ActiveX Control (OCX) for ISA-Bus Series boards

User's Manual

[Version 1.0]

A626X OCX is for A-626/628

A8111X OCX is for A-8111

A812X OCX is for A-812 PG

A821X OCX is for A-821PGH/L

A822X OCX is for A-822PGH/L

A823X OCX is for A-823PGH/L

A826X OCX is for A-826PG

DIOX OCX is for DIO Series

ISOX OCX is for ISO Series

ISO813X OCX is for ISO-813

ISOAD32X OCX is for ISO-AD32H/L

ISODAX OCX is for ISO-DA8/DA16

ISOLDX OCX is for ISO-LDH/L

TMC10X OCX is for TMC-10

Table of Contents

1. Introduction

The 626X (A8111X, A812X, A821X, A822X, A823X A826X, DIOX, ISOX, ISO813X, ISOAD32X, ISODAX, ISOLDX, TMC10X) is an ActiveX Control component (OCX) for ISA-Bus series boards. It enables you to develop programs in a quick and easy way. Before using this driver, users need to install the A626X (A8111X, A812X, A821X, A822X, A823X A826X, DIOX, ISOX, ISO813X, ISOAD32X, ISODAX, ISOLDX, TMC10X) 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 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 ISA DAQ card for OCX.

Table1.1

OCX	Card model
626X	A-626, A-628
A8111X	A-8111
A812X	A-812 PG
A821X	A-821PGH, A-821PGL
A822X	A-822PGH, A-822PGL
A823X	A-823PGH, A-823PGL
A826X	A-826PG
DIOX	DIO-24, DIO-48, DIO-64/3, DIO-64/6, DIO-96, DIO-144, P8R8DIO
ISOX	ISO-P32C32, ISO-P64, ISO-C64, ISO-730
ISO813X	ISO-813
ISOAD32X	ISO-AD32H, ISO-AD32L
ISODAX	ISO-DA8, ISO-DA16
ISOLDX	ISO-LDH, ISO-LDL
TMC10X	TMC-10

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

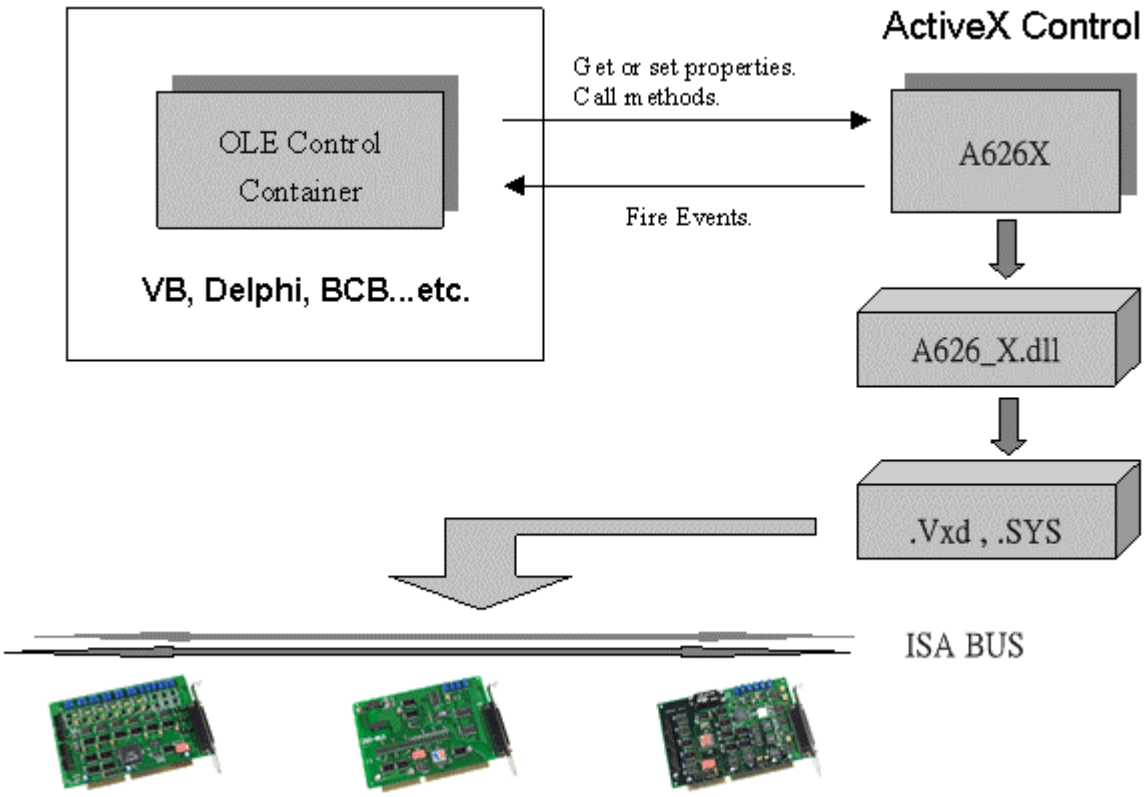

Figure 1.1

2. Data Type

Table2.1

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. Method/Property/Configuration Code Tables

3.1 Methods And Properties Of ActiveX Control

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

Table3.1

	A626X for Windows 95/98/Me/2000/XP/NT
property	short ErrorCode;
property	BSTR ErrorString;
method	void DriverInit();
method	void DriverClose();
method	short GetDLLVersion();
method	short DigitalIn(short wBase, short wCardType);
method	void DigitalOut(short wBase, short wCardType, short wHexValue);
method	void AnalogOutHex(short wBase, short wChannel, short wCardType, short wHexValue);
method	void AnalogOut(short wBase, short wChannel, short wCardType, short wJump, float fValue);

Table3.2

	A8111X for Windows 95/98/Me/2000/XP/NT
property	long ErrorCode;
property	BSTR ErrorString;
method	void DriverInit();
method	short GetDllVersion();
method	short GetDriverVersion();
method	short DigitalIn(short wBase);
method	void DigitalOut(short wBase, short wHexValue);
method	short ADPollingHex(short wBase, short wChannel, short wConfig);
method	void ADMultiPollingHex(short wBase, short wChannel, short wConfig, short* wBuf, short wCount);
method	float ADPolling(short wBase, short wChannel, short wConfig);
method	void ADMultiPolling(short wBase, short wChannel, short wConfig, float* fBuf, short wCount);
method	void ADMultiPacer(short wBase, short wChannel, short wConfig, float* fBuf, short wCount, short Counter1, short Counter2);
method	float GetHextoFloat(short wHexValue, short wConfig);
method	void AnalogOutHex(short wBase, short wHexValue);
method	void AnalogOut(short wBase, short wConfig, float fValue);
method	void DriverClose();
method	void InstallIrq(short wBase, short wIrq, long dwCount);
method	void ADIrqStart(short wChannel, short wConfig, short wCounter1, short wCounter2);
method	void RemoveIrq();
method	long GetIrqCount();

method	void GetIrqHexBuffer(long dwNum, short* wBuf);
method	void GetIrqFloatBuffer(long dwNum, float* fBuf);

Table3.3

	A812X for Windows 95/98/Me/NT
property	long ErrorCode;
property	BSTR ErrorString;
method	short GetDllVersion();
method	short GetDriverVersion();
method	short DigitalIn(short wBase);
method	void DigitalOut(short wBase, short wHexValue);
method	short ADPollingHex(short wBase, short wChannel, short wConfig);
method	void ADMultiPollingHex(short wBase, short wChannel, short wConfig, short* wBuf, short wCount);
method	float ADPolling(short wBase, short wChannel, short wConfig, short wJump);
method	void ADMultiPolling(short wBase, short wChannel, short wConfig, short wJump, float* fBuf, short wCount);
method	float GetHextoFloat(short wHexValue, short wConfig, short wJump);
method	void AnalogOutHex(short wBase, short wChannel, short wHexValue);
method	void AnalogOut(short wBase, short wChannel, short wJump, float fValue);
method	void DriverInit();
method	void DriverClose();
method	void ADIrqStart(short wChannel, short wConfig, short wJump, short wCounter1, short wCounter2);
method	void RemoveIrq();
method	long GetIrqCount();
method	void GetIrqHexBuffer(long dwNum, short* wBuf);
method	void GetIrqFloatBuffer(long dwNum, float* fBuf);
method	void InstallIrq(short wBase, short wIrq, long dwCount);

Table3.4

	A812X for Windows 2000/XP
property	long ErrorCode;
property	BSTR ErrorString;
method	short GetDllVersion();
method	short GetDriverVersion();
method	id(5)] short DigitalIn();
method	void DigitalOut(short wHexValue);
method	short ADPollingHex(short wChannel, short wConfig);
method	float ADPolling(short wChannel, short wConfig, short wJump);
method	void ADMultiPollingHex(short wChannel, short wConfig, short* wBuf, short wCount);
method	void ADMultiPolling(short wChannel, short wConfig, short wJump, float* fBuf, short wCount);
method	float GetHextoFloat(short wHexValue, short wConfig, short wJump);
method	void AnalogOutHex(short wChannel, short wHexValue);
method	void AnalogOut(short wChannel, short wJump, float fValue);

method	short DriverInit();
method	void DriverClose();
method	void ActiveBoard(short wBoardNo);
method	void InstallIrq(long* hEvent, long dwCount);
method	void ADIrqStart(short wChannel, short wConfig, short wJump, short wCounter1, short wCounter2);
method	long GetIrqCount();
method	void GetIrqHexBuffer(long dwNum, short* wBuf);
method	void GetIrqFloatBuffer(long dwNum, float* fBuf);
method	void RemoveIrq();
method	void ADIrqStop();
method	void SetCounter(short wCounterNo, short wCounterMode, long dwCounterValue);
method	long ReadCounter(short wCounterNo, short wCounterMode);

Table3.5

	A821X for Windows 95/98/Me/NT
property	long ErrorCode;
property	BSTR ErrorString;
method	void DriverInit();
method	id(4)] void DriverClose();
method	short GetDllVersion();
method	short GetDriverVersion();
method	void SetCounter(short wBase, short wCounterNo, short wCounterMode, long dwCounterValue);
method	long ReadCounter(short wBase, short wCounterNo, short wCounterMode);
method	short DigitalIn(short wBase);
method	void DigitalOut(short wBase, short wHexValue);
method	float ADPolling(short wBase, short wChannel, short wConfig, short wCardType);
method	short ADPollingHex(short wBase, short wChannel, short wConfig, short wCardType);
method	void ADMultiPollingHex(short wBase, short wChannel, short wConfig, short wCardType, short* wBuf, short wCount);
method	void ADMultiPolling(short wBase, short wChannel, short wConfig, short wCardType, float* fBuf, short wCount);
method	float GetHextoFloat(short wHexValue, short wConfig, short wCardType);
method	void AnalogOutHex(short wBase, short wHexValue);
method	void AnalogOut(short wBase, short wJump, float fValue);
method	void InstallIrq(short wBase, short wIrq, long* hEvent, long dwCount);
method	void ADIrqStart(short wChannel, short wConfig, short wCardType, short wCounter1, short wCounter2);
method	long GetIrqCount();
method	void GetIrqHexBuffer(long dwNum, short* wBuf);
method	void GetIrqFloatBuffer(long dwNum, float* fBuf);
method	void ADIrqStop();

method	void RemoveIrq();

Table3.6

	A821X for Windows 2000/XP
property	long ErrorCode;
property	BSTR ErrorString;
method	short GetDllVersion();
method	short GetDriverVersion();
method	short DigitalIn();
method	void DigitalOut(short wHexValue);
method	short ADPollingHex(short wChannel, short wConfig, short wCardType);
method	float ADPolling(short wChannel, short wConfig, short wCardType);
method	void ADMultiPollingHex(short wChannel, short wConfig, short wCardType, short* wBuf, short wCount);
method	void ADMultiPolling(short wChannel, short wConfig, short wCardType, float* fBuf, short wCount);
method	float GetHextoFloat(short wHexValue, short wConfig, short wCardType);
method	void AnalogOutHex(short wHexValue);
method	void AnalogOut(short wJump, float fValue);
method	short DriverInit();
method	void DriverClose();
method	void ActiveBoard(short wBoardNo);
method	void InstallIrq(long* hEvent, long dwCount);
method	void ADIrqStart(short wChannel, short wConfig, short wCardType, short wCounter1, short wCounter2);
method	void ADIrqStop();
method	void RemoveIrq();
method	long GetIrqCount();
method	void GetIrqHexBuffer(long dwNum, short* wBuf);
method	void GetIrqFloatBuffer(long dwNum, float* fBuf);

Table3.7

	A822X for Windows 95/98/Me/NT
property	long ErrorCode;
property	BSTR ErrorString;
method	void SetTriggerMode(short wTriggerMode);
method	void ADIrqStart(short wChannel, short wConfig, short wCardType, short wCounter1, short wCounter2);
method	void ADIrqStop();

method	void ADMultiPolling(short wBase, short wChannel, short wConfig, short wCardType, float* fBuf, short wCount);
method	void ADMultiPollingHex(short wBase, short wChannel, short wConfig, short wCardType, short* wBuf, short wCount);
method	float ADPolling(short wBase, short wChannel, short wConfig, short wCardType);
method	short ADPollingHex(short wBase, short wChannel, short wConfig, short wCardType);
method	void AnalogOut(short wBase, short wChannel, short wJump, float fValue);
method	void AnalogOutHex(short wBase, short wChannel, short wHexValue);
method	short DigitalIn(short wBase);
method	void DigitalOut(short wBase, short wHexValue);
method	void DriverClose();
method	void DriverInit();
method	short GetDllVersion();
method	short GetDriverVersion();
method	float GetHextoFloat(short wHexValue, short wConfig, short wCardType);
method	long GetIrqCount();
method	void GetIrqHexBuffer(long dwNum, short* wBuf);
method	void GetIrqFloatBuffer(long dwNum, float* fBuf);
method	void InstallIrq(short wBase, short wIrq, long* hEvent, long dwCount);
method	void RemoveIrq();
method	long ReadCounter(short wBase, short wCounterNo, short wCounterMode);
method	void SetCounter(short wBase, short wCounterNo, short wCounterMode, long dwCounterValue);

Table3.8

	A822X for Windows 2000/XP
property	long ErrorCode;
property	BSTR ErrorString;
method	void ActiveBoard(short wBoardNo);
method	void ADIrqStart(short wChannel, short wConfig, short wCardType, short wCounter1, short wCounter2);
method	void ADIrqStop();
method	void ADMultiPolling(short wChannel, short wConfig, short wCardType, float* fBuf, short wCount);
method	void ADMultiPollingHex(short wChannel, short wConfig, short wCardType, short* wBuf, short wCount);
method	float ADPolling(short wChannel, short wConfig, short wCardType);
method	short ADPollingHex(short wChannel, short wConfig, short wCardType);
method	void AnalogOut(short wChannel, short wJump, float fValue);

method	void AnalogOutHex(short wChannel, short wHexValue);
method	short DigitalIn();
method	void DigitalOut(short wHexValue);
method	void DriverClose();
method	short DriverInit();
method	short GetDllVersion();
method	short GetDriverVersion();
method	float GetHextoFloat(short wHexValue, short wConfig, short wCardType);
method	long GetIrqCount();
method	void GetIrqFloatBuffer(long dwNum, float* fBuf);
method	void GetIrqHexBuffer(long dwNum, short* wBuf);
method	void InstallIrq(long* hEvent, long dwCount);
method	void RemoveIrq();
method	void SetTriggerMode(short wTriggerMode);
method	void SetCounter(short wCounterNo, short wCounterMode, long dwCounterValue);
method	long ReadCounter(short wCounterNo, short wCounterMode);

Table3.9

	A823X for Windows 95/98/Me/NT
property	long ErrorCode;
property	BSTR ErrorString;
method	short GetDllVersion();
method	short GetDriverVersion();
method	short DigitalIn(short wBase);
method	void DigitalOut(short wBase, short wHexValue);
method	float ADPolling(short wBase, short wChannel, short wConfig, short wCardType);
method	void ADMultiPollingHex(short wBase, short wChannel, short wConfig, short wCardType, short* wBuf, short wCount);
method	void ADMultiPolling(short wBase, short wChannel, short wConfig, short wCardType, float* fBuf, short wCount);
method	void AnalogOutHex(short wBase, short wChannel, short wHexValue);
method	void AnalogOut(short wBase, short wChannel, short wJump, float fValue);
method	void DriverInit();
method	void DriverClose();
method	void InstallIrq(short wBase, short wIrq, long* hEvent, long dwCount);
method	void ADIrqStart(short wChannel, short wConfig, short wCardType, short wCounter2);
method	void ADIrqStop();

method	long GetIrqCount();
method	void GetIrqHexBuffer(long dwNum, short* wBuf);
method	void GetIrqFloatBuffer(long dwNum, float* fBuf);

Table3.10

	A823X for Windows 2000/XP
property	long ErrorCode;
property	BSTR ErrorString;
method	short GetDllVersion();
method	short GetDriverVersion();
method	short DigitalIn();
method	void DigitalOut(short wHexValue);
method	void ActiveBoard(short wBoardNo);
method	void ADIrqStart(short wChannel, short wConfig, short wCardType, short wCounter2);
method	void ADIrqStop();
method	void ADMultiPolling(short wChannel, short wConfig, short wCardType, float* fBuf, short wCount);
method	void ADMultiPollingHex(short wChannel, short wConfig, short wCardType, short* wBuf, short wCount);
method	float ADPolling(short wChannel, short wConfig, short wCardType);
method	short ADPollingHex(short wChannel, short wConfig, short wCardType);
method	void AnalogOut(short wChannel, short wJump, float fValue);
method	void AnalogOutHex(short wChannel, short wHexValue);
method	void DriverClose();
method	short DriverInit();
method	float GetHextoFloat(short wHexValue, short wConfig, short wCardType);
method	long GetIrqCount();
method	void GetIrqFloatBuffer(long dwNum, float* fBuf);
method	void GetIrqHexBuffer(long dwNum, short* wBuf);
method	void InstallIrq(long* hEvent, long dwCount);
method	void RemoveIrq();
method	void SetTriggerMode(short wTriggerMode);

Table3.11

	A826X for Windows 95/98/Me/NT
property	long ErrorCode;
property	BSTR ErrorString;
method	short GetDllVersion();
method	short GetDriverVersion();

method	short DigitalIn(short wBase);
method	void DigitalOut(short wBase, short wHexValue);
method	float ADPolling(short wBase, short wChannel, short wConfig);
method	void ADMultiPollingHex(short wBase, short wChannel, short wConfig, short* wBuf, short wCount);
method	void ADMultiPolling(short wBase, short wChannel, short wConfig, float* fBuf, short wCount);
method	void AnalogOutHex(short wBase, short wChannel, short wHexValue);
method	void AnalogOut(short wBase, short wChannel, short wJump, float fValue);
method	void DriverInit();
method	void DriverClose();
method	void InstallIrq(short wBase, short wIrq, long* hEvent, long dwCount);
method	void ADIrqStart(short wChannel, short wConfig, short wCounter1, short wCounter2);
method	void ADIrqStop();
method	long GetIrqCount();
method	void GetIrqHexBuffer(long dwNum, short* wBuf);
method	void GetIrqFloatBuffer(long dwNum, float* fBuf);

Table3.12

	A826X for Windows 2000/XP
property	long ErrorCode;
property	BSTR ErrorString;
method	void ActiveBoard(short wBoardNo);
method	void ADIrqStart(short wChannel, short wConfig, short wCounter1, short wCounter2);
method	void ADIrqStop();
method	void ADMultiPolling(short wChannel, short wConfig, float* fBuf, short wCount);
method	void ADMultiPollingHex(short wChannel, short wConfig, short* wBuf, short wCount);
method	float ADPolling(short wChannel, short wConfig);
method	short ADPollingHex(short wChannel, short wConfig);
method	void AnalogOut(short wChannel, short wJump, float fValue);
method	void AnalogOutHex(short wChannel, short wHexValue);
method	short DigitalIn();
method	void DigitalOut(short wHexValue);
method	void DriverClose();
method	short DriverInit();
method	short GetDllVersion();
method	short GetDriverVersion();
method	float GetHextoFloat(short wHexValue, short wConfig);
method	long GetIrqCount();

method	void GetIrqFloatBuffer(long dwNum, float* fBuf);
method	void GetIrqHexBuffer(long dwNum, short* wBuf);
method	void InstallIrq(long* hEvent, long dwCount);
method	void RemoveIrq();
method	void SetTriggerMode(short wTriggerMode);

Table3.13

	DIOX for Windows 95/98/Me/NT
property	long ErrorCode;
property	BSTR ErrorString;
method	short GetDllVersion();
method	void DriverInit();
method	void OutputByte(short wPortAddr, short bOutputVal);
method	void OutputWord(short wPortAddr, short wOutputVal);
method	short InputByte(short wPortAddr);
method	short InputWord(short wPortAddr);
method	void DriverClose();
method	short GetDriverVersion();
method	void InstallIrq(short wBase, short wIrq, long* hEvent);
method	void RemoveIrq();
method	long GetIrqCount();

Table3.14

	DIOX for Windows 2000/XP
property	long ErrorCode;
property	BSTR ErrorString;
method	short GetDllVersion();
method	short GetDriverVersion();
method	short DriverInit();
method	void DriverClose();
method	void ActiveBoard(short wBoardNo);
method	void OutputByte(short wPortAddr, short bOutputVal);
method	void OutputWord(short wPortAddr, short wOutputVal);
method	short InputByte(short wPortAddr);
method	short InputWord(short wPortAddr);
method	void InstallIrq(long* hEvent);
method	void RemoveIrq();
method	long GetIrqCount();

method	void D48SetCounter(short wCounterNo, short wCounterMode, long lCounterValue);
method	void D64SetCounter(short wCounterNo, short wCounterMode, long lCounterValue);
method	long D48ReadCounter(short wCounterNo, short wCounterMode);
method	long D64ReadCounter(short wCounterNo, short wCounterMode);
method	void GetConfigAddressSpace(short* wBaseAddr, short* wCurrentBoard);

Table3.15

	ISOX for Windows 95/98/Me/NT
property	long ErrorCode;
property	BSTR ErrorString;
method	short GetDllVersion();
method	short GetDriverVersion();
method	void OutputByte(short wPortAddr, short bOutputVal);
method	void OutputWord(short wPortAddr, short wOutputVal);
method	short InputByte(short wPortAddr);
method	short InputWord(short wPortAddr);
method	void DriverInit();
method	void DriverClose();
method	void InstallIrq(short wBase, short wIrq, long* hEvent);
method	void RemoveIrq();
method	long GetIrqCount();
method	void RestIrqCount();

Table3.16

	ISOX for Windows 2000/XP
property	long ErrorCode;
property	BSTR ErrorString;
method	short GetDllVersion();
method	short GetDriverVersion();
method	short DriverInit();
method	void DriverClose();
method	void GetConfigAddressSpace(short* wBaseAddr, short* wCurrentBoard);
method	void ActiveBoard(short wBoardNo);
method	void OutputByte(short wPortAddr, short bOutputVal);
method	void OutputWord(short wPortAddr, short wOutputVal);
method	short InputByte(short wPortAddr);
method	short InputWord(short wPortAddr);
method	void InstallIrq(long* hEvent);

method	void RemoveIrq();
method	long GetIrqCount();

Table3.17

	ISO813X for Windows 95/98/Me/2000/XP/NT
property	long ErrorCode;
property	BSTR ErrorString;
method	short GetDllVersion();
method	short GetDriverVersion();
method	void DriverInit();
method	void DriverClose();
method	short ADPollingHex(short wBase, short wChannel, short wConfig);
method	void ADMultiPollingHex(short wBase, short wChannel, short wConfig, short* wBuf, short wCount);
method	float ADPolling(short wBase, short wChannel, short wConfig, short wJump);
method	void ADMultiPolling(short wBase, short wChannel, short wConfig, short wJump, float* fBuf, short wCount);
method	float GetHextoFloat(short wHexValue, short wConfig, short wJump);
method	short InputByte(short wPortAddr);
method	short InputWord(short wPortAddr);
method	void OutputByte(short wPortAddr, short wOutputVal);
method	void OutputWord(short wPortAddr, short wOutputVal);

Table3.18

	ISOAD32X for Windows 95/98/Me/2000/XP/NT
property	long ErrorCode;
property	BSTR ErrorString;
method	short GetDllVersion();
method	void DriverInit();
method	void DriverClose();
method	void CheckBoard(short wBoard, short wBase, short wSingDiff, short wIrq);
method	void ActiveBoard(short wBoard);
method	float GetHextoFloat(short wHexValue, short wConfig, short wJump, short wCardType);
method	short ADPollingHex(short wChannel, short wConfig, short wJump);
method	void ADMultiPollingHex(short wChannel, short wConfig, short wJump, short* wBuf, short wCount);

Table3.19

	ISODAX for Windows 95/98/Me/2000/XP/NT
property	long ErrorCode;
property	BSTR ErrorString;
method	short GetDllVersion();
method	void DriverInit();
method	void CheckBoard(short wBoard, short wBase, short wIrq);
method	void DriverClose();
method	void ActiveBoard(short wBoard);
method	void DigitalOut(short wHexValue);
method	short DigitalIn();
method	short GetPowerOnValue(short wChannel);
method	void SetPowerOnValue(short wChannel, short wValue);
method	void AnalogOutHex(short wChannel, short wHexValue);

Table3.20

	ISOLDX for Windows 95/98/Me/2000/XP/NT
property	long ErrorCode;
property	BSTR ErrorString;
method	short GetDllVersion();
method	void SetDIMode(short bDIMode);
method	short GetDIMode();
method	short ReadbackDO();
method	short GetLatchDI();
method	void SetDABias(short wDA);
method	void SetLED(short bBase, short bDelta);
method	void GetLED(short* bBase, short* bDelta);
method	void DriverInit();
method	void CheckBoard(short wBoard, short wBase, short wIrq);
method	void ActiveBoard(short wBoard);
method	void ADMultiPacer(short wChannel, short wConfig, short wCardType, float* fBuf, short wCount, short c1);
method	void ADMultiPacerHex(short wChannel, short wConfig, short wCardType, short* wBuf, short wCount, short c1);
method	float ADPolling(short wChannel, short wConfig, short wCardType);
method	short ADPollingHex(short wChannel, short wConfig, short wCardType);
method	void ClearLatchDI(short wChannel);
method	void DriverClose();
method	short DigitalIn();

method	void DigitalOut(short wHexValue);
method	void GetConfigAddressSpace(short* wBase, short* wIrq);
method	short GetDABias();

Table3.21

	TMC10X for Windows 95/98/Me/2000/XP/NT
property	long ErrorCode;
property	BSTR ErrorString;
method	void DriverInit();
method	void DriverClose();
method	short GetDllVersion();
method	short GetDriverVersion();
method	short InputByte(short wPortAddr);
method	short InputWord(short wPortAddr);
method	void OutputByte(short wPortAddr, short bOutputVal);
method	void OutputWord(short wPortAddr, short wOutputVal);

3.2 Analog Input Signal Range Configuration Code Table

■ **A- 8111 :**

Table3.22

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	+/- 5V	0
Bipolar	+/- 2.5V	1
Bipolar	+/- 1.25V	2
Bipolar	+/- 0.625V	3
Bipolar	+/- 0.3125V	4

■ **A- 812PG :**

Table3.23

Bipolar/Unipolar	Range(JP4=5v)	Range(JP4=10v)	Configuration Code
Bipolar	+/- 5V	+/- 10V	0
Bipolar	+/- 2.5V	+/- 5V	1
Bipolar	+/- 1.25V	+/- 2.5V	2
Bipolar	+/- 0.625V	+/- 1.25V	3
Bipolar	+/- 0.3125V	+/- 0.625V	4

■ **A- 821PGL :**

Table3.24

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	+/- 5V	0
Bipolar	+/- 2.5V	1
Bipolar	+/- 1.25V	2
Bipolar	+/- 0.0625V	3

■ **A- 821PGH :**

Table3.25

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	+/- 5V	0
Bipolar	+/- 0.5V	1
Bipolar	+/- 0.05V	2
Bipolar	+/- 0.005V	3

■ **A- 822PGL :**

Table3.26

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	+/- 5V	0
Bipolar	+/- 2.5V	1
Bipolar	+/- 1.25V	2
Bipolar	+/- 0.625V	3
Unipolar	0V ~ 10V	4
Unipolar	0V ~ 5V	5
Unipolar	0V ~ 2.5V	6
Unipolar	0V ~ 1.25V	7
Bipolar	+/- 10V	8

■ **A- 822PGH :**

Table3.27

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	+/- 5V	0
Bipolar	+/- 0.5V	1
Bipolar	+/- 0.05V	2
Bipolar	+/- 0.005V	3
Unipolar	0 ~ 10V	4
Unipolar	0 ~ 1V	5
Unipolar	0 ~ 0.1V	6
Unipolar	0 ~ 0.01V	7
Bipolar	+/- 10V	8
Bipolar	+/- 1V	9
Bipolar	+/- 0.1V	10
Bipolar	+/- 0.01V	11

■ **A- 823PGL :**

Table3.28

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	+/- 5V	0
Bipolar	+/- 2.5V	1
Bipolar	+/- 1.25V	2
Bipolar	+/- 0.0625V	3
Unipolar	0V ~ 10V	4
Unipolar	0V ~ 5V	5
Unipolar	0V ~ 2.5V	6
Unipolar	0V ~ 1.25V	7
Bipolar	+/- 10V	8

■ **A- 823PGH :**

Table3.29

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	+/- 5V	0
Bipolar	+/- 0.5V	1
Bipolar	+/- 0.05V	2
Bipolar	+/- 0.005V	3
Unipolar	0 ~ 10V	4
Unipolar	0 ~ 1V	5
Unipolar	0 ~ 0.1V	6
Unipolar	0 ~ 0.01V	7
Bipolar	+/- 10V	8
Bipolar	+/- 1V	9
Bipolar	+/- 0.1V	10
Bipolar	+/- 0.01V	11

■ **A-826PG :**

Table3.30

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	+/- 10V	0
Bipolar	+/- 5V	1
Bipolar	+/- 2.5V	2
Bipolar	+/- 1.25V	3

■ **ISO-LDH :**

Table3.31: Analog input range (Channel 0):

Gain Value	Input Signal Range	Configuration Code
1	0~10V	0
10	0~1V	1
100	0~0.1V	2
1000	0~0.01V	3

Table3.32: Strain gauge input range (Channel 1):

Gain Value	Input Signal Range	Configuration Code
400	0~37.5 mV	0
4,000	0~15 mV	1
40,000	0~12.75 mV	2

■ **ISO-LDL :**

Table3.33: Analog input range (Channel 0):

Gain Value	Input Signal Range	Configuration Code
1	0~10V	0
2	0~5V	1
4	0~2.5V	2
8	0~1.25V	3

Table3.34: Strain gauge input range (Channel 1):

Gain Value	Input Signal Range	Configuration Code
400	0 ~10 mV	0
800	0 ~1 mV	1
1,600	0 ~0.1 mV	2
3,200	0~0.01 mV	3

■ **ISO-813 :**

Table3.35

Bipolar/Unipolar(JP2)	Input Range(JP1:10V)	Input Range(JP1:20V)	Configuration Code
Bipolar	+/- 5V	+/- 10V	0
Bipolar	+/- 2.5V	+/- 5V	1
Bipolar	+/- 1.25V	+/- 2.5V	2
Bipolar	+/- 0.625V	+/- 1.25V	3
Bipolar	+/- 0.3125	+/- 0.625	4
Unipolar	0~10V	Not use	0
Unipolar	0~5V	Not use	1
Unipolar	0~2.5V	Not use	2
Unipolar	0~1.25V	Not use	3
Unipolar	0~0.625V	Not use	4

■ **ISO-AD32L :**

Table3.36

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	+/- 10V	0x80
Bipolar	+/- 5V	0x90
Bipolar	+/- 2.5V	0xA0
Bipolar	+/- 1.25V	0xB0
Bipolar	+/- 0.625V	0x30
Unipolar	0V ~ 10V	0x00
Unipolar	0V ~ 5V	0x10
Unipolar	0V ~ 2.5V	0x20
Unipolar	0V ~ 1.25V	0x30

■ **ISO-AD32H :**

Table3.37

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	+/- 10V	0x80
Bipolar	+/- 5V	0x00
Bipolar	+/- 1V	0x90
Bipolar	+/- 0.5V	0x10
Bipolar	+/- 0.1V	0xA0
Bipolar	+/- 0.05V	0x20
Bipolar	+/- 0.01V	0xB0
Bipolar	+/- 0.005V	0x30
Unipolar	0V ~ 10V	0x00
Unipolar	0V ~ 1V	0x10
Unipolar	0V ~ 0.1V	0x20
Unipolar	0V ~ 0.01V	0x30

4. Interface of ISA-bus board

The interface of ISA series boards is for 626X, A8111X, A812X, A821X, A822X, A823X A826X, DIOX, ISOX, ISO813X, ISOAD32X, ISODAX, ISOLDX and TMC10X.

4.1 General Properties

Table 4.1

Property Name	Data Type	Data Size	Access mode	Run-time Only	Description
ErrorCode	long	4 Bytes	Read/Write	No	Get/set 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) A626X1 or A626X2 is for A626X OCX,

(2) A8111X1 or A8111X2 is for A8111X OCX

(3) A812X1 or A812X2 is for A812X OCX

(4) A821X1 or A821X2 is for A821X OCX

(5) A822X1 or A822X2 is for A822X OCX

(6) A823X1 or A823X2 is for A823X OCX

(7) A826X1 or A826X2 is for A826X OCX

(8) DIOX1 or DIOX2 is for DIOX OCX

(9) ISOX1 or ISOX2 is for ISOX OCX

(10) ISO813X1 or ISO813X2 is for ISO813X OCX

(11) ISOAD32X1 or ISOAD32X2 is for ISOAD32X OCX

(12) ISODAX1 or ISODAX2 is for ISODAX OCX

(13) ISOLDX1 or ISOLDX2 is for ISOLDX OCX

(14) TMC10X1 or TMC10X2 is for TMC10X OCX.

4.1.1 ErrorCode

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

- **Return:**
  (long) Error code.
- **Example:**

```
If (Control.ErrorCode <>0) Then
' Error occurs
' Do something
' Exit sub
End If
```

4.1.2 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) Error message.
- **Example:**

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

4.2 Table of ErrorCode and ErrorString

Table 4.2

Error Code	Error ID	Error String	Comment
0	**NoError**	(OK ! No Error!)	
1	**DriverOpenError**	Device driver cannot be opened.	
2	**DriverNoOpen**	Users have to call the DriverInit function firstly.	
3	**GetDriverVersionError**	Get driver version error.	
4	**InstallIrqError**	Cannot install interrupt service.	
5	**ClearIntCountError**	Cannot clear interrupt count.	
6	**GetIntCountError**	Cannot get interrupt count.	
7	**GetBufferError**	Cannot get AD buffer.	
8	**GetBoardInfError**	Cannot get board information.	
9	**ParameterError**	At least one parameter error.	
100	**AdError1**	Get AD Hex value error.	
-200	**AdError2**	Get AD float value error.	
11	**AllocateMemoryError**	Fail to allocate the memory buffer.	
12	**CardTypeError**	Card type setting error.	
13	**TimeoutError**	Time Out.	
14	**OtherError**	Other Error.	
15	**ConfigCodeError**	Invalid Configuration Code.	
16	**IntStopError**	Cannot stop interrupt service.	
17	**IntRemoveError**	Cannot remove interrupt resource.	
18	**BaseAddrError**	Base address setting error.	
19	**ReadEEPROMError**	Cannot get EEPROM data.	
20	**ActiveBoardError**	Cannot active the board.	
21	**ChScanInstallIrqError**	Cannot install interrupt service for channel scan.	
22	**HandshakeError**	Handshake error.	
23	**ChannelNoError**	Invalid channel number.	
24	**ResetIntCountError**	Cannot reset interrupt count.	
25	**CheckBoardError**	Cannot find the board on the system.	
26	**ExceedBoardNumber**	Invalid board number.	
27	**SingDiffError**	Single-end or differential setting error.	
28	**IrqNoError**	IRQ number setting error.	

In order to make the descriptions more simplified and clear, the attributes for the both the

input and output parameter of the functions are given as **[input]** and **[output]** respectively, as shown in following table.

Table 4.3

Keyword	Setting parameter by user before calling this function	Get the data/value from this parameter after calling this function
[In]	Yes	No
[Out]	No	Yes
[In, Out]	Yes	Yes

Note: All of the parameters need to be allocated spaces by the user.

4.3 General Methods

Table 4.4

Method Name	Data type of returned value	Data size	Descriptions
DriverInit	short/void	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
CheckBoard	void		Check the hardware board and active this board.
ActiveBoard	void		Activate specific board installed in the system.
DriverClose	void		Close the Driver and release the resource from the device.

4.3.1 DriverInit

This method is used to start the ISA series board 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:**

  void DriverInit()

  short DriverInit()

- **Parameters:**

  None.

- **Return:**

  None / Total boards of defined ISA series boards.

- **Example:**

```
Control.DriverInit   'Initialize the driver
Control.DigitalOut  &h220, 5   ‘Base address:&h220 , output value=5
```

4.3.2 GetDllVersion

This method is used to get the DLL version information of ISA series board driver.

- **Prototype:**

  short GetDllVersion()

- **Parameters:**

  None.

- **Return:**

  DLL version of ISA series board.

- **Example:**

```
'For example: If it returns &h250, then the version is 2.50.
wVer = Control.GetDllVersion()
```

4.3.3 GetDriverVersion

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

- **Prototype:**

  short GetDriverVersion()

- **Parameters:**

  None.

- **Return:**

  Driver version of ISA series board.

- **Example:**

```
'For example: If it returns &h200, then the version is 2.00.
wVer = Control.GetDriverVersion()
```

- **Comment :**

  This function is not for A626X, ISOAD32X, ISODAX, ISOLDX OCX.

4.3.4 CheckBoard

This function will check the hardware board. If all checks are OK, this function will active this board.

- **Prototype:**

  void CheckBoard (short wBoard,short wBase,short wSingDiff,short wIrq)

  void CheckBoard (short wBoard,short wBase, short wIrq)

- **Parameters:**

  wBoard : [In] Board number.

  wBase : [In] I/O port base address, for example: 0x220.

  wSingDiff : [In] 0 = single-ended type, Others = differential type.

  wIrq : [In] Board IRQ number.

- **Return:**

  none

- **Example:**

```
Control.DriverInit  'Initialize the driver
Control. CheckBoard  0, &h220, 0, 7   'Board 0, base address: &h220, S.E, irq=7
```

- **Comment :**

  These functions are only for ISOAD32X, ISODAX, ISOLDX OCX.

4.3.5 ActiveBoard

This method is used to activate specific board installed in the system.

- **Prototype:**

  void ActiveBoard(short wBoardNo)

- **Parameters:**

  wBoardNo : [In]Board number

- **Return:**

none

- **Example:**

```
wTotalBoard = Control.DriverInit()   'Initialize the driver and get the total boards
Control.ActiveBoard 0          'Set the active board to 0
```

- **Comment :**

This function is only for A812X, A821X, A822X, A823X A826X, DIOX, ISOX, ISOAD32X, ISODAX, ISOLDX OCX.

4.3.6 DriverClose

Stop and close the driver of ISA 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 ()

- **Parameters:**

None.

- **Return:**

None.

- **Example:**

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

4.4 Digital Input/Output Methods

Table 4.5

Method Name	Data type of returned value	Data size	Descriptions
DigitalIn	short	2 bytes	Get the 16 bits data from the digital input port.
DigitalOut	void		Send the 16 bits data to digital output port.
SetDIMode	void		Set the working mode of digital input channels.
GetDIMode	short	2 bytes	Get the working mode information of digital input channels.
ReakbackDO	short	2 bytes	Read back digital output value .
SetLED	void		Set the BASE and DELTA configuration of TTL/LED indicator.
GetLED	void		Get the AD BASE and DELTA configuration value.
ClearLatchDI	void		Clear the latch states of 8 isolated digital input channels.
GetLatchDI	short	2 bytes	Obtain the 8 isolated digital input latch states from hardware.

4.4.1 DigitalIn

This method is used to read the 16 bits data from the digital input port.

- **Prototype:**

  short DigitalIn ()

  short DigitalIn (short wBase)

  short DigitalIn (short wBase, short wCardType)

- **Parameters:**

  wBase : [In] I/O port base address, for example, 0x220

  wCardType : [In] Card type : 0→ A-626, 1→ A-628

- **Return:**

  16 bits data from digital input port

- **Example:**

```
Control.DriverInit   'Initialize the driver
Print Control.DigitalIn(&h220)     ‘Print the digital input value
```

- **Comment :**

  These functions are not for DIOX, ISOX, ISO813X, ISOAD32X, TMC10X OCX.

4.4.2 DigitalOut

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

- **Prototype:**

  void DigitalOut (short wHexValue)

  void DigitalOut (short wBase, short wHexValue)

  void DigitalOut (short wBase, short wCardType, short wHexValue)

- **Parameters:**

  wBase : [In] I/O port base address

  wCardType : [In] Card type : 0→ A-626, 1→ A-628

  wHexValue : [In] 16 bits data send to digital output port

- **Return:**

  None.

- **Example:**

```
Control.DriverInit  'Initialize the driver
Control.DigitalOut &h220, 5   ‘Digital output = 5  (0000 0101)
```

- **Comment :**

  These functions are not for DIOX, ISOX, ISO813X, ISOAD32X, TMC10X OCX.

4.4.3 SetDIMode

This mothod is used to set the working mode of 8 isolated digital input channels. The setting of Bit 0 ~7 is corresponding to the channel 0~7, respectively. If the corresponding bit

is set to 0, that means the corresponding DI channel is working on active HIGH mode. On the other hand, if the corresponding bit is set to 1, that means the corresponding DI channel is working on active LOW mode.

- **Prototype:**

  void SetDIMode (short bDIMode)

- **Parameters:**

  bDIMode: [In] Working mode of 8 isolated digital input channels

- **Return:**

  None.

- **Example:**

```
Control.DriverInit  'Initialize the driver
Control.ActiveBoard 0  ‘select action board=0
Control. SetDIMode 1  ‘Channel_0 is working on active LOW mode
```

- **Comment :**

  This function is only for ISOLDX OCX.

4.4.4 GetDIMode

This mothod is used to get the working mode information of 8 isolated digital input channels.

- **Prototype:**

  short GetDIMode ()

- **Parameters:**

  None

- **Return:**

  Working mode of 8 isolated digital input channels.

- **Example:**

```
Control.DriverInit   'Initialize the driver
```

```
Control.ActiveBoard 0   ‘select action board=0
bDIMode=Control.GetDIMode()    ‘Get DI working mode.
```

- **Comment :**

  This function is only for ISOLDX OCX.

4.4.5 ReakbackDO

This mothod is used to read back digital output value from the embedded controller of ISO-LD board.

- **Prototype:**

  short ReakbackDO ()

- **Parameters:**

  None.

- **Return:**

  Digital output value from the embedded controller of the board.

- **Example:**

```
Control.DriverInit   'Initialize the driver
Control.ActiveBoard 0  ‘select action board=0
bDIMode=Control. ReakbackDO ()   ‘Read back the O.C digital output value.
```

- **Comment :**

  This function is only for ISOLDX OCX.

4.4.6 SetLED

This mothod is used to set the BASE and DELTA configuration of TTL/LED indicator.

- **Prototype:**

void SetLED (short bBase,short bDelta)

- **Parameters:**

  bBase : [In] AD base value.

  bDelta : [In] AD delta value. bBase+bDelta*7 must small than 255

- **Return:**

  None

- **Example:**

```
Control.DriverInit  'Initialize the driver
Control.ActiveBoard 0  ‘Select action board=0
Control. SetLED  &h123, &h100)   ‘Set BASE=&h123, DELTA=&h100
```

- **Comment :**

  This function is only for ISOLDX OCX.

4.4.7 GetLED

This mothod is used to get the AD BASE and DELTA configuration value of TTL/LED indicator from the embedded controller of ISO-LD board.

- **Prototype:**

  void GetLED (short* bBase,short* bDelta)

- **Parameters:**

  bBase : [Out] Base value of AD .

  bDelta : [Out] Delta value.

- **Return:**

  None

- **Example:**

```
Control.DriverInit   'Initialize the driver
Control.ActiveBoard 0  ‘Select action board=0
Control.GetLED  bBase, bDelta   ‘Get the value of BASE and DELTA.
```

- **Comment :**

  This function is only for ISOLDX OCX.

4.4.8 ClearLatchDI

This mothod is used to clear the latch states of 8 isolated digital input channels. Bit 0 ~7 is the status of channel 0~7, respectively. bit=0 means the latch state is clear. bit=1 means the latch state is not clear.

- **Prototype:**

  void ClearLatchDI (short wChannel)

- **Parameters:**

  wChannel : [In] The latch states of 8 digital input channels.

- **Return:**

  None

- **Example:**

```
Control.DriverInit  'Initialize the driver.
Control.ActiveBoard 0  ‘Select action board=0.
Control.ClearLatchDI   ‘Clear latch DI value.
```

- **Comment :**

  This function is only for ISOLDX OCX.

4.4.9 GetLatchDI

This mothod is used to obtain the 8 isolated digital input latch states from hardware. Bit 0 ~7 depicts the status of the channel 0~7, respectively. The DI working mode introduced in user manual Sec 5.2.6 will not take effect on this function.

- **Prototype:**

short GetLatchDI ()

- **Parameters:**

  None.

- **Return:**

  8 isolated digital input latch states.

- **Example:**

```
Control.DriverInit  'Initialize the driver.
Control.ActiveBoard 0  ‘Select action board=0.
wVer=Control.GetLatchDI()   ‘Get latch DI value.
```

- **Comment :**

  This function is only for ISOLDX OCX.

4.5 Input/Output/Get Address Methods

Table 4.6

Method Name	Data type of returned value	Data size	Descriptions
GetConfigAddressSpace	void		Get the card information.
InputByte	short	2 bytes	Input the 8 bit data from the desired I/O port.
InputWord	short	2 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.

4.5.1 GetConfigAddressSpace

This method is used to get the IO base address or specified information of ISA serial boards.

- **Prototype:**

  void GetConfigAddressSpace(short* wBaseAddr, short* wCurrentBoard)

  void GetConfigAddressSpace(short* wBaseAddr, short* wIrq)

- **Parameters:**

  wBaseAddr : [Out] Base address of this board

  wCurrentBoard : [Out] Board-number of current card.

  wIrq : [Out] IRQ number of this board

- **Return:**

  None.

- **Example:**

```
wTotalBoard = Control.DriverInit()  'Initialize the driver and get the total boards
Control.ActiveBoard 0   'Set the active board to 0
Control. GetConfigAddressSpace nBaseAddr, nIrq  'Get base address and IRQ
```

- **Comment :**

These functions are only for DIOX, ISOX, ISOLDX OCX.

4.5.2 InputByte

This method is used to read the 8 bits data from the desired I/O port.

- **Prototype:**

  short InputByte(short wPortAddr)

- **Parameters:**

  wPortAddr : [In] I/O port addresses

- **Return:**

  16 bits data with the leading 8 bits are all 0

- **Example:**

```
Control.DriverInit   'Initialize the driver
nVal = Control.InputByte(wBaseAddr + &hC4)     ' Get input value from I/O port.
```

- **Comment :**

  This function is only for DIOX, ISOX, ISO813X, TMC10X OCX.

4.5.3 InputWord

This method is used to read the 16 bits data from the desired I/O port.

- **Prototype:**

  short InputWord(short wPortAddr)

- **Parameters:**

  wPortAddr : [In] I/O port address.

- **Return:**

  16 bits data from the desired I/O port.

- **Example:**

```
Control.DriverInit   'Initialize the driver
Control.OutputByte wBaseAddr, 1   ' Enable all DI/DO
nVal = Control.InputWord(wBaseAddr + &hC4)     ' Get input value from I/O port.
```

- **Comment :**

  This function is only for DIOX, ISOX, ISO813X, TMC10X OCX.

4.5.4 OutputByte

This method is used to send the 8 bits data to the desired I/O port.

- **Prototype:**

  void OutputByte(short wPortAddr, short wOutputVal)

- **Parameters:**

  wPortAddr : [In] I/O port addresses.

  wOutputVal : [In]16 bits data with the leading 8 bits are all 0

- **Return:**

  None

- **Example:**

```
Control.DriverInit  'Initialize the driver
Control.OutputByte wBaseAddr, 1          ' Enable all DI/DO
Control.OutputByte wBaseAddr+&hC4, 5   ' Output value 5 to I/O port .
```

- **Comment :**

  This function is only for DIOX, ISOX, ISO813X, TMC10X OCX.

4.5.5 OutputWord

This method is used to send the 16 bits data to the desired I/O port.

- **Prototype:**

  void OutputWord(short wPortAddr, short wOutputVal)

- **Parameters:**

  wPortAddr : [In]I/O port addresses.

  wOutputVal : [In]16 bits data send to I/O port.

- **Return:**

  None

- **Example:**

```
Control.DriverInit  'Initialize the driver
Control.OutputByte wBaseAddr, 1    ' Enable all DI/DO
Control.OutputWord wBaseAddr+&hC4, 5   ' Output value 5 to I/O port .
```

- **Comment :**

  This function is only for DIOX, ISOX, ISO813X, TMC10X OCX.

4.6 Analog Output Methods

Table 4.7

Method Name	Data type of returned value	Data size	Descriptions
AnalogOutHex	void		Output analog voltage value (hexadecimal format).
AnalogOut	void		Output analog voltage value (float value).
SetDABias	void		Set DA voltage output for DC bias adjustment.
GetDABias	short	2 bytes	Get the information of DA bias voltage output setting.
SetPowerOnValue	void		Set the power-on value of DA.
GetPowerOnValue	short	2 bytes	Read the DA power-on value.

4.6.1 AnalogOutHex

This method is used to output the analog voltage data to the analog output port. The analog output value is in hexadecimal format.

- **Prototype:**

  void AnalogOutHex (short wHexValue)

  void AnalogOutHex (short wBase, short wHexValue)

  void AnalogOutHex (short wChannel, short wHexValue)

  void AnalogOutHex (short wBase,short wChannel, short wHexValue)

  void AnalogOutHex (short wBase,short wChannel,short wCardType,short wHexValue)

- **Parameters:**

  wBase : [In] I/O port base address.

  wChannel : [In] Channel of D/A.

  wHexValue : Analog output value by hexadecimal format.

  wCardType : [In] Card type of the board. Refer to the following table4.8.

Table 4.8

Value	Card Type				
0	A-626	A-821PGL	A-822PGL	A-823PGL	ISO-LDL
1	A-628	A-821PGH	A-822PGH	A-823PGH	ISO-LDH

- **Return:**

  None

- **Example:**

```
Control.DriverInit  'Initialize the driver.
Control.AnalogOutHex &h220, 0, &hFFF
'Base address=&h220, Channel=0,  Analog output value=&hFFF.
```

- **Comment :**

  These functions are not for DIOX, ISOX, ISO813X, ISOAD32X, ISOLDX, TMC10X OCX.

4.6.2 AnalogOut

This method is used to output the analog voltage data to the analog output port through the active board. The analog output value is in float format.

- **Prototype:**

  void AnalogOut (short wJump,float fValue)

  void AnalogOut (short wBase, short wJump, float fValue)

  void AnalogOut (short wBase, short wConfig, float fValue)

  void AnalogOut (short wChannel, short wJump, float fValue)

  void AnalogOut (short wBase,short wChannel, short wJump,float fValue)

  void AnalogOut (short wBase,short wChannel,short wCardType,short wJump,float fValue)

- **Parameters:**

  wBase : [In] I/O port base address.

  wChannel : [In] Channel of D/A.

  wConfig : [In] Configuration code. Refer to section 3.2.

wCardType : [In] Card type selected. Refer to table 4.8 of section 4.6.1.

wJump : [In] Setting of Unipolar/Bipolar, Analog input range(10V/20V).

Table 4.9

	A-626/628	A-812	A -821	A -822	A -823	A -826	ISO-813	ISO-AD32
Unipolar_5V(0~5V)	0	0	0	0	0	0	0	x
Unipolar_10V(0~10V)	1	1	1	1	1	1	1	0
Bipolar_ 5V(-5~5V)	2	x	x	x	2	x	2	x
Bipolar_10V(-10~10V)	3	x	x	x	3	x	3	1

fValue : [In] Analog output value by float format

- **Return:**

None

- **Example:**

```
Control.DriverInit  'Initialize the driver.
Control.AnalogOut &h220, 0, 5
'Base address=&h220, Config=0,  Analog output value=5V.
```

- **Comment :**

These functions are not for DIOX, ISOX, ISO813X, ISODAX, ISOAD32X, ISOLDX, TMC10X OCX.

4.6.3 SetDABias

This method is used to set the 12 bits DA voltage output for DC bias adjustment.

- **Prototype:**

void SetDABias (short wDA)

- **Parameters:**

wDA: [In]Digital output value

- **Return:**

None

- **Example:**

```
Control.DriverInit  'Initialize the driver
Control.ActiveBoard 0  ‘select action board=0
Control.SetDABias 0   ‘Set DA bias to 0.
```

- **Comment :**

  This function is only for ISOLDX OCX.

4.6.4 GetDABias

This method is used to obtain the information of the 12-bit DA bias voltage output setting from the embedded controller of ISO-LD board.

- **Prototype:**

  short GetDABias ()

- **Parameters:**

  None

- **Return:**

  12-bit DA bias voltage output setting

- **Example:**

```
Control.DriverInit  'Initialize the driver
Control.ActiveBoard 0  ‘select action board=0
nVal=Control.GetDABias()   ‘Get DA bias value.
```

- **Comment :**

  This function is only for ISOLDX OCX.

4.6.5 SetPowerOnValue

This method is used to set the power-on value of D/A.

- **Prototype:**

  void SetPowerOnValue(short wChannel, short wValue)

- **Parameters:**

  wChannel : [In] D/A channel

  wValue : [In] 14-bit data send to D/A

- **Return:**

  None

- **Example:**

```
wTotalBoard=Control.DriverInit  'Initialize the driver and get total board.
Control.CheckBoard 0,&h220, 7
'Check and active board 0, base address=&h220,Irq number=7.
Control.SetPowerOnValue 0, 2   'Set power on DA value , channel=0, value=2V
```

- **Comment :**

  This function is only for ISODAX OCX.

4.6.6 GetPowerOnValue

This method is used to read the 14-bit D/A power-on value.

- **Prototype:**

  short GetPowerOnValue(short wChannel)

- **Parameters:**

  wChannel : [In] D/A channel number

- **Return:**

  14-bit power-on DA value

- **Example:**

```
wTotalBoard=Control.DriverInit   'Initialize the driver and get total board.
Control.CheckBoard 0, &h220, 7
 ‘Check and active board 0, base address=&h220, Irq number=7.
nVal=Control.GetPowerOnValue(0)   ‘Get channel 0 power on DA value.
```

- **Comment :**

  This function is only for ISODAX OCX.

4.7 Analog input methods

Table 4.10

Method Name	Data type of returned value	Data size	Descriptions
ADPolling	float	4 bytes	Perform one AD conversion by polling method.
ADMultiPolling	void		Perform a number of A/D conversions by polling.
ADPollingHex	short	2 bytes	Perform one AD conversion by polling method.
ADMultiPollingHex	void		Perform a number of A/D conversions by polling.
ADMultiPacer	void		Perform a number of A/D conversions by pacer trigger and software transfer.
ADMultiPacerHex	void		Perform a number of A/D conversions by pacer trigger and software transfer.
GetHextoFloat	float	4 bytes	Convert the analog input value from hexadecimal to floating format.

4.7.1 ADPolling

This method is used to perform one AD conversion by polling method. The analog input value is in float format.

- **Prototype:**

  float ADPolling (short wBase,short wChannel, short wConfig)

  float ADPolling (short wBase,short wChannel, short wConfig, short wJump)

  float ADPolling (short wChannel, short wConfig, short wJump)

  float ADPolling (short wBase,short wChannel, short wConfig,short wCardType)

  float ADPolling (short wChannel, short wConfig,short wCardType)

  float ADPolling (short wChannel, short wConfig)

- **Parameters:**

  wBase : [In] I/O port base address, for example, 0x220

  wChannel : [In] A/D channel number

  wConfig : [In] Configuration code. Refer to section 3.2.

wCardType : [In] Card type selected. Refer to table 4.8 of section 4.6.1.

wJump : [In] Setting of Bipolar/Unipolar and AI range (10V/20V).
Refer to table 4.9 of section 4.6.2.

- **Return:**

Analog input value in float format.

- **Example:**

```
Control.DriverInit  'Initialize the driver
V0=Control. ADPolling (&h220, 0, 0)
' Start the AD conversion by polling and save data into V0
```

- **Comment :**

These functions are not for A626X, DIOX, ISOX, ISODAX, ISOAD32X, TMC10X OCX.

4.7.2 ADMultiPolling

This method is used to perform a number of A/D conversions by polling. This subroutine is very similar to ADPolling except that this subroutine will perform wCount of conversions instead of just one conversion. The analog input values are in float format.

- **Prototype:**

void ADMultiPolling (short wBase,short wChannel, short wConfig,float* fBuf,short wCount)

void ADMultiPolling (short wBase,short wChannel, short wConfig, short wJump,float* fBuf,short wCount)

void ADMultiPolling (short wChannel, short wConfig, short wJump,float* fBuf,short wCount)

void ADMultiPolling (short wBase,short wChannel, short wConfig,short wCardType,float* fBuf,short wCount)

void ADMultiPolling (short wChannel, short wConfig,short wCardType,float* fBuf,short wCount)

void ADMultiPolling (short wChannel, short wConfig,float* fBuf,short wCount)

- **Parameters:**

wBase : [In] I/O port base address, for example, 0x220

wChannel : [In]A/D channel number.

wConfig : [In]Configuration code. Refer to section 3.2.

wCardType : [In]Card type selected. Refer to table 4.8 of section 4.6.1.

wJump : [In] Setting of Bipolar/Unipolar and AI range (10V/20V).
Refer to table 4.9 of section 4.6.2.

fBuf : [Out] Starting address of the data buffer

wCount : [In] Number of A/D conversions will be performed

- **Return:**

None

- **Example:**

```
Control.DriverInit  'Initialize the driver
Control. ADMultiPolling &h220, 0, 0, fBuf, 256
‘Start the AD conversions by polling and save data into fBuf[]
```

- **Comment :**

These functions are not for A626X, DIOX, ISOX, ISODAX, ISOAD32X, ISOLDX, TMC10X OCX.

4.7.3 ADPollingHex

This method is used to perform one AD conversion by polling method. The analog input value is in hexadecimal format.

- **Prototype:**

short ADPollingHex (short wBase,short wChannel, short wConfig)

short ADPollingHex (short wChannel, short wConfig, short wJump)

short ADPollingHex (short wBase,short wChannel, short wConfig,short wCardType)

short ADPollingHex (short wChannel, short wConfig,short wCardType)

short ADPollingHex (short wChannel, short wConfig)

- **Parameters:**

wBase : [In] I/O port base address, for example, 0x220

wChannel : [In] A/D channel number.

wConfig : [In] Configuration code. Refer to section 3.2.

wCardType : [In] Card type selected. Refer to table 4.8 of section 4.6.1.

- **Example:**

```
Control.DriverInit  'Initialize the driver
nVal=Control. ADPollingHex (&h220, 0, 0)
'Start the AD conversion by polling and save data into nVal
```

- **Comment :**

These functions are not for A626X, DIOX, ISOX, ISODAX, TMC10X OCX.

4.7.4 ADMultiPollingHex

This method is used to perform a number of A/D conversions by polling. This subroutine is very similar to ADPollingHex except that this subroutine will perform wCount of conversions instead of just one conversion. The analog input values are in hexadecimal format.

- **Prototype:**

void ADMultiPollingHex (short wBase,short wChannel, short wConfig,short* wBuf, short wCount)

void ADMultiPollingHex (short wChannel, short wConfig, short wJump,short* wBuf, short wCount)

void ADMultiPollingHex (short wBase,short wChannel, short wConfig,short wCardType,short* wBuf,short wCount)

void ADMultiPollingHex (short wChannel, short wConfig,short wCardType, short* wBuf,short wCount)

void ADMultiPollingHex (short wChannel, short wConfig,short* wBuf,short wCount)

- **Parameters:**

wBase : [In] I/O port base address, for example, 0x220

wChannel : [In] A/D channel number.

wConfig : [In] Configuration code. Refer to section 3.2.

wCardType : [In] Card type selected. Refer to table 4.8 of section 4.6.1.

wJump : [In] Setting of Bipolar/Unipolar and AI range (10V/20V).
Refer to table 4.9 of section 4.6.2.

wBuf : [Out] Starting address of the data buffer

wCount : [In] Number of A/D conversions will be performed

- **Return:**

None

- **Example:**

```
Control.DriverInit  'Initialize the driver
Control. ADMultiPollingHex &h220, 0, 0, wBuf, 256
'Start the AD conversions by polling and save data into wBuf[].
```

- **Comment :**

These functions are not for A626X, DIOX, ISOX, ISODAX,ISOLDX, TMC10X OCX.

4.7.5 ADMultiPacer

This method is used to perform a number of A/D conversions by pacer trigger and software transfer. This subroutine is very similar to ADMultiPolling() except that this method will trigger the AD converter by the Pacer .The A/D data are stored in a data buffer in float format. The **fBuf** is the starting address of this data buffer.

- **Prototype:**

void ADMultiPacer (short wBase,short wChannel, short wConfig,float* fBuf,short wCount,short wCounter1,short wCounter2)

void ADMultiPacer (short wChannel, short wConfig,short wCardType float* fBuf,short wCount,short wCounter1)

- **Parameters:**

  wBase : [In] I/O port base address, for example, 0x220

  wChannel : [In] A/D channel number.

  wConfig : [In] Configuration code. Refer to section 3.2.

  wCardType : [In] Card type selected. Refer to table 4.8 of section 4.6.1.

  fBuf : [Out] Starting address of the data buffer

  wCount : [In] Number of A/D conversions will be performed

  wCounter1 : [In] The value of Counter 1

  wCounter2 : [In] The value of Counter 2

Sampling rate = Clock Source / (wCounter1 * wCounter2)

Table 4.11

Card	A-8111	A-812	A-821	A-822	A-823	A-826	ISO-LD
Clock source	2MHz	2MHz	2MHz	2MHz	2MHz	2MHz	1MHz

- **Return:**

  None

- **Example:**

```
Control.DriverInit  'Initialize the driver
Control. ADMultiPacer &h220, 0, 0, fBuf, 256, 100, 20
 'Start the AD conversions by pacer trigger and save data into fBuf[]
```

- **Comment :**

  These functions are only for A8111X, ISOLDX OCX.

4.7.6 ADMultiPacerHex

This method is used to perform a number of A/D conversions by pacer trigger and software transfer. This subroutine is very similar to ADMultiPollingHex() except that this method will trigger the AD converter by the Pacer .The A/D data are stored in a data buffer in hexadecimal format. The **wBuf** is the starting address of this data buffer.

- **Prototype:**

  void ADMultiPacerHex (short wChannel, short wConfig,short wCardType,short* wBuf, short wCount,short wCounter1)

- **Parameters:**

  wChannel : [In] A/D channel number.

  wConfig : [In] Configuration code. Refer to section 3.2.

  wCardType : [In] Card type selected. Refer to table 4.8 of section 4.6.1.

  wBuf : [Out] Starting address of the data buffer

  wCount : [In] Number of A/D conversions will be performed

  wCounter1 : [In] The value of Counter 1

  Sampling rate = Clock Source / wCounter1.

  (Refer to table 4.11 of section 4.7.5.)

- **Return:**

  None

- **Example:**

```
Control.DriverInit  'Initialize the driver
Control. ADMultiPacerHex  &h220, 0, 0, wBuf, 256, 100, 20
'Start the AD conversions by pacer trigger and save data into wBuf[]
```

- **Comment :**

  This function is only for ISOLDX OCX .

4.7.7 GetHextoFloat

This method is used to convert the analog input value from hexadecimal to floating

format, depending on GainCode,Bipolar/Unipolar and 10V/20V settings.

- **Prototype:**

  float GetHextoFloat(short wHexValue, short wConfig);

  float GetHextoFloat(short wHexValue, short wConfig,short wJump);

  float GetHextoFloat(short wHexValue, short wConfig,short wCardType);

  float GetHextoFloat(short wHexValue, short wConfig,short wJump,short wCardType);

- **Parameters:**

  wHexValue : [In] Hexadecimal Value.

  wConfig : [In] Configuration code. Refer to section 3.2.

  wCardType : [In] Card type selected. Refer to table 4.8 of section 4.6.1.

  wJump : [In] Setting of Bipolar/Unipolar and AI range (10V/20V).
  Refer to table 4.9 of section 4.6.2.

- **Return:**

  float value converted from hexadecimal value.

- **Example:**

```
Control.DriverInit   'Initialize the driver
fVal=Control.GetHextoFloat (&hFFF, 0)
‘Transfer the value in Hex format(&hFFF) to float format(fVal).
```

- **Comment :**

  These functions are not for A626X, DIOX, ISOX, ISODAX, ISOLDX, TMC10X OCX .

4.8 Interrupt Methods

Table 4.12

Method Name	Data type of returned value	Data size	Descriptions
SetTriggerMode	void		Set the trigger mode for internal or external.
InstallIrq	void		Install interrupt handler for a specific IRQ.
RemoveIrq	void		Remove the IRQ service routine.
GetIrqCount	long	4 bytes	Get interrupt counter value in the device driver.
ResetIrqCount	void		Clear the counter value on the device driver for the interrupt.
GetIrqHexBuffer	void		Copy the transferred interrupted data into the user's buffer(Hex format).
GetIrqFloatBuffer	void		Copy the transferred interrupted data into the user's buffer(float format).
ADIrqStart	void		Start the interrupt transfer.
ADIrqStop	void		Stop the interrupt transfer.

4.8.1 SetTriggerMode

This method is used to set the trigger mode for internal or external. The default value is setting to internal trigger mode if the user does not use this function.

- **Prototype:**

  void SetTriggerMode (short wTriggerMode)

- **Parameters:**

  wTriggerMode : [In] 0 : Internal Trigger Mode

  1 : External Trigger Mode

- **Return:**

  None

- **Example:**

```
Control.DriverInit  'Initialize the driver
Control.SetTriggerMode 0   ‘Set internal trigger mode.
```

- **Comment :**

This function is only for A822X, A823X, A826X OCX .

4.8.2 InstallIrq

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

- **Prototype :**

void InstallIrq(short wBase,short wIrq,long dwCount)

void InstallIrq(long* hEvent,long dwCount)

void InstallIrq(short wBase, short wIrq, long* hEvent, long dwCount)

void InstallIrq(long* hEvent)

- **Parameters:**

wBase : [In] I/O port base address, for example, 0x220

wIrq : [In] IRQ channel number.

hEvent : [In] Address of a Event handle. The user's program must call the Windows API function "CreateEvent()" to create the event-object.

dwCount :[In] The desired A/D entries count for interrupt transfer.

- **Return:**

None

- **Example:**

```
Control.DriverInit  ’Initialize the driver
hEvent = CreateEvent( NULL, TRUE, FALSE, NULL )
Control .InstallIrq wBase, nIRQ, hEvent, DataNo  ‘ Install the interrupt
  ..........................................
Control.RemoveIrq   ‘Stop Interrupt!!
 ..........................................
```

- **Comment :**

  These functions are not for A626X, ISO813X, ISOAD32X, ISODAX, ISOLDX, TMC10X OCX.

4.8.3 RemoveIrq

This method is used to remove the IRQ service routine.

- **Prototype:**

  void RemoveIrq ()

- **Parameters:**

  None

- **Return:**

  None

- **Example:**

```
Control.DriverInit  'Initialize the driver
hEvent = CreateEvent( NULL, TRUE, FALSE, NULL )
Control .InstallIrq wBase, nIRQ, hEvent, DataNo  ' Install the interrupt
  ..........................................
Control.RemoveIrq   'Stop Interrupt!!
 ..........................................
```

- **Comment :**

  This function is not for A626X, ISO813X, ISOAD32X, ISODAX, ISOLDX, TMC10X OCX.

4.8.4 GetIrqCount

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

- **Prototype:**

  long GetIrqCount ()

- **Parameters:**

  None

- **Return:**

  The counter-value of interrupt transferred.

- **Example:**

```
Control.DriverInit   'Initialize the driver
hEvent = CreateEvent( NULL, TRUE, FALSE, NULL )
Control .InstallIrq wBase, nIRQ, hEvent, DataNo  ' Install the interrupt
  ..........................................
Control.RemoveIrq   'Stop Interrupt!!
 ..........................................
CloseHandle hEvent
dwIntCount = Control.GetIrqCount ()     'Get interrupt count
```

- **Comment :**

  This function is only for A8111X, A812X, A821X, A822X, A823X, A826X, DIOX, ISOX OCX.

4.8.5 ResetIrqCount

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

- **Prototype:**

  void ResetIrqCount ()

- **Parameters:**

  None

- **Return:**

  None

- **Example:**

```
Control.DriverInit   'Initialize the driver
Control .ResetIrqCount   'Reset interrupt count
hEvent = CreateEvent( NULL, TRUE, FALSE, NULL )
Control .InstallIrq  wBase, nIRQ, hEvent, DataNo  'Install the interrupt
  ..........................................
```

- **Comment :**

  This function is only for ISOX OCX.

4.8.6 GetIrqHexBuffer

This method is used to copy the transferred interrupted data into the user's buffer.

- **Prototype:**

  void GetIrqHexBuffer (long dwNum,short* wBuf)

- **Parameters:**

  dwNum: [In] Data number to be copied.

  wBuf : [Out] The address of wBuf(In short format).

- **Return:**

  None

- **Example:**

```
Control.DriverInit   'Initialize the driver
hEvent = CreateEvent( NULL, TRUE, FALSE, NULL )
Control .InstallIrq  wBase, nIRQ, hEvent, DataNo  'Install the interrupt
  ……..
Control .ADIrqStart  wADChNo, wCfgCode,wCardType, c1, c2
 ……..
Control.ADIrqStop
```

```
Control.GetIrqHexBuffer  DataNo, wBuf   'Copy the data buffer from device driver
```

- **Comment :**

  This function is only for A8111X, A812X, A821X, A822X, A823X, A826X OCX.

4.8.7 GetIrqFloatBuffer

This method is used to copy the transferred interrupted data into the user's buffer.

- **Prototype:**

  void GetIrqHexBuffer (long dwNum,float* fBuf)

- **Parameters:**

  dwNum: [In] Data number to be copied.

  fBuf : [Out] The address of fBuf (In float format).

- **Return:**

  None

- **Example:**

```
Control.DriverInit   'Initialize the driver
Control .InstallIrq  wBase, nIRQ, hEvent, DataNo   'Install the interrupt
 ……..
Control .ADIrqStart  wADChNo, wCfgCode,wCardType, c1, c2
 ……..
Control.ADIrqStop
Control. GetIrqFloatBuffer  DataNo, fBuf 'Copy the data buffer from device driver
```

- **Comment :**

  This function is only for A8111X, A812X, A821X, A822X, A823X, A826X OCX.

4.8.8 ADIrqStart

This method is used to start the interrupt transfer for a specific A/D channel and programming the gain code and sampling rate.

- **Prototype:**

  void ADIrqStart (short wChannel,short wConfig,short wCounter1,short wCounter2)

  void ADIrqStart (short wChannel,short wConfig,short wJump,short wCounter1,
  short wCounter2)

  void ADIrqStart (short wChannel,short wConfig, short wCardType,short wCounter1,
  short wCounter2)

  void ADIrqStart (short wChannel,short wConfig,short wCardType,short wCounter2)

- **Parameters:**

  wChannel : [In] A/D channel number.

  wConfig : [In] Configuration code. Refer to section 3.2.

  wCardType : [In] Card type selected. Refer to table 4.8 of section 4.6.1.

  wJump : [In] Setting of Bipolar/Unipolar and AI range (10V/20V).
  Refer to table 4.9 of section 4.6.2.

  wCounter1 : [In] The value of Counter 1

  wCounter2 : [In] The value of Counter 2

  Sampling rate = Clock Source / (wCounter1 * wCounter2) .

  (Refer to table 4.11 of section 4.7.5.)

- **Return:**

  None

- **Example:**

```
Control.DriverInit   'Initialize the driver
hEvent = CreateEvent( NULL, TRUE, FALSE, NULL )
Control .InstallIrq  wBase, nIRQ, hEvent, DataNo  'Install the interrupt
 ……..
Control .ADIrqStart  wADChNo, wCfgCode,wCardType, c1, c2
 ……..
Control.ADIrqStop
```

- **Comment :**

  These functions are only for A8111X, A812X, A821X, A822X, A823X, A826X OCX.

4.8.9 ADIrqStop

This method is used to stop the interrupt transfer.

- **Prototype:**

  void ADIrqStop ()

- **Parameters:**

  None

- **Return:**

  None

- **Example:**

```
Control.DriverInit   'Initialize the driver
 ……..
Control .ADIrqStart  wADChNo, wCfgCode, wCardType, c1, c2
 ……..
Control.ADIrqStop
```

- **Comment :**

  This function is only for A812X, A821X, A822X, A823X, A826X OCX .

4.9 Counter Methods

Table 4.13

Method Name	Data type of returned value	Data size	Descriptions
SetCounter	void		Set the counter's mode and value.
ReadCounter	long	4 bytes	Read the counter value.

4.9.1 SetCounter

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

- **Prototype:**

  void SetCounter (short wCounterNo,short wCounterMode,long dwCounterVal)

  void SetCounter (short wBase,short wCounterNo,short wCounterMode,long dwCounterVal)

  void D48SetCounter (short wCounterNo,short wCounterMode,long dwCounterVal)

  void D64SetCounter (short wCounterNo,short wCounterMode,long dwCounterVal)

- **Parameters:**

  wBase : [In] I/O port base address.

  wCounterNo : [In] 8254 chip's counter number.

  wCounterMode : [In] The counter mode : 0 to 5

  wCounterVal : [In] The 16 bits value for the counter to counting. Only the lower WORD is valid.

- **Return:**

  None

- **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 :**

  These functions are only for A812X, A822X, DIOX OCX.

4.9.2 ReadCounter

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

- **Prototype:**

  long ReadCounter (short wCounterNo,short wCounterMode)

  long ReadCounter (short wBase,short wCounterNo,short wCounterMode)

  long D48 ReadCounter (short wCounterNo,short wCounterMode)

  long D64 ReadCounter (short wCounterNo,short wCounterMode)

- **Parameters:**

  wBase : [In] I/O port base address.

  wCounterNo : [In] 8254 chip's counter number.

  wCounterMode : [In] The counter mode : 0 to 5

- **Return:**

  The specified counter's value. (Only low-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 :**

These functions are only for A812X, A822X, DIOX OCX .

5. General Events

The A626X A8111X, A812X, A821X, A822X, A823X A826X, DIOX, ISOX, ISO813X, ISOAD32X, ISODAX, ISOLDX and TMC10X have only one "Event", as shown in the following:

5.1 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.

- **Example:**

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