

A horizontal bar with a blue and grey striped pattern on the left and a grey background on the right. The text is centered on the grey background.

**Renesas Starter Kit for R8C/38C
Tutorial Manual**

RENESAS SINGLE-CHIP MICROCOMPUTER
R8C FAMILY R8C/3X SERIES

Table of Contents

Chapter 1. Preface	1
Chapter 2. Introduction.....	2
Chapter 3. Tutorial Project Workspace	3
Chapter 4. Project Workspace	4
4.1. Introduction.....	4
4.2. Creating a new Project Workspace	4
4.3. Build Configurations and Debug Sessions	5
4.3.1. Build Configuration	5
4.3.2. Debug Session.....	5
Chapter 5. Building the Tutorial Project	6
5.1. Building Code	6
5.2. Connecting the debugger	6
5.3. Connecting to the target with the E8a	7
Chapter 6. Downloading and Running the Tutorial	9
Chapter 7. Project Files.....	15
7.1. Standard Project Files	15
7.1.1. Initialisation code (resetprg.c / resetprg.h)	15
7.1.2. Board initialisation code (hwsetup.c / hwsetup.h).....	16
7.1.3. Main tutorial code (main.c / main.h).....	17
Chapter 8. Additional Information.....	18

Chapter 1. Preface

Cautions

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.

Trademarks

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

Copyright

© 2010 Renesas Electronics Europe Ltd. All rights reserved.

© 2010 Renesas Electronics Corporation. All rights reserved.

© 2010 Renesas Solutions Corporation. All rights reserved.

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

Glossary

CD	Compact Disc	MCU	Microcontroller Unit
CPU	Central Processing Unit	PC	Program Counter
E8a	E8a On-chip debugger module	RAM	Random Access Memory
HEW	High-performance Embedded Workshop	ROM	Read Only Memory
LCD	Liquid Crystal Display	RSK	Renesas Starter Kit
LED	Light Emitting Diode	USB	Universal Serial Bus

Chapter 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' build demonstrating 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.

NOTE: 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, the compiler tool-chains or the E8a Emulator – please consult the relevant user manuals for more in-depth information.

Chapter 3. Tutorial Project Workspace

The workspace includes all of the files for two build configurations. The 'tutorial' code is common to both the 'Debug' and the 'Release' build configurations. The 'tutorial' 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 same files.

Maintaining only one set of project files means that projects are more controllable.

Chapter 4. Project Workspace

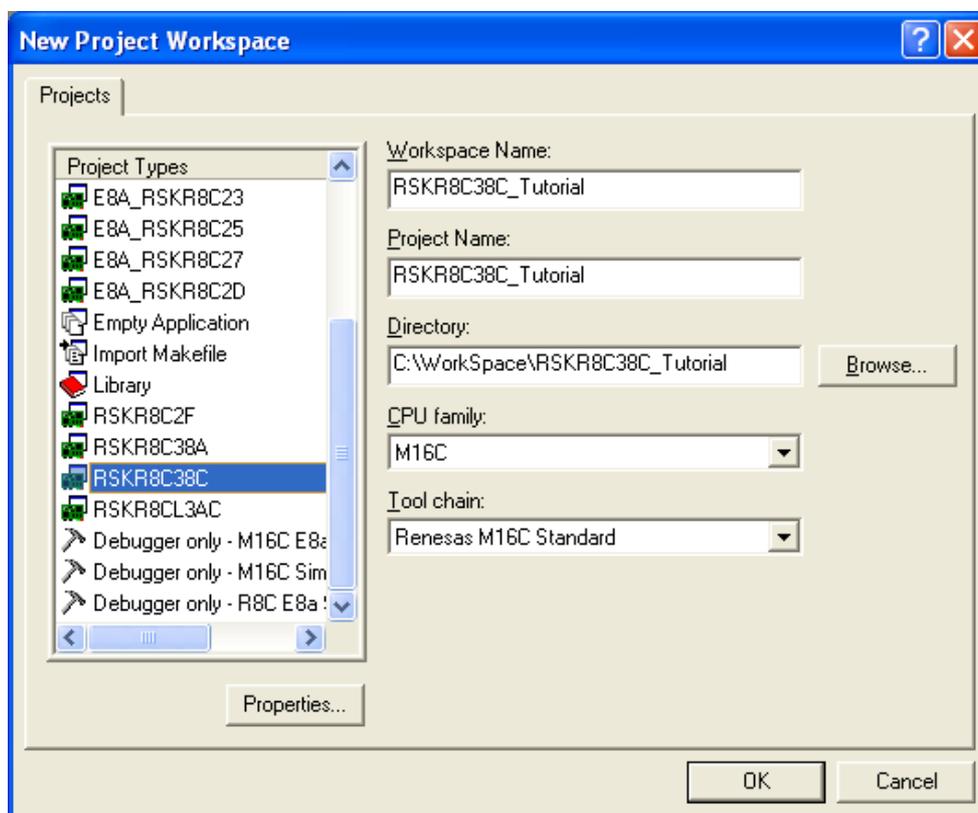
4.1. Introduction

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

4.2. Creating a new Project Workspace

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.



The example above shows the New Project Workspace dialog with the RSKR8C38C selected.

- Select CPU family as 'M16C' and Tool chain as 'Renesas M16C Standard' (R8C is supported by the same compiler as M16C).
- Select the 'RSKR8C38C' Project type from the project list for the RSK.
- 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; this name may be changed.
Note: HEW 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 RSK Project Generator wizard.

The next dialog presents the example projects available. Choose the 'Tutorial' code which will be explained later in this manual. There is also an option for 'Sample code' which provides examples for using various peripherals. This will open a new dialog allowing the selection of many code examples for the peripheral modules on the device. The final option is for an 'application' code build 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.

- Select "Tutorial" as the type of project to generate and then click "Next".
- Click "Finish" to create the project

The project generator wizard will display a confirmation dialog. Press 'OK' to create the project and insert the necessary files.

A tree showing all the files in this project will appear in HEW.

- 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 HEW User's 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-able code.

A common difference between the two builds may be the optimization settings. With the optimization turned on the debugger may seem to execute code in an unexpected order. To assist in debugging it is often helpful to turn off optimization on 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 RSK however one will always start 'Debug' and include the type of debug interface. The alternate selection will be 'DefaultSession'. The purpose of the 'Debug' session is to allow the use of different debugger tools or different debugger settings on the same project.

Select 'SessionR8C_E8a_SYSTEM' debug session.



Chapter 5. Building the Tutorial Project

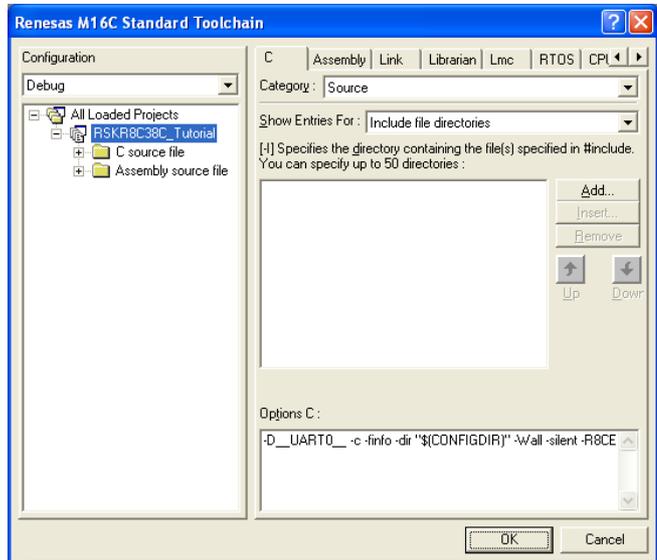
The 'tutorial' project build settings have been pre-configured in the tool-chain options. To view the tool chain options select the 'Build' Menu item and the relevant tool-chain. This should be the first option(s) on the drop down menu.

The dialog that is displayed will be specific to the tool-chain selected.

The configuration pane on the left hand side will exist on all the tool-chain 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' dropdown lists to be aware of the options available.

When complete, close the dialog box by clicking <OK>.



5.1. Building Code

There are three shortcuts available for building the project.

1. Select the 'Build All' toolbar button. 

This will build everything in the project that has not been excluded from the build. This includes the standard library.

2. Select 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.

3. Press '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 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 an external 5V power source can be connected at PWR connector taking care to ensure the correct polarity and voltage.

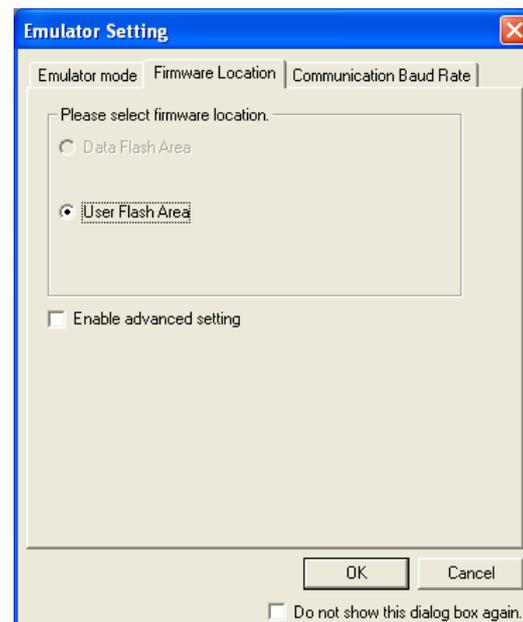
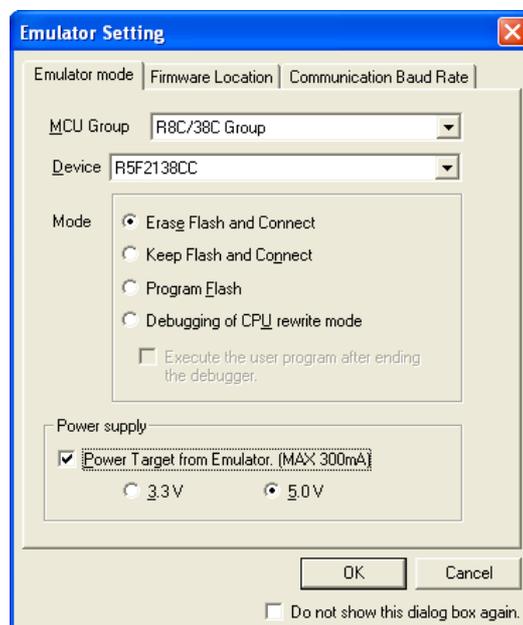
The Quick Start Guide provided with the RSK board gives detailed instructions on how to connect the E8a to the host computer. The following assumes that the steps in the Quick Start Guide have been followed and the E8a drivers have been installed.

- Fit the LCD module to the connector marked 'LCD' on the RSK, so that it lies above 'J3'. Ensure all the pins of the connector are correctly inserted in the socket.
- Connect the E8a debugger to the USB port on your computer.
- Connect the E8a debugger to the target hardware ensuring that it is plugged into the connector marked 'E8A' on the RSK.

5.3. Connecting to the target with the E8a

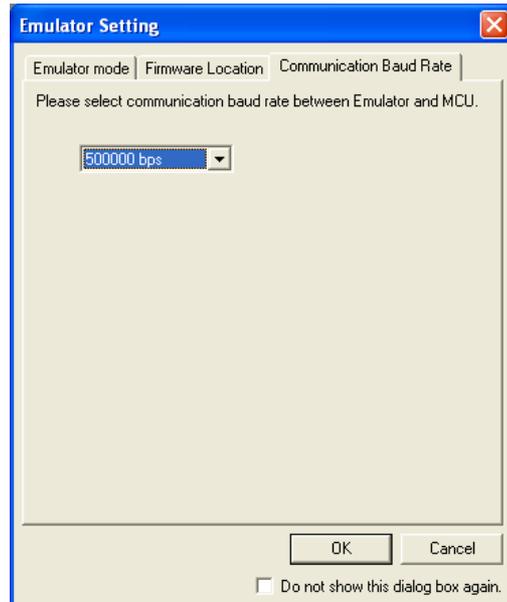
This section will take you through the process of connecting to the device, programming the Flash and executing the code.

- Select the 'SessionR8C_E8a_SYSTEM' debug session.
- Click the <Connect> button on the debug toolbar. 
- Select the correct MCU Group type (R8C/38C Group illustrated).
- Select the correct device type (R5F2138CC as illustrated).
- Select "Erase Flash and Connect".
- If the E8a is to provide power to the CPU board, select "Power Target from Emulator" and choose the "5.0V" option. Otherwise connect a 5V centre positive supply.
- Choose "User Flash Area" in "Firmware Location" tab.
- Choose "5000000bps" in "Communication Baud Rate" tab. Click <OK>.
- The flash memory write program is then downloaded to the microcontroller.



- The Output window in HEW will state 'Connected'

Note: The connection to the target will activate the debugger buttons on the HEW toolbar.



Now is a good time to save the HEW session.

- Select 'File' | 'Save Session'.

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

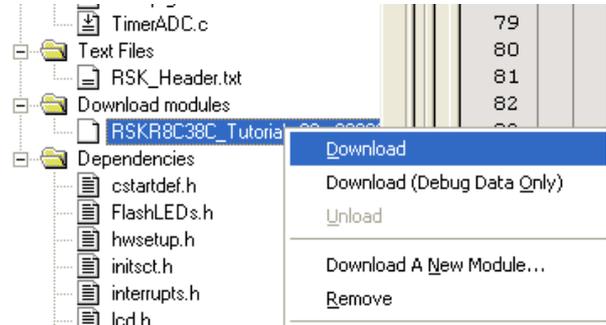
- Select 'File' | 'Save Workspace'.

Chapter 6. Downloading and Running the Tutorial

Once the code has been built in HEW it needs to be downloaded to the RSK.

There will now be an additional category in the workspace view for 'Download Modules'

- Right click on the download module listed and select 'Download'.



On completion, the debugger and code are ready to be executed.

To start debugging, we need to reset the debugger and target.

- Press 'Reset CPU' on the Debug Toolbar.



The File window should open the 'Tutorial' code at the entry point. An arrow marks the current position of the program counter.

```

/****FUNC COMMENT*****/
* Outline      : start
* Description   : Power on reset function. This function executes following to
*               : power on reset. It first calls hardware initialisation
*               : function & then 'main()' function.
* Argument     : none
* Return value  : none
/****FUNC COMMENT END*****/

void start(void)
086B0 {
086B0     /* Set interrupt stack pointer */
086B0     ↗ _isp_ = &_istack_top;
086B4     /* Change protect mode register */
086B4     prcr = 0x02U;
086B8     /* Set processor mode register */
086B8     pm0 = 0x00U;
086BB     /* Change protect mode register */
086BB     prcr = 0x00U;
086BE     /* Set flag register */
086BE     _flg_ = __F_value__;
086BE     #if __STACKSIZE__ != 0
086C2     /* Set user stack pointer */
086C2     _sp_ = &_stack_top;
086C2     #endif
086C6     /* Setting 400H (Do not change) */
086C6     _sb_ = 0x400U;
086CA     /* Set variable vector's address */
086CE     _intbh_ = 0x00U;
086CE     _asm(" ldc #(topof vector)&OFFFh, INTBL");

086D2     /* Initialize each sections */
086D2     initsct();

086D6     #if __HEAPSIZE__ != 0
086D6     /* Initialize heap */
086D6     heap_init();
086D6     #endif
086DA     #if __STANDARD_IO__ != 0
086DA     /* Initialize standard I/O */
086DA     _init();
086DE     #endif
086DE     /* Initialize FB register for debugger */
086DE     _fb_ = 0U;

086E2     /* Set up the hardware */
086E2     HardwareSetup();

086E5     /* Call main() routine */
086E5     main();

086E5     /* Call exit */
086E5     exit();
086E5 }
/*****
End of function start
*****/

```

We will now skip over the initialisation code and proceed to the main tutorial.

- Open the file called 'resetprg.c' by double clicking it in the project navigator.
- Place a breakpoint at the call to main();
Breakpoints can be set by double clicking in the column containing the PC arrow next to the line to break at; or selecting the line and pressing F9; or right click on the line and select 'Toggle breakpoint'. Alternatively set an eventpoint, by clicking in the column to the left of the breakpoint column. Maximum eight eventpoints can be set at a time. Eventpoints do not require programming the flash memory, and thus are faster to use.

- Press 'Go' on the Debug Toolbar.



The code will execute to the breakpoint. At this point, all the device initialisation will have been completed.

- Press 'Step In' on the Debug Toolbar.

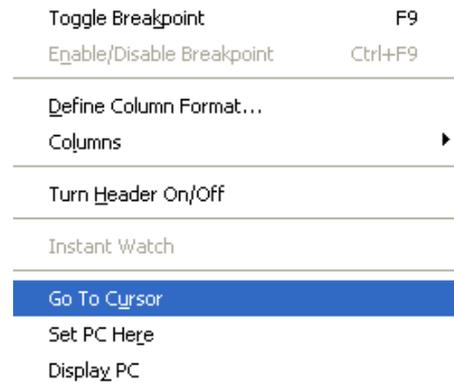


The code window will open 'main.c' and show the new position of the program counter.

```
085B0  /*"FUNC COMMENT"*****  
085B0  * Outline      : main  
085B0  * Description  : Main program. This function calls timer, ADC & LCD  
085B0             initialisation functions. The user LEDs flashes until  
085B0             the user presses a switch on the RSK.  
085B0  * Argument    : none  
085B0  * Return value : none  
085B0  /*"FUNC COMMENT END"*****/  
  
085B0  void main(void)  
085B0  {  
085B0     /* Reset the LCD module. */  
085B0     InitialiseDisplay();  
  
085B4     /* Display Renesas Splash Screen. */  
085C4     DisplayString(LCD_LINE1, "Renesas");  
085C4     DisplayString(LCD_LINE2, NICKNAME);  
  
085D5     /* Flash the user LEDs for some time or until a push button is pressed. */  
085D5     FlashLEDs();  
  
085D9     /* Flash the user LEDs at a rate set by the user potentiometer (ADC) using  
085D9             interrupts. */  
085D9     TimerADC();  
  
085DD     /* Demonstration of initialised variables. Use this function with the  
085DD             debugger. */  
085DD     Statics_Test();  
  
085E0     /* This function must not exit */  
085E4     while(1);  
085E4 }  
085E4  
085E4  /*****  
085E4  End of function main  
085E4  *****/
```

- Insert a breakpoint on the call to the 'TimerADC()' function.

- Right click on the 'FlashLEDs();' function and select 'Go to cursor'.



The code will execute to the selected line and stop. An automatic breakpoint was inserted in the code and then removed after calling the break.

- Press 'Step Over' on the Debug Toolbar.



The code will run and flash the LEDs 200 times. The debugger will not exit until all 200 flashes have completed or a switch is pressed on the RSK.

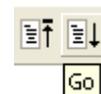
- If the LEDs are still flashing, press any user switch on the RSK to exit the 'FlashLEDs();' function.

The code will run to the breakpoint we previously set on the 'TimerADC' function.

The 'TimerADC' function initialises an interrupt on an available internal timer. On overflow of the timer module an interrupt is generated. In the 'TimerADC' code version the interrupt reads the last ADC conversion for the external potentiometer and uses the result to set the next overflow value. The ADC conversion is then re-started.

The interrupt initialisation is completed as part of the hardware setup. This is contained in the file 'interrupts.c'.

- Open the file 'interrupts.c' by double clicking on the file in the workspace view.
- Review this file and find the interrupt function that changes the LED pins, _timer_rc().
- Set a breakpoint on the line where the LED pins are modified.
- Press <Go> or <F5> to run the code from the position of the PC.



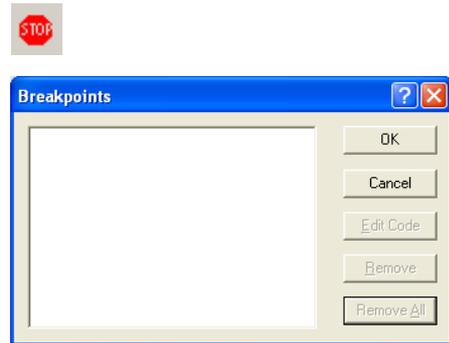
The code will stop in the interrupt routine. It is now possible to step through the interrupt function.

- Remove the breakpoint in the interrupt by double clicking again before exiting the function.
- Press <Go> to run the code from the current position of the PC.



The code will now run to the infinite loop at the end of main() function. The user LEDs should now be flashing. You can modify the flashing rate by adjusting the potentiometer on the board.

- Click <Halt> icon available on 'Debug' toolbar
- Press 'CTRL-B' to open the breakpoint window.
- Select 'Remove All'
- Press <OK>.



- Open the file 'main.c'
- Insert a breakpoint at 'Statics_Test()' function.

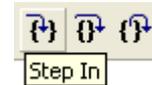
The Statics_Test() is used to demonstrate that the initialisation routine has successfully copied all the initialised variables from storage in flash to RAM.

- Press <Reset Go> on the Debug Toolbar.



The code will stop at the breakpoint. (Press a button to bypass the flashing LED test.)

- Press <Step In> on the Debug Toolbar.



It is possible to monitor variables while debugging the code. To set up a 'watch' on a variable place the mouse over the variable. If the variable is available in the current context a tool-tip will be displayed with the current value of the variable.

- Hover, the mouse over the 'ucStr' variable to see the tooltip value. Then Right click on the variable name and select 'Instant Watch'.

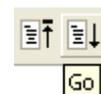
A dialog will open showing the variable and allowing further details to be explored.

- Press <Add>

The dialog will close and a new pane will open in the workspace containing the variable.

It is possible to see that the string has been successfully initialised to 'STATIC '.

- Set a breakpoint on the call to 'DisplayString();' inside the for loop.
- Press 'Go' to run the code from the current PC position.



When the program stops you can see the modified string displayed on the second line of the LCD.

Inspection of the watch pane will show that the first character of the variable string has been replaced with the first character of the constant replacement string.

- Remove the breakpoint
- Right click on the 'DisplayString();' function call after the loop and select 'Go to cursor'.

This shows that the variable was initialised at program start up and can be overwritten with 'TESTTEST'.

You have now run the 'tutorial' code and used many of the common features of the debugger. We suggest that you review the rest of the 'tutorial code'; as many functions have important information on the operation of the code, the compiler directives and comments on when they should or must be used. Please refer to Chapter 7 for more information on the project files.

Chapter 7. Project Files

7.1. Standard Project Files

The RSK tutorials are configured so that it is possible to provide the same 'tutorial' code on multiple RSK products. This allows the evaluation of the different processor cores using equivalent code. To achieve this, the following files are common between all device cores and Tool-chains.

Each of the tutorial files has detailed comment text describing the function of each code entry. Please refer to the source code for greater detail on the purpose and operation of the compiler specific details.

7.1.1. Initialisation code (resetprg.c / resetprg.h)

This is the entry point of the main 'tutorial' code.

```
/*"FUNC COMMENT"*****
* Outline      : start
* Description  : Power on reset function. This function executes following to
*               power on reset. It first calls hardware initialisation
*               function & then 'main()' function.
* Argument     : none
* Return value : none
/*"FUNC COMMENT END"*****/

void start(void)
{
    /* Set interrupt stack pointer */
    _isp_ = &_istack_top;
    /* Change protect mode register */
    prcr = 0x02U;
    /* Set processor mode register */
    pm0 = 0x00U;
    /* Change protect mode register */
    prcr = 0x00U;
    /* Set flag register */
    _flg_ = __F_value__;
#if __STACKSIZE__ != 0
    /* Set user stack pointer */
    _sp_ = &_stack_top;
#endif
    /* Setting 400H (Do not change) */
    _sb_ = 0x400U;
    /* Set variable vector's address */
    _intbh_ = 0x00U;
    _asm(" ldc #(topof vector)&OFFFh, INTBL");

    /* Initialize each sections */
    initsct();

#if __HEAPSIZE__ != 0
    /* Initialize heap */
    heap_init();
#endif
#if __STANDARD_IO__ != 0
    /* Initialize standard I/O */
    _init();
#endif
    /* Initialize FB register for debugger */
    _fb_ = 0U;

    /* Set up the hardware */
    HardwareSetup();

    /* Call main() routine */
    main();

    /* Call exit */
    exit();
}
/*"*****
End of function start
*****"*/
```

Initialisation of the variables used in the C compilers and initialisation of stack pointers are completed in the '_INIT_SCT' function for the H8 and SH compilers. The call to 'hardwaresetup()' will initialise the device hardware and peripherals ready for the tutorial software.

The call to 'main()' will start the main demonstration code.

7.1.2. Board initialisation code (hwsetup.c / hwsetup.h)

There are four common stages to the configuration of the microcontroller device. The code to demonstrate this is therefore split into four functions. Each function is written specifically for the device supported. The function calls are shown below.

```
/*"FUNC COMMENT"*****
* Outline      : HardwareSetup
* Description   : Sets up the hardware.
*              : This function calls the hardware initialization functions to
*              : configure the CPU operating frequency, port pins & required
*              : on-chip modules in order to setup the RSK for the main
*              : application.
* Argument     : none
* Return value : none
*"FUNC COMMENT END"*****/

void HardwareSetup(void)
{
    /* Configures CPU clock */
    ConfigureOperatingFrequency();

    /* Configures port pins */
    ConfigurePortPins();

    /* Enables required on-chip peripherals */
    EnablePeripheralModules();

    /* Configures interrupts */
    ConfigureInterrupts();
}
/*****
End of function HardwareSetup
*****/
```

7.1.3. Main tutorial code (main.c / main.h)

The main tutorial code is common to all tutorial projects. The display initialisation and string display functions operate on the LCD display module. Check compatibility with ks0066u controller and pin connection on the schematic before connecting a LCD module not supplied by Renesas.

```
/*"FUNC COMMENT"*****
* Outline      : main
* Description   : Main program. This function calls timer, ADC & LCD
                  initialisation functions. The user LEDs flashes until
                  the user presses a switch on the RSK.
* Argument     : none
* Return value  : none
*"FUNC COMMENT END"*****/

void main(void)
{
    /* Reset the LCD module. */
    InitialiseDisplay();

    /* Display Renesas Splash Screen. */
    DisplayString(LCD_LINE1, "Renesas");
    DisplayString(LCD_LINE2, NICKNAME);

    /* Flash the user LEDs for some time or until a push button is pressed. */
    FlashLEDs();

    /* Flash the user LEDs at a rate set by the user potentiometer (ADC) using
       interrupts. */
    TimerADC();

    /* Demonstration of initialised variables. Use this function with the
       debugger. */
    Statics_Test();

    /* This function must not exit */
    while(1);
}
/*****
End of function main
*****/
```

Chapter 8. Additional Information

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.

Further information available for this product can be found on the Renesas website at:

http://www.renesas.com/renesas_starter_kits

General information on Renesas Microcontrollers can be found at the following website.

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

Regional (English language) sites can be accessed from the Global site, or directly by going to:

Europe: <http://renesas.eu>

America: <http://america.renesas.com>

Asia: <http://sg.renesas.com>

Renesas Starter Kit for R8C/38C

Tutorial Manual

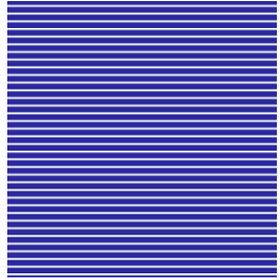
Publication Date Rev.1.00 01.APR.2010

Published by: Renesas Electronics Europe Ltd.

Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire,
SL8 5FH, UK

©2010 Renesas Electronics Europe and Renesas Solutions Corp., All Rights Reserved.

Renesas Starter Kit for R8C/38C
Tutorial Manual



Renesas Electronics Europe Ltd.

Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, UK