WIND RIVER

通过JTAG芯片级调试加速嵌入式Linux设备的开发

目录

介绍

在传统上，对嵌入式Linux产品的调试需要将硬件和软件组合起来使用 – JTAG工具用于硬件的启用，而基于代理的解决方案则用于软件开发。这些基于JTAG和代理的工具通常解决的都是某一方面的问题，而不是针对集成Linux开发而设计的。

风河公司将传统的JTAG硬件调试和Linux内核配置融合到一起，改变了开发人员调试Linux的方式；补丁管理、以及用户空间应用程序开发、调试和分析都在一个基于Eclipse的集成开发环境（IED）中进行，被称为风河Workbench。当传统的基于代理的解决方案在技术或经济上不可行的时候，这种能力可以允许开发人员使用JTAG连接。

JTAG连接可以在两种常见的情况下使用：一种情况是当基于代理的调试没有可用的以太网连接时；另一种情况是当开发人员需要解决在Linux内核或用户空间中发生的问题，而且需要进行系统模式调试时。通过风河公司的下一代JTAG工具，开发人员可以对硬件、引导加载程序、Linux内核和用户空间进行调试，发现系统崩溃事件以及在内核、用户空间和目标设备之间发生的其他问题。此外，使用风河Linux操作系统的开发人员还可以在代理和JTAG调试间进行无缝地转换。当基于代理的调试不可用或者过于昂贵时，这些芯片级调试的创新为项目提供了强大的备选方案。

Linux操作系统概览

要对Linux进行故障诊断，首先要理解Linux操作系统以及调试人员需要分析交互作用的领域。Linux通常包含多个相互作用的组件并采用了复杂的内存映射和管理方案。对基于Linux的嵌入式系统进行调试时，需要使用强大的工具，能够揭露操作系统的运行以及操作系统与硬件的交互作用。Linux操作系统的主要组成部分包括Linux内核、内核模块和应用程序软件。Linux内核是操作系统的核心并且拥有最高的权限。Linux内核模块是操作系统的基本要素，可以被动态地加载或卸载，并且经常用于设备驱动程序。当一个Linux内核模块被加载后，它就将具有与Linux内核相同级别的权限。Linux内核将所有的软件划分为两类，或者运行在用户模式下，或者运行在Linux内核模式下。Linux系统中运行在用户模式下的应用程序具有较少的权限。例如，这些应用程序无法直接访问Linux内核的内存或硬件。这主要是为了防止应用程序对底层系统造成破坏。因此，所有系统级别的访问都是通过Linux内核提供的。这样应用程序软件对系统的访问就受到控制，包括外围设备和内存。

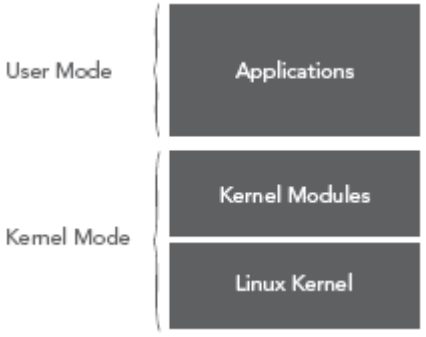

图 1：Linux 操作系统的组成部分

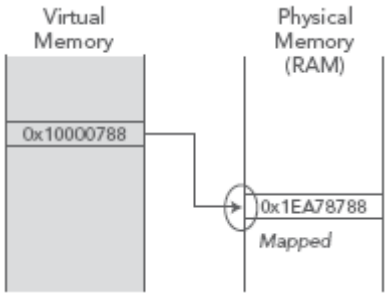

图 2：Linux内存管理

在Linux系统中，应用程序通常运行在用户模式下，而设备驱动程序则以内核模式运行。这种方案有助于保护核心（Linux内核）避免可能的系统崩溃或系统可用性的降低，但是代价则是降低了性能，因为Linux内核模块通常要比用户空间的应用程序性能要好。

与很多嵌入式设备中采用简单的“扁平”或“静态”分配的8位、16位和32位操作系统不同，Linux采用了先进的虚拟寻址技术。通过虚拟寻址技术，系统中加载和运行的Linux内核模块和应用程序就像是处于连续的内存地址空间，而软件在物理地址中的实际位置则可能分散在多个内存区段中。Linux内核负责处理Linux内核模块和应用程序的内存映射和内存分配。Linux内核和Linux内核模块使用的是静态映射内存管理，使用一个定义的常量（例如偏移量）将虚拟地址映射到物理地址。用户模式应用程序使用的则是动态内存映射，可能会使用复杂的算法来映射物理内存中的数据。

Linux内核还可以将进程分配到内存中运行；这些进程可能包括Linux内核模块或用户模式应用程序。可以将这些进程分配到内存中，完成特定的任务，然后就可以在不再需要的时候从内存中删除，从而提供更好的系统内存管理。因此，当从应用程序到Linux内核和内核模块遍历内存时，必须要小心，以考虑到Linux内核的静态和动态地址变换能力以及将进程动态分配到内存中的能力。

对嵌入式Linux设备进行调试的复杂性

Linux在嵌入式设备中的应用正在持续地快速增长。根据技术市场研究公司Venture Development Corporation（VDC）的调查，有23%的新项目将会使用Linux[1]。因为开发涉及到引导加载程序、Linux内核、Linux内核模块和应用程序，因此Linux嵌入式项目可能会非常复杂。Linux开发人员必须要对付很多问题，包括为引导加载程序建立目标设备配置文件、从用户模式到内核模式的双向Linux虚拟寻址、映射内核符号信息、并且排除用户和内核空间内复杂的缺陷故障。通过内核GNU调试器（KGDB）或GNU调试器（GDB）这样基于代理的调试解决方案，很难解决以上这些问题。

调试Linux时需要面对的挑战

- 硬件初始化
- 调试引导加载程序
- 访问指令和数据高速缓存
- 访问Linux内核中的初始化代码
- 对应用程序和Linux内核间的交互作用进行调试

引导加载程序（Bootloader）

Linux依赖于引导加载程序来启动操作系统。引导加载程序是驻留在闪存或其他非易失性存储设备中的代码，在系统加电或重置的时候执行。CPU、内存和关键I/O及外设的底层初始化都是由引导加载程序来执行的。引导加载程序开始执行后，它将会把自己复制到RAM中，然后启动Linux操作系统。

引导加载程序的调试问题

对引导加载程序的调试可能会相当复杂，因为这是以硬件为中心的，而且开发人员必须要应对如何在启动后将引导加载程序从闪存重新定位到RAM的问题。在新式的芯片级系统（SoC）处理器中，可能有数百个配置寄存器需要被初始化，这需要熟记几千页关于特定设置的文档。如果寄存器的值设置错误，那么当启动Linux内核或调试应用程序时，可能会造成后续程序问题。手动设置寄存器将是一个非常繁复的过程。

引导加载程序开发中的另一个常见的挑战是如何对引导加载程序将Linux载入到RAM中，然后启动操作系统时出现的问题进行调试。当引导加载程序将Linux内核载入到物理内存时，内核会创建自己的虚拟内存映射，这样往往就很难确定哪些是内核正在使用的地址。因此，当内核执行内存不足时，要确定如何重新定位Linux操作系统的符号表是相当复杂的。但是，基于代理的调试解决方案不支持引导加载程序调试，因为在这一过程中操作系统还没有起作用。因此，开发人员需要依赖JTAG工具来对这一关键流程进行调试。

如果开发人员需要花费大量的时间来开发和调试引导加载程序，那么他们就无法将注意力集中到设备软件和应用程序的稳定和调试上。

简化引导加载程序的调试

KGDB和GDB等基于代理的调试解决方案无法用于引导加载程序的调试。不过，JTAG调试解决方案提供了强大的功能，可以帮助开发人员快速、有效地对引导加载程序进行测试并排除故障。通过JTAG调试解决方案，开发人员可以方便地设置寄存器值并轻易地熟记寄存器内容。通过设置硬件断点的功能则可以在闪存中分步执行代码，从而能够快速地确定错误的源头。IDE支持分解和混合模式，可以同时查看源代码和汇编代码，从而简化调试流程。强大的符号管理功能可以用于代码从闪存到RAM的重新定位，为调试提供帮助。

一些JTAG调试解决方案还能够在不使用引导加载程序的情况下载入Linux内核，从而提供了额外的好处。这种功能是通过引导行（boot line）参数实现的，对于那些希望在引导加载程序就绪和可用之前就开始系统开发的项目管理者来说，这种功能特别有用。通过引导行功能的JTAG解决方案可以实现引导加载程序开发和操作系统的稳定同时进行，这有助于缩短开发周期。

Linux内核和内核模块

Linux内核和内核模块是Linux操作系统的核心组件。引导加载程序对系统进行初始化后，首先启动的是Linux内核，然后再根据需要载入Linux内核模块。

在操作系统启用的过程中，开发人员关注于Linux操作系统的优化或定制以及内核模块的开发。在这一阶段，硬件和软件间的交互作用被密切地监视着。对于Linux内核的调试来说，开发人员需要拥有对寄存器、高速缓存和其他底层数据的可见度。Linux内核模块的调试也将需要初始化代码的透明性，同时还要处理RAM中动态分配的内存。

KGDB要求稳定的Linux内核和定制的硬件接口（例如设备驱动程序）都就绪后，代理才能工作。基于代理的调试不具有对硬件的可见性，而且也无法提供理解硬件和Linux内核间交换作用所需的完整诊断功能。代理要求对Linux内核的探测，这可能会对嵌入式设备造成副作用。

当使用代理来调试Linux内核和内核模块时，需要考虑的其他事项还包括当遇到断点时系统的停止或冻结。例如，KGDB将不会停止CPU（尤其是在多核或多重处理环境中）来让开发人员检查系统的当前状态。KGDB需要操作系统来使代理工作，因此对于崩溃系统的调试没有帮助。此外，KGDB还需要主机操作系统和目标设备之间的通讯端口，例如以太网端口。因此，要使用代理对Linux内核模式进行调试就需要建立起通讯信道，带有正常工作的IP堆栈、稳定的Linux内核、以及正常工作的设备驱动程序。在操作系统的稳定阶段，这可能会是一个挑战，因为这种通讯资源可能还没有建立，或者通讯本身就需要被调试。

使用JTAG进行调试

需要有针对Linux内核和内核模块的全面调试解决方案来为目标系统验证和稳定Linux内核。基于JTAG的调试解决方案在这一工作流程中体现了重要的价值。基于JTAG的调试工具提供的功能包括查看局部/全局符号和寄存器，以及查看数据和指令缓存的内容。一些商业JTAG解决方案还提供了从虚拟内存到物理内存间的无缝地址映射，使开发人员能够查看正确的内存地址和内容。这些解决方案可以对Linux内核模块的初始化进行调试，而且能够多次载入和卸载这些模块，而无需断开和重新连接目标系统。

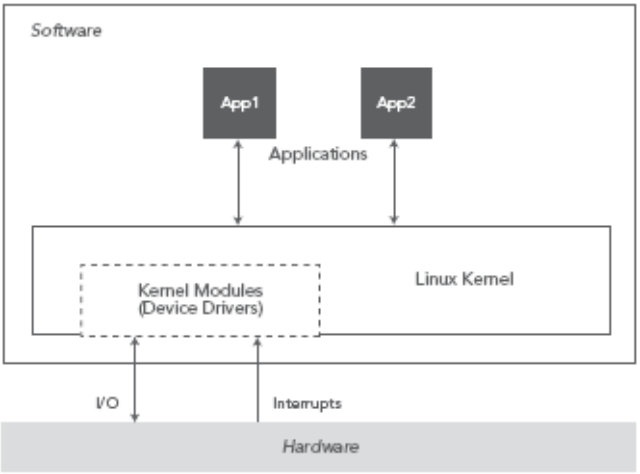

图 3：Linux中的软件和硬件交互

JTAG调试解决方案可以完全暂停系统来检查操作系统和应用程序的状态，这是一个非常强大的功能，对于Linux内核和Linux内核模块这样的系统模式调试非常有用。在系统模式调试中，开发人员可以停止整个系统，包括处理器、操作系统、所有的线程、以及中断处理。以这种方式停止系统后，就可能对硬件和软件进行详细的观察，而且还可以重新启动系统并在启动后逐步执行代码。

JTAG解决方案可以应用于KGDB无法工作的条件下，尤其是对于Linux内核发生致命错误和目标设备崩溃的情况。这些能力对于设备驱动程序和操作系统的稳定特别有用。

Linux应用程序调试

Linux应用程序是在Linux内核控制下运行的用户程序，通过系统调用来访问系统资源。Linux内核对系统调用进行处理，并决定如何提供对硬件和内存的访问。

对于用户模式应用程序的调试，开发人员需要对应用程序线程的直接访问，并且可以启动和停止某个线程并查看变量和堆栈。由于一个应用程序可能包含多个进程和线程，因此有必要停止所有与被调试应用程序线程相关的所有线程，包括可能会对应用程序产生影响的线程。此外可能还需要监视不同进程和CPU的外围寄存器。GDB只能在线程水平上工作，并且只能停止一个线程，无法同时停止整个系统或多个线程。

在用户模式下进行调试的开发人员需要从用户模式通过系统调用进入Linux内核模式，然后返回用户模式。由于Linux中采用的虚拟内存管理结构，因此在这两个模式间转换时要跟踪内存地址是很有挑战性的。我们无法依赖于同样的物理地址。在这一过程中，正确地跟踪内存映射和内存分配非常关键。因此，代理解决方案需要同时使用GDB和KGDB来跟踪进入到Linux内核和内核模块的系统调用。同时使用多个基于代理的调试工具可能会使调试过程更加复杂。

Linux应用程序开发人员可能会遇到嵌入式系统中的设备没有以太网端口或串行通信端口的情况，此时将无法使用基于代理的调试方法。即使设备上有通讯端口，代理也需要接口就绪后才能工作。如果通讯端口没有就绪或者本身就需要进行调试，或者IP堆栈或代理软件所需的内存不可用，那么将无法进行基于代理的调试。

基于JTAG的调试提供了对运行于Linux用户模式下应用程序的深入可见度。对于进行系统调用的应用程序，双模式的JTAG调试解决方案可以同时提供对Linux内核和用户模式的可见度，而且这些都不会对Linux内核进行探测。双模式的解决方案可以查看所有的应用程序线程、这些线程的上下文、有哪些线程进入了Linux内核、以及使用的参数和线程变量。对于缺少通讯端口的设备（例如移动设备、医疗设备、汽车系统等），基于JTAG的解决方案比基于代理的调试提供了更多的优势。

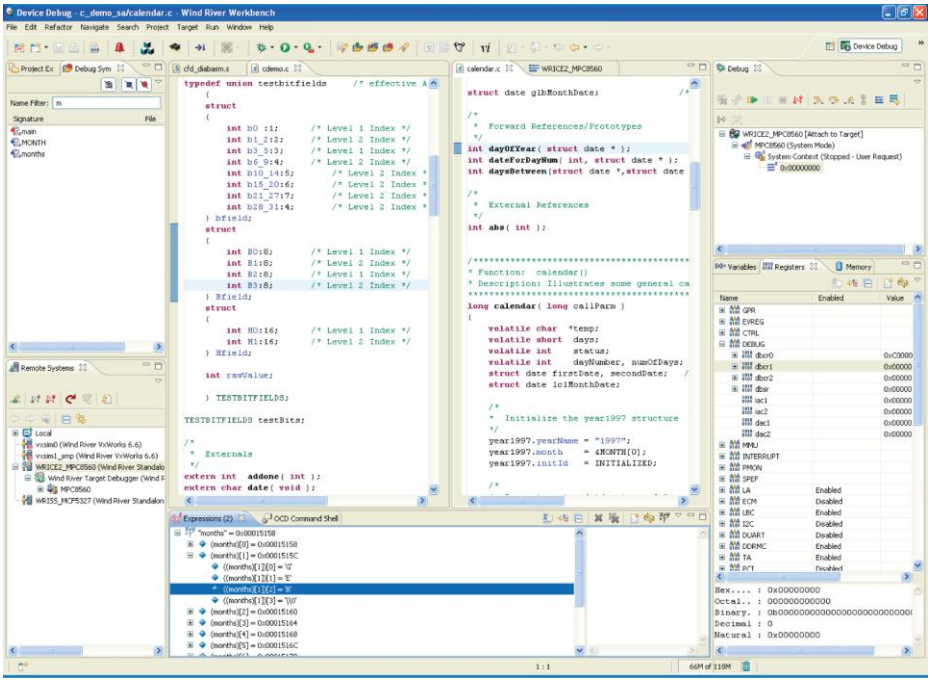

图 4：用于Linux应用程序，基于Eclipse的IDE

系统模式调试可以暂停处理器并检查操作系统和所有线程的状态，从而简化了对多线程应用程序的调试。我们前面已经提过，很多问题都是因为多个线程之间的交互作用引起的。由于基于代理的调试方法无法同时停止所有的线程，因此很难找出问题，从而造成项目的调试时间被大大延长。系统模式调试可以详细地查看系统的当前状态（例如每个线程的状态、变量等），在给定的时间点可以完整地查看系统，因此与基于代理的调试方法相比是更好的解决方案。

基于JTAG的调试解决方案采用的是与目标硬件非侵入式的连接。它可以连接到已经运行、遇到故障的系统。这种连接无需改变处理器寄存器的状态和同步Linux内核及应用程序的上下文就可以进行调试。例如，当基于Linux的系统处于死锁状态时。通过JTAG解决方案，开发人员可以在不改变系统状态的情况下连接到目标系统。然后开发人员就可以查看Linux内核对象、应用程序上下文，从而确定引起故障的线程、系统调用、和系统调用使用的参数。这是一种端到端的解决方案，尤其适用于基于代理的开发解决方案无法使用的情况。JTAG调试简化了开发工具，从而缩短了调试时间。

针对基于JTAG的Linux调试的风河解决方案

风河公司提供的基于JTAG的芯片级调试解决方案并不是要替换流行的基于代理的解决方案，而更多的是一种补充。KGDB和GDB等基于代理而不是基于JTAG的解决方案可能不适用于早期的开发阶段，例如引导加载程序调试、操作系统稳定、以及设备驱动程序调试等。对于仅依赖于基于代理的调试的产品开发来说，这可能会造成影响。

一般来说，大多数设备都带有用于调试和测试的JTAG端口。不过，很多消费者产品（尤其是那些对价格很敏感、空间受限、或者存储器受限的设备）都不具有执行基于代理的调试所需的通讯端口。在另外一些情况中，由于严格限制的物理和存储器空间要求、安全问题、或者缺乏IP堆栈，也无法使用基于代理的调试。基于JTAG的解决方案对于这些环境都是非常理想的。

JTAG还提供了对总体系统的重要观察，而这在使用代理的调试中是无法实现的。由于Linux用户和Linux内核的双模式结构，当开发人员需要同时获得有关应用程序、操作系统、以及基础硬件的相关信息时，使用GDB和KGDB这样基于代理的调试可能就会造成困难。JTAG解决方案则可以同时提供对这些区域的重要可见度，从而更加方便、快速地进行调试。

图 5：风河基于JTAG的芯片级调试解决方案

风河公司针对Linux调试的高级解决方案

风河Workbench芯片级调试版运行在工业标准的基于Eclipse的平台上，为嵌入式Linux系统提供了先进的基于JTAG的调试功能。Workbench芯片级调试支持风河Linux。对于用户自主开发或Semiconductor Linux发行版本，风河公司扩展了芯片级调试功能，从而可以对kernel.org的Linux进行基于JTAG的内核和用户模式调试。风河调试解决方案支持多种目标硬件，包括最先进的多核处理器。

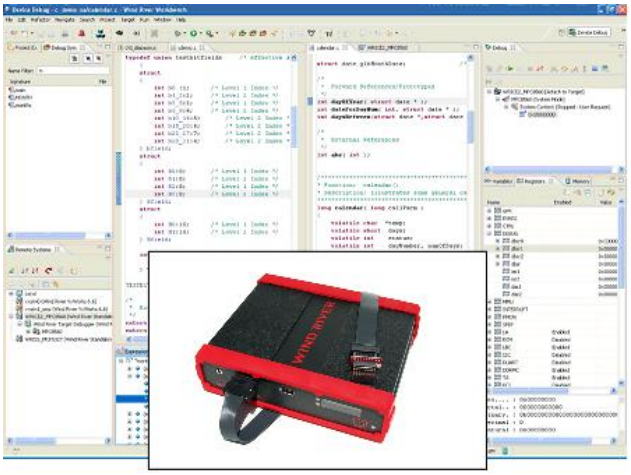

图 6：风河Workbench芯片级调试版和用于基于JTAG调试的风河ICE

Workbench芯片级调试

- 引导加载程序开发，对在新硬件中启用Linux时发生的问题进行调试。
- 内核模式调试，可以排除操作系统和设备驱动程序的问题。
- 用户模式调试，不带内核探测，通讯信道，或IP堆栈。
- 用户模式调试中的高级功能，可以允许开发人员查看系统和应用程序上下文，包括Linux内核是如何进入的、是哪个（或那些）线程进入的、使用的参数、以及线程变量。
- 全面的系统模式调试：停止整个处理器或全部线程的断点。

针对整个Linux开发流程的风河芯片级调试

风河Workbench为Linux开发人员提供了灵活性，可以使用JTAG连接来实现硬件启用；调试Linux内核、中间件和用户模式应用程序；并且在适当的情况下可以无缝地转换到基于代理的风河Linux调试，所有这些都在同一个IDE中实现。这些功能增强了多个开发团队间的合作，并缩短了解决问题所需的时间。

风河公司的解决方案用来满足日趋复杂的系统带来的需求。通过对目标系统的远程访问，分布在全球的开发团队可以进行合作，从而提高生产力。连接和配置管理可以简化对多个目标的连接，支持在单一的用户界面中对整个系统的调试。这些连接可以是核心、处理器或进程。风河芯片级调试支持包括JTAG工具和代理在内的多种调试连接类型，为包括引导加载程序启用、Linux内核和Linux内核模块稳定在内的Linxu系统调试以及应用程序开发提供了最大了灵活性。

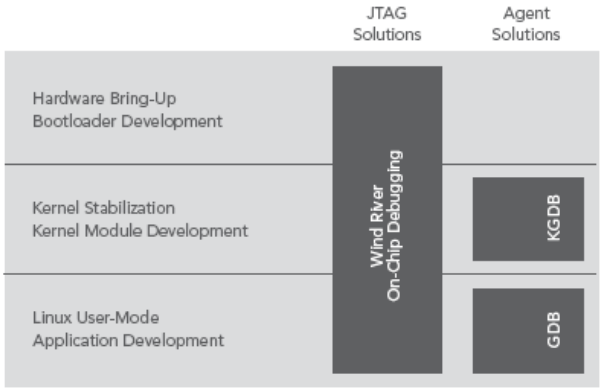

图 7：风河芯片级调试涵盖了广泛的Linux开发内容

结论

随着嵌入式系统的日趋复杂，开发人员在进行调试时也面临着越来越多的挑战。在当前复杂的环境中，传统上基于代理的Linux调试解决方案对用户模式和Linux内核模式的调试在效果和效率上都不是很令人满意。Eclipse等基于标准的开发环境针对Linux环境进行了专门设计，JTAG调试集成到这种环境中后，可以在Linux开发中起到非常有价值的作用，并改善“编辑-编译-调试”流程。

风河公司提供的这些解决方案现在提供了扩展芯片级调试使用的创新工具，并且在基于代理的调试不可用或者不经济的情况下提供了强大的替换方案。这些芯片级调试的独特功能与基于代理的调试有效地结合起来，在最为复杂的环境中也可以改善调试性能。产品的上市时间变得越来越重要，通过风河公司提供的芯片级调试解决方案，厂商们可以缩短开发时间并降低成本。

注释

1. VDC，“嵌入式系统市场中的Linux”，嵌入式软件市场情报计划，2007年11月。

WIND RIVER

Wind River 就在您身边

北京代表处	北京市朝阳区望京中环南路9号望京大厦B座18层	邮编：100102	电话：010-84777100	传真：010-64398189
上海代表处	上海市西藏路585号新金桥广场3-H,I,J室	邮编：200003	电话：021-63585586/87/89/90	传真：021-63585591
深圳代表处	深圳市福田区车公庙天安数码时代大厦A座606室	邮编：518040	电话：0755-25333408/3418/4508/4518	传真：0755-25334318
西安代表处	西安市高新区科技二路68号西安软件园秦风阁H103	邮编：710075	电话：029-87607208	传真：029-87607209
成都代表处	成都市高新区天府软件园二期D7 14层	邮编：610041	电话：028-65318000	传真：028-65319983

关于风河更多内容请访问：http://www.windriver.com.cn Email：inquiries-ap-china@windriver.com

WIND RIVER