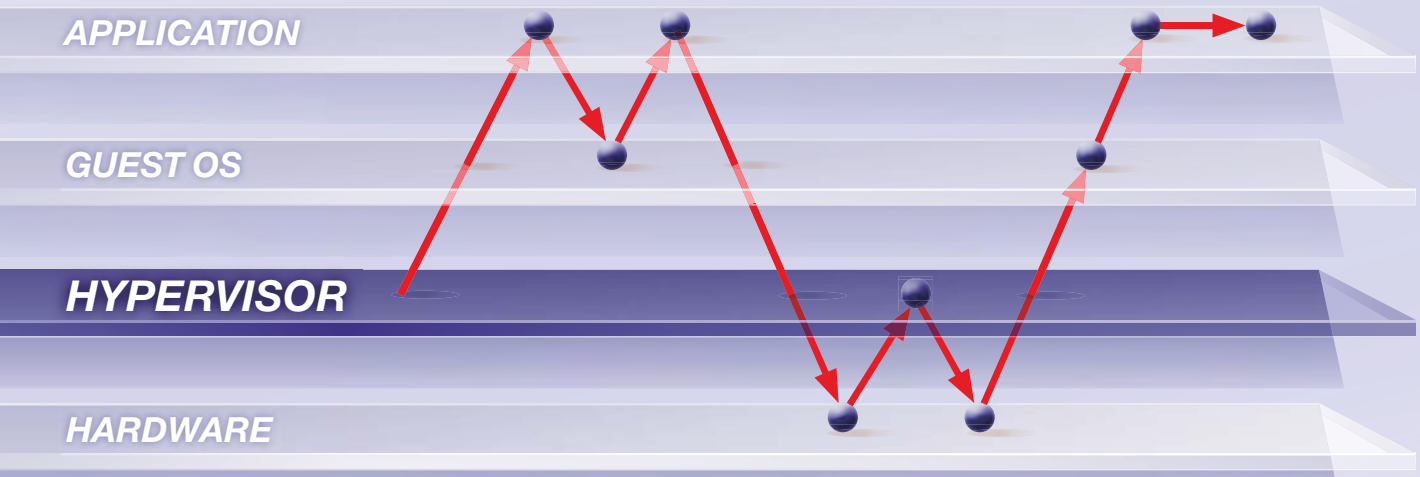
NEWS 2017

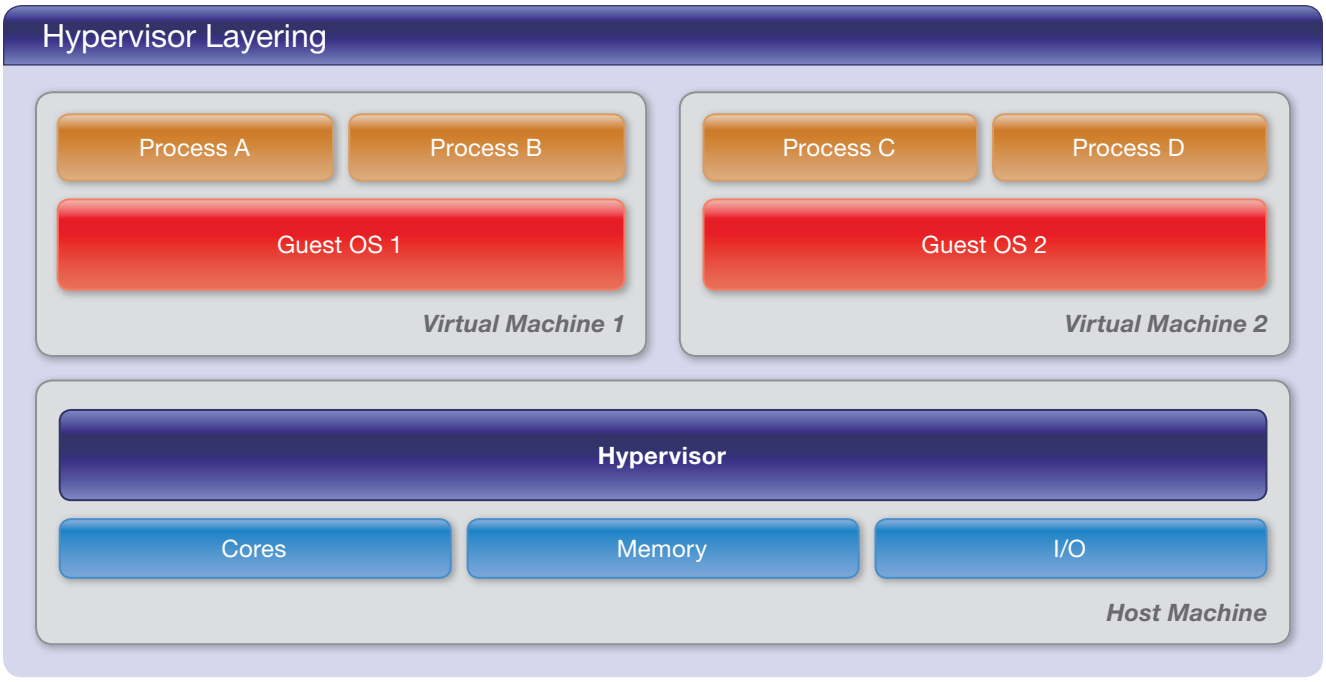


Seamless debugging through all software layers

内容

LAUTERBACH
DEVELOPMENT TOOLS

Hypervisor 调试



2017 年 4 月，劳特巴赫将提供运行性能更好的新管理程序 Hypervisor 的软件支持。本文介绍的参考案例为具有两个 Linux 客户端的 Xen 虚拟机管理程序在 LeMaker（Cortex-A53）的 HiKey 板上运行。

嵌入式系统中的虚拟化 Virtualization

虚拟化概念允许在单个硬件平台上并行多个操作系统。目前，在嵌入式系统中虚拟化的应用越来越多。例如，在汽车的驾驶舱中，由 AUTOSAR 操作系统监视的实时应用程序与基于 Android 的用户界面在相同硬件平台上并行运行。作为虚拟化的核心，虚拟化管理程序来确保一切都能够可靠高效地工作。

Hypervisor（也称为虚拟机监视器）是实现以下两项任务的软件层：

1. 启动和管理虚拟机（VM）。
2. 将虚拟机的物理硬件资源虚拟化。

在 VM 上运行的操作系统被称为子操作系统 (GUEST OS)。Guest OS 对虚拟化硬件资源的所有访问由 Hypervisor 映射到物理资源。

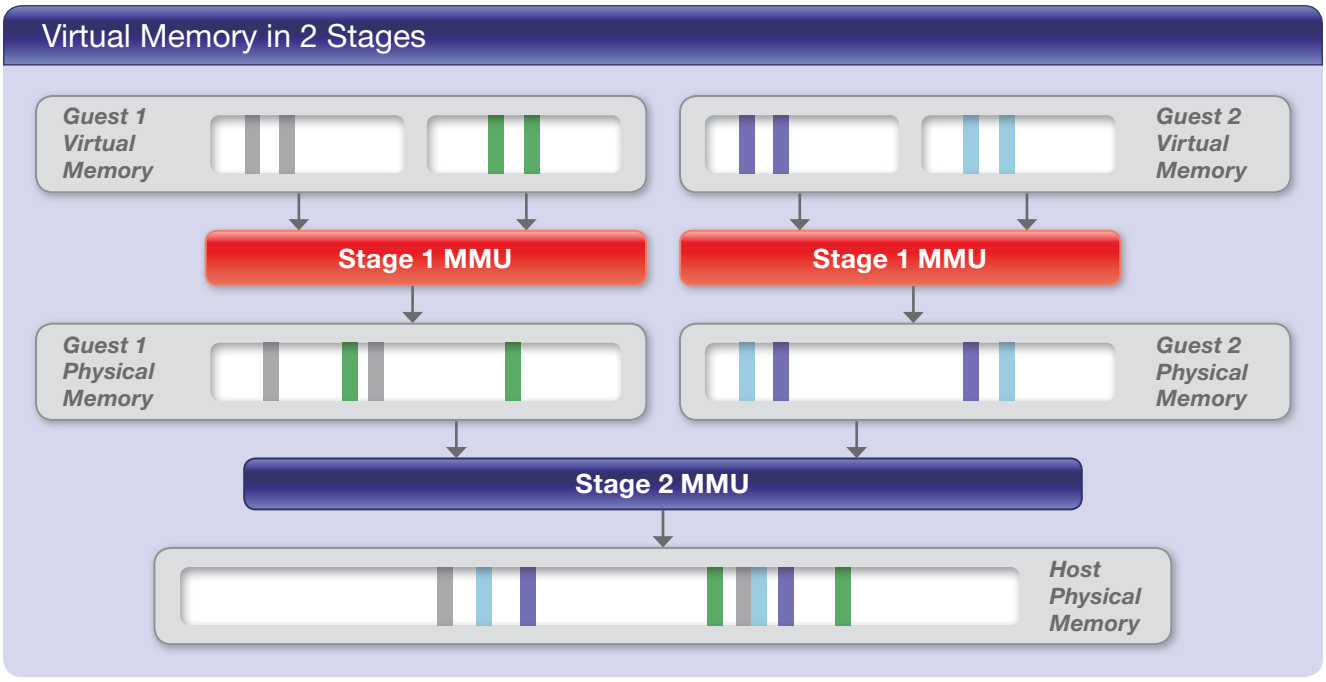
CPU 虚拟化对于调试很重要。每个虚拟机分配为一个或多个虚拟 CPU（vCPUs）。vCPUs 的数量不必与硬件平台上可用的实际 CPU 内核数量相同。

内存虚拟化同样重要。VMs 不会直接访问实际的物理内存，但将访问虚拟化的子系统物理内存。Hypervisor 管理每个 VM 的单独页表，以控制对物理内存的访问。由于至少在像 Linux 这样的操作系统上，应用程序进程无论如何都使用虚拟地址，所以调试器必须处理两个阶段的地址转换：

- 子系统的虚拟内存到子系统的物理内存
- 子系统物理内存到主机的物理内存

请参考 “Virtual Memory in 2 Stages” 图：

- The Stage 1 MMUs 映射信息由其每个子操作系统的页表处理。
- The Stage 2 MMUs 映射信息由 Hypervisor 系统的页表处理。



调试概念的进一步扩展

2016 年，为了满足 Hypervisor 的调试需求，劳特巴赫系统地扩展了 TRACE32 产品的支持。内容如下：

- Machine ID 已添加到 TRACE32 命令语法中。通过 Machine ID 调试器能够访问活动的 VM 以及所有非活动 VMs 的上下文。当其中一个处理器内核已分配给某个 VM 执行程序时，该 VM 就被认为是活动的。
- 使用新的 hypervisor-awareness，调试器可以调试和可视化 hypervisor 的各个 VM。
- 可以支持同时调试多个不同的操作系统。
- 现在调试器可以访问所有非活动子系统的页表，而不是像以前那样只能访问活动子系统页表。

所有扩展的最重要目标是支持无缝调试整个系统。这意味着，当系统停在某断点处时，您可以检查和修改每个进程、所有 VM 的当前状态，以及 hypervisor 和真实硬件的当前状态。此外，您可以在代码中的任何位置设置程序断点。

近 20 年来，劳特巴赫提供的无限调试功能（对于 Linux 这样的操作系统的支持）构成了所有扩展实现的起点。

因此，对最重要的操作系统调试概念简要总结如下：

进程运行在操作系统的专有虚拟地址空间。TRACE32 OS-awareness 和 TRACE32 MMU 支持允许用户跨进程边界的无缝调试：

- 借助 Space ID，可以直接访问每个进程的虚拟地址空间。
- 借助TASK选项,可以显示当前寄存器集和堆栈结构。

Machine ID

如果操作系统在虚拟机上运行，这个概念如何扩展？

1. 首先，必须唯一地标识每个虚拟机。为此，TRACE32 为每个 VM 分配一个编号，即机器 ID。虚拟机管理程序的机器 ID 为 0。正如空间 ID 用于标识进程的虚拟地址空间一样,机器 ID 用于标识 VM 的专用地址空间。
2. 为了显示进程的寄存器集和堆栈帧，调试器必须知道进程在哪个 VM 上和哪个子操作系统上运行。为达到此目的，引入了 MACHINE 选项。

TRACE32 Commands

Traditional OS-Aware Debugging	`Data.dump`	`<space_id>:<virtual_address>`
	`Data.LOAD.Elf <file>`	`<space_id>:<virtual_address>`
	`Register.view`	`/TASK <process_name>`
	`Frame.view`	`/TASK <process_name>`
< NEW > Hypervisor Debugging	`Data.dump`	**`<machine_id>:::`**`<space_id>::<virtual_address>`
	`Data.LOAD.Elf <file>`	**`<machine_id>:::`**`<space_id>::<virtual_address>`
	`Register.view`	**`/MACHINE <machine_id>`** `/TASK <process_name>`
	`Frame.view`	**`/MACHINE <machine_id>`** `/TASK <process_name>`

这两个扩展足以允许调试器访问跨进程边界的所有信息。上图的 “TRACE32 Commands” 概述了 hypervisor 调试扩展的与传统的用于操作系统解析的 TRACE32 命令的语法比较。

Hypervisor Awareness

与 OS-awareness 功能类似，现在出现了 hypervisor-awareness 功能。此功能为调试器提供在硬件平台上运行的 hypervisor 的所有信息。然而，hypervisor-awareness 需要加载 hypervisor 的调试符号。调试器可以创建所有子系统的概述。下图的参考案例的 “Guest-List” 屏幕截图 - Xen，Cortex-A53 - 显示以下信息：

- VM IDs 和 VM 状态，每个 VM 的 vCPUs 数
- Stage 2 页表（vttb）的起始地址

特定 hypervisor 的解析由劳特巴赫创建并提供给其客户。所有当前支持的 hypervisor 的概述显示在第 5 页的 “Currently Supported Hypervisors” 表中。

Guest List

B::EXTension.DOMain

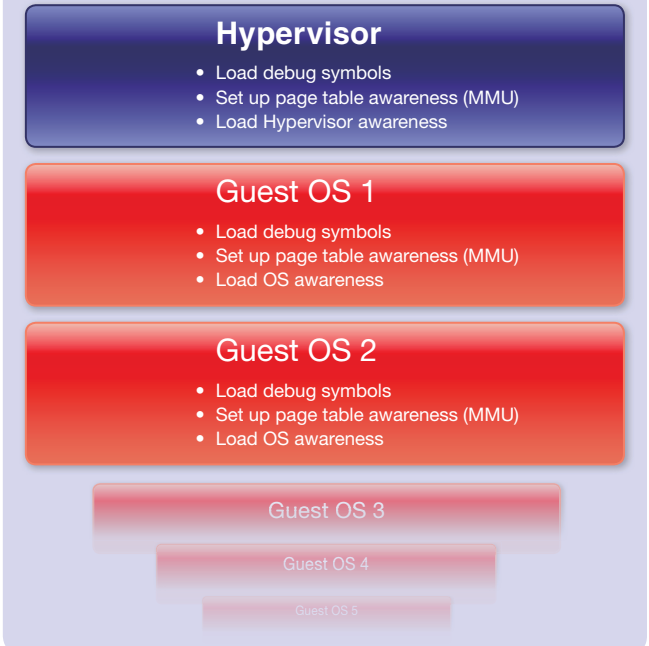
magic	id	mid	mem	nb_vcpus	vttb	state
0000800010078000	0	3	1536M	8	0003000090044000	blocked
000080000FF31000	1	1	1024M	8	000100008708A000	blocked
00008000070AE000	2	2	512M	8	0002000090076000	running

Debugger Configuration

扩展的调试概念如何影响 TRACE32 的调试？让我们先看看配置。配置 hypervisor 以及每个子操作系统都需要执行以下步骤：

1. 载入调试的符号表
2. 设置页表解析（MMU）
3. 载入 TRACE32 hypervisor-awareness 和 TRACE32 OS-awareness。

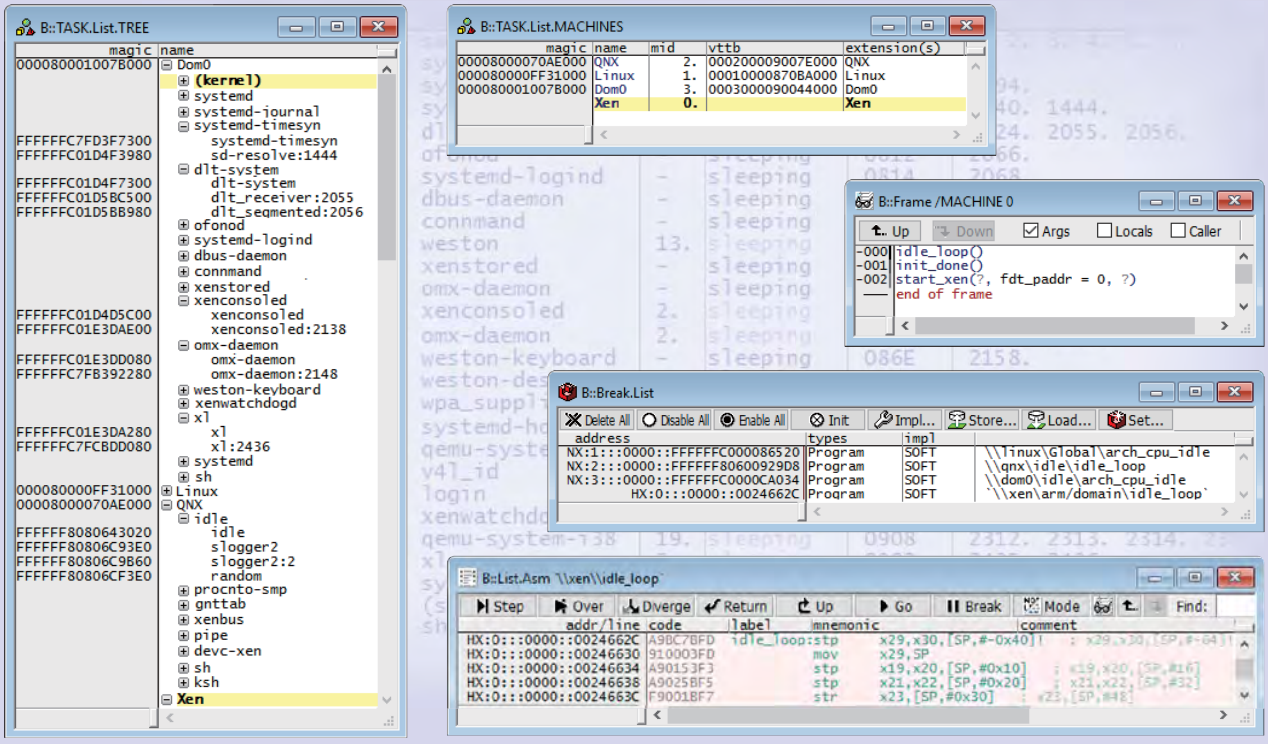
Debugger Configuration



Xen Hypervisor on Cortex-A53

Global Task List

Virtual Machine List



“ 调试器配置 ” 图显示出各个配置步骤的概述。

调试过程

调试器的操作必须经常解决矛盾的要求。一个用户组需要简单直观的操作，而另一个用户组需要最大的灵活性和完整的脚本功能。让我们先来看看直观的操作。基本思路实际上很简单：如果调试器在断点处停止，则 GUI 触发断点的应用程序进程。

如果您对不同的应用程序进程感兴趣，那么只需打开 TRACE32 全局任务列表即可。在整个系统上执行的所有任务都列在那里。您可以通过双击任务来选择要在 GUI 中显示的任务。全局任务列表还提供了一种为特定任务设置程序断点的简单方法。由于调试符号与加载 .elf 文件时的机器 ID 和空间 ID 相关联，因此调试时可以根据常用名称来寻址函数和变量。

使用 TRACE32 命令可以获得最大的灵活性和完整的脚本功能。扩展这些命令的语法如上所述。

小结

由于劳特巴赫公司已经系统地将众所周知的用于 OS-aware 的概念扩展到 hypervisor 调试，经少量练习后，TRACE32 用户将很容易开始 hypervisor 的调试。

Currently Supported Hypervisors

KVM	Wind River Hypervisor 2.x
VxWorks 653 3.x	Xen
	(*more to follow*)

英特尔® x86/x64 - 工具升级

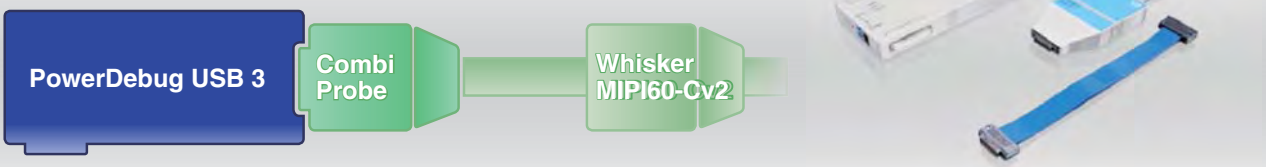
今年 1 月，劳特巴赫推出了新的 CombiProbe Whisker MIPI60-Cv2。TRACE32 CombiProbe 和 TRACE32 QuadProbe 现在可为 Converged Intel® MIPI60 连接器提供相同的调试功能：

- 标准 JTAG，Intel® debug hooks，具备 Pmode 和 I2C 总线
- 合并调试端口（两个 JTAG chains）
- 英特尔 ® Survivability（threshold，slew rate，...）

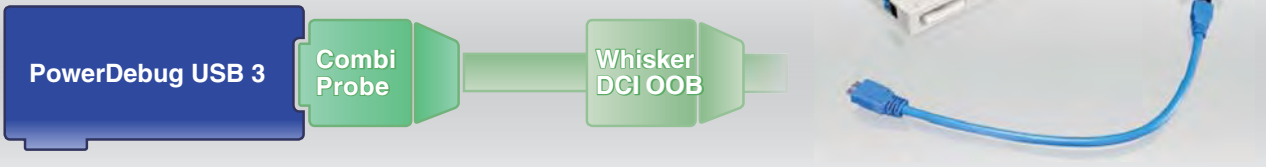
但是，这些调试工具具有不同的应用领域。TRACE32 QuadProbe 是专为服务器处理器设计的，是一种专用的调试工具，支持对具有多达四个调试接口的目标板上的数百个线程进行 SMP 调试。带有 MIPI60-Cv2 Whisker 的 TRACE32 CombiProbe 专为客户端和移动设备处理器设计，可以捕获和评估系统跟踪数据以及其增强的调试功能。跟踪功能包括支持一个带有标称带宽的 4 位及一个 8 位跟踪端口。

带有 DCI OOB Whisker 的 TRACE32 CombiProbe 专门用于调试和跟踪没有专用调试接口的平台或器件。如果芯片包含一个 DCI 管理器，目标和调试器可以直接通过 USB3 接口交换调试和跟踪信息。用于交换信息的 DCI 协议支持标准 JTAG 和 Intel® debug hooks 以及用于记录系统跟踪情况的跟踪信息。

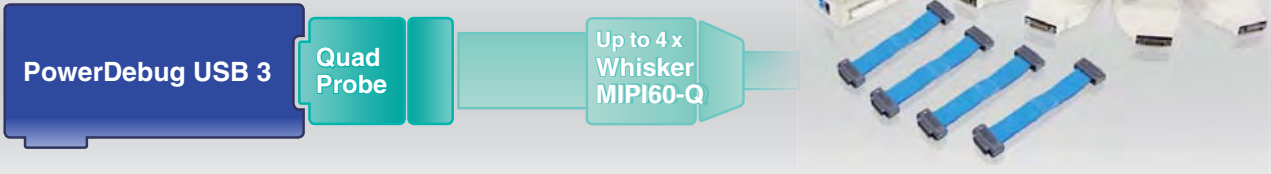
Debugger and System Trace for Intel® Converged MIPI60 Connector
(devices and client applications)



Debugger and System Trace for USB3 Connector
(all applications)



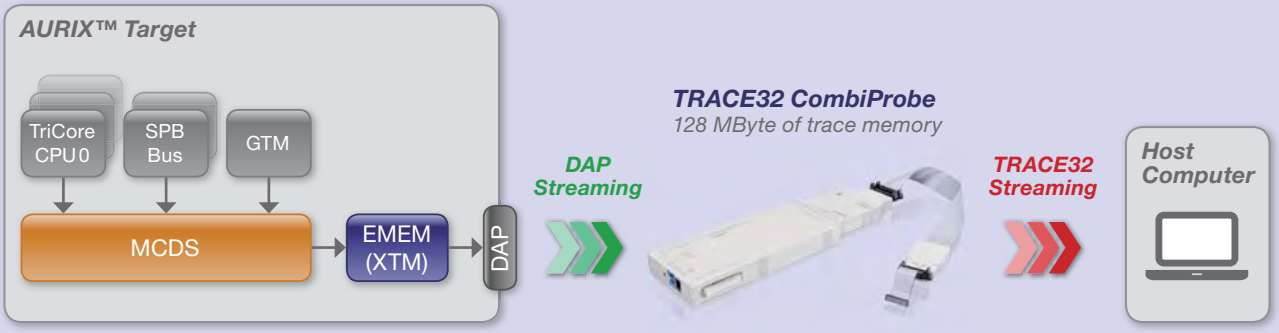
Debugger for Intel® Converged MIPI60 Connector
(server)



LAUTERBACH DEVELOPMENT TOOLS

TRACE32 CombiProbe TriCore DAP

DAP Streaming and TRACE32 Streaming



劳特巴赫自 2016 年 10 月起为 Infineon 的 AURIX™系列提供新的 CombiProbe TriCore DAP。这意味着现在 TRACE32 可以为使用不具备 AGBT 接口的目标硬件的 AURIX 用户提供全面的运行时 (run-time) 分析。

DAP Streaming

CombiProbe 实现了一种名为 DAP Streaming 的新技术：片内跟踪存储器的内容在程序执行时被读取，并完全传输到 CombiProbe 的 128MB 跟踪存储器。为了实现跟踪，芯片必须提供高速调试接口，AURIX™ DAP 接口满足相应的要求：DAP 频率高达 160 MHz，数据宽度高达 30MB/s。尽管 DAP 带宽不足以传输整个程序流，但仍可执行大量分析：

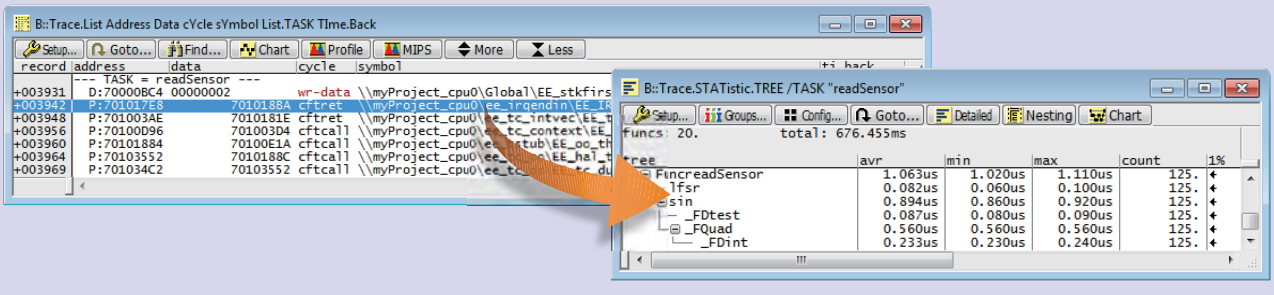
- 使用 Compact Function Trace（CFT）进行函数运行时测量。此跟踪是一种特殊的程序跟踪模式，其中仅为函数调用（cftcall）和函数返回（cftret）生成跟踪数据。下图的 “Compact Function Trace” 显示了 TRACE32 基于此跟踪数据计算的调用树和运行时详细信息的示例。
- 分析运行过程中所选变量的内容变化。
- tasks，ISRs 和 OS services 的运行时测量。

TRACE32 Streaming

如果 128MB 跟踪存储器 CombiProbe 不够大，不足以记录所有相关的跟踪数据，则可以将 DAP Streaming 与 TRACE32 Streaming 相组合。TRACE32 Streaming 在 CombiProbe 接收到跟踪数据后立即传输到计算机主机，并将其存储在文件中。这样可以记录连续的几个 TB 跟踪数据，为进行长期程序分析提供了可能性。

更多详细信息可登陆：
www.lauterbach.com/8467

Compact Function Trace



ARTI – AUTOSAR Run-Time Interface

新的 ARTI 标准为了满足当前汽车行业对于操作系统调试和跟踪的要求而特别设计。劳特巴赫作为 AUTOSAR 官方开发合作伙伴，积极参与了本标准的设计工作，计划于 2018 年年初发布该标准。

ORTI 标准自 2003 年以来一直在汽车行业中使用，帮助数千名开发人员调试和分析其 AUTOSAR 系统。但该标准正在优雅的老去，需要更新以满足当前行业的需求。

目标

软件开发的新方法、多核、多 ECU 系统以及对实时的关键系统验证日益增长的需求 - 所有这些因素在新标准中都需要考虑在内。有很多将被列入 ARTI 标准的新调试、跟踪和分析功能作为专有解决方案已经在行业中被使用，这意味着其功能性已得到证明。而目需要解决的是的是开发过程所使用各种工具之间的标准接口。下文将举例说明。

导出跟踪数据

自 2014 年以来，劳特巴赫一直与各种用于验证和优化汽车软件的工具制造商密切合作。TRACE32 导出其记录的实时跟踪数据，然后将其加载到外部工具中进行全面分析。那么目前缺少什么?

1. 构建工具创建的 ORTI 文件仅包含有关任务、操作系统服务和 ISR 的信息，但不包含有关何时启动或终止任务的信息，以及可运行的信息。在导出之前，必须在 TRACE32 中手动添加此缺失信息。
2. 跟踪数据导出没有标准化格式。这意味着外部工具必须能够读取专有的 TRACE32 格式。

新的 ARTI 标准将缩小这两个差距。构建工具提供的 ARTI 文件将包含有关所有 AUTOSAR 对象的信息，同时将跟踪数据的导出标准化。

结语

所有重要的工具制造商和工具使用者共同参与起草了新的 ARTI 标准，它必将和以前版本一样地成功以及长久地被应用。

如需了解更多信息，请登陆官网 www.lauterbach.cn
或发邮件至：info_cn@lauterbach.com

LEADING through Technology