REJ10J2126-0100

Renesas Starter Kit for R8C/35C

Tutorial Manual

RENESAS MCU
R8C Family R8C/3x Series

Rev.1.00
Revision date: 01.Apr.2010

Renesas Electronics Europe Ltd.
www.renesas.com

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

Renesas Europe Website: http://eu.renesas.com/

Renesas Global Website: http://renesas.com/

Glossary

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

Chapter 2. Introduction

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

- How do I compile, link, download, and run a simple program on the Renesas Starter Kit?
- How do I build an embedded application?
- How do I use Renesas development 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 Renesas Starter Kit Quick Start Guide have been completed. Please refer to the Quick Start Guide for details of installation and configuration of the Renesas Starter Kit.

NOTE: These tutorials are designed to show you how to use the Renesas Starter Kit and are not intended as a comprehensive introduction to the High-performance Embedded Workshop 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 then downloaded in a ‘Debug’ situation.

The build configuration menu in High-performance Embedded Workshop 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

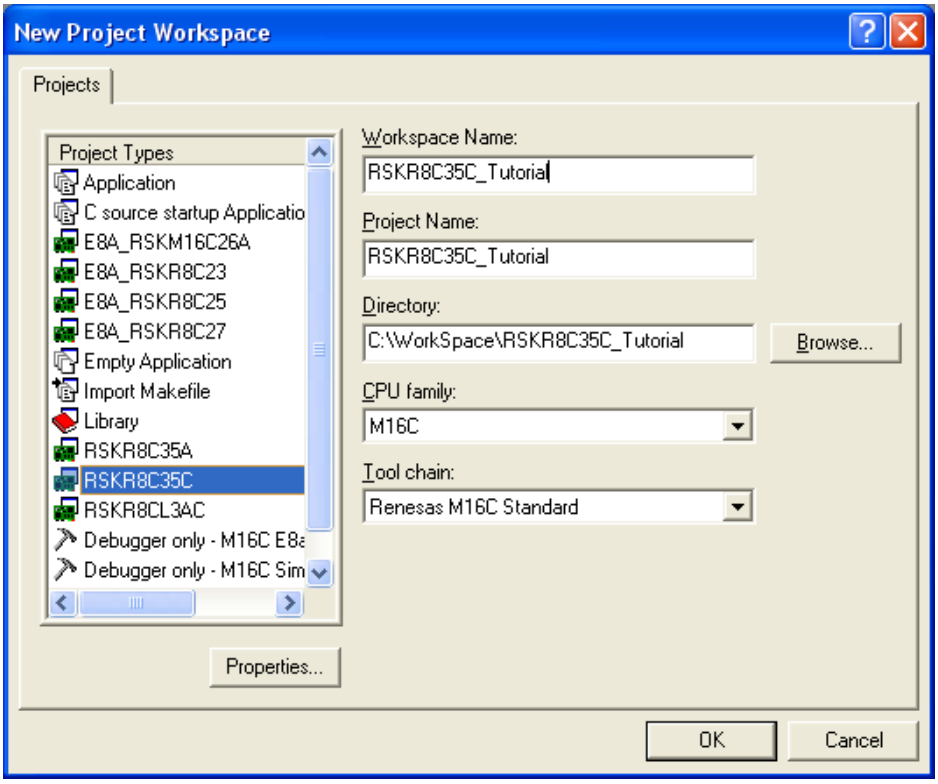
4.1. Introduction

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. 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 RSKR8C35C selected.

- Select the M16C CPU family and "Renesas M16C Standard" Toolchain (R8C is supported by the same compiler as M16C).
- Select the "RSKR8C35C" 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; this name may be changed.

  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 presents the three types of example project available:

1. Tutorial: This is the one of interest at this time; the code is explained later in this manual.
2. Sample code: this provides examples for using various peripherals. If you select this and click <Next> it will open a new dialog allowing the selection of many code examples for the peripheral modules of the device.
3. 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.

- 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 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 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.

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 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 Renesas Starter Kit types however one will always start Debug and include the type of debug interface. The alternate selection will be 'SessionR8C_E8a_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 'SessionR8C_E8a_SYSTEM' debug session.

Chapter 5. Building the Tutorial Project

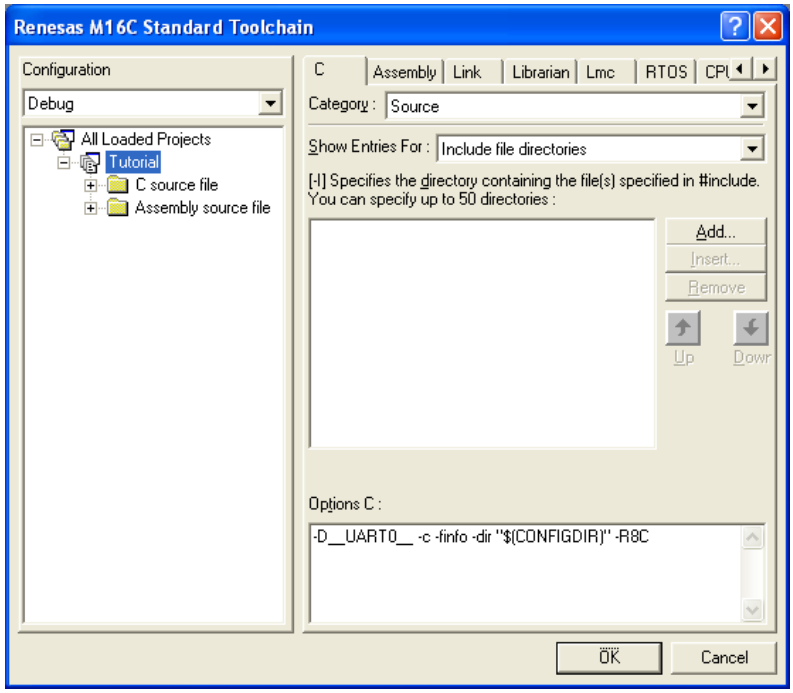
The tutorial project build settings have been pre-configured in the tool-chain options. To view the tool chain options select the 'Build' menu item and the relevant tool-chain. This should be the first option(s) on the drop down menu.

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' drop down lists to be aware of the options available.
  Note: 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.

1. Selecting 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.

2. Selecting 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.

3. 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.

The Quick Start Guide provided with the Renesas Starter Kit 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.

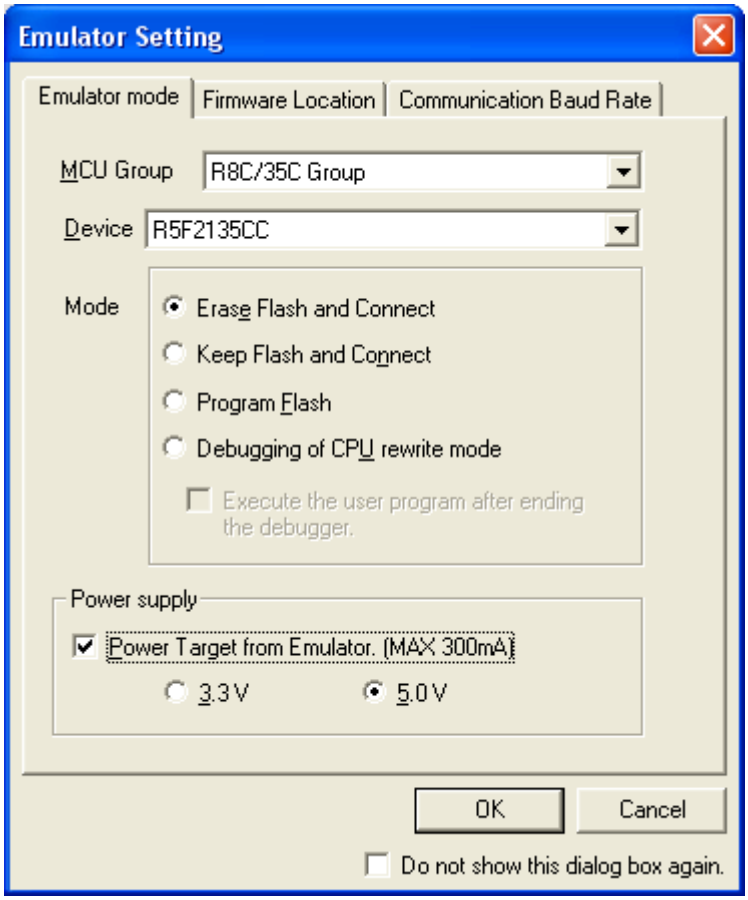
- Fit the LCD module to LCD connector on the board, so it lies above J4. Ensure all the pins of the connector are correctly inserted in the socket.
- Connect the E8a debugger to the USB port on your computer.
- Connect the E8a Debugger to the target hardware ensuring that it is plugged into the connector marked 'E8A'.
- If supplying external power to the board this can be turned on now.

5.3. Connecting to the target with the E8a

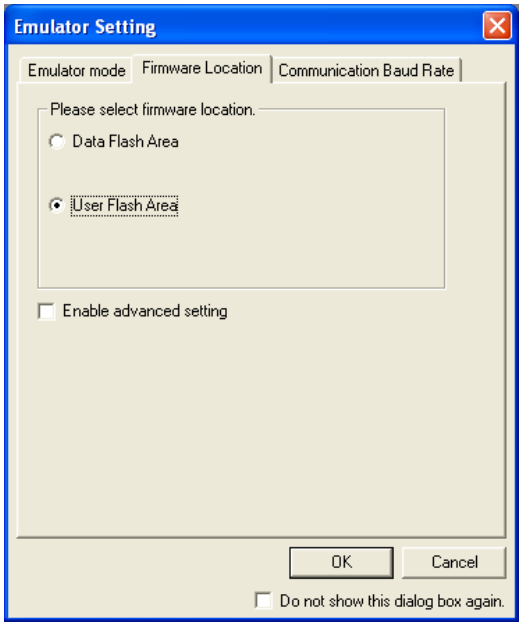
This section will take you through the process of connecting to the device, programming the Flash and executing the code.

- Select the 'SessionR8C_E8a_SYSTEM' debug session.
- Click the <Connect> button on the debug toolbar. 
- Select the correct MCU Group type (R8C/35C Group illustrated).
- Select the correct device type (R5F2135CC 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.

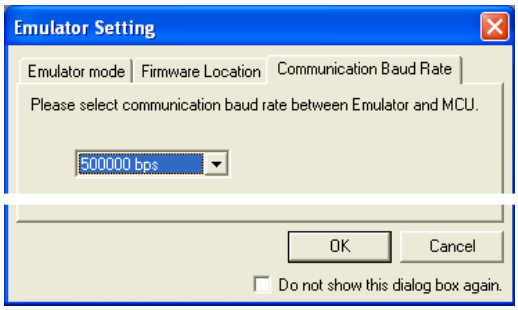
Otherwise connect a 5V centre positive supply.



- Choose “User Flash Area” in “Firmware Location” tab.

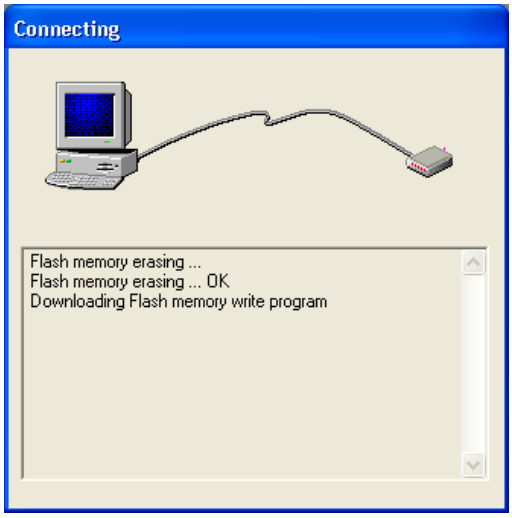


- Choose “500000bps” in “ Communication Baud Rate” tab.
- Click <OK>.



- The Flash Memory write program will be downloaded to the target.
- The Output window in High-performance Embedded Workshop will show ‘Connected’.

  Note: The connection to the target will activate the debugger buttons on the High-performance Embedded Workshop toolbar.



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’.

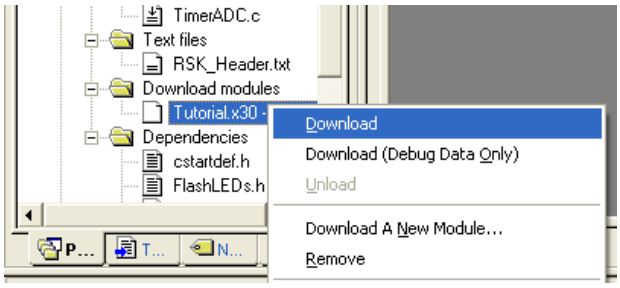
Chapter 6. Downloading and Running the Tutorial

Once the code has been built in High-performance Embedded Workshop it needs to be downloaded to the Renesas Starter Kit.

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

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

  Note: The output display will show "Flash memory write end" to confirm the completion of the download.

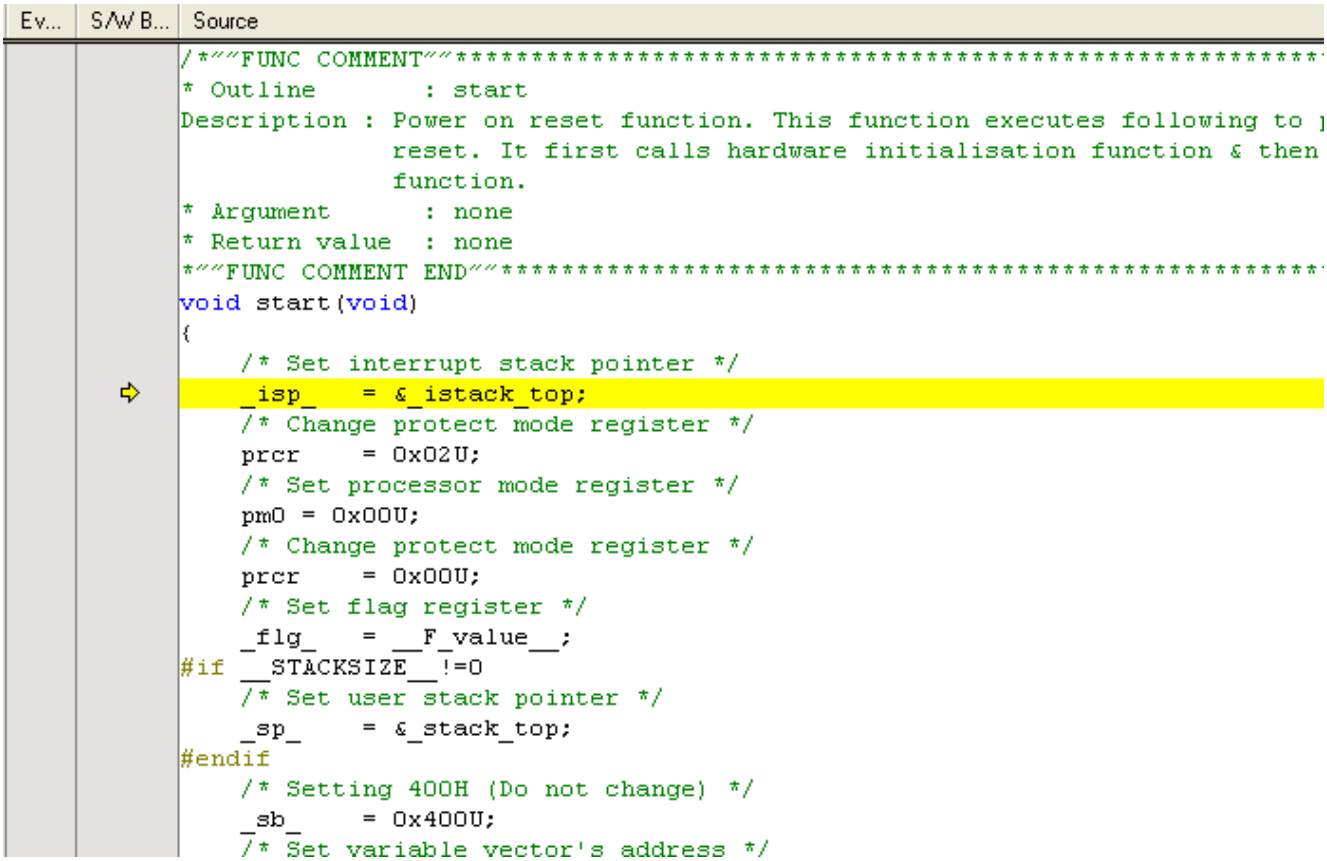


On completion the debugger and code are ready to be executed.

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.



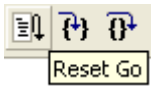
We will now skip over the initialisation code and proceed to the main tutorial.

- Open the file called 'resetprg.c'.

- Place a breakpoint at the call to `main();`

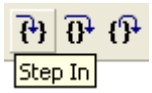
  Breakpoints can only be set when the E8a is connected to the target and the module is downloaded. Breakpoints can be set by double clicking in the grey column containing the PC arrow next to the line to break at; or selecting the line and pressing F9; or right click on the line and select 'Toggle breakpoint'. Alternatively set an eventpoint, by clicking in the column to the left of the breakpoint column. Up to 8 eventpoints can be set. Eventpoints do not require programming the flash memory and so are faster to use.

- Press 'Reset Go' on the debug toolbar.

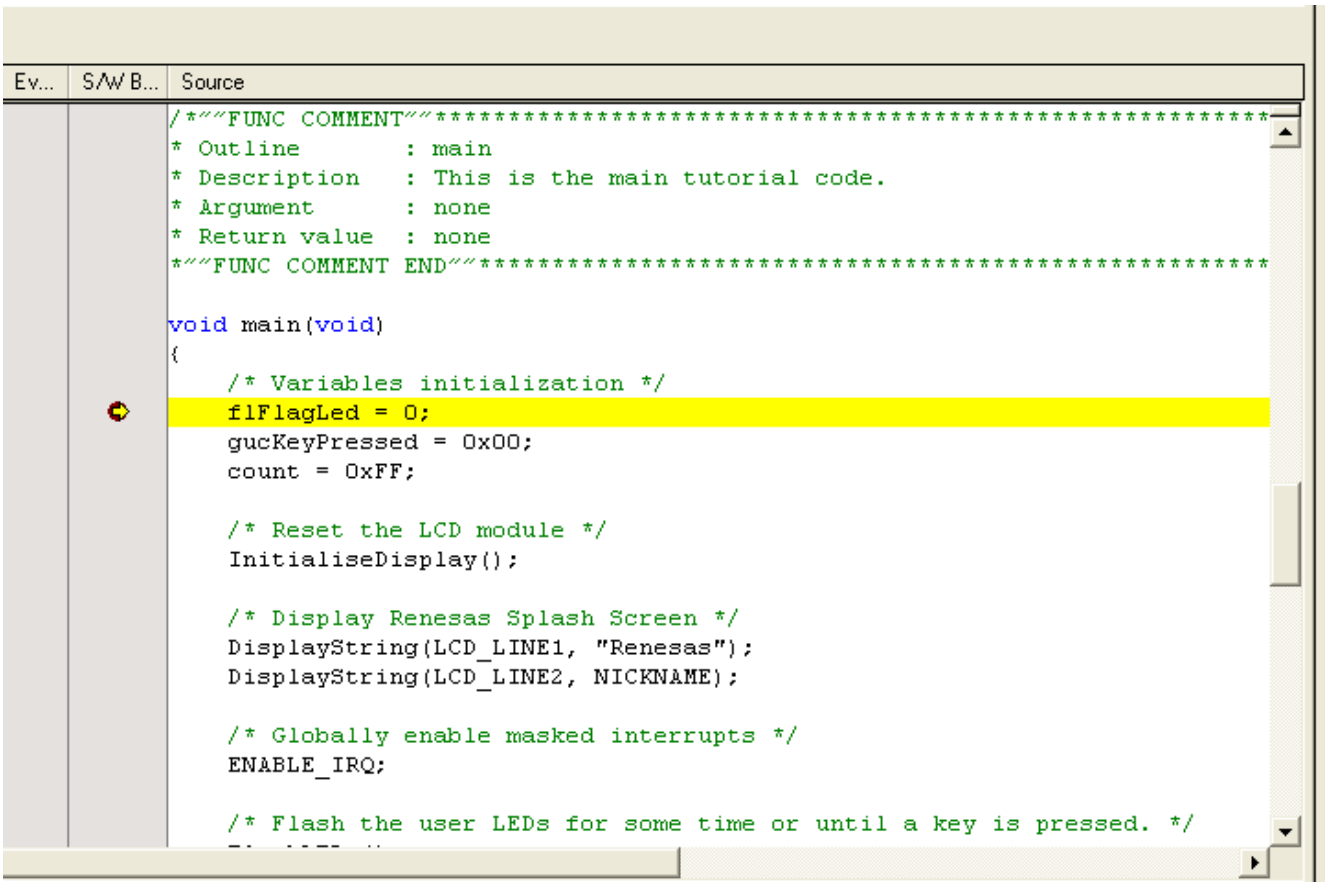


The code will execute to the breakpoint. At this point all the device initialisation will have been completed.

- Press 'Step In' on the debug toolbar.

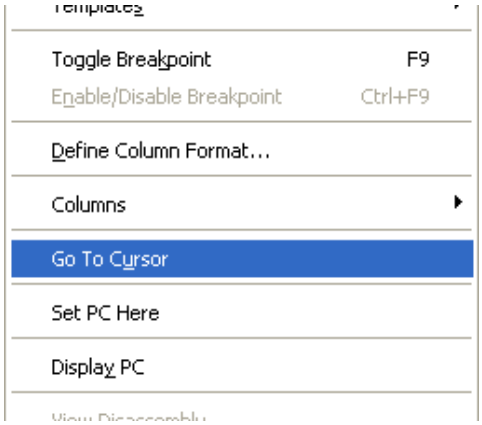


The code window will open 'main.c' and show the new position of the program counter.



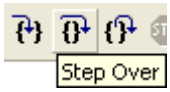
- Insert a breakpoint on the call to the `TimerADC();` function.

- Right click on the `FlashLEDs();` function and select 'Go To Cursor' as illustrated.



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 button is pressed on the board.

- If the LEDs are still flashing press the SW1 button on the board 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 a compare match in the timer module an interrupt is generated. In the `TimerADC` code, the interrupt reads the last AD conversion for the external potentiometer and uses the result to set the next compare match value. The AD 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 window.
- Review this file and find the interrupt function that changes the LED pins; `_timer_rc(void)`.
- Set a breakpoint on the line where the LED pins are modified.
- Press <Go> or <F5> to run the code from the current PC position.

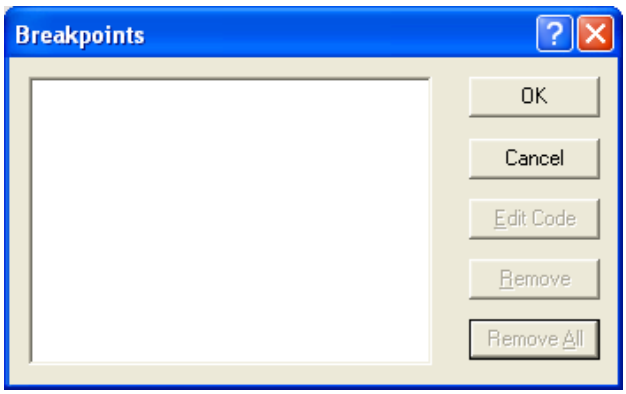
The code will stop in the interrupt routine. It is now possible to step through the interrupt function.

- Remove the breakpoint in the interrupt by double clicking again before exiting the function.
- 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();` function. The user LEDs will now be flashing. You can modify the flashing rate by adjusting the potentiometer on the board.

- Press <Stop> on the debug toolbar.

- Press ‘CTRL-B’ to open the breakpoint window.

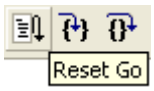


This dialog is not available if user debugged using eventpoints as suggested at top of page 10. ‘CTRL-E’ will open the eventpoint window. Eventpoins can be viewed in Breakcondition tab of event window.

- Press <Remove All>.
- Press <OK>.
- Open the file ‘main.c’.
- Insert a breakpoint on `Statics_Test();`.

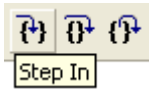
The `Statics_Test()` function is used to demonstrate that the initialisation has successfully copied all initialised variables from storage in flash to RAM.

- Press <Reset Go> on the debug toolbar.



The code will stop at the breakpoint. (Press one of the board buttons to bypass the flashing LED test.)

- Press <Step In> on the debug toolbar.



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 tooltip 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 Renesas Starter Kit tutorials are configured so that it is possible to provide the same tutorial code on multiple Renesas Starter Kit 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 Toolchains.

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.

```
44  void start(void)
45  {
46      /* Set interrupt stack pointer */
47      _isp_   = &_istack_top;
48      /* Change protect mode register */
49      prcr    = 0x02U;
50      /* Set processor mode register */
51      pm0 = 0x00U;
52      /* Change protect mode register */
53      prcr    = 0x00U;
54      /* Set flag register */
55      _flg_   = __F_value__;
56  #if __STACKSIZE__!=0
57      /* Set user stack pointer */
58      _sp_    = &_stack_top;
59  #endif
60      /* Setting 400H (Do not change) */
61      _sb_    = 0x400U;
62      /* Set variable vector's address */
63      _intbh_ = 0x00U;
64      _asm("  ldc #(topof vector)&0FFFFh,INTBL");
65
66      /* Initlalize each sections */
67      initsct();
68
69  #if __HEAPSIZE__ != 0
70
71      /* Initialize heap */
72      heap_init();
73  #endif
74  #if __STANDARD_IO__ != 0
75      /* Initialize standard I/O */
76      _init();
77  #endif
78      /* Initialize FB register for debugger */
79      _fb_ = 0U;
80
81      /* Set up the hardware  */
82      HardwareSetup();
83
84      /* Call main() routine */
85      main();
86
87      /* Call exit */
88      exit();
89  }
```

Initialisation of the variables used in the C compilers and initialisation of stack pointers are completed in the `initsct()` function for the compiler.

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.

```
41  /***********************************************************************************
42  Function Name : HardwareSetup
43  Description   : Sets up the hardware.
44          This function makes a call to initialisation functions to configure the CPU
45          operating frequency, port pins & relevant on-chip modules (i.e. such as ADC,
46          timer etc.) in order to setup the RSK for the main application.
47  Parameters    : none
48  Return value  : none
49  ***********************************************************************************/
50  void HardwareSetup(void)
51  {
52      ConfigureOperatingFrequency();
53      ConfigurePortPins();
54      EnablePeripheralModules();
55      ConfigureInterrupts();
56  }
57
58  /***********************************************************************************
59  End of function HardwareSetup
60  ***********************************************************************************/
```

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 an LCD module not supplied by Renesas.

```
/***********************************************************************************
Function Name:  main
Description:    Main function
Parameters:     None
Return value:   None
***********************************************************************************/
void main(void)
{
    /* Variables initialization */
    flFlagLed = 0;
    gucKeyPressed = 0x00;
    count = 0xFF;

    /* Reset the LCD module */
    InitialiseDisplay();

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

    /* Globally enable masked interrupts */
    ENABLE_IRQ;

    /* 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 funtion with the debugger. */
    Statics_Test();

    /* End of the user program. This function must not exit. */
    while (1)
    {
        /* Wait forever */
    }
}
/***********************************************************************************
End of function main
***********************************************************************************/
```

Chapter 8. Additional Information

For details on how to use High-performance Embedded Workshop, refer to the High-performance Embedded Workshop 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 websites.

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

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

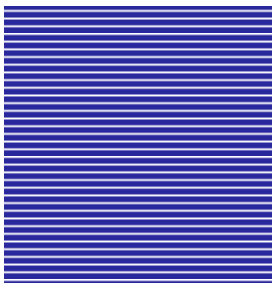
Europe: http://eu.renesas.com

Americas: http://america.renesas.com

Asia: http://sg.renesas.com

Renesas Starter Kit for R8C/35C
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, United Kingdom

Renesas Starter Kit for R8C/35C
Tutorial Manual



Renesas Electronics Europe Ltd.
Dukes Meadow, Millboard Road, Bourne End Buckinghamshire SL8 5FH, United Kingdom