TRACE32®
IN-CIRCUIT EMULATORS & DEBUGGERS
ONE SYSTEM FITS ALL!

NEWS

2005

NEWS FOR THE EMBEDDED SYSTEMS CONFERENCE

Lauterbach Set to Further Expand

Another year has drawn to a close and again a new milestone has been established in the company´s history. In 2004, sales once more increased by 30%. This in mind, we have been able to plan the new year with great confidence, reporting new developments, right from the start: On January 1, 2005, Lauterbach opened an office in Milan, Italy (see page 12).

What are our plans for 2005?

Although we anticipate only moderate growth of the development tool market, in the near future, our goal is to continue expanding at the same rate in 2005. Market analysis reveals significant regional differences in the various areas of the world where we are represented. Thus, we will focus our efforts accordingly.

North America and Europe

In North America and Europe, where Lauterbach has been well established for many years, we predict stable yet moderate expansion. For these markets, maintaining our technology lead, will be the deciding factor for a successful 2005.

Asia

In the coming months, we expect Asia to be the region with the greatest growth potential for tool manufacturers. Essentially there are two reasons for our positive outlook: first of all, significant economic growth is, by and large, predicted in Asia, especially in China; secondly, for some time now, we have observed more and more Western companies, especially from Europe, are outsourcing software and hardware development projects to Asia, mainly to India and China.

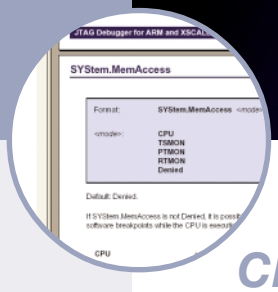
Of course, regional differences in Asia also have to be considered. For example, in Japan, a zero growth development tool market was already shared by local manufacturers. Nevertheless, Lauterbach was able to gain significant market share and, in its second year, our establishment in Yokohama has already generated a profit.

The situation in China is different. In this region we set the course for the years to come. We are confident that we can establish ourselves as a European company, by virtue of our leading-edge technology, high quality and expeditious support of new processors. In the future however, we expect more competition from local suppliers that will force prices down by using their low cost production capabilities. In lieu of Lauterbach's existing strong presence in Japan, Korea, and India, we plan to set up a branch in China in 2005.

*Now, back to the US. At the upcoming **Embedded Systems Conference** in San Francisco, you'll have the opportunity to see for yourself the innovative power and performance of the TRACE32 PowerTools. We will be delighted to welcome you to our exhibit. Plus, you can read more about these tools in the following pages.*

LAUTERBACH

Visit www.lauterbach.com for more information



CFI Flash Programming

From May 2005, Lauterbach products will support automatic generation of flash declarations using the Common Flash Interface (CFI).

Until now, it was necessary to have comprehensive knowledge of the exact device type and arrangement of a flash component to be able to program flash content. The CFI query mode automatically obtains all parameters required for flash programming from CFI-compatible flash components. Using this data, TRACE32 is able to automatically create a flash declaration. The user only has to set the following two parameters:

- *Start address of the flash component*
- *Data bus width over which the flash component is connected to the microcontroller*

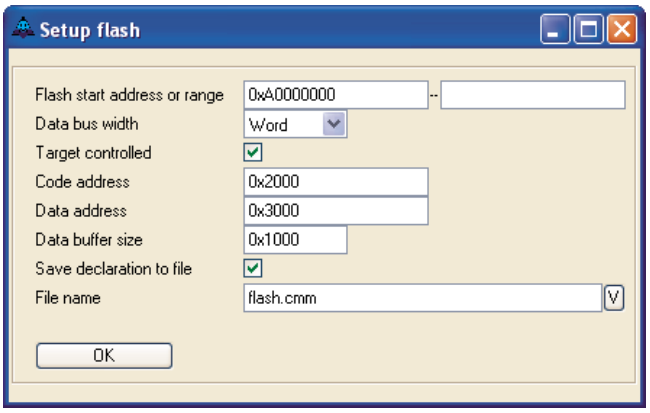
Naturally, flash declarations generated in this way can be converted into executable scripts.

TRACE32 will support CFI for the following flash programming methods:

***TRACE32 tool-based flash programming:** This method involves programming flash content using only the TRACE32 software. This obviates the need for extra resources on the target system. This is a useful process to use at the start of a project because no additional parameters need to be defined.*

***Target-controlled flash programming:** The advantage of this approach is that the algorithm used for flash programming runs on the target system. This means that considerably less communication with the TRACE32 software on the host is required. As a result, target-controlled flash programming is approximately 20 times faster than the tool-based flash programming method. For this method, RAM areas must be declared on the target system for:*

- *The flash algorithm*
- *Data exchange between TRACE32 software and the flash algorithm*



Every time an operation has to be performed on the flash component, TRACE32 loads the flash algorithm to target RAM for execution by the cpu.

The buffer size determines how much data will be programmed when the flash algorithm is executed once.

Lauterbach provides binary files of the flash algorithms for the majority of processor architectures and flash components on the TRACE32 software CD.

Real-time kernels now supported

- ***LINUX 2.6** for ARM, MIPS, PowerPC, SH4 and XScale*
- ***QNX 6.3** (QNX Software Systems) for ARM, PowerPC, SH4 and XScale*
- ***Quadros** (Quadros Inc.) for TMS320C55x*
- ***Symbian OS V8.0a** and **Symbian OS EKA2** (Symbian) for ARM*
- ***ThreadX** (Express Logic) for StarCore*
- ***Windows CE** (Microsoft) for SH4*
- ***Windows CE 4.x** (Microsoft) eXDI Driver for ARM, SH4 and XScale*
- ***Windows CE 5.0** (Microsoft) for ARM, SH4 and XScale*
- ***µIPLUS** (Accelerated Technology) for ARM and PowerPC*

Cache Analysis

In October 2004, Lauterbach introduced a cache analysis function for the TRACE32 PowerTrace microprocessor development tool.

A cache is a small, fast memory that is usually integrated directly in the microcontroller. To prevent the microcontroller from having to retrieve each instruction or data from the slow external memory, individual program sections are temporarily stored in the cache. However, the cache can only store a limited number of instructions or data required because of its small size (4 KB to 128 KB). As a result, different parts of the program compete for space in the cache and replace one another. The memory address and the cache architecture determine what data gets stored in the cache and what to evict. However, each time cache contents are evicted, they have to be reloaded to the cache when the program needs them again. This requires access to the external memory, which, if possible, should be avoided for the following reasons:

- *On the basis of current external memory access times, it can be assumed that it generally takes 10 to 30 times longer to retrieve an instruction or data from the external memory than to retrieve data already stored in cache. This means that every time external memory is accessed, program execution time greatly increases. Using memory benchmarking, TRACE32 can determine the actual time it takes to access individual cache memories and external memory on the target system (see Figure 1).*
- *Every time external memory is accessed, power consumption of the overall system increases considerably. This is a major disadvantage, especially for portable, battery-operated devices.*

Preventing unnecessary evictions from cache can reduce the frequency with which the external memory is accessed. Unnecessary eviction always occurs when

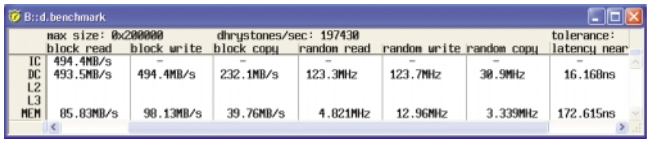


Figure 1: Using a benchmark test program, TRACE32 determines access rates for all cache memories and external memory.

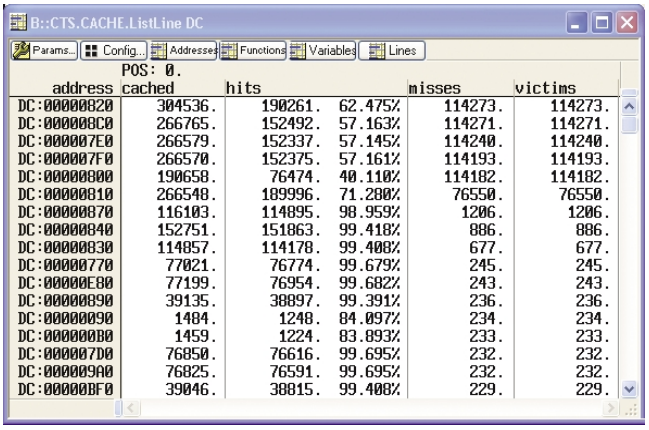


Figure 2: Many memory addresses compete for just a few cache lines, whereas other cache lines are hardly used.

several program instructions or data compete for just a few cache lines, whilst other cache lines are seldom used (see Figure 2).

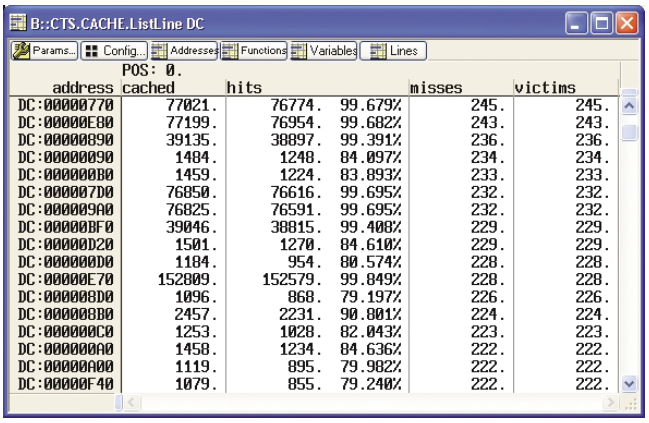


Figure 3: Competition for cache lines is evenly distributed.

Cache efficiency can be optimized by modifying the relocation information in the linker command file so that functions and data are placed in the memory in such a way that competition for cache lines is distributed as evenly as possible (see Figure 3).

The Lauterbach cache analysis method gives developers an analysis tool that helps them to trace unnecessary evictions and, as a result, optimize cache efficiency.

Implementation

Before cache analysis is described in more detail, here are some definitions of important terms:

***Cache hit:** An instruction or data required by the microcontroller is already located in the cache.*

***Cache miss:** An instruction or data required by the microcontroller that is not located in the cache, and therefore has to be loaded from external memory.*

***Cache victim:** An instruction or data that has been evicted from the cache in order to create space (after a cache miss) for the program part currently required.*

***Cache line:** The cache is organized in lines. A cache line contains 8, 16 or 32 bytes, which is a multiple of the bus width of the microcontroller. When a cache miss occurs, the new contents are loaded into the cache line in blocks (burst accesses to the external memory).*

The following steps are required to carry out a cache analysis for a selected system function (for example, image compression for a cellular phone) using TRACE32 PowerTrace:

1. Predefinition of the cache structure

The TRACE32 software identifies the cache structure of the microcontroller used by the target system.

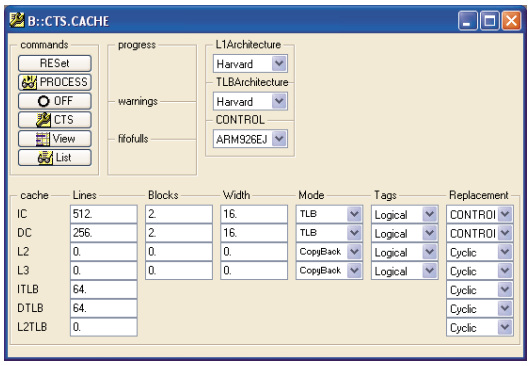


Figure 4: Automatic definition of the cache structure of the microcontroller

All currently available cache structures and cache hierarchies, including level-2 and level-3 caches, are supported. In particular, TRACE32 also supports cache structures configured using a memory protection unit (MPU) or a memory management unit (MMU) (see Figure 4).

2. Recording of program and data flow for this function in the trace memory

Cache analysis is based on the program and data flow recorded in the trace memory. TRACE32 PowerTrace provides a 128-MFrame trace memory, which makes it possible to record up to, for example, 10 seconds of program run-time for the ARM-ETM.

3. Performing the cache analysis

The cache analysis first determines the cache hit, cache miss, and cache victim rates for the recorded program and data flow, based on the defined cache structure (see Figure 5).

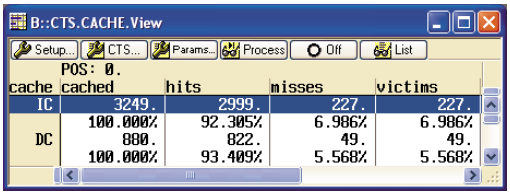


Figure 5: Analysis of the cache hit / cache miss / cache victim rates for the instruction cache (IC) and the data cache (DC)

To be able to use this information to reduce the number of cache victims, the following information is required:

- *Which cache lines have a particularly high cache victim rate?*
- *Which instructions are competing for these cache lines?*
- *Are any cache lines not being or hardly being used?*

The analysis shows (see Figure 7) that the instructions at the addresses 0x8150, 0xA150, and 0xC150 are competing for cache line 0x15. The above example uses a cache in which each cache line has two locations in the cache (2-way associative cache). Three program addresses are competing for one cache line; therefore, eviction will always occur, resulting in cache victims. At the same time, cache line 0x19 is completely unused. By

B::CTS.CACHE.ListLine IC

Params... Config... Addresses Functions Variables Lines

POS: 0.

address	cached	hits		misses	victims
IC:00000150	768.	602.	78.385%	164.	164.
IC:00000160	896.	831.	92.745%	63.	63.
IC:00000170	320.	318.	99.375%	0.	0.
IC:00000180	32.	31.	96.875%	0.	0.
IC:00000190	0.	0.		0.	0.

Figure 6: Analysis of the cache hit / cache miss / cache victim rates for the individual lines of the instruction cache

B::CTS.CACHE.ListRequests IC 0x150

Params... Config... Addresses Functions Variables Lines

POS: 0.

address	cached	hits		misses	victims
IC:00008150	384.	352.	91.666%	31.	31.
IC:0000A150	128.	58.	45.312%	69.	69.
IC:0000C150	256.	192.	75.000%	64.	64.

Figure 7: List of the program addresses competing for cache line 0x15 of the instruction cache

redirecting the address of one of the competing program instructions to the address 0x190, the competition can be removed from cache line 0x15 and there will be no further cache victims.

Relationship Between Cache Efficiency and Program Run-time

As described in the previous section, cache analysis is based on the program and data flow recorded in the trace memory. As all entries in the trace memory have a time stamp, the trace contents are used as the basis for run-time measurements. This has the advantage that the direct relationship between program run-time and cache efficiency can be measured and verified using TRACE32 PowerTrace.

Figure 6 shows that the analyzed function has a cache hit rate of 78% and a cache victim rate of 22% for the cache line 0x15. The high eviction rate is, of course, also immediately apparent in the run-time of the function. The average run-time for the function (sievebad) is 77.6 us (see Figure 8).

By redirecting the address of one of the competing instructions to the address 0x190, the competition is removed from cache line 0x15. Only two addresses are now competing for this cache line; therefore, there is no further evictions. An analysis of the optimized function (sievegood) shows an immediate and considerable improvement in run-time performance. The run-time is now only 7.9 us on average (see Figure 8).

Using a combination of cache analysis and run-time measurements, developers can use TRACE32 PowerTrace to test whether optimized cache efficiency actually leads to the predicted improvement in run-times. TRACE32 PowerTrace can also be used to investigate if code optimizations impair cache efficiency and therefore do not improve program run-times.

Summary

The latest cache analysis tool from Lauterbach provides developers of embedded systems with the following benefits:

- *Optimization of program run-times*
- *Reduction in the power consumption of the entire system*

Cache analysis can be performed on all microcontrollers, cache architectures, and cache hierarchies. If required, the predefined cache structure of microcontrollers can also be modified. In this way, an alternative cache structure can be tested to determine if its use will lead to greater cache efficiency and therefore shorten program run-times. The result can then be taken into account for future designs.

Cache analysis was developed in close cooperation with leading mobile phone manufacturers. It is yet another demonstration of Lauterbach's leading role in the development tool market.

B::Trace.STATistic.DURation sievebad

Setup... Chart Zoom Zoom Move Move Init

samples: 10. avr: 77.605us min: 77.445us max: 79.045us

up to	count	ratio	1%	2%	5%	10%	20%	50%	100
77.440us	0.	0.000%							
78.080us	9.	90.000%							
78.720us	0.	0.000%							
79.360us	1.	10.000%							
80.000us	0.	0.000%							

B::Trace.STATistic.DURation sievegood

Setup... Chart Zoom Zoom Move Move Init

samples: 9. avr: 7.904us min: 7.005us max: 14.460us

up to	count	ratio	1%	2%	5%	10%	20%	50%	100
7.040us	0.	0.000%							
9.600us	8.	88.888%							
12.160us	0.	0.000%							
14.720us	1.	11.111%							
17.280us	0.	0.000%							

Figure 8: Run-time analysis of the functions sievebad and sievegood using TRACE32 PowerTrace

Multiple Licenses in Debug Cables

Since early 2004, a TRACE32 debug cable can include several licenses. This article gives a short overview of the possible applications of multiple licenses and how to use them.

A license matrix (4 columns and 3 rows) has been placed in the debug cable to enable administration of multiple licenses. Multiple licenses were introduced for the following purposes:

New generations of processors

For example, a customer using a TRACE32 debug cable for ARM9 in a project can upgrade this debug cable with an eXtension license for ARM11 debugging.

ARM7	ARM9	ARM11	Cortex

Example of multiple licenses within the ARM architecture

This type of upgrading within the same processor architecture is possible if the new generation of processors physically uses the same debug interface.

Complex on-chip trace

Some processor architectures have large on-chip trace memories, which also usually provide complex filter and trigger capabilities. These on-chip traces are generally configured and read out via the debug interface. An eXtension license is required to make trace configuration and trace evaluation available when using TRACE32 software.

ARM11	ETB Trace Support		

An eXtension license is required for the configuration of an Embedded Trace Buffer (ETB) and the analysis of the trace contents.

In addition to the ARM Embedded Trace Buffer (ETB), the following complex on-chip traces require an eXtension license:

- *EJTAG on-chip trace for the MIPS architecture*
- *TriCore TC1796ED on-chip trace from Infineon*

Multicore debugging

From spring 2005, multicore designs can be tested and integrated using a single debugger hardware. One debug cable handles the common on-chip debug interface of all cores. For more information, see page 7.

- ***Multicore designs with identical cores***

  Multicore designs that are based on identical cores require a license for the processor architecture plus a multicore license. For example, to debug a Motorola MSC8102, which contains four StarCore DSPs, you need a debug cable that includes a StarCore and a multicore license.

StarCore	Multicore License		

Example of multiple licenses for a multicore design with identical cores. This example is for a design with four StarCore DSPs.

- ***Multicore designs with different cores***

  Multicore designs, in which a RISC processor is responsible for system control and a signal processor handles the data processing, require a debug cable that includes licenses for both architectures. The DSP license is referred to as an Additional license.

ARM9			
Teak/ TeakLite for JAM Interface			

Example of multiple licenses for a multicore design with different cores. This example is for a design with ARM9 and TeakLite DSP.

***In short**: Multicore debugging is only possible if the debug cable includes more than one license.*

New Concepts for Multicore Debugging

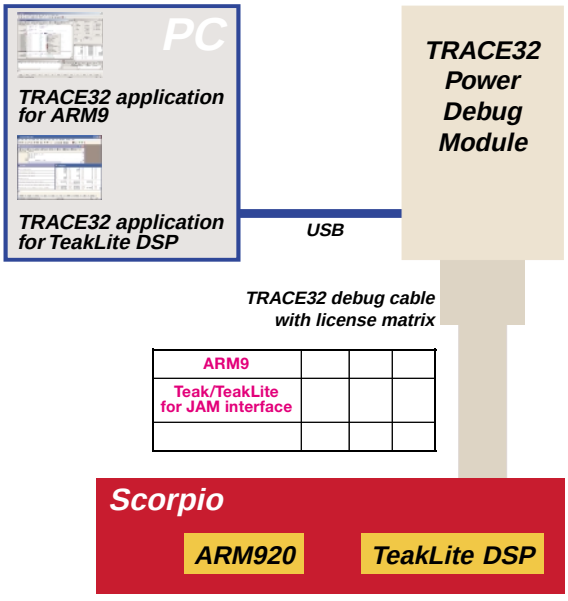
For the past three years, the Lauterbach TRACE32 ICD In-Circuit Debugger has supported debugging of multicore SoCs. Lauterbach is now expanding its multicore debugging concept based on experience gained from many customer projects.

To achieve optimum functionality and performance, several cores are increasingly being integrated to create a system-on-chip (SoC). Use of RISC processors merged with one or several DSPs is widespread; however, other combinations are also in use. In multicore designs, generally only one debug interface is provided for all cores in order to save on pins and costs.

Control of several cores over a common debug interface

This new concept means that only one PowerDebug Module and one debug cable are required to debug all cores in an SoC. For this to work, the debug cable must include either individual licenses for all cores used or a multicore license (see also page 6).

For each core that is debugged, a separate TRACE32 application runs on the host. The common TRACE32 system software on the PowerDebug Module ensures that debug commands issued by an application are forwarded to the correct core.



Debugging ARM920 and TeakLite DSP in Scorpio using TRACE32 ICD

Here are two examples that illustrate this concept:

***Scorpio**: Scorpio from Samsung is an SoC that contains an ARM920 and a TeakLite DSP. To debug both cores at the same time, the debug cable must include a license for ARM9 and a license for TeakLite (see diagram).*

***MSC8102**: MSC8102 is an SoC from freescale that contains four StarCore DSPs. To debug all four DSPs at the same time, it is sufficient if the debug cable includes a StarCore license and a multicore license.*

Start / Stop synchronization

Start/stop synchronization naturally plays an important role in simultaneous debugging several cores. Both features only work if the SoC supports both synchronized starting and stopping. For example, if the individual cores are connected over a breakswitch, all cores can be stopped instruction-accurate. Furthermore, a programmable breakswitch allows users to configure which cores in the SoC should stop each other synchronously. If the SoC does not support synchronized stopping, and there is no actual physical way in which the cores can stop each other in the target system, then the debugger can only implement a loosly coupled synchronized stop. The same applies to start synchronizations: If the individual cores in the SoC can be started at the same time using their debugging logic (for example, using a special JTAG command), then the debugger can implement a synchronized start. Otherwise, the debugger has to start the cores individually in quick succession.

Shared trace port

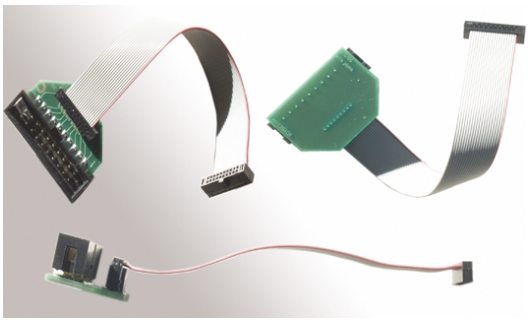
Many developers who use a multicore SoC design do not want to forgo the advantages of a real-time trace. Naturally, the number of pins for the trace port has also been reduced to save costs. In the meantime, the first SoC designs are available in which several cores share the same trace port. In current designs, developers can specify which core can use the trace port exclusively. There are also SoCs in which several cores use one trace port at the same time (for example, in the CoreSight concept from ARM or for NEXUS).

New for TRACE32 ICD Debuggers

Half-size adapters

Since the beginning of 2004, Lauterbach has provided half-size adapters.

The plugs on TRACE32 debug cables generally require standard dual-row connectors (pins with 100-mil spacing, pin rows with 100-mil spacing) on the target system. The new half-size adapters allow to use connectors with 50-mil spacing. This reduces the space required on the target system for debugger connections by about a quarter.



Half-size adapters for debug cables

Half-size adapters are available for all 14, 16, and 20-pin debug cables. The connecting cable is 3.9 inches in length.

http://www.lauterbach.com/adhalfsize.html

Optimization of download times

Download times have been considerably improved for the following processor families:

- *PowerQUICC III can achieve a download rate of 4.3 MB/s with a 50-MHz JTAG clock.*
- *MPC55xx can achieve a download rate of 2.2 MB/s with a 120-MHz core clock and a 40-MHz JTAG clock.*
- *ARM11 can achieve a download rate of 3MB/s with a 50-MHz JTAG clock.*

Faster download times were made possible by optimizing the software and enhancing the debug cable hardware.

Altera	**NIOS II**	now
AMCC	**PPC440EP**	now
ARM	**Cortex-M** **Cortex-A** **Cortex-R**	Q2/05 Q3/05 Q3/05
CEVA	**CEVA-X 1620**	Q1/05
freescale	**S12X** **MPC521x** **MPC7448** **MPC83xx** **MPC8541/MPC8555** **MPC8548** **MSC711x (StarCore)**	now Q1/05 Q1/05 Q1/05 now Q2/05 now
IBM	**PPC970FX** **STB02500**	Q2/05 Q1/05
Intel	**IXP23xx (XScale)**	Q1/05
NEC	**V4133** **VR55xx** **V850 (N-Wire)**	Q1/05 Q1/05 now
Renesas	**M32R**	Q1/05
ST	**Nomadic (ARM, MMDSP)** **NEXUS-MMDSP**	Q1/05 Q1/05
Texas Instruments	**TMS320C2x** **TMS320C54x** **TMS320C6x**	Q2/05 Q1/05 now
XILINX	**Virtex-II Pro (PPC405) in single and multi-core design**	now
ZSP	**ZSP500**	Q2/05

New for ARM Architecture

New JTAG clock modes for ARM processors

Lauterbach supports several new JTAG clock modes for its ARM debugger. The new JTAG clock modes allow the debug interface to clock at considerably higher frequencies.

Higher JTAG frequencies allow, for example:

- *Faster download of code/data to the target memory.*
- *Faster upload of data from the on-chip trace memory (ETB) on the host system.*

All ARM designs, whereby the JTAG clock can run independently of the CPU clock, could previously only achieve a JTAG frequency of maximum 25 MHz. The following new JTAG clock modes are now available:

***CTCK**: This new JTAG clock mode can be used with the new debug cable that has been available since the start of 2004. It increases the frequency of the JTAG clock by up to 35 MHz.*

***CRTCK**: If TCK on the target is connected*

- *directly to RTCK*
- *via a driver to RTCK*

then the CRTCK JTAG clock mode can be used. This mode enables a JTAG clock frequency of up to 50 MHz.

Synthetic ARM cores (ARMxxx-S) have to work with RTCK as JTAG clock. RTCK must be used when synchronization between the JTAG clock and the CPU clock is required. With the RTCK JTAG clock mode, the JTAG clock works, at maximum, at a sixth of the frequency of the CPU clock. In this case, the maximum possible frequency is 20 MHz.

***ARTCK**: The new ARTCK JTAG clock mode uses a special acceleration mode and therefore provides a JTAG clock that is, at maximum, half as fast as the CPU clock. This can reach approximately 50 MHz.*

Support for NEON technology

ARM NEON technology provides a comprehensive instruction set for embedded designs, intended mainly for multimedia and signal processing. As soon as the first ARM processor with NEON technology is available, Lauterbach will be able to provide the appropriate tools.

Support for CoreSight

CoreSight is the new ARM standard for debugging and tracing multicore SoCs. CoreSight contains many new concepts, such as a single-wire debug interface, monitoring of variables during execution of the program, simultaneous tracing of different cores, processor buses, and peripheral components. Lauterbach will naturally support all these new components with its PowerTools products.

New Debugger for V850E/V850ES Cores

Lauterbach's TRACE32 ICD and PowerTrace development tools now support all NEC processors with V850E and V850ES cores.

The TRACE32 ICD debugger provides user-friendly debugging in assembler and high-level languages over the N-Wire interface of the V850 for all standard compilers. Naturally, all on-chip breakpoints and triggers are also supported.

TRACE32 PowerTrace can record the program and data flow of all processors with extended N-Wire interfaces in a 512-MB trace memory. This run-time information supports systematic troubleshooting and comprehensive performance analysis.

Furthermore, certain models from the V850 family have an NBD interface. Memories can be read and written over this interface while the program is running in real-time. Lauterbach provides a small additional hardware to implement this functionality (NBD Box for Real-Time-Memory-Access V850).

http://www.lauterbach.com/bdmv850.html

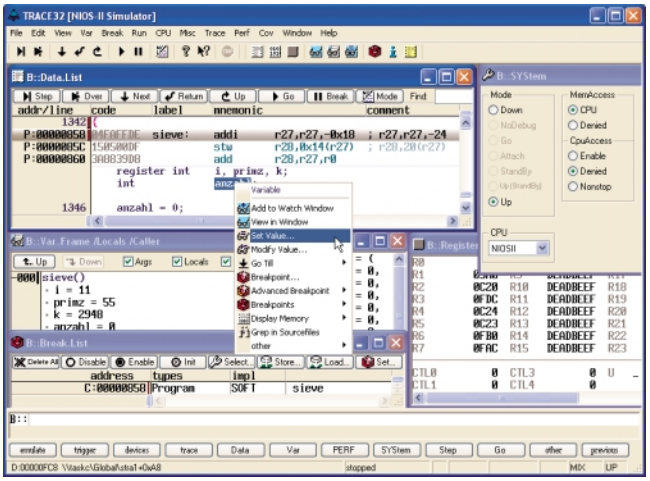
Debugger for NIOS II

Since the beginning of 2005, Lauterbach has introduced a debugger for the NIOS II soft core from Altera.

NIOS II is a 16/32-bit soft core. Together with application-specific peripheral components, NIOS II can be used to easily design an SOPC (System On a Programmable Chip). At the same time, on-chip debugging and trace logic can also be integrated into the soft core.

The on-chip debugging supports straightforward debugging at assembler and high-language levels for the GNUPro C/C++ compiler. As the on-chip debugging interface uses the same 10-pin connector as the Altera Byteblaster, Lauterbach also supports reprogramming of the SOPC using TRACE32 ICD.

The integration of the trace logic into the SOPC makes program and data flow visible, using a 19-pin trace port. After this information has been recorded in the 512-Mbyte trace memory of TRACE32 PowerTrace, systematic trouble-shooting, extensive run-time measurements, and code coverage analysis can be performed.



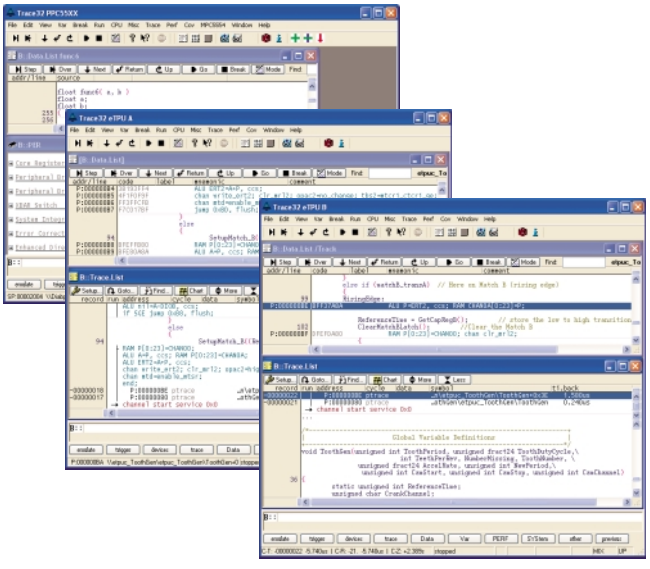
An SOPC can contain several NIOS II soft cores; Lauterbach tools have been designed from the beginning, to support multicore debugging for NIOS II.

eTPU Debugger for MPC55xx

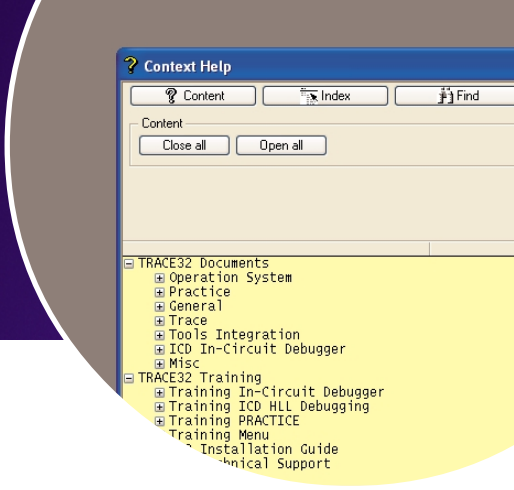
Since October 2004, Lauterbach development tools have supported debugging and tracing of both eTPUs in the MPC55xx family from freescale.

The basic idea behind testing eTPUs is to start an individual application for both the core and each eTPU. This way, each application can load and debug its own program and also display its own NEXUS messages. The core, the DMA controller and both eTPUs can generate NEXUS messages that are broadcasted via the same trace port.

For the MPC55xx Lauterbach has developed for the first time a multi-source trace. The main task of the TRACE32 system software is to assign trace packages to the individual applications and to ensure that the developer can clearly see the time relationship between core and eTPU trace data.



For the debugging and tracing of core/DMA, eTPUA and eTPUB individual applications are started



Lauterbach APIs

With its APIs, Lauterbach provides software interfaces that enable:

- *A TRACE32 development tool to be controlled by an external application*
- *A TRACE32 development tool's set of functions to be extended by means of an external application*

Here is an overview of the individual TRACE32 APIs:

TRACE32 API

The TRACE32 API is the standard API used to control a TRACE32 development tool using an external application. The following function-specific APIs are part of TRACE32 API:

- ***Visual Basic API** provides a precompiled DLL, which enables a TRACE32 development tool to be controlled using a Visual Basic program.*
- ***FDX API** provides C libraries, enabling an external application that runs on the host to rapidly exchange data with the application on the target system. The TRACE32 development tool serves as a communication interface for the FDX communication.*
- ***JTAG API** provides C libraries that enable an external application to communicate with a JTAG TAP controller in the target system using TRACE32 ICD debugger hardware. This way, users can implement, for example, boundary scan test programs or basic functions of a debugger for another core in the JTAG chain of the microcontroller without their own hardware interface.*

The following APIs have independent software interfaces:

Simulator API

The Simulator API enables users to define timers, external communication interfaces, and other peripheral components for Lauterbach instruction set simulators. This means that an instruction set simulator can also be used to test programs that operate timers, or exchange data over communication interfaces.

Communication API

For a number of microcontrollers, Lauterbach offers ROM-Monitor-based debuggers. These debuggers communicate with the ROM-Monitor installed on the target system over an RS232 interface or over the interface provided by the Lauterbach EPROM-Simulator hardware.

The Communication API now enables a ROM-Monitor-based debugger to be extended using an external application in such a way that it can also be operated using other communication interfaces such as Ethernet or CAN.

Kernel Awareness API

TRACE32 development tools typically support most real-time operating systems currently available on the market. The main functions supported are:

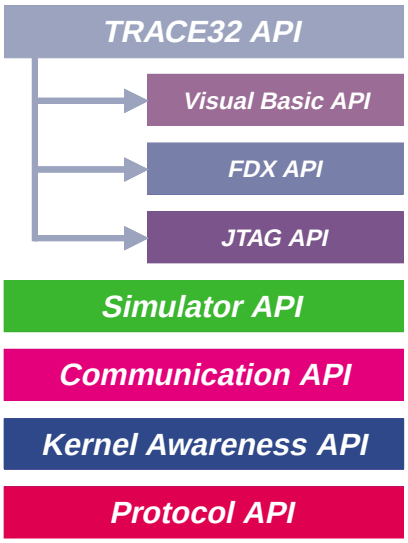
- *Display of operating system resources during debugging*
- *Comprehensive methods of analyzing the run-time behavior of tasks*

The Kernel Awareness API enables users of TRACE32 development tools to customize this operating system support to their proprietary real-time operating systems.

Protocol API

The TRACE32 PowerProbe and PowerIntegrator products allow to record the time behavior of any communication interface as well as any bus protocol. TRACE32 software already supports the evaluation and the display of important communication and bus protocols.

The Protocol API enables raw trace data to be converted into higher-level protocols using an external application, even if TRACE32 software provides no customization of the communication or bus protocol. This means that run-time behavior recorded in the trace can be analyzed quickly and intuitively.



TRACE32-FIRE
the superfast fully integrated emulator

Visit www.lauterbach.com for more information

In-Circuit Emulator for the S12X Family

Coinciding with the release of the BDM debugger for the freescale S12X family in August 2004, development of the in-circuit emulator TRACE32 FIRE was also completed by the end of the year.

The in-circuit emulator provides the following important advantages over the BDM debugger:

***Emulation memory**: The on-chip flash is replaced by emulation RAM. This allows you to load the program you want to test more quickly and, at the same time, provides an unlimited number of software breakpoints for debugging.*

***Additional read/write breakpoints**: The in-circuit emulator can use its own resources for setting read/write breakpoints. This means that program execution can be stopped on reading or writing memory/variables. Two read/write breakpoints can also be linked to a data value.*

***Code coverage/variables analysis**: By making use of its own resources, the in-circuit emulator provides a code coverage analysis as well as a comprehensive analysis of read/write accesses to memory and variables.*

***512-K frame trace memory**: When using the in-circuit emulator, the 64 entries of the on-chip trace memory are no longer required for the S12X core. The on-chip trace functions are available solely for working on the XGATE co-processor.*

***Port analyzer**: Lauterbach provides port analyzers for all of its emulators. A port analyzer records the signals from all ports without any additional connections and displays a direct time relationship with program execution.*

In addition, the FIRE emulator for the S12X family is now easier to adapt to the target system (see picture).

If required, the S12X core module can be separated from the FIRE basic system using Mictor flex extensions. This makes it easier to adapt the system to the target hardware.

New Branch in Italy

Maurizio Menegotto, director of Lauterbach Srl

In time for the beginning of 2005, Lauterbach opened a new branch in Italy. Lauterbach Srl, located in Milan, strengthens Lauterbach's position in the Italian development tool market. The director of Lauterbach Srl, Maurizio Menegotto, has many years experience in sales and technical support of Lauterbach products.

Please inform us:

If you would like us to remove your name from our mailing list, send an e-mail to:
info_us@lauterbach.com

LAUTERBACH