PCH8603W1 WIN2000/XP 驱动程序使用说明书

请您务必阅读《使用纲要》，他会使您事半功倍!

目　　录

提醒用户：

通常情况下，WINDOWS 系统在安装时自带的 DLL 库和驱动不全，所以您不管使用那种语言编程，请您最好先安装上Visual C++6.0 版本的软件，方可使我们的驱动程序有更完备的运行环境。

有关设备驱动安装和产品二次发行请参考 PCH8603W1Inst.doc 文档。

第一章 版权信息与命名约定

第一节、版权信息

第二节、命名约定

一、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PCHxxxx_则被省略。如PCH8603W1_CreateDevice 则写为CreateDevice。

二、函数名及参数中各种关键字缩写规则

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。

第二章 PCH 设备专用函数接口介绍

第一节、设备驱动接口函数列表（每个函数省略了前缀“PCH8603W1_”）

函数名	函数功能	备注
① 设备对象操作函数		
CreateDevice	创建设备对象	
ReleaseDevice	释放设备对象	
GetDeviceCurrentID	取得 CAN 卡设备逻辑ID和物理ID	
ListDeviceDlg	列表设备	

使用需知

Visual C++ & C++Builder&LabWindow：

首先将 PCH8603W1.h 和 PCH8603W1.lib 两个驱动库文件从相应的演示程序文件夹下复制到您的源

程序文件夹中，然后在您的源程序头部添加如下语句，以便将驱动库函数接口的原型定义信息和驱动接口导入库(PCH8603W1.lib)加入到您的工程中。

#include “PCH8603W1.H”

在 VC 中，为了使用方便，避免重复定义和包含，您最好将以上语句放在StdAfx.h 文件。一旦完成了以上工作，那么使用设备的驱动程序接口就跟使用VC/C++Builder/LabWindow 自身的各种函数，其方法一样简单，毫无二别。

关于PCH8603W1.h 和 PCH8603W1.lib 两个文件均可在演示程序文件夹下面找到。

Visual Basic:

首先将PCH8603W1.Bas 驱动模块头文件从 VB 的演示程序文件夹下复制到您的源程序文件夹中，然后将此模块文件加入到您的 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PCH8603W1.Bas 模块文件即可，一旦完成以上工作后，那么使用设备的驱动程序接口就跟使用 VB 自身的各种函数，其方法一样简单，毫无二别。

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，我们不保证能完全顺利运行。

Delphi:

首先将 PCH8603W1.Pas 驱动模块头文件从 Delphi 的演示程序文件夹下复制到您的源程序文件夹中，然后将此模块文件加入到您的 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PCH8603W1.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。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：“PCH8603W1”。如：

```
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
PCH8603W1;  // 注意： 在此加入驱动程序接口单元PCH8603W1
```

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 的驱动程序接口的详细说明请参考其演示源程序。

第二节、设备对象管理函数原型说明

♦ 创建设备对象函数

函数原型：

Visual C++ & C++ Builder&LabWindow：

HANDLE CreateDevice(int DeviceLgcID)

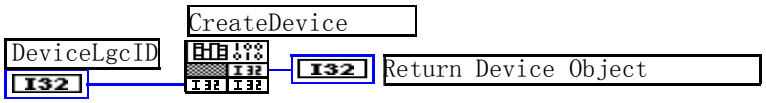
Visual Basic：

Declare Function CreateDevice Lib "PCH8603W1" (ByVal DeviceLgcIDAs Long) As Long

Delphi：

Function CreateDevice(DeviceLgcID:LongInt):LongInt; StdCall; External 'PCH8603W1' Name 'CreateDevice';

LabView:



功能：该函数负责创建设备对象，并返回其设备对象句柄。

参数：

DeviceID设备逻辑ID(Identifier)标识号。当向同一个Windows 系统中加入若干相同类型的设备时，系统将以该设备的“基本名称”与 ID 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PCH8603W1 AD 模板时，系统则以“PCH8603W1”作为基本名称，再以 ID 的初值组合成该设备的标识符“PCH8603W1-0”来确认和管理这第一个设备，若用户接着再添加第二个 PCH8603W1 AD 模板时，则系统将以“PCH8603W1-1”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 CAN 设备时，ID 应置 0，第二应置 1，也以此类推。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。

备注：创建完成后，如果释放 CAN 需要用PCH8603W1_ReleaseDevice 来关闭 CAN卡。

相关函数： ReleaseDevice

♦ 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&LabWindow:

BOOL ReleaseDevice(HANDLE hDevice)

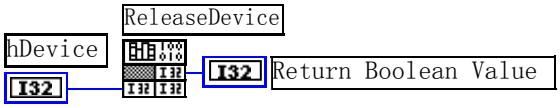
Visual Basic:

Declare Function ReleaseDevice Lib "PCH8603W1" (ByVal hDevice As Long) As Boolean

Delphi:

Function ReleaseDevice(hDevice : LongInt):Boolean; StdCall; External 'PCH8603W1' 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。

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，再一次执行这个函数前，必须执行一次ReleaseDevice函数，以释放由CreateDevice占用的系统软硬件资源，如系统内存等。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。

- **取得当前设备对象句柄指向的设备所在的设备 ID**

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL GetDeviceCurrentID(HANDLE hDevice, PLONG DeviceLgcID, PLONG DevicePhysID)
```

Visual Basic:

```
Declare Function GetDeviceCurrentID Lib "PCH8603W1" (ByVal hDevice As Long, _
                                                     ByRef DeviceLgcID As Long,_
                                                     ByRef DevicePhysID As Long) As Boolean
```

Delphi:

```
Function GetDeviceCurrentID(hDevice : LongInt;
                            DeviceLgcID : PLong;
                            DevicePhysID :PLong): Boolean;
                            StdCall; External 'PCH8603W1' Name ' GetDeviceCurrentID';
```

LabView:

请参考相关演示程序。

功能：取得指定设备对象所代表的设备在设备链中的物理设备 ID 号和逻辑 ID 号。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

DeviceLgcID 返回设备逻辑ID。

DevicePhysID 返回设备物理ID。

返回值：若成功，则返回 TRUE, 否则返回 FALSE。

相关函数：CreateDevice ReleaseDevice

- **列表计算机系统中的所有的该PCI设备**

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL ListDeviceDlg(HANDLE hDevice)
```

Visual Basic:

```
Declare Function ListDeviceDlg Lib "PCH8603W1" (ByVal hDevice As Long) As Boolean
```

Delphi:

```
Function ListDeviceDlg (hDevice : LongInt): Boolean;StdCall; External 'PCH8603W1' Name ' ListDeviceDlg' ;
```

LabView:

请参考相关演示程序。

功能：列表系统当中的所有的该PCI设备。

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。
相关函数：CreateDevice ReleaseDevice

♦**取得设备总台数**

函数原型：

Visual C++ & C++Builder&LabWindow:
```
Long GetDeviceCount(HANDLE hDevice)
```
Visual Basic:
```
Declare Function GetDeviceCount Lib "PCH8603W1" (ByVal hDevice As Long) As Long
```
Delphi:
```
Function GetDeviceCount (hDevice : LongInt): LongInt;StdCall; External 'PCH8603W1' Name ' GetDeviceCount' ;
```
LabView:
请参考相关演示程序。

功能：取得设备总台数。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
返回值：返回设备的数量
相关函数：CreateDevice ReleaseDevice

第三节、AD数据读取函数

♦**初始化设备**

函数原型：

Visual C++ & C++Builder&LabWindow:
```
BOOL InitDeviceAD(HANDLE hDevice,
                  PPCH8603W_PARA_AD pADPara)
```
Visual Basic:
```
Declare Function InitDeviceAD Lib "PCH8603W1" (ByVal hDevice As Long, _
                                               ByRef  pADPara As PCH8603W_PARA_AD) As Boolean
```
Delphi:
```
Function InitDeviceAD (hDevice : LongInt;
          pADPara : PPCH8603W_PARA_AD): Boolean;StdCall; External 'PCH8603W1' Name ' InitDeviceAD' ;
```
LabView:
请参考相关演示程序。

功能：初始化设备。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
pADPara 硬件参数，它仅在此函数中决定硬件状态
返回值：当返回TRUE后，设备即准备就绪
相关函数：CreateDevice ReleaseDevice

♦ **启动设备**

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL StartDeviceAD(HANDLE hDevice)
```

Visual Basic:

```
Declare Function StartDeviceAD Lib "PCH8603W1" (ByVal hDevice As Long) As Boolean
```

Delphi:

```
Function StartDeviceAD(hDevice : LongInt): Boolean;StdCall; External 'PCH8603W1' Name ' StartDeviceAD' ;
```

LabView:

请参考相关演示程序。

功能：在初始化之后，启动设备。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

返回值：当返回TRUE后，设备启动成功

相关函数：CreateDevice　　ReleaseDevice

♦ **读取设备上的AD数据(程序非空方式)**

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL ReadDeviceProAD_Npt(HANDLE hDevice,
                         ULONG ADBuffer[],
                         LONG nReadSizeWords,
                         PLONG nRetSizeWords);
```

Visual Basic:

```
Declare Function  ReadDeviceProAD_Npt Lib "PCH8603W1" (ByVal hDevice As Long, _
                                                       ByRef ADBuffer As Long, _
                                                       ByVal nReadSizeWords As Long, _
                                                       ByRef nRetSizeWrods As Long) As Boolean
```

Delphi:

```
Function  ReadDeviceProAD_Npt(hDevice : LongInt;
          ADBuffer:PLongWord;
          nReadSizeWords:LongInt;
          nRetSizeWords:PLong): Boolean;StdCall; External 'PCH8603W1' Name '  ReadDeviceProAD_Npt' ;
```

LabView:

请参考相关演示程序。

功能：当AD标志有效时，用此函数读取设备上的AD数据(程序非空方式)。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

ADBuffer 接受原始AD数据的用户缓冲区

nReadSizeWords 相对于偏位点后读入的数据长度(字)

nRetSizeWords 返回实际读取的长度(字)

返回值：若成功，则返回TRUE, 否则返回 FALSE。

相关函数： CreateDevice ReleaseDevice

♦ **在AD采样过程中取得设备的各种状态**

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL GetDevStatusAD(HANDLE hDevice,
                    PPCH8603W_STATUS_AD pADStatus)
```

Visual Basic:

```
Declare Function  GetDevStatusAD Lib "PCH8603W1" (ByVal hDevice As Long, _
                                                  ByRef pADStatus As PCH8603W_STATUS_AD) As Boolean
```

Delphi:

```
Function  GetDevStatusAD(hDevice : LongInt;
    pADStatus :PPCH8603W_STATUS_AD): Boolean;StdCall; External 'PCH8603W1' Name ' GetDevStatusAD' ;
```

LabView:

请参考相关演示程序。

功能：在AD采样过程中取得设备的各种状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pADStatus AD的各种信息结构体

返回值：若成功，则返回TRUE, 否则返回 FALSE。

相关函数： CreateDevice ReleaseDevice

♦ **在AD采读取设备上的AD数据(程序半满方式)**

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL ReadDeviceProAD_Half(HANDLE hDevice,
                          ULONG ADBuffer[],
                          LONG nReadSizeWords,
                          PLONG nRetSizeWords)
```

Visual Basic:

```
Declare Function  ReadDeviceProAD_Half  Lib "PCH8603W1" (ByVal hDevice As Long, _
                                                          ByRef ADBuffer As Long, _
                                                          ByVal nReadSizeWords As Long, _
                                                          ByRef  nRetSizeWords As Long) As Boolean
```

Delphi:

```
Function  ReadDeviceProAD_Half(hDevice : LongInt;
        ADBuffer : PLongWord;
        nReadSizeWords : LongInt;
        nRetSizeWords : PLong): Boolean;StdCall; External 'PCH8603W1' Name ' ReadDeviceProAD_Half' ;
```

LabView:

请参考相关演示程序。

功能：当AD标志有效时，用此函数读取设备上的AD数据(程序半满方式)。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

ADBuffer 接受原始AD数据的用户缓冲区

nReadSizeWords 相对于偏位点后读入的数据长度(字)
nRetSizeWords 返回实际读取的长度(字)
返回值：若成功，则返回TRUE, 否则返回 FALSE。
相关函数： CreateDevice ReleaseDevice

♦ **DMA读AD数据**

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL ReadDeviceDmaAD(HANDLE hDevice,
                     PULONG pADBuffer,
                     ULONG nReadSizeWords,
                     PLONG nRetSizeWords)
```

Visual Basic:

```
Declare Function  ReadDeviceDmaAD  Lib "PCH8603W1" (ByVal hDevice As Long, _
                                                     ByRef ADBuffer As Long, _
                                                     ByVal nReadSizeWords As Long, _
                                                     ByRef  nRetSizeWords As Long) As Boolean
```

Delphi:

```
Function  ReadDeviceDmaAD(hDevice : LongInt;
          ADBuffer : PLongWord;
          nReadSizeWords : LongInt;
          nRetSizeWords : PLong): Boolean;StdCall; External 'PCH8603W1' Name 'ReadDeviceDmaAD' ;
```

LabView:

请参考相关演示程序。

功能：当AD标志有效时，用此函数读取设备上的AD数据(程序半满方式)。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
ADBuffer 将用于接受数据的用户缓冲区
nReadSizeWords 读入的数据长度
nRetSizeWords 返回实际读取的数据长度
返回值：若成功，则返回TRUE, 否则返回 FALSE。
相关函数： CreateDevice ReleaseDevice

♦ **暂停设备**

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL StopDeviceAD(HANDLE hDevice)
```

Visual Basic:

```
Declare Function  StopDeviceAD Lib "PCH8603W1" (ByVal hDevice As Long) As Boolean
```

Delphi:

```
Function  StopDeviceAD(hDevice : LongInt;): Boolean;StdCall; External 'PCH8603W1' Name 'StopDeviceAD' ;
```

LabView:

请参考相关演示程序。

功能：在启动设备之后，暂停设备。
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。

相关函数：CreateDevice ReleaseDevice

♦ **关闭AD设备**

函数原型：

Visual C++ & C++Builder&LabWindow:

BOOL ReleaseDeviceAD(HANDLE hDevice)

Visual Basic:

Declare Function ReleaseDeviceAD Lib "PCH8603W1" (ByVal hDevice As Long) As Boolean

Delphi:

Function ReleaseDeviceAD(hDevice : LongInt;): Boolean;StdCall; External 'PCH8603W1' Name 'ReleaseDeviceAD' ;

LabView:

请参考相关演示程序。

功能：关闭AD设备, 禁止传输, 且释放资源。
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。

相关函数：CreateDevice ReleaseDevice

第四节、数DA数据读取函数

♦ **设置触发电平**

函数原型：

Visual C++ & C++Builder&LabWindow:

BOOL SetDevTrigLevelDA(HANDLE hDevice,float fTrigLevelVolt)

Visual Basic:

Declare Function SetDevTrigLevelDA Lib "PCH8603W1" (ByVal hDevice As Long, _
ByVal fTrigLevelVolt As Single) As Boolean

Delphi:

Function SetDevTrigLevelDA (hDevice : LongInt;
fTrigLevelVolt :Single): Boolean;StdCall; External 'PCH8603W1' Name 'SetDevTrigLevelDA ' ;

LabView:

请参考相关演示程序。

功能：设置触发电平(mV)。
参数：

hDevice 设备对象句柄，它应由CreateDevice创建。
fTrigLevelVolt 触发电平, 范围为-10000 – +10000mV
返回值：若成功，则返回TRUE, 否则返回 FALSE。

相关函数：CreateDevice ReleaseDevice

♦ **动态改变采样频率**

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL SetDevFrequencyDA(HANDLE hDevice,
                       LONG nFrequency,
                       int nDAChannel)
```

Visual Basic:

```
Declare Function  SetDevFrequencyDA Lib "PCH8603W1" (ByVal hDevice As Long, _
                                                     ByVal nFrequency As Long, _
                                                     ByVal nDAChannel As Long) As Boolean
```

Delphi:

```
Function  SetDevFrequencyDA (hDevice : LongInt;
          nFrequency : Single;
          nDAChannel : LongInt): Boolean;StdCall; External 'PCH8603W1' Name   'SetDevFrequencyDA '
```

LabView:

请参考相关演示程序。

功能：在DA转换过程中，动态改变采样频率

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nFrequency DA采样频率(Hz)

nDAChannel DA通道号[0, 1]

返回值：若成功，则返回TRUE, 否则返回 FALSE。

相关函数：CreateDevice　　ReleaseDevice

♦ **初始化设备**

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL InitDeviceDA(HANDLE hDevice,
                  LONG SegmentCount,
                  PCH8603W_SEGMENT_INFO SegmentInfo[],
                  PPCH8603W_PARA_DA pDAPara,
                  int nDAChannel)
```

Visual Basic:

```
Declare Function  InitDeviceDA Lib "PCH8603W1" (ByVal hDevice As Long, _
                                                ByVal SegmentCount As Long, _
                                                ByRef SegmentInfo() As PCH8603W_SEGMENT_INFO, _
                                                ByRef  pDAPara As PCH8603W_PARA_DA, _
                                                ByVal nDAChannel As Long) As Boolean
```

Delphi:

```
Function  InitDeviceDA (hDevice : LongInt;
          SegmentCount : LongInt;
          SegmentInfo: Array[] 0f PCH8603W_SEGMENT_INFO;
          pDAPara : PPCH8603W_PARA_DA;
          nDAChannel : LongInt): Boolean;StdCall; External 'PCH8603W1' Name   ' InitDeviceDA '
```

LabView:

请参考相关演示程序。

功能：初始化设备，当返回TRUE后，设备即准备就绪
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
SegmentCount 总工作段数[1，65536]
SegmentInfo 段信息集合
pDAPara 硬件参数，它仅在此函数中决定硬件状态
nDAChannel DA通道号[0，1]
返回值：若成功，则返回TRUE，否则返回 FALSE。
相关函数：CreateDevice ReleaseDevice

♦将DA数据写入板载RAM中(程序方式)

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL WriteDeviceBulkDA(HANDLE hDevice,
                       SHORT DABuffer[],
                       LONG nWriteOffsetWords,
                       LONG nWriteSizeWords,
                       PLONG nRetSizeWords,
                       int nDAChannel)
```

Visual Basic:

```
Declare Function  WriteDeviceBulkDA Lib "PCH8603W1" (ByVal hDevice As Long, _
                                                     ByRef DABuffer As Integer, _
                                                     ByVal  nWriteOffsetWords As Long, _
                                                     ByVal  nWriteSizeWords As  Long, _
                                                     ByRef  nRetSizeWords As  Long, _
                                                     ByVal nDAChannel As Long) As Boolean
```

Delphi:

```
Function  WriteDeviceBulkDA (hDevice : LongInt;
          DABuffer : PSmallInt;
          nWriteOffsetWords :Long;
          nWriteSizeWords : Long;
          nRetSizeWords : PLong;
          nDAChannel :LongInt): Boolean;StdCall; External 'PCH8603W1' Name   ' WriteDeviceBulkDA  '
```

LabView:
请参考相关演示程序。

功能：初始在DA输出前，用此函数将DA数据写入板载RAM中(程序方式)
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
DABuffer 携带原始DA数据的用户缓冲区
nWriteOffsetWords 相对于该通道物理RAM零位置的偏移点(字)
nWriteSizeWords 相对于偏位点后写入的数据长度(字)
nRetSizeWords 返回实际写出的长度(字)
nDAChannel DA通道号[0，1]

返回值：若成功，则返回TRUE, 否则返回 FALSE。

相关函数：CreateDevice　　ReleaseDevice

♦ **将板载RAM中的数据读回计算机(程序方式)**

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL ReadDeviceBulkDA(HANDLE hDevice,
                      SHORT DABuffer[],
                      LONG nReadOffsetWords,
                      LONG nReadSizeWords,
                      PLONG nRetSizeWords,
                      int nDAChannel)
```

Visual Basic:

```
Declare Function  ReadDeviceBulkDA Lib "PCH8603W1" (ByVal hDevice As Long, _
                                                    ByRef DABuffer As Integer, _
                                                    ByVal  nReadOffsetWords As Long, _
                                                    ByVal  nReadSizeWords As  Long, _
                                                    ByRef  nRetSizeWords As  Long, _
                                                    ByVal nDAChannel As Long) As Boolean
```

Delphi:

```
Function  ReadDeviceBulkDA (hDevice : LongInt;
          DABuffer : PSmallInt;
          nReadOffsetWords:Long;
          nReadSizeWords: Long;
          nRetSizeWords : PLong;
          nDAChannel :LongInt): Boolean;StdCall; External 'PCH8603W1' Name    ' ReadDeviceBulkDA'
```

LabView:

请参考相关演示程序。

功能：用此函数将板载RAM中的数据读回计算机(程序方式)

参数：

hDevice　设备对象句柄，它应由CreateDevice创建。

DABuffer　携带原始DA数据的用户缓冲区

nReadOffsetWords 相对于该通道物理RAM零位置的偏移点(字)

nReadSizeWords 相对于偏位点后写入的数据长度(字)

nRetSizeWords 返回实际读出的长度(字)

nDAChannel　DA通道号[0, 1]

返回值：若成功，则返回TRUE, 否则返回 FALSE。

相关函数：CreateDevice　　ReleaseDevice

♦ **使能DA设备**

函数原型：

Visual C++ & C++Builder&LabWindow:

BOOL EnableDeviceDA(HANDLE hDevice,int nDAChannel)

Visual Basic:

```
Declare Function  EnableDeviceDA Lib "PCH8603W1" (ByVal hDevice As Long, _
                                                  ByVal nDAChannel As Long) As Boolean
```

Delphi:

```
Function  EnableDeviceDA (hDevice : LongInt;
                          nDAChannel :LongInt): Boolean;StdCall; External 'PCH8603W1' Name ' EnableDeviceDA'
```

LabView:

请参考相关演示程序。

功能：用此函数将板载RAM中的数据读回计算机(程序方式)

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nDAChannel DA通道号[0, 1]

返回值：若成功，则返回TRUE, 否则返回 FALSE。

相关函数：CreateDevice ReleaseDevice

♦产生软件触发事件

函数原型：

Visual C++ & C++Builder&LabWindow:

BOOL SetDeviceTrigDA(HANDLE hDevice,BOOL bSetSyncTrig,int nDAChannel)

Visual Basic:

```
Declare Function  SetDeviceTrigDA Lib "PCH8603W1" (ByVal hDevice As Long, _
                                                   ByVal  bSetSyncTrig As Boolean, _
                                                   ByVal nDAChannel As Long) As Boolean
```

Delphi:

```
Function  SetDeviceTrigDA (hDevice : LongInt;
                           bSetSyncTrig :Boolean
                           nDAChannel :LongInt): Boolean;StdCall; External 'PCH8603W1' Name ' SetDeviceTrigDA'
```

LabView:

请参考相关演示程序。

功能：当设备使能允许后，产生软件触发事件（只有触发源为软件触发时有效）

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bSetSyncTrig 是否置同步触发

nDAChannel DA通道号[0, 1]

返回值：若成功，则返回TRUE, 否则返回 FALSE。

相关函数：CreateDevice ReleaseDevice

♦采样过程中取得设备的各种状态

函数原型：

Visual C++ & C++Builder&LabWindow:

BOOL GetDevStatusDA(HANDLE hDevice,PPCH8603W_STATUS_DA pDAStatus,int nDAChannel)

Visual Basic:

```
Declare Function  GetDevStatusDA Lib "PCH8603W1" (ByVal hDevice As Long,  _
                                                  ByRef  pDAStatus As PCH8603W_STATUS_DA, _
                                                  ByVal nDAChannel As Long) As Boolean
```

Delphi:

```
Function  GetDevStatusDA (hDevice : LongInt;
                          pDAStatus :PPCH8603W_STATUS_DA
                          nDAChannel :LongInt): Boolean;StdCall; External 'PCH8603W1' Name ' GetDevStatusDA '
```

LabView:

请参考相关演示程序。

功能：在DA采样过程中取得设备的各种状态,返回值表示函数是否成功

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pDAStatus DA的各种信息结构体

nDAChannel DA通道号[0, 1]

返回值：若成功，则返回TRUE,否则返回 FALSE。

相关函数：CreateDevice ReleaseDevice

♦禁止设备

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL DisableDeviceDA(HANDLE hDevice,int nDAChannel)
```

Visual Basic:

```
Declare Function  DisableDeviceDA Lib "PCH8603W1" (ByVal hDevice As Long, _
                                                   ByVal nDAChannel As Long) As Boolean
```

Delphi:

```
Function  DisableDeviceDA (hDevice : LongInt;
                           nDAChannel :LongInt): Boolean;StdCall; External 'PCH8603W1' Name ' DisableDeviceDA'
```

LabView:

请参考相关演示程序。

功能：在使设备之后，禁止设备

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nDAChannel DA通道号[0, 1]

返回值：若成功，则返回TRUE,否则返回 FALSE。

相关函数：CreateDevice ReleaseDevice

♦禁止DA设备，且释放资源

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL ReleaseDeviceDA(HANDLE hDevice,int nDAChannel)
```

Visual Basic:

```
Declare Function  ReleaseDeviceDA Lib "PCH8603W1" (ByVal hDevice As Long,  _
```

```
                                                    ByVal nDAChannel As Long) As Boolean
```

Delphi:

```
Function  ReleaseDeviceDA (hDevice : LongInt;
                    nDAChannel :LongInt): Boolean;StdCall; External 'PCH8603W1' Name ' ReleaseDeviceDA'
```

LabView:

请参考相关演示程序。

功能：禁止DA设备，且释放资源

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nDAChannel DA通道号[0, 1]

返回值：若成功，则返回TRUE, 否则返回 FALSE。

相关函数：CreateDevice ReleaseDevice

♦**DA单点输出**

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL WriteDeviceOneDA(HANDLE hDevice,ULONG  ulDataCode,int nDAChannel)
```

Visual Basic:

```
Declare Function  WriteDeviceOneDA Lib "PCH8603W1" (ByVal hDevice As Long, _
                                                    ByVal ulDataCode As Long
                                                    ByVal nDAChannel As Long) As Boolean
```

Delphi:

```
Function  WriteDeviceOneDA (hDevice : LongInt;
                    ulDataCode:LongWord
                    nDAChannel :LongInt): Boolean;StdCall; External 'PCH8603W1' Name ' WriteDeviceOneDA '
```

LabView:

请参考相关演示程序。

功能：DA单点输出

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

ulDataCode 输入码值(0-4095)

nDAChannel DA通道号[0, 1]

返回值：若成功，则返回TRUE, 否则返回 FALSE。

相关函数：CreateDevice ReleaseDevice

第五节、数字I/O输入输出函数

♦**取得开关量状态**

函数原型：

Visual C++ & C++Builder&LabWindow:

```
BOOL GetDeviceDI(HANDLE hDevice,
                    BYTE bDISts[8])
```

Visual Basic:

```
Declare Function  GetDeviceDI Lib "PCH8603W1" (ByVal hDevice As Long, _
```

ByRef bDISts As Byte) As Boolean

Delphi:

Function GetDeviceDI (hDevice : LongInt;
bDISts : PByte): Boolean;StdCall; External 'PCH8603W1' Name ' GetDeviceDI '

LabView:

请参考相关演示程序。

功能：取得开关量状态

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDISts 开关输入状态(注意：必须定义为个字节元素的数组)

返回值：若成功，则返回TRUE, 否则返回 FALSE。

相关函数：CreateDevice ReleaseDevice

♦输出开关量状态

函数原型：

Visual C++ & C++Builder&LabWindow:

BOOL SetDeviceDO(HANDLE hDevice,
BYTE bDOSts[8])

Visual Basic:

Declare Function SetDeviceDO Lib "PCH8603W1" (ByVal hDevice As Long, _
ByRef bDOSts As Byte) As Boolean

Delphi:

Function SetDeviceDO (hDevice : LongInt;
bDOSts : PByte): Boolean;StdCall; External 'PCH8603W1' Name ' SetDeviceDO '

LabView:

请参考相关演示程序。

功能：输出开关量状态

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDOSts 开关输出状态(注意：必须定义为个字节元素的数组)

返回值：若成功，则返回TRUE, 否则返回 FALSE。

相关函数：CreateDevice ReleaseDevice

♦获取输出开关量状态

函数原型：

Visual C++ & C++Builder&LabWindow:

BOOL RetDeviceDO(HANDLE hDevice,
BYTE bDOSts[8])

Visual Basic:

Declare Function RetDeviceDO Lib "PCH8603W1" (ByVal hDevice As Long, _
ByRef bDOSts As Byte) As Boolean

Delphi:

```
Function  RetDeviceDO(hDevice : LongInt;
bDOSts : PByte): Boolean;StdCall; External 'PCH8603W1' Name ' RetDeviceDO '
```

LabView:

请参考相关演示程序。

功能：获取输出开关量状态

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDOSts 开关输出状态(注意：必须定义为个字节元素的数组)

返回值：若成功，则返回TRUE, 否则返回 FALSE。

相关函数：CreateDevice ReleaseDevice

第三章 硬件参数结构

第一节、AD硬件参数（_PCH8603W_PARA_AD）

Visual C++ & C++Builder&LabWindow:

```
typedef struct _PARA_AD
{
        LONG ADMode;          // AD模式选择(连续/分组方式)
        LONG FirstChannel;    // 首通道[0,15]
        LONG LastChannel;     // 末通道[0,15],要求末通道必须大于或等于首通道
        LONG Frequency;       // 采集频率,单位为Hz, [1, 500000]
        LONG GroupInterval;   // 分组时的组间间隔(单位：微秒)[1, 419430]
        LONG LoopsOfGroup;    // 组内循环次数[1, 255]
        LONG Gains;                     // 增益设置
        LONG InputRange;                // 模拟量输入量程范围
        LONG TriggerMode;     // 触发模式选择
        LONG TriggerType;               // 触发类型选择(边沿触发/脉冲触发)
        LONG TriggerDir;                // 触发方向选择(正向/负向触发)
        LONG TriggerSource;             // 触发源选择
        LONG TrigLevelVolt;             // 触发电平(±mV)
        LONG TrigWindow;                // 触发灵敏窗[1, 65535], 单位纳秒
        LONG ClockSource;               // 时钟源选择(内/外时钟源)
        LONG GroundingMode;             // 接地方式（单端或双端选择）
} PCH8603W_PARA_AD, *PPCH8603W_PARA_AD;
```

Visual Basic:

```
Type  _PARA_AD
LONG ADMode;          ‘ AD模式选择(连续/分组方式)
        ByVal FirstChannel As Long    ‘ 首通道[0,15]
        ByVal LastChannel As Long     ‘ 末通道[0,15],要求末通道必须大于或等于首通道
        ByVal Frequency As Long       ‘采集频率,单位为Hz, [1, 500000]
        ByVal GroupInterval As Long   ‘分组时的组间间隔(单位：微秒)[1, 419430]
        ByVal LoopsOfGroup As Long      ‘组内循环次数[1, 255]
        ByVal Gains As Long                   ‘增益设置
        ByVal InputRange As Long              ‘模拟量输入量程范围
        ByVal TriggerMode As Long     ‘ 触发模式选择
        ByVal TriggerType As Long             ‘ 触发类型选择(边沿触发/脉冲触发)
        ByVal TriggerDir As Long              ‘ 触发方向选择(正向/负向触发)
        ByVal TriggerSource As Long           ‘ 触发源选择
```

```
        ByVal TrigLevelVolt As Long          ‘触发电平(±mV)
        ByVal TrigWindow As Long             ‘ 触发灵敏窗[1, 65535], 单位纳秒
        ByVal ClockSource As Long            ‘时钟源选择(内/外时钟源)
        ByVal GroundingMode As Long          ‘ 接地方式（单端或双端选择）
End Type
```

Delphi:

```
Type
  _PPARA_AD = ^ _PARA_AD ; // 指针类型结构
  _PARA_AD  = record // 标记为记录型

        FirstChannel  : LongInt; // 首通道[0,15]
        LastChannel   : LongInt; // 末通道[0,15],要求末通道必须大于或等于首通道
        Frequency   : LongInt ;//采集频率,单位为Hz, [1, 500000]
        GroupInterval   : LongInt; //分组时的组间间隔(单位：微秒)[1, 419430]
        LoopsOfGroup   : LongInt;          //组内循环次数[1, 255]
        Gains   : LongInt;                 //增益设置
        InputRange   : LongInt;            //模拟量输入量程范围
        TriggerMode   : LongInt ;  //触发模式选择
        TriggerType   : LongInt; // 触发类型选择(边沿触发/脉冲触发)
        TriggerDir   : LongInt;    // 触发方向选择(正向/负向触发)
        TriggerSource   : LongInt;         // 触发源选择
        TrigLevelVolt   : LongInt;         //触发电平(±mV)
        TrigWindow   : LongInt;            // 触发灵敏窗[1, 65535], 单位纳秒
        ClockSource  : LongInt;            //时钟源选择(内/外时钟源)
        GroundingMode  : LongInt ;         //接地方式（单端或双端选择）
End
```

LabView:

请参考其演示程序。

硬件参数说明：此结构主要用于AD采样的实际硬件参数。

ADMode工作模式选择。它的其选项值如下表：

常量名	常量值	功能定义
PCH8603W_ADMODE_SEQUENCE	0X00	连续采样
PCH8603W_ADMODE_GROUP	0X01	分组采样

InputRange模拟量输入范围：

常量名	常量值	功能定义
PCH8603W_INPUT_N10000_P10000mV	0X00	±10000mV
PCH8603W_INPUT_N5000_P5000mV	0X01	±5000mV
PCH8603W_INPUT_N2500_P2500mV	0X02	±2500mV
PCH8603W_INPUT_0_P10000mV	0X03	0~10000mV

Gains使用的硬件增益选项：

常量名	常量值	功能定义
PCH8603W_GAINS_1MULT	0X00	1倍增益(使用AD8251放大器)
PCH8603W_GAINS_2MULT	0X01	2倍增益(使用AD8251放大器)
PCH8603W_GAINS_4MULT	0X02	4倍增益(使用AD8251放大器)
PCH8603W_GAINS_8MULT	0X03	8倍增益(使用AD8251放大器)

TriggerMode成员变量所使用触发模式选项：

常量名	常量值	功能定义
PCH8603W_TRIGMODE_SOFT	0X00	软件触发(属于内触发)
PCH8603W_TRIGMODE_POST	0X01	硬件后触发(属于外触发)

TriggerSource触发源信号所使用的选项：

常量名	常量值	功能定义
PCH8603W_TRIGSRC_DTR_AD	0X00	选择外部DTR作为触发源
PCH8603W_TRIGSRC_ATR_AD	0X01	选择外部ATR作为触发源

TriggerType触发类型所使用的选项：

常量名	常量值	功能定义
PCH8603W_TRIGTYPE_EDGE	0X00	边沿触发
PCH8603W_TRIGTYPE_PULSE	0X01	脉冲触发(电平)

bClockOutput成员变量所使用内部和外部时钟源选项：

常量名	常量值	功能定义
PCH8603W_CLOCKOUT_DISABLE	0X00	禁止本卡上的自带时钟向外输出
PCH8603W_CLOCKOUT_ENABLE	0X01	允许本卡上的自带时钟向外输出

GroundingMode使用的模拟信号接地方式选项：

常量名	常量值	功能定义
PCH8603W_GNDMODE_SE	0X00	单端方式(SE:Single end)
PCH8603W_GNDMODE_DI	0X01	双端方式(DI:Differential)

第二节、AD采样的实际硬件参数

Visual C++ & C++Builder&LabWindow:

```
typedef struct _STATUS_AD
{
LONG bNotEmpty;                 // 板载FIFO存储器的非空标志，=TRUE非空，= FALSE 空
LONG bHalf;                     // 板载FIFO存储器的半满标志，=TRUE半满以上,= FALSE 半满以下
LONG bDynamic_Overflow;  // 板载FIFO存储器的动态溢出标志，= TRUE已发生溢出，= FALSE未发生溢出
LONG bStatic_Overflow;   // 板载FIFO存储器的静态溢出标志，= TRUE已发生溢出，= FALSE 未发生溢出
LONG bConverting;        // AD是否正在转换，=TRUE:表示正在转换，=FALS表示转换完成
LONG bTriggerFlag;              // 触发标志, =TRUE表示触发事件发生，    =FALSE表示触发事件未发生
} STATUS_AD, *PSTATUS_AD;
```

Visual Basic:

```
Type _ STATUS_AD
ByVal bNotEmpty As Long                 ‘板载FIFO存储器的非空标志，=TRUE非空，= FALSE 空
ByVal bHalf As Long             ‘板载FIFO存储器的半满标志，=TRUE半满以上,= FALSE 半满以下
ByVal bDynamic_Overflow As Long
‘板载FIFO存储器的动态溢出标志，= TRUE已发生溢出，= FALSE未发生溢出
ByVal bStatic_Overflow As Long
‘板载FIFO存储器的静态溢出标志，= TRUE已发生溢出，= FALSE 未发生溢出
ByVal bConverting As Long ‘ AD是否正在转换，=TRUE:表示正在转换，=FALS表示转换完成
ByVal bTriggerFlag As Long      ‘ 触发标志, =TRUE表示触发事件发生，    =FALSE表示触发事件未发生
End Type
```

Delphi:

```
Type
  _PSTATUS_AD= ^ _STATUS_AD; // 指针类型结构
  _STATUS_AD = record // 标记为记录型

        bNotEmpty : LongInt;    // 板载FIFO存储器的非空标志，=TRUE非空，= FALSE 空
        bHalf : LongInt; // 板载FIFO存储器的半满标志，=TRUE半满以上,= FALSE 半满以下
        bDynamic_Overflow : LongInt;
        // 板载FIFO存储器的动态溢出标志，= TRUE已发生溢出，= FALSE未发生溢出
        bStatic_Overflow : LongInt;
        // 板载FIFO存储器的静态溢出标志，= TRUE已发生溢出，= FALSE 未发生溢出
        bConverting : LongInt;    // AD是否正在转换，=TRUE:表示正在转换，=FALS表示转换完成
        bTriggerFlag : LongInt;  // 触发标志, =TRUE表示触发事件发生，    =FALSE表示触发事件未发生
End
```

LabView:

请参考其演示程序。

Visual C++ & C++Builder&LabWindow:

```
typedef struct STATUS_DMA
{
        LONG iCurSegmentID;  // 当前段缓冲ID,表示DMA正在传输的缓冲区段
        LONG bSegmentSts[PCH8603W_MAX_SEGMENT_COUNT];
        // 各个缓冲区的新旧状态,=1表示该相应缓冲区数据为新,否则为旧
        LONG bBufferOverflow;// 返回溢出状态
} PCH8603W_STATUS_DMA, *PPCH8603W_STATUS_DMA;
```

Visual Basic:

```
Type  STATUS_DMA
        ByVal iCurSegmentID As Long;  ‘ 当前段缓冲ID,表示DMA正在传输的缓冲区段
        ByVal bSegmentSts(0 to PCH8603W_MAX_SEGMENT_COUNT-1) As Long;
        ‘各个缓冲区的新旧状态,=1表示该相应缓冲区数据为新,否则为旧
        ByVal bBufferOverflow As Long;          ‘返回溢出状态
End Type
```

Delphi:

```
Type
  PSTATUS_DMA= ^ STATUS_DMA; // 指针类型结构
  STATUS_DMA = record // 标记为记录型

        iCurSegmentID : LongInt;          // 当前段缓冲ID,表示DMA正在传输的缓冲区段
        bSegmentSts:Array[0..PCH8603W_MAX_SEGMENT_COUNT-1] 0f  LongInt;
        // 各个缓冲区的新旧状态,=1表示该相应缓冲区数据为新,否则为旧
        bBufferOverflow : LongInt;;       // 返回溢出状态
End
```

LabView:

请参考其演示程序。

第三节、各段信息

Visual C++ & C++Builder&LabWindow:

```
typedef struct  _SEGMENT_INFO
{
        LONG SegLoopCount;              // 每个段在大循环中的小循环次数,取值为[1, 16777215]
```

```
        LONG SegmentSize;              // 每个段在RAM中的长度(单位：字/点)
} SEGMENT_INFO, *PSEGMENT_INFO;
```

Visual Basic:

```
Type _SEGMENT_INFO
        ByVal SegLoopCount As Long;          ‘ 每个段在大循环中的小循环次数,取值为[1, 16777215]
        ByVal SegmentSize As Long;           ‘ 每个段在RAM中的长度(单位：字/点)
End Type
```

Delphi:

```
Type
  _PSEGMENT_INFO= ^ _SEGMENT_INFO; // 指针类型结构
  _SEGMENT_INFO = record // 标记为记录型
        SegLoopCount:LongInt;          // 每个段在大循环中的小循环次数,取值为[1, 16777215]
        SegmentSize:LongInt;           // 每个段在RAM中的长度(单位：字/点)
End
```

LabView:

请参考其演示程序。

第四节、DA的工作参数

Visual C++ & C++Builder&LabWindow:

```
        typedef struct  _PARA_DA
        {
                LONG OutputRange;              // 输出量程
                LONG Frequency;       // 点频率[0.010Hz, 1MHz], 为正数时单位Hz，为负数时单位为.001Hz
                LONG LoopCount;       // 整过RAM的大循环次数,=0:无限循环, =n:表示n次循环(1<n<32768)
                LONG TriggerMode;              // 触发模式选择
                LONG TriggerSource;            // 触发源选择
                LONG TriggerDir;               // 触发方向选择
                LONG bSingleOut;               // 是否单点输出
                LONG ClockSource;              // 时钟源选择
        } PARA_DA, *PPARA_DA;
```

Visual Basic:

```
Type  _PARA_DA
        ByVal OutputRange As Long              ‘ 输出量程
        ByVal Frequency As Long    ‘ 点频率[0.010Hz, 1MHz], 为正数时单位Hz，为负数时单位为.001Hz
        ByVal LoopCount As Long        ‘ 整过RAM的大循环次数,=0:无限循环, =n:表示n次循环(1<n<32768)
        ByVal TriggerMode As Long              ‘ 触发模式选择
        ByVal TriggerSource As Long            ‘ 触发源选择
        ByVal TriggerDir As Long               ‘触发方向选择
        ByVal bSingleOut As Long               ‘ 是否单点输出
        ByVal ClockSource As Long              ‘ 时钟源选择End Type
```

Delphi:

```
Type
  _PPARA_DA= ^_PARA_DA; // 指针类型结构
  _PARA_DA= record // 标记为记录型
                OutputRange:LongInt;           // 输出量程
                Frequency:LongInt;     // 点频率[0.010Hz, 1MHz], 为正数时单位Hz，为负数时单位为.001Hz
                LoopCount:LongInt;     // 整过RAM的大循环次数,=0:无限循环, =n:表示n次循环(1<n<32768)
                TriggerMode:LongInt;           // 触发模式选择
                TriggerSource:LongInt;         // 触发源选择
                TriggerDir:LongInt;            // 触发方向选择
```

```
        bSingleOut:LongInt;          // 是否单点输出
        ClockSource:LongInt;         // 时钟源选择
End
```

LabView:

请参考其演示程序。

OutputRange所使用的选项：

常量名	常量值	功能定义
PCH8603W_OUTPUT_0_P5000mV	0X00	0～5000mV
PCH8603W_OUTPUT_0_P10000mV	0X01	0～10000mV
PCH8603W_OUTPUT_N10000_P10000mV	0X02	±10000mV
PCH8603W_OUTPUT_N5000_P5000mV	0X03	±5000mV

TriggerMode成员变量触发模式选项：

常量名	常量值	功能定义
PCH8603W_TRIGMODE_SINGLE	0X00	单次触发
PCH8603W_TRIGMODE_CONTINUOUS	0X01	连续触发
PCH8603W_TRIGMODE_STEPED	0X02	单步触发
PCH8603W_TRIGMODE_BURST	0X03	紧急触发

TriggerSource成员变量触发源选项：

常量名	常量值	功能定义
PCH8603W_TRIGSRC_SOFT_DA	0X00	软件触发
PCH8603W_TRIGSRC_ATR_DA	0X01	ATR硬件模拟触发
PCH8603W_TRIGSRC_DTR_DA	0X02	DTR硬件数字触发

TriggerDir成员触发方向选项：

常量名	常量值	功能定义
PCH8603W_TRIGDIR_NEGATIVE	0X00	负向触发(低脉冲/下降沿触发)
PCH8603W_TRIGDIR_POSITIVE	0X01	正向触发(高脉冲/上升沿触发)
PCH8603W_TRIGDIR_POSIT_NEGAT	0X02	正负向触发(高/低脉冲或上升/下降沿触发)

AD硬件参数PCH8603W_PARA_AD中的ClockSource时钟源所使用的选项

DA硬件参数PCH8603W_PARA_DA中的ClockSource成员变量时钟源选项：

常量名	常量值	功能定义
PCH8603W_CLOCKSRC_IN	0X00	内部时钟
PCH8603W_CLOCKSRC_OUT	0X01	外部时钟

第五节、DA采样的实际硬件参数

Visual C++ & C++Builder&LabWindow:

```
typedef struct _STATUS_DA
{
        LONG bEnable; // DA使能启动标志, =TRUE表示DA已被使能, = FALSE表示DA被禁止
        LONG bTrigFlag;
          // 触发标志是否有效，=TRUE表示触点标有效，= FALSE表示无效（即触发点未到）
        LONG bConverting;      // DA是否正在转换，=TRUE:表示正在转换，= FALS表示转换完成
        LONG nCurSegNum;
    // 可读取的RAM段号，取值为[0, SegmentCount-1], (注SegmentCount为InitDeviceDA函数的参数)
        LONG nCurSegAddr;     // 可读取的RAM段地址
        LONG nCurLoopCount; // 当前总循环次数
```

```
        LONG nCurSegLoopCount; // 当前段循环次数
    } STATUS_DA, *PSTATUS_DA;
```

Visual Basic:

```
Type  _STATUS_DA
        ByVal bEnable As Long;
        ‘ DA使能启动标志, ＝TRUE表示DA已被使能, = FALSE表示DA被禁止
        ByVal bTrigFlag As Long
          ‘ 触发标志是否有效，=TRUE表示触点标有效，= FALSE表示无效（即触发点未到）
        ByVal bConverting As Long      ‘ DA是否正在转换，=TRUE:表示正在转换，= FALS表示转换完成
        ByVal nCurSegNum As Long
      ‘ 可读取的RAM段号，取值为[0, SegmentCount-1], (注SegmentCount为InitDeviceDA函数的参数)
        ByVal nCurSegAddr As Long     ‘可读取的RAM段地址
        ByVal nCurLoopCount As Long’ 当前总循环次数
        ByVal nCurSegLoopCount  As Long’当前段循环次数
```

Delphi:

```
Type
  _PSTATUS_DA= ^_STATUS_DA; // 指针类型结构
  _STATUS_DA= record // 标记为记录型
        bEnable:LongInt;// DA使能启动标志, ＝TRUE表示DA已被使能, = FALSE表示DA被禁止
        bTrigFlag:LongInt;
          // 触发标志是否有效，=TRUE表示触点标有效，= FALSE表示无效（即触发点未到）
        bConverting:LongInt;     // DA是否正在转换，=TRUE:表示正在转换，= FALS表示转换完成
        nCurSegNum:LongInt;
      // 可读取的RAM段号，取值为[0, SegmentCount-1], (注SegmentCount为InitDeviceDA函数的参数)
        nCurSegAddr:LongInt;    // 可读取的RAM段地址
        nCurLoopCount:LongInt; // 当前总循环次数
        nCurSegLoopCount:LongInt; // 当前段循环次数
End
```

LabView:

请参考其演示程序。