文章编号：1672-2892(2008)01-0046-05

基于 EZ-USB FX2 的单向传输系统设计与实现

李 波[1a]，刘嘉勇[1a]，蒋 瑜[1a]，刘现魁[1b]

(1.四川大学 a.信息安全研究所；b.电子信息学院，四川 成都 610064)

摘 要：为了满足不同安全等级主机间单向数据传输的应用要求，介绍了一种基于 EZ-USB FX2 实现的单向传输系统设计方案，采用通用串行总线 USB2.0 芯片 Cy7c68013 和先入先出存储器 IDT72v06，通过 Cy7c68013 通用可编程接口与外部先入先出的连接，实现了在不同安全域主机之间单向、高速、安全可靠的数据传输。

关键词：单向传输；通用串行总线；Cy7c68013 芯片；通用可编程接口；先入先出

中图分类号：TP309.2 文献标识码：A

Design and Implementation of Data Uni-directional Transmission System Based on EZ-USB FX2

LI Bo[1a]，LIU Jia-yong[1a]，JIANG Yu[1a]，LIU Xian-kui[1b]

(1a.Information Security Institute；1b.School of Electronics and Information Engineering，Sichuan University，Chengdu Sichuan 610064，China)

Abstract: To satisfy the application requirements for host-computers uni-directional data transmission in different security ratings, a design scheme of uni-directional transmission system based on EZ-USB FX2 is presented in this paper. And the system which adopts the Universal Serial Bus(USB)2.0 chip Cy7c68013 and IDT72v06 First in First out(FIFO) memory, realizes uni-directional, high-speed, safe and reliable data transmission between the host-computers in different security domains, with hardware design of external FIFO through General Programmable Interface(GPIF) and application program.

Key words: uni-directional transmission；USB2.0；Cy7c68013；GPIF；FIFO

根据有关部门对计算机单向信息传输的具体要求，不同安全域间数据传输只能由低密级网络主机向高密级网络主机传输，在物理和逻辑上不能存在逆向数据通道。目前，为了解决敏感信息的安全问题，一些特定的网络或主机采用了物理隔离技术，如硬件卡、数据转播系统、空气开关和安全隔离网闸(Gap of All Protocol，GAP)等，但这些技术或多或少存在一定的问题，比如资源的利用率低，存在一定的安全隐患，访问速度慢，成本过高，数据传输在逻辑上仍然存在双向交换等。通用串行总线(USB)作为新一代计算机接口，具有快速、低成本、热插拔和即插即用等特点，广泛用于在主机与各种外设之间进行数据传输。Cypress半导体公司推出的USB2.0接口芯片EZ-USB FX2(Cy7c68013)，将高性能USB引擎和增强8051内核有机结合，并集成通用可编程接口(GPIF)来代替外部接口电路，使得开发过程简单迅速。本文介绍Cy7c68013芯片内GPIF与外部FIFO连接设计以及相应软件的编写，较好地解决了不同安全域主机间单向数据传输的应用要求。

1 硬件设计及原理

1.1 Cy7c68013 芯片简介

Cy7c68013(EZ-USB FX2)是 Cypress 公司生产的新一代高速 USB 系列，它集成了 USB2.0 收发器、串行接口引擎(Serial Interface Engine，SIE)、增强的 8051 微控制器、8.5 kB 的 RAM、4 kB 的 FIFO 存储器以及 GPIF，其内部结构见图 1[1]。

FX2 采用 3 种接口模式：端口、GPIF 和 Slave FIFO 模式[2]。在端口模式中，所有的 I/O 引脚都是 I/O 通用

收稿日期：2007-08-06；修回日期：2007-09-27

端口；在 GPIF 模式下，I/O 端口 B (PORTB)和 I/O 端口 D(PORTD)作为 16 位接口与 FX2 FIFO 端点 EP2,EP4, EP6 和 EP8 相连，而 FX2 FIFO 是被内部 GPIF 控制的；在 Slave FIFO 模式下，外部逻辑电路或外部处理器直接与 FX2 的端点 FIFO 连接，直接对 FX2 FIFO 进行控制，因此在此模式下 GPIF 模式是不起作用的。

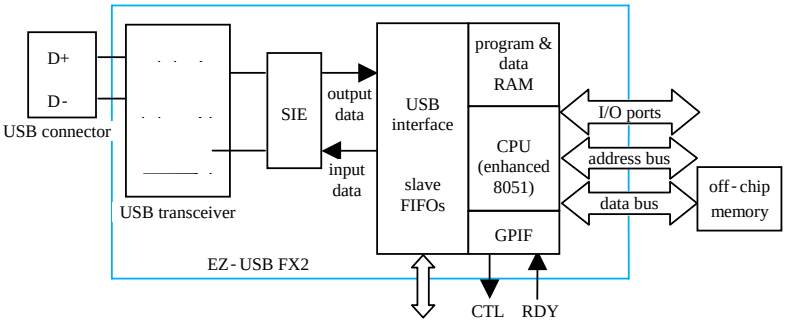


Fig.1 EZ-USB FX2 package simplified block diagram
图 1 EZ-USB FX2 的内部结构框图

本系统采用GPIF模式，GPIF实际是一个对于FX2端点FIFO的内部主控制器，即由芯片内部的硬件电路控制数据传输，可以完全独立于内部的8051核，它使FX2与外界设备无需建立接口电路就可实现“ 无胶合 ”连接。GPIF核心是一个可编程状态机(programmable state machine)，可产生6个“ 控制 ”输出互补晶体管逻辑(Complementary Transistor Logic，CTL)、9个“ 地址 ”输出GPIFADR以及6个外部和2个内部准备输入信号(RDY)。4个用户定义的波形描述符控制这个状态机，1个用作FIFO读，1个用作FIFO写，1个用作单字节/字读，1个用作单字节/字写。在GPIF模式下，FX2这种独创性结构可使数据传输率达到USB2.0允许的最大带宽56 MB/s[3]。

1.2 IDT72v06 FIFO 存储器简介

IDT公司的IDT72v06芯片是一种异步的FIFO，该芯片存储容量为16 kB，读写周期为25 ns，工作电压为3.3 V。IDT72v06有3个状态标志位：FIFO空(EF)、FIFO满(FF)和FIFO半满(HF)，另外提供2个控制信号：读数据(RD)和写数据(WR)，输入/输出信号总线D0~D8/Q0~Q8，扩展端扩展输入(XI)、扩展输出(XO)和首片选(FT)用来进行字深和字长的扩展，以便于多个芯片的组合使用，RS为复位端，若需将内部数据重新读出还可用控制端重传(RT)来实现。

图2为IDT72v06内部结构框图[4]。用户可根据这些标志位来控制写入和读出操作，由于存在内部读写指针，因此该芯片能够顺序读写数据，不需要一般RAM所需的地址线进行地址操作，就能通过芯片在一端写操作，而在另一端读操作，因而能够达到很高的传输速度。

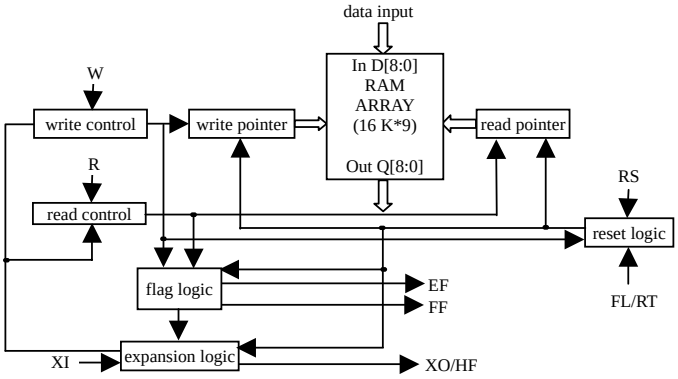


Fig.2 IDT72v06 FIFO package block diagram
图 2 IDT72v06 结构框图

1.3 硬件连接及其原理

本系统采用GPIF模式，通过控制外部FIFO IDT72v06与FX2内部端点FIFO之间的数据传输，从而实现不同安全级别主机之间信息的单向传输，系统硬件连接见图3，主要由USB2.0控制器Cy7c68013、FIFO存储器IDT72v06和PC主机组成。

发送和接收端 EZ-USB FX2 的 FD0~FD7 作为 8 位数据线分别与外部 FIFO D[0:7]和 Q[0:7]相连，用于数据传输；控制线 CTL 和 RDY 分别为 GPIF 的输入和输出控制信号，用于读写选择标志及外部 FIFO 空满状态[5]。

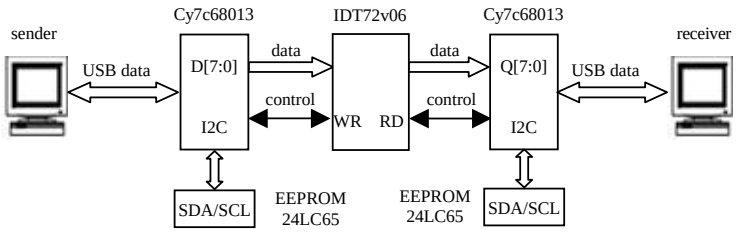


Fig.3 FX2 interconnect to external asynchronous FIFO(IDT72v06)
图 3 系统硬件连接原理图

工作过程如下：发送端PC机通过USB接口将数据发送到Cy7c68013的端点FIFO中，当内部端点缓冲区达到指定大小容量后，使用GPIF模式自动将数据打包传送给外部FIFO IDT72v06。当写满外部FIFO时，通知接收端Cy7c68013将数据从外部FIFO中取出并发送到接收端PC机。数据的单向传输是通过IDT72v06的数据存取来控制的，外部FIFO器件IDT72v06是先进先出的存储器，数据写端口只能接收而不能发送数据，数据读端口只能发送却不能接收数据，发送端的Cy7c68013数据接口连接IDT72v06的写控制，接收端的Cy7c68013数据接口连接IDT72v06的读控制，使发送端Cy7c68013只能向IDT72v06写入数据，而接收端Cy7c68013只能读出数据，因此通过这样的读写控制就能实现数据的单向传输。同时GPIF控制其总线在进行数据传输时能够不通过8051而直接与外部FIFO进行数据交换，以及在编写固件程序中使用外部FIFO的FF,EF和HF标志位的变化情况，使其数据能够自动、高速、单向连续不断地从发送端主机到接收端主机。

2 系统软件设计

2.1 FX2 固件程序设计

所谓固件程序就是固化在设备USB控制器内部程序存储器中或外部扩展的程序存储器中的程序。在使用EZ-USB FX2芯片进行应用开发中，利用EZ-USB FX2固件框架可以简化和加速开发基于该芯片的外围设备。固件程序的编译调试通过Keil Software公司的Keil uVision2开发环境，它符合ANSI C标准，且生成的程序代码执行效率很高，所需的内存空间极小，几乎可以和汇编语言媲美。

固件程序流程见图4，主要分成Cy7c68013初始化配置和读写外部FIFO两部分，包括初始化内部状态变量并设置寄存器，处理标准USB设备请求以及挂起时的电源管理，提供USB描述符表及用户编写的外设功能代码。Cy7c68013采用GPIF模式，将端点2配置成AutoOut模式，4缓冲，每缓冲512 byte；端点6配置成AutoIn模式，4缓冲，每缓冲512 byte。

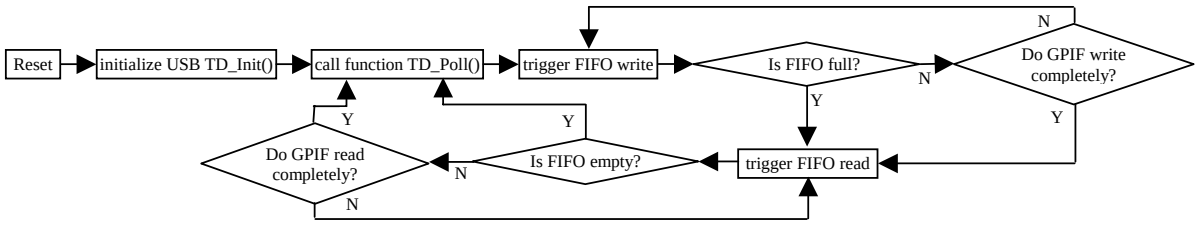


Fig.4 Firmware flow diagram
图 4 固件程序流程图

GPIF 控制器核心是一个可编程的状态机，必须设计波形描述符(Waveform Descriptor)来控制，即 GPIF 编程。采用 Cypress 公司提供的 GPIF TOOL，可以方便地生成波形描述符的源代码，直接添加到工程中即可使用。系统中的 2 片 Cy7c68013 执行的操作相近，具有对称性，发送端 Cy7c68013 的 GPIF 调用的是 FIFOWR(FIFO 写)波形，使用端点 EP2，接收端 Cy7c68013 的 GPIF 调用的是 FIFORD(FIFO 读)波形，使用端点 EP6，二者使用的端点和调用波形不同，但其流程相似[6]。图 5 为 FIFOWR 的波形描述符，由于篇幅所限，FIFORD 略过。

```
// GPIF Waveform 3: FIFOWR
//
// Interval     0         1         2         3         4         5         6      Idle (7)
//              ---------- --------- --------- --------- --------- --------- ------ --------
//
// AddrMode Same Val  Same Val  Same Val  Same Val  Same Val  Same Val  Same Val
// DataMode NO Data   Activate  NO Data   NO Data   NO Data   NO Data   NO Data
// NextData SameData  SameData  NextData  SameData  SameData  SameData  SameData
// Int Trig No Int    No Int    No Int    No Int    No Int    No Int    No Int
// IF/Wait  Wait 1    Wait 1    IF        Wait 1    Wait 1    Wait 1    Wait 1
//   Term A                     TCXpire
//   LFunc                      AND
//   Term B                     TCXpire
// Branch1                      Then 0
// Branch0                      ElseIdle
// Re-Exec                      No
// Sngl/CRC Default   Default   Default   Default   Default   Default   Default
// WEN#          0         0         0         0         0         0         0          0
// REN#          0         0         0         0         0         0         0          0
// OE#           0         0         0         0         0         0         0          0
// REN#          1         0         1         1         1         1         1          1
// unused        0         0         0         0         0         0         0          0
// unused        0         0         0         0         0         0         0          0
```

Fig.5 FIFOWR waveform descriptor
图 5 FIFOWR 波形描述符

固件程序以FIFOWR为例，初始化及读FIFO关键代码如下：

```
void TD_Init(void)
EP2CFG = 0xA0；          SYNCDELAY；                    // EP2OUT, bulk, size 512, 4x buffer
EP6CFG = 0xE0；          SYNCDELAY；                    // EP6IN, bulk, size 512, 4x buffer
FIFORESET = 0x02；       SYNCDELAY；                    // reset EP2 FIFO
FIFORESET = 0x06；       SYNCDELAY；                    // reset EP6 FIFO
EP2FIFOCFG = 0x10；      SYNCDELAY；                    // auto out mode
EP6FIFOCFG = 0x08；      SYNCDELAY；                    // auto in mode
GpifInit ()；              SYNCDELAY；                  // initialize GPIF registers
EP2GPIFFLGSEL = 0x01；   SYNCDELAY；                    // For EP2OUT, GPIF uses EF flag
EP6GPIFFLGSEL = 0x02；     SYNCDELAY；                  // For EP6IN, GPIF uses FF flag
void TD_Poll(void)
{if ( GPIFTRIG & 0x80 )                                 // if GPIF interface IDLE
    { if ( DATA_READY )                                 // if external data is not ready
      {if ( !( EP68FIFOFLGS & 0x01 ) )                  // if EP6 FIFO is not full
        {GPIFTCB1 = 0x01；  SYNCDELAY；                 // high speed mode
          GPIFTCB0 = 0x00；  SYNCDELAY；
          Setup_FLOWSTATE_Read()；SYNCDELAY；           // setup flowstate registers
          GPIFTRIG = GPIFTRIGRD | GPIF_EP6；            // launch GPIF FIFO READ Transaction
          SYNCDELAY；
          while( !( GPIFTRIG & 0x80 ) )                 // poll GPIFTRIG.7 GPIF Done bit
          {；
           }SYNCDELAY；
    }}}}
```

2.2 USB 设备驱动程序设计

FX2的驱动程序包括2部分：固件下载USB设备驱动程序EZ-loader Driver和EZ-USB通用设备驱动程序(General Purpose Driver，GPD)。固件下载驱动程序主要用于最终实现的产品能在上电时自动完成下载固件到外部存储器(例如EPROM和EEPROM)以及设备重枚举。开发EZ-loader驱动程序需要Windows DDK,Visual C++编译器和Cypress公司提供的Hex2c.exe工具，由于EZ-USB软件开发包提供的EZ-loader驱动程序只需经过少量修改就可以支持一个专门的设备，因此本系统就可以利用该模板快速地开发出设备驱动程序[7]。

通用设备驱动程序是一个可用于基于EZ-USB的计算机外围设备接口的通用设备驱动程序，主要提供应用程序与USB设备请求和数据传输的接口。通用驱动程序一般不需要开发者编写，可以使用Cypress公司提供的已经编译形成的驱动ezusb.sys。如果需要实现系统特有的通信功能，就必须在此基础上加以修改，编写USB通用设备驱动程序需要Microsoft WDM DDK和Visual C++。

2.3 应用程序设计

主机应用程序是操作系统与用户的接口，WIN32系统把每一个设备都抽象为文件，通过调用对文件操作的应用编程接口(API)函数，就可以实现与设备的通信。利用API函数，通过设备的驱动，向指定的设备发送正确的控制码和数据，然后分析它的响应，就可以达到数据传输的目的。这里主要介绍一个重要的接口函数DeviceIo Control[8]，通过调用它来实现对设备的访问，包括获取信息、发送命令和交换数据等，用ReadFile和WriteFile来进行批量传输。

```
Bool DeviceIocontrol(
    handle hdevice,                 // 设备句柄
    dword dwiocontrolcode,          // 控制码
    lpvoid lpinbuffer,              // 输入数据缓冲区指针
    dword ninbuffersize,            // 输入数据缓冲区长度
    lpvoid lpoutbuffer,             // 输出数据缓冲区指针
    dword noutbuffersize,           // 输出数据缓冲区长度
    lpdword lpbytesreturned         // 输出数据实际长度单元长度
    lpoverlapped lpoverlapped       // 重叠操作结构指针
);
```

本设计采用的开发工具有Visual C++和Windows 2000 DDK。应用程序包括发送端模块、接收端模块和发件箱模块3部分，图6为应用程序中信息传输发送端传输文件的界面。

Fig.6 Information transmission interface
图 6 信息传输发送端传输文件界面

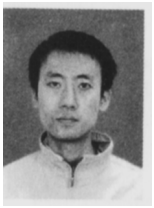
3 结论

通过USB2.0芯片Cy7c68013和IDT72v06 FIFO存储器的硬件连接设计，以及系统软件设计，实现了在不同安全域主机之间单向的数据传输。芯片在GPIF模式下，读写操作及打包传输不需要经过CPU干预，同时外部FIFO存储器读写数据时不用进行地址操作，因此可以达到较高的传输速度，经过实际测试，最高速率可达到4 MB/s，完全满足在某些特定环境下安全、快速、可靠的数据传输要求。

参考文献：

[1] Cypress Semiconductor Corporation. EZ-USB FX2 Technical Reference Manual[Z]. 2001.
[2] 胡晓军,张爱成. USB 接口开发技术[M]. 西安:西安电子科技大学出版社, 2005.
[3] 钱峰. EZ-USB FX2 单片机原理、编程及应用[M]. 北京:北京航空航天大学出版社, 2006.
[4] Integrated Device Technology,Inc. IDT72V06 3.3 VOLT Cmos Asynchronous FIFO Datasheet[S]. DSC-3033/3, 2003.
[5] 李亭,李华. GPIF 与 FIFO 接口设计[J]. 电测与仪表, 2006,43(6):56-58.
[6] 周云锋,单甘霖,王鑫. FX2 的波形描述符设计及应用[J]. 微计算机信息, 2005,21(2):158-159.
[7] 边海龙,贾少华. USB 2.0 设备的设计与开发[M]. 北京:人民邮电出版社, 2004.
[8] Cypress. EZ-USB General Purpose Driver Specification[EB/OL]. [1999-02-23]. http://www.cypress.com.

作者简介：



李 波(1982-)，男，山东省泰安市人，在读硕士研究生，研究方向为信息处理与信息安全. E-mail: lib_scu@163.com.

刘嘉勇(1962-)，男，成都市人，教授，研究方向为信息安全理论与应用、网络信息处理与信息安全.

蒋 瑜(1983-)，男，四川省眉山市人，在读硕士研究生，研究方向为网络与信息系统安全.

刘现魁(1982-)，男，河北省邯郸市人，在读硕士研究生，研究方向为激光与光通信.