

TRACE 32°

DEBUG & TRACE TOOLS for Arm®-based SoCs





Arm® support at a glance

More than 25 years of experience in Arm[®] debugging enable us to provide the best-in-class debug and trace tools for Arm[®]-based systems:

- Multicore debugging and tracing for any mixture of Arm[®] and DSP cores
- > Support for all CoreSight[™] components to debug and trace an entire SoC
- > Powerful code coverage and run-time analysis of functions and tasks
- OS-aware debugging of kernel, libraries, tasks of all commonly used OSes

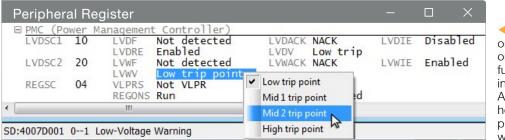


DISCOVER MORE: www.lauterbach.com/arm

Debugging Arm®-based Systems

The Lauterbach debug tools for Arm[®] accompany you throughout the whole development process, from the early pre-silicon phase by debugging on an instruction set simulator or a virtual prototype, over board bring-up, till quality and maintenance work on the final product.

The features range from simple step/go/break, programming of on-chip flash, external NAND, eMMC, parallel and serial NOR flash devices, support for NEON, VFP and SVE units, to OS-aware debug and trace concepts for 32-bit and 64-bit multicore systems that make use of the latest security technologies.



The peripheral registers on the chip are displayed on a logical level. The function bits can easily be interpreted and modified. A pull-down menu helps to select the peripheral module you want to inspect.

Multicore Debugging Core 0 Nover Diverge C Up ► Step 🖌 Return ▶ Go 🚺 Break 🛛 🕅 Mode label addr/line code MX:00001BD0 0B150274 mnemonic comment 20,w19,w21 <= SIZE) 821 while (k MX:00001BD4 14000006 822 MX:00001BD8 90000000 MX:00001BDC F946F401 Ox1BEC flags[k] = FALSE; adrp x0,0x1000 ldr x1,[x0,#0 Core 1 ¢ Up Nover 🕁 Diverge II Break 🕅 Mode ► Step ✓ Return ▶ Go x1,[x0,#0 x0,w20 addr/line code MX:00001BE0 MX:00001BE4 93407E80 sxtw label mnemonic comment 3820683F vtriplearray[0][0][0] = 1; adrp x0.0x1000 dr x0.[x0.#0xE06]k += prime; 719 823 MX:00001BE8 MX:00001730 9000000 MX:00001734 F9470400 MX:00001738 52800021 OB150294 w20,w20,v while (k <= SIZE) ; x0, [x0, #3592] 821 E) { w20,#0x12 0x1BD8 MX:00001BEC 7100449F cmp b.le MX:0000173C 3900001 720 MX:00001740 9000000 MX:00001744 F9470400 MX:00001748 52800041 w1, #0x1 w1, [x0]] = 2; x0,0x1000 x0, [x0,#0xE08] w1,#0x2 w1,#0x2 MX:00001BF0 vtriplearray[1][0][0] 4FEE4D adrp 1dr mov vtriplearray[0][1][0] = 3; adrp x0,0x1000] dr x0,[x0,#0xE08] MX:0000174C 39003001 MX:00001740 MX:00001750 MX:00001754 F9470400 : x0, [x0, #3592] ∨

▲ TRACE32[®] supports simultaneous debug and trace of homogeneous and heterogeneous multicore and multiprocessor systems with one debug tool. Start/stop synchronisation of the cores and a time-correlated display of the traced data gives you a global view of the system's state and the interplay of the cores.

DISCOVER MORE: www.lauterbach.com/features/multicore-debugging-and-tracing

OS-Aware Debugging and Tracing

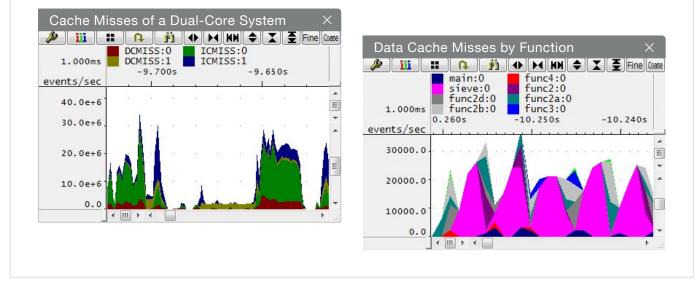
Lauterbach supports debugging of system and application software of all commonly used operating systems. The hypervisor awareness functionality allows seamless debugging of the overall system if multiple operating systems run in parallel on a single hardware platform:

- > Non-intrusive access to task lists and other kernel information by OS awareness
- > Non-intrusive access to guest (VM) list and guest information by hypervisor awareness
- > Full support for symmetric (SMP) and asymmetric (AMP) multiprocessing
- > Stack coverage and call hierarchy of waiting tasks
- > Statistics and graphic display of task and function run-time
- > Support for secure hypervisors

Display <u>P</u> rocesses			
Display <u>p</u> s-like			List of Machines and Tasks – \Box $ imes$
Display <u>T</u> asks			File Edit View Var Break Run CPU Misc Trace Perf Cov Cortex-A53 Xen Dom0 Linux FreeRTOS Window Help
Display <u>M</u> odules			H H M ↓ 4' C ▶ II ⊠ ? K? © ⊠ III ■ 🚳 🚳 🚳 🔞 🛊 1 (⊅
Display <u>F</u> ile System	-		B:TASKListMACHINES
Process Debugging	•	Load Symbols	magic name mid vttb extension(s) state
Module Debugging	•	Delete Symbols	00080007FF51000 Dom0 1. 000100007AEF8000 Dom0 runnable
Library Debugging	•	Debug Process on main	D0080007AED8000 Linux 2. 0002000079FB0000 Linux running D00800079F76000 FreeRTOS 3. 0003000079F4E000 FreeRTOS running
Symbol Autoloader		Watch Processes	
Symbol Autoloader		C D 10000	, B::EXT.FreeRTOS.TaskList
<u>D</u> isplay Kernel Log		Scan Process MMU Pages Scan All MMU Tables	igic name num prio state 148480 IDLE 4. 0. running Caller
Linux Terminal	7		13F360 xenbusTas 1. 4. blocked -000 prvCheckTasksWaitingTermination() r014320 SieveDemo 3. blocked -001 prvIdeTask(pvParameters = 0x0) l0147420 SieveDemo 3. blocked -end of frame
– Configure Terminal			10147420 sieveDemo 3. 3. blocked • end of frame
Generate ramdump			
Help Linux Awareness			

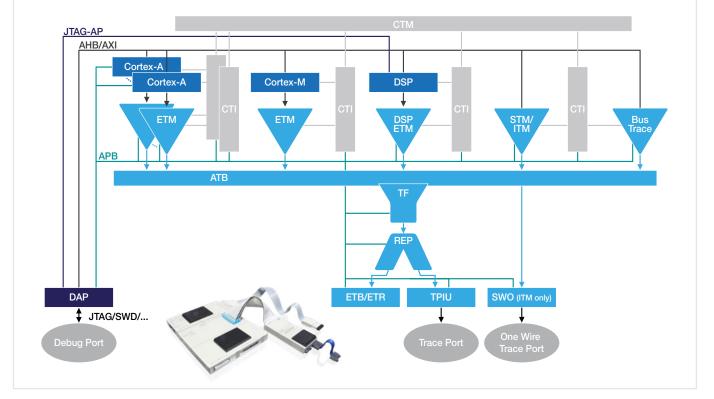
Utilization of the Performance Monitors

Many Arm[®] based devices include hardware counters with the ability to count specific hardware events like execution of certain kinds of instructions, cache or TLB events, or stall cycles caused by a specific reason. TRACE32[®] can sample these counters periodically. The results can be correlated with program trace information. This provides you with statistics on the system behaviour and system performance for finding bottlenecks and tuning the application.



Full Support for Arm[®] CoreSight[™] Technology

Arm[®] CoreSight[™] technology offers the chip designer various components to extend the core's debug functionality with the objective of debugging and tracing an entire system-on-chip. A joint JTAG or Serial Wire Debug interface allows real-time access to the on-chip buses and control of the cores and the CoreSight system itself. A common trace bus combines trace data from multiple sources like processor program and data trace, system trace information and accesses to the memory bus. TRACE32[®] displays time-correlated traces of multiple sources which had been stored in on-chip trace memory or emitted from a common trace port.



Backward Debugging

Trace-based debugging allows developers to reconstruct the core context for any trace sampling point. You can re-debug a traced program section and watch how memory, registers and variables are changing. You can even step back in time and get a true high-level language (HLL) trace listing showing register and stack variables.

PAST	Step 🛛 😕 Over 🗍	🛃 Diverge	e) 🛷 Return) 🕑 Up	K Step	🗧 🗧 Over	🔶 Entry	📙 Off 🛛 🔀 I	Mode
	addr/line c	ode	label	mnemon	ic		comment		
		35C0000		cmpeq	r12,#0x0)	; r12,#0		
0 (NOW)	SR:0000F6F8 1			bne	0xF730				
0 (11011)	100		AC terms						
	186		dcval =], quantpt	r[DCTSIZ	E*0
	SR:0000F6FC	ID2C0F0		ldrsh	r12,[r2]				-
	Source Le	wal Trac	o Dienlav	Showing	Local Vari	ahlae		— п	
							U.	•	
	🔑 Setup 🔰	CIS	Goto	Find	TREE	Chart	Chart	More	Le
	record		Goto	Find		Chart	Chart	More	Le:
					TREE	Chart	Chart	More	_
	record	inptr	· = 0x0001	LE4A6		Chart	Chart	•(_
	record 000140 -	inptr	r = 0x0001	LE4A6 0001B934					-
	record	inptr quant	= 0x0001 ptr = 0x0 mp3 = DEQ	LE4A6 0001B934			wantptr[DC1		-
	record • 000140 • 231	inptr quant	r = 0x0001	LE4A6 0001B934					-
	record	inptr quant	r = 0x0001 ptr = 0x0 mp3 = DEQ = -20	LE4A6 0001B934					-

Tracing and Profiling

Powerful trace filter and trigger enable you to get the trace information of interest into the few kilobytes of on-chip trace memory or the up to 8 GB of TRACE32's off-chip trace memory. For long-term trace, streaming modes can be used which convey the trace data to the hard disk or to an application running on the host while recording.

This provides the capability of searching for bugs which only show up when running in real-time. In addition, there are various analysis functions, for example: run-time statistics on functions and tasks, analysis of the function nesting, or cache performance.

. Groups E Co	nfig 🔒 🗛 Go	to	ed Resting	Chart					
uncs: 10.	total:	1.036s							
	1		1	(1993)	1	14	70/	50/	1.00
							276	576	10%
						6.807%			
						6.103%			
— func4						5.516%			
- func20									
Caller G	rapn					_		~	
🌽 Setup	Groups	Config	Goto	Find	▲ In ► O	ut KNFull			
	-846.0	5 Oms	-846.000	Oms	-845.950	ms -	845.900ms		
range 🖬							1 .		
main 🐠			Sec 557 - 5						
func2								68 D	
						2.0		S. 18	
			· ·	· · · · · · · · · · · ·			1.000	87 - E	
		195 No 397	2 6	2 - 1	_ · · ·	St 15 15	2003-00	2 - C	
	10 10 10		3 B	8 8 B			18 18		
	2 A A						1.5	- 53	
	15	16	× 2	9 16 18	 a b a a b a a a b a a		10 18	38	
sieve w	4.3 SA	0 K. 10		an 10 10	10 14 1 4			1	
	uncs: 10. ree func2 func2a func2a func4 func7 func8 func20 caller G Setup range func2 to func2 func20 caller G func2 to func2 fu	uncs: 10. total: ree total func2 10. total: func2 10. total: f	uncs: 10. total: 1.036s ree total min func2 func2a func2a func2a func2a func2	uncs: 10. total: 1.036s ree total min max func2 225.071ms 18.500us 18.500us func2 133.826ms 11.000us 11.000us func4 57.180ms 4.700us 4.700us func7 121.660ms 10.000us 10.000us func8 55.959ms 4.600us 1.100us func8 13.382ms 1.100us 1.100us func8 4.700us 4.700us 4.700us func8 4.700us 4.700us 4.700us func8 13.382ms 1.100us 1.100us func9 476.868ms 30.200us 30.200us Caller Graph Setup iiGroups Config Goto range func2 4.700us 4.600us func2 4.700us 4.700us 4.700us func2 4.700us	uncs: 10. total: 1.036s main 1.036s func2 225.071ms 18.500us 18.500us func2a 133.826ms func4 57.180ms func7 121.660ms func20 13.382ms func20 146.868ms 30.200uc 30.200uc 30.200uc 30.200uc func20 11.600s func20 11.600s func20 11.600s func20 11.600s func20 11.600s func20 11.600s	uncs: 10. total: 1.036s ree total min max avr count □ main 1.036s - 1.036s 1.036s 1.00/1) □ func2a 133.826ms 11.000us 11.000us 11.000us 12166. - func4 57.180ms 4.700us 4.700us 5.200us 5.200us 12166. - func7 121.660ms 10.000us 10.000us 10.000us 12166. - func20 13.382ms 1.100us 1.100us 1.100us 12166. - func20 13.382ms 1.100us 1.100us 1.100us 12165. - func20 13.382ms 1.100us 1.100us 1.100us 1.2165. - func20 13.382ms 1.100us 1.100us 1.100us 1.2165. - func20 13.382ms 1.100us 1.100us 1.100us 1.2165. - a46.050ms -846.000ms -845.950 - a46.050ms -845.950 - a46.050ms -845.950 - a46.050ms -845.950 - a46.050ms -84	uncs: 10. total: 1.036s ree total min max avr count intern% 1% ■ func2 ■ func2a □ func2b □ func2b □ func2 i i 1.036s 1.036s 1.036s 1.036s 1.036s 1.036s 1.036s 1.036s 1.036s 1.036s 1.036s 1.036s 1.036s 1.000us 12166. 6.807% 5.200us 5.200us 5.200us 5.200us 12166. 6.103% 4.700us 4.700us 12166. 5.516% 121.660ms 10.000us 10.000us 10.000us 1.000us 12166. (0/1) 1.737% 5.398% - func2 13.382ms 1.100us	uncs: 10. total: 1.036s ree total intern% 1.036s intern% 1.000us intern%	uncs: 10. total: 1.036s ree total: 1.036s Imain 1.036s 1.000us Imain 1.036s 1.036s Imain 1.036s 1.000us Imain 1.000us 10.000us Imain 1.100us 1.100us Imain 1.100us 1.100us Imain 1.100us 1.100us Imain 1.100us 1.100us 1.100us Imain 1.100us 1.100us 1.100us 1.100us Imain Imain Imain Imain Imain Imain Imain Imain Imain Imain Imain Imain Imain Imain I

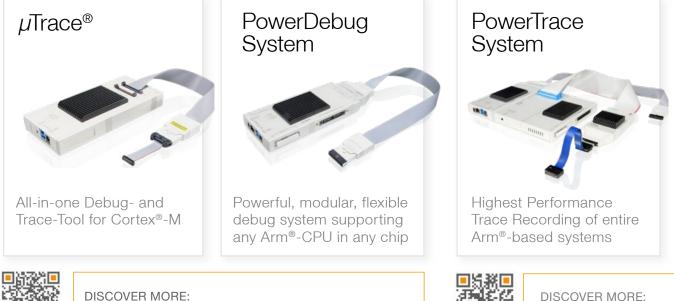
Trace-based Code Coverage

Real-time code coverage verifies even complex metrics like MC/DC without prior code instrumentation. It is suitable to assist in the development of software for safety-related systems in compliance with ISO 26262, DO-178C, IEC 61508, IEC 62304 and EN 50128. Qualification kits are available.

	Multi-Level Co	de Coverage	e Overview			\Box \times	
		🕲 List	+ Add 🛛 🔀 L	oad 😰 Save	🛛 🛛 Init		
	rce/c/i	if00.c \116 f00.c \1720 f00.c \2122	partial ok 1 ok 1	executed 0% 60.869%	50%	100 branches 12.500%	
	rce/c/i rce/c/i	f00.c \2326 f00.c \2728	taken never	50.000%		50.000%	
	rce/c/i	f00.c \2929	never	0.000%			
Detailed	High-Level Lang	f00.c \2929	never	0.000%	- 0	× Code Co	overage
	rce/c/i	foo.c \2929 guage and AS	never				
	High-Level Lang	foo.c \2929 guage and AS	SM Analysis Label Mnem				ric —
Coverage	High-Level Lang	f00.c \2929 guage and AS Code \\test_arm\if00	SM Analysis Label Mnem	onics Comm		tion SourceMet	ric —
Coverage never	High-Level Lang Addr/Line	f00.c \2929 guage and AS Code \\test_arm\if00 <i>st</i>	SM Analysis Label Mnemo D\if00.c	onics Comm		tion SourceMet	ric <u> </u>
Coverage never never	urce/c/i High-Level Lang Addr/Line 29	f00.c \2929 guage and AS Code \\test_arm\if0(<i>st</i> E3 A0 00 01	SM Analysis Label Mnem D\if00.c ate = true;	onics Comm		tion SourceMet ObjectCode Statement Decision	ric <u> </u>
Coverage never never never	rce/c/i High-Level Lang Addr/Line 29 NSR: 800081cc	f00.c \2929 guage and AS Code \\test_arm\if0(<i>st</i> E3 A0 00 01	SM Analysis Label Mnem D\if00.c ate = true; mov	onics Comm : r0,#0x1		tion SourceMet ObjectCode Statement Decision CONDition	ric <u> </u>
	NSR: 800081D0	f00.c \2929 guage and AS Code \\test_arm\if0(st E3A00001 E12FFF1E }	SM Analysis Label Mnem D\if00.c ate = true; mov	onics Comm r0,#0x1 r14		tion SourceMet ObjectCode Statement Decision	ric <u> </u>

DISCOVER MORE: www.lauterbach.com/use-cases/tool-qualification

HARDWARE-Based Solutions for all Arm[®]-Cortex[®]-Cores

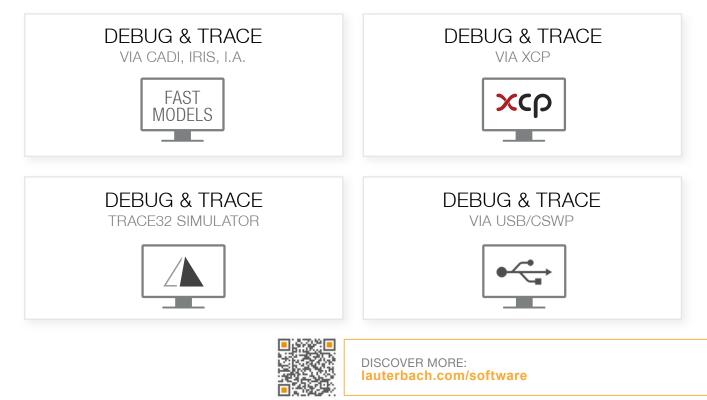




DISCOVER MORE: lauterbach.com/debugger



SOFTWARE-Only Solutions



lauterbach.com

lauterbach.com/trace