

ATK-UT70DLL 使用说明文档

UT70USB 测试仪上位机二次开发库

使用文档

正点原子
广州市星翼电子科技有限公司

修订历史

版本	日期	原因
V1.0	2024/02/02	第一次发布

目 录

1 说明	1
2 工程简介	1
2.1 开发环境	1
3 使用说明	1
3.1 实例化类	1
3.2 设备连接状态	1
3.3 设备连接或者断开	1
3.4 Pin 包	2
3.5 获取设备版本信息	2
3.6 设备 UI	2
3.7 帧率设置	4
3.8 重置电量计数统计	5
3.9 电压电流数据上传	5
3.10 快充协议	6
4 源码示例	10
5 其他	14

1 说明

ATK-UT70DLL.dll 是配套正点原子产品 UT70 数控电源使用的，用户可以利用动态库来自定义开发 UT70 数控电源配套上位机。本文档使用示例为 C# 工程的示例。本文档中以下出现动态库或者本库都是指 ATK-UT70DLL.dll

2 工程简介

2.1 开发环境

系统: windows10
环境: .net framework
语言: C#
开发工具: Microsoft Visual Studio 2019

图 2.2.1

3 使用说明

3.1 实例化类

```
ATKUT70 ut70 = new ATKUT70(); //初始化 UT70 设备操作类
```

3.2 设备连接状态

```
/// <summary>  
/// 设备的连接状态 true 是连接  
/// </summary>  
public bool ConnState
```

3.3 设备连接或者断开

```
/// <summary>  
/// 连接设备  
/// </summary>
```

```
/// <returns></returns>
public bool ConnDev()

/// <summary>
/// 断开连接
/// </summary>
public void Disconnect()
```

3.4 Pin 包

```
/// <summary>
/// 发送 pin
/// </summary>
/// <param name="val">默认 1 调用无需传入参数</param>
/// <returns></returns>
public bool Pin(byte val=1)
```

3.5 获取设备版本信息

```
/// <summary>
/// 获取设备版本信息
/// </summary>
public bool GetDevVer()

/// <summary>
/// 设备序列号上传
/// byte 设备硬件版本
/// string 设备序列号，设备得唯一标识号
/// </summary>
public event Action<byte, string> DevSNEvent;
```

3.6 设备 UI

```
/// <summary>
/// 获取设备的UI编号
/// </summary>
public bool GetDevUI()

/// <summary>
/// 设置 切换设备得UI界面
```

```
/// </summary>
/// <param name="devUI"></param>
/// <returns></returns>
public bool SetDevUI (DevUI devUI)
```

```
/// <summary>
/// 设备界面
/// </summary>
public enum DevUI :byte{
    /// <summary>
    /// 基本测量
    /// </summary>
    NORMAL=0,
    /// <summary>
    /// 波形显示
    /// </summary>
    WAVEFORM=1,
    /// <summary>
    /// 数据记录
    /// </summary>
    DATALOG=2,
    /// <summary>
    /// 功能应用
    /// </summary>
    FUNUSE=3,
    /// <summary>
    /// 手动触发
    /// </summary>
    MANUAL=4,
    /// <summary>
    /// 自动检测
    /// </summary>
    AUTO = 5,
    /// <summary>
    /// 监控模式
    /// </summary>
    MONITOR=11,
    /// <summary>
    /// 线阻模式
    /// </summary>
    LINER=18,
    /// <summary>
    /// DASH检测
    /// </summary>
```

```
DASHCHECK=19,  
/// <summary>  
/// EMARKER检测  
/// </summary>  
EMARKERCHECK=20,  
/// <summary>  
/// 设置  
/// </summary>  
SETFACE=29,  
OTHER=0xFF,  
}
```

3.7 帧率设置

```
/// <summary>  
/// 获取设备上传频率信息  
/// </summary>  
public bool GetFrequency()  
  
/// <summary>  
/// 设备帧率信息上传  
/// </summary>  
public event Action<Frequency> FrequencyEvent;  
  
/// <summary>  
/// 设置设备频率  
/// </summary>  
/// <param name="frequency">设备帧率</param>  
public void SetFrequency(Frequency frequency)  
  
/// <summary>  
/// 设备数据上传帧率  
/// </summary>  
public enum Frequency : byte  
{  
    _4SPS,  
    _10SPS,  
    _50SPS,  
    _200SPS,  
    _500SPS,  
    _1000SPS,  
  
    Err,
```

```
}
```

3.8 重置电量计数统计

```
/// <summary>
/// 重置电量计数
/// </summary>
public void ResteAH()
```

3.9 电压电流数据上传

```
/// <summary>
/// 接收到电压电流数据 事件接口
/// </summary>
public event Action<List<CVDataModel>> ReceCVDataEvent;
```

```
/// <summary>
/// 电压电流数据上传模型
/// </summary>
public class CVDataModel : EventArgs
{
    float v;          //电压
    float i;          //电流
    float dp;         //d+
    float dm;         //d-
    float r;          //电阻
    float p;          //功率
    float wh;         //能量
    float ah;         //电量

    /// <summary>
    /// 电压
    /// </summary>
    public float V { get => v; set => v = value; }
    /// <summary>
    /// 电流
    /// </summary>
    public float I { get => i; set => i = value; }
    /// <summary>
    /// D+
```

```
    /// </summary>
    public float Dp { get => dp; set => dp = value; }
    /// <summary>
    /// D-
    /// </summary>
    public float Dm { get => dm; set => dm = value; }
    /// <summary>
    /// 电阻
    /// </summary>
    public float R { get => r; set => r = value; }
    /// <summary>
    /// 功率
    /// </summary>
    public float P { get => p; set => p = value; }
    /// <summary>
    /// 能量
    /// </summary>
    public float Wh { get => wh; set => wh = value; }

    /// <summary>
    /// 电量
    /// </summary>
    public float AH { get => ah; set => ah = value; }
}
```

3.10 快充协议

```
    /// <summary>
    /// 设置 FCP
    /// </summary>
    /// <param name="fcp"></param>
    public void SetProtocol_FCP(FCP fcp)

    /// <summary>
    /// 设置 QC2
    /// </summary>
    /// <param name="qc2"></param>
    public void SetProtocol_QC2(QC2 qc2)

    /// <summary>
    /// 设置 QC3
    /// </summary>
    /// <param name="qc3"></param>
    public void SetProtocol_QC3(QC3 qc3)
```

```
/// <summary>
/// 设置 SCP
/// </summary>
/// <param name="scp"></param>
public void SetProtocol_SCP(SCP scp)
```

```
/// <summary>
/// 设置 PD
/// </summary>
/// <param name="pd"></param>
public void SetProtocol_PD(PD pd)
```

```
/// <summary>
/// 设置 AFC
/// </summary>
/// <param name="afc"></param>
public void SetProtocol_AFC(AFC afc)
```

```
/// <summary>
/// 自动检测快充协议
/// </summary>
public bool AutoDetectionProtocol()
```

```
public enum FCP
{
    FCP_5V,
    FCP_9V,
    FCP_12V,
}
```

```
public enum QC2
{
    QC2_5V,
    QC2_9V,
    QC2_12V,
    QC2_20V,
}
```

```
public enum QC3
{
    QC3_200MVL,
    QC3_200MVR,
}
```

```
public enum SCP
{
    SCP_20MVL,
    SPC_20MVR,
}

public enum PD
{
    PD_5V,
    PD_9V,
    PD_12V,
    PD_15V,
    PD_20V,
    PD_28V,
    PD_PPSL,
    PD_PPSR,
}

public enum AFC
{
    AFC_5V,
    AFC_9V,
    AFC_12V
}

/// <summary>
/// 快充协议上传
/// FastChargeType 协议类型
/// 值
/// </summary>
public event Action<FastChargeType, int> FastChargeEvent;
获取快充是否支持某个电压操作 (int & VoltageParameter.FAST_CHARGE_PROTOCOL_5V) > 0 true
支持, false不支持
public class VoltageParameter
{
    /*
        各种协议所支持的对应电压操作范围

    */
    public const int FAST_CHARGE_PROTOCOL_5V = 0x00000080;
    public const int FAST_CHARGE_PROTOCOL_6V = 0x00000100;
    public const int FAST_CHARGE_PROTOCOL_7V = 0x00000200;
```

```
public const int FAST_CHARGE_PROTOCOL_8V = 0x00000400;
public const int FAST_CHARGE_PROTOCOL_9V = 0x00000800;
public const int FAST_CHARGE_PROTOCOL_10V = 0x00001000;
public const int FAST_CHARGE_PROTOCOL_11V = 0x00002000;
public const int FAST_CHARGE_PROTOCOL_12V = 0x00004000;
public const int FAST_CHARGE_PROTOCOL_15V = 0x00008000;
public const int FAST_CHARGE_PROTOCOL_16V = 0x00010000;
public const int FAST_CHARGE_PROTOCOL_18V = 0x00020000;
public const int FAST_CHARGE_PROTOCOL_20V = 0x00040000;
public const int FAST_CHARGE_PROTOCOL_25V = 0x00080000;
public const int FAST_CHARGE_PROTOCOL_28V = 0x00100000;
public const int FAST_CHARGE_PROTOCOL_30V = 0x00200000;
public const int FAST_CHARGE_PROTOCOL_36V = 0x00400000;
public const int FAST_CHARGE_PROTOCOL_48V = 0x00800000;

/// <summary>
/// QC3 SCP PD 里面的加减对应数据位
/// </summary>
public const int FAST_CHARGE_PROTOCOL_PPSI = 0x01000080;
}

/// <summary>
/// 快充协议说明上传
/// bool  协议是否检测正常，可用，true正常
/// string  协议检测得描述信息
/// </summary>
public event Action<FastChargeType, bool, string> FastChargeDescEvent;

public enum FastChargeType : byte {
    BC1_2 = 0,
    QC2,
    QC3,
    FCP,
    SCP,
    AFC,
    PE1,
    PE2,
    VOOC,
    PD,
    DASH,
    EMarker,
    VFCP,
    APPLE2_4,
    Err = 0xff,
```

```
}
```

4 源码示例

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using ATK_UT70DLL;

namespace ATK_UT70Demo
{
    class Program
    {
        static ATKUT70 ut70 = null; //UT70设备操作类

        static void Main(string[] args)
        {
            ut70 = new ATKUT70(); //初始化 UT70设备操作类

            ut70.PinEvent += OnPinEvent; //pin
            ut70.DevSNEvent += OnDevSNEvent; //设备序列号，硬件版本等
            ut70.DevUIChanageEvent += OnDevUIChanageEvent; //设备的UI界面信息上传
            ut70.FastChargeEvent += OnFastChargeEvent; //快充协议上传
            ut70.FastChargeDescEvent += OnFastChargeDescEvent; //快充协议说明上传
            ut70.FrequencyEvent += OnFrequencyEventEvent; //频率信息
            ut70.ReceCVDataEvent += OnReceCVDataEvent; //电压电流数据接收事件

            ut70.ResteAH(); //重置电量 能量计数 将电量计数清0 重新开始计算
            ut70.ConnDev(); //连接设备

            if (!ut70.ConnState)
            {
                Console.WriteLine("设备未连接");
                Console.ReadKey();
                return;
            }

            //ut70.Pin(0); //pin设备，设备通讯状态
```

```
// ut70.GetDevVer(); //获取设备版本
//ut70.GetDevUI(); //获取设备屏幕UI界面
//ut70.SetDevUI(DevUI.NORMAL); //设置设备的UI界面

//ut70.SetFrequency(Frequency._4SPS); //设置频率
//ut70.GetFrequency(); //获取频率

//ut70.SetProtocol_PD(PD.PD_5V); //设置协议
//ut70.AutoDetectionProtocol(); //自动检测快充协议

//ut70.Disconnect();
Console.ReadKey();
}

/// <summary>
/// 电压电流数据
/// </summary>
/// <param name="cVDatas"></param>
static void OnReceCVDataEvent(List<CVDataModel> cVDatas)
{
    for (int i = 0; i < cVDatas.Count; i++)
    {
        Console.WriteLine("电压: " + cVDatas[i].V);
        Console.WriteLine("\t电流: " + cVDatas[i].I);
        Console.WriteLine("\tDp: " + cVDatas[i].Dp);
        Console.WriteLine("\tDm: " + cVDatas[i].Dm);
        Console.WriteLine("\tR: " + cVDatas[i].R);
        Console.WriteLine("\t电量: " + cVDatas[i].Wh);
        Console.WriteLine("\t能量: " + cVDatas[i].AH);
    }

    Console.WriteLine("");
}

/// <summary>
/// pin
/// </summary>
/// <param name="pin"></param>
static void OnPinEvent(bool pin)
{
    Console.WriteLine("通讯Pin: " + pin);
}

/// <summary>
```

```
/// 设备SN
/// </summary>
static void OnDevSNEvent(byte ver, string devsn)
{
    Console.WriteLine("硬件版本: " + "V" + (ver / 10f).ToString("0.0"));
    Console.WriteLine("设备SN: " + devsn);
}

//设备UI改变后设备回传信息
static void OnDevUIChanageEvent(DevUI devUI)
{
    Console.WriteLine("设备的UI:" + devUI);
}

/// <summary>
/// 快充协议上传
/// </summary>
/// <param name="type"></param>
/// <param name="value"></param>
static void OnFastChargeEvent(FastChargeType type, int value)
{
    Console.WriteLine("协议类型: " + type);
    Console.WriteLine("是否支持 5V " + ((value &
VoltageParameter.FAST_CHARGE_PROTOCOL_5V) > 0));
    Console.WriteLine("是否支持 9V " + ((value &
VoltageParameter.FAST_CHARGE_PROTOCOL_9V) > 0));
    Console.WriteLine("是否支持 12V " + ((value &
VoltageParameter.FAST_CHARGE_PROTOCOL_12V) > 0));
    Console.WriteLine("是否支持 20V " + ((value &
VoltageParameter.FAST_CHARGE_PROTOCOL_20V) > 0));
    /*采用类似方式, 逐个算出当前协议是否支持某个电压操作*/
}

/// <summary>
/// 快充协议说明上传
/// </summary>
/// <param name="type"></param>
/// <param name="success"></param>
/// <param name="info"></param>
static void OnFastChargeDescEvent(FastChargeType type, bool success, string info)
{
    Console.WriteLine("协议类型:" + type);
    Console.WriteLine("协议解析结果: "+success);
    Console.WriteLine("协议信息: "+info);
}
```

```
    }

    ///频率上传
    static void OnFrequencyEventEvent(Frequency frequency)
    {
        Console.WriteLine("频率"+frequency);
    }

    static string ByteToHexStr(byte[] by)
    {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < by.Length; i++)
        {
            sb.Append(by[i].ToString("X2") + " "); //小写
        }
        return sb.ToString();
    }
}
```

5 其他

1、购买地址：

天猫：<https://zhengdianyuanzi.tmall.com>

淘宝：<https://openedv.taobao.com>

2、资料下载

模块资料下载地址：<http://www.openedv.com/docs/index.html>（更新模块地址，找运营要）

3、技术支持

公司网址：www.alientek.com

技术论坛：<http://www.openedv.com/forum.php>

在线教学：www.yuanzige.com

B 站视频：<https://space.bilibili.com/394620890>

传真：020-36773971

电话：020-38271790

