嵌入式远程调试器保护模式下调试功能的实现

姚放吾,丁 皞
(南京邮电大学 计算机学院,江苏 南京 210003)

摘 要:嵌入式远程调试器为嵌入式系统开发者提供了一个友好的软件调试平台。探讨了保护模式下的用户目标程序的调试方法,并详细阐述如何运用运行在远程设备上的监控程序实现实模式下监控环境与保护模式下用户环境之间的相互切换。监控程序能够判别用户目标当前运行模式,通过Inter处理器在保护模式或实模式下的断点中断和单步中断,实现用户目标程序的单步运行、运行到断点以及查看或修改内存和寄存器等符号化调试功能,能够让用户在远程调试环境中方便地使用数据共享、多任务以及特权保护等技术。

关键词:远程调试;保护模式;调试功能
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2011)04-0242-04

Realization of Debugging Function in Protected Mode of Embedded Remote Debugger

YAO Fang-wu, DING Hao
(College of Computer, Nanjing University of Posts & Telecommunication, Nanjing 210003, China)

Abstract: Embedded remote debugger provides a friendly software platform for the developers. Studys the debugging methods of user target program in protected mode, and introduces detailedly how to run the remote device monitoring program to achieve the switch between monitor environment in real mode and user environment in protected mode. Using the features of Intel processer of breakpoint interruption and singal step interruption, the monitor program can determine the current operating mode of user and achieve the functions such as single step running, running to the breakpoint, and showing & verrfying the contents of memory and registers of the remote device. The program can provide with a user-friendly debugging environment applying data sharing and multi-task and privilege protection technology conveniently.

Key words: remote debugging; protected mode; debugging function

0 引 言

在嵌入式系统软件开发过程中,软件的开发周期以及开发成本与软件调试器密切相关,调试工具的优劣是影响开发效率的一个重要因素[1]。传统的嵌入式系统的调试手段主要有软件仿真或是通过处理器硬件调试,这两种方法因其高昂的开发维护费用和较长的研制周期,无法很好地胜任调试需求[2]。另外在一些特殊环境下,系统可能不支持本地调试器的运行,于是远程软件调试就成为嵌入式系统调试的重要手段[3]。远程调试技术采用宿主机/目标机开发环境,通过在宿主机上进行源文件编写,并利用交叉编译器和调试器进行软件调试[4]。

笔者设计的调试器主要运用于没有操作系统的嵌入式设备环境[5]。在软件调试过程中,调试器运行于主机环境中,被调试的软件则运行目标机上。主机上的调试器通过串口、USB接口与目标机进行通信,控制目标机上程序的运行,实现对目标程序的调试。

上述调试器主要是在实模式下实现,用户可能需要对保护模式目标程序进行调试,需要远程调试器为其提供保护模式下的调试功能。保护模式能为用户提供更大的寻址空间,实现代码和数据的共享以及特权保护等功能,所以设计并实现远程调试器保护模式调试功能有重要的意义。保护模式与实模式的寻址方式存在差异,实模式与保护模式下的目标程序实现远程调试的机制与方法有所不同。基于如上问题,笔者设计出保护模式下远程设备调试方法,能够同时实现实模式和保护模式下的目标程序的调试功能。

1 嵌入式远程调试器的系统架构

嵌入式远程调试器主要分为上位机(主机端)和

收稿日期:2010-09-07;**修回日期**:2010-12-09
基金项目:江苏省企校合作项目(2008发04)
作者简介:姚放吾(1953-),男,教授,硕士生导师,研究方向为嵌入式技术及其在通信中的应用、并行计算机及其体系结构。

下位机(目标机)两个部分,这两个部分分别自治地运行在不同的硬件平台。上位机实现对用户程序的编辑、编译,目标文件数据和代码的提取,与下位机的连接及通信,通过符号化调试观察目标程序的执行状态,控制程序的执行过程,修改程序的运行状态[6]。下位机监控程序接收上位机发送的目标程序和调试命令,被调试的目标程序启动后根据上位机的控制命令,实现目标程序的调试运行,并将采集的调试信息通过通讯模块发送至上位机。上位机与下位机的通讯有串口和 usb 两种方式。其总体架构如图 1 所示。

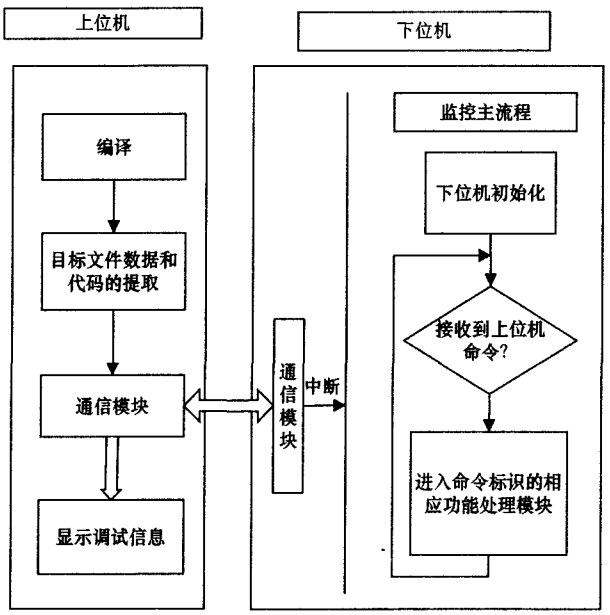

图 1　调试器总体框架

2　下位机监控原理

远程调试器符号化调试的实现主要由下位机完成。符号化调试功能包括设置断点、单步运行、查看和修改寄存器、内存单元信息等[7]。下位机运行着一个监控程序,负责用户目标程序的接受,对上位机调试命令作处理,并通过通讯模块向上位机返回调试信息。监控程序与用户目标程序的数据段和代码段分别存放固定区域,当执行用户代码时,监控程序先将自己的运行环境储存(包括所有的普通寄存器和状态寄存器),并还原用户的运行环境[8]。调试过程中用户目标程序产生调试中断(包括断点中断以及单步中断)时保存用户程序的运行环境,收集调试信息,并返回监控程序,等待上位机下一个命令。

3　保护模式下调试功能的实现

用户的保护模式目标程序中有实模式和保护模式之间的切换。对用户目标程序进行调试时,下位机执行的用户程序处于实模式时,原有的调试机制可以实现调试功能。当用户目标程序进入保护模式时,需要另外一套机制完成监控程序和用户程序之间的切换。

3.1　保护模式下地址查询的特点

和实模式下一样,保护模式下程序运行的实质仍是"CPU 执行指令,操作相关数据",因此实模式下的各种代码段、数据段、堆栈段、中断服务程序仍然存在,且功能、作用不变[9]。保护模式下引入描述符来描述各种数据段,若干个描述符集中在一起组成描述符表(全局描述符表 GDT、局部描述符表 LDT、中断描述符表 IDT)。保护模式与实模式相比,段选择子代替了段值。在保护方式下,虚拟地址空间中存储单元的地址由段选择子和段内偏移两部分组成。选择子确定描述符,描述符确定段基地址,段基地址与偏移之和就是线性地址。"地址转换"由描述符表来完成,从这个意义上说,描述符表是一张地址转换函数表。

3.2　实模式与保护模式切换

用户目标程序可能在实模式或保护模式两种模式下运行,所以监控程序装载用户目标程序运行环境时需要判断用户当前运行模式并采用不同处理方法。下面主要叙述用户程序运行在保护模式下的处理方法。

下位机监控程序始终运行在实模式下,当用户目标程序运行到保护模式环境下,监控程序与用户目标程序之间的切换就涉及到实模式与保护模式的切换。监控程序与用户目标程序之间的切换分为两个部分:一是监控程序切换到用户目标程序;二是用户目标程序产生调试中断(保护模式下的断点中断和单步中断)时跳转到监控程序。

监控程序跳转到用户目标程序时需要装载用户保护模式环境。用户的 GDT、IDT 存放在固定的位置,监控程序先将 GDT、IDT 的基址和边界装入 GDTR(全局描述符表寄存器)、IDTR(局部描述符表寄存器),再将用户目标程序的选择子装入段寄存器。

调试中断完成用户运行环境到监控程序的切换,并储存上位机需要的调试信息。在保护模式下,断点中断和单步中断的中断类型码(3 和 1)作为 IDT 的索引号使用。CPU 取得中断类型码后乘以 8 再和 IDTR 中的基地址相加,从 IDT 中选择索引号所指示的陷阱门,CPU 再将门描述符中的选择子加载到 CS 中,由选择子所确定的描述符中的基地址以及门描述符中的偏移量确定中断处理程序的线性入口地址。中断处理程序在返回监控程序前需要完成保护模式到实模式的切换,所以整个中断处理程序前半段运行在保护模式下,后半段运行在实模式下。下位机处理流程如图 2 所示。

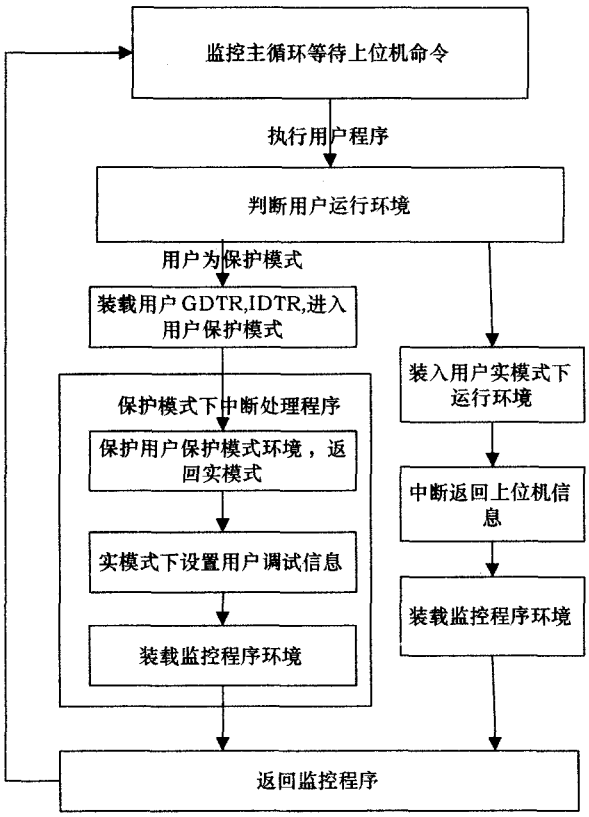

图 2 下位机处理流程

3.3 内存映像设计

在一个系统中，GDT 和 IDT 只有一张，局部描述符表可以有若干张，每个任务可以有一张。描述符表本身也是一种数据段，需要存储在监控程序数据段或者用户程序数据段。调试中断处理程序位于监控程序代码段，中断处理程序所保存的用户运行环境及调试信息等数据内容位于监控程序数据段。用户程序的描述符表中需要有描述符指向监控程序代码段及数据段才能调用调试中断处理程序。根据上述因素用户描述符表中需要插入监控程序的描述符。上位机从用户目标程序数据段中提取用户全局描述符表和中断描述符表单独下发给下位机，下位机再将它们存放到一个固定位置，并将监控程序描述符插入用户描述符表中。描述符表的内存映像如图 3 所示。

3.4 保护模式下运行到断点的实现

在计算机程序的调试技术中，断点是最常用的技术之一[10]。程序员可以在程序代码中有疑问处设置断点，迫使程序运行到该处暂停，通过调试工具查看进程的栈结构、变量值等来分析判断程序运行的正确性[4]。在保护模式下完成此功能需要两个步骤：

1）设置断点。

保护模式下断点的设置方法和实模式下的方法基本相同，下位机在接收到用户设置的断点地址命令时，把断点地址存放于监控程序的内存缓冲区中。监控程序接收到上位机的运行到断点调试命令后，保存监控程序的运行环境，还原用户目标程序保护模式下的运行环境，打开 CPU 状态寄存器的 T 标志，以实现单步运行后，产生单步中断。CPU 执行一条用户指令后，进入保护模式下的单步中断服务处理程序[3]。单步中断从内存缓冲区中读取断点地址，将所有断点处的代码保存并替换为 CC(INT 3)，还原用户目标程序的运行环境后继续执行，直到运行到断点处，进入保护模式下断点中断服务程序。

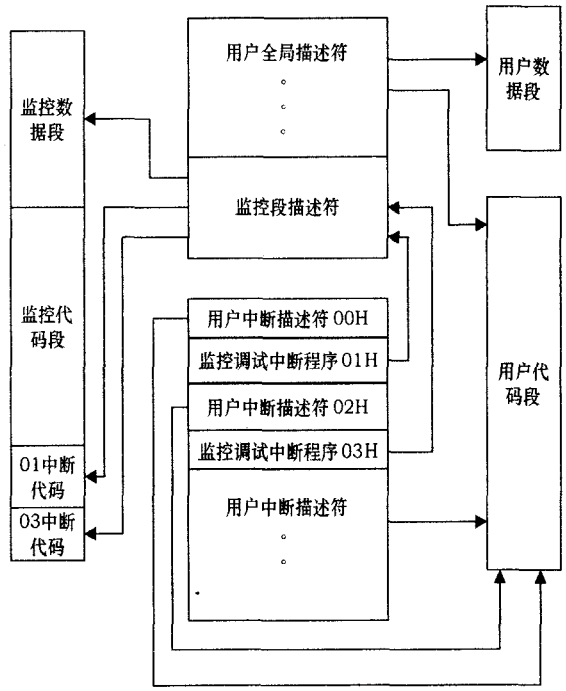

图 3 内存映像

2）保护模式下断点中断处理。

断点中断服务程序首先需要完成用户环境的保存，此时的中断服务程序处于保护模式。在用户环境数据保存过程中需要用到用户堆栈段，另外在执行断点中断指令时，用户当前的标志寄存器、ECS、EIP 被压入用户堆栈段，所以需要将用户堆栈的的 EIP、ECS、标志寄存器先后出栈，以还原到产生断点中断前的状态[11]。中断程序从保护模式跳转到实模式后，加载监控程序堆栈段，还原所有的断点信息（即把 INT3 指令还原为先前的用户代码），并储存相关调试信息。最后还原监控程序运行环境，并把监控程序的 CS 和 EIP 压栈，使用 IRET 指令，用栈中的 CS 和 EIP 修改当前的用户 CS 和 EIP 从而跳转到监控程序继续执行。

3.5 保护模式下单步运行的实现

单步运行分为单步进入和单步跳跃两种情况。单步跳跃是指实现程序的单条代码的运行（如果是执行某个过程或函数，则运行完整个过程或函数），而单步进入时程序则会运行到过程或函数中的第一条指令

处。单步运行的具体实现方法如下:

1)单步进入 。

单步运行主要通过单步中断来实现。监控程序接收到上位机的单步调试命令,保存监控程序的运行环境,还原用户程序的运行环境,打开 CPU 状态寄存器的 T 标志(即:置 T 标志为 1),以实现单步运行后进入单步中断服务程序。还原用户保护模式下运行环境,通过 IRET 指令跳转到用户程序上次到执行的位置,继续执行。CPU 在执行一条用户指令后,进入保护模式下的 T 中断服务处理程序。T 中断服务处理程序和断点中断处理程序类似,需要完成用户保护模式环境与监控程序实模式环境的切换,以及用户运行环境和调试信息的储存。

2)单步跳跃。

单步跳跃的实现方法与运行到断点类似,需要使用断点中断和单步中断共同完成。因为单步跳跃需要执行完整个函数或过程,所以上位机需要将跳跃地址(函数或过程体的下一条指令的起始地址)传输给下位机,下位机使用保护模式下的单步中断将该地址处的用户代码替换为 CC,还原用户程序的运行环境并继续执行。当用户程序执行到断点处时执行断点中断服务程序,切换到实模式下的监控程序,便实现了单步跳跃功能。

4　结束语

远程调试器是嵌入式开发的一个重要工具,文中给出了一个在实模式远程调试器的基础上实现保护模式下调试功能的方案,使远程调试器能够实现实模式和保护模式两种环境下的目标程序的调试功能。

参考文献:

[1] 李红卫,嵌入式远程调试工具的研究与实现[J]. 微计算机信息,2009,25(1-2):87-89.

[2] 乔　容,彭思鹏,柏桂枝. 嵌入式软件源码级交叉调试器的设计与实现[J]. 舰船电子工程,2004,24(3):56-60,101.

[3] Azzerini B, Lopriore L. Program debugging environments:design and utilization[M]. New York: Ellis Horwood,1992.

[4] 章　辉,桥　井,高　桥,等. 基于仿真器的源码级调试器设计与实现[J]. 计算机工程与设计,2010,31(8):1685 - 1688.

[5] 包　磊,姚放吾. 基于远程设备的汇编源码调试器的设计与实现[J]. 计算机技术与发展 2009,19(11):155-158.

[6] 张　欣,栾　新. 基于 ARM 芯片的嵌入式交叉调试系统[J]. 微计算机信息,2007,6-2:128-130.

[7] 王亚磊,姚放吾,罗　威,等. 基于 386EX 的嵌入式软件调试器的设计与实现[J]. 计算机技术与发展,2007,17(s):264-266.

[8] 吴志雄. 一种易于扩展的交叉调试器设计及其实现[J]. 电子技术,2007,36(11):26-29.

[9] 杨季文. 80x86 汇编语言程序设计教程[M]. 北京:清华大学出版社,1998.

[10] Rosenberg J . How Debuggers Work[M] . New York : Wiley Computer Publishing,1996.

[11] 曲玉昭,陆华奇,于忠清. 基于事件和周期任务的嵌入式系统设计[J]. 计算机技术与发展,2006,16(10):175-177.

(上接第 241 页)

在水下探测系统软件开发的各阶段,使用 UML 对系统进行建模,可以通过建立不同的模型对系统进行描述,取得良好效果。这种标准化、独立性的模块化软件设计方法能够有效地促进水下探测系统的规范化、通用化设计,使得软件模块重用成为可能。

参考文献:

[1] 李　博,周穗华. 水下探测系统内部总线设计[J]. 仪器仪表学报,2008,29(8):30-34.

[2] 吴　际,金茂忠. UML 面向对象分析[M]. 北京:北京航天航空大学出版社,2002:2-6.

[3] Booch G,Rumbaugh J,Jacobson I. The Unified Modeling Language User Guide[M]. [s. l.]:Addison Wesley Longman, 1999:Appendix B.

[4] 冀振燕. UML 系统分析设计与应用案例[M]. 北京:人民邮电出版社,2003:2-10.

[5] 王小平,宣乐飞,张　蔚. 基于 UML 的嵌入式实时控制系统的建模与实现[J]. 计算机技术与发展,2006,16(7):239-244.

[6] 王建新,姚放吾. 基于 UML 的软硬件协同设计方法[J]. 计算机技术与发展,2006,16(1):96-98.

[7] Gomaa H. Designing Concurrent, Distributed, and Real-Time Applications with UML[M]. [s. l.]:Pearson Education, Inc. 2000.

[8] 李宏伟. 开放式数控系统分布式体系结构及其实现策略研究[D]. 天津:天津大学机械工程学院,2005.

[9] 史先传. 基于 CAN 总线的开放式数控系统硬件设计与实现[D]. 南京:南京航空航天大学,2001.

[10] 陈宁宁. 开放式网络控制系统的研究与应用[D]. 杭州:浙江大学电气工程学院,2006.

[11] 杨家龙. 基于网络环境的舰船机舱动力装置监控系统技术研究[D]. 哈尔滨:哈尔滨工业大学,2006.

[12] 谢林柏. 网络化控制系统中若干问题的研究[D]. 武汉:华中科技大学,2004.