基于 ARM7 和 uClinux 系统的 USB 主机控制器的设计与实现*

吴仁伟[1],王光明[1],梁德祥[2],邢汉承[1]
(1. 东南大学 计算机科学与工程学院,江苏 南京　210096;2. 南京微辰信息有限公司,江苏 南京　210016)

摘　要:针对嵌入终端产品中大多不具备 USB 功能,提出了一种基于以 ARM7 处理器 S3C44B0X 和 USB 主控芯片 CY7C67200 为核心的可扩展设计方案,并结合 uClinux 操作系统给出了相应的软硬件设计方法,通过实际测试分析了该方案的性能及可行性。

关键词:嵌入式系统;S3C44B0X;USB 主机控制器;uClinux
中图分类号:TP368.1　　**文献标识码**:A　　**文章编号**:1671 - 5322(2007)01 - 0037 - 05

USB(通用串行总线)是一种广泛使用于个人计算机及个人消费电子行业的标准智能型串行接口。它的优势在于其即插即用的方便性。外接设备一旦连接到 USB 接口,就可被自动反应并正确识别,不需要设置开关或跳线,同时不必重启系统它就可以工作。此外,通过增加集线器,主机上还可以添加更多的端口,为更多的外围设备提供空间。USB 接口由于其通用性好、传输方式多样、成本低、支持即插即用、易于扩展且便于使用,使得 USB 接口技术在 PC 机和嵌入式系统中得到广泛的应用[1]。

ARM 是一种低功耗、高性能的 32 位嵌入式微处理器,在各类嵌入式产品中占有较大的市场份额。各个 CPU 生产厂家通过取得 ARM 公司的授权使用 ARM 核来生产各种 CPU,目前市场上基于 ARM 的嵌入式微处理器多数不具有 USB 功能,鉴于 USB 控制器的诸多优点,考虑通过增加额外的芯片作为扩展模块,为这类系统提供 USB 功能。

1　USB 系统概述

一个基本的 USB 系统可以分为 3 个主要部分:即 USB 主机、USB 设备和 USB 连接[2]。USB 连接是指一种 USB 器件和 USB 主机进行通信的方法,主要包括总线的拓扑、USB 系统各层之间的关系、数据流动的模式以及 USB 的调度机制。

USB 主机控制器是 USB 设备连接主机系统提供主机接口的部件,它是一个由硬件、软件和固件组成的复合体。USB 设备可以分为两种:即用于扩张设备连接 USB 集线器和可以为主机系统提供某种用途的 USB 功能器件。USB 线缆用于实现外设到主机或 USB 集线器连接,它由 4 根线组成,其中一根是电源线 VBus,一根是地线 GND,其余两根是用于差动信号传输的数据线(D + 、D -),而且 D + 、D - 具有检测设备速度功能。

在软件和硬件层次上,USB 主机都处于 USB 系统的核心,它不仅包含了和 USB 外设进行通信的 USB 主机控制器及用于连接的 USB 接口,更重要的是主机系统是 USB 软件系统的载体。任何一次 USB 传输都是有 USB 主机发起和控制的,USB 设备只有和 USB 主机建立连接才能完成相应的功能,USB 系统的这种主从式结构决定了两个 USB 设备或是两个 USB 主机之间不能完成数据通信。同时 USB 主机上的电源管理系统可以支持设备的挂起和远程唤醒等高级特性,对于总线上长时间不用的设备将其设置为挂起状态,等到有数据传输或是其它动作时再唤醒设备并执行相应的 USB 操作[3]。

* 收稿日期:2006 - 11 - 10

作者简介:吴仁伟(1982 -),男,河南省信阳市人,硕士研究生,主要研究方向为嵌入式系统及其应用。

2 系统设计方案

为了实现USB的各种功能,必须实现USB协议中规定的各种控制需求,同时又省去USB硬件设计部分带来的复杂性,可以采用高度集成化的具有USB功能的芯片来达到这个目的,本文采用了基于ARM7的嵌入式处理器S3C44B0X和Cypress的CY7C67200连接的方案。

2.1 CY7C67200的特点

CY7C67200是Cypress公司生产的新一代USB Host/Slave接口控制器,带有48 MHz主频16位RISC MCU、并且支持全速USB2.0 OTG协议,它的双功能端口即可作为USB主机又可作为全速或低速的USB设备,同时支持存储设备类;内部集成两个USB接口,每个接口都可以设置成主机或设备;附带4K*16b掩模ROM、8K*16b程序/数据RAM,I2C、UART、HSS、SPI、HPI等接口。由于CY7C67200内部带有16位的RISC MCU,它可以独立作为USB主机使用,通过使用其提供的外部接口,又可以作为其他处理器的协处理器使用,本文采用的就是协处理器方式[4]。

2.2 S3C44B0X的特点

S3C44B0X微处理器是Samsung公司为低成本、低功耗的应用产品而设计的,可以使用在移动手持终端设备和互联网产品中。该芯片采用0.25 μm CMOS工艺和SAMBA11总线结构(SAMSUNG ARM CPU嵌入式控制器总线结构)设计,核心逻辑部件建立于ARM公司的ARM7TDMI RISC处理器上,最高运行时钟频率达66 MHz,并带有8kB的指令和数据Cache,由于ARM7DMI中没有集成存储管理部件,特别适用于uClinux和uCos操作系统[5]。

2.3 设计原理

CY7C67200提供了HSS、HPI、SPI三种可以作为协处理器的工作模式,其中HPI接口方式提供了外部处理器通过DMA方式来访问其内部Ram的能力,通过这种方式大概提供了16MB/s的数据处理能力,这个速度远远高于其它两种方式的处理速度(HSS、SPI提供最大不超过2MB/s的速度)。HPI的DMA处理能力是由其内部提供的寄存器(数据、状态、地址和mailbox寄存器)来实现的,其中mailbox寄存器的引入解决了外部处理器和CY7C67200之间的数据通信问题:如果中断路由寄存器中开启CY7C67200向外部处理器路由信息功能,当外部处理器将数据准备好数据时向mailbox寄存器写入对应的消息,将产生HPI mailbox RX Full中断通知CY7C67200内部的处理器进行相应的处理;当外部处理器从mailbox寄存器读出消息时,将产生HPI mailbox TX empty中断通知CY7C67200内部的处理器。而且当CY7C67200向mailbox寄存器写入消息时,会通过HPI接口产生中断通知外部处理器来读此消息。

完成对HPI接口中4个寄存器的寻址需要2根地址线,CY7C67200内部提供了2根地址线来完成此功能,通过将CY7C67200映射到S3C44B0X的一个存储空间上,访问这些寄存器的操作被简化为对存储器的读写。

3 主机控制器驱动程序

3.1 uClinux USB主机协议栈

uClinux USB主机协议栈可以分成3个主要层次:USB设备驱动、USB Core和USB主机控制器驱动程序(HCD)[5]。USB设备驱动程序确定虚拟连接,配置并同设备进行交互。设备驱动将数据装配成USB传输请求块并使用由一组通用的数据接口(API)将数据传输请求提交给USB Core;USB Core是USB协议栈的中间层,位于设备驱动(上层)和HCD(下层)之间。USB Core处理设备的枚举与配置,按需要安装和卸载设备驱动程序,并为设备驱动程序和主机控制器驱动程序各自提供一组标准的接口函数,系统上任何USB设备和主机的驱动程序必须提供这些接口函数的具体实现;主机控制器驱动程序是主机控制器的硬件抽象层,它向USB协议栈隐藏了具体控制器的硬件控制实现细节。HCD从USB Core接收USB请求,然后负责解析这些请求并创建USB传输事务,当整个系统的带宽允许时发起USB传输并将数据通过USB总线送到目的地,图1所示为uClinux系统上USB协议栈结构图[6]。

3.2 主机控制器驱动程序

USB主机控制器驱动程序处于Linux的USB协议栈的最底层,为CY7C67200提供支持。主机控制器驱动程序实现一组由USB Core定义的标准接口函数(API)与USB Core通信。CY7C67200驱动程序需要完成以下功能。

3.2.1 主机控制器初始化

USB主机控制器驱动程序必须进行必要的设置才能使USB Core和CY7C67200处于正常工作

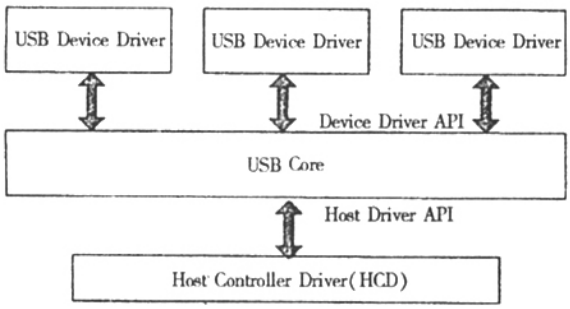

图 1　USB 协议栈结构

Fig. 1　Linux USB Host Stack

状态。驱动程序中初始化过程如下:

(1)对 CY7C67200 复位,并根据硬件连接方式设置其工作模式(HPI、HSS、SPI 等)。

(2)初始化 CY7C67200 内部寄存器并为每个端口注册一个。

(3)向 USB Core 注册主机控制器驱动程序和 USB 总线接口。

(4)初始化中断为电平触发,同时向系统申请中断号并向 USB core 注册中断处理函数。

(5)设置各个 SIE 的初始状态,并对 SIE 的每个端口初始化,包括每个端口需要用的数据结构、所属总线以及每个端口上附带的 URB 队列链表的初始化。

(6)为主机控制器上的每个端口创建根虚拟集线器并将这个集线器注册到 USB Core 中。

经过上述的初始化,建立了 CY7C67200 的初始运行状态。

3.2.2　虚拟根集线器

按照 USB2.0 标准,主机控制器承担根集线器的作用同时提供多端口支持。这个根集线器功能在 HCD 层实现。根集线器的功能可以分成以下几个部分:根集线器标识,设置和取消根集线器的特征,端口控制和端口状态查询。

对于 USB 系统来说,集线器本身是一个设备,而一个设备必须有设备描述符,因此 HCD 必须提供一组描述根集线器特征和配置的描述符,USB 协议栈在初始化根集线器时使用这些描述符来配置根集线器。这些描述符的具体信息请参照 USB2.0 标准;当 USB 协议能够正确识别根集线器以后,HCD 可以设置或者清除根集线器的特征。这些特征可以是某个特定端口的,也可以根集线器本身的,包括复位端口、使能端口、挂起端口、改变端口状态;根集线器允许 USB 协议栈发送 USB 请求来控制各个端口,可以进行的操作如下:复位端口、使能/禁止端口、挂起/恢复端口;HCD 还需要监控和更新 USB 端口的状态,HCD 使用一个变量来监测端口状态,另一个变量来监测端口状态的变化。当端口状态变化时,HCD 更新端口状态变量和端口状态改变变量。HCD 使用一个单独的处理程序来轮询端口的状态变化。当端口状态变化了,HCD 就通知 USB 协议做对应的处理。

3.2.3　USB 请求管理

Linux USB 协议栈为了方便数据管理和层间数据交互,在软件层次上定义了一个数据结构封装了 USB 传输请求,称为 USB 传输请求块(USB Request Block),简称 URB[3]。CY7C67200 驱动程序中为了区分 USB 协议中规定的传输类型,维护了 5 个活动的 URB 传输队列,分别是控制传输 URB 队列、中断传输 URB 队列、批量传输 URB 队列、实时传输 URB 队列和待删除 URB 队列。

CY7C67200 的驱动程序中实现了由一系列 USB 协议栈提供的接口函数,其中留给设备驱动提交数据传输请求的接口函数是 sumit_urb,用户提交的数据请求交给客户端驱动程序,客户端驱动程序根据设备端请求调用 submit_urb 将 URB 提交给 USB core,再由 USB core 转交给主机控制器驱动程序,主机控制器驱动程序根据下列步骤将其挂接到各个 URB 队列中:(1)从 URB 中得到提交该 URB 的设备在总线上的地址;(2)如果地址是虚拟根集线器的地址则交给根集线器驱动程序处理;(3)如果地址说明该 URB 属于附接的设备则从 URB 中提取端点信息;(4)CY7C67200 的驱动程序将 URB 挂接到各个端点的 URB 队列中,同时如果该端点的 URB 队列为空,HCD 就将其转交到活动的 URB 链表中;(5)在主机控制器产生 SOF 以后,HCD 负责将活动链表中的 URB 装填到帧中并处理这些数据,图 2 所示为提交 URB 算法流程。

3.2.4　数据调度

USB 协议中规定的一次 USB 传输的时间称为一帧,时间是 1ms,数据传输发生在帧中。帧开始是由控制器产生 SOF 来标记的。每帧中传输包括 USB 协议中规定的 4 种事务类型,4 种事务按如下顺序进行调度:实时传输、中断传输、控制传输和批量传输。这些传输的带宽分配是在 USB core 中完成的,实时传输最多占用 90% 的带宽,中断传输最多占用 10%,控制和批量传输使用剩

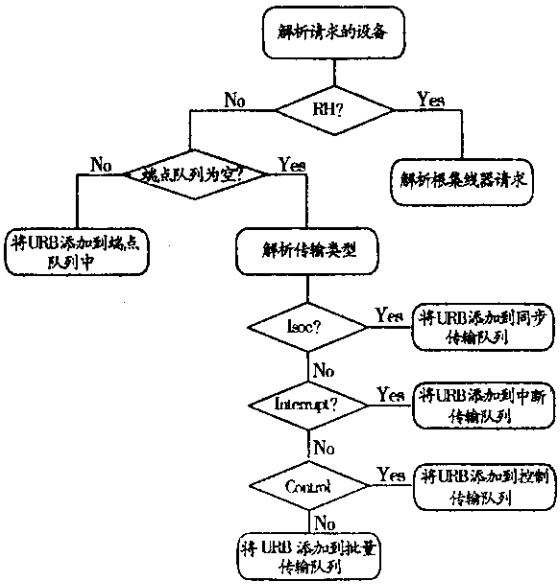

图2　Linux USB 主机堆栈

Fig. 2　Linux USB Host Stack

下的带宽。如果系统带宽不够用,协议栈不再接收设备驱动程序发出的数据请求。尽管 HCD 不必进行带宽管理,但是它必须为每帧创建一个传输描述符(TD)链表,TD 从前述的 USB 请求块中附带的信息转换而成,这些 TD 需要提交给 CY7C67200 来处理,它的结构必须符合 CY7C67200 的要求,CY7C67200 中使用的 TD 如下:

```
typedef struct td
 {
unsigned short baseAddr ;//数据基地址
unsigned short port_length ;//端口号/数据长度
unsigned char pid_ep ;//包 ID/端点号
unsigned char devAddr ;//设备地址
unsigned char control ;//TD 控制
unsigned char status ;//传输(完成)状态
unsigned char retryCount ;//重试次数
unsigned char flag ;//与数据相关的标志
unsigned short nextTDAddr ;//下一个 TD 的地址
} td_t ;
```

HCD 中用到的调度算法如下:1) HCD 扫描活动的实时 URB 队列,如果队列中存在 URB,通过比较 URB 结构中指定的开始帧是否跟当前的帧号是否相同来决定在哪一帧中传输;2) 同样道理,HCD 扫描中断传输 URB 队列,如果 URB 中的时间间隔超过了当前帧的时间那么它就要在当前帧中传输;3) 如果还有可用带宽,HCD 将转化控制和批量 URB 为 TD 并添加到这一帧中;4) 在 TD 添加完成以后,HCD 将整帧复制到 CY7C67200 内部的缓冲区中,CY7C67200 的硬件处理逻辑在 SOF 产生时进行数据传输。图 3 所示为调度算法流程图。

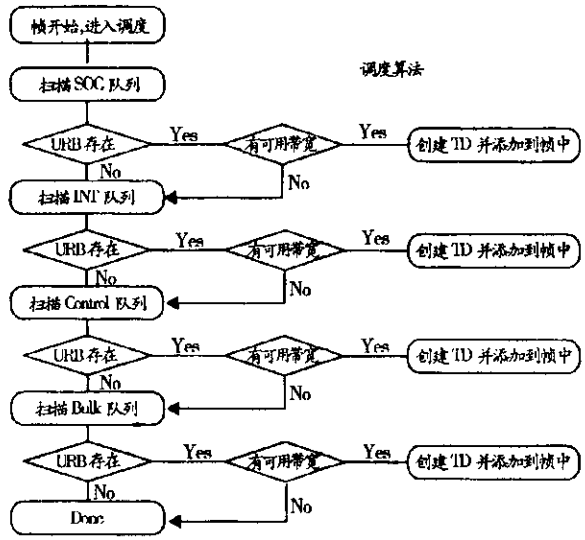

图3　调度算法流程图

Fig. 3　Schedule Algorithm Flow Diagram

3.2.5　中断处理

中断是 CY7C67200 和 S3C44B0X 相互交换信息的一种方式,主要用来处理事件通知,CY7C67200 驱动程序中主要处理了 3 类硬件中断:(1)设备检测中断;(2) SOF 中断;(3)“帧结束”中断。下面分别介绍几种中断的处理方式。

(1)设备检测中断

USB 设备的通常检测方法是当一个 USB 设备插入到一个 USB 集线器的某个端口时,USB 集线器硬件逻辑检测到其管脚 D + 或 D – 上的电平变化并由此得到设备速度,并在下一次主机通过中断交互查询时将设备状态向主机报告。主机查询到集线器端口的状态变化后,会调度若干个控制交互获取信息并向集线器发出打开端口的命令。通过这些交互传输新插上的 USB 设备就出现在 USB 总线上了。CY7C67200 提供了反映设备状态的寄存器,可以通过读取这个寄存器来判断是否有设备连接在其下游端口上。CY7C67200 的驱动程序中采用的设备检测方法是在每个端口中注册一个定时器,定时器每隔一段时间对端口发出一次复位中断,由对应的中断处理函数来读取设备状态寄存器。

(2) SOF 中断

SOF 中断每毫秒产生一次。这个中断用来处

理前一帧所有的数据传输,包括从 CY7C67200 内部的存储区中拷贝数据到 S3C44B0X 内存中并调用已经完成的 URB 的回调函数。

(3)“帧结束”中断

USB“帧结束”中断发生在 CY7C67200 将一帧数据传输到 USB 设备之后,HCD 需要按如下过程处理:①从 CY7C67200 中得到传输描述符(TD);②HCD 对每个 TD 进行错误检测,如果存在错误就进行相应的处理;③HCD 从活动 URB 队列中取下 URB,将 URB 转换成需要在下一帧中处理的 TD 并提取出数据;④HCD 将 TD 和数据拷贝 CY7C67200 中;⑤CY7C67200 将在下一帧处理这些 TD。

4　性能分析及结论

表 1　正交试验表

Table 1　Experimental Data Table

传输方式	Bulk only	ISO only
传输速度	700 ~ 800KB/s	< = 1MB/s

如表 1 中所示,CY7C67200 的传输速度跟所用的传输类型有很大关系,其中 Bulk only 是系统挂接高速 U 盘时的传输速度,U 盘上使用 fat32 文件系统,实际测试时拷贝文件的速度达到 700 ~ 800kB/s,速度的变化跟当时的系统负载有一定关系,如果系统上其他任务占用了较多的 CPU 时间,系统的传输速度稍慢一些。在移植了一个 USB 摄像头以后,测试了 CY7C67200 仅采用 ISO 方式的数据传输速度,USB 摄像头发送 ISO 类型的数据包来捕获图像,在 URB 的回调函数中计算得到的数据大小并用应用程序统计数据总量,得到的数据传输速度不超过 1MB/s。两类传输都接近于 USB1.1 协议中规定的全速设备的极限传输速度。

随着嵌入式系统的广泛应用,USB 设备接口也将广泛应用于各种数字产品中,本文为这类产品提供了一种可行方案,CY7C67200 可以提供 Host 和 Slave 接口,同外部处理器的连接方式也相当简单,可以在本文论述的基础上稍加更改就可以应用于其他产品中。本文论述的方案已经应用于数字机顶盒中,用于扩展大容量存储设备并提供对 USB 人体学设备的支持。

参考文献:

[1] 王成儒,李英伟. USB2.0 原理与工程开发[M]. 北京:国防工业出版社,2004.

[2] Compaq, Hewlett - Packard. Universal Serial Bus Specification Revision 2.0[EB/OL]. http://www.usb.org/developers/docs/usb_20_05122006.zip. (2000 - 4 - 27).

[3] 毛德超,胡希明. Linux 内核源代码与情景分析[M]. 杭州:浙江大学出版社,2001.

[4] Cypress Electronics. CY7C67200 Users Manual[M/CD]. http://download.cypress.com/published content/publish/design_resources/developer_kits/contents/cy3663___ez_otg___ez_host_development_kit_20.pdf. (2003 - 5 - 28).

[5] Samsung Electronics. S3C44B0X RISC MICROPROCESSOR Users Manual[M/CD]. http://www.samsung.com/Products/Semiconductor/MobileSoC/ApplicationProcessor/ARM7Series/S3C44B0/S3C44B0UsersManaual.pdf. (2002 - 12 - 09).

[6] MindShare, Inc Don Anderson Dave Dzatko. Universal serial bus system architecture[M]. 北京:中国电力出版社,2003.

The Design and Realization of An USB HOST Controller Based on ARM7 and uClinux System

WU Ren - wei[1], WANG Guang - ming[1], LIANG De - xiang[2], XING Han - cheng[1]

(1. School of Computer Science and Engineering, Southeast University, Jiangsu Nanjing　210096, China;
2. Nanjng Micro Aurora Information Technology Co. Ltd, Jiangsu Nanjing　210096, China)

Abstract: Since most Embedded terminators have no USB interface, this paper puts forward an extended scheme based on S3C44B0X and CY7C67200. Software and hardware methods are given interms of uClinux OS. Also, some tests have done to analysis the performance and feasibility.

Keywords: Embedded System; S3C44B0X; USB Host Controller; uClinux