Application Note 168

Tracing with RVD

Released on: December, 2006

DAI0168B

Application Note 168

Tracing with RVD

Release Information

The following changes have been made to this application note.

Table 1 Change history

Date	Issue	Change
September 2006	A	First release
December 2006	B	Update example for scatterloading and RVD v3.0.1

Proprietary Notice

Words and logos marked with ® and ™ are registered trademarks owned by ARM Limited, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

1 Introduction

1.1 Scope

Trace is a powerful debug tool supported by ARM cores that have an Embedded Trace Macrocell (ETM). Trace provides a capture of program instruction flow and data accesses, without impacting the real-time performance of an application. For this reason, trace is an invaluable tool when traditional halted debug methods can not be used for resolving real-time application issues.

This Application Note provides an introduction to tracing with ARM's RealView Debugger (RVD). RVD provides a powerful front end for configuring trace and displaying the results of a trace capture. Since only a limited amount of information can be collected from trace, it is important that your trace capture is properly set to isolate the area of interest for debugging or profiling. This Application Note will focus on the mechanics of performing trace capture using auto trace, trace points, trace ranges and triggers.

A sample application is used as an example throughout the Application Note. This example provides a framework for a series of trace scenarios which can be followed along as a tutorial or used as a reference to make a similar trace capture in your own application.

1.2 Assumptions

This Application Note is written for ETM-enabled targets interfaced to RVD through RealView ICE using an Embedded Trace Buffer or RealView Trace (RVT). Some of the trace scenarios will work on the RealView Instruction Set Simulator (RVISS), but tracing RVISS targets is notably different as defined in the RVD Trace User Guide. A working knowledge of RVD and RVT is assumed.

1.3 References

You may find the following references useful when reading the Application Note:

- Embedded Trace Macrocell Architecture Specification - ARM IHI 0014
- RVDS 2.2 RVD / RVT Tutorial
- RealView Debugger Command Line Reference Guide - ARM DUI 0175
- RealView Debugger Essentials Guide - ARM DUI 0181
- RealView Debugger Trace User Guide - ARM DUI 0322
- RealView Debugger User Guide - ARM DUI 0153
- RealView ICE and RealView Trace User Guide - ARM DUI 0155.

2 Sample Application

The sample application is supplied in the file TRACE.C. It simulates a small system that reads a set of input data samples, computes the sample average and then outputs the average followed by a variable number of input data samples. It yields code that is easy to follow and provides a framework for common instruction and data trace scenarios.

The application is designed to run on any hardware platform because it simulates data input and output rather than relying on specific peripherals. Data sampling and processing is initiated on a random time basis using the rand() function. Instead of reading data from an input device (such as an analog-to-digital converter), new input data is generated from previous input data. The sample average and input samples are output by writing to a fixed address in memory (intended to simulate the write buffer of a serial port).

The batch file BUILD.BAT is supplied to build the application using RealView Development Suite. The batch file compiles TRACE.C and links the application using the scatterloading file TRACE.SCAT. The scatterloading file places the executable image at 0x8000, followed by the RW and ZI data sections. The application uses a one-region memory model, with the heap and stack placed 256 bytes after the ZI section. The simulated write buffer is located in a separate section at address 0x20000. To run the application from a different address or relocate any of the memory sections, you must modify the TRACE.SCAT file accordingly.

Note

1) The sample application was built using RealView Development Suite v3.0 SP1. If you build the application using a different version of the tools, you may collect slightly different trace captures from those presented in Sections 4 - 8. Any difference in trace captures can be attributed to the difference in assembly code generated by the compiler.

2) The application must be run on RVD with semihosting enabled because it contains calls to printf().

3) The application control loop contains a call to printf() which provides feedback on program execution. Standard semihosting places the target in a debug state which slows program execution and impacts trace profiling results. You can comment out the call to printf() and rebuild the progam to make the program execute faster. If you comment out the call to printf(), your trace captures may no longer match those in Sections 4 - 8.

3 Configuring Trace

3.1 Trace Interfaces

Trace can be collected only from ARM cores that feature an Embedded Trace Macrocell (ETM). The ETM generates trace information based on your trace settings. Trace information output by the ETM must first be stored so that it may be sent to RVD for analysis. The ETM allows two different interfaces for collecting trace data - a dedicated trace port and external buffer or an on-chip Embedded Trace Buffer.

Collecting Trace from an External Trace Port using RVI/RVT

If your target utilizes a trace port, trace data is collected by RealView Trace (RVT). RVT sits on top of RealView ICE (RVI) and the two are connected through a 60-pin and 80-pin connector. RVT interfaces to the trace port through a 38-pin Mictor connector. When a trace capture is made, the ETM fills the contents of RVT memory with trace data through the trace port. The trace data is then uploaded to RVD for analysis via RVI:

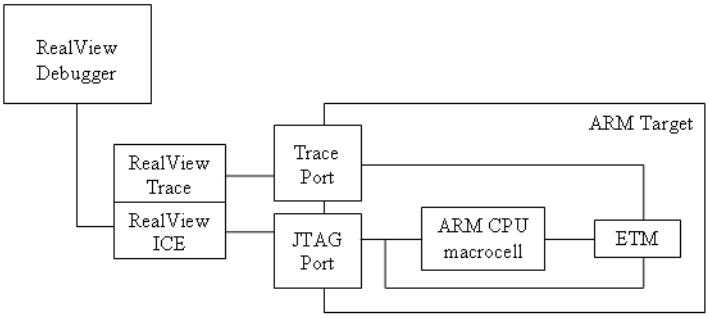


Figure 1

——— **Note** ———

This is the default configuration expected by RVD.

———————————

Collecting Trace from an ETB using RVI

If your ETM features an Embedded Trace Buffer (ETB), then you don't need a trace port or RealView Trace to make a trace capture. Instead, the ETB collects trace information and sends it to RVD through the JTAG port via RVI:

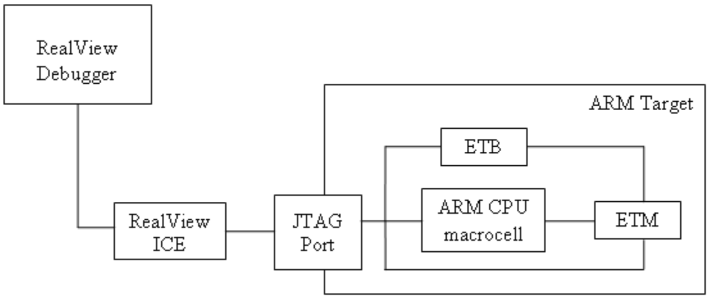
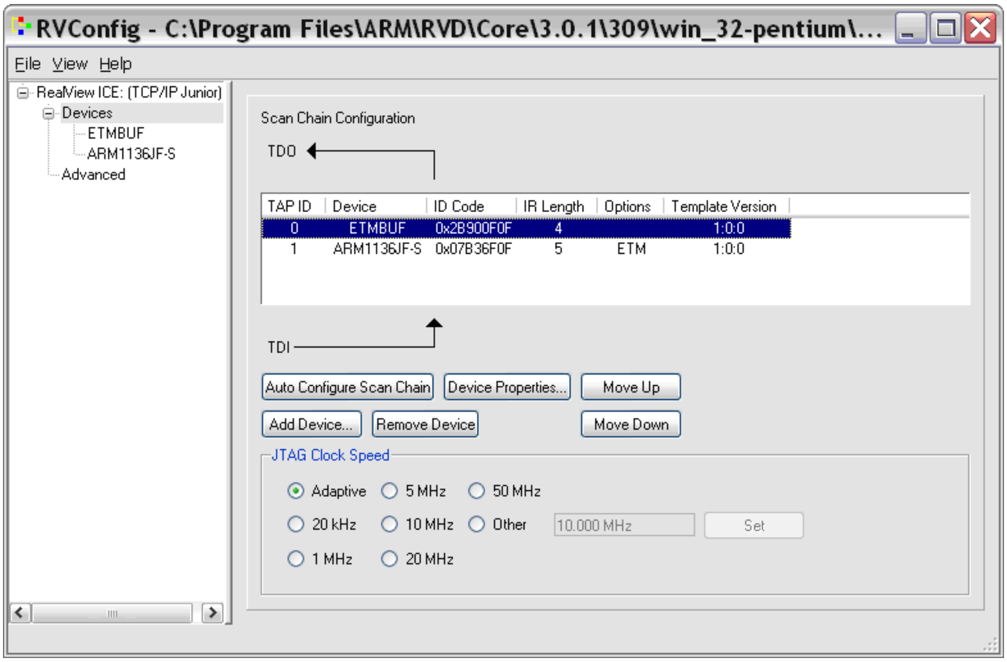


Figure 2

——— **Note** ———

The trace interface is implementation defined. It will either be a 38-pin Mictor connector or JTAG interface. Some designs may provide both interfaces. You must configure the tools to use an ETB since they default to using an external trace port.

If your target does feature an ETB, you will see it when you configure the RVI scan chain from the Connection Control window in RVD. The ETB will be identified as "ETMBUF" by the RVI configuration utility:



RVConfig - C:\Program Files\ARM\RVD\Core\3.0.1\309\win_32-pentium\...

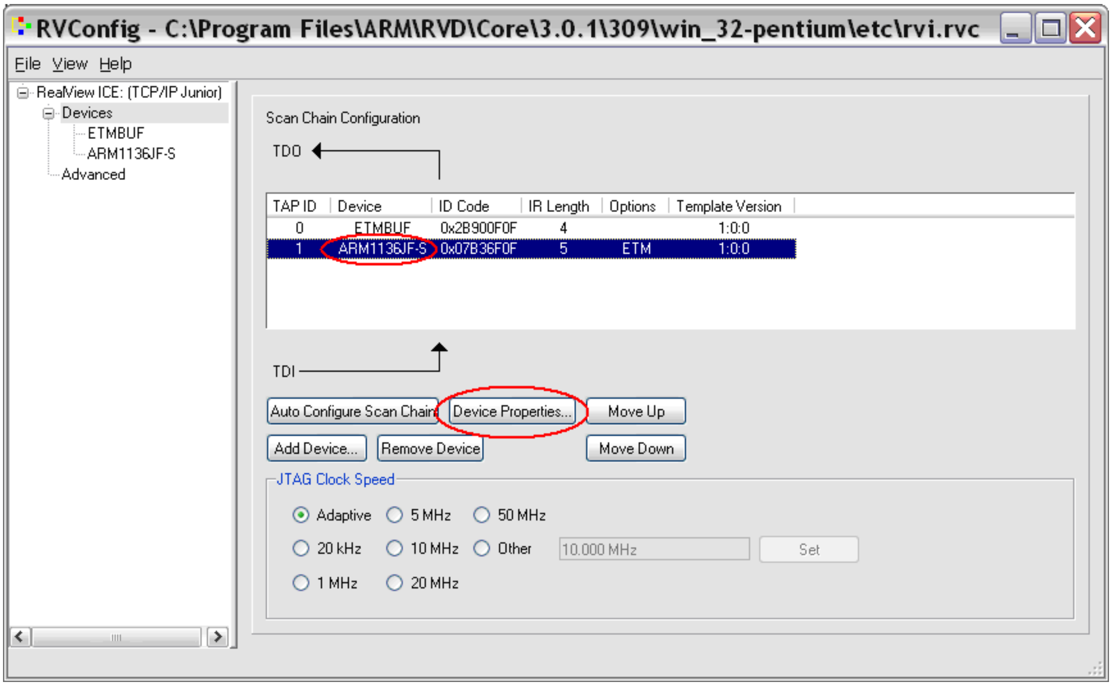
TAP ID	Device	ID Code	IR Length	Options	Template Version
0	ETMBUF	0x2B900F0F	4		1:0:0
1	ARM1136JF-S	0x07B36F0F	5	ETM	1:0:0

JTAG Clock Speed

● Adaptive ○ 5 MHz ○ 50 MHz
○ 20 kHz ○ 10 MHz ○ Other 10.000 MHz
○ 1 MHz ○ 20 MHz

Figure 3

If you want to trace using ETB, you must enable this option by selecting your ARM core in the scan chain and clicking "Device Properties":



RVConfig - C:\Program Files\ARM\RVD\Core\3.0.1\309\win_32-pentium\etc\rvi.rvc

- RealView ICE: (TCP/IP Junior)
  - Devices
    - ETMBUF
    - ARM1136JF-S
  - Advanced

Scan Chain Configuration

TDO

TAP ID	Device	ID Code	IR Length	Options	Template Version
0	ETMBUF	0x2B900F0F	4		1:0:0
1	ARM1136JF-S	0x07B36F0F	5	ETM	1:0:0

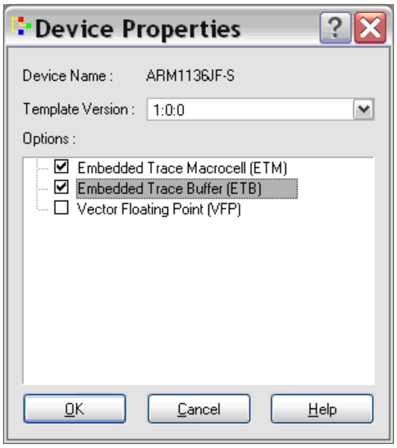
TDI

JTAG Clock Speed

● Adaptive ○ 5 MHz ○ 50 MHz
○ 20 kHz ○ 10 MHz ○ Other 10.000 MHz
○ 1 MHz ○ 20 MHz

Figure 4

Click the checkbox for the Embedded Trace Buffer and select OK:



Device Properties

Device Name : ARM1136JF-S

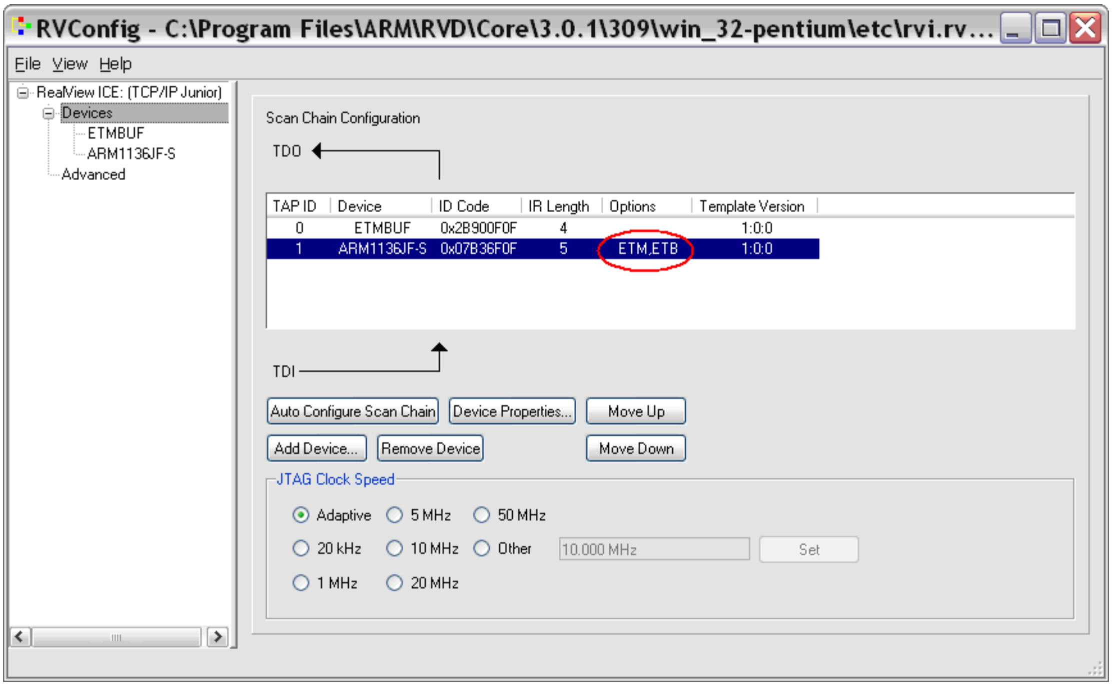
Template Version : 1:0:0

Options :

☑ Embedded Trace Macrocell (ETM)
☑ Embedded Trace Buffer (ETB)
☐ Vector Floating Point (VFP)

Figure 5

The RV Configuration utility will now show that the ETB will be used for tracing:



TAP ID	Device	ID Code	IR Length	Options	Template Version
0	ETMBUF	0x2B900F0F	4		1:0:0
1	ARM1136JF-S	0x07B36F0F	5	ETM,ETB	1:0:0

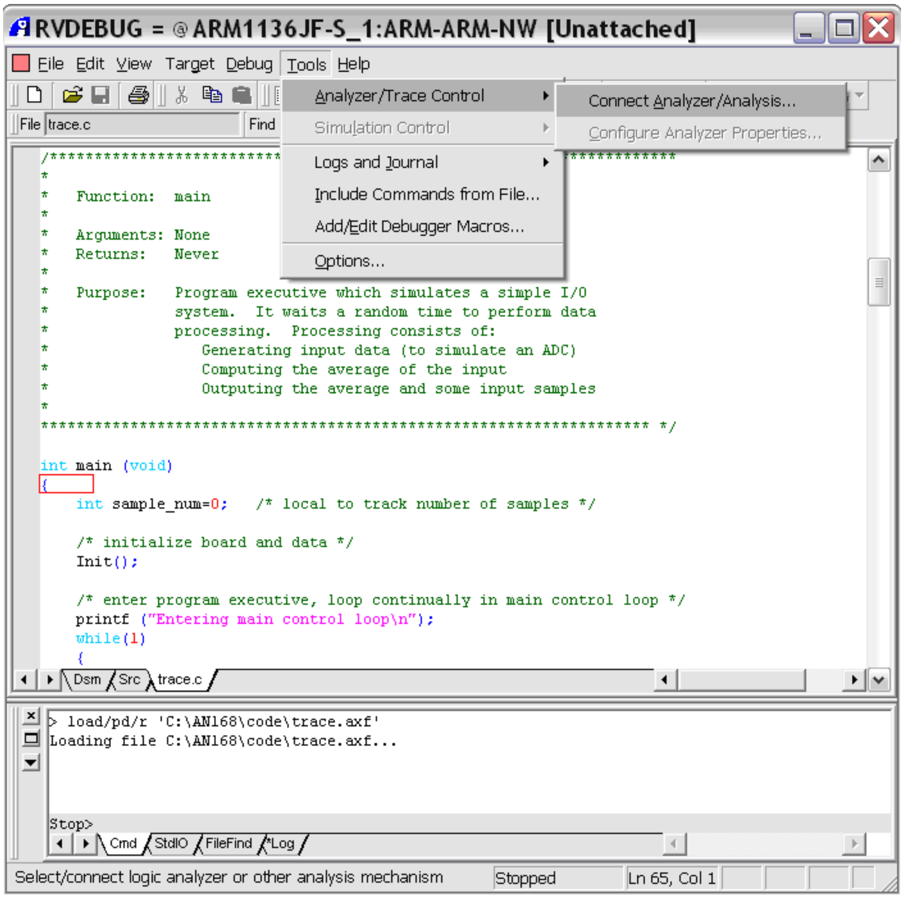
JTAG Clock Speed

● Adaptive ○ 5 MHz ○ 50 MHz
○ 20 kHz ○ 10 MHz ○ Other 10.000 MHz
○ 1 MHz ○ 20 MHz

Figure 6

3.2 Enabling and Configuring Trace

Trace is enabled and configured through RVD. Before you can configure trace, trace must first be enabled from the "Tools - Analyzer/Trace Control - Connect Analyzer/Analysis" menu.

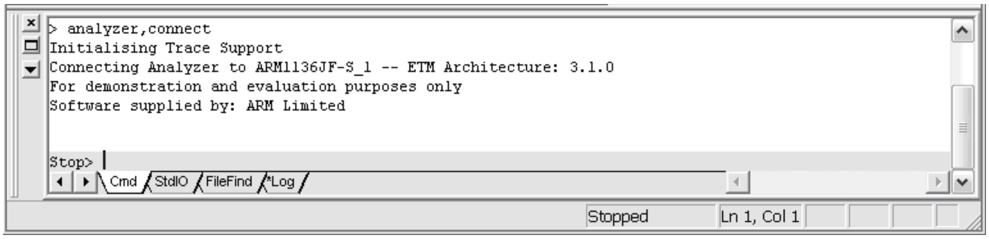

RVDEBUG = @ ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Target Debug Tools Help
File trace.c
Find
Analyzer/Trace Control
Simulation Control
Logs and Journal
Include Commands from File...
Add/Edit Debugger Macros...
Options...
Connect Analyzer/Analysis...
Configure Analyzer Properties...
Dsm
Src
trace.c
Cmd
StdIO
FileFind
Log
Select/connect logic analyzer or other analysis mechanism
Stopped
Ln 65, Col 1

```
/**************************                   *************
*
*   Function:  main
*
*   Arguments: None
*   Returns:   Never
*
*   Purpose:   Program executive which simulates a simple I/O
*              system.  It waits a random time to perform data
*              processing.  Processing consists of:
*                 Generating input data (to simulate an ADC)
*                 Computing the average of the input
*                 Outputing the average and some input samples
*
******************************************************************** */

int main (void)
{
    int sample_num=0;   /* local to track number of samples */

    /* initialize board and data */
    Init();

    /* enter program executive, loop continually in main control loop */
    printf ("Entering main control loop\n");
    while(1)
    {
```

```
> load/pd/r 'C:\AN168\code\trace.axf'
Loading file C:\AN168\code\trace.axf...

Stop>
```

Figure 7

After you enable trace, a message will be displayed indicating which ARM core and ETM you have connected to:

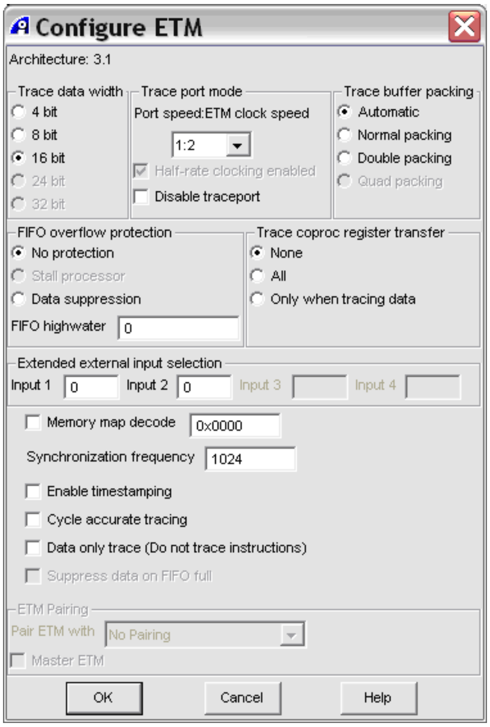


```
> analyzer,connect
Initialising Trace Support
Connecting Analyzer to ARM1136JF-S_1 -- ETM Architecture: 3.1.0
For demonstration and evaluation purposes only
Software supplied by: ARM Limited

Stop>
```

Figure 8

If you receive an error when enabling trace, verify that your target does have an ETM and that it is supported by your version of RVI firmware. If you are tracing with RVT, check that RealView ICE (RVI) and RVT are properly connected and RVT is connected to your target through the 38-pin Mictor connector. If you are tracing with an Embedded Trace Buffer (ETB), check that you have properly enabled ETB trace from the RVI Configuration Utility as described in Section 3.1.

Once trace is enabled, you can configure it from the "Tools - Analyzer/Trace Control - Configure Analyzer Properties" menu. The options available in the ETM Configuration dialogue depend upon your ETM architecture version and the features it has enabled. You can expect some dialogue options to be unavailable for selection (i.e., grayed out). A sample configuration dialogue for the ETM v3.1 architecture is shown below.



Configure ETM

Architecture: 3.1

Trace data width
○ 4 bit
○ 8 bit
● 16 bit
○ 24 bit
○ 32 bit

Trace port mode
Port speed:ETM clock speed 1:2
☑ Half-rate clocking enabled
☐ Disable traceport

Trace buffer packing
● Automatic
○ Normal packing
○ Double packing
○ Quad packing

FIFO overflow protection
● No protection
○ Stall processor
○ Data suppression
FIFO highwater 0

Trace coproc register transfer
● None
○ All
○ Only when tracing data

Extended external input selection
Input 1 0 Input 2 0 Input 3 Input 4

☐ Memory map decode 0x0000
Synchronization frequency 1024
☐ Enable timestamping
☐ Cycle accurate tracing
☐ Data only trace (Do not trace instructions)
☐ Suppress data on FIFO full

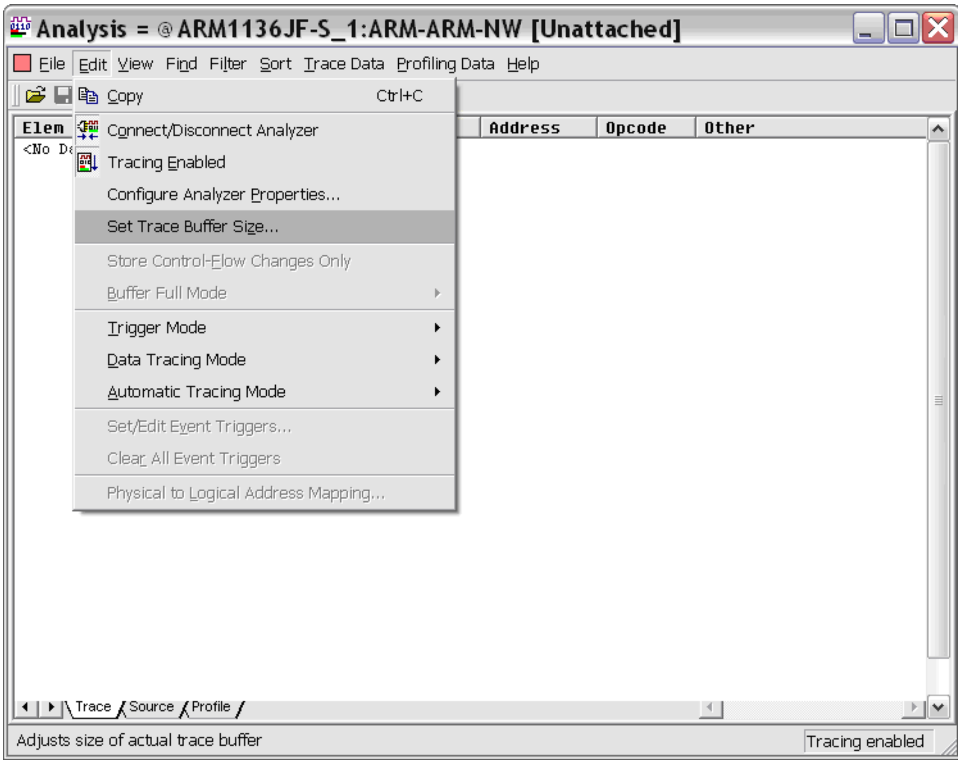
ETM Pairing
Pair ETM with No Pairing
☐ Master ETM

Figure 9

The remainder of this Application Note assumes that trace is properly enabled and the ETM is properly configured. Consult the Technical Reference Manual for your specific ETM and the RVD Trace User Guide for guidelines on configuring the ETM.

3.3 Reducing the Trace Buffer Size

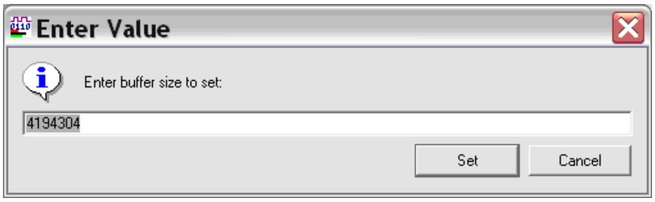
RVD sets the default trace buffer size for RVT to the maximum (4,194,304). This results in a tremendous amount of trace information that can take RVD several seconds to process and display. If you prefer to capture less data and have it displayed more quickly, you can adjust the trace buffer size using the "Edit - Set Trace Buffer Size" menu from the Analysis window:


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Copy Ctrl+C
Connect/Disconnect Analyzer
Tracing Enabled
Configure Analyzer Properties...
Set Trace Buffer Size...
Store Control-Flow Changes Only
Buffer Full Mode
Trigger Mode
Data Tracing Mode
Automatic Tracing Mode
Set/Edit Event Triggers...
Clear All Event Triggers
Physical to Logical Address Mapping...
Elem
<No Da
Address
Opcode
Other
Trace
Source
Profile
Tracing enabled

Adjusts size of actual trace buffer

Figure 10

Valid trace buffer sizes are 16,384 to 4,194,304. If you are using an ETB, the trace buffer size is based on the design of your target's ETM and it can not be modified.



Enter buffer size to set:

4194304

Figure 11

Note

A trace buffer size of 1,000,000 should be sufficient to replicate the trace captures in Sections 4 - 8.

3.4 Target Specific Information

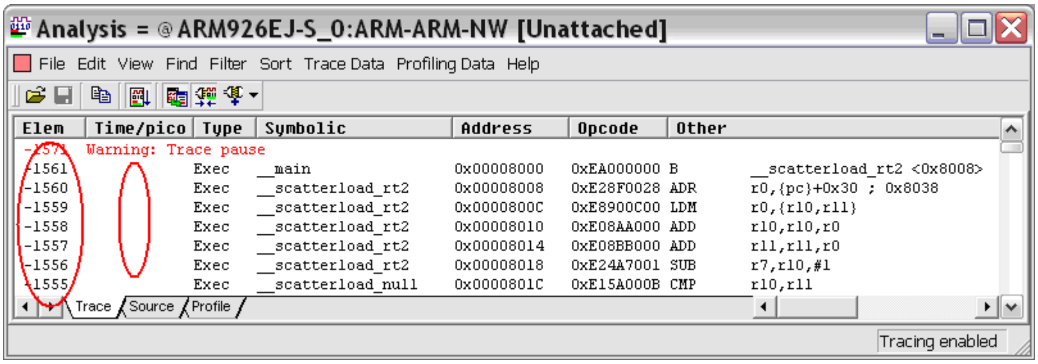
The trace captures presented in this Application Note were made using an ARM1136JF-S Core Module with RVD v3.0. This target features an ETM11RV which is based on ETM Architecture v3.1. Trace captures made from ETM v1.x architectures will appear different in RVD v3.0:

- For v1.x ETMs, the Elem (element) column starts with "-N" and decrements to the last element "0". The element number may not always decrement by one. The decrement factor depends on the contents of each trace packet and the trace buffer packing setting. For v3.x ETMs, the element column starts with 0 and increments sequentially to "N".

- For v1.x ETMs, the Time/cycle column of the Analysis window is unpopulated if Cycle Accurate trace or Timestamping is not enabled. For v3.x ETMs, the Time/cycle column provides an instruction count if Cycle Accurate trace or Timestamping is not enabled.

The following captures, from an ARM926 (ETM v1.3) and ARM1136 (ETM v3.1) respectively, demonstrate these differences:

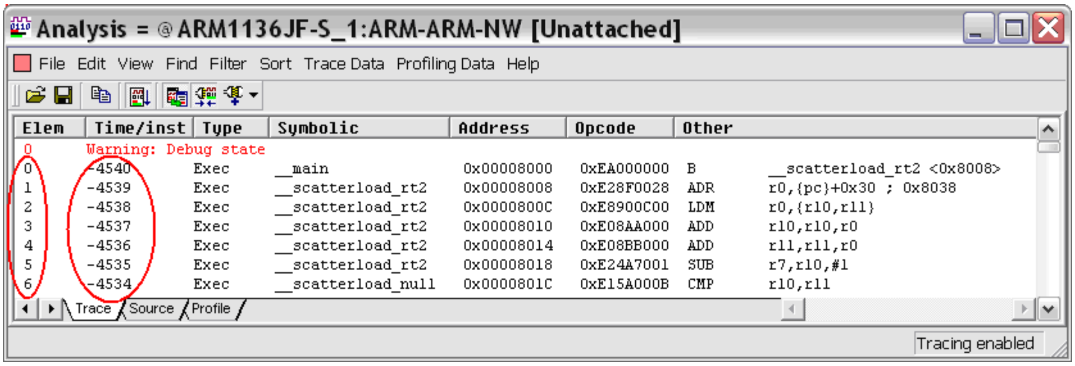
Trace from ARM926EJ-S:


Analysis = @ARM926EJ-S_0:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Time/pico	Type	Symbolic	Address	Opcode		Other
-1571	Warning: Trace pause						
-1561		Exec	__main	0x00008000	0xEA000000	B	__scatterload_rt2 <0x8008>
-1560		Exec	__scatterload_rt2	0x00008008	0xE28F0028	ADR	r0,{pc}+0x30 ; 0x8038
-1559		Exec	__scatterload_rt2	0x0000800C	0xE8900C00	LDM	r0,{r10,r11}
-1558		Exec	__scatterload_rt2	0x00008010	0xE08AA000	ADD	r10,r10,r0
-1557		Exec	__scatterload_rt2	0x00008014	0xE08BB000	ADD	r11,r11,r0
-1556		Exec	__scatterload_rt2	0x00008018	0xE24A7001	SUB	r7,r10,#1
-1555		Exec	__scatterload_null	0x0000801C	0xE15A000B	CMP	r10,r11

Figure 12

Trace from ARM1136JF-S:


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Time/inst	Type	Symbolic	Address	Opcode		Other
0	Warning: Debug state						
0	-4540	Exec	__main	0x00008000	0xEA000000	B	__scatterload_rt2 <0x8008>
1	-4539	Exec	__scatterload_rt2	0x00008008	0xE28F0028	ADR	r0,{pc}+0x30 ; 0x8038
2	-4538	Exec	__scatterload_rt2	0x0000800C	0xE8900C00	LDM	r0,{r10,r11}
3	-4537	Exec	__scatterload_rt2	0x00008010	0xE08AA000	ADD	r10,r10,r0
4	-4536	Exec	__scatterload_rt2	0x00008014	0xE08BB000	ADD	r11,r11,r0
5	-4535	Exec	__scatterload_rt2	0x00008018	0xE24A7001	SUB	r7,r10,#1
6	-4534	Exec	__scatterload_null	0x0000801C	0xE15A000B	CMP	r10,r11

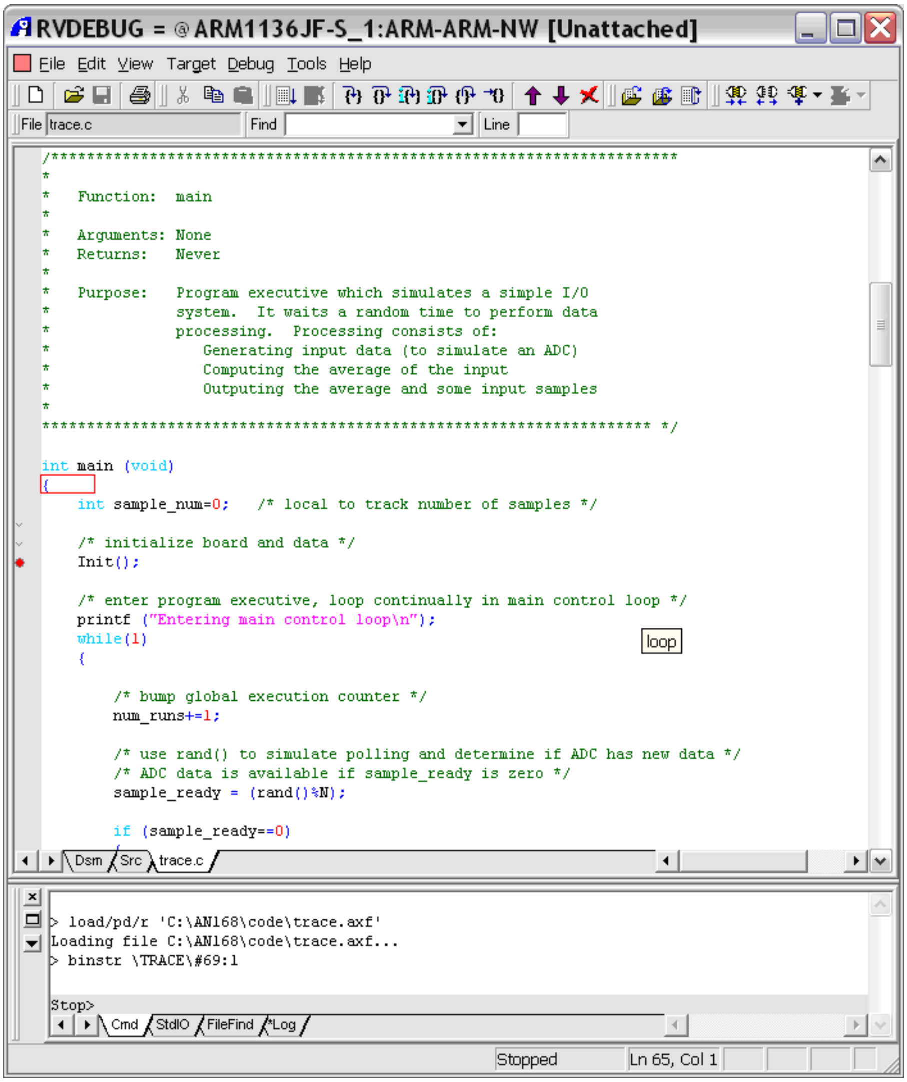
Figure 13

4 Auto Tracing

After trace is enabled and configured as described in Section 3, RVD places your target in an Auto Tracing mode by default. This means that if you run and halt your program, instruction trace will automatically be captured.

4.1 Tracing from Program Entry

To observe how Auto Tracing works, load TRACE.AXF to your target. Place a breakpoint at the call to Init() from main(), as shown below.



RVDEBUG = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Target Debug Tools Help
File trace.c
Find
Line
loop
Dsm
Src
trace.c
Cmd
StdIO
FileFind
Log
Stopped
Ln 65, Col 1

```
/***************************************************************************
*
*   Function:  main
*
*   Arguments: None
*   Returns:   Never
*
*   Purpose:   Program executive which simulates a simple I/O
*              system.  It waits a random time to perform data
*              processing.  Processing consists of:
*                 Generating input data (to simulate an ADC)
*                 Computing the average of the input
*                 Outputing the average and some input samples
*
************************************************************************ */

int main (void)
{
    int sample_num=0;   /* local to track number of samples */

    /* initialize board and data */
    Init();

    /* enter program executive, loop continually in main control loop */
    printf ("Entering main control loop\n");
    while(1)
    {

        /* bump global execution counter */
        num_runs+=1;

        /* use rand() to simulate polling and determine if ADC has new data */
        /* ADC data is available if sample_ready is zero */
        sample_ready = (rand()%N);

        if (sample_ready==0)
```

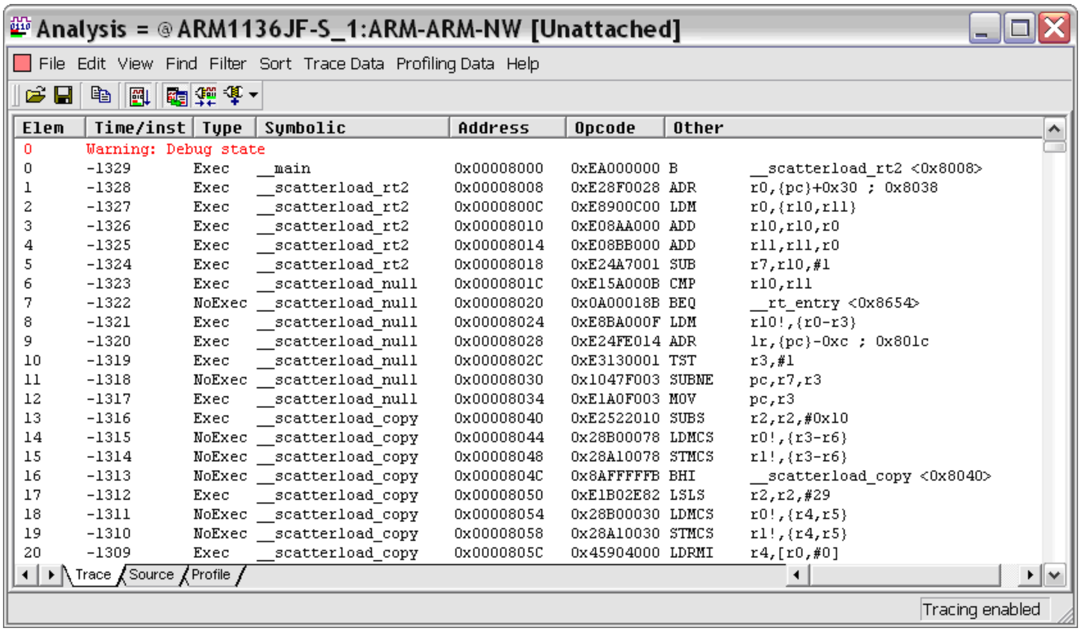
```
> load/pd/r 'C:\AN168\code\trace.axf'
Loading file C:\AN168\code\trace.axf...
> binstr \TRACE\#69:1

Stop>
```

Figure 14

Run to the breakpoint. When the breakpoint is reached, program execution is stopped and the contents of the trace buffer are sent to RVD for display.

A trace capture is viewed in the Trace Analysis window. The Trace Analysis window can be opened from the "View - Analysis Window" menu. This option is only available when trace is enabled.

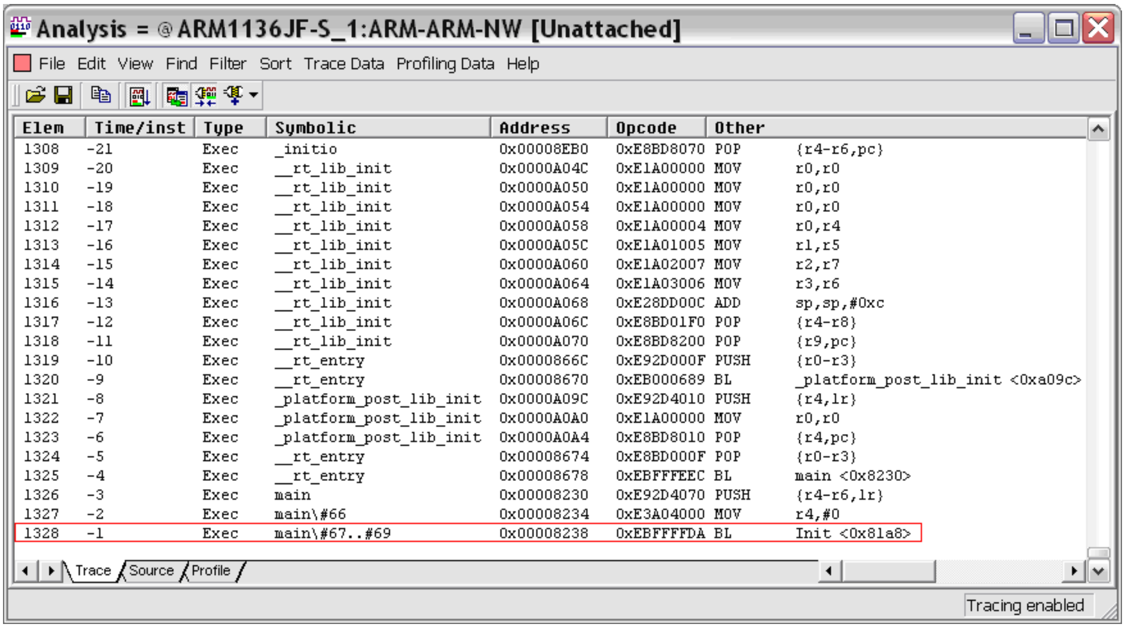
Open the Trace Analysis and observe your Auto Tracing capture. You should see that tracing begins from your program entry point, 0x8000 if built with the supplied files.


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Time/inst	Type	Symbolic	Address	Opcode	Other
0	Warning: Debug state					
0	-1329	Exec	__main	0x00008000	0xEA000000 B	__scatterload_rt2 <0x8008>
1	-1328	Exec	__scatterload_rt2	0x00008008	0xE28F0028 ADR	r0,{pc}+0x30 ; 0x8038
2	-1327	Exec	__scatterload_rt2	0x0000800C	0xE8900C00 LDM	r0,{r10,r11}
3	-1326	Exec	__scatterload_rt2	0x00008010	0xE08AA000 ADD	r10,r10,r0
4	-1325	Exec	__scatterload_rt2	0x00008014	0xE08BB000 ADD	r11,r11,r0
5	-1324	Exec	__scatterload_rt2	0x00008018	0xE24A7001 SUB	r7,r10,#1
6	-1323	Exec	__scatterload_null	0x0000801C	0xE15A000B CMP	r10,r11
7	-1322	NoExec	__scatterload_null	0x00008020	0x0A00018B BEQ	__rt_entry <0x8654>
8	-1321	Exec	__scatterload_null	0x00008024	0xE8BA000F LDM	r10!,{r0-r3}
9	-1320	Exec	__scatterload_null	0x00008028	0xE24FE014 ADR	lr,{pc}-0xc ; 0x801c
10	-1319	Exec	__scatterload_null	0x0000802C	0xE3130001 TST	r3,#1
11	-1318	NoExec	__scatterload_null	0x00008030	0x1047F003 SUBNE	pc,r7,r3
12	-1317	Exec	__scatterload_null	0x00008034	0xE1A0F003 MOV	pc,r3
13	-1316	Exec	__scatterload_copy	0x00008040	0xE2522010 SUBS	r2,r2,#0x10
14	-1315	NoExec	__scatterload_copy	0x00008044	0x28B00078 LDMCS	r0!,{r3-r6}
15	-1314	NoExec	__scatterload_copy	0x00008048	0x28A10078 STMCS	r1!,{r3-r6}
16	-1313	NoExec	__scatterload_copy	0x0000804C	0x8AFFFFFB BHI	__scatterload_copy <0x8040>
17	-1312	Exec	__scatterload_copy	0x00008050	0xE1B02E82 LSLS	r2,r2,#29
18	-1311	NoExec	__scatterload_copy	0x00008054	0x28B00030 LDMCS	r0!,{r4,r5}
19	-1310	NoExec	__scatterload_copy	0x00008058	0x28A10030 STMCS	r1!,{r4,r5}
20	-1309	Exec	__scatterload_copy	0x0000805C	0x45904000 LDRMI	r4,[r0,#0]

Figure 15

The trace capture contains the application initialization performed by the C library. If you examine the program's assembly code from the "Dsm" source tab in RVD and scroll to the end of the trace buffer, you will see that tracing ends with the instruction immediately preceding the call to Init() in main(). This instruction is highlighted in red below.

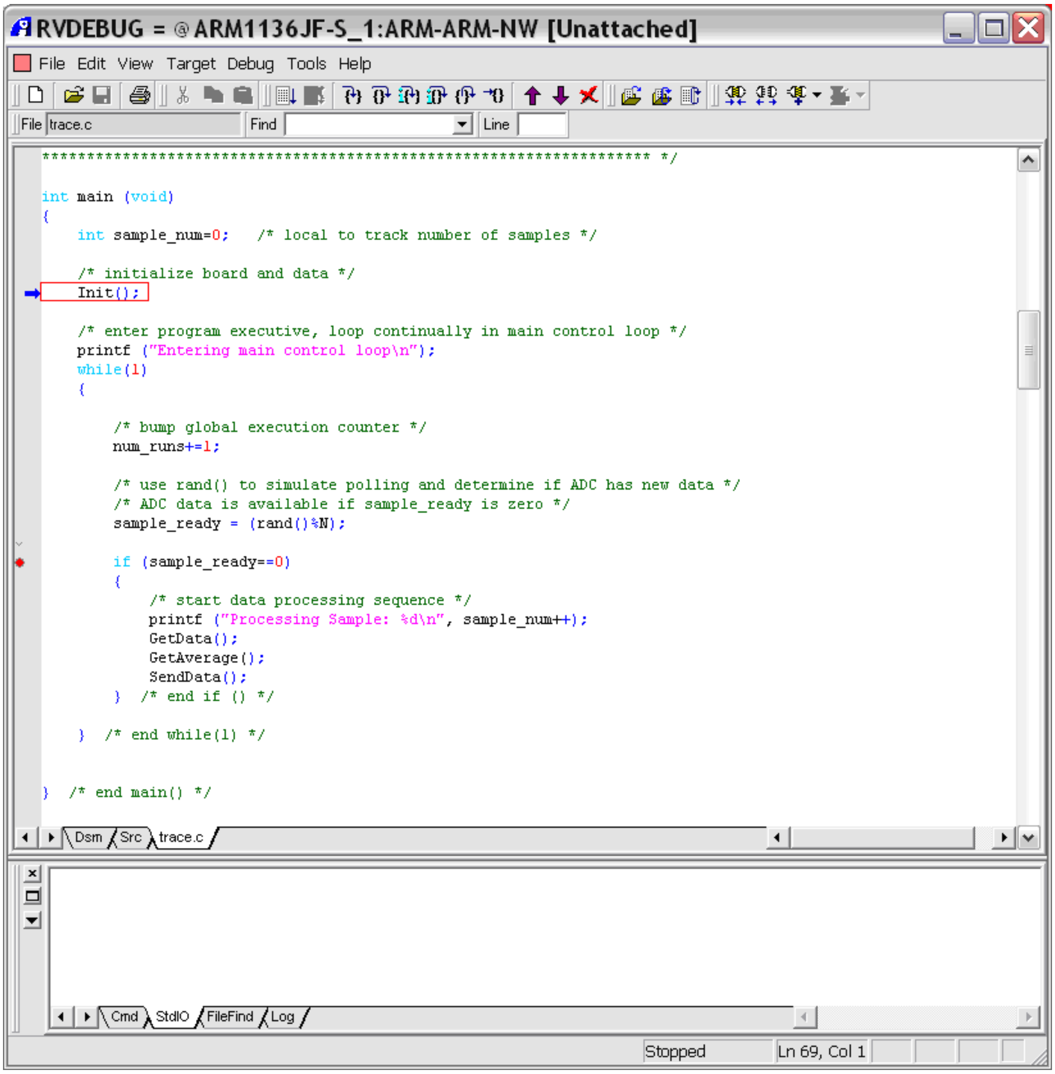

Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Time/inst	Type	Symbolic	Address	Opcode	Other
1308	-21	Exec	_initio	0x00008EB0	0xE8BD8070 POP	{r4-r6,pc}
1309	-20	Exec	__rt_lib_init	0x0000A04C	0xE1A00000 MOV	r0,r0
1310	-19	Exec	__rt_lib_init	0x0000A050	0xE1A00000 MOV	r0,r0
1311	-18	Exec	__rt_lib_init	0x0000A054	0xE1A00000 MOV	r0,r0
1312	-17	Exec	__rt_lib_init	0x0000A058	0xE1A00004 MOV	r0,r4
1313	-16	Exec	__rt_lib_init	0x0000A05C	0xE1A01005 MOV	r1,r5
1314	-15	Exec	__rt_lib_init	0x0000A060	0xE1A02007 MOV	r2,r7
1315	-14	Exec	__rt_lib_init	0x0000A064	0xE1A03006 MOV	r3,r6
1316	-13	Exec	__rt_lib_init	0x0000A068	0xE28DD00C ADD	sp,sp,#0xc
1317	-12	Exec	__rt_lib_init	0x0000A06C	0xE8BD01F0 POP	{r4-r8}
1318	-11	Exec	__rt_lib_init	0x0000A070	0xE8BD8200 POP	{r9,pc}
1319	-10	Exec	__rt_entry	0x0000866C	0xE92D000F PUSH	{r0-r3}
1320	-9	Exec	__rt_entry	0x00008670	0xEB000689 BL	_platform_post_lib_init <0xa09c>
1321	-8	Exec	_platform_post_lib_init	0x0000A09C	0xE92D4010 PUSH	{r4,lr}
1322	-7	Exec	_platform_post_lib_init	0x0000A0A0	0xE1A00000 MOV	r0,r0
1323	-6	Exec	_platform_post_lib_init	0x0000A0A4	0xE8BD8010 POP	{r4,pc}
1324	-5	Exec	__rt_entry	0x00008674	0xE8BD000F POP	{r0-r3}
1325	-4	Exec	__rt_entry	0x00008678	0xEBFFFEEC BL	main <0x8230>
1326	-3	Exec	main	0x00008230	0xE92D4070 PUSH	{r4-r6,lr}
1327	-2	Exec	main\#66	0x00008234	0xE3A04000 MOV	r4,#0
1328	-1	Exec	main\#67..#69	0x00008238	0xEBFFFFDA BL	Init <0x81a8>

Figure 16

4.2 Tracing to Another Breakpoint

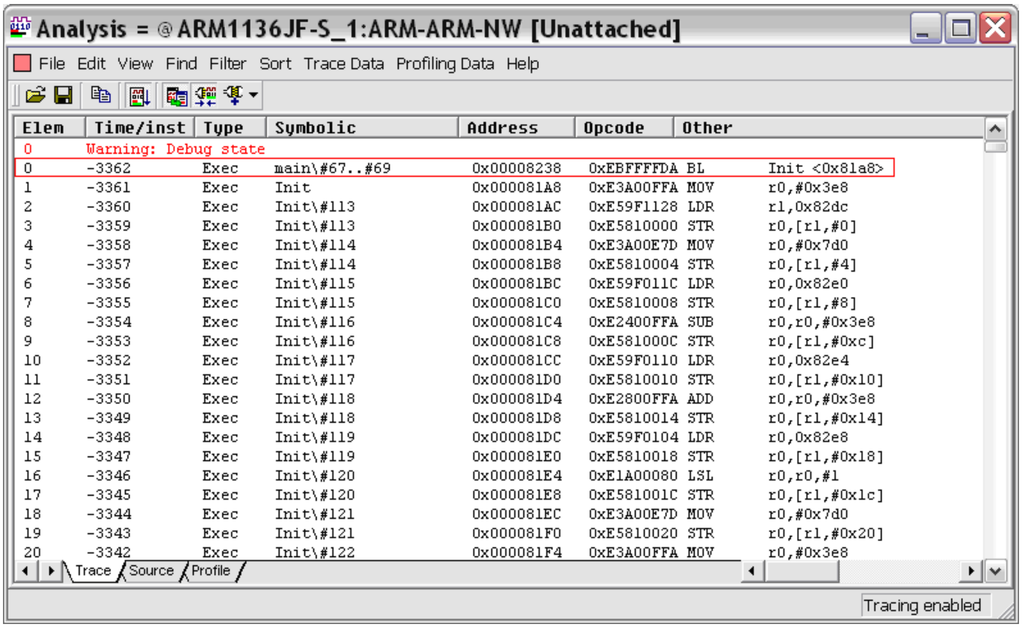
As a follow on to the capture made in Section 4.1, remove the breakpoint at the call to Init() and place another breakpoint in the while(1) loop at the instruction if (sample_ready == 0), as shown below.


RVDEBUG = @ ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Target Debug Tools Help
File trace.c
Find
Line
Dsm
Src
trace.c
Cmd
StdIO
FileFind
Log
Stopped
Ln 69, Col 1

```
******************************************************************** */

int main (void)
{
    int sample_num=0;   /* local to track number of samples */

    /* initialize board and data */
    Init();

    /* enter program executive, loop continually in main control loop */
    printf ("Entering main control loop\n");
    while(1)
    {

        /* bump global execution counter */
        num_runs+=1;

        /* use rand() to simulate polling and determine if ADC has new data */
        /* ADC data is available if sample_ready is zero */
        sample_ready = (rand()%N);

        if (sample_ready==0)
        {
            /* start data processing sequence */
            printf ("Processing Sample: %d\n", sample_num++);
            GetData();
            GetAverage();
            SendData();
        }  /* end if () */

    }  /* end while(1) */


}  /* end main() */
```

Figure 17

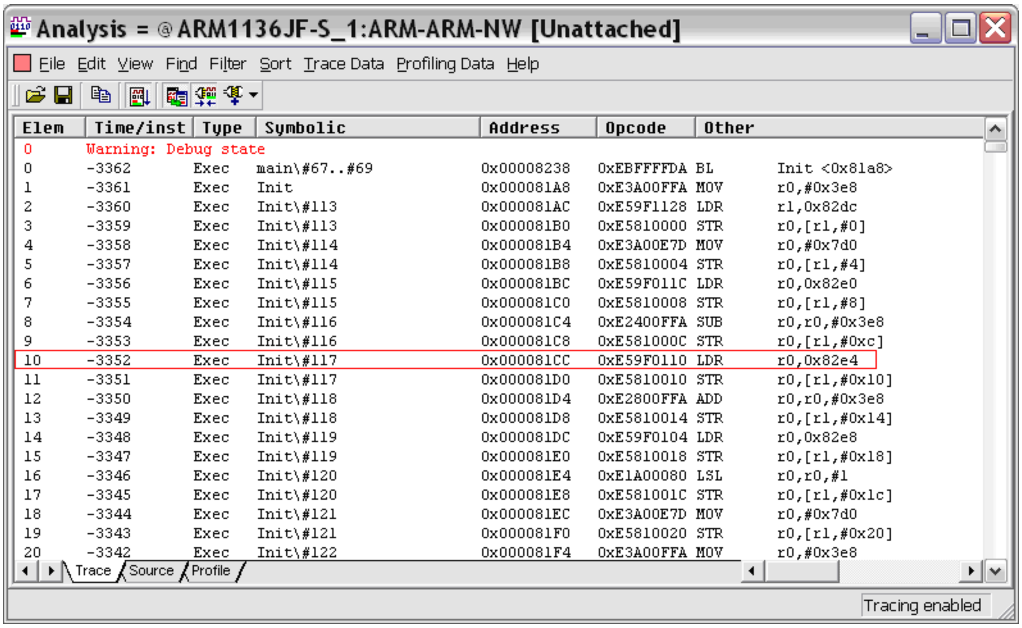
Run to the new breakpoint from the current PC location and inspect the Trace Analysis window. First notice that the old trace has been discarded. The new trace begins with the instruction BL Init, which corresponds to the function call to Init(). This instruction is highlighted by the red box below.


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Time/inst	Type	Symbolic	Address	Opcode	Other
0	Warning: Debug state					
0	-3362	Exec	main\#67..#69	0x00008238	0xEBFFFFDA BL	Init <0x81a8>
1	-3361	Exec	Init	0x000081A8	0xE3A00FFA MOV	r0,#0x3e8
2	-3360	Exec	Init\#113	0x000081AC	0xE59F1128 LDR	r1,0x82dc
3	-3359	Exec	Init\#113	0x000081B0	0xE5810000 STR	r0,[r1,#0]
4	-3358	Exec	Init\#114	0x000081B4	0xE3A00E7D MOV	r0,#0x7d0
5	-3357	Exec	Init\#114	0x000081B8	0xE5810004 STR	r0,[r1,#4]
6	-3356	Exec	Init\#115	0x000081BC	0xE59F011C LDR	r0,0x82e0
7	-3355	Exec	Init\#115	0x000081C0	0xE5810008 STR	r0,[r1,#8]
8	-3354	Exec	Init\#116	0x000081C4	0xE2400FFA SUB	r0,r0,#0x3e8
9	-3353	Exec	Init\#116	0x000081C8	0xE581000C STR	r0,[r1,#0xc]
10	-3352	Exec	Init\#117	0x000081CC	0xE59F0110 LDR	r0,0x82e4
11	-3351	Exec	Init\#117	0x000081D0	0xE5810010 STR	r0,[r1,#0x10]
12	-3350	Exec	Init\#118	0x000081D4	0xE2800FFA ADD	r0,r0,#0x3e8
13	-3349	Exec	Init\#118	0x000081D8	0xE5810014 STR	r0,[r1,#0x14]
14	-3348	Exec	Init\#119	0x000081DC	0xE59F0104 LDR	r0,0x82e8
15	-3347	Exec	Init\#119	0x000081E0	0xE5810018 STR	r0,[r1,#0x18]
16	-3346	Exec	Init\#120	0x000081E4	0xE1A00080 LSL	r0,r0,#1
17	-3345	Exec	Init\#120	0x000081E8	0xE581001C STR	r0,[r1,#0x1c]
18	-3344	Exec	Init\#121	0x000081EC	0xE3A00E7D MOV	r0,#0x7d0
19	-3343	Exec	Init\#121	0x000081F0	0xE5810020 STR	r0,[r1,#0x20]
20	-3342	Exec	Init\#122	0x000081F4	0xE3A00FFA MOV	r0,#0x3e8

Figure 18

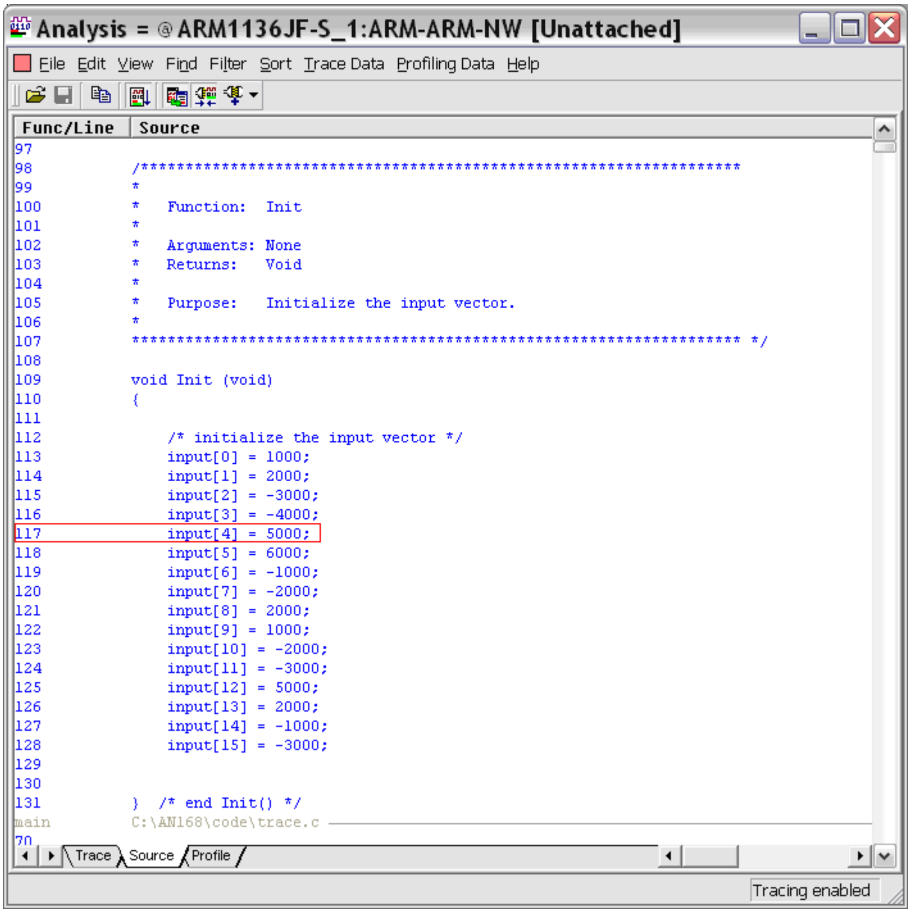
The Analysis window supports source code tracking to help you follow the execution of your program. Highlight a sample from Init(), such as trace Element 10 shown below.


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Time/inst	Type	Symbolic	Address	Opcode	Other
0	Warning: Debug state					
0	-3362	Exec	main\#67..#69	0x00008238	0xEBFFFFDA BL	Init <0x81a8>
1	-3361	Exec	Init	0x000081A8	0xE3A00FFA MOV	r0,#0x3e8
2	-3360	Exec	Init\#113	0x000081AC	0xE59F1128 LDR	r1,0x82dc
3	-3359	Exec	Init\#113	0x000081B0	0xE5810000 STR	r0,[r1,#0]
4	-3358	Exec	Init\#114	0x000081B4	0xE3A00E7D MOV	r0,#0x7d0
5	-3357	Exec	Init\#114	0x000081B8	0xE5810004 STR	r0,[r1,#4]
6	-3356	Exec	Init\#115	0x000081BC	0xE59F011C LDR	r0,0x82e0
7	-3355	Exec	Init\#115	0x000081C0	0xE5810008 STR	r0,[r1,#8]
8	-3354	Exec	Init\#116	0x000081C4	0xE2400FFA SUB	r0,r0,#0x3e8
9	-3353	Exec	Init\#116	0x000081C8	0xE581000C STR	r0,[r1,#0xc]
10	-3352	Exec	Init\#117	0x000081CC	0xE59F0110 LDR	r0,0x82e4
11	-3351	Exec	Init\#117	0x000081D0	0xE5810010 STR	r0,[r1,#0x10]
12	-3350	Exec	Init\#118	0x000081D4	0xE2800FFA ADD	r0,r0,#0x3e8
13	-3349	Exec	Init\#118	0x000081D8	0xE5810014 STR	r0,[r1,#0x14]
14	-3348	Exec	Init\#119	0x000081DC	0xE59F0104 LDR	r0,0x82e8
15	-3347	Exec	Init\#119	0x000081E0	0xE5810018 STR	r0,[r1,#0x18]
16	-3346	Exec	Init\#120	0x000081E4	0xE1A00080 LSL	r0,r0,#1
17	-3345	Exec	Init\#120	0x000081E8	0xE581001C STR	r0,[r1,#0x1c]
18	-3344	Exec	Init\#121	0x000081EC	0xE3A00E7D MOV	r0,#0x7d0
19	-3343	Exec	Init\#121	0x000081F0	0xE5810020 STR	r0,[r1,#0x20]
20	-3342	Exec	Init\#122	0x000081F4	0xE3A00FFA MOV	r0,#0x3e8

Figure 19

Now click on the Source tab in the Analysis window and observe the effects of source code tracking. Namely, line 117 in the source file TRACE.C is now highlighted. This line of C code corresponds to the highlighted element from the Trace tab. Now highlight another line from the Source tab and toggle back to the Trace tab. Observe that source code tracking can be used from both tabs.



Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Func/Line
Source
Trace
Source
Profile
Tracing enabled

```

        /**********************************************************************
        *
        *   Function:  Init
        *
        *   Arguments: None
        *   Returns:   Void
        *
        *   Purpose:   Initialize the input vector.
        *
        ********************************************************************** */

        void Init (void)
        {

            /* initialize the input vector */
            input[0] = 1000;
            input[1] = 2000;
            input[2] = -3000;
            input[3] = -4000;
            input[4] = 5000;
            input[5] = 6000;
            input[6] = -1000;
            input[7] = -2000;
            input[8] = 2000;
            input[9] = 1000;
            input[10] = -2000;
            input[11] = -3000;
            input[12] = 5000;
            input[13] = 2000;
            input[14] = -1000;
            input[15] = -3000;


        }  /* end Init() */
main    C:\AN168\code\trace.c
```

Figure 20

Note

Source code tracking from the Analysis window also causes source code tracking to occur in the RVD code window. You can bring the PC back into focus by right-clicking in the RVD code window and selecting "Scope to PC". This has no effect on the Analysis window.

If you scroll through the entire trace buffer, you will see that over 3000 samples are captured. Most of these are attributed to the printf() for printing the message "Entering main control loop". The ARM libraries are supplied only in object form. If you try and view the source code of traced library code, you will see the message "no source available" in the Analysis window.

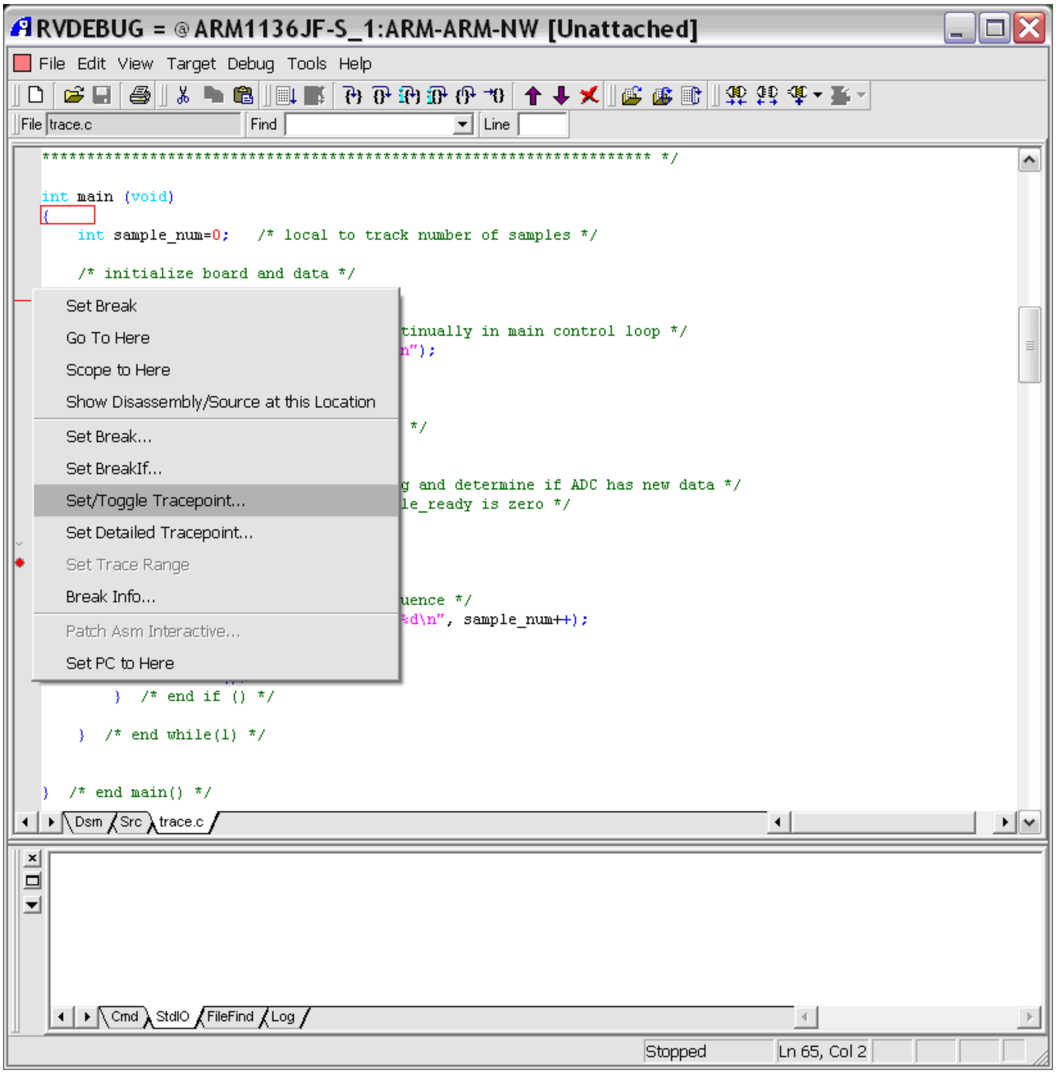
5 Trace Start and Stop Points

Trace Start and Stop points are used to define points in your code where tracing will start and stop. They are most useful for reducing the amount of trace which can be generated from Auto Tracing and they allow one to focus on specific areas of program execution.

5.1 Using a Trace Start Point

Trace Start points are used to initiate trace capture and do not require a matching Trace Stop point. For instance, to bypass the capture of the application startup code, you could place a Trace Start point at the call to Init() and use a breakpoint to end the trace capture.

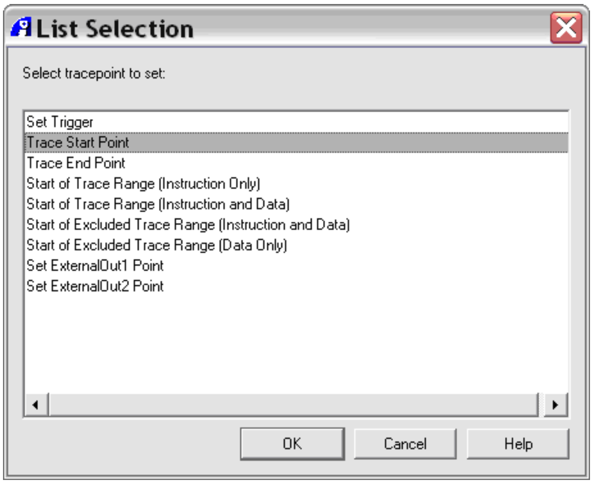
Place the cursor in the left gutter of the Init() line in the source window, click the left or right mouse button and select Set/Toggle Tracepoint as shown below.



```
******************************************************************* */

int main (void)
{
    int sample_num=0;   /* local to track number of samples */

    /* initialize board and data */

                                        tinually in main control loop */
                                        n");

                                         */

                                        g and determine if ADC has new data */
                                        le_ready is zero */

                                        uence */
                                        %d\n", sample_num++);

        }  /* end if () */

    }  /* end while(1) */

}  /* end main() */
```

Figure 21

From the List Selection dialogue, select "Trace Start Point" and click "OK" as shown below.

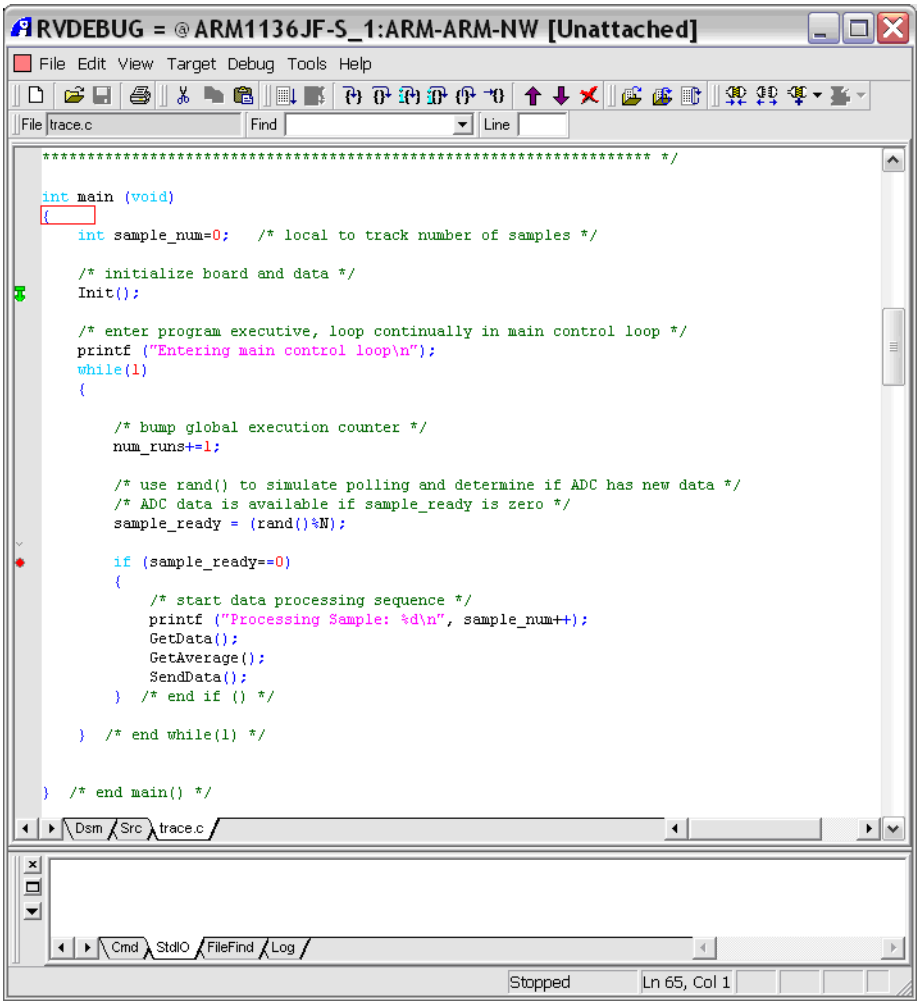


Select tracepoint to set:

- Set Trigger
- Trace Start Point
- Trace End Point
- Start of Trace Range (Instruction Only)
- Start of Trace Range (Instruction and Data)
- Start of Excluded Trace Range (Instruction and Data)
- Start of Excluded Trace Range (Data Only)
- Set ExternalOut1 Point
- Set ExternalOut2 Point

Figure 22

This operation will place a Trace Start point on the call to Init(). The Trace Start point appears in the gutter as a green downward arrow as shown below.


RVDEBUG = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Target Debug Tools Help
File trace.c
Find
Line
Dsm
Src
trace.c
Cmd
StdIO
FileFind
Log
Stopped
Ln 65, Col 1

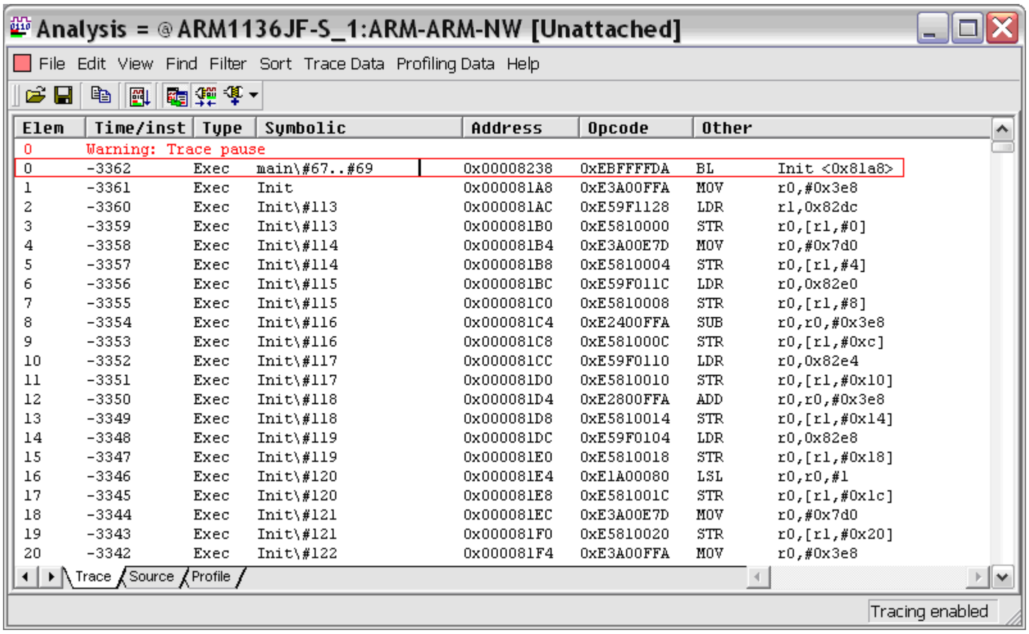
```
********************************************************************** */

int main (void)
{
    int sample_num=0;   /* local to track number of samples */

    /* initialize board and data */
    Init();

    /* enter program executive, loop continually in main control loop */
    printf ("Entering main control loop\n");
    while(1)
    {

        /* bump global execution counter */
        num_runs+=1;

        /* use rand() to simulate polling and determine if ADC has new data */
        /* ADC data is available if sample_ready is zero */
        sample_ready = (rand()%N);

        if (sample_ready==0)
        {
            /* start data processing sequence */
            printf ("Processing Sample: %d\n", sample_num++);
            GetData();
            GetAverage();
            SendData();
        }  /* end if () */

    }  /* end while(1) */


}  /* end main() */
```

Figure 23

Reload the image using the "Target - Reload Image to Target" menu. Observe that reloading the image from this menu retains your breakpoint and tracepoint. This reload operation will be used throughout the Application Note.

Run to the breakpoint. Notice that the trace capture is the same trace as that from Section 4.2. Namely, the application startup code is not traced, and tracing begins with the BL Init instruction at address 0x8238.

There is one minor difference in the captures. This capture begins with "Warning: Trace pause" to indicate that the target was running when trace became activated from executing the Trace Start point. The capture made with Auto Tracing in Section 4.2 starts with "Warning: Debug state" to indicate that the target was in debug mode (halted) just before tracing began.

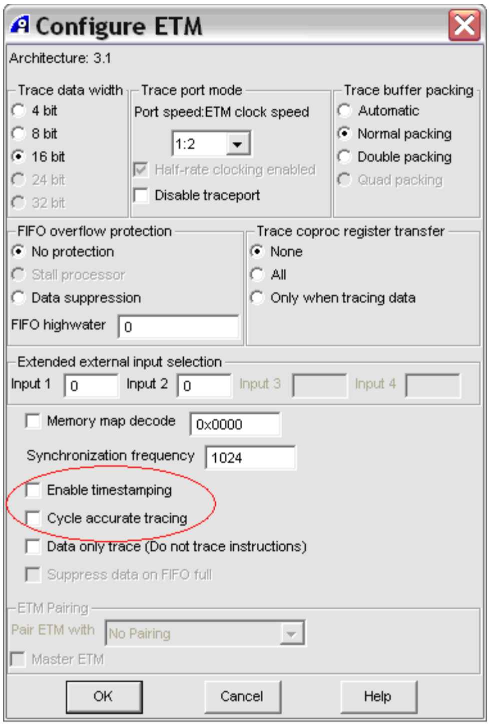


Elem	Time/inst	Type	Symbolic	Address	Opcode	Other	
0	Warning: Trace pause						
0	-3362	Exec	main\#67..#69	0x00008238	0xEBFFFFDA	BL	Init <0x81a8>
1	-3361	Exec	Init	0x000081A8	0xE3A00FFA	MOV	r0,#0x3e8
2	-3360	Exec	Init\#113	0x000081AC	0xE59F1128	LDR	r1,0x82dc
3	-3359	Exec	Init\#113	0x000081B0	0xE5810000	STR	r0,[r1,#0]
4	-3358	Exec	Init\#114	0x000081B4	0xE3A00E7D	MOV	r0,#0x7d0
5	-3357	Exec	Init\#114	0x000081B8	0xE5810004	STR	r0,[r1,#4]
6	-3356	Exec	Init\#115	0x000081BC	0xE59F011C	LDR	r0,0x82e0
7	-3355	Exec	Init\#115	0x000081C0	0xE5810008	STR	r0,[r1,#8]
8	-3354	Exec	Init\#116	0x000081C4	0xE2400FFA	SUB	r0,r0,#0x3e8
9	-3353	Exec	Init\#116	0x000081C8	0xE581000C	STR	r0,[r1,#0xc]
10	-3352	Exec	Init\#117	0x000081CC	0xE59F0110	LDR	r0,0x82e4
11	-3351	Exec	Init\#117	0x000081D0	0xE5810010	STR	r0,[r1,#0x10]
12	-3350	Exec	Init\#118	0x000081D4	0xE2800FFA	ADD	r0,r0,#0x3e8
13	-3349	Exec	Init\#118	0x000081D8	0xE5810014	STR	r0,[r1,#0x14]
14	-3348	Exec	Init\#119	0x000081DC	0xE59F0104	LDR	r0,0x82e8
15	-3347	Exec	Init\#119	0x000081E0	0xE5810018	STR	r0,[r1,#0x18]
16	-3346	Exec	Init\#120	0x000081E4	0xE1A00080	LSL	r0,r0,#1
17	-3345	Exec	Init\#120	0x000081E8	0xE581001C	STR	r0,[r1,#0x1c]
18	-3344	Exec	Init\#121	0x000081EC	0xE3A00E7D	MOV	r0,#0x7d0
19	-3343	Exec	Init\#121	0x000081F0	0xE5810020	STR	r0,[r1,#0x20]
20	-3342	Exec	Init\#122	0x000081F4	0xE3A00FFA	MOV	r0,#0x3e8

Figure 24

5.2 Using Trace Start/Stop Points to Time Function Execution

A Trace Start and Trace Stop point pair can be used to determine the execution time of a function. To obtain timing information from trace, either Cycle Accurate trace or Timestamping must be enabled. These selections are made from the ETM Configuration dialogue. By default both options are disabled.



Configure ETM

Architecture: 3.1

Trace data width
○ 4 bit
○ 8 bit
● 16 bit
○ 24 bit
○ 32 bit

Trace port mode
Port speed:ETM clock speed
1:2
☑ Half-rate clocking enabled
□ Disable traceport

Trace buffer packing
○ Automatic
● Normal packing
○ Double packing
○ Quad packing

FIFO overflow protection
● No protection
○ Stall processor
○ Data suppression
FIFO highwater 0

Trace coproc register transfer
● None
○ All
○ Only when tracing data

Extended external input selection
Input 1 0 Input 2 0 Input 3 Input 4

□ Memory map decode 0x0000

Synchronization frequency 1024

□ Enable timestamping
□ Cycle accurate tracing
□ Data only trace (Do not trace instructions)
□ Suppress data on FIFO full

ETM Pairing
Pair ETM with No Pairing
□ Master ETM

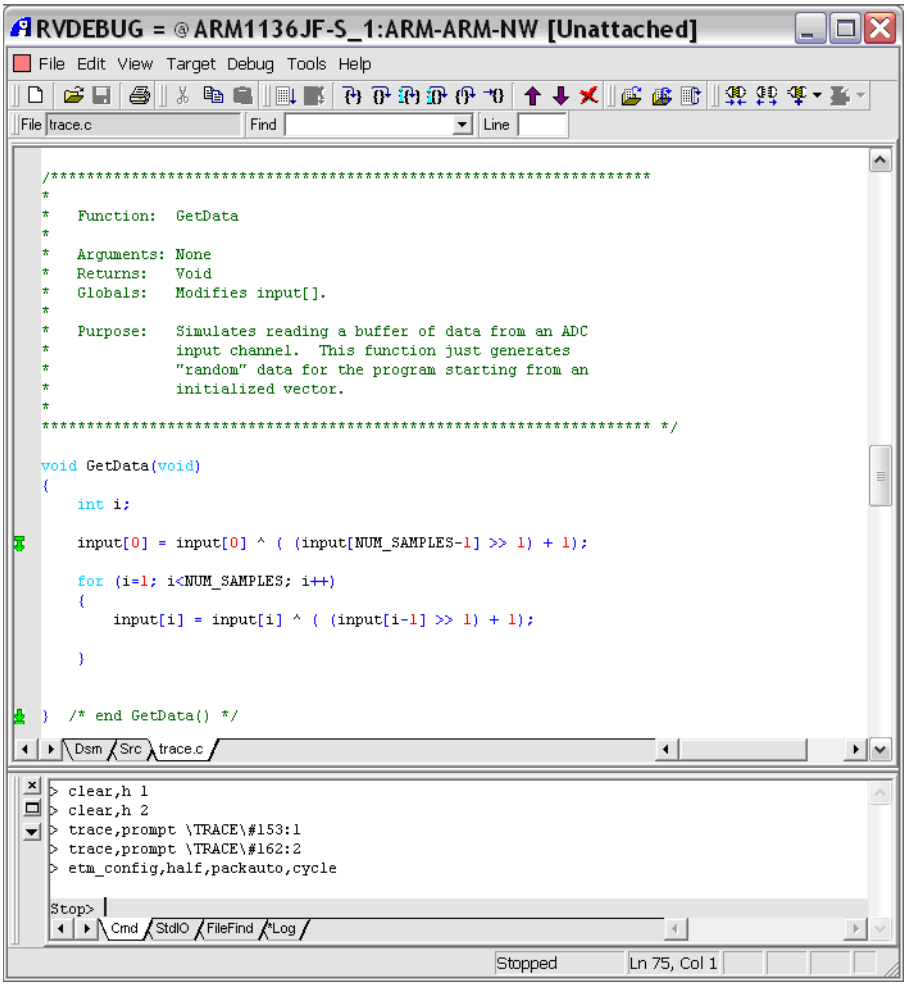
Figure 25

Note

Time stamps are applied by RealView Trace as the trace capture is made, so this option is not available if tracing with an Embedded Trace Buffer.

In this scenario, Cycle Accurate trace is used in conjunction with Trace Start and Stop points to determine the execution time of GetData(). Remove the previous setup and replace it with the following setup:

- Trace Start is placed in GetData() at the line which sets input[0].
- Trace Stop is placed at the closing brace of GetData().
- Enable Cycle Accurate tracing from the ETM Configuration dialogue.

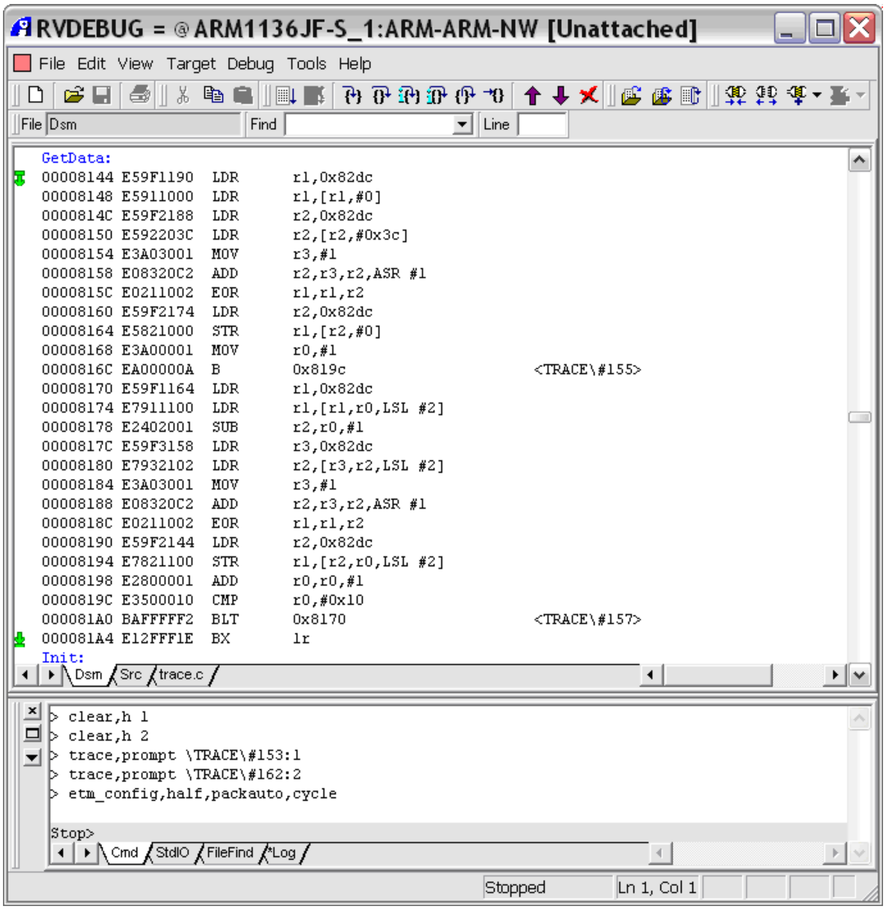

RVDEBUG = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Target Debug Tools Help
File trace.c
Find
Line
Dsm
Src
trace.c
Cmd
StdIO
FileFind
*Log
Stopped
Ln 75, Col 1

```
/**********************************************************************
*
*   Function:  GetData
*
*   Arguments: None
*   Returns:   Void
*   Globals:   Modifies input[].
*
*   Purpose:   Simulates reading a buffer of data from an ADC
*              input channel.  This function just generates
*              "random" data for the program starting from an
*              initialized vector.
*
********************************************************************** */

void GetData(void)
{
    int i;

    input[0] = input[0] ^ ( (input[NUM_SAMPLES-1] >> 1) + 1);

    for (i=1; i<NUM_SAMPLES; i++)
    {
        input[i] = input[i] ^ ( (input[i-1] >> 1) + 1);

    }


}  /* end GetData() */
```

```
> clear,h 1
> clear,h 2
> trace,prompt \TRACE\#153:1
> trace,prompt \TRACE\#162:2
> etm_config,half,packauto,cycle

Stop>
```

Figure 26

The Disassembly source view (from the Dsm tab) of GetData() and the trace points are shown below.


RVDEBUG = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Target Debug Tools Help
File Dsm
Find
Line
Dsm Src trace.c
Cmd StdIO FileFind *Log
Stopped
Ln 1, Col 1

```
GetData:
00008144 E59F1190  LDR      r1,0x82dc
00008148 E5911000  LDR      r1,[r1,#0]
0000814C E59F2188  LDR      r2,0x82dc
00008150 E592203C  LDR      r2,[r2,#0x3c]
00008154 E3A03001  MOV      r3,#1
00008158 E08320C2  ADD      r2,r3,r2,ASR #1
0000815C E0211002  EOR      r1,r1,r2
00008160 E59F2174  LDR      r2,0x82dc
00008164 E5821000  STR      r1,[r2,#0]
00008168 E3A00001  MOV      r0,#1
0000816C EA00000A  B        0x819c                    <TRACE\#155>
00008170 E59F1164  LDR      r1,0x82dc
00008174 E7911100  LDR      r1,[r1,r0,LSL #2]
00008178 E2402001  SUB      r2,r0,#1
0000817C E59F3158  LDR      r3,0x82dc
00008180 E7932102  LDR      r2,[r3,r2,LSL #2]
00008184 E3A03001  MOV      r3,#1
00008188 E08320C2  ADD      r2,r3,r2,ASR #1
0000818C E0211002  EOR      r1,r1,r2
00008190 E59F2144  LDR      r2,0x82dc
00008194 E7821100  STR      r1,[r2,r0,LSL #2]
00008198 E2800001  ADD      r0,r0,#1
0000819C E3500010  CMP      r0,#0x10
000081A0 BAFFFFF2  BLT      0x8170                    <TRACE\#157>
000081A4 E12FFF1E  BX       lr
Init:
```

```
> clear,h 1
> clear,h 2
> trace,prompt \TRACE\#153:1
> trace,prompt \TRACE\#162:2
> etm_config,half,packauto,cycle

Stop>
```

Figure 27

Reload the program using the "Target - Reload Image to Target" menu such that your trace points are retained. Run the program for a few seconds and then stop it. Observe the program output in the RVD StdIO output window:

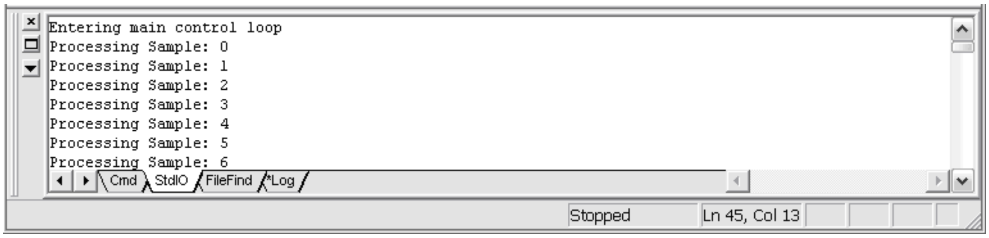
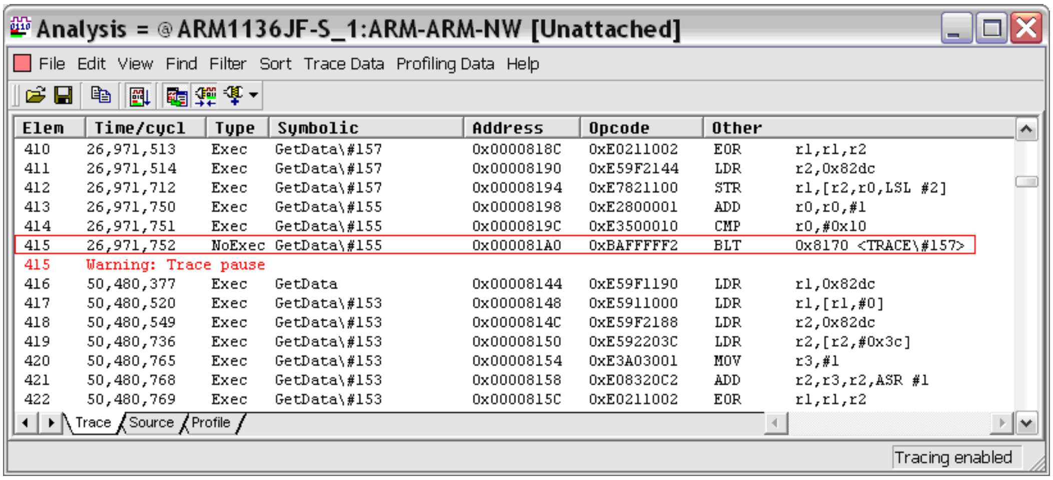

```
Entering main control loop
Processing Sample: 0
Processing Sample: 1
Processing Sample: 2
Processing Sample: 3
Processing Sample: 4
Processing Sample: 5
Processing Sample: 6
```

Figure 28

Open the Trace Analysis window, scroll through the trace capture and observe the following:

- Only instructions from GetData() are traced.
- A continuous trace segment begins with the LDR instruction at 0x8144 and ends with the BLT instruction at 0x81A0.
- Just before trace is paused, the final BLT instruction is not executed and is marked "NoExec" in the Type column.
- The BX LR instruction which returns from GetData() is not traced due to the placement of the Trace Stop point (see the previous Dsm source view).

- Trace discontinuities are marked by "Warning: Trace Pause". This message represents the execution between the exit of GetData() and the entry of GetData() when trace is not captured.
- After the trace pause, tracing always starts again from the Trace Start point.


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

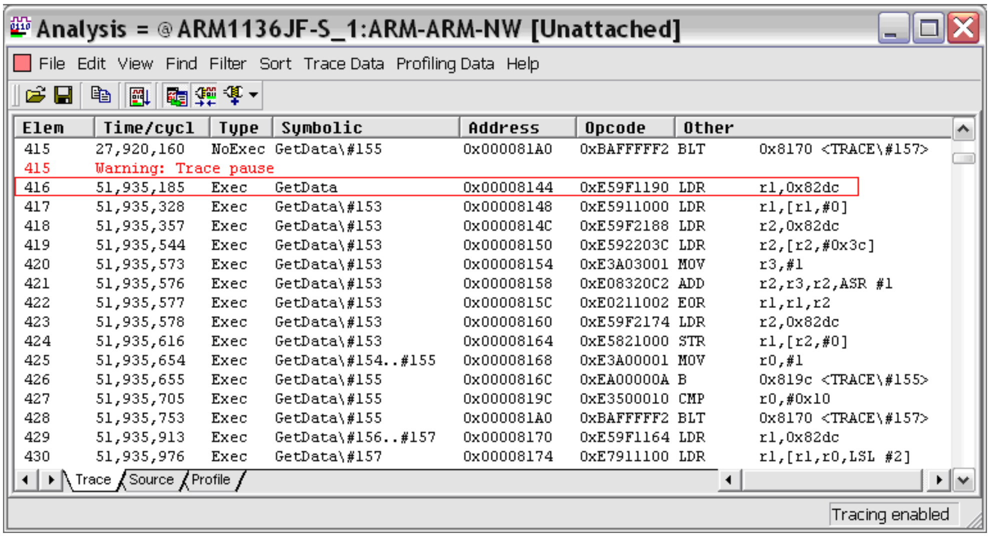
```
Elem  Time/cycl   Type    Symbolic          Address     Opcode      Other
410   26,971,513  Exec    GetData\#157      0x0000818C  0xE0211002  EOR     r1,r1,r2
411   26,971,514  Exec    GetData\#157      0x00008190  0xE59F2144  LDR     r2,0x82dc
412   26,971,712  Exec    GetData\#157      0x00008194  0xE7821100  STR     r1,[r2,r0,LSL #2]
413   26,971,750  Exec    GetData\#155      0x00008198  0xE2800001  ADD     r0,r0,#1
414   26,971,751  Exec    GetData\#155      0x0000819C  0xE3500010  CMP     r0,#0x10
415   26,971,752  NoExec  GetData\#155      0x000081A0  0xBAFFFFF2  BLT     0x8170 <TRACE\#157>
415   Warning: Trace pause
416   50,480,377  Exec    GetData           0x00008144  0xE59F1190  LDR     r1,0x82dc
417   50,480,520  Exec    GetData\#153      0x00008148  0xE5911000  LDR     r1,[r1,#0]
418   50,480,549  Exec    GetData\#153      0x0000814C  0xE59F2188  LDR     r2,0x82dc
419   50,480,736  Exec    GetData\#153      0x00008150  0xE592203C  LDR     r2,[r2,#0x3c]
420   50,480,765  Exec    GetData\#153      0x00008154  0xE3A03001  MOV     r3,#1
421   50,480,768  Exec    GetData\#153      0x00008158  0xE08320C2  ADD     r2,r3,r2,ASR #1
422   50,480,769  Exec    GetData\#153      0x0000815C  0xE0211002  EOR     r1,r1,r2
```

Figure 29

Note

The effects of Cycle Accurate trace can be observed in the second trace column, "Time/cycle". By default RVD will display the time in cycles. This column only contains valid timing information if Cycle Accurate trace or Timestamping is enabled.

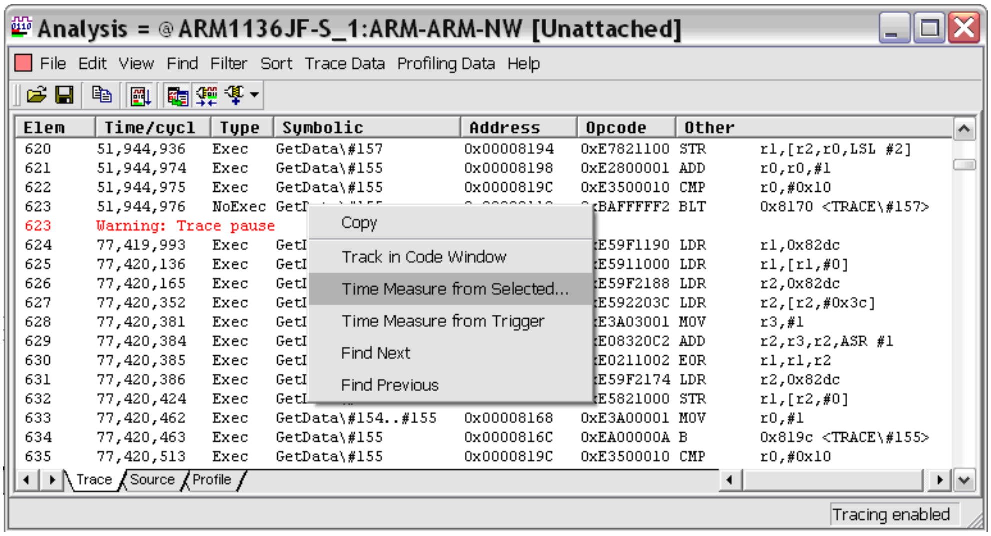
To determine the execution time of GetData(), highlight the instruction after a trace pause message (this will be the instruction at the Trace Start point):


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

```
Elem  Time/cycl   Type    Symbolic             Address     Opcode          Other
415   27,920,160  NoExec  GetData\#155         0x000081A0  0xBAFFFFF2 BLT  0x8170 <TRACE\#157>
415   Warning: Trace pause
416   51,935,185  Exec    GetData              0x00008144  0xE59F1190 LDR  r1,0x82dc
417   51,935,328  Exec    GetData\#153         0x00008148  0xE5911000 LDR  r1,[r1,#0]
418   51,935,357  Exec    GetData\#153         0x0000814C  0xE59F2188 LDR  r2,0x82dc
419   51,935,544  Exec    GetData\#153         0x00008150  0xE592203C LDR  r2,[r2,#0x3c]
420   51,935,573  Exec    GetData\#153         0x00008154  0xE3A03001 MOV  r3,#1
421   51,935,576  Exec    GetData\#153         0x00008158  0xE08320C2 ADD  r2,r3,r2,ASR #1
422   51,935,577  Exec    GetData\#153         0x0000815C  0xE0211002 EOR  r1,r1,r2
423   51,935,578  Exec    GetData\#153         0x00008160  0xE59F2174 LDR  r2,0x82dc
424   51,935,616  Exec    GetData\#153         0x00008164  0xE5821000 STR  r1,[r2,#0]
425   51,935,654  Exec    GetData\#154..#155   0x00008168  0xE3A00001 MOV  r0,#1
426   51,935,655  Exec    GetData\#155         0x0000816C  0xEA00000A B    0x819c <TRACE\#155>
427   51,935,705  Exec    GetData\#155         0x0000819C  0xE3500010 CMP  r0,#0x10
428   51,935,753  Exec    GetData\#155         0x000081A0  0xBAFFFFF2 BLT  0x8170 <TRACE\#157>
429   51,935,913  Exec    GetData\#156..#157   0x00008170  0xE59F1164 LDR  r1,0x82dc
430   51,935,976  Exec    GetData\#157         0x00008174  0xE7911100 LDR  r1,[r1,r0,LSL #2]
```

Figure 30

Without clicking in the Analysis window, scroll down to the next Trace Pause message. Right-click the BLT instruction just above the Trace Pause message and select "Time Measured from Selected…":



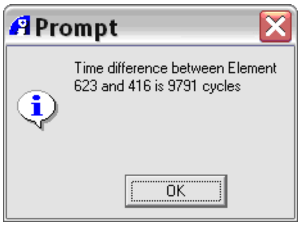
Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]

File Edit View Find Filter Sort Trace Data Profiling Data Help

```
Elem   Time/cycl   Type    Symbolic           Address      Opcode        Other
620    51,944,936  Exec    GetData\#157       0x00008194   0xE7821100 STR     r1,[r2,r0,LSL #2]
621    51,944,974  Exec    GetData\#155       0x00008198   0xE2800001 ADD     r0,r0,#1
622    51,944,975  Exec    GetData\#155       0x0000819C   0xE3500010 CMP     r0,#0x10
623    51,944,976  NoExec  GetI                            BAFFFFF2 BLT       0x8170 <TRACE\#157>
623    Warning: Trace pause
624    77,419,993  Exec    GetI                            E59F1190 LDR       r1,0x82dc
625    77,420,136  Exec    GetI                            E5911000 LDR       r1,[r1,#0]
626    77,420,165  Exec    GetI                            E59F2188 LDR       r2,0x82dc
627    77,420,352  Exec    GetI                            E592203C LDR       r2,[r2,#0x3c]
628    77,420,381  Exec    GetI                            E3A03001 MOV       r3,#1
629    77,420,384  Exec    GetI                            E08320C2 ADD       r2,r3,r2,ASR #1
630    77,420,385  Exec    GetI                            E0211002 EOR       r1,r1,r2
631    77,420,386  Exec    GetI                            E59F2174 LDR       r2,0x82dc
632    77,420,424  Exec    GetI                            E5821000 STR       r1,[r2,#0]
633    77,420,462  Exec    GetData\#154..#155 0x00008168   0xE3A00001 MOV     r0,#1
634    77,420,463  Exec    GetData\#155       0x0000816C   0xEA00000A B       0x819c <TRACE\#155>
635    77,420,513  Exec    GetData\#155       0x0000819C   0xE3500010 CMP     r0,#0x10
```

Figure 31

RVD will return the time difference between the two trace samples. In this instance, the time difference is the execution time of GetData(). The same measurement can be obtained by manually subtracting the time of element 416 (51,935,185) from the time of element 623 (51,944,976):



Prompt

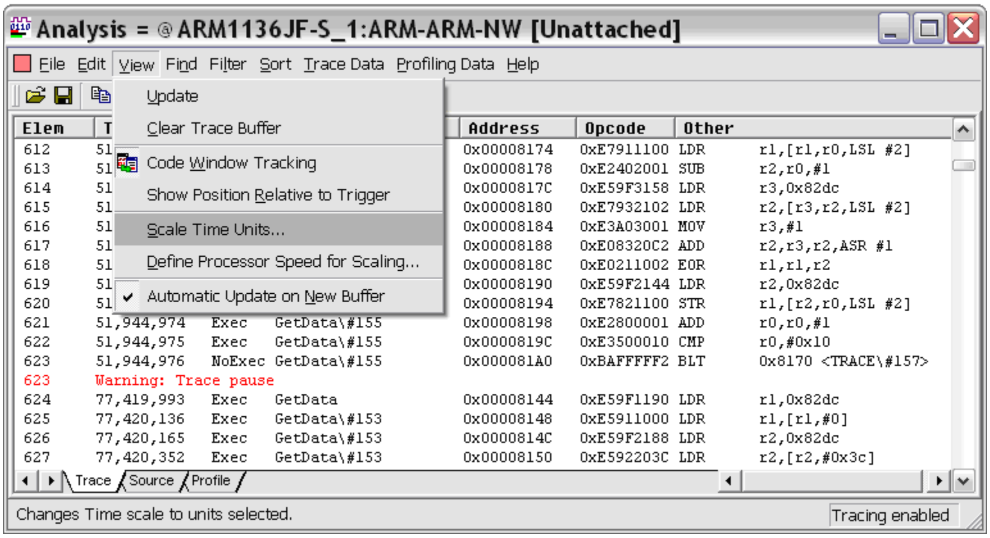
Time difference between Element 623 and 416 is 9791 cycles

Figure 32

Note

Execution time is hardware dependent. Your measured execution time for GetData() may be different.

The time unit of the trace elements and the computed time difference can also be displayed in absolute time (i.e., µS). This selection is made from the "View - Scale Time Units" menu from the Analysis Window:

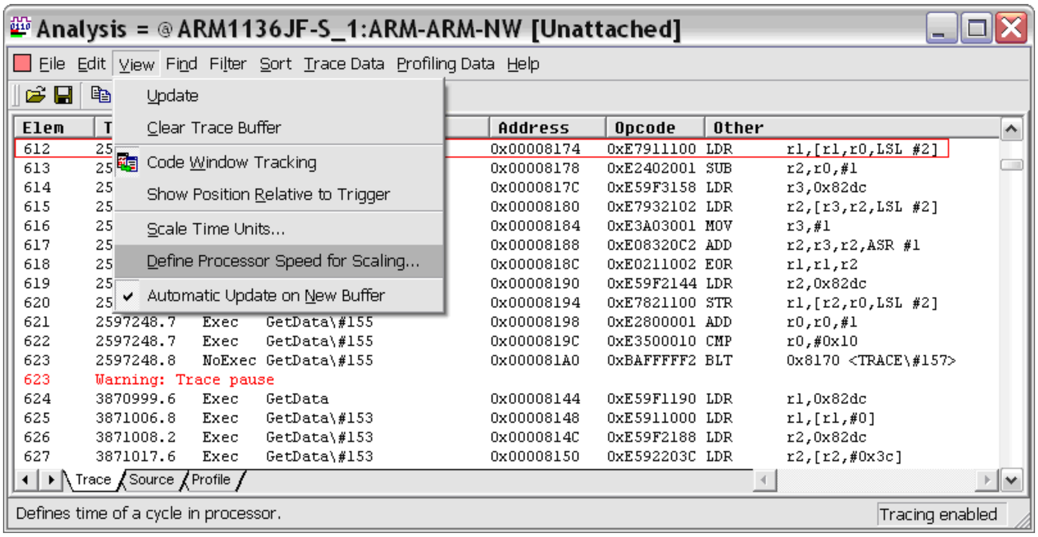

Analysis = @ ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Update
Clear Trace Buffer
Code Window Tracking
Show Position Relative to Trigger
Scale Time Units...
Define Processor Speed for Scaling...
✓ Automatic Update on New Buffer
Trace
Source
Profile
Tracing enabled

```
Elem  T                                  Address     Opcode          Other
612   51                                 0x00008174  0xE7911100 LDR  r1,[r1,r0,LSL #2]
613   51                                 0x00008178  0xE2402001 SUB  r2,r0,#1
614   51                                 0x0000817C  0xE59F3158 LDR  r3,0x82dc
615   51                                 0x00008180  0xE7932102 LDR  r2,[r3,r2,LSL #2]
616   51                                 0x00008184  0xE3A03001 MOV  r3,#1
617   51                                 0x00008188  0xE08320C2 ADD  r2,r3,r2,ASR #1
618   51                                 0x0000818C  0xE0211002 EOR  r1,r1,r2
619   51                                 0x00008190  0xE59F2144 LDR  r2,0x82dc
620   51                                 0x00008194  0xE7821100 STR  r1,[r2,r0,LSL #2]
621   51,944,974  Exec   GetData\#155    0x00008198  0xE2800001 ADD  r0,r0,#1
622   51,944,975  Exec   GetData\#155    0x0000819C  0xE3500010 CMP  r0,#0x10
623   51,944,976  NoExec GetData\#155    0x000081A0  0xBAFFFFF2 BLT  0x8170 <TRACE\#157>
623   Warning: Trace pause
624   77,419,993  Exec   GetData         0x00008144  0xE59F1190 LDR  r1,0x82dc
625   77,420,136  Exec   GetData\#153    0x00008148  0xE5911000 LDR  r1,[r1,#0]
626   77,420,165  Exec   GetData\#153    0x0000814C  0xE59F2188 LDR  r2,0x82dc
627   77,420,352  Exec   GetData\#153    0x00008150  0xE592203C LDR  r2,[r2,#0x3c]
```

Changes Time scale to units selected.

Figure 33

In order to convert cycles to absolute time units you must enter a processor speed from the "View - Define Processor Speed for Scaling…" menu. The default processor speed is 20 MHz. Any changes made to the time units and processor speed take immediate effect on the trace capture.

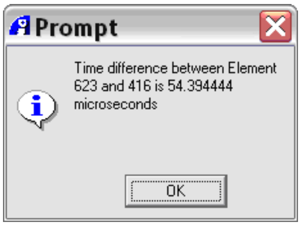

Analysis = @ ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Update
Clear Trace Buffer
Code Window Tracking
Show Position Relative to Trigger
Scale Time Units...
Define Processor Speed for Scaling...
✓ Automatic Update on New Buffer
Trace
Source
Profile
Tracing enabled

```
Elem  T                                  Address     Opcode          Other
612   25                                 0x00008174  0xE7911100 LDR  r1,[r1,r0,LSL #2]
613   25                                 0x00008178  0xE2402001 SUB  r2,r0,#1
614   25                                 0x0000817C  0xE59F3158 LDR  r3,0x82dc
615   25                                 0x00008180  0xE7932102 LDR  r2,[r3,r2,LSL #2]
616   25                                 0x00008184  0xE3A03001 MOV  r3,#1
617   25                                 0x00008188  0xE08320C2 ADD  r2,r3,r2,ASR #1
618   25                                 0x0000818C  0xE0211002 EOR  r1,r1,r2
619   25                                 0x00008190  0xE59F2144 LDR  r2,0x82dc
620   25                                 0x00008194  0xE7821100 STR  r1,[r2,r0,LSL #2]
621   2597248.7  Exec   GetData\#155     0x00008198  0xE2800001 ADD  r0,r0,#1
622   2597248.7  Exec   GetData\#155     0x0000819C  0xE3500010 CMP  r0,#0x10
623   2597248.8  NoExec GetData\#155     0x000081A0  0xBAFFFFF2 BLT  0x8170 <TRACE\#157>
623   Warning: Trace pause
624   3870999.6  Exec   GetData          0x00008144  0xE59F1190 LDR  r1,0x82dc
625   3871006.8  Exec   GetData\#153     0x00008148  0xE5911000 LDR  r1,[r1,#0]
626   3871008.2  Exec   GetData\#153     0x0000814C  0xE59F2188 LDR  r2,0x82dc
627   3871017.6  Exec   GetData\#153     0x00008150  0xE592203C LDR  r2,[r2,#0x3c]
```

Defines time of a cycle in processor.

Figure 34

Set the processor speed to your core speed and perform another time measurement using absolute time. If you are using a 180 MHz clock, the 9791 cycles equate to 54.39 µs.



Time difference between Element 623 and 416 is 54.394444 microseconds

Figure 35

In your real application, Trace Start and Stop points may be particularly useful if you are interested in timing a function that is interrupted by an interrupt service routine (ISR), because the ISR will also be traced. This allows you to see the occurrence of ISR disruptions and determine the absolute maximum execution time for a function.

6 Trace Ranges

Trace Ranges provide an alternate method to Trace Start / Stop points for controlling the region of your program that is traced. Trace Ranges allow you to define a continuous address range for tracing. If the program counter (PC) resides within the defined address range, trace is captured. If the PC is outside the address range, trace is not captured.

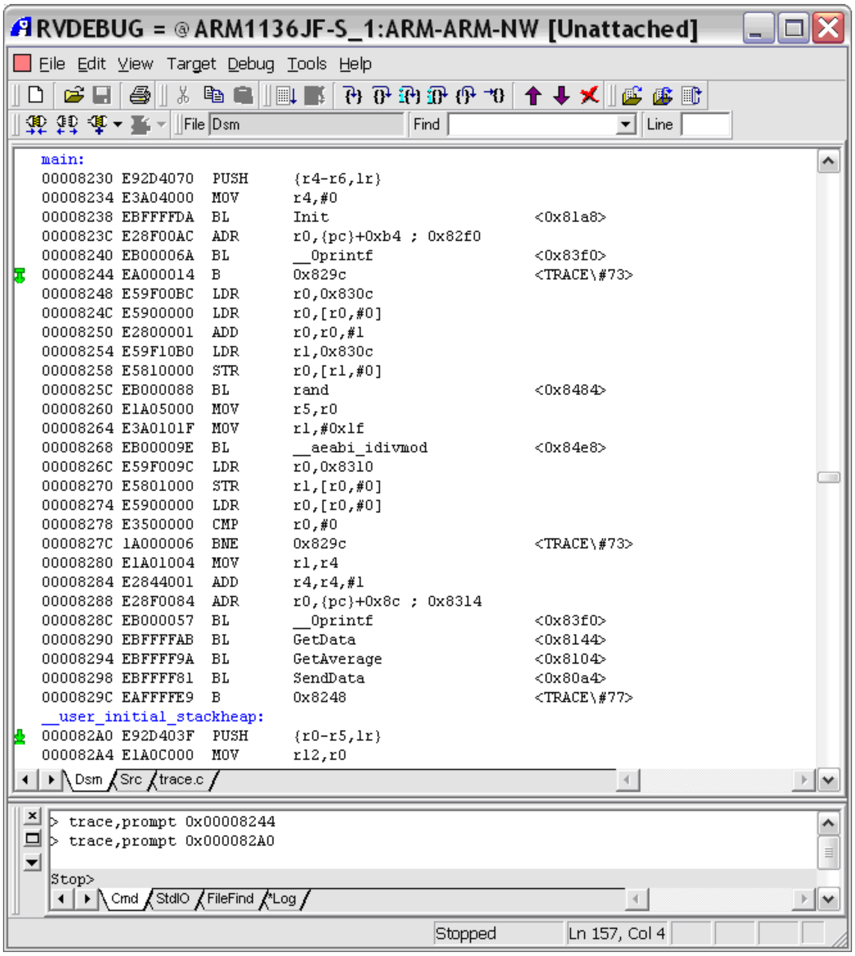
6.1 Comparing a Trace Range with Trace Start / Stop Points

To observe the difference between tracing with Trace Ranges and Start / Stop Points, the while(1) loop of the sample program will be traced using each method.

Tracing while(1) with Trace Ranges

Reload TRACE.AXF and remove all trace points and breakpoints. Using the Disassembly source view, place the trace range:

- Start of Trace Range (Instruction Only) at 0x8244 (on B 0x829C)
- End of Trace Range (Instruction Only) at 0x82A0 (just after B 0x8248)


RVDEBUG = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Target Debug Tools Help
File Dsm
Find
Line
Dsm
Src
trace.c
Cmd
StdIO
FileFind
Log
Stopped
Ln 157, Col 4

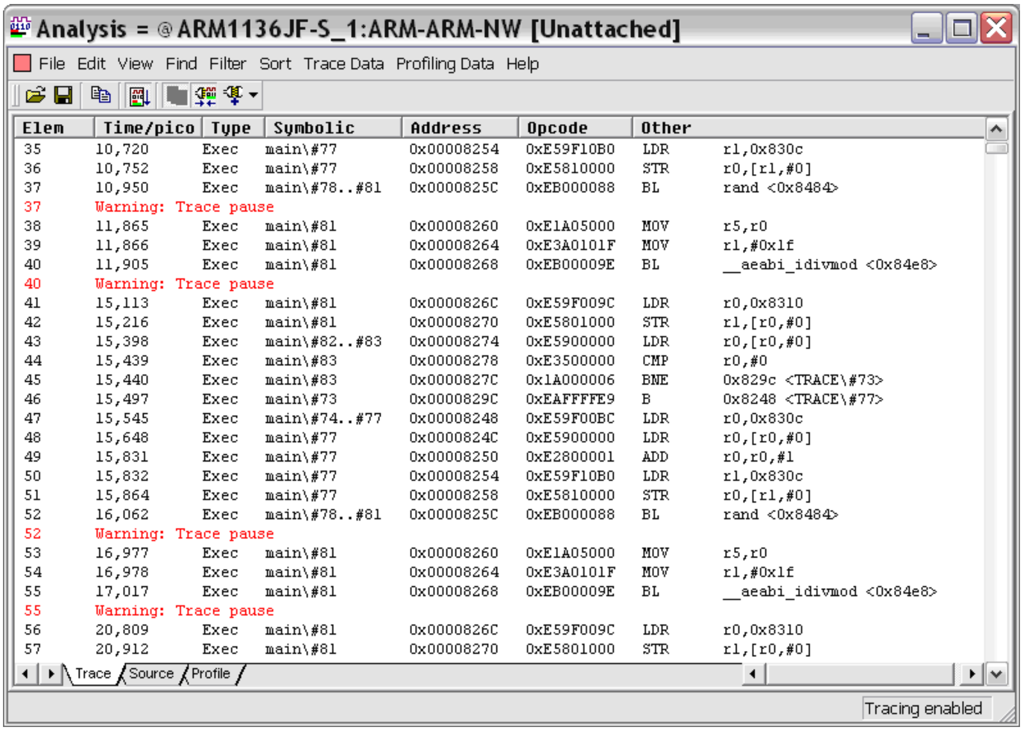
```
main:
00008230 E92D4070  PUSH     {r4-r6,lr}
00008234 E3A04000  MOV      r4,#0
00008238 EBFFFFDA  BL       Init                         <0x81a8>
0000823C E28F00AC  ADR      r0,{pc}+0xb4 ; 0x82f0
00008240 EB00006A  BL       __0printf                    <0x83f0>
00008244 EA000014  B        0x829c                       <TRACE\#73>
00008248 E59F00BC  LDR      r0,0x830c
0000824C E5900000  LDR      r0,[r0,#0]
00008250 E2800001  ADD      r0,r0,#1
00008254 E59F10B0  LDR      r1,0x830c
00008258 E5810000  STR      r0,[r1,#0]
0000825C EB000088  BL       rand                         <0x8484>
00008260 E1A05000  MOV      r5,r0
00008264 E3A0101F  MOV      r1,#0x1f
00008268 EB00009E  BL       __aeabi_idivmod              <0x84e8>
0000826C E59F009C  LDR      r0,0x8310
00008270 E5801000  STR      r1,[r0,#0]
00008274 E5900000  LDR      r0,[r0,#0]
00008278 E3500000  CMP      r0,#0
0000827C 1A000006  BNE      0x829c                       <TRACE\#73>
00008280 E1A01004  MOV      r1,r4
00008284 E2844001  ADD      r4,r4,#1
00008288 E28F0084  ADR      r0,{pc}+0x8c ; 0x8314
0000828C EB000057  BL       __0printf                    <0x83f0>
00008290 EBFFFFAB  BL       GetData                      <0x8144>
00008294 EBFFFF9A  BL       GetAverage                   <0x8104>
00008298 EBFFFF81  BL       SendData                     <0x80a4>
0000829C EAFFFFE9  B        0x8248                       <TRACE\#77>
__user_initial_stackheap:
000082A0 E92D403F  PUSH     {r0-r5,lr}
000082A4 E1A0C000  MOV      r12,r0
```

```
> trace,prompt 0x00008244
> trace,prompt 0x000082A0

Stop>
```

Figure 36

Run the program for several seconds and stop it. Inspect the Analysis window and note the following:

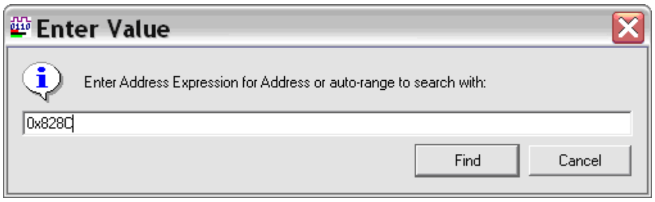
- Only instructions residing within the address range are traced.
- A "Warning: Trace pause" message is generated every time a function is called from the while(1) loop.
- Most of the trace capture consists of the same four C lines, which call rand() and check the value of sample_ready.


Analysis = @ ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

```
Elem  Time/pico  Type  Symbolic      Address     Opcode      Other
35    10,720     Exec  main\#77      0x00008254  0xE59F10B0  LDR   r1,0x830c
36    10,752     Exec  main\#77      0x00008258  0xE5810000  STR   r0,[r1,#0]
37    10,950     Exec  main\#78..#81 0x0000825C  0xEB000088  BL    rand <0x8484>
37    Warning: Trace pause
38    11,865     Exec  main\#81      0x00008260  0xE1A05000  MOV   r5,r0
39    11,866     Exec  main\#81      0x00008264  0xE3A0101F  MOV   r1,#0x1f
40    11,905     Exec  main\#81      0x00008268  0xEB00009E  BL    __aeabi_idivmod <0x84e8>
40    Warning: Trace pause
41    15,113     Exec  main\#81      0x0000826C  0xE59F009C  LDR   r0,0x8310
42    15,216     Exec  main\#81      0x00008270  0xE5801000  STR   r1,[r0,#0]
43    15,398     Exec  main\#82..#83 0x00008274  0xE5900000  LDR   r0,[r0,#0]
44    15,439     Exec  main\#83      0x00008278  0xE3500000  CMP   r0,#0
45    15,440     Exec  main\#83      0x0000827C  0x1A000006  BNE   0x829c <TRACE\#73>
46    15,497     Exec  main\#73      0x0000829C  0xEAFFFFE9  B     0x8248 <TRACE\#77>
47    15,545     Exec  main\#74..#77 0x00008248  0xE59F00BC  LDR   r0,0x830c
48    15,648     Exec  main\#77      0x0000824C  0xE5900000  LDR   r0,[r0,#0]
49    15,831     Exec  main\#77      0x00008250  0xE2800001  ADD   r0,r0,#1
50    15,832     Exec  main\#77      0x00008254  0xE59F10B0  LDR   r1,0x830c
51    15,864     Exec  main\#77      0x00008258  0xE5810000  STR   r0,[r1,#0]
52    16,062     Exec  main\#78..#81 0x0000825C  0xEB000088  BL    rand <0x8484>
52    Warning: Trace pause
53    16,977     Exec  main\#81      0x00008260  0xE1A05000  MOV   r5,r0
54    16,978     Exec  main\#81      0x00008264  0xE3A0101F  MOV   r1,#0x1f
55    17,017     Exec  main\#81      0x00008268  0xEB00009E  BL    __aeabi_idivmod <0x84e8>
55    Warning: Trace pause
56    20,809     Exec  main\#81      0x0000826C  0xE59F009C  LDR   r0,0x8310
57    20,912     Exec  main\#81      0x00008270  0xE5801000  STR   r1,[r0,#0]
```

Figure 37

Search through the trace capture using the "Find - Find Address Expression" menu from the Analysis Window. Search for 0x828C, the address of the BL __0printf instruction for printing the "Processing Sample" message:

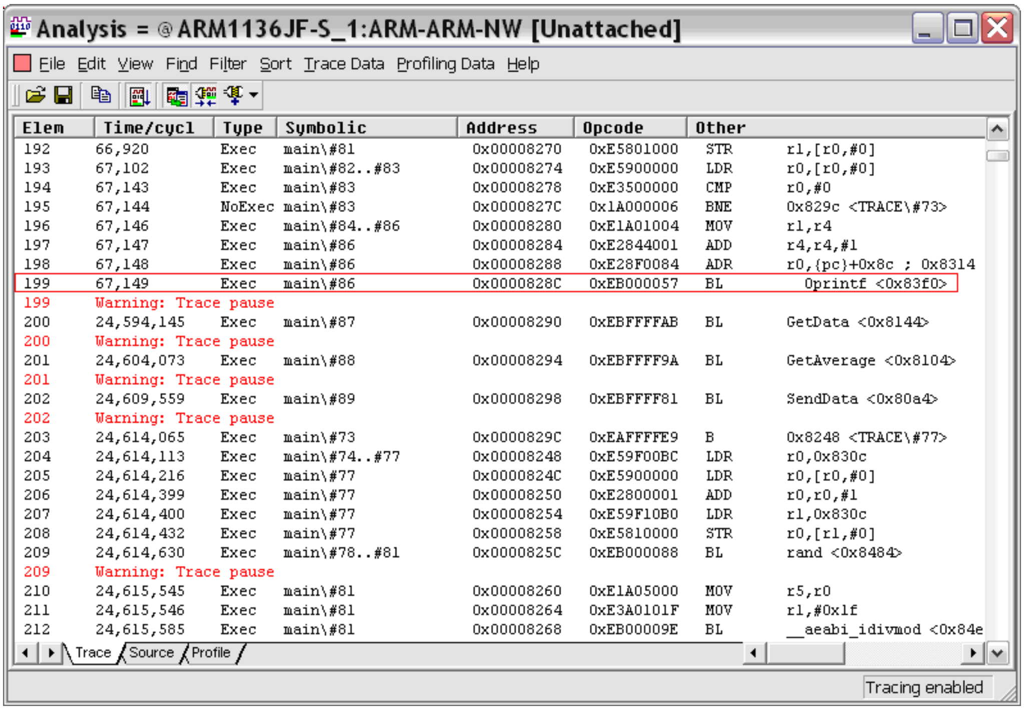


Enter Address Expression for Address or auto-range to search with:

0x828C

Figure 38

You will be brought to the next occurrence in the trace buffer where sample_ready equals 0 and processing occurs. Note that the functions __0printf(), GetData(), GetAverage() and SendData() are not actually traced because they lie outside the active trace range. Discontinuities in the trace capture are marked with "Warning: Trace pause".



Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

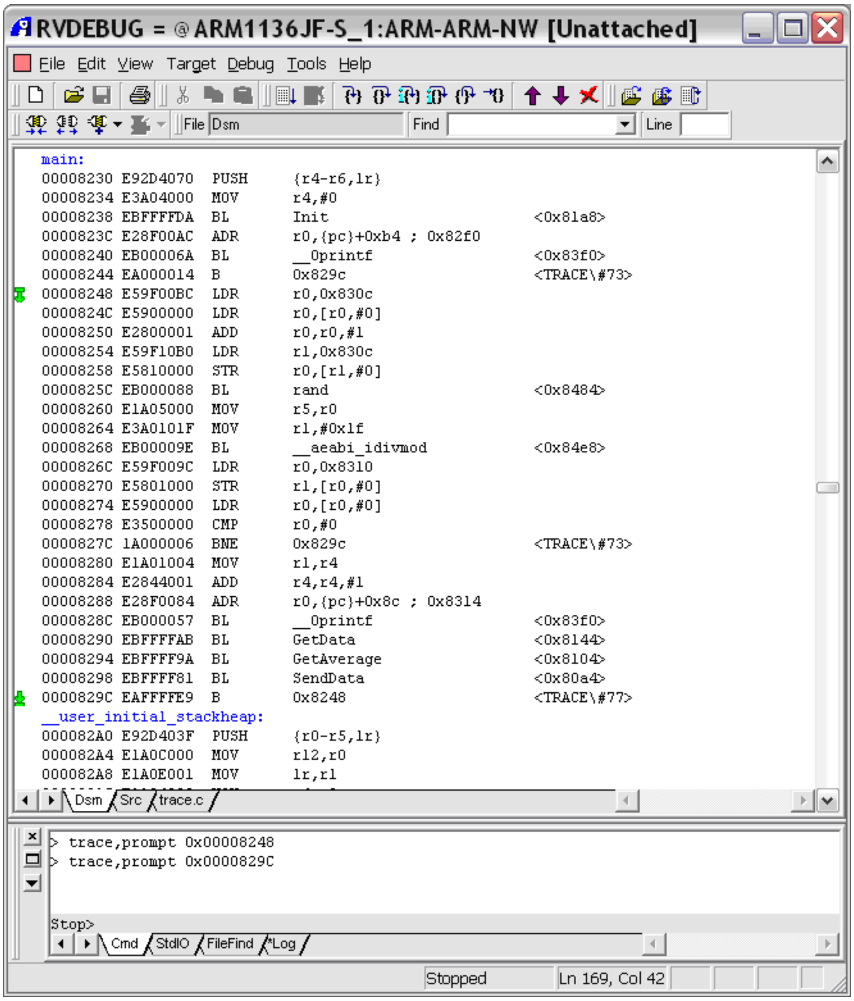
```
Elem  Time/cycl    Type   Symbolic        Address     Opcode      Other
192   66,920       Exec   main\#81        0x00008270  0xE5801000  STR    r1,[r0,#0]
193   67,102       Exec   main\#82..#83   0x00008274  0xE5900000  LDR    r0,[r0,#0]
194   67,143       Exec   main\#83        0x00008278  0xE3500000  CMP    r0,#0
195   67,144       NoExec main\#83        0x0000827C  0x1A000006  BNE    0x829c <TRACE\#73>
196   67,146       Exec   main\#84..#86   0x00008280  0xE1A01004  MOV    r1,r4
197   67,147       Exec   main\#86        0x00008284  0xE2844001  ADD    r4,r4,#1
198   67,148       Exec   main\#86        0x00008288  0xE28F0084  ADR    r0,{pc}+0x8c ; 0x8314
199   67,149       Exec   main\#86        0x0000828C  0xEB000057  BL       0printf <0x83f0>
199   Warning: Trace pause
200   24,594,145   Exec   main\#87        0x00008290  0xEBFFFFAB  BL     GetData <0x8144>
200   Warning: Trace pause
201   24,604,073   Exec   main\#88        0x00008294  0xEBFFFF9A  BL     GetAverage <0x8104>
201   Warning: Trace pause
202   24,609,559   Exec   main\#89        0x00008298  0xEBFFFF81  BL     SendData <0x80a4>
202   Warning: Trace pause
203   24,614,065   Exec   main\#73        0x0000829C  0xEAFFFFE9  B      0x8248 <TRACE\#77>
204   24,614,113   Exec   main\#74..#77   0x00008248  0xE59F00BC  LDR    r0,0x830c
205   24,614,216   Exec   main\#77        0x0000824C  0xE5900000  LDR    r0,[r0,#0]
206   24,614,399   Exec   main\#77        0x00008250  0xE2800001  ADD    r0,r0,#1
207   24,614,400   Exec   main\#77        0x00008254  0xE59F10B0  LDR    r1,0x830c
208   24,614,432   Exec   main\#77        0x00008258  0xE5810000  STR    r0,[r1,#0]
209   24,614,630   Exec   main\#78..#81   0x0000825C  0xEB000088  BL     rand <0x8484>
209   Warning: Trace pause
210   24,615,545   Exec   main\#81        0x00008260  0xE1A05000  MOV    r5,r0
211   24,615,546   Exec   main\#81        0x00008264  0xE3A0101F  MOV    r1,#0x1f
212   24,615,585   Exec   main\#81        0x00008268  0xEB00009E  BL     __aeabi_idivmod <0x84e
```

Figure 39

Tracing while(1) with Trace Start / Stop Points

Reload TRACE.AXF and remove the Trace Range that you set in Section 6.1.1. Using the Disassembly source view, place the Trace Start and Stop points:

- Trace Start at 0x8248 (on LDR r0, 0x830C)
- Trace Stop at 0x829C (on B 0x8248)


RVDEBUG = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Target Debug Tools Help
File Dsm
Find
Line
Dsm
Src
trace.c
Cmd
StdIO
FileFind
Log
Stopped
Ln 169, Col 42

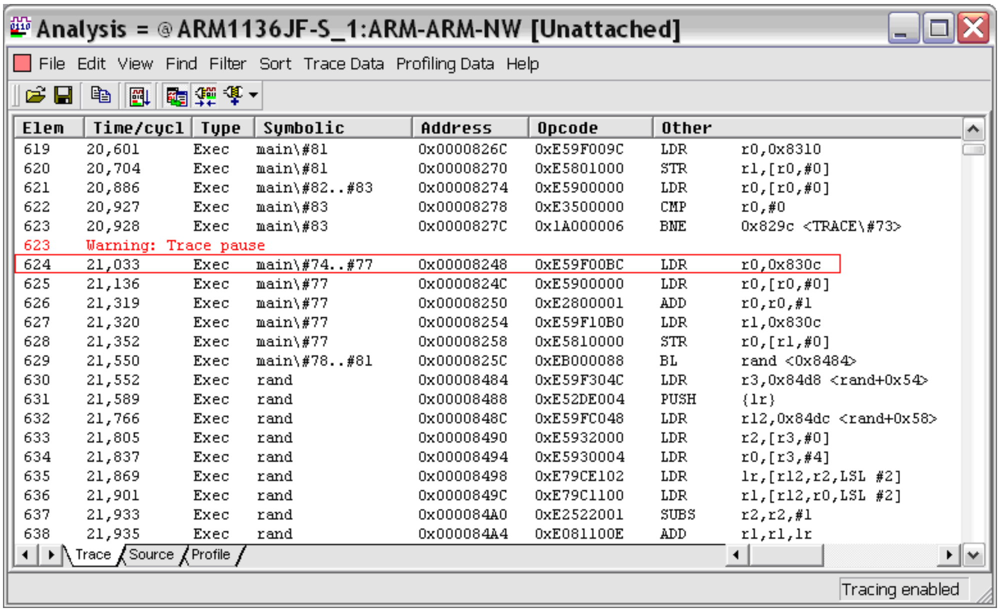
```
main:
00008230 E92D4070  PUSH     {r4-r6,lr}
00008234 E3A04000  MOV      r4,#0
00008238 EBFFFFDA  BL       Init                          <0x81a8>
0000823C E28F00AC  ADR      r0,{pc}+0xb4 ; 0x82f0
00008240 EB00006A  BL       __0printf                     <0x83f0>
00008244 EA000014  B        0x829c                        <TRACE\#73>
00008248 E59F00BC  LDR      r0,0x830c
0000824C E5900000  LDR      r0,[r0,#0]
00008250 E2800001  ADD      r0,r0,#1
00008254 E59F10B0  LDR      r1,0x830c
00008258 E5810000  STR      r0,[r1,#0]
0000825C EB000088  BL       rand                          <0x8484>
00008260 E1A05000  MOV      r5,r0
00008264 E3A0101F  MOV      r1,#0x1f
00008268 EB00009E  BL       __aeabi_idivmod               <0x84e8>
0000826C E59F009C  LDR      r0,0x8310
00008270 E5801000  STR      r1,[r0,#0]
00008274 E5900000  LDR      r0,[r0,#0]
00008278 E3500000  CMP      r0,#0
0000827C 1A000006  BNE      0x829c                        <TRACE\#73>
00008280 E1A01004  MOV      r1,r4
00008284 E2844001  ADD      r4,r4,#1
00008288 E28F0084  ADR      r0,{pc}+0x8c ; 0x8314
0000828C EB000057  BL       __0printf                     <0x83f0>
00008290 EBFFFFAB  BL       GetData                       <0x8144>
00008294 EBFFFF9A  BL       GetAverage                    <0x8104>
00008298 EBFFFF81  BL       SendData                      <0x80a4>
0000829C EAFFFFE9  B        0x8248                        <TRACE\#77>
__user_initial_stackheap:
000082A0 E92D403F  PUSH     {r0-r5,lr}
000082A4 E1A0C000  MOV      r12,r0
000082A8 E1A0E001  MOV      lr,r1
```

```
> trace,prompt 0x00008248
> trace,prompt 0x0000829C

Stop>
```

Figure 40

Run the program for several seconds and stop it. Inspect the Analysis window and note the following:

- The while(1) loop and all function calls made from while(1) are traced
- The only instruction not traced is the Trace Stop point. This instruction (B 0x8248 located at address 0x829C) is used to return to the top of the while(1) loop
- A "Warning: Trace pause" message is generated when the Trace Stop point is reached.
- After the trace pause message, the first traced instruction is the Trace Start point (LDR r0, 0x830C located at 0x8248).


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

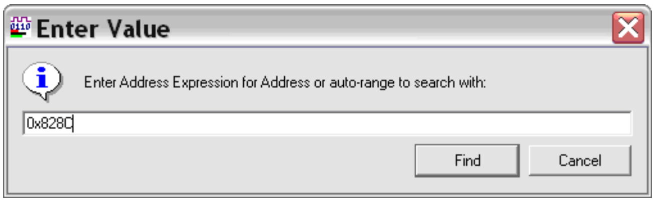
```
Elem  Time/cycl  Type  Symbolic       Address     Opcode      Other
619   20,601     Exec  main\#81       0x0000826C  0xE59F009C  LDR   r0,0x8310
620   20,704     Exec  main\#81       0x00008270  0xE5801000  STR   r1,[r0,#0]
621   20,886     Exec  main\#82..#83  0x00008274  0xE5900000  LDR   r0,[r0,#0]
622   20,927     Exec  main\#83       0x00008278  0xE3500000  CMP   r0,#0
623   20,928     Exec  main\#83       0x0000827C  0x1A000006  BNE   0x829c <TRACE\#73>
623   Warning: Trace pause
624   21,033     Exec  main\#74..#77  0x00008248  0xE59F00BC  LDR   r0,0x830c
625   21,136     Exec  main\#77       0x0000824C  0xE5900000  LDR   r0,[r0,#0]
626   21,319     Exec  main\#77       0x00008250  0xE2800001  ADD   r0,r0,#1
627   21,320     Exec  main\#77       0x00008254  0xE59F10B0  LDR   r1,0x830c
628   21,352     Exec  main\#77       0x00008258  0xE5810000  STR   r0,[r1,#0]
629   21,550     Exec  main\#78..#81  0x0000825C  0xEB000088  BL    rand <0x8484>
630   21,552     Exec  rand           0x00008484  0xE59F304C  LDR   r3,0x84d8 <rand+0x54>
631   21,589     Exec  rand           0x00008488  0xE52DE004  PUSH  {lr}
632   21,766     Exec  rand           0x0000848C  0xE59FC048  LDR   r12,0x84dc <rand+0x58>
633   21,805     Exec  rand           0x00008490  0xE5932000  LDR   r2,[r3,#0]
634   21,837     Exec  rand           0x00008494  0xE5930004  LDR   r0,[r3,#4]
635   21,869     Exec  rand           0x00008498  0xE79CE102  LDR   lr,[r12,r2,LSL #2]
636   21,901     Exec  rand           0x0000849C  0xE79C1100  LDR   r1,[r12,r0,LSL #2]
637   21,933     Exec  rand           0x000084A0  0xE2522001  SUBS  r2,r2,#1
638   21,935     Exec  rand           0x000084A4  0xE081100E  ADD   r1,r1,lr
```

Figure 41

Note

On some ETM v1.x targets, the "Warning: Trace pause" message may not be generated because the instruction located at the trace stop point (at address 0x8274) may also be traced.

Search through the trace capture using the "Find - Find Address Expression" menu from the Analysis Window. Search for 0x828C, the address of the BL __0printf instruction for printing the "Processing Sample" message:

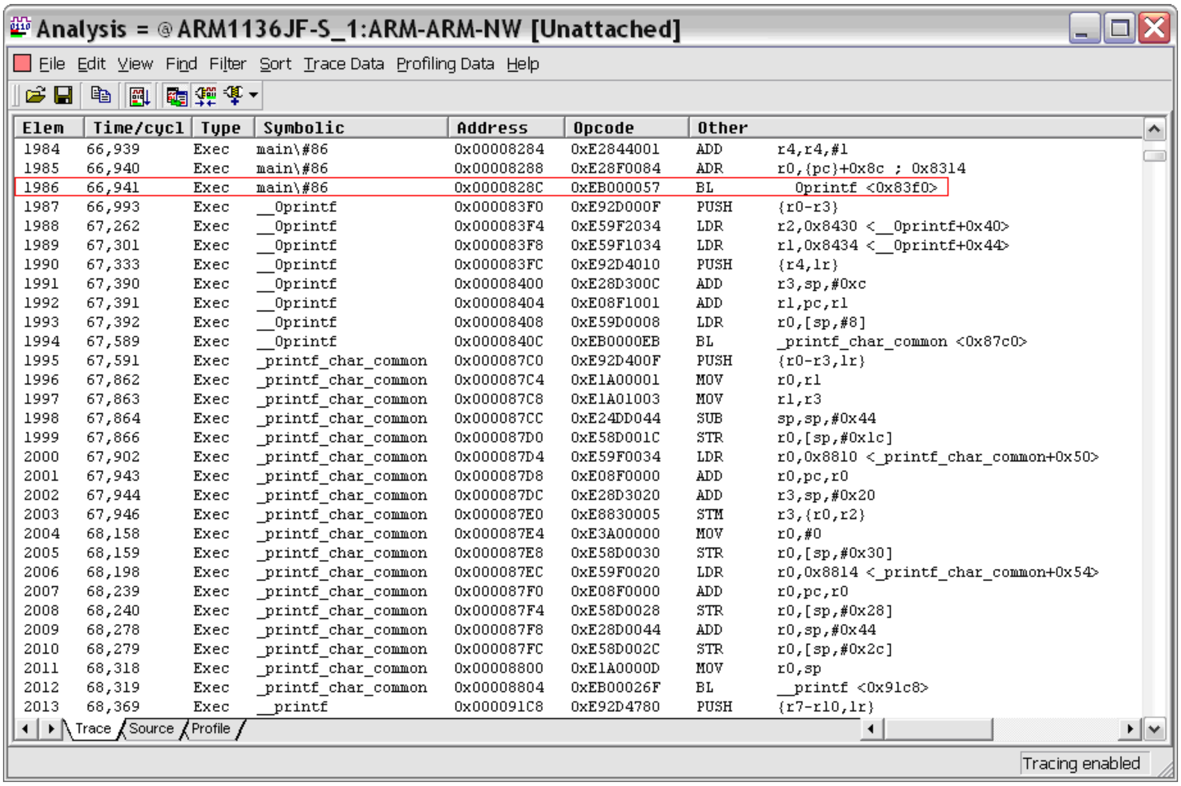


Enter Address Expression for Address or auto-range to search with:

0x828C

Figure 42

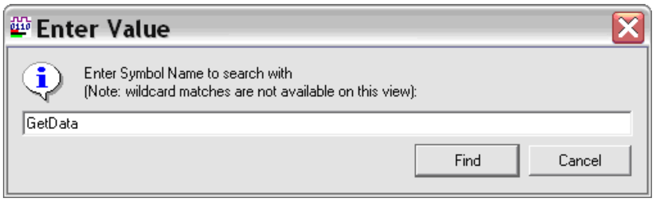
You are brought to the next point in the buffer where sample_ready equals 0 and processing occurs. Note that now the entire __0printf function is traced, unlike the Trace Range capture made in Section 6.1.1:


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

```
Elem  Time/cycl Type  Symbolic              Address     Opcode      Other
1984  66,939    Exec  main\#86              0x00008284  0xE2844001  ADD      r4,r4,#1
1985  66,940    Exec  main\#86              0x00008288  0xE28F0084  ADR      r0,{pc}+0x8c ; 0x8314
1986  66,941    Exec  main\#86              0x0000828C  0xEB000057  BL         0printf <0x83f0>
1987  66,993    Exec  __0printf             0x000083F0  0xE92D000F  PUSH     {r0-r3}
1988  67,262    Exec  __0printf             0x000083F4  0xE59F2034  LDR      r2,0x8430 <__0printf+0x40>
1989  67,301    Exec  __0printf             0x000083F8  0xE59F1034  LDR      r1,0x8434 <__0printf+0x44>
1990  67,333    Exec  __0printf             0x000083FC  0xE92D4010  PUSH     {r4,lr}
1991  67,390    Exec  __0printf             0x00008400  0xE28D300C  ADD      r3,sp,#0xc
1992  67,391    Exec  __0printf             0x00008404  0xE08F1001  ADD      r1,pc,r1
1993  67,392    Exec  __0printf             0x00008408  0xE59D0008  LDR      r0,[sp,#8]
1994  67,589    Exec  __0printf             0x0000840C  0xEB0000EB  BL       _printf_char_common <0x87c0>
1995  67,591    Exec  _printf_char_common   0x000087C0  0xE92D400F  PUSH     {r0-r3,lr}
1996  67,862    Exec  _printf_char_common   0x000087C4  0xE1A00001  MOV      r0,r1
1997  67,863    Exec  _printf_char_common   0x000087C8  0xE1A01003  MOV      r1,r3
1998  67,864    Exec  _printf_char_common   0x000087CC  0xE24DD044  SUB      sp,sp,#0x44
1999  67,866    Exec  _printf_char_common   0x000087D0  0xE58D001C  STR      r0,[sp,#0x1c]
2000  67,902    Exec  _printf_char_common   0x000087D4  0xE59F0034  LDR      r0,0x8810 <_printf_char_common+0x50>
2001  67,943    Exec  _printf_char_common   0x000087D8  0xE08F0000  ADD      r0,pc,r0
2002  67,944    Exec  _printf_char_common   0x000087DC  0xE28D3020  ADD      r3,sp,#0x20
2003  67,946    Exec  _printf_char_common   0x000087E0  0xE8830005  STM      r3,{r0,r2}
2004  68,158    Exec  _printf_char_common   0x000087E4  0xE3A00000  MOV      r0,#0
2005  68,159    Exec  _printf_char_common   0x000087E8  0xE58D0030  STR      r0,[sp,#0x30]
2006  68,198    Exec  _printf_char_common   0x000087EC  0xE59F0020  LDR      r0,0x8814 <_printf_char_common+0x54>
2007  68,239    Exec  _printf_char_common   0x000087F0  0xE08F0000  ADD      r0,pc,r0
2008  68,240    Exec  _printf_char_common   0x000087F4  0xE58D0028  STR      r0,[sp,#0x28]
2009  68,278    Exec  _printf_char_common   0x000087F8  0xE28D0044  ADD      r0,sp,#0x44
2010  68,279    Exec  _printf_char_common   0x000087FC  0xE58D002C  STR      r0,[sp,#0x2c]
2011  68,318    Exec  _printf_char_common   0x00008800  0xE1A0000D  MOV      r0,sp
2012  68,319    Exec  _printf_char_common   0x00008804  0xEB00026F  BL       __printf <0x91c8>
2013  68,369    Exec  __printf              0x000091C8  0xE92D4780  PUSH     {r7-r10,lr}
```

Figure 43

To confirm that other function calls from the while(1) loop have been traced, search for them by name using the "Find - Find Symbol Name" menu from the Analysis Window. No parenthesis should be used when entering the function name:

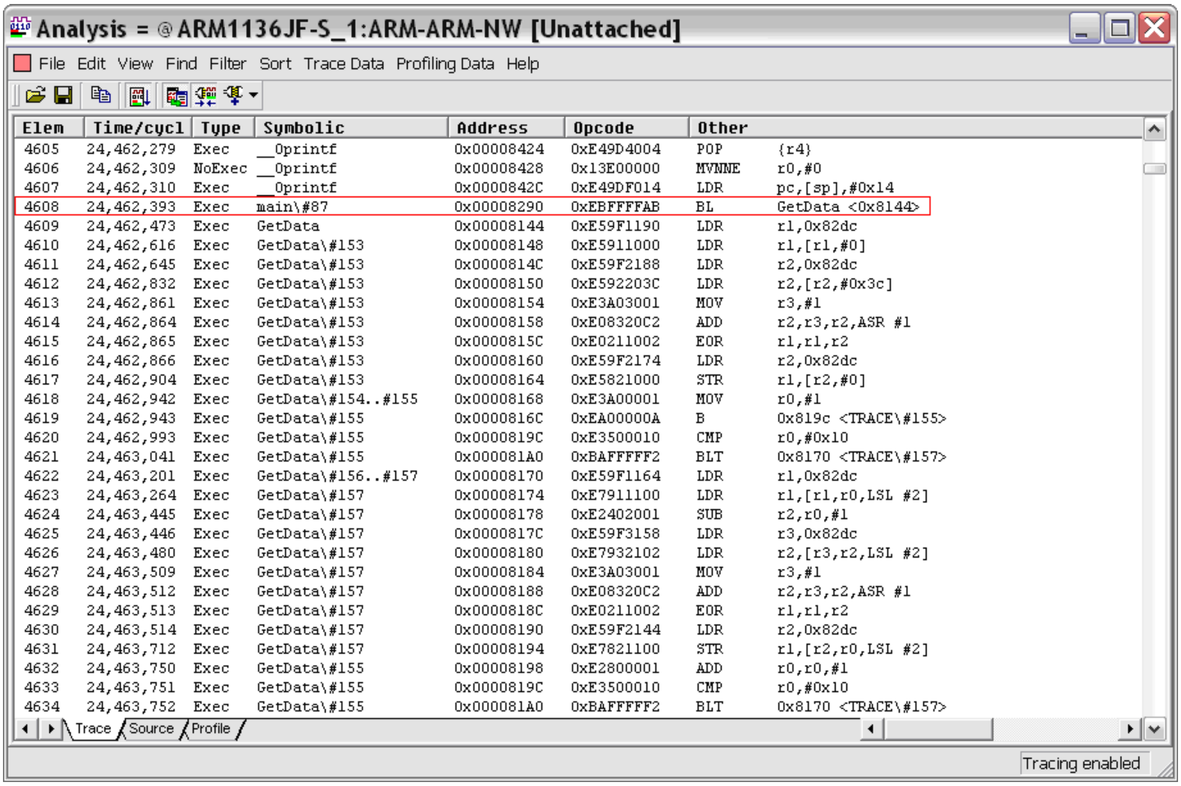


Enter Symbol Name to search with
(Note: wildcard matches are not available on this view):

GetData

Figure 44

The Symbol Name search for GetData will bring you to the next occurrence in the trace buffer of a trace element from GetData():


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Time/cycl	Type	Symbolic	Address	Opcode	Other	
4605	24,462,279	Exec	__0printf	0x00008424	0xE49D4004	POP	{r4}
4606	24,462,309	NoExec	__0printf	0x00008428	0x13E00000	MVNNE	r0,#0
4607	24,462,310	Exec	__0printf	0x0000842C	0xE49DF014	LDR	pc,[sp],#0x14
4608	24,462,393	Exec	main\#87	0x00008290	0xEBFFFFAB	BL	GetData <0x8144>
4609	24,462,473	Exec	GetData	0x00008144	0xE59F1190	LDR	r1,0x82dc
4610	24,462,616	Exec	GetData\#153	0x00008148	0xE5911000	LDR	r1,[r1,#0]
4611	24,462,645	Exec	GetData\#153	0x0000814C	0xE59F2188	LDR	r2,0x82dc
4612	24,462,832	Exec	GetData\#153	0x00008150	0xE592203C	LDR	r2,[r2,#0x3c]
4613	24,462,861	Exec	GetData\#153	0x00008154	0xE3A03001	MOV	r3,#1
4614	24,462,864	Exec	GetData\#153	0x00008158	0xE08320C2	ADD	r2,r3,r2,ASR #1
4615	24,462,865	Exec	GetData\#153	0x0000815C	0xE0211002	EOR	r1,r1,r2
4616	24,462,866	Exec	GetData\#153	0x00008160	0xE59F2174	LDR	r2,0x82dc
4617	24,462,904	Exec	GetData\#153	0x00008164	0xE5821000	STR	r1,[r2,#0]
4618	24,462,942	Exec	GetData\#154..#155	0x00008168	0xE3A00001	MOV	r0,#1
4619	24,462,943	Exec	GetData\#155	0x0000816C	0xEA00000A	B	0x819c <TRACE\#155>
4620	24,462,993	Exec	GetData\#155	0x0000819C	0xE3500010	CMP	r0,#0x10
4621	24,463,041	Exec	GetData\#155	0x000081A0	0xBAFFFFF2	BLT	0x8170 <TRACE\#157>
4622	24,463,201	Exec	GetData\#156..#157	0x00008170	0xE59F1164	LDR	r1,0x82dc
4623	24,463,264	Exec	GetData\#157	0x00008174	0xE7911100	LDR	r1,[r1,r0,LSL #2]
4624	24,463,445	Exec	GetData\#157	0x00008178	0xE2402001	SUB	r2,r0,#1
4625	24,463,446	Exec	GetData\#157	0x0000817C	0xE59F3158	LDR	r3,0x82dc
4626	24,463,480	Exec	GetData\#157	0x00008180	0xE7932102	LDR	r2,[r3,r2,LSL #2]
4627	24,463,509	Exec	GetData\#157	0x00008184	0xE3A03001	MOV	r3,#1
4628	24,463,512	Exec	GetData\#157	0x00008188	0xE08320C2	ADD	r2,r3,r2,ASR #1
4629	24,463,513	Exec	GetData\#157	0x0000818C	0xE0211002	EOR	r1,r1,r2
4630	24,463,514	Exec	GetData\#157	0x00008190	0xE59F2144	LDR	r2,0x82dc
4631	24,463,712	Exec	GetData\#157	0x00008194	0xE7821100	STR	r1,[r2,r0,LSL #2]
4632	24,463,750	Exec	GetData\#155	0x00008198	0xE2800001	ADD	r0,r0,#1
4633	24,463,751	Exec	GetData\#155	0x0000819C	0xE3500010	CMP	r0,#0x10
4634	24,463,752	Exec	GetData\#155	0x000081A0	0xBAFFFFF2	BLT	0x8170 <TRACE\#157>

Figure 45

Now search for the symbols rand, GetAverage and SendData to show that all functions from the while(1) loop are traced.

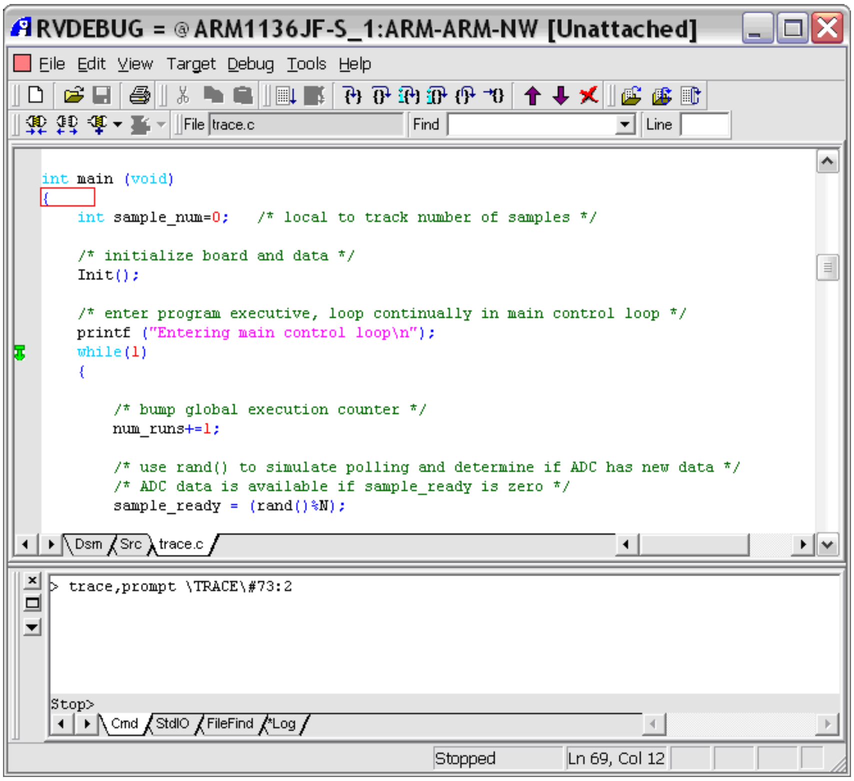
6.2 Trace Range Exclusions

Trace Range Exclusions are used to prevent trace from being captured over a specified address range. Trace exclusions are particularly useful if you have a function that is frequently called, but you don't want it traced. You may decide to exclude code portions from your trace capture so you can save the limited trace memory for areas that are of greater interest. For instance, trusted library functions may be good candidates for trace exclusion if you require more trace visibility of your own written code.

In this scenario a Trace Start Point is used to enable trace capture, and an Excluded Trace Range is used to prevent the function GetData() from being traced. Remove the previous trace setup. Using the C source view, place the following trace points:

- Trace Start Point in main() at while(1)
- Start Excluded Trace Range (Inst + Data) at start of GetData()
- End Excluded Trace Range (Inst + Data) at end of GetData()

The location of the Trace Start Point is shown here:

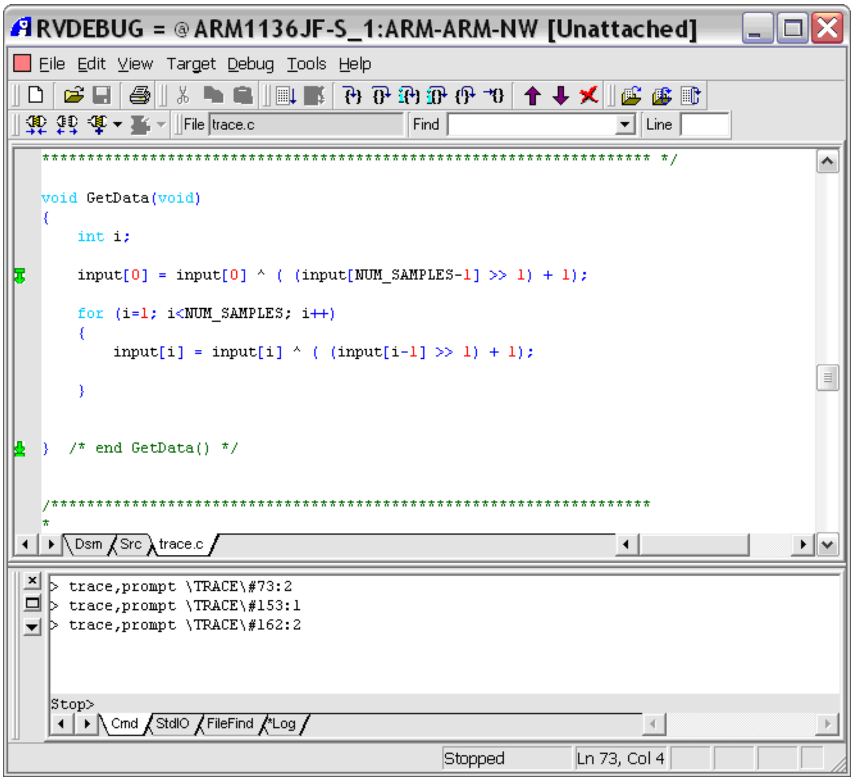

RVDEBUG = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Target Debug Tools Help
File trace.c
Find
Line
Dsm
Src
trace.c
Cmd
StdIO
FileFind
*Log
Stopped
Ln 69, Col 12

```
int main (void)
{
    int sample_num=0;   /* local to track number of samples */

    /* initialize board and data */
    Init();

    /* enter program executive, loop continually in main control loop */
    printf ("Entering main control loop\n");
    while(1)
    {

        /* bump global execution counter */
        num_runs+=1;

        /* use rand() to simulate polling and determine if ADC has new data */
        /* ADC data is available if sample_ready is zero */
        sample_ready = (rand()%N);
```

```
> trace,prompt \TRACE\#73:2

Stop>
```

Figure 46

The location of the Excluded Trace Range is shown here:

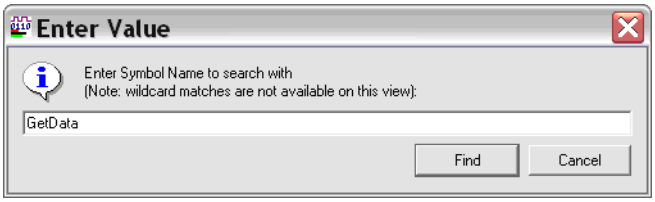

RVDEBUG = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Target Debug Tools Help
File trace.c
Find
Line
Dsm
Src
trace.c
Cmd
StdIO
FileFind
*Log
Stopped
Ln 73, Col 4

```
******************************************************************** */

void GetData(void)
{
    int i;

    input[0] = input[0] ^ ( (input[NUM_SAMPLES-1] >> 1) + 1);

    for (i=1; i<NUM_SAMPLES; i++)
    {
        input[i] = input[i] ^ ( (input[i-1] >> 1) + 1);

    }


}  /* end GetData() */


/********************************************************************
*
```

```
> trace,prompt \TRACE\#73:2
> trace,prompt \TRACE\#153:1
> trace,prompt \TRACE\#162:2

Stop>
```

Figure 47

Reload TRACE.AXF, run the program for a few seconds and stop it. Open the Trace Analysis window, scroll through it and observe that the while(1) loop is traced. Perform a search for GetData using the "Find - Find Symbol Name" menu from the Analysis Window:

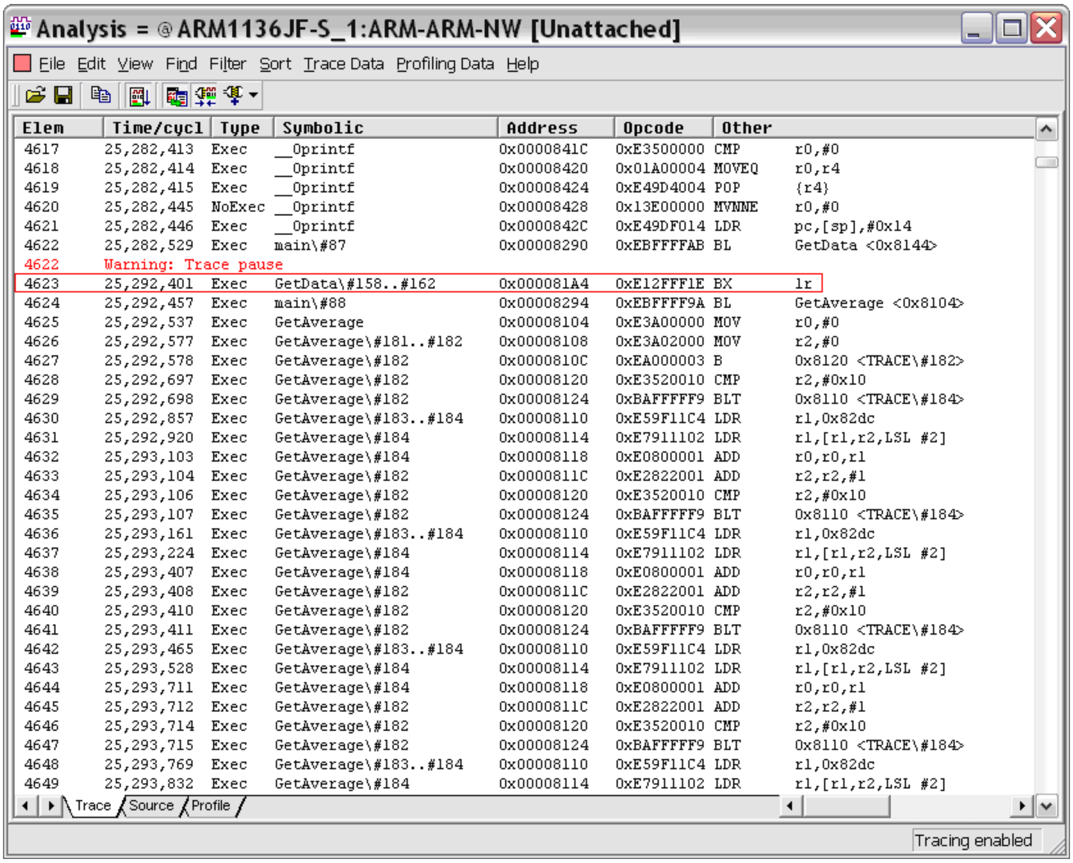


Enter Symbol Name to search with
(Note: wildcard matches are not available on this view):

GetData

Figure 48

Observe that there is only one instruction traced from GetData(), while all the other functions called from the while(1) loop are completely traced. The BX LR instruction from GetData() is used to return from the function. It is present in the trace capture because the end of the excluded trace range fell at this address (0x81A4) when the trace range was set from the C source view.

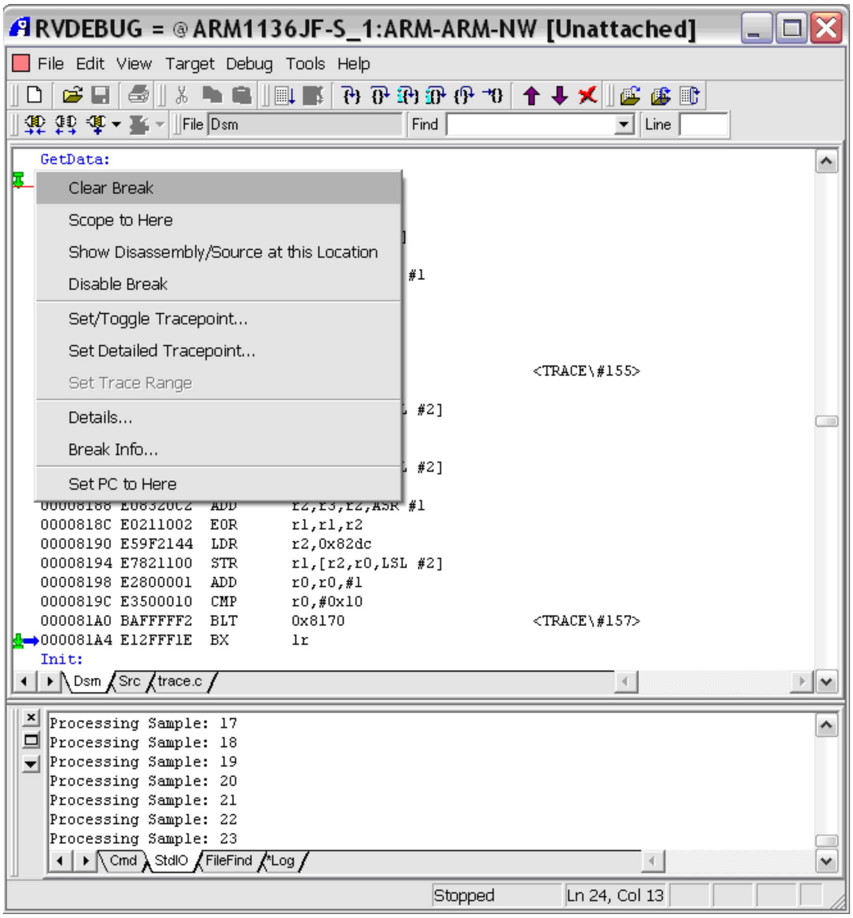

Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Time/cycl	Type	Symbolic	Address	Opcode	Other
4617	25,282,413	Exec	__0printf	0x0000841C	0xE3500000 CMP	r0,#0
4618	25,282,414	Exec	__0printf	0x00008420	0x01A00004 MOVEQ	r0,r4
4619	25,282,415	Exec	__0printf	0x00008424	0xE49D4004 POP	{r4}
4620	25,282,445	NoExec	__0printf	0x00008428	0x13E00000 MVNNE	r0,#0
4621	25,282,446	Exec	__0printf	0x0000842C	0xE49DF014 LDR	pc,[sp],#0x14
4622	25,282,529	Exec	main\#87	0x00008290	0xEBFFFFAB BL	GetData <0x8144>
4622	Warning: Trace pause					
4623	25,292,401	Exec	GetData\#158..#162	0x000081A4	0xE12FFF1E BX	lr
4624	25,292,457	Exec	main\#88	0x00008294	0xEBFFFF9A BL	GetAverage <0x8104>
4625	25,292,537	Exec	GetAverage	0x00008104	0xE3A00000 MOV	r0,#0
4626	25,292,577	Exec	GetAverage\#181..#182	0x00008108	0xE3A02000 MOV	r2,#0
4627	25,292,578	Exec	GetAverage\#182	0x0000810C	0xEA000003 B	0x8120 <TRACE\#182>
4628	25,292,697	Exec	GetAverage\#182	0x00008120	0xE3520010 CMP	r2,#0x10
4629	25,292,698	Exec	GetAverage\#182	0x00008124	0xBAFFFFF9 BLT	0x8110 <TRACE\#184>
4630	25,292,857	Exec	GetAverage\#183..#184	0x00008110	0xE59F11C4 LDR	r1,0x82dc
4631	25,292,920	Exec	GetAverage\#184	0x00008114	0xE7911102 LDR	r1,[r1,r2,LSL #2]
4632	25,293,103	Exec	GetAverage\#184	0x00008118	0xE0800001 ADD	r0,r0,r1
4633	25,293,104	Exec	GetAverage\#182	0x0000811C	0xE2822001 ADD	r2,r2,#1
4634	25,293,106	Exec	GetAverage\#182	0x00008120	0xE3520010 CMP	r2,#0x10
4635	25,293,107	Exec	GetAverage\#182	0x00008124	0xBAFFFFF9 BLT	0x8110 <TRACE\#184>
4636	25,293,161	Exec	GetAverage\#183..#184	0x00008110	0xE59F11C4 LDR	r1,0x82dc
4637	25,293,224	Exec	GetAverage\#184	0x00008114	0xE7911102 LDR	r1,[r1,r2,LSL #2]
4638	25,293,407	Exec	GetAverage\#184	0x00008118	0xE0800001 ADD	r0,r0,r1
4639	25,293,408	Exec	GetAverage\#182	0x0000811C	0xE2822001 ADD	r2,r2,#1
4640	25,293,410	Exec	GetAverage\#182	0x00008120	0xE3520010 CMP	r2,#0x10
4641	25,293,411	Exec	GetAverage\#182	0x00008124	0xBAFFFFF9 BLT	0x8110 <TRACE\#184>
4642	25,293,465	Exec	GetAverage\#183..#184	0x00008110	0xE59F11C4 LDR	r1,0x82dc
4643	25,293,528	Exec	GetAverage\#184	0x00008114	0xE7911102 LDR	r1,[r1,r2,LSL #2]
4644	25,293,711	Exec	GetAverage\#184	0x00008118	0xE0800001 ADD	r0,r0,r1
4645	25,293,712	Exec	GetAverage\#182	0x0000811C	0xE2822001 ADD	r2,r2,#1
4646	25,293,714	Exec	GetAverage\#182	0x00008120	0xE3520010 CMP	r2,#0x10
4647	25,293,715	Exec	GetAverage\#182	0x00008124	0xBAFFFFF9 BLT	0x8110 <TRACE\#184>
4648	25,293,769	Exec	GetAverage\#183..#184	0x00008110	0xE59F11C4 LDR	r1,0x82dc
4649	25,293,832	Exec	GetAverage\#184	0x00008114	0xE7911102 LDR	r1,[r1,r2,LSL #2]

Figure 49

You must be aware that this is a common occurrence when placing trace points from the C source window (from the Src tab). The effects of this phenomenon can be even greater if you build your application with a high level of compiler optimization. Precise control of trace point placement can be achieved using the Disassembly source view.

To exclude all instructions of GetData() from the trace (including the BX LR which was previously traced), the Trace Range Exclusion must be set using the Disassembly source view. Click on the Dsm tab and remove the previous trace range by clicking on either trace range point and selecting "Clear Break":



```
GetData:
                                        ]
                                        #1

                                                        <TRACE\#155>
                                         #2]

                                         #2]
00008188 E08320C2  ADD      r2,r3,r2,ASR #1
0000818C E0211002  EOR      r1,r1,r2
00008190 E59F2144  LDR      r2,0x82dc
00008194 E7821100  STR      r1,[r2,r0,LSL #2]
00008198 E2800001  ADD      r0,r0,#1
0000819C E3500010  CMP      r0,#0x10
000081A0 BAFFFFF2  BLT      0x8170                  <TRACE\#157>
000081A4 E12FFF1E  BX       lr
Init:
```

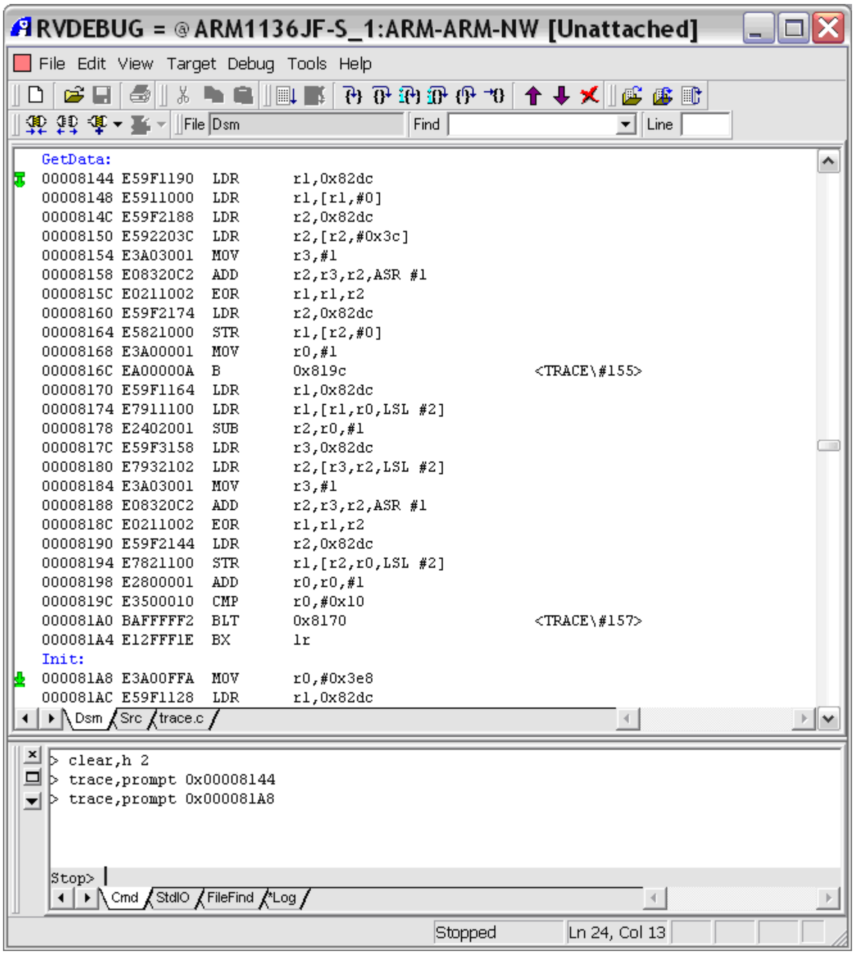
```
Processing Sample: 17
Processing Sample: 18
Processing Sample: 19
Processing Sample: 20
Processing Sample: 21
Processing Sample: 22
Processing Sample: 23
```

Figure 50

Note

Do not remove the Trace Start Point.

Place the new Excluded Trace Range at address locations 0x8144 (the first instruction in GetData()) and 0x81A8 (the first instruction in Init()):

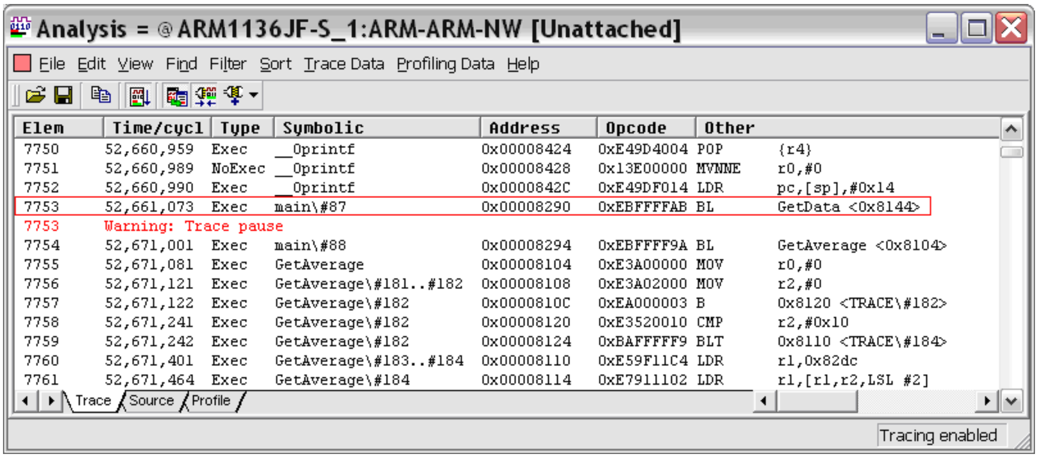

RVDEBUG = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Target Debug Tools Help
File Dsm
Find
Line
Dsm
Src
trace.c
Cmd
StdIO
FileFind
Log
Stopped
Ln 24, Col 13

```
GetData:
00008144 E59F1190  LDR     r1,0x82dc
00008148 E5911000  LDR     r1,[r1,#0]
0000814C E59F2188  LDR     r2,0x82dc
00008150 E592203C  LDR     r2,[r2,#0x3c]
00008154 E3A03001  MOV     r3,#1
00008158 E08320C2  ADD     r2,r3,r2,ASR #1
0000815C E0211002  EOR     r1,r1,r2
00008160 E59F2174  LDR     r2,0x82dc
00008164 E5821000  STR     r1,[r2,#0]
00008168 E3A00001  MOV     r0,#1
0000816C EA00000A  B       0x819c                     <TRACE\#155>
00008170 E59F1164  LDR     r1,0x82dc
00008174 E7911100  LDR     r1,[r1,r0,LSL #2]
00008178 E2402001  SUB     r2,r0,#1
0000817C E59F3158  LDR     r3,0x82dc
00008180 E7932102  LDR     r2,[r3,r2,LSL #2]
00008184 E3A03001  MOV     r3,#1
00008188 E08320C2  ADD     r2,r3,r2,ASR #1
0000818C E0211002  EOR     r1,r1,r2
00008190 E59F2144  LDR     r2,0x82dc
00008194 E7821100  STR     r1,[r2,r0,LSL #2]
00008198 E2800001  ADD     r0,r0,#1
0000819C E3500010  CMP     r0,#0x10
000081A0 BAFFFFF2  BLT     0x8170                     <TRACE\#157>
000081A4 E12FFF1E  BX      lr
Init:
000081A8 E3A00FFA  MOV     r0,#0x3e8
000081AC E59F1128  LDR     r1,0x82dc
```

```
> clear,h 2
> trace,prompt 0x00008144
> trace,prompt 0x000081A8

Stop>
```

Figure 51

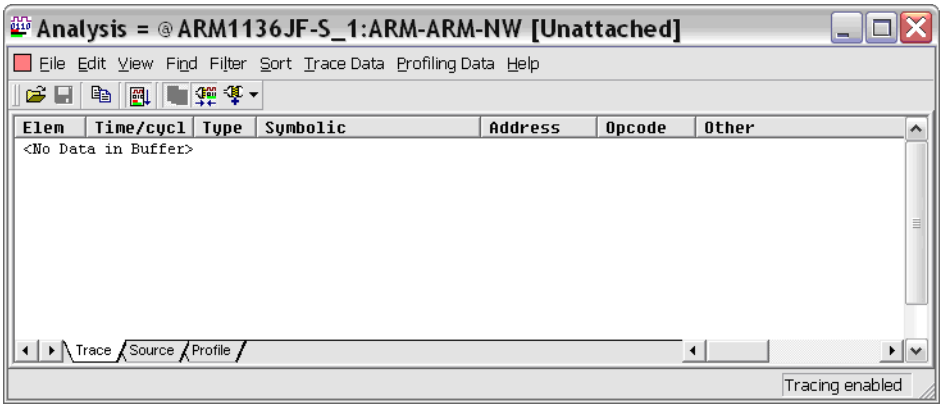
Reload the program, run it for a few seconds and stop it. Observe that the new trace condition results in a trace capture with all instructions of GetData() removed:


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

```
Elem    Time/cycl  Type    Symbolic                 Address     Opcode           Other
7750    52,660,959 Exec    __0printf                0x00008424  0xE49D4004 POP   {r4}
7751    52,660,989 NoExec  __0printf                0x00008428  0x13E00000 MVNNE r0,#0
7752    52,660,990 Exec    __0printf                0x0000842C  0xE49DF014 LDR   pc,[sp],#0x14
7753    52,661,073 Exec    main\#87                 0x00008290  0xEBFFFFAB BL    GetData <0x8144>
7753    Warning: Trace pause
7754    52,671,001 Exec    main\#88                 0x00008294  0xEBFFFF9A BL    GetAverage <0x8104>
7755    52,671,081 Exec    GetAverage               0x00008104  0xE3A00000 MOV   r0,#0
7756    52,671,121 Exec    GetAverage\#181..#182    0x00008108  0xE3A02000 MOV   r2,#0
7757    52,671,122 Exec    GetAverage\#182          0x0000810C  0xEA000003 B     0x8120 <TRACE\#182>
7758    52,671,241 Exec    GetAverage\#182          0x00008120  0xE3520010 CMP   r2,#0x10
7759    52,671,242 Exec    GetAverage\#182          0x00008124  0xBAFFFFF9 BLT   0x8110 <TRACE\#184>
7760    52,671,401 Exec    GetAverage\#183..#184    0x00008110  0xE59F11C4 LDR   r1,0x82dc
7761    52,671,464 Exec    GetAverage\#184          0x00008114  0xE7911102 LDR   r1,[r1,r2,LSL #2]
```

Figure 52

Interestingly, if you run and stop the program again after it is halted, the Trace Analysis window will be empty and report "<No Data in Buffer>" :



Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]

File Edit View Find Filter Sort Trace Data Profiling Data Help
Elem
Time/cycl
Type
Symbolic
Address
Opcode
Other
Trace
Source
Profile

```
<No Data in Buffer>
```

Tracing enabled

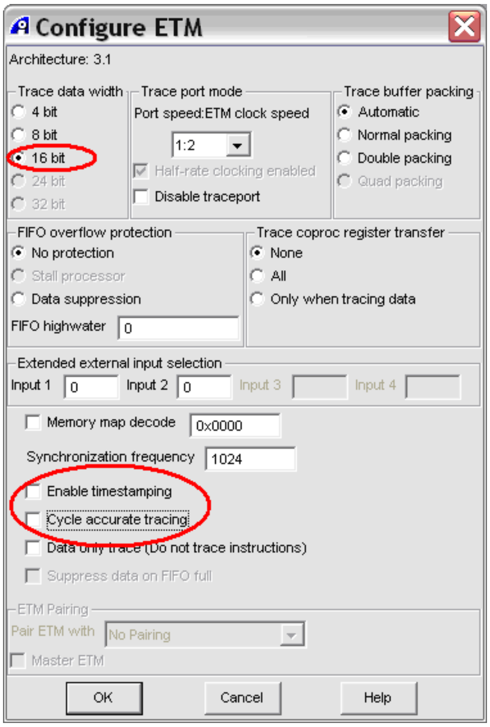
Figure 53

Trace data is not captured because the trace conditions have not been met. The Trace Start Point (at the while(1) statement) is only executed once in the program (not every time through the loop). For trace to be collected again, you will have to reload TRACE.AXF so that the Trace Start Point instruction is executed.

7 Tracing Data

Trace is customarily used to determine program execution flow and Sections 4 - 6 of this Application Note have described methods of capturing Instruction Trace. However, there are times when you require visibility of the data accesses made by your program. The Embedded Trace Macrocell (ETM) provides the ability to trace data and this section describes how data trace can be performed from RVD.

Tracing data is a very data intensive operation and may result in trace overflows if you are not tracing with an ETB. Trace overflows occur when the internal FIFO of the ETM is full and new trace data is prevented from being stored. To minimize the possibility of trace overflow, configure your ETM to use the maximum possible trace data width (typically 16 bits) and disable Cycle Accurate trace:



Configure ETM

Architecture: 3.1

Trace data width: ○ 4 bit ○ 8 bit ● 16 bit ○ 24 bit ○ 32 bit

Trace port mode: Port speed:ETM clock speed 1:2; ☑ Half-rate clocking enabled; □ Disable traceport

Trace buffer packing: ● Automatic ○ Normal packing ○ Double packing ○ Quad packing

FIFO overflow protection: ● No protection ○ Stall processor ○ Data suppression; FIFO highwater 0

Trace coproc register transfer: ● None ○ All ○ Only when tracing data

Extended external input selection: Input 1 0 Input 2 0 Input 3 Input 4

□ Memory map decode 0x0000

Synchronization frequency 1024

□ Enable timestamping

□ Cycle accurate tracing

□ Data only trace (Do not trace instructions)

□ Suppress data on FIFO full

ETM Pairing: Pair ETM with No Pairing; □ Master ETM

Figure 54

7.1 Tracing Data With Auto Trace

After trace is first enabled and configured as described in Section 3, RVD places your target in Auto Tracing mode. By default, only instructions are traced. To enable Auto Tracing for instructions and data, use the "Edit - Automatic Tracing Mode" menu from the Analysis window and select "Instructions and Data" as shown below:

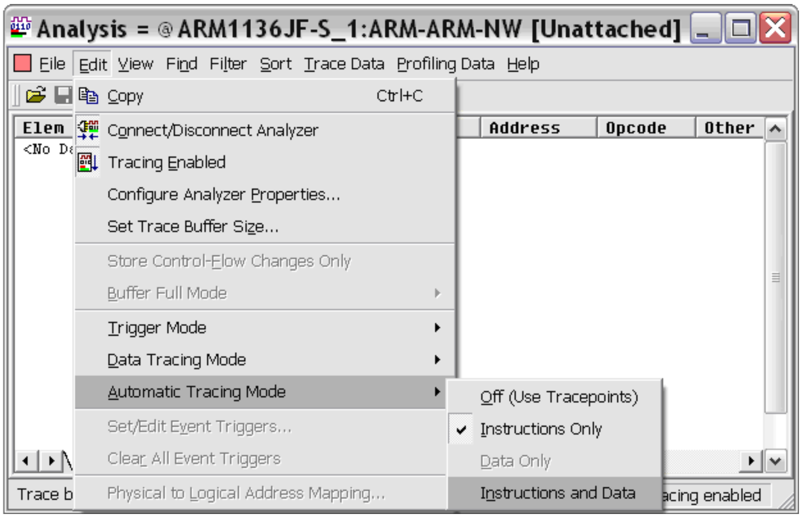

Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Copy Ctrl+C
Connect/Disconnect Analyzer
Tracing Enabled
Configure Analyzer Properties...
Set Trace Buffer Size...
Store Control-Flow Changes Only
Buffer Full Mode
Trigger Mode
Data Tracing Mode
Automatic Tracing Mode
Set/Edit Event Triggers...
Clear All Event Triggers
Physical to Logical Address Mapping...
Off (Use Tracepoints)
✓ Instructions Only
Data Only
Instructions and Data
Elem
<No Da
Address
Opcode
Other
Trace b
acing enabled

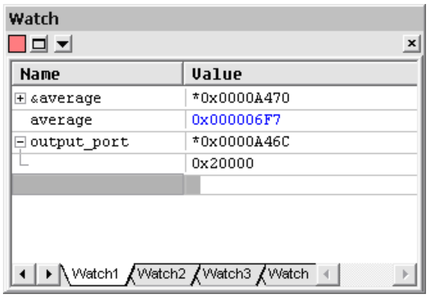
Figure 55

——— **Note** ———

Auto Tracing is also enabled if trace was used with trace points and all the trace points are removed.

————————

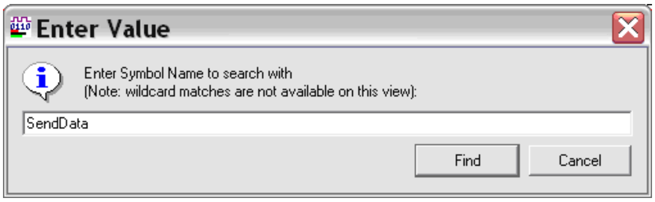
Reload TRACE.AXF and remove any previous trace points. Add a Watch window in RVD. In the Watch window add "&average", "average" and "output_port":



Name	Value
⊞ &average	*0x0000A470
average	0x000006F7
⊟ output_port	*0x0000A46C
└	0x20000

Figure 56

Run the program for a few seconds and stop it. Search for the symbol "SendData" using the "Find - Find Symbol Name" menu from the Analysis Window:

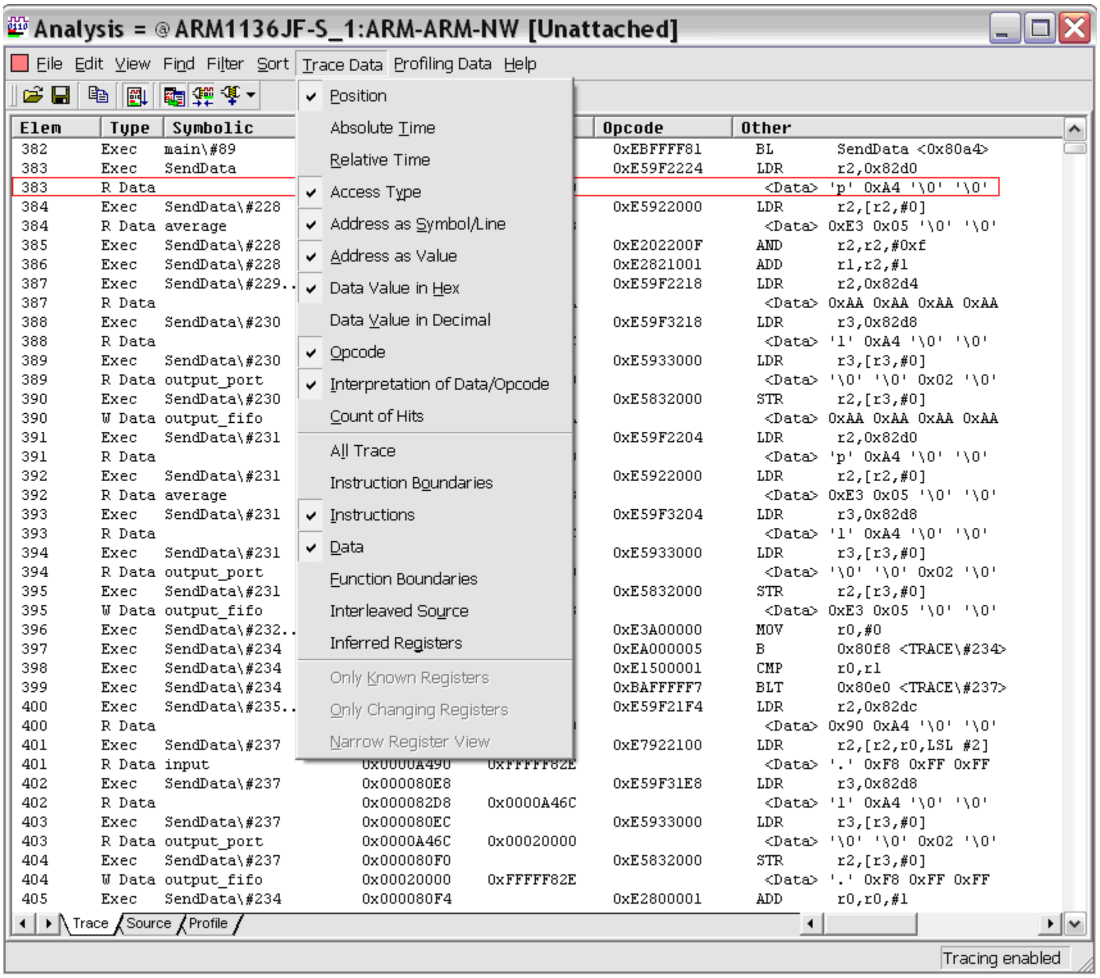


Enter Symbol Name to search with
(Note: wildcard matches are not available on this view):

SendData

Figure 57

To make the Analysis display more legible, use the "Trace Data" menu to select only the Position, Access Type, Address as Symbol/Line, Address as Value, Data Value in Hex, Opcode, Interpretation of Data/Opcode, Instructions and Data columns:



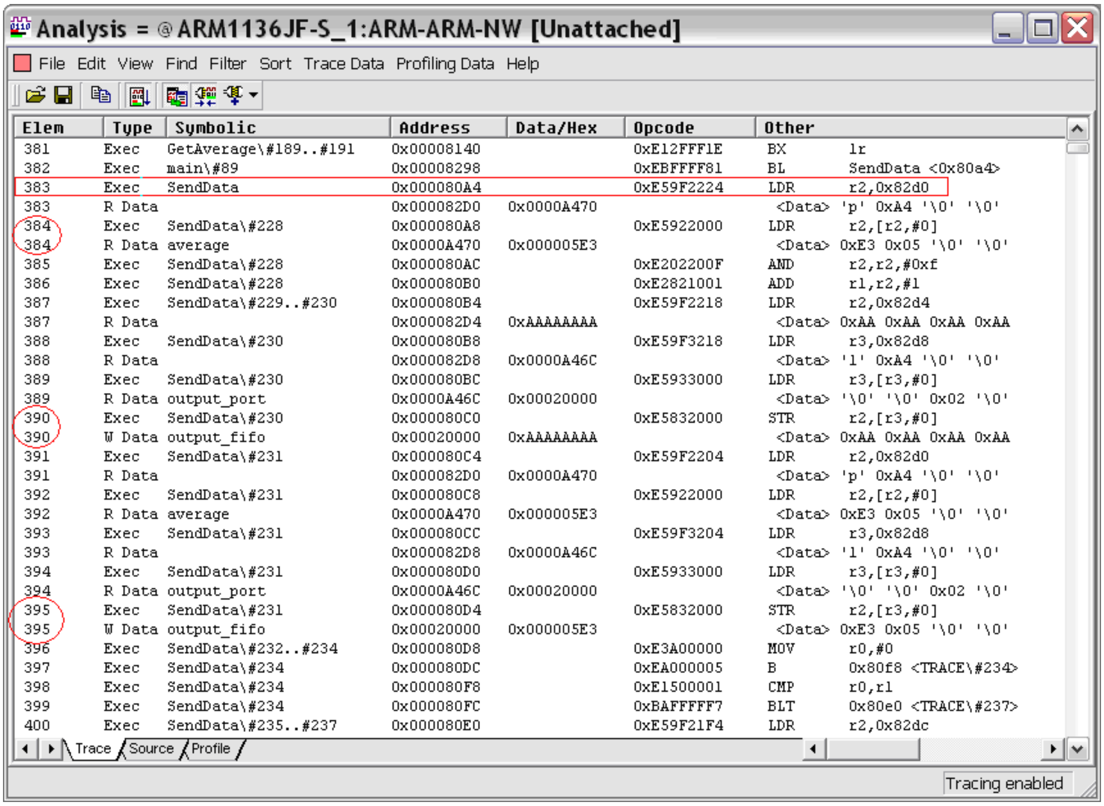
```
Elem   Type   Symbolic                                        Opcode      Other
382    Exec   main\#89                                        0xEBFFFF81  BL      SendData <0x80a4>
383    Exec   SendData                                        0xE59F2224  LDR     r2,0x82d0
383    R Data                                                              <Data> 'p' 0xA4 '\0' '\0'
384    Exec   SendData\#228                                   0xE5922000  LDR     r2,[r2,#0]
384    R Data average                                                      <Data> 0xE3 0x05 '\0' '\0'
385    Exec   SendData\#228                                   0xE202200F  AND     r2,r2,#0xf
386    Exec   SendData\#228                                   0xE2821001  ADD     r1,r2,#1
387    Exec   SendData\#229..                                 0xE59F2218  LDR     r2,0x82d4
387    R Data                                                              <Data> 0xAA 0xAA 0xAA 0xAA
388    Exec   SendData\#230                                   0xE59F3218  LDR     r3,0x82d8
388    R Data                                                              <Data> 'l' 0xA4 '\0' '\0'
389    Exec   SendData\#230                                   0xE5933000  LDR     r3,[r3,#0]
389    R Data output_port                                                  <Data> '\0' '\0' 0x02 '\0'
390    Exec   SendData\#230                                   0xE5832000  STR     r2,[r3,#0]
390    W Data output_fifo                                                  <Data> 0xAA 0xAA 0xAA 0xAA
391    Exec   SendData\#231                                   0xE59F2204  LDR     r2,0x82d0
391    R Data                                                              <Data> 'p' 0xA4 '\0' '\0'
392    Exec   SendData\#231                                   0xE5922000  LDR     r2,[r2,#0]
392    R Data average                                                      <Data> 0xE3 0x05 '\0' '\0'
393    Exec   SendData\#231                                   0xE59F3204  LDR     r3,0x82d8
393    R Data                                                              <Data> 'l' 0xA4 '\0' '\0'
394    Exec   SendData\#231                                   0xE5933000  LDR     r3,[r3,#0]
394    R Data output_port                                                  <Data> '\0' '\0' 0x02 '\0'
395    Exec   SendData\#231                                   0xE5832000  STR     r2,[r3,#0]
395    W Data output_fifo                                                  <Data> 0xE3 0x05 '\0' '\0'
396    Exec   SendData\#232..                                 0xE3A00000  MOV     r0,#0
397    Exec   SendData\#234                                   0xEA000005  B       0x80f8 <TRACE\#234>
398    Exec   SendData\#234                                   0xE1500001  CMP     r0,r1
399    Exec   SendData\#234                                   0xBAFFFFF7  BLT     0x80e0 <TRACE\#237>
400    Exec   SendData\#235..                                 0xE59F21F4  LDR     r2,0x82dc
400    R Data                                                              <Data> 0x90 0xA4 '\0' '\0'
401    Exec   SendData\#237                                   0xE7922100  LDR     r2,[r2,r0,LSL #2]
401    R Data input          0x0000A490      0xFFFFF82E                    <Data> '.' 0xF8 0xFF 0xFF
402    Exec   SendData\#237  0x000080E8                       0xE59F31E8  LDR     r3,0x82d8
402    R Data                0x000082D8      0x0000A46C                    <Data> 'l' 0xA4 '\0' '\0'
403    Exec   SendData\#237  0x000080EC                       0xE5933000  LDR     r3,[r3,#0]
403    R Data output_port    0x0000A46C      0x00020000                    <Data> '\0' '\0' 0x02 '\0'
404    Exec   SendData\#237  0x000080F0                       0xE5832000  STR     r2,[r3,#0]
404    W Data output_fifo    0x00020000      0xFFFFF82E                    <Data> '.' 0xF8 0xFF 0xFF
405    Exec   SendData\#234  0x000080F4                       0xE2800001  ADD     r0,r0,#1
```

Figure 58

——— **Note** ———

The traces in Section 7 are made with Cycle Accurate trace disabled and Timestamping disabled. If you require timing information from your data trace, enable either Cycle Accurate trace or Timestamping as shown in Section 5.2. To view timing information in the Analysis window, select "Absolute Time" or "Relative Time" from the "Trace Data" menu.

Your trace capture should now have the following columns of data:


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Type	Symbolic	Address	Data/Hex	Opcode	Other
381	Exec	GetAverage\#189..#191	0x00008140		0xE12FFF1E	BX lr
382	Exec	main\#89	0x00008298		0xEBFFFF81	BL SendData <0x80a4>
383	Exec	SendData	0x000080A4		0xE59F2224	LDR r2,0x82d0
383	R Data		0x000082D0	0x0000A470		<Data> 'p' 0xA4 '\0' '\0'
384	Exec	SendData\#228	0x000080A8		0xE5922000	LDR r2,[r2,#0]
384	R Data	average	0x0000A470	0x000005E3		<Data> 0xE3 0x05 '\0' '\0'
385	Exec	SendData\#228	0x000080AC		0xE202200F	AND r2,r2,#0xf
386	Exec	SendData\#228	0x000080B0		0xE2821001	ADD r1,r2,#1
387	Exec	SendData\#229..#230	0x000080B4		0xE59F2218	LDR r2,0x82d4
387	R Data		0x000082D4	0xAAAAAAAA		<Data> 0xAA 0xAA 0xAA 0xAA
388	Exec	SendData\#230	0x000080B8		0xE59F3218	LDR r3,0x82d8
388	R Data		0x000082D8	0x0000A46C		<Data> 'l' 0xA4 '\0' '\0'
389	Exec	SendData\#230	0x000080BC		0xE5933000	LDR r3,[r3,#0]
389	R Data	output_port	0x0000A46C	0x00020000		<Data> '\0' '\0' 0x02 '\0'
390	Exec	SendData\#230	0x000080C0		0xE5832000	STR r2,[r3,#0]
390	W Data	output_fifo	0x00020000	0xAAAAAAAA		<Data> 0xAA 0xAA 0xAA 0xAA
391	Exec	SendData\#231	0x000080C4		0xE59F2204	LDR r2,0x82d0
391	R Data		0x000082D0	0x0000A470		<Data> 'p' 0xA4 '\0' '\0'
392	Exec	SendData\#231	0x000080C8		0xE5922000	LDR r2,[r2,#0]
392	R Data	average	0x0000A470	0x000005E3		<Data> 0xE3 0x05 '\0' '\0'
393	Exec	SendData\#231	0x000080CC		0xE59F3204	LDR r3,0x82d8
393	R Data		0x000082D8	0x0000A46C		<Data> 'l' 0xA4 '\0' '\0'
394	Exec	SendData\#231	0x000080D0		0xE5933000	LDR r3,[r3,#0]
394	R Data	output_port	0x0000A46C	0x00020000		<Data> '\0' '\0' 0x02 '\0'
395	Exec	SendData\#231	0x000080D4		0xE5832000	STR r2,[r3,#0]
395	W Data	output_fifo	0x00020000	0x000005E3		<Data> 0xE3 0x05 '\0' '\0'
396	Exec	SendData\#232..#234	0x000080D8		0xE3A00000	MOV r0,#0
397	Exec	SendData\#234	0x000080DC		0xEA000005	B 0x80f8 <TRACE\#234>
398	Exec	SendData\#234	0x000080F8		0xE1500001	CMP r0,r1
399	Exec	SendData\#234	0x000080FC		0xBAFFFFF7	BLT 0x80e0 <TRACE\#237>
400	Exec	SendData\#235..#237	0x000080E0		0xE59F21F4	LDR r2,0x82dc

Figure 59

Note

Your capture may have different element numbers and different values in the Data/Hex and Other columns.

In the trace capture observe that the first data access in SendData is at trace element 383. This access is a read from address 0x82D0 of data 0xA470. This read is generated from the execution of the LDR r2, 0x82D0 instruction (boxed in red). This read is for the computation of the local variable "num_xmit", which resides on the stack:

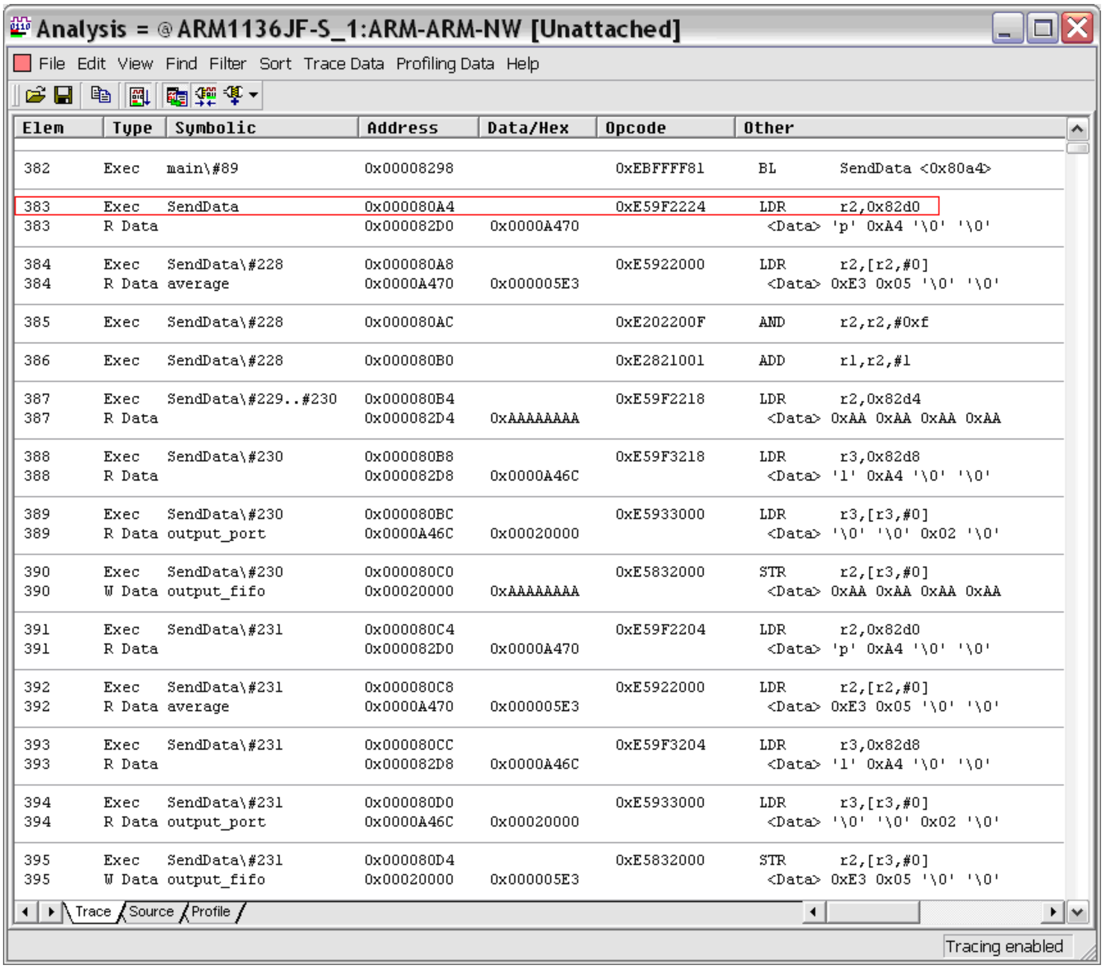
```
int i, num_xmit;
num_xmit = (average & 0xF) + 1;
```

Trace element 384 contains the next data access and this is a read of the global variable "average". The Watch window shows that "average" is stored at 0xA470. Note how "average" also appears in the Symbolic column. In this particular run, average has a value of 0x5E3.

This segment of the trace capture contains two data write operations at trace elements 390 and 395. These are the respective writes of the packet header (0xAAAAAAAA) and computed average (0x5E3) to address 0x20000 (the address of output_fifo, pointed to by *output_port):

```
*output_port = HEADER;   /* output header packet */
*output_port = average;  /* output average value */
```

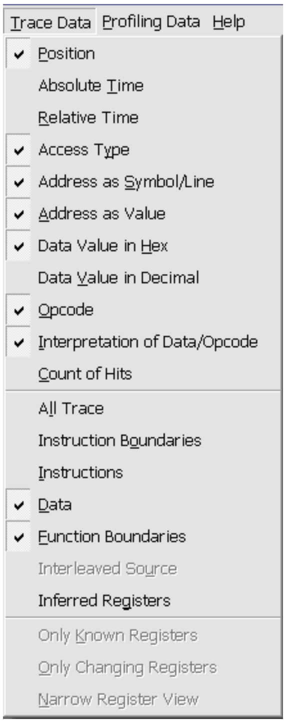
To further improve the trace display, enable "Instruction Boundaries" from the "Trace Data" menu in the Analysis window. This provides a slightly clearer view for interpreting which instruction performed the data access. This view still includes entries which don't make data accesses, such as elements 385 and 386:


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

```
Elem  Type    Symbolic              Address     Data/Hex    Opcode      Other

382   Exec    main\#89              0x00008298              0xEBFFFF81  BL      SendData <0x80a4>

383   Exec    SendData              0x000080A4              0xE59F2224  LDR     r2,0x82d0
383   R Data                        0x000082D0  0x0000A470               <Data> 'p' 0xA4 '\0' '\0'

384   Exec    SendData\#228         0x000080A8              0xE5922000  LDR     r2,[r2,#0]
384   R Data  average               0x0000A470  0x000005E3               <Data> 0xE3 0x05 '\0' '\0'

385   Exec    SendData\#228         0x000080AC              0xE202200F  AND     r2,r2,#0xf

386   Exec    SendData\#228         0x000080B0              0xE2821001  ADD     r1,r2,#1

387   Exec    SendData\#229..#230   0x000080B4              0xE59F2218  LDR     r2,0x82d4
387   R Data                        0x000082D4  0xAAAAAAAA               <Data> 0xAA 0xAA 0xAA 0xAA

388   Exec    SendData\#230         0x000080B8              0xE59F3218  LDR     r3,0x82d8
388   R Data                        0x000082D8  0x0000A46C               <Data> '1' 0xA4 '\0' '\0'

389   Exec    SendData\#230         0x000080BC              0xE5933000  LDR     r3,[r3,#0]
389   R Data  output_port           0x0000A46C  0x00020000               <Data> '\0' '\0' 0x02 '\0'

390   Exec    SendData\#230         0x000080C0              0xE5832000  STR     r2,[r3,#0]
390   W Data  output_fifo           0x00020000  0xAAAAAAAA               <Data> 0xAA 0xAA 0xAA 0xAA

391   Exec    SendData\#231         0x000080C4              0xE59F2204  LDR     r2,0x82d0
391   R Data                        0x000082D0  0x0000A470               <Data> 'p' 0xA4 '\0' '\0'

392   Exec    SendData\#231         0x000080C8              0xE5922000  LDR     r2,[r2,#0]
392   R Data  average               0x0000A470  0x000005E3               <Data> 0xE3 0x05 '\0' '\0'

393   Exec    SendData\#231         0x000080CC              0xE59F3204  LDR     r3,0x82d8
393   R Data                        0x000082D8  0x0000A46C               <Data> '1' 0xA4 '\0' '\0'

394   Exec    SendData\#231         0x000080D0              0xE5933000  LDR     r3,[r3,#0]
394   R Data  output_port           0x0000A46C  0x00020000               <Data> '\0' '\0' 0x02 '\0'

395   Exec    SendData\#231         0x000080D4              0xE5832000  STR     r2,[r3,#0]
395   W Data  output_fifo           0x00020000  0x000005E3               <Data> 0xE3 0x05 '\0' '\0'
```

Figure 60

If you want to focus solely on data accesses, you can enable "Function Boundaries", disable "Instruction Boundaries" and disable "Instructions" from the "Trace Data" menu:



Trace Data Profiling Data Help

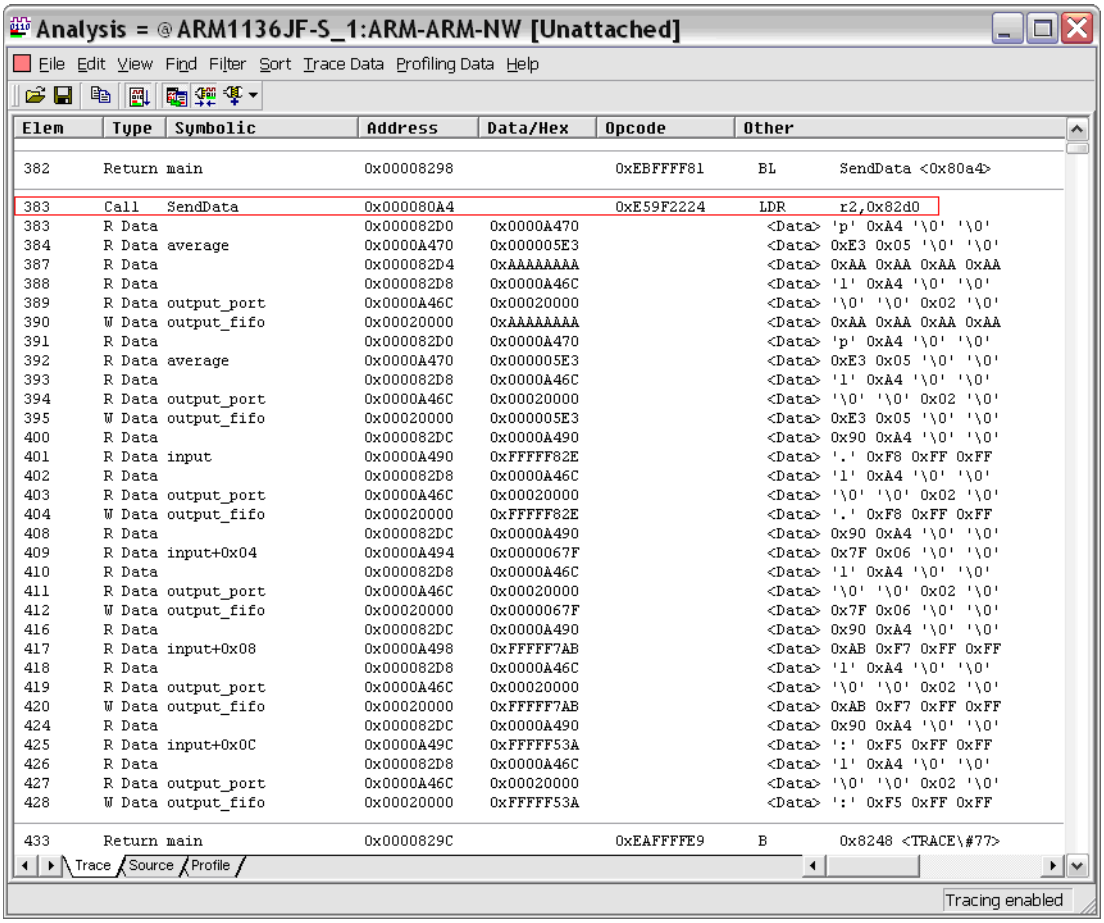
- ✓ Position
- Absolute Time
- Relative Time
- ✓ Access Type
- ✓ Address as Symbol/Line
- ✓ Address as Value
- ✓ Data Value in Hex
- Data Value in Decimal
- ✓ Opcode
- ✓ Interpretation of Data/Opcode
- Count of Hits

- All Trace
- Instruction Boundaries
- Instructions
- ✓ Data
- ✓ Function Boundaries
- Interleaved Source
- Inferred Registers

- Only Known Registers
- Only Changing Registers
- Narrow Register View

Figure 61

This setting provides a clean view of data accesses and still provides some visibility to general program location:

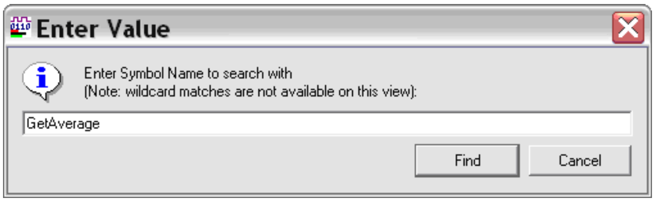


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

```
Elem    Type    Symbolic         Address      Data/Hex     Opcode       Other

382     Return main              0x00008298                0xEBFFFF81   BL      SendData <0x80a4>

383     Call    SendData         0x000080A4                0xE59F2224   LDR     r2,0x82d0
383     R Data                   0x000082D0   0x0000A470                 <Data> 'p' 0xA4 '\0' '\0'
384     R Data average           0x0000A470   0x000005E3                 <Data> 0xE3 0x05 '\0' '\0'
387     R Data                   0x000082D4   0xAAAAAAAA                 <Data> 0xAA 0xAA 0xAA 0xAA
388     R Data                   0x000082D8   0x0000A46C                 <Data> 'l' 0xA4 '\0' '\0'
389     R Data output_port       0x0000A46C   0x00020000                 <Data> '\0' '\0' 0x02 '\0'
390     W Data output_fifo       0x00020000   0xAAAAAAAA                 <Data> 0xAA 0xAA 0xAA 0xAA
391     R Data                   0x000082D0   0x0000A470                 <Data> 'p' 0xA4 '\0' '\0'
392     R Data average           0x0000A470   0x000005E3                 <Data> 0xE3 0x05 '\0' '\0'
393     R Data                   0x000082D8   0x0000A46C                 <Data> 'l' 0xA4 '\0' '\0'
394     R Data output_port       0x0000A46C   0x00020000                 <Data> '\0' '\0' 0x02 '\0'
395     W Data output_fifo       0x00020000   0x000005E3                 <Data> 0xE3 0x05 '\0' '\0'
400     R Data                   0x000082DC   0x0000A490                 <Data> 0x90 0xA4 '\0' '\0'
401     R Data input             0x0000A490   0xFFFFF82E                 <Data> '.' 0xF8 0xFF 0xFF
402     R Data                   0x000082D8   0x0000A46C                 <Data> 'l' 0xA4 '\0' '\0'
403     R Data output_port       0x0000A46C   0x00020000                 <Data> '\0' '\0' 0x02 '\0'
404     W Data output_fifo       0x00020000   0xFFFFF82E                 <Data> '.' 0xF8 0xFF 0xFF
408     R Data                   0x000082DC   0x0000A490                 <Data> 0x90 0xA4 '\0' '\0'
409     R Data input+0x04        0x0000A494   0x0000067F                 <Data> 0x7F 0x06 '\0' '\0'
410     R Data                   0x000082D8   0x0000A46C                 <Data> 'l' 0xA4 '\0' '\0'
411     R Data output_port       0x0000A46C   0x00020000                 <Data> '\0' '\0' 0x02 '\0'
412     W Data output_fifo       0x00020000   0x0000067F                 <Data> 0x7F 0x06 '\0' '\0'
416     R Data                   0x000082DC   0x0000A490                 <Data> 0x90 0xA4 '\0' '\0'
417     R Data input+0x08        0x0000A498   0xFFFFF7AB                 <Data> 0xAB 0xF7 0xFF 0xFF
418     R Data                   0x000082D8   0x0000A46C                 <Data> 'l' 0xA4 '\0' '\0'
419     R Data output_port       0x0000A46C   0x00020000                 <Data> '\0' '\0' 0x02 '\0'
420     W Data output_fifo       0x00020000   0xFFFFF7AB                 <Data> 0xAB 0xF7 0xFF 0xFF
424     R Data                   0x000082DC   0x0000A490                 <Data> 0x90 0xA4 '\0' '\0'
425     R Data input+0x0C        0x0000A49C   0xFFFFF53A                 <Data> ':' 0xF5 0xFF 0xFF
426     R Data                   0x000082D8   0x0000A46C                 <Data> 'l' 0xA4 '\0' '\0'
427     R Data output_port       0x0000A46C   0x00020000                 <Data> '\0' '\0' 0x02 '\0'
428     W Data output_fifo       0x00020000   0xFFFFF53A                 <Data> ':' 0xF5 0xFF 0xFF

433     Return main              0x0000829C                0xEAFFFFE9   B       0x8248 <TRACE\#77>
```

Figure 62

Perform a search for the symbol GetAverage using "Find - Find Symbol Name" from the Analysis Window:

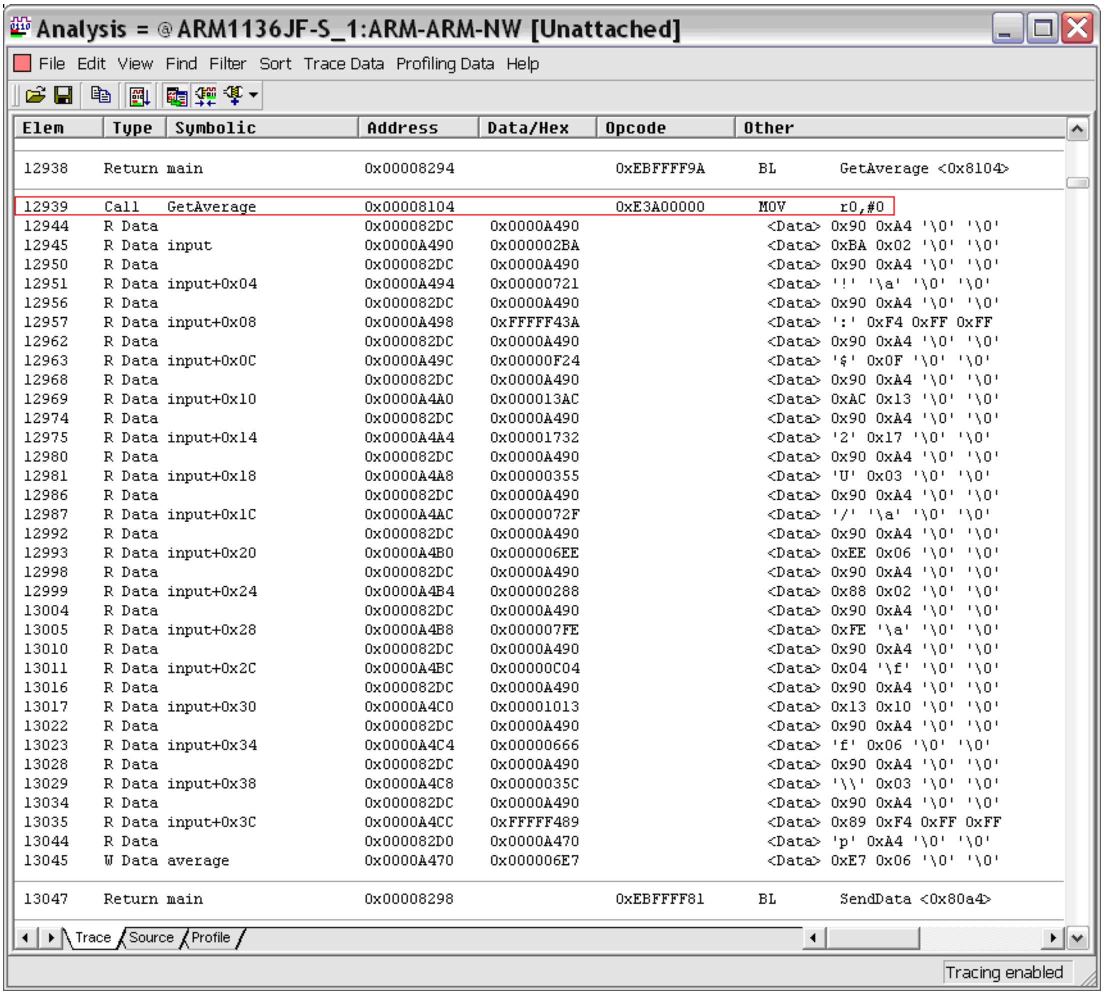


Enter Symbol Name to search with
(Note: wildcard matches are not available on this view):

GetAverage

Figure 63

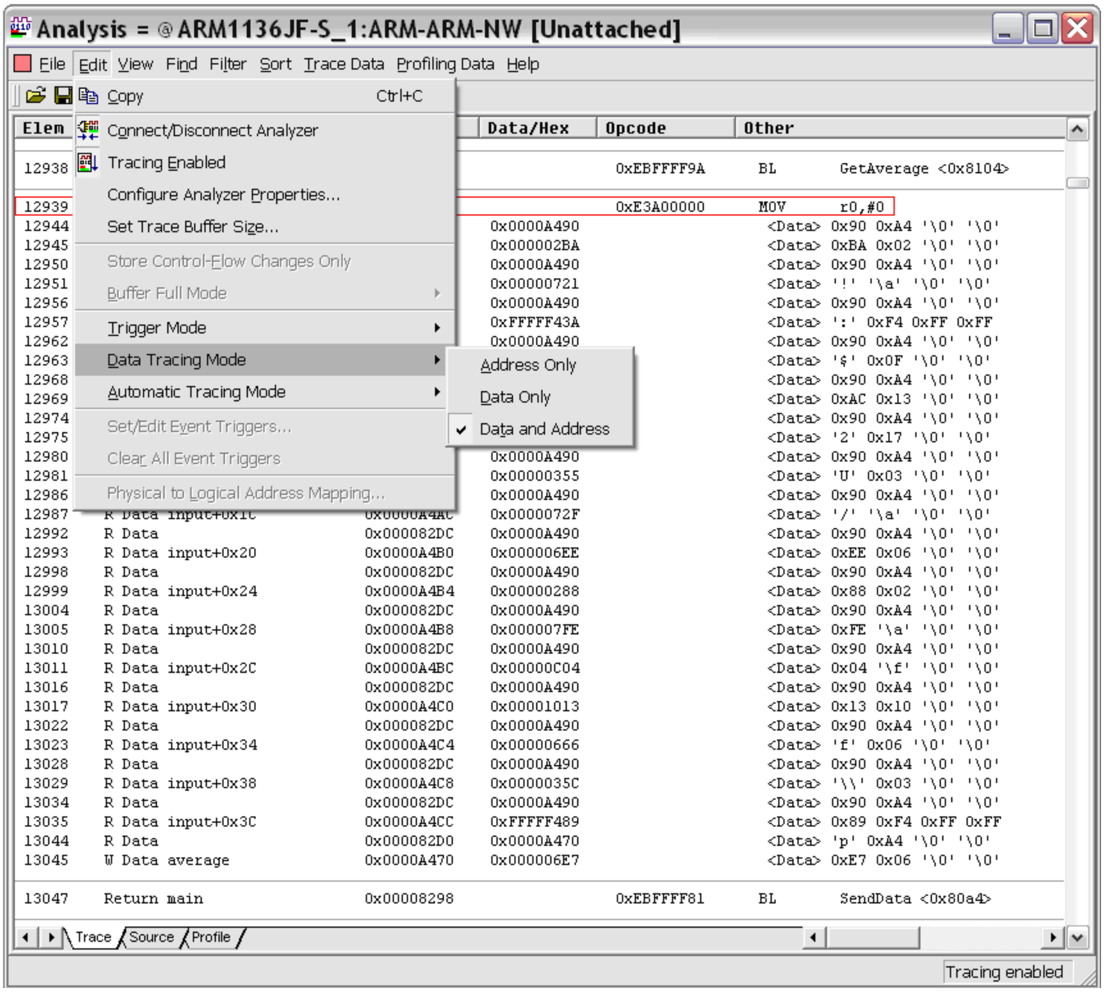
In GetAverage(), observe how the array input[ ] is accessed sequentially to compute "average" and how the array elements are represented in the Symbolic column. The last data operation in the function is a write to "average". In this particular run, the average of array input[ ] is 0x6E7.


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Type	Symbolic	Address	Data/Hex	Opcode	Other
12938	Return	main	0x00008294		0xEBFFFF9A	BL GetAverage <0x8104>
12939	Call	GetAverage	0x00008104		0xE3A00000	MOV r0,#0
12944	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
12945	R Data	input	0x0000A490	0x000002BA		<Data> 0xBA 0x02 '\0' '\0'
12950	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
12951	R Data	input+0x04	0x0000A494	0x00000721		<Data> '!' '\a' '\0' '\0'
12956	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
12957	R Data	input+0x08	0x0000A498	0xFFFFF43A		<Data> ':' 0xF4 0xFF 0xFF
12962	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
12963	R Data	input+0x0C	0x0000A49C	0x00000F24		<Data> '$' 0x0F '\0' '\0'
12968	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
12969	R Data	input+0x10	0x0000A4A0	0x000013AC		<Data> 0xAC 0x13 '\0' '\0'
12974	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
12975	R Data	input+0x14	0x0000A4A4	0x00001732		<Data> '2' 0x17 '\0' '\0'
12980	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
12981	R Data	input+0x18	0x0000A4A8	0x00000355		<Data> 'U' 0x03 '\0' '\0'
12986	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
12987	R Data	input+0x1C	0x0000A4AC	0x0000072F		<Data> '/' '\a' '\0' '\0'
12992	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
12993	R Data	input+0x20	0x0000A4B0	0x000006EE		<Data> 0xEE 0x06 '\0' '\0'
12998	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
12999	R Data	input+0x24	0x0000A4B4	0x00000288		<Data> 0x88 0x02 '\0' '\0'
13004	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
13005	R Data	input+0x28	0x0000A4B8	0x000007FE		<Data> 0xFE '\a' '\0' '\0'
13010	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
13011	R Data	input+0x2C	0x0000A4BC	0x00000C04		<Data> 0x04 '\f' '\0' '\0'
13016	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
13017	R Data	input+0x30	0x0000A4C0	0x00001013		<Data> 0x13 0x10 '\0' '\0'
13022	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
13023	R Data	input+0x34	0x0000A4C4	0x00000666		<Data> 'f' 0x06 '\0' '\0'
13028	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
13029	R Data	input+0x38	0x0000A4C8	0x0000035C		<Data> '\\' 0x03 '\0' '\0'
13034	R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
13035	R Data	input+0x3C	0x0000A4CC	0xFFFFF489		<Data> 0x89 0xF4 0xFF 0xFF
13044	R Data		0x000082D0	0x0000A470		<Data> 'p' 0xA4 '\0' '\0'
13045	W Data	average	0x0000A470	0x000006E7		<Data> 0xE7 0x06 '\0' '\0'
13047	Return	main	0x00008298		0xEBFFFF81	BL SendData <0x80a4>

Figure 64

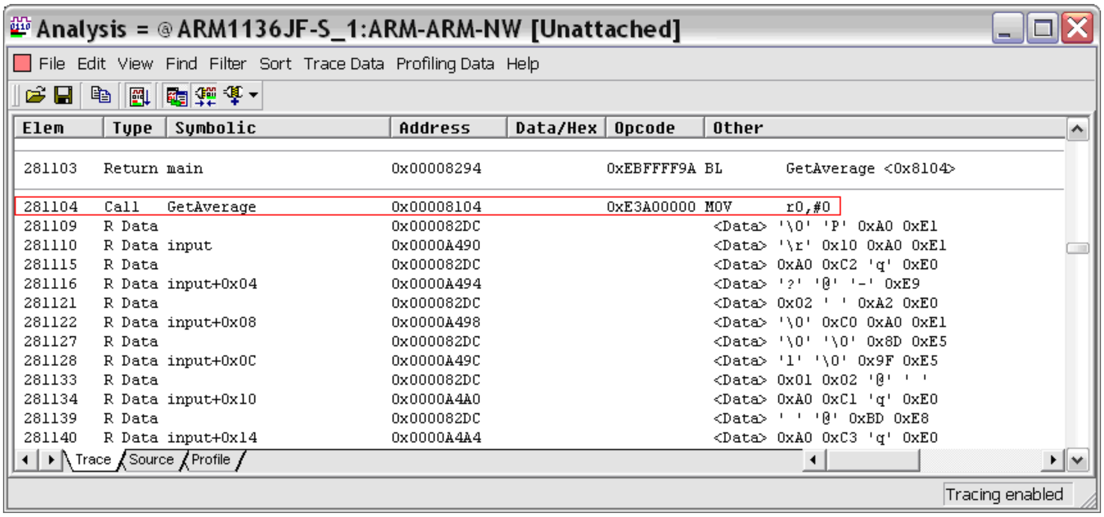
In the preceding Data Trace captures, both the data address and the value of the data were traced. RVD allows you to select what data is traced using the "Edit - Data Tracing Mode" menu as shown below.



```
Elem                                       Data/Hex     Opcode       Other

12938                                                   0xEBFFFF9A   BL      GetAverage <0x8104>

12939                                                   0xE3A00000   MOV     r0,#0
12944                                      0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
12945                                      0x000002BA                  <Data> 0xBA 0x02 '\0' '\0'
12950                                      0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
12951                                      0x00000721                  <Data> '!' '\a' '\0' '\0'
12956                                      0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
12957                                      0xFFFFF43A                  <Data> ':' 0xF4 0xFF 0xFF
12962                                      0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
12963                                                                  <Data> '$' 0x0F '\0' '\0'
12968                                                                  <Data> 0x90 0xA4 '\0' '\0'
12969                                                                  <Data> 0xAC 0x13 '\0' '\0'
12974                                                                  <Data> 0x90 0xA4 '\0' '\0'
12975                                                                  <Data> '2' 0x17 '\0' '\0'
12980                                      0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
12981                                      0x00000355                  <Data> 'U' 0x03 '\0' '\0'
12986                                      0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
12987    R Data input+0x1C     0x0000A4AC  0x0000072F                  <Data> '/' '\a' '\0' '\0'
12992    R Data                0x000082DC  0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
12993    R Data input+0x20     0x0000A4B0  0x000006EE                  <Data> 0xEE 0x06 '\0' '\0'
12998    R Data                0x000082DC  0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
12999    R Data input+0x24     0x0000A4B4  0x00000288                  <Data> 0x88 0x02 '\0' '\0'
13004    R Data                0x000082DC  0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
13005    R Data input+0x28     0x0000A4B8  0x000007FE                  <Data> 0xFE '\a' '\0' '\0'
13010    R Data                0x000082DC  0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
13011    R Data input+0x2C     0x0000A4BC  0x00000C04                  <Data> 0x04 '\f' '\0' '\0'
13016    R Data                0x000082DC  0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
13017    R Data input+0x30     0x0000A4C0  0x00001013                  <Data> 0x13 0x10 '\0' '\0'
13022    R Data                0x000082DC  0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
13023    R Data input+0x34     0x0000A4C4  0x00000666                  <Data> 'f' 0x06 '\0' '\0'
13028    R Data                0x000082DC  0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
13029    R Data input+0x38     0x0000A4C8  0x0000035C                  <Data> '\\' 0x03 '\0' '\0'
13034    R Data                0x000082DC  0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
13035    R Data input+0x3C     0x0000A4CC  0xFFFFF489                  <Data> 0x89 0xF4 0xFF 0xFF
13044    R Data                0x000082D0  0x0000A470                  <Data> 'p' 0xA4 '\0' '\0'
13045    W Data average        0x0000A470  0x000006E7                  <Data> 0xE7 0x06 '\0' '\0'

13047    Return main           0x00008298               0xEBFFFF81   BL      SendData <0x80a4>
```

Figure 65

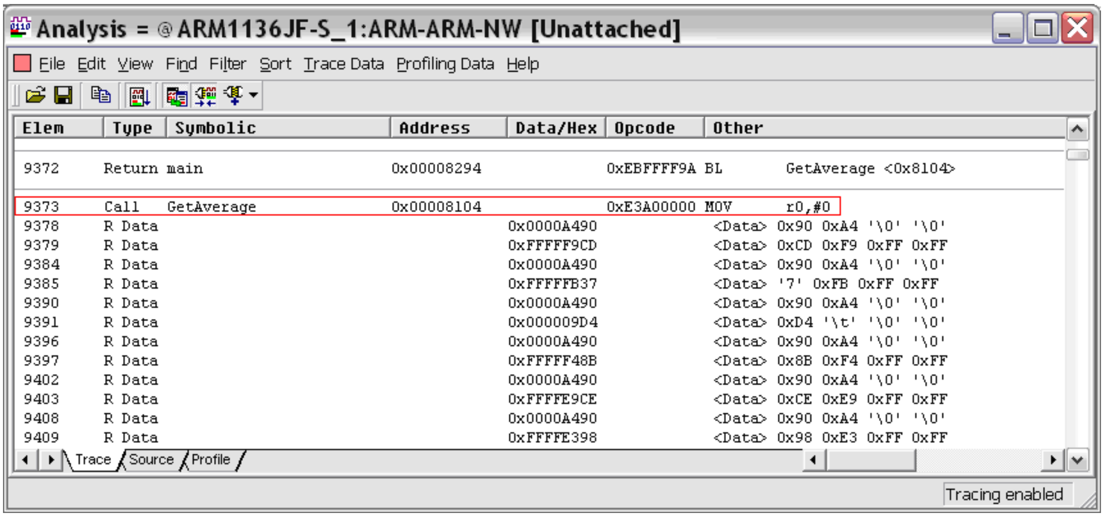
The default Data Tracing mode is "Data and Address". You can use the "Address Only" and "Data Only" options to reduce the amount of data collected by the ETM, and this will lessen the possibility of trace overflow. If the same capture is made with Data Tracing Mode set to "Address Only", data value information is not traced and the Data / Hex column is empty:


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

```
Elem     Type  Symbolic                   Address      Data/Hex  Opcode     Other

281103   Return main                      0x00008294             0xEBFFFF9A BL       GetAverage <0x8104>

281104   Call   GetAverage                0x00008104             0xE3A00000 MOV      r0,#0
281109   R Data                           0x000082DC                         <Data> '\0' 'P' 0xA0 0xE1
281110   R Data input                     0x0000A490                         <Data> '\r' 0x10 0xA0 0xE1
281115   R Data                           0x000082DC                         <Data> 0xA0 0xC2 'q' 0xE0
281116   R Data input+0x04                0x0000A494                         <Data> '?' '@' '-' 0xE9
281121   R Data                           0x000082DC                         <Data> 0x02 ' ' 0xA2 0xE0
281122   R Data input+0x08                0x0000A498                         <Data> '\0' 0xC0 0xA0 0xE1
281127   R Data                           0x000082DC                         <Data> '\0' '\0' 0x8D 0xE5
281128   R Data input+0x0C                0x0000A49C                         <Data> 'l' '\0' 0x9F 0xE5
281133   R Data                           0x000082DC                         <Data> 0x01 0x02 '@' ' '
281134   R Data input+0x10                0x0000A4A0                         <Data> 0xA0 0xC1 'q' 0xE0
281139   R Data                           0x000082DC                         <Data> ' ' '@' 0xBD 0xE8
281140   R Data input+0x14                0x0000A4A4                         <Data> 0xA0 0xC3 'q' 0xE0
```

Figure 66

If another capture is made with Data Tracing Mode set to "Data Only", data address information is not traced and the Address column is empty for all data accesses:


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

```
Elem     Type  Symbolic                   Address      Data/Hex  Opcode     Other

9372     Return main                      0x00008294             0xEBFFFF9A BL       GetAverage <0x8104>

9373     Call   GetAverage                0x00008104             0xE3A00000 MOV      r0,#0
9378     R Data                                        0x0000A490            <Data> 0x90 0xA4 '\0' '\0'
9379     R Data                                        0xFFFFF9CD            <Data> 0xCD 0xF9 0xFF 0xFF
9384     R Data                                        0x0000A490            <Data> 0x90 0xA4 '\0' '\0'
9385     R Data                                        0xFFFFFB37            <Data> '7' 0xFB 0xFF 0xFF
9390     R Data                                        0x0000A490            <Data> 0x90 0xA4 '\0' '\0'
9391     R Data                                        0x000009D4            <Data> 0xD4 '\t' '\0' '\0'
9396     R Data                                        0x0000A490            <Data> 0x90 0xA4 '\0' '\0'
9397     R Data                                        0xFFFFF48B            <Data> 0x8B 0xF4 0xFF 0xFF
9402     R Data                                        0x0000A490            <Data> 0x90 0xA4 '\0' '\0'
9403     R Data                                        0xFFFFE9CE            <Data> 0xCE 0xE9 0xFF 0xFF
9408     R Data                                        0x0000A490            <Data> 0x90 0xA4 '\0' '\0'
9409     R Data                                        0xFFFFE398            <Data> 0x98 0xE3 0xFF 0xFF
```

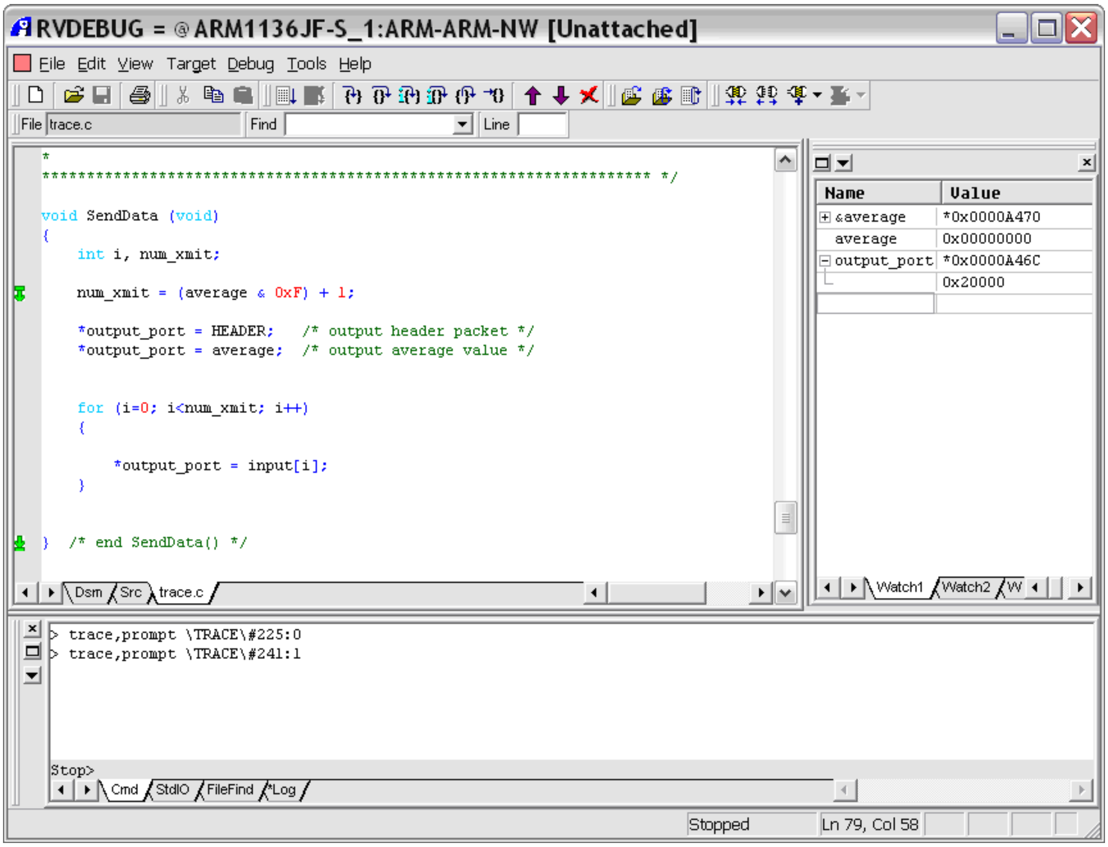
Figure 67

7.2 Tracing Data from an Instruction Trace Range

In Section 6 Trace Ranges were used to isolate the region of trace to a specified address range. This same trace operation can be used to also capture data accesses. This type of trace capture is valuable if you need to analyze how a particular segment of code is accessing data memory.

Remove any previous trace points and trace SendData() using the following Trace Range:

- Start of Trace Range (Instruction and Data) at opening brace
- End of Trace Range (Instruction and Data) at closing brace


RVDEBUG = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Target Debug Tools Help
File trace.c
Find
Line
Dsm
Src
trace.c
Watch1
Watch2
W
Cmd
StdIO
FileFind
Log
Stopped
Ln 79, Col 58

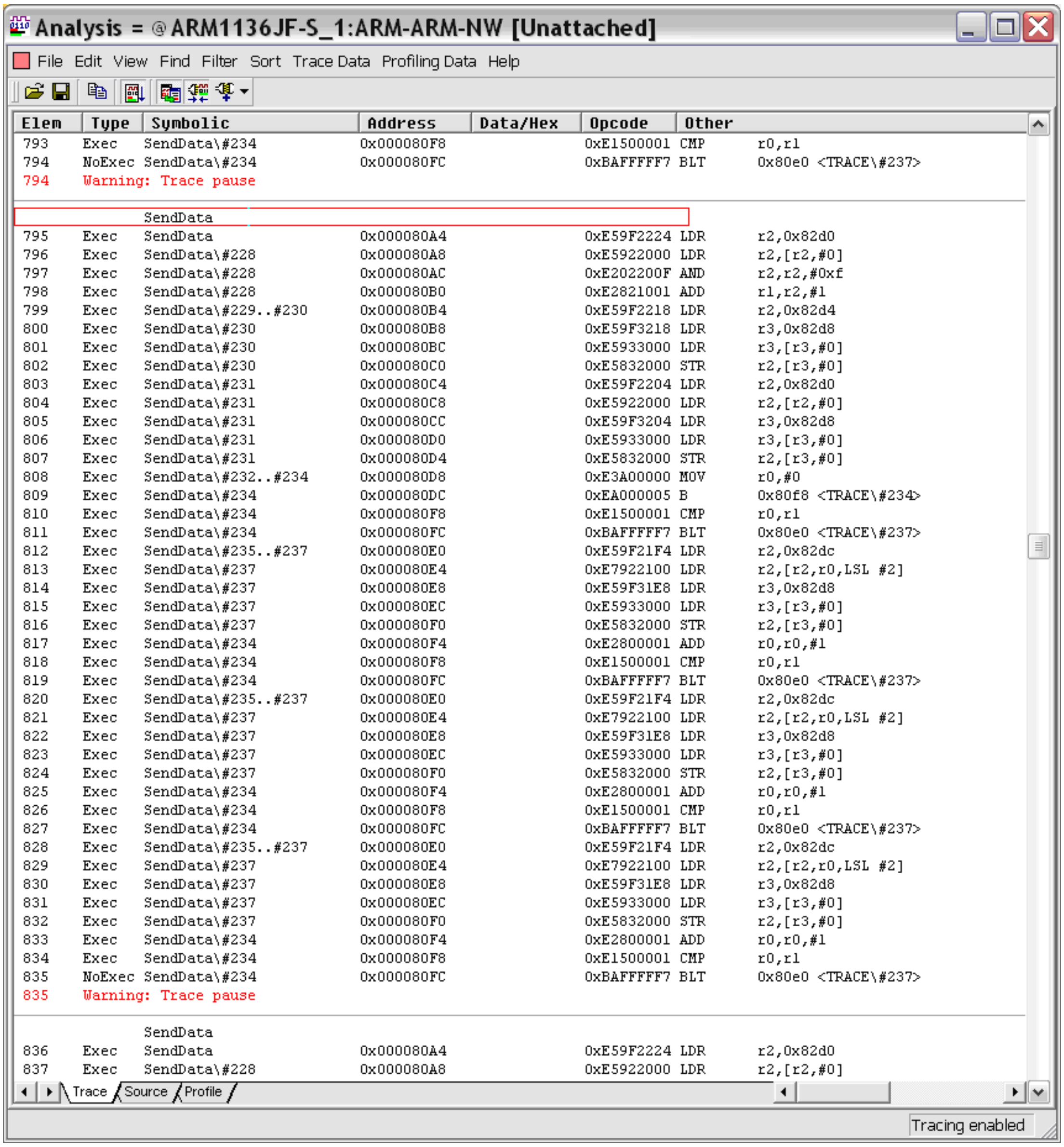
```
*
*********************************************************************** */

void SendData (void)
{
    int i, num_xmit;

    num_xmit = (average & 0xF) + 1;

    *output_port = HEADER;    /* output header packet */
    *output_port = average;   /* output average value */


    for (i=0; i<num_xmit; i++)
    {

        *output_port = input[i];
    }


}  /* end SendData() */
```

Name	Value
⊞ &average	*0x0000A470
average	0x00000000
⊟ output_port	*0x0000A46C
	0x20000

```
> trace,prompt \TRACE\#225:0
> trace,prompt \TRACE\#241:1

Stop>
```

Figure 68

Reload TRACE.AXF and change the Data Tracing Mode back to "Data and Address" if you modified it in Section 7.1. Run the program for a few seconds and stop it. Open the Analysis window. Enable "Instructions" and "Function Boundaries" and disable "Data" from the "Trace Data" menu and observe that trace is paused between runs of SendData():


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

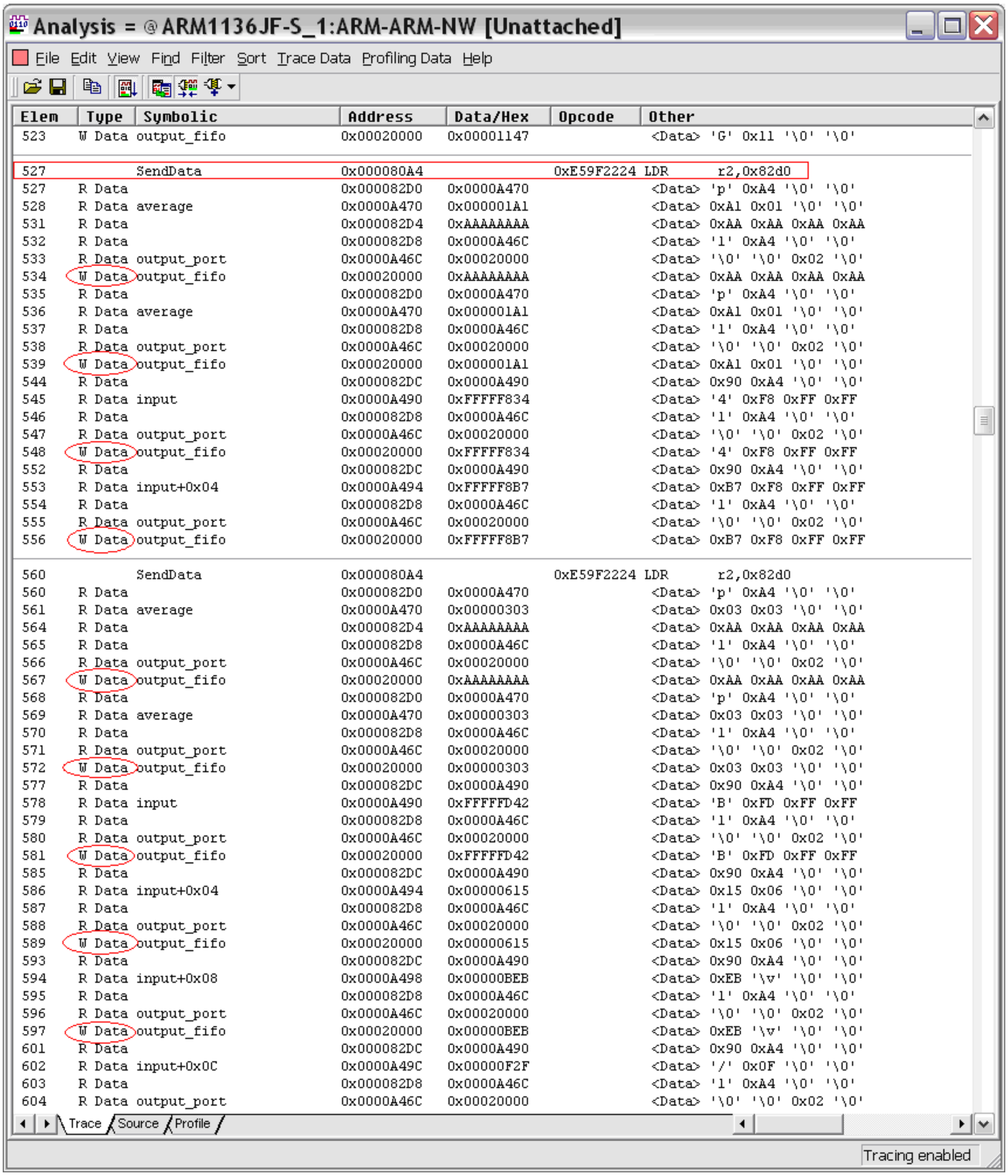
```
Elem  Type   Symbolic             Address     Data/Hex  Opcode          Other
793   Exec   SendData\#234        0x000080F8            0xE1500001 CMP  r0,r1
794   NoExec SendData\#234        0x000080FC            0xBAFFFFF7 BLT  0x80e0 <TRACE\#237>
794   Warning: Trace pause

             SendData
795   Exec   SendData             0x000080A4            0xE59F2224 LDR  r2,0x82d0
796   Exec   SendData\#228        0x000080A8            0xE5922000 LDR  r2,[r2,#0]
797   Exec   SendData\#228        0x000080AC            0xE202200F AND  r2,r2,#0xf
798   Exec   SendData\#228        0x000080B0            0xE2821001 ADD  r1,r2,#1
799   Exec   SendData\#229..#230  0x000080B4            0xE59F2218 LDR  r2,0x82d4
800   Exec   SendData\#230        0x000080B8            0xE59F3218 LDR  r3,0x82d8
801   Exec   SendData\#230        0x000080BC            0xE5933000 LDR  r3,[r3,#0]
802   Exec   SendData\#230        0x000080C0            0xE5832000 STR  r2,[r3,#0]
803   Exec   SendData\#231        0x000080C4            0xE59F2204 LDR  r2,0x82d0
804   Exec   SendData\#231        0x000080C8            0xE5922000 LDR  r2,[r2,#0]
805   Exec   SendData\#231        0x000080CC            0xE59F3204 LDR  r3,0x82d8
806   Exec   SendData\#231        0x000080D0            0xE5933000 LDR  r3,[r3,#0]
807   Exec   SendData\#231        0x000080D4            0xE5832000 STR  r2,[r3,#0]
808   Exec   SendData\#232..#234  0x000080D8            0xE3A00000 MOV  r0,#0
809   Exec   SendData\#234        0x000080DC            0xEA000005 B    0x80f8 <TRACE\#234>
810   Exec   SendData\#234        0x000080F8            0xE1500001 CMP  r0,r1
811   Exec   SendData\#234        0x000080FC            0xBAFFFFF7 BLT  0x80e0 <TRACE\#237>
812   Exec   SendData\#235..#237  0x000080E0            0xE59F21F4 LDR  r2,0x82dc
813   Exec   SendData\#237        0x000080E4            0xE7922100 LDR  r2,[r2,r0,LSL #2]
814   Exec   SendData\#237        0x000080E8            0xE59F31E8 LDR  r3,0x82d8
815   Exec   SendData\#237        0x000080EC            0xE5933000 LDR  r3,[r3,#0]
816   Exec   SendData\#237        0x000080F0            0xE5832000 STR  r2,[r3,#0]
817   Exec   SendData\#234        0x000080F4            0xE2800001 ADD  r0,r0,#1
818   Exec   SendData\#234        0x000080F8            0xE1500001 CMP  r0,r1
819   Exec   SendData\#234        0x000080FC            0xBAFFFFF7 BLT  0x80e0 <TRACE\#237>
820   Exec   SendData\#235..#237  0x000080E0            0xE59F21F4 LDR  r2,0x82dc
821   Exec   SendData\#237        0x000080E4            0xE7922100 LDR  r2,[r2,r0,LSL #2]
822   Exec   SendData\#237        0x000080E8            0xE59F31E8 LDR  r3,0x82d8
823   Exec   SendData\#237        0x000080EC            0xE5933000 LDR  r3,[r3,#0]
824   Exec   SendData\#237        0x000080F0            0xE5832000 STR  r2,[r3,#0]
825   Exec   SendData\#234        0x000080F4            0xE2800001 ADD  r0,r0,#1
826   Exec   SendData\#234        0x000080F8            0xE1500001 CMP  r0,r1
827   Exec   SendData\#234        0x000080FC            0xBAFFFFF7 BLT  0x80e0 <TRACE\#237>
828   Exec   SendData\#235..#237  0x000080E0            0xE59F21F4 LDR  r2,0x82dc
829   Exec   SendData\#237        0x000080E4            0xE7922100 LDR  r2,[r2,r0,LSL #2]
830   Exec   SendData\#237        0x000080E8            0xE59F31E8 LDR  r3,0x82d8
831   Exec   SendData\#237        0x000080EC            0xE5933000 LDR  r3,[r3,#0]
832   Exec   SendData\#237        0x000080F0            0xE5832000 STR  r2,[r3,#0]
833   Exec   SendData\#234        0x000080F4            0xE2800001 ADD  r0,r0,#1
834   Exec   SendData\#234        0x000080F8            0xE1500001 CMP  r0,r1
835   NoExec SendData\#234        0x000080FC            0xBAFFFFF7 BLT  0x80e0 <TRACE\#237>
835   Warning: Trace pause

             SendData
836   Exec   SendData             0x000080A4            0xE59F2224 LDR  r2,0x82d0
837   Exec   SendData\#228        0x000080A8            0xE5922000 LDR  r2,[r2,#0]
```

Figure 69

Note

Your trace capture may differ.

Now enable "Data" and disable "Instructions" from the "Trace Data" menu. Scroll through the buffer and observe that each time SendData() runs there is not a consistent number of words written to output_fifo (address 0x20000):


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

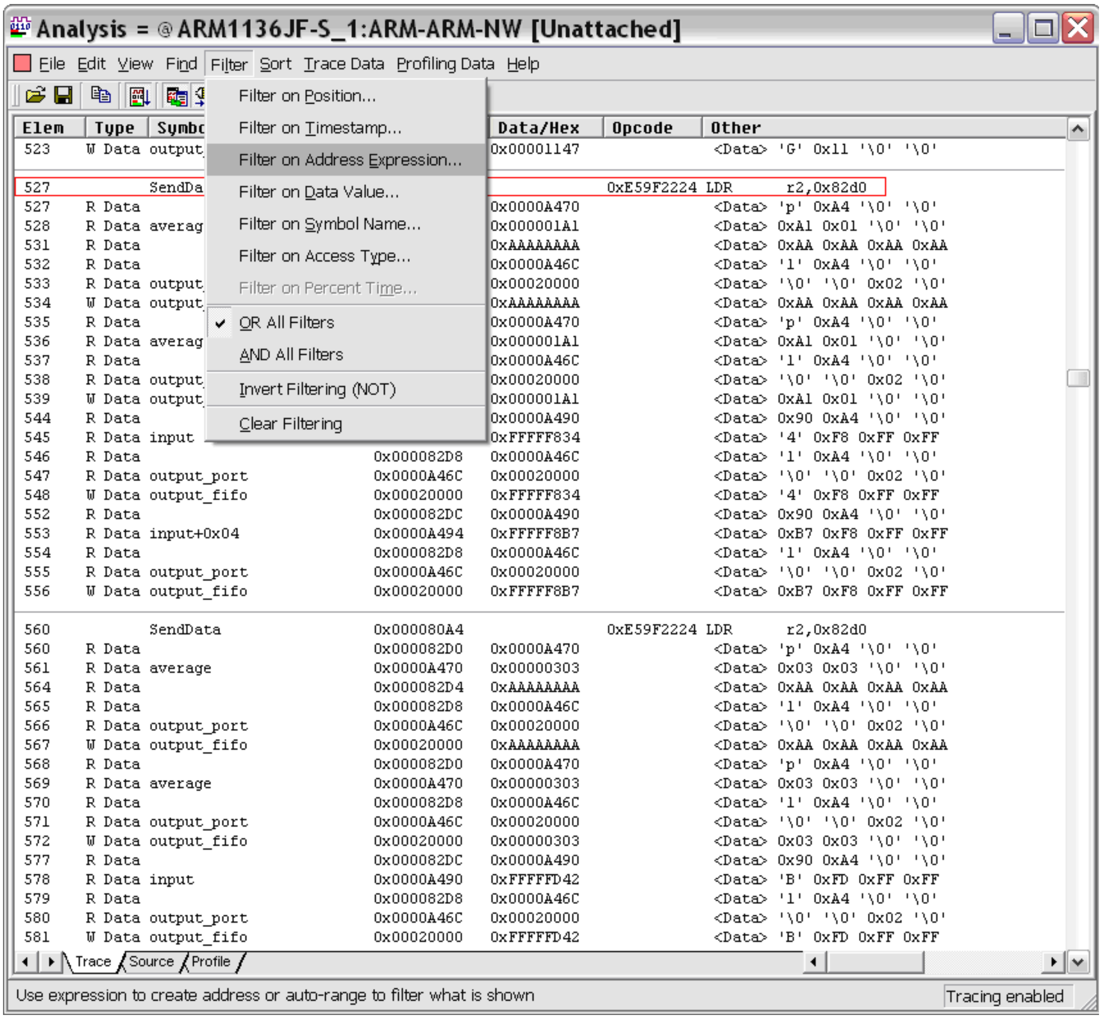
```
Elem  Type   Symbolic                 Address     Data/Hex    Opcode    Other
523   W Data output_fifo              0x00020000  0x00001147            <Data> 'G' 0x11 '\0' '\0'

527          SendData                 0x000080A4              0xE59F2224 LDR      r2,0x82d0
527   R Data                          0x000082D0  0x0000A470            <Data> 'p' 0xA4 '\0' '\0'
528   R Data average                  0x0000A470  0x000001A1            <Data> 0xA1 0x01 '\0' '\0'
531   R Data                          0x000082D4  0xAAAAAAAA            <Data> 0xAA 0xAA 0xAA 0xAA
532   R Data                          0x000082D8  0x0000A46C            <Data> '1' 0xA4 '\0' '\0'
533   R Data output_port              0x0000A46C  0x00020000            <Data> '\0' '\0' 0x02 '\0'
534   W Data output_fifo              0x00020000  0xAAAAAAAA            <Data> 0xAA 0xAA 0xAA 0xAA
535   R Data                          0x000082D0  0x0000A470            <Data> 'p' 0xA4 '\0' '\0'
536   R Data average                  0x0000A470  0x000001A1            <Data> 0xA1 0x01 '\0' '\0'
537   R Data                          0x000082D8  0x0000A46C            <Data> '1' 0xA4 '\0' '\0'
538   R Data output_port              0x0000A46C  0x00020000            <Data> '\0' '\0' 0x02 '\0'
539   W Data output_fifo              0x00020000  0x000001A1            <Data> 0xA1 0x01 '\0' '\0'
544   R Data                          0x000082DC  0x0000A490            <Data> 0x90 0xA4 '\0' '\0'
545   R Data input                    0x0000A490  0xFFFFF834            <Data> '4' 0xF8 0xFF 0xFF
546   R Data                          0x000082D8  0x0000A46C            <Data> '1' 0xA4 '\0' '\0'
547   R Data output_port              0x0000A46C  0x00020000            <Data> '\0' '\0' 0x02 '\0'
548   W Data output_fifo              0x00020000  0xFFFFF834            <Data> '4' 0xF8 0xFF 0xFF
552   R Data                          0x000082DC  0x0000A490            <Data> 0x90 0xA4 '\0' '\0'
553   R Data input+0x04               0x0000A494  0xFFFFF8B7            <Data> 0xB7 0xF8 0xFF 0xFF
554   R Data                          0x000082D8  0x0000A46C            <Data> '1' 0xA4 '\0' '\0'
555   R Data output_port              0x0000A46C  0x00020000            <Data> '\0' '\0' 0x02 '\0'
556   W Data output_fifo              0x00020000  0xFFFFF8B7            <Data> 0xB7 0xF8 0xFF 0xFF

560          SendData                 0x000080A4              0xE59F2224 LDR      r2,0x82d0
560   R Data                          0x000082D0  0x0000A470            <Data> 'p' 0xA4 '\0' '\0'
561   R Data average                  0x0000A470  0x00000303            <Data> 0x03 0x03 '\0' '\0'
564   R Data                          0x000082D4  0xAAAAAAAA            <Data> 0xAA 0xAA 0xAA 0xAA
565   R Data                          0x000082D8  0x0000A46C            <Data> '1' 0xA4 '\0' '\0'
566   R Data output_port              0x0000A46C  0x00020000            <Data> '\0' '\0' 0x02 '\0'
567   W Data output_fifo              0x00020000  0xAAAAAAAA            <Data> 0xAA 0xAA 0xAA 0xAA
568   R Data                          0x000082D0  0x0000A470            <Data> 'p' 0xA4 '\0' '\0'
569   R Data average                  0x0000A470  0x00000303            <Data> 0x03 0x03 '\0' '\0'
570   R Data                          0x000082D8  0x0000A46C            <Data> '1' 0xA4 '\0' '\0'
571   R Data output_port              0x0000A46C  0x00020000            <Data> '\0' '\0' 0x02 '\0'
572   W Data output_fifo              0x00020000  0x00000303            <Data> 0x03 0x03 '\0' '\0'
577   R Data                          0x000082DC  0x0000A490            <Data> 0x90 0xA4 '\0' '\0'
578   R Data input                    0x0000A490  0xFFFFFD42            <Data> 'B' 0xFD 0xFF 0xFF
579   R Data                          0x000082D8  0x0000A46C            <Data> '1' 0xA4 '\0' '\0'
580   R Data output_port              0x0000A46C  0x00020000            <Data> '\0' '\0' 0x02 '\0'
581   W Data output_fifo              0x00020000  0xFFFFFD42            <Data> 'B' 0xFD 0xFF 0xFF
585   R Data                          0x000082DC  0x0000A490            <Data> 0x90 0xA4 '\0' '\0'
586   R Data input+0x04               0x0000A494  0x00000615            <Data> 0x15 0x06 '\0' '\0'
587   R Data                          0x000082D8  0x0000A46C            <Data> '1' 0xA4 '\0' '\0'
588   R Data output_port              0x0000A46C  0x00020000            <Data> '\0' '\0' 0x02 '\0'
589   W Data output_fifo              0x00020000  0x00000615            <Data> 0x15 0x06 '\0' '\0'
593   R Data                          0x000082DC  0x0000A490            <Data> 0x90 0xA4 '\0' '\0'
594   R Data input+0x08               0x0000A498  0x00000BEB            <Data> 0xEB '\v' '\0' '\0'
595   R Data                          0x000082D8  0x0000A46C            <Data> '1' 0xA4 '\0' '\0'
596   R Data output_port              0x0000A46C  0x00020000            <Data> '\0' '\0' 0x02 '\0'
597   W Data output_fifo              0x00020000  0x00000BEB            <Data> 0xEB '\v' '\0' '\0'
601   R Data                          0x000082DC  0x0000A490            <Data> 0x90 0xA4 '\0' '\0'
602   R Data input+0x0C               0x0000A49C  0x00000F2F            <Data> '/' 0x0F '\0' '\0'
603   R Data                          0x000082D8  0x0000A46C            <Data> '1' 0xA4 '\0' '\0'
604   R Data output_port              0x0000A46C  0x00020000            <Data> '\0' '\0' 0x02 '\0'
```

Figure 70

If you reference TRACE.C, you will find that SendData() transmits a packet header (0xAAAAAAAA), the computed average and a variable number of input samples. The number of input samples written in SendData(), num_xmit, is determined by the computed average:

```
num_xmit = (average & 0xF) + 1;
```

You can further isolate accesses to a particular address by applying a post-processing filter. Apply a filter to examine all accesses to 0x20000. To do this, use the "Filter - Filter on Address Expression" menu:

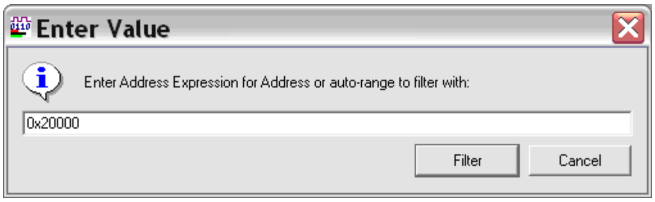


```
Elem    Type    Symbo                             Data/Hex    Opcode          Other
523     W Data  output                            0x00001147                  <Data> 'G' 0x11 '\0' '\0'

527             SendDa                                        0xE59F2224 LDR      r2,0x82d0
527     R Data                                    0x0000A470                  <Data> 'p' 0xA4 '\0' '\0'
528     R Data  averag                            0x000001A1                  <Data> 0xA1 0x01 '\0' '\0'
531     R Data                                    0xAAAAAAAA                  <Data> 0xAA 0xAA 0xAA 0xAA
532     R Data                                    0x0000A46C                  <Data> '1' 0xA4 '\0' '\0'
533     R Data  output                            0x00020000                  <Data> '\0' '\0' 0x02 '\0'
534     W Data  output                            0xAAAAAAAA                  <Data> 0xAA 0xAA 0xAA 0xAA
535     R Data                                    0x0000A470                  <Data> 'p' 0xA4 '\0' '\0'
536     R Data  averag                            0x000001A1                  <Data> 0xA1 0x01 '\0' '\0'
537     R Data                                    0x0000A46C                  <Data> '1' 0xA4 '\0' '\0'
538     R Data  output                            0x00020000                  <Data> '\0' '\0' 0x02 '\0'
539     W Data  output                            0x000001A1                  <Data> 0xA1 0x01 '\0' '\0'
544     R Data                                    0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
545     R Data  input                             0xFFFFF834                  <Data> '4' 0xF8 0xFF 0xFF
546     R Data                        0x000082D8  0x0000A46C                  <Data> '1' 0xA4 '\0' '\0'
547     R Data  output_port           0x0000A46C  0x00020000                  <Data> '\0' '\0' 0x02 '\0'
548     W Data  output_fifo           0x00020000  0xFFFFF834                  <Data> '4' 0xF8 0xFF 0xFF
552     R Data                        0x000082DC  0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
553     R Data  input+0x04            0x0000A494  0xFFFFF8B7                  <Data> 0xB7 0xF8 0xFF 0xFF
554     R Data                        0x000082D8  0x0000A46C                  <Data> '1' 0xA4 '\0' '\0'
555     R Data  output_port           0x0000A46C  0x00020000                  <Data> '\0' '\0' 0x02 '\0'
556     W Data  output_fifo           0x00020000  0xFFFFF8B7                  <Data> 0xB7 0xF8 0xFF 0xFF

560             SendData              0x000080A4              0xE59F2224 LDR      r2,0x82d0
560     R Data                        0x000082D0  0x0000A470                  <Data> 'p' 0xA4 '\0' '\0'
561     R Data  average               0x0000A470  0x00000303                  <Data> 0x03 0x03 '\0' '\0'
564     R Data                        0x000082D4  0xAAAAAAAA                  <Data> 0xAA 0xAA 0xAA 0xAA
565     R Data                        0x000082D8  0x0000A46C                  <Data> '1' 0xA4 '\0' '\0'
566     R Data  output_port           0x0000A46C  0x00020000                  <Data> '\0' '\0' 0x02 '\0'
567     W Data  output_fifo           0x00020000  0xAAAAAAAA                  <Data> 0xAA 0xAA 0xAA 0xAA
568     R Data                        0x000082D0  0x0000A470                  <Data> 'p' 0xA4 '\0' '\0'
569     R Data  average               0x0000A470  0x00000303                  <Data> 0x03 0x03 '\0' '\0'
570     R Data                        0x000082D8  0x0000A46C                  <Data> '1' 0xA4 '\0' '\0'
571     R Data  output_port           0x0000A46C  0x00020000                  <Data> '\0' '\0' 0x02 '\0'
572     W Data  output_fifo           0x00020000  0x00000303                  <Data> 0x03 0x03 '\0' '\0'
577     R Data                        0x000082DC  0x0000A490                  <Data> 0x90 0xA4 '\0' '\0'
578     R Data  input                 0x0000A490  0xFFFFFD42                  <Data> 'B' 0xFD 0xFF 0xFF
579     R Data                        0x000082D8  0x0000A46C                  <Data> '1' 0xA4 '\0' '\0'
580     R Data  output_port           0x0000A46C  0x00020000                  <Data> '\0' '\0' 0x02 '\0'
581     W Data  output_fifo           0x00020000  0xFFFFFD42                  <Data> 'B' 0xFD 0xFF 0xFF
```

Use expression to create address or auto-range to filter what is shown

Figure 71

Enter 0x20000 from the Address Expression dialogue:

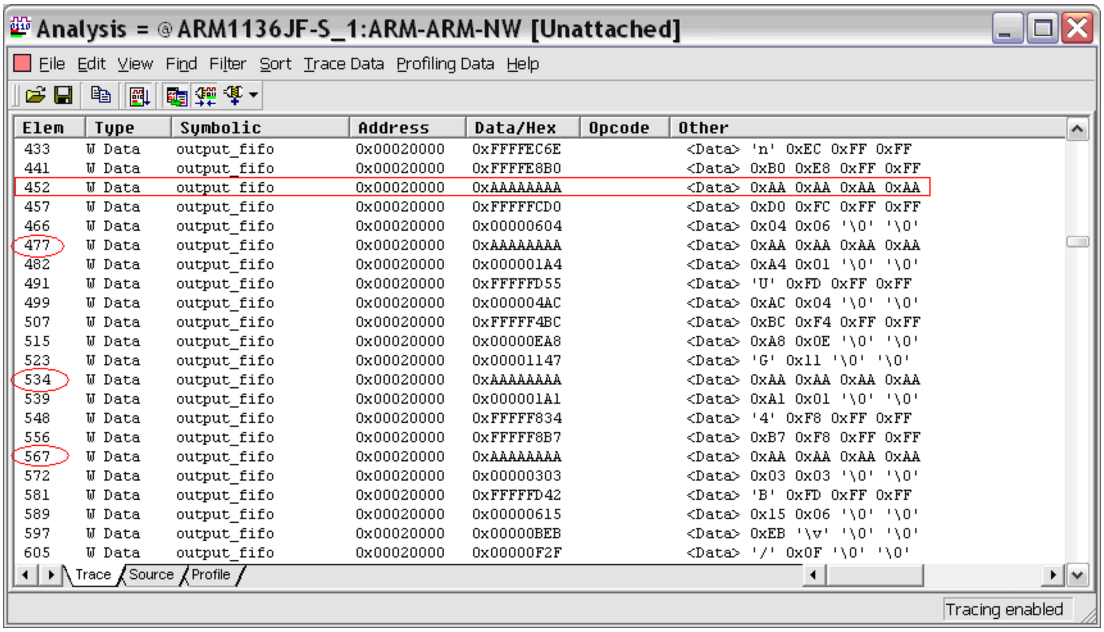


Enter Address Expression for Address or auto-range to filter with:

0x20000

Figure 72

You can now more readily observe all the accesses to 0x20000. Find a trace element that writes the packet header word, 0xAAAAAAAA. Observe that each packet consists of the header word, the computed average and 1 to 16 input samples (based on the least-significant nibble of the average).


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Type	Symbolic	Address	Data/Hex	Opcode	Other
433	W Data	output_fifo	0x00020000	0xFFFFEC6E		<Data> 'n' 0xEC 0xFF 0xFF
441	W Data	output_fifo	0x00020000	0xFFFFE8B0		<Data> 0xB0 0xE8 0xFF 0xFF
452	W Data	output_fifo	0x00020000	0xAAAAAAAA		<Data> 0xAA 0xAA 0xAA 0xAA
457	W Data	output_fifo	0x00020000	0xFFFFFCD0		<Data> 0xD0 0xFC 0xFF 0xFF
466	W Data	output_fifo	0x00020000	0x00000604		<Data> 0x04 0x06 '\0' '\0'
477	W Data	output_fifo	0x00020000	0xAAAAAAAA		<Data> 0xAA 0xAA 0xAA 0xAA
482	W Data	output_fifo	0x00020000	0x000001A4		<Data> 0xA4 0x01 '\0' '\0'
491	W Data	output_fifo	0x00020000	0xFFFFFD55		<Data> 'U' 0xFD 0xFF 0xFF
499	W Data	output_fifo	0x00020000	0x000004AC		<Data> 0xAC 0x04 '\0' '\0'
507	W Data	output_fifo	0x00020000	0xFFFFF4BC		<Data> 0xBC 0xF4 0xFF 0xFF
515	W Data	output_fifo	0x00020000	0x00000EA8		<Data> 0xA8 0x0E '\0' '\0'
523	W Data	output_fifo	0x00020000	0x00001147		<Data> 'G' 0x11 '\0' '\0'
534	W Data	output_fifo	0x00020000	0xAAAAAAAA		<Data> 0xAA 0xAA 0xAA 0xAA
539	W Data	output_fifo	0x00020000	0x000001A1		<Data> 0xA1 0x01 '\0' '\0'
548	W Data	output_fifo	0x00020000	0xFFFFF834		<Data> '4' 0xF8 0xFF 0xFF
556	W Data	output_fifo	0x00020000	0xFFFFF8B7		<Data> 0xB7 0xF8 0xFF 0xFF
567	W Data	output_fifo	0x00020000	0xAAAAAAAA		<Data> 0xAA 0xAA 0xAA 0xAA
572	W Data	output_fifo	0x00020000	0x00000303		<Data> 0x03 0x03 '\0' '\0'
581	W Data	output_fifo	0x00020000	0xFFFFFD42		<Data> 'B' 0xFD 0xFF 0xFF
589	W Data	output_fifo	0x00020000	0x00000615		<Data> 0x15 0x06 '\0' '\0'
597	W Data	output_fifo	0x00020000	0x00000BEB		<Data> 0xEB '\v' '\0' '\0'
605	W Data	output_fifo	0x00020000	0x00000F2F		<Data> '/' 0x0F '\0' '\0'

Figure 73

In the trace above, SendData() has run four times:

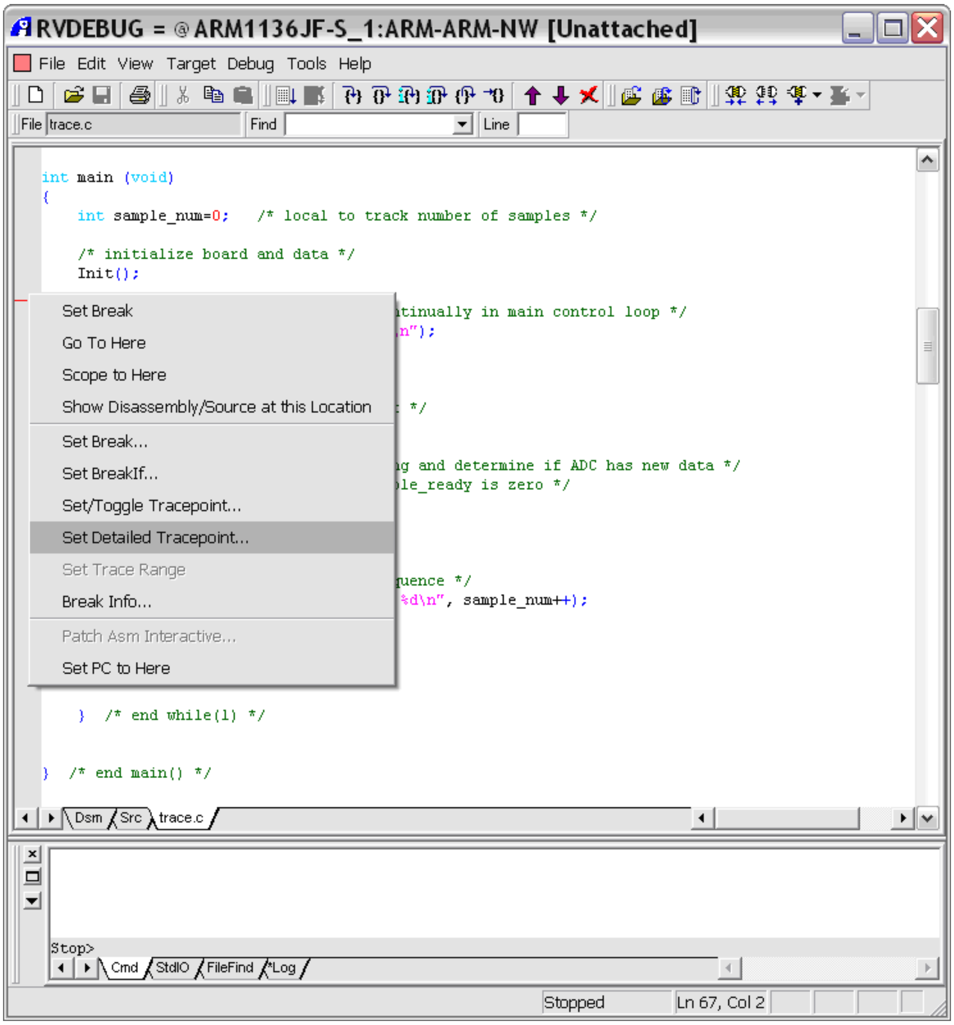
Table 2

Header Element	Average	Input Words Written	Total Words Written
452	0xFFFFFCD0	1	3
477	0x000001A4	5	7
534	0x000001A1	2	4
567	0x00000303	4	6

7.3 Tracing Data Accesses to a Specific Address

In the previous scenario, data trace was captured based on execution from a specified address range. This type of capture is useful for most situations. However, an alternate method of data trace is provided by the ETM based solely on data accesses (not instruction accesses). This allows you to trace specific data accesses made from anywhere in your program. You may find this type of trace useful if you have a global variable that is corrupted by a stray write or if you want to see how frequently a variable is accessed.

In this scenario only accesses to *output_port (address 0x20000) are traced. You can configure this type of capture using a Detailed Trace Point. Delete all previous trace points and reload TRACE.AXF. Click on the left gutter of the Source window and select "Set Detailed Tracepoint".



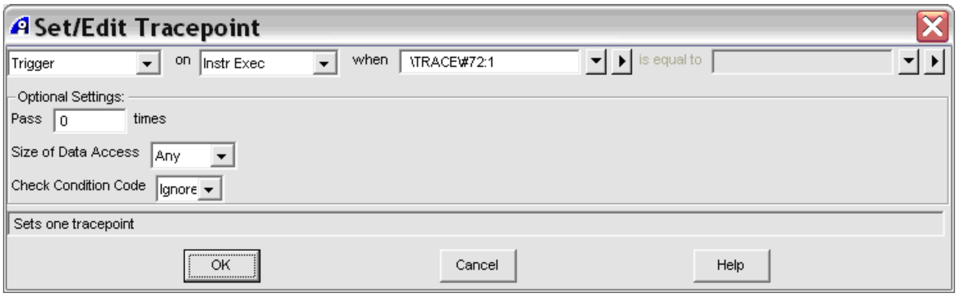
```
int main (void)
{
    int sample_num=0;   /* local to track number of samples */

    /* initialize board and data */
    Init();

                                          tinually in main control loop */
                                          n");


                                          */

                                          g and determine if ADC has new data */
                                          le_ready is zero */



                                          uence */
                                          %d\n", sample_num++);


    }  /* end while(1) */

}  /* end main() */
```

```
Stop>
```

Stopped   Ln 67, Col 2

Figure 74

The Set/Edit Tracepoint dialogue will open. This allows you to set both simple and detailed trace points:



Trigger on Instr Exec when \TRACE\#72:1 is equal to

Optional Settings:

Pass 0 times

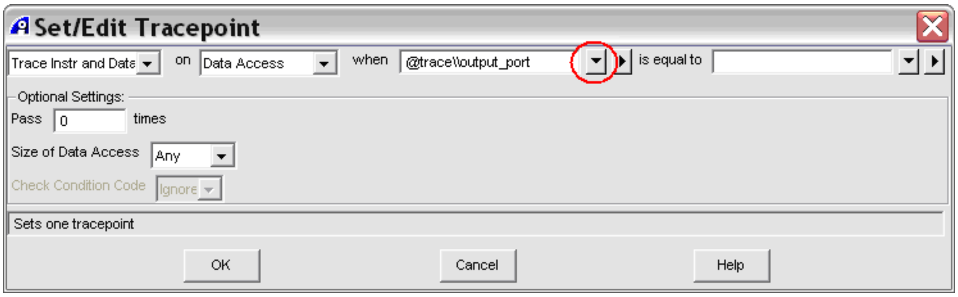
Size of Data Access Any

Check Condition Code Ignore

Sets one tracepoint

Figure 75

Use this dialogue to configure an instruction and data trace for all accesses of output_port, the pointer to output_fifo. The top three parameters need to be modified:



Set/Edit Tracepoint

Trace Instr and Data on Data Access when @trace\output_port is equal to

Optional Settings:

Pass 0 times

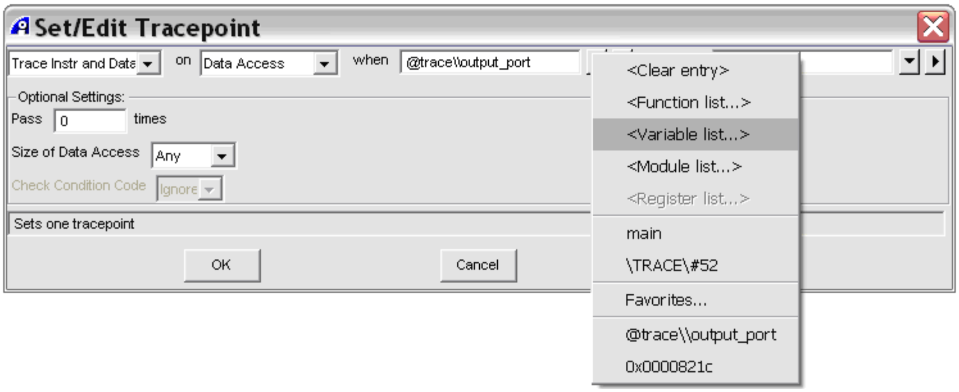
Size of Data Access Any

Check Condition Code Ignore

Sets one tracepoint

Figure 76

Use the circled down selector to open the <Variable list…> menu and browse for "output_port of @trace". When this variable is selected it will appear as "@trace\\output_port" in the dialogue:



Set/Edit Tracepoint

Trace Instr and Data on Data Access when @trace\output_port

Optional Settings:

Pass 0 times

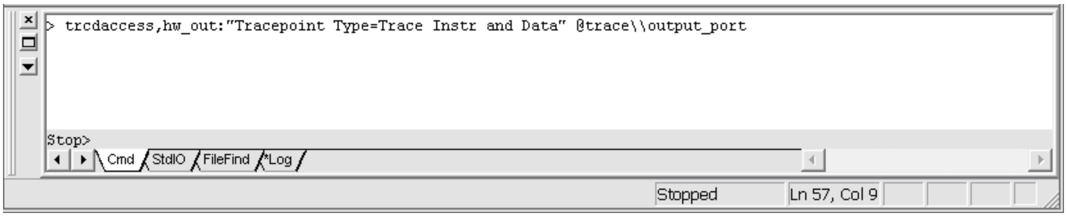
Size of Data Access Any

Check Condition Code Ignore

Sets one tracepoint

Figure 77

Click the "OK" button to save the trace point. Observe that the Cmd tab of the output window displays the command used to set the trace point. If you prefer, you may manually enter trace points using the command line from the Cmd tab of the output window. A full description of the command line interface is provided in the RealView Debugger Command Line Reference Guide.

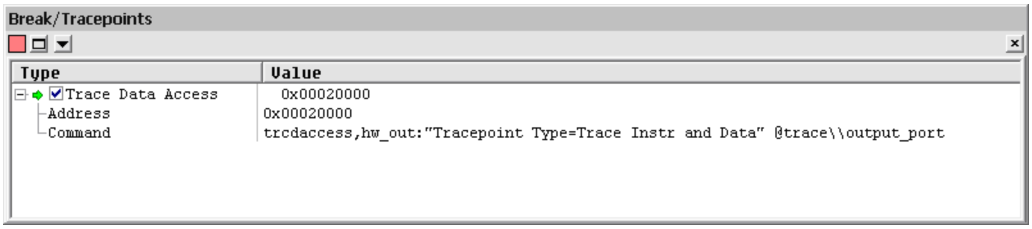


```
> trcdaccess,hw_out:"Tracepoint Type=Trace Instr and Data" @trace\\output_port



Stop>
```

Stopped Ln 57, Col 9

Figure 78

Trace points can be viewed, modified and cleared using the "View - Break/Tracepoints" menu from RVD. View the trace point you just set to confirm that it is set correctly. Since output_port is a pointer, all accesses to the value of the pointer when the trace point was set (0x20000) will be traced.

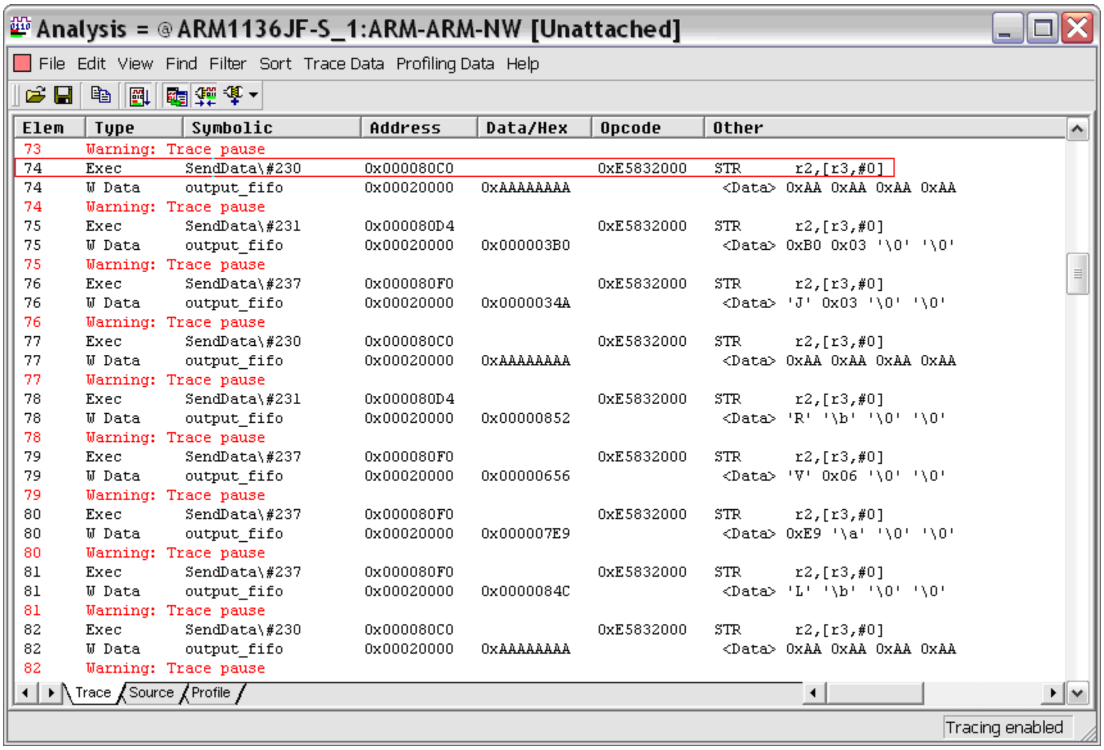


Type	Value
☑ Trace Data Access	0x00020000
Address	0x00020000
Command	trcdaccess,hw_out:"Tracepoint Type=Trace Instr and Data" @trace\\output_port

Figure 79

Run the program for a few seconds and stop it. Open the Analysis window and configure the view of the trace capture. If you applied a trace filter from Section 7.2, remove the filter using the "Filter - Clear Filtering" menu from the Analysis window. Also enable the Data/Hex and Instruction trace view by selecting "Data Value in Hex" and "Instructions" in the "Trace Data" menu (see Section 7.1).

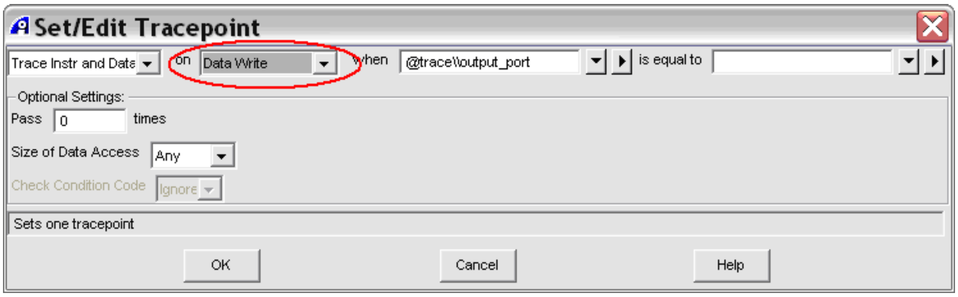
Now scroll through the trace buffer and observe that only three instructions from SendData() are traced. Trace is paused between each traced instruction. The three instructions are all STR instructions from address 0x80C0 (which writes the packet header), 0x80D4 (which writes the average) and 0x80F0 (which writes the input samples):


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Type	Symbolic	Address	Data/Hex	Opcode	Other
73	Warning: Trace pause					
74	Exec	SendData\#230	0x000080C0		0xE5832000	STR r2,[r3,#0]
74	W Data	output_fifo	0x00020000	0xAAAAAAAA		<Data> 0xAA 0xAA 0xAA 0xAA
74	Warning: Trace pause					
75	Exec	SendData\#231	0x000080D4		0xE5832000	STR r2,[r3,#0]
75	W Data	output_fifo	0x00020000	0x000003B0		<Data> 0xB0 0x03 '\0' '\0'
75	Warning: Trace pause					
76	Exec	SendData\#237	0x000080F0		0xE5832000	STR r2,[r3,#0]
76	W Data	output_fifo	0x00020000	0x0000034A		<Data> 'J' 0x03 '\0' '\0'
76	Warning: Trace pause					
77	Exec	SendData\#230	0x000080C0		0xE5832000	STR r2,[r3,#0]
77	W Data	output_fifo	0x00020000	0xAAAAAAAA		<Data> 0xAA 0xAA 0xAA 0xAA
77	Warning: Trace pause					
78	Exec	SendData\#231	0x000080D4		0xE5832000	STR r2,[r3,#0]
78	W Data	output_fifo	0x00020000	0x00000852		<Data> 'R' '\b' '\0' '\0'
78	Warning: Trace pause					
79	Exec	SendData\#237	0x000080F0		0xE5832000	STR r2,[r3,#0]
79	W Data	output_fifo	0x00020000	0x00000656		<Data> 'V' 0x06 '\0' '\0'
79	Warning: Trace pause					
80	Exec	SendData\#237	0x000080F0		0xE5832000	STR r2,[r3,#0]
80	W Data	output_fifo	0x00020000	0x000007E9		<Data> 0xE9 '\a' '\0' '\0'
80	Warning: Trace pause					
81	Exec	SendData\#237	0x000080F0		0xE5832000	STR r2,[r3,#0]
81	W Data	output_fifo	0x00020000	0x0000084C		<Data> 'L' '\b' '\0' '\0'
81	Warning: Trace pause					
82	Exec	SendData\#230	0x000080C0		0xE5832000	STR r2,[r3,#0]
82	W Data	output_fifo	0x00020000	0xAAAAAAAA		<Data> 0xAA 0xAA 0xAA 0xAA
82	Warning: Trace pause					

Figure 80

This capture was configured to trace all accesses to *output_port. RVD allows one to specify the data access type - either a read or write. In this program only writes are made to *output_port, so the same trace capture can be obtained using a trace point that specifies "Data Write" instead of "Data Access":



Set/Edit Tracepoint

Trace Instr and Data on Data Write when @trace\\output_port is equal to

Optional Settings:

Pass 0 times

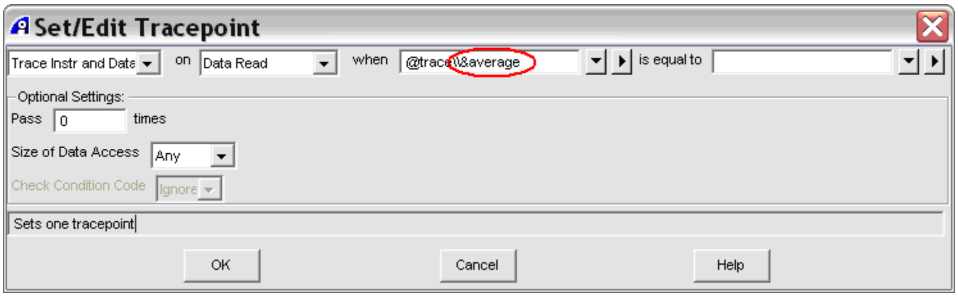
Size of Data Access Any

Check Condition Code Ignore

Sets one tracepoint

Figure 81

The variable "average" is both read and written in the program. To trace only the reads of "average", delete the previous trace point and set the following trace point:



Set/Edit Tracepoint

Trace Instr and Data on Data Read when @trace\\&average is equal to

Optional Settings:

Pass 0 times

Size of Data Access Any

Check Condition Code Ignore

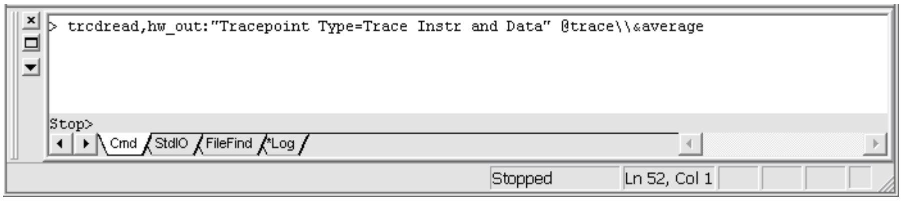
Sets one tracepoint

Figure 82

Note

To trace accesses to a variable, you must manually precede the symbol name with an ampersand to de-reference the address of the variable. To trace accesses to average, you must select "average @trace" from the variable browser. This will be displayed as "@trace\\average" in the dialogue. You must then manually add the ampersand to the "when" field to create "@trace\\&average".

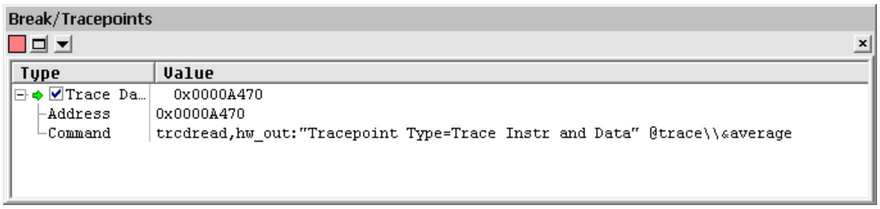
This results in the following command:



```
> trcdread,hw_out:"Tracepoint Type=Trace Instr and Data" @trace\\&average

Stop>
```

Figure 83

Confirm that the trace point is properly set using "View - Break/Tracepoints" from RVD :

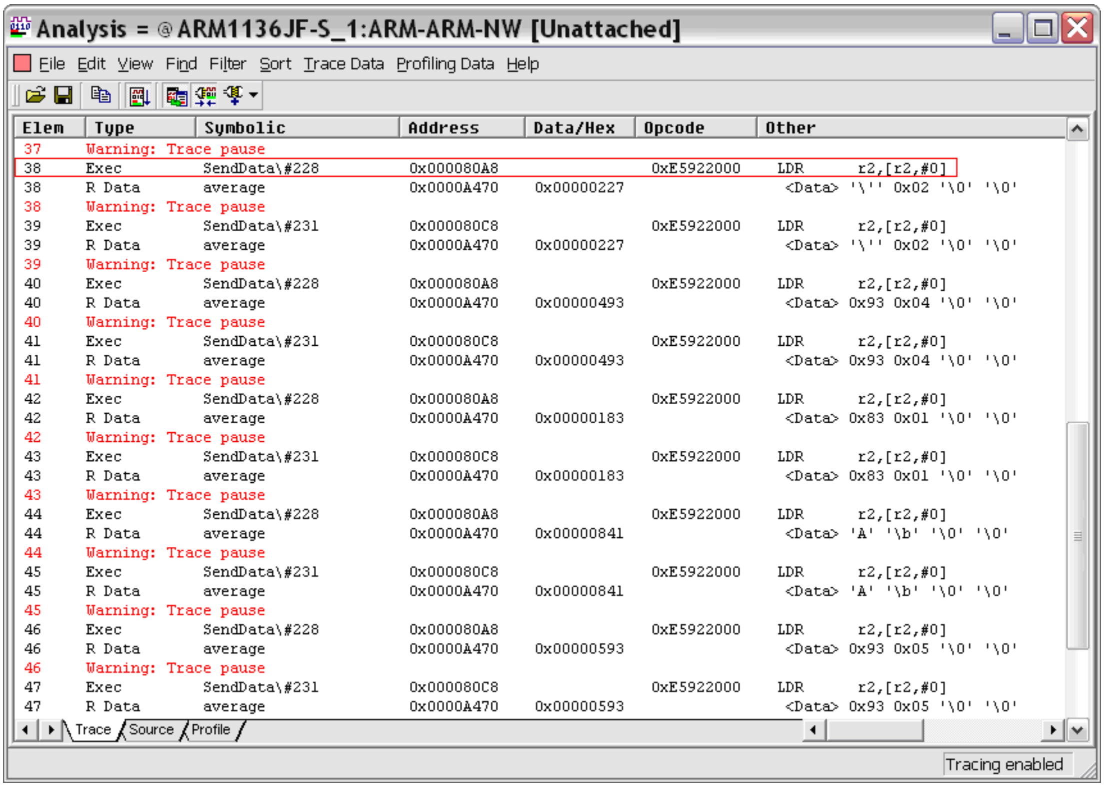


Break/Tracepoints

Type	Value
☑ Trace Da...	0x0000A470
Address	0x0000A470
Command	trcdread,hw_out:"Tracepoint Type=Trace Instr and Data" @trace\\&average

Figure 84

Run the program for a few seconds and stop it, you will see that average is only read from two instructions. It is read both times from SendData: once to compute the number of input samples to output (from address 0x80A8), and a second time so that it may be output (from address 0x80C8). In both cases the value of average will always be the same (0x227 in the highlighted line), but it will vary the next time SendData() executes.



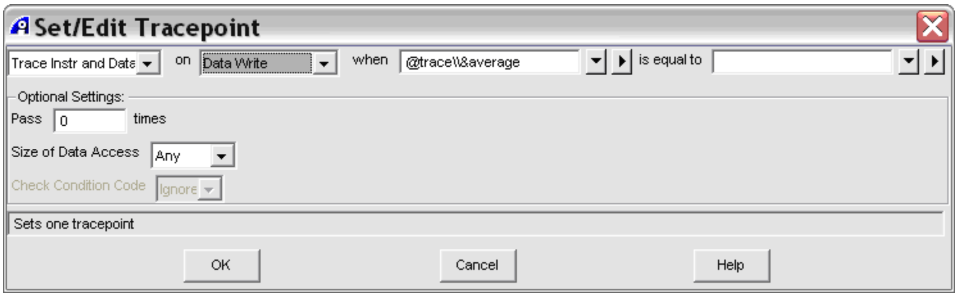
Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]

```
Elem  Type          Symbolic          Address      Data/Hex     Opcode        Other
37    Warning: Trace pause
38    Exec          SendData\#228     0x000080A8                0xE5922000    LDR      r2,[r2,#0]
38    R Data        average           0x0000A470   0x00000227                  <Data> '\'' 0x02 '\0' '\0'
38    Warning: Trace pause
39    Exec          SendData\#231     0x000080C8                0xE5922000    LDR      r2,[r2,#0]
39    R Data        average           0x0000A470   0x00000227                  <Data> '\'' 0x02 '\0' '\0'
39    Warning: Trace pause
40    Exec          SendData\#228     0x000080A8                0xE5922000    LDR      r2,[r2,#0]
40    R Data        average           0x0000A470   0x00000493                  <Data> 0x93 0x04 '\0' '\0'
40    Warning: Trace pause
41    Exec          SendData\#231     0x000080C8                0xE5922000    LDR      r2,[r2,#0]
41    R Data        average           0x0000A470   0x00000493                  <Data> 0x93 0x04 '\0' '\0'
41    Warning: Trace pause
42    Exec          SendData\#228     0x000080A8                0xE5922000    LDR      r2,[r2,#0]
42    R Data        average           0x0000A470   0x00000183                  <Data> 0x83 0x01 '\0' '\0'
42    Warning: Trace pause
43    Exec          SendData\#231     0x000080C8                0xE5922000    LDR      r2,[r2,#0]
43    R Data        average           0x0000A470   0x00000183                  <Data> 0x83 0x01 '\0' '\0'
43    Warning: Trace pause
44    Exec          SendData\#228     0x000080A8                0xE5922000    LDR      r2,[r2,#0]
44    R Data        average           0x0000A470   0x00000841                  <Data> 'A' '\b' '\0' '\0'
44    Warning: Trace pause
45    Exec          SendData\#231     0x000080C8                0xE5922000    LDR      r2,[r2,#0]
45    R Data        average           0x0000A470   0x00000841                  <Data> 'A' '\b' '\0' '\0'
45    Warning: Trace pause
46    Exec          SendData\#228     0x000080A8                0xE5922000    LDR      r2,[r2,#0]
46    R Data        average           0x0000A470   0x00000593                  <Data> 0x93 0x05 '\0' '\0'
46    Warning: Trace pause
47    Exec          SendData\#231     0x000080C8                0xE5922000    LDR      r2,[r2,#0]
47    R Data        average           0x0000A470   0x00000593                  <Data> 0x93 0x05 '\0' '\0'
```

Tracing enabled

Figure 85

Now modify the trace point to trace data writes to &average:



Set/Edit Tracepoint

Trace Instr and Data on Data Write when @trace\\&average is equal to

Optional Settings:

Pass 0 times

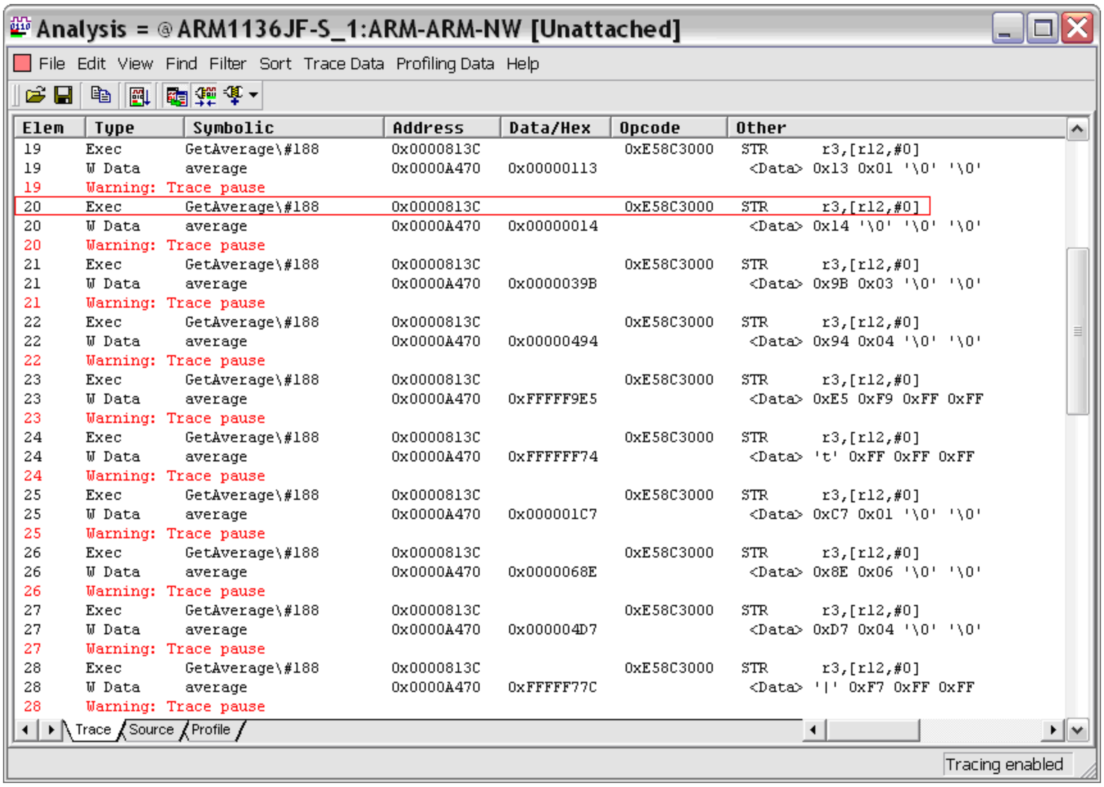
Size of Data Access: Any

Check Condition Code: Ignore

Sets one tracepoint

Figure 86

Run the program for a few seconds (it doesn't have to be re-loaded) and stop it. Observe that &average is only written to at one location in the program (in GetAverage()):

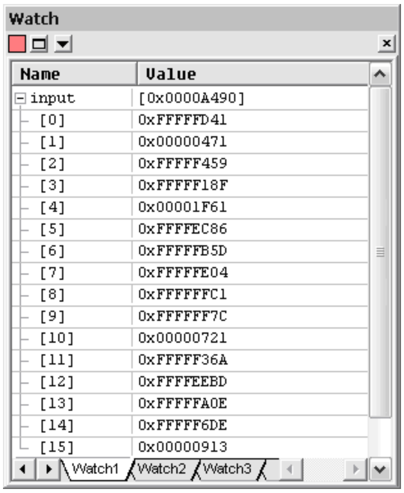

Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

```
Elem  Type        Symbolic          Address     Data/Hex    Opcode      Other
19    Exec        GetAverage\#188   0x0000813C              0xE58C3000  STR       r3,[r12,#0]
19    W Data      average           0x0000A470  0x00000113               <Data> 0x13 0x01 '\0' '\0'
19    Warning: Trace pause
20    Exec        GetAverage\#188   0x0000813C              0xE58C3000  STR       r3,[r12,#0]
20    W Data      average           0x0000A470  0x00000014               <Data> 0x14 '\0' '\0' '\0'
20    Warning: Trace pause
21    Exec        GetAverage\#188   0x0000813C              0xE58C3000  STR       r3,[r12,#0]
21    W Data      average           0x0000A470  0x0000039B               <Data> 0x9B 0x03 '\0' '\0'
21    Warning: Trace pause
22    Exec        GetAverage\#188   0x0000813C              0xE58C3000  STR       r3,[r12,#0]
22    W Data      average           0x0000A470  0x00000494               <Data> 0x94 0x04 '\0' '\0'
22    Warning: Trace pause
23    Exec        GetAverage\#188   0x0000813C              0xE58C3000  STR       r3,[r12,#0]
23    W Data      average           0x0000A470  0xFFFFF9E5               <Data> 0xE5 0xF9 0xFF 0xFF
23    Warning: Trace pause
24    Exec        GetAverage\#188   0x0000813C              0xE58C3000  STR       r3,[r12,#0]
24    W Data      average           0x0000A470  0xFFFFFF74               <Data> 't' 0xFF 0xFF 0xFF
24    Warning: Trace pause
25    Exec        GetAverage\#188   0x0000813C              0xE58C3000  STR       r3,[r12,#0]
25    W Data      average           0x0000A470  0x000001C7               <Data> 0xC7 0x01 '\0' '\0'
25    Warning: Trace pause
26    Exec        GetAverage\#188   0x0000813C              0xE58C3000  STR       r3,[r12,#0]
26    W Data      average           0x0000A470  0x0000068E               <Data> 0x8E 0x06 '\0' '\0'
26    Warning: Trace pause
27    Exec        GetAverage\#188   0x0000813C              0xE58C3000  STR       r3,[r12,#0]
27    W Data      average           0x0000A470  0x000004D7               <Data> 0xD7 0x04 '\0' '\0'
27    Warning: Trace pause
28    Exec        GetAverage\#188   0x0000813C              0xE58C3000  STR       r3,[r12,#0]
28    W Data      average           0x0000A470  0xFFFFF77C               <Data> '|' 0xF7 0xFF 0xFF
28    Warning: Trace pause
```

Figure 87

7.4 Tracing Data Accesses to an Address Range

In Section 7.3 data traces are configured to trace accesses to a specific address - either *output_port or &average. The ETM also allows data trace to be configured for an address range. In this scenario, all read accesses to the input[ ] array will be traced. Add the input[ ] array to the Watch window to determine the location of the array in memory. The 16-word array resides from 0xA490 (index 0) through 0xA4CF (index 15):

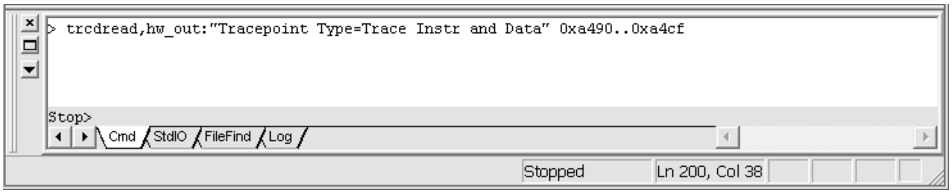


Name	Value
⊟ input	[0x0000A490]
[0]	0xFFFFFD41
[1]	0x00000471
[2]	0xFFFFF459
[3]	0xFFFFF18F
[4]	0x00001F61
[5]	0xFFFFEC86
[6]	0xFFFFFB5D
[7]	0xFFFFFE04
[8]	0xFFFFFFC1
[9]	0xFFFFFF7C
[10]	0x00000721
[11]	0xFFFFF36A
[12]	0xFFFFEEBD
[13]	0xFFFFFA0E
[14]	0xFFFFF6DE
[15]	0x00000913

Figure 88

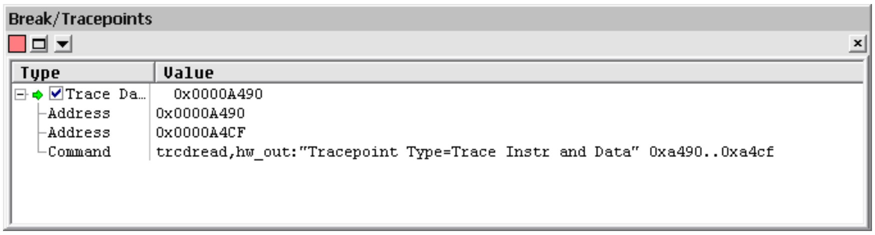
To trace data reads from a memory range, the command line interface must be used. Reload TRACE.AXF, remove any previous trace points and set the following data trace point using the command window:

```
> trcdread,hw_out:"Tracepoint Type=Trace Instr and Data" 0xa490..0xa4cf
```



```
> trcdread,hw_out:"Tracepoint Type=Trace Instr and Data" 0xa490..0xa4cf

Stop>
```

Figure 89



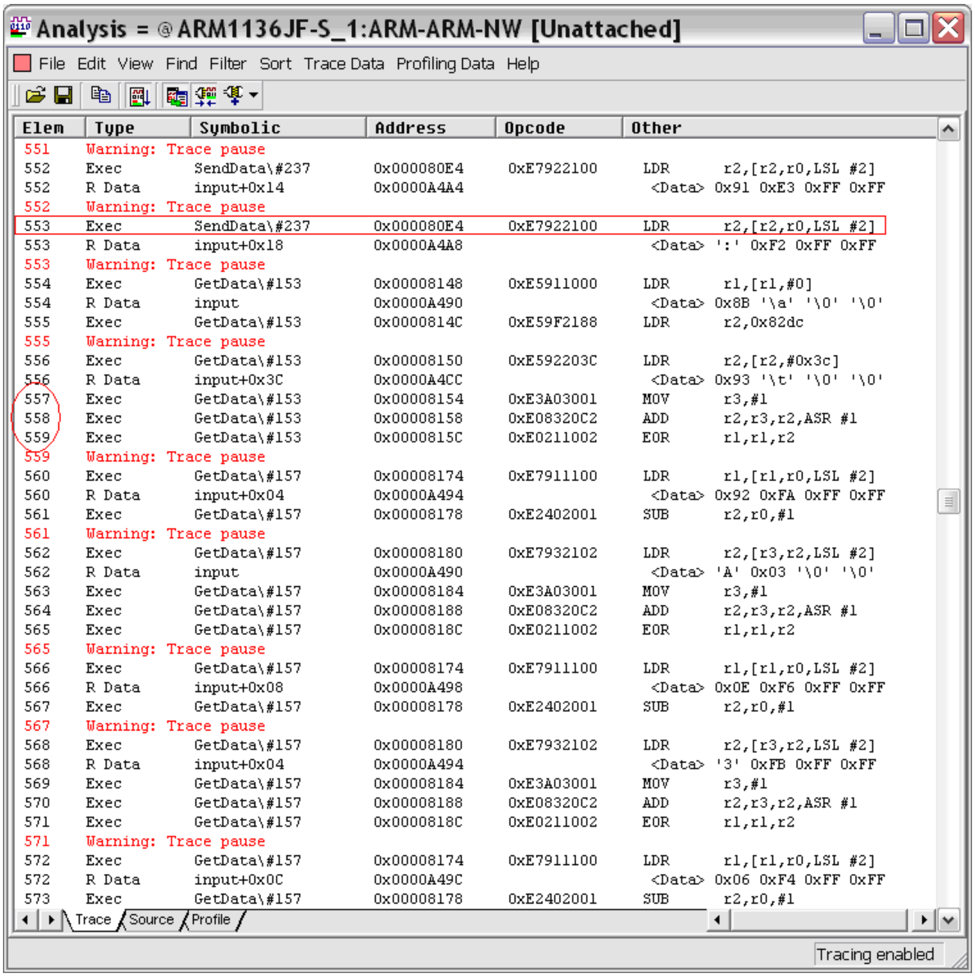
Break/Tracepoints

Type	Value
☑ Trace Da…	0x0000A490
Address	0x0000A490
Address	0x0000A4CF
Command	trcdread,hw_out:"Tracepoint Type=Trace Instr and Data" 0xa490..0xa4cf

Figure 90

Now run the program for a few seconds and stop it. Scroll through the Analysis window and observe the following:

- Read accesses to input[ ] are made from GetData(), GetAverage() and SendData(). This is most readily observed by clicking on the "Source" tab in the Analysis window.
- All 16 words of input[ ] are read and traced
- Some instructions that don't access the data trace range are traced



Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

```
Elem  Type      Symbolic         Address     Opcode      Other
551   Warning: Trace pause
552   Exec      SendData\#237    0x000080E4  0xE7922100  LDR      r2,[r2,r0,LSL #2]
552   R Data    input+0x14       0x0000A4A4               <Data> 0x91 0xE3 0xFF 0xFF
552   Warning: Trace pause
553   Exec      SendData\#237    0x000080E4  0xE7922100  LDR      r2,[r2,r0,LSL #2]
553   R Data    input+0x18       0x0000A4A8               <Data> ':' 0xF2 0xFF 0xFF
553   Warning: Trace pause
554   Exec      GetData\#153     0x00008148  0xE5911000  LDR      r1,[r1,#0]
554   R Data    input            0x0000A490               <Data> 0x8B '\a' '\0' '\0'
555   Exec      GetData\#153     0x0000814C  0xE59F2188  LDR      r2,0x82dc
555   Warning: Trace pause
556   Exec      GetData\#153     0x00008150  0xE592203C  LDR      r2,[r2,#0x3c]
556   R Data    input+0x3C       0x0000A4CC               <Data> 0x93 '\t' '\0' '\0'
557   Exec      GetData\#153     0x00008154  0xE3A03001  MOV      r3,#1
558   Exec      GetData\#153     0x00008158  0xE08320C2  ADD      r2,r3,r2,ASR #1
559   Exec      GetData\#153     0x0000815C  0xE0211002  EOR      r1,r1,r2
559   Warning: Trace pause
560   Exec      GetData\#157     0x00008174  0xE7911100  LDR      r1,[r1,r0,LSL #2]
560   R Data    input+0x04       0x0000A494               <Data> 0x92 0xFA 0xFF 0xFF
561   Exec      GetData\#157     0x00008178  0xE2402001  SUB      r2,r0,#1
561   Warning: Trace pause
562   Exec      GetData\#157     0x00008180  0xE7932102  LDR      r2,[r3,r2,LSL #2]
562   R Data    input            0x0000A490               <Data> 'A' 0x03 '\0' '\0'
563   Exec      GetData\#157     0x00008184  0xE3A03001  MOV      r3,#1
564   Exec      GetData\#157     0x00008188  0xE08320C2  ADD      r2,r3,r2,ASR #1
565   Exec      GetData\#157     0x0000818C  0xE0211002  EOR      r1,r1,r2
565   Warning: Trace pause
566   Exec      GetData\#157     0x00008174  0xE7911100  LDR      r1,[r1,r0,LSL #2]
566   R Data    input+0x08       0x0000A498               <Data> 0x0E 0xF6 0xFF 0xFF
567   Exec      GetData\#157     0x00008178  0xE2402001  SUB      r2,r0,#1
567   Warning: Trace pause
568   Exec      GetData\#157     0x00008180  0xE7932102  LDR      r2,[r3,r2,LSL #2]
568   R Data    input+0x04       0x0000A494               <Data> '3' 0xFB 0xFF 0xFF
569   Exec      GetData\#157     0x00008184  0xE3A03001  MOV      r3,#1
570   Exec      GetData\#157     0x00008188  0xE08320C2  ADD      r2,r3,r2,ASR #1
571   Exec      GetData\#157     0x0000818C  0xE0211002  EOR      r1,r1,r2
571   Warning: Trace pause
572   Exec      GetData\#157     0x00008174  0xE7911100  LDR      r1,[r1,r0,LSL #2]
572   R Data    input+0x0C       0x0000A49C               <Data> 0x06 0xF4 0xFF 0xFF
573   Exec      GetData\#157     0x00008178  0xE2402001  SUB      r2,r0,#1
```

Figure 91

The most interesting item in the capture is that the even some instructions that don't make a data access are traced (such as circled elements 557 - 559). Even though this appears to be a problem with the trace, this behavior is defined in the ETM Architecture Specification. Namely, when a data access is made within the specified data range, trace is activated and stays active until a data access is made outside the range.

The instruction at element 556 performs a read of input[15] and this activates trace. The three circled instructions only perform internal register operations so they are also traced. If you reference the Dsm tab of the main RVD window you will see that the next executed instruction is "LDR r2, 0x82DC" (located at 0x8160). This instruction makes a data access outside the trace range so it is not traced and trace is deactivated (paused).

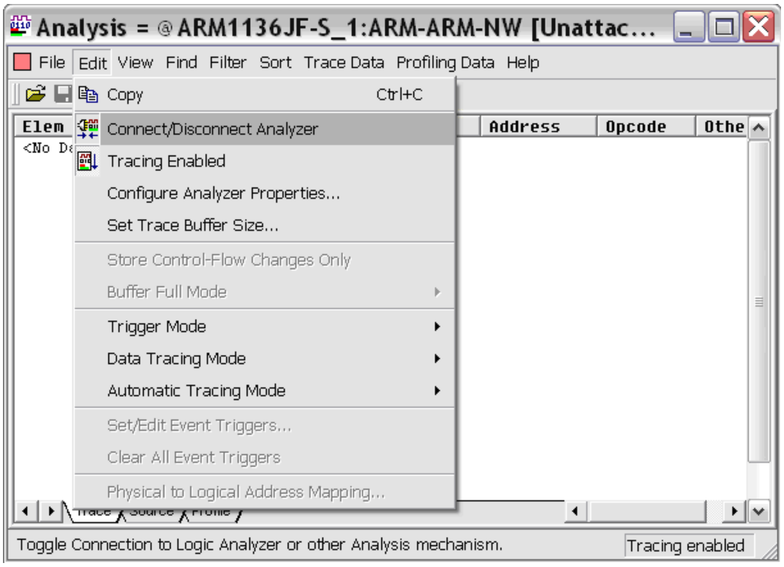
8 Tracing a Running Target

The trace examples presented in Sections 4 - 7 generate trace displays by halting the running target. When a target is halted with trace enabled, the captured trace buffer stored in RealView Trace or the target's ETB is automatically dumped to RVD for display. There may be times when your target can not be stopped to generate a trace capture because it is controlling hardware which must continue to operate. In these situations you will have to make a trace capture from a running target.

There are two methods to perform a capture with a running target. You can either disable trace or use a trace trigger. Both of these scenarios are described in this section.

8.1 Connecting the Analyzer and Disabling Trace

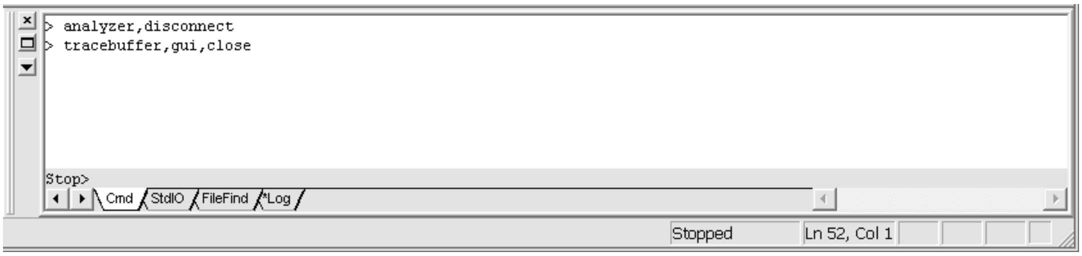
Remove all trace points and breakpoints that you may have set. Ensure that the trace analyzer is not connected. If the trace analyzer is connected, disconnect it using the "Edit - Connect/Disconnect Analyzer" menu from the Analysis window:


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattac...
File Edit View Find Filter Sort Trace Data Profiling Data Help
Copy Ctrl+C
Connect/Disconnect Analyzer
Tracing Enabled
Configure Analyzer Properties...
Set Trace Buffer Size...
Store Control-Flow Changes Only
Buffer Full Mode
Trigger Mode
Data Tracing Mode
Automatic Tracing Mode
Set/Edit Event Triggers...
Clear All Event Triggers
Physical to Logical Address Mapping...
Elem
<No Da
Address
Opcode
Othe
Trace
Source
Profile
Tracing enabled

Toggle Connection to Logic Analyzer or other Analysis mechanism.

Figure 92

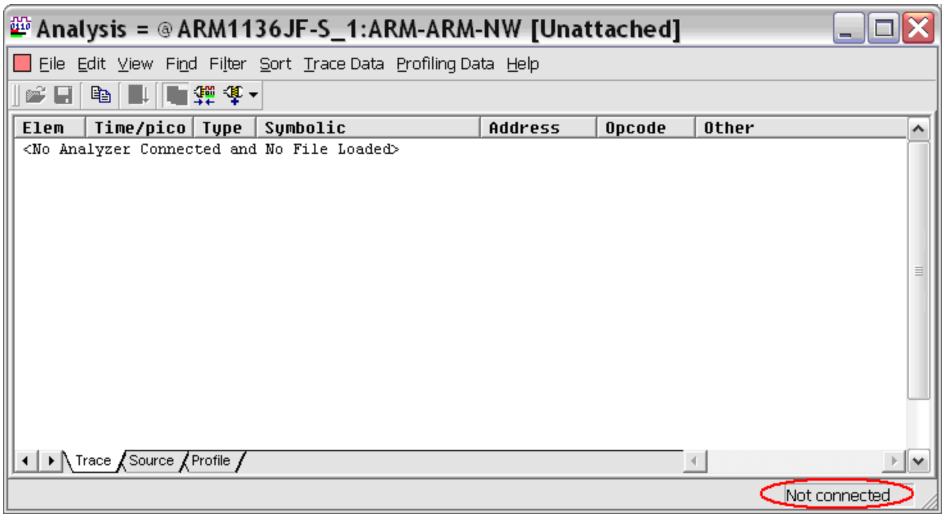
When you disconnect the trace analyzer from the Analysis window, the following message is displayed in the RVD command window:



```
> analyzer,disconnect
> tracebuffer,gui,close
```

Figure 93

Additionally, the status bar of the Analysis window indicates that the trace analyzer is not connected. When the trace analyzer is not connected, trace is disabled on the target:

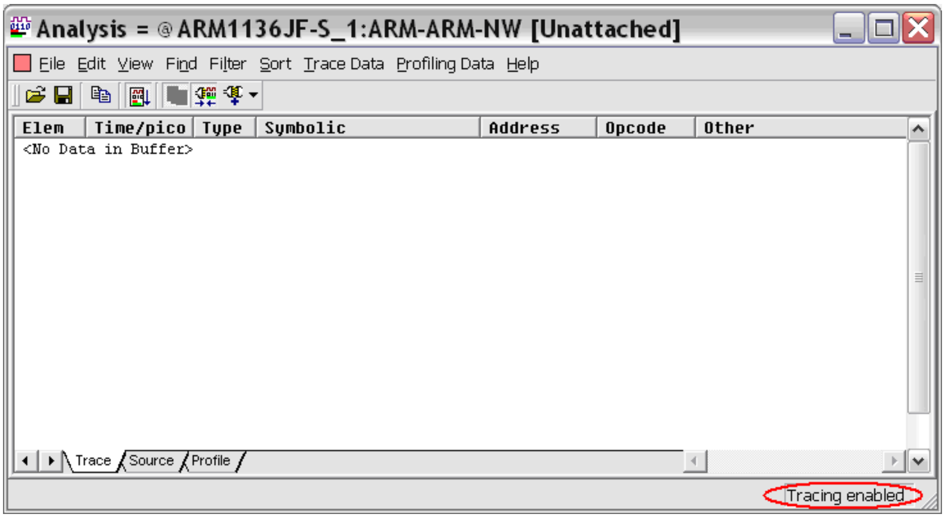

Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Elem
Time/pico
Type
Symbolic
Address
Opcode
Other
Trace
Source
Profile

<No Analyzer Connected and No File Loaded>

Not connected

Figure 94

Load TRACE.AXF and run the program. Connect the trace analyzer from the RVD "Tools - Analyzer/Trace Control - Connect Analyzer/Analysis" menu. When the analyzer is first connected, trace is enabled with Auto Tracing (see Section 4). In the status bar, observe that trace is now enabled:


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Elem
Time/pico
Type
Symbolic
Address
Opcode
Other
Trace
Source
Profile

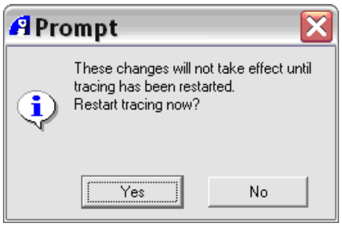
<No Data in Buffer>

Tracing enabled

Figure 95

Even though tracing is enabled, no trace data is displayed in RVD until the target is halted, tracing is disabled or a trigger point is reached. When the trace buffer is full, trace continues to be captured by overwriting the oldest trace sample in the buffer.

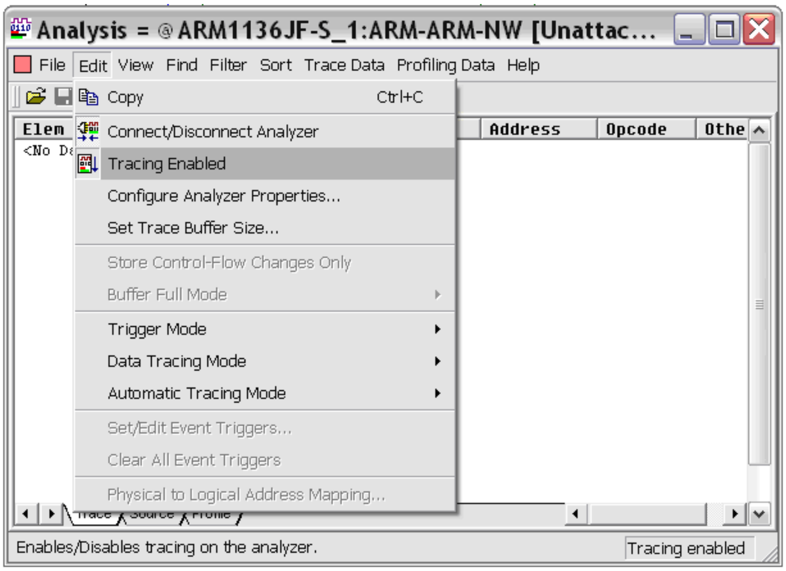
If you need to change the ETM configuration, you can do this even while the target is running. Use the "Tools - Analyzer/Trace Control - Configure Analyzer Properties" menu as described in Section 3. For configuration changes to take effect, tracing is stopped and restarted while your target continues to run. If you do make a change in the configuration, RVD prompts you to allow the change to occur. If you changed the ETM configuration, click the "Yes" button in the RVD prompt:



These changes will not take effect until tracing has been restarted.
Restart tracing now?

Figure 96

With the program running and trace enabled, disable trace capture from the "Edit - Tracing Enabled" menu in the Analysis window. This selection toggles tracing on and off:

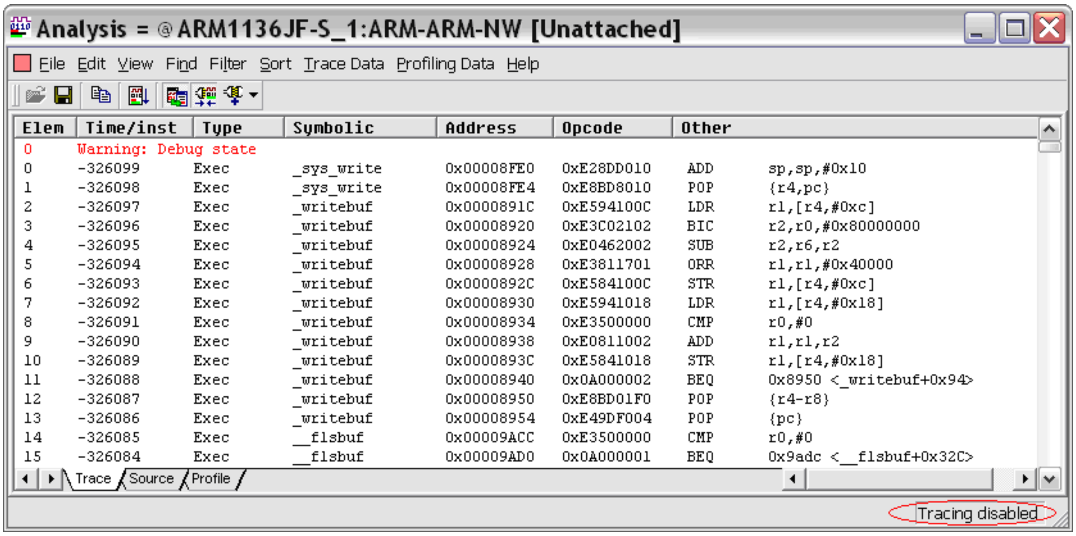


Enables/Disables tracing on the analyzer.

Tracing enabled

Figure 97

Observe that when the selection is made, tracing is disabled, the trace buffer is dumped to RVD and the target continues to run:

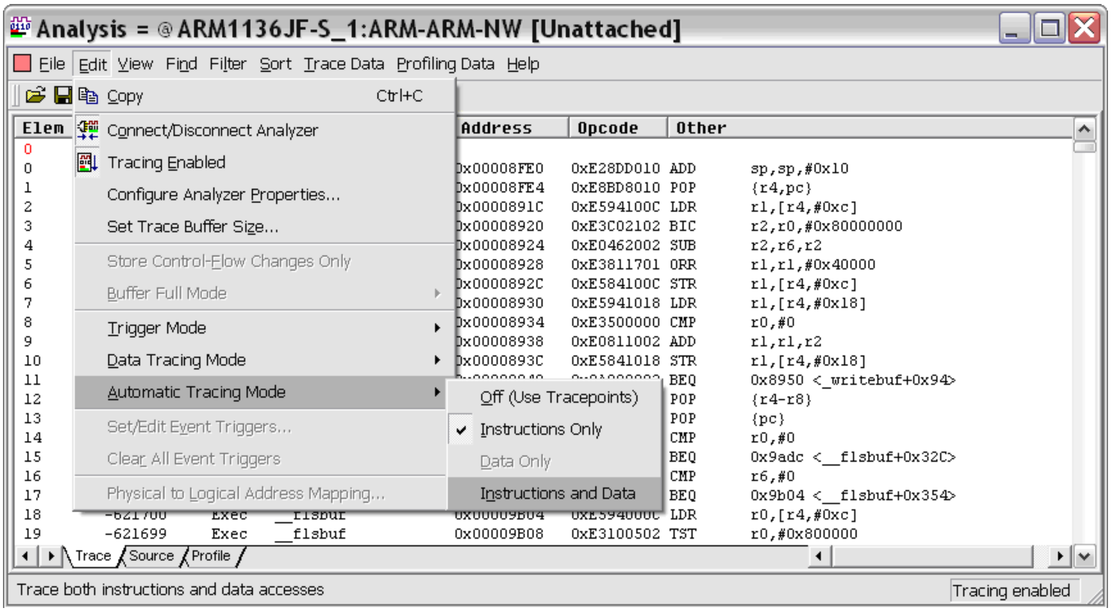


```
Elem  Time/inst  Type    Symbolic       Address      Opcode       Other
0     Warning: Debug state
0     -326099    Exec    _sys_write     0x00008FE0   0xE28DD010   ADD      sp,sp,#0x10
1     -326098    Exec    _sys_write     0x00008FE4   0xE8BD8010   POP      {r4,pc}
2     -326097    Exec    _writebuf      0x0000891C   0xE594100C   LDR      r1,[r4,#0xc]
3     -326096    Exec    _writebuf      0x00008920   0xE3C02102   BIC      r2,r0,#0x80000000
4     -326095    Exec    _writebuf      0x00008924   0xE0462002   SUB      r2,r6,r2
5     -326094    Exec    _writebuf      0x00008928   0xE3811701   ORR      r1,r1,#0x40000
6     -326093    Exec    _writebuf      0x0000892C   0xE584100C   STR      r1,[r4,#0xc]
7     -326092    Exec    _writebuf      0x00008930   0xE5941018   LDR      r1,[r4,#0x18]
8     -326091    Exec    _writebuf      0x00008934   0xE3500000   CMP      r0,#0
9     -326090    Exec    _writebuf      0x00008938   0xE0811002   ADD      r1,r1,r2
10    -326089    Exec    _writebuf      0x0000893C   0xE5841018   STR      r1,[r4,#0x18]
11    -326088    Exec    _writebuf      0x00008940   0x0A000002   BEQ      0x8950 <_writebuf+0x94>
12    -326087    Exec    _writebuf      0x00008950   0xE8BD01F0   POP      {r4-r8}
13    -326086    Exec    _writebuf      0x00008954   0xE49DF004   POP      {pc}
14    -326085    Exec    __flsbuf       0x00009ACC   0xE3500000   CMP      r0,#0
15    -326084    Exec    __flsbuf       0x00009AD0   0x0A000001   BEQ      0x9adc <__flsbuf+0x32C>
```

Tracing disabled

Figure 98

Toggle tracing on and off from the "Edit - Tracing Enabled" menu a few times and observe how new trace captures are made each time tracing is disabled. With tracing enabled and your program running, set the Automatic Tracing Mode to "Instructions and Data" from the Edit menu:


Analysis = @ ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Copy Ctrl+C
Connect/Disconnect Analyzer
Tracing Enabled
Configure Analyzer Properties...
Set Trace Buffer Size...
Store Control-Flow Changes Only
Buffer Full Mode
Trigger Mode
Data Tracing Mode
Automatic Tracing Mode
Set/Edit Event Triggers...
Clear All Event Triggers
Physical to Logical Address Mapping...
Off (Use Tracepoints)
✓ Instructions Only
Data Only
Instructions and Data
Trace
Source
Profile

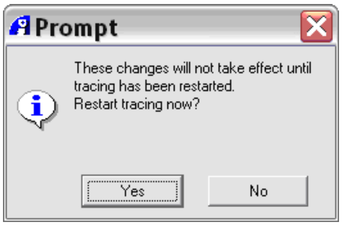
Elem				Address	Opcode	Other
0						
0				x00008FE0	0xE28DD010	ADD sp,sp,#0x10
1				x00008FE4	0xE8BD8010	POP {r4,pc}
2				x0000891C	0xE594100C	LDR r1,[r4,#0xc]
3				x00008920	0xE3C02102	BIC r2,r0,#0x80000000
4				x00008924	0xE0462002	SUB r2,r6,r2
5				x00008928	0xE3811701	ORR r1,r1,#0x40000
6				x0000892C	0xE584100C	STR r1,[r4,#0xc]
7				x00008930	0xE5941018	LDR r1,[r4,#0x18]
8				x00008934	0xE3500000	CMP r0,#0
9				x00008938	0xE0811002	ADD r1,r1,r2
10				x0000893C	0xE5841018	STR r1,[r4,#0x18]
11						BEQ 0x8950 <_writebuf+0x94>
12						POP {r4-r8}
13						POP {pc}
14						CMP r0,#0
15						BEQ 0x9adc <__flsbuf+0x32C>
16						CMP r6,#0
17						BEQ 0x9b04 <__flsbuf+0x354>
18	-621700	Exec	__flsbuf	0x00009B04	0xE594000C	LDR r0,[r4,#0xc]
19	-621699	Exec	__flsbuf	0x00009B08	0xE3100502	TST r0,#0x800000

Trace both instructions and data accesses

Tracing enabled

Figure 99

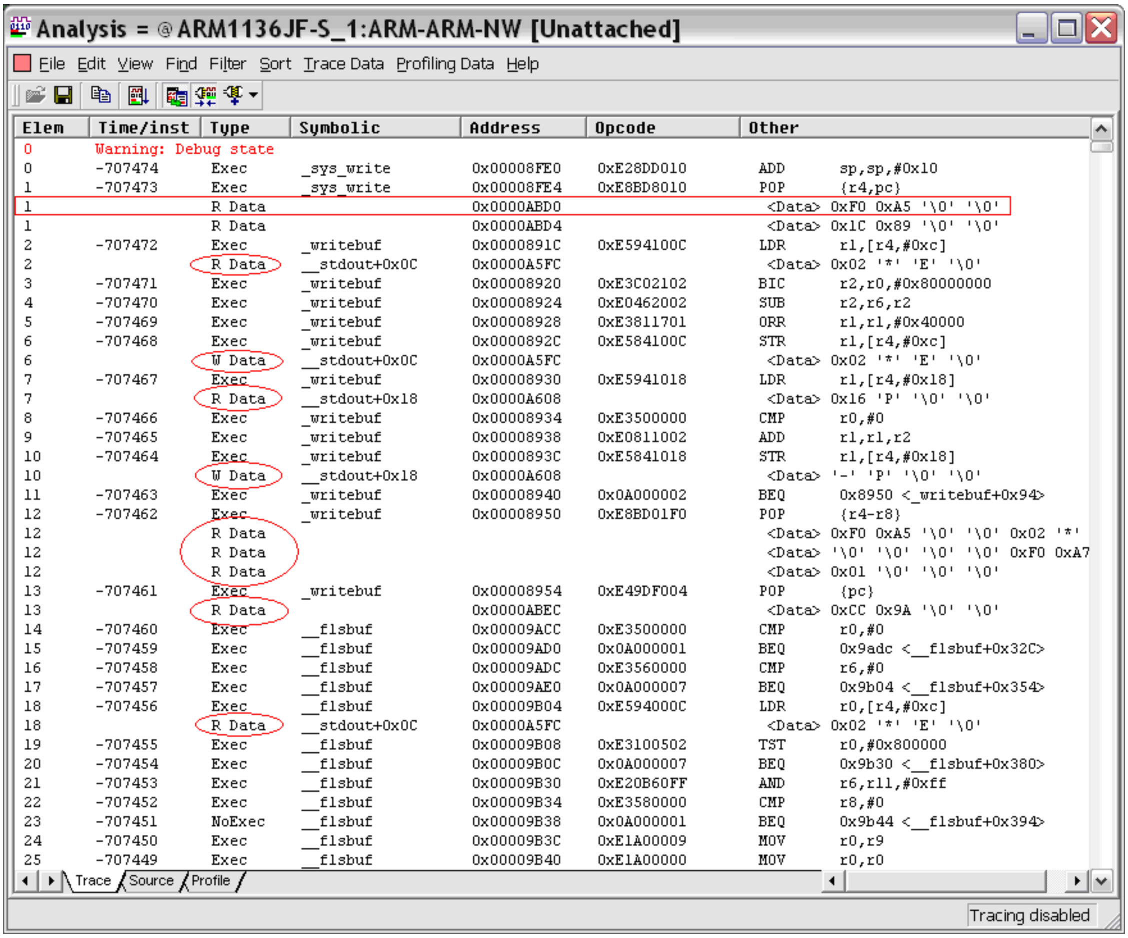
And click "Yes" for the prompt to restart tracing with the new trace settings:



These changes will not take effect until tracing has been restarted.
Restart tracing now?

Figure 100

Now disable trace from the "Edit - Tracing Enabled" menu in the Analysis window to perform the trace capture. Observe how both instruction execution and data accesses are now traced:


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile

```
Elem  Time/inst  Type     Symbolic          Address      Opcode        Other
0     Warning: Debug state
0     -707474    Exec     _sys_write        0x00008FE0   0xE28DD010    ADD    sp,sp,#0x10
1     -707473    Exec     _sys_write        0x00008FE4   0xE8BD8010    POP    {r4,pc}
1                R Data                     0x0000ABD0                  <Data> 0xF0 0xA5 '\0' '\0'
1                R Data                     0x0000ABD4                  <Data> 0x1C 0x89 '\0' '\0'
2     -707472    Exec     _writebuf         0x0000891C   0xE594100C    LDR    r1,[r4,#0xc]
2                R Data   __stdout+0x0C     0x0000A5FC                  <Data> 0x02 '*' 'E' '\0'
3     -707471    Exec     _writebuf         0x00008920   0xE3C02102    BIC    r2,r0,#0x80000000
4     -707470    Exec     _writebuf         0x00008924   0xE0462002    SUB    r2,r6,r2
5     -707469    Exec     _writebuf         0x00008928   0xE3811701    ORR    r1,r1,#0x40000
6     -707468    Exec     _writebuf         0x0000892C   0xE584100C    STR    r1,[r4,#0xc]
6                W Data   __stdout+0x0C     0x0000A5FC                  <Data> 0x02 '*' 'E' '\0'
7     -707467    Exec     _writebuf         0x00008930   0xE5941018    LDR    r1,[r4,#0x18]
7                R Data   __stdout+0x18     0x0000A608                  <Data> 0x16 'P' '\0' '\0'
8     -707466    Exec     _writebuf         0x00008934   0xE3500000    CMP    r0,#0
9     -707465    Exec     _writebuf         0x00008938   0xE0811002    ADD    r1,r1,r2
10    -707464    Exec     _writebuf         0x0000893C   0xE5841018    STR    r1,[r4,#0x18]
10               W Data   __stdout+0x18     0x0000A608                  <Data> '-' 'P' '\0' '\0'
11    -707463    Exec     _writebuf         0x00008940   0x0A000002    BEQ    0x8950 <_writebuf+0x94>
12    -707462    Exec     _writebuf         0x00008950   0xE8BD01F0    POP    {r4-r8}
12               R Data                                                 <Data> 0xF0 0xA5 '\0' '\0' 0x02 '*'
12               R Data                                                 <Data> '\0' '\0' '\0' '\0' 0xF0 0xA7
12               R Data                                                 <Data> 0x01 '\0' '\0' '\0'
13    -707461    Exec     _writebuf         0x00008954   0xE49DF004    POP    {pc}
13               R Data                     0x0000ABEC                  <Data> 0xCC 0x9A '\0' '\0'
14    -707460    Exec     __flsbuf          0x00009ACC   0xE3500000    CMP    r0,#0
15    -707459    Exec     __flsbuf          0x00009AD0   0x0A000001    BEQ    0x9adc <__flsbuf+0x32C>
16    -707458    Exec     __flsbuf          0x00009ADC   0xE3560000    CMP    r6,#0
17    -707457    Exec     __flsbuf          0x00009AE0   0x0A000007    BEQ    0x9b04 <__flsbuf+0x354>
18    -707456    Exec     __flsbuf          0x00009B04   0xE594000C    LDR    r0,[r4,#0xc]
18               R Data   __stdout+0x0C     0x0000A5FC                  <Data> 0x02 '*' 'E' '\0'
19    -707455    Exec     __flsbuf          0x00009B08   0xE3100502    TST    r0,#0x800000
20    -707454    Exec     __flsbuf          0x00009B0C   0x0A000007    BEQ    0x9b30 <__flsbuf+0x380>
21    -707453    Exec     __flsbuf          0x00009B30   0xE20B60FF    AND    r6,r11,#0xff
22    -707452    Exec     __flsbuf          0x00009B34   0xE3580000    CMP    r8,#0
23    -707451    NoExec   __flsbuf          0x00009B38   0x0A000001    BEQ    0x9b44 <__flsbuf+0x394>
24    -707450    Exec     __flsbuf          0x00009B3C   0xE1A00009    MOV    r0,r9
25    -707449    Exec     __flsbuf          0x00009B40   0xE1A00000    MOV    r0,r0
```

Tracing disabled

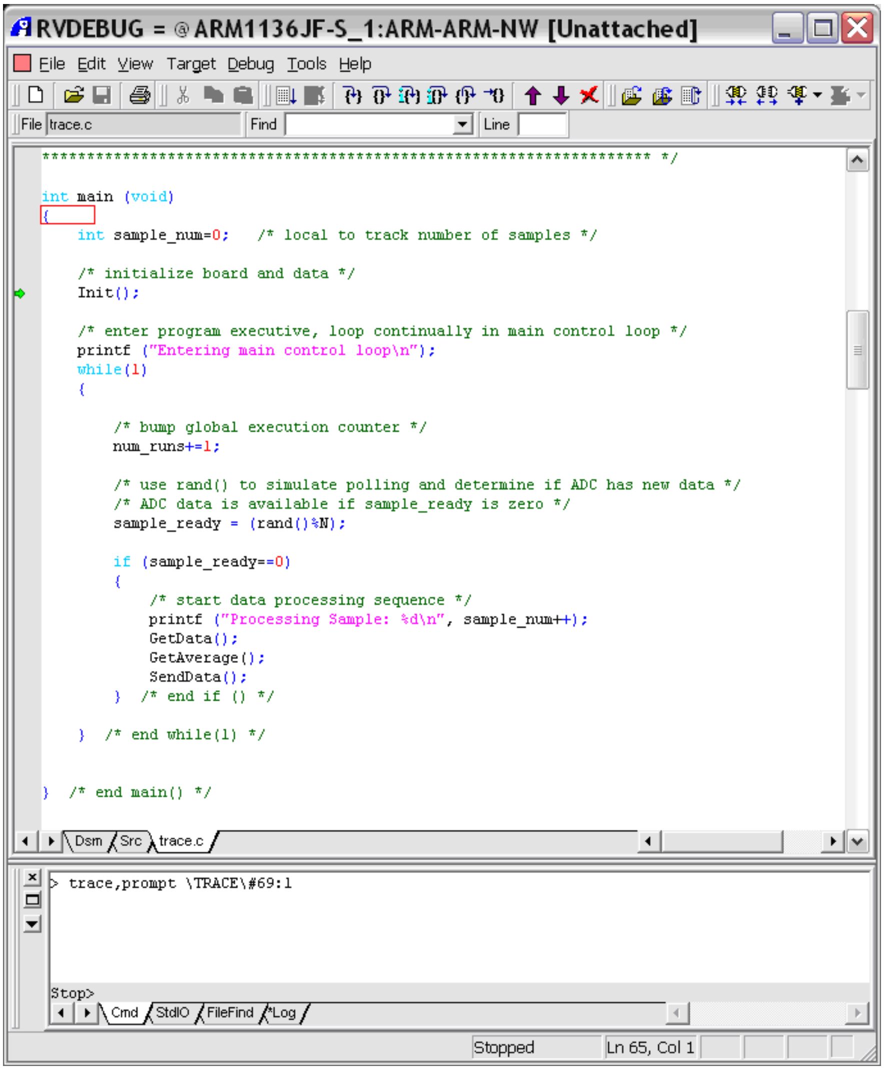
Figure 101

8.2 Simple Trace Triggers

Triggers are used to initiate a trace capture based upon a user-defined trigger condition. When the trigger condition is met, the target will continue to run and the trace buffer will fill with new data. When the trace buffer is full, the contents of the buffer will be dumped to RVD for analysis. After the trace capture is made, trace is no longer enabled even though your target continues running.

In this scenario a trace capture similar to that from Section 4 will be made with the use of a Trace Trigger instead of Auto Tracing with breakpoints. A trigger will be used to make a trace capture which includes the execution of the C library startup code and entry into the while(1) loop of main().

If you disabled trace in Section 8.1, enable trace as described in Section 3. Set the Automatic Tracing Mode back to "Instructions Only" from the Edit menu in the Analysis Window. Reload TRACE.AXF and remove all trace points. Place a Trigger point at the call to Init() in main():

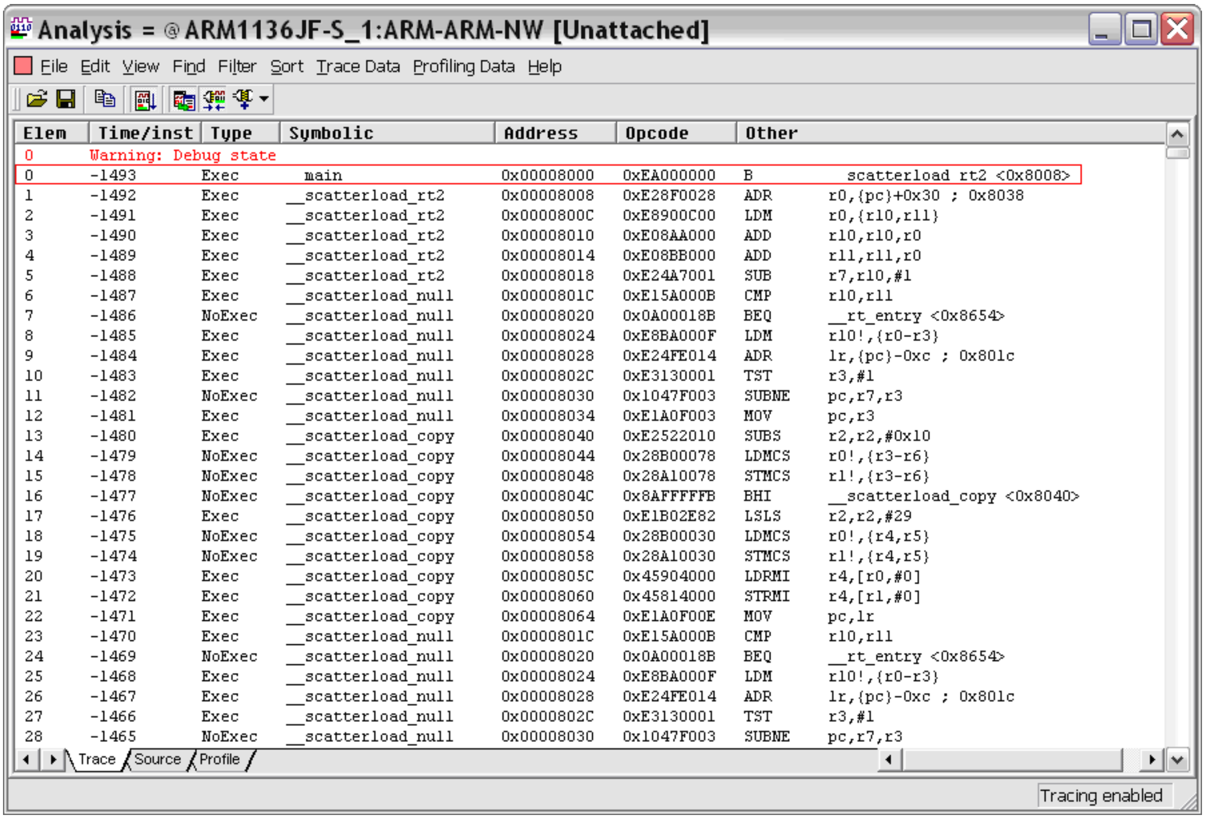

RVDEBUG = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Target Debug Tools Help
File trace.c
Find
Line
Dsm
Src
trace.c
Cmd
StdIO
FileFind
*Log
Stopped
Ln 65, Col 1

```
******************************************************************** */

int main (void)
{
    int sample_num=0;   /* local to track number of samples */

    /* initialize board and data */
    Init();

    /* enter program executive, loop continually in main control loop */
    printf ("Entering main control loop\n");
    while(1)
    {

        /* bump global execution counter */
        num_runs+=1;

        /* use rand() to simulate polling and determine if ADC has new data */
        /* ADC data is available if sample_ready is zero */
        sample_ready = (rand()%N);

        if (sample_ready==0)
        {
            /* start data processing sequence */
            printf ("Processing Sample: %d\n", sample_num++);
            GetData();
            GetAverage();
            SendData();
        }  /* end if () */

    }  /* end while(1) */


}  /* end main() */
```

```
> trace,prompt \TRACE\#69:1

Stop>
```

Figure 102

Open the Trace Analysis window from the "View - Analysis" menu in RVD. Run the program. If you are using RVT, observe that almost immediately the "TRIG" LED lights on the bottom right of the unit. This indicates the trace trigger has been reached. Wait several seconds for the Analysis window to populate with trace data as your program continues to run. Examine the Analysis window and observe that tracing starts from the program entry point (0x8000):

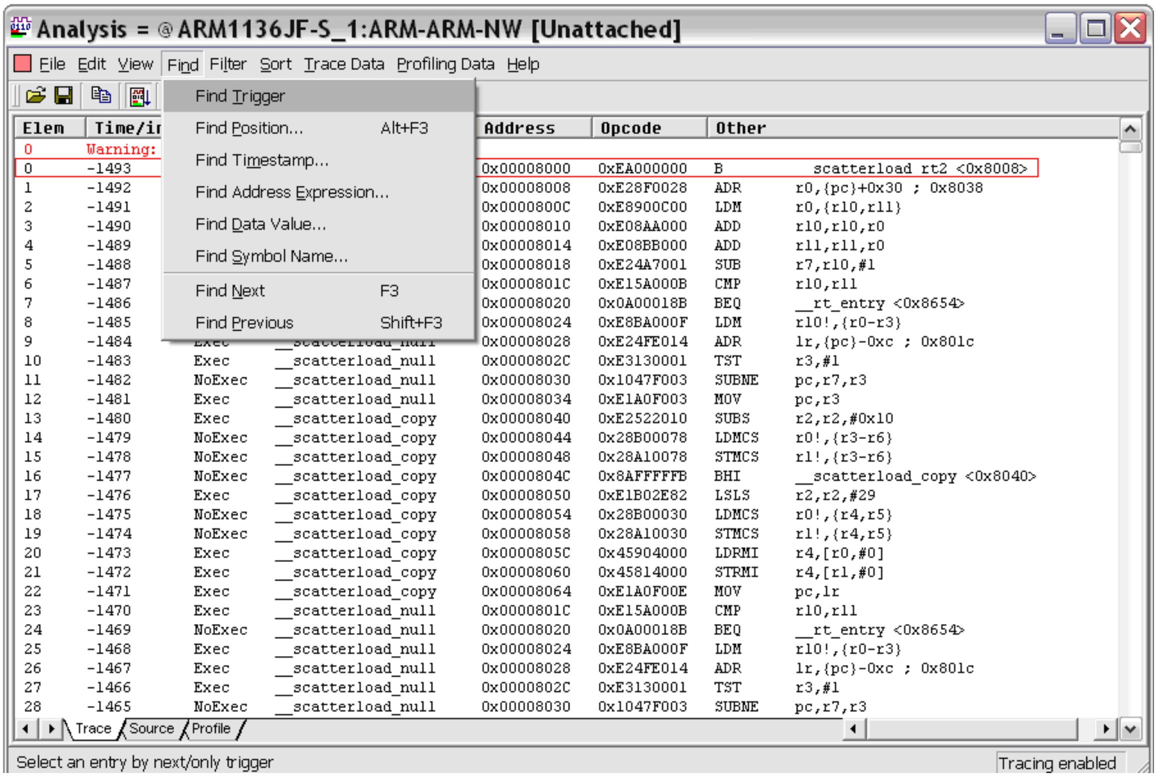


Analysis = @ ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Time/inst	Type	Symbolic	Address	Opcode	Other
0	Warning: Debug state					
0	-1493	Exec	main	0x00008000	0xEA000000	B scatterload rt2 <0x8008>
1	-1492	Exec	__scatterload_rt2	0x00008008	0xE28F0028	ADR r0,{pc}+0x30 ; 0x8038
2	-1491	Exec	__scatterload_rt2	0x0000800C	0xE8900C00	LDM r0,{r10,r11}
3	-1490	Exec	__scatterload_rt2	0x00008010	0xE08AA000	ADD r10,r10,r0
4	-1489	Exec	__scatterload_rt2	0x00008014	0xE08BB000	ADD r11,r11,r0
5	-1488	Exec	__scatterload_rt2	0x00008018	0xE24A7001	SUB r7,r10,#1
6	-1487	Exec	__scatterload_null	0x0000801C	0xE15A000B	CMP r10,r11
7	-1486	NoExec	__scatterload_null	0x00008020	0x0A00018B	BEQ __rt_entry <0x8654>
8	-1485	Exec	__scatterload_null	0x00008024	0xE8BA000F	LDM r10!,{r0-r3}
9	-1484	Exec	__scatterload_null	0x00008028	0xE24FE014	ADR lr,{pc}-0xc ; 0x801c
10	-1483	Exec	__scatterload_null	0x0000802C	0xE3130001	TST r3,#1
11	-1482	NoExec	__scatterload_null	0x00008030	0x1047F003	SUBNE pc,r7,r3
12	-1481	Exec	__scatterload_null	0x00008034	0xE1A0F003	MOV pc,r3
13	-1480	Exec	__scatterload_copy	0x00008040	0xE2522010	SUBS r2,r2,#0x10
14	-1479	NoExec	__scatterload_copy	0x00008044	0x28B00078	LDMCS r0!,{r3-r6}
15	-1478	NoExec	__scatterload_copy	0x00008048	0x28A10078	STMCS r1!,{r3-r6}
16	-1477	NoExec	__scatterload_copy	0x0000804C	0x8AFFFFFB	BHI __scatterload_copy <0x8040>
17	-1476	Exec	__scatterload_copy	0x00008050	0xE1B02E82	LSLS r2,r2,#29
18	-1475	NoExec	__scatterload_copy	0x00008054	0x28B00030	LDMCS r0!,{r4,r5}
19	-1474	NoExec	__scatterload_copy	0x00008058	0x28A10030	STMCS r1!,{r4,r5}
20	-1473	Exec	__scatterload_copy	0x0000805C	0x45904000	LDRMI r4,[r0,#0]
21	-1472	Exec	__scatterload_copy	0x00008060	0x45814000	STRMI r4,[r1,#0]
22	-1471	Exec	__scatterload_copy	0x00008064	0xE1A0F00E	MOV pc,lr
23	-1470	Exec	__scatterload_null	0x0000801C	0xE15A000B	CMP r10,r11
24	-1469	NoExec	__scatterload_null	0x00008020	0x0A00018B	BEQ __rt_entry <0x8654>
25	-1468	Exec	__scatterload_null	0x00008024	0xE8BA000F	LDM r10!,{r0-r3}
26	-1467	Exec	__scatterload_null	0x00008028	0xE24FE014	ADR lr,{pc}-0xc ; 0x801c
27	-1466	Exec	__scatterload_null	0x0000802C	0xE3130001	TST r3,#1
28	-1465	NoExec	__scatterload_null	0x00008030	0x1047F003	SUBNE pc,r7,r3

Figure 103

Search for the Trigger in the trace buffer using the "Find - Find Trigger" menu:



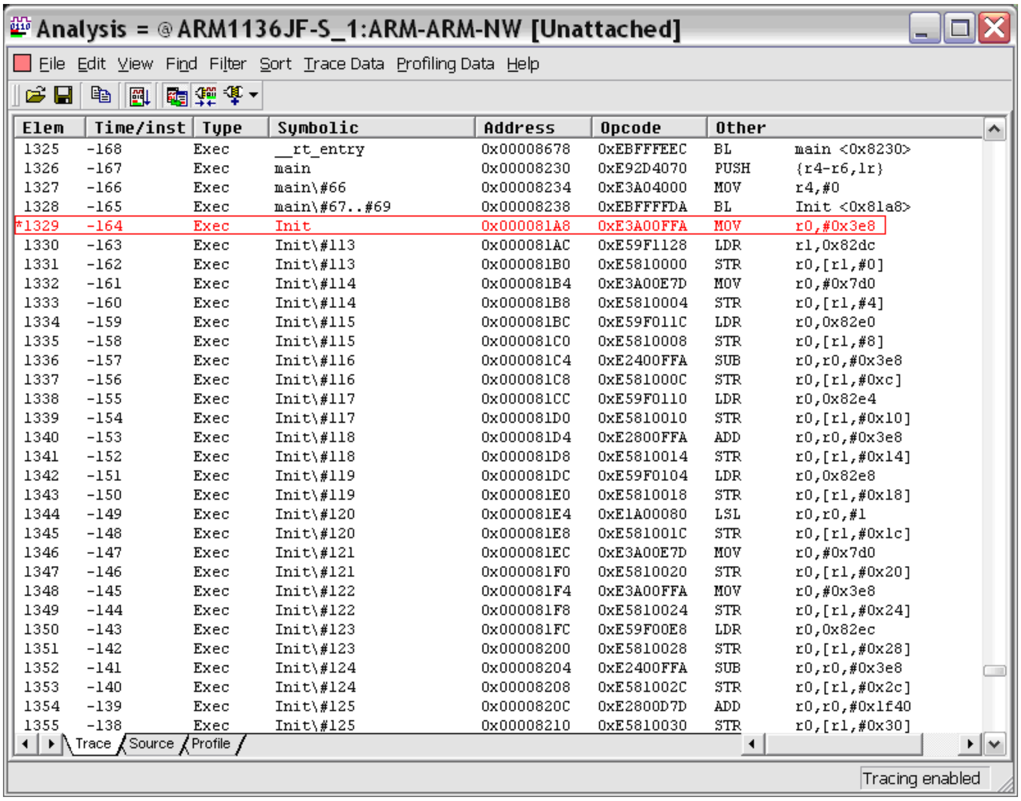
Elem	Time/ir			Address	Opcode	Other
0	Warning:					
0	-1493			0x00008000	0xEA000000	B __scatterload_rt2 <0x8008>
1	-1492			0x00008008	0xE28F0028	ADR r0,{pc}+0x30 ; 0x8038
2	-1491			0x0000800C	0xE8900C00	LDM r0,{r10,r11}
3	-1490			0x00008010	0xE08AA000	ADD r10,r10,r0
4	-1489			0x00008014	0xE08BB000	ADD r11,r11,r0
5	-1488			0x00008018	0xE24A7001	SUB r7,r10,#1
6	-1487			0x0000801C	0xE15A000B	CMP r10,r11
7	-1486			0x00008020	0x0A00018B	BEQ __rt_entry <0x8654>
8	-1485			0x00008024	0xE8BA000F	LDM r10!,{r0-r3}
9	-1484	Exec	__scatterload_null	0x00008028	0xE24FE014	ADR lr,{pc}-0xc ; 0x801c
10	-1483	Exec	__scatterload_null	0x0000802C	0xE3130001	TST r3,#1
11	-1482	NoExec	__scatterload_null	0x00008030	0x1047F003	SUBNE pc,r7,r3
12	-1481	Exec	__scatterload_null	0x00008034	0xE1A0F003	MOV pc,r3
13	-1480	Exec	__scatterload_copy	0x00008040	0xE2522010	SUBS r2,r2,#0x10
14	-1479	NoExec	__scatterload_copy	0x00008044	0x28B00078	LDMCS r0!,{r3-r6}
15	-1478	NoExec	__scatterload_copy	0x00008048	0x28A10078	STMCS r1!,{r3-r6}
16	-1477	NoExec	__scatterload_copy	0x0000804C	0x8AFFFFFB	BHI __scatterload_copy <0x8040>
17	-1476	Exec	__scatterload_copy	0x00008050	0xE1B02E82	LSLS r2,r2,#29
18	-1475	NoExec	__scatterload_copy	0x00008054	0x28B00030	LDMCS r0!,{r4,r5}
19	-1474	NoExec	__scatterload_copy	0x00008058	0x28A10030	STMCS r1!,{r4,r5}
20	-1473	Exec	__scatterload_copy	0x0000805C	0x45904000	LDRMI r4,[r0,#0]
21	-1472	Exec	__scatterload_copy	0x00008060	0x45814000	STRMI r4,[r1,#0]
22	-1471	Exec	__scatterload_copy	0x00008064	0xE1A0F00E	MOV pc,lr
23	-1470	Exec	__scatterload_null	0x0000801C	0xE15A000B	CMP r10,r11
24	-1469	NoExec	__scatterload_null	0x00008020	0x0A00018B	BEQ __rt_entry <0x8654>
25	-1468	Exec	__scatterload_null	0x00008024	0xE8BA000F	LDM r10!,{r0-r3}
26	-1467	Exec	__scatterload_null	0x00008028	0xE24FE014	ADR lr,{pc}-0xc ; 0x801c
27	-1466	Exec	__scatterload_null	0x0000802C	0xE3130001	TST r3,#1
28	-1465	NoExec	__scatterload_null	0x00008030	0x1047F003	SUBNE pc,r7,r3

Select an entry by next/only trigger

Tracing enabled

Figure 104

Observe that the focus (red letterbox) is brought to the trigger location. The trigger is designated with an asterisk by the Element number:


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Time/inst	Type	Symbolic	Address	Opcode	Other	
1325	-168	Exec	__rt_entry	0x00008678	0xEBFFFEEC	BL	main <0x8230>
1326	-167	Exec	main	0x00008230	0xE92D4070	PUSH	{r4-r6,lr}
1327	-166	Exec	main\#66	0x00008234	0xE3A04000	MOV	r4,#0
1328	-165	Exec	main\#67..#69	0x00008238	0xEBFFFFDA	BL	Init <0x81a8>
*1329	-164	Exec	Init	0x000081A8	0xE3A00FFA	MOV	r0,#0x3e8
1330	-163	Exec	Init\#113	0x000081AC	0xE59F1128	LDR	r1,0x82dc
1331	-162	Exec	Init\#113	0x000081B0	0xE5810000	STR	r0,[r1,#0]
1332	-161	Exec	Init\#114	0x000081B4	0xE3A00E7D	MOV	r0,#0x7d0
1333	-160	Exec	Init\#114	0x000081B8	0xE5810004	STR	r0,[r1,#4]
1334	-159	Exec	Init\#115	0x000081BC	0xE59F011C	LDR	r0,0x82e0
1335	-158	Exec	Init\#115	0x000081C0	0xE5810008	STR	r0,[r1,#8]
1336	-157	Exec	Init\#116	0x000081C4	0xE2400FFA	SUB	r0,r0,#0x3e8
1337	-156	Exec	Init\#116	0x000081C8	0xE581000C	STR	r0,[r1,#0xc]
1338	-155	Exec	Init\#117	0x000081CC	0xE59F0110	LDR	r0,0x82e4
1339	-154	Exec	Init\#117	0x000081D0	0xE5810010	STR	r0,[r1,#0x10]
1340	-153	Exec	Init\#118	0x000081D4	0xE2800FFA	ADD	r0,r0,#0x3e8
1341	-152	Exec	Init\#118	0x000081D8	0xE5810014	STR	r0,[r1,#0x14]
1342	-151	Exec	Init\#119	0x000081DC	0xE59F0104	LDR	r0,0x82e8
1343	-150	Exec	Init\#119	0x000081E0	0xE5810018	STR	r0,[r1,#0x18]
1344	-149	Exec	Init\#120	0x000081E4	0xE1A00080	LSL	r0,r0,#1
1345	-148	Exec	Init\#120	0x000081E8	0xE581001C	STR	r0,[r1,#0x1c]
1346	-147	Exec	Init\#121	0x000081EC	0xE3A00E7D	MOV	r0,#0x7d0
1347	-146	Exec	Init\#121	0x000081F0	0xE5810020	STR	r0,[r1,#0x20]
1348	-145	Exec	Init\#122	0x000081F4	0xE3A00FFA	MOV	r0,#0x3e8
1349	-144	Exec	Init\#122	0x000081F8	0xE5810024	STR	r0,[r1,#0x24]
1350	-143	Exec	Init\#123	0x000081FC	0xE59F00E8	LDR	r0,0x82ec
1351	-142	Exec	Init\#123	0x00008200	0xE5810028	STR	r0,[r1,#0x28]
1352	-141	Exec	Init\#124	0x00008204	0xE2400FFA	SUB	r0,r0,#0x3e8
1353	-140	Exec	Init\#124	0x00008208	0xE581002C	STR	r0,[r1,#0x2c]
1354	-139	Exec	Init\#125	0x0000820C	0xE2800D7D	ADD	r0,r0,#0x1f40
1355	-138	Exec	Init\#125	0x00008210	0xE5810030	STR	r0,[r1,#0x30]

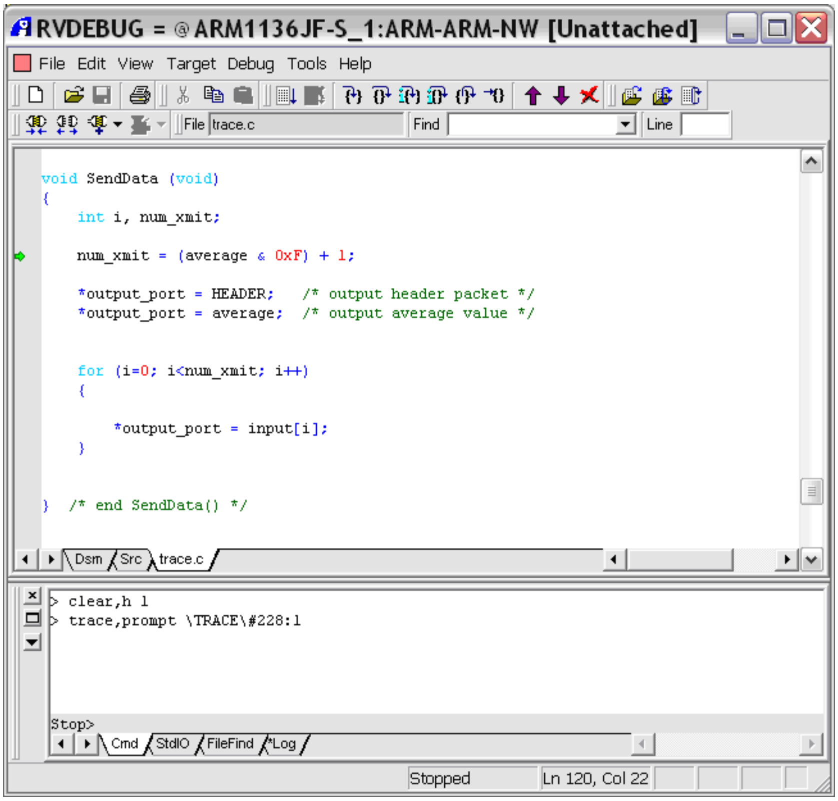
Figure 105

8.3 Advanced Trace Triggers

For more complex trace situations on running targets, you can use a conditional trigger point. A conditional trigger point allows you to specify the number of times an event (such as instruction execution or data access) must occur for trace to trigger. You can also specify a specific data access as a trigger point, similar to an RVD watchpoint. The exact capability of a conditional trigger point is based on the event resources supported by your ETM.

Using a Pass Counter for Program Execution

In this scenario a trigger is used to capture trace on the 10th execution of SendData(). Remove your previous trigger point and set a simple trace trigger at the computation of num_xmit in SendData():



RVDEBUG = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Target Debug Tools Help
File trace.c
Find
Line
Dsm
Src
trace.c
Cmd
StdIO
FileFind
Log

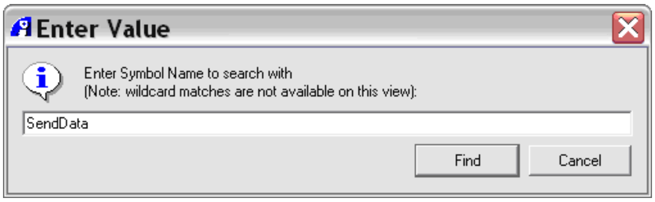
```
void SendData (void)
{
    int i, num_xmit;

    num_xmit = (average & 0xF) + 1;

    *output_port = HEADER;   /* output header packet */
    *output_port = average;  /* output average value */


    for (i=0; i<num_xmit; i++)
    {

        *output_port = input[i];
    }


}  /* end SendData() */
```

```
> clear,h 1
> trace,prompt \TRACE\#228:1
Stop>
```

Stopped   Ln 120, Col 22

Figure 106

Reload TRACE.AXF and run it. Wait for the Analysis window to populate with your trace capture. After your trace is displayed, search for SendData using the "Find - Find Symbol Name" menu from the Analysis window:

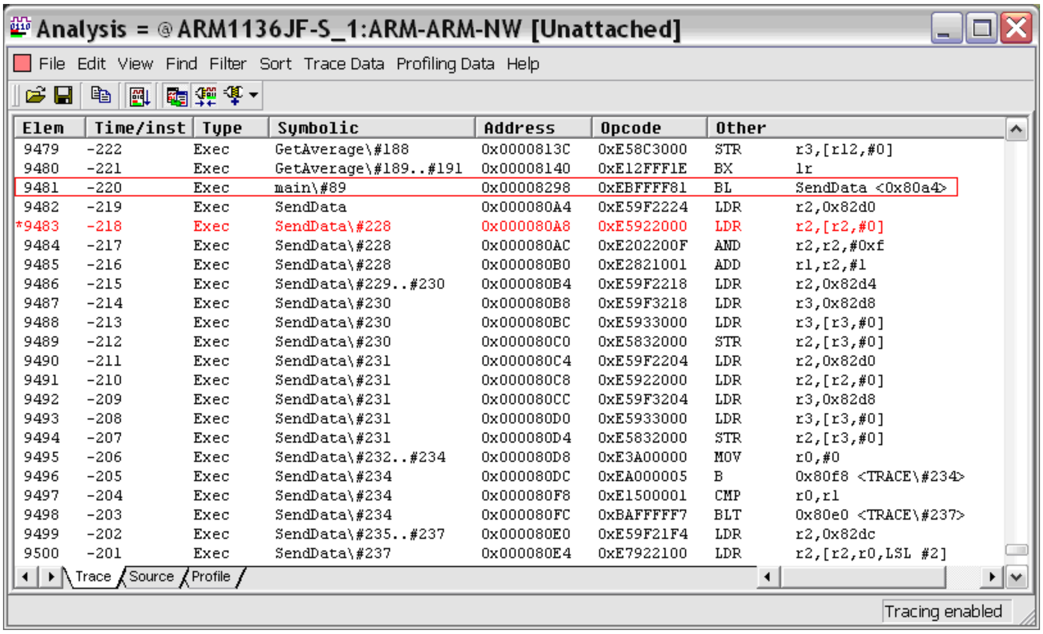


Enter Symbol Name to search with
(Note: wildcard matches are not available on this view):

SendData

Figure 107

Observe that the trace trigger is located two instructions after the very first call to SendData() at trace element 9481:

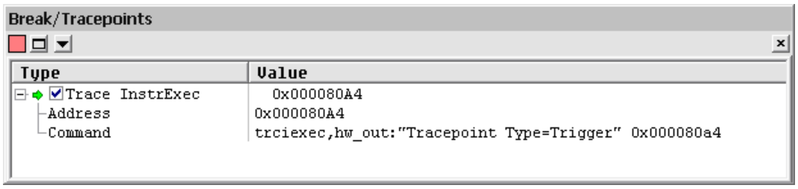


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Time/inst	Type	Symbolic	Address	Opcode	Other	
9479	-222	Exec	GetAverage\#188	0x0000813C	0xE58C3000	STR	r3,[r12,#0]
9480	-221	Exec	GetAverage\#189..#191	0x00008140	0xE12FFF1E	BX	lr
9481	-220	Exec	main\#89	0x00008298	0xEBFFFF81	BL	SendData <0x80a4>
9482	-219	Exec	SendData	0x000080A4	0xE59F2224	LDR	r2,0x82d0
*9483	-218	Exec	SendData\#228	0x000080A8	0xE5922000	LDR	r2,[r2,#0]
9484	-217	Exec	SendData\#228	0x000080AC	0xE202200F	AND	r2,r2,#0xf
9485	-216	Exec	SendData\#228	0x000080B0	0xE2821001	ADD	r1,r2,#1
9486	-215	Exec	SendData\#229..#230	0x000080B4	0xE59F2218	LDR	r2,0x82d4
9487	-214	Exec	SendData\#230	0x000080B8	0xE59F3218	LDR	r3,0x82d8
9488	-213	Exec	SendData\#230	0x000080BC	0xE5933000	LDR	r3,[r3,#0]
9489	-212	Exec	SendData\#230	0x000080C0	0xE5832000	STR	r2,[r3,#0]
9490	-211	Exec	SendData\#231	0x000080C4	0xE59F2204	LDR	r2,0x82d0
9491	-210	Exec	SendData\#231	0x000080C8	0xE5922000	LDR	r2,[r2,#0]
9492	-209	Exec	SendData\#231	0x000080CC	0xE59F3204	LDR	r3,0x82d8
9493	-208	Exec	SendData\#231	0x000080D0	0xE5933000	LDR	r3,[r3,#0]
9494	-207	Exec	SendData\#231	0x000080D4	0xE5832000	STR	r2,[r3,#0]
9495	-206	Exec	SendData\#232..#234	0x000080D8	0xE3A00000	MOV	r0,#0
9496	-205	Exec	SendData\#234	0x000080DC	0xEA000005	B	0x80f8 <TRACE\#234>
9497	-204	Exec	SendData\#234	0x000080F8	0xE1500001	CMP	r0,r1
9498	-203	Exec	SendData\#234	0x000080FC	0xBAFFFFF7	BLT	0x80e0 <TRACE\#237>
9499	-202	Exec	SendData\#235..#237	0x000080E0	0xE59F21F4	LDR	r2,0x82dc
9500	-201	Exec	SendData\#237	0x000080E4	0xE7922100	LDR	r2,[r2,r0,LSL #2]

Figure 108

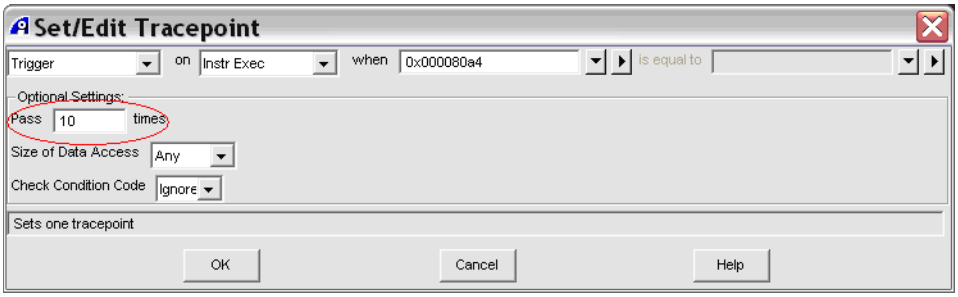
Halt your target. Now display the trigger using the "View - Break/Tracepoints" menu:



Type	Value
☑ Trace InstrExec	0x000080A4
Address	0x000080A4
Command	trciexec,hw_out:"Tracepoint Type=Trigger" 0x000080a4

Figure 109

Right-click on the trace point and select "Edit Break/Tracepoint". Modify the trace point such that the trigger occurs after the instruction is executed 10 times. Do this is by setting the "Pass" parameter to "10":



Trigger on Instr Exec when 0x000080a4 is equal to

Optional Settings:

Pass 10 times

Size of Data Access Any

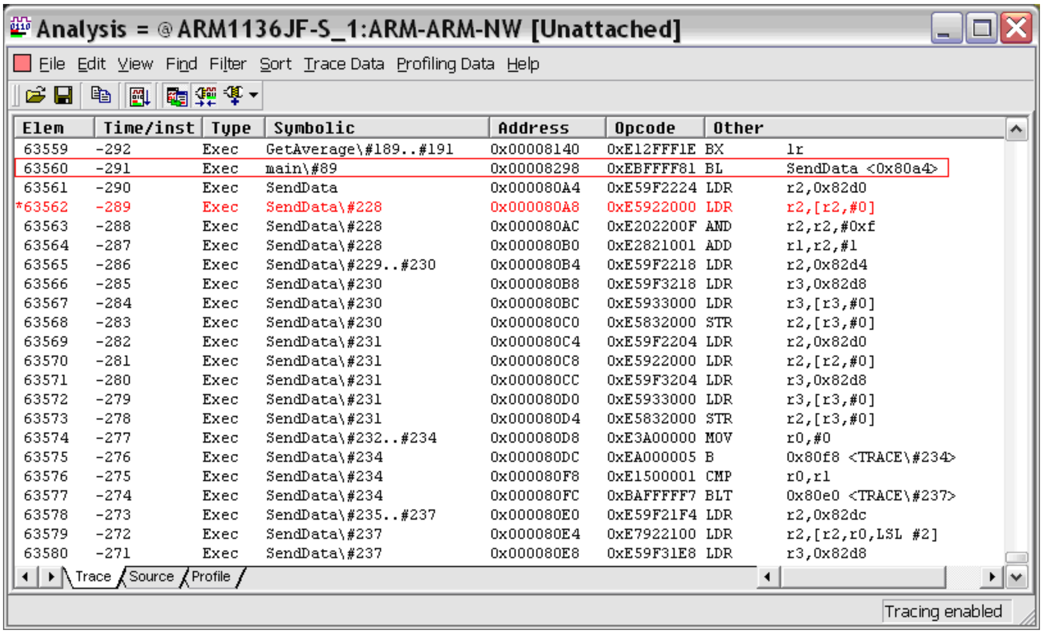
Check Condition Code Ignore

Sets one tracepoint

Figure 110

Clear the previous trace from the Analysis window using the "View - Clear Trace Buffer" menu. Reload TRACE.AXF and run the program again. Wait for the Analysis window to populate with the new trace capture.

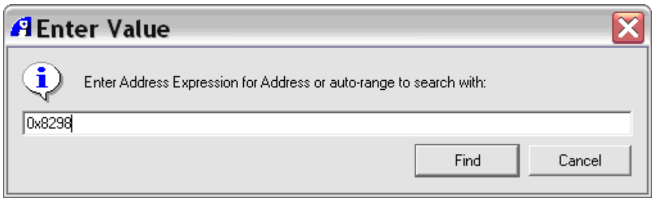
Observe that the new capture starts from the program entry point at 0x8000. Search the buffer again for the trigger point using "Find - Find Trigger". Observe that the trigger is now located at element 63562:


Analysis = @ ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

```
Elem    Time/inst Type  Symbolic                 Address     Opcode        Other
 63559  -292      Exec  GetAverage\#189..#191    0x00008140  0xE12FFF1E BX   lr
 63560  -291      Exec  main\#89                 0x00008298  0xEBFFFF81 BL   SendData <0x80a4>
 63561  -290      Exec  SendData                 0x000080A4  0xE59F2224 LDR  r2,0x82d0
*63562  -289      Exec  SendData\#228            0x000080A8  0xE5922000 LDR  r2,[r2,#0]
 63563  -288      Exec  SendData\#228            0x000080AC  0xE202200F AND  r2,r2,#0xf
 63564  -287      Exec  SendData\#228            0x000080B0  0xE2821001 ADD  r1,r2,#1
 63565  -286      Exec  SendData\#229..#230      0x000080B4  0xE59F2218 LDR  r2,0x82d4
 63566  -285      Exec  SendData\#230            0x000080B8  0xE59F3218 LDR  r3,0x82d8
 63567  -284      Exec  SendData\#230            0x000080BC  0xE5933000 LDR  r3,[r3,#0]
 63568  -283      Exec  SendData\#230            0x000080C0  0xE5832000 STR  r2,[r3,#0]
 63569  -282      Exec  SendData\#231            0x000080C4  0xE59F2204 LDR  r2,0x82d0
 63570  -281      Exec  SendData\#231            0x000080C8  0xE5922000 LDR  r2,[r2,#0]
 63571  -280      Exec  SendData\#231            0x000080CC  0xE59F3204 LDR  r3,0x82d8
 63572  -279      Exec  SendData\#231            0x000080D0  0xE5933000 LDR  r3,[r3,#0]
 63573  -278      Exec  SendData\#231            0x000080D4  0xE5832000 STR  r2,[r3,#0]
 63574  -277      Exec  SendData\#232..#234      0x000080D8  0xE3A00000 MOV  r0,#0
 63575  -276      Exec  SendData\#234            0x000080DC  0xEA000005 B    0x80f8 <TRACE\#234>
 63576  -275      Exec  SendData\#234            0x000080F8  0xE1500001 CMP  r0,r1
 63577  -274      Exec  SendData\#234            0x000080FC  0xBAFFFFF7 BLT  0x80e0 <TRACE\#237>
 63578  -273      Exec  SendData\#235..#237      0x000080E0  0xE59F21F4 LDR  r2,0x82dc
 63579  -272      Exec  SendData\#237            0x000080E4  0xE7922100 LDR  r2,[r2,r0,LSL #2]
 63580  -271      Exec  SendData\#237            0x000080E8  0xE59F31E8 LDR  r3,0x82d8
```

Figure 111

Scroll back to the top of the trace buffer, and search for address 0x8298 using the "Find - Find Address Expression" menu. As shown above, address 0x8298 contains the call to SendData:



Enter Address Expression for Address or auto-range to search with:

0x8298

Figure 112

Repeat the search 9 times using the "Find - Find Next" menu (or "F3") to confirm that the trigger occurred on the 10th execution of the num_xmit computation. Observe that after the 9 searches you return to trace element 63560.

Note

If you are tracing with an ETB, your trace buffer may not be large enough to hold all 10 executions of SendData(). To see the effects of the pass counter, reduce the trigger's pass count accordingly and perform another trace capture.

Using a Pass Counter for Data Accesses

In this scenario a trigger is used to capture the 2000th data write to *output_port. Begin by enabling data trace by setting the Automatic Tracing Mode to "Instructions and Data" as in Section 7.1:

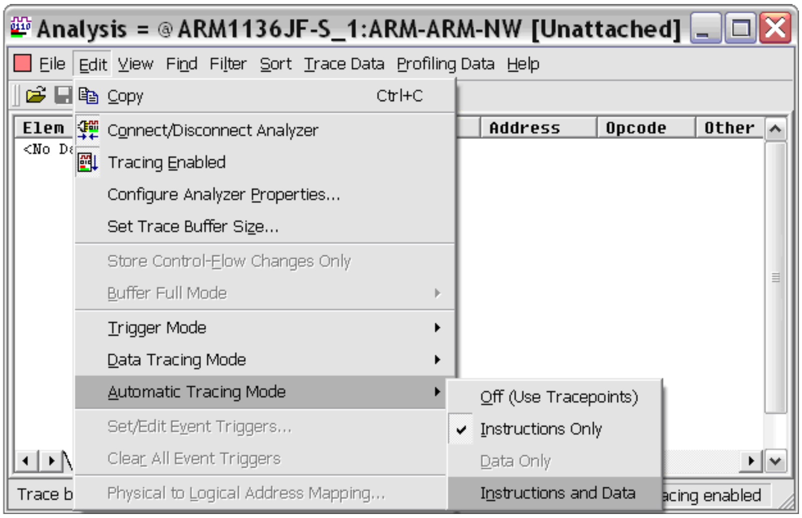
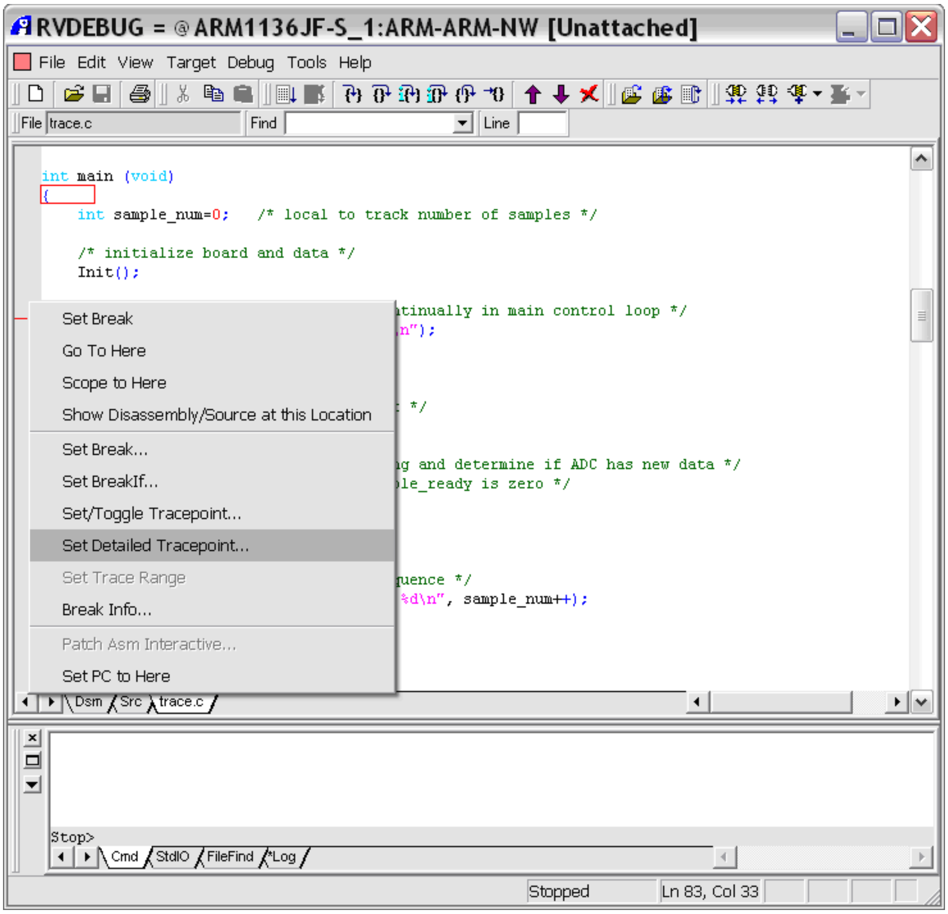

Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Copy Ctrl+C
Connect/Disconnect Analyzer
Tracing Enabled
Configure Analyzer Properties...
Set Trace Buffer Size...
Store Control-Flow Changes Only
Buffer Full Mode
Trigger Mode
Data Tracing Mode
Automatic Tracing Mode
Set/Edit Event Triggers...
Clear All Event Triggers
Physical to Logical Address Mapping...
Off (Use Tracepoints)
Instructions Only
Data Only
Instructions and Data
Elem
<No Da
Address
Opcode
Other
Trace b
acing enabled

Figure 113

Remove the trigger point set in Section 8.3.1. Set a new trigger point by clicking anywhere in the left gutter and selecting "Set Detailed Tracepoint…":

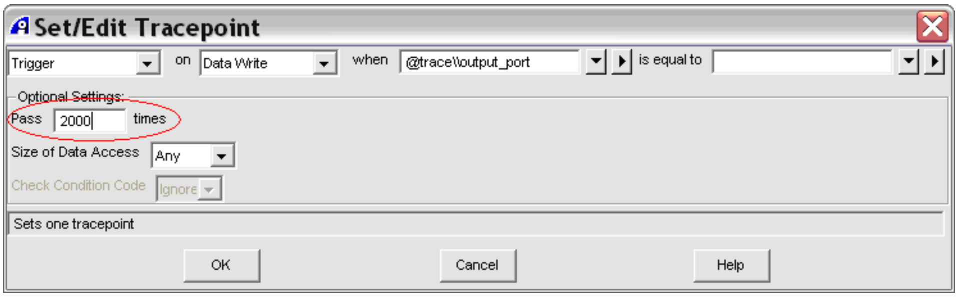


```
int main (void)
{
    int sample_num=0;   /* local to track number of samples */

    /* initialize board and data */
    Init();

                                        tinually in main control loop */
                                        n");

                                        */

                                        g and determine if ADC has new data */
                                        ble_ready is zero */

                                        quence */
                                        %d\n", sample_num++);
```

```
Stop>
```

Figure 114

Following the instructions in Section 7.3, set the trigger to activate on the 2000th data write to output_port:



Trigger on Data Write when @trace\\output_port is equal to

Optional Settings:

Pass 2000 times

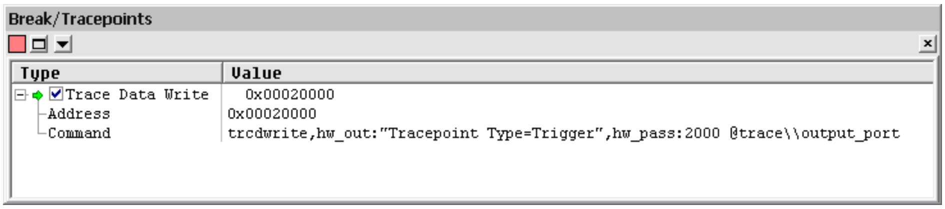
Size of Data Access Any

Check Condition Code Ignore

Sets one tracepoint

Figure 115

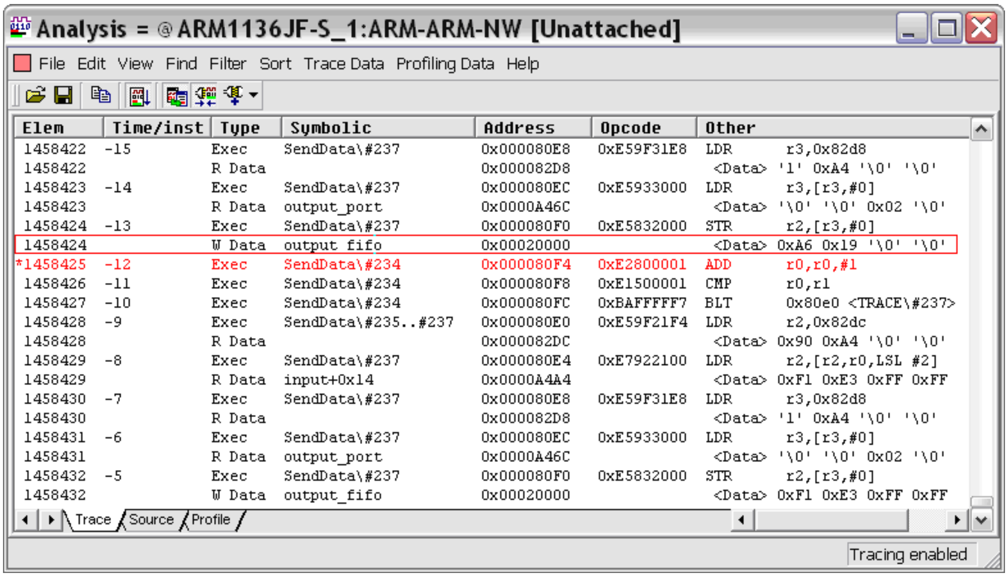
Inspect the trace point using "View - Break/Tracepoints" from RVD to confirm that it has been set correctly:



Type	Value
☑ Trace Data Write	0x00020000
Address	0x00020000
Command	trcdwrite,hw_out:"Tracepoint Type=Trigger",hw_pass:2000 @trace\\output_port

Figure 116

Reload TRACE.AXF and run it. If you are using RVT, observe that the TRIG LED lights soon after the program prints "Processing Sample: 200" in RVD. Wait several seconds for RVD to display the trace. When the trace is displayed, search for the trigger point using "Find - Trigger". Observe that the trigger point occurs just after the 2000th write:



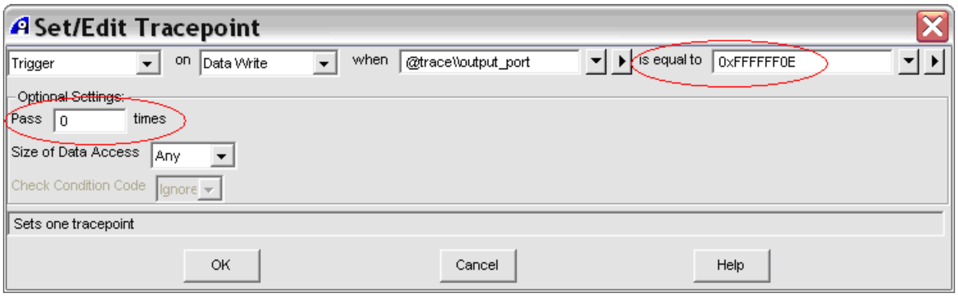
Elem	Time/inst	Type	Symbolic	Address	Opcode	Other
1458422	-15	Exec	SendData\#237	0x000080E8	0xE59F31E8	LDR r3,0x82d8
1458422		R Data		0x000082D8		<Data> '1' 0xA4 '\0' '\0'
1458423	-14	Exec	SendData\#237	0x000080EC	0xE5933000	LDR r3,[r3,#0]
1458423		R Data	output_port	0x0000A46C		<Data> '\0' '\0' 0x02 '\0'
1458424	-13	Exec	SendData\#237	0x000080F0	0xE5832000	STR r2,[r3,#0]
1458424		W Data	output fifo	0x00020000		<Data> 0xA6 0x19 '\0' '\0'
*1458425	-12	Exec	SendData\#234	0x000080F4	0xE2800001	ADD r0,r0,#1
1458426	-11	Exec	SendData\#234	0x000080F8	0xE1500001	CMP r0,r1
1458427	-10	Exec	SendData\#234	0x000080FC	0xBAFFFFF7	BLT 0x80e0 <TRACE\#237>
1458428	-9	Exec	SendData\#235..#237	0x000080E0	0xE59F21F4	LDR r2,0x82dc
1458428		R Data		0x000082DC		<Data> 0x90 0xA4 '\0' '\0'
1458429	-8	Exec	SendData\#237	0x000080E4	0xE7922100	LDR r2,[r2,r0,LSL #2]
1458429		R Data	input+0x14	0x0000A4A4		<Data> 0xF1 0xE3 0xFF 0xFF
1458430	-7	Exec	SendData\#237	0x000080E8	0xE59F31E8	LDR r3,0x82d8
1458430		R Data		0x000082D8		<Data> '1' 0xA4 '\0' '\0'
1458431	-6	Exec	SendData\#237	0x000080EC	0xE5933000	LDR r3,[r3,#0]
1458431		R Data	output_port	0x0000A46C		<Data> '\0' '\0' 0x02 '\0'
1458432	-5	Exec	SendData\#237	0x000080F0	0xE5832000	STR r2,[r3,#0]
1458432		W Data	output_fifo	0x00020000		<Data> 0xF1 0xE3 0xFF 0xFF

Figure 117

Triggering on a Specific Data Accesses

A trace trigger can also be used to capture a specific data value access to a specific data address. In this scenario the trace trigger will occur when the value 0xFFFFFF0E is written to *output_port. In TRACE.AXF, the value 0xFFFFFF0E is the 100th computed average so we expect this value to be written to *output_port.

Edit the trigger point set in Section 8.3.2 to activate when the value 0xFFFFFF0E is written to *output_port:



Set/Edit Tracepoint

Trigger on Data Write when @trace\output_port is equal to 0xFFFFFF0E

Optional Settings:

Pass 0 times

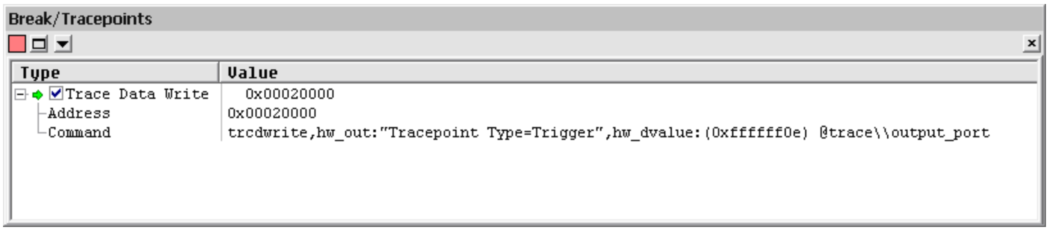
Size of Data Access: Any

Check Condition Code: Ignore

Sets one tracepoint

Figure 118

Inspect the trace point using "View - Break/Tracepoints" from RVD to confirm that it has been set correctly:

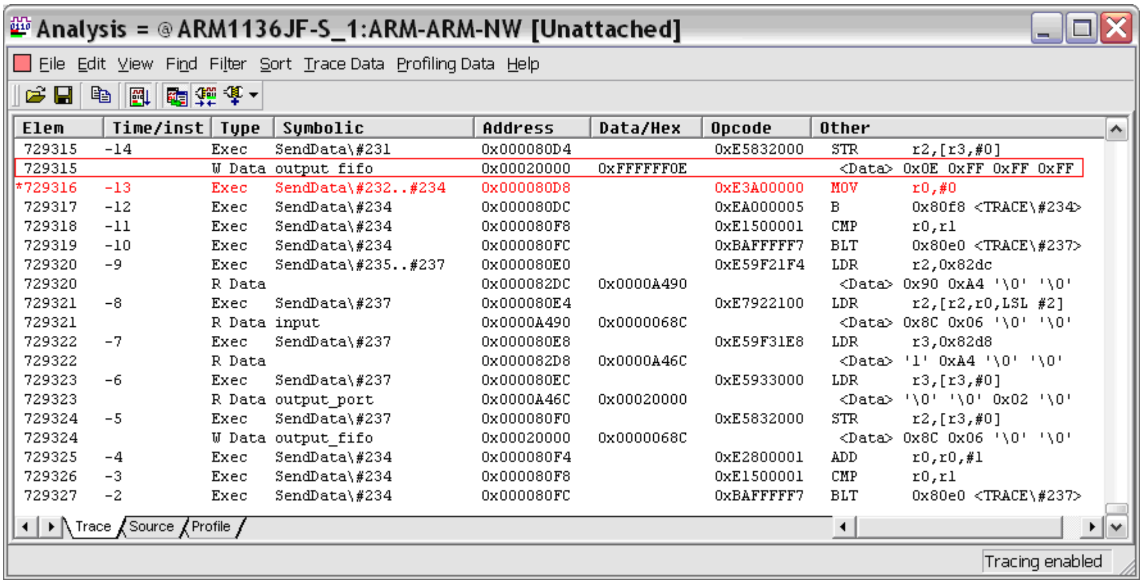


Break/Tracepoints

Type	Value
☑ Trace Data Write	0x00020000
Address	0x00020000
Command	trcdwrite,hw_out:"Tracepoint Type=Trigger",hw_dvalue:(0xffffff0e) @trace\\output_port

Figure 119

Reload TRACE.AXF and run it. If you are using RVT, observe that the TRIG LED lights soon after the program prints "Processing Sample: 100" in RVD. Wait several seconds for RVD to display the trace. When the trace is displayed, search for the trigger point using "Find - Trigger". Observe that the trigger point occurs just after the write of 0xFFFFFF0E to address 0x20000 (*output_port):



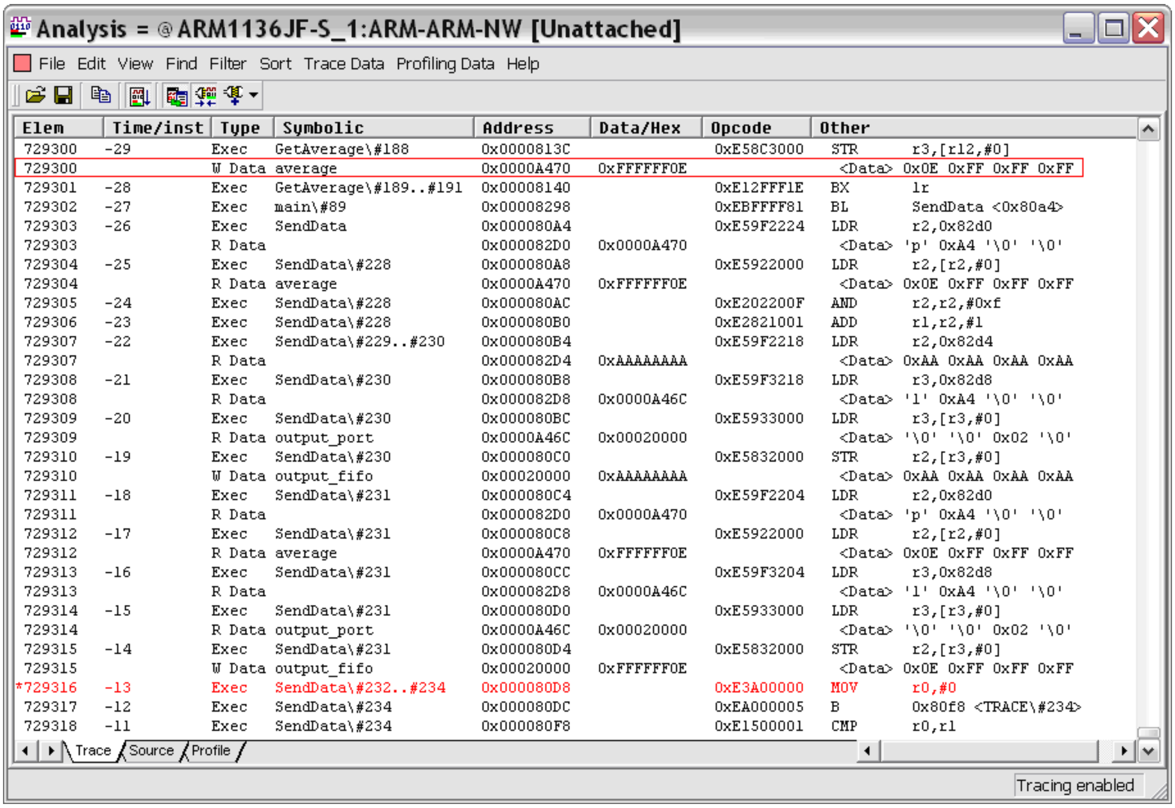
Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]

File Edit View Find Filter Sort Trace Data Profiling Data Help

Elem	Time/inst	Type	Symbolic	Address	Data/Hex	Opcode	Other
729315	-14	Exec	SendData\#231	0x000080D4		0xE5832000	STR r2,[r3,#0]
729315		W Data	output fifo	0x00020000	0xFFFFFF0E		<Data> 0x0E 0xFF 0xFF 0xFF
*729316	-13	Exec	SendData\#232..#234	0x000080D8		0xE3A00000	MOV r0,#0
729317	-12	Exec	SendData\#234	0x000080DC		0xEA000005	B 0x80f8 <TRACE\#234>
729318	-11	Exec	SendData\#234	0x000080F8		0xE1500001	CMP r0,r1
729319	-10	Exec	SendData\#234	0x000080FC		0xBAFFFFF7	BLT 0x80e0 <TRACE\#237>
729320	-9	Exec	SendData\#235..#237	0x000080E0		0xE59F21F4	LDR r2,0x82dc
729320		R Data		0x000082DC	0x0000A490		<Data> 0x90 0xA4 '\0' '\0'
729321	-8	Exec	SendData\#237	0x000080E4		0xE7922100	LDR r2,[r2,r0,LSL #2]
729321		R Data	input	0x0000A490	0x0000068C		<Data> 0x8C 0x06 '\0' '\0'
729322	-7	Exec	SendData\#237	0x000080E8		0xE59F31E8	LDR r3,0x82d8
729322		R Data		0x000082D8	0x0000A46C		<Data> 'l' 0xA4 '\0' '\0'
729323	-6	Exec	SendData\#237	0x000080EC		0xE5933000	LDR r3,[r3,#0]
729323		R Data	output_port	0x0000A46C	0x00020000		<Data> '\0' '\0' 0x02 '\0'
729324	-5	Exec	SendData\#237	0x000080F0		0xE5832000	STR r2,[r3,#0]
729324		W Data	output_fifo	0x00020000	0x0000068C		<Data> 0x8C 0x06 '\0' '\0'
729325	-4	Exec	SendData\#234	0x000080F4		0xE2800001	ADD r0,r0,#1
729326	-3	Exec	SendData\#234	0x000080F8		0xE1500001	CMP r0,r1
729327	-2	Exec	SendData\#234	0x000080FC		0xBAFFFFF7	BLT 0x80e0 <TRACE\#237>

Figure 120

Scroll up in the trace buffer and observe that in fact 0xFFFFFF0E is the average value computed in GetAverage and written to "average" (located at 0xA470):


Analysis = @ARM1136JF-S_1:ARM-ARM-NW [Unattached]
File Edit View Find Filter Sort Trace Data Profiling Data Help
Trace
Source
Profile
Tracing enabled

Elem	Time/inst	Type	Symbolic	Address	Data/Hex	Opcode	Other
729300	-29	Exec	GetAverage\#188	0x0000813C		0xE58C3000	STR r3,[r12,#0]
729300		W Data	average	0x0000A470	0xFFFFFF0E		<Data> 0x0E 0xFF 0xFF 0xFF
729301	-28	Exec	GetAverage\#189..#191	0x00008140		0xE12FFF1E	BX lr
729302	-27	Exec	main\#89	0x00008298		0xEBFFFF81	BL SendData <0x80a4>
729303	-26	Exec	SendData	0x000080A4		0xE59F2224	LDR r2,0x82d0
729303		R Data		0x000082D0	0x0000A470		<Data> 'p' 0xA4 '\0' '\0'
729304	-25	Exec	SendData\#228	0x000080A8		0xE5922000	LDR r2,[r2,#0]
729304		R Data	average	0x0000A470	0xFFFFFF0E		<Data> 0x0E 0xFF 0xFF 0xFF
729305	-24	Exec	SendData\#228	0x000080AC		0xE202200F	AND r2,r2,#0xf
729306	-23	Exec	SendData\#228	0x000080B0		0xE2821001	ADD r1,r2,#1
729307	-22	Exec	SendData\#229..#230	0x000080B4		0xE59F2218	LDR r2,0x82d4
729307		R Data		0x000082D4	0xAAAAAAAA		<Data> 0xAA 0xAA 0xAA 0xAA
729308	-21	Exec	SendData\#230	0x000080B8		0xE59F3218	LDR r3,0x82d8
729308		R Data		0x000082D8	0x0000A46C		<Data> 'l' 0xA4 '\0' '\0'
729309	-20	Exec	SendData\#230	0x000080BC		0xE5933000	LDR r3,[r3,#0]
729309		R Data	output_port	0x0000A46C	0x00020000		<Data> '\0' '\0' 0x02 '\0'
729310	-19	Exec	SendData\#230	0x000080C0		0xE5832000	STR r2,[r3,#0]
729310		W Data	output_fifo	0x00020000	0xAAAAAAAA		<Data> 0xAA 0xAA 0xAA 0xAA
729311	-18	Exec	SendData\#231	0x000080C4		0xE59F2204	LDR r2,0x82d0
729311		R Data		0x000082D0	0x0000A470		<Data> 'p' 0xA4 '\0' '\0'
729312	-17	Exec	SendData\#231	0x000080C8		0xE5922000	LDR r2,[r2,#0]
729312		R Data	average	0x0000A470	0xFFFFFF0E		<Data> 0x0E 0xFF 0xFF 0xFF
729313	-16	Exec	SendData\#231	0x000080CC		0xE59F3204	LDR r3,0x82d8
729313		R Data		0x000082D8	0x0000A46C		<Data> 'l' 0xA4 '\0' '\0'
729314	-15	Exec	SendData\#231	0x000080D0		0xE5933000	LDR r3,[r3,#0]
729314		R Data	output_port	0x0000A46C	0x00020000		<Data> '\0' '\0' 0x02 '\0'
729315	-14	Exec	SendData\#231	0x000080D4		0xE5832000	STR r2,[r3,#0]
729315		W Data	output_fifo	0x00020000	0xFFFFFF0E		<Data> 0x0E 0xFF 0xFF 0xFF
*729316	-13	Exec	SendData\#232..#234	0x000080D8		0xE3A00000	MOV r0,#0
729317	-12	Exec	SendData\#234	0x000080DC		0xEA000005	B 0x80f8 <TRACE\#234>
729318	-11	Exec	SendData\#234	0x000080F8		0xE1500001	CMP r0,r1

Figure 121

Scroll further up the trace buffer and look for another value that is written to *output_port. Modify the trace point to trigger on the data write to *output_port of this new value. Reload TRACE.AXF and repeat the trace capture using the new trigger point. Confirm that the trigger point of the second capture is located at the correct location.