User's Manual

32

RX63T Group

Renesas Starter Kit Tutorial Manual

RENESAS MCU
RX Family / RX600 Series

Renesas Electronics
www.renesas.com

Rev.2.00 Apr 2014

Notice

1. 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.
2. 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.
3. 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.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

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

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.
6. 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.
7. 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 systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please 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.
9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
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.

(2012.4)

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 is 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 RSK hardware functionality, and electrical characteristics. It is intended for users designing sample code on the RSK platform, using the many different incorporated peripheral devices.

The manual comprises of an overview of the capabilities of the RSK product, but does not intend to be a guide to embedded programming or hardware design. Further details regarding setting up the RSK and development environment can found in the User's manual.

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 RX63T 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.	RSKRX63T User's Manual	R20UT0957EG
Tutorial	Provides a guide to setting up RSK environment, running sample code and debugging programs.	RSKRX63T Tutorial Manual	R20UT0958EG
Quick Start Guide	Provides simple instructions to setup the RSK and run the first sample, on a single A4 sheet.	RSKRX63T Quick Start Guide	R20UT0959EG
Schematics	Full detail circuit schematics of the RSK.	RSKRX63T Schematics	R20UT0956EG
Hardware Manual	Provides technical details of the RX63T Group microcontroller.	RX63T Group Hardware Manual	R01UH0238EJ

2. List of Abbreviations and Acronyms

Abbreviation	Full Form
ADC	Analog-to-Digital Converter
API	Application Programming Interface
CMT	Compare Match Timer
CPU	Central Processing Unit
DVD	Digital Versatile Disc
E1	Renesas On-chip Debugging Emulator
GUI	Graphical User Interface
GPT	General PWM Timer
IRQ	Interrupt Request
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MTU	Multi-Function Timer Pulse Unit
PC	Personal Computer
RAM	Random Access Memory
ROM	Read Only Memory
RSK	Renesas Starter Kit
RTC	Realtime Clock
SAU	Serial Array Unit
SCI	Serial Communications Interface
SFR	Special Function Registers
TAU	Timer Array Unit
TPU	Timer Pulse Unit
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
WDT	Watchdog timer

Table of Contents

1. Overview

1.1 Purpose

This RSK is an evaluation tool for Renesas microcontrollers. This manual describes how to get the RSK tutorial started, and basic debugging operations.

1.2 Features

This RSK provides an evaluation of the following features:

- Renesas microcontroller programming
- User code debugging
- User circuitry such as switches, LEDs and a potentiometer
- Sample application
- Sample peripheral device initialisation code

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

2. Introduction

This manual is designed to answer, in tutorial form, the most common questions asked about using a Renesas Starter Kit (RSK). The tutorials help explain the following:

- How do I compile, link, download and run a simple program on the RSK?
- How do I build an embedded application?
- How do I use Renesas' tools?

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

- 'Debug' is a project built with the debugger support included.
- 'Release' is a project with optimised compile options, producing code suitable for release in a product.

Files referred to in this manual are installed using the project generator as you work through the tutorials. The tutorial examples in this manual assume that installation procedures described in the RSK Quick Start Guide have been completed. Please refer to the quick start guide for details of preparing the configuration.

These tutorials are designed to show you how to use the RSK and are not intended as a comprehensive introduction to the High-performance Embedded Workshop (HEW) debugger, compiler toolchains or the E1 emulator. Please refer to the relevant user manuals for more in-depth information.

2.1 Note Regarding Source Code

During the project generation, it is possible that the line numbers for source code illustrated in this document do not match exactly with that in the actual source files. It is also possible that the source address of instructions illustrated in this manual differs from a user's code compiled from the same source. These differences are minor, and do not affect the functionality of the sample code or the validity of this accompanying manual.

3. Tutorial Project Workspace

The workspace includes all of the files for two build configurations, ‘Build’ and ‘Release’. The tutorial code is common to both build configurations; and is designed to show how code can be written, debugged and then downloaded without the debug monitor in a ‘Release’ situation.

The build configuration menu in High-performance Embedded Workshop (HEW) allows the project to be configured such that certain files may be excluded from each of the build configurations. This allows the inclusion of the debug monitor within the Debug build, and its exclusion in the Release build. Contents of common C files are controlled with defines set up in the build configuration options and #ifdef statements within the source files. Maintaining only one set of project files means that projects are more controllable.

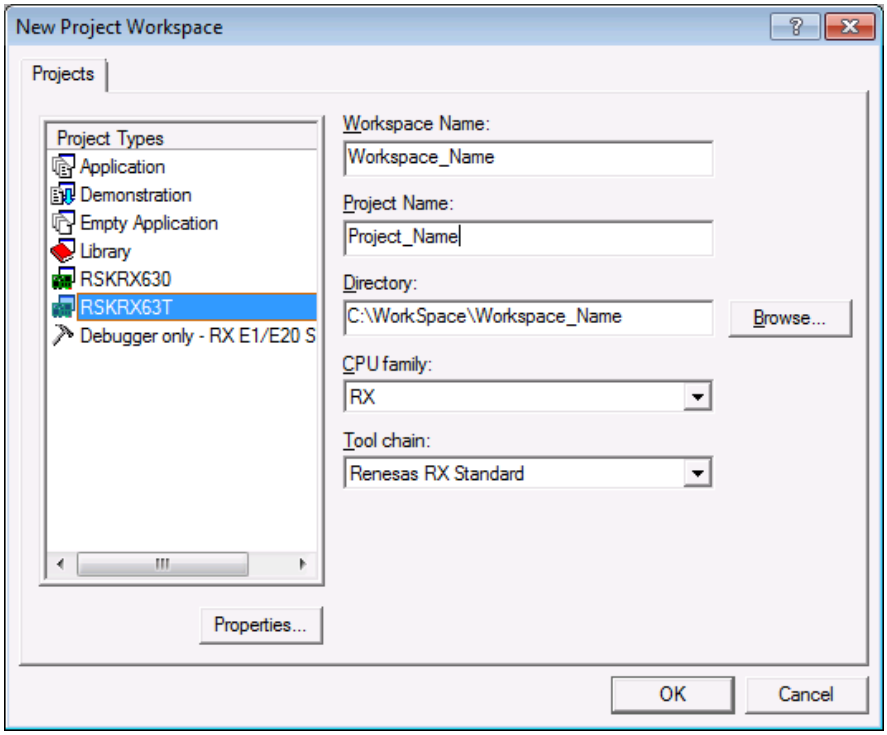
4. Project Workspace

4.1 Introduction

High-performance Embedded Workshop is an integrated development tool that allows the user to write, compile, program and debug a software project on any of the Renesas Microcontrollers. High-performance Embedded Workshop will have been installed during the installation of the software support for the Renesas Starter Kit product. This manual will describe the stages required to create and debug the supplied tutorial code.

4.2 Starting HEW

To look at the program, start High-performance Embedded Workshop from the Windows Start Menu. Open a new tutorial workspace from the [File > New Workspace…] menu or select 'Create a new project workspace' when presented with the 'Welcome!' dialog box.



The example above shows the 'New Project Workspace' dialog box with the RSKRX63T selected.

- Select the RX CPU family and 'Renesas RX Standard' toolchain.
- Select the 'RSKRX63T' project type from the left-hand projects list.
- Enter a name for the workspace – all your files will be stored under a directory with this name.
- The project name field will be pre-filled to match the workspace name above, but this name may be changed manually.
- Note: High-performance Embedded Workshop allows you to add multiple projects to a workspace. You may add the sample code projects later so you may wish to choose a suitable name for the tutorial project now.
- Click [OK] to start the Renesas Starter Kit Project Generator wizard.

The next dialog box presents the three types of example project available:

- Tutorial: this is the one of interest at this time – the code is explained later in this manual.
- Sample Code: This provides examples for using various peripherals. If you select this and click <Next> it will open a new dialog box, allowing the selection of many code examples for the peripheral modules of the device.
- Application: where the debugger is configured but there is no program code. This project is suitable for the user to add code without having to configure the debugger.

The project generator wizard will display a confirmation dialog box. Press [OK] to create the project and insert the necessary files. A tree showing all the files in this project will appear in High-performance Embedded Workshop.

To view the file 'main.c', double click on the file in the Workspace window. A new window will open showing the code.

4.3 Build Configurations and Debug Sessions

The workspace that has been created contains two build configurations and two debug sessions. The Build Configuration allows the same project to be built but with different compiler options. The options available to the user are described fully in the High-performance Embedded Workshop Manual.

4.3.1 Build Configuration

The build configurations are selected from the left hand drop down list on the toolbar. The options available are Debug and Release. The debug build is configured for use with the debugger. The Release build is configured for final ROM-programmable code.

A common difference between the two builds may be the optimisation settings. With optimisation turned on, the Debugger may seem to execute code in an unexpected order. To assist in debugging it is often helpful to turn optimisation off for the code being debugged.

• Select the 'Debug' build configuration	

4.3.2 Debug Session

The debug sessions are selected from the right hand drop down list on the toolbar. The options vary between Renesas Starter Kit types however one will always be 'DefaultSession' and the other will include the type of debug interface, in this case 'SessionRX_E1_E20_SYSTEM'. The purpose of the debug sessions is to allow the use of different debugger tools or different debugger settings on the same project.

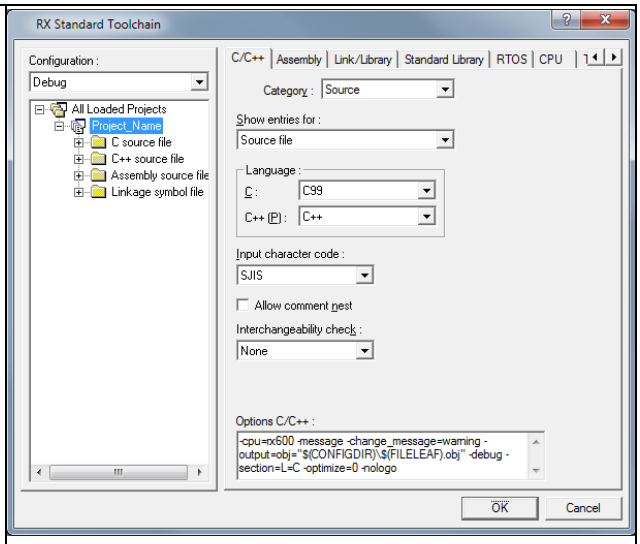
• Select the session: "SessionRX_E1_E20_SYSTEM"	

5. Building the Tutorial Program

The tutorial project build settings have been pre-configured in the toolchain options. To view the toolchain options select the ‘Build’ menu item and the relevant toolchain. This should be the first option on the drop down menu. The dialog box that is displayed will be specific to the toolchain selected.

The Configuration pane on the left hand side will exist on all the toolchain options. It is important when changing any setting to be aware of the current configuration that is being modified. If you wish to modify multiple or all build configurations this is possible by selecting ‘All’ or ‘Multiple’ from the ‘Configuration’ drop down list.

- Review the options on each of the tabs and ‘Category’ drop down lists to be aware of the options available. For the purposes of the tutorial, leave all options at default.
- When complete close the dialog box by clicking [OK]



5.1 Building Code

There is a choice of three shortcuts available for building the project.

• Selecting the ‘Build All’ toolbar button. This will build everything in the project that has not been excluded from the build. The standard library is built only once.	
• Selecting the ‘Build’ toolbar button. This will build all files that have changed since the last build. The standard library will not be built unless an option has been changed.	
• Pressing [F7]. This is equivalent to pressing the ‘Build’ button described above.	

Build the project now by pressing [F7] or pressing one of the build icons as shown above. During the build each stage will be reported in the Output Window. The build will complete with an indication of any errors and warnings encountered during the build.

5.2 Connecting the Debugger

For this tutorial it is not necessary to provide an external power supply to the board. The power will be obtained from the USB port. Please be aware that if you have too many devices connected to your USB port it may be shut down by Windows. If this happens remove some devices and try again. Alternatively provide an external power source taking care to ensure the correct polarity and voltage.

Other sample code supplied with this RSK will require a variable power supply; in which case a 5V power supply, adjustable to 0V if LVD sample is run, should be used. Refer to the RSKRX63T User Manual for further details.

The Quick Start Guide provided with the Renesas Starter Kit board gives detailed instructions on how to connect the E1 to the host computer. The following assumes that the steps in the Quick Start Guide have been followed and the E1 drivers have been installed.

- Fit the LCD module to LCD connector on the board, via the header marked 'LCD'. Ensure all the pins of the connector are correctly inserted in the socket.
- Connect the E1 Debugger to a free USB port on your computer.
- Connect the E1 Debugger to the target hardware ensuring that it is plugged into the connector marked 'E1'.
- If supplying external power to the board, it can be turned on now.

5.3 Connecting to the Target with the E1 Debugger

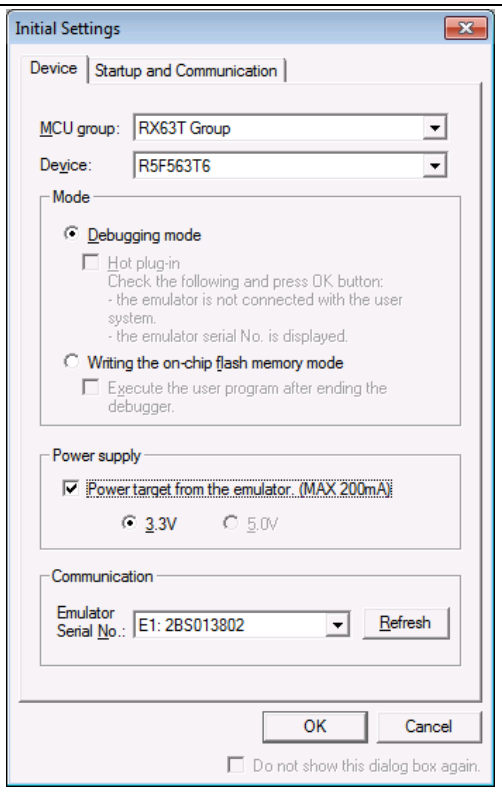
This section will take you through the process of connecting to the device, programming the Flash and executing the code.

Please note that the "Emulator Mode" wizard shown here will only appear the FIRST time you connect to the target within a project. On subsequent connections the "Emulator Setting" dialog box will appear please choose the same options to connect.

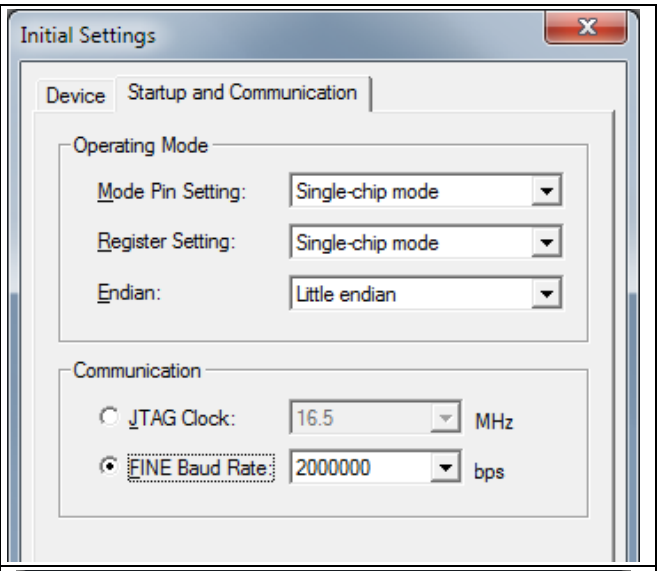
- Select the 'SessionRX_E1_E20_SYSTEM' debug session.

- Click the [Connect] button on the debug toolbar.

- The 'Initial Settings' configuration dialog box will appear. Ensure the follow configurations are set:
  - MCU group: RX63T Group
  - Device: R5F563T6
  - Mode: Debugging mode
- If the E1 is to provide power to the CPU board, select 'Power Target from Emulator' and choose the "3.3V" option. Otherwise connect a suitable power supply (refer to the RSKRX63T User Manual for details).

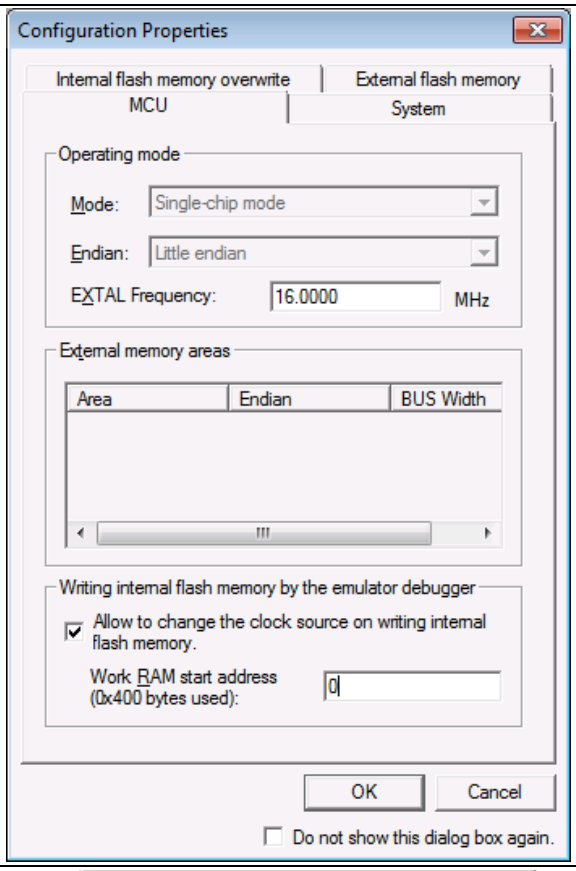


- Click the ‘Startup and Communication’ tab and ensure the Mode Pin, Register & Endian settings match the screenshot opposite. Ensure the FINE Baud Rate is set to 2000000bps. Once these settings have been confirmed, click the [OK] button to continue.
- The Flash Memory write program will be downloaded to the target.



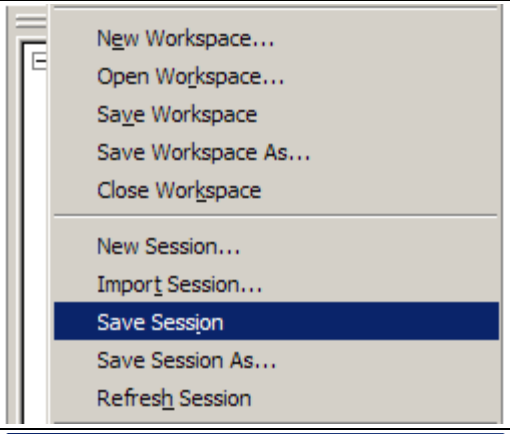
- The ‘Connecting…’ dialog box will appear, showing the status of the connection process. Under default settings, this dialog box will disappear once the connection is complete.

- Once the debugger has connected, the 'Configuration Properties' dialog box will appear.
- Ensure the following configurations are set:
  - Mode: Single-chip mode
  - Endian: Little Endian
  - Input Clock (EXTAL): 16.0000 MHz
  - Work RAM Start Address: 0
- Once the settings have been reviewed, click [OK] to proceed. The Output window in High-performance Embedded Workshop will show ‘Connected’.
- The connection to the target will activate the debugger buttons on the HEW toolbar. The function of these buttons will be explained in subsequent sections of this tutorial.



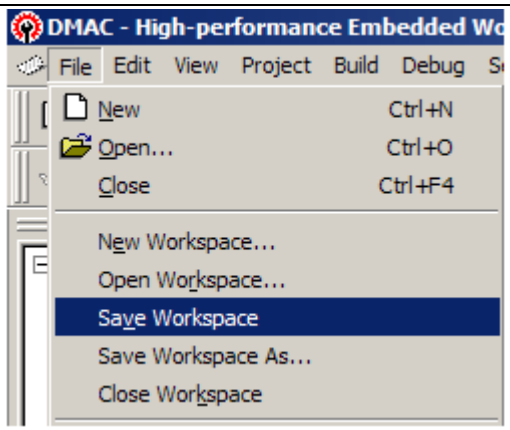
Now is a good time to save the High-performance Embedded Workshop session.

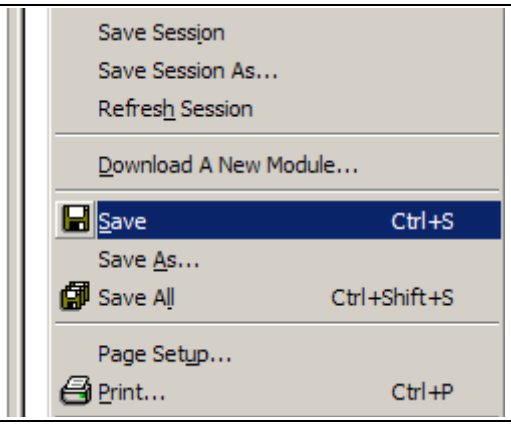
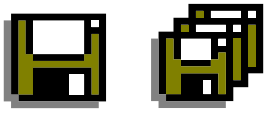
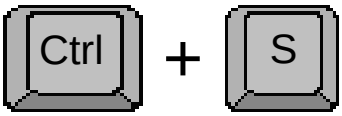
- Select ‘File’ | ‘Save Session’.



If you have changed any workspace settings now is a good time to save the workspace.

- Select ‘File’ | ‘Save Workspace’.



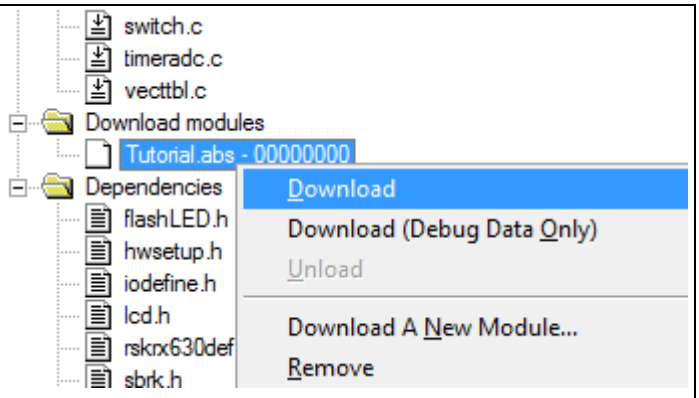
If you make any changes to files in HEW and want to preserve these change, you can save them by: • Select 'File' | 'Save'. • Or by using the Ctrl+S keyboard shortcut	
You can also save files by clicking the 'Save' or 'Save All' buttons from the HEW toolbar.	
You can also save files using the following keyboard shortcut:	

6. Downloading and Running the Tutorial

6.1 Downloading the Program Code

Now the code has been built in HEW it needs to be downloaded to the RSK. As you are now connected to the target you should see an additional category in the workspace view called ‘Download modules’

- Right click on the download module listed and select ‘Download’
- On completion the debugger and code are ready to be executed



6.2 Running the Tutorial

Once the program has been downloaded onto the RSK device, the program can be executed. Click the ‘Reset Go’ button to begin the program. It is recommended that you run through the program once first, and then continue to the review section.

7. Reviewing the Tutorial Program

This section will look at each section of the tutorial code, how it works, and how it could be altered to be implemented into more complex code.

7.1 Program Initialisation

Before the main program can run, the microcontroller must be configured. The following parts of the tutorial program are used exclusively for initialising the RSK device so that the main function can execute correctly. The initialisation code is run every time the device is reset via the reset switch or from a power reboot.

Ensuring the tutorial program has been downloaded onto the RX63T; press the 'Reset CPU' button on the Debug toolbar.	

- The File window will open the Tutorial code at the entry point. An arrow and a yellow highlight marks the current position of the program counter.
- Use these buttons to switch between 'source, disassembly and mixed modes'.


Source mode Button

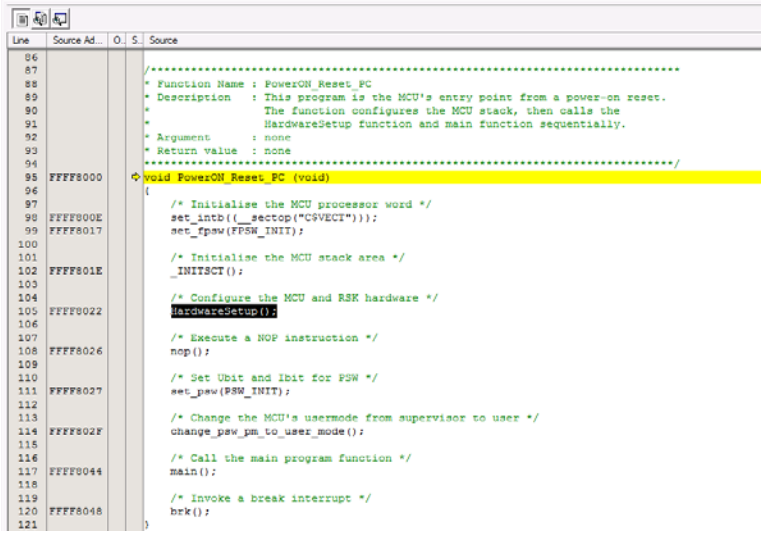

Mixed mode Button


Disassembly mode Button

Ensure the view is switched to 'source' before continuing.

```
Line  Source Ad...  O.  S.  Source
 83                         /*****************************************************************************/
 84                         /* MCU usermode switcher function declaration */
 85                         static void change_psw_pm_to_user_mode(void);
 86
 87                         /*****************************************************************************
 88                         * Function Name : PowerON_Reset_PC
 89                         * Description   : This program is the MCU's entry point from a power-on reset.
 90                         *                 The function configures the MCU stack, then calls the
 91                         *                 HardwareSetup function and main function sequentially.
 92                         * Argument      : none
 93                         * Return value  : none
 94                         *****************************************************************************/
 95  FFFF8000               void PowerON_Reset_PC (void)
 96                         {
 97                             /* Initialise the MCU processor word */
 98  FFFF800E                   set_intb((__sectop("C$VECT")));
 99  FFFF8017                   set_fpsw(FPSW_INIT);
100
101                             /* Initialise the MCU stack area */
102  FFFF801E                   _INITSCT();
103
104                             /* Configure the MCU and RSK hardware */
105  FFFF8022                   HardwareSetup();
106
107                             /* Execute a NOP instruction */
108  FFFF8026                   nop();
109
110                             /* Set Ubit and Ibit for PSW */
111  FFFF8027                   set_psw(PSW_INIT);
112
113                             /* Change the MCU's usermode from supervisor to user */
114  FFFF802F                   change_psw_pm_to_user_mode();
115
116                             /* Call the main program function */
117  FFFF8044                   main();
118
119                             /* Invoke a break interrupt */
120  FFFF8048                   brk();
121                         }
```

- Highlight the HardwareSetup function call by double-clicking the function name at line 105 as illustrated.

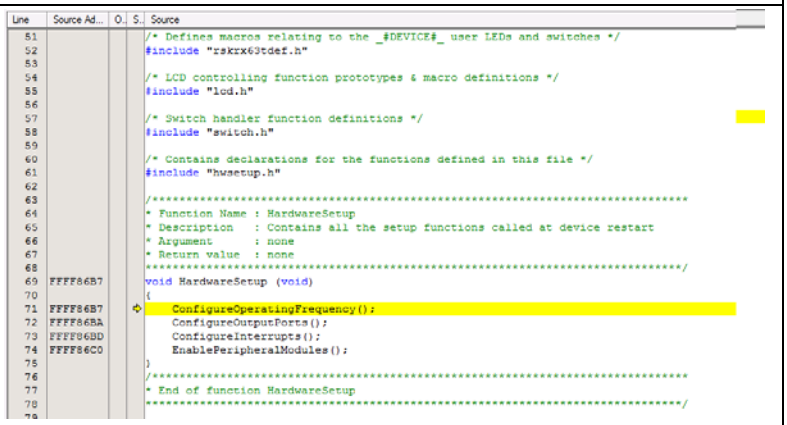


```
Line  Source Ad...  O.  S.  Source
 86
 87                         /***********************************************************************
 88                         * Function Name : PowerON_Reset_PC
 89                         * Description   : This program is the MCU's entry point from a power-on reset.
 90                         *                 The function configures the MCU stack, then calls the
 91                         *                 HardwareSetup function and main function sequentially.
 92                         * Argument      : none
 93                         * Return value  : none
 94                         ***********************************************************************/
 95  FFFF8000               void PowerON_Reset_PC (void)
 96                         {
 97                             /* Initialise the MCU processor word */
 98  FFFF800E                   set_intb((__sectop("C$VECT")));
 99  FFFF8017                   set_fpsw(FPSW_INIT);
100
101                             /* Initialise the MCU stack area */
102  FFFF801E                   _INITSCT();
103
104                             /* Configure the MCU and RSK hardware */
105  FFFF8022                   HardwareSetup();
106
107                             /* Execute a NOP instruction */
108  FFFF8026                   nop();
109
110                             /* Set Ubit and Ibit for PSW */
111  FFFF8027                   set_psw(PSW_INIT);
112
113                             /* Change the MCU's usermode from supervisor to user */
114  FFFF802F                   change_psw_pm_to_user_mode();
115
116                             /* Call the main program function */
117  FFFF8044                   main();
118
119                             /* Invoke a break interrupt */
120  FFFF8048                   brk();
121                         }
```

- Click the ‘Go to Cursor’ button to run the program up to this point.

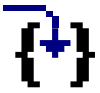
- Click ‘Step In’ to enter the HardwareSetup function.

- The program counter should now move to the HardwareSetup function definition. This function groups together several key functions that are used to ensure the device is setup correctly before the main program is executed.

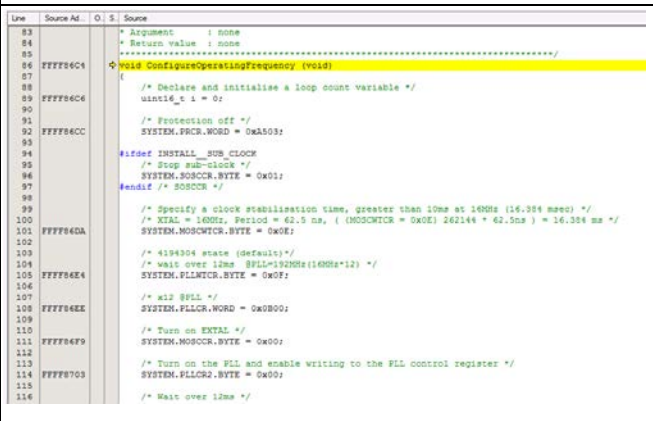


```
Line  Source Ad...  O.  S.  Source
 51                         /* Defines macros relating to the _#DEVICE#_ user LEDs and switches */
 52                         #include "rskrx63tdef.h"
 53
 54                         /* LCD controlling function prototypes & macro definitions */
 55                         #include "lcd.h"
 56
 57                         /* Switch handler function definitions */
 58                         #include "switch.h"
 59
 60                         /* Contains declarations for the functions defined in this file */
 61                         #include "hwsetup.h"
 62
 63                         /***********************************************************************
 64                         * Function Name : HardwareSetup
 65                         * Description   : Contains all the setup functions called at device restart
 66                         * Argument      : none
 67                         * Return value  : none
 68                         ***********************************************************************/
 69  FFFF86B7               void HardwareSetup (void)
 70                         {
 71  FFFF86B7                   ConfigureOperatingFrequency();
 72  FFFF86BA                   ConfigureOutputPorts();
 73  FFFF86BD                   ConfigureInterrupts();
 74  FFFF86C0                   EnablePeripheralModules();
 75                         }
 76                         /***********************************************************************
 77                         * End of function HardwareSetup
 78                         ***********************************************************************/
```

- Click ‘Step In’ again to enter the ConfigureOperatingFrequency function.

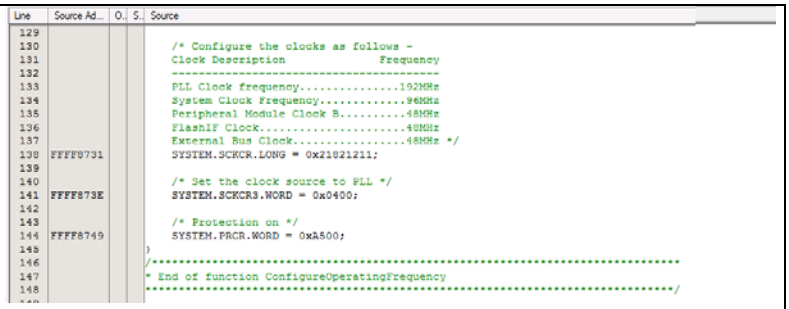


- The ConfigureOperatingFrequency function is used to set the speed of the system clocks.



```
Line  Source Ad...  O.  S.  Source
 83                         * Argument      : none
 84                         * Return value  : none
 85                         ***********************************************************************/
 86  FFFF86C4               void ConfigureOperatingFrequency (void)
 87                         {
 88                             /* Declare and initialise a loop count variable */
 89  FFFF86C6                   uint16_t i = 0;
 90
 91                             /* Protection off */
 92  FFFF86CC                   SYSTEM.PRCR.WORD = 0xA503;
 93
 94                         #ifdef INSTALL_SUB_CLOCK
 95                             /* Stop sub-clock */
 96                             SYSTEM.SOSCCR.BYTE = 0x01;
 97                         #endif /* SOSCCR */
 98
 99                             /* Specify a clock stabilisation time, greater than 10ms at 16MHz (16.384 msec) */
100                             /* XTAL = 16MHz, Period = 62.5 ns, ( (MOSCWTCR = 0x0E) 262144 * 62.5ns ) = 16.384 ms */
101  FFFF86DA                   SYSTEM.MOSCWTCR.BYTE = 0x0E;
102
103                             /* 4194304 state (default)*/
104                             /* wait over 12ms  8PLL=192MHz(16MHz*12) */
105  FFFF86E4                   SYSTEM.PLLWTCR.BYTE = 0x0F;
106
107                             /* x12 @PLL */
108  FFFF86EE                   SYSTEM.PLLCR.WORD = 0x0B00;
109
110                             /* Turn on EXTAL */
111  FFFF86F9                   SYSTEM.MOSCCR.BYTE = 0x00;
112
113                             /* Turn on the PLL and enable writing to the PLL control register */
114  FFFF8703                   SYSTEM.PLLCR2.BYTE = 0x00;
115
116                             /* Wait over 12ms */
```

- The list of initialised system clocks is shown on the opposite screenshot.
- We will now skip past the hardware setup functions to look at the tutorial's main program code.

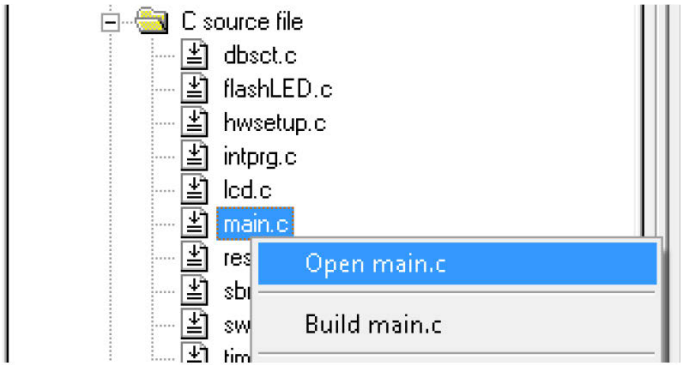


For further details regarding hardware configuration, please refer to the RSKRX63T User's Manual and the RX63T Hardware Manual.

7.2 Main Function

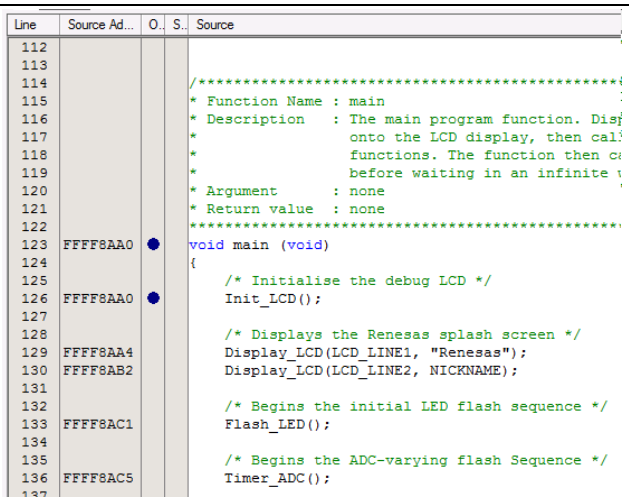
This section will examine the program code called from within the main() function.

- Find the main.c file from the file tree on the left hand side, then double click it to open the file, or right-click the filename and select ‘Open main.c’.



- Place an event at the call to main(); by double clicking in the On-Chip Breakpoint column next to the line to stop at.

  Note that two event points will appear because they share the same source address.

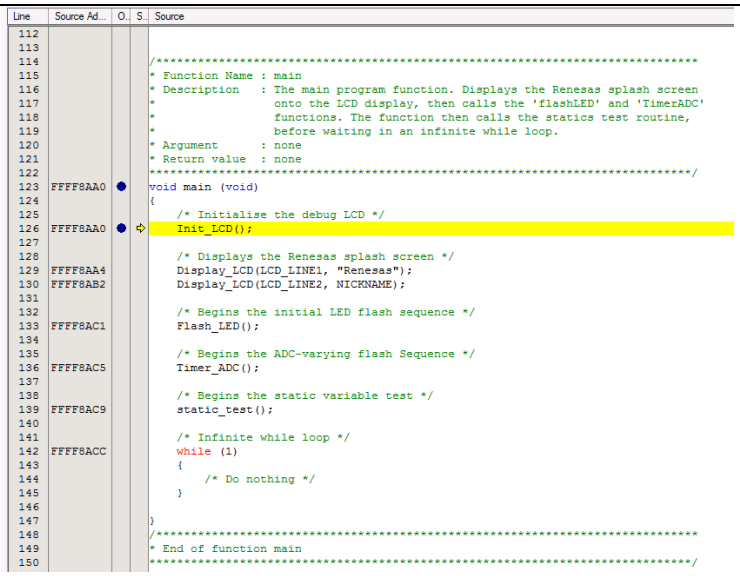


- It is also possible to set breakpoints by selecting the line to stop at and pressing the [F9] key.

The E1 emulator features advanced logic-based event point trigger system, and full instruction on its use is outside the scope of this tutorial. For further details, please refer to the RX Family E1/E20 Emulator User’s Manual

- Press ‘Reset Go’ on the Debug toolbar.

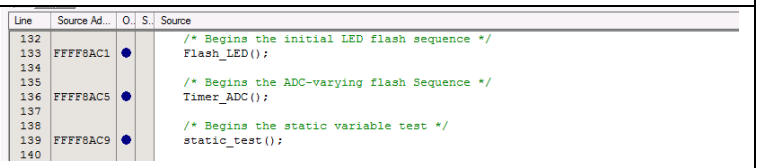
- The code will execute to the event point. At this point all the device initialisation will have been completed. The code window will open ‘main.c’ and show the new position of the program counter.



```
Line  Source Ad...  O..  S..  Source
112
113
114                          /*****************************************************************************
115                          * Function Name : main
116                          * Description   : The main program function. Displays the Renesas splash screen
117                          *                 onto the LCD display, then calls the 'flashLED' and 'TimerADC'
118                          *                 functions. The function then calls the statics test routine,
119                          *                 before waiting in an infinite while loop.
120                          * Argument      : none
121                          * Return value  : none
122                          *****************************************************************************/
123  FFFF8AA0  ●             void main (void)
124                          {
125                              /* Initialise the debug LCD */
126  FFFF8AA0  ●  ➪          Init_LCD();
127
128                              /* Displays the Renesas splash screen */
129  FFFF8AA4                    Display_LCD(LCD_LINE1, "Renesas");
130  FFFF8AB2                    Display_LCD(LCD_LINE2, NICKNAME);
131
132                              /* Begins the initial LED flash sequence */
133  FFFF8AC1                    Flash_LED();
134
135                              /* Begins the ADC-varying flash Sequence */
136  FFFF8AC5                    Timer_ADC();
137
138                              /* Begins the static variable test */
139  FFFF8AC9                    static_test();
140
141                              /* Infinite while loop */
142  FFFF8ACC                    while (1)
143                              {
144                                  /* Do nothing */
145                              }
146
147                          }
148                          /*****************************************************************************
149                          * End of function main
150                          *****************************************************************************/
```

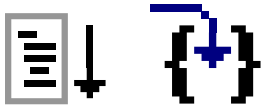
- Support for the LCD display is included in the tutorial code. We do not need to be concerned about the details of the LCD interface – except that the interface is write-only and so is not affected if the LCD display is attached or not.

- Insert event points on the Flash_LED, Timer_ADC and static_test function calls.

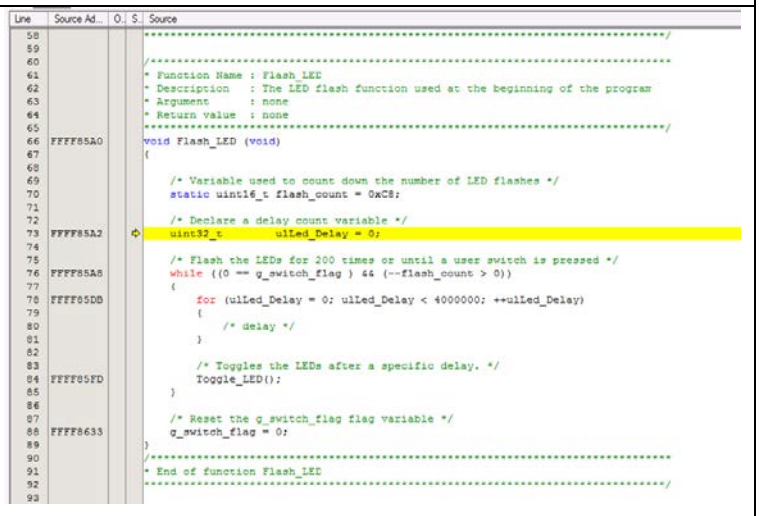


```
Line  Source Ad...  O..  S..  Source
132                              /* Begins the initial LED flash sequence */
133  FFFF8AC1  ●                 Flash_LED();
134
135                              /* Begins the ADC-varying flash Sequence */
136  FFFF8AC5  ●                 Timer_ADC();
137
138                              /* Begins the static variable test */
139  FFFF8AC9  ●                 static_test();
140
```

- Press ‘Go’ to run the program up to the Flash_LED event point, then press ‘Step In’, to move the program counter to the beginning of the Flash_LED function definition.

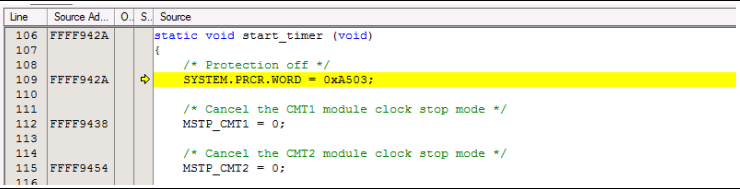


- The Flash_LED function toggles the LEDs, through the Toggle_LED function at regular intervals.
- The ‘while’ statement checks the g_switch_flag variable for switch press detections and the value of the flash_count variable, which counts down with every LED flash. Once a switch has been pressed or the count variable reaches zero, the function exits the ‘while’ loop.



```
Line  Source Ad...  O..  S..  Source
58                           *****************************************************************************/
59
60                           /*****************************************************************************
61                           * Function Name : Flash_LED
62                           * Description   : The LED flash function used at the beginning of the program
63                           * Argument      : none
64                           * Return value  : none
65                           *****************************************************************************/
66   FFFF85A0                void Flash_LED (void)
67                           {
68
69                               /* Variable used to count down the number of LED flashes */
70                               static uint16_t flash_count = 0xC8;
71
72                               /* Declare a delay count variable */
73   FFFF85A2  ➪                 uint32_t        ulLed_Delay = 0;
74
75                               /* Flash the LEDs for 200 times or until a user switch is pressed */
76   FFFF85A8                    while ((0 == g_switch_flag ) && (--flash_count > 0))
77                               {
78   FFFF85DB                        for (ulLed_Delay = 0; ulLed_Delay < 4000000; ++ulLed_Delay)
79                                   {
80                                       /* delay */
81                                   }
82
83                                   /* Toggles the LEDs after a specific delay. */
84   FFFF85FD                        Toggle_LED();
85                               }
86
87                               /* Reset the g_switch_flag flag variable */
88   FFFF8633                    g_switch_flag = 0;
89                           }
90                           /*****************************************************************************
91                           * End of function Flash_LED
92                           *****************************************************************************/
93
```

- Click ‘Go’ to resume the program, and then push any switch to proceed. The program should halt at the event point set on the Timer_ADC function call.



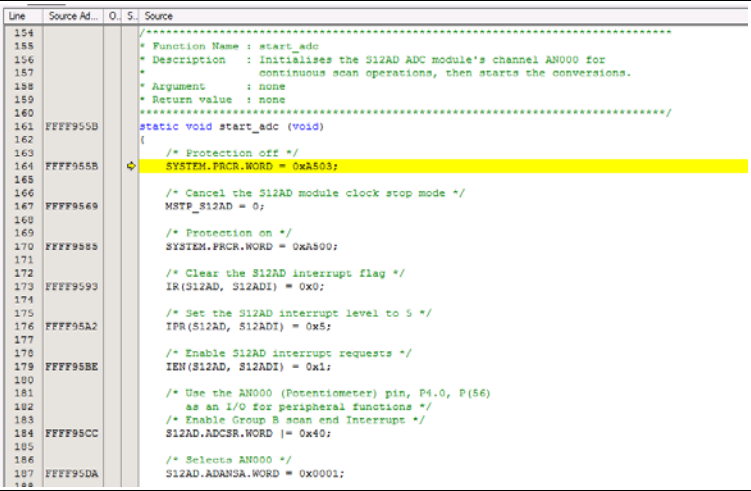
```
Line  Source Ad...  O..  S..  Source
106  FFFF942A                static void start_timer (void)
107                          {
108                              /* Protection off */
109  FFFF942A  ➪                 SYSTEM.PRCR.WORD = 0xA503;
110
111                              /* Cancel the CMT1 module clock stop mode */
112  FFFF9438                    MSTP_CMT1 = 0;
113
114                              /* Cancel the CMT2 module clock stop mode */
115  FFFF9454                    MSTP_CMT2 = 0;
116
```

- Press ‘Step In’ twice to step into the start_timer function.

- The start_timer function configures the timer used to periodically flash the LEDs.

- Press ‘Step Out’ button to exit the start_timer function, then press ‘Step In’. The program should now reach the start_adc function.

- The start_adc function configures the ADC unit to make repeat conversions of the voltage from the potentiometer RV1.

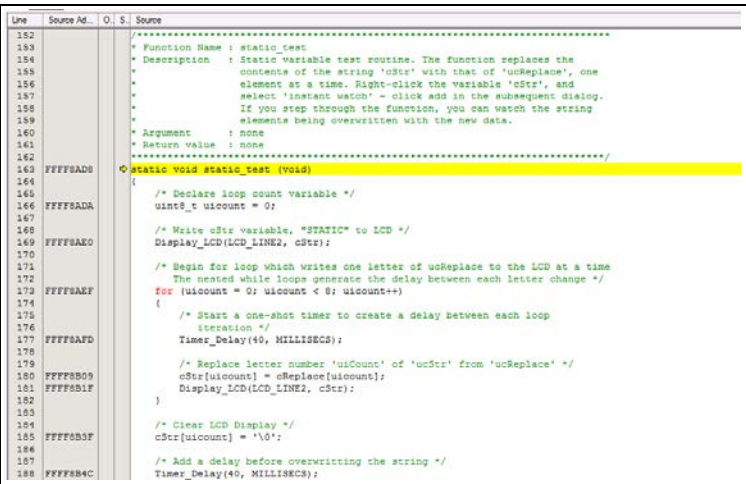


```
Line  Source Ad...  O..  S..  Source
154                          /***********************************************************************
155                          * Function Name : start_adc
156                          * Description   : Initialises the S12AD ADC module's channel AN000 for
157                          *                 continuous scan operations, then starts the conversions.
158                          * Argument      : none
159                          * Return value  : none
160                          ***********************************************************************/
161  FFFF955B                static void start_adc (void)
162                          {
163                              /* Protection off */
164  FFFF955B                    SYSTEM.PRCR.WORD = 0xA503;
165
166                              /* Cancel the S12AD module clock stop mode */
167  FFFF9569                    MSTP_S12AD = 0;
168
169                              /* Protection on */
170  FFFF956B                    SYSTEM.PRCR.WORD = 0xA500;
171
172                              /* Clear the S12AD interrupt flag */
173  FFFF9593                    IR(S12AD, S12ADI) = 0x0;
174
175                              /* Set the S12AD interrupt level to 5 */
176  FFFF95A2                    IPR(S12AD, S12ADI) = 0x5;
177
178                              /* Enable S12AD interrupt requests */
179  FFFF95BE                    IEN(S12AD, S12ADI) = 0x1;
180
181                              /* Use the AN000 (Potentiometer) pin, P4.0, P(56)
182                                 as an I/O for peripheral functions */
183                              /* Enable Group B scan end Interrupt */
184  FFFF95CC                    S12AD.ADCSR.WORD |= 0x40;
185
186                              /* Selects AN000 */
187  FFFF95DA                    S12AD.ADANSA.WORD = 0x0001;
```

- Press F5 to resume the code, where it will then halt at the break point on the static_test function call.

- Press F11 to step into the function.

- The static_test function initialises a character string with the contents of a static variable; then gradually replaces it, letter by letter, with another static string.
- Click ‘Go’ or press F5 to resume the program code. You should observe the word ‘STATIC’ appear on the second LCD line, to be gradually replaced with the string ‘TESTTEST’. The program then reverts the LCD back to the original message of ‘RX63T-L’.



```
Line  Source Ad...  O.  S..  Source
152                          /*****************************************************************************
153                          * Function Name : static_test
154                          * Description   : Static variable test routine. The function replaces the
155                          *                 contents of the string 'cStr' with that of 'ucReplace', one
156                          *                 element at a time. Right-click the variable 'cStr', and
157                          *                 select 'instant watch' - click add in the subsequent dialog.
158                          *                 If you step through the function, you can watch the string
159                          *                 elements being overwritten with the new data.
160                          * Argument      : none
161                          * Return value  : none
162                          *****************************************************************************/
163 FFFF8AD8                 static void static_test (void)
164                          {
165                              /* Declare loop count variable */
166 FFFF8ADA                     uint8_t uicount = 0;
167
168                              /* Write cStr variable, "STATIC" to LCD */
169 FFFF8AE0                     Display_LCD(LCD_LINE2, cStr);
170
171                              /* Begin for loop which writes one letter of ucReplace to the LCD at a time
172                                 The nested while loops generate the delay between each letter change */
173 FFFF8AEF                     for (uicount = 0; uicount < 8; uicount++)
174                              {
175                                  /* Start a one-shot timer to create a delay between each loop
176                                     iteration */
177 FFFF8AFD                         Timer_Delay(40, MILLISECS);
178
179                                  /* Replace letter number 'uiCount' of 'ucStr' from 'ucReplace' */
180 FFFF8B09                         cStr[uicount] = cReplace[uicount];
181 FFFF8B1F                         Display_LCD(LCD_LINE2, cStr);
182                              }
183
184                              /* Clear LCD Display */
185 FFFF8B3F                     cStr[uicount] = '\0';
186
187                              /* Add a delay before overwritting the string */
188 FFFF8B4C                     Timer_Delay(40, MILLISECS);
```

8. Additional Information

Technical Support
For details on how to use High-performance Embedded Workshop (HEW), refer to the HEW manual available on the CD or from the web site.

For information about the RX63T Group microcontrollers refer to the RX63T Group hardware manual.

For information about the RX63T assembly language, refer to the RX Family Software Manual.

Online technical support and information is available at: http://www.renesas.com/rskrx63t

Technical Contact Details

***Please refer to the contact details listed in section 7 of the* "Quick Start Guide"**

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

REVISION HISTORY	RSKRX63T Tutorial Manual

Rev.	Date	Description	
		Page	Summary
1.00	Feb 07, 2013	—	First Edition issued
2.00	Apr 14, 2014	—	Wholly-revised edition

Renesas Starter Kit Tutorial Manual

Publication Date: Rev. 2.00 Apr 14, 2014

Published by: Renesas Electronics Corporation

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

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

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, 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: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

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: +852-2265-6688, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

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.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 3.0

RX63T Group

Renesas Electronics Corporation

R20UT0958EG0200