USB5561 数据采集卡

驱动程序使用手册

北京阿尔泰科技发展有限公司

V6.01.00

ART Technology

目　录

1 规范与约定

1.1 关键字缩写命名约定

缩写	全称	汉语意思	缩写	全称	汉语意思
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	FREQ/Freq	Frequency	频率

1.2 数据类型

1.2.1 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 to 127	char	无此数据类型 用 Byte 代替	ShortInt
U8	无符号 8 位整型数	0 to 255	unsigned char	Byte	Byte
I16	有符号 16 位整型数	-32768 to +32767	short	Integer	SamllInt
U16	无符号 16 位整型数	0 to 65535	unsigned short	无此数据类型 用 Integer 代替	Word
I32	有符号 32 位整型数	-2147483648 to 2147483647	int(long)	Long	LongInt
U32	无符号 32 位整型数	0 to 4294967295	unsigned int(long)	无此数据类型 用 Long 代替	LongWord/ Cardinal
I64	有符号 64 位整型数	-9223372036854775808 to 9223372036854775807	__int64		Int64
U64	无符号 64 位整型数	0 to 1844674407370955161	unsigned __int64		无此数据类型 用 Int64 代替
F32	32 位单精度浮点数	-3.402823E38 to 3.402823E38	float	Single	Single
F64	64 位双精度浮点数	-1.797683134862315E308 to 1.797683134862315E309	double	Double	Double
F64L	64 位多精度浮点数	1.189731495357231765E+4932 to 3.362103143112093506 3E-4932	long double		Extnded

1.2.2 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

1.2.3 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“USB5561_”省略掉，只保留其产品统一的功能群组名和动作名称，如“DEV_Create”、“AI_InitTask”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 驱动函数的导入方法

为了用户在演示工程中或开发工程中更明确的看出驱动头文件的信息，所有语言的驱动头文件都是以产品名称为基本名，以各种语言的相关特征为扩展名。如下表：

语言	函数接口头文件	函数接口导入库	默认所在安装位置
Microsoft Visual C++	USB5561.h	USB5561.lib	C:\ART\USB5561\Include
Microsoft Visual Basic	USB5561.bas	无	C:\ART\USB5561\Include
Borland C++ Builder	USB5561.h	USB5561.lib	C:\ART\USB5561\Include
Borland Delphi	USB5561.pas	无	C:\ART\USB5561\Include
NI LabVIEW	USB5561.vi	无	C:\ART\USB5561\Include
NI LabWindows/CVI	USB5561.h	USB5561.lib	C:\ART\USB5561\Include

注：（1）、USB5561.h 是产品的最基础的头文件，强烈建议用户关注和使用该头文件中的函数接口以实现 INT、CTR、DIO 等功能。

（2）、USB5561RSV.h 是产品的保留头文件，为了凸现 USB5561.h 中关键函数的基础功能和保证用户在使用主要函数接口的简单易用性，阿尔泰不对保留头文件（RSV）中的函数作专门的文字型说明和售后的技术支持。

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：

（1）、将 USB5561_32.dll 从 Windows\System32 中复制到安装包中；(64 位系统建议使用 USB5561_64.dll)

（2）、将 USB5561.inf、USB5561.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 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，通过调用 DEV_Create()函数可以创建无限多个设备对象的实例，并返回与设备实例关联的对象句柄 hDevice。有了这个句柄，用户就拥有了对该设备开放功能的所有控制权。如 INT_StartTask()使用 hDevice 句柄开始中断接收任务，DIO_ReadPort()函数可用实现数字量的端口数据的读取等。最后通过 DEV_Release()函数将 hDevice 释放掉。

2.4 DIO 数字量的输入输出

当用户调用 DEV_Create()函数创建了 hDevice 设备对象句柄后，可调用 DIO_ReadPort()函数实现数字量的端口输入操作，调用 DIO_ReadLines()或 DIO_ReadLine()实现数字量的线输入操作，可调用 DIO_WritePort()函数实现数字量的端口输出操作，调用 DIO_WriteLines()或 DIO_WriteLine()实现数字量的线输出操作。

2.5 INT 中断功能

当用户调用 DEV_Create()函数创建了 hDevice 设备对象句柄后，INT_InitTask()初始化中断任务，此函数返回中断事件句柄，在下面读中断数据包时可以在此事件被触发时读取。调用 INT_StartTask()函数打开接收中断事件任务，接着在中断事件被触发时即可调用 INT_ReadData()函数读中断数据包，停止读取中断事件时调用 INT_StopTask()来停止中断接收任务，之后调用 INT_ReleaseTask()来释放此次中断任务。若不打算继续使用设备句柄，可以调用 DEV_Release()释放设备句柄。

2.6 用户开发所必须的函数

首先，不管用户购买的是什么产品，其设备对象管理函数(以“DEV”为关键字段)对用户都是必须的，特别是 DEV_Create()、DEV_Release()两个函数则是必不可少的。因为对于所有的设备访问都要有这两个函数的帮助。

3 主要功能组函数介绍

3.1 DEV 设备对象管理函数原型说明

DEV_Create()

函数原型：

Visual C++ :

HANDLE DEV_Create(U32 nDeviceIdx, BOOL bUsePhysIdx)

功能：创建设备对象句柄(hDevice)，成功返回实际句柄，失败则返回 INVALID_HANDLE_VALUE(-1),可调用 GetLastError()分析错误原因。

参数：

nDeviceIdx 入口参数，设备序号(Device Index)。设备序号有两种：逻辑序号(Logical Index)和物理序号(Physical Index)。逻辑序号的定义是：当向同一台计算机系统中加入若干张相同类型的 USB 卡时，驱动程序自动以逻辑号来管理每张卡。比如某台计算机系统中插入该卡共四张，则根据操作系统的加载顺序依次分配的逻辑号为 0、1、2、3。因为每个设备的逻辑号是不能事先由用户硬性决定的，而是由操作系统加载设备时的顺序决定的。而设备物理号则可以由用户事先对硬件进行配置决定的号，这个号是固定的，跟插入 USB 的顺序没有关系。究竟使用哪一种设备号，由参数 bUsePhysIdx 决定。当使用物理序号时，其取值范围为[0, 255]。

bUserPhysIdx 入口参数，是否使用物理序号，=FALSE(0)：不使用物理序号而使用逻辑序号；=TRUE(1)：使用物理序号。所有设备均支持逻辑号，但不一定支持物理号，本设备支持。

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)，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： DEV_Create() DEV_GetCount() DEV_GetCurrentIdx() DEV_Release()

DEV_GetCount()

函数原型：

Visual C++ :

U32 DEV_GetCount(void);

功能：取得该设备在系统中的总数量(Get device count)。

参数：无。

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： DEV_Create() DEV_GetCount() DEV_GetCurrentIdx() DEV_Release()

DEV_GetCurrentIdx()

函数原型：

Visual C++ :

```
BOOL DEV_GetCurrentIdx (HANDLE hDevice，
                        U32* pLgcIdx,
                        U32* pPhysIdx);
```

功能：取得指定设备物理号和逻辑号(Get logical and physical index of the device)。

参数：

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

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

pPhysIdx 出口参数，取得设备的物理索引号(Physical Index)，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： DEV_Create()　DEV_GetCount()　DEV_GetCurrentIdx()　DEV_Release()

DEV_Release()

函数原型：

Visual C++ :

```
BOOL DEV_Release(HANDLE hDevice)
```

功能：释放设备对象（Release device object）， 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 DEV_Create()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： DEV_Create()

3.2 DIO 数字量输入输出函数原型说明

DIO_GetParam()

函数原型：

Visual C++ :

```
BOOL DIO_GetParam(HANDLE hDevice, U32 nPort, PUSB5561_DIO_PARAM pDIOParam);
```

功能：初始化指定端口的工作参数 (Initialize task parameter for digital input)。

参数：

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

nPort 端口号，取值范围[0,1]，Port0 为单向 DI，Port1 为单向 DO。

PUSB5561_DIO_PARAM 出口参数，DIO 工作参数结构体，详细定义参考《第四节、DIO_PARAM (DIO 数字量工作参数结构体)》。

返回值：如果成功，返回 TRUE，否则返回 FALSE。

相关函数： DEV_Create() DIO_InitTask() DIO_ReadPort() DIO_WritePort() DIO_ReadLines() DIO_WriteLines() DIO_ReadLine() DIO_WriteLine() DIO_ReleaseTask() DEV_Release()

DIO_InitTask()

函数原型：

Visual C++ :

BOOL DIO_InitTask(HANDLE hDevice, U32 nPort, PUSB5561_DIO_PARAM pDIOSts);

功能：初始化指定 DIO 端口的工作参数 (Initialize task parameter for digital input)。

参数：

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

nPort 端口号，取值范围[0,1]，Port0 为单向 DI，Port1 为单向 DO。

PUSB5561_DIO_PARAM pDIOSts 入口参数，DIO 端口工作参数结构体，详细定义参考《第四节、DIO_PARAM (DIO 数字量工作参数结构体)》。

返回值：如果成功，返回 TRUE，否则返回 FALSE。

相关函数： DEV_Create() DIO_GetParam() DIO_ReadPort() DIO_WritePort() DIO_ReadLines() DIO_WriteLines() DIO_ReadLine() DIO_WriteLine() DIO_ReleaseTask() DEV_Release()

DIO_ReadPort()

函数原型：

Visual C++ :

BOOL DIO_ReadPort (HANDLE hDevice, U32 nPort, U32* pPortData);

功能：读取 DIO 的端口数据 (Read port data for digital input or output)。

参数：

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

nPort 入口参数，端口号，本设备仅支持两个端口，故次此参数取值范围为[0,1]，0 对应单向 DI，1 对应单向 DO。

pPortData 返回的端口数据, Port0 对应有效位 Bit[7:0], Port1 对应有效位 Bit[5:0] 。

返回值：如果成功，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： DEV_Create() DIO_GetParam() DIO_InitTask() DIO_WritePort() DIO_ReadLines() DIO_WriteLines() DIO_ReadLine() DIO_WriteLine() DIO_ReleaseTask() DEV_Release()

DIO_WritePort()

函数原型：

Visual C++ :

BOOL DIO_WritePort (HANDLE hDevice, U32 nPort, U32 nPortData);

功能：写入 DO 端口数据 (Write port data for digital input or output)。

参数：

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

nPort 入口参数，端口号，本设备只有一个 DO 端口对应 Port1，故次此参数取值恒为 1。

nPortData 端口数据, Port1 对应有效位 Bit[5:0]。

返回值：如果成功，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： DEV_Create() DIO_GetParam() DIO_InitTask() DIO_ReadPort() DIO_ReadLines() DIO_WriteLines() DIO_ReadLine() DIO_WriteLine() DIO_ReleaseTask() DEV_Release()

DIO_ReadLines()

函数原型：

Visual C++ :

BOOL DIO_ReadLines (HANDLE hDevice, U32 nPort, U32 bLineDataArray[8]);

功能：读取 DIO 的多线数据 (Read lines data for digital input or output)。

参数：

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

nPort 入口参数，端口号，本设备仅支持两个端口，故次此参数取值范围为[0,1]，0 对应单向 DI，1 对应单向 DO。

bLineDataArray 出口参数，同时返回端口中各线的状态值 bLineDataArray[n]=0:表示关(或低)状态, =1 表示开(或高)状态。

返回值：如果成功，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： DEV_Create() DIO_GetParam() DIO_InitTask() DIO_ReadPort() DIO_WritePort() DIO_WriteLines() DIO_ReadLine() DIO_WriteLine() DIO_ReleaseTask() DEV_Release()

DIO_WriteLines()

函数原型：

Visual C++:

BOOL DIO_WriteLines(HANDLE hDevice, U32 nPort, U32 bLineDataArray[8]);

功能：写入 DO 的多线数据 (Write lines data for digital output)。

参数：

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

nPort 入口参数，端口号，本设备只有一个 DO 端口对应 Port1，故次此参数取值恒为 1。

bLineDataArray 出口参数，端口中各线的状态值 bLineDataArray[n]=0:表示关(或低)状态, =1 表示开(或高)状态。

返回值：如果成功，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： DEV_Create() DIO_GetParam() DIO_InitTask()
DIO_ReadPort() DIO_WritePort() DIO_ReadLines()
DIO_ReadLine() DIO_WriteLine() DIO_ReleaseTask()
DEV_Release()

DIO_ReadLine()

函数原型：

Visual C++ :

BOOL DIO_ReadLine (HANDLE hDevice, U32 nPort, U32 nLine, U32* pLineData);

功能：读取 DIO 的单线数据 (Read line data for digital input)。

参数：

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

nPort 入口参数，端口号，本设备仅支持两个端口，故次此参数取值范围为[0,1]，0 对应单向 DI，1 对应单向 DO。

nLine 入口参数，指定端口中的线号。线号, Port0 取值范围[0, 7], Port1 取值范围[0, 5]

pLineData 出口参数，从指定端口中指定线号上读入的线数据，线数据只有两种取值：0（低）或 1（高）。

返回值：如果成功，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： DEV_Create() DIO_GetParam() DIO_InitTask()
DIO_ReadPort() DIO_WritePort() DIO_ReadLines()
DIO_WriteLines() DIO_WriteLine() DIO_ReleaseTask()
DEV_Release()

DIO_WriteLine()

函数原型：

Visual C++:

BOOL DIO_WriteLine(HANDLE hDevice, U32 nPort, U32 nLine, U32 bLineData);

功能：写入 DO 的单线数据 (Write line data for digital output)。

参数：

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

nPort 入口参数，端口号，本设备只有一个 DO 端口对应 Port1，故次此参数取值恒为 1。

nLine 入口参数，指定端口中的线号。线号, 取值范围[0, 7]

pLineData 出口参数，从指定端口中指定线号上读入的线数据，线数据只有两种取值：0（低）或 1（高）。

返回值：如果成功，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： DEV_Create() DIO_GetParam() DIO_InitTask()
DIO_ReadPort() DIO_WritePort() DIO_ReadLines()
DIO_WriteLines() DIO_ReadLine() DIO_ReleaseTask()
DEV_Release()

DIO_ReleaseTask()

函数原型：

Visual C++ :

BOOL DIO_ReleaseTask (HANDLE hDevice, U32 nPort);

功能：释放 DIO 任务。

参数：

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

nPort 入口参数，端口号，端口号, 取值范围为[0, 1], 保持与初始化函数中端口号一致。

返回值：如果成功，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： DEV_Create() DIO_GetParam() DIO_InitTask() DIO_ReadPort() DIO_WritePort() DIO_ReadLines() DIO_WriteLines() DIO_ReadLine() DIO_WriteLine() DEV_Release()

DIO 控制流程

(1) DEV_Create() 创建设备句柄

(2) DIO_GetParam() 获取 DIO 对应参数（根据需要执行此步骤，此步骤可省略）

(3) DIO_InitTask() 根据需要设置 DIO 参数

(4) DIO_ReadPort()或 DIO_WritePort()、DIO_ReadLines()、DIO_WriteLines()、DIO_ReadLine()、DIO_WriteLine()实时读写 DI、DO 端口或线数据

(5) DEV_Release() 释放设备句柄

可以反复执行第(4)步，以对数字量输入输出进行操作。

3.3 INT 中断函数原型说明

INT_GetParam()

函数原型：

Visual C++ :

BOOL INT_GetParam(HANDLE hDevice, U32 nIntSource, PUSB5561_INT_PARAM pIntParam);

功能：获取中断相关参数 (Get INT Param)。

参数：

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

nIntSource 入口参数， 中断源, 取值范围[0,7]。

pINTParam 出口参数，中断参数结构体，具体定义参见《4.2 INT_PARAM（INT 中断工作参数结构体）》。

返回值：如果成功，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： DEV_Create() INT_InitTask() INT_StartTask() INT_ReadData() INT_StopTask() INT_ReleaseTask() DEV_Release()

INT_InitTask()

函数原型：

Visual C++:

BOOL INT_InitTask(HANDLE hDevice,
USB5561_INT_PARAM pIntParam,
HANDLE* pIntEvent);

功能：初始化中断任务。使用此函数一方面配置中断工作参数，另一方面创建中断事件，当板卡上传中断数据包时，此事件会被触发，上位机接收到中断事件后即可调用读中断数据函数来获取中断数据包信息。操作流程参见 INT 控制流程。

参数：

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

pINTParam 出口参数，中断参数结构体，具体定义参见《4.2 INT_PARAM（INT 中断工作参数结构体）》。

pIntEvent 出口参数，返回中断事件对象句柄，当设备接收到中断信号时，此中断事件会被触发。

返回值：如果成功，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： DEV_Create() INT_GetParam() INT_StartTask()
INT_ReadData() INT_StopTask() INT_ReleaseTask()
DEV_Release()

INT_StartTask()

函数原型：

Visual C++:

BOOL INT_StartTask(HANDLE hDevice, U32 nIntSource);

功能：开始中断任务。

参数：

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

nIntSource 入口参数，中断源，取值范围[0,7]，务必与初始化任务函数中的中断参数结构体中的中断源参数保持一致。

返回值：如果成功，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： DEV_Create() INT_GetParam() INT_InitTask()
INT_ReadData() INT_StopTask() INT_ReleaseTask()
DEV_Release()

INT_ReadData()

函数原型：

Visual C++:

BOOL INT_ReadData(HANDLE hDevice, U32* nReadCount, USB5561_INT_INFO* pIntInfo);

功能：初始化中断任务。

参数：

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

nReadCount 出口参数，本次读到的中断个数。

pIntInfo 出口参数，中断数据包数组，有效元素个数等于上一个参数 nReadCount 的返回值，所以在读取中断数据包时请根据返回的 **nReadCount** 参数来确定此次返回的中断数据包个数。中断数据包结构体详细定义参见《4.3 INT_INFO（INT 中断数据包参数结构体）》。

返回值：如果成功，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： DEV_Create() INT_GetParam() INT_InitTask() INT_StartTask() INT_StopTask() INT_ReleaseTask() DEV_Release()

INT_StopTask()

函数原型：

Visual C++ :

BOOL INT_StopTask(HANDLE hDevice, U32 nIntSource);

功能：停止中断任务。

参数：

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

nIntSource 入口参数，中断源，取值范围[0,7]，务必与开始中断任务函数中的中断源参数保持一致。

返回值：如果成功，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： DEV_Create() INT_GetParam() INT_InitTask() INT_ReadData() INT_StartTask() INT_ReleaseTask() DEV_Release()

INT_ReleaseTask()

函数原型：

Visual C++ :

BOOL INT_StopTask(HANDLE hDevice, U32 nIntSource);

功能：释放中断任务。

参数：

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

nIntSource 入口参数，中断源，取值范围[0,7]，务必与开始中断任务函数中的中断源参数保持一致。

返回值：如果成功，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： DEV_Create() INT_GetParam() INT_InitTask() INT_ReadData() INT_StartTask() INT_StopTask() DEV_Release()

INT 控制流程

(1) DEV_Create() 创建设备句柄；

(2) INT_GetParam() 获取 INT 对应参数（根据需要执行此步骤，此步骤可省略）；

(3) INT_InitTask() 根据需要设置 INT 工作参数并且创建中断事件等；

(4) INT_ReadData() 实时读 INT 数据。可在等待 INT_InitTask()函数中的中断事件触发时读取中断数据，也可定时调用此函数读取中断数据，用户根据需要编程实现；

(5) DEV_Release() 释放设备句柄。

可以反复执行第(4)步，以读取中断数据操作。

4 各种结构体描述

4.1 DIO_PARAM（DIO 数字量工作参数结构体）

Visual C++ :

```
typedef struct _USB5561_DIO_PARAM
{
    BOOL bDIOStatus[8];
} USB5561_DIO_PARAM, *PUSB5561_DIO_PARAM;
```

bDIOStatus

当此结构体对应 DI 功能时，bDIOStatus[0, 7]分别表示 DI0~15 输入是否反向，0=正常，1=反向。

当此结构体对应 DO 功能时，bDIOStatus[0, 5]分别表示 DO0~5 的初始状态值，0=低电平（关），1=高电平（开）。

相关函数： DEV_Create() DIO_GetParam() DIO_InitTask() DEV_Release()

4.2 INT_PARAM（INT 中断工作参数结构体）

Visual C++ :

```
typedef struct _USB5561_INT_PARAM
{
    U32 nIntSource;
    U32 nIntEdge;
} USB5561_INT_PARAM, *PUSB5561_INT_PARAM;
```

nIntSource

中断源参数。取值范围[0:7]：0=DI0 1=DI1 2=DI2 3=DI3 4=DI4 5=DI5 6=DI6 7=DI7。

nIntEdge

中断边沿参数。取值范围[0:2]：0=下降沿触发，1=上升沿触发，2=变化即触发（上升沿下降沿都触发）。

相关函数： DEV_Create() INT_GetParam() INT_InitTask() DEV_Release()

4.3 INT_INFO（INT 中断数据包参数结构体）

Visual C++ :

```
typedef struct _USB5561_INT_INFO
{
    U32 nIntSource;
    U32 nDIStatus;
    U32 nIntCount;
} USB5561_INT_INFO, *PUSB5561_INT_INFO;
```

nIntSource

中断源参数。 (0=DI0 1=DI1 2=DI2 3=DI3 4=DI4 5=DI5 6=DI6 7=DI7)。

nDIStatus

DI 状态参数。该参数表示接收到中断时，板卡捕获的 DI 状态，有效位为[0:15]，分别对应 DI0~DI15，0 表示低电平，1 表示高电平。

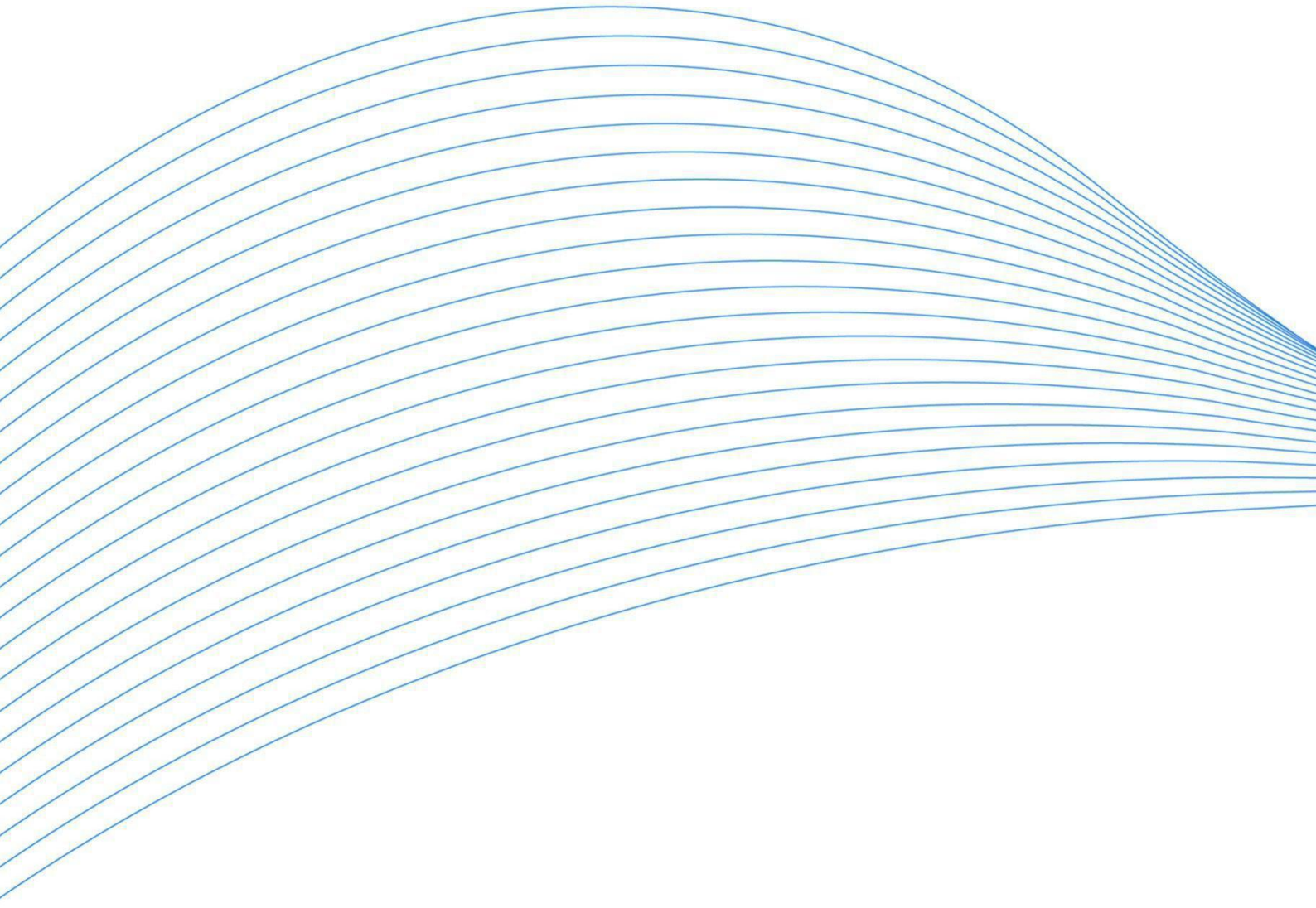
nIntCount

中断序号参数。表示使能中断之后，板卡接收到的中断次数。

相关函数： DEV_Create() INT_InitTask() DEV_Release()

5 修改历史

修改时间	版本号	修改内容
2017.7.18	V6.03.00	第一版



北京阿尔泰科技发展有限公司
服务热线：400-860-3335
邮编：100086
传真：010-62901157