

国咨 USB 电话模块开发指南

一、概述：

本产品是西安国咨软件有限公司于 2015 年推出的一个固定电话模块，通过 USB 接口和上位机通讯，上位机可以通过 USB 接口控制电话的摘机、挂机、拨号，也可以往电话线路里播放语音；上位机也可以通过该 USB 接口获取电话线路状态，例如电话的摘挂机状态、振铃信号、来电号码、DTMF 按键、忙音等。

由于本产品使用 USB 接口，因此具有很大的灵活性，上位机除了支持 WINDOWS 之外，也支持 Android 和 Linux。

本产品按照 GB/T 15279 国家标准要求设计生产，电话模块部分的功能在各行各业取得了长足的应用，具有很高的可靠性和灵活性。

二、与模块通讯：

本模块插入上位机的标准 USB 插孔后，会自动被识别为一个 U 盘，名字为 GTRecorder，盘符下有二个文件：TMP.DAT 和 TMP.STA。上位机通过对这二个文件的读写，实现与电话线路的交互。

1、读文件 TMP.DAT：可以得到电话线路实时音频数据用于电话录音，也可以获得来电号码（FSK），DTMF 按键（或 DTMF 来电号码）、忙音、振铃音。

2、读文件 TMP.STA：可以得到电话线路实时电压，用于确定电话机的摘机挂机状态。

3、写文件 TMP.DAT：主要用于播放语音到电话线路，实现留言电话的功能。也可以播放合成的 DTMF 语音，实现拨号。

4、写文件 TMP.STA：实现模拟摘机挂机，命令模块拨号，也可以设置录音的音量。

三、软件编程：

1、操作系统对文件的操作一般都有缓存，而与电话模块的通讯是实时的，不允许有缓存，因此，访问电话模块的文件要用以下几个函数：

WINDOWS: CreateFile(), SetFilePointer(), ReadFile(), WriteFile(), CloseHandle()

LINUX: open(), lseek(), read(), write(), close()

ANDROID: 待定

2、编程细节：

1)、打开文件：

```
char FullPath[32];
```

```
char DiskPath[8]={ "G:"};
```

```

FullPath[0]=0;

strcat(FullPath,DiskPath);

strcat(FullPath,"\\TMP.DAT");

DeviceDAT[DeviceCount]=CreateFile(FullPath,  GENERIC_READ  |  GENERIC_WRITE, FILE_SHARE_READ  |
FILE_SHARE_WRITE, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL |FILE_FLAG_NO_BUFFERING, 0);

FullPath[0]=0;

strcat(FullPath,DiskPath);

strcat(FullPath,"\\TMP.STA");

DeviceSTA[DeviceCount]=CreateFile(FullPath,  GENERIC_READ  |  GENERIC_WRITE, FILE_SHARE_READ  |
FILE_SHARE_WRITE, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL |FILE_FLAG_NO_BUFFERING, 0);

if (DeviceDAT[DeviceCount]==INVALID_HANDLE_VALUE || DeviceSTA[DeviceCount]==INVALID_HANDLE_VALUE)
{

    DeviceCount=-1; //-1 表示有错误发生

    break;

}

```

2)、读数据文件:

```

void __fastcall TGoodtelDevice::ReadData(char DeviceIndex,unsigned char * data)
{

    DWORD num;

    if (ErrorDevice[DeviceIndex]==0) {

        SetFilePointer(DeviceDAT[DeviceIndex] ,0,NULL,FILE_BEGIN);

        if (!ReadFile(DeviceDAT[DeviceIndex] ,data, BytesToRead, &num, NULL))

ReportError(DeviceIndex,"Error while read data.");// BytesToRead 为 512;

    }

}

```

3)、读状态文件:

```

struct TDeviceStatus {

    unsigned char WorkModes[8]; //不用

    unsigned char voltages[8]; //voltages[0]为电话线实时电压, 其他 7 个值用于多路录音盒

    char HardwareNo[16]; //固件序列号

```

```

    unsigned char CPUUsage; //不用。用于多路录音盒
    unsigned char channels; //不用。用于多路录音盒
    char VoltageValve; //不用
    char levels[8]; //不用
    long long int SignalTimes[8]; //不用
    long long int USBBoxTick; //固件的时间--从上电开始计时，单位为 1ms
    short HardwareVersion; //固件版本--例如 0x110 表示 1.16 版本，0x11E 为 1.30 版本
    //您所设置的每个通道的录音音量---音量有 0123 四个值，0 为调试模式，123 分别是小中大
    unsigned char volumes[8]; //只用 volumes[0]，其他 7 个值用于多路录音盒。
    //固件的播放缓冲区的上下半区的空闲标志，=0 表示上半区是空的，=1 表示下半区是空的
    char PlayEmptyHalf[8]; //只使用 PlayEmptyHalf[0]，其他 7 个值用于多路录音盒
    char blank[512]; //留空，保证这个结构体至少 512 字节
};

void __fastcall TGoodtelDevice::ReadStatus(char DeviceIndex)
{
    DWORD num;
    if (ErrorDevice[DeviceIndex]==0) {
        SetFilePointer(DeviceSTA[DeviceIndex] ,0, NULL, FILE_BEGIN);
        if (!ReadFile(DeviceSTA[DeviceIndex] ,&DeviceStatus, 512, &num,
NULL))ReportError(DeviceIndex,"Error while read status.");
        else VoltageAVR[DeviceIndex]=(VoltageAVR[DeviceIndex]*9+DeviceStatus.voltages[0])/10;
    }
}

```

4)、通过写入文件向模块发送命令：

注：当前版本的硬件支持 4 个命令：挂机 0，摘机 1，设置录音音量（10—13，13 表示调试模式）。

```

void __fastcall TGoodtelDevice::WriteCommand(char DeviceIndex,unsigned char v)
{
    unsigned char buffer[512];
    DWORD BytesWritten;
    buffer[0]=0x55; buffer[1]=0xAA; buffer[2]=0x66; buffer[3]=0xBB; buffer[4]=v;
}

```

```

        if (ErrorDevice[DeviceIndex]==0) {
            SetFilePointer(DeviceSTA[DeviceIndex] ,0,NULL,FILE_BEGIN);
            if (!WriteFile(DeviceSTA[DeviceIndex] ,buffer, 512, &BytesWritten,
NULL))ReportError(DeviceIndex,"Error while write command.");
        }
    }
}

```

5)、关闭文件:

```

CloseHandle(DeviceDAT[DeviceIndex]);
CloseHandle(DeviceSTA[DeviceIndex]);

```

6)、解析音频数据及线路事件 - 读文件 TMP.DAT 之后执行:

建议使用线程, 循环调用。每次读取 512 个字节, 会得到 8 个数据包, 每个数据包 64 个字节。

音频数据包: 第 0 个字节如果为 0, 则表示这是一个音频数据包, 第 1 个字节为这个数据包的信号强度。其余 62 个字节为具体的数据, 数据采用 8000hz/8bit PCM 格式。

线路事件数据包: 第 0 个字节如果为 100, 则表示这是一个事件包, 第 1 个数据为事件类型, 事件类型定义如下:

```

//各种事件
#define EVENT_FSK 3
#define EVENT_DTMF 4
#define EVENT_RING 5
#define EVENT_RING_END 6
#define EVENT_BUSY 7

```

废数据包: 第 0 个数据为其他值, 请丢弃这个数据包。

演示代码:

```

struct FSKDATA
{
    char DateTime[16];
    char CallerID[16];
    char SelfID[16];
    char CallerName[16];
}

```

```

FSKDATA *FSKData;

BytesRead=512;

PackSize=62;

i=0;

while (i<BytesRead) {

    //一个有效数据包

    if (DataBuffer[i]==0) {

        //把 62 个数据取出来

        memcpy(aData,DataBuffer,PackSize+2);

        //保存到文件——自己判断是否摘机

        SaveToFile(aData);

        packs++;

    } else if (DataBuffer[i]==100)

    switch (DataBuffer[i+1]) { //显示事件

    case EVENT_FSK:

        ReportMessage(" FSK: " );

        FSKData=DataBuffer+2;

        FSKStr=FSKData-> CallerID ;

        ReportMessage(FSKStr);

        break;

    case EVENT_DTMF:

        sprintf(s, " DTMF:  %d  ...  %c  ",(DataBuffer[i+2]+DataBuffer[i+3]*256) /

1000),(DataBuffer[i+2]+DataBuffer[i+3]*256) mod 1000));

        ReportMessage(s);

        break;

    case EVENT_RING:

        ReportMessage(" Ring... ");

        break;

    case EVENT_RING_END:

        ReportMessage(" Ring end. " );

    }

    i++;
}

```

```

        break;

    case EVENT_BUSY:

        ReportMessage(" Busy. " +IntToStr(DataBuffer[i+2]));

        break;

    }

    i=i+PackSize+2;

}

```

7、拨号：

//固件拨号，高级命令实际上就是固件拨号命令——以后升级，可能会增加其他命令

//command=255

//字符串 s 是要拨打的号码

```

void __fastcall TGoodtelDevice::WriteCommandAdvanced(char DeviceIndex, unsigned char command,
String s)
{
    unsigned char buffer[512];

    DWORD BytesWritten;

    int i, len;

    buffer[0]=0x55;  buffer[1]=0xAA;   buffer[2]=0x66;  buffer[3]=0xBB; buffer[4]=command;

    //C++Builder 的字符串，保存到数组 buffer 里

    len = s.Length();

    for (i=0; i<len; i++) buffer[5+i] = s[i+1];

    buffer[5+len] = 0;

    //固件拨号命令，支持参数（可以缺省）V,L,B

    buffer[6+len] = 'V';    buffer[7+len] = 24;                //拨号音量=24

    buffer[8+len] = 'L';    buffer[9+len] = 128;    buffer[10+len] = 0; //拨号音的长度=128 + 0 *
256; ---- 双字节 unsigned int, 单位毫秒

    buffer[11+len] = 'B';    buffer[12+len] = 24;                //偏置电压，默认 24

    if (ErrorDevice[DeviceIndex]==0) {

```

```

        SetFilePointer(DeviceSTA[DeviceIndex] ,0,NULL,FILE_BEGIN);

        if (!WriteFile(DeviceSTA[DeviceIndex] ,buffer, 512, &BytesWritten, NULL))
ReportError(DeviceIndex,"Error while write command.");

    }

}

```

8、播放语音：

```

void __fastcall TGoodtelDevice::WriteData(char DeviceIndex,unsigned char *data)
{
    DWORD BytesWritten;

    if (ErrorDevice[DeviceIndex]==0) {

        SetFilePointer(DeviceDAT[DeviceIndex] ,0,NULL,FILE_BEGIN);

        if (!WriteFile(DeviceDAT[DeviceIndex] ,data, 512, &BytesWritten, NULL))
ReportError(DeviceIndex,"Error while write data.");

    }

}

```

//往模块播放语音

//由于模块硬件设计的特殊性，播放的数据可能需要往上偏移一点，adjust 表示了偏移的幅度，adjust=32
即可

```

void __fastcall TGoodtelDevice::PlayData(char DeviceIndex,unsigned char * data,unsigned char
adjust)

```

```

{
    #define SectorSize 512

    unsigned char d[SectorSize];

    int i,v;

    ReadStatus(DeviceIndex);

    //要判断固件的播放状态

    //为了避免冲突，固件的播放缓冲区被分成了上下二个半区，固件正在播放一个半区时，你才能写入到
另一个半区

    if ((HalfToWrite[DeviceIndex]>1) ||

(HalfToWrite[DeviceIndex]==DeviceStatus.PlayEmptyHalf[0]))

```

```
{  
    //偏移  
    for (i=0;i<SectorSize;i++) {  
        v=d[i]+adjust;  
        if (v>255) d[i]=255; else d[i]=v;  
    }  
    WriteData(DeviceIndex,d);  
    HalfToWrite[DeviceIndex]=1-DeviceStatus.PlayEmptyHalf[0]; //切换到另一个半区  
}  
}
```

西安国咨软件有限公司

<http://www.gooods.com>

2024.10