LAUTERBACH
DEVELOPMENT TOOLS

Hexagon-ETM Trace

Hexagon-ETM Trace

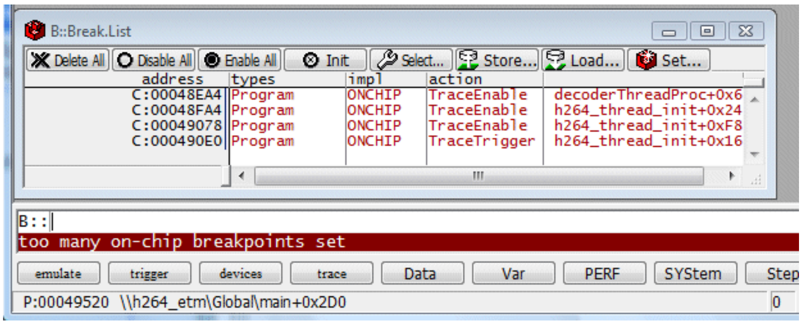
Controlling the ETM

The ETM triggering and filtering can be controlled in two ways:

- By the setting in the **ETM.state** window combined with the breakpoints.
- By the **ETM.Set** commands.

The user gets exclusive control over the ETM trigger blocks, sequencer levels etc. by using the **ETM.Set** commands. The settings within the **ETM.state** window and breakpoints are only programmed to unused ETM resources.

If not enough ETM resource are left, to program the settings within the **ETM.state** window and/or the breakpoints, the following error message is displayed.



```
          address  types    impl    action        
C:00048EA4 Program  ONCHIP  TraceEnable   decoderThreadProc+0x6
C:00048FA4 Program  ONCHIP  TraceEnable   h264_thread_init+0x24
C:00049078 Program  ONCHIP  TraceEnable   h264_thread_init+0xF8
C:000490E0 Program  ONCHIP  TraceTrigger  h264_thread_init+0x16
```

```
B::
too many on-chip breakpoints set
```

```
P:00049520  \\h264_etm\Global\main+0x2D0    0
```

ETM.state/Breakpoints

The following settings in the **ETM.state** window can be used to control the ETM triggering and filtering:

ETM.ContextID ON	Broadcast TID/ASID together with the instruction flow.
ETM.TraceTNUM *<tnum>*	Broadcast only instructions executed by the specified hardware thread.
ETM.TraceASID *<asid>*	Broadcast only instructions executed within the address space of the specified ASID.
ETM.TraceTID *<tid>* | *<bitmask>*	Broadcast only instructions executed by the specified software thread.

The following breakpoints use ETM ressources:

; Stop the program execution after the specified instruction was executed a specified number of times
; (up to 4 to single instruction addresses, up to 4 instruction address ranges)

Break.Set *<address>* | *<range>* **/Program /Onchip /COUNT** *<number>*

Stop the program execution after the instruction at address 0x4dd84 was executed 20. times.

```
Break.Set 0x4dd84 /Program /Onchip /COUNT 20.

…

; delete breakpoint
Break.Delete
```

; Set memory access breakpoint, data value possible
; (up to 4 accesses to single addresses, up to 2 accesses to address ranges)

Break.Set *<address>* | *<range>* **/ReadWrite** | **/Read** | **/Write**

Break.Set *<address>* | *<range>* **/***<access>* **/Data.auto** *<data>* | **/Data.Byte** *<data>*

Break.Set *<address>* | *<range>* **/***<access>* **/Data.Word** *<data>* | **/Data.Long** *<data>*

Stop the program execution when a write access to address 0x37FFFB0 occurs. Broadcast the complete instruction flow.

```
Break.Set 0x37FFFB0 /Write

…

Break.Delete                                    ; delete breakpoint
```

Stop the program execution when the byte 0x8c is written to the address 0x37FFFB0. Broadcast the complete instruction flow.

```
Break.Set 0x37FFFB0 /WRite /Data.Byte 0xc8

…

Break.Delete                                   ; delete breakpoint
```

Stop the program execution when an instruction executed by hardware thread 2 writes to the address 0x37FFFB0. Broadcast only the instructions executed by hardware thread 2.

```
ETM.TraceTNUM 2.                     ; broadcast only instructions
                                     ; executed by hardware thread 2

Break.Set 0x37FFFB0 /Write           ; stop program execution when an
                                     ; instruction executed on hardware
                                     ; thread 2 writes to address
                                     ; 0x37FFFB0

…

Break.Delete                         ; delete breakpoint

ETM.TraceTNUM                        ; remove hardware thread filter
```

Stop the program execution when an instruction executed by hardware thread 2. writes to the address 0x37FFFB0. Broadcast the complete instruction flow.

```
ETM.Set Address T0 Write 0x37FFFB0   ; program trigger block T0 for
                                     ; write access to address
ETM.Set TNUM T0 2.                   ; 0x37FFFB0 by hardware thread 2.

ETM.Set S0TO1 T0                     ; change sequencer level from S0
                                     ; to S1 when T0 matches

ETM.Set STOP S1                      ; stop program execution when
                                     ; sequencer is in level S1

…

ETM.CLEAR                            ; undo ETM.Set settings
```

; Set memory access breakpoint, data value possible, one counter
(up to 1)

Break.Set *<address>* | *<range>* **/***<access> <data_def>* **/COUNT** *<number>*
Var.Break.Set *<hll_expression>* **/***<access> <data>* **/COUNT** *<number>*

Stop the program execution after the byte 0x8c is written to the address 0x37FFFB0 10 times

```
Break.Set 0x37FFFB0 /Write /Data.Byte 0xc8 /COUNT 10.

…

; delete breakpoint
Break.Delete
```

; Broadcast only the execution of the specified instructions
; (up to 8 single instructions or up to 4 instruction ranges)

Break.Set *<address>* | *<range>* **/Program /TraceEnable**

; Broadcast only the instructions that perform the specified data access
; not data valueallowed
; (up to 6 single address accesses or up to 3 access ranges)

Break.Set *<address>* | *<range>* **/ReadWrite** | **/Read** | **/Write /TraceEnable**
Var.Break.Set *<hll_expression>* **/ReadWrite** | **/Read** | **/Write /TraceEnable**

Broadcast only the execution of the instruction at address 0x4dd84.

```
Break.Set 0x4dd84 /Program /TraceEnable

…

MergedAnalyzer.List                              ; display the result

…

Break.Delete                                     ; delete breakpoint
```

Broadcast only the execution of the instruction at address 0x4dd84 in the software thread 0x11.

```
ETM.ContextID ON                                        ; broadcast TID together
                                                        ; with instruction flow

ETM.TraceTID 0x11                                       ; broadcast only the
                                                        ; execution of the
Break.Set 0x4dd84 /Program /TraceEnable                 ; instruction at the
                                                        ; address 0x4dd84 in the
                                                        ; software thread 0x11

…

MergedAnalyzer.List                                     ; display the result

…

Break.Delete                                            ; delete breakpoint

ETM.TraceTID                                            ; remove TID filter

ETM.ContextID OFF                                       ; broadcast instruction
                                                        ; flow without TID
```

; Broadcast only the execution of the instructions between TraceON/TraceOFF
; (up to 2 pairs)

Break.Set *<address>* | *<range>* **/Program /TraceON**
Break.Set *<address>* | *<range>* **/ReadWrite** | **/Read** | **/Write /TraceON**
Var.Break.Set *<hll_expression>* **/ReadWrite** | **/Read** | **/Write /TraceON**

Break.Set *<address>* | *<range>* **/Program /TraceOFF**
Break.Set *<address>* | *<range>* **/ReadWrite** | **/Read** | **/Write /TraceOFF**
Var.Break.Set *<hll_expression>* **/ReadWrite** | **/Read** | **/Write /TraceOFF**

Start broadcasting the instruction flow after the instruction at the address 0x4dd84 was executed. Stop the program execution when the trace buffer is full.

```
Trace.Mode Leash                            ; switch the trace to Leash
                                            ; mode

Break.Set 0x4dd84 /Program /TraceON         ; start the broadcasting of
                                            ; of the instruction flow
                                            ; after the instruction at
                                            ; address 0x4dd84 was executed

…

Break.Delete                                ; delete breakpoint

Trace.Mode Fifo                             ; switch trace back to its
                                            ; default mode
```

Start broadcasting the instruction flow after the instruction at the address 0x4dd84 was executed by the hardware thread 3. Stop broadcasting the instruction flow after the instruction at the address 0x4ff98 was executed by the hardware thread 3. Broadcast only the instruction flow of hardware thread 3.

```
ETM.TraceTNUM 3.                            ; broadcast only instructions
                                            ; executed by hardware thread
                                            ; 3

Break.Set 0x4dd84 /Program /TraceON         ; start broadcasting the
                                            ; instruction flow after
                                            ; the instruction at the
                                            ; address 0x4dd84 was executed
                                            ; by the hardware thread 3

Break.Set 0x4ffa8 /Program /TraceOFF        ; stop broadcasting the
                                            ; instruction flow after
                                            ; the instruction at the
                                            ; address 0x4dd84 was executed
                                            ; by the hardware thread 3

…

3Analyzer.List                              ; list instruction flow
                                            ; broadcasted for hardware
                                            ; thread 3

…

Break.Delete                                ; delete breakpoints

ETM.TraceTNUM                               ; broadcast the complete
                                            ; instruction flow
```

; Stop trace recording when the specified address is executed
; (up to 4 single instructions or up to 4 instruction ranges)

Break.Set *<address>* | *<range>* **/Program /TraceTrigger**

; Stop trace recording when the specified data access occurred
; (up 4 single data accesses or up to 2 data access ranges)

Break.Set *<address>* | *<range>* **/Program /TraceTrigger**
Break.Set *<address>* | *<range>* **/ReadWrite** | **/Read** | **/Write /TraceTrigger**
Break.Set *<address>* | *<range>* **/***<access>* **/Data.auto** *<data>* | **/Data.Byte** *<data>* **/TraceTrigger**
Break.Set *<address>* | *<range>* **/***<access>* **/Data.Word** *<data>* | **/Data.Long** *<data>* **/TraceTrigger**

Stop the trace recording after a write access to the address 0x37FFFB0 occurred and another 100 000. records were sampled to the trace buffer. Broadcast the complete instruction flow.

```
Break.Set 0x37FFFB0 /Write /TraceTrigger

MergedAnalyzer.TDelay 100000.                ; specify trigger delay

Go

Break

MergedAnalyzer.List                          ; list the result

MergedAnalyzer.GOTO 0.                       ; move the trace cursor
                                             ; to the trigger point

…

Break.Delete                                 ; delete the breakpoint

MergedAnalyzer.TDelay                        ; clear the trigger delay
```

Stop the trace recording after a write access to the address 0x37FFFB0 by the hardware thread 2. occurred and another 100 000. records were sampled to the trace buffer. Broadcast the complete instruction flow. Broadcast only the instructions executed by the hardware thread 2.

```
ETM.TraceTNUM 2.                                ; specify hardware thread
                                                ; 2. for the breakpoint
                                                ;
                                                ; broadcast only
                                                ; instructions executed
                                                ; by hardware thread 2.

Break.Set 0x37FFFB0 /Write /TraceTrigger

2Analyzer.TDelay 100000.                        ; specify trigger delay


Go

Break

2Analyzer.List                                  ; list the result

2Analyzer.GOTO 0.                               ; move the trace cursor
                                                ; to the trigger point

…

Break.Delete                                    ; delete the breakpoint

2Analyzer.TDelay                                ; clear the trigger delay

ETM.TraceTNUM                                   ; clear the TraceTNUM
                                                : value
```

Commands

ETM — Embedded Trace Macrocell

ETM.ATBSIZE — ATB bus size

Format:	**ETM.ATBSIZE 32** | **64**

Default: 64.

Sets ATB bus size.

ETM.BBC — Broadcast all branches

Format:	**ETM.BBC** [**ON** | **OFF**]

OFF (default)	The ETM broadcasts only the address information when the processor branches to a location that cannot be directly inferred from the source code.
ON	The ETM broadcasts the address information for all branches or jumps.

ETM.CLEAR — Clear trace and sequencer settings

Format:	**ETM.CLEAR**

Switches the ETM ON, clears the trace and clears all setting done by the command **ETM.Set**.

Format: **ETM.CLOCK** *<frequency>*
(alias for **<trace>.CLOCK**)

Tells the debugger the core clock frequency of the traced core.

Format: **ETM.ContextID** [**ON** | **OFF**]

Default: OFF

Provide the TID in the synchronisation packets if ON. Precondition is that the RTOS operates the TID.

Since the ETM broadcasts only instruction trace information, **ETM.ContextID** has to be switched to ON if an RTOS is used and command like **Trace.STATistic.Func**, **TraceSTATistic.TaskFunc** are used.

```
…
TASK.CONFIG rtoslinux
…
ETM.ContextID ON
…
Trace.STATistic.TaskFunc
```

ETM.CycleAccurate Cycle accurate tracing

Format:	**ETM.CycleAccurate** [**ON** | **OFF**]

OFF (default)	The ETM broadcasts only the information which instructions were executed. Timestamps are generated by TRACE32.
ON	The ETM broadcasts the information which instructions were executed, but additionally information about the tread stalls. No timestamps are generated by TRACE32.

Cycle accurate tracing can be used to observe the exact number of cycles that a particular code sequence takes to execute. If cycle accurate tracing is used, trace information is generated for each clock cycle. In this case the *<core_clock>* can be used to calculate the timestamps for the trace information.

```
ETM.CycleAccurate ON

ETM.FillPort ON                       ; compress trace information

Trace.CLOCK 500.MHz                   ; specify the <core_clock> as
                                      ; base for the trace timestamps

Trace.List                            ; display the trace information
```

ETM.CycleAccurateShift Shift cycle accurate timestamps

Format:	**ETM.CycleAccurate** [**ON** | **OFF**]

Shifts cycle accurate timestamps by one instruction.

ETM.CycleCoarse Use coarse cycle accurate mode

Format:	**ETM.CycleCoarse** [**ON** | **OFF**]

Uses coarse cycle accurate mode.

ETM.CallReturnOnly Trace only functions calls and returns

Format:	**ETM.CallReturnOnly** [**ON** | **OFF**]

Traces only functions calls and returns.

ETM.DataTrace Define broadcast of load/store address tracing

Format:	**ETM.DataTrace** *<def>* (ETMv4 and higher)
<def>:	**OFF**
	Address
	ReadAddress
	WriteAddress

Defines how data accesses are broadcast.

OFF	Address tracing in disabled.
Address	Load and store addresses are broadcasted.
ReadAddress	Load addresses are broadcasted.
Write Address	Store addresses are broadcasted.

ETM.DataTraceSelect Select with units or cluster are traced

Format:	**ETM.DataTraceSelect** [**ON** | **OFF**]

Selects with units or cluster are traced with data tracing.

ETM.DISableClockOff

Compress trace information

Format:	**ETM.DISableClockOff** [**ON** | **OFF**] (ETMv4 and higher)

OFF (default)	Use clock gating in wait/stop state.
ON	Disable clock gating in wait/stop state.

ETM.FillPort

Compress trace information

Format:	**ETM.FillPort** [**ON** | **OFF**]

OFF (default)	The trace information is organized byte-wise. The ETM broadcasts trace information as soon as it is available.
ON	The ETM collects the trace bytes until the full trace port width is used for broadcasting.

ETM.FillPort is automatically switched on, when **ETM.CycleAccurate** is switched on. This results in a better recording density in the trace memory.

Example for a 16-bit trace port:

```
ETM.FillPort OFF            ; if trace information is generated sparsely
                            ; single bytes are broadcasted

ETM.FillPort ON             ; if trace information is generated sparsely
                            ; trace information is only broadcasted
                            ; after 2 byte of trace information is
                            ; available
```

ETM.FillPort ON results in a less accurate timestamp, since trace information is time-stamped after it is broadcasted. For this reason it is recommended to use cycle accurate tracing here.

```
ETM.FillPort ON

ETM.CycleAccurate ON
```

ETM.ForceAtom Force atom

Format:	**ETM.ForceAtom** [**ON** | **OFF**]

Refer to the Hexagon ETM reference manual for more information.

ETM.ForceLdStKillAtom Force atom

Format:	**ETM.ForceLdStKillAtom** [**ON** | **OFF**]

Refer to the Hexagon ETM reference manual for more information.

ETM.GlobalSyncs GSYNC control

Format:	**ETM.GlobalSyncs** [**ON** | **OFF**] (ETMv4 and higher)

OFF	Disable Gsync packets.
ON	Enable Gsync packets.

ETM.GSyncPeriod Specify GSYNC period

Format:	**ETM.GSyncPeriod** *<period>*

Defines period between GSYNC packets (GSYNC_PERIOD). Default *<period>* is 0x4000 for off-chip tracing and 0x1000 for the ETB.

ETM.GSyncPeriodDisable — GSYNC period disable

Format:	**ETM.GSyncPeriodDisable** [**ON** | **OFF**]

Refer to the Hexagon ETM reference manual for more information.

ETM.GSyncWithUserPkt — GSYNC with USERPKT

Format:	**ETM.GSyncWithUserPkt** [**ON** | **OFF**]

Refer to the Hexagon ETM reference manual for more information.

ETM.IgnoreAtom — Ignore E/N atoms

Format:	**ETM.IgnoreAtom** [**ON** | **OFF**]

Default: OFF.

Ignores E/N atoms when set to ON.

ETM.IgnoreSyncOverflow — Handle ignore synch-overflow bit

Format:	**ETM.IgnoreSyncOverflow** [**ON** | **OFF**]

OFF (default)	Disable ignore synch-overflow bit.
ON	Enable ignore synch-overflow bit.

ETM.LoopTrace — Control for loop back branch packets

Format:	**ETM.LoopTrace** [**ON** | **OFF**]

OFF (default)	A branch address packet is only generated at the first loop back.
ON	A branch address packet is generated for every loop back.

ETM.OFF — Switch ETM off

Format:	**ETM.OFF**

Disables ETM functionality.

There is no need to disable the ETM functionality. In case of general problems with a new processor or if a ETM power-down should be enforced it can be reasonable.

ETM.ON

Switch ETM on

Format: **ETM.ON**

Enables ETM functionality.

ETM.PortMode

Select ETM mode

Format: **ETM.PortMode 1/2** | **1/4** | **1/6** | **1/8** | **1/10** | **ETB**

Specify the data rate of the trace port relative to the Hexagon clock.

```
ETM.PortMode 1/4                    ; trace information is broadcasted
                                    ; off-chip

                                    ; the data rate for the trace port
                                    ; is <core_clock>/4

ETM.PortMode ETB                    ; the on-chip Embedded Trace Buffer
                                    ; is used to record trace
                                    ; information
```

ETM.PortSize

Define trace port width

Format: **ETM.PortSize OFF** | **8** | **16** | **32**

Defines the number of pins used for the broadcasting of the trace packets.

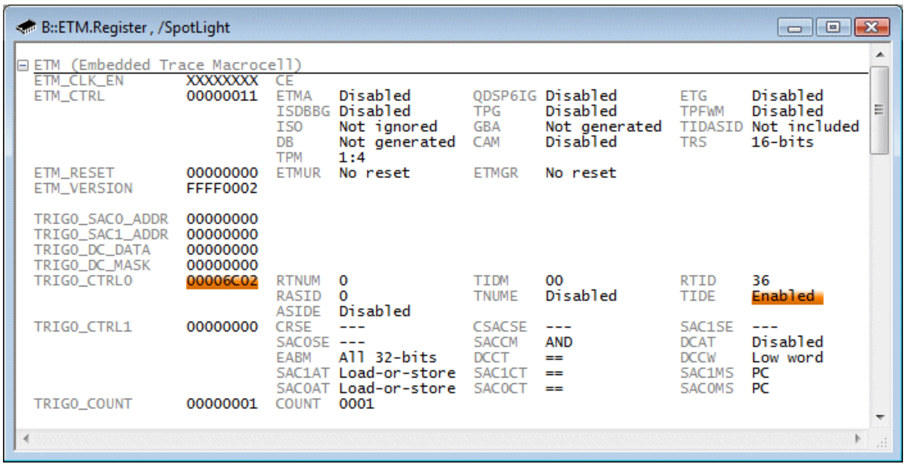
Format: **ETM.Register** [<*file*>] [**/**<*option*>]

<*option*>: **SpotLight** | **DualPort** | **Track** | **AlternatingBackGround**
CORE <*core_number*>

Displays the ETM register.

<*option*>	For a description of the options, see **PER.view**.

```
ETM.Register                         ; display the ETM registers based
                                     ; on the description of the
                                     ; default PER-file

ETM.Register perqetm6.per            ; display the ETM registers based
                                     ; on the description of the
                                     ; specified PER-file

ETM.Register , /SpotLight            ; display the ETM registers based
                                     ; on the description of the
                                     ; default PER-file

                                     ; mark changes
```


B::ETM.Register , /SpotLight

```
ETM (Embedded Trace Macrocell)
  ETM_CLK_EN        XXXXXXXX  CE
  ETM_CTRL          00000011  ETMA    Disabled        QDSP6IG Disabled        ETG     Disabled
                              ISDBBG  Disabled        TPG     Disabled        TPFWM   Disabled
                              ISO     Not ignored     GBA     Not generated   TIDASID Not included
                              DB      Not generated   CAM     Disabled        TRS     16-bits
                              TPM     1:4
  ETM_RESET         00000000  ETMUR   No reset        ETMGR   No reset
  ETM_VERSION       FFFF0002

  TRIG0_SAC0_ADDR   00000000
  TRIG0_SAC1_ADDR   00000000
  TRIG0_DC_DATA     00000000
  TRIG0_DC_MASK     00000000
  TRIG0_CTRL0       00006C02  RTNUM   0               TIDM    00              RTID    36
                              RASID   0               TNUME   Disabled        TIDE    Enabled
                              ASIDE   Disabled
  TRIG0_CTRL1       00000000  CRSE    ---             CSACSE  ---             SAC1SE  ---
                              SACOSE  ---             SACCM   AND             DCAT    Disabled
                              EABM    All 32-bits     DCCT    ==              DCCW    Low word
                              SAC1AT  Load-or-store   SAC1CT  ==              SAC1MS  PC
                              SAC0AT  Load-or-store   SAC0CT  ==              SAC0MS  PC
  TRIG0_COUNT       00000001  COUNT   0001
```

```
SYStem.CpuAccess Enable                 ; enable intrusive run-time memory
                                        ; access

ETM.Register , /DualPort                ; display the ETM registers based
                                        ; on the description of the
                                        ; default PER-file

                                        ; update while program is running


Go
```

ETM.RESet   Reset ETM settings

Format:	**ETM.RESet**

Reset the ETM settings and clear the ETM (see also **ETM.Clear**).

Format:	**ETM.Set** *<def>* **ETM.Set** *<action>*
<def>:	**Address** *<trigger_block> <access> <address>* [*<sequencer_level>*] [*<access> <address>* [*<sequencer_level>*]] **Address** *<trigger_block> <access> <range>* [*<sequencer_level>*] **ASID** *<trigger_block> <asid>* **TID** *<trigger_block> <value>*|*<bitmask>* **TNUM** *<trigger_block> <tnum>* **Data** *<trigger_block>* **==** | **!=** | **==B** | **!=B** |*<data>* *<value>* | *<float>* | *<bitmask>* **Count** *<trigger_block> <count>* **PROFILEMODE** *<counter> <event>* [*<tnum>* …]
<action>:	**Filter** *<trigger_block> <sequencer_level>* **CountReload** *<trigger_block> <sequencer_level>* *<state_transition> <trigger_block_match>* **Trigger** *<sequencer_level>* **STOP** *<sequencer_level>* **EXTOUT** *<sequencer_level>* **INTERRUPT** *<sequencer_level>* **PROFILE** *<count> <sequencer_level>*
<trigger_ block>:	**T0** | **T1** | **T2** | **T3**
<access>:	**Read** | **Write** | **ReadWrite** | **Program**
<sequencer_ level>:	**S0** | **S1** | **S2** | **ALL**

<state_ transition>:	**S0TO1** | **S0TO2** | **S1TO0** | **S1TO2** | **S2TO0** | **S2TO1**
<trigger_ block_ match>:	**T0** | **!T0** | **T1** | **!T1** | **T2** | **!T2** | **T3** | **!T3** | **T0&&T1** | **!T0&&T1** | **T0&&!T1** | **!T0&&!T1** | **T0&&T2** | **!T0&&T2** | **T0&&!T2** | **!T0&&!T2** | **T0&&T3** | **!T0&&T3** | **T0&&!T3** | **!T0&&!T3** | **T1&&T2** | **!T1&&T2** | **T1&&!T2** | **!T1&&!T2** | **T1&&T3** | **!T1&&T3** | **T1&&!T3** | **!T1&&!T3** | **T2&&T3** | **!T2&&T3** | **T2&&!T3** | **!T2&&!T3** | **T0&&T1&&T2&&T3** | **T0||T1** | **!T0||T1** | **T0||!T1** | **!T0||!T1** | **T0||T2** | **!T0||T2** | **T0||!T2** | **!T0||!T2** | **T0||T3** | **!T0||T3** | **T0||!T3** | **!T0||!T3** | **T1||T2** | **!T1||T2** | **T1||!T2** | **!T1||!T2** | **T1||T3** | **!T1||T3** | **T1||!T3** | **!T1||!T3** | **T2||T3** | **!T2||T3** | **T2||!T3** | **!T2||!T3** | **T0||T1||T2||T3** | **EXTIN** | **FALSE**
<counter>:	**C0** | **C1** | **C2** | **C3** | **C4** | **C5**
<event>:	**OFF** | **DCMISS** | **DCSTALL** | **DCCONFLICT** | **ICMISS** | **ICSTALL** | **ITLBMISS** | **DTLBMISS** | **STALLS**

ETM.Set Address *<trigger_block> <access> <address>* [*<sequencer_level>*] [*<access> <address>* [*<sequencer_level>*]]	
ETM.Set Address *<trigger_block> <access> <range> <sequencer_level>*	
<trigger_block>	**T0 | T1 | T2 | T3**
<access>	**Read | Write | ReadWrite | Program**
<sequencer_level>	**S0 | S1 | S2 | ALL**

The command **ETM.Set Address** configures the address comparators in the specified ETM trigger block. If a *<sequencer_level>* is specified:

- The instruction that performed the specified memory access is broadcasted.
- The specified instruction is broadcasted

```
; reset ETM settings
ETM.Clear

; configure one address comparator of in trigger block T0 to
; address 0x37fffb0/write access
ETM.Set Address T0 Write 0x37fffb0

; S0 is the start level of the sequencer
;the sequencer transits from S0 to S1 on the match of T0
ETM.Set S0TO1 T0

; configure one address comparator of in trigger block T1 to
; address 0x48858/execute
; broadcast all matches of trigger block T1 if the sequencer is in
; level S1
ETM.Set Address T1 Program 0x48858 S1
```

```
; reset ETM settings
ETM.Clear
; ETM.Set Address <trigger_block> <access> <address> <sequencer_level>
ETM.Set Address T0 Write 0x37fffb0 ALL

; equivalent Break.Set command
Break.Set 0x37fffb0 /Write /TraceEnable
```

The following command can be used to display the result:

```
MergedAnalyzer.List
```

Additional examples:

```
ETM.Set Address T0 Write 0x37fffb0 S0

ETM.Set Address T0 Program 0x4dd84 S2
```

Each ETM trigger block provides two single address comparators:

```
; reset ETM settings
ETM.Clear
; ETM.Set Address <trigger_block> <access> <address> <sequencer_level> \
; <access> <address> <sequencer_level>
ETM.Set Address T0 Read 0x37fffb0 ALL Write 0x37fffd0 ALL

; equivalent Break.Set command
Break.Set 0x37fffb0 /Read /TraceEnable
Break.Set 0x37fffd0 /Write /TraceEnable
```

```
ETM.Set Address T0 Read 0x37fffb0--0x37fffd0 ALL

ETM.Set Address T0 Program 4dd84--4dde0 ALL
```

ASID (Address Space Identifier)

ETM.Set ASID	*<trigger_block> <asid>*
<trigger_block>	**T0** | **T1** | **T2** | **T3**
<asid>	*<number>*

TID (Software Thread Identifier)

ETM.Set TID	*<trigger_block> <tid>*
<trigger_block>	**T0** | **T1** | **T2** | **T3**
<tid>	*<value>*|*<bitmask>*

TNUM (Hardware Thread Number)

ETM.Set TNUM	*<trigger_block> <tnum>*
<trigger_block>	**T0** | **T1** | **T2** | **T3**
<tnum>	*<number>*

Data (Data Comparator)

ETM.Set Data	*<trigger_block>* **==** | **!=** | **==B** | **!=B** |*<data>* *<value>* | *<float>* | *<bitmask>*
<trigger_block>	**T0** | **T1** | **T2** | **T3**

```
ETM.CLEAR                                   ; reset ETM configuration

ETM.Set Address T0 Write 0x37FFFC8 ALL      ; configure one address
                                            ; comparator in trigger
                                            ; block T0 to address
                                            ; 0x37FFFC8/write access


; ETM.Set Data <trigger_block> <data>       ; configure data comparator
ETM.Set Data T0 0x1                         ; in trigger block T0 for the
                                            ; match 0x1

MergedAnalyzer.List                         ; display the result
```

```
ETM.Set Data T0 == 0x1                 ; configure data comparator
                                       ; in trigger block T0 for the
                                       ; match 0x1

ETM.Set Data T0 != 0x1                 ; configure data comparator
                                       ; in trigger block T0 for the
                                       ; match !0x1

ETM.Set Data T0 ==B 0x4b               ; configure data comparator
                                       ; in trigger block T0 for any
                                       ; byte match 0x4b

ETM.Set Data T0 !=B 0x0                ; configure data comparator
                                       ; in trigger block T0 for any
                                       ; byte not matching 0x0

ETM.Set Data T0 == 0yxxxxxx11          ; configure data comparator
                                       ; in trigger block T0 for the
                                       ; match 0yxxxxxx11
```

Counter

ETM.Set Count	*<trigger_block> <count>*
<trigger_block>	**T0** | **T1** | **T2** | **T3**
<count>	*<number>*

Stop the program execution after the instruction at address 0x4dd84 was executed 20. times by the hardware thread 2. Broadcast only the execution of the address 0x4dd84.

```
ETM.CLEAR

ETM.Set Address T0 Program 0x4dd84 S0     ; configure one address
                                          ; comparator in trigger block
ETM.Set TNUM T0 2.                        ; T0 to the execution address
                                          ; 0x4dd84
ETM.Set Count T0 20.
                                          ; configure the trigger block
                                          ; T0 for the hardware thread 2
                                          ; and counter value 20.

                                          ; broadcast all cycles
                                          ; matching trigger block T0
                                          ; as long as sequencer level
                                          ; S0 is active

ETM.Set S0TO1 T0                          ; change to sequencer level S1
                                          ; after counted down

ETM.Set STOP S1                           ; stop the program execution
                                          ; when sequencer level S1
                                          ; becomes active

…

2Analyzer.List                            ; list the trace contents
                                          ; assigned to hardware thread
                                          ; 2

…

ETM.CLEAR
```

Filter

```
ETM.Set Address t2 Porgram  0x4dd84

ETM.Set Filter T2 ALL
```

ETM.Set Trigger	*<trigger_block> <sequencer_level>*
<trigger_block>	**T0 | T1 | T2 | T3**

```
; stop trace recording after the instruction at the address 0x48858
; was executed

; clear ETM program
ETM.Clear

; program address comparator T0
ETM.Set Address T0 Program 0x48858

; change from sequencer level S0 to S1 on T0 match
ETM.Set S0TO1 T0

; generate a trace trigger when sequencer is in level S1
ETM.Set Trigger S1
```

Equivalent **Break.Set** command:

```
Break.Set 0x48858 /Program /TraceTrigger
```

```
; stop trace recording after 0x4d was written to the address 0x37fffb0

ETM.Clear                                    ; clear ETM program

ETM.Set Address T0 Write 0x37fffb0           ; program address
                                             ; comparator T0

ETM.Set DAta T0 == 0x4d                      ; program data comparator
                                             ; T0

ETM.Set S0TO1 T0                             ; change from sequencer
                                             ; level S0 to S1 if TO
                                             ; after T0 occurred

ETM.Set Trigger S1                           ; generate a trace
                                             ; trigger when sequencer
                                             ; is in level S1
```

Equivalent **Break.Set** command:

```
Break.Set 0x37fffb0 /Write /Data.Word 0x4d /TraceTrigger
```

ETM.Set PROFILEMODE	*<counter_name> <event>* [*<tnum>*]
<counter_name>	**C0 | C1 | C2 | C3 | C4 | C5**
<event>	**OFF | DCMISS | DCSTALL | DCCONFLICT | ICMISS | ICSTALL | ITLBMISS | DTLBMISS | STALLS**

ETM.Set PROFILE	*<cycles> <sequencer_level>*
<sequencer_level>	**S0 | S1 | S2 | ALL**

```
; reset ETM programming registers
ETM.CLEAR

; Counter0 counts all stalls
ETM.Set PROFILEMODE C0 STALLS

; broadcast profile packet all 500. cycles
ETM.Set PROFILE 500. ALL

; display the result
MergedAnalyzer.List Counter0 DEFault List.NoDummy.OFF
```

```
; reset ETM programming registers
ETM.CLEAR

; Counter0 counts all stalls of hardware thread 0x3
ETM.Set PROFILEMODE C0 STALLS 0x3

; broadcast profile packet all 500. cycles
ETM.Set PROFILE 500. ALL

; display the result
3Analyzer.List Counter0 DEFault List.NoDummy.OFF
```

Format:	**ETM.STALL** [**ON** ❘ **OFF**] (ETMv4 and higher)

OFF (default)	Don´t use stall to avoid ETM fifo overflow.
ON	Commit stall to avoid ETM fifo overflow.

ETM.SynchPeriod — Specify ISYNC period

Format: **ETM.SynchPeriod** *<period>*

Defines period between instruction synchronization packets (ISYNC_PERIOD). Default *<period>* is 0x400 for off-chip tracing and 0x100 for the ETB.

ETM.TestBusBeforeSleep — TestBus before sleep

Format: **ETM.TestBusBeforeSleep** [**ON** | **OFF**]

Refer to the Hexagon ETM reference manual for more information.

ETM.TestBusSampleGroup — TestBus sample group

Format: **ETM.TestBusSampleGroup** [**ON** | **OFF**]

Refer to the Hexagon ETM reference manual for more information.

ETM.TestBusSweepingMode — TestBus sweeping mode

Format: **ETM.TestBusSweepingMode** [**ON** | **OFF**]

Enables/disables TestBus sweeping mode.

ETM.TestBusTrace TestBus tracing control

Format:	**ETM.TestBusTrace** [**ON** | **OFF**]

ON	Enable TestBus tracing.
OFF (default)	TestBus tracing is disabled.

ETM.TestBusTraceMUX Select bus for ‘TestBus’ tracing

Format:	**ETM.TestBusTraceMUX** *<mux>*

Select bus for TestBus Tracing (TEST_BUS_CTRL).

ETM.TestBusTracePeriod Specify test bus sampling time

Format:	**ETM.TestBusTracePeriod** *<cycle>*

Specify TestBus sampling time in cycles. (TEST_BUS_CTRL).

ETM.TestBusTriggerMode TestBus trigger mode

Format:	**ETM.TestBusTriggerMode** [**ON** | **OFF**]

Enables/disables TestBus trigger mode.

ETM.TImeMode — Improve ETM timestamp information

Format:	**ETM.TImeMode** *<mode>*
<mode>:	**OFF** | **External** | **ExternalInterpolated** | **SyncTimeStamps** | **AsyncTimeStamps** | **CycleAccurate** **CycleAccurate+External** **CycleAccurate+ExternalTrack** **CycleAccurate+SyncTimeStamps** **CycleAccurate+AsyncTimeStamps**

ETM.TImeSyncs — Time syncs

Format: **ETM.TImeSyncs** [**ON** | **OFF**]

Selects when timestamp information packets for CoreSight time correlation are generated.

ETM.Trace — Broadcasting of instruction flow

Format: **ETM.Trace** [**ON** | **OFF**]

ON	The ETM broadcasts the instruction flow information (default).
OFF	The ETM is not broadcasting the instruction flow information. Only the ETM triggers are active.

ETM.TraceASID — Broadcast instruction trace of specified ASID

Format: **ETM.TraceASID** *<asid>*

(no example code for testing available).

ETM.TraceID — Set ETM trace ID manually

Format:	**ETM.TraceID** *<id>*

CoreSight only.

ETM.TracePriority — Define priority of ETM messages

Format:	**ETM.TracePriority** *<priority>*

CoreSight only.

ETM.TraceTID — Broadcast instruction trace of specified software thread

Format:	**ETM.TraceTID** *<value>*|*<bitmask>*

Broadcast the instruction trace of the specified software thread.

```
ETM.TraceTID 0x11

ETM.TraceTID 0yxxxx0110
```

ETM.TraceTNUM Broadcast instruction trace of specified hardware thread

Format: **ETM.TraceTNUM** *<tnum>*

Broadcast the instruction trace of specified hardware thread.

```
ETM.TraceTNUM 2.                      ; limit the instruction trace to
                                      ; hardware thread 2

…                                     ; fill the trace buffer

2Analyzer.List                        ; display the trace information
                                      ; generated for the hardware
                                      ; thread 2
```

```
ETM.TraceTNUM 2.                      ; limit the instruction trace to
                                      ; hardware thread 2

…                                     ; fill the trace buffer

CORE.select 2.                        ;

Trace.List
```

ETM.TSyncWithUserPkt TSYNC with user packet

Format: **ETM.TSyncWithUserPkt** [**ON** | **OFF**]

Refer to the Hexagon ETM reference manual for more information.

ETM.state Display ETM setup

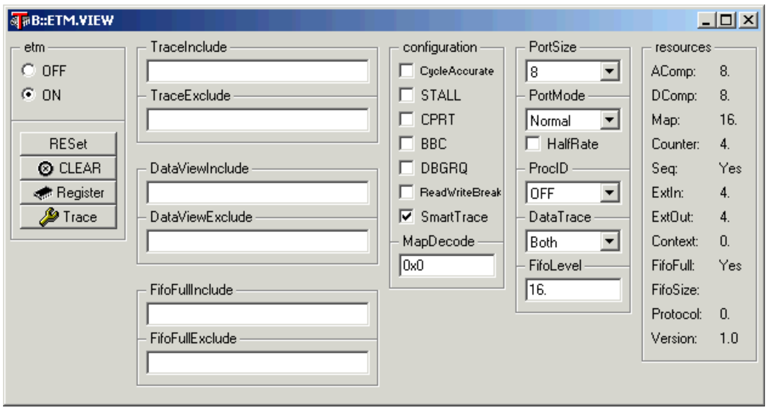
Format: **ETM.state**

Shows main ETM setup window.

Format:	**ETM.UserPktDisable** [**ON** | **OFF**]

ON	Disable UserPkt.
OFF (default)	Enable UserPkt.

Keyword for the Trace Display



etm: OFF [✓ ON]

TraceInclude:
TraceExclude:
DataViewInclude:
DataViewExclude:
FifoFullInclude:
FifoFullExclude:

configuration:
- ☐ CycleAccurate
- ☐ STALL
- ☐ CPRT
- ☐ BBC
- ☐ DBGRQ
- ☐ ReadWriteBreak
- ☑ SmartTrace

MapDecode: 0x0

PortSize: 8
PortMode: Normal
☐ HalfRate
ProcID: OFF
DataTrace: Both
FifoLevel: 16.

resources	
AComp:	8.
DComp:	8.
Map:	16.
Counter:	4.
Seq:	Yes
ExtIn:	4.
ExtOut:	4.
Context:	0.
FifoFull:	Yes
FifoSize:	
Protocol:	0.
Version:	1.0

TP0, TP1, TP2 … **TCTL**	Display the level of the specified trace port pin. Display the level of the TRACECTL pin.
ETM	Display the level of all trace port pins.
TP	Display the hex. value broadcasted by [TP0..TPn].
TPC	Display the broadcasted bytestream.
CORE	Display the number of the core that broadcasted the trace information.
Counter<n>	Display profiling event count.

```
Trace.List TP3 TP4 TCTL

Trace.List ETM

Trace.List TP TPC

MergedAnalyzer.List

Trace.List Counter2 DEFault
```

Analyzer.*<sub_cmd>*	The **Analyzer** commands perform on the trace information of the current core.
Onchip.*<sub_cmd>*	The **Onchip** commands perform on the trace information of the current core.
*<core>***Analyzer.***<sub_cmd>*	The **Analyzer** commands perform on the trace information of *<core>*.
*<core>***Onchip.***<sub_cmd>*	The **Onchip** commands perform on the trace information of *<core>*.
MergedAnalyzer.*<sub_cmd>*	The **Analyzer** commands perform on the trace information of all cores.
MergedOnchip.*<sub_cmd>*	The **Onchip** commands perform on the trace information of all cores.

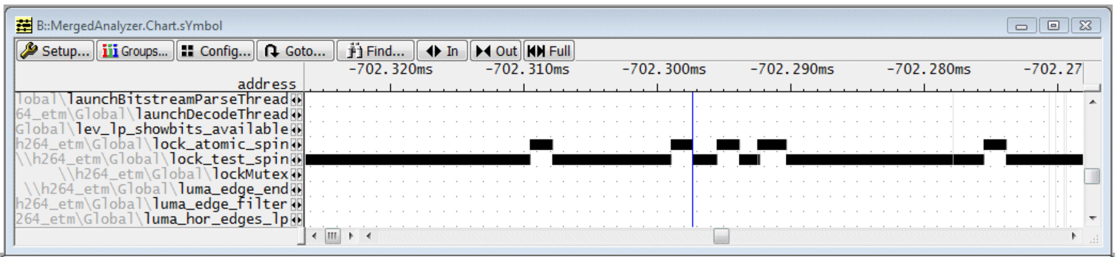
```
Core.select 2.                        ; select core 2 for visualization
                                      ; by TRACE32

Analyzer.List                         ; display the trace information of
                                      ; core 2
```

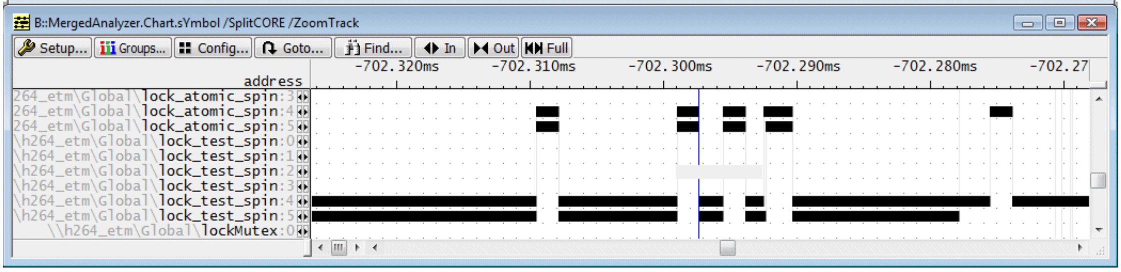
```
Core.select 0.                        ; select core 0 for visualization
                                      ; by TRACE32

Onchip.Chart.sYmbol                   ; perform a flat function run-time
                                      ; analysis based on the trace
                                      ; informatin of core 0
```

```
2Analyzer.List                        ; display the trace information of
                                      ; core 2

2Analyzer.STATistic.TASK              ; perform a task run-time analysis
                                      ; on the trace information of
                                      ; core 2
```

```
2Onchip.List                              ; display the on-chip trace
                                          ; information of core 2

2Onchip.Chart.sYmbol                      ; perform a flat function run-time
                                          ; analysis on the on-chip trace
                                          ; information of core 2
```

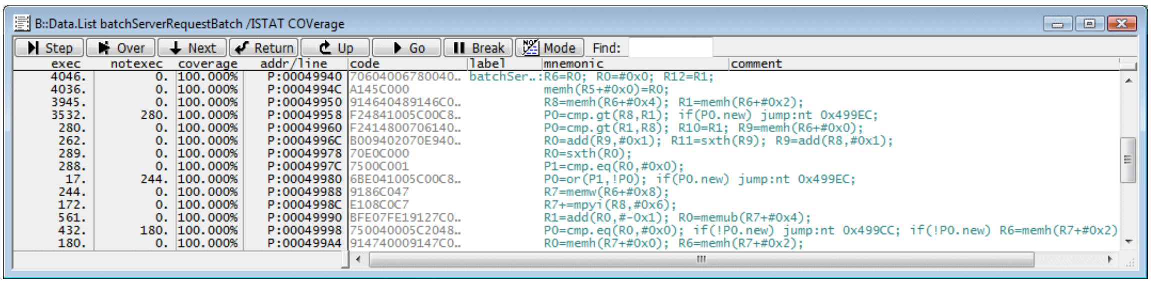
```
; perform a flat function run-time analysis based on the trace
; information of all cores
; merge the result of all cores
MergedAnalyzer.Chart.sYmbol

; perform a flat function run-time analysis based on the trace
; information of all cores
; split the result by cores
MergedAnalyzer.Chart.sYmbol /SplitCORE
```


B::MergedAnalyzer.Chart.sYmbol
Setup... Groups... Config... Goto... Find... In Out Full
-702.320ms -702.310ms -702.300ms -702.290ms -702.280ms -702.27
address
lobal\launchBitstreamParseThread
64_etm\Global\launchDecodeThread
Global\lev_lp_showbits_available
h264_etm\Global\lock_atomic_spin
\\h264_etm\Global\lock_test_spin
\\h264_etm\Global\lockMutex
\\h264_etm\Global\luma_edge_end
h264_etm\Global\luma_edge_filter
264_etm\Global\luma_hor_edges_lp

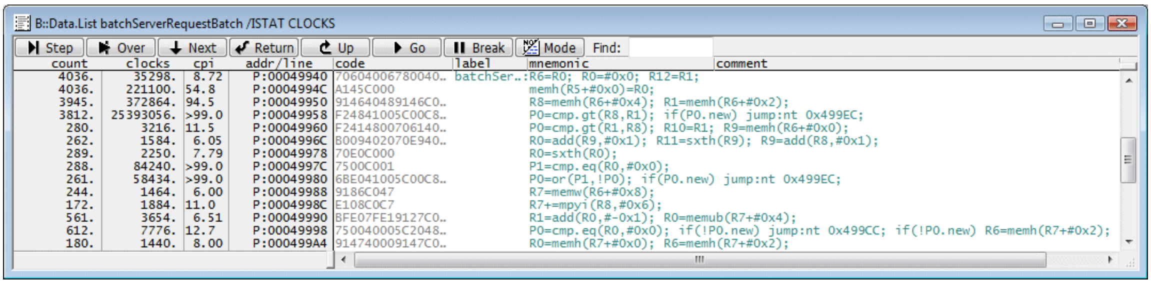

B::MergedAnalyzer.Chart.sYmbol /SplitCORE /ZoomTrack
Setup... Groups... Config... Goto... Find... In Out Full
-702.320ms -702.310ms -702.300ms -702.290ms -702.280ms -702.27
address
264_etm\Global\lock_atomic_spin:3
264_etm\Global\lock_atomic_spin:4
264_etm\Global\lock_atomic_spin:5
\h264_etm\Global\lock_test_spin:0
\h264_etm\Global\lock_test_spin:1
\h264_etm\Global\lock_test_spin:2
\h264_etm\Global\lock_test_spin:3
\h264_etm\Global\lock_test_spin:4
\h264_etm\Global\lock_test_spin:5
\\h264_etm\Global\lockMutex:0

Instruction statistic

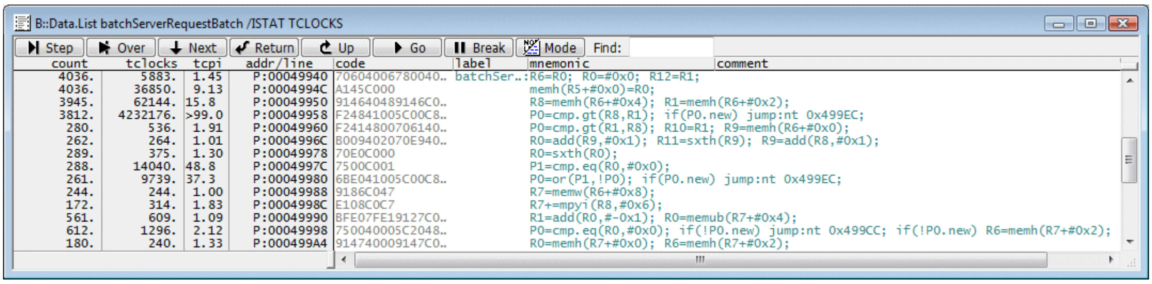
```
…

; switch cycle accurate tracing on to get accurate timing results
ETM.CycleAccurate ON

; inform TRACE32 about your core clock
Trace.CLOCK 500.MHz

Trace.Mode Leash

; reset ISTAT database
MergedAnalyzer.ISTATistic.RESet

; start program, the program execution is automatically stopped as
; soon as the trace buffer is full
Go

; add trace contents to ISTAT database
MergedAnalyzer.ISTATistic.add

; display ISTAT run-time results on function base
MergedAnalyzer.ISTATistic.ListFunc

; list details on code coverage of function batchServerRequestBatch
Data.List batchServerRequestBatch /ISTAT COVerage

; list run-time details of function batchServerRequestBatch
; based on core clock information
Data.List batchServerRequestBatch /ISTAT CLOCK

; list run-time details of function batchServerRequestBatch
; based on thread clock information (1/6 core clock)
Data.List batchServerRequestBatch /ISTAT TCLOCK

…
```



```
    exec   notexec  coverage   addr/line  code              label       mnemonic                                                                     comment
   4046.        0.  100.000%  P:00049940  70604006780040..  batchSer..:R6=R0; R0=#0x0; R12=R1;
   4036.        0.  100.000%  P:0004994C  A145C000                      memh(R5+#0x0)=R0;
   3945.        0.  100.000%  P:00049950  914640489146C0..              R8=memh(R6+#0x4); R1=memh(R6+#0x2);
   3532.      280.  100.000%  P:00049958  F24841005C00C8..              P0=cmp.gt(R8,R1); if(P0.new) jump:nt 0x499EC;
    280.        0.  100.000%  P:00049960  F2414800706140..              P0=cmp.gt(R1,R8); R10=R1; R9=memh(R6+#0x0);
    262.        0.  100.000%  P:0004996C  B009402070E940..              R0=add(R9,#0x1); R11=sxth(R9); R9=add(R8,#0x1);
    289.        0.  100.000%  P:00049978  70E0C000                      R0=sxth(R0);
    288.        0.  100.000%  P:0004997C  7500C001                      P1=cmp.eq(R0,#0x0);
     17.      244.  100.000%  P:00049980  6BE041005C00C8..              P0=or(P1,!P0); if(P0.new) jump:nt 0x499EC;
    244.        0.  100.000%  P:00049988  9186C047                      R7=memw(R6+#0x8);
    172.        0.  100.000%  P:0004998C  E108C0C7                      R7+=mpyi(R8,#0x6);
    561.        0.  100.000%  P:00049990  BFE07FE19127C0..              R1=add(R0,#-0x1); R0=memub(R7+#0x4);
    432.      180.  100.000%  P:00049998  750040005C2048..              P0=cmp.eq(R0,#0x0); if(!P0.new) jump:nt 0x499CC; if(!P0.new) R6=memh(R7+#0x2)
    180.        0.  100.000%  P:000499A4  914740009147C0..              R0=memh(R7+#0x0); R6=memh(R7+#0x2);
```


B::Data.List batchServerRequestBatch /ISTAT CLOCKS

```
   count      clocks   cpi    addr/line  code               label        mnemonic                                                                    comment
   4036.      35298.   8.72   P:00049940 70604006780040..   batchSer..:R6=R0; R0=#0x0; R12=R1;
   4036.     221100.  54.8    P:0004994C A145C000                      memh(R5+#0x0)=R0;
   3945.     372864.  94.5    P:00049950 914640489146C0..              R8=memh(R6+#0x4); R1=memh(R6+#0x2);
   3812.   25393056. >99.0    P:00049958 F24841005C00C8..              P0=cmp.gt(R8,R1); if(P0.new) jump:nt 0x499EC;
    280.       3216.  11.5    P:00049960 F2414800706140..              P0=cmp.gt(R1,R8); R10=R1; R9=memh(R6+#0x0);
    262.       1584.   6.05   P:0004996C B009402070E940..              R0=add(R9,#0x1); R11=sxth(R9); R9=add(R8,#0x1);
    289.       2250.   7.79   P:00049978 70E0C000                      R0=sxth(R0);
    288.      84240. >99.0    P:0004997C 7500C001                      P1=cmp.eq(R0,#0x0);
    261.      58434. >99.0    P:00049980 6BE041005C00C8..              P0=or(P1,!P0); if(P0.new) jump:nt 0x499EC;
    244.       1464.   6.00   P:00049988 9186C047                      R7=memw(R6+#0x8);
    172.       1884.  11.0    P:0004998C E108C0C7                      R7+=mpyi(R8,#0x6);
    561.       3654.   6.51   P:00049990 BFE07FE19127C0..              R1=add(R0,#-0x1); R0=memub(R7+#0x4);
    612.       7776.  12.7    P:00049998 750040005C2048..              P0=cmp.eq(R0,#0x0); if(!P0.new) jump:nt 0x499CC; if(!P0.new) R6=memh(R7+#0x2);
    180.       1440.   8.00   P:000499A4 914740009147C0..              R0=memh(R7+#0x0); R6=memh(R7+#0x2);
```


B::Data.List batchServerRequestBatch /ISTAT TCLOCKS

```
   count     tclocks  tcpi    addr/line  code               label        mnemonic                                                                    comment
   4036.       5883.   1.45   P:00049940 70604006780040..   batchSer..:R6=R0; R0=#0x0; R12=R1;
   4036.      36850.   9.13   P:0004994C A145C000                      memh(R5+#0x0)=R0;
   3945.      62144.  15.8    P:00049950 914640489146C0..              R8=memh(R6+#0x4); R1=memh(R6+#0x2);
   3812.    4232176. >99.0    P:00049958 F24841005C00C8..              P0=cmp.gt(R8,R1); if(P0.new) jump:nt 0x499EC;
    280.        536.   1.91   P:00049960 F2414800706140..              P0=cmp.gt(R1,R8); R10=R1; R9=memh(R6+#0x0);
    262.        264.   1.01   P:0004996C B009402070E940..              R0=add(R9,#0x1); R11=sxth(R9); R9=add(R8,#0x1);
    289.        375.   1.30   P:00049978 70E0C000                      R0=sxth(R0);
    288.      14040.  48.8    P:0004997C 7500C001                      P1=cmp.eq(R0,#0x0);
    261.       9739.  37.3    P:00049980 6BE041005C00C8..              P0=or(P1,!P0); if(P0.new) jump:nt 0x499EC;
    244.        244.   1.00   P:00049988 9186C047                      R7=memw(R6+#0x8);
    172.        314.   1.83   P:0004998C E108C0C7                      R7+=mpyi(R8,#0x6);
    561.        609.   1.09   P:00049990 BFE07FE19127C0..              R1=add(R0,#-0x1); R0=memub(R7+#0x4);
    612.       1296.   2.12   P:00049998 750040005C2048..              P0=cmp.eq(R0,#0x0); if(!P0.new) jump:nt 0x499CC; if(!P0.new) R6=memh(R7+#0x2);
    180.        240.   1.33   P:000499A4 914740009147C0..              R0=memh(R7+#0x0); R6=memh(R7+#0x2);
```

Benchmark Counters

The ETM contains six counters which can count various performance events e.g. instruction cache misses, stalls. Counter values are broadcasted every 1000. clock cycles by default.

TRACE32 maintains these six counters with the command group **BMC.<sub_cmd>**.

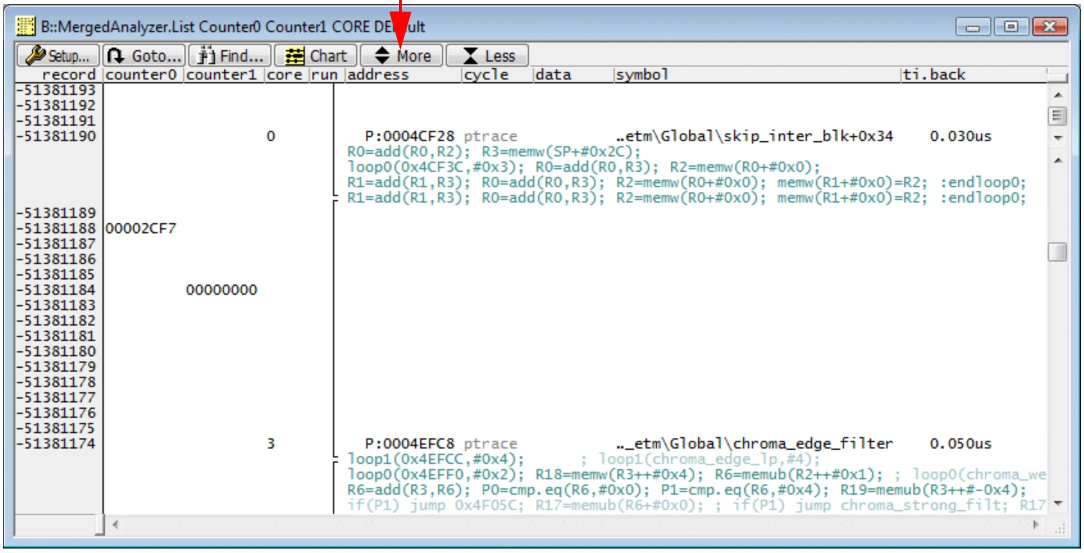
Please refer to **"Hexagon Debugger"** (debugger_hexagon.pdf) for a description of the Hexagon specific **BMC** commands and to **"BMC"** (general_ref_b.pdf) for information about *architecture-independent* **BMC** commands.

Example 1:

Count the stalls and the instruction cache stalls while the program is running. Broadcast the counter values all 5000. clock cycles.

```
; Reset BMC system
BMC.RESet

; advise the ETM to broadcast the counter values all 5000. clock cycles
BMC.CyclePeriod 5000.

; counter 0 counts all stalls
BMC.Counter0 STALLS

; counter 1 counts instruction cache stalls
BMC.Counter1 ICSTALL

; enable the counter broadcasting
BMC.ON

Go

Break

; display trace listing with counter values
MergedAnalyzer.List Counter0 Counter1 DEFault

; for the counter display push the More button in the MergedAnalyzer.List
; window
```

Push the **More** button to get the counter display



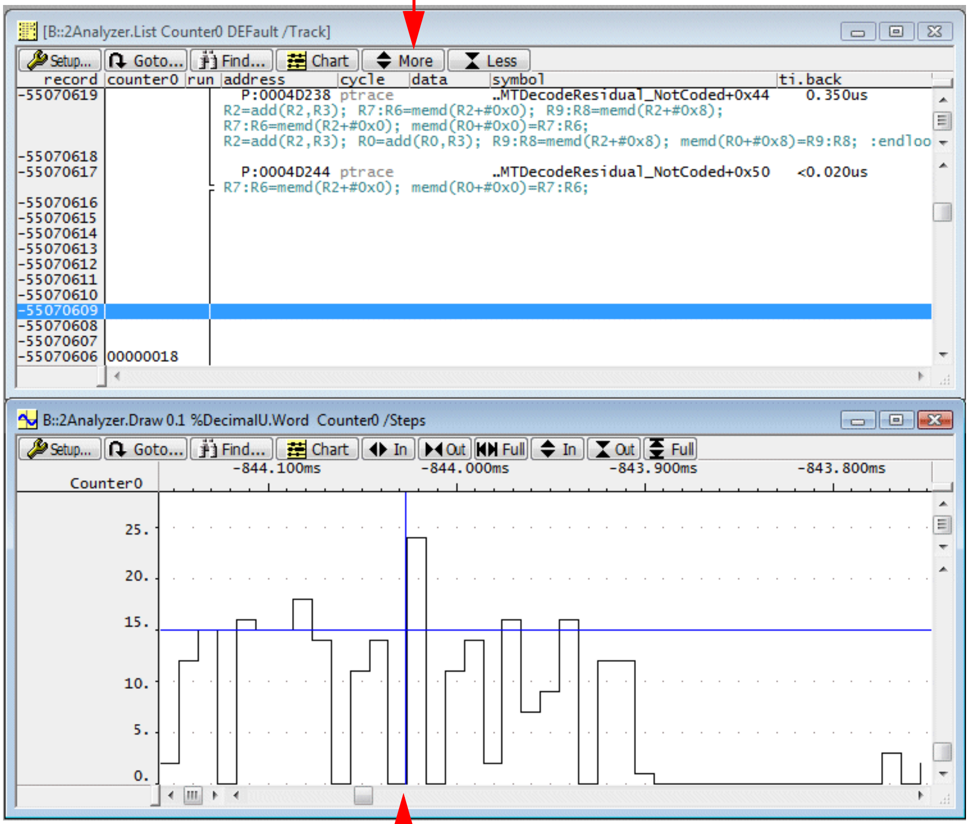
```
    record counter0 counter1 core run address        cycle   data    symbol                          ti.back
-51381193
-51381192
-51381191
-51381190                     0         P:0004CF28 ptrace            ..etm\Global\skip_inter_blk+0x34    0.030us
                                      R0=add(R0,R2); R3=memw(SP+#0x2C);
                                      loop0(0x4CF3C,#0x3); R0=add(R0,R3); R2=memw(R0+#0x0);
                                      R1=add(R1,R3); R0=add(R0,R3); R2=memw(R0+#0x0); memw(R1+#0x0)=R2; :endloop0;
                                      R1=add(R1,R3); R0=add(R0,R3); R2=memw(R0+#0x0); memw(R1+#0x0)=R2; :endloop0;
-51381189
-51381188 00002CF7
-51381187
-51381186
-51381185
-51381184          00000000
-51381183
-51381182
-51381181
-51381180
-51381179
-51381178
-51381177
-51381176
-51381175
-51381174                     3         P:0004EFC8 ptrace            .._etm\Global\chroma_edge_filter    0.050us
                                      loop1(0x4EFCC,#0x4);      ; loop1(chroma_edge_lp,#4);
                                      loop0(0x4EFF0,#0x2); R18=memw(R3++#0x4); R6=memub(R2++#0x1); ; loop0(chroma_we
                                      R6=add(R3,R6); P0=cmp.eq(R6,#0x0); P1=cmp.eq(R6,#0x4); R19=memub(R3++#-0x4);
                                      if(P1) jump 0x4F05C; R17=memub(R6+#0x0); ; if(P1) jump chroma_strong_filt; R17
```

Example 2:

Count all stalls for hardware thread 2. Broadcast the counter all 2000. clock cycles. Inspect peak areas afterwards.

```
; advise the ETM to broadcast the counter values all 2000. clock cycles
BMC.CyclePeriod 2000.

; counter 0 counts all stalls of hardware thread 2.
; instruction trace broadcasting is limited to hardware thread 2.
BMC.Counter0 STALLS
ETM.TraceTNUM 2.

; enable the counter broadcasting
BMC.ON

Go

Break

; display trace listing with counter value
; window cursor follows 2Analyzer.Draw window cursor movement
2Analyzer.List Counter0 DEFault /Track

; display timing of counter window graphically
; 2Analyzer.DRAW <scaling> <format> <item> /<formatting_option>
2Analyzer.Draw 0.1 %DecimalU.Word  Counter0 /Steps

…

; remove hardware thread filter
ETM.TraceTNUM

; Reset BMC system
BMC.RESet
```

Push the **More** button to get the counter display



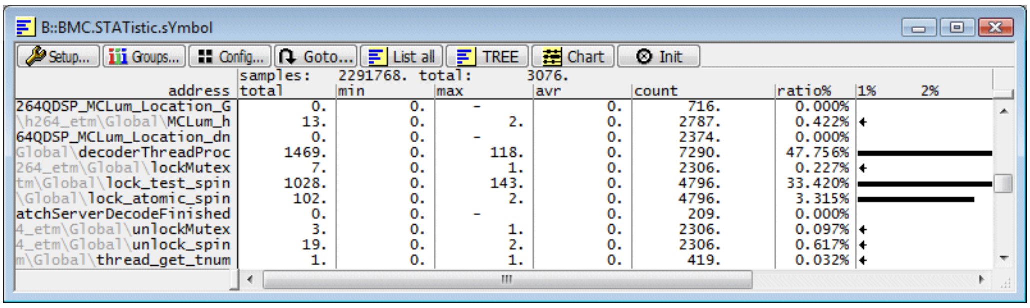
```
 record   |counter0 |run |address        |cycle  |data    |symbol                          |ti.back
-55070619                  P:0004D238 ptrace           ..MTDecodeResidual_NotCoded+0x44     0.350us
                        R2=add(R2,R3); R7:R6=memd(R2+#0x0); R9:R8=memd(R2+#0x8);
                        R7:R6=memd(R2+#0x0); memd(R0+#0x0)=R7:R6;
                        R2=add(R2,R3); R0=add(R0,R3); R9:R8=memd(R2+#0x8); memd(R0+#0x8)=R9:R8; :endloo
-55070618
-55070617                  P:0004D244 ptrace           ..MTDecodeResidual_NotCoded+0x50    <0.020us
                        R7:R6=memd(R2+#0x0); memd(R0+#0x0)=R7:R6;
-55070616
-55070615
-55070614
-55070613
-55070612
-55070611
-55070610
-55070609
-55070608
-55070607
-55070606 00000018
```

Move cursor to step to inspect peak area in **2Analyzer.List**

Example 3:

Perform a flat function run-time analysis to check the distribution of instruction cache misses.

```
BMC.RESet                          ; reset benchmark counter system

BMC.CyclePeriod 500.               ; advise the ETM to broadcast the
                                   ; counter values all 500. clock
                                   ; cycles

BMC.Counter0 ICMISS                ; counter 0 count the instruction
                                   ; cache misses

BMC.ON                             ; enable the benchmark counters

Go

Break

BMC.SELect Counter0                ; select counter 0 for the entered
                                   ; statistic command

BMC.STATistic.sYmbol               ; check the distribution of the
                                   ; instruction cache misses
```



```
                          samples:   2291768.  total:      3076.
                  address total      min        max        avr        count         ratio%  1%      2%
264QDSP_MCLum_Location_G        0.         0.         -          0.        716.   0.000%
\h264_etm\Global\MCLum_h       13.         0.         2.         0.       2787.   0.422%
64QDSP_MCLum_Location_dn        0.         0.         -          0.       2374.   0.000%
Global\decoderThreadProc     1469.         0.       118.         0.       7290.  47.756%
264_etm\Global\lockMutex        7.         0.         1.         0.       2306.   0.227%
tm\Global\lock_test_spin     1028.         0.       143.         0.       4796.  33.420%
\Global\lock_atomic_spin      102.         0.         2.         0.       4796.   3.315%
atchServerDecodeFinished        0.         0.         -          0.        209.   0.000%
4_etm\Global\unlockMutex        3.         0.         1.         0.       2306.   0.097%
4_etm\Global\unlock_spin       19.         0.         2.         0.       2306.   0.617%
m\Global\thread_get_tnum        1.         0.         1.         0.        419.   0.032%
```