# MCMAG30FDevice.dll使用说明

## 主要内容

模块文件：MCMag30FDevice.dll

命名空间：MCMag30FDevice

依赖库：CyUSB.dll

设备驱动程序：cyusb3.inf、cyusb3.cat、cyusb3.sys

该模块包含内容如下：

### 1、基本数据结构

/// <summary>

/// 设备状态

/// </summary>

public enum DeviceStatus

{

UNKNOWN = 0, // 未知状态

DISCONNECTED = 1, // 未连接

UNAVAILABLE = 2, // 不可用

CONNECTED = 3, // 连接可用

RUNNING = 4, // 正在采集数据

}

/// <summary>

/// AC/DC 模式

/// </summary>

public enum ADCMode

{

UNKNOWN = 0, // 未知状态

AC = 1, // AC

DC = 2, // DC

}

/// <summary>

/// 通道模式

/// </summary>

public enum ChanMode

{

UNKNOWN = 0, // 未知状态

SINGLE\_END = 1, // Single-end

DIFFERENTIAL = 2, // Differential

}

/// <summary>

/// 采样频率

/// </summary>

public enum SamplingFreq

{

UNKNOWN = 0, // 未知状态

FREQ\_100 = 1, // 100Hz

FREQ\_1000 = 2, // 1000Hz

FREQ\_10000 = 3, // 10000Hz

}

/// <summary>

/// 采样模式

/// </summary>

public enum SamplingMode

{

UNKNOWN = 0, // 未知状态

SINGLE = 1, // 单次采集

CONTINUOUS = 2 // 连续采集

}

/// <summary>

/// 采集到的设备数据

/// </summary>

public class DeviceData

{

public double[] dChannelDatas;

}

/// <summary>

/// 错误码

/// </summary>

public enum ErrorCode

{

SUCCESS = 0, // 成功

FAILURE = 1, // 一般错误

TRANSMISSION\_ERROR = 2, // 传输错误

}

### 2、核心功能类

/// <summary>

/// 多通道磁场数据采集设备管理器

/// </summary>

public class MultiChanMagCollectDeviceManager

{

/// <summary>

/// 设备AC/DC模式改变事件

/// </summary>

public event DeviceADCModeChangedEventHandler DeviceADCModeChangedEvent;

/// <summary>

/// 设备通道模式改变事件

/// </summary>

public event DeviceChanModelChangedEventHandler DeviceChanModelChangedEvent;

/// <summary>

/// 设备接收到采集数据的事件

/// </summary>

public event DeviceDataReceivedEventHandler DeviceDataReceivedEvent;

/// <summary>

/// 设备发生异常的事件

/// </summary>

public event DeviceExceptionHappenedEventHandler DeviceExceptionHappenedEvent;

/// <summary>

/// 设备采样频率改变事件

/// </summary>

public event DeviceSamplingFreqEventHandler DeviceSamplingFreqEvent;

/// <summary>

/// 设备状态改变事件

/// </summary>

public event DeviceStatusChangedEventHandler DeviceStatusChangedEvent;

/// <summary>

/// 设备AC/DC模式改变事件处理器

/// </summary>

/// <param name="nDeviceID">设备ID</param>

/// <param name="eADCMode">AC/DC模式</param>

public delegate void DeviceADCModeChangedEventHandler(uint nDeviceID, ADCMode eADCMode);

/// <summary>

/// 设备通道模式改变事件处理器

/// </summary>

/// <param name="nDeviceID">设备ID</param>

/// <param name="eChanMode">通道模式</param>

public delegate void DeviceChanModelChangedEventHandler(uint nDeviceID, ChanMode eChanMode);

/// <summary>

/// 接收到设备采集数据的事件处理器

/// </summary>

/// <param name="nDeviceID">设备ID</param>

/// <param name="sDeviceData">设备数据数组</param>

public delegate void DeviceDataReceivedEventHandler(uint nDeviceID, ref DeviceData[] sDeviceData);

/// <summary>

/// 设备异常事件处理器

/// </summary>

/// <param name="nDeviceID">设备ID</param>

/// <param name="e">异常对象</param>

public delegate void DeviceExceptionHappenedEventHandler(uint nDeviceID, *Exception* e);

/// <summary>

/// 设备采样频率改变事件处理器

/// </summary>

/// <param name="nDeviceID">设备ID</param>

/// <param name="eSamplingFreq">采样频率</param>

public delegate void DeviceSamplingFreqEventHandler(uint nDeviceID, SamplingFreq eSamplingFreq);

/// <summary>

/// 设备状态改变事件处理器

/// </summary>

/// <param name="nDeviceID">设备ID</param>

/// <param name="eDeviceStatus">设备状态</param>

public delegate void DeviceStatusChangedEventHandler(uint nDeviceID, DeviceStatus eDeviceStatus);

/// <summary>

/// 获取唯一实例

/// </summary>

/// <returns>磁场设备管理器的唯一实例</returns>

public static MultiChanMagCollectDeviceManager GetInstance();

/// <summary>

/// 获取所有已连接的设备

/// </summary>

/// <returns>所有已连接设备的ID与设备的Dictionary</returns>

public *Dictionary*<uint, MultiChanMagCollectDevice> GetAllDevices();

/// <summary>

/// 返回设备状态

/// </summary>

/// <param name="nDeviceID">设备ID，由VID和PID构成</param>

/// <returns>设备状态</returns>

public DeviceStatus GetDeviceStatus(uint nDeviceID);

/// <summary>

/// 获取所有使用的设备

/// </summary>

/// <returns>所有使用的设备ID与设备对象的Dictionary</returns>

public *Dictionary*<uint, MultiChanMagCollectDevice> GetInUseDevices();

/// <summary>

/// 设置哪些设备是使用的

/// </summary>

/// <param name="sInUseDevices">使用的设备ID与设备的Dictionary</param>

public void SetInUseDevices(*Dictionary*<uint, MultiChanMagCollectDevice> sInUseDevices);

/// <summary>

/// 设置所有使用设备的AC/DC模式

/// </summary>

/// <param name="eADCMode">AC/DC模式</param>

/// <returns>返回设置错误码</returns>

public ErrorCode SetADCMode(ADCMode eADCMode);

/// <summary>

/// 设置所有使用设备的通道模式

/// </summary>

/// <param name="eChanMode">通道模式</param>

/// <returns>返回设置错误码</returns>

public ErrorCode SetChanMode(ChanMode eChanMode);

/// <summary>

/// 设置所有使用设备的采样频率

/// </summary>

/// <param name="eSamplingFreq">采样频率</param>

/// <returns>返回设置错误码</returns>

public ErrorCode SetSamplingFreq(SamplingFreq eSamplingFreq);

/// <summary>

/// 启动采集线程，开始采集数据

/// </summary>

/// <returns>返回错误码</returns>

public ErrorCode StartAcquireData();

/// <summary>

/// 停止采集线程，停止采集数据

/// </summary>

/// <returns>返回错误码</returns>

public ErrorCode StopAcquireData();

}

/// <summary>

/// 多通道磁场数据采集设备

/// </summary>

public class MultiChanMagCollectDevice

{

/// <summary>

/// 构造函数

/// </summary>

/// <param name="usbDevice">USB设备对象</param>

/// <param name="nDeviceID">设备ID</param>

public MultiChanMagCollectDevice(*CyUSBDevice* usbDevice, uint nDeviceID);

/// <summary>

/// 接受到采集数据事件

/// </summary>

public event DataReceivedEventHandler DataReceivedEvent;

/// <summary>

/// 异常事件

/// </summary>

public event ExceptionEventHandler ExceptionEvent;

/// <summary>

/// 接受到采集数据事件处理器

/// </summary>

/// <param name="nDeviceID">设备ID</param>

/// <param name="sDeviceData">接收到的数据</param>

public delegate void DataReceivedEventHandler(uint nDeviceID, ref DeviceData[] sDeviceData);

/// <summary>

/// 异常事件处理器

/// </summary>

/// <param name="nDeviceID">设备ID</param>

/// <param name="e">异常</param>

public delegate void ExceptionEventHandler(uint nDeviceID, *Exception* e);

/// <summary>

/// 主要处理一些开始采集之前的设置

/// </summary>

/// <returns>设置是否成功</returns>

public bool BeforeStartAcquireData();

/// <summary>

/// 启动采集线程

/// </summary>

public void StartAcquireThread();

/// <summary>

/// 停止采集数据

/// </summary>

/// <returns>是否成功</returns>

public bool StopAcquireData();

/// <summary>

/// 主要处理停止采集之后的工作

/// </summary>

/// <returns>处理是否成功</returns>

public bool AfterStopAcquireData();

/// <summary>

/// 获取AC/DC模式

/// </summary>

/// <returns>AC/DC模式</returns>

public ADCMode GetADCMode();

/// <summary>

/// 设置AC/DC模式

/// </summary>

/// <param name="eADCMode">AC/DC模式</param>

/// <returns>返回设置错误码</returns>

public ErrorCode SetADCMode(ADCMode eADCMode);

/// <summary>

/// 获取通道模式

/// </summary>

/// <returns>通道模式</returns>

public ChanMode GetChanMode();

/// <summary>

/// 设置通道模式

/// </summary>

/// <param name="eChanMode">通道模式</param>

/// <returns>返回设置错误码</returns>

public ErrorCode SetChanMode(ChanMode eChanMode);

/// <summary>

/// 获取采样频率

/// </summary>

/// <returns>采样频率</returns>

public SamplingFreq GetSamplingFreq();

/// <summary>

/// 设置采样频率

/// </summary>

/// <param name="eSamplingFreq">采样频率</param>

/// <returns>是否设置成功</returns>

public ErrorCode SetSamplingFreq(SamplingFreq eSamplingFreq);

/// <summary>

/// 返回设备状态

/// </summary>

/// <returns>设备状态</returns>

public DeviceStatus GetDeviceStatus();

/// <summary>

/// 设备是否可用

/// </summary>

/// <returns>是否可用</returns>

public bool IsAvailable();

/// <summary>

/// 返回设备是否正在采集数据的状态

/// </summary>

/// <returns>是否正在采集数据</returns>

public bool IsRunning();

}

## 使用方式

该模块的使用方式较为简单。您不需要自己构造设备对象MultiChanMagCollectDevice，仅需要构造一个全局唯一的多通道磁场数据采集设备管理器实例即可——调用MultiChanMagCollectDeviceManager.GetInstance()，该管理器将会自动监控连接到计算机的所有设备，您可以通过调用GetAllDevices()得到当前连接到计算机的所有设备ID与设备对象的Dictionary。如果在使用该模块的过程中设备存在插拔的情况，通过注册DeviceStatusChangedEvent的事件处理器也可以得到相应的通知。

但是，当前连接的所有设备默认情况下通过设备管理器是不能直接使用的。出于灵活性考虑，您可以通过调用SetInUseDevices()函数将设备设置为可以使用。这样通过设备管理器调用的各种关于设备的函数就是对所有可使用设备可用的。

使用该模块的一般调用流程如下：

