TRACE32®

Debug & Trace

ARM®

LAUTERBACH
DEVELOPMENT TOOLS

ARM® support at a glance

More than 18 years of experience in ARM debugging enable us to provide the best-in-class debug and trace tools for ARM based systems:

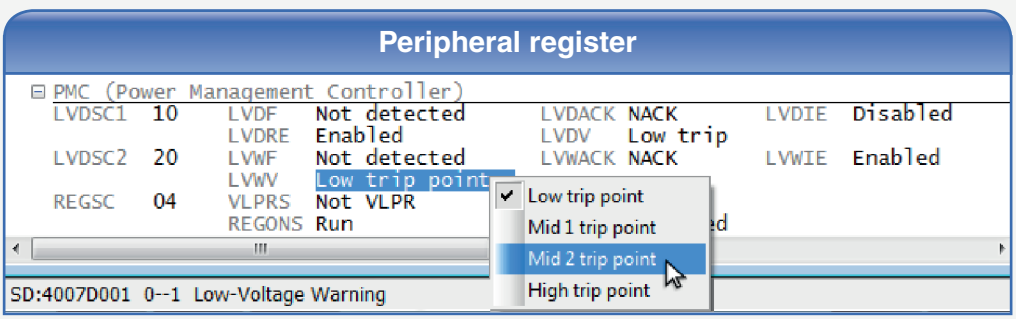
- Multicore debugging and tracing for any mixture of ARM and DSP cores
- Support for all CoreSight components to debug and trace an entire SoC
- Powerful code coverage and run-time analysis of functions and tasks
- OS-aware debugging of kernel, libraries, tasks of all commonly used OSes

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 and VFP units, to OS-aware debug and trace concepts for 32-bit and 64-bit multicore systems.

Peripheral register

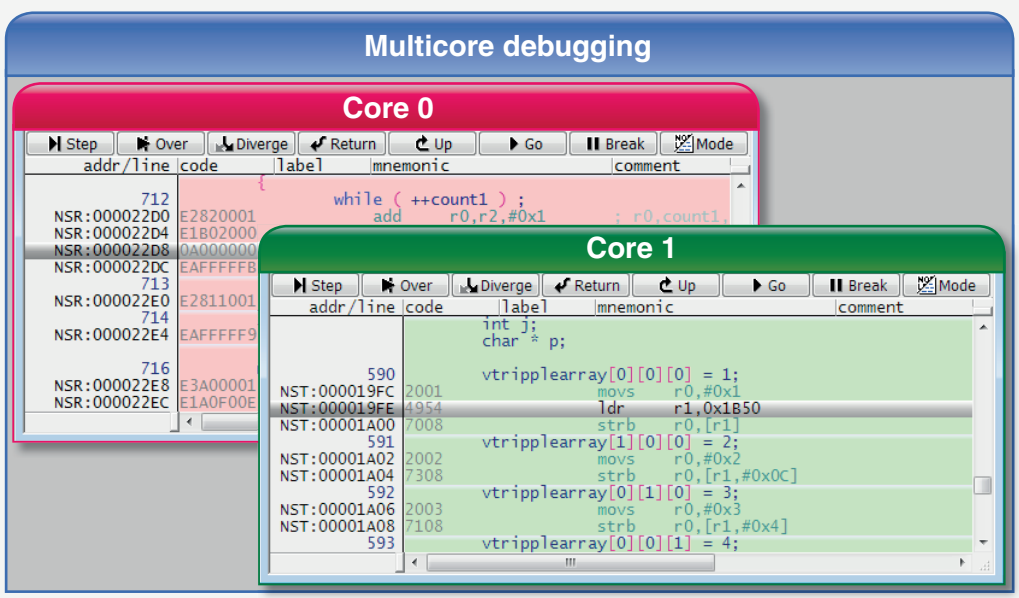


The peripheral registers on the chip are displayed on a logical level. The function bits can easily be interpreted and modified. A pull-down menu helps to select the peripheral module you want to inspect.

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.

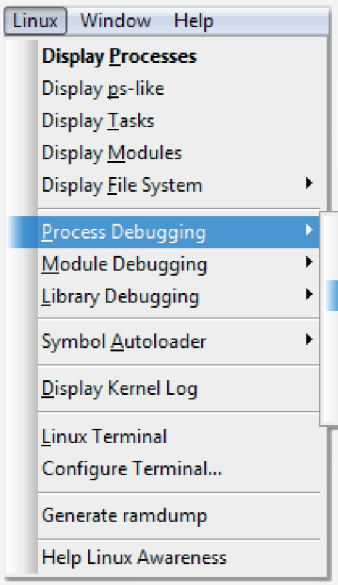
Multicore debugging



OS-Aware Debugging and Tracing

Debugging of system and application software of all commonly used operating systems:

- Stop Mode Debugging: the whole system is stopped to analyse the overall system state
- Run Mode Debugging: the process of interest is stopped whilst kernel and other processes keep going
- Full support for symmetric (SMP) and asymmetric (AMP) multiprocessing
- Non-intrusive access to task lists and other kernel information
- Stack coverage and call hierarchy of waiting tasks
- Statistics and graphic display of task and function run-time



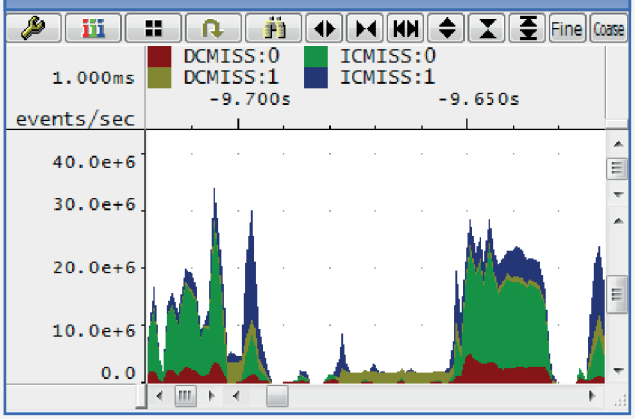
List of processes

magic	command	#thr	state	spaceid	pids
C08EA540	swapper/0	56.	running	0000	0. 2. 3. 4. 6. 7. 8. 9. 10.
EFFDFBC0	init	-	sleeping	0001	1.
EFC7DC00	ueventd	-	sleeping	0359	857.
EFC82960	sh	-	sleeping	04F3	1267.
EFC7F0A0	servicemanager	-	sleeping	04F4	1268.
EFF769E0	vold	3.	sleeping	04F5	1269. 1288. 1316.
EFC7F380	netd	7.	sleeping	04F6	1270. 1473. 1474. 1477. 147
	debuggerd	-	sleeping	04F7	1271.
	surfaceflinger	8.	running	04F8	1272. 1488. 1491. 1492. 149
	zygote	4.	sleeping	04F9	1273. 1880. 1881. 1882.
	drmserver	2.	sleeping	04FA	1274. 1483.
	mediaserver	5.	sleeping	04FB	1275. 1484. 1485. 1486. 154
	dbus-daemon	-	sleeping	04FC	1276.
	installd	-	sleeping	04FD	1277.
	keystore	-	sleeping	04FE	1278.
	uim	-	sleeping	0501	1281.
	adbd	4.	sleeping	05D2	1490. 1497. 1498. 1499.
	system_server	67.	current	05E4	1508. 1512. 1513. 1514. 151
	com.android.syst	10.	sleeping	0624	1572. 1576. 1577. 1578. 157
	android.process.	12.	sleeping	0633	1587. 1591. 1592. 1593. 159
	com.android.inpu	10.	sleeping	0641	1601. 1607. 1609. 1610. 161
EFD19960	com.android.phon	21.	sleeping	0652	1618. 1623. 1624. 1625. 162
ED425320	com.android.laun	12.	sleeping	065C	1628. 1635. 1637. 1640. 164
EFD886C0	com.android.smsp	10.	sleeping	068C	1676. 1686. 1688. 1689. 169
ED5290A0	android.process.	15.	sleeping	0692	1682. 1691. 1692. 1694. 169
ED912900	com.android.desk	12.	sleeping	06BC	1724. 1726. 1728. 1730. 173
ED529940	com.android.prov	12.	sleeping	06D7	1751. 1753. 1754. 1756. 175
ECF36C60	com.android.exch	14.	sleeping	06EA	1770. 1774. 1775. 1776. 177

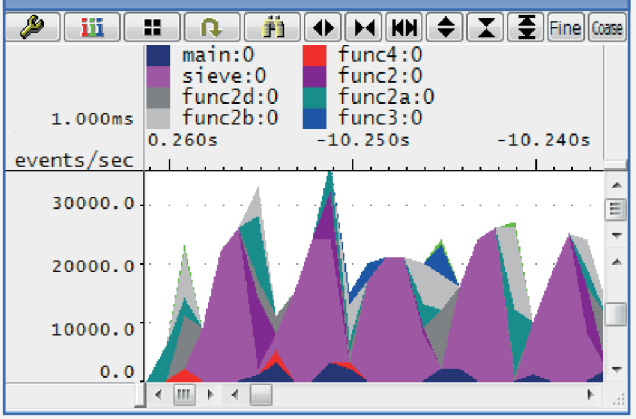
Utilization of the Performance Monitors

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.

Cache misses of a dual-core system

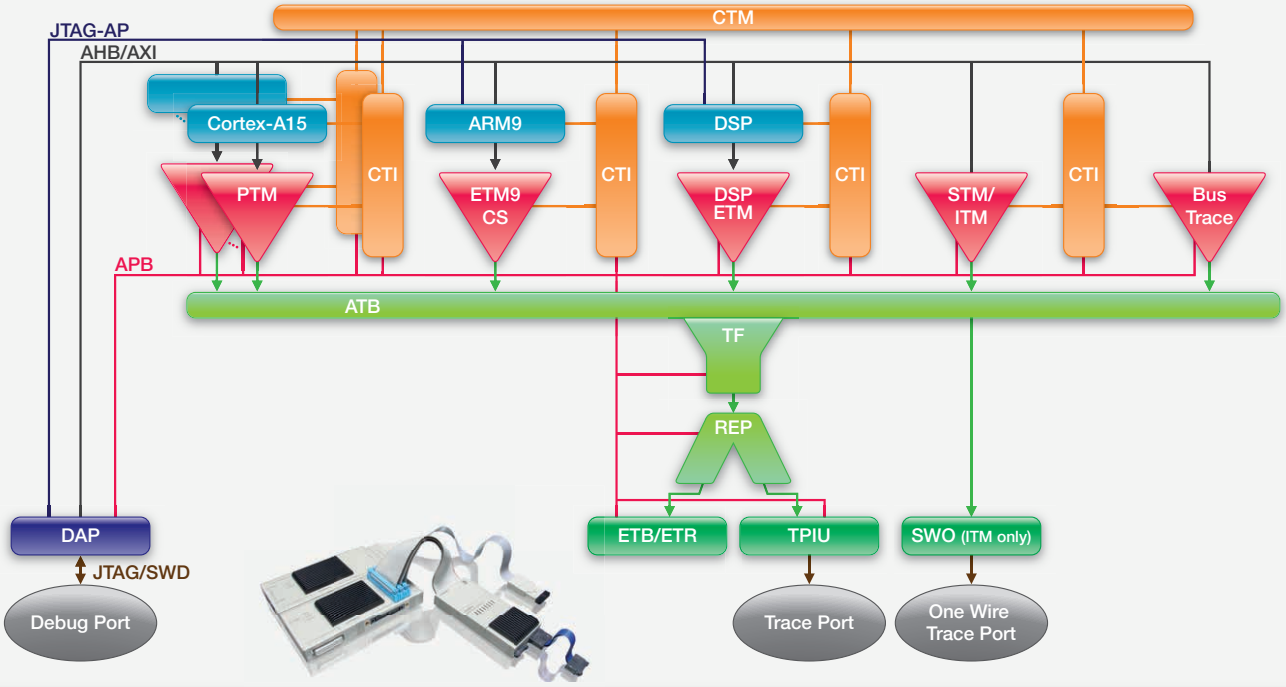


Data cache misses by function



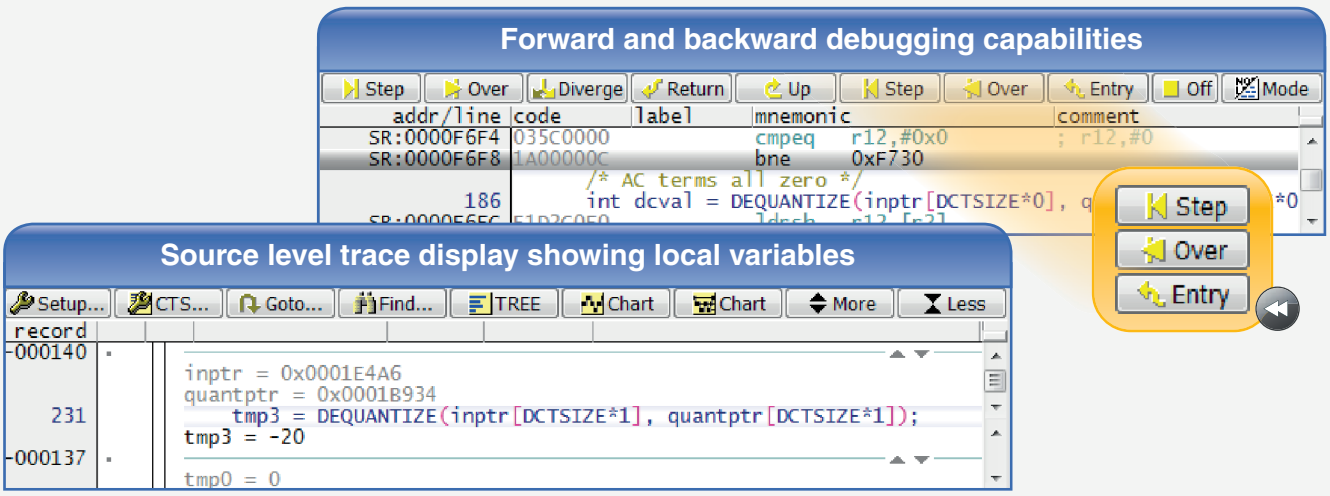
Full Support for ARM® CoreSight™ Technology

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

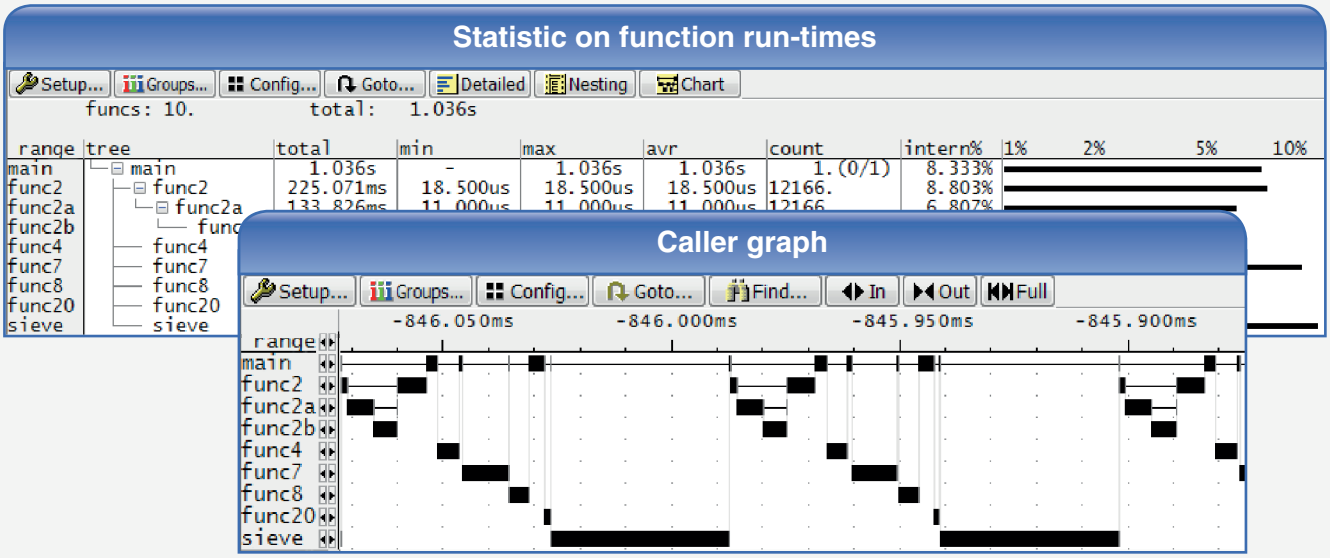
Trace-based debugging allows developers to reconstruct the core context for any trace sampling point. You can re-debug a traced program section and watch how memory, registers and variables are changing. You can even step back in time and get a true high-level language trace listing showing register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



Tracing and Profiling

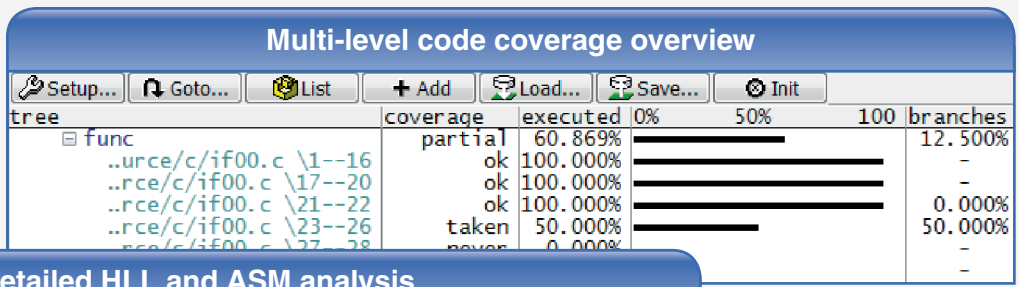
Powerful trace filter and trigger enable you to get the trace information of interest into the few kilobytes of on-chip trace memory or the up to 4 GB of TRACE32's off-chip trace memory. For long-term trace, streaming modes can be used which convey the trace data to the hard disk or to an application 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.



Trace-based Code Coverage

An integrated real-time code coverage facility verifies statement and decision/branch coverage 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, and IEC 62304.



Detailed HLL and ASM analysis

Coverage	Addr/Line	Code	Label	Mnemonics	Comment/Justification
		\\test_arm\if00\if00.c			
never	29	*state = true;*			
never	NSR:800081CC	E3A00001		mov	r0,#0x1
never	NSR:800081D0	E12FFF1E		bx	r14
not taken	30	*}*			
not taken	31	*else if (incr < decr)*			
ok	NSR:800081D4	E1510002		cmp	r1,r2

ARM® support at a glance

More than 18 years of experience in ARM debugging enable us to provide the best-in-class debug and trace tools for ARM based systems:

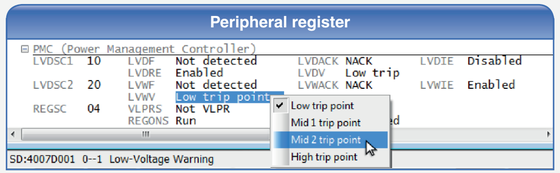
- Multicore debugging and tracing for any mixture of ARM and DSP cores
- Support for all CoreSight components to debug and trace an entire SoC
- Powerful code coverage and run-time analysis of functions and tasks
- OS-aware debugging of kernel, libraries, tasks of all commonly used OSes

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 and VFP units, to OS-aware debug and trace concepts for 32-bit and 64-bit multicore systems.

Peripheral register

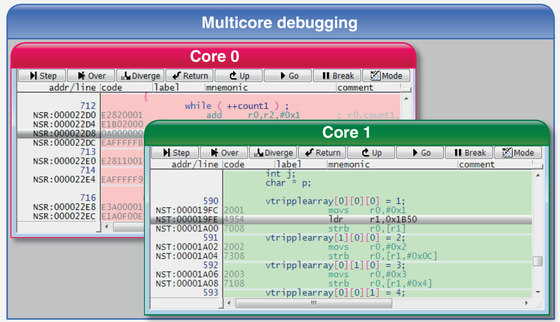


The peripheral registers on the chip are displayed on a logical level. The function bits can easily be interpreted and modified. A pull-down menu helps to select the peripheral module you want to inspect.

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.

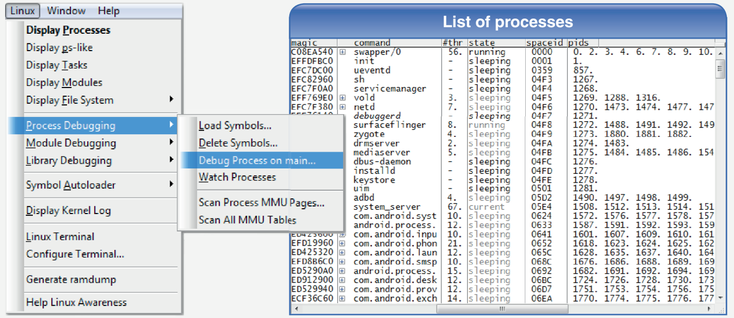
Multicore debugging



OS-Aware Debugging and Tracing

Debugging of system and application software of all commonly used operating systems:

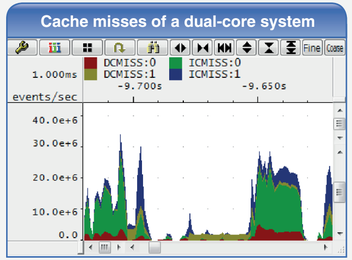
- Stop Mode Debugging: the whole system is stopped to analyse the overall system state
- Run Mode Debugging: the process of interest is stopped whilst kernel and other processes keep going
- Full support for symmetric (SMP) and asymmetric (AMP) multiprocessing
- Non-intrusive access to task lists and other kernel information
- Stack coverage and call hierarchy of waiting tasks
- Statistics and graphic display of task and function run-time



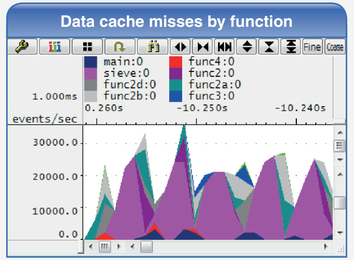
Utilization of the Performance Monitors

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.

Cache misses of a dual-core system

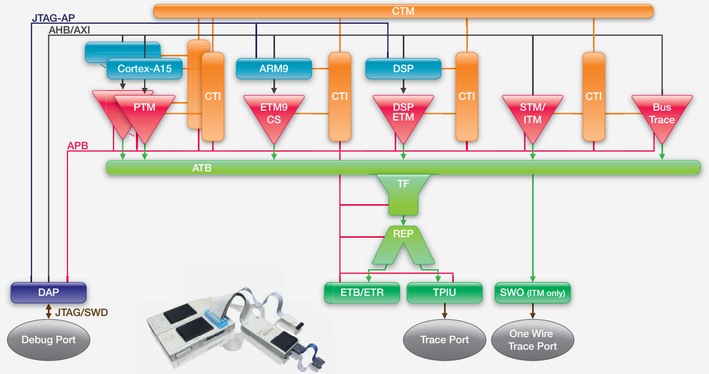


Data cache misses by function



Full Support for ARM® CoreSight™ Technology

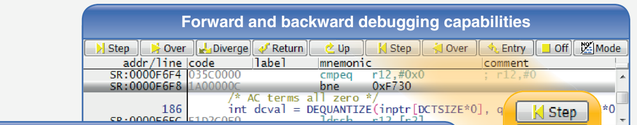
ARM CoreSight technology offers the chip designer various components to extend the core's debug functionality with the objective of debugging and tracing an entire system-on-chip. A joint JTAG or Serial Wire Debug interface allows real-time access to the on-chip buses and control of the cores and the 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.



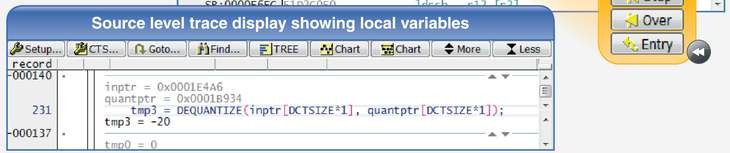
Backward Debugging

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 trace listing showing register and stack variables.

Forward and backward debugging capabilities



Source level trace display showing local variables



Debugger and Trace for ARM®

Power Debug USB3

Debugger

Debugging via JTAG or Serial Wire Debug interface.

Optional support for on-chip tracing (ETB, TMC).

Debug and Trace for Small Trace Ports

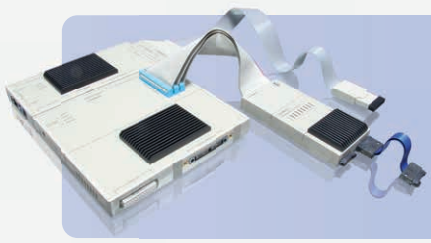
Optimized for, but not limited to Cortex-M based devices. It can capture an up to 4-bit wide Cortex-M or System Trace Port.

Combi Probe

Power Trace PX

Debug and Trace

Debug and trace solution for parallel (TPIU) or serial (HSSTP) trace ports.



High-End Debug and Trace

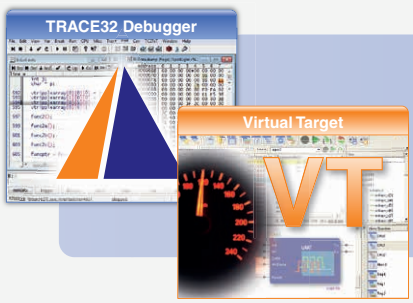
Debug and trace solution with large trace memory for fast parallel (TPIU) or serial (HSSTP) trace ports.

Power Trace II

µTrace®

Debug and Trace for Cortex®-M

Cost-effective all-in-one debug and trace solution for Cortex-M based chips, only.



Debugger for Virtual Targets

Debugging and tracing using TRACE32 on software models before a first hardware prototype is available.

Front-End

For more information visit: ***www.lauterbach.com/bdmarm.html***

Lauterbach GmbH
Altlaufstraße 40 • 85635 Höhenkirchen • Tel: +49 8102 9876 0 • sales@lauterbach.com
Visit our official Youtube channel and discover our new tutorials: http://www.youtube.com/user/lauterbachgmbh

LAUTERBACH
DEVELOPMENT TOOLS