

TxPrnMod.dll 要和应用程序放在同一个目录下

本连接文件可以用于并口，串口，USB 口，网口。

使用时的定义如下

```
#define TX_TYPE_NONE      0
#define TX_TYPE_USB      1   这个是 USB 口
#define TX_TYPE_LPT      2   这个是并口
#define TX_TYPE_COM      3   这个是串口
#define TX_TYPE_NET      6   这个是网口
```

## 函数说明

**BOOL WINAPI TxOpenPrinter(DWORD Type, DWORD\_PTR**

**Idx);**

这个函数是在使用打印机时，第一要用到的函数

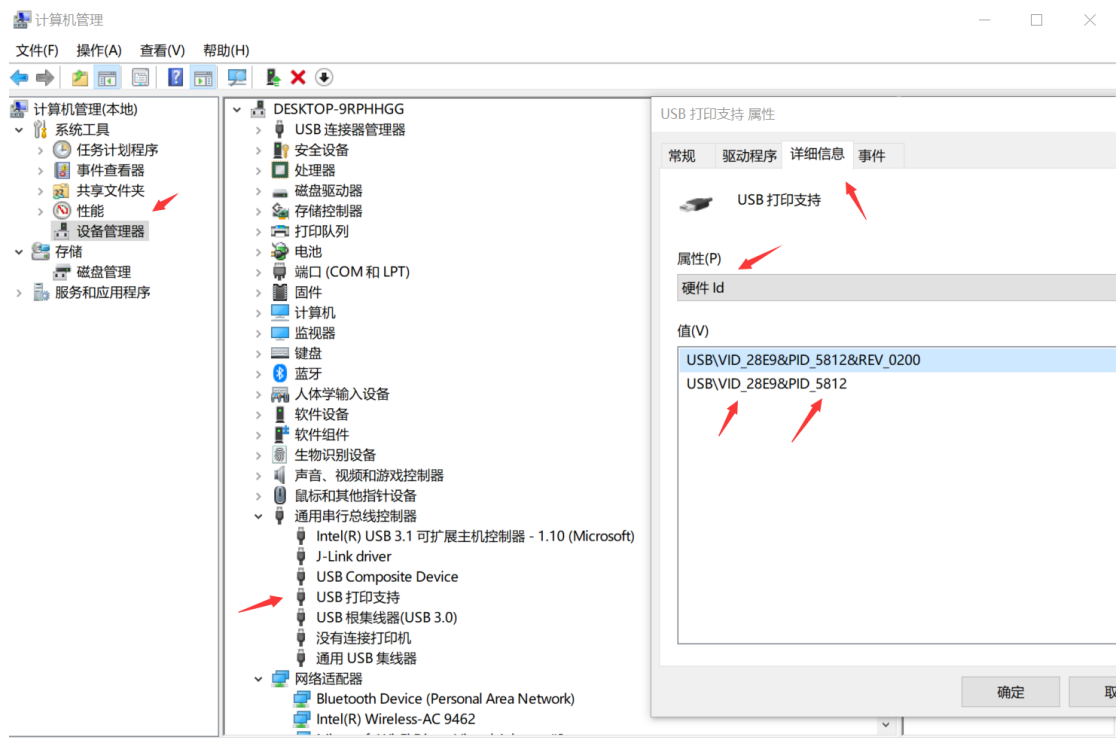
其中的 Type 见前面的定义，若等于 1 就是使用 USB 口，Idx 从 0 开始，是指端口号

比如使用串口的话，Idx=0,就是用 com1,=1 就是用 com2。

使用 USB 的话，Idx=0, 就是本打印机第 1 次插到电脑的 USB 口生成的那个 USB00x,如果同时电脑的 USB 同时插了

2 台 USB 的打印机，那么 Idx=1,就是使用第 2 台的打印。

在同时接了 2 台 USB 打印机的情况下，为了准确定位打印机，建议使用 VID,PID 组合参数的方式来打开想要使用的打印机。



上图是 1 台 USB 打印机的 VID 和 PID 的值，那么如何使用 VID,PID 来进行调用呢？

只需要把 TxOpenPrinter 调用的第 2 个参数改一下，第 2 个参数 Idx，如果小于 0x10000，则是按之前的序号工作。如果使用 VID,PID 参数组合的按这样的方式进行计算：

Idx=(VID<<16)+PID，或者 Idx=VID\*65536+PID。其他操作不变。

使用网口的话，第1个参数为6，第2个参数Idx为“IP+端口”在字符串，如"192.168.1.10:4567"。  
如果省略端口，则默认为9100。

下面是打印状态的各种状况的定义

#define TX_STAT_NOERROR	0x0008	无故障
#define TX_STAT_SELECT	0x0010	处于联机状态
#define TX_STAT_PAPEREND	0x0020	缺纸
#define TX_STAT_BUSY	0x0080	繁忙
#define TX_STAT_DRAW_HIGH	0x0100	钱箱接口的电平（整机使用的，模块无用）
#define TX_STAT_COVER	0x0200	打印机机芯的盖子打开
#define TX_STAT_ERROR	0x0400	打印机错误
#define TX_STAT_RCV_ERR	0x0800	可恢复错误（需要人工干预）
#define TX_STAT_CUT_ERR	0x1000	切刀错误
#define TX_STAT_URCV_ERR	0x2000	不可恢复错误
#define TX_STAT_ARCV_ERR	0x4000	可自动恢复的错误
#define TX_STAT_PAPER_NE	0x8000	快要没有纸了

## **DWORD WINAPI TxGetStatus();**

获取打印机状态

返回值为 TX\_STAT\_XXXX，调用失败则为 0。

返回值 TX\_STAT\_XXXX 见上面的位定义，是各种状态的组合，是按位来处理的，实际判断时要将返回的

TX\_STAT\_XXXX 按上面的位定义来按位判断，得到打印的状态

特别提示：在使用 USB 口时最好先使用这个函数来读取打印机的状态,可以很快获得打印机的是否忙，是否有纸和是否错误的状态，其它的状态不能获得。若打印机有纸和无故障的不忙的情况下，可以调用下面的函数  
来获取其它的信息，比如纸将尽等等

特别说明：因为历史遗留原因，之前有 2 种机器的这个函数返回比较特殊，不适用于上面的参数定义，那 2 种机器只能返回 4 种值，具体状态和对应的值如下：

有纸： 0x58

有纸+抬头： 0xd0

无纸： 0x70

无纸+抬头： 0xf0

## **DWORD WINAPI TxGetStatus2();**

获取打印机状态

返回值为 TX\_STAT\_XXXX，调用失败则为 0。

与上一函数类似，区别是 USB 接口通过指令查询状态。这个函数下，USB 可以获得打印机更多的信息。最好是在调用前一个函数确认打印有纸，无故障，不忙的情况下调用此函数。

如果是并口或 USB 接口的，只是想判断是否有纸的，建议用 TxGetStatus()，因为这个主要是给并口和 USB 接口使用的，只是判断是否有纸。如果是想判断更多的信息，比如纸将尽等，那要用 TxGetStatus2()，注意这个函数并口不可以使用的，只能是串口和 USB 接口，网口可以使用。

那么用 TxGetStatus2()得到的值要如何判断呢？

我们这个是按位来定义的，把所以故障信息定义为不同的位，如果某位=1 那么就是那位对应的故障了。

可以用个简单的算法，比如要判断是否纸将尽，就用快要没有纸了对应的定义 0x8000 和调用 TxGetStatus2()得到的值“与”一下，如果结果不为 0，那么就是纸快要用了，如果等于 0，那说明纸还有比较多的。

那些故障位，可以只判断常见的故障就可以了

### **void WINAPI TxClosePrinter();**

关闭所连接的打印机。

### **BOOL WINAPI TxWritePrinter(const char\* buf, DWORD len);**

输出指定的缓冲

\*buf : 为缓冲区指针

len : 为缓冲区长度

把数据直接发送给打印机，特殊适用于本开发包没有定义的函数。开发者可以直接把指令或数据发给打印机。

### **DWORD WINAPI TxReadPrinter(char\* buf, DWORD len);**

从打印机读取数据

\*buf : 为接收缓冲

len: 为欲读取的字节数

这个是读取打印机返回值的函数，适用于发送了带返回值的指令。比如发 1 条查询打印机纸将尽的指令（10 04 04），就可以用这个函数要读取打印机的返回值。

### **void WINAPI TxInit();**

发送初始化指令，初始化打印机。

**void WINAPI TxOutputString(char\* str);**

输出字符串（以\0 结束）。就是把字符串直接发到打印机。

**void WINAPI TxNewline();**

输出回车、换行，就是向打印机发送 0x0D,0x0A 的数据，打印机接收到后，会将缓冲区中的数据打印并走纸 1 行。

**void WINAPI TxOutputStringLn(char\* str);**

输出字符串（以\0 结束），并自动添加回车、换行

**void WINAPI TxResetFont();**

恢复字体效果（大小、粗体等）为原始状态。

**BOOL WINAPI TxPrintImage(const char\* path);**

打印图形文件

path = 图形文件完整路径

就是把指定路径的图形文件打印出来。

**void WINAPI TxSetupSerial(DWORD attr);**

设置串口参数

串口设置参数的定义如下

```
#define TX_SER_BAUD_MASK    0xFF000000    波特率
#define TX_SER_BAUD9600    0x00000000    9600 的波特率
#define TX_SER_BAUD19200   0x01000000    19200 的波特率
#define TX_SER_BAUD38400   0x02000000    38400 的波特率
#define TX_SER_BAUD57600   0x03000000    57600 的波特率
#define TX_SER_BAUD115200  0x04000000    115200 的波特率

#define TX_SER_DATA_MASK    0x00FF0000    数据位
#define TX_SER_DATA_8BITS   0x00000000    8 位数据位
#define TX_SER_DATA_7BITS   0x00010000    7 为数据位

#define TX_SER_PARITY_MASK  0x0000FF00    校验
#define TX_SER_PARITY_NONE 0x00000000    无校验
#define TX_SER_PARITY_EVEN 0x00000100    偶校验
#define TX_SER_PARITY_ODD  0x00000200    奇校验

#define TX_SER_STOP_MASK    0x000000F0    停止位
```

```

#define TX_SER_STOP_1BITS 0x00000000 1 位停止位
#define TX_SER_STOP_15BITS 0x00000010 1.5 位停止位
#define TX_SER_STOP_2BITS 0x00000020 2 位停止位

#define TX_SER_FLOW_MASK 0x0000000F 流控制
#define TX_SER_FLOW_NONE 0x00000000 无流控
#define TX_SER_FLOW_HARD 0x00000001 硬件流控（DTR/DSR 方式）
#define TX_SER_FLOW_SOFT 0x00000002 软件流控（XON/XOFF 方式）

```

attr 参数是波特率，数据位，校验位，停止位，流控位的组合

```
TxSetupSerial(TX_SER_BAUD38400|TX_SER_DATA_8BITS|TX_SER_PARITY_NONE|TX_SER_STOP_1BITS|TX_SER_FLOW_HARD);
```

上面的例子就是设置串口位 38400 的波特率，8 位数据位，无校验，1 位停止位，硬件流控。

## **BOOL WINAPI TxPrintBarcode(DWORD type, char\* str);**

打印条码

type 是打印的条码的类型，后面是符合要求的条码数据

条码类型的定义

```

#define TX_BAR_UPCA 65
#define TX_BAR_UPCE 66
#define TX_BAR_EAN13 67
#define TX_BAR_EAN8 68
#define TX_BAR_CODE39 69
#define TX_BAR_ITF 70
#define TX_BAR_CODABAR 71
#define TX_BAR_CODE93 72
#define TX_BAR_CODE128 73

```

如下例子是打印 UPCA 的条码，数据为“12345678901”

```
TxPrintBarcode(TX_BAR_UPCA,"12345678901");
```

注意 CODE128 是特殊的条码，不能这样直接调用，因为要先设置使用的子集，所以要使用 CODE128 码，要用直接发送指令的函数，先设置选用的子集然后再发条码数据。如果确实需要使用 CODE128 码的话，可以联系技术支持工程师。

## **void WINAPI TxPrintQRcode(const char \*data, WORD len);**

打印 QR 码

param data : 内容指针

param len : 内容长度

打印 QR 码的顺序是先要设置打印 QR 码的点的尺寸，然后设置纠错等级，然后发送要打印的数据（请注意目前只能支持到版本 7）。

下面是打印 1 个 QR 码并走纸的例子调用

```
char* url="http://www.baidu.com";
TxDoFunction(TX_QR_DOTSIZE,6,0);
TxDoFunction(TX_QR_ERRLEVEL,TX_QR_ERRLEVEL_M,0);
TxPrintQRcode(url,strlen(url));
TxDoFunction(TX_FEED,100,0);
```

**void WINAPI TxDoFunction(DWORD func, int param1, int param2);**

执行特殊功能。就是向打印机发送设置指令，或着走纸指令，定位指令等，具体要见下面做好的定义

有的功能需要传入两个参数，有的功能只需要一个参数（此时 param2=0 就可以了）。

TxDoFunction 的 func 的功能类型定义

**#define TX\_FONT\_SIZE 1**

控制字体的放大倍数功能，后面的 2 个参数 0 为正常大小，递增 1 为增大 1 倍，如此类推，最大为 7；参数 1(param1)为字体宽的倍数，参数 2(param2)为字体高的倍数。

下面的例子就是使用 4 倍宽 3 倍高的字体：

```
TxDoFunction(TX_FONT_SIZE,TX_SIZE_3X,TX_SIZE_2X);
```

**#define TX\_FONT\_ULINE 2**

控制是否有下划线功能，只需 1 个参数，param1=TX\_ON(有下划线)，=TX\_OFF(无小划线)，param2=0 就可以了。

下面的例子就是使用下划线的调用：

```
TxDoFunction(TX_FONT_ULINE,TX_ON,0);
```

**#define TX\_FONT\_BOLD 3**

控制是否粗体功能，只需 1 个参数，param1=TX\_ON(使用粗体)，=TX\_OFF(不使用粗体)，param2=0 就可以了。

下面的例子就是使用粗体的调用

```
TxDoFunction(TX_FONT_BOLD,TX_ON,0);
```

**#define TX\_SEL\_FONT 4**

选择英文字体功能，也是只需要 1 个参数，TX\_FONT\_A(12X24) 或 TX\_FONT\_B(9X17)，param2=0 就可以了。

下面的例子就是使用 9X17 的字体调用的

```
TxDoFunction(TX_SEL_FONT,TX_FONT_B,0);
```

**#define TX\_FONT\_ROTATE 5**

字体是否旋转 90 度功能，只需要 1 个参数。

下面的例子就是使用 9X17 的字体调用的

```
TxDoFunction(TX_FONT_ROTATE,TX_ON,0);
```

#### **#define TX\_ALIGN 6**

控制对齐功能，参数为 TX\_ALIGN\_XXX，只需要 1 个参数。

#define TX\_ALIGN\_LEFT 0 左对齐的设置参数

#define TX\_ALIGN\_CENTER 1 对中设置的参数

#define TX\_ALIGN\_RIGHT 2 右对齐的设置参数

下面例子是对中打印的调用

```
TxDoFunction(TX_ALIGN,TX_ALIGN_CENTER,0);
```

```
TxOutputStringLn("center");
```

#### **#define TX\_CHINESE\_MODE 7**

中文/英文模式切换功能，只需要 1 个参数。

下面的例子是进入中文方式的调用

```
TxDoFunction(TX_CHINESE_MODE,TX_ON,0);
```

#### **#define TX\_FEED 10**

执行走纸功能，只需要 1 个参数，参数以 0.125 毫米为单位，这个参数最大不能超过 255.

下面的例子是走纸 30 毫米的调用

```
TxDoFunction(TX_FEED,240,0);
```

#### **#define TX\_UNIT\_TYPE 11**

设置动作单位(无效)

#### **#define TX\_CUT 12**

执行切纸功能，第一参数指明类型，第二参数指明切纸前的走纸距离

切纸的类型有 2 种:

#define TX\_CUT\_FULL 0 全切的设置参数(和黑标检测关联)

#define TX\_CUT\_PARTIAL 1 半切的设置参数 (和黑标检测关联)

#define TX\_PURECUT\_FULL 2 全切 (与黑标无关的切纸)

#define TX\_PURECUT\_PARTIAL 3 半切 (与黑标无关的切纸)

下面的例子就是不走纸直接全切的调用。

```
TxDoFunction(TX_CUT, PURECUT_FULL ,0);
```

#### **#define TX\_HOR\_POS 13**

绝对水平定位功能，只需要 1 个参数，参数以 0.125 毫米为单位。

下面的例子就是定位在离左边 20mm 的地方开始处理字符数据 (或着说打印也可以)。这个只是一行有效的。

```
TxDoFunction(TX_HOR_POS,160,0);
```

#### **#define TX\_LINE\_SP 14**

设置行间距功能，只需要 1 个参数，参数是以点数为单位的。

下面的例子就是设置行间距为 30 点（默认的参数）

```
TxDoFunction(TX_LINE_SP,30,0);
```

#### **#define TX\_BW\_REVERSE      15**

设置字体是否黑白翻转功能，只需要 1 个参数

下面的例子是使用黑白翻转打印功能的调用。

```
TxDoFunction(TX_BW_REVERSE,TX_ON,0);
```

#### **#define TX\_UPSIDE\_DOWN      16**

设置是否倒置打印功能，只需要 1 个参数

下面的例子是使用倒置打印功能的调用。

```
TxDoFunction(TX_UPSIDE_DOWN,TX_ON,0);
```

#### **#define TX\_INET\_CHARS      17**

选择国际字符集功能，只需要 1 个参数,通常这个设置也是在英文方式下使用的，只是针对 12 个特定的 ASCII 码不同国家的使用的字形不同，默认是 0，表示是使用美国的字符集。

下面的例子是设置国际字符集为 1

```
TxDoFunction(TX_INET_CHARS,1,0);
```

#### **#define TX\_CODE\_PAGE      18**

选择字符代码页功能，只需要 1 个参数，通常是 0~n,表示选择的代码页参数，详见打印机说明书

一般默认是 0，是表示 PC437 的代码页，这个功能要只在英文方式下有效。

下面的例子是设置字符代码页为 3

```
TxDoFunction(TX_CODE_PAGE,3,0);
```

#### **#define TX\_CH\_ROTATE      19**

设定汉字旋转功能，只需要 1 个参数，可以表示 3 种选择，如下所示：

```
#define TX_CH_ROTATE_NONE 0 不旋转
```

```
#define TX_CH_ROTATE_LEFT 1 向左旋转
```

```
#define TX_CH_ROTATE_RIGHT 2 向右旋转
```

下面的例子是设置汉字向左旋转的功能

```
TxDoFunction(TX_CH_ROTATE,TX_CH_ROTATE_LEFT,0);
```

#### **#define TX\_CHK\_BMARK      20**

寻找黑标黑标

TX\_CHK\_BMARK 是执行黑标检测，这个动作本身是不需要参数的，所以就是

```
TxDoFunction(TX_CHK_BMARK,0,0)
```

#### **#define TX\_SET\_BMARK      21**

设置黑标相关偏移量功能

这里有 2 个参数要设置，起始打印位置相对于黑标检测位置的偏移量和切/撕纸位置相对于黑标检测位置的偏移量。

TxDoFunction(TX\_SET\_BMARK,TX\_BM\_START,参数), 这样是设置起始打印位置相对



于黑标检测位置的偏移量。

TxDoFunction(TX\_SET\_BMARK,TX\_BM\_TEAR ,参数), 这样是切/撕纸位置相对于黑标检测位置的偏移量。

## **#define TX\_PRINT\_LOGO        22**

打印已下载好的 LOGO 的功能，只需 1 个参数，可以表示 4 种选择，如下所示：

- #define TX\_LOGO\_1X1        0    对应 203X203 的点密度，就是正常大小。
- #define TX\_LOGO\_1X2        1    对应 203x101 的点密度，就是 LOG 在水平方向上放大了 2 倍。
- #define TX\_LOGO\_2X1        2    对应 101x203 的点密度，就是 LOG 在垂直方向上放大了 2 倍。
- #define TX\_LOGO\_2X2        3    对应 101x101 的点密度，就是 LOG 在水平和垂直方向上放大了 2 倍。

下面的例子就是正常打印下载的 LOGO

```
TxDoFunction(TX_PRINT_LOGO,TX_LOGO_1x1,0);
```

## **#define TX\_BARCODE\_HEIGHT    23**

设定条码高度功能,只需要 1 个参数,单位为点数

下面的例子是设置的条码打印的高度为 15 点：

```
TxDoFunction(TX_BARCODE_HEIGHT,15,0);
```

## **#define TX\_BARCODE\_WIDTH    24**

设定条码宽度功能,只需要 1 个参数,单位为点数, 不能小于 2.

下面的例子是设置的条码打印的宽度高度为 3 点：

```
TxDoFunction(TX_BARCODE_WIDTH,3,0);
```

## **#define TX\_BARCODE\_FONT      25**

选择条码 HRI 字符的打印位置,只需要 1 个参数，可以表示 4 种选择，如下所示：

- #define TX\_BAR\_FONT\_NONE    0    不打印
- #define TX\_BAR\_FONT\_UP      1    打印在条码的上面
- #define TX\_BAR\_FONT\_DOWN    2    打印在条码的下面
- #define TX\_BAR\_FONT\_BOTH    3    条码的上面下面都打印

下面的例子就是设置打印条码时条码的 HRI 字符打印在条码的下面

```
TxDoFunction(TX_BARCODE_FONT,TX_BAR_FONT_DOWN,0);
```

## **#define TX\_FEED\_REV          26**

执行反向走纸功能，只需要 1 个参数，参数以 0.125 毫米为单位。

下面的例子是反向走纸 30 毫米的调用

```
TxDoFunction(TX_FEED_REV,240,0);
```

## **#define TX\_QR\_DOTSIZE        27**

设置 QR 码的点大小，2<param1<10

下面的例子是设置 QR 码 6 点大小的调用

```
TxDoFunction(TX_QR_DOTSIZ,6,0);
```

**#define TX\_QR\_ERRLEVEL 28**

设置 QR 码的纠错等级，有 4 个等级可以选择，等级越高，可以支持的字符数越少（在同一个版本的情况下）

```
#define TX_QR_ERRLEVEL_L 0x31
```

```
#define TX_QR_ERRLEVEL_M 0x32
```

```
#define TX_QR_ERRLEVEL_Q 0x33
```

```
#define TX_QR_ERRLEVEL_H 0x34
```

下面的例子是设置为 L 的纠错等级的调用

```
TxDoFunction(TX_QR_ERRLEVEL,TX_QR_ERRLEVEL_L,0);
```

TxDoFunction 的参数定义（需和功能类型对应）

```
#define TX_ON 1 功能设置有效的参数
```

```
#define TX_OFF 0 功能设置无效的参数
```

```
#define TX_CUT_FULL 0 全切的设置参数(和黑标检测关联)
```

```
#define TX_CUT_PARTIAL 1 半切的设置参数（和黑标检测关联）
```

```
#define TX_PURECUT_FULL 2 全切（与黑标无关的切纸）
```

```
#define TX_PURECUT_PARTIAL 3 半切（与黑标无关的切纸）
```

```
#define TX_FONT_A 0 12X24 点阵的设置参数
```

```
#define TX_FONT_B 1 9X17 点阵的设置参数
```

```
#define TX_ALIGN_LEFT 0 左对齐的设置参数
```

```
#define TX_ALIGN_CENTER 1 对中设置的参数
```

```
#define TX_ALIGN_RIGHT 2 右对齐的设置参数
```

控制字体倍数的参数设置

```
#define TX_SIZE_1X 0 正常大小
```

```
#define TX_SIZE_2X 1 2 倍大小
```

```
#define TX_SIZE_3X 2 3 倍大小
```

```
#define TX_SIZE_4X 3 4 倍大小
```

```
#define TX_SIZE_5X 4 5 倍大小
```

```
#define TX_SIZE_6X 5 6 倍大小
```

```
#define TX_SIZE_7X 6 7 倍大小
```

```
#define TX_SIZE_8X 7 8 倍大小
```

```
#define TX_UNIT_PIXEL 0 对应 TX_UNIT_TYPE
```

```
#define TX_UNIT_MM 1
```

#define TX\_CH\_ROTATE\_NONE 0 对应 TX\_CH\_ROTATE

#define TX\_CH\_ROTATE\_LEFT 1

#define TX\_CH\_ROTATE\_RIGHT 2

#define TX\_BM\_START 1 起始打印位置相对于黑标检测位置的偏移量

#define TX\_BM\_TEAR 2 切/撕纸位置相对于黑标检测位置的偏移量

#define TX\_LOGO\_1X1 0 对应 TX\_PRINT\_LOGO

#define TX\_LOGO\_1X2 1 /\*203x101\*/

#define TX\_LOGO\_2X1 2 /\*101x203\*/

#define TX\_LOGO\_2X2 3 /\*101x101\*/

#define TX\_BAR\_FONT\_NONE 0 对应 TX\_BARCODE\_FONT

#define TX\_BAR\_FONT\_UP 1

#define TX\_BAR\_FONT\_DOWN 2

#define TX\_BAR\_FONT\_BOTH 3

QR 码纠错等级参数

#define TX\_QR\_ERRLEVEL\_L 0x31

#define TX\_QR\_ERRLEVEL\_M 0x32

#define TX\_QR\_ERRLEVEL\_Q 0x33

#define TX\_QR\_ERRLEVEL\_H 0x34

### 关于例子程序的使用说明

那个 DEMO.EXE 的程序是使用 DLL 的例子程序，里面的使用可以见 DEMO.C,那个 DEMO.EXE 的程序需要在命令行下运行的，运行后，会有对话选择，分别选择 USB,并口，串口的