目录

一,安装及卸载

嘟咔组态软件是一组嵌入在微软的软件开发平台 Visual Studio 里（简称 VS），并以控件的形式显示在工具箱内的组件。故在安装本嘟咔组态软件之前，请务必先安装好微软开发平台 Visual Studio。目前支持包括 Visual Studio 2010 以后的版本(包括 VS2010)。

1.1 XP，Win7 及 Win8 环境下安装

先安装好 Visual Studio（如果之前已安装后，则无需再安装），然后安装嘟咔软件即可。

1.2 卸载

通过“开始”->“程序”->“Dooka”->“Uninstall.exe”即可卸载。如图

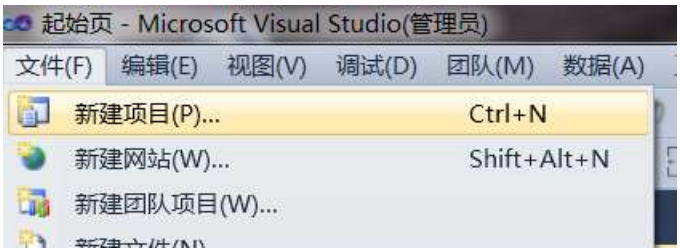
二,新建一个项目

安装完成后，准备好需要连接的设备（比如三菱 PLC）及所需通讯线。然后启动 Visual Studio，建立一个窗口程序。嘟咔组态目前只支持 C# Winform 和 VB 类型的窗体项目。

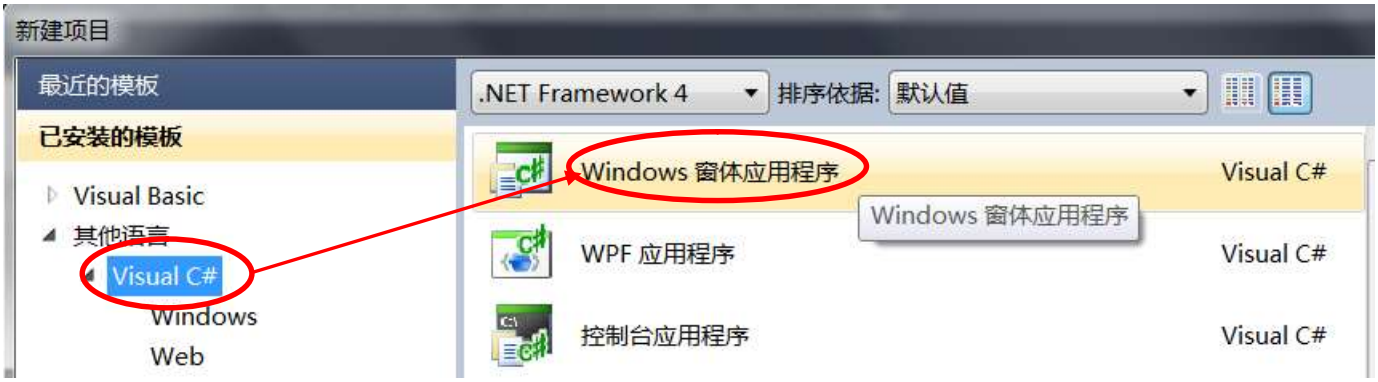
2.1 新建一个 C# Winform 窗体项目

2.1.1 新建一个项目

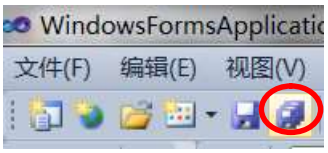
启动 Visual Studio（比如 VS2010），通过“文件”->“新建项目”建立新项目。



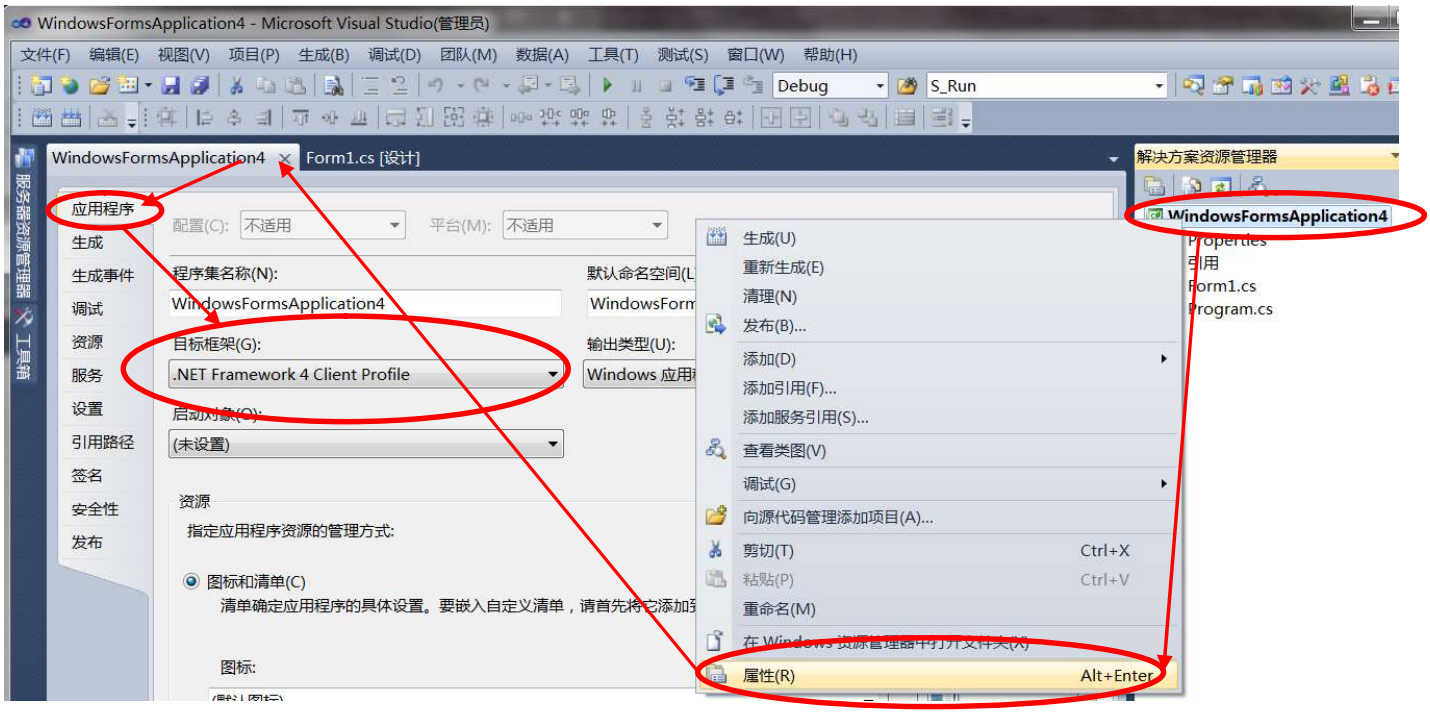
在弹出的窗口中新建 C# Winform 窗体项目。如下图所示：



新建完项目后，先保存项目。如下图。因为嘟咔组件需要在 Microsoft.NET Framework 4 以上框架中才更好地运行。而 VS2010 新建的项目默认框架为”Microsoft.NET Framework 4 Client Profile”。所以需要更改新建立的项目框架。（VS2012 和 VS2013 没有这个问题）



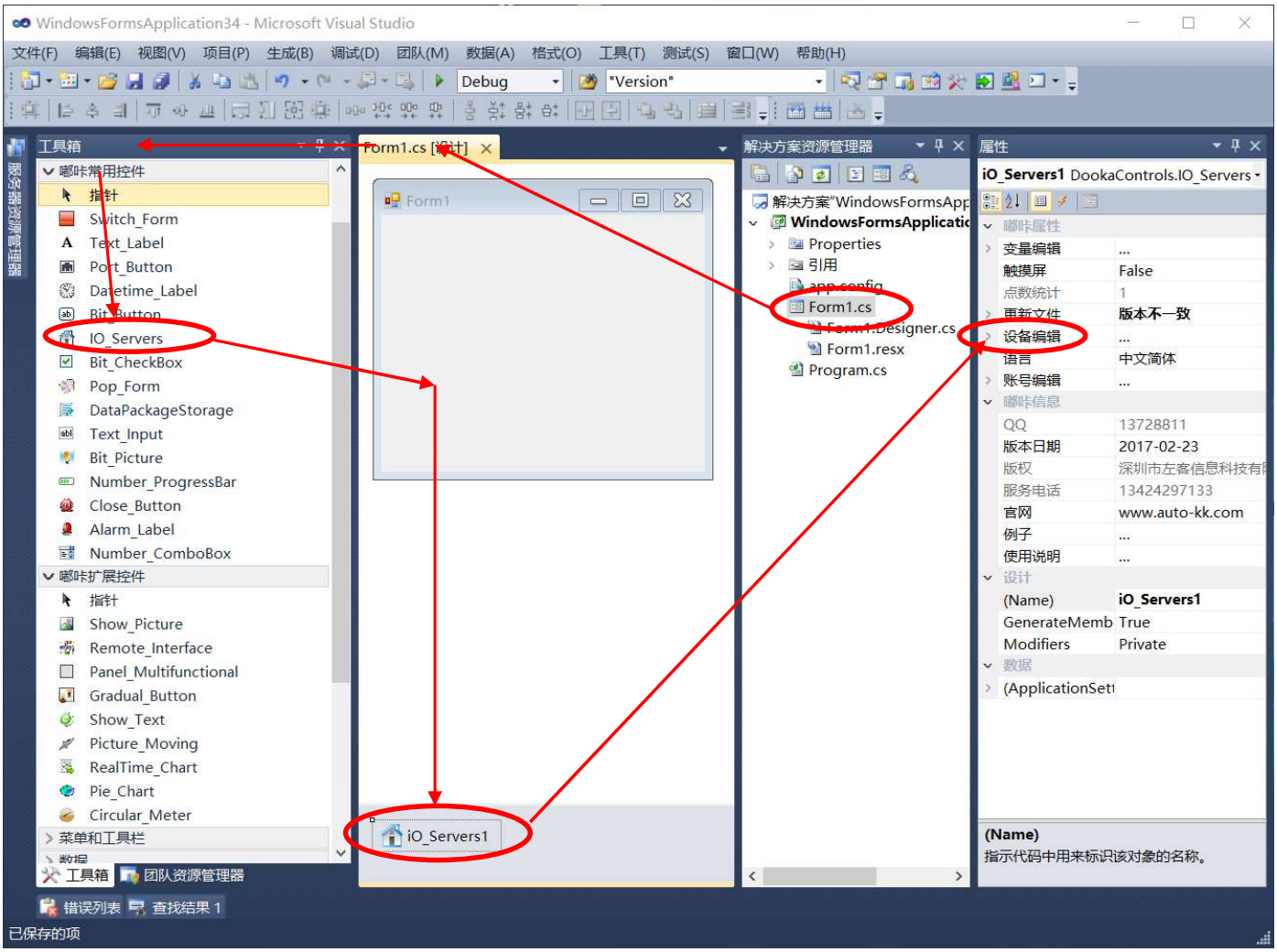
在“解决方案资源管理器”中通过右键单击项目，在弹出的菜单栏中选择“属性”。如下图：



把目标框架改成“.NET Framework 4”并保存。

2.1.2 “设备宿主控件”

双击项目中的主窗体“Form1.cs”，然后在工具箱的“嘟咔常用控件”选项卡中选择“设备宿主控件(IO_Servers)”并添加 。（提示：如果工具箱没有显示，可用组合快捷键“Ctlr+Alt+X”打开）



2.1.3 设备编辑框

添加“设备宿主控件”后，主窗体的正下方将显示刚添加的宿主控件“iO_Servers1”。如上图所示。右键单击刚添加的“iO_Servers1”控件，在弹出的菜单中选择“属性”,在“属性”工具栏中单击“设备编辑”项最右边的按钮弹出“设备编辑”框。如下图：

“设备编辑”框分为 4 个栏目，分别为左上的“IO 设备栏”，左下的“端口栏”，右上的“设备栏”和右下的“注解栏”。

“IO 设备栏”：通过树形控件显示目前所支持连接的设备协议。

“设备栏”：显示和设置已添加的设备。

“端口栏”：显示和设置已添加的端口。

“注解栏”：显示如何设置设备及端口的主要参数。

在“IO 设备栏”中依次展开，找到将要连接设备所用的协议。比如，通过电脑串口连接三菱 FX 系列 PLC 的 FX-232-BD。我们选择“MC 协议(格式 1)”。双击“MC 协议(格式 1)”节点，新建一个设备。设备新建完成后，需要为设备指定一个通讯端口。如果没有合适的通讯端口。则可通过“端口栏”新建一个连接端口并指定。如上图所示。

温馨提示:可以同时添加多个设备及多个端口。多个设备也可以同时指定同一个端口。

设备栏中各子项：

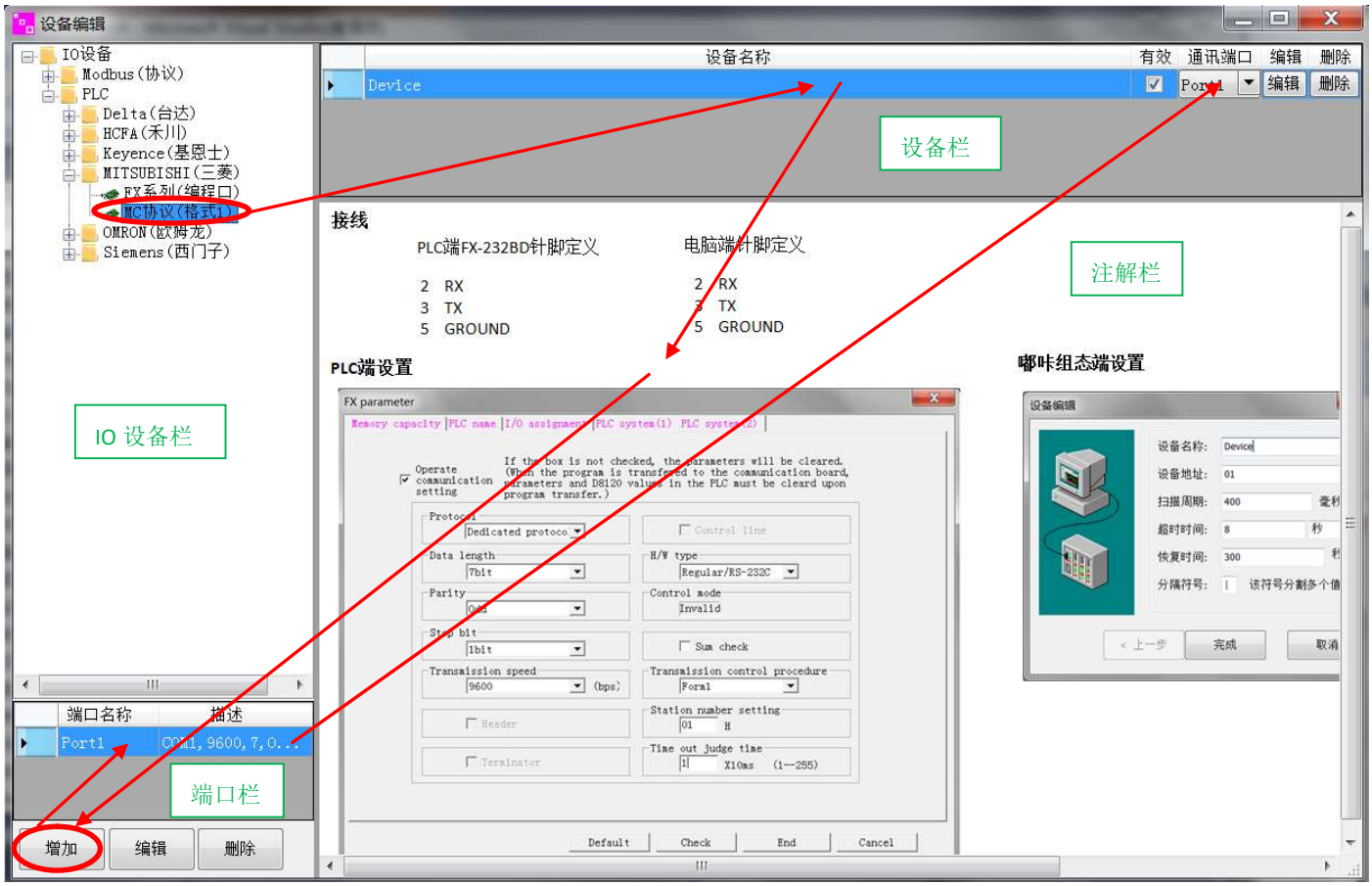
设备名称：唯一识别设备的名称。

有效：设置运行时，设备是否需要加载。如果不选中，则不加载与该设备相关的组件。

通讯端口：指定通过哪一个端口与设备通讯。一个端口可同时由多个设备指定。（比如 RS485 通讯）

编辑：修改与设备相关的参数

删除：删除该设备及所有该设备的变量。



2.1.4 变量编辑框

设备添加完成后，接下来编辑要访问的设备变量，即设备中的输入输出信号或存储单元。在“iO_Servers1”控件的属性中单击“变量编辑”项最右边的按钮弹出“变量编辑”框（参见添加“设备宿主控件”一节）。如下图：

“变量编辑”框分为 2 个栏目，分别为左边“目录栏”和右边“变量栏”。

“目录栏”为树形控件，其各个根节点为不同设备节点。当变量很多时，为了便于管理，每个节点均可建新的节点。即右键单击节点新建文件夹节点。选中每个节点时，“变量栏”将显示该节点下所定义的所有变量。

“变量栏”有两个活页，分别为“编辑”和“选择”。当需要操作变量时，选择“编辑”活页，并通过上面的工具栏 “插入行”和“删除行”添加或删除变量。当为控件选择一个变量时，可选择“选择”活页，并通过双击某一变量选中并退出。当项目所需的变量编程完成后，退出(退出即可自动保存)。

下面一一介绍各项：

设备变量信息:

名称：设备变量的简称。

存储类型：设备存储器地址的类型

偏置地址：设备存储器的地址。

数量：一次读取的数量，最少一个。如果是字符串数据类型，则有时候需要指定该字符串的最大字符数。

数据类型：设备变量值的数据类型。

访问方式：只读，只能读操作；只写，只能写操作；读写，能读能写操作

最小值，最大值：操作该变量值时限制修改范围。如果为空，则不限制范围。

ID：唯一识别该变量的编号，最小值为 1。（代码层操作时该项非常有用）

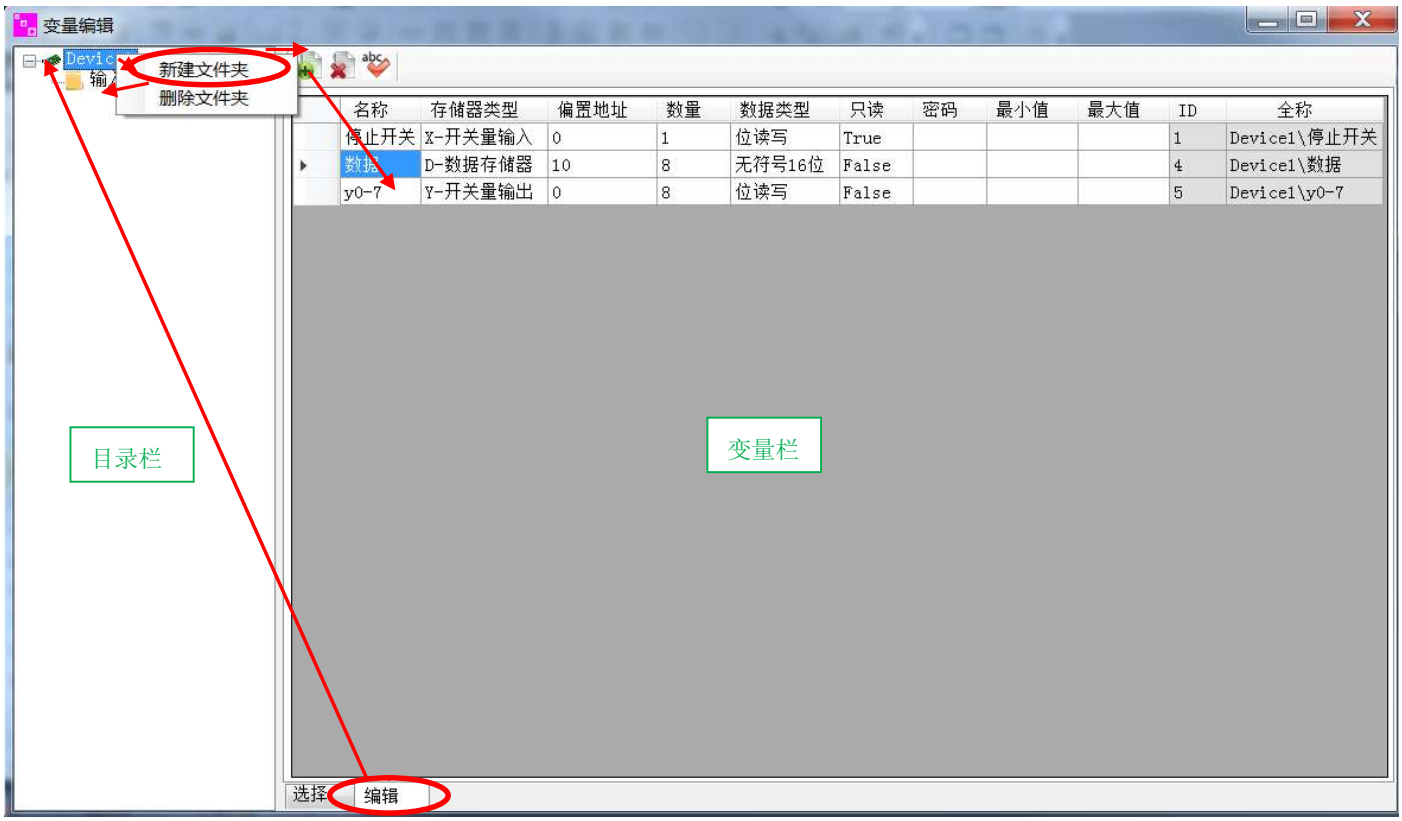
全称：唯一识别该变量的名称路径。（代码层操作时该项非常有用）

根据变量ID获取名称路径： Dooka.Controls.IO_Servers.VariableIdToName(ID)

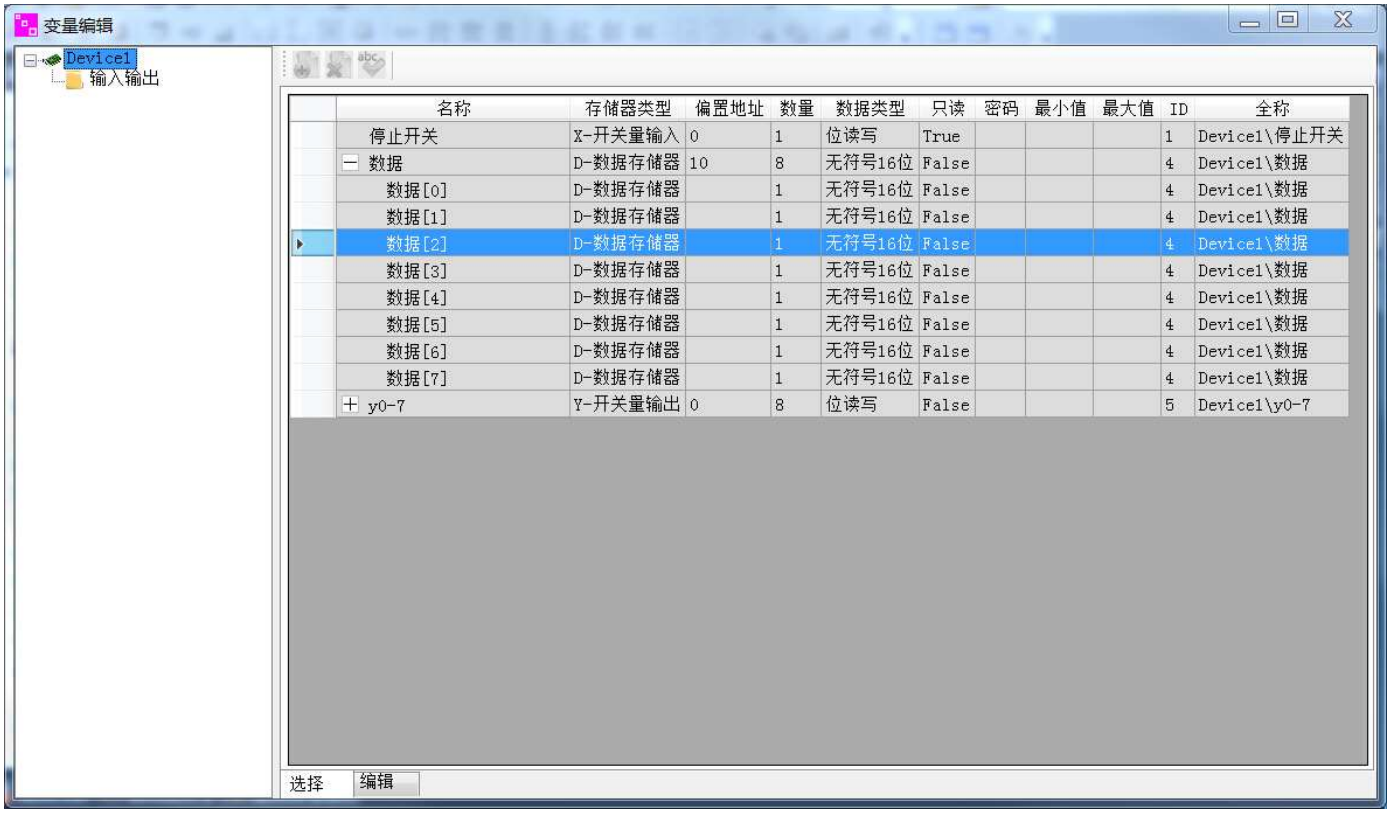
根据变量名称路径获取ID ： Dooka.Controls.IO_Servers.VariableNameToId (VariableName)

每一个设备变量将由一个Dooka.Controls.VariableWithIndex类对象管理，透过该类对象和静态函数可以对设备变量进行读写操作。(查看高级应用篇查看详细内容)

可以通过新建文件夹的方式进行变量管理。



编辑活页

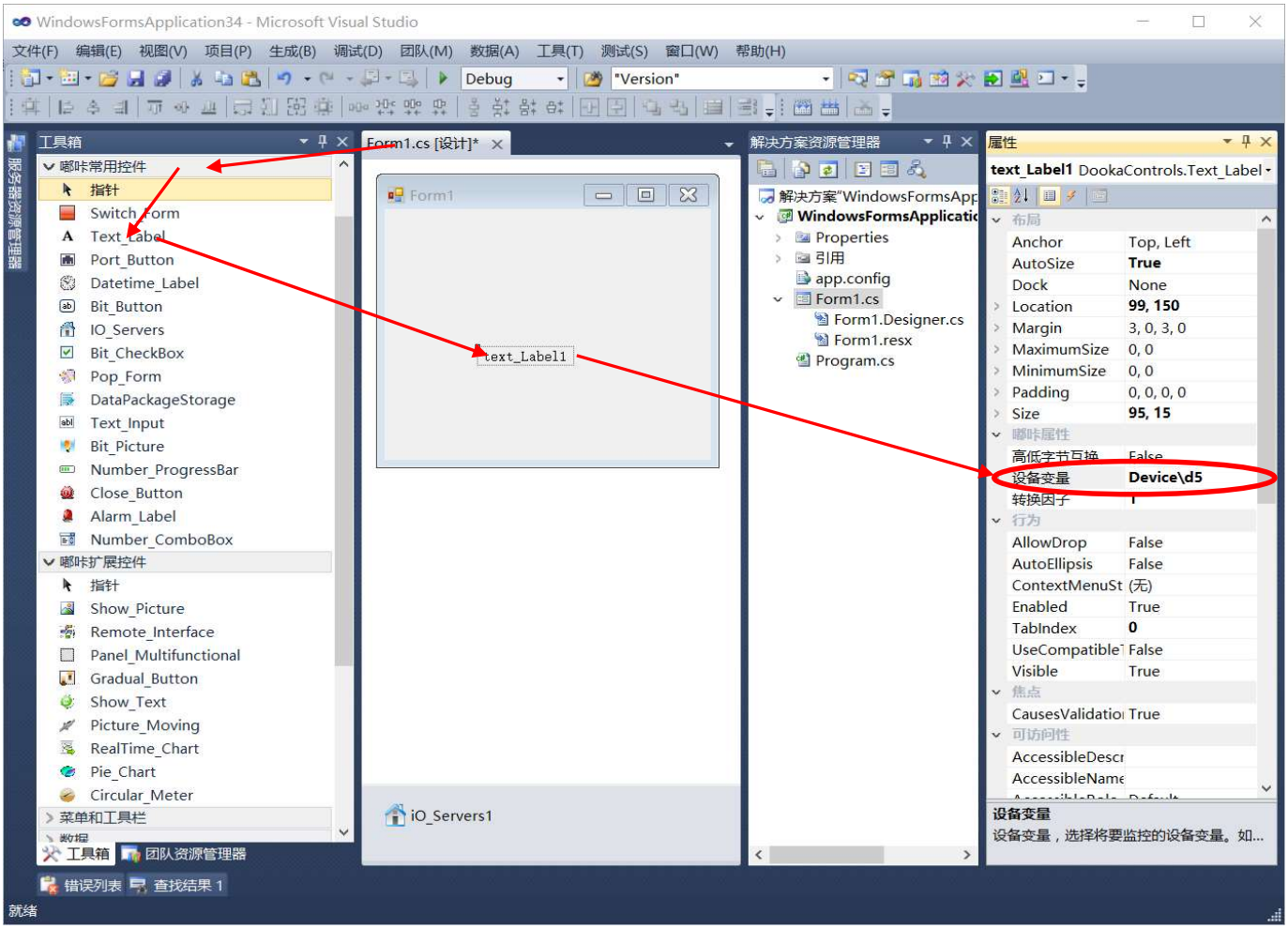


选择活页

名称	存储器类型	偏置地址	数量	数据类型	只读	密码	最小值	最大值	ID	全称
停止开关	X-开关量输入	0	1	位读写	True				1	Device1\停止开关
数据	D-数据存储器	10	8	无符号16位	False				4	Device1\数据
数据[0]	D-数据存储器		1	无符号16位	False				4	Device1\数据
数据[1]	D-数据存储器		1	无符号16位	False				4	Device1\数据
数据[2]	D-数据存储器		1	无符号16位	False				4	Device1\数据
数据[3]	D-数据存储器		1	无符号16位	False				4	Device1\数据
数据[4]	D-数据存储器		1	无符号16位	False				4	Device1\数据
数据[5]	D-数据存储器		1	无符号16位	False				4	Device1\数据
数据[6]	D-数据存储器		1	无符号16位	False				4	Device1\数据
数据[7]	D-数据存储器		1	无符号16位	False				4	Device1\数据
y0-7	Y-开关量输出	0	8	位读写	False				5	Device1\y0-7

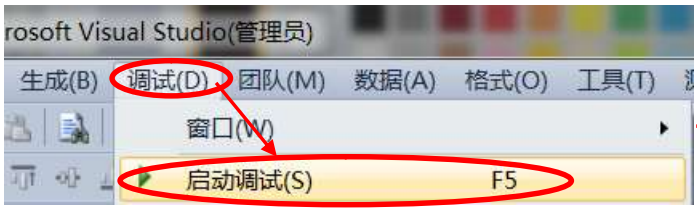
2.1.5 常用控件

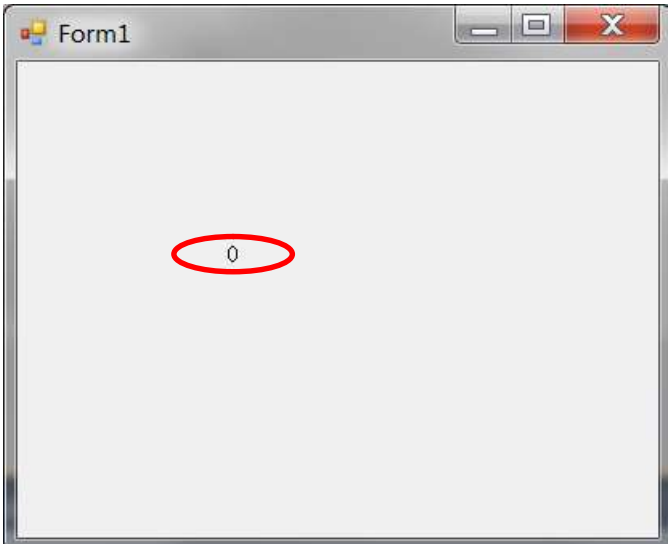
设备及端口设置好后，在工具箱的“嘟咔常用控件”选项卡中选择“Text_Label”控件并添加到主窗体里（如下图）。在新加的控件属性栏中“嘟咔属性”选项卡中的“设备变量”中选择一个已编辑的设备变量。如果没有，可以新编辑一个合适的变量。



2.1.6 运行

通过上面的工作，我们已完成了一个非常简单的项目。现在，我们可以试运行一下看结果。



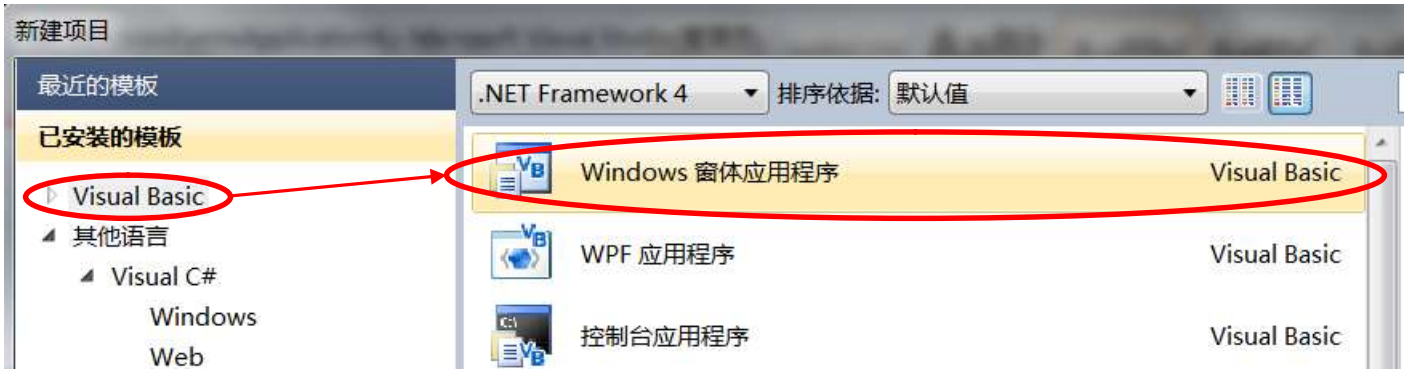


运行结果如右图所示：
监控的设备变量值为“0”；

2.2 新建一个 VB 窗体项目

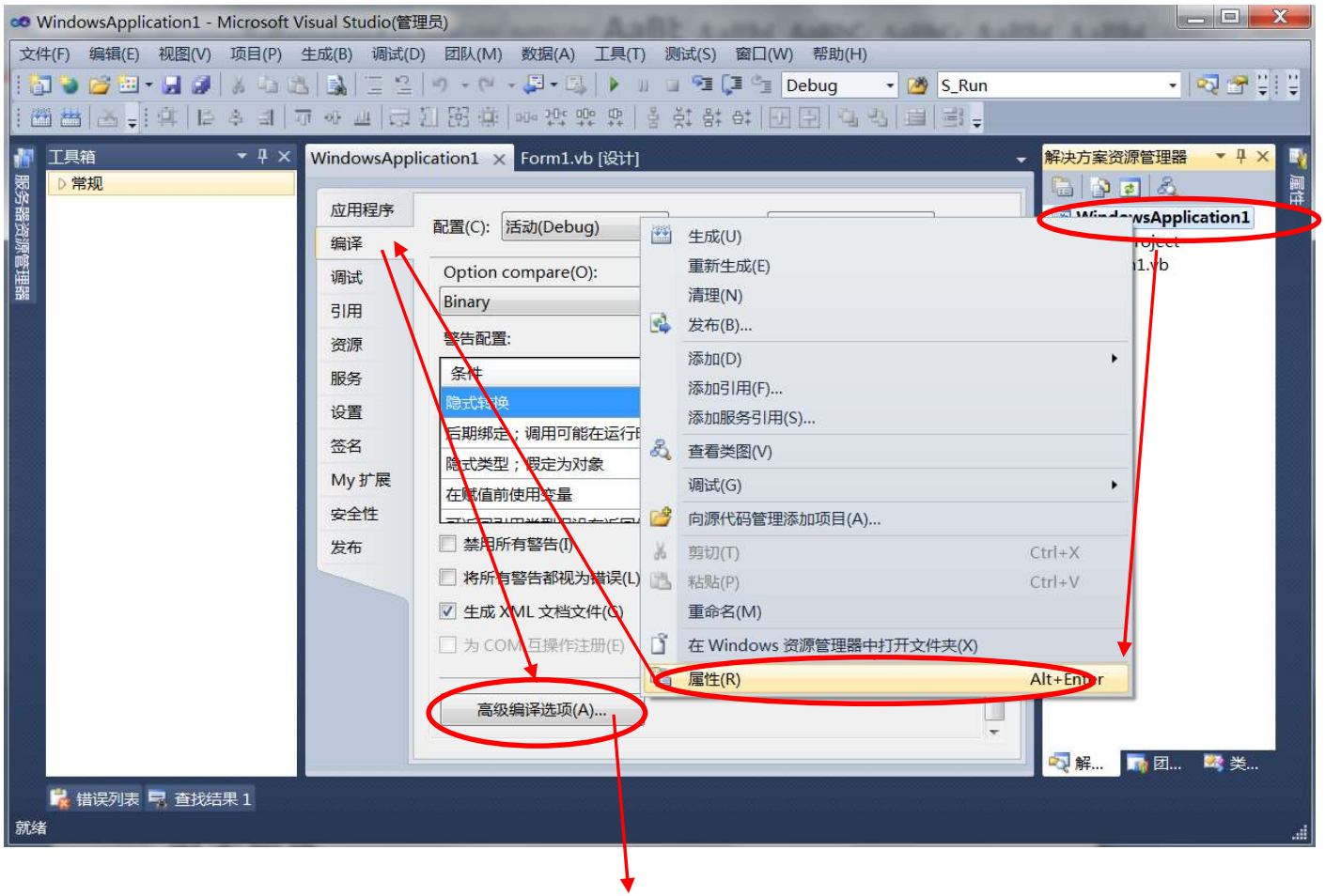
VB 窗体项目的使用跟上面 C# Winform 项目差不多。所不同的有以下几点：

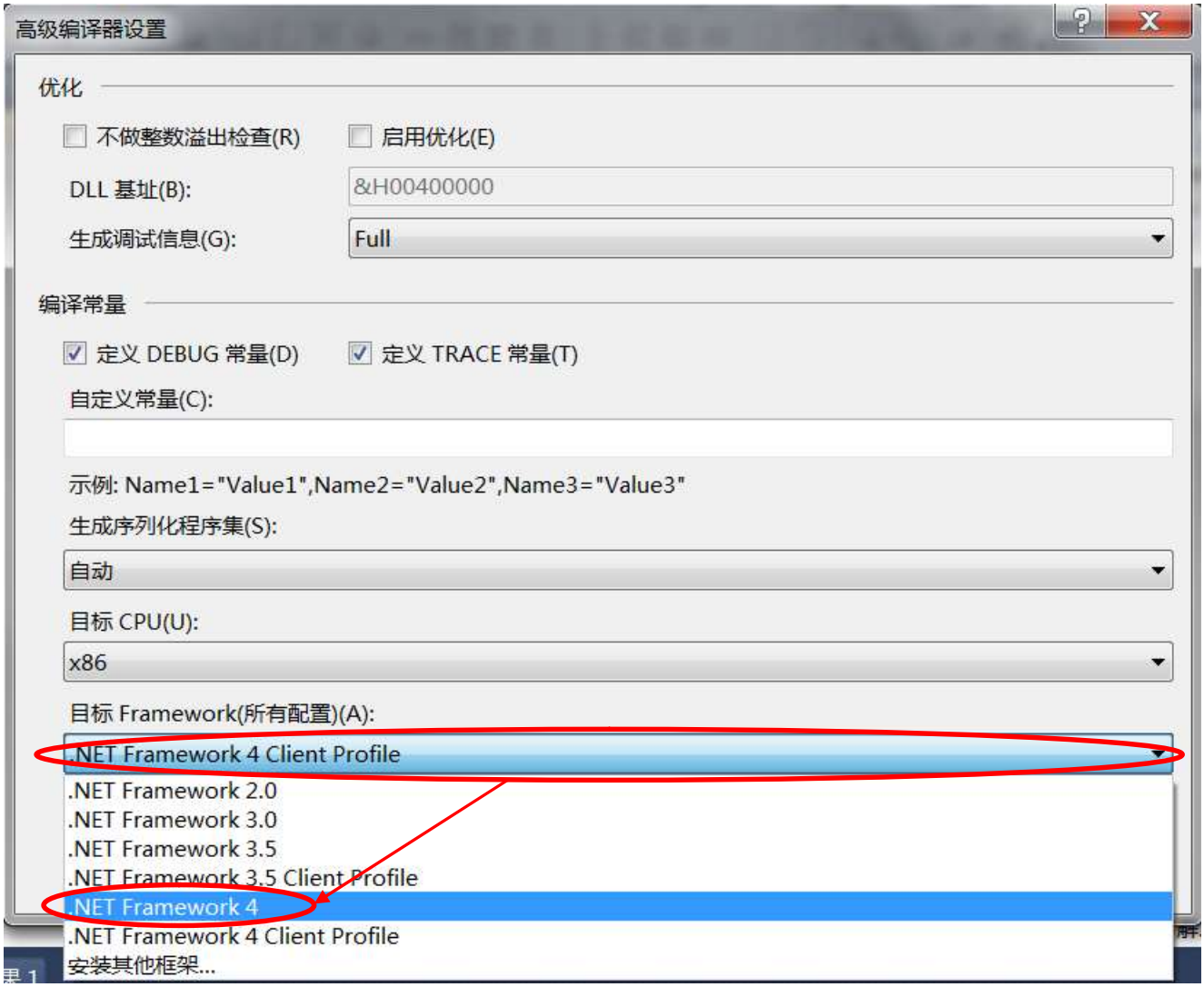
新建项目类型不同：上面的例子是新建一个 C# Winform 窗体项目。而本例是新建一个 VB 窗体项目。如下图：



如果是在 VS2010 平台下，则需要更改目标框架（如下图）。

其余的如添加“设备宿主控件”，“设备编辑”，“变量编辑”，添加监控控件，运行等参照上面的例子。

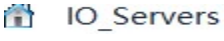




三,组态控件

本章依次介绍各个嘟咔控件的属性及使用说明。每个嘟咔控件除了既有 VS 基本属性外，还具备嘟咔关于设备的特有属性。（选中每一项属性，最下边都会有关于该属性的简要说明哦!）

3.1 “设备宿主控件(IO_Servers)”控件



在连接一个或多个设备之前，首先需要设置好设备的属性及连接所用端口。为了做这些必备工作，需要在项目的主窗体里添加该控件，即在“工具箱”里展开“嘟咔常用控件”选项卡，添加“IO_Servers”到窗口中(一个项目不管连接多少个设备，只需添加一个即可)。在添加的控件属性窗口中(如右图)，有以下几个设置：

设备编辑：编辑将要连接的设备属性及所用连接端口（参见 2.1.3 节）。

变量编辑：编辑将要访问的设备变量，即设备的输入输出状态及存储器（参见 2.1.4 节）。

触摸屏：指示项目是否用在带触摸屏功能的显示屏上。如果为“True”，则带输入的控件将自动开启小键盘。

语言：选择用户的使用语言。

账号管理：创建和编辑用户相关账号和权限。
点数统计：统计项目已编辑了访问设备变量的点数量。
更新文件：当打开由低版本嘟咔组态软件编辑的项目时将显示“版本不一致”。点击使之更新到当前版本。
版本日期：显示当前版本的日期。点击将自动连接到我们官网并查询是否有更新的版本。

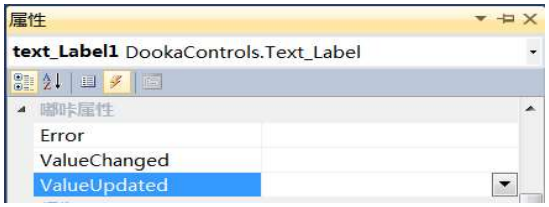
跟设备有关的事件：
OnConnected：当设备通讯成功时引发的事件。事件中的函数参数”Value”为设备的名称。
OnDisconnect：当设备通讯失败时引发的事件。事件中的函数参数”Value”为设备的名称。并不是一次读或写操作失败就判定为设备连接失败。要连续读或写失败的时间超过设备的”超时时间”才算为通讯失败。

3.2 “变量显示(Text_Label)”控件

可以访问并显示设备的输入输出状态及存储器值。

跟嘟咔有关的属性：
设备变量：指定访问的设备变量。
索引号：当要访问的设备变量数量为多个时，指定是哪一个。0 为第一个。
转换因子：控件显示的值为设备变量乘以转换因子。比如设备变量的值为 100，转换因子为 0.1，则控件显示为 10。
高低字节互换：当监控的设备变量类型为 ASC 码时有效。
附加前后缀：显示变量值时附加前缀或后缀或前后缀。以‘*’号代表变量值，比如变量值为10，则“*S”将显示为“10S”。“P=*S”则为“P=10S”。如果没有‘*’，则默认为后缀。如“V”，则显示为“10V”!"

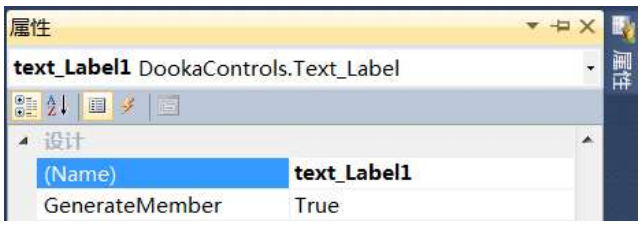
跟设备有关的事件：
Error：当访问设备变量出错时，将引发该事件。
ValueChanged：当访问的设备变量值发生改变时，将引发该事件。
ValueUpdated：当访问的设备变量更新时，将引发该事件。



比如 ValueUpdated 事件：

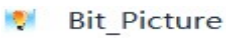
```
private void text_Label1_ValueUpdated(object sender, string e)
{
    //sender,本控件的对象
    //e，该设备变量的值。
}
```

跟设备有关的代码：
控件名称.Variable 为设备变量的对象。如右图，text_Label1.Variable 为设备变量的对象。通过该对象可以访问设备变量的常用参数和成员函数。比如设备变量值，索引号，写操作等。

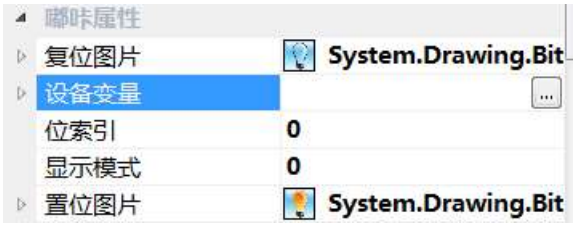


VariableWithIndex类有以下常用成员：
Value：从缓存中获取设备变量的值。
AllValues:返回一个字符串数组。从缓存中获取设备变量的所有值。如果只有一个值，则返回一个数量的字符串数组。
WriteValue：往设备(PLC)中写操作。

3.3 “位监视图(Bit_Picture)”控件



指示灯控件，显示位状态。



跟嘟咔有关的属性：

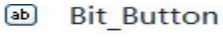
设备变量：指定访问的设备变量。

复位图片：当指定的设备变量值为 0 或 false 时显示的图片。

置位图片：当指定的设备变量值为 1 或 true 时显示的图片。

位索引：当指定的设备变量类型不是位类型时，指明哪一个位。

3.4 “多功能按钮(Bit_Button)”控件



监控位类型的设备变量的按钮控件。

跟嘟咔有关的属性：

设备操作变量：指定操作的设备变量(位类型)。

设备监视变量：指定访问的设备变量。

操作模式：

置位：单击按钮，操作变量置位。

复位：单击按钮，操作变量复位。

点动：当鼠标按下时，操作变量置位。当鼠标释放时，操作变量复位。

反转：单击按钮，操作变量取反，即复位时置位，置位时复位。

弹出确认窗口：指定在操作时是否弹出确认窗口。

复位文本：指定当监视变量为 0 或 false 时要显示的文本。

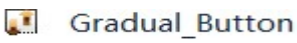
置位文本：指定当监视变量为 1 或 true 时要显示的文本。

复位图片：指定当监视变量为 0 或 false 时要显示的图片。

置位图片：指定当监视变量为 1 或 true 时要显示的图片。

位索引：当指定的监视变量类型不是位类型时，指明哪一个位。

3.5 “递增递减按钮(Gradual_Button)”控件



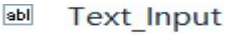
按一定的增减量修改设备变量的值。

跟嘟咔有关的属性：

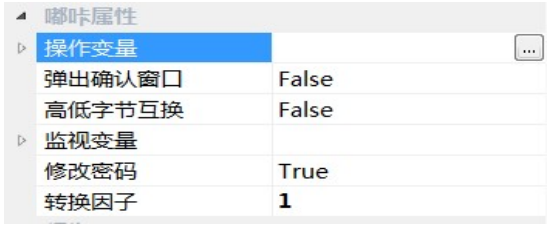
递增减值：指定每次单击按钮，对应的设备变量值将增加或减少的值。(正整数为递增，负整数为递减)

设备变量：需要操作的设备变量。

3.6 “变量输入(Text_Input)”控件



监控设备的输入输出状态及存储器值。



跟嘟咔有关的属性：

操作变量：指定操作的设备变量。

监视变量：指定访问的设备变量。

弹出确认窗口：指定在操作时是否弹出确认窗口。

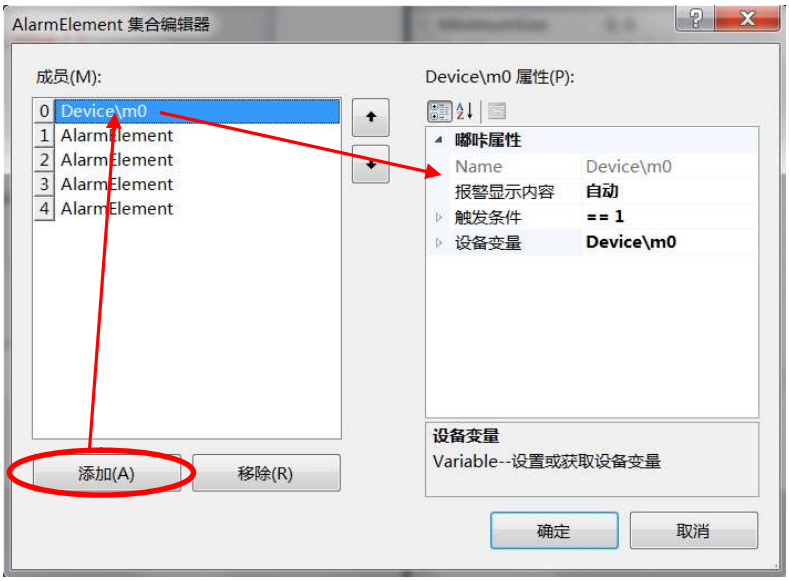
高低字节互换：当监控的设备变量类型为 ASC 码时有效。
转换因子：控件显示的值为设备变量*转换因子。
附加前后缀：显示变量值时附加前缀或后缀或前后缀。以‘*’号代表变量值，比如变量值为 10，则“*S”将显示为“10S”。“P=*S”则为“P=10S”。如果没有‘*’，则默认为后缀。如“V”，则显示为“10V”!"

3.7 “报警控件(Alarm_Label)”控件

报警提示控件。通常用作全局报警提醒。因此，最好把该控件放置在主窗体里。当没有报警时，该控件是隐藏的。

跟嘟咔有关的属性：

导出文件：把故障信息集合里的内容导出为.CSV 文件。
导入文件：把.CSV 文件内容导入到故障信息集合里。建议导入文件前，先导出文件，熟悉文件的格式后再按规则改写其他内容，然后再导入。
交替间隔：当多个故障信息发生时，指定多少秒轮流显示其他故障信息。当为 0 时，将不轮流显示其他故障信息。
闪烁显示：是否闪烁地显示故障信息。
记录保留天数：指定最大保留记录天数，可以用 Alarm_View 控件浏览报警记录。0 为不存储任何记录。
没有报警时隐藏：指定在没有报警发生时是否不显示。
故障信息集合：编辑故障条目。如下图：通过“添加按钮”可以添加若干个故障条目。
设备变量：指定要访问的设备变量。
报警显示内容：当条件满足时显示的内容。
触发条件：设置报警发生的条件。



跟设备有关的事件：

Alarm：当故障发生时引发的事件。
Clear：所有报警清除时引发的事件。
ElementStatusSwitch：任何一个报警条目状态发生改变时引发的事件。

3.8 “画面切换按钮(Switch_Form)”控件

方便在多个画面中来回切换显示。

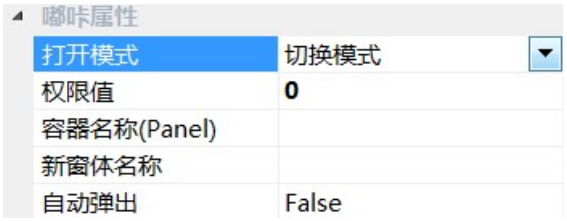
跟嘟咔有关的属性：

新窗体名称：指定要打开的窗体名称。

打开模式：指定以何种方式打开新窗体。

切换模式：以子窗体的方式打开指定窗体。

弹出模式：以对话框的方式打开指定窗体。



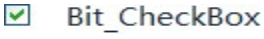
权限值：打开新窗体所需要的权限。权限值越高，权限越大，高权限值的用户可以打开底权限设置。0 为无需任何权限。

容器名称：打开模式为“切换模式”有效。打开的新窗体放置在指定的 Panel 容器中。如果没有指定容器名称，则以所在窗体的母窗体为母窗体，新开的窗体为子窗体。

自动弹出：指定当画面切换按钮加载后是否自动打开指定窗体。

将另有一个章节专门说明该控件的使用说明。

3.9 “选择框(Bit_CheckBox)”控件



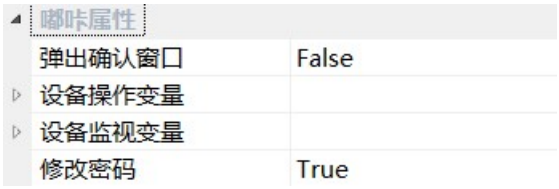
以选择框的方式监控位设备变量

跟嘟咔有关的属性：

设备操作变量：指定操作的设备变量(位类型)。

设备监视变量：指定访问的设备变量(位类型)。

弹出确认窗口：指定操作时是否弹出确认窗口。



3.10 “多功能组合框(Number_ComboBox)”控件

以组合框的方式监控设备变量。

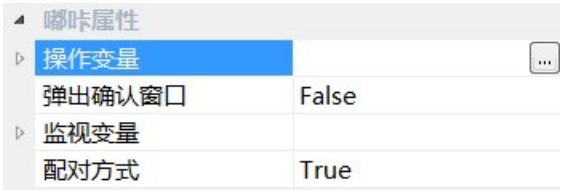
跟嘟咔有关的属性：

操作变量：指定操作的设备变量。

监视变量：指定访问的设备变量。

弹出确认窗口：指定操作时是否弹出确认窗口。

配对方式：组合框与设备变量值的配对方式。True 为组合框索引值跟设备值配对，false 为组合框内容与设置值配对。



3.11 “弹出窗体(Pop_Form)”控件

当满足某一条件时，自动弹出对话框。建议把该控件放置在主窗体中。当程序运行后，控件将自动隐藏。如果把控件放置在没有打开的窗体上，则无效。

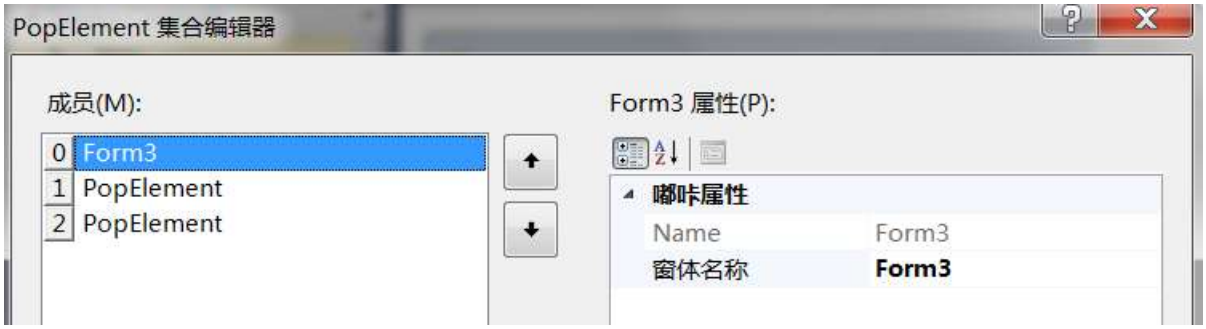
跟嘟咔有关的属性：

设备变量：指定访问的设备变量。

弹出窗口集合：设置条件及打开窗体名称条目。如下图：

窗体名称：指定打开的窗体名称。如果为空，则不动作。

当设备变量值为对应的索引号时，该索引号对应的条目将其作用。



3.12 “进度条(Number_ProgressBar)”控件

以进度条的方式显示设备变量值。

跟嘟咔有关的属性：

设备变量：指定访问的设备变量。

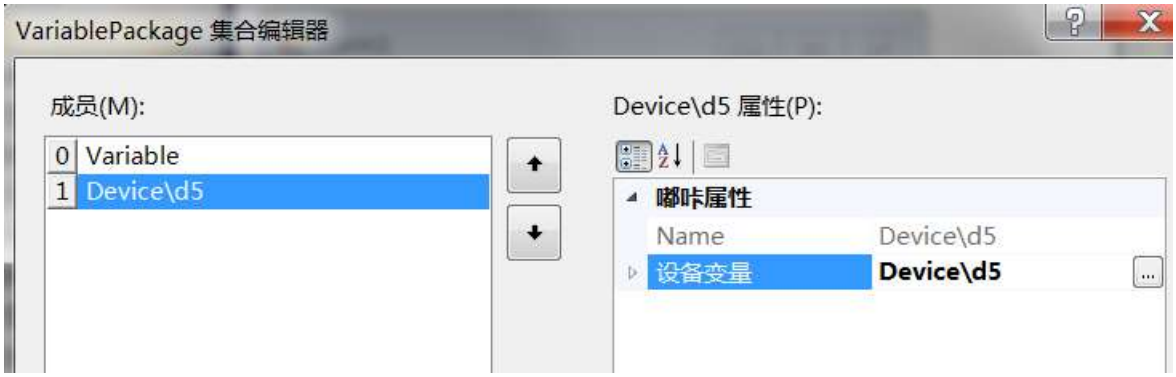
3.13 “实时曲线图表(RealTime_Chart)”控件

以图表的方式显示多个设备变量值。每一个变量值对应一条曲线。

跟嘟咔有关的属性：

刷新时间：指定曲线的刷新时间。

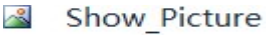
设备变量集合：定义多条曲线。如下图：每一个条目对应一条曲线。



3.14 “饼图(Pie_Chart)”控件

以分割饼的方式显示多个设备变量值相对应的比例。

3.15 “多图片显示(Show_Picture)”控件



继承 Picture 控件，把设备变量值对应一组图片显示在控件上。

跟嘟咔有关的属性：

外观属性展示类：把设备变量值跟一个控件的外观属性对应起来的类。

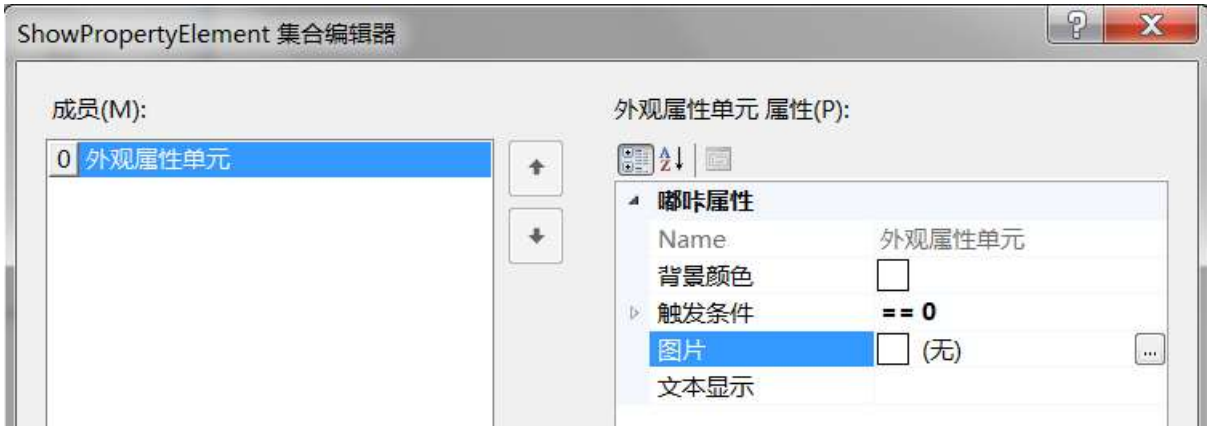
设备变量：指定访问的设备变量。

数字对应属性集合：如下图：当条件满足时，控件的外观将展示相对应的设置。

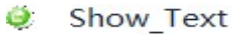
背景颜色：指定控件的背景颜色。

触发条件：指定触发条件。

图片：指定要显示的图片。
文本显示：指定要显示的文本内容。



3.16 “多文本显示(Show_Text)”控件



继承 Label 控件，把设备变量值对应一组文本显示在控件上。

3.17 “日期时间显示(Datetime_Label)”控件

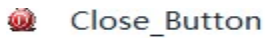
显示当前日期时间控件

跟控件有关的属性：
显示日期：指定是否需要显示日期。False 为只显示当前时间，不显示日期。True 为同时显示日期和时间。

3.18 “数据包存储(DataPackageStorage)”控件

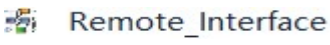
配合 PLC 程序，通过事件包的方式自动存储相关数据，是一种可靠，有效地数据存储方式。

3.19 “关闭按钮(Close_Button)”控件



关闭程序按钮控件。

3.20 “UDP/IP 远程接口(Remote_Interface)”控件



提供 UDP/IP 远程监控设备接口。读写指令格式如下：
读指令：设备名称
写指令：设备名称,值
读写指令的回应：设备名称,值,错误代码
读写成功的话，错误代码为 0。否则为非零

跟嘟咔有关的属性：
端口号：UDP/IP 服务器的端口号。

跟设备有关的事件：
ReceivedCommand：收到一个远程命令时引发的事件。可用该事件把远程命令转换成标准命令。
SendingRespond：回应一个远程命令时引发的事件。可用该事件把标准响应数据转换成用户格式。

注:接口采用”GB2312”编码

3.21 “图片移动(Picture_Moving)”控件

控件的位置坐标”Location”随着设备变量值的变化而改变。

跟嘟咔有关的属性：

横坐标转换：控件在横坐标的描述(X 坐标)

变量起始值：描述物体在设备中的起始位置值。

变量终止值：描述物体在设备中的起始位置值。

像素起始值：控件在界面中的位置通常以像素值来描述。当变量值等于”变量起始值”时，控件的位置对应为该值。

像素终止值：控件在界面中的位置通常以像素值来描述。当变量值等于”变量终止值”时，控件的位置对应为该值。

设备变量：对应的设备变量。变量的类型应为整型。

纵坐标转换：控件在纵坐标的描述(Y 坐标)

变量起始值：描述物体在设备中的起始位置值。

变量终止值：描述物体在设备中的起始位置值。

像素起始值：控件在界面中的位置通常以像素值来描述。当变量值等于”变量起始值”时，控件的位置对应为该值。

像素终止值：控件在界面中的位置通常以像素值来描述。当变量值等于”变量终止值”时，控件的位置对应为该值。

设备变量：对应的设备变量。变量的类型应为整型。

温馨提示：如果多个图片叠加，图片透明部分需要显示底层图片，则需要修改控件的”BackColor”属性为”Transparent”

3.22 “多功能容器 (Panel_Multifunctional)”控件

一个可以拖动一组联动物件随设备变量值移动，隐藏或有效的控件。比如：在设备中，一个平台有许多物件会随着该平台的移动而移动。在软件界面中，Panel_Multifunctional 控件可充当这样的平台，在其上面的图片随着平台的移动而移动。

跟嘟咔有关的属性：

横坐标转换：控件在横坐标的描述(X 坐标)

变量起始值：描述物体在设备中的起始位置值。

变量终止值：描述物体在设备中的起始位置值。

像素起始值：控件在界面中的位置通常以像素值来描述。当变量值等于”变量起始值”时，控件的位置对应为该值。

像素终止值：控件在界面中的位置通常以像素值来描述。当变量值等于”变量终止值”时，控件的位置对应为该值。

设备变量：对应的设备变量。变量的类型应为整型。

纵坐标转换：控件在纵坐标的描述(Y 坐标)

变量起始值：描述物体在设备中的起始位置值。

变量终止值：描述物体在设备中的起始位置值。

像素起始值：控件在界面中的位置通常以像素值来描述。当变量值等于"变量起始值"时，控件的位置对应为该值。

像素终止值：控件在界面中的位置通常以像素值来描述。当变量值等于"变量终止值"时，控件的位置对应为该值。

设备变量：对应的设备变量。变量的类型应为整型。

Enabled：提示是否已启用该控件

True:启用该控件，包括控件里的子控件。

False:不启用该控件，包括控件里的子控件。

Variable：选用一个设备变量来判断该控件是否启用。满足条件为启用，否则为不启用。

Visible：确定该控件是可见的还是隐藏的

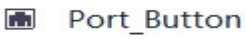
True:该控件为可见的

False:该控件为隐藏的。

Variable：选用一个设备变量来判断该控件是否可见或隐藏。满足条件为可见，否则为隐藏。

温馨提示：如果多个图片叠加，图片透明部分需要显示底层图片，则需要修改控件的"BackColor"属性为"Transparent"

3.23 "端口设置按钮(Port_Button)"控件



设置设备所用端口参数。

跟嘟咔有关的属性：

设备名称：指定要设置端口的设备名称

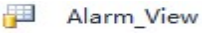
3.24 "仪表显示(Circular_Meter)"控件

以仪表盘的形式显示设备变量值。

跟嘟咔有关的属性：

设备变量：指定访问的设备变量。

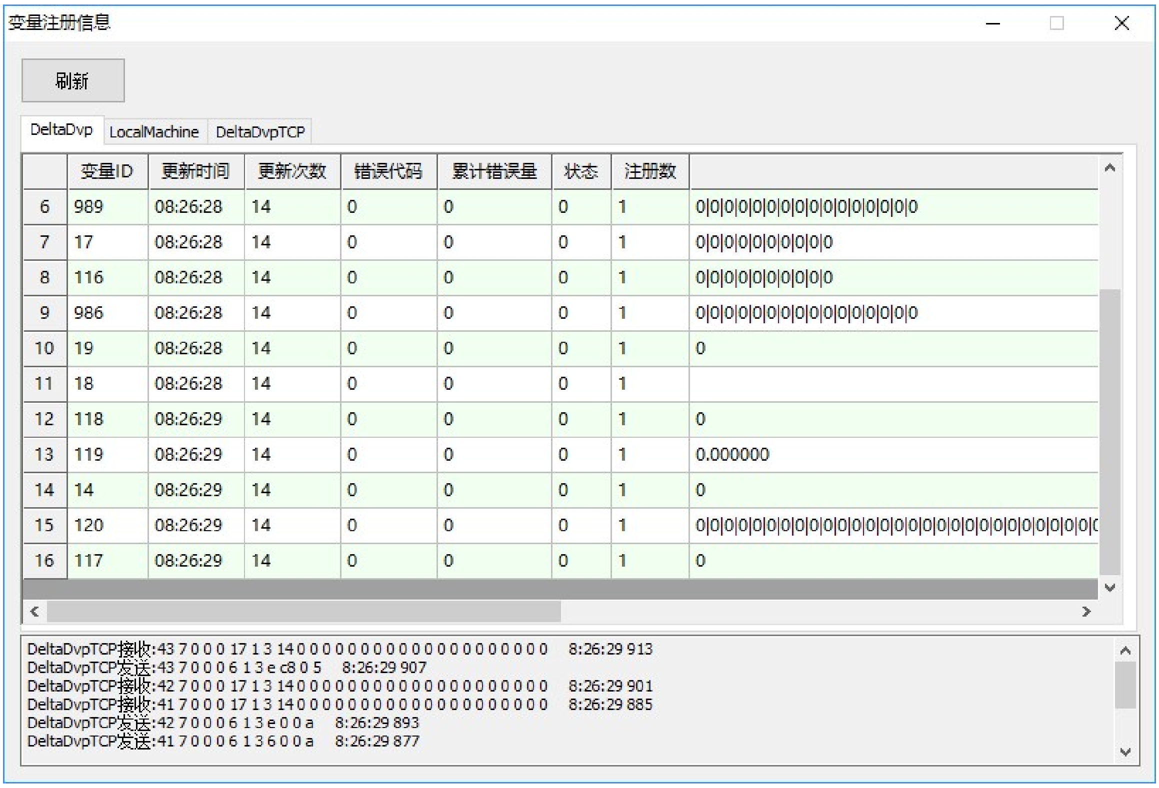
3.25 "报警记录浏览(Alarm_View)"控件



浏览报警记录，配合 Alarm_Label 控件使用。

3.26 “设备变量通讯查看(Register_View)”控件

添加该控件到窗体，运行时点击该控件讲弹出如下统计窗口。



<table>
<tr><th></th><th>变量ID</th><th>更新时间</th><th>更新次数</th><th>错误代码</th><th>累计错误量</th><th>状态</th><th>注册数</th><th></th></tr>
<tr><td>6</td><td>989</td><td>08:26:28</td><td>14</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0</td></tr>
<tr><td>7</td><td>17</td><td>08:26:28</td><td>14</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0|0|0|0|0|0|0|0|0|0</td></tr>
<tr><td>8</td><td>116</td><td>08:26:28</td><td>14</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0|0|0|0|0|0|0|0|0|0</td></tr>
<tr><td>9</td><td>986</td><td>08:26:28</td><td>14</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0</td></tr>
<tr><td>10</td><td>19</td><td>08:26:28</td><td>14</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>11</td><td>18</td><td>08:26:28</td><td>14</td><td>0</td><td>0</td><td>0</td><td>1</td><td></td></tr>
<tr><td>12</td><td>118</td><td>08:26:29</td><td>14</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>13</td><td>119</td><td>08:26:29</td><td>14</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0.000000</td></tr>
<tr><td>14</td><td>14</td><td>08:26:29</td><td>14</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>15</td><td>120</td><td>08:26:29</td><td>14</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0</td></tr>
<tr><td>16</td><td>117</td><td>08:26:29</td><td>14</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
</table>

在该窗口中，每一个设备对应着一个活页。表格中各列的说明如下：

“变量 ID”：设备变量的 ID 号码。

“更新时间”：该变量从设备中最近的一次更新时间点。

“更新次数”：该变量从运行到现在，从设备中读取到的成功次数。

“错误代码”：该变量最近一次读取数据发生的错误代码。

“累计错误量”：该变量从运行到现在，读取数据发生错误的累计数量。

“注册数”：该变量被引用的数量。如果为 0，则为了节省通讯带宽，将不从设备中更新该变量。

“变量值”：该变量的当前值。如果为多个值，则用分隔符隔开。

四，高级应用(功能函数)

嘟咔组态软件与设备(PLC)的通讯采取多线程的方式。每一个设备的读写操作都由一个子线程管理。添加和编辑设备时有对设备详尽的描述。每个设备包含许多的设备变量，每个设备变量在编辑时有详尽的描述。从设备更新的变量值将存储在缓存中。为了节省通讯带宽，并不是所有已编辑的设备变量在运行时都从设备那更新值。

4.1 关于设备的函数

```
/// <summary>
/// 获取项目内所有设备名称
/// </summary>
/// <returns>返回所有设备名称。如果没有，则返回null</returns>
string[]  Dooka.Controls.IO_Servers.GetAllDevicesName()

/// <summary>
/// 查看指定设备是否使用
/// </summary>
/// <param name="DeviceName">设备名称</param>
/// <returns>有效则返回true，否则false</returns>
bool  Dooka.Controls.IO_Servers.IsDeviceEnable(string  DeviceName)

/// <summary>
/// 设置指定设备是否使用。需要重新启动客户程序设置才有效。
/// </summary>
/// <param name="DeviceName">设备名称</param>
/// <param name="Enable">是否有效</param>
void  Dooka.Controls.IO_Servers.SetDeviceEnable(string  DeviceName, bool  Enable)

/// <summary>
/// 获取变量所在的设备名称
/// </summary>
/// <param name="VariableName">变量全称</param>
/// <returns>设备名称</returns>
string  Dooka.Controls.IO_Servers.GetDeviceNameFromVariable (string  VariableName)

/// <summary>
///获取变量所在的设备名称
/// </summary>
/// <param name="ID">变量ID</param>
/// <returns>设备名称</returns>
string  Dooka.Controls.IO_Servers.GetDeviceNameFromVariable(uint  ID)

/// <summary>
/// 唤醒已经超时的设备
/// </summary>
/// <param name="DeviceName">指定要唤醒的设备名称，如果为空，则唤醒所有设备</param>
void  Dooka.Controls.IO_Servers.WakeUpDevice(string  DeviceName)
```

添加设备时，会指定“超时时间”和“恢复时间”。当与设备连续通讯失败时间超过“超时时间”后，将自行关闭与该设备的通讯。等到达“恢复时间”后再尝试与设备通讯。在关闭与该设备的通讯期间，调用该函数将恢复尝试与设备的通讯。(当“超时时间”和“恢复时间”设置值一样时，将时时刻刻尝试与设备的通讯。)

在多个设备共用一个端口时，会关闭通讯有故障的设备以提高端口的利用率。

4.2 关于端口的函数

```
///   <summary>
///   弹出设备所用端口参数设置对话框。重新启动客户程序才有效
///   </summary>
///   <param name="DeviceName">设备名称</param>
/// <returns></returns>
void  Dooka.Controls.IO_Servers.PortSettingDlg(string  DeviceName)

/// <summary>
/// 获取设备所用端口名称
/// </summary>
/// <param name="DeviceName">设备名称</param>
/// <returns>返回端口名称。如果没有，则返回null</returns>
string  Dooka.Controls.IO_Servers.GetPortName(string  DeviceName)
```

4.3 关于项目的函数

```
/// <summary>
/// 获取客户项目语言设置
/// </summary>
/// <returns>语言代码</returns>
 LanguageEncoding  Dooka.Controls.IO_Servers.GetLanguage()

/// <summary>
/// 设置语言
/// </summary>
/// <param name="code">语言代码</param>
void  Dooka.Controls.IO_Servers.  SetLanguage(LanguageEncoding  code)

/// <summary>
/// 获取客户项目的“触摸屏”项设置
/// </summary>
/// <returns> </returns>
bool  Dooka.Controls.IO_Servers.  IsTouchScreen()

/// <summary>
/// 设置“触摸屏”项
/// </summary>
/// <param name="value">设置值</param>
 void  Dooka.Controls.IO_Servers.  SetTouchScreen(bool  value)

/// <summary>
/// 弹出软键盘
/// </summary>
/// <param name="arguments">软键盘弹出位置参数。格式为“x坐标值 y坐标值”</param>
void  Dooka.Controls.IO_Servers.  OpenScreenKeyboard(string  arguments)
```

```
/// <summary>
/// 关闭软键盘
/// </summary>
void  Dooka.Controls.IO_Servers.  CloseScreenKeyboard()
```

4.4 关于设备变量的函数

每一个设备都包含许多个设备变量。在“IO_Servers”→“设备编辑”项中可以添加和编辑变量，每个变量都由详尽的描述。每一个设备变量代表着设备中某一个或多个存储器。在嘟咔组态软件中，每个设备变量都由一个VariableWithIndex类对象管理。客户项目运行时，并不是所有已添加的设备变量都会从设备处读取。从设备中读取的变量值都会存放在缓存中。

每一个设备变量都会有一个ID或全称唯一识别。

ID：一个大于0的整数。

全称：类似文件路径的字符串，由’\’隔开。如：”KV1000\测试数据操作\数据包”。最前面部分为设备名称‘KV1000’。后面部分为变量名称‘数据包’。

全称扩展名：全称后面含有‘[n]’样式的。如：”KV1000\测试数据操作\数据包[2]”。其中‘[2]’代表在该变量众多值中的一个值的索引号2(0为起始号)。

```
/// <summary>
/// 从设备变量全称中获取变量ID
/// </summary>
/// <param name="VariableName">变量全称</param>
/// <returns>返回变量ID。如果不存在该变量，则返回0</returns>
uint  Dooka.Controls.IO_Servers.  VariableNameToId(string  VariableName)

/// <summary>
/// 从设备变量ID中获取变量全称
/// </summary>
/// <param name="ID">变量ID</param>
/// <returns>返回变量全称。如果不存在该ID，则返回空字符串</returns>
string  Dooka.Controls.IO_Servers.  VariableIdToName(uint  ID)

/// <summary>
/// 获取设备变量的数据类型。一个变量可以包含许多不同数据类型的数据。
/// </summary>
/// <param name="ID">变量ID</param>
/// <param name="Index">值索引号</param>
/// <returns>返回数据类型。如果没有，则返回0</returns>
uint  Dooka.Controls.IO_Servers.  GetVariableDataType(uint  ID, uint  Index)

/// <summary>
/// 获取设备变量的值数量
/// </summary>
/// <param name="ID">变量ID</param>
/// <returns>返回值数量</returns>
uint  Dooka.Controls.IO_Servers.  GetVariableAccessCount(uint  ID)
```

```
/// <summary>
/// 获取变量的只读属性
/// </summary>
/// <param name="ID">变量ID</param>
/// <returns>如果为只读，则返回true。否则返回false</returns>
Bool  Dooka.Controls.IO_Servers.GetVariableReadOnly(uint  ID)

/// <summary>
/// 异步写变量的值。由于跟设备的通讯有延迟性。该函数为异步操作，执行该函数将立刻返回，对
/// 变量的操作将在后台排队执行。在恶劣的通讯环境下，该函数不能保证对变量成功地操作。
/// </summary>
/// <param name="ID">变量ID</param>
/// <param name="Index">值索引号。如果变量只有一个值或数据类型为字符串，则忽略该参数
</param>
/// <param name="Value">值</param>
/// <returns>返回出错信息，否则返回空</returns>
string  Dooka.Controls.VariableWithIndex.WriteValue(uint  ID, uint  Index, string
Value)

/// <summary>
///异步写变量的值。由于跟设备的通讯有延迟性。该函数为异步操作，执行该函数将立刻返回，对
/// 变量的操作将在后台排队执行。在恶劣的通讯环境下，该函数不能保证对变量成功地操作。
/// </summary>
/// <param name="ID">设备ID</param>
/// <param name="Values">值数组</param>
/// <returns>返回出错信息，否则返回空</returns>
string  Dooka.Controls. VariableWithIndex.WriteValue(uint  ID, string[]  Values)

/// <summary>
///异步写变量的值。由于跟设备的通讯有延迟性。该函数为异步操作，执行该函数将立刻返回，对
/// 变量的操作将在后台排队执行。在恶劣的通讯环境下，该函数不能保证对变量成功地操作。
/// </summary>
/// <param name="VariableNameEx">变量全称扩展名</param>
/// <param name="Value">值</param>
/// <returns>返回出错信息，否则返回空</returns>
string  Dooka.Controls. VariableWithIndex.WriteValue(string  VariableNameEx, string
Value)

/// <summary>
/// 从缓存中获取设备变量的值。
/// </summary>
/// <param name="ID">变量ID</param>
/// <returns>返回变量的值数组。如果该变量还没有从设备中更新，则返回空字符串</returns>
string[]  Dooka.Controls. VariableWithIndex.GetVariableValue(uint  ID)

/// <summary>
/// 从缓存中获取设备变量的值。
/// </summary>
```

```
/// <param name="VariableNameEx">变量全称扩展名</param>
/// <returns>返回变量的值。如果该变量还没有从设备中更新，则返回空字符串</returns>
string  Dooka.Controls.VariableWithIndex.GetVariableValue(string  VariableNameEx)

/// <summary>
///从缓存中获取设备变量的值
/// </summary>
/// <param name="VariableName">变量全称</param>
/// <returns>返回变量的值数组。如果该变量还没有从设备中更新，则返回空字符串</returns>
string[]  Dooka.Controls.VariableWithIndex.GetVariableValues(string  VariableName)

/// <summary>
/// 由变量全称扩展名分解成变量名称和索引值
/// </summary>
/// <param name="NameEx">变量全称扩展名</param>
/// <param name="Name">变量全称</param>
/// <param name="Index">索引值</param>
void  Dooka.Controls.VariableWithIndex.ExplainNameAndIndex(string NameEx, ref string Name,
ref uint Index)

/// <summary>
/// 动态设置一个VariableWithIndex对象的变量名称
/// </summary>
/// <param name=" VariableName ">变量全称</param>
void  对象.SetName(string  VariableName)

/// <summary>
/// 动态设置一个VariableWithIndex对象的变量ID
/// </summary>
/// <param name=" id ">变量ID</param>
void  对象.SetName(uint  id)

/// <summary>
/// 同步读取变量值
/// </summary>
/// <param name="VariableName">变量名称</param>
/// <param name="pErrorCode">返回错误码。非零为错误</param>
/// <returns>返回变量值</returns>
string[] Dooka.Controls.IO_Servers.SyncVariableRead(string VariableName, ref int pErrorCode)

/// <summary>
/// 同步改写变量值
/// </summary>
/// <param name="VariableName">变量名称</param>
/// <param name="arrValues">变量值</param>
/// <returns>错误码。0为写成功，非0为写错误</returns>
int  Dooka.Controls.IO_Servers.SyncVariableWrite(string VariableName, string[] arrValues)
```

```
/// <summary>
/// 同步改写变量值
/// </summary>
/// <param name="VariableNameEx">变量全称扩展名。如果没有扩展名，则默认为[0]</param>
/// <param name="Value">变量值</param>
/// <returns>错误码。0为写成功，非0为写错误</returns>
int  Dooka.Controls.IO_Servers.SyncVariableWrite(string VariableNameEx, string Value)

///    <summary>
///    异步读取变量的值。当从设备中更新变量值后，将执行回调函数。
///    </summary>
///    <param name="VariableName">变量全称</param>
///    <param name="CallBackMark">回调标志，该值会返回到回调函数参数中</param>
///    <param name="pro">回调函数，原型 void pro(Dooka.Controls .VariableOperateEventArgs
e)</param>
/// <returns></returns>
void  Dooka.Controls.IO_Servers.AsynReadVariable(string  VariableName,uint
CallBackMark, VariableOperateCallBack  pro)

Dooka.Controls .VariableOperateEventArgs 类
ErrorCode：错误代码，0为正常无错。MarkOrID：回调标志或变量ID号。
Values：变量值数组
///    <summary>
///    异步读取变量的值。当从设备中更新变量值后，将执行回调函数。
///    </summary>
///    <param name="ID">变量ID</param>
///    <param name="pro">回调函数，原型 void pro(Dooka.Controls .VariableOperateEventArgs
e)</param>
/// <returns></returns>
void  Dooka.Controls.IO_Servers.AsynReadVariable(uint  ID, VariableOperateCallBack
pro)

///    <summary>
///    异步写变量的值。
///    </summary>
///    <param name="VariableNameEx">变量全称扩展名。如果没有扩展名，则默认为[0]</param>
///    <param name="Value">变量的值</param>
///    <param name="CallBackMark">回调标志，该值会返回到回调函数参数中</param>
///    <param name="pro">回调函数，原型 void pro(Dooka.Controls .VariableOperateEventArgs
e)</param>
/// <returns>返回出错信息。否则返回空字符串</returns>
string  Dooka.Controls.IO_Servers.AsynWriteVariable(string  VariableNameEx, string
Value, uint  CallBackMark, VariableOperateCallBack  pro)

///    <summary>
///    异步写变量的值。
```

```
///    </summary>
///    <param name="VariableName">变量全称</param>
///    <param name="Values">变量值数组</param>
///    <param name="CallBackMark">回调标志，该值会返回到回调函数参数中</param>
///    <param name="pro">回调函数，原型 void pro(Dooka.Controls .VariableOperateEventArgs
e)</param>
/// <returns>返回出错信息。否则返回空字符串</returns>
string  Dooka.Controls.IO_Servers.AsynWriteVariable(string  VariableName, string[]
Values, uint  CallBackMark, VariableOperateCallBack  pro)

///    <summary>
///    异步写变量的值。
///    </summary>
///    <param name="ID">变量ID</param>
///    <param name="Values">变量值数组</param>
///    <param name="CallBackMark">回调标志，该值会返回到回调函数参数中</param>
///    <param name="pro">回调函数，原型 void pro(Dooka.Controls .VariableOperateEventArgs
e)</param>
/// <returns>返回出错信息。否则返回空字符串</returns>
string  Dooka.Controls.IO_Servers.AsynWriteVariable(uint  ID, string[]  Values, uint
CallBackMark, VariableOperateCallBack  pro)
```

为了节省通讯带宽，并不是所有已添加的设备变量都会从设备处更新。嘟咔组态组件会选择性地循环从设备处更新部分变量值。客户程序运行时，如果一个窗体没有打开，则该窗体上的嘟咔控件所引用的设备变量将不会被嘟咔组态组件自动更新。

设备变量可以通过手动注册让其循环从设备处更新。通过注销让其停止更新，节省通讯带宽。

```
/// <summary>
/// 监控设备变量。当每次从设备更新该变量后，将执行回馈函数pro(uint  ID)。
/// 如果不再需要监控，请使用StopMonitorVariable函数卸下监控
/// </summary>
/// <param name="ID">变量ID</param>
/// <param name="pro">回馈函数。原型: void pro(uint  ID)</param>
void  Dooka.Controls.IO_Servers.StartMonitorVariable(uint  ID, VariableUpdateCallBack
pro)
```

回馈函数中的参数ID表示更新的变量ID。通过该ID，可以获取变量的所有细节。

```
/// <summary>
/// 监控设备变量。当每次从设备更新该变量后，将执行回馈函数pro(uint  ID)。
/// 如果不再需要监控，请使用StopMonitorVariable函数卸下监控
/// </summary>
/// <param name="VariableName">变量全称</param>
/// <param name="pro">回馈函数。原型:void pro(uint  ID)</param>
void  Dooka.Controls.IO_Servers.StartMonitorVariable(string  VariableName,
VariableUpdateCallBack  pro)
```

回馈函数中的参数ID表示更新的变量ID。通过该ID，可以获取变量的所有细节。

```
/// <summary>
/// 卸下监控设备变量。跟StartMonitorVariable函数配对使用。
/// </summary>
/// <param name="ID">变量ID</param>
/// <param name="pro">回馈函数。当为null时，卸下整个对该变量的监控。原型:pro(uint ID)</param>
void  Dooka.Controls.IO_Servers.StopMonitorVariable(uint  ID, VariableUpdateCallBack pro)
```

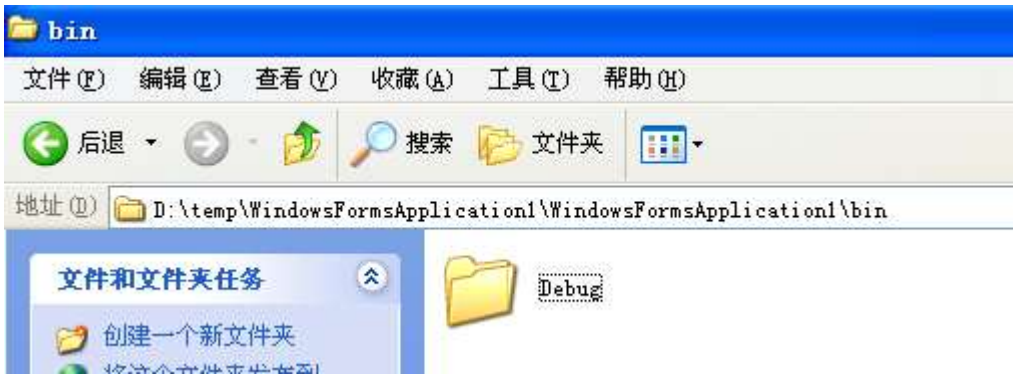
```
/// <summary>
/// 卸下监控设备变量。跟StartMonitorVariable函数配对使用。
/// </summary>
/// <param name="VariableName">变量全称</param>
/// <param name="pro">回馈函数。当为null时，卸下整个对该变量的监控。原型:pro(uint ID)</param>
void  Dooka.Controls.IO_Servers.StopMonitorVariable(string VariableName, VariableUpdateCallBack  pro)

/// <summary>
/// 监控设备变量值是否改变。当每次检测到变量值改变后，将执行回馈函数pro(uint ID,string OldValue,string CurrentValue)。如果不再监控，请使用StopMonitorVariable函数卸下监控。
/// </summary>
/// <param name="ID">要监控的设备变量ID</param>
/// <param name="pro">回馈函数。原型:pro(uint ID,string OldValue,string CurrentValue)</param>
public static void StartMonitorVariableChange(uint ID,VariableChangeCallBack  pro)

/// <summary>
/// 监控设备变量值是否改变。当每次检测到变量值改变后，将执行回馈函数。如果不再监控，请使用StopMonitorVariable函数卸下监控。
/// </summary>
/// <param name="VariableName">要监控的设备变量名称</param>
/// <param name="pro">回馈函数。原型:pro(uint ID,string OldValue,string CurrentValue)</param>
public static void StartMonitorVariableChange(string VariableName, VariableChangeCallBack pro)
```

五,调试，运行

如需更详细说明，请参看微软的 MSDN。项目编辑完成后，在“解决方案资源管理器”中右击项目。在弹出的窗口中选择“调试” -> “启动新实例”。或参考 2.1.6 节

六,发布

项目开发并测试完成后。在项目的保存路径中有一个“Debug”文件夹，把整个“Debug”拷贝到用户电脑中，然后运行该文件夹内的 EXE 文件即可。如果 EXE 文件运行不了，很大原因说明用户电脑没有安装相对应的 Microsoft.NET Framework 框架，则需安装框架。（用户电脑无需安装 VS 和 Dooka 组态控件）



七,授权

在发布“..\Debug”文件夹中运行“激活.exe”文件。如下图：

两种激活方式：(一旦注册，本设备将一直有效，重装系统，程序更新也不会影响)

1， 把激活界面生成的“注册机”发给我们，我们将根据该“注册机”产生一个注册码返还给您，将注册码输进去点“注册”按钮即可弹出注册成功信息。(“激活”窗口的右边二维码含有注册机信息。可用手机扫描该二维码得到注册机发送给我们。)

2， “非商业用处”：单击”注册”直接在线注册。

八,更新

嘟咔组态控件的设备驱动和控件的更新请登录 http://www.auto-kk.com/

如果没有你需要连接的设备的驱动，或在使用过程中有什么问题。请联系我们。

联系方式：

电话：13424297133
微信号：13424297133
QQ: 13728811