NEWS 2010

LAUTERBACH
DEVELOPMENT TOOLS

DEBUGGER, REAL-TIME TRACE, LOGIC ANALYZER

Stephan Lauterbach während einer Podiumsdiskussion auf der IP/ESC in Grenoble

Erfolgreich im Gespräch

Der Schlüssel zu unserem Erfolg ist die gute Kommunikation mit unseren Kunden und Partnern. Geeignete Entwicklungswerkzeuge können wir deshalb rechtzeitig bereitstellen, weil wir über technische Neuerungen unserer Partner und über den Bedarf unserer Kunden gut informiert sind. Bei den TRACE32-Innovationen stehen die Wünsche unserer Kunden an erster Stelle. Gute Kontakte zu allen Partnern aus der Embedded Industrie bilden dafür den Ausgangspunkt. Erfolgreiche Kommunikation erfordert Engagement, dazu stehen wir auch in 2010.

Expertenforum

Um den Kontakt zu unseren Kunden zu intensivieren, haben wir im Oktober 2009 erstmals ein TRACE32-Expertenforum an unserem Firmensitz in Höhenkirchen-Siegertsbrunn veranstaltet. Grundidee dieses Forums ist es, unsere Kunden mit Experten aus der TRACE32-Entwicklung zusammenzubringen. Einen ausführlichen Bericht zu diesem Event und unsere Pläne für 2010 finden Sie auf der letzten Seite dieses Newsletters.

Standardisierungsgremien

Neben dem direkten Gespräch mit unseren Partnern aus der Embedded Industrie ist die Mitarbeit in Standardisierungsgremien eine zunehmend wichtiger werdende Form des Austauschs und der Beziehungspflege. Die Ergebnisse aus diesen Gremien fließen bereits seit Jahren in unsere Produkte ein. Unser Newsletter möchte Sie auch über dieses Engagement ausführlich informieren.

Podiumsdiskussionen

Podiumsdiskussionen auf Fachmessen erlauben es ebenfalls aktuelle und zukünftige Marktanforderungen mit unseren Kunden und Partnern zu erörtern. Im November 2009 beteiligte sich Stephan Lauterbach beispielsweise im Rahmen der IP/ESC in Grenoble (Frankreich) an einer solchen Gesprächsrunde. Auf Initiative der Firma ARM wurde eingehend über die Zukunft der Debug- und Tracetechnologie diskutiert.

Wir freuen uns schon auf die embedded world 2010 in Nürnberg, um mit Ihnen ins Gespräch zu kommen. Besuchen Sie uns auf unserem Stand – Halle 10.0 Stand 325!

INHALT

Neuer TRACE32-Installer für Windows 7

Mit der neuen TRACE32-DVD vom Dezember 2009 (Build 20817/Release) führt Lauterbach den offiziellen Support für Windows 7 ein.

Der TRACE32-Installer wurde erweitert, um die TRACE32 USB-Treiberinstallation unter Windows 7 zu vereinfachen. Dies war notwendig, da Windows 7 Treiber automatisch nur noch vom *Windows Update Server* bzw. aus einem vorinstallierten *Driver Package* installiert.

Sollten Sie keine neue DVD zur Hand haben, können Sie den neuen *TRACE32 USB driver installer for Windows XP/Vista/7 (32-bit and 64-bit)* unter folgendem Link herunterladen:

http://www.lauterbach.com/faq/t32usb_setup.exe

Handhabung und Look-and-Feel von TRACE32 bleiben auch unter Windows 7 erhalten. Um das Sicherheitsmodell von Windows noch besser zu unterstützen und automatisierte Installationen zu erleichtern, wurden die ausführbaren Dateien auf der DVD vom Dezember 2009 erstmals signiert. Das Signieren des USB-Treibers erfolgte bereits 2007.

Programmierung von Seriellen FLASH-Bausteinen

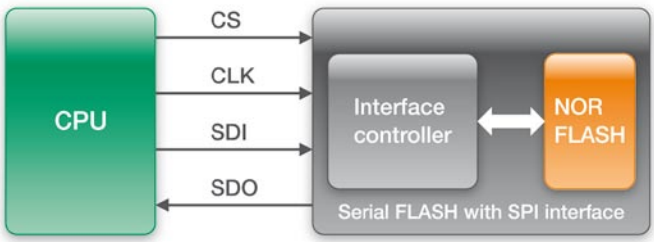

Bild 1: NOR-FLASH mit vorgeschaltetem *Interface Controller*

Embedded Designs sind zunehmend mit seriellen FLASH-Bausteinen ausgestattet. Lauterbach hat frühzeitig auf diesen Trend reagiert. Seit Mitte 2009 wird das Programmieren serieller FLASH-Bausteine mit den TRACE32-Debuggern unterstützt. Im Jahr 2010 soll dieser Support nochmals erheblich ausgebaut werden.

Geringe Pinzahl, kleine Bauform und reduzierter Stromverbrauch machen serielle FLASH-Bausteine zu einer kostengünstigen Alternative zu NOR/NAND-FLASH-Bausteinen. Die zugrunde liegende Idee ist eigentlich ganz einfach: Dem NOR/NAND-FLASH wird ein *Interface Controller* vorgeschaltet, der es erlaubt, den Baustein über einen SPI- bzw. MMC-Bus zu programmieren und auszulesen (siehe Bild 1).

Die TRACE32-Unterstützung für serielle FLASH-Bausteine umfasst das Programmieren und das Auslesen des Inhalts. Die Darstellung des FLASH-Inhalts als klassischer Hex-Dump erlaubt eine schnelle Überprüfung der programmierten Daten (siehe Bild 2).

Um herauszufinden, ob TRACE32 Ihr serielles FLASH bereits unterstützt, überprüfen Sie bitte die beiden folgenden Listen auf unserer Web-Seite:

Supported NAND/SERIAL FLASH Controller
http://www.lauterbach.com/ylistnand.html

Supported NOR/NAND/SERIAL FLASH Devices
http://www.lauterbach.com/ylist.html

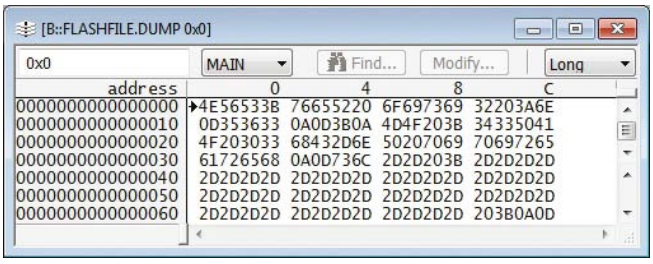

Bild 2: Darstellung des Inhalts eines seriellen FLASH

Debugger für die Intel® Atom™ Prozessor-Familie

Seit Oktober 2009 bietet Lauterbach auch Entwicklungswerkzeuge für die Intel® Atom™ Prozessoren an. Linux-Debugging wird bereits in vollem Umfang unterstützt, Windows CE ist für Anfang 2010 geplant.

Unterstützte Derivate		
Intel®	**LA-3776 (Atom)**	
	• 230	• N270
	• 330	• N280
	• D410	• N450
	• D510	• Z5XX
	weitere Derivate sind geplant	

Differentielles Laden

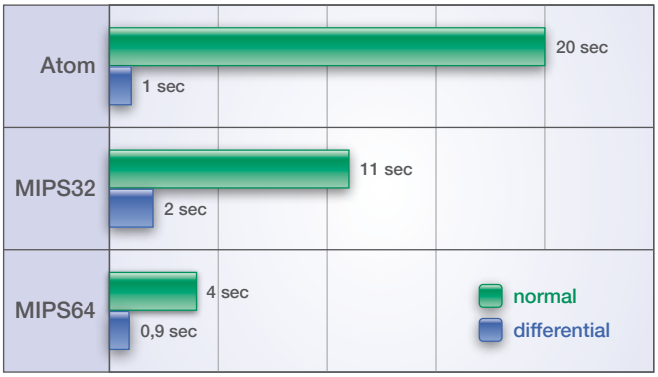

Bild 3: Das Laden einer 4 MByte großen Datei lässt sich durch differentielles Laden erheblich beschleunigen.

Ein typischer Debug-Zyklus umfasst folgende Schritte: Programm debuggen – Fehler finden – Fehler beheben – Programm kompilieren – Programm neu laden. Selbstverständlich soll jeder einzelne Schritt schnell und ohne große Wartezeiten durchführbar sein.

Längere Wartezeiten können jedoch dann entstehen, wenn sehr große Programme über eine langsame JTAG-Schnittstelle in das RAM des Zielsystems heruntergeladen werden müssen. Mittels des so genannten differentiellen Ladens kann hier Abhilfe geschaffen werden. Die Ladebeschleunigung ist dann am größten, wenn sich das neu kompilierte Programm vom alten, bereits geladenen nur geringfügig unterscheidet.

Die Grundidee beim differentiellen Laden ist, dass der Debugger eine Kopie des bereits geladenen Programms vorhält. Beim Laden des Neukompilats wird zunächst eine Differenztabelle gebildet. Die Differenztabelle enthält komprimiert alle Informationen, die notwendig sind, um das alte, bereits geladene Programm in das neu kompilierte überzuführen.

Der Debugger lädt nun nur noch die Differenztabelle ins Zielsystem herunter. Da die Differenztabelle in der Regel 30- bis 100-mal kleiner ist, als das neu zu ladende Programm, reduziert sich die Datenmenge, die über die JTAG-Schnittstelle übertragen werden muss, erheblich. Ein *Load Agent* auf dem Zielsystem dekomprimiert dann die Differenztabelle und sorgt dafür, dass am Ende das Neukompilat im Speicher steht.

Die Messungen in Bild 3 wurden für eine Testdatei durchgeführt, bei der sich 1% des Programms geändert hat.

1. Atom Architektur
CPU: Z530P von Intel
Prozessorfrequenz: 1,6 GHz
JTAG-Frequenz: 20 MHz
Normaler Download: 204 KByte/s

2. MIPS32 Architektur
CPU: BCM7325 von Broadcom
Prozessorfrequenz: 167 MHz
JTAG-Frequenz: 20 MHz
Normaler Download im Turbo Mode: 370 KByte/s

3. MIPS64 Architektur
CPU: OCTEON Plus CN58XX von Cavium
Prozessorfrequenz: 950 MHz
JTAG-Frequenz: 50 MHz
Normaler Download im Turbo Mode: 1 MByte/s

Neu unterstützte Prozessoren

Neue Derivate	
ARC	**LA-3750 (ARC)** • ARC 601/ARC 630
ARM	**LA-7843 (Cortex-A/R)** • Cortex-A5/Cortex-A5MPCore • Cortex-A9/Cortex-A9MPCore **LA-7844 (Cortex-M)** • Cortex-M0/Cortex-M1
ATMEL	**LA-3779 (AVR32)** • AVR32 (Q2/2010) **LA-7844 (Cortex-M)** • AT91SAM3U
Broadcom	**LA-7760 (MIPS32)** • BCM3380 • BCM56xxx/5836 • BCM6362/6368/6550 • BCM7401/7402/7403
CEVA	**LA-3711 (CEVA-X)** • CEVA-X1641 • CEVA-XC (Q2/2010) **LA-3774 (TeakLite-III)** • TeakLite-III
Cavium	**LA-7761 (MIPS64)** • Octeon CN54xx/CN56xx • Octeon CN63xx **LA-7765 (ARM11)** • ECONA CNS3XXX
Cortus	**LA-3778 (APS)** • APS-IP (Q2/2010)
Energy Micro	**LA-7844 (Cortex-M)** • EFM32
Freescale	**LA-7736 (MCS12X)** • MC9S12G **LA-7742 (ARM9)** • i.MX23/i.MX25 **LA-7732 (ColdFire)** • V1 ColdFire Core **LA-7753 (MPC55xx)/ LA-7630 (NEXUS MPC55xx)** • MPC5643L **LA-7764 (PowerQUICC III)** • QorIQ P1013/P1022/P4080
Infineon	**LA-7756 (TriCore)** • TC1167/TC1197/TC1337 • TC1367/TC1387/TC1387ED • TC1782/TC1782ED
Infineon (Forts.)	**LA-7759 (XC2000/C166S V2)** • XC2000ED • XC2200/XC2300 Family • XC2700/XE166 Family • XGOLD110
LSI	**LA-7834 (StarCore)** • StarPro25xx/26xx
Marvell	**LA-7742 (ARM9)** • 88AP128/162/166/168 • MV76100/78100/78200 **LA-7765 (ARM11)** • 88SV581X-V6 **LA-7843 (Cortex-A/R)** • 88SV581X-V7 **LA-7762 (XScale)** • PXA93x/PXA950
MIPS	**LA-7760 (MIPS32)** • MIPS32 1004K/1004KF • MIPS32 1004K CPS • MIPS32 M14K/M14Kc
NEC	**LA-3777 (78K0R)** • 78K0R/Fx3, 78K0R/Kx3 **LA-7835 (V850)** • V850E2/Px4 • 70F3502/70F3504/70F3506
NXP	**LA-7844 (Cortex-M)** • LPC13xx **LA-7742 (ARM9)** • LPC29xx
ST Microelectronics	**LA-7753 (MPC55xx)/ LA-7630 (NEXUS MPC55xx)** • SPC56EL60 **LA-7844 (Cortex-M)** • STM32F105/STM32F107
Tensilica	**LA-3760 (Xtensa)** • Xtensa 8
Texas Instruments	**LA-7847 (TMS320C28X)** • TMS320F28232 • TMS320F28234/F28235 **LA-7838 (TMS320C6400)** • TMS320TC6424 • TMS320TCI6482/I6488 **LA-7843 (Cortex-A/R)/ LA-7838 (TMS320C6400)** • AM3505/AM3517 (Sitara) • OMAP4430/OMAP4440 **LA-7742 (ARM9)/ LA-7841 (TMS320C6700)** • OMAP-L137/OMAP-L138

Debugging von AMP- und SMP-Systemen

Viele Multicore-Prozessoren können als AMP- oder SMP-Systeme betrieben werden. Angepasst an die Betriebsart kommen unterschiedliche Debug- und Trace-Konzepte zur Anwendung. Am Beispiel des ARM Cortex-A9 MPCore werden diese TRACE32-Konzepte vorgestellt.

Debug-Konzepte

Im Gespräch mit Kunden stellen wir immer wieder fest, dass es für die beiden Begriffe

- **AMP – Asymmetrisches Multiprocessing**
- **SMP – Symmetrisches Multiprocessing**

recht unterschiedliche Interpretationen gibt. Deshalb beschreiben wir im Folgenden, wie die Firma Lauterbach diese Begriffe verwendet und welche Auswirkungen unsere Sichtweise auf die Konfiguration und die Bedienung des TRACE32-Debuggers hat.

Wie der Begriff „Multiprocessing“ schon vermuten lässt, arbeiten mehrere Cores in einem Embedded System zusammen. Entscheidend für das Debugging ist die Frage, wie die Systemaufgaben auf die einzelnen Cores verteilt werden.

Debug-Konzept für AMP-Systeme

In AMP-Systemen werden jedem Core bestimmte Aufgaben fest zugeteilt. Die Entscheidung über die Zuteilung wird in der Designphase des Systems getroffen. Dabei werden neben Cores für allgemeine Aufgaben oft zusätzlich Spezialcores ausgewählt, die für bestimmte Funktionen optimiert sind, z.B. DSPs.

Beim Debugging von AMP-Systemen wird für jeden Core eine eigene TRACE32-Instanz gestartet (siehe Bild 4). Dies hat zwei Gründe:

1. Ein AMP-System kann unterschiedliche Core-Architekturen enthalten.
2. Jeder Core arbeitet einen separaten Teil der Applikation ab. Das heißt, die Symbol- und Debug-Informationen sind größtenteils exklusiv dem jeweiligen Core zugeordnet.

Da die Cores jedoch nicht unabhängig voneinander arbeiten, sondern parallel und gemeinsam alle Applikationsaufgaben lösen, muss es möglich sein, alle Cores synchron zu starten und zu stoppen. Nur so lässt sich das Zusammenspiel der Cores und damit die Gesamtapplikation umfassend testen.

Das synchrone Starten und Stoppen aller Cores kann unterschiedlich realisiert sein. Idealerweise unterstützt der Multicore-Prozessor dies durch eine interne Synchronisierungslogik. Fehlt diese, übernimmt TRACE32 die Synchronisation. Ein spezieller Algorithmus berechnet dabei JTAG-Sequenzen, um alle Cores möglichst zeitnah zu steuern. »

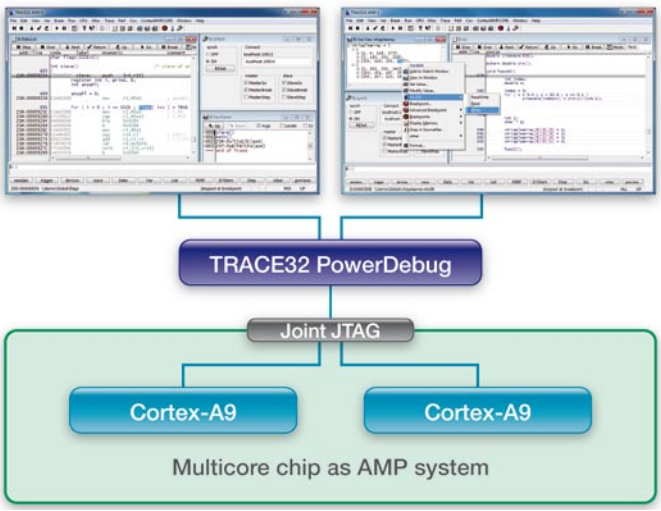

Bild 4: Beim Debugging von AMP-Systemen wird für jeden Core eine eigene TRACE32-Instanz gestartet.

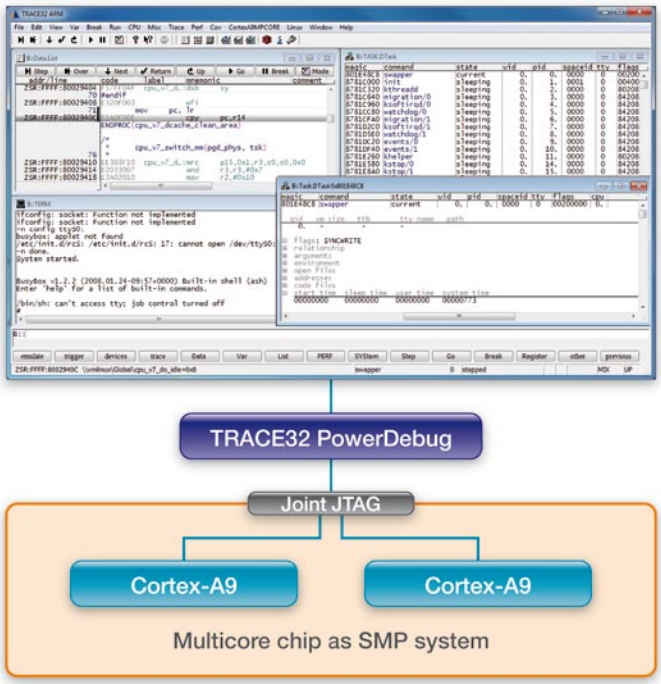

Bild 5: Beim Debugging von SMP-Systemen wird für alle Cores nur eine einzige TRACE32-Instanz gestartet.

Debug-Konzept für SMP-Systeme

Im Gegensatz zu AMP-Systemen, bei denen genau definiert ist, welche Aufgaben von welchem Core bearbeitet werden, wird diese Zuordnung bei SMP-Systemen nicht mehr vom Systemdesigner, sondern von einem SMP-Betriebssystem übernommen. Damit die Aufgaben frei jedem Core zugeteilt werden können, müssen alle Cores vom gleichen Typ sein.

Die Aufgabenzuteilung erfolgt dynamisch, das heißt abhängig vom aktuellen Systemzustand. Eine Aufgabeneinheit, die ein Betriebssystem zuteilen kann, wird als Task oder Thread bezeichnet. Ein zur Bearbeitung anstehender Task wird, vereinfacht gesagt, dem Core zugeteilt, der gerade frei ist.

Für das Debugging von SMP-Systemen wird nur **eine** TRACE32-Instanz geöffnet, von der aus alle Cores kontrolliert werden (siehe Bild 5 auf der vorigen Seite). Da für den Entwickler das Debugging eines einzelnen Tasks im Vordergrund steht, visualisiert die TRACE32-Bedienoberfläche den Zustand des Gesamtsystems immer aus Sicht eines einzelnen Tasks bzw. aus Sicht des Cores, auf dem dieser Task gerade läuft. Selbstverständlich lässt sich diese Visualisierung auch auf andere Tasks bzw. Cores umschalten.

TRACE32 übernimmt eine ähnliche Funktion wie das SMP-Betriebssystem. Es organisiert das Debuggen aller Cores, ohne dass sich der Entwickler um die Details des SMP-Systems kümmern muss. Wird beispielsweise ein Breakpoint gesetzt, so sorgt TRACE32 dafür, dass dieser in allen Cores eingetragen wird. Dies ist notwendig, da zum Zeitpunkt des Breakpoint-Setzens noch nicht klar ist, welcher Core den Programmteil mit dem Breakpoint ausführen wird. Hält ein Core dann am Breakpoint, werden automatisch auch alle anderen Cores gestoppt. Die Anzeige in TRACE32 wechselt auf den Task bzw. den Core, bei dem der Breakpoint zugeschlagen hat. Wird das Programm wieder gestartet, laufen alle Cores gemeinsam los.

Das Debuggen von SMP-Systemen mit TRACE32 ist einfach. Nachdem eine TRACE32-Instanz gestartet und für das SMP-System konfiguriert wurde, arbeitet man damit im Wesentlichen genauso, als würde man nur einen Core debuggen.

Trace-Konzepte

Auch die Darstellung und Auswertung von Traceinformation wird von TRACE32 unterschiedlich gehandhabt, abhängig davon, ob die Tracedaten von einem AMP- oder einem SMP-System erzeugt werden. Für AMP-Systeme wird die Traceauswertung für die einzelnen Cores weitgehend eigenständig durchgeführt. Die Traceinformationen für ein SMP-System hingegen lassen sich, abhängig von der Fragestellung, für einen einzelnen Task, für einen einzelnen Core bzw. für das Gesamtsystem auswerten.

Trace-Konzept für AMP-Systeme

Da das Debugging der einzelnen Cores eines AMP-Systems über separate TRACE32-Instanzen durchgeführt wird, erfolgt auch die Darstellung der Traceinformationen in den einzelnen Bedienoberflächen. AMP-Systeme können aus verschiedenartigen Cores bestehen, deshalb kommen unter Umständen unterschiedliche Traceprotokolle zum Einsatz. Die einzelnen Protokolle lassen sich durch ihre Aufteilung in die einzelnen Bedienoberflächen separat dekodieren und für die Auswertung aufbereiten.

Um die Zusammenarbeit der Cores zu testen und komplexe Systemfehler schnell zu finden, ist es möglich, die einzelnen Tracedarstellungen in ihrem zeitlichen Bezug zueinander darzustellen. TRACE32-PowerTrace stellt dafür eine gemeinsame Zeitbasis zur Verfügung. Diese erlaubt es, in einer Bedienoberfläche einen Zeitpunkt in der Tracedarstellung auszuwählen und in allen anderen Bedienoberflächen genau den Befehl zu sehen, der dort zum etwa gleichen Zeitpunkt ausgeführt wurde (siehe Bild 6). »

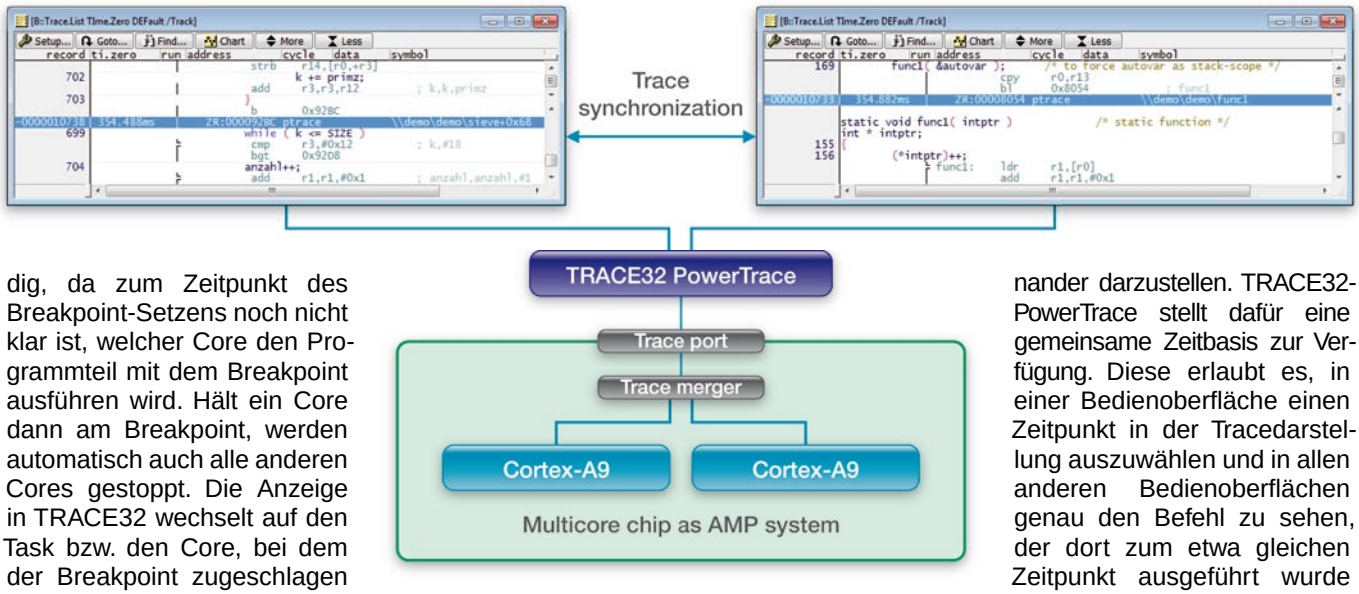

Bild 6: Beim Tracen von AMP-Systemen wird die Traceinformation für jeden Core in seiner Bedienoberfläche dargestellt. Eine zeitliche Synchronisierung zwischen den Bedienoberflächen ist möglich.

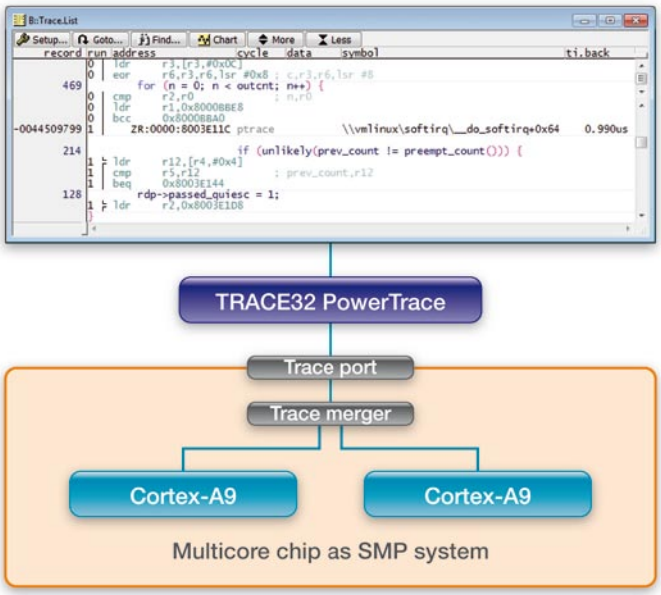

Bild 7: Beim Tracen von SMP-Systemen werden die Informationen für alle Cores in einem gemeinsamen Tracespeicher abgelegt.

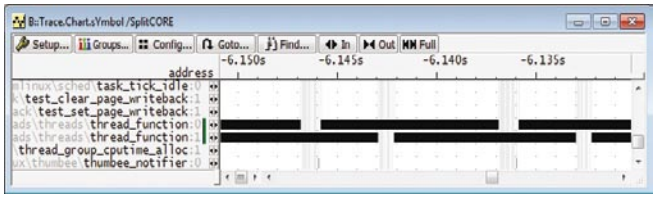

Bild 8: Die Traceauswertung zeigt, von welchen Cores die einzelnen Programmabschnitte bearbeitet wurden.

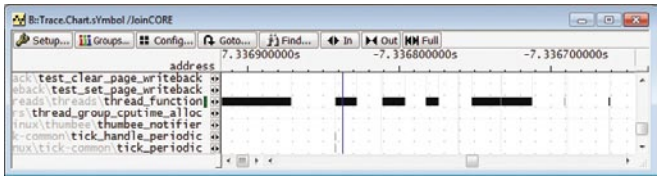

Bild 9: Die Traceauswertung analysiert das SMP-System als Ganzes; von welchem Core welcher Programmabschnitt bearbeitet wurde, spielt keine Rolle.

Trace-Konzept für SMP-Systeme

Obwohl die Informationen zum Programmablauf in einem SMP-System für alle Cores in einem gemeinsamen Tracespeicher abgelegt werden (siehe Bild 7), ist es von Vorteil, dass TRACE32 unterschiedliche Sichtweisen auf diese Information anbietet.

Für die Fehlersuche in einem Task bzw. für task-spezifische Laufzeitmessungen lässt sich die Traceinformation gezielt für einen einzelnen Task darstellen.

Stehen Fragestellungen wie „Von welchen Cores wurde mein Task bearbeitet?“ oder „Wie sieht die Laufzeit-Auslastung meiner Cores aus?“ im Vordergrund, ist es zweckdienlich, die Traceinformation für alle Cores gemeinsam darzustellen. Bild 8 zeigt eine solche Darstellung. Die Core-Nummer (0 oder 1) gibt hier an, auf welchen Cores die einzelnen Programmabschnitte gelaufen sind.

Untersucht man das SMP-System als Ganzes, hat man unter Umständen kein Interesse daran, zu erfahren von welchem Core welcher Task bzw. Programmabschnitt bearbeitet wurde. Auch für diese Sichtweise auf das SMP-System bietet TRACE32 Darstellungsoptionen (siehe Bild 9).

Bei der Aufbereitung und Auswertung der Traceinformation für SMP-Systeme gibt es noch viel Spielraum für zukünftige Erweiterungen. Lauterbach wird auch 2010, unter Berücksichtigung der Anforderungen seiner Kunden, neue Darstellungs- und Auswertefunktionen hinzufügen.

Aktuelles zum Thema RTOS-Debugging

Neu unterstützte RTOS	
FAMOS für ARM	verfügbar
Linux für Atom	verfügbar
SMP Linux für MIPS64	verfügbar
OKL4 für ARM	verfügbar
OS21 für ARM	verfügbar
SMP Symbian OS für ARM	verfügbar
Windows CE für Atom	Q1/2010

Anpassung an neue RTOS-Versionen

- LynxOS 5.0 für PowerPC
- MQX 3.x für ColdFire
- OSE 5.4
- QNX 6.4
- VxWorks 6.4
- µC/OS-III

Erweiterungen

- Partitionen- und MPU-/MMU-Unterstützung für µC/OS-II
- Paged Breakpoints für Symbian OS
- RTP-Unterstützung für VxWorks

Standardisierungsaktivitäten

Viele Kunden wünschen sich eine stärkere Standardisierung der Debug- und Tracetechnologie. Um die Entwicklung zweckmäßiger Standards mitzugestalten, arbeitet Lauterbach seit vielen Jahren in einer Reihe von internationalen Standardisierungsgremien mit. Diese aktive Mitarbeit macht es möglich, dass alle Standards bereits zum Zeitpunkt ihrer Verabschiedung in vollem Umfang von TRACE32 unterstützt werden.

ORTI-Standard

Einen der ersten Standards, den Lauterbach aktiv mitgestaltet hat, ist der ORTI-Standard. Bei diesem Standard handelt es sich um eine Beschreibungssprache, die den Aufbau sowie das Speichermapping der OSEK-Betriebssystem-Objekte für ein *RTOS Aware Debugging* aufbereitet und in einer Datei ablegt. Dieser innerhalb der Entwicklungspartnerschaft AUTOSAR entwickelte Standard wird seit 2003 von allen OSEK-Betriebssystemanbietern (ETAS Group, Vector u.a.) verwendet.

Das Arbeiten mit einem ORTI-File erfolgt so: Aus der Benutzerkonfiguration eines OSEK-Betriebssystems erzeugt der so genannte *OSEK System Builder* das ORTI-File. Dieses wird dann in den TRACE32-Debugger geladen, um ein *OSEK Aware Debugging* anzubieten (siehe Bild 10).

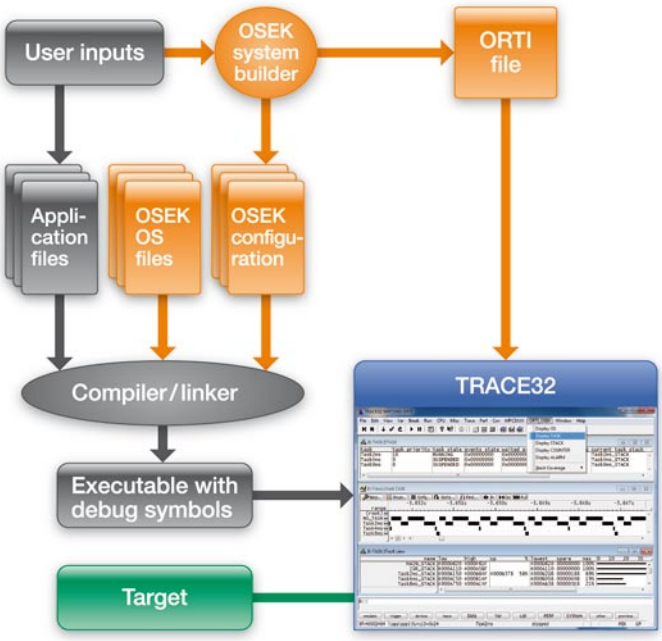

Bild 10: Der *OSEK System Builder* erzeugt das ORTI-File, das nach dem Laden in TRACE32 ein *OSEK Aware Debugging* erlaubt.

AUTomotive Open System ARchitecture

www.autosar.org

Arbeitsgruppe
OSEK/VDX Debug Interface Working Group

Standard
OSEK Run Time Interface Version 2.2, November 2005
http://portal.osek-vdx.org/files/pdf/specs/orti-a-22.pdf
http://portal.osek-vdx.org/files/pdf/specs/orti-b-22.pdf

TRACE32 kann so dem Entwickler folgende Funktionen anbieten (soweit der *OSEK System Builder* die dazu notwendigen Informationen im ORTI-File ablegt hat):

- Intuitive Darstellung der OSEK-Ressourcen (Bild 11 zeigt beispielsweise die Alarm-Liste)
- Task-spezifische Breakpoints
- Eine Task-Stack Analyse
- Eine Task-Context Anzeige
- Analyse der Task-Laufzeiten (siehe Bild 13)
- Analyse der Service-Laufzeiten

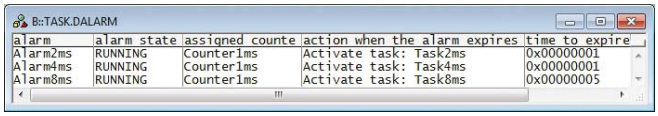

Bild 11: Die Alarm-Liste eines OSEK-Betriebssystems in TRACE32

NEXUS-Standard

Bereits 1998 wurde im Rahmen des NEXUS 5001 Forums ein erster Vorstoß zur Standardisierung einer Debug- und Traceschnittstelle für Embedded Prozessoren unternommen. 2003 wurde dann der NEXUS-Standard verabschiedet. Dieser umfasst:

- Eine JTAG-Schnittstelle, in der Regel IEEE1149.1
- Mechanismen, die es dem Debugger erlauben, Speicher zu lesen und zu schreiben, auch während der Prozessor das Programm abarbeitet
- Ein message-basiertes Traceprotokoll (Programmfluss- sowie Datentrace)
- Debugging und Tracing auch für Multicore-Prozessoren
- Einen Hardware-Layer
- Standard-NEXUS-Stecker

»

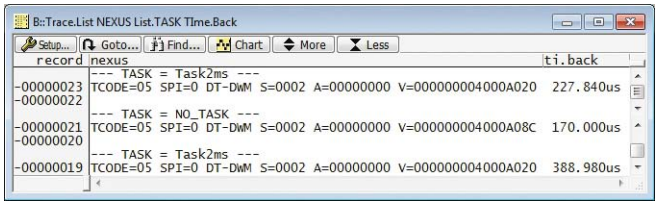

Bild 12: NEXUS-Messages zur Aufzeichnung von Task-Wechseln

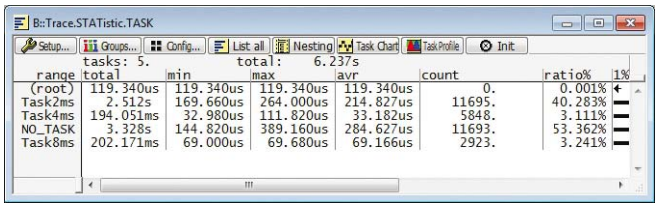

Bild 13: Analyse der Task-Laufzeiten für ein OSEK-Betriebssystem

Lauterbach unterstützt den NEXUS-Standard bereits seit 2001 für verschiedene Prozessorarchitekturen, aktuell vor allem für die in der Automobil-Industrie weit verbreitete MPC55xx/MPC56xx-Familie von Freescale und ST Microelectronics. Bild 12 und 13 zeigen, wie das NEXUS-Traceprotokoll verwendet wird, um die Laufzeiten von OSEK-Tasks zu messen.

Da der NEXUS-Standard viele Freiheiten bei der Implementierung lässt, hat Lauterbach 2008 seine NEXUS-Hardware noch einmal vollständig überarbeitet. Die neue FPGA-basierte Hardware lässt sich nun flexibel an alle NEXUS-Implementierungen anpassen. Als neue Features unterstützt sie auch eine Abtastpunkt-Optimierung für die Tracesignale und das neue JTAG-Protokoll IEEE 1149.7. Mehr zum Thema IEEE 1149.7 finden Sie auf Seite 10.

Der NEXUS-Standard von 2003 wird aktuell überarbeitet. Obwohl die neue Version des Standards noch nicht verabschiedet ist, gibt es bereits Prozessoren aus der MPC56xx-Familie, die diesen Standard implementiert haben. Seit Anfang 2010 wird das neue Protokoll von TRACE32 per Software-Update unterstützt.

Nexus 5001™

NEXUS 5001™ FORUM

www.nexus5001.org

Standard
Standard for a Global Embedded Processor
Debug Interface Version 2, Dezember 2003
http://www.nexus5001.org/st/ieee_isto_5001_2003.pdf

Serieller Trace für QorIQ

Bild 14: Der TRACE32 *High-Speed Serial Trace* für den QorIQ

Zur embedded world 2010 in Nürnberg wird Lauterbach seinen neuen *QorIQ High-Speed Serial Trace* (siehe Bild 14) vorstellen. Die Tracetechnologie des QorIQ P4xxx basiert auf einem NEXUS-Protokoll, einem AURORA-basierten Hardware-Layer und einem Stecker, der in einer Arbeitsgruppe der Power.org Organisation spezifiziert wurde.

Der erste unterstützte Prozessor wird der QorIQ P4080 vom Freescale sein. Der P4080 SoC verfügt über 8 *Power Architecture* e500mc Kerne, die in SMP- oder AMP-Konfiguration arbeiten können. Gemischte Konfigurationen aus SMP- und AMP-Gruppen sind ebenfalls möglich. Ausführliche Information zum Thema „Debugging von AMP- und SMP-Systemen“ finden Sie in einem Artikel auf der Seite 5 dieses Newsletters.

Serielle Traceschnittstellen haben gegenüber parallelen die folgenden Vorteile: Pinreduktion durch serielle Übertragung und hoher Datendurchsatz durch die Übertragung mit differentiellen Signalen.

Die Fülle an Tracedaten verlangt natürlich nach einem entsprechend großen Tracespeicher. Dieser wird vom Lauterbach-Produkt PowerTrace II mit einem Speicherausbau von bis zu 4 GByte bereitgestellt.

Der *High-Speed Serial Trace* für den QorIQ ist für maximal vier high-speed Kanäle ausgelegt. Folgende Übertragungsraten werden unterstützt:

- max. 6,25 GBit/s je Kanal bei bis zu 3 Kanälen
- max. 3,125 GBit/s je Kanal bei 4 Kanälen

Der Abgriff der Tracedaten ist über ein Steckersystem der Firma Samtec realisiert. Lauterbach bietet für die verschiedenen Varianten des Steckers Adapter an. »

Power.org

www.power.org

Arbeitsgruppe
Common Debug Interface Technical Subcommittee

Standard
Power.org™ Standard for Physical Connection for High-Speed Serial Trace, Juli 2008
http://www.power.org/resources/downloads/Power_CDI_Physical_Connection_for_HSST_APPROVED_v1.0.pdf

IEEE Standards Association

www.standards.ieee.org

Arbeitsgruppe: 1149.7

Standard
Verabschiedung des Standards ist für Q1/2010 geplant

cJTAG / IEEE 1149.7

Mit IEEE 1149.7, auch bekannt als cJTAG (compact JTAG), wird der traditionelle JTAG-Standard IEEE 1149.1 an die aktuellen technischen Anforderungen angepasst. Für Lauterbach als Debugger-Hersteller ist innerhalb dieses Standards vor allem die Definition einer 2-Pin Debugschnittstelle interessant.

Die gesamte Kommunikation zwischen Debugger und Core wird bei IEEE 1149.7 über die Pins TCK und TMS abgewickelt. Die Neuerungen gegenüber der Standard-Debug-Kommunikation sind in Kürze:

- IEEE 1149.7 ist lediglich nach außen eine 2-Pin-Schnittstelle. Im Chip setzt der 1149.7 Controller die Kommunikation wieder auf Standard-JTAG um (siehe Bild 15).
- Das IEEE 1149.7 Protokoll ist, vereinfacht ausgedrückt, gegenüber der Standard-JTAG-Schnittstelle serialisiert (siehe Bild 16).

Da das TMS-Signal für Standard-JTAG ein unidirektionales Signal ist, musste Lauterbach seine Debug-Kabel auf das bidirektionale TMS umstellen.

- Alle Debug-Kabel für die ARM-/Cortex-Architekturen wurden bereits zum Jahresbeginn 2008 umgestellt (*Debug Cabel V4*).
- Das Debug-Kabel für die MPC55xx-Architektur wird seit September 2009 in einer aktualisierten Version (*OnCE Debug Cable with serial JTAG*) ausgeliefert.

Die neuen Debug-Kabel unterstützen selbstverständlich weiterhin auch den traditionellen 1149.1 Standard.

TRACE32 wurde schon im Oktober 2009 erfolgreich mit cJTAG getestet. Mit einer Freigabe muss allerdings noch bis zur offiziellen Verabschiedung des 1149.7 Standards gewartet werden. »

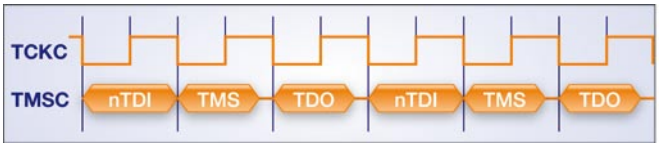

Bild 16: Vereinfachte Darstellung des 1149.7 Protokolls

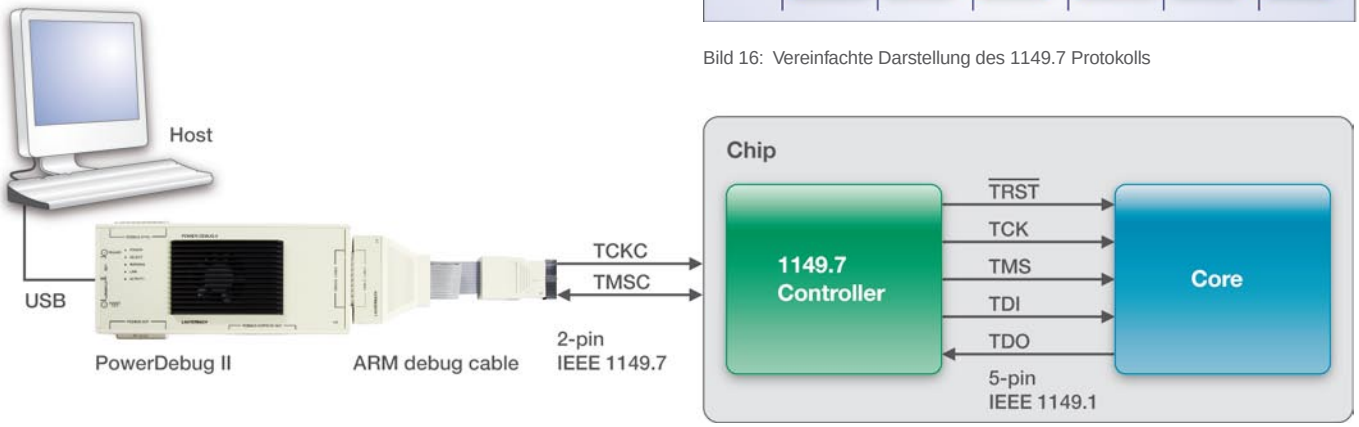

Bild 15: Das IEEE 1149.7 Protokoll wird im Chip auf das Standard-JTAG Protokoll umgesetzt.

Open SoC Design Platform for Reuse and Integration of IPs

SPRINT

www.sprint-project.net

Arbeitsgruppe: Debug and Analysis

Standard: Multi-Core Debug API v1.0, April 2009
http://www.lauterbach.com/sprint_mcd_api_v1_0.zip

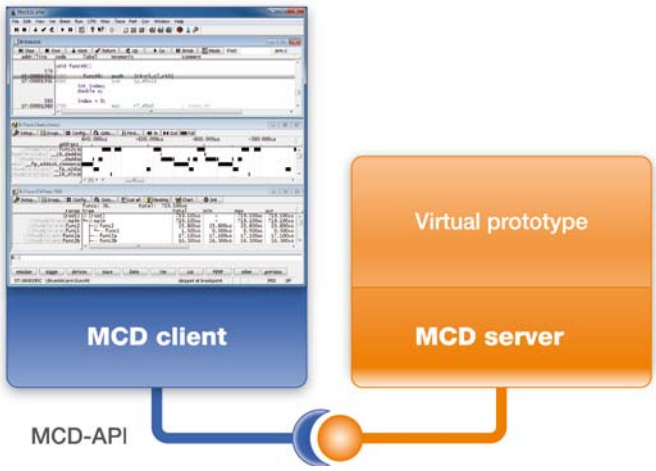

Bild 18: Das Debugging des Virtual Prototypes erfolgt über die MCD-API.

MCD-API

Die MCD-API definiert eine C-Schnittstelle für das Debugging von Multicore-Systemen. Dabei macht es keinen Unterschied, ob es sich bei dem Multicore-System um eine reale Zielhardware oder um eine Software-Simulation handelt. Lauterbach sieht aktuell zwei Anwendungen für die MCD-API:

1. MCD-API als Standardschnittstelle zu TRACE32, als Teil der neuen MCD-Remote-API
2. MCD-API als Standardschnittstelle zu *Virtual Prototypes*

1. Aktualisierte TRACE32 Remote-API

Mit der Lauterbach Remote-API kann eine externe Anwendung TRACE32 steuern. Sie ermöglicht beispielsweise die Automatisierung von Regressionstests.

Die sich stetig verändernden Anforderungen unserer Kunden sind über die Jahre in die Remote-API eingeflossen. Aktuell erfordert die steigende Anzahl von Multicore-Systemen weitere, umfassende Anpassungen.

Statt die aktuelle Remote-API zu erweitern, hat sich Lauterbach entschlossen, seine Remote-API basierend auf dem MCD-Standard neu aufzusetzen. Aller Voraussicht nach deckt die MCD-API alleine nicht alle TRACE32-Anforderungen ab, so dass TRACE32-spezifische Funktionen dazukommen werden (siehe Bild 17).

Die erste Version der neuen TRACE32 MCD-Remote-API wird voraussichtlich zur Jahresmitte 2010 zur Verfügung stehen. Zu diesem Zeitpunkt wird die aktuelle, so genannte Legacy-Remote-API „eingefroren“. Für unsere Kunden bedeutet dies, dass wir selbstverständlich weiterhin Support für die Legacy-API gewährleisten, neue Funktionalitäten werden dort jedoch nicht mehr implementiert.

2. Standardschnittstelle zu Virtual Prototypes

Seit fünf Jahren kann TRACE32 zum Debuggen so genannter *Virtual Prototypes* verwendet werden. Dabei stellen die Anbieter von *Virtual Prototypes* in der Regel eigene APIs für das Debugging zur Verfügung. Mit der MCD-API besteht eine standardisierte Schnittstelle zwischen Debugger und *Virtual Prototype* (siehe Bild 18). Diese hat folgende Vorteile:

- Schnelle Anpassung an neue *Virtual Prototypes*
- Großer, leistungsfähiger und gut ausgetesteter Funktionsumfang

Bild 17: Über die Remote-API kann TRACE32 durch eine externe Anwendung gesteuert werden.

TRACE32-Expertenforum

Rückblick – Expertenforum Automotive

Am 28. Oktober 2009 veranstalteten wir in unseren neuen Räumlichkeiten des ARCONE Technologie Centers zum ersten Mal das TRACE32-Expertenforum mit dem Schwerpunktthema Automotive.

Viele Kunden und Partner nutzten die Chance, sich neueste Informationen aus erster Hand einzuholen. Dabei konnten sich die Teilnehmer über die Neuerungen in der Automobil-Software informieren und wie diese bereits von TRACE32 unterstützt werden. So befasste sich die einleitende Vortragsreihe mit dem Thema AUTOSAR und seinem OSEK-Betriebssystem, sowie dessen Beschreibungsschnittstelle ORTI. Abgerundet wurde dieser Themenbereich mit Beispielen der Integration von TRACE32 in die UML-Entwicklungsumgebung Rhapsody, sowie die Einbindung in die grafische Testumgebung LabView.

Nach einer stärkenden Pause ging es noch tiefer in die Details. Dabei konnten sich die Teilnehmer für eine von drei Expertenrunden entscheiden. Zur Auswahl standen die Themen TriCore, PowerPC und Logikanalysatoren. Hier standen unsere Entwickler zu ihrem Fachgebiet Rede und Antwort. Den Abschuss bildete ein gemeinsames Abendessen, bei dem in gemütlicher Atmosphäre noch das ein oder andere technische Thema diskutiert wurde.

Expertenforum ARM 5. Mai 2010

Debugging

- Programmieren von NOR/NAND/SERIAL FLASH
- Linux-Debugging
- DCC, Virtual Terminal, Semihosting
- Multicore-Debugging

Tracen mit der ARM-ETM

- Basiswissen ETM
- CoreSight-ETM
- 4-Bit-ETM für Cortex-M
- Trace-Darstellung und Auswertung

Ausblick – Expertenforum ARM

Der rundum positive Eindruck sämtlicher Teilnehmer unseres ersten Expertenforums hat uns gezeigt, wie wichtig dieser Informationsaustausch ist. Deshalb werden wir weitere Veranstaltungen dieser Art mit anderen Schwerpunkten anbieten.

Am 5. Mai 2010 veranstalten wir ein Expertenforum mit dem Schwerpunktthema ARM. Hierbei werden Sie einen tiefen Einblick in die Debug-Möglichkeiten der verschiedenen ARM- und Cortex-Cores erhalten und können mit unseren Experten sämtliche Aspekte und Neuheiten zum Thema Debugging und Tracing für diese Architekturen diskutieren. Die geplanten Themen sehen Sie in der obigen Programmvorschau. Aktuelle Informationen können Sie jederzeit über unsere Homepage abrufen.

WELTWEITE NIEDERLASSUNGEN

- **Deutschland**
- Frankreich
- Großbritannien
- Italien
- USA
- China
- Japan

In allen anderen Ländern vertreten durch kompetente Partner

BENACHRICHTIGEN SIE UNS

Falls sich Ihre Adresse geändert hat oder Sie kein Mailing mehr von uns erhalten möchten, schicken Sie bitte eine E-Mail an **info@lauterbach.com**