RTU6103 远程终端设备

WIN2000/XP 驱动程序使用说明书

阿尔泰科技发展有限公司

产品研发部修订

请您务必阅读《使用纲要》，他会使您事半功倍!

目　录

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 RTUxxxx_则被省略。如 RTU6103_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 使用纲要

第一节、使用上层用户函数，高效、简单

如果您只关心通道及频率等基本参数，而不必了解复杂的硬件知识和控制细节，那么我们强烈建议您使用上层用户函数，它们就是几个简单的形如Win32 API的函数，具有相当的灵活性、可靠性和高效性。诸如InitDeviceAD、InitDeviceProAD 、 InitDeviceDmaAD 、 ReadDeviceProAD-Npt 等 。 而 底 层 用 户 函 数 如 WriteRegisterULong 、ReadRegisterULong、WritePortByte、ReadPortByte……则是满足了解硬件知识和控制细节、且又需要特殊复杂控制的用户。但不管怎样，我们强烈建议您使用上层函数（在这些函数中，您见不到任何设备地址、寄存器端口、中断号等物理信息，其复杂的控制细节完全封装在上层用户函数中。）对于上层用户函数的使用，您基本上不必参考硬件说明书，除非您需要知道板上插座等管脚分配情况。

第二节、如何管理设备

由于我们的驱动程序采用面向对象编程，所以要使用设备的一切功能，则必须首先用CreateDevice函数创建一个设备对象句柄hDevice，有了这个句柄，您就拥有了对该设备的绝对控制权。然后将此句柄作为参数传递给相应的驱动函数，如InitDeviceProAD可以使用hDevice句柄以程序查询方式初始化设备的AD部件，ReadDeviceProAD Npt函数可以用hDevice句柄实现对AD数据的采样读取等。最后可以通过ReleaseDevice将hDevice释放掉。

第三节、如何实现数字量的简便操作

当您有了hDevice设备对象句柄后，便可用SetDeviceDO函数实现数字量的输出操作，其各路数字量的输出状态由其bDOSts[16]中的相应元素决定。由GetDeviceDI函数实现数字量的输入操作，其各路数字量的输入状态由其bDOSts[16]中的相应元素决定。

第四节、哪些函数对您不是必须的

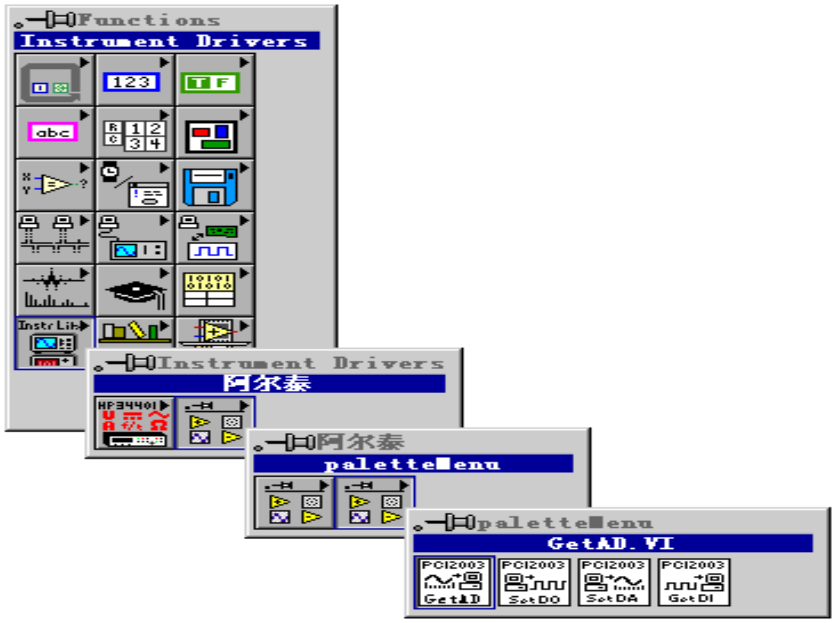
公共函数如CreateFileObject，WriteFile，ReadFile等一般来说都是辅助性函数，除非您要使用存盘功能。如果您使用上层用户函数访问设备，那么GetDeviceAddr，WriteRegisterByte，WriteRegisterWord，WriteRegisterULong，ReadRegisterByte，ReadRegisterWord，ReadRegisterULong等函数您可完全不必理会，除非您是作为底层用户管理设备。而WritePortByte，WritePortWord，WritePortULong，ReadPortByte，ReadPortWord，ReadPortULong则对RTU用户来讲，可以说完全是辅助性，它们只是对我公司驱动程序的一种功能补充，对用户额外提供的，它们可以帮助您在NT、Win2000 等操作系统中实现对您原有传统设备如ISA卡、串口卡、并口卡的访问，而没有这些函数，您可能在基于Windows NT架构的操作系统中无法继续使用您原有的老设备。

第三章 RTU 设备操作函数接口介绍

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的首末通道、采样频率等，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AD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InitDeviceProAD）告诉设备我要使用多少个通道，采样频率是多少赫兹等，然后便可以用ReadDeviceProAD Npt（或ReadDeviceProAD Half）函数指定每次采集的点数，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RTU总线复杂的控制协议，同是还可以省掉您许多繁琐的工作，比如您不用去了解RTU的资源配置空间、PNP即插即用管理，而只须用GetDeviceAddr函数便可以同时取得指定设备的物理基地址和虚拟线性基地址。这个时候您便可以用这个虚拟线性基地址，再根据硬件使用说明书中的各端口寄存器的功能说明，然后使用ReadRegisterULong和WriteRegisterULong对这些端口寄存器进行 32 位模式的读写操作，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将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，先明白自己是上层用户还是底层用户，因为在《设备驱动接口函数总列表》中的备注栏里明确注明了适用对象。

另外需要申明的是，在本章和下一章中列明的关于 LabView 的接口，均属于外挂式驱动接口，他是通过 LabView 的 Call Labrary Function 功能模板实现的。它的特点是除了自身的语法略有不同以外，每一个基于 LabView 的驱动图标与 Visual C++、Visual Basic、Delphi 等语言中每个驱动函数是一一对应的，其调用流程和功能是完全相同的。那么相对于外挂式驱动接口的另一种方式是内嵌式驱动。这种驱动是完全作为 LabView 编程环境中的紧密耦合的一部分，它可以直接从 LabView 的 Functions 模板中取得，如下图所示。此种方式更适合上层用户的需要，它的最大特点是方便、快捷、简单，而且可以取得它的在线帮助。关于 LabView 的外挂式驱动和内嵌式驱动更详细的叙述，请参考 LabView 的相关演示。



LabView 内嵌式驱动接口的获取方法

第一节、设备驱动接口函数总列表（每个函数省略了前缀“RTU6103_”）

函数名	函数功能	备注
① 设备对象管理函数		
CreateCOM(LONG lPortNum)	创建设备对象	上层及底层用户
InitCOM	初始与模块之间的通信参数	上层及底层用户
ReleaseCOM	释放设备对象	上层及底层用户
② 模块信息取得/修改函数		
GetDevInfoCOM	设备校准	上层用户
SetDevInfoCOM	初始化 AD 部件准备传输	上层用户
③ 查询模块设备是否在线(DHCP)		
SearchDevice	创建设备	上层用户
④ 设备对象管理函数		
CreateENet	创建设备	上层用户
ReleaseENet	释放设备对象	上层用户
⑤ 模块信息取得/修改函数		
GetDevInfoCOM	读取设备信息(类型、地址、波特率、校验)	上层用户
SetDevInfoCOM	修改模块信息(地址、波特率、校验)	上层用户
⑥查询模块设备是否在线(DHCP)		
SearchDevice	初始化 RTU 设备 AD 部件，如通道等	上层用户
⑦ 设备对象管理函数		
CreateENet	创建设备	上层用户
ReleaseENet	释放设备对象	上层用户
⑧ 模块信息取得/修改函数		
GetENetworkConfig	获得设备的网络配置信息	上层用户
SetENetworkConfig	设置设备的网络配置信息	上层用户
⑨ 获得设备性能指标		
GetVerInfo	获取模块信息(设备名称、版本)	上层用户
GetDevCapability	取得 AD、DI、DO、DA、RTC 等设备数量	上层用户
⑩ AD 数据读取函数		
GetDeviceDA	读取 AD 模拟量输入	上层用户
GetModeAD	获得模拟量模式	上层用户
SetModeAD	设置 AD 模式	上层用户

⑾ DA 数据读取函数		
GetDeviceDA	回读 DA 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
⑿ DI 输入输出操作函数		
GetDeviceDI	读取开关量输入	上层用户
⒀ DO 数字量输出函数		
GetDeviceDO	回读开关量输出	上层用户
SetDeviceDO	设置 DO 开关量输出值	上层用户
⒁ 模块时间函数		
GetTime	取得模块时间	上层用户
SetTime	取得模块时间	上层用户
⒂ 模块温湿度函数		
GetEvrmTmpt	获得环境温度	上层用户
GetEvrmHum		上层用户
⒃ 读取最后一个错误函数		
GetLastErr	获得最后一个错误	上层用户
⒄ 注册表函数		
SaveIPAddress	保存 IP 到注册表	上层用户
LoadIPAddress	载入 IP 到应用程序	上层用户
⒅ 输入输出任意二进制字符		
WriteDeviceBYTE	直接写设备	上层用户
ReadDeviceBYTE	直接读设备	上层用户

使用需知：

Visual C++：

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\RTU6103\INCLUDE\RTU6103.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 RTU6103.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。

Visual Basic:

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模块文件(*.Bas)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 RTU6103.Bas 模块文件，该文件的路径为用户安装驱动程序后其子目录 Samples\VB 下面。

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，我们不能保证完全顺利运行。

第二节、设备对象管理函数原型说明

◆ **创建设备对象函数（逻辑号）**

函数原型：

Visual C++：

HANDLE CreateCOM (LONG lPortNum)

Visual Basic：

Declare Function CreateCOM Lib "RTU6103" (Optional ByVal DeviceID As lPortNum) As Long

功能：该函数创建设备对象

相关函数： CreateCOM ReleaseCOM InitCOM

♦ **初始与模块之间的通信参数**

函数原型：

Visual C++:

```
int InitCOM (HANDLE hDevice
             LONG    lBaud,
             BOOL    bCheck
             LONG    lTimeOut = RTU6XXX_DEFAULT_TIMEOUT
```

Visual Basic:

```
Declare Function GetDeviceCount Lib "RTU6103" (ByVal hDevice As Long
                                               ByVal lBaud As Long
                                               ByVal bCheck As Long
                                               ByVal lTimeOut = RTU6XXX_DEFAULT_TIMEOUT ) As Integer
```

功能：初始与模块之间的通信参数。

参数：

hDevice设备对象句柄，它应由CreateCOM创建

lBaud 波特率

bCheck 是否校验

lTimeOut = RTU6XXX_DEFAULT_TIMEOUT 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：返回系统中 RTU6103 的数量。

相关函数： CreateCOM ReleaseCOM InitCOM

♦ **释放设备对象**

函数原型：

Visual C++:

```
BOOL ReleaseCOM (HANDLE hDevice)
```

Visual Basic:

```
Declare Function ReleaseCOM Lib "RTU6103" (ByVal hDevice As Long ) As Boolean
```

LabVIEW：

请参考相关演示程序。

功能：释放设备对象。

参数：hDevice设备对象句柄，它应由CreateCOM创建。

返回值：若成功，则弹出对话框控件列表所有 RTU6103 设备的配置情况。

相关函数： CreateCOM ReleaseCOM InitCOM

第三节、模块信息取得/修改函数

♦ **读取设备信息(类型、地址、波特率、校验)**

函数原型：

Visual C++:

```
BOOL GetDevInfoCOM (HANDLE hDevice
                    PRTU6XXX_DEVICE_INFO pInfo
                    LONG   lDeviceID)
```

Visual Basic:

```
Declare Function GetDevInfoCOM Lib "RTU6103" (ByVal hDevice As Long
                                              ByVal RTU6XXX_DEVICE_INFO pInfo As Long
                                              ByVal lDeviceID As Long) As Boolean
```

LabVIEW：

功能：读取设备信息(类型、地址、波特率、校验)。

参数：hDevice设备对象句柄，它应由CreateDevice创建。

lDeviceID 设备地址

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

相关函数：GetDevInfoCOM SetDevInfoCOM

- **修改设备信息(类型、地址、波特率、校验)**

函数原型：

Visual C++:

```
BOOL SetDevInfoCOM (HANDLE hDevice
                    PRTU6XXX_DEVICE_INFO Info
                    LONG   lDeviceID)
```

Visual Basic:

```
Declare Function SetDevInfoCOM Lib "RTU6103" (ByVal hDevice As Long
                                              ByVal RTU6XXX_DEVICE_INFO Info As Long
                                              ByVal lDeviceID As Long) As Boolean
```

LabVIEW：

功能：读取设备信息(类型、地址、波特率、校验)。

参数：hDevice设备对象句柄，它应由CreateDevice创建。

lDeviceID 设备地址

RTU6XXX_DEVICE_INFO Info 设备信息

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

相关函数：GetDevInfoCOM SetDevInfoCOM

第四节、查询模块设备是否在线(DHCP)

- **创建设备**

函数原型：

Visual C++:

```
BOOL SearchDevice (Char szIP[],
                   LONG lPort,
                   PLONG plCount,
                   LONG   lSendTimeout,
                   LONG   lRcvTimeout)
```

Visual Basic:

```
Declare Function SearchDevice Lib "RTU6103" ( ByVal szIP[] As Long
                                              ByVal lPort As Long
                                              ByVal plCount As Long
                                              ByVal lSendTimeout As Long
                                              ByVal lRcvTimeout As Long) As Boolean
```

LabVIEW：

请参考相关演示程序。

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TRUE，否则返回FALSE，用户可以用GetLastErrorEx捕获错误码。
相关函数：SearchDevice

第五节、设备对象管理函数

◆ **创建设备**

函数原型：

Visual C++:

```
BOOL _CreateENet (Char szIP[],
                  LONG lPort,
                  LONG  lSendTimeout,
                  LONG  lRcvTimeout)
```

Visual Basic:

```
Declare Function _CreateENet Lib "RTU6103" (ByVal szIP[] As Long
                                            ByVal lPort As Long
                                             ByVal lSendTimeout As Long
                                             ByVal lRcvTimeout As Long) As Boolean
```

LabVIEW：

请参考相关演示程序。

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，用户可以用GetLastErrorEx捕获错误码。
相关函数：CreateENet　ReleaseENet

◆ **释放设备对象**

函数原型：

Visual C++:

```
BOOL ReleaseENet ( HANDLE hDevice )
```

Visual Basic:

```
Declare Function ReleaseENet Lib "RTU6103" (ByVal hDevice As Long ) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：释放设备对象。
参数：hDevice 设备对象句柄。
返回值：如果调用成功，则返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。
相关函数：CreateENet　ReleaseENet

第六节、模块信息取得/修改函数

◆ **获得设备的网络配置信息**

Visual C++:

BOOL GetENetworkConfig (HANDLE hDevice,
PDEVICE_NET_INFO pNetInfo,)

Visual Basic:

Declare Function GetENetworkConfig Lib "RTU6103" (
ByVal hDevice As Long, _
ByRef PDEVICE_NET_INFO pNetInfo As Integer,_) As Boolean

LabVIEW:

请参考相关演示程序。

功能：获得设备的网络配置信息。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

PDEVICE_NET_INFO pNetInfo 网络配置信息。

返回值：如果调用成功，则返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数: GetENetworkConfig SetENetworkConfig

♦ **设置设备的网络配置信息**

Visual C++:

BOOL SetENetworkConfig (HANDLE hDevice,
PDEVICE_NET_INFO NetInfo,)

Visual Basic:

Declare Function SetENetworkConfig Lib "RTU6103" (
ByVal hDevice As Long, _
ByRef PDEVICE_NET_INFO NetInfo As Integer,_) As Boolean

LabVIEW:

请参考相关演示程序。

功能：获得设备的网络配置信息。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

PDEVICE_NET_INFO NetInfo 网络配置信息。

返回值：如果调用成功，则返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数: GetENetworkConfig SetENetworkConfig

第七节、获得设备性能指标

♦ **获取模块信息(设备名称、版本)**

函数原型：

Visual C++:

BOOL GetVerInfo (HANDLE hDevice,
PRTU6XXX_MODULEINFO pModuleInfo,
LONG lDeviceID = 0)

Visual Basic:

Declare Function GetVerInfo Lib "RTU6103" (ByVal hDevice As Long,_
ByVal pModuleInfo As Long,_
ByVal lDeviceID = 0 As Long,_)As Boolean

LabVIEW:

请参考相关演示程序。

功能：获取模块信息(设备名称、版本)

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

PRTU6XXX_MODULEINFO pModuleInfo， 模块信息
lDeviceID = 0，设备地址
返回值：若调用成功则返回TRUE，否则返回FALSE，用户可以调用GetLastErrorEx函数取得当前错误码。
相关函数： GetDevCapability GetVerInfo

◆ **获取模块信息(设备名称、版本)**

函数原型：
Visual C++:

```
BOOL GetDevCapability ( HANDLE hDevice,
                        PRTU6XXX_CAPABILITY pCapability,
                        LONG lDeviceID = 0)
```

Visual Basic:

```
Declare Function GetDevCapability Lib "RTU6103" (ByVal hDevice As Long,_
                                                 ByVal pCapability As Long,_
                                                 ByVal lDeviceID = 0 As Long,_)As Boolean
```

LabVIEW:

请参考相关演示程序。
功能：获取模块信息(设备名称、版本)
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
PRTU6XXX_CAPABILITY pCapability，获得设备性能
lDeviceID = 0，设备地址
返回值：若调用成功则返回TRUE，否则返回FALSE，用户可以调用GetLastErrorEx函数取得当前错误码。
相关函数： GetDevCapability GetVerInfo

第八节、AD 数据读取函数

◆ **读取 AD 模拟量输入**

函数原型：
Visual C++:

```
BOOL ReadDeviceAD ( HANDLE hDevice,
                    float   lpADBuffer[],
                    LONG lFirstChannel = 0,
                    LONG lLastChannel = 0,
                    LONG lDeviceID = 0)
```

Visual Basic:

```
Declare Function ReadDeviceAD Lib "RTU6103" (ByVal hDevice As Long, _
                                             ByRef lpADBuffer[] As Integer,_
                                             ByVal lFirstChannel = 0 As Long ,_
                                             ByRef lLastChannel = 0 As Long
                                             ByRef lDeviceID = 0 As Long) As Boolean
```

LabVIEW:
请参考相关演示程序。

功能：读取 AD 模拟量输入。
参数：
hDevice 设备对象句柄
lpADBuffer[] 接收 AD 数据的用户缓冲区 注意:lpADBuffer 最好大于等于 lLastChannel - lFirstChannel +1
lFirstChannel = 0 首通道
lLastChannel = 0 末通道
lDeviceID = 0 设备地址。
返回值：如果成功的 ，则返回TRUE，否则返回FALSE， 用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数: ReadDeviceAD GetModeAD SetModeAD

- **获得模拟量模式**

函数原型:

Visual C++:

```
BOOL GetModeAD ( HANDLE hDevice,
                 LONG lADRange[],
                 LONG lADGrounding[]
                 LONG lFirstChannel = 0
                 LONG lLastChannel = 0,
                 LONG lDeviceID = 0)
```

Visual Basic:

```
Declare Function GetModeAD Lib "RTU6103" (ByVal hDevice As Long, _
                                          ByRef lADRange[]As Integer,_
                                          ByVal lADGrounding[] As Long ,_
                                          ByRef lFirstChannel = 0 As Long
                                          ByRef lLastChannel = 0 As Long
                                          ByRef lDeviceID = 0 As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能:读取 AD 模拟量输入。

参数:

hDevice 设备对象句柄

lpADBuffer[] 接收 AD 数据的用户缓冲区 注意:lpADBuffer 最好大于等于 lLastChannel - lFirstChannel +1

lFirstChannel = 0 首通道

lADRange[] AD 量程

lADGrounding[] AD 接地方式

lLastChannel = 0 末通道

lDeviceID = 0 设备地址。

返回值:如果成功的 ,则返回TRUE,否则返回FALSE, 用户可用GetLastErrorEx捕获当前错误码,并加以分析。

相关函数: ReadDeviceAD GetModeAD SetModeAD

- **设置 AD 模式**

函数原型:

Visual C++:

```
BOOL SetModeAD ( HANDLE hDevice,
                 const LONG lADRange[],
                 const LONG lADGrounding[]
                 LONG lFirstChannel = 0,
                 LONG lLastChannel = 0,
                 LONG lDeviceID = 0)
```

Visual Basic:

```
Declare Function SetModeAD Lib "RTU6103" (ByVal hDevice As Long, _
                                          ByRef lADRange[] As Integer,_
                                          ByRef lADGrounding[] As Integer,_
                                          ByVal lFirstChannel = 0 As Long ,_
                                          ByRef lLastChannel = 0 As Long
                                          ByRef lDeviceID = 0 As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能:设置 AD 模式。

参数：
hDevice 设备对象句柄
lADGrounding[] AD 接地方式
lFirstChannel = 0 首通道
lADRange[] AD 量程
lLastChannel = 0 末通道
lDeviceID = 0 设备地址。
返回值：如果成功的 ，则返回TRUE，否则返回FALSE， 用户可用GetLastErrorEx捕获当前错误码，并加以分析。
相关函数： ReadDeviceAD GetModeAD SetModeAD

第九节、DA 数据读取函数

♦ 回读 DA 输出值

函数原型：
Visual C++:

```
BOOL GetDeviceDA (HANDLE hDevice
                  PFLOAT fpDAValue,
                  LONG lChannel,
                  LONG lDeviceID = 0)
```

Visual Basic:

```
Declare Function GetDeviceDA ( Lib "RTU6103" (ByVal hDevice As Long
                                              ByVal fpDAValue As Long
                                              ByVal lChannel As Long
                                              ByVal lDeviceID = 0 As Long)As Boolean
```

LabVIEW:
请参考相关演示程序。

功能：回读 DA 输出值。
参数：
hDevice 设备对象句柄
fpDAValue DA 当前值
lChannel 首通道号
lDeviceID = 0 设备地址。
返回值：如果调用成功，则返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。
相关函数： GetDeviceDA WriteDeviceDA GetRangeDA SetRangeDA

♦ 设定单通道 DA

函数原型：
Visual C++:

```
BOOL WriteDeviceDA (HANDLE hDevice
                    PFLOAT fDAData,
                    LONG lChannel,
                    LONG lDeviceID = 0)
```

Visual Basic:

```
Declare Function WriteDeviceDA Lib "RTU6103" (ByVal hDevice As Long
                                              ByVal fDAData As Long
                                              ByVal lChannel As Long
                                              ByVal lDeviceID = 0 As Long)As Boolean
```

LabVIEW:
请参考相关演示程序。

功能：设定单通道 DA。
参数：
hDevice 设备对象句柄
fDAData DA 输出值
lChannel 首通道号
lDeviceID = 0 设备地址。
返回值：如果调用成功，则返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。
相关函数：GetDeviceDA WriteDeviceDA GetRangeDA SetRangeDA

◆ **读取模拟量输出量程**

函数原型：

Visual C++:

```
BOOL GetRangeDA (HANDLE hDevice
                 LONG lDARange[],
                 LONG lFirstChannel = 0,
                 LONG lLastChannel = 0
                 LONG lDeviceID = 0)
```

Visual Basic:

```
Declare Function GetRangeDA Lib "RTU6103" (ByVal hDevice As Long
                                           ByVal lDARange[] As Long
                                           ByVal l lFirstChannel = 0As Long
                                           ByVal l lLastChannel = 0As Long
                                           ByVal lDeviceID = 0 As Long)As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：读取模拟量输出量程。
参数：
hDevice 设备对象句柄
lDARange[] 输出模式
lChannel 首通道号
lLastChannel = 0 末通道号
lDeviceID = 0 设备地址。
返回值：如果调用成功，则返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。
相关函数：GetDeviceDA WriteDeviceDA GetRangeDA SetRangeDA

◆ **设置模拟量输出量程**

函数原型：

Visual C++:

```
BOOL SetRangeDA (HANDLE hDevice
                 LONG lDARange[],
                 LONG lFirstChannel = 0,
                 LONG lLastChannel = 0
                 LONG lDeviceID = 0)
```

Visual Basic:

```
Declare Function SetRangeDA Lib "RTU6103" (ByVal hDevice As Long
                                           ByVal lDARange[] As Long
                                           ByVal l lFirstChannel = 0As Long
                                           ByVal l lLastChannel = 0As Long
                                           ByVal lDeviceID = 0 As Long)As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：读取模拟量输出量程。
参数：
hDevice 设备对象句柄
lDARange[] 输出模式
lChannel 首通道号
lLastChannel = 0 末通道号
lDeviceID = 0 设备地址。
返回值：如果调用成功，则返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。
相关函数：GetDeviceDA WriteDeviceDA GetRangeDA SetRangeDA

第十节、DI 输入输出操作函数

◆ **读取开关量输入**

函数原型：
Visual C++:

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

Visual Basic:

```
Declare Function GetDeviceDI Lib "RTU6103" (ByVal hDevice As Long
                                            ByVal byDIStatus[] As Long
                                            ByVal l lFirstChannel = 0As Long
                                            ByVal l lLastChannel = 0As Long
                                            ByVal lDeviceID = 0 As Long)As Boolean
```

LabVIEW:
请参考相关演示程序。

功能：读取开关量输入。
参数：
byDIStatus[] DI 值
fpDAValue DA 当前值
lFirstChannel = 0 首通道号
lLastChannel = 0 末通道号
lDeviceID = 0 设备地址。
返回值：如果调用成功，则返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。
相关函数：GetDeviceDI

第十一节、DO 数字量输出函数

◆ **回读开关量输出**

函数原型：
Visual C++:

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

Visual Basic:

```
Declare Function GetDeviceDO Lib "RTU6103" (ByVal hDevice As Long
                                            ByVal byDOSts[] As Long
                                            ByVal l lFirstChannel = 0As Long
                                            ByVal l lLastChannel = 0As Long
                                            ByVal lDeviceID = 0 As Long)As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：回读开关量输出。

参数：

byDOSts[]　DO 值

fpDAValue DA 当前值

lFirstChannel = 0　首通道号

lLastChannel = 0 末通道号

lDeviceID = 0　设备地址。

返回值：如果调用成功，则返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数：GetDeviceDO　SetDeviceDO

♦ **设置 DO 开关量输出值**

函数原型：

Visual C++:

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

Visual Basic:

```
Declare Function GetDeviceDI Lib "RTU6103" (ByVal hDevice As Long
                                            ByVal byDOSts[] As Long
                                            ByVal l lFirstChannel = 0As Long
                                            ByVal l lLastChannel = 0As Long
                                            ByVal lDeviceID = 0 As Long)As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：设置 DO 开关量输出值。

参数：

byDOSts[]　DO 输出值

fpDAValue DA 当前值

lFirstChannel = 0　首通道号

lLastChannel = 0 末通道号

lDeviceID = 0　设备地址。

返回值：如果调用成功，则返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数：GetDeviceDO　SetDeviceDO

第十二节、模块时间函数

♦**取得模块时间**

函数原型：

Visual C++:

```
BOOL GetTime ( HANDLE hDevice,
                    PRTU6XXX_TIME pTime,
```

LONG lDeviceID = 0)

Visual Basic:

Declare Function GetTime Lib "RTU6103" (ByVal hDevice As Long, _
ByVal pTime As Long, _
ByRef lDeviceID = 0 As Integer,_) As Boolean

LabVIEW:

请参考相关演示程序。

功能：取得模块时间。

参数：

hDevice 设备对象句柄。

pTime 时间

lDeviceID = 0 设备地址

返回值：如果初始化设备对象成功，则返回TRUE， 否则返回FALSE， 用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数： GetTime SetTime

◆ **取得模块时间**

函数原型：

Visual C++:

BOOL SetTime (HANDLE hDevice,
PRTU6XXX_TIME pTime,
LONG lDeviceID = 0)

Visual Basic:

Declare Function SetTime Lib "RTU6103" (ByVal hDevice As Long, _
ByVal pTime As Long, _
ByRef lDeviceID = 0 As Integer,_) As Boolean

LabVIEW:

请参考相关演示程序。

功能：取得模块时间。

参数：

hDevice 设备对象句柄。

pTime 时间

lDeviceID = 0 设备地址

返回值：如果初始化设备对象成功，则返回TRUE， 否则返回FALSE， 用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数： GetTime SetTime

第十三节、模块温湿度函数

◆ **获得环境温度**

函数原型：

Visual C++:

BOOL GetEvrmTmpt (HANDLE hDevice,
PLONG lpEvrmTmpt,
LONG lDeviceID = 0)

Visual Basic:

Declare Function GetEvrmTmpt Lib "RTU6103" (ByVal hDevice As Long, _
ByVal lpEvrmTmp As Long, _
ByRef lDeviceID = 0 As Integer,_) As Boolean

LabVIEW:

请参考相关演示程序。

功能：获得环境温度。

参数：

hDevice 设备对象句柄。

lpEvrmTmp 环境温度

lDeviceID = 0 设备地址

返回值：如果初始化设备对象成功，则返回TRUE， 否则返回FALSE， 用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数： GetEvrmTmpt GetEvrmHum

♦ **获得环境湿度**

函数原型：

Visual C++:

```
BOOL GetEvrmHum (HANDLE hDevice,
                 PLONG lpEvrmHum,
                 LONG lDeviceID = 0)
```

Visual Basic:

```
Declare Function GetEvrmHum Lib "RTU6103" (ByVal hDevice As Long, _
                                           ByVal lpEvrmHum As Long, _
                                           ByRef lDeviceID = 0 As Integer,_ ) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：获得环境湿度。

参数：

hDevice 设备对象句柄。

lpEvrmHum 环境湿度

lDeviceID = 0 设备地址

返回值：如果初始化设备对象成功，则返回TRUE， 否则返回FALSE， 用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数： GetEvrmTmpt GetEvrmHum

第十四节、读取最后一个错误函数

♦ **获得最后一个错误**

函数原型：

Visual C++:

```
BOOL GetLastErr (HANDLE hDevice,
                 LONG lDeviceID = 0)
```

Visual Basic:

```
Declare Function GetLastErr Lib "RTU6103" (ByVal hDevice As Long, __
                                           ByRef lDeviceID = 0 As Integer,_ ) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：获得最后一个错误。

参数：

hDevice 设备对象句柄。

lDeviceID = 0 模块地址

返回值：如果初始化设备对象成功，则返回TRUE， 否则返回FALSE， 用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数： GetLastErr

第十五节、注册表函数

♦ 保存 IP 到注册表

函数原型：

Visual C++:

```
BOOL SaveIPAddress (char szIP[])
```

Visual Basic:

```
Declare Function SaveIPAddress Lib "RTU6103" (ByVal szIP[]As Long, ) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：保存 IP 到注册表。

参数：

char szIP[] 保存 IP 到注册表。

返回值：如果初始化设备对象成功，则返回TRUE， 否则返回FALSE， 用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数： SaveIPAddress LoadIPAddress

♦ 载入 IP 到应用程序

函数原型：

Visual C++:

```
BOOL LoadIPAddress (char szIP[])
```

Visual Basic:

```
Declare Function LoadIPAddress Lib "RTU6103" (ByVal szIP[]As Long, ) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：载入 IP 到应用程序。

参数：

char szIP[] 载入 IP 到应用程序。

返回值：如果初始化设备对象成功，则返回TRUE， 否则返回FALSE， 用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数： SaveIPAddress LoadIPAddress

第十六节、输入输出任意二进制字符

♦ 直接写设备

函数原型：

Visual C++:

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

Visual Basic:

```
Declare Function WriteDeviceBYTE Lib "RTU6103" (ByVal hDevice As Long,
                                                ByVal byWriteBuf As Long,
                                                ByVal llength As Integer,_
                                                ByVal timeout=100 As Integer) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：直接写设备。

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

byWriteBuf d 写的数据

llength 数据长度
timeout=100 超时范围(mS)
返回值：若成功，返回TRUE，否则返回FALSE，您可以调用GetLastErrorEx函数取得错误或错误字符信息。
相关函数： WriteDeviceBYTE ReadDeviceBYTE

♦ **直接读设备**

函数原型：
Visual C++:

```
BOOL ReadDeviceBYTE ( HANDLE hDevice,
                      BYTE*  byWriteBuf,
                      Long llength,
                      long timeout=100)
```

Visual Basic:

```
Declare Function ReadDeviceBYTE Lib "RTU6103" (ByVal hDevice As Long,
                                               ByVal byWriteBuf As Long,
                                               ByVal llength As Integer,_
                                               ByVal timeout=100 As Integer) As Boolean
```

LabVIEW:
请参考相关演示程序。
功能：直接读设备。
参数：
hDevice 设备对象句柄。
byWriteBuf d 写的数据
llength 数据长度
timeout=100 超时范围(mS)
返回值：若成功，返回TRUE，否则返回FALSE，您可以调用GetLastErrorEx函数取得错误或错误字符信息。
相关函数： WriteDeviceBYTE ReadDeviceBYTE

第四章 硬件参数结构

第一节、开关量输入的参数介绍（RTU6103_PARA_AD）

Visual C++:

```
typedef struct    _RTU6XXX_PARA_DI        //  数字量输入参数(1 为高电平)
{
    BYTE DI0;          // 0 通道
    BYTE DI1;          // 1 通道
    BYTE DI2;          // 2 通道
    BYTE DI3;          // 3 通道
    BYTE DI4;          // 4 通道
    BYTE DI5;          // 5 通道
    BYTE DI6;          // 6 通道
    BYTE DI7;          // 7 通道
    BYTE DI8;          // 8 通道
    BYTE DI9;          // 9 通道
    BYTE DI10;         // 10 通道
    BYTE DI11;         // 11 通道
    BYTE DI12;         // 12 通道
    BYTE DI13;         // 13 通道
    BYTE DI14;         // 14 通道
    BYTE DI15;         // 15 通道
```

```
} RTU6XXX_PARA_DI, *PRTU6XXX_PARA_DI;
```

Visual Basic：

```
Private Type RTU6103_PARA_AD
    DI0 As Long             ' 0 通道
    DI1 As Long             ' 1 通道
    DI2 As Long             ' 2 通道
    DI3 As Long             '3 通道
    DI4 As Long             '4 通道
    DI5 As Long             ' 5 通道
    DI6 As Long             ' 6 通道
    DI7 As Long             ' 7 通道
    DI8 As Long             ' 8 通道
    DI9 As Long             ' 9 通道
    DI10 As Long            ' 10 通道
    DI11 As Long            ' 11 通道
    DI12 As Long            ' 12 通道
    DI13As Long             '13 通道
    DI14 As Long            '14 通道
    DI15 As Long            '15 通道
End Type
```

LabVIEW:

请参考相关演示程序。

第二节、开关量输出的参数介绍（RTU6103_PARA_AD）

Visual C++:

```
typedef struct    _RTU6XXX_PARA_DI          //数字量输出参数
{
    BYTE DO0;          // 0 通道
    BYTE DO1;          // 1 通道
    BYTE DO2;          // 2 通道
    BYTE DO3;          // 3 通道
    BYTE DO4;          // 4 通道
    BYTE DO5;          // 5 通道
    BYTE DO6;          // 6 通道
    BYTE DO7;          // 7 通道
    BYTE DO8;          // 8 通道
    BYTE DO9;          // 9 通道
    BYTE DO10;         // 10 通道
    BYTE DO11;         // 11 通道
    BYTE DO12;         // 12 通道
    BYTE DO13;         // 13 通道
    BYTE DO14;         // 14 通道
    BYTE DO15;         // 15 通道
} RTU6XXX_PARA_DI, *PRTU6XXX_PARA_DI;
```

Visual Basic：

```
Private Type RTU6103_PARA_AD
    DO0 As Long             ' 0 通道
    DO1 As Long              ' 1 通道
```

```
    DO2 As Long          ' 2 通道
    DO3 As Long          '3 通道
    DO4 As Long          '4 通道
    DO5 As Long          ' 5 通道
    DO6 As Long              ' 6 通道
    DO7 As Long          ' 7 通道
    DO8 As Long          ' 8 通道
    DO9 As Long          ' 9 通道
    DO10 As Long             ' 10 通道
    DO11 As Long         ' 11 通道
    DO12 As Long         ' 12 通道
    DO13As Long          '13 通道
    DO14 As Long         '14 通道
    DO15 As Long         '15 通道
End Type
```

LabVIEW:

请参考相关演示程序。

第三节、模拟量输入通道配置结构体（RTU6103_PARA_AD）

Visual C++:

```
typedef struct _RTU6XXX_ADCHANNEL_ARRAY
{
    BYTE bChannel0; // 1，有效；0，无效
    BYTE bChannel1;
    BYTE bChannel2;
    BYTE bChannel3;
    BYTE bChannel4;
    BYTE bChannel5;
    BYTE bChannel6;
    BYTE bChannel7;
}RTU6XXX_ADCHANNEL_ARRAY, *PRTU6XXX_ADCHANNEL_ARRAY;Visual Basic：
```

Visual Basic：

```
Private Type RTU6103_PARA_AD
    bChannel0 As Long        ' 1，有效；0，无效
    bChannel1 As Long
    bChannel2 As Long
    bChannel3 As Long
    bChannel4 As Long
    bChannel5 As Long
    bChannel6 As Long
    bChannel7 As Long
End Type
```

LabVIEW:

请参考相关演示程序。

第四节、时间结构体（RTU6103_PARA_AD）

Visual C++:

```
typedef struct _RTU6XXX_TIME
{
    LONG lYear;
    LONG lMonth;
    LONG lDay;
    LONG lWeek;
    LONG lHour;
    LONG lMinute;
    LONG lSecond;
}RTU6XXX_TIME, *PRTU6XXX_TIME;
```

Visual Basic：

```
Private Type RTU6103_PARA_AD
    lYear As Long
    lMonth As Long
    lDay As Long
    lWeek As Long
    lHour As Long
    lMinute As Long
    lSecond As Long
End Type
```

LabVIEW:

请参考相关演示程序。

第五节、设备性能指标(通道配置)结构体 RTU6103_PARA_AD）

Visual C++:

```
typedef struct _RTU6XXX_CAPABILITY
{
    LONG lAI;
    LONG lAO;
    LONG lDI;
    LONG lDO;
    LONG lCNT;
    LONG lRTC;
}RTU6XXX_CAPABILITY, *PRTU6XXX_CAPABILITY;
```

Visual Basic：

```
Private Type RTU6103_PARA_AD
    lAI As Long
    lAO As Long
    lDI As Long
    lDO As Long
    lCNT As Long
    lRTC As Long
End Type
```

LabVIEW:

请参考相关演示程序。

第六节、模块信息（RTU6XXX_GetVerInfo 使用）

Visual C++:

```
typedef struct   _RTU6XXX_MODULEINFO
{
    char    strModuleName[2];   // 模块名称
    char    strVerInfo[6];      // 版本信息
} RTU6XXX_MODULEINFO, *PRTU6XXX_MODULEINFO;
```

Visual Basic：

```
Private Type RTU6103_PARA_AD
    strModuleName[2] As Long      ' 模块名称
    strVerInfo[6] As Long         ' 版本信息
End Type
```

LabVIEW:

请参考相关演示程序。

第七节、设备基本信息的结构体

Visual C++:

```
typedef struct _RTU6XXX_DEVICE_INFO
    {
        LONG    DeviceID;           // 模块 ID 号(SetDeviceInfo 时，为设备的新 ID)
        LONG    BaudRate;           // 波特率
        LONG    bParity;            // 有无校验
        } RTU6XXX_DEVICE_INFO, *PRTU6XXX_DEVICE_INFO;
```

Visual Basic：

```
Private Type RTU6103_PARA_AD
    DeviceID   As Long      ' 模块 ID 号(SetDeviceInfo 时，为设备的新 ID)
    BaudRate   As Long      ' 波特率
    bParity    As Long      ' 有无校验
End Type
```

LabVIEW:

请参考相关演示程序。

第八节、设备网络信息

Visual C++:

```
typedef struct _DEVICE_NET_INFO
 {
    char      szIP[16];                 // IP 地址, // 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
    LONG     lHTTPPort;                 // HTTP 端口号
```

```
}DEVICE_NET_INFO, *PDEVICE_NET_INFO;
```

Visual Basic：

```
Private Type RTU6103_PARA_AD
    szIP[16]      As Long              '   IP 地址, // IP 地址, "192.168.2.70"
    SubnetMask[16]   As Long           ' 子网掩码, "255.255.255.255"
    Gateway[16]   As Long              ' 网关, "192.168.2.1"
    MAC[20]   As Long       ' 网卡物理地址,   "00-01-02-03-04-05",用户一般不可修改
    lTCPPort   As Long        'TCP 端口号
    lHTTPPort   As Long     'HTTP 端口号
End Type
```

LabVIEW:

请参考相关演示程序。

以太网连接模式：

常量名	常量值	功能定义
RTU6XXX_LINK_UDP	0x00	UDP 模式
RTU6XXX_LINK_TCP	0x01	TCP 模式

模拟量输入量程(电压类型) 供 RTU6XXX_SetModeAD 函数中的 lADRange 参数使用

常量名	常量值	功能定义
RTU6XXX_VOLT_N5_P5	0x00	-5～+5V
RTU6XXX_VOLT_N0_P5	0x01	0～+5V
RTU6XXX_VOLT_N10_P10	0x02	-10～+10V
RTU6XXX_VOLT_N0_P10	0x03	0～+10V

电流量程。

常量名	常量值	功能定义
RTU6XXX_CURRENT_N4_P20	0x0A	4～+20mA

模拟量输入模式 供 RTU6XXX_SetModeAD 函数中的 lADGrounding 参数使用

常量名	常量值	功能定义
RTU6XXX_GNDMODE_SE	0x00	单端方式(SE:Single end)
RTU6XXX_GNDMODE_DI	0x01	双端方式(DI:Differential)

串口号(以此类推) 供 RTU6XXX_CreateDevice 使用，可根据自身需要扩充。

常量名	常量值	功能定义
RTU6XXX_COM1	0x01	COM1
RTU6XXX_COM2	0x02	COM2
RTU6XXX_COM3	0x03	COM3
RTU6XXX_COM4	0x04	COM4
RTU6XXX_COM5	0x05	COM5

波特率选择 供 RTU6XXX_SetDeviceInfo 和 RTU6XXX_GetDeviceInfo 中的 PRTU6XXX_DEVICE_INFO 使用

常量名	常量值
RTU6XXX_BAUD_1200	0x00
RTU6XXX_BAUD_2400	0x01

RTU6XXX_BAUD_4800	0x02
RTU6XXX_BAUD_9600	0x03
RTU6XXX_BAUD_19200	0x04
RTU6XXX_BAUD_38400	0x05
RTU6XXX_BAUD_56700	0x06
RTU6XXX_BAUD_115200	0x07
RTU6XXX_DEFAULT_TIMEOUT	-1

错误代号 由 RTU6XXX_GetLastErr 函数取得最后一次错误代码，它的选项值如下表：

常量名	常量值	功能定义
RTU6XXX_ERROR_LINK	-1	设备连接异常
RTU6XXX_ERROR_DATA	-2	数据接收异常

第五章 数据格式转换与排列规则

第一节、AD 原码 LSB 数据转换成电压值的换算方法

首先应根据设备实际位数屏蔽掉不用的高位，然后依其所选量程，按照下表公式进行换算即可。这里只以缓冲区 ADBuffer[]中的第 1 个点 ADBuffer[0]为例。

量程(mV)	计算机语言换算公式(ANSI C 语法)	Volt 取值范围（mV）
±10000mV	Volt = (20000.00/65536) * (ADBuffer[0]&0xFFF) – 10000.00	[-10000, +9997.55]
±5000mV	Volt = (10000.00/65536) * (ADBuffer[0]&0xFFF) – 5000.00	[-5000, +4998.77]
0～10000mV	Volt = (10000.00/65536) * (ADBuffer[0]&0xFFF)	[0, +9998.77]
0～5000mV	Volt = (5000.00/65536) * (ADBuffer[0]&0xFFF)	[0, +4997.55]

下面举例说明各种语言的换算过程（以±10000mV 量程，16 位精度为例）

Visual C++：

Lsb = ADBuffer[0]&0xFFF;

Volt = (20000.00/65536) * Lsb -10000.00;

Visual Basic：

Lsb = ADBuffer [0] And &HFFF

Volt = (20000.00/65536) * Lsb – 10000.00

LabVIEW:

请参考相关演示程序。

第二节、AD 采集函数的 ADBuffer 缓冲区中的数据排放规则

单通道采集，当通道总数首末通道相等时，假如此时首末通道=5。其排放规则如下：

数据缓冲区索引号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	…
通道号	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	…

两通道采集（假如 FirstChannel=0，LastChannel=1）：

数据缓冲区索引号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	…
通道号	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	…

四通道采集（假如 FirstChannel=0，LastChannel=1）：

数据缓冲区索引号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	…
通道号	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	…

其他通道方式以此类推。

如果用户是进行连续不间断循环采集，即用户只进行一次初始化设备操作，然后不停的从设备上读取 AD 数据，那么需要用户特别注意的是应处理好各通道数据排列和对齐的问题，尤其是在任意通道数采集时。否则，用户无将规则排放在缓冲区中的各通道数据正确分离出来。那怎样正确处理呢？我们建议的方法是，每次从设备上读取的点数应置为所选通道数量的整数倍长，这样便能保证每读取的这批数据在缓冲区中的相应位置始终固定对应于某一个通道的数据。比如用户要求对 1、2 两个 AD 通道的数据进行连续循环采集，则置每次读取长度为其 2 的整倍长 2n(n 为每个通道的点数)，这里设为 2048。试想，如此一来，每次读取的 2048 个点中的第一个点始终对应于 1 通道数据，第二个点始终对应于 2 通道，第三个点再应于 1 通道，第四个点再对应于 2 通道……以此类推。直到第 2047 个点对应于 1 通道数据，第 2048 个点对应 2 通道。这样一来，每次读取的段长正好包含了从首通道到末通道的完整轮回，如此一来，用户只须按通道排列规则，按正常的处理方法循环处理每一批数据。而对于其他情况也是如此，比如 3 个通道采集，则可以使用 3n(n 为每个通道的点数)的长度采集。为了更加详细地说明问题，请参考下表（演示的是采集 1、2、3 共三个通道的情况）。由于使用连续采样方式，所以表中的数据序列一行的数字变化说明了数据采样的连续性，即随着时间的延续，数据的点数连续递增，直至用户停止设备为止，从而形成了一个有相当长度的连续不间断的多通道数据链。而通道序列一行则说明了随着连续采样的延续，其各通道数据在其整个数据链中的排放次序，这是一种非常规则而又绝对严格的顺序。但是这个相当长度的多通道数据链则不可能一次通过设备对象函数如 ReadDeviceProAD_X 函数读回，即便不考虑是否能一次读完的问题，仅对于用户的实时数据处理要求来说，一次性读取那么长的数据，则往往是相当矛盾的。因此我们就得分若干次分段读取。但怎样保证既方便处理，又不易出错，而且还高效呢？ 还是正如前面所说，采用通道数的整数倍长读取每一段数据。如表中列举的方法 1（为了说明问题，我们每读取一段数据只读取 2n 即 3*2＝6 个数据）。从方法 1 不难看出，每一段缓冲区中的数据在相同缓冲区索引位置都对应于同一个通道。而在方法 2 中由于每次读取的不是通道整数倍长，则出现问题，从表中可以看出，第一段缓冲区中的 0 索引位置上的数据对应的是第 1 通道，而第二段缓冲区中的 0 索引位置上的数据则对应于第 2 通道的数据，而第三段缓冲区中的数据则对应于第 3 通道……，这显然不利于循环有效处理数据。

在实际应用中，我们在遵循以上原则时，应尽可能地使每一段缓冲足够大，这样，可以一定程度上减少数据采集程序和数据处理程序的 CPU 开销量。

数据序列	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	. . .
通道序列	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	. . .
方法 1	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	
缓冲区号	第一段缓冲						第二段缓冲区						第三段缓冲区						第 n 段缓冲			
方法 2	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	
	第一段缓冲区				第二段缓冲区				第三段缓冲区				第四段缓冲区				第五段缓冲区				第 n 段缓冲	

第三节、AD 测试应用程序创建并形成的数据文件格式

首先该数据文件从始端 0 字节位置开始往后至第 HeadSizeBytes 字节位置宽度属于文件头信息，而从 HeadSizeBytes 开始才是真正的 AD 数据。HeadSizeBytes 的取值通常等于本头信息的字节数大小。文件头信息包含的内容如下结构体所示。对于更详细的内容请参考 Visual C++高级演示工程中的 UserDef.h 文件。

```
typedef struct _FILE_HEADER
{
    LONG HeadSizeBytes;                 // 文件头信息长度
    LONG FileType;                      // 该设备数据文件共有的成员
    LONG BusType;                       // 设备总线类型(DEFAULT_BUS_TYPE)
    LONG DeviceNum;                     // 该设备的编号(DEFAULT_DEVICE_NUM)
    LONG HeadVersion;                   // 头信息版本(D31-D16=Major,D15-D0=Minijor) = 1.0
    LONG VoltBottomRange;               // 量程下限(mV)
    LONG VoltTopRange;                  // 量程上限(mV)
    LONG StaticOverFlow;                // 同批文件识别码
    RTU6103_PARA_AD ADPara;             // 保存硬件参数
    LONG HeadEndFlag;                   // 头信息结束位
} FILE_HEADER，*PFILE_HEADER;
```

AD 数据的格式为 16 位二进制格式，它的排放规则与在 ADBuffer 缓冲区排放的规则一样，即每 16 位二进制(字)数据对应一个 16 位 AD 数据。您只需要先开辟一个 16 位整型数组或缓冲区，然后将磁盘数据从指定位置(即双字节对齐的某个位置)读入数组或缓冲区，然后访问数组中的每个元素，即是对相应 AD 数据的访问。

第四节、DA 电压值转换成 LSB 原码数据的换算方法

量程（伏）	计算机语言换算公式	Lsb 取值范围
0～5000mV	Lsb=Volt/(5000.00/4096)	[0，4095]
0～10000mV	Lsb=Volt/(10000.00/4096)	[0，4095]
±5000mV	Lsb=Volt/(10000.00/4096)+2048	[0，4095]
±10000mV	Lsb=Volt/(20000.00/4096)+2048	[0，4095]

第六章 上层用户函数接口应用实例

第一节、简易程序演示说明

一、怎样使用GetDeviceDA函数读取AD数据

Visual C++:

其详细应用实例及正确代码请参考 Visual C++测试与演示系统，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程。

[程序]丨[阿尔泰测控演示系统]丨[RTU6103 4 路 AD、4 路 DA 和 32 路 DIO 卡]丨[Microsoft Visual C++]丨[简易代码演示]丨[DA 方式]

二、怎样使用GetDeviceDI函数进行更便捷式数字量输入操作

Visual C++:

其详细应用实例及正确代码请参考 Visual C++测试与演示系统，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程。

[程序]丨[阿尔泰测控演示系统]丨[RTU6103 4 路 AD、4 路 DA 和 32 路 DIO 卡]丨[Microsoft Visual C++]丨[简易代码演示]丨[DIO...]

三、怎样使用SetDeviceDO函数进行更便捷式数字量输出操作

Visual C++:

其详细应用实例及正确代码请参考 Visual C++测试与演示系统，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程。

[程序]丨[阿尔泰测控演示系统]丨[RTU6103 4 路 AD、4 路 DA 和 32 路 DIO 卡]丨[Microsoft Visual C++]丨[简易代码演示]丨[DIO...]

第二节、高级程序演示说明

高级程序演示了本设备的所有功能，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程(主要参考 RTU6103.h 和 ADDoc.cpp)。

[程序]丨[阿尔泰测控演示系统]丨[RTU6103 4 路 AD、4 路 DA 和 32 路 DIO 卡]丨[Microsoft Visual C++]丨[高级代码演示]

其默认存放路径为：系统盘\ART\RTU6103\SAMPLES\VC\ADVANCED

其他语言的演示可以用上面类似的方法找到。