REG10J0023-0100

Renesas Starter Kit

RSK H8S2215R Tutorial Manual

RENESAS SINGLE-CHIP MICROCOMPUTER

Rev.1.00
Revision date 12.01.2007

Renesas Technology 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

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

Glossary

ADC	Anaglog to Digital Conversion	NMI	Non-Maskable Interrupt
HMON	Embedded Monitor	LED	Light Emitting Diode
RSK	Renesas Starter Kit	LCD	Liquid Crystal Display
RSO	Renesas Solutions Organisation.	USB	Universal Serial Bus
HEW	High performance Embedded Workshop	FDT	Flash Development Toolkit
CPU	Central Processing Unit	RTC	Real Time Clock
IRQ	Interrupt ReQuest		

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 or the compiler tool-chains – 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.

The HMON monitor code is provided in a pre-compiled library for inclusion in the user code. This library must be included in the tool chain linker settings. There are some configuration options provided to the user. These are provided in the following files - hmonserialconfiguser.c, hmonconfiguser.c, hmonconfiguser.h and hmonserialconfiguser.h. For more detailed information on this please refer to HMon User Manual.

Chapter 4. Project Workspace

4.1. Introduction

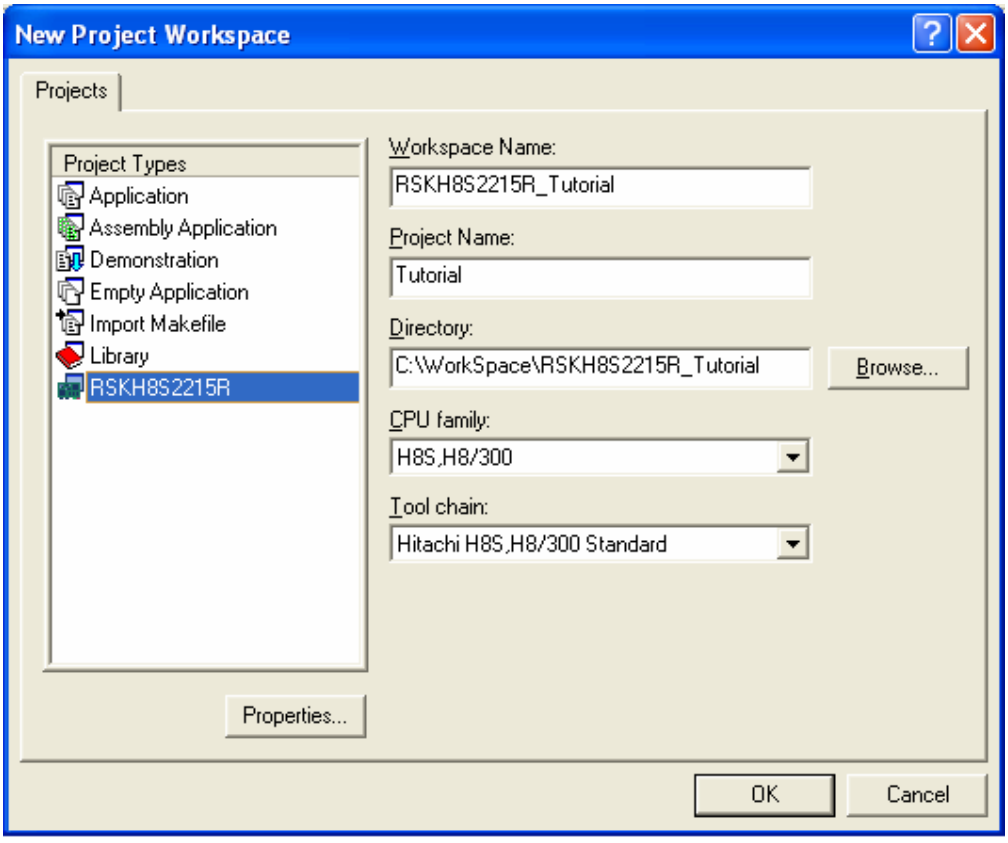
HEW is an integrated development tool that allows the user to write, compile, program and debug a software project on any of the Renesas Microcontrollers. HEW will have been installed during the software installation for the RSK product.

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 (i.e. Start Menu > Programs > Renesas > High-performance Embedded Workshop > High-performance Embedded Workshop) 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 RSKH8S2215R selected.

- Select the 'H8S,H8/300' CPU family and 'Hitachi H8S,H8/300 Standard' Tool chain for the RSK
- Select the 'RSKH8S2215R' 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

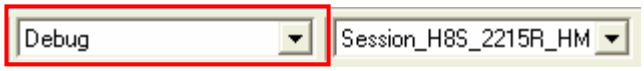
The workspace that has been created contains two Build Configurations and two Debug Sessions. The Build Configuration allows the same project to be built but with different compiler options. The options available to the user are described fully in the HEW Users Manual.

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.

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 the optimisation when the code being debugged.

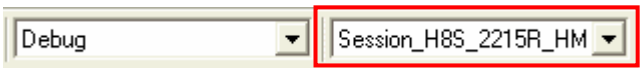
- Select the Debug Build Configuration.



4.3.2. Debug Session

The debug sessions are selected from the right hand drop down list on the tool bar. The options may vary between RSKs 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 'Session_H8S_2215R_HMon' debug session.



Chapter 5. Building the Tutorial Project

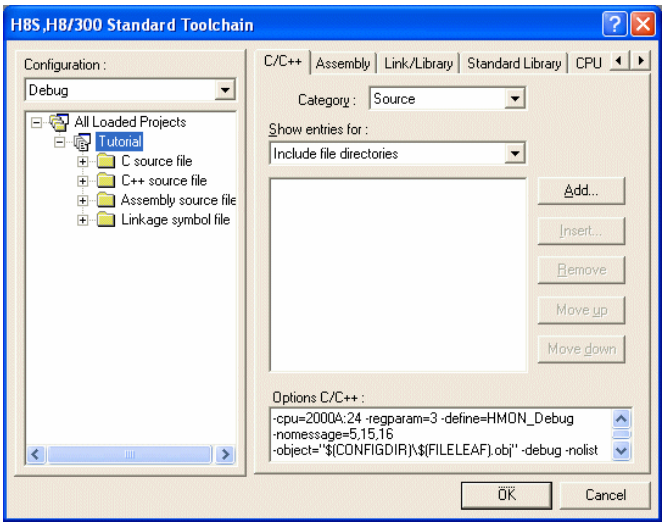
The tutorial project build settings have been pre-configured in the tool-chain options. To view the tool chain options select the 'Build' Menu item and the relevant tool-chain. This should be the first option(s) on the drop down menu.

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

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 <Cancel>. Please note that, if you have modified any of the options and want to save it, you need to click <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 toolchain option settings are modified.

- 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 provided by the E8 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 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 E8 to the host computer. The following assumes that the steps in the Quick Start Guide have been followed and the E8 drivers have been installed.

- Connect the E8 debugger to the USB port on your computer.
- Connect the E8 Debugger to the target hardware ensuring that it is plugged into the connector marked 'E8' which is nearest the power connector.
- If supplying external power to the board this can be turned on now.

5.3. Connecting to the target with E8Direct & HMon

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

The E8 provides an interface called 'E8DIRECT' to allow HMon embedded debugger to connect to the target device.

To provide flash programming capabilities the FDT Flash configuration wizard must be configured. This process is only required once in a project. The settings provide information to HMon to allow re-programming of the device.

- Select the FDT Wizard from the FDT Tool Bar 

If the flash kernel is already configured then a confirmation window will open with the kernel settings. To modify any of the settings just double click on the item. If FDT is already configured then please proceed to section 5.3.1 'Connecting to HMon'.

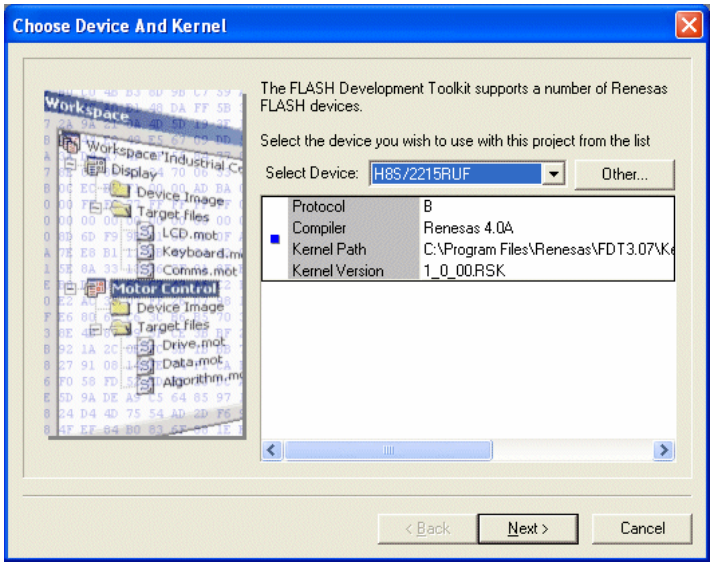
The first stage is to select the specific device from the installed kernels. The installer will have added the kernel to the FDT registry so it will appear under the device list drop down menu. If for any reason the kernel has to be re-build such as the crystal frequency on the board has changed, then the installation path is:

<FDT Installation>\Kernels\ProtB\2215RU\Renesas\1_0_00.RSK

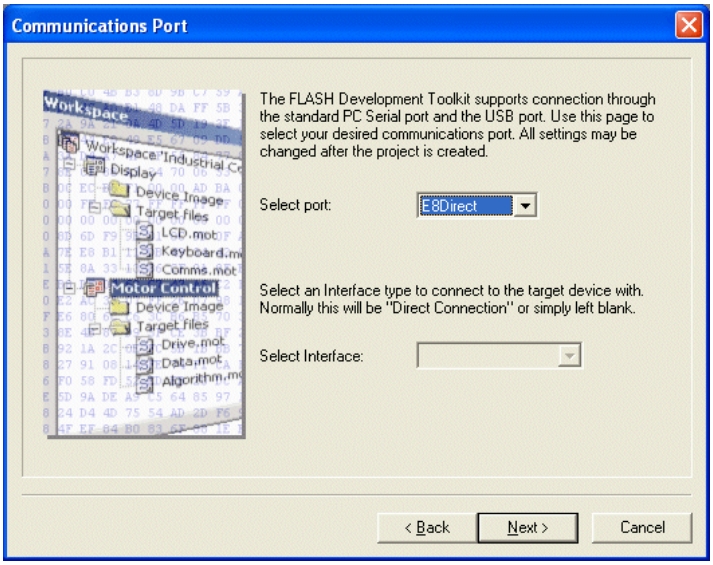
- Select the Device on the RSK from the drop down list.
- In the sub pane, select the kernel version that ends in '.RSK'.
- Press <Next>.

If you have copied the kernel to another location to modify for a different crystal frequency, the device will not be listed in the sub pane. In this case:

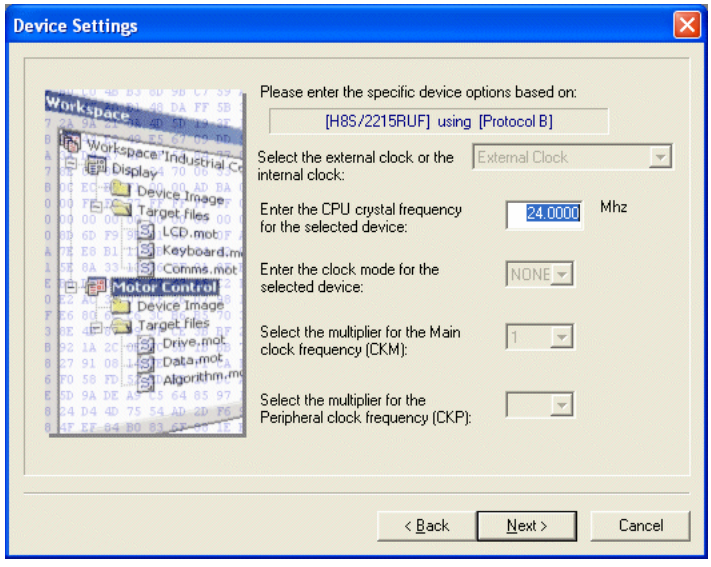
- Press <Other…> and navigate to your modified kernel.



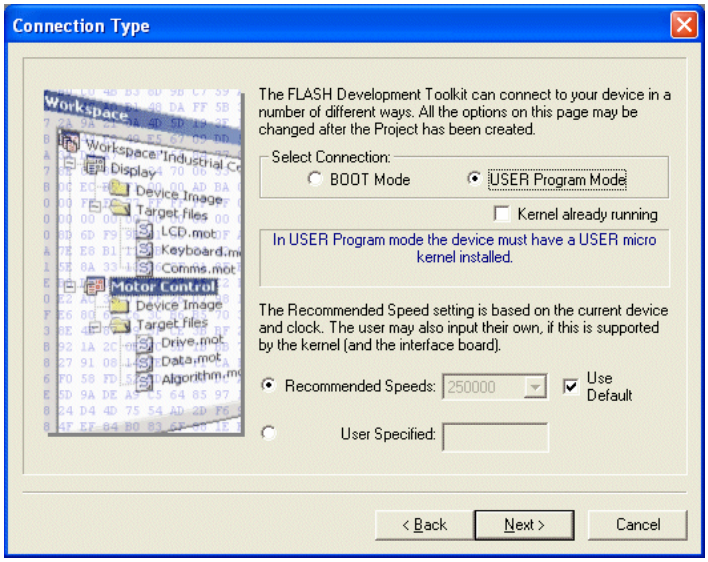
- Select E8DIRECT as the communication Port
- Press <Next>.



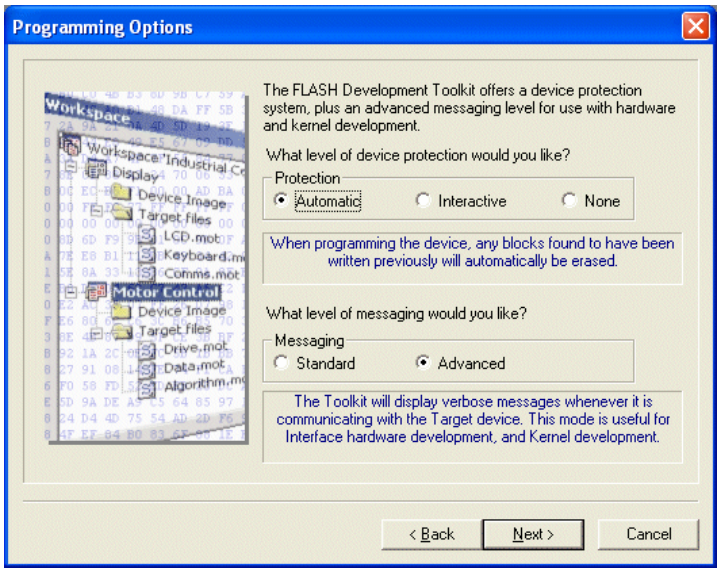
- The default settings are suitable for an un-modified RSK board.
- Confirm the Crystal Frequency matches the board.
- Confirm the main clock frequency multiplier.
- Confirm the peripheral clock multiplier.
- Press <Next>.



- Ensure that ‘USER Program Mode’ is selected.
- Confirm that ‘Use Default’ is selected.
- Press <Next>.



- Confirm the default selections of ‘Automatic’ and ‘Advanced’.
- Press <Next>.



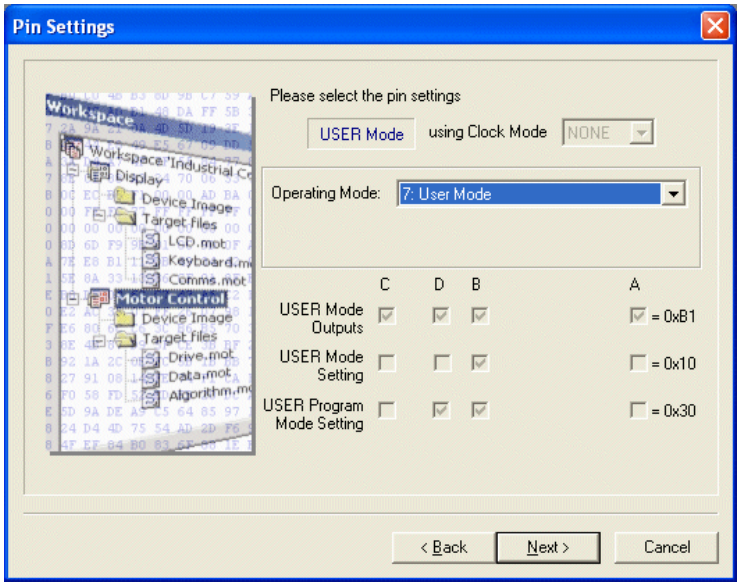
- The following warning dialog will be displayed.
- Press <OK>.

To communicate with the RSK FDT and HMon need to be able to change the operating mode of the microcontroller. To do this there are settings to control the state of the Mode pins via the E8Direct interface. These settings are confirmed in the following screens.

Unless you are very sure that the mode pin settings need to change, do not modify the default settings.

Damage to the microcontroller can be sustained with incorrect settings.

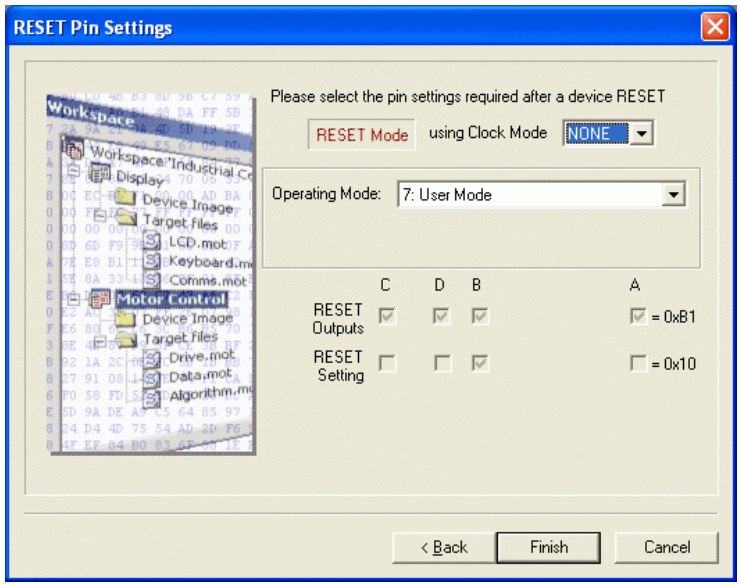
- Confirm the mode pin settings.
- Press <Next>.



- The following warning dialog will be displayed.
- Press <OK>.

Damage to the microcontroller can be sustained with incorrect settings.

- Confirm the mode pin settings.
- Press <Finish>.



The Flash configuration has now been completed.

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

- Select [File' -> 'Save Workspace'].

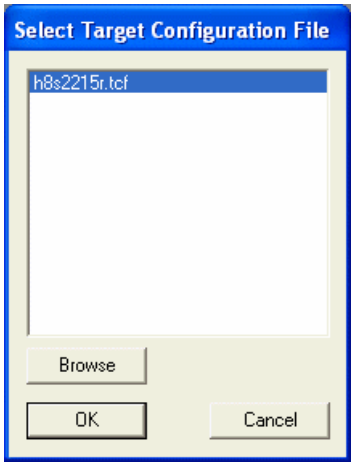
5.3.1. Connecting To HMon

We can now attempt to connect to the target device.

- Press the Green ‘Connect’ Icon.

HMon is able to discover the internal flash configuration of the device from the FDT kernel we configured earlier. HMon also needs information on the location of the IO registers and internal / external RAM. This information is stored in a ‘.TCF’ file. The TCF file is supplied and registered with HEW. As it is possible to have TCF files with slightly different configurations the following dialog is displayed to allow selection.

- Select the TCF file that applies to your RSK.
- Press <OK>.



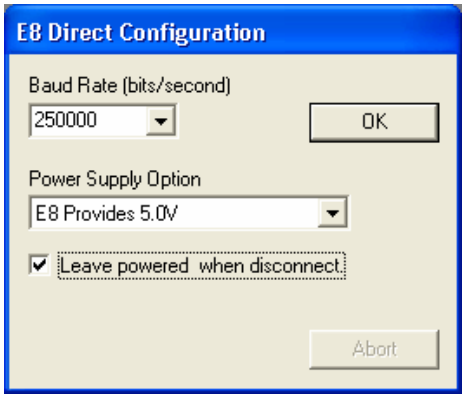
A further dialog will be displayed to confirm if it acceptable to assume only one E8 device will be connected to the host computer while using this project. This is the preferred operating mode. While it is possible to use more than one E8 device this is not recommended.

• Press ‘Yes’ to assume only one E8 device is connected while in this project.	

Note: If you have not followed the Quick Start Guide, additional driver messages may be displayed here. Please refer to the Quick Start Guide for driver installation.

The following dialog allows the selection of the baud rate and power options for connection to the target board and monitor code. The default settings are shown below. If the target board is modified with a different crystal frequency then this setting will need to be changed. (Note in this case the kernel will also need to be re-compiled).

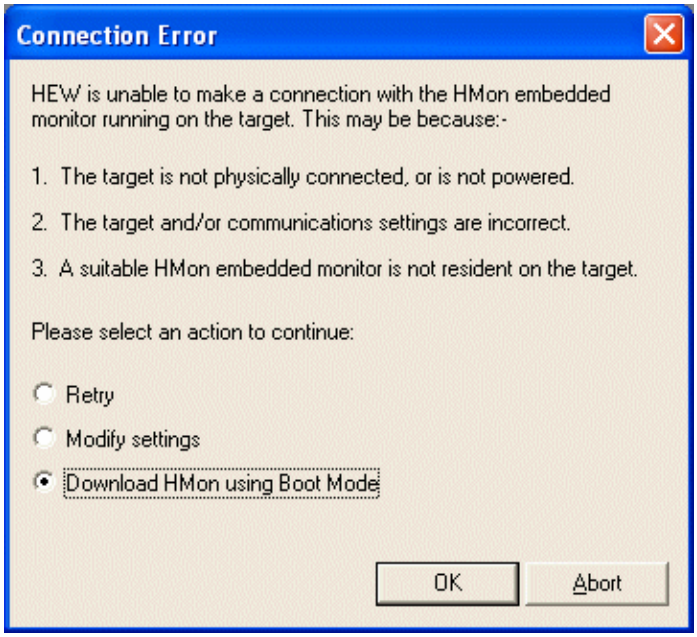
- Confirm the settings shown and press <OK>.



HMon will attempt to connect to the RSK. If it succeeds then the Output window in HEW will show the 'Connected' message. In this case please proceed to Chapter 6.

If the connection fails you will be returned to the previous dialog with the 'Abort' button enabled. This is likely to be caused by a connection error with the RSK and the E8. It can also be caused if the RSK has not got a working copy of the HMon target code programmed. Check the settings and if all settings are OK but the connection still fails then a Boot Mode download will be required. Press <Abort>.

- In the "Connection Error" dialog box, select 'Download HMon using Boot Mode'.
- Press <OK>.



Click <OK> to any warning dialogs. Click <Finish> in Boot Mode pin configuration dialog.

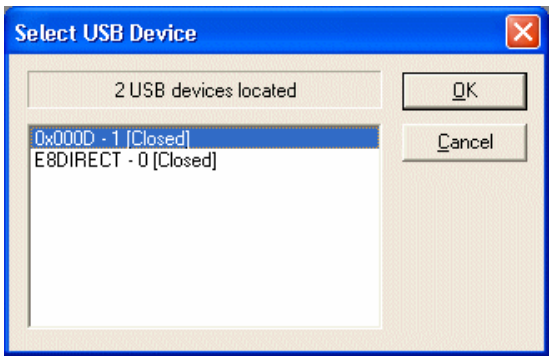
You will be presented with "Select USB Device" dialog box.

Warning: Please ensure that the RSK is powered before connecting a USB cable to the USBDirect port. Failing to do this may damage the board.

Connect another USB cable to the 'USB' connector on the RSK and spare USB port on your PC.

If this is the first time you are connecting the USBDirect port to the PC, you need to install the driver required for USBDirect. Follow the steps given on the screen to install the driver.

- Ensure '0x000D – 1 [Closed]' is selected and click <OK>.

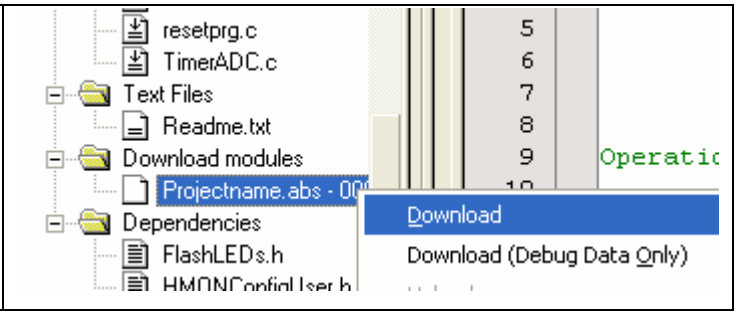


On completion of the download HMon will automatically re-connect to the monitor and revert back to User programming mode.

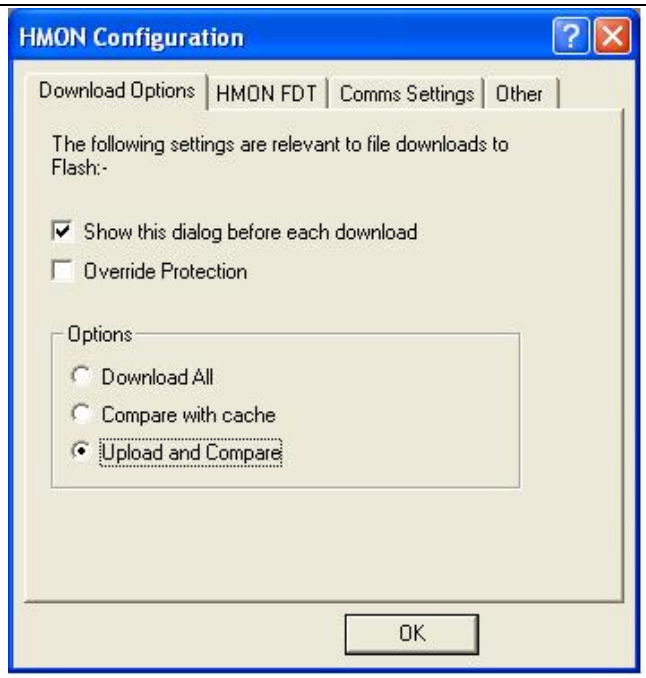
Chapter 6. Downloading and Running the Tutorial

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’	

The download options dialog will be displayed.

This dialog provides various options for HMon operation and downloading. This dialog allows the re-configuration of any HMon communication settings. Please review the settings in each tab. HMon includes a cache of the current device program. This allows the number of reprogramming cycles to be reduced by not programming areas of the device that have not changed between compilations. On first connection the cache is empty. Selecting ‘Upload and Compare’ will read the program back from the device before comparing it to the cache. • Select any of the download options shown. • Press ‘OK’.	

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 Tool Bar. 

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

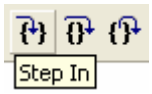
```
66                  void PowerON_Reset(void)
67                  {
68                      /* Renesas H8/SH Compiler Intrinsic function to copy initialised data to RAM */
69                      /* This function references the structure in the 'dbsct.c' source file */
70  00ffa000        ➪   _INITSCT();
71
72                      /*  Setup the hardware. */
73  00ffa004            HardwareSetup();
74
75                      /*  Call user 'main()'. */
76  00ffa008            main();
```

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(); 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’
- Press ‘Reset Go’ on the Debug Tool Bar.

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

- Press ‘Step In’ on the Debug Tool Bar.



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

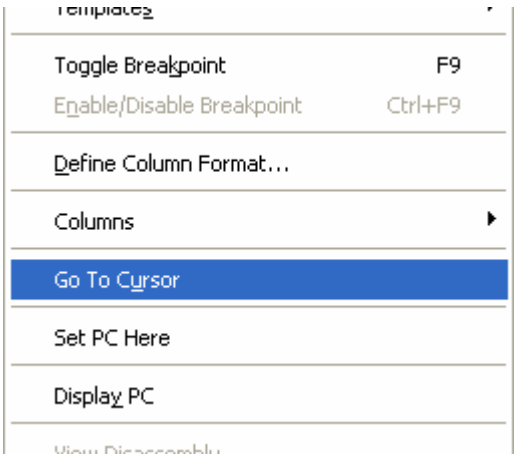
```
66
67  00ffa4e0        ➪void main(void)
68                  {
69                      unsigned char ucCount;
70                      unsigned char ucConvert;
71                      unsigned short usDelay;
72
73                      /*  Reset the LCD module. */
74  00ffa4e4            InitialiseDisplay();
75
76                      /*  Display Renesas Splash Screen. */
77  00ffa4e8            DisplayString(LCD_LINE1,"Renesas");
78  00ffa4f4            DisplayString(LCD_LINE2,NICKNAME);
79
80                      /*  Flash the user LEDs for some time or until a key is pressed. */
81  00ffa500            FlashLEDs();
82
83                      /*  Flash the user LEDs at a rate set by the user potentiomenter (ADC) using
84                          interrupts. */
85  00ffa504            TimerADC();
86
87                      /*  Demonstration of initialised variables. Use this function with the
88                          debugger.   */
89  00ffa508            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 a breakpoint on the ‘TimerADC();’ function call.

```
85  00ffa504  ●     TimerADC();
```

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



The code will run to the selected line and stop. A temporary breakpoint was automatically inserted in the code and then removed when the program stopped at the breakpoint.

- 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 any of user switches (i.e. SW1, SW2 or SW3) 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 breakpoint 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_TGI1B_TPU1(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.

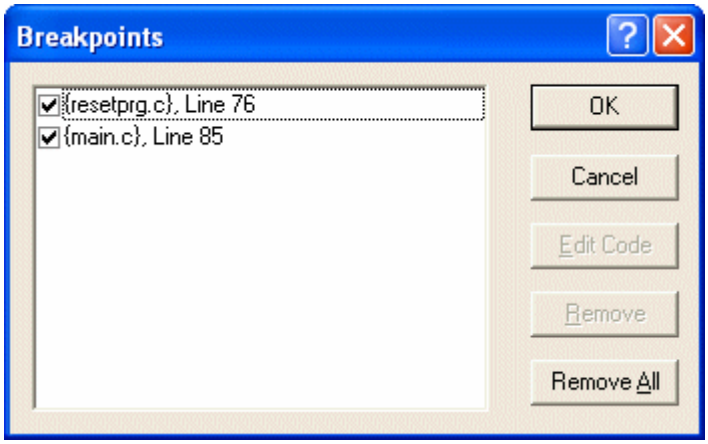
The code will stop in the interrupt routine. It is now possible to step through the interrupt function.

- Remove the breakpoint in the interrupt by double clicking again before exiting the function.
- Press '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. You can modify the flashing rate by adjusting the potentiometer on the board.

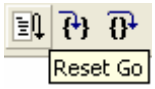
• Press 'Stop' on the debug tool bar.	

- Press 'CTRL-B' to open the breakpoint window.
- Select 'Remove All'
- Press <OK>.



- Open the file 'main.c'
- Insert a breakpoint at the call to the function '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 Tool Bar.	

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

• 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'.

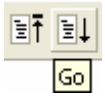
A dialog will open showing the variable and allowing further details to be explored.

- 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 breakpoint 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 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’.

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 / Tool-chains.

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 / resetprg.h)

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.

.

```
66                  void PowerON_Reset(void)
67                  {
68                      /* Renesas H8/SH Compiler Intrinsic function to copy initialised data to RAM */
69                      /* This function references the structure in the 'dbsct.c' source file */
70  00ffa000        ➪  _INITSCT();
71
72                      /*  Setup the hardware. */
73  00ffa004            HardwareSetup();
74
75                      /*  Call user 'main()'. */
76  00ffa008            main();
```

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)

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.

```
/***********************************************************************************
Function Name:  HardwareSetup
Description:    Sets up hardware
Parameters: none
Return value: none
***********************************************************************************/

void HardwareSetup(void)
{
    ConfigureOperatingFrequency();

    ConfigurePortPins();

    EnablePeripheralModules();

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

```
60                  Function Name : main
61                  Description   : Main function.
62                          This function calls timer, ADC & LCD initialisation functions. The user
63                          LEDs flashes until the user presses a switch on the RSK.
64                  Parameters    : none
65                  Return value  : none
66                  ******************************************************************************/
67  00ffa4e0        void main(void)
68                  {
69                      unsigned char ucCount;
70                      unsigned char ucConvert;
71                      unsigned short usDelay;
72
73                      /*  Reset the LCD module. */
74  00ffa4e4            InitialiseDisplay();
75
76                      /*  Display Renesas Splash Screen. */
77  00ffa4e8            DisplayString(LCD_LINE1,"Renesas");
78  00ffa4f4            DisplayString(LCD_LINE2,NICKNAME);
79
80                      /*  Flash the user LEDs for some time or until a key is pressed. */
81  00ffa500            FlashLEDs();
82
83                      /*  Flash the user LEDs at a rate set by the user potentiomenter (ADC) using
84                          interrupts. */
85  00ffa504            TimerADC();
86
87                      /*  Demonstration of initialised variables. Use this function with the
88                          debugger.   */
89  00ffa508            Statics_Test();
90
91                      /*  End of user program. This function must not exit. */
92  00ffa50a            while(1);
93                  }
94                  /******************************************************************************
95                  End of function main
```

Chapter 8. Additional Information

For details on how to use High-performance Embedded Workshop (HEW), refer to the HEW User manual available on the CD or from the Manual Navigator installed with this product.

For more information on the configuration and use of HMon please refer to the HMon User Manual installed from the CD.

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

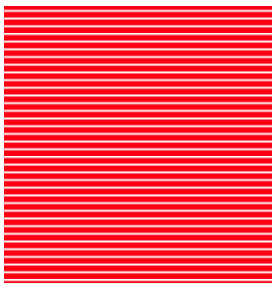
http://www.renesas.com/renesas_starter_kits

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

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

Renesas Starter Kit for H8S2215R
Tutorial Manual
Publication Date Rev.1.00 12.01.2007
Published by: Renesas Technology Europe Ltd.
Dukes Meadow, Millboard Road, Bourne End Buckinghamshire
SL8 5FH, United Kingdom

Renesas Starter Kit for H8S2215R
Tutorial Manual



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