

Training Intel® Processor Tracing

Release 09.2023



TRACE32 Online Help	
TRACE32 Directory	
TRACE32 Index	
TRACE32 Training	
Training Intel® x86/x64	
Training Intel® Processor Tracing	1
Protocol Description	6
Basic Trace Packets	6
OS-Aware Tracing	7
Time Information	8
Tool Timestamp (POWER TRACE II / POWER TRACE III only)	8
CycleAccurate Tracing	10
Synchronization Time	10
Trace Configuration	11
Off-chip Trace	11
SDRAM Trace	18
Trace Errors	22
ERRORS	22
TARGET FIFO OVERFLOW	25
TRACE32 Abstractions	27
SystemTrace	27
(Core) Trace	29
Displaying the Trace Contents	31
Influencing Factors on the Trace Information	31
Settings in the TRACE32 Trace Configuration Window	32
States of the Trace	43
The AutoInit Command	44
Basic Display Commands	45
Default Listing	45
Basic Formatting	47
Correlating the Trace Listing with the Source Listing	48
Browsing through the Trace Buffer	49
Find a Specific Record	50
Display Items	51
Default Display Items	51
Further Display Items	54
Belated Trace Analysis	60

Save the Trace Information to an ASCII File	61
Postprocessing with TRACE32 Instruction Set Simulator	62
Export STP Byte Stream	67
Trace Control by Filters	68
TraceEnable	69
TraceOFF	71
OS-Aware Tracing	73
Process Switch Packets	73
Program Flow and Process Switches	75
Process Runtime Analysis	76
Time Chart	77
Statistic	78
Find Process Switches in the Trace	79
OS-aware Filtering	81
Filtering by Privilege Level	81
Filtering by Process	82
Filter on Function executed by Process	84
Belated Analysis	86
Example for Linux	86
Trace-based Debugging (CTS)	88
Setup	88
Get Started	89
Forward and Backward Debugging	92
Forward Debugging	92
Backward Debugging	92
CTS Technique	93
Function Run-Time Analysis - Basic Concept	94
Software under Analysis (no OS, OS or OS+MMU)	94
Flat vs. Nesting Analysis	94
Basic Knowledge about Flat Analysis	95
Basic Knowledge about Nesting Analysis	96
Summary	98
Flat Function-Runtime Analysis	99
Function Time Chart	99
Default Time Chart	99
Core Options	100
TASK Options	101
Function Run-time Statistic	103
Further Commands	104
Nesting Function Analysis OS	105
Survey	106

	range Column	107
	Default Results	110
	Net Results	112
	Interrupt Details	114
	Time in Other Tasks	115
	Tree Display	116
Ś	Structure your Trace Evaluation	117
ç	Structure your Trace Evaluation GROUPs for OS-aware Tracing	117 117
S	-	
S	GROUPs for OS-aware Tracing	117
	GROUPs for OS-aware Tracing GROUP Status ENable	117 118
	GROUPs for OS-aware Tracing GROUP Status ENable GROUP Status ENable+Merge	117 118 119

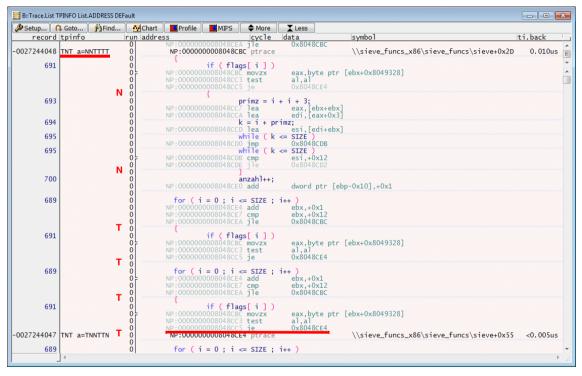
Version 10-Oct-2023

Basic Trace Packets

To enable a trace tool to reconstruct the instruction execution sequence the following trace packets are generated:

TNT packets

Taken Not Taken packets track the direction of up to 6 conditional branches. Since the address at which the program execution continues when the branch was taken is part of the source code TNT packets provide sufficient information to reconstruct the instruction execution sequence.



Target IP packets

Ret instructions, register indirect calls and similar instructions as well as exception and interrupts cause the generation of a Target IP packet. Since the address at which the program execution continues is only known at run-time, a Target IP packet contains this address fully or in a compressed format.

🛿 Setup 🛛 🖪	🖡 Goto 🏾 🎁 Find 🛛 🚹	Chart Profile	MIPS	More Less					
record	tpinfo	ru	n addres		cycle	data	symbol	ti.back	
027232833	Target IP 4b Zext B	0 LIP=0x8048C80		NP:000000008048CE		0x8048CBC	\sieve+0x5D	<0.005us	
				}					
704		0		return anzahl; NP:000000008048CE	C mov	eax,[ebp-0x10]			
705		0 0 0 0 0	}	NP:000000008048CE NP:000000008048CF NP:000000008048CF NP:00000008048CF NP:00000008048CF	F add 2 pop 3 pop 4 pop	esp,+0x10 ebx esi edi ebp			
27232827	TNT a=TTTTTT	0 0 0		NP:000000008048CF	6 ret	cop	\main+0x29F	0.010us	
		Ō		/** func40()	;**/				
	4								ł

OS-Aware Tracing

Paging Information Packet (PIP)

x86/x64 processors have a CR3 control register that contains the Process Context Identifier (PCID). On every context switch the corresponding PCID is loaded to CR3.

Intel[®] PT generates a Paging Information Packet (PIP) when a write to CR3 occurs.

B::Analyzer.List	TPINFO DEFault List.TASK /CORE 1.							×
Setup 📭 record	Goto 🎒 Find Mchart MProf tpinfo	ile ■MIPS ♦ More X Less run address	cycle	data	symbol		ti.back	
38		if (likely(prev ! cmp r12,r13 je 0xFFFFFFF8284						4 III +
	TIP CRO.PEEL CR3=0xBC49000 TNT a=NTNTNT	XP:05E8:FFFFFFF8284 this_cpu_ mov gs:[0x12B80],r	write(cpu_t 12 ; gs:cpu	lbstate.active_mm	ux\sched/core_	_schedule+0x2EA	<0.005u 1.040u	
	4	•	memory),					+

Tool Timestamp (POWER TRACE II / POWER TRACE III only)

POWER TRACE II / POWER TRACE III timestamps the trace information when it is stored in its trace buffer.

The resolution of the POWER TRACE II / POWER TRACE III timestamp is 5 ns.

8 trace record have always an identical timestamp. There are two reasons for this:

- The TRACE32 recording technology.
- The smallest Intel[®] PT packet is one byte.

🌽 Setup	🔒 Goto	Find	Chart	Profile	MIPS	More	Less			
record	d address			cycle	symbol				ti.zero	
-0000000226	5	NP:000000	0008048C80	ptrace	\\sieve_t	uncs_x86	sieve_fund	cs\main+0x29F	-0.010us	
00000022									-0.010us	:
-000000224			0008048CA6					cs\sieve+0x17	0.000us	
00000022			0008048CA6		\\sieve_f	uncs_x86	sieve_fund	cs\sieve+0x17	0.000us	
0000000222			0008048CA6					cs\sieve+0x17	0.000us	
-0000000221			0008048CD2					cs\sieve+0x43	0.000us	
-0000000220			0008048CE0					cs\sieve+0x51	0.000us	8
-0000000219			0008048CE0					cs\sieve+0x51	0.000us	
-0000000218			0008048CBC					cs\sieve+0x2D	0.000us	
-0000000217			0008048CBC					cs\sieve+0x2D	0.000us	
-0000000216			0008048CBC					cs\sieve+0x2D	0.010us	
-000000021			0008048CBC					cs\sieve+0x2D	0.010us	
-0000000214			0008048CE4					cs\sieve+0x55	0.010us	
-0000000213			0008048CBC					cs\sieve+0x2D	0.010us	8
-0000000212			0008048CE4					cs\sieve+0x55	0.010us	•
000000021		NP:000000	0008048CEC	ptrace	\\sieve_f	uncs_x86	sieve_fun	cs\sieve+0x5D	0.010us	
000000021									0.010us	
-0000000209									0.010us	
0000000208									0.020us	
0000000207			0008048C80					cs\main+0x29F	0.020us	-
0000000206		NP:000000	0008048C8A	ptrace	\\sieve_f	uncs_x86	sieve_fund	cs\main+0x2A9	0.020us	L
000000020	5								0.020us	

In the standard trace display timestamp information is displayed for the first record with the new timestamp. All following records with an identical timestamp show <0.005us.

B::Trace.List				
Setup	Goto	🛱 Find 🚻 Chart 📕 Profile 📕 MIPS 🗢 More	e Less	
record	run addr		symbol	ti.back
-0000000224	0 jle 0 mov 0 add 0 cmp 0 jle 0 mov 0 add 0 cmp 0 jle	0x8048CA6 NF:00000008048CA6 ptrace byte ptr [ebx+0x8049328],0x1 ebx,+0x1 0x8048CA6 byte ptr [ebx+0x8049328],0x1 ebx,+0x1 ebx,+0x1 0x8048CA6	\sieve+0x17	0.010us
	0 mov 0 add 0 cmp 0 jle 0 mov 0 add	<pre>byte ptr [ebx+0x8049328],0x1 ebx,+0x1 ebx,+0x12 0x8048CA6 byte ptr [ebx+0x8049328],0x1 ebx,+0x1</pre>		
	0 cmp 0 jle 0 add 0 cmp 0 jle 0 mov 0 add	ebx,+0x12 0x8048CA6 byte ptr [ebx+0x8049328],0x1 ebx,+0x12 0x8048CA6 byte ptr [ebx+0x8049328],0x1 ebx,+0x1		
-0000000223	0 cmp 0 jle 0 0 mov	ebs,+0x12 0x8048CA6 NP:000000008048CA6 ptrace byte ptr [ebx+0x8049328],0x1	\sieve+0x17	<0.005us

If configured Intel[®] PT can generate cycle count information. The cycle count information indicates how much core clocks it took to execute a program section.

Setup	Goto	I Chart	Profile	MIPS	Mor	Less				
	tpinfo		cloc	ks.b	run addre		cycle	data	symbol	
00000231	TNT a=TTNTNT	cc=0x23		35.	0 jle 0 add 0 cmp 0 jle 0 add 0 jle 0 add 0 jle 0 add 0 jle 0 add 0 jle	byte ptr ebx,+0x1 0x8048CA1 byte ptr ebx,+0x1 ebx,+0x1 0x8048CA1	8048CA6 ptrace [ebx+0x8049328 6 [ebx+0x8049328 2 6 [ebx+0x8049328 2 5 [ebx+0x8049328 2],0x1	\\sieve_funcs_x86\sieve_funcs\sieve+0x17	7
689					0 0 mov 0 jmp 0 cmp 0 jle	for (i = 0 ebx,0x0 0x8048CE ebx,+0x1 0x8048CB	2	i++)		
691					0 0 0 0 0 test 0 je	eax,byte al,al 0x8048CE4	(flags[i]) ptr [ebx+0x804 4	9328]		
693					0 0 lea 0 lea	۱ eax,[ebx- edi,[eax-		+ i + 3;		
694					0 0 lea	esi,[edi-	k = i + pr	imz;		
695					0 0 jmp	0x8048CD	while (k	<= SIZE)		
695					0 0 0 0 1 1 e	esi,+0x12 0x8048CD	while (k	<= SIZE)		
00000209	TNT a=TTTTTN	cc=0x26		38.	0	NP:00000000	8048CD2 ptrace		<pre>\\sieve_funcs_x86\sieve_funcs\sieve+0x4</pre>	3
697 698						byte ptr	[esi+0x8049328	ags[k] = FA],0x0 += primz;	ALSE;	

Cycle accurate tracing requires up to 2 times more bandwidth.

Synchronization Time

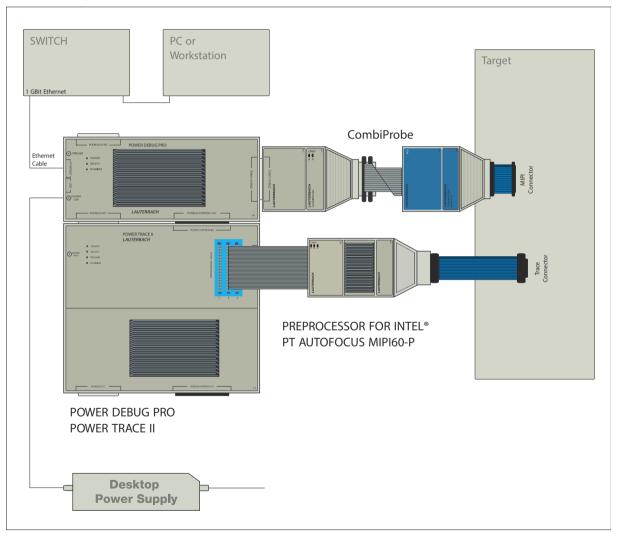
Not implemented yet.

Off-chip Trace

Recording the trace information exported via a PTI (Parallel Trace Interface) requires:

• A POWER TRACE II hardware (1 GByte, 2 GByte or 4 GByte of trace memory) or a POWER TRACE III hardware (4 GByte or 8 GByte of trace memory)

TRACE32 PowerView uses the name **Analyzer** to refer to the trace memory within POWER TRACE II / POWER TRACE III.



An Preprocessor for Intel[®] Atom[™] AUTOFOCUS 600 MIPI

The following configuration steps are required for off-chip tracing:

1. Configure Parallel Trace Interface on target.

Configuration is required for:

- PTI port size
- PTI frequency
- GPIO pins used for PTI

The following commands are provided for this purpose:

; write <value> to the configuration register addressed by A:<physical_address> ; in the specified <format> PER.Set.simple A:<physical_address> %<format> <value> ; write <value> to the memory location addressed by A:<physical_address> ; in the specified <format> Data.Set A:<physical_address> %<format> <value>

Data.Set is equivalent to PER.Set.simple if the configuration register is memory mapped.

The access class A: allow to use the physical address for the write operations.

```
Per.Set.simple A:0xf9009000 %Long 0x3e715
Data.Set A:0xf9009000 %Long 0x3e715
```

Please refer to your chip manual for the physical addresses of the configuration registers.

2. Configure TRACE32 for a PTI that exports STP (System Trace Protocol) packets.

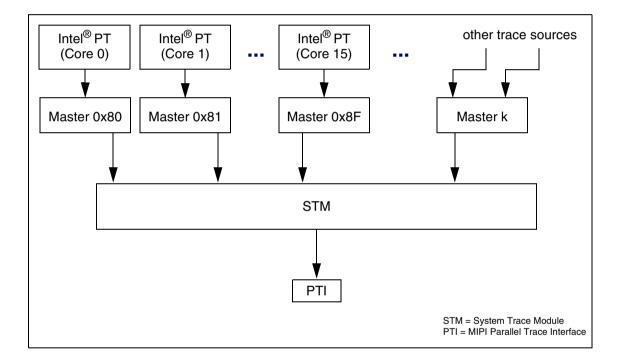
SYStem.CONFIG STM Mode STP64	; inform TRACE32 that your ; chip provides a STM that ; generated 64-bit STPv1 ; packets
STM.PortSize 16.	; inform TRACE32 that your ; PTI size is 16 pins

3. Inform TRACE32 which core traces you want to analyze.

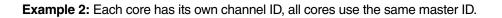
IPT.TraceID <value> <bitmask></bitmask></value>	Specify which masters/channels (that produce Intel [®]
	PT trace information) you want to analyze with the help of TRACE32.

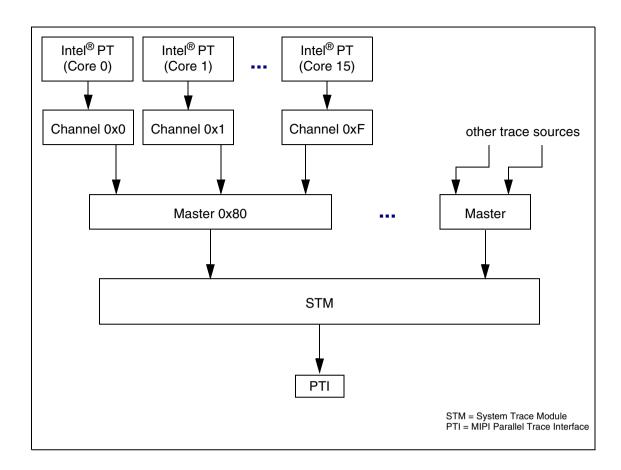
<value></value>	<i><value></value></i> is a 32-bit number. The first 16 bits represent the master ID, the last 16 bits represent the channel ID.
<bitmask></bitmask>	bitmask representation of <i><value></value></i>

Example 1: Each core has its own master ID.



IPT.TraceID 0x00800000	; master ID 0x80 is used to export Intel $^{\textcircled{R}}$; PT trace information for core 0
IPT.TraceID 0x008x0000	; master ID 0x80, 0x81, 0x82 are used to ; export Intel [®] PT trace information ; master ID 0x80 represents core 0 ; the other master IDs consecutively ; represent core 1 to core 15





IPT.TraceID 0x0080000x	; master ID 0x80 is used to export Intel [®] ; PT ; trace information ; channel ID 0x0 represents core 0 ; the other channel IDs consecutively ; represent core 1 to core 15

4. Enable Intel[®] PT on the target and allow TRACE32 to configure it.

IPT.ON

5. Calibrate the *Preprocessor for Intel[®] Atom™ AUTOFOCUS 600 MIPI* for recording.

TRACE32 supports three methods of generating outputs on the trace lines for calibration.

- On-chip test pattern generator (not tested yet).
- Test executable provided by Lauterbach.
- Application program.

Please be aware that TRACE32 PowerView displays "Analyzer data capture o.k." only if:

- All trace lines toggled while calibration is performed.
- There are no short circuits between the trace lines.
- An error-free trace decoding was possible.

Test executable provided by Lauterbach

In order to use the test executable provided by Lauterbach for calibration, the following command sequence is recommended.

```
; example for a free-running clock (Tangier)
AREA.view
    ; open TRACE32 Message AREA
    ; to observe calibration
    ; results
Analyzer.THreshold VCC
    ; advise TRACE32 to use
    ; 1/2 VCC as threshold level
    ; for the trace signals
Analyzer.AutoFocus /NoTHreshold
    ; start the calibration by
    ; using test executable
    ; advise TRACE32 to keep
    ; the threshold level
```

A manual setup is required if your target is using a gated clock. Refer to "Manual Setup" in AutoFocus User's Guide, page 18 (autofocus_user.pdf) for assistance.

Application program

In order to use the application program for calibration, the following command sequence is recommended.

```
; example for a free-running clock (Tangier)
AREA.view
                                      ; open TRACE32 Message AREA
                                      ; to observe calibration
                                      ; results
Data.LOAD.Elf demo x86.elf /PlusVM
                                      ; download application program
                                      ; to the target,
                                      ; in order to perform trace
                                      ; decoding while the
                                      ; application program is
                                      ; running, the program code
                                      ; has to be copied to the
                                      ; TRACE32 Virtual Memory
Go
                                      ; start the execution of the
                                      ; application program
Analyzer.THreshold VCC
                                      ; advise TRACE32 to use
                                      ; 1/2 VCC as threshold level
                                      ; for the trace signals
Analyzer.AutoFocus /NoTHreshold
                                      ; start the calibration
                                      ; advise TRACE32 to keep
                                      ; the threshold level
WAIT 1.s
                                      ; wait 1 second
Break
                                      ; stop the program execution
```

A manual setup is required if your target is using a gated clock. Refer to "Manual Setup" in AutoFocus User's Guide, page 18 (autofocus_user.pdf) for assistance.

After a successful configuration of the off-chip tracing the following command can be used to inspect the STP packet stream:

STMAnalyzer.List

Display STP packet stream recorded to POWER TRACE II / POWER TRACE III.

B::STMAnalyzer.L	ist							×
🌽 Setup 🔒 G	oto 🎁 Find	Chart 📕	Profile	MIPS	More	Less		
record ru	n address		cycle	data		symbol	ti.back	- 1
-000000364	C:0000	00000800000		B08C3BB0	8C1FB08C		0.010us	
-000000356	C:0000	00000800000	d64	8C67B08C	5CB08C57		0.010us	=
-000000322			master		80		0.170us	*
-000000321	C:0000	00000800000			00		<0.005us	-
-000000319	C:0000	00000800000			7F8C6FB0		0.010us	
-000000311		00000800000			57756D6E		0.010us	
-000000302	C:0000	00000800000			8C80B066		0.010us	
-000000294	C:0000	00000800000		79734957	756D6E7E		0.010us	
-000000285	C:0000	00000800000			80B0664F		0.010us	
-000000277		00000800000			6D6E7E75		1.835us	
-000000268		00000800000		7F7F8C80	B0664F79		0.010us	
-000000260	C:0000	00000800000	d64	49577560	6E7E757F		0.010us	
-000000226			master		80		0.170us	
-000000225	C:0000	00000800000			00		<0.005us	
-000000223	C:0000	00000800000		7F8C80B0	664F7973		0.010us	
-000000215		00000800000			7E757F7F		0.010us	
-000000206		00000800000			4F797349		0.010us	
-000000198		00000800000			757F7F7F		0.010us	
-000000189		00000800000			79734957		0.010us	
-000000181		00000800000			7F7F7F8C		1.725us	
-000000172		00000800000			73495775		0.010us	
-000000164	C:0000	00000800000	d64	6E7E757F	7F7F8C80		0.010us	
-000000130			master		80		0.170us	
-000000129		00000800000			00		<0.005us	
-000000127	C:0000	00000800000	d64	664F7973	4957756D		0.010us	-
								in 1922

The Intel[®] PT based core traces can be displayed by the following command:

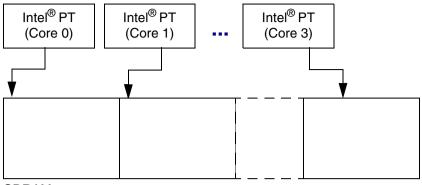
Analyzer.List

Display all core trace information decoded out of the STP packet stream.

	Goto	nd 🚺 🔂 Chart 🛛 🧮 F		More		
record	run address		cycle data	symbol	ti.back	
-0027244056		esi,+0x12 0x8048CD2 P:000000008048CE0) ptrace	\sieve+0x51	0.015us	
700	0 0 0 add	} ar dword ptr [ebp-	nzahl++; -0x10],+0x1			-
689	0 0 add 0 cmp 0 jle	<pre>for (i = 0 ; i <= ebx,+0x1 ebx,+0x12 0x8048CBC</pre>	= SIZE ; i++)			
691	0 0 movzx 0 test 0 je	{ if (flags eax,byte ptr [e al,al 0x8048CE4				
693	0 0 0 lea	eax,[ebx+ebx]	rimz = i + i + 3;			
694	0 lea 0 lea	edi,[eax+0x3] k esi,[edi+ebx]	= i + primz;			
695	0 0 jmp		nile (k <= SIZE)			

SDRAM Trace

If the Intel[®] PT trace information is routed to SDRAM, a fixed amount of memory is assigned to each core. The max. SDRAM size per core is currently 4 MByte.



SDRAM

Configure TRACE32

1. Advise TRACE32 to read the trace information from SDRAM.

Trace.METHOD Onchip

TRACE32 reads the onchip trace via JTAG.

2. Provide further details on the SDRAM configuration to TRACE32.

Onchip.Buffer IPT	; inform TRACE32 that the SDRAM ; provides Intel [®] PT trace ; information
Onchip.Buffer BASE 0x5000000	; inform TRACE32 that the SDRAM ; allocated for Intel [®] PT trace ; starts at address 0x5000000
Onchip.Buffer SIZE 0x1000000	; inform TRACE32 that the SDRAM ; allocated for Intel [®] PT trace has ; a size of 16 MByte

METHOD			0
O Analyzer O CA	nalyzer (Onchip (ART CLOGGER SNOOP Integrate	r © FDX © LA or ○ Probe ○ IProbe
state	used	ACCESS TDelay	
OISable		auto 🔻	TrOnchip
OFF	16777216.		- <u>PT</u>
Arm	- SIZE	CLOCK	📉 🚫 BMC
C TRIGGER	16777216.		
O break			
	Mode	Buffer	
commands	Fifo	C LBR	
RESet	Stack	O BTS	
🛛 🛇 Init	🔘 Leash	● IPT	
SnapShot		BASE	
List	SLAVE	0x5000000	
🗹 AutoArm		- SIZE	
🔲 AutoInit		0x1000000	
SelfArm			

3. Enable Intel[®] PT on the target and allow TRACE32 to configure it.

IPT.ON

If the command **Onchip.List** is used, TRACE32 merges the $Intel^{\ensuremath{\mathbb{R}}}$ PT traces from the individual cores as follows:

SDRAM block core 0	SDRAM block core 1	SDRAM block core 2	SDRAM block core 3
Intel [®] PT packet 1			
Intel [®] PT packet 2	Intel [®] PT packet 2	Intel [®] PT packet 2	
Intel [®] PT packet 3	Intel [®] PT packet 3	Intel [®] PT packet 3	
Intel [®] PT packet 4	Intel [®] PT packet 4	Intel [®] PT packet 4	
Intel [®] PT packet 5	Intel [®] PT packet 5	Intel [®] PT packet 5	
	Intel [®] PT packet 6	Intel [®] PT packet 6	
	Intel [®] PT packet 7		

Onchip.List

Intel [®] PT packet 1 of core 0
Intel [®] PT packet 1 of core 1
Intel [®] PT packet 1 of core 2
Intel [®] PT packet 1 of core 3
Intel [®] PT packet 2 of core 0
Intel [®] PT packet 2 of core 1
Intel [®] PT packet 2 of core 2
Intel [®] PT packet 3 of core 0

This procedure will change with the decoding of synchronisation packets.

Onchip.List

; cores

Onchip.List /CORE 1

; display trace listing for core 1

p seu	p) 📭 (Goto)	Find)	Chart	Profile	MIPS	More	Less		
rec	run ad			cycle	data	symbol	10000000000			
	2 mo		esi,[eb	p-0x4]						
	2 mo		ebx,[eb							
	2 mo		esp,ebp							
	2 po 2 re		ebp						T	
0005	2 re		E1 • 40078	550 ptrace					A	
	3 2 pu		ebp							
	3 mo		ebp,esp							
	3 pu		edi							
	3 pu		esi							
	3 pu		ebx	0.007						
	3 le		esp,[es	p-0x2CJ						
	3 mo		ebx, [eb							
	3 te 3 ie		ebx,ebx 0x40078							
	3 mo		esi,[eb:							
	3 mo		edi,esi	~1						
	3 mo		edx, esi							
	3 an		edi,0x2	000						
	3 an	d	edx,0xC							
	3 mo	v	[ebp-0x]							
	3 jn	e	0x40078	5C2						
		ence								
	3 mo		eax,0xF	FFFFFF					E	
		terrupt							1000	
0004	0			6C8 ptrace			al\apic_t			
0003	1	NP:02.	ID:CIA80	6C8 ptrace		.ux\GIO	pal\apic_t	imer_inter	rrupt	
									P	
	」 ∢			And a second second second second						
]•									
مال] ∢	10000001						wanna an		
- B::Oi] ≮ nchip.List	/CORE 1.								
a.a.a.161.05		and a state of the second	Find	M Chart	Profile	MIPS	♠ More	X Less		
setu	nchip.List	Goto]	Find)	Chart cycle	Profile	MIPS	Amore	Less		
setu	p] 📭 (Goto]	ebx,[eb	cycle p-0x0C]	and the second se	and the second s	◆ More	Less		
setu	p A	Goto] dress v	ebx,[eb	cycle p-0x0C] p-0x8]	and the second se	and the second s	◆ More	Less		
setu	p A	Goto]	ebx,[eb esi,[eb edi,[eb	cycle p-0x0C] p-0x8] p-0x4]	and the second se	and the second s	◆ More	Less		
setu	p A G	Goto]	ebx,[eb esi,[eb edi,[eb esp,ebp	cycle p-0x0C] p-0x8] p-0x4]	and the second se	and the second s	◆ More	Less		
setu	p ad	Goto)	ebx,[eb esi,[eb edi,[eb	cycle p-0x0C] p-0x8] p-0x4]	and the second se	and the second s	◆ More	Less		
Setu rec	p A G	Goto)	ebx,[eb esi,[eb edi,[eb esp,ebp ebp	cycle p-0x0C] p-0x8] p-0x4]	data	symbol				
setu	p run add mo mo mo po re	Goto j dress	ebx,[eb esi,[eb edi,[eb esp,ebp ebp 1D:C14A0	cycle p-0x0C] p-0x8] p-0x4]	data	symbol			dirqs_on_thunk	
Setu rec	p Pun add run add mo mo mo po re	Goto)	ebx,[eb esi,[eb edi,[eb esp,ebp ebp 1D:C14A0 edx	cycle p-0x0C] p-0x8] p-0x4]	data	symbol				
Setu rec	p ad	Goto	ebx,[eb esi,[eb edi,[eb esp,ebp ebp 1D:C14A0 edx ecx	cycle p-0x0C] p-0x8] p-0x4]	data	symbol				
Setu rec	p A mo mo mo mo mo re po re	Goto	ebx,[eb esi,[eb edi,[eb esp,ebp ebp 1D:C14A0 edx	cycle p-0x0C] p-0x8] p-0x4]	data	symbol				
Setu rec	p ad	Boto	ebx, [eb esi, [eb edi, [eb esp, ebp ebp 1D:C14A0 edx ecx eax	cycle p-0x0C] p-0x8] p-0x4]	data	symbol	ux\Global\	trace_har		

Trace Errors

ERRORS

If the trace contains ERRORS, please try to set up a proper trace recording before you start to evaluate or analyze the trace contents.

ERRORS can be caused by the following:

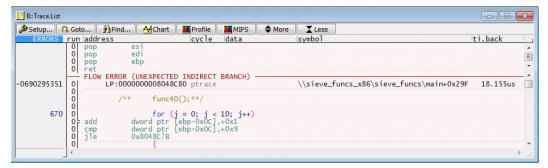
• TRACE32 detected an invalid trace packet. TRACE32 additionally displays the error indicator HARDERROR, if it is likely that the error was caused by pin problems.

B::Trace.List					
setup	🗘 Goto 🎁 Find 🚺 Chart	Profile	MIPS	More Less	
HARDERRORS	run address	cycle	data	symbol	ti.back
	0 add esi,+0x1 0 cmp esi,+0x2 0 jle 0x8048750 — FLOW ERROR (BAD TRACE	DACKET) -			× III
		PACKETY		m	\$ h. 4

• TRACE32 could not decode the packet.

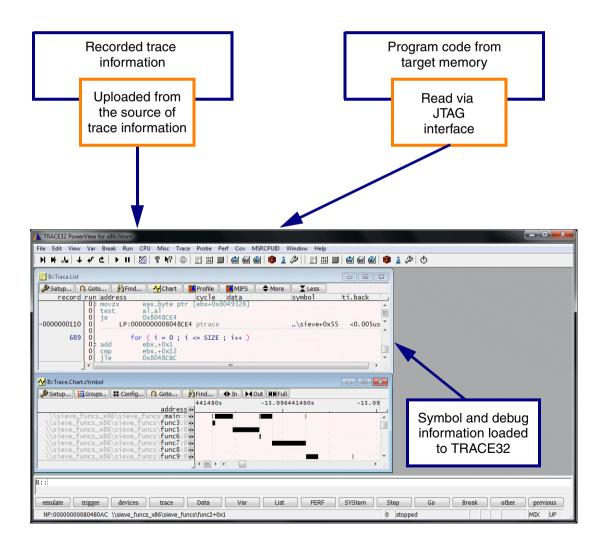
B::Trace.List				8
setup 👔	🕽 Goto	🎁 Find 🔂 Chart 🔛 Profile 🔛 MIPS 🔶 More	Less	
ERRORS	run a	ddress cycle data	symbol	
-0046160128	0 F	LOW ERROR (ERROR IN WRAPPER PROTOCOL)	\\sieve_funcs_x86\sieve_funcs\main+0x29F	A E
	000000000000000000000000000000000000000	/** func40();**/		*
670	0 c	for (j = 0; j < 10; j++) dd dword ptr [ebp-0x0C],+0x1 mp dword ptr [ebp-0x0C],+0x9 le 0x804867B		
672	0	sieve(); all 0x8048C8F ; sieve		-

The trace information is not consistent with the program code in the target memory.



Background: In order to provide an intuitive trace display the following sources of information are merged:

- The trace information recorded.
- The program code from the target memory read via the JTAG interface.
- The symbol and debug information already loaded to TRACE32.



The TRACE32 function Trace.FLOW.ERROR() returns the number of ERRORS as a hex. number.

PRINT %Decimal Trace.FLOW.ERRORS()	; display the number of ERRORS
	; as a decimal number in the ; TRACE32 PowerView Message Line
	, indelsz rowerview nebbage bine

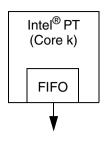
B:: 14 emulate							
emulate	trigger	devices	trace	Data	Var	List	PERF
NP:00000008048C8F \\sieve_funcs_x86\sieve_funcs\sieve							

To find ERRORS in the trace use the keyword FLOWERROR on the Expert page of the Trace Find dialog.

		$\overline{}$								
B::Trace.List										- • ×
Setup	Goto		Chart	CVC IC	MIPS data	And the More	Less symbol	ti.back		
-0030656653	FLOW	/ ERROR (ER		RAPPER PRO	DISCOL)	🛐 Trace Find				
	0	/**	func40)();**/		Expert	© Cycle	Group	Changes	© Up
670	0 add	dwo	for (j	<pre>= 0; j < ebp-0x0C] ebp-0x0C]</pre>	10; j++	_ items			🔘 Signal	Own
	0 cmp 0 jle	0x8	3048C7B	[eph-oxoc]	,+023	FLOWERR	OR			-
672	0 call	8x0	3048C8F	sieve(); s	ieve					
	0 }									
	0	flags[SIZ	Έ+1];			Find Next	Find First	Find Here Find	All Clear	Cancel
681	0 { 0 push)							-
] < .									h. ₹

Trace.Find FLOWERROR

Inside each Intel[®] PT generation module trace packets are queued to a FIFO buffer in order to send them out to the STM/SDRAM.



If trace packets are generated faster than can be sent out, the FIFO buffer can overflow and trace packets are lost.

The affected Intel[®] PT generates a Buffer Overflow packet (FUP.OVF) to indicate that its FIFO is full and trace packets are no longer generated.

A Asynchronous Flow Update packet, that provides the address of the next instruction that will be executed, is generated to indicate that the packet generation now continues.

The TRACE32 function **Trace.FLOW.FIFOFULL()** returns the number of TARGET FIFO OVERFLOWs as a hex. number.

PRINT %Decimal Trace.FLOW.FIFOFULL() ; display the number of TARGET ; FIFO OVERFLOWs as a decimal ; number in the TRACE32 ; PowerView Message Line

To find TARGET FIFO OVERFLOWs in the trace use the keyword FIFOFULL on the Expert page in the **Trace Find** dialog.

Fi Trace Find				
Expert	Cycle	O Group	Changes Signal	UpDown
– items –				
FIFOFULL				•
Find Next	Find First	Find Here Find	All Clear	Cancel

TARGET FIFO OVERFLOWs are strictly speaking not errors. They can occur in normal operation.

Since gaps in the instruction execution sequence are likely to disturb the nesting trace analyses, TRACE32 explicitly points them out.

SystemTrace

.

Depending on where the STP packets are stored, the following TRACE32 command groups can be used to analyze and display these packets:

• **STMAnalyzer.**<*sub_cmd*>, if the STP packets are stored in the trace memory provided by POWER TRACE II / POWER TRACE III.

B::STMAnalyzer.Li	st							×
Setup 📭 Go	oto 🎁 Find	Chart 🛛	Profile	MIPS	More	Less		
record ru	n address		cycle	data		symbol	ti.back	- 1
-000000364	C:0000	00000800000		B08C3BB0	08C1FB08C		0.010us	
-000000356	C:0000	00000800000	d64	8C67B080	SCB08C57		0.010us	
-000000322			master		80		0.170us	
-000000321	C:0000	00000800000			00		<0.005us	-
-000000319		00000800000			7F8C6FB0		0.010us	
-000000311		00000800000			957756D6E		0.010us	
-000000302		00000800000			8C80B066		0.010us	
-000000294	C:0000	00000800000			756D6E7E		0.010us	
-000000285	C:0000	00000800000		7F7F7F80	80B0664F		0.010us	
-000000277	C:0000	00000800000		73495775	56D6E7E75		1.835us	
-000000268		00000800000)B0664F79		0.010us	
-000000260	C:0000	00000800000	d64	49577560	06E7E757F		0.010us	
-000000226			master		80		0.170us	
-000000225		00000800000			00		<0.005us	
-000000223		00000800000			0664F7973		0.010us	
-000000215	C:0000	00000800000	d64		27E757F7F		0.010us	
-000000206	C:0000	00000800000	d64	8C80B066	54F797349		0.010us	
-000000198		00000800000			2757F7F7F		0.010us	
-000000189		00000800000			79734957		0.010us	
-000000181		00000800000			57F7F7F8C		1.725us	
-000000172		00000800000			973495775		0.010us	
-000000164	C:0000	00000800000	d64	6E7E757F	7F7F8C80		0.010us	
-000000130			master		80		0.170us	-
-000000129		00000800000			00		<0.005us	
-000000127	C:0000	00000800000	d64	664F797	34957756D		0.010us	-
								11. A

STMLA.<*sub_cmd*>, if the STP packets were recorded without a TRACE32 trace tool, and if they were loaded to TRACE32 PowerView for analysis.

The command groups usable in your current configuration can be get from the TRACE32 PowerView Softkey line.

B::		•					
emulate trigger devices	trace	Data	Var	List	other	previous	
NP:000000008048975 \\sieve_funcs_x86\sieve_funcs\	unc25	0 st	opped			MIX UP	

Push **trace** to get access to all command groups that analyze trace information.

B::			
Trace COVerage ISTATistic MIPS CTS ETA SystemTrace	other	previous	
NP:000000008048975 \\sieve_funcs_x86\sieve_funcs\func25 0 stopped		MIX UP	зđ
			

Push other to see more command groups.

B:: STMTrace PrintfTrace akeMoreTrace Analyzer NP:000000000449975 \\sieve_funcs_x86\sieve_funcs\funcs\func25	
STMTrace PrintfTrace akeMoreTrace Analyzer	STMAnalyzer CAnalyzer CIProbe other previous
NP:000000008048975 \\sieve_funcs_x86\sieve_funcs\func25	0 stopped MIX UP
	•

POWER TRACE II / POWER TRACE III is used in the current configuration, so the command group STMAnalyzer is enabled.

B::		
Integrator Port LA STMLA		previous
NP:000000008048975 \\sieve_funcs_x86\sieve_funcs\func25	0 stopped	MIX UP

The command group STMLA is always enabled.

TRACE32 PowerView offers the following abstraction, since most *<sub_cmd>* are identical for all command groups:

SystemTrace.METHOD Analyzer LA	
SystemTrace.METHOD Analyzer	; inform TRACE32 PowerView that the ; STP packets are stored in POWER ; TRACE II
SystemTrace.List	; List STP packet stream

Depending on where the trace packets are stored, the following TRACE32 command groups can be used to analyze and display the core trace information:

 Analyzer.<sub_cmd>, if the STP packets are stored in the trace memory provided by POWER TRACE II / POWER TRACE III.

🌽 Setup 🛛 💦	Goto	Find Chart	Profile	MIPS	More	Less		
record	run addre		cycle	e data		symbol	ti.back	
0021526081 368	0 jle 0 mov 0 jmp 0 cmp 0 jle	0x8048583 NP:0000000080 for (v3 = 0 esi,0x0 0x8048599 esi,+0x2 0x8048594				\func10+0x3D	<0.005us	, m
369	0 0 add	v17 + ebx.esi	= v3;					
368	0 0 add 0 cmp 0 jle	for (v3 = 0 esi,+0x1 esi,+0x2 0x8048594		v3++)				
369	0	v17 +	= v3;					

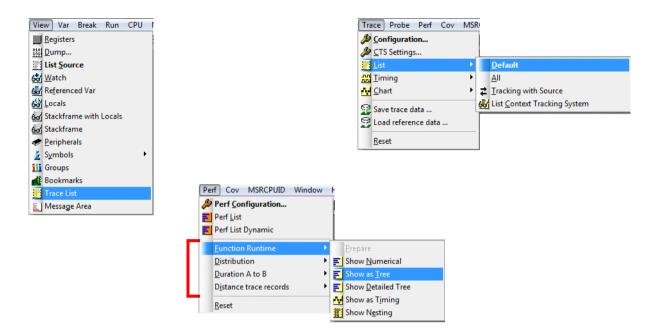
- **Onchip.**<*sub_cmd*>, if the Intel[®] PT trace packets are stored in the target SDRAM.
- **LA.***<sub_cmd>*, if the trace packets were recorded without a TRACE32 trace tool, and if they were loaded to TRACE32 PowerView for analysis.

TRACE32 PowerView offers the following abstraction, since most *<sub_cmd>* are identical for all command groups:

Trace.METHOD Analyzer | Onchip | LA Trace.METHOD Analyzer ; inform TRACE32 PowerView that the ; trace packets are stored in POWER ; TRACE II Trace.List ; List core trace information

Selecting the trace METHOD has the following additional consequences:

All **Trace**.<*sub_cmd>* commands offered in the TRACE32 PowerView menu apply to the selected trace METHOD.



TRACE32 is advised to use the trace information from the trace specified by METHOD as source for the trace evaluations of the following command groups:

COVerage. <sub_cmd></sub_cmd>	Trace-based code coverage
ISTAT. <sub_cmd></sub_cmd>	Detailed instruction analysis
MIPS. <sub_cmd></sub_cmd>	MIPS analysis

Influencing Factors on the Trace Information

The main influencing factor on the trace information is the **Intel[®] PT**. It specifies what type of trace information is generated for the user.

Basics about the trace messages are described in "Protocol Description", page 6.

Advanced setting can be found in "Trace Control by Filters", page 68.

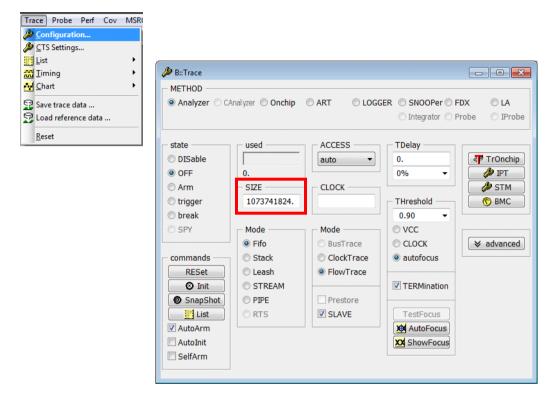
Another important influencing factor are the settings in the **TRACE32 Trace Configuration** window. They specify how much trace information can be recorded and when the trace recording is stopped.

Settings in the TRACE32 Trace Configuration Window

The Mode settings in the Trace Configuration window specify how much trace information can be recorded and when the trace recording is stopped.

The following modes are provided, if the Trace.METHOD Analyzer is selected:

 Fifo, Stack, Leash Mode: allow to record as much trace records as indicated in the SIZE field of the Trace Configuration window.



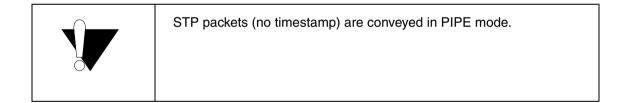
STREAM Mode: STREAM mode specifies that the trace information is immediately streamed to a file on the host computer. STREAM mode allows a trace memory size of several T Frames.

• **PIPE Mode:** PIPE mode specifies that the trace information is immediately streamed to a named pipe on the host computer.

PIPE mode creates the path to convey trace raw data to an application outside of TRACE32 PowerView. The named pipe has to be created by the receiving application before TRACE32 can connect to it.

Trace.Mode PIPE	
Trace.PipeWrite <pipe_name></pipe_name>	Connect to named pipe
Trace.PipeWrite \\.\pipe\ <pipe_name></pipe_name>	Connect to named pipe (Windows)
Trace.PipeWrite	Disconnect from named pipe

...
Trace.Mode PIPE ; switch trace to PIPE mode
Trace.PipeWRITE \\.\pipe\pproto00 ; connect to named pipe
; (Windows)
...
Trace.PipeWRITE ; disconnect from named pipe



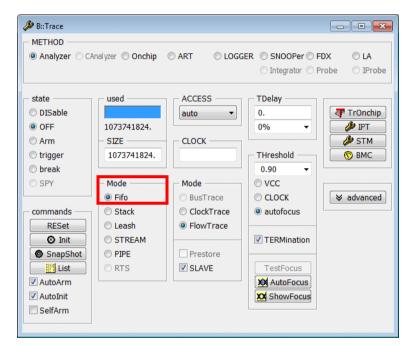
If the Trace.METHOD Onchip is selected only Fifo mode can be used:

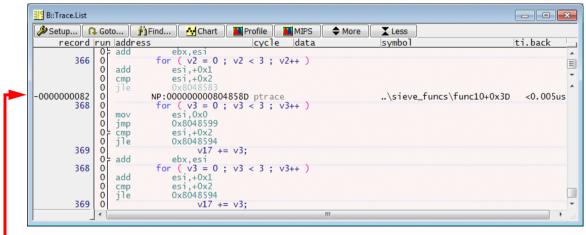
• **Fifo:** allows to record as much trace records as indicated in the **SIZE** field of the Trace Configuration window.

Tra	ce Probe Perf Cov	MSR
ø	Configuration	
Þ	CTS Settings	
	<u>L</u> ist	- +
-55 77	<u>T</u> iming	
M	<u>C</u> hart	- +
P	Save trace data	
9	Load reference data	
	<u>R</u> eset	

🎾 B::Trace			-	- • •
METHOD				
O Analyzer O CA	nalyzer Onchip 	◎ ART ◎ LOG	GER © SNOOPer © FDX © Integrator © Prob	
state	used	ACCESS	TDelay	
OISable		auto 🔻		TrOnchip
OFF	16777216.		· · · · · · · · · · · · · · · · · · ·	/P T
Arm	- SIZE	CLOCK		🚫 ВМС
TRIGGER	16777216.			
O break				
	Mode	Buffer		
commands —	Fifo	C LBR		
RESet	Stack	O BTS		
🛛 🛛 Init	C Leash	IPT		
SnapShot		BASE		
List	SLAVE	0x5000000		
AutoArm		- SIZE		
AutoInit		0x1000000		
SelfArm				

Trace.Mode Fifo ; default mode ; when the trace memory is full ; the newest trace information will ; overwrite the oldest one ; the trace memory contains all ; information generated until the ; program execution stopped

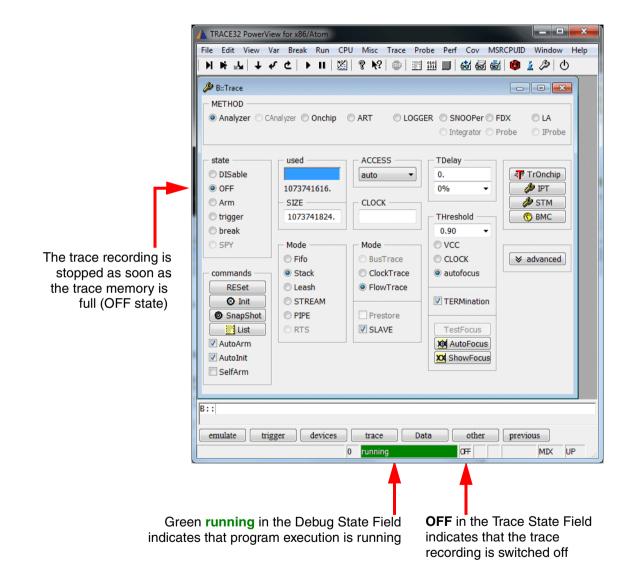




In Fifo mode negative record numbers are used. The last record gets the smallest negative number.

Trace.Mode Stack

; when the trace memory is full ; the trace recording is stopped ; the trace memory contains all ; information generated directly ; after the start of the program ; execution



TRACE32 needs to read the program code from the target memory in order to display the core trace information. This is not possible while the program execution is running. This is the reason why the **Trace.List** window indicates **NOACCESS**.

			Chart	Profile	MIPS	More	Less	
NOACCE	SS run add	ress		cycle	data		symbol	
								11.

Stop the program execution to allow TRACE32 to read the program code from the target. Or if you need to display the core trace information while the program execution is running, load a copy of the program code to the **TRACE32 Virtual Memory**.

Data.LOAD.Elf < <i>file</i> > /PlusVM	Load the program code to the target and to the TRACE32 Virtual Memory.

Since the trace recording starts with the program execution and stops, when the trace memory is full, positive record numbers are used in **Stack** mode. The first record in the trace gets the smallest positive number.

Setup 🚺	🕽 Goto 🧃	Find	Chart	Profile	MIPS	More	Less		
record	run addre	ss		cyc	le data		symbol		ti.back
+000000073	0	NP:000	00000080	48B5F ptra	ace		sieve_	funcs\main+0x17E	
638	0 0 0 mov 0 call	dw Ox	func1 ord ptr 8048793	1 <mark>(5);</mark> [esp],0x5	func11				
438	0 int x 0 {					/* multipl	e returns	; */	
439	0 push 0 mov 0 cmp	switc dw	p,esp h (x) ord ptr	[ebp+0x8]	,+0x6				
+000000074	0 cmp 0 ja 0 0 mov	NP:000	80487E0 00000080 x,[ebp+0	4879C ptra	ace		sieve_	_funcs\func11+0x9	<0.005u

Trace.Mode Leash

- ; when the trace memory is nearly ; full the program execution is ; stopped
- ; Leash mode uses the same record ; numbering scheme as Stack mode

A TRACE32 PowerVi	ew for x86/Atom			
File Edit View V	/ar Break Run CF	U Misc Trace Pr	obe Perf Cov MS	RCPUID Window Help
H H H H + 4	ר¢ ► 🕅	🔋 🕅 🚳 🧮	III 🔳 🐼 🗟 🚮	😢 🯄 🌽 🕛
B::Trace				
	nalyzer 🔘 Onchip (ART OLOGG	ER 🔘 SNOOPer 🔘 FE	
			Integrator O Pr	
state	used	ACCESS	- TDelay	
© DISable		auto 💌	0.	TrOnchip
OFF	966373376.		0% -	/ IPT
Arm	SIZE	CLOCK		STM
🔘 trigger	1073741824.		THreshold	S BMC
🔘 break			0.90 🔻	
© SPY	Mode	Mode	© VCC	
	Fifo	O BusTrace	CLOCK	
commands	Stack	ClockTrace	autofocus	
RESet	Leash	FlowTrace		
Ø Init	STREAM		TERMination	
SnapShot		Prestore		
List	© RTS	SLAVE	TestFocus	
AutoArm			AutoFocus	
AutoInit			X ShowFocus	
SelfArm				
B::				
emulate trig	gger devices	trace Di	ata Var	other previous
NP:00000008048CC3	\\sieve_funcs_x86\sieve_func	s∖⊧ 0 stopped		MIX UP

The program execution is **stopped** as soon as the trace buffer is nearly full.

Since stopping the program execution when the trace buffer is nearly full requires some logic/time, **used** is smaller then the maximum **SIZE.**

Trace.Mode STREAM	; stream the recorded trace ; information to a file on the host ; computer
	; STREAM mode uses the same record ; numbering scheme as Stack mode

The trace information is immediately streamed to a file on the host computer after it was placed into the trace memory. This procedure extends the size of the trace memory to several T Frames.

• STREAM mode requires a 64-bit host computer and a 64-bit TRACE32 executable to handle the large trace record numbers.

By default the streaming file is placed into the TRACE32 temp. directory (**OS.PresentTemporaryDirectory()**).

The command **Trace.STREAMFILE** *<file>* allows to specify a different name and location for the streaming file.

Trace.STREAMFILE d:\temp\mystream.t32

; specify the location for ; your streaming file

TRACE32 stops the streaming when less then 1 GByte free memory is left on the drive by default.

The command Trace.STREAMFileLimit <+/- limit in bytes> allows a user-defined free memory limitation.

```
Trace.STREAMFileLimit 5000000000. ; streaming file is limited to
; 5 GByte
Trace.STREAMFileLimit -5000000000. ; streaming is stopped when less
; the 5 GByte free memory is left
; on the drive
```

Please be aware that the streaming file is deleted as soon as you de-select the STREAM mode or when you exit TRACE32.

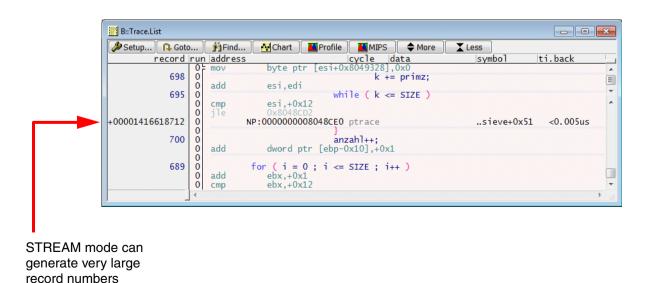
At high data rates your host computer might have problems saving the trace data to the streaming file. The command **Trace.STREAMCompression** allow to configure a better compression.

Trace.STREAMCompression HIGH

In STREAM mode the used field is split:

Number of records buffered by the trace memory of POWER TRACE II / POWER TRACE III B::Trace.state - • • METHOD Analyzer O CAnalyzer O Onchip O ART ○ LOGGER ○ SNOOPer ○ FDX O LA Integrator O Probe O IProbe state used ACCESS TDelay OISable auto 0. TrOnchip 4717104448. / IPT 0% • Arm SIZE STM STM CLOCK THreshold C BMC Trigger O break 0.90 • Mode Mode ○ vcc SPY C Fifo ○ СГОСК BusTrace commands Stack ClockTrace autofocus RESet Leash ElowTrace Ø Init STREAM TERMination Prestore SnapShot PIPE List O RTS SLAVE TestFocus AutoArm AutoFocus AutoInit XX ShowFocus SelfArm

Number of records saved to streaming file



STREAM mode can only be used if the average data rate at the trace port does not exceed the maximum transmission rate of the host interface in use. Peak loads at the trace port are intercepted by the memory in POWER TRACE II / POWER TRACE III, which can be considered to be operating as a large FIFO.

If the average data rate at the trace port exceeds the maximum transmission rate of the host interface in use, a **PowerTrace FIFO Overrun** occurs. TRACE32 stops streaming and empties the POWER TRACE II / POWER TRACE III FIFO. Streaming is re-started after the POWER TRACE II / POWER TRACE III FIFO is empty.

A PowerTrace FIFO Overrun is indicated as follows:

1. A I in the **used** area of the Trace Configuration window indicates an overrun of the POWER TRACE II / POWER TRACE III FIFO.

✤ B::Trace				- • •
METHOD				
Analyzer O CA	nalyzer 🔘 Onchip	○ ART ○ LOGG	ER 🔘 SNOOPer 🔘 I	FDX 🔘 LA
			🔘 Integrator 🔘 I	Probe 💿 IProbe
– state –––––	used	ACCESS	- TDelay	[]
OISable		auto 👻	0.	TrOnchip
OFF	1318846464.!		0% -	🥬 IPT
Arm	- SIZE	СГОСК		🥬 STM
© trigger			THreshold	🛞 BMC
Dreak			0.90 -	
SPY	Mode	Mode	© VCC	
L	Fifo	O BusTrace	© CLOCK	Sector Secto
commands	Stack	ClockTrace	autofocus	
RESet	🔘 Leash	FlowTrace		
🛛 🛇 Init	STREAM		TERMination	
SnapShot	O PIPE	Prestore		
List	© RTS	SLAVE	TestFocus	
🗷 AutoArm			AutoFocus	
🗷 AutoInit			X ShowFocus	
SelfArm				
L				

2. The OVERRUN is indicated in all trace display windows.

B::Trace.List				
🖉 Setup 🕻	🕻 Goto 🏾 🎁 Find 🖉 Chi	art 📕 Profile 📕 MIPS 🔷 N	Aore Less	
OVERRUN	run address	cycle data	symbol	ti.back
689	0 je 0x8048C 0 for (i = 0 add ebx,+0x 0 cmp ebx,+0x 0 jle 0x8048C	0 ; i <= SIZE ; i++) 1 12		* *
+0588212177		08048CBC ptrace	\sieve+0x2D	21.475s
691	0 { 0 if 0 movzx eax,byt 0 test al,al 0 je 0x8048C	(flags[i]) e ptr [ebx+0x8049328] E4		
	0 ft			→ ±

OVERRUNs are not visible at record level.

A large ti.back value (tool timestamp only) can be considered as an OVERRUN indicator.

Trace.FindAll 7	FIme.Back	10.s50.s	;	find all trace records with
			•	a timestamp between 10.s and 50.s

The trace buffer can either sample or allows the read-out for information display.

METHOD				
Analyzer O CA	Analyzer 💿 Onchip	○ ART ○ LOGG	ER OSNOOPer OI	_
			🔘 Integrator 🔘 I	Probe 🔘 IProbe
state	used	ACCESS	TDelay	
DISable		auto 🔻	0.	TrOnchip
OFF	249932328.		0% -	🥬 IPT
Arm	- SIZE	CLOCK		STM 🥬
🛇 trigger	1073741824.		THreshold	🛞 ВМС
🔘 break			0.90 -	
SPY	Mode	Mode	© VCC	
	Fifo	O BusTrace	CLOCK	Sector Secto
commands	© Stack	ClockTrace	autofocus	
RESet	🔘 Leash	FlowTrace		
🛛 🛛 Init	© STREAM		TERMination	
SnapShot	© PIPE	Prestore		
List	© RTS	SLAVE	TestFocus	
AutoArm			AutoFocus	
🗹 AutoInit			XX ShowFocus	
SelfArm				

States of the Trace	
DISable	The trace is disabled.
OFF	The trace is not sampling. The trace contents can be analyzed and displayed.
Arm	The trace is sampling. There is no access to the trace contents.

The current state of the trace is always indicated in the Trace State field of the TRACE32 state line.

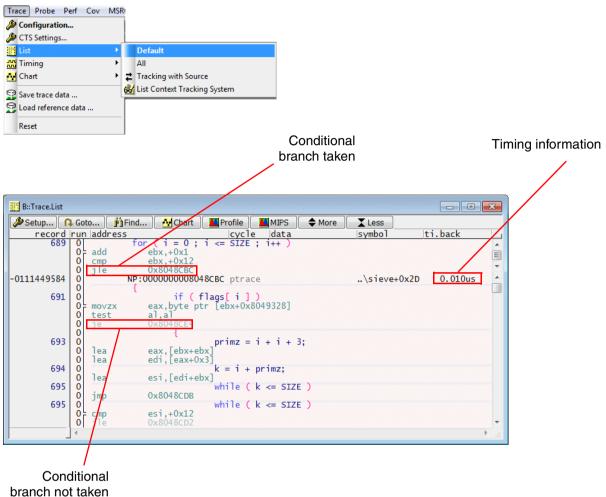
8::		
emulate trigger devices trace Data Var List PERF SYStem	other	previous
running	Trace	HLL UP
	DISable	
	OFF	
	 ARM 	

Since Intel[®] PT does not provide a mean to indicate a trigger, the Trace states **trigger** and **break** are never reached.

METHOD				
Analyzer O CA	Analyzer 🔘 Onchip	© ART © LOGO	SER 🔘 SNOOPer 🔘 F	
			🔘 Integrator 🔘 F	Probe 🔘 IProbe
state	used	ACCESS	TDelay	
OISable		auto 👻	0.	TrOnchip
OFF	249932328.		0% -	🥬 IPT
Arm	- SIZE	CLOCK		🥔 STM
trigger	1073741824.		THreshold	🚫 BMC
🔘 break			0.90 👻	
SPY SPY	Mode	Mode	© vcc	
	Fifo	BusTrace	CLOCK	Sector Secto
commands	Stack	ClockTrace	autofocus	
RESet	🔘 Leash	FlowTrace		
🛛 Init	C STREAM		TERMination	
SnapShot	© PIPE	Prestore		
List	© RTS	SLAVE	TestFocus	
🗹 AutoArm	L		AutoFocus	
🗹 AutoInit			XX ShowFocus	
SelfArm				

Init Button	Clear the trace memory. All other settings in the Trace configuration window remain valid.
AutoInit CheckBox	ON: The trace memory is cleared whenever the program execution is started (Go, Step).

Default Listing



(pastel printed)

The trace information for all cores is displayed by default in the **Trace.List** window. The column run and the coloring of the trace information are used for core indication.

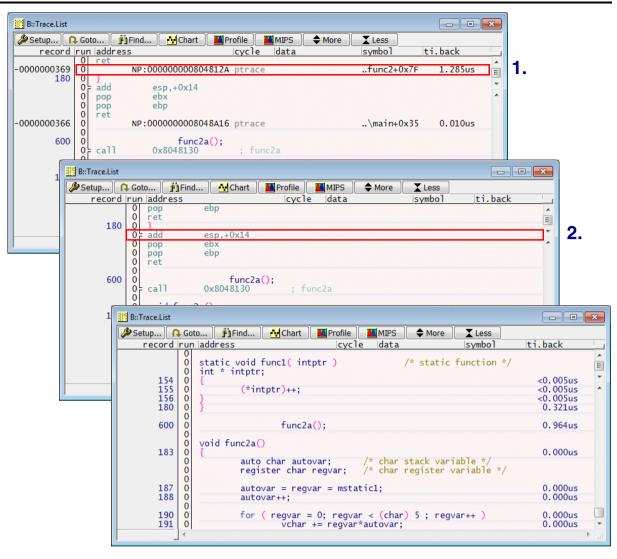
B::Trace.List						
setup 📭	Goto]	Find Chart Profile	MIPS 🗧 More	Less	
	run	addr		cycle data		ti.back
-000000234	1		XP:FFFF:FFFFFFF82843EA3	ptrace	\\vmlinux\spinlock_raw_spin_unlock+0x33	<0.005us 🔺
	1		raw_spin_unlock(loc	k);		E
170	1	}				-
	1 li	pop	rbp			
-000000231		ret	XP:FFFF:FFFFFFF820EDC65	ntrace	\\vmlinux\chip\handle_edge_irq+0x85	0.040us
541		3	XP.1111.1111111020EDC03	perace	((with thick (chilp (hand te_edge_th d+oxos	0.04003
511	11	pop	rbx			
	1	pop	r12			
	1	pop	rbp			
000000000707		ret	VD. FFFF. FFFFFFFF. 2067080))];)[]()	
-000000207	ŏ		XP:FFFF:FFFFFFF820C7988	ptrace	\\vmlinux\kernel/smp\generic_exec_single+0x78	
			cpuid(op, &eax, &ebx,	Secy Sedy).		
	ŏ		cpuru(op; dcax; dcbx;	acca, acua,,		
	0		return edx;			
	0	}				-
						<u>ار</u> ا
-						

Trace.List /CORE <n>

The option CORE allows a per core display of the trace information.

B::Trace.List /COR	E0.		×
🌽 Setup 🔃 🔒 Go	to 🎁 Find 🚰 Chart 📕 Profile 📕 MIPS 🗢 More 🗶 Less		
record r	in laddress cycle data symbol	ti.back	
	* csd_lock/csd_unlock used to serialize access to per-cpu csd resources		
	* * For non-synchronous ipi calls the csd can still be in use by the * previous function call. For multi-cpu calls its even more interesting * as we'll have to ensure no other cpu is observing our csd. */		• •
	static void csd_lock_wait(struct call_single_data *csd)		
105 -0000000207	{	8 2.680us	
	<pre>cpuid(op, &eax, &ebx, &ecx, &edx); return edx; }</pre>		

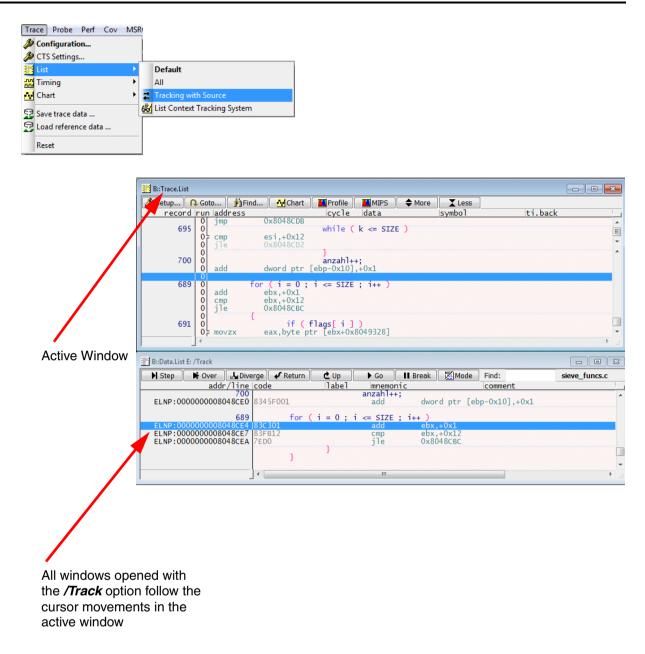
Basic Formatting



1. time Less	Suppress the display of the ptrace packets.
2. time Less	Suppress the display of the assembly code.

The More button works vice versa.

Correlating the Trace Listing with the Source Listing



Tracking between the Trace Listing and the Source Listing is based on the program addresses.

Browsing through the Trace Buffer

B::Trace.List	Profile MIPS	Less symbol ti.back
0 ret -0000002000 0 NP:0000000080 0 650 0 func1		main+0x1DE 0.010us
0 mov dword ptr	Next L	mark Goto irst Trigger Zero ast Ref Track Cancel

Pg↑	Scroll page up.
Pg ↓	Scroll page down.
Ctrl - Pg ↑	Go to the first record sampled in the trace buffer.
Ctrl - Pg ↓	Go to the last record sampled in the trace buffer.

The **Trace.List** window provides a "**Find...**" button to open the **Trace Find** dialog. The **Trace Find** dialog allows to search for events of interest in the trace.

B::Trace.List													3
🖉 Setup 🚺	🕽 Got) 🎁	Find	Chart	Profile) 🔳 M	IIPS 🔵 🗢 M	1ore	Less				
record					сус	le d	ata		symbol		t	i.back	
-000000082	0	add cmp jle	e 0	bx,+0x1 bx,+0x12 x8048CBC 000000080					\sieve	_funcs\sie	eve+0x2D		• •
691		movzx test je	a	ax,byte p 1,al x8048CE4	flags[i tr [ebx+0	x80493	Fi Trace Fir		Oycle	🔘 Group	Chang	Jes 🔿 Up	3
689		add cmp jle	e	(i = 0 ; bx,+0x1 bx,+0x12 x8048CBC	i <= SIZ	E ; i+	– address /	expre/	ession		© Signal	● Down▼	
691	0	movzx test je	a	if (ax,byte p 1,al x8048CE4	flags[i tr [ebx+0]) x80493	Cycle —		▼ Data			•	
693	0	lea	e	ax,[ebx+e	primz bx]	= i +	Find Next	t) (Fir	nd First Fin	d Here Fin	d All Cle	ar Cancel]

Example: Find the entry to the function func10.

B::Trace.List							C	
	Goto Find <u>run address</u> func10() 0 {	Chart		MIPS 🔶 More data	Less symbol]	ti.ba	*
360	0	ebp ebp,esp esi eps egister i, j; egister v1, v egister v9, v L7 = 0:	/2, v3, v4, v5 /10, v11, v12,	Trace Find	Ocycle	© Group	Changes	Up O Down
361	0 mov	ebx,0x0 or (i = 0; esi,0x0 0x8048566 esi,+0x2 0x8048561	i < 3 ; i++)	address / exp func10	Data		•	
362 361	0 add 0 add 0 cmp	v17 += ebx,esi or (i = 0 ; esi,+0x1 esi,+0x2	: i; i < 3 ; i++)	Find Next	Find First Fin	d Here) Find	All Clear	Cancel

A detailed description of the **Trace Find dialog** can be found in "**Application Note for Trace.Find**" (app_trace_find.pdf).

Default Display Items

B::Trace.List								×
🌽 Setup 🚺					MIPS And Ante	Less		
record			is	cycle	data	symbol	ti.back	
-0111449333	0	ret	NP:0000000080	048B5A ptrace		main+0x179	1.175us	Ê
636	000	call	func1 0x8048550	1 <mark>0();</mark> ; func1	.0			*
355	0000	func10 {						
	000000000000000000000000000000000000000	push mov push push	ebp, ebp,esp esi ebx register i, j register v1, register v9,	; v2, v3, v4, v5 v10, v11, v12,	5, v6, v7, v8; v13, v14, v15,	v16, v17;		-

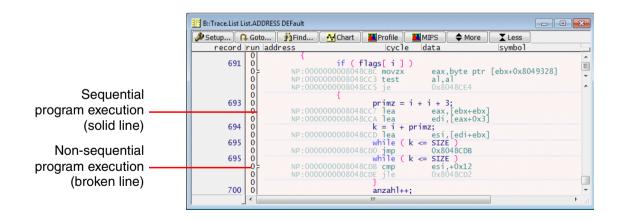
Column record

Displays the record numbers

• Column run

The column run displays some graphic element to provide a quick overview on the instruction execution sequence.

Trace.List List.ADDRESS DEFault



The column run also indicates Interrupts and TRAPs.

B::Trace.List		
🖉 Setup 📭	Goto) 👔 Find) 🧮 Mrofile) 🔛 MIPS) 🗢 More) 🗶 Less	
		ti.back 🔄
105	0 static void csd_lock_wait(struct call_single_data *csd) 0 { while (csd->flags & CSD_FLAG_LOCK) 0 test byte ptr [r12+0x20],0x1 interrupt	
-0193607871	0 xP:FFF:FFFFFF8284AD80 ptrace \\vmlinux\Global\apic_timer_interrupt 0 nop reboot_interrupt smp_reboot_interrupt	0.120us
	0 #endif 0 #ifdef CONFIG_X86_UV 0 apicinterrupt UV_BAU_MESSAGE \ 0 uv_bau_message_intr1 uv_bau_message_interrupt	•
		E. ◀

• Column cycle

The main cycle type is:

- ptrace (program trace information)

• Column address/symbol

B::Trace.List		×
🌽 Setup 🚺	🕽 Goto 🏥 Find 🛛 🚰 Chart 🛛 🌉 Profile 🤇 🌉 MIPS 🛛 🜩 More 🛛 🗶 Less	_
record	run address cycle data symbol ti.back	
-0111449333		-
636	0 0 0- call 0x8048550 ; func10	T A
355	0 0 func10() 0 {	
	0; push ebp 0 mov ebp,esp 0 push esi 0 push ebx 0 register v1, v2, v3, v4, v5, v6, v7, v8; 0 register v9, v10, v11, v12, v13, v14, v15, v16, v17; 	-

The address column shows the following information:

<access class>:<address>

Access Classes					
NP	Program address in 32-bit Protected Mode				
ХР	Program address in 64-bit mode				

Information on the other available access classes can be found in "Intel® x86/x64 Debugger" (debugger_x86.pdf).

The **symbol column** shows the corresponding symbolic address.

Setup	G Goto	Find	Chart	Profile	MIPS	More	Less		
						more	Less		
	d address		0000040605	cycle	symbol		10.0.175	ti.back	
-002649157			000804868E		\sieve_f			<0.005us	•
-002649157			00080486A6		\sieve_f			<0.005us	E
-002649156			00080486C1	ptrace	\sieve_f			<0.005us	-
-002649156 -002649156			00080486D9		\sieve_f			0.010us <0.005us	
-002649156			00080486F4 000804870C		\sieve_f			<0.005us	
-002649156			0008048700	ptrace	\sieve_f			<0.005us	
-002649156					\sieve_f			<0.005us	
-002649156			000804873F 000804875A	ptrace	\sieve_f			<0.005us	
-002649156			0008048754		\sieve_f			<0.005us	
-002649156			0008048772		\sieve_f				
-002649155			000804878D		86\sieve			<0.010us	
-002649155			0008048590		86\sieve			<0.005us	
-002649155			00080487E3		6\sieve			<0.005us	
-002649155			0008048B6B		86\sieve			► 0.010us	
-002649154			0008048839		6\sieve_				
-002649152			0008048836		6\sieve			0.740us	
-002649152			0008048836		6\sieve				
-002649151			0008048836		6\sieve_				
-002649151			0008048B87	ptrace	86\sieve			<0.005us	
-002649151			0008048B93		86\sieve			<0.005us	-
				P Mu u					

The **ti.back column** shows the time distance to the previous timestamped record.

For details on the TRACE32 tool timestamp refer to "Tool Timestamp (POWER TRACE II / POWER TRACE III only)", page 8.

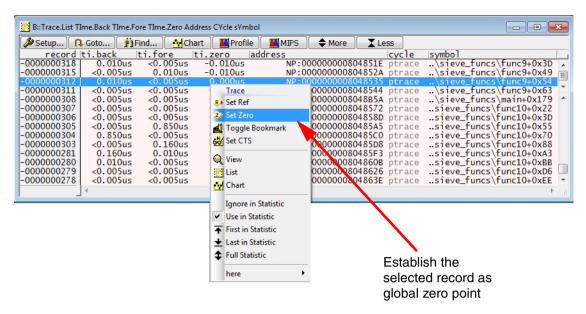
Time Information

Time.Back Time relative to the previous record (red)	
Time.Fore	Time relative to the next record (green).
Time.Zero	Time relative to the global zero point.

Trace.List TIme.Back TIme.Fore TIme.Zero Address CYcle sYmbol

B::Trace.List 1	Elme.Back TIm	ne.Fore TIme.Zer	o Address CYcle s	Ymbol							×
setup (🕽 Goto	Find	Chart 🛛 🔼 Pr	ofile MIPS	♦ More	Le	SS				
record	ti.back	ti.fore	ti.zero	address			cyc1	e sy	/mbol		
-000000332					8000000000		ptra		\sieve_funcs\		
-0000000329					000000008		ptra		\sieve_funcs\		
-0000000326	0.010				000000008		ptra		\sieve_funcs\		
-0000000325					000000008		ptra		sieve_funcs		
-0000000322					000000008		ptra		sieve_funcs		
-0000000318					000000008				sieve_funcs		
-0000000315					000000008				sieve_funcs		
-0000000312					000000008				<pre>\sieve_funcs</pre>		
-0000000311	<0.005				0000000008				<pre>\sieve_funcs</pre>		
-0000000308					0000000008				sieve_funcs		
-0000000307	<0.005				8000000000				sieve_funcs\f		
-0000000306					000000008				sieve_funcs\f		
-0000000305	<0.005				0000000008				sieve_funcs\f		
-0000000304	0.850				8000000000				sieve_funcs\f		
-0000000303	<0.005				8000000000				sieve_funcs\f		
-0000000281	0.160				8000000000				sieve_funcs\f		
-0000000280					8000000000		ptra		sieve_funcs\f		
-0000000279	<0.005				0000000008		ptra		sieve_funcs\f		
-0000000278	<0.005				8000000000				sieve_funcs\f		
-0000000277	<0.005				8000000000				eve_funcs\fi		
-0000000276	<0.005				8000000000				ieve_funcs\fi		
-000000275	<0.005				8000000000		ptra		eve_funcs\fi		
-000000274	<0.005				8000000000				eve_funcs\fi		
-000000273	<0.005	us 0.010	us 1.020	us NP:	8000000000	0486C1	ptra	ce	ieve_funcs\fu	unc10+0x1/1	<u> </u>
	•)	h. d

Set the Global Zero Point (PowerTrace II only)



ZERO.RESet (tool timestamp only)	TIme.Zero is the zero point of the timestamp counter commonly used by all TRACE32 hardware modules.
ZERO.offset <time></time>	TIme.Zero is the zero point of the timestamp counter commonly used by all TRACE32 hardware modules minus the specified <i><time></time></i> .

```
PRINT Trace.RECORD.TIME(-99.); print the timestamp of<br/>; record -99.ZERO.offset Trace.RECORD.TIME(-99.); specify the time of record<br/>; -99. as global zero point
```

• Cycle Accurate Tracing Pros.

Provides how much core clocks it took to execute a program section.

Allows to synchronize traces from different trace sources if Time Synchronization packets are available (not implemented yet).

• Cycle Accurate Tracing Cons.

Cycle accurate tracing requires up 2 times more bandwidth.

Cycle accurate tracing and changing core clock while recording

Cycle accurate tracing has to be enabled in the IPT configuration window.

🥔 В::ІРТ		
ipt OFF ON	selection TraceOS TraceUSER	
commands RESet Ø CLEAR	CR3	
Register Trace	SuperTimeSync	

IPT.CycleAccurate ON

Advise Intel[®] PT to generate cycle count information.

; advise TRACE32 to display a trace listing with ; cycle count information (CLOCKS.Back) ; advise TRACE32 to suppress the display ; of the timestamp information (TIme.Back.OFF) Trace.List CLOCKS.Back DEFault TIme.Back.OFF

The following command allows to specify this display as default for the Trace.List window.

SETUP.ALIST CLOCKS.Back DEFault TIme.Back.OFF

B:: Warning: target clock frequency unknown		
trigger devices trace Data	Var	List
NP:000000008048CD9 \\sieve_funcs_x86\sieve_fu	ncs\sieve+0	к4А

TRACE32 displays the warning above when the recorded trace information is analyzed and displayed the first time. This warning points out that all displayed time information (TIme.Back, TIme.Zero) might be inaccurate.

🎾 Setup 🛛 🔼 Go				MIPS	More	Less	
clo		un address		cycle	data		symbol
000000303	11.	D add cmp jle LF	ebx,+0x1 ebx,+0x12 0x8048CBC :000000008048	CEC ptrace	2		sieve_funcs\sieve+0x5D
704	()))))))))))))	} return anzahl eax,[ebp-0	; x10]			
705) add) pop) pop) pop) pop) pop	esp,+0x10 ebx esi edi ebp				
000000299			000000008048	C80 ptrace	2		sieve_funcs\main+0x29F
	(/** func4	0();**/			
670		5	for (dword ptr dword ptr 0x8048C7B	j = 0; j < [ebp-0x0C] [ebp-0x0C]	: 10; j++ ,+0x1 ,+0x9)	
•							

Cycle count information relative to the previous record

; advise TRACE32 to display a trace listing with the decoded trace packet ; (TPINFO) $% \left(\left(\left(TPINFO\right) \right) \right) \right)$

Trace.List TPINFO DEFault

B::Trace.List TI	PINFO DEFault		
🌽 Setup 🔒	Goto Find Chart	Profile MIPS 🗢 More 🗶 Less	
	tpinfo	run address cycle data	symbol
-0000000150 216	Target IP 2b BLIP=0x8A25	0 test al,al 0 jne 0x80481E0 0 LP:000000008048214 ptrace 0 } 0 leave	\func2c+0x5F
	TNT a=TTTTTN	0 ret 0 LP:000000008048A25 ptrace	cs\main+0x44
606	•	0 func2d(); 0 call 0x8048216 ; func2d m	

There are several ways for a belated trace analysis:

1. Save a part of the trace contents into a file (ASCII, CSV or XML format) and analyze this trace contents outside of TRACE32 PowerView.

B::PRinTer		- • •
© printer	type WIN (Windows Default)	•
© ClipBoard	ClipType ASCIIE (ASCII ENHANCED) FileType	•
FILE	ASCIIE (ASCII ENHANCED) ASCIIE (ASCII ENHANCED) ASCII (ASCII) ASCIIP (ASCII) ASCIIP (ASCII PAGED)	-
	CSV (Comma Separated Value) WS (WORDSTAR) WSX (WORDSTAR SPECIAL) XML (Extensible Markup Language) PSP10 (POSTSCRIPT/P COURIER 10cpi) PSP12 (POSTSCRIPT/P COURIER 12cpi) PSP20 (POSTSCRIPT/P COURIER 12cpi) PSL10 (POSTSCRIPT/L COURIER 10cpi) PSL12 (POSTSCRIPT/L COURIER 12cpi) PSL12 (POSTSCRIPT/L COURIER 12cpi) PSL12 (POSTSCRIPT/L COURIER 12cpi) PSL12 (POSTSCRIPT/L COURIER 12cpi) PSL20 (POSTSCRIPT/L COURIER 12cpi) PSL20 (POSTSCRIPT/L COURIER 12cpi) PSP510 (POSTSCRIPT/P SWISS 10cpi) PSP512 (POSTSCRIPT/P SWISS 15cpi) PSP110 (POSTSCRIPT/P SWISS 15cpi) PSP112 (POSTSCRIPT/P TIMES 12cpi) PSP115 (POSTSCRIPT/P TIMES 12cpi) PSP115 (POSTSCRIPT/P TIMES 15cpi)	

- 2. Save the trace contents in a compact format into a file. Load the trace contents at a subsequent date into a TRACE32 Instruction Set Simulator and analyze it there.
- 3. Export the STP byte stream to postprocess it with an external tool.

Saving a part of the trace contents to an ASCII file requires the following steps:

1. Select **Printer Setting...** in the **File** menu to specify the file name and the output format.

File Edit View Var Break Run Cf Image: Search for Script Image: Search for Script Image: Search for Script Image: Search for Script		
Open File Load File Type File Dump File	B::PRinTer	type
Stop Command Printer Settings	© ClipBoard	ClipType ASCIIE (ASCII ENHANCED)
Window Print Window Screenshot to File x exit	I FILE	FileType ASCIIE (ASCII ENHANCED) C:\T32_Atom\test_run.lst

PRinTer.FileType ASCIIE	; specify output format ; here enhanced ASCII
PRinTer.FILE test_run.lst	; specify the file name

2. It might make sense to save only a part of the trace contents into the file. Use the record numbers to specify the trace part you are interested in.

TRACE32 provides the command prefix **WinPrint.** to redirect the result of a display command into a file.

```
; save the trace record range (-8976.)--(-2418.) into the ; specified file 
WinPrint.Trace.List (-8976.)--(-2418.)
```

3. Analyze the result outside of TRACE32.

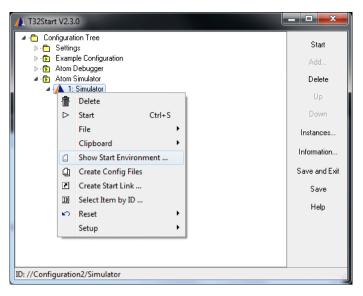
1. Save the contents of the trace memory into a file.

Trace Probe Perf Cov	MSRO		
🔑 Configuration			
🔑 CTS Settings			
List			
ᇌ Timing	- +		
Chart			
Save trace data Load reference data		B::Trace.SAVE	
		_ filename	
Reset		testrun1	browse
		range	mode
			▼ ◎ BusTrace
			FlowTrace
		III ZIP	
		Ok	Cancel

The default extension for the trace file is .ad.

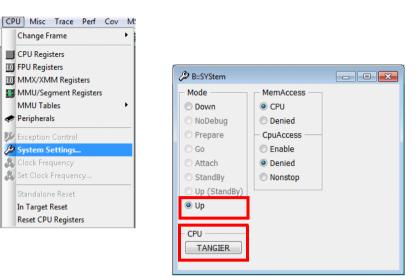
Trace.SAVE testrun1

2. Start a TRACE32 Instruction Set Simulator (PBI=SIM).



A Start Environment Viewer - Atom Simulator/Simulator		x
Batch Job		
C: cd C:\T32_Atom_Nightly C:\T32_Atom_Nightly\bin\windows64\t32mx64.exe -c C:\TMP\T32_1000148.t32		4 III +
4		F
T32 Configuration File C:\TMP\T32_1000148.t32		
;This configuration file is generated with T32Start2		*
;=======		
;Environment Variables		
OS=		
ID=T32_1000148		
TMP=C:\TMP SYS=C:\T32 Atom Nightly		
HELP=C:\T32 Atom Nightly\pdf		
hebr-c: (132_Atom_wightiy)pdi		
; Standard License File used		
:T32 API Access		=
; not used		
;T32 Intercom		
; not used		
;T32 GDB		
; not used		
Connection to Host		
PBI=SIM		
;Screen Settings SCREEN=		
HEADER=TRACE32 PowerView for Intel x86 0 [SIM 0]		
4		F.
		_
Edit History Settings Save Batch Job As Save Config As	Close	

3. Select your target CPU within the simulator. Then establish the communication between TRACE32 and the simulator.



4. Load the trace file.

Trace Perf Cov MSRCPUID		
Configuration	B::B::Trace.LOAD *	
CTS Settings List	C v v v v v v v v v v v v v v v v v v v	
쨆 Timing 🕨 🕨	Organize 🔻 New folder	:= - 🗌 🔞
Save trace data Load reference data Reset	 VirtualBox _CAE _A4902Logs _CAEPRG _Contacts _Desktop _DiaPortable _dosbox _Downloads _Favorites _Links _My Documents _ << _ 	No preview available.
	File name: testrun.ad 🗸 🗸	Current (*.ad)
	-	

Trace.LOAD testrun

5. Display the trace contents.

B::Trace.l	ist								×
🌽 Setup	Goto	Find	Chart	Profile	MIPS	More	Less		
record	run addres	55		cyc	le data		symbol	ti.back	
-00000136	0 add 0 cmp 0 jle 0 add 0 cmp 0 jle 0 add 0 add 0 add 0 cmp 0 jle 0 jle 0 jle 0 jle 0 jle 0 jle 0 jle 0 add 0 ada 0 ada 0 ada 0 ada 0 ada	0x8 byt: esi 0x8 LP:0 dwo ebx 0x8 ebx 0x8 eax al, 0x8 eax elx eax eax eax	+0x12)48CD2 e ptr [es: ,edi .40x12)48CD2)48CD2)000000008(rd ptr [el ,+0x1 ,+0x12)48CBC ,byte ptr		ace 0x1			0.015us	* • ·
LOAD								,	h.

LOAD indicates that the source for the trace information is the loaded file.

6. Load symbol and debug information if you need it.

Data.LOAD.Elf sieve_funcs_x86.elf /NoCODE

The TRACE32 Instruction Set Simulator provides the same trace display and analysis commands as the TRACE32 debugger.

Postprocessing of recorded trace information with the TRACE32 Instruction Set Simulator becomes more complex if an operating system that uses dynamic memory management to handle processes/task is used (e.g. Linux).
dynamic memory management to handle processes/task is used (e.g. Linux).

Script version

Save the trace contents in the recording TRACE32 instance:

```
Trace.SAVE testrun.ad
```

Prepare the TRACE32 Instruction Set Simulator for off-line processing of the trace information:

```
SYStem.CPU TANGIER
SYStem.Up
Trace.LOAD testrun.ad
Data.LOAD.Elf sieve_funcs_x86.elf /NoCODE
Trace.List
```

Trace.EXPORT.TracePort *<file>* Export trace raw data (no timestamps).

SystemTrace.EXPORT.TracePort mytest1.ad

Intel[®] PT provides 2 address ranges for trace control. The smallest range size is 4 bytes.

TRACE32 PowerView provides access to these address ranges by the action field in the Break.Set dialog.

🞯 B::Break.Set — address / expres	ssion ———		
			🕶 📑 🗆 HLL
type	options		
Program	EXclude	Temporary	auto 👻
ReadWrite	NOMARK	DISable	- action
Read		DISableHIT	stop 👻
Write	DATA		stop
O default			Spot
			Alpha Beta
Ok	Add	Delete	Charly
	, idd	Delete	Delta
			Echo
			TraceEnable
			TraceOFF

The 2 address ranges can be used for the following purposes:

TraceEnable advises Intel[®] PT to generate program flow information for the specified address range only.

TraceOFF advises $Intel^{\ensuremath{\mathbb{R}}}$ PT to stop the generation of program flow information as soon as a specified address range is reached.

Both filters are programmed to all Intel[®] PT in an SMP configuration.

Example 1: Advise Intel[®] PT to generate program flow information only for function func10.

1. Set a Program breakpoint to the address range of func10 and select the action TraceEnable.

🔯 B::Break.Set			
- address / expres	ssion		
func10			👻 🔢 🗹 HLL
- type	_ options		- implementation
Program	EXclude	Temporary	auto 👻
ReadWrite	NOMARK	DISable	- action
Read		DISableHIT	TraceEnable 👻
Write	DATA		
Ø default		-	Sector Secto
Ok	Add	Delete	Cancel

2. Start and stop the program execution.

3. Display the result.

TRACE ENABLE indicates the start of the message generation. It might be necessary to search for it.

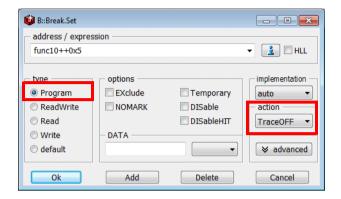
B::Trace.List											- • •
Setup	Goto	Find	Chart	Profile	MIPS	♦ More	Less				
<u> </u>	run add	ress		cycl	e data	sy	/mbol				ti.back
428	0 jle 0 add		x8048	Trace Find					- • •		<pre>A</pre>
427	0 0 add	for	bx,es (v15 si,+0 si,+0	Expert	Cycle	© Grou	р	Changes	🔘 Up		т А
	0 cmp 0 ile	e e	si,+0 x8048					Signal	Own		
-0000000113	0	NP:00	00000	items						10+0x222	0.010us
428		e	bx,es	TRACEENAB	IE						
427	0	for	(v15	TRACELIAD					· ·		
	0 add 0 cmp	e e	si,+0 └─ si,+0								
	0 j1e	i Ö	x8048								
428			v. bx,es								
427	0	for	(v15								
	0 add	e	si,+0 si,+0								
	0 cmp 0 ile	e 0			-						
429	0		(v16	Find Next	Find First	Find Here	Find A	ll Clear	Cancel		
_	•										ti ⊀

B::Trace.List				
🌽 Setup 🔒	Go	to 🏥 Find 🕂 Chart 🏼 🏙 Profile 🖉	MIPS 🔷 More 🛛 🗶 Less	
record	run	address cycle	data symbol	ti.back
-0006002364	0000	cmp esi,+0x2 jle 0x8048783 NP:00000000804878D ptrace	\\sieve_f	uncs_x86\sieve_funcs\func10+0x23D <0.005us
432	000	return v17; mov eax,ebx		Â
433	00000	pop ebx pop esi pop ebp TRACE ENABLE		
-0006002358	0	NP:000000008048550 ptrace	\\sieve_f	uncs_x86\sieve_funcs\func10 0.010us
355		<pre>func10() { push ebp mov ebp,esp push esi register i, j; register v1, v2, v3, v4, v register v9, v10, v11, v12</pre>	5, v6, v7, v8; , v13, v14, v15, v16, v17;	
360	0	v17 = 0;		-
	4			

```
Break.Delete /ALL
Var.Break.Set func10 /Program /TraceEnable
Go
…
Break
Trace.List
```

Example 2: Advise Intel[®] PT to stop the generation of program flow information as soon as function func10 is entered.

1. Set a Program breakpoint to the start address of func10 plus 5 bytes and select the action TraceOFF.



2. Start the program execution.

TRACE32 has, unfortunately, no way to detect that $\text{Intel}^{\$}$ PT stopped the generation of trace information.

Off-chip trace: Since the Analyzer is recording STP packets, **used** could increase, because other trace sources continue generating STP packets.

✤ B::Trace				- • •
METHOD				
Analyzer	CAnalyzer 🔘 Onchip	C ART C LOGG	ER 🔘 SNOOPer 🔘 FD	X 🔘 LA
			🔘 Integrator 🔘 Pro	obe 💿 IProbe
state	used	ACCESS	TDelay	
DISable		auto 👻	0.	TrOnchip
OFF	271464.		0% -	🥬 IPT
Arm	- SIZE	СLOCК		🌽 STM
🔘 trigger	1073741824.		- THreshold	🛞 BMC
🔘 break			0.90 -	

Onchip trace: TRACE32 can not read the filling level of the onchip trace while recording.

- 3. Stop the program execution.
- 4. Display the result.

🔢 B::Trace.List						3
🖉 Setup	Goto.	🛉 Find 🛃 Chart 🛛 🧮	Profile MIPS	♦ More Less		
	run a	ıddress	cycle data	symbol	ti.back	
-0000000140	0	le 0x80484F8 NP:000000008048544	ptrace	\\sieve_funcs_x86\sieve_funcs\func9	+0x63 <0.005us	-
	0	}				+
351	0 0 n	return &stat1 10v eax.0x8049028	: eax.stat1			
352	0 p 0 p	idd esp.+0x14 pop ebx pop esi pop ebp et				
-0000000136	ŏ.	NP:00000008048854	ptrace	\\sieve_funcs_x86\sieve_funcs\main+	0x179 0.010us	
636		func10();	; func10		=	H
<u> </u>	4					▼

Break.Delete /ALL

Break.Set func10++0x5 /Program /TraceOFF

Go

•••

Break

Trace.List

OS-aware tracing requires that OS-aware debugging is configured. For more information refer to "OS-aware Debugging" (glossary.pdf).

Process Switch Packets

x86/x64 processors have a CR3 control register that contains the Process Context Identifier (PCID). On every context switch the corresponding PCID is loaded to CR3.

Intel[®] PT generates a Paging Information Packet (PIP) when a write to CR3 occurs.

B::Trace.List TP	INFO List.TASK DEFault			- <u>×</u>
🌽 Setup 🔒	Goto) 👘 Find) 🛃 Chart) 🌉 Pro			
	tpinfo ru	un address	cycle data symbol ti.back	
		L mov L pop L pop L pop L ret	eax,r12d r12 r13 rbp	* 4 11 *
40		L #ifdef CO L mov	<pre>DNFIG_SMP this_cpu_write(cpu_t]bstate.state, TLBSTATE_OK); dword ptr gs:[0x128C8],0x1</pre>	
38		L cmp je	<pre>if (likely(prev != next)) { r12,r13 0xFFFFFFF82847FA2</pre>	
-0985525675 -0985525669 41	PIP CR0.PG=1 CR3=0x385D2000 TNT a=NTNTNT	L	<pre>owner FFF880038452A30 <</pre>	
70		} lock bts	<pre>} else {</pre>	Ŧ
] <			E de

TRACE32 names the cycle type owner if the PCID loaded to CR3 can be assigned to a process.

The command **TASK.List.tasks** can be used the check all assignments currently known to TRACE32. The **traceid** represents the PCID in this display.

B::TASK.List.tasks			
magic name	id	space	traceid
FFFF8800385C6DB0 vold:223	223.	183.	0x00B7 385FF000 🔺
FFFF880039AA4380 vold:224	224.	183.	0x00B7 385FF000
FFFF8800384521C0 nvm_server	188.	188.	0x00BC 385EC000
FFFF880038452A30 netd	189.	189.	0x00BD 385D2000
FFFF88002FD74BF0 netd:517	517.	189.	0x00BD 385D2000
FFFF88002FD74380 netd:518	518.	189.	0x00BD 385D2000
FFFF88002FD71950 netd:521	521.	189.	0x00BD 385D2000
FFFF88002FD75460 netd:522	522.	189.	0x00BD 385D2000
FFFF88002FD76DB0 netd:523	523.	189.	0x00BD 385D2000
FFFF88002FD70000 netd:524	524.	189.	0x00BD 385D2000
FFFF8800384532A0 debuggerd	190.	190.	0x00BE 38485000
FFFF880038453B10 surfaceflinger	191.	191.	0x00BF 38601000
FFFF880038450000 surfaceflinger:Binder_1	231.	191.	0x00BF 38601000 -
			►

[B::Trace.List TF	PINFO List.TASK DEFault /Track]							×
	Goto	Profile N		Less				
	tpinfo	run addres		cycle	data	symbol	ti.back	
		1 pop 1 pop 1 ret	r13 rbp					*
38			rcx,qword	rev != next)) { ptr [rbp-0x40]				•
	contextid: 0000		rbx,r14 OxFFFFFFF	F82174C19				_
	PIP CR0.PG=1 CR3=0xCD9300 TNT a=NNNNNT	1	XP:FFFF:FFFFF CONFIG_SMP	FFF821747F3 ptrac	xt 000000000000009300	00 \flush_old_exec+0x303	<0.005u 0.040u	
40 41		1 1 mov	dword ptr	_cpu_write(cpu_t) gs:[0x12BC8],0x1 _cpu_write(cpu_t)				
11		1 mov		gs:[0x0],rbx "iq" ((u8)C	ONST_MASK(nr))	next),		
		1	} else {	: "memory");				
	1,							· .

TRACE32 names the cycle type context if the PCID loaded to CR3 can not be assigned to a process.

The fact that the PCID can not be assigned to a process results in the following,:

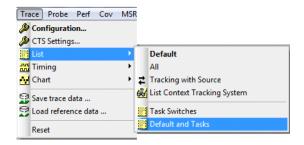
• Since TRACE32 does not require the PCID to decode trace information for the common address range, full trace decoding is possible.

• For all other address ranges a decoding of the trace information is not possible. The cycle type **unknown** is used for trace information that can not be decoded.

B::Trace.List Lis									1 23
🌽 Setup 🔒	Goto	Find Cha	art 📕 Profile	MIPS	More	Less			
	run add	ress		cycle	data	sy	mbol	ti.back	L
-1016945789	1	XP:FFFF:00	000000F770C9F2	unknown				<0.005us	
-1016945788	1	XP:FFFF:00	000000F770C9F2	unknown				<0.005us	E
-1016945786	1	XP:FFFF:00	000000F770C9F2	unknown				<0.005us	-
-1016945781	1	XP:FFFF:00	000000F76E7F00	unknown				0.040us	
-1016945780	1	XP:FFFF:00	000000F76E7F00	unknown				<0.005us	•
-1016945777	1	XP:FFFF:00	000000F76E81C2	unknown				<0.005us	
-1016945776	1	XP:FFFF:00	000000F76E81C2	unknown				0.040us	
-1016945775	1	XP:FFFF:00	00000F76E81C2	unknown				<0.005us	-
	1								. A

NOTE:The Real-Time Instruction Trace (RTIT), doesn't feature the process switching
packets. If multiple user space applications are traced, it is only possible to decode
the trace packets of the kernel. The cycle type unknown is used for the user space
trace packets. For decoding the trace packets of a user application, it is necessary to
filter the process of interest using the CR3 filter.
RTIT was implemented on very few devices, then it was extended to the Intel
Processor Trace which supports the process switching trace. The RTIT trace is
also covered by TRACE32 using the IPT command group.

Program Flow and Process Switches



B::Trace.List List	.TASK DEFault	X
	Goto ∯Find MChart Profile MIPS ♦ More Less	
-0987128735	<pre>1 cmp rsi,rdx 1 cmovbe rdx,rsi 1 /* tlb_flushall_shift is on balance point, details in commit log */ 1 /* tlb_flushall_shift is on balance point, details in commit log */ 1 if ((end - start) >> PAGE_SHIFT > act_entries >> tlb_flushall_shift) 1 shr edx,cl cmp rax,rdx 0 xFFFFFFF82034468 1 task: netd (FFFF880038452A30) 1 </pre>	•
-0987128729	<pre>XP:00BD:FFFFFF82034468 ptrace\flush_tlb_mm_range+0x1F8 <0.005us } static inline unsigned long native_read_cr3(void) { unsigned long val; asm volatile("mov %%cr3,%0\n\t" : "=r" (val), "=m" (force_order));</pre>	Ŧ
	4	

Trace.List List.TASK DEFault

- ; display trace listing with
- ; decoded task switch information

NOTE: This is a process switch analysis, since Paging Information Packets (PIP) only indicate process switches, but no thread switches.

magic	name	id	space		traceid
FFFF880038BCBB10	adbd:296	296.	241.	0x00F1	33EC9000
FFFF880038BCDCD0		297.	241.	0x00F1	33EC9000
FFFF880033F5D460	kdfrax	302.	0.	0x0000	030C6000
FFFF880033F5DCD0		524.	524.	0x020C	2F803000
FFFF88002F8290E0	system_server:GC	528.	524.	0x020C	2F803000
FFFF88002F82E540	system_server:Signal_Catcher	529.	524.	0x020C	2F803000
FFFF88002F829950	system_server:JDWP	530.	524.	0x020C	2F803000
	system_server:Compiler	531.	524.	0x020C	2F803000
		·			•

Threads do not have their own traceid

Perf Cov MSRCPUID Linu	x Win
Perf Configuration	
Perf List	
Perf List Dynamic	
Eunction Runtime	•
Distribution	•
Duration A to B	•
Distance trace records	•
Task Runtime	<u>P</u> repare
Task Eunction Runtime	Show <u>N</u> umerical
Reset	₩ Show as <u>T</u> iming
<u>never</u>	Tracking with Trace List

🎾 Setup 🚺 🚺 Groups 🔛 Config.	📭 Goto) 👘	Find	♦ In	HOu	t MMF	ull					
	-8.50	0 s			-8.0	00s		-	-7.50	0s		
r ange 🚯									1			
(unknown):0 💀												
(unknown):1 💀												
system_server:1												
nvm_server:1							<u>i</u>					
netd:1 💀												
contextid:0xCD70000:1												
contextid:0xCD93000:1												
swapper/0:1 💀						Sala and			11.1			
rild:1												
com.android.browser:1 🐻							- -					
			1000				•••••••••••••••••••••••••••••••••••••••					

contextid:<*trace_id*> indicates process switches for which the <*trace_id*> can not be assigned to a process.

Trace.Chart.TASK [/SplitCORE]	Display process time chart - graphical display - split the results per core - sort the results per recording order
Trace.Chart.TASK /MergeCORE	Display process time chart - graphical display - merge the results of all cores
	Trace information is analyzed independently for each core. The time chart summarizes these results to a single result.

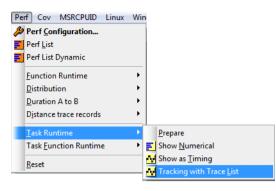
The recording time before the first Paging Information Packet (PIP) is assigned to the (unknown) task.

Perf Cov MSRCPUID Line	ux Win
Perf Configuration	
Perf List	
Perf List Dynamic	
Eunction Runtime	•
Distribution	•
Duration A to B	•
Distance trace records	•
Task Runtime	Prepare
Task <u>F</u> unction Runtime	Show <u>N</u> umerical
<u>R</u> eset	Show as <u>Timing</u> Tracking with Trace <u>List</u>

B::Trace.STATistic.TASK								~
Ø Setup III Groups III Co	nfig 🗾 Detai	led [Nesting	Chart	Profile				
	tasks: 18.	to	otal: 8.60	i9s				
range	total	min	max	avr	count	ratio%	1%	
(unknown):0	8.669s	-	8.669s	8.669s	0.	-	+	
(unknown):1	80.357ms	80.357ms	80.357ms	80.357ms	0.	-	+	8
system_server:1	5.725s	4.625us	5.383s	212.032ms	27.	66.036%	-	-
nvm_server:1	244.484ms	81.890us	242.902ms	40.747ms	6.	2.820%	-	-
netd:1	1.116ms	2.605us	267.280us	111.633us	10.	0.012%		E
contextid:0xCD70000:1	1.920ms	1.920ms	1.920ms	1.920ms	1.	0.022%		
contextid:0xCD93000:1	2.195ms	4.330us	2.064ms	548.821us	4.	0.025%	+	
swapper/0:1	2.579s	109.235us	492.556ms	184.240ms	14.	29.753%	-	-
rild:1	435.045us	117.705us	198.870us	145.015us	3.	0.005%		
com.android.browser:1	16.972ms	47.455us	8.422ms	4.243ms	4.	0.195%		
com.android.music:1	1.373ms	4.685us	351.215us	85.786us	16.	0.015%		
zygote:1	7.790ms	307.480us	3.727ms	1.948ms	4.	0.089%	+	S -
	4		111				100000000	

Trace.STATistic.TASK [/SplitCORE]	Process runtime statistic - numerical display - split the results per core - sort the results per recording order
Trace.STATistic.TASK /MergeCORE	Process runtime statistic - numerical display - merge the results of all cores Trace information is analyzed independently for each core. The statistic summarizes these results to a single
	result.

1. Open a process time chart window and a trace listing with decoded process switch information. Link both windows by using the /Track option.



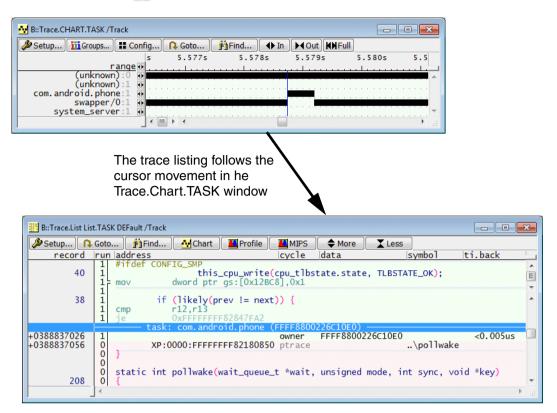
Trace.Chart.TASK /Track ; open process time chart ; window Trace.List List.TASK DEFault /Track ; open a default trace ; listing that includes ; process information ; both windows use the /Track option ; a window opened with the /Track option follows the cursor movement ; of the active window ; tracking between trace windows is based on the timestamp

; information

B::Trace.CHART.TASK /Track				
Setup 🚺 Groups 🔡 Conf	ig 📭 Goto	👘 Find 🔶 In	Out KNFull	
	000us	5.000s	10.000s	
r ange 🐽				
(unknown):0 💀				
(unknown):1 💀				
com. android. phone:1				
swapper/0:1				
system_server:1				-
	4 III + 4			E. A

🧾 (B::Trace.List Li	ASK DEFault /Track]	×
🌽 Setup 🔒	to 🚰 Find MChart 📕 Profile 📕 MIPS 🔶 More 🗶 Less	
record	n address cycle data symbol ti.back	
40	<pre>#ifdef CONFIG_SMP this_cpu_write(cpu_t]bstate.state, TLBSTATE_OK); mov dword ptr gs:[0x12BC8],0x1</pre>	1
38	<pre>if (likely(prev != next)) { cmp r12,r13 ie</pre>	
+0388837026 +0388837056	task: com.android.phone (FFFF8800226C10E0) owner FFFF8800226C10E0 <0.005us XP:0000:FFFFFFF82180850 ptrace\pollwake	C
208	} static int pollwake(wait_queue_t *wait, unsigned mode, int sync, void *key) {	
		Þ

2. Use the arrow keys \bullet of the process of interest to move to next state change.



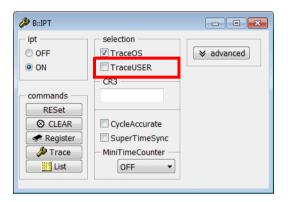
Filtering by Privilege Level

Intel[®] PT can be advised to generate program flow information only for:

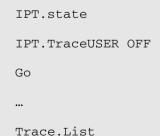
- privilege level 0
- all privilege levels greater than 0

Example: Advise Intel[®] PT to generate only program flow information for privilege level 0.

1. Uncheck TraceUSER in the IPT configuration window.



- 2. Start and stop the program execution.
- 3. Display the result.

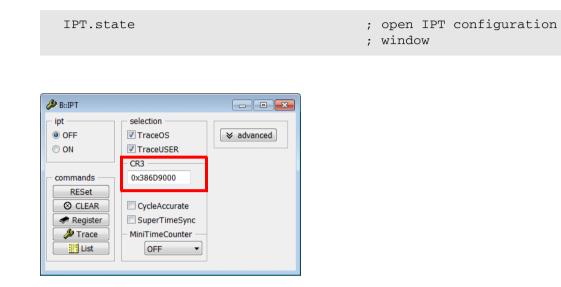


Filtering by Process

Intel[®] PT can be advised to generate program flow information only for a process of interest.

Example: Advise Intel[®] PT to generate only program flow information for the process "logcat".

1. Program the CR3 filter via the IPT window.



; specify the <trace_id> of the process "logcat" IPT.CR3 0x386D9000

; IPT.CR3 TASK.PROC.NAME2TRACEID("logcat")

TASK.PROC.NAME2TRACEID(*<process_name>*) Returns the *<trace_id>* of specified process.

- 2. Start and stop the program execution.
- 3. Display the result.

B::Trace.List										• X
🖉 Setup 🔒	Goto	(ři	Find Chart	Profile	MIPS	◆ More	Less			
	run	addres			cycle	data		symbol	ti.back	
1506	1 1 1 1	} pop pop xchg ret	trace_save_cr rbx rbp eax,eax	ndline(tsk	;					< III + 4
+0000028219	1	rec	XP:00D8:FFFFF	FF8210E6B	F ptrace			sched_switch+0x2F	13.194ms	
66	1 1 1 1	mov test je	if (!tracer_e eax,dword eax,eax 0xFFFFFFFF ENABLE	ptr [rip+	sched_st 0x10DC35E	opped)				
+0000028276	00000	TRACE	XP:00D8:FFFFF	: "iq"	((u8)~CC	NST_MASK(r X "btr %1,	nr)));	\schedule+0x31F		
	00	lock btr	[r13+0x2F(
	0 0 0 0		/* Re Toad_	e-load pag _cr3(next-	e tables >pgd);	*/				- -

TRACE ENABLE indicates the re-start of the program flow trace generation.

Please be aware that TRACE32 decodes all trace information for the process specified in the command **IPT.CR3** *<trace_id>*. Intel[®] PT does not generate Paging Information Packet (PIP) in this scenario.

Example: Advise Intel[®] PT to generate only program flow information when the function "logger_poll" is running in the context of the process "logcat".

1. Set a Program breakpoint to the address range of the function logger_poll and select the action TraceEnable.

🔯 B::Break.Set			
- address / expres	sion		
logger_poll			🔻 🔝 🗸 HLL
type Program ReadWrite Read Write	options	 Temporary DISable DISableHIT 	implementation auto action TraceEnable
Ø default		-	
Ok	Add	Delete	Cancel

2. Program the CR3 filter via the IPT window.

IPT.state

- ; open IPT configuration
- ; window



IPT.CR3 TASK.PROC.NAME2TRACEID("logcat")

- 3. Start and stop the program execution.
- 4. Display the result.

Setup 🔒	Goto 🛉 Find 🔥 Char	t Profile MIPS	🔷 More 🛛 🗶 Le	255	
record	run address	cycle	data	symbol	ti.back
662	1 ret 1 mutex_unloo 1 call 0xFFFFF	_off != reader->r_of : = POLLIN POLLRD :k(&log->mutex); :FF828464D0; mutex_u	NORM;		
+0000738672	1	FFFF8262CA67 ptrace		\\vmlinux\logger\logger_poll+0x87	<0.005us
665	1 return ret; 1 mov eax,ebx 1 pop rbx 1 pop r12 1 pop r13 1 pop rb1 1 ret				
+0000738686	TRACE ENABLE	FFFF8262C9E0 ptrace		\\vmlinux\logger\logger_poll	0.080us

TRACE ENABLE indicates the re-start of the program flow trace generation.

Postprocessing of recorded trace information with the TRACE32 Instruction Set Simulator requires complex preparations if an operating system that uses dynamic memory management to handle processes is used (e.g. Linux).

The following information has to be store after recording and re-loaded to the TRACE32 Instruction Set Simulator:

- The recorded trace information
- The whole kernel address space (code and data)
- The core registers
- All MMU-related registers
- The settings of the Debugger Address Translation (TRACE32 command group: TRANSlation)

Example for Linux

The **Generate RAM Dump** command in the **Linux** menu provides a store framework. It generates a **CMM file** that summarizes all commands for the TRACE32 Instruction Set Simulator.

Linux Window Help					
💑 Display Processes					
🖧 Display ps-like					
💑 Display Tasks					
💑 Display Modules					
💑 Display File System 🕨					
Process Debugging					
Module Debugging	A Store the trace for analys	is with the TRACE32 Sir	mulato	r 👝 🤅	
Library Debugging					
🔮 Symbol Autoloader 🔹 🕨	TRACE File Name:	trace.ad			
(2) Options	Binary Image :	image.bin			V
💑 Display Kernel Log	Kernel Logical Range:	0xFFFFFFF80000		0xFFFFFFF9FFFF	
Linux Terminal	Kernel Physical Range:	0x0		0x1FFFFFFF	
Configure Terminal					
🔐 Generate RAM Dump	Kernel Symbol File:	C:\T32_Atom_Night	tly∖Tar	igier_RTIT\tng-b0/v	
101 Generate roam bump					
📷 Linux Awareness Manual	Store Registers to	regs.cmm			
	Scan Translations to	translations.cmm			V
	CMM File:	start.cmm			
			2	Help STOR	E
	L				

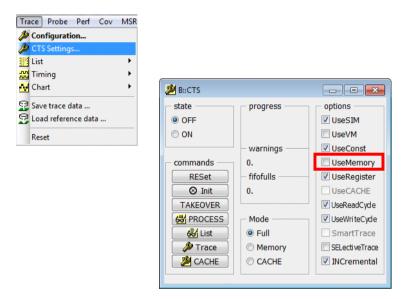
If you start a TRACE32 Instruction Set Simulator and run the generated script, the recorded trace information can be analyzed there. Please be aware that additional settings might be necessary e.g. the specification of the search paths for the C/C++ sources.

🎾 Setup 🛛 💦	Goto	Find	Chart	Profile	MIPS	More	Less			
<u>^</u>	run addr		· · · · · ·		cycle	data	sy	mbol	ti.back	
40	1 #ifd 1 1 mov	ef CONFI		pu_write(s:[0x12BC8	z pu_tlbs 3],0x1	tate.state,	TLBSTATE	_0K);		0
38	1 1 cmp 1 je	r (likely(pre 12,r13)xFFFFFFFF8 rild (FFFF	32847FA2						
0939133235 0939133246	1 0 0 mov 0 xor 0 mov 0 mov 0 call	XP:00 r e	000:FFFFFFF di,rbx ecx,ecx edx,0x1 esi,0x3 0xFFFFFFFF8	FA0086CAB	owner ptrace	FFFF880012	382A30		<0.005us	
	0 0 * @ 0 * @	q: the w mode: wh	vaitqueue nich thread	ls						0

Trace-based debugging allows to re-run the recorded program section within TRACE32 PowerView.

Setup

Since Intel[®] PT does not provide any information on read/write accesses, **UseMemory** has to be unchecked. A full explanation on this is given later in the chapter "**CTS Technique**", page 93.



CTS.UseMemory OFF

Specify the starting point for the trace re-run by selecting **Set CTS** from the Trace pull-down menu. The starting point in the example below is the entry to the function **activate_task** executed by core 1.

up 📭 🖓	oto 🛉 Find 🚺 Chart 🔛 Profile	MIPS 🔷 More	Less		
rı	in address	cycle data	symbol		ti.back
1322 1 1 1 1 1 1 1	i activate_task(rq, p, en mov edx,0x5 mov rsi,rbx mov rdi,r12 call 0xFFFFFFF820A08E0;	activate_task			
782 1	. void activate_task(struct rq *r	q, struct task_str	uct *p, int flag	S) Trace	
1	nop dword ptr [rax+rax+0	x0]		R+ Set Ref	
783 1	if (task_contributes_to mov rax, gword ptr [rsi]	_load(p))		Z. Set Zero	
1	mov rax,qword ptr [rsi]			Toggle Bookmark	
702	void activate_task(struct rq *r	q, struct task_str	uct *p, int flag	(S) 😸 Set CTS	
782 1	l push rbp				
783 1	if (task_contributes_to	_load(p))		Q View	
1	test al,0x2			List	
1	void activate_task(struct rq *r	q, struct task_str	uct *p, int flag	(s) Chart	
782 1				Ignore in Statistic	
	. indv i up,i sp			 Use in Statistic 	
-				First in Statistic	
				🛨 Last in Statistic	

Selecting **Set CTS** has the following effect:

• TRACE32 PowerView will use the preceding trace packet as starting point for the trace re-run.

B::Trace.List										×
Setup 🖡	Go	to)	🛱 Find 🚺 Chart	Profile	MIPS	♦ More	Less			
	run	addr	ess		cycle	data	S	ymbol	ti.back	
	1 1 1	pop pop ret	r13 rbp							× III
-0470020438	1		XP:FFFF:FFFFFF	F8284945F	ptrace			x\spinlock_raw_spin_lock+0x1F	<0.005us	-
138	1 1 1 1	} pop pop	raw_spin_loc rbx rbp	k(lock);						Ô
-0470020435	11	ret	XP:FFFF:FFFFFF	E820431CC	ntrace			ched/core\try_to_wake_up+0x17C	<0.005us	
1467	1		ttwu_do_activa					enea, cor c (cr y_co_nakc_aproxi; c		
	1	mo∨	rdi,r15							
	1	mov call	rsi,rbx 0xFFFFFFFF	20A1890;	twu_do_	activate.c	onstprop.	93		
1358	1 1 1 1 1 1		rbp,rsp			task_struc	t *p, int	wake_flags)		-
	1									H. 1000

• The TRACE32 PowerView GUI does no longer show the current state of the target system, but it shows the target state as it was, when the starting point instruction was executed. This display mode is called CTS View.

CTS View means:

- The instruction pointers of all cores are set to the values they had when the starting point instruction was executed.

- The content of the core registers of all cores is reconstructed (as far as possible) to the values they had when the starting point instruction was executed. If TRACE32 can not reconstruct the content of a register it is displayed as empty.

TRACE32 PowerView uses a yellow look-and-feel to indicate CTS View.

The Off button in the source listing can be used to switch off the CTS View.

▲ TRACE32 PowerView for Intel x86/x64	
File Edit View Var Break Run CPU Misc Trace Probe Perf Cov M	ISRCPUID Linux Window Help
N M ™ ↑ 4. 6 ▶ II 🕅 3. K5 ◎ 📰 🏢 🚳 🚳 🚳	M 🕲 1 🖉
₩ B::List /CORE 0	
	ver the Entry Off Mode Find: random.c
addr/line code label	mnemonic comment
575 w = rol32 xp:0000:FFFFFFF823B6868 83E703	(*bytes++, input_rotate & 31) ^ f->pool[i & 3] ^ A
576 f XP:0000:FFFFFF823B686B 4989F2	->pool[(i + 1) & 3]; mov r10,rsi
	(*bytes++, input_rotate & 31) ^ f->pool[i & 3] ^ add rdx,+0x1
AP:0000:FFFFFF623B8886E 4883C201	·
	m
B::Register.view /CORE 0	
CF _ RAX 28 RBX 000000204CCF858E	
PF P RCX B8 RDX FFFF88003F203C40 AF	
ZF Z R8 FFFF88003882C618 R9 FFFFB0A6 SF _ R10 3 R11 77A0	
TF R12 01AE R13 FFFF88003F20F760 IF R14 1 R15 FFFF88003D2C0828	
DF _ RBP FFF88003F203C78 RSP FFFF88003F203C30	
OFRFLAGS 46 RIP FFFFFF823B686B	
· · · · · · · · · · · · · · · · · · ·	
B:List /CORE 1	
→ Step → Over → Diverge → Return 2 Up → Step → Over addr/line code label	er the Entry Off Mode Find: core.c
1466 raw_spin_lock(&rq- XP:0000:FFFFFFF820A31C4 4C89FF	mov rdi,r15
XP:0000:FFFFFF820A31C7 1467 1467 ttwu_do_activate(r	<pre>call 0xFFFFFF82849440 ; _raw_spin_lock rq, p, 0);</pre>
XP:0000:FFFFFF820A31CC 4C89FF XP:0000:FFFFFF820A31CF 4889DE	mov rdi,r15 mov rsi,rbx
XP:0000:FFFFFF820A31D2 E889E6FFFF 1468 raw_spin_unlock(&r	call 0xFFFFFFF820A1890 ; ttwu_do_activate.c
B::Register.view /CORE1	B::Trace.List
CF RAX RBX	Setup Coto Setup Coto Setup Chart Profile MIPS More Less
PF RCX RDX AF RSI D480 RDI	run address cycle
ZF R8 R9 SF R10 R11	138 1raw_spin_lock(lock);
TF R12 R13	1 pop rbx 1 pop rbp
IF R14 R15 DF RBP RSP	1 ret
OF RFLAGS RIP FFFFFF820A31CC -	-0470020435 1 XP:FFF:FFFFFF820A31CC ptrac 1467 1 ttwu_do_activate(rq, p, 0);
	1 mov rdi,r15 1 mov rsi,rbx
	1 call 0xFFFFFFF820A1890; ttwu_d
	1 static void
	1358 1 ttwu_do_activate(struct rq *rq, struc 1 nop
	1 push rbp 1 mov rbp,rsp -
B::	
trigger devices trace Data Var List	PERF SYStem Step other previous
XP:0000:FFFFFF623B686B \\mlinux\random\add_interrupt_randomness+0x FFFF88003D27E540	0 CTS (-470020435.0.) MIX UP

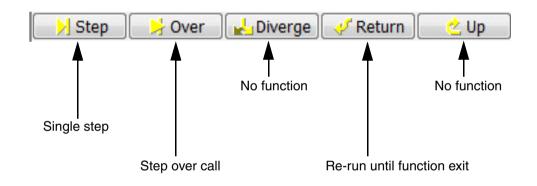
B::		
trigger devices trace Data Var List	PERF SYStem Step	other previous
XP:0000:FFFFFFF823B686B \\vmlinux\random\add_interrupt_randomness+0xt FFFF88003D27E540	0 CTS (-470020435.0.)	MIX UP

TRACE32 PowerView displays the state of the target as it was when the instruction of the trace record -470020435.0 was executed

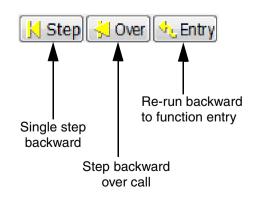
Now you can start to re-run the recorded program section within TRACE32 PowerView by forward or backward debugging.

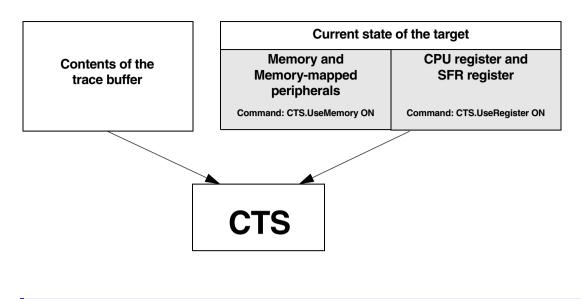
Forward debugging comma	inds	Backward debu	ugging commands	
[B::List /CORE 1]				
🔁 👌 Step 🛛 👌 Over 🚽 Diverge	Return 🔄 🖄 Up	🖣 Step 🗧 Over 🔩 Entry	📙 Off 🛛 🖄 Mode 🛛 Find:	core.c
addr/line	code	label mnemonio	comme	nt
1467		activate(rq, p, 0)	;	A
XP:0000:FFFFFFF820A31CC		mov	rdi,r15	
XP:0000:FFFFFFF820A31CF		mov	rsi,rbx	
XP:0000:FFFFFFF820A31D2		call	0xFFFFFFF820A1890	; ttwu_do_activate.c
1468		_unlock(&rq->lock)		
	4C89FF	mov	rdi,r15	
XP:0000:FFFFFFF820A31DA	E881617A00	call	0xFFFFFFF82849360	; _raw_spin_unlock
	<pre>#endif /* CONFIG_</pre>	_SMP */		τ.
J .	•	III		E de la constante de la const

Forward Debugging



Backward Debugging





CTS.UseMemory ON Default setting within TRACE32

If **CTS.UseMemory** is ON and TRACE32 detects that a memory address was not changed by the recorded program section, TRACE32 PowerView displays the current content of this memory in CTS display mode.

Since Intel[®] PT does not provide any information on read/write accesses and since most read/write accesses are done by using an indirect address, TRACE32 can not detect which memory content was changed. This is the reason why **CTS.UseMemory** has to be set to OFF.

If **CTS.UseMemory** is switch OFF, but your memory contains constants, you can configure TRACE32 to use these constants by the following commands:

MAP.CONST <address_range> CTS.MapConst ON

CTS.UseRegister ON Default setting within TRACE32

If **CTS.UseRegister** is ON and TRACE32 detects that a register was not changed by the recorded program section, TRACE32 PowerView displays the current content of this register in CTS display mode.

CTS.UseRegister has to be set to OFF, if you are using **Stack** mode for tracing.

Software under Analysis (no OS, OS or OS+MMU)

For the use of the function run-time analysis it is helpful to differentiate between three types of application software:

- 1. Software without operating system (abbreviation: **no OS**)
- 2. Software that includes an operating system (abbreviation: **OS**)
- 3. Software with an operating system that uses dynamic memory management to handle processes/tasks (**OS+MMU**).

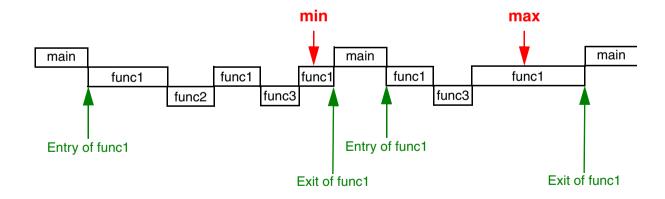
Flat vs. Nesting Analysis

TRACE32 provides two methods to analyze function run-times:

- Flat analysis
- Nesting analysis

The flat function run-time analysis bases on the symbolic instruction addresses of the trace entries. The time spent by an instruction is assigned to the corresponding function/symbol region.

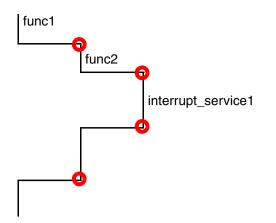
B::Trace.List	/Track						
🖉 Setup	🖪 Goto 🧃	jFind 🔤	Chart 🚺	Profile 🛛 🛄 M	IPS 🔶 More	Less	
record	d run addres			cycle d	ata	symbol	ti.back
150	0 pop 0 ret	ebp	,edx				
-0080696073 346		NP:00000	0000804851	E ptrace uncl(&stat	2.).	\func9+0x	(<u>3D</u> <0.005us 📤
7+6	0 mov 0 call	dwor o 0x804	d ptr [esp	,0x804902C ; func1	; dword ptr [esp],stat2	•
B::Trace.Cha	irt.sYmbol /Track	c					
Setup	iii Groups	Config	Goto	Find	MOut KHFull]	
(v				1.526017860		<u></u>	017840s
	addres	<u>ss 💀 🔜 </u>					
x86\sieve x86\sieve_f x86\sieve_f	ancs\func10 funcs\main funcs\func1 funcs\func8 funcs\func8		•				A P
B::Trace.STA	Tistic.sYmbol						
	Groups				ree Chart es: 910402147		
address to	otal mi 023.292ms	in n 0.014us	25.961ms	avr 0.509us	count 1814635.(0	ratio% /1) 60.503%	1% 2%
s\main:0 1 func10:0 \func1:0 func13:0 \func2:0	19.907ms 51.830ms 44.717ms 44.369ms 29.282ms	0.000us 0.025us 0.000us 0.006us 0.000us	1.288us 1.087us 0.978us 1.110us 1.435us	119.907ms 0.286us 0.035us 0.244us 0.161us	1. (1 181464. (1 1270241. 181464. 181463.	/0) 7.857% /0) 3.396% 2.930% 2.907% 1.918%	
	27.709ms 17.761ms 17.300ms	0.000us 0.000us 0.000us	0.991us 1.385us 1.388us 	0.153us 0.098us 0.095us		1.815% 1.163% 1.133%	



min	shortest time continuously in the address range of the function/symbol region
max	longest time continuously in the address range of the function/symbol region

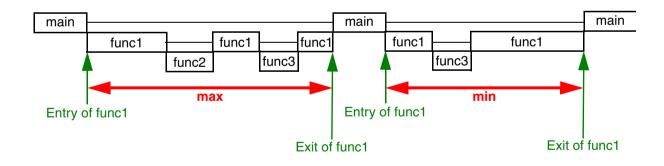
The function nesting analysis analyses only high-level language functions.

B::Trace.List /Track	
	ti.back
156 0 mov [eax],edx 156 0 } 0 pop ebp or et	A E
-0080696073 0 NP:00000000804851E ptrace\func9+0x3D 346 0 func1(&stat2);	<0.005us
dword ptr [esp],0x804902C; dword ptr [esp],stat2 call 0x8048099 ; func1	-
	F
🚟 B::Trace.Chart.Func /Track	
	7840s
x86\sieve_funcs\func1:00 x86\sieve_funcs\func1:00 x86\sieve_funcs\func1:00 x86\sieve_funcs\func8:000	
x86\sieve_funcs\func9:0	▼
x86\sieve_funcs\func9:000	
x86\sieve_funcs\func9:00	



In order to display a nesting function run-time analysis TRACE32 analyzes the structure of the program execution by processing the trace information. The focus is put on the transition between functions (see picture above). The following events are of interest:

- 1. Function entries
- 2. Function exits
- 3. Entries to interrupt service routines
- 4. Exits of interrupt service routines
- 5. Entries to TRAP handlers
- 6. Exits of TRAP handlers



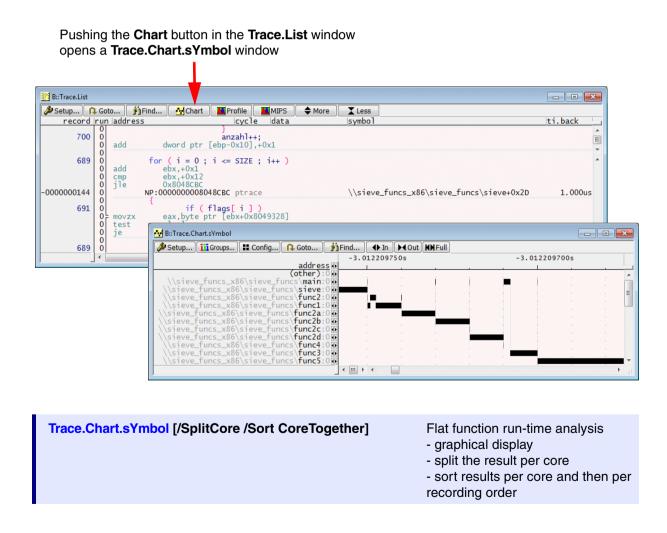
min	shortest time within the function including all subfunctions and traps
max	longest time within the function including all subfunctions and traps

The nesting analysis provides more details on the structure and the timing of the program run, but it is much more sensitive then the flat analysis. Missing or tricky function entries/exits may require additional setups before nesting analysis can be used.

NOTE: As long a TRACE32 does not support Synchronisation Time, cycle accurate tracing should be disabled for all kind of runtime measurement.

Function Time Chart

Default Time Chart

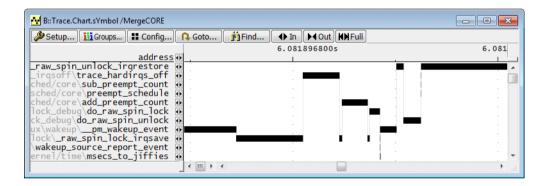


🎾 Setup 🛛 🚻 Gr	oups	∩ G	oto		Find	.][·	🕀 In	HO	ut) I	Full												
							-11	.494	3378	300s				-1	1.4	9433	3770	00s				
	addres	s 🕪 🚬						1								1						
	(other):0																200				0.000	
	(other):1	H E																				0.0
-langwell\ ln w	_irg_handler:0	H																				
adomain\ira	find_mapping:0	- D					1															
irg domain 1	egacy_revmap:0	N.																				
	\ira_to_desc:0		0.000				1															
	_tree_lookup:0				8 T ()						1	0.000										
	okup_element:0						<u> </u>															
	c_handle_irg:0			1.1				_														
uv/intol idl	e\intel_idle:1	- 11 I						— .														
	_enter_state:1	- 10																				
	ng\ktime_get:1																					
		•																				
	tsc\read_tsc:1													SIR	240	205			1.1.1.1	1993		10
nux\chip\ han	dle_edge_irq:0	٩Þ.,																				
(\spinlock\ _r	aw_spin_lock:0																					

Trace.Chart.sYmbol [/SplitCORE] /Sort CoreSeparated

Flat function run-time analysis

- graphical display
- split the result per core
- sort the results per recording order



Trace.Chart.sYmbol /MergeCORE

Flat function run-time analysis

- graphical display
- merge the results of all cores

Trace information is analyzed independently for each core. The time chart summarizes these results to a single result.

Trace.Chart.sYmbol /SplitTASK

Display function time chart including process information (OS, OS+MMU only)

		mg 14	Goto	Find	↓ I	n	MO	ut	KH Fi	ull																
							5.1	877	6799	900s			5.	877	680	000	s			5.	877	7680	010	0s		
				addres	5 H F			1						1												
	pick_next_ta				•																					
wakeup_pre	empt_entity.																									
\\vmlinux\f	air\set_next	_entity	@system	_server:1																						Ē
nux\fair\up	date_stats_w	ait_end	@system	_server:1	A F																					e l
\vmlinux\fa	ir\dequeue	_entity	@system	_server:1	A																					
\\vm]	inux\rbtree\				H																					
\\vmli	nux\rbtree\r	b erase	@svstem	server:1	Đ																					
ched switch	probe_sched	switch	@svstem	server:1	N							1						i i				-	_	÷.	1	
	ing_record_				N										1											
	trace\ tracin				N.														1							
	acer_tracin				HH								1		C			-								
	ouffer_recor														1											
	vent_task_sc																	1.1								
	<_wait_gueue																									
	utex\futex_w				66																					

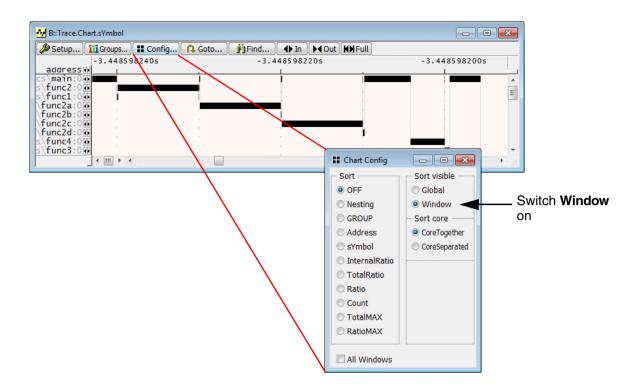
@ <task_name></task_name>	Process name information
@(unknown)	Function was recorded before first process switch information was recorded

Trace.Chart.sYmbol /TASK <name>

Display function time chart for specified process (OS, OS+MMU only)

🛿 Setup 🚺 Groups 📲 Config 📭 Goto 🎁	Find 🚯 In 🕅 Out 🕅 Full		
	5.878483120s	5.878483140s	
address 🚯		1	
x\sched/core\schedule@logcat:1 🚯			
\debug_smp_processor_id@logcat:1 🚯			
rf_event_task_sched_out@logcat:1 🔢			
(root)@(unknown):0 💀			
(root)@(unknown):1 🚯			
\process_64\switch_to@logcat:1 🚯			
al\native_load_gs_index@logcat:1 🔢			
mlinux\Global\ gs_change @logcat:1 🗰			
core\finish_task_switch@logcat:1 🚯			
ck_raw_spin_unlock_irg@logcat:1 🚯			

@ <task_name></task_name>	Process name information
(root)@(unknown)	Everything outside of the specified process.



If **Window** in the **Sort visible** field is switched ON in the **Chart Config** window, the functions that are active at the selected point of time are visualized in the scope of the **Trace.Chart.sYmbol** window. This is helpful especially if you scroll horizontally.

Analog to the timing diagram there is also a numerical analysis.

B::Trace.STA	Tistic.sYmbol							
Setup	👖 Groups 🔠	Config	oto 🗾 Deta	iled 📕 Tree	🔄 🕂 Chart 🛛 🌉 Pr	ofile		
	items: 34.	to	tal: 575.30	57ms sample	es: 349113863.			
address	total	min	max		count	ratio%	1%	2%
(other):0	0.000us	0.000us	-	0.000us	0.	0.000%		*
ncs\main:0	45.536ms	0.000us	1.322us	45.536ms	1.(1/0)	7.914%		
cs\sieve:0	349.960ms	0.010us	2.115us	0.503us	695860.	60.824%		E
cs\func2:0	10.340ms	0.000us	1.229us	0.149us	69587.	1.797%		
cs\func1:0	16.793ms	0.000us	0.851us	0.034us	487109.	2.918%		
s\func2a:0	6.714ms	0.000us	1.398us	0.096us	69587.	1.166%	-	
s\func2b:0	6.515ms	0.000us	1.457us	0.094us	69587.	1.132%		
s\func2c:0	6.696ms	0.000us	1.556us	0.096us	69587.	1.163%		
s\func2d:0	6.588ms	0.000us	1.444us	0.095us	69587.	1.145%		
cs\func4:0	2.387ms	0.000us	0.603us	0.034us	69587.	0.414%		
cs\func3:0	4.517ms		1.172us	0.065us	69587.	0.785%		-
co (runco.o	4	0.00003	1.1/203	0.00503	00007.	0.7050	•	
-	•							

survey	
item	number of recorded functions/symbol regions
total	time period recorded by the trace
samples	total number of recorded changes of functions/symbol regions (program flow continuously in the address range of a function/symbol region)

function details	
address	function/symbol region name (here per core)
	(other) program sections that can not be assigned to a function/symbol region
total	time period in the function/symbol region during the recorded time period
min	shortest time continuously in the address range of the function/symbol region
max	longest time continuously in the address range of the function/symbol region
avr	average time continuously in the address range of the function/symbol region (calculated by total/count)

count	number of new entries (start address executed) into the address range of the function/symbol region
ratio	ratio of time in the function/symbol region with regards to the total time period recorded

Pushing the **Config** button provides the possibility to specify a different column layout and a different sorting criterion for the address column. By default the functions/symbol regions are sorted by their recording order.

🖉 Setup 🔰	👖 Groups 🛛 🔡 C	Config 📭 🤇	Soto	F Deta	ailed	Tree 7		Chart	🔣 Pi	rofile		
	items: 34.	t	otal:	575.3	67ms	sample:	s: 3	491138	63.			
address	total	min	max		avr	6	cour	t		ratio%	1%	2%
(other):0 s\main:0 s\sieve:0 s\func2:0	0.000us 45.536ms 349.960ms 10.340ms	0.000us 0.000us 0.010us 0.000us	2.	322us 115us 229us	45	.000us .536ms .503us .149us		95860. 69587.	(1/0)	0.000% 7.914% 60.824% 1.797%		
<pre>\$\func1:0 {func2a:0 {func2b:0</pre>	16.793ms	0.000us nfig	0.	851us		.034us	4	87109.	[2.918%		
func2c:0	- Sort	- Sort v	isible –		wailab	le	_		sele	cted		
\func4:0	OFF	Glob	bal		IAME				Tota			
(Tunes.o	Nesting	Win	dow		GROUP				MIN			•
	© GROUP	- Sort o	ore		otalM	IN				Rage		
	Address	Core	Together		otalM		6	->	Cou			
	© sYmbol		Separate	, F	RatioM				Rati	-		
	 InternalRa 		Separate	- F	RatioM/				BAR	.LOG		
	0				AR.LI	-		<-				
	TotalRatio					AR.LOG						
	Ratio					AR.LIN						
	Count 🔘				CountM							
	© TotalMAX			0	CountM	IAX						
	© RatioMAX											

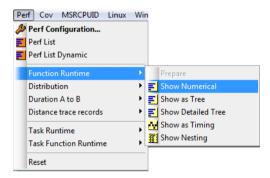
Further Commands

Trace.PROfileChart.sYmbol	Display dynamic program behavior graphically.
MIPS.PROfileChart.sYmbol	Display MIPS for all program symbols graphically.
MIPS.STATistic.sYmbol	Display MIPS for all program symbols numerically.

Function nesting analysis for OS requires that OS-aware debugging is configured. For more information refer to "**OS-aware Debugging**" (glossary.pdf).

Trace.STATistic.Func

- Nesting function run-time analysis
- numeric display
- core information is discarded exceptions are the
- @(unknown) task and the @(interrupt) task



E: Trace.STATistic.FUNC							E	- 0	x		
🌶 Setup 🚺 Groups 🚼 Config 🗛 Goto ╤] Detailed 🗱 Nesting 🙀 Chart											
funcs: 1882. total: 76.572ms intr: 172.160us											
range total min max lavr count lintern% 1% 2%											
\\vmlinux\int_sqrt\int_sqrt@swapper/0	6.011us	0.040us	0.765us	0.140us	43.	0.003%	+				
_pm_debug\pmu_get_new_cstate@swapper/0	32.522ms	0.001us	4.982ms	1.807ms	18.	7.376%			_		
ntel_soc_mrfld\could_do_s0ix@swapper/0	21.225ms	0.001us	4.981ms	1.179ms	18.	13.859%			_		
<pre>x\sched/core\cpumask_set_cpu@swapper/0</pre>	2.086ms	1.043ms	2.086ms	1.043ms	2.(0/1)	1.362%					
linux\sched/core\native_set_ldt@logcat	0.038us	0.038us	0.038us	0.038us	1.	<0.001%					
vmlinux\hrtimer\hrtimer_cancel@logcat	0.241us	0.241us	0.241us	0.241us	1.	<0.001%	+				
x\hrtimer\hrtimer_try_to_cancel@logcat	0.233us	0.233us	0.233us	0.233us	1.	<0.001%	+				
x\sched/core\schedule@surfaceflinger	59.021us	2.405us	6.968us	3.689us	16.(1/1)	0.008%	+				
<pre>\process_64\switch_to@surfaceflinger</pre>	4.596us	0.001us	1.402us	0.287us	16.	0.002%	+				
debug_smp_processor_id@surfaceflinger	7.039us	0.001us	0.728us	0.031us	227.	0.004%	+				
	•		III						F .		

Survey

B::Trace.STATistic.FUNC								
🖉 Setup) 👖 Config 🔒 Goto) 📰	Detailed [lesting 🔤 Ch	art					
survey	funcs: 1882	. to	tal: 76.57	2ms intr:	172.160us			
range	total	min	max	avr	count	intern% 1%	2%	
\vmlinux\int_sqrt\int_sqrt@swapper/0	6.011us	0.040us	0.765us	0.140us	43.	0.003% +		_
<pre>m_debug\pmu_get_new_cstate@swapper/0</pre>	32.522ms	0.001us	4.982ms	1.807ms	18.	7.376%		
el_soc_mrfld(could_do_s0ix@swapper/0	21.225ms	0.001us	4.981ms	1.179ms	18.	13.859%		_
sched/core\cpumask_set_cpu@swapper/0	2.086ms	1.043ms	2.086ms	1.043ms	2. (0/1)	1.362%	-	
nux\sched/core\ native_set_ldt @logcat	0.038us	0.038us	0.038us	0.038us	1.	<0.001% +		
mlinux\hrtimer\ hrtimer_cancel @logcat	0.241us	0.241us	0.241us	0.241us	1.	<0.001% +		
hrtimer\hrtimer_try_to_cancel@logcat	0.233us	0.233us	0.233us	0.233us	1.	<0.001% 🗲		
sched/core\schedule@surfaceflinger	59.021us	2.405us	6.968us	3.689us	16.(1/1)	0.008% +		
rocess_64\switch_to@surfaceflinger	4.596us	0.001us	1.402us	0.287us	16.	0.002% +		
<pre>ebug_smp_processor_id@surfaceflinger</pre>	7.039us	0.001us	0.728us	0.031us	227.	0.004% +		
	4		111					

survey	
func	number of functions in the trace
total	total measurement time
intr	total time in interrupt service routines

OVERFLOW funcs: 174. total: 1.392s intr: 902.302us stack overflow at 186892464.

survey (issue indication)	
stopped: <time></time>	The analyzed trace recording contains program stops. <i><time></time></i> indicates the total time the program execution was stopped.
<number> problems</number>	The nesting analysis contains problems. Please contact support@lauterbach.com .
<number> workarounds</number>	The nesting analysis contains issues, but TRACE32 found solutions for them. It is recommended to perform a sanity check on the proposed solutions.
stack overflow at <record></record>	The nesting analysis exceeds the nesting level 200. It is highly likely that the function exit for an often called function is missing. The command Trace.STATistic.TREE can help you to identify the function. If you need further help please contact support@lauterbach.com .
stack underflow at <record></record>	The nesting analysis exceeds the nesting level 200. It is highly likely that the function entry for an often executed function is missing. The command Trace.STATistic.TREE can help you to identify the function. If you need further help please contact support@lauterbach.com .

B::Trace.STATistic.FUNC							E		×	
Setup 🗓 Groups 😫 Config 📭 Goto 🛃 Detailed 🔄 Nesting 🗌 😴 Chart										
funcs: 1882. total: 76.572ms intr: 172.160us										
range	total I	min	max	avr	count	intern%	1%	2%		
\\vmlinux\int_sqrt\int_sqrt@swapper/0	6.011us	0.040us	0.765us	0.140us	43.	0.003%	+			
pm_debug\pmu_get_new_cstate@swapper/0	32.522ms	0.001us	4.982ms	1.807ms	18.	7.376%			_	
ntel_soc_mrfld\could_do_s0ix@swapper/0	21.225ms	0.001us	4.981ms	1.179ms	18.	13.859%			_	
x\sched/core\ cpumask_set_cpu @swapper/0	2.086ms	1.043ms	2.086ms	1.043ms	2.(0/1)	1.362%				
linux\sched/core\native_set_ldt@logcat	0.038us	0.038us	0.038us	0.038us	1.	<0.001%	+			
\vmlinux\hrtimer\hrtimer_cancel@logcat	0.241us	0.241us	0.241us	0.241us	1.	<0.001%	+			
x\hrtimer\hrtimer_try_to_cancel@logcat	0.233us	0.233us	0.233us	0.233us	1.	<0.001%	+			
x\sched/core\schedule@surfaceflinger	59.021us	2.405us	6.968us	3.689us	16.(1/1)	0.008%	+			
\process_64\switch_to@surfaceflinger	4.596us	0.001us	1.402us	0.287us	16.	0.002%	+			
\debug_smp_processor_id@surfaceflinger	7.039us	0.001us	0.728us	0.031us	227.	0.004%	+		-	
	•		m						 Id 	

columns	
range (NAME)	function name, sorted by their recording order as default

\\vmlinux\hrtimer\hrtimer_cancel@logcat

HLL function hrtimer_cancel running in process @logcat.

Please be aware that no core information is provided for processes and their functions.

Trace.STATistic.Func /TASK <task_magic> | <task_name> | <task_id>

Trace.STATistic.Func /TASK "logcat"

B::Trace.STATistic.Func /TASK "	logcat"									
🖉 Setup 📔 Groups 📔 Config 🚺 🗛 Goto 💽 Detailed 🛛 🗱 Nesting 🖯 🖼 Chart										
funcs: 250. total: 132.670us										
range	total r	nin	max	avr	count	intern% 1%	2%			
(root)	132.670us	-	132.670us	132.670us	-	27.928%	-			
ux\Global\ia32_syscall	89.107us	2.240us	21.268us	8.911us	10.	2.626%				
mpat\compat_sys_select	52.967us	6.250us	20.837us	10.593us	5.	0.314%				
compat_core_sys_select	50.768us	5.324us	20.584us	10.154us	5.	0.305% 🗲				
linux\select\do_select	49.403us	4.971us	20.470us	9.881us	5.	1.626%				
x\read_write\sys_write	31.858us	9.856us	11.701us	10.619us	3.	0.001% +				
x\read_write\ vfs_write	31.855us	9.855us	11.700us	10.618us	3.	0.343%				
ad_write\do_sync_write	28.045us	8.736us	10.459us	9.348us	3.	0.060% 🗲	*			
	•		111				► at			

Interrupt service routines are assigned to the @(interrupt) task. Core information is provided for the @(interrupt) task.

E B:: Trace.STATistic.FUNC						
Setup II Groups Config 🗣 Goto 🗾 Detailed 🖉 Nesting	Chart					
	funcs: 1882	. to	otal: 76.57	2ms intr:	172.160us	
range		min	max		count	intern% 1%
(root)@(interrupt):1	0.000us	-	-	0.000us	-	0.000%
→\\vmlinux\Global\apic_timer_interrupt@(interrupt):1	172.160us	5.210us	9.095us	6.376us	27.	0.004% 🗲 🚽
\trace_irqsoff\trace_hardirqs_off_caller@(interrupt):1	0.757us	0.000us	0.040us	0.014us	54.	<0.001% +
\\vmlinux\apic\smp_apic_timer_interrupt@(interrupt):1	162.406us	4.730us	8.904us	6.015us	27.	0.003% +
linux\apic_flat_64\native_apic_mem_write@(interrupt):1	0.872us	0.000us	0.104us	0.032us	27.	<0.001%
\\vmlinux\softirq\irq_enter@(interrupt):1	29.043us	0.674us	1.765us	1.076us	27.	0.001%
<pre>\smp_processor_id\debug_smp_processor_id@(interrupt):1</pre>	11.004us	0.000us	0.320us	0.016us	673.	0.007%
\\vmlinux\rcutree\rcu_irq_enter@(interrupt):1	6.486us	0.041us	1.246us	0.240us	27.	0.002% +
<pre>vmlinux\trace_irgsoff\trace_hardirgs_off@(interrupt):1</pre>	4.589us	0.000us	0.358us	0.012us	385.	0.002% +
inux\rcutree\rcu_eqs_exit_common.isra.52@(interrupt):1	0.861us	0.003us	0.370us	0.032us	27.	<0.001%
\\vmlinux\softirg\local_bh_disable@(interrupt):1	4.754us	0.034us	0.687us	0.176us	27.	<0.001% +
\\vmlinux\softirg_local_bh_disable@(interrupt):1	3.849us	0.001us	0.566us	0.110us	35.	0.002% +
vmlinux\trace_irgsoff\trace_softirgs_off@(interrupt):1	0.295us	0.000us	0.068us	0.008us	35.	<0.001% + -
	•		'm			

An arrow before the interrupt function indicates the function executed after the interrupt occurred:

→\\vmlinux\Global\apic_timer_interrupt@(interrupt):1

The unknown Task

All function recorded before the first process switch is recorded are assigned to the @(unknown) task. Core information is provided for the @(unknown) task.

B::Trace.STATistic.FUNC							
Ø Setup III Groups III Config Q Goto F D	etailed [📳 Nes	ting 🗌 🚟 Char	t				
	funcs: 1882	2. to	otal: 76.57	2ms intr:	172.160us		
range	total	min	max	avr	count	intern% 1%	
<pre>sched/core\add_preempt_count@(unknown):0</pre>	2.830ms	0.000us	0.348us	0.016us	172865.(1/0)	1.847%	
mlinux\intel_idle\intel_idle@(unknown):1	14.832ms	1.195ms	4.981ms	2.472ms	6. (1/0)	2.405%	
nlock_raw_spin_lock_irqsave@(unknown):0	3.752ms	0.001us	0.721us	0.043us	86768. (1/0)	0.848% 🗲 🔚	
nlock_debug\do_raw_spin_lock@(unknown):0	3.609ms	0.000us	0.720us	0.021us	172865.	2.356%	
vmlinux\sched/core\wake_up@(unknown):0	75.603ms	0.010us	75.512ms	18.090us	43730. (1/20	1.158% -	
\sched/core\wake_up_common@(unknown):0	75.520ms	0.000us	75.512ms	17.956us	43726. (0/20	0.384%	
_raw_spin_unlock_irgrestore@(unknown):0	2.755ms	0.000us	0.735us	0.032us	86786.	0.779%	
ock_debug\do_raw_spin_unlock@(unknown):0	2.125ms	0.000us	0.280us	0.012us	172879.	1.387%	
e_irqsoff\trace_hardirqs_off@(unknown):0	548.909us	0.000us	0.089us	0.003us	173544.	0.358% 🗲 🔻	

Default Results

B::Trace.STATistic.FUNC								
🌽 Setup 👖 Groups 🔡 Config 🔃 Goto 🗐	Detailed Ne	esting 🗌 🚾 Cha	art					
	funcs: 1882.	to	tal: 76.57	2ms intr: 2	172.160us			
range	total n	nin Ir	nax	avr	count	intern% 19	6 2%	
\\vmlinux\int_sqrt\int_sqrt@swapper/0	6.011us	0.040us	0.765us	0.140us	43.	0.003% +		
_pm_debug\pmu_get_new_cstate@swapper/0	32.522ms	0.001us	4.982ms	1.807ms	18.	7.376%		
ntel_soc_mrfld(could_do_s0ix@swapper/0	21.225ms	0.001us	4.981ms	1.179ms	18.	13.859%		
x\sched/core\cpumask_set_cpu@swapper/0	2.086ms	1.043ms	2.086ms	1.043ms	2.(0/1)	1.362%		
linux\sched/core\native_set_ldt@logcat	0.038us	0.038us	0.038us	0.038us	1.	<0.001% +		
\vmlinux\hrtimer\hrtimer_cancel@logcat	0.241us	0.241us	0.241us	0.241us	1.	<0.001% +		
x\hrtimer\hrtimer_try_to_cancel@logcat	0.233us	0.233us	0.233us	0.233us	1.	<0.001% +		
x\sched/core\schedule@surfaceflinger	59.021us	2.405us	6.968us	3.689us	16.(1/1)	0.008% +		
\process_64\switch_to@surfaceflinger	4.596us	0.001us	1.402us	0.287us	16.	0.002% +		
\debug_smp_processor_id@surfaceflinger	7.039us	0.001us	0.728us	0.031us	227.	0.004% +		-
	•		m					Strain and a

columns (cont.)	
total	total time within the function
min	shortest time between function entry and exit, time spent in interrupt service routines is excluded No min time is displayed if a function exit was never executed.
max	longest time between function entry and exit, time spent in interrupt service routines is excluded
avr	average time between function entry and exit, time spent in interrupt service routines is excluded

B::Trace.STATistic.FUNC								
Setup II Config Goto	Detailed [🖪 N	lesting 🛛 🚟 Ch	art					
	funcs: 1882	. to	tal: 76.57	2ms intr:	172.160us			
range	total	min	max	avr	count	intern% 1%	2%	
\\vmlinux\int_sqrt\int_sqrt@swapper/0	6.011us	0.040us	0.765us	0.140us	43.	0.003% +		
_pm_debug\pmu_get_new_cstate@swapper/0	32.522ms	0.001us	4.982ms	1.807ms	18.	7.376%		_
ntel_soc_mrfld\could_do_s0ix@swapper/0	21.225ms	0.001us	4.981ms	1.179ms	18.	13.859%		_
x\sched/core\cpumask_set_cpu@swapper/0	2.086ms	1.043ms	2.086ms	1.043ms	2. (0/1)	1.362%	-	
linux\sched/core\native_set_ldt@logcat	0.038us	0.038us	0.038us	0.038us	1.	<0.001% +		
\vmlinux\hrtimer\hrtimer_cancel@logcat	0.241us	0.241us	0.241us	0.241us	1.	<0.001% +		
x\hrtimer\hrtimer_try_to_cancel@logcat	0.233us	0.233us	0.233us	0.233us	1.	<0.001% +		
x\sched/core\schedule@surfaceflinger	59.021us	2.405us	6.968us	3.689us	16. (1/1)	0.008% +		
\process_64\switch_to@surfaceflinger	4.596us	0.001us	1.402us	0.287us	16.	0.002% +		
\debug_smp_processor_id@surfaceflinger	7.039us	0.001us	0.728us	0.031us	227.	0.004% +		-
	•		m					

columns (cont.)	
count	number of times within the function

If function entries or exits are missing, this is displayed in the following format:

<ti>times within the function >. (<number of missing function entries>/<number of missing function exits>).</ti>

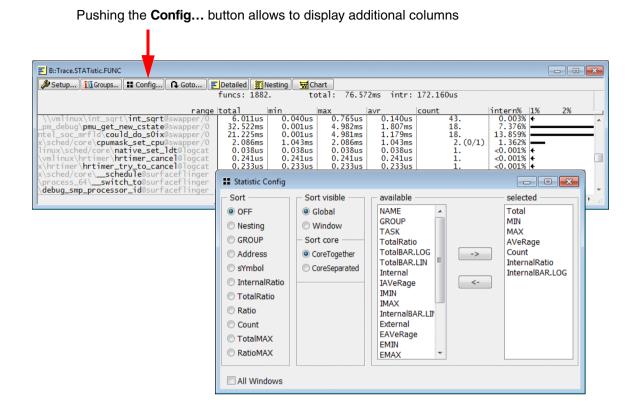
3671.(0/1)

Interpretation examples:

- 1. 2. (2/0): 2 times within the function, 2 function entries missing
- 2. 4. (0/3): 4 times within the function, 3 function exits missing
- 3. 11. (1/1): 11 times within the function, 1 function entry and 1 function exit is missing.

	If the number of missing function entries or exits is higher the 1 the analysis performed by the command Trace.STATistic.Func might fail due to nesting problems. A detailed view to the trace contents is recommended.
--	--

columns (cont.)	
intern% (InternalRatio, InternalBAR.LOG)	ratio of time within the function without subfunctions, TRAP handlers, interrupts (net time)



B::Trace.STATistic.FUNC						
Ø Setup ↓ Goto ↓ Config ↓ Goto ↓	Detailed [Nesting 🛛 🖼 Ch	art			
	funcs: 1882	?. to	tal: 76.57	2ms intr:	172.160us	
range	total	internal	iavr	imin	imax	avr
\\vmlinux\int_sqrt\int_sqrt@swapper/0	6.011us	6.011us	0.140us	0.040us	0.765us	0.140us 🔺
_pm_debug\pmu_get_new_cstate@swapper/0	32.522ms	11.297ms	627.621us	0.000us	4.982ms	1.807ms
ntel_soc_mrfld\could_do_s0ix@swapper/0	21.225ms	21.225ms	1.179ms	0.000us	4.981ms	1.179ms
x\sched/core\cpumask_set_cpu@swapper/0	2.086ms	2.086ms	1.043ms	2.086ms	2.086ms	1.043ms
linux\sched/core\native_set_ldt@logcat	0.038us	0.038us	0.038us	0.038us	0.038us	0.038us
\vmlinux\hrtimer\hrtimer_cancel@logcat	0.241us	0.008us	0.008us	0.008us	0.008us	0.241us
x\hrtimer\hrtimer_try_to_cancel@logcat	0.233us	0.042us	0.042us	0.042us	0.042us	0.233us
x\sched/core\schedule@surfaceflinger	59.021us	12.326us	0.770us	0.398us	1.907us	3.689us
\process_64\switch_to@surfaceflinger	4.596us	3.249us	0.203us	0.001us	0.870us	0.287us
\debug_smp_processor_id@surfaceflinger	7.039us	7.039us	0.031us	0.000us	0.728us	0.031us 🔻
	4					► a
-						m

columns (cont.) - tim	columns (cont.) - times only in function				
Internal	total time between function entry and exit without called sub-functions, TRAP handlers, interrupt service routines				
IAVeRage	average time between function entry and exit without called sub- functions, TRAP handlers, interrupt service routines				
IMIN	shortest time between function entry and exit without called sub- functions, TRAP handlers, interrupt service routines				
ΙΜΑΧ	longest time spent in the function between function entry and exit without called sub-functions, TRAP handlers, interrupt service routines				
InternalRatio	<internal function="" of="" time="">/<total measurement="" time=""> as a numeric value.</total></internal>				
InternalBAR	<internal function="" of="" time="">/<total measurement="" time=""> graphically.</total></internal>				

E::Trace.STATistic.FUNC								- • ×
Setup 👖 Groups 🔡 Config 📭 C	ioto 🔚 Detail	ed) [Nesting	Chart					
	funcs: 1882	2. to	otal: 76.57	'2ms intr:	172.160us			
		eintr			intrcount	max	avr	count
<pre>[le\cpuidle_enter_state@swapper/0</pre>	53.448ms	121.051us	9.095us	965.860us	19.	4.983ms	2.813ms	19. 🔺
\timekeeping\ktime_get@swapper/0		-	-	0.018us		0.552us	0.121us	51.
\\vmlinux\tsc\ read_tsc @swapper/0	1.462us	-	-	0.000us	-	0.238us	0.029us	51.
soff\trace_hardirgs_on@swapper/0		-	-	0.000us	-	0.303us	0.026us	177.
nel/time\ns_to_timeval@swapper/0	0.964us	-	-	0.037us	-	0.077us	0.046us	21
el/time\ns_to_timespec@swapper/0		-	-	0.000us	-	0.306us	0.030us	39. —
idle\cpuidle_idle_call@swapper/0	58.778ms	135.578us	120.672us	966.982us	21.	56.466ms	5.497ms	20.
inux\menu\menu_reflect@swapper/0	0.528us	-	-	0.000us	-	0.044us	0.025us	21.
debug_smp_processor_id@swapper/0	12.309us	-	-	0.000us	-	0.426us	0.020us	600.
/process\arch_cpu_idle@swapper/0	58.778ms	135.578us	120.672us	967.019us	21.	56.466ms	5.574ms	21
	•		III					

columns (cont.) - interrupt times					
ExternalINTR total time the function was interrupted					
ExternalINTRMAX	max. time one function pass was interrupted				
INTRCount	number of interrupts that occurred during the function run-time				

funcs: 1882		setup 🚻 Groups 🔛 Config 📭 Goto 🗾 Detailed 🛛 🧱 Nesting 🗖 📆 Chart										
	funcs: 1882. total: 76.572ms intr: 172.160us											
	taskcount	etask	etaskmax				count	_				
	-	-	-					2.				
	4.	171.020us	89.085us	2.725us	161.386us	49.569us		6.				
2.349us		-	-	0.120us	1.205us	0.391us		6.				
1.691us	-	-	-	0.098us	0.660us	0.282us		6.				
	4.	171.020us	89.085us	27.817us	40.221us	34, 019us		2.				
	1 12	-	-					2.				
	2	14 045us	8 21 Sus					5				
	5.							2: 0				
	2.	14.04005	0.21303					2				
0.16105	-	-	-					<u><u></u></u>				
0.139us	-	-	-	0.060us	0.0/9us	0.0/0us		Ζ.				
	0.322us 297.415us 2.349us 1.691us 68.038us 3.114us 39.675us 39.414us 0.161us	0.322us - 297.415us 4. 2.349us - 1.691us - 68.038us 4. 3.114us - 39.675us 2. 39.414us 2. 0.161us -	0.322us 297.415us 4. 171.020us 2.349us 68.038us 4. 171.020us 3.114us - 171.020us 39.675us 2. 14.045us 0.161us - 14.045us	0.322us - - - 297.415us 4. 171.020us 89.085us 2.349us - - - 1.691us - - - 68.038us 4. 171.020us 89.085us 3.114us - - - 39.675us 2. 14.045us 8.215us 39.414us 2. 14.045us 8.215us 0.139us - - -	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$				

columns - process related information							
TASKCount number of tasks that interrupt the function/task							
ExternalTASK	total time in other tasks						
ExternalTASKMAX	max. time 1 function/task pass was interrupted by a task						

Trace.STATistic.TREE Nesting function run-time analysis - tree display
Pert List Pert List Pert List Distribution Distrinte_then_ressue

Trace.STATistic.TREE /TASK "rild"

GROUPs for OS-aware Tracing

TRACE32 PowerView provides the GROUP command to structure the trace evaluation.

If you use a target OS such a Linux, the following groups are created by the Lauterbach scripts and Lauterbach OS menus:

- A GROUP "kernel", color RED, to mark the OS kernel.
- A GROUP "droid", color BLUE, to mark virtual machine byte code e.g. Android/Dalvik.
- A GROUP <process_name> per process, color GREEN.
- A GROUPs <module_name> per kernel module, color YELLOW.

III B::GROUP.List		×
Constant Constant	Image: Second	
		Ŧ
]].	ł	at

A group can have the following statuses:

- enable
- enable + merge
- enable + hide

KReset O Disable All Enable All O Hide All Show All Store	SLoad	Crea	ate) 踲	Create Task	
proup	enable	hide	merge	color	
∃ "droid"	√			BLUE	
X:0000:00000004000000000000006FFFFFF	V V			BLUE	
∃ "kernel"	V V			RED	
X:0000:FFFFFFF80000000FFFFFFFFFFFFFFFFF	√			RED	
B "Process1"	√			GREEN	
X:0555:00000000000000000000000FFFFFFF	√			GREEN	
B "Process2"	√			GREEN	
X:0559:000000000000000000000000000000000	√			GREEN	
"other"	- V				

If a GROUP is enabled:

• The trace information recorded for the group members is marked with the color assigned to the group.

B::Trace.List Lis	t.TASK DEFau	lt							- • •
🌽 Setup 🔃	Goto	Find Chart	Profile	MIPS	A More	Less			
	run addre			cycle	data		symbol	ti.back	
-0736219411	1 je 1 1 out:	0xFFFFFFF XP:011A:FFFFF		ptrace			smp_processor_id+0x25	<0.005us	Ē
52	1 pop 1 mov 1 pop 1 pop 1 pop 1 pop	return this_ rbx eax,r12d r12 r13 rbp	•						•
072624.044.0		- task: com.andr	oid.browser		30010DC654	0)		0.005	
-0736219410 -0736219388		XP:0000:FFFFF - group "kernel"		owner ptrace	FFFF8800		\chip\handle_edge_irq	<0.005us	
		might be nec controller h of the loop u the handler loop is left	vas running	isable (is requi es the i . If all	(mask) the ires to re interrupts pending	interrup enable th which ha interrupt	t depending on the e interrupt inside ve arrived while s are handled, the		
495	0 void 0 hand 0 { 0 nop 0 push 0 mov 0 push	rbp rbp,rsp r12	[rax+rax+0	x0]	ct irq_des	c *desc)			
496	0 lea	raw_spin_locl r12,rsi+0		ск);					→ E 4

Group-based trace analyses commands are provided e.g. Trace.STATistic.GROUP.

B::Trace.STATistic.GROUP						_	- • •			
🌽 Setup 🚺 Groups 🔛 Config 🕞 Goto 🗾 Detailed 📰 Tree 🛃 Chart 🖉 Profile										
items: 8. total: 58.288ms samples: 37865565.										
address	total	min	max	avr	count	ratio% 19	6 2%			
(other):0	10.379ms	0.000us	1.912us	0.034us	306535.	17.805%				
group "kernel":0	5.890ms	0.000us	2.010us	0.031us	191936. (1/0)	10.105%				
group "Process2":0	25.055ms	0.000us	1.800us	0.063us	396829.	42.983%				
group "Process1":0	16.965ms	0.000us	2.502us	0.095us	179379.	29.104%	=			
(other):1	45.171ms	0.000us	9.163ms	2.626us	17201. (0/1)	77.496%				
group "Process2":1	477.943us	0.000us	5.514us	0.098us	4860.	0.819% +				
group "Process1":1	359.292us	0.000us	12.388us	0.185us	1943.	0.616% +				
group "kernel":1	12.233ms	0.000us	9.944ms	1.030us	11875.	20.987%	· · ·			
	•		III							

•

🛠 Reset 🛛 🔘 Disable All 🔘 Enable All 🖉 O Hide All 🔘 Show All 😨 Store	SLoad	Creat	e] 뙙	Create Task	
group	enable	hide n	nerge	color	
🖻 "droid"	V 1		V	BLUE	
X:0000:00000004000000000000006FFFFFF	V V		Ý	BLUE	
🗆 "kernel"	V V		Ý	RED	
X:0000:FFFFFFF80000000FFFFFFFFFFFFFFFFF	l V		Ý	RED	
"Process1"	V V			GREEN	
X:0555:00000000000000000000000FFFFFFF	V V			GREEN	
🗉 "Process2"	V V		√	GREEN	
X:0559:00000000000000000000000FFFFFFF	V V		V V	GREEN	
"other"	l V				

If a GROUP is enabled and merge is checked:

• The group represents its members in all trace analysis windows. No details about group members are displayed.

🙀 B::Trace.Chart.sYmbol										×
Setup 🚺 Groups	E Config	Goto	Find	🔶 In	MOut k	NFull				
		-10.531990	000s			-10.5	5319895	600s		
address	HB	1					1			_
group "kernel":0	•									
group "Process2":0										
_pm_wakeup_event:0	HE									
	HE									
d/core\ <u>wake_up</u> :0	HE									-
		•	m						1	F
,	_									

🛠 Reset 🛛 🔘 Disable All 🔵 Enable All 🗍 🔿 Hide All 🗍 🕥 Show All 💱 Store	SLoad	Crea	ate] 踲	Create Task	
group	enable	hide	merge	color	
🖻 "droid"	V	V		BLUE	
X:0000:00000004000000000000006FFFFFF	V V	V V		BLUE	
🗉 "kernel"	V V	V		RED	
X:0000:FFFFFFF80000000FFFFFFFFFFFFFFFF	√	√		RED	
Process1"	√			GREEN	
X:0555:0000000000000000000000FFFFFFF	√			GREEN	
Process2"	√			GREEN	
X:0559:000000000000000000000000FFFFFFF	√			GREEN	
"other"	√				

If a GROUP is enabled and hide is checked:

- The group represents its members in all trace analysis windows. No details about group members are displayed (same as merge checked).
- The trace information recorded for the group members is hidden in the **Trace.List** window.

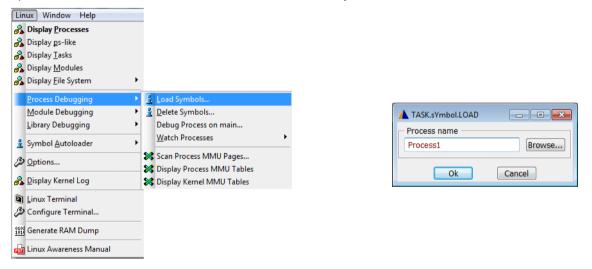
B::Trace.List List.TASK DEFault			
🎾 Setup 🚺 Goto 🎁 Find Mart 🔛 Profile	MIPS Avre	Less	
run address	cycle data	symbol	ti.back
109 0 mov edi,0x1 0 call 0xFFFFFFF8284CC90; group "kernel" 0 spin_acquire(&lock->de 0 /*);	0.571us

GROUP Creation

The GROUPs "kernel" and "droid" are typically created in the start-up script that sets up the OS-aware debugging.

GROUP.Create <group_name> <address_range> I <color>

certification certification



For more details about GROUPs refer to the **GROUP** command group.