PISO-DA2/DA2U

Software Manual

[ver. 1.0.0, March 2012]

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

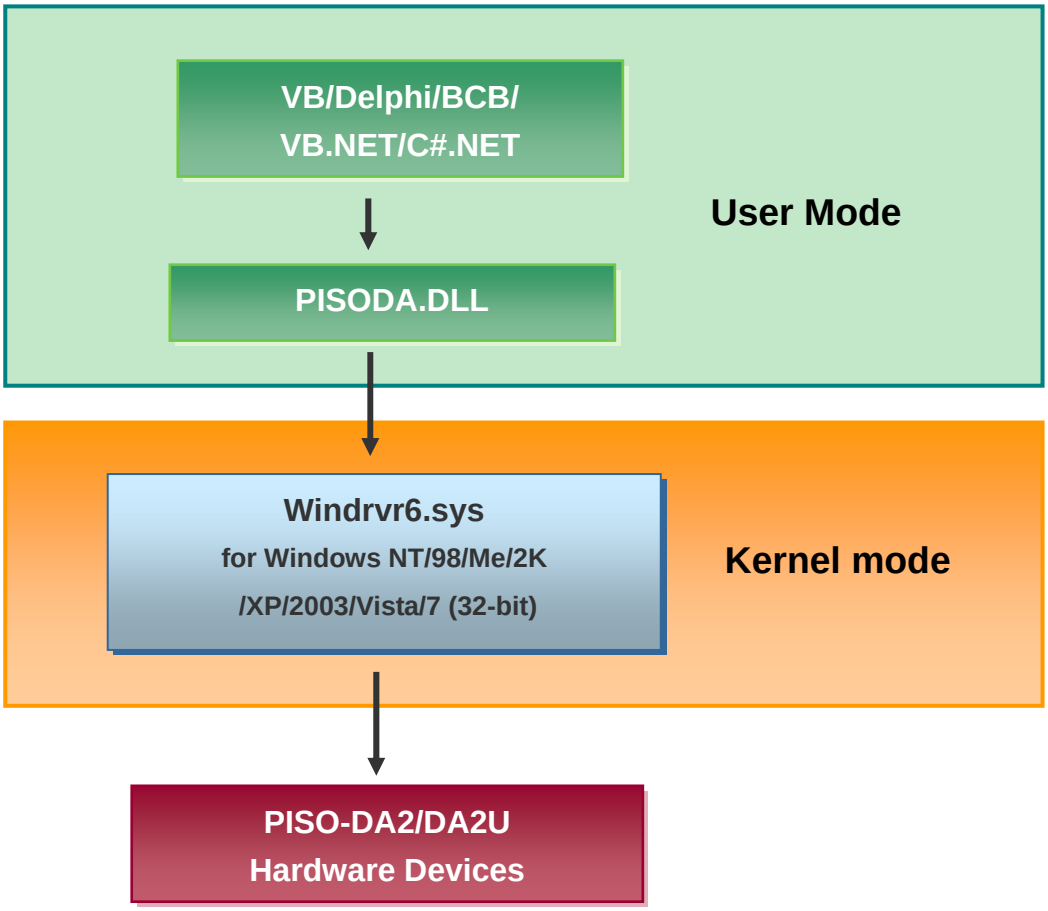
Trademark

Names are used for identification only and may be registered trademarks of their respective companies.

Table of Contents

1. DLL Function Description

The PISO-DA2 series classic DLL driver is the collection of function calls of the PISO-DA2/DA2U cards for Windows 98/Me/NT/2000 and 32-bit Windows XP/2003/Vista/7 system. The application structure is presented as following figure. The user application program developed by designate tools like VB, Delphi and Borland C++ Builder can call PISODA.DLL driver in user mode. And then DLL driver will bypass the function call to Windrvr6.sys to access the hardware system.



1.1 Reference

Please refer to the following user manuals:

- **PnPInstall.pdf:**
  Describes how to install the PnP (Plug and Play) driver for PCI card under Windows 95/98/2000/XP/2003/Vista/7(32-bit).
  http://ftp.icpdas.com.tw/pub/cd/iocard/pci/napdos/pci/manual/

- **SoftInst.pdf:**
  Describes how to install the software package under Windows 95/98/ 2000/XP/2003/Vista/7(32-bit).
  http://ftp.icpdas.com.tw/pub/cd/iocard/pci/napdos/pci/manual/

- **CallDll.pdf:**
  Describes how to call the DLL functions with VC++6, VB6, Delphi4 and Borland C++ Builder 4.
  http://ftp.icpdas.com.tw/pub/cd/iocard/pci/napdos/pci/manual/

- **ResCheck.pdf:**
  Describes how to check the resources I/O Port address, IRQ number and DMA number for add-on cards under Windows 95/98/2000/XP/2003/ Vista/7(32-bit).
  http://ftp.icpdas.com.tw/pub/cd/iocard/pci/napdos/pci/manual/

- **PISO-DA2_Hardware_Manual.pdf:**
  PISO-DA2/DA2U hardware manual.
  http://ftp.icpdas.com.tw/pub/cd/iocard/pci/napdos/pci/piso-da2/manual/

1.2 ErrorCode and ErrorString Table

For the most errors, it is recommended to check:

1. Does the device driver installs successful?
2. Does the card have plugged?
3. Does the card conflicts with other device?
4. Close other applications to free the system resources.
5. Try to use another slot to plug the card.
6. Restart your system to try again.

Error Code	Error ID	Error String
0	PISODA_NoError	OK
1	PISODA_ActiveBoardError	This board can not be activated.
2	PISODA_ExceedFindBoards	The board number exceeds the current total board number (N).
3	PISODA_DriverNoOpen	Kernel driver can't be found.
4	PISODA_BoardNoActive	The board is not activated
5	PISODA_WriteEEPROMError	Fail to write data to EEPROM.
6	PISODA_ParameterError	Parameter is null or out of range

2. Declaration Files

After the drivers are installed, the relevant demo programs, development libraries and declaration header files for the different development environments will be available in the following locations.

For detailed PISO-DA2 series classic Windows driver installed information, please refer to Quick Start Guide.
CD:\NAPDOS\PCI\PISO-DA2\Manual\QuickStart\
http://ftp.icpdas.com.tw/pub/cd/iocard/pci/napdos/pci/piso-da2/manual/quickstart/

The demo program is contained in:
CD:\NAPDOS\PCI\PISO-DA2\DLL\Demo\
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/piso-da2/dll/demo/

- BCB 3 → for Borland C++ Builder 3
  PISODA.H → Header files
  PISODA.LIB → Linkage library for BCB only
- Delphi5 → for Delphi 5
  PISODA.PAS → Declaration files
- VB6 → for Visual Basic 6
  PISODA.BAS → Declaration files
- VB.NET2005 → for VB.NET2005
  PISODA.vb → Visual Basic Source files
- CSharp2005 → for C#.NET2005
  PISODA.cs → Visual C# Source files

A list of available demo programs is as follows:

- DEMO1: Get cards information
- DEMO2: D/A output
- DEMO3: Read/Write from/to EEPROM and software calibration.
- DEMO4: Two cards D/A output

DEMO1: Get cards information

Following figure is the result for the demo_1 program. It can be applied to obtain the hardware information of the PISO-DA2/DA2U board.

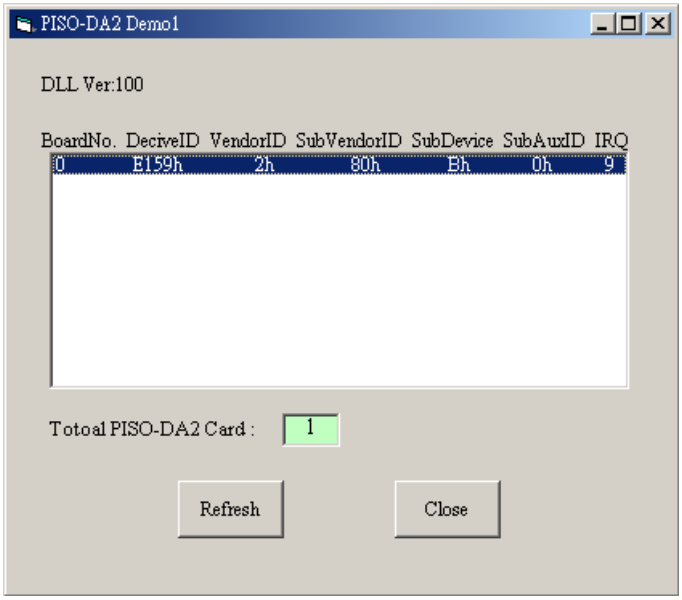


Figure 2-1: The demo_1 program

DEMO2: D/A output

This demo program can be applied to drive the voltage or current output for channel 1 and 2 independently. And the jumper setting statuses of the hardware are also displayed on the graphic interface for setting checking. For more information, please refer to the hardware register, which is presented in section 5.

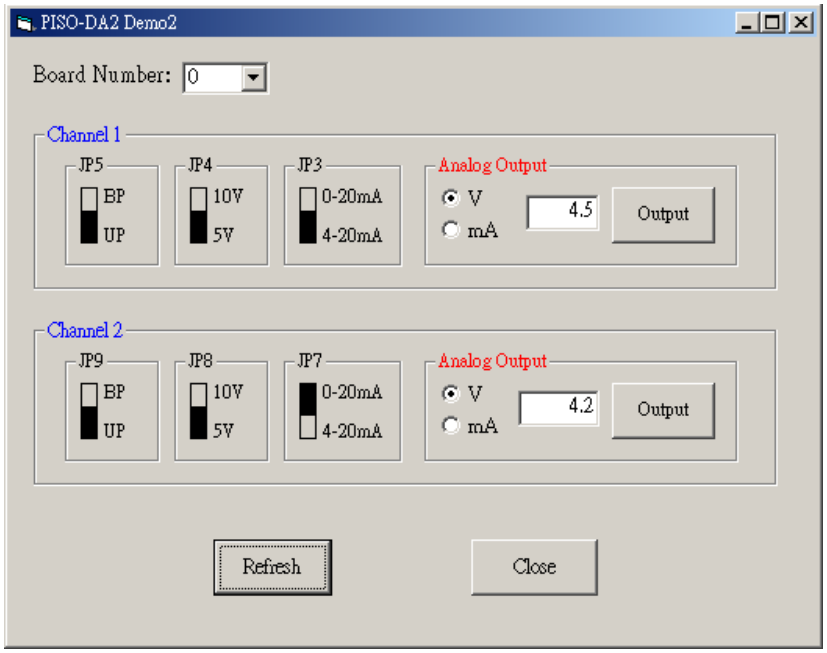


Figure 2-2: The demo_2 program

DEMO3: D/A output

This program demonstrates the method for how to write the data to EEPROM and then read them out. Note that this method is based on the PSIODA_ReadEEP and PISODA_WriteEEP, which access the hardware by a word at one time.

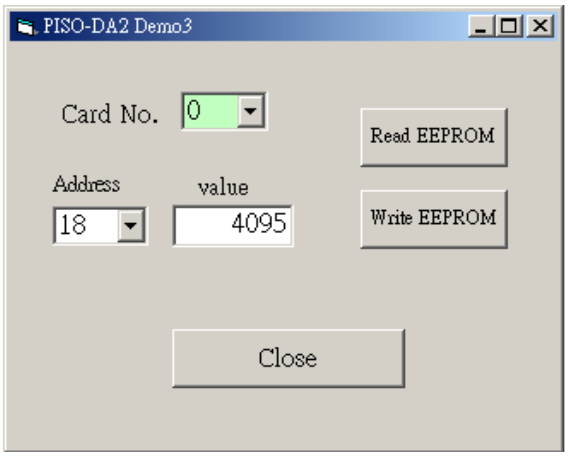


Figure 2-3: The demo_3 program

DEMO4: Multiboard D/A output

This demo program presents the same function as DEMO2 to output analog signal, but using 2 boards instead of one board.

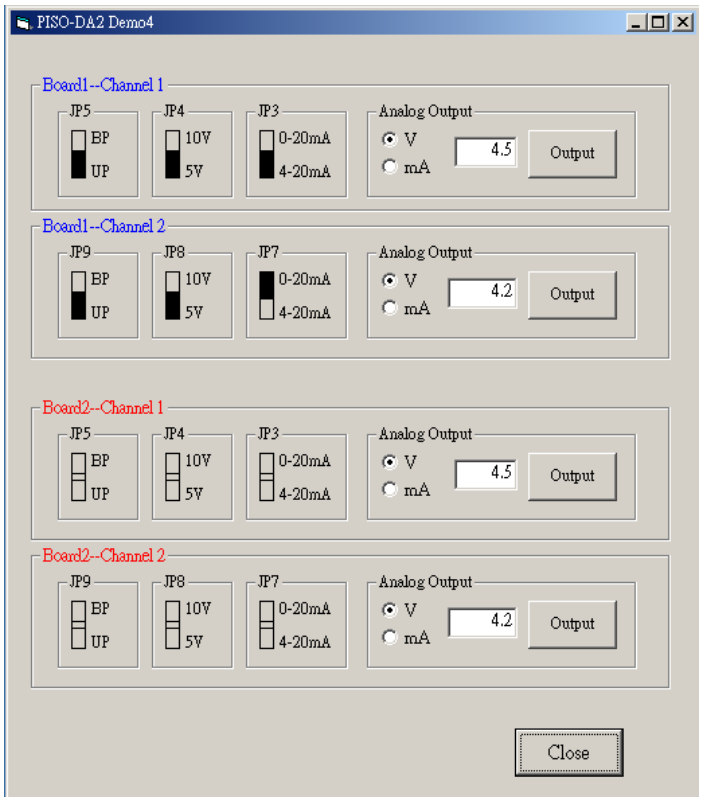


Figure 2-4: The demo_4 program

2.1 PISODA.H

```
#ifdef __cplusplus
    #define EXPORTS extern "C" __declspec (dllimport)
#else
    #define EXPORTS
#endif

#define PISODA_NoError                  0x00
#define PISODA_ActiveBoardError         0x01
#define PISODA_ExceedFindBoards         0x02
#define PISODA_DriverNoOpen             0x03
#define PISODA_BoardNoActive            0x04
#define PISODA_WriteEEPROMError         0x05
#define PISODA_ParameterError           0x06


EXPORTS WORD CALLBACK PISODA_GetDllVersion();

EXPORTS WORD CALLBACK PISODA_ActiveBoard(BYTE BoardNo);
EXPORTS WORD CALLBACK PISODA_CloseBoard(BYTE BoardNo);

EXPORTS WORD CALLBACK PISODA_TotalBoard();
EXPORTS WORD CALLBACK PISODA_GetCardInf(BYTE BoardNo, DWORD *dwVID,
DWORD *dwDID, DWORD *dwSVID,
                                        DWORD *dwSDID, DWORD *dwSAuxID,
DWORD *dwIrq);
EXPORTS BYTE CALLBACK PISODA_IsBoardActive(BYTE BoardNo);

EXPORTS WORD CALLBACK PISODA_DA_Hex(BYTE BoardNo,BYTE
bChannel,WORD wValue);
EXPORTS WORD CALLBACK PISODA_DA(BYTE BoardNo,BYTE bChannel,BYTE
bOpt,float fValue);

EXPORTS WORD CALLBACK PISODA_ReadJumper(BYTE BoardNo, BYTE *Jumper);

EXPORTS WORD CALLBACK PISODA_ReadEEP(BYTE BoardNo,WORD *wValue);
EXPORTS WORD CALLBACK PISODA_WriteEEP(BYTE BoardNo,WORD *wValue);

EXPORTS void CALLBACK PISODA_OutputByte(BYTE BoardNo,DWORD
dwOffset,BYTE bValue);
EXPORTS BYTE CALLBACK PISODA_InputByte(BYTE BoardNo,DWORD dwOffset);
EXPORTS void CALLBACK PISODA_OutputWord(BYTE BoardNo,DWORD
dwOffset,WORD wValue);
EXPORTS WORD CALLBACK PISODA_InputWord(BYTE BoardNo,DWORD dwOffset);
```

2.2 PISODA.PAS

```
unit  PISODA;         { PISODA.dll interface unit }

interface
type PSingle=^Single;
type PWord=^Word;
type DWORD=Cardinal;

const

 PISODA_NoError                  = 0;
 PISODA_ActiveBoardError         = 1;
PISODA_ExceedFindBoards          = 2;
 PISODA_DriverNoOpen             = 3;
 PISODA_BoardNoActive            = 4;
 PISODA_BoardNumIsZero           = 5;
 PISODA_WriteEEPROMError         = 6;
 PISODA_ParameterError           = 7;


function PISODA_GetDllVersion:WORD; StdCall;
function PISODA_ActiveBoard( BoardN:BYTE):WORD; StdCall;
function PISODA_CloseBoard( BoardNo:BYTE):WORD; StdCall;
function PISODA_TotalBoard:WORD; StdCall;

function PISODA_GetCardInf( BoardNo:BYTE;Var dwVID:LongInt; Var
dwDID:LongInt;Var dwSVID:LongInt;Var dwSDID:LongInt;Var
dwSAuxID:LongInt;Var dwIrq:LongInt):WORD; StdCall;

function PISODA_IsBoardActive( BoardNo:BYTE):BYTE; StdCall;


procedure PISODA_OutputByte( BoardNo:BYTE;dwOffset:LongInt; bValue:BYTE);
StdCall;

function PISODA_InputByte( BoardNo:BYTE;dwOffset:LongInt):WORD; StdCall;

procedure
PISODA_OutputWord( BoardNo:BYTE;dwOffset:LongInt;wValue:WORD); StdCall;

function PISODA_InputWord( BoardNo:BYTE;dwOffset:LongInt):LongInt; StdCall;


function PISODA_DA_Hex( BoardNo:BYTE; bChannel:BYTE;
wValue:WORD):WORD; StdCall;

function PISODA_DA( BoardNo:BYTE; bChannel:BYTE; bOpt:BYTE;
fValue:Single):WORD; StdCall;


function PISODA_ReadJumper( BoardNo:BYTE;var Jumper:BYTE):WORD;StdCall;
function PISODA_ReadEEP( BoardNo:BYTE;var wValue:Word):WORD; StdCall;
function PISODA_WriteEEP( BoardNo:BYTE;wValue:PWord):WORD; StdCall;

implementation

function PISODA_GetDllVersion;

        external 'PISODA.DLL' name 'PISODA_GetDllVersion';

function PISODA_ActiveBoard;
```

```
        external 'PISODA.DLL' name 'PISODA_ActiveBoard';
function PISODA_CloseBoard;
        external 'PISODA.DLL' name 'PISODA_CloseBoard';
function PISODA_TotalBoard;
        external 'PISODA.DLL' name 'PISODA_TotalBoard';
function PISODA_GetCardInf;
        external 'PISODA.DLL' name 'PISODA_GetCardInf';
function PISODA_IsBoardActive;
        external 'PISODA.DLL' name 'PISODA_IsBoardActive';
procedure PISODA_OutputByte;
         external 'PISODA.DLL' name 'PISODA_OutputByte';
function PISODA_InputByte;
        external 'PISODA.DLL' name 'PISODA_InputByte';
procedure PISODA_OutputWord;
         external 'PISODA.DLL' name 'PISODA_OutputWord';
function PISODA_InputWord;
        external 'PISODA.DLL' name 'PISODA_InputWord';
function PISODA_DA_Hex;
        external 'PISODA.DLL' name 'PISODA_DA_Hex';
function PISODA_DA;
        external 'PISODA.DLL' name 'PISODA_DA';
function PISODA_ReadJumper;
        external 'PISODA.DLL' name 'PISODA_ReadJumper';
function PISODA_ReadEEP;
        external 'PISODA.DLL' name 'PISODA_ReadEEP';
function PISODA_WriteEEP;
        external 'PISODA.DLL' name 'PISODA_WriteEEP';
end.
```

2.3 PISODA.BAS

```
Attribute VB_Name = "Module1"
Option Explicit

' ============== Error code ==============
' PISODA_NoError                00
' PISODA_ActiveBoardError       01
' PISODA_ExceedFindBoards       02
' PISODA_DriverNoOpen           03
' PISODA_BoardNoActive          04
' PISODA_WriteEEPROMError       05
' PISODA_ParameterError         06

Declare Function PISODA_GetDllVersion Lib "PISODA.dll" () As Integer

Declare Function PISODA_TotalBoard Lib "PISODA.dll" () As Byte

Declare Function PISODA_GetCardInf Lib "PISODA.dll" (ByVal BoardNo As Byte,
dwVID As Long, _
dwDID As Long, dwSVID As Long, dwSDID As Long, dwSAuxID As Long, Irq As
Long) As Integer


Declare Function PISODA_ActiveBoard Lib "PISODA.dll" (ByVal BoardNo As Byte)
As Integer
Declare Function PISODA_CloseBoard Lib "PISODA.dll" (ByVal BoardNo As Byte)
As Integer

Declare Sub PISODA_OutputByte Lib "PISODA.dll" _
  (ByVal BoardNo As Byte, ByVal Offset As Long, ByVal bData As Byte)
Declare Sub PISODA_OutputWord Lib "PISODA.dll" _
  (ByVal BoardNo As Byte, ByVal Offset As Long, ByVal wData As Integer)

Declare Function PISODA_InputByte Lib "PISODA.dll" _
  (ByVal BoardNo As Byte, ByVal dwOffset As Long) As Byte
Declare Function PISODA_InputWord Lib "PISODA.dll" _
  (ByVal BoardNo As Byte, ByVal dwOffset As Long) As Integer

Declare Function PISODA_DA_Hex Lib "PISODA.dll" _
  (ByVal BoardNo As Byte, ByVal bChannel As Byte, ByVal wValue As Integer) As
Integer
Declare Function PISODA_DA Lib "PISODA.dll" _
  (ByVal BoardNo As Byte, ByVal bChannel As Byte, ByVal bOpt As Byte, ByVal
fValue As Single) As Integer

Declare Function PISODA_ReadJumper Lib "PISODA.dll" _
  (ByVal BoardNo As Byte, Jumper As Byte) As Integer

Declare Function PISODA_ReadEEP Lib "PISODA.dll" _
  (ByVal BoardNo As Byte, wData As Integer) As Integer
Declare Function PISODA_WriteEEP Lib "PISODA.dll" _
  (ByVal BoardNo As Byte, wData As Integer) As Integer
```

```
' Hex(String)->Dec(Long)
Public Function H2D(Cnum As String) As Long
 Dim s As String, v As Long, L As Long, i As Long
 Dim Leng As Integer
 Leng = Len(Cnum)
   For i = 1 To Leng
      s = UCase(Mid(Cnum, i, 1))
      If s >= "A" Then
         v = Asc(s) - 55
      Else
         v = CLng(s)
      End If
      L = L + v * 16 ^ (Len(Cnum) - i)
   Next
 H2D = L
End Function
```

2.4 PISODA.vb

```
Imports System.Runtime.InteropServices

Module PISODA2
    Public Const PISODA_NoError = &H0
    Public Const PISODA_ActiveBoardError = &H1
    Public Const PISODA_ExceedFindBoards = &H2
    Public Const PISODA_DriverNoOpen = &H3
    Public Const PISODA_BoardNoActive = &H4
    Public Const PISODA_WriteEEPROMError = &H5
    Public Const PISODA_ParameterError = &H6

    <DllImport("PISODA.dll")> _
    Public Function PISODA_GetDllVersion() As Integer

    End Function
    <DllImport("PISODA.dll")> _
    Public Function PISODA_ActiveBoard(ByVal BoardNo As Byte) As Short

    End Function
    <DllImport("PISODA.dll")> _
    Public Function PISODA_CloseBoard(ByVal BoardNo As Byte) As Integer

    End Function
    <DllImport("PISODA.dll")> _
    Public Function PISODA_TotalBoard() As Integer

    End Function
    <DllImport("PISODA.dll")> _
    Public Function PISODA_GetCardInf(ByVal BoardNo As Byte, ByRef dwVID As
Long, ByRef dwDID As Long, ByRef dwSVID As Long, ByRef dwSDID As Long,
ByRef dwSAuxID As Long, ByRef dwIrq As Long) As Integer

    End Function

    <DllImport("PISODA.dll")> _
    Public Function PISODA_IsBoardActive(ByVal BoardNo As Byte) As Byte

    End Function
    <DllImport("PISODA.dll")> _
    Public Function PISODA_2DA_Hex(ByVal BoardNo As Byte, ByVal bChannel As
Byte, ByVal wValue As Integer) As Integer

    End Function
    <DllImport("PISODA.dll")> _
    Public Function PISODA_DA(ByVal BoardNo As Byte, ByVal bChannel As Byte,
ByVal bOpt As Byte, ByVal fValue As Single) As Short

    End Function
    <DllImport("PISODA.dll")> _
    Public Function PISODA_ReadJumper(ByVal BoardNo As Byte, ByRef Jumper As
Byte) As Short
```

```
    End Function
    <DllImport("PISODA.dll")> _
    Public Function PISODA_ReadEEP(ByVal BoardNo As Byte, ByRef wValue As
Short) As Short

    End Function
    <DllImport("PISODA.dll")> _
    Public Function PISODA_WriteEEP(ByVal BoardNo As Byte, ByRef wValue As
Short) As Short

    End Function
    <DllImport("PISODA.dll")> _
    Public Sub PISODA_OutputByte(ByVal BoardNo As Byte, ByVal dwOffset As
Long, ByVal bValue As Byte)

    End Sub
    <DllImport("PISODA.dll")> _
    Public Function PISODA_InputByte(ByVal BoardNo As Byte, ByVal dwOffset As
Long) As Byte

    End Function
    <DllImport("PISODA.dll")> _
    Public Sub PISODA_OutputWord(ByVal BoardNo As Byte, ByVal dwOffset As
Long, ByVal wValue As Integer)

    End Sub
    <DllImport("PISODA.dll")> _
    Public Function PISODA_InputWord(ByVal BoardNo As Byte, ByVal dwOffset As
Long) As Integer

    End Function
End Module
```

2.5 PISODA.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;

namespace PISO_DA2_Ns
{
   public class PISODA2
   {
      public const int NoError = 0;
      public const int ActiveBoardError = 1;
      public const int ExceedFindBoards = 2;
      public const int DriverNoOpen = 3;
      public const int BoardNoActive = 4;
      public const int WriteEEPROMError = 5;
      public const int ParameterError = 6;


      [DllImport("PISODA.dll", EntryPoint = "PISODA_GetDllVersion")]
      public static extern  int GetDllVersion();


      [DllImport("PISODA.dll", EntryPoint = "PISODA_ActiveBoard")]
      public static extern short ActiveBoard(byte BoardNo);


      [DllImport("PISODA.dll", EntryPoint = "PISODA_CloseBoard")]
      public static extern int CloseBoard(byte BoardNo);


      [DllImport("PISODA.dll", EntryPoint = "PISODA_TotalBoard")]
      public static extern int TotalBoard();


      [DllImport("PISODA.dll", EntryPoint = "PISODA_GetCardInf")]
      public static extern int GetCardInf(byte BoardNo , out long dwVID, out long
dwDID , out long dwSVID , out long dwSDID , out long dwSAuxID , out long
dwIrq );


      [DllImport("PISODA.dll", EntryPoint = "PISODA_IsBoardActive")]
      public static extern  byte IsBoardActive( byte BoardNo );


      [DllImport("PISODA.dll", EntryPoint = "PISODA_2DA_Hex")]
      public static extern int PISODA_2DA_Hex(byte BoardNo, byte bChannel, int
wValue);
```

```
        [DllImport("PISODA.dll", EntryPoint = "PISODA_DA")]
        public static extern  short DA( byte BoardNo , byte  bChannel , byte
bOpt ,float  fValue ) ;


        [DllImport("PISODA.dll", EntryPoint = "PISODA_ReadJumper")]
        public static extern  short ReadJumper( byte BoardNo , out byte Jumper );


        [DllImport("PISODA.dll", EntryPoint = "PISODA_ReadEEP")]
        public static extern short ReadEEP( byte  BoardNo, out short wValue ) ;


        [DllImport("PISODA.dll", EntryPoint = "PISODA_WriteEEP")]
        public static extern short WriteEEP( byte BoardNo , out short wValue ) ;


        [DllImport("PISODA.dll", EntryPoint = "PISODA_OutputByte")]
        public static extern void  OutputByte( byte BoardNo , long  dwOffset , byte
bValue );


        [DllImport("PISODA.dll", EntryPoint = "PISODA_InputByte")]
        public static extern byte InputByte( byte BoardNo , long  dwOffset ) ;


        [DllImport("PISODA.dll", EntryPoint = "PISODA_OutputWord")]
        public static extern void  OutputWord( byte BoardNo ,  long dwOffset , int
wValue );


        [DllImport("PISODA.dll", EntryPoint = "PISODA_InputWord")]
        public static extern int InputWord(byte BoardNo, long dwOffset);

    }

}
```

3. Function Descriptions

All of the functions provided for PISO-DA2/DA2U are listed as below and the detail information for every function will be presented in the following section. However, in order to make the description simplify and clearly, the attribute of the input and output parameter of the function is indicated as [input] and [output] respectively, as shown in following Table.

Keyword	Set parameter by user before calling this function?	Get the data from this parameter after calling this function?
[Input]	**Yes**	No
[Output]	No	**Yes**

Reference	Function Definition
Sec. 3.1	WORD PISODA_GetDllVersion();
Sec. 3.2	WORD PISODA_ActiveBoard(BYTE BoardNo);
Sec. 3.3	WORD PISODA_CloseBoard(BYTE BoardNo);
Sec. 3.4	WORD PISODA_TotalBoard();
Sec. 3.5	WORD PISODA_GetCardInf(BYTE BoardNo, DWORD *dwVID, DWORD *dwDID, DWORD *dwSVID, DWORD *dwSDID, DWORD *dwSAuxID, DWORD *dwIrq);
Sec. 3.6	BYTE PISODA_IsBoardActive(BYTE BoardNo);
Sec. 3.7	WORD PISODA_DA_Hex(BYTE BoardNo,BYTE bChannel,WORD wValue);
Sec. 3.8	WORD PISODA_DA(BYTE BoardNo,BYTE bChannel,BYTE bOpt,float fValue);
Sec. 3.9	WORD PISODA_ReadJumper(BYTE BoardNo, BYTE *Jumper);
Sec. 3.10	WORD PISODA_ReadEEP(BYTE BoardNo,WORD *wValue);
Sec. 3.11	WORD PISODA_WriteEEP(BYTE BoardNo,WORD *wValue);
Sec. 3.12	BYTE PISODA_InputByte(BYTE BoardNo,DWORD dwOffset);
Sec. 3.13	void PISODA_OutputByte(BYTE BoardNo, DWORD dwOffset, BYTE bValue);
Sec. 3.14	WORD PISODA_InputWord(BYTE BoardNo,DWORD dwOffset);
Sec. 3.15	void PISODA_OutputWord(BYTE BoardNo,DWORD dwOffset,WORD wValue);

3.1 PISODA_GetDllVersion

- **Description:**
  Obtain the version information of the PISODA.DLL driver.

- **Syntax:**
  WORD **PISODA_GetDllVersion**(Void);

- **Parameters:**
  None

- **Returns:**
  DLL version information.
  For example: if 101 (hex) is return, it means driver version is 1.01.

3.2 PISODA_ActiveBoard

- **Description:**
  Activate the device. It must be called once before using the other functions of PISO-DA2/DA2U board.

- **Syntax:**
  WORD **PISODA_ActiveBoard**(BYTE **BoardNo**);

- **Parameters:**

BoardNo	[Input]	PISO-DA2/DA2U board number (1~16).

- **Returns:**

PISODA_NoError	OK.
PISODA_DriverNoOpen	Kernel driver can not be found.
PISODA_ExceedFindBoards	BoardNo exceeds the current total board number (N).
PISODA_ActiveBoardError	This board can not be activated..

3.3 PISODA_CloseBoard

- **Description:**
  Stop and close the PISO-DA2/DA2U kernel driver and release the resources the device resource from computer device resource. This method must be called once before exiting the user's application program.

- **Syntax:**
  WORD **PISODA_CloseBoard**(BYTE **BoardNo**);

- **Parameters:**

BoardNo	[Input]	PISO-DA2/DA2U board number (0~15).

- **Returns:**

PISODA_NoError	OK.
PISODA_DriverNoActive	The board is not activated.
PISODA_ExceedFindBoards	BoardNo exceeds the current total board number (N).

3.4 PISODA_TotalBoard

- **Description:**
  Obtain the total board number of PISO-DA2 boards installed in the PCI bus.

- **Syntax:**
  WORD CALLBACK **PISODA_TotalBoard**(void);

- **Parameters:**
  None

- **Returns:**
  Return the total board number.

3.5 PISODA_GetCardInf

- **Description:**
  Obtain the information of PISO-DA2/DA2U boards, which include vender ID, device ID and interrupt number.

- **Syntax:**
  WORD CALLBACK **PISODA_GetCardInf**(BYTE **BoardNo**, DWORD ***dwVID**, DWORD ***dwDID**, DWORD ***dwSVID**, DWORD ***dwSDID**, DWORD ***dwSAuxID**, DWORD ***dwIrq**);

- **Parameters:**

BoardNo	[Input]	PISO-DA2/DA2U board number
***dwVID**	[Output]	vendor ID of this board
***dwDID**	[Output]	device ID of this board
***dwSVID**	[Output]	sub-vendor ID of this board
***dwSDID**	[Output]	sub-device ID of this board
***dwSAuxID**	[Output]	sub-auxiliary ID of this board
***dwIrq**	[Output]	logical interrupt number of this board

- **Returns:**

PISODA_NoError	OK.
PISODA_DriverNoOpen	Kernel driver can not be found.
PISODA_ExceedFindBoards	BoardNo exceeds the current total board number (N).

3.6 PISODA_IsBoardActive

- **Description:**
  Obtain the information about the specific board is active or not.

- **Syntax:**
  BYTE **PISODA_IsBoardActive**(BYTE **BoardNo**);

- **Parameters :**

BoardNo	[Input]	PISO-DA2/DA2U Board number

- **Returns:**
  “0” means the board is inactive.
  “1” means the board is active.

3.7 PISODA_DA_Hex

■ **Description:**

Output the analog output data in Hex format through channel 1 or 2 of the PSIO-DA2/DA2U. Note that the analog output can be configured voltage or current sink type, which is dependent on the hardware jumper setting.

■ **Syntax:**

WORD **PISODA_DA_Hex**(BYTE **BoardNo**, BYTE **bChannel**, WORD **wValue**);

■ **Parameters:**

BoardNo	[Input]	PISO-DA2/DA2U board number.
bChannel	[Input]	D/A Channel number 1 or 2.
wValue	[Input]	Analog output value (0x~0xfff)

■ **Returns:**

PISODA_NoError	OK.
PISODA_DriverNoOpen	Kernel driver can not be found.
PISODA_ExceedFindBoards	BoardNo exceeds the current total board number (N).
PISODA_BoardNoActive	The board is not activated.
PISODA_ParameterError	wValue is out of range.

3.8 PISODA_DA

■ **Description:**

Output the analog output data in float format through channel 1 or 2 of the PSIO-DA2/DA2U. Note that the analog output can be configured voltage or current sink type, which is dependent on the hardware jumper setting.

■ **Syntax:**

WORD **PISODA_DA**(BYTE **BoardNo**, BYTE **bChannel**, BYTE **bOpt**, float **fValue**);

■ **Parameters:**

BoardNo	[Input]	PISO-DA2/DA2U board number.
bChannel	[Input]	D/A Channel number 1 or 2.
bOpt	[Input]	0 for voltage output. 1 for current sink.
wValue	[Input]	Analog output value.

■ **Returns:**

PISODA_NoError	OK.
PISODA_DriverNoOpen	Kernel driver can not be found.
PISODA_ExceedFindBoards	BoardNo exceeds the current total board number (N).
PISODA_BoardNoActive	The board is not activated.
PISODA_ParameterError	wValue is out of range.

Note: Refer to DEMO2 for more information.

3.9 PISODA_ReadJumper

■ **Description:**

Obtain the configuration status of the jumper JP3, JP4, JP5, JP7, JP8, and JP9. Please call PISODA_ActiveBoard first before using this function.

■ **Syntax:**

WORD **PISODA_ReadJumper**(BYTE **BoardNo**, BYTE ***Jumper**);

■ **Parameters:**

BoardNo	[Input]	PISO-DA2/DA2U board number.
***Jumper**	[Output]	A value of jumper status. Please refer to following table.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	JP9	JP8	JP7	JP5	JP4	JP3

Table 3.1: Jumper Status

Jumper	Jumper value=0	Jumper value =1
JP3 (ch1)	Current output is 0-20 mA	Current output is 4-20 mA
JP4 (ch1)	Reference voltage is –10 V	Reference voltage is –5 V
JP5 (ch1)	Bipolar setting	Unipolar setting
JP7 (ch2)	Current output is 0-20 mA	Current output is 4-20 mA
JP8 (ch2)	Reference voltage is –10 V	Reference voltage is –5 V
JP9 (ch2)	Bipolar setting	Unipolar setting

Example:

If *Jumper is 0x27 (0010 0111), then it means that JP3=1, JP4=1, JP5=1, JP7=0, JP8=0,J P9=1.

■ **Returns:**

PISODA_NoError	OK.
PISODA_DriverNoOpen	Kernel driver can not be found.
PISODA_ExceedFindBoards	BoardNo exceeds the current total board number (N).
PISODA_BoardNoActive	The board is not activated.

3.10 PISODA_ReadEEP

■ **Description:**

Obtain the 64 words(128 bytes) data from the EEPROM of the PISO-DA2/DA2U board. Please call PISODA_ActiveBoard first before using this function.

■ **Syntax:**

WORD **PISODA_ReadEEP**(BYTE **BoardNo**,WORD ***wValue**);

■ **Parameters:**

BoardNo	[Input]	PISO-DA2/DA2U board number.
***wValue**	[Output]	The first word (16-bit) of data from EEPROM

■ **Returns:**

PISODA_NoError	OK.
PISODA_DriverNoOpen	Kernel driver can not be found.
PISODA_ExceedFindBoards	BoardNo exceeds the current total board number (N).
PISODA_BoardNoActive	The board is not activated.

Note: Refer to DEMO3 for more information.

3.11 PISODA_WriteEEP

■ **Description:**

Write 64 words (128 bytes) data into the EEPROM of the PISO-DA2/DA2U board. Please call PISODA_ActiveBoard first before using this function.

■ **Syntax:**

WORD **PISODA_WriteEEP**(BYTE **BoardNo**, WORD ***wValue**);

■ **Parameters:**

wBoards	[Input]	PISO-DA2/DA2U board number.
***wValue**	[Input]	The first word (16-bit) of data.

■ **Returns:**

PISODA_NoError	OK.
PISODA_DriverNoOpen	Kernel driver can not be found.
PISODA_ExceedFindBoards	BoardNo exceeds the current total board number (N).
PISODA_BoardNoActive	The board is not activated.
PISODA_WriteEEPROMError	Fail to write data to EEPROM.

Note: Refer to DEMO3 for more information.

3.12 PISODA_InputByte

- **Description:**
  Obtain a byte data from the specific address mapping of the PISO-DA2/DA2U board. Please call PISODA_ActiveBoard first before using this function. This function is designed for advance user to access the hardware data based on the register of PISO-DA2/DA2U.

- **Syntax:**
  BYTE **PISODA_InputByte**(BYTE **BoardNo**, DWORD **dwOffset**);

- **Parameters:**

BoardNo	[Input]	PISO-DA2/DA2U board number.
dwOffset	[Input]	The offset value of the base address of the PISO-DA2/DA2U board for the mapping address, from 0 to 0xff.

- **Returns:**
  One byte value or data.

3.13 PISODA_OutputByte

- **Description:**
  Write a byte data to the defined address of the PISO-DA2/DA2U board. This function is designed for advance user to write into the hardware based on the register of PISO-DA2/DA2U.

- **Syntax:**
  void **PISODA_OutputByte**(BYTE **BoardNo**, DWORD **dwOffset**, BYTE **bValue**);

- **Parameters:**

BoardNo	[Input]	PISO-DA2/DA2U board number.
dwOffset	[Input]	The offset value of the base address of the PISO-DA2/DA2U board for the mapping address, from 0 to 0xff.
bValue	[Input]	A byte value for output.

- **Returns:**
  None

3.14 PISODA_InputWord

- **Description:**
  Obtain a word (two bytes) data from the specific mapping address of the PISO-DA2/DA2U board. Please call PISODA_ActiveBoard first before using this function. This function is designed for advance user to access the hardware data based on the register of PISO-DA2/DA2U.

- **Syntax:**
  BYTE **PISODA_InputWord**(BYTE **BoardNo**, DWORD **dwOffset**);

- **Parameters:**

BoardNo	[Input]	PISO-DA2 board number
dwOffset	[Input]	The offset of base address of the PISO-DA2/DA2u board for the mapping address, from 0 to 0xff.

- **Returns:**
  One word value or data.

3.15 PISODA_OutputWord

- **Description:**
  Write a word(two bytes) data to the defined address of the PISO-DA2/DA2U board. This function is designed for advance user to write into the hardware based on the register of PISO-DA2/DA2U.

- **Syntax:**
  void **PISODA_OutputWord**(BYTE **BoardNo**, DWORD **dwOffset**, WORD **wValue**);

- **Parameters:**

BoardNo	[Input]	PISO-DA2 board number
dwOffset	[Input]	The offset of base address of the PISO-DA2/DA2u board for the mapping address, from 0 to 0xff.
wValue	[Input]	A word value.

- **Returns:**
  None