TRACE32®
IN-CIRCUIT EMULATORS & DEBUGGERS
ONE SYSTEM FITS ALL!
NEWS
2004

Visit www.lauterbach.com for more information

NEWS FOR THE EMBEDDED SYSTEMS CONFERENCE

In 25 years from a one-man operation to global market leader - a German success story

The early years

Back in 1979 Lothar Lauterbach founded his one-man operation Lauterbach Datentechnik. Initially he earned his living as a contractor. His work required a debugger and since there was no good, reasonably priced product on the market he decided to develop his own. The outcome was the first in-house software product, that could run on CPM systems, the ***TRACE80 debugger*** *for the Z80. In 1982 his younger brother Stephan Lauterbach joined the company. In the beginning as a computer science major doing his internship, but soon as full-time employee doing his studies on the side. The fruits of their work was the first real* ***TRACE80 in-circuit emulator*** *developed for the Z80. In those days the leading edge CAD tool of their choice was adhesive crepe tape, which they used for PCB layout on a 10:1 scale. In 1984 the first employees were recruited and the company moved into two sublet rooms at MBB in Ottobrunn near Munich. Now numbering a total of 5 employees they set about the further development, production and marketing of their tools. Adhesive tape was replaced by the company's first real CAD system, allowing more efficient PCB development. The existing emulators were modified for use with the 8085, the NSC800 and also the HD64180. Lauterbach also offered suitable host computers in the form of the Commander desktop computer and portable SCOUT model which was more or less a predecessor of the first laptops. Both computers were based on the Z80 microprocessor and the CPM operating system the mainstream OS at the time. For the first time these products were presented to a wide audience at the 1985 Hanover Fair. The high-level language debugger for C and PLM was applauded by the fairgoers as a real sensation. TRACE80 quickly became the company's cash cow generating annual revenues of about $ 3 million in the years 1985/86.*

First product advertisment published in Germany 1985

The next challenge

The foundation was laid for the next major milestone, the development of a universal in-circuit emulator. This emulator was based on a modular design, which allowed support of a large variety of 8 to 32-bit microcontrollers and at the same time offering ground-breaking new technical features. In consequence it lead to the development of the ***TRACE32-ICE*** *for Motorola's legendary MC68000. In 1986 the new challenges and opportunities required further growth of the company. Additional employees were hired to strengthen the development team and a semidetached house was rented in Neubiberg near Munich to provide extra office space. Stephan Lauterbach developed his first flexible Window interface under CPM and MS-DOS in the "children's room" while Lothar sat at his PC in the "bedroom" designing a universal hardware concept.*

It followed the most challenging, but also the most exciting time in the company's history. The development of the new emulator dragged on longer than

LAUTERBACH

Visit www.lauterbach.com for more information

planned. At the same time Lauterbach was now entering a market segment where it had to compete against well established brand-name competitors.

TRACE32-ICE becomes established

Stephan Lauterbach

In autumn 1987 the company moved into its own building at the present location in Hofolding. In order to finance this expansion the company was effectively under pressure to succeed. The leanest years in the company's history followed. Nevertheless, it succeeded in winning customers one by one against the "big" players under these difficult conditions. Lauterbach had only one sales argument: **better technology.** Buying motives such as "long proven", "customer success stories" or "well established producer" were the monopoly of competition. All these attributes now apply to Lauterbach as well, but technology and quality are still the company's most important objectives.

After this difficult period TRACE32-ICE at last became an established best seller at the beginning of the 1990s. First-class technology, quality and the prompt roll-out of new emulator modules keeping up with new microcontroller derivatives finally won customers' confidence.

Lothar Lauterbach

On-chip debugging with TRACE32-ICD and PowerTools

In 1994 the company's revenues came exclusively from sales of classic in-circuit emulators. That year Lothar Lauterbach set the course for the company's continued success in the future by initiating the development of the first BDM debugger. The **TRACE32-ICD** in-circuit debugger was born targeting Motorola's 68332. The company soon followed up with debuggers for many other microcontrollers and has achieved an excellent track record in terms of units sold. In 2001 the total number of TRACE32-ICD debuggers shipped passed the 10,000 mark. In 2004 Lauterbach expects to increase the total number of systems shipped to be beyond 25,000. Sales growth over the past few years had averaged over 25% per annum.

Global presence

Becoming a global player has become an important part of the company's development. At the beginning of the 1990s Lauterbach teamed up with sales partners in Europe and the U.S.A. In the mid-1990s sales activities were expanded to cover the Panasian market as well. In 1995 the first own sales office was established in the U.S., followed by sales offices in England (2001) and Japan (2002). Together with its worldwide sales partners Lauterbach is thus ideally positioned to deal with the challenges of the years to come.

All Lauterbach products at a glance:
Emulators - TRACE80 (1984), TRACE32-ICE (1991), TRACE32-FIRE (1997),
Debuggers - TRACE32-ICD (1995), PowerTrace (2001), PowerIntegrator (2003)

25-year anniversary

First things first: to all our existing customers a big "thank you" for your support over the last 25 years. To those new to Lauterbach products a warm welcome and we hope our technology can help bring your projects to a swift and successful conclusion.

Debuggers focus on DSPs

In 2003 the main emphasis with new processor architectures supported by the TRACE32-ICD in-circuit debugger was on digital signal processors.

LSI Logic	**ZSP**	now
Motorola	**DSP56800E**	now
Texas Instruments	**TMS320C55x** **TMS320C55x in OMAP**	now now
CEVA	**all licensed cores with JAM JTAG Interface** **TeakLite** **Teak** **XpertTeak**	 now Q1/04 Q1/04
StarCore LLC **Motorola**	**all licensed cores** **MSC8101** **MSC8102**	 Q1/04 Q2/04

Table 1: DSP processor architectures now supported

Data-intensive applications such as image transmission or the Internet browser in mobile phones require increasingly powerful processor architectures. The large volumes of data occurring in these applications are generally processed quickly and efficiently by digital signal processors. Two trends are currently observable for embedded designs in the communications industry:

- *The use of multicore SoCs whereby a RISC processor controls the system and a number of signal processors are responsible for the data processing.*
- *Integration of several signal processors on one SoC for efficient data processing.*

At the beginning of 2003, for the first time ever, Lauterbach made debugging of a RISC and a DSP possible with joint debugger hardware by supporting the OMAP from Texas Instruments. The OMAP contains both a TMS320C55x DSP and an ARM925/926 core. Both cores can be started and stopped synchronously.

In response to growing demand Lauterbach has added a number of other DSPs to its product range. DSPs are supported both as self-contained microcontrollers and as licensed cores which are usually integrated into a multicore SoC. See also Table 1.

Work is currently in progress on the development of a debugger for Motorola's MSC81xx architecture. This architecture is based on a licensed DSP produced by StarCore LLC. Initially the MSC8101 will be supported. Support for the MSC8102 will soon follow and for the first time Lauterbach will provide a debugging environment in which a processor containing four DSP cores can be tested.

New JTAG debuggers

ARM	**ARM10 Core** **ARM11 Core**	now now
IBM	**PPC750FX, PPC750GX**	now
Philips	**LPC21xx** **(ARM7TDMI based)**	now
Motorola	**MCS08** **MPC5200, MPC55xx** **MPC74xx** **MPC85xx**	now now now now

Table 2: Standard processor architectures now supported

In 2003 Lauterbach also increased the number of standard processor architectures supported by the TRACE32-ICD in-circuit debugger.

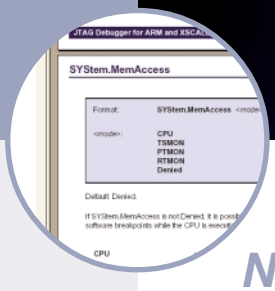
The ETM trace and the support for the onchip Embedded Trace Buffer is also already available for the ARM11. Support packages are already offered for the PowerIntegrator for tracing the program and data flow of the PowerQUICC III MPC8560.

New NEXUS debuggers

Motorola	**MAC71xx** **MPC5500**	now now

Table 3: NEXUS debuggers now supported

Lauterbach has now added the MAC71xx/MPC5500 processors to the list of Motorola processors with NEXUS technology that it supports. This further extends Lauterbach's market-leading position in this sector.



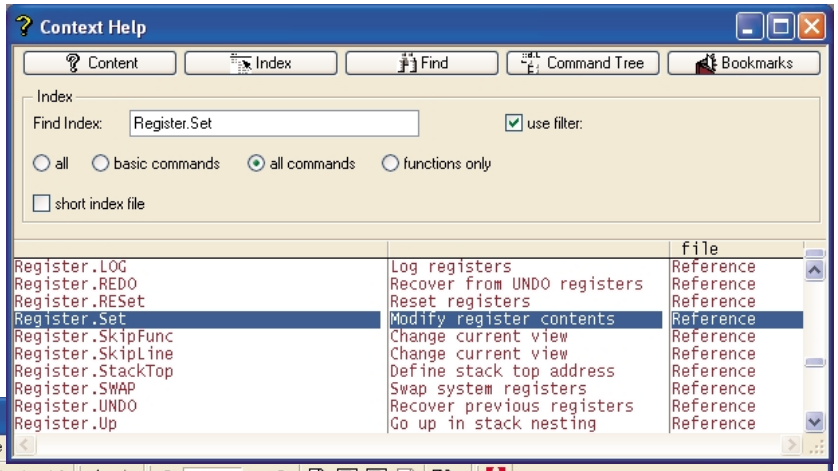
New Online Help

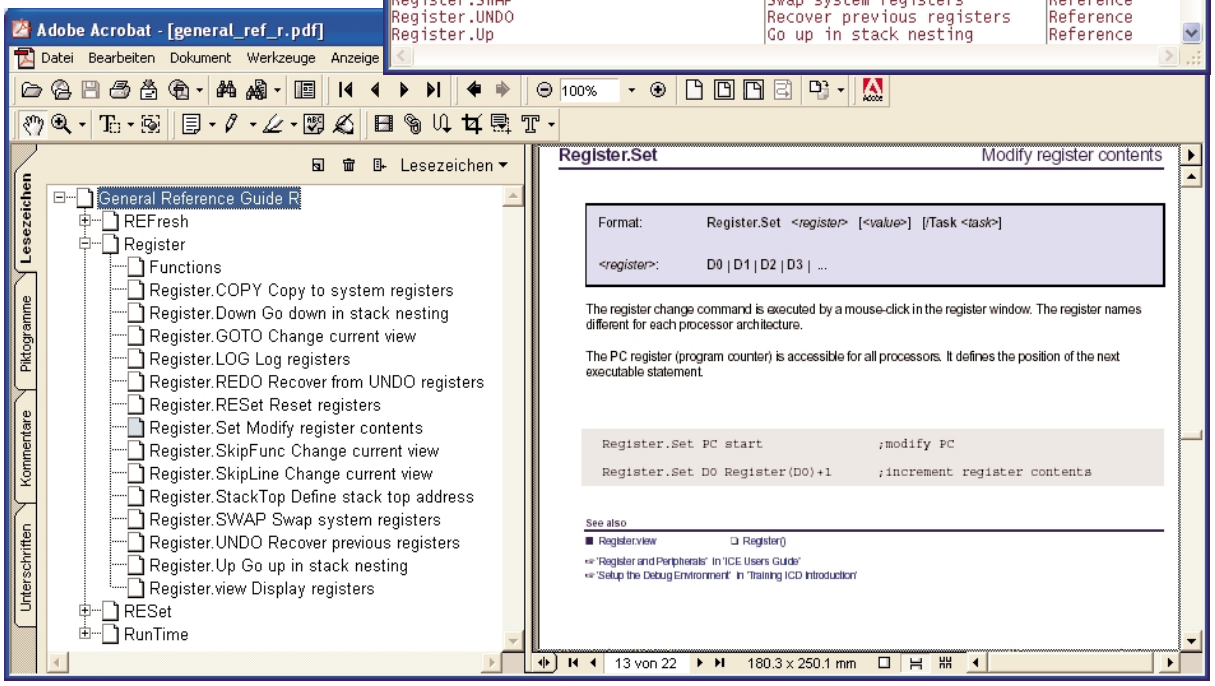
At the Embedded Systems Conference 2004 in San Francisco Lauterbach is presenting its new PDF-based online Help function. The main new features are as follows:

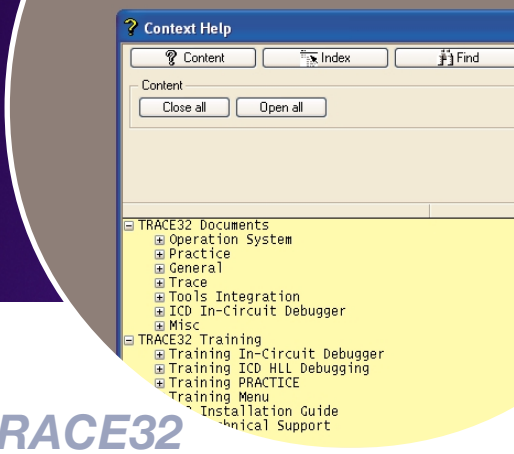
- *The online Help is matched to the specific development tool in use, to the processor architecture of the target system and also the RTOS employed. The entire online Help can also be used unfiltered if required.*
- *A complete Tooltip text system provides important information about the input windows and commands easily and clearly.*
- *The complete training documents are integrated into the online Help so that TRACE32 users can familiarize themselves with specific individual system functions by means of examples.*
- *The individual parts of the online Help are interlinked via cross references. TRACE32 users can quickly obtain an overview of similar or supplementary commands in this way.*
- *After an automatic update check the latest version of the Help files can be downloaded via the Internet.*

The new PDF-based online Help offers the same "look and feel" for Windows, LINUX and all the many workstation types we support. Further information is available on the Internet at:

www.lauterbach.com/help.html







Debugging of "User level" processes with TRACE32

Application debugging under Linux, QNX and WindowsCE

Testing user processes with a hardware debugger presents a real challenge for the debugger as well as for the developer. If an application is started, for example in Linux, a process is first generated; the corresponding application is then loaded from the operating system and linked, and finally started automatically. Since the debugger (and the developer) do not know the load address for the process before the application is started, no symbols can be loaded in advance. But without the symbol information it is not possible to set breakpoints. The application therefore starts up without any checks, possibly even running into the first bug. This problem is compounded by the fact that the operating systems also reprogram the MMU when generating and starting the process. This reprogramming of the MMU has to be correctly followed by the debugger in order for it to be able to follow and debug the new process correctly.

Because of these adverse factors in order to be able to debug a process, until recently, it was necessary, to enter a complicated sequence of commands either manually or under the control of a script.

1.) First of all a breakpoint was required in the kernel of the operating system to ensure that the program stopped whenever a process was generated.

2.) After the program run had stopped at the breakpoint it was then necessary to check whether the process just generated was the one that was to be debugged. If not, the program could be restarted without any further actions.

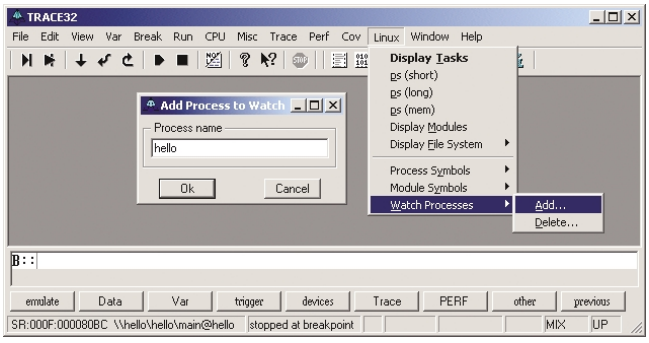


Figure 1: Activating the Process Watch System

3.) When the breakpoint located the required process the debugger would then be able to ascertain the destination to which the process had been loaded. This then made it possible to load the matching symbol information into the debugger and to set a breakpoint to main().

4.) At this point a "Go" would start the process which would then stop at main(). After this it was still necessary to scan the MMUs for the process. Debugging of the process could then begin.

Although this method works it is extremely awkward to implement. To simplify the debugging of processes Lauterbach has now implemented a **Process Watch System** *for a number of different operating systems and processors. This system recognizes whether a process that is required for debugging is present in the system, whether it has just started or has already finished. It automatically loads and deletes the relevant symbol information and the MMU tables. The user need only specify the name of the wanted processes and does not need to carry out any further actions.*

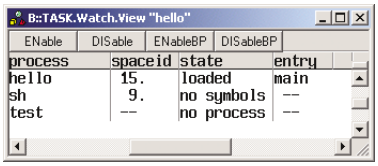
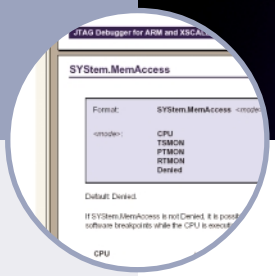


Figure 2: The Process Watch System monitoring 3 processes

Although the Process Watch System can monitor up to ten processes it is recommended that the number should be restricted to a far lower figure to avoid having a detrimental effect on the reaction speed of the debugger. As Figure 1 shows, the Process Watch System can be activated in order to observe a process simply by selecting from the menu (Linux in the example). It is then possible to add further processes by using the same selection procedure. Figure 2 depicts an activated Process Watch System with three processes.

The **Process Watch System** *operates as follows: when the debugger is stopped, it searches the target for the selected processes. If a matching process is found, the debugger automatically loads the symbol information and the corresponding MMU entries. If the process is not currently in the target, the symbols and MMU tables that had already been loaded may possibly be cleared from*



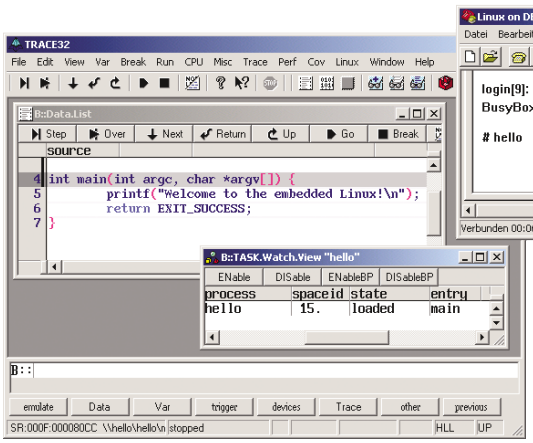


Figure 3: Start of a process

the debugger. If the loading address for the process has changed (possibly because it was terminated or restarted), the debugger recognizes this and initiates reloading of the symbols and scanning of the MMUs in order to operate according to the new conditions.

If necessary the debugger can detect the generation of a new process and stop the process at its entry point at main(). This makes it possible to debug the process from the beginning. Since this ***Process Watch System*** *uses a breakpoint for recognizing the generation of a process, this can be deactivated to avoid influencing the real time response of the target system. When the automatic stop is disabled during process generation the new symbol information will still be loaded after the next manual break.*

Figure 3 illustrates the situation after the process has been started in the console. The debugger has stopped the process at main(). At the next stop after termination of the process the symbols are then removed again from the debugger (Figure 4). If the process is restarted the debugger stops again at main(), and the developer can process a new debug cycle.
The Process Watch System is already available on various processors for Linux, QNX and Windows-CE.

Real-time cores now supported

- *DSP/BIOS for TMS320C55x and OMAP*
- *Jaluna C5 for PowerPC*
- *OS21 for ST40*
- *OSEck for TMS320C55x and OMAP*
- *RTXC Quadros for ColdFire, also planned for ARM and StarCore*
- *ThreadX now also planned for StarCore*
- *Windows CE.net for ARM and XScale, also planned for SH4 and MIPS*

3rd party tool integration

- *Windows CE Platform Builder eXDI, planned*

Monitor solutions

- *OSE SDM run mode debugger for PowerPC*
- *GDB front end debugger for Linux, planned*

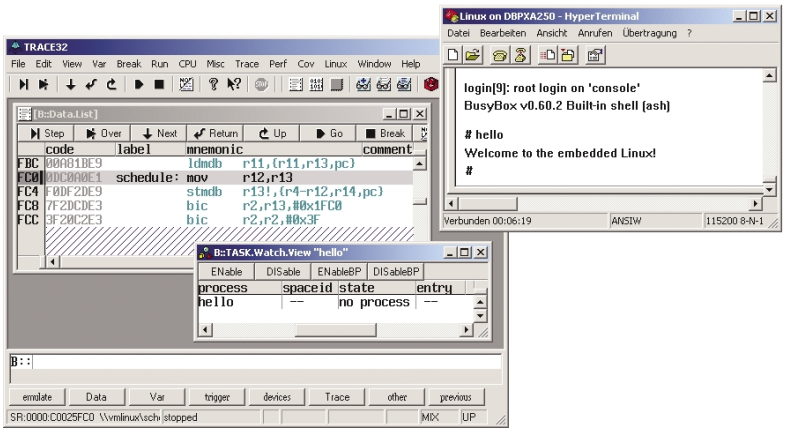


Figure 4: Process ended

Tips for using the ARM-ETM

One of the most regular questions our customers ask is "do we have any tips on the optimum use and configuration for the ARM-ETM". The purpose of this article is to try and summarize the most important options for users of the ETM on the ARM7 and the ARM9, and to provide a preview of the ARM11.

The ETM (Embedded Trace Macrocell) is an on-chip real time trace module for the ARM architectures. This trace module provides the following functions:

- *Output of the entire program and data flow via a special trace port*
- *Triggers and filters for controlling this output*

This trace output can be recorded and analyzed with the TRACE32 PowerTrace from Lauterbach, a development tool consisting of a fully featured JTAG debug system and a 512-Mframe deep trace memory.

The trace port

The ETM configuration first of all defines a physical trace port, see Figure 1.

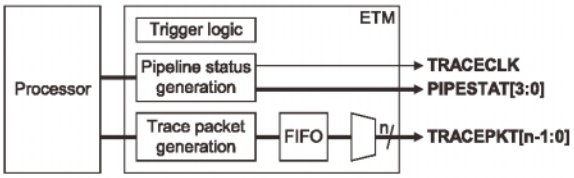


Figure 1: ETM trace port for ARM7/ARM9

TRACECLK: *The ETM is normally set to operate at the same frequency as the ARM processor.*

PIPESTAT: *In each clock cycle the ETM outputs details of events in the instruction pipeline of the processor (e.g. instruction taken, branch taken etc.) via the pipeline status pins (3 pins for the ARM7 or 4 pins for the ARM9)*

TRACEPKT: *The program flow is always presented via TRACEPKT (4/8 or 16 pins). For this purpose the compressed branch destination address is output in a trace packet for each indirect branch. In these packets the term "indirect branch" is used in a very wide sense as it means all program branches for which the branch destination is not defined prior to the program run time (i.e. loads/adds to the PC and exceptions, but not branch instructions themselves). With the aid of the source code and this information the entire program flow can be reconstructed. For data accesses the ETM user can personally decide whether they want to make only the data address, only the data contents or both available via the trace port.*

Whereas program only trace is very low bandwidth and can be traced through a 4-bit port, when data trace is enabled the amount of trace bandwidth required increases rapidly. Since the entire address and data information has to be output via the 4/8 or 16 pins of TRACEPKT, it is obvious that this data flow can possibly overload the trace port. This is most likely to occur when several data accesses are carried out in quick succession. In order to prevent overloading of the trace port a FIFO is connected upstream of TRACEPKT. This provides intermediate storage for all trace packets that cannot be output via TRACEPKT immediately.

Dimensioning of the ETM

ARM Limited now offers a number of standard configurations for the ARM-ETM. From our experience a port width of 8 or 16 pins for TRACEPKT and a FIFO size of at least 45 bytes has proved viable in practice (known as Mediumplus or Large). For ETM ports with lower specifications you should be aware of the possible limitations, especially with the data flow output, as this can very quickly result in an overflow of the FIFO.

FIFO overflow

A FIFO overflow occurs whenever there are so many trace packets queuing to be output that the FIFO connected upstream of TRACEPKT is no longer able to provide intermediate storage for all of them. The result is that important trace packets can be lost. There are two methods available to prevent this highly undesirable situation.

1.) FIFOFULL logic

Most ARM-ETM standard configurations contain FIFOFULL logic. With the aid of this logic it is possible to stop program execution (STALL) whenever the FIFO threatens to overflow. This strategy does, however, have the disadvantage that it slows down the program run time. An additional disadvantage is that the ARM-ETM itself does not provide any facility for determining later, during

the trace analysis, how high the run time loss actually was.

2.) Reduction of the trace packets

Another way of preventing a FIFO overflow is by reducing the number of trace packets produced by the data accesses. An option would be to limit the output of the data accesses to write cycles. This already makes a FIFO overflow considerably less likely. However if one wishes to see the entire data flow for effective debugging, Lauterbach recommends limiting the output of the trace packets for the data accesses to read cycles. The TRACE32-PowerTrace can reconstruct all write cycles on the basis of the read cycles by means of special algorithms. See also ***www.lauterbach.com/cts.gif****.*

If these strategies for reducing the trace packets are not sufficient to prevent a FIFO overflow it is possible to utilize the filters of the ARM-ETM to further reduce the data flow. The first recommendation is to filter out the stack area from the data flow as the stack operations in particular often result in a FIFO overflow when changing functions. Another very obvious idea is to program the trace filters so that only data accesses to selected variables/ memory areas are recorded.

Smart Trace

When using TRACE32-PowerTrace even a large number of gaps in the trace due to overflow of the FIFO is not necessarily a problem. A special process - Smart Trace – makes it possible to fill in most of these gaps in the trace. Exceptions and read accesses to ports are the only things that cannot always be reconstructed completely. See also figure 2 and ***www.lauterbach.com/smarttrace.html****.*

Tips for the hardware design

In order to achieve optimum recording of the trace information by TRACE32-PowerTrace, the following points should be taken into account

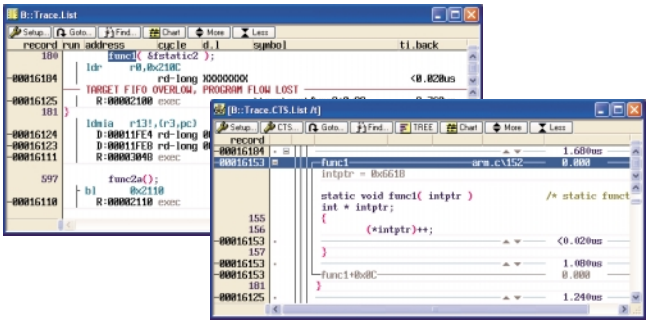


Figure 2: Filling the trace gaps by reconstructing the program and data flow with SmartTrace

for the design of the destination hardware:

- *Lines for the ETM signals should be as short as possible and of equal length*
- *3 ns setup and 1 ns hold time*
- *Destination system should be capable of driving a 50-ohm parallel termination*

ETM for ARM11

The ETM for the ARM11 offers three key new features.

1.) Higher compression rates

Program trace is improved seven fold. Data trace, which matters most for overflow considerations, is improved on average 25% through leading zero removal.

2.) Decoupling of the trace clock from the processor clock

The trace information, specifically the pipeline status information, is no longer output synchronized to the processor clock. Instead the pipeline status information is included in the trace packet as the header.

3.) Suppression of the data packets in the case of FIFO overflow

Instead of stopping program execution if the FIFO threatens to overflow, the output of the data flow is suppressed.

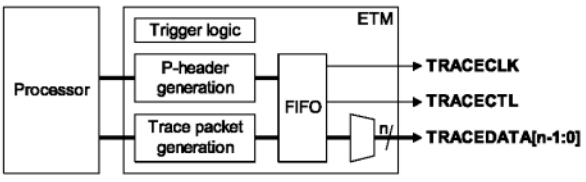
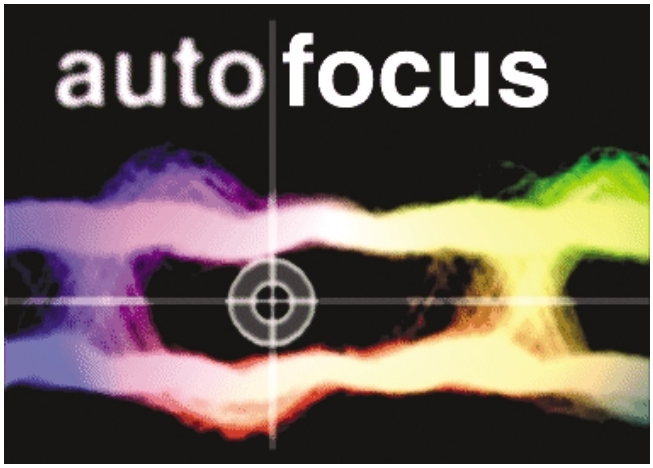


Figure 3: ETM trace port for ARM11

Optimal sampling of trace channels



Today microcontrollers that output trace information via the trace port at frequencies beyond 200 MHz are no longer unusual. The time window for sampling the valid data is thus becoming ever narrower. Consequently small run time differences between the trace channels or other minor faults in the target hardware can easily become a source of errors for the trace recording. As a result fine tuning of the sampling instant is becoming an increasingly important factor. For this reason Lauterbach is rolling out a new generation of preprocessors and NEXUS adapters in mid-2004 that will have an autofocus capability. Autofocus supports the automatic setting of the optimum sampling instant as well as automatic setup of other hardware features through FPGA-based hardware.

One of the main functions of a preprocessor/NEXUS adapter is to receive high speed data from the trace port of the target systems' microcontroller and transfer it at lower speed into the trace memory. At high frequencies the quality of the trace signals can be impaired by a number of different factors including line length, resonances, cross-talk from other signals, tolerances of the components, and so forth. If the sampling instant for the trace data is not optimally selected the error rate in the recording of the program and data flow increases.

Up until now the sampling instant had to be correct by design and only the setting of the reference voltage has been automatic on most preprocessors/NEXUS adapters. The reference voltage could be adjusted manually to reduce the error rate and it was possible to switch on a termination voltage to improve the signal amplitude. However, this concept might not be sufficient at high frequencies beyond 200 MHz. A considerably more complex procedure has been developed to assure signal integrity at those frequencies.

Assuring signal integrity takes place in two steps:

1. Hardware adjustment

As soon as the microcontroller outputs data via the trace port the preprocessor/NEXUS adapter determines the following data via FPGA-based hardware:

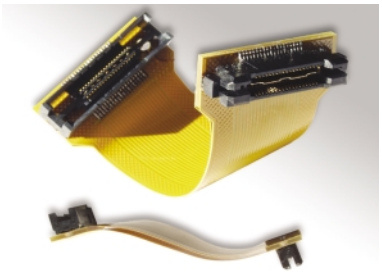
- *the ideal termination voltage*
- *the ideal reference voltage separately for all clock and data channels*
- *the ideal clock delay, and*
- *the ideal delay for every single data line*

2. Testing the trace channels

For testing the trace channels the microcontroller must then output a fixed data pattern via the trace port. This data is then transferred to the trace memory by the preprocessor/NEXUS adapter and can be verified by the TRACE32 software. That way it is possible to verify that the automatic hardware adjustment was successful.If necessary the fine tuning of the sampling instant can be re-adjusted for individual trace channels.

The first NEXUS adapter equipped with autofocus functionality will be the NEXUS adapter for the Super10, followed by the preprocessor for the ARM-ETM.

Flex extensions



Mictor flex extension, 55 mm, vertical to vertical

Since mid-2003 Lauterbach has offered a range of "Flex Extensions" for the various different preprocessors/NEXUS adapters. These extensions guarantee good transmission quality, even at high frequencies, between the destination system and the TRACE32 development tool. See also

www.lauterbach.com/adflex.html

Complex trigger unit

Anyone who has struggled with complex error scenarios or random system crashes during software integration will appreciate the advantages of an efficient trigger unit. The benefits include:

- *planned, systematic troubleshooting*
- *stopping the trace recording or program execution at specific points*
- *optimal utilization of the trace memory*
- *selective recording and analysis of information*
- *improved run time monitoring of the entire system*

The efficient use of a trigger unit also calls for a good knowledge of the hardware and software of the destination system as this is the basis for developing suitable trigger strategies. However, the work invested in this preparation is always more than compensated by substantial time savings in the debugging process.

In order to provide the developer with more effective support in tracing difficult system faults the following Lauterbach development tools have been equipped since November 2003 with a "complex trigger unit":

- ***PowerTrace***
- ***PowerProbe***
- ***PowerIntegrator***

We will now review the trigger and filter functions of this "complex trigger unit" as it applies to the various members of the PowerTools family.

PowerTrace

Although the complex trigger unit is a permanent component of the PowerTrace hardware, different software is needed for each of the individual processor architectures.

1. Bus trace systems

Architectures that visualize the entire program and data flow on the external address and data bus are easiest to work with. An example of this type of architecture is the Infineon EGOLD.

If, for example, you only want to record the cycles of a particular function or only the write accesses to a specific variable, the address visible on the address bus can be used as the input event for the complex trigger unit. See also Figure 1.

2. FlowTrace systems

Processor architectures that output the program and data flow via a special trace port usually operate by using highly compressed trace data. This compression allows the entire program and data flow to be accessed via a narrow trace port even at high frequencies.

In order to be able to use this highly compressed data for the complex trigger unit it is necessary to reconstruct the complete program and data addresses out of the compressed data in real-time. At the time of writing the only architecture that allows a problem free implementation of such a reconstruction is the SH4 family from Renesas Technology. All other architectures still rely upon the filter and trigger options that the semiconductor manufacturer has implemented.

3. NEXUS systems

With architectures that have a NEXUS interface it is not possible to undertake a runtime reconstruction of the program addresses. But it is normally possible to carry out a reconstruction of the data addresses.

The NEXUS protocol does, however, have a number of message types that can be used effectively as input events for the complex trigger unit:

- *Watchpoint hit messages*
- *Ownership trace messages*
- *Hardware event messages, etc.*

As an example the NEXUS interface can indicate that a specific function has written a defined data value to a

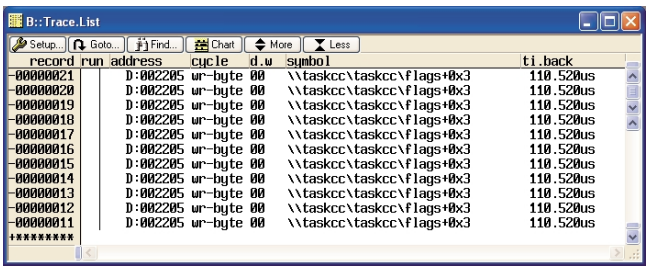


Figure 1: Selective data trace for a bus trace (Infineon EGOLD)

TRACE32-POWERTOOLS

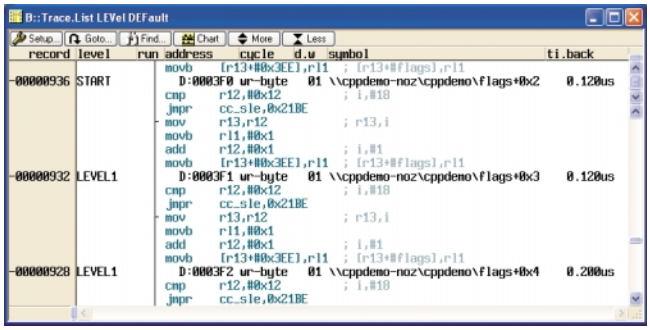


Figure 2: Changing the trigger level after writing 1 to flags[2] (STMicroelectronics NEXUS Super10)

variable via a "watchpoint hit message" . This watchpoint hit message can then be used by the complex trigger unit to record a specific number of cycles in the trace and stop the program after that. Figure 2 illustrates an example of the use of the complex trigger unit for NEXUS.

After the processor architecture has been configured for the PowerTrace the complex trigger unit offers the following options:

- *Filtering of the trace information*
- *Definition of complex trigger conditions that can lead to termination of the trace recording or program execution*
- *Time supervision of individual program sections*
- *Generation of a trigger signal for an external device*

The complex trigger unit has the following resources at its disposal:

- *At least 2 address/data comparators*
- *3 x 45-bit time and event counters*
- *4 trigger levels*
- *2 trigger flags*
- *2 trace markers*

PowerProbe

With the Lauterbach PowerProbe the developer can integrate signals from port lines, communication interfaces and external interrupts into the debug environment. The PowerProbe supports 64 input channels with a sampling rate up to 100 MHz or 32 input channels with a sampling rate up to 200 MHz or 16 input channels with a sampling rate up to 400 MHz. The trace has a memory depth of 256K entries.

The complex trigger unit includes 8 data selectors which generate trigger events. Each data selector can be configured to trigger on a freely definable data pattern over all 64 input pins.

Together with three 45-bit time/event counters, 4 trigger levels, etc. complex trigger conditions can now be defined which:

- *filter the trace information*
- *interrupt recording to the trace memory*
- *start the pattern generator*
- *generate trigger signals for external devices*
- *or ensure that the debugger stops program execution*

Figure 3 shows an example in which the states of input channels x.0-x.2 are only recorded in the trace memory as long as input channel x.3 is active.

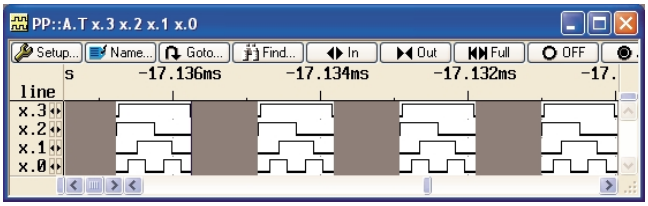


Figure 3: Selective recording of x.0-x.2 as long as x.3 is active

PowerIntegrator

In summer 2003 the PowerIntegrator was added to the PowerTools product family to provide a universal bus trace and further extend the product family. The PowerIntegrator can be connected to the target system via standard probes to record bus trace data. A range of support packages provide a quick and simple means of transferring the recorded input signals into a program and data flow trace. The trace contents processed in this way can then be dis-

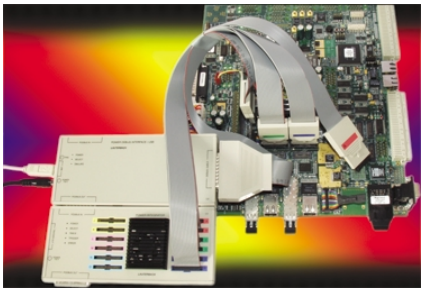


Figure 4: TRACE32-ICD in-circuit debugger and PowerIntegrator on a PowerQUICC III evaluation board

TRACE32-ICD
PowerTools
the complete solution
for system on-chip designs

Visit www.lauterbach.com for more information

played and evaluated together with the debug information.

The PowerIntegrator has 204 freely configurable data channels, of which 12 have the option of being used as clock inputs. The following sampling rates are possible via active probes:

- **500 MHz in timing mode**
- **200 MHz in state mode**

The sampling threshold for the signals can be selected optionally in a range from 0 to 5 V.

The PowerIntegrator has a trace memory depth of 512 KFrames. Each entry in the trace memory is provided with a time stamp. The time stamp has a width of 48 bits and operates at a resolution of 4 ns.

In a similar way as the PowerProbe, the input events for the complex trigger unit can be signals from individual data channels or data channels can be combined to form a trigger word (max. 204 Bit). When using the PowerIntegrator the most important trigger words will be on the address and data bus. Both trigger words are already predefined by the support package and are thus available directly.

The time monitoring of a function provides an example of the use of the complex trigger unit when using the PowerIntegrator. One trigger word is defined for the entry address and another for the exit address of the function. By setting up the filtering only the entry and exit address of the function are then recorded in the trace memory. The result can also be evaluated statistically by the PowerIntegrator. See also Figure 5.

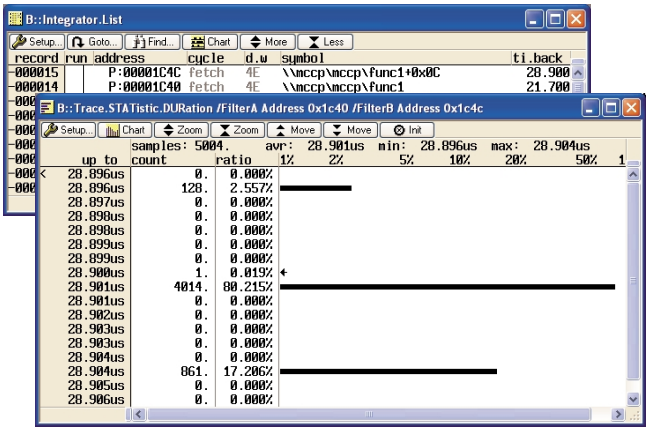


Figure 5: Time monitoring of a function through selective recording of the function entries and exits

Please send me information on:

We use the following processors:

_____

I am interested in the following development tools:

- ☐ ***TRACE32-ICD***
- ☐ ***TRACE32-PowerTrace***
- ☐ ***TRACE32-PowerIntegrator***

Sender:

Name _____

Company _____

Address _____

Phone _____

email _____

LAUTERBACH

Phone: (508) 303 6812 FAX: (508) 303 6813

email: info_us@lauterbach.com