NET2991/2991A/2991B
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

V6.00.00

阿尔泰科技
www.art-control.com

关于本手册

本手册为阿尔泰科技推出的 NET2991/NET2991A/NET2991B 数据采集卡驱动程序使用手册，其中包括版权信息与命名约定、使用纲要、各功能操作流程介绍、设备操作函数接口介绍、硬件参数结构、数据格式转换与排列规则、上层用户函数接口应用实例、共用函数介绍、修改历史等。

文档版本：V6.00.00

目　录

1 版权信息与命名约定

1.1 版权信息

1.2 命名约定

为简化文字内容，突出重点，本文中提到的函数名通常为基本功能名部分，其前缀设备名如NETxxxx_则被省略。如 NET2991_CreateDevice 则写为 CreateDevice。如果是 NET2991A 或 NET2991B 相应的地方置换为相应的名称即可，如 NET2991_CreateDevice，在 NET2991A 上即为 NET2991A_CreateDevice。

表 1-2-1：函数名及参数中各种关键字缩写

缩写	全称	汉语意思	缩写	全称	汉语意思
Dev	Device	设备	DI	Digital Input	数字量输入
Pro	Program	程序	DO	Digital Output	数字量输出
Int	Interrupt	中断	CNT	Counter	计数器
Dma	Direct Memory Access	直接内存存取	DA	Digital convert to Analog	数模转换
AD	Analog convert to Digital	模数转换	DI	Differential	(双端或差分) 注：在常量选项中
Npt	Not Empty	非空	SE	Single end	单端
Para	Parameter	参数	DIR	Direction	方向
SRC	Source	源	ATR	Analog Trigger	模拟量触发
TRIG	Trigger	触发	DTR	Digital Trigger	数字量触发
CLK	Clock	时钟	Cur	Current	当前的
GND	Ground	地	OPT	Operate	操作
Lgc	Logical	逻辑的	ID	Identifier	标识
Phys	Physical	物理的			

以上规则不局限于该产品。

1.3 数据类型

表 1-3-1：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	LongInt
U32	无符号 32 位整型数	0 to 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 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

表 1-3-2：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

表 1-3-3：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

2 使用纲要

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，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。而底层用户则是需了解硬件知识和控制细节、且又需要特殊复杂控制的用户。但不管怎样，我们强烈建议您使用上层函数（在这些函数中，您见不到任何设备地址、寄存器端口、中断号等物理信息，其复杂的控制细节完全封装在上层用户函数中。）对于上层用户函数的使用，您基本上不必参考硬件说明书，除非您需要知道板上连接器管脚分配情况。

2.2 如何管理设备

由于我们的驱动程序采用面向对象编程，所以要使用设备的一切功能，则必须首先用 DEV_Create 函数创建一个设备句柄 hDevice，有了这个句柄，您就拥有了对该设备的绝对控制权。然后将此句柄作为参数传递给相应的驱动函数。最后可以通过 DEV_Release 将 hDevice 释放掉。

2.3 如何取得 AI 数据

当您有了 hDevice 设备句柄后，便可用 AI_InitTask 函数初始化 AI 部件，关于采样通道、频率等参数的设置是由这个函数的 pAIParam 参数结构体决定的。您只需要对这个 pAIParam 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 AI_StartTask 即可启动 AD 部件，开始 AI 采样，然后便可用 AI_ReadBinary 反复读取 AI 数据以实现连续不间断采样。当您需要暂停设备时，执行 AI_StopTask，当您需要关闭 AI 设备时，DEV_Release 便可帮您实现(注：此时 hDevice 将无效)。

另外，为了高速有效的保存文件，本程序把保存文件封装在动态库里，您只要利用 FILE_Create 函数创建文件路径名即可。

具体流程请参考《各功能操作流程介绍》章节。

2.4 哪些函数对您不是必须的

公共函数一般来说都是辅助性函数，这些函数您可完全不必理会，除非您是作为底层用户管理设备。公共函数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、Win7 等操作系统中实现对您原有传统设备如 ISA 卡、串口卡、并口卡的访问，而没有这些函数，您可能在基于 Windows NT 架构的操作系统中无法继续使用您原有的老设备。

2.5 系统硬件要求

系统：Win7 32/64bit

CPU：双核以上

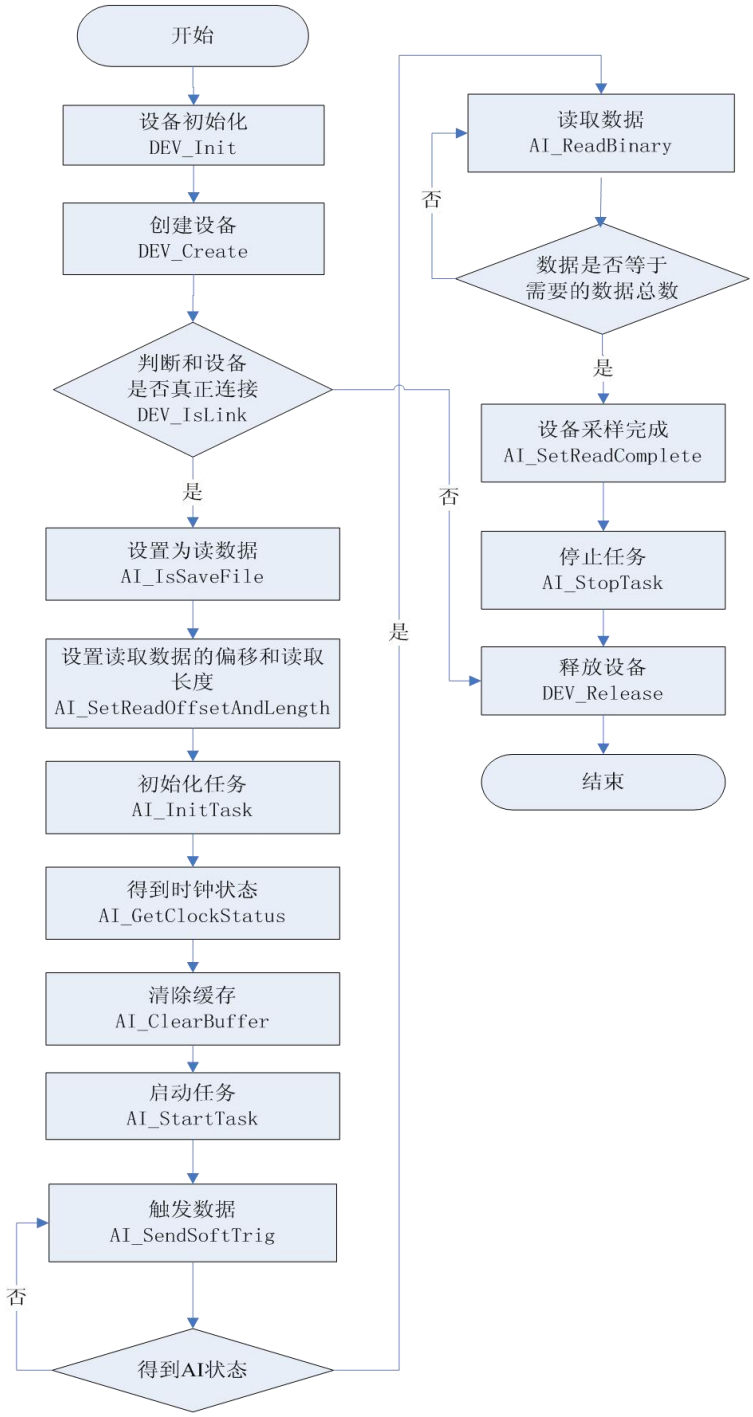
内存：3G 以上

硬盘：请使用固态及性能相当或以上硬盘，写入速度大于 200MB/s

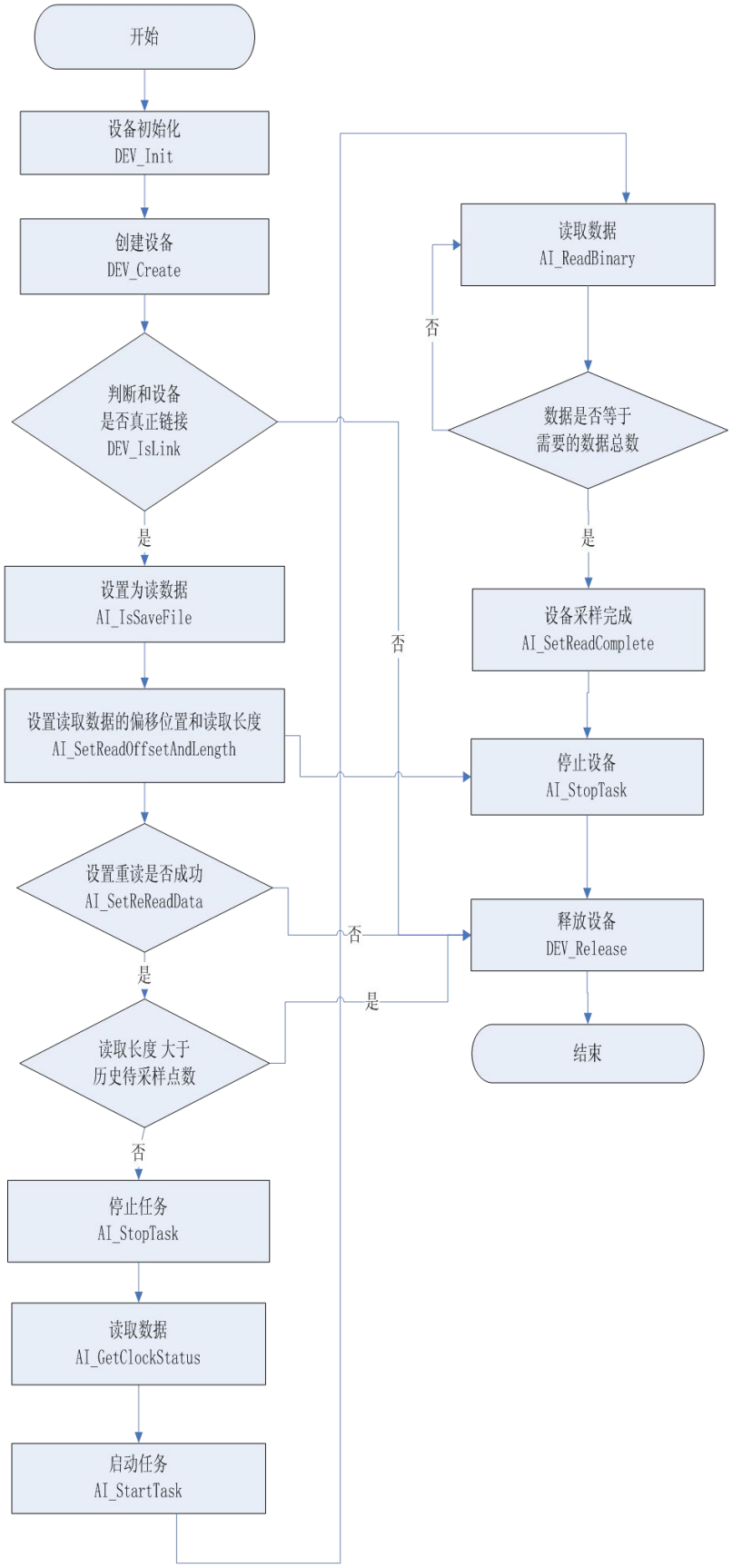
3 各功能操作流程介绍

注意：有限采集下，启动任务 AI_Start 必须单独设置在一个循序中启动任务。

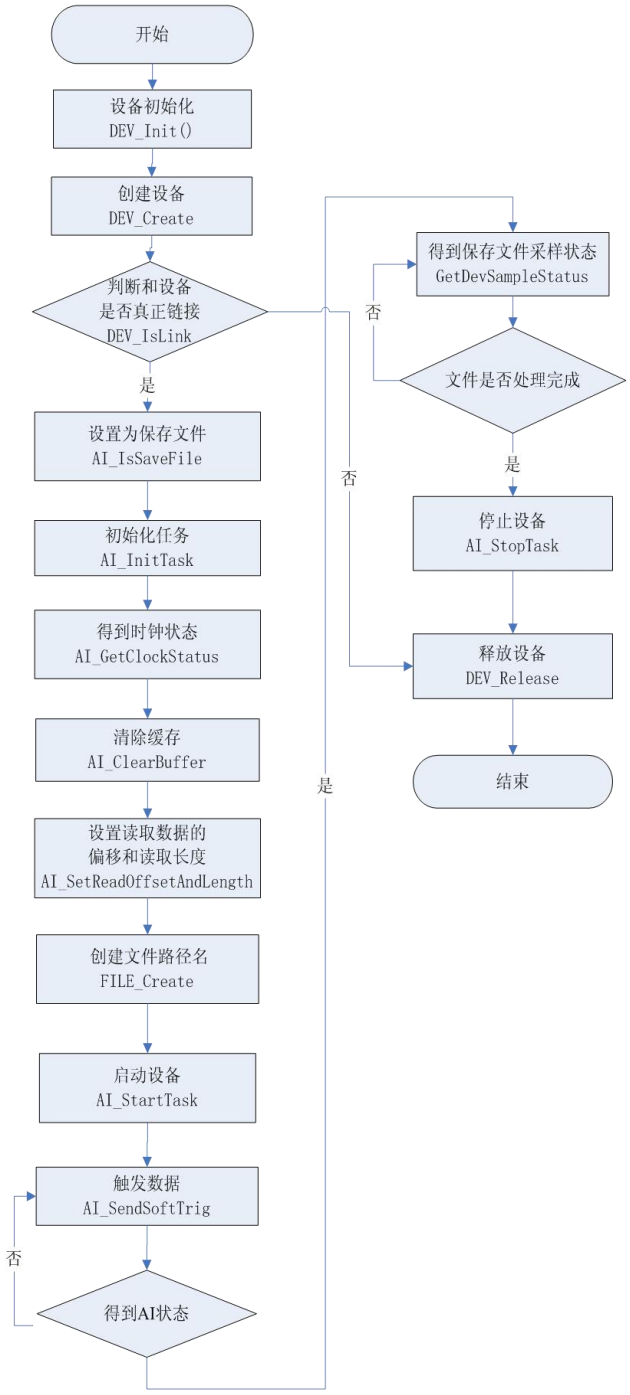
3.1 AI 有限采集模式读数据操作流程



3.2 AI 有限采集模式重读数据操作流程



3.3 AI 有限采集模式保存文件操作流程

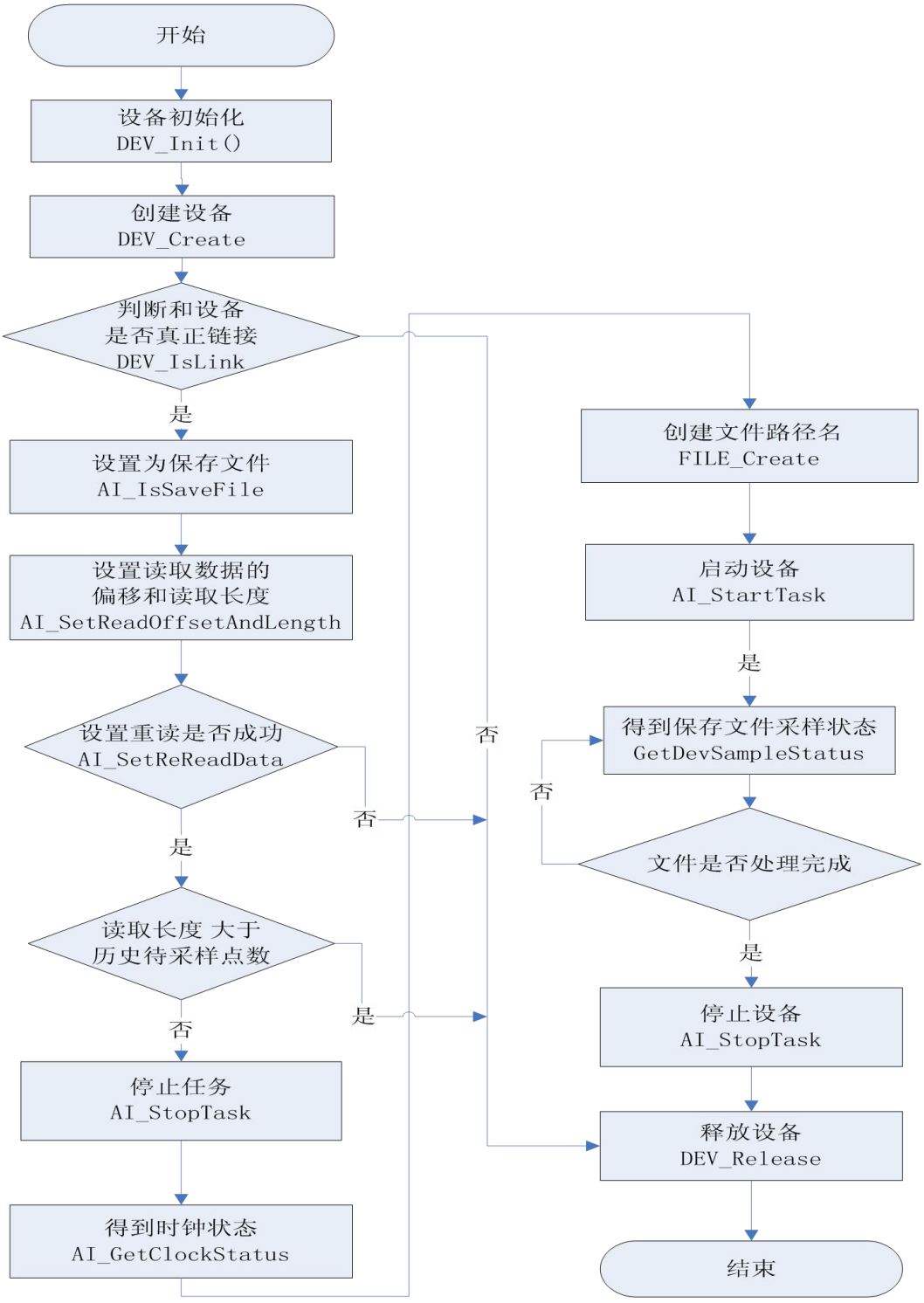


注：

1、 为了提高文件保存的效率，文件保存只需要用户利用 FILE_Create 创建文件路径及文件名，其它的操作都在 dll 库里完成。

2、 在得到状态失败或强制停止后请使用 FILE_Close 函数关闭已经创建的文件。

3.4 AI 有限采集模式重读保存文件操作流程

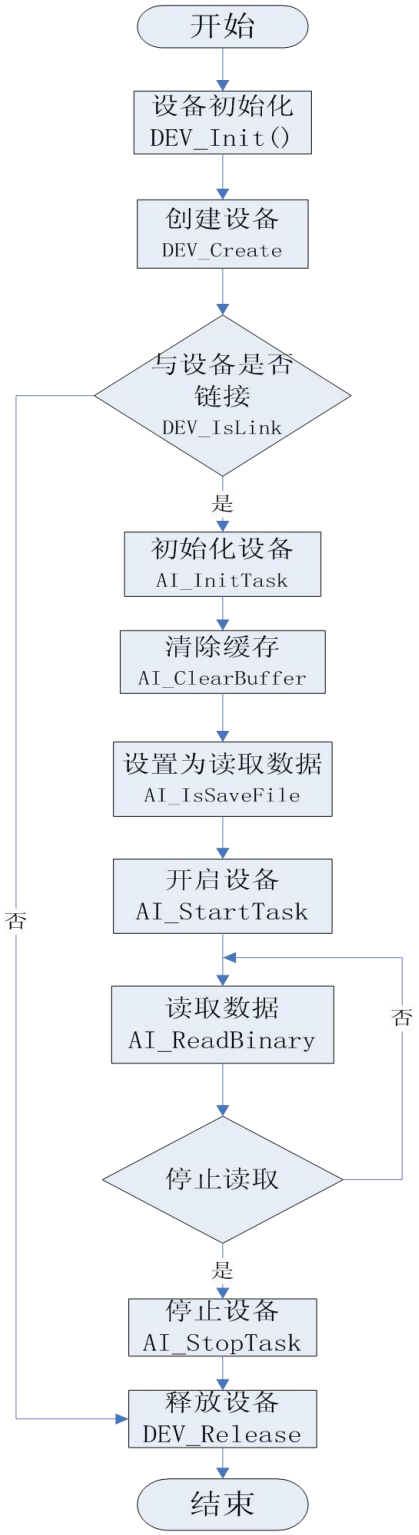


注：

1、为了提高文件保存的效率，文件保存只需要用户利用 FILE_Create 创建文件路径及文件名，其它的操作都在 dll 库里完成。

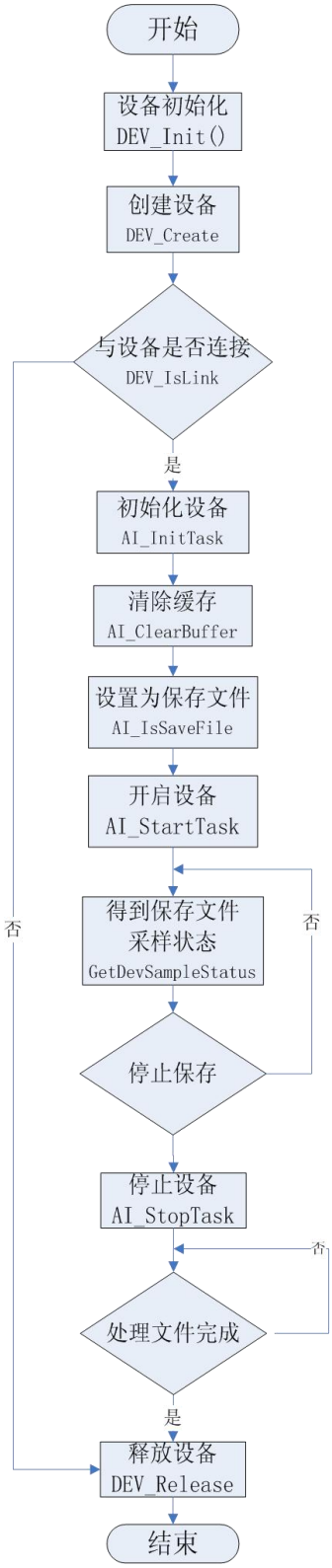
2、在得到状态失败或强制停止后请使用 FILE_Close 函数关闭已经创建的文件。

3.5 AI 连续采集模式读数据操作流程



注意：连续模式下，根据 UDP 的特性，以及考虑便于处理数据，设置每次读取数据的字节长度为通道数量的整数倍，但不超过 1280 字节。

3.6 AI 连续采集模式保存文件操作流程



注：为了提高文件保存的效率，文件保存只需要用户利用 FILE_Create 创建文件路径及文件名，其它的操作都在 dll 库里完成。

4 设备操作函数接口介绍

由于我公司的设备应用于各种不同的领域，有些用户可能根本不关心硬件设备的控制细节，只关心首末通道、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数据。这方面的用户我们称之为上层用户。那么还有一部分用户不仅对硬件控制熟悉，而且由于应用对象的特殊要求，则要直接控制设备的每一个端口，这是一种复杂的工作，但又是必须的工作，我们则把这一群用户称之为底层用户。因此总的看来，上层用户要求简单、快捷，他们最希望在软件操作上所要面对的全是他们最关心的问题，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总线复杂的控制协议，同是还可以省掉您许多繁琐的工作。

综上所述，用户使用我公司提供的驱动程序软件包将极大的方便和满足您的各种需求。但为了您更省心，别忘了在您正式阅读下面的函数说明时，先明白自己是上层用户还是底层用户。

4.1 设备驱动接口函数总列表

表 4-1-1：驱动接口函数总列表（每个函数省略了前缀“NET2991_”）

函数名	函数功能	备注
① 操作设备函数		
DEV_Init	对设备采集做一些初始工作	上层用户
DEV_Create	创建设备对象	上层及底层用户
DEV_Release	关闭设备,禁止传输,且释放资源	上层及底层用户
DEV_IsLink	判读设备是否真正连接	上层用户
② 设备网络参数		
DEV_SetNetInfo	设置网络参数	上层用户
DEV_GetNetInfo	得到网络参数	上层用户
③ AI 模拟量输入实现函数		
AI_InitTask	初始化 AI 采集任务	上层用户
AI_ClearBuffer	清除 AI 硬件缓存	上层用户
AI_StartTask	启动采集任务	上层用户
AI_SendSoftTrig	发送软件触发事件	上层用户
AI_GetStatus	取得 AI 各种状态	上层用户
AI_GetClockStatus	得到时钟状态,外部 10M 时钟时有效	上层用户
AI_SetReadOffsetAndLength	设置读取数据的偏移位置和读取长度(有限时模式下使用)	上层用户
AI_ReadBinary	读取采样数据(二进制原码数据序列)	上层用户
AI_SetReadComplete	某设备采样完成	上层用户
AI_StopTask	停止采集任务	上层用户

AI_IsSaveFile	是否保存文件	上层用户
AI_SetReReadData	设置是否重新读取	上层用户
GetDevSampleStatus	得到 dll 里设备采样的状态	上层用户

Visual C++：

① 要使用如下函数关键的问题是必须在您的源程序中包含如下语句：#include “C:\Art\NET2991\INCLUDE\NET2991.H”（ 采用默认路径和默认板号），用户需根据自己的板号和安装情况确定 NET2991.H 文件的正确路径。

② 用户也可以把此文件拷到您的源程序目录中，然后加入如下语句：#include “NET2991.H”

Visual Basic:

要使用如下函数一个关键的问题是首先必须将我们提供的模块文件(*.Bas)加入到您的 VB 工程中。其方法是选择 VB 编程环境中的工程(Project)菜单，执行其中的"添加模块"(Add Module)命令，在弹出的对话中选择 NET2991.Bas 模块文件，该文件的路径为用户安装驱动程序后其子目录 Samples\VB 下面。

请注意，因考虑 Visual C++和 Visual Basic 两种语言的兼容问题，在下列函数说明和示范程序中，所举的 Visual Basic 程序均是需要编译后在独立环境中运行。所以用户若在解释环境中运行这些代码，我们不能保证完全顺利运行。

LabVIEW/CVI :

LabVIEW 是美国国家仪器公司(National Instrument)推出的一种基于图形开发、调试和运行程序的集成化环境，是目前国际上唯一的编译型的图形化编程语言。在以 PC 机为基础的测量和工控软件中，LabVIEW 的市场普及率仅次于 C++/C 语言。LabVIEW 开发环境具有一系列优点，从其流程图式的编程、不需预先编译就存在的语法检查、调试过程使用的数据探针，到其丰富的函数功能、数值分析、信号处理和设备驱动等功能，都令人称道。关于 LabView/CVI 的进一步介绍请见本文最后一部分关于 LabView 的专述。其驱动程序接口单元模块的使用方法如下：

（1）、在 LabView 中打开 NET2991.VI 文件，用鼠标单击接口单元图标，比如 DEV_Create 图标

然后按 Ctrl+C 或选择 LabView 菜单 Edit 中的 Copy 命令，接着进入用户的应用程序 LabView 中，按 Ctrl+V 或选择 LabView 菜单 Edit 中的 Paste 命令，即可将接口单元加入到用户工程中，然后按以下函数原型说明或演示程序的说明连接该接口模块即可顺利使用。

（2）、根据 LabView 语言本身的规定，接口单元图标以黑色的较粗的中间线为中心，以左边的方格为数据输入端，右边的方格为数据的输出端，设备对象句柄、用户分配的数据缓冲区、要求采集的数据长度等信息从接口单元左边输入端进入单元，待单元接口被执行后，需要返回给用户的数据从接口单元右边的输出端输出，其他接口完全同理。

（3）、在单元接口图标中，凡标有“I32”为有符号长整型 32 位数据类型，“U16”为无符号短整型 16 位数据类型，“[U16]”为无符号 16 位短整型数组或缓冲区或指针，“[U32]”与“[U16]”同理，只是位数不一样。

4.2 操作设备函数原型说明

♦ 设备初始化

函数原型：

Visual C++：

```
BOOL _DEV_Init();
```

Visual Basic:

```
Declare Function _DEV_Init    Lib “NET2991”();
```

功能：对设备采样做一些初始工作

参数：无

返回值：若成功，则返回 TRUE，否则返回 FALSE。

♦ 创建设备对象

函数原型：

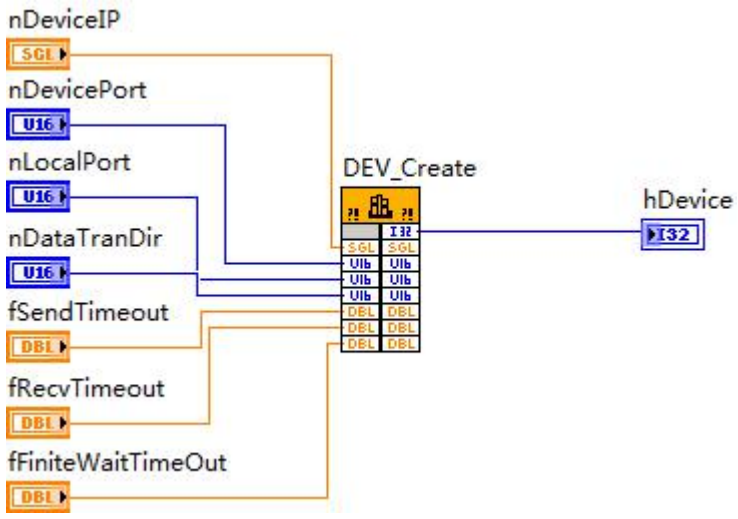
Visual C++：

```
HANDLE _DEV_Create( U32 nDeviceIP,
                    U16 nDevicePort,
                    U16 nLocalPort,
                    U16 nDataTranDir,
                    F64 fSendTimeout,
                    F64 fRecvTimeout,
                    F64 fFiniteWaitTimeOut);
```

Visual Basic:

```
Declare Function _DEV_Create Lib “NET2991”(ByVal svName As Single,
                    ByVal nDevicePort As Long,
                    ByVal nLocalPort As Long,
                    ByVal nDataTranDir As Long,
                    ByVal fSendTimeout As Double,
                    ByVal fRecvTimeout As Double,
                    ByVal fFiniteWaitTimeOut As Double) As Long;
```

Labview:



功能：创建设备对象(Create device object)，并返回其设备句柄 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：

nDeviceIP　设备 IP。

nDevicePort　设备端口号。

nLocalPort　本地端口号。

nDataTranDir　数据传输方向 0:客户端方向,1:服务器方向。

fSendTimeout　发送超时。

fRecvTimeout　接收超时。

fFiniteWaitTimeOut 有限模式下数据等待超时

返回值：如果执行成功，则返回设备句柄；如果没有成功，则返回错误码。

相关参数： DEV_Release　DEV_IsLink　AI_InitTask
AI_ClearBuffer　AI_StartTask　AI_SendSoftTrig　AI_GetStatus
AI_GetClockStatus　AI_SetReadOffsetAndLength　AI_ReadBinary
AI_StopTask　AI_IsSaveFile　AI_SetReReadData
DEV_SetNetInfo　DEV_GetNetInfo　AI_SetReadComplete

♦ 释放设备对象

函数原型：

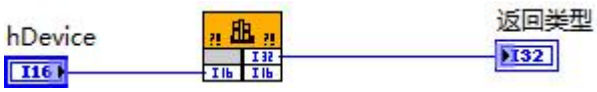
Visual C++:

BOOL DEV_Release(HANDLE hDevice);

Visual Basic:

Declare Function ReleaseDevice Lib “NET2991” (ByVal hDevice As Long) As Boolean;

LabView:



功能：关闭设备,禁止传输,且释放资源。

参数：

hDevice　设备对象句柄,它由 DEV_Create()函数创建。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release　DEV_IsLink　AI_InitTask
AI_ClearBuffer　AI_StartTask　AI_SendSoftTrig　AI_GetStatus
AI_GetClockStatus　AI_SetReadOffsetAndLength　AI_ReadBinary
AI_StopTask　AI_IsSaveFile　AI_SetReReadData
DEV_SetNetInfo　DEV_GetNetInfo　AI_SetReadComplete

♦ 判断与设备连接

函数原型：

Visual C++:

BOOL DEV_IsLink(HANDLE hDevice);

Visual Basic:

Declare Function DEV_IsLink Lib “NET2991” (ByVal hDevice As Long) As Boolean;

LabView:

请参考相关演示程序。

功能：判断是否和设备真正连接。

参数：

hDevice　设备对象句柄,它由 DEV_Create()函数创建。

返回值：若初始化设备对象成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release　DEV_IsLink　AI_InitTask
AI_ClearBuffer　AI_StartTask　AI_SendSoftTrig　AI_GetStatus
AI_GetClockStatus　AI_SetReadOffsetAndLength　AI_ReadBinary
AI_StopTask　AI_IsSaveFile　AI_SetReReadData
DEV_SetNetInfo　DEV_GetNetInfo　AI_SetReadComplete

4.3　设备网络参数原型说明

♦ 设置网络参数

函数原型：

Visual C++:

```
BOOL DEV_SetNetInfo(HANDLE hDevice,
                    PDEVICE_NET_INFO pNetInfo);
```

Visual Basic:

```
Declare Function DEV_SetNetInfo Lib “NET2991” ( ByVal hDevice As Long,
                    ByRef pNetInfoAs PDEVICE_NET_INFO) As Boolean;
```

LabView:

请参考相关演示程序。

功能：设置网络参数。

参数：

hDevice　设备对象句柄,它由 DEV_Create()函数创建。

pNetInfo　网络参数结构体，详细介绍请参考 NET2991.h 或 NET2991.Bas 或 NET2991.Pas 函数原型定义文件或《硬件参数结构》章节。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release　DEV_IsLink　AI_InitTask
AI_ClearBuffer　AI_StartTask　AI_SendSoftTrig　AI_GetStatus
AI_GetClockStatus　AI_SetReadOffsetAndLength　AI_ReadBinary
AI_StopTask　AI_IsSaveFile　AI_SetReReadData
DEV_SetNetInfo　DEV_GetNetInfo　AI_SetReadComplete

♦ 得到网络参数

函数原型：

Visual C++:

```
BOOL DEV_GetNetInfo(HANDLE hDevice,
                    PDEVICE_NET_INFO pNetInfo);
```

Visual Basic:

```
Declare Function GetNetCfg Lib “NET2991” ( ByVal hDevice As Long,
```

ByRef pNetInfo As PDEVICE_NET_INFO) As Boolean;

LabView:

请参考相关演示程序。

功能：得到网络参数，其中第二个参数是结构体，里面有网络参数信息。

参数：

hDevicve　设备对象句柄,它由 DEV_Create()函数创建。

pNetInfo　网络参数结构体，详细介绍请参考 NET2991.h 或 NET2991.Bas 或 NET2991.Pas 函数原型定义文件或《硬件参数结构》章节。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release　DEV_IsLink　AI_InitTask

AI_ClearBuffer　AI_StartTask　AI_SendSoftTrig　AI_GetStatus

AI_GetClockStatus　AI_SetReadOffsetAndLength　AI_ReadBinary

AI_StopTask　AI_IsSaveFile　AI_SetReReadData

DEV_SetNetInfo　DEV_GetNetInfo　AI_SetReadComplete

4.4 AI 模拟量输入实现函数原型说明

♦ 初始化设备

函数原型：

Visual C++:

```
BOOL AI_InitTask (HANDLE hDevice,
                  PNET2991_AI_PARAM pAIParam,
                  HANDLE* pSampEvent);
```

Visual Basic:

```
Declare Function InitDeviceAD Lib "NET2991" (ByVal hDevice As Long,
                                             ByRef pAIParam As PNET2991_AI_PARAM,
                                             ByRef pSampEvent As Long) As Boolean;
```

LabView:

请参考相关演示程序。

功能：初始化设备。它负责初始化设备对象中的 AD 部件，为设备的操作就绪做有关准备工作。

参数：

hDevice　设备对象句柄,它由 DEV_Create()函数创建。

pAIParam　AI 工作参数，它仅在此函数中决定硬件初始状态和各工作模式。详细介绍请参考 NET2991.h 或 NET2991.Bas 或 NET2991.Pas 函数原型定义文件或《硬件参数结构》章节。

pSampEvent　返回采样事件对象句柄,当设备中出现可读数据段时会触发此事件，参数=NULL,表示不需要此事件句柄。

返回值：若初始化设备对象成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release　DEV_IsLink　AI_InitTask
AI_ClearBuffer　AI_StartTask　AI_SendSoftTrig　AI_GetStatus
AI_GetClockStatus　AI_SetReadOffsetAndLength　AI_ReadBinary
AI_StopTask　AI_IsSaveFile　AI_SetReReadData
DEV_SetNetInfo　DEV_GetNetInfo　AI_SetReadComplete

♦ 启动设备

函数原型：

Visual C++:

```
BOOL AI_StartTask (HANDLE hDevice);
```

Visual Basic:

```
Declare Function AI_StartTask Lib "NET2991" (ByVal hDevice As Long) As Boolean;
```

LabView:

请参考相关演示程序。

功能：启动采集任务。它必须在调用 AI_InitTask 后才能调用此函数。该函数除了启动 AI 设备开始转换以外，不改变设备的其他任何状态。

多设备调用时，请初始化设备后单独循环此函数启动采集任务。

参数：

hDevice　设备对象句柄,它由 DEV_Create()函数创建。

返回值：若调用成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release　DEV_IsLink　AI_InitTask

AI_ClearBuffer AI_StartTask AI_SendSoftTrig AI_GetStatus
AI_GetClockStatus AI_SetReadOffsetAndLength AI_ReadBinary
AI_StopTask AI_IsSaveFile AI_SetReReadData
DEV_SetNetInfo DEV_GetNetInfo AI_SetReadComplete

◆ 暂停设备

函数原型：

Visual C++:

BOOL AI_StopTask (HANDLE hDevice);

Visual Basic:

Declare Function AI_StopTask Lib “NET2991” (ByVal hDevice As Long) As Boolean;

LabView:

请参考相关演示程序。

功能：暂停设备。它必须在调用 AI_StartTask 后才能调用此函数。该函数除了停止 AI 设备不再转换以外，不改变设备的其他任何状态。

参数：

hDevice 设备对象句柄,它由 DEV_Create()函数创建。

返回值：若调用成功，则返回 TRUE，且 AD 转换立即停止，否则返回 FALSE。

相关参数： DEV_Release DEV_IsLink AI_InitTask
AI_ClearBuffer AI_StartTask AI_SendSoftTrig AI_GetStatus
AI_GetClockStatus AI_SetReadOffsetAndLength AI_ReadBinary
AI_StopTask AI_IsSaveFile AI_SetReReadData
DEV_SetNetInfo DEV_GetNetInfo AI_SetReadComplete

◆ 软件触发

函数原型：

Visual C++:

BOOL AI_SendSoftTrig (HANDLE hDevice);

Visual Basic:

Declare Function AI_SendSoftTrig Lib “NET2991” (ByVal hDevice As Long) As Boolean;

LabView:

请参考相关演示程序。

功能：发送软件触发事件(Send Software Trigger),软件触发也叫强制触发。

参数：

hDevice 设备对象句柄,它由 DEV_Create()函数创建。

返回值：若调用成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release DEV_IsLink AI_InitTask
AI_ClearBuffer AI_StartTask AI_SendSoftTrig AI_GetStatus
AI_GetClockStatus AI_SetReadOffsetAndLength AI_ReadBinary
AI_StopTask AI_IsSaveFile AI_SetReReadData
DEV_SetNetInfo DEV_GetNetInfo AI_SetReadComplete

♦ 取得 AI 各种状态

函数原型：

Visual C++:

```
BOOL AI_GetStatus (HANDLE hDevice,
                   PNET2991_AI_STATUS pAIStatus);
```

Visual Basic:

```
Declare Function AI_GetStatus Lib “NET2991” (ByVal hDevice As Long
                   ByRef pAIStatus As PNET2991_AI_STATUS) As Boolean;
```

LabView:

请参考相关演示程序。

功能：发送软件触发事件(Send Software Trigger),软件触发也叫强制触发。

参数：

hDevice 设备对象句柄,它由 DEV_Create()函数创建。

pAIStatus AI 状态结构体,详细介绍请参考 NET2991.h 或 NET2991.Bas 或 NET2991.Pas 函数原型定义文件或《硬件参数结构》章节。

返回值：若调用成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release DEV_IsLink AI_InitTask
AI_ClearBuffer AI_StartTask AI_SendSoftTrig AI_GetStatus
AI_GetClockStatus AI_SetReadOffsetAndLength AI_ReadBinary
AI_StopTask AI_IsSaveFile AI_SetReReadData
DEV_SetNetInfo DEV_GetNetInfo AI_SetReadComplete

♦ 得到时钟状态,外部 10M 时钟时有效

函数原型：

Visual C++:

```
BOOL AI_GetClockStatus(HANDLE hDevice,
                   PNET2991_AI_STATUS pAIStatus);
```

Visual Basic:

```
Declare Function AI_ GetClockStatus Lib “NET2991” (ByVal hDevice As Long
                   ByRef pAIStatus As PNET2991_AI_STATUS) As Boolean;
```

LabView:

请参考相关演示程序。

功能：得到时钟状态,外部 10M 时钟时有效

参数：

hDevice 设备对象句柄,它由 DEV_Create()函数创建。

PAIStatus AI 状态结构体。

返回值：若调用成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release DEV_IsLink AI_InitTask
AI_ClearBuffer AI_StartTask AI_SendSoftTrig AI_GetStatus
AI_GetClockStatus AI_SetReadOffsetAndLength AI_ReadBinary
AI_StopTask AI_IsSaveFile AI_SetReReadData
DEV_SetNetInfo DEV_GetNetInfo AI_SetReadComplete

♦ 清除设备缓存

函数原型：

Visual C++:

BOOL AI_ClearBuffer (HANDLE hDevice);

Visual Basic:

Declare Function AI_ClearBuffer Lib “NET2991” (ByVal hDevice As Long) As Boolean;

LabView:

请参考相关演示程序。

功能：清除设备缓存，读取数据进行准备。

参数：

hDevice　设备对象句柄,它由 DEV_Create()函数创建。

返回值：若调用成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release　DEV_IsLink　AI_InitTask
AI_ClearBuffer　AI_StartTask　AI_SendSoftTrig　AI_GetStatus
AI_GetClockStatus　AI_SetReadOffsetAndLength　AI_ReadBinary
AI_StopTask　AI_IsSaveFile　AI_SetReReadData
DEV_SetNetInfo　DEV_GetNetInfo　AI_SetReadComplete

♦ 设置读取数据的长度

函数原型：

Visual C++:

```
BOOL AI_SetReadOffsetAndLength (  HANDLE hDevice,
                                  I32 nReadOffset,
                                  I32 nReadLength);
```

Visual Basic:

```
Declare Function SetReadOffsetAndLength Lib “NET2991” ( ByVal hDevice As Long,
                                  ByVal nReadOffset As Long,
                                  ByVal nReadLength As Long) As Boolean;
```

LabView:

请参考相关演示程序。

功能：设置读取数据的偏移位置和读取长度(有限时模式下使用)。

参数：

hDevice　设备对象句柄,它由 DEV_Create()函数创建。

nReadOffset　采样数据的偏移位置，参考点是整个采样序列的 0 位置(单位:字)

nReadLength　读取数据的长度

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release　DEV_IsLink　AI_InitTask
AI_ClearBuffer　AI_StartTask　AI_SendSoftTrig　AI_GetStatus
AI_GetClockStatus　AI_SetReadOffsetAndLength　AI_ReadBinary
AI_StopTask　AI_IsSaveFile　AI_SetReReadData
DEV_SetNetInfo　DEV_GetNetInfo　AI_SetReadComplete

♦ 读取采样数据

函数原型：

Visual C++:

```
BOOL AI_ReadBinary(HANDLE hDevice,
                   U32* pReadChannel,
                   U16 nBinArray[],
                   U32 nReadSampsPerChan,
                   U32* pSampsPerChanRead,
                   U32* pAvailSampsPerChan,
                   F64 fTimeout);
```

Visual Basic:

```
Declare Function AI_ReadBinary Lib "NET2991" ( ByVal hDevice As Long,
                                               ByRef pReadChannel As Long,
                                               ByRef nBinArray As Integer,
                                               ByVal nReadSampsPerChan As Long,
                                               ByRef pSampsPerChanRead As Long,
                                               ByRef pAvailSampsPerChan As Long,
                                               ByVal fTimeout As Double
                                               ) As Boolean;
```

LabView:

请参考相关演示程序。

功能：读取采样数据。

参数：

hDevice　设备对象句柄,它由 DEV_Create()函数创建。

pReadChannel　返回的读取数据的通道号(物理通道号),取值范围[0, 16], 0-15 对应 AI0-AI15, 16 对应 DI,有限模式使用，连续模式==NULL。

nBinArray[]　模拟数据数组(二进制原码数组),用于返回采样的二进制原码数据，取值区间由各通道采样时的采样范围决定(单位:V)。

nReadSampsPerChan　每通道请求读取的点数(单位：点)。

pSampsPerChanRead　返回每通道实际读取的点数(单位：点), =NULL,表示无须返回

pAvailSampsPerChan　任务中还存在的可读点数, =NULL,表示无须返回

fTimeout　超时时间，单位：秒, -1:无超时

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release　DEV_IsLink　AI_InitTask
AI_ClearBuffer　AI_StartTask　AI_SendSoftTrig　AI_GetStatus
AI_GetClockStatus　AI_SetReadOffsetAndLength　AI_ReadBinary
AI_StopTask　AI_IsSaveFile　AI_SetReReadData
DEV_SetNetInfo　DEV_GetNetInfo　AI_SetReadComplete

♦ 数据采集完成

函数原型：

Visual C++:

```
BOOL AI_SetReadComplete (HANDLE hDevice);
```

Visual Basic:

Declare Function AI_ClearBuffer Lib “NET2991” (ByVal hDevice As Long) As Boolean;

LabView:

请参考相关演示程序。

功能：非文件保存的情况下，某设备数据采集完成设置。

参数：

hDevice　设备对象句柄,它由 DEV_Create()函数创建。

返回值：若调用成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release　DEV_IsLink　AI_InitTask
AI_ClearBuffer　AI_StartTask　AI_SendSoftTrig　AI_GetStatus
AI_GetClockStatus　AI_SetReadOffsetAndLength　AI_ReadBinary
AI_StopTask　AI_IsSaveFile　AI_SetReReadData
DEV_SetNetInfo　DEV_GetNetInfo　AI_SetReadComplete

♦ 停止采集任务

函数原型：

Visual C++:

BOOL AI_StopTask(HANDLE hDevice);

Visual Basic:

Declare Function AI_StopTask Lib “NET2991” (ByVal hDevice As Long) As Boolean;

LabView:

请参考相关演示程序。

功能：停止采集任务。

参数：

hDevice　设备对象句柄,它由 DEV_Create()函数创建。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release　DEV_IsLink　AI_InitTask
AI_ClearBuffer　AI_StartTask　AI_SendSoftTrig　AI_GetStatus
AI_GetClockStatus　AI_SetReadOffsetAndLength　AI_ReadBinary
AI_StopTask　AI_IsSaveFile　AI_SetReReadData
DEV_SetNetInfo　DEV_GetNetInfo　AI_SetReadComplete

♦ 是否保存文件

函数原型：

Visual C++:

BOOL AI_IsSaveFile(HANDLE hDevice,
U32 bSaveFile);

Visual Basic:

Declare Function AI_IsSaveFile Lib “NET2991” (ByVal hDevice As Long,
ByVal　bSaveFileAs Long) As Boolean;

LabView:

请参考相关演示程序。

功能：是否保存文件。

参数：

hDevice　设备对象句柄,它由 DEV_Create()函数创建。

SaveFile　0：不保存文件(图形数字显示)，1：保存文件。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release　DEV_IsLink　AI_InitTask
AI_ClearBuffer　AI_StartTask　AI_SendSoftTrig　AI_GetStatus
AI_GetClockStatus　AI_SetReadOffsetAndLength　AI_ReadBinary
AI_StopTask　AI_IsSaveFile　AI_SetReReadData
DEV_SetNetInfo　DEV_GetNetInfo　AI_SetReadComplete

♦ 设置是否重新读取已有数据

函数原型：

Visual C++:

```
BOOL _AI_SetReReadData(HANDLE hDevice,
                       PNET2991_AI_PARAM pAIParam,
                       U32* pLastSampsPerChan,
                       U32 bReRead);
```

Visual Basic:

```
Declare Function AI_SetReReadData Lib “NET2991” ( ByVal hDevice As Long,
                                                  ByVal bReRead As Long) As Boolean;
```

LabView:

请参考相关演示程序。

功能：设置是否重新读取已有的数据。

参数：

hDevice　设备对象句柄,它由 DEV_Create()函数创建。

PNET2991_AI_PARAM pAIParam　AI 工作参数，它仅在此函数中决定硬件初始状态和各工作模式。

pLastSampsPerChan　历史待采样点数。

bReRead　设置是否重新读取。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关参数： DEV_Release　DEV_IsLink　AI_InitTask
AI_ClearBuffer　AI_StartTask　AI_SendSoftTrig　AI_GetStatus
AI_GetClockStatus　AI_SetReadOffsetAndLength　AI_ReadBinary
AI_StopTask　AI_IsSaveFile　AI_SetReReadData
DEV_SetNetInfo　DEV_GetNetInfo　AI_SetReadComplete

♦ 得到 dll 里设备采样的状态

函数原型：

Visual C++:

```
BOOL GetDevSampleStatus(PDEVICE_SAMP_STS pDevSampSts);
```

Visual Basic:

```
Declare Function GetDevSampleStatus Lib “NET2991” ( ByVal hDevice As Long) As Boolean;
```

LabView:

请参考相关演示程序。

功能：发送读取命令，此命令发送后设备会马上发送数据，此时就要进行读数据的操作。

参数：

pDevSampSts　设备采样状态。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关参数：DEV_Release　　DEV_IsLink

5 硬件参数结构

5.1 AI 硬件结构参数介绍

Visual C++:

```
typedef   struct   _NET2991_AI_PARAM
   {
      I8   szDevName[32];
      U32 nDeviceIP;
      U16 nDevicePort;
      U16 nLocalPort;
      NET2991_CH_PARAM CHParam[17];
      F64 fSampleRate;
      U32 nSampleMode;
      U32 nSampsPerChan;
      U32 nClockSource;
      U32 nReserved0;
      U32 nTriggerSource;
      U32 nTriggerDir;
      F32 fTriggerLevel;
      I32 nDelaySamps;
      U32 nReTriggerCount;
      U32 bMasterEn;
      U32 nReserved1;
      U32 nReserved2;
} NET2991_AI_PARAM, *PNET2991_AI_PARAM;

typedef struct _NET2991_CH_PARAM
{
    U32 bChannelEn;
    U32 nSampleRange;
    U32 nRefGround;
    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
} NET2991_CH_PARAM, *PNET2991_CH_PARAM;
```

此结构主要用于设定设备 AI 硬件参数值，用这个参数结构对设备进行硬件配置完全由 AI_InitTask 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。

结构体部分参数说明：

结构体 NET2991_AI_PARAM 中，NET2991_CH_PARAM CHParam[17]结构体数组中，

nSampleRange　模拟量输入量程选择

常量名	常量值	功能定义
NET2991_AI_SAMPRANGE_N10_P10V	0x00	±10V
NET2991_AI_SAMPRANGE_N5_P5V	0x01	±5V
NET2991_AI_SAMPRANGE_N2D5_P2D5V	0x02	±2.5V
NET2991_AI_SAMPRANGE_N1D25_P1D25V	0x03	±1.25V

nRefGround　模拟量输入模式选择

常量名	常量值	功能定义
NET2991_AI_REFGND_RSE	0x00	接地参考单端
NET2991_AI_REFGND_DIFF	0x01	差分输入

结构体 NET2991_AI_PARAM 其它参数：

nSampleMode　采样模式选择

常量名	常量值	功能定义
NET2991_AI_SAMPMODE_FINITE	0x02	有限采样
NET2991_AI_SAMPMODE_CONTINUOUS	0x03	连续采样

nTriggerSource　触发源选择

常量名	常量值	功能定义
NET2991_AI_TRIGSRC_NONE	0x00	无触发(软件触发)
NET2991_AI_TRIGSRC_DIGITAL	0x01	外部数字量触发
NET2991_AI_TRIGSRC_ANALOG	0x02	外部模拟量触发
NET2991_AI_TRIGSRC_SYNC	0x03	同步触发

nTriggerDir　触发方向选择

常量名	常量值	功能定义
NET2991_AI_TRIGDIR_FALLING	0x00	下降沿触发
NET2991_AI_TRIGDIR_RISING	0x01	上升沿触发
NET2991_TRIGDIR_POSIT_BOTH	0x02	变化(上升下降均触发)

nClockSource　时钟源选择

常量名	常量值	功能定义
NET2991_AI_CLOCKSRC_LOCAL	0x00	本地时钟(通常为本地晶振时钟 OSCCLK),也叫内部时钟
NET2991_AI_CLOCKSRC_CLKIN_10M	0x01	外部参考 10M 时钟定时触发
NET2991_AI_CLOCKSRC_MAIN_BOARD	0x02	主卡时钟
NET2991_AI_CLOCKSRC_CLKIN	0x03	外部采样时钟

5.2 AI 采样的硬件工作状态结构体

Visual C++:

```
typedef struct _NET2991_AI_STATUS
{
    U32 bTaskDone;
    U32 bTriggered;
    U32 bFreq10M;
    U32 nClockInFreq;
    U32 nSampTaskState;
    U32 nAvailSampsPerChan;
    U32 nMaxAvailSampsPerChan;
    U32 nBufSampsPerChan;
    I64 nSampsPerChanAcquired;
    U32 nHardOverflowCnt;
    U32 nSoftOverflowCnt;
    U32 nInitTaskCnt;
    U32 nReleaseTaskCnt;
    U32 nStartTaskCnt;
    U32 nStopTaskCnt;
    U32 nTransRate;
    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
    U32 nReserved3;
    U32 nReserved4;
} NET2991_AI_STATUS, *PNET2991_AI_STATUS;
```

此结构体在 AI_GetStatus 中使用。

结构体部分参数说明：

bTaskDone　AI 采样任务是否结束 1:表示已结束; 0:表示未结束

bTriggered　AI 是否被触发，1:表示已被触发; 0:表示未被触发(默认)

bFreq10M　外部输入信号是否 10M

nClockInFreq　外部时钟信号频率，单位 Hz

nSampTaskState　采样任务状态, =1:正常，其它值表示有异常情况

nAvailSampsPerChan　每通道有效点数，只有它大于当前指定读数长度时才能调用 AI_ReadAnalog()立即读取指定长度的采样数据

nMaxAvailSampsPerChan　自 AI_StartTask()后每通道出现过的最大有效点数，状态值范围[0, nBufSampsPerChan],它是为监测采集软件性能而提供，如果此值越趋近于 1，则表示意味着性能越高，越不易出现溢出丢点的可能。

nBufSampsPerChan　每通道缓冲区大小(采样点数）

nSampsPerChanAcquired　每通道已采样点数(自启动 AI_StartTask()之后所采样的点数)，这个只

是给用户的统计数据

nHardOverflowCnt　硬件溢出计数(在不溢出情况下恒等于 0)

nSoftOverflowCnt　软件溢出计数(在不溢出情况下恒等于 0)

nInitTaskCnt　初始化采样任务的次数(即调用 AI_InitTask()的次数)

nReleaseTaskCnt　释放采样任务的次数(即调用 AI_ReleaseTask()的次数)

nStartTaskCnt　启动采样任务的次数(即调用 AI_StartTask()的次数)

nStopTaskCnt　停止采样任务的次数(即调用 AI_StopTask()的次数)

nTransRate　传输速率，即每秒传输点数(sps)，作为 USB 及应用软件传输性能的监测信息

nReserved0-4　保留字段(暂未定义)

5.3 设备网络参数信息结构体

Visual C++:

```
typedef struct _DEVICE_NET_INFO
{
    U32 nDeviceIP;
    U16 nDevicePort;
    U16 nReserved0;
    U64 nMAC;
    U32 bOneline;
    U32 nSubnetMask;
    U32 nGateway;
    U32 nReserved1;
} DEVICE_NET_INFO, *PDEVICE_NET_INFO;
```

结构体部分参数说明：

nDeviceIP　IP 地址　192.168.0.1

nDevicePort　端口号　最大 65535

nReserved0　保留字段(暂未定义)

nMAC　网卡物理地址,用户一般不可更改

bOneline　在线状态，1:在线; 0:离线(下线)

nSubnetMask　子网掩码, "255.255.255.0"

nGateway　网关, "192.168.0.1"

nReserved1　保留字段(暂未定义)

5.4 dll 内部设备采样状态

```
typedef struct _DevSampSts
  {
    U32 nDevIP;
    I32 nSampSts;
    I32 nChan;
    I8 szFileName[256];
    U32 nReserved0;
    U64 ullFileSize;
    }DEVICE_SAMP_STS, *PDEVICE_SAMP_STS;
```

其中设备采样状态 nSampSts,定义如下：

```
#define NET2991_AI_SAMPSTS_NONE                    0 // 无采样状态
#define NET2991_AI_SAMPSTS_REDUCESPEED_FAIL        1 // 降速失败， 有限时返回
#define NET2991_AI_FILE_READCH          2 // 保存文件时正在读取的通道号,有限时返回
#define NET2991_AI_SAMPSTS_SAVEFILE                3 // 正在保存文件
#define NET2991_AI_SAMPSTS_SAVEFILE_COMPLETE       4 // 保存文件完成
#define NET2991_AI_SAMPSTS_START_PROCESSFFILE      5 // 开始文件处理
```

```
#define NET2991_AI_SAMPSTS_FINISH_PROCESSFFILE        6 // 完成文件处理
```

注：如需其他自定义的状态请从数值 100 开始定义。

6 数据格式转换与排列规则

6.1 AI 原码 LSB 数据转换成电压的换算方法

本设备的第一个数据通道默认为 16 个 DI 组合而成的数据，不进行换算，只以 16 进制显示。

根据所选量程，按照下表公式进行换算即可，假如设备数据组合好后放置在 USHORT 类型的 AIBuffer[]缓冲中(AIBuffer 的数据是以低字节在前，高字节在后的两字节组合而成)，这里以第一个点 AIBuffer[0]为例。

表 6-1-1：AD 原码 LSB 数据转换成电压值的换算方法。（以缓冲区 AIBuffer[]第 1 个点为例。）

量程	计算机语言换算公式(ANSI C 语法)	Volt 取值范围（mV）
±10000mV	Volt = (20000.00/65536)*(AIBuffer[0] &0xFFFF) – 10000.00	[-10000, +9999.69]
±5000mV	Volt = (10000.00/65536)* (AIBuffer[0] &0xFFFF) – 5000.00	[-5000, +4999.84]
±2500mV	Volt = (5000.00/65536) * (AIBuffer[0] &0xFFFF) – 2500.00	[-2500, +2499.92]
±1250mV	Volt = (2500.00/65536) * (AIBuffer[0] &0xFFFF) – 1250.00	[-1250, +1249.96]

下面举例说明各种语言的换算过程(以±10000mV 量程为例)：

Visual C++：

```
Lsb = AIBuffer[0] &0xFFFF;
Volt = (20000.00/65536) * Lsb -10000.00;
```

Visual Basic:

```
Lsb = AIBuffer[0] And &HFFFF;
Volt = (20000.00/65536) * Lsb – 10000.00;
```

LabVIEW:

请参考相关演示程序。

6.2 AI 采集函数的 AIBuffer 缓冲区中的数据排放规则

AIBuffer 的数据是以低字节在前，高字节在后的两字节组合而成。

具体根据采集模式 AIBuffer 数据排放方式不同。

在有限方式中，读某个通道，AIBuffer 此时就对应某个通道的数据。如读通道 0：

数据缓冲区索引号	0	1	2	3	4	5	6	7	8	…
通道号	0	0	0	0	0	0	0	0	0	…

在连续模式中，每次都是所选的全通道读取，如下表(以全选 17 个通道为例)：

数据缓冲区索引号	0	1	2	3	…	14	15	16	17	18	19	20	…
通道号	0	1	2	3	…	14	15	16	0	1	2	3	…

6.3 AI 测试应用程序创建并形成的数据文件格式

首先该数据文件从始段 0 字节位置开始往后至 nThisBytes 字节位置宽度输入文件头信息，而从 nThisBytes 开始才是真正的数据。nThisBytes 的取值通常等于本头信息的字节数大小。文件头信息包含如下结构体所示，对于更详细的内容请参考 Visual C++高级演示程序工程中的 HeadFormat.h 文件。

```
typedef struct _CHAN_INFO
    {
        ULONG nChannel;
        char strChanName[36];
        double fMaxVolt;
        double fMinVolt;
        double fAmplitude;
        double fHalfOfAmp;
        double fCodeWidth;
        double fAmpFactor;
        ULONG nPolarity;
        ULONG nReserved0;
        ULONG nReserved1;
        ULONG nReserved2;
    } FCHAN_INFO, *PFCHAN_INFO;

typedef struct _FILE_HEAD_INFO
    {
        ULONG nHeadBeginFlag;
        ULONG nThisBytes;
        ULONG nFileType;
        ULONG nBusType;
        ULONG nVendorID;
        ULONG nProductID;
        ULONG nHeadVersion;
        ULONG nDllVer;
        ULONG nFirmwaveVer;
        SYSTEMTIME Time;
        char strProductName[32];
        ULONG nMaskCode;
        ULONG nXORCode;
        LONG nMaxCode;
        LONG nMinCode;
        ULONG nDataType;
        ULONG bChanCross;
        ULONG nSegmentCount;
        LONGLONG nTotalPoints;
        LONGLONG nCHPoints;
```

```
    double fCHSampleRate;
    ULONG nSampChanCount;
    ULONG nReserved0;
    FCHAN_INFO ChanInfo[MAX_AD_CHANNELS];
    NET2991_PARA_AD ADParam;
    ULONG nReserved1;
    ULONG nHeadEndFlag;
} FILE_HEAD_INFO, *PFILE_HEAD_INFO;
```

结构体参数说明：

bChanCross　通道交叉排放，有限模式为 FALSE 非交叉排放,连续模式为 TRUE 交叉排放。

nSegmentCount　触发段数(即采样段数)，对于有限模式是选取的通道数，如选择了 10 个通道，nSegmentCount 就等于 10；有限模式 nSegmentCount 等于 1。

nTotalPoints　各段的总采样点数。就是采样数据的总数，按字计算。

nCHPoints　各通道的点数，即每个通道的采样的数据点数，字为单位。

fCHSampleRate　各通道的采样速率(Hz，即 NET2991.h 文件中 NET2991_PARA_AD 结构体中的 fFrequency。

nSampChanCount　每段的采样通道数量，对于有限模式，nSampChanCount 等于 1，对于连续模式为所选取的通道数，如选取 10 个采样通道，则 nSampChanCount 等于 10。

对于文件长度解析：

数据长度(字节) = 各通道点数(nCHPoints) * 每段的采样通道数量(nSampChanCount) * 采样段数(nSegmentCount) * 2；

整个文件的大小 = 数据长度(字节) + 文件头(FILE_HEAD_INFO 结构体)的大小(27280 字节);

有限模式每段只有一个通道数据,例如只有 3 个通道 AI0,AI1,AI2,则每段的形式为 AI0 AI0 AI0 ... | AI1

AI1 AI1... | AI2 AI2 AI2... ,即 3 个采样段数；

连续模式的每段包括所有通道数据，例如只有 3 个通道 AI0,AI1,AI2,则每段的形式为 AI1 AI2 AI3 AI1AI2 AI3 AI1 AI2 AI3...,即 1 个采样段数。

7 上层用户函数接口应用实例

如果您想快速的了解驱动程序的使用方法和调用流程，以最短的时间建立自己的应用程序，那么我们强烈建议您参考相应的简易程序。此种程序属于工程级代码，可以直接打开不用作任何配置和代码修改即可编译通过，运行编译链接后的可执行程序，即可看到预期效果。

如果您想了解硬件的整体性能、精度、采样连续性等指标以及波形显示、数据存盘与分析、历史数据回放等功能，那么请参考高级演示程序。特别是许多不愿意编写任何程序代码的用户，您可以使用高级程序进行采集、显示、存盘等功能来满足您的要求。甚至可以用我们提供的专用转换程序将高级程序采集的存盘文件转换成相应格式，即可在 Excel、MatLab 第三方软件中分析数据（此类用户请最好选用通过 Visual C++制作的高级演示系统）。

7.1 简易程序演示说明

7.1.1 怎样进行 AD 数采操作

Visual C++:

其详细应用实例及正确代码请参考 Visual C++简易演示系统及源程序，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程(主要参考 NET2991.h 和 Sys.cpp)。

[程序]丨**[阿尔泰测控演示系统]**丨**[NET2991 数据采集卡]**丨**[Microsoft VC++]**丨[简易代码演示]丨[AI 简易源程序]

其默认存放路径为：系统盘\NET\NET2991\SAMPLES\VC\SIMPLE\AD

7.2 高级程序演示说明

高级程序演示了本设备的所有功能，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程(主要参考 NET2991.h 和 Sys.cpp)。

[程序]丨**[阿尔泰测控演示系统]**丨**[NET2991 数据采集卡]**丨**[Microsoft VC++]**丨[高级代码演示]丨[演示源程序]

其默认存放路径为：系统盘\NET\NET2991\SAMPLES\VC\ADVANCED

其他语言的演示可以用上面类似的方法找到。

8 共用函数介绍

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。

8.1 公用接口函数总列表

表 6-1-1：公用接口函数总列表（每个函数省略了前缀“NET2991_”）

函数名	函数功能	备注
① 校准函数		
AI_SelCal	AI 校准	
② 得到设备版本信息函数		
DEV_GetVersion	得到设备版本信息函数	
③ 处理文件函数		
FILE_Create	创建文件	
④ 用于文件保存时出错后，关闭文件		
FILE_Close	关闭文件	

8.2 文件操作函数原型说明

此数据文件的保存方式为分批保存方式，当数据大小等于某个设定大小后，先把数据保存到一个临时文件，直到数据分批保存完成，会形成一些临时的小文件，最终会把这些临时文件组合成完整的文件。

♦ 创建文件

函数原型：

Visual C++:

```
BOOL FILE_Create(HANDLE hDevice,
                 char* szFilePath,
                 char* szFileName);
```

Visual Basic:

```
Declare Function FILE_Create Lib “NET2991” ( ByVal hDevice As Long,
                                             ByRef svPath As String,
                                             ByRef svFile As String ) As Boolean;
```

LabView:

请参考相关演示程序。

功能：创建文件，使用 VB/LabView 请使用此函数。此文件创建后，从设备采集的数据最终保存在此文件中，保存文件的其他操作在 dll 内部实现。

参数：

svPath　设置要保存文件的路径。

svFile　设置要保存的文件名。

返回值：如果执行成功，返回 TRUE，反之 FALSE。

♦ 创建文件

函数原型：

Visual C++:

BOOL FILE_Close(HANDLE hDevice);

Visual Basic:

Declare Function FILE_Close Lib “NET2991” (ByVal hDevice As Long) As Boolean;

LabView:

请参考相关演示程序。

功能：用于文件保存时出错后，关闭文件。

参数：

hDevice 设备对象句柄,它由 DEV_Create()函数创建。

返回值：如果执行成功，返回 TRUE，反之 FALSE。

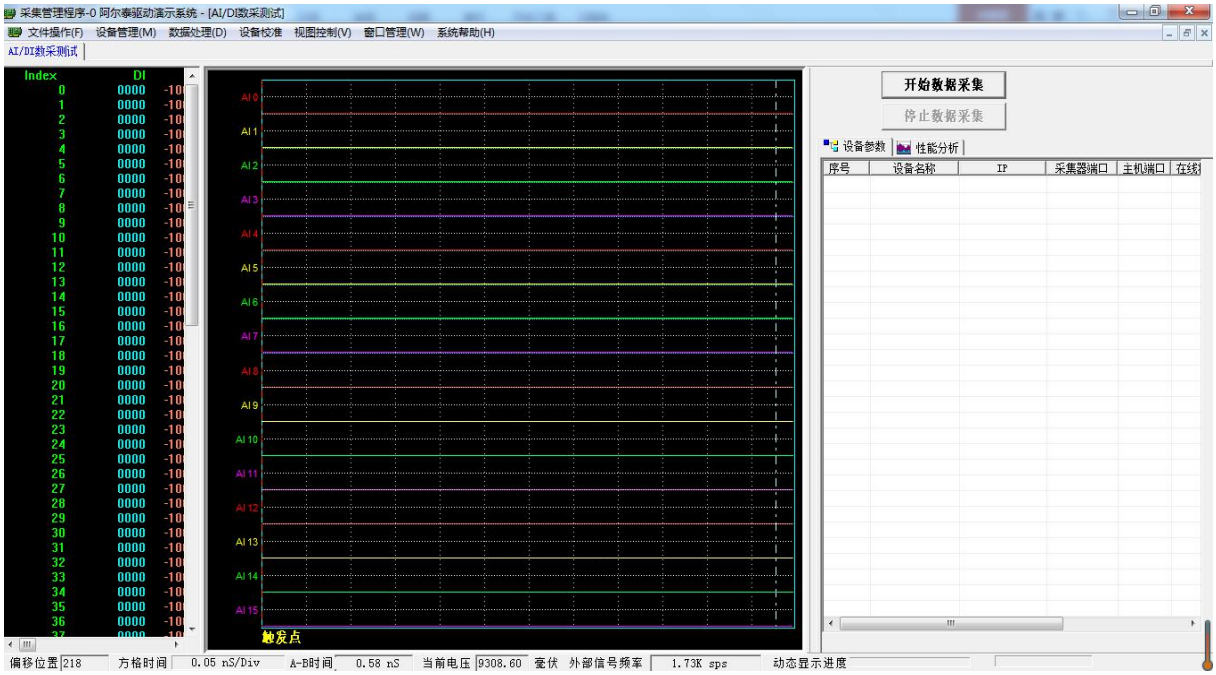
9 修改历史

修改时间	版本号	修改原因及内容
2017.6.22	V6.00.00	第一版

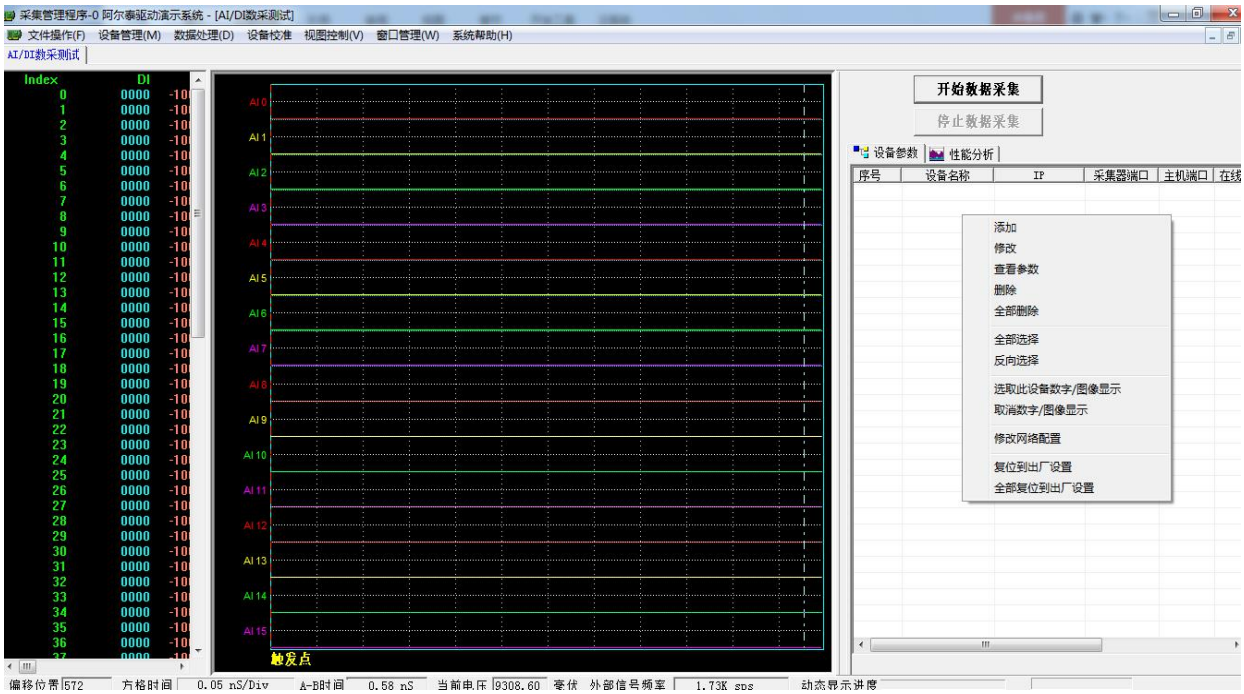
附录

此处主要介绍 NET2991 单独采集设备的使用方式。

1、打开 NET2991 程序后，即可看到下面界面:

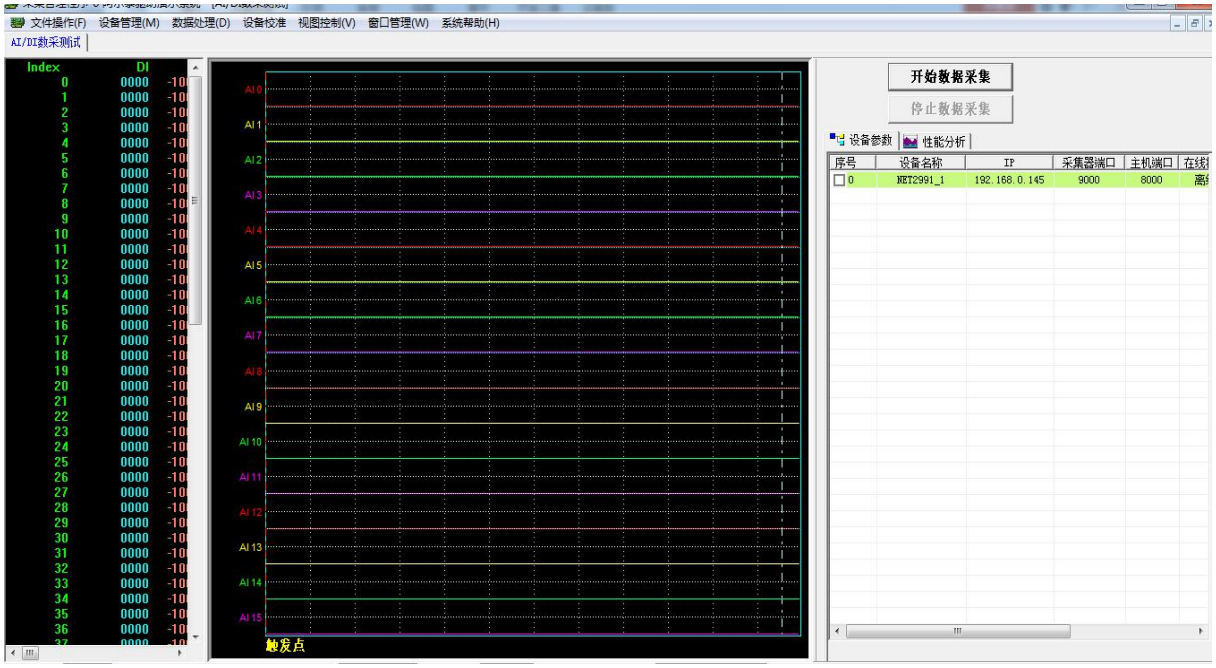


2、在最右边的设备参数界面下点击鼠标右键,如下：



在此可以添加设备、修改设备信息、删除已添加的设备，选择要数字/图像显示的设备或取消显示，修改网络参数。

添加设备后，为下图：



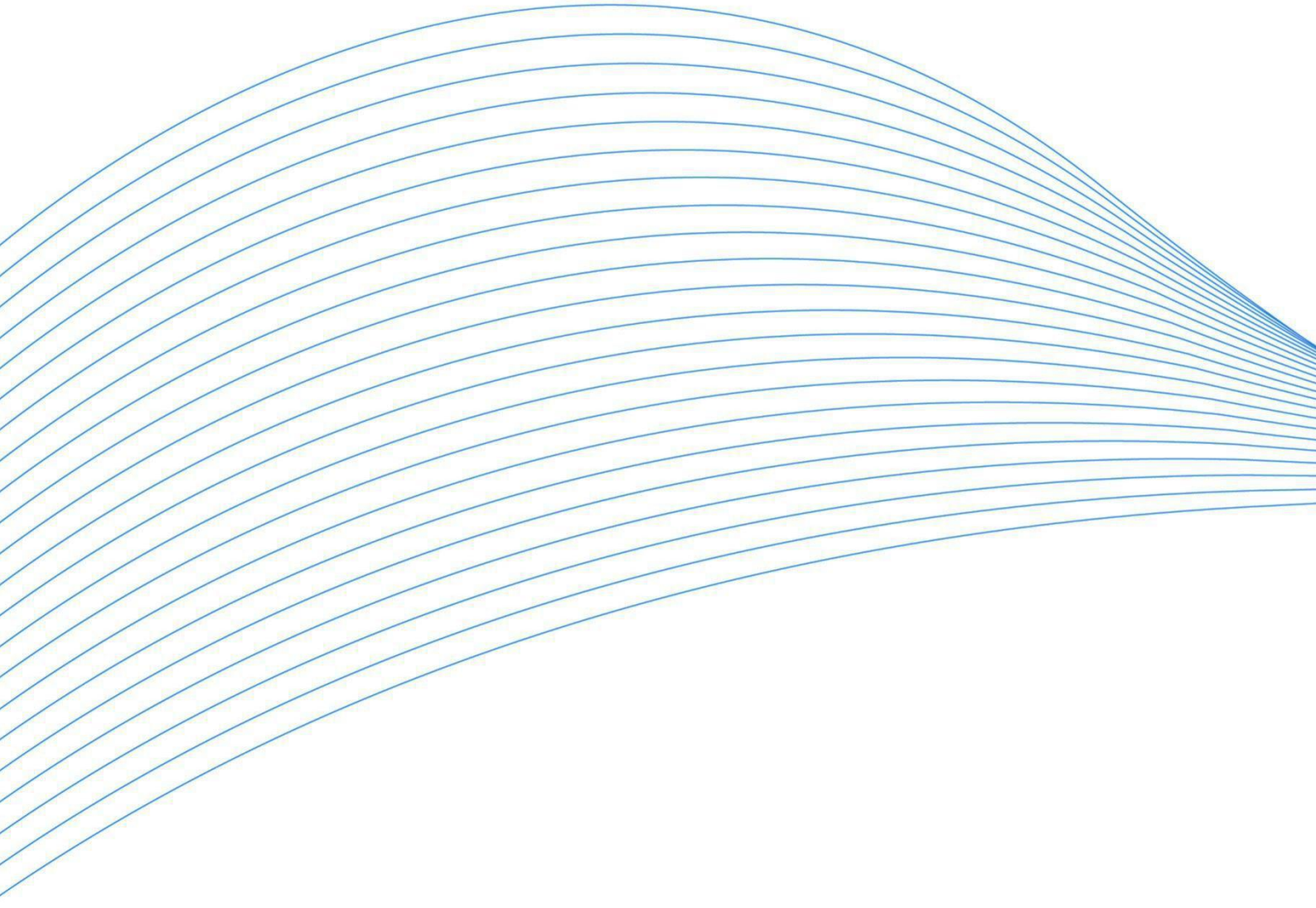
鼠标左键点击序号列中的多选框，即可选择此设备进行采集操作，如需多设备，请重复 2、3 的操作，如不需要采集某设备，可点击序号列的多选框，取消选中。

如此即可完成采集前配置，如果需要保存文件，可选择菜单栏“文件操作”——>“新建文件操作”，即为文件保存。

NET2991 软件采集数据前，请在菜单栏“设备管理”——>“数据回传方向”，选择采集管理器方向，在“设备管理”——>“采样模式”选择是有限还是连续模式。

如需重新读取硬件已有的历史数据，可选择菜单栏“数据处理”——>“重新传输”，“重新传输”菜单栏选择中前为“√”即为重新传输，再次点击前无“√”即取消重新传输功能。

另外注意的是，多卡采集的话，设备名称、设备 IP、主机端口不能重复，同时也要注意不要配置已经占用的主机端口。



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