DAME3000N驱动程序使用说明书

请您务必阅读《使用纲要》，他会使您事半功倍!

目　　录

目录

提醒用户：

通常情况下，WINDOWS 系统在安装时自带的 DLL 库和驱动不全，所以您不管使用那种语言编程，请您最好先安装上Microsoft Visual Studio 2005版本的软件，方可使我们的驱动程序有更完备的运行环境。

有关设备驱动安装和产品二次发行请参考 DAME3000N.doc 文档。

第一章 版权信息与命名约定

第一节、版权信息

第二节、命名约定

一、为简化文字内容，突出重点，本文中提到的函数名通常为基本功能名部分，其前缀设备名如DAME3000Nxxxx_则被省略。如DAME3000N_CreateDevice 则写为CreateDevice。二、函数名及参数中各种关键字缩写规则

缩写	全称	汉语意思	缩写	全称	汉语意思
Dev	Device	设备	DI	Digital Input	数字量输入
Pro	Program	程序	DO	Digital Output	数字量输出
Int	Interrupt	中断	CNT	Counter	计数器
Dma	Direct Memory Access	直接内存存取	DA	Digital convert to Analog	数模转换
AD	Analog convert to Digital	模数转换	DI	Differential	(双端或差分) 注：在常量选项中
Npt	Not Empty	非空	SE	Single end	单端
Para	Parameter	参数	DIR	Direction	方向
SRC	Source	源	ATR	Analog Trigger	模拟量触发
TRIG	Trigger	触发	DTR	Digital Trigger	数字量触发
CLK	Clock	时钟	Cur	Current	当前的
GND	Ground	地	OPT	Operate	操作
Lgc	Logical	逻辑的	ID	Identifier	标识
Phys	Physical	物理的			

以上规则不局限于该产品。

第二章 DAME3000N设备函数接口介绍

第一节、设备驱动接口函数列表（每个函数省略了前缀“DAME3000N_”）

函数名	函数功能	备注
① 设备对象操作函数		
CreateDevice	创建设备对象	
ReleaseDevice	释放设备对象	
GetNetworkConfig	获得设备的网络配置信息	
SetNetworkConfig	设置设备的网络配置信息	

使用需知

Visual C++ & C++Builder：

首先将DAME3000N.h和 DAME3000N.lib两个驱动库文件从相应的演示程序文件夹下复制到您的源程序文件夹中，然后在您的源程序头部添加如下语句，以便将驱动库函数接口的原型定义信息和驱动接口导入库(DAME3000N.lib)加入到您的工程中。

#include “DAME3000N.H”

在 VC 中，为了使用方便，避免重复定义和包含，您最好将以上语句放在StdAfx.h 文件。一旦完成了以上工作，那么使用设备的驱动程序接口就跟使用VC/C++Builder 自身的各种函数，其方法一样简单，毫无二别。

关于DAME3000N.h 和 DAME3000N.lib 两个文件均可在演示程序文件夹下面找到。

Visual Basic:

首先将DAME3000N.Bas 驱动模块头文件从 VB 的演示程序文件夹下复制到您的源程序文件夹中，然后将此模块文件加入到您的 VB 工程中。其方法是选择 VB 编程环境中的工程(Project)菜单，执行其中的"添加模块"(Add Module)命令,在弹出的对话中选择DAME3000N.Bas 模块文件即可，一旦完成以上工作后，那么使用设备的驱动程序接口就跟使用 VB 自身的各种函数，其方法一样简单，毫无二别。

请注意，因考虑Visual C++和 Visual Basic 两种语言的兼容问题，在下列函数说明和示范程序中，所举的Visual Basic 程序均是需要编译后在独立环境中运行。所以用户若在解释环境中运行这些代码，我们不保证能完全顺利运行。

LabView / CVI:

LabVIEW 是美国国家仪器公司(National Instrument)推出的一种基于图形开发、调试和运行程序的集成化环境，是目前国际上唯一的编译型的图形化编程语言。在以 PC 机为基础的测量和工控软件中，LabVIEW 的市场普及率仅次于C++/C 语言。LabVIEW 开发环境具有一系列优点，从其流程图式的编程、不需预先编译就存在的语法检查、调试过程使用的数据探针，到其丰富的函数功能、数值分析、信号处理和设备驱动等功能，都令人称道。关于LabView/CVI 的驱动程序接口的详细说明请参考其演示源程序。

C#:

首先将DAME3000N. cs驱动模块头文件从 C#的演示程序文件夹下复制到您的源程序文件夹中，然后将此文件加入到您的 C#工程中。其方法是C#编程环境中的工程(Project)菜单，执行其中的"添加现有项"命令,在弹出的对话框中选择DAME3000N.cs文件即可，一旦完成以上工作后，那么使用设备的驱动程序接口就跟使用 C#自身的各种函数，其方法一样简单，毫无二别。

VB.NET:

首先将DAME3000N. vb驱动模块头文件从VB.NET 的演示程序文件夹下复制到您的源程序文件夹中，然后将此文件加入到您的 VB.NET工程中。其方法是VB.NET编程环境中的工程(Project)菜单，执行其中的"添加现有项"命令,在弹出的对话框中选择DAME3000N.vb文件即可，一旦完成以上工作后，那么使用设备的驱动程序接口就跟使用 VB.NET自身的各种函数，其方法一样简单，毫无二别。

第二节、设备对象管理函数原型说明

♦ 创建设备对象函数

函数原型：

Visual C++ & C++ Builder：

```
HANDLE CreateDevice(char[]szIP,LONG LPort, LONG LSendTimeout,LONG LRcvTimeout)
```

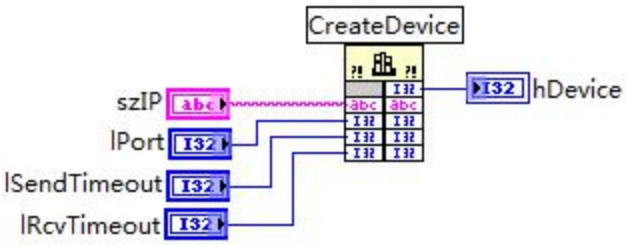
C#：

```
public static extern IntPtrCreateDevice(Char[] szIP,Int32 lPort, Int32 lSendTimeout,Int32 lRcvTimeout);
```

Visual Basic：

```
Declare Function CreateDevice Lib "DAME3000N" (byval szIP as string, _
                                               byval LPort as Long, _
                                               byval LSendTimeout as Long, _
                                               byval  LRcvTimeout as Long) As Long
```

LabView:



功能：该函数负责创建设备对象，并返回其设备对象句柄。

参数：

szIP设备IP地址。当向同一个Windows 系统中加入若干相同类型的设备时，系统将以该设备的“基本名称”与 IP地址名称后缀的标识符来确认和管理该设备。比如若用户往Windows 系统中加入一个DAME3000N模板时，系统则以“DAME3000N”作为基本名称，再以 IP的初值组合成该设备的标识符“DAME3000N-IP”来确认和管理这一个设备。若再添加，则以此类推;

lPort，是TCP端口;

lSendTimeout，发送数据的超时间隔;

lRcvTimeout，接收数据的超时间隔;

返回值：如果执行成功，则返回设备对象句柄hDevice;如果没有成功，则返回错误码INVALID_HANDLE_VALUE。由于此函数已带容错处理，即若出错，它会自动弹出一个对话框告诉您出错的原因。您只需要对此函数的返回值作一个条件处理即可，别的任何事情您都不必做。

备注：创建完成后，如果释放 Device需要用DAME3000N_ReleaseDevice 来关闭。

相关函数：ReleaseDevice

♦ **释放设备对象所占的系统资源及设备对象**

函数原型：

Visual C++ & C++Builder:

BOOL ReleaseDevice(HANDLE hDevice)

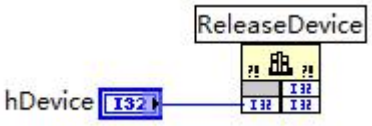
Visual Basic:

Declare Function ReleaseDevice Lib "DAME3000N" (ByVal hDevice As Long) As Boolean

C#:

public static extern Int32 ReleaseDevice(IntPtr hDevice);

LabView:



功能：释放设备对象所占用的系统资源及设备对象自身。

参数：hDevice设备对象句柄，它应由CreateDevice创建。

返回值：若成功，则返回TRUE, 否则返回 FALSE, 用户可以用 GetErrInfo 捕获错误码。

相关函数：CreateDevice

应注意的是，CreateDevice必须和ReleaseDevice函数一一对应，即当您执行了一次CreateDevice，再一次执行这些函数前，必须执行一次ReleaseDevice函数，以释放由CreateDevice占用的系统软硬件资源，如系统内存等。只有这样，当您再次调用CreateDevice函数时，那些软硬件资源才可被再次使用。

♦ **复位**

DAME3000N卡

◆ 函数原型：

Visual C++ & C++Builder:

BOOL DeviceReset (HANDLE hDevice)

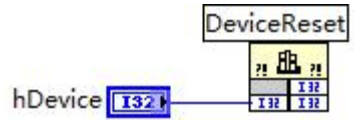
Visual Basic:

Declare Function DeviceReset Lib "DAME3000N" (ByVal hDevice As Long) As Boolean

C#:

public static extern Int32DeviceReset (IntPtr hDevice);

LabView:



功能：复位整个设备，相当于它与PC机端重新建立。其效果与重新插上设备等同。一般在出错情况下，想软复位来建决重连接问题，就可以调用该函数解决此问题。

参数：hDevice设备对象句柄，它应由CreateDevice创建。由它指向要复位的设备。

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数：CreateDevice，ReleaseDevice

◆获取DAME3000N设备的网络配置信息：

函数原型：

Visual C++ & C++Builder:

BOOL GetNetWorkConfig (HANDLE hDevice,PDEVICE_NET_INFO pNetInfo)

Visual Basic:

Declare Function GetNetWorkConfig Lib "DAME3000N" (ByVal hDevice As Long,_

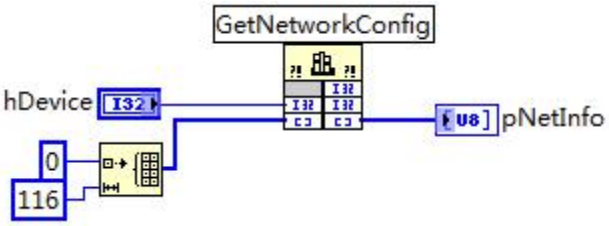
ByRef pNetInfo As PDEVICE_NET_INFO)As Boolean

C#:

public static extern Int32 GetNetworkConfig(IntPtr hDevice,

ref _DEVICE_NET_INFO pNetInfo);

LabVIEW：



功能：列表系统中DAME3000N的网络配置信息。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pNetInfo设备网络信息。

返回值：若成功，则弹出对话框控件列表所有 DAME3000N设备的网络配置情况。

相关函数：CreateDevice，ReleaseDevice

◆ 设置DAME3000的网络配置信息

函数原型:

Visual C++ & C++Builder：

BOOL SetNetWorkConfig (HANDLE hDevice,PDVICE_NET_INFO pNetInfo)

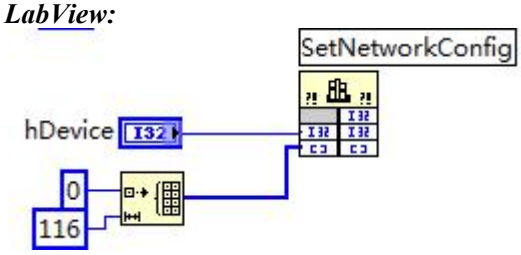
Visual Basic:

```
Declare Function SetNetWorkConfig Lib "DAME3000N" (ByVal hDevice As Long,_
                                                   ByRef pNetInfo As PDEVICE_NET_INFO )As Boolean
```

C#：

```
public static extern Int32 SetNetworkConfig(IntPtr hDevice,
                                            ref _DEVICE_NET_INFO pNetInfo);
```

LabView:



功能：它负责初始化设备对象，为设备操作就绪有关工作。

参数：

hDevice设备对象句柄,它应由CreateDevice创建。

pNetInfo设备网络信息。

返回值：如果初始化设备对象成功,则返回 TRUE。否则返回 FALSE。

相关函数: CreateDevice，ReleaseDevice

第三节、AI操作函数原型说明

♦获得模拟量通道输入值

函数原型：

Visual C++ & C++Builder:

```
BOOL AIReadData (HANDLEhDevice,
                 UNSHORT  lAIValue[],
                 LONG lFirstChannel=0,
                 LONG lLastChannel=0)
```

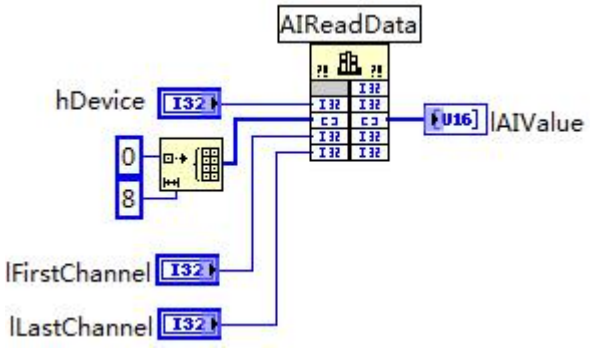
Visual Basic:

```
Declare Function AIReadData Lib "DAME3000N" (ByVal hDevice As Long, _
                                             ByRef lAIValue As Long, _
                                             ByVal lFirstChannel As Long,_
                                             ByVal lLastChannel As Long) As Boolean
```

C#：

```
public static extern Int32 AIReadData(IntPtr hDevice,
                                      UInt16  []lAIValue,
                                      Int32  lFirstChannel,
                                      Int32  lLastChannel)
```

LabView:



功能：读 AI通道内容。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。
lAIValue 返回的AI数据。
lFirstChannel 首通道号
lLastChannel 末通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数：CreateDevice，ReleaseDevice

♦设置模拟量输入模式

函数原型：

Visual C++ & C++Builder:

```
BOOL AISetRange(HANDLEhDevice, LONG     lAIMode[],LONG lFirstChannel,LONG lLastChannel)
```

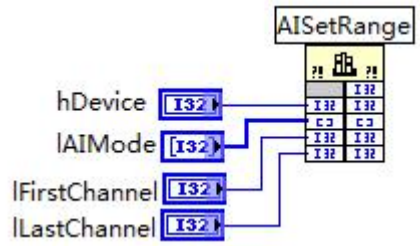
Visual Basic:

```
Declare Function AISetRange"DAME3000N" (ByVal hDevice As Long,_
                                        ByRef lAIMode AsLONG,_
                                        ByVal  lFirstChannel As LONG , _
                                        ByVal   lLastChannel As LONG) As Boolean
```

C#:

```
public static extern Int32 AISetRange(IntPtr hDevice,
                                      Int32  []lAIMode ,
                                             Int32  lFirstChannel,
                                             Int32  lLastChannel)
```

LabView:



功能：设置模拟量输入模式。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。
lAMode AI输入模式。
lFirstChannel 首通道号
lLastChannel 末通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数：CreateDevice，ReleaseDevice

第四节、DO操作函数原型说明

♦设置DO输出模式

函数原型：

Visual C++ & C++Builder:

```
BOOL DOSetMode(HANDLEhDevice,
                        LONG Mode[],
                        LONG lFirstChannel,
```

LONG lLastChannel)

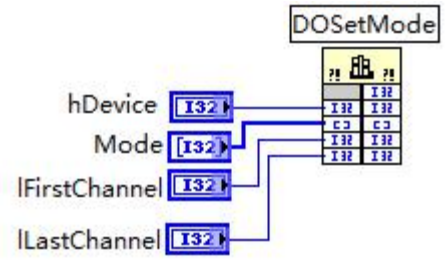
Visual Basic:

```
Declare Function DOSetMode"DAME3000N" (ByVal hDevice As Long,_
                                       ByRef  Mode AsLONG,_
                                       ByVal  lFirstChannel As LONG , _

                                       ByVal   lLastChannel As LONG) As Boolean
```

C#：

```
public static extern Int32 DOSetMode (IntPtr hDevice,
                                      Int32  []Mode ,
                                      Int32  lFirstChannel,
                                      Int32  lLastChannel)
```

LabView:



功能：设置DO模式输出状态。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。
Mode DO输出模式。
lFirstChannel 首通道号
lLastChannel 末通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数：CreateDevice，ReleaseDevice

♦设置DO输出值

函数原型：

Visual C++ & C++Builder:

```
BOOL DOSetValue(HANDLEhDevice,
                Byte byDOSts[],
                LONG lFirstChannel,
                LONG lLastChannel)
```

Visual Basic:

```
Declare Function DOSetValue"DAME3000N" (ByVal hDevice As Long,_
                                        ByRef  byDOSts AsByte ,_
                                        ByVal  lFirstChannel As LONG , _
                                        ByVal   lLastChannel As LONG) As Boolean
```

C#：

```
public static extern Int32 DOSetValue(IntPtr hDevice,
                                       Byte []byDOSts ,
                                       Int32  lFirstChannel,
                                       Int32  lLastChannel)
```

LabView:

请参考相关演示程序。

功能：设置DO模式输出值。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

byDOSts输出模式。

lFirstChannel 首通道号

lLastChannel 末通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数：CreateDevice，ReleaseDevice

♦设置DO输出值(单通道设置)

函数原型：

Visual C++ & C++Builder:

```
BOOL DOSetValue_Line(HANDLEhDevice,
                     BOOL byDOSts,
                     LONG lChannel)
```

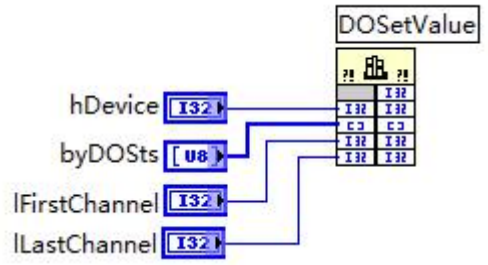
Visual Basic:

```
Declare Function DOSetValue_Line"DAME3000N" (ByVal hDevice As Long,_
                                             ByVal  byDOSts AsBOOL ,_
                                             ByVal  lChannel As LONG) As Boolean
```

C#:

```
public static extern Int32 DOSetValue_Line(IntPtr hDevice,
                                           BOOL byDOSts ,
                                           Int32  lChannel)
```

LabView:



功能：设置DO模式输出值。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

byDOSts输出模式。

lChannel通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数：CreateDevice，ReleaseDevice

♦设置脉冲输出占空比

函数原型：

Visual C++ & C++Builder:

```
BOOL DOSetPulseWidth(HANDLEhDevice,
                     ULONG lLowWidth[],
                     ULONG lHighWidth[],
                     LONG lFirstChannel=0,
```

LONG lLastChannel=0)

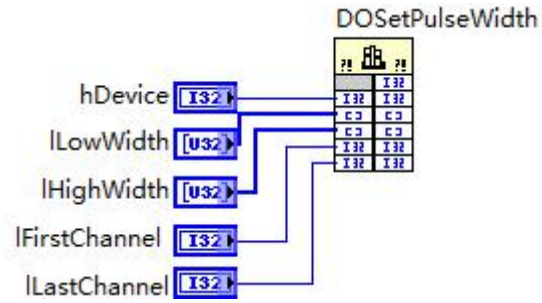
Visual Basic:

```
Declare Function DOSetPulseWidth"DAME3000N" (ByVal hDevice As Long,_
                                             ByRef lLowWidth AsLong,_
                                             ByRef lHighWidth As Long,_
                                             ByVal  lFirstChannel As Long,_
                                             ByVal  lLastChannel As Long) As Boolean
```

C#：

```
public static extern Int32 DOSetPulseWidth(IntPtr hDevice,
                                           Int32  []lLowWidth,
                                           Int32  []lHighWidth,
                                           Int32  lFirstChannel,
                                           Int32  lLastChannel)
```

LabView:



功能：设置脉冲输出占空比。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。
lLowWidth 低电平宽度。
lHighWigth 高电平宽度
lFirstChannel 首通道号
lLastChannel 末通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数：CreateDevice，ReleaseDevice

♦设置固定脉冲输出

函数原型：

Visual C++ & C++Builder:

```
BOOL DOSetPulseOutputCount(HANDLEhDevice,
                           ULONG PuOutputNum[],
                           LONG lFirstChannel=0,
                           LONG lLastChannel=0)
```

Visual Basic:

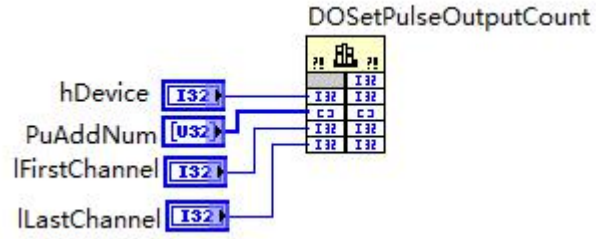
```
Declare Function DOSetPulseOutputCount"DAME3000N" (ByVal hDevice As Long,_
                                                   ByRef PuOutputNum As Long,_
                                                   ByVal  lFirstChannel As Long,_
                                                   ByVal  lLastChannel As Long) As Boolean
```

C#：

```
public static extern Int32 DOSetPulseOutputCount(IntPtr hDevice,
                                                 Int32  []PuOutputNum ,
                                                 Int32  lFirstChannel,
```

Int32 lLastChannel)

LabView:



功能：设置固定脉冲输出。（0x00时连续脉冲输出）

参数：

hDevice设备对象句柄，它应由CreateDevice创建。
PuOutputNum 脉冲个数
lFirstChannel 首通道号
lLastChannel 末通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数：CreateDevice，ReleaseDevice

♦ **设置由低到高输出延时时间**

函数原型：

Visual C++ & C++Builder:

```
BOOL DOSetLowToHighTime(HANDLEhDevice,
                        ULONG DleayTime[],
                        LONG lFirstChannel,
                        LONG lLastChannel)
```

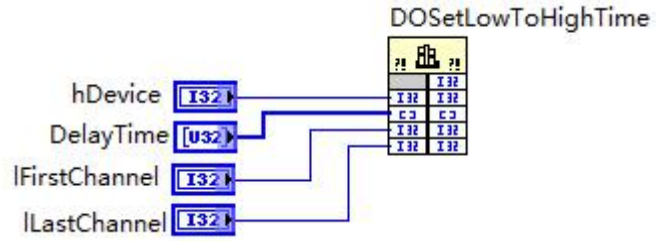
Visual Basic:

```
Declare Function DOSetLowToHighTime"DAME3000N" (ByVal hDevice As Long,_
                                                ByRef DleayTime As Long,_
                                                ByVal  lFirstChannel As Long,_
                                                ByVal  lLastChannel As Long) As Boolean
```

C#:

```
public static extern Int32 DOSetLowToHighTime(IntPtr hDevice,
                                              UInt32  []DleayTime,
                                              Int32  lFirstChannel,
                                              Int32  lLastChannel)
```

LabView:



功能：设置低到高输出延时时间。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

DelalyTime 延时时间
lFirstChannel 首通道号
lLastChannel 末通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数： CreateDevice，ReleaseDevice

♦ **设置由高到低输出延时时间**

函数原型：

Visual C++ & C++Builder:

```
BOOL DOSetHighToLowTime(HANDLEhDevice,
                        ULONG DleayTime[],
                        LONG lFirstChannel,
                        LONG lLastChannel)
```

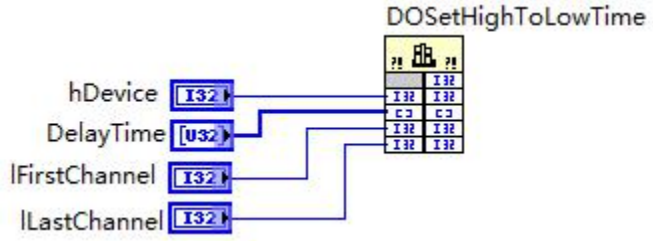
Visual Basic:

```
Declare Function DOSetHighToLowTime"DAME3000N" (ByVal hDevice As Long,_
                                                ByRef DleayTime As Long,_
                                                ByVal  lFirstChannel As Long,_
                                                ByVal  lLastChannel As Long) As Boolean
```

C#:

```
public static extern Int32 DOSetHighToLowTime(IntPtr hDevice,
                                               UInt32  []DleayTime ,
                                               Int32  lFirstChannel,
                                               Int32  lLastChannel)
```

LabView:



功能：设置高到底输出延时时间

参数：

hDevice设备对象句柄，它应由CreateDevice创建。
DelalyTime 延时时间
lFirstChannel 首通道号
lLastChannel 末通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数： CreateDevice，ReleaseDevice

第五节、DA操作函数原型说明

♦ **回读DA输出值**

函数原型：

Visual C++ & C++Builder:

```
BOOL AOReadData(HANDLEhDevice,
                PLONG lpDAValue,
```

LONG lChannel)

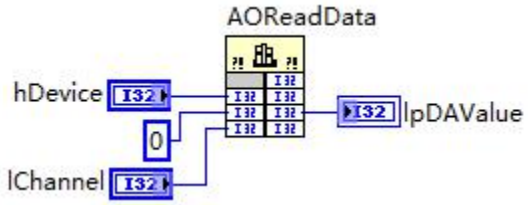
Visual Basic:

Declare Function AOReadData"DAME3000N" (ByVal hDevice As Long,_
ByRef lpDAValue As Long,_
ByVal lChannel As Long) As Boolean

C#：

public static extern Int32 AOReadData (IntPtr hDevice,
Ref Int32 lpDAValue ,
Int32 lChannel)

LabView:



功能：回读DA输出值

参数：

hDevice设备对象句柄，它应由CreateDevice创建。
lpDAValue DA当前输出值
lChannel 通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数：CreateDevice，ReleaseDevice

第六节、DI操作函数原型说明

♦设置DI模式状态

函数原型：

Visual C++ & C++Builder:

BOOL DISetMode(HANDLEhDevice,
LONG Mode[],
LONG lFirstChannel,
LONG lLastChannel)

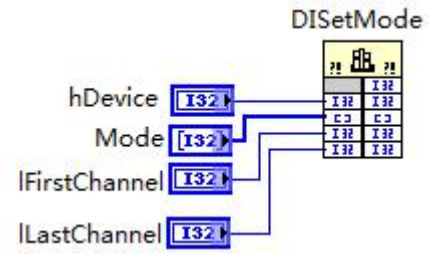
Visual Basic:

Declare Function DISetMode"DAME3000N" (ByVal hDevice As Long,_
ByRef Mode As Long,_
ByVal lFirstChannel As Long,_
ByVal lLastChannel As Long) As Boolean

C#：

public static extern Int32 DISetMode(IntPtr hDevice,
Int32 []Mode ,
Int32 lFirstChannel,
Int32 lLastChannel)

LabView:



功能：设置DI模式状态

参数：

hDevice设备对象句柄，它应由CreateDevice创建。
Mode DI模式
lFirstChannel 首通道号
lLastChannel 末通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数：CreateDevice，ReleaseDevice

♦**获取DI值**

函数原型：

Visual C++ & C++Builder:

```
BOOL DIGetValue(HANDLEhDevice,
                BYTE byDIStatus[],
                LONG lFirstChannel,
                LONG lLastChannel)
```

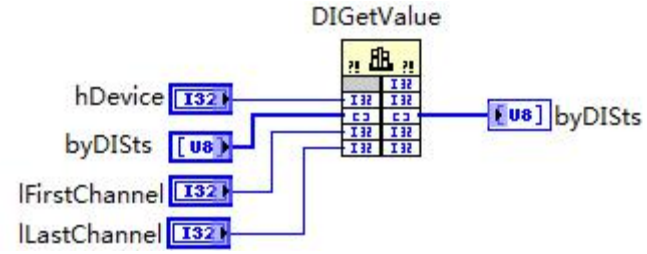
Visual Basic:

```
Declare Function DIGetValue"DAME3000N" (ByVal hDevice As Long,_
                                        ByRef byDIStatus As BYTE ,_
                                        ByVal  lFirstChannel As Long,_

                                        ByVal  lLastChannel As Long) As Boolean
```

C#：

```
public static extern Int32 DIGetValue(IntPtr hDevice,
                                       Byte []byDIStatus ,
                                       Int32  lFirstChannel,
                                       Int32  lLastChannel)
```

LabView:



功能：获取DI值

参数：

hDevice设备对象句柄，它应由CreateDevice创建。
byDIStatus DI值
lFirstChannel 首通道号
lLastChannel 末通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数： CreateDevice，ReleaseDevice

♦设置计数初值

函数原型：

Visual C++ & C++Builder:

```
BOOL DISetCNTInitialValue(HANDLEhDevice,
                          ULONG lInitialValue[],
                          LONG lFirstChannel,
                          LONG lLastChannel)
```

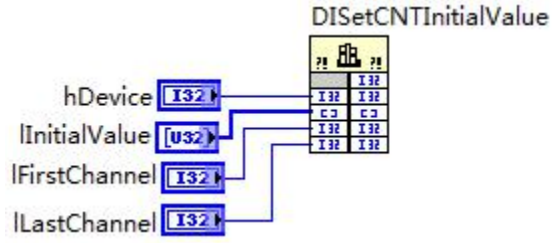
Visual Basic:

```
Declare Function DISetCNTInitialValue"DAME3000N" (ByVal hDevice As Long,_
                                                  ByRef lInitialValue As Long,_
                                                  ByVal  lFirstChannel As Long,_
                                                  ByVal  lLastChannel As Long) As Boolean
```

C#：

```
public static extern Int32 DISetCNTInitialValue(IntPtr hDevice,
                                                UInt32  []lInitialValue ,
                                                Int32  lFirstChannel,
                                                Int32  lLastChannel)
```

LabView:



功能：设置计数初值

参数：

hDevice设备对象句柄，它应由CreateDevice创建。
lInitialValue 计数初值
lFirstChannel 首通道号
lLastChannel 末通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数： CreateDevice，ReleaseDevice

♦设置计数开始或停止

函数原型：

Visual C++ & C++Builder:

```
BOOL DISetStartCNT(HANDLEhDevice,
                   BYTE bDICNTSts[],
                   LONG lFirstChannel,
                   LONG lLastChannel)
```

Visual Basic:

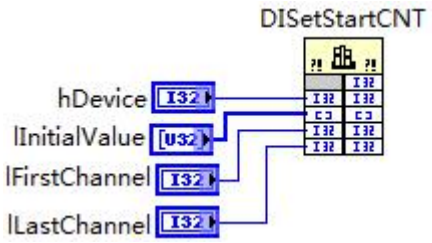
```
Declare Function DISetStartCNT"DAME3000N" (ByVal hDevice As Long,_
```

```
                                        ByRef bDICNTSts As BYTE ,_
                                        ByVal  lFirstChannel As Long,_
                                        ByVal  lLastChannel As Long) As Boolean
```

C#：

```
public static extern Int32 DISetStartCNT(IntPtr hDevice,
                                         Byte   []bDICNTSts ,
                                         Int32  lFirstChannel,
                                         Int32  lLastChannel)
```

LabView:



功能：设置计数开始或停止

参数：

hDevice设备对象句柄，它应由CreateDevice创建。
bDICNTSts 计数器状态 1-开始 0-停止
lFirstChannel 首通道号
lLastChannel 末通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数：CreateDevice，ReleaseDevice

♦清除计数值

函数原型：

Visual C++ & C++Builder:

```
BOOL DIClearCNTValue(HANDLEhDevice,LONG lChannel)
```

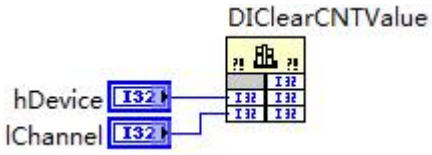
Visual Basic:

```
Declare Function DIClearCNTValue"DAME3000N" (ByVal hDevice As Long, _
                                             ByVal  lChannel As Long) As Boolean
```

C#：

```
public static extern Int32 DIClearCNTValue(IntPtr hDevice, Int32  lChannel )
```

LabView:



功能：清除计数值

参数：

hDevice设备对象句柄，它应由CreateDevice创建。
lChannel 通道号

返回值：若成功，则返回 TRUE,否则返回FALSE。

相关函数：CreateDevice，ReleaseDevice

第三章　硬件参数结构

第一节、DAME3000N设备网络配置结构介绍（_DEVICE_NET_INFO）

Visual C++ & C++Builder:

```
typedef struct _DEVICE_NET_INFO
 {
     char       SzIP[16];              //IP地址, // IP地址, "192.168.2.70"
     char      SubnetMask;//子网掩码, "255.255.255.255"
      Char      Gateway[16];    // 网关, "192.168.2.1"
      char      MAC[20];        // 网卡物理地址,  "00-01-02-03-04-05",用户一般不可修改
      LONGlTCPPort;             // TCP端口号
      LONG    lHTTPPort;        // HTTP端口号
      LONG   lUDPPort;          // UDP端口号
      LONG   lMode;             // 连接模式(0: 静态模式 1: DHCP动态模式)
      char      chDevName[32];// Device self name
} DEVICE_NET_INFO, *PDEVICE_NET_INFO;
```

C#:

```
public struct _DEVICE_NET_INFO
{
   public SByte szIP0;                                    // IP地址, // IP地址, "192.168.2.70"
   public SByte szIP1;
   public SByte szIP2;
   public SByte szIP3;
   public SByte szIP4;
   public SByte szIP5;
   public SByte szIP6;
   public SByte szIP7;
   public SByte szIP8;
   public SByte szIP9;
   public SByte szIP10;
   public SByte szIP11;
   public SByte szIP12;
   public SByte szIP13;
   public SByte szIP14;
   public SByte szIP15;

   public SByte SubnetMask0;                              // 子网掩码, "255.255.255.255"
   public SByte SubnetMask1;
   public SByte SubnetMask2;
   public SByte SubnetMask3;
   public SByte SubnetMask4;
   public SByte SubnetMask5;
   public SByte SubnetMask6;
   public SByte SubnetMask7;
   public SByte SubnetMask8;
   public SByte SubnetMask9;
   public SByte SubnetMask10;
   public SByte SubnetMask11;
   public SByte SubnetMask12;
   public SByte SubnetMask13;
   public SByte SubnetMask14;
   public SByte SubnetMask15;
```

```
public SByte Gateway0;                          // 网关, "192.168.2.1"
public SByte Gateway1;

public SByte Gateway2;
public SByte Gateway3;
public SByte Gateway4;
public SByte Gateway5;
public SByte Gateway6;
public SByte Gateway7;
public SByte Gateway8;
public SByte Gateway9;
public SByte Gateway10;
public SByte Gateway11;
public SByte Gateway12;
public SByte Gateway13;
public SByte Gateway14;
public SByte Gateway15;

public SByte MAC0;              // 网卡物理地址, "00-01-02-03-04-05",用户一般不可修改
public SByte MAC1;
public SByte MAC2;
public SByte MAC3;
public SByte MAC4;
public SByte MAC5;
public SByte MAC6;
public SByte MAC7;
public SByte MAC8;
public SByte MAC9;
public SByte MAC10;
public SByte MAC11;
public SByte MAC12;
public SByte MAC13;
public SByte MAC14;
public SByte MAC15;
public SByte MAC16;
public SByte MAC17;
public SByte MAC18;
public SByte MAC19;

public Int32 lTCPPort;                          // TCP端口号
public Int32 lHTTPPort;                         // HTTP端口号

public Int32 lUDPPort;                          // UDP端口号
public Int32 lMode;                             // 连接模式(0: 静态模式 1: DHCP动态模式)

public SByte chDevName0;                        // Device self name
public SByte chDevName1;
public SByte chDevName2;
public SByte chDevName3;
public SByte chDevName4;
public SByte chDevName5;
public SByte chDevName6;
public SByte chDevName7;
public SByte chDevName8;
public SByte chDevName9;
public SByte chDevName10;
public SByte chDevName11;
public SByte chDevName12;
```

```
    public SByte chDevName13;

    public SByte chDevName14;

    public SByte chDevName15;
    public SByte chDevName16;
    public SByte chDevName17;
    public SByte chDevName18;
    public SByte chDevName19;
    public SByte chDevName20;
    public SByte chDevName21;
    public SByte chDevName22;
    public SByte chDevName23;
    public SByte chDevName24;
    public SByte chDevName25;
    public SByte chDevName26;
    public SByte chDevName27;
    public SByte chDevName28;
    public SByte chDevName29;
    public SByte chDevName30;
    public SByte chDevName31;
}
```

Visual Basic：

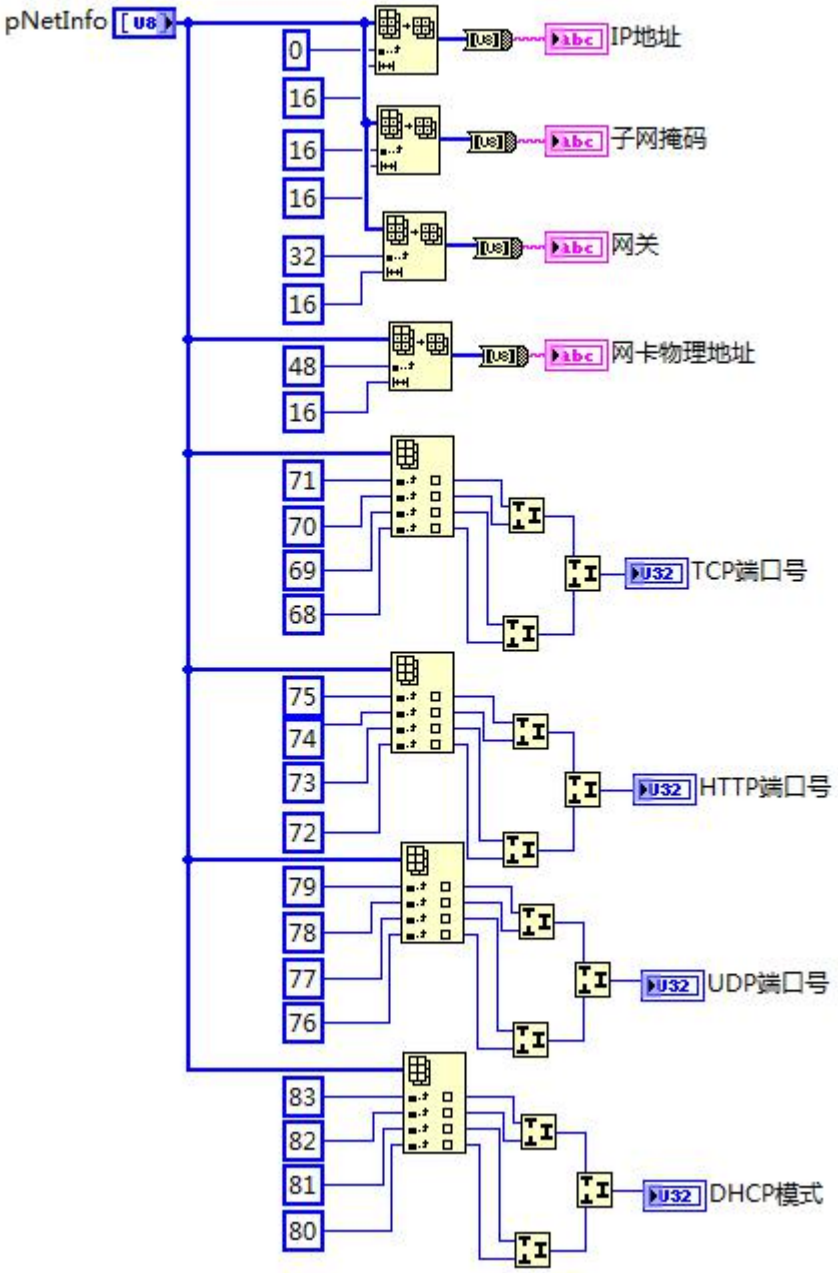
```
    Type DEVICE_NET_INFO
    szIP0 As Byte        ' IP地址, "192.168.2.80"
    szIP1 As Byte
    szIP2 As Byte
    szIP3 As Byte
    szIP4 As Byte
    szIP5 As Byte
    szIP6 As Byte
    szIP7 As Byte
    szIP8 As Byte
    szIP9 As Byte
    szIP10 As Byte
    szIP11 As Byte
    szIP12 As Byte
    szIP13 As Byte
    szIP14 As Byte
    szIP15 As Byte

    SubnetMask0 As Byte        ' 子网掩码, "255.255.255.255"
    SubnetMask1 As Byte
    SubnetMask2 As Byte
    SubnetMask3 As Byte
    SubnetMask4 As Byte
    SubnetMask5 As Byte
    SubnetMask6 As Byte
    SubnetMask7 As Byte
    SubnetMask8 As Byte
    SubnetMask9 As Byte
    SubnetMask10 As Byte
    SubnetMask11 As Byte
    SubnetMask12 As Byte
    SubnetMask13 As Byte
    SubnetMask14 As Byte
    SubnetMask15 As Byte

    Gateway0 As Byte        ' 网关, "192.168.2.1"
```

```
Gateway1 As Byte
Gateway2 As Byte
Gateway3 As Byte

Gateway4 As Byte
Gateway5 As Byte
Gateway6 As Byte
Gateway7 As Byte
Gateway8 As Byte
Gateway9 As Byte
Gateway10 As Byte
Gateway11 As Byte
Gateway12 As Byte
Gateway13 As Byte
Gateway14 As Byte
Gateway15 As Byte

MAC0 As Byte       ' 网卡物理地址, "00-01-02-03-04-05",用户一般不可修改
MAC1 As Byte
MAC2 As Byte
MAC3 As Byte
MAC4 As Byte
MAC5 As Byte
MAC6 As Byte
MAC7 As Byte
MAC8 As Byte
MAC9 As Byte
MAC10 As Byte
MAC11 As Byte
MAC12 As Byte
MAC13 As Byte
MAC14 As Byte
MAC15 As Byte
MAC16 As Byte
MAC17 As Byte
MAC18 As Byte
MAC19 As Byte


lTCPPort As Long          ' TCP端口号
lHTTPPort As Long          ' HTTP端口号
lUDPPort As Long          ' UDP端口号
lMode As Long            ' 连接模式(0: 静态模式  1: DHCP动态模式)

chDevName0 As Byte       ' 网卡物理地址, "00-01-02-03-04-05",用户一般不可修改
chDevName1 As Byte
chDevName2 As Byte
chDevName3 As Byte
chDevName4 As Byte
chDevName5 As Byte
chDevName6 As Byte
chDevName7 As Byte
chDevName8 As Byte
chDevName9 As Byte
chDevName10 As Byte
chDevName11 As Byte
chDevName12 As Byte
chDevName13 As Byte
chDevName14 As Byte
chDevName15 As Byte
chDevName16 As Byte
```

chDevName17 As Byte
chDevName18 As Byte
chDevName19 As Byte

chDevName20 As Byte
chDevName21 As Byte
chDevName22 As Byte
chDevName23 As Byte
chDevName24 As Byte
chDevName25 As Byte
chDevName26 As Byte
chDevName27 As Byte
chDevName28 As Byte
chDevName29 As Byte
chDevName30 As Byte
chDevName31 As Byte
End Type

LabVIEW:



第四章 软件使用说明

第一节、上电及初始化

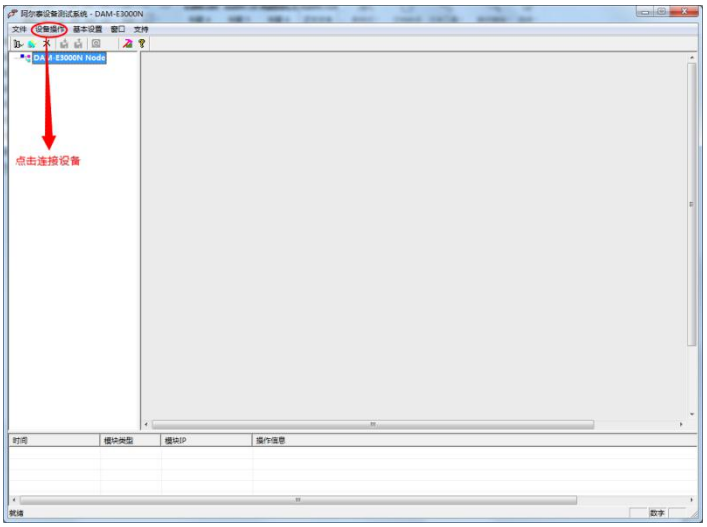
1) 连接电源：“＋Vs”接电源正，“GND”接地，模块供电要求：+7V— +30V。

2) 连接网线：DAME3000N网线连接到计算机。

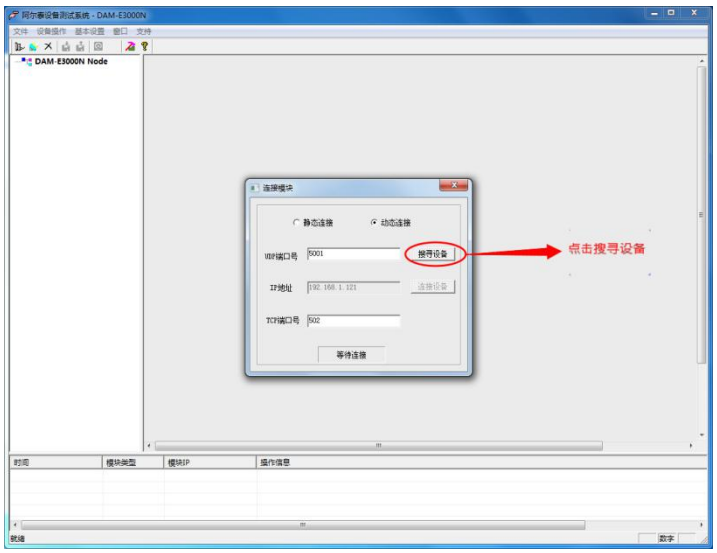
3) 复位：断电情况下，将端子 INIT*和 GND 短接，重新上电，电源模块指示灯快速闪烁 3 次，待指示灯停止闪烁后， 再断电，将 INIT*和 GND 断开，再次上电模块恢复出厂设置完成。

第二节、上位机软件的基本用法

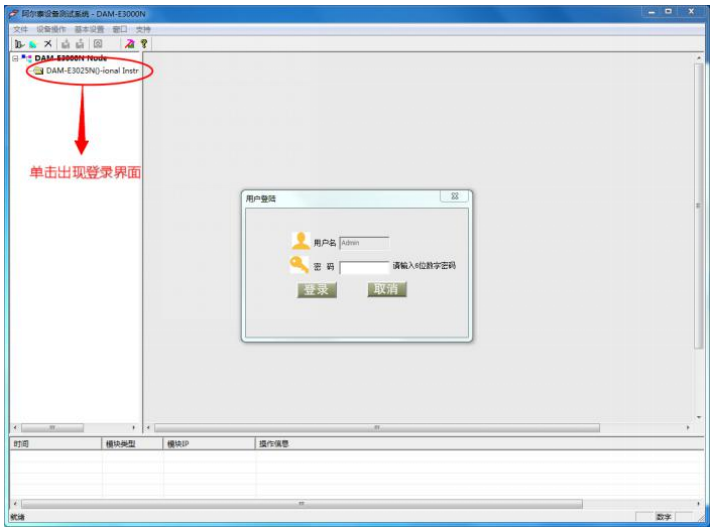
1）选择设备操作栏下的连接设备进行设备的连接



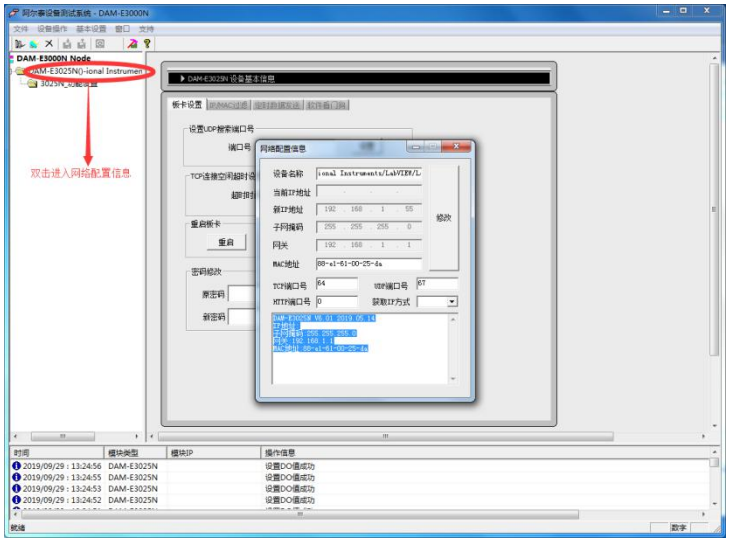
2）点击搜寻设备 搜索同局域网内的设备



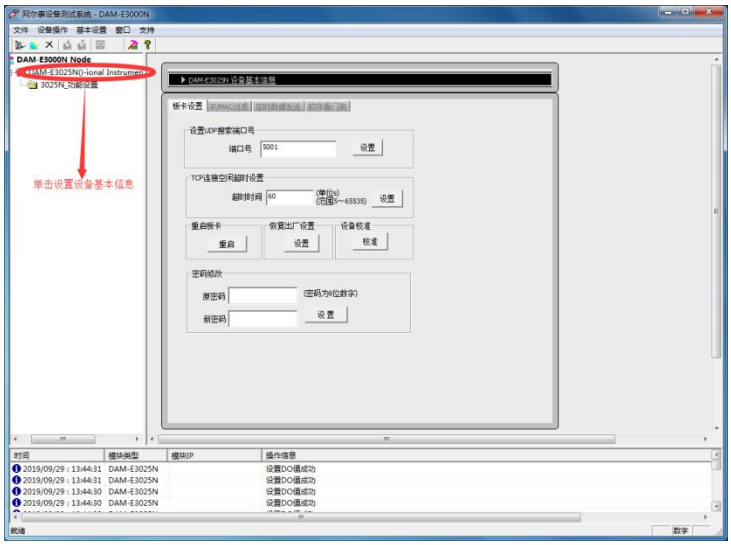
3)用户登陆（初始密码为666666）



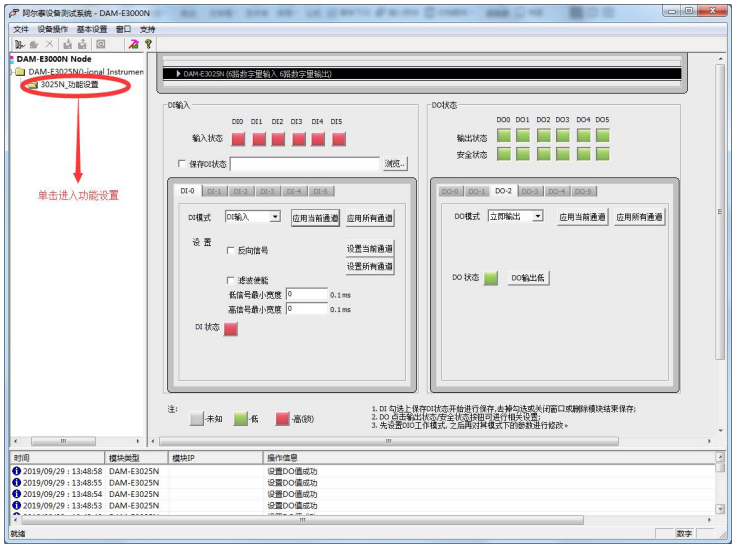
4）设置网络配置信息



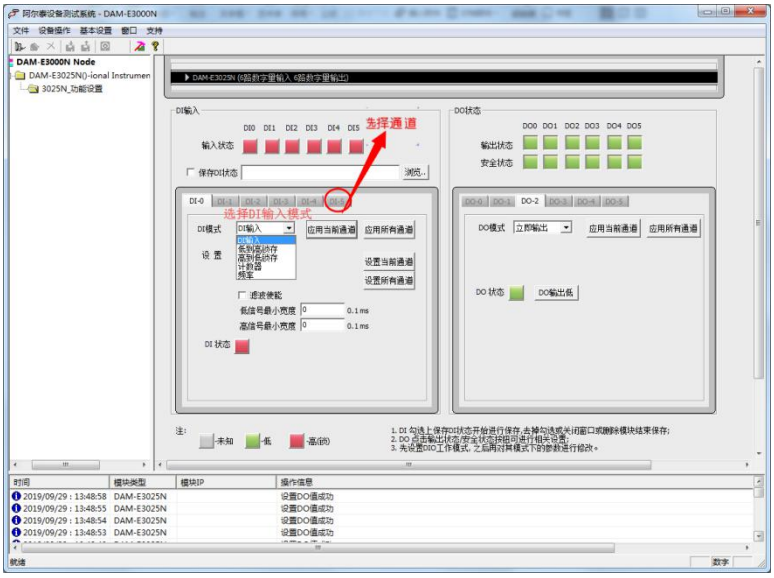
5）设置设备基本信息



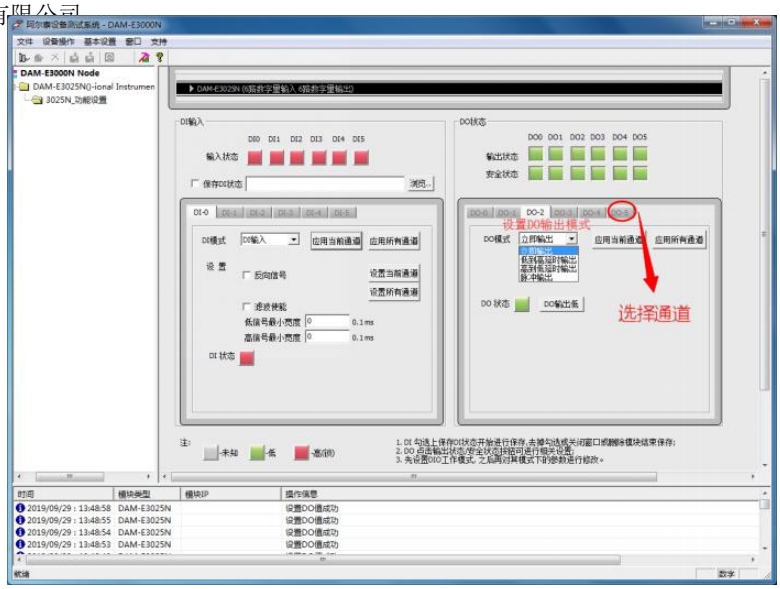
6）查看设备的功能



7）DI功能的操作方法



8）DO功能的操作方法


设置DO输出模式
选择通道