**目录**

[EnumDevPath(枚举设备) 2](#_Toc476813462)

[EnumProtoCode（枚举指令码） 3](#_Toc476813463)

[OpenDev（打开设备） 4](#_Toc476813464)

[DevIoCtrl（控制通道） 5](#_Toc476813465)

[Write（打印数据发送） 6](#_Toc476813466)

[Read（设备数据接收） 7](#_Toc476813467)

[CloseDev（关闭设备） 8](#_Toc476813468)

[OpenLog（打开日志） 9](#_Toc476813469)

[SetLogRecord（写日志信息） 10](#_Toc476813470)

[CloseLog（关闭日志） 11](#_Toc476813471)

[GetLogRecord（获取日志信息） 12](#_Toc476813472)

[OpenDfu（固件加载） 13](#_Toc476813473)

[CloseDfu（关闭固件更新） 14](#_Toc476813474)

[fwInfo（固件信息） 15](#_Toc476813475)

[DFUStart（固件更新） 16](#_Toc476813476)

[LoadDevCfg（配置格式） 17](#_Toc476813477)

[GetDevCfgInfo（获取配置信息） 18](#_Toc476813478)

[SetDevCfgInfo（设置配置信息） 19](#_Toc476813479)

[附录一 参数内容说明 20](#_Toc476813480)

# EnumDevPath(枚举设备)

* **函数原型**

WORD EnumDevPath(WORD portType, OUT LPTSTR pathStr,IN OUT DWORD &sizes,WORD vID,WORD pID)

* **定义**

枚举设备的路径

* **返回值**

成功，返回枚举到的路径数量

失败，返回0

* **参数**

portType（In）

枚举端口类型，详细设置内容参考附录[1.1](#_1.1__portType参数内容)。

pathStr（Out）

路径字符，已多字符串的形式存放，字符串之间以0分割，以00结束。

sizes（In&Out）

（In）输入pathStr的分配空间大小，字符单位；

（Out）返回pathStr总长度，字符单位，含结束00。

vID（In）

当为0时，表示枚举所有的设备路径；

当为非0时，表示只枚举vID与其相同的设备路径。

pID（In）

当为0时，表示枚举所有的设备路径；

当为非0时，表示只枚举pID与其相同的设备路径。

* **说明**

1. 当枚举网络设备(NET\_PORT、NET\_CPORT、NET\_DPORT)时,vID用于设置设,备的UDP端口号；
2. sizes（In）需要大于或等于实际返回的有效字符串长度，否则函数返回的设备路径会小于实际数。

# EnumProtoCode（枚举指令码）

* **函数原型**

LPCTSTR EnumProtoCode(WORD idx,WORD &code)

* **定义**

枚举支持的指令码名称与对应的值（十进制）

* **返回值**

成功，返回指令码名称字符串

失败，返回NULL

* **参数**

Idx（In）

指令码编号（十六进制），用来获取指令码名称及值编号

Code(Out)

返回指令码的值（十进制）

# OpenDev（打开设备）

* **函数原型**

PORT\_HANDLE OpenDev(CONST LPPORTINFO lpInfo)

* **定义**

打开设备端口

* **返回值**

成功，返回设备端口句柄；

失败，返回INVALID\_PORT\_HANDLE（-1）。

* **参数**

lpInfo（In）

结构体变量，需要打开设备的端口信息，详细解释请参考附录[1.2](#_1.2___PORTINFO结构体)

* **说明**

1. lpInfo中，portType需要与path的类型保持一致，否则会打开失败，可以直接引用EnumDevPath(枚举设备)中的portType；
2. 设备端口打开成功后，设备重启，调用此函数会给一个新的序号值，原来的编号保留但已无法指向设备；
3. 最多可以同时打开10个设备端口，一个有效的句柄代表一个设备端口；
4. 设备不重启情况下，同一path（设备端口地址），portMode如果选择PORTINFO\_PMODE\_DATA（数据端口）时，可以连续调用此函数，赋予新的编号直至9，选择PORTINFO\_PMODE\_CTRL（控制端口）与PORTINFO\_PMODE\_ALL（所有端口）时，返回INVALID\_PORT\_HANDLE（-1）；

# DevIoCtrl（控制通道）

* **函数原型**

BOOL DevIoCtrl (PORT\_HANDLE pHandle, LPCTSTR ctrlCodeStr, CONST BYTE \*inDats, DWORD inLen, OUT DEVACK\_INFO \*outDats)

* **定义**

打开设备控制数据

* **返回值**

成功，返回TRUE

失败，返回FALSE

* **参数**

handle（In）

设备句柄，可以选择当前打开设备中的一个进行发送，跟OpenDev函数中的返回值对应

ctrlCodeStr（In）

设备提供的控制功能码，与EnumProtoCode中返回的指令码名称字符串关联，具体请参考附录[1.3](#_1.3)

InDats（In）

发送给设备的数据

inLen（In）

设备接收inDats的实际数据长度，字节单位

outDats（Out）

设备执行控制码时，设备的应答数据，详细解释请参考附录[1.4](#_1.4__DEVACK_INFO结构体)

* **说明**

1. 必须打开设备的控制端口；
2. 当ctrlCodeStr为“CTRL\_DEVCMD\_ST”，inDtas[0]的应为设备命令码。

# Write（打印数据发送）

* **函数原型**

BOOL Write (PORT\_HANDLE pHandle, LPCTSTR ctrlCodeStr, CONST BYTE \*inDats, DWORD inLen, OUT DEVACK\_INFO \*outDats)

* **定义**

写数据到设备的数据端口

* **返回值**

成功，返回TRUE

失败，返回FALSE

* **参数**

handle（In）

设备句柄，可以选择当前打开设备中的一个进行发送，跟OpenDev函数中的返回值对应

inDats（In）

需要写到设备的数据

inLen（In）

数据长度，字节单位

outLen（Out）

实际写到设备的数据长度

* **说明**

必须打开设备的数据端口

# Read（设备数据接收）

* **函数原型**

BOOL Read (PORT\_HANDLE pHandle, BYTE \*inDats, DWORD inLen, OUT DWORD \*outLen, OVERLAPPED \*lpOvlpd)

* **定义**

从设备的数据端口读数据

* **返回值**

成功，返回TRUE

失败，返回FALSE

* **参数**

pHandle（In）

设备句柄，可以选择当前打开设备中的一个进行发送，跟OpenDev函数中的返回值对应

inDats（Out）

需要接收设备数据的缓冲

inLen（Out）

缓冲长度，字节单位

outLen（Out）

实际读到设备数据长度

* **说明**

必须打开设备的数据端口

# CloseDev（关闭设备）

* **函数原型**

BOOL CloseDev (PORT\_PHANDLE pPHandle)

* **定义**

关闭设备端口

* **返回值**

成功，返回TRUE

失败，返回FALSE

* **参数**

pHandle（In）

设备句柄，可以选择当前打开设备中的一个进行发送，跟OpenDev函数中的返回值对应

* **说明**

关闭后，设备的数据/控制端口都将关闭，并释放设备句柄，如果有设备打开将重新被使用。

# OpenLog（打开日志）

* **函数原型**

BOOL OpenLog(LPCTSTR filePath)

* **定义**

启用设备日志，设置缓存路径

* **返回值**

成功，返回TRUE

失败，返回FALSE

* **参数**

filePath（In）

日志文件路径

为NULL时，默认缓存路径为“C:\Users\*当前用户*\AppData\Local\WdevObj\ wDevObj.log”

* **说明**

1. 当filePath为空时，默认缓存路径为C:\Users\*当前用户*\AppData\Local\WdevObj\ wDevObj.log；
2. 相同的日志路径，打开时将被覆盖。

# SetLogRecord（写日志信息）

* **函数原型**

void SetLogRecord (LPCTSTR str, BOOL newLine, BOOL timeMode, BOOL saveEn);

* **定义**

自定义日志信息写入到OpenLog函数设置路径下

* **返回值**

无

* **参数**

str（In）

写入的日志信息

newLine（In）

TRUE 表示添加换行到str结尾

FALSE 表示不添加换行到str结尾

timeMode（In）

TRUE 表示添加时间到str开头

FALSE 表示不添加时间到str开头

saveEn（In）

TRUE 表示将缓存信息保存在日志文件中

FALSE 表示不将缓存信息保存在日志文件中

# CloseLog（关闭日志）

* **函数原型**

void CloseLog(void)

* **定义**

关闭日志

* **返回值**

无

* **参数**

无

# GetLogRecord（获取日志信息）

* **函数原型**

LPCTSTR GetLogRecord(DWORD &cnts,BOOL reset)

* **定义**

获取日志中的字符信息

* **返回值**

成功，返回日志信息有效字符串

失败，返回NULL

* **参数**

cnts（Out）

返回值分配空间大小，如果分配空间大小不足以装下返回的字符串，则会返回NULL

reset（In）

TRUE，表示当前日志信息获取后删除

FALSE，表示不删除当前日志信息

# OpenDfu（固件加载）

* **函数原型**

BOOL OpenDfu(PORT\_PHANDLE pPHandle, LPCTSTR filePath,HWND hWnd)

* **定义**

加载固件文件并进行校验

* **返回值**

成功，TRUE

失败，FALSE

* **参数**

pHandle（In）

设备句柄，可以选择当前打开设备中的一个进行发送，跟OpenDev函数中的返回值对应

filePath（In）

固件文件路径

hWnd（Out）

接收固件更新信息的窗口句柄

* **说明**

必须打开设备的控制端口

# CloseDfu（关闭固件更新）

* **函数原型**

BOOL (PORT\_PHANDLE pPHandle)

* **定义**

关闭固件更新

* **返回值**

成功，TRUE

失败，FALSE

* **参数**

pHandle（In）

设备句柄，可以选择当前打开设备中的一个进行发送，跟OpenDev函数中的返回值对应

# fwInfo（固件信息）

* **函数原型**

BOOL fwInfo (PORT\_PHANDLE pPHandle, IN OUT DFU\_FWINFO & fwInfo, BOOL getMode)

* **定义**

固件信息获取与修改

* **返回值**

成功，返回TRUE

失败，返回FALSE

* **参数**

fwInfo（In&Out）

结构体变量，固件信息获取与设置，详细解释请参考附录[1.5](#_1.4__DFU_FWINFO结构体)

getMode（In）

TRUE，获取固件信息

FALSE，设置固件信息

* **说明**

1. 当getMode为FALSE时，进行固件信息设置dfuID需要与当前固件一致，否则会设置失败

# DFUStart（固件更新）

* **函数原型**

BOOL DFUStart (PORT\_PHANDLE pPHandle, DWORD tags)

* **定义**

开启固件更新

* **返回值**

成功，返回TRUE

失败，返回FALSE

* **参数**

pHandle（In）

设备句柄，可以选择当前打开设备中的一个进行发送，跟OpenDev函数中的返回值对应

Tag（In）

为0时，设备更新完成后，设备不重启

为1时，设备更新完成后，设备重启

# LoadDevCfg（配置格式）

* **函数原型**

BOOL LoadDevCfg (PORT\_PHANDLE pPHandle, LPCTSTR xmlPath)

* **定义**

加载设备配置的格式信息

* **返回值**

成功，返回TRUE

失败，返回FALSE

* **参数**

pHandle（In）

设备句柄，可以选择当前打开设备中的一个进行发送，跟OpenDev函数中的返回值对应

XmlPath（In）

格式信息的文件路径

* **说明**

1. 格式信息XmlPath以xml格式表述；
2. 当pHandle有效，即使XmlPath不为空且有效，也从设备中加载；当pHandle无效，则从XmlPath中加载。

# GetDevCfgInfo（获取配置信息）

* **函数原型**

BOOL GetDevCfgInfo(PORT\_HANDLE pHandle,void\* lpParam, OUT LPTSTR buf, IN OUT WORD &cnts, WORD tag, WORD noLoad)

* **定义**

获取设备配置数据

* **返回值**

成功，返回TRUE

失败，返回FALSE

* **参数**

pHandle（In）

设备句柄，可以选择当前打开设备中的一个进行发送，跟OpenDev函数中的返回值对应

lpParam（In）

配置项

buf（Out）

获取到的相关配置信息

cnts（In& Out）

（In）buf初始化大小；

（Out）返回buf有效字符串大小

tag（In）

1. 表示获取格式信息
2. 表示获取配置数据

noLoad（In）

0，从设备中加载全部配置信息

非0，从缓冲区中加载配置信息

* **说明**

1. 当tag等于0时，lpParam为指向(WORD\*)的配置项索引号；tag等于1时，lpParam为指向字符串(LPCTSTR)的配置项名。
2. 返回值为FALSE时，如果cnts也为0表示超出配置项的范围获取失败，cnts不为0表示需要缓冲的长度。

# SetDevCfgInfo（设置配置信息）

* **函数原型**

BOOL SetDevCfgInfo (PORT\_HANDLE pHandle,LPCTSTR name, LPCTSTR val, WORD cnts, WORD saveToDev)

* **定义**

设置设备配置信息

* **返回值**

成功，返回TRUE

失败，返回FALSE

* **参数**

pHandle（In）

设备句柄，可以选择当前打开设备中的一个进行发送，跟OpenDev函数中的返回值对应

name（In）

配置项名称，与GetDevCfgInfo函数tag为1时返回的buf的name值对应

val（In）

设置的数据，以字符串形式表示

Cnts（In）

无效

saveToDev（In）

0，设置在缓存区中

非0，设置到设备中

# 附录一 参数内容说明

声明头文件WDevDef.h

## 1.1 portType参数内容

|  |  |
| --- | --- |
| *名称* | *解释* |
| *INVALID\_PORT* | 无效 |
| *UART\_PORT* | 串口 |
| *LPT\_PORT* | 并口 |
| *USB\_PORT* | USB口 |
| *USBPRN\_PORT* | USB打印端口 |
| *HID\_PORT* | USB-HID端口 |
| *NET\_PORT* | 网口 |
| *NET\_DPORT* | 网口数据端口 |
| *NET\_CPORT* | 网口控制端口 |

## 1.2 PORTINFO结构体

typedef struct \_PORTINFO {

LPCTSTR path;

LONG readTimeout;

LONG writeTimeout;

WORD portType;

WORD portMode;

}PORTINFO, \*LPPORTINFO;

|  |  |
| --- | --- |
| *参数* | *解释* |
| path | 设备端口地址（与枚举函数中的pathStr一致） |
| readTimeout | 读端口数据超时时间，-1与0表示默认 |
| writeTimeout | 写端口数据超时时间，-1与0表示默认 |
| portType | 设备端口地址类型（与枚举函数中的portType一致） |
| portMode | 打开设备端口类型  PORTINFO\_PMODE\_DATA 0x01 数据端口  PORTINFO\_PMODE\_CTRL 0x02 控制端口  PORTINFO\_PMODE\_ALL 0x03 所有端口,包括数据端口和控制端口 |

## 1.3 指令码明细

|  |  |  |
| --- | --- | --- |
| *指令码编号* | *指令码字符串* | *解释* |
| 0X00 | CTRL\_CMD\_NOUSE |  |
| 0X01 | CTRL\_CMDGET\_MODEL | 获取设备型号 |
| 0X02 | CTRL\_CMDGET\_DEVNO | 获取设备序列号 |
| 0X03 | CTRL\_CMDGET\_PROTVER | 获取得实仿真规范版本号 |
| 0X04 | CTRL\_CMD\_CONNT | 设备连接 |
| 0X05 | CTRL\_CMD\_UNCONNT | 设备断开 |
| 0X06 | CTRL\_CMDSET\_PSW | 设置密码 |
| 0X07 | CTRL\_CMDGET\_USERDAT | 获取自定义标识 |
| 0X08 | CTRL\_CMDSET\_USERDAT | 设置自定义标识 |
| 0X09 | CTRL\_CMDGET\_DEVSTAT | 获取设备状态 |
| 0X0A | CTRL\_CMDGET\_PWSSTAT | 获取加密状态 |
| 0X0B | CTRL\_CMDSET\_ENCRYPT | 设置加密 |
| 0X0C | CTRL\_CMDGET\_DEVINFO | 获取设备信息 |
| 0X0D | CTRL\_CMD\_RESETCFG | 恢复出厂设置 |
| 0X0E | CTRL\_CMD\_CLSBUF | 清除缓存 |
| 0X0F | CTRL\_CMDGET\_STATISINFO | 获取数据统计 |
| 0X10 | CTRL\_CMDGET\_MAINTAININFO | 获取设备维修信息 |
| 0X11 | CTRL\_CMD\_DEVRESTART | 设置设备重启 |
| 0X12 | CTRL\_CMD\_DEVCHKSLF | 设备自检 |
| 0X13 | CTRL\_CMDGET\_WORKMODE | 获取工作模式 |
| 0X14 | CTRL\_CMDSET\_WORKMODE | 设置工作模式 |
| 0X15 | CTRL\_CMDGET\_VERINFO | 获取固件版本号 |
| 0X16 | CTRL\_CMD\_UDFMODE | 设置固件升级模式 |
| 0X17 | CTRL\_CMDGET\_CFGINFOS | 获取设备配置信息 |
| 0X1A | CTRL\_CMDGET\_CFGFMT |  |
| 0X1F | CTRL\_CMDGET\_CFGINFOE |  |
| 0X20 | CTRL\_CMDSET\_CFGINFOS | 设备配置 |
| 0X28 | CTRL\_CMDSET\_CFGINFOE |  |
| 0X29 | CTRL\_CMDGET\_DATAS | 数据块传输 |
| 0X2F | CTRL\_CMDGET\_DATAE |  |

## 1.4 DEVACK\_INFO结构体

typedef struct \_DEVACK\_INFO {

BYTE \*lpBuf;

WORD bufLen;

WORD datLen;

BYTE ackCode;

}DEVACK\_INFO;

|  |  |
| --- | --- |
| *参数* | *解释* |
| lpBuf | 设备返回的数据 |
| bufLen | 缓冲大小 |
| datLen | 数据长度 |
| ackCode | 应答码，0表示成功 |

## 1.5 DFU\_FWINFO结构体

typedef struct \_DFU\_FWINFO {

BYTE dfuID[5];

BYTE ver[4];

BYTE fwDesc[DFU\_FWTAG\_LEN + 1];

}DFU\_FWINFO;

|  |  |
| --- | --- |
| *参数* | *解释* |
| dfuID | 固件校验码，需要与设备匹配才能加载成功 |
| ver | 固件版本号，16进制显示 |
| fwDesc | 固件标签 |

# 附录二 配置文件XML格式说明

XML文件部分内容如下：

<CfgInfo DEVID="1" CFGID="1" CFGLen="80" CFGVer="1" CFGChk="0" Counts="45">

<CfgItem Name="数据位" Ofs="56" Bits="1" Type="3" Tag="0" Datas="8;7"/>

<CfgItem Name="AD\_Black值" Ofs="272" Bits="16" Type="1" Tag="1" MinVal="0" MaxVal="65535"/>

**说明：**

1、CfgInfo 中：

|  |  |  |
| --- | --- | --- |
| 名称 | 表示内容 | 字节数 |
| DEVID | 设备ID | 1 |
| CFGID | 配置ID | 1 |
| CFGLen | 配置长度 | 2 |
| CFGVer | 配置版本 | 1 |
| CFGChk | 配置校验位 | 1 |

Counts：表示配置项的个数。

1. CfgItem 中：

Name：表示配置项的名称；

Ofs：表示配置项从校验位后一项开始计算的比特位偏移量；

Bits：表示改配置项的值长度（单位bits）

Type：表示配置项取值的类型，具体表示如下：

0-无效

1-数值

2-字符

3-选择

Tag：表示配置项取值的表示方式，具体表示如下：

0-默认

1-类型为数值的配置项的值是低位前高位后

2-类型为字符的配置项的值以十六进制表示

Datas：配置项的取值可选项；

MinVal：配置项取值的最小值；

MaxVal：配置项取值的最大值；