裸机网口通信

驱动组 王彬 2016.06.03

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 时间 | 作者 | 说明 |
| V1.0 | 2016.6.3 | 王彬 | 第一版本draft |
|  |  |  |  |

# 1:WinPcap

## 1.1：简介

WinPcap是windows平台下的开源的网络抓包和分析工具。有时候有些应用需要直接获取网络上的raw数据包，而不经过操作系统的协议的干预。Winpcap能够很好的获取这些raw packets。它是通过安装设备驱动的方式来将动态库加载到win32内核。

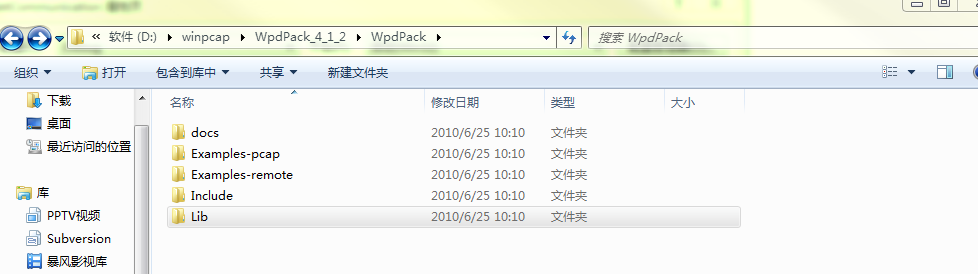
## 1.2：安装

Winpcap在安装wireshark的时候自动提示是否安装，因此可以单独安装winpcap或者直接安装wireshark。现在使用的版本是winpcap4.1.2版本，网址<http://www.winpcap.org>。

wireshark使用的是1.6.8的版本

## 1.3：使用

安装好winpcap后使用方法如下，在网站上下载其源码包，解压后会有如下目录



1：将include文件夹拷贝到工程目录下，然后在属性中包含该include目录和其中的子目录（可采用相对路径的方式，方便拷贝给他人使用）

2：将lib目录下对应x86还是x64的系统的lib库加到工程中使用。

3：在工程中需要使用其函数时包含对应的头文件即可.

4：winpcap库必须在管理员的模式下才能够使用，否则会导致读取不到设备列表

# 2：NetCommunication库

## 2.1：Packet说明

根据《GTSD41调试接口通信协议20160603》中的定义，定义数据包的结构如下所示

#pragma pack(1)

typedef struct packetformat

{

//header format start

Uint8 dmac[MAC\_ADDR\_BYTE\_LENTH]; // mac address

Uint8 smac[MAC\_ADDR\_BYTE\_LENTH]; // source mac address

Uint8 type0; // used for identify device

Uint8 type1; // used for identify device

Uint8 lengthbit7\_0; // length[7:0]

Uint8 lengthbit9\_8; // {6’b000100, length[9:8]}

Uint8 index; // index number

Uint8 dma\_numbit7\_0; // dma\_num[7:0]

Uint8 dma\_numbit8; // {cmd,6’b0,dma\_num[8]}

Uint8 dmaAddrbit7\_0; // dma addr[7:0]

Uint8 dmaAddrbit15\_8; // dma addr[15:8]

//header format end

Uint8 dataSection[MAX\_PACKET\_LEN - PACKET\_HEADER\_LENTH]; // user data section 最大包长减去包头长度

////first six data byte used for transfer the cmd

//Uint8 data0\_CmdIdbit7\_0; // cmd id bit7:0

//Uint8 data1\_CmdIdbit11\_8\_axisnum; // cmd id bit11:8 15:12 --axis number

//Uint8 data2\_CmdType; // cmd type 0:read 1:write

//Uint8 data3; // rsv

//Uint8 data4\_ret; // return type

//Uint8 data5; // rsv

}PACKET\_FORMAT;

#pragma pack()

其中数据的前6个字节定义为固定用途，如上面注释部分，实际数据是从数据的第七个开始放置。

Packet文件中主要实现对数据包的一些操作，下面是一些需要注意的地方

1：目标mac都是ff说明是广播地址

2：源mac地址初始化时可以随意写，后面会获取本地的mac地址，更新发送的mac地址进行通信

3：type0和type1需要按照文档说明上面进行设置

4：操作fpga的时候是不需要考虑前面6个固定意义的数据的，这6个数据只有在和dsp进行通信的时候才有意义。和fpga通信时不需要考虑。

### 2.1.1：函数说明

|  |  |  |
| --- | --- | --- |
| 序号 | 函数名 | 函数说明 |
| 1 | decoder\_returnflag(int16 mode) | 解析指令的返回值，通过返回值判断是否指令正确返回 |
| 2 | decoder\_dsp\_packet(Uint8 index, int16\* pData, int16 &dma\_num) | 解析dsp返回的数据，实际数据是从第七个开始的。也就是前面固定的6个数据不放到最终的数据buffer中 |
| 3 | decoder\_fpga\_packet(Uint8 index, int16\* pData, int16 &dma\_num) | 不需要考虑前面6个固定意义的数据 |
| 4 | get\_local\_mac(const int8\* adapterName) | 获取本地mac地址 |
| 5 | fillPacket(int16 dma\_addr, int16\* pData, int16 dma\_num, int16 FPGAmode) | 填数据包 |

## 2.2：Net说明

该部分主要用来操作winpcap，像打开设备，关闭设备，发送接收等功能。

### 2.2.1：函数说明

|  |  |  |
| --- | --- | --- |
| 序号 | 函数名 | 函数说明 |
| 1 | init\_net(const int8\* adapter, int32 port) | 初始化网口，并在其中设置监听该端口的数据filter |
| 2 | tx\_packet() | 发送包 |
| 3 | wait\_response() | 接收数据，读取1000次如果还没有读到就认为读取失败 |
| 4 |  |  |
| 5 |  |  |

## 2.3：库函数接口说明

|  |  |  |
| --- | --- | --- |
| 序号 | 函数名 | 函数说明 |
| 1 | int16 NetCom\_Open(); | 打开通信 |
| 2 | int16 NetCom\_Close(); | 关闭通信 |
| 3 | int16 NetCom\_FPGA\_ComHandler(int16 mode, int16 dma\_addr, int16\* pData, int16 dma\_num); | FPGA通信处理函数  输入：  mode: 1:写 0：读  dma\_addr: dma地址  pData: 输入数据  dma\_num: 输入数据长度  输出：pData: 输出数据  dma\_num: 输出数据长度  返回：0成功，其他参看错误列表。 |
| 4 | int16 NetCom\_DSP\_ComHandler(int16 mode, int16 dma\_addr, int16\* pData, int16 dma\_num); | Dsp通信处理函数  输入：  mode: 1:写 0：读  dma\_addr: dma地址  pData: 输入数据  dma\_num: 输入数据长度  输出：pData: 输出数据  dma\_num: 输出数据长度  返回：0成功，其他参看错误列表。 |
| 5 |  |  |

打开网口的时候需要遍历电脑上所有的网口，向网口发送数据（FPGA的双口ram首地址）连续发送十次数据，如果有5次以上返回说明是该网口。

还有些辅助的找到网口的信息的一些函数。这些函数属于内部使用的。