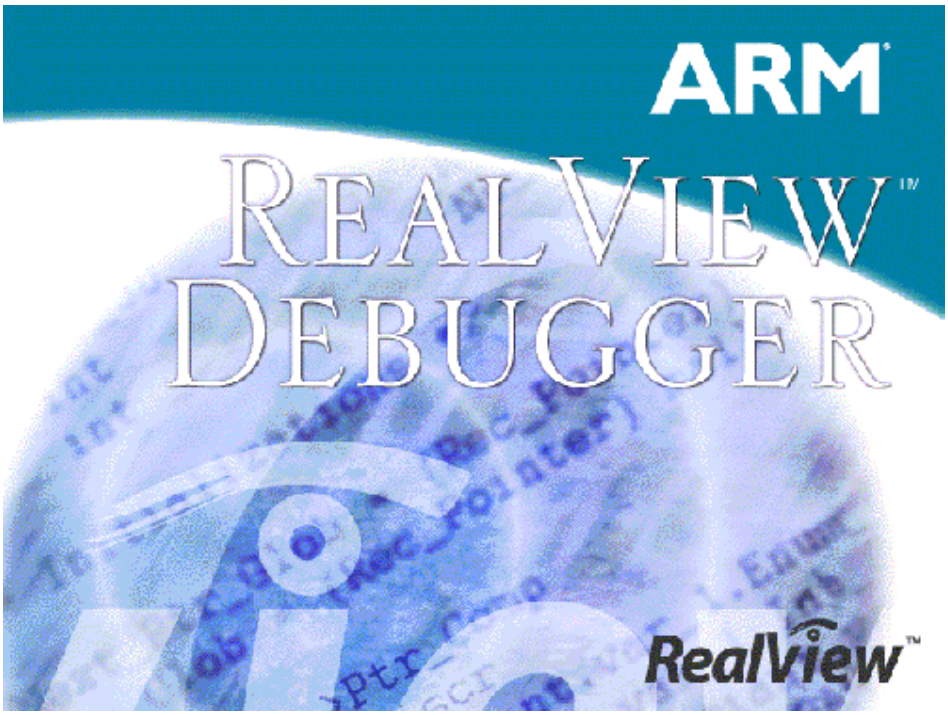
RVDS 2.2
RealView Debugger
&
RealView ICE Tutorial



250v02

Introduction

Aim

This tutorial provides you with a basic introduction to using RealView ICE (RVI) version 1.2 with the ARM RealView Debugger (RVD) version 1.8.

It consists of:

- **Session 1** Configuring the target
- **Session 2** A Simple Hello World project
- **Session 3** Full embedded application
- **Appendix** Integrator AP Motherboard Switch Settings.

Pre-requisites

You should be familiar with Microsoft DOS/Windows, and have a basic knowledge of the C programming language. The ARM RealView Debugger (version 1.8 or greater), RVDS 2.2 (or equivalent code generation tools) and RealView ICE (1.2 or greater) should be available.

The examples are based around the ARM Integrator AP development platform. An ARM920T core module is shown in this tutorial, but any ARM core module is suitable.

The tutorial projects must be installed on your PC.

Note: Explanation of File Extensions:

`.c`	C source file.
`.h`	C header file.
`.o`	object file.
`.prj`	Project file, as used by the RealView Debugger.
`.axf`	ARM Executable file, as produced by **`armlink`**.
`.txt`	ASCII text file.

Additional information

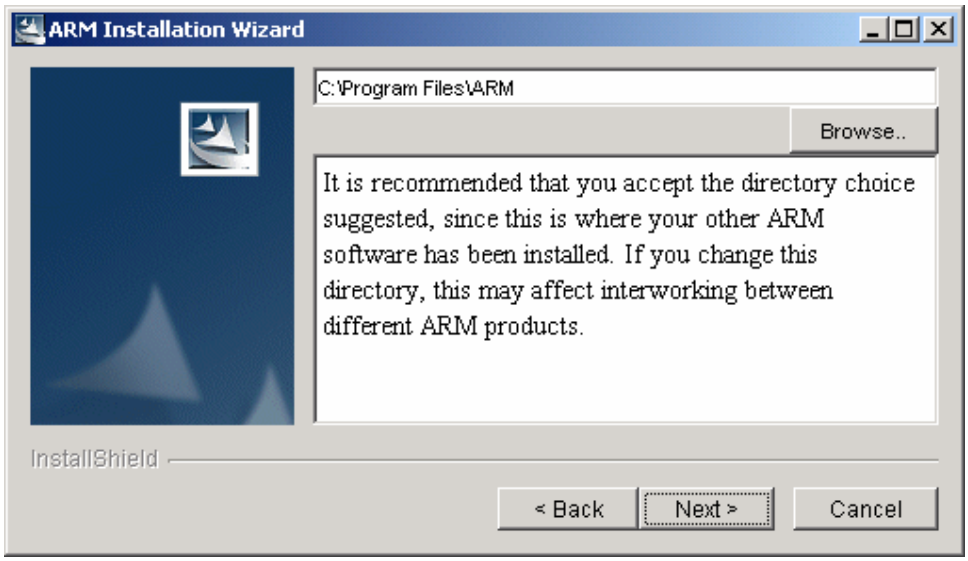
This tutorial is not designed to provide detailed documentation of RVD and RVI. Full documentation is provided with the products.

Further help can be accessed by pressing *F1* when running RVD, from the help menu. The documentation is also available in PDF format. This can be found by going to *Start → Programs → ARM → RealView Developer Suite 2.2 → PDF Documentation*

Before you Begin

1. Install the RealView ICE software.

Follow the instructions on the RealView ICE CD for installation. Make sure that you install the software in the folder you are using for all of your ARM RealView tools.



2. Update the RVD installation

The next time RVD is started it will detect that there are some new target connections are available and will ask you to confirm the required configuration changes:

3. Configure RealView ICE unit

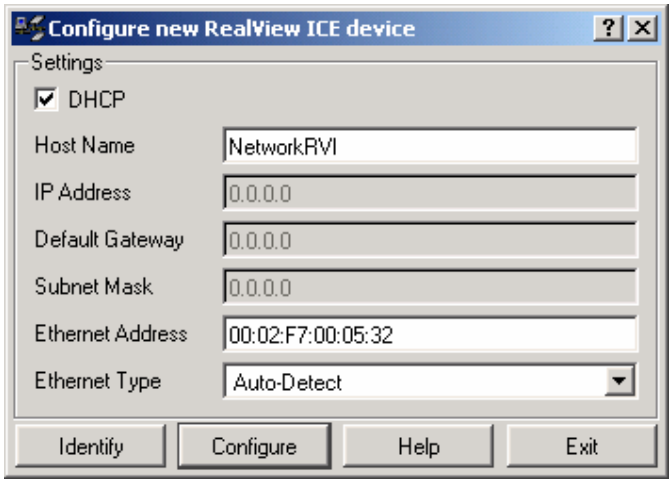
You can connect to your RealView ICE unit using either a local network connection with a DHCP server, or a direct connection when a local network with a DHCP server is not available.

Local network connection

To use a local network connection your computer and RealView ICE unit must be connected to a TCP/IP network which also contains a DHCP server.

1. Launch "RealView ICE Config IP" by selecting *Programs→ARM RealView ICE v1.2→RealView ICE Config IP* from the Windows *Start* menu.

2. Select *RVI→Configure New* from the menu bar to open the configuration dialog box. Make sure that DHCP is selected, and enter a host name and the Ethernet address of your RealView ICE unit. The Ethernet address is a 12-digit number printed on a white sticker located next to the power connection of the RealView ICE unit (e.g. 0002F7000532). After the data is entered, click the *Configure* button and then the *Exit* button.



3. Close the RVI Config IP window.

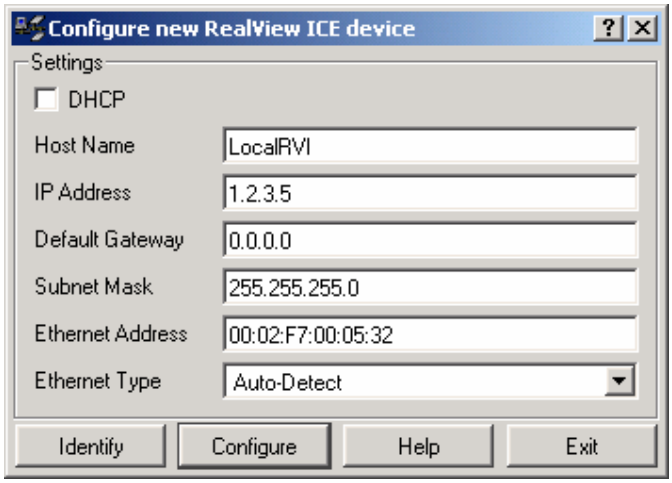
Direct connection

To use a direct connection, you must connect your computer to your RealView ICE unit using either an Ethernet hub or directly using an Ethernet cross-over cable. Using this approach, both the computer and the RealView ICE unit must be configured with static IP addresses.

1. Assign a static IP address to your computer (e.g. 1.2.3.4 using the appropriate technique for your operating system).

2. Assign a static IP address to your RealView ICE unit. To do this, launch "RealView ICE Config IP" by selecting *Programs→ARM RealView ICE v1.2→RealView ICE Config IP* from the Windows *Start* menu.

3. Select *RVI→Configure New* from the menu bar to open the configuration dialog box. Make sure that DHCP is not selected and enter a host name, IP address, subnet mask and the Ethernet address of your RealView ICE unit. The Ethernet address is a 12-digit number printed on a white sticker located next to the power connection of the RealView ICE unit (e.g. 0002F7000532). After the data is entered, click the *Configure* button and then the *Exit* button.



4. Close the RVI Config IP window.

Files for the exercises

Two sets of files are included for the exercises in this tutorial:

`c:\rvds22_tutorial\rvi`	These are the normal files, for use with the Integrator/AP board.
`c:\rvds22_tutorial\rvi_cp`	These files include the changes necessary for the examples to work on the Integrator/CP board.

Session 1 – Configuring the Target

Configure the Integrator board for RAM to appear at address 0x0 (i.e. for the bootROM to run at power up which with remap RAM to 0x0):

Set the Integrator AP Motherboard switches to:
Switch S1-1 on & S1-4 on.

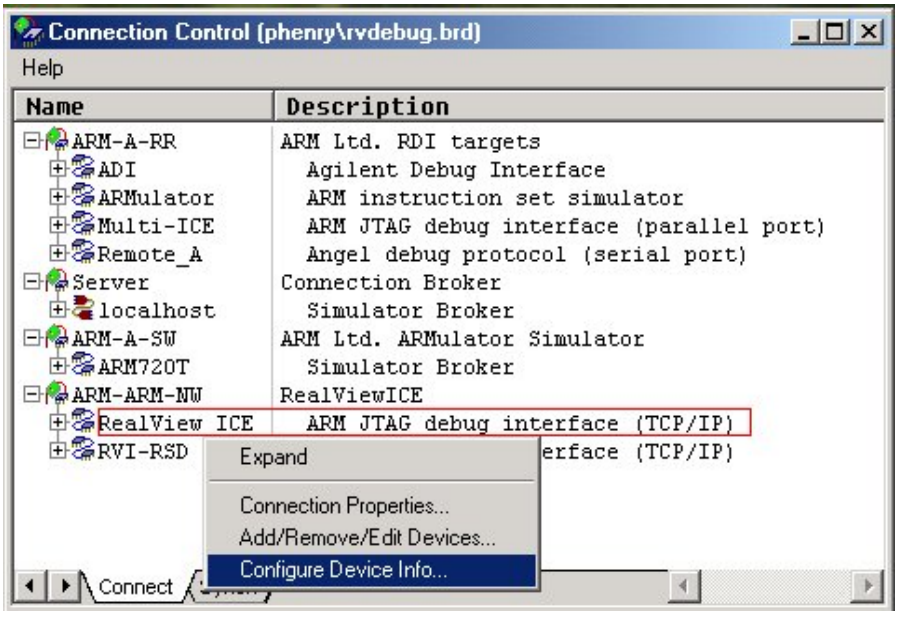
Connect RVI JTAG cable to the Integrator board JTAG header connector. Power up the Integrator board and RealView ICE.

Connect the debugger to the RealView ICE Unit, and configure the ICE connection to the ARM core on the Integrator board by following these steps:

Launch RVD by selecting *Programs* → *ARM* → *RealView Developer Suite 2.2* → *RealView Debugger* from the Windows *Start* menu.

From RVD select *Target* → *Connect To Target* to open the Connection Control window (alt +0).

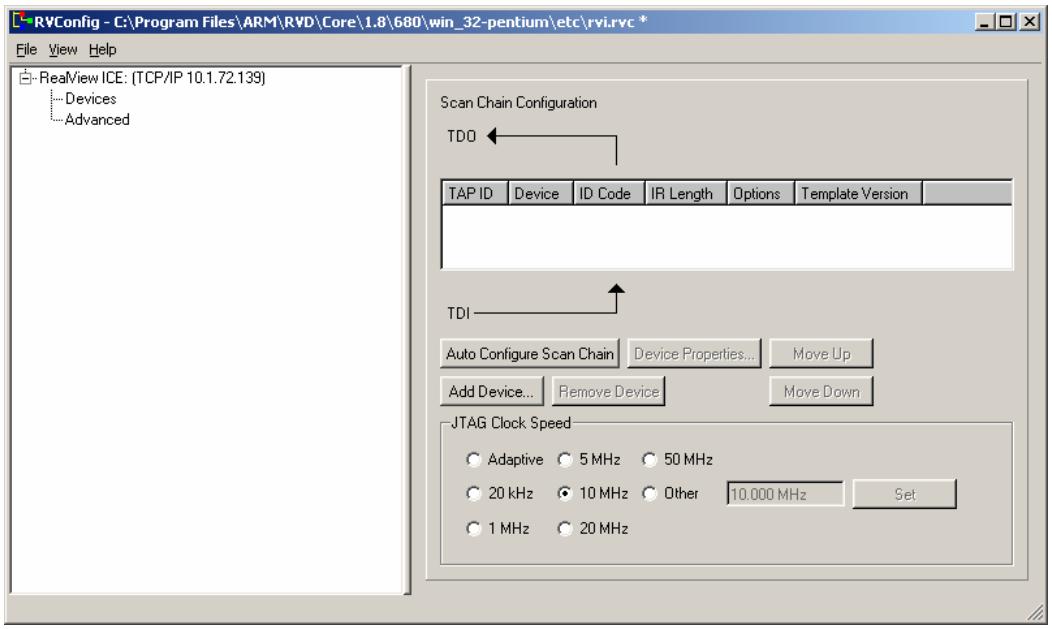
In the Connection Control window, right click on the *RealView ICE* branch of the *ARM-ARM-NW* entry, and select *Configure Device Info* to launch the ICE Configuration Dialogue (RV Config).



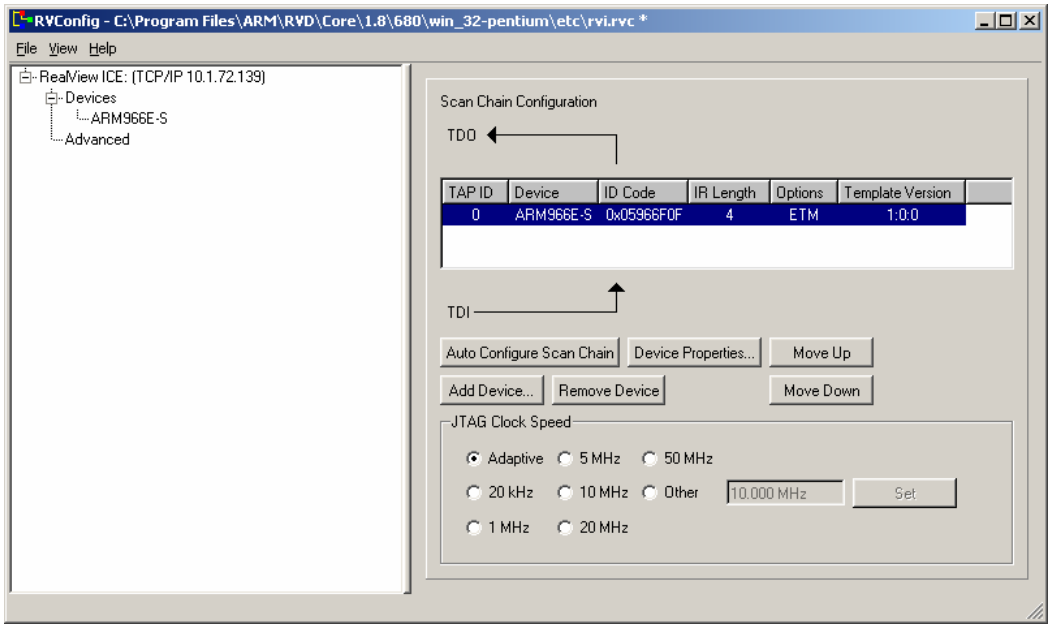
Note: The *'Configure Device Info...'* entry will **not** be shown if you are already connected to a target.

In the *RealView ICE Browser* window, determine which RealView ICE unit that you wish to connect to and click to select that Unit and click *Connect*.

The *Scan Chain Configuration* window is shown as below:



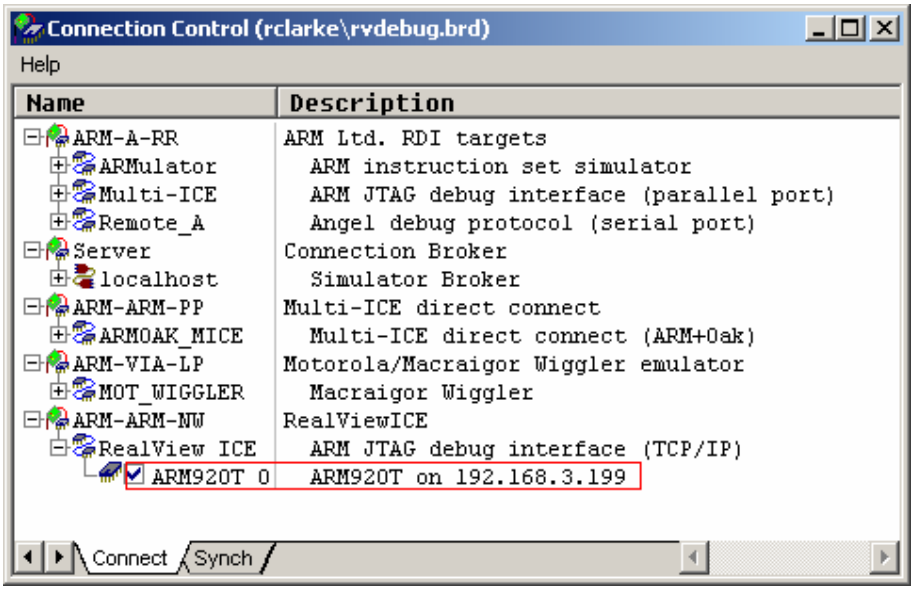
Make sure that the *Devices* item is highlighted under the RealView ICE tree. Click on the *Auto Configure Scan Chain* button to connect the RealView ICE unit to the target hardware.



Verify that the correct core is listed under the Devices entry, and also in the Scan Chain Configuration dialog.

Select *File* → *Save* to save the selected device configuration, and then select *File* → *Exit* to close the RV Config dialog.

In the Debugger *Connection Control* window, expand the *RealView ICE* branch of the *ARM-ARM-NW* entry, and select the ARM920T target by clicking in the checkbox alongside it.

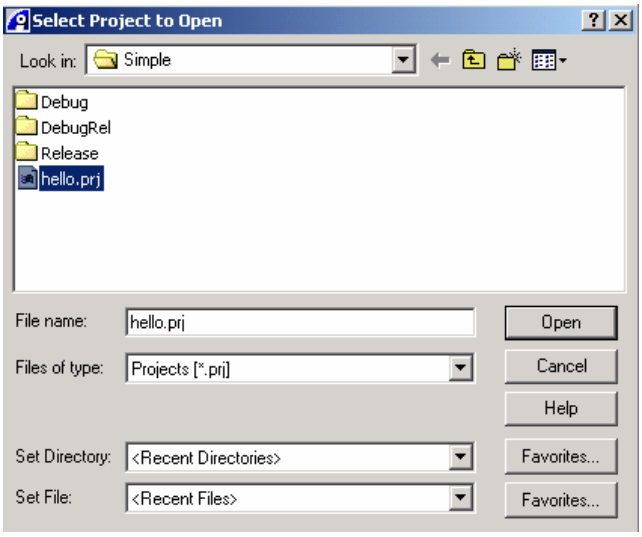


Note: RVD will remember previous connections and these will appear ticked checkboxes. For this tutorial, uncheck any other connections if they exist.

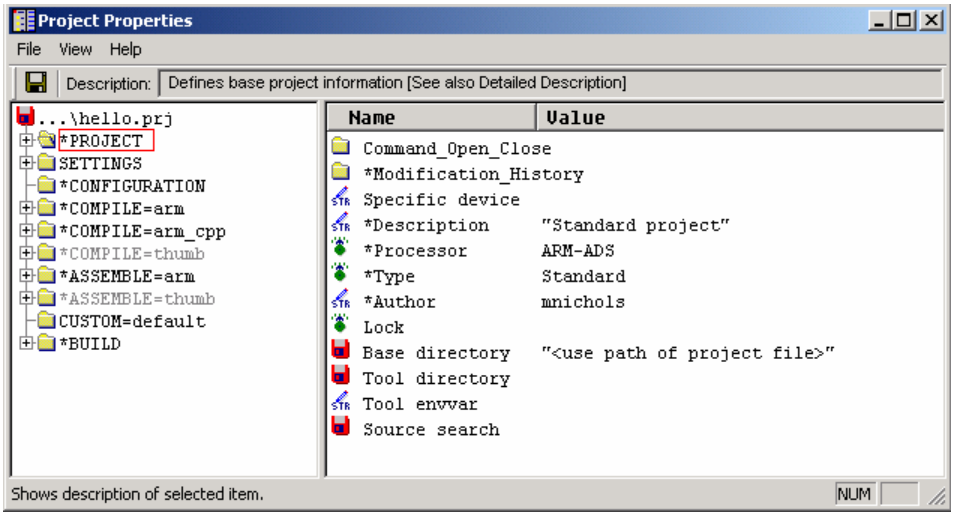
Close the *Connection Control* window and return to the main debugger window.

Session 2 – Simple Hello World Project

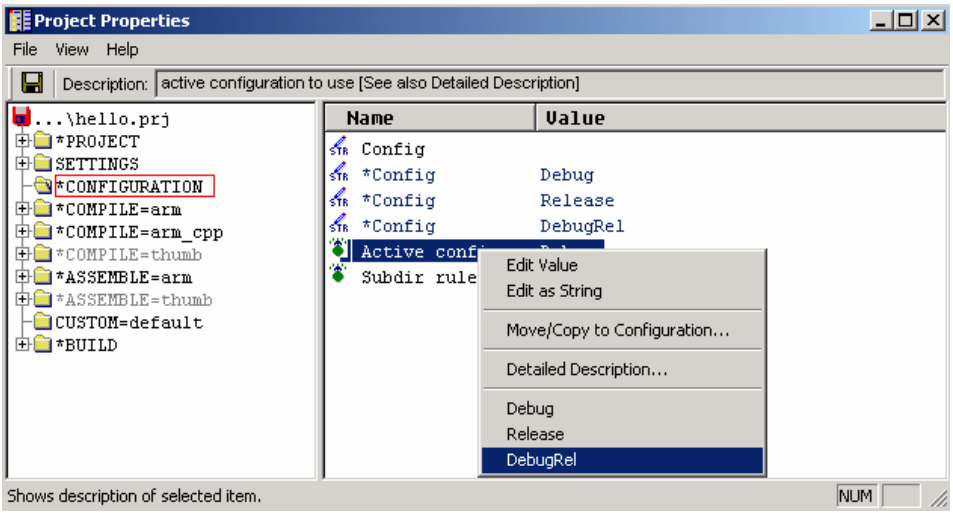
Select *Project→ Open Project* from the RVD main menu. Locate the file "hello.prj" supplied in the **`c:\rvds22_tutorial\rvi\simple`** directory and click *Open*.



Now select *Project→ Project Properties* from the RVD main menu to view the properties for the project "hello.prj".



In the *Configuration* folder of the Project Properties window, change the active configuration to DebugRel by right clicking on the *Active config* entry. This is equivalent to using –g -O1 command line options for the compiler. DebugRel provides an adequate debug view together with good optimization.

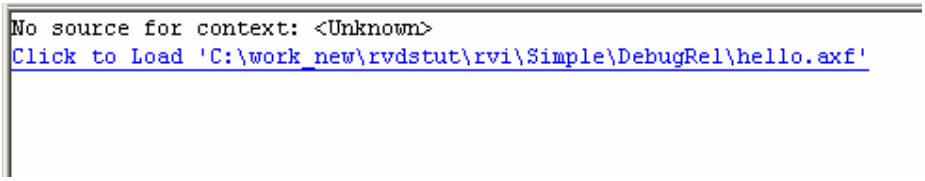


Save the project by selecting *File → Save and Close* from the Project Properties window. This will cause RVD to generate a makefile for your project. The details of this can be seen in the RVD output pane *Build* tab.

Note: If you make any modifications to your project you must save your project, to regenerate the make file, before building it.

Now select *Build→ Rebuild All* from the RVD main menu to build the project. The details of the build process can be seen in the RVD output pane build tab.

Load the “hello.axf” image file into RVD by clicking on the hyperlink in the *Src* tab of the Code pane.



The Code pane shows the image is loaded and the red box indicates the current execution position.

```
#include <stdio.h>

void subroutine (void)
{ printf ("Hello from subroutine\n");
}

int main (void)
{ printf ("Hello from Main\n");
  subroutine();
  printf ("And goodbye\n");
  return (0);
}
```

When RealView Debugger first loads an image the File Editor pane contains tabs to allow program execution to be seen:

- the *Src* tab shows the current context, that is the location of the PC at the entry point
- the *Dsm* tab displays disassembled code with intermixed C/C++ source lines and, if available, the location of the PC.

RealView Debugger uses autoscope to show the context at main by default. For this reason it does not set a breakpoint on main. Please refer to the RealView Debugger documentation for more details.

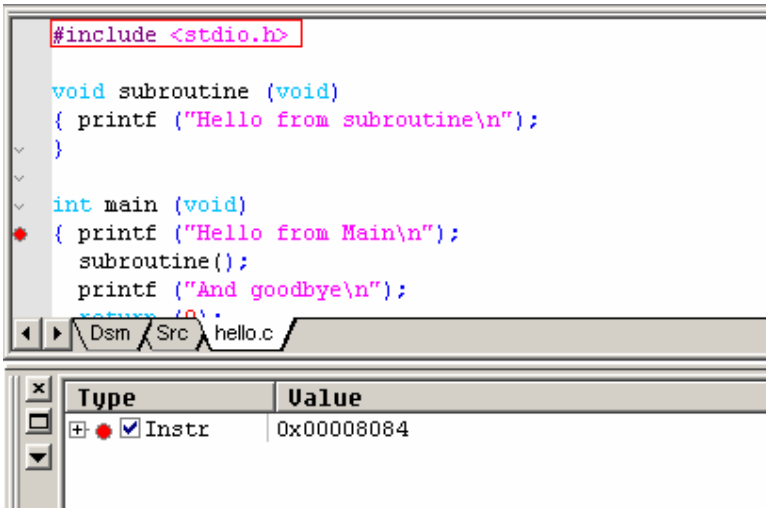
Select *Debug* → *Run* from the menu *(F5)*.

Execution begins. The Output pane at the bottom of the window shows the *StdIO* tab which performs console I/O operations for the current image. The program prints some text to the output window and closes.

Select *File→ Reload Image to Target* from the menu.

RVD will load the image ready for debugging. Again the current execution position is shown at `main()`.

Set a breakpoint on main by double clicking in the grey bar (at the left hand side of the *Code* pane). From the RVD main menu select *View→ Break/Tracepoint* to view information about breakpoints you have created.



Select *Debug→ Run* from the menu (F5).

Execution now halts at `main()` after initialisation of the C library.

Clear the breakpoint by selecting it in the *Break/Tracepoint* pane, right clicking and selecting *Clear* from the menu.

Select *Go* (F5*)* to continue executing the program and finish the example.

Session 3 – Full embedded application

This session uses a more complex embedded example designed to run on the ARM Integrator/AP platform. The sources for this example are in the **`c:\rvds22_tutorial\rvi\embedded`** directory, or **`c:\rvds22_tutorial\rvi_cp\embedded`** if you are using the Integrator/CP board.

Details of the differences between the Integrator/AP and Integrator/CP can be found in the Integrator/CP documentation.

Ensure any currently open projects are closed. From the RVD main menu select *Project→ Close Project*.

Select *Target→ Connect To Target* from the menu to open the *Connection Control* window. Expand the *RealView ICE* branch of the *ARM-ARM-NW* entry, and disconnect from the ARM920T target by unchecking the processor checkbox.

Ensure the Integrator AP Motherboard switches are set to:
Switch S1-1 on & S1-4 on.

Following a reset, these settings cause the boot monitor to run and then poll waiting for an input.

Reset the AP motherboard.

Select *Project→ Open Project* from the RVD main menu. Locate the file “ledflash.prj” supplied in the appropriate directory and click *Open*.

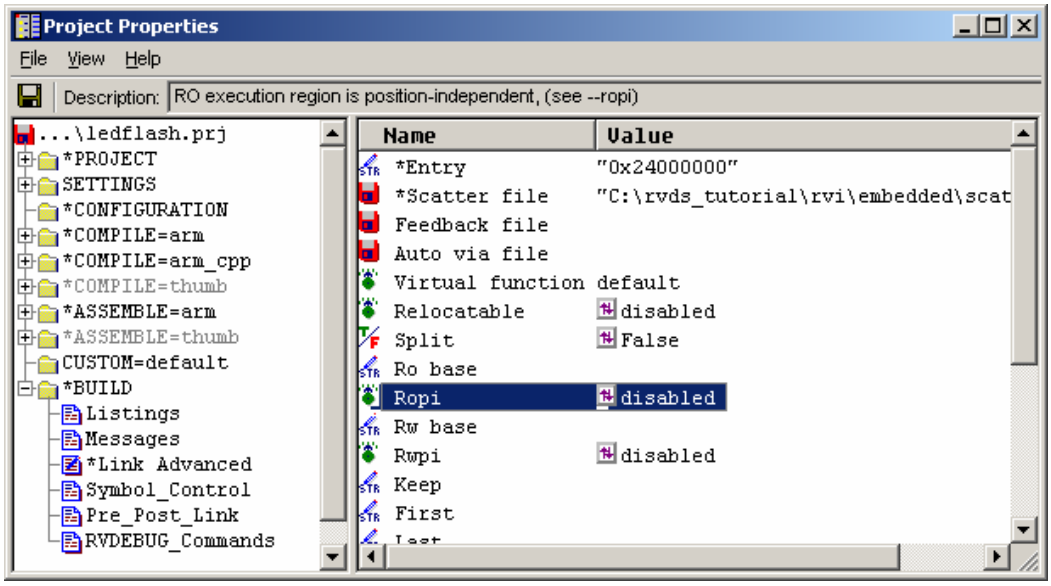
From the RVD main menu use *File→ Open* to view “scatter.txt”.

The scatter-loading file “scatter.txt” shows:

- One Load Region		Start 0x24000000, size 0x4000000
- Two Execution Regions	Flash	Start 0x24000000, size 0x4000000
	Ram	Start 0x00000000, size 0x003FFFF

Add the scatter file to the project. In the *Project Properties* window navigate to the *Build* folder, link advanced entry. Right click on the scatter file entry and select *edit as filename* to reference the scatter.txt file in the embedded directory.

In the same project folder, set the entry point for the image to 0x24000000



Select *File → Save and* Close from the menu to save the project to regenerate the make file.

Now select *Build→ Rebuild All* from the RVD main menu to build the project "ledflash.prj". An image file "ledflash .axf" is generated in the DebugRel directory of the project.

The image we have built is designed to be downloaded to flash on the Integrator board file.

In order to program flash, RVD temporarily downloads a flash programming routine into RAM on your target. This routine is then executed by the target processor.

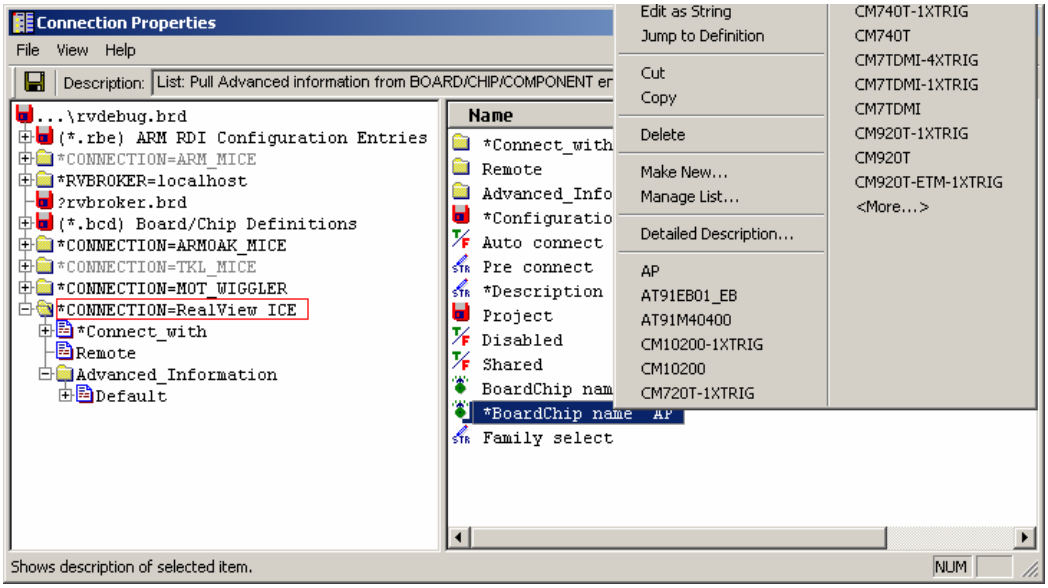
This process is invoked automatically when you attempt to download an image into an area of flash on the board. In order for RVD to know that Flash memory exists at the specified load address it must have access to an appropriate "bcd" (Board Chip Definition) file for your target. A prebuilt bcd file is provided for the Integrator platform.

We need to add the bcd file for the Integrator AP to the current connection.

Open the Connection Properties window by selecting Target→ *Connection Properties* from the RVD main menu.

In the Connection Properties window, select the *RealView ICE* connection to access the RealView ICE connection properties.

Right click on the *BoardChip name* property in the right hand pane and select *AP* from the context menu:



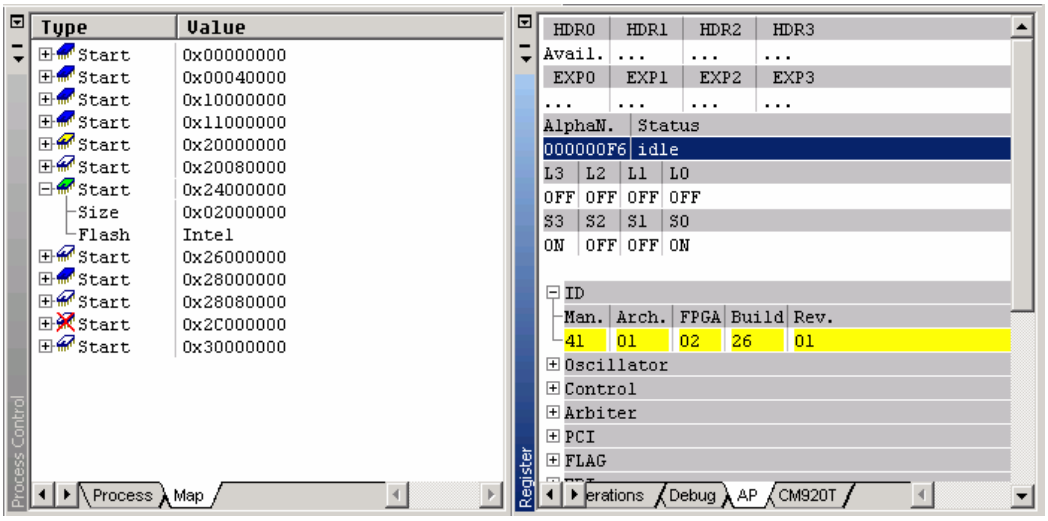
Next, perform the same operation again, but this time select *CM920T* from the context menu. If it is not present select the <*More...*> entry and locate *CM920T* from the resulting list selection dialog.

Note: This example was written using an Integrator/AP with a *CM920T* core module. You may have to select alternative entries according to your target.

The *Connection Properties* window should now reflect the updated settings. Select *File→ Save and Close* to save the changes and close the *Connection Properties* window

Exit and restart the RealView Debugger. When the RealView Debugger has restarted, open the Connection Control window, and reconnect to the target by checking the processor checkbox.

Using *View→Registers* and *View → Process Control* from the RVD main menu, ensure you have pane views selected which display the *Register* pane – *AP* tab and the Process Control pane – *Map* tab.



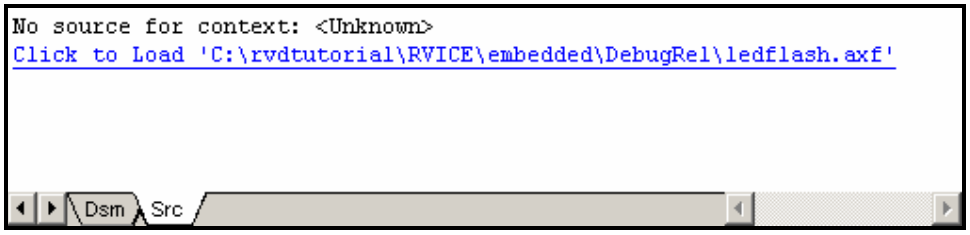
You will see that extra information specific to the target is displayed in those two panes.

In the *Process Control – Map* view, the memory map of the ARM Integrator is displayed. Note the green block at address 0x24000000 which represents the AP boards application flash.

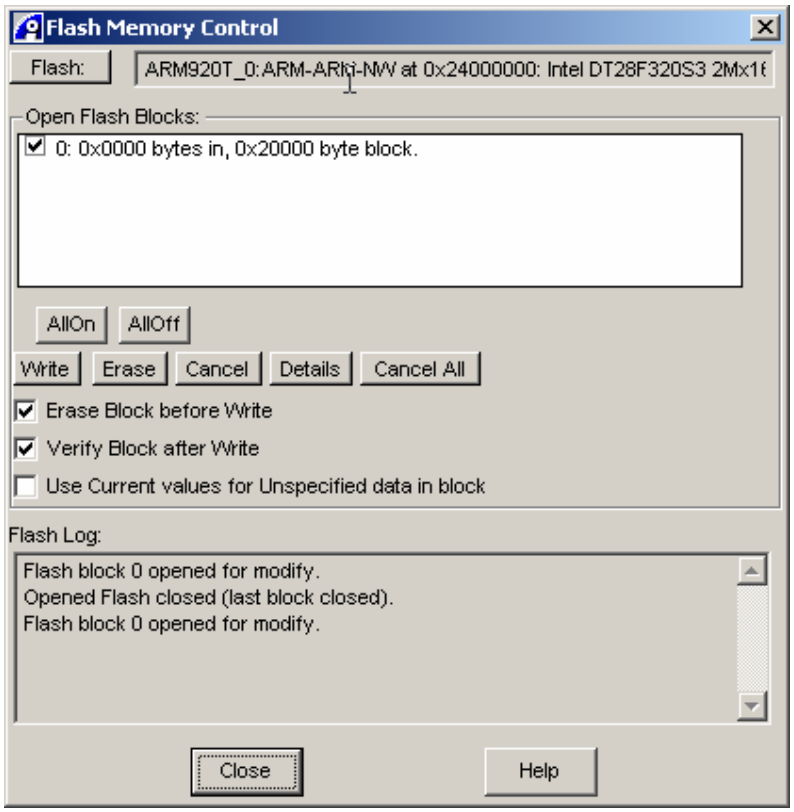
In the *Register – AP* view, enumerated representations of the AP hardware appear. For example: the status of the four switches S0 to S3 and the LED's L0 to L3 are shown.

Right click on *L0* and select *ON*. The green LED on the Integrator AP mother board should illuminate.

Click on the hyperlink in the code window, as shown below, to load image to the target



The Flash Memory Control window appears.



Click *Write* to load the image into flash. The results of the flash programming sequence are show in the Flash log window.

When RVD has finished programming the image into flash memory on the target board, click *Close*.

Select *Debug→Set PC to Entry Point* from the RVD main menu.

Once the image is loaded the code pane shows the contents of the file “init.s” at the image entry point.

```
        ENTRY

; --- Perform ROM/RAM remapping, if required
    IF :DEF: ROM_RAM_REMAP

; On reset, an aliased copy of ROM is at 0x0.
; Continue execution from 'real' ROM rather than aliased copy
        LDR     pc, =Instruct 2

; Remap by setting Remap bit of the CM_ctl register
        LDR     r1, =CM_ctl_reg
        LDR     r0, [r1]
        ORR     r0, r0, #Remap_bit
        STR     r0, [r1]

; RAM is now at 0x0.
; The exception vectors (in vectors.s) must be copied from ROM to the RAM
; The copying is done later by the C library code inside __main
```

Dsm | Src | init.s | vectors.s+ | main.c+

Select *Debug→Run* from the menu *(F5)*.

The LEDs should flash on the Integrator board.

Disconnect from the target in the Connection Control window and close RVD.

The image should now be programmed into flash and can be run as a standalone embedded image.

Ensure the Integrator AP Motherboard switches S1-1 is off.

These settings will cause the board to boot directly from the application flash.

Power cycle the AP motherboard.

The LEDs on the Integrator board should start to flash again.

Debugging from reset

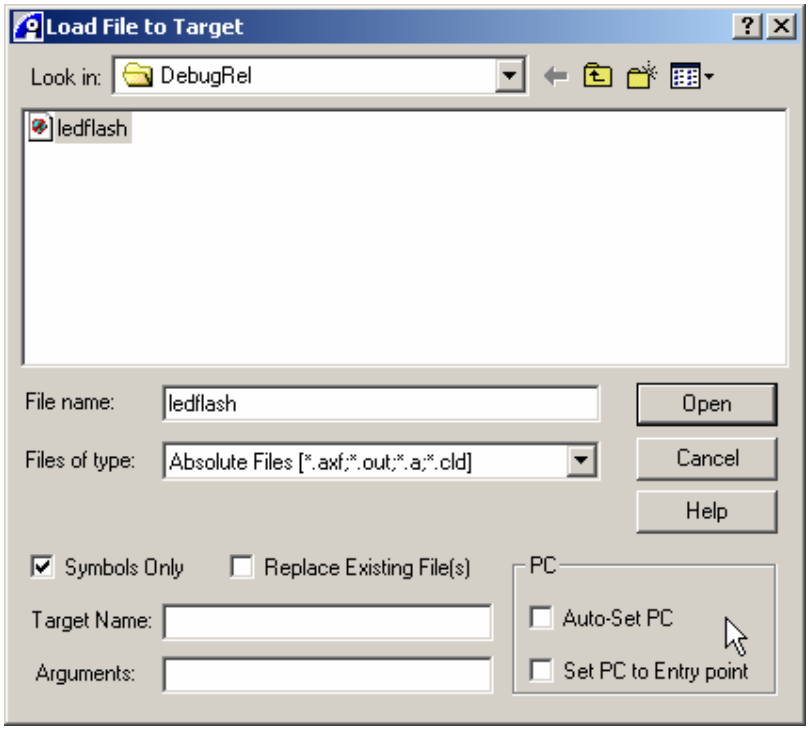
With the Integrator running (i.e.LEDs flashing).

Launch RVD, and use the Connection Control window to reconnect RealView ICE to your target.

The LEDs stop flashing as execution is halted by the debugger.

Select *Target→ Load Image...* from the RVD main menu.

The dialog box contains controls to configure the way the image is loaded.

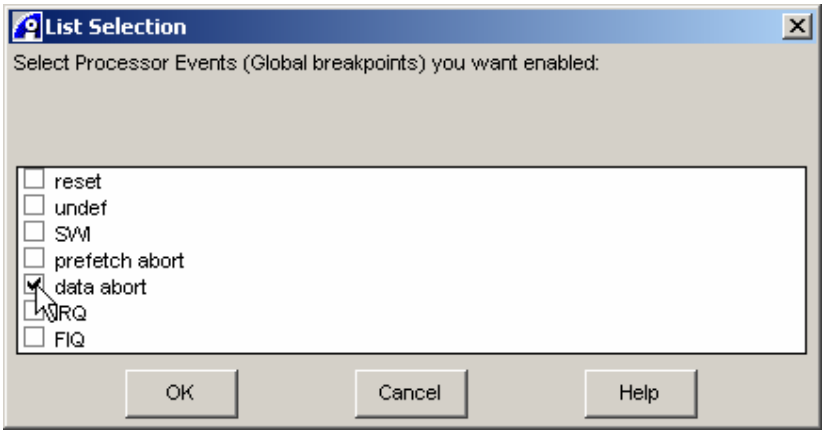


Select '*Symbols Only'* and click *Open*.

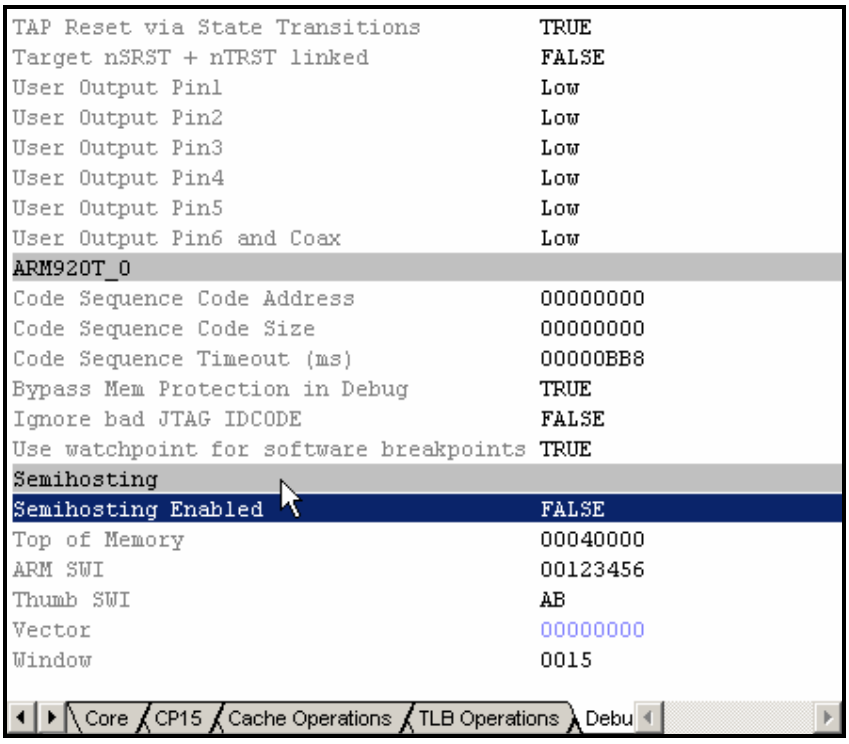
Important: You must ensure the *'Replace Existing Files'*, *'Auto-Set PC'* and *'Set PC to Entry Point'* checkboxes are cleared. This is not done by default.

In order to ensure there are sufficient hardware watchpoint units available to set breakpoints and single step code in ROM, we need to clear the vector catch and semihosting options in the debugger (typically this is only required for ARM7 based targets).

To clear vector catch in RVD, select *Debug→Processor Exceptions*. In the List Selection dialog, clear all of the vector catch entries.



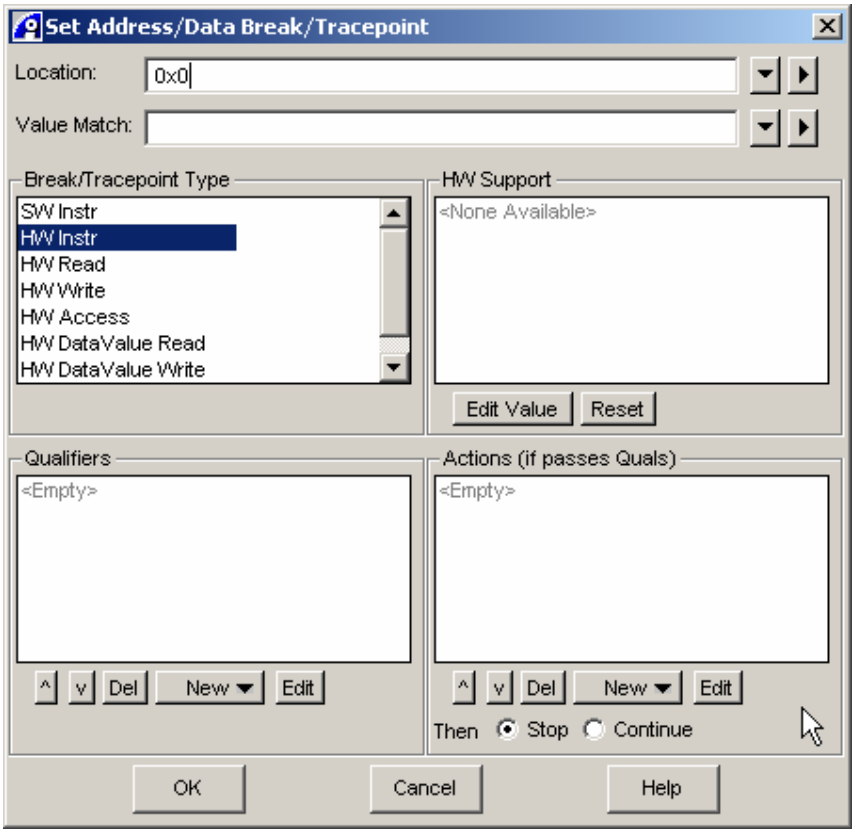
To disable semihosting in RVD, select *View→Registers* and navigate to the *Debug* tab. Right click on the current entry for '*semihosting_enabled'* and select *'FALSE'*



These changes will only remain for this debug session and while connected to the current target. Permanent changes, can be made by modifying the properties of a particular connection.

We will now set a hardware breakpoint at address 0x0 to allow us to debug the target from reset.

From the RVD main menu select *Debug→ Breakpoints→ Set/Edit Breakpoint…* to open the Set Address Break window.

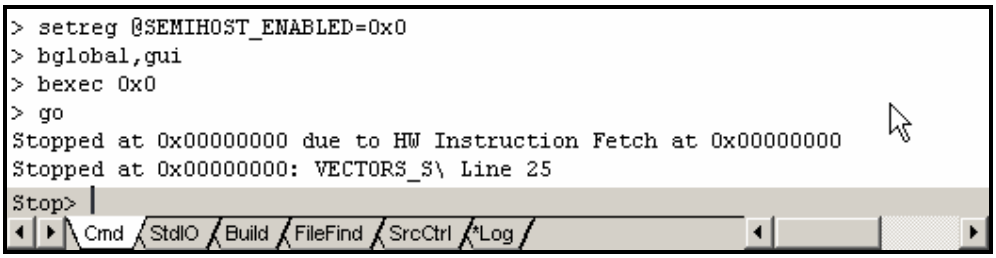


In the *Location* field, type 0x0 to set a breakpoint at this address. Depending on your version of RealView ICE firmware, you may need to select '*HW Instr*' to set a hardware breakpoint. Click *OK*.

The new breakpoint should be visible in the Breakpoints pane view.

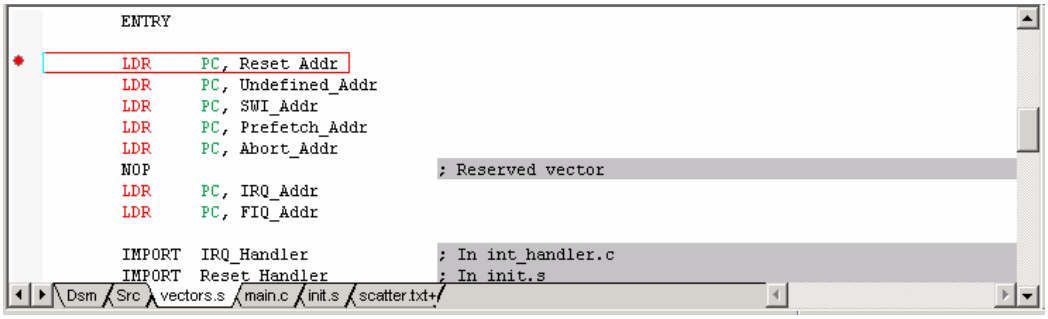
Select *Debug→Run* from the menu *(F5)*.
Reset the Integrator using the green reset button on the Integrator motherboard (Note: only reset the board while the target is running).

In the RealView Debugger output console, the following warning appears:



Depending on your version of RealView ICE firmware, you may also get the warning message “nSRST has been activated”.

The Code window shows execution halted at the breakpoint and displays the source from the file vectors.s.



The information that RVD gets from the debug symbols show that vectors.s is at 0x0 and the associated source code is shown when you click the *Src* tab.

Click on the *Dsm* tab to view the disassembly.

```
>>> VECTORS_S\#25          LDR     PC, Reset_Addr
    init:
    RW:00000000 E59FF02C   LDR     pc,0x34                  <VECTORS S\#32>
>>> VECTORS_S\#26          LDR     PC, Undefined_Addr
    RW:00000004 E3A012C1   MOV     r1,#0x1000000c
>>> VECTORS_S\#27          LDR     PC, SWI_Addr
    RW:00000008 E5910000   LDR     r0,[r1,#0]
>>> VECTORS_S\#28          LDR     PC, Prefetch_Addr
    RW:0000000C E3800004   ORR     r0,r0,#4
>>> VECTORS_S\#29          LDR     PC, Abort_Addr
    RW:00000010 E5810000   STR     r0,[r1,#0]
>>> VECTORS_S\#30          NOP                              ; Reserved vector
    RW:00000014 E321F0D3   MSR     CPSR_c,#0xd3
>>> VECTORS_S\#31          LDR     PC, IRQ_Addr
    RW:00000018 E59FD018   LDR     r13,IRQ_Addr             <0x38>
>>> VECTORS_S\#32          LDR     PC, FIQ_Addr
    RW:0000001C E321F0D2   MSR     CPSR_c,#0xd2
    Reset_Addr:
    RW:00000020 E59FD014   <Data> 0x14 0xD0 0x9F 0xE5
```

Note that the actual disassembly (shown in black) does not match the source. The disassembly shows what is actually in memory at address 0x0.

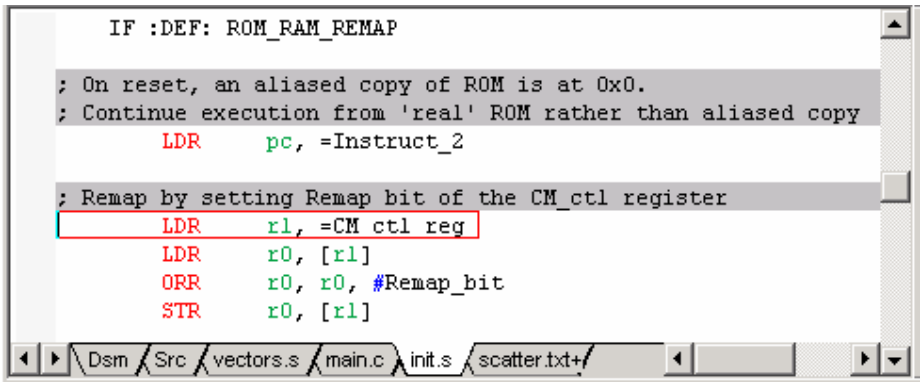
As an alias of flash has temporarily been mapped to 0x0 , the disassembly shows the code from the file “init.s”. The first instruction in this code is a branch to the label “Instruct_2”.

This instruction disassembles to a LDR from a PC relative literal pool. The contents of this literal pool (at address 0x34) are shown = 0x24000004. This means that the next instruction is executed we will jump from the alias to continue execution from actual flash at address 0x24000004.

Single step (F10) to execute this instruction.

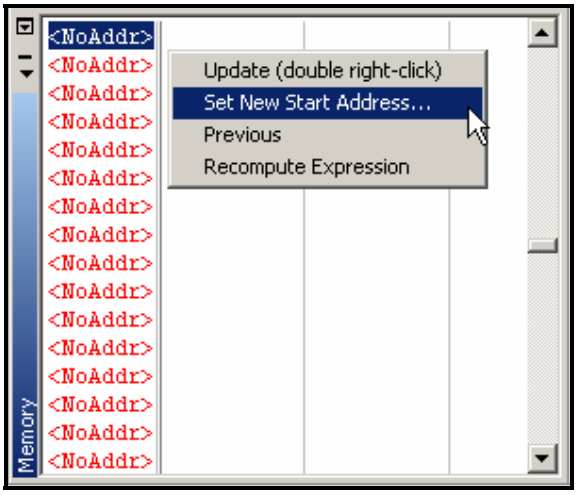
Following execution of the LDR instruction the Dsm pane now shows the current address as 0x24000004.

Click on the *Src* tab. RVD displays the source for init.s

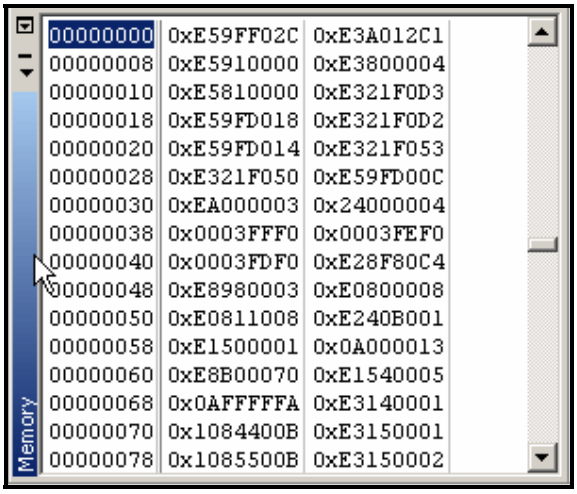


Single step (F10) until the STR instruction is highlighted by the red box.

Open a memory pane view. Right click inside the address column of the pane and select *'Set New Start Address'*. Set the address start value = 0x0.



The memory window should look like the one below.



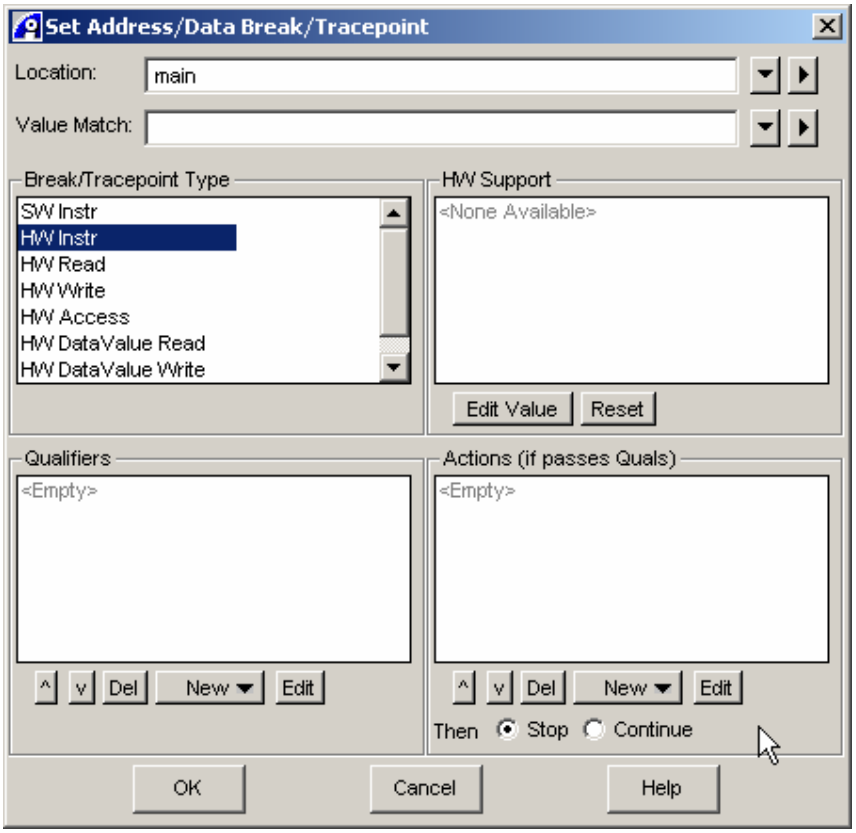
Single step the ‘STR’ instruction. The whole memory display changes, indicating the memory at this location has changed.

Remap has now taken place and the memory shown at 0x0 is now RAM.

Remove the existing breakpoint on address 0x0 by double clicking on the entry in the Break/Tracepoint pane.

Set a new hardware breakpoint on ‘main’ by selecting *Debug→Breakpoint→Set/Edit Breakpoint…* from the main menu.

Type *‘main’* for the location and click *OK*.



Select *Debug→Run* from the menu *(F5)*.

RVD will now run through the C library initialisation code and halt at the entry to your C code. You can now single step and debug the C source.

Appendix – Integrator AP motherboard switch Settings

S[1]-1 OFF Boot from Application flash. Flash at 0x24000000 is aliased to 0x0
Your application code must remap to put RAM back at 0x0.

S[1]-1 ON Boot from Boot ROM. Boot monitor remaps RAM to 0x0 for you.
If S[1]-4 ON - wait polling
If S[1]-4 OFF - jump to application flash