Application Note for Trace.DRAW

Release 02.2025

Application Note for Trace.DRAW

Application Note for Trace.DRAW

Version 13-Feb-2025

Introduction

Many Embedded devices feature some kind of trace port. The most prevalent are those that provide branch trace, which is sufficient to reconstruct program flow. Some, more complete, trace interfaces can also provide data trace - the means to track a value changing over time. This application note is designed to help the reader understand the ways that data values can be captured, analyzed, displayed and exported.

The examples shown in this document use the target's off-chip trace. The techniques described here are equally applicable to an onchip trace and TRACE32 'pseudo-trace' methods such as FDX, SNOOPer, LOGGER, etc. where data is logged by the user. This also includes ITM software generated trace from Cortex-M devices. This document will not detail how to configure the target trace port; that is covered in more detail elsewhere.

Intended Audience

Developers who want to:

- Monitor variable values non-intrusively
- Analyze, draw or chart the value of a variable as it changes over time
- Export these data values for analysis by external tools

Prerequisites

It is assumed that TRACE32 has been correctly configured for the target and the symbols for the application under test have been loaded. The trace port must also be configured to provide data trace and TRACE32 must be able to collect this data. The reader should also be familiar with programming embedded systems.

Related Documents

Trace covers a wide range of processors and technologies. The documents listed below contain more information on configuring, collecting and analyzing trace data with TRACE32. This is by no means an exhaustive list.

"MicroTrace for Cortex-M User's Guide" (microtrace_cortexm.pdf)

“Training Cortex-M Tracing” (training_cortexm_etm.pdf)

“Training MPC5xxx/SPC5xx Nexus Tracing” (training_nexus_mpc5500.pdf)

“MCDS User’s Guide” (mcds_user.pdf)

“Application Note for the SNOOPer Trace” (app_snooper.pdf)

“Application Note for Trace.Find” (app_trace_find.pdf)

“Trace” in General Commands Reference Guide T, page 117 (general_ref_t.pdf)

Please also check the **Processor Architecture Manual** for your selected processor.

Restrictions

Some devices only provide small on-chip trace buffers for storing collected trace data; these are seldom sufficient for managing and analyzing trace data.

Trace Licenses

A trace pre-processor or a TRACE32 PowerTrace Serial module suitable for the target architecture is required for off-chip trace collection. On-chip trace collection can be performed via the debug interface and an appropriate license in the debug cable is required for this.

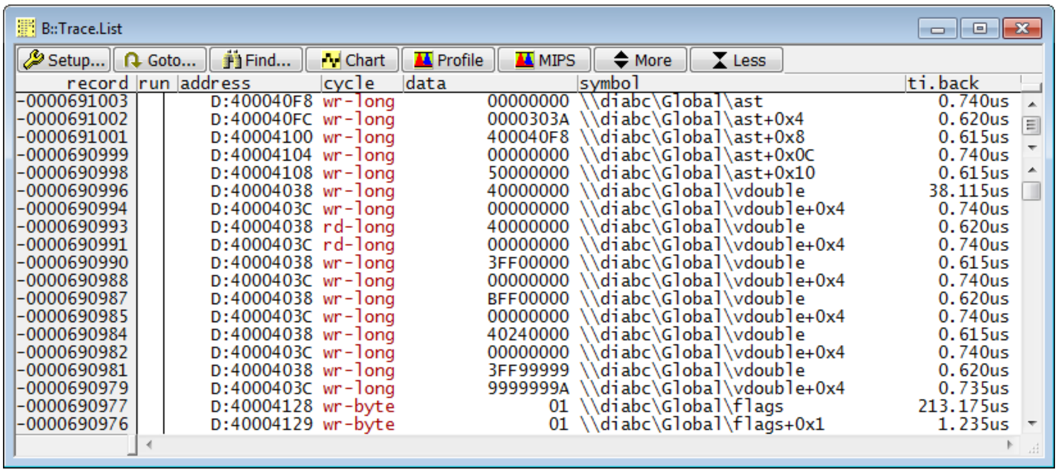
The ‘pseudo-trace’ methods such as FDX, SNOOPer, LOGGER, etc. do not require any trace licenses.

Data Capture

Full Trace

Not all targets with a trace port support data trace. Please refer to the target guides for your processor for more details.

The image below shows a typical list of all reads and writes captured via the trace port.

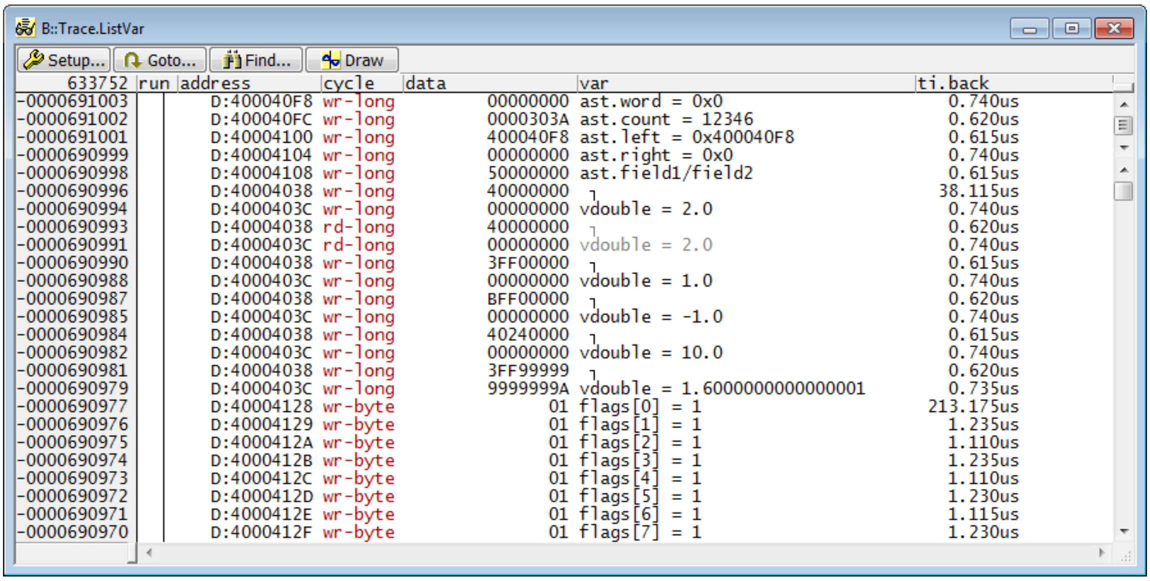


```
      record |run |address          |cycle    |data            |symbol                             |ti.back
-0000691003              D:400040F8 wr-long          00000000 \\diabc\Global\ast               0.740us
-0000691002              D:400040FC wr-long          0000303A \\diabc\Global\ast+0x4           0.620us
-0000691001              D:40004100 wr-long          400040F8 \\diabc\Global\ast+0x8           0.615us
-0000690999              D:40004104 wr-long          00000000 \\diabc\Global\ast+0x0C          0.740us
-0000690998              D:40004108 wr-long          50000000 \\diabc\Global\ast+0x10          0.615us
-0000690996              D:40004038 wr-long          40000000 \\diabc\Global\vdouble          38.115us
-0000690994              D:4000403C wr-long          00000000 \\diabc\Global\vdouble+0x4       0.740us
-0000690993              D:40004038 rd-long          40000000 \\diabc\Global\vdouble           0.620us
-0000690991              D:4000403C rd-long          00000000 \\diabc\Global\vdouble+0x4       0.740us
-0000690990              D:40004038 wr-long          3FF00000 \\diabc\Global\vdouble           0.615us
-0000690988              D:4000403C wr-long          00000000 \\diabc\Global\vdouble+0x4       0.740us
-0000690987              D:40004038 wr-long          BFF00000 \\diabc\Global\vdouble           0.620us
-0000690985              D:4000403C wr-long          00000000 \\diabc\Global\vdouble+0x4       0.740us
-0000690984              D:40004038 wr-long          40240000 \\diabc\Global\vdouble           0.615us
-0000690982              D:4000403C wr-long          00000000 \\diabc\Global\vdouble+0x4       0.740us
-0000690981              D:40004038 wr-long          3FF99999 \\diabc\Global\vdouble           0.620us
-0000690979              D:4000403C wr-long          9999999A \\diabc\Global\vdouble+0x4       0.735us
-0000690977              D:40004128 wr-byte                01 \\diabc\Global\flags           213.175us
-0000690976              D:40004129 wr-byte                01 \\diabc\Global\flags+0x1         1.235us
```

The columns show:

record	The trace record number
address	The address that was the target of a load or store operation
cycle	The type of cycle (read or write) and the access width
data	The value that was read or written
symbol	If this matches a symbol in the application it will be shown here
ti.back	The time since the previous entry in the list

The command **Trace.ListVar** may also be used which filters out just the reads and writes to variables from a complete trace and displays the cycle type, name and value.

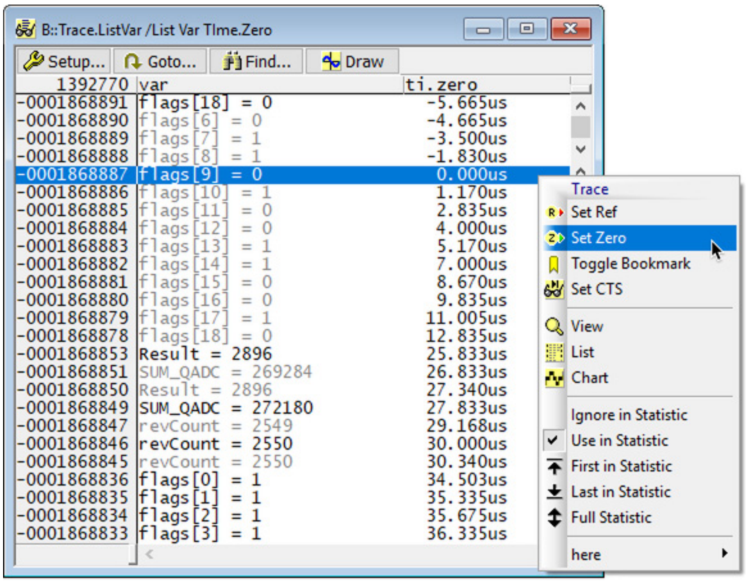


```
     633752 run address      cycle   data            var                              ti.back
-0000691003       D:400040F8 wr-long       00000000 ast.word = 0x0                     0.740us
-0000691002       D:400040FC wr-long       0000303A ast.count = 12346                  0.620us
-0000691001       D:40004100 wr-long       400040F8 ast.left = 0x400040F8              0.615us
-0000690999       D:40004104 wr-long       00000000 ast.right = 0x0                    0.740us
-0000690998       D:40004108 wr-long       50000000 ast.field1/field2                  0.615us
-0000690996       D:40004038 wr-long       40000000 ┐                                 38.115us
-0000690994       D:4000403C wr-long       00000000 vdouble = 2.0                      0.740us
-0000690993       D:40004038 rd-long       40000000 ┐                                  0.620us
-0000690991       D:4000403C rd-long       00000000 vdouble = 2.0                      0.740us
-0000690990       D:40004038 wr-long       3FF00000 ┐                                  0.615us
-0000690988       D:4000403C wr-long       00000000 vdouble = 1.0                      0.740us
-0000690987       D:40004038 wr-long       BFF00000 ┐                                  0.620us
-0000690985       D:4000403C wr-long       00000000 vdouble = -1.0                     0.740us
-0000690984       D:40004038 wr-long       40240000 ┐                                  0.615us
-0000690982       D:4000403C wr-long       00000000 vdouble = 10.0                     0.740us
-0000690981       D:40004038 wr-long       3FF99999 ┐                                  0.620us
-0000690979       D:4000403C wr-long       9999999A vdouble = 1.6000000000000001       0.735us
-0000690977       D:40004128 wr-byte             01 flags[0] = 1                     213.175us
-0000690976       D:40004129 wr-byte             01 flags[1] = 1                       1.235us
-0000690975       D:4000412A wr-byte             01 flags[2] = 1                       1.110us
-0000690974       D:4000412B wr-byte             01 flags[3] = 1                       1.235us
-0000690973       D:4000412C wr-byte             01 flags[4] = 1                       1.110us
-0000690972       D:4000412D wr-byte             01 flags[5] = 1                       1.230us
-0000690971       D:4000412E wr-byte             01 flags[6] = 1                       1.115us
-0000690970       D:4000412F wr-byte             01 flags[7] = 1                       1.230us
```

Please be aware of the following:

- In order to display the content of a variable of the type float, it might be necessary to combine two cycles (see the variable vdouble in the screenshot above).
- It is not possible to display the variable value of bitfields (see ast.field1 in the screenshot above).

The command **Trace.ListVar** has many options to adjust the way the data is displayed. Another example can be seen below which shows the variables, the values read (in gray), the values written (in black) and time since the zero point. The zero point reference marker, which by default is the start of the debug session, can be manually adjusted by the user, for example from the right-click Trace pull-down menu in any trace view window.



```
      1392770 var                          ti.zero
-0001868891 flags[18]  = 0                -5.665us
-0001868890 flags[6]  = 0                 -4.665us
-0001868889 flags[7]  = 1                 -3.500us
-0001868888 flags[8]  = 1                 -1.830us
-0001868887 flags[9]  = 0                  0.000us
-0001868886 flags[10]  = 1                 1.170us
-0001868885 flags[11]  = 0                 2.835us
-0001868884 flags[12]  = 0                 4.000us
-0001868883 flags[13]  = 1                 5.170us
-0001868882 flags[14]  = 1                 7.000us
-0001868881 flags[15]  = 0                 8.670us
-0001868880 flags[16]  = 0                 9.835us
-0001868879 flags[17]  = 1                11.005us
-0001868878 flags[18]  = 0                12.835us
-0001868853 Result  = 2896                25.833us
-0001868851 SUM_QADC  = 269284            26.833us
-0001868850 Result  = 2896                27.340us
-0001868849 SUM_QADC  = 272180            27.833us
-0001868847 revCount  = 2549              29.168us
-0001868846 revCount  = 2550              30.000us
-0001868845 revCount  = 2550              30.340us
-0001868836 flags[0]  = 1                 34.503us
-0001868835 flags[1]  = 1                 35.335us
-0001868834 flags[2]  = 1                 35.675us
-0001868833 flags[3]  = 1                 36.335us
```

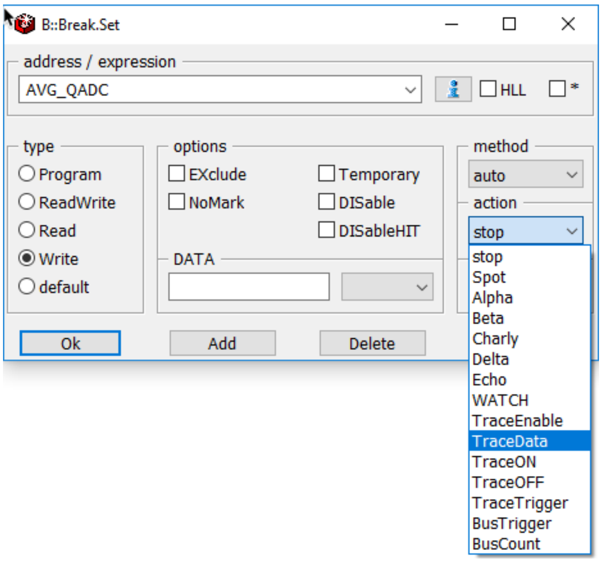
Tracing all data accesses can place a lot of strain on the trace port bandwidth. Many devices support stall or suppress options which causes the core to halt or to suppress the generation of certain kinds of trace data when internal FIFOs are in danger of overflowing and losing trace data. This minimizes the rick of losing data but may introduce some latency into a running system.

Trace data losses are called FLOWERRORs within TRACE32. **“Trace Errors”** in TRACE32 Concepts, page 62 (trace32_concepts.pdf) for more details.

The following PRACTICE script snippet can be used to programmatically check for FLOWERRORs in the trace.

```
Trace.Find FLOWERROR /ALL

IF FOUND.COUNT()==0
(
   PRINT "No flowerrors in the trace"
)
ELSE
(
    PRINT FOUND.COUNT() " flowerrors in the trace"
)
```

Many devices with a trace port also support some kind of on-chip event filtering to reduce the amount of trace data that is generated. Where these are available, TRACE32 will use the breakpoint features to control them. For more information, see **Break.Set**. The control features are implemented on the target chip and, as such, are limited in number and scope. Please refer to your chipset documentation for more details about the supported capabilities. These extra 'trace control' breakpoints can be accessed from the regular breakpoint control interface. An example can be seen in the picture below.


B::Break.Set

address / expression: AVG_QADC   ☐ HLL   ☐ *

type: ○ Program ○ ReadWrite ○ Read [✓ Write] ○ default

options: ☐ EXclude ☐ Temporary ☐ NoMark ☐ DISable ☐ DISableHIT

DATA:

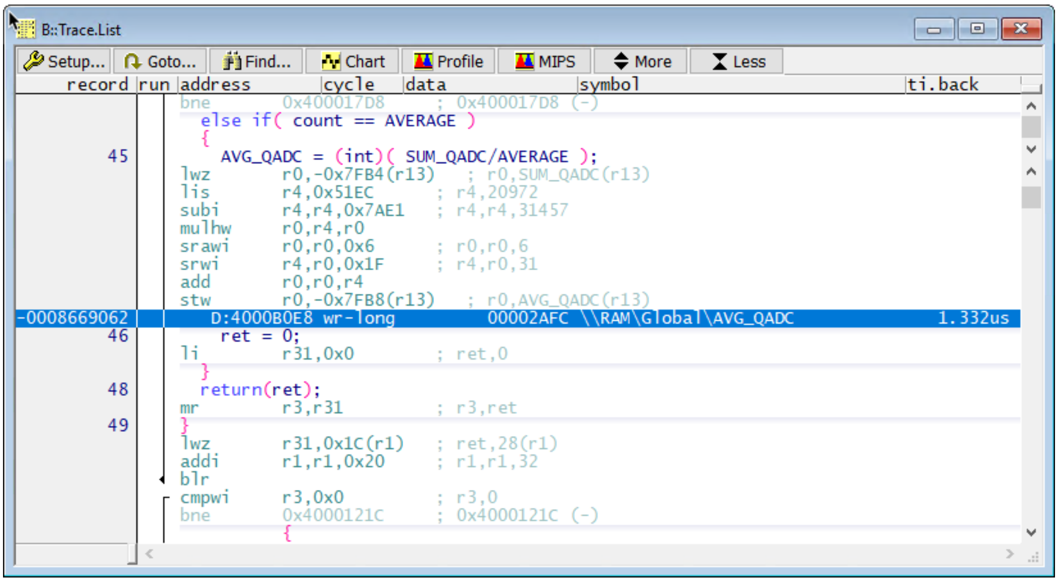
method: auto

action: stop

stop, Spot, Alpha, Beta, Charly, Delta, Echo, WATCH, TraceEnable, TraceData, TraceON, TraceOFF, TraceTrigger, BusTrigger, BusCount

Ok   Add   Delete

Setting a trace breakpoint like this will prevent all other data accesses to generate trace data: only those marked by the TraceData will be included in the generated trace.The image below shows program flow trace and a trace packet of a data write (highlighted) and is an example of how setting a filter affects the generated data.

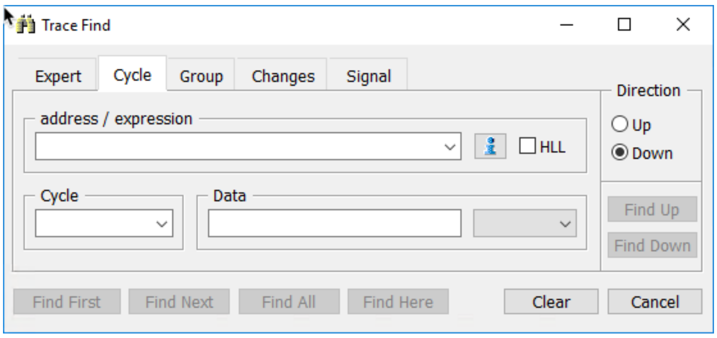

B::Trace.List

Setup...   Goto...   Find...   Chart   Profile   MIPS   More   Less

```
   record |run |address          |cycle    |data            |symbol                                          |ti.back
                bne          0x400017D8      ; 0x400017D8 (-)
                  else if( count == AVERAGE )
                  {
       45             AVG_QADC = (int)( SUM_QADC/AVERAGE );
                lwz          r0,-0x7FB4(r13)    ; r0,SUM_QADC(r13)
                lis          r4,0x51EC          ; r4,20972
                subi         r4,r4,0x7AE1       ; r4,r4,31457
                mulhw        r0,r4,r0
                srawi        r0,r0,0x6          ; r0,r0,6
                srwi         r4,r0,0x1F         ; r4,r0,31
                add          r0,r0,r4
                stw          r0,-0x7FB8(r13)    ; r0,AVG_QADC(r13)
-0008669062        D:4000B0E8 wr-long           00002AFC  \\RAM\Global\AVG_QADC              1.332us
       46             ret = 0;
                li           r31,0x0            ; ret,0
                  }
       48         return(ret);
                mr           r3,r31             ; r3,ret
       49       }
                lwz          r31,0x1C(r1)       ; ret,28(r1)
                addi         r1,r1,0x20         ; r1,r1,32
                blr
                cmpwi        r3,0x0             ; r3,0
                bne          0x4000121C         ; 0x4000121C (-)
                             {
```

Trace Searching

The trace data can be searched using the commands **Trace.Find** and **Trace.FindAll**. Clicking the 'Find' button in any trace view window will open a user interface to provide easy access to these functions. It looks like this.

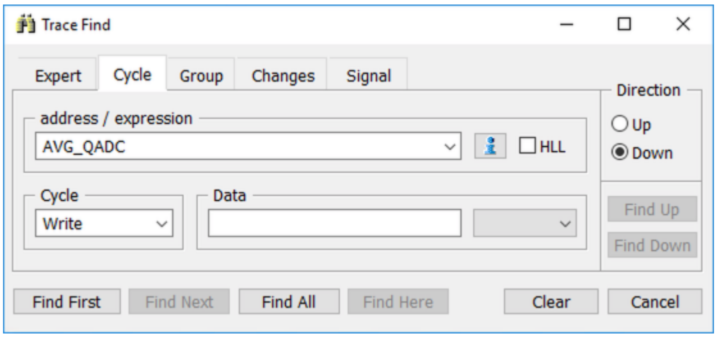


Locate all Writes to a Variable

To locate all writes to a variable:

1. Set the Address/Expression field to the variable name (AVG_QADC).

2. Set the **Cycle** drop-down list to **Write**.



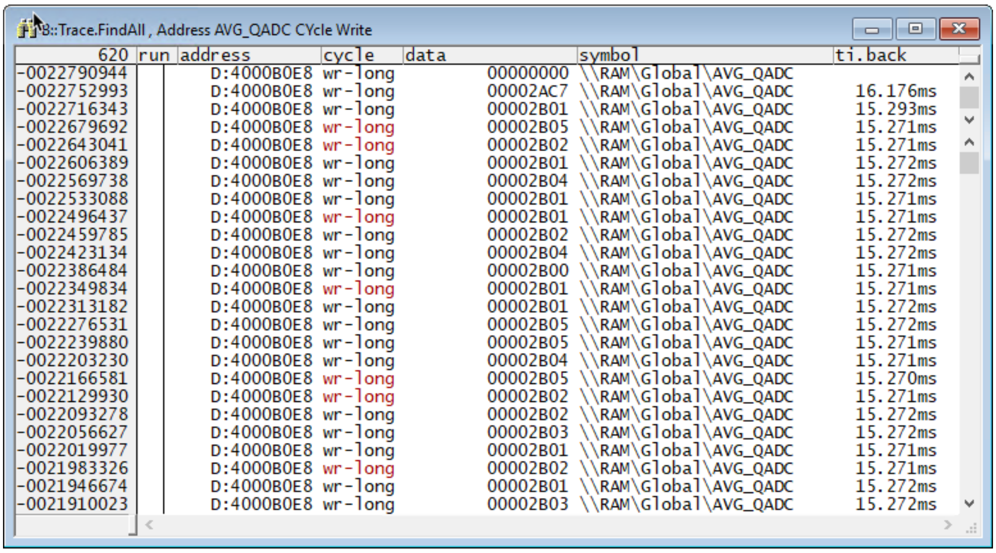
address / expression: AVG_QADC

Cycle: Write

Data:

Direction: Up [✓ Down]

3. Click **Find All** to view the results.



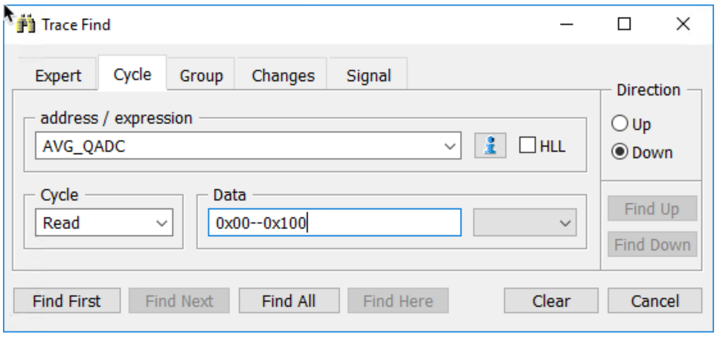
```
B::Trace.FindAll , Address AVG_QADC CYcle Write
        620 run address        cycle   data           symbol                  ti.back
-0022790944         D:4000B0E8 wr-long      00000000 \\RAM\Global\AVG_QADC
-0022752993         D:4000B0E8 wr-long      00002AC7 \\RAM\Global\AVG_QADC   16.176ms
-0022716343         D:4000B0E8 wr-long      00002B01 \\RAM\Global\AVG_QADC   15.293ms
-0022679692         D:4000B0E8 wr-long      00002B05 \\RAM\Global\AVG_QADC   15.271ms
-0022643041         D:4000B0E8 wr-long      00002B02 \\RAM\Global\AVG_QADC   15.271ms
-0022606389         D:4000B0E8 wr-long      00002B01 \\RAM\Global\AVG_QADC   15.272ms
-0022569738         D:4000B0E8 wr-long      00002B04 \\RAM\Global\AVG_QADC   15.272ms
-0022533088         D:4000B0E8 wr-long      00002B01 \\RAM\Global\AVG_QADC   15.271ms
-0022496437         D:4000B0E8 wr-long      00002B01 \\RAM\Global\AVG_QADC   15.271ms
-0022459785         D:4000B0E8 wr-long      00002B02 \\RAM\Global\AVG_QADC   15.272ms
-0022423134         D:4000B0E8 wr-long      00002B04 \\RAM\Global\AVG_QADC   15.272ms
-0022386484         D:4000B0E8 wr-long      00002B00 \\RAM\Global\AVG_QADC   15.271ms
-0022349834         D:4000B0E8 wr-long      00002B01 \\RAM\Global\AVG_QADC   15.271ms
-0022313182         D:4000B0E8 wr-long      00002B01 \\RAM\Global\AVG_QADC   15.272ms
-0022276531         D:4000B0E8 wr-long      00002B05 \\RAM\Global\AVG_QADC   15.272ms
-0022239880         D:4000B0E8 wr-long      00002B05 \\RAM\Global\AVG_QADC   15.271ms
-0022203230         D:4000B0E8 wr-long      00002B04 \\RAM\Global\AVG_QADC   15.271ms
-0022166581         D:4000B0E8 wr-long      00002B05 \\RAM\Global\AVG_QADC   15.270ms
-0022129930         D:4000B0E8 wr-long      00002B02 \\RAM\Global\AVG_QADC   15.271ms
-0022093278         D:4000B0E8 wr-long      00002B02 \\RAM\Global\AVG_QADC   15.272ms
-0022056627         D:4000B0E8 wr-long      00002B03 \\RAM\Global\AVG_QADC   15.272ms
-0022019977         D:4000B0E8 wr-long      00002B01 \\RAM\Global\AVG_QADC   15.271ms
-0021983326         D:4000B0E8 wr-long      00002B02 \\RAM\Global\AVG_QADC   15.271ms
-0021946674         D:4000B0E8 wr-long      00002B01 \\RAM\Global\AVG_QADC   15.272ms
-0021910023         D:4000B0E8 wr-long      00002B03 \\RAM\Global\AVG_QADC   15.272ms
```

Selecting any of the items in this display will cause any open **Trace.List** window to synchronize to the same point in time.

Locate Dedicated Reads to a Variable

To locate all reads from a variable where the value is less than 0x100:

1. Set the Address/Expression field to the variable name (AVG_QADC).
2. Set the **Cycle** drop-down list to **Read**.
3. Set the data value to the required range (0x00--0x100). Ranges are supported but greater than and less than are not.


Trace Find

address / expression: AVG_QADC

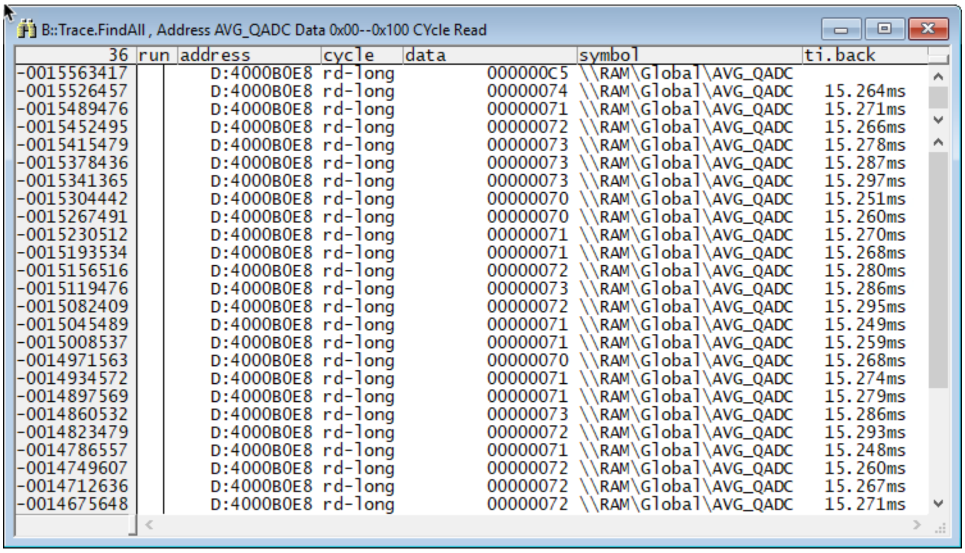
HLL: ☐

Cycle: Read

Data: 0x00--0x100

Direction: Up [✓ Down]

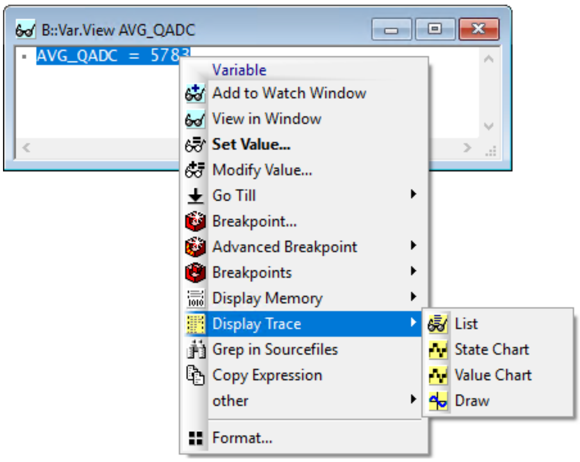
4. Click **Find All** to view the results.


```
B::Trace.FindAll , Address AVG_QADC Data 0x00--0x100 CYcle Read
          36 run address          cycle   data              symbol                  ti.back
-0015563417        D:4000B0E8 rd-long          000000C5 \\RAM\Global\AVG_QADC
-0015526457        D:4000B0E8 rd-long          00000074 \\RAM\Global\AVG_QADC   15.264ms
-0015489476        D:4000B0E8 rd-long          00000071 \\RAM\Global\AVG_QADC   15.271ms
-0015452495        D:4000B0E8 rd-long          00000072 \\RAM\Global\AVG_QADC   15.266ms
-0015415479        D:4000B0E8 rd-long          00000073 \\RAM\Global\AVG_QADC   15.278ms
-0015378436        D:4000B0E8 rd-long          00000073 \\RAM\Global\AVG_QADC   15.287ms
-0015341365        D:4000B0E8 rd-long          00000073 \\RAM\Global\AVG_QADC   15.297ms
-0015304442        D:4000B0E8 rd-long          00000070 \\RAM\Global\AVG_QADC   15.251ms
-0015267491        D:4000B0E8 rd-long          00000070 \\RAM\Global\AVG_QADC   15.260ms
-0015230512        D:4000B0E8 rd-long          00000071 \\RAM\Global\AVG_QADC   15.270ms
-0015193534        D:4000B0E8 rd-long          00000071 \\RAM\Global\AVG_QADC   15.268ms
-0015156516        D:4000B0E8 rd-long          00000072 \\RAM\Global\AVG_QADC   15.280ms
-0015119476        D:4000B0E8 rd-long          00000073 \\RAM\Global\AVG_QADC   15.286ms
-0015082409        D:4000B0E8 rd-long          00000072 \\RAM\Global\AVG_QADC   15.295ms
-0015045489        D:4000B0E8 rd-long          00000071 \\RAM\Global\AVG_QADC   15.249ms
-0015008537        D:4000B0E8 rd-long          00000071 \\RAM\Global\AVG_QADC   15.259ms
-0014971563        D:4000B0E8 rd-long          00000070 \\RAM\Global\AVG_QADC   15.268ms
-0014934572        D:4000B0E8 rd-long          00000071 \\RAM\Global\AVG_QADC   15.274ms
-0014897569        D:4000B0E8 rd-long          00000071 \\RAM\Global\AVG_QADC   15.279ms
-0014860532        D:4000B0E8 rd-long          00000073 \\RAM\Global\AVG_QADC   15.286ms
-0014823479        D:4000B0E8 rd-long          00000072 \\RAM\Global\AVG_QADC   15.293ms
-0014786557        D:4000B0E8 rd-long          00000071 \\RAM\Global\AVG_QADC   15.248ms
-0014749607        D:4000B0E8 rd-long          00000072 \\RAM\Global\AVG_QADC   15.260ms
-0014712636        D:4000B0E8 rd-long          00000072 \\RAM\Global\AVG_QADC   15.267ms
-0014675648        D:4000B0E8 rd-long          00000072 \\RAM\Global\AVG_QADC   15.271ms
```

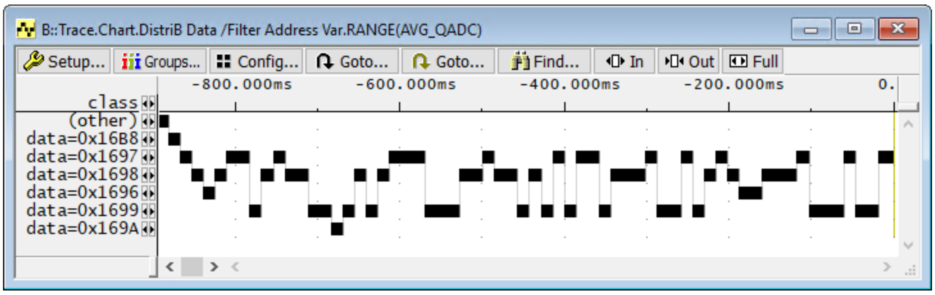
Monitoring a Variable Over Time

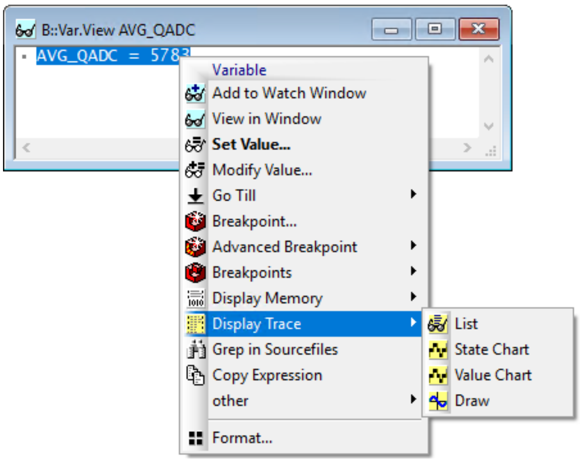
Static Analysis

The values assigned to a variable over the duration of the trace sampling period can be displayed in a number of formats. These will work equally well with a full trace or a filtered trace. Right-clicking a variable in any view window will provide access to the trace analysis options. It will look like this.

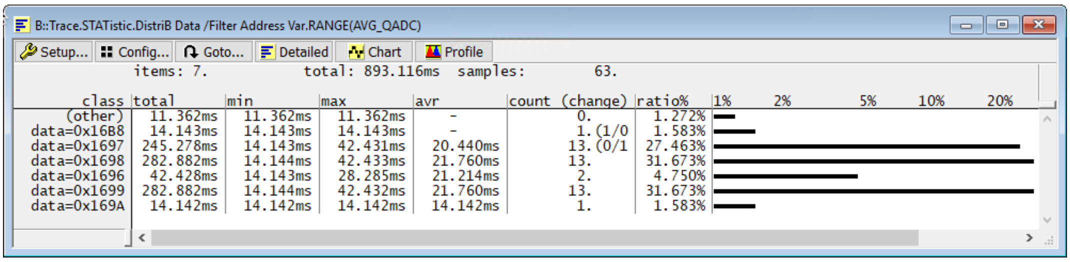


If you select **Value Chart** this opens a window which shows how the variable value changes over time (command Trace.Chart.DistriB). The **`Data`** and **`/Filter`** options narrow down what is to be displayed.





If you select **State Chart** a list of all values that were assigned to a variable can be displayed (command **Trace.STATistic.DistriB**).


B::Trace.STATistic.DistriB Data /Filter Address Var.RANGE(AVG_QADC)
Setup...
Config...
Goto...
Detailed
Chart
Profile

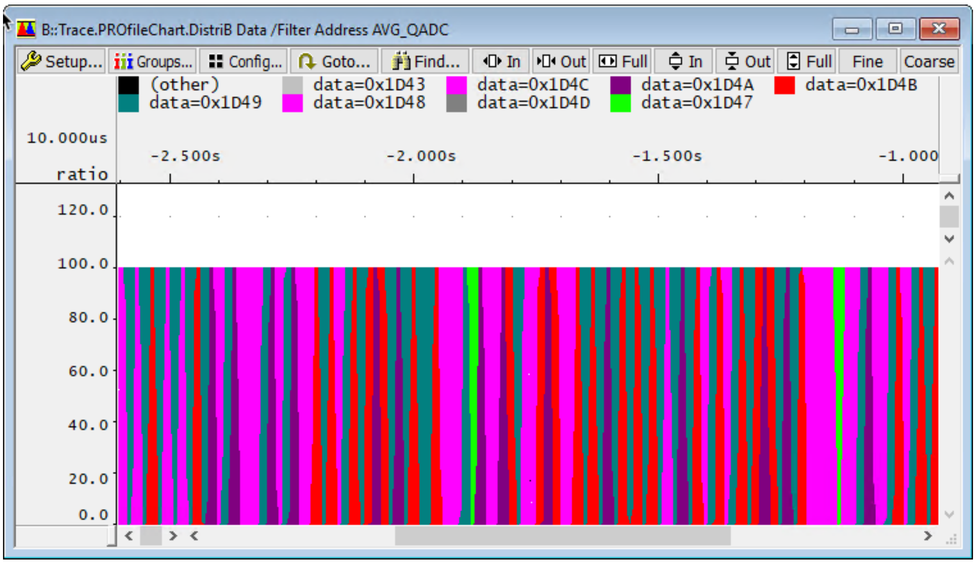
```
            items: 7.            total: 893.116ms  samples:        63.

      class |total      |min        |max        |avr        |count  (change) |ratio%  |1%      2%        5%      10%      20%
    (other) | 11.362ms  | 11.362ms  | 11.362ms  |    -      |     0.         |  1.272%|
data=0x16B8 | 14.143ms  | 14.143ms  | 14.143ms  |    -      |     1. (1/0    |  1.583%|
data=0x1697 |245.278ms  | 14.143ms  | 42.431ms  | 20.440ms  |    13. (0/1    | 27.463%|
data=0x1698 |282.882ms  | 14.144ms  | 42.433ms  | 21.760ms  |    13.         | 31.673%|
data=0x1696 | 42.428ms  | 14.143ms  | 28.285ms  | 21.214ms  |     2.         |  4.750%|
data=0x1699 |282.882ms  | 14.144ms  | 42.432ms  | 21.760ms  |    13.         | 31.673%|
data=0x169A | 14.142ms  | 14.142ms  | 14.142ms  | 14.142ms  |     1.         |  1.583%|
```

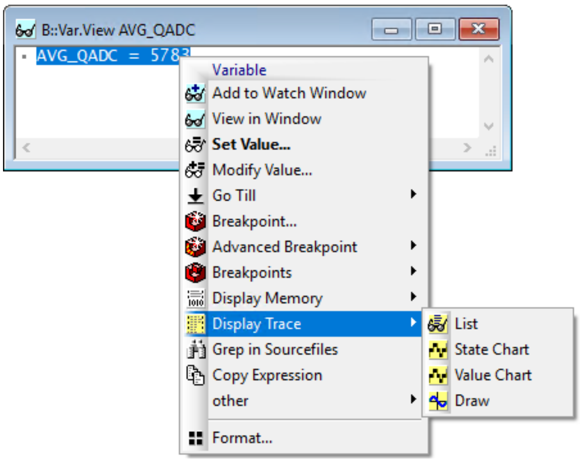
The columns shown are

class	The data value assigned to the address that is being monitored.
total	The total time for which the address held this value.
min	The minimum length of time that this value was held at this address.
max	The maximum length of time that the variable held this value.
avr	The mean length of time that the variable held this value.
count	The number of times this value was assigned to the variable.
ratio	The total time that this value was held as a ratio of the total time sampled.
	A bar graph to show the ratio value graphically.

Clicking the ‘Config’ button allows the user to add or remove columns from the display.

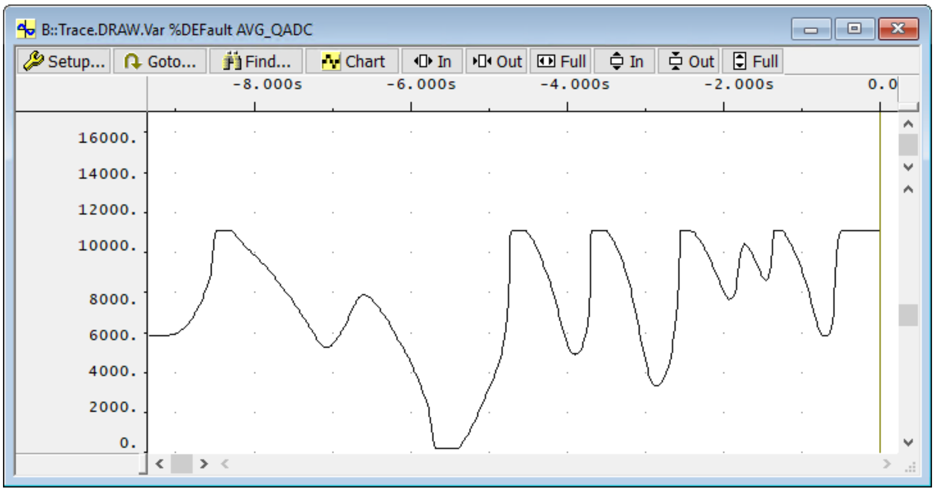
Clicking the 'Profile' button in the **Trace.STATistic.DistriB** window uses the command **Trace.PROfileChart.DistriB** to show a graphical representation of the values as a percentage of a timeslice. The width of the timeslice is dependent upon the scale of the x axis.





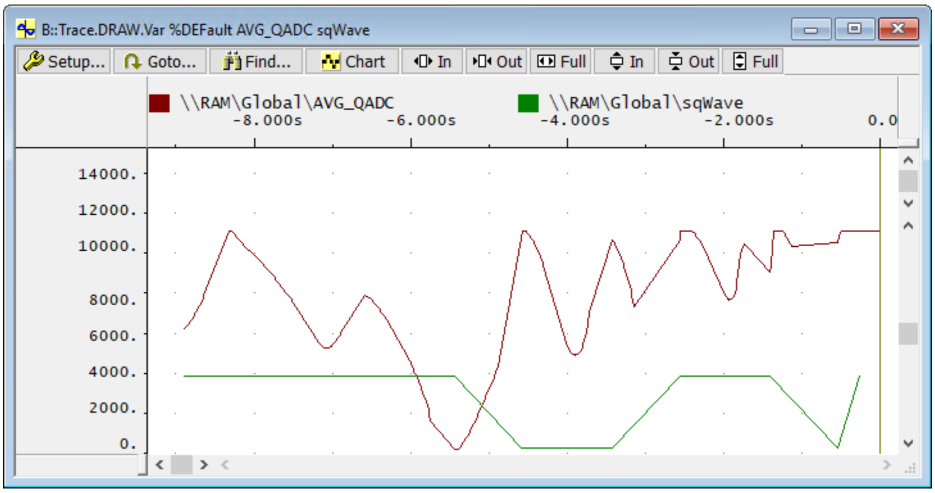
If you select **Draw** (Trace.DRAW.Var) the value of a variable is plotted against time. The display of **Trace.DRAW.Var** refers to the variable type by default. Adding **`%DEFault`** causes TRACE32 to treat all following arguments as variable names.

```
Trace.DRAW.Var %DEFault AVG_QADC
```



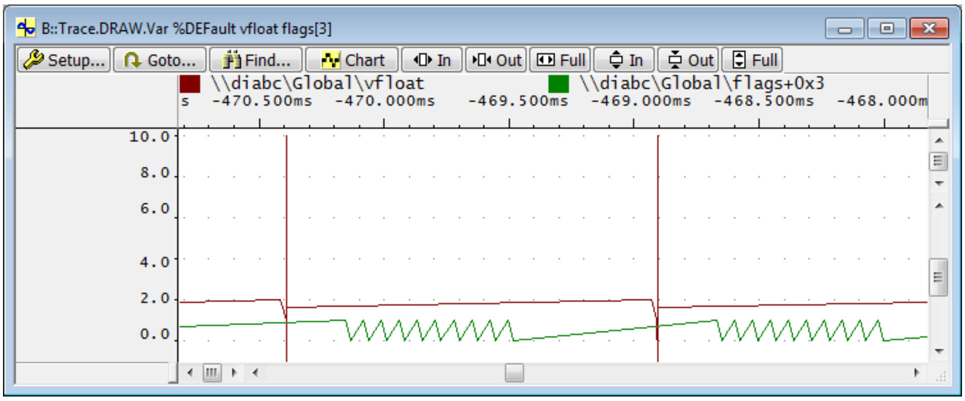
Trace.DRAW.Var can draw multiple variables in the same window. Each line will be colored differently:

```
Trace.DRAW.Var %DEFault AVG_QADC sqWave
```



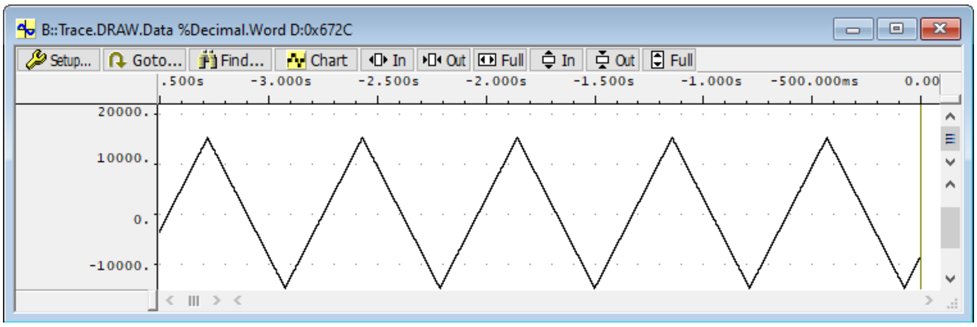
Please be aware of the following:

- It only makes sense to display two or more variables in a window if they have a similar value range.
- If one of the variables is of type float, float scale is used.



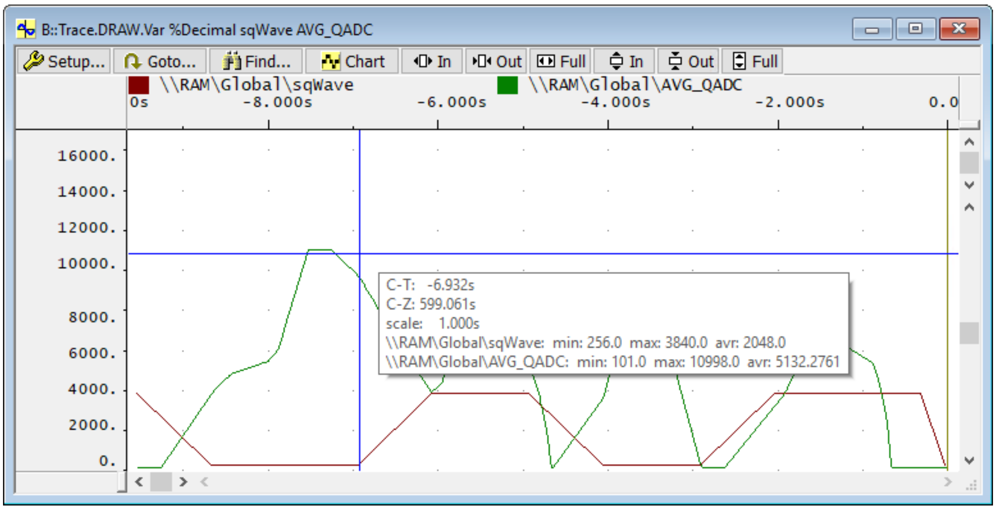
Using the command **Trace.DRAW.Data** for an address, the contents of a memory location can be plotted against time. **Trace.DRAW.Data** requires a definition of the access bus width.

```
; plot a 16-bit value at address 0x672C
Trace.DRAW.Data %Decimal.Word 0x672C

; if no access width is specified, the access width is determined by the
; size of the <data_range>
Trace.DRAW.Data %Decimal 0x672C--0x672D
```



Mouse Support

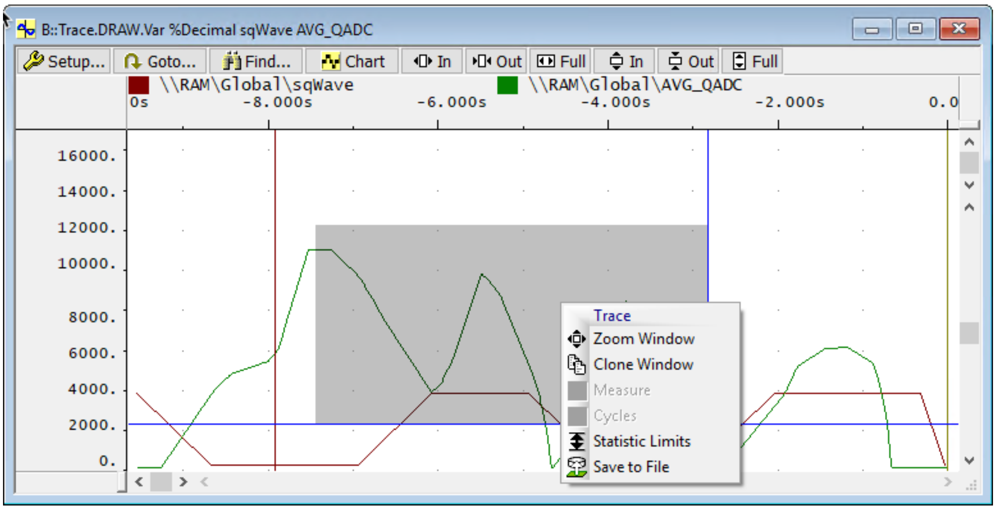
Left-clicking a **Trace.DRAW** window will pop-up a balloon text with more details about the chart.


B::Trace.DRAW.Var %Decimal sqWave AVG_QADC
Setup... Goto... Find... Chart In Out Full In Out Full
\\RAM\Global\sqWave
\\RAM\Global\AVG_QADC
0s -8.000s -6.000s -4.000s -2.000s 0.0
16000. 14000. 12000. 10000. 8000. 6000. 4000. 2000. 0.

```
C-T:   -6.932s
C-Z: 599.061s
scale:   1.000s
\\RAM\Global\sqWave:  min: 256.0  max: 3840.0  avr: 2048.0
\\RAM\Global\AVG_QADC:  min: 101.0  max: 10998.0  avr: 5132.2761
```

C-T	Current time (selected by user) to Trigger time. This is often the end of the trace recording unless a trigger has been programmed.
C-Z	Current time to zero time. This is usually when the debugger started or a user-defined zero point.
scale	The time scale along the x axis.
other	A list of all variables displayed, showing min, max and mean values.

An area of the chart can be selected by left-clicking and dragging. Right-clicking within this area opens a different menu.



- The window can be zoomed to the selected area (Zoom Window).
- The selected area can be opened in a new window (Clone Window).
- The limits for statistical analyses can be set to only include the selected area.
  - The set limits are used for all subsequent **Trace.STATistic** commands (Statistic Limits).
- The trace buffer that covers the selected area can be saved to a file. See **Trace.SAVE** for more information.

Synchronizing Windows

Drag & Drop

The simplest way to synchronize two trace view windows is via the Drag & Drop interface.

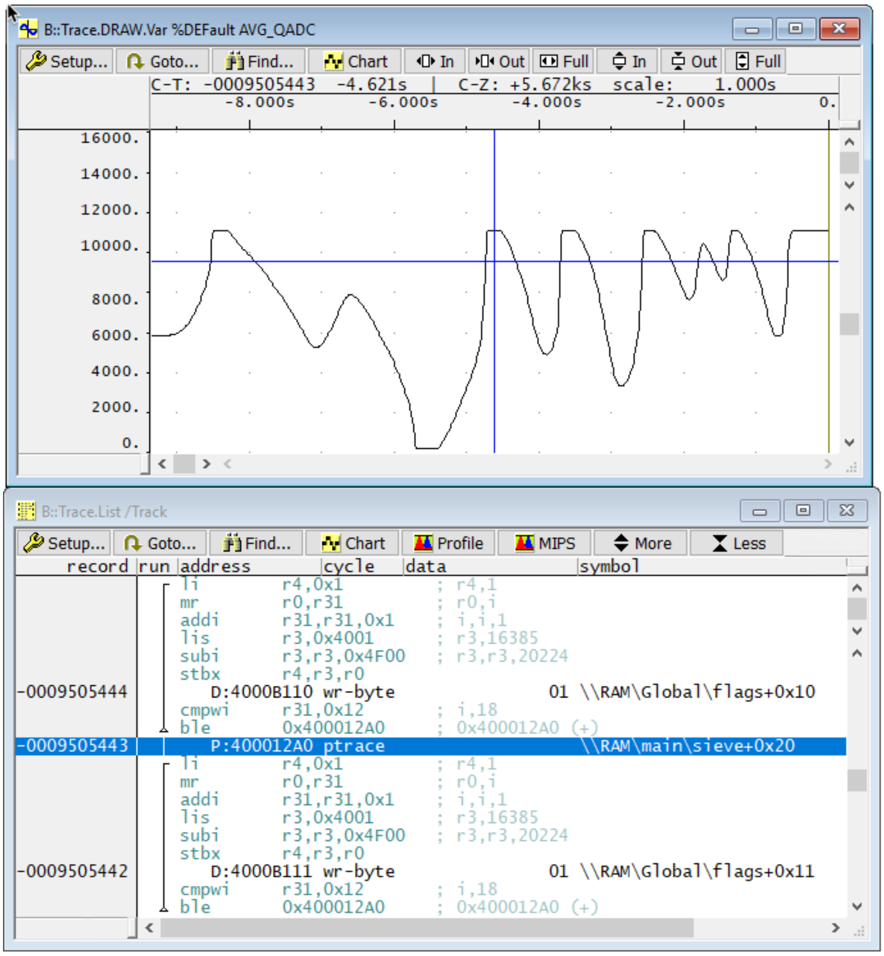
1. Open a **Trace.DRAW.Var** window and a **Trace.List** window.
2. Left-click a point in the **Trace.DRAW.Var** window.
3. Left-click the same point but don't release the mouse button press.
4. Holding the left mouse button, drag the mouse to the **Trace.List** window
5. The view in the **Trace.List** window will snap to the selected point in the other window.

It may take a little practice to get it right but the mouse hotspot is very forgiving for the second click and subsequent drag operation.

Track Option

Any trace view window that is opened with the `/Track` option will automatically jump to synchronize with a user placed cursor in any other trace view window. The tracking is based upon the timestamp information captured as part of the trace data. If no timestamps are available (such as on-chip trace) then the record number is used.

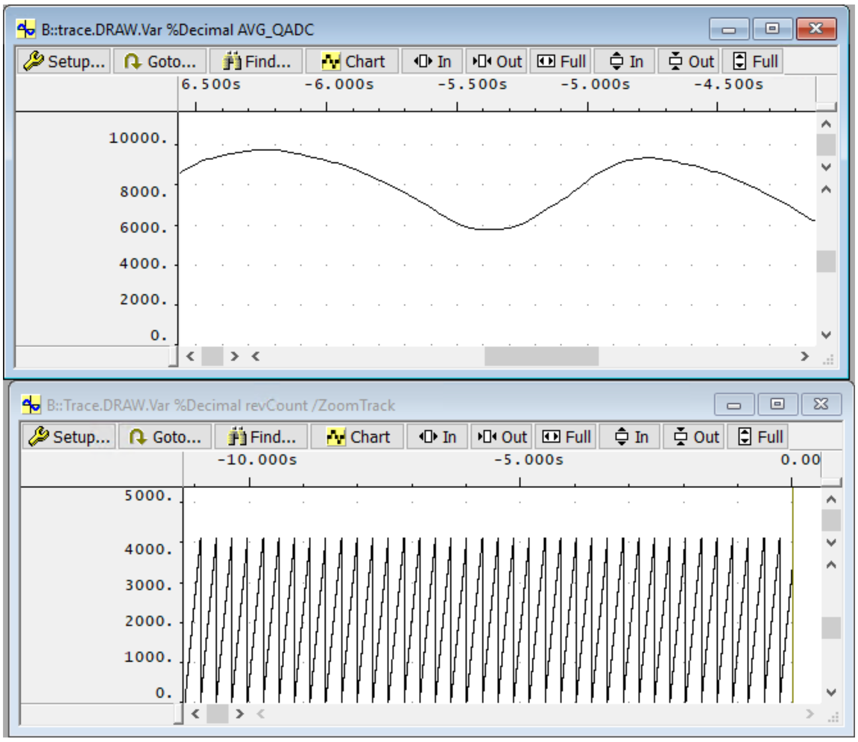
An example can be seen here. The user has left-clicked the **Trace.DRAW.Var** window and the **Trace.List** window has synchronized to that point. The blue cross-hairs indicate where the user has clicked the mouse button in the top window.



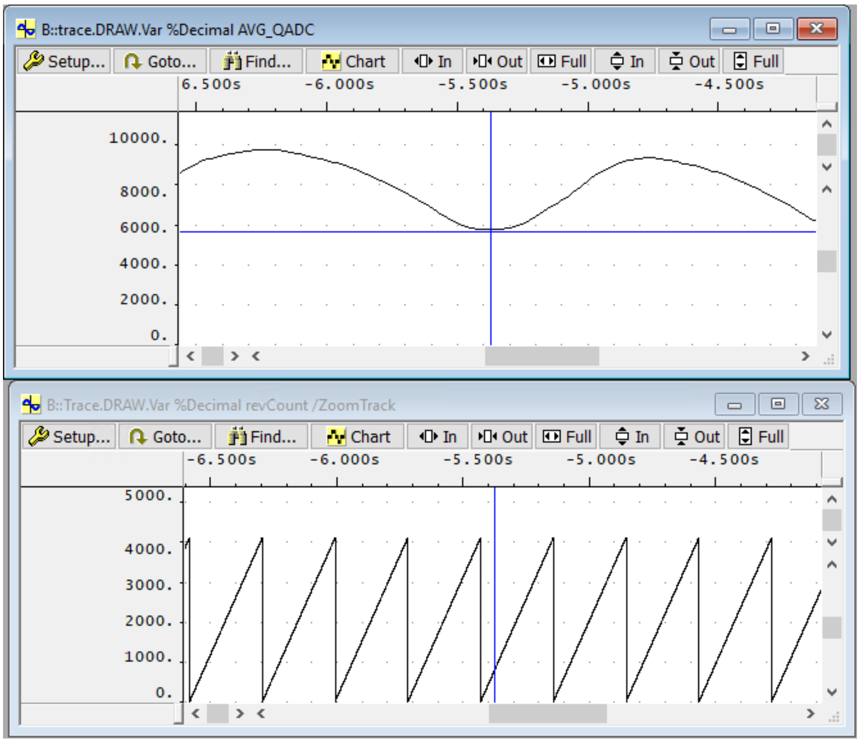
B::Trace.DRAW.Var %DEFault AVG_QADC
Setup... Goto... Find... Chart In Out Full In Out Full
C-T: -0009505443 -4.621s | C-Z: +5.672ks scale: 1.000s
-8.000s -6.000s -4.000s -2.000s 0.
16000. 14000. 12000. 10000. 8000. 6000. 4000. 2000. 0.
B::Trace.List /Track
Setup... Goto... Find... Chart Profile MIPS More Less

```
      record |run |address          |cycle  |data          |symbol
                  li        r4,0x1       ; r4,1
                  mr        r0,r31       ; r0,i
                  addi      r31,r31,0x1  ; i,i,1
                  lis       r3,0x4001    ; r3,16385
                  subi      r3,r3,0x4F00 ; r3,r3,20224
                  stbx      r4,r3,r0
-0009505444          D:4000B110 wr-byte              01 \\RAM\Global\flags+0x10
                  cmpwi     r31,0x12     ; i,18
                  ble       0x400012A0   ; 0x400012A0 (+)
-0009505443          P:400012A0 ptrace                 \\RAM\main\sieve+0x20
                  li        r4,0x1       ; r4,1
                  mr        r0,r31       ; r0,i
                  addi      r31,r31,0x1  ; i,i,1
                  lis       r3,0x4001    ; r3,16385
                  subi      r3,r3,0x4F00 ; r3,r3,20224
                  stbx      r4,r3,r0
-0009505442          D:4000B111 wr-byte              01 \\RAM\Global\flags+0x11
                  cmpwi     r31,0x12     ; i,18
                  ble       0x400012A0   ; 0x400012A0 (+)
```

Windows linked with the **`/ZoomTrack`** option will not only jump to the user selected point but will adjust their scale to match that of the user selected window. Consider the before and after images on this page. In the second set, the scale of the lower window has been automatically adjusted to match that of the upper window.



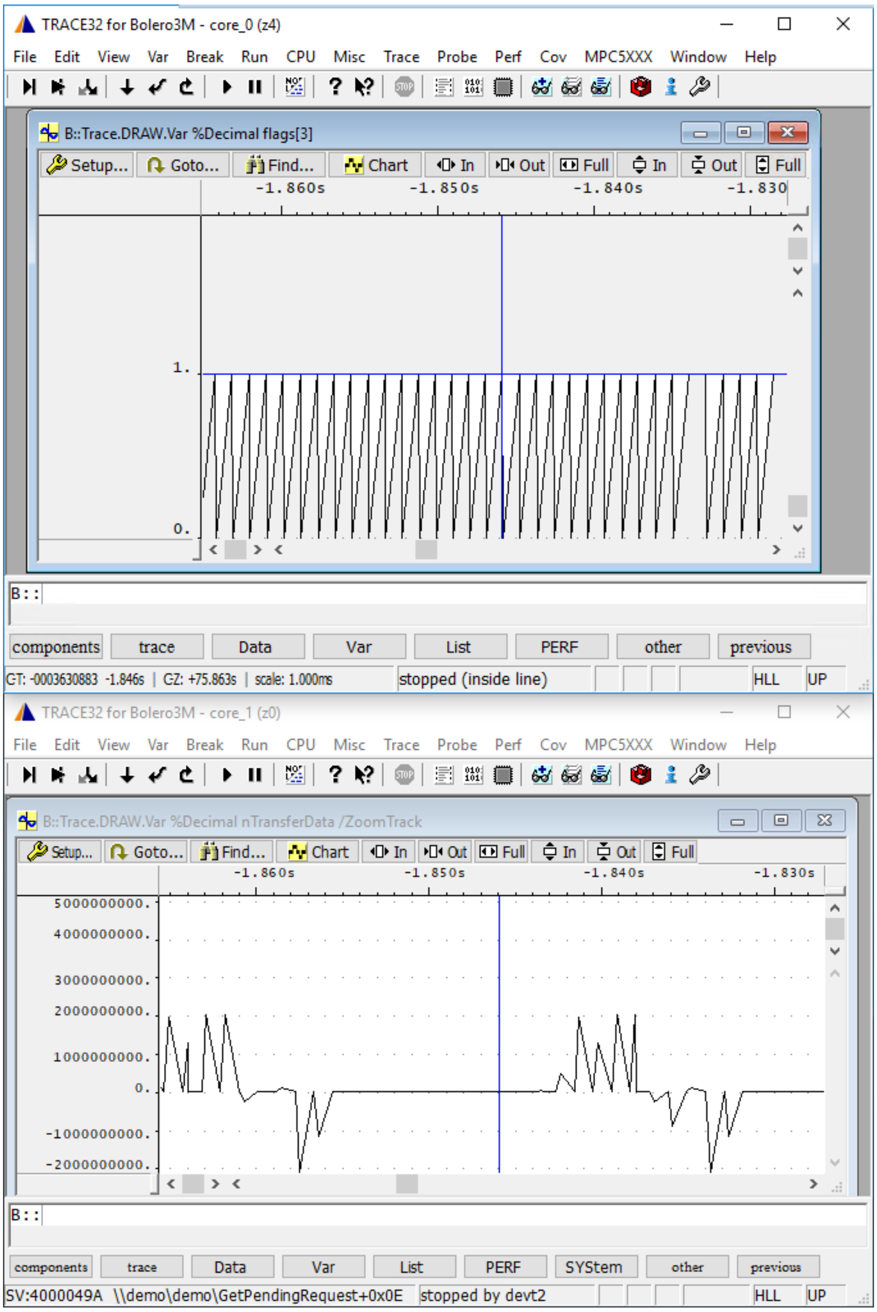
B::trace.DRAW.Var %Decimal AVG_QADC
Setup... Goto... Find... Chart In Out Full In Out Full
6.500s -6.000s -5.500s -5.000s -4.500s
10000. 8000. 6000. 4000. 2000. 0.
B::Trace.DRAW.Var %Decimal revCount /ZoomTrack
-10.000s -5.000s 0.00
5000. 4000. 3000. 2000. 1000. 0.



B::trace.DRAW.Var %Decimal AVG_QADC
Setup... Goto... Find... Chart In Out Full In Out Full
6.500s -6.000s -5.500s -5.000s -4.500s
10000. 8000. 6000. 4000. 2000. 0.
B::Trace.DRAW.Var %Decimal revCount /ZoomTrack
-6.500s -6.000s -5.500s -5.000s -4.500s
5000. 4000. 3000. 2000. 1000. 0.

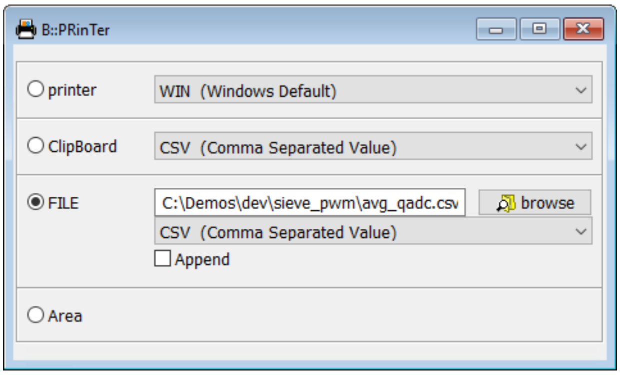
Synchronizing Between TRACE32 Instances

If AMP debugging is configured and the TRACE32 instances are synchronised via **InterCom** then the trace displays can also be synchronized between them. This allows the `/Track` and `/ZoomTrack` options to work across multiple TRACE32 PowerView GUI. The time synchronisation between instances of TRACE32 is configured using the command **SYnch.XTrack**. The image below shows the lower instance of TRACE32 synchronised and zoomed to a user click event in the upper instance of TRACE32.


TRACE32 for Bolero3M - core_0 (z4)
File Edit View Var Break Run CPU Misc Trace Probe Perf Cov MPC5XXX Window Help
B::Trace.DRAW.Var %Decimal flags[3]
Setup... Goto... Find... Chart In Out Full In Out Full
-1.860s -1.850s -1.840s -1.830
1.
0.
B::
components trace Data Var List PERF other previous
GT: -0003630883 -1.846s | GZ: +75.863s | scale: 1.000ms
stopped (inside line)
HLL UP
TRACE32 for Bolero3M - core_1 (z0)
File Edit View Var Break Run CPU Misc Trace Probe Perf Cov MPC5XXX Window Help
B::Trace.DRAW.Var %Decimal nTransferData /ZoomTrack
Setup... Goto... Find... Chart In Out Full In Out Full
-1.860s -1.850s -1.840s -1.830s
5000000000.
4000000000.
3000000000.
2000000000.
1000000000.
0.
-1000000000.
-2000000000.
B::
components trace Data Var List PERF SYStem other previous
SV:4000049A \\demo\demo\GetPendingRequest+0x0E
stopped by devt2
HLL UP

Exporting Data

To export trace data, first configure the TRACE32 printer. This is done using the **PRinTer** command which can be accessed from the 'File' menu. The control window looks like this.



B::PRinTer

○ printer — WIN (Windows Default)

○ ClipBoard — CSV (Comma Separated Value)

● FILE — C:\Demos\dev\sieve_pwm\avg_qadc.csv — browse

CSV (Comma Separated Value)

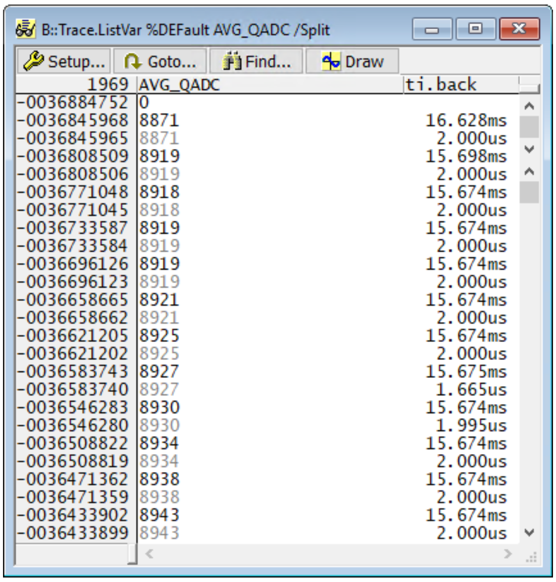
☐ Append

○ Area

The example above has been configured to create a CSV file with the specified name. The command line equivalent would be:

```
PRinTer.FILE "C:/demos/dev/sieve_pwm/avg_qadc.csv" CSV
```

Collect and display the required data using **Trace.ListVar**, **Trace.FindAll** or **Trace.List.** The example below shows trace of both data reads (in grey) and data writes (in black).



B::Trace.ListVar %DEFault AVG_QADC /Split

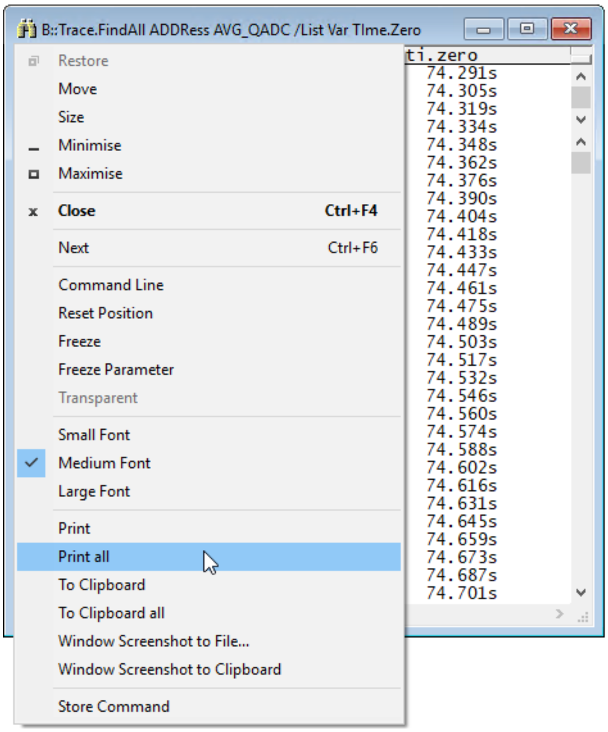
```
        1969 AVG_QADC                  ti.back
-0036884752 0
-0036845968 8871                      16.628ms
-0036845965 8871                       2.000us
-0036808509 8919                      15.698ms
-0036808506 8919                       2.000us
-0036771048 8918                      15.674ms
-0036771045 8918                       2.000us
-0036733587 8919                      15.674ms
-0036733584 8919                       2.000us
-0036696126 8919                      15.674ms
-0036696123 8919                       2.000us
-0036658665 8921                      15.674ms
-0036658662 8921                       2.000us
-0036621205 8925                      15.674ms
-0036621202 8925                       2.000us
-0036583743 8927                      15.675ms
-0036583740 8927                       1.665us
-0036546283 8930                      15.674ms
-0036546280 8930                       1.995us
-0036508822 8934                      15.674ms
-0036508819 8934                       2.000us
-0036471362 8938                      15.674ms
-0036471359 8938                       2.000us
-0036433902 8943                      15.674ms
-0036433899 8943                       2.000us
```

Export the selected data by prefixing the window command with the command **WinPrint** or by left-clicking the menu icon and selecting 'Print' or 'Print all' from the menu. For example:

```
WinPrint.Trace.ListVar %DEFault AVG_QADC /Split
```

or alternatively:

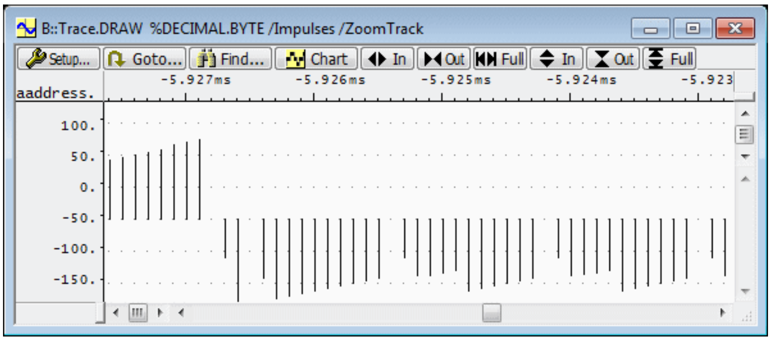
```
WinPrint.Trace.FindAll ADDRess Var.RANGE(AVG_QADC) /List Var TIme.Zero
```

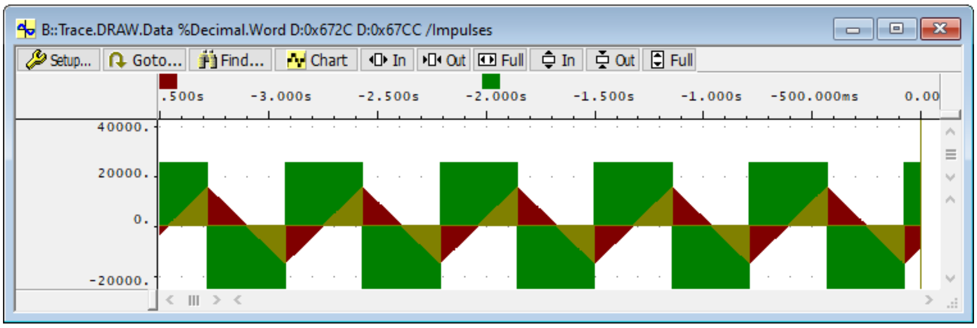


```
ti.zero
 74.291s
 74.305s
 74.319s
 74.334s
 74.348s
 74.362s
 74.376s
 74.390s
 74.404s
 74.418s
 74.433s
 74.447s
 74.461s
 74.475s
 74.489s
 74.503s
 74.517s
 74.532s
 74.546s
 74.560s
 74.574s
 74.588s
 74.602s
 74.616s
 74.631s
 74.645s
 74.659s
 74.673s
 74.687s
 74.701s
```

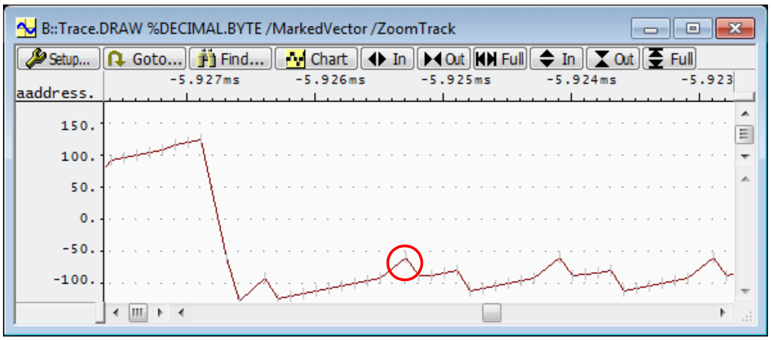
LOG: Displays the data values in a logarithmic format.

Impulses: Draws each data value as a single pulse.

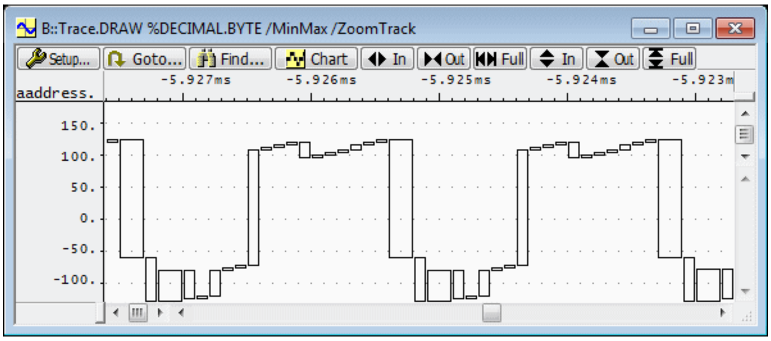

B::Trace.DRAW %DECIMAL.BYTE /Impulses /ZoomTrack


B::Trace.DRAW.Data %Decimal.Word D:0x672C D:0x67CC /Impulses

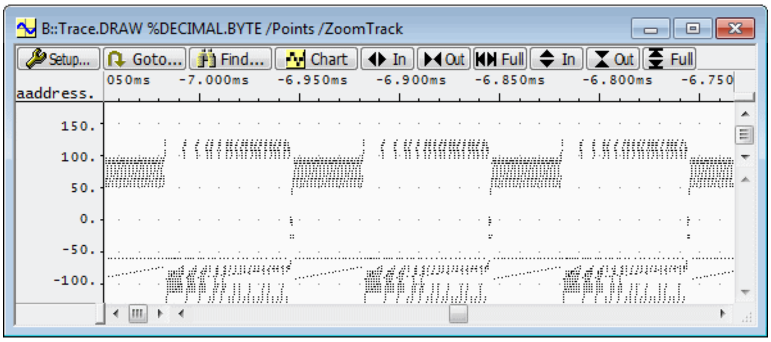
MarkedVector: Highlights the trace records that contain a data value by short, vertical lines.


B::Trace.DRAW %DECIMAL.BYTE /MarkedVector /ZoomTrack

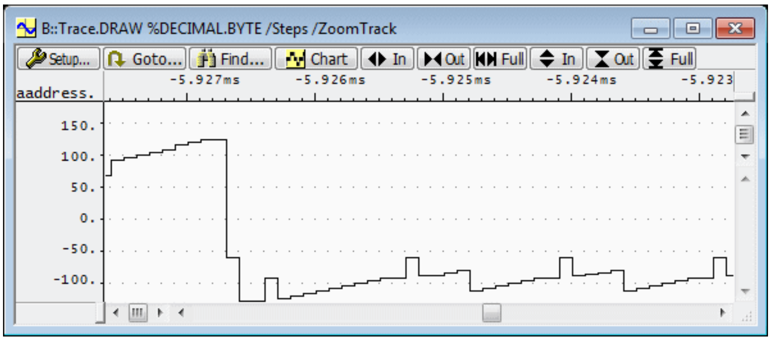
MinMax: Displays the minimum and maximum data value for each range.


B::Trace.DRAW %DECIMAL.BYTE /MinMax /ZoomTrack

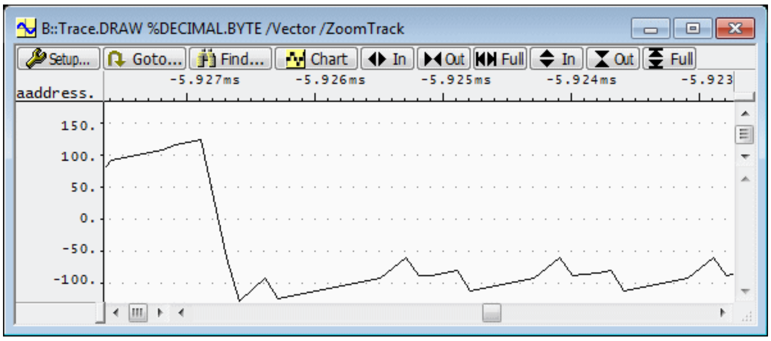
Point: Displays each data value as a dot.


B::Trace.DRAW %DECIMAL.BYTE /Points /ZoomTrack
Setup... Goto... Find... Chart In Out Full In Out Full
050ms -7.000ms -6.950ms -6.900ms -6.850ms -6.800ms -6.750
aaddress.
150.
100.
50.
0.
-50.
-100.

Steps: Connects the dots for the data values by steps.


B::Trace.DRAW %DECIMAL.BYTE /Steps /ZoomTrack
Setup... Goto... Find... Chart In Out Full In Out Full
-5.927ms -5.926ms -5.925ms -5.924ms -5.923
aaddress.
150.
100.
50.
0.
-50.
-100.

Vector: Connects the dots for the data values by vectors (default).


B::Trace.DRAW %DECIMAL.BYTE /Vector /ZoomTrack
Setup... Goto... Find... Chart In Out Full In Out Full
-5.927ms -5.926ms -5.925ms -5.924ms -5.923
aaddress.
150.
100.
50.
0.
-50.
-100.