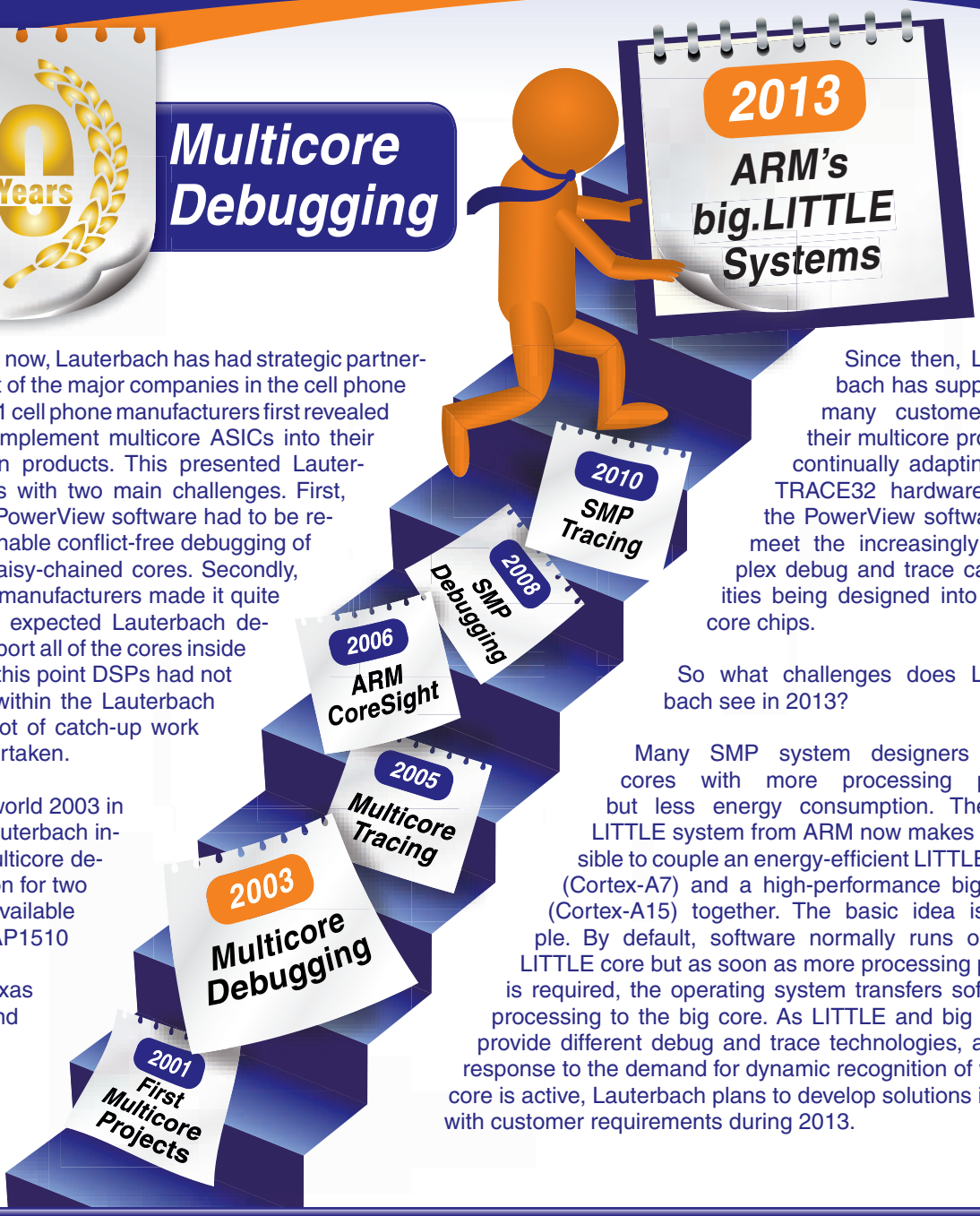
LAUTERBACH
DEVELOPMENT TOOLS

NEWS 2013

10 Years

Multicore Debugging



For many years now, Lauterbach has had strategic partnerships with most of the major companies in the cell phone industry. In 2001 cell phone manufacturers first revealed their plans to implement multicore ASICs into their next generation products. This presented Lauterbach designers with two main challenges. First, the TRACE32 PowerView software had to be re-developed to enable conflict-free debugging of two or more daisy-chained cores. Secondly, the cell phone manufacturers made it quite clear that they expected Lauterbach debuggers to support all of the cores inside an ASIC. Until this point DSPs had not been a focus within the Lauterbach portfolio so a lot of catch-up work had to be undertaken.

At embedded world 2003 in Nuremberg, Lauterbach introduced its multicore debugging solution for two commercially available chips: the OMAP1510 (TMS320C55x, ARM9) from Texas Instruments and the S-GOLD (ARM9, OAK DSP) from Infineon.

Since then, Lauterbach has supported many customers in their multicore projects, continually adapting the TRACE32 hardware and the PowerView software to meet the increasingly complex debug and trace capabilities being designed into multicore chips.

So what challenges does Lauterbach see in 2013?

Many SMP system designers want cores with more processing power but less energy consumption. The big.LITTLE system from ARM now makes it possible to couple an energy-efficient LITTLE core (Cortex-A7) and a high-performance big core (Cortex-A15) together. The basic idea is simple. By default, software normally runs on the LITTLE core but as soon as more processing power is required, the operating system transfers software processing to the big core. As LITTLE and big cores provide different debug and trace technologies, and in response to the demand for dynamic recognition of which core is active, Lauterbach plans to develop solutions in line with customer requirements during 2013.

NEWS 2013 CONTENTS

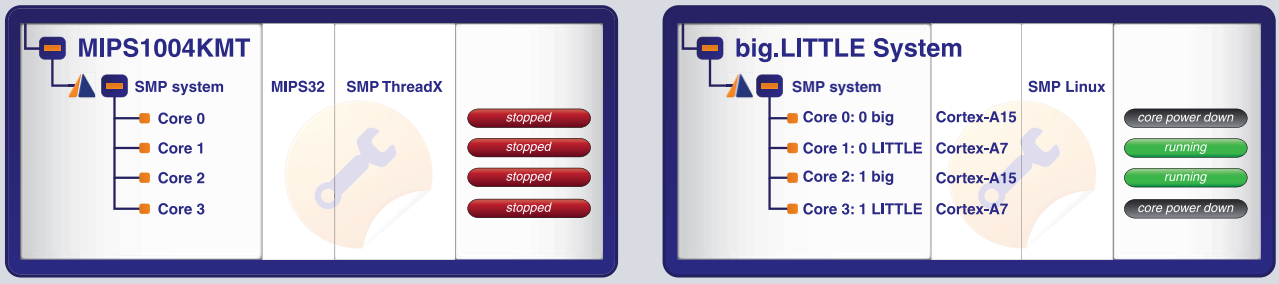
TRACE32 Multicore Strategy

Lauterbach has supported debugging and tracing of multicore chips for more than 10 years.

Flexibility

A long-standing aim for Lauterbach has been to make its TRACE32 hardware and software as flexible as possible. Every core combination, every multicore topology, every multicore operation mode, and even the most complex debug and trace infrastructures are all supported by TRACE32. This flexibility also means that TRACE32 supports debugging and tracing of both AMP systems and SMP systems. For an overview of the most important differences in debugging these two systems, see the tables on pages 2 and 3.

SMP multicore configurations

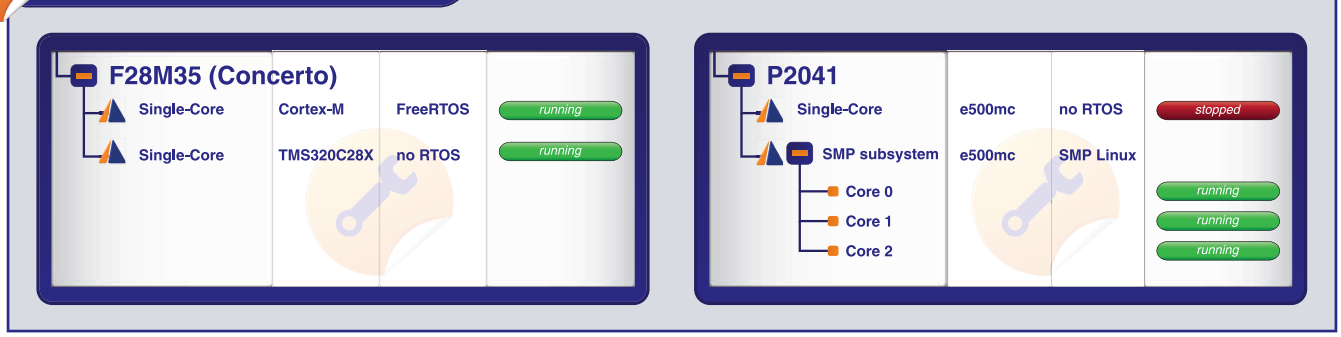


SMP System — Symmetric MultiProcessing	
Target System Layout	An SMP system consists of two or more cores. These are usually identical, or at least instruction-set compatible.
Task Assignment/ Operating System	A single SMP operating system assigns tasks to the cores (dynamically or statically).
Number of TRACE32 Instances	Only one TRACE32 instance is started for debugging an SMP system. This instance controls all cores and displays all information.
Synchronized Core Start/Stop	All cores are started and stopped synchronously.
On-Chip Breakpoints	On-chip breakpoints are programmed in parallel in the debug registers of all cores.
Trace Filters and Triggers	Trace filters and triggers are programmed in parallel in the trace registers of all cores.
Trace Display	Trace information can be displayed either for all cores at once or individually for each core.
Profiling	Runtime can be measured for each core individually or for the whole system.

70+ Supported Processor Architectures

Another important principle within Lauterbach is to support a wide variety of processor architectures including: standard cores, DSPs, FPGAs with embedded soft-cores, and configurable cores. Each new core is integrated into TRACE32 in a way that ensures that the core can be debugged as a single-core subsystem within an AMP system. For any architecture, SMP debugging and tracing is added to the debugger as soon as the first SMP-capable chip is launched by the chip manufacturer. For SMP-capable chips it is particularly important to adapt the TRACE32 OS-awareness. This adaptation must take into account whether the SMP operating system assigns processes to the cores dynamically during runtime, or whether some or all of the processes are assigned statically.

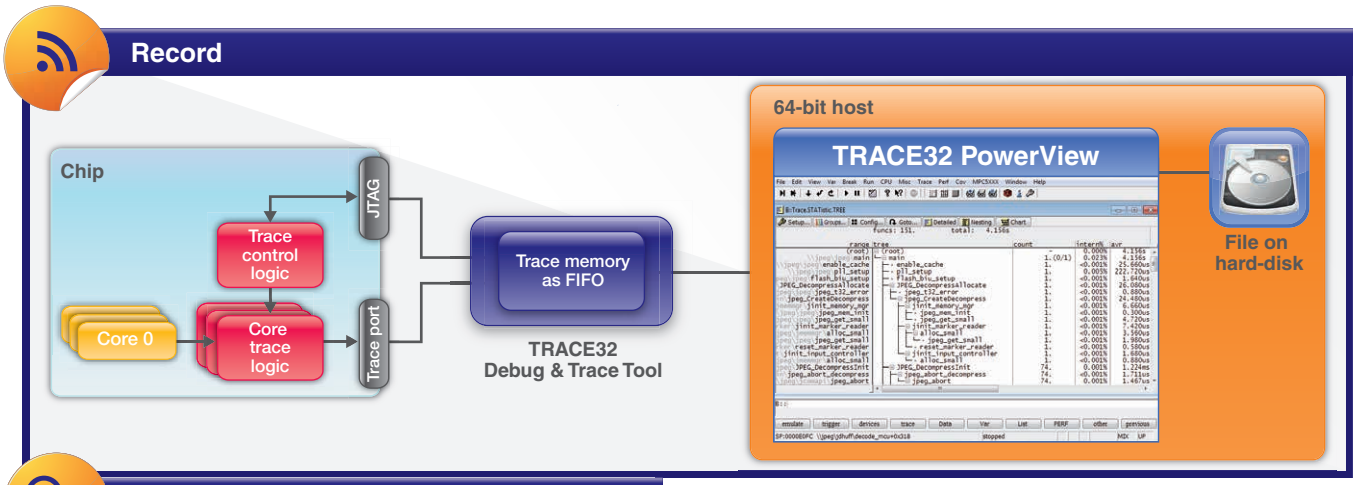
AMP multicore configurations

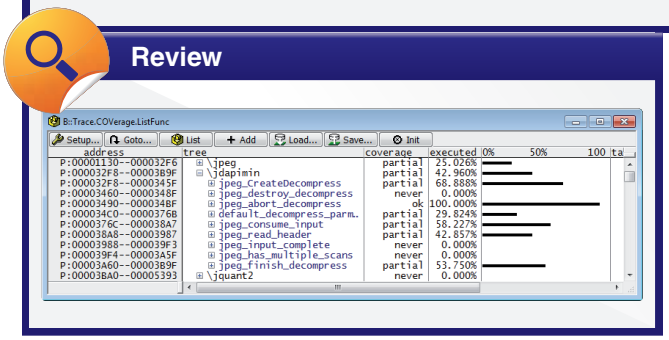


AMP System — Asymmetric MultiProcessing	
Target System Layout	An AMP system consists of several subsystems: individual cores and/or SMP systems.
Task Assignment/ Operating System	Tasks are assigned to the subsystems during the design phase. An operating system only controls one subsystem.
Number of TRACE32 Instances	Two or more TRACE32 instances are started for debugging an AMP system. Each TRACE32 instance controls a complete subsystem and displays its information.
Synchronized Core Start/Stop	All subsystems can be started and stopped synchronously (configurable).
On-Chip Breakpoints	On-chip breakpoints are programmed independently for each subsystem.
Trace Filters and Triggers	Trace filters and triggers are programmed independently for each subsystem.
Trace Display	A TRACE32 instance displays trace information for all the cores controlled by that instance.
Profiling	A TRACE32 instance can measure the runtime for all the cores controlled by that instance. A global timestamp allows to directly display the timing connection between the subsystems.

Code Coverage: Documenting Results

As of November 2012, TRACE32 PowerView provides new features to document the results of code coverage analysis. Features that have newly been added are a comment function and an XML export.





Trace-Based Code Coverage

Proof of statement coverage and decision coverage is often required for quality assurance of products in markets such as medical and avionics. For many embedded systems, the specification requires that highly-optimized code is tested in real-time. Code instrumentation and runtime falsification are forbidden. Lauterbach's trace-based code coverage system provides customers with proof of statement coverage and decision coverage. However, the processor or multicore chip used must fulfill the following requirements:

The target cores must have an on-chip trace logic that generates information about the execution of instructions on the cores. At the same time, the processor or multicore chip must have a trace port with sufficient bandwidth so that the complete trace information can be recorded by an external tool.

For average data transfer rates up to 60 MB/s, the trace data can be streamed to the host computer during recording. This means that several TBytes of trace information can be recorded during each test run.

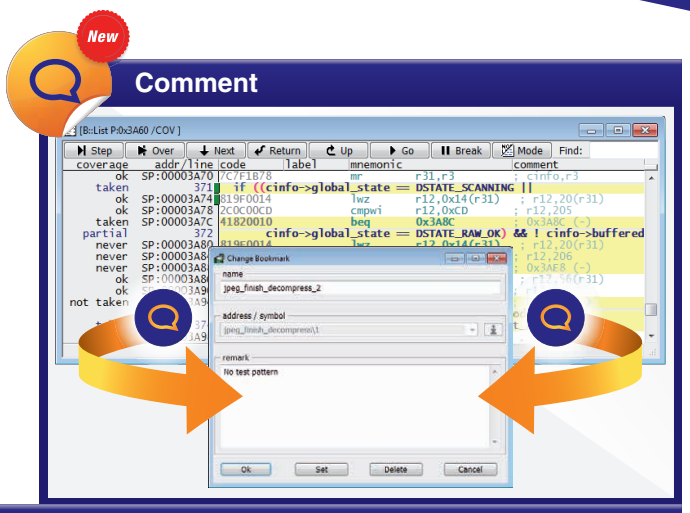
As the trace information is available at assembler level, the following proofs can be provided:

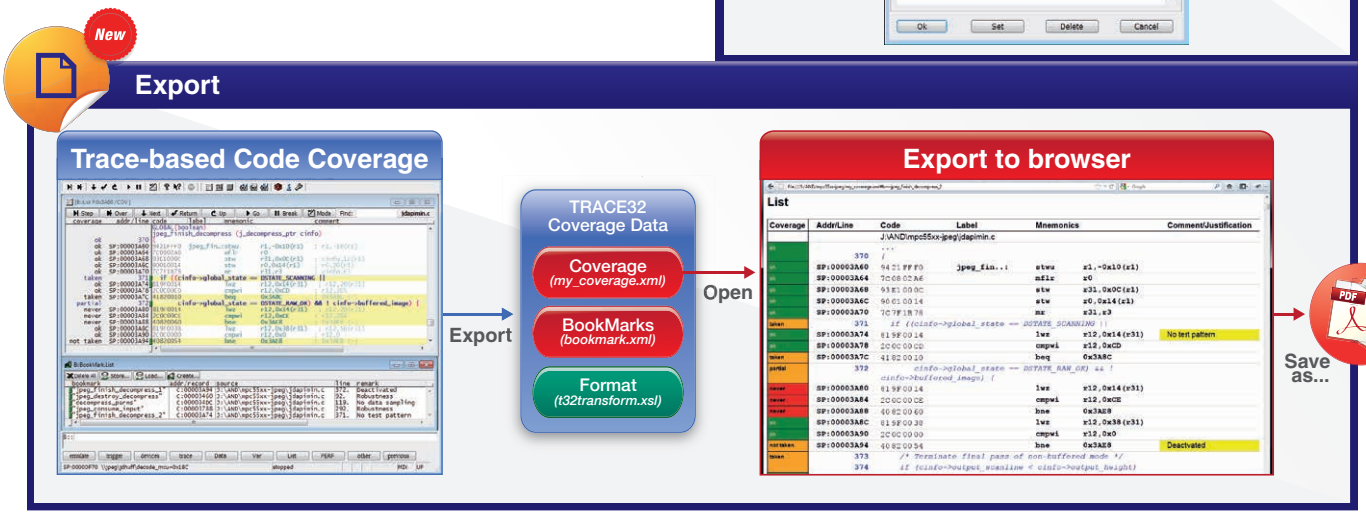
- **Object Statement Coverage**
  It proofs that each line of assembly code was executed at least once during the system test.
- **Object Branch Coverage**
  It proves that each conditional branch was both taken and not taken at least once.

For the lines of high-level language code, statement coverage and decision coverage can easily be derived from this analysis.

Comment Function

Generally, developers write test cases to prove that an embedded system complies fully with all requirements. These are then the basis for the system test.





To collect data for code coverage analysis, the trace tool records all information about the instructions executed during the system test (**Record**). The recorded trace information is managed by TRACE32 PowerView in a code coverage database. This provides numerous ways in which the user can analyze and display the code coverage results (**Review**).

After testing is completed, the tester has to decide:

- Is a section of code that was not executed supposed to fulfill a requirement? If so, a suitable test case must be created for the next system test run.
- If a section of code that was not executed is supposed to fulfill a requirement that is not testable in the current system configuration, the new TRACE32 comment function can be used to explain the reason it's included (**Comment**).
- Is there any "dead" code? This must be removed from the software.

XML Export

After system test completes, you need to document the results of the code coverage analysis. Exporting the results in XML format is now supported in TRACE32 PowerView. These files can be exported:

1. The assembly code and high-level language code, as well as code-coverage tagging (my_coverage.xml).
2. High-level results of the code coverage analysis e.g. module or function coverage.
3. Comments that explain why individual code sections are permissible, even though they were not executed during the test (bookmark.xml).

Lauterbach provides a transformation file for an intuitive display of the results in a Web browser (t32transform.xsl). If necessary, the results can also be saved as a PDF file.

New Processors/Chips

Altera	**Cortex-A/-R** • Cyclone V SoC
Analog Devices	**Cortex-M** • ADuCM36x
AppliedMicro	**PPC40x** • PPC405EX, PPC405EXr **PPC44x** • SMP for APM PacketPro
ARM	**Cortex-A5x (ARMv8)** • Cortex-A53 • Cortex-A57
Atmel	**Cortex-M** • ATSAM4
Axis	**MIPS32** • ARTPEC-4
Broadcom	**MIPS32** • BCM47186 • BCM6318, BCM6828 • BCM7346, BCM7356 • BCM7418, BCM7425
BroadLight	**MIPS32** • BL25580
CEVA	**CEVA-X** • CEVA-XC323 **TeakLite-III** • CEVA-TeakLite-4
Energy Micro	**Cortex-M** • EFM32LGxxx, EFM32WGxxx • EFM32ZGxxx
Freescale	**ColdFire+/V1** • MCF51AC/AG/CN/EM • MCF51JE/JM/MM/QE • MCF51JF/JU/QM/QU **Cortex-A/-R** • Vybrid F Series **Cortex-M** • Kinetis L • Vybrid Series **MPC85XX/QorIQ e500** • P1010, P1012, P1014 • P1017, P1021, P1023 **QorIQ 32-Bit** • P2040, P2041 **QorIQ 64-Bit** • B4220, B4420, B4860 • P5021, P5040, T10XX • T2080, T2081, T4160, T4240
Freescale (Cont.)	**PX-Series** • PXD1005, PXD1010, PXD2020 • PXN2020, PXN2120, PXR40xx • PXS2005, PXS2010, PXS30xx **Qorivva MPC5xxx** • MPC5743K, MPC5744K • MPC5744P, MPC5746M, • MPC5748G, MPC5777M **S12Z** • S12ZVH, S12ZVM **StarCore** • B4220, B4420, B4860
Hilscher	**ARM9** • NETX 51
Infineon	**Cortex-M** • XMC4000 Family • TC2D5T/D7T, TC2D5TE/D7TE • TC275T/277T, TC275TE/277TE **TriCore** • TC2D5T/D7T, TC2D5TE/D7TE • TC275T/277T, TC275TE/277TE
Intel®	**Atom™/x86** • Atom Z2460/CE2600/N2800 • Core i3/i5/i7 3rd Generation
Marvell	**ARM11** • MV78130v6, MV78160v6 • MV78230v6, MV78260v6 **Cortex-A/-R** • MV78130v7, MV78160v7 • MV78230v7, MV78260v7
Mobileye	**MIPS32** • EyeQ3
NEC	**MIPS32** • EMMA3 Series
NVIDIA	**Cortex-A/-R** • TEGRA 3
NXP	**Beyond** • JN5168 **Cortex-M** • LPC43xxx, LPC800
Renesas	**Cortex-A/-R** • R-Car H1 **MIPS32** • RT3352 **RH850** • RH850/E1x, RH850/F1x **RL78** • RL78D1A/F1x/G1x/I1A/Lxx **RX** • RX630, RX631, RX63N **SH** • SH7267

New Processors/Chips

Renesas (Cont.)	**V850** • V850E2/Fx4-L • V850E2/Mx4 Multicore
Samsung	**Cortex-A/-R** • Exynos 4212, Exynos 4412 • Exynos 5250 • S5PV210
Sigma Designs	**MIPS32** • SMP8634, SMP8654
ST-Ericsson	**Cortex-A/-R** • DB8540 **MMDSP** • DB8540
STMicro-electronics	**Cortex-A/-R** • SPEAr1310, SPEAr1340 **Cortex-M** • STM32 F3, STM32 F4
STMicro-electronics (Cont.)	**SPC5xx** • SPC56AP60, SPC56AP64 • SPC560P54, SPC560P60 • SPC574K70, SPC574K72 • SPC574L74, SPC57EM80 • SPC57HM90
Synopsys	**ARC** • ARC-EM 1.1
Texas Instruments	**Cortex-A/-R** • RM4 Series **Cortex-M** • F28M35 Concerto • LM4F Series **MSP430** • MSP430FR5xx **TMS320C28X** • C28346 • F28022, F28027, F28M35 **TMS320C55X** • C5535 **TMS320C6x00** • C6655, C6657, C6713

UEFI Debugging for ARM

In 2012, Lauterbach increased support for debugging UEFI BIOS. The following UEFI BIOS variants are now supported:

- InsydeH2O for Atom and x86
- Intel BLDK for Atom and x86 New
- TianoCore for ARM/Cortex New

To enable UEFI debugging, a TRACE32 extension is required. For detailed information on UEFI debugging, see: **www.lauterbach.com/uefi.html**

Enhancements to Target-OS

- FreeRTOS for Beyond and ColdFire
- Linux for Beyond and x86 64-bit
- OSEK/ORTI SMP
- QNX for x86
- Quadros for CEVA-X
- RTX-ARM v4
- SMX for ColdFire
- SYS/BIOS for TMS320C6x00
- VxWorks for x86
- µC/OS-II for TMS320C28X
- µC/OS-III for SH

New Target-OS

DEOS for PowerPC	available
Linux for ARMv8 (64-bit)	planned
OKL4 5.0 for ARM	available
Windows Standard (XP, Vista, Win7, Win8) for x86 32/64-bit	planned
µT-kernel for ARM	available

embeddedworld2013 Exhibition&Conference
hall 4, booth 210
26 - 28 february 2013

µTrace for Cortex™-M Family

Starting June 2013, a low-cost debugger for the Cortex-M family will be available from Lauterbach. Due to the high market penetration of Cortex-M processors an **all-in-one solution** has been developed, that will provide the following features:

µTrace Characteristics

- Support for more than 1000 different Cortex-M processors
- USB 3 interface to the host computer
- Standard JTAG, Serial Wire Debug, and cJTAG
- 256 MByte trace memory
- 34-pin half-size connector for target hardware and adapters for a wide variety of other connectors
- Voltage range 0.3V to 3.3V, 5V tolerance

Debug Features

- C/C++ debugging
- Simple and complex breakpoints
- Read and write memory during program runs
- Flash programming
- OS-aware debugging
- Multicore debugging of two or more Cortex-M cores

Trace Features

- 4-bit ETMv3 in Continuous mode
- ITM over TPIU and Serial Wire Output
- Multicore tracing
- Streaming trace information to the host computer for long-term tracing, streaming rate up to 100 MByte/s
- Analysis of task and function runtimes
- Code coverage analysis
- Trace evaluation even during recording
- Energy measurement using TRACE32 Analog Probe

As with all Lauterbach products, µTrace is controlled by the TRACE32 PowerView GUI.

ARM CONNECTED

I.P.

LET US KNOW

If you have changed your address or if you do not want to receive any mail from us, please send an e-mail to:
mailing@lauterbach.com