PCIE9759C 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		Extended

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“PCIE9759C_”省略掉，只保留其产品统一的功能群组名和动作名称，如“CreateDevice()”、“InitDeviceAD()”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 InitDeviceAD()、StartDeviceAD()、ReadDeviceAD()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 PCIE9759C_32.dll 从 Windows\System32 中复制到安装包中；

二、将 PCIE9759C.inf、PCIE9759C.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 InitDeviceAD()可以使用 hDevice 句柄以程序查询方式初始化设备的 AD 部件，ReadDeviceAD()函数可以用 hDevice 句柄实现对 AD 数据的采样读取。最后可以通过 ReleaseDeviceAD()将 AD 设备释放以及通过 ReleaseDevice()将 hDevice 释放掉。

2.4 如何用 DMA 方式取得 AD 数据

当您有了 hDevice 设备对象句柄后，便可用 InitDeviceAD()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 pADPara 参数结构体（PCIE9759C_PARA_AD）决定的。您只需要对这个 pADPara 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 StartDeviceAD()即可启动 AD 部件，开始 AD 采样，然后便可用 ReadDeviceAD()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 StopDeviceAD()，当您需要关闭 AD 设备时，ReleaseDeviceAD()便可帮您实现（但设备对象 hDevice 依然存在，若需要关闭 PCIE 设备，请继续使用 ReleaseDevice()，释放设备对象 hDevice）。具体步骤如下：

(1) CreateDevice()创建设备句柄；
(2) InitDeviceAD()初始化 AD 设备对象；
(3) StartDeviceAD()启动 AD 设备；
(4) ReadDeviceAD()读取 AD 数据；
(5) ReleaseDeviceAD()释放 AD 设备对象；
(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。

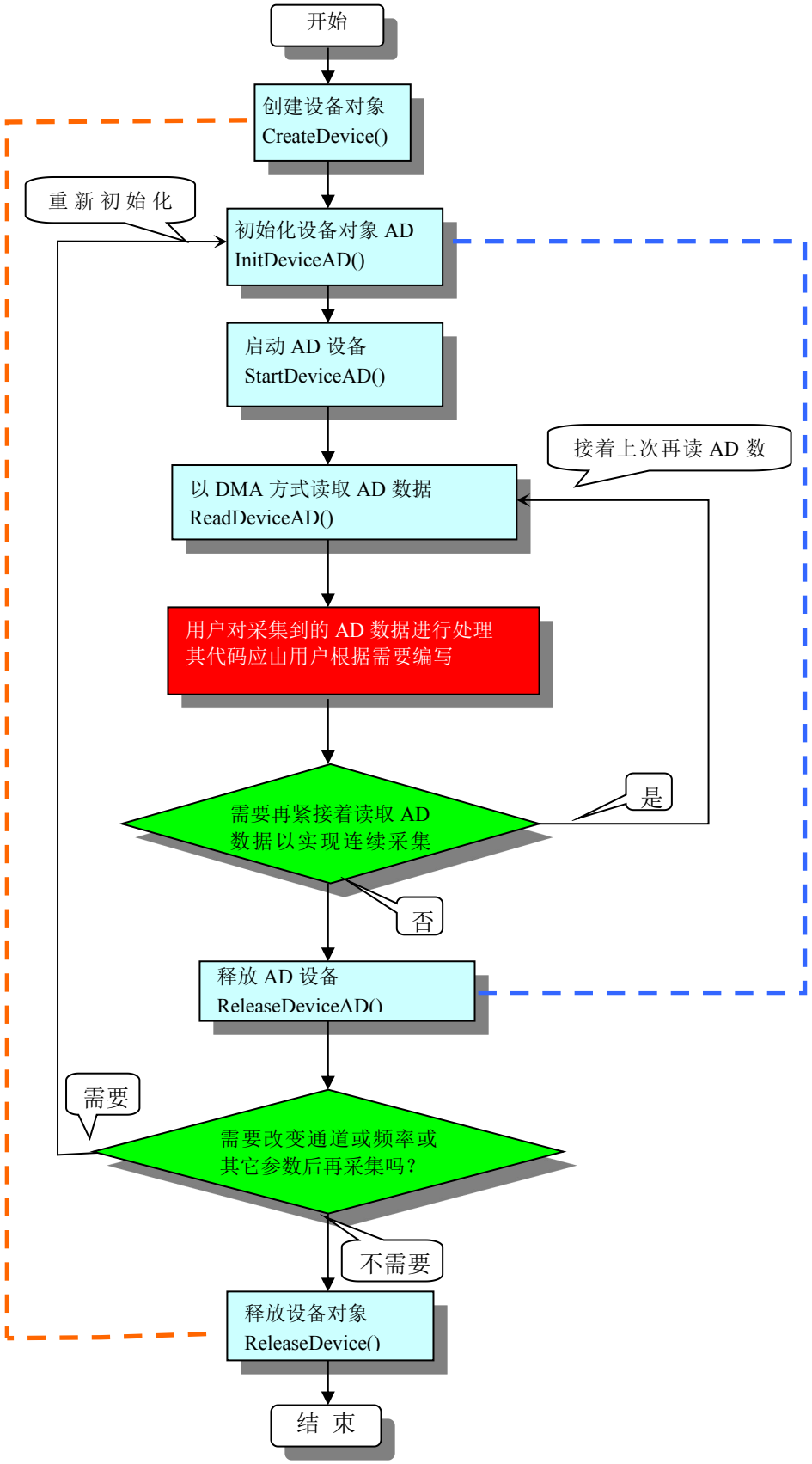

图 2.1 DMA 方式 AD 采集实现过程

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 InitDeviceAD()和 ReleaseDeviceAD()两个函数的对称关系（即只有在 ReleaseDeviceAD()之后才能再次 InitDeviceAD()，切记！)。

2.5 如何实现开关量简单操作

当您有了 hDevice 设备对象句柄后，便可用 SetDir()函数设置数字量端口各通道方向，其各路开关量的输入输出方向由其 bLineDirArray[16]中的相应元素决定，其中 bLineDirArray[n]=0 表示输入, =1 表示输出。当作为输入端口时，可以使用 ReadPort()、ReadLines()以及 ReadLine()分别读取端口的端口值、端口多线值以及端口线值；当作为输出端口时，可以使用 WritePort()、WriteLines()以及 WriteLine()分别输出端口的端口值、端口多线值以及端口线值。

2.6 哪些函数对您不是必须的

如果您使用上层用户函数访问设备，那么 GetDeviceBar()，GetDevVersion()，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 PCIE 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，我们则把这些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 InitDeviceAD()）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 ReadDeviceAD()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 PCIE 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 PCIE 的资源配置空间，而只须用 GetDeviceBar()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“PCIE9759C_”）

函数名	函数功能	备注
①设备对象管理函数		
CreateDevice()	用逻辑号创建设备对象	上层及底层用户
GetDeviceCount()	取得同一种 PCIE 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 PCIE 设备的各种配置	上层及底层用户
ReleaseDevice()	关闭设备，且释放 PCIE 总线设备对象	上层及底层用户
②AD 数据读取函数		
ADCalibration()	设备校准	上层用户
InitDeviceAD()	初始化 PCIE 设备上的 AD 部件准备传输	上层用户
StartDeviceAD()	在初始化之后，启动设备	上层用户
ReadDeviceAD()	读取设备上的 AD 数据	上层用户
StopDeviceAD()	在启动设备之后，暂停设备	上层用户
ReleaseDeviceAD()	关闭 AD 设备,禁止传输,且释放资源	上层用户
③AD 的硬件参数操作函数		
SaveParaAD()	将当前的 AD 采样参数保存至系统中	上层用户
LoadParaAD()	将 AD 采样参数从系统中读出	上层用户
ResetParaAD()	将 AD 采样参数恢复至出厂默认值	上层用户
④DIO 数字量输入输出实现函数		
SetDir()	设置数字量端口各通道方向	上层用户
GetDir()	回读数字量端口各通道方向	上层用户
ReadPort()	读数字量端口值	上层用户
WritePort()	写数字量端口值	上层用户
ReadLines()	读数字量端口多线值	上层用户
WriteLines()	写数字量端口多线值	上层用户
ReadLine()	读数字量端口线值	上层用户
WriteLine()	写数字量端口线值	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++:

HANDLE DEVAPI FAR PASCAL PCIE9759C_CreateDevice(int DeviceLgcID = 0)

功能：用逻辑号创建设备对象，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：

DeviceLgcID 设逻辑 ID(Identifier)标识号。当向同一个 Windows 系统中加入若干相同类型的 PCIE 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 PCIE9759C AD 模板时，驱动程序则以“PCIE9759C”作为基本名称，再以 DeviceID 的初值组合成该设备的标识符“PCIE9759C-0”来确认和管理这第一个设备，若用户接着再添加第二个 PCIE9759C AD 模板时，则系统将以“PCIE9759C-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 PCIE 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++:

int DEVAPI FAR PASCAL PCIE9759C_GetDeviceCount(HANDLE hDevice)

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++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_GetDeviceCurrentID(HANDLE hDevice,
                                                    PLONG DeviceLgcID,
                                                    PLONG DevicePhysID)
```

功能：取得指定设备逻辑号和物理号。

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++:

BOOL DEVAPI FAR PASCAL PCIE9759C_ListDeviceDlg(HANDLE hDevice)

功能：以对话框窗体方式列表系统当中的所有的该 PCIE 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++:

BOOL DEVAPI FAR PASCAL PCIE9759C_ReleaseDevice(HANDLE hDevice)

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 数据读取函数

ADCalibration ()

函数原型：

Visual C++:

BOOL DEVAPI FAR PASCAL PCIE9759C_ADCalibration(HANDLE hDevice);

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：如果校准设备成功，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() ADCalibration() InitDeviceAD()
StartDeviceAD() ReadDeviceAD() StopDeviceAD()
ReleaseDeviceAD() ReleaseDevice()

InitDeviceAD ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_InitDeviceAD(
                                        HANDLE hDevice,
                                        PPCIE9759C_PARA_AD pADPara);
```

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（PCIE9759C_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() ADCalibration() InitDeviceAD()
StartDeviceAD() ReadDeviceAD() StopDeviceAD()
ReleaseDeviceAD() ReleaseDevice()

StartDeviceAD ()

函数原型：

Visual C++:

BOOL DEVAPI FAR PASCAL PCIE9759C_StartDeviceAD(HANDLE hDevice)

功能：启动 AD 设备，它必须在调用 InitDeviceAD()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() ADCalibration() InitDeviceAD()
StartDeviceAD() ReadDeviceAD() StopDeviceAD()
ReleaseDeviceAD() ReleaseDevice()

ReadDeviceAD ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_ReadDeviceAD(
                                        HANDLE hDevice,
```

USHORT ADBuffer[],
ULONG nSizePoints,
PULONG pRetPoints,
PULONG pAvailSampsPoints,
double fTimeout);

功能：用此函数读取设备上的 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 入口参数，将用于接受原始 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》。。

nSizePoints 入口参数，读取数据点数，指定一次 ReadDeviceAD()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。

pRetPoints 入口参数，返回实际读取点数, =NULL,表示无须返回

pAvailSampsPoints 入口参数，剩余点数, =NULL,表示无须返回

fTimeout 入口参数，超时时间单位:秒(S)

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() ADCalibration() InitDeviceAD() StartDeviceAD() ReadDeviceAD() StopDeviceAD() ReleaseDeviceAD() ReleaseDevice()

StopDeviceAD ()

函数原型：

Visual C++:

BOOL DEVAPI FAR PASCAL PCIE9759C_StopDeviceAD(HANDLE hDevice);

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 StartDeviceAD()函数重新启动 AD，此时 AD 会按照暂停以前的状态开始转换。

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() ADCalibration() InitDeviceAD() StartDeviceAD() ReadDeviceAD() StopDeviceAD() ReleaseDeviceAD() ReleaseDevice()

ReleaseDeviceAD ()

函数原型：

Visual C++:

BOOL DEVAPI FAR PASCAL PCIE9759C_ReleaseDeviceAD(HANDLE hDevice)

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() ADCalibration() InitDeviceAD() StartDeviceAD() ReadDeviceAD() StopDeviceAD() ReleaseDeviceAD() ReleaseDevice()

3.4 AD 硬件参数操作函数原型说明

SaveParaAD ()

函数原型：

Visual C++:

BOOL DEVAPI FAR PASCAL PCIE9759C_SaveParaAD(
HANDLE hDevice,
PPCIE9759C_PARA_AD pADPara)

功能：将当前的 AD 采样参数保存至系统中。

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（PCIE9759C_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 () SaveParaAD() LoadParaAD() ResetParaAD() ReleaseDevice()

LoadParaAD ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_LoadParaAD(
                                    HANDLE hDevice,
                                    PPCIE9759C_PARA_AD pADPara)
```

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（PCIE9759C_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() LoadParaAD() ResetParaAD() ReleaseDevice()

ResetParaAD ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_ResetParaAD(
                                    HANDLE hDevice,
                                    PPCIE9759C_PARA_AD pADPara);
```

功能：将 AD 采样参数恢复至出厂默认值。

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（PCIE9759C_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() LoadParaAD() ResetParaAD() ReleaseDevice()

3.5 DIO 数字量输入输出实现函数说明

SetDir ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_SetDir(
                                    HANDLE hDevice,
                                    ULONG bLineDirArray[16]);
```

功能：设置数字量端口各通道方向。

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。

bLineDirArray[16]入口参数，线方向缓冲区，端口中各线的方向 bLineDirArray[n]=0 表示输入, =1 表示输出。

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： SetDir() GetDir() ReadPort() WritePort() ReadLines() WriteLines() ReadLine () WriteLine()

GetDir ()

函数原型：

Visual C++:

BOOL DEVAPI FAR PASCAL PCIE9759C_GetDir(
HANDLE hDevice,
ULONG bLineDirArray[16]);

功能：回读数字量端口各通道方向。

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。

bLineDirArray[16]出口参数，线方向缓冲区，端口中各线的方向 bLineDirArray[n]=0 表示当前通道为输入通道，=1 表示当前通道为输出通道。

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：SetDir() GetDir() ReadPort() WritePort() ReadLines() WriteLines() ReadLine () WriteLine()

ReadPort ()

函数原型：

Visual C++:

BOOL DEVAPI FAR PASCAL PCIE9759C_ReadPort(
HANDLE hDevice,
ULONG* pPortData);

功能：读数字量端口值。

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。

pPortData 出口参数，返回的端口数据，有效位 Bit[15:0], 每一位带表一线值，开关量输入值或开关量输出回读值。

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：SetDir() GetDir() ReadPort() WritePort() ReadLines() WriteLines() ReadLine () WriteLine()

WritePort ()

函数原型：

Visual C++:

BOOL DEVAPI FAR PASCAL PCIE9759C_ReadPort(
HANDLE hDevice,
ULONG* pPortData);

功能：写数字量端口值。

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。

pPortData 入口参数，端口数据，有效位 Bit[15:0]，每一位带表一线值，线方向为输入时写入该位的值无效，方向为输出时=0 表示关(或低)状态，=1 表示开(或高)状态。

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：SetDir() GetDir() ReadPort() WritePort() ReadLines() WriteLines() ReadLine () WriteLine()

ReadLines ()

函数原型：

Visual C++:

BOOL DEVAPI FAR PASCAL PCIE9759C_ReadLines(
HANDLE hDevice,
ULONG bLineDataArray[16]);

功能：读数字量端口多线值。

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。

bLineDataArray[16]出口参数，线数据缓冲区，每一元素带表一线值，开关量输入值或开关量输出回读值，=0 表示关(或低)状态，=1 表示开(或高)状态。

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：SetDir() GetDir() ReadPort() WritePort() ReadLines() WriteLines() ReadLine () WriteLine()

WriteLines ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_WriteLines(
                                    HANDLE hDevice,
                                    ULONG bLineDataArray[16]);
```

功能：写数字量端口多线值。

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。

bLineDataArray[16]入口参数，线数据缓冲区，每一元素带表一线值,线方向为输入时写入该位的值无效，方向为输出时=0 表示关(或低)状态，=1 表示开(或高)状态。

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：SetDir() GetDir() ReadPort() WritePort() ReadLines() WriteLines() ReadLine () WriteLine()

ReadLine ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_ReadLine(
                                    HANDLE hDevice,
                                    ULONG nLine,
                                    ULONG* pLineData);
```

功能：读数字量端口线值。

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。

nLine 入口参数，线号[0, 15]。

pLineData 出口参数，读回来的线数据，取值 0（表示低电平）或 1（表示高电平）。

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：SetDir() GetDir() ReadPort() WritePort() ReadLines() WriteLines() ReadLine () WriteLine()

WriteLine ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_WriteLine(
                                    HANDLE hDevice,
                                    ULONG nLine,
                                    ULONG bLineData);
```

功能：写数字量端口线值。

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。

nLine 入口参数，线号[0, 15]。

pLineData 入口参数，线数据，0 表示该位输出低电平，1 表示该位输出高电平。

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：SetDir() GetDir() ReadPort() WritePort() ReadLines() WriteLines() ReadLine () WriteLine()

4 各种结构体描述

4.1 AD 采样的实际硬件参数（PCIE9759C_PARA_AD）

Visual C++:

```
typedef struct _PCIE9759C_PARA_AD
{
    LONG bChannelArray[4];    // 采样通道选择阵列，分别控制 4 个通道，=1 表示该通道采样，=0 不采样
    LONG InputRange[4];       // 模拟量输入量程选择阵列，分别控制 4 个通道
    LONG RefGround[4];        // 地参考方式
    LONG Frequency;           // 采集频率, 单位为 Hz, 最高采样率 10MHz
    LONG TriggerMode;         // 触发模式选择
    LONG TriggerType;         // 触发类型选择
    LONG TriggerDir;          // 触发方向选择(正向/负向触发)
    LONG TriggerSource;       // 触发源选择
    LONG TrigLevelVolt;       // 触发电平(量程-10V~10V)
    LONG TrigWindow;          // 触发灵敏窗, 单位 25 纳秒
    LONG RefClkSrc;           // 参考时钟源选择
    LONG CnvClkSrc;           // 转换时钟源选择
    LONG bClockOutput;        // 允许转换时钟输出 CLKOUT, =TRUE:允许输出, =FALSE:禁止输出
    LONG SyncTrigSrc;         // 同步触发源选择
    LONG bMasterEn;           // 主设备使能
                              // =0：从设备，通过 Trigger 信号接收主设备发送的同步触发信号
                              // =1：主设备，通过 Trigger 信号为从设备发送自身的触发信号
} PCIE9759C_PARA_AD, *PPCIE9759C_PARA_AD;
```

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 InitDeviceAD()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：

bChannelArray[4] 采样通道选择阵列，分别控制 4 个通道，=1 表示该通道采样，=0 不采样。

InputRange[4] 模拟量输入量程选择阵列，分别控制 4 个通道，每一个位取值如下表：

常量名	常量值	功能定义
PCIE9759C_INPUT_N10000_P10000mV	0x00	±10000mV 量程
PCIE9759C_INPUT_N5000_P5000mV	0x01	±5000mV 量程
PCIE9759C_INPUT_N2500_P2500mV	0x02	±2500mV 量程
PCIE9759C_INPUT_N1250_P1250mV	0x03	±1250mV 量程

RefGround[4] 地参考方式，每一个位取值如下表：

常量名	常量值	功能定义
PCIE9759C_REFGND_RSE	0x00	单端方式
PCIE9759C_REFGND_DIFF	0x01	差分方式

Frequency 采集频率, 单位为 Hz, 最高采样率 10MHz。

TriggerMode 触发模式选择，取值如下表：

常量名	常量值	功能定义
PCIE9759C_TRIGMODE_SOFT	0x00	软件触发(属于内触发)
PCIE9759C_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
PCIE9759C_TRIGTYPE_EDGE	0x00	边沿触发
PCIE9759C_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
PCIE9759C_TRIGDIR_NEGATIVE	0x00	负向触发(低脉冲/下降沿触发)
PCIE9759C_TRIGDIR_POSITIVE	0x01	正向触发(高脉冲/上升沿触发)
PCIE9759C_TRIGDIR_POSIT_NEGAT	0x02	正负向触发(高/低脉冲或上升/下降沿触发)

TriggerSource 触发源选择，取值如下表：

常量名	常量值	功能定义
PCIE9759C_TRIGSRC_DTR	0x00	选择 DTR 作为触发源
PCIE9759C_TRIGSRC_ATR	0x01	选择 ATR 作为触发源
PCIE9759C_TRIGSRC_TRIG	0x02	Trigger 信号触发（用于多卡同步）

TrigLevelVolt 触发电平(量程-10V~10V)。

TrigWindow 触发灵敏窗，单位 25 纳秒

RefClkSrc 参考时钟源选择，取值如下表：

常量名	常量值	功能定义
PCIE9759C_REFCLKSRC_IN	0x00	使用内部参考时钟
PCIE9759C_REFCLKSRC_10M	0x01	使用主卡 10M 时钟

CnvClkSrc 转换时钟源选择，取值如下表：

常量名	常量值	功能定义
PCIE9759C_CNVCLKSRC_IN	0x00	使用内部转换时钟
PCIE9759C_CNVCLKSRC_EXT	0x01	使用外部转换时钟,由 EXT_CLK 管脚输入

bClockOutput 允许转换时钟输出 CLKOUT, =TRUE:允许输出, =FALSE:禁止输出。

SyncTrigSrc 同步触发源选择。

bMasterEn 主设备使能，=0：为从设备，通过 Trigger 信号接收主设备发送的同步触发信号；=1：为主设备，通过 Trigger 信号为从设备发送自身的触发信号；注：在多模块同步系统中，只能设定其中一个设备为主设备，其余需设定为从设备，如果系统中只有一个设备或者有多个设备但是不要求同步，需将所有设备设定为从设备

相关函数： CreateDevice () InitDeviceAD() SaveParaAD()
LoadParaAD() ResetParaAD() ReleaseDevice ()

4.2 AD 采样的实际硬件参数（PCIE9759C_STATUS_AD）

Visual C++:

```
typedef struct _PCIE9759C_STATUS_AD
{
    LONG bNotEmpty;          // 板载 FIFO 存储器的非空标志
    LONG bHalf;              // 板载 FIFO 存储器的半满标志
    LONG bDynamic_Overflow; // 板载 FIFO 存储器的动态溢出标志
    LONG bStatic_Overflow;   // 板载 FIFO 存储器的静态溢出标志
} PCIE9759C_STATUS_AD, *PPCIE9759C_STATUS_AD;
```

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

相关函数： CreateDevice() 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， +9999.69]
±5000mV	Volt = Lsb * (10000 / 65536)-5000	[-5000， +4998.77]
±2500mV	Volt = Lsb * (5000 / 65536)-2500	[-2500， +2498.55]
±1250mV	Volt = Lsb * (2500 / 65536)-1250	[-1250， +1249.38]

下面以±10000mV 量程为例加以说明：

Visual C++

```
ADData = ADBuffer[Index]&0xFFFF;
fVolt = (float)((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

当通道总数为 1 时，即为单通道采集，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

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

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 ReadDeviceAD 函数直接取得 AD 数据

Visual C++:

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

[程序]丨[阿尔泰测控演示系统]丨[PCIE9759C AD、DIO (V6.00.01)]丨[Microsoft VC++ 6.0]丨[简易代码演示]丨[AD 简易应用程序]

6.2 怎样读取与输出 DIO 多线数据

Visual C++:

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

[程序]丨[阿尔泰测控演示系统]丨[PCIE9759C AD、DIO (V6.00.01)]丨[Microsoft VC++ 6.0]丨[简易代码演示]丨[Lines 简易应用程序]

6.3 怎样读取与输出 DIO 线数据

Visual C++:

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

[程序]丨[阿尔泰测控演示系统]丨[PCIE9759C AD、DIO (V6.00.01)]丨[Microsoft VC++ 6.0]丨[简易代码演示]丨[Line 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“PCIE9759C_”）

函数名	函数功能	备注
①内存映射寄存器直接操作及读写函数		
GetDeviceBar()	取得指定的指定设备寄存器组 BAR 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
②I/O 端口直接操作及读写函数		
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附加操作函数		
CreateSystemEvent()	创建内核事件对象，供 InitDeviceDmaAD 和 VB 子线程等函数使用	底层用户
ReleaseSystemEvent()	释放内核事件对象	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

GetDeviceBar ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_GetDeviceBar(
                                        HANDLE hDevice,
                                        __int64 pbPCIBar[6]);
```

功能：取得指定的指定设备寄存器组 BAR 地址

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。

pbPCIBar[6]出口参数，返回 PCI BAR 所有地址。

返回值：如果调用成功，则返回 TRUE，它表明正确的返回了板卡的地址信息，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。

返回值：

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

GetDevVersion ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_GetDevVersion(
                                        HANDLE hDevice,
                                        PULONG pulFmwVersion,
                                        PULONG pulDriverVersion);
```

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： GetDeviceBar() GetDevVersion() WriteRegisterByte() WriteRegisterWord() WriteRegisterULong() ReadRegisterByte() ReadRegisterWord() ReadRegisterULong()

WriteRegisterByte ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_WriteRegisterByte(
                                        HANDLE hDevice,
                                        __int64 pbLinearAddr,
                                        ULONG OffsetBytes,
                                        BYTE Value);
```

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。

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

OffsetBytes 入口参数，相对于基地址 pbLinearAddr 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： GetDeviceBar() GetDevVersion() WriteRegisterByte() WriteRegisterWord() WriteRegisterULong() ReadRegisterByte() ReadRegisterWord() ReadRegisterULong()

WriteRegisterWord ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_WriteRegisterWord(
                                        HANDLE hDevice,
                                        __int64 pbLinearAddr,
                                        ULONG OffsetBytes,
                                        WORD Value)
```

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。

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

OffsetBytes 入口参数，相对于基地址 PbLinearAddr 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： GetDeviceBar() GetDevVersion() WriteRegisterByte() WriteRegisterWord() WriteRegisterULong() ReadRegisterByte() ReadRegisterWord() ReadRegisterULong()

WriteRegisterULong ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_WriteRegisterULong(
                                        HANDLE hDevice,
                                        __int64 pbLinearAddr,
                                        ULONG OffsetBytes,
                                        ULONG Value)
```

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。

PbLinearAddr 入口参数，指定映射寄存器的线性基地址，它的值应由 GetDeviceBar()确定。
OffsetBytes 入口参数，相对于基地址 PbLinearAddr 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： GetDeviceBar() GetDevVersion() WriteRegisterByte()
WriteRegisterWord() WriteRegisterULong() ReadRegisterByte()
ReadRegisterWord() ReadRegisterULong()

ReadRegisterByte ()

函数原型：
Visual C++:

```
BYTE DEVAPI FAR PASCAL PCIE9759C_ReadRegisterByte(
                                    HANDLE hDevice,
                                    __int64 pbLinearAddr,
                                    ULONG OffsetBytes)
```

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。
PbLinearAddr 入口参数，指定映射寄存器的线性基地址，它的值应由 GetDeviceBar()确定。
OffsetBytes 入口参数，相对于基地址 PbLinearAddr 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： GetDeviceBar() GetDevVersion() WriteRegisterByte()
WriteRegisterWord() WriteRegisterULong() ReadRegisterByte()
ReadRegisterWord() ReadRegisterULong()

ReadRegisterWord ()

函数原型：
Visual C++:

```
WORD DEVAPI FAR PASCAL PCIE9759C_ReadRegisterWord(
                                    HANDLE hDevice,
                                    __int64 pbLinearAddr,
                                    ULONG OffsetBytes)
```

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。
PbLinearAddr 入口参数，指定映射寄存器的线性基地址，它的值应由 GetDeviceBar()确定。
OffsetBytes 入口参数，相对于基地址 PbLinearAddr 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： GetDeviceBar() GetDevVersion() WriteRegisterByte()
WriteRegisterWord() WriteRegisterULong() ReadRegisterByte()
ReadRegisterWord() ReadRegisterULong()

ReadRegisterULong ()

函数原型：
Visual C++:

```
ULONG DEVAPI FAR PASCAL PCIE9759C_ReadRegisterULong(
                                    HANDLE hDevice,
                                    __int64 pbLinearAddr,
                                    ULONG OffsetBytes)
```

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。
PbLinearAddr 入口参数，指定映射寄存器的线性基地址，它的值应由 GetDeviceBar()确定。
OffsetBytes 入口参数，相对于基地址 PbLinearAddr 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： GetDeviceBar() 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

WritePortByte ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_WritePortByte(
                                        HANDLE hDevice,
                                        __int64 pPort,
                                        BYTE Value);
```

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。

pPort 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++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_WritePortWord(
                                        HANDLE hDevice,
                                        __int64 pPort,
                                        WORD Value)
```

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。

pPort 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++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_WritePortULong(
                                        HANDLE hDevice,
                                        __int64 pPort,
                                        ULONG Value)
```

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。

pPort 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++:

```
BYTE DEVAPI FAR PASCAL PCIE9759C_ReadPortByte(
                                        HANDLE hDevice,
```

__int64 pPort)

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。

pPort 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++:

```
WORD DEVAPI FAR PASCAL PCIE9759C_ReadPortWord(
                                    HANDLE hDevice,
                                    __int64 pPort)
```

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。

pPort 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++:

```
ULONG DEVAPI FAR PASCAL PCIE9759C_ReadPortULong(
                                    HANDLE hDevice,
                                    __int64 pPort)
```

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。

pPort 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 附加操作函数原型说明

CreateSystemEvent ()

函数原型：

Visual C++:

```
HANDLE DEVAPI FAR PASCAL PCIE9759C_CreateSystemEvent(void);
```

功能：创建内核事件对象，供 VB 子线程等函数使用

返回值：若成功，则返回内核事件对象句柄。

相关函数： CreateSystemEvent() ReleaseSystemEvent()

ReleaseSystemEvent ()

函数原型：

Visual C++:

```
BOOL DEVAPI FAR PASCAL PCIE9759C_ReleaseSystemEvent(HANDLE hEvent);
```

功能：释放内核事件对象

参数：

hEvent 内核事件对象句柄，它应由 CreateSystemEvent()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： CreateSystemEvent() ReleaseSystemEvent()