第 34 卷　第 3 期　　计　算　机　工　程　　2008 年 2 月
Vol.34　No.3　　**Computer Engineering**　　**February 2008**

·工程应用技术与实现·　　文章编号：1000—3428(2008)03—0269—03　　文献标识码：A　　中图分类号：TP311.5

一种 USB 从设备访问主设备的方法

张　志，余松煜

（上海交通大学通信工程系，上海 200030）

摘　要：介绍了一种 USB 从设备访问主设备的方法。该方法利用 USB 从设备端存储器中建立的交换缓存区，无须拔插 flash 存储卡即可实现 USB 从设备端对主设备端文件的直接访问，从而使 USB 从设备可访问远大于自身存储空间的文件。通过该方法可实现手持播放器以低成本和省电的方式扩展外接硬盘。

关键词：USB 总线；USB 从设备；USB 主设备；直接访问；存储器

File Access Method from USB Device to USB Host

ZHANG Zhi, YU Song-yu

(Communication Engineering Department, Shanghai Jiaotong University, Shanghai 200230)

【Abstract】 A file access method from USB device to USB host is introduced. By using switch buffer built in memory of USB device, this method realizes the direct access to the file in USB host from USB device, which makes USB device access much larger external file than its storage capability. This method makes it possible that a portable player extends external harddisk with low cost and power saving.

【Key words】 USB bus; USB device; USB host; direct access; storage device

随着集成电路技术的发展，基于手持设备的视频播放器日益普及，由于内部Flash存储颗粒成本较高，因此外扩硬盘成为一种迫切的需求。但基于成本考虑，这些手持播放器普遍只带USB从设备端口[1]，需要拔插flash存储卡到专用硬盘上复制文件，并且要求存储卡容量足够容纳整个视频文件。本文提出一种USB从设备访问USB主设备的方法，使用支持USB主设备的大容量存储器(硬盘)，在从设备存储器中建立交换缓存区，让操作者在从设备端操作；从设备端上层软件把其对于主设备存储器文件的访问“请求”按一定的协议写在自身存储器的一个“请求”缓存区里(文件或特殊存储扇区)。然后把存储器的控制权交给主设备端，主设备端读取从设备存储器中的这个“请求”缓存区，并执行“请求”；文件管理“请求”进行文件的复制、移动、删除等操作；对于文件“读请求”则把主设备端的文件块复制到从设备端交换缓存区中的缓存块，对于文件“写请求”则把从设备端交换缓存区中的缓存块复制到主设备端的文件中；当从设备再次获得其存储器控制权时，要访问的文件已经全部或部分在存储器中，从而实现对其的读写访问[2]。

1　USB 主从设备硬件架构

USB 主从设备硬件架构如图 1 所示(虚线代表逻辑连接)。

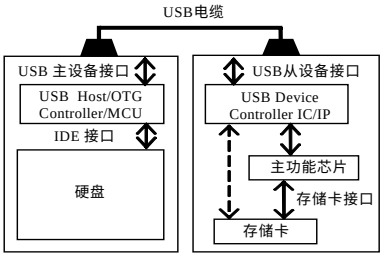

图 1　硬件框架

主设备将手机、PDA、PMP、数码相机等USB从设备统一识别为一个UMS(USB Mass Storage Class)存储设备[1]，然后交换硬盘中的文件与USB从设备中的文件。本方法基于该硬件架构，通过软件使USB从设备端的主功能芯片成为主控者、USB主设备由主要控制者变为从控制者。

2　软件框架

USB 从设备访问 USB 主设备的方法基于常用的 USB 硬件架构，完全通过软件实现。其软件系统框架与数据流图如图 2 所示。

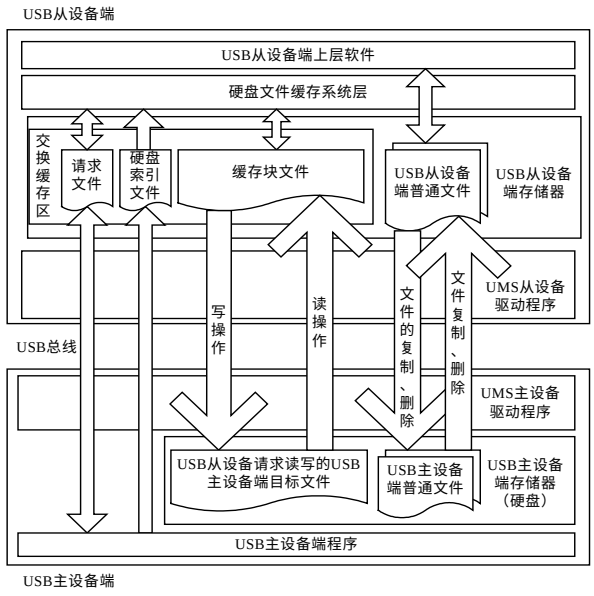

图 2　软件框架与数据流图

作者简介：张　志(1976－)，男，高级工程师，主研方向：视频通信，电子系统设计，手持多媒体产品设计；余松煜，教授

收稿日期：2007-03-20　**E-mail：**zhi_zhang@witchain.com

软件的核心是 USB 从设备存储器中建立的交换缓存区，由配制文件、请求文件、硬盘索引文件、缓存块文件等组成，USB 主从设备均可访问该交换缓存区，USB 从设备把操作请求写在请求文件中，USB 主设备对其读取并执行，而大文件的传输依赖于缓存块文件的交换。

3 USB 从设备端软件

USB 从设备端软件顺序读的流程如图 3 所示。由于手持播放器上的视频播放器、MP3 音乐播放器、照片浏览器、文字阅读器等软件主要的操作都是顺序读，或者以顺序读为主要方式、随机读为次要方式，没有写操作，因此，本文仅对顺序读进行详细描述。

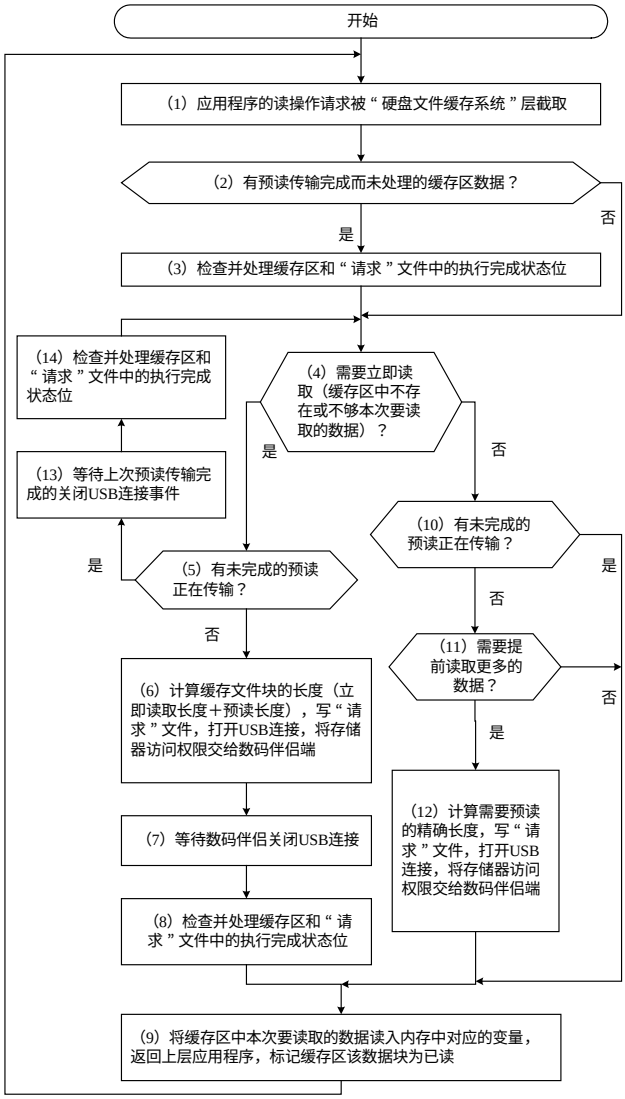

图 3　USB 从设备端应用程序顺序读流程

USB2.0 高速(480 Mb/s)从设备控制器由于成本低而普遍应用于各种手持播放器设备，相对于在手持播放器上使用较贵但却只支持全速(12 Mb/s)的USB1.1 主设备控制器[2-3]来扩展外接硬盘的方式而言，以USB从设备直接访问USB主设备的方式更优，除成本低以外，其传输速率也更高，避免了硬盘一直工作，而是以高速传输文件块之后即让硬盘停转，达到省电的目的。

缓存区数据交换将不可避免地带来传输延迟，因此，要实现连续播放，需要用比较好的算法来进行大缓存预读。以视频播放为例，一种实现预读的方法是对每一个视频播放器的流程进行仔细分析，然后将读取文件的提前量写入播放器程序的每一个过程。但这种方法必须熟知各种视频编码格式的规范，而编码格式种类繁多、差异较大，其实现的工作量很大，因此，本文提出一种智能判断各文件读取速度的预读算法，该算法基于所建立的“硬盘文件缓存系统”层，把所有文件操作函数重定向到该层，然后调用对应的文件操作函数即可对硬盘文件进行预读、传输、从缓存中读取数据块等操作。以 C 语言视频播放器为例，在所有源文件的头文件最前面插入以下重定向宏定义：

```
#define fopen bufferSystem_fopen
#define fclose bufferSystem_fclose
#define fread bufferSystem_fread
#define fwrite bufferSystem_fwrite
#define fget bufferSystem_fget
#define fput bufferSystem_fput
#define fseek bufferSystem_fseek
```

然后重写各文件操作函数，如果是本地文件操作，则调用本地文件操作函数，如果是硬盘文件操作，则采用预读的算法提前从硬盘传输到本地存储器缓存区，从缓存中读取数据块返回上层应用程序，以下是其中 2 个函数的例子：

```
FILE * bufferSystem_fopen (const char *file, const char *flags)
{ …
 if（isFilePathLocal（file）==TRUE）
        return fopen (file, flags)；
 else  {
return hardisk_fopen (file, flags);     //在这里返回一个内部管理
//文件句柄，并记下该文件句柄，以后凡是使用该句柄的文件读操作
//都要受预读算法管理   }}
int bufferSystem_fread(unsigned char * buf, int size, int count,
FILE *fd )
{ …
   if（isTheFileLocal（fd）==TRUE）
        return fread(buf, size, count, fd);
 else  {
   return hardisk_fread(buf, size, count, fd);   //采用预读的算法
//提前从硬盘传输到本地存储器缓存区，从缓存中读取数据块返回上
//层应用程序   }
}
…
```

其他函数结构类似，这里不再赘述。

预读是算法的核心，其难点在于预读时间提前量的计算，下面就各相关因素进行分析。

(1)总缓存区大小

缓存区设置得越大，就能缓存越多的数据、提高缓存命中率（当某一次读取文件数据时，数据已预读到缓存区，则称“缓存命中”，如果文件还在硬盘，需要立即读取，造成应用程序的等待，则称“缓存未命中”）、减少传输次数、减少硬盘启动次数、让系统省电。但是，缓存区也不能无限制开大，因为直接访问的优点之一，就是要在“USB 从设备端”存储器空间有限的情况下访问大文件。缓存区占用空间越小，供其他应用使用的空间就越多。而且很多数码相机、手机、MP4 随机赠送的卡只有 16 MB 或 32 MB，想要在这些卡上直接使用，就更要控制缓存区大小，因此，缓存区的大小尽量不要超过 16 MB。

(2)单个缓存块大小/缓存块个数

在缓存区总大小确定的情况下，单个缓存块个数越少，

即单个缓存块越大，传输次数就越少，优点是硬盘启动次数减少、系统省电，但缺点是不够灵活，当同时有多个通道分别在文件不同位置顺序读写时(例如一个视频通道从文件头开始读，2 个音频通道从文件中部开始读)，就会造成缓存频繁切换的抖动，因此，必须有足够多的缓存区。一个折中的方法是将单个缓存块开小一些，根据一个通道读取的速度来决定单次传输缓存块的个数，即读得快的通道一次多传几个缓存块，而读得慢的通道一次少传几个。一般来说，如果视频码率是 1 Mb/s，1 min 的数据量为 60 s×1 Mb/s=60 Mb=8 MB，音频码率如果是 64 Kb/s 的话，双通道 1 min 数据量为 2×60s×64 Kb/s=7.5 Mb=1 MB，如果以 1MB 为单个缓存块大小，音频占一个，视频占 8 个，1 min 启动一次传输，传输时间大约要 10 s。这样，设置 1 个 15 MB 的总缓存区，将其分为 15 个 1 MB 的缓存块。实际上还可以根据各手持播放器所支持的视频码率进行调整。

(3)预读的时机

预读的时机是根据实际读取的速度来计算提前量的，比如要 1 min 进行一次传输，已知 1 Mb/s 的视频通道与 2 个 64 Kb/s 的音频通道，1 min 的数据量分别是 8 MB 和 1 MB，传输时间 10 s，则提前 30 s 开始传输应该来得及，30 s 时剩余的未播放数据为 4.5 MB，总缓存区为 15 MB，正好可以留出 10 MB 给后 1 min 存放缓存数据。也可以根据各手持播放器所支持视频码率的实际情况进行调整。

(4)初次传输与启动时间

预读的一个问题是最初播放时的第 1 次传输，没有读取历史速度的依据，甚至没有文件头，只有文件名和文件大小可供参考，所以难以选择初始传输的大小。要想尽快开始播放，就要尽可能少地传输，但传输过少，万一遇上大码流的视频文件，则播放之后没多久就会出现停顿，必须等待第 2 次传输的完成。所幸一般手持多媒体播放器所能支持的最高码流都是一定的，不会经常变化，而本文的直接访问技术需要针对播放器单独开发和调试，并非支持所有机型，因此，可以根据实际硬件性能来调试定制初始的 2 次传输的大小，而且只要对最高码率的视频调试出一个不会造成播放后停顿的最短启动时间即可。一般对于 1 MB 的缓存块、1 Mb/s 的码率，第 1 次传输一个缓存块，第 2 次预读 5 个~8 个缓存块，以后根据实际读取的速度来计算。

图 3 主要步骤在各种情况下的走向如下：

(1)第 1 次读/立即读/遇突然读取速度变快而读取未命中：

1→2→4→5→6→7→等待→8→9→1

(2)第 2 次读/正常预读：

1→2→4→10→11→12→9→1

(3)大部分情况下读取命中(数据已在缓存区)：

1→2→4→10→11→9→1 或 1→2→4→ 10→1

(4)正常预读完成：

1→2→3→4→10→11→9→1

(5)预读传输过程中遇突然读取速度变快而读取未命中：

1→2→4→5→15→等待→16→4→11→12→13→14→10→1

或发生更糟的情况(上次预读的还不够本次读取，等待 2 次)：

1→2→4→5→15→等待→16→4→5→6→7→8→等待→9→10→1

4 USB 主设备端软件

USB 主设备端软件流程图如图 4 所示，由于本方法在实际应用中使用 USB OTG 设备，它既可以作为 USB 主设备与 USB 从设备相连，也可以作为 USB 从设备与 PC 相连，因此需要在软件中对接入 USB 端口类型进行判断。作为直接访问方式工作时，该 USB 主设备端的任务是执行 USB 从设备端写在“请求”文件中的各项命令。

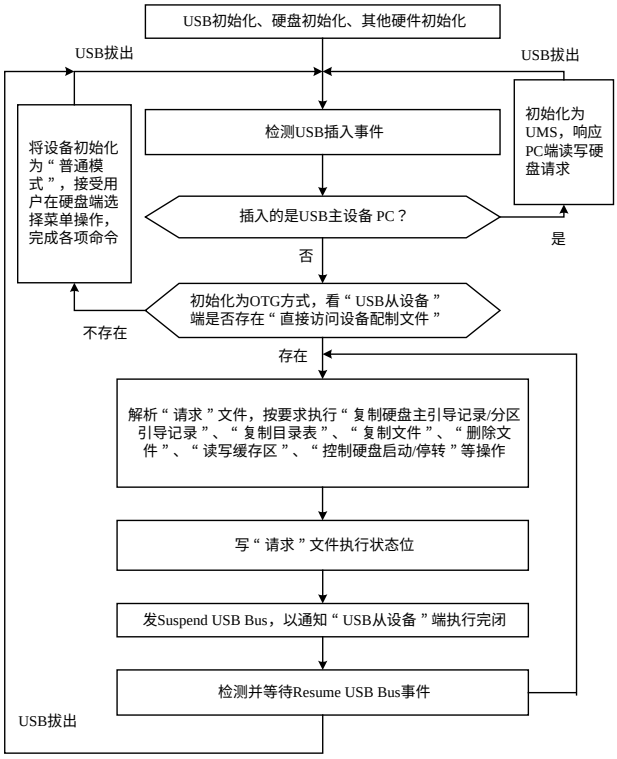

图 4 USB 主设备端软件流程

5 结束语

本文提出了通过在 USB 从设备端的存储器中建立交换缓存区来实现 USB 从设备对主设备直接访问的方法，对该方法的 USB 主从设备软硬件实现进行了描述，其关键是 USB 从设备端顺序读的“硬盘文件缓存系统”层的实现、缓存区大小的计算以及预读的提前量预估算法。该方法的优点在于：

(1)兼容性好，不用修改传统的硬件架构和底层驱动程序；

(2)手持播放器无须拔插 flash 存储卡，即可直接访问外扩硬盘中大于自身存储器剩余空间的文件；

(3)可实现手持播放器以低成本和省电的方式扩展外接硬盘。

参考文献

[1] Anderson D. USB 系统体系[M]. 精英科技, 译. 北京: 中国电力出版社, 2001.

[2] 上海怀瑾计算机科技有限公司. 直读专利介绍[Z]. (2006-10-12). http://www.witchain.com/sps_2.htm.

[3] Compaq 公司, Hewlett-Packard 公司, Intel 公司, 等. Universal Serial Bus Specification[Z]. 2 版. (2000-04-27). http://www.usb.org.