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.

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.

User's Manual

Renesas Starter Kit

RSK SH7201 Tutorial Manual

RENESAS SINGLE-CHIP MICROCOMPUTER

Renesas Electronics
www.renesas.com

Rev.2.00 2007.01

Table of Contents

Chapter 1. Preface 3
Chapter 2. Introduction 4
Chapter 3. Tutorial Project Workspace 5
Chapter 4. Project Workspace 6
4.1. Introduction 6
4.2. Creating a new Project Workspace 6
4.3. Build Configurations and Debug Sessions 7
4.3.1. Build Configuration 7
4.3.2. Debug Session 7
Chapter 5. Building the Tutorial Project 8
5.1. Building Code 8
5.2. Connecting the debugger 8
5.3. Connecting to the target with E8Direct & HMon 9
5.3.1. Connecting To HMon 12
Chapter 6. Downloading and Running the Tutorial 14
Chapter 7. Project Files 19
7.1. Standard Project Files 19
7.1.1. Initialisation code (resetprg.c) 19
7.1.2. Board initialisation code (hwsetup.c) 23
7.1.3. Main tutorial code (main.c) 24
Chapter 8. Additional Information 25

Chapter 1. Preface

Cautions

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

All rights reserved. No one is permitted to reproduce or duplicate, in any form, a part or this entire document without the written permission of Renesas Technology Europe Limited.

Trademarks

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

Copyright

© Renesas Technology Europe Ltd. 2006. All rights reserved.

© Renesas Technology Corporation. 2006. All rights reserved.

© Renesas Solutions Corporation. 2006. All rights reserved.

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

Glossary

BRR	Baud Rate Register	ISR	Interrupt Service Routine
ERR	Error Rate	PWM	Pulse Width Modulation
HMON	Embedded Monitor	CPU	Central Processing Unit
RSK	Renesas Starter Kit	PC	Program Counter
E8	E8 On-chip debugger module	IRQ	Interrupt Request
HEW	High performance Embedded Workshop	NMI	Non-Maskable Interrupt
CCR	Condition Code Register	RTE	Return from Exception
EXR	Extended control Register	LED	Light Emitting Diode
SR	Status Register	RTC	Real Time Clock

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 toolchains or E8 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 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 hmonserialconfiguser.c and hmonconfiguser.c files and the header files; hmonserialconfiguser.h, hmonserialstruct.h, and hmonconfiguser.c. More information on this is provided in the HMon User Manual.

Chapter 4. Project Workspace

4.1.Introduction

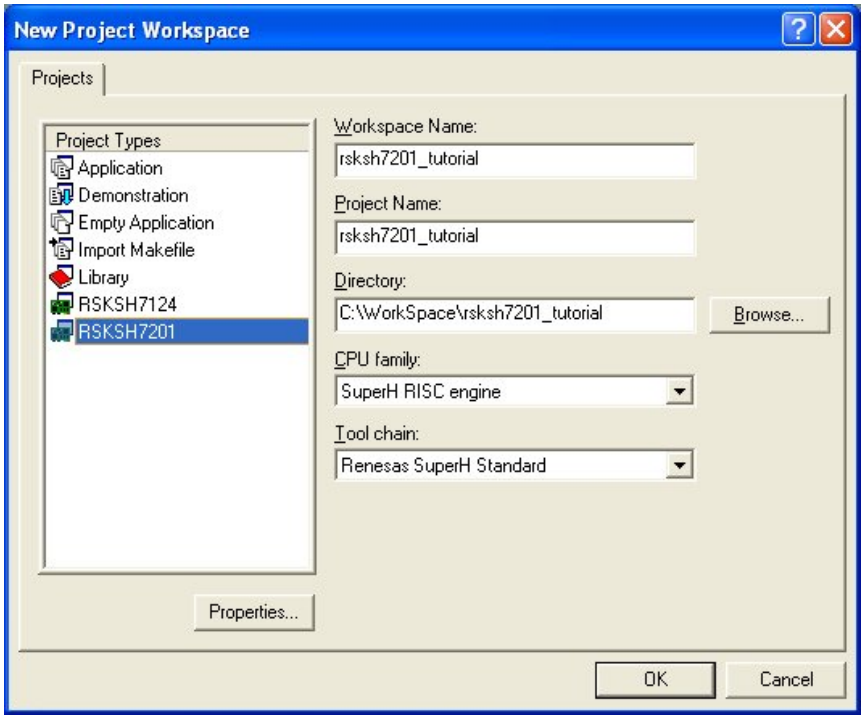
HEW is an integrated development tool that allows the user to write, compile, program and debug a software project on any of the Renesas Microcontrollers. HEW will have been installed during the software installation for the RSK product.

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 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 RSKSH7201 selected.

- Select the 'SuperH RISC Engine' CPU family and 'Renesas SuperH Standard' Toolchain for the RSK
- Select the '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 'Session_SH2A_7201_HMon' debug session.

Chapter 5. Building the Tutorial Project

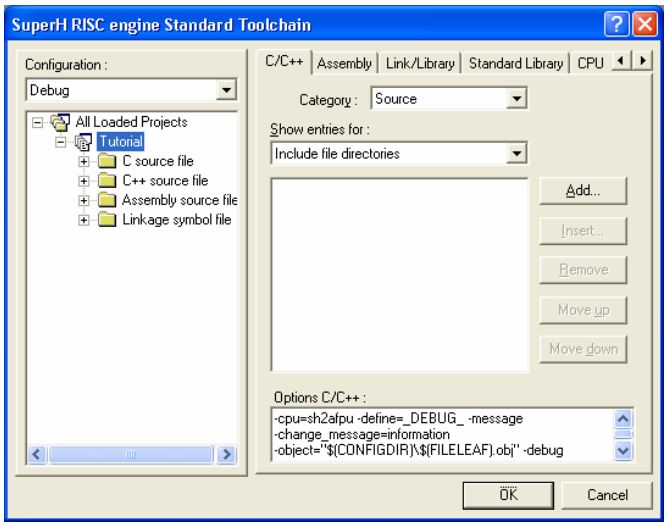
The tutorial project build settings have been pre-configured in the toolchain options. To view the tool chain options select the 'Build' Menu item and the relevant toolchain. This should be the first option(s) on the drop down menu.

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 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 a 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 the re-programming of the device.

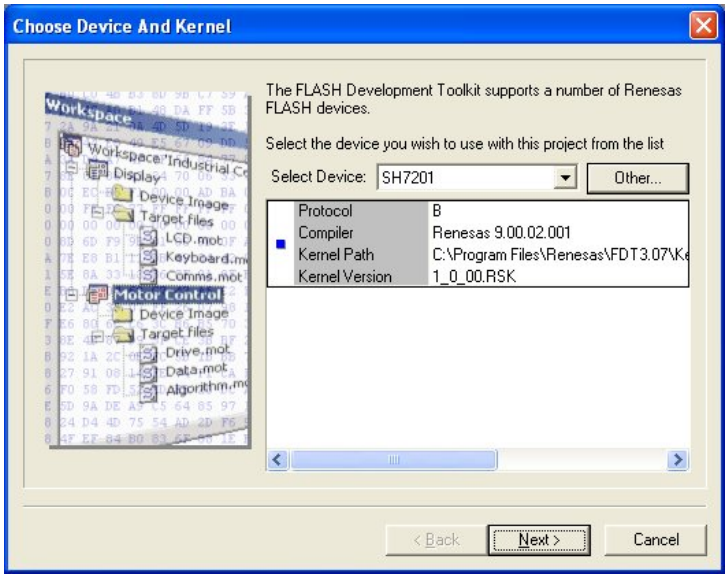
- Select the FDT Wizard from the FDT Tool Bar

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\7201\Renesas\1_0_00.RSK

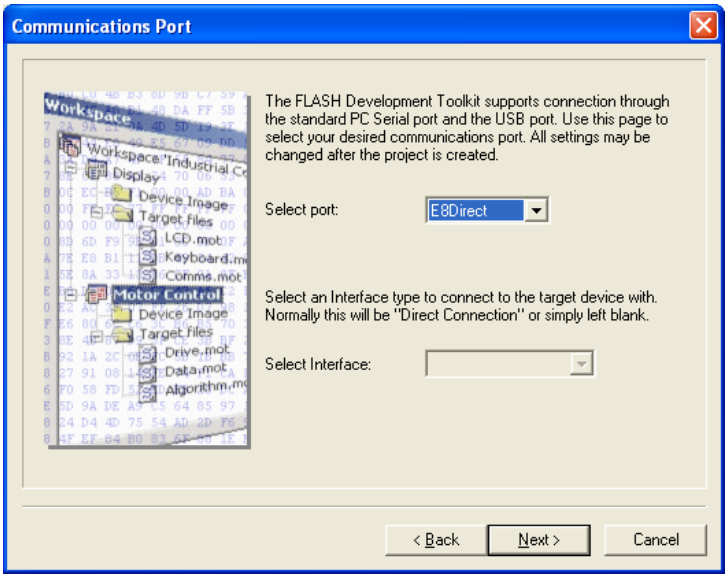
- Select the Device on the RSK from the drop down list.
- In the sub pane, select the kernel version that ends in '.RSK'.
- Press <Next>.



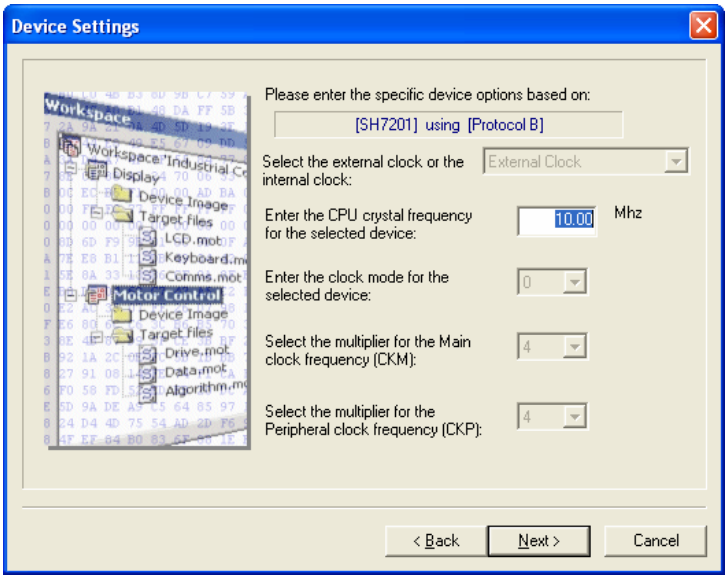
If you have copied the kernel to another location to modify for a different crystal frequency, the device will not be listed in the sub pane. In this case:

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

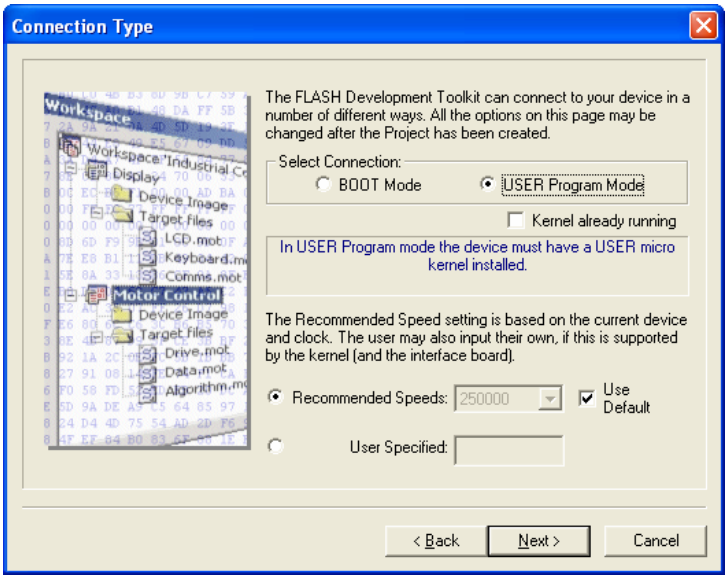
- Select E8DIRECT as the communication Port
- Press <Next>.



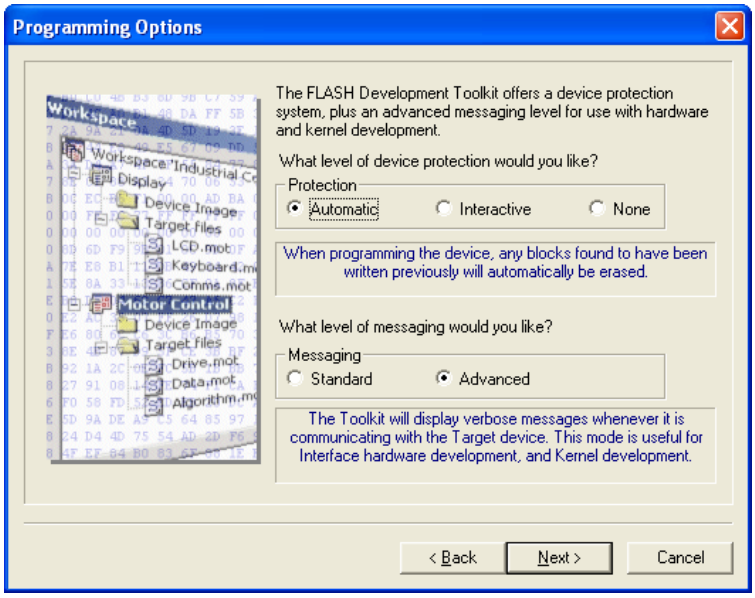
- The default settings are suitable for an un-modified RSK board.
- Confirm the Crystal Frequency matches the board.
- Confirm the main clock frequency multiplier.
- Confirm the peripheral clock multiplier.
- Press <Next>.



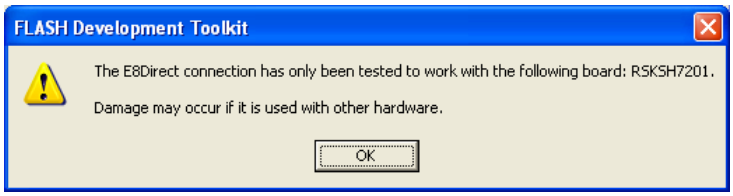
- Ensure that 'USER Program Mode' is selected.
- Confirm that 'Use Default' is selected.
- Press <Next>.



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



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

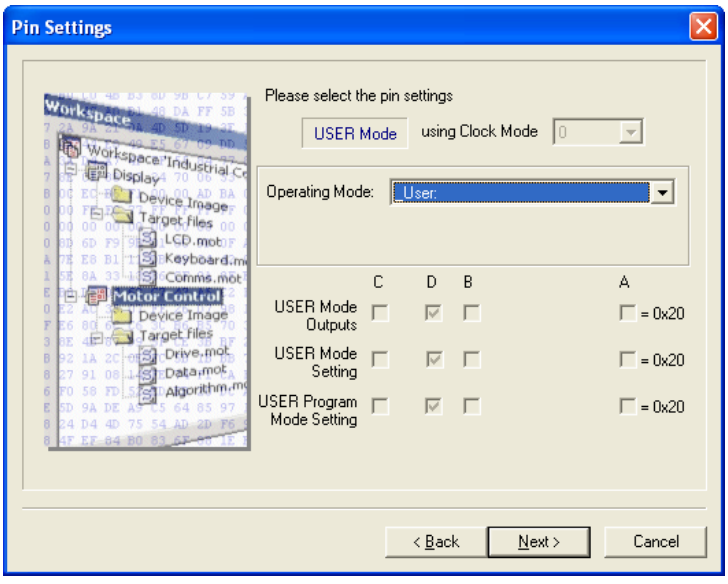


To communicate with the RSK FDT and HMon need to be able to change the operating mode of the microcontroller. To do this there are settings to control the state of the Mode pins via the E8Direct interface. These settings are confirmed in the following screens.

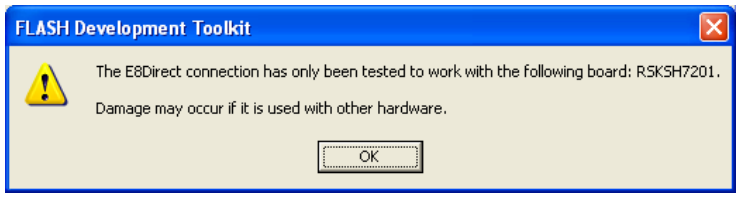
Unless you are very sure that the mode pin settings need to change. Do not modify the default settings.

Damage to the microcontroller can be sustained with incorrect settings.

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

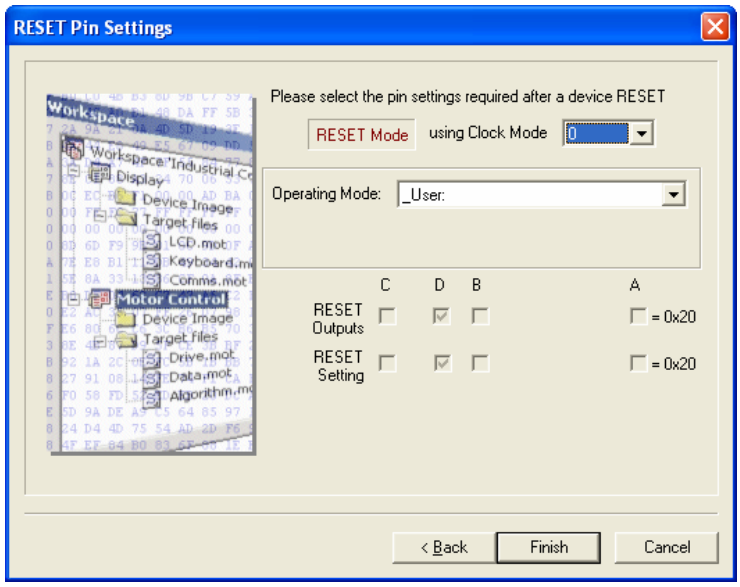


- The following warning dialog will be displayed.
- Press <OK>.



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>.

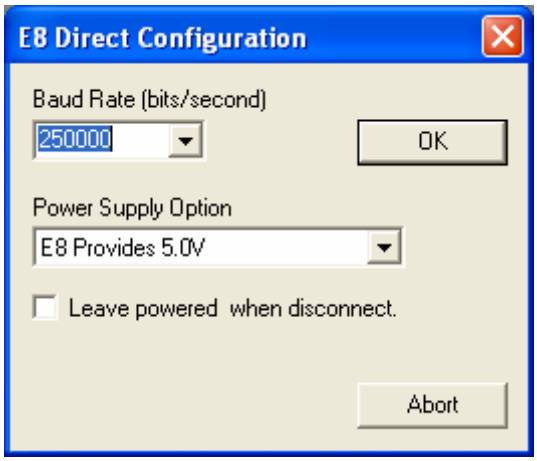
A further dialog will be displayed to confirm if it acceptable to assume only one E8 device will be connected to the host computer while using this project. This is the preferred operating mode. While it is possible to use more than one E8 device this is not recommended.

- 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).

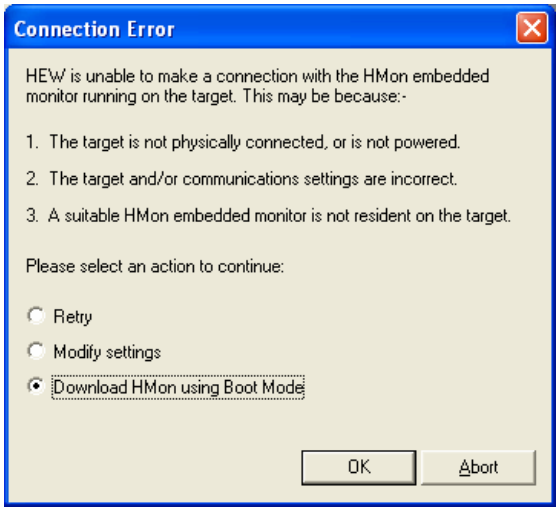
- Confirm the settings shown and press <OK>.



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. 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>.

- Select 'Download HMon using Boot Mode'.
- Press <OK>.



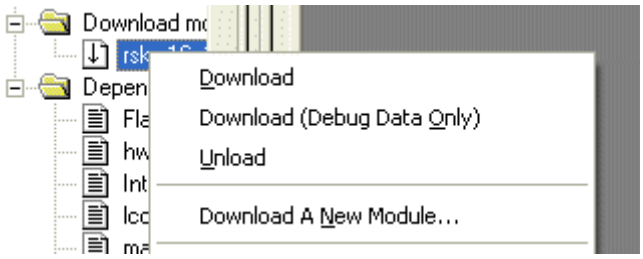
On completion of the download HMon will automatically re-connect to the monitor and revert back to User programming mode.

Chapter 6.Downloading and Running the Tutorial

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

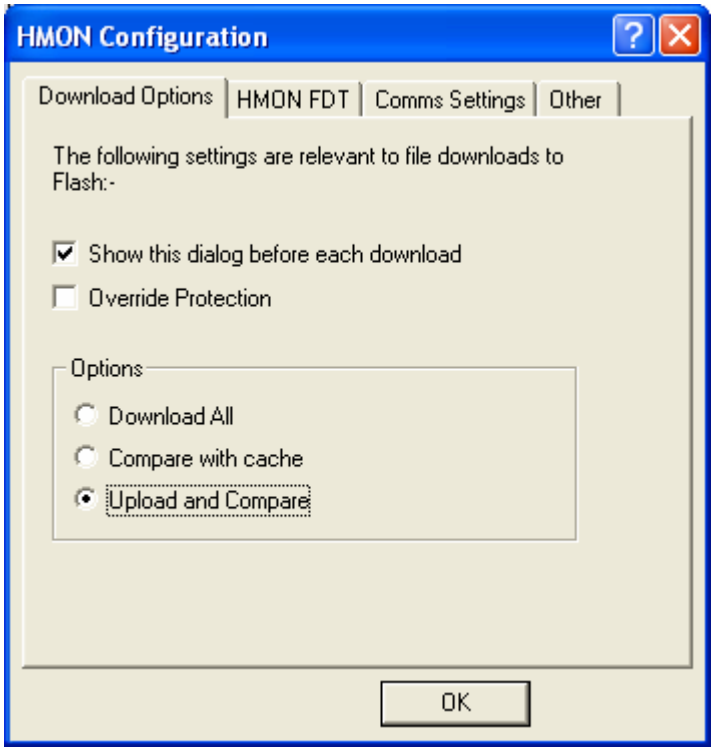
Now that you are connected to the target you should see an additional category in the workspace view called 'Download Modules'

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



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.

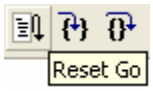
The File window will open the Tutorial code at the entry point. An arrow and a yellow highlight marks the current position of the program counter.

```
void POWER_ON_RESET(void)
{
    size_t  stRamSize;
#ifdef _DEBUG_
    /* Raise the imask to just below HMON. This is so when a CPU reset is performed
       via the HMON GUI it behaves more like the real CPU and does not accept
       interrupts until the mask is lowered */
    set_imask(HMON_DEFAULT_INTERRUPT_LEVEL - 1);

#endif
    /* 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 */
    rstInitialiseLEDs();
    /* Set the Frequency control register */
    rstInitialiseCPG();
    /* Initialise all the ports */
    rstInitialisePORTs();
    /* Configure and check SDRAM */
    stRamSize = rstConfigureSDRAM();
    /* If the SDRAM is not working or of insufficient size */
    if (stRamSize < RSK_SDRAM_SIZE)
    {
        /* Flash the LEDs to show the error */
        rstFatalError();
    }
    /* Initialise the C/C++ memory sections */
    INITSCT();
```

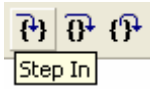
We will now skip over the initialisation code and proceed to the main tutorial.

- Scroll down the file, in the function CStartUp you will see the call to main.
- Place 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 (Please note that you may have to press the button several times).

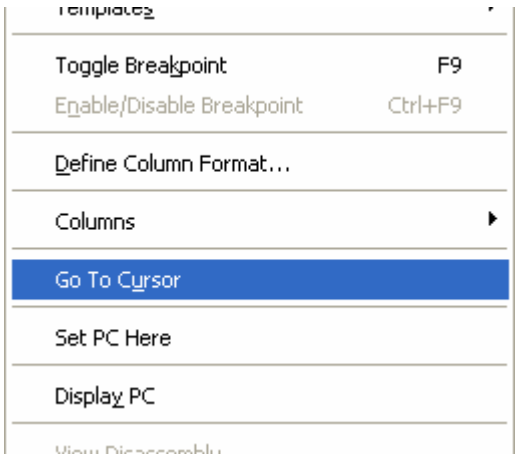


The code window will open ‘main.c’ and show the new position of the program counter.

```
65                  /*****************************************************************************
66                  Function Name : main
67                  Description   : Main function.
68                          This function calls timer, ADC & LCD initialisation functions. The user
69                          LEDs flashes until the user presses a switch on the RSK.
70                  Parameters    : none
71                  Return value  : none
72                  *****************************************************************************/
73  0800069c        void main(void)
74                  {
75                      /*  Reset the LCD module. */
76  0800069e            InitialiseDisplay();
77
78                      /*  Display Renesas Splash Screen. */
79  080006a2            DisplayString(LCD_LINE1,"Renesas");
80  080006b2            DisplayString(LCD_LINE2,NICKNAME);
81
82                      /*  Flash the user LEDs for some time or until a key is pressed. */
83  080006c0            FlashLEDs();
84
85                      /*  Flash the user LEDs at a rate set by the user potentiomenter (ADC) using
86                          interrupts. */
87  080006ca            TimerADC();
88
89                      /*  Demonstration of initialised variables.
90                          Use this function with the debugger. */
91  080006d4            Statics_Test();
92
93                      /*  End of user program. This function must not exit. */
94  080006de            while(1);
95  080006f0        }
96                  /*****************************************************************************
97                  End of function main
98                  *****************************************************************************/
```

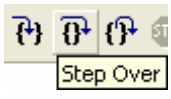
Support for the LCD display is included in the tutorial code. We do not need to be concerned about the details of the LCD interface – except that the interface is write-only and so is not affected if the LCD display is attached or not.

- Insert a breakpoint on the ‘TimerADC();’ function call.
- 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 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 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_MTU2_1_TGIA1(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.

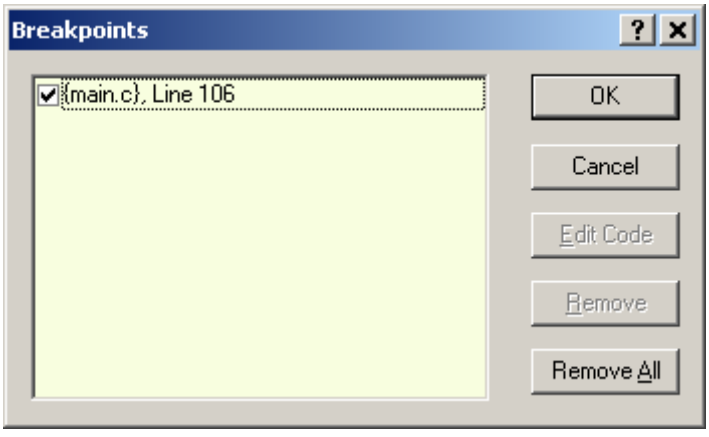
The code will stop in the interrupt routine. It is now possible to step through the interrupt function.

- 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. If the RSK supports an ADC you can modify the flashing rate by adjusting the potentiometer on the board.

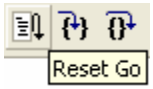
- Press 'Stop' on the debug tool bar.

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

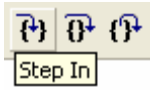


- Open the file ‘main.c’
- Insert a breakpoint on ‘StaticsTest();’.

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

- Press ‘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 / 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 code after a power on reset. POWER_ON_RESET is the entry point defined by the Power on reset vector.

```
#pragma entry POWER_ON_RESET

void POWER_ON_RESET(void)
{
    size_t  stRamSize;
#ifdef _DEBUG_
    /* Raise the imask to just below HMON. This is so when a CPU reset is performed
       via the HMON GUI it behaves more like the real CPU and does not accept
       interrupts until the mask is lowered */
    set_imask(HMON_DEFAULT_INTERRUPT_LEVEL - 1);

#endif
    /* 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 */
    rstInitialiseLEDs();
    /* Set the Frequency control register */
    rstInitialiseCPG();
    /* Initialise all the ports */
    rstInitialisePORTs();
    /* Configure and check SDRAM */
    stRamSize = rstConfigureSDRAM();
    /* If the SDRAM is not working or of insufficient size */
    if (stRamSize < RSK_SDRAM_SIZE)
    {
        /* Flash the LEDs to show the error */
        rstFatalError();
    }
    /* Enable cache */
    P_CACHE.CCR1.LONG = (BIT_11 | BIT_8 | BIT_0);
    /* Initialise the C/C++ memory sections */
    INITSCT();
    /* Make sure all the data gets into physical memory */
    writeBackCache();
    /* Set the VBR after the sections have been mapped. Since the vector table
       can be located in ROM or RAM it may need to be mapped. If it is mapped
       then the VBR can't be set until the section initialisation is complete */
    set_vbr(gVectorTable);
    /* Jump to the C Start Up function and load the stack pointer (R15) with
       the value defined by the linker from section "S" */
    set_sp_jmp(OUL, CStartUp, (void*)__secend("S"));
}
```

Unlike other RSK products the SH2A has a different initialisation procedure. This is because the ROM (External FLASH memory) is much slower than the SDRAM, therefore to obtain the best performance from the system; the initialisation routines should copy the executable code into SDRAM. The SH7201 also has incredibly fast interrupt handling facilities, so to get the best performance the interrupt vector table

should reside in a fast memory too. The fastest memory of all is the internal RAM and this is the best place for DSP functions, time critical functions, interrupt service routines and the Interrupt Vector Table. When the CPU receives a power on reset the value of the PC and SP are loaded from the reset vector table. Initially the PC is set to H'FFF88000 (the top of internal RAM). This is so the initialisation code can configure the external memory interface before continuing with system initialisation. Before the SDRAM interface can be configured the Clock Pulse Generator needs to be set-up, since this unit generates the bus clock signals. This is performed by the rstInitialiseCPG() function call. Then the ports that make up the SDRAM interface are configured. Lastly the SDRAM interface is setup, the routine supplied here also performs a quick non-destructive test to make sure that the SDRAM is functional and of the expected size. The section initialisation (INITSCT) function is usually provided by the C run time library but in this case it is in the file dbsct.c. The INITSCT function copies the program code and initialised data segments to their destinations. The table below details the ROM section names, functions and mapped locations for a release build of the code. The C$DSEC section contains a table of pointers to the sections in ROM and the destinations in RAM.

Release				
Section	ROM Section	Mapped Section	Mapped Location	Function
Reset Vector Table	CRSTVECT	N/A	N/A	System Initialisation
Initialisation Code	PINIT			
Initialisation Constants	CINIT			
Initialised Data Table	C$DSEC			
Uninitialised Data Table	C$BSEC			
Program Code	P	MP	SDRAM	Main Application Code
Constant Data	C	MC		
Initialised Data	D	MD		
Interrupt Vector Table	DVECTTBL	MDVECTTBL	Internal RAM H'FFF80000 to H'FFF83FFF	Interrupt Vector Table
Kernel Code	PKERNEL	MPKERNEL		Time Critical Code
Kernel Constants	CKERNEL	MCKERNEL		
Kernel Initialised Data	DKERNEL	MDKERNEL	Internal RAM H'FFF84000 to H'FFF87FFF	Time Critical initialised data

Table 7-1: Linker sections for Release build

The debug build is similar although many of the sections are not mapped because HMON downloads the program code and data directly to the destination RAM. However, HMON also requires its own sections. To enable HMON to start it uses the power on and manual reset vectors and then requires an additional section CUSERRSTVECT which contains the application's power on and manual reset vectors. Also the main HMON code must be located in ROM. This enables HMON to start and download the user's application code after a power on reset. It is important that the MPHMON_KERNEL is mapped into internal RAM and the MPHMON_NO_CACHE section is mapped into an area of memory which is not cached (see section 9.3 of the SH7201 hardware manual for more details).

Debug				
Section	ROM Section	Mapped Section	Mapped Location	Function
Reset Vector Table	CRSTVECT	N/A	N/A	System Initialisation
User Reset Vectors	CUSERRSTVECT			
Initialisation Code	PINIT			
Initialisation Constants	CINIT			
HMON Code	PHMON			HMON Code
HMON Constants	CHMON			
HMON kernel code	PHMON_KERNEL	MPHMON_KERNEL	Internal RAM	HMON Code
HMON code that must reside in non cached memory	PHMON_NO_CACHE	MPHMON_NO_CACHE	Internal RAM	HMON Code
Interrupt Vector Table	DVECTTBL	MDVECTTBL	Internal RAM	Interrupt Vector Table

Table 7-2: Linker sections for Debug build

After section initialisation the remainder of the system initialisation can now take place. The function CStartUp calls the hardware setup function, then performs any other initialisation required for the C run time library.

Finally the main function is called.

```
/******************************************************************************
Function Name: CStartUp
Description:   C run time library start up
Parameters:    none
Return value:  none
******************************************************************************/
static void CStartUp(void)
{
    /* Set up the hardware */
    HardwareSetup();
    /* Remove the comment when you use global class object */
//    _CALL_INIT();
    /* Remove the comment when you use ANSI CRT IO functions */
//  _INIT_IOLIB();
    /* Remove the comment when you use errno */
//    errno = 0;
    /* Remove the comment when you use rand() */
//    srand((unsigned int)1);
    /* Remove the comment when you use strtok() */
//    _slptr = (void*)0;
    /* Unmask all interrupts */
    set_imask(0);
    nop();
    /* Switch the LEDs off */
    RSK_LED0 = RSK_LED_OFF;
    RSK_LED1 = RSK_LED_OFF;
    RSK_LED2 = RSK_LED_OFF;
    RSK_LED3 = RSK_LED_OFF;
    /* Call the main entry function */
    main();
    /* Remove the comment when you use ANSI CRT IO functions */
//    _CLOSEALL();
    /* Remove the comment when you use global class object */
//    _CALL_END();
    /* Power down hardware */
    //HardwarePowerDown();
    sleep();
}
/******************************************************************************
End of function CStartUp
******************************************************************************/
```

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)

Further hardware initialisation should be performed in this function. For debug purposes, the interrupt priority registers are initialised to zero (as they would be after a power on reset) bar the one used for the serial port used by HMON. This stops an interrupt being processed during the hardware setup routines when the code is re-run by HMON. In the case of the tutorial the interrupts are configured and the LCD interface is initialised.

```
Return value:  none
******************************************************************************/
void HardwareSetup(void)
{
#ifdef _DEBUG_
                                    /* Reset all interupt priorities bar the SCIF
                                       used for HMON */
    P_INTC.IPR01.WORD = 0;
    P_INTC.IPR02.WORD = 0;
    P_INTC.IPR05.WORD = 0;
    P_INTC.IPR06.WORD = 0;
    P_INTC.IPR07.WORD = 0;
    P_INTC.IPR08.WORD = 0;
    P_INTC.IPR09.WORD = 0;
    P_INTC.IPR10.WORD = 0;
#if SCIF_PORT_INDEX == 0
    P_INTC.IPR11.WORD &= 0x00F0;
#elif SCIF_PORT_INDEX == 1
    P_INTC.IPR11.WORD &= 0x000F;
#else
    P_INTC.IPR11.WORD = 0;
#endif
#if SCIF_PORT_INDEX == 2
    P_INTC.IPR12.WORD &= 0xF000;
#elif SCIF_PORT_INDEX == 3
    P_INTC.IPR12.WORD &= 0x0F00;
#elif SCIF_PORT_INDEX == 4
    P_INTC.IPR12.WORD &= 0x00F0;
#elif SCIF_PORT_INDEX == 5
    P_INTC.IPR12.WORD &= 0x000F;
#else
    P_INTC.IPR12.WORD = 0;
#endif
#if SCIF_PORT_INDEX == 6
    P_INTC.IPR13.WORD &= 0xF000;
#elif SCIF_PORT_INDEX == 7
    P_INTC.IPR13.WORD &= 0x0F00;
#else
    P_INTC.IPR13.WORD = 0;
#endif
    P_INTC.IPR14.WORD = 0;
    P_INTC.IPR15.WORD = 0;
    P_INTC.IPR16.WORD = 0;
#endif

    /* 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;

    /* TODO: Initialise any hardware as required by the user's application */
}
/******************************************************************************
End of function HardwareSetup
```

7.1.3.Main tutorial code (main.c)

The main tutorial code is common to all tutorial projects. The display initialisation and string display functions operate on the LCD display module. In addition to the tutorial code several samples of use of the on chip peripherals are supplied. Check compatibility with a ks0066u controller and pin connection on the schematic before connecting an LCD module not supplied by Renesas.

```
/******************************************************************************
Function Name : main
Description   : Main function.
        This function calls timer, ADC & LCD initialisation functions. The user
        LEDs flashes until the user presses a switch on the RSK.
Parameters    : none
Return value  : none
******************************************************************************/
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
Publication Date Rev.2.00 15.JAN.2007
Published by: Renesas Technology Europe Ltd.
Duke's Meadow, Millboard Road, Bourne End
Buckinghamshire SL8 5FH, United Kingdom

©2006 Renesas Technology Europe and Renesas Solutions Corp., All Rights Reserved.

Renesas Starter Kit for SH7201
Tutorial Manual

Renesas Electronics Corporation
1753, Shimonumabe, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8668 Japan

REG10J0032-0200