Measuring function run-times with BenchMark Counters (BMC).

2024-07-08 - Comments (0) - TRACE32 PowerView

BenchMark Counters (performance monitors) can be used to measure function run-times. This is especially useful if no trace is available.

The BenchMark Counters allow measuring executed clock cycles. The run-times can then be calculated if the processor clock is known.

The measurement can be performed without stopping the program execution, if the target architecture supports the BMC ATOB-mode. Otherwise, the execution has to be stopped to start/stop the counter. Search for the command **BMC.<*counter*>.ATOB** in your Processor Architecture Manual, to check if your target architecture supports the ATOB mode. The BMC ATOB-mode is not supported by Arm processors.

1. Measuring function run-times with BMC ATOB-Mode

If the target architecture supports the ATOB mode, then the measurements can be performed without stopping the program execution. This mode enables event triggered counter start/stop.

1.1. Setting Alpha and Beta Breakpoints

The events are defined using **Alpha** and **Beta** breakpoints set with the command **Break.Set** or **Break.SetFunc**. Refer to General Commands Reference Guide B for more information about these commands.

Every time the **Alpha** condition triggers, the counter is started. The counter stops when the **Beta** breakpoint condition is triggered.

Examples:

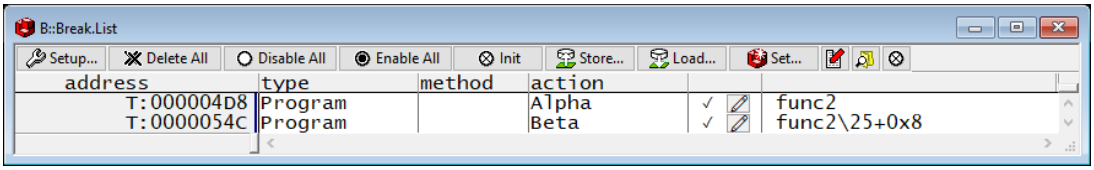
1. Set Alpha and Beta breakpoints at the entry and exit of func2 using **Break.SetFunc**:

```
Break.SetFunc func2
```

1. Set Alpha and Beta breakpoints at the entry and exit of func2 using **Break.Set**:

```
Break.Set func2 /Alpha
Break.Set sYmbol.EXIT(func2) /Beta
```

Both examples will set the following breakpoints:



Note

The **Break.SetFunc** command as well as the **sYmbol.EXIT()** PRACTICE function requires that the function exit is known is unique. If the function has no clear exit point (e.g. because of compiler optimization) or has multiple exit points, then the **Beta** breakpoints need to be set manually.

The time include in the case between the **Alpha** and **Beta** events, i.e. it includes sub-function calls and interrupts. If you are interested in the run-time of the function code only, then you can use the following breakpoints:

```
Var.Break.Set sieve /Alpha
Var.Break.Set sieve /Beta /Exclude
```

Note

Var.Break.Set sets the breakpoint on the whole function range.

The breakpoint with the **/EXCLUDE** option corresponds to the following breakpoints:

```
Break.Set 0--<funcstart>-1 /Beta
Break.Set <funcend>--0xFFFFFFFF /Beta
```

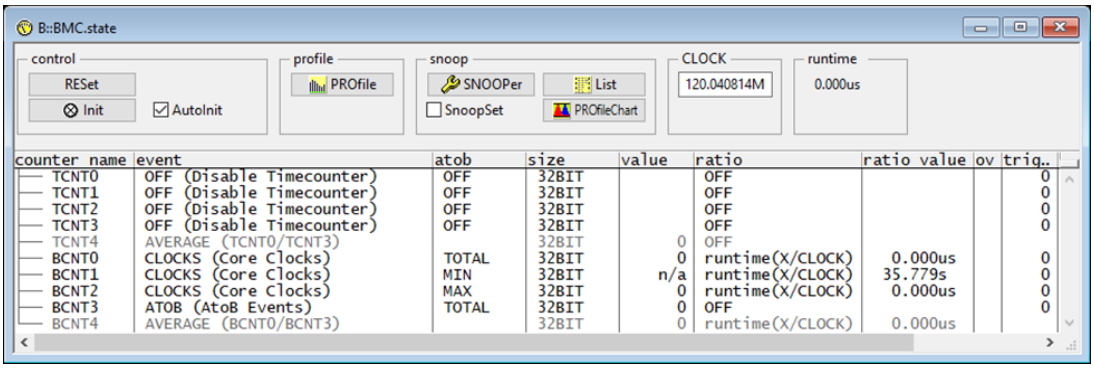
1.2. Enabling the BMC counters

The required settings depend on the target processor. Please refer to the description of the BMC command group in your Processor Architecture Manual and to the examples below for more information.

1.3. Examples

1.3.1. RH850

The ATOB mode allows to measure the total, minimal and maximal run-times as well as the number of function calls based on the **BCNT0..BCNT3** counters:



Additionally, **Alpha** and **Beta** breakpoints needs to be set at entry and exit of the selected function. RH850 supports 1 **Alpha** and 7 **Beta** breakpoints. Multiple **Beta** breakpoint are useful in case of multiple function exit points.

The clock needs additionally to be set using the command **BMC.CLOCK.**

Below is an example script to measure the run-time of the function sieve:

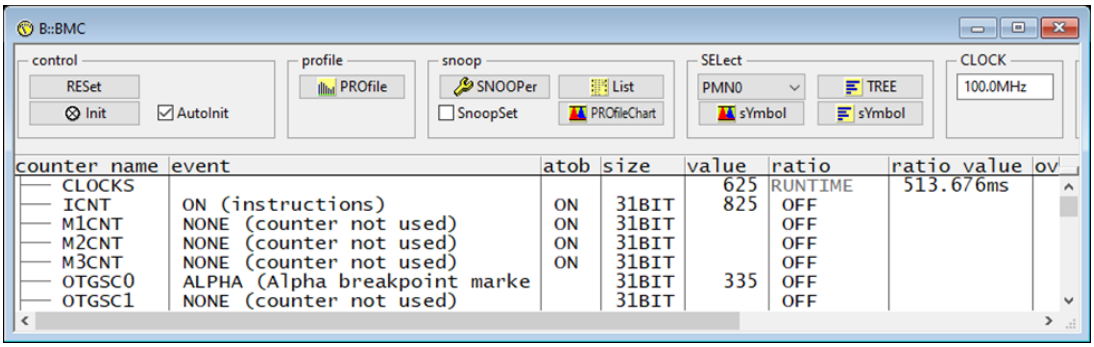
```
BMC.CLOCK 120MHZ ; core clock frequency, e.g. 120 MHz
BMC.Init ON

BMC.BCNT0.EVENT.CLOCKS ; AtoB TotalTime
BMC.BCNT0.ATOB.TOTAL
BMC.BCNT0.RATIO.runtime(X/CLOCK)

BMC.BCNT1.EVENT.CLOCKS ; AtoB MinTime
BMC.BCNT1.ATOB.MIN
BMC.BCNT1.RATIO.runtime(X/CLOCK)

BMC.BCNT2.EVENT.CLOCKS ; AtoB MaxTime
BMC.BCNT2.ATOB.MAX
BMC.BCNT2.RATIO.runtime(X/CLOCK)

BMC.BCNT3.EVENT.ATOB ; AtoB Events
BMC.BCNT3.ATOB.TOTAL
BMC.BCNT3.RATIO.OFF

Break.Delete
;set up counter start / stop events
Break.SetFunc sieve

;run measurement (for 10 seconds)
BMC.Init

Go
```

```
Wait 10s
Break
```

Please refer for more information to the description of the BMC.<counter>.ATOB command in the RH850 Debugger and Trace manual as well as to the Application Note Benchmark Counter RH850.

1.3.2. TriCore

The **ICNT** counter can be used to measure the run-time. The clock can automatically be detected using the **CLOCK.ON** command. **BMC.OTGSC0.EVENT Alpha** can additionally be used to count the number of function calls (Alpha events).



Additionally, Alpha and Beta breakpoints needs to be set at entry and exit of the selected function.

This allows to measure the total and average run-time. The TriCore BenchMark Counters do not support getting the minimum and maximum run-times.

Below is an example script to measure the run-time of the function sieve:

```
CLOCK.ON
BMC.Init ON
BMC.ICNT.EVENT ON
BMC.ICNT.ATOB ON

Break.Delete
;set up counter start / stop events
Break.SetFunc sieve

;run measurement (for 10 seconds)
BMC.Init

Go
Wait 10s
Break
```

Refer also to the demo scripts in the TRACE32 installation under **demo/tricore/etc/bmc**

2. Measuring function run-times without BMC ATOB-Mode

Some processor architectures do not support BMC ATOB-mode (e.g. Arm). In this case the program execution needs to be stopped at the function entry and exit. Example:

```
Go <function_entry>
<set up counter>
Go <function_exit>
<check counter results>
```

RunTime.Mode BMC

For Arm, TriCore and Xtensa, the procedure described above can be automated using the **RunTime** command group and **SPOT** breakpoints.

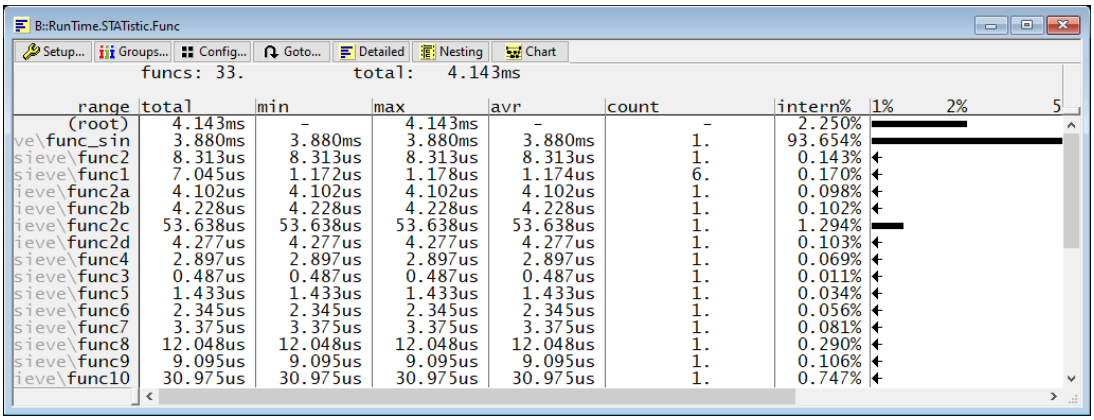
Note

SPOT breakpoints allow to stop the program execution shortly to update the TRACE32 screen when the breakpoint is hit. As soon as the screen is updated, the program execution continues.

Example:

```
; Set SPOT breakpoint at the entry of all functions starting with
func*
sYmbol.ForEach "Break.SetFunc * /SPOT" func*

; Set the BMC clock, e.g. here 600Mhz
BMC.CLOCK 600Mhz

; Set the RunTime mode to BMC
RunTime.Mode BMC

; Prepare RunTime recording
RunTime.OFF

; Run the program execution, e.g. for three seconds
Go
WAIT 3.s
Break
```

The results can then be displayed with the **RunTime.STATistic** command group, e.g. **RunTime.STATistic.Func**


B::RunTime.STATistic.Func
Setup...
Groups...
Config...
Goto...
Detailed
Nesting
Chart
funcs: 33.
total: 4.143ms
range
total
min
max
avr
count
intern%
1%
2%
5
(root) 4.143ms - 4.143ms - - 2.250%
ve\func_sin 3.880ms 3.880ms 3.880ms 3.880ms 1. 93.654%
sieve\func2 8.313us 8.313us 8.313us 8.313us 1. 0.143%
sieve\func1 7.045us 1.172us 1.178us 1.174us 6. 0.170%
ieve\func2a 4.102us 4.102us 4.102us 4.102us 1. 0.098%
ieve\func2b 4.228us 4.228us 4.228us 4.228us 1. 0.102%
ieve\func2c 53.638us 53.638us 53.638us 53.638us 1. 1.294%
ieve\func2d 4.277us 4.277us 4.277us 4.277us 1. 0.103%
sieve\func4 2.897us 2.897us 2.897us 2.897us 1. 0.069%
sieve\func3 0.487us 0.487us 0.487us 0.487us 1. 0.011%
sieve\func5 1.433us 1.433us 1.433us 1.433us 1. 0.034%
sieve\func6 2.345us 2.345us 2.345us 2.345us 1. 0.056%
sieve\func7 3.375us 3.375us 3.375us 3.375us 1. 0.081%
sieve\func8 12.048us 12.048us 12.048us 12.048us 1. 0.290%
sieve\func9 9.095us 9.095us 9.095us 9.095us 1. 0.106%
ieve\func10 30.975us 30.975us 30.975us 30.975us 1. 0.747%