

通用串口库使用说明

1. 简介

本文主要介绍了Bcd_SerialLib动态库的相关接口和使用说明，旨在让大家更方便的使用串口通信相关功能。目标读者是需要使用串口通信编程的开发或者测试人员。

2. 相关信息定义

```
/* 串口相关信息 */
typedef struct _BCD_SERIAL_INFO_
{
    unsigned int nComNum;           // 端口号
    unsigned int nBaudrate;         // 波特率
    unsigned int nDataBits;         // 数据位
    unsigned int nParity;           // 校验位
    unsigned int nStopBits;         // 停止位
    unsigned int nEnableEvent;      // 是否利用事件打开
    unsigned int nEnableDtrControl; // Dtr控制使能
    unsigned int nEnableRtsControl; // Rts控制使能
    unsigned int arrReserved[8];    // 保留字段
}BCD_SERIAL_INFO;

/* 读取数据时，触发的回调函数类型 */
typedef void (__stdcall * SRL_ReadCallBackFunc)(void * pInfo, void * pUser);

/* 错误码信息 */
#define SRL_RET_OK          (0)           // 返回值正常
#define SRL_RET_ERR        (-1)          // 内部流程出错
#define SRL_RET_RES        (-2)          // 内部资源分配出错
#define SRL_RET_ORDER      (-3)          // 调用顺序出错
#define SRL_RET_TWICE      (-4)          // 重复调用
#define SRL_RET_PARAM      (-5)          // 设置参数异常
#define SRL_RET_NODATA     (-6)          // 无串口数据
#define SRL_RET_BUFLLEN    (-7)          // 缓存数据长度超过最大缓存长度(1024)
```

3. 相关接口定义

```
/**
 * @fn      BCD_SRL_GetLibVersion()
 * @brief   获取版本信息
 * @param   无
 * @return  返回串口库版本信息
 */
BCD_SERIALDLL_API unsigned int __stdcall BCD_SRL_GetLibVersion();

/**
 * @fn      BCD_SRL_CreateHandle()
 * @brief   创建串口对应句柄
 * @param   handle [OUT] 待创建的串口句柄
 * @param   pstSerialInfo [IN] 待创建的串口信息
 * @return  成功，返回SRL_RET_OK；失败，见错误返回值
 */
BCD_SERIALDLL_API int __stdcall BCD_SRL_CreateHandle(OUT void ** handle,
                                                    IN const BCD_SERIAL_INFO * pstSerialInfo);

/**
 * @fn      BCD_SRL_Write()
 * @brief   向串口写入信息
 */
```

```

* @param handle [IN] 对应的句柄
* @param strInfo [IN] 写入串口的信息
* @param nLen [IN] 写入信息的长度
* @return 成功, 返回SRL_RET_OK; 失败, 见错误返回值
*****/
BCD_SERIALDLL_API int __stdcall BCD_SRL_Write(IN void * handle,
                                              IN const char * strInfo,
                                              IN const unsigned int nLen);

/*****
* @fn BCD_SRL_Read()
* @brief 从串口中读取信息
* @param handle [IN] 对应的句柄
* @param strInfo [OUT] 读取串口数据
* @param nBufLen [IN] 输入Buf对应的长度
* @param nRecvLen [OUT] 收到的数据长度
* @return 成功, 返回SRL_RET_OK; 失败, 见错误返回值
*****/
BCD_SERIALDLL_API int __stdcall BCD_SRL_Read(IN void * handle,
                                              OUT char * strInfo,
                                              IN const unsigned int nBufLen,
                                              OUT unsigned int & nRecvLen);

/*****
* @fn BCD_SRL_ReadBlock()
* @brief 从串口中读取确定长度的信息
* @param handle [IN] 对应的句柄
* @param nBlockLen [IN] 设定的报文块大小
* @param strInfo [OUT] 读取串口数据
* @param nBufLen [IN] 收到的数据长度
* @return 成功, 返回SRL_RET_OK; 失败, 见错误返回值
*****/
BCD_SERIALDLL_API int __stdcall BCD_SRL_ReadBlock(IN void * handle,
                                                  IN const unsigned int nBlockLen,
                                                  OUT char * strInfo,
                                                  IN const unsigned int nBufLen);

/*****
* @fn BCD_SRL_RigisterReadCallBack()
* @brief 注册读取串口后信息的回调函数
* @param handle [IN] 对应的句柄
* @param pFunc [IN] 注册读取回调函数（可为NULL）
* @param pUser [IN] 注册读取回调函数对应的类（可为NULL）
* @return 成功, 返回SRL_RET_OK; 失败, 见错误返回值
*****/
BCD_SERIALDLL_API int __stdcall BCD_SRL_RigisterReadCallBack(IN void * handle,
                                                             IN SRL_ReadCallBackFunc pFunc,
                                                             IN void * pUser);

/*****
* @fn BCD_SRL_GetSerialInfo()
* @brief 获取串口设置信息
* @param handle [IN] 对应的句柄
* @param pstSerialInfo [OUT] 获取对应的串口信息
* @return 成功, 返回SRL_RET_OK; 失败, 见错误返回值
*****/
BCD_SERIALDLL_API int __stdcall BCD_SRL_GetSerialInfo(IN void * handle,
                                                       OUT BCD_SERIAL_INFO * pstSerialInfo);

/*****
* @fn BCD_SRL_DestroyHandle()
* @brief 销毁串口句柄, 释放相应资源
* @param 无
* @return 成功, 返回SRL_RET_OK; 失败, 见错误返回值
*****/

```

```
BCD_SERIALDLL_API int __stdcall BCD_SRL_DestroyHandle(IN void * handle);
```

4. 使用Demo

```
#include <iostream>
#include <Windows.h>
#include "Bcd_SerialCmdLib.h"

// 添加Bcd_SerialDll.lib引用
#pragma comment(lib, "Bcd_SerialDll.lib")

using namespace std;

static void * g_hSerial = NULL;    // 全局串口句柄，用于串口句柄的读写

void __stdcall SerialReadCallBack(void * pInfo, void * pUser)
{
    // 回调函数。如果接收到的串口信息长度大于3，则回复"Hello, serial."信息
    if (strlen((char *)pInfo) > 3)
    {
        BCD_SRL_Write(g_hSerial,
                        "Hello, serial.",
                        strlen("Hello, serial.));
    }

    return;
}

void main()
{
    /* 初始化相关参数 */
    unsigned int nRecvLen = 0;           // 串口返回信息的长度
    char strRecvInfo[128] = {0};        // 收到的串口信息

    /* 设置串口信息 */
    BCD_SERIAL_INFO stSerialInfo = {0};
    stSerialInfo.nComNum = 4;
    stSerialInfo.nBaudrate = 9600;
    stSerialInfo.nDataBits = 8;
    // 设置0，表示校验位为No（No-0;Odd-1;Even-2;Mark-3;Space-4）
    stSerialInfo.nParity = 0;
    // 设置0，表示停止位为1（1-0;1.5-2;2-2）
    stSerialInfo.nStopBits = 0;
    stSerialInfo.nEnableEvent = 0;
    stSerialInfo.nEnableDtrControl = 0;
    stSerialInfo.nEnableRtsControl = 0;

    /* 根据串口信息，创建串口句柄 */
    int nRet = BCD_SRL_CreateHandle(&g_hSerial, &stSerialInfo);
    if (nRet != SRL_RET_OK)
    {
        cout << "BCD_SRL_CreateHandle failed..." << endl;
        Sleep(3000);
        return;
    }
    cout << "BCD_SRL_CreateHandle succ..." << endl;

    /* 设置串口读取信息时的回调函数（如无需回调，可省略） */
    nRet = BCD_SRL_RegisterReadCallBack(g_hSerial,
                                        SerialReadCallBack,
                                        NULL);

    if (nRet != SRL_RET_OK)
```

```

{
    cout << "BCD_SRL_RigisterReadCallBack failed..." << endl;
    Sleep(3000);
    return;
}
cout << "BCD_SRL_RigisterReadCallBack succ..." << endl;

/* 持续读取串口信息 */
while (1)
{
    Sleep(10);
    nRet = BCD_SRL_Read(g_hSerial,
                        strRecvInfo,
                        sizeof(strRecvInfo),
                        nRecvLen);

    if (SRL_RET_OK == nRet)
    {
        // 读取串口信息成功后，触发回调
        cout << "Serial recv : " << strRecvInfo << endl;
    }
}

BCD_SRL_DestroyHandle(g_hSerial);

return;
}

```

5. 补充说明

读取注册回调函数在串口读取信息的情况下被触发。如无需要，可以不设置。

6. 版本信息

序号	变更时间	版本信息	变更人员	变更说明
1	2018.01.15	v1.0.0	Chunel	新建
2	2018.03.15	v1.0.3	Chunel	添加了获取 版本信息的接口