# 连接设备

DevicePDF dev = new DevicePDF();

dev.connectDevice();

dev.DoRead();

# 二. 读数据

var data = dev.Data;

data实例已经包括所有片子中数据.

# 三.写数据

修改dev.Data对应各项的值，然后执行

dev.DoWrite();

# 四. Data格式参考

DeviceData即dev.Data对应各属性项说明

Get为读set为写，只有get为只读，只有set为只写，同时存在为可读可写，定制时也可以酌情修改.

其实大多说明都存在代码注释中。

|  |  |  |
| --- | --- | --- |
| 属性名称 | 说明 | 定制说明 |
| string AlarmDelay { get; } | 15 报警延时 (Alarm Delay)\* 15字节 以秒为单位，5段报警延时时间（每个延时3字节） | 通常这个属性存在是为了后面0-4计算方便 |
| int AlarmDelay0 { get; set; } | 第一段报警延时 | 如果不需要删除即可 |
| int AlarmDelay1 { get; set; } | 2 | 如果不需要删除即可 |
| int AlarmDelay2 { get; set; } | 3 | 如果不需要删除即可 |
| int AlarmDelay3 { get; set; } | 4 | 如果不需要删除即可 |
| int AlarmDelay4 { get; set; } | 5 | 如果不需要删除即可 |
| string AlarmLimits { get; set; } | 报警阀值 (Alarm Limits)\* 10字节 5段报警阀值（每个阈值2字节） | 通常这个属性存在是为了后面0-4计算方便 |
| string AlarmLimits0 { get; set; } | 1 | 如果不需要删除即可 |
| string AlarmLimits1 { get; set; } | 2 | 如果不需要删除即可 |
| string AlarmLimits2 { get; set; } | 3 | 如果不需要删除即可 |
| string AlarmLimits3 { get; set; } | 4 | 如果不需要删除即可 |
| string AlarmLimits4 { get; set; } | 5 | 如果不需要删除即可 |
| int AlarmMark { get; } | 报警标志 1 0:未报警，1：已报警 | 如果不需要删除即可 |
| int AlarmMode { get; set; } | 报警模式 (Alarm Mode)0为不报警 1 1为单次报警(Single Alarm)，2为多次报警( Multi Alarm) | 如果不需要删除即可 |
| string AlarmTimeLimit { get; } | 各区域超温次数限制 超温次数，最大250次 | 通常这个属性存在是为了后面0-4计算方便 |
| int AlarmTimeLimit0 { get; set; } | 1 | 如果不需要删除即可 |
| int AlarmTimeLimit1 { get; set; } | 2 | 如果不需要删除即可 |
| int AlarmTimeLimit2 { get; set; } | 3 | 如果不需要删除即可 |
| int AlarmTimeLimit3 { get; set; } | 4 | 如果不需要删除即可 |
| int AlarmTimeLimit4 { get; set; } | 5 | 如果不需要删除即可 |
| string AlarmType { get; } | 报警方式 (Alarm Type)\* 5 5段报警方式，第1位代表本报警使能，第2位代表这个上限报报警，否则是下限报警。第8位代表累计报警(Cumulative Alarm)  /// 或是单次报警(Single Event)（每个方式1字节） "0b11000001  /// value&0x80>0 则，开启了报警 否则未开启  ///value&0x40>0 则, 开启了上限报警 否则未开启  ///value&0x01>0 则，开启了累计报警 否则，单次报警 | 通常这个属性存在是为了后面0-4计算方便 |
| byte AlarmType0 { get; set; } | 1 | 如果不需要删除即可 |
| byte AlarmType1 { get; set; } | 2 | 如果不需要删除即可 |
| byte AlarmType2 { get; set; } | 3 | 如果不需要删除即可 |
| byte AlarmType3 { get; set; } | 4 | 如果不需要删除即可 |
| byte AlarmType4 { get; set; } | 5 | 如果不需要删除即可 |
| string AllZoneAlartStatus { get; } | 各区域报警状态 5 0代表没有报警，1代表有报警 | 为数组输出方便而存在 |
| int[] AllZoneAlartStatusArray { get; } | 五段数组值 | 如果不需要删除即可 |
| string AllZoneOverTempTimesStat { get; } | 各区域超温次数统计 10 | 各区域超温次数统计 10 |
| int[] AllZoneOverTempTimesStatArray { get; } | 五段数组值 | 如果不需要删除即可 |
| string AllZoneOverTempTimeStat { get; } | 各区域超温时间统计  /// 时间以秒为单位 | 如果不需要删除即可 |
| int[] AllZoneOverTempTimeStatArray { get; } | 五段数组值 | 如果不需要删除即可 |
| string AvgTemp { get; } | 平均温度 | 如果不需要删除即可 |
| int Battery { get; } | 电量 | 如果不需要删除即可 |
| string[] Calcultion { get; } | 校准值(Calibration) 6 三段校准,第一位是个位，第二位是小数位 0.1;1.0;2.0  /// 返回三组值  笔误命名错误，担心发生兼容问题，保持错误 | 如果不需要删除即可 |
| string CRC { get; set; } | CRC 2 全部数据的CRC，保证设置的东西是正确的。 | 如果不需要删除即可 |
| string Description { get; set; } | 27 任务摘要(Description) 128 | 如果不需要删除即可 |
| TempSen.DevicePDF dev { get; set; } | 逻辑变量和输出无关 | 如果不需要删除即可 |
| string DevModel { get; } | 设备型号(Model) hex string .  Hex指十六进制 | 如果不需要删除即可 |
| string DevNo { get; } | 串号（Serial No.） 10 10byte，可以是字母（大写A-Z）或数字（0-9） | 如果不需要删除即可 |
| int FirmwareVersion { get; set; } | 固件版本 1 1-256 | 如果不需要删除即可 |
| string FirstAlarmTime { get; } | 五段首次报警时间 | 为数组输出方便而存在 |
| DateTime[] FirstAlarmTimeArray { get; } | 数组输出 | 如果不需要删除即可 |
| string GetTempValue(int index); | 获取第n个点的温度值 | 此为方便，建议保留，有其他地方依赖 |
| string HardWareAddr { get; } | 硬件地址 4 MAC地址 192.168.105.188 | 如果不需要删除即可 |
| int LogCount { get; } | 记录点数 3 | 如果不需要删除即可 |
| int LogCycle { get; set; } | 记录周期(Log Cycle) 2 h 以小时为单位,最大2730天 | 如果不需要删除即可 |
| DateTime LogEndTime { get; } | 结束时间 6 | 如果不需要删除即可 |
| int LogInterval { get; set; } | 记录间隔(Log Interval) 2 s 以秒为单位,最大18小时 | 如果不需要删除即可 |
| DateTime LogNowTime { get; } | 当前时间 6 | 如果不需要删除即可 |
| DateTime LogStartTime { get; } | 开始时间(第一个记录点) 6 | 如果不需要删除即可 |
| string MarkTime { get; } | 中间值 | 如果不需要删除即可 |
| string[] MarkTimeArray { get; } | Mark对应输出 | 如果不需要删除即可 |
| int MarkTimeCount { get; } | Mark次数 | 如果不需要删除即可 |
| string MaxTemp { get; } | 最高温 | 如果不需要删除即可 |
| DateTime MaxTempTime { get; } | 对应时间 | 如果不需要删除即可 |
| string MinTemp { get; } | 最低温 | 如果不需要删除即可 |
| DateTime MinTempTime { get; } | 对应时间 | 如果不需要删除即可 |
| string MKT { get; } | Mkt | 如果不需要删除即可 |
| string MonitorLocation { get; set; } |  | 如果不需要删除即可 |
| string PackageNumber { get; set; } |  | 如果不需要删除即可 |
| string PalletNumber { get; set; } |  | 如果不需要删除即可 |
| int ProtocolVersion { get; set; } |  | 如果不需要删除即可 |
| int RunMode { get; set; } | 运行模式（Run Mode） 1 开始/停止；循环 "0代表开始/停止 1代表循环" | 如果不需要删除即可 |
| int RunStatus { get; } | 运行状态(RunStatus) 1 0:未设定，1：已设定， 2：记录中。3：已停止 | 如果不需要删除即可 |
| string StartConditionTemp { get; set; } | 到温启动条件 2 暂无更多说明 | 如果不需要删除即可 |
| DateTime StartConditionTime { get; set; } | 定时启动时间 6 定时时间，一定要比设定时间晚。年/月/日/时/分/秒 | 如果不需要删除即可 |
| int StartDelay { get; set; } | 启动延时时间 (Start Delay) 2 m 以分钟为单位，最大45天 | 如果不需要删除即可 |
| string StartMode { get; set; } | 启动方式（Start Mode） 1 0xF8 = 按键启动； 0x8F = 定时启动； 0x67 = 到温启动 | 如果不需要删除即可 |
| string STDEV { get; } |  | 如果不需要删除即可 |
| int TemperatureUnit { get; set; } | 20 温度格式（Temperature Unit） 1 "0: C  /// 1: F" | 如果不需要删除即可 |
| System.Collections.Generic.List<string> TempList { get; } | 各温度点值的列表，长度为温度点个数 | 如果不需要删除即可 |
| System.Collections.Generic.List<string> TempListWithTime { get; } | 带时间的温度点值和时间列表 | 如果不需要删除即可 |
| byte TimeZone { get; set; } | 19 时区(Time Zone) 1 设置地跟GMT的时时间，如中国（+80） | 如果不需要删除即可 |
| string TripNo { get; set; } | 运输号,可以是字母（大写A-Z）或数字（0-9） | 如果不需要删除即可 |
| DateTime WriteTime { get; set; } | 18 设置时间 (Write Time) 6 年/月/日/时/分/秒  /// 该时间输入输出都为utc时间！！切切 | 如果不需要删除即可 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 五. 数据字典



# 六. Demo代码

static void Main(string[] args)

{

//////////////////////////////////////////

DevicePDF dev = new DevicePDF();

dev.connectDevice();

dev.DoRead();

//dev.Data.Description = "tttttest";

//dev.Data.Operator = "zhurenjie did";

var data = dev.Data;

//通过反射，在屏幕上打印所有数据

Type t = data.GetType();

PropertyInfo[] infos = t.GetProperties();

foreach (PropertyInfo info in infos)

{

var value = t.GetProperty(info.Name).GetValue(dev.Data,null);

//t.GetProperty(info.Name).SetValue(dev.Data, 111);

Console.WriteLine(info.Name+":"+value);

if (value.GetType() == typeof(string[]))

{

string[] vs = (string[])value;

for (int i = 0; i < vs.Length; i++)

Console.WriteLine(info.Name + i.ToString()+":" + vs[i]);

}

if (value.GetType() == typeof(List<string>))

{

Console.WriteLine("press enter to show List:");

Console.ReadLine();

List<string> vs = (List<string>)value;

for (int i = 0; i < vs.Count; i++)

Console.WriteLine(info.Name + i.ToString() + ":" + vs[i]);

}

}

//测试写入

Console.WriteLine("press enter to Write:");

Console.ReadLine();

dev.Data.Description = "20110916";

dev.Data.LogInterval = 1;

dev.Data.LogCycle = 1;

dev.Data.RunMode = 1;

dev.Data.StartMode = "F8";

dev.Data.TripNo = "xxxxxxx";

dev.Data.Description = "abcdefghijklmnopqrstuvwxyz";

dev.DoWrite();

Console.WriteLine("press enter to exit:");

Console.ReadLine();

}