PCH2004 任意波形发生器

WIN2000/XP 驱动程序使用说明书

阿尔泰科技发展有限公司

产品研发部修订

目录

第一章 版权信息

第二章 绪论

第一节、使用上层用户函数，高效、简单

如果您只关心通道及频率等基本参数，而不必了解复杂的硬件知识和控制细节，那么我们强烈建议您使用上层用户函数,它们就是几个简单的形如 Win32 API 的函数,具有相当的灵活性、可靠性和高效性。诸如 InitDeviceDA 等。而底层用户函数如 WritePortWord、ReadPortWord……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，您基本上可以不必参考硬件说明书，除非您需要知道板上 D 型插座等管脚分配情况。

第二节、如何管理设备

由于我们的驱动程序采用面向对象编程，所以要使用设备的一切功能，则必须首先用 CreateDevice 函数创建一个设备对象句柄 hDevice，有了这个句柄，您就拥有了对该设备的绝对控制权。然后将此句柄作为参数传递给其他函数， 如 InitDeviceDA 可以使用 hDevice 句柄以程序查询方式初始化设备 DA 部件，最后可以通过 ReleaseDevice 将 hDevice 释放掉。

第三节、如何实现 DA 的简便输出

当您有了 hDevice 设备对象句柄后，然后反复调用 WriteDeviceProDA 函数输出每一个 DA 数据。

第四节、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，除非您要使用存盘功能。而 WritePortWord，ReadPortWord 则对 PCH 用户来讲，可以说完全是辅助性的，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）。

第三章 PCH 设备操作函数接口介绍

第一节、设备驱动接口函数列表

（下表中每个函数省略了前缀“PCH2004_”）

函数	函数功能	备注
①设备对象操作函数		
CreateDevice	创建设备对象	上层及底层用户
ReleaseDevice	关闭设备，且释放总线设备对象	上层及底层用户
②DA 数据采集操作函数		
WriteDeviceProDA	DA 输出函数	上层用户

使用需知：

要使用如下函数关键的问题是：

首先，必须在您的源程序中包含如下语句：

#include “C:\Art\PCH2004\INCLUDE\PCH2004.H”

注：以上语句采用默认路径和默认板号，应根据您的板号和安装情况确定 PCH2004.H 文件的正确路径，当然也可以把此文件拷到您的源程序目录中。

其次，您还应该在 Visual C++编译环境软件包的 Project Setting 对话框的 Link 属性页中的 Object/Library Module 输入行中加入指令 C:\Art\PCH2004\PCH2004.LIB 。

或者：单击 Visual C++编译环境软件包的 Project 菜单中的 Add To Project 的菜单项，在此项中再单击 Files…，在随后弹出的对话框中选择 PCH2004.Lib，再单击“确定”，即可完成。

注：以上语句采用默认路径和默认板号，应根据您的板号和安装情况确定 PCH2004.LIB 的路径，当然也可以把此文件拷到您的源程序目录中。

另外，在 Visual C++演示工程的目录下，也有相应的 PCH2004.h 和 PCH2004.Lib 文件。

C++ Builder:

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头文件(PCH2004.H)写进您的源程序头部。如：#include “\Art\PCH2004\Include\PCH2004.h”。然后再将 PCH2004.Lib 库文件分别加入到您的 C++ Builder 工程中。其具体办法是选择 C++ Builder 集成开发环境中的工程(Project)菜单中的“添加”（Add to Project）命令，在弹出的对话框中分别选择文件类型：Library file (*.lib) , 即可选择 PCH2004.Lib 文件。该文件的路径为用户安装驱动程序后其子目录 Samples\C_Builder 下。

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 PCH2004.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。

Delphi:

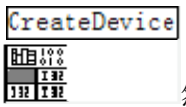
要使用如下函数一个关键的问题是首先必须将我们提供的单元模块文件（*.Pas）加入到您的 Delphi 工程中。其方法是选择 Delphi 编程环境中的 View 菜单,执行其中的"Project Manager"命令,在弹出的对话中选择*.exe 项目，再单击鼠标右键，最后 Add 指令，即可将 PCH2004.Pas 单元模块文件加入到工程中。或者在 Delphi 的编程环境中的 Project 菜单中，执行 Add To Project 命令，然后选择*.Pas 文件类型也能实现单元模块文件的添加。该文件的路径为用户安装驱动程序后其子目录 Samples\Delphi 下面。最后请在使用驱动程序接口的源程序文件中的头部的 Uses 关键字后面的项目中加入：“PCH2004”。如：

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  PCH2004;  // 注意： 在此加入驱动程序接口单元 PCH2004
```

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：

一、 在 LabView 中打开 PCH2004.VI 文件，用鼠标单击接口单元图标，比如 CreateDevice 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续该接口模块即可顺利使用。

二、 根据 LabView 语言本身的规定，接口单元图标以黑色的较粗的中竖线为中心，以左边的方格为数据输入端，右边的方格为数据的输出端，如 ReadDeviceProAD 接口单元，左边为设备对象句柄、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。

三、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。

第二节、设备对象管理函数原型说明

◆ 创建设备对象函数（逻辑号）

函数原型：

Visual C++ & C++Builder：

HANDLE CreateDevice (WORD BaseAddress)

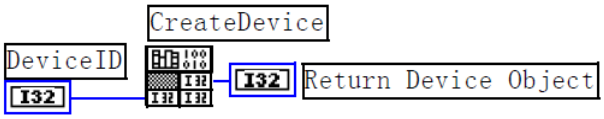
Visual Basic：

Declare Function CreateDevice Lib "PCH2004" (ByVal BaseAddress As Integer) As long

Delphi：

Function CreateDevice(BaseAddress :Word):Integer;
StdCall; External 'PCH2004' Name ' CreateDevice';

LabVIEW：



功能：该函数使用逻辑号创建设备对象，并返回其设备对象句柄 BaseAddress。只有成功获取 BaseAddress，您才能实现对该设备所有功能的访问。

参数：BaseAddress 基地址。

返回值：如果执行成功，则返回设备对象句柄；如果没有成功，则返回错误码 INVALID_HANDLE_VALUE。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： CreateDevice ReleaseDevice

Visual C++ & C++Builder 程序举例：

```
    :
HANDLE hDevice;                                  // 定义设备对象句柄
BaseAddress= CreateDevice ( 0 );                 // 创建设备对象,并取得设备对象句柄
if(BaseAddress == INVALIDE_HANDLE_VALUE);        // 判断设备对象句柄是否有效
{
    return;                                      // 退出该函数
 }
    :
```

Visual Basic 程序举例：

```
    :
Dim BaseAddress As Long                          ’ 定义设备对象句柄
BaseAddress = CreateDevice ( 0 )                 ’ 创建设备对象, 并取得设备对象句柄
If BaseAddress = INVALID_HANDLE_VALUE Then       ’ 判断设备对象句柄是否有效
Else
    Exit Sub                                     ’ 退出该过程
End If
    :
```

◆ **释放设备对象所占的系统资源及设备对象**

函数原型：

Visual C++ & C++Builder:

```
BOOL ReleaseDevice(HANDLE hDevice)
```

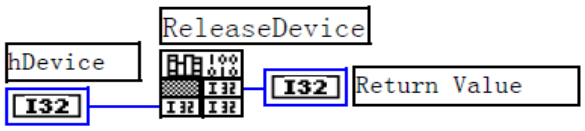
Visual Basic:

```
Declare Function ReleaseDevice Lib "PCH2004" (ByVal hDevice As Long ) As Boolean
```

Delphi:

```
Function ReleaseDevice(hDevice : Integer):Boolean;
                        StdCall; External 'PCH2004' Name ' ReleaseDevice';
```

LabVIEW:



功能：释放设备对象所占用的系统资源及设备对象自身。

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

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

相关函数： CreateDevice

应注意的是，CreateDevice 必须和 ReleaseDevice 函数一一对应，即当您执行了一次 CreateDevice 后，再一次执行这些函数前，必须执行一次 ReleaseDevice 函数，以释放由 CreateDevice 占用的系统软硬件资源，只有这样，当您再次调用 CreateDevice 函数时，那些软硬件资源才可被再次使用。

第三节、DA 操作函数原型说明

◆ **输出 DA 数据**

函数原型:

Visual C++ & C++Builder:

```
BOOL WriteDeviceProDA ( HANDLE hDevice,
                        WORD nDAData,
                        int nDAChannel)
```

Visual Basic:

```
 Declare Function WriteDeviceProDA Lib "PCH2004" (
                        ByVal hDevice As Long, _
                        ByVal nDAData As Integer,_
                        nDAChannel As Long) As Boolean
```

Delphi:

```
Function WriteDeviceProDA (hDevice : Integer;
```

nDAData: SmallInt;
nDAChannel: LongInt):Boolean;
StdCall; External 'PCH2004' Name ' WriteDeviceProDA ';

LabView：

功能：向指定通道上输出一个点的 DA 数据

参数：

hDevice 设备对象句柄,它应由 CreateDevice 创建。

nDAData 准备输出的 DA 原始数据，注意它的换算关系请参考数据转换章节。

nDAChannel DA 通道，取值范围为 0-7，若等于 0xFFFF，则本函数用同样的值 nDAData 同时输出到所有通道上。

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

相关函数： CreateDevice ReleaseDevice

第四章 数据格式转换与排列规则

DA 的电压值如何转换成输出到 DA 转换器的 LSB 原码数据

量程(伏)	计算机语言换算公式	Lsb 取值范围
0～5000mV	Lsb = Volt / (5000 / 4096)	[0，4095]
0～10000mV	Lsb = Volt / (10000 / 4096)	[0，4095]
0～10800mV	Lsb = Volt / (10800 / 4096)	[0，4095]
±5000mV	Lsb = Volt / (5000 / 4096)	[0，4095]
±10000mV	Lsb = Volt / (10000 / 4096) + 2048	[0，4095]
±10800mV	Lsb = Volt / (10800 /4096) + 2048	[0，4095]

第五章 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）。

第一节、简易程序演示说明

怎样进行批量 DA 数据输出

其详细应用实例及工程级代码请参考 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 PCH2004.h 和 Sys.cpp)。

[程序] □[阿尔泰测控演示系统]□ [Microsoft Visual C++]□ [简易代码演示]□ [DA 批量输出演示源程序]

其简易程序默认存放路径为：系统盘\PCH\PCH2004\SAMPLES\VC\SIMPLE\DA\BULK

其他语言的演示可以用上面类似的方法找到。

第二节、高级程序演示说明

高级程序演示了本设备的所有功能，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开

基于 VC 的 Sys 工程(主要参考 PCH2004.h 和 DADoc.cpp)。

[程序] □[阿尔泰测控演示系统]□ [Microsoft Visual C++]□ [高级代码演示]

其默认存放路径为：系统盘\PCH\ PCH2004\SAMPLES\VC\ADVANCED

其他语言的演示可以用上面类似的方法找到。

第六章 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。

第一节、公用接口函数列表

函数名	函数功能	备注
ISA 总线 I/O 端口操作函数		
WritePortWord	以字(16Bit)方式写 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

第二节、IO 端口读写函数

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 WritePortWordEx 或 ReadPortWordEx 等有“Ex”后缀的函数即可。

◆ **以双字(16Bit)方式写 I/O 端口**

函数原型:

Visual C++ & C++ Builder:

```
BOOL WritePortWord (HANDLE hDevice, UINT nPort, WORD Value)
```

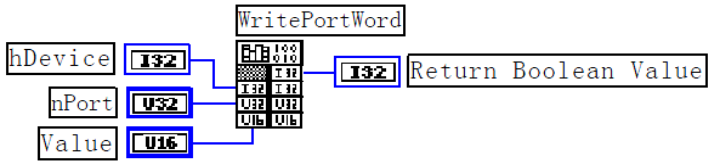
Visual Basic:

```
Declare Function WritePortWord Lib "PCH2004" (ByVal hDevice As Long, _
                                              ByVal nPort As Long, _
                                              ByVal Value As Integer) As Boolean
```

Delphi:

```
Function WritePortWord(hDevice : Integer; nPort:LongWord; Value:Word):Boolean;
                       StdCall; External 'PCH2004' Name 'WritePortWord';
```

LabVIEW:



功能：以双字(16Bit)方式写 I/O 端口。

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

nPort 设备的 I/O 端口号。

Value 写入由 nPort 指定端口的值。

返回值：若成功，返回 TRUE，否则返回 FALSE，用户可用 GetLastError 捕获当前错误码。

相关函数： CreateDevice WritePortWord ReadPortWord

◆ **以双字节(16Bit)方式读 I/O 端口**

函数原型:

Visual C++ & C++ Builder:

WORD ReadPortWord(HANDLE hDevice, UINT nPort)

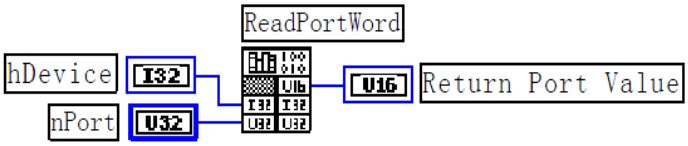
Visual Basic:

```
Declare Function ReadPortWord Lib "PCH2004" (ByVal hDevice As Long, _
                                             ByVal nPort As Long ) As Integer
```

Delphi:

```
Function ReadPortWord(hDevice : Integer; nPort:LongWord):Word;
                      StdCall; External 'PCH2004' Name 'ReadPortWord';
```

LabVIEW:



功能：以双字节(16Bit)方式读 I/O 端口。

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

nPort 设备的 I/O 端口号。

返回值：返回由 nPort 指定的端口的值。

相关函数： CreateDevice WritePortWord ReadPortWord