To our customers,

Old Company Name in Catalogs and Other Documents

   On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

RENESAS

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.

RENESAS

Renesas Starter Kit for SH7201

Tutorial Manual

RENESAS SINGLE-CHIP MICROCOMPUTER

Table of Contents

Chapter 1. Preface

Cautions

This document may be, wholly or partially, subject to change without notice.

Trademarks

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

Copyright

Website: http://www.renesas.com/

Glossary

ADC	Analog to Digital Converter		LCD	Liquid Crystal Display
RSK	Renesas Starter Kit		USB	Universal Serial Bus
HEW	High performance Embedded Workshop		PC	Program Counter
CPU	Central Processing Unit		LED	Light Emitting Diode
E10A	'E10A for Starter Kit' debugger			

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, compiler toolchains or E10A debugger – 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 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.

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 software installation for the RSK product.

To begin using the RSK, 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 or from its icon:

Open a new tutorial workspace from the [File -> New Workspace…] menu or select 'Create a new project workspace' when presented with the 'Welcome!' dialog.
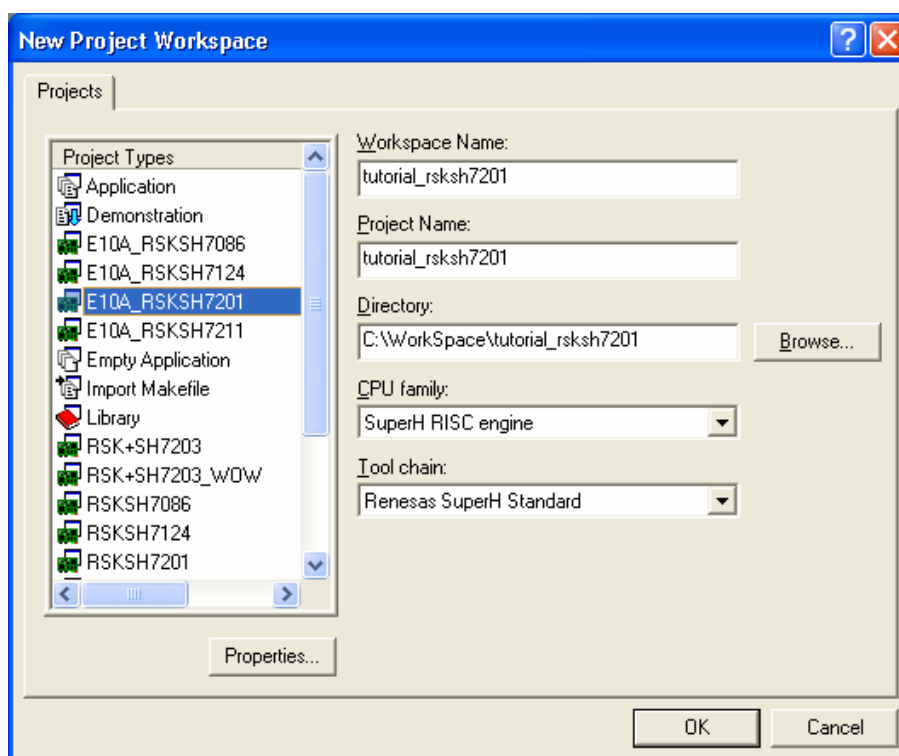
The example above shows the New Project Workspace dialog with the 'E10A_RSKSH7201' selected.

- Select the 'SuperH RISC Engine' CPU family and 'Renesas SuperH Standard' Toolchain for the RSK

- Select the 'E10A_RSKSH7201' Project type for the RSK from the project 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; 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 of the device. The final option is for an application 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 the HEW Workspace window.

- 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 Users Manual.

4.3.1. Build Configuration

The build configurations are selected from the left hand drop down list on the tool bar. 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 for the RSK.

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 off optimisation on the code being debugged.

- Select the 'Debug' Build Configuration.
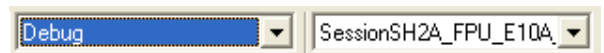
4.3.2. Debug Session

The debug sessions are selected from the right hand drop down list on the tool bar. The options vary between RSK however one will always start Debug and include the type of debug interface. The alternate selection will be 'DefaultSession'. This purpose of the debug session is to allow the use of different debugger targets or different debugger settings on the same project.

- Select the 'SessionSH2A_FPU_E10A_USB_SYSTEM' debug session.

# Chapter 5. Building the Tutorial Project

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

The dialog that is displayed will be specific to the selected toolchain.

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.

When complete close the dialog box by clicking <OK>.
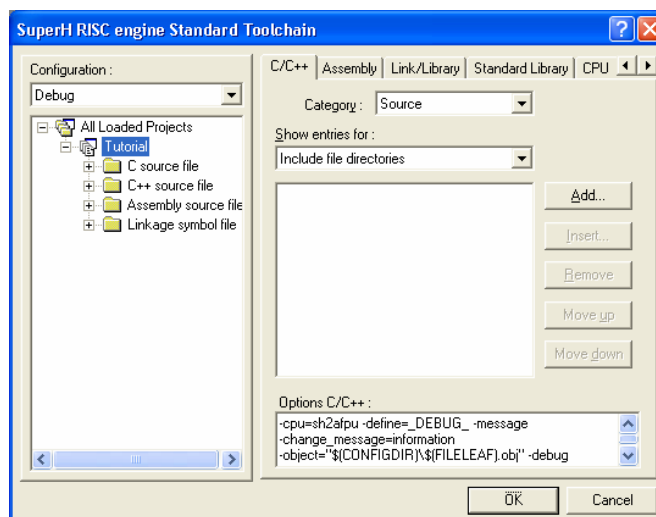
5.1. Building Code

There are three short cuts available for building the project.

- Select the 'Build All' tool bar button.

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

- Select the 'Build' tool bar 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.

- 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 necessary for you to power the RSK from the supplied PSU. The E10A module will be powered via the USB cable. 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 you can provide an external power source, 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 E10A module to the host computer. The following assumes that the steps in the Quick Start Guide have been followed and the E10A module drivers have been installed.

- Connect the E10A to the host machine's USB port

- Connect the E10A module to the target hardware ensuring that it is plugged into the connector marked 'E10A'.

- Turn on the power to the RSK

- Click the connect button from the Tool Bar.

- On 'Select 'Emulator mode' dialog

  For Device select 'SH7201'

  For Mode select 'E10A_USB Emulator'

- Click <OK>
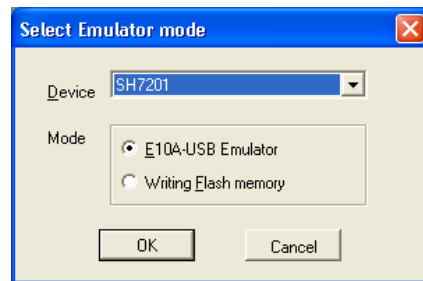
- Press the RES button on the RSK board.
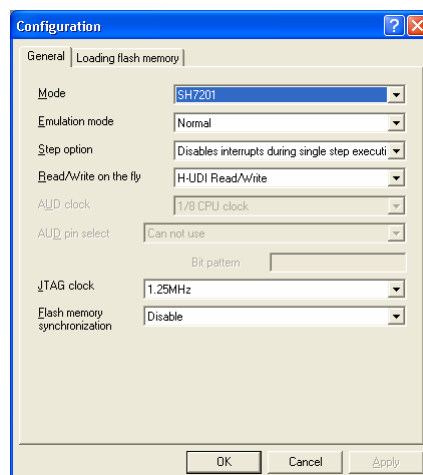
- Click <OK>

- The system should then connect.
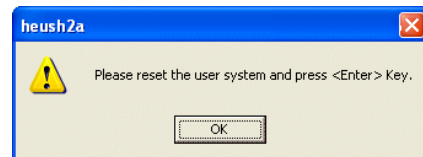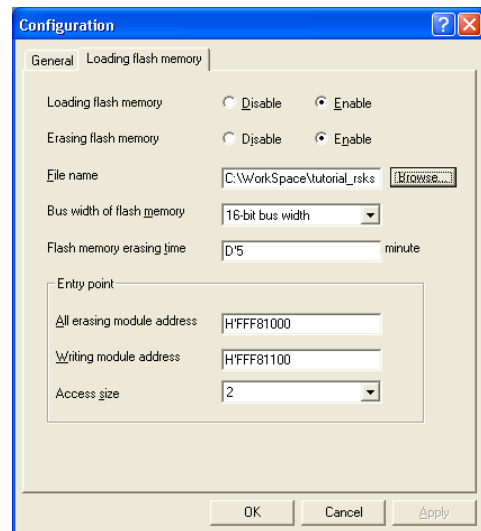
- Click Emulator System from Tool bar.

- Switch to the 'Loading flash memory' Tab

Note: Please ensure all settings for both tabs of the 'Configuration Dialog' match the diagrams shown.

- Select 'Enable' for 'Loading and Erasing flash memory'

- Browse and select the 'fmtool_e10a.mot' file in your workspace.

- Click <OK>

Chapter 6.Downloading and Running the Tutorial

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

Now that you are 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'.
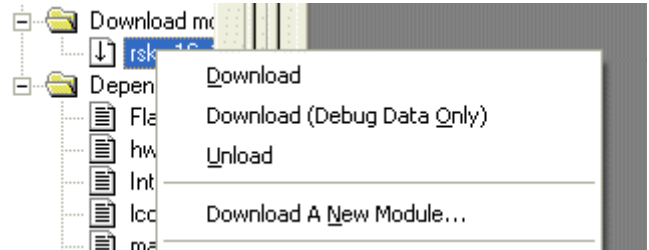
On completion the debugger and code are ready to be executed.

> Note: In order to Retain HMON Bootloader on the old boards, Flash Block 0 has not been used (Neither erased nor programmed) in this application. So it is advisable NOT to reset the target (click on "Go" button on the debug toolbar instead) after downloading the application. There is a special script exists which sets the following registers to the addresses given below, resetting the CPU will change this value resulting in unexpected operation.
>
> PC = 0x00010800
>
> SR = 0x08FF7FF0

- Press 'Go' on the Debug Tool Bar.

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.

```
000000A4        void POWER_ON_RESET(void)
                {
                    size_t   stRamSize;

                    /* Initialise the port pins that control the LEDS. These can be used
                        to ouput low level error messages and give an early sign of life
000000A6     ⇨      rstInitialiseLEDs();
                    /* Set the Frequency control register */
000000AA            rstInitialiseCPG();
                    /* Initialise all the ports */
000000B6            rstInitialisePORTs();
                    /* Configure and check SDRAM */
000000C0            stRamSize = rstConfigureSDRAM();
                    /* If the SDRAM is not working or of insufficient size */
000000D6            if (stRamSize < RSK_SDRAM_SIZE)
                    {
                        /* Flash the LEDs to show the error */
000000DE                rstFatalError();
                    }
```

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

- Scroll down the file, in the function CStartUp you will see the call to main.

- Place an eventpoint at the call to main(); by double clicking in the **Event Column** next to the line to stop at.

- Press 'Go' on the Debug Tool Bar.

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

- Press 'Step In' on the Debug Tool Bar (Please note that you may have to press the button several times).
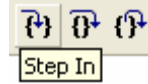
The code window will open 'main.c' and show the new position of the program counter.

```
00001050        void main(void)
                {
                    /*  Reset the LCD module. */
00001052            InitialiseDisplay();

                    /*  Display Renesas Splash Screen. */
00001056            DisplayString(LCD_LINE1,"Renesas");
00001066            DisplayString(LCD_LINE2,NICKNAME);

                    /*  Flash the user LEDs for some time or until a key is pressed. */
00001074            FlashLEDs();

                    /*  Flash the user LEDs at a rate set by the user potentiomenter (ADC)
                        interrupts. */
0000107E            TimerADC();

                    /*  Demonstration of initialised variables.
                        Use this function with the debugger. */
00001088            Statics_Test();
```
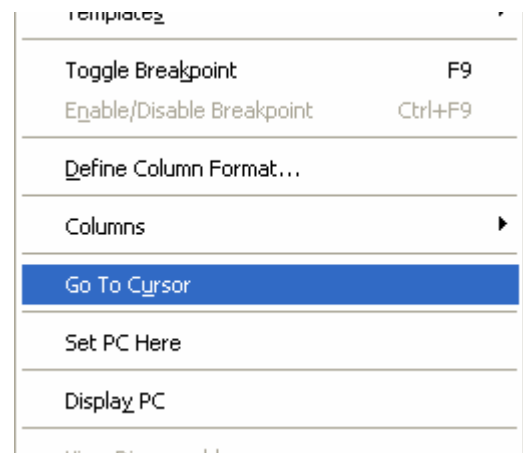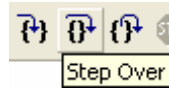
Support for the LCD display is included in the tutorial code. We do not need to be concerned about the details of the LCD interface – except that the interface is write-only and so is not affected if the LCD display is attached or not.

- Insert an eventpoint on the 'TimerADC();' function call.

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

Templates	►
Toggle Breakpoint	F9
Enable/Disable Breakpoint	Ctrl+F9
Define Column Format...	
Columns	►
Go To Cursor	
Set PC Here	
Display PC	
View Disassembly	

The code will run to the selected line and stop.

- Press 'Step Over' on the Debug Tool Bar.

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

- If the LEDs are still flashing press the SW1 button on the RSK to exit the FlashLEDs() function.

The code will run to the eventpoint we previously set on the Timer function.

There are several versions of the timer function depending upon the peripherals available in the device. The default function is TimerADC which we shall demonstrate here.

The timer function initialises an interrupt on an available internal timer. On a compare match in 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 compare match value. The ADC conversion is then re-started.

The interrupt initialisation is performed as part of the hardware setup. This is located 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, INT_TMR0_CMIA0 (void)

- Set an eventpoint on the line where the LED pins are modified.

- Press 'Go' or 'F5' to run the code from the current PC position.

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

- Remove the eventpoint in the interrupt by double clicking again before exiting the function.

- Press 'Step Over' to step over the instruction and observe the LEDs turn off.

- Press 'Go' to run the code from the current PC position.

The code will now run to the infinite loop at the end of Main(). The user LEDs should now be flashing. If the RSK supports an ADC you can modify the flashing rate by adjusting the potentiometer on the board.

- Press 'Stop' on the debug tool bar.

- Open the file 'main.c'
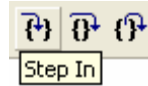
- Insert an eventpoint on 'StaticsTest();'.

The statics test is used to demonstrate that the initialisation has successfully copied all initialised variables from storage in flash to RAM.

- Press 'Go' on the Debug Tool Bar.

The code will stop at the eventpoint.

- Press 'Step In' on the Debug Tool Bar.

It is possible to monitor variables during debugging of 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 eventpoint on the 'DisplayString();' function call inside the 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 eventpoint

- 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'.

The modified string is also displayed on the LCD

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 / Toolchains.

Each of the tutorial files has expanded 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)

This is the entry point of the main tutorial code. Depending upon the compiler used this file may be the actual entry point of the software or may be called during the initial setup of the environment.

```
              #pragma section ResetPRG

              /* The stack pointer is iniailised from the value in the reset vector table
                 (gcResetVectorTable in vecttbl.c). This is set to point at the end of the
                 upper section of internal RAM (H'FFF84000 to H'FFF87FFF). The stack is
                 pre-decremented so it can be loaded with H'FFF88000 */

00010800    ⇨ void POWER_ON_RESET(void)
              {
                  size_t  stRamSize;

                  /* Initialise the port pins that control the LEDS. These can be used
                      to ouput low level error messages and give an early sign of life */
00010804          rstInitialiseLEDs();
                  /* Set the Frequency control register */
00010808          rstInitialiseCPG();
                  /* Initialise all the ports */
00010814          rstInitialisePORTs();
                  /* Configure and check SDRAM */
0001081E          stRamSize = rstConfigureSDRAM();
                  /* If the SDRAM is not working or of insufficient size */
00010834          if (stRamSize < RSK_SDRAM_SIZE)
                  {
```

Initialisation of the variables used in the C compilers and initialisation of stack pointers are completed in the _INITSCT 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)

This file contains the initial hardware setup required to put this RSK in known state. The code configures the required interrupts. The code also configures the port pins as required by the application. The function calls are shown below.

```c
void HardwareSetup(void)
{
    /* Configure the interrupts */
    ConfigureInterrupts();

 /* LCD initialization */

    /* Register Select pin set as o/p */
    P_PORT.C.IOR.BIT.B11 = 1;
    /* Display Enable pin set as o/p */
    P_PORT.C.IOR.BIT.B12 = 1;

    /* Port-D IO Register */
    /* LCD_D4 - D7 & LED Port Output */
    P_PORT.D.IOR.BIT.B0 = 1;
    P_PORT.D.IOR.BIT.B1 = 1;
    P_PORT.D.IOR.BIT.B2 = 1;
    P_PORT.D.IOR.BIT.B3 = 1;
```

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 a ks0066u controller and pin connection on the schematic before connecting an LCD module not supplied by Renesas.

```c
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 key is pressed. */
    FlashLEDs();

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

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

    /*  End of user program. 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/resesas_starter_kits

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

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

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

Europe: http://eu.renesas.com

Americas: http://america.renesas.com

Asia: http://sg.renesas.com

Renesas Starter Kit for SH7201
Tutorial Manual