PXI8191 开发指南

目　　录

1 规范与约定

1.1 关键字缩写命名约定

缩写	全称	定义	缩写	全称	定义
DEV/Dev	Device	设备	DIR/Dir	Direction	方向
AI	Analog Input	模拟量输入	CPLG	Coupling	耦合
AO	Analog Output	模拟量输出	ATR	Analog Trigger	模拟量触发
DI	Digital Input	数字量单向输入	DTR	Digital Trigger	数字量触发
DO	Digital Output	数字量单向输出	Cur	Current	当前的
DIO	Digital Input/Output	数字量双向输入输出	ID	Identifier	标识
CTR	Counter	计数器或定时器	Idx	Index	索引
PARAM/Param	Parameter	参数	DI	Differential	差分(接地方式)
TRIG/Trig	Trigger	触发	SE	Single end	单端(接地方式)
CLK	Clock	时钟	REG	Register	寄存器
GND	Ground	地	Sens	Sensitivity	灵敏度
AGND	Analog Ground	模拟地	Pt	Point	点
DGND	Digital Ground	数字地	Pts	Points	点数
Lgc	Logical	逻辑的	Chan/CH	Channel	通道号
Phys	Physical	物理的	AUX	Auxiliary	辅助
Pio	Program I/O	软件 IO 传输模式	Buf	Buffer	缓冲
Int	Interrupt	中断传输模式	En	Enable	允许或使能
Dma	Direct Memory Access	直接内存存取(传输方式)	SRC/Src	Source	源
SAMP/Samp	Sample	采样			

1.2 数据类型

基本数据类型

类型名称	类型描述	数据范围	各编程语言支持类型		
			C/C++/CVI/C_Builder	Visual Basic	Pascal(Delphi)
I8	有符号 8 位整型数	-128 ~ 127	char	无此数据类型用 Byte 代替	ShortInt
U8	无符号 8 位整型数	0 ~ 255	unsigned char	Byte	Byte
I16	有符号 16 位整型数	-32768 ~ +32767	short	Integer	SamllInt
U16	无符号 16 位整型数	0 ~ 65535	unsigned short	无此数据类型用 Integer 代替	Word
I32	有符号 32 位整型数	-2147483648 ~ 2147483647	int	Long	LongInt
U32	无符号 32 位整型数	0 ~ 4294967295	unsigned int	无此数据类型用 Long 代替	LongWord/ Cardinal
I64	有符号 64 位整型数	-9223372036854775808 ~ 9223372036854775807	__int64		Int64
U64	无符号 64 位整型数	0 ~ 1844674407370955161	unsigned __int64		无此数据类型用 Int64 代替
F32	32 位单精度浮点数	-3.402823e38 ~ 3.402823e38	float	Single	Single
F64	64 位双精度浮点数	-1.797683134862315e308 ~ 1.797683134862315e309	double	Double	Double
F64L	64 位多精度浮点数	1.189731495357231765e +4932~ 3.3621031431120935063e -4932	long double		Extnded

Visual C++扩展数据类型

Visual C++基本数据类型	Visual C++扩展数据类型	Visual C++扩展指针类型
char	CHAR	PCHAR
unsigned char	UCHAR/BYTE	PUCHAR/PBYTE
short	SHORT	PSHORT
unsigned short	WORD/USHORT	PUSHORT/PWORD
int	long/LONG/ INT/BOOL	PLONG/PINT/PBOOL
unsigned long	ULONG	PULONG
float	FLOAT	PFLOAT
double	无	无

布尔变量数据类型

编程语言类型	布尔变量命名	字节数
Visual C++	bool	1
	BOOL	4
Visual Basic	Boolean	2(-1=真；0=假)
C++Builder	BOOL	4
Delphi	Boolean, ByteBool	1
	WordBool	2
	BOOL, LongBool	4

1.3 特别约定

为简化文字，同时又为了体现阿尔泰公司所注重的标准化、重用化、人性化、可扩展化，尽可能的延伸后续设计，保护用户的前期投资，便将文档中的产品标识前缀名“PXI8191_”省略掉，只保留其产品统一的功能群组名和动作名称，如“CreateDevice()”、“InitDeviceProAD()”等关键部分，尽可能让用户看到的就是最关心的功能部分，且只要功能一样，那么其命名形式、参数形式、参数取值也尽可能一样。但在实际的头文件和代码中此前缀是不能省略掉的。

凡是以“b”为前缀的变量或参数，均表示布尔型（bool），其取值总是为 TRUE 或 FALSE(即 1 或 0)；

凡是以“n”为前缀的变量或参数，均表示整型(integer)，包括 8 位、16 位、32 位、64 位有符号数和无符号数。（由于整型变量最普通，因此如果没有标注前缀的变量或参数，通常可以理解为整型）；

凡是以“f”为前缀的变量或参数，均表示浮点型(float/double)；

凡是变量或参数中带有“Buffer”或“Buf”等字样的，均表示为缓冲或数组或指针(指针必须指向有一定长度的连续内存空间)。

2 使用提要

2.1 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，便可如能所需，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 InitDeviceProAD()、ReadDeviceProAD_Npt()、ReadDeviceProAD_Half()等。而底层用户函数如 WriteRegisterULong()、ReadRegisterULong()、WriteRegisterByte()、ReadRegisterByte()……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，您基本上可以必参考硬件说明书，除非您需要知道板上 D 型插座等管脚分配情况。

2.2 产品二次发布

如果用户使用阿尔泰公司的某款产品已做好了应用系统的开发，准备向市场发布，那么用户需要做的部分工作有：

一、将 PXI8191.dll 从 Windows\System32 中复制到安装包中；

二、将 PXI8191.inf、PXI8191.sys 从安装光盘相应产品文件夹下的 Driver 中复制到安装盘中。

2.3 如何管理设备

阿尔泰的设备驱动程序采用的是面向对象编程技术，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其他函数，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_X()函数（Npt 非空方式或者 Half 半满方式）可以用 hDevice 句柄实现对 AD 数据的采样读取。最后可以通过 ReleaseDeviceProAD()将 AD 设备释放以及通过 ReleaseDevice()将 hDevice 释放掉。

2.4 用非空方式取得 AD 数据

当您有了 hDevice 设备对象句柄后，便可用 InitDeviceProAD()函数初始化 AD 部件，关于采样通道、频率等的参数的设置是由这个函数的 pPara 参数结构体（PXI8191_PARA_AD）决定的。您只需要对这个 pPara 参数结构体的各个成员赋值即可实现所有硬件参数和设备状态的初始化。然后用 StartDeviceProAD()即可启动 AD 部件，开始 AD 采样，然后便可用 ReadDeviceProAD_Npt()反复读取 AD 数据以实现连续不间断采样。当您需要暂停设备时，执行 StopDeviceProAD()，当您需要关闭 AD 设备时，ReleaseDeviceProAD()便可帮您实现（但设备对象 hDevice 依然存在，若需要关闭 PXI 设备，请继续使用 ReleaseDevice()，释放设备对象 hDevice）。（注：ReadDeviceProAD_Npt()虽然主要面对批量读取，高速连续采集而设计，但亦可用它以单点或几点的方式读取 AD 数据，以满足慢速采集需要）。具体步骤如下：

(1) CreateDevice() 创建设备句柄；
(2) InitDeviceProAD()初始化 AD 设备对象；
(3) StartDeviceProAD()启动 AD 设备；
(4) ReadDeviceProAD_Npt()以非空查询方式读取 AD 数据；
(5) ReleaseDeviceProAD()释放 AD 设备；
(6) ReleaseDevice()释放设备对象；

如果每次采集参数相同的情况下，可以在第 4 步处循环进行，即可实现单点实时循环采样；如果每次采集参数有所不同，则可以在 2、3、4、5 步间重复进行。具体执行流程请看下面的图 2.1。

2.5 用半满方式取得 AD 数据

当您有了 hDevice 设备对象句柄后，便可用 InitDeviceProAD()函数初始化 AD 部件，关于采样通道、频率等的参数的设置是由这个函数的 pPara 参数结构体决定的。您只需要对这个 pPara 参数结构体的各个成员简单赋值即可实现所有硬件参数和设备状态的初始化。然后用 StartDeviceProAD()即可启动 AD 部件，开始 AD 采样，接着调用 GetDevStatusProAD()函数以查询 AD 的存储器 FIFO 的半满状态，如果达到半满状态，即可用 ReadDeviceProAD_Half()函数读取一批半满长度（或半满以下）的 AD 数据，然后再接着再查询 FIFO 的半满状态，若有效再读取，就这样反复查询状态反复读取 AD 数据即可实现连续不间断采样。当您需要暂停设备时，执行 StopDeviceProAD()，当您需要关闭 AD 设备时，ReleaseDeviceProAD()便可帮您实现（但设备对象 hDevice 依然存在，若需要关闭 PXI 设备，请继续使用 ReleaseDevice()，释放设备对象 hDevice）。（注：ReadDeviceProAD_Half()函数在半满状态有效时也可以单点或几点的方式读取 AD 数据，只是到下一次半满信号到来时的时间间隔会变得非常短，而不再是半满间隔）。具体步骤如下：

(1) CreateDevice() 创建设备句柄；
(2) InitDeviceProAD()初始化 AD 设备对象；
(3) StartDeviceProAD()启动 AD 设备；
(4) GetDevStatusProAD()查 FIFO 半满状态
(5) ReadDeviceProAD_Half()若 FIFO 状态半满，则以半满方式读取 AD 数据；
(7) ReleaseDeviceProAD()释放 AD 设备；
(6) ReleaseDevice()释放设备对象；

如果在采集参数相同的情况下，可以在第 3 与 4 步处循环进行，即可实现循环采样；如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。请参考流程图 2.2。

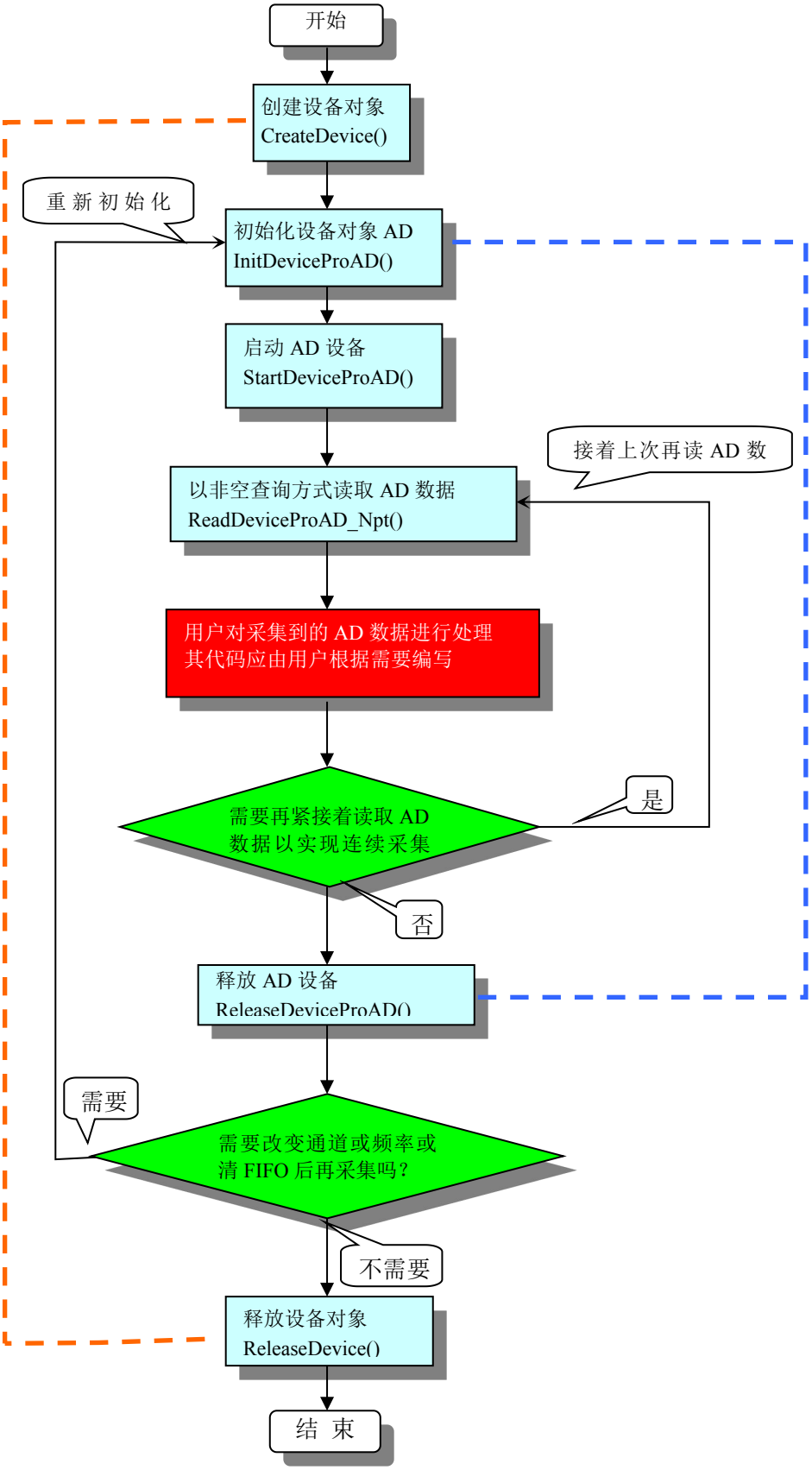


图 2.1 非空查询方式 AD 采集实现过程

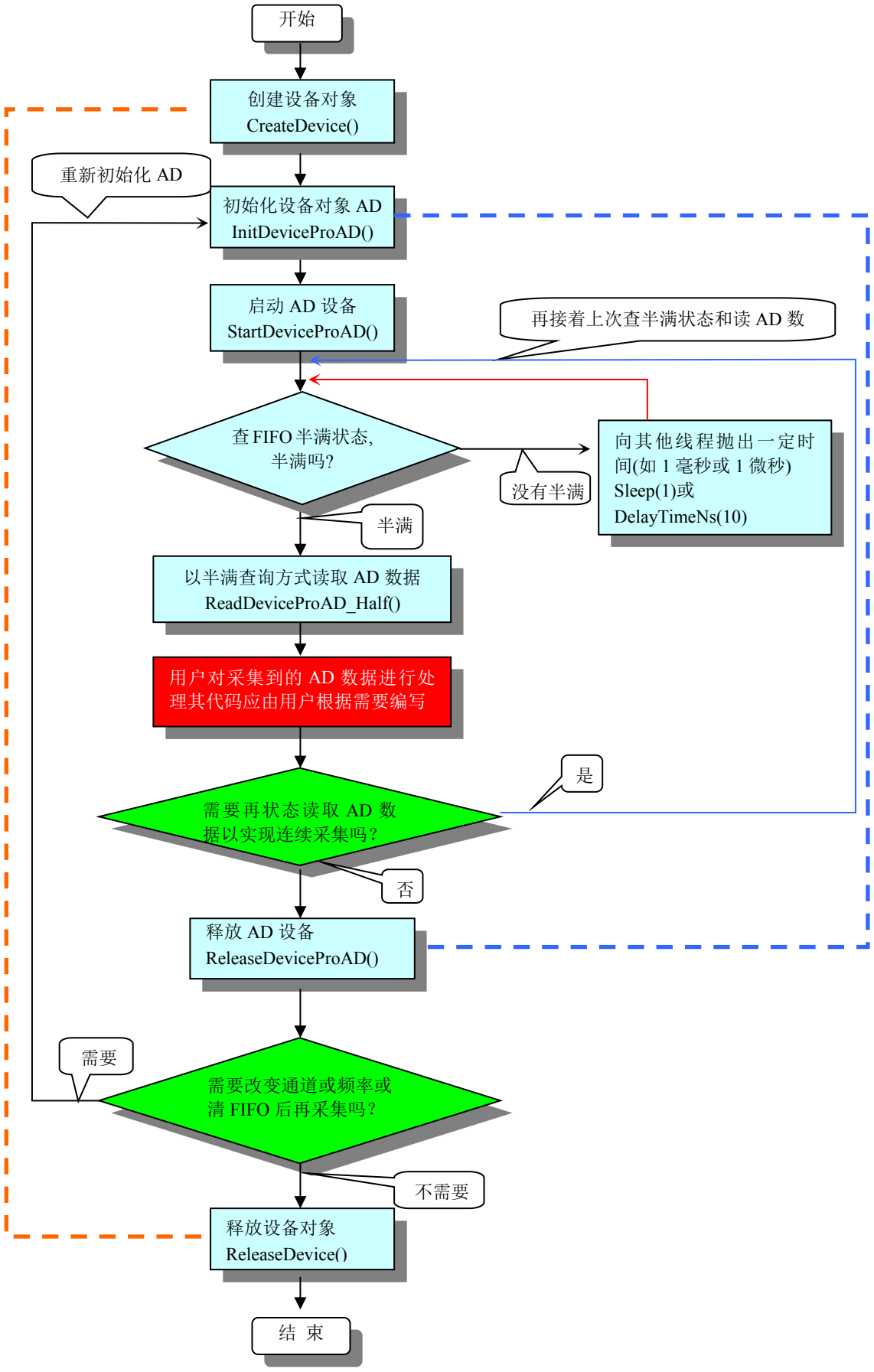


图 2.2 半满方式 AD 采集实现过程

注意：上面图 2.1、图 2.2 中虚线表示对称关系。较粗的虚线表示 CreateDevice()和 ReleaseDevice()两个函数的对称关系是：最初执行一次 CreateDevice()，在结束时就须执行一次 ReleaseDevice() (但并不是说只有 ReleaseDevice()后才能再次 CreateDevice()，因为阿尔泰的驱动程序是可以重入的)。而较细的虚线则表示 InitDeviceProAD()和 ReleaseDeviceProAD()两个函数的对称关系（即只有在 ReleaseDeviceProAD()之后才能再次 InitDeviceProAD()，切记！)。

2.6 哪些函数对您不是必须的

当公共函数如 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 PXI 用户来说，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，您可能在新操作系统中无法继续使用您原有的老设备（除非您自己愿意去编写复杂的硬件驱动程序)。

3 主要功能组函数介绍

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 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()）函数只须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 PXI 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 PXI 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《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。

3.1 设备驱动接口函数总列表（每个函数省略了前缀“PXI8191_”）

函数名	函数功能	备注
① 设备对象操作函数		
CreateDevice()	创建 PXI 设备对象	上层及底层用户
GetDeviceCount()	取得同一种 PXI 设备的总台数	上层及底层用户
GetDeviceCurrentID()	取得指定设备物理号和逻辑号	上层及底层用户
ListDeviceDlg()	列表所有同一种 PXI 设备的各种配置	上层及底层用户
ReleaseDevice()	关闭设备，且释放 PXI 总线设备对象	上层及底层用户
②AD 程序查询方式采样操作函数		
StartDeviceProAD()	启动 AD 设备，开始转换	上层用户
InitDeviceProAD()	初始化 PXI 设备上的 AD 部件准备传输	上层用户
SetDevFreqenceAD()	可动态改变 AD 采样频率	上层用户
GetDevStatusProAD ()	取得当前 PXI 设备 FIFO 半满状态	上层用户
ReadDeviceProAD_Npt()	连续读取当前 PXI 设备上的 AD 数据	上层用户
ReadDeviceProAD_Half()	连续批量读取 PXI 设备上的 AD 数据	上层用户
StopDeviceProAD()	暂停 AD 设备	上层用户
ReleaseDeviceProAD()	释放设备上的 AD 部件	上层用户
③AD 硬件参数系统保存、读取函数		
SaveParaAD()	写入设备硬件参数	上层用户
LoadParaAD()	读入硬件参数	上层用户

3.2 设备对象管理函数原型说明

CreateDevice ()

函数原型：

Visual C++ / LabWindows:

HANDLE DEVAPI FAR PASCAL PXI8191_CreateDevice(int DeviceID = 0)

Visual Basic：

Declare Function PXI8191_CreateDevice Lib "PXI8191" (Optional ByVal DeviceID As Long = 0) As Long

Visual Basic.net：

Declare Function PXI8191_CreateDevice Lib "PXI8191" (Optional ByVal DeviceID As int32 = 0) As IntPtr

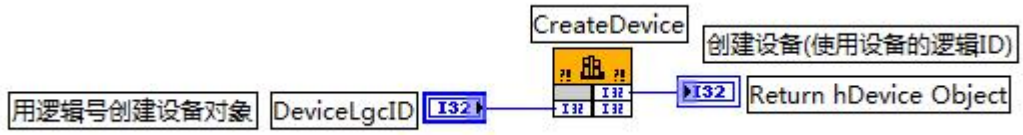
C#：

```
[DllImport("PXI8191.DLL")]
public static extern IntPtr PXI8191_CreateDevice(Int32 DeviceLgcID);
```

Dlephi：

```
Function PXI8191_CreateDevice(DeviceLgcID : Integer = 0) : LongInt; Stdcall;
                                    External 'PXI8191.dll' Name 'PXI8191_CreateDevice'
```

LabVIEW：



功能：创建设备对象，并返回其设备对象句柄 hDevice。只有成功获取 hDevice，用户才能顺利调用其它相关的接口函数以实现对设备的控制。

参数：

DeviceID 设备 ID(Identifier)标识号。当向同一个 Windows 系统中加入若干相同类型的 PXI 设备时，我们的驱动程序将以该设备的“基本名称”与 DeviceID 标识值为名称后缀的标识符来确认和管理该设备。比如若用户往 Windows 系统中加入第一个 PXI8191 AD 模板时，驱动程序则以“PXI8191”作为基本名称，再以 DeviceID 的初值组合成该设备的标识符“PXI8191-0”来确认和管理这第一个设备，若用户接着再添加第二个 PXI8191 AD 模板时，则系统将以“PXI8191-1”来确认和管理第二个设备，若再添加，则以此类推。所以当用户要创建设备句柄管理和操作第一个 PXI 设备时，DeviceID 应置 0，第二个应置 1，也以此类推。但默认值为 0。

返回值：如果执行成功，则返回设备对象句柄；如果执行失败，则返回错误码 INVALID_HANDLE_VALUE(或-1)，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。

相关函数： ReleaseDevice()

GetDeviceCount ()

函数原型：

Visual C++ / LabWindows:

```
int DEVAPI FAR PASCAL PXI8191_GetDeviceCount(HANDLE hDevice)
```

Visual Basic：

```
Declare Function PXI8191_GetDeviceCount Lib "PXI8191" (ByVal hDevice As Long) As Long
```

Visual Basic.net：

```
Declare Function PXI8191_GetDeviceCount Lib "PXI8191" (ByVal hDevice As IntPtr) As int32
```

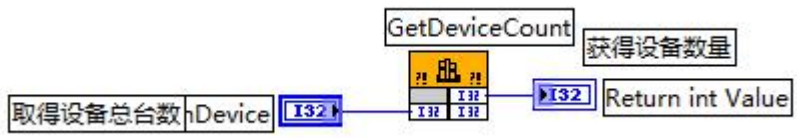
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_GetDeviceCount(IntPtr hDevice)
```

Dlephi：

```
Function PXI8191_GetDeviceCount(hDevice : LongInt): Integer; Stdcall;
                                    External 'PXI8191.dll' Name 'PXI8191_GetDeviceCount'
```

LabVIEW：



功能：取得该设备在系统中的总数量。

参数：hDevice 设备对象句柄，它应由 CreateDevice()创建。

返回值：如果有设备存在，则返回实际的设备数量，若该设备不存在，则返回 0 值，此则有两种可能的情况存在：其一，设备根本不存在，致使驱动程序无法安装加载。其二、设备存在，但其驱动程序未正确安装。对于具体原因，可以在操作系统的“设备管理器”中查看是否有该设备的信息项。在返回 0 时，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： CreateDevice() ReleaseDevice()

GetDeviceCurrentID ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_GetDeviceCurrentID(HANDLE hDevice,
                                                  PLONG DevicePhysID,
                                                  PLONG DeviceLgcID)
```

Visual Basic：

```
Declare Function PXI8191_GetDeviceCurrentID Lib "PXI8191" (ByVal hDevice As Long,
                                                           ByRef DevicePhysID As Long,
                                                           DeviceLgcID As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_GetDeviceCurrentID Lib "PXI8191" (ByVal hDevice As IntPtr,
                                                           ByRef DevicePhysID As int32,
                                                           ByRef DeviceLgcID As int32) As int32
```

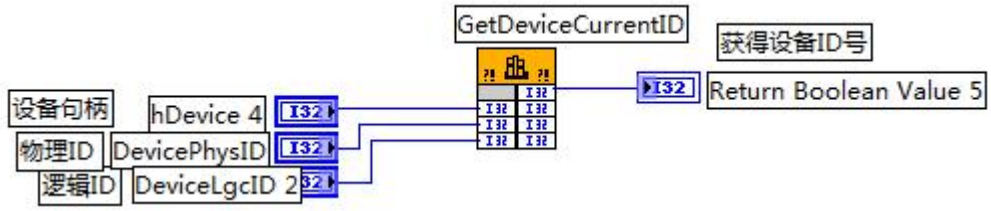
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_GetDeviceCurrentID(IntPtr hDevice,
                                                       ref Int32 DeviceLgcID,
                                                       ref Int32 DevicePhysID);
```

Dlephi：

```
Function PXI8191_GetDeviceCurrentID( hDevice    : LongInt;
                                     DeviceLgcID : Pointer;
                                     DevicePhysID : Pointer): Boolean; Stdcall;
                                     External 'PXI8191.dll' Name 'PXI8191_GetDeviceCurrentID'
```

LabVIEW：



功能：取得指定设备物理号和逻辑号。

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

DeviceLgcID 出口参数，取得设备的逻辑索引号，取值范围为[0, 255]。如果=NULL 则表示忽略此参数。

DevicePhysID 出口参数，取得设备的物理索引号，取值范围为[0, 255]，具体值由 hDevice 指定的硬件决定。如果=NULL 则表示忽略此参数。

返回值：如果成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： CreateDevice() ReleaseDevice()

ListDeviceDlg ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_ListDeviceDlg(HANDLE hDevice)
```

Visual Basic：

```
Declare Function PXI8191_ListDeviceDlg Lib "PXI8191" (ByVal hDevice As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_ListDeviceDlg Lib "PXI8191" (ByVal hDevice As IntPtr) As int32
```

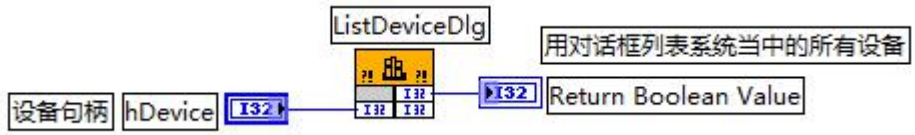
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_ListDeviceDlg(IntPtr hDevice)
```

Dlephi：

```
Function PXI8191_ListDeviceDlg( hDevice    : LongInt): Boolean;    Stdcall;
                                External 'PXI8191.dll' Name 'PXI8191_ListDeviceDlg'
```

LabVIEW：



功能：以对话框窗体方式列表系统当中的所有的该 PXI 设备。

参数：**hDevice** 入口参数，设备对象句柄，由 CreateDevice()函数创建。

返回值：如果成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： CreateDevice() ReleaseDevice()

ReleaseDevice ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_ReleaseDevice(HANDLE hDevice)
```

Visual Basic：

```
Declare Function PXI8191_ReleaseDevice Lib "PXI8191" (ByVal hDevice As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_ReleaseDevice Lib "PXI8191" (ByVal hDevice As IntPtr) As int32
```

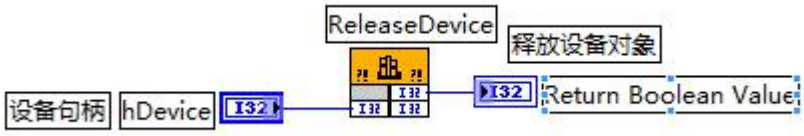
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_ReleaseDevice(IntPtr hDevice)
```

Dlephi：

```
Function PXI8191_ReleaseDevice( hDevice : LongInt): Boolean; Stdcall;
                                External 'PXI8191.dll' Name 'PXI8191_ReleaseDevice'
```

LabVIEW：



功能：释放设备对象，包括释放所占用的系统资源。

参数：**hDevice** 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

返回值：如果成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： CreateDevice()

3.3 AD 程序查询方式采样操作函数原型说明

SetDevFrequencyAD ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_SetDevFrequencyAD(
                                        HANDLE hDevice,
                                        LONG nFrequency);
```

Visual Basic：

```
Declare Function PXI8191_SetDevFrequencyAD Lib "PXI8191" (
                                        ByVal hDevice As Long, _
                                        ByVal nFrequency As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_SetDevFrequencyAD Lib "PXI8191" ( _
                                        ByVal hDevice As IntPtr, _
                                        ByVal nFrequency As int32) As int32
```

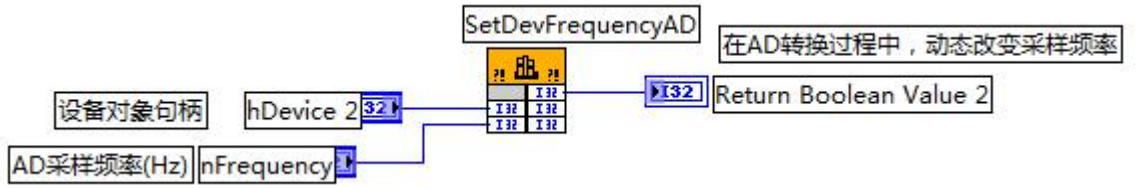
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_SetDevFrequencyAD(
                                        IntPtr hDevice,
                                        Int32 nFrequency)
```

Dlephi：

```
Function PXI8191_SetDevFrequencyAD(
                                    hDevice : LongInt;
                                    nFrequency : LongInt): Boolean;Stdcall;
                                    External 'PXI8191.dll' Name 'PXI8191_SetDevFrequencyAD'
```

LabVIEW：



功能：在 AD 转换过程中，动态改变采样频率。

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

nFrequency 入口参数，AD 采样频率(Hz)，。

返回值：如果改变采样频率成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：CreateDevice()　InitDeviceProAD ()　StartDeviceProAD ()
GetDevStatusProAD ()　ReadDeviceProAD_Npt ()　ReadDeviceProAD_Half ()
StopDeviceProAD ()　ReleaseDeviceProAD ()　ReleaseDevice()

InitDeviceProAD ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_InitDeviceProAD(
                                    HANDLE hDevice,
                                    PPXI8191_PARA_AD pADPara);
```

Visual Basic：

```
Declare Function PXI8191_InitDeviceProAD Lib "PXI8191" (
                                    ByVal hDevice As Long, _
                                    ByRef pADPara As PXI8191_PARA_AD) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_InitDeviceProAD Lib "PXI8191" ( _
                                    ByVal hDevice As IntPtr, _
                                    ByRef pADPara As PXI8191_PARA_AD) As int32
```

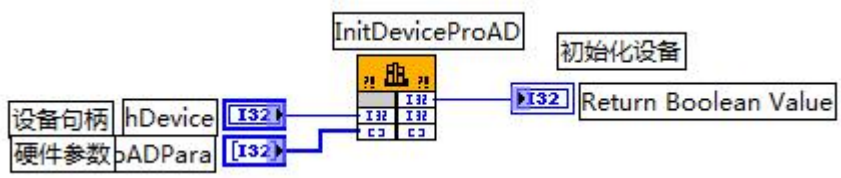
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_InitDeviceProAD(
                                    IntPtr hDevice,
                                    ref PXI8191_PARA_AD pADPara)
```

Dlephi：

```
Function PXI8191_InitDeviceProAD(
                                    hDevice : LongInt;
                                    pADPara : PPXI8191_PARA_AD): Boolean;Stdcall;
                                    External 'PXI8191.dll' Name 'PXI8191_InitDeviceProAD'
```

LabVIEW：



功能：初始化设备，当返回 TRUE 后,设备即准备就绪。

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

pADPara 入口参数，设备对象参数结构，它决定了设备对象的各种状态及工作方式,如 AD 采样通道、采样频率等。关于该参数结构，请参考《各种结构体描述》\《AD 采样的实际硬件参数（PXI8191_PARA_AD）》。

返回值：如果调用成功，则返回 TRUE，设备初始化完成，设备即准备就绪，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

CreateDevice()	SetDevFrequencyAD ()	StartDeviceProAD ()
GetDevStatusProAD ()	ReadDeviceProAD_Npt ()	ReadDeviceProAD_Half ()
StopDeviceProAD ()	ReleaseDeviceProAD ()	ReleaseDevice()

StartDeviceProAD ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_StartDeviceProAD(HANDLE hDevice)
```

Visual Basic：

```
Declare Function PXI8191_StartDeviceProAD Lib "PXI8191" (ByVal hDevice As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_StartDeviceProAD Lib "PXI8191" (ByVal hDevice As IntPtr) As int32
```

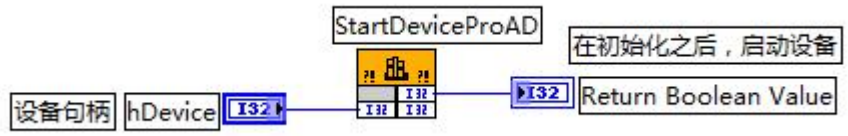
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_StartDeviceProAD(IntPtr hDevice)
```

Dlephi：

```
Function PXI8191_StartDeviceProAD( hDevice: LongInt): Boolean;Stdcall;
                                   External 'PXI8191.dll' Name 'PXI8191_StartDeviceProAD'
```

LabVIEW：



功能：启动 AD 设备，它必须在调用 InitDeviceProAD()后才能调用此函数。该函数除了启动 AD 设备开始转换以外，不改变设备的其他任何状态。

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

CreateDevice()	SetDevFrequencyAD ()	InitDeviceProAD ()
GetDevStatusProAD ()	ReadDeviceProAD_Npt ()	ReadDeviceProAD_Half ()
StopDeviceProAD ()	ReleaseDeviceProAD ()	ReleaseDevice()

GetDevStatusProAD ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_GetDevStatusProAD(
                                   HANDLE hDevice,
                                   PPXI8191_STATUS_AD pADStatus);
```

Visual Basic：

```
Declare Function PXI8191_GetDevStatusProAD Lib "PXI8191" ( _
                                   ByVal hDevice As Long, _
                                   ByRef pADStatus As PXI8191_STATUS_AD) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_GetDevStatusProAD Lib "PXI8191" ( _
                                   ByVal hDevice As IntPtr, _
                                   ByRef pADStatus As PXI8191_STATUS_AD) As int32
```

C#：

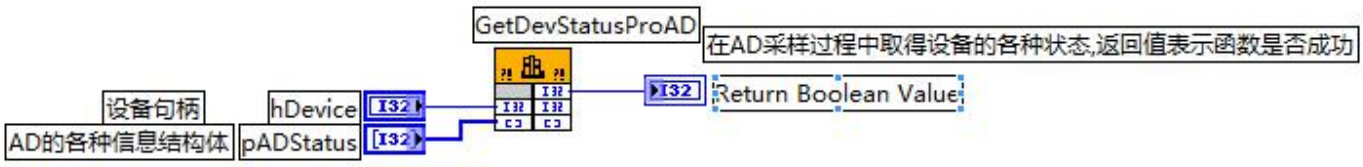
```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_GetDevStatusProAD(
                                   IntPtr hDevice,
                                   ref PXI8191_STATUS_AD pADStatus)
```

Dlephi：

```
Function PXI8191_GetDevStatusProAD(
```

```
                                                hDevice    : LongInt;
                                                pADStatus : PPXI8191_STATUS_AD): Boolean;Stdcall;
                                                External 'PXI8191.dll' Name 'PXI8191_GetDevStatusProAD'
```

LabVIEW：



功能：在半满方式读取 AD 数据时，一旦用户使用 StartDeviceProAD()后，应立即用此函数查询 FIFO 存储器的状态（半满标志、非空标志、溢出标志）。我们通常用半满标志去同步半满读操作。当半满标志有效时，再紧接着用 ReadDeviceProAD_Half ()读取 FIFO 中的半满有效 AD 数据。

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

pADStatus 出口参数，AD 设备的各种信息结构体，包括非空标志、半满标志、动态溢出标志、静态溢出标志、AD 是否正在转换以及触发标志。关于该参数结构，请参考《各种结构体描述》\《AD 采样的实际硬件参数（PXI8191_STATUS_AD)》。

返回值：如果成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。若用户选择半满查询方式读取 AD 数据：则当 GetDevStatusProAD()函数取得的 bHalf 等于 TRUE，应立即调用 ReadDeviceProAD_Half()读取 FIFO 中的半满数据。否则用户应继续循环轮询 FIFO 半满状态，直到有效为止。注意在循环轮询期间，可以用 Sleep 函数抛出一定时间给其他应用程序(包括本应用程序的主程序和其它子线程)，以提高系统的整体数据处理效率。

相关函数： CreateDevice() SetDevFrequencyAD () InitDeviceProAD ()
StartDeviceProAD () ReadDeviceProAD_Npt () ReadDeviceProAD_Half ()
StopDeviceProAD () ReleaseDeviceProAD () ReleaseDevice()

ReadDeviceProAD_Npt ()

函数原型:

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_ReadDeviceProAD_Npt(
                                                HANDLE hDevice,
                                                WORD ADBuffer[],
                                                LONG nReadSizeWords,
                                                PLONG nRetSizeWords);
```

Visual Basic：

```
Declare Function PXI8191_ReadDeviceProAD_Npt Lib "PXI8191" (_
                                                ByVal hDevice As Long, _
                                                ByRef ADBuffer As Integer, _
                                                ByVal nReadSizeWords As Long, _
                                                ByRef nRetSizeWords As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_ReadDeviceProAD_Npt Lib "PXI8191" ( _
                                                ByVal hDevice As IntPtr, _
                                                ByRef ADBuffer As uint16, _
                                                ByVal nReadSizeWords As int32, _
                                                ByRef nRetSizeWords As int32) As int32
```

C#：

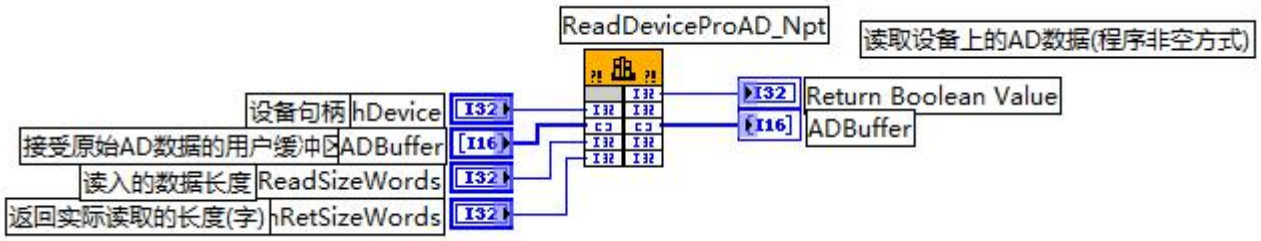
```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_ReadDeviceProAD_Npt(
                                                IntPtr hDevice,
                                                ref UInt16 ADBuffer,
```

```
                                                    Int32 nReadSizeWords,
                                                    ref Int32 nRetSizeWords);
```

Dlephi：

```
Function PXI8191_ReadDeviceProAD_Npt(
                                                    hDevice : LongInt;
                                                    ADBuffer : Pointer;
                                                    nReadSizeWords : LongInt;
                                                    nRetSizeWords : Pointer): Boolean;Stdcall;
                                                    External 'PXI8191.dll' Name 'PXI8191_ReadDeviceProAD_Npt'
```

LabVIEW：



功能：一旦用户使用 StartDeviceProAD()后，可用此函数读取设备上的 AD 数据。此函数使用 FIFO 的非空标志进行读取 AD 数据。

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

ADBuffer[] 出口参数，接受原始 AD 数据的用户缓冲区，关于如何将这些 AD 数据转换成相应的电压值，请参考《数据格式转换与排列规则》\《AD 原码 LSB 数据转换成电压值的换算方法》。

nReadSizeWords 入口参数，指定一次 ReadDeviceAD_Npt()操作应读取多少字数据到用户缓冲区。注意此参数的值不能大于用户缓冲区 ADBuffer 的最大空间。此参数值只与 ADBuffer 指定的缓冲区大小有效，而与 FIFO 存储器大小无效。

nRetSizeWords 出口参数，返回实际读取的长度(字)。

返回值： 如果函数调用成功则返回 TRUE，否则返回 FALSE，可以调用 Win32 API 函数 GetLastError()以取得进一步的错误码信息 nErrorCode。

注释：此函数也可用于单点读取和几个点的读取，只需要将 nReadSizeWords 设置成 1 或相应值即可。

相关函数： CreateDevice() SetDevFrequencyAD () InitDeviceProAD ()
StartDeviceProAD () GetDevStatusProAD () ReadDeviceProAD_Half ()
StopDeviceProAD () ReleaseDeviceProAD () ReleaseDevice()

ReadDeviceProAD_Half ()

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_ReadDeviceProAD_Half(
                                                    HANDLE hDevice,
                                                    WORD ADBuffer[],
                                                    LONG nReadSizeWords,
                                                    PLONG nRetSizeWords);
```

Visual Basic：

```
Declare Function PXI8191_ReadDeviceProAD_Half Lib "PXI8191" (_
                                                    ByVal hDevice As Long, _
                                                    ByRef ADBuffer As Integer, _
                                                    ByVal nReadSizeWords As Long, _
                                                    ByRef nRetSizeWords As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_ReadDeviceProAD_Half Lib "PXI8191" ( _
                                                    ByVal hDevice As IntPtr, _
                                                    ByRef ADBuffer As uint16, _
                                                    ByVal nReadSizeWords As int32, _
                                                    ByRef nRetSizeWords As int32) As int32
```

C#：

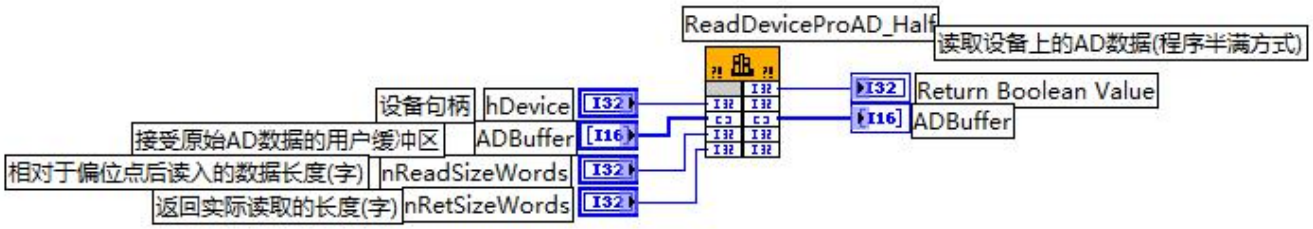
```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_ReadDeviceProAD_Half(
                                                    IntPtr hDevice,
                                                    ref UInt16 ADBuffer,
```

```
                                        Int32 nReadSizeWords,
                                        ref Int32 nRetSizeWords);
```

Dlephi：

```
Function PXI8191_ReadDeviceProAD_Half(
                                        hDevice : LongInt;
                                        ADBuffer : Pointer;
                                        nReadSizeWords : LongInt;
                                        nRetSizeWords : Pointer): Boolean;Stdcall;
                                        External 'PXI8191.dll' Name 'PXI8191_ReadDeviceProAD_Half'
```

LabVIEW：



功能：一旦用户使用 GetDevStatusProAD()后取得 pADStatus 中的 bHalf 等于 TRUE(即半满状态有效)时，可用此函数读取设备上 FIFO 中的半满 AD 数据。

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

ADBuffer[]出口参数，接受原始 AD 数据的用户缓冲区，关于如何将这些 AD 数据转换成相应的电压值，请参考《数据格式转换与排列规则》\《AD 原码 LSB 数据转换成电压值的换算方法》。

nReadSizeWords 入口参数，指定一次 ReadDeviceProAD_Half ()操作应读取多少字数据到用户缓冲区。注意此参数的值不能大于用户缓冲区 ADBuffer 的最大空间，而且应等于 FIFO 总容量的二分之一(如果用户有特殊需要可以小于 FIFO 的二分之一长)。比如设备上配置了 1K FIFO，即 1024 字，那么这个参数应指定为 512。

nRetSizeWords 出口参数，返回实际读取的长度(字)。

返回值：如果函数调用成功则返回 TRUE，否则返回 FALSE，可以调用 Win32 API 函数 GetLastError()以取得进一步的错误码信息 nErrorCode。

相关函数： CreateDevice() SetDevFrequencyAD () InitDeviceProAD ()
StartDeviceProAD () GetDevStatusProAD () ReadDeviceProAD_Npt()
StopDeviceProAD () ReleaseDeviceProAD () ReleaseDevice()

StopDeviceProAD ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_StopDeviceProAD(HANDLE hDevice);
```

Visual Basic：

```
Declare Function PXI8191_StopDeviceProAD Lib "PXI8191" (ByVal hDevice As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_StopDeviceProAD Lib "PXI8191" (ByVal hDevice As IntPtr) As int32
```

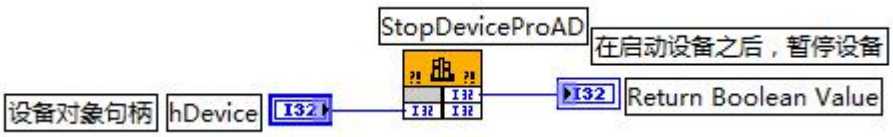
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_StopDeviceProAD(IntPtr hDevice);
```

Dlephi：

```
Function PXI8191_StopDeviceProAD( hDevice : LongInt): Boolean;Stdcall;
                                  External 'PXI8191.dll' Name 'PXI8191_StopDeviceProAD'
```

LabVIEW：



功能：在启动设备之后，暂停设备。该函数除了停止 AD 设备不再转换以外，不改变设备的其他任何状态。此后您可再调用 StartDeviceProAD()函数重新启动 AD，此时 AD 会按照暂停以前的状态（如 FIFO 存储器位置、通道位置）开始转换。

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

返回值：如果调用成功，则返回 TRUE，且 AI 立刻停止转换， 否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： CreateDevice() SetDevFrequencyAD () InitDeviceProAD ()
StartDeviceProAD () GetDevStatusProAD () ReadDeviceProAD_Npt()
ReadDeviceProAD_Half () ReleaseDeviceProAD () ReleaseDevice()

ReleaseDeviceProAD ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_ReleaseDeviceProAD(HANDLE hDevice)
```

Visual Basic：

```
Declare Function PXI8191_ReleaseDeviceProAD Lib "PXI8191" ( ByVal hDevice As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_ReleaseDeviceProAD Lib "PXI8191" ( ByVal hDevice As IntPtr) As int32
```

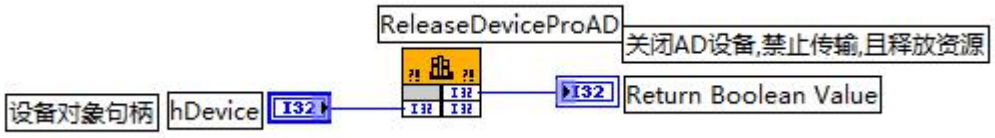
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_ReleaseDeviceProAD(IntPtr hDevice);
```

Dlephi：

```
Function PXI8191_ReleaseDeviceProAD(hDevice : LongInt): Boolean; Stdcall;
                                    External 'PXI8191.dll' Name 'PXI8191_ReleaseDeviceProAD'
```

LabVIEW：



功能：关闭 AD 设备,禁止传输,且释放资源。

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： CreateDevice() SetDevFrequencyAD () InitDeviceProAD ()
StartDeviceProAD () GetDevStatusProAD () ReadDeviceProAD_Npt()
ReadDeviceProAD_Half () StopDeviceProAD () ReleaseDevice()

3.4 AD 硬件参数操作函数原型说明

SaveParaAD ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_SaveParaAD(
                                    HANDLE hDevice,
                                    PPXI8191_PARA_AD pADPara)
```

Visual Basic：

```
Declare Function PXI8191_SaveParaAD Lib "PXI8191" (_
                                    ByVal hDevice As Long,_
                                    ByRef pADPara As PXI8191_PARA_AD) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_SaveParaAD Lib "PXI8191" (_
                                    ByVal hDevice As IntPtr,_
                                    ByRef pADPara As PXI8191_PARA_AD) As int32
```

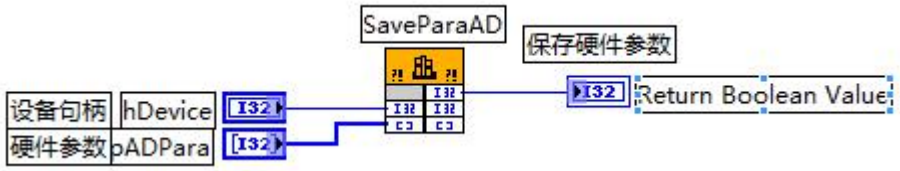
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_SaveParaAD(
                                    IntPtr hDevice,
                                    ref PXI8191_PARA_AD pADPara)
```

Dlephi：

```
Function PXI8191_SaveParaAD(
                                    hDevice : LongInt;
                                    pADPara : PPXI8191_PARA_AD): Boolean;Stdcall;
                                    External 'PXI8191.dll' Name 'PXI8191_SaveParaAD'
```

LabVIEW：



功能：将当前用户设置的硬件参数保存至系统中。

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

pADPara 入口参数，设备对象参数结构，它决定了设备对象的各种状态及工作方式,如 AD 采样通道、采样频率等。关于具体结构，请参考《各种结构体描述》\《AD 采样的实际硬件参数（PXI8191_PARA_AD）》。

返回值：如果成功，则返回 TRUE， 否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： CreateDevice () LoadParaAD() ReleaseDevice()

LoadParaAD ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_LoadParaAD(
                                    HANDLE hDevice,
                                    PPXI8191_PARA_AD pADPara)
```

Visual Basic：

```
Declare Function PXI8191_LoadParaAD Lib "PXI8191" (_
                                    ByVal hDevice As Long,_
                                    ByRef pADPara As PXI8191_PARA_AD) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_LoadParaAD Lib "PXI8191" (_
                                    ByVal hDevice As IntPtr, _
                                    ByRef pADPara As PXI8191_PARA_AD) As int32
```

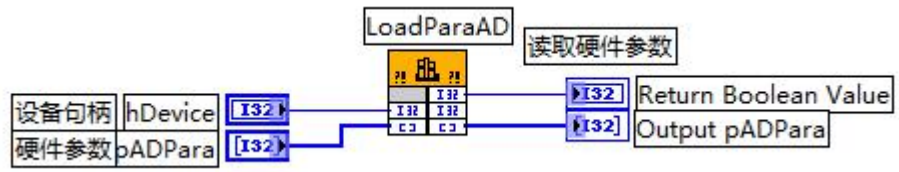
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_LoadParaAD(
                                    IntPtr hDevice,
                                    ref PXI8191_PARA_AD pADPara)
```

Dlephi：

```
Function PXI8191_LoadParaAD(
                                    hDevice : LongInt;
                                    pADPara : PPXI8191_PARA_AD): Boolean;Stdcall;
                                    External 'PXI8191.dll' Name 'PXI8191_LoadParaAD'
```

LabVIEW：



功能：将当前用户设置的硬件参数从系统中读出。

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

pADPara 入口参数，设备对象参数结构，它决定了设备对象的各种状态及工作方式,如 AD 采样通道、采样频率等。关于具体结构，请参考《各种结构体描述》\《AD 采样的实际硬件参数（PXI8191_PARA_AD）》。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： CreateDevice () SaveParaAD() ReleaseDevice()

4 各种结构体描述

4.1 AD 采样的实际硬件参数（PXI8191_PARA_AD）

Visual C++ / LabWindows:

```
typedef struct _PXI8191_PARA_AD
{
    LONG ADMode;            // AD 模式选择(连续/分组方式)
    LONG FirstChannel;      // 首通道[0,31]
    LONG LastChannel;       // 末通道[0,31],要求末通道必须大于或等于首通道
    LONG Frequency;         // 采集频率,单位为 Hz
    LONG InputRange;        // 输入量程设置
    LONG GroupInterval;     // 分组时的组间间隔(单位：微秒)
    LONG LoopsOfGroup;      // 组内循环次数[1, 65535]
    LONG TriggerMode;       // 触发模式选择
    LONG TriggerType;       // 触发类型选择(边沿触发/脉冲触发)
    LONG TriggerDir;        // 触发方向选择(正向/负向触发)
    LONG ClockSource;       // 时钟源选择(内/外时钟源)
    LONG bClockOutput;      // 允许时钟输出,=TRUE:允许输出
    LONG OutClockSource;    // 时钟输入输出源
    LONG bClockSourceDir;   // 是否将时钟信号输出到 PXI 总线,=TRUE:允许输出, =FALSE:允许输入
    LONG GroundingMode;     // 接地模式
} PXI8191_PARA_AD, *PPXI8191_PARA_AD;
```

Visual Basic：

```
Type PXI8191_PARA_AD
    ADMode          As Long
    FirstChannel    As Long
    LastChannel     As Long
    Frequency       As Long
    InputRange      As Long
    GroupInterval   As Long
    LoopsOfGroup    As Long
    TriggerMode     As Long
    TriggerType     As Long
    TriggerDir      As Long
    ClockSource     As Long
    bClockOutput    As Long
    OutClockSource  As Long
    bClockSourceDir As Long
    GroundingMode   As Long
End Type
```

Visual Basic.net：

```
Structure PXI8191_PARA_AD
    Dim ADMode          As Int32
    Dim FirstChannel    As Int32
    Dim LastChannel     As Int32
    Dim Frequency       As Int32
    Dim InputRange      As Int32
    Dim GroupInterval   As Int32
    Dim LoopsOfGroup    As Int32
    Dim TriggerMode     As Int32
    Dim TriggerType     As Int32
    Dim TriggerDir      As Int32
    Dim ClockSource     As Int32
```

```
    Dim bClockOutput        As Int32
    Dim OutClockSource      As Int32
    Dim bClockSourceDir     As Int32
    Dim GroundingMode       As Int32
End Structure
```

C#：

```
public struct PXI8191_PARA_AD
{
    public Int32 ADMode;
    public Int32 FirstChannel;
    public Int32 LastChannel;
    public Int32 Frequency;
    public Int32 InputRange;
    public Int32 GroupInterval;
    public Int32 LoopsOfGroup;
    public Int32 TriggerMode;
    public Int32 TriggerType;
    public Int32 TriggerDir;
    public Int32 ClockSource;
    public Int32 bClockOutput;
    public Int32 OutClockSource;
    public Int32 bClockSourceDir;
    public Int32 GroundingMode;
}
```

Dlephi：

```
type
PPXI8191_PARA_AD = ^PXI8191_PARA_AD;
PXI8191_PARA_AD = record
    ADMode :          LongInt;
    FirstChannel :    LongInt;
    LastChannel :     LongInt;
    Frequency :       LongInt;
    InputRange :      LongInt;
    GroupInterval :   LongInt;
    LoopsOfGroup :    LongInt;
    TriggerMode :     LongInt;
    TriggerType :     LongInt;
    TriggerDir :      LongInt;
    ClockSource :     LongInt;
    bClockOutput :    LongInt;
    OutClockSource :  LongInt;
    bClockSourceDir : LongInt;
    GroundingMode :   LongInt;
end;
```

LabVIEW:

请参考相关演示程序。

结构体作用：用于 AD 采样的实际硬件参数。用这个参数结构对设备进行硬件配置完全由 InitDeviceProAD() 函数自动完成。用户只需要对这个结构体中的各成员简单赋值即可。

结构体变量：

ADMode AD 模式选择，取值如下表：

常量名	常量值	功能定义
PXI8191_ADMODE_SEQUENCE	0x00	连续采样
PXI8191_ADMODE_GROUP	0x01	分组采样

连续采集模式：表示所有通道在采样过程中均按相等时间间隔转换，即所有被采样的点在时间轴上其间隔完全相等。即在图 4.1 中的过程，若没有 mt 的组间间隔时间 GroupInterval，即属于连续采样的模式。

分组采集模式：表示所有采样的数据点均以指定的通道数分成若干组，组内各通道数据点按等间隔采样，其频率由 Frequency 参数决定，组与组之间则有相当的间隔时间，其间隔长度由参数 GroupInterval 决定，可以精确到微

秒。如图 4.1 即是分组采样的整过情况。

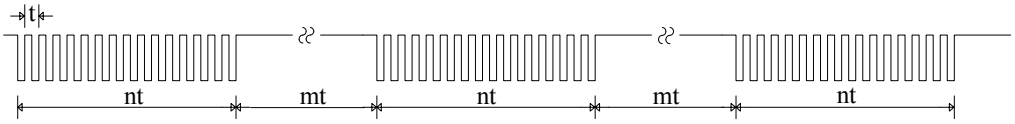


图 4.1

说明：t=1/Frequency
mt = GroupInterval
n = ChannelCount

FirstChannel 首通道选择，取值为[0,31]。

LastChannel 末通道选择，取值为[0,31]，要求末通道必须大于或等于首通道

Frequency 采集频率，单位为 Hz，其范围应根据具体的设备而定，但其最小值为 1Hz。切忌不能等于 0，本设备的 AD 采样频率取值范围为[1Hz, 250KHz]。

若为外时钟(即 ClockSource= PCI8191_CLOCKSRC_OUT)，且为连续采集（即 ADMode= PCI8191_ADMODE_SEQUENCE)时，此参数自动失效，因为外时钟代替了此参数的功能。

InputRange 输入量程设置，取值如下表：

常量名	常量值	功能定义
PXI8191_INPUT_N10000_P10000mV	0x00	±10000mV

GroupInterval 分组时的组间间隔（单位：微秒），一般情况下，此间隔时间应不小于组内相邻两通道的间隔。

LoopsOfGroup 组内循环次数，取值为[1, 65535]

TriggerMode 触发模式选择，取值如下表：

常量名	常量值	功能定义
PXI8191_TRIGMODE_SOFT	0x00	软件触发(属于内触发)
PXI8191_TRIGMODE_POST	0x01	硬件后触发(属于外触发)

TriggerType 触发类型选择，取值如下表：

常量名	常量值	功能定义
PXI8191_TRIGTYPE_EDGE	0x00	边沿触发
PXI8191_TRIGTYPE_PULSE	0x01	脉冲触发(电平)

TriggerDir 触发方向选择，取值如下表：

常量名	常量值	功能定义
PXI8191_TRIGDIR_NEGATIVE	0x00	负向触发(低脉冲/下降沿触发)
PXI8191_TRIGDIR_POSITIVE	0x01	正向触发(高脉冲/上升沿触发)
PXI8191_TRIGDIR_POSIT_NEGAT	0x02	正负向触发(高/低脉冲或上升/下降沿触发)

ClockSource 时钟源选择，取值如下表：

常量名	常量值	功能定义
PXI8191_CLOCKSRC_IN	0x00	内部时钟

bClockOutput 允许时钟输出选择，若为 TRUE，则允许输出。

OutClockSource 时钟输入输出源选择

bClockSourceDir 是否将时钟信号输出到 PXI 总线选择，若为 TRUE，则允许输出，若为 FALSE，则允许输入。

GroundingMode 接地模式选择，取值如下表：

常量名	常量值	功能定义
PXI8191_GNDMODE_SE	0x00	单端方式(SE:Single end)
PXI8191_GNDMODE_DI	0x01	双端方式(DI:Differential)

相关函数： CreateDevice () InitDeviceProAD() SaveParaAD()
LoadParaAD() ReleaseDevice()

4.2 AD 采样的实际硬件参数（PXI8191_STATUS_AD）

Visual C++ / LabWindows:

```
typedef struct _PXI8191_STATUS_AD
{
    LONG bNotEmpty;          // 板载 FIFO 存储器的非空标志，=TRUE 非空，=FALSE 空
    LONG bHalf;              // 板载 FIFO 存储器的半满标志，=TRUE 半满以上，=FALSE 半满以下
    LONG bDynamic_Overflow; // 板载 FIFO 存储器的动态溢出标志，=TRUE 已发生溢出，=FALSE 未发生溢出
```

```
    LONG bStatic_Overflow;      // 板载 FIFO 存储器的静态溢出标志，= TRUE 已发生溢出， = FALSE 未发生溢出
    LONG bConverting;           // AD 是否正在转换，=TRUE:表示正在转换，=FALS 表示转换完成
    LONG bTriggerFlag;          // 触发标志，=TRUE 表示触发事件发生，=FALSE 表示触发事件未发生
} PXI8191_STATUS_AD, *PPXI8191_STATUS_AD;
```

Visual Basic：

```
Type PXI8191_STATUS_AD
    bNotEmpty           As Long
    bHalf               As Long
    bDynamic_Overflow   As Long
    bStatic_Overflow    As Long
    bConverting         As Long
    bTriggerFlag        As Long
End Type
```

Visual Basic.net：

```
Structure PXI8191_STATUS_AD
    Dim bNotEmpty As int32
    Dim bHalf As int32
    Dim bDynamic_Overflow As int32
    Dim bStatic_Overflow As int32
    Dim bConverting As int32
    Dim bTriggerFlag As int32
End Structure
```

C#：

```
public struct PXI8191_STATUS_AD
{
    public Int32 bNotEmpty;
    public Int32 bHalf;
    public Int32 bDynamic_Overflow;
    public Int32 bStatic_Overflow;
    public Int32 bConverting;
    public Int32 bTriggerFlag;
}
```

Dlephi：

```
type
PPXI8191_STATUS_AD = ^PXI8191_STATUS_AD;
PXI8191_STATUS_AD = record
    bNotEmpty :         LongInt;
    bHalf:              LongInt;
    bDynamic_Overflow:  LongInt;
    bStatic_Overflow :  LongInt;
    bConverting :       LongInt;
    bTriggerFlag :      LongInt;
end
```

LabVIEW:

请参考相关演示程序。

结构体作用：此结构体主要用于查询 AD 的各种状态，以便同步各种数据采集和处理过程。

结构体变量：

bNotEmpty 板载 FIFO 存储器的非空标志，等于 TRUE 为非空，等于 FALSE 为空

bHalf 板载 FIFO 存储器的半满标志，等于 TRUE 为半满以上，等于 FALSE 为半满以下

bDynamic_Overflow 板载 FIFO 存储器的动态溢出标志，等于 TRUE 已发生溢出，等于 FALSE 未发生溢出

bStatic_Overflow 板载 FIFO 存储器的静态溢出标志，等于 TRUE 已发生溢出，等于 FALSE 未发生溢出

bConverting AD 是否正在转换，等于 TRUE:表示正在转换，等于 FALS 表示转换完成

bTriggerFlag 触发标志，等于 TRUE 表示触发事件发生，等于 FALSE 表示触发事件未发生

相关函数： CreateDevice()　　GetDevStatusProAD()　　ReleaseDevice()

5 数据格式转换与排列规则

5.1 AD 原码 LSB 数据转换成电压值的换算方法

在换算过程中弄清模板精度（即 Bit 位数）是很关键的，因为它决定了 LSB 数码的总宽度 CountLSB。比如 8 位的模板 CountLSB 为 256。而本设备的 AD 为 16 位，则为 65536。其他类型同理均按 2^n=LSB 总数（n 为 Bit 位数）换算即可。

设从设备上读入的某个 AD 原码数据经高位求补后为变量 Lsb，其对应的电压为变量 Volt（单位 mV）

量程(毫伏)	计算机语言换算公式	Volt 取值范围 mV
±10000mV	Volt = Lsb * (20000 / 65536)-10000	[-10000， +10000]

下面以±10000mV 量程为例加以说明：

Visual C++/ LabWindows：

```
ADData = ADBuffer[Index]&0xFFFF;
fVolt = (float)((20000.00/65536) * ADData - 10000.00);
```

Visual Basic：

```
ADData = ADBuffer(Index) And 65535
fVolt = (20000# / 65536) * ADData - 10000#
```

Visual Basic.net：

```
ADData = ADBuffer(Index) And &HFFFF
fVolt = ((20000.0 / 65536) * ADData - 10000.0)
```

C#：

```
ADData = (UInt16)(ADBuffer[Index] & 0xFFFF);
fVolt = (float)((20000.00 / 65536) * ADData - 10000.00);
```

Delphi:

```
ADData := ADBuffer[Index] and $FFFF;
fVolt := (20000.00/65536) * ADData - 10000.00;
```

5.2 AD 采集函数的 ADBuffer 缓冲区中的数据排放规则

当通道总数 ChannelCount 等于 1 时，即为单通道采集，假如此时 ChannelArray[0].ADChannel=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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	…

两通道采集(CH0 – CH1，即 ChannelArray[0].ADChannel=0;　ChannelArray[1].ADChannel=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(CH0 - CH3，即 ChannelArray[0].ADChannel=0;
ChannelArray[1].ADChannel=1;ChannelArray[2].ADChannel=2;　ChannelArray[3].ADChannel=3)

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。

6 上层用户函数接口应用实例

6.1 怎样使用 ReadDeviceProAD_Npt 函数直接取得 AD 数据

Visual C++ & C++Builder:

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[PXI8191 32 路 AD 卡 (V6.00.11)]丨[Microsoft VC++ 6.0]丨[简易代码演示]丨[非空简易应用程序]

6.2 怎样使用 ReadDeviceProAD_Half 函数直接取得 AD 数据

Visual C++ & C++Builder:

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[PXI8191 32 路 AD 卡 (V6.00.11)]丨[Microsoft VC++ 6.0]丨[简易代码演示]丨[半满简易应用程序]

7 公共函数介绍

这部分函数不参与本设备的实际操作，它只是为您编写数据采集与处理程序时的有力手段，使您编写应用程序更容易，使您的应用程序更高效。

7.1 公用接口函数总列表（每个函数省略了前缀“PXI8191_”）

函数名	函数功能	备注
①PXI 总线内存映射寄存器操作函数		
GetDeviceAddr()	取得指定 PXI 设备寄存器操作基地址	底层用户
GetDevVersion()	获取设备固件及程序版本	底层用户
WriteRegisterByte()	以字节(8Bit)方式写寄存器端口	底层用户
WriteRegisterWord()	以字(16Bit)方式写寄存器端口	底层用户
WriteRegisterULong()	以无符号双字(32Bit)方式写寄存器端口	底层用户
ReadRegisterByte()	以字节(8Bit)方式读寄存器端口	底层用户
ReadRegisterWord()	以字(16Bit)方式读寄存器端口	底层用户
ReadRegisterULong()	以无符号双字(32Bit)方式读寄存器端口	底层用户
②ISA 总线 I/O 端口操作函数		
WritePortByte()	以字节(8Bit)方式写 I/O 端口	用户程序操作端口
WritePortWord()	以字(16Bit)方式写 I/O 端口	用户程序操作端口
WritePortULong()	以无符号双字(32Bit)方式写 I/O 端口	用户程序操作端口
ReadPortByte()	以字节(8Bit)方式读 I/O 端口	用户程序操作端口
ReadPortWord()	以字(16Bit)方式读 I/O 端口	用户程序操作端口
ReadPortULong()	以无符号双字(32Bit)方式读 I/O 端口	用户程序操作端口
③文件操作函数		
CreateFileObject()	初始设备文件对象	底层用户
WriteFile()	请求文件对象写用户数据到磁盘文件	底层用户
ReadFile()	请求文件对象读数据到用户空间	底层用户
SetFileOffset()	设置文件指针偏移	底层用户
GetFileLength()	取得文件长度	底层用户
ReleaseFile()	释放已有的文件对象	底层用户
GetDiskFreeBytes()	获得指定盘符的磁盘空间(使用 64 位变量)	底层用户
④线程操作函数		
CreateSystemEvent()	创建内核事件对象	底层用户
ReleaseSystemEvent()	释放内核事件对象	底层用户
CreateVBThread()	创建 VB 子线程	底层用户
TerminateVBThread()	释放 VB 子线程	底层用户
⑤辅助函数		
kbhit()	探测用户是否有击键动作(在控制台应用程序 Console 中且在非 VC 语言中)	底层用户

7.2 内存映射寄存器直接操作及读写函数原型说明

GetDeviceAddr ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_GetDeviceAddr(
                                        HANDLE hDevice,
                                        PULONG LinearAddr,
```

```
                                        PULONG PhysAddr,
                                        int RegisterID=0);
```

Visual Basic：

```
Declare Function PXI8191_GetDeviceAddr Lib "PXI8191" (_
                                        ByVal hDevice As Long,_
                                        ByRef LinearAddr As Long,_
                                        ByRef PhysAddr As Long, _
                                        ByVal RegisterID As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_GetDeviceAddr Lib "PXI8191" ( _
                                        ByVal hDevice As IntPtr, _
                                        ByRef LinearAddr As uint32, _
                                        ByRef PhysAddr As uint32, _
                                        ByVal RegisterID As int32) As int32
```

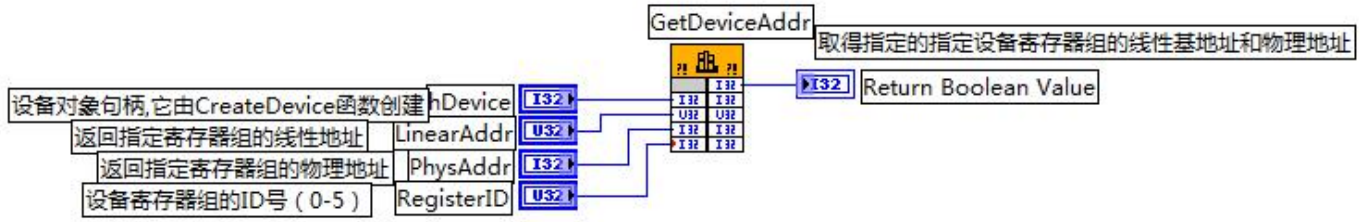
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32    PXI8191_GetDeviceAddr(
                                        IntPtr hDevice,
                                        ref UInt32 LinearAddr,
                                        ref UInt32 PhysAddr,
                                        Int32 RegisterID);
```

Dlephi：

```
Function PXI8191_GetDeviceAddr(
                                        hDevice : LongInt;
                                        LinearAddr : Pointer;
                                        PhysAddr : Pointer;
                                        RegisterID : Integer = 0) : Boolean; Stdcall;
                                        External 'PXI8191.dll' Name 'PXI8191_GetDeviceAddr';
```

LabVIEW:



功能：取得指定的指定设备 ID 号的映射寄存器的线性基地址，返回设备总数

参数:

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

LinearAddr 出口参数,指针参数，用于返回所取得的映射寄存器指向的线性地址，它可用于 WriteRegisterX 或 ReadRegisterX（X 代表 Byte、ULong、Word）等函数，以便于访问设备寄存器。它指明该设备位于系统空间的虚拟位置。

PhysAddr 出口参数，所取得的映射寄存器指向的物理地址，它指明该设备位于系统空间的物理位置。

RegisterID 入口参数，指定映射寄存器的 ID 号，其取值范围为[0，5]，通常情况下，用户应使用 0 号映射寄存器，特殊情况下，我们为用户加以申明。

返回值：如果调用成功，则返回 TRUE，它表明由 RegisterID 指定的映射寄存器的无符号 32 位线性地址和物理地址被正确返回，否则返回 FALSE，同时还要检查其 LinearAddr 和 PhysAddr 是否为 0，若为 0 则依然视为失败，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

返回值：

相关函数： GetDeviceAddr()　GetDevVersion()　WriteRegisterByte()　WriteRegisterWord()　WriteRegisterULong()　ReadRegisterByte()　ReadRegisterWord()　ReadRegisterULong()

GetDevVersion ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_GetDevVersion(
                                        HANDLE hDevice,
                                        PULONG pulFmwVersion,
                                        PULONG pulDriverVersion);
```

Visual Basic：

```
Declare Function PXI8191_GetDevVersion Lib "PXI8191" ( _
                                        ByVal hDevice As Long, _
                                        ByRef pulFmwVersion As Long, _
                                        ByRef pulDriverVersion As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_GetDevVersion Lib "PXI8191" ( _
                                        ByVal hDevice As IntPtr, _
                                        ByRef pulFmwVersion As UInt32, _
                                        ByRef pulDriverVersion As UInt32) As Int32
```

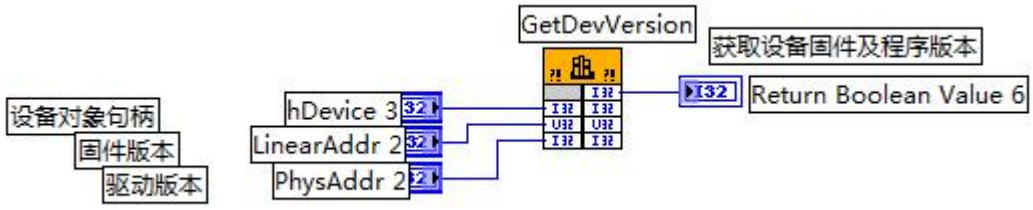
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_GetDevVersion(
                                        IntPtr hDevice,
                                        ref UInt32 pulFmwVersion,
                                        ref UInt32 pulDriverVersion)
```

Dlephi：

```
Function PXI8191_GetDevVersion(
                                        hDevice : LongInt;
                                        pulFmwVersion : PLongWord;
                                        pulDriverVersion : PLongWord): Boolean;Stdcall;
                                        External 'PXI8191.dll' Name 'PXI8191_GetDevVersion'
```

LabVIEW:



功能：获取设备固件及程序版本

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

pulFmwVersion 出口参数，获取固件版本。

pulDriverVersion 出口参数，获取驱动版本。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

GetDeviceAddr()	GetDevVersion()	WriteRegisterByte()
WriteRegisterWord()	WriteRegisterULong()	ReadRegisterByte()
ReadRegisterWord()	ReadRegisterULong()	

WriteRegisterByte ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_WriteRegisterByte(
                                        HANDLE hDevice,
                                        ULONG LinearAddr,
                                        ULONG OffsetBytes,
                                        BYTE Value);
```

Visual Basic：

```
Declare Function PXI8191_WriteRegisterByte Lib "PXI8191" ( _
                                        ByVal hDevice As Long, _
                                        ByVal LinearAddr As Long, _
                                        ByVal OffsetBytes As Long, _
                                        ByVal Value As Byte) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_WriteRegisterByte Lib "PXI8191" ( _
                                        ByVal hDevice As IntPtr, _
                                        ByVal LinearAddr As UInt32, _
                                        ByVal OffsetBytes As UInt32, _
                                        ByVal Value As Byte) As Int32
```

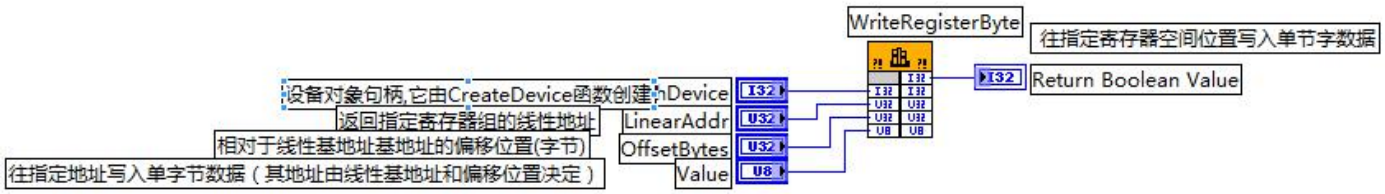
C#：

```
[DllImport("PXI8191.DLL")]
ublic static extern Int32    PXI8191_WriteRegisterByte(
                                        IntPtr hDevice,
                                        UInt32 LinearAddr,
                                        UInt32 OffsetBytes,
                                        Byte Value)
```

Dlephi：

```
Function PXI8191_WriteRegisterByte(
                                        LinearAddr : LongWord;
                                        OffsetBytes : LongWord;
                                        Value : Byte) : Boolean; Stdcall;
                                        External 'PXI8191.dll' Name 'PXI8191_WriteRegisterByte'
```

LabVIEW:



功能：往设备的映射寄存器空间指定地址写入单节字（即 8 位）数据

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

LinearAddr 入口参数，指定映射寄存器的线性基地址，它的值应由 GetDeviceAddr()确定。

OffsetBytes 入口参数，相对于基地址 LinearAddr 的偏移字节数。

Value 入口参数，需要往指定地址写入的单字节（即 8 位）数据。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：GetDeviceAddr()　GetDevVersion()　WriteRegisterByte()　WriteRegisterWord()　WriteRegisterULong()　ReadRegisterByte()　ReadRegisterWord()　ReadRegisterULong()

WriteRegisterWord ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_WriteRegisterWord(
                                        HANDLE hDevice,
                                        ULONG LinearAddr,
                                        ULONG OffsetBytes,
                                        WORD Value)
```

Visual Basic：

```
Declare Function PXI8191_WriteRegisterWord Lib "PXI8191" ( _
                                        ByVal hDevice As Long, _
                                        ByVal LinearAddr As Long, _
                                        ByVal OffsetBytes As Long, _
                                        ByVal Value As Boolean) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_WriteRegisterWord Lib "PXI8191" ( _
```

```
                                                    ByVal hDevice As IntPtr, _
                                                    ByVal LinearAddr As UInt32, _
                                                    ByVal OffsetBytes As UInt32, _
                                                    ByVal Value As uInt16) As Int32
```

C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32    PXI8191_WriteRegisterWord(
                                                    IntPtr hDevice,
                                                    UInt32 LinearAddr,
                                                    UInt32 OffsetBytes,
                                                    UInt16 Value)
```

Dlephi：

```
Function PXI8191_WriteRegisterWord(
                                                    hDevice : LongInt;
                                                    LinearAddr : LongWord;
                                                    OffsetBytes : LongWord;
                                                    Value : Word) : Boolean; Stdcall;
                                                    External 'PXI8191.dll' Name 'PXI8191_WriteRegisterWord'
```

LabVIEW:

功能：往设备的映射寄存器空间指定地址写入双节字（即 16 位）数据

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

LinearAddr 入口参数，指定映射寄存器的线性基地址，它的值应由 GetDeviceAddr()确定。

OffsetBytes 入口参数，相对于基地址 LinearAddr 的偏移字节数。

Value 入口参数，需要往指定地址写入的双字节（即 16 位）数据。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：GetDeviceAddr()　GetDevVersion()　WriteRegisterByte()　WriteRegisterWord()　WriteRegisterULong()　ReadRegisterByte()　ReadRegisterWord()　ReadRegisterULong()

WriteRegisterULong ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_WriteRegisterULong(
                                                    HANDLE hDevice,
                                                    ULONG LinearAddr,
                                                    ULONG OffsetBytes,
                                                    ULONG Value)
```

Visual Basic：

```
Declare Function PXI8191_WriteRegisterULong Lib "PXI8191" ( _
                                                    ByVal hDevice As Long, _
                                                    ByVal LinearAddr As Long, _
                                                    ByVal OffsetBytes As Long, _
                                                    ByVal Value) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_WriteRegisterULong Lib "PXI8191" ( _
                                                    ByVal hDevice As IntPtr, _
                                                    ByVal LinearAddr As UInt32, _
                                                    ByVal OffsetBytes As UInt32, _
                                                    ByVal Value As UInt32) As Int32
```

C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32    PXI8191_WriteRegisterULong(
                                                    IntPtr hDevice,
                                                    UInt32 LinearAddr,
                                                    UInt32 OffsetBytes,
                                                    UInt32 Value)
```

Dlephi：

```
Function PXI8191_WriteRegisterULong(
                                                hDevice : LongInt;
                                                LinearAddr : LongWord;
                                                OffsetBytes : LongWord;
                                                Value : LongWord) : Boolean; Stdcall;
                                                External 'PXI8191.dll' Name 'PXI8191_WriteRegisterULong'
```

LabVIEW:

功能：往设备的映射寄存器空间指定地址写入四节字（即 32 位）数据

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

LinearAddr 入口参数，指定映射寄存器的线性基地址，它的值应由 GetDeviceAddr()确定。

OffsetBytes 入口参数，相对于基地址 LinearAddr 的偏移字节数。

Value 入口参数，需要往指定地址写入的四字节（即 32 位）数据。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：GetDeviceAddr()　GetDevVersion()　WriteRegisterByte()　WriteRegisterWord()　WriteRegisterULong()　ReadRegisterByte()　ReadRegisterWord()　ReadRegisterULong()

ReadRegisterByte ()

函数原型：

Visual C++ / LabWindows:

```
BYTE DEVAPI FAR PASCAL PXI8191_ReadRegisterByte(
                                                    HANDLE hDevice,
                                                    ULONG LinearAddr,
                                                    ULONG OffsetBytes)
```

Visual Basic：

```
Declare Function PXI8191_ReadRegisterByte Lib "PXI8191" ( _
                                                    ByVal hDevice As Long, _
                                                    ByVal LinearAddr As Long, _
                                                    ByVal OffsetBytes As Long) As Byte
```

Visual Basic.net：

```
Declare Function PXI8191_ReadRegisterByte Lib "PXI8191" ( _
```

```
                                                    ByVal hDevice As IntPtr, _
                                                    ByVal LinearAddr As UInt32, _
                                                    ByVal OffsetBytes As UInt32) As Byte
```

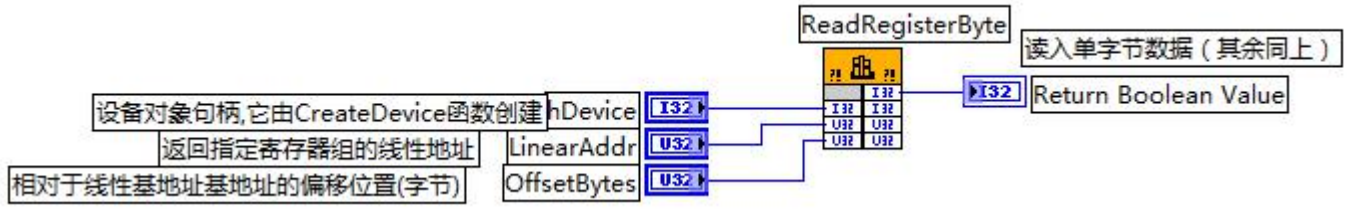
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Byte    PXI8191_ReadRegisterByte(
                                                    IntPtr hDevice,
                                                    UInt32 LinearAddr,
                                                    UInt32 OffsetBytes)
```

Dlephi：

```
Function PXI8191_ReadRegisterByte(
                                                    hDevice : LongInt;
                                                    LinearAddr : LongWord;
                                                    OffsetBytes : LongWord) : Byte; Stdcall;
                                                    External 'PXI8191.dll' Name 'PXI8191_ReadRegisterByte'
```

LabVIEW:



功能：从设备的映射寄存器空间指定地址读出单节字（即 8 位）数据

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

LinearAddr 入口参数，指定映射寄存器的线性基地址，它的值应由 GetDeviceAddr()确定。

OffsetBytes 入口参数，相对于基地址 LinearAddr 的偏移字节数。

返回值：从设备的映射寄存器空间指定地址读出单节字（即 8 位）数据。

相关函数：GetDeviceAddr()　GetDevVersion()　WriteRegisterByte()　WriteRegisterWord()　WriteRegisterULong()　ReadRegisterByte()　ReadRegisterWord()　ReadRegisterULong()

ReadRegisterWord ()

函数原型：

Visual C++ / LabWindows:

```
WORD DEVAPI FAR PASCAL PXI8191_ReadRegisterWord(
                                                    HANDLE hDevice,
                                                    ULONG LinearAddr,
                                                    ULONG OffsetBytes)
```

Visual Basic：

```
Declare Function PXI8191_ReadRegisterWord Lib "PXI8191" ( _
                                                    ByVal hDevice As Long, _
                                                    ByVal LinearAddr As Long, _
                                                    ByVal OffsetBytes As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_ReadRegisterWord Lib "PXI8191" ( _
                                                    ByVal hDevice As IntPtr, _
                                                    ByVal LinearAddr As UInt32, _
                                                    ByVal OffsetBytes As UInt32) As Int32
```

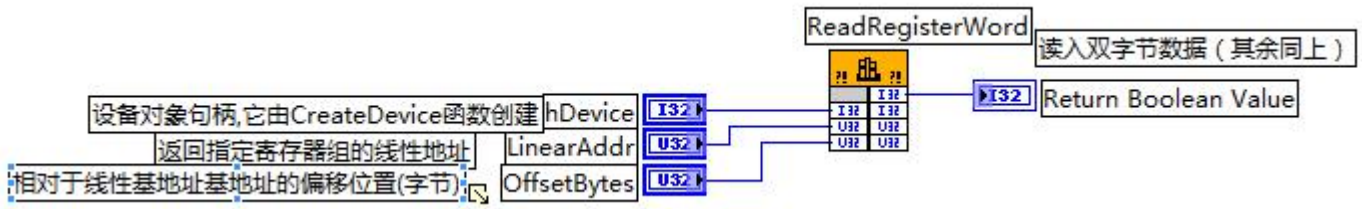
C#：

```
[DllImport("PXI8191.DLL")]
public static extern UInt16    PXI8191_ReadRegisterWord(
                                                    IntPtr hDevice,
                                                    UInt32 LinearAddr,
                                                    UInt32 OffsetBytes)
```

Dlephi：

```
Function PXI8191_ReadRegisterWord(
                                        hDevice : LongInt;
                                        LinearAddr : LongWord;
                                        OffsetBytes : LongWord) : Word; Stdcall;
                                        External 'PXI8191.dll' Name 'PXI8191_ReadRegisterWord'
```

LabVIEW:



功能：从设备的映射寄存器空间指定地址读出双节字（即 16 位）数据

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

LinearAddr 入口参数，指定映射寄存器的线性基地址，它的值应由 GetDeviceAddr()确定。

OffsetBytes 入口参数，相对于基地址 LinearAddr 的偏移字节数。

返回值：从设备的映射寄存器空间指定地址读出双节字（即 16 位）数据。

相关函数：GetDeviceAddr()　GetDevVersion()　WriteRegisterByte()　WriteRegisterWord()　WriteRegisterULong()　ReadRegisterByte()　ReadRegisterWord()　ReadRegisterULong()

ReadRegisterULong ()

函数原型：

Visual C++ / LabWindows:

```
ULONG DEVAPI FAR PASCAL PXI8191_ReadRegisterULong(
                                        HANDLE hDevice,
                                        ULONG LinearAddr,
                                        ULONG OffsetBytes)
```

Visual Basic：

```
Declare Function PXI8191_ReadRegisterULong Lib "PXI8191" ( _
                                        ByVal hDevice As Long, _
                                        ByVal LinearAddr As Long, _
                                        ByVal OffsetBytes As Long) As Long
```

Visual Basic.net：

```
Declare Function PXI8191_ReadRegisterULong Lib "PXI8191" ( _
                                        ByVal hDevice As IntPtr, _
                                        ByVal LinearAddr As UInt32, _
                                        ByVal OffsetBytes As UInt32) As UInt32
```

C#：

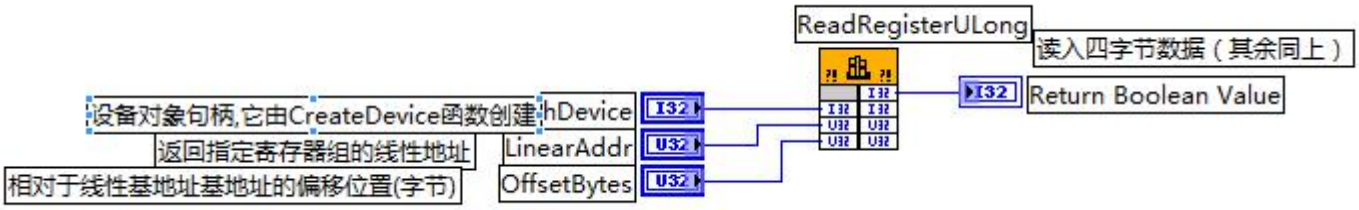
```
[DllImport("PXI8191.DLL")]
public static extern UInt32    PXI8191_ReadRegisterULong(
                                        IntPtr hDevice,
                                        UInt32 LinearAddr,
                                        UInt32 OffsetBytes)
```

Dlephi：

```
Function PXI8191_ReadRegisterULong(
                                        hDevice : LongInt;
                                        LinearAddr : LongWord;
                                        OffsetBytes : LongWord) : LongWord; Stdcall;
```

External 'PXI8191.dll' Name 'PXI8191_ReadRegisterULong'

LabVIEW:



功能：从设备的映射寄存器空间指定地址读出四节字（即 32 位）数据

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

LinearAddr 入口参数，指定映射寄存器的线性基地址，它的值应由 GetDeviceAddr()确定。

OffsetBytes 入口参数，相对于基地址 LinearAddr 的偏移字节数。

返回值：从设备的映射寄存器空间指定地址读出四节字（即 32 位）数据。

相关函数：GetDeviceAddr()　GetDevVersion()　WriteRegisterByte()　WriteRegisterWord()　WriteRegisterULong()　ReadRegisterByte()　ReadRegisterWord()　ReadRegisterULong()

7.3 I/O 端口直接操作及读写函数原型说明

注意：若您想在 WIN2K 系统的 User 模式中直接访问 I/O 端口，那么您可以安装光盘中 ISA\CommUser 目录下的公用驱动，然后调用其中的 WritePortByteEx 或 ReadPortByteEx 等有“Ex”后缀的函数即可。

WritePortByte ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_WritePortByte(
                                        HANDLE hDevice,
                                        UINT nPort,
                                        BYTE Value);
```

Visual Basic：

```
Declare Function PXI8191_WritePortByte Lib "PXI8191" ( _
                                        ByVal hDevice As Long, _
                                        ByVal nPort As Long, _
                                        ByVal Value As Byte) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_WritePortByte Lib "PXI8191" (_
                                        ByVal hDevice As IntPtr, _
                                        ByVal nPort As UInt32,_
                                        ByVal Value As Byte) As Int32
```

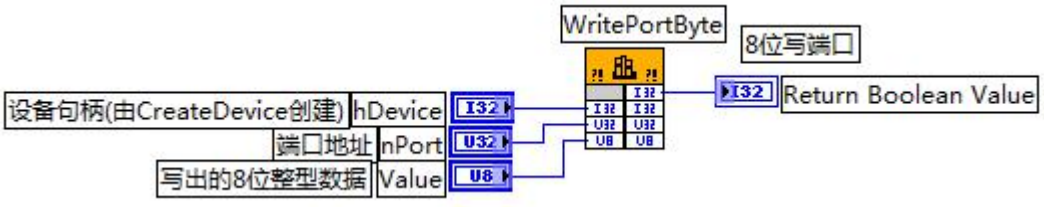
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32    PXI8191_WritePortByte(
                                        IntPtr hDevice,
                                        UInt32 nPort,
                                        Byte Value)
```

Dlephi：

```
Function PXI8191_WritePortByte(
                                        hDevice : LongInt;
                                        nPort : LongWord;
                                        Value : Byte) : Boolean; Stdcall;
                                        External 'PXI8191.dll' Name 'PXI8191_WritePortByte'
```

LabVIEW:



功能：以单字节（即 8 位）方式写 I/O 端口

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

nPort 入口参数，设备的 I/O 端口号。

Value 入口参数，写出的单字节（即 8 位）数据。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： WritePortByte() WritePortWord() WritePortULong()
ReadPortByte () ReadPortWord () ReadPortULong ()

WritePortWord ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_WritePortWord(
                                        HANDLE hDevice,
                                        UINT nPort,
                                        WORD Value)
```

Visual Basic：

```
Declare Function PXI8191_WritePortWord Lib "PXI8191" (_
                                        ByVal hDevice As Long,_
                                        ByVal nPort As Long, _
                                        ByVal Value As Boolean) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_WritePortWord Lib "PXI8191" (_
                                        ByVal hDevice As IntPtr, _
                                        ByVal nPort As UInt32,_
                                        ByVal Value As Int32) As Int32
```

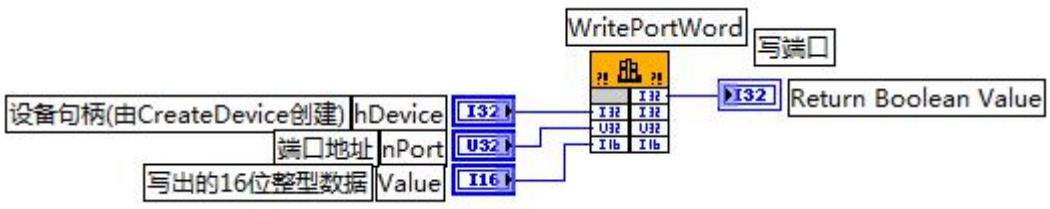
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32    PXI8191_WritePortWord(
                                        IntPtr hDevice,
                                        UInt32 nPort,
                                        UInt16 Value)
```

Dlephi：

```
Function PXI8191_WritePortWord(
                                        hDevice : LongInt;
                                        nPort : LongWord;
                                        Value : WORD) : Boolean; Stdcall;
                                        External 'PXI8191.dll' Name 'PXI8191_WritePortWord'
```

LabVIEW:



功能：以双字节（即 16 位）方式写 I/O 端口

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

nPort 入口参数，设备的 I/O 端口号。

Value 入口参数，写出的双字节（即 16 位）数据。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： WritePortByte() WritePortWord() WritePortULong()
ReadPortByte () ReadPortWord () ReadPortULong ()

WritePortULong ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_WritePortULong(
                                        HANDLE hDevice,
                                        UINT nPort,
                                        ULONG Value)
```

Visual Basic：

```
Declare Function PXI8191_WritePortULong Lib "PXI8191" (_
                                        ByVal hDevice As Long, _
                                        ByVal nPort As Long, _
                                        ByVal Value As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_WritePortULong Lib "PXI8191" (_
                                        ByVal hDevice As IntPtr,_
                                        ByVal nPort As UInt32,_
                                        ByVal Value As UInt32) As Int32
```

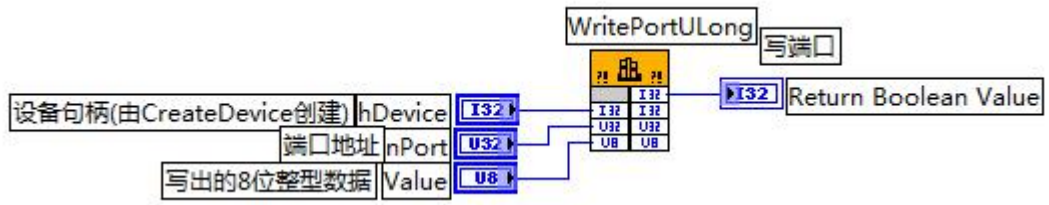
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32    PXI8191_WritePortULong(
                                        IntPtr hDevice,
                                        UInt32 nPort,
                                        UInt32 Value)
```

Dlephi：

```
Function PXI8191_WritePortULong(
                                        hDevice : LongInt;
                                        nPort : LongWord;
                                        Value : LongWord) : Boolean; Stdcall;
                                        External 'PXI8191.dll' Name 'PXI8191_WritePortULong'
```

LabVIEW:



功能：以四字节（即 32 位）方式写 I/O 端口

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

nPort 入口参数，设备的 I/O 端口号。

Value 入口参数，写出的四字节（即 32 位）数据。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

WritePortByte()	WritePortWord()	WritePortULong()
ReadPortByte ()	ReadPortWord ()	ReadPortULong ()

ReadPortByte ()

函数原型：

Visual C++ / LabWindows:

```
BYTE DEVAPI FAR PASCAL PXI8191_ReadPortByte(
                                        HANDLE hDevice,
                                        UINT nPort)
```

Visual Basic：

```
Declare Function PXI8191_ReadPortByte Lib "PXI8191" (_
                                        ByVal hDevice As Long, _
                                        ByVal nPort As Long) As Byte
```

Visual Basic.net：

```
Declare Function PXI8191_ReadPortByte Lib "PXI8191" (_
```

```
                                                    ByVal hDevice As IntPtr, _
                                                    ByVal nPort As UInt32) As Byte
```

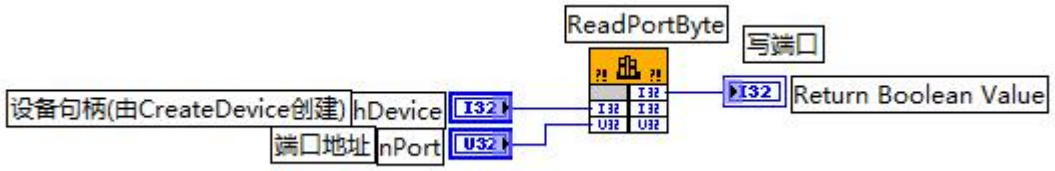
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Byte PXI8191_ReadPortByte(
                                                    IntPtr hDevice,
                                                    UInt32 nPort)
```

Dlephi：

```
Function PXI8191_ReadPortByte(
                                                    hDevice : LongInt;
                                                    nPort : LongWord) : Byte; Stdcall;
                                                    External 'PXI8191.dll' Name 'PXI8191_ReadPortByte'
```

LabVIEW:



功能：以单字节（即 8 位）方式读 I/O 端口

参数：

hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。

nPort 入口参数，设备的 I/O 端口号。

返回值：从设备的端口地址读出单字节（即 8 位）数据。

相关函数： WritePortByte() WritePortWord() WritePortULong()
ReadPortByte () ReadPortWord () ReadPortULong ()

ReadPortWord ()

函数原型：

Visual C++ / LabWindows:

```
WORD DEVAPI FAR PASCAL PXI8191_ReadPortWord(
                                                    HANDLE hDevice,
                                                    UINT nPort)
```

Visual Basic：

```
Declare Function PXI8191_ReadPortWord Lib "PXI8191" (_
                                                    ByVal hDevice As Long, _
                                                    ByVal nPort As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_ReadPortWord Lib "PXI8191" (_
                                                    ByVal hDevice As IntPtr, _
                                                    ByVal nPort As UInt32) As Int32
```

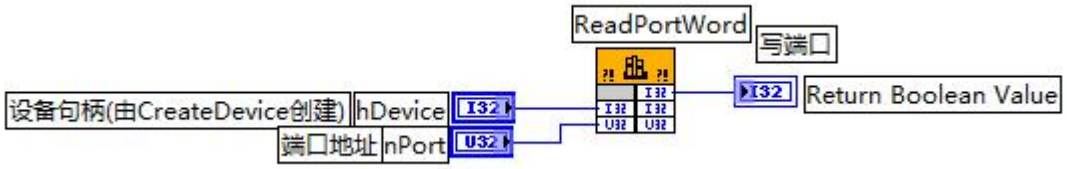
C#：

```
[DllImport("PXI8191.DLL")]
public static extern UInt16    PXI8191_ReadPortWord(
                                                    IntPtr hDevice,
                                                    UInt32 nPort)
```

Dlephi：

```
Function PXI8191_ReadPortWord(
                                                    hDevice : LongInt;
                                                    nPort : LongWord) : Word; Stdcall;
                                                    External 'PXI8191.dll' Name 'PXI8191_ReadPortWord'
```

LabVIEW:



功能：以双字节（即 16 位）方式读 I/O 端口
参数：
hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。
nPort 入口参数，设备的 I/O 端口号。
返回值：从设备的端口地址读出双字节（即 16 位）数据。
相关函数： WritePortByte() WritePortWord() WritePortULong()
ReadPortByte () ReadPortWord () ReadPortULong ()

ReadPortULong ()

函数原型：
Visual C++ / LabWindows:

```
ULONG DEVAPI FAR PASCAL PXI8191_ReadPortULong(
                                        HANDLE hDevice,
                                        UINT nPort)
```

Visual Basic：

```
Declare Function PXI8191_ReadPortULong Lib "PXI8191" (
                                        ByVal hDevice As Long,
                                        ByVal nPort As Long) As Long
```

Visual Basic.net：

```
Declare Function PXI8191_ReadPortULong Lib "PXI8191" (
                                        ByVal hDevice As IntPtr,
                                        ByVal nPort As UInt32) As UInt32
```

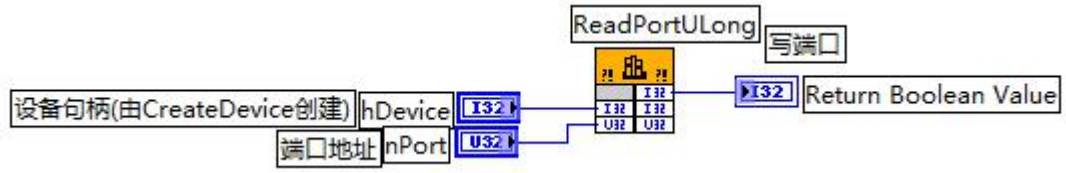
C#：

```
[DllImport("PXI8191.DLL")]
public static extern UInt32    PXI8191_ReadPortULong(
                                        IntPtr hDevice,
                                        UInt32 nPort)
```

Dlephi：

```
Function PXI8191_ReadPortULong(
                                        hDevice : LongInt;
                                        nPort : LongWord) : LongWord; Stdcall;
                                        External 'PXI8191.dll' Name 'PXI8191_ReadPortULong'
```

LabVIEW:



功能：以四字节（即 32 位）方式读 I/O 端口
参数：
hDevice 入口参数，设备对象句柄，由 CreateDevice()函数创建，该句柄指向要访问的设备。
nPort 入口参数，设备的 I/O 端口号。
返回值：从设备的端口地址读出四字节（即 32 位）数据。
相关函数： WritePortByte() WritePortWord() WritePortULong()
ReadPortByte () ReadPortWord () ReadPortULong ()

7.4 文件操作函数原型说明

CreateFileObject ()

函数原型：

Visual C++ / LabWindows:

```
HANDLE DEVAPI FAR PASCAL PXI8191_CreateFileObject(
                                                  HANDLE hDevice,
                                                  LPCTSTR strFileName,
                                                  int Mode);
```

Visual Basic：

```
Declare Function PXI8191_CreateFileObject Lib "PXI8191" ( _
                                                  ByVal hDevice As Long, _
                                                  ByVal strFileName As String, _
                                                  ByVal Mode As Long) As Long
```

Visual Basic.net：

```
Declare Function PXI8191_CreateFileObject Lib "PXI8191" ( _
                                                  ByVal hDevice As IntPtr, _
                                                  ByVal NewFileName As String, _
                                                  ByVal Mode As IntPtr) As IntPtr
```

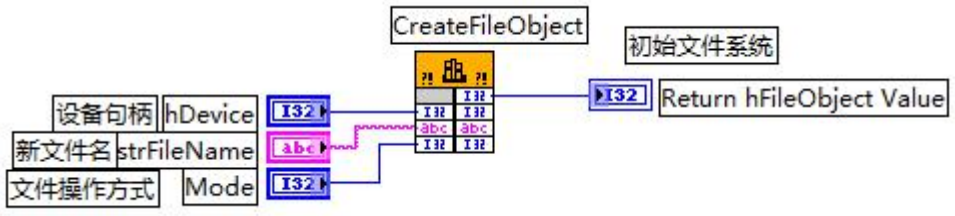
C#：

```
[DllImport("PXI8191.DLL")]
public static extern IntPtr PXI8191_CreateFileObject(
                                                  IntPtr hDevice,
                                                  ref String strFileName,
                                                  int Mode)
```

Dlephi：

```
Function PXI8191_CreateFileObject(
                                                  hDevice : LongInt;
                                                  NewFileName : Pointer;
                                                  Mode : LongInt) : LongInt; Stdcall;
                                                  External 'PXI8191.dll' Name 'PXI8191_CreateFileObject'
```

LabVIEW:



功能：初始设备文件系统

参数：

hDevice 设备对象句柄,它应由 CreateDevice()创建。

strFileName 与新文件对象关联的磁盘文件名，可以包括盘符和路径等信息。在 C 语言中，其语法格式如：”C:\\PXI8191\\Data.Dat”，在 Basic 中，其语法格式如：”C:\PXI8191\Data.Dat”。

Mode 文件操作方式，所用的文件操作方式控制字定义如下(可通过或指令实现多种方式并操作)。

常量名	常量值	功能定义
PXI8191_modeRead	0x0000	只读文件方式
PXI8191_modeWrite	0x0001	只写文件方式
PXI8191_modeReadWrite	0x0002	既读又写文件方式
PXI8191_modeCreate	0x1000	如果文件不存可以创建该文件，否则重建此文件，并清 0
PXI8191_typeText	0x4000	以文本方式操作文件

返回值：若成功，则返回文件对象句柄。

相关函数： CreateFileObject() WriteFile() ReadFile()
SetFileOffset () GetFileLength() ReleaseFile()
GetDiskFreeBytes()

WriteFile ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_WriteFile(
                                        HANDLE hFileObject,
                                        PVOID pDataBuffer,
                                        LONG nWriteSizeBytes);
```

Visual Basic：

```
Declare Function PXI8191_WriteFile Lib "PXI8191" ( _
                                        ByVal hFileObject As Long, _
                                        ByRef pDataBuffer As Integer, _
                                        ByVal nWriteSizeBytes As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_WriteFile Lib "PXI8191" ( _
                                        ByVal hFileObject As IntPtr, _
                                        ByRef pDataBuffer As Int16, _
                                        ByVal nWriteSizeBytes As UInt32) As Int32
```

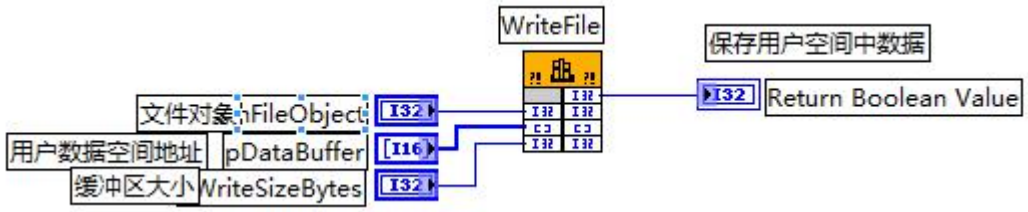
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_WriteFile(
                                        IntPtr hFileObject,
                                        ref UInt16 pDataBuffer,
                                        Int32 nWriteSizeBytes)
```

Dlephi：

```
Function PXI8191_WriteFile(
                                        hFileObject : LongInt;
                                        pDataBuffer : Pointer;
                                        nWriteSizeBytes : LongInt) : Boolean; Stdcall;
                                        External 'PXI8191.dll' Name 'PXI8191_WriteFile'
```

LabVIEW:



功能：保存用户空间中数据，通过向设备对象发送“写磁盘消息”，设备对象便会以最快的速度完成写操作。注意为了保证写入的数据是可用的，这个操作将与用户程序保持同步，但与设备对象中的环形内存池操作保持异步，以得到更高的数据吞吐量。

参数：

hFileObject 设备对象句柄，它应由 CreateFileObject()创建。

pDataBuffer 用户数据空间地址。

nWriteSizeBytes 数据缓冲区大小(字节)。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：CreateFileObject()　WriteFile()　ReadFile()　SetFileOffset ()　GetFileLength()　ReleaseFile()　GetDiskFreeBytes()

ReadFile ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_ReadFile(
                                        HANDLE hFileObject,
```

```
                                        PVOID pDataBuffer,
                                        LONG nOffsetBytes,
                                        LONG nReadSizeBytes);
```

Visual Basic：

```
Declare Function PXI8191_ReadFile Lib "PXI8191" ( _
                                        ByVal hFileObject, _
                                        ByRef pDataBuffer As Integer, _
                                        ByVal nOffsetBytes As Long, _
                                        ByVal nReadSizeBytes As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_ReadFile Lib "PXI8191" ( _
                                        ByVal hFileObject As IntPtr, _
                                        ByRef pDataBuffer As Int16, _
                                        ByVal OffsetBytes As UInt32, _
                                        ByVal nReadSizeBytes As UInt32) As Int32
```

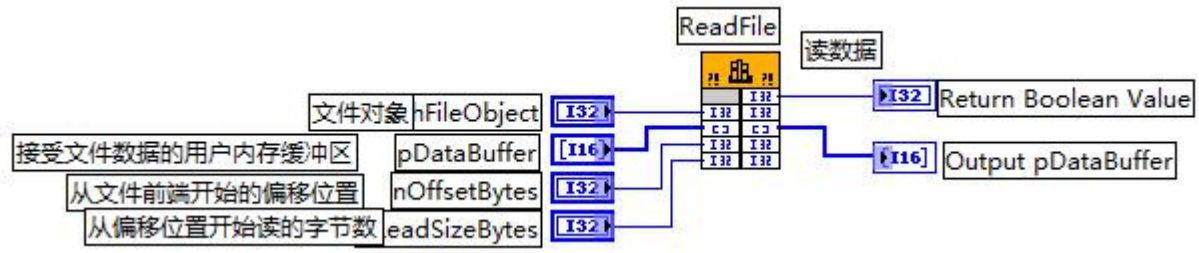
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_ReadFile(
                                        IntPtr hFileObject,
                                        ref UInt16 pDataBuffer,
                                        Int32 OffsetBytes,
                                        Int32 nReadSizeBytes)
```

Dlephi：

```
Function PXI8191_ReadFile(
                                        hFileObject : LongInt;
                                        pDataBuffer : Pointer;
                                        OffsetBytes : LongInt;
                                        nReadSizeBytes : LongInt) : Boolean; Stdcall;
                                        External 'PXI8191.dll' Name 'PXI8191_ReadFile'
```

LabVIEW:



功能：从已保存的文件读数据

参数：

hFileObject 设备对象句柄，它应由 CreateFileObject()创建。

pDataBuffer 接收文件数据的用户内存缓冲区。

nOffsetBytes 从文件前端开始的偏移位置。

nReadSizeBytes 从偏移位置开始读的字节数。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：CreateFileObject()　WriteFile()　ReadFile()
SetFileOffset ()　GetFileLength()　ReleaseFile()
GetDiskFreeBytes()

SetFileOffset ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_SetFileOffset(
                                        HANDLE hFileObject,
                                        LONG nOffsetBytes);
```

Visual Basic：

```
Declare Function PXI8191_SetFileOffset Lib "PXI8191" ( _
                                        ByVal hFileObject As Long, _
                                        ByVal nOffsetBytes As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_SetFileOffset Lib "PXI8191" ( _
                                        ByVal hFileObject As IntPtr, _
                                        ByVal nOffsetBytes As UInt32) As Int32
```

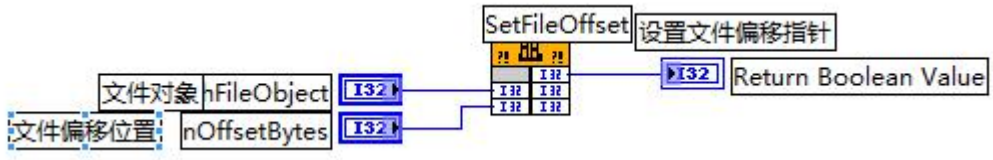
C#：

```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_SetFileOffset(
                                        IntPtr hFileObject,
                                        Int32 nOffsetBytes)
```

Dlephi：

```
Function PXI8191_SetFileOffset(
                                        hFileObject : LongInt;
                                        nOffsetBytes : LongInt) : Boolean; Stdcall;
                                        External 'PXI8191.dll' Name 'PXI8191_SetFileOffset'
```

LabVIEW:



功能：设置文件偏移指针，指定文件读写的起点。

参数：

hFileObject 文件对象句柄，它应由 CreateFileObject()创建。

nOffsetBytes 文件偏移位置（以字为单位）。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：CreateFileObject()　WriteFile()　ReadFile()　SetFileOffset ()　GetFileLength()　ReleaseFile()　GetDiskFreeBytes()

GetFileLength ()

函数原型：

Visual C++ / LabWindows:

```
ULONG DEVAPI FAR PASCAL PXI8191_GetFileLength(HANDLE hFileObject);
```

Visual Basic：

```
Declare Function PXI8191_GetFileLength Lib "PXI8191" (ByVal hFileObject As Long) As Long
```

Visual Basic.net：

```
Declare Function PXI8191_GetFileLength Lib "PXI8191" (ByVal hFileObject As IntPtr) As UInt32
```

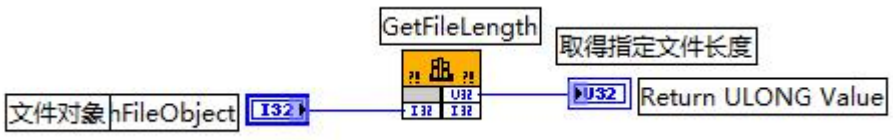
C#：

```
[DllImport("PXI8191.DLL")]
public static extern UInt32 PXI8191_GetFileLength(IntPtr hFileObject)
```

Dlephi：

```
Function PXI8191_GetFileLength(hFileObject : LongInt) : LongWord; Stdcall;
                                        External 'PXI8191.dll' Name 'PXI8191_GetFileLength'
```

LabVIEW:



功能：取得指定文件长度（字节）
参数：
hFileObject 文件对象句柄，它应由 CreateFileObject()创建。
返回值：返回指定文件的字节数
相关函数： CreateFileObject() WriteFile() ReadFile()
SetFileOffset () GetFileLength() ReleaseFile()
GetDiskFreeBytes()

ReleaseFile ()

函数原型：
Visual C++ / LabWindows:
```
BOOL DEVAPI FAR PASCAL PXI8191_ReleaseFile(HANDLE hFileObject);
```
Visual Basic：
```
Declare Function PXI8191_ReleaseFile Lib "PXI8191" (ByVal hFileObject As Long) As Boolean
```
Visual Basic.net：
```
Declare Function PXI8191_ReleaseFile Lib "PXI8191" (ByVal hFileObject As IntPtr) As Int32
```
C#：
```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_ReleaseFile(IntPtr hFileObject)
```
Dlephi：
```
Function PXI8191_ReleaseFile(hFileObject : LongInt) : Boolean; Stdcall;
                                                External 'PXI8191.dll' Name 'PXI8191_ReleaseFile'
```

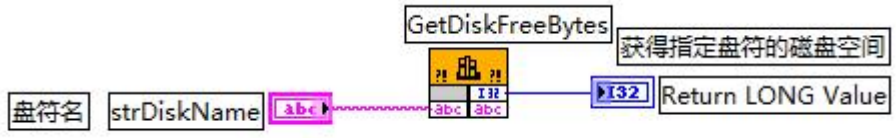
LabVIEW:

功能：释放设备文件对象
参数：
hFileObject 文件对象句柄，它应由 CreateFileObject()创建。
返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。
相关函数： CreateFileObject() WriteFile() ReadFile()
SetFileOffset () GetFileLength() ReleaseFile()
GetDiskFreeBytes()

GetDiskFreeBytes ()

函数原型：
Visual C++ / LabWindows:
```
ULONGLONG DEVAPI FAR PASCAL PXI8191_GetDiskFreeBytes(
                                                LPCTSTR strDiskName);
```
Visual Basic：
```
Declare Function PXI8191_GetDiskFreeBytes Lib "PXI8191" ( _
                                                ByVal strDiskName As String) As Boolean
```
Visual Basic.net：
```
Declare Function PXI8191_GetDiskFreeBytes Lib "PXI8191" ( _
                                                ByVal DiskName As String) As UInt64
```
C#：
```
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_GetDiskFreeBytes(
                                            ref String strDiskName)
```
Dlephi：
```
Function PXI8191_GetDiskFreeBytes(
                                            DiskName : Pointer) : Int64; Stdcall;
                                            External 'PXI8191.dll' Name 'PXI8191_GetDiskFreeBytes'
```

LabVIEW:



功能：获得指定盘符的磁盘空间(注意使用 64 位变量)
参数：
strDiskName 盘符名,如 C 盘为"C:\\", D 盘为"D:\\"。
返回值：指定盘符的磁盘空间大小
相关函数： CreateFileObject() WriteFile() ReadFile()
SetFileOffset () GetFileLength() ReleaseFile()
GetDiskFreeBytes()

7.5 线程操作函数原型说明

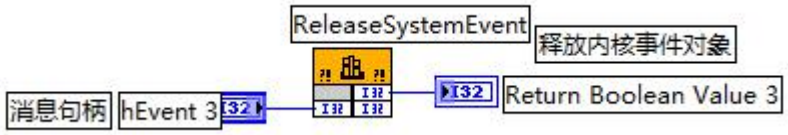
CreateSystemEvent ()

函数原型：
Visual C++ / LabWindows:
HANDLE DEVAPI FAR PASCAL PXI8191_CreateSystemEvent(void);
Visual Basic：
Declare Function PXI8191_CreateSystemEvent Lib "PXI8191" () As Long
Visual Basic.net：
Declare Function PXI8191_CreateSystemEvent Lib "PXI8191" (ByVal void) As Int32
C#：
[DllImport("PXI8191.DLL")]
public static extern IntPtr PXI8191_CreateSystemEvent();
Dlephi：
Function PXI8191_CreateSystemEvent() : LongInt; Stdcall;
External 'PXI8191.dll' Name 'PXI8191_CreateSystemEvent'
LabVIEW:

功能：创建系统内核事件对象，它将被用于中断事件响应或数据采集线程同步事件
参数：无
返回值：若成功，返回系统内核事件对象句柄，否则返回－1(或 INVALID_HANDLE_VALUE)
相关函数： CreateSystemEvent() ReleaseSystemEvent() CreateVBThread()
TerminateVBThread()

ReleaseSystemEvent ()

函数原型：
Visual C++ / LabWindows:
BOOL DEVAPI FAR PASCAL PXI8191_ReleaseSystemEvent(HANDLE hEvent);
Visual Basic：
Declare Function PXI8191_ReleaseSystemEvent Lib "PXI8191" (ByVal hEvent As Long) As Boolean
Visual Basic.net：
Declare Function PXI8191_ReleaseSystemEvent Lib "PXI8191" (ByVal hEvent As intptr) As int32
C#：
[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_ReleaseSystemEvent(IntPtr hEvent)
Dlephi：
Function PXI8191_ReleaseSystemEvent(hEvent : LongInt) : Boolean; Stdcall;
External 'PXI8191.dll' Name 'PXI8191_ReleaseSystemEvent'
LabVIEW:



功能：释放内核事件对象

参数：

hEvent。hEvent 内核事件对象。它应由 CreateSystemEvent()成功创建的对象

返回值：如果成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：CreateSystemEvent() ReleaseSystemEvent() CreateVBThread() TerminateVBThread()

CreateVBThread ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_CreateVBThread(
                                        HANDLE* hThread,
                                        LPTHREAD_START_ROUTINE RoutineAddr);
```

Visual Basic：

```
Declare Function PXI8191_CreateVBThread Lib "PXI8191" (_
                                        ByRef hThread As Long,_
                                        ByVal RoutineAddr As String) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_CreateVBThread Lib "PXI8191" (
                                        ByRef hThread As IntPtr,
                                        ByVal RoutineAddr As String) As Int32
```

功能：创建 VB 子线程

参数：

hThread 出口参数，若成功创建子线程，该参数将返回所创建的子线程的句柄，当用户操作这个子线程时将用到这个句柄，如启动线程,暂停线程,以及删除线程等。

RoutineAddr 入口参数，作为子线程运行的函数的地址，在实际使用时,请用 AddressOf 关键字取得该子线程函数的地址,再传递给 CreateVBThread()函数。

返回值：如果成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：CreateSystemEvent() ReleaseSystemEvent() CreateVBThread() TerminateVBThread()

TerminateVBThread ()

函数原型：

Visual C++ / LabWindows:

```
BOOL DEVAPI FAR PASCAL PXI8191_TerminateVBThread(HANDLE hThread);
```

Visual Basic：

```
Declare Function PXI8191_TerminateVBThread Lib "PXI8191" (ByVal hThread As Long) As Boolean
```

Visual Basic.net：

```
Declare Function PXI8191_TerminateVBThread Lib "PXI8191" (ByVal hThread As intptr) As int32
```

功能：在 VB 中删除由 CreateVBThread()创建的子线程对象

参数：

hThread 入口参数，指向需要删除的子线程对象的句柄，它应由 CreateVBThread()创建。

返回值：如果成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：CreateSystemEvent() ReleaseSystemEvent() CreateVBThread() TerminateVBThread()

7.6 辅助函数原型说明

kbhit ()

函数原型：

Visual C++ / LabWindows:

BOOL DEVAPI FAR PASCAL PXI8191_kbhit(void);

Visual Basic：

Declare Function PXI8191_kbhit Lib "PXI8191" () As Boolean

Visual Basic.net：

Declare Function PXI8191_kbhit Lib "PXI8191" () As int32

C#：

[DllImport("PXI8191.DLL")]
public static extern Int32 PXI8191_kbhit()

Dlephi：

Function PXI8191_kbhit() : Boolean; Stdcall; External 'PXI8191.dll' Name 'PXI8191_kbhit'

LabVIEW:

功能：探测用户是否有击键动作(在控制台应用程序 Console 中且在非 VC 语言中)

参数：无

返回值：如果成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。