# DAM6600

# 分布式 RS-485/以太网数据采集及控制系统

# 用户手册

版本：V6.005

# 目 录

# 第一章、简介

## 1.1 概述

DAM6000 系列能够通过多通道 I/O 模块进行数据采集和过程控制，为工业应用提供了灵活的数据采集和控制应用方案。此产品系统由两部分组成：基座（主单元）和 I/O 模块。主单元部分分为两个类别：DAM6600——分布式 RS-485/以太网数据采集及控制系统和 DAM6800——可视型数据采集及控制系统。

DAM6600——分布式RS-485/以太网数据采集及控制系统同时支持工业以太网总线、RS-485总线、CAN总线，可供客户选择多种通讯方式，系统内核小，运行速度快，模块化分布为系统提供了灵活的系统配置。DAM6600可用于多种工业环境，所带模块覆盖了工业I/O标准信号，包括：模拟量输入/输出、热电偶、热电阻、数字量输入/输出、继电器输出、计数/频率等。

## 1.2 特点

DAM**6600**——分布式 RS-485/以太网数据采集及控制系统包括 CPU 模块，电源模块，8 槽底座并带有 RS232、RS485 总线，各 I/O 模块。其中 CPU 模块、电源模块、8 槽底座统称基座（主单元）。

采用高性能的 32 位 RISC 处理器（ARM7TDMI 核）

带有 RJ-45 的 10/100M base-T 以太网接口

256Kbytes 内部高速 Flash

16Kbytes 内部高速 SRAM

1Mbytes 外部高速 SRAM

256Mbytes（Nand）+2Mbytes（Nor）外部 Flash 存储器

256bytesEEPROM

低功耗的精确 RTC 实时时钟。

兼容 CAN2.0A、CAN2.0B 的 CAN 总线控制器（**CAN 总线软件正在升级中，目前不建议使用**）

主频高达 60MHz（外部晶振为 18.432MHz）

1 个三线制的 RS232 串口用于 CPU 程序更新

2 个 RS232/RS485 可选串口、1 个 RS485 接口用于通信

可同时带 8 个 16 路 I/O 模块，控制本地 128 个 I/O 点

8 槽底座可任意配置模块。

支持 Modbus RTU 和 Modbus TCP（Server、Client）协议

## 1.3 产品图片

# 第二章、使用方法

## 2.1 工作要求

**环境温度：**

储存温度：-25～+85℃

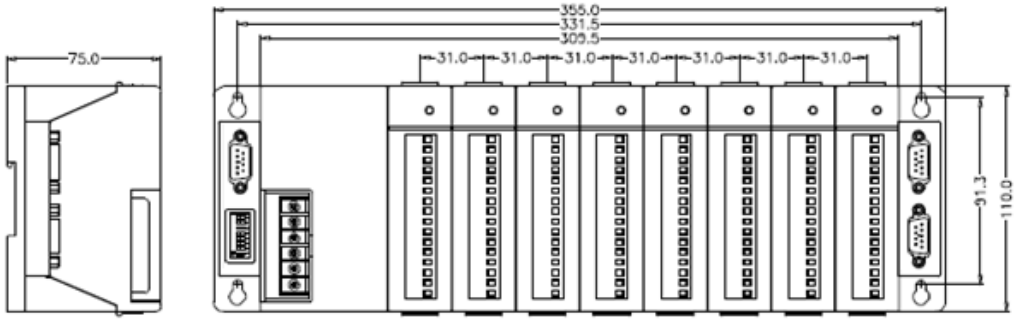
工作温度：-10～+70℃

工作湿度：5～95%

工作电压：+12V～+36Vdc（整体系统工作电压）

## 2.2 尺寸

尺寸如图所示：




## 2.3 模块安装

DAM6600 系统外围接口灵活，客户可根据自己的需要选择合适的配置 IO 模块，具体可选 IO 模块参考第三章模块选型。在 DAM6600 系统底座上有 8 个槽可供插接 IO 模块板，插接方法如图所示。如果客户不需要配置全 8 个 IO 模块，没有配置模块的槽可用不带开孔的外壳覆盖。

## 2.4 固定安装

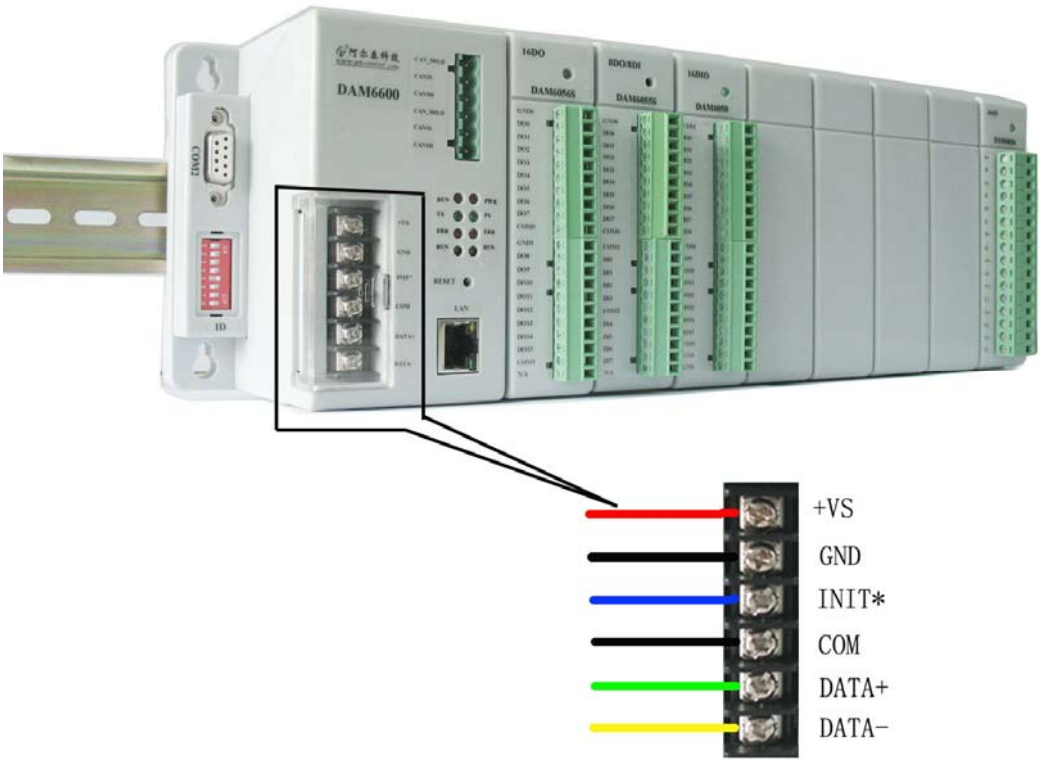
DAM6600 可安装到垂直墙体上或者 DIN 轨道上。

**壁挂式安装：**将系统通过螺钉安装在墙体上，安装所用螺钉为#7（4mm 直径）。

**DIN 轨道安装：**DAM6600 系统可通过 DIN 轨道安装到机柜中。系统的底部有三个导轨锁定夹，安装时先将锁定夹拨出，将系统挂接到导轨上，然后将系统推平压入导轨，将锁定夹拨回以固定系统。如果想要移除系统，首先将锁定夹拨出，然后轻轻举起拿下系统。


## 2.5 面板说明

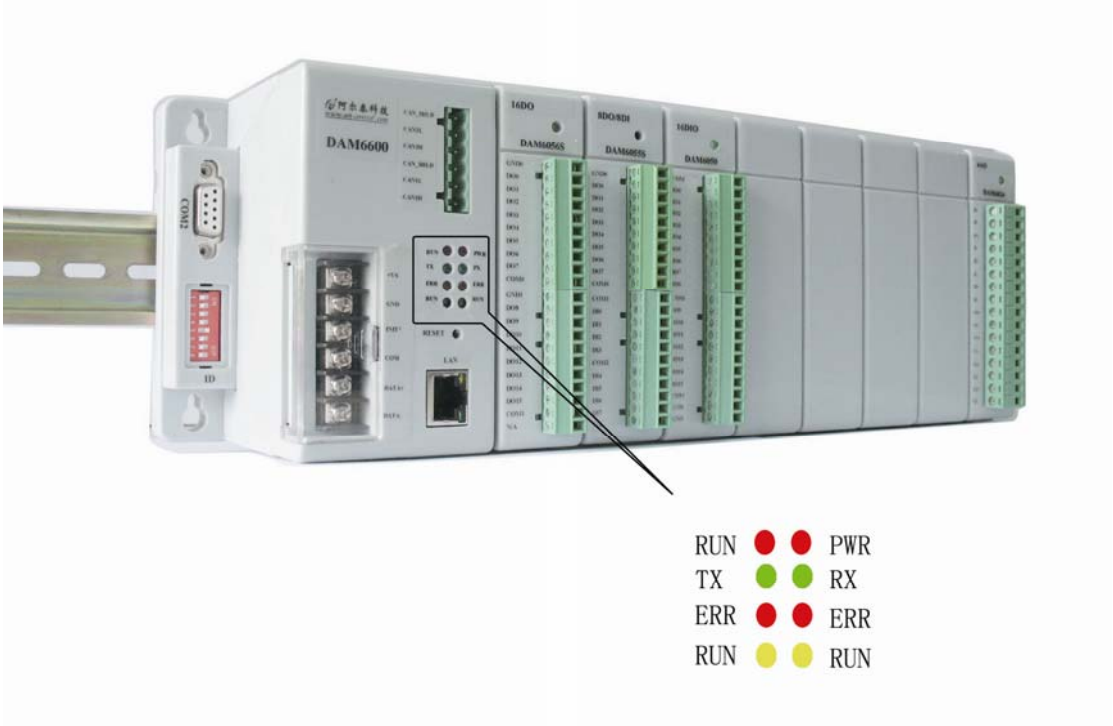
**直流电源接线：**如图所示，系统供电电压为+12~+36Vdc，在+Vs 和 GND 之间接线。建议接线使用红色线和黑色线，红色线接电源正极（+Vs），黑色线接电源负极（GND），线径至少 ϕ2mm。接线端子除电源接线外，还有四个接口，作用分别为：INIT*用于恢复系统出厂状态，当INIT*输入逻辑低电平时，系统所有用户设置值失效。COM提供了RS485总线的参考地平面。DATA+和DATA-提供RS485总线差分接线端。RS485总线接口为软件设置的COM1口，PC机可通过RS485总线连接32个DAM6600设备，RS485总线传输速率可达115200bps。




**LED 灯指示：**系统主面板上有 8 个 LED 灯，如下图所示，作用如下表。

| | |
|---|---|
| PWR（红色） | CPU 电源指示灯 |
| RUN（红色） | CPU 运行灯 |
| TX（绿色） | 串行口发送数据指示灯 |
| RX（绿色） | 串行口接收数据指示灯 |
| | |

| CAN 总线指示灯，目前 CAN 总线软件正在升级中，以下指示灯功能待定 | |
|---|---|
| ERR（左侧红色） | CAN 总线 1 错误指示灯 |
| ERR（右侧红色） | CAN 总线 2 错误指示灯 |
| RUN（左侧黄色） | CAN 总线 1 运行指示灯 |
| RUN（右侧黄色） | CAN 总线 2 运行指示灯 |



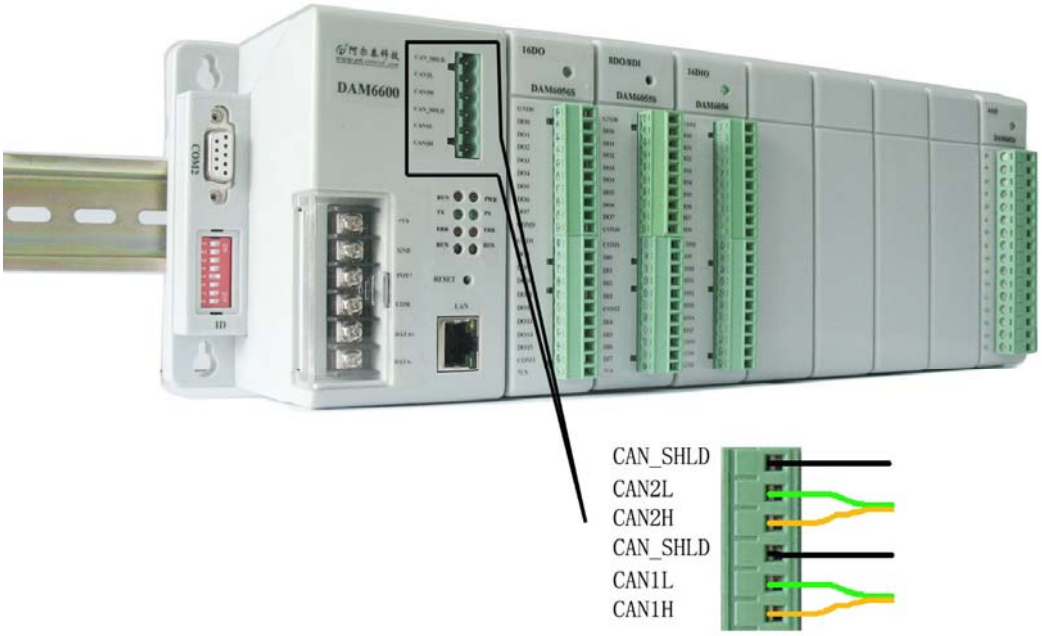

此外，每个 IO 模块板上端有一个绿色指示灯，指示 IO 模块正在运行；个别数字量输入/输出模块还有端口状态指示灯，如 DAM6051D、DAM6056D 等等。

**复位按键：**在 LED 指示灯下方有“RESET”按键孔，用于复位系统操作，当上位机和本机断开重连或者更换输入输出模块时，由于本机还处于刚才的工作方式，此时可直接点击复位按键复位控制。

**ID 号设置：**当多个 DAM6600 同时工作时，串口通信为区分每个系统，需要将 8 位拨码开关设置为不同数值。拨码开关拨到“ON”时，此位为逻辑 0，拨码开关拨到“OFF”时，此位为逻辑 1。此设置在使用串口连接 DAM6000.exe 软件时有用。

**注意**：ID 号可选范围为 1 到 255，即 01h 到 FFh。系统默认 **00h 为初始化设置。**


**CAN总线接口：**DAM6600带有两路CAN总线，软件正在升级中，硬件连接如图所示，CAN_SHILD为屏蔽线，可不接。CAN总线接口可以与上位机连线，也可以与其他DAM6600连线，PC机也可通过CAN总线连接多个DAM6600设备。




**以太网接口：**DAM6600 提供了 10/100MBase_T 以太网通信接口，可与电脑、路由器等通过 CAT5 双绞线相连，直连交叉均可，传输距离可达 100m。如下图所示。

本机出厂默认设置为：IP 地址：192.168.2.80

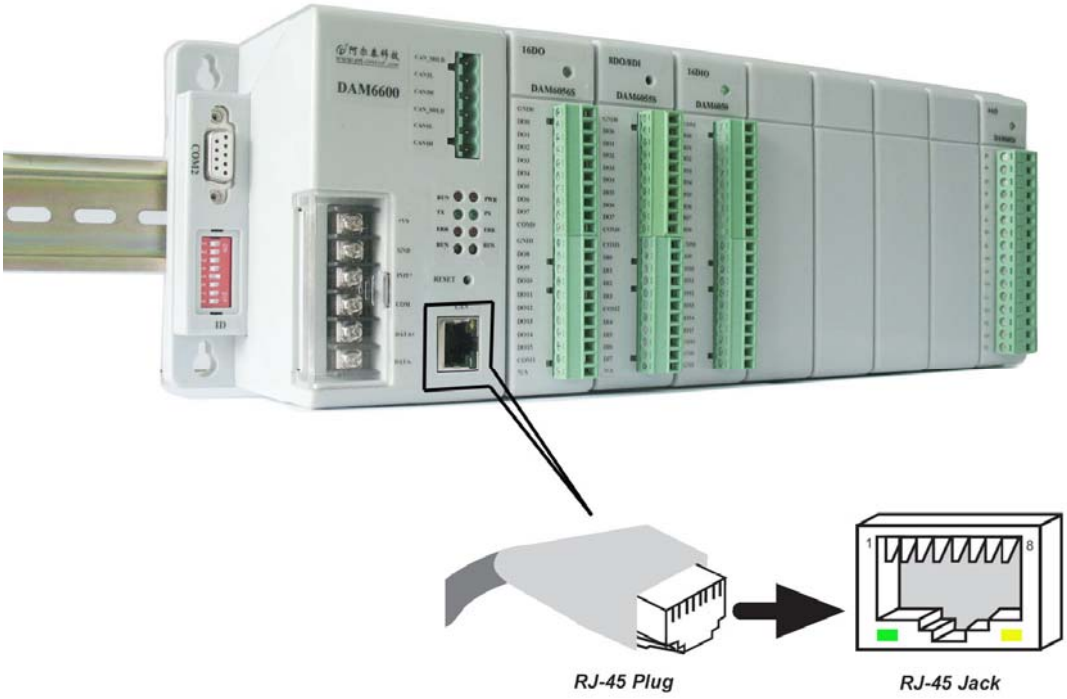
端口号：502

默认网关：192.168.2.1

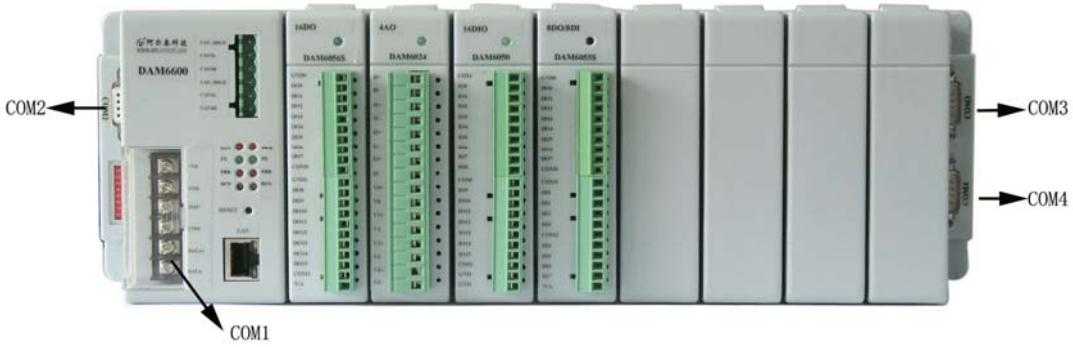
可通过软件 DAM6000.exe 修改 IP 地址。

RJ45 网口上有两个 LED 灯，功能如下：

| | 状态 | 功能 |
|---|---|---|
| 绿色 LED | 亮 | 100Mbps |
| | 灭 | 10Mbps |
| 橙色 LED | 闪烁 | 数据通信 |
| | 灭 | 无数据 |

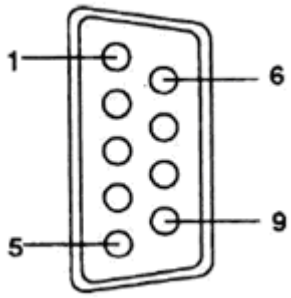



**COM 口：**本机器有 4 个 COM 口，其中 COM1 在电源端子侧，是 RS-485 总线；COM2 为 3 线 RS232，DB9 母头接口；COM3、COM4 是 RS232(9 线)/RS485 可选端口，均为 DB9 公头接口。本机提供的串行口通信速率最高达 115200bps，出厂默认设置波特率 9600bps。可通过软件 DAM6000.exe 修改 IP 地址。




（1）COM1 接口为 RS485 接口，前面已经讲述。

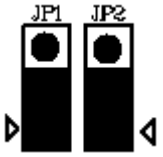
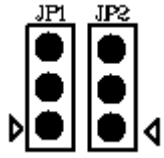
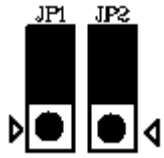
（2）COM2 接口为 3 线 RS232，使用 DB9 母头插座，用于 CPU 程序更新，引脚定义从系统输出侧来说的，连接计算机时要使用交叉 DB9 串口线，定义如下：



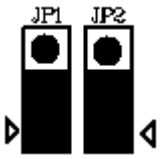
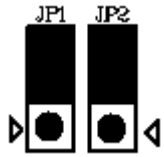

DB9 母头接口

| **Pin NO.** | **Description** |
|---|---|
| **Pin 1** | Not Used |
| **Pin 2** | Data Send（TXD） |
| **Pin 3** | Data Receive（RXD） |
| **Pin 4** | Not Used |
| **Pin 5** | RS232 Signal Ground (GND) |
| **Pin 6** | Not Used |
| **Pin 7** | Not Used |
| **Pin 8** | Not Used |
| **Pin 9** | Not Used |

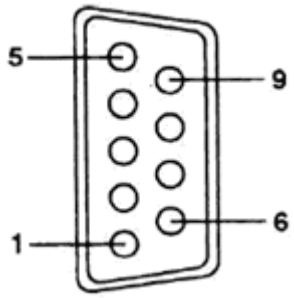
（3）COM3、COM4 接口为 RS232、RS485 可选接口，接口功能通过系统内部底板上跳线 JP1、JP2 选择，如下表所示：

| | 跳线方式 |
|---|---|
| RS232（9 线） |  |
| RS232（9 线） |  <br>不接跳线状态下 |
| RS485 |  |

还有另外两个跳线，JP3、JP4 用来选择 RS485 终端匹配电阻的：

| | 跳线方式 |
|---|---|
| 终端匹配电阻为 300R |  |
| 终端匹配电阻为 120R |  |

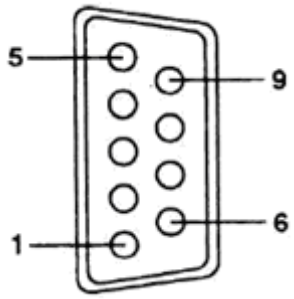
COM3、COM4 引脚定义相同，9 线 RS232 接口采用 DB9 公头插座，引脚定义从系统输出侧来说的，连接计算机时要使用交叉 DB9 串口线，定义如下：




DB9 公头接口

| **Pin NO.** | **Description** |
|---|---|
| **Pin 1** | Carrier Detect（DCD） |
| **Pin 2** | Data Send（RXD） |
| **Pin 3** | Data Receive（TXD） |
| **Pin 4** | Data Terminal Ready（DTR） |
| **Pin 5** | RS232 Signal Ground (GND) |
| **Pin 6** | Data Set Ready（DSR） |
| **Pin 7** | Request to Send（RTS） |
| **Pin 8** | Clear to Send（CTS） |
| **Pin 9** | Ring Indicator（RI） |

RS485 接口引脚定义如下：

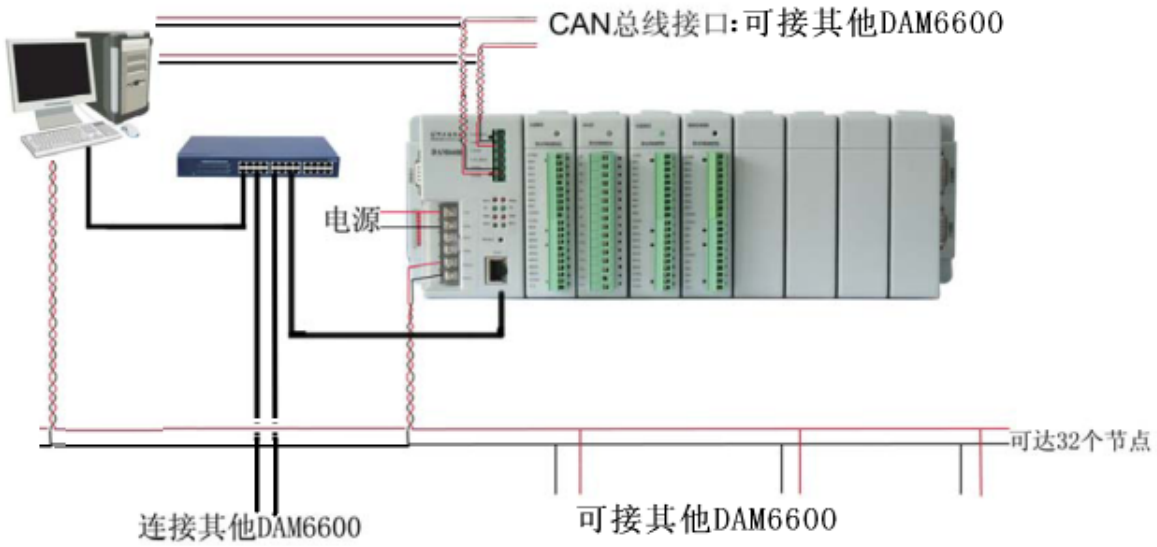



DB9 公头接口

| **Pin NO.** | **Description** |
|---|---|
| **Pin 1** | RS485 Data- |
| **Pin 2** | Not Used |
| **Pin 3** | Not Used |
| **Pin 4** | RS485 Data+ |
| **Pin 5** | RS485 Signal Ground (GND) |
| **Pin 6** | Not Used |
| **Pin 7** | Not Used |
| **Pin 8** | Not Used |
| **Pin 9** | Not Used |

**各模块端口：**各模块端口接线参考模块使用说明书中的接线方法。

## 2.6 系统连接




## 2.7 配件

标配：RS232 串口线（交叉）一条
光盘一张

# 第三章、IO 模块说明

客户可自行选择 IO 模块配置，目前可选 IO 模块有数字量输入/输出，模拟量输入/输出、计数器等功能。具体模块功能见模块使用说明书，选型目录见下表。

**DAM6000 系列 I/O 模块选型表**

| 模块 | | DAM 6013（改版中） | DAM 6017 | DAM 6018（改版中） | DAM 6024 | DAM 6050 | DAM 6051D | DAM 6051S | DAM 6052 | DAM 6055S | DAM 6056D | DAM 6056S | DAM 6060 | DAM 6068 | DAM 6069 | DAM 6080 | DAM 6081（改版中） |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 模拟量输入 | 分辨率 | 16 位 | 16 位 | 16 位 | | | | | | | | | | | | | |
| | 输入通道 | 3 | 8 | 7 | | | | | | | | | | | | | |
| | 采样速率 | 10 | 100 | 10 | | | | | | | | | | | | | |
| | 电压输入 | —— | ±150mV<br>±500mV<br>± 1 V<br>± 5 V<br>±10V | ±15mV<br>±50mV<br>±100mV<br>±500mV<br>±1V<br>±2.5 V | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 电流输入 | —— | ± 20 mA | ±20 mA | | | | | | | | | | | | |
| | 直接传感器输入 | Pt100 或 Cu50、Cu100 RTD | —— | J、K、T、E、R、S、B、N、C、WRe5-WRe26 | | | | | | | | | | | | |
| 模拟量输出 | 分辨率 | | | | 12 | | | | | | | | | | | |
| | 电压输出 | | | | 0-10V | | | | | | | | | | | |
| | 电流输出 | | | | 0-20mA 4-20mA | | | | | | | | | | | |
| 数字 | 数字量输 | | | | | 16 路 DIO（可选择） | 16 带 LED | 16 带 LED | 8 | 8 带 LED | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 量输入输出 | 入通道 | | | | | | | | | | | | | | | |
| | 数字量输出通道 | | | | | | | | | 8 带 LED | 16 带 LED | 16 带 LED | 6 路继电器（2 个 A 型和 4 个 C 型） | 8 路继电器（A 型） | 8 个功率继电器（A 型） | | 4 路 |
| 计数器 32 位 | 通道 | | | | | | | | | | | | | | | 4 | 4 |
| | 输入频率 | | | | | | | | | | | | | | | 5000 Hz(最高) | 1MHz（最高） |
| | 模式 | | | | | | | | | | | | | | | 频率、加/减计数器，双向继电器 | 频率、加/减计数器，双向继电器 |
| 隔离 | | 3000V dc | 3000V dc | 3000V dc | 3000V dc | | | 2500V dc | 5000V rms | 2500V dc | | 2500V dc | | | | 2500 Vrms | 2500 Vrms |

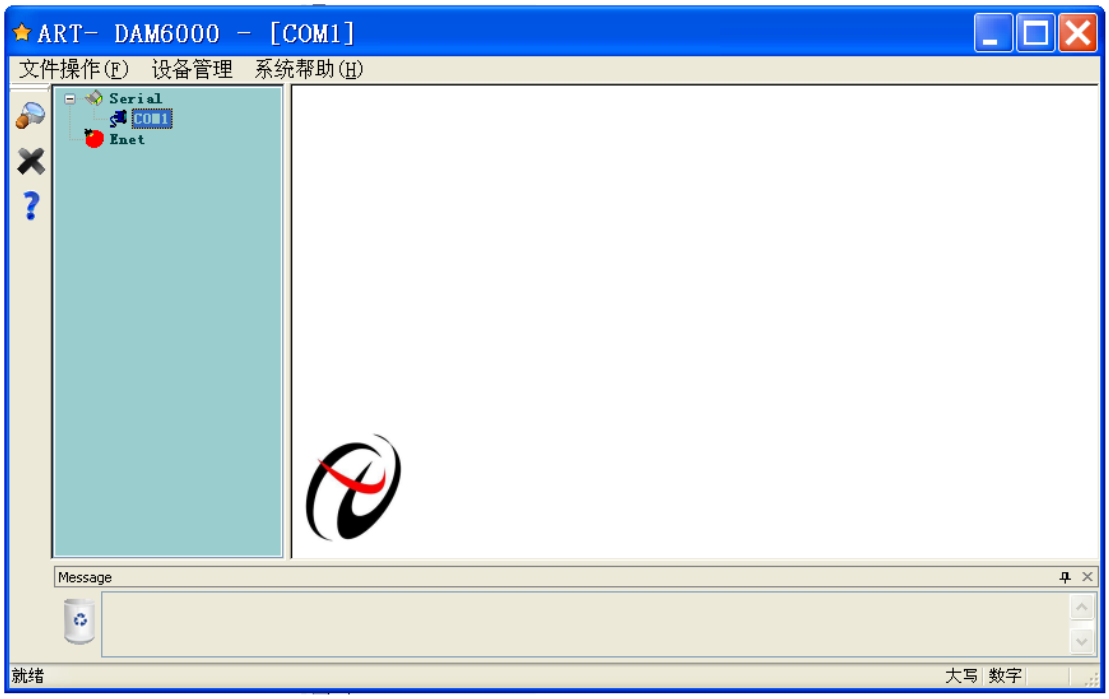
# 第四章、软件应用

## 4.1 接线方法

模块供电要求：直流电压+12V～+36Vdc，“+Vs”接电源正，“GND”接地。将系统 LAN 以太网接口通过 CAT-5 双绞线连到计算机上，或者也可通过 COM1、COM2、COM3、COM4 接口与计算机相接，具体接口方法见第二章 COM 接口连线。上电后 PWR 指示灯（红色）常量，RUN 指示灯（红色）处于闪烁状态，表示系统正在运行。

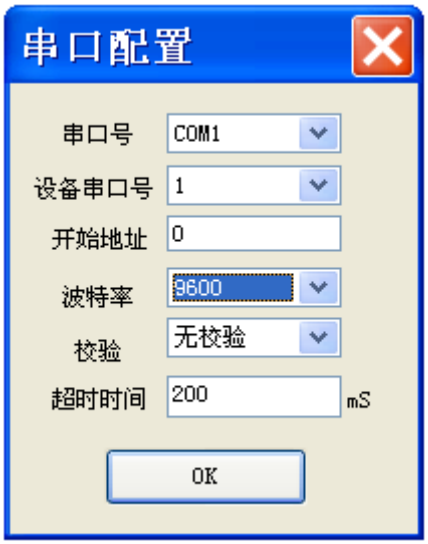
## 4.2 软件说明

安装完成 DAM6000.exe 软件后，打开 DAM6000.exe，可以看到如图所示的开始界面。

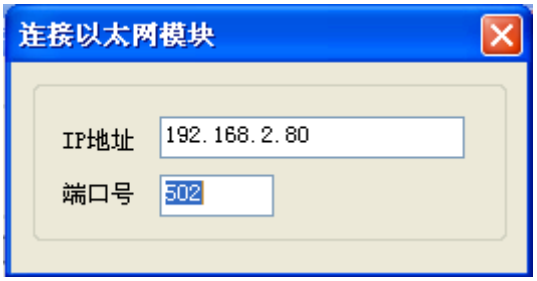



可选择串口连接或者以太网连接计算机和系统，点击放大镜图标，或者任务栏中的“设备管理”下拉框中“搜索设备”，出现如下图标：


当使用串口连接时，在串口前方框打“√”，并可在“串口配置”中配置计算机串口的串口号、设备串口号、波特率、起始地址、校验位、搜索时间以及。起始地址是根据底板上的 ID 拨码开关来设置的，拨码开关拨到多少，填多少。设备串口号对应于 DAM6600 的相应串口号，计算机连接到 DAM6600 的哪个串口，设备串口号就设为几，DAM6600 的串口说明参考章节 2.5 面板说明的 COM 口说明部分。




当选择网口连接时，在以太网前方框打“√”，并可在“以太网配置”中配置系统的 IP 地址、端口号。默认系统 IP 地址为 192.168.2.80，端口号默认 502。



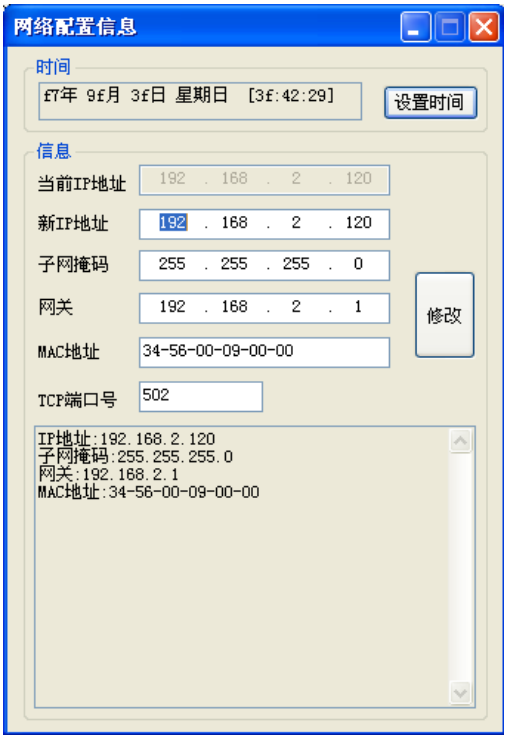

设置完后，点击“搜寻”，搜索到后在开始界面的左侧出现“地址：xxx（DAM6000）”字样。点击此字样，就会在开始界面右侧出现目前系统所带 IO 模块。开始界面下侧的“Message”中显示了各种操作信息，包括操作时间，模块类型，模块地址，以及操作失败，成功以及模块状态等信息。


点击右侧上方的模块型号，即可进入模块的控制界面。

如果想退出操作，可点击左侧叉形图标，或者任务栏中的“设备管理”下拉框中“删除设备”，即可退出操作。每次更换模块，或者修改模块 ID 号，需要将硬件设备断电重启，并用软件重新搜索设备才能保证系统正常工作。

如果想修改 DAM6600 系统的串口波特率，可在软件搜集到设备后，双击左侧 COM 下“地址：xxx（DAM6000）”，出现如下画面：选择对应的 DAM6600 设备串口号（计算机连接到 DAM6600 的哪个串口，设备串口号就设为几），设置好时间、波特率等后，点击修改即可，修改完后，需要将系统重新启动。


如果想修改 DAM6600 系统的 IP 地址，可在软件搜集到设备后，双击左侧 Enet 下“IP 192.168.2.80（DAM6000）”，出现如下画面：设置好时间、IP 地址等后，点击修改即可，修改完后，需要将系统重现启动。


网络配置信息
时间
f7年 9f月 3f日 星期日 [3f:42:29]
设置时间
信息
当前IP地址 192 . 168 . 2 . 120
新IP地址 192 . 168 . 2 . 120
子网掩码 255 . 255 . 255 . 0
网关 192 . 168 . 2 . 1
MAC地址 34-56-00-09-00-00
修改
TCP端口号 502
IP地址:192.168.2.120
子网掩码:255.255.255.0
网关:192.168.2.1
MAC地址:34-56-00-09-00-00

# 第五章、软件函数说明

由于我公司的设备应用于各种不同的领域，有些用户可能根本不关心硬件设备的控制细节、只关心AD、DA、DI、DO等，然后就能通过一两个简易的采集函数便能轻松得到所需要的数据。这方面的用户我们称之为上层用户。那么还有一部分用户不仅对硬件控制熟悉，而且由于应用对象的特殊要求，则要直接控制设备的每一个端口，这是一种复杂的工作，但又是必须的工作，我们则把这一群需要直接跟设备端口打交道的用户称之为底层用户。因此总的看来，上层用户要求简单，快捷，他们最希望他们在软件操作上所要面对的全是他们最关心的问题，比如在正式采集数据之前，只须用户调用一个简易的初始化函数（如SetRangeAD）告诉设备需要输入的模式是什么等，然后便可以用ReadDeviceAD函数即可实现数据连续不间断采样。而关于设备的物理地址、端口分配及功能定义等复杂的硬件信息则与上层用户无任何关系。那么对于底层用户则不然。他们不仅要关心设备的物理地址，还要关心虚拟地址、端口寄存器的功能分配，甚至每个端口的Bit位都要了如指掌，看起来这是一项相当复杂、繁琐的工作。但是这些底层用户一旦使用我们提供的技术支持，则不仅可以让您不必熟悉TCP/IP、UDP复杂的控制协议，同时还可以省掉您许多繁琐的工作。这个时候您便可以用CreateCOM或CreateENet创建的句柄，再根据硬件使用说明书中各种命令字的功能说明，然后使用WriteDeviceChar和ReadDeviceChar对这些模块进行读写操作，即可实现设备的所有控制。

综上所述，用户使用我公司提供的驱动程序软件包极大的方便和满足您的各种需求。但为了您更省心，别忘了在您正式阅读下面的函数说明时，先得明白自己是上层用户还是底层用户，因为在《第一节 接口函数列表》中的备注栏里明确注明了适用对象。

## 5.1 设备驱动接口函数列表（每个函数省略了前缀“DAM6000_”）

表5-1设备驱动接口函数列表

| 函数名 | 函数功能 | 备注 |
|---|---|---|
| 串口操作函数 | | |
| CreateCOM | 创建串口设备对象 | 上层及底层用户 |
| InitCOM | 初始化模块之间的通信参数 | |
| ReleaseCOM | 关闭设备，且释放串口设备对象 | 上层及底层用户 |
| GetDevInfoCOM | 读取模块信息 | 上层用户 |
| SetDevInfoCOM | 修改模块信息 | 上层用户 |
| 远程配置(以太网) 操作函数 | | |
| CreateENet | 创建以太网设备 | |
| ReleaseENet | 释放以太网设备对象 | |
| GetENetworkConfig | 获得设备的网络配置信息 | |
| SetENetworkConfig | 设置设备的网络配置信息 | |
| SearchDevice | 查询模块设备是否在线(DHCP) | |
| SearchDeviceEx | 查询模块设备是否在线(DHCP) | |
| 共用函数 | | |
| GetModuleConfig | 获得模块配置参数 | |
| **1、AD**读取函数 | | |
| ReadDeviceAD | 读取模拟量输入 | 上层用户 |
| GetRangeAD | 获得模拟量输入量程 | 上层用户 |
| SetRangeAD | 设置模拟量输入量程 | 上层用户 |

| | | |
|---|---|---|
| SetZeroAD | 调整AD零点 | 上层用户 |
| SetFullAD | 调整AD满度 | 上层用户 |
| **2、DA** 输出函数 | | |
| GetDeviceDA | 回读模拟量输出值 | 上层用户 |
| WriteDeviceDA | 设定单通道DA | 上层用户 |
| GetRangeDA | 读取模拟量输出量程 | 上层用户 |
| SetRangeDA | 设置模拟量输出量程 | 上层用户 |
| SetZeroDA | 调整DA零点 | 上层用户 |
| SetFullDA | 调整DA满度 | 上层用户 |
| **3、DI** 输入函数 | | |
| GetDeviceDI | 获取DI输入值 | 上层用户 |
| **4、DO** 输出函数 | | |
| SetDeviceDO | 设置DO当前输出值 | 上层用户 |
| GetDeviceDO | 读取DO输出值 | 上层用户 |
| GetPowerOnValueDO | 获取DO上电初始值 | 上层用户 |
| SetPowerOnValueDO | 修改DO上电初始值 | 上层用户 |
| GetSafeValueDO | 读DO安全值 | 上层用户 |
| SetSafeValueDO | 修改设备中的DO安全值 | 上层用户 |
| 5、计数器 | | |
| SetCounterMode | 对各个计数器进行参数设置 | |
| SetCounterFilterValue | 设置滤波系数 | |
| ClrCounterSts | 清计数器状态 | |
| GetCounterSts | 获得计数器设备硬件参数状态 | |
| StartCounter | 启动计数器计数 | |
| StopCounter | 停止计数器计数 | |
| GetCounterCurVal | 取得计数器当前值 | |
| ResetCounter | 计数器复位 | |
| 6、模块时间函数 | | |
| GetTime | 取得模块时间 | |
| SetTime | 设置模块时间 | |
| 7、读取最后一个错误函数 | | |
| GetLastErr | 获得最后一个错误 | |
| 8、注册表函数 | | |
| SaveIPAddress | 保存IP到注册表 | |
| LoadIPAddress | 载入IP到应用程序 | |
| 9、Modbus协议函数 | | |
| ReadCoils | 读继电器状态 | |
| ReadDiscretes | 读开关量输入 | |
| ReadMultiRegs | 读保持寄存器 | |
| ReadInputRegs | 读输入寄存器 | |
| WriteCoil | 设置单个继电器 | |
| WriteSingleReg | 设置单个保持寄存器 | |
| ForceMultiCoils | 设置多个继电器 | |

| | | |
|---|---|---|
| WriteMultiRegs | 设置多个保持寄存器 | |
| ReadRegFile | 读取文件记录 | |
| WriteRegFile | 写文件记录 | |
| 10、输入输出任意二进制字符 | | |
| WriteDeviceBYTE | 直接写设备 | |
| ReadDeviceBYTE | 直接读设备 | |

使用需知：

Visual C++ & C++Builder：

要使用如下函数关键的问题是：

首先，必须在您的源程序中包含如下语句：

#include “C:\Art\ DAM6000\INCLUDE\ DAM6000.H”

注：以上语句采用默认路径和默认板号，应根据您的板号和安装情况确定DAM6000.H文件的正确路径，当然也可以把此文件拷到您的源程序目录中。

其次，您还应该在Visual C++编译环境软件包的Project Setting对话框的Link属性页中的Object/Library Module输入行中加入指令C:\Art\ DAM6000 \ DAM6000.LIB

或者：单击Visual C++编译环境软件包的Project菜单中的Add To Project的菜单项，在此项中再单击Files…，在随后弹出的对话框中选择DAM6000.Lib，再单击“确定”，即可完成。

注：以上语句采用默认路径和默认板号，应根据您的板号和安装情况确定DAM6000.LIB的路径，当然也可以把此文件拷到您的源程序目录中。

另外，在Visual C++演示工程的目录下，也有相应的DAM6000.h和DAM6000.Lib文件。

为了驱动程序和相关接口尽量精炼快速，所以没有加任何调试代码，因此用户在使用**VC**接口的时候应使用发行版本进行源代码编译（**Win32 Release**），而不应该使用调试版本（**Win32 Debug**）。具体方法是在源代码编译前，执行**Build**总菜单中的**Set Active Configuration**子菜单命令，便可实现其发行版的设置，然后再编译，即可生成发行版的应用程序。

另外，要在**VB**环境中用子线程以实现高速、连续数据采集与存盘，请务必使用**VB5.0**版本。当然如果您有**VB6.0**的最新版，也可以实现子线程操作。

C++ Builder:

要使用如下函数一个关键的问题是首先必须将我们提供的头文件

(DAM6000.H)写进您的源程序头部。如：#include “\Art\ DAM6000 \Include\ DAM6000.h”然后再将DAM6000.Lib库文件分别加入到您的C++ Builder工程中。其具体办法是选择C++ Builder集成开发环境中的工程(Project)菜单中的“添加”（Add to Project）命令，在弹出的对话框中分别选择文件类型：Library file (*.lib) , 即可选择DAM6000.Lib文件。该文件的路径为用户安装驱动程序后其子目录Samples\C_Builder下。

## 5.2 串口函数原型说明

### 5.2.1 串口设备对象管理函数原型说明

◆ 创建设备对象

***Visual C++ & C++Builder：***

HANDLE CreateCOM (LONG lPortNum)

功能：创建设备对象。

参数：

lPortNum 串口号，例如串口1，2，3，4，……

| 常量名 | 常量值 | 功能定义 |
|---|---|---|
| DAM6000_COM1 | 0x01 | 串口1 |
| DAM6000_COM2 | 0x02 | 串口2 |
| DAM6000_COM3 | 0x03 | 串口3 |
| DAM6000_COM4 | 0x04 | 串口4 |

返回值：返回一个串口句柄，以后便可用此句柄对串口进行操作。

相关函数：InitCOM　　ReleaseCOM

◆ 初始化串口参数

***Visual C++ & C++Builder：***

BOOL InitCOM ( HANDLE hDevice,
LONG lBaud,
BOOL bCheck,
LONG lTimeOut = DAM600_DEFAULT_TIMEOUT)

功能：初始化串口参数。

参数：

hDevice 由CreateCOM返回的串口句柄。

lBaud 用来初始化串口的波特率。

| 常量名 | 常量值 | 功能定义 |
|---|---|---|
| DAM6000_BAUD_1200 | 0x00 | 1200波特率 |
| DAM6000_BAUD_2400 | 0x01 | 2400波特率 |
| DAM6000_BAUD_4800 | 0x02 | 4800波特率 |
| DAM6000_BAUD_9600 | 0x03 | 9600波特率 |
| DAM6000_BAUD_19200 | 0x04 | 19200波特率 |
| DAM6000_BAUD_38400 | 0x05 | 38400波特率 |
| DAM6000_BAUD_57600 | 0x06 | 57600波特率 |
| DAM6000_BAUD_115200 | 0x07 | 115200波特率 |

bCheck 初始化串口是否有校验。

lTimeOut 初始化串口的超时时间。主要用于接收数据，如果为-1，则使用默认超时时间。

返回值：若成功，则返回TRUE，否则返回FALSE。

相关函数：CreateCOM　　ReleaseCOM

◆ 释放设备对象

***Visual C++ & C++Builder：***

BOOL ReleaseCOM (HANDLE hDevice)

功能：释放设备对象。

参数：hDevice 由CreateCOM返回的串口句柄。

返回值：若成功，则返回TRUE，否则返回FALSE。

相关函数：CreateCOM

### 5.2.2 串口设备配置获取/修改函数原型说明

♦ 读取模块信息

***Visual C++ & C++Builder：***

```
BOOL GetDevInfoCOM (HANDLE hDevice,
                    PDAM6000_DEVICE_INFO pInfo,
                    LONG lDeviceID)
```

功能：读取模块信息(类型、地址、波特率、校验)。

参数：

hDevice 由CreateCOM返回的串口句柄。

pInfo 设备信息结构体。DAM6000_DEVICE_INFO结构体如下：

```
typedef struct _DAM6000_DEVICE_INFO
{
    LONG    VendorID;       // 底板设备号
    LONG    UnitID;         // 底板拨码开关号
    LONG    BaudRate;       // 波特率
    LONG    DataBit;        // 数据位
    LONG    bParity;        // 有无校验
    LONG    StopBits;       // 停止位
} DAM6000_DEVICE_INFO, *PDAM6000_DEVICE_INFO;
```

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE，否则返回FALSE。

相关函数：CreateCOM　SetDevInfoCOM　ReleaseCOM

♦修改模块信息

***Visual C++ & C++Builder：***

```
BOOL SetDevInfoCOM (HANDLE hDevice,
                    DAM6000_DEVICE_INFO Info ,
                    LONG lDeviceID)
```

功能：修改模块信息(地址、波特率、校验)。

参数：

hDevice 由CreateCOM返回的串口句柄。

Info 设备信息结构体。DAM6000_DEVICE_INFO结构体如下：

```
typedef struct _DAM6000_DEVICE_INFO
{
    LONG    VendorID;       // 底板设备号
    LONG    UnitID;         // 底板拨码开关号
    LONG    BaudRate;       // 波特率
    LONG    DataBit;        // 数据位
    LONG    bParity;        // 有无校验
    LONG    StopBits;       // 停止位
} DAM6000_DEVICE_INFO, *PDAM6000_DEVICE_INFO;
```

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE，否则返回FALSE。

相关函数：CreateCOM　GetDevInfoCOM　ReleaseCOM

## 5.3 远程配置(以太网)函数原型说明

### 5.3.1 设备对象管理函数

♦**创建设备**

HANDLE CreateENet(char szIP[],
LONG lPort,
LONG lSendTimeout,
LONG lRcvTimeout)

功能：创建设备。

参数：

szIP 设备IP(如"192.168.1.2")。

lPort TCP/UDP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：若成功，则返回TRUE，否则返回FALSE。

相关函数：ReleaseENet

♦ 释放设备对象

***Visual C++ & C++Builder：***

BOOL ReleaseENet (HANDLE hDevice)

功能：释放设备对象。

参数：hDevice 由CreateENet返回的串口句柄。

返回值：若成功，则返回TRUE，否则返回FALSE。

相关函数：CreateENet

### 5.3.2 模块信息取得/修改函数原型说明

♦ 获得设备的网络配置信息

***Visual C++ & C++Builder：***

BOOL GetENetworkConfig (HANDLE hDevice,
PDEVICE_NET_INFO pNetInfo)

功能：获得设备的网络配置信息。

参数：

hDevice 由CreateENet返回的串口句柄。

pNetInfo网络配置信息。

返回值：若成功，则返回TRUE，否则返回FALSE。

相关函数：CreateENet SetENetworkConfig ReleaseENet

♦ 设置设备的网络配置信息

***Visual C++ & C++Builder：***

BOOL SetENetworkConfig (HANDLE hDevice,
PDEVICE_NET_INFO NetInfo)

功能：设置设备的网络配置信息。

参数：

hDevice 由CreateENet返回的串口句柄。
NetInfo网络配置信息。
返回值：若成功，则返回TRUE，否则返回FALSE。
相关函数：CreateENet　　GetENetworkConfig　　ReleaseENet

### 5.3.3 查询模块设备是否在线(DHCP)

♦ 创建设备

***Visual C++ & C++Builder：***

```
int    SearchDevice(char szIP[],
                    LONG    lPort,
                    PLONG   plCount,
                    LONG    lSendTimeout,
                    LONG    lRcvTimeout)
```

功能：创建设备。
参数：
szIP 设备IP列表。
lPort UDP端口。
plCount 设备数量。
lSendTimeout 发送数据的超时间隔。
lRcvTimeout 接收数据的超时间隔。
返回值：若成功，则返回创建的设备数。
相关函数：CreateENet　　ReleaseENet

♦ 创建设备

***Visual C++ & C++Builder：***

```
int    SearchDeviceEx (char    szIP[],
                       LONG    lTCPPort[],
                       LONG    lUDPPort,
                       PLONG   plCount,
                       LONG    lSendTimeout,
                       LONG    lRcvTimeout)
```

功能：创建设备。
参数：
szIP 设备IP列表。
lTCPPort TCP端口。
lUDPPort UDP端口。
plCount 设备数量。
lSendTimeout 发送数据的超时间隔。
lRcvTimeout 接收数据的超时间隔。
返回值：若成功，则返回创建的设备数。
相关函数：CreateENet　　ReleaseENet

## 5.4 读取模块配置参数

♦ 获得模块配置参数

***Visual C++ & C++Builder：***

BOOL GetModuleConfig (HANDLE hDevice,
PMODULE_CONFIG_INFO pModulecfg,
LONG lDeviceID = 0)

功能：获得模块配置参数。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

pModulecfg 模块配置信息。

lDeviceID 设备地址。

返回值：若成功，则返回TRUE，否则返回FALSE。

相关函数： CreateCOM ReleaseCOM
或 CreateENet ReleaseENet

## 5.5 AD 模拟量输入操作函数原型说明

♦ 读取**AD**模拟量输入

***Visual C++ & C++Builder：***

BOOL ReadDeviceAD(HANDLE hDevice,
WORD wpADBuffer[],
LONG lFirstChannel = 0,
LONG lLastChannel = 0,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：读取AD模拟量输入。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

wpADBuffer 接收AD数据的用户缓冲区。

lFirstChannel 首通道。

lLastChannel 末通道。注意：末通道要大于等于首通道，wpADBuffer最好大于等于lLastChannel - lFirstChannel +1。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM ReleaseCOM
或 CreateENet ReleaseENet

♦ 获得模拟量输入量程

***Visual C++ & C++Builder*:**

BOOL GetRangeAD (HANDLE hDevice,
LONG lADRange[],
LONG lFirstChannel = 0,

LONG lLastChannel = 0,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：获得模拟量输入模式。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

lADRange AD量程。

电压类型有：

| 常量名 | 常量值 | 功能定义 |
|---|---|---|
| DAM6000_VOLT_N15_P15 | 0x01 | ±15mV |
| DAM6000_VOLT_N50_P50 | 0x02 | ±50mV |
| DAM6000_VOLT_N100_P100 | 0x03 | ±100mV |
| DAM6000_VOLT_N150_P150 | 0x04 | ±150mV |
| DAM6000_VOLT_N500_P500 | 0x05 | ±500mV |
| DAM6000_VOLT_N1_P1 | 0x06 | ±1V |
| DAM6000_VOLT_N25_P25 | 0x07 | ±2.5V |
| DAM6000_VOLT_N5_P5 | 0x08 | ±5V |
| DAM6000_VOLT_N10_P10 | 0x09 | ±10V |
| DAM6000_VOLT_N0_P5 | 0x0D | 0～5V |
| DAM6000_VOLT_N0_P10 | 0x0E | 0～10V |
| DAM6000_VOLT_N0_P25 | 0x0F | 0～2.5V |

电流类型有：

| 常量名 | 常量值 | 功能定义 |
|---|---|---|
| DAM6000_CUR_N0_P10 | 0x00 | 0～10mA |
| DAM6000_CUR_N20_P20 | 0x0A | ±20mA |
| DAM6000_CUR_N0_P20 | 0x0B | 0～20mA |
| DAM6000_CUR_N4_P20 | 0x0C | 4～20mA |
| DAM6000_CUR_N0_P22 | 0x80 | 0～22mA |

lFirstChannel 首通道。

lLastChannel 末通道。注意：末通道要大于等于首通道。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM SetRangeAD ReleaseCOM
或 CreateENet SetRangeAD ReleaseENet

◆ 设置模拟量输入量程

***Visual C++ & C++Builder***：

BOOL SetRangeAD(HANDLE hDevice,
const LONG lADRange[],
LONG lFirstChannel = 0,
LONG lLastChannel = 0,

LONG lSlot = 0,
LONG lDeviceID = 0)

功能：设置模拟量输入量程。

hDevice 由CreateCOM或CreateENet返回的串口句柄。

lADRange AD量程。

电压类型有：

| 常量名 | 常量值 | 功能定义 |
|---|---|---|
| DAM6000_VOLT_N15_P15 | 0x01 | ±15mV |
| DAM6000_VOLT_N50_P50 | 0x02 | ±50mV |
| DAM6000_VOLT_N100_P100 | 0x03 | ±100mV |
| DAM6000_VOLT_N150_P150 | 0x04 | ±150mV |
| DAM6000_VOLT_N500_P500 | 0x05 | ±500mV |
| DAM6000_VOLT_N1_P1 | 0x06 | ±1V |
| DAM6000_VOLT_N25_P25 | 0x07 | ±2.5V |
| DAM6000_VOLT_N5_P5 | 0x08 | ±5V |
| DAM6000_VOLT_N10_P10 | 0x09 | ±10V |
| DAM6000_VOLT_N0_P5 | 0x0D | 0～5V |
| DAM6000_VOLT_N0_P10 | 0x0E | 0～10V |
| DAM6000_VOLT_N0_P25 | 0x0F | 0～2.5V |

电流类型有：

| 常量名 | 常量值 | 功能定义 |
|---|---|---|
| DAM6000_CUR_N0_P10 | 0x00 | 0～10mA |
| DAM6000_CUR_N20_P20 | 0x0A | ±20mA |
| DAM6000_CUR_N0_P20 | 0x0B | 0～20mA |
| DAM6000_CUR_N4_P20 | 0x0C | 4～20mA |
| DAM6000_CUR_N0_P22 | 0x80 | 0～22mA |

lFirstChannel 首通道。

lLastChannel 末通道。注意：末通道要大于等于首通道。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM GetRangeAD ReleaseCOM
或 CreateENet GetRangeAD ReleaseENet

◆ 调整AD零点

***Visual C++ & C++Builder***:

BOOL SetZeroAD (HANDLE hDevice,
LONG lADZero,
LONG Channel = 0,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：调整AD零点。

参数：
hDevice 由CreateCOM或CreateENet返回的串口句柄。
lADZero AD零点。
Channel 通道号。
lSlot 卡槽号。
lDeviceID 模块地址，用于区别不同的模块。
返回值：若成功，则返回TRUE， 否则返回FALSE。
相关函数： CreateCOM ReadDeviceAD ReleaseCOM
或 CreateENet ReadDeviceAD ReleaseENet

- 调整AD满度

***Visual C++ & C++Builder***:

```
BOOL SetFullAD (HANDLE hDevice,
                LONG   lADFull,
                LONG   Channel = 0,
                LONG   lSlot = 0,
                LONG   lDeviceID = 0)
```

功能：调整AD满度。
参数：
hDevice 由CreateCOM或CreateENet返回的串口句柄。
lADFull AD满度。
lChannel 通道号。
lSlot 卡槽号。
lDeviceID 模块地址，用于区别不同的模块。
返回值：若成功，则返回TRUE， 否则返回FALSE。
相关函数： CreateCOM ReadDeviceAD ReleaseCOM
或 CreateENet ReadDeviceAD ReleaseENet

## 5.6 DA 模拟量输出操作函数原型说明

- 回读模拟量输出值

***Visual C++ & C++Builder***:

```
BOOL GetDeviceDA(HANDLE hDevice,
                 PLONG plDAValue,
                 LONG  lChannel = 0,
                 LONG  lSlot = 0,
                 LONG  lDeviceID = 0)
```

功能：回读模拟量输出值。
参数：
hDevice 由CreateCOM或CreateENet返回的串口句柄。
plDAValue模块当前的模块理输出值，取值为(0-4095)。与电压的换算关系如下表：

| 量程(伏) | 计算机语言换算公式 | Lsb取值范围 |
|---|---|---|
| ±5000mV | Lsb = Volt / (10000 / 4096) + 2048 | [0，4095] |
| 0～10000mV | Lsb = Volt/ (10000 / 4096 ) | [0，4095] |

| | | |
|---|---|---|
| ±10000mV | Lsb = Volt/ (20000 / 4096 ) + 2048 | [0，4095] |

lChannel 通道号。
lSlot 卡槽号。
lDeviceID 模块地址，用于区别不同的模块。
返回值：若成功，则返回TRUE， 否则返回FALSE。
相关函数： CreateCOM WriteDeviceDA ReleaseCOM
或 CreateENet WriteDeviceDA ReleaseENet

- 设定单通道DA

***Visual C++ & C++Builder：***

```
BOOL WriteDeviceDA(HANDLE hDevice,
                   LONG    DAValue,
                   LONG    lChannel = 0,
                   LONG    lSlot = 0,
                   LONG    lDeviceID = 0)
```

功能：设定单通道DA。
参数：
hDevice 由CreateCOM或CreateENet返回的串口句柄。
DAValue DA输出当前值。
lChannel 通道号。
lSlot 卡槽号。
lDeviceID 模块地址，用于区别不同的模块。
返回值：若成功，则返回TRUE， 否则返回FALSE。
相关函数： CreateCOM GetDeviceDA ReleaseCOM
或 CreateENet GetDeviceDA ReleaseENet

- 读取模拟量输出量程

***Visual C++ & C++Builder:***

```
BOOL GetRangeDA (HANDLE hDevice,
                 PLONG    plDARange,
                 LONG     lChannel = 0,
                 LONG     lSlot = 0,
                 LONG     lDeviceID = 0)
```

功能：读取模拟量输出量程。
参数：
hDevice 由CreateCOM或CreateENet返回的串口句柄。
plDARange 输出量程。
lChannel 通道号。
lSlot 卡槽号。
lDeviceID 模块地址，用于区别不同的模块。
返回值：若成功，则返回TRUE， 否则返回FALSE。
相关函数： CreateCOM SetRangeDA ReleaseCOM
或 CreateENet SetRangeDA ReleaseENet

♦ 设置模拟量输出量程

***Visual C++ & C++Builder*:**

BOOL SetRangeDA (HANDLE hDevice,
LONG lDARange,
LONG lChannel = 0,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：设置模拟量输出量程。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

lDARange输出量程。

lChannel 通道号。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM GetRangeDA ReleaseCOM
或 CreateENet GetRangeDA ReleaseENet

♦ 调整DA零点

***Visual C++ & C++Builder*:**

BOOL SetZeroDA ( HANDLE hDevice,
LONG lDAZero,
LONG lChannel = 0,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：调整DA零点。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

lDAZero DA零点。

lChannel 通道号。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM GetDeviceDA ReleaseCOM
或 CreateENet GetDeviceDA ReleaseENet

♦ 调整DA满度

***Visual C++ & C++Builder*:**

BOOL SetFullDA ( HANDLE hDevice,
LONG lDAFull,
LONG lChannel = 0,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：调整DA满度。
参数：
hDevice 由CreateCOM或CreateENet返回的串口句柄。
lDAFull DA满度。
lChannel 通道号。
lSlot 卡槽号。
lDeviceID 模块地址，用于区别不同的模块。
返回值：若成功，则返回TRUE， 否则返回FALSE。
相关函数： CreateCOM GetDeviceDA ReleaseCOM
或 CreateENet GetDeviceDA ReleaseENet

## 5.7 DI 数字量输入操作函数原型说明

◆ 读取开关量输入

***Visual C++ & C++Builder***:

```
BOOL GetDeviceDI( HANDLE hDevice,
                  BYTE  byDIStatus[],
                  LONG lFirstChannel = 0,
                  LONG lLastChannel = 0,
                  LONG lSlot = 0,
                  LONG lDeviceID = 0)
```

功能：读取开关量输入。
参数：
hDevice 由CreateCOM或CreateENet返回的串口句柄。
byDIStatus DI值。
lFirstChannel 首通道。
lLastChannel 末通道。注意：末通道要大于等于首通道。
lSlot 卡槽号。
lDeviceID 模块地址，用于区别不同的模块。
返回值：若成功，则返回TRUE， 否则返回FALSE。
相关函数： CreateCOM ReleaseCOM
或 CreateENet ReleaseENet

## 5.8 DO 数字量输出操作函数原型说明

◆ 回读开关量输出

***Visual C++ & C++Builder:***

```
BOOL GetDeviceDO( HANDLE hDevice,
                  BYTE    byDOSts[],
                  LONG    lFirstChannel = 0,
                  LONG    lLastChannel = 0,
                  LONG    lSlot = 0,
                  LONG    lDeviceID = 0)
```

功能：回读开关量输出。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

byDOSts 当前DO输出值。

lFirstChannel 首通道。

lLastChannel 末通道。注意：末通道要大于等于首通道。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM SetDeviceDO ReleaseCOM

或 CreateENet SetDeviceDO ReleaseENet

- 设置数字量输出的值

***Visual C++ & C++Builder:***

```
BOOL SetDeviceDO( HANDLE hDevice,
                  BYTE    byDOSts[],
                  LONG    lFirstChannel = 0,
                  LONG    lLastChannel = 0,
                  LONG    lSlot = 0,
                  LONG    lDeviceID = 0)
```

功能：设置数字量输出的值。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

byDOSts DO输出值。

lFirstChannel 首通道。

lLastChannel 末通道。注意：末通道要大于等于首通道。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM GetDeviceDO ReleaseCOM

或 CreateENet GetDeviceDO ReleaseENet

- 获得数字量输出的上电值

***Visual C++ & C++Builder:***

```
BOOL GetPowerOnValueDO( HANDLE hDevice,
                        BYTE    byPowerOn[],
                        LONG    lFirstChannel = 0,
                        LONG    lLastChannel = 0,
                        LONG    lSlot = 0,
                        LONG    lDeviceID = 0)
```

功能：获得数字量输出的上电值。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

byPowerOn 上电值。

lFirstChannel 首通道。

lLastChannel 末通道。注意：末通道要大于等于首通道。
lSlot 卡槽号。
lDeviceID 模块地址，用于区别不同的模块。
返回值：若成功，则返回TRUE， 否则返回FALSE。
相关函数： CreateCOM SetPowerOnValueDO ReleaseCOM
或 CreateENet SetPowerOnValueDO ReleaseENet

◆ 设置数字量输出的上电值

***Visual C++ & C++Builder:***

```
BOOL SetPowerOnValueDO( HANDLE hDevice,
                        BYTE   byPowerOn[],
                        LONG   lFirstChannel = 0,
                        LONG   lLastChannel = 0,
                        LONG   lSlot = 0,
                        LONG   lDeviceID = 0)
```

功能：设置数字量输出的上电值。
参数：
hDevice 由CreateCOM或CreateENet返回的串口句柄。
byPowerOn 上电值。
lFirstChannel 首通道。
lLastChannel 末通道。注意：末通道要大于等于首通道。
lSlot 卡槽号。
lDeviceID 模块地址，用于区别不同的模块。
返回值：若成功，则返回TRUE， 否则返回FALSE。
相关函数： CreateCOM GetPowerOnValueDO ReleaseCOM
或 CreateENet GetPowerOnValueDO ReleaseENet

◆ 获得数字量输出的安全值

***Visual C++ & C++Builder:***

```
BOOL GetSafeValueDO ( HANDLE hDevice,
                      BYTE   byDOSafe[],
                      LONG   lFirstChannel = 0,
                      LONG   lLastChannel = 0,
                      LONG   lSlot = 0,
                      LONG   lDeviceID = 0)
```

功能：获得数字量输出的安全值。
参数：
hDevice 由CreateCOM或CreateENet返回的串口句柄。
byDOSafe 安全值。
lFirstChannel 首通道。
lLastChannel 末通道。注意：末通道要大于等于首通道。
lSlot 卡槽号。
lDeviceID 模块地址，用于区别不同的模块。
返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM SetSafeValueDO ReleaseCOM
或 CreateENet SetSafeValueDO ReleaseENet

♦ 设置数字量输出的安全值

***Visual C++ & C++Builder:***

BOOL SetSafeValueDO( HANDLE hDevice,
BYTE byDOSafe[],
LONG lFirstChannel = 0,
LONG lLastChannel = 0,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：设置数字量输出的安全值。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

byDOSafe 安全值。

lFirstChannel 首通道。

lLastChannel 末通道。注意：末通道要大于等于首通道。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM GetSafeValueDO ReleaseCOM
或 CreateENet GetSafeValueDO ReleaseENet

## 5.9 计数器函数原型说明

♦ 对各个计数器进行参数设置

***Visual C++ & C++Builder:***

BOOL SetCounterMode(HANDLE hDevice,
PDAM6000_PARA_CNT pCNTPara,
LONG lChannel = 0,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：对各个计数器进行参数设置。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

pCNTPara 基于各通道的计数器参数。

lChannel 通道号。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter
StopCounter GetCounterCurVal ResetCounter
ReleaseCOM

或 CreateENet SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter
StopCounter GetCounterCurVal ResetCounter
ReleaseENet

◆ 设置滤波系数

***Visual C++ & C++Builder:***

BOOL SetCounterFilterValue (HANDLE hDevice,
LONG lFilterValue,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：设置滤波系数。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

lFilterValue 滤波系数。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter
StopCounter GetCounterCurVal ResetCounter
ReleaseCOM
或 CreateENet SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter
StopCounter GetCounterCurVal ResetCounter
ReleaseENet

◆ 清计数器状态

***Visual C++ & C++Builder:***

BOOL ClrCounterSts (HANDLE hDevice,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：清计数器状态。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter
StopCounter GetCounterCurVal ResetCounter
ReleaseCOM
或 CreateENet SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter

StopCounter GetCounterCurVal ResetCounter
ReleaseENet

♦ 获得计数器设备硬件参数状态

***Visual C++ & C++Builder:***

BOOL GetCounterSts (HANDLE hDevice,
PDAM6000_CNT_STATUS pStsCNT,
LONG lChannel = 0,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：获得计数器设备硬件参数状态。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

pStsCNT 返回值。

lChannel 通道号。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter
StopCounter GetCounterCurVal ResetCounter
ReleaseCOM

或 CreateENet SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter
StopCounter GetCounterCurVal ResetCounter
ReleaseENet

♦ 启动计数器计数

***Visual C++ & C++Builder:***

BOOL StartCounter (HANDLE hDevice,
LONG lChannel = 0,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：启动计数器计数。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

lChannel 通道号。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter
StopCounter GetCounterCurVal ResetCounter
ReleaseCOM

或 CreateENet SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter
StopCounter GetCounterCurVal ResetCounter
ReleaseENet

♦ 停止计数器计数

***Visual C++ & C++Builder:***

BOOL StopCounter (HANDLE hDevice,
LONG lChannel = 0,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：停止计数器计数。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

lChannel 通道号。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter
StopCounter GetCounterCurVal ResetCounter
ReleaseCOM
或 CreateENet SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter
StopCounter GetCounterCurVal ResetCounter
ReleaseENet

♦ 取得计数器当前值

***Visual C++ & C++Builder:***

BOOL GetCounterCurVal (HANDLE hDevice,
PULONG pulCNTVal,
LONG lChannel = 0,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：取得计数器当前值。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

pulCNTVal 计数值。

lChannel 通道号。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter

StopCounter GetCounterCurVal ResetCounter
ReleaseCOM
或 CreateENet SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter
StopCounter GetCounterCurVal ResetCounter
ReleaseENet

◆ 计数器复位

***Visual C++ & C++Builder:***

BOOL ResetCounter (HANDLE hDevice,
LONG lChannel = 0,
LONG lSlot = 0,
LONG lDeviceID = 0)

功能：计数器复位。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

lChannel 通道号。

lSlot 卡槽号。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter
StopCounter GetCounterCurVal ResetCounter
ReleaseCOM
或 CreateENet SetCounterMode SetCounterFilterValue
ClrCounterSts GetCounterSts StartCounter
StopCounter GetCounterCurVal ResetCounter
ReleaseENet

## 5.10 模块时间函数原型说明

◆ 取得模块时间

***Visual C++ & C++Builder:***

BOOL GetTime (HANDLE hDevice,
PDAM6000_TIME pTime,
LONG lDeviceID = 0)

功能：取得模块时间。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

pTime 时间。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM SetTime ReleaseCOM
或 CreateENet SetTime ReleaseENet

♦ 设置模块时间

***Visual C++ & C++Builder:***

BOOL SetTime (HANDLE hDevice,
PDAM6000_TIME pTime,
LONG lDeviceID = 0)

功能：设置模块时间。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

pTime 时间。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM GetTime ReleaseCOM
或 CreateENet GetTime ReleaseENet

## 5.11 读取错误信息函数原型说明

♦ **读取最后一个错误函数**

***Visual C++ & C++ Builder:***

LONG GetLastErr (HANDLE hDevice,
LONG lDeviceID = 0)

功能：读取最后一个错误函数。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

lDeviceID 模块地址，用于区别不同的模块。

返回值：若成功，则返回错误信息。

相关函数： CreateCOM ReleaseCOM
或 CreateENet ReleaseENet

## 5.12 注册表函数原型说明

♦ **保存 IP 到注册表**

***Visual C++ & C++ Builder:***

BOOL SaveIPAddress (char szIP[])

功能：保存 IP 到注册表。

参数：szIP IP 地址。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM LoadIPAddress ReleaseCOM
或 CreateENet LoadIPAddress ReleaseENet

♦ **载入 IP 到应用程序**

***Visual C++ & C++ Builder:***

BOOL LoadIPAddress (char szIP[])

功能：载入 IP 到应用程序。

参数：szIP IP 地址。
返回值：若成功，则返回TRUE， 否则返回FALSE。
相关函数： CreateCOM SaveIPAddress ReleaseCOM
或 CreateENet SaveIPAddress ReleaseENet

## 5.13 Modbus 协议函数原型说明

♦ 读继电器状态

***Visual C++ & C++ Builder:***

```
BOOL ReadCoils (HANDLE hDevice,
                int    addr,
                int    len,
                BYTE   bCoilsFlag[],
                int    nDeviceID = 0)
```

功能：读继电器状态。
参数：
hDevice 由CreateCOM或CreateENet返回的串口句柄。
addr 地址。
len 长度。
bCoilsFlag 继电器状态。
nDeviceID 设备地址。
返回值：若成功，则返回TRUE， 否则返回FALSE。
相关函数： CreateCOM ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseCOM
或 CreateENet ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseENet

♦ 读开关量输入

***Visual C++ & C++ Builder:***

```
BOOL ReadDiscretes (HANDLE  hDevice,
                    int     addr,
                    int     len,
                    BYTE    bCoilsFlag[],
                    int     nDeviceID = 0)
```

功能：读开关量输入。
参数：
hDevice 由CreateCOM或CreateENet返回的串口句柄。
addr 地址。
len 长度。
bDIState 开关量状态。

nDeviceID 设备地址。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseCOM
或 CreateENet ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseENet

◆ 读保持寄存器

***Visual C++ & C++ Builder:***

```
BOOL ReadMultiRegs (HANDLE hDevice,
                    int    addr,
                    int    len,
                    BYTE   buf [],
                    int    nDeviceID = 0)
```

功能：读保持寄存器。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

addr 地址。

len 长度。

buf 数据。

nDeviceID 设备地址。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseCOM
或 CreateENet ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseENet

◆ 读输入寄存器

***Visual C++ & C++ Builder:***

```
BOOL ReadInputRegs (HANDLE hDevice,
                    int    addr,
                    int    len,
                    BYTE   buf [],
                    int    nDeviceID = 0)
```

功能：读输入寄存器。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。
addr 地址。
len 长度。
buf 数据。
nDeviceID 设备地址。
返回值：若成功，则返回TRUE， 否则返回FALSE。
相关函数： CreateCOM ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseCOM
或 CreateENet ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseENet

♦设置单个继电器

***Visual C++ & C++ Builder:***

```
BOOL WriteCoil (HANDLE hDevice,
                int    addr,
                BYTE   status,
                int    nDeviceID = 0)
```

功能：设置单个继电器。
参数：
hDevice 由CreateCOM或CreateENet返回的串口句柄。
addr 地址。
status 数据。
nDeviceID 设备地址。
返回值：若成功，则返回 TRUE， 否则返回 FALSE。
相关函数： CreateCOM ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseCOM
或 CreateENet ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseENet

♦设置单个保持寄存器

***Visual C++ & C++ Builder:***

```
BOOL WriteSingleReg (HANDLE hDevice,
                     int    addr,
                     short  val,
                     int    nDeviceID = 0)
```

功能：设置单个保持寄存器。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

addr 地址。

val 数据。

nDeviceID 设备地址。

返回值：若成功，则返回 TRUE， 否则返回 FALSE。

相关函数： CreateCOM ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseCOM
或 CreateENet ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseENet

- 设置多个继电器

***Visual C++ & C++ Builder:***

```
BOOL ForceMultiCoils (HANDLE    hDevice,
                      int       addr,
                      int       len,
                      BYTE      bDOState [],
                      int       nDeviceID = 0)
```

功能：设置多个继电器。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

addr 地址。

len 长度。

bDOState 数据。

nDeviceID 设备地址。

返回值：若成功，则返回TRUE， 否则返回FALSE。

相关函数： CreateCOM ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseCOM
或 CreateENet ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseENet

- 设置多个保持寄存器

***Visual C++ & C++ Builder:***

```
BOOL WriteMultiRegs (HANDLE    hDevice,
                     int       addr,
                     int       len,
```

BYTE buf [],
int nDeviceID = 0)

功能：设置多个保持寄存器。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

addr 地址。

len 长度。

buf 数据。

nDeviceID 设备地址。

返回值：若成功，则返回 TRUE， 否则返回 FALSE。

相关函数： CreateCOM ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseCOM
或 CreateENet ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseENet

- 读取文件记录

***Visual C++ & C++ Builder:***

BOOL ReadRegFile (HANDLE hDevice,
int index,
int addr,
int len,
BYTE buf [],
int nDeviceID = 0)

功能：读取文件记录。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

index 文件号。

addr 地址。

len 长度。

buf 数据。

nDeviceID 设备地址。

返回值：若成功，则返回 TRUE， 否则返回 FALSE。

相关函数： CreateCOM ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseCOM
或 CreateENet ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseENet

♦ 写文件记录

***Visual C++ & C++ Builder:***

```
BOOL WriteRegFile (HANDLE   hDevice,
                   int      index,
                   int      addr,
                   int      len,
                   BYTE     buf [],
                   int      nDeviceID = 0)
```

功能：写文件记录。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

index 文件号。

addr 地址。

len 长度。

buf 数据。

nDeviceID 设备地址。

返回值：若成功，则返回 TRUE， 否则返回 FALSE。

相关函数： CreateCOM ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseCOM
或 CreateENet ReadCoils ReadDiscretes
ReadMultiRegs ReadInputRegs WriteCoil
WriteSingleReg ForceMultiCoils WriteMultiRegs
ReadRegFile WriteRegFile ReleaseENet

## 5.14 输入输出任意二进制字符

♦ 直接写设备

***Visual C++ & C++ Builder:***

```
int WriteDeviceBYTE (HANDLE hDevice,
                     BYTE*  byWriteBuf,
                     long llength,
                     long timeout=100)
```

功能：直接写设备。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

byWriteBuf 写的数据。

llength 数据长度。

timeout 超时范围(mS)。

返回值：返回写的字节数。

相关函数： CreateCOM ReadDeviceBYTE ReleaseCOM
或 CreateENet ReadDeviceBYTE ReleaseENet

◆ 直接读设备

***Visual C++ & C++ Builder:***

int ReadDeviceBYTE (HANDLE hDevice,
BYTE* byReadBuf,
long llength,
long timeout=100)

功能：直接读设备。

参数：

hDevice 由CreateCOM或CreateENet返回的串口句柄。

byReadBuf 读取的数据。

llength 数据长度。

timeout 超时范围(mS)。

返回值：返回读取的字节数。

相关函数： CreateCOM WriteDeviceBYTE ReleaseCOM
或 CreateENet WriteDeviceBYTE ReleaseENet

## 5.15 结构体介绍

### 5.15.1 串口设备基本信息的结构体介绍（DAM6000_DEVICE_INFO）

**Visual C++ & C++Builder:**

```
typedef struct _DAM6000_DEVICE_INFO
{
    LONG    VendorID;           // 底板设备号
    LONG    UnitID;             // 底板拨码开关号
    LONG    BaudRate;           // 波特率
    LONG    DataBit;            // 数据位
    LONG    bParity;            // 有无校验
    LONG    StopBits;           // 停止位
} DAM6000_DEVICE_INFO, *PDAM6000_DEVICE_INFO;
```

BaudRate

| 常量名 | 常量值 | 功能定义 |
|---|---|---|
| DAM6000_BAUD_1200 | 0x00 | 1200波特率 |
| DAM6000_BAUD_2400 | 0x01 | 2400波特率 |
| DAM6000_BAUD_4800 | 0x02 | 4800波特率 |
| DAM6000_BAUD_9600 | 0x03 | 9600波特率 |
| DAM6000_BAUD_19200 | 0x04 | 19200波特率 |
| DAM6000_BAUD_38400 | 0x05 | 38400波特率 |
| DAM6000_BAUD_57600 | 0x06 | 57600波特率 |
| DAM6000_BAUD_115200 | 0x07 | 115200波特率 |

### 5.15.2 以太网设备网络信息的结构体介绍

```
typedef struct _DEVICE_NET_INFO
{
    char      szIP[16];              // IP 地址, "192.168.2.70"
    char SubnetMask[16];             // 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
    LONG    lTCPPort;                // TCP 端口号
}DEVICE_NET_INFO, *PDEVICE_NET_INFO;
```

### 5.15.3 时间结构体介绍

```
typedef struct _DAM6000_TIME
{
    LONG lYear;
    LONG lMonth;
    LONG lDay;
    LONG lWeek;
    LONG lHour;
    LONG lMinute;
    LONG lSecond;
}DAM6000_TIME, *PDAM6000_TIME;
```

### 5.15.4 模块配置结构体介绍

```
typedef struct _MODULE_CONFIG_INFO
{
    LONG    lDeviceType[8];                  // 卡槽中的模块设备类型
}MODULE_CONFIG_INFO, *PMODULE_CONFIG_INFO;
```

### 5.15.5 计数器参数配置结构体介绍

```
typedef struct _DAM6000_PARA_CNT                 // 基于各通道的计数器参数结构体
{
    LONG WorkMode;               // 计数器/频率工作模式
    LONG FreqBuildTime;          // 测频器建立时间, 单位: s
    ULONG InitVal;               // 计数器初始值
    ULONG MaxVal;                // 计数器最大值
    ULONG MinVal;                // 计数器最小值
} DAM6000_PARA_CNT, *PDAM6000_PARA_CNT;
```

WorkMode

| 常量名 | 常量值 | 功能定义 |
|---|---|---|
| DAM6000_WORKMODE_CNT | 0x00 | 计数器 |
| DAM6000_WORKMODE_FREQ | 0x01 | 频率器 |

FreqBuildTime

| 常量名 | 常量值 | 功能定义 |
|---|---|---|
| DAM6000_FREQTIME_10MS | 0x00 | 测频定时 10mS |
| DAM6000_FREQTIME_1S | 0x01 | 测频定时 1S |

```
typedef struct _DAM6000_CNT_STATUS                // 计数器硬件参数状态结构体
{
    LONG bCNTOverflow;      // 计数器是否上溢
    LONG bCNTUnderflow;     // 计数器是否下溢
    LONG bCNTUpperlimit;    // 计数器是否超过上限
    LONG bCNTLowerlimit;    // 计数器是否超过下限
} DAM6000_CNT_STATUS, *PDAM6000_CNT_STATUS;
```

# 第六章、产品保修

在公司售出的产品包装中，用户将会找到这本说明书和板卡，同时还有产品质保卡。产品质保卡请用户务必妥善保存，当该产品出现问题需要维修时，请用户将产品质保卡同产品一起，寄回本公司，以便我们能尽快的帮用户解决问题。