

基于 EZ-USB 的 CCD 信号采集系统

刘小军, 刘栋斌

(中国科学院长春光学精密机械与物理研究所, 长春 130033)

摘要: 针对航天应用领域某有效载荷设备的 CCD 信号采集需要, 提出一种基于 USB 2.0 的 CCD 信号采集方案, 其中包括硬件接口设计和软件设计。采用 FPGA 加 EZ-USB 的系统架构完成硬件接口设计, USB 接口芯片工作于从模式, 信号传输采用批量传输模式, 利用 VC++6.0 平台设计出信号采集软件。经 ISE 和 Modelsim 仿真以及 EZ-USB 硬件平台检验, 系统实现了 CCD 信号采集, 且运行稳定。

关键词: USB 采集; CCD 信号采集; 固件; 驱动程序

CCD Signal Acquisition System Based on EZ-USB

LIU Xiao-jun, LIU Dong-bin

(Changchun Institute of Optic, Fine Mechanic and Physics, Chinese Academy of Sciences, Changchun 130033)

【Abstract】 For the need of Charge Coupled Device(CCD) signal acquisition of some payload in spaceflight, a CCD signal acquisition scheme based on USB 2.0 is presented, including the hardware interface design and software design. The hardware architecture is implemented with FPGA and EZ-USB, the USB interface IC is in slave mode and the signal transmission mode is batch transmission. The signal acquisition application is designed on the platform of VC++6.0. The system realizes CCD signal acquisition and runs steady, which is proved with the emulator ISE and Modelsim, and the hardware platform based on EZ-USB

【Key words】 USB acquisition; Charge Coupled Device(CCD) signal acquisition; firmware; driver

电荷耦合器件(Charge Coupled Device, CCD)是摄像、摄影设备里一个极其重要的部件, 它起到将光信号转换成电信号的作用, 1 个 CCD 信号包括 3 个电平: 复位电平, 参考电平和信号电平。实际代表光信号强度的是参考电平和信号电平之差, 所以在将模拟信号转换成数字信号以前, 需要将这 2 个信号相减(相关双采样), 然后将差值进行 A/D 转换。一种普遍的方法是使用 CCD 集成信号处理器, 只需要外加必要的采样时钟和输入 CCD 信号, 便能完成上述功能。EXAR 公司的 98L63 便是这样一款芯片, 它除了能完成相关双采样和 12 位 ADC 输出外, 还能提供诸如暗电平自校准、可编程增益放大、孔径延迟等可编程特性, 是一款比较优秀的集成 CCD 信号处理器。

1 USB 2.0 简介

CCD 信号总是和图像或视频相关的, 图像和视频的数据量通常都比较大, 要进行 CCD 信号的传输, 就需要选择一种支持高速传输的接口, USB 正好满足这个要求, 除了支持高速传输, USB 还因为支持热插拔和易扩展而得到了广泛的应用。USB 设备给用户的感觉是“USB 非常方便, 非常易用”, 然而, USB 设备对于其开发者来说, 绝对算得上是一个挑战。总的来说, 1 个基于 USB 的系统主要分为 3 个部分: 主机, 设备和互连, 如图 1 所示。在任何 USB 系统中, 只有一个主机(任何通信都是由主机发起的)。USB 和主机系统的接口称为主控制器, 它可由硬件、固件和软件综合实现; 设备可以分为功能性设备和集线器 2 种, 前者作为系统的功能扩展设备, 后者作为 USB 设备的扩展连接点; 互连定义了主机和外设的连接和通信方式, 包括总线拓扑结构、内部分层关系、数据传输模型和总线访问控制等几个部分^[1]。

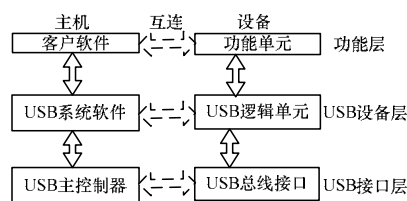


图 1 USB 系统层次结构

进行 USB 数据通信需要硬件 IC 的支持, 这类芯片主要分为 2 类: 一类是芯片本身没有集成微处理器, 必须外加微处理器才能完成数据通信; 另一类是内部集成有微处理器的 USB 接口芯片, 如 CYPRESS 公司的 EZ-USB 系列。EZ-USB FX2 系列接口芯片使用比较广泛, 满足 USB 2.0 协议, 内部集成有增强型 8051 微处理器、USB 智能引擎, 能够方便地开发出满足用户要求的 USB 设备。

2 系统方案

本系统的目的是将 CCD 集成信号处理器的输出(CCD 信号经过相关双采样和 A/D 转换)经由 USB 接口实时传给 PC, CCD 信号处理器的控制由 FPGA 完成, FPGA 还接收主机的命令, 然后翻译成控制 USB 接口芯片的信号。除此之外, 还利用 FPGA 自带的 RAM 生成一个 FIFO, 用作 USB 与主机通信的缓存。主机端应用程序以波形的形式显示采集到的数据和控制 USB 的通信过程。本文的重点是介绍 USB 接口方面的知识, 即 USB 与外设的接口设计和主机端固件、应用程

作者简介: 刘小军(1981-), 男, 硕士研究生, 主研方向: 数字信号处理; 刘栋斌, 副研究员、硕士

收稿日期: 2008-12-16 E-mail: liuxj5502@163.com

序以及驱动程序的编写。图 2 是系统方案的框图。

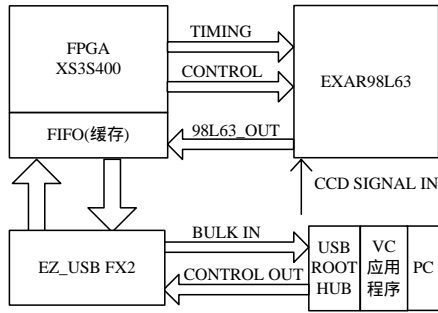


图 2 系统方案

3 硬件设计

USB 接口芯片通过 USB 电缆直接和主机连接，固件程序的下载、控制字节的输出和 USB 返回的状态信息和数据都由 USB 电缆传输。由于 EZ-USB FX2 内部只有 RAM，掉电或复位后程序丢失，为了避免每次上电手动下载程序，可以利用 USB 芯片本身已经固化好的程序下载逻辑。这个逻辑每次上电或复位后，会自动检测其 IIC 总线上是否有 EEPROM。如果有，它会进一步检测 EEPROM 的第 1 个字节，如果第 1 个字节是 C0，说明 EEPROM 中存有设备的 VID 和 PID，这时 USB 会根据 EEPROM 中的 PID 和 VID 加载相应的驱动程序，但固件程序仍需要手动下载(这个过程也叫 C0 加载)；如果 EEPROM 的第 1 个字节是 C2，说明这时挂在 IIC 总线的 EEPROM 存有 IIC 格式的固件程序，固件加载逻辑这时会自动从 EEPROM 下载程序到 USB 的内部 RAM 中然后运行(这个过程也叫 C2 加载)。

USB 和外设的连接有 2 种方式：SLAVE(从)方式和 GPIF(主)方式。当采用 SLAVE 方式的时候，读写 USB 的时钟由外部控制器(这里是 FPGA)提供，它们分别是 SLRD 和 SLWR。相应的，USB 反馈端点 FIFO 的状态(如空或满)给外部控制器，以便外部控制器决定读写时间；当采用 GPIF 模式的时候，读写时钟由 USB 给出，读写时序的编写可以借助 Gpif Designer 工具完成。PA[7...0]是一组复用 I/O 口，当 USB 工作在不同模式的时候代表不同的含义，如 PA4/FIFOADR0 和 PA5/FIFOADR1。当 USB 工作在 FIFO SLAVE 模式的时候，其值代表 USB 的端点 FIFO(哪个端点 FIFO)，不用的时候可以用作通用 I/O 口。最后，USB 和外部逻辑的数据传输是通过 16 位数据总线完成的，它们是 PD[15...0]/FD[15...0]，不需要数据传输的时候也可以用作通用 I/O 口。另外，128 脚封装的 USB 还可以外接程序存储器，方便硬件的调试和程序存储空间扩展。硬件的连接方式如图 3 所示。

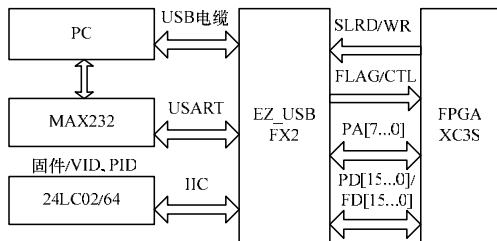


图 3 系统硬件互连示意图

4 软件设计

4.1 固件设计

固件框架完成了一个简单的任务循环，框架首先初始化

内部的状态变量，然后调用用户初始化函数 TD_Init()。从该函数返回后，框架初始化 USB 接口到未配置状态并使能中断。然后每隔 1 s 进行一次设备重枚举，直到端点接到一个 SETUP 包。一旦接到 SETUP 包，框架将开始交互的任务调度，具体过程如图 4 所示。

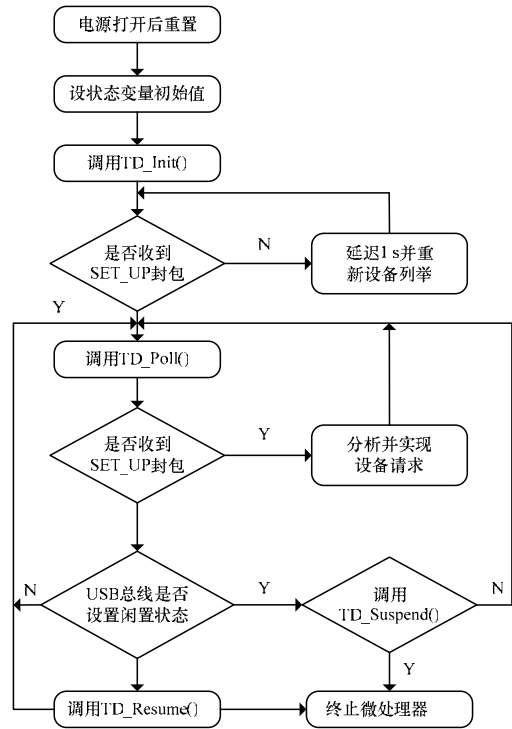


图 4 固件程序框架结构

固件是一种软件，是一种担任系统最基础、最底层工作的软件。实际上，固件决定了一个设备所具有的功能，改变固件，就是改变设备的特性，就可以改变设备的功能^[2]。一个固件的完善和稳定决定了相应设备的性能，所谓的固件升级，刷固件就是这个意思。

CYPRESS 公司为其 USB 芯片提供了一个基于 KEIL C51 环境开发的固件框架，当用户开发自己的 USB 设备的时候，只需实例化其中的 PERIPH.C 模块即可。这个框架主要由以下几部分组成：

- (1)FW.C，框架源代码，它包含了程序框架的 MAIN 函数，管理整个 51 内核的运行。
- (2)PERIPH.C，外围设备控制文件，它负责系统周边器件的互联。固件的设计主要针对这个文件，用户必须根据自己系统的需要，实例化这个文件，以实现所需的功能。在这个文件中有几个函数是比较关键的：

1)TD_Init(void)，该函数在框架初始化期间被调用。在设备重枚举和任务调度启用之前调用该函数，用来初始化用户的全局状态变量，并可规定各种端点资源的使用(包含中断)以及配置外围接口的输入/输出等。

2)TD_Poll(void)，在设备工作期间被重复调用，它包含一个执行外设功能的状态机。其负责系统中循环任务的处理，主要是对各个端点的状态进行查询，处理各种 OUT 或 IN 端点的交互。值得说明的一点是，这种处理只是辅助性质的，大部分工作由硬件自动完成。

3)DR_VendorCmd(void)，主要负责用户自定义命令的译码工作，用户请求通过端点 I/O 传输给内核。由于 EZ-USB

FX2 上 SIE 硬件的支持, 用户只需查询固定地址单元即可获得当前的命令代码。

(3)DSCR.A51 是描述表文件, 它负责 USB 设备的描述工作, 包括设备描述符、配置描述符、端点描述符和字符串描述符, 主要用于描述设备的各种信息。

(4)2 个包含文件 EZUSB.LIB 和 USBJMPTB.OBJ, 前者是 EZUSB 函数库的二进制文件, 后者是 USB 的中断向量表。

4.2 应用程序设计

利用 VC++6.0 编写 PC 端的应用程序, 实现把 USB 数据以波形的方式显示出来。由于波形可能超过客户区的大小, 需要能对波形进行调整, 比如周期的增大与减小、波形的左右或上下移动等功能。另外, 还需要通过应用程序实现对数据采集方式的控制。

鉴于 CYPRESS 已经提供了应用程序的框架, 要实现自己所需的特殊功能, 只需要在此基础上进行扩展完善即可。本应用程序主要增加了以下几个类^[3]:

(1)CSplitterFrame 主要用于将原来的窗口客户区进行分割, 一部分用于显示数据, 另一部分用于显示波形。源程序如下:

```

BOOL CSplitterFrame::OnCreateClient(LPCREATESTRUCT lpcs,
CCreateContext *pContext)
{
    m_wndSplitter.CreateStatic(this,1,2);
    //创建 2 个一行两列的静态 view
    m_wndSplitter.CreateView(0,0,pContext->m_pNewViewClass,CSi
ze(300,0),pContext);
    m_wndSplitter1.CreateStatic (&m_wndSplitter,2,1, WS_CHILD|
WS_VISIBLE|WS_HSCROLL |WS_VSCROLL, m_wndSplitter.
IdFromRowCol,1));
    //将第 2 个 view 分成两行一列 2 个 view
    m_wndSplitter1.CreateView (0,0,RUNTIME_CLASS (CControl),
CSize(0,80),pContext);
    //创建第 2 个 view, 默认高度 80 像素
    m_wndSplitter1.CreateView (1,0,RUNTIME_ CLASS (CWave
Show), CSize(0,0), pContext);
    //创建第 3 个 view
    SetActiveView((CView*)m_wndSplitter.GetPane(0,0));
    //设置当前 view
    return TRUE;}

```

(2)CWaveShow 用于显示波形和对其进行动态调整。

波形的显示在 OnDraw()函数中实现, 首先获得 USB 输入的缓冲区指针, 缓冲区中存储的数据代表 CCD 信号的幅值。要显示波形, 只需每隔一定的像素单元在客户区画一个点(点的纵坐标由 CCD 信号的幅值决定), 然后将这些点连起来即可。

```

for(i=curpos,j=0;i<curpos+600/m_WaveStep;j+=m_WaveStep)
{ // curpos 代表水平滚动条的当前位
//m_WaveStep 表示像素的宽度
// m_vcurpos 代表垂直滚动条的当前位置
if(m_WaveStep!= m_inclflag)
{//水平滚动条的位置反映当前待显数据的起始位置
pDC->MoveTo(j,-(unsigned char) pBlkBuf[i]+ m_vcurpos);
pDC->LineTo(j+m_WaveStep, -(unsigned char) pBlkBuf[i]+ m_
vcurpos);
pDC->MoveTo(j+m_WaveStep, -(unsigned char) pBlkBuf[i]+ m_
vcurpos);
pDC->LineTo(j+m_WaveStep, -(unsigned char) pBlkBuf[i+1]+ m_

```

```

vcurpos);}}

```

波形的调整利用成员变量 m_WaveStep 和 OnHScroll()以及 OnVScroll()函数实现。首先在 CWaveShow 类中增加函数 OnHScroll()和 OnVScroll(), 用于响应滚动条的左右或上下移动以及拖动等动作, 并将反映滚动条位置的变量和 OnDraw()函数联系起来即可。

```

void CWaveShow::OnVScroll(UINT nSBCode,UINT nPos,
CScrollBar* pScrollBar)
{SetScrollRange(SB_VERT,0,tprc.Height(),TRUE);
//设置滚动条的范围
pScrollBar->GetScrollRange(&m_vminpos,&m_vmaxpos);
m_vmaxpos = pScrollBar->GetScrollLimit();
//将滚动条的最大值赋给 m_vmaxpos
m_vcurpos = pScrollBar->GetScrollPos();
//获取滚动条的当前位置
switch (nSBCode)
{ case SB_LINEUP: //向左滚动
if (m_vcurpos > m_vminpos)
m_vcurpos--; break; //未到最左边, 改变当前值
case SB_LINEDOWN: // 向右滚动
if (m_vcurpos < m_vmaxpos)
m_vcurpos++;break;
case SB_THUMBTRACK: // 拖动滚动条到指定位置
m_vcurpos = nPos; // nPos 滚动条被拖到的位置
break;
pScrollBar->SetScrollPos(m_vcurpos);
... } }

```

4.3 EZ Loader 驱动程序设计

对于一个基于 EZ-USB 芯片的设备, 如果具有上电固件自动下载的功能, 那么它只需要少量的存储器(例如 EPROM, EEPROM 等)用来存储设备的 VID/PID 即可。VID/PID 与主机系统中制定的设备驱动程序相关联, EZ Loader 就是这样的一个设备驱动程序, 其功能是下载固件到外设的 EZ-USB 芯片中, 开发自己的 EZ Loader 非常容易, 只需要几个固定的步骤, 在此不作详细讨论^[4]。

5 结束语

经过 ISE 和 Modelsim 仿真以及基于 EZ-USB 硬件平台的检验, 本系统完成了通过 USB 总线采集 CCD 信号(由 FPGA 模拟产生)的功能。图 5 显示的是用 FPGA 产生的 CCD 模拟信号, 最后一行信号是将 FPGA 输出的 8 位 CCD 模拟信号直接叠加的效果。图 6 为 USB 采集到的 CCD 模拟信号及其波形。USB 读数的方式为批量传输方式, 系统上电时固件能自动加载, 也可以将 IIC 格式的固件下载到 EEPROM 中。

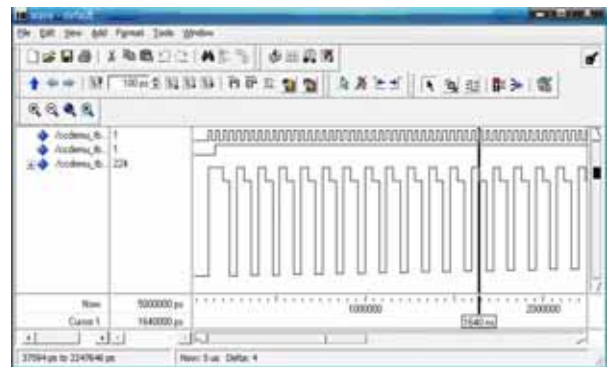


图 5 FPGA 产生的 CCD 模拟信号

(下转第 229 页)