

RVDS 2.2 RealView Debugger & RealView Trace Tutorial



251v02

Introduction

Aim

This guide provides the user with a basic introduction to the ARM RealView Trace unit and some guidance for its initial setup and use with ARM's RealView Debugger.

The tutorial is split into two sections:

Section 1 – Setting up the RealView Trace unit.

Section 2 – Using RealView Trace with RealView Debugger.

Pre-requisites

This guide assumes the user has access to a PC workstation with the following tools installed:

RealView Developer Suite 2.2.

RealView ICE 1.2

Additional information

This tutorial is not designed to provide detailed documentation of RealView Debugger, RealView ICE and RealView Trace. 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*

Section 1: Setting up the RealView Trace unit



For full, detailed information on this topic please refer to chapter 6 of the RealView ICE User Guide.



Requirements include access to a Windows PC with the following:

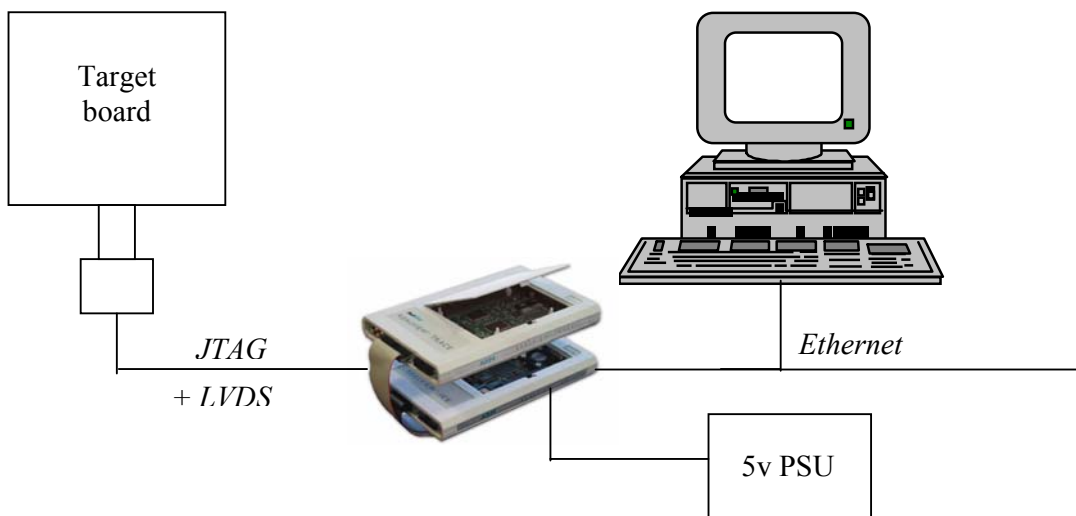
RealView Developer Suite 2.2
RealView ICE 1.2

A suitable target will also be needed, such as an Integrator core module with an ETM connected to the PC via a RealView ICE unit and a RealView Trace TPA.

1.1 - Setting up the hardware

The RealView Trace unit should be securely mounted on the RVI unit as described in section 6.4.2 of the RVI User Guide.

Using the cables provided, connect the RealView Trace unit to the target board, the network and the power supply as follows:



The 'T piece' adapter must be used in order to attach the RealView Trace ribbon cable to the MICTOR connector on the target board.



The RealView Trace unit does not need additional power; it can obtain power directly from the RealView ICE.

Section 2: Using RealView Trace with RVD

This section provides an introduction to using the RealView Trace unit with RealView Debugger to perform trace capture.



For full, detailed information on this topic please refer to the RealView Ice user guide.

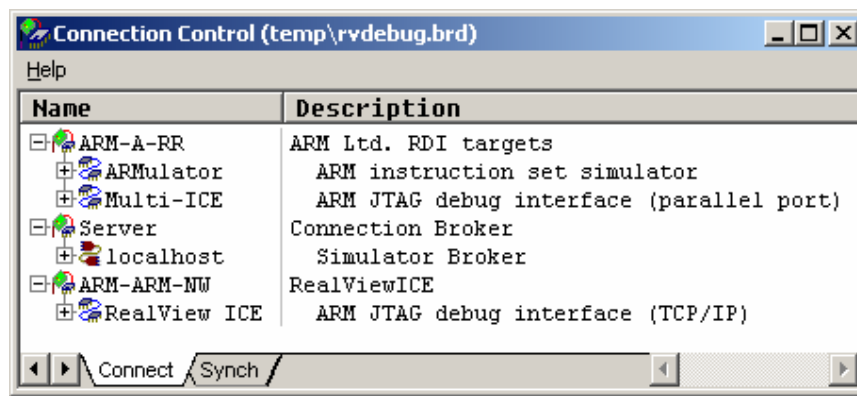
2.1 - Configuring the Target



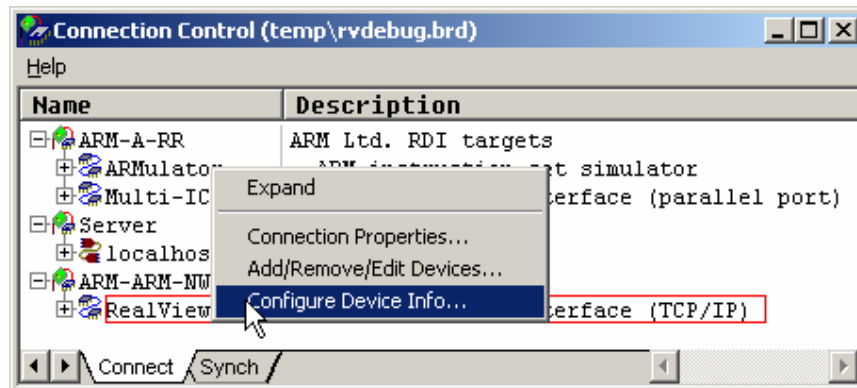
Launch RVD by selecting *Programs → ARM → RealView Developer Suite v2.2 → RealView Debugger v1.8* from the Windows Start menu.



Select *Target → Connect To Target* from the menu. The *Connection Control* window appears:



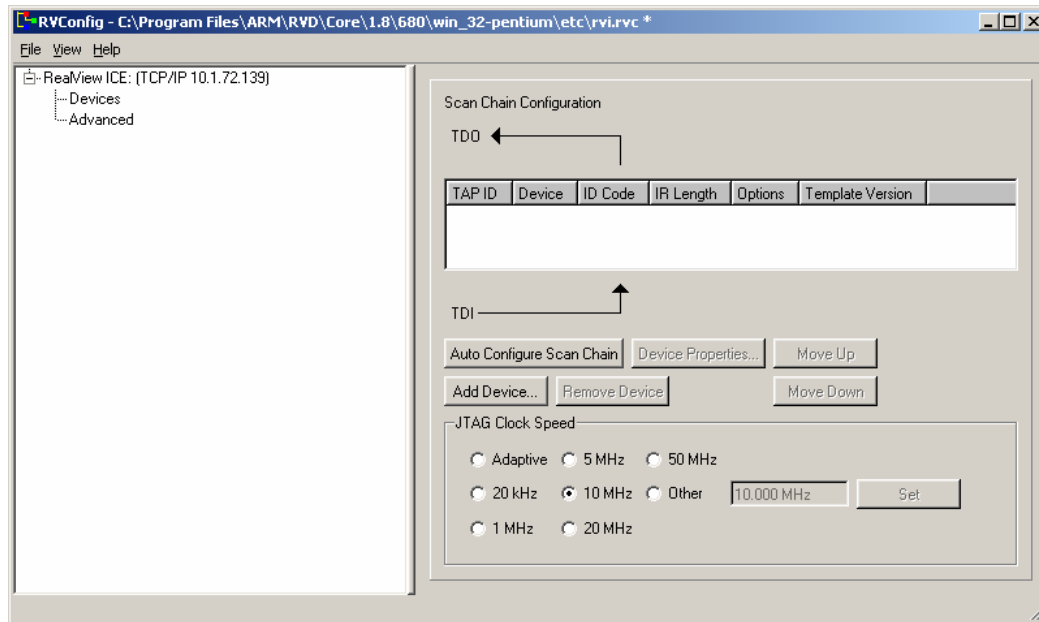
Right click on the *RealView ICE* branch of the *ARM-A-NW* entry in the dialog and select *Configure Device Info...*:



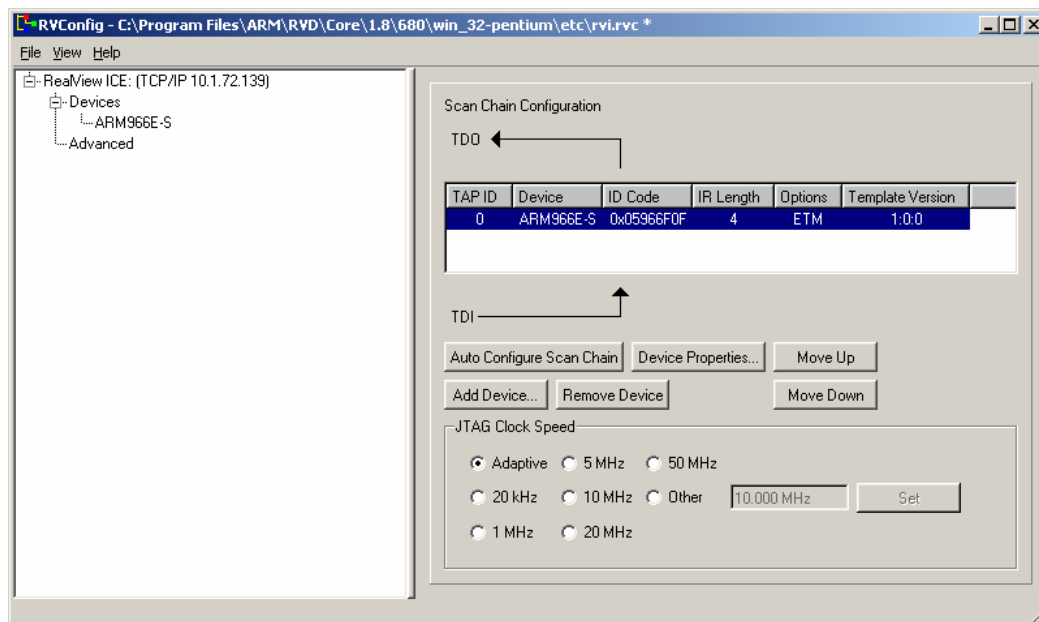


In the *RealView ICE Browser* window, determine which RealView ICE unit that you wish to connect to and click to select that Unit and click *Connect*.

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



Click *Auto Configure Scan Chain* to detect the connected target

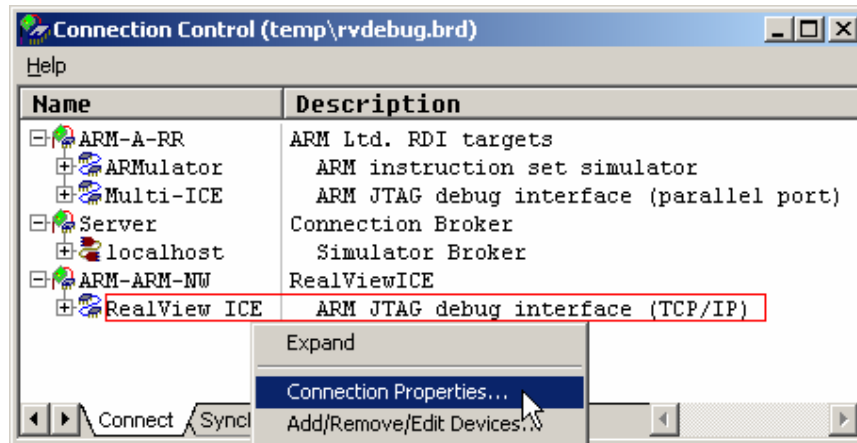




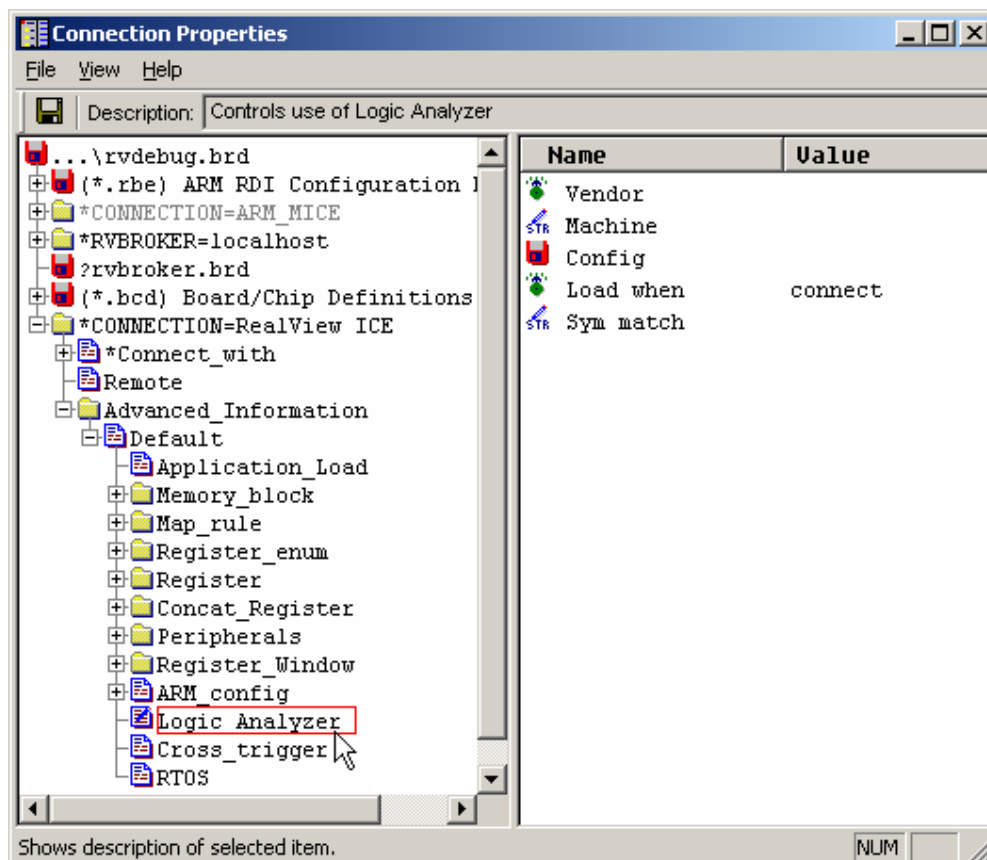
Select *File* → *Save* to save the selected device configuration, and then select *File* → *Exit* to close the RVI Config dialog.



Return to the *Connection Control* window. Right click on the *RealView ICE* entry and select *Connection Properties* from the menu.

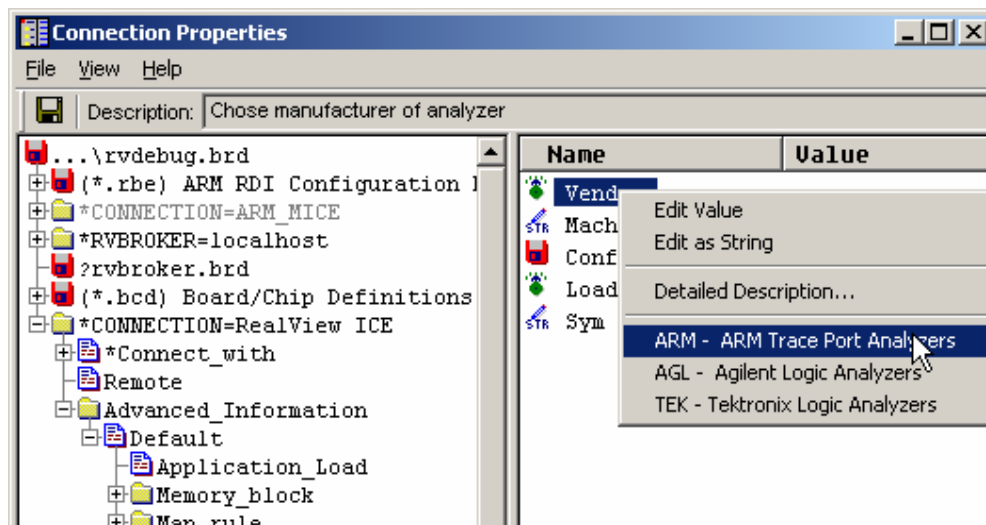


Expand the branches in the *Connection Properties* dialog to display the *Logic Analyzer* entry

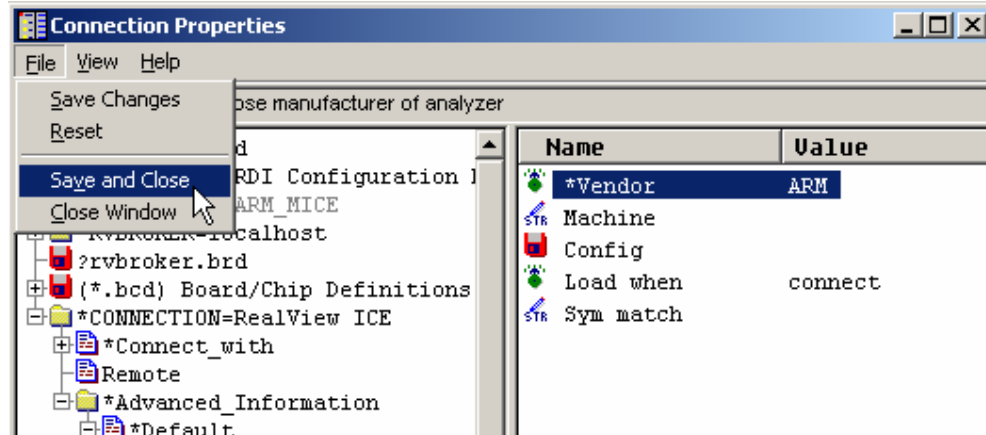




Right click on the *Vendor* entry in the right hand column and select *ARM* from the context menu *Logic Analyzer* entry



Select *File* → *Save and Close* to confirm the changes and close the *Connection Properties* window (note the *ARM* vendor entry).



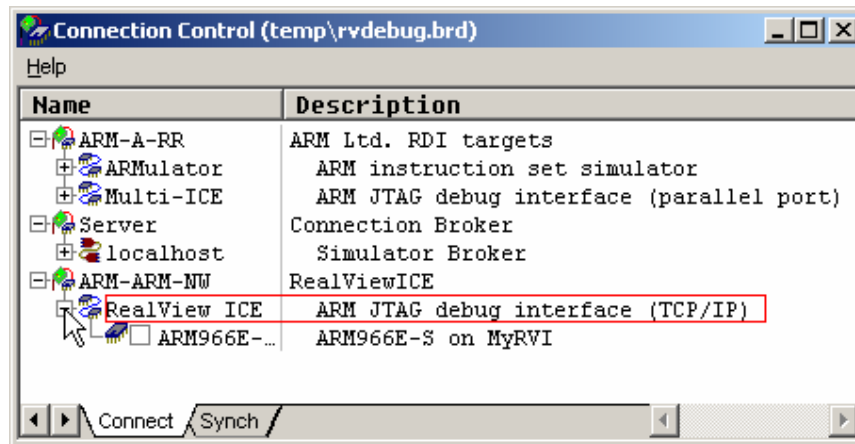
These changes tell RVD that when we connect to the RealView ICE unit, we will also be using the ARM RealView Trace device.



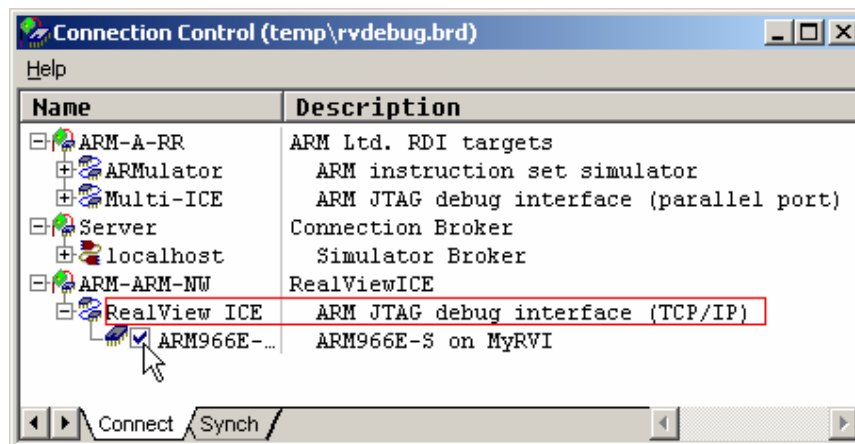
You **MUST** always set the desired *Connection Properties* BEFORE you connect to your target. If you need to change the properties when you are connected, you must disconnect, make the changes, and then reconnect.



Return to the *Connection Control* window and click on the cross to the left of the *RealView ICE* branch to open it.



Click in the box to the left of the processor name to connect to it.



2.2 – Performing Trace Capture

This section uses two simple examples to provide an introduction to using the tools to perform trace capture.



The files needed for these examples can be found in:
c:\rvds22_tutorial\rvt

Example 1 – Tracing a function

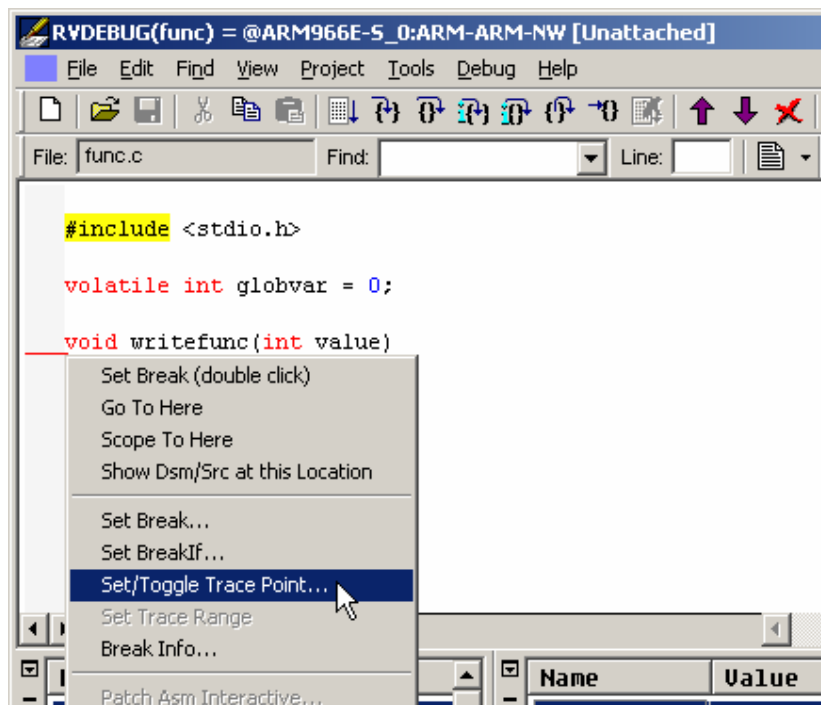
This is a simple example that shows how trace capture can be done for a specific function, **writelfunc**.



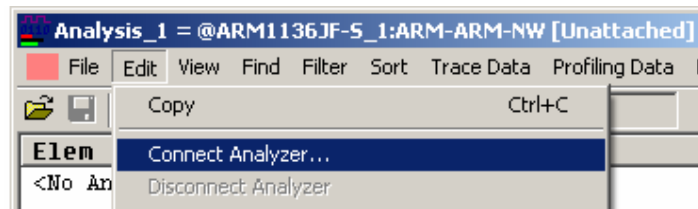
Load the example image **func.axf** by selecting *Target → Load Image* from the menu or clicking on the *Load Image* button on the toolbar.



Once loaded the image disassembly will be displayed in the debugger code window. Click the *Src* tab in the code pane.



You should see “Connecting to Analyzer...” in the Cmd tab of the output pane. If not, you must connect to the analyzer. Select *View → Analysis Window* from the menu. In the Analysis Window, select *Edit → Connect Analyzer...* from the menu as shown below.



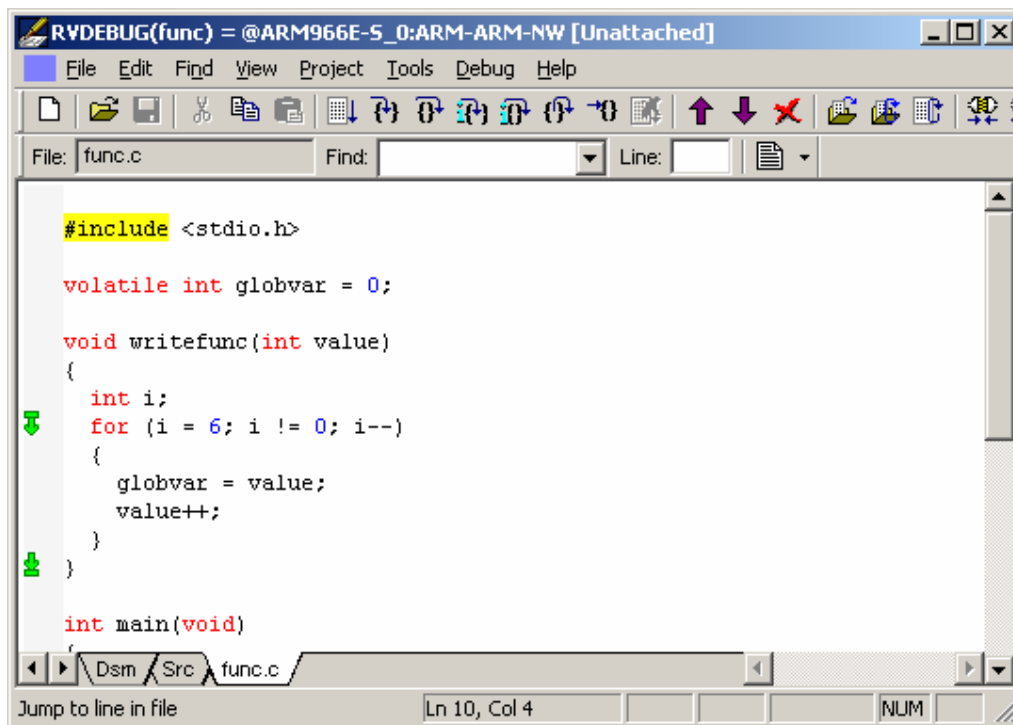
Now locate the start of the function in the code pane and right click in the pale grey margin just to the left of it. Select *Set/Toggle Trace Point* from the context menu. Select *Start of Trace Range (Instruction and Data)* from the list that appears and click *OK* to set the tracepoint.



Locate the end of the function, then right click in margin and select *Set/Toggle Trace Point* then select *End of Trace Range (Instruction and Data)* and click *OK* to set a trace stop point. The source code display should now show the recently set trace points:



If the tracepoint icons have not appeared you can refresh the display by clicking on the *Src* tab in the *Code* window.



Start executing the image by selecting *Debug* → *Run (F5)* from the menu.



Locate the *Output* pane at the bottom of the debugger window and enter a number between 1 and 10 when prompted for input.



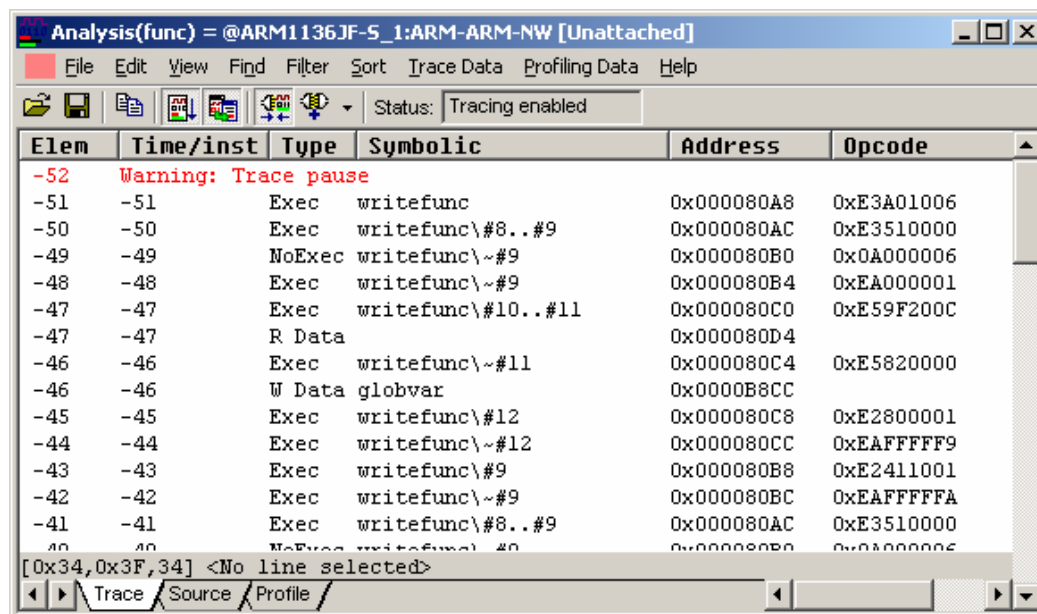
Execution halts. Select *View* → *Analysis Window* from the menu to view the captured trace.

The columns can be interpreted as follows:

<i>Elem</i>	The element number in the current trace buffer. If a trace trigger has been set then element 0 will appear at the trigger point. Otherwise element 0 will appear as the last element.
<i>Time/Inst</i>	This will show relative times in the appropriate units if a clock speed has been specified and timestamps have been enabled.
<i>Type</i>	Exec : Indicates that this instruction was executed NoExec : A conditional instruction that was not executed R Data : A data read W Data : A data write
<i>Address</i>	Indicates the address an instruction was fetched from, or the address data was read from or written to.
<i>Symbolic</i>	Gives the module name and line number for the corresponding source code.

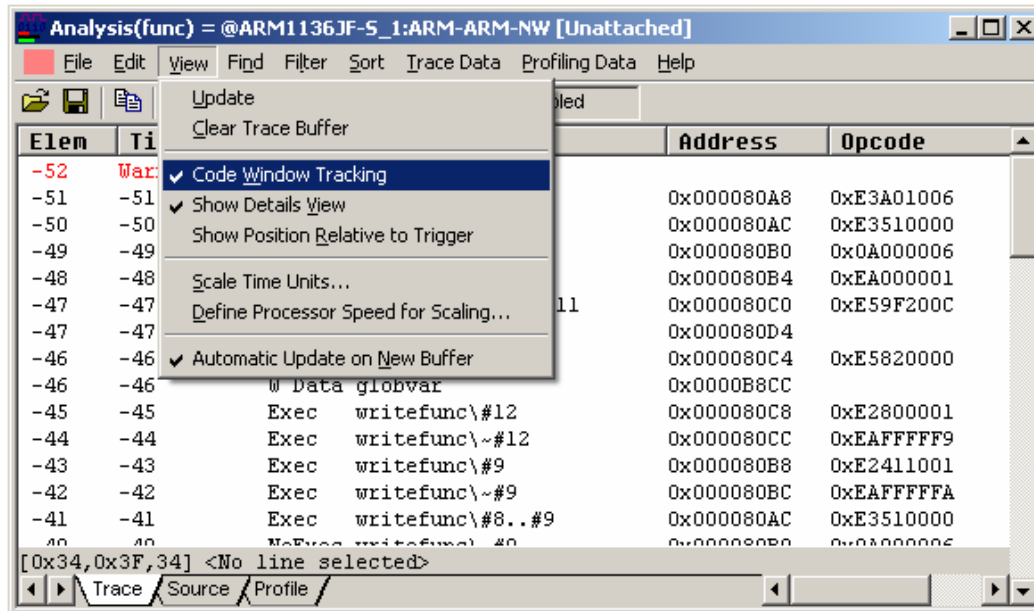


From the menu select *Trace Columns* and ensure *Data Value in Decimal* is selected:





Using the *View* menu, ensure that *Code Window Tracking* is also enabled:



Scroll through the captured trace to see the source code and disassembly synchronized with the trace data.

Note how the instructions traced correspond to the operation of **writelfunc**. Successive writes to memory can be seen as **W Data** entries.

Example 2 – Tracing a write to a global variable

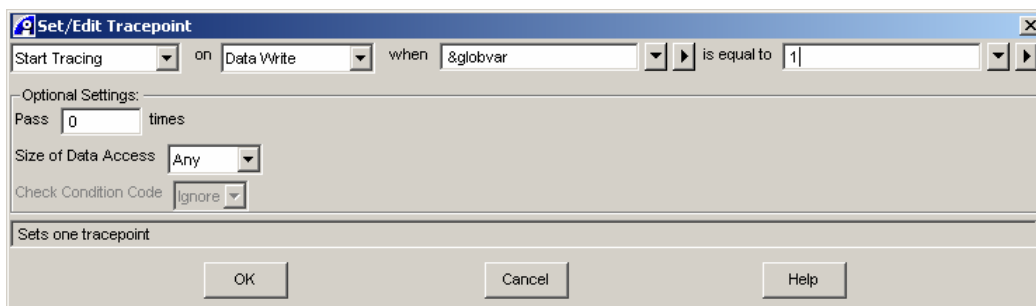
This example shows how trace capture can be turned on and off according to the value of a global variable.



Load the example image **glob.axf** by selecting *Target → Load Image* from the menu or clicking on the *Load Image* button on the toolbar.



Select *Debug → Tracepoints → Set Tracepoint...* from the menu to launch the *Tracepoint* dialog.



Ensure that *Start Tracing* is selected in the first drop-down list. Highlight the *Data Write* entry from the second list.



Enter **&globvar** in the *Location* box and **1** in the “*is equal to*” box. Check that the full line reads “*Start Tracing on Data Write when &globvar is equal to 1*” as shown above.

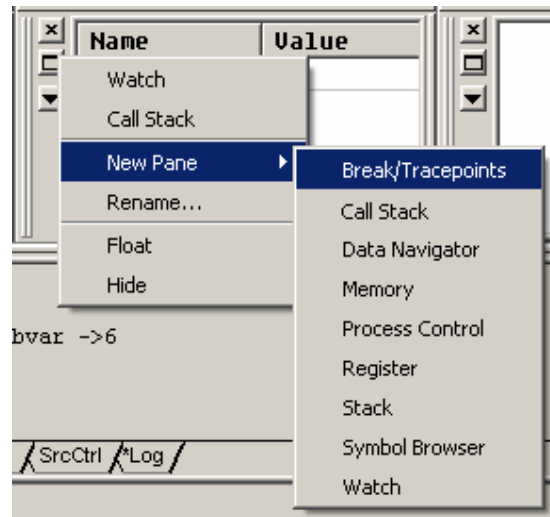
The tracepoint is now completely defined:



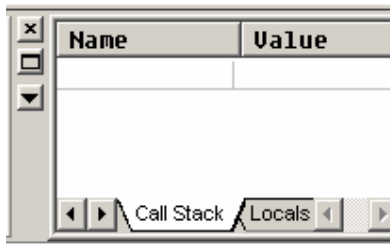
Click *OK* to set the tracepoint.



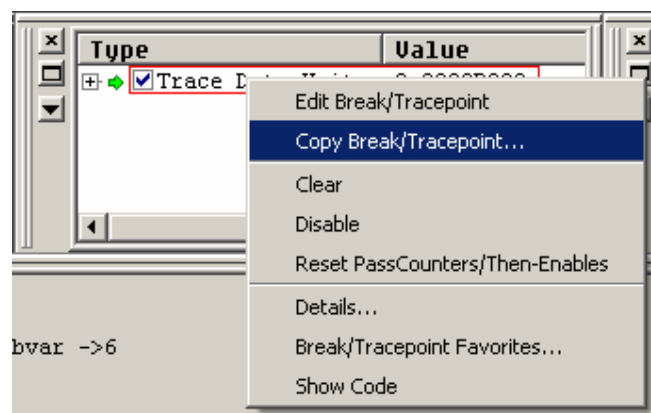
Use the *Pane Content* menu and go to *New Pane* → *Break/Tracepoints* to change the *Watch* pane to show the tracepoint.



The *Pane Content* menu can be accessed by clicking on the small icon that resembles a box:



Right click on the current tracepoint and select *Copy Break/Tracepoint* to create a second tracepoint based on the existing one. The *Set/Edit Tracepoint* dialog is shown again.



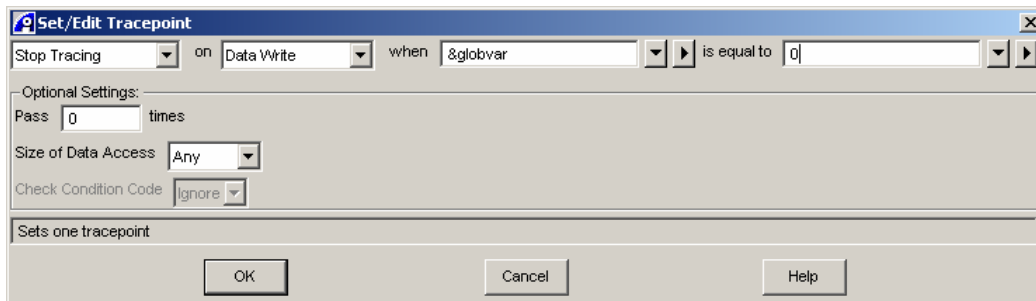


Right click the new tracepoint and select *Edit Break/Tracepoint...*



Use the *Tracepoints* dialog to configure the new tracepoint as follows:
Stop Tracing on Data Write when &globvar is equal to 0

The tracepoint is now completely defined:



Click *OK* to set the second tracepoint.

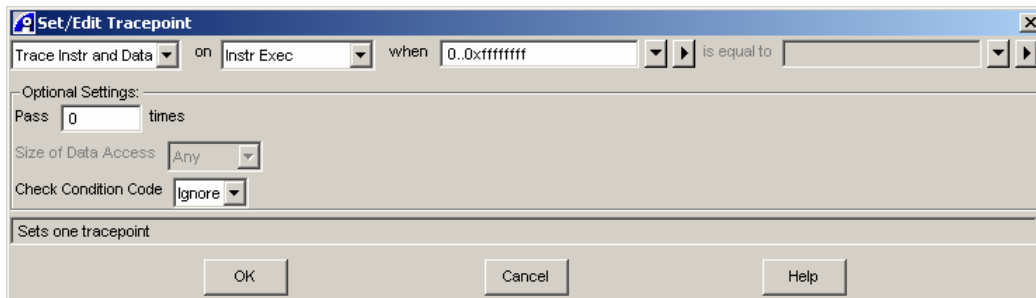


Select *Debug* → *Tracepoints* → *Set Tracepoint...* from the menu to show the *Tracepoint* dialog again.



Use the *Tracepoints* dialog to configure the final tracepoint as follows:
Trace Instr and Data on Instr Exec when 0..0xffffffff

The final tracepoint is now completely defined:



Click *OK* to set the final tracepoint.



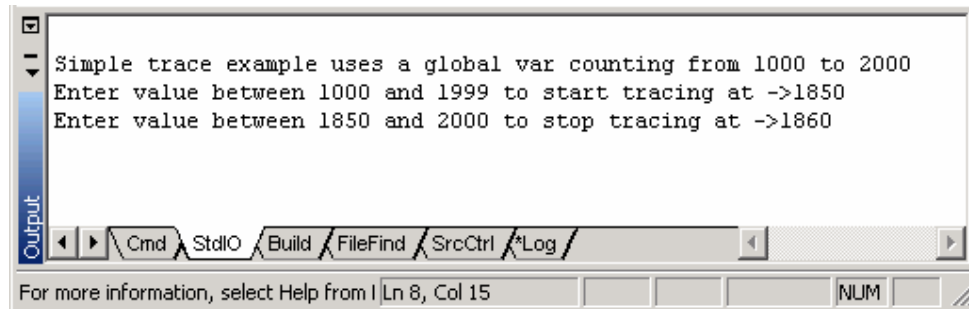
This final tracepoint enables data tracing globally. The trace capture is restricted by the first two breakpoints.



Start executing the image by selecting *Debug* → *Run (F5)* from the menu.



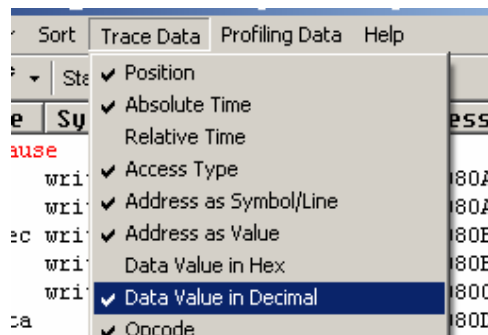
When prompted enter two values at which to start and then stop trace capture:



View the trace capture in the *Analysis* window



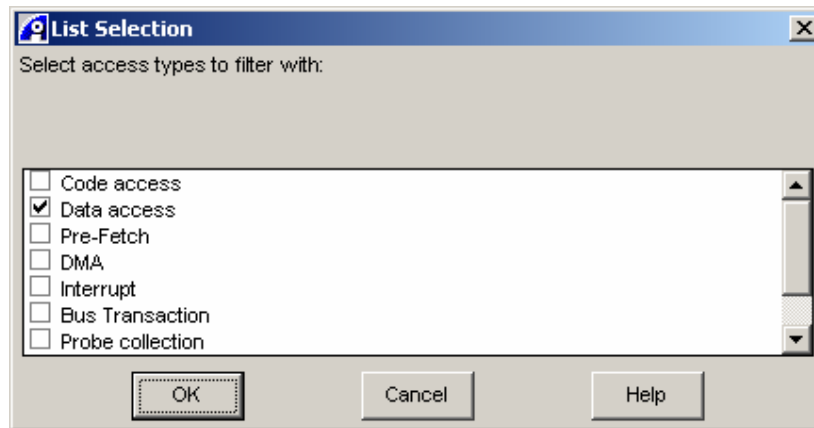
Select *Trace Columns* → *Data Value in Decimal* from the menu in the *Analysis* window to view the data reads and writes.



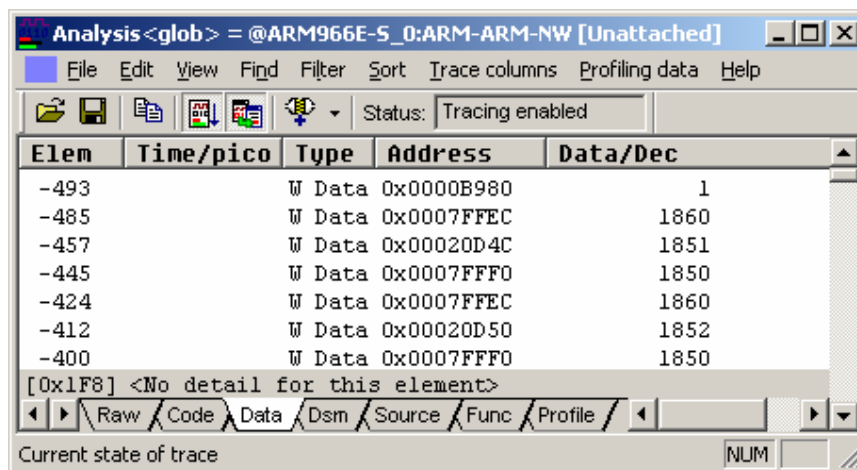
Note how the **R Data** and **W Data** entries in the *Type* column correspond to the values chosen.



Select *Filter* → *Filter on Access Type Match...* from the menu.



Clear all of the checkboxes except *Data access* and click *OK*.
The analysis window shows the trace filtered to give a concise view of the data accesses during the trace:



Select *Filter* → *Clear Filtering* from the menu in the *Analysis* window ready for the next exercise.

Example 3 – Using trace on a running system

This example illustrates the use of a RealView Trace unit with RealView Debugger to configure and capture trace from a running system.



Load the image **dhry.axf** by selecting *Target → Load Image* from the menu or clicking on the *Load Image* button on the toolbar.

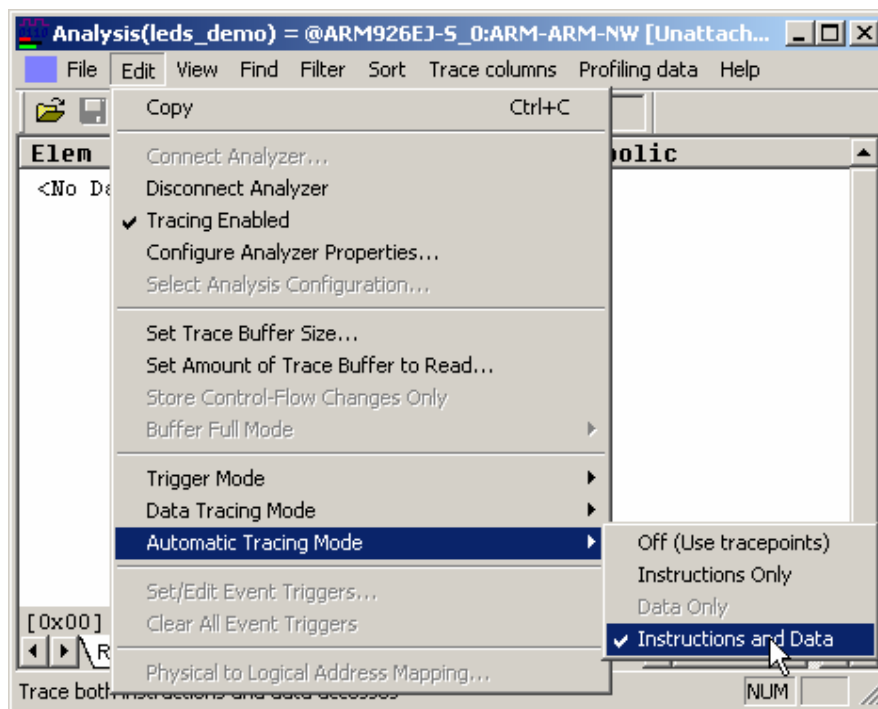
This is an infinite loop of the standard benchmarking code - Dhrystone.



Start executing the image by selecting *Debug → Run (F5)* from the menu.



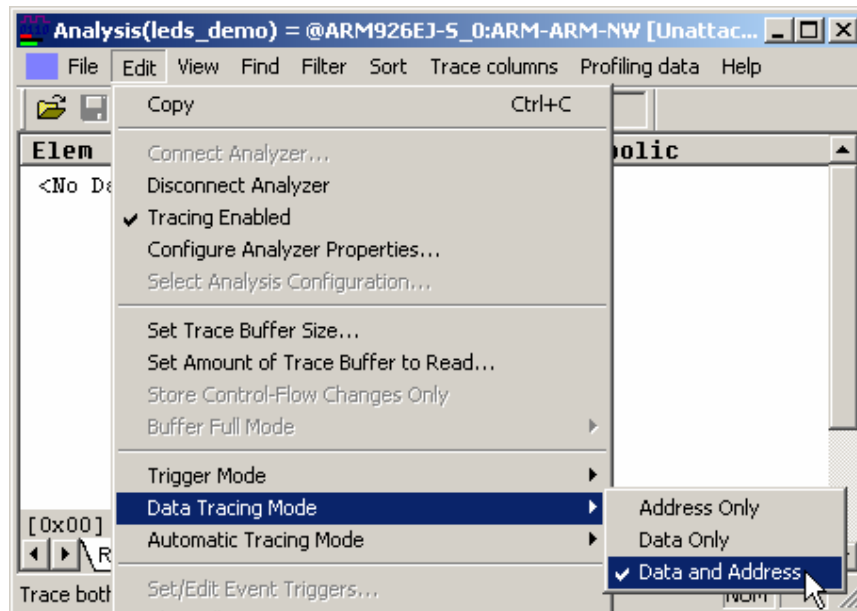
Select *Edit → Automatic Tracing Mode → Instructions and Data* from the menu in the *Analysis* window.



The debugger may prompt you to restart tracing. We would like to make further changes to the trace configuration before we restart tracing, therefore click *No*.



Select *Edit* → *Data Tracing Mode* → *Data and Address* from the menu in the *Analysis* window.



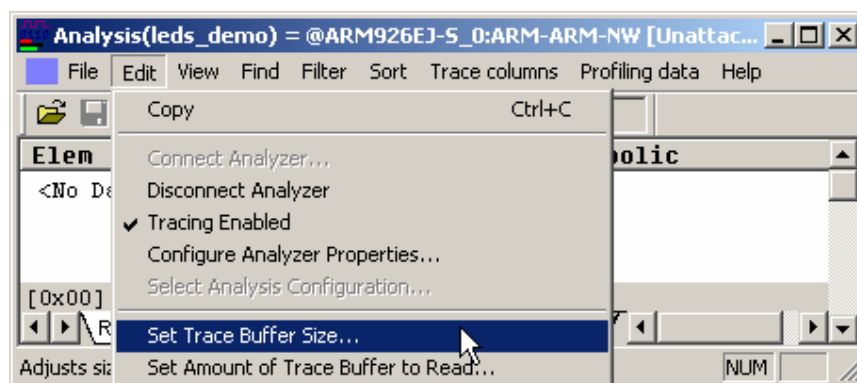
Again, the debugger may prompt you to restart tracing. Click *No*.



The debugger is now configured to automatically capture trace for both instructions & data.

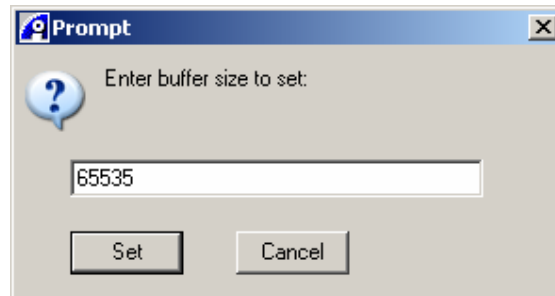


Select *Edit* → *Set Trace Buffer Size...* from the *Analysis* window menu.

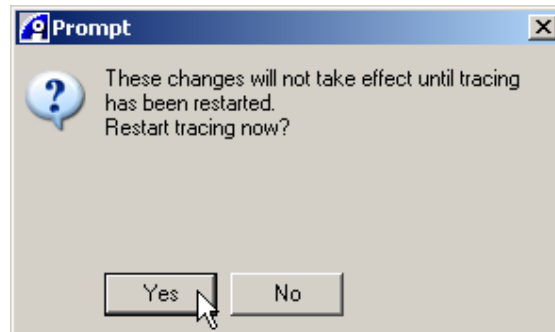




Enter **65535** at the prompt to select the new *Trace Buffer Size* to collect and click *Set*.



The debugger will prompt you a final time to restart tracing and allow the RealView ICE unit to reprogram the ETM on the target. Click *Yes*.



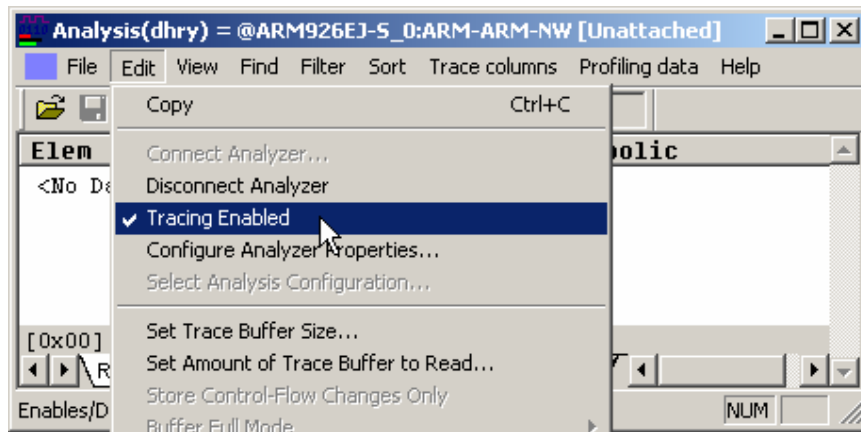
Note that it is not necessary to stop the target to reprogram the ETM.



Selecting a small trace buffer size will mean that you will only have to wait for a shorter time for the debugger to retrieve the trace data from the trace unit.



Select *Edit* → *Tracing Enabled* from the menu in the *Analysis* window to uncheck the selection and stop the trace capture.



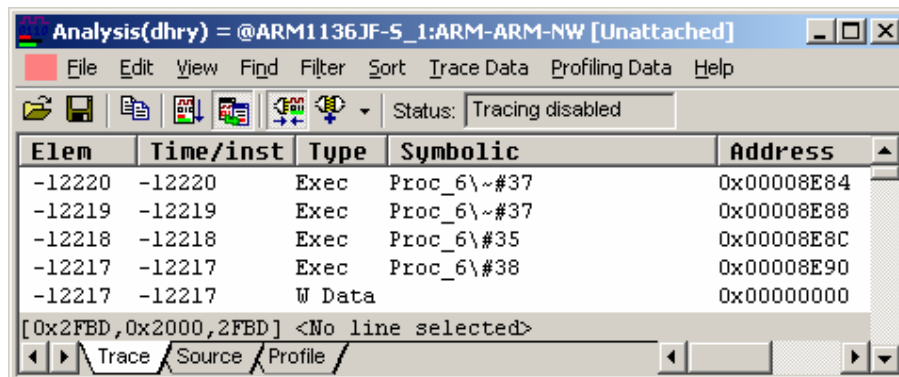
The debugger will collect the currently captured trace from the RVT unit.



Note that the debugger window will show that the target is still running:



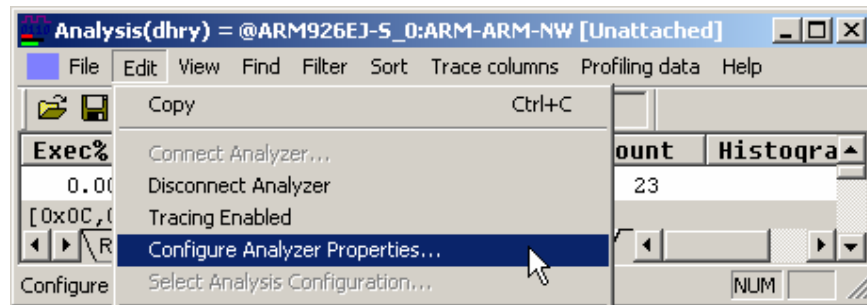
The *Analysis* window should display results similar to the following:



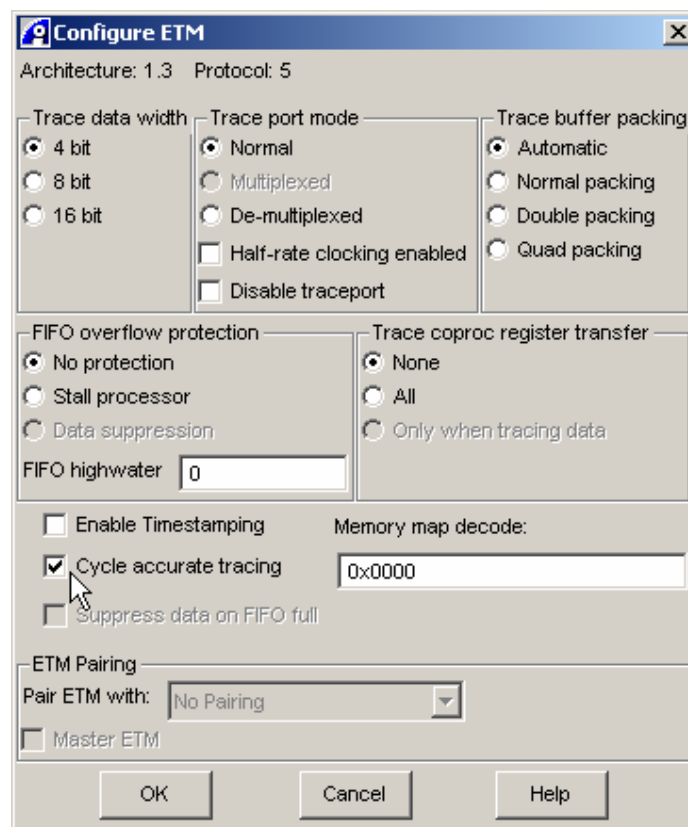
Note that the exact results displayed will be different depending on which sections of the code were being executed while trace was enabled.



Select *Edit* → *Configure Analyzer Properties...* from the menu in the *Analysis* window to open the ETM configuration dialog.



Ensure the *Cycle accurate tracing* option is selected then click *OK* to confirm the changes and reprogram the ETM.



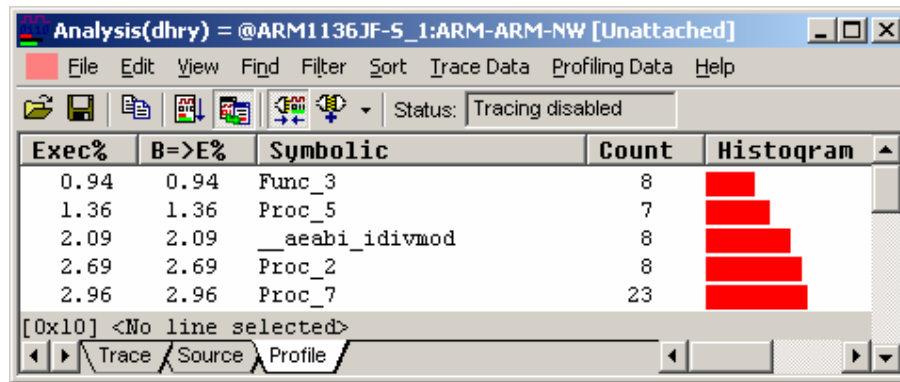
Note that this will automatically re-enable tracing.



Select *Edit* → *Tracing Enabled* from the menu in the *Analysis* window to uncheck the selection and stop the trace capture.



Click on the *Profile* tab at the bottom of the *Analysis* window to display profiling information for the current trace buffer.



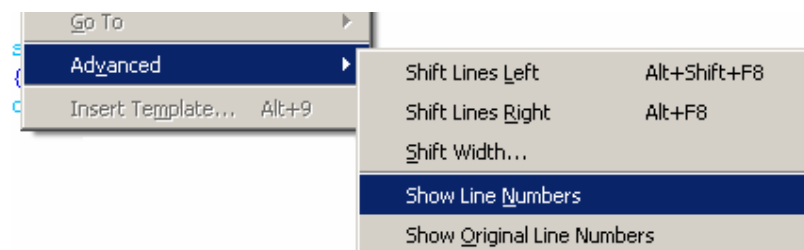
The columns can be interpreted as follows:

<i>Exec%</i>	Shows the time spent within a function as a percentage of the current trace buffer.
<i>B=>E%</i>	Shows the percentage of time spent in a function beginning to end, including time within child functions.
<i>Symbolic</i>	Gives the corresponding function name.
<i>Count</i>	Shows the number of times a function has been executed. Note main shows 0 because it was not entirely traced
<i>Histogram</i>	Representation of time spent per function. Note for RVT this will only appear for if timestamps or cycle accurate tracing is enabled. By default some logarithmic scaling is performed on histogram lengths.

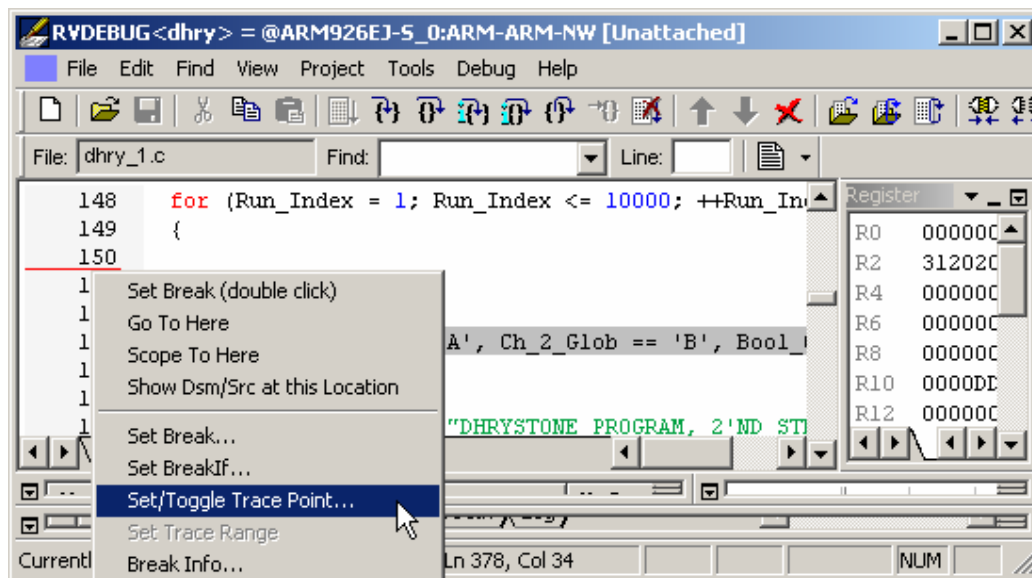
It is possible to set tracepoints inside the source code while the target is running. This allows the user to focus on areas of interest for specific trace capture.



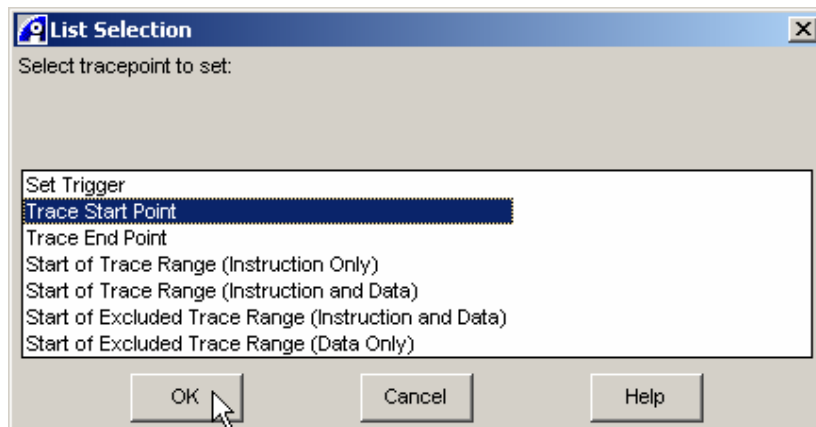
Click on the *Src* tab of the code pane in the main RVD window, then select *Edit* → *Advanced* → *Show Line Numbers* to display the source file line numbers in the code window.



Within *dhry1_c*, locate line 150 within the main **for** loop in the body of the **main** function. Right click in the grey margin and select *Set/Toggle Trace Point...* to display the *Tracepoint List Selection* dialog.

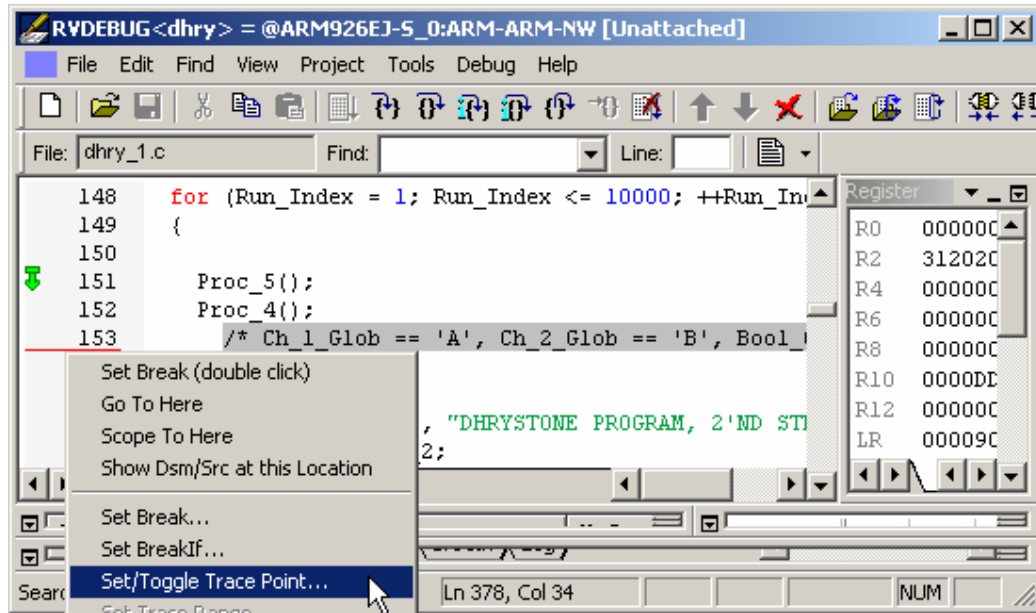


Select *Trace Start Point* from the list and click *OK* to set the tracepoint.

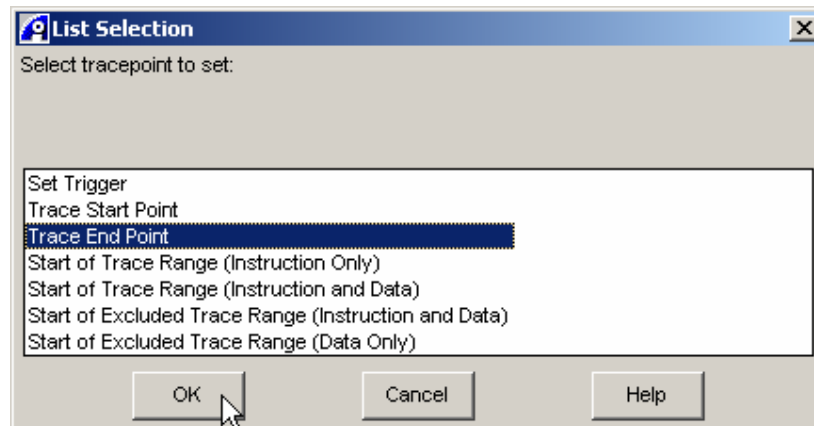




Locate line 153 then right click in the grey margin and select *Set/Toggle Trace Point...* to display the *Tracepoint List Selection* dialog again.



Select *Trace End Point* from the list and click *OK* to set the tracepoint.



The new trace selection is now completely defined:

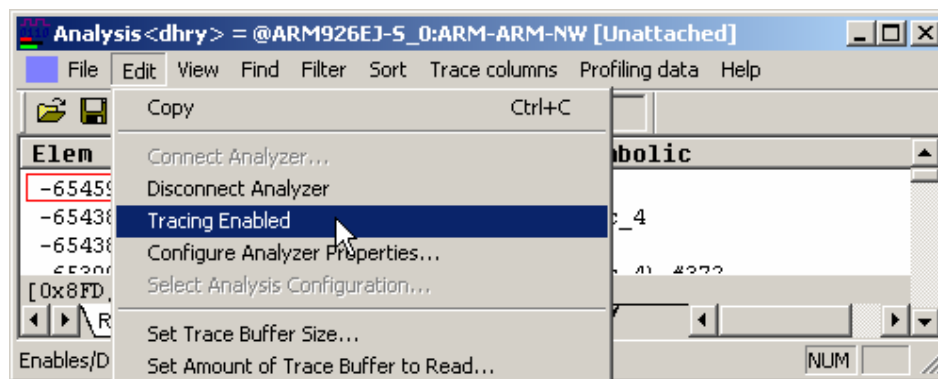
```

RVDEBUG<dhry> = @ARM926EJ-S_0:ARM-ARM-NW
File: dhry_1.c Find:
148     for (Run_Index = 1; Run_Index <=
149     {
150
151     Proc_5();
152     Proc_4();
153     /* Ch_1_Glob == 'A', Ch_2_Glob == 'B'
154     Int_1_Loc = 2;
155     Int_2_Loc = 3;
156     strcpy (Str_2_Loc, "DHRYSTONE"

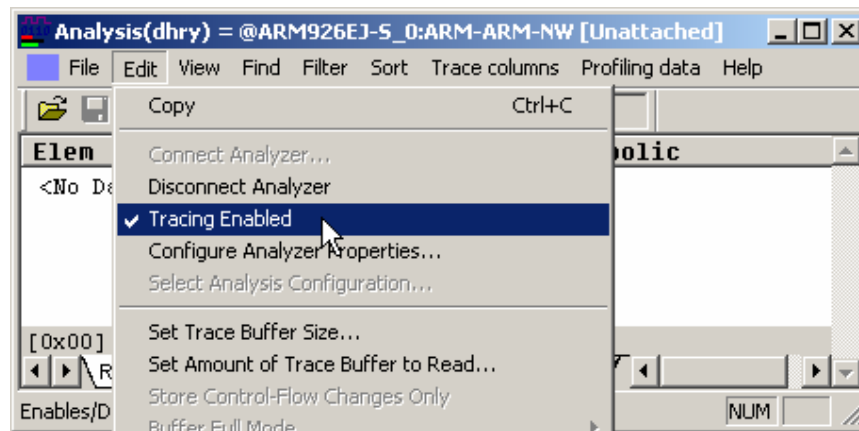
```



Select *Edit*→*Tracing Enabled* from the menu in the *Analysis* window to check the selection and enable trace capture.

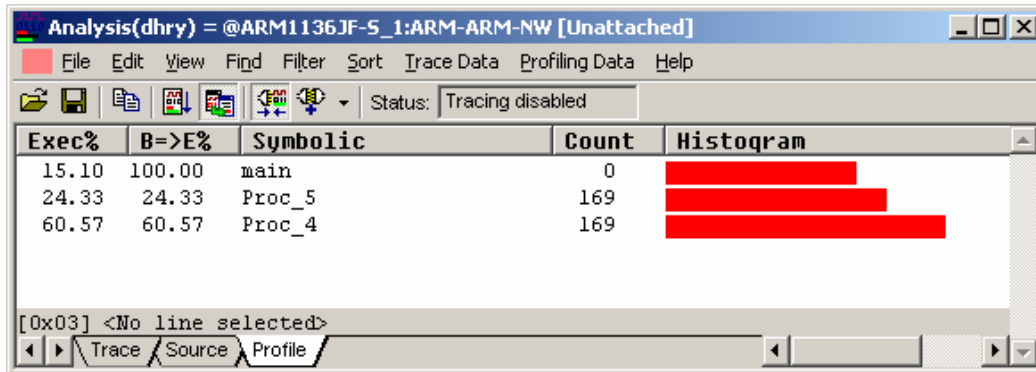


Select *Edit*→*Tracing Enabled* from the menu in the *Analysis* window to uncheck the selection and stop the trace capture.





Ensure the *Profile* tab at the bottom of the *Analysis* window is selected to display profiling information for the new trace buffer.



The profiling information has been updated to display data around our selected area of source only.



Remember that it is not necessary to stop the target during these trace configurations.



Reconfigure the tracepoints to capture another trace buffer, or close the debugger to complete the exercise.