal reference voltage
✓ ⇒ I/O Ports	Conversion start trigger setting	
Config_PORT	Start trigger source	
Config_SCI12	Software trigger	v.
Config_SCI8		
Config_CMT0	Interrupt setting	
Config_CMT1	Enable AD conversion end interr	upt (S12ADI0) Priority Level 15 (highest) V
Config_CMT2	▼ Advance setting	
	Add/Average AD value setting	
		AN002 AN003 AN004
	AN005 AN006	AN007 AN016 AN017
	AN018 AN019	AN020 AN027
	Temperature sensor output	Internal reference voltage
	A/D conversion select	
	High-speed	O Low-current
	High-Potential reference voltage se	etting
	AVCC0	○ VREFH0
	Low-Potential reference voltage set	tting
	AVSS0	○ VREFL0
	Self diagnosis setting	
	Mode	Unused ~
	Voltage used	ov 🗸
	Disconnection detection assist setti	Unused ~
	Charge setting	
	Period	2 ADCLK 🗸
	Data registers setting	
	Data placement	Right-alignment V
	Automatic clearing	Disable automatic clearing \sim
	Addition/Average mode select	Addition mode \checkmark
	Addition count	1-time 🗸
	Data storage buffer setting	
	Disable	○ Enable
	Window function setting	○ Enable
	0	
	Window A/B operation setting	Enable commission window P
	Enable comparison window A Enable comparison window B	
	Window A/B complex condition	Window A comparison condition matched OR window B comparison condition matched (S12ADWUMELC is output in other cases)
		(srewowomeechs output in other cases)

Figure 4-38 Config_S12AD0 setting (1)



A/D comparison A setting		
Reference data 0 for comparison	0	
Reference data 1 for comparison	0	
Use comparator for AN000	Reference data 0 > A/D-converted value	\sim
Use comparator for AN001	Reference data 0 > A/D-converted value	\sim
Use comparator for AN002	Reference data 0 > A/D-converted value	\sim
Use comparator for AN003	Reference data 0 > A/D-converted value	\sim
Use comparator for AN004	Reference data 0 > A/D-converted value	\sim
Use comparator for AN005	Reference data 0 > A/D-converted value	\sim
Use comparator for AN006	Reference data 0 > A/D-converted value	\sim
Use comparator for AN007	Reference data 0 > A/D-converted value	\sim
Use comparator for AN016	Reference data 0 > A/D-converted value	\sim
Use comparator for AN017	Reference data 0 > A/D-converted value	\sim
Use comparator for AN018	Reference data 0 > A/D-converted value	\sim
Use comparator for AN019	Reference data 0 > A/D-converted value	\sim
Use comparator for AN020	Reference data 0 > A/D-converted value	\sim
Use comparator for AN027	Reference data 0 > A/D-converted value	\sim
Use comparator for Temperature sensor output	Reference data 0 > A/D-converted value	\sim
Use comparator for Internal reference voltage	Reference data 0 > A/D-converted value	\sim
A/D comparison B setting		
Reference data 0 for comparison	0	
Reference data 1 for comparison	0	
Comparison B channel	Unused	\sim
	Reference data 0 > A/D-converted value	\sim
Input sampling time setting		
AN000/Self-diagnosis	0.407 (µs) (Actual value: 0.40	7)
AN001	0.407 (µs) (Actual value: 0.40	7)
AN002	0.407 (µs) (Actual value: 0.40	7)
AN003	0.407 (µs) (Actual value: 0.40	7)
AN004	0.407 (μs) (Actual value: 0.40	7)
AN005	0.407 (µs) (Actual value: 0.40	7)
AN006	0.407 (µs) (Actual value: 0.40	7)
AN007	0.407 (µs) (Actual value: 0.40	7)
AN016-AN020, AN027	0.407 (µs) (Actual value: 0.40	7)
Temperature sensor output	5 (µs) (Actual value: 2.50	0)
Internal reference voltage	5 (µs) (Actual value: 2.50	0)
	(Total conversion time: 1.074µs)	
Event link control setting		
ELC scan end event generation condition	On completion of all scans \sim	

Figure 4-39 Config_S12AD0 setting (2)



4.6 The 'Pins' tabbed page

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

*SC_Tutorial.scfg 🛛				
Hardware Resource	Ŧ		Jªz	品
Type filter text				
👼 All				•
Clock generator				
Clock frequency accuracy n		rem	ent d	
Operating mode control	icasu	i cin	enco	
System control				
5 On-chip emulator				
Interrupt controller unit				
✓ [™] ₄ Multi-function timer pulse	unit 2			
MTU0				
MTU1				
MTU2				
MTU3				
MTU4				
A Port output enable 2				
16-bit timer pulse unit				
TPU1				
TPU2				
TPU3				
🗊 TPU4				
👽 TPU5				
✓ 4 8-bit timer				
TMR0				
TMR1				
TMR2				
TMR3				
✓ ## Serial communications inte	rface			
SCI1				
COIE			>	Ť
•				
Pin Function Pin Number				
Overview Board Clocks Components	Pin	s In	terru	ipts

Figure 4-40 The 'Pins' tabbed page

4.6.1 Change pin assignment of a software component

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

🌼 *SC_Tutorial.scfg 🛛	
Pin configuration	n
Hardware Resource	⊞ ⊡ ↓ <mark>a</mark> z <mark>&</mark> &
Type filter text	

Figure 4-41 Change view to show by Hardware Resource

Select the Config_ICU of Software Components. In the Pin Function list -> Assignment column, change the pin assignment IRQ0 to P30, IRQ1 to P31. Ensure the 'Enable' tick box of IRQ0 and IRQ1 are checked, as shown in **Figure 4-42**.

ype filter text	type filter	text		All	
 Losp r_bsp compare Match Timer Config_CMT0 Config_CMT1 Config_CMT2 Interrupt Controller Config_ICU Config_PORT SCI/SCIF Asynchronous Mode Config_SCI8 SPI Clock Synchronous Mode Config_SC12 Single Scan Mode S12AD Config_S12AD0 	Enabled	Function IRQ0 IRQ1 IRQ4 IRQ5 IRQ6 IRQ7 NMI	Assignment P 330/MTIOC48/TMRI3/POE8#/RTCIC0/RXD1/SMIS P 31/MTIOC4D/TMCI2/RTCIC1/CTS1#/RTS1#/SS1#. Not assigned Not assigned Not assigned Not assigned Not assigned	Direction I I None None None None	Remarks

Figure 4-42 Configure pin assignment - Config_ICU

Select the Config_SCI8 of Software Components. In the Pin Function list -> Assignment column, Ensure the 'Enable' tick box of RXD8 and TXD8 are checked and Assignment column of RXD8 is PC6 and TXD8 is PC7 as shown in **Figure 4-43**.

Type filter text	type filter	text			All	
 Tbsp r_bsp r_bsp r_bsp config_CMT0 Config_CMT1 Config_CMT2 Mathematical Controller Config_ICU Config_PORT Config_PORT SCI/SCIF Asynchronous Mode Config_SCI8 Config_SCI8 Config_SCI12 Single Scan Mode S12AD Config_S12AD0	Enabled	Function CTS8# RTS8# RXD8 SCK8 TXD8	Assignment Not assigned Not assigned PC6/MTIOC3C/MTCLKA/TMCI2/RXD8/SMISO8/SS Not assigned PC7/UB/MTIOC3A/MTCLKB/TMO2/TXD8/SMOSI8 	🖉 Not assigned	None I	Remarks

Figure 4-43 Configure pin assignment - Config_SCI8



Select the Config_SCI12 of Software Components. In the Pin Function list -> Assignment column, Ensure the 'Enable' tick box of SCK12 and SMOSI12 are checked and Assignment column of SCK12 is PE0, SMOSI12 is PE1 as shown in **Figure 4-44**.

Type filter text	type filter	text			All	
 Image: Solution of the second sec	Enabled	Function SCK12 SMISO12 SMOS112 SS12#	Assignment PE0/SCK12/AN016 Not assigned PE1/MTIOC4C/TXD12/TXDX12/SIOX12/SMOSI12/S Not assigned	Pin Number / G9 / Not assigned / J9 / Not assigned	Direction IO None None	Remarks

Figure 4-44 Configure pin assignment - Config_SCI12

Select the Config_S12AD0 of software components. In the Pin Function list -> Assignment column, Ensure the 'Enable' tick box of AN000, AVCC0 and AVSS0 are checked and Assignment column of AN000 is P40 as shown in **Figure 4-45**.

Intervent Finite Control Finit Control Finite Control Finite Cont	ype filter text	type filter t	text			All	
AVSS0 # AVSS0 # A10 I	 J. r_bsp r_bsp r_bsp Config_CMT0 Config_CMT1 Config_CMT2 Sconfig_ICMZ Config_ICU Sconfig_ICU Sconfig_ICU Scl/SCIF Asynchronous Mode Config_SCI8 Single Scan Mode S12AD 		Function ADTRGO# AN000 AN001 AN002 AN003 AN004 AN005 AN005 AN005 AN005 AN005 AN007 AN016 AN017 AN018 AN019 AN020 AN027	Not assigned P40/AN000 Not assigned Not assigned	 Not assigned C9 Not assigned 	Direction None None None None None None None No	
VREFL0 / Not assigned / Not assigned None			AVSS0 VREFH0	AVSS0Not assigned	/ A10 / Not assigned		

Figure 4-45 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 4-46**.

∰ *SC_Tutorial.scfg 🔀		
Pin configuration		👸 🗎

Figure 4-46	Generate Code Button
-------------	----------------------

The Console pane should report 'Code generation is successful', as shown **Figure 4-47** below.

📮 Console 🔀	🗟 🛃 🔛 📑 🗲 🖛	📑 🗝 🗖
Smart Configurator Output		
M04000001: File generated:src\smc gen\Config SCI12\Config SCI12.h		^
M04000001: File generated:src\smc gen\Config SCI12\Config SCI12.c		
M04000001: File generated:src\smc gen\Config SCI12\Config SCI12 user.c		
M04000001: File generated:src\smc gen\Config S12AD0\Config S12AD0.h		
M04000001: File generated:src\smc gen\Config S12AD0\Config S12AD0.c		
M04000001: File generated:src\smc gen\Config S12AD0\Config S12AD0 user.c		
M04000001: File generated:src\smc gen\general\r cg macrodriver.h		
M04000001: File generated:src\smc gen\general\r cg userdefine.h		
M04000001: File generated:src\smc gen\general\r smc entry.h		
M04000001: File generated:src\smc gen\general\r cg hardware setup.c		
M04000001: File generated:src\smc gen\general\r cg cmt.h		
M04000001: File generated:src\smc gen\general\r cg icu.h		
M04000001: File generated: <u>src\smc gen\general\r cg port.h</u>		
M04000001: File generated: <u>src\smc gen\general\r cg s12ad.h</u>		
M04000001: File generated: <u>src\smc gen\general\r cg sci.h</u>		
M04000001: File generated: <u>src\smc gen\general\r cg rspi.h</u>		
M05000012: File generated: <u>src\smc gen\r pincfg\Pin.h</u>		
M05000012: File generated: <u>src\smc gen\r pincfg\Pin.c</u>		
M06000002: File generated: <u>src\smc gen\general\r smc interrupt.c</u>		
M06000002: File generated: <u>src\smc gen\general\r smc interrupt.h</u>		
M0000002: Code generation is successful		
M03000004: File modified: <u>src\smc gen\r config\r bsp config.h</u>		
		~
<		>

Figure 4-47 Smart Configurator console

4.7 Building the Project

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

Project Explorer 💥 📄 🔄 🌣 🗖 🗖
✓ 😂 SC_Tutorial
> 🔊 Includes
🗸 📴 src
✓
> 👝 Config_CMT0
> 👝 Config_CMT1
> Gonfig_CMT2
> 🔁 Config_ICU
> 👝 Config_PORT
> 👝 Config_S12AD0
> Gonfig_SCI12
> 👝 Config_SCI8
> 🗁 general
> 🗁 r_bsp
> 🔁 r_config
> 🔁 r_pincfg
> C SC_Tutorial.c
> 🧀 trash
SC_Tutorial HardwareDebug.launch
SC_Tutorial.scfg

Figure 4-48 Generated folder structure

Switch back to the 'C/C++' perspective using the button on the top right of the e² studio workspace. Select SC_Tutorial in the Project Explorer pane, then use 'Build Project' from the 'Project' menu or the button to build the tutorial. The project will build with no errors.



5. User Code Integration

In this section, the remaining application code is added to the project. Source files found in the RSSK Web Installer are copied into the workspace and the user is directed to add code in the user areas of the code generator files.

Code must be inserted into the user code area within many Smart Configurator-generated files in this project, these user code areas are delimited by comments as follows:

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

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

5.1 **Project Settings**

Change the optimization level of the build configuration 'HardwareDebug' before building the project. With the SC_Tutorial project selected, right- click and select [Properties], or use the shortcut keys [Alt] + [Enter] to open the Properties window.	Project Explorer Image: Solution all [Hand) Includes Open in New Window Image: Solution all Pland Copy Image: Solution all Pland Paste Image: Solution all Pland Plant Image: Solution all Plant Plant
 Navigate to 'C/C++ Build -> Settings ->Compiler -> Optimization. Select 'Level 0: Do not perform optimization' from the Optimization level pull-down. 	Settings Image Configuration: Configuration: HardwareDebug [Active] Image Configurations Image Configuration: Image Configurations Image Configurations Image Configuration: Image Configuration level Image Configuration level Image Compiler Optimization level Image Configuration Image Compiler Optimization type Image Complexition Image Compiler Image Complexition Image Complexition Image Complexition: Image Complexition: Image Complexition Image Complexition: Image Complexition:
Press the 'Apply and Close' button to close Properties window.	Apply and Close Cancel



5.2 LCD Code Integration

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

- ∙ascii.c
- ∙ascii.h
- ·r_okaya_lcd.c
- •r_okaya_lcd.h

Copy these files in to the src folder below the workspace. These files will be automatically added to the project as shown in **Figure 5-1**.

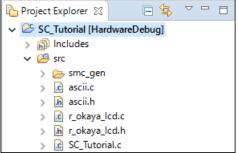


Figure 5-1 Adding files to the project



In the e² studio Project Tree, expand the 'src\smc_gen\general' folder and open the file 'r_cg_userdefine.h' by double-clicking on it. Insert the following #defines in between the user code delimiter comments as shown below.

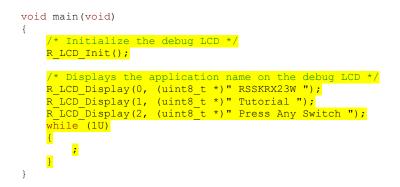
/* Start user code for macro define. Do not edit comment generated here */
#define TRUE (1)
#define FALSE (0)

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

In the e² studio Project Tree, expand the 'src' folder and open the file 'SC_Tutorial.c' by double-clicking on it. Add header files near the declaration '#include r_smc_entry.h'.

#include "r_smc_entry.h"
#include "r_okaya_lcd.h"
#include "r_cg_userdefine.h"

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



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



5.2.1 SPI Code

}

The Okaya LCD display is driven by the SPI Master that was configured using Smart Configurator in §4.5.6. In the e2 studio Project Tree, expand the 'src\smc_gen\Config_SCI12' folder and open the file 'Config_SCI12.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_SCI12_SPIMasterTransmit(uint8_t * const tx_buf, const uint16_t tx_num);

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

Now, open the Config_SCI12_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 gs_sci12_txdone;

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

Insert the following code in the transmit end call-back function for SCI12:

```
static void r_Config_SCI12_callback_transmitend(void)
```

```
/* Start user code for r_Config_SCI12_callback_transmitend. Do not edit comment generated here */
gs_sci12_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 SCI12 SPIMasterTransmit
^{\star} 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 SCI12 SPIMasterTransmit (uint8 t * const tx buf,
                             const uint16 t tx num)
{
  MD STATUS status = MD OK;
   /* Clear the flag before initiating a new transmission */
  gs sci12 txdone = FALSE;
   /* Send the data using the API */
  status = R_Config_SCI12_SPI_Master_Send(tx_buf, tx_num);
   /* Wait for the transmit end flag */
  while (FALSE == gs scil2 txdone)
  {
     /* Wait */
   }
   return (status);
End of function R SCI12 SPIMasterTransmit
```

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



5.2.2 CMT Code

The LCD code needs to insert delays to meet the timing requirements of the display module. This is achieved using the dedicated timer which was configured using Smart Configurator in §4.5.2. Open the file 'src\smc_gen\Config_CMT0\Config_CMT0.h' and insert the following code in the user area for function at the end of the file:

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

void R_CMT_MsDelay(const uint16_t millisec);

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

Open the file 'Config_CMT0_user.c' and insert the following code in the user area for global at the beginning of the file:

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

static volatile uint8_t gs_one_ms_delay_complete = FALSE;

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

Scroll down to the r_Config_CMT0_cmi0_interrupt function and insert the following line in the user code area:

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

/* Start user code for r_Config_CMT0_cmi0_interrupt. Do not edit comment generated here */

gs one ms delay complete = TRUE;

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

}

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

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

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

5.3 Additional include paths

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

e ² Properties for SC_Tutorial			– 🗆 X
type filter text Resource 	Settings		← → ⇒ ▼
 C/C++ Build Settings Tool Chain Editor C/C++ General Run/Debug Settings 	Configuration: HardwareDebug [A		Manage Configurations
	 S Common CPU PIC/PID Miscellaneous S Compiler Compiler Advanced Object List Optimization Advanced Output MISRA C Rule Check Miscellaneous 	Include file directories "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:/\${ProjName}/src/smc.g "\${workspace_loc:}\${ProjName}/src/smc.g "\${workspace_loc:}\${ProjName}/src/smc.g "\${workspace_loc:}\${ProjName}/src/smc.g "\${workspace_loc:}\${ProjName}/src/smc.g "\${workspace_loc:}\${ProjName}/src/smc.g} "\${workspace_loc:}\${ProjName}/src/smc.g} "\${workspace_loc:}\${ProjName}/src/smc.g} "\${workspace_loc:}\${ProjName}/src/smc.g} "\${workspace_loc:}\${ProjName}/src/smc.g} "\${workspace_loc:}\${ProjName}/src/smc.g} "\${workspace_loc:}\${ProjName}/src/smc.g} "\${workspace_loc:}\${ProjName}/src/smc.g} "\${workspace_loc:}\${ProjName}/src/smc.g} "\${workspace_loc:}\${ProjName}/src/smc.g}} "}	en/r_config)" en/Config_CMT0}" en/Config_CMT1}" en/Config_CUT2}" en/Config_PORT3" en/Config_SCI8}" en/Config_SCI8}" en/Config_SCI121" en/Config_SI2AD0)" en/Config_SI2AD0)" en/Config_S12AD0)"

Figure 5-2 Adding additional search paths

In the 'Add directory path' dialog, click the 'Workspace...' button and in the 'Folder selection' dialog browse to the 'SC_Tutorial/src' folder and click 'OK'. e² studio formats the path as shown in **Figure 5-3** below.

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

Figure 5-3 Adding workspace search path

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



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

The project may now be run using the debugger as described in §6. The program will display 'RSSKRX23W Tutorial Press Any Switch' on three lines in the LCD display.

5.4 Switch Code Integration

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

- rsskrx23wdef.h
- ·r_rssk_switch.c
- ·r_rssk_switch.h

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

The switch code uses interrupt code in the files Config_ICU.h, Config_ICU.c and Config_ICU_user.c and timer code in the files Config_CMT1.h, Config_CMT1.c, Config_CMT1_user.c, Config_CMT2.h, Config_CMT2.c and Config_CMT2_user.c as described in §4.5.2. and §4.5.3 It is necessary to provide additional user code in these files to implement the switch press/release detection and de-bouncing required by the API functions in r_rssk_switch.c.

5.4.1 Interrupt Code

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

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

/* Function prototypes for detecting and setting the edge trigger of ICU_IRQ */
uint8_t R_ICU_IRQIsFallingEdge(const uint8_t irq_no);
void R_ICU_IRQSetFallingEdge(const uint8_t irq_no, const uint8_t set_f_edge);
void R_ICU_IRQSetRisingEdge(const uint8_t irq_no, const uint8_t set_r_edge);

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



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

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

```
* Function Name: R ICU IRQIsFallingEdge
* Description : This function returns 1 if the specified ICU IRQ is set to
            falling edge triggered, otherwise 0.
* Arguments : uint8_t irq_no
* Return Value : 1 if falling edge triggered, 0 if not
                                         uint8 t R ICU IRQIsFallingEdge (const uint8 t irq no)
  uint8 t falling edge trig = 0x0;
  if (ICU.IRQCR[irq_no].BYTE & _04_ICU_IRQ_EDGE_FALLING)
     falling edge trig = 1;
  }
  return (falling edge trig);
}
                               End of function R ICU IROIsFallingEdge
                              *****
                    ******
      * Function Name: R ICU IRQSetFallingEdge
* Description : This function sets/clears the falling edge trigger for the
           specified ICU IRQ.
         : uint8 t irq_no
* Arguments
           uint8_t set_f_edge, 1 if setting falling edge triggered, 0 if
            clearing
* Return Value : None
                   **********
void R_ICU_IRQSetFallingEdge (const uint8_t irq_no, const uint8_t set_f_edge)
  if (1 == set f edge)
  {
     ICU.IRQCR[irq no].BYTE |= 04 ICU IRQ EDGE FALLING;
  }
  else
  {
     ICU.IRQCR[irq no].BYTE &= (uint8 t) ~ 04 ICU IRQ EDGE FALLING;
  }
}
* End of function R_ICU_IRQSetFallingEdge
                            ***********
* Function Name: R ICU IRQSetRisingEdge
* Description : This function sets/clear the rising edge trigger for the
           specified ICU IRQ.
* Arguments
          : uint8_t irq_no
           uint8_t set_r_edge, 1 if setting rising edge triggered, 0 if
            clearing
* Return Value : None
                 void R_ICU_IRQSetRisingEdge (const uint8_t irq_no, const uint8_t set_r_edge)
  if (1 == set r edge)
  {
     ICU.IRQCR[irq no].BYTE |= 08 ICU IRQ EDGE RISING;
  }
  else
  {
     ICU.IRQCR[irq_no].BYTE &= (uint8_t) ~_08_ICU_IRQ_EDGE_RISING;
  }
* End of function R ICU IRQSetRisingEdge
               * * * *
                            *****
```



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

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

/* Defines switch callback functions required by interrupt handlers */ #include "r_rssk_switch.h"

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

In the same file insert the following code in the user code area inside the function r_Config_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_irq0_interrupt:

```
/* Start user code for r_Config_ICU_irq0_interrupt. Do not edit comment generated here */
/* Switch 2 callback handler */
R_SWITCH_IsrCallback2();
/* End user code. Do not edit comment generated here */
```

5.4.2 De-bounce Timer Code

In the e² studio Project Tree, expand the 'src\smc_gen\Config_CMT1' folder and open the 'Config_CMT1_user.c' file and insert the following code in the user code area for include near the top of the file:

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

```
/* Defines switch callback functions required by interrupt handlers */
#include "r_rssk_switch.h"
```

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

In the Config_CMT1_user.c' file, insert the following code in the user code area inside the function r_Config_CMT1_cmi1_interrupt:

/* Start user code for r_Config_CMT1_cmi1_interrupt. Do not edit comment generated here */

/* Stop this timer - we start it again in the de-bounce routines */
R_Config_CMT1_Stop();
/* Call the de-bounce call back routine */
R_SWITCH DebounceIsrCallback();

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

In the e² studio Project Tree, expand the 'src\smc_gen\Config_CMT2' folder and open the file 'Config_CMT2_user.c' file and insert the following code in the user code area for include near the top of the file:

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

/* Defines switch callback functions required by interrupt handlers */ #include "r_rssk_switch.h"

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

Open the Config_CMT2_user.c file and insert the following code in the user code area inside the function r_Config_CMT2_cmi2_interrupt:

/* Start user code for r_Config_CMT2_cmi2_interrupt. Do not edit comment generated here */
/* Stop this timer - we start it again in the de-bounce routines */
R_Config_CMT2_Stop();
/* Call the de-bounce call back routine */
R_SWITCH_DebounceIsrCallback();
/* End user code. Do not edit comment generated here */

5.4.3 Main Switch and ADC Code

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

In the e2 studio Project Tree, expand the 'src\smc_gen\general' folder and open the file 'r_cg_userdefine.h'. Insert the following code the user code area, resulting in the code shown below:

/* Start user code for function. Do not edit comment generated here */
extern volatile uint8_t g_adc_trigger;
/* End user code. Do not edit comment generated here */

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

#include "r_smc_entry.h"
#include "r_okaya_lcd.h"
#include "r_cg_userdefine.h"
#include "Config_S12AD0.h"
#include "r_rssk_switch.h"
/* Variable for flagging user requested ADC conversion */
volatile uint8_t g_adc_trigger = FALSE;
/* Prototype declaration for cb switch_press */
static void cb_switch_press (void);
/* Prototype declaration for get_adc */
static uint16_t get_adc(void);
/* Prototype declaration for lcd_display_adc */
static void lcd_display_adc (const uint16_t adc_result);



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

```
void main(void)
{
    /* Initialize the switch module */
    R_SWITCH_Init();
    /* Set the call back function when SW1 or SW2 is pressed */
    R SWITCH SetPressCallback(cb_switch_press);
     /* Initialize the debug LCD */
    R LCD Init ();
     /* Displays the application name on the debug LCD */
    R_LCD_Display(0, (uint8_t *)" RSSKRX23W ");
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 = FALSE;
         }
         <mark>else</mark>
         {
              /* do nothing */
         }
    }
}
```

Then add the definition for the switch call-back, get_adc and lcd_display_adc functions below the main function, as shown below:

```
* Function Name : cb switch press
* Description : Switch press callback function. Sets g_adc_trigger flag.
* Argument
         : none
* Return value : none
           static void cb switch press (void)
{
  /* Check if switch 1 or 2 was pressed */
  if (g_switch_flag & (SWITCHPRESS_1 | SWITCHPRESS_2))
  {
     /* set the flag indicating a user requested A/D conversion is required */
    g_adc_trigger = TRUE;
     /* Clear flag */
    g switch flag = 0 \times 0;
  }
}
* End of function cb switch press
   *****
```

```
* Function Name : get adc
* Description : Reads the ADC result, converts it to a string and displays
             it on the LCD panel.
* Argument : none
* Return value : uint16 t adc value
          * * * * * *
                            static uint16_t get_adc (void)
   /* A variable to retrieve the adc result */
  uint16 t adc result = 0;
   /* Start a conversion */
  R Config S12AD0 Start();
   /* Wait for the A/D conversion to complete */
   while (FALSE == g_adc_complete)
   {
      /* Wait */
     nop();
   }
   /* Stop conversion */
  R_Config_S12AD0_Stop();
   /* Clear ADC flag */
   g adc complete = FALSE;
   R Config S12AD0 Get ValueResult(ADCHANNEL0, &adc result);
   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.
         it on the Los result : uint16_t add result
* Argument
* Return value : none
                                       *********************************
                    *****
static void lcd display adc (const uint16 t adc result)
   /* Declare a temporary variable */
  uint8 t a;
   /* Declare temporary character string */
  char lcd_buffer[11] = " ADC: XXXH";
   /* Convert ADC result into a character string, and store in the local.
    Casting to ensure use of correct data type. */
   a = (uint8 t)((adc result & 0x0F00) >> 8);
   lcd_buffer[6] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
   a = (uint8_t) ((adc_result & 0 \times 00F0) >> 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));</pre>
   /* Display the contents of the local string lcd_buffer */
   R LCD Display(3, (uint8 t *)lcd buffer);
    End of function lcd_display_adc
```



In the e² studio Project Tree, expand the 'src\smc_gen\Config_S12AD0' folder and open the file 'Config_S12AD0.h' by double-clicking on it. Insert the following code in the user code area for function, resulting in the code shown below:

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

/* Flag indicates when A/D conversion is complete */ extern volatile uint8_t g_adc_complete;

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

Open the file Config_S12AD0_user.c and insert the following code in the user code area for global, resulting in the code shown below:

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

/* Flag indicates when A/D conversion is complete */
volatile uint8_t g_adc_complete;

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

Insert the following code in the user code area of the r_Config_S12AD0_interrupt function, resulting in the code shown below:

```
static void r_Config_S12AD0_interrupt(void)
{
    /* Start user code for r_Config_S12AD0_interrupt. Do not edit comment generated here */
    g_adc_complete = TRUE;
    /* End user code. Do not edit comment generated here */
}
```

Select 'Build Project' from the 'Project' menu, or use the ⁵ button. e² studio will build the project with no errors.

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

5.5 Debug Code Integration

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

```
∙r_rssk_debug.c
∙r_rssk_debug.h
```

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

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

```
/* Macro for definition of serial debug transmit function - user edits this */
#define SERIAL_DEBUG_WRITE (R_SCI8_AsyncTransmit)
```

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



5.6 UART Code Integration

5.6.1 SCI Code

In the e² studio Project Tree, expand the 'src\smc_gen\Config_SCI8' folder and open the file 'Config_SCI8.h' by double-clicking on it. Insert the following code in the user code area at the end of the file:

/* Start user code for function. Do not edit comment generated here */
/* Exported functions used to transmit a number of bytes and wait for completion */
MD_STATUS R_SCI8_AsyncTransmit(uint8_t * const tx_buf, const uint16_t tx_num);
/* Character is used to receive key presses from PC terminal */

extern uint8_t g_rx_char;

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

Open the file 'Config_SCI8_user.c'. Insert the following code in the user area for global near the beginning of the file:

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

/* Global used to receive a character from the PC terminal */
uint8_t g_rx_char;
/* Flag used locally to detect transmission complete */

static volatile uint8_t gs_sci8_txdone;

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

In the same file, insert the following code in the user code area inside the r_Config_SCI8_callback_transmittend function:

static void r_Config_SCI8_callback_transmitend (void)

/* Start user code for r_Config_SCI8_callback_transmitend. Do not edit comment generated here */

gs_sci8_txdone = TRUE;

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



In the same file, insert the following code in the user code area inside the r_Config_SCI8_callback_receiveend function:

```
static void r_Config_SCI8_callback_receiveend(void)
{
    /* Start user code for r_Config_SCI8_callback_receiveend. Do not edit comment generated here */
    /* Check the contents of g_rx_char */
    if (('c' == g_rx_char) || ('C' == g_rx_char))
    {
        g_adc_trigger = TRUE;
    }
    /* Set up SCI8 receive buffer and callback function again */
    R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);
    /* End user code. Do not edit comment generated here */
}
```

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

```
* Function Name: R SCI8 AsyncTransmit
* Description : This function sends SCI8 data and waits for the transmit end flag.
* Arguments : tx_buf -
              transfer buffer pointer
          tx_num -
              buffer size
* Return Value : status -
             MD OK or MD ARGERROR
MD_STATUS R_SCI8_AsyncTransmit(uint8 t * const tx buf, const uint16 t tx num)
{
  MD_STATUS status = MD_OK;
  /* Clear the flag before initiating a new transmission */
  gs_sci8_txdone = FALSE;
  /* Send the data using the API */
  status = R Config SCI8 Serial Send(tx buf, tx num);
  /* Wait for the transmit end flag */
  while (FALSE == gs_sci8_txdone)
  {
     /* Wait */
  }
  return (status);
}
 * End of function R_SCI8_AsyncTransmit
*****
```



5.6.2 Main UART code

{

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

```
#include "r_smc_entry.h"
#include "r_okaya_lcd.h"
#include "r_cg_userdefine.h"
#include "Config_S12AD0.h"
#include "r_rssk_switch.h"
#include "r_rssk_debug.h"
#include "Config_SCI8.h"
/* Variable for flagging user requested ADC conversion */
volatile uint8_t g_adc_trigger = FALSE;
/* Prototype declaration for cb switch press */
static void cb_switch_press (void);
/* Prototype declaration for get adc */
static uint16_t get_adc(void);
/* Prototype declaration for lcd display adc */
static void lcd display adc (const uint16 t adc result);
/* Prototype declaration for uart_display_adc */
static void uart display adc(const uint8 t gs adc count, const uint16 t adc result);
/* Variable to store the A/D conversion count for user display */
static uint8_t gs_adc_count = 0;
```

Add the following highlighted code in the main function:

```
void main(void)
    /* Initialize the switch module */
    R SWITCH Init();
    /* Set the call back function when SW1 or SW2 is pressed */
    R SWITCH SetPressCallback(cb_switch_press);
    /* Initialize the debug LCD */
    R LCD Init();
    /* Displays the application name on the debug LCD */
    R_LCD_Display(0, (uint8_t *)" RSSKRX23W ");
R_LCD_Display(1, (uint8_t *)" Tutorial ");
    R_LCD_Display(2, (uint8_t *)" Press Any Switch ");
    /* Start the A/D converter */
    R_Config_S12AD0_Start();
    /* Set up SCI8 receive buffer and callback function */
    R Config SCI8 Serial Receive((uint8 t *)&g rx char, 1);
    /* Enable SCI8 operations */
    R Config SCI8 Start();
    while (1U)
    {
        uint16 t adc result;
        /* Wait for user requested A/D conversion flag to be set (SW1 or SW2) */
        if (TRUE == g_adc_trigger)
        {
             /* Call the function to perform an A/D conversion */
            adc result = get adc();
             /* Display the result on the LCD */
            lcd display adc(adc result);
             /* Increment the gs_adc_count */
            if (16 == (++gs_adc_count))
            {
                 gs adc count = 0;
            }
             /* Send the result to the UART */
            uart display adc(gs adc count, adc result);
```

RENESAS

```
/* Reset the flag */
  g_adc_trigger = FALSE;
}
else
{
   /* do nothing */
}
```

}

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

```
* Function Name : uart_display_adc
* Description : Converts add result to a string and sends it to the UART.

* Argument : uint8_t : gs_add_count
             uint16 t: adc result
* Return value : none
              static void uart_display_adc (const uint8_t gs_adc_count, const uint16_t adc_result)
{
   /* Declare a temporary variable */
  char a;
   /* Declare temporary character string */
   static char uart buffer[] = "ADC xH Value: xxxH\r\n";
   /* Convert ADC result into a character string, and store in the local.
    Casting to ensure use of correct data type. */
   a = (char) (gs 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));
   a = (char) ((adc result \& 0x00F0) >> 4);
   uart buffer[15] = (char)((a < 0x0A) ? (a + 0x30) : (a + 0x37));
   a = (char) (adc_result \& 0x000F);
   uart buffer[16] = (char) ((a < 0x0A) ? (a + 0x30) : (a + 0x37));
   /* Send the string to the UART */
   R DEBUG Print(uart buffer);
 End of function uart display_adc
```

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

The project may now be run using the debugger as described in §6. Connect the RSSK USBCN0 port to a USB port on a PC. If this is the first time the RSSK has been connected to the PC then a device driver will be installed automatically. Open Device Manager, the virtual COM port will be appeared under 'Port (COM & LPT)' as 'RSSK USB Serial Port (COMx)', where x is a number.

Open a terminal program, such as HyperTerminal, on the PC with the same settings as for SCI8 (Baudrate: 19200, Data Length: 8, Parity Bit: None, Stop Bit: 1, Flow Control: None).

When any switch is pressed, or when 'c' is sent via the COM port, the program will perform an A/D conversion of the voltage level on the ADPOT line and display the result on the LCD panel and send the result to the PC terminal program via the SCI8.



5.7 LED Code Integration

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

```
#include "r_smc_entry.h"
#include "r_okaya_lcd.h"
#include "r_cg_userdefine.h"
#include "Config_S12AD0.h"
#include "r_rssk_switch.h"
#include "r_rssk_debug.h"
#include "Config_SC18.h"
#include "rsskrx23wdef.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 gs_adc_count, const uint16_t adc_result);
```

```
/* Variable to store the A/D conversion count for user display */
static uint8_t gs_adc_count = 0;
```

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

Add the following highlighted code in the main function:

```
void main(void)
{
    /* Initialize the switch module */
    R_SWITCH_Init();
    /* Set the call back function when SW1 or SW2 is pressed */
    R SWITCH SetPressCallback(cb_switch_press);
    /* Initialize the debug LCD */
    R LCD Init();
    /* Displays the application name on the debug LCD */
    R LCD_Display(0, (uint8_t *)" RSSKRX23W ");
R LCD_Display(1, (uint8_t *)" Tutorial ");
R_LCD_Display(2, (uint8_t *)" Press Any Switch ");
    /* Start the A/D converter */
    R Config S12AD0 Start();
    /* Set up SCI8 receive buffer and callback function */
    R_Config_SCI8_Serial_Receive((uint8_t *)&g_rx_char, 1);
    /* Enable SCI8 operations */
    R_Config SCI8 Start();
```



```
while (1U)
{
    uint16_t adc_result;
    /* Wait for user requested A/D conversion flag to be set (SW1 or SW2) */
    if (TRUE == g adc trigger)
    {
        /* Call the function to perform an A/D conversion */
        adc result = get adc();
        /* Display the result on the LCD */
        lcd display adc(adc result);
        /* Increment the gs adc count <mark>and display using the LEDs</mark> */
        if (16 == (++gs_adc_count))
        {
            gs adc count = 0;
        led display count(gs adc count);
        /* Send the result to the UART */
        uart_display_adc(gs_adc_count, adc_result);
        /* Reset the flag */
        g_adc_trigger = FALSE;
    }
    else
    {
        /* do nothing */
    }
}
```

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

}

Select 'Build Project' from the 'Project' menu, or use the ¹ button. e² studio will build the project with no errors.

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



6. Debugging the Project

In the Project Explorer pane, ensure that the 'SC_Tutorial' project is selected. To enter the debug configurations, click upon the arrow next to the debug button and select 'Debug Configuration'.

e ² Debug Configurations		×
Create, manage, and run configurations		Ť.
Image: Second Secon	Name: SC_Tutorial HardwareDebug Image: Main the Debugger Startup Project: SC_Tutorial SC_Tutorial Image: C/C++ Application: HardwareDebug/SC_Tutorial.x Variables Build (if required) before launching Search Project Build Configuration: Search Project Build Configuration: Search Project Build Configuration: Select Automatically O Enable auto build O Disable auto build Image: Output of the Distribution of the Dis	Browse Browse
Filter matched 13 of 15 items	Re⊻ert	Appl <u>y</u>
?	<u>D</u> ebug	Close

Figure 6-1 Debug Configurations

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

Ensure that in 'Connection Settings' tab that the 'Power Target From The Emulator (MAX 200mA)' is set to Yes, and 'Main Clock Source' is set to HOCO.

For more information on powering the RSSKRX23W please refer to the User's Manual.

Name: SC_Tutorial HardwareDebug		
📄 Main 🔅 Debugger 🌔 Startup 🔲 Common 🧤 S	ource	
Debug hardware: E2 Lite (RX) V Target Device:	R5F523W8	
GDB Settings Connection Settings Debug Tool Setting	s	
✓ Clock		^
Main Clock Source	HOCO	¥
Extal Frequency[MHz]	12.0000	
Permit Clock Source Change On Writing Internal F	I Yes	¥
 Connection with Target Board 		
Emulator	(Auto)	
Connection Type	Fine	\checkmark
JTag Clock Frequency[MHz]	6.00	\sim
Fine Baud Rate[Mbps] 1.50		¥
Hot Plug	No	\sim
✓ Power		
Power Target From The Emulator (MAX 200mA)	Yes	~
Supply Voltage[V]	3.3	\sim

Figure 6-2 Connection Settings

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

Connect the E2 Lite to the PC and the RSSK E1/E2 Lite connector. Connect the Pmod LCD to the PMOD2 connector.

In the Project Explorer pane, ensure that the 'SC_Tutorial' project is selected. To debug the project, click the button. The dialog shown in **Figure 6-4** will be displayed.

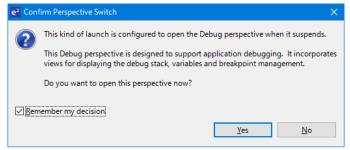


Figure 6-3 Perspective Switch Dialog

Click 'Remember my decision' to skip this dialog later. Click 'Yes' to confirm that the debug window perspective will be used. The debugger will start up and the code will stop at the Smart Configurator function 'PowerOn_Reset_PC' as shown in **Figure 6-5**.

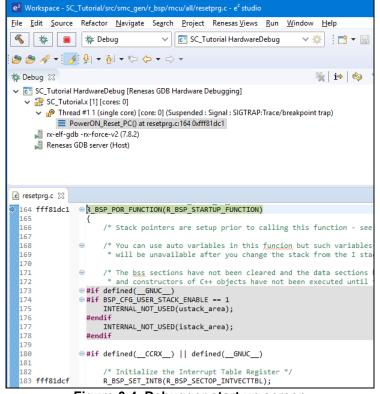


Figure 6-4 Debugger start up screen

For more information on the e² studio debugger refer to the Tutorial manual. To run the code click the **button**. The debugger will stop again at the beginning of the main function. Press **button** again to run the code.



7. Additional Information

Technical Support

For details on how to use e^2 studio, refer to the help file by opening e^2 studio, then			
selecting Help > Help Contents from the	Window	Help	
menu bar.	* 1	6	Welcome
		0	Help Contents
		2	Search
			Show Contextual Help

For information about the RX23W group microcontroller refer to 'RX23W Group User's Manual: Hardware'.

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

Technical Contact Details

Please refer to the contact details listed in section 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 GmbH.

© 2019 Renesas Electronics Europe GmbH. All rights reserved.

© 2019 Renesas Electronics Corporation. All rights reserved.



	RX23W Group
REVISION HISTORY	Renesas Solution Starter Kit for RX23W
	Smart Configurator Tutorial Manual For e ² studio

Rev.	Date	Description		
		Page	Summary	
1.00	Aug.30.19	_	First Edition issued	

RX23W Group Renesas Solution Starter Kit for RX23W Smart Configurator Tutorial Manual For e² studio

Publication Date: Rev. 1.00 Aug.30.19

Published by: Renesas Electronics Corporation

RX23W Group

