Architecture des systèmes à processeurs - Guide rapide d'utilisation - Renesas Starter Kit (RSK) for M32C/87

Christophe BLANC
Université Blaise Pascal
IUT GEII - S2
Email: christophe.blanc@lasmea.univ-bpclermont.fr
Site : www.christophe-blanc.info

17 mars 2009

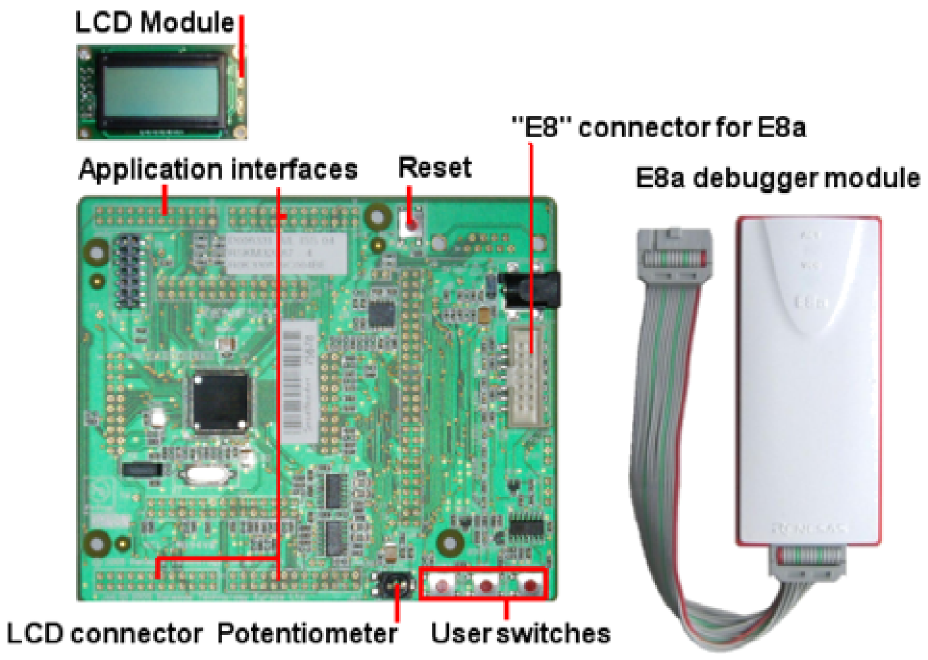


FIGURE 1 – Renesas Starter Kit for M32C/87

1 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.

These tutorials are designed to show you how to use the RSK and are not intended as a comprehensive introduction to the High-performance Embedded Workshop (HEW) debugger, the compiler tool-chains or the E8a module – please consult the relevant user manuals for more in-depth information (Start Menu – > All Programs – > Renesas – > High-performance Embedded Workshop – > Manual Navigator).

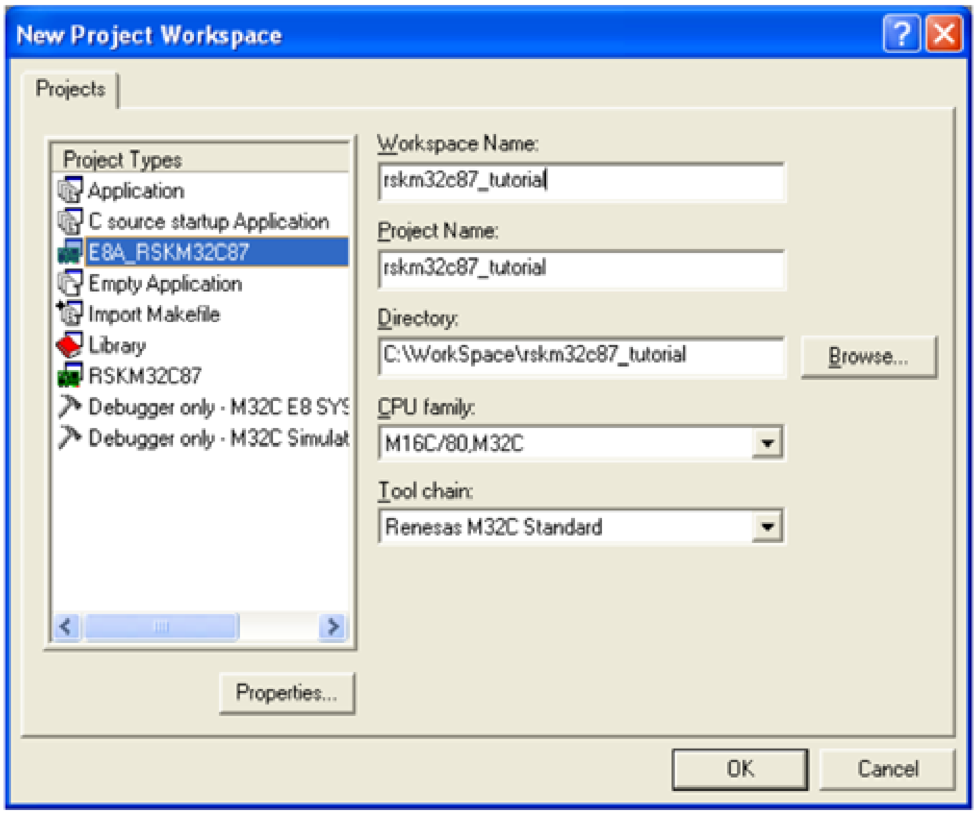
2 Project Workspace

2.1 Introduction

HEW[1] is an integrated development tool that allows the user to write, compile, program and debug a software project on any of the Renesas Microcontrollers. HEW will have been installed during the installation of the software support for the RSK product. This manual will describe the stages required to create and debug the supplied tutorial code.

2.2 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.



Project Types
- Application
- C source startup Application
- E8A_RSKM32C87
- Empty Application
- Import Makefile
- Library
- RSKM32C87
- Debugger only - M32C E8 SYS
- Debugger only - M32C Simulat

Workspace Name: rskm32c87_tutorial
Project Name: rskm32c87_tutorial
Directory: C:\WorkSpace\rskm32c87_tutorial
CPU family: M16C/80,M32C
Tool chain: Renesas M32C Standard

The example above shows the New Project Workspace dialog with the RSKM32C87 selected
- Select the 'M16C/80,M32C' CPU family and Tool-chain for the RSK
- Select the 'E8A_RSKM32C87' Project type for the RSK from the project list
- Enter a name (tutorial) 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.
- 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 HEW.
- To view the file 'main.c', double click on the file in the Workspace window. A new window will open showing the code.

1. High-performance Embedded Workshop

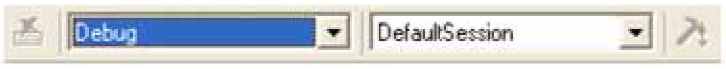
3 Build Configurations and Debug Sessions

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.

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 optimisation on the code being debugged.

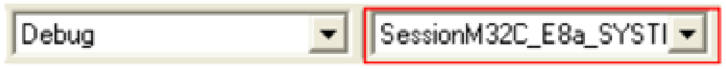
– Select the 'Debug Build' Configuration.



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 sessions is to allow the use of different debugger tools or different debugger settings on the same project.

– Select the 'SessionM32C_E8a_System' debug session.



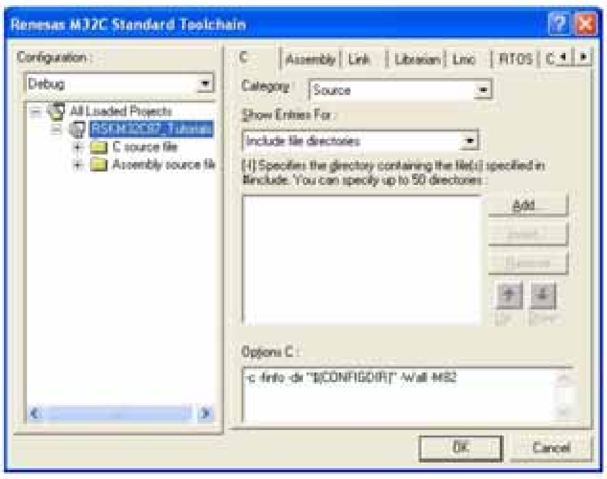
4 Building the Tutorial Project

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 figure **??** 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.

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



[-I] Specifies the directory containing the file(s) specified in #include. You can specify up to 50 directories.

Options C :

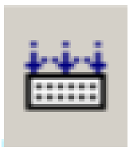
-c -finfo -dir "$(CONFIGDIR)" -Wall -M82

FIGURE 2 – Configuration pane

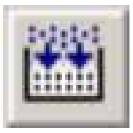
4.1 Building Code

There are three short cuts available for building the project.

1. 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.
2. 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.
3. Press ‘F7’
   This is equivalent to pressing the ‘Build’ button described above.
   Build the project now by pressing ‘F7’ or pressing one of the build icons as shown above.
   During the build each stage will be reported in the Output Window.

4.2 Connecting the debugger

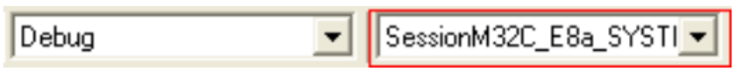
For this tutorial it is not necessary to provide an external power supply to the board. The power will be obtained from the USB port. Please be aware that if you have too many devices connected to your USB port it may be shut down by windows. If this happens remove some devices and try again. Alternatively provide an external power source taking care to ensure the correct polarity and voltage.

1. Connect the E8a debugger to the USB port on your computer.
2. Connect the E8a debugger to the target hardware ensuring that it is plugged into the connector marked J8/E8, located near the power connector.
3.

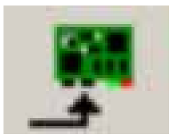
4.3 Connecting to the target with the E8a

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

1. Select the ‘SessionM32C_E8a_System’ debug session.

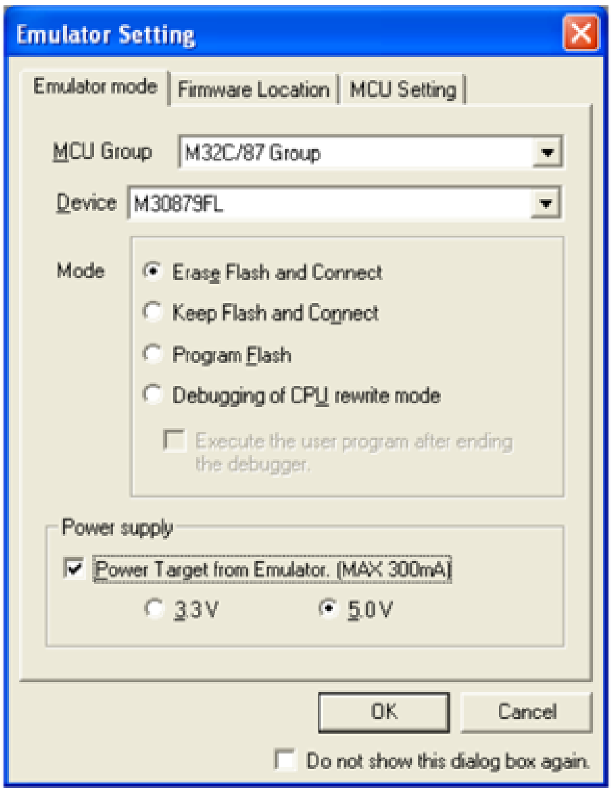
   

2. Click the <Connect> button on the debug toolbar

3. The Emulator setting dialog will be shown.
4. Select the correct “MCU Group” (M32C/87 Group) and “Device” (M30879FL)
5. Select ‘Erase Flash and Connect’.

6. If the E8a is to provide the power to the RSK board, select ‘Power Target from Emulator’ and choose the ‘5.0V’ option.


Emulator Setting

Emulator mode | Firmware Location | MCU Setting

MCU Group: M32C/87 Group

Device: M30879FL

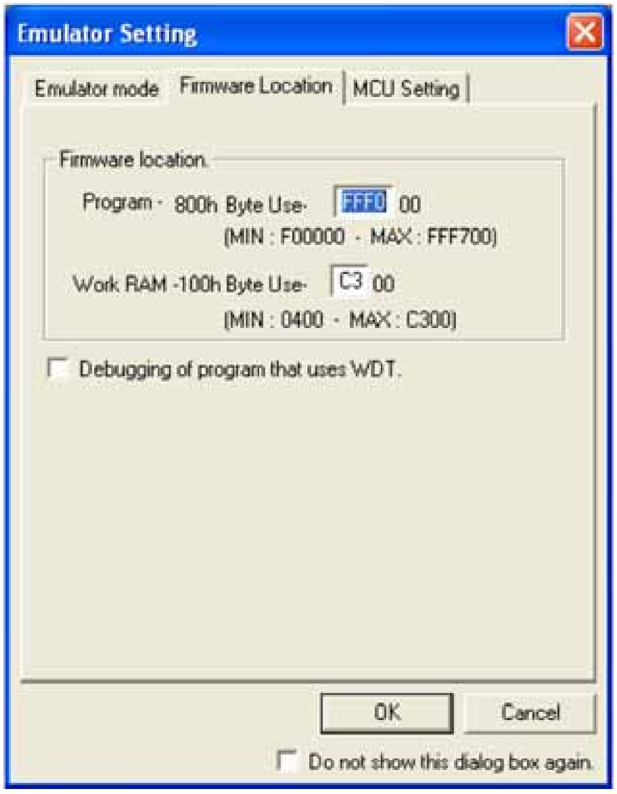
Mode:
- [✓] Erase Flash and Connect
- [ ] Keep Flash and Connect
- [ ] Program Flash
- [ ] Debugging of CPU rewrite mode
  - [ ] Execute the user program after ending the debugger.

Power supply
- [✓] Power Target from Emulator. (MAX 300mA)
- [ ] 3.3V [✓] 5.0V

OK | Cancel

[ ] Do not show this dialog box again.

7. Select the “Firmware Location” tab.
8. Select FFF000 for the Firmware location for the program. It is recommended that the E8a firmware location is towards the top of the memory map.
9. Select C300 for the Work RAM location. It is recommended that the Work RAM is towards the top of the memory map.
10. Ensure that the checkbox “Debugging of program that uses WDT” is not checked.


Emulator Setting

Emulator mode | Firmware Location | MCU Setting

Firmware location

Program - 800h Byte Use- FFF0 00
(MIN : F00000 - MAX : FFF700)

Work RAM -100h Byte Use- C3 00
(MIN : 0400 - MAX : C300)

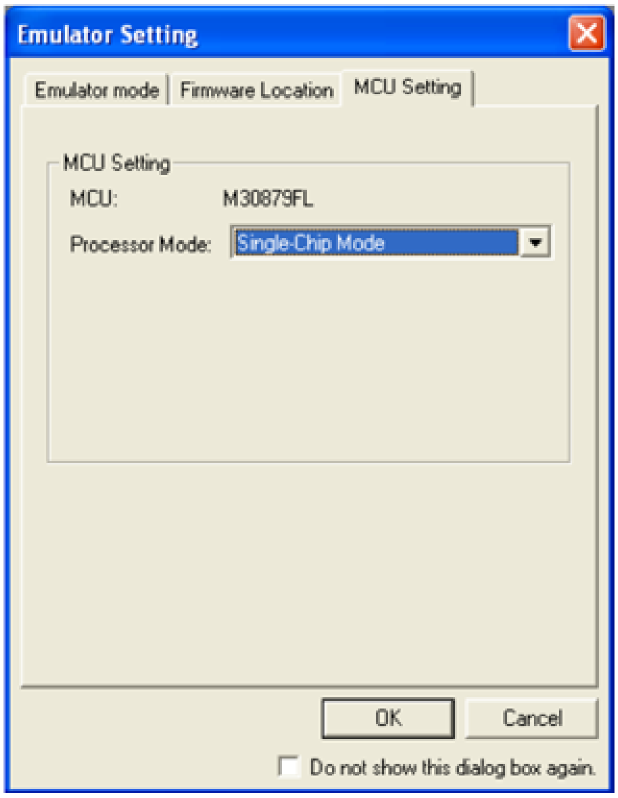
[ ] Debugging of program that uses WDT.

OK | Cancel

[ ] Do not show this dialog box again.

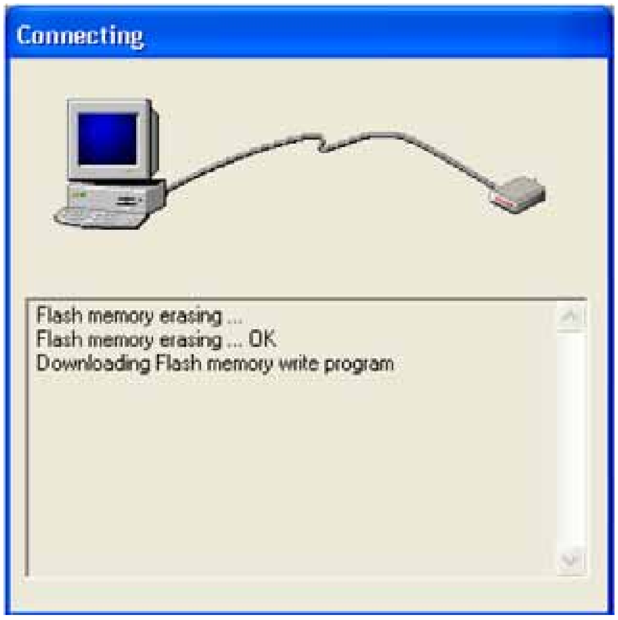
11. Select the MCU Setting tab.

12. Select the processor mode “Single-Chip Mode”.
13. Click <OK>.



MCU Setting
MCU: M30879FL
Processor Mode: Single-Chip Mode

14. The Flash Memory write program is downloaded.
15. The Output window in HEW will state ‘Connected’.



```
Flash memory erasing ...
Flash memory erasing ... OK
Downloading Flash memory write program
```

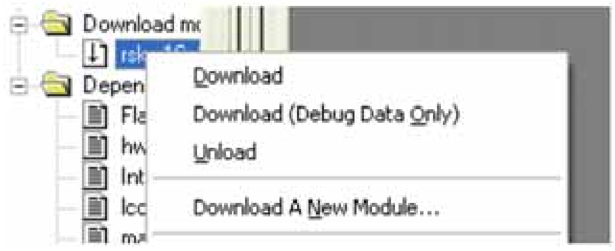
Now is a good time to save the HEW session.

16. Select ‘File’ — ‘Save Session’.
If you have changed any workspace settings now is a good time to save the workspace.
17. Select ‘File’ — ‘Save Workspace’.

5 Downloading and Running the Tutorial

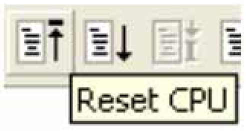
Once the code has been built in HEW it needs to be downloaded to the RSK. There will now be an additional category in the workspace view for ‘Download Modules’

– Right click on the download module listed and select ‘Download Module’

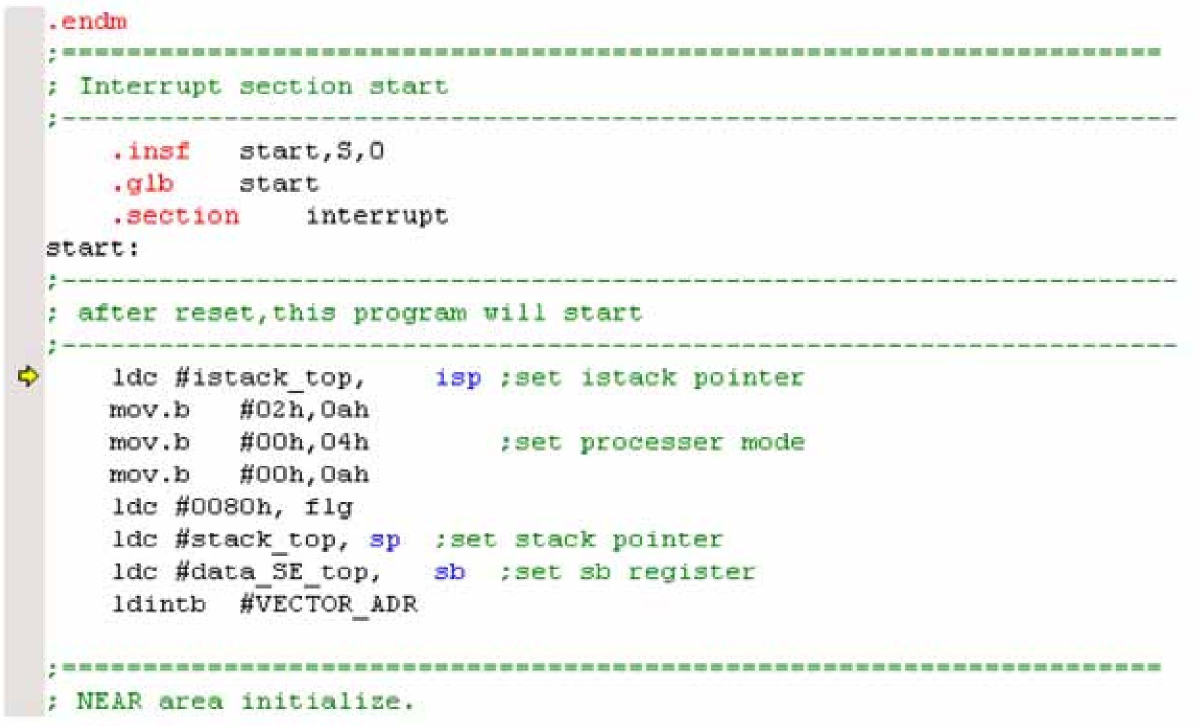


On completion the debugger and code are ready to be executed. To start debugging we need to reset the debugger and target.

– Press ‘Reset CPU’ on the Debug Tool Bar.



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



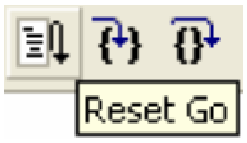
```
.endm
;======================================================================
; Interrupt section start
;----------------------------------------------------------------------
    .insf   start,S,0
    .glb    start
    .section    interrupt
start:
;----------------------------------------------------------------------
; after reset,this program will start
;----------------------------------------------------------------------
    ldc #istack_top,    isp ;set istack pointer
    mov.b   #02h,0ah
    mov.b   #00h,04h        ;set processer mode
    mov.b   #00h,0ah
    ldc #0080h, flg
    ldc #stack_top, sp  ;set stack pointer
    ldc #data_SE_top,   sb  ;set sb register
    ldintb  #VECTOR_ADR

;======================================================================
; NEAR area initialize.
```

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

– Open the file called ‘resetprg.c’.

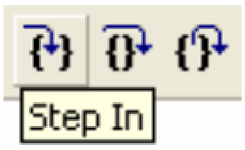
– Place a breakpoint at the call to main() ;
Breakpoints can be set by double clicking in the column containing the PC arrow next to the line to break at ; or selecting the line and pressing F9 ; or right click on the line and select ‘Toggle breakpoint’.

– Press ‘Reset Go’ on the Debug Tool Bar.

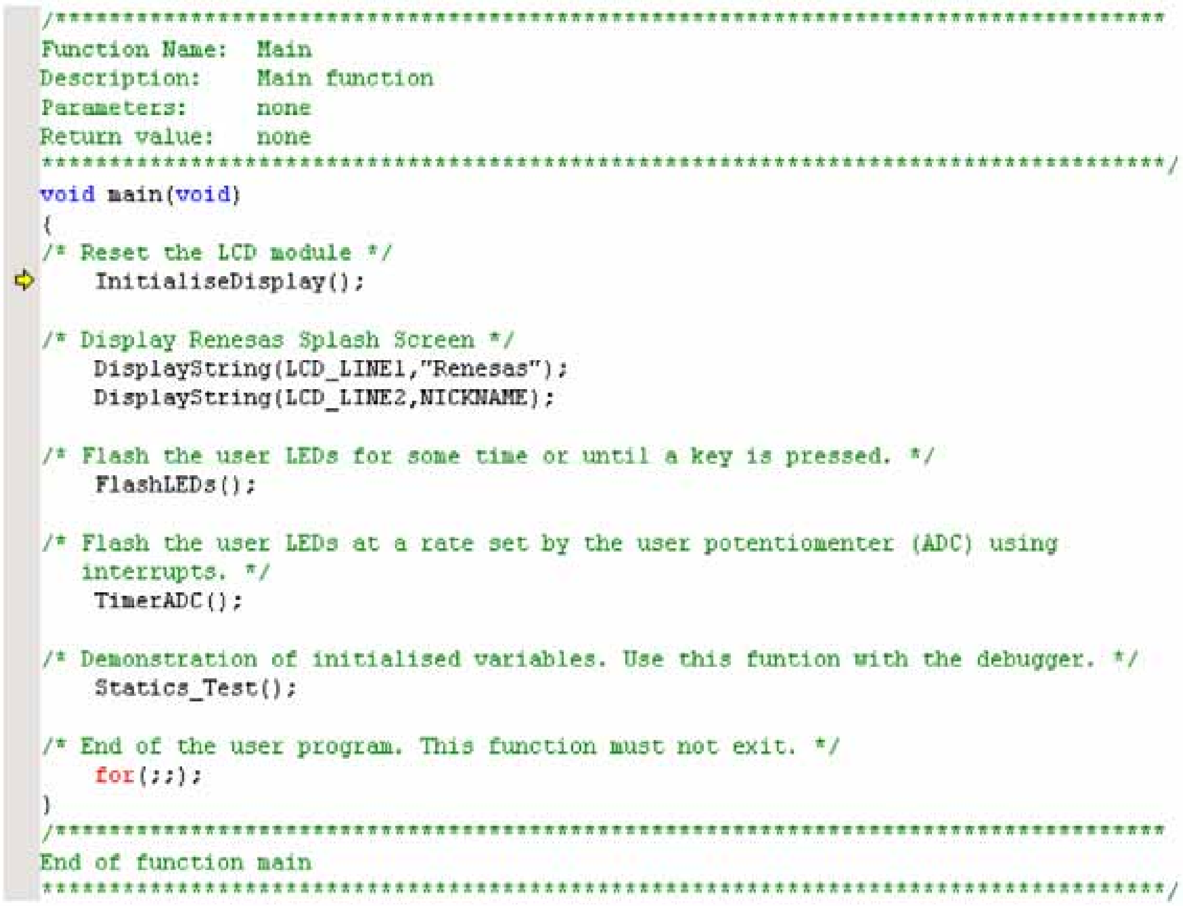


The code will execute to the breakpoint. At this point all the device initialisation will have been completed.

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



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

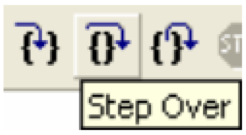


```
/*******************************************************************************
Function Name:  Main
Description:    Main function
Parameters:     none
Return value:   none
*******************************************************************************/
void main(void)
{
/* Reset the LCD module */
    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 funtion with the debugger. */
    Statics_Test();

/* End of the user program. This function must not exit. */
    for(;;);
}
/*******************************************************************************
End of function main
*******************************************************************************/
```

– Insert a breakpoint on the call to the ‘TimerADC’ function.
– Right click on the ‘FlashLEDs()’; function and select ‘Go to cursor’.

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

– Press ‘Step Over’ on the Debug Tool Bar.

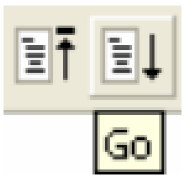


The code will run and flash the LEDs 200 times. The debugger will not exit until all 200 flashes have completed or a button is pressed on the RSK.

– 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 TimerADC function.

– Remove the breakpoint set on the TimerADC function 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. 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.

You have now run the tutorial code and used many of the common features of the debugger.