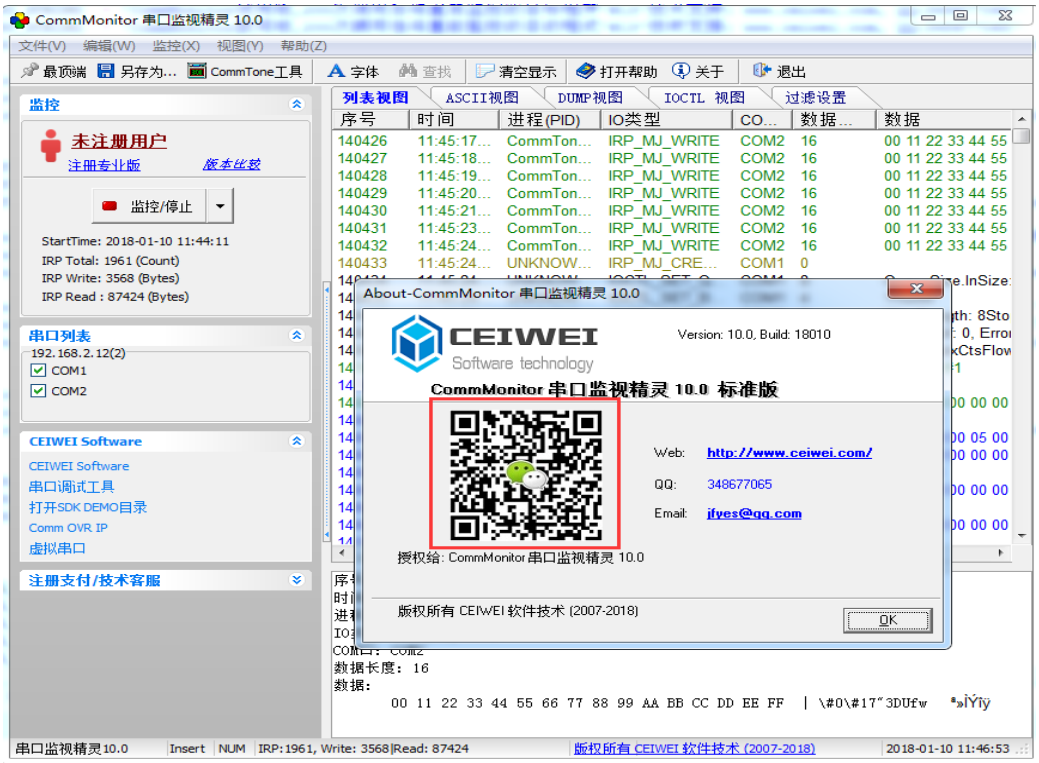
目录



CommMonitor 用户手册

Ver: 10.0 2018-01-15

CommMonitor 串行端口监视精灵是用于 RS232 / RS422 / RS485 端口监控的专业强大的系统实用程序软件。CommMonitor 监视显示，记录和分析系统中的所有串行端口活动。这是追踪应用程序或驱动程序开发，串行设备测试和优化等过程中可能出现的问题的理想方法。还提供过滤、搜索、数据导出和强大的数据拦截功能，可以将指定端口的数据流、控制流信息拦截并保存下来，供分析之用。如察看端口状态的变化（波特率、数据位、校验位、停止位），拦截上行、下行的数据，处理速度快，拦截效率高，并可以以十六进制、ASCII 字符形式显示，全面支持 Unicode 。

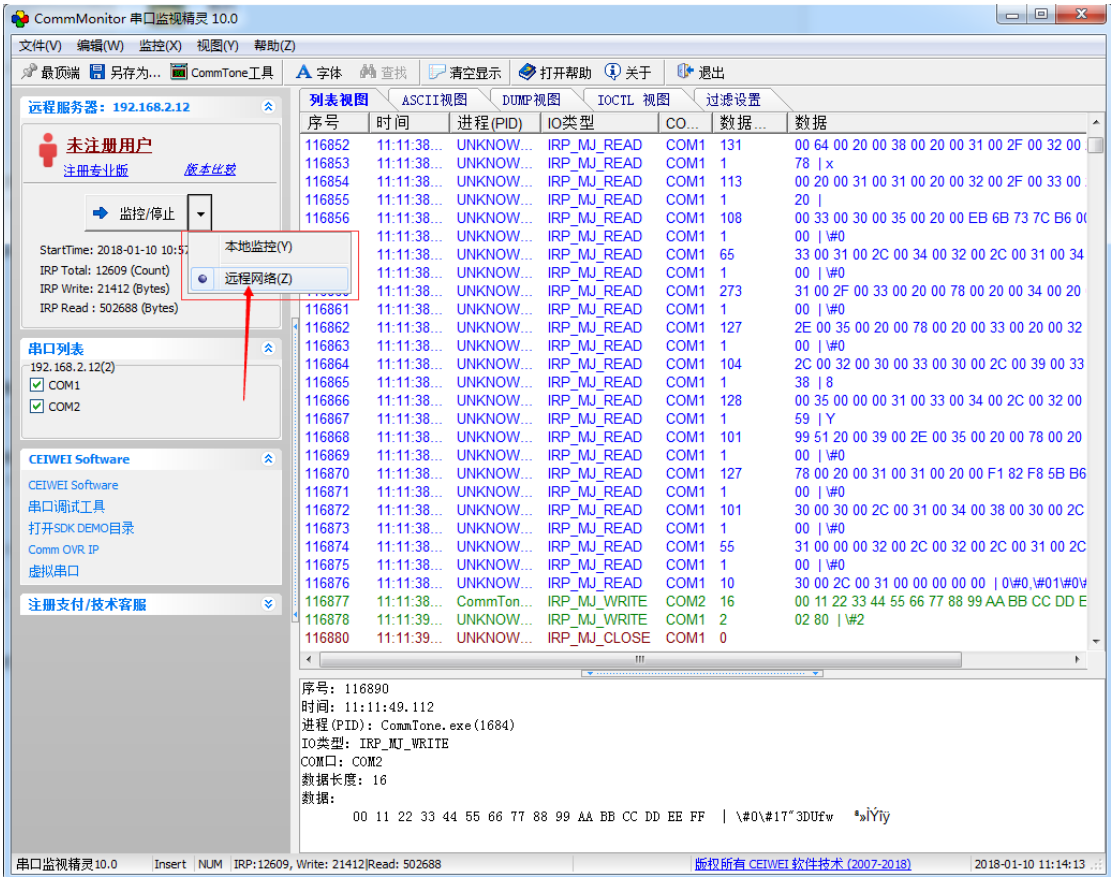
新增支持 Socket TCP/UDP 网络转发功能。

支持 Windows 系统版本：WinXP、Win2003、WinVista、Win7、Win2008、Win8、Win2012、Win2016、Win10，32/64 位系统，驱动程序已签名，完全支持 64 位 Windows 系统。

语言支持：简体中文、繁体中文、英文三种语言。

使用帮助

开启监控



如图：打开【监控/停止】

【本地监控】是指监控当前安装在此电脑主机上的串口；

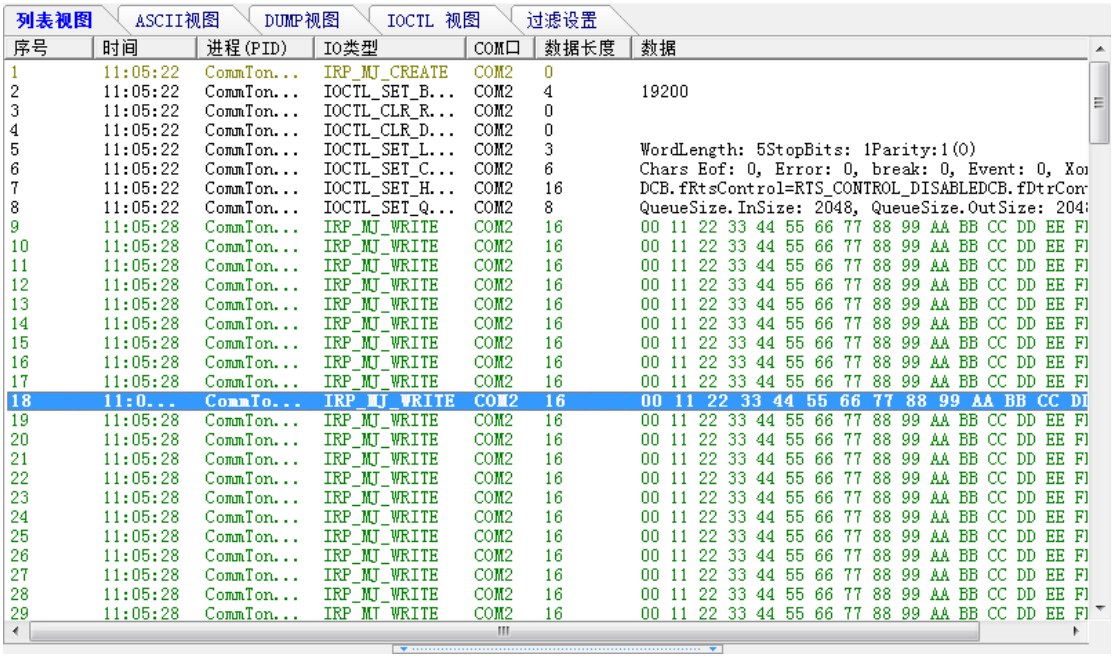
【远程网络】是指连接远程电脑主机上的监控服务，远程电脑上必须安装 CommMonitor10 并开启服务，默认安装是开启的。

备注:

开启监控时，会自动扫描系统有无串口设备(包括虚拟、USB 转串口、标准串口)，如果没有串口设备，它会自动提示没有串口设备，而放弃当前启动监控。

当系统有可以用的串口时，则尝试加载串口内核驱动绑定串口设备进行监控。

视图说明



1、列表视图：

当前所有 IRP 包 数据列表视图，数据为 HEX/ASCII 码显示。表头包括: IRP_MJ FUNCTION, 目标进程(PID),COM 号，数据长度，数据。

2、ASCII 视图：

当前 IRP_MJ_WRITE/IRP_MJ_READ 数据视图，分为 ASCII 码显示。数据包括: 目标进程,COM 号，MJ 名称，数据长度，数据。

3、DUMP 视图:

当前 IRP_MJ_WRITE/IRP_MJ_READ 数据视图，分为 HEX/ASCII 码显示。数据包括: 目标进程,COM 号，MJ 名称，数据长度，数据。

4、IOCTL 视图

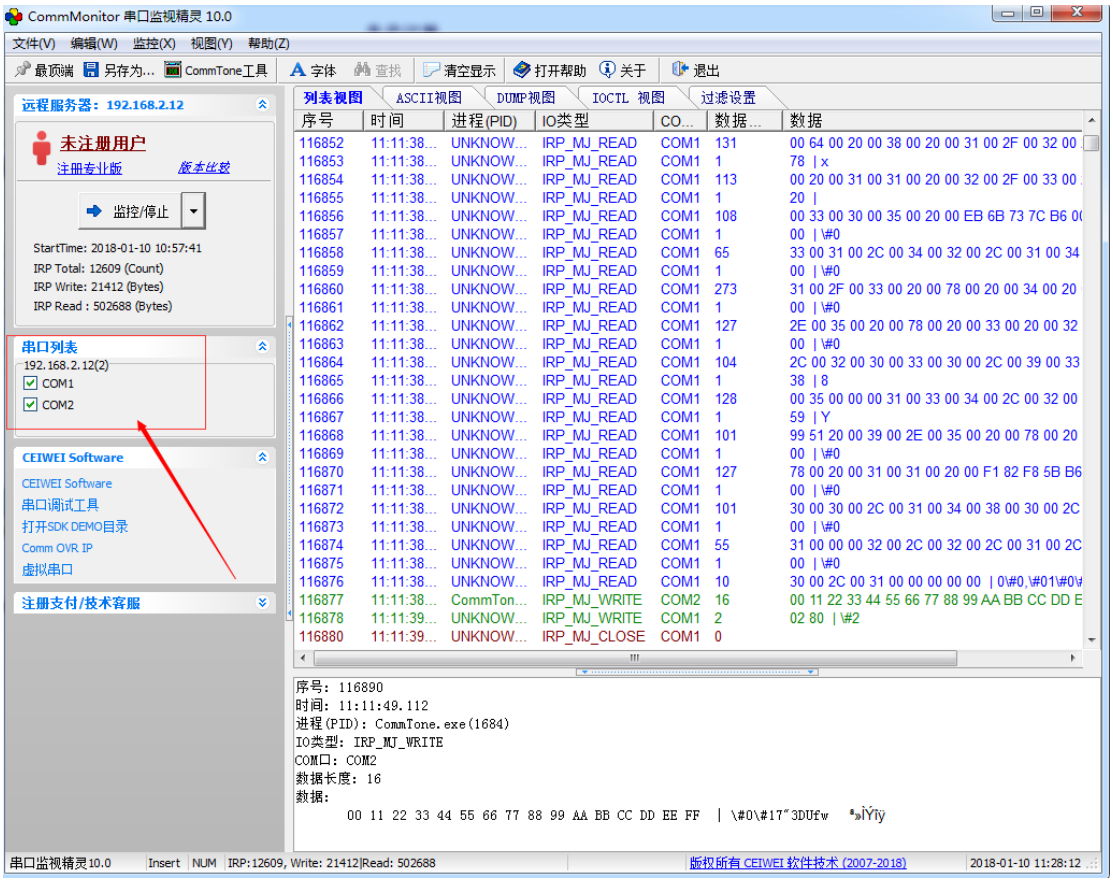
当前 IOCTLs 过滤数据放入该视图；内容包括: 目标进程、COM 号、IOCTLs 名称、数据。

视图查找

请参看：查找数据

串口列表/过滤

如下图【串口列表】视图组，对某个或多个串口监控： 如果需要监控的串口，可以前面 √ 打上，否则反之。



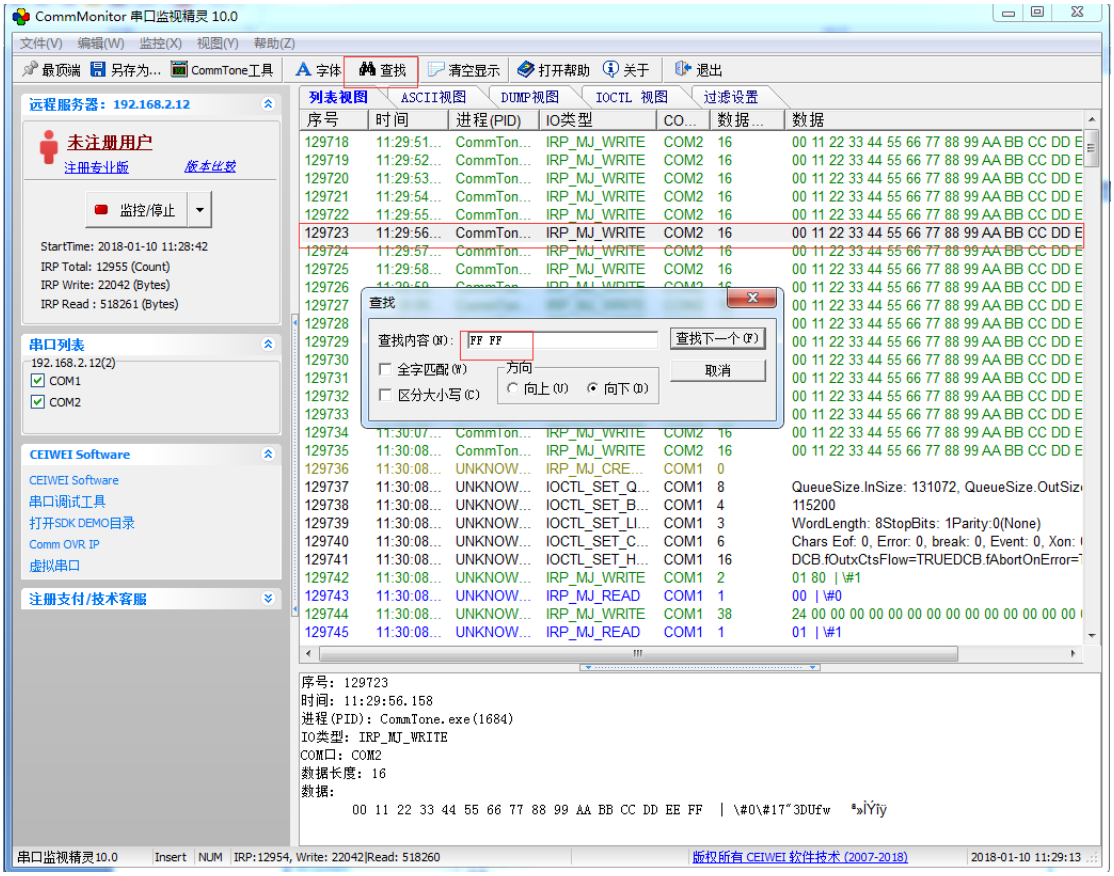
备注:

当前串口列表视图组，会自动感知串口硬件 PNP 功能即插即用的（虚拟、或 USB 转串口）。当（虚拟、或 USB 转串口）安装/卸载完成会自动处理当前监控状态，如果是新安装的串口，内核驱动会自动绑定监控的；如果是卸载，内核驱动会自动卸载绑定监控。

查找数据

1. 打开【编辑】菜单->【查找】|【查找下一个】

2. 如下图【工具条】视图->【查找】



备注:

查找根据当前窗口的活动控制视图来确定查找的对像，如果当前是列表视图，光标焦点放在列表视图上，只要该视图有数据则【查找】按钮就会变得可用否则是灰色的。

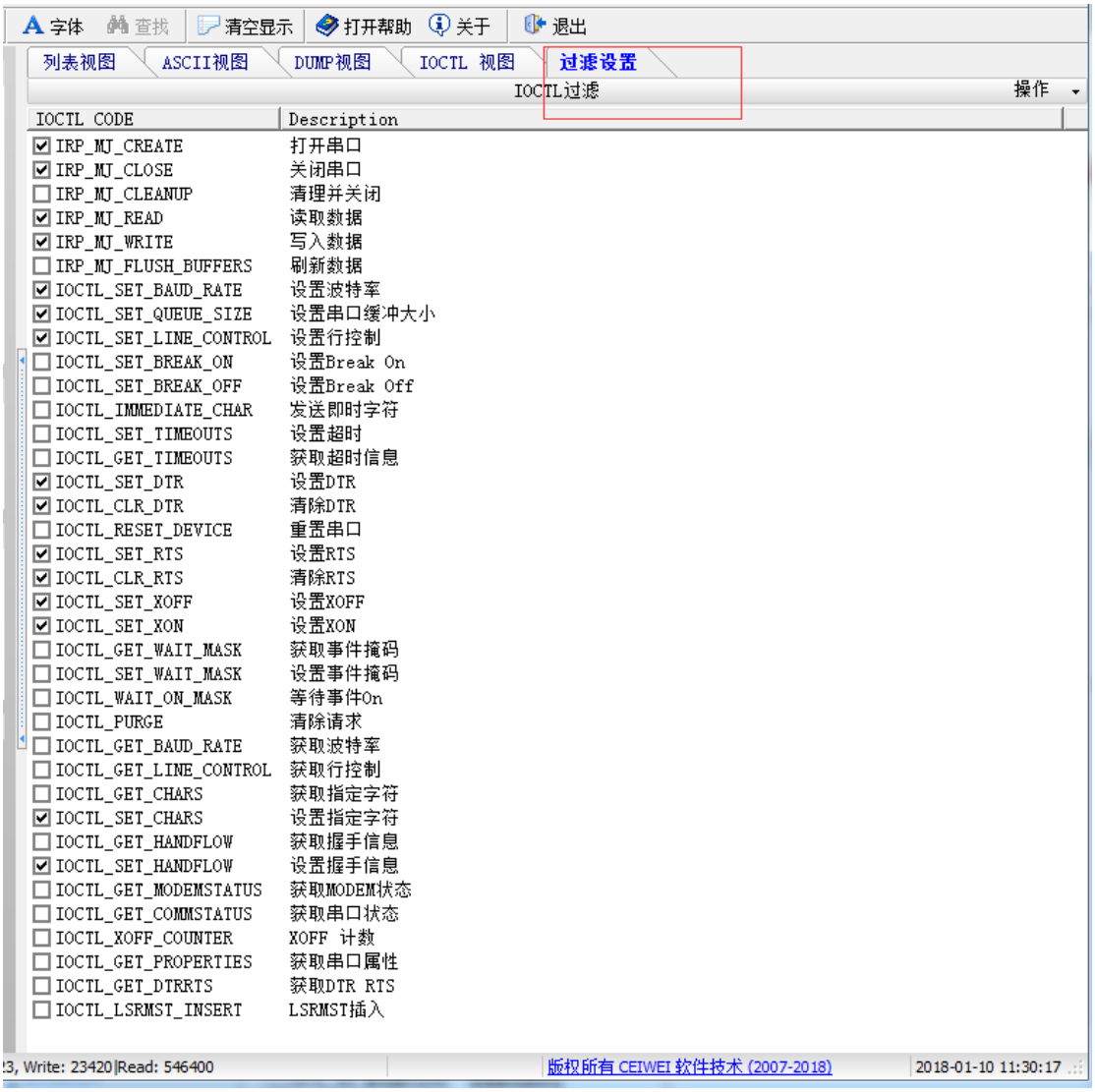
其他视图：ASCII，DUMP，IOCTL 视图一样，需要有数据和光标焦点放在视图窗口上。

查找对话框是 Windows 默认，会随本地化的变化而变化。

查找下一个，按 F3 和 Windows 记事本查找功能一样。

IOCTLs 过滤

1. 打开如下图【过滤设置】。
2. 操作按钮，包括【全部选择】、【全部清除】、【反选】当前过滤视图的选项。

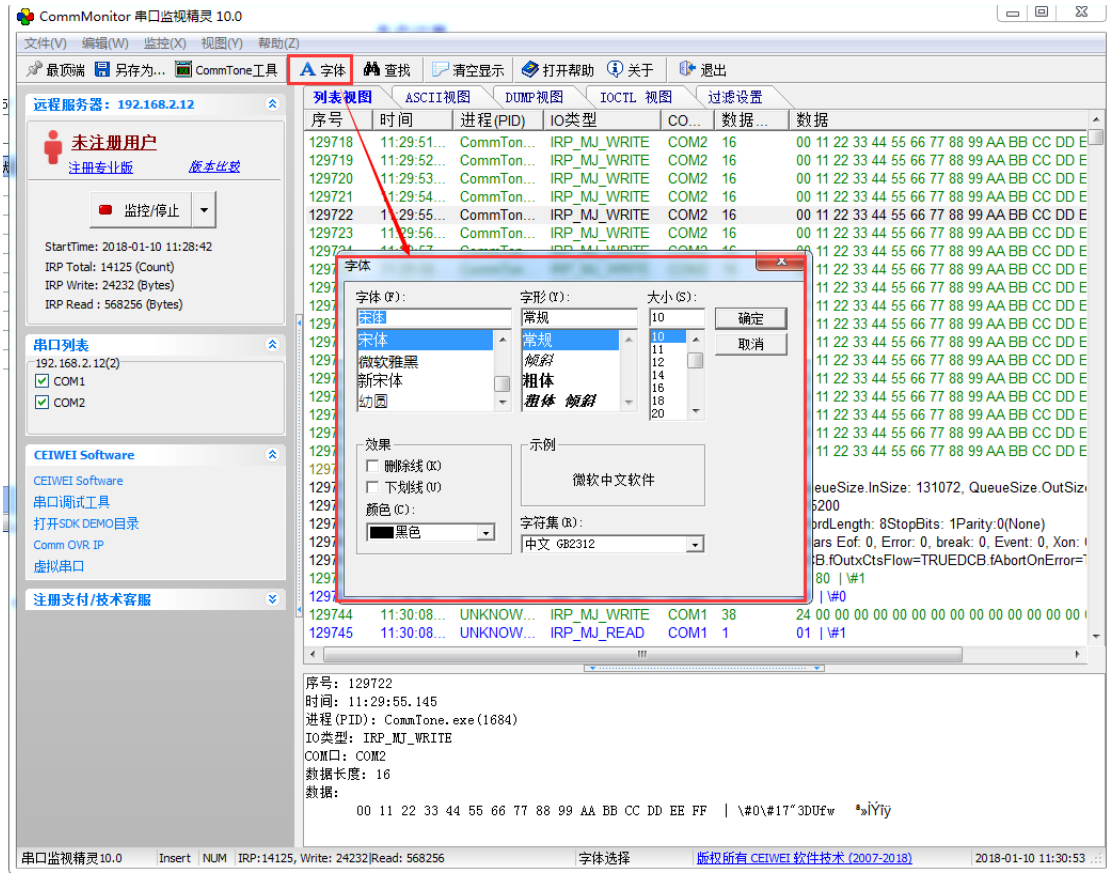


备注:

设置串口事件 IOCTL 过滤，当设置完后，需要离开【过滤设置】视图，系统会自动应用当前 IOCTL 设置到内核驱动。

选择字体

操作按钮【字体】。

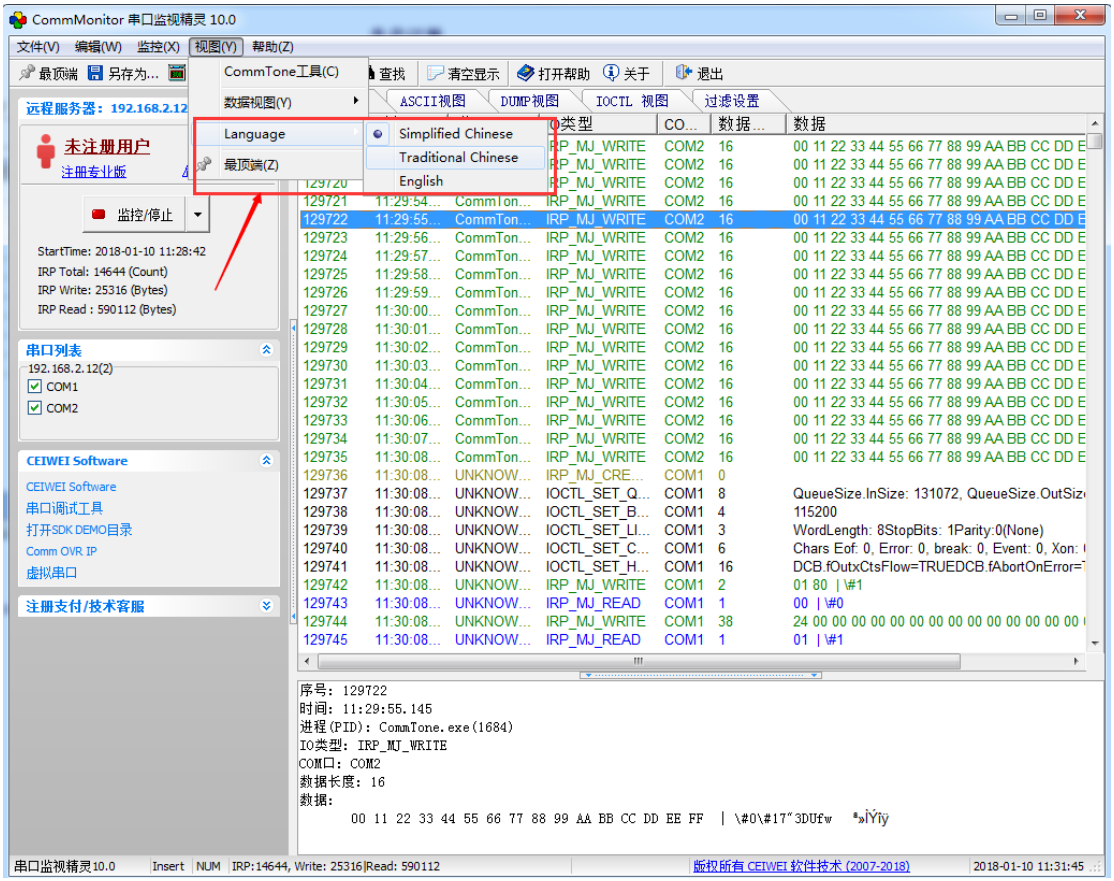


备注:

当按下字体钮后，可更改 列表视图，ASCII,DUMP 视图的字体。

语言选择

打开【视图】菜单->【Language】如下图：

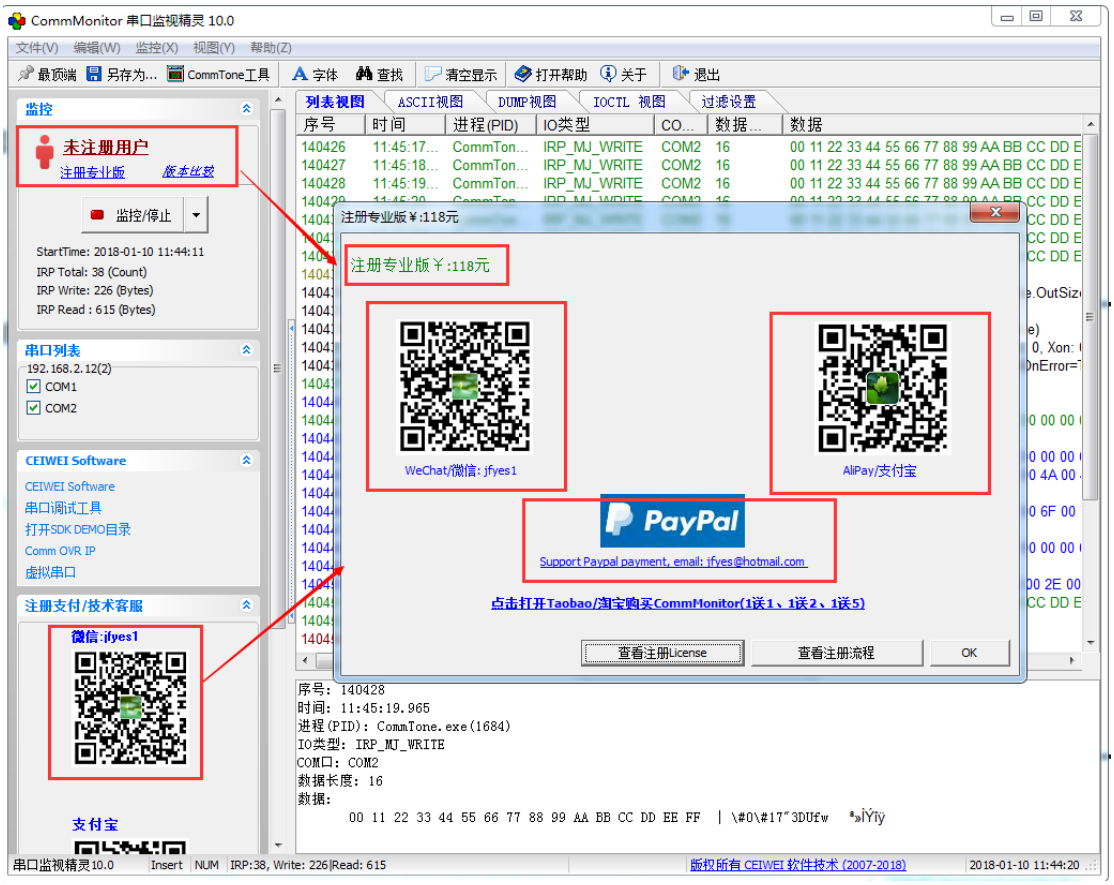


备注:

暂支简体中文，繁体中文，英文。

用户注册

未注册用户单击【用户图标】弹出支付窗口以完成付款并注册。
当前支持微信、支付宝、Paypal 支付。



备注：

未注册用户完成支付后，请将 CommMonitor 安装目录下的 request.license 文件发给开发者生成专业版授权文件，以完成注册。

OEM SDK 二次开发

ActiveX 接口(CommMonitorX.ocx)

CommMonitor ActiveX OCX SDK 开发手册

1、Demo\ 目录的 Delphi、VS2012(C#,VC,VB.net)

\Demo\Delphi	Delphi Demo
\Demo\VS\2012\C#	C# Demo
\Demo\VS\2012\VB.net	VB.net Demo
\Demo\VS\2012\VC++	VC.net Demo
\Demo\Bin	编译后输出的 EXE 目录，其 CommMonitorX.ocx 也在此目录

2、CommMonitorX.ocx 注册方法

1.注册 OCX 组件: Reg.bat ;
2.反注册组件: UnReg.bat。

3、相关常量

IRP_MJ_CREATE	=$00; //串口打口
IRP_MJ_CLOSE	=$02; //串口关闭
IRP_MJ_READ	=$03; //读取数据
IRP_MJ_WRITE	=$04; //写入数据
IRP_MJ_DEVICE_CONTROL	=$0E; //控制码
IRP_MJ_CLEANUP	=$12; //清理串口实例

CommMonitorX.ocx 的调用接口 Delphi 描述

4、方法：

StartMonitor： 开始监控；

@Param：cKey 调用 Key 认证，对于正式版本才有效，demo 填写空字符串；
@Param：cPortName 指定要监控的串口号；
@Return：调用成功返回 True，否则返回 False；
function StartMonitor(const cKey: WideString; const cPortName: WideString): WordBool;safecall;

StopMonitor：停止监控，无参数；

@Return： 调用成功返回 True，否则返回 False；
function StopMonitor(): WordBool; safecall;

PauseMonitor：暂停/恢复监控；

@Param：bPause 指定 True 为暂停监控，False 恢复监控；
@Return：调用成功返回 True，否则返回 False；
function PauseMonitor(bPause: WordBool): WordBool; safecall;

SerialCtrlCode：I OCTL 转换

@Param：ctlCode 监控得到的 CtrlCode
@Return：转换成功返回可识别的 Ctrl 码，否则无效 0；
function SerialCtrlCode(ctlCode: LongWord): LongWord; safecall;

About：关于对话框，无参数，无返回值；

procedure About(); safecall;

StartNet: 启动连接远程网络 Socket

@Param: cKey 调用 Key 认证，对于正式版本才有效，demo 填写空字符串；

@Param：cPortName 指定要监控的串口号；

@Return：调用无返回 Void

procedure StartNet(const cKey: WideString; dwSocketType: Shortint; const sIPAddr: WideString; dwPort: Word); safecall;

StopNet: 停止网络 Socket

参数无；

procedure StopNet; safecall;

5、事件：

OnMonitor：二进制数据蒲获事件

参数：dwIndex 蒲获数据的序号

参数：dTime 蒲获数据的时间，精确到毫秒级

参数：sPortName 为当前蒲获串口的名称

参数：dwCtrlCode 为控制码或 IRP_MJ function Code; 大于$400 的为控制码 (IRP_MJ_DEVICE_CONTROL)

参数：dwProcessID 为当前蒲获串口所在的进程的 PID

参数：sProcessName 当前蒲获串口所在的进程名称

参数：vtData 为二进制数据

参数：dwSize 为二进制数据的长度

procedure OnMonitor(dwIndex: LongWord; dTime: TDateTime; const sPortName: WideString; dwCtrlCode: LongWord; dwProcessID: LongWord; const sProcessName: WideString; var vtData: OleVariant; dwSize: Integer);

OnAscii： ASCII 码数据蒲获事件

参数：dwIndex 蒲获数据的序号

参数：dTime 蒲获数据的时间，精确到毫秒级

参数：sPortName 为当前蒲获串口的名称

参数：dwCtrlCode 为控制码或 IRP_MJ function Code; 大于$400 的为控制码 (IRP_MJ_DEVICE_CONTROL)

参数：dwProcessID 为当前蒲获串口所在的进程的 PID

参数：sProcessName 当前蒲获串口所在的进程名称

参数：sData 它会将蒲获的数据以 16 进制格式化输出为可识别 ASCII 码数据,同时 ctlCode 码也会格式化输出

参数：dwSize 为二进制数据的长度，不是 16 进制格式化输出的长度

procedure OnAscii(dwIndex: LongWord; dTime: TDateTime; const sPortName: WideString; dwCtrlCode: LongWord; dwProcessID: LongWord; const sProcessName: WideString; const sData: WideString; dwSize: Integer);

OnNetStatus: 网络事件

参数：SockType 网络 Socket 的类型 0=TCP/1=UDP

参数：Status 网络 Socket 的连接状态 0=已连接/1=已断开

procedure(SockType: Shortint; Status: Shortint) ;

6、Serial Control code 常量说明：

经过 SerialCtrlCode 函数转换后，可以对应如下控码信息。

IOCTL_SERIAL_SET_BAUD_RATE = 1;
IOCTL_SERIAL_SET_QUEUE_SIZE = 2;
IOCTL_SERIAL_SET_LINE_CONTROL = 3;
IOCTL_SERIAL_SET_BREAK_ON = 4;
IOCTL_SERIAL_SET_BREAK_OFF = 5;
IOCTL_SERIAL_IMMEDIATE_CHAR = 6;
IOCTL_SERIAL_SET_TIMEOUTS = 7;
IOCTL_SERIAL_GET_TIMEOUTS = 8;
IOCTL_SERIAL_SET_DTR = 9;
IOCTL_SERIAL_CLR_DTR = 10;
IOCTL_SERIAL_RESET_DEVICE = 11;
IOCTL_SERIAL_SET_RTS = 12;
IOCTL_SERIAL_CLR_RTS = 13;
IOCTL_SERIAL_SET_XOFF = 14;
IOCTL_SERIAL_SET_XON = 15;
IOCTL_SERIAL_GET_WAIT_MASK = 16;
IOCTL_SERIAL_SET_WAIT_MASK = 17;
IOCTL_SERIAL_WAIT_ON_MASK = 18;
IOCTL_SERIAL_PURGE = 19;
IOCTL_SERIAL_GET_BAUD_RATE = 20;
IOCTL_SERIAL_GET_LINE_CONTROL = 21;
IOCTL_SERIAL_GET_CHARS = 22;
IOCTL_SERIAL_SET_CHARS = 23;
IOCTL_SERIAL_GET_HANDFLOW = 24;
IOCTL_SERIAL_SET_HANDFLOW = 25;

```
IOCTL_SERIAL_GET_MODEMSTATUS        =  26;
IOCTL_SERIAL_GET_COMMSTATUS         =  27;
IOCTL_SERIAL_XOFF_COUNTER           =  28;
IOCTL_SERIAL_GET_PROPERTIES         =  29;
IOCTL_SERIAL_GET_DTRRTS             =  30;
IOCTL_SERIAL_LSRMST_INSERT          =  31;

//Serenum reserves function codes between 128 and 255.  Do not use.
IOCTL_SERIAL_CONFIG_SIZE            =  32;
IOCTL_SERIAL_GET_COMMCONFIG         =  33;
IOCTL_SERIAL_SET_COMMCONFIG         =  34;
IOCTL_SERIAL_GET_STATS              =  35;
IOCTL_SERIAL_CLEAR_STATS            =  36;
IOCTL_SERIAL_GET_MODEM_CONTROL      =  37;
IOCTL_SERIAL_SET_MODEM_CONTROL      =  38;
IOCTL_SERIAL_SET_FIFO_CONTROL       =  39;
```

备注:

ActiveX 开发接口是试用版和发布的内核驱动是一样的，这个 OCX 控件试用版本的限制是每调用 200 次读写串口数据就会自动终止目标程序，并将试用的字符通过串口数据传到界面上来。