ActiveX Control (OCX) for PCI-Bus Series boards

User's Manual

[Version 1.1]

PIODAX OCX is for PIO-DA

PIODIOX OCX is for PIO-DIO

PISODIOX OCX is for PISO-DIO

PISO725X OCX is for PISO-725

TMC12X OCX is for PCI-TMC12A

PISO813X OCX is for PISO-813

PCIPRX OCX is for PCI-R16/P8R8 /P16C16/P16OR16

Table of Contents

1. Introduction

The PCIPRX (PIODAX, PIODIOX, PISO725X, PISODIOX, TMC12X, PISO813X) is an ActiveX Control component (OCX) for PCI-Bus series boards. It enables you to develop programs in a quick and easy way. Before using this driver, users need to install the PCIPRX (PIODAX, PIODIOX, PISO725X, PISODIOX, TMC12X, PISO813X) OCX drivers into the system firstly and then insert the OCX component into the selected software development tools. Finally, you can make use of this OCX just like the general ActiveX Control components (OCX) included in your development tools.

Please also refer to "ActiveX Control (OCX) Installation Manual". It contains the following topics:

1. Installing the software into your system.
2. Installing/Uninstalling the ActiveX Control into/from Visual Basic 5.0
3. Installing/Uninstalling the ActiveX Control into/from Delphi 5.0
4. Installing/Uninstalling the ActiveX Control into/from Borland C++ Builder 3.0

Following table shows the current supported products of PCI DAQ card for OCX.

OCX	Card model
TMC12X	PCI-TMC12A
PIODIOX	PIO-D168, PIO-D144, PIO-D96, PIO-D64, PIO-D56, PIO-D48, PIO-D24
PISO725X	PISO-725
PISO813X	PISO-813
PIODAX	PIO-DA16, PIO-DA8, PIO-DA4
PCIPRX	PCI-P8R8, PCI-P16R16, PCI-P16C16, PCI-P16OR16
PISODIOX	PISO-P8R8, PISO-P8SSR8AC, PISO-P8SSR8DC PISO-P32A32,PISO-P32C32,PISO-A64,PISO-P64, PISO-C64,PISO-730, PISO-730A

The following tables are the function lists for the specific OCX.

	PCIPRX
property	short ErrorCode;
property	BSTR ErrorString;
property	short ActiveBoard;
method	short DriverInit();
method	long GetConfigAddressSpace(short nAddrNum);
method	short GetDllVersion();
method	short GetDriverVersion();
method	void DigitalOut(short nOutputValue);
method	short DigitalIn();
method	void DriverClose();

	PIODAX
property	long ErrorCode;
property	BSTR ErrorString;
property	short ActiveBoard;
method	short DriverInit();
method	long GetConfigAddressSpace(short nAddrNum);
method	short GetDriverVersion();
method	short GetDllVersion();
method	long DigitalIn();
method	void DigitalOut(long lOutputValue);
method	void OutVoltage(short nChannel, float OutputValue);
method	void OutCurrent(short nChannel, float fOutputValue);
method	short InputByte(long lBaseAddr);
method	void OutputByte(long lBaseAddr, short nOutputData);
method	void OutputWord(long lBaseAddr, long lOutputData);
method	long InputWord(long lBaseAddr);
method	void SetCounter(short nCounterNo, short nCounterMode, long nCounterVal);
method	void ResetIrqCount();
method	void InstallIrq(long* hEvent, short nIrqSource, short nActiveMode);
method	void RemoveIrq();
method	long GetIrqCount();
method	void DriverClose();

	PIODIOX
property	BSTR ErrorString;
property	short ErrorCode;
property	short ActiveBoard;
method	short DriverInit();
method	long GetConfigAddressSpace(short nAddrNum);
method	short GetDllVersion();
method	short GetDriverVersion();
method	short InputByte(long lBaseAddr);
method	long InputWord(long lBaseAddr);
method	void OutputByte(long lBaseAddr, short nOutputData);
method	void OutputWord(long lBaseAddr, long lOutputData);
method	short DigitalIn(short nPort);
method	void DigitalOut(short nPort, short nOutputValue);
method	void SetCounter(short nCounterNo, short nCounterMode, long nCounterVal);
method	long ReadCounter(short nCounterNo, short nCounterMode);
method	long ResetIrqCount();
method	void InstallIrq(long* hEvent, short nIrqSource, short nActiveMode);
method	void RemoveIrq();
method	void GetIrqCount();
method	long D48Freq();
method	void D48InstallIrq(long* lHandle, short nIrqMask, short nActiveMode);
method	void D48RemoveIrq();
method	long D48GetIrqCount();
method	void SaveIrqActiveFlag();
method	short GetIrqActiveFlag(short FlagNum);
method	void DriverClose();

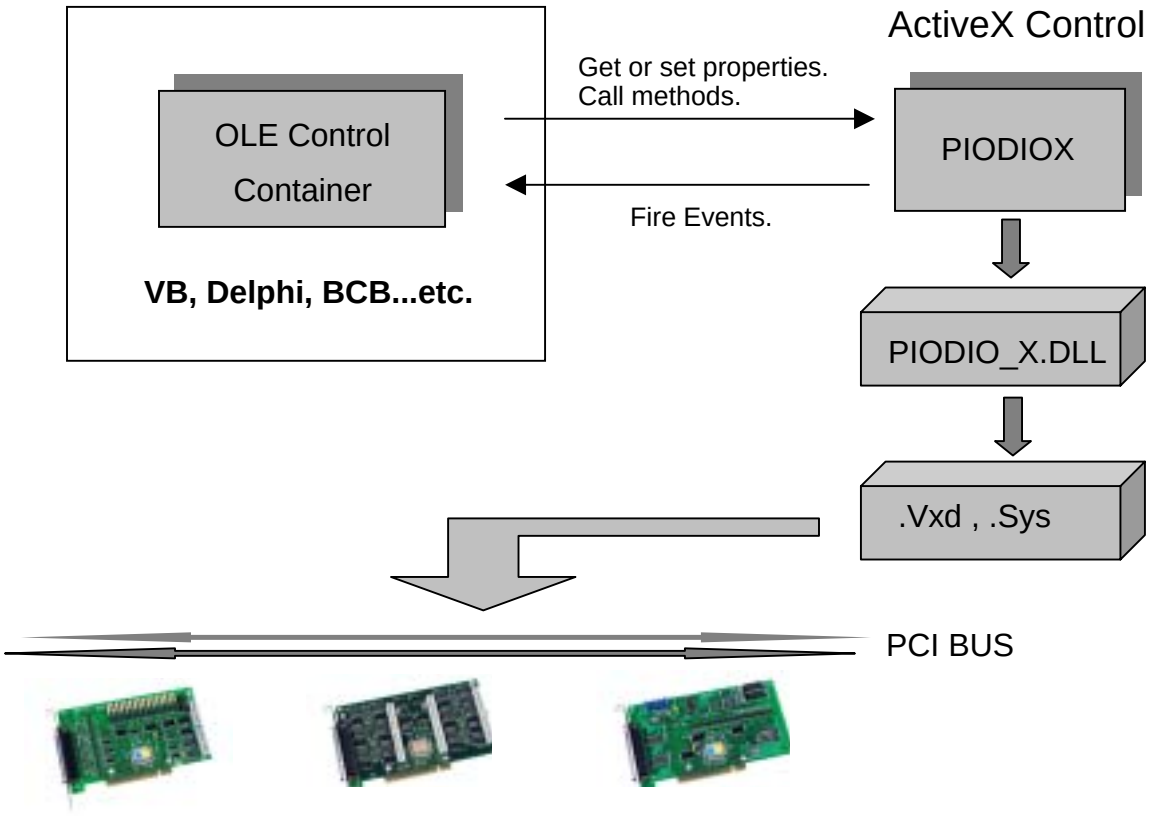
	PISO725X
property	short ErrorCode;
property	BSTR ErrorString;
property	short ActiveBoard;
method	short DriverInit();
method	long GetConfigAddressSpace(short nAddrNum);
method	short GetDriverVersion();
method	short GetDllVersion();
method	void OutputByte(long lBaseAddr, short nOutputData);
method	short InputByte(long lBaseAddr);
method	void OutputWord(long lBaseAddr, long lOutputData);
method	long InputWord(long lBaseAddr);
method	void DigitalOut(short nOutputValue);
method	short DigitalIn(short nPort);
method	void InstallIrq(long* hEvent);
method	void RemoveIrq();
method	short GetIrqActiveFlag(short FlagNum);
method	void SaveIrqActiveFlag();
method	void DriverClose();

	TMC12X
property	short ErrorCode;
property	BSTR ErrorString;
property	short ActiveBoard;
method	short DriverInit();
method	long GetConfigAddressSpace(short nAddrNum);
method	short GetDriverVersion();
method	short GetDllVersion();
method	void OutputWord(short nPortAddress, short nOutData);
method	void InputWord(short nPortAddress);
method	short DigitalIn();
method	void DigitalOut(short nOutputValue);
method	void Select8254(short nWhich8254);
method	void SetCounter(short nCounterNo, short nCounterMode, long lCounterVal);
method	long ReadCounter(short nCounterNo, short nCounterMode);
method	short GetIrqNo();
method	void DriverClose();

	PISODIOX
property	short ErrorCode;
property	BSTR ErrorString;
property	short ActiveBoard;
method	short DriverInit();
method	long GetConfigAddressSpace(short nAddrNum);
method	short GetDllVersion();
method	short GetDriverVersion();
method	void OutputByte(long lBaseAddr, short nOutputData);
method	void OutputWord(long lBaseAddr, long lOutputData);
method	short InputByte(long lBaseAddr);
method	long InputWord(long lBaseAddr);
method	void DigitalOut(short nPort, short nOutputValue);
method	short DigitalIn(short nPort);
method	void ResetIrqCount();
method	void InstallIrq(long* hEvent, short nIrqSource, short nActiveMode);
method	void RemoveIrq();
method	long GetIrqCount();
method	void DriverClose();

	PISO813X
property	long ErrorCode;
property	BSTR ErrorString;
property	short ActiveBoard;
method	short DriverInit();
method	long GetConfigAddressSpace();
method	short GetDllVersion();
method	short GetDriverVersion();
method	void OutputByte(long lBaseAddr, short nOutputData);
method	void OutputWord(long lBaseAddr, long lOutputData);
method	short InputByte(long lBaseAddr);
method	long InputWord(long lBaseAddr);
method	void SetChannelGain(short nChannel, short nGainCode);
method	short AnalogInHex();
method	float GetHextoFloat(short nHex, short nGainCode, short nJUMP, short nBipolar);
method	float AnalogIn(short nJump, short nBipolar);
method	void AnalogInMultiHex(short* wAdsBuf, long lCount);
method	void AnalogInMulti(short nJump20V, short nBipolar, float* fAdsBuf, long lCount);
method	void DriverClose();

The following figure gives the programming system architecture for the ActiveX Control.



2. Data Type

ActiveX Control Data Type	Data Size	BCB	Delphi	VB
short	2 Bytes	Short	SmallInt	Integer
long	4 Bytes	Long	LongInt	Long
float	4 Bytes	Float	Single	Single
LPCTSTR		Wchar_t *	String	String
BSTR		AnsiString	String	String

3. Interface of PCI-bus board

The interface of PCI series boards is for PCIPRX, PIODAX, PIODIOX, PISO725X, PISODIOX, TMC12X and PISO813X.

3.1. General Properties

Property Name	Data Type	Data Size	Access mode	Run-time Only	Description
ErrorCode	long(short)	4/2 (Bytes)	Read/Write	No	Get/select the Error Code
ErrorString	BSTR		Read only	No	Get the Error Message

Note: In the following examples, please replace the "Control" word by your Control-Object name. For example:

(1) PCIPRX1 or PCIPRX2 is for PCIPRX OCX,

(2) PIODAX 1or PIODAX2 is for PIODAX OCX,

(3) PIODIOX1 or PIODIOX2 is for PIODIOX OCX.

(4) PISO725X1 or PISO725X2 is for PISO725X OCX.

(5) PISODIOX1 or PISODIOX2 is for PISODIOX OCX.

(6) PISO813X1 or PISO813X2 is for PISO813X OCX.

(7) TMC12X1 or TMC12X2 is for TMC12X OCX.

3.1.1. ErrorCode

This property records any error code (includes 0: no-error) produced by software function driver after you use any method or other properties. Users should check on this property to make sure no error occurred after using any function.

Return: (long)(short)

Example:

```
If Control.ErrorCode <>0  then
' Error occurs
' Do something
' Exit sub
End if
```

Comment: Please refer to the ErrorString property to understand the meaning of error message.

3.1.2. ErrorString

This property provides message information according to error code when an error has been occurred. That is, users can get the meaning of error message from the ErrorString property.

Return: (BSTR)

Example:

```
strError = Control.ErrorString    'Get the error message.
```

3.1.3. Table of ErrorCode and ErrorString

Error Code	Error ID	Error String	Comment
0	**NoError**	(OK ! No Error!)	
1	**DrivrOpenError**	Device driver can't be opened	
2	**DriverNoOpen**	Users have to call the DriverInit function firstly	
3	**GetDriverVersionError**	Get driver version error	
4	**InsrallIrqError**	Can't install interrupt service	
5	**ClearIntCountError**	Can't clear interrupt count	
6	**GetIntCountError**	Can't get interrupt count	
7	**RegisterApcError**	Register Apc Error	
8	**RemoveIrqError**	Can't remove interrupt resource	
9	**FindBoardError**	Check your card	
10	**ExceedBoardNumber**	The Max. boards is : 8	
11	**ResetError**	Can't reset interrupt count	
12	**IrqMaskError**	Irq-Mask is 1,2,4,8 or 1 to 0xF	
13	**ActiveModeError**	Active-Mode is 1,2 or 1 to 3	
14	**GetActiveFlagError**	Can't get interrupt active flag	
15	**ActiveFlagEndOfQueue**	The flag queue is empty	
0xffff	**TimeOutError**	Time Out	
-100	**AdError2**	AD value Error	

3.2. General Methods

Method Name	Data type of returned value	Data size	Descriptions
DriverInit	short	2 bytes	Open the driver and allocate the resource for the device,and get the total boards.
GetDLLVersion	short	2 bytes	Get the DLL file version.
GetDriverVersion	short	2 bytes	Get the driver version in the system
DriverClose	void		Close the Driver and release the resource from the device.

3.2.1. DriverInit

This method is used to start the PCI series board's driver and allocate the computer resource for the device. This method must be called once before calling or using control methods or other properties

Prototype: short DriverInit()

Return: Total boards of defined PCI series boards.

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0 'Set the active board to 0
Print Control.GetDllVersion        'Print the DLL version
```

3.2.2. GetDllVersion

This method is used to get the DLL version information of PCI series board's driver.

Prototype: short GetDllVersion()

Return: DLL version of PCI series board.

Example:

```
'For example: If it returns 0x250, then the version is 2.50.
wVer = Control.GetDllVersion
```

3.2.3. GetDriverVersion

This method is used to obtain the driver version information of PCI series board from .Vxd driver of window 9X or .Sys driver of windows NT/2000.

Prototype: short GetDriverVersion()

Return: Driver version from operation system.

Example:

```
'For example: If it returns 0x200, then the version is 2.00.
wVer = Control.GetDriverVersion
```

3.2.4. DriverClose

Stop and close the OCX Driver of PCI series board and release the device resource from computer device resource. This method must be called once before exiting the user's application program.

Prototype: void DriverClose ()

Example:

```
TotalBoards = Control.DriverInit   'initial driver
Control.ActiveBoard = 0   'select action board=0
            …
            …
Control.DriverClose   'release the device resource
```

3.3. General Properties

Properties Name	Data type of returned value	Data size	Descriptions
ActiveBoard	short	2 bytes	Get or select active PCI series boards installed in the system.

3.3.1. ActiveBoard

This property is used to activate one of the PCI series boards installed in the system. This function must be called once before the DigitalIn, DigitalOut and etc. functions are used. It also allows users to get or select the **ActiveBoard**.

Return: (short) Active board number.

Default: 0 (ActiveBoard=0)

Example:

```
wTotalBoard = Control.DriverInit   'Initialize the driver and get the total boards
Control.ActiveBoard=0          ‘Set the active board to 0
Board=Control.ActiveBoard   ‘Get the active board value
```

3.4. Digital Input/Output Methods

Method Name	Data type of returned value	Data size	Descriptions
DigitalIn	short	2 bytes	Get digital input value.
DigitalOut	void		Output digital value.

These methods are for PCIPRX, PIODAX, PIODIOX, PISO725X, PISODIOX, and TMC12X OCX.

In these cards listed in followings table, you can switch the ports of eight digital channels in group to be output and/or input. And every digital component has 8 digital channels.

Card Model	Port Number (DI / DO)	OCX	Card Model	Port Number (DI / DO)	OCX
PIO-D24	3 / 3	PIODIOX	PIO-D168	21/21	PIODIOX
PIO-D48	6 / 6	PIODIOX	PISO-P8R8	1 / 1	PISODIOX
PIO-D56	5 / 5	PIODIOX	PISO-P32A32/P32C32	4 / 4	PISODIOX
PIO-D64	4 / 4	PIODIOX	PISO-A64/C64	0 / 8	PISODIOX
PIO-D96	12/12	PIODIOX	PISO-P64	8 / 0	PISODIOX
PIO-D144	18/18	PIODIOX	PISO-730/730A	4 / 4	PISODIOX

3.4.1. DigitalIn

This method is used to obtain the digital input value from the active PCI series board.

Prototype1: short DigitalIn ()

Return: 16 bits data from Digital input port

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0    ‘Set the active board to 0
Print Control.DigitalIn      ‘Print the digital input value
```

Comment: This function is only for PCIPRX/PISO725X/TMC12X/PIODAX OCX.

Prototype2: short DigitalIn (short nPort)

Parameters: nPort : digital input port number of PCI card.

Return: 8 bits data from specified Digital input port

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0    ‘Set the active board to 0
Print Control.DigitalIn 0      ‘Print the Port-0 digital input value
```

Comment :This function is only for PISODIOX/PIODIOX OCX.

3.4.2. DigitalOut

This method is used to output the digital value through the active PCI series board.

Prototype1: void DigitalOut (long wDo)

Parameters: wDo: Digital output value

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0   ‘Set the active board to 0
Control.DigitalOut  5       ‘Digital output : 5
```

Comment :This function is only for PCIPRX/PISO725X/TMC12X/PIODAX OCX.

Prototype2: void DigitalOut (short nPort, short nOutputValue)

Parameters: nPort : digital output port number of PCI card

.nOutputValue: 8 bit Digital output value.

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0   ‘Set the active board to 0
Control.DigitalOut  1, 5       ‘Digital output port=1   output value=5
```

Comment :This function is only for PISODIOX/PIODIOX OCX.

3.5. Input/Output/Get Address Methods

Method Name	Data type of returned value	Data size	Descriptions
GetConfigAddressSpace	long	4 bytes	Get the card information.
InputByte	short	2 bytes	Input the 8 bit data from the desired I/O port.
InputWord	long	4 bytes	Input the 16 bit data from the desired I/O port.
OutputByte	void		Send the 8 bits data to the desired I/O port.
OutputWord	void		Send the 16 bits data to the desired I/O port.

InputByte and OutputByte methods are not for TMC12X/PCIPRX OCX.

InputWord and OutputWord methods are not for PCIPRX OCX.

3.5.1. GetConfigAddressSpace

This method is used to get the base address/IO address or specified information of PCI serial boards. Please refer to the following table.

Prototype: long GetConfigAddressSpace(short nAddrNum)

Parameters: nAddrNum : specified parameter as following table.

Return: Card information, refer to the following table.

nAddrNum / ocx	0	1	2	3	4	5	6
PIODIOX	AddrBase	SubVendor	SubDevice	SubAux	SlotBus	SlotDevice	
PISODIOX	AddrBase	SubVendor	SubDevice	SubAux	SlotBus	SlotDevice	
PIODAX	AddrBase	IrqNo	SubVendor	SubDevice	SubAux	SlotBus	SlotDevice
PIOS725X	AddrBase	IrqNo	SubVendor	SubDevice	SubAux	SlotBus	SlotDevice
PISO813X		IrqNo	SubVendor	SubDevice	SubAux	SlotBus	SlotDevice
PCIPRX	TypeID			Address2			
TMC12X	AddrBase	IrqNo					

AddrBase : base address of the board control word

SubVendor : subVendor ID of this board.

SubDevice : subDevice ID of this board.

SubAux : subAux ID of this board.

SlotBus : hardware slot ID1 of this board.

SlotDevice : hardware slot ID2 of this board.

IrqNo : allocated IRQ channel number of this board.

TypeID : 0 → this board is PCI-P16R16 , 1 → this board is PCI-P8R8

Address2 : I/O port address

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0   ‘Set the active board to 0
Print Control. GetConfigAddressSpace (0)       ‘print the base address
```

3.5.2. InputByte

This method is used to obtain the 8 bit data from the desired digital input port.

Prototype: short InputByte(long lAddrBase)

Parameters: lAddrBase: I/O port addresses

Return: 16 bits data with the leading 8 bits are all 0

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0              'Set the active board to 0
wBaseAddr =Control. GetConfigAddressSpace (0)      'get  the AddrBase
Control.OutputByte wBaseAddr, 1                    ' enable all DI/DO
InVal2 = Control.InputByte(wBaseAddr + &HC4)       ' input value from Port 1
```

Comment: This function is not for PCIPRX/TMC12X OCX.

3.5.3. InputWord

This method is used to input the 16 bit data from the desired digital input port.

Prototype: long InputWord(long lAddrBase)

Parameters: lAddrBase: I/O port addresses

Return: 32 bit data. Only the low WORD is valid.

Comment: This function is not for PCIPRX OCX.

3.5.4. OutputByte

This method is used to send the 8 bits data to the desired digital output port.

Prototype: void OutputByte(long lAddrBase, short nOutputVal)

Parameters: lAddrBase: I/O port addresses.

nOutputVal: 8 bit data send to I/O port.

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0   ‘Set the active board to 0
wBaseAddr =Control. GetConfigAddressSpace (0)       ‘get  the AddrBase
Control.OutputByte wBaseAddr, 1           ' enable all DI/DO
InVal2 = Control.InputByte(wBaseAddr + &HC4)   ' input value to Port 1
```

Comment :This function is not for PCIPRX/TMC12X OCX.

3.5.5. OutputWord

This method is used to send the 16 bits data to the desired digital output port.

Prototype: void OutputWord(long nAddrBase, long nOutData)

Parameters: nAddrBase: I/O port addresses.

nOutputVal: 32 bit data send to I/O port. Only the low WORD is valid.

Comment :This function is not for PCIPRX OCX.

3.6. Analog Output Methods

Method Name	Data type of returned value	Data size	Descriptions
OutVoltage	void		Output analog voltage value.
OutCurrent	void		Output analog current value.

These methods are only for PIODAX OCX.

3.6.1. OutVoltage

This method is used to output the analog voltage data to the analog output port through the active PIO-DA board. Firstly, users need to call **Control.ActiveBoard** method to activate the selected PIO-DA board and then use this function to output analog voltage value.

Prototype: void OutVoltage (short nChannel, float fOutputValue)

Parameters: nChannel :The channel number of PIO-DA card

fOutputValue: Analog output value by float format

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0          'Set the active board to 0
Control.OutVoltage  0, 5.5  'Channel Number=0  Analog output value=5.5v.
```

Comment :This function is only for PIODAX OCX .

3.6.2. OutCurrent

This method is used to output the analog current data to the analog output port through the active PIO-DA board. Firstly, users need to call **Control.ActiveBoard** method to activate the selected board and then use this function to output analog current value. The analog output value is in float format.

Prototype: void OutCurrent (short nChannel, float fOutputValue)

Parameters: nChannel :The channel number of PIO-DA card

fOutputValue : Analog output current value by float format

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0          'Set the active board to 0
Control.OutCurrent  0, 15    'Channel Number=0  Analog output value=15 (mA).
```

Comment : This function is only for PIODAX OCX .

3.7. Analog input methods

Method Name	Data type of returned value	Data size	Descriptions
SetChannelGain	void		Select the AD channel's gain code.
AnalogIn	float	4 bytes	Get Analog input value by float format.
AnalogInHex	short	2 bytes	Get Analog input value by hex format.
AnalogInMulti	void		Get Analog input values by float format.
AnalogInMultiHex	void		Get Analog inputs value by hex format.
GetHextoFloat	float	4 bytes	Get float value converted from Hex value.

These methods are only for PISO813X OCX.

3.7.1. SetChannelGain

This method provides a function to setup AD channel gain code. This subroutine can be used to setup the AD channel of the active PISO-813 board. In order to enable this function, users need to call the **ActiveBoard** method to activate the selected PISO-813 board.

Prototype: void SetChannelGain(short nChannel, short nGainCode)

Parameters: nChannel: Channel number of PISO-813 card

nGainCode: Gain code, the value is 0 to 4.

JP2 : Bipolar

GAIN	JP1 : 10V Input Range	JP1 : 20V Input Range	GAIN2	GAIN1	GAIN0	Gain-Code
1	± 5V	± 10V	0	0	0	0
2	± 2.5V	± 5V	0	0	1	1
4	± 1.25V	± 2.5V	0	1	0	2
8	± 0.625V	± 1.25V	0	1	1	3
16	Not use	± 0.625	1	0	0	4

JP2 : Unipolar

GAIN	JP1 : 10V Input Range	JP1 : 20V Input Range	GAIN2	GAIN1	GAIN0	Gain-Code
1	0~10V	Not use	0	0	0	0
2	0~5V	Not use	0	0	1	1
4	0~2.5V	Not use	0	1	0	2
8	0~1.25V	Not use	0	1	1	3
16	0~0.625V	Not use	1	0	0	4

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0             'Set the first board to active
Control.SetChannelGain 0, 0   ' Channel :0  , +/- 10V range
V0=Control. AnalogIn 1,1        'V0: Channel 0 analog input value by float format.
```

Comment : This function is only for PISO813X OCX.

3.7.2. AnalogIn

This method performs the AD conversion by polling method. Function **Control.SetChannelGain** can be used to set up or change analog input channel and gain code. And then **Control.AnalogIn** bases on this configuration to obtain analog input value from the defined channel by the float format according to active PISO-813 board. In first, users need to call **Control.ActiveBoard** method to activate the selected PISO-813 board.

Prototype: float AnalogIn(short nJump, short nBipolar)

Parameters: nJump : 1→20V(HW default) , 0→10V

nBipolar :1→Bipolar (HW default) , 0→Unipolar

Return: Analog input value in float format.

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0             ‘Set the first board to active
Control.SetChannelGain 0, 0   ' Channel :0  , +/- 10V range
V0=Control. AnalogIn 1,1        ’V0: Channel 0 analog input value by float format.
```

Comment : This function is only for PISO813X OCX .

3.7.3. AnalogInHex

This method performs the AD conversion by polling method. Function **Control.SetChannelGain** can be used to set up or change analog input channel and gain code. And then **Control.AanlogInHex** bases on this configuration to obtain analog input value from defined channel by the hex format according to active PISO-813 board. Firstly, users need to call **Control.ActiveBoard** method to activate selected PISO-813 board.

Prototype: short AnalogInHex()

Return: Analog input value in hex format.

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0             ‘Set the first board to active
Control.SetChannelGain 0, 0   ' Channel :0  , +/- 10V range
V0=Control. AnalogInHex         ’V0: Channel 0 analog value by Hex format.
```

Comment : This function is only for PISO813X OCX .

3.7.4. AnalogInMulti

This method performs multiple AD conversions by software polling trigger during one batch time. Function **SetChannelGain** can be used to change channel or gain code, and the **AnalogInMulti** function bases on the setting of **SetChannelGain** to get AD data in float format. This function also refers to the current active PISO-813 board by using the function **Control.ActiveBoard**.

Prototype: void AnalogInMulti(short nJump20V, short nBipolar, float FAR* fAdsBuf, long lCount)

Parameters: nJump : 1→20V(HW default) , 0→10V

nBipolar : 1→Bipolar (HW default) , 0→Unipolar

fAdsBuf:Starting address of AD data buffer (16 bits), these data will be automatically computed based on the setting of the **SetChannelGain** function.

lCount :Number of AD conversions will be performed.

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0 ‘‘Set the first board to active
Control.SetChannelGain 0, 0   ' Channel :0  , +/- 10V range
Control. AnalogInMulti 1,1,fAdBuf(0), 100
‘Start the AnalogInMulti function and save data into fAdBuf
```

Comment : This function is only for PISO813X OCX.

3.7.5. AnalogInMultiHex

This method performs multiple AD conversions by software polling trigger during one batch time. Function **SetChannelGain** can be used to change channel or configuration code. And the **AnalogInMultiHex** function bases on the setting of **SetChannelGain** to get AD data in hex format. This function also refers to the current active PISO813 board by using the function **Control.ActiveBoard**.

Prototype: void AnalogInMultiHex(short FAR* wAdsBuf, long lCount)

Parameters: wAdsBuf : Starting address of AD data buffer (16 bits), these data will be automatically computed based on the setting of the **SetChannelGain** function.
lCount : Number of AD conversions will be performed.

Return: Analog input value which is automatically computed based on the setting of **SetChannelGain** .

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0      ‘Set the first board to active
Control.SetChannelGain 0, 0   ' Channel :0  , +/- 10V range
Control. AnalogInMultiHex fAdBuf(0), 100
‘Start the AnalogInMultiHex function and save data into fAdBuf
```

Comment : This function is only for PISO813X OCX.

3.7.6. GetHextoFloat

This method is used to convert the analog input value from Hex to floating format, depending on GainCode,Bipolar/Unipolar and 10V/20V settings.

Prototype: float GetHextoFloat(short nHex, short nGainCode, short nJUMP, short nBipolar);

Parameters: nHex : Hex value 0~0x0FFF

nGainCode :Gain code, the value is 0 to 4 , refer to 3.7.1

nJUMP : 1→20V(HW default) , 0→10V

nBipolar : 1→Bipolar (HW default) , 0→Unipolar

Return: float value converted from Hex value.

Example:

```
wTotalBoard = Control.DriverInit  'Initialize the driver and get the total boards
Control.ActiveBoard=0             'Set the first board to active
Control.SetChannelGain 0, 0   ' Channel :0  , +/- 10V range
V0=Control. AnalogInHex         'V0: Channel 0 analog value by Hex format
V1=Control.GetHextoFloat(V0,0,0,0)  'V1:float value converted from V0
```

Comment : This function is only for PISO813X OCX.

3.8. Interrupt Methods

Method Name	Data type of returned value	Data size	Descriptions
InstallIrq	void		Install interrupt handler for a specific IRQ.
RemoveIrq	void		Remove the IRQ service routine.
ResetIrqCount	void		Clear the counter value on the device driver for the interrupt.
GetIrqCount	long	4 bytes	Get interrupt counter value in the device driver.

These methods are for PIODAX/PIOSODIOX/PISO725X/PIODIOX OCX.

ResetIrqCount and GetIrqCount are not for PISO725X OCX.

3.8.1. InstallIrq

This method is used to install interrupt handler for a specific IRQ level n.

Prototype 1: void InstallIrq(long * hEvent, short nIrqSource, short nActiveMode)

Prototype 2: void InstallIrq(long * hEvent) for **PISO725X only**.

Parameters: hEvent : The user must use the CreateEvent function to create the event object and obtain its handle and pass the handle into this function.

nIrqSource: What the Interrupt-Source to be used ? Please refer to hardware's manual for the detail information.

Card No.	InterruptSource	Description
PIO-D48	0	PC3/PC7 from Port-2
	1	PC3/PC7 from Port-5
	2	Cout0
	3	Cout2
PIO-D56/D24	0	PC0
	1	PC1
	2	PC2
	3	PC3
PIO-D64	0	EXTIRQ
	1	EVTIRQ
	2	TMRIRQ
PIO-D96	0	P2C0
	1	P5C0
	2	P8C0
	3	P11C0
PIO-D144	0	P2C0
	1	P2C1
	2	P2C2
	3	P2C3
PISO-730(A)	0	DI0
	1	DI1
PIO-DA16/DA8/DA4	0	INT0
	1	INT1

ActiveMode: How to trigger the interrupt ?

This can be ActiveHigh or ActiveLow.

nActiveMode	Description
0	ActiveLow
1	ActiveHigh

Example:

```
wTotalBoard = Control.DriverInit 'Initialize the driver and get the total boards
Control.ActiveBoard= 0 'Set the first card to active.
Control.ResetIrqCount  'Reset interrupt count
hEvent = CreateEvent(0, False, False, 0)
Control.InstallIrq hEvent, 0, 1      'Install interrupt handle
 .....
Control.RemoveIrq   'Stop Interrupt!!
 .....
CloseHandle hEvent
dwIntCount = Control.GetIrqCount      'Get interrupt count
```

Comment : This function is for PIODAX/PIOSODIOX/PISO725X/PIODIOX OCX.

3.8.2. RemoveIrq

This method is used to remove the IRQ service routine.

Prototype: void RemoveIrq ()

Example:

```
wTotalBoard = Control.DriverInit 'Initialize the driver and get the total boards
Control.ActiveBoard= 0 'Set the first card to active.
Control.ResetIrqCount  'Reset interrupt count
hEvent = CreateEvent(0, False, False, 0)
Control.InstallIrq hEvent, 0, 1      'Install interrupt handle
.....
Control.RemoveIrq   'Stop Interrupt!!
 .....
CloseHandle hEvent
dwIntCount = Control.GetIrqCount      'Get interrupt count
```

Comment : This function is for PIODAX/PIOSODIOX/PISO725X/PIODIOX OCX.

3.8.3. ResetIrqCount

This method is used to clear the counter value on the device driver for the interrupt.

Prototype: void ResetIrqCount ()

Example:

```
wTotalBoard = Control.DriverInit 'Initialize the driver and get the total boards
Control.ActiveBoard= 0 'Set the first card to active.
Control.ResetIrqCount  'Reset interrupt count
hEvent = CreateEvent(0, False, False, 0)
Control.InstallIrq hEvent, 0, 1      'Install interrupt handle
 ..........................................
Control.RemoveIrq   'Stop Interrupt!!
 ..........................................
CloseHandle hEvent
dwIntCount = Control.GetIrqCount      'Get interrupt count
```

Comment : This function is for PIODAX/PIOSODIOX/PIODIOX OCX.

3.8.4. GetIrqCount

This method is used to get the interrupt counter value in the device driver.

Prototype: long GetIrqCount ()

Return: counter value of interrupt.

Example:

```
wTotalBoard = Control.DriverInit 'Initialize the driver and get the total boards
Control.ActiveBoard= 0 'Set the first card to active.
Control.ResetIrqCount  'Reset interrupt count
hEvent = CreateEvent(0, False, False, 0)
Control.InstallIrq hEvent, 0, 1      'Install interrupt handle
 ..........................................
Control.RemoveIrq   'Stop Interrupt!!
 ..........................................
CloseHandle hEvent
dwIntCount = Control.GetIrqCount      'Get interrupt count
```

Comment : This function is for PIODAX/PIOSODIOX/PIODIOX OCX.

3.9. Specific Interrupt Methods

Method Name	Data type of returned value	Data size	Descriptions
D48InstallIrq	void		Install interrupt handle for a specific IRQ.
D48RemoveIrq	void		Remove the IRQ service routine.
D48GetIrqCount	long	4 bytes	Get the counter value on the device driver for the interrupt.
D48Freq	long	4 bytes	Obtain the number of times which interrupt have happened.

These methods are only for PIO-D48 board.

.

Method Name	Data type of returned value	Data size	Descriptions
SaveIrqActiveFlag	void		Save active low and active high flag from the device's queue.
GetIrqActiveFlag	short	2 bytes	Get active low and active high flag from the device's queue.

These methods are only for PIO-D48 and PISO-725 boards.

3.9.1. D48InstallIrq

This method is used to install the IRQ service routine. This function supports multiple interrupt-source and the Active-Mode can be set as "Active-Low only", "Active-High only" and "Active-Low or Active-High". This function is only for PIO-D48 board.

Prototype: void D48InstallIrq(long * lHandle, short nIrqMask, short nActiveMode)

Parameters: lHandle : address of a event handle. The user must use the CreateEvent function to create the event object.

nIrqMask :what the interrupt-source to be use?

nIrqMask	Description
1	INT_CHAN_0: PC3/PC7 from Port-2
2	INT_CHAN_0: PC3/PC7 from Port-5
4	INT_CHAN_0: Cout0
8	INT_CHAN_0: Cout2

This function supports 4 interrupt-source at a time,thus users can use multi interrupt-source like 1+2+8.

nActiveMode : How to trigger the interrupt ? This can be ActiveHigh or ActiveLow.

nActiveMode	Description
1	PIOD48_ActiveLow
2	PIOD48_ActiveHigh

Example:

```
wTotalBoard = Control.DriverInit 'Initialize the driver and get the total boards
Control.ActiveBoard= 0 'Set the first card to active.
Control.ResetIrqCount  'Reset interrupt count
hEvent = CreateEvent(0, False, False, 0)
Control.D48InstallIrq hEvent, 1, 1      'Install interrupt handle
 .....
Control.D48RemoveIrq   'Stop Interrupt!!
 .....
CloseHandle hEvent
```

Comment : This function is only for PIO-D48 board.

3.9.2. D48RemoveIrq

This method is used to remove the IRQ service routine. This function is only for PIO-D48 board.

Prototype: void D48RemoveIrq()

Example:

```
wTotalBoard = Control.DriverInit 'Initialize the driver and get the total boards
Control.ActiveBoard= 0 'Set the first card to active.
Control.ResetIrqCount  'Reset interrupt count
hEvent = CreateEvent(0, False, False, 0)
Control.D48InstallIrq hEvent, 1, 1      'Install interrupt handle
 ..........................................
Control.D48RemoveIrq   'Stop Interrupt!!
 ..........................................
CloseHandle hEvent
```

Comment : This function is only for PIO-D48 board.

3.9.3. D48GetIrqCount

This method is used to obtain the **Interrupt-Counter** value in the device driver. The Interrupt-Counter will be increased (in the ISR) when the interrupt is triggered. When the interrupt setting to Active-High only or Active-Low only, some of the interrupt signal will be ignored and the Interrupt-Counter will not be increased. This function is only for PIO-D48 board.

Prototype: long D48GetIrqCount ()

Retuen:counter value.

Example:

```
wTotalBoard = Control.DriverInit 'Initialize the driver and get the total boards
Control.ActiveBoard= 0 'Set the first card to active.
Control.ResetIrqCount  'Reset interrupt count
hEvent = CreateEvent(0, False, False, 0)
Control.D48InstallIrq hEvent, 1, 1      'Install interrupt handle
 .....
Control.D48RemoveIrq   'Stop Interrupt!!
 .....
CloseHandle hEvent
dwIntCount = Control.D48GetIrqCount      'Get interrupt count value
```

Comment : This function is only for PIO-D48 board.

3.9.4. D48Freq

This method is used to measure the signal frequency. Users have to connect the signal(+) with CN1.Pin29, and connect the signal(-) with CN1.Pin19. The Counter-0 and Counter-1 will be used to measure the frequency. Therefore, users shouldn't use Counter-0 and Counter-1 for other purposes. This function is only for PIO-D48 board.

Prototype: long D48Freq ()

Return: Signal frequency value.

Example:

```
wTotalBoard = Control.DriverInit ’Initialize the driver and get the total boards
Control.ActiveBoard= 0 ‘Set the first card to active.
Control.ResetIrqCount  ‘Reset interrupt count
lFrequency = Control.D48Freq  ‘Get frequency value
```

Comment : This function is only for PIO-D48 board.

3.9.5. SaveIrqActiveFlag

This method is used to save the active low and active high flags from the device's queue. (First-in-First-out, Buffer Size: 2000 flags for High/Low).

Prototype: void SaveIrqActiveFlag ()

Example:

```
wTotalBoard = Control.DriverInit 'Initialize the driver and get the total boards
Control.ActiveBoard= 0 'Set the first card to active.
Control.ResetIrqCount  'Reset interrupt count
hEvent = CreateEvent(0, False, False, 0)
Control.D48InstallIrq hEvent, 1, 1      'Install interrupt handle
 .....
Control.D48RemoveIrq   'Stop Interrupt!!
 .....
CloseHandle hEvent
Control.SaveIrqActiveFlag   'Save interrupt active high and low flags
ActiveFlag_High=Control.GetIrqActiveFlag(1)   'Get interrupt active high flags
ActiveFlag_Low=Control.GetIrqActiveFlag(0)   'Get interrupt active low flags
```

Comment : This function is only for PIO-D48/PISO-725 boards.

3.9.6. GetIrqActiveFlag

This method is used to get the active low or active high flags from the device's queue. (First-in-First-out, Buffer Size: 2000 flags for High/Low).

.

Prototype: short GetIrqActiveFlag (short FlagNum)

Parameters: FlagNum : 0→Active Low , 1→Active Hihg

Return: active low or active high flags.

Example:

```
wTotalBoard = Control.DriverInit 'Initialize the driver and get the total boards
Control.ActiveBoard= 0 'Set the first card to active.
Control.ResetIrqCount  'Reset interrupt count
hEvent = CreateEvent(0, False, False, 0)
Control.D48InstallIrq hEvent, 1, 1      'Install interrupt handle
 .....
Control.D48RemoveIrq   'Stop Interrupt!!
 .....
CloseHandle hEvent
Control.SaveIrqActiveFlag  'Save interrupt active high and low flags
ActiveFlag_High=Control.GetIrqActiveFlag(1)  'Get interrupt active high flags
ActiveFlag_Low=Control.GetIrqActiveFlag(0)  'Get interrupt active low flags
```

Comment : This function is only for PIO-D48/PISO-725 boards.

3.10. Counter Methods

Method Name	Data type of returned value	Data size	Descriptions
Select8254	void		Select 8254 chip, which you want to programming.
SetCounter	void		Set the counter's mode and value.
ReadCounter	long	4 bytes	Read the counter value.

These methods are only for PIO-D48,PIO-D64,PIO-DA and TMC-12 boards.

Select8254 function is only for PCI-TMC12A.

ReadCounter function is not for PIO-DA board.

3.10.1. Select8254

This method is used to select the 8254 chip, which you want to programming. The valid value is between 0 and 3 . This function is only for TMC12X OCX.

Prototype: void Select8254 (short nWhich8254)

Parameters: nWhich8254 : 8254 chip you select, value : 0~3

Example:

```
wTotalBoard = Control.DriverInit 'Initialize the driver and get the total boards
Control.ActiveBoard= 0 'Set the first card to active.
Control.Select8254 0    'Select first 8254 chip
Control.SetCounter 0, 0, 40000   'counter:0   mode:0    counter value:40000
Sleep 1
wRetVal = Control.ReadCounter    ' read the counter value
```

Comment: This function is only for TMC12X OCX.

3.10.2. SetCounter

This method is used to set the mode and value of the specified counter.

Prototype: void SetCounter (short nCounterNo,sort nCounterMode,long nCounterVal)

Parameters: nCounterNo: The Counter-Number

0 – 2 : PIO-D48,PIO-DA ,PCI-TMC12A

0 – 5 :PIO-D64(0 to 2: Chip-0, 3 to 5: Chip-1)

nCounterMode: The Counter-Mode: 0 to 5

nCounterVal: The 16 bits value for the counter to counting.

only the lower WORD is valid.

Example:

```
wTotalBoard = Control.DriverInit 'Initialize the driver and get the total boards
Control.ActiveBoard= 0 'Set the first card to active.
Control.SetCounter 0, 0, 40000  'counter:0   mode:0    counter value:40000
Sleep 1
wRetVal = Control.ReadCounter   ' read the counter value
```

Comment: This function is only for PIO-D48, PIO-D64, PIO-DA and PCI-TMC12A boards.

Note: Before use **SetCounter** function in PCI -TMC12A, please use **Select8254** function firstly.

3.10.3. ReadCounter

This method is used to read the value of the specified counter.

Prototype: long ReadCounter ()

Return: counter value.

Example:

```
wTotalBoard = Control.DriverInit ’Initialize the driver and get the total boards
Control.ActiveBoard= 0 ‘Set the first card to active.
Control.SetCounter 0, 0, 40000   ‘counter:0   mode:0    counter value:40000
Sleep 1
wRetVal = Control.ReadCounter    ‘ read the counter value
```

Comment: This function is only for PIO-D48, PIO-D64 and PCI-TMC12A boards.

3.11. General Events

The PCIPRX, PIODIOX, PISODIOX, PISO725X, PISO813X, PIODAX and TMC12X have only one "Event", as shown in the following:

3.11.1. **OnError**

This event is used for default procedure and it is called when an error occurs. You could code an error massage when necessary.

Prototype: void OnError(long lErrorCode)

Parameters: lErrorCode : Error code in "long" data type.

The OCX passes this argument into the procedure. Please Refer to it for further using.

Example:

```
MsgBox "Error Code: " + Str(lErrorCode) + Chr(13) _
                    + "Error Message: " + Control.ErrorString
```