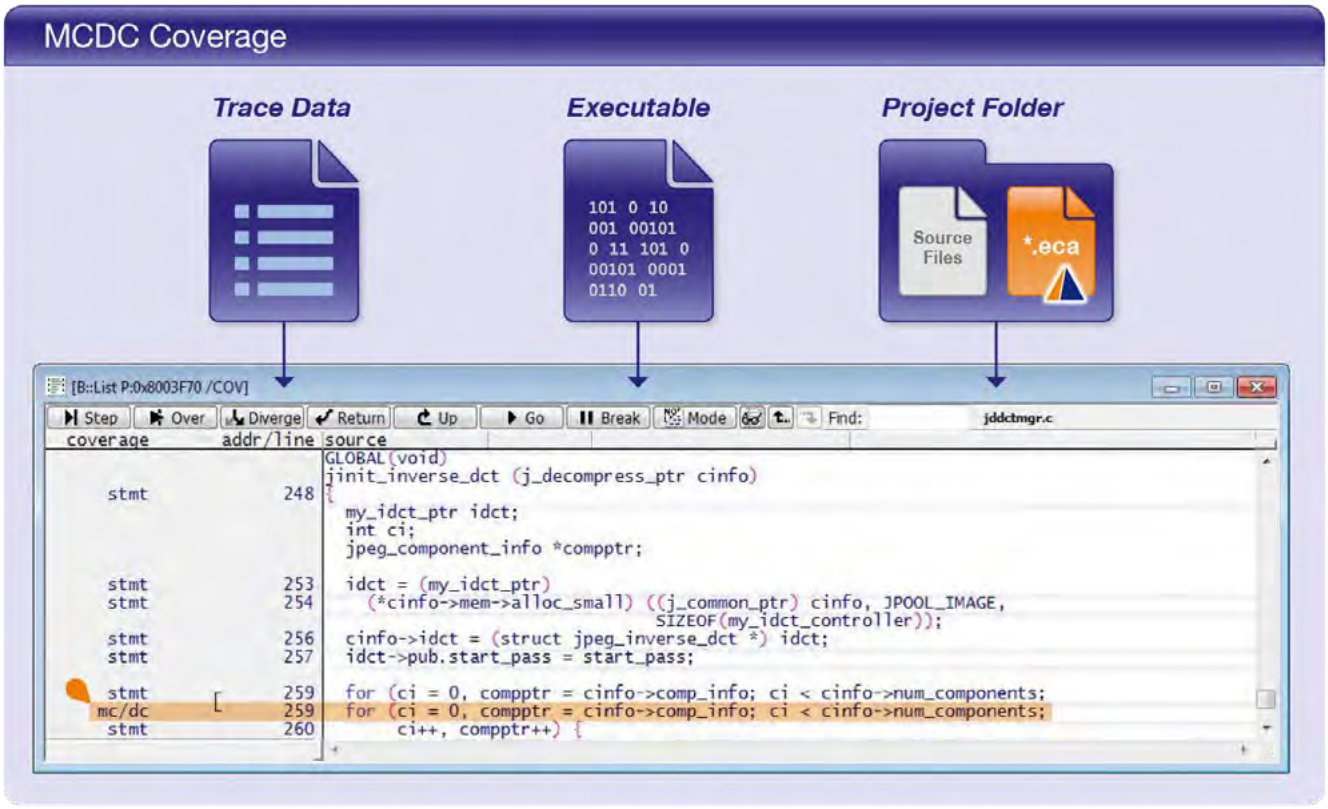
NEWS 2018

中文版

目录

LAUTERBACH
DEVELOPMENT TOOLS

基于跟踪的MCDC覆盖测试



劳特巴赫(Lauterbach)将于2018年3月推出基于跟踪的修正条件/判定(MC/DC)覆盖测试功能。TRACE32现已支持在源代码层级所有重要的代码覆盖测试。

依据DO-178C作出的定义 [3]

语句覆盖：程序中每一条语句至少被调用一次。

判定覆盖：程序中每一入口和出口点已至少被调用一次，并且程序中每一个判定所有可能的结果至少已呈现一次。

修正条件/判定覆盖：程序中每一个入口和出口点已被调用至少一次，程序中一个判定的每一个条件所有可能的结果至少已呈现一次，程序中每一个判定所有可能的结果至少已呈现一次，并且一个判定中的每一个条件已显示出对该判定的结果具有独立的影响。一个条件通过以下方式显示出对一个判定的结果具有独立的影响：(1) 只改变该条件，同时保持所有其他可能条件固定不变，或者 (2) 只改变该条件，同时保持所有可能影响该结果的其他可能条件固定不变。

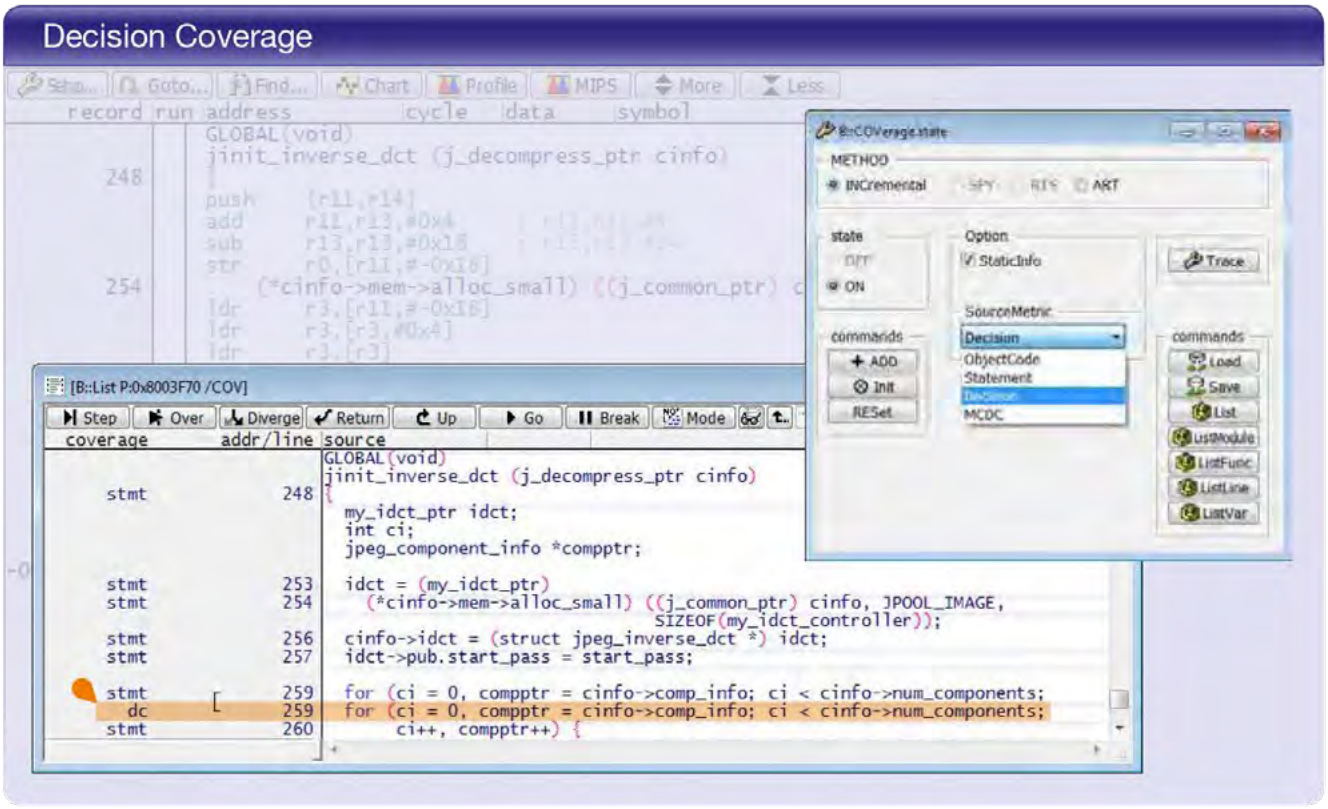
在关键性安全系统的开发过程中，使用代码覆盖测试证明该软件经过了全面、彻底的测试。诸如ISO 26262和DO-178C之类的软件开发标准规定了代码覆盖率验证是开发周期中必不可少的一部分。

基于跟踪的覆盖测试

作为实时跟踪工具的供应商，劳特巴赫提供基于跟踪的代码覆盖测量手段，不需要对目标代码进行任何插装。程序执行信息首先在目标代码级得到跟踪。这使得以下覆盖度量可以很容易地证明：

- **对象语句覆盖** 证明在测试过程中每一条汇编指令已被执行至少一次。
- **对象分支覆盖** 证明每一个条件跳转被执行至少一次或者一次也没有被执行。

目标代码级别的代码覆盖率分析一直都是所有TRACE32跟踪工具功能的组成部分。由于增加了语句与判定覆盖，劳特巴赫的工具从2017年2月起还提供源代码层级覆盖的证明（参见本页顶部的“判定覆盖”图



解）。许多客户现在还想将其TRACE32跟踪工具用于执行需求日益扩大的MCDC覆盖测量。

MCDC代码覆盖

过去，人们曾一直以为对象分支覆盖的证明在源代码层级足以替代MCDC覆盖测量。然而，在航空业，Commercial Aviation Safety Team (CAST) 表明了与此截然相反的观点（参见[1]）。

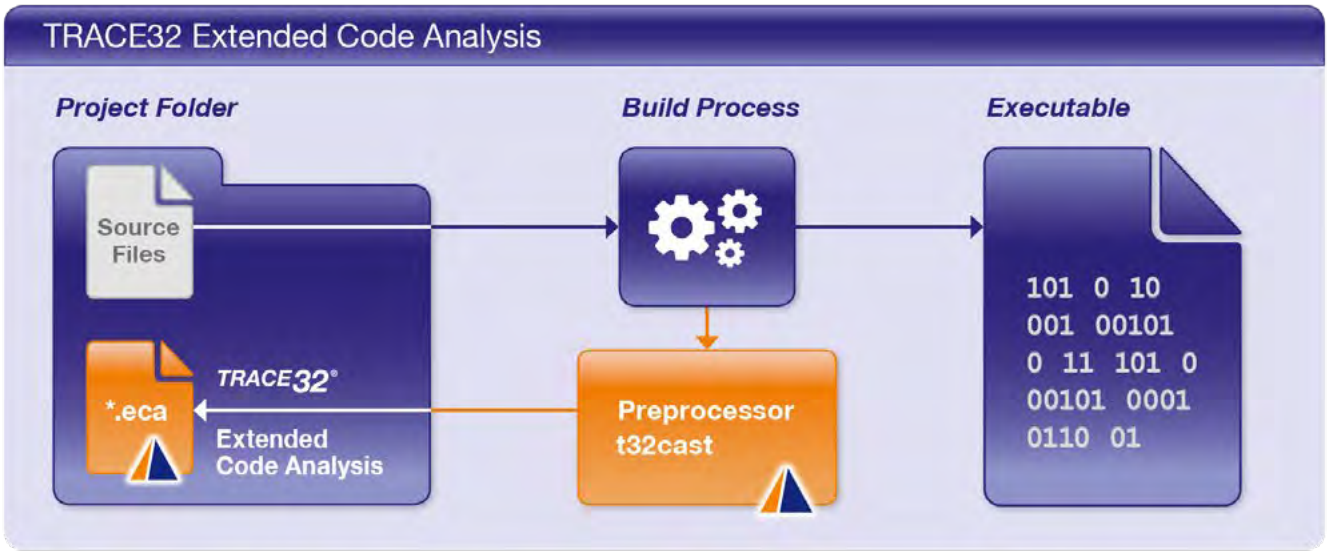
当前，大多数人依赖源代码插装以便能够收集MCDC覆盖数据。劳特巴赫工程师们为自己设定的目标是，提供MCDC覆盖测试而无需对目标代码进行任何改变。他们研究了该领域现有的许多白皮书和出版物。每一种方式都倡导其自身所采取的途径，不过就整体而言，有一些基本的相似之处：

1. 为了确保能够根据跟踪数据证明MCDC覆盖，源代码内一个判定的结构和位置均必须已知。
2. 同时，源代码中的每一个条件均必须在目标代码层级由一个条件跳转或条件指令来表示，而非使用该条件的算术表示法。
3. 在进行MCDC覆盖分析时，目标代码内一个判定的结构和位置必须已知。

为了不进行代码插装而实施MCDC覆盖测量，根据#1要求，必需有关源代码结构的附加信息。目前，这并非是编译器所生成调试信息的组成部分。此外，还必须确保编译器生成目标代码的方式符合#2要求。

AdaCore

如果执行MCDC覆盖分析的软件工具和代码编译器来自同一个供应商，则可以通过跟踪记录很容易地证明MCDC覆盖。该编译器可以将源代码内每个判定的结构与位置相关的必要信息包含在调试信息内。AdaCore公司（参见[2]）提供这样一种解决方案。此外，AdaCore还提供用于生成跟踪数据的目标仿真解决方案。



对于必须独立证明MCDC覆盖以便在目标硬件上最终实现的客户而言，AdaCore提供一种接口可以导入通过TRACE32记录的跟踪数据，以进行所要求的分析。

t32cast

从2018年3月起，劳特巴赫的客户还可以通过基于实时跟踪记录的TRACE32功能证明MCDC覆盖。劳特巴赫提供t32cast命令行工具用以实现该目的，该工具用于分析C/C++源代码。由此为每份源代码文件生成执行MCDC覆盖测量所必需的判定结构信息。因此必须对构建过程进行调整以便允许生成该信息（参见页面顶部的“TRACE32扩展代码分析”图解）。该t32cast命令行工具是不依赖于任何编译器，并且可以容易地整合进客户现有的编译环境。

一旦用户在TRACE32内启动MCDC覆盖测量，TRACE32会自动加载由t32cast工具生成的所有必要的.eca文件。随后，TRACE32借助调试信息能够将源代码层级的判定映射为目标代码。

然而，在该过程中，用户必须确保所选的编译器在目标代码层级通过条件跳转或条件指令表示源代码内的每个条件，例如通过禁用优化功能的做法。

结论

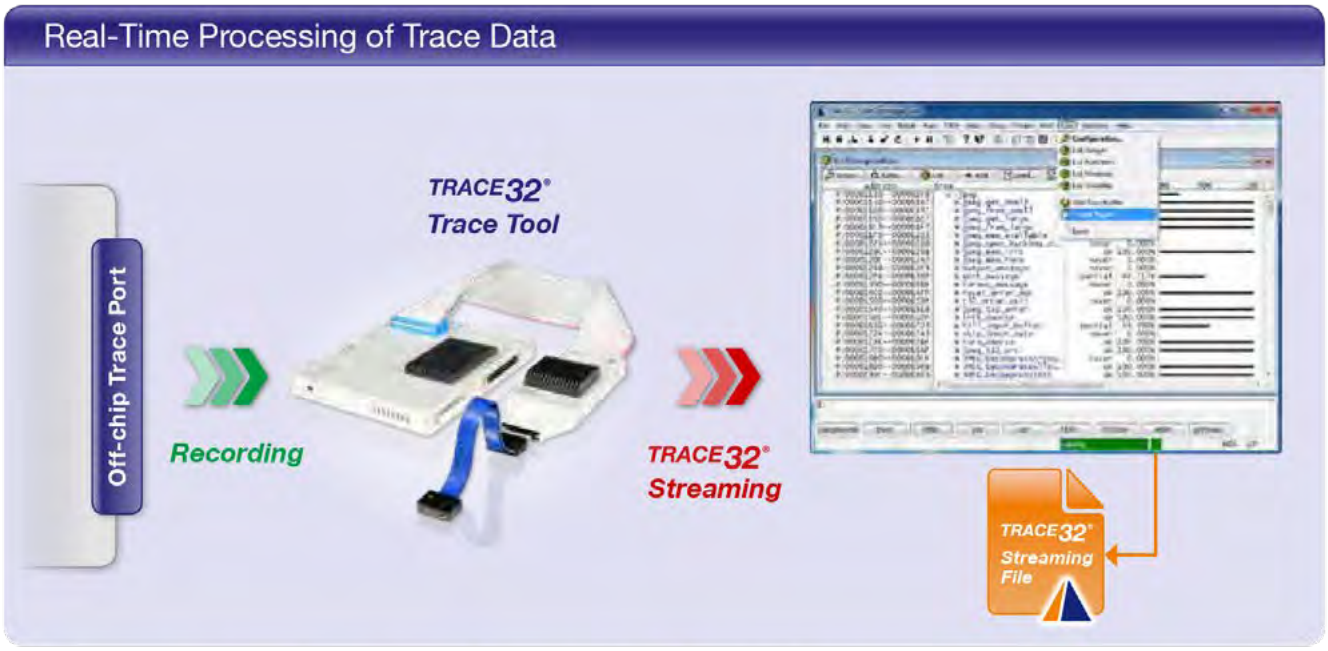
TRACE32 MCDC覆盖的使用可以独立于编译器和处理器架构。对于无法抛弃代码优化的用户而言，即使在实现关键性安全系统时，也无法减少为基于跟踪的MCDC覆盖而进行的优化。

原则上，日后应受高度推崇的做法是，编译器可以经过配置而在目标代码层级将判定仅编译为条件跳转或条件指令，与所选择的优化层级无关，并且为所有其余部分全部执行所选择的优化。

参考资料

[1] CAST-17 立场文件（2003年1月）。目标代码的结构覆盖
[2] Comar, C., Guitton, J., Hainque, O., & Quinot, T.（2012年5月）。修正判定条件覆盖(MCDC)和对象分支覆盖条件的形式化与对照。在嵌入式实时软件与系统(ERTS)会议上。
[3] RTCA Inc.（2011年12月）RTCA/DO-178C 机载系统和设备合格审定中的软件考虑

实时代码覆盖



实时数据解析(Real-time Profiling, RTS)是劳特巴赫赋予跟踪模式的名称，在该过程中跟踪数据被实时传送至主机并立即得到分析。这可以实现在屏幕上实时追踪代码覆盖分析的结果显示。2017年末，该RTS模式（可能自2009年起已被用于Arm-ETMv3）在更多跟踪协议上被应用。《支持RTS的跟踪协议》表显示所有当前被支持的跟踪协议的概述。

基本条件

以下基本条件适用于所有被支持的跟踪协议：

1. 为解码跟踪数据必须使用程序代码。如果在程序执行过程中，从内存读取代码的速度过于缓慢，则必须在开始实时分析之前将代码加载进TRACE32。这意味着实时分析仅能针对静态程序而执行。
2. 跟踪数据的实时分析仅在跟踪端口的平均数据速率不超过主计算机最大数据传输率的情况下才起作用。在现行TRACE32跟踪工具配备一个USB3接口的情况下，对主计算机的最大数据传输率约为180 MByte/s。该传输率是2009年时数据速率的三倍。
3. 跟踪数据的实时分析目前可以针对单核以及SMP（对称多处理器）多核跟踪数据流而进行。面向AMP（非对称多处理器）多核跟踪数据流的实现目前仍处在开发阶段。
4. 实时代码覆盖测量可以用于对象语句覆盖(Object Statement Coverage)和对象分支覆盖(Object Branch Coverage)的度量，以及用于语句覆盖(Statement Coverage)和判定覆盖(Decision Coverage)。

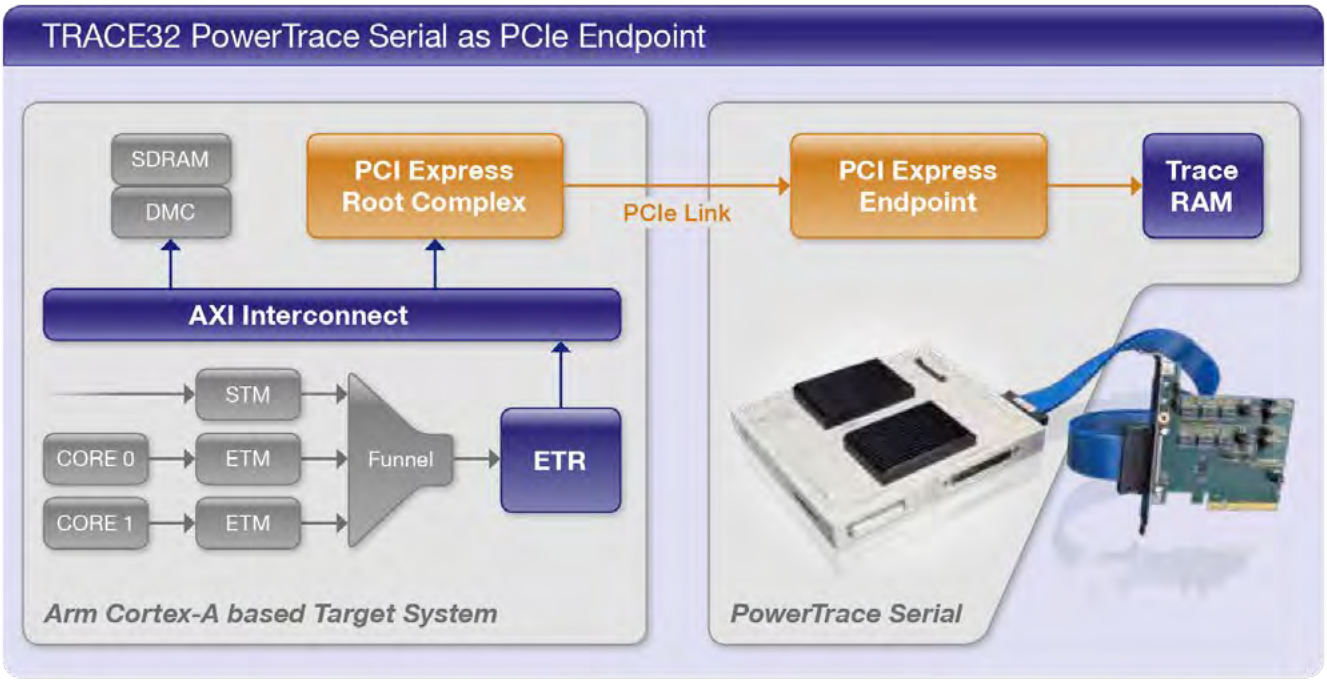
一般情况下，跟踪数据在经过分析之后便不再需要。不过，TRACE32在分析过程中提供将跟踪数据保存为所谓流文件(streaming file)的选项。这样，即使在完成代码覆盖测量之后，也可以实现对跟踪数据进行详细再验证。正如同传统的TRACE32代码覆盖测量一样，RTS模式允许创建全面测试报告。

RTS – Supported Trace Protocols

- ETM v3 on Arm / Cortex®
- PTM on Arm / Cortex®
- ETM v4 on Arm / Cortex®
- MCDS on Infineon TriCore™
- Nexus on NXP MPC5xxx / STM SPC5xx
- Nexus on NXP PPC QorIQ®

others on request

通过PCI Express实现跟踪



当芯片配有PCIe接口时，就可以通过该接口利用外部跟踪工具对跟踪数据进行记录和分析。劳特巴赫的一些客户已经采用了该跟踪技术并将其应用于Arm Cortex-A®或基于 NXP Power Architecture 的 QorIQ®处理器。

跟踪工具作为一个PCIe端点

首先，让我们简要概述通过 PCI Express 进行跟踪的工作原理。作为跟踪工具，TRACE32 PowerTrace Serial被设计为能够作为一个PCIe端点而运行。为此，该跟踪工具必须连接至目标，并在软件（例如：引导装载程序）开始对端点进行枚举和配置之前就已经处于运行状态。在这种配置过程中，每个端点均在系统内存中分配有一段地址范围。这样，针对该端点的数据就可以直接写入该地址。

然后，目标系统的跟踪硬件组件必须配置为将跟踪数据写入刚被分配至TRACE32 PowerTrace Serial端点的该地址范围。这对于绝大多数处理器而言均可行。在此之后就可以开始跟踪。

Characteristics

Variable data rate
- Gen1 250 MByte/s per lane
- Gen2 500 MByte/s per lane
- Gen3 984 MByte/s per lane

Variable port width (1, 2, 4 or 8 lanes)

Full-size card adapter available,
mini-PCIe card adapter planned

4 GigaByte of trace memory,
trace decoding for all standard protocols

摘要

通过PCIe进行跟踪简便易行。对于未配备专用跟踪接口的目标系统而言，该技术为记录大量跟踪数据提供了一种卓越方法。在通过PCIe之类的外部接口进行跟踪和通过专用跟踪端口进行跟踪之间的主要差别可以归结如下：

- 仅在目标软件已配置 PCIe 根联合体 (Root Complex) 之后才能开始跟踪记录。这是分配给操作系统的一项任务。
- 同时，严格而言，通过PCIe进行跟踪不再具有“非侵入性”。它要求使用一部分系统内存。此外，这种跟踪会与其他端点争夺PCIs总线的带宽。

从Wind River向TRACE32的无缝迁移

从 2014 年起，Wind River 便不再提供 JTAG 调试器，越来越多现有用户转而使用TRACE32进行产品的维护与深入开发。在同Wind River密切合作的过程中，劳特巴赫已经系统地加强了TRACE32软件对Wind River产品的支持，并且对该系统的任何基础调整均以无缝方式实现，以确保用户可以像他们习惯的那样保持工作。

支持所有处理器

大多数转向使用TRACE32的客户使用的是Power Architecture™系列的处理器。劳特巴赫在1997年就已经将对该架构的支持纳入其产品系列。改用TRACE32的客户可以完全信赖已经过验证的调试解决方案。

脚本转换器

程序代码通常被编入目标闪存，然后才能开始调试。TRACE32为此使用了一个脚本。处理器配置寄存器的设置（特别是SDRAM设置）是该脚本的一个重要组成部分。Wind River曾在一个所谓的“Register Configuration File.”中定义这些必需的设置。劳特巴赫提供一种特殊的转换程序用于将该文件转换成为一个TRACE32脚本。在使用专业调试器多年之后，客户往往积累了用于各种目的的大量测试脚本。劳特巴赫提供脚本转换器可以将这些复杂的测试脚本转换到TRACE32环境中。

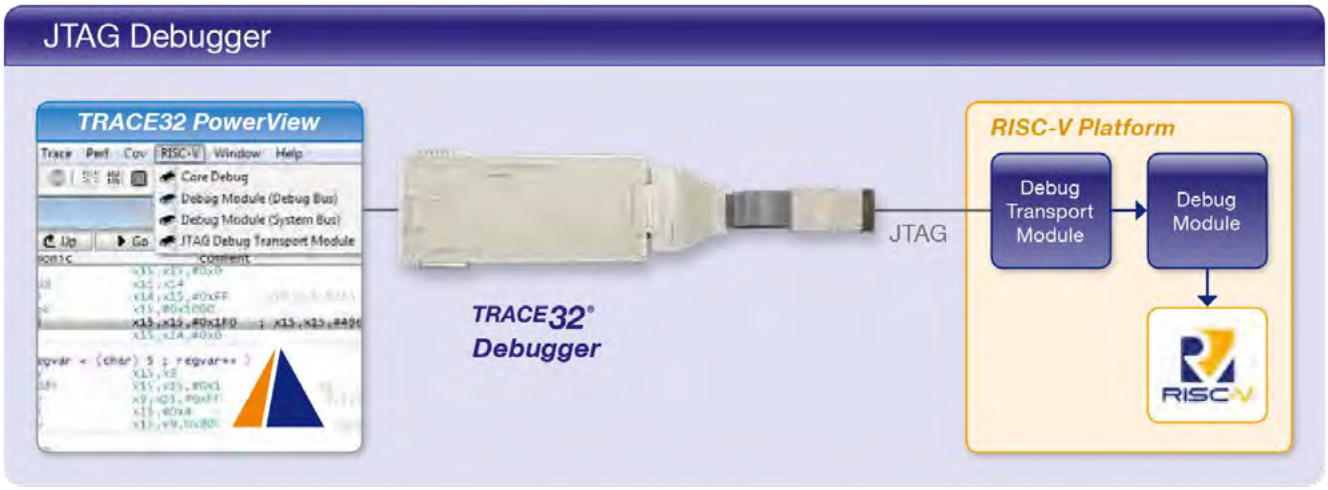
Wind River Workbench

在使用经过验证的相同系统多年之后改用一种新的调试器对于许多工程师而言是一个令人焦虑不安的时刻。为此，劳特巴赫在2015年就已经强化了其调试器作为TCF代理的作用。现在可以使用Wind River Workbench作为调试的集成开发环境(IDE)与作为调试后端TRACE32调试器一起使用。

支持所有Wind River产品

对于Wind River的软件产品，TRACE32都做了全面的支持，例如：Wind River VxWorks，Wind River Linux，或Wind River Hypervisor。劳特巴赫对Wind River产品的支持具有丰富的经验，TRACE32自1996年起就开始支持Wind River VxWorks的调试。对Wind River Linux的全面支持也于2000年推出。目前，TRACE32还对Wind River Workbench已不再支持的Wind River产品提供支持。这其中包括VxWorks 7、VxWorks 653 v3以及Wind River Hypervisor v3。

用于RISC-V的TRACE32调试器



劳特巴赫于2017年11月推出了新的RISC-V调试器。目前首批支持的芯片为SiFive的E31 Core Complex (32-bit)和 E51 (64-bit) Core Complex。

RISC-V是一种开放的指令集架构(ISA)，基于成熟的RISC原则并在RISC-V基金会(https://riscv.org)的指导下进行组织、规划和进一步开发。RISC-V最初为学术研究而建立，目前正在逐渐进入嵌入式市场，专业化的硬件调试器由此变得不可或缺。

劳特巴赫对RISC-V调试器的实现方式是基于开放源码规范“RISC-V External Debug Support”，该规范预计于2018年由RISC-V基金会采纳。该规范的目标是实现灵活的停机模式调试。一个RISC-V核心的每个硬件线程均可以直接从重置进行调试。

目前，为了能使用TRACE32调试RISC-V处理器，这些处理器必须配备JTAG DTM（调试传输模块）。该调试传输模块被规定为一种独立的、可置换的模块，允许芯片制造商具有通过一种不同通信接口实现对所谓调试模块(Debug Module)的访问的自由度。由于在各种不同调试通信接口方面拥有大量经验，劳特巴赫在实现该任务方面是能够胜任的合作伙伴。

通过所谓的调试模块，TRACE32调试器可以访问处理器的所有标准调试功能。这些是该规范中有关调试寄存器(Debug Register)和所谓的抽象命令(Abstract Commands)的组成部分。此外，该规范还允许设计专有的调试功能。RISC-V调试器已经支持不同的标准ISA扩展，例如作为压缩指令或浮点指令，不过它也支持客户特定的ISA扩展。