JCWLib 开发库使用说明（v0.02）

发布日期 2019-11-28

一、 关于本库

开发环境

本库运行于 windows 操作系统下，目前提供 x86 平台的开发库，后续完善 x64 版本。本库基于 mfc140 版本开发，需要在使用的计算机上安装 mfc140 的运行库。

功能界限：

本库用于成都衍衍视觉科技开发的接触网几何参数测量装置（简称几何装置）通信，用于获取检测结果

二、 常规流程

1、连接设备：创建设备对象，然后通过设置设备 IP 地址、端口号，配置设备的连接参数，最后启动连接；配置参数的过程可逐一调用函数完成，也可以加载配置文件完成；建议使用加载配置方式，参数可动态修改而不需要修改应用程序。

2、获取数据：JCWLib 库使用回调获取检测数据的方式，需要注册数据的回调函数，几何装置测量到数据后，将调用回调函数传递检测结果给应用层，应用层应当及时接收处理数据，回调函数不能阻塞，也不能处理过长时间。

3、断开连接

采集结束后，应当断开数据连接，并且释放设备对象

4、代码流程例程：

```
void fnOnJCWResult(void* ptag, JCWJH& d) //回调函数处理
{
    //TODO  添加应用的处理代码
    Return;
}

void test()
{

    JCWHANDLE        m_hjcw;
    m_hjcw = Jcw_InitInstance();
    Jcw_SetResultCallBack(m_hjcw, null, fnOnJCWResult);
    Jcw_LoadConfig(m_hjcw,   “jcwconfig.ini”);
    if (JCW_OK != Jcw_Start(m_hjcw))
    {
        AfxMessageBox("打开通信失败");
    }
```

```
    //TODO  等待处理完成
    Jcw_Stop(m_hjcw);
    Jcw_UninitInstance(m_hjcw);
    m_hjcw = NULL;

}
```

三、数据结构：

本库支持采集数据功能，同时辅助用户保存接触网定位数据，可辅助于定位分析；

1、 采集数据结构

使用一个结构体存储检测的所有结果。用户筛选自己需要的信息即可。

```
struct JCWJH
{
public:
    short sVersion;        //数据的版本
    short sBagSize;        //数据包大小
    UINT64    uiSyncID;        //同步的ID
    double    dTimestamp;          //测量的时间戳（s）
    UINT64    uiFrameNo;           //测量的帧号

    UINT      uiLineNum;           //测量到的导线数量，对应 数组：jcx
    JCXData jcx[JCWJH_MAX_LINE_NUM];//动态几何参数，通过 uiLineNum 表示有效内容数量
    JCXData jcxComp[JCWJH_MAX_LINE_NUM];      //静态几何参数;通过 uiLineCompsateNum
表示有效内容数量

    POINTF    pntCompLeft;   //左补偿
    POINTF    pntCompRight; //右补偿

    JCWPosi posi;              //是否定位点
    JCWType lt;                    //导线的类型
    UINT      uiLineCompsateNum;      //补偿后有效的导线数量，对应数组:jcxComp
};
```

2、 辅助定位数据

辅助定位信息有 3 个基础对象，线路信息 JCP_MesStart；支柱信息 JCP_PoleName；距离信息 JCP_Distance。

1）、启动检测时

记录起始的线路信息，包括线路名称，上行线或下行线，公里标变化方向，起始支柱位置（站区、锚段、支柱号）；

2）、检测过程中

只要发生距离变化，就应当设置一次距离信息；距离信息包含：当前公里标、当前累加的距离，当前车速，当前的脉冲 ID（如果时脉冲编码器，记录脉冲编号累加和，如果时距离传感器，记录累加距离和）；如果发生了支柱号码变化，或者检测系统有了定位信息时，也应当设置支柱信息

由于担心逻辑发生混乱，建立只要发生距离变化（车辆移动），就设置支柱信息、距离信息。这样 JCWLib 库就会记录信息。

```
typedef struct
{
    double          dKiloMark;          //公里标，单位km
    UINT64          uiDistance;         //累计的距离，单位1mm，只要车动了，就累加距离
    UINT64          uiPulseID;      //脉冲iD，使用脉冲编码器的统计编码信息，或者其他统计的标识距离的信号
    float       fSpeed;
}JCP_Distance;      //检测距离信息
typedef struct
{
    char        strStation[MAX_POSITION_NAME_LENGHT];     //站区名
    char        strMaoName[MAX_POSITION_NAME_LENGHT]; //锚段名
    char        strPoleName[MAX_POSITION_NAME_LENGHT]; //支柱名
    int             iPoleMark;          //定位标志，是否支柱
}JCP_PoleName;      //支柱名


typedef struct
{
    char strLineName[MAX_POSITION_NAME_LENGHT];             //线路名称，比如xxx一号线
    bool bLineDirection; //线路方向      上行线、下行线
    bool bKiloMarkInc;          //公里标是否递增; true =  递增；   false =  递减

    JCP_PoleName pole_start;        //起始的位置
    JCP_Distance    distance_start; //起始的位置
}JCP_MesStart;                          //检测的线路信息
```

四、函数说明

一）、检测函数说明

头文件 JCWMes.h

```
//获取开发库版本号
const char*     Jcw_GetLibVersionStr();
```

说明：或者开发库的版本信息，以文本方式呈现；编码使用 GB2312

返回值：返回开发库（动态库）的版本信息

```
//功能：构造一个实例
JCWHANDLE Jcw_InitInstance();
```
说明：创建一个实例，并且返回实例的句柄；用户可用该句柄操作设备的控制

```
//功能：释放一个实例
JCW_RET Jcw_UninitInstance(JCWHANDLE h);

//写入配置文件
JCW_RET Jcw_SaveConfig(JCWHANDLE h, const char* strCfgFile);

//读取一份配置
JCW_RET Jcw_LoadConfig(JCWHANDLE h, const char* strCfgFile);

//功能：设置布置参数，如：轨距
JCW_RET Jcw_SetDevArrangeParam(JCWHANDLE h, JCDevSetupParam p);

//功能：设置自定义数据，一般不要调用此接口，此接口用于设置特殊的参数
//iDataTypeID 数据的类型标志值
//pDatas 数据的指针
//uiDataSize 数据的大小
JCW_RET Jcw_SetCustomData(JCWHANDLE h, JMDEVID id, int iDataTypeID, const void* pDatas,
UINT uiDataSize);


//功能：获取自定义数据，一般不要调用此接口，此接口用于设置特殊的参数
//iDataTypeID 数据的类型标志值
//pBuff 用于存储数据的指针
//uiBuffSize 缓存块的内存大小
//uiDataSize 数据的大小
JCW_RET Jcw_GetCustomData(JCWHANDLE h, JMDEVID id, int iDataTypeID, const void* pBuff, IN
UINT uiBuffSize, OUT UINT& uiDataSize);

//功能：设置设备连接地址信息
//id            设备ID
//strDevIP      设备的IP地址
//uiDevPort         设备的通信端口；非特殊情况，填写0，表示使用默认端口
JCW_RET Jcw_SetDevAddress(JCWHANDLE h, JMDEVID id, const char* strDevIP, UINT uiDevPort);
```
说明：设置设备的网络地址；

JCWHANDLE h ：设备句柄

JMDEVID id，设备的功能编号

const char* strDevIP ：设备的网络IP，使用GB2312编码

UINT uiDevPort: 设备的网络通信端口；默认值为6771；如果用户填0，则使用默认值6771，如果指定

了其他具体端口，则使用指定端口

```
//功能：获取已经设置的设备地址列表
//idArray        设备ID的缓冲区
//uiBuffNum          缓冲区的数量，数组大小；可理解为 uiBuffNum = sizeof(idArray) / sizeof(JMDEVID)
//uiDevNumber 已设置的设备数量
JCW_RET Jcw_GetDevIDList(JCWHANDLE h, JMDEVID* idArray, IN UINT uiBuffNum, OUT UINT& uiDevNumber);


//功能：获取设备连接地址信息
//id             设备ID
//strDevIP       设备的IP地址
//uiDevPort          设备的通信端口；非特殊情况，填写0，表示使用默认端口
JCW_RET Jcw_GetDevAddress(JCWHANDLE h, JMDEVID id, OUT char* strDevIPBuff, OUT UINT& uiDevPort);


//回调函数的定义
typedef void(*Jcw_fnJCWResultCallBack)(const void*, const JCWJH&);

//功能：设置检测结果数据回调的接口函数指针
JCW_RET Jcw_SetResultCallBack(JCWHANDLE h, const void* pTag, Jcw_fnJCWResultCallBack fnCB);

//功能：设置单一设备的检测数据的回调
JCW_RET Jcw_SetDevResultCallBack(JCWHANDLE h, JMDEVID id, const void* pTag, Jcw_fnJCWResultCallBack fnCB);


//点云回调函数
//UINT64  系统同步id，或者帧号
//UINT           单帧图像的第n根导线
//POINTF*        点云的空间点
//UINT           2d点的数量
typedef void(*Jcw_fnPointCloudCallBack)(const void*, UINT64, UINT, const POINTF*, const UINT);

//功能：设置设备的点云回调，用于显示及二次处理
JCW_RET Jcw_SetDevPointCloudCallBack(JCWHANDLE h, const void* pTag, Jcw_fnPointCloudCallBack fnCB);

//功能：启动数据接收工作，设置参数完毕后调用本接口
JCW_RET Jcw_Start(JCWHANDLE h);
```

```
//功能：停止数据接收工作
JCW_RET Jcw_Stop(JCWHANDLE h);

    enum
    {
        JCW_DEVSTATE_NOWORK = -1,          //未工作
        JCW_DEVSTATE_DISCONNECT = 0,       //未连接
        JCW_DEVSTATE_WORKING = 1,          //正常工作状态
    };

//功能：返回设备的连接状态，状态值参数以上枚举变量值
//iState = JCW_DEVSTATE_NOWORK 。。。
JCW_RET Jcw_GetDevState(JCWHANDLE h, JMDEVID id, OUT int& iState);

//功能：获取设备的测量帧率
JCW_RET Jcw_GetDevFPS(JCWHANDLE h, JMDEVID id, OUT float& fFPS);

//功能：获取设备连接的时长，单位秒(s)
JCW_RET Jcw_GetDevConnTime(JCWHANDLE h, JMDEVID id, OUT UINT& uiTimeSec);

//功能：设置曝光时间，单位微秒(us)
JCW_RET Jcw_SetExposureTime(JCWHANDLE h, JMDEVID id, const UINT uiExpoTime);

//功能：获取曝光时间，单位微秒(us)
JCW_RET Jcw_GetExposureTime(JCWHANDLE h, JMDEVID id, OUT UINT& uiExpoTime);

//功能：设置相机采集的高度
//uiOffsetY  相机的y方向偏移量
//uiHeight  相机采集图像的高度
JCW_RET Jcw_SetCamHei(JCWHANDLE h, JMDEVID id, const UINT uiOffsetY, const    UINT
uiHeight);

//功能：获取相机的高度
//uiOffsetY  相机的y方向偏移量
//uiHeight  相机采集图像的高度
JCW_RET Jcw_GetCamHei(JCWHANDLE h, JMDEVID id, OUT UINT& uiOffSetY, OUT UINT&
uiHeight);

//功能：设置安装的参数
//参数值为欧拉矩阵
JCW_RET Jcw_SetDevSetupParam(JCWHANDLE h, JMDEVID id, const double dXA, const double dXB,
const double dXC, const double dYA, const double dYB, const double dYC);
```

```
//功能：获取设备安装的参数
//参数值为欧拉矩阵
JCW_RET Jcw_GetDevSetupParam(JCWHANDLE h, JMDEVID id, OUT double& dXA, OUT double&
dXB, OUT double& dXC, OUT double& dYA, OUT double& dYB, OUT double& dYC);

//功能：设置安装的倾斜角
//dAngle 要修正的角度
//dXOffset 水平值的修正值；计算方式 结果.x = 检测.x + dXOffset
//dYOffset 垂直值得修正量  计算方式 结果.y = 检测.y + dYOffset
JCW_RET Jcw_SetDevSetupAngle(JCWHANDLE h, JMDEVID id, const double dAngle, const double
dXOffset, const double dYOffset);


//设置设备的同步ID，为客户程序关联数据同步
//uiSyncID 用户指定的ID，在之后的所有数据中，都将带有此数值的syncid，直到重新调用更新数值
JCW_RET Jcw_SetSyncID(JCWHANDLE h, UINT64 uiSyncID);

//功能：创建一个保存调试包的文件，并且开始保存
//strFilePath 保存的文件名
JCW_RET Jcw_CreateDebugDataFile(JCWHANDLE h, const char* strFilePath);

//功能：停止保存调试数据，并且关闭结调试包的文件
JCW_RET Jcw_CloseDebugDataFile(JCWHANDLE h);

//功能：采集检测目标的轮廓点
//iGrabState 是否采集
JCW_RET Jcw_GrabTargetProfile(JCWHANDLE h, JMDEVID id, int iGrabState);
```

二）、辅助定位函数说明

头文件 JCWPosition.h

注意：*这些功能都是把信息记录到调试文件包中（*.jdbg）的；所以使用这些函数，首先要创建调试文件包（Jcw_CreateDebugDataFile），或者设备已经开始采集 （调用 Jcw_Start），成功创建了调试文件后，使用以下的函数功能，才能把信息记录到文件中。*

1、JCW_RET Jcw_POSI_SetBeginPosition(JCWHANDLE h, const JCP_MesStart& start, std::*string*& strUserMark);

说明：设置启动检测时的线路信息，可同时备注启动的信息，比如现场人员要记录的一些其他信息。

JCWHANDLE h 设备句柄

const JCP_MesStart& start 启动时的线路信息
std::*string*& strUserMark 用户备注信息

2、JCW_RET Jcw_POSI_SetRealTimePosition(JCWHANDLE h, const JCP_PoleName& pole, const JCP_Distance& distance);
说明：设置当前的实时位置

JCWHANDLE h 设备句柄
const JCP_PoleName& pole 当前的支柱位置
const JCP_Distance& distance 当前的距离信息

3、JCW_RET Jcw_POSI_SetRealTimeExtData(JCWHANDLE h, const JC_POSI_EXTTYPE pet, const std::*string*& strData);
说明：设置拓展的信息；支持的拓展信息目前有RFID，用户备注；如果当前读取到RFID信息，可以设置提供，以便于后期分析检测定位状态

JCWHANDLE h 设备句柄
const JC_POSI_EXTTYPE pet 拓展信息的类型
const std::*string*& strData 拓展信息的内容

4、JCW_RET Jcw_POSI_SetEndPosition(JCWHANDLE h, const JCP_PoleName& pole, const JCP_Distance& distance, std::*string*& strUserMark);
说明：设置检测结束时候的信息
由于有时候结束时候，检测的定位并不准确，需要人工确认最后的位置状态，可记录此信息，这样后期定位分析便可知起始、停止的位置。

JCWHANDLE h 设备句柄
JCP_PoleName& pole 停车时的支柱位置
JCP_Distance& distance 停车时的距离信息
std::*string*& strUserMark 停车时，备注的信息