2011年简讯

LAUTERBACH
DEVELOPMENT TOOLS

DEBUGGER, REAL-TIME TRACE, LOGIC ANALYZER

始终保持技术领先

Lauterbach 是各种最新调试技术的全球领先者和行业内的潮流引导者。三十多年来，Lauterbach 一直致力于为嵌入式行业提供最先进的开发工具，并始终保持技术领先地位。

这也帮助我们迅速获得全球所有大中型半导体厂商的认可，许多年来，参与新技术开发与实施的各种机构和个人积极需求与 Lauterbach 合作，广泛的合作为 Lauterbach 带来源源不断的灵感和开发思路，最终转化为先进的产品。

此外，Lauterbach 始终强调以客户为中心，例如，我们的 TRACE32 用户提出的各种优秀的要求和建议，为我们的产品开发带来及大的帮助。在尽可能情况下，我们都会以最快的速度采纳客户提出的合理建议，并将它们体现在下一款调试器产品中。

Lauterbach 身处行业领先位置，关于行业发展的未来趋势有何预见呢？在不久的将来，市场上会出现哪些技术?

Android 系统调试

Android 系统调试显然是一个重要的研究课题。市场上不断涌现出各种针对虚拟机（VM）的移动电话硬件架构独立的应用程序。谷歌的 Android 操作系统与 Dalvik VM 目前非常流行，而当上层的应用程序，虚拟机和操作系统以及地层的硬件平台共同协同工作时，可能会出现各种复杂的问题，所有层面都有可能是错误原。为了调试和解决这些问题，就需要从 Java 应用程序到Linux硬件驱动程序，所有的软件层都必须做到透明。

应一些手机厂商的要求，Lauterbach 于 2010 年中开始开发一款面向 *VM Debugging Awareness* 的应用程序接口（API）。Android 在这里被用作一个基准平台。目标是为用户提供一个开放的界面，不管是开源程序提供者，还是闭源 VM 虚拟机供应商，都可以使用 TRACE32 进行调试，从而优化他们的产品。 »

目录

关于虚拟机调试识别插件（VM Debugging Awareness）及其状态的详细信息，请参见第 6 页“面向 VM Debugging Awareness 的 API”。

能耗分析

随着全球气候变暖问题日益显现以及各种“绿色”电子系统的纷纷面世，如何测量嵌入系统的能耗逐渐成为行业关注的焦点。在现代技术刊物中，关于电池驱动设备与低功耗微处理器的文章随处可见。该领域内越来越多的新技术赢得了各种创新奖项。

然而，在移动终端领域，待机时间和工作时间一直是一个重要议题。多年来，行业内已采取了各种降低系统能耗的措施。但如果不能通过一个软件时刻来监控嵌入式系统，并使各个硬件工作在的剩电模式，则这些措施很难真正见效。

自 2006 年初以来，Lauterbach 工具已开始支持各种测量功能，能够对嵌入式系统内的软件与功耗之间的相互影响进行比较分析。该技术从 2010 年中开始用在 TRACE32 CombiProbe 中。详细信息请参见第 11 页“使用 CombiProbe 进行能量分析”。

多核调试

虽然多核芯片用于嵌入式系统中已有 10 年历史，且 Lauterbach 早在 2001 年就开发出多核芯片调试器，但该领域仍然是一个技术热点。由于目前市场上关于增强内部系统操作可见性的需求正日益提高，因此在芯片调试结构内集成新型跟踪单元亦已成为大势所趋。

以前只能够对单核生成跟踪信息，而现在可以有很多的其他跟踪源：

a) 通过这些跟踪源，即使芯片内部总线上的传输信息亦可以看到：

- 使用 AMBA AHB 追踪宏单元（HTM）的 ARM CoreSight 跟踪调试技术
- 使用系统外围总线（SPB）和局部存储总线（LMB）的 MCDS 跟踪技术，用于英飞凌（Infineon）公司的 TriCore 处理器
- 用于德州仪器 (TI) 芯片的 RAM 跟踪端口
- DMA 和 FlexRay 跟踪技术，用于 NEXUS Power Architecture 架构

b) 跟踪源会生成芯片内部 IP（知识产权）的跟踪信息，例如特殊中断跟踪信息。

c) 跟踪源容许输出软件生成的跟踪信息，例如：

- 用于 ARM CoreSight 跟踪调试技术的指令跟踪宏单元（Instrumentation Trace Macrocell）
- 用于 ARM CoreSight 跟踪调试技术的系统跟踪宏单元（STM）

TRACE32 调试器重视不断改进和持续开发，确保支持最新的跟踪源，而且保证调试器的配置简单，以及能够提供所得信息的综合分析功能。

串行端口跟踪

为了实现内部芯片流程的可见性，需要采集和处理大量跟踪数据，因此复杂的多核芯片和高性能处理器必须具有更大的带宽和以及更快速的跟踪端口。

为了满足芯片制造商的要求，Lauterbach 开发出串行跟踪端口，这是近几年来行业内非常重要的创新产品之一。硬盘制造商很多年前就开始使用串行接口实现硬盘与计算机之前的高速数据交换，但直到 2008 年才首次使用 ARM 的高速串行跟踪端口（HSSTP）输出跟踪信息。在同一年内，Lauterbach 开始推出采用串行端口技术的跟踪工具。

在此期间，还有一些其他处理器使用串行跟踪接口。关于这一领域的最新发展，请参阅第 9 页文章“串行跟踪端口应用增长”。

更大的跟踪存储器

快速跟踪接口与高数据传输率显然都需要更大的跟踪存储空间，否则，将根本无法存储复杂嵌入系统大型程序的记录与长时间程序分析。 »

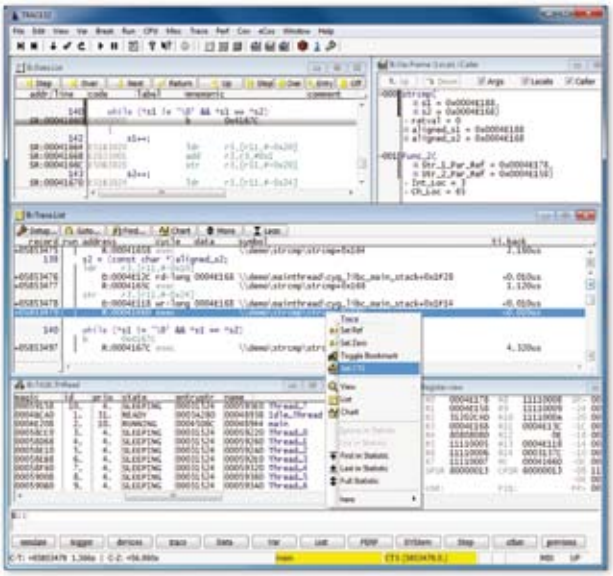

图 1：为确保跟踪调试程序快速执行，平均需要 4GB 的跟踪存储空间

然而，如果不能提供跟踪信息快速处理的实现，即使提供再多的跟踪存储空间也没有任何意义。高要求的跟踪分析功能（例如*基于跟踪的调试系统*）更是如此（参见图 1）。随着 SDRAM 芯片性能的不断提高、高速计算机及 GB 级以太网的出现，Lauterbach 于 2007 年开始推出 4GB 内存的 PowerTrace II 跟踪工具。

在 2008 年中，Lauterbach 开始开发一种集跟踪记录、分析和*实时流*传输于一体的新方法。我们之所以开发这种方法，正是考虑到客户需要长时间代码覆盖分析技术、综合系统运行时间分析技术、以及需要更长跟踪记录时间以定位随机事件。

*实时跟踪系统*的新特性是：跟踪数据必须在被工具记录的同时也被传输给主机。然后，主机收到跟踪数据后立即进行分析，另外，跟踪数据在被分析时，还可以保存在硬盘上。

仅当跟踪数据的所有处理步骤均达到最佳速度时，*实时跟踪系统*才能发挥作用。这种技术可用于跟踪数据的传输和分析、以及彻底搜索硬盘文件内的跟踪信息。

即使是传统的跟踪方法，同样能够从这种最新的速度优化方法中获益。例如，我们正计划将跟踪压缩方法（最初为*实时跟踪系统*设计）用于传统的跟踪方法。关于跟踪数据压缩的详细信息，请参见第 10 页。

基于跟踪的调试

基于跟踪的调试（又称为 CTS = 情境跟踪系统）允许我们重新调试被跟踪的程序。TRACE32 使这种技术最终成为可能，用户可以在 PowerView 图形用户界面内，为每条单独的跟踪记录重建目标系统状态描述，包括跟踪记录、存储内容、可变状态、跟踪源和任务列表、栈帧，等等。

在为基于跟踪的调试选择一个起点后，即可使用所有调试指令。这些指令由 TRACE32 根据跟踪记录进行重建后执行。对于基于跟踪的调试系统能够支持后退一步或返回起始点这类操作，大多数用户均表示赞赏。

基于跟踪的调试系统同时提供了一系列其他有用的功能：

- 高级语言程序及各种局部变量显示跟踪数据
- 程序运行分析与函数调用嵌套
- 修正跟踪间隙，当所产生的跟踪数据量超过跟踪端口可传输的数据量时，可能就会形成跟踪间隙。

www.lauterbach.com/cts.html

观点

除了目前的趋势外，我们在调试技术还取得许多新开发成果。如果您仔细阅读我们的《2011 年简讯》，也许就能从中发现一两项开发技术可能会对你有用。

新支持的处理器

新处理器产品	
Actel	**LA-7844 (Cortex-M)** • A2F060, A2F200, A2F500
AppliedMicro	**LA-7723 (PPC400)** • APM80186, APM821x1 • APM86290 **LA-7752 (PPC44x)** • PPC460SX
ARM	**LA-7843 (Cortex-A/R)** • Cortex-A15 • Cortex-A15 MPCore **LA-7844 (Cortex-M)** • Cortex-M4 • SC000, SC300
Atmel	**LA-7844 (Cortex-M)** • AT91SAM3S, AT91SAM3N **LA-3779 (AVR32)** • AT32UC3A/B/C/D/L
Broadcom	**LA-7760 (MIPS32)** • BCM3549/35230/4748 • BCM5354/5358/5331X • BCM6816/6328/6369 • BCM7407/7413/7420
Cavium	**LA-7761 (MIPS64)** • CN63XX
Ceva	**LA-3711 (CEVA-X)** • CEVA-X1643, CEVA-XC
Cortus	**LA-3778 (APS)** • APS3/B/BS/S
Cypress	**LA-7844 (Cortex-M)** • PSoC5
Faraday	**LA-7742 (ARM9)** • FA726TE
Freescale	**LA-7736 (MCS12X)** • MCS9S12GC/GN/Q **LA-7732 (ColdFire)** • MCF5301x, MCF5441x **LA-7845 (StarCore)** • MSC8156 **LA-7742 (ARM9)** • i.MX28 **LA-7843 (Cortex-A/R)** • i.MX53 **LA-7844 (Cortex-M)** • Kinetis
Freescale （续上）	**LA-7753 (MPC55xx/56xx)** • MPC5602D/P • MPC564XA/B/C/S • MPC567XF/R **LA-7729 (PowerQUICC II)** • MPC830X **LA-7764 (PowerQUICC III)** • P10xx, P20xx, P40xx • P3041 (2H/2011) • P5010, P5020 (2H/2011)
Fujitsu	**LA-7844 (Cortex-M)** • FM3
Infineon	**LA-7756 (TriCore)** • TC1182, TC1184 • TC1782, TC1782ED • TC1784, TC1784ED • TC1791, TC1791ED • TC1793, TC1793ED • TC1798, TC1798ED **LA-7759 (XC2000/C166S V2)** • XC22xxH/I/L/U • XC23xxC/D/E/S • XC27x2/x3/x7/x8 • XE16xFH/FU/FL
Intel®	**LA-3776 (Atom™/x86)** • E6xx, Z6xx, N470 • Core i3/i5/i7, Core2 Duo
Lantiq	**LA-7760 (MIPS32)** • XWAY xRX200
LSI	**LA-7765 (ARM11)** • StarPro2612, StarPro2716 **LA-7845 (StarCore)** • StarPro2612, StarPro2716
Marvell	**LA-7742 (ARM9)** • 88F6282, 88F6283, 88F6321 • 88F6322, 88F6323 **LA-7765 (ARM11)** • 88AP510-V6 **LA-7843 (Cortex-A/R)** • 88AP510-V7
MIPS	**LA-7760 (MIPS32)** • MIPS M14K, MIPS M14KC
Netlogic	**LA-7761 (MIPS64)** • XLR, XLS
NXP	**LA-7844 (Cortex-M)** • LPC11xx • EM773

新处理器产品	
Ralink	**LA-7760 (MIPS32)** • RT3052, RT3662
Renesas	**LA-3777 (78K0R/RL78)** • 78K0R/Hx3/Lx3/Ix3 • 78F804x, 78F805x • RL78/G12, RL78/G13 **LA-3786 (RX)** • RX610/6108/621/62N/630
STMicro-electronics	**LA-7753 (MPC55xx/56xx)** • SPC560D/P, SPC56APxx • SPC564Axx, SPC56ELxx **LA-7844 (Cortex-M)** • STM32F100, STM32L15x
ST-Ericsson	**LA-7843 (Cortex-A/R)** • DB5500, DB8500
Tensilica	**LA-3760 (Xtensa)** • LX3

Texas Instruments	**LA-3713 (MSP430)** • MSP430xG461x • MSP430x20x1/x2/x3 **LA-7742 (ARM9)** • AM1707/1808/1810 **LA-7843 (Cortex-A/R)** • OMAP36xx **LA-7838 (TMS320C6x00)** • OMAP36xx
Toshiba	**LA-7742 (ARM9)** • TMPA900, TMPA910 **LA-7844 (Cortex-M)** • TMPM330, TMPM370
Trident	**LA-7760 (MIPS32)** • HiDTV PRO-QX
Wintegra	**LA-7760 (MIPS32)** • WinPath3, WinPath3-SL
Zoran	**LA-7760 (MIPS32)** • COACH 12

在快速模型中跟踪虚拟平台

Lauterbach 自 2010 年 11 月起开始提供支持 ARM 快速原型的跟踪系统。

在产品开发过程中，我们通常使用硬件的软件仿真平台，这样就可以同时开始进行软件开发，无需等待第一个硬件平台完成。通过快速原型工具，ARM 为用户提供一系列软件包，帮助用户便利的完成基于 ARM 的硬件产品设计的工作。

从 2008 年开始，Lauterbach 支持通过 CADI 接口调试快速仿真模型。现在又提供了模型跟踪接口支持，5.1 版即能够支持快速仿真模型跟踪接口。为了正确处理跟踪信息并保存在虚拟目标中，调试器制造商可以加载单独的跟踪插件。图 2 简单示意了 TRACE32 与快速仿真模型之间的相互通信机制。

关于调试虚拟平台的详细信息，请参阅：

www.lauterbach.com/frontend.html

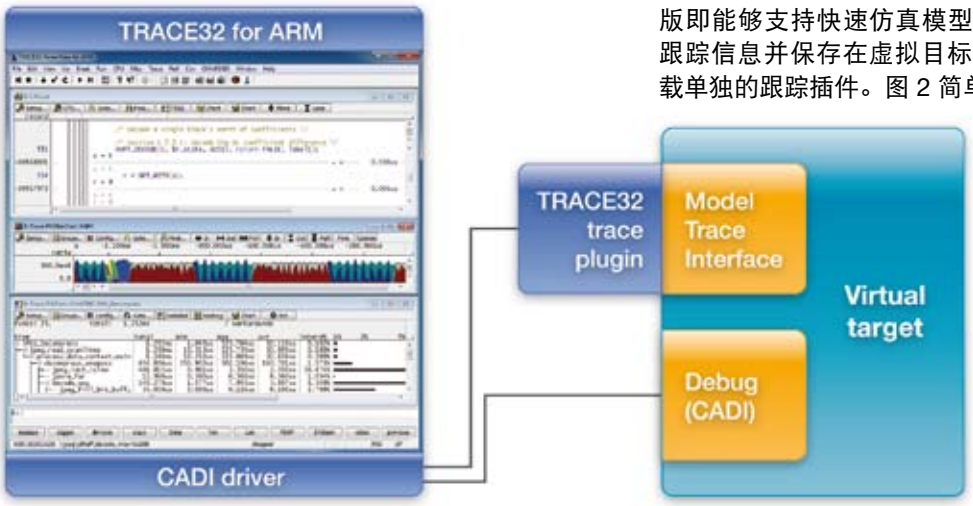

图 2：TRACE32 支持虚拟目标的调试与跟踪。

面向 VM Debugging Awareness 的 API

自 2006 年以来，Lauterbach 支持 Java 程序的调试，适用于 Java 虚拟机 J2ME CLDC、J2ME CDC 和 Kaffe。由于虚拟机越来越受到欢迎，因此虚拟机供应商的数目正在快速增涨。目前，并非所有虚拟机都是开源的，为了让虚拟机供应商及其用户能够根据其虚拟机特性，灵活的调整调试功能，Lauterbach 从 2010 年中开始致力于开发一种新的解决方案。

以 Android Dalvik 虚拟机在 ARM 核的实现，做为停止模式下开发虚拟机应用程序接口的范例.

两个“调试世界”

对于系统开发者，Android 是一个开源软件栈，包括以下组件（见图 3）：

- Linux 内核及其硬件驱动程序。
- Android Runtime 与 Dalvik 虚拟机以及一系列程序库：经典 Java 内核库，Android 特殊库、C/C++ 程序库。
- Java 应用程序及其支持的应用构架。

Android 软件可采用各种语言编写：

- Linux 内核、一些程序库与 Dalvik 虚拟机代码可采用 C、C++ 或 Assembler 编写。

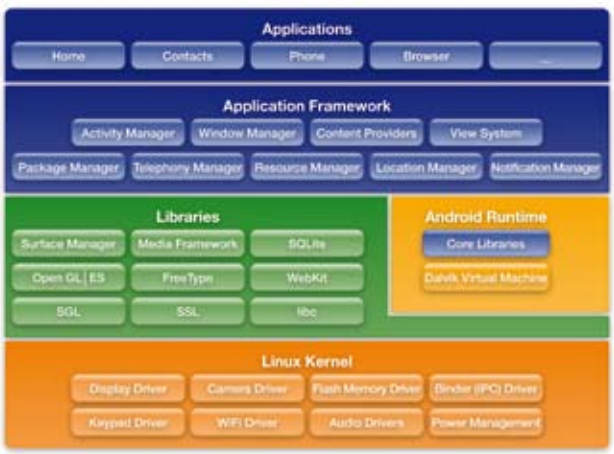

图 3：开源安卓软件栈。

- 虚拟机应用程序及其支持的应用构架可采用 Java 语言编写。

每个代码块都在单独的“调试世界”内测试。

调试 C/C++ 程序和汇编程序代码

通过使用 JTAG 接口，采用 C/C++ 和汇编器编写的 Android 程序可以在目标硬件上以停止模式调试。在停止模式调试时，TRACE32 调试器可直接与 Android 硬件平台的处理器通讯（见图 4）。

图 4：在停止模式调试中，调试器可直接与安卓硬件平台上的处理器通讯。

停止调试模式的特点是：当处理器被停止以进行调试时，整个 Android 系统亦停止运行。

停止模式调试具有以下主要优势：

- 只需一个有效的 JTAG 接口即可实现调试器与处理器之间的通讯。
- 无需在目标上加载调试服务程序，因此非常适合于测试已发布软件。
- 它允许实时测试，因此能够有效调试仅在实时情况下才出现的问题。 »

目前，停止模式调试暂不支持在 Dalvik VM 等虚拟机上调试 VM 应用程序，因此要实现所有软件层上均能够透明调试仍然需要一段时间。

调试 Java 代码

Android 系统的 Java 代码通常采用集成到 Eclipse 中的 Android 开发工具（ADT）进行测试。adb 服务器—adb 表示安卓调试桥，通过主机上的 USB 或以太网实现与目标上的 adb 后台程序之间的通讯（图 5）。

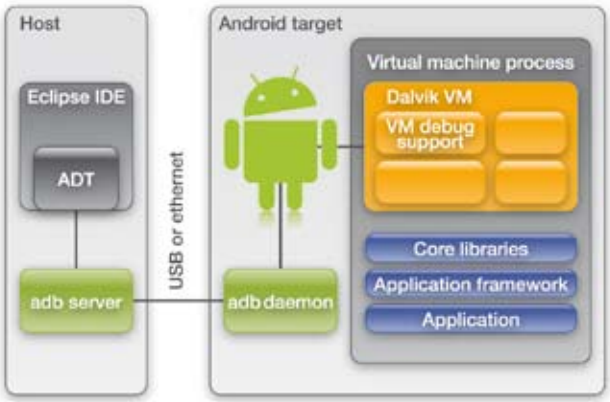

图 5：集成到 Eclipse 内的安卓开发工具（ADT）用于调试 Java 代码。

使用 ADT 调试的先决条件是：VM 应用程序包含调试信息，安卓调试支持程序（adb 后台程序）在硬件平台上运行。

使用 ADT 调试 Java 代码是非常合适的，不过有几种情形 ADT 可能并不能带来帮助，它们是：

- 使用已发布代码的早期代码（例如:初始化代码，Linux 内核等）。
- Java 应用程序与 C/C++ 服务程序或 Linux 硬件驱动程序相互作用时发生的错误。
- Adb 服务器与 adb 后台程序之间发生通信故障后进行的调试。

VM Aware 停止模式调试

为了在实时条件下详细测试 Android 系统，包括从 Java 应用程序，linux 内核到 Linux 硬件驱动程序等，Lauterbach 现已在其停止模式调试系统中添加了VM 调试识别插件。

JTAG 调试器可直接与安卓硬件平台上的处理器通讯。因此，调试器可在处理器停止运行后访问所有系统信息。调试器现在采用“简便的艺术”风格界面为用户提供正确的信息，摆脱抽象的字与字节描述，帮助用户更好地理解。

一个抽象层使 TRACE32 用户可以在几个虚拟地址空间内选择调试操作系统软件，另一个抽象层是 Java 调试，目前仍独立于操作系统调试。

要调试运行在类似安卓系统内虚拟机上的应用程序（虚拟机本身是安装在操作系统进程内），操作系统调试与 Java 调试必须联合起来。为了解决这个复杂的新挑战，Lauter-bach 正在开发一种全新的、开放和易于扩展的解决方案。

开源解决方案

在未来，Lauterbach 的停止模式调试系统将支持以下抽象层：

- 高级语言调试
- 目标操作系统调试识别
- VM 调试识别

高级语言调试是 TRACE32 软件的一个固定组件，可配置用于装载符合程序的符号表和调试信息。 »

Dalvik 虚拟机

Dalvik 是在安卓系统内使用的虚拟机的名称。Dalvik 虚拟机是执行 Java 字节代码的处理器的软件模型。虚拟机允许编写独立于处理器的软件。即使你改换新硬件平台，只需移植虚拟机即可使用应用软件。

用于 VM 运行的已编译软件在已移植其虚拟机的任何平台上均可自动运行。

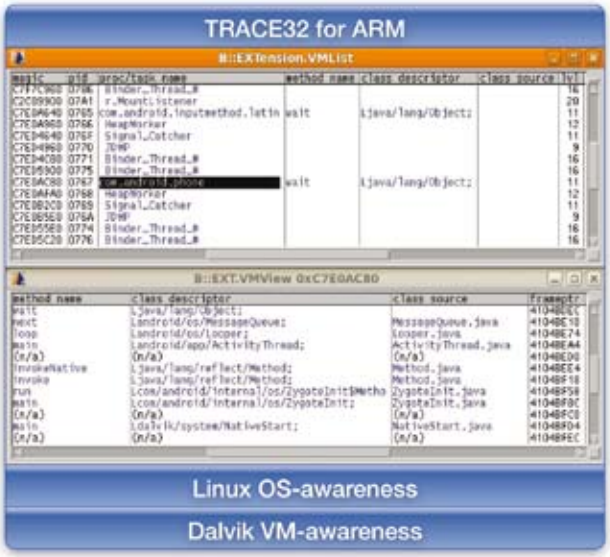

图 6：　为完成参考实现，Linux 操作系统识别插件和 Dalvik 虚拟机识别插件都必须装载到 TRACE32 中。

目标操作系统调试识别必须始终由 TRACE32 用户配置。已有一些适用于所有通用操作系统的配置实例。RTOS API 为专用操作系统提供扩展选择方案。

VM 调试识别插件是 TRACE32 软件的固定组件，适用于 J2ME CLDC。J2ME CDC 和 Kaffe 等虚拟机。所有其他虚拟机必须通过 VM API 进行单独的修改。可用于广受青睐的安卓 Dalvik VM 的配置也已经完成。

Lauterbach 提供适用于开源的操作系统和虚拟机的解决方案，使闭源虚拟机供应商也可以编写其产品的 TRACE32 虚拟机识别插件，并提供给其客户。

参考实现

为了全面调试基于 ARM 的安卓系统（从 Java 应用程序到 Linux 硬件驱动程序），TRACE32 需要集成以下扩展（参见图 6）：

- 自 1998 年起，Lauterbach 提供一种 Linux 操作系统识别插件。
- Dalvik 虚拟机识别插件，该插件可从 Lauterbach 主页下载。该插件只能配置用于所使用的平台。

www.lauterbach.com/vmandroid.html

现在可识别和列出正运行的所有 Java 应用程序（图 6 中的 EXTension.VMList），并分析和查看所选 Java 应用的 VM 栈（图 6 中的 EXTension.VMView）。

下一步是计划显示 VM 正在运行的源代码。开发目标显然是提供适用于 VM 应用的停止模式调试功能，以及提供现代调试器的其他所有功能。

新支持的 RTOS	
用于 ARM 的 DSP/BIOS	2011 第 2 季度
OSEK/ORTI SMP	2011 第 2 季度
用于 ARM 的 T-Kernel	可提供
用于 ARM 的 Windows Embedded Compact 7	可提供
用于 ARM 的 µC/OS-III	可提供

扩展与新的 RTOS 版本

- TRACE32 经改进后的脚本适用于 Timesys 嵌入式 Linux。
- OSEK/ORTI 现在也可以确信生成任务切换的 NEXUS 所属跟踪信息。这使得 TRACE32 可为 MPC55xx/MPC56xx 提供任务运行时意识功能，即使 NEXUS 生成的是无数据跟踪的信息。

下面的版本已改进或正计划改进：

- OSEck 4.0
- QNX 6.5.0
- 用于 ARM 的 Symbian^3
- 计划用于 Q1/2011 的 Symbian^4
- 用于 Atom™ 的 Windows CE6

串行跟踪端口应用增长

更快、更高、更强！这不仅是体育运动的座右铭，同时已成为微电子行业内的核心准则。越来越快的时钟速度和更大的进程并行化，使我们的处理器的处理速度在过去几十年内实现难以置信的提升，因此设计师们将这条座右铭用在跟踪信息传输领域亦不足为奇。

处理器必须通过跟踪接口传输其内部程序运行的详细信息，这促使跟踪接口技术行业不得不努力紧跟信息的快速增长步伐。对于大多数嵌入式系统开发商而言，如果无法获得重要的信息，很难想象将如何完成开发工作。因此他们尽一切可能提高跟踪接口的数据吞吐量。多年来，提高时钟频率和增大跟踪端口总线宽度是提高数据吞吐量的有效方法。

不过，这些措施都必须付出代价。更宽的跟踪端口需要占据更多宝贵的封装引脚，而且时钟脉冲频率提高时信号质量势必下降，因此需要对跟踪总线的所有信息进行补偿。Lauterbach 采用基于自动对焦技术的先进算法，因此完全能够保证无差错地记录高频跟踪信号。

随着处理器架构继续通过并行化实现提速和提升处理复杂性，跟踪接口正开始使用过去在其他领域应用的高速数据传输方法。高速串行数据传输方法已用于 SATA、光纤通道、PCI Express 接口和 USB 3.0（超高速 USB）中。极高的数据传输速率弥补了只有少数差分数据线的缺点。

在芯片上集成高速串行接口是非常昂贵的，并可能带来问题。例如，I/O Pad 必须以更高速度运行。但随着越来越多串口接口达到千兆赫级的传输频率，我们已积累了足够的经验，确保可解决串行跟踪端口应用中产生的大多数问题。

2008 年，ARM 公司实现了这项技术，使高速串行跟踪端口 HSSTP 用于短程跟踪数据传输。这很快引起 AMCC（使用 Titan）、飞思卡尔（使用 QorIQ 处理器 P4040 和 P4080）及迈威（使用 SETM3）等公司的相应。

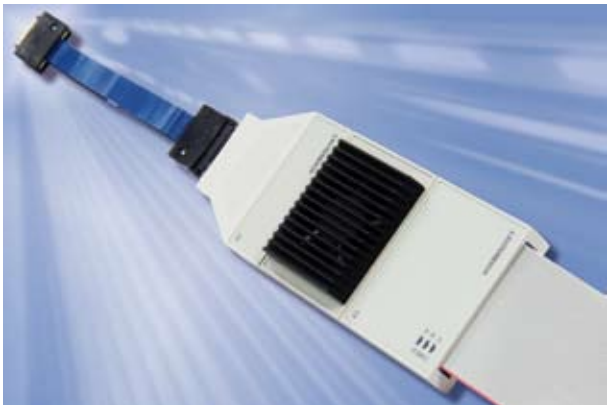

图 7：通过修改以上固件和软件，通用硬件可支持各种串行跟踪接口协议。

Lauterbach 在 2008 年设计了一款用于串行跟踪的硬件接口。根据 Aurora 协议开发了一款通用的预处理器。只需更改固件和软件即可记录任何可选通讯协议。这表明，我们的系统完全能够适应以后串行跟踪协议的不断变化。

支持的串行跟踪端口		
AMCC	APM83290 程序流	2009 年
ARM-HSSTP	ETMv3, PTM, CoreSight ETMv3, CoreSight PTM 程序流， 程序流与 Context-ID	2008 年
飞思卡尔	NEXUS QorIQ P4040 and P4080 分支跟踪与所有权跟踪信息、数据写消息	2010 年
迈威-SETM3	CoreSight ETMv3 程序流， 程序流与 Context-ID	2009 年

为实时流提供更高数据传输率

“实时流”是跟踪数据必须在记录时同时传输给主机，然后，主机收到跟踪数据后立即进行分析。这要求将大量数据从跟踪工具实时传输到主机上，用于 CPU 密集型应用程序和多核系统中时尤其如此。为了使 TRACE32 可适用于这些应用，需要使用跟踪工具 PowerTrace II 在跟踪数据被传输给主机之前对数据进行压缩。TRACE32 软件从 2010 年 12 月起就支持该功能。

*实时流*用于支持 ARM 跟踪协议 ETMv3 和 PTM。

硬件压缩

向主机传输数据的最高速率不够，仍然是*实时数据流*所面临的瓶颈。即使是在跟踪工具与主机之间采用点对点 GB 以太网接口，网络最大传输速率仍然只能达到 500M 比特 / 秒。最高传输速率必须足以无损耗地将跟踪数据从跟踪端口传输到主机上。

为了能够估计实际要传输的数据量，重要的是要了解*实时流*的条件：

1. *实时流*的主要应用是代码覆盖率和运行时间的测量。对于这两种功能，由于只需要输出程序跟踪信息，因此传输速率是完全够用的。为了得到一个非常准确的运行时间测量，可以启用精确的周期跟踪。
2. 要准确估计所需的数据传输速率，你只需考虑跟踪端口的平均负载即可。跟踪端口的最大负载采用 PowerTrace II 跟踪，它可以看作是一个很大的 FIFO（最大 4GB）。图 8 示意了跟踪端口用于 Cortex 内核时的平均 / 最大载荷。有效负载由 Cortex 内核上运行的应用程序最终决定。

通过在 PowerTrace II 中采取基于 FPGA 的硬件压缩措施，跟踪接口向主机传输数据的最大传输速率可提高到 3.2G 比特 / 秒。

纯长期跟踪

在*实时流*传输期间，如果跟踪数据已被分析并保存到硬盘中，则 Lauterbach 认为这是一次*长期跟踪*。

为了提供支持其他协议（例如 Nexus）的*长期跟踪*功能，Lauterbach 现正在将纯数据流保存到硬盘上而不同时进行分析。这表示，对于 64 位的主机操作系统，跟踪记录最多可达到 1 万亿帧。

关于*实时流*和*长期跟踪*的详细信息，请访问 Lauterbach 主页：
www.lauterbach.com/tracesinks.html

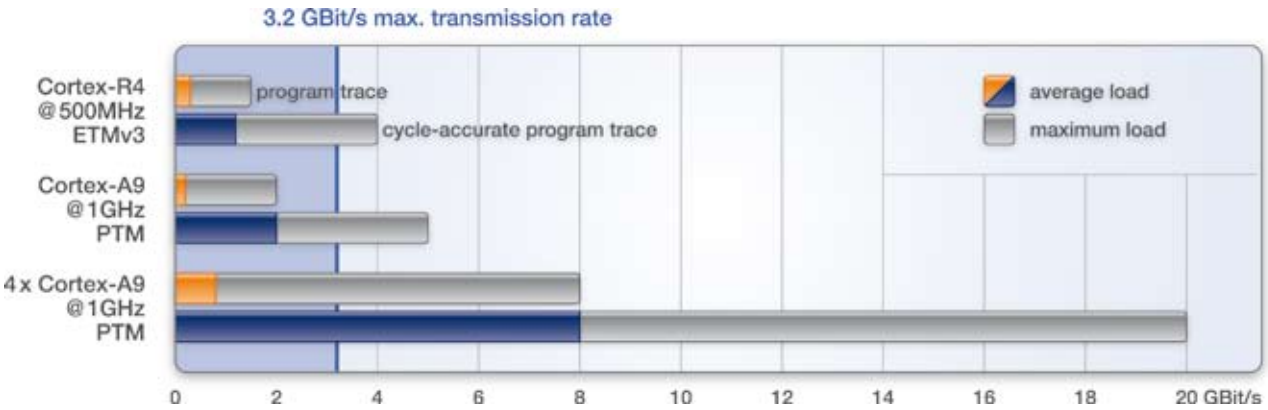

图 8：3.2G 比特/秒的传输速率通常足以支持在记录数据的同时将跟踪信息传输给主机。

使用 CombiProbe 进行能量分析

TRACE32 CombiProbe 现在也可以用于测量应用程序能耗。

可进行以下分析：

- 电流 / 电压分析界面最多可显示与处理器上运行代码直接有关的三个测量点。
- 可从各单独功能函数分析整个系统的能量消耗。

程序的哪一部分消耗能量最多？程序改进对嵌入式系统的能量需求有何影响？其中有些问题现在可以使用 CombiProbe 加以解决。

要确定程序每个点的能量消耗，必须收集下列测量数据：

- 程序流必须通过处理器的跟踪端口输出。
- 在目标硬件上的合适测量点测量电流与电压数据。

通过连接 *TRACE32 模拟探头*到 CombiProbe 上，可测量电压和电流值最多可达到三个应用。

CombiProbe

CombiProbe 是一根调试电缆，同时包含一个 128MB 的跟踪存储器。CombiProbe 专用于带 4 位跟踪端口的处理器。程序流记录功能目前支持以下跟踪协议：

- 连续模式下的 ARM-ETMv3 (ARM)
- 用于 PIC32 的 IFLOW 跟踪 (Microchip)
- 用于 X-GOLD102 和 X-GOLD110（英飞凌）的MCDS 跟踪

www.lauterbach.com/cobstm.html

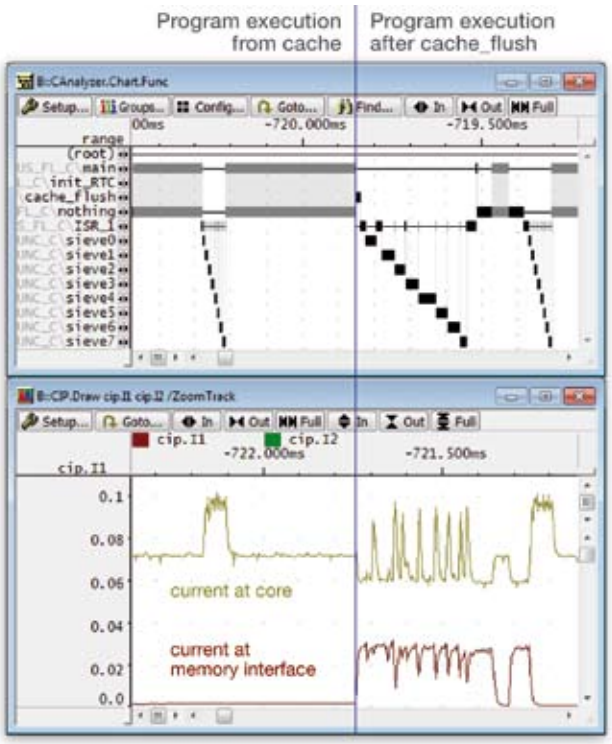

图 9：程序段未从缓存区内运行时，需要更大的电流。

由于所有的测量数据都通过 CombiProbe 的统一定时器记录时间，因此可以非常容易和快速地找出已执行程序代码、功率消耗及系统电压波形之间的之间的直接联系。

从图 9 中可以得知，当程序段从外部存储器运行而不是从缓存区运行时，不仅需要更长的处理时间，而且外部存储器消耗更多的能量。

图 10 为能量消耗统计分析数据。

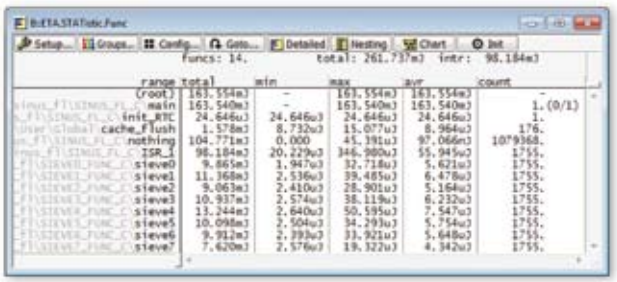

图 10：各函数的最小、最大、平均能量消耗统计。

SMP 分析

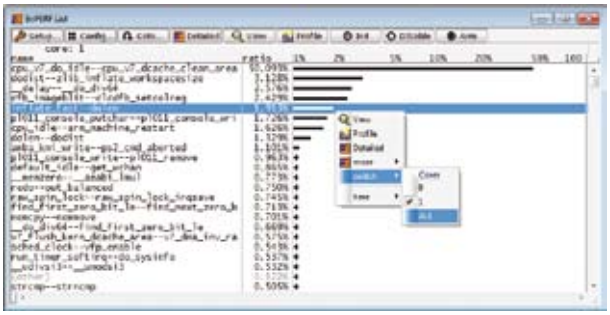

图 11： 基于抽样的性能分析，示意了各单独代码段与总运行时间之比（表示为百分比）。结果即可显示为 SMP 系统的各内核（这里是内核 1）的运行时间，也可显示为所有内核总运行时间。

基于抽样的分析方法于 2010 年全面改进。其中重要的创新措施包括采用了新的测量思路、自校准采样率及 SMP 系统扩展。

面向函数的 SMP 分析

从 2010 年 10 月起，Lauterbach 工具可收集 SMP 系统概要数据。为了进行函数层次的数据分析，TRACE32 循环读取各个内核的程序计数器，并将读取值保存在一个数据库中。然后，分析结果即可显示为 SMP 系统的各内核的运行时间，也可显示为所有内核总运行时间。

大多数芯片都能够在处理器运行的同时允许读取程序计数器，目前已经支持在以下架构下的实时测量：

- **ARM / Cortex:** ARM11 MPCore, Cortex-A5 MPCore, Cortex-A9 MPCore, Cortex-A15 MPCore
- **MIPS32:** MIPS34K, MIPS1004K
- **MIPS64:** Broadcom BCM7420

如果芯片的调试逻辑不允许非侵入性读取这些信息，则需要定期暂停各内核运行，以便读取相关信息。

全球机构

- 美国
- 德国
- 法国
- 英国
- 意大利
- 中国
- 日本

在其他国家由经验丰富的合作商提供服务

基于抽样的分析

图 1：基于抽样的分析，定期读取程序计数器或变量，包括当前任务的 ID。根据所读取的信息，将功能或任务的运行时间与总运行时间之比表示为百分值。

对称多道处理（SMP）

一个由相同内核组成的多核芯片可配置为一个 SMP 系统。在程序运行时（而不是运行之前），由一个 SMP 操作系统将待处理的进程（任务）动态分配到各内核上。要调试 SMP 系统，只需打开一个 TRACE 32 实例，即可监控所有的内核。

SMP 任务分析

要实现一个任务分析功能，必须定期从存储器内读取各内核的任务 ID。大多数芯片都允许物理存储器在运行程序时进行读取操作。如果芯片调试逻辑支持该功能，则可进行实时测量。

- **ARM/Cortex:** ARM11 MPCore, Cortex-A5 MPCore, Cortex-A9 MPCore, Cortex-A15 MPCore
- **Power Architecture 架构：** MPC8641D, MPC8572, QorIQ

否则，必须定期暂停各内核运行，以便读取所需的数据。

劳特巴赫扩大在华销售支持网络

----设立深圳（广东）新办事处（华南）

劳特巴赫的客户群有近 30% 来自以广东省为代表的华南地区。中国沿深圳 — 广州走廊设立了大量的高科技设计中心，劳特巴赫需要缩短与客户的距离。深圳办事处的主要任务是支持设计工程师对通信、汽车和医疗应用中涉及的复杂的多芯实施项目进行调试。

联系地址：
中华人民共和国深圳市南山区桃园路 1 号西海明珠大厦 1406
邮编：518052
电话：+86 (0) 755-8621 0671
传真：+86 (0) 755-862 10675

让我们知道您的最新动态

如果您的地址已更改，或者如果你不再希望留在我们的收件人名单中，请给我们发送邮件：

info_cn@lauterbach.com.