

基于 USB2.0 的高效数字图象处理系统设计与实现

肖 辉¹, 杨维剑²

(1. 四川理工学院自动化与电子信息学院, 四川 自贡 643000 2 四川理工学院计算机学院, 四川 自贡 643000)

摘 要: 文章介绍了一种基于 Cypress 公司的 EZ-USB FX2 与 Omnivision 公司的 OV9620 芯片以及 MAX 公司的 PLDEPM7128 组成的高效视频信号采集、压缩、传输与控制的系统设计与实现, 以及相应的固件程序设计。

关键词: USB2.0 接口技术; 数字图像实时采集

中图分类号: TP393.4

文献标识码: A

引 言

图象采集器的传统方法是利用模拟 (Analogic) 摄像头采集图象输出视频信号 (Video) 再经 V/F 转换和单片机处理, 最后传送给 PC 机, 缺点是不方便、图象质量差和传输距离近等。

随着数字技术的发展, USB 总线的实时数据采集系统包括多路开关、A/D 转换器、单片机和 USB 接口芯片等。USB 接口芯片用于存储、采集数据并将其上传 PC, 同时也接收 PC 机 USB 控制器的控制信息。

PDUSBD12 集成了 SoftConnect GoodLink 可编程时钟输出、低频晶振、终端电阻、多配置 FIFO 存储器、收发器和电压调整器等。该系统完全支持 USB1.1 规范。可以用 FDUSBD12 和 MCS-51 单片机组成的系统。这种系统的优点是克服了模拟系统的传输数据处理不方便、抗干扰能力差等缺点, 但还存在由于频带窄 (低速模式不超过 1.5MB/S 全速模式不超过 12MB/S), 传输速率低 (不超过每秒 10 帧) 满足不了图象实时传输要求 (24 帧/秒以上) 等缺点。本文选用的 Cypress 公司的 EZ-USB FX2 是一款集成 USB2.0 的微处理器, 它集成了 USB2.0 收发器、SIE (串行接口引擎)、增强的 8051 微控制器和可编程的外围接口。FX2 这种独创性结构可使数据传输率达到 56Mbytes/s (即 USB2.0 允许的最大带宽), 实现高分辨、高清晰和高传输速率的图象采集器的设计^[1]。

1 图象采集原理

采集原理如图 1 所示。在图 1 中, OV9620 芯片作为图像采集芯片 (芯片设定为 0.5-30 帧/s 工作频率为 24MHz) 所采集的图像数据在 PLD 的控制下, 直接送到 CY7C68013 的 FIFO 缓冲区, 再由 CY7C68013 内部的 USB2.0 接口通过 USB 电缆将其图像数据传输到计算机 (PC)。带有 USB 接口的 EZ-USB 系列处理器 CY68013 有 FIFO (FX2) 和 GPIF 两种工作方式, 我们采用 FX2 方式, FX2 有 8K 片上 RAM, 地址为 0x0000-0x1FFF; 512 字节 SRAM, 地址为 0xE000-0xE1FF。通过固件将 CY7C68013 片内的 SRAM 作为外部 RAM 寻址。FX2 保留几 K (0xE200-0xFFFF) 数据地址空间作为控制 状态寄存器和端点缓冲器^[2-5]。

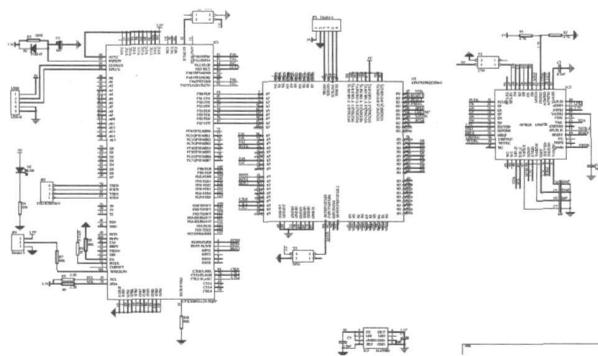


图 1 图象采集原理

2 固件设计

该系统软件包括固件程序、应用程序和驱动程序。其中,固件程序是指运行在设备 CPU 中的程序,是整个程序设计的核心,可采用汇编语言和 C 语言设计。只有在该程序运行时,外设才能称之为具有给定功能的外部设备。固件开发主要是通过通过对 68013 进行初始化设置、对 USB 请求进行处理来实现与 USB 兼容的外围设备所需的基本功能。固件程序的流程图如图 2 所示^[6]。

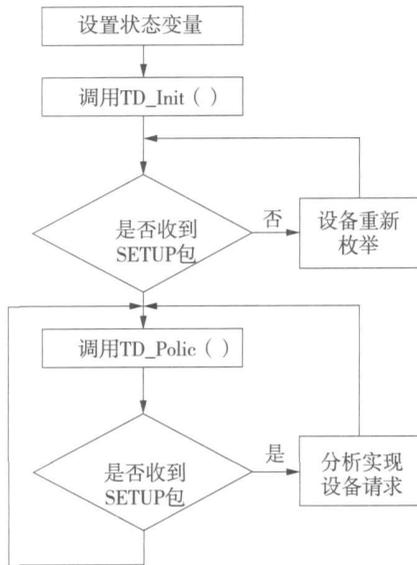


图 2 固件构架流程图

3 驱动程序设计

在 Windows 操作系统下,主机与采集设备之间的 USB 通信必须通过设备驱动程序来传输。设备驱动程序知道如何与系统的 USB 驱动程序及存取设备的应用程序沟通。系统的上位机应用程序的工作流程是,首先初始化,获得设备的句柄,然后对缓冲区进行读取数据,读取完数据后,再填入位图缓冲区,但是在填入的时候,必须先要提取行号,以便程序能够正确地对缓冲区进行填写。要特别指出地是,OV 图像芯片输出的是 R-G-R-G,或者 G-B-G-B 的数据,所以还要进行灰度计算^[14-5]。

设备初始化:灰度计算。

```
void CDisplayView : DrawGray2D ( CDC * pDC,
HANDLE hDB, int con)
```

```
{
    CPalette pal;
    CPalette* OHPal;
    BIIMAPINFO &bmInfo = * ( LPBIIMAPINFO )
```

```
int nColors= bmInfo.lmHeader.bClrUsed;
lmHeader.bClrUsed 1 << bmInfo.bBits-
Count
if( pDC -> GetDeviceCaps( RASTERCAPS)& RC_
PALETTE && nColors<= 256 )
{
    UNT nSize= sizeof( LOGPALETTE) + ( sizeof
(PALETTEENTRY)* nColors);
    LOGPALETTE * pLP = ( LOGPALETTE* ) new
BYTE[ nSize];
    pLP -> paVersion= 0x300;
    pLP -> paNumEntries= nColors;
    for( int i= 0; i< nColors; i++ )
    {
        int nGray= con+ i; pLP -> paPalEntry[ i]. peRed
= nGray; pLP -> paPalEntry[ i]. peGreen = nGray;
pLP -> paPalEntry[ i]. peBlue = nGray;
pLP -> paPalEntry[ i]. peFlags= 0;
    }
    pal.CreatePalette( pLP);
    delete[] pLP;
    OHPal= pDC -> SelectPalette(&pal, FALSE);
    pDC -> RealizePalette();
}
else if( ( pDC -> GetDeviceCaps( RASTERCAPS)
& RC_PALETTE) == 0 && nColors<= 256 )
{
    for( int i= 0; i< nColors; i++ )
    {
        bmInfo.lmColors[ i]. rgbRed= nGray;
        bmInfo.lmColors[ i]. rgbGreen = nGray;
        bmInfo.lmColors[ i]. rgbBlue = nGray;
    }
}
}
```

图形显示

```
isoControl.PackSize = 1024* 3;
isoControl.PackCount = 512;
isoControl.PpeNum = 0;
isoControl.BufferCount = 2;
isoControl.FramesPerBuffer = 16;
ULONG nBytes = 0;
PUCHAR buffer = NULL;
ULONG bytesToRead;
bytesToRead = isoControl.PackCount*
( isoControl.PackSize + sizeof(USBD_ISO_PACKET
```

```

_DESCRIPTOR);
buffer= (UCHAR) malloc( bytesToRead);
if (! buffer)
{
    MessageBox(NULL, “内存溢出!”, “信息提示”,
    MB_OK);
    return 3;
}
bResult= DeviceIoControl( pdlg-> GetDocument
() -> phand
    IOCTL_EZUSB_READ_ISO_BUFFER,
    & IsoControl
    sizeof( ISO_TRANSFER_CONTROL),
    buffer,
    bytesToRead
    & nBytes,
    NULL);
if (bResult!= TRUE)
{
    MessageBox( NULL, “读数据错误!”, “信息提
示”, MB_OK);
    free( buffer);
    return 3;
}
ptr= buffer;
int i;
int j;
int k;
int flag= 1;
for( i= 0; i< 480; i++)
{
    for( j= 0; j< 640; j++)
    {
        k= ptr[ * 1024+ 1021]* 256+ ptr[ * 1024+
1020];
    }
}

```

```

k%= 480;
pDb[ k* 640+ j]= ptr[ * 1024+ j];
} //pdlg-> GetDocument() -> mycount[ i]= ptr[ i
* 1024+ 641]* 256+ ptr[ * 1024+ 640]; } free( buff
er);
pdlg-> SetRedraw( true);
pdlg-> Invalidate( false);
}

```

利用批量传输时通过打包传输, 每个包的大小为 512字节, 将读取的包写到数据缓冲区中, 所以每次都要调用 DeviceIoControl函数。由于调用 DeviceIoControl函数的时间为 20ms左右, 必须减少调用次数才能提高传输的速度。因此每次 DeviceIoControl读取的数据量的大小成为批量传输的关键。

在驱动里通过 MaximumTransferSize定义每次 DeviceIoControl向缓冲区写入的数据的大小, 相对于 USB2. 0协议本身每个包的大小为 512字节应该是足够了, 经验证明, 在 20MB 采样时钟时传输数据给主机完成实时显示是没有问题的。由于驱动程序开发并非本文重点, 因此对驱动开发不再阐述。

参考文献:

- [1] 雷丰中. USB2. 0视频图像采集系统设计 [J]. 湖南工程学院学报: 自然科学版, 2005 6(2): 1-3
- [2] 许永和. EZ-USB FX 系列单片机 USB 外围设备设计与应用 [M]. 北京: 北京航空航天大学出版社, 2003
- [3] 吴勇. 线阵 CCD 数据采集与 USB2. 0 传输 [J]. 光学仪器, 2005 2(1): 17-20
- [4] 杨亚莉. 基于 CCD 的图像采集系统设计 [J]. 武汉科技学院学报, 2007, 9(9): 42-44
- [5] 李明伟. 一种高速线阵 CCD 图像数据采集系统 [J]. 仪器仪表学报, 2005 8(8): 716-718
- [6] 李刚. 基于 USB2. 0 的高速线阵 CCD 数据采集系统 [J]. 长春理工大学学报: 自然科学版, 2008 3(1): 15-18

Design and Realization of High-resolution Digital Image Collecting System Based on USB 2. 0

XIAO Hui¹, YANG Weirjian²

(1. School of Automation and Electronic Information, Shuan University of Science & Engineering Zigong 643000, China

2. School of Computer Science, Sichuan University of Science & Engineering Zigong 643000, China)

Abstract A high-activity acquisition, compression, transmission and control system of vision signals is designed and realized in the paper, which is based on the combination of EZ-USB FX2 of Cypress Company, OV9620 chips of Omnivision Company and PLDE1M 7128 of MAX Company. The associating firmware programs are also discussed.

Key words USB2. 0 interface technology, real-time digital image acquisition