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 \rightarrow Programs $\rightarrow ARM \rightarrow RealView Developer Suite 2.2 <math>\rightarrow$ 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.

ARM Installation Wizard		
	C:\Program Files\ARM	
\leq		Browse
	It is recommended that you accept the direct suggested, since this is where your other AF software has been installed. If you change the directory, this may affect interworking betw different ARM products.	tory choice RM his zeen
InstallShield		
	< Back Next >	Cancel

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.
- Select *RV1→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.

🖋 Configure new Real¥iew ICE device 🔗 🔀			
-Settings			
🔽 DHCP			
Host Name	NetworkRVI		
IP Address	0.0.0.0		
Default Gateway	0.0.0.0		
Subnet Mask	0.0.0.0		
Ethernet Address	00:02:F7:00:05:32		
Ethernet Type	Auto-Detect		
Identify	Configure Help Exit		

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.

💐 Configure new F	RealView ICE device	? ×
Settings		
DHCP		
Host Name	LocalRVI	
IP Address	1.2.3.5	
Default Gateway	0.0.0	
Subnet Mask	255.255.255.0	
Ethernet Address	00:02:F7:00:05:32	
Ethernet Type	Auto-Detect	•
Identify	Configure Help	Exit

4. Close the RVI Config IP window.

Files for the exercises

Two sets of files are include	d 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.

θ

_____` ₩₩₩

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):

0	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:

From RVD select
$$Target \rightarrow Connect To Target$$
 to open the Connection
Control window (alt +0).

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

phenry\rvdebug.brd)		- O ×
Description		
Image: Second perform Image: ARM-A-RR ARM Ltd. RDI targets Image: ADI Agilent Debug Interface Image: ARMulator ARM instruction set simulator Image: ARMulator ARM instruction set simulator Image: ARMulator ARM instruction set simulator Image: ARM JTAG debug interface (parallel port) Image: ARM JTAG debug protocol (serial port) Image: ARM-A-SW Image: ARM-A-SW		port)
ARM JTAG debug in	nterface (TCP/IP)	
pand nnection Properties d/Remove/Edit Devices pfigure Device Info	erface (TCP/IP)	
	phenry/rvdebug.brd) Description ARM Ltd. RDI target Agilent Debug Int ARM instruction is ARM JTAG debug in Angel debug prote Connection Broker Simulator Broker ARM Ltd. ARMulator Simulator Broker RealViewICE ARM JTAG debug in pand nnection Properties d/Remove/Edit Devices nfigure Device Info	phenry\rvdebug.brd) Description ARM Ltd. RDI targets Agilent Debug Interface ARM instruction set simulator ARM JTAG debug interface (parallel Angel debug protocol (serial port) Connection Broker Simulator Broker ARM Ltd. ARMulator Simulator Simulator Broker RealViewICE ARM JTAG debug interface (TCP/IP) pand erface (TCP/IP) nnection Properties d/Remove/Edit Devices nfigure Device Info

0	Note: The 'Configure Device Info' entry will not be shown if you are already connected to a target.
	In the <i>RealView ICE Browser</i> window, determine which RealView ICE unit that you wish to connect to and click to select that Unit and click <i>Connect</i> .

The Scan Chain Configuration window is shown as below:

RVConfig - C:\Program Files\ARM\RVD\Core\1.8\680	\\win_32-pentium\etc\rvi.rvc *	
<u>F</u> ile <u>V</u> iew <u>H</u> elp		
⊡-ReaWiew ICE: (TCP/IP 10.1.72.139) Devices Advanced	Scan Chain Configuration TDD TAP ID Device ID Code IR Length Options Template Version	_
	TDI	
	JTAG Clock Speed	
	C Adaptive C 5 MHz C 50 MHz C 20 kHz C 10 MHz C 0ther 10.000 MHz Set C 1 MHz C 20 MHz	
1		

Make sure that the <i>Devices</i> item is highlighted under the RealView ICE tree. Click on the <i>Auto Configure Scan Chain</i> button to connect the RealView ICE unit to the target hardware.
<i>6</i>

	win_32-pentium\etc\rvi.rvc *	- 🗆 ×
<u>F</u> ile ⊻iew <u>H</u> elp		
Er RealView [EF] F-RealView ICE: (TCP/IP 10.1.72.139) C-Devices L-ARM966E-S Advanced	Scan Chain Configuration TDD TAP ID Device ID Code IR Length Options Template Version 0 ARM966E-S 0x05966F0F 4 ETM 1:0:0 TDI Auto Configure Scan Chain Device Properties Move Up Add Device Remove Device Move Down JTAG Clock Speed © Adaptive © 5 MHz © 50 MHz © 20 kHz © 10 MHz © 0ther 10.000 MHz Set © 1 MHz © 20 MHz	
		11.

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



θ

Select $File \rightarrow Save$ to save the selected device configuration, and then select $File \rightarrow Exit$ 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.

🚱 Connection Control (r	clarke\rvdebug.brd)
Help	
Name	Description
🗆 🚱 ARM-A-RR	ARM Ltd. RDI targets
🕂 🕾 🖓 ARMulator	ARM instruction set simulator
🖶 🕾 Multi-ICE	ARM JTAG debug interface (parallel port)
🗄 🕾 Remote_A	Angel debug protocol (serial port)
🛛 🗁 🥵 Server	Connection Broker
🗄 🚭 localhost	Simulator Broker
🗆 🗁 🚰 ARM-ARM-PP	Multi-ICE direct connect
🗄 🚟 ARMOAK_MICE	Multi-ICE direct connect (ARM+Oak)
E-RAARM-VIA-LP	Motorola/Macraigor Wiggler emulator
H WOT_WIGGLER	Macraigor Wiggler
E-R ARM-ARM-NW	RealViewICE
RealView ICE	ARM JTAG debug interface (TCP/IP)
-₩ ARM920T 0	ARM920T on 192.168.3.199
Connect Synch	

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



0

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

Session 2 – Simple Hello World Project



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

Select Pro	ject to Open	<u>? ×</u>
Look in: 🔂	Simple 💌 🗲 🛍	💣 🎟 •
Debug DebugRel Release		
File name:	hello.prj	Open
Files of type:	Projects [*.prj]	Cancel
		Help
Set Directory:	<recent directories=""></recent>	Favorites
Set File:	<recent files=""></recent>	Favorites



Now select $Project \rightarrow Project$ 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.

Project Properties			_ 🗆 ×
File View Help			
Description: active configuration to) use [See also Detailed	Description]	
<pre>\hello.prj\hello.prj</pre>	Name STR Config	Value	
<pre>*CONFIGURATION *COMPILE=arm *COMPILE=arm *COMPILE=arm</pre>	STR *Config STR *Config STR *Config	Debug Release DebugRel	
COMPILE=thumb	 Active conf Subdir rule 	Edit Value Edit as String	
⊢ <mark>⊟</mark> CUSTOM=default ⊕		Move/Copy to Configuration Detailed Description	_
		Debug Release	
		DebugRel	
Shows description of selected item.			NUM ///

Save the project by selecting $File \rightarrow Save$ 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 $Build \rightarrow Rebuild All$ 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.

لأستنس
###

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.



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.

ß

<u>.</u>

Select *File* \rightarrow *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 $View \rightarrow Break/Tracepoint$ to view information about breakpoints you have created.

#include <st< th=""><th>tdio.h></th></st<>	tdio.h>
void subrout	tine (void)
{ printf (")	<pre>fello from subroutine\n");</pre>
~ }	
int main (m	aid)
/ nrintf ("	Hello from Main\n"):
subroutine	±();
printf ("	and goodbye\n"):
Contrary (Q)	·
Image: A state of the state	hello.c
🔺 Type	Value
📃 🕀 🔶 🗹 Insti	0x00008084

Select $Debug \rightarrow Run$ from the menu (F5).

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

ند الم	Clear the breakpoint by selecting it in the <i>Break/Tracepoint</i> pane, right clicking and selecting <i>Clear</i> from the menu.
	Select <i>Go</i> (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 $Project \rightarrow Close \ Project$.

Select $Target \rightarrow Connect To Target$ 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.

A Reset the AP motherboard.



Select $Project \rightarrow Open \ Project$ 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* \rightarrow *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 0x0000000, 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. <u>___</u> In the same project folder, set the entry point for the image to *** 0x24000000 _ 🗆 🗵 Project Properties <u>File View H</u>elp Description: RO execution region is position-independent, (see --ropi) 🚽...\ledflash.prj ٠ ٠ Name Value 🕂 💼 * PROJECT 🚮 *Entry "0x24000000" 🗄 🚞 SETTINGS . 🚽 *Scatter file 🖬 Feedback file "C:\rvds_tutorial\rvi\embedded\scat - 🚞 * CONFIGURATION 🕂 💼 *COMPILE=arm 📕 Auto via file 🗄 🚞 *COMPILE=arm_cpp 🖇 Virtual function default 🕂 🧰 *COMPILE=thumb 🕂 🚞 *ASSEMBLE=arm 🍯 Relocatable 🔁 disabled 🕂 💼 *ASSEMBLE=thumb 🏹 Split 🔁 False - CUSTOM=default 👫 Ro base 🗄 🚞 *BUILD 🔁 disabled 🍋 Ropi 🔁 Listings 🚮 Rw base 🖹 Messages 🍍 Rwpi 🔁 disabled -🛃 *Link Advanced 👫 Keep -🔁 Symbol_Control 🚮 First 🔁 Pre_Post_Link 🖹 RVDEBUG_Commands Taet • ۲

د.....ک ایکیکی Select $File \rightarrow Save and$ Close from the menu to save the project to regenerate the make file.

Now select $Build \rightarrow Rebuild All$ 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 \rightarrow 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 core module. You may have to select alternative entries according to your target.



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

Using $View \rightarrow Registers$ and $View \rightarrow Process \ Control$ 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.

▼	Туре	Value		E H	DRO	HDR.	1 HDF	2 н	DR3	_
Ţ	🕀 🛹 Start	0x00000000		- Av	ail.					
	🕀 🌌 Start	0x00040000		E	XPO	EXP:	l EXI	2 E	KP3	
	🕀 📶 Start	0x10000000		· · ·						
	🕀 📶 Start	0x11000000		Al	phaN	. Sta	atus			
	🗄 🛃 Start	0x20000000		00	0000	76 id.	le			
	🗄 📶 Start	0x20080000		L3	L2	Ll	LO			
	🖃 🌱 Start	0x24000000		OF	F OF	OFF	OFF			
	-Size	0x02000000		53	\$2	S1	S0			
	- Flash	Intel		ON	OF	OFF	ON			
	⊞∰ Start	0x26000000								
	⊞ M Start	0x28000000			TD					
	🗄 🎬 Start	0x28080000			Man	àrch	FPGA	Build	Dev	
	H Start	0x2C000000			<u>41</u>		02	26	01	
	⊞# Start	0x30000000		F	AI Naci	lletor	- <mark>02</mark>	20	01	
					Cont:	col	L			
ē					oone. Arbii	.01				
l ti					ALDI	Jer				
U S				<u>ت</u> اھ	FUL					
ies:					r LAG			_		
Pro	I ► Process	Map /	▼ ►	ě 🖣	▶ era	tions /	(Debug)		:м920т 🖊	 -

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.



	Click on the hyperlink in the code window, as shown below, to load				
0(1999)	image to the target				
No sour	ce for context: <unknown></unknown>				
Click t	<pre>co Load 'C:\rvdtutorial\RVICE\embedded\DebugRel\ledflash.axf'</pre>				

The Flash Memory Control window appears.

Image: A Destruction A Des

Plash Memory Control
Flash: ARM920T_0:ARM-ARKi-NW at 0x24000000: Intel DT28F320S3 2Mx18
Open Flash Blocks:
✓ 0: 0x0000 bytes in, 0x20000 byte block.
Frees Black before 10/20
Verity Block after Write
Use Current values for Unspecified data in block
Flash Log:
Flash block 0 opened for modify. Opened Flash closed (last block closed). Flash block 0 opened for modify.
Close Help

شہ

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*.

د
####

Select $Debug \rightarrow Set PC$ 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.



Select *Debug* \rightarrow *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).

لیسیم	
퐬蜡	

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.



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

🔗 Load File to	o Target	<u>? ×</u>
Look in: 🔂	DebugRel 🔽 🗲 🛍 🕻	* 🎟 •
(@) ledflash		
File name:	ledflash	Open
Files of type:	Absolute Files [*.axf;*.out;*.a;*.cld]	Cancel
		Help
Symbols O	nly 🔲 Replace Existing File(s) ү РС	
Target Name:	Auto-Se	
Arguments:	Set PC t	o Entry point

Select	<i>Symbols</i>	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.

θ

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 $Debug \rightarrow Processor Exceptions$. In the List Selection dialog, clear all of the vector catch entries.

List Selection	x
Select Processor Events (Global breakpoints) you want enabled:	
u i prefetch abort I III data abort	
LURQ	
OK Cancel Help	



å

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

TAP Reset via State Transitions	TRUE		
Target nSRST + nTRST linked	FALSE		
User Output Pinl	Low		
User Output Pin2	Low		
User Output Pin3	Low		
User Output Pin4	Low		
User Output Pin5	Low		
User Output Pin6 and Coax	Low		
ARM920T_0			
Code Sequence Code Address	0000000		
Code Sequence Code Size	0000000		
Code Sequence Timeout (ms)	00000BB8		
Bypass Mem Protection in Debug	TRUE		
Ignore bad JTAG IDCODE	FALSE		
Use watchpoint for software breakpoints	TRUE		
Semihosting			
Semihosting Enabled 🔨	FALSE		
Top of Memory	00040000		
ARM SUI	00123456		
Thumb SWI	AB		
Vector	0000000		
Window	0015		
✓ ► Core CP15 Cache Operations TLB Operations Debu			

0

మి

.

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 $Debug \rightarrow Breakpoints \rightarrow$ Set/Edit Breakpoint... to open the Set Address Break window.

Set Address/Data Break/Tracepoin	t 🔀
Location: 0x0	- F
Value Match:	••
Break/Tracepoint Type SW Instr HW Instr HW Read HW Write HW Access HW DataValue Read HW DataValue Write	HW Support <none available=""> Edit Value Reset</none>
Qualifiers	Actions (if passes Quals)
<empty></empty>	<empty></empty>
^ v Del New ▼ Edit	^ v Del New ▼ Edit Then
ОК Са	ncel Help

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.

<u>.</u>

#####

≣↓

0

Select $Debug \rightarrow Run$ 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:

> setreg @SEMIHOST_ENABLED=0x0	
> bglobal,gui	
> bexec 0x0	
> go	
Stopped at 0x00000000 due to HW Instruction Fetch at 0x00000000	5
Stopped at 0x00000000: VECTORS_S\ Line 25	
Stop>	
Crnd / StdlO / Build / FileFind / SrcCtrl /*Log /	•

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.

	ENTRY		
	IDP	DC Deset Addr	
1	LDR	PC, Undefined Addr	
	LDR	PC, SWI_Addr	
	LDR	PC, Prefetch_Addr	
	LDR	PC, Abort_Addr	
	NOP	; Re	eserved vector
	LDR	PC, IRQ_Addr	
	LDR	PC, FIQ_Addr	
	IMPORT	IRQ_Handler ; In	n int_handler.c
	IMPORT	Rese <u>t Handler ; In</u>	n init.s
	· \Dsm / Src λ vect	ors.s / main.c / init.s / scatter.txt+/	

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.

د الله	Click on the <i>Dsm</i> tab to view the disassembly.

>>> VECTORS_S\#25	LDR	PC, Reset_Addr	
init:			
►→RW:00000000 E59FF02C	LDR	pc,0x34	<vectors s\#32=""></vectors>
>>> VECTORS_S\#26	LDR	PC, Undefined_Addr	
RW:00000004 E3A012C1	MOV	rl,#0x1000000c	
>>> VECTORS_S\#27	LDR	PC, SWI_Addr	
RW:0000008 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></data>	Ox14 OxDO Ox9F OxE5	

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

IF :DEF: ROM	RAM_REMAP	
; On reset, an al ; Continue execut	liased copy of ROM is at 0x0. tion from 'real' ROM rather than aliased copy	
; Remap by settin	pc, =1nstruct_2	
LDR	rl, =CM ctl reg	
LDR 1	r0, [r1]	
ORR 1	r0, r0, #Remap_bit	
STR 1	r0, [r1]	
 ↓ Dsm / Src / vector 	s.s / main.c / init.s / scatter.txt+/	.

....` *******

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.

▼	00000000	0xE59FF02C	0xE3A012C1	
Ţ	00000008	0xE5910000	0xE3800004	
	00000010	0xE5810000	OxE321F0D3	
	00000018	0xE59FD018	OxE321F0D2	
	00000020	0xE59FD014	0xE321F053	
	00000028	0xE321F050	0xE59FD00C	
	00000030	0xEA000003	0x24000004	
	00000038	0x0003FFF0	0x0003FEF0	
1	00000040	0x0003FDF0	0xE28F80C4	
	00000048	0xE8980003	0xE0800008	
	00000050	0xE0811008	0xE240B001	
	00000058	0xE1500001	0x0A000013	
	00000060	0xE8B00070	0xE1540005	
Σ	00000068	OxOAFFFFFA	0xE3140001	
Тğ.	00000070	0x1084400B	0xE3150001	
Ψ	00000078	0x1085500B	0xE3150002	_



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

	-			
◄	00000000	0xE59FF018	0xE59FF018	
Ţ	00000008	0xE59FF018	0xE59FF018	
	00000010	0xE59FF018	0xE1A00000	
	00000018	0xE59FF018	0xE59FF018	
	00000020	0x24000014	0x00000040	
	00000028	0x00000044	0x00000048	
	00000030	0x0000004C	0x00000000	
	00000038	0x24000120	0x00000050	
	00000040	OxEAFFFFFE	OXEAFFFFFE	
	00000048	OxEAFFFFFE	OXEAFFFFFE	
	00000050	OxEAFFFFFE	0x0000008	
	00000058	0x05010208	0x00000100	
	00000060	0x00000000	0x0000002D	
≥	00000068	0x00000000	0x00000000	
Ê	00000070	0x00000000	0x00000000	
Me				-

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.
د ا	Set a new hardware breakpoint on 'main' by selecting $Debug \rightarrow Breakpoint \rightarrow Set/Edit Breakpoint$ from the main menu.

Type 'main' for the location and click OK.

🙆 Set Address/Data Break/Tracepoin	t 🔀
Location: main	▼ ▶
Value Match:	
Break/Tracepoint Type	HW Support
SW Instr	<none available=""></none>
HW Instr	
HW Read	
HVV Vvrite	
HW/ Access	
HW Data Value Write	
	Edit Value Reset
Qualifiers	Actions (if passes Quals)
<empty></empty>	<empty></empty>
VDelNew ▼Edit	vDelNew ▼Edit
	Then 💿 Stop 🔿 Continue
ОК Са	ncel Help

	Select $Debug \rightarrow Run$ 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