TRACE32®
IN-CIRCUIT-EMULATOREN & DEBUGGER
ONE SYSTEM FITS ALL!

# NEWS

2002

## NEUHEITEN ZUR EMBEDDED SYSTEMS 2002

Weitere Informationen im Internet: www.lauterbach.com

## *10.000 TRACE32-ICD Debugger ausgeliefert*

*Dass bei Lauterbach „Best-of-class"-Emulatoren gebaut werden, war 1994 bei Kunden schon bekannt. Dessen war man sich auch bei Lauterbach bewusst, denn der große Erfolg bis dahin gab einem schließlich recht. Ob man aber zu dem Zeitpunkt für den 68332 einen BDM Debugger entwickeln und auf den Markt bringen sollte, darüber war man sich jedoch noch nicht im Klaren. Schließlich würde man hier ein ganz anderes Marktsegment als gewohnt betreten.*

*Würde ein solches Tools dem Image der Firma als Lieferant von High-End-Systemen gerecht? Wären die Kunden mit den eingeschränkten Möglichkeiten eines solchen Tools zufrieden? Schließlich durfte ein solches Tool nicht sehr viel kosten - käme man dann überhaupt auf entsprechende Stückzahlen, um damit Geld zu verdienen? Die Antwort war trotzdem schnell gefunden: ja man musste hier einsteigen, denn die Kunden wollten ein solches Tool, und wenn Lauterbach es nicht anbieten würde, dann würden es andere tun.*

*So wurde also 1994 der erste TRACE32-ICD entwickelt, auch hier mit der Prämisse, das allerbeste aus den technischen Möglichkeiten herauszuholen. Im nächsten Jahr schon, also **1995**, wurde der **erste TRACE32-ICD** für den 68332 vorgestellt und ausgeliefert. Es folgten dann Schlag auf Schlag TRACE32-ICD für **68HC12, 68HC16, ARM7** und **PowerPC**. Das mit den genügend Stückzahlen klappte natürlich nicht sofort, im ersten Jahr wurden genau 26 Stück verkauft - keine sehr ermutigende Bilanz. Doch dann folgten in den nächsten Jahren stete Steigerungen dieses Geschäfts. Im Jahr 2001 konnte dann ein ganz besonderer Erfolg gefeiert werden, die Schwelle von **10.000** ausgelieferten **TRACE32-ICD** Debuggern wurde überschritten - wer hätte das 1994 gedacht!*

*Mit den Jahren kam auch die Breite an unterstützten Architekturen, mittlererweile sind es 25 verschiedene. TRACE32-ICD ist auch längst nicht mehr „nur" ein Debugger, sondern kann durch zahlreiche Komponenten, wie etwa **Echtzeit-Trace** oder Logikanalysator ergänzt werden.*


*Mit den Jahren waren auch die Anforderungen der Kunden an diese Tools gestiegen - z.B. war Software Quality Assurance (SQA) zu einem wichtigen Thema geworden. Um diesen Anforderungen gerecht zu werden, wurde im letzten Jahr der mächtigere Verwandte des TRACE32-ICD, die **TRACE32-PowerTool** Familie vorgestellt. Mit dieser neuesten Produktgeneration ist nun endgültig bewiesen, dass On-Chip-Debug-Tools weit mehr leisten können, als manch einer vermutet, wenn er an JTAG-Debugger oder ähnliches denkt.*

*Was sich hier und natürlich auch bei unseren In-Circuit-Emulatoren der neuesten Generation getan hat, lesen Sie bitte auf den nächsten Seiten dieser **TRACE32-News**.*

*Besuchen Sie uns doch am besten gleich auf der **Embedded Systems 2002** in Nürnberg, um sich hautnah von den neuen Leistungsmerkmalen unserer Tools zu überzeugen. Wir laden Sie dazu recht herzlich ein, eine kostenlose Gastkarte können Sie gerne bei uns anfordern.*

LAUTERBACH

# Produktübersicht

*Mit der Produktlinie TRACE32-PowerTools bietet Lauterbach Hochleistungs-Entwicklungswerkzeuge auch für System-on-Chip (SoC) Designs. TRACE32-PowerTools umfaßt folgende Produkte:*

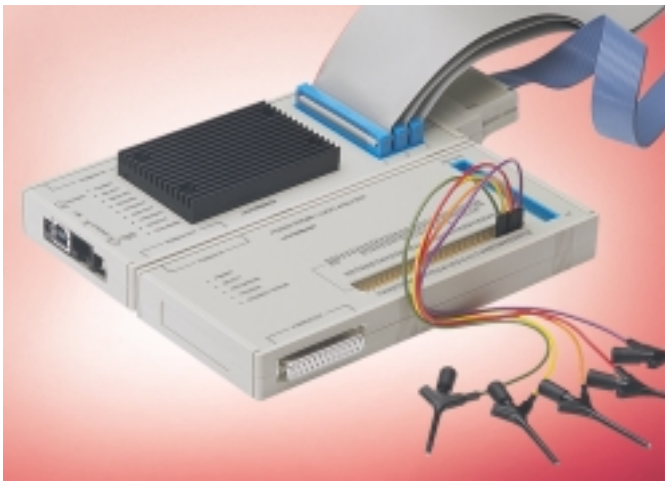
- ***PowerDebug/Ethernet***
- ***PowerTrace***
- ***PowerProbe***
- ***PowerIntegrator***

*Ziel der neuen Linie ist es, schnelle und leistungsfähige Entwicklungswerkzeuge anzubieten, die umfassende Analysen des Laufzeitverhaltens von SoCs erlauben, sowie alle notwendigen Qualitätssicherungstests unterstützen.*

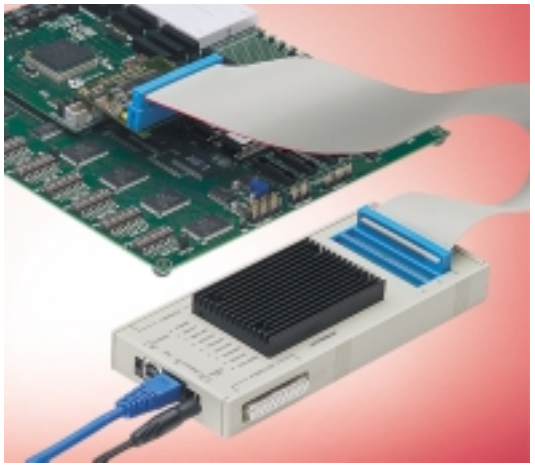
*Durch die Integration von Portleitungen, Kommunikationsschnittstellen, Interrupt etc. in die Entwicklungsumgebung soll eine einheitliche Bedienung und ein optimales Zusammenspiel aller beim Test und der Integration von SoCs eingesetzten Werkzeuge erreicht werden.*

*Das brandneue Produkt PowerIntegrator ist für die schnelle Inbetriebnahme von Evaluation Boards ausgelegt und kann natürlich auch für die Produktentwicklung eingesetzt werden. Über seine 204 Kanäle können sowohl alle Busse als auch alle wichtigen Peripheriesignale aufgezeichnet und überwacht werden.*

***PowerDebug/Ethernet*** *ist ein schneller, professioneller Debugger für alle gängigen On-Chip Debugging Standards.*

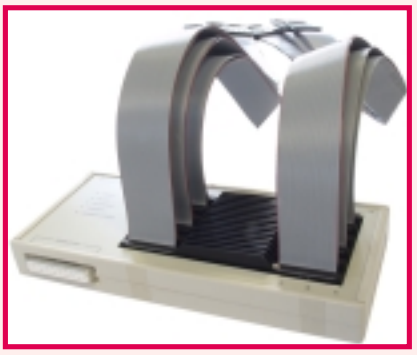


***PowerProbe*** *bietet die Möglichkeit bis zu 64 Signale in die Debug-Umgebung zu integrieren und ihr Zeitverhalten auszuwerten.*



***PowerTrace*** *ist ein Debugger und ein 16 MFrames großer Tracespeicher in einer Box. Es werden alle gängigen On-Chip Debugging Standards und Traceports (NEXUS, ARM-ETM ...) unterstützt.*

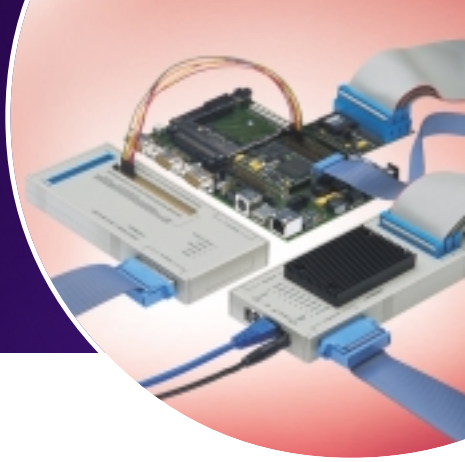
NEU



***PowerIntegrator*** *ist ein Logic Analyser mit dem 204 Kanäle bis zu einer Abtastfrequenz von 500 MHz aufgezeichnet werden können. Als Anschlüsse werden MICTOR- und Standard 100 mil Stecker unterstützt.*

DIE NEUE PRODUKTLINIE

# TRACE32-POWERTOOLS



## PowerDebug/Ethernet

*PowerDebug/Ethernet ist der schnellste Debugger, der alle gängigen On-Chip Debugging Standards unterstützt. Für den Anschluß an das Hostsystem steht integriert eine USB- bzw. Ethernet Schnittstelle zur Verfügung.*

*Die Hardware des Debuggers ist universell. Die Anpassung an eine bestimmte Architektur wird durch ein architekturspezifisches Debug Kabel und die entsprechende Software erreicht. Ein schneller 32-Bit RISC Prozessor im Power Debug/Ethernet Modul garantiert eine gute Gesamtsystemperformance, eine professionelle Unterstützung von komplexen Debug-Funktionen und ein schnelles Download. Tabelle 1 zeigt beispielhaft einige Ladezeiten. Grundlage für die Messungen bildete ein Binärfile, das über eine 100 MBit Ethernet Schnittstelle geladen wurde.*

| Prozessor | Ladezeit |
|---|---|
| **TC10GP (TriCore)** | 1,25 MByte/s bei 25 MHz JTAG Clock |
| **ARM940** | 1,0 MByte/s bei 30 MHz JTAG Clock |
| **MPC860** | 1,7 MByte/s bei 40 MHz BDM Clock |
| **SH4** | 820 KByte/s bei 20 MHz JTAG Clock |

**Tabelle 1**

## PowerTrace

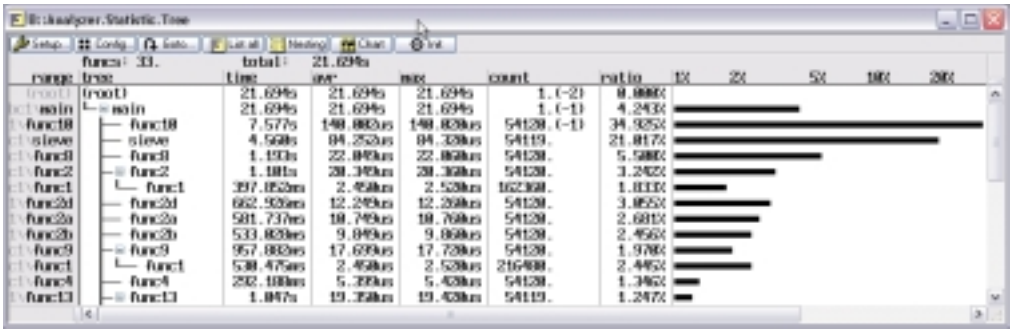


**Bild 1: Analyse der Funktionslaufzeiten und des Aufrufverhaltens**

*PowerTrace ist ein PowerDebugger und ein* ***16 MFrame*** *großer Tracespeicher in einer Box. Auch von diesem Tool werden alle On-Chip Debugging Standards und zusätzlich alle gängigen Traceports wie NEXUS, ARM-ETM ... unterstützt.*

*Die PowerTrace Hardware ist universell. Die Anpassung an eine bestimmte Architektur erfolgt über:*

- *Einen architekturspezifischen NEXUS Adapter und die dazugehörige Software.*
- *Ein architekturspezifisches Debug Kabel und einen architekturspezifischen Preprozessor zum Abtasten der Tracedaten sowie über die dazugehörige Software.*

*Das neue Konzept von TRACE32-PowerTrace ermöglicht es, umfassende Laufzeitmessungen und Performance Analysen sowie Qualitätssicherungstests durchzuführen. Diese neuen Power Features basieren hauptsächlich auf dem sehr großen Tracespeicher. Laufzeitmessungen und Analysen des Aufrufverhaltens des Programms lassen sich beispielsweise über einen längeren Zeitraum durchführen. Bild 1 zeigt eine solche Analyse, die mit einem NEXUS Debugger für den MPC565 durchgeführt wurde.*

*Basierend auf den umfassenden Informationen aus dem Tracespeicher läßt sich auch eine Codeabdeckungs-Analyse durchführen. Zunächst wird per Mouse-Click der Traceinhalt in eine Coverage Datenbank übertragen. Dort wird dann per Software ausgewertet, inwieweit der gesamte Code durchlaufen wurde. Durch eine Analyse des Sprungverhaltens läßt sich außerdem ermitteln, ob während des*

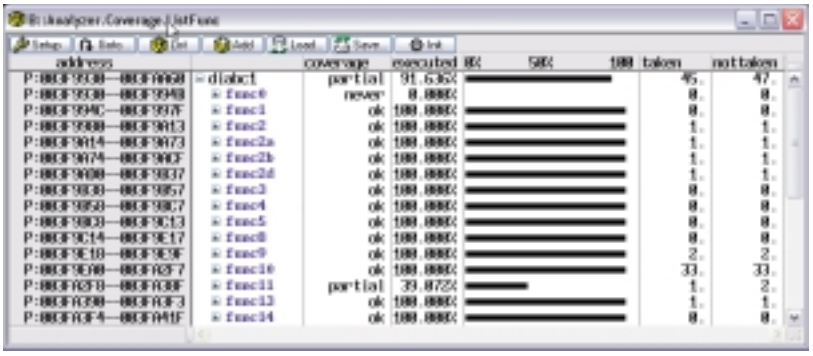


**Bild 2: Codeabdeckung und Analyse des Sprungverhaltens**



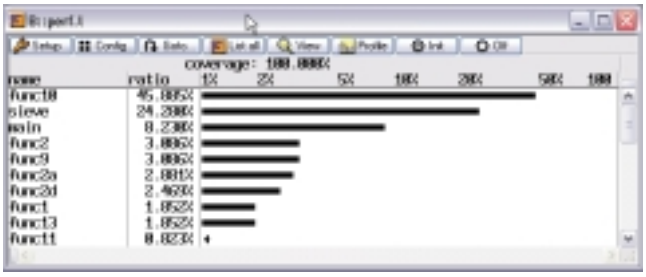


Bild 3: Langzeit-Performance Analyse

*Testlaufs alle Programmzweige durchlaufen wurden. Eine solche Auswertung zeigt Bild 2.*

*Zudem bietet der PowerTrace noch eine Langzeit-Performance Analyse. Um beispielsweise zu ermitteln, welche Funktion anteilig an der Gesamtlaufzeit die meiste Zeit verbraucht, wird alle 2 ms die Aufzeichnung im Tracespeicher kurz angehalten, um den aktuellen Programmzähler zu ermitteln. Diese Messung wird dann statistisch ausgewertet . Durch dieses Meßverfahren bleibt das Echtzeitverhalten vollkommen unbeeinflußt.*

## PowerProbe

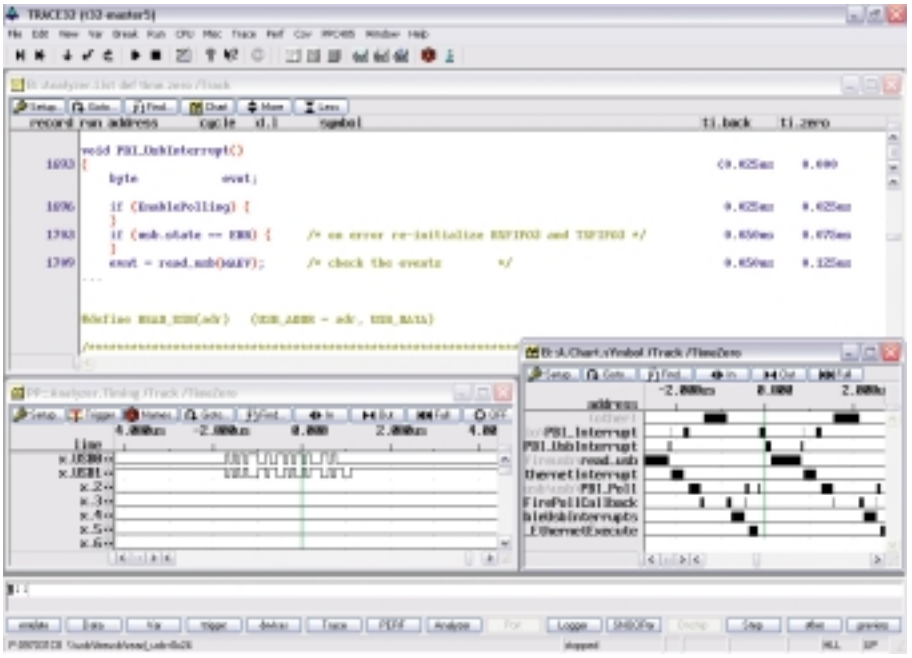


Bild 4: Source Code, Programmfluß und Steuersignale auf einen Blick

*Bei den Tests und der Integration von System-on-Chip Designs spielt das Zeitverhalten einzelner Portleitungen, Kommunikationsschnittstellen und Interrupts eine wesentliche Rolle. Zudem gibt es immer einige Zielsystemsignale, die entscheidend für das Gesamtverhalten des Systems sind.*

*An einem typische Entwicklungsplatz für ein SoC Design wird heute zum einen ein Debugger/ Trace eingesetzt, um den Programmtest und die Analyse des Echtzeitverhaltens des Programms durchzuführen. Gleichzeitig werden die Signale, die vom Programm zur Steuerung der Zielhardware gesetzt und überwacht werden, von einem zweiten Gerät, meist einem Logic Analyser, der von einem anderen Hersteller stammt, aufgezeichnet und ausgewertet. Kämen alle eingesetzen Werkzeuge aus einer Hand, hätte das für den Entwickler entscheidende Vorteile:*

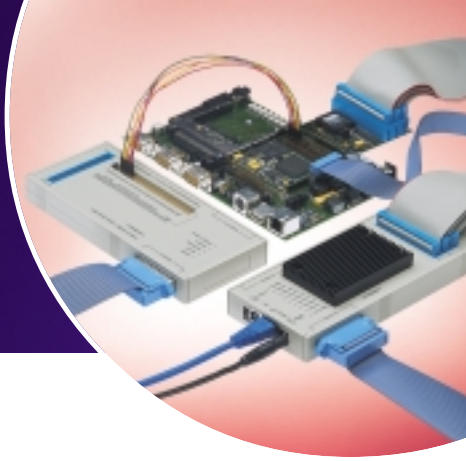
- *Einheitliche Bedienung und gleichzeitige Darstellung*
- *Zeitkorrelation zwischen den einzelnen Aufzeichnungen*
- *Kostengünstige Lösung*

*Mit der PowerProbe bietet Lauterbach nun diese Möglichkeit. Bis zu 64 Signale können einfach in die Debug-Umgebung integriert werden. Bild 4 zeigt, wie nun Source Code, Programmfluß-Informationen und ausgewählte Steuersignale gemeinsam dargestellt werden können. Die PowerProbe ist eine separate Box, die an jede TRACE32 Debugger/Trace Konfiguration angesteckt werden kann.*

*Es ist zudem möglich die PowerProbe zusammen mit einer TRACE32 Hostschnittstelle (LPT oder Ethernet) als eigenständiges Werkzeug zu verwenden.*

# DIE NEUE PRODUKTLINIE

# TRACE32-POWERTOOLS



*Basiskenndaten der PowerProbe sind:*

- ***64 Eingangskanäle bis 100 MHz***
- ***16 Eingangskanäle mit 400 MHz***
- ***Eingangsspannung 0-5 Volt, (max. 15 Volt)***
- ***Schwellenspannung 1.4 Volt, einstellbar 1.0 Volt***
- ***128K/256K Tracetiefe***
- ***10 ns Zeitstempel bei State Aufzeichnung***

*Für die Aufzeichnung der Eingangskanäle sind folgende Aufzeichnungsmodi möglich:*

- *Aufzeichnung mit einer frei wählbaren Abtastrate.*
- *Transitionaler Mode, d.h. es wird immer dann aufgezeichnet, wenn sich der Zustand eines Signals ändert. Dieser Modus eignet sich besonders zur Aufzeichnung von langsamen Peripheriesignalen.*
- *Aufzeichnung von 16 ausgewählten Signalen bei sehr hoher Frequenz.*

*TRACE32-PowerView bietet natürlich vollen Bedienungskomfort. Es können so:*

- *Eingangssignalen logische Namen und Polaritäten zugeordnet werden.*
- *Zur einfacheren Auswertung mehrere Signale zu einem Hex-Wort zusammengefaßt werden.*
- *Funktional zusammengehörige Signale, wie beispielsweise alle UART Leitungen, gruppiert und gemeinsam dargestellt werden.*

*Zudem werden bei der Tracedarstellung Standard-Schnittstellen-Protokolle wie RS232, CAN, USB ... unterstützt. Bild 5 zeigt eine solche Tracedarstellung.*

*Für die Analyse von Fehlerzuständen verfügt die PowerProbe über eine eigene Triggerung. Folgende Triggerquellen sind möglich:*

- *64 Signale bei synchroner Abtastung*
- *8 Signale bei asynchroner Triggerung*

***Beispiel 1 (synchrone Abtastung):*** *Die PowerProbe stoppt mit Hilfe des Debuggers die Programmausführung, wenn ausgewählte Eingangskanäle einen bestimmten Zustand haben.*

***Beispiel 2 (asynchrone Triggerung):*** *Die PowerProbe stoppt mit Hilfe des Debuggers die Programmausführung, wenn ein ausgewähltes Eingangssignal länger als 100 ns aktiv ist.*

*Die PowerProbe gibt es in zwei Ausbaustufen: Eine einfache Variante mit einer Tracetiefe von 128K und eine Vollvariante mit einer Tracetiefe von 256K. Die Vollvariante umfaßt zudem einen Triggersequenzer zum Auffinden komplexer Fehler. Dieser stellt 4 Triggerebenen, 3 universelle Zeit-/Ereigniszähler, 4 Datenselektoren etc. zur Verfügung. Zur Programmierung des Triggersequenzers kann eine Trigger Dialog Box verwendet werden.*

*Die Vollvariante bietet zudem einen Pattern Generator, über den mittels einer eigenen Makrosprache auf 9 Kanälen Testsequenzen generiert werden können, um Signale für die Eingangskanäle zu simulieren.*

*Zur Adaption der PowerProbe an das Zielsystem können Standard Meßclips verwendet werden. Für SOPC (System-On-a-Programmable Chip) Designs bzw. ASICs steht ein spezieller Meßadapter zur Verfügung. Eine umfassende Beschreibung zum Einsatz dieses Meßadapters finden Sie auf der nächsten Seite.*

*Ausführliche Informationen über die PowerProbe finden Sie unter:* ***http://www.lauterbach.com/powerprobe.html.***

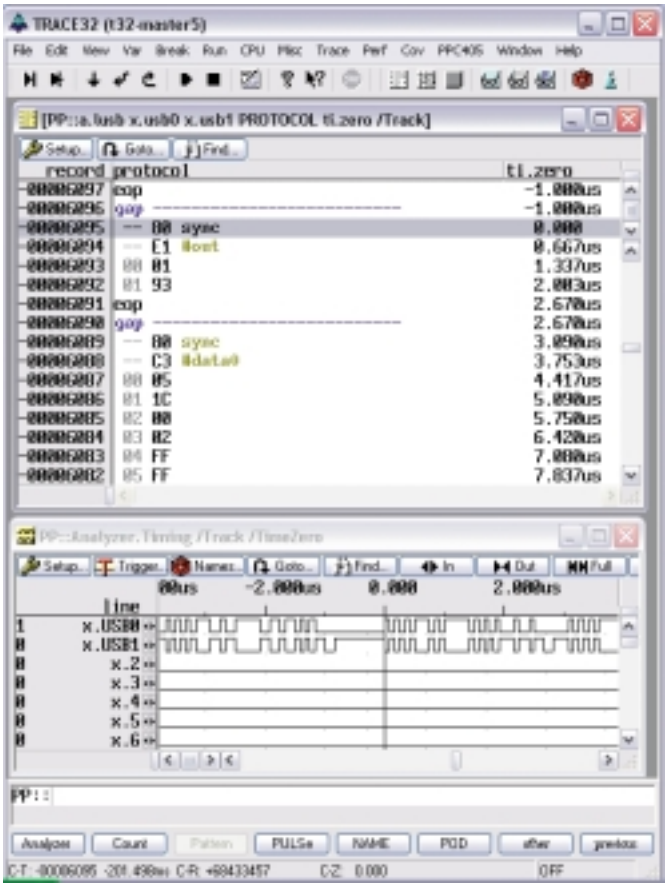



***Bild 5: Tracedarstellung als USB-Protokoll***



# Messen statt simulieren

*Eine kritische Phase in der Produktentwicklung ist die Systemintegration, d.h. das Zusammenführen von Hard- und Software zu einem lauffähigen System. Bei Entwicklungen, die auf Standard 8- bzw. 16-Bit Mikroprozessoren basierten, wurden hier üblicherweise In-Circuit Emulatoren zur Analyse der Software und zur Überprüfung der Hardware-Funktionen eingesetzt. Bei hochintegrierten ASICs oder System-On-a-Programmable-Chip (SOPC) Designs, die auf 32-Bit RISC Prozessoren basieren, ist das nicht mehr möglich, da weder der Adress-/Datenbus noch die periphere Hardware direkt zugänglich sind.*

*Für das Testen der Software und für die Überwachung des Programm- und Datenflusses stellen die meisten Cores heute On-Chip Debugging Logik und sogenannte Traceports zur Verfügung. Bei der Systemintegration spielen jedoch besonders die Peripheriefunktionen eine wichtige Rolle. Ihre Interaktion mit den Hardwarekomponenten der Zielumgebung und der Software kann auf unterschiedliche Weise getestet werden.*

## Software Simulation

*Um auch in abgeschlossenen Systemen eine Systemintegration durchführen zu können, wurde als Ergänzung zu der auf VHDL oder VERILOG basierenden RTL-Simulation die sogenannte Co-Simulation von Hard- und Software entwickelt. Die Simulation ist ein hervorragendes Mittel zur Verifikation überschaubarer, geschlossener Systeme.*

*Bei Embedded Designs, die sehr stark durch externe Ereignisse geprägt sind, zeigen sich jedoch sehr schnell die Grenzen dieser Simulation. Zwischen den Testvektoren und der Realität treten nämlich große Abdeckungsprobleme auf. Zudem lassen sich größere Systeme nicht mehr in einer angemessenen Zeit per Software simulieren.*

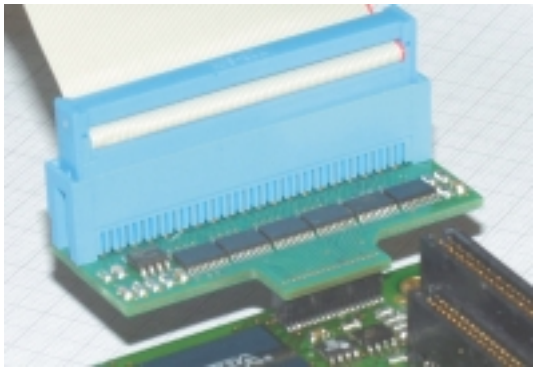


**Bild 6: SOC Adapter in der Zielhardware**

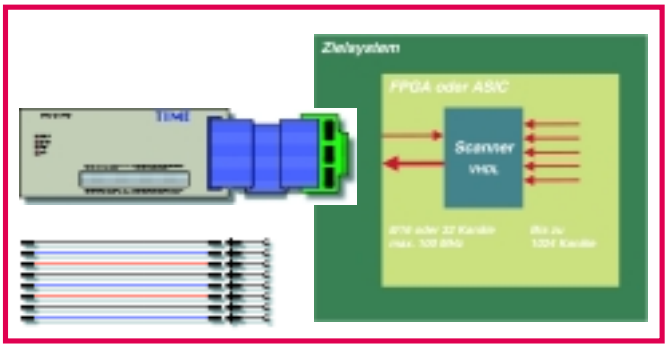



**Bild 7: PowerProbe mit SOC Adapter**

## Hardware Simulation

*Um diese Leistungslücke der Software Simulation zu umgehen, wurden Hardware Simulatoren entwickelt, die es erlauben das Chip-Design in Hardware zu modellieren, bevor das ASIC produziert wird. Das Hardware Modell kann dabei bereits in der Zielumgebung getestet werden, um so schon möglichst früh mit der Systemintegration zu starten.*

*Der Nachteil ist hier, dass das Gesamtsystem nur bis zu einer Maximalfrequenz von etwa 1 MHz emuliert werden kann und eine Hardware Simulation außerdem sehr teuer ist.*

## Test Pins

*Bei SOPC Designs besteht die Möglichkeit über Testpins einzelne Peripheriesignale herauszuführen und diese dann in ihrer Interaktion mit dem Gesamtsystem zu messen. Diese Methode erlaubt zwar maximale Frequenzen, ist jedoch wegen des erheblichen Recompilierungs-Aufwands sehr zeitintensiv.*

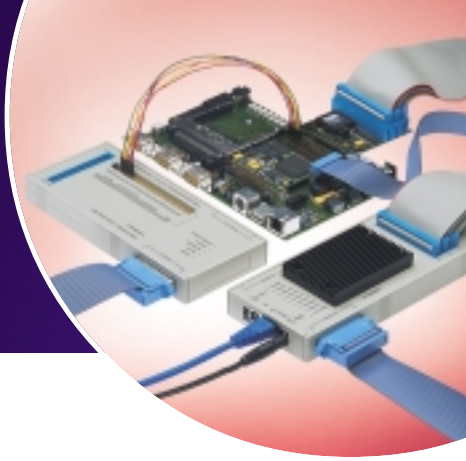
## SOC Trace

*Die von Lauterbach entwickelte SOC-Trace Technologie integriert die traditionelle, von In-Circuit Emulatoren bekannte Meß- und Auswertetechnik auf ASICs oder SOPC Designs. Dabei wurden folgende Zielsetzungen verfolgt:*

- *Einfache Integration in ASICs oder FPGA Technologie*
- *Messungen möglichst mit voller Systemfrequenz*
- *Einfache Adaption*
- *Einfache Bedienung*

DIE NEUE PRODUKTLINIE

# TRACE32-POWERTOOLS



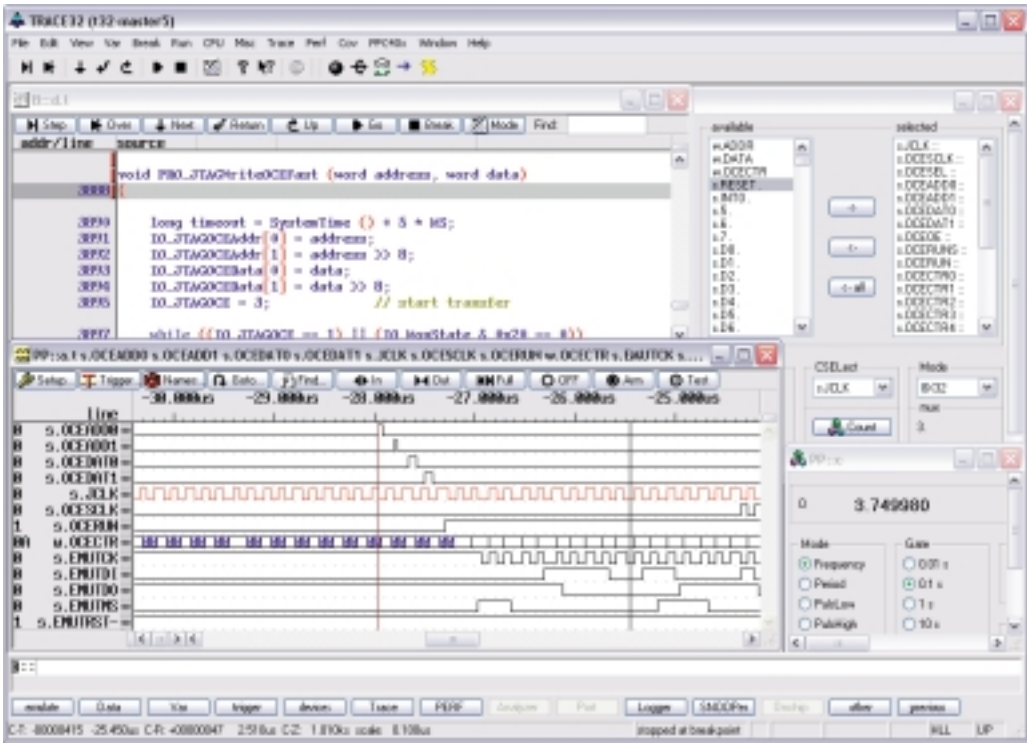


**Bild 8: Source Code und Peripheriesignale in einer einheitlichen Oberfläche**

- *Debug-Informationen, Programm- und Datenfluß sowie Peripheriesignale in einer einheitlichen Oberfläche*
- *Optimales Zusammenspiel aller Entwicklungswerkzeuge*
- *Zeitkorrelation zwischen Programm-/Datenfluß und Peripheriesignalen*
- *Kostengünstige Lösung, die für jeden Entwicklungsarbeitsplatz realisierbar ist*

*Für den SOC-Trace steht ein SOC Adapter für die PowerProbe zur Verfügung, an den bis zu 32 Signale angeschlossen werden können. Siehe dazu Bild 6 und 7.*

*Der SOC Trace ist ein Logic Analyzer mit transitionaler Aufzeichnung, der seine Meßsignale aus einem sogenannten Scanner-Modul bezieht. Das Scanner-Modul ist ein VHDL-Block, der in das ASIC oder FPGA während der Entwicklungsphase eingebettet werden kann. Es kann bis zu 1024 Kanäle im Zeitmultiplex an den SOC Trace übertragen. Dazu arbeitet das Scanner-Modul mit Blöcken von 32 Kanälen. Die Multiplex-Frequenz beträgt 100 MHz, so dass alle 1024 Kanäle gleichzeitig mit einer Frequenz von 3 MHz abgetastet werden können. Für die meisten Anwendungen kommt man jedoch mit 2 Kanalblöcken aus, so dass die maximale Abtastfrequenz bei 50 MHz liegt. Der Vorteil des Scanner-Moduls ist, dass keine Recompilierung und kein Fitten des FPGAs notwendig ist.*

*Die aufgezeichneten Signale werden in der PowerProbe demultiplext und dann als Zeitdiagramm dargestellt. Selbstverständlich lassen sich dabei auch komplexe Triggerbedingungen setzen.*

*Nachdem der SOC Trace mehr als ein Jahr im Entwicklungslabor eingesetzt wurde, läßt sich eine erhebliche Zeitersparnis bei der Systemintegration feststellen. Musste vorher zur Messung interner Knoten eines FPGAs neu kompiliert werden, genügen jetzt wenige Mouse-Clicks, um ein gewünschtes Signal zu messen und auszuwerten.*

*Zum Anschluß eines SOC Traces sollten beim Hardware-Design nach Möglichkeit 0,8 mm Feinraster- oder MICTOR-Stecker vorgesehen werden. Detaillierte Informationen zu den Steckern finden Sie unter:*

***http://www.lauterbach.com/adsoccon.html***

## PowerIntegrator

*Als Messeneuheit wird Lauterbach zur Embedded Systems 2002 in Nürnberg das neueste Mitglied der PowerTools Produktfamilie, den PowerIntegrator vorstellen.*

*Der PowerIntegrator ist ein Logic Analyzer mit den 204 Kanäle bis zu einer Abtastfrequenz von 500 MHz aufgezeichnet werden können. Damit können dann schnelle Prozessor- oder Kommunikationsbusse aufgezeichnet und in die Debug-Umgebung integriert werden.*

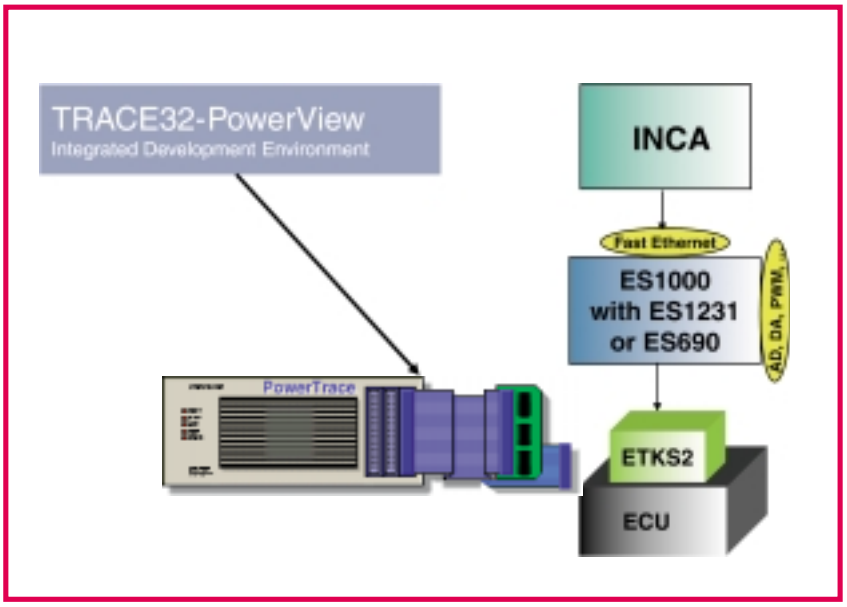
*Lauterbach möchte mit diesem Tool eine kostengünstige Lösung für jeden Entwicklungsarbeitsplatz anbieten.*



# Kooperation mit ETAS GmbH

*Lauterbach hat seit 2001 einen neuen Partner, die ETAS GmbH.*

*Es handelt sich hierbei um eine Kooperation, bei der es keine finanziellen Beteiligungen gibt. Beide Partner werden weiterhin unabhängig ihre Produkte vermarkten. Der Kern der Kooperation ist eine technische Abstimmung der jeweiligen Produkte, um zu einer Gesamtlösung zu kommen. Diese Gesamtlösung wird dann von beiden Partnern bei den Kunden empfohlen und dort bei Bedarf auch gemeinsam präsentiert. Das Kerngeschäft der ETAS GmbH liegt ja im Automotive-Bereich und auch Lauterbach ist dort sehr stark positioniert, daher gibt es schon seit langem viele gemeinsame Kunden. Aus jahrelanger Zusammenarbeit kennt man das gegenseitige hohe technische Niveau und die ähnlichen Entwicklungsphilosophien. Bei den Produkten gibt es keinerlei Überschneidung, sondern sie ergänzen sich hervorragend. Zum Einsatz kommen soll die gemeinsame Lösung speziell im Bereich Motorsteuerung.*




*Ziel ist es, dass die TRACE32 Debugger und Emulatoren integraler Bestandteil der ETAS Toolkette werden. Bei der ersten Stufe der Gesamtlösung weist das neue ETKS2 von ETAS eine offene, standardisierte Nexusschnittstelle auf, an die der TRACE32-PowerTrace angeschlossen wird. Eine enge Abstimmung bei der Nexusschnittstelle war notwendig. Nach erfolgreichen, gemeinsamen Präsentationen Mitte 2001 bei Kunden in den USA konnten schon erste Projekte gewonnen werden. Diese erste gemeinsame Lösung ist bereits ausgeliefert und bei Kunden im Einsatz.*

*Aus der Sicht von ETAS kann nun der Kunde alle Aufgaben von deren V-Modell mit einem sorgfältig abgestimmten Gesamtsystem erledigen. Aus Lauterbach´s Sicht kann man jetzt ein Produkt anbieten, das mit dem Produkt eines führenden Toolherstellers kompatibel ist. Konkret nützt dem Kunden die Kooperation, indem er jetzt bei Kalibrierungsaufgaben, etwa bei Erprobungsfahrten, einen direkten Zugriff über die Debuggeroberfläche TRACE32-PowerView auf das ETKS2 von ETAS hat und damit im Labor oder direkt im Fahrzeug debuggen kann.*

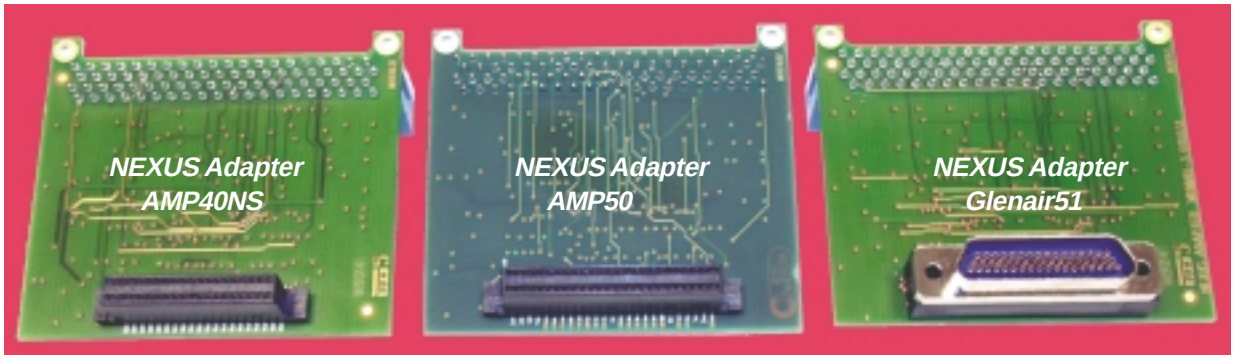
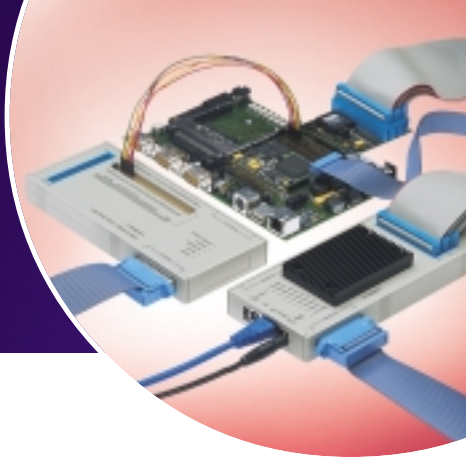



*Bild 9: Nexus Adapter für MPC56x*

DIE NEUE PRODUKTLINIE

# TRACE32-POWERTOOLS



## NEXUS Adapter

*Für den Anschluß von TRACE32-PowerTrace an den NEXUS Port der PowerPC Derivate Spanisch Oak (MPC565), Silver Oak (MPC561) und Green Oak (MPC563) stehen 3 verschiedene Adapter zur Verfügung.*

### 1. Nexus Adapter AMP40NS

*Dabei handelt es sich um einen 40-poligen Adapter auf der Grundlage der Auxilary Version des Motorola/ETAS Adapters. Für diesen Adapter steht auch ein Konverter vom Large NEXUS Modell auf das Small NEXUS Modell zur Verfügung. Beim Small NEXUS Modell ist der NEXUS Auxiliary Output nur noch 2 Bit breit. Alle für das Small NEXUS Modell notwendigen Signale sind in der Steckerbelegung für den Adapter grün gekennzeichnet.*

### 2. Nexus Adapter AMP50

*Dieser Adapter orientiert sich an der NEXUS Norm. Er basiert auf der non-robust, Auxilary Version des Connectors C. Verglichen mit dem 40-poligen Adapter bietet der 50-polige Adapter keine Erweiterung bezüglich des NEXUS Ports. Durch die zusätzlichen Signale werden lediglich spezielle Funktionen bereit gestellt. So dient beispielsweise das Signal /MSEO1 zur Erkennung eines ETKs von ETAS.*

*Auch für den NEXUS Adapter AMP50 gibt es einen Konverter vom Large NEXUS Modell auf das Small NEXUS Modell. Alle für das Small NEXUS Modell notwendigen Signale sind in der Steckerbelegung für den Adapter grün gekennzeichnet.*

| | | |
|---|---|---|
| 1 UBATT | 19 MDO0 | 36 GND |
| 2 UBATT | 20 GND | 37 MDO4 |
| 3 VSTBY | 21 MCKO | 38 GND |
| 4 TOOL_IO0 | 22 GND | 39 MDO5 |
| 5 TOOL_IO1 | 23 /EVTO | 40 GND |
| 6 TOOL_IO2 | 24 GND | 41 MDO6 |
| 7 /RESET | 25 /MSEO0 | 42 GND |
| 8 VREF | 26 VEN_IO | 43 MDO7 |
| 9 /EVTI | 27 MDO1 | 44 GND |
| 10 GND | 28 GND | 45 MDI2 |
| 11 /RSTI | 29 MDO2 | 46 GND |
| 12 GND | 30 GND | 47 MDI3 |
| 13 /MSEI | 31 MDO3 | 48 GND |
| 14 GND | 32 GND | 49 VEN_IO1 |
| 15 MDI0 | 33 MDI1 | 50 GND |
| 16 GND | 34 GND | 51 PORT0 |
| 17 MCKI | 35 /MSEO1 | |
| 18 GND | | |

**Pinbelegung für den NEXUS Adapter für die MPC56x Familie/Glenair51**

| | |
|---|---|
| 1 /RESET | 2 VREF |
| 3 /EVTI | 4 VALTREF |
| 5 /RSTI | 6 VENDOR_IO1 |
| 7 /MSEI | 8 GND |
| 9 MCKI | 10 GND |
| 11 MDI0 | 12 GND |
| 13 RES. | 14 GND |
| 15 RES. | 16 VENDOR_IO2 |
| 17 /EVTO | 18 GND |
| 19 /MSEO | 20 GND |
| 21 MCKO | 22 GND |
| 23 MDO0 | 24 GND |
| 25 MDO1 | 26 GND |
| 27 MDO2 | 28 GND |
| 29 MDO3 | 30 GND |
| 31 MDO4 | 32 GND |
| 33 MDO5 | 34 GND |
| 35 MDO6 | 36 GND |
| 37 MDO7 | 38 GND |
| 39 MDI1 | 40 GND |

**Pinbelegung für den NEXUS Adapter für die MPC56x Familie/ AMP40NS**

| | |
|---|---|
| 1 UBATT | 2 UBATT |
| 3 VSTBY | 4 TOOL_IO0 |
| 5 TOOL_IO1 | 6 TOOL_IO2 |
| 7 /RESET | 8 VREF |
| 9 /EVTI | 10 GND |
| 11 /RSTI | 12 GND |
| 13 /MSEI | 14 GND |
| 15 MDIO | 16 GND |
| 17 MCKI | 18 GND |
| 19 MDO0 | 20 GND |
| 21 MCKO | 22 GND |
| 23 /EVTO | 24 GND |
| 25 /MSEO0 | 26 VEN_IO0 |
| 27 MDO1 | 28 GND |
| 29 MDO2 | 30 GND |
| 31 MDO3 | 32 GND |
| 33 MDI1 | 34 GND |
| 35 /MSEO1 | 36 GND |
| 37 MDO4 | 38 GND |
| 39 MDO5 | 40 GND |
| 41 MDO6 | 42 GND |
| 43 MDO7 | 44 GND |
| 45 MDI2 | 46 GND |
| 47 MDI3 | 48 GND |
| 49 VEN_IO1 | 50 GND |

**Pinbelegung für den NEXUS Adapter für die MPC56x Familie/ AMP50**

### 3. Nexus Adapter Glenair51

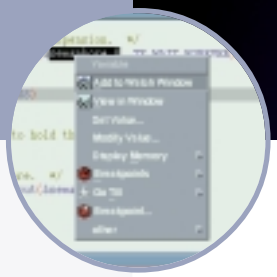
*Mit diesem Adapter bietet Lauterbach eine robuste Version des NEXUS Adapters. Durch die Verwendung eines Steckers der Firma Glenair, die auf elektrische Verbindungstechnik spezialisiert ist, kann mit diesem Adapter auch bei Vibrationen und Erschütterungen, wie sie beispielsweise bei einem Test im Auto auftreten, eine stabile Verbindung zum NEXUS Port garantiert werden.*

*Detailierte Informationen finden Sie unter.*

***http://www.lauterbach.com/nexlist.html***

*Bild 9 auf der linken Seite zeigt die Unterseiten der einzelnen NEXUS Adapter.*

*Im Frühjahr 2002 wird eine neue Version der NEXUS Adapter zur Verfügung stehen. Neu ist, daß jeder NEXUS Adapter mit einem 8-poligen Zusatzstecker ausgestattet wird, der 4 Eingangs- und 4 Ausgangskanäle zur Verfügung stellt.*



# Intelligente Trace-Auswertung für ARM-ETM



*Durch eine Integration der Embedded Trace Macrocell - kurz ETM - können ARM7/ARM9 basierte ASICs bzw. Prozessoren mit On-Chip Trace- und Triggermöglichkeiten ausgestattet werden. Die Ausgabe des Programm- und Datenflusses erfolgt über einen sogenannten Traceport. Typischerweise ist ein Traceport beim ARM7/ARM9 zwischen 8 und 20 Pins breit. Dabei sind 3 Pins fest für den Pipelinestatus reserviert. Über diese Pins wird pro Clockzyklus ausgegeben, was in der Execute Stufe der Pipeline passiert (beispielsweise Instruction executed, Branch executed, Wait etc.). Ein weiterer Pin wird jeweils für den Traceclock und ein Synchronisationssignal benötigt. Für die Ausgabe der eigentlichen Tracepakete sind dann zwischen 4 und 16 Pins vorgesehen. Mögliche Tracepakete sind:*

- *Sprungzieladressen indirekter Sprünge*
- *Adressen einzelner bzw. aller Datentransfers, sowie die Daten selbst*

*Da hier für jeden Clockzyklus Informationen aufgezeichnet werden, arbeiten Entwicklungswerkzeuge heute mit wesentlich größeren Tracespeichern. Lauterbach hat beispielsweise seinen neuen PowerTrace mit einem 16 MFrames großen Tracespeicher ausgerüstet. Was passiert nun, wenn mehr Daten anfallen, als über den Traceport übertragen werden können? Dies kann immer dann auftreten, wenn entweder viele indirekte Sprünge aufeinanderfolgen oder viele Datentransfers durchgeführt werden.*

*Auch für diesen Fall stellt die Embedded Trace Macrocell Mechanismen bereit. Während die Pipeline-Informationen immer clocksynchron ausgegeben werden, ist den Pins für die Ausgabe der Tracepakete ein FIFO-Speicher vorgeschaltet, indem bis zu 64 8-Bit Pakete zwischengespeichert werden können. Problematisch wird es erst, wenn auch der FIFO-Speicher nicht mehr alle ankommenden Tracepakete aufnehmen kann. Es kommt zu einem FIFO Überlauf und damit zur Gefahr des Verlustes wichtiger Tracedaten.*

*Grundsätzlich werden beim FIFO Überlauf keine Tracepakete mehr angenommen. Die ETM signalisiert über die Pipeline-Information den FIFO Überlauf und kann dafür sorgen, dass der FIFO zunächst vollständig entleert wird. Ist der FIFO wieder aufnahmebereit, wird ein Tracepaket ausgegeben, das die vollständige Adresse der gerade ausgeführten Instruktion enthält. Durch diese Information kann sich das Entwicklungswerkzeug, das die Tracepakete aufzeichnet, wieder aufsynchronisieren.*

*Die ETM stellt es dem Entwickler zudem frei, ob der Prozessor die Programmabarbeitung anhalten soll, wenn es zu einem FIFO-Überlauf gekommen ist. Damit wird ein Verlust von Traceinformation vermieden und man nimmt bewußt eine Verletzung des Echtzeitbetriebes während der Testphase in Kauf. Steht für den Entwickler jedoch der Echtzeitbetrieb im Vordergrund, wird er sich dafür entscheiden, dass der Prozessor auch im Falle eines FIFO Überlaufs weiterläuft. Dadurch gehen dann natürlich wichtige Traceinformationen verloren. Bild 10 zeigt eine Tracedarstellung, bei der es zu einem solchen Datenverlust gekommen ist.*

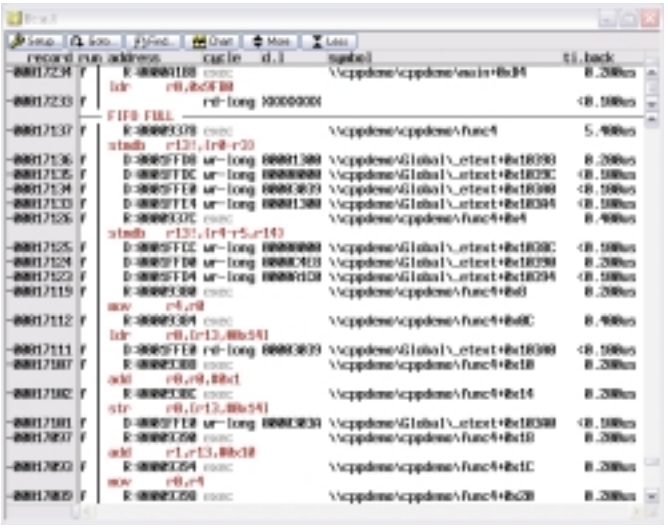
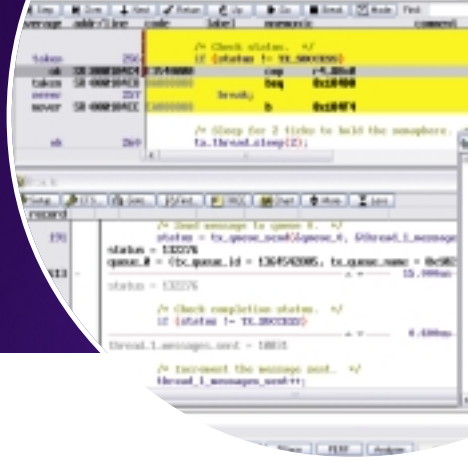



*Bild 10: Standard Tracedarstellung mit einer Lücke in der Aufzeichnung, die durch einen FIFO Overflow verursacht wurde.*

## SmartTrace

*Ein Tracedaten-Verlust kann unter Umständen mit dem von Lauterbach entwickelten SmartTrace ausgeglichen werden. Sein komplexer Algorithmus arbeitet mit folgenden Basis-Informationen:*

- *Adresse A der letzten Instruktion, die vor dem Tracedaten-Verlust ausgeführt wurde*



- *Adresse B der Instruktion, ab der die Tracedaten wieder korrekt ausgegeben werden*
- *Anzahl der Clockzyklen, die zwischen diesen beiden Instruktionen vergangen sind*
- *Maximale und minimale Anzahl von Clockzyklen, die von einem Befehl benötigt werden, beispielsweise minimale Anzahl 1 und maximale Anzahl 8*

*SmartTrace untersucht nun, ob es über direkte Sprünge einen eindeutigen Weg von Adresse A zur Adresse B gibt, der in der ermittelten Anzahl der Clockzyklen mit den verwendeten Instruktionen erreicht werden kann. Existiert ein eindeutiger Weg lassen sich die verlorenen Tracedaten so rekonstruieren. Eine solche Rekonstruktion zeigt Bild 11. Ist dies nicht der Fall, muß der eingesetzte Algorithmus erweitert werden.*

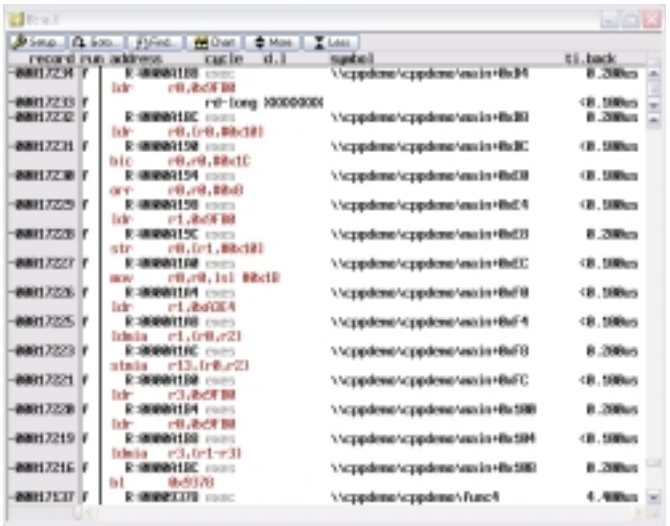


*Bild 11: Tracedarstellung nachdem die Lücke durch den SmartTrace Algorithmus rekonstruiert wurde. Die rekonstruierten Programmschritte sind mit exes gekennzeichnet (das s steht hier für synthetisch).*

## Context Tracking System

*Die durch einen FIFO-Überlauf entstandene Lücke in der Traceaufzeichnung läßt sich also nicht allein durch die Analyse der direkten Sprünge füllen. Zur Rekonstruktion des Programmflusses müssen auch die indirekten Sprünge herangezogen werden.*

*In diesem Artikel werden unter dem Begriff "indirekter Sprung" alle Programmverzweigungen zusammengefaßt, bei denen erst zur Laufzeit feststeht, an welcher Adresse die Programmabarbeitung fortgesetzt wird. Die Sprungzieladresse eines indirekten Sprunges wird, sofern es sich nicht um eine Exception handelt i.a. aus einem Register, vom Stack oder von einer Speicherzelle in den Program Counter geladen. Hier setzt nun der Context Tracking System Algorithmus - kurz CTS - an. Auch dabei handelt es sich um ein von Lauterbach entwickeltes Verfahren.*

*Die Grundidee von CTS ist, dass basierend auf den Informationen aus dem Tracespeicher der Kontext des Zielsystems für jeden im Tracespeicher aufgezeichneten Eintrag noch einmal hergestellt wird. Kontext meint hier die Register- und Speicherinhalte. Wie arbeitet CTS? Hauptinstrument ist ein Instruction Set Simulator, der die im Tracespeicher aufgezeichneten Programmschritte noch einmal abarbeitet. Bild 12 zeigt die Arbeitsweise von CTS.*

*Auf diese Weise lassen sich natürlich auch die Register-, Speicher- und Stackinhalte für die Instruktion noch einmal herstellen, die vor dem Tracedaten-Verlust ausgeführt wurde. Läuft der Instruction Set Simulator nun in die Tracelücke hinein und kommt zu einem indirekten Sprung bei dem der Program Counter aus einem Register, vom Stack oder aus einer Speicherzelle geladen wird, kann der Programmfluß*

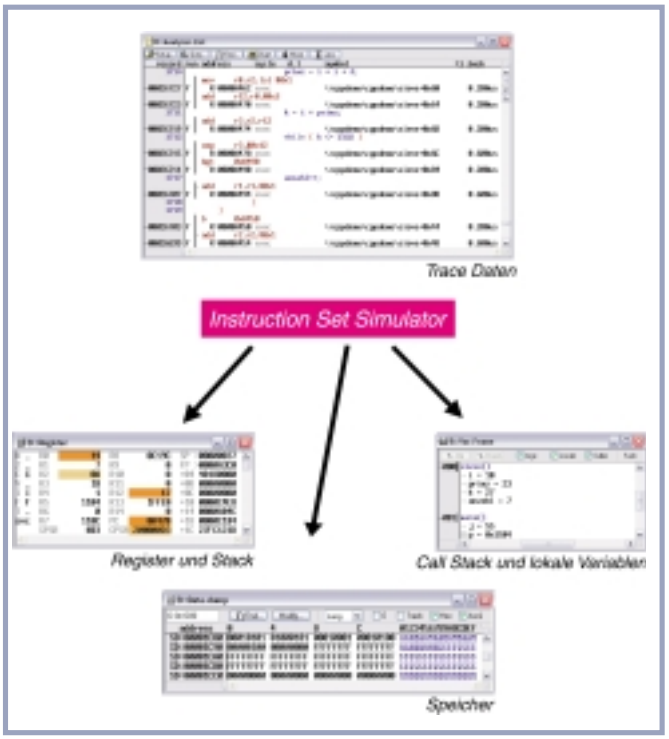
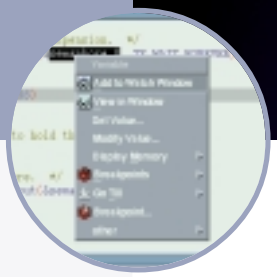



*Bild 12: CTS stellt mit Hilfe des Instruction Set Simulators die Inhalte von Registern, Speicher und Stack für jeden Eintrag im Tracespeicher noch einmal her.*





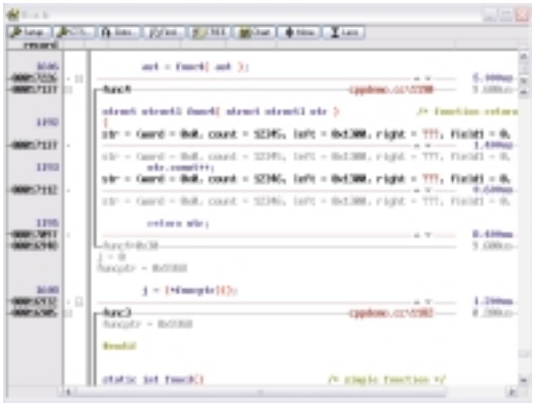


*Bild 13: Auch indirekte Sprünge innerhalb einer Tracelücke können rekonstruiert werden.*

*auch für diese Tracelücke rekonstruiert werden. Schrittweise versuchen so CTS und SmartTrace gemeinsam alle Lücken zu schließen. Ist dies möglich, kann das Entwicklungswerkzeug einen vollständigen Hochsprachen-Trace darstellen. Von besonderem Vorteil ist hier zudem, dass auch alle Register- und Stackvariablen im Tracespeicher sichtbar sind. Eine solche Tracedarstellung zeigt Bild 13.*

*Dieses Verfahren stößt natürlich auch an Grenzen. Es ist beispielsweise nicht möglich in einer Tracelücke auftretende Exceptions zu rekonstruieren. Generell läßt sich aber sagen, dass der Traceinhalt auch bei relativ vielen Aufzeichnungslücken meist vollständig wieder hergestellt werden kann. Die Funktion des Tracespeichers als wichtiges Instrument zur effizienten Fehlersuche bleibt also auch bei einer lückenhaften Aufzeichnung erhalten.*

## Software Lizenzschlüssel für TRACE32

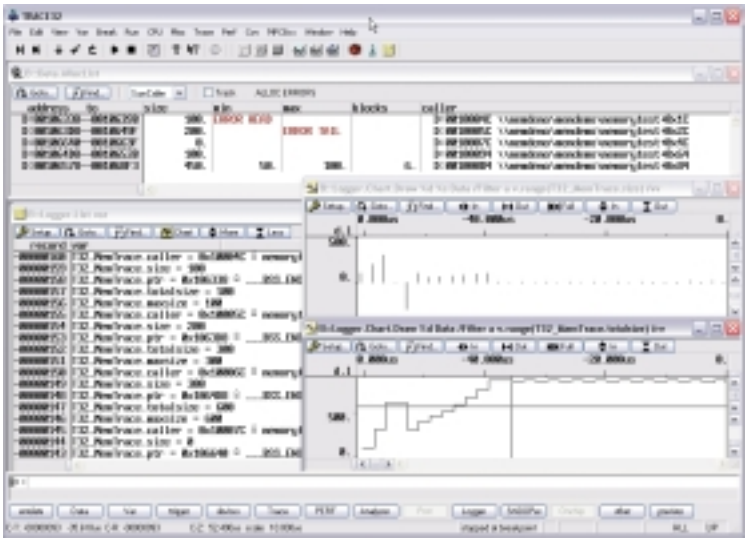
*Ab 2. April 2002 kann eine neue Version der TRACE32-Software nur noch dann installiert werden, wenn:*

- *Für das Entwicklungssystem noch die Software-Garantie gilt. Eine Software-Garantie besteht bis 1 Jahr nach dem Kauf des Systems.*
- *Für das Entwicklungssystem ein gültiger Software-Wartungsvertrag besteht.*

*Beim Abschluß eines Software-Wartungsvertrags erhalten Sie einen neuen Software-Lizenzschlüssel, der eine uneingeschränkte Installation von neuen Updates der TRACE32-Software während der Laufzeit des Wartungsvertrags erlaubt. Die aktuellste Software kann dann jederzeit vom Lauterbach Update-Server* **http://updates.lauterbach.com** *geladen werden.*

*Informationen zur Verwendung des Software-Lizenzschlüssels finden Sie unter:* **http://www.lauterbach.com/license.html**. *Dort steht ein PDF Dokument zur Verfügung, das alle Details beschreibt.*

## Memory Allocation Analyse



*TRACE32-PowerView bietet nun auch Kommandos zur Analyse der dynamischen Speicher-Allokierung.*

*Folgende Funktionalität ist nun möglich:*

- *Auflisten aller allokierten Speicherblöcke inklusiv eventueller Schreibzugriffe außerhalb der Blockgrenzen (ERROR HEAD, ERROR TAIL)*
- *Programmabbruch bei Schreibzugriffen außerhalb der Blockgrenzen*
- *Auflistung aller malloc/free Aufrufe aus dem Tracespeicher*
- *Graphische Analysen des verwendeten Heap-Speichers sowie der Speicherallokierungen*

# Debuggen auf System- und User-Ebene

**Die Komplexität der Prozessoren und deren MMUs nimmt beständig zu. Alle modernen Mikrokontroller bieten inzwischen ein umfangreiches Speichermanagement, so zum Beispiel die Prozessoren der Familien ARM und PowerPC. Multiprozess- und HA-Systeme nutzen deren Möglichkeiten bis zum letzten Bit - eine Herausforderung für Hardware-Debugging Systeme. Die TRACE32-Produkte von Lauterbach bieten eine volle MMU-Unterstützung. In Verbindung mit OS-Awareness ist ein Debuggen sowohl auf Systemebene als auch auf der eingeschränkten User-Ebene möglich. Dabei kann der Anwender stets auf den kompletten Speicher und auf alle Prozesse zugreifen - inklusive aller verfügbaren Symbol- und Hochspracheninformation.**

*„Wo ist mein Programm“ fragen sich viele Software-Entwickler, die mit einem JTAG-Debugger oder einem In-Circuit Emulator zwar alles Drumherum sehen, wie den OS Kernel (z.B. Linux), die Interrupt-Routinen oder die Device-Treiber, aber eben nicht ihren User Prozess finden (und debuggen) können. Denn die Betriebssysteme für Embedded Systeme holen auf: Memory Management wird auch in diesem Bereich immer mehr ein Thema, nicht zuletzt durch den andauernden Siegeszug von embedded Linux.*

*„Klassische“ Applikationen nutzen einen linearen Adressraum ohne irgendwelche Adressumsetzung (siehe Bild 14). Einfache (billige) Prozessoren, kleine Betriebssysteme und wenig Speicherbedarf sind unschlagbare Argumente für lineare Systeme. Sie werden in der Regel komplett mit Betriebssystem und allen Tasks erstellt, gelinkt und als Komplett-Paket ins ROM oder Flash des Gerätes gebrannt. Hardware-Debugger haben hier ein leichtes Spiel. Die erzeugte Applikation enthält alle Debug-Informationen, die ein Debugger braucht, wie z.B. die Adressen der Tasks, OS-Routinen usw. Die Applikation arbeitet mit absoluten Adressen, wodurch eine Adress-Umrechnung nicht gebraucht wird. Allerdings haben diese Systeme – wenn überhaupt – nur eingeschränkte Schutzmechanismen und es mangelt an einem Mindestmaß an Flexibilität. Gerade diese Eigenschaften werden aber mehr und mehr von einem Betriebssystem gefordert. Hochverfügbarkeit (High Availability, HA) ist hier zum Schlagwort geworden.*


**Bild 14: Linearer Adressraum**

*Zum einen will man seine Software gegen Fehler durch Fehlertoleranz schützen. Dies erreicht man durch Schutzmechanismen (Memory Protection), wie z.B. Schreibschutz auf bestimmte Speicherbereiche. Versucht ein Prozess auf eine geschützte Speicherstelle zu schreiben, so wird ein Fehler signalisiert und das Betriebssystem oder die Applikation kann dementsprechend reagieren (z.B. durch Restart des Prozesses). Bei komplexeren MMU- (Memory Management Unit) Systemen hat jeder Prozess seinen eigenen Adressraum, der i.a. jeweils auf der gleichen virtuellen Adresse liegt. Während ein User-Prozess läuft, „sieht“ der Prozessor nur den Adressbereich dieses Prozesses und den des Betriebssystems (Bild 15). Ein Überschreiben eines anderen Prozesses ist gar nicht möglich, da dieser Bereich zu diesem Zeitpunkt überhaupt nicht adressierbar ist. Der Scheduler im Betriebssystem hat die Aufgabe, beim Prozesswechsel die MMU so umzuprogrammieren, dass der jeweilige Prozess gerade im Adressraum des Prozessors liegt.*

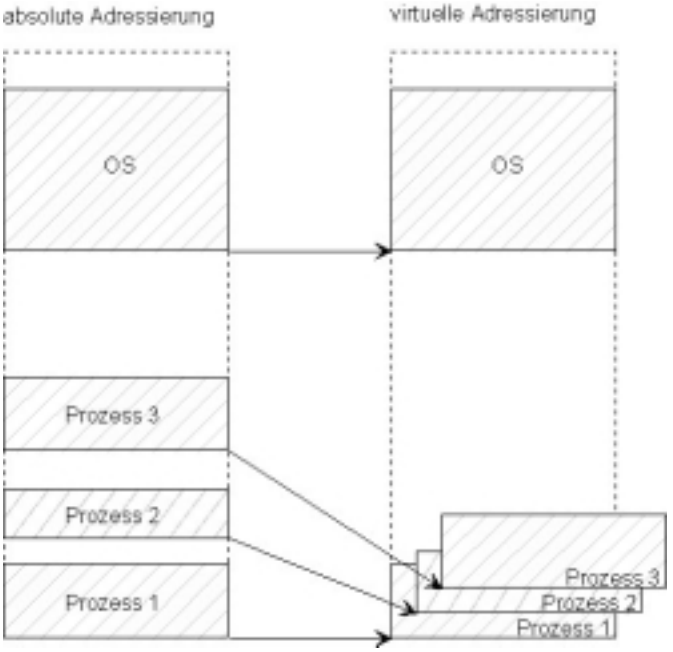
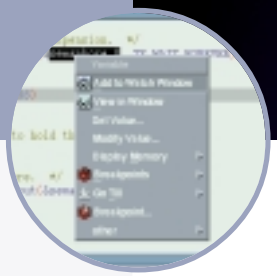



**Bild 15: Virtueller Adressraum**

*Eine andere Eigenschaft der MMU basierten Systeme ist die dynamische Nachladbarkeit von Prozessen. Da die Prozesse nicht mehr auf physikalischen, hardwareabhängigen Adressen gelinkt werden müssen, können sie alle auf eine*





*logische Adresse gelegt werden. Das Betriebssystem lädt den Prozess dann in einen gerade freien Speicherbereich des Zielsystems und programmiert die MMU dann so, dass der logische Adressraum dieses Prozesses auf den physikalischen Adressbereich umgesetzt wird.*

*Spätestens jetzt fangen die Probleme für Hardware-Debugger an. Diese heißen so, weil sie normalerweise auf der Hardware-Ebene arbeiten, und dabei absolute Adressen nutzen. Das muss auch so sein, denn man erwartet von einem guten Debugger, dass man den gesamten Speicher sehen und debuggen kann, und nicht nur den kleinen Bereich, den momentan der Prozessor zu Gesicht bekommt. Die Software arbeitet jedoch komplett mit virtuellen (= logischen) Adressen (bis auf den meistens in Assembler geschriebenen Scheduler). D.h. auch jegliche Symbol-Informationen in den Applikations-Dateien zeigen auf virtuelle Adressen. Um nun einerseits auf den gesamten physikalischen Adressraum zugreifen zu können, und damit auch auf alle im System vorhandenen Prozesse, muss der Debugger mit absoluten Adressen arbeiten. Andererseits ist die für Debugger essentielle Symbolinformation nur als virtuelle Adressen vorhanden.*

*Der Debugger benötigt somit eine eigene Umrechnungsmöglichkeit von absoluten in virtuelle Adressen, und umgekehrt. Eine einfach 1:1 Abbildung reicht dabei nicht aus, da mehrere Prozesse ja gleiche virtuelle, aber physikalisch unterschiedliche Adressräume haben können (siehe Bild 15). Der Debugger muss daher die Prozesszugehörigkeit in die Umrechnung mit einbeziehen.*

*Es gibt nun prinzipiell drei Möglichkeiten, dem Debugger diese Adress-Umrechnung bekannt zu geben. Zum ersten kann man per Hand (oder eher: per Skript) die einzelnen Bereiche angeben. Vorteil: Der Debugger weiß bereits vor dem Starten des Systems/der Prozesse, wo er den entsprechenden Bereich findet, und man kann vor Starten des Programms schon Watchpoints o.ä. setzen. Diese Methode funktioniert allerdings nur, wenn eine statische Nutzung der MMU vorliegt, wenn also*

- *die Adressumsetzung während des Programmlaufs nicht geändert wird (z.B. beim Laden eines Prozesses),*
- *das Betriebssystem immer die gleiche Adressumsetzung nutzt und*
- *diese Umsetzung dem Anwender bekannt ist.*

*Diese Bedingungen treffen allerdings bei den wenigsten embedded Betriebssystemen zu.*

*Die zweite Möglichkeit, die der Debugger besitzt, um die Adressumsetzung zu rekonstruieren, ist das Lesen der MMU selbst. Natürlich enthält die MMU des Prozessors alle Informationen, die für eine Adressumrechnung im momentanen Betrieb nötig sind. Der Debugger kann diese Informationen auslesen und zurückrechnen. Es ergibt sich schließlich eine akkurate Rekonstruktion einer Momentaufnahme. Interessant ist dies freilich nur, wenn die MMU (und die daraus resultierenden Tabellen) die vollständige Information enthalten. Bei „höheren" Betriebssystemen ist dies meistens nicht der Fall. Dort „sieht" selbst die MMU nur den jeweiligen Prozess. Beim Prozesswechsel wird dann die komplette MMU umprogrammiert. U.U. ist es bei*

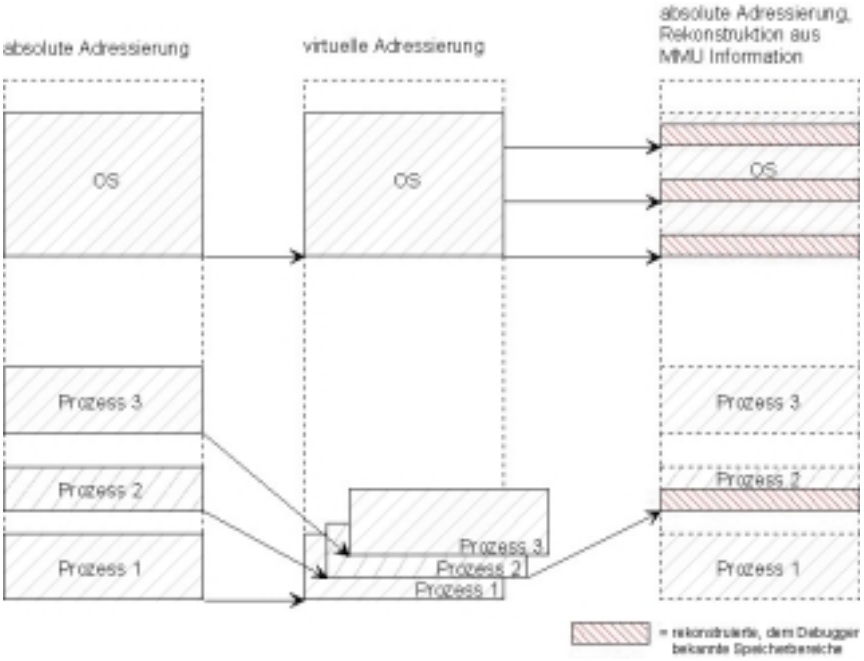



*Bild 16: Rekonstruktion der Adressumrechung aus der MMU-Information*

# TRACE32-POWERVIEW

*komplexeren MMUs (z.B. PowerPC 603e) sogar so, dass nur vereinzelte Pages („Page Table Entries") aus der MMU nachvollziehbar sind. Selbst wenn der Debugger alle Informationen, die der Prozess, die MMU und die angehängten Tabellen preisgeben, auswertet, kann er nur ein fragmentiertes Abbild des absoluten Adressraumes rekonstruieren. (Bild 16). Die komplette Information über die Adressberechnungen der Prozesse ist hier ausschließlich dem Betriebssystem bekannt.*

*Dies führt direkt zu der dritten Methode: der Debugger nutzt direkt Informationen aus dem Betriebssystem, um die Übersetzungs-Tabelle zu rekonstruieren. Da das Betriebssystem als einzige Instanz ja die komplette Information zur Verfügung haben muss, enthält es Tabellen und Variablen, welche die komplette Speichernutzung beschreiben. Was liegt näher, als diese Information zu nutzen? Mit Hilfe dieser Tabellen kann der Debugger eine vollständige Rekonstruktion der Adressumsetzung berechnen (Bild 17). Nur in diesem Fall ist es möglich, zu jedem beliebigen Zeitpunkt im Debugger auch jeden beliebigen Speicherbereich zu sehen und zu manipulieren. Selbst wenn die Applikation im Prozess 1 angehalten wurde, und der Prozessor daher nur den Adressraum des Prozesses 1 sieht, kann der Debugger auch auf andere Prozesse zugreifen, und z.B. einen Breakpoint auf eine Funktion des Prozesses 2 setzen.*

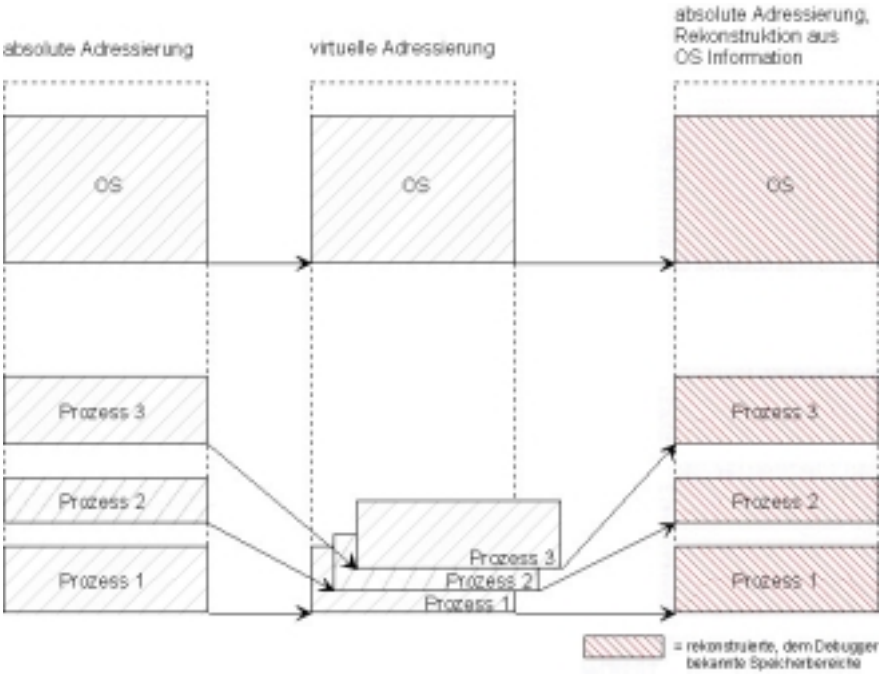



*Bild 17: Rekonstruktion der Adressumrechung aus der OS-Information*

*Um diese Funktionalität zu bieten, muss der Debugger tiefgreifende Internas des Betriebssystems kennen und auswerten können. Dazu zählt neben der MMU-Tabelle auch die Prozess-Verwaltung, die über den Scheduler eng mit der MMU verknüpft ist. Leider gibt es kein einheitliches Format, weder für die MMU Tabellen, noch für die Prozessverwaltung. Jeder Hersteller vertraut hier auf sein eigenes „Süppchen". Der Debugger muss also an das jeweilige Betriebssystem angepasst werden; er muss eine „OS Awareness" besitzen.*

*Der Debugger TRACE32 von Lauterbach enthält eine konfigurierbare OS Awareness, die an jedes Betriebssystem adaptiert werden kann. Damit bietet der TRACE32 nicht nur eine für jeden Prozessor angepasste vollständige MMU-Unterstützung, sondern auch eine intensive OS-Awareness, die oben beschriebenes Debugging ermöglicht. Lauterbach bietet fertige Lösungen für die Betriebssysteme ChorusOS (Sun Microsystems, auf Power PC), EPOC (Symbian, auf ARM), LynxOS (LynuxWorks, auf PowerPC) und natürlich Linux (diverse, auf PowerPC) an.*

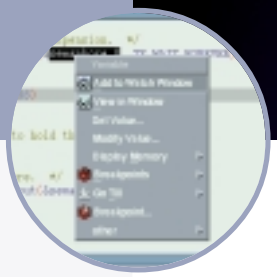
## Neu unterstützte Echtzeitkerne

- ***LynxOS von LynuxWorks Inc.***
  *mit MMU Unterstützung für MPC860 und MPC8260*
- ***QNX von QNX Software Systems Ltd.***
  *mit MMU Unterstützung für SH4*
- ***µC/OS-II von Micriµm Inc.***
  *für C166, TriCore, HC08, ColdFire*
- ***embOS von Segger GmbH***
  *für ARM*
- ***OSEK/ORTI V2.1***

*Eine Auflistung aller von TRACE32-PowerView unterstützten Echtzeitkerne finden Sie unter:*

***http://www.lauterbach.com/xlist.html***

TRACE32 PowerView
die offene und intuitive Benutzeroberfläche



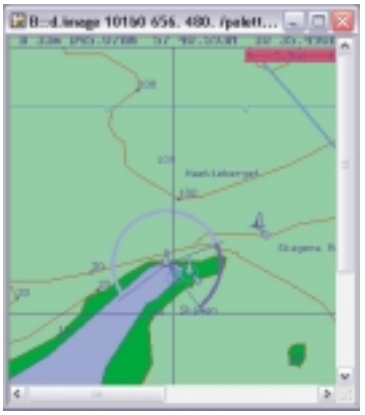
Weitere Informationen im Internet: www.lauterbach.com

## Neue Debugger TRACE32-ICD

| | |
|---|---|
| **Hitachi** | **SH4** |
| **Infineon** | **C166S V2** |
| **Intel** | **XScale** |
| **MIPS** | **MIPS32**<br>**MIPS64** |

*Mit TRACE32-ICD für den MIPS64 entwickelte Lauterbach erstmals einen 64-Bit Debugger.*

## Darstellung von Bitmap-Images



*TRACE32-Power-View ermöglicht es nun, sich Bitmap-Images aus dem Zielsystemspeicher anzuschauen. Dabei werden verschiedene Bitmap Formate unterstützt. Zusammen mit einem programmierbaren Dialog-Fenster können graphische Schnittstellen auf dem echten Prozessor simuliert und getestet werden, bevor die eigentliche Anzeige-Hardware verfügbar ist.*

## TRACE32-FIRE für C166S V2

*Nachdem bereits seit Mitte 2001 der Debugger für den C166S V2 ausgeliefert wird, stellt Lauterbach zur Embedded Systems 2002 in Nürnberg zusätzlich seinen TRACE32-FIRE In-Circuit Emulator für diese Prozessorfamilie vor. Die Basis für den Emulator bilden die entsprechenden Bondout Prozessoren in Flip-Chip Technologie.*

*Alle Derivate des C166S V2 können mit 0 Waitstates bis zu einer Frequenz von 40 MHz emuliert werden. Ein eigenes Triggermodul unterstützt das Setzen von Breakpoints und von komplexen Triggerpunkten auch auf alle On-Chip Speicher, sowie eine umfassende Code-Abdeckungsanalyse. Eine Liste aller von TRACE32-FIRE unterstützten Prozessoren finden Sie unter:*
**http://www.lauterbach.com/fire.html**

*Bitte schicken Sie mir Informationsmaterial zu:*

*Wir setzen folgende Prozessoren ein:*

_____

*Ich interessiere mich für folgende Entwicklungswerkzeuge:*

- ☐ ***TRACE32-ICD***
- ☐ ***TRACE32-PowerTools***
- ☐ ***TRACE32-PowerProbe***
- ☐ ***TRACE32-FIRE***

***Absender:***

***Name*** _____

***Firma*** _____

***Adresse*** _____

_____

***Telefon*** _____

***Email*** _____

**LAUTERBACH**
***FAX: ++49 8104 8943-30***