

RL78/G13

Renesas Starter Kit Software Help Manual (CubeSuite+)

RENESAS MCU
RL Family / G1X Series

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

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
 - “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
 - “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
 - “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

Disclaimer

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

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

Precautions

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

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

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

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

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

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

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

How to Use This Manual

1. Purpose and Target Readers

This manual is designed to provide the user with an understanding of the sample projects.

The manual comprises of an overview of each sample and what is required by the user to successfully execute the samples.

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

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

The following documents apply to the RL78G13 Group. Make sure to refer to the latest versions of these documents. The newest versions of the documents listed may be obtained from the Renesas Electronics Web site.

Document Type	Description	Document Title	Document No.
User's Manual	Describes the technical details of the RSK hardware.	RSKRL78G13 User's Manual	R20UT0459EG
Tutorial	Provides a guide to setting up RSK environment, running sample code and debugging programs.	RSKRL78G13 Tutorial Manual	R20UT0460EG
Quick Start Guide	Provides simple instructions to setup the RSK and run the first sample, on a single A4 sheet.	RSKRL78G13 Quick Start Guide	R20UT0461EG
Software Manual	Describes the functionality of the sample code.	RSKRL78G13 Software Help Manual	R20UT0746EG
Schematics	Full detail circuit schematics of the RSK.	RSKRL78G13 Schematics	R20UT0458EG
Hardware Manual	Provides technical details of the RL78G13 microcontroller.	RSKRL78G13 Hardware Manual	R01UH0146EJ

2. List of Abbreviations and Acronyms

Abbreviation	Full Form
ADC	Analogue to Digital Converter
DMAC	Direct Memory Access Controller
IIC	Inter Integrated Circuit
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LIN	Local Interconnect Network
LVD	Low Voltage Detection
MCU	Microcontroller Unit
PC	Personal Computer
RSK	Renesas Starter Kit
RTC	Real Time Clock
WDT	Watchdog Timer

Table of Contents

1. Overview.....	8
1.1 Purpose.....	8
2. Introduction.....	9
3. RSK Sample Code Concept.....	10
3.1 Sample Code Structure.....	10
3.2 List of Sample Code.....	11
4. Tutorial Sample.....	12
4.1 Tutorial.....	12
4.1.1 Description.....	12
4.1.2 Operation.....	12
4.2 Application.....	13
4.2.1 Description.....	13
5. Peripheral Samples.....	14
5.1 ADC_OneShot.....	14
5.1.1 Description.....	14
5.1.2 Operation.....	14
5.2 ADC_Repeat.....	14
5.2.1 Description.....	14
5.2.2 Operation.....	15
5.3 LIN.....	15
5.3.1 Description.....	15
5.3.2 Operation.....	15
5.4 Async_Serial.....	16
5.4.1 Description.....	16
5.4.2 Operation.....	16
5.5 Sync_Serial.....	16
5.5.1 Description.....	16
5.5.2 Operation.....	16
5.6 LVD.....	17
5.6.1 Description.....	17
5.6.2 Operation.....	17
5.7 IIC_Master.....	17
5.7.1 Description.....	17
5.7.2 Operation.....	17
5.8 IIC_Slave.....	18
5.8.1 Description.....	18
5.8.2 IIC Slave Commands.....	18
5.8.3 Operation.....	19
5.9 Timer.....	19
5.9.1 Description.....	19
5.9.2 Operation.....	19
5.10 DMAC.....	20
5.10.1 Description.....	20
5.10.2 Operation.....	20
5.11 WDT.....	20
5.11.1 Description.....	20
5.11.2 Operation.....	21
5.12 RTC.....	21
5.12.1 Description.....	21

5.12.2	Operation.....	21
5.13	Snooze.....	21
5.13.1	Description.....	21
5.13.2	Operation.....	22
6.	Additional Information.....	23

1. Overview

1.1 Purpose

This RSK is an evaluation tool for Renesas microcontrollers. This manual explains the operation of the sample code provided, and its interaction with CubeSuite+'s Code Generator. Code Generator is a tool within CubeSuite+ used for generating code for configuring the on-board peripherals and modules depending on user setting selections. Code Generator can also generate code for interrupt handling as well as peripheral start and stop functions.

This manual is not intended to be a tutorial on using Code Generator, or how Code Generator works – it simply aims to explain to the reader how the Code Generator was used to develop the sample codes.

2. Introduction

This document explains by text the functionality of the sample code and the steps required to observe the correct operation of each sample.

3. RSK Sample Code Concept

3.1 Sample Code Structure

The basic structure of all RSK sample code is shown in Figure 2-1 below. The first two functions, 'hwinit' and 'systeminit', configure the MCU before the main program code executes.

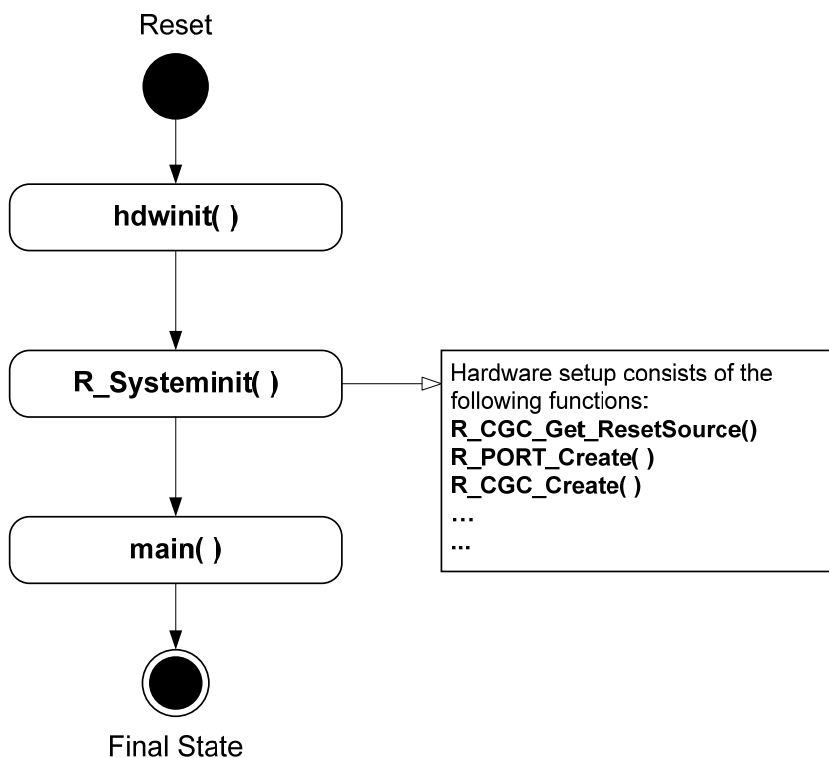


Figure 2-1: Sample Code Structure

All sample code projects are configured with the main function as the entry point after a successful build and programming of the board. The program code is automatically executed after programming the device and stopped at the first instruction inside the main function. As a result, the user will not be able to step through the hardware initialisation code. CubeSuite+ permits the user to change the program entry point by specifying another function other than main. Please refer to the Tutorial manual for more information on this.

The sample code descriptions in this manual focus on the program execution within the main function.

3.2 List of Sample Code

Table 2-2 below lists the sample code supplied with the RSKRL78G13T, and describes their function.

Sample Code	Description
ADC_Oneshot	Demonstrates usage of the 10bit and 8bit ADC module, in one shot mode.
ADC_Repeat	Demonstrates usage of the 10bit and 8bit ADC module, in repeat mode.
Application	Blank project, used for development.
Async_Serial	Demonstrates usage of the SCI module, in asynchronous mode.
DMAC	Demonstrates usage of the DMAC module, by performing interrupt requested transfers.
IIC	Demonstrates usage of the IIC module in master or slave mode.
LIN	Demonstrates usage of the LIN module in master or slave mode.
LVD	Demonstrates usage of the LVD circuit to generate interrupts on low voltage detections.
Sync_Serial	Demonstrates usage of the SCI module, in synchronous mode.
Timer	Demonstrates usage of the Timer's multifunction capabilities in 4 different modes.
Tutorial	Demonstrates basic usage of the debugger, and RSK hardware.
Watchdog	Demonstrates usage of the watchdog timer, by causing a WDT overflow interrupt.

Table 2-1: Sample Code List

4. Tutorial Sample

4.1 Tutorial

The sample code in this section is basic tutorial code, used to demonstrate basic usage of the RSK and help the user to begin writing his/her own basic sample code.

4.1.1 Description

The tutorial sample code demonstrates basic usage of the debugger and RSK hardware, and is common to all RSKs. This sample is supplied programmed onto the MCU, and executes out of the box when power is applied.

The sample calls three main functions to demonstrate port pin control, interrupt usage and C variable initialisation. These functions are shown in Figure 3-1 below.

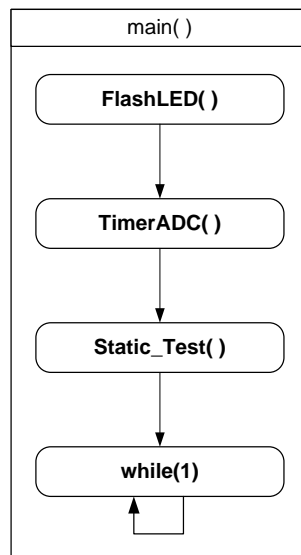


Figure 4-1: Tutorial Sample Flow

4.1.2 Operation

1. All peripherals and modules are configured prior to the program calling the main function.
2. The tutorial code initialises the LCD module, and displays 'Renesas' on the first line of the LCD, and the name of the MCU on the second line.
3. The tutorial code calls the FlashLED function, which creates a delay to toggle the LEDs repeatedly and waits in a loop until either a switch is pressed or the LEDs flash 200 times.
4. The tutorial then calls the TimerADC function which starts an ADC channel configured to run continuously. It also starts a periodic timer channel.
5. When the timer channel's period elapses, the user LEDs are toggled once. The A/D result is read then used to update the timer's period.
6. After calling TimerADC and setting up the timer & ADC interrupts, the tutorial calls the Statics_Test function.
7. The Statics_Test function displays the string STATIC on the second line of the debug LCD, and replaces it letter by letter with the constant string, TESTTEST. Once replacement is complete, the LCD reverts back to its original display. The tutorial then waits in an infinite while loop. The user LEDs toggling continues to run due to the periodic timer. The user may vary the period by varying the ADC's potentiometer shaft.

4.2 Application

4.2.1 Description

The application sample is intended as a starting platform for the user to write his/her own code. The sample includes all the necessary initialisation code and configuration settings from previous samples. The main() function contains no sample code, and performs no additional functionality.

5. Peripheral Samples

The sample code in this section provides examples of initialisation and usage of some of the MCU's peripheral modules. The sample code also provides examples of how to debug MCU peripherals.

5.1 ADC_OneShot

5.1.1 Description

This sample code demonstrates usage of the on-chip 10-bit and 8-bit analogue to digital converter (ADC), in one shot mode. The sample configures the ADC to read from the potentiometer fitted to the RSK (RV1) when user switch 'SW3' is pressed.

Note: The potentiometer is fitted to offer an easy method of supplying a variable analogue input to the microcontroller. It does not necessarily reflect the accuracy of the controllers ADC. Refer to the device hardware manual for further details.

5.1.2 Operation

1. All peripherals and modules are configured prior to the program calling the main function. The A/D channel connected to the potentiometer is initially configured for 10-bit precision operations.
2. The sample first initialises the debug LCD, and displays instructions on the screen.
3. The sample then enables the external interrupts connected to the user switches.
4. The sample then waits in an infinite while loop, and the rest of the sample's functionality is completed through interrupts.
5. When switch SW1 is pressed, the corresponding external interrupt handler is called. This starts an A/D conversion and waits for conversion to complete. After completion, the interrupt handler checks the precision mode status. The result is saved according to the precision mode before converting the integer A/D value to a specially arranged string in a buffer array. The string is then displayed on the debug LCD before the interrupt handler exits and program execution returns to the main function.
6. Pressing switch SW2 calls the corresponding external interrupt handler which changes the precision mode to 8-bits and sets the precision mode status to '2' to indicate selection of 8-bits precision. The interrupt handler exits and returns to the main function.
7. Pressing switch SW3 calls the corresponding interrupt handler which changes the precision mode to 10-bits and sets the precision mode status to '3' to indicate that the current mode is 10-bits.

5.2 ADC_Repeat

5.2.1 Description

This sample code demonstrates usage of the on-chip 10-bit and 8-bit analogue to digital converter (ADC), in repeat mode. The sample configures the ADC to repeatedly take readings of the potentiometer voltage (RV1). The sample then updates the conversion value displayed on the LCD, through periodic interrupts from the timer module.

Note: The potentiometer is fitted to offer an easy method of supplying a variable analogue input to the microcontroller. It does not necessarily reflect the accuracy of the controllers ADC. Refer the device hardware manual for further details.

5.2.2 Operation

1. All peripherals and modules are configured prior to the program calling the main function. The A/D channel connected to the potentiometer is initially configured for 10-bit precision operations.
2. The sample first initialises the debug LCD, and displays the name of the sample.
3. The sample then enables the external interrupts connected to the user switches SW1 and SW3.
4. The sample then starts the ADC and the periodic timer channel.
5. The sample then waits in an infinite while loop, and the rest of the sample's functionality is completed through interrupts.
6. On completion of an A/D conversion, the A/D interrupt handler is called. It checks to see which user switch was last pressed before saving the A/D result using the appropriate format, being 10-bit or 8-bit.
7. When user switch SW1 is pressed, the corresponding external interrupt handler. The handler changes the precision mode to 8-bits and sets a flag to indicate that user switch SW1 was pressed.
8. When user switch SW3 is pressed, the corresponding external interrupt handler. The handler changes the precision mode to 10-bits and sets a flag to indicate that user switch SW3 was pressed.
9. The timer channel's interrupt handler is called on each timer overflow. The A/D integer result is converted and stored in a specially arranged string in a buffer array before being displayed on the debug LCD.

5.3 LIN

5.3.1 Description

This sample code demonstrates usage of Local Interconnect Network (LIN), configured in master or slave mode, depending on user selection. The mode can be selected by including or excluding the pre-processor instruction in:

```
#define MASTER_LIN
```

Note: The slave mode requires resistor link changes. Please refer to the Description.txt file for the resistor references. This sample code requires an external 5V regulated power supply.

5.3.2 Operation

1. Using a three-way LIN connector, connect the RSKRL78G13 to the slave/master device.
2. Connect and turn on the 5V regulated power supply.
3. All peripherals and modules are configured prior to the program calling the main function.
4. The sample initialises the LCD module, and displays either "LIN Mstr" or "LIN Slav" on the first line of the debug LCD, depending on the selected operation mode.
5. The sample then enables the external interrupt INTP0.
6. The off-chip LIN transceiver is enabled and user LED1 is turned off.
7. In master mode, the sample periodically transmits the LIN header until data reception from the slave is detected. Detection is handled using interrupts. The corresponding interrupt handler stores the received data to a specified address range. On detection of slave data reception, the function LIN_Master_Receive is also called; stopping the timer operation and disabling serial operations. A checksum is calculated using the received data before being compared to the received checksum. If the checksums match, the received data is converted to a string and displayed on the debug LCD. The process is repeated infinitely until program execution is stopped.
8. In slave mode, the sample periodically awaits the LIN header transmitted by the master. The sample uses interrupts to detect the LIN header data. When the received slave address matches 0x10, function LIN_Slave_Transmit is called. This function starts an A/D conversion, waits for the conversion to complete, stores the integer result, converts it to a specially arranged string buffer array then transmits the data. The function exits and returns to main. The received slave address storage is cleared before the next header is received. The process is repeated infinitely until program execution is stopped.

5.4 Async_Serial

5.4.1 Description

This sample code demonstrates usage of serial array unit (SAU), configured in asynchronous mode. The SAU module is setup to communicate to a PC running a terminal emulator program, via an RS-232 cable.

5.4.2 Operation

1. Before the sample begins, the user should connect the RSK to a PC via an RS-232 cable and start the terminal program.
2. All peripherals and modules are configured prior to the program calling the main function.
3. The sample initialises the LCD module, and displays 'Async' on the first line and 'Serial' on the second.
4. The sample enables SAU0's channel 0 for RS-232 operations and starts the interval timer.
5. The rest of the program is handled at interrupt level.
6. The interval timer generates a periodic interrupt every 100ms, handled by the corresponding interrupt handler. The interrupt handler checks the character entered in the terminal emulator, and transmits an incrementing ASCII number (loops back to 0 after 9) to the terminal display if the character was not 'z'. LED0 is turned on to indicate transmission is in progress. If the character was 'z' transmission is stopped and LED0 is turned off.

5.5 Sync_Serial

5.5.1 Description

This sample code demonstrates usage of serial array unit (SAU), configured in synchronous mode. The SAU module is setup to perform loop back communication between two SAU channels, using a 3-wire interface.

Make the following connections:

TxD0 [JA2- pin 6]	↔	RxD1 [JA6- pin 7]
RxD0 [JA2- pin 8]	↔	TxD1 [JA6- pin 8]
SCK00 [JA2- pin 10]	↔	SCK10 [JA6-pin 10]

5.5.2 Operation

1. All peripherals and modules are configured prior to the program calling the main function.
2. The sample initialises the LCD module, and displays 'Sync' on the first line and 'Serial' on the second.
3. The sample then calls the Sync_Transfer function which clears the reserved SAU data buffers. SAU channels CSI00 and CSI10 are enabled and operational with data transfer between the two channels. Channel CSI00 is configured as the master and outputs the clocking signal to channel CSI10 which is configured as the slave. Both channel's data transmission and reception are handled using interrupts. A delay is created using a timer channel to allow the transfer to complete before the user can check the outcome. A "Failure" message will be displayed on the second line of the debug LCD when an error occurs during data transfer, otherwise data transfer between the two channels are compared at the end of transfer and "Success" is displayed on the second line of the debug LCD. If any of the data did not match, the message "Failure" is displayed instead.
4. The sample then enters an infinite while loop.

5.6 LVD

5.6.1 Description

In this sample, the LVD (Low Voltage Detection) circuit is configured to generate an interrupt when the power supply equals or falls below the detection level. The detection level is approximately 4.06Volts.

5.6.2 Operation

1. All peripherals and modules are configured prior to the program calling the main function.
2. The sample initialises the LCD module, and displays 'LVD' on the first line.
3. The sample calls enable LVD operations as well as start a periodic timer before entering an infinite while loop.
4. The rest of the sample is handled at interrupt level.
5. As long as power supply is maintained above 4.06 Volts, the period timer will generate interrupts which will synchronously flash all user LEDs. The timer's interrupt handler checks the status of a user-defined flag indicating user LEDs synchronisation. If the flag was set, user LEDs LED0-LED2 are toggled to synchronise the LEDs with LED3. The second line of the debug LCD is cleared before all user LEDs' states are toggled.
6. On detection of a low power supply, the LVD interrupt handler is called. It stops the timer and displays "Detected" on the second line of the debug LCD if the LVD flag is set and turns off user LEDs LED0-LED2. LED3 is left turned on; with the user-defined synchronisation flag cleared to indicate that the user LEDs are not synchronised. The timer is re-started.

5.7 IIC_Master

5.7.1 Description

This sample demonstrates usage of the IIC unit in master mode, by performing read and write operations to an EEPROM memory device or an RSKRL78G13 running the IIC_Slave sample. The sample is configured to work with 8-bit word addressed EEPROM devices. A total of 8 data bytes are transferred to the slave device starting from slave memory location 0. When writing to an EEPROM memory device, the second data is incremented after each write (from 0 to 9). For writes to a slave RSKRL78G13, the second data byte is incremented instead. The sample is configured for communication with an EEPROM memory device. To use an RSKRL78G13 as the slave, uncomment the following line found in source file `r_cg_typedefine.h`:

```
///define SLAVE_RL78G13
```

Note: This sample code requires an external 5V regulated power supply.

Make the following connections between the RSKRL78G13 and the slave device:

SDAA0 [JA1, pin 25]	↔	Device SDA
SCLA0 [JA1, pin 26]	↔	Device SDA
CON_5V [JA1, pin 25]	↔	Device Vcc
0v [JA1, pin 25]	↔	Device Vss

Note: Use a 4.7KΩ pull-up resistor on the data and clock line.

5.7.2 Operation

1. Connect and turn on the 5V regulated power supply.
2. All peripherals and modules are configured prior to the program calling the main function.
3. The sample initialises the LCD module, and displays "IIC Mstr" on the first line.

4. The sample then enables user-switch interrupts before calling the Master_EEPROM function.
5. Inside the Master_EEPROM function, the second buffer array location is set to 0x30 which represents the ASCII character '0'.
6. The sample enters an infinite while loop with the program never returning to the main function. Data transfer is handled using interrupts with transmission or reception selected by SW1 and SW3 switch presses, respectively. A write operation should always precede a read operation to verify data transferred to the slave device. If a read operation is done before a write, data is read from the slave device but not displayed on the second line of the LCD. Instead, the message "IIC Push" is displayed on the first line and "SW1->SW3" displayed on the second line informing the user to press SW1 then SW3.
7. Pressing switch SW2 allows the user to display the last data sent to the slave device.
8. After a write and read operations have taken place, data sent and received are compared. If data mismatch occurs, messages "IIC Mstr" "W/R Diff" are displayed on the first and second LCD lines, respectively.

5.8 IIC_Slave


5.8.1 Description

This sample demonstrates usage of the IIC unit in slave mode, by simulating a simple 8 byte EEPROM memory device.

5.8.2 IIC Slave Commands

(1) Write Operation


- To write to the simulated EEPROM, the master should send a start condition followed by the EEPROM device address (default address: 0x10), and wait for an ACK (acknowledgement) signal from the slave.
- The master should proceed by sending 8 bytes of data, including the 8bit EEPROM memory address (not used) and then wait for an ACK response from the slave. The memory address byte is disregarded since the slave saves the received bytes starting at the same address for each 8 byte transfer.
- The master should wait for an ACK response after each data byte. The slave is configured and expects to receive 8 bytes of data.
- Once the final byte has been sent, the master should send a stop signal to end the transaction.
- The simulated EEPROM's internal address pointer will auto increment with each byte written during the transfer.



(2)

(3) Read Operation

- The read operation will always start from a specified internal simulated EEPROM memory pointer, and auto increment to the next byte until the address reaches the maximum value and a stop condition is detected.
- The EEPROM slave should then reply with an ACK signal, and send the data located at the current memory location and auto increment the internal pointer to the next byte location.
- In order to read another byte, the master should send an ACK signal. The master should repeat this until 8 bytes have been read, and should end the operation with a stop condition.



Make the following connections between the RSKRL78G13 and the slave device:

SDAA0 [JA1, pin 25] ↔ Device SDA
 SCLA0 [JA1, pin 26] ↔ Device SDA

CON_5V [JA1, pin 25]	↔	Device Vcc
0v [JA1, pin 25]	↔	Device Vss

Note: Use a 4.7KΩ pull-up resistor on the data and clock line.

5.8.3 Operation

1. All peripherals and modules are configured prior to the program calling the main function.
2. The sample initialises the LCD module, and displays “IIC Slav” on the first line.
3. The sample then calls the Slave_EEPROM function to specify the start addresses of data to be sent to and received from the master device. The sample returns to the main function with data transfers handled at interrupt level.
4. Data received from the master device is stored in the 8-byte buffer array ‘Master_Data’. The same data is also sent to the master when a slave read is requested by the master.

5.9 Timer

5.9.1 Description

This sample configures the timer array unit (TAU) channels to run one of four applications. A total of four TAU channels have been used in the sample. The four TAU channels used are channel 0, channel 1, channel 3 and channel 5. The implemented applications are Square-wave output, Capture, Event Count and PWM (pulse width modulation). The project is configured to run the PWM application by default. To change the application change the following pre-processor directive definition with a value corresponding to desired application definition number:

```
#define __Current_TMR_APP 1
```

This can be found in the file r_cg_typedefine.h

5.9.2 Operation

1. All peripherals and modules are configured prior to the program calling the main function.
2. The sample initialises the LCD module.
3. Depending on the selected application, the user will observe one of the following:
 4. When PWM is selected, connect an oscilloscope to pin 9 of JA5.
 5. For PWM output the sample displays “Tr.PWM” on the first line of the debug LCD, enables switches SW1 and SW2’s external interrupts before starting TAU channel 0 which is configured to output a PWM waveform. The sample then enters the main while loop, and the rest of the functionality is performed at interrupt level. TAU channel 0’s period is set to 1 ms with the duty varying between 0%-90%. The waveform is output on channel 0’s output pin TO01 observed on pin 9 of application header JA5. Pressing switch SW1 will freeze the duty cycle from varying and display the percentage on line two of the debug LCD, subsequently, switch SW2 presses resumes the variations and clears the debug LCD’s line 2.
 6. When Square-wave output is selected, connect an oscilloscope to pin 20 of JA2.
 7. For Square-wave output the sample displays “Tr.SqOut” on the first line of the debug LCD and “JA2-20” on the second. A 1KHz square-wave is output on channel 5’s output pin TO05 observed on pin 20 of the application header JA2.
 8. For Interval measurement the sample displays “Tr.Intvl” on the first line of the debug LCD and “Push SW1” on the second. The external interrupt pin connected to switch SW1 is enabled before the sample enters an infinite while loop. The rest of the sample is handled using interrupts. TAU’s channel 5 is configured to periodically generate interrupts at 1ms intervals. Pressing down on SW1 generates an interrupt handled by the corresponding interrupt handler. The interrupt handler firstly verifies the signal edge detection before either starting

or stopping channel 5's counting. If the edge detection is falling edge detection the interrupt count variable `gTimerTick` is cleared and the edge detection is changed to rising edge detection. Channel 5's counting is then started prior to the interrupt handler exiting and returning to the main function. The count continues for the duration of SW1 being held down. TAU Channel 5's interrupt handler is called several times, incrementing `gTimer_Tick` by a value of 1, within this duration. As soon as SW1 is released, the corresponding interrupt handler is called to stop TAU channel 5's counting, check if the counting is less than 10000. If less, the hex count value is converted to a binary coded decimal (BCD) string and displayed on the second line of the debug LCD in milliseconds. If the value exceeds 10000 then the string " > 10s" is displayed instead to indicate that the duration exceeds 10 seconds. The signal detection is changed back to falling edge detection before exiting the interrupt handler and returning to the main function until the next time SW1 is pressed.

9. For Event Count the sample displays "Tr.Event" on the first line of the debug LCD and "Count: " on the second. Count variable `gEventCount` is cleared before TAU channel 3's count is started. The sample then enters an infinite while loop with the rest of the application handled by interrupts. TAU channel 3 is configured to be clocked by an external clock signal on its input signal pin TI03. This pin is multiplexed with the output signal TO03 as well as the external event interrupt signal (INTP4) connected to SW3. The external even interrupt is disabled and the falling edge signals of SW3 clocks TAU channel 3. An interrupt request is generated on each falling edge signal and handled by the corresponding interrupt handler. The interrupt handler checks if the count of detected events is less than 99 and increments the count if this is true, otherwise the count is set to one. The hex count value is converted to a string and displayed on the debug LCD's second line.

5.10 DMAC

5.10.1 Description

This sample demonstrates usage of the DMAC (Direct Memory Access Controller), by performing a DMA transfer to an incrementing location in an array when switch SW1 is pressed.

- Open the Memory window from the menu bar View → Memory → Memory1
- Scroll to address range 0xFFA00 – 0xFFBFF
- Observe the random data stored within these locations.

5.10.2 Operation

1. All peripherals and modules are configured prior to the program calling the main function.
2. The sample initialises the LCD module, and displays "DMAC" on the first line and "Push SW1" on the second. All user LEDs are turned on.
3. The sample enables switch SW1 interrupts and waits for the user to press SW1.
4. The sample then calls the `DMAC_DataTransfer` function, which configures the DMAC and channel 0 of SAU0. The SAU0's channel 0 is configured for transmission with each transmitted data byte transferred to memory by the DMAC inside a while loop. A total transfer of 512 bytes takes place before the loop is exited.
5. All user LEDs are turned off once the transfer ends. The second line of the LCD module will display "END".
6. The sample then enters an infinite while loop.
7. Stop the program execution and observe the memory contents.
8. Memory locations 0xFFA00 – 0xFFBFF should be filled up with the string "Renesas RL78G13".

5.11 WDT

5.11.1 Description

This sample demonstrates usage of the WDT (Watch Dog Timer). The WDT has been configured to generate an interrupt when the timer's count reaches 75% of its period. A TAU channel is configured to generate periodic interrupts whose period varies with the ADC value set by the potentiometer's positioning and restart the WDT's count. Before executing the program, ensure the potentiometer is turned fully counter-clockwise.

5.11.2 Operation

1. All peripherals and modules are configured prior to the program calling the main function.
2. The sample initialises the LCD module, and displays “WDT” on the first line and “Running.” on the second.
3. The sample then starts the WDT, ADC and TAU channel before entering an infinite while loop. All subsequent operations are handled by interrupts.
4. The ADC continuously converts the voltage set by the potentiometer and the result is stored in a global variable `gWDT_Period_Updater` within the ADC’s interrupt handler. The TAU channel’s interrupt handler is called on every count overflow. It stops the channel’s count, checks if the less than 75% of the period and restarts the WDT’s count if this is true. The TAU channel’s period is updated using the ADC result multiplied by a factor of 58. The user LEDs’ states are toggled and the TAU channel’s counting is restarted. Turning the potentiometer clockwise will reduce the user TAU channel’s period and user LEDs’ flashing rate; eventually resulting to the TAU channel’s period being greater than the WDT’s and resulting in the WDT reaching 75% and generating an interrupt. The WDT interrupt handler stops the ADC and TAU operations, displays “75%” on the second line of the debug LCD before entering an infinite while loop. To prevent the WDT causing a device reset, the WDT’s count is restarted within the loop.
5. Reset the RSK to restart the test.

5.12 RTC

5.12.1 Description

This sample demonstrates usage of the RTC (Real Time Clock) with the time displayed on the debug LCD. The RTC is configured to use the sub-clock to generate 1 Hz periodic interrupts and output a 1Hz signal on the RTC1HZ pin which is also multiplexed with other signals, though they are not used. The 1Hz can be observed on pin 16 of J2.

5.12.2 Operation

1. All peripherals and modules are configured prior to the program calling the main function.
2. The sample initialises the LCD module, and displays “RTC” on the first line and “00:00:00” on the second.
3. The sample then enables the output of the RTC’s 1Hz signal before enabling the RTC module and entering an infinite while loop.
4. The rest of the sample is handled using interrupts.
5. Each 1Hz count, corresponding to 1 second, generates an interrupt. The interrupt handler executes the callback function `RTC_ConstPeriodInterruptCallback` which reads the RTC registers, converts them to ASCII equivalent values then displayed on the debug LCD.

5.13 Snooze

5.13.1 Description

This sample demonstrates low power standby mode by placing the CPU in stop mode to reduce current consumption. The interval timer’s interrupt is used to wake the CPU from stop mode before triggering an A/D conversion independent of the CPU (snooze). All the contents of registers, flags and data memory just before the stop mode is set are held. The I/O port output latches and output buffer statuses are also held. The snooze mode can only be specified when the internal high-speed oscillation clock is selected as the main clock.

By default, the debug LCD is not used. The LCD can be used to view the A/D conversion results. To use the LCD,

open the `r_cg_userdefine.h` file and uncomment the following line:

```
//#define USE_LCD
```

5.13.2 Operation

1. Remove R22 and connect an ammeter on J6. Set the ammeter for current measurement.
2. All peripherals and modules are configured prior to the program calling the main function.
3. The sample initialises the LCD module, displays “Snooze” on the first line and “PressSW1” on the second line if the `//#define USE_LCD` is uncommented.
4. SW1 and SW2 interrupts are enabled before the sample then enters an infinite while loop.
5. Within the loop, pressing SW1 will generate an interrupt which in turn starts an A/D conversion with the CPU operating in normal mode. Adjust the potentiometer and press SW1 to carry out another conversion. Note the current consumption.
6. Press SW2 to execute the STOP instruction and stop CPU operations. The main clock, the internal high speed oscillator, is turned off during the stop mode. Note the current consumption; this will be significantly less than the current noted in Step 5.
7. Press SW3 to start the interval timer. The interval timer’s period end interrupt causes the CPU to exit the stop mode, and trigger an A/D conversion without operating the CPU (snooze mode). The result is displayed on the LCD, if enabled. The ADC disables the snooze mode function once the CPU is woken up.

6. Additional Information

Technical Support

For details on how to use CubeSuite+, refer to [Help > Tutorial] from the menu bar within the CubeSuite+.

For information about the RL78G13 series microcontrollers refer to the RL78G13 Group hardware manual.

For information about the RL78G13 assembly language, refer to the RL78 Series Software Manual.

Online technical support and information is available at: <http://www.renesas.com/rskrl78g13>

Technical Contact Details

csc@renesas.com

General information on Renesas Microcontrollers can be found on the Renesas website at:

<http://www.renesas.com/>

Trademarks

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

Copyright

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

© 2011 Renesas Electronics Europe Limited. All rights reserved.

© 2011 Renesas Electronics Corporation. All rights reserved.

© 2011 Renesas Solutions Corp. All rights reserved.

REVISION HISTORY	RSKRL78G13 Tutorial Manual
------------------	----------------------------

Rev.	Date	Description	
		Page	Summary
1.00	Jun 21, 2011	—	First Edition issued

Renesas Starter Kit Software Help Manual

Publication Date: Rev.1.00 Jun 21, 2011

Published by: Renesas Electronics Corporation



Renesas Electronics Corporation

<http://www.renesas.com>

SALES OFFICES

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

Renesas Electronics America Inc.

2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited

1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited

Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH

Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.

7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.

Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited

Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.

7F, No. 363 Fu Shing North Road Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.

1 harbourFront Avenue, #06-10, Keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.

Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.

11F, Samik Laviel' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

RL78/G13 Group



Renesas Electronics Corporation

R20UT0746-0100