使用 I²C 总线实现 TMS320VC5509A 引导装载设计

向　展,裴昌幸,易运晖
(西安电子科技大学　综合业务网国家重点实验室　陕西 西安　710071)

摘　要:重点研究 TI 公司的 C55x 系列 DSP 芯片的引导装载方法。选择 TI 公司的 TMS320VC5509A 处理器,详细介绍了使用基于 ARM7TDMI 内核的 LPC2138,采用 I²C 总线模式对 DSP 实现程序上电引导装载的过程。文中还给出相关的硬件连接、自举表的建立以及 ARM 端中断服务程序的设计方法和实例。实验证明,采用该引导装载方法的系统具有优良的性能和很强的实用性。

关键词:引导装载;DSP;I²C 总线;ARM
中图分类号:TP311　　**文献标识码**:B　　**文章编号**:1004-373X(2006)18-080-03

A Bootloader Design of TMS320VC5509A Using I²C Bus

XIANG Zhan, PEI Changxing, YI Yunhui
(State Key Lab of Integrated Service Networks, Xidian University, Xi'an, 710071, China)

Abstract: This paper researches a bootloader about C55x DSP. It takes the TMS320VC5509A for example, and selects the I²C bus mode to boot the DSP with LPC2138, a ARM based on ARM7TDMI core. It also gives the circuits frame, a boot table building and the design of interrupt service routine in ARM. A system adopted the method performs well in experiments.

Keywords: bootloader; DSP; I²C bus; ARM

1　引　言

DSP 芯片的 Bootloader 程序用于上电时将用户程序从外部非易失性、慢速存储器或外部控制器中装载到片内高速 RAM 中,保证用户程序在 DSP 内部高速运行。TI 公司的 C55x 系列 DSP 芯片提供多种装载模式,主要包括 HPI 引导装载、串行 E²ROM 引导装载、并行引导装载、串行口引导装载、I²C 总线 E²ROM 引导装载等。通常使用的是并行引导装载模式,该方式引导速度快实现简单,但是体积和功耗也较大。随着串行接口存储设备容量的提高,串行引导方式体积小、功耗低的优势便显现出来了。所以使用 ARM 的串行接口对 DSP 进行引导装载,不仅能省去存储芯片,而且利用 ARM 的 ISP 功能,可以根据需要改变用户程序,有利于系统的维护和升级。

本文以 TMS320VC5509A 芯片引导装载为例,详细介绍了利用 ARM 通过 I²C 串行引导方式来实现程序的引导装载,其他引导过程可参考相关技术资料[1]。

2　Bootloader 程序简介

TMS320VC5509A 是 TI 公司一款 16 位定点低功耗 DSP 芯片,其指令周期最快为 5 ns,片内拥有 128×16 k 高速 RAM,性价比很高,被广泛用于嵌入式手持设备、通信、数据采集等领域。

TI 公司的 DSP 芯片出厂时,在片内 ROM 中固化有引导装载程序(Bootloader),其主要功能就是将外部的程序装载到片内 RAM 中运行,以提高系统的运行速度。C55x 系列 DSP 其 Bootloader 程序位于片内 ROM 空间的 0xFF0000～0xFF8000 处。进入 Bootloader 程序后,程序先对 DSP 进行初始化,配置 DSP 的堆栈寄存器、中断寄存器和 DSP 状态寄存器,保证在引导装载用户程序时不会被中断,从而导致程序加载失败。

由于 DSP 可以通过自举表对寄存器进行修改,需要注意在 Bootloader 程序运行时,尽量不要修改 Bootloader 程序配置过的中断控制寄存器,否则会导致不可预料的后果。

2.1　I²C 引导模式硬件连接

为了通过 I²C 总线来实现对 DSP 的引导装载,通常情况是选择具有 I²C 总线接口的 E²ROM,电路框图如图 1 所示。其中 GPIO0～GPIO3 是用来选择 Bootloader 引导模式,当 DSP 复位后对这 4 个管脚电平采样,根据不同的组合进入到对应的 Bootloader 程序,表 1 列出了 GPIO0～GPIO3 的管脚不同状态的组合以及对应的 Bootloader 引导方式。SDL 和 SDA 分别为 I²C 的时钟和数据线,其上拉电阻的大小取决于所连接 I²C 设备的多少[2]。

收稿日期:2006-05-24

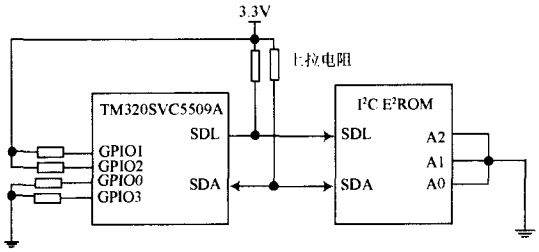

图 1 DSP 与 I²C E²ROM 通过 I²C 总线相连

表 1 不同组合及其对应引导方式

IO0	IO3	IO2	IO1	引导源
0	0	0	1	24 b 寻址 SPI 接口 E²ROM
0	0	1	0	USB
0	0	1	1	I²C 接口 E²ROM
0	1	0	1	EHPI 多元模式
0	1	1	0	EHPI 非多元模式
1	0	0	0	直接在外部并行存储器中执行
1	0	0	1	16 b 寻址 SPI 接口 E²ROM
1	0	1	0	8 b 并行 EMIF
I	0	1	1	16 b 并行 EMIF
I	1	1	0	16 b 标准串行引导
1	1	1	1	8 b 标准串行引导

如果通过 I²C 总线对 DSP 实现引导装载，对存储数据的 I²C 设备有如下几点要求：

(1) 该设备首先必须兼容 Philips 的 I²C 总线规范 V2.1，工作在从设备模式，并且其从设备地址为 0x50。

(2) 设备内部使用两个字节寻址，即在接收到主机写命令后，其后接收到的数据是 16 位的地址数据。

(3) 对设备读取时，相关设备必须支持自动寻址增量。即每读一次，其内部地址指针自增 1，保证程序按顺序读出。

常用的 I²C 接口 E²ROM 有 ST 公司的 M24 系列及 Philips 的 PCF85 系列的 E²ROM，根据程序大小选择相应的芯片。需要注意的是 I²C 引导模式最多支持 64 kB 的数据。

在 I²C 引导模式运行时，DSP 作为主设备来控制 I²C 总线的时钟。对于 DSP 来说 SCL 必须满足根据方程(1)所得到的速率。而 I²C 引导模式支持的最高时钟速率为 400 kHz，所以如果想利用 I²C 引导模式，DSP 上电时输入时钟就不能大于 12 MHz。

$$\text{SCL(高)} = \text{SCL(低)} = 15 \times (\text{DSP 输入时钟周期}) \quad (1)$$

2.2 I²C 引导模式数据存储方式

为了能正确地将数据从外部存储器搬移至 DSP 内部，用户程序需要将数据按照一定格式存储在 E²ROM 中，按照这些格式存储的数据便是自举表(Boot table)。自举表是 Bootloader 程序能正常运行的保证，只有将数据按照自举表的要求存储，用户程序才能被搬移到 DSP 内部正常运行。在自举表中除了用户数据外还需要一些 Bootloader 控制数据，如程序入口地址(entry point address)、寄存器配置(register configurations)和可编程延迟(programmable delay)等，自举表的结构如表 2 所示。

表 2 自举表结构

32 位程序入口地址			
32 位寄存器配置计数器 n			
16 位寄存器地址(register 1)		16 位寄存器内容(register 1)	
……(register n addr)		……(register n contents)	
16 位延迟指示器		16 位延迟计数器	
32 位段字节计数器(section 1)			
32 位段起始地址(section 1)			
数据(字节)	数据(字节)	数据(字节)	数据(字节)
数据(字节)	数据(字节)	数据(字节)	数据(字节)
……(section n)			
32 位全 0 数据(Boot 表结束标志)			

其中程序入口地址在将用户程序搬移至 DSP 内部后，用户程序从该地址处开始运行，通常情况是中断向量表的 reset 处。在 Bootloader 搬移数据之前，如果需要可以改变某些寄存器的值，如 DSP 的 clock 配置寄存器、EMIF 配置等。通过自举表配置这些寄存器后，需要一定时间才能正常工作，否则会导致引导程序失败。延迟计数器是让 Bootloader 推迟相应的 CPU 周期再进行数据搬移，确保引导程序正常工作。由于 DSP 是采用分段格式来组织数据的，如代码段、数据段和用户自定义数据段等，所以生成的自举表也是按照对应格式来建立的分段存储。这样有利于程序维护，实现模块化设计。

在自举表的最后，是连续的 4 个字节的全零数据，其目的是为了告诉 DSP 程序引导完成，可以转至程序入口执行，同时 DSP 也发出指示给外部存储设备告知引导结束。在 I²C 模式中，作为主机接收设备的 DSP 信号将会在接收到结束标志后在数据总线上给出停止标志，用来结束数据传输。

要建立自举表，可以利用 TI 提供的 HEX 转换程序(HEX55.exe)，将生成的链接文件转换成用于存储器的数据格式[3]。首先，需要建立一个 CMD(链接命令文件 linker command file)文件，输入需要的链接选项，HEX55 利用该文件提供的各种选项来转换文件。下面是一个为 I²C 引导方式建立的 CMD 文件和其选项以及具体含义：

```
Boot_in.out          /* 要转换的链接文件，文件名用户自定义 */
-boot                /* 转化成自举表的形式 */
-a   /* 输出 ASCII 码文件格式(其他文件格式见文献[4]) */
-e startaddr         /* 引导完成后程序入口地址，startaddr
                        是用户定义的 16 进制数据地址 */
-v5510:2             /* 自举表格式版本，早期的 HEX55 程序会
                        将其转换成以前的版本格式 */
-o my_prog.hex       /* 输出文件名，文件名用户自定义 */
```

通常情况下还需定义采用某种方式如 -serial8、-parallel16等选项，表示采用何种 Boot 方式从而生成对应的存储格式，由于采用了 I²C 模式来引导，所以这些选

项可以不使用。另外还可以使用－reg _ config 和－delay 选项,分别来设置需要改变的寄存器值以及需要延迟的 CPU 周期数。最后需要注意的是 HEX55 程序要使用 v2.1及后续版本,早期版本生成的自举表不能正确引导程序。

3　ARM 端设计

上面介绍了利用 I²C 接口的 E²ROM 来实现引导装载的硬件连接和需要的数据存储形式。实际利用 ARM 自身的 I²C 控制器,将自举表存储在 ARM 的 FLASH 中,并且让 ARM 按照 I²C 引导模式中 E²ROM 的时序向 DSP 发送对应的数据,实现对 DSP 的引导装载。所选用的 ARM 是 Philips 的 LPC2138。LPC2138 是基于 32 b ARM7 的内核,内部拥有多达 512 k 的高速 FLASH 和 32 k 的静态 RAM,其工作频率可达 60 MHz。其 I²C 总线控制器支持 I²C 所有工作模式,这样在用于引导 DSP 时就不用使用端口来模拟 I²C 时序,使用十分方便。引导 DSP 时,只要 ARM 按照对应的顺序来发送数据,就能实现 DSP 的程序引导。使用 ARM 引导时,只需将图 1 中的 SCL 和 SDA 分别与 ARM 的 SCL 和 SDA 连接即可。

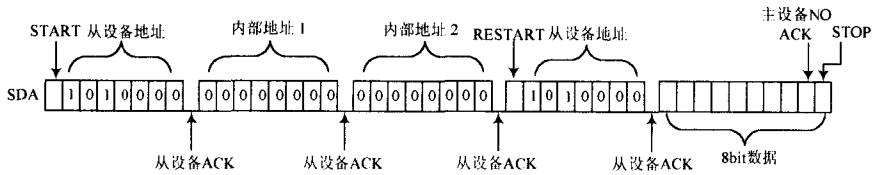

图 2　读数据时序

Bootloader 使用 I²C 读取数据时其时序如图 2 所示。引导开始后,DSP 首先会使用随机读取指令(random read command)从 0x0000 地址处读取数据,该读取指令由一个虚假的写指令和当前地址读取指令(current address read command)组成。ARM 正确响应该指令后,DSP 便继续采用当前地址读取指令读取剩余数据。

在 LPC2138 中,I²C 总线有专门的控制器,并且每次接收到数据后对应的 I²C 状态寄存器会以不同的代码来表示当前 I²C 总线状态,用户可以根据不同的状态来进行下一步的操作[3],整个引导过程就是 ARM 根据不同的总线状态来发送或接收相应的数据。使用 ARM 引导 DSP 程序加载时,ARM 作为从设备工作,在两种工作模式之间切换,分别为从设备接收(slave receiver 和从设备传输(slave transmitter)。

下面介绍 ARM 端程序的运行状况。程序中首先通过 I²C 地址寄存器(I2ADDR)将 ARM 的从设备地址设置为 0x50,再利用 I²C 置位控制寄存器(I2CONSET),将其中的 I2EN 和 AA 置 1,这样 ARM 就工作在从设备模式。一旦 I²C 总线接收到有效数据,程序就进入到中断服务程序中运行,用户程序根据 I²C 状态寄存器(I2STAT)的值判断当前状态从而进入下一步操作,图 3 为中断服务程序工作流程。

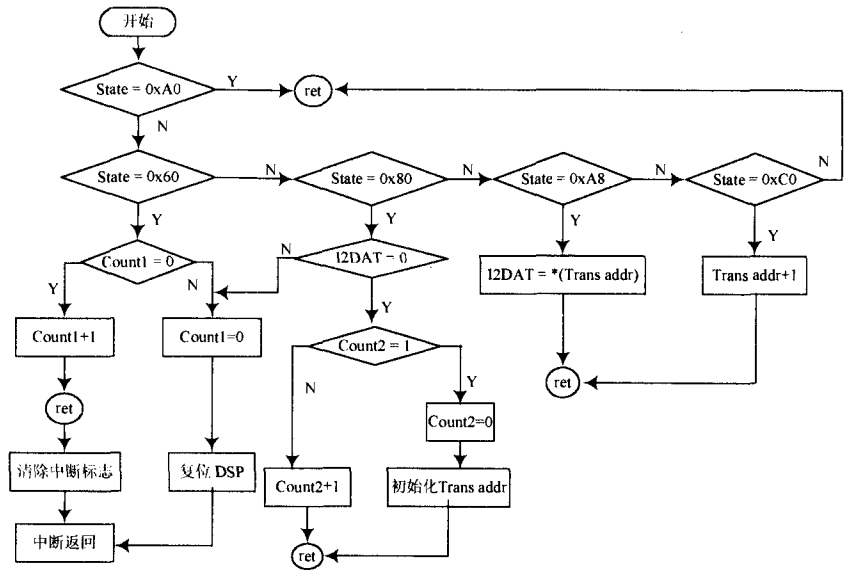

图 3　I²C 中断服务程序

图 3 中 State 代表接收到数据后,I²C 状态寄存器中的值,不同的值代表总线上可能出现的各种状态。0xA0 代表程序收到的是 start 或是 stop 标志,返回主程序继续等待中断。0x60 代表接收到自己的从设备地址和写命令,并返回 ACK 信号。Count1 代表收到的写命令次数,由于整个引导过程中只能接收到一次写命令,所以只要 Count 大于 1 则接收出错,需要重新启动引导程序,利用 ARM 控制 DSP 的复位信号重新开始引导过程,直至成功引导。0x80 表示 ARM 进入了从设备接收数据状态,并且前边已经收到了本机的设备地址,此次接收到了数据并返回 ACK 信号。I2DAT 在从设备接收模式时存储接收到的数据,发送模式时存储待传输的数据。引导开始后,连续两次接收的数据应该为 0,若并不为 0,表示引导程序出错,需要复位 DSP 重新开始接收。

Count2 为连续两次接收数据 0 的计数器,一旦满足条件,将发送缓冲区的首地址取出存储在 Trans _ addr 中。0xA8 代表接收到当前地址读取命令,一旦接收到此命令,将待发送数据取出送入发送数据寄存器 I2DAT,以便下一次传输时将数据送出。0xC0 表示数据发送成功,而且

(下转第 85 页)

(2) 接受子代理注册 MIB 域的请求;

(3) 接收和发送 SNMP 协议消息;

(4) 实现除管理操作以外的代理应该实现的功能;

(5) 提供对与管理框架相关的 MIB 对象的支持;

(6) 根据子代理注册的 MIB 域,发送和接收 AgentX 协议消息来访问管理信息;

(7) 转发由子代理产生的通知信息。

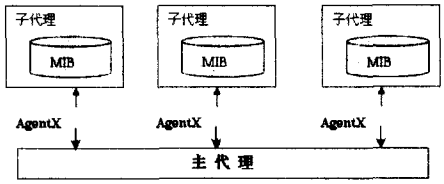

图 4 基于 AgentX 协议的代理结构

子代理需要实现以下功能:

(1) 与主代理建立 AgentX 会话;

(2) 向主代理注册 MIB 域;

(3) 实例化管理对象;

(4) 将其注册的 MIB 域中的对象标识符与实际管理变量进行绑定;

(5) 对管理变量进行管理操作;

(6) 初始化通知信息。

6 系统实现

本系统采用标准的 SNMP 协议和纯 Java 语言,这样有利于保证系统的平台无关性和通用性,并分别在 Windows 2000 及 Linux 平台上实现了该系统。测试的环境分为 3 种:系统的 C 端和 S 端运行在同一个主机上;系统的 C 端和 S 端运行在 2 个以上主机上,但是所有主机使用的是同一种平台(如:Windows 2000);系统的 C 端和 S 端运行在 2 个以上主机上,并且各个主机使用的是不同平台(如:Windows 2000,Linux 等)。经过测试,验证了该仿真系统的平台无关性和通用性,运行在不同平台上的仿真系统 C 端和 S 端能够很好的通信,完成仿真模拟功能。

7 结 语

我们开发的 SNMP 仿真系统已经成功地应用于教学中,曾作为学院网络管理课程的优秀教学课件被多次使用,获得了师生的一致好评。考虑到教学的灵活性和学生学习的自主性,该系统支持单机运行和网络运行,既可以在机房通过局域网进行试验,也可以用一台机器进行试验,易于使用和管理。该系统的下一个版本,将与 Web 技术相结合,通过 Internet,用户可以远程查看 SNMP 管理端和代理端的工作过程,进一步扩大该系统的可利用价值。

参 考 文 献

[1] 叶绿,叶红. 基于 SNMP 的上网浏览的信息监控系统的研究与实现[J]. 计算机应用,2004,24(z1):34-35,44.

[2] 杨家海,任宪坤,王沛瑜. 网络管理原理与实现技术[M]. 北京:清华大学出版社,2000.

[3] 朱思峰. 基于 SNMP 的由器流量监控系统的设计及实现[J]. 现代电子技术,2005,28(13):35-36,39.

(上接第 82 页)

没有收到 ACK 信号,意味着当前地址读取命令结束,此时将发送缓冲区地址加 1,取出下一次待发送数据地址。这样便完成了 1 个字节数据的发送。整个引导过程一直到 DSP 收到自举表结束标志后停止。需要注意的是,I^2C 中断标志位需要通过软件清除,每次中断返回时都必须用 I^2C 清零控制寄存器(I2CONCLR)手动清除 I^2C 控制寄存器中的中断标志。

按照上述方法就完成了 I^2C 引导装载模式,用户可以在程序中加入测试程序,通过控制 GPIO 高低变化生成脉冲,利用示波器观察从而判断程序引导是否成功。

4 结 语

本文提出的引导方式已经成功地应用于一款低功耗、小型户数传设备当中。由于 DSP 的程序存储在 ARM 中,因而可以利用 ARM 的扩展接口,将程序下载至 ARM 中,免去了对外部存储器的编程,特别有利于设备的升级和维护。

参 考 文 献

[1] Texas Instruments, Using the TMS320VC5503/VC5507/VC5509/VC5509A Bootloader, 2004.

[2] Philips Semiconductors. The I^2C-Bus Specification Version 2.1, 2000.

[3] 张勇,陈天麒. C/C++ 语言硬件程序设计——基于 TMS320C-5000 系列 DSP[M]. 西安:西安电子科技大学出版社,2003.

[4] Texas Instruments. TMS320C55x Assembly Language Tools User's Guide (Rev. G), 2003.

[5] Philips Semiconductors. LPC213x User Manual, 2005.

[6] 张克满,何格夫. 用虚拟机 I^2C 总线技术实现 SAA7111 的初始化[J]. 国外电子元器件,2005(1):26-29.