

User's Guide



Shop online at

www.omega.com

e-mail: info@omega.com



OME-PISO-813

PCI Data Acquisition Board

Windows Software Manual



OMEGAnet® Online Service
www.omega.com

Internet e-mail
info@omega.com

Servicing North America:

USA:
ISO 9001 Certified

One Omega Drive, P.O. Box 4047
Stamford CT 06907-0047
TEL: (203) 359-1660 FAX: (203) 359-7700
e-mail: info@omega.com

Canada:

976 Bergar
Laval (Quebec) H7L 5A1, Canada
TEL: (514) 856-6928 FAX: (514) 856-6886
e-mail: info@omega.ca

For immediate technical or application assistance:

USA and Canada: Sales Service: 1-800-826-6342 / 1-800-TC-OMEGA®
Customer Service: 1-800-622-2378 / 1-800-622-BEST®
Engineering Service: 1-800-872-9436 / 1-800-USA-WHEN®
TELEX: 996404 EASYLINK: 62968934 CABLE: OMEGA

Mexico:

En Español: (001) 203-359-7803 e-mail: espanol@omega.com
FAX: (001) 203-359-7807 info@omega.com.mx

Servicing Europe:

Benelux:

Postbus 8034, 1180 LA Amstelveen, The Netherlands
TEL: +31 (0)20 3472121 FAX: +31 (0)20 6434643
Toll Free in Benelux: 0800 0993344
e-mail: sales@omegaeng.nl

Czech Republic:

Frystatska 184, 733 01 Karviná, Czech Republic
TEL: +420 (0)59 6311899 FAX: +420 (0)59 6311114
Toll Free: 0800-1-66342 e-mail: info@omegashop.cz

France:

11, rue Jacques Cartier, 78280 Guyancourt, France
TEL: +33 (0)1 61 37 29 00 FAX: +33 (0)1 30 57 54 27
Toll Free in France: 0800 466 342
e-mail: sales@omega.fr

Germany/Austria:

Daimlerstrasse 26, D-75392 Deckenpfronn, Germany
TEL: +49 (0)7056 9398-0 FAX: +49 (0)7056 9398-29
Toll Free in Germany: 0800 639 7678
e-mail: info@omega.de

United Kingdom:

ISO 9002 Certified

One Omega Drive, River Bend Technology Centre
Northbank, Irlam, Manchester
M44 5BD United Kingdom
TEL: +44 (0)161 777 6611 FAX: +44 (0)161 777 6622
Toll Free in United Kingdom: 0800-488-488
e-mail: sales@omega.co.uk

It is the policy of OMEGA to comply with all worldwide safety and EMC/EMI regulations that apply. OMEGA is constantly pursuing certification of its products to the European New Approach Directives. OMEGA will add the CE mark to every appropriate device upon certification.

The information contained in this document is believed to be correct, but OMEGA Engineering, Inc. accepts no liability for any errors it contains, and reserves the right to alter specifications without notice.

WARNING: These products are not designed for use in, and should not be used for, patient-connected applications.

Table of Contents

1. INTRODUCTION.....	4
1.1 REFERENCE	4
1.2 A/D GAIN CODE TABLE.....	5
2. DECLARATION FILES	6
2.1 PISO813.H	7
2.2 PISO813.BAS	9
2.3 PISO813.PAS.....	12
3. DESCRIPTIONS OF FUNCTIONS.....	16
3.1 TEST FUNCTIONS	16
3.1.1 PISO813_GetDllVersion	16
3.1.2 PISO813_ShortSub	17
3.1.3 PISO813_FloatSub	17
3.2 I/O FUNCTIONS	18
3.2.1 PISO813_OutputByte	18
3.2.2 PISO813_InputByte	18
3.2.3 PISO813_OutputWord	19
3.2.4 PISO813_InputWord	19
3.3 DRIVER FUNCTIONS	20
3.3.1 PISO813_GetDriverVersion	20
3.3.2 PISO813_DriverInit.....	20
3.3.3 PISO813_DriverClose	21
3.3.4 PISO813_GetConfigAddressSpace	21
3.4 A/D FUNCTIONS	22
3.4.1 PISO813_SetChGain	22
3.4.2 PISO813_AD2F.....	22
3.4.3 PISO813_AD_Hex.....	23
3.4.4 PISO813_AD_Float.....	23
3.4.5 PISO813_ADs_Hex	24
3.4.6 PISO813_ADs_Float.....	25
4. PROGRAM ARCHITECTURE.....	26
5. REPORTING PROBLEM	27

1. Introduction

The OME-PISO-813 driver software is a collection of digital I/O and Analog-Input subroutines for the OME-PISO-813 data acquisition cards for Windows 95/98/NT/ME/200/XP applications. These subroutines are written in the C language and perform a variety of analog and digital I/O operations.

The PISO813.DLL includes easy-to-use subroutines for developing data acquisition applications. A program can call these DLL functions from VC++, VB, Delphi, and BORLAND C++ Builder. To simplify the development process, several demonstration programs are provided.

1.1 Reference

Please refer to the following user manuals for additional documentation on the OME-PISO-813:

- **PnPInstall.pdf:**
Describes how to install the PnP (Plug and Play) driver for PCI cards under Windows 95/98.
- **SoftInst.pdf:**
Describes how to install the software.
- **CallDll.pdf:**
Describes how to call the DLL functions with VC++5, VB5, Delphi3 and Borland C++ Builder 3.
- **ResCheck.pdf:**
Describes how to check the I/O Port address, IRQ number and DMA number for add-on cards under Windows.

1.2 A/D Gain Code table

OME-PISO-813 Bipolar mode GAIN CONTROL CODE TABLE

- JP2 : Bipolar

GAIN	JP1 : 10V Input Range	JP1 : 20V Input Range	GAIN2	GAIN1	GAIN0	Gain-Code
1	± 5V	± 10V	0	0	0	0x0
2	± 2.5V	± 5V	0	0	1	0x1
4	± 1.25V	± 2.5V	0	1	0	0x2
8	± 0.625V	± 1.25V	0	1	1	0x3
16	Not used	± 0.625	1	0	0	0x4

OME-PISO-813 Unipolar mode GAIN CONTROL CODE TABLE

- JP2 : Unipolar

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

2. Declaration Files

For Borland C++ Builder 3

PISO813.H ← Header file

PISO813.LIB ← Linkage library for BCB3 only

For Delphi 3

PISO813.PAS ← Declaration file

For Visual Basic

PISO813.BAS ← Declaration file

For Visual C++ 5

PISO813.H ← Header file

PISO813.LIB ← Linkage library for VC5 only

2.1 PISO813.H

```

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

/***** define PISO813 relative address *****/
#define PISO813_AD_LO      0xd0 // Analog to Digital, Low Byte
#define PISO813_AD_HI      0xd4 // Analog to Digital, High Byte
#define PISO813_SET_CH     0xE0 // channel selecting
#define PISO813_SET_GAIN   0xE4 // PGA gain code
#define PISO813_SOFT_TRIG 0xF0 // A/D trigger control register

/***** define the gain mode *****/
#define PISO813_BI_1       0x00
#define PISO813_BI_2       0x01
#define PISO813_BI_4       0x02
#define PISO813_BI_8       0x03
#define PISO813_BI_16      0x04

#define PISO813_UNI_1       0x00
#define PISO813_UNI_2       0x01
#define PISO813_UNI_4       0x02
#define PISO813_UNI_8       0x03
#define PISO813_UNI_16      0x04

/***** return code *****/
#define PISO813_NoError      0
#define PISO813_DriverOpenError  1
#define PISO813_DriverNoOpen    2
#define PISO813_GetDriverVersionError 3
#define PISO813_CallDriverError  4
#define PISO813_FindBoardError   5
#define PISO813_ExceedBoardNumber 6
#define PISO813_TimeOutError     0xffff
#define PISO813_AdError2        -100.0

// ID
#define PISO_813                0x800A00

```

// Test functions

```
EXPORTS float      CALLBACK PISO813_FloatSub(float fA, float fB);
EXPORTS short     CALLBACK PISO813_ShortSub(short nA, short nB);
EXPORTS WORD      CALLBACK PISO813_GetDllVersion(void);
```

// Driver functions

```
EXPORTS WORD      CALLBACK PISO813_DriverInit(void);
EXPORTS void      CALLBACK PISO813_DriverClose(void);
EXPORTS WORD      CALLBACK PISO813_SearchCard(WORD *wBoards,
        DWORD dwPIOCardID);
EXPORTS WORD      CALLBACK PISO813_GetDriverVersion(
        WORD *wDriverVersion);
EXPORTS WORD      CALLBACK PISO813_GetConfigAddressSpace(
        WORD wBoardNo, DWORD *wAddrBase, WORD *wIrqNo,
        WORD *wSubVendor, WORD *wSubDevice, WORD *wSubAux,
        WORD *wSlotBus, WORD *wSlotDevice);
```

// DIO functions

```
EXPORTS void      CALLBACK PISO813_OutputWord(
        DWORD wPortAddress, DWORD wOutData);
EXPORTS void      CALLBACK PISO813_OutputByte(DWORD wPortAddr,
        WORD bOutputValue);
EXPORTS DWORD     CALLBACK PISO813_InputWord(
        DWORD wPortAddress);
EXPORTS WORD      CALLBACK PISO813_InputByte(DWORD wPortAddr);
```

// AD functions

```
EXPORTS WORD      CALLBACK PISO813_SetChGain(
        DWORD wAddrBase, WORD wChannel , WORD wGainCode);
EXPORTS WORD      CALLBACK PISO813_AD_Hex(DWORD wAddrBase);
EXPORTS WORD      CALLBACK PISO813_ADs_Hex(DWORD wAddrBase,
        WORD *wBuffer, DWORD dwDataNo);
EXPORTS float     CALLBACK PISO813_AD_Float(DWORD wAddrBase,
        WORD wJump20v, WORD wBipolar);
EXPORTS float     CALLBACK PISO813_ADs_Float(DWORD wAddrBase,
        WORD wJump20v, WORD wBipolar, float *fBuffer, DWORD dwDataNo);
EXPORTS float     CALLBACK PISO813_AD2F(WORD whex,
        WORD wGainCode, WORD wJump20v, WORD wBipolar);
```


2.2 PISO813.BAS

Attribute VB_Name = "PISO813"

```

'***** define PISO813 relative address *****/
Global Const PISO813_AD_LO      = &HD0 'Analog to Digital, Low Byte
Global Const PISO813_AD_HI      = &HD4 'Analog to Digital, High Byte
Global Const PISO813_SET_CH     = &HE0 'channel selecting
Global Const PISO813_SET_GAIN   = &HE4 'PGA gain code
Global Const PISO813_SOFT_TRIG  = &HF0 'A/D trigger control register
'***** define the gain mode *****/
Global Const PISO813_BI_1       = &H0
Global Const PISO813_BI_2       = &H1
Global Const PISO813_BI_4       = &H2
Global Const PISO813_BI_8       = &H3
Global Const PISO813_BI_16      = &H4

Global Const PISO813_UNI_1      = &H0
Global Const PISO813_UNI_2      = &H1
Global Const PISO813_UNI_4      = &H2
Global Const PISO813_UNI_8      = &H3
Global Const PISO813_UNI_16     = &H4
'***** return code *****/
Global Const PISO813_NoError     = 0
Global Const PISO813_DriverOpenError = 1
Global Const PISO813_DriverNoOpen = 2
Global Const PISO813_GetDriverVersionError = 3
Global Const PISO813_CallDriverError = 4
Global Const PISO813_FindBoardError = 5
Global Const PISO813_ExceedBoardNumber = 6
Global Const PISO813_TimeOutError = &HFFFF
Global Const PISO813_AdError2    = -100#

' ID
Global Const PISO_813            = &H800A00

```

' The Test functions

Declare Function PISO813_ShortSub Lib "PISO813.dll" (ByVal a As Integer,
ByVal b As Integer) As Integer

Declare Function PISO813_FloatSub Lib "PISO813.dll" (ByVal a As Single,
ByVal b As Single) As Single

Declare Function PISO813_GetDllVersion Lib "PISO813.dll" () As Integer

' The Driver functions

Declare Function PISO813_DriverInit Lib "PISO813.dll" () As Integer

Declare Sub PISO813_DriverClose Lib "PISO813.dll" ()

Declare Function PISO813_SearchCard Lib "PISO813.dll" (_
wBoards As Integer, ByVal dwPIOCardID As Long) As Integer

Declare Function PISO813_GetDriverVersion Lib "PISO813.dll" (_
wDriverVersion As Integer) As Integer

Declare Function PISO813_GetConfigAddressSpace Lib "PISO813.dll" (_
ByVal wBoardNo As Integer, wAddrBase As Long, wIrqNo As Integer, _
wSubVendor As Integer, wSubDevice As Integer, wSubAux As Integer, _
wSlotBus As Integer, wSlotDevice As Integer) As Integer

Declare Function PISO813_ActiveBoard Lib "PISO813.dll" (_
ByVal wBoardNo As Integer) As Integer

Declare Function PISO813_WhichBoardActive Lib "PISO813.dll" () As Integer

' DIO functions

Declare Sub PISO813_OutputByte Lib "PISO813.dll" (_
ByVal address As Long, ByVal dataout As Integer)

Declare Sub PISO813_OutputWord Lib "PISO813.dll" (_
ByVal address As Long, ByVal dataout As Long)

Declare Function PISO813_InputByte Lib "PISO813.dll" (_
ByVal address As Long) As Integer

Declare Function PISO813_InputWord Lib "PISO813.dll" (_
ByVal address As Long) As Long

' AD functions

Declare Function PISO813_SetChGain Lib "PISO813.dll" (_

```
    ByVal wAddrBase As Long, ByVal wChannel As Integer, _
    ByVal wGainCode As Integer) As Integer
Declare Function PISO813_AD_Hex Lib "PISO813.dll" ( _
    ByVal wAddrBase As Long) As Integer
Declare Function PISO813_ADs_Hex Lib "PISO813.dll" ( _
    ByVal wAddrBase As Long, wBuffer As Integer, _
    ByVal dwDataNo As Long) As Integer
Declare Function PISO813_AD_Float Lib "PISO813.dll" ( _
    ByVal wAddrBase As Long, ByVal wJump10v As Integer, _
    ByVal wBipolar As Integer) As Single
Declare Function PISO813_ADs_Float Lib "PISO813.dll" ( _
    ByVal wAddrBase As Long, ByVal wJump10v As Integer, _
    ByVal wBipolar As Integer, fBuffer As Single, _
    ByVal dwDataNo As Long) As Single
Declare Function PISO813_AD2F Lib "PISO813.dll" (ByVal whex as integer, _
    ByVal wGainCode as integer, ByVal wJump20v as integer, _
    ByVal wBipolar as integer) as Single
```

2.3 PISO813.PAS

```

unit PISO813;      { PISO813.dll interface unit }

interface

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

const

//***** define PISO813 relative address *****/
PISO813_AD_LO      =$d0;  // Analog to Digital, Low Byte
PISO813_AD_HI      =$d4;  // Analog to Digital, High Byte
PISO813_SET_CH     =$E0;  // channel selecting
PISO813_SET_GAIN   =$E4;  // PGA gain code
PISO813_SOFT_TRIG  =$F0;  // A/D trigger control register

//***** define the gain mode *****/
PISO813_BI_1       =$00;
PISO813_BI_2       =$01;
PISO813_BI_4       =$02;
PISO813_BI_8       =$03;
PISO813_BI_16      =$04;

PISO813_UNI_1      =$00;
PISO813_UNI_2      =$01;
PISO813_UNI_4      =$02;
PISO813_UNI_8      =$03;
PISO813_UNI_16     =$04;

//***** return code *****/
PISO813_NoError    =0;
PISO813_DriverOpenError  =1;
PISO813_DriverNoOpen  =2;

```

```

PISO813_GetDriverVersionError =3;
PISO813_CallDriverError      =4;
PISO813_FindBoardError       =5;
PISO813_ExceedBoardNumber    =6;
PISO813_TimeOutError         =$ffff;
PISO813_AdError2             =-100.0;

```

```

// ID
PISO_813                      = $800A00;

```

// Test functions

```

function PISO813_ShortSub(nA : smallint; nB : smallint) : smallint; StdCall;
function PISO813_FloatSub(fA : single; fB : single)      : single; StdCall;
function PISO813_GetDllVersion                          : word; StdCall;

```

// Driver functions

```

function PISO813_DriverInit                               : word; StdCall;
procedure PISO813_DriverClose                             ; StdCall;
function PISO813_SearchCard(var wBoards:WORD;
    dwPIOCardID:LongInt):WORD; StdCall;
function PISO813_GetDriverVersion(var wDriverVer: word) : word; StdCall;
function PISO813_GetConfigAddressSpace(wBoardNo:word;
    var wAddrBase:LongInt; var wIrqNo:word; var wSubVerdor:word;
    var wSubDevice:word; var wSubAux:word; var wSlotBus:word;
    var wSlotDevice:word ): word; StdCall;

```

// DIO functions

```

procedure PISO813_OutputByte(wPortAddr : LongInt; bOutputVal : Word)
    ; StdCall;
procedure PISO813_OutputWord(wPortAddr : LongInt; wOutputVal : LongInt)
    ; StdCall;
function PISO813_InputByte(wPortAddr : LongInt )      : word; StdCall;
function PISO813_InputWord(wPortAddr : LongInt )     : LongInt; StdCall;

```

// AD functions

```

function PISO813_SetChGain(wAddrBase : LongInt; wChannel:Integer;
    wGainCode:Integer):Word; StdCall;
function PISO813_AD_Hex( wAddrBase : LongInt):Word; StdCall;
function PISO813_ADs_Hex( wAddrBase : LongInt; wBuffer:PWord;
    dwDataNo:LongInt):Word; StdCall;
function PISO813_AD_Float( wAddrBase : LongInt; wJump20v:Integer;
    wBipolar:Integer):Single; StdCall;
function PISO813_ADs_Float(wAddrBase : LongInt; wJump20v:Integer;
    wBipolar:Integer; fBuffer:PSingle; dwDataNo:LongInt):Single; StdCall;
function PISO813_AD2F(whex:Word; wGainCode:Word; wJump20v:Word;
    wBipolar:Word):Single; StdCall;

```

implementation

// Test functions

```

function PISO813_ShortSub;                external 'PISO813.DLL' name
    'PISO813_ShortSub';
function PISO813_FloatSub;                external 'PISO813.DLL' name
    'PISO813_FloatSub';
function PISO813_GetDllVersion;          external 'PISO813.DLL' name
    'PISO813_GetDllVersion';

```

// Driver functions

```

function PISO813_DriverInit;              external 'PISO813.DLL' name
    'PISO813_DriverInit';
procedure PISO813_DriverClose;            external 'PISO813.DLL' name
    'PISO813_DriverClose';
function PISO813_SearchCard;              external 'PISO813.DLL' name
    'PISO813_SearchCard';
function PISO813_GetDriverVersion;        external 'PISO813.DLL' name
    'PISO813_GetDriverVersion';
function PISO813_GetConfigAddressSpace;   external 'PISO813.DLL' name
    'PISO813_GetConfigAddressSpace';

```

```

//function PISO813_ActiveBoard;           external 'PISO813.DLL' name
    'PISO813_ActiveBoard';

```

```

//function PISO813_WhichBoardActive;     external 'PISO813.DLL' name
    'PISO813_WhichBoardActive';

```

```

// DIO functions
procedure PISO813_OutputByte;      external 'PISO813.DLL' name
    'PISO813_OutputByte';
procedure PISO813_OutputWord;     external 'PISO813.DLL' name
    'PISO813_OutputWord';
function PISO813_InputByte;       external 'PISO813.DLL' name
    'PISO813_InputByte';
function PISO813_InputWord;       external 'PISO813.DLL' name
    'PISO813_InputWord';

// AD functions
function PISO813_SetChGain;        external 'PISO813.DLL' name
    'PISO813_SetChGain';
function PISO813_AD_Hex;          external 'PISO813.DLL' name
    'PISO813_AD_Hex';
function PISO813_ADs_Hex;         external 'PISO813.DLL' name
    'PISO813_ADs_Hex';
function PISO813_AD_Float;        external 'PISO813.DLL' name
    'PISO813_AD_Float';
function PISO813_ADs_Float;       external 'PISO813.DLL' name
    'PISO813_ADs_Float';
function PISO813_AD2F;            external 'PISO813.DLL' name
    'PISO813_AD2F';

end.

```

3. Descriptions of Functions

In this chapter, the following keywords are used relating to function parameters:

Keyword	Parameter Set by user before calling this function?	Read data/value from this parameter after calling this function ?
[Input]	Yes	No
[Output]	No	Yes
[Input, Output]	Yes	Yes

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

3.1 TEST FUNCTIONS

3.1.1 PISO813_GetDIIVersion

- **Description:**
Gets the version number of PISO813.DLL
- **Syntax:**
WORD PISO813_GetDIIVersion(void)
- **Parameter:**
None
- **Return:**
200(hex) for version 2.00

3.1.2 PISO813_ShortSub

- **Description:**

Subtracts two numbers of short data type ($nA - nB$). This function is provided for testing the DLL linkage.

- **Syntax:**

short PISO813_ShortSub(short nA, short nB)

- **Parameter:**

nA : [Input] 2 bytes short data type value

nB : [Input] 2 bytes short data type value

- **Return:**

The value of $nA - nB$

3.1.3 PISO813_FloatSub

- **Description:**

Subtracts two numbers of float data type ($fA - fB$). This function is provided for testing the DLL linkage.

- **Syntax:**

float PISO813_FloatSub(float fA, float fB)

- **Parameter:**

fA : [Input] 4 bytes floating point value

fB : [Input] 4 bytes floating point value

- **Return:**

The value of $fA - fB$

3.2 I/O FUNCTIONS

3.2.1 PISO813_OutputByte

- **Description :**

This subroutine will write 8 bit data to the desired I/O port.

- **Syntax :**

```
void PISO813_OutputByte(DWORD wPortAddr, WORD bOutputVal);
```

- **Parameter :**

wPortAddr : [Input] I/O port address, please refer to function PISO813_GetConfigAddressSpace.

Only the low WORD is valid.

bOutputVal : [Input] 8 bit data to be written to the I/O port.

Only the low BYTE is valid.

- **Return:**

None

3.2.2 PISO813_InputByte

- **Description :**

This subroutine will read the 8 bit data from the desired I/O port.

- **Syntax :**

```
WORD PISO813_InputByte(DWORD wPortAddr);
```

- **Parameter :**

wPortAddr : [Input] I/O port address, please refer to function PISO813_GetConfigAddressSpace().

Only the low WORD is valid.

- **Return:**

16 bit data with the leading 8 bits all 0.

(Only the low BYTE is valid.)

3.2.3 PISO813_OutputWord

- **Description :**

This subroutine will send the 16 bit data to the desired I/O port.

- **Syntax :**

```
void PISO813_OutputWord(DWORD wPortAddr, DWORD wOutputVal);
```

- **Parameter :**

wPortAddr : [Input] I/O port address, please refer to function
PISO813_GetConfigAddressSpace().

Only the low WORD is valid.

wOutputVal : [Input] 16 bit data send to I/O port.

Only the low WORD is valid.

- **Return:**

None

3.2.4 PISO813_InputWord

- **Description :**

This subroutine will input 16 bit data from the desired I/O port.

- **Syntax :**

```
DWORD PISO813_InputWord(DWORD wPortAddr);
```

- **Parameter :**

wPortAddr : [Input] I/O port address, please refer to function
PISO813_GetConfigAddressSpace().

Only the low WORD is valid.

- **Return:**

16 bit data. Only the low WORD is valid.

3.3 DRIVER FUNCTIONS

3.3.1 PISO813_GetDriverVersion

- **Description :**

This subroutine will read the version number of OME-PISO-813 driver.

- **Syntax :**

WORD PISO813_GetDriverVersion(WORD *wDriverVersion);

- **Parameter :**

wDriverVersion : [Output] address of wDriverVersion

- **Return:**

PISO813_NoError : OK

PISO813_DriverNoOpen : The OME-PISO-813 driver is not open

PISO813_GetDriverVersionError : Read driver version error

3.3.2 PISO813_DriverInit

- **Description :**

This subroutine will open the OME-PISO-813 driver and allocate the resources for the device. This function must be called before calling other OME-PISO-813 functions.

- **Syntax :**

WORD PISO813_DriverInit();

- **Parameter :**

None

- **Return:**

PISO813_NoError : OK

PISO813_DriverOpenError : open OME-PISO-813 Driver error

3.3.3 PISO813_DriverClose

- **Description :**

This subroutine will close the OME-PISO-813 Driver and release the resources from the device. This function must be called before exiting the user's application.

- **Syntax :** void PISO813_DriverClose();

- **Parameter :** None

- **Return:** None

3.3.4 PISO813_GetConfigAddressSpace

- **Description :**

Get the I/O address of OME-PISO-813 board n.

- **Syntax :**

```
WORD PISO813_GetConfigAddressSpace  
    ( WORD wBoardNo,  DWORD *wAddrBase,  WORD *wlrqNo,  
      WORD *wSubVendor, WORD *wSubDevice,  WORD *wSubAux,  
      WORD *wSlotBus,  WORD *wSlotDevice);
```

- **Parameter :**

wBoardNo : [Input] OME-PISO-813 board number

wAddrBase : [Output] The base address of OME-PISO-813 board.
Only the low WORD is valid.

wlrqNo : [Output] The IRQ number that the OME-PISO-813 board
using.

wSubVendor : [Output] Sub Vendor ID.

wSubDevice : [Output] Sub Device ID.

wSubAux : [Output] Sub Aux ID.

wSlotBus : [Output] Slot Bus number.

wSlotDevice : [Output] Slot Device ID.

- **Return:**

PISO813_NoError : OK

PISO813_FindBoardError : handshake check error

PISO813_ExceedBoardError : wBoardNo is invalidated

3.4 A/D Functions

3.4.1 PISO813_SetChGain

- **Description:**

This subroutine sets the channel number and Gain-Code (Refer to Section 1.2) for the AD converter.

- **Syntax:**

```
WORD PISO813_SetChGain(DWORD wBase, WORD wChannel,
                       WORD wGainCode);
```

- **Parameter:**

wBase : [Input] I/O port base address, please refer the PISO813_GetConfigAddressSpace().

wChannel : [Input] A/D channel number, 0 to 31.

wGainCode : [Input] The value is 0 to 4, refer to Sec. 1.2.

- **Return:**

PISO813_NoError : OK

3.4.2 PISO813_AD2F

- **Description:**

This subroutine will convert the hex value to a floating point value depending on GainCode , Bipolar/Unipolar and 10v/20v.

- **Syntax:**

```
float PISO813_AD2F(WORD wHexValue, WORD wGainCode,
                  WORD wJump20v , WORD wBipolar);
```

- **Parameter:**

wHexValue : [Input] Hex Value 0 to 0x0FFF

wGainCode : [Input] The value is 0 to 4.
Refer to Sec. 1.2 for detailed information.

wJump20v : [Input] 1:20v(HW default) 0:10v

wBipolar : [Input] 1:Bipolar(HW default) 0:Unipolar

- **Return:**

PISO813_AdError2 : A/D converter error (return -100.0)

Other value : The **floating-point value** of the A/D conversion (-10 to 10)

3.4.3 PISO813_AD_Hex

- **Description:**

This subroutine will perform an A/D conversion by software polling. The A/D converter is 12 bits for OME-PISO-813. Refer to PISO813_SetChGain().

- **Syntax:**

WORD PISO813_AD_Hex(DWORD wBase);

- **Parameter:**

wBase : [Input] I/O port base address, please refer to PISO813_GetConfigAddressSpace().

- **Return:**

PISO813_TimeOutError : A/D converter error (return 0xffff)
 Other value : The **Hex value** of the A/D conversion (0 ~ 0x0fff)

3.4.4 PISO813_AD_Float

- **Description:**

This subroutine will perform an A/D conversion by software polling. The A/D converter is 12 bits for OME-PISO-813. This subroutine will compute the result according to the **configuration code** (Section 1.2). Refer to PISO813_SetChGain().

- **Syntax:**

float PISO813_AD_Float(DWORD wBase, WORD wJump20v, WORD wBipolar);

- **Parameter:**

wBase : [Input] I/O port base address, please refer to PISO813_GetConfigAddressSpace().

wJump20v : [Input] 1:20v(HW default) 0:10v

wBipolar : [Input] 1:Bipolar(HW default) 0:Unipolar

- **Return:**

PISO813_AdError2 : A/D converter error (return -100.0)

Other value : The **floating-point value** of the A/D conversion (-10 to 10)

3.4.5 PISO813_ADs_Hex

- **Description:**

This subroutine will perform a number of A/D conversions by software polling. This subroutine is very similar to PISO813_AD_Hex except that this subroutine will perform wCount of conversions instead of just one conversion. After the A/D conversions, the A/D data is stored in a buffer in Hex format. The **wBuf** is the starting address of this data buffer. Refer to PISO813_SetChGain().

- **Syntax:**

```
WORD PISO813_ADs_Hex(DWORD wBase, WORD wBuf[],
                    DWORD wCount);
```

- **Parameter:**

wBase	:	[Input] I/O port base address, please refer to PISO813_GetConfigAddressSpace().
wBuf	:	[Output] Starting address of the data buffer The user must allocate space for this buffer and send the address to the function. This function will fill the buffer with the data. The user can access the data in the buffer after calling this function.
wCount	:	[Input] Number of A/D conversions to be performed

- **Return:**

PISO813_TimeOutError	:	A/D converter error (0xffff)
PISO813_NoError	:	Operation is OK

3.4.6 PISO813_ADs_Float

- **Description:**

This subroutine will perform a number of A/D conversions by software polling. This subroutine is very similar to PISO813_AD_Float except that this subroutine will perform wCount of conversions instead of just one conversion. Then the A/D data is stored in a data buffer in Float format. The **fBuf** is the starting address of this data buffer. Refer to PISO813_SetChGain().

- **Syntax:**

```
WORD PISO813_ADs_Float(DWORD wBase, WORD wJump20v,
                      WORD wBipolar, float fBuf[], DWORD wCount);
```

- **Parameter:**

wBase : [Input] I/O port base address, refer to PISO813_GetConfigAddressSpace().

wJump20v : [Input] 1:20v(HW default) 0:10v

wBipolar : [Input] 1:Bipolar(HW default) 0:Unipolar

fBuf : [Output] Starting address of the data buffer
(in float format)

The user must allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. The user can analyze these data from the buffer after calling this function.

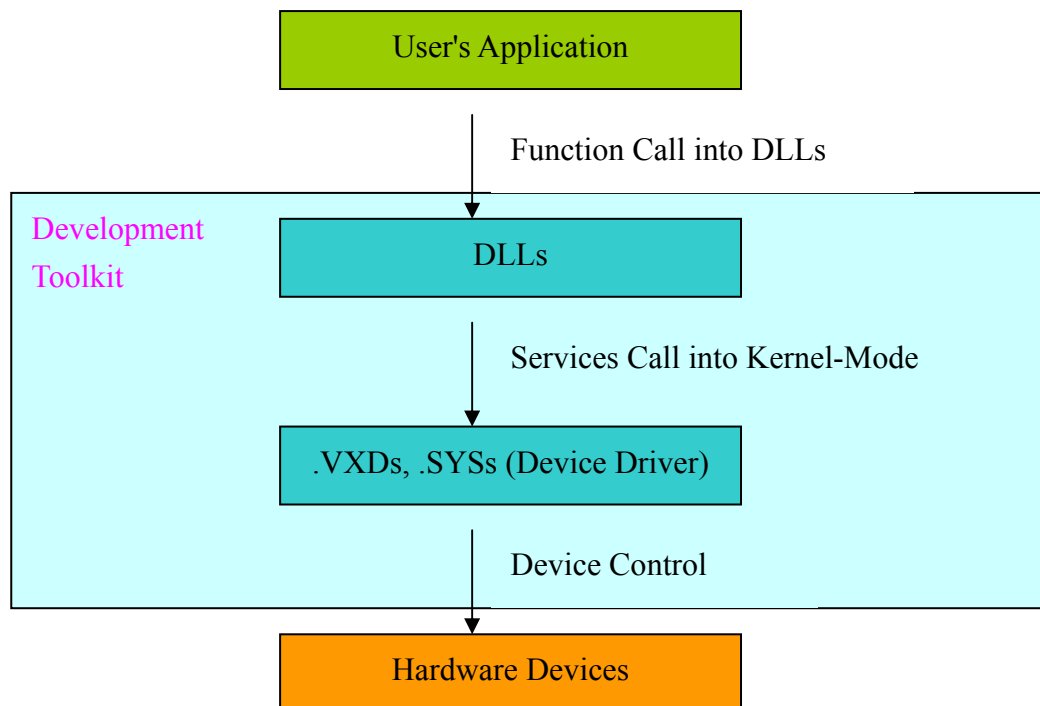
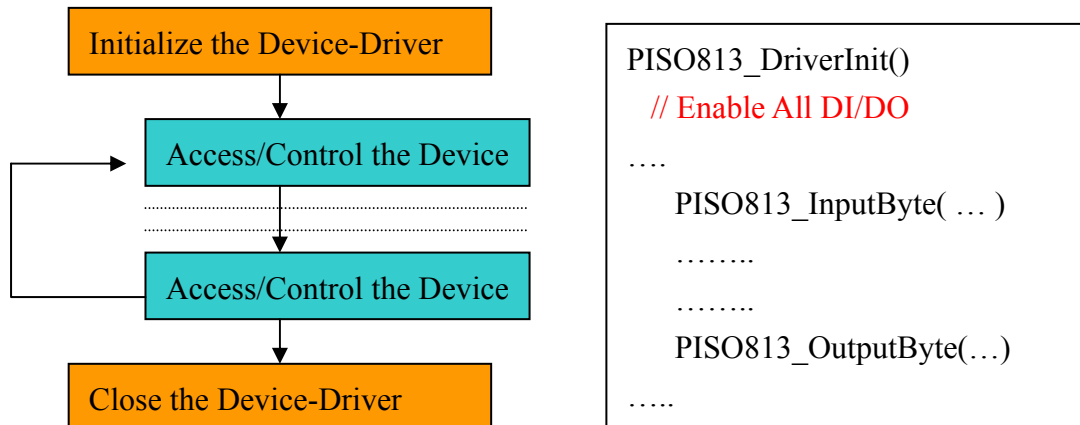
wCount : [Input] Number of A/D conversions will be performed

- **Return:**

PISO813_TimeOutError : A/D converter error (0xffff)

PISO813_NoError : Operation is OK

4. Program Architecture



5. Reporting Problem

Technical support is available at no charge as described below. The best way to report problems is send electronic mail to

das@omega.com

When reporting problems, please include the following information:

- 1) Is the problem reproducible? If so, how?
- 2) What **platform** are you using? For example, Windows 3.1, Windows for Workgroups, Windows NT 4.0, etc.
- 3) Which OMEGA **products** are you using?
- 4) If a dialog box with an **error message** was displayed, please include the full text of the dialog box, including the text in the title bar.
- 5) If the problem involves **other programs** or **hardware devices**, what devices or programs are you using?

E-mail: das@omega.com

Web Site: omega.com



WARRANTY/DISCLAIMER

OMEGA ENGINEERING, INC. warrants this unit to be free of defects in materials and workmanship for a period of 13 months from date of purchase. OMEGA's WARRANTY adds an additional one (1) month grace period to the normal one (1) year product warranty to cover handling and shipping time. This ensures that OMEGA's customers receive maximum coverage on each product.

If the unit malfunctions, it must be returned to the factory for evaluation. OMEGA's Customer Service Department will issue an Authorized Return (AR) number immediately upon phone or written request. Upon examination by OMEGA, if the unit is found to be defective, it will be repaired or replaced at no charge. OMEGA's WARRANTY does not apply to defects resulting from any action of the purchaser, including but not limited to mishandling, improper interfacing, operation outside of design limits, improper repair, or unauthorized modification. This WARRANTY is VOID if the unit shows evidence of having been tampered with or shows evidence of having been damaged as a result of excessive corrosion; or current, heat, moisture or vibration; improper specification; misapplication; misuse or other operating conditions outside of OMEGA's control. Components which wear are not warranted, including but not limited to contact points, fuses, and triacs.

OMEGA is pleased to offer suggestions on the use of its various products. However, OMEGA neither assumes responsibility for any omissions or errors nor assumes liability for any damages that result from the use of its products in accordance with information provided by OMEGA, either verbal or written. OMEGA warrants only that the parts manufactured by it will be as specified and free of defects. OMEGA MAKES NO OTHER WARRANTIES OR REPRESENTATIONS OF ANY KIND WHATSOEVER, EXPRESS OR IMPLIED, EXCEPT THAT OF TITLE, AND ALL IMPLIED WARRANTIES INCLUDING ANY WARRANTY OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE HEREBY DISCLAIMED. LIMITATION OF LIABILITY: The remedies of purchaser set forth herein are exclusive, and the total liability of OMEGA with respect to this order, whether based on contract, warranty, negligence, indemnification, strict liability or otherwise, shall not exceed the purchase price of the component upon which liability is based. In no event shall OMEGA be liable for consequential, incidental or special damages.

CONDITIONS: Equipment sold by OMEGA is not intended to be used, nor shall it be used: (1) as a "Basic Component" under 10 CFR 21 (NRC), used in or with any nuclear installation or activity; or (2) in medical applications or used on humans. Should any Product(s) be used in or with any nuclear installation or activity, medical application, used on humans, or misused in any way, OMEGA assumes no responsibility as set forth in our basic WARRANTY/DISCLAIMER language, and, additionally, purchaser will indemnify OMEGA and hold OMEGA harmless from any liability or damage whatsoever arising out of the use of the Product(s) in such a manner.

RETURN REQUESTS/INQUIRIES

Direct all warranty and repair requests/inquiries to the OMEGA Customer Service Department. BEFORE RETURNING ANY PRODUCT(S) TO OMEGA, PURCHASER MUST OBTAIN AN AUTHORIZED RETURN (AR) NUMBER FROM OMEGA'S CUSTOMER SERVICE DEPARTMENT (IN ORDER TO AVOID PROCESSING DELAYS). The assigned AR number should then be marked on the outside of the return package and on any correspondence.

The purchaser is responsible for shipping charges, freight, insurance and proper packaging to prevent breakage in transit.

FOR WARRANTY RETURNS, please have the following information available BEFORE contacting OMEGA:

1. Purchase Order number under which the product was PURCHASED,
2. Model and serial number of the product under warranty, and
3. Repair instructions and/or specific problems relative to the product.

FOR NON-WARRANTY REPAIRS, consult OMEGA for current repair charges. Have the following information available BEFORE contacting OMEGA:

1. Purchase Order number to cover the COST of the repair,
2. Model and serial number of the product, and
3. Repair instructions and/or specific problems relative to the product.

OMEGA's policy is to make running changes, not model changes, whenever an improvement is possible. This affords our customers the latest in technology and engineering.

OMEGA is a registered trademark of OMEGA ENGINEERING, INC.

© Copyright 2002 OMEGA ENGINEERING, INC. All rights reserved. This document may not be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of OMEGA ENGINEERING, INC.

Where Do I Find Everything I Need for Process Measurement and Control? OMEGA...Of Course!

Shop online at www.omega.com

TEMPERATURE

- Thermocouple, RTD & Thermistor Probes, Connectors, Panels & Assemblies
- Wire: Thermocouple, RTD & Thermistor
- Calibrators & Ice Point References
- Recorders, Controllers & Process Monitors
- Infrared Pyrometers

PRESSURE, STRAIN AND FORCE

- Transducers & Strain Gages
- Load Cells & Pressure Gages
- Displacement Transducers
- Instrumentation & Accessories

FLOW/LEVEL

- Rotameters, Gas Mass Flowmeters & Flow Computers
- Air Velocity Indicators
- Turbine/Paddlewheel Systems
- Totalizers & Batch Controllers

pH/CONDUCTIVITY

- pH Electrodes, Testers & Accessories
- Benchtop/Laboratory Meters
- Controllers, Calibrators, Simulators & Pumps
- Industrial pH & Conductivity Equipment

DATA ACQUISITION

- Data Acquisition & Engineering Software
- Communications-Based Acquisition Systems
- Plug-in Cards for Apple, IBM & Compatibles
- Datalogging Systems
- Recorders, Printers & Plotters

HEATERS

- Heating Cable
- Cartridge & Strip Heaters
- Immersion & Band Heaters
- Flexible Heaters
- Laboratory Heaters

ENVIRONMENTAL MONITORING AND CONTROL

- Metering & Control Instrumentation
- Refractometers
- Pumps & Tubing
- Air, Soil & Water Monitors
- Industrial Water & Wastewater Treatment
- pH, Conductivity & Dissolved Oxygen Instruments