ACLS-DLL2 ver. 5.0

Software Driver for
Windows 3.11, Win-95/98, Win-NT/2000

Function Reference Manual

Manual Rev. 5.00: 12, May 2000

The information in this document is subject to change without prior notice in order to improve reliability, design and function and does not represent a commitment on the part of the manufacturer.

In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

Trademarks

IBM PC is a registered trademark of International Business Machines Corporation. Intel is a registered trademark of Intel Corporation. Other product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

CONTENTS

How to Use This Guide

This manual is designed to help you use the ACLS-DLL2 software driver for NuDAQ multi-function cards ACL-6126, ACL-6128, ACL-8216, ACL-8316, ACL-8111, ACL-8113, ACL-8113A and ACL-8112 series. The manual describes how to install and use the library to meet your requirements and help you program your own software application. It is divided into four chapters:

- Chapter 1, "Using ACLS-DLL2 Functions” gives the important information about how to apply the function descriptions in this manual to your programming language and environment.

- Chapter 2, " Function Description” gives the detailed description of each function call ACL-DLL2 provided.

- Appendix A, “Status Code” lists the status codes returned by ACLS-DLL2 functions, as well as their meaning.

1

Using ACLS-DLL2 Functions

ACLS-DLL2 is the Microsoft Windows drivers for NuDAQ ISA-bus multi-function cards ACL-6126, ACL-6128, ACL-8111, ACL-8113, ACL-8113A, ACL-8112DG/HG, ACL-8112PG, ACL-8216 and ACL-8316/12. They are high performance data acquisition drivers for developing custom applications under Windows 3.1, Windows 95/98, Win-NT 4.0 and Win-2000. These drivers are DLLs (Dynamic-Link Library) for using under Windows. They can work with any Windows programming language that allows calls to a DLL, such as Microsoft C/C++, Microsoft Visual Basic.

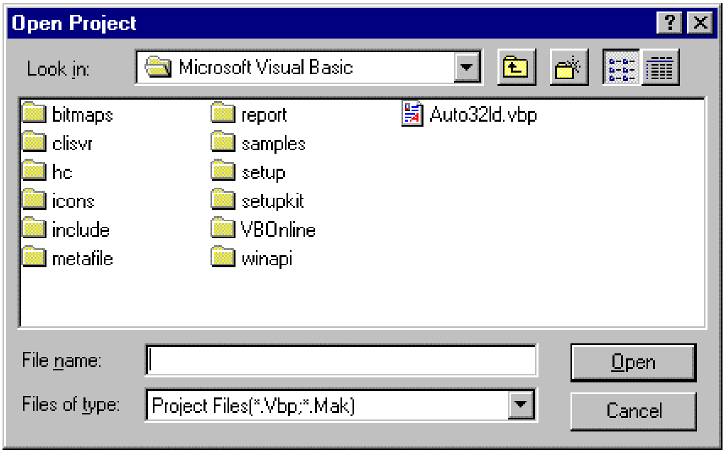
1.1 The fundamentals of Building Windows Application with ACLS-DLL2

1.1.1 Creating An Application Using Visual Basic and ACLS-DLL2

To create a data acquisition application using ACLS-DLL2 and Visual Basic, follow these steps after entering Visual Basic:

step 1. Open the project in which you want to use ACLS-DLL2. This can be a new or existing project

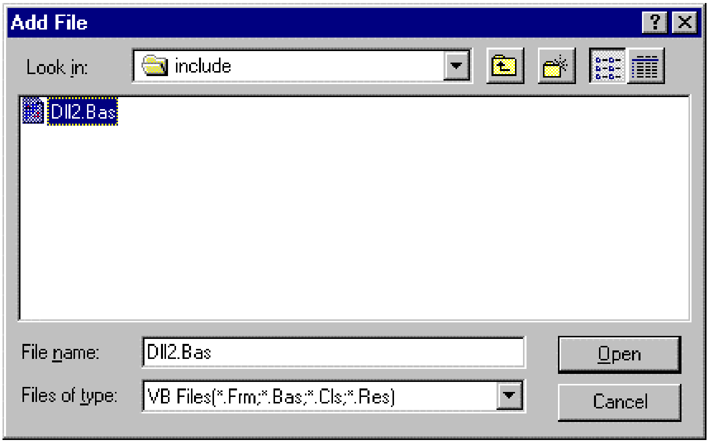
Open a new project by selecting the New Project command from the File menu. If it is an existing project, open it by selecting the Open Project command from the File menu. Then the Open Project dialog box appears.



Changed directory to the place the project file located. Double-click the project file name in the File Name list to load the project.

step 2. Add file DLL2.BAS into the project if this file is not included in the project. This file contains all the procedure declarations and constants that you can use to develop your data acquisition application.

From the File menu, select the Add File command. The Add File window appears, displaying a list of files in the current directory.



Look in: include

Dll2.Bas

File name: Dll2.Bas

Files of type: VB Files(*.Frm;*.Bas;*.Cls;*.Res)

Select DLL2.BAS from the Files list by double clicking on it. If you can't find this file in the list, make sure the list is displaying files from the correct directory. By default, DLL2.BAS is installed in C:\ACL-DLL2\INCLUDE.

step 3. Design the interface for the application.

To design the interface, you place the desired elements, such as command button, list box, text box, etc., on the Visual Basic form. These are standard controls from the Visual Basic Toolbox. To place a control on a form, you just move pointer to Toolbox, select the desired control and draw it on the form. Or you can double-click the control icon in the Toolbox to place it on the form.

step 4. Set properties for the controls.

To view the property list, click the desired control and then choose the Properties command from the View menu or press F4, or you can also click the Properties button on the toolbar.

step 5. Write the event code.

The event code defines the action you want to perform when an event occurs. To write the event code, double-click the desired control or form to view the code module and then add code you want. You can call the functions that declared in the file DLL2.BAS to perform data acquisition operations.

step 6. Run your application.

To run the application, choose Start from the Run menu, or click the Start icon ▶ on the toolbar (you can also press F5).

step 7. Distribute your application.

Once you have finished a project, you can save the application as an executable (.EXE) file by using the Make EXE File command on the File menu. And once you have saved your application as an executable file, you've ready to distribute it. When you distribute your application, remember also to include the ACLS-DLL2's DLL and driver files. These files should be copied to their appropriate directory as section 2.1.3 described.

1.1.2 Creating An Application Using Microsoft Visual C/C++ and ACLS-DLL2

To create a data acquisition application using ACLS-DLL2 and Microsoft Visual C/C++, follow these steps after entering Visual C/C++:

step 1. Open the project in which you want to use ACLS-DLL2. This can be a new or existing project

step 2. Include header file DLL2.H in the C/C++ source files that call ACLS-DLL2 functions. DLL2.H contains all the function declarations and constants that you can use to develop your data acquisition application. Incorporate the following statement in your code to include the header file.

```
#include  "DLL2.H"
```

step 3. Build your application.

Setting the appropriate compile and link options, then build your application by selecting the Build command from Build menu (Visual C/C++ 4.0) or Project menu (Visual C/C++ 1.52). Remember to link appropriate ACLS-DLL2's import libraries.

1.2 ACLS-DLL2 Functions Overview

Each NuDAQ multi-function card has its own DLL driver. How to use these DLL to build your own application has been described in section 1.1. The function calls in these DLLs use intuitive names that reflect the operations they perform. For example, W_8111_AD_Set_Channel sets the A/D reading channel.

The functionality of these function calls can be classified to the following capabilities,

1. Initialization : setup the hardware base I/O address
2. A/D conversion : performs analog to digital conversion
3. D/A conversion : performs digital to analog conversion
4. Digital I/O : input or output digital signals
5. Timer/Counter : Timer/Counter operation

1.3 Functions Naming Convention

The functions of ACL-DLL2 use full-names to represent the real meaning of the functions. The naming convention rules are:

W_{hardware_model}_{action_name}. e.g. **W_8111_Initial** ().

1.4 Data Types

We defined some data types in DLL2.H. These data types are used by ACLS-DLL2 library. We suggest you to use these data types in your application programs. The following table shows the data type names and their ranges.

Type Name	Description	Range
U8	8-bit ASCII character	0 to 255
I16	16-bit signed integer	-32768 to 32767
U16	16-bit unsigned integer	0 to 65535
I32	32-bit signed integer	-2147483648 to 2147483647
U32	32-bit unsigned integer	0 to 4294967295
F32	32-bit single-precision floating-point	-3.402823E38 to 3.402823E38
F64	64-bit double-precision floating-point	-1.797683134862315E308 to 1.797683134862315E309
Boolean	Boolean logic value	TRUE, FALSE

2

Function Reference

This chapter contains a detailed explanation of each ACLS-DLL1 function. The functions are arranged by Hardware products.

2.1 6126 Software DLL Driver

In this section, the ACL-6126's (ACL-726‘s) software DLL drivers are described. The function names of Windows 3.11, Window 95/98, Windows NT/2000 versions are the same. So, users do not need to learn the difference between them. The application’s portability between these three systems can be very high.

Note : All functions of the ACL-6126 can be applied to the ACL-726 directly. That is, users can use the 6126.DLL for both ACL-6126 and ACL-726 data acquisition cards.

2.1.1 W_6126_Initial

@ Description

An ACL-6126 card is initialized according to the card number, its corresponding base address, and IRQ level. If the ACL-6126 card will not perform interrupt operation, the argument irq is useless. Every NuDAQ ACL-6126 card has to be initialized by this function before calling other functions.

@ Syntax

Microsoft C/C++

int W_6126_Initial(int card_number, int base_address, int irq)

Visual Basic

Windows 3.11 Version:

W_6126_Initial (ByVal card_number As Integer, ByVal base_address As Integer, ByVal Irq As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_6126_Initial (ByVal card_number As Long, ByVal base_address As Long, ByVal Irq As Long) As Long

@ Argument

card_number : The card number to be initialized. If all the ACL-6126 cards only perform software polling, eight cards can be initialized and the valid card numbers are CARD_1, CARD_2, …, CARD_8. However, if the ACL-6126 cards are operated in *Windows NT* system and will perform *interrupt operation*, only two cards can be initialized and the card number must be CARD_1 or CARD_2.

base_address : the I/O port base address of the card.

Irq : the IRQ channel number used to transfer D/A data for this card. If the ACL-6126 card will not perform interrupt operation, this argument is useless.

Note: Since Windows NT arrange resources to devices at system startup time, under Windows NT environment, parameter irq is useless. You can not change IRQ level at run time. Please use *DLL2 Driver Registry Utility* to set IRQ level before running application. Please refer to section 1.6 "ACLS-DLL2 Device Driver Handling in Win-NT/2000".

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError

2.1.2 W_6126_Switch_Card_No

@ Description

This function is used on multi-cards system. After the ACL-6126 cards are initialized by W_6126_Initial function, you can use this function to select which one you want to operate.

@ Syntax

Microsoft C/C++

```
int W_6126_Switch_Card_No (int card_number)
```

Visual Basic

Windows 3.11 Version:

```
W_6126_Switch_Card_No (ByVal card_number As
        Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_6126_Switch_Card_No (ByVal card_number As Long)
        As Long
```

@ Argument

card_number : The card number of the card that is set to be active. If all the ACL-6126 cards only perform software polling, the valid card numbers are CARD_1, CARD_2, …, CARD_8. However, if the ACL-6126 cards are operated in *Windows NT* system and perform *interrupt operation*, the card number must be CARD_1 or CARD_2.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber

2.1.3 W_6126_DI

@ Description

This function is used to read data from digital input port. There are 16-bit digital inputs on the ACL-6126. The bit 0 to bit 7 are defined as **low byte** and the bit 8 to bit 15 are defined as **high byte**.

@ Syntax

Microsoft C/C++

```
int W_6126_DI (int port_number, unsigned char *di_data)
```

Visual Basic

Windows 3.11 Version:

```
W_6126_DI (ByVal port_number As Integer, di_data As
          Byte) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_6126_DI (ByVal port_number As Long, di_data As
          Byte) As Long
```

@ Argument

port_number : To indicate which port is read, DI_LOW_BYTE

or DI_HIGH_BYTE.
DI_LOW_BYTE : bit 0 ~ bit 7,
DI_HIGH_BYTE : bit8 ~ bit15

di_data : return value from digital port.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.1.4 W_6126_DI _Channel

@ Description

This function is used to read data from digital input channels (bit). There are 16 digital input channels on the ACL-6126. When performs this function, the digital input port is read and the value of the corresponding channel is returned.

* channel means each bit of digital input ports.

@ Syntax

Microsoft C/C++

```
int W_6126_DI_Channel (int di_ch_no, unsigned int
        *di_data)
```

Visual Basic

Windows 3.11 Version:

```
W_6126_DI_Channel (ByVal di_ch_no As Integer, di_data
        As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_6126_DI_Channel (ByVal di_ch_no As Long, di_data
        As Long) As Long
```

@ Argument

di_ch_no : the DI channel number, the value has to be set between 0 and 15.
di_data : return value, either 0 or 1.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDIChannel

2.1.5 W_6126_DO

@ Description

This function is used to write data to digital output ports. There are 16 digital outputs on the ACL-6126, they are divided to two ports, DO_LOW_BYTE and DO_HIGH_BYTE. The channel 0 to channel 7 are defined as DO_LOW_BYTE port and the channel 8 to channel 15 are defined as DO_HIGH_BYTE port.

@ Syntax

Microsoft C/C++

```
int W_6126_DO (int port_number, unsigned char do_data)
```

Visual Basic

Windows 3.11 Version:

```
W_6126_DO (ByVal port_number As Integer, ByVal
        do_data As Byte) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_6126_DO (ByVal port_number As Long, ByVal do_data
        As Byte) As Long
```

@ Argument

port_number : DO_LOW_BYTE or DO_HIGH_BYTE
do_data : the value written to digital output port

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.1.6 W_6126_DA

@ Description

This function is used to write data to D/A converters. There are six Digital-to-Analog conversion channels on the ACL-6126. The resolution of each channel is 12-bit, i.e. the range is from 0 to 4095.

@ Syntax

Microsoft C/C++

```
int W_6126_DA (int da_ch_no, unsigned int da_data)
```

Visual Basic

Windows 3.11 Version:

```
W_6126_DA (ByVal da_ch_no As Integer, ByVal da_data
          As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_6126_DA (ByVal da_ch_no As Long, ByVal da_data
          Long) As Long
```

@ Argument

da_ch_no : the DA channel number, the valid data is :

0	D/A CH1
1	D/A CH2
2	D/A CH3
3	D/A CH4
4	D/A CH5
5	D/A CH6

da_data : D/A converted value, if the value is greater than 4095, the higher 4-bits are negligent.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDAChannel

2.1.7 W_6126_INTOP_Start

@ Description

The function will perform D/A conversion N times with interrupt data transfer by using external trigger. It will take place in the background which will not be stopped until the Nth conversion has been completed or your program execute W_6126_INTOP_Stop() function to stop the process. After calling this function, it is necessary to check the status of the operation by using the function W_6126_INTOP_Status(). The function performs D/A conversion on the D/A channels that Set_INT_Op() specified.

@ Syntax

Microsoft C/C++

```
int W_6126_INTOP_Start (int count)
```

Visual Basic

Windows 3.11 Version:

```
W_6126_INT_Start (ByVal count As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_6126_INTOP_Start (ByVal count As Long) As Long
```

@ Argument

count :	the numbers of D/A conversion

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.1.8 W_6126_INTOP_Status

@ Description

Since the W_6126_INTOP_Start() function is executed in background, you can issue the function W_6126_INTOP_Status() to check the status of interrupt operation.

@ Syntax

Microsoft C/C++

int W_6126_INTOP_Status (int *status , int *count)

Visual Basic

Windows 3.11 Version:

W_6126_INT_Status (status As Integer, count As Integer)
As Integer

Win-95/98, Win-NT/2000 Version:

W_6126_INTOP_Status (status As Long, count As Long)
As Long

@ Argument

status :	status of the INT data transfer DA_INT_STOP : D/A INT is completed DA_INT_RUN : D/A INT is not completed
count :	current conversion count number.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ADINTNotSet

2.1.9 W_6126_INTOP_Stop

@ Description

This function is used to stop the interrupt data transfer function. The number of the data transferred is stored in *count*.

@ Syntax

Microsoft C/C++

int W_6126_INTOP_Stop (int *count)

Visual Basic

Windows 3.11 Version:

W_6126_INT_Stop (count As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_6126_INTOP_Stop (count As Long) As Long

@ Argument

count : the number of D/A data which has been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ADINTNotSet

2.1.10 W_6126_INT_Enable

@ Description

This function is only available in Window 95 driver, Windows NT and Windows 2000 driver. The function is used to initialize and start up the interrupt operation. After calling this function, every time an interrupt request signal is generated, a software event is signaled. So that in your program, your can use wait operation to wait for the event. When the event is signaled, it means an interrupt is generated. Please refer to the sample program *6126int.c*.

Note : The W_6126_INT_Enable and W_6126_INT_Disable is a pair of functions. That is, as the W_6126_INT_Enable is called, the W_6126_INT_Disable has to follow up behind it. Otherwise, the interrupt signal generation will not stop.

@ Syntax

Microsoft C/C++ (Win-95/98, Win-NT & Win-2000)

int W_6126_INT_Enable(HANDLE *hIntEvent)

Visual Basic (Win-95/98, Win-NT & Win-2000)

W_6126_INT_Enable (hIntEvent As Long) As Long

@ Argument

hIntEvent : the handle of the event for interrupt signals.

@ Return Code

ERR_NoError
ERR_INTNotSet

2.1.11 W_6126_INT_Disable

@ Description

This function is only available in Window 95 driver, Windows NT and Win-2000 driver. This function is used to stop the interrupt signal generation.

Note : This function has to be called after the W_6126_INT_Enable is called.

@ Syntax

Microsoft C/C++ (Win-95/98, Win-NT & Win-2000)

int W_6126_INT_Disable()

Visual Basic (Win-95/98, Win-NT & Win-2000)

W_6126_INT_Disable () As Long

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.1.12 Set_INT_Op

@ Description

This function is used to specify the D/A channel and data buffer that will be used for D/A conversion with interrupt data transfer. There are six D/A channels on ACL-6126. Each channel can be set for D/A interrupt data transfer. You can set as many channels as you need. For example, you may set D/A channels 0, 2, and 5 for D/A interrupt data transfer by calling Set_INT_Op() three times --- Set_INT_Op(0, buf1), Set_INT_Op(2, buf2), Set_INT_Op(5, buf3). (buf1, buf2, and buf3 are data buffer address) After setting the D/A channel and its buffer, you can call *W_6126_INTOP_Start()* to start D/A interrupt data transfer.

@ Syntax

Microsoft C/C++

```
int Set_INT_Op (int da_ch, unsigned int *da_buffer)
```

Visual Basic

Windows 3.11 Version:

```
Set_INT_Op (ByVal da_ch As Integer, da_buffer As
        Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
Set_INT_Op (ByVal da_ch As Long, da_buffer As Long)
        As Long
```

@ Argument

da_ch : the D/A channel number, the value has to be set between 0 and 5.

da_buffer : the start address of the memory buffer to store the D/A data, the buffer size must be large than the number of D/A conversion.

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the *da_buffer* argument. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDAChannel

@ Example

Microsoft C/C++

```
int da_buf0[1024],  da_buf1[1024], da_buf2[1024];
       .
       .
       .
Set_INT_Op(0, da_buf0);
Set_INT_Op(1, da_buf1);
Set_INT_Op(4, da_buf2);
W_6126_INTOP_Start(1000);  /* 1000 times of D/A interrupt
                              data transfer on channel 0, 1,
                              and 4 */
       .
       .
       .
```

2.1.13 Reset_INT_Op

@ Description

This function is used to reset the D/A channel and buffer settings of D/A interrupt data transfer. (set by calling Set_INT_Op())

@ Syntax

Microsoft C/C++

```
int Reset_INT_Op()
```

Visual Basic

Windows 3.11 Version:

```
Reset_INT_Op() As Integer
```

Win-95/98, Win-NT/2000 Version:

```
Reset_INT_Op() As Long
```

@ Return Code

ERR_NoError

2.2 6128 Software DLL Driver

In this section, the ACL-6128's (ACL-728‘s) software DLL drivers are described. The function names of Windows 3.11, Window 95/98, Windows NT/2000 versions are the same. So, users do not need to learn the difference between them. The application’s portability between these three systems can be very high.

Note : All functions of the ACL-6128 can be applied to the ACL-728 directly. That is, users can use the 6128.DLL for both ACL-6128 and ACL-728 DAS cards.

2.2.1 W_6128_Initial

@ Description

An ACL-6128 card is initialized according to the card number and its corresponding base address. Every ACL-6128 card has to be initialized by this function before calling other functions.

@ Syntax

Microsoft C/C++

```
int W_6128_Initial (int card_number, int base_address)
```

Visual Basic

Windows 3.11 Version:

```
W_6128_Initial (ByVal card_number As Integer, ByVal
        base_address As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_6128_Initial (ByVal card_number As Long, ByVal
        base_address As Long) As Long
```

@ Argument

card_number : The card number to be initialized, totally 8 cards can be initialized, the card number must be within the range of 0 and 7.

base_address : the I/O port base address of the card.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError

2.2.2 W_6128_Switch_Card_No

@ Description

This function is used on multi-cards system. After the ACL-6128 cards are initialized by W_6128_Initial() function, you can use this function to select which one you want to operate.

@ Syntax

Microsoft C/C++

```
int W_6128_Switch_Card_No (int card_number)
```

Visual Basic

Windows 3.11 Version:

```
W_6128_Switch_Card_No (ByVal card_number As
        Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_6128_Switch_Card_No (ByVal card_number As Long)
        As Long
```

@ Argument

card_number : The card number of the card that is set to be active. The valid value ranges within 0 and 7.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber

2.2.3 W_6128_DA

@ Description

This function is used to write data to D/A converters. There are two Digital-to-Analog conversion channels on the ACL-6128. The resolution of each channel is 12-bit, i.e. the range is from 0 to 4095.

@ Syntax

Microsoft C/C++

int W_6128_DA (int da_ch_no, unsigned int da_data)

Visual Basic

Windows 3.11 Version:

W_6128_DA (ByVal da_ch_no As Integer, ByVal da_data As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_6128_DA (ByVal da_ch_no As Long, ByVal da_data As Long) As Long

@ Argument

da_ch_no : the D/A channel number, , the valid data is

0	Channel CH1
1	Channel CH2

da_data : D/A converted value, if the value is greater than 4095, the higher 4-bits are negligent.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDAChannel

2.3 8111 Software DLL Driver

In this section, the ACL-8111's (ACL-711B‘s) software DLL drivers are described. The function names of Windows 3.11, Window 95/98, Windows NT/2000 versions are the same. So, users do not need to learn the difference between them. The application’s portability between these three systems can be very high.

Note : All functions of the ACL-8111 can be applied to the ACL-711B directly. That is, users can use the 8111.DLL for both ACL-8111 and ACL-711B data acquisition cards.

2.3.1 W_8111_Initial

@ Description

An ACL-8111 card is initialized according to the card number and its corresponding base address. Each ACL-8111 multi-function data acquisition card has to be initialized by this function before calling other functions.

@ Syntax

Microsoft C/C++

```
int W_8111_Initial (int card_number, int base_address)
```

Visual Basic

Windows 3.11 Version:

```
W_8111_Initial (ByVal card_number As Integer, ByVal
        base_address As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8111_Initial (ByVal card_number As Long, ByVal
        base_address As Long) As Long
```

@ Argument

card_number : The card number to be initialized. If all the ACL-8111 cards only perform software polling, eight cards can be initialized and the valid card numbers are CARD_1, CARD_2, …, CARD_8. However, if the ACL-8111 cards are operated in *Windows NT* system and will perform *interrupt operation*, only two cards can be initialized and the card number must be CARD_1 or CARD_2.

base_address : the I/O port base address of the card.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError

2.3.2 W_8111_Switch_Card_No

@ Description

After the ACL-8111 cards are initialized by above function, you can use this function to select which one you want to operate.

@ Syntax

Microsoft C/C++

```
int W_8111_Switch_Card_No (int card_number)
```

Visual Basic

Windows 3.11 Version:

```
W_8111_Switch_Card_No (ByVal card_number As
        Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8111_Switch_Card_No (ByVal card_number As Long)
        As Long
```

@ Argument

card_number : The card number of the card that is set to be active. If all the ACL-8111 cards only perform software polling, eight cards can be initialized and the valid card numbers are CARD_1, CARD_2, …, CARD_8. However, if the ACL-8111 cards are operated in *Windows NT* system and will perform *interrupt operation*, the card number must be CARD_1 or CARD_2.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber

2.3.3 W_8111_DI

@ Description

This function is used to read data from digital input port. There are 16-bit digital inputs on the ACL-8111. The bit 0 to bit 7 are defined as **low byte** and the bit 8 to bit 15 are defined as **high byte**.

@ Syntax

Microsoft C/C++

```
int W_8111_DI (int port_number, unsigned char *di_data)
```

Visual Basic

Windows 3.11 Version:

```
W_8111_DI (ByVal port_number As Integer, di_data As
        Byte) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8111_DI (ByVal port_number As Long, di_data As
        Byte) As Long
```

@ Argument

port_number : To indicate which port to read, DI_LOW_BYTE or DI_HIGH_BYTE.

DI_LOW_BYTE : bit 0 ~ bit 7,
DI_HIGH_BYTE : bit8 ~ bit15

di_data : return value from digital port.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.3.4 W_8111_DI _Channel

@ Description

This function is used to read data from digital input channels (bit). There are 16 digital input channels on the ACL-8111. When performs this function, the digital input port is read and the value of the corresponding channel is returned.

* channel means each bit of digital input ports.

@ Syntax

Microsoft C/C++

```
int W_8111_DI_Channel (int di_ch_no, unsigned int
          *di_data)
```

Visual Basic

Windows 3.11 Version:

```
W_8111_DI_Channel (ByVal di_ch_no As Integer, di_data
          As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8111_DI_Channel (ByVal di_ch_no As Long, di_data
          As Long) As Long
```

@ Argument

di_ch_no : the DI channel number, the value has to be set from 0 to 15.

di_data : return value, either 0 or 1.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDIChannel

2.3.5 W_8111_DO

@ Description

This function is used to write data to digital output ports. There are 16 digital outputs on the ACL-8111, they are divided to two ports, DO_LOW_BYTE and DO_HIGH_BYTE. The channel 0 to channel 7 are defined as DO_LOW_BYTE port and the channel 8 to channel 15 are defined as the DO_HIGH_BYTE port.

@ Syntax

Microsoft C/C++

```
int W_8111_DO (int port_number, unsigned char do_data)
```

Visual Basic

Windows 3.11 Version:

```
W_8111_DO (ByVal port_number As Integer, ByVal
        do_data As Byte) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8111_DO (ByVal port_number As Long, ByVal do_data
        As Byte) As Long
```

@ Argument

port_number :	DO_LOW_BYTE or DO_HIGH_BYTE
do_data :	value will be written to digital output port

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.3.6 W_8111_DA

@ Description

This function is used to write data to D/A converters. There is one Digital-to-Analog conversion channel on the ACL-8111. The resolution of the channel is 12-bit, i.e. the range is from 0 to 4095.

@ Syntax

Microsoft C/C++

```
int W_8111_DA (unsigned int da_data)
```

Visual Basic

Windows 3.11 Version:

```
W_8111_DA (ByVal da_data As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8111_DA (ByVal da_data As Long) As Long
```

@ Argument

da_data : D/A converted value, if the value is greater than 4095, the higher 4 bits are negligent.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.3.7 W_8111_AD_Set_Channel

@ Description

This function is used to set A/D channel by means of writing data to A/D channel multiplexer register. There are 8 single-ended A/D channels in ACL-8111, so the channel number should be set between 0 to 7 only. The initial state is channel 0 which is the default setting by the ACL-8111 hardware configuration.

@ Syntax

Microsoft C/C++

int W_8111_AD_Set_Channel (int ad_ch_no)

Visual Basic

Windows 3.11 Version:

W_8111_AD_Set_Channel (ByVal ad_ch_no As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8111_AD_Set_Channel (ByVal ad_ch_no As Long) As Long

@ Argument

ad_ch_no : channel number to perform A/D conversion

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADChannel

2.3.8 W_8111_AD_Set_Gain

@ Description

This function is used to set the A/D gain by means of writing data to the gain control register. The initial value of gain is '1' which is the default setting by the ACL-8111 hardware. The relationship between gain and input voltage ranges is specified by following table:

Input Range (V)	Gain	Gain Code
±5 V	X 1	AD_GAIN_1
±2.5 V	X 2	AD_GAIN_2
±1.25 V	X 4	AD_GAIN_4
±0.625 V	X 8	AD_GAIN_8
±0.3125V	X 16	AD_GAIN_16

@ Syntax

Microsoft C/C++

int W_8111_AD_Set_Gain (int ad_gain)

Visual Basic

Windows 3.11 Version:

W_8111_AD_Set_Gain (ByVal ad_gain As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8111_AD_Set_Gain (ByVal ad_gain As Long) As Long

@ Argument

ad_gain : the programmable gain of A/D conversion, the possible value is:
AD_GAIN_1, AD_GAIN_2, AD_GAIN_4, AD_GAIN_8, and AD_GAIN_16.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADGain

2.3.9 W_8111_AD_Set_Mode

@ Description

This function is used to set the A/D trigger and data transfer mode by means of writing data to the mode control register. The hardware initial state of the ACL-8111 is set as A8111_AD_MODE_0, i.e. software (internal) trigger with program polling data.

A/D Mode	Description
A8111_AD_MODE_0	Software Trigger, Polling Transfer
A8111_AD_MODE_1	Software Trigger, Interrupt Transfer
A8111_AD_MODE_2	External Trigger, Polling Transfer
A8111_AD_MODE_3	External Trigger, Interrupt Transfer

A8111_AD_MODE_4	Timer Trigger, Polling Transfer
A8111_AD_MODE_5	Timer Trigger, Interrupt Transfer

@ Syntax

Microsoft C/C++

int W_8111_AD_Set_Mode (int irq_no, int ad_mode)

Visual Basic

Windows 3.11 Version:

W_8111_AD_Set_Mode (ByVal irq_no As Integer, ByVal ad_mode As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8111_AD_Set_Mode (ByVal irq_no As Long, ByVal ad_mode As Long) As Long

@ Argument

irq_no : interrupt IRQ level

ad_mode : A/D trigger and data transfer mode. The possible values are: A8111_AD_MODE_0, A8111_AD_MODE_1, A8111_AD_MODE_2, A8111_AD_MODE_3, A8111_AD_MODE_4, A8111_AD_MODE_5

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidMode

2.3.10 W_8111_AD_Soft_Trig

@ Description

This function is used to trigger the A/D conversion by software. When the function is called, a trigger pulse will be generated and the converted data will be stored in the address Base+4 and Base+5, and can be retrieved by function W_8111_AD_Aquire().

@ Syntax

Microsoft C/C++

int W_8111_AD_Soft_Trig (void)

Visual Basic

Windows 3.11 Version:

W_8111_AD_Soft_Trig( ) As Integer

Win-95/98, Win-NT/2000 Version:

W_8111_AD_Soft_Trig( ) As Long

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.3.11 W_8111_AD_Aquire

@ Description

This function is used to poll the A/D conversion data. It will trigger the A/D conversion, and read the 12-bit A/D data until the data is ready ('data ready' bit becomes low).

@ Syntax

Microsoft C/C++

int W_8111_AD_Aquire (int *ad_data)

Visual Basic

Windows 3.11 Version:

W_8111_AD_Aquire (ad_data As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8111_AD_Aquire (ad_data As Long) As Long

@ Argument

ad_data :	12 bits A/D converted value, the value should be within 0 and 4095.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_AD_AquireTimeOut

2.3.12 W_8111_CLR_IRQ

@ Description

This function is used to clear interrupt request which is requested by the ACL-8111. If you use interrupt to transfer A/D converted data, you should use this function to clear interrupt request status; otherwise new interrupt signal could not be generated.

@ Syntax

Microsoft C/C++

```
int W_8111_CLR_IRQ (void)
```

Visual Basic

Windows 3.11 Version:

```
W_8111_CLR_IRQ( ) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8111_CLR_IRQ( ) As Long
```

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.3.13 W_8111_AD_INT_Start

@ Description

The function will perform A/D conversion N times with interrupt data transfer by using timer pacer (internal clock trigger). It will take place in the background which will not be stopped until the Nth conversion has been completed or your program execute W_8111_AD_INT_Stop() function to stop the process. After calling this function, it is necessary to check the status of the operation by using the function W_8111_AD_INT_Status(). The function performs on single A/D channel with fixed gain.

@ Syntax

Microsoft C/C++

```
int W_8111_INT_Start (int ad_ch_no, int ad_gain,
        int irq_ch_no, int count, unsigned short *ad_buffer,
        unsigned int c1, unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_8111_AD_INT_Start (ByVal ad_ch_no As Integer,
        ByVal ad_gain As Integer, ByVal irq_ch_no As
        Integer, ByVal count As Integer, ad_buffer As
        Integer, ByVal c1 As Integer, ByVal c2 As Integer)
        As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8111_AD_INT_Start (ByVal ad_ch_no As Long, ByVal
        ad_gain As Long, ByVal irq_ch_no As Long, ByVal
        count As Long, ad_buffer As Integer, ByVal c1 As
        Long, ByVal c2 As Long) As Long
```

@ Argument

ad_ch_no :	A/D channel number
ad_gain :	A/D gain value, the possible values are: AD_GAIN_1, AD_GAIN_2, AD_GAIN_4, AD_GAIN_8, and AD_GAIN_16.
irq_ch_no :	IRQ channel number used to transfer A/D data, the possible value is defined in file DLL2.H

count : the number of A/D conversion

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must be large than the number of A/D conversion.

c1 : the 16-bit timer frequency divider of timer channel #1

c2 : the 16-bit timer frequency divider of timer channel #2

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidIRQChannel,
ERR_InvalidTimerValue

@ Example

Visual Basic

```
Dim ad_buf(1024) As Integer
Dim Channel As Integer, Gain As Integer, Irq As Integer
Dim ad_count As Integer, c1 As Integer, c2 As Integer
Dim Ret As Integer
      .  .  .
Ret = W_8111_AD_INT_Start(Channel, Gain, Irq, ad_count,
        ad_buf(0), c1, c2)
      .  .  .
```

2.3.14 W_8111_AD_INT_Status

@ Description

Since the W_8111_AD_INT_Start() function is executed in background, you can issue the function W_8111_AD_INT_Status() to check the status of interrupt transfer operation.

@ Syntax

Microsoft C/C++

int W_8111_AD_INT_Status (int *status , int *count)

Visual Basic

Windows 3.11 Version:

W_8111_AD_INT_Status (status As Integer, count As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8111_AD_INT_Status (status As Long, count As Long) As Long

@ Argument

status :	status of the interrupt data transfer AD_INT_STOP : A/D INT is completed AD_INT_RUN : A/D INT is not completed
count :	current conversion count number.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ADINTNotSet

2.3.15 W_8111_AD_INT_Stop

@ Description

This function is used to stop the interrupt data transfer operation. After calling this function, the internal A/D trigger is disabled and the A/D timer is stopped. The number of the data transferred is stored in *count*, no matter whether the AD interrupt data transfer is stopped by this function or by W_8111_AD_INT_Start() itself.

@ Syntax

Microsoft C/C++

```
int W_8111_AD_INT_Stop (int *count)
```

Visual Basic

Windows 3.11 Version:

```
W_8111_AD_INT_Stop (count As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8111_AD_INT_Stop (count As Long) As Long
```

@ Argument

count :	the number of A/D data which has been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ADINTNotSet

2.3.16W_8111_AD_ContINT_Start

@ Description

The function will perform continuous A/D conversions with interrupt data transfer by using timer pacer (internal clock trigger). It will take place in the background which will not be stopped until your program execute W_8111_AD_INT_Stop() function to stop the process. After calling this function, it is necessary to check the status of the operation by using the function W_8111_AD_DblBufferHalfReady().

@ Syntax

Microsoft C/C++

```
int W_8111_ContINT_Start (int ad_ch_no, Boolean
          autoscan, int ad_gain, int irq_ch_no, int count,
unsigned short *ad_buffer, unsigned int c1,
unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_8111_AD_ContINT_Start (ByVal ad_ch_no As Integer,
        ByVal auto_scan As Integer, ByVal ad_gain As
        Integer, ByVal irq_ch_no As Integer, ByVal count
        As Integer, ad_buffer As Integer, ByVal c1 As
        Integer, ByVal c2 As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8111_AD_ContINT_Start (ByVal ad_ch_no As Long,
        ByVal auto_scan As Integer, ByVal ad_gain As
        Long, ByVal irq_ch_no As Long, ByVal count As
        Long, ad_buffer As Integer, ByVal c1 As Long,
        ByVal c2 As Long) As Long
```

@ Argument

ad_ch_no : A/D channel number

If autoscan is enabled, the A/D channel scan sequence will be: 0, 1, 2, 3,…[ad_ch_no], 0, 1, …, [ad_ch_no], …
If autoscan is disabled, only the data from channel [ad_ch_no] will be converted.

autoscan: FALSE: autoscan is disabled
TRUE: autoscan is enabled

ad_gain : A/D gain value, the possible values are: AD_GAIN_1, AD_GAIN_2, AD_GAIN_4, AD_GAIN_8, and AD_GAIN_16.

irq_ch_no : IRQ channel number used to transfer A/D data, the possible value is defined in file DLL2.H

count : the number of A/D conversion

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must be large than the number of A/D conversion.

c1 : the 16-bit timer frequency divider of timer channel #1

c2 : the 16-bit timer frequency divider of timer channel #2

@ Return Code

ERR_NoError
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidIRQChannel,
ERR_InvalidTimerValue
ERR_AD_INTNotSet

2.3.17W_8111_AD_SCANINT_Start

@ Description

This function is used to start automatic channel scan . If autoscan mode is started and the end channel number is set as n by argument **ad_ch_no**, the data will be converted automatically from channel 0 to channel n.
For example, the channel is set as 4 and autoscan is started, the A/D conversion sequence will be 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, ... If the autoscan is finished, the converted channel will be kept at the specified channel, i.e. channel 4.

@ Syntax

Microsoft C/C++

```
int W_8111_AD_SCANINT_Start( int ad_ch_no, int
    ad_gain , int irq_no, int count , unsigned short
    *ad_buffer , unsigned int c1 , unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_8111_AD_ SCANINT_Start (ByVal ad_ch_no As
        Integer, ByVal ad_gain As Integer, ByVal irq_ch_no
        As Integer, ByVal count As Integer, ad_buffer As
        Integer, ByVal c1 As Integer, ByVal c2 As Integer)
        As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8111_AD_ SCANINT_Start (ByVal ad_ch_no As Long,
        ByVal ad_gain As Long, ByVal irq_ch_no As Long,
        ByVal count As Long, ad_buffer As Integer, ByVal
        c1 As Long, ByVal c2 As Long) As Long
```

@ Argument

ad_ch_no : end A/D channel number for AutoScan

ad_gain : A/D gain value, the possible values are: AD_GAIN_1, AD_GAIN_2, AD_GAIN_4, AD_GAIN_8, and AD_GAIN_16.

irq_ch_no : IRQ channel number used to transfer A/D data, the possible value is defined in file DLL2.H

count : the number of A/D conversion

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must be large than the number of A/D conversion.

c1 : the 16-bit timer frequency divider of timer channel #1

c2 : the 16-bit timer frequency divider of timer channel #2

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADChannel
ERR_AD_InvalidGain
ERR_InvalidIRQChannel
ERR_InvalidTimerValue

2.3.18W_8111_AD_DblBufferHalfReady

@ Description

Checks whether the next half buffer of data in circular buffer is ready for transfer during double-buffered analog input operation.

@ Syntax

Microsoft C/C++

int W_8111_AD_DblBufferHalfReady (BOOLEAN *bHalfReady)

Visual Basic

W_8111_AD_DblBufferHalfReady (bHalfReady As Long) As Long

@ Argument

bHalfReady : Whether the next half buffer of data is available.If *HalfReady* = TRUE, you can call **`W_8111_AD_DblBufferTransfer()`** to copy the data to your user buffer.

@ Return Code

ERR_NoError
ERR_InvalidMode

2.3.19W_8111_AD_DblBufferTransfer

@ Description

Depending on the continuous AI function selected, half of the data in circular buffer will be logged into the user buffer.You can execute this function repeatedly to return sequential half buffers of the data.

@ Syntax

Microsoft C/C++

int W_8111_AD_DblBufferTransfer (USHORT *pwBuffer)

Visual Basic

W _8111_AD_DblBufferTransfer (pwBuffer As Integer) As Long

@ Argument

pwBuffer: The user buffer. An integer array to which the data is to be copied.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.3.20 W_8111_AD_Timer

@ Description

This function is used to set up Timer #1 and Timer #2. The c1 and c2 arguments are used as frequency dividers for generating constant A/D sampling rate dedicatedly. It is possible to stop the pacer trigger by setting any one of the dividers as 0. Because the A/D conversion rate is limited due to the conversion time of the A/D converter, the highest sampling rate of ACL-8111 can not exceed 30 KHz. The multiplication of the dividers must be larger than 70.

@ Syntax

Microsoft C/C++

```
int W_8111_AD_Timer (unsigned int c1, unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_8111_AD_Timer (ByVal c1 As Integer, ByVal c2 As
        Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8111_AD_Timer (ByVal c1 As Long, ByVal c2 As Long)
        As Long
```

@ Argument

c1 :	frequency divider of timer #1
c2 :	frequency divider of timer #2,

Note : the A/D sampling rate is equal to : 2MHz / (c1*c2), when c1 = 0 or c2 = 0, the pacer trigger will be stopped.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerValue

2.4 8112 Software DLL Driver

In this section, the ACL-8112 Series' software DLL drivers are described. This DLL library can support both ACL-8112DG and ACL-8112HG. The function names of Windows 3.11, Window 95/98, Windows NT/2000 versions are the same. So, users do not need to learn the difference between them. The application's portability between these three systems can be very high.

2.4.1 W_8112_Initial

@ Description

An ACL-8112DG/HG card is initialized according to the card number and the corresponding base address. Each ACL-8112 multi-function data acquisition card has to be initialized by this function before calling other functions.

Note: In this library, if you want to operate DMA or interrupt operation, only two ACL-8112DG/HG/PG cards can be initialized. The reason is only two DMA channels are supported in the card.

@ Syntax

Microsoft C/C++

```
int W_8112_Initial (int card_number, int base_addresss)
```

Visual Basic

Windows 3.11 Version:

```
W_8112_Initial (ByVal card_number As Integer, ByVal
        base_address As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8112_Initial (ByVal card_number As Long, ByVal
        base_address As Long) As Long
```

@ Argument

card_number : The card number to be initialized. If all the ACL-8112DG/HG cards only perform software polling, eight cards can be initialized and the valid card numbers are CARD_1, CARD_2, …, CARD_8. However, if the ACL-8111DG/HG cards are operated in *Windows NT* system and will perform *interrupt or DMA data transfer*, only two cards can be initialized and the card number must be CARD_1 or CARD_2.

base_address : the I/O port base address of the card.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError

2.4.2 W_8112_Switch_Card_No

@ Description

After initialized more than one ACL-8112 cards, this function is used to select which card is used currently.

@ Syntax

Microsoft C/C++

```
int W_8112_Switch_Card_No (int card_number)
```

Visual Basic

Windows 3.11 Version:

```
W_8112_Switch_Card_No (ByVal card_number As
        Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8112_Switch_Card_No (ByVal card_number As Long)
        As Long
```

@ Argument

card_number : The card number of the card that is set to be active. If all the ACL-8112DG/HG cards only perform software polling, the valid card numbers are CARD_1, CARD_2, …, CARD_8. However, if the ACL-8112DG/HG cards are operated in *Windows NT* system and will perform *interrupt or DMA data transfer*, only two cards can be initialized and the card number must be CARD_1 or CARD_2.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber

2.4.3 W_8112_DI

@ Description

This function is used to read data from digital input port. There are 16 digital inputs on the ACL-8112DG/HG. The bit 0 to bit 7 are defined as **low byte** and the bit 8 to bit 15 are defined as **high byte**.

@ Syntax

Microsoft C/C++

```
int W_8112_DI (int port_number, unsigned char *di_data)
```

Visual Basic

Windows 3.11 Version:

```
W_8112_DI (ByVal port_number As Integer, di_data As
        Byte) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8112_DI (ByVal port_number As Integer, di_data As
        Byte) As Long
```

@ Argument

port_number : To indicate which port is read, DI_LOW_BYTE or DI_HIGH_BYTE.

DI_LOW_BYTE : bit 0 ~ bit 7
DI_HIGH_BYTE : bit8 ~ bit15

di_data : return value from digital port.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.4.4 W_8112_DI _Channel

@ Description

This function is used to read data from digital input channels (bit). There are 16 digital input channels on the ACL-8112DG/HG. When performs this function, the digital input port is read and the value of the corresponding channel is returned.

* channel means each bit of digital input ports.

@ Syntax

Microsoft C/C++

```
int _8112_DI_Channel (int di_ch_no, unsigned int
        *di_data)
```

Visual Basic

Windows 3.11 Version:

```
W_8112_DI_Channel (ByVal di_ch_no As Integer, di_data
        As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8112_DI_Channel (ByVal di_ch_no As Long, di_data
        As Long) As Long
```

@ Argument

di_ch_no : the DI channel number, the value has to be set between 0 and 15.

di_data : return value, either 0 or 1.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDIChannel

2.4.5 W_8112_DO

@ Description

This function is used to write data to digital output ports. There are 16 digital outputs on the ACL-8112DG/HG, they are divided to two ports, DO_LOW_BYTE and DO_HIGH_BYTE. The channel 0 to channel 7 are defined in DO_LOW_BYTE port and the channel 8 to channel 15 are defined as the DO_HIGH_BYTE port.

@ Syntax

Microsoft C/C++

```
int W_8112_DO (int port_number, unsigned char do_data)
```

Visual Basic

Windows 3.11 Version:

```
W_8112_DO (ByVal port_number As Integer, ByVal
        do_data As Byte) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8112_DO (ByVal port_number As Long, ByVal do_data
        As Byte) As Long
```

@ Argument

port_number :	DO_LOW_BYTE or DO_HIGH_BYTE
do_data :	value will be written to digital output port

@ Return Code

ERR_NoError
ERR_BoardNoInit

ERR_PortError

2.4.6 W_8112_DA

@ Description

This function is used to write data to D/A converters. There are two Digital-to-Analog conversion channels on the ACL-8112DG/HG. The resolution of each channel is 12-bit, i.e. the range is from 0 to 4095.

@ Syntax

Microsoft C/C++

int W_8112_DA (int da_ch_no, unsigned int da_data)

Visual Basic

Windows 3.11 Version:

W_8112_DA (ByVal da_ch_no As Integer, ByVal da_data As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8112_DA (ByVal da_ch_no As Long, ByVal da_data As Long) As Long

@ Argument

da_ch_no : D/A channel number, the valid data is :

0	Channel AO1
1	Channel AO2

da_data : D/A converted value, if the value is greater than 4095, the higher 4 bits are negligent.

@ Return Code

ERR_NoError
ERR_BoardNoInit

ERR_InvalidDAChannel

2.4.7 W_8112_AD_Input_Mode

@ Description

This function is used to set A/D input mode to single-ended or differential mode. The default mode of A/D input is single-ended, so the A/D channel number can be set between 0 to 15. If the A/D mode is set as differential, the input channel can be selected from channel 0 to 7 only.

@ Syntax

Microsoft C/C++

```
int W_8112_AD_Input_Mode (int mode)
```

Visual Basic

Windows 3.11 Version:

```
W_8112_AD_Input_Mode (ByVal mode As Integer) As
          Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8112_AD_Input_Mode (ByVal mode As Long) As Long
```

@ Argument

mode :	SIGNLE_ENDED : singled-ended mode is set DIFFERENTIAL : differential mode is set

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADMode

2.4.8 W_8112_AD_Set_Channel

@ Description

This function is used to set A/D channel by means of writing data to the A/D channel multiplexer register. There are 16 single-ended A/D channels in ACL-8112, so the channel number should be set between 0 and 15 only. The initial state is channel 0 which is the default setting by the ACL-8112 hardware configuration.

@ Syntax

Microsoft C/C++

int W_8112_AD_Set_Channel (int ad_ch_no)

Visual Basic

Windows 3.11 Version:

W_8112_AD_Set_Channel (ByVal ad_ch_no As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8112_AD_Set_Channel (ByVal ad_ch_no As Long) As Long

@ Argument

ad_ch_no : channel number to perform A/D conversion

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADChannel

2.4.9 W_8112_AD_Set_Range

@ Description

This function is used to set the A/D range by means of writing data to the range control register. The major difference between 8112DG, 8112HG, and 8112PG is each card supports different gains which affect the input voltage range of each card. This is the only difference between these cards. Each card's gain and its corresponding A/D input ranges are listed as below.

The initial value of gain is '1', which is set by the ACL-8112 hardware.

**** ACL-8112DG**

Input Range (V)	Gain	Gain Code
±10 V	X 0.5	AD_B_10_V
±5 V	X 1	AD_B_5_V
±2.5 V	X 2	AD_B_2_5_V
±1.25 V	X 4	AD_B_1_25_V
±0.625 V	X 8	AD_B_0_625_V
0V ~ 10 V	X 1	AD_U_10_V
0V ~ 5 V	X 2	AD_U_5_V
0V ~ 2.5 V	X 4	AD_U_2_5_V
0V ~ 1.25 V	X 8	AD_U_1_25_V

**** ACL-8112HG**

Input Range (V)	Gain	Gain Code
±5 V	X 1	AD_B_5_V
±0.5 V	X 10	AD_B_0_5_V
±0.05 V	X 100	AD_B_0_05_V
±0.005 V	X 1000	AD_B_0_005_V
0V ~ 10 V	X 1	AD_U_10_V
0V ~ 1 V	X 10	AD_U_1_V
0V ~ 0.1 V	X 100	AD_U_0_1_V
0V ~ 0.01 V	X 1000	AD_U_0_01_V
±10V	X 0.5	AD_B_10_V
±1V	X 5	AD_B_1_V
±0.1V	X 50	AD_B_0_1_V
±0.01V	X 500	AD_B_0_01_V

**** ACL-8112PG :**

If input voltage range is set to ±5 V (JP9),

Input Range (V)	Gain	Gain Code
±5 V	X 1	AD_GAIN_1
±2.5 V	X 2	AD_GAIN_2
±1.25 V	X 4	AD_GAIN_4
±0.625 V	X 8	AD_GAIN_8
±0.3125V	X 16	AD_GAIN_16

If input voltage range is set to ±10 V (JP9),

Input Range (V)	Gain	Gain Code
±10 V	X 1	AD_GAIN_1
±5 V	X 2	AD_GAIN_2
±2.5 V	X 4	AD_GAIN_4
±1.25 V	X 8	AD_GAIN_8
±0.625 V	X 16	AD_GAIN_16

Note : This function will not check if you set a right gain code for different data acquisition cards, so you should be very careful what kind of data acquisition card you use, and set a right Gain code.

@ Syntax

Microsoft C/C++

int W_8112_AD_Set_Range (int ad_range)

Visual Basic

Windows 3.11 Version:

W_8112_AD_Set_Range (ByVal ad_range As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8112_AD_Set_Range (ByVal ad_range As Long) As Long

@ Argument

ad_range : the programmable gain of A/D conversion, the possible values are:

* ACL-8112DG :
AD_B_10_V, AD_B_5_V, AD_B_2_5_V, AD_B_1_25_V,
AD_B_0_625_V, AD_U_10_V, AD_U_5_V, AD_U_2_5_V,
AD_U_1_25_V

* ACL-8112HG :
AD_B_5_V, AD_B_0_5_V, AD_B_0_05_V, AD_B_0_005_V,
AD_U_10_V, AD_U_1_V, AD_U_0_1_V, AD_U_0_01_V,
AD_B_10_V, AD_B_1_V, AD_B_0_1_V, AD_B_0_01_V

* ACL-8112PG :
AD_GAIN_1, AD_GAIN_2, AD_GAIN_4, AD_GAIN_8, AD_GAIN_16

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADGain

2.4.10 W_8112_AD_Set_Mode

@ Description

This function is used to set the A/D trigger and data transfer mode by means of writing data to the mode control register. The hardware initial state of the ACL-8112DG/HG is set as A8112_AD_MODE_1 software (internal) trigger with program polling.

A/D Mode	Description
A8112_AD_MODE_0	External Trigger, Software Polling
A8112_AD_MODE_1	Software Trigger, Software Polling
A8112_AD_MODE_2	Timer Trigger, DMA Transfer
A8112_AD_MODE_3	External Trigger, DMA Transfer
A8112_AD_MODE_4	External Trigger, Interrupt Transfer
A8112_AD_MODE_5	Software Trigger, Interrupt Transfer
A8112_AD_MODE_6	Timer Trigger, Interrupt Transfer
A8112_AD_MODE_7	Not Used

@ Syntax

Microsoft C/C++

int W_8112_AD_Set_Mode (int ad_mode)

Visual Basic

Windows 3.11 Version:

W_8112_AD_Set_Mode (ByVal ad_mode As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8112_AD_Set_Mode (ByVal ad_mode As Long) As Long

@ Argument

ad_mode : A/D trigger and data transfer mode

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidMode

2.4.11 W_8112_AD_Soft_Trig

@ Description

This function is used to trigger the A/D conversion by software. When the function is called, a trigger pulse will be generated and the converted data will be stored in the address Base+4 and Base+5, and can be retrieved by function W_8112_AD_Aquire().

@ Syntax

Microsoft C/C++

int W_8112_AD_Soft_Trig (void)

Visual Basic

Windows 3.11 Version:

W_8112_AD_Soft_Trig ( ) As Integer

Win-95/98, Win-NT/2000 Version:

W_8112_AD_Soft_Trig ( ) As Long

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.4.12 W_8112_AD_Aquire

@ Description

This function is used to poll the A/D conversion data. It will trigger the A/D conversion, and read the 12 bits A/D data until the data is ready ('data-ready' bit becomes to low).

@ Syntax

Microsoft C/C++

int W_8112_AD_Aquire (int *ad_data)

Visual Basic

Windows 3.11 Version:

W_8112_AD_Aquire (ad_data As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8112_AD_Aquire (ad_data As Long) As Long

@ Argument

ad_data :	12 bits A/D converted value, the value should be within 0 and 4095.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_AD_AquireTimeOut

2.4.13 W_8112_CLR_IRQ

@ Description

This function is used to clear interrupt request which is requested by the ACL-8112. If you use interrupt to transfer A/D converted data, you should use this function to clear interrupt request status, otherwise the new interrupt signal can not be generated.

@ Syntax

Microsoft C/C++

int W_8112_CLR_IRQ (void)

Visual Basic

Windows 3.11 Version:

W_8112_CLR_IRQ () As Integer

Win-95/98, Win-NT/2000 Version:

W_8112_CLR_IRQ () As Long

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.4.14 W_8112_AD_DMA_Start

@ Description

The function will perform A/D conversion N times with DMA data transfer by using the pacer trigger (internal timer trigger) or external trigger source. It will take place in the background and will not be stopped until the N-th conversion has been completed or your program executes W_8112_AD_DMA_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function W_8112_AD_DMA_Status(). The function performs on single A/D channel with fixed A/D range.

Note: W_8112_AD_DMA_Start() and W_8112_AD_DMA_Stop() are pair function, i.e., you have to call W_8112_AD_DMA_Stop() after W_8112_AD_DMA_Start(), otherwise the A/D converted data will not be stored in the buffer you specified.

@ Syntax

Microsoft C/C++

```
int W_8112_DMA_Start (int ad_ch_no, int ad_range, int
          dma_ch_no, int irq_ch_no, int count , unsigned
          short *ad_buffer, unsigned int c1, unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_8112_DMA_Start (ByVal ad_ch_no As Integer, ByVal
          ad_range As Integer, ByVal dma_ch_no As Integer,
          ByVal irq_ch_no As Integer, ByVal count As
          Integer, ad_buffer As Integer,  ByVal c1 As Integer,
          ByVal c2 As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8112_DMA_Start (ByVal ad_ch_no As Long, ByVal
          ad_range As Long, ByVal dma_ch_no As Long,
          ByVal irq_ch_no As Long, ByVal count As Long,
          ad_buffer As Integer,  ByVal c1 As Long, ByVal c2
          As Long) As Long
```

@ Argument

ad_ch_no :	A/D channel number
ad_gain :	A/D range value. Please refer to section 2.4.9 for valid range value.
dma_ch_no :	DMA channel number, DMA_CH_1 or DMA_CH_3
irq_ch_no :	IRQ channel number, used to stop DMA
count :	the number of A/D conversion to perform
ad_buffer :	the start address of the memory buffer to store the A/D data, the buffer size must be larger than the number of A/D conversion.
c1 :	the 16-bit timer frequency divider of timer channel #1
c2 :	the 16-bit timer frequency divider of timer channel #2

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidDMAChannel,
ERR_InvalidIRQChannel, ERR_InvalidTimerValue

2.4.15 W_8112_AD_ContDMA_Start

@ Description

The function will perform continuous A/D conversions with DMA data transfer by using the pacer trigger (internal timer trigger) or external trigger source.
It will take place in the background and will not be stopped until your program executes W_8112_AD_DMA_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function W_8112_AD_DblBufferHalfReady().The function performs on single A/D channel with fixed A/D range.

Note: W_8112_AD_ContDMA_Start() and W_8112_AD_DMA_Stop() are pair function, i.e., you have to call W_8112_AD_DMA_Stop() after W_8112_AD_ContDMA_Start(), otherwise the A/D conversion will never stop .

@ Syntax

Microsoft C/C++

```
int W_8112_ContDMA_Start (int ad_ch_no, int ad_range,
          int dma_ch_no, int irq_ch_no, int count , unsigned
          short *ad_buffer, unsigned int c1, unsigned int c2)
```

Visual Basic

```
W_8112_ContDMA_Start (ByVal ad_ch_no As Long,
          ByVal ad_range As Long, ByVal dma_ch_no As
          Long, ByVal irq_ch_no As Long, ByVal count As
          Long, ad_buffer As Integer,  ByVal c1 As Long,
          ByVal c2 As Long) As Long
```

@ Argument

ad_ch_no : A/D channel number
ad_gain : A/D range value. Please refer to section 2.4.9 for valid range value.
dma_ch_no : DMA channel number, DMA_CH_1 or DMA_CH_3
irq_ch_no : IRQ channel number, used to stop DMA
count : the number of A/D conversion to perform
ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must be larger than the number of A/D conversion.

c1 :	the 16-bit timer frequency divider of timer channel #1
c2 :	the 16-bit timer frequency divider of timer channel #2

Note： While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError
ERR_AD_DMANotSet
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidDMAChannel,
ERR_InvalidIRQChannel, ERR_InvalidTimerValue

2.4.16 W_8112_AD_DMA_Status

@ Description

Since the W_8112_AD_DMA_Start function executes in background, you can issue the function W_8112_AD_DMA_Status() to check its operation status.

@ Syntax

Microsoft C/C++

```
int W_8112_AD_DMA_Status (int *status , int *count)
```

Visual Basic

Windows 3.11 Version:

```
W_8112_AD_DMA_Status (status As Integer, count As
        Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

W_8112_AD_DMA_Status (status As Long, count As Long) As Long

@ Argument

status :	status of the DMA data transfer AD_DMA_STOP : A/D DMA is completed AD_DMA_RUN : A/D DMA is not completed
count :	the number of A/D data which has been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ADDMANotSet

2.4.17 W_8112_AD_DMA_Stop

@ Description

This function is used to stop the DMA data transfer. After executing this function, the internal A/D trigger is disable and the A/D timer (timer #1 and #2) is stopped. The function returns the number of the data which has been transferred, no matter the A/D DMA data transfer is stopped by this function or by the DMA terminal count ISR.

This function has to be called after W_8112_AD_DMA_Start() function issued. Otherwise, all converted data will not be saved into the memory buffer you specified in your program.

@ Syntax

Microsoft C/C++

```
int W_8112_AD_DMA_Stop (int *count)
```

Visual Basic

Windows 3.11 Version:

W_8112_AD_DMA_Stop (count As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8112_AD_DMA_Stop (count As Long) As Long

@ Argument

count : the number of A/D converted data which has been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ADDMANotSet

2.4.18 W_8112_AD_INT_Start

@ Description

This function will perform A/D conversion N times with interrupt data transfer by using internal pacer trigger or external trigger source. It will take place in the background which will not be stopped until the N-th conversion has been completed or your program execute W_8112_AD_INT_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function W_8112_AD_INT_Status(). The function is performed on single A/D channel with fixed gain.

Note: W_8112_AD_INT_Start() and W_8112_AD_INT_Stop() are a pair function, i.e., you have to call W_8112_AD_INT_Stop() after W_8112_AD_INT_Start(), otherwise the A/D converted data will not be stored in the buffer you had specified.

@ Syntax

Microsoft C/C++

```
int W_8112_AD_INT_Start (int ad_ch_no, int ad_gain,
        int irq_ch_no, int count, unsigned short *ad_buffer,
        unsigned int c1, unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

W_8112_AD_INT_Start (ByVal ad_ch_no As Integer, ByVal ad_gain As Integer, ByVal irq_ch_no As Integer, ByVal count As Integer, ad_buffer As Integer, ByVal c1 As Integer, ByVal c2 As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8112_AD_INT_Start (ByVal ad_ch_no As Long, ByVal ad_gain As Long, ByVal irq_ch_no As Long, ByVal count As Long, ad_buffer As Integer, ByVal c1 As Long, ByVal c2 As Long) As Long

@ Argument

ad_ch_no : A/D channel number

ad_gain : A/D range value. Please refer to section 2.4.9 for valid range value.

irq_ch_no : IRQ channel number

count : the numbers of A/D conversion

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must large than the number of A/D conversion. Only the lower 12 bits of each data element in ad_buffer is meaningful. The upper 4 bits may contains some data, but this data should be ignored.

c1 : the 16-bit timer frequency divider of timer channel #1

c2 : the 16-bit timer frequency divider of timer channel #2

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidIRQChannel,
ERR_InvalidTimerValue

@ Example

Visual Basic (Win-95/98, Win-NT/2000 Version)

```
Dim ad_buf(1024) As Integer
Dim Channel As Long, Gain As Long, Irq As Long
Dim ad_count As Long, c1 As Long, c2 As Long
Dim Ret As Long
      .
      .
ad_count = 1024
      .
      .
Ret = W_8112_AD_INT_Start (Channel, Gain, Irq, ad_count,
        ad_buf(0), c1, c2)       .  .  .
```

2.4.19 W_8112_AD_INT_Status

@ Description

Since the W_8112_AD_INT_Start() function executes in background, you can issue the function W_8112_AD_INT_Status() to check the status of interrupt operation.

@ Syntax

Microsoft C/C++

```
int W_8112_AD_INT_Status (int *status , int *count)
```

Visual Basic

Windows 3.11 Version:

```
W_8112_AD_INT_Status (status As Integer, count As
          Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

W_8112_AD_INT_Status (status As Long, count As Long)
As Long

@ Argument

status :	status of the interrupt data transfer AD_INT_STOP : interrupt A/D is completed AD_INT_RUN : interrupt A/D is not completed
count :	the number of A/D data which has been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.4.20 W_8112_AD_INT_Stop

@ Description

This function is used to stop the interrupt data transfer function. After executing this function, the internal A/D trigger is disabled and the A/D timer is stopped. The function returns the number of the data which has been transferred, no matter whether if the A/D interrupt data transfer is stopped by this function or by the W_8112_AD_INT_Start() itself.

This function has to be called after W_8112_AD_INT_Start() function issued. Otherwise, all converted data will not be saved into the memory buffer you had specified in W_8112_AD_INT_Start() function call.

@ Syntax

Microsoft C/C++

int W_8112_AD_INT_Stop (int *count)

Visual Basic

Windows 3.11 Version:

W_8112_AD_INT_Stop (count As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8112_AD_INT_Stop (count As Long) As Long

@ Argument

count : the number of A/D data which have been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.4.21W_8112_AD_ContINT_Start

@ Description

The function will perform continuous A/D conversions with interrupt data transfer by using timer pacer (internal clock trigger). It will take place in the background which will not be stopped until your program execute W_8112_AD_INT_Stop() function to stop the process. After calling this function, it is necessary to check the status of the operation by using the function W_8112_AD_DblBufferHalfReady().

@ Syntax

Microsoft C/C++

```
int W_8112_ContINT_Start (int ad_ch_no, Boolean
          autoscan , int ad_gain, int irq_ch_no, int count,
          unsigned short *ad_buffer, unsigned int c1,
          unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_8112_AD_ContINT_Start (ByVal ad_ch_no As Integer,
          ByVal auto_scan As Integer, ByVal ad_gain As
          Integer, ByVal irq_ch_no As Integer, ByVal count
```

As Integer, ad_buffer As Integer, ByVal c1 As Integer, ByVal c2 As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8112_AD_ContINT_Start (ByVal ad_ch_no As Long, ByVal auto_scan As Integer, ByVal ad_gain As Long, ByVal irq_ch_no As Long, ByVal count As Long, ad_buffer As Integer, ByVal c1 As Long, ByVal c2 As Long) As Long

@ Argument

ad_ch_no : A/D channel number

If autoscan is enabled, the A/D channel scan sequence will be: 0, 1, 2, 3,…[ad_ch_no], 0, 1, …, [ad_ch_no], …
If autoscan is disabled, only the data from channel [ad_ch_no] will be converted.

autoscan:	FALSE: autoscan is disabled TRUE: autoscan is enabled
ad_gain :	A/D range value. Please refer to section 2.4.9 for valid range value.
irq_ch_no :	IRQ channel number used to transfer A/D data, the possible value is defined in file DLL2.H
count :	the number of A/D conversion
ad_buffer :	the start address of the memory buffer to store the A/D data, the buffer size must be large than the number of A/D conversion.
c1 :	the 16-bit timer frequency divider of timer channel #1
c2 :	the 16-bit timer frequency divider of timer channel #2

@ Return Code

ERR_NoError
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidIRQChannel,
ERR_InvalidTimerValue
ERR_AD_INTNotSet

2.4.22W_8112_AD_SCANINT_Start

@ Description

This function is used to start automatic channel scan . If autoscan mode is started and the end channel number is set as n by argument **ad_ch_no**, the data will be converted automatically from channel 0 to channel n.
For example, the channel is set as 4 and autoscan is started, the A/D conversion sequence will be 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, ... If the autoscan is finished, the converted channel will be kept at the specified channel, i.e. channel 4.

@ Syntax

Microsoft C/C++

```
int W_8112_AD_SCANINT_Start( int ad_ch_no, int
    ad_gain , int irq_no, int count , unsigned short
    *ad_buffer , unsigned int c1 , unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_8112_AD_ SCANINT_Start (ByVal ad_ch_no As
        Integer, ByVal ad_gain As Integer, ByVal irq_ch_no
        As Integer, ByVal count As Integer, ad_buffer As
        Integer, ByVal c1 As Integer, ByVal c2 As Integer)
        As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8112_AD_ SCANINT_Start (ByVal ad_ch_no As Long,
        ByVal ad_gain As Long, ByVal irq_ch_no As Long,
        ByVal count As Long, ad_buffer As Integer, ByVal
        c1 As Long, ByVal c2 As Long) As Long
```

@ Argument

ad_ch_no :	end A/D channel number for AutoScan
ad_gain :	A/D range value. Please refer to section 2.4.9 for valid range value.
irq_ch_no :	IRQ channel number
count :	the numbers of A/D conversion

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must large than the number of A/D conversion. Only the lower 12 bits of each data element in ad_buffer is meaningful. The upper 4 bits may contains some data, but this data should be ignored.

c1 : the 16-bit timer frequency divider of timer channel #1

c2 : the 16-bit timer frequency divider of timer channel #2

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADChannel
ERR_AD_InvalidGain
ERR_InvalidIRQChannel
ERR_InvalidTimerValue

2.4.23W_8112_AD_DblBufferHalfReady

@ Description

Checks whether the next half buffer of data in circular buffer is ready for transfer during an double-buffered analog input operation.

@ Syntax

Microsoft C/C++

```
int W_8112_AD_DblBufferHalfReady ( BOOLEAN
        *bHalfReady)
```

Visual Basic

```
W _8112_AD_DblBufferHalfReady (bHalfReady As Long)
        As Long
```

@ Argument

bHalfReady : Whether the next half buffer of data is available.If *HalfReady* = TRUE, you can call **`W_8112_AD_DblBufferTransfer()`** to copy the data to your user buffer.

@ Return Code

ERR_NoError
ERR_InvalidMode

2.4.24W_8112_AD_DblBufferTransfer

@ Description

Depending on the continuous AI function elected, half of the data in circular buffer will be logged into the user buffer.
You can execute this function repeatedly to return sequential half buffers of the data.

@ Syntax

Microsoft C/C++

```
int W_8112_AD_DblBufferTransfer (USHORT *pwBuffer)
```

Visual Basic

```
W _8112_AD_DblBufferTransfer (pwBuffer As Integer) As
        Long
```

@ Argument

pwBuffer: The user buffer. An integer array to which the data is to be copied.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.4.25 W_8112_AD_Timer

@ Description

This function is used to setup the Timer #1 and Timer #2. The values of c1 and c2 are used as frequency dividers for generating constant A/D sampling rate dedicatedly. It is possible to stop the pacer trigger by setting any one of the dividers as 0. Because the A/D conversion rate is limited due to the conversion time of the A/D converter, the highest sampling rate of the ACL-8112 can not exceed 100 KHz. The multiplication of the dividers must be larger than 20.

@ Syntax

Microsoft C/C++

int W_8112_AD_Timer(unsigned int c1, unsigned int c2)

Visual Basic

Windows 3.11 Version:

W_8112_AD_Timer (ByVal c1 As Integer, ByVal c2 As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8112_AD_Timer (ByVal c1 As Long, ByVal c2 As Long) As Long

@ Argument

c1 : frequency divider of timer #1
c2 : frequency divider of timer #2

Note : the A/D sampling rate is equal to : 2MHz / (c1*c2), when c1 = 0 or c2 = 0, the pacer trigger will be stopped.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerValue

2.4.26 W_8112_Timer_Start

@ Description

The Timer #0 on the ACL-8112 can be freely programmed by the users. This function is used to program the Timer #0. This timer can be used as frequency generator if internal clock is used. It also can be used as event counter if external clock is used. All the 8253 modes are available. Please refer to "Timer/Counter 8253" in 8112's user's manual Appendix B.

@ Syntax

Microsoft C/C++

```
int W_8112_Timer_Start (int timer_mode, unsigned int c0)
```

Visual Basic

Windows 3.11 Version:

```
W_8112_Timer_Start (ByVal timer_mode As Integer,
          ByVal c0 As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8112_Timer_Start (ByVal timer_mode As Long, ByVal
          c0 As Long) As Long
```

@ Argument

timer_mode : the 8253 timer mode, the possible values are : TIMER_MODE0, TIMER_MODE1, TIMER_MODE2, TIMER_MODE3, TIMER_MODE4, TIMER_MODE5.

c0 : the counter value of timer

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerMode

2.4.27 W_8112_Timer_Read

@ Description

This function is used to read the counter value of the Timer #0.

@ Syntax

Microsoft C/C++

int W_8112_Timer_Read (unsigned int *counter_value)

Visual Basic

Windows 3.11 Version:

W_8112_Timer_Read (counter_value As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8112_Timer_Read (counter_value As Long) As Long

@ Argument

counter_value : the counter value of the Timer #0

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.4.28 W_8112_Timer_Stop

@ Description

This function is used to stop the timer operation. The timer is set to the 'One-shot' mode with counter value '0'. That is, the clock output signal will be set to high after executing this function.

@ Syntax

Microsoft C/C++

int W_8112_Timer_Stop (unsigned int *counter_value)

Visual Basic

Windows 3.11 Version:

W_8112_Timer_Stop (counter_value As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8112_Timer_Stop (counter_value As Long) As Long

@ Argument

counter_value : the current counter value of the Timer #0

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.4.29 W_8112_DMA_InitialMemoryAllocated

@ Description

This function is only available in Windows NT and Windows 2000 system. This function returns the available memory size for DMA data transfer in the device driver in argument ***MemSize***. While performming analog input with DMA data transfer, the analog input size can not exceed this size.

@ Syntax

Microsoft C/C++

W_8112_DMA_InitialMemoryAllocated(int *MemSize)

Visual Basic

Win-NT/2000 Version:

W_8112_DMA_InitialMemoryAllocated(MemSize As Long)
As Long

@ Argument

MemSize : the available memory size for DMA data transfer in device driver of ACL-8112DG/HG.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.5 8112PG Software DLL Driver

In this section, the ACL-8112PG's software DLL drivers are described. The function names of Windows 3.11, Window 95/98, and Windows NT/2000 versions are the same. So, users do not need to learn the difference between them. The application's portability between these three systems can be very high.

2.5.1 W_812_Initial

@ Description

An ACL-8112PG card is initialized according to the card number and the corresponding base address. Each ACL-8112PG multi-function data acquisition card has to be initialized by this function before calling other functions.

Note: In this library, if you want to operate DMA or interrupt operation, only two ACL-8112PG cards can be initialized. The reason is only two DMA channels are supported in the card.

@ Syntax

Microsoft C/C++

```
int W_812_Initial (int card_number, int base_addresss)
```

Visual Basic

Windows 3.11 Version:

```
W_812_Initial (ByVal card_number As Integer, ByVal
          base_address As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_Initial (ByVal card_number As Long, ByVal
          base_address As Long) As Long
```

@ Argument

card_number : The card number to be initialized. If all the ACL-8112PG cards only perform software polling, eight cards can be initialized and the valid card numbers are CARD_1, CARD_2, …, CARD_8. However, if the ACL-8111PG cards are operated in *Windows NT* system and will perform *interrupt or DMA data transfer*, only two cards can be initialized and the card number must be CARD_1 or CARD_2.

base_address : the I/O port base address of the card.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError

2.5.2 W_812_Switch_Card_No

@ Description

After initialized more than one ACL-8112PG cards, this function is used to select which card is used currently.

@ Syntax

Microsoft C/C++

```
int W_812_Switch_Card_No (int card_number)
```

Visual Basic

Windows 3.11 Version:

```
W_812_Switch_Card_No (ByVal card_number As Integer)
        As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_Switch_Card_No (ByVal card_number As Long)
        As Long
```

@ Argument

card_number : The card number of the card that is set to be active. If all the ACL-8112PG cards only perform software polling, the valid card numbers are CARD_1, CARD_2, …, CARD_8. However, if the ACL-8112PG cards are operated in *Windows NT* system and will perform *interrupt or DMA data transfer*, only two cards can be initialized and the card number must be CARD_1 or CARD_2.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber

2.5.3 W_812_DI

@ Description

This function is used to read data from digital input port. There are 16 digital inputs on the ACL-8112PG. The bit 0 to bit 7 are defined as **low byte** and the bit 8 to bit 15 are defined as **high byte**.

@ Syntax

Microsoft C/C++

```
int W_812_DI (int port_number, unsigned char *di_data)
```

Visual Basic

Windows 3.11 Version:

```
W_812_DI (ByVal port_number As Integer, di_data As
        Byte) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_DI (ByVal port_number As Integer, di_data As
        Byte) As Long
```

@ Argument

port_number : To indicate which port is read, DI_LOW_BYTE or DI_HIGH_BYTE.
DI_LOW_BYTE : bit 0 ~ bit 7

DI_HIGH_BYTE : bit8 ~ bit15

di_data : return value from digital port.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.5.4 W_812_DI _Channel

@ Description

This function is used to read data from digital input channels (bit). There are 16 digital input channels on the ACL-8112PG. When performs this function, the digital input port is read and the value of the corresponding channel is returned.

* channel means each bit of digital input ports.

@ Syntax

Microsoft C/C++

```
int _812_DI_Channel (int di_ch_no, unsigned int *di_data)
```

Visual Basic

Windows 3.11 Version:

```
W_812_DI_Channel (ByVal di_ch_no As Integer, di_data
        As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_DI_Channel (ByVal di_ch_no As Long, di_data As
        Long) As Long
```

@ Argument

di_ch_no : the DI channel number, the value has to be set between 0 and 15.

di_data : return value, either 0 or 1.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDIChannel

2.5.5 W_812_DO

@ Description

This function is used to write data to digital output ports. There are 16 digital outputs on the ACL-8112PG, they are divided to two ports, DO_LOW_BYTE and DO_HIGH_BYTE. The channel 0 to channel 7 are defined in DO_LOW_BYTE port and the channel 8 to channel 15 are defined as the DO_HIGH_BYTE port.

@ Syntax

Microsoft C/C++

```
int W_812_DO (int port_number, unsigned char do_data)
```

Visual Basic

Windows 3.11 Version:

```
W_812_DO (ByVal port_number As Integer, ByVal
        do_data As Byte) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_DO (ByVal port_number As Long, ByVal do_data
        As Byte) As Long
```

@ Argument

port_number : DO_LOW_BYTE or DO_HIGH_BYTE
do_data : value will be written to digital output port

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.5.6 W_812_DA

@ Description

This function is used to write data to D/A converters. There are two Digital-to-Analog conversion channels on the ACL-8112PG. The resolution of each channel is 12-bit, i.e. the range is from 0 to 4095.

@ Syntax

Microsoft C/C++

```
int W_812_DA (int da_ch_no, unsigned int da_data)
```

Visual Basic

Windows 3.11 Version:

```
W_812_DA (ByVal da_ch_no As Integer, ByVal da_data
        As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_DA (ByVal da_ch_no As Long, ByVal da_data As
        Long) As Long
```

@ Argument

da_ch_no : D/A channel number, the valid data is :

0	Channel AO1
1	Channel AO2

da_data : D/A converted value, if the value is greater than 4095, the higher 4 bits are negligent.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDAChannel

2.5.7 W_812_AD_Set_Channel

@ Description

This function is used to set A/D channel by means of writing data to the A/D channel multiplexer register. There are 16 single-ended A/D channels in ACL-8112PG, so the channel number should be set between 0 and 15 only. The initial state is channel 0 which is the default setting by the ACL-8112PG hardware configuration.

@ Syntax

Microsoft C/C++

int W_812_AD_Set_Channel (int ad_ch_no)

Visual Basic

Windows 3.11 Version:

W_812_AD_Set_Channel (ByVal ad_ch_no As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_812_AD_Set_Channel (ByVal ad_ch_no As Long) As Long

@ Argument

ad_ch_no : channel number to perform A/D conversion

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADChannel

2.5.8 W_812_AD_Set_Gain

@ Description

This function is used to set the A/D range by means of writing data to the range control register. The major difference between 8112DG, 8112HG, and 8112PG is each card supports different ranges which affect the input voltage range of each card. This is the only difference between these cards. Each card's gain and its corresponding A/D input ranges are listed as below.

The initial value of gain is '1', which is set by the ACL-8112PG hardware.

**** ACL-8112PG :**
If input voltage range is set to ±5 V (JP9),

Input Range (V)	Gain	Gain Code
±5 V	X 1	AD_GAIN_1
±2.5 V	X 2	AD_GAIN_2
±1.25 V	X 4	AD_GAIN_4
±0.625 V	X 8	AD_GAIN_8
±0.3125V	X 16	AD_GAIN_16

If input voltage range is set to ±10 V (JP9),

Input Range (V)	Gain	Gain Code
±10 V	X 1	AD_GAIN_1
±5 V	X 2	AD_GAIN_2
±2.5 V	X 4	AD_GAIN_4
±1.25 V	X 8	AD_GAIN_8
±0.625 V	X 16	AD_GAIN_16

Note : This function will not check if you setup a right gain code for different data acquisition cards, so you should be very careful what kind of data acquisition card you use, and setup a right Gain code.

@ Syntax

Microsoft C/C++

```
int W_812_AD_Set_Gain (int ad_gain)
```

Visual Basic

Windows 3.11 Version:

W_812_AD_Set_Gain (ByVal ad_gain As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_812_AD_Set_Gain (ByVal ad_gain As Long) As Long

@ Argument

ad_gain : the programmable gain of A/D conversion, the possible values are:

* ACL-8112PG :
AD_GAIN_1, AD_GAIN_2, AD_GAIN_4, AD_GAIN_8, AD_GAIN_16

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADGain

2.5.9 W_812_AD_Set_Mode

@ Description

This function is used to set the A/D trigger and data transfer mode by means of writing data to the mode control register. The hardware initial state of the ACL-8112PG is set as A8112_AD_MODE_1 software (internal) trigger with program polling.

A/D Mode	Description
A8112_AD_MODE_0	External Trigger, Software Polling
A8112_AD_MODE_1	Software Trigger, Software Polling
A8112_AD_MODE_2	Timer Trigger, DMA Transfer
A8112_AD_MODE_3	External Trigger, DMA Transfer
A8112_AD_MODE_4	External Trigger, Interrupt Transfer
A8112_AD_MODE_5	Software Trigger, Interrupt Transfer
A8112_AD_MODE_6	Timer Trigger, Interrupt Transfer

A8112_AD_MODE_7	Not Used

@ Syntax

Microsoft C/C++

int W_812_AD_Set_Mode (int ad_mode)

Visual Basic

Windows 3.11 Version:

W_812_AD_Set_Mode (ByVal ad_mode As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_812_AD_Set_Mode (ByVal ad_mode As Long) As Long

@ Argument

ad_mode : A/D trigger and data transfer mode

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidMode

2.5.10 W_812_AD_Soft_Trig

@ Description

This function is used to trigger the A/D conversion by software. When the function is called, a trigger pulse will be generated and the converted data will be stored in the address Base+4 and Base+5, and can be retrieved by function W_812_AD_Aquire().

@ Syntax

Microsoft C/C++

int W_812_AD_Soft_Trig (void)

Visual Basic

Windows 3.11 Version:

W_812_AD_Soft_Trig ( ) As Integer

Win-95/98, Win-NT/2000 Version:

W_812_AD_Soft_Trig ( ) As Long

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.5.11 W_812_AD_Aquire

@ Description

This function is used to poll the A/D conversion data. It will trigger the A/D conversion, and read the 12 bits A/D data until the data is ready ('data-ready' bit becomes to low).

@ Syntax

Microsoft C/C++

int W_812_AD_Aquire (int *ad_data)

Visual Basic

Windows 3.11 Version:

W_812_AD_Aquire (ad_data As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_812_AD_Aquire (ad_data As Long) As Long

@ Argument

ad_data : 12 bits A/D converted value, the value should be within 0 and 4095.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_AD_AquireTimeOut

2.5.12 W_812_CLR_IRQ

@ Description

This function is used to clear interrupt request which is requested by the ACL-8112PG. If you use interrupt to transfer A/D converted data, you should use this function to clear interrupt request status, otherwise the new interrupt signal can not be generated.

@ Syntax

Microsoft C/C++

```
int W_812_CLR_IRQ (void)
```

Visual Basic

Windows 3.11 Version:

```
W_812_CLR_IRQ () As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_CLR_IRQ () As Long
```

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.5.13 W_812_AD_DMA_Start

@ Description

The function will perform A/D conversion N times with DMA data transfer by using the pacer trigger (internal timer trigger) or external trigger source. It will take place in the background and will not be stopped until the N-th conversion has been completed or your program executes W_812_AD_DMA_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function W_812_AD_DMA_Status(). The function performs on single A/D channel with fixed A/D range.

Note: W_812_AD_DMA_Start() and W_812_AD_DMA_Stop() are pair function, i.e., you have to call W_812_AD_DMA_Stop() after W_812_AD_DMA_Start(), otherwise the A/D converted data will not be stored in the buffer you specified.

@ Syntax

Microsoft C/C++

```
int W_812_DMA_Start (int ad_ch_no, int ad_range, int
          dma_ch_no, int irq_ch_no, int count , unsigned
          short *ad_buffer, unsigned int c1, unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_812_DMA_Start (ByVal ad_ch_no As Integer, ByVal
          ad_range As Integer, ByVal dma_ch_no As Integer,
          ByVal irq_ch_no As Integer, ByVal count As
          Integer, ad_buffer As Integer,  ByVal c1 As Integer,
          ByVal c2 As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_DMA_Start (ByVal ad_ch_no As Long, ByVal
          ad_range As Long, ByVal dma_ch_no As Long,
          ByVal irq_ch_no As Long, ByVal count As Long,
          ad_buffer As Integer,  ByVal c1 As Long, ByVal c2
          As Long) As Long
```

@ Argument

ad_ch_no : A/D channel number

ad_gain : A/D range value. Please refer to section 2.5.8 for valid range value.

dma_ch_no : DMA channel number, DMA_CH_1 or DMA_CH_3

irq_ch_no : IRQ channel number, used to stop DMA

count : the number of A/D conversion to perform

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must be larger than the number of A/D conversion.

c1 : the 16-bit timer frequency divider of timer channel #1

c2 : the 16-bit timer frequency divider of timer channel #2

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidDMAChannel,
ERR_InvalidIRQChannel, ERR_InvalidTimerValue

2.5.14 W_812_AD_ContDMA_Start

@ Description

The function will perform continuous A/D conversions with DMA data transfer by using the pacer trigger (internal timer trigger) or external trigger source.

It will take place in the background and will not be stopped until your program executes W_812_AD_DMA_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function W_812_AD_ DblBufferHalfReady().The function performs on single A/D channel with fixed A/D range.

Note: W_812_AD_ContDMA_Start() and W_812_AD_DMA_Stop() are pair function, i.e., you have to call W_812_AD_DMA_Stop() after W_812_AD_ContDMA_Start(), otherwise the A/D conversion will never stop .

@ Syntax

Microsoft C/C++

```
int W_812_ContDMA_Start (int ad_ch_no, int ad_range,
        int dma_ch_no, int irq_ch_no, int count , unsigned
        short *ad_buffer, unsigned int c1, unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_812_ContDMA_Start (ByVal ad_ch_no As Integer,
        ByVal  ad_range As Integer, ByVal dma_ch_no As
        Integer, ByVal irq_ch_no As Integer, ByVal count
        As Integer, ad_buffer As Integer,  ByVal c1 As
        Integer, ByVal c2 As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_ContDMA_Start (ByVal ad_ch_no As Long, ByVal
        ad_range As Long, ByVal dma_ch_no As Long,
        ByVal irq_ch_no As Long, ByVal count As Long,
        ad_buffer As Integer,  ByVal c1 As Long, ByVal c2
        As Long) As Long
```

@ Argument

ad_ch_no : A/D channel number

ad_gain : A/D range value. Please refer to section 2.5.8 for valid range value.

dma_ch_no : DMA channel number, DMA_CH_1 or DMA_CH_3

irq_ch_no : IRQ channel number, used to stop DMA

count : the number of A/D conversion to perform

ad_buffer :	the start address of the memory buffer to store the A/D data, the buffer size must be larger than the number of A/D conversion.
c1 :	the 16-bit timer frequency divider of timer channel #1
c2 :	the 16-bit timer frequency divider of timer channel #2

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError
ERR_AD_DMANotSet
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidDMAChannel,
ERR_InvalidIRQChannel, ERR_InvalidTimerValue

2.5.15 W_812_AD_DMA_Status

@ Description

Since the W_812_AD_DMA_Start function executes in the background, you can issue the function W_812_AD_DMA_Status() to check its operation status.

@ Syntax

Microsoft C/C++

```
int W_812_AD_DMA_Status (int *status , int *count)
```

Visual Basic

Windows 3.11 Version:

W_812_AD_DMA_Status (status As Integer, count As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_812_AD_DMA_Status (status As Long, count As Long) As Long

@ Argument

status :	status of the DMA data transfer AD_DMA_STOP : A/D DMA is completed AD_DMA_RUN : A/D DMA is not completed
count :	the number of A/D data which has been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ADDMANotSet

2.5.16 W_812_AD_DMA_Stop

@ Description

This function is used to stop the DMA data transfer. After executing this function, the internal A/D trigger is disabled and the A/D timer (timer #1 and #2) is stopped. The function returns the number of the data which has been transferred, no matter the A/D DMA data transfer is stopped by this function or by the DMA terminal count ISR.

This function has to be called after W_812_AD_DMA_Start() function issued. Otherwise, all converted data will not be saved into the memory buffer you specified in your program.

@ Syntax

Microsoft C/C++

```
int W_812_AD_DMA_Stop (int *count)
```

Visual Basic

Windows 3.11 Version:

W_812_AD_DMA_Stop (count As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_812_AD_DMA_Stop (count As Long) As Long

@ Argument

count : the number of A/D converted data which has been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ADDMANotSet

2.5.17 W_812_AD_INT_Start

@ Description

This function will perform A/D conversion N times with interrupt data transfer by using internal pacer trigger or external trigger source. It will take place in the background which will not be stopped until the N-th conversion has been completed or your program execute W_812_AD_INT_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function W_812_AD_INT_Status(). The function is performed on single A/D channel with fixed gain.

Note: W_812_AD_INT_Start() and W_812_AD_INT_Stop() are a pair function, i.e., you have to call W_812_AD_INT_Stop() after W_812_AD_INT_Start(), otherwise the A/D converted data will not be stored in the buffer you had specified.

@ Syntax

Microsoft C/C++

int W_812_AD_INT_Start (int ad_ch_no, int ad_gain, int irq_ch_no, int count, unsigned short *ad_buffer, unsigned int c1, unsigned int c2)

Visual Basic

Windows 3.11 Version:

W_812_AD_INT_Start (ByVal ad_ch_no As Integer, ByVal ad_gain As Integer, ByVal irq_ch_no As Integer, ByVal count As Integer, ad_buffer As Integer, ByVal c1 As Integer, ByVal c2 As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_812_AD_INT_Start (ByVal ad_ch_no As Long, ByVal ad_gain As Long, ByVal irq_ch_no As Long, ByVal count As Long, ad_buffer As Integer, ByVal c1 As Long, ByVal c2 As Long) As Long

@ Argument

ad_ch_no : A/D channel number

ad_gain : A/D range value. Please refer to section 2.5.8 for valid range value.

irq_ch_no : IRQ channel number

count : the numbers of A/D conversion

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must large than the number of A/D conversion. Only the lower 12 bits of each data element in ad_buffer is meaningful. The upper 4 bits may contains some data, but this data should be ignored.

c1 : the 16-bit timer frequency divider of timer channel #1

c2 : the 16-bit timer frequency divider of timer channel #2

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidIRQChannel,
ERR_InvalidTimerValue

@ Example

Visual Basic (Win-95/98, Win-NT/2000 Version)

```
Dim ad_buf(1024) As Integer
Dim Channel As Long, Gain As Long, Irq As Long
Dim ad_count As Long, c1 As Long, c2 As Long
Dim Ret As Long
      .
      .
ad_count = 1024
      .
      .
Ret = W_812_AD_INT_Start (Channel, Gain, Irq, ad_count,
        ad_buf(0), c1, c2)      . . .
```

2.5.18 W_812_AD_INT_Status

@ Description

Since the W_812_AD_INT_Start() function executes in background, you can issue the function W_812_AD_INT_Status() to check the status of interrupt operation.

@ Syntax

Microsoft C/C++

int W_812_AD_INT_Status (int *status , int *count)

Visual Basic

Windows 3.11 Version:

W_812_AD_INT_Status (status As Integer, count As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_812_AD_INT_Status (status As Long, count As Long) As Long

@ Argument

status : status of the interrupt data transfer
AD_INT_STOP : interrupt A/D is completed
AD_INT_RUN : interrupt A/D is not completed

count : the number of A/D data which has been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.5.19 W_812_AD_INT_Stop

@ Description

This function is used to stop the interrupt data transfer function. After executing this function, the internal A/D trigger is disable and the A/D timer is stopped. The function returns the number of the data which has been transferred, no matter whether if the A/D interrupt data transfer is stopped by this function or by the W_812_AD_INT_Start() itself.

This function has to be called after W_812_AD_INT_Start() function issued. Otherwise, all converted data will not be saved into the memory buffer you had specified in W_812_AD_INT_Start() function call.

@ Syntax

Microsoft C/C++

```
int W_812_AD_INT_Stop (int *count)
```

Visual Basic

Windows 3.11 Version:

```
W_812_AD_INT_Stop (count As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_AD_INT_Stop (count As Long) As Long
```

@ Argument

count : the number of A/D data which have been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.5.20 W_812_AD_ContINT_Start

@ Description

The function will perform continuous A/D conversions with interrupt data transfer by using timer pacer (internal clock trigger). It will take place in the background which will not be stopped until your program execute W_812_AD_INT_Stop() function to stop the process. After calling this function, it is necessary to check the status of the operation by using the function W_812_AD_DblBufferHalfReady().

@ Syntax

Microsoft C/C++

```
int W_812_ContINT_Start (int ad_ch_no, Boolean
          autoscan, int ad_gain, int irq_ch_no, int count,
          unsigned short *ad_buffer, unsigned int c1,
          unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_812_AD_ContINT_Start (ByVal ad_ch_no As Integer,
          ByVal auto_scan As Integer, ByVal ad_gain As
          Integer, ByVal irq_ch_no As Integer, ByVal count
          As Integer, ad_buffer As Integer, ByVal c1 As
          Integer, ByVal c2 As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_AD_ContINT_Start (ByVal ad_ch_no As Long,
          ByVal auto_scan As Integer, ByVal ad_gain As
          Long, ByVal irq_ch_no As Long, ByVal count As
          Long, ad_buffer As Integer, ByVal c1 As Long,
          ByVal c2 As Long) As Long
```

@ Argument

ad_ch_no : A/D channel number

If autoscan is enabled, the A/D channel scan sequence will be: 0, 1, 2, 3,…[ad_ch_no], 0, 1, …, [ad_ch_no], …
If autoscan is disabled, only the data from channel [ad_ch_no] will be converted.

autoscan: FALSE: autoscan is disabled
TRUE: autoscan is enabled

ad_gain : A/D range value. Please refer to section 2.5.8 for valid range value.

irq_ch_no : IRQ channel number

count : the numbers of A/D conversion

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must large than the number of A/D conversion. Only the lower 12 bits of each data element in ad_buffer is meaningful. The upper 4 bits may contains some data, but this data should be ignored.

c1 : the 16-bit timer frequency divider of timer channel #1

c2 : the 16-bit timer frequency divider of timer channel #2

@ Return Code

ERR_NoError
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidIRQChannel,
ERR_InvalidTimerValue
ERR_AD_INTNotSet

2.5.21 W_812_AD_SCANINT_Start

@ Description

This function is used to start automatic channel scan . If autoscan mode is started and the end channel number is set as n by argument **ad_ch_no**, the data will be converted automatically from channel 0 to channel n.
For example, the channel is set as 4 and autoscan is started, the A/D conversion sequence will be 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, ... If the autoscan is finished, the converted channel will be kept at the specified channel, i.e. channel 4.

@ Syntax

Microsoft C/C++

```
int W_812_AD_SCANINT_Start( int ad_ch_no, int
    ad_gain , int irq_no, int count , unsigned short
    *ad_buffer , unsigned int c1 , unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_812_AD_ SCANINT_Start (ByVal ad_ch_no As Integer,
        ByVal ad_gain As Integer, ByVal irq_ch_no As
        Integer, ByVal count As Integer, ad_buffer As
        Integer, ByVal c1 As Integer, ByVal c2 As Integer)
        As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_AD_ SCANINT_Start (ByVal ad_ch_no As Long,
          ByVal ad_gain As Long, ByVal irq_ch_no As Long,
          ByVal count As Long, ad_buffer As Integer, ByVal
          c1 As Long, ByVal c2 As Long) As Long
```

@ Argument

ad_ch_no : end A/D channel number for AutoScan

ad_gain : A/D range value. Please refer to section 2.5.8 for valid range value.

irq_ch_no : IRQ channel number used to transfer A/D data, the possible value is defined in file DLL2.H

count : the number of A/D conversion

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must be large than the number of A/D conversion.

c1 : the 16-bit timer frequency divider of timer channel #1

c2 : the 16-bit timer frequency divider of timer channel #2

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADChannel
ERR_AD_InvalidGain
ERR_InvalidIRQChannel
ERR_InvalidTimerValue

2.5.22 W_812_AD_DblBufferHalfReady

@ Description

Checks whether the next half buffer of data in circular buffer is ready for transfer during an double-buffered analog input operation.

@ Syntax

Microsoft C/C++

```
int W_812_AD_DblBufferHalfReady ( BOOLEAN
        *bHalfReady)
```

Visual Basic

```
W_812_AD_DblBufferHalfReady (bHalfReady As Long) As
        Long
```

@ Argument

bHalfReady : Whether the next half buffer of data is available.If *HalfReady* = TRUE, you can call **`W_812_AD_DblBufferTransfer()`** to copy the data to your user buffer.

@ Return Code

ERR_NoError
ERR_InvalidMode

2.5.23 W_812_AD_DblBufferTransfer

@ Description

Depending on the continuous AI function elected, half of the data in circular buffer will be logged into the user buffer . You can execute this function repeatedly to return sequential half buffers of the data.

@ Syntax

Microsoft C/C++

```
int W_812_AD_DblBufferTransfer (USHORT *pwBuffer)
```

Visual Basic

```
W_812_AD_DblBufferTransfer (pwBuffer As Integer) As
        Long
```

@ Argument

pwBuffer: The user buffer. An integer array to which the data is to be copied.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.5.24 W_812_AD_Timer

@ Description

This function is used to setup the Timer #1 and Timer #2. The values of c1 and c2 are used as frequency dividers for generating constant A/D sampling rate dedicatedly. It is possible to stop the pacer trigger by setting any one of the dividers as 0. Because the A/D conversion rate is limited due to the conversion time of the A/D converter, the highest sampling rate of the ACL-8112PG can not exceed 100 KHz. The multiplication of the dividers must be larger than 20.

@ Syntax

Microsoft C/C++

```
int W_812_AD_Timer( unsigned int c1, unsigned int c2 )
```

Visual Basic

Windows 3.11 Version:

```
W_812_AD_Timer (ByVal c1 As Integer, ByVal c2 As
          Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_AD_Timer (ByVal c1 As Long, ByVal c2 As Long)
          As Long
```

@ Argument

c1 :	frequency divider of timer #1
c2 :	frequency divider of timer #2

Note : the A/D sampling rate is equal to : 2MHz / (c1*c2), when c1 = 0 or c2 = 0, the pacer trigger will be stopped.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerValue

2.5.25 W_812_Timer_Start

@ Description

The Timer #0 on the ACL-8112PG can be freely programmed by the users. This function is used to program the Timer #0. This timer can be used as frequency generator if internal clock is used. It also can be used as event counter if external clock is used. All the 8253 modes are available. Please refer to "Timer/Counter 8253" in 812's user's manual Appendix B.

@ Syntax

Microsoft C/C++

```
int W_812_Timer_Start (int timer_mode, unsigned int c0)
```

Visual Basic

Windows 3.11 Version:

```
W_812_Timer_Start (ByVal timer_mode As Integer, ByVal
        c0 As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_Timer_Start (ByVal timer_mode As Long, ByVal
        c0 As Long) As Long
```

@ Argument

timer_mode : the 8253 timer mode, the possible values are :
TIMER_MODE0, TIMER_MODE1,
TIMER_MODE2, TIMER_MODE3,
TIMER_MODE4, TIMER_MODE5.

c0 : the counter value of timer

@ Return Code

ERR_NoError
ERR_BoardNoInit

ERR_InvalidTimerMode

2.5.26 W_812_Timer_Read

@ Description

This function is used to read the counter value of the Timer #0.

@ Syntax

Microsoft C/C++

```
int W_812_Timer_Read (unsigned int *counter_value)
```

Visual Basic

Windows 3.11 Version:

```
W_812_Timer_Read (counter_value As Integer) As
        Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_812_Timer_Read (counter_value As Long) As Long
```

@ Argument

counter_value : the counter value of the Timer #0

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.5.27 W_812_Timer_Stop

@ Description

This function is used to stop the timer operation. The timer is set to the 'One-shot' mode with counter value '0'. That is, the clock output signal will be set to high after executing this function.

@ Syntax

Microsoft C/C++

int W_812_Timer_Stop (unsigned int *counter_value)

Visual Basic

Windows 3.11 Version:

W_812_Timer_Stop (counter_value As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_812_Timer_Stop (counter_value As Long) As Long

@ Argument

counter_value : the current counter value of the Timer #0

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.5.28 W_812_DMA_InitialMemoryAllocated

@ Description

This function is only available in Windows NT and Windows 2000 system. This function returns the available memory size for DMA data transfer in the device driver in argument ***MemSize***. While performming analog input with DMA data transfer, the analog input size can not exceed this size.

@ Syntax

Microsoft C/C++

W_812_DMA_InitialMemoryAllocated(int *MemSize)

Visual Basic

Win-NT/2000 Version:

W_812_DMA_InitialMemoryAllocated(MemSize As Long)
As Long

@ Argument

MemSize : the available memory size for DMA data transfer in device driver of ACL-8112PG.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.6 8113 / 8113A Software DLL Driver

In this section, the software DLL drivers of ACL-8113/8113A are described. The function names of Windows 3.11, Window 95/98, Windows NT/2000 versions are the same. So, users do not need to learn the difference between them. The application's portability between these three systems can be very high.

Note : All functions of the ACL-8113 can be applied to the ACL-813 directly. That is, users can use the 8113.DLL for both ACL-8113 and ACL-813 data acquisition cards.

2.6.1 W_8113_Initial / W_8113A_ Initial

@ Description

An ACL-8113/8113A card is initialized according to the card number and the corresponding base address. Each ACL-8113/8113A multi-function data acquisition card has to be initialized by this function before calling other functions.

@ Syntax

Microsoft C/C++

```
int W_8113_Initial (int card_number, int base_addresss)
int W_8113A_Initial (int card_number, int base_addresss)
```

Visual Basic

Windows 3.11 Version:

```
W_8113_Initial (ByVal card_number As Integer, ByVal
        base_address As Integer) As Integer
W_8113A_Initial (ByVal card_number As Integer, ByVal
        base_address As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8113_Initial (ByVal card_number As Long, ByVal
        base_address As Long) As Long
W_8113A_Initial (ByVal card_number As Long, ByVal
        base_address As Long) As Long
```

@ Argument

card_number : The card number to be initialized, at most 8 cards can be initialized in one system, the card number must be within 0 and 7.

base_address : the I/O port base address of the card.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError

2.6.2 W_8113_ActCard_Set / W_8113A_ActCard_Set

@ Description

This function is used on multi-card system. After the ACL-8113/8113A cards are initialized by W_8113(A)_Initial() function, you can use this function to select which one you want to operate.

Note: With this library, up to eight ACL-8113/8113A cards can be initialized.

@ Syntax

Microsoft C/C++

```
int _8113_ActCard_Set (int card_number)
int _8113A_ActCard_Set (int card_number)
```

Visual Basic

Windows 3.11 Version:

```
W_8113_ActCard_Set (ByVal card_number As Integer) As
        Integer
W_8113A_ActCard_Set (ByVal card_number As Integer)
        As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8113_ActCard_Set (ByVal card_number As Long) As
        Long
W_8113A_ActCard_Set (ByVal card_number As Long) As
        Long
```

@ Argument

card_number : The card number to be initialized, totally 8 cards can be initialized. The card number must be within the range of 0 and 7.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber

2.6.3 W_8113_Channel_Select/Deselect/Clear/ChannelNo_Get W_8113A_Channel_Select/Deselect/Clear/ChannelNo_Get

@ Description

The library functions can perform the A/D conversions on multiple channels at once. You may select multiple channels to perform the A/D conversions on. The channels are not necessary to select as contiguous, i.e. the channels can be selected in any order, but the conversion sequence will be in numerical order. The functions that support A/D channel selection and de-selection are the following:

W_8113_Channel_Select / **W_8113A_Channel_Select**:
selects a particular channel for conversion.
W_8113_Channel_Deselect / **W_8113A_Channel_Deselect**:
removes a channel from the list of selected channels.
W_8113_Channel_Clear / **W_8113A_Channel_Clear**:
clears all the channels from the list of slected list, and no channel is selected.
W_8113_ChannelNo_Get / **W_8113A_ChannelNo_Get**:
returns the number of selected channels.

@ Syntax

Microsoft C/C++

```
int W_8113_Channel_Select (int channel)
int W_8113_Channel_Deselect (int channel)
int W_8113_Channel_Clear (void)
int W_8113_ChannelNo_Get (int *no)

int W_8113A_Channel_Select (int channel)
int W_8113A_Channel_Deselect (int channel)
int W_8113A_Channel_Clear (void)
int W_8113A_ChannelNo_Get (int *no)
```

Visual Basic

Windows 3.11 Version:

```
W_8113_Channel_Select (ByVal  channel As Integer) As
        Integer
W_8113_Channel_Deselect (ByVal  channel As Integer)
        As Integer
W_8113_Channel_Clear () As Integer
W_8113_ChannelNo_Get (no As Integer) As Integer

W_8113A_Channel_Select (ByVal  channel As Integer) As
        Integer
W_8113A_Channel_Deselect (ByVal  channel As Integer)
        As Integer
W_8113A_Channel_Clear () As Integer
W_8113A_ChannelNo_Get (no As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8113_Channel_Select (ByVal  channel As Long) As
        Long
W_8113_Channel_Deselect (ByVal  channel As Long) As
        Long
W_8113_Channel_Clear () As Long
W_8113_ChannelNo_Get (no As Long) As Long

W_8113A_Channel_Select (ByVal  channel As Long) As
        Long
W_8113A_Channel_Deselect (ByVal  channel As Long)
        As Long
```

```
W_8113A_Channel_Clear () As Long
W_8113A_ChannelNo_Get (no As Long) As Long
```

@ Argument

channel : A/D channel number to select/deselect (0 ... 31)
no : number of channels currently selected

@ Return Code

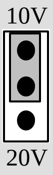
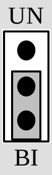
ERR_NoError
ERR_BoardNoInit
ERR_InvalidADChannel

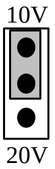
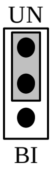
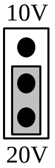
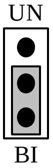
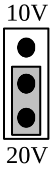
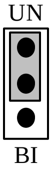
2.6.4 W_8113_Gain_Select / W_8113A_Gain_Select

@ Description

This function is used to set the A/D gain by means of writing data to the gain control register. It will effect the A/D input when different gain is set. The initial value of gain is '1' which is the default setting by the ACL-8113/8113A hardware.

The JP1 and JP2 are used to control A/D input modes - 10V or 20V, and Unipolar or Bipolar. The relationships between input range and gain code are listed below.

Range & Mode	JP1	JP2	Gain : Input Voltage Range
Input Range 10V & Bipolar Mode **(Default)**			X1 : -5V ~ +5V X2 : -2.5V ~ +2.5V X4 : -1.25V ~ +1.25V X8 : -0.625V ~ +0.625V

Input Range 10V & Unipolar Mode			X1 : 0V ~ 10V X2 : 0V ~ 5V X4 : 0V ~ 2.5V X8 : 0V ~ 1.25V
Input Range 20V & Bipolar Mode			X1 : -10V ~10V X2 : -5V ~ +5V X4 : -2.5V ~ +2.5V X8 : -1.25V ~ +1.25V
Input Range 20V & Unipolar Mode			X1 : Not Used X2 : 0V ~ 10V X4 : 0V ~ 5V X8 : 0V ~ 2.5V

Gain	Gain Code
X1	AD_GAIN_1
X2	AD_GAIN_2
X4	AD_GAIN_4
X8	AD_GAIN_8

@ Syntax

Microsoft C/C++

```
int W_8113_Gain_Select (int ad_gain)
int W_8113A_Gain_Select (int ad_gain)
```

Visual Basic

Windows 3.11 Version:

```
W_8113_Gain_Select (ByVal ad_gain As Integer) As
        Integer
```

W_8113A_Gain_Select (ByVal ad_gain As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8113_Gain_Select (ByVal ad_gain As Long) As Long
W_8113A_Gain_Select (ByVal ad_gain As Long) As Long

@ Argument

ad_gain : the programmable gain of A/D conversion, the possible values is AD_GAIN_1, AD_GAIN_2, AD_GAIN_4, and AD_GAIN_8.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADGain

2.6.5 W_8113_AD_Aquire / W_8113A_AD_Aquire

@ Description

This function is used to poll the A/D conversion data. It will trigger the A/D conversion, and read the 12 bits A/D data until the data is ready ('data ready' bit becomes to low).

@ Syntax

Microsoft C/C++

```
int W_8113_AD_Aquire (int *ad_data)
int W_8113A_AD_Aquire (int *ad_data)
```

Visual Basic

Windows 3.11 Version:

W_8113_AD_Aquire (ad_data As Integer) As Integer
W_8113A_AD_Aquire (ad_data As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8113_AD_Aquire (ad_data As Long) As Long
W_8113A_AD_Aquire (ad_data As Long) As Long

@ Argument

ad_data : 12 bits A/D converted value, the value should be within 0 and 4095.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_AD_AquireTimeOut

2.6.6 W_8113_MAD_Aquire / W_8113A_MAD_Aquire

@ Description

This function performs one A/D conversion on each of the selected channels, and puts the data in the array 'Data_8113'. If the channels in the selected list are 3, 8, 9, and 15, then the converted values for channel 3 will be stored in Data_8113[3], channel 8 in Data_8113[8], ..., etc. After using W_8113(A)_MAD_Aquire, you should use W_8113(A)_Get_MAD_Data function to get the converted data stored in array Data_8113.

@ Syntax

Microsoft C/C++

```
int W_8113_MAD_Aquire (void)
int W_8113A_MAD_Aquire (void)
```

Visual Basic

Windows 3.11 Version:

```
W_8113_MAD_Aquire () As Integer
W_8113A_MAD_Aquire () As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8113_MAD_Aquire () As Long
W_8113A_MAD_Aquire () As Long
```

@ Return Code

ERR_NoRrror

ERR_BoardNoInit

2.6.7 W_8113_Get_MAD_Data / W_8113A_Get_MAD_Data

@ Description

After using W_8113(A)_MAD_Aquire function to perform A/D conversion, this function is called to get the converted data stored in array 'Data_8113' (please refer to section 2.6.6 for the details).

@ Syntax

Microsoft C/C++

```
int W_8113_Get_MAD_Data(unsigned int *ad_data_array)
int W_8113A_Get_MAD_Data(unsigned int
        *ad_data_array)
```

Visual Basic

Windows 3.11 Version:

```
W_8113_Get_MAD_Data (ad_data_array As Integer) As
        Integer
W_8113A_Get_MAD_Data (ad_data_array As Integer) As
        Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8113_Get_MAD_Data(ad_data_array As Long) As
        Long
W_8113A_Get_MAD_Data(ad_data_array As Long) As
        Long
```

@ Return Code

ERR_NoRrror

2.7 8216 Software DLL Driver

In this section, the ACL-8216's software DLL drivers are described. The function names of Windows 3.11, Window 95/98, and Windows NT/2000 versions are the same. So, users do not need to learn the difference between them. The application's portability between these three systems can be very high.

2.7.1 W_8216_Initial

@ Description

An ACL-8216 card is initialized according to the card number and the corresponding base address. Each ACL-8216 multi-function data acquisition card has to be initialized by this function before calling other functions.

Note: In this library, if you want to operate DMA or interrupt operation, only two ACL-8216 cards can be initialized. The reason is only two DMA channels are supported in the card.

@ Syntax

Microsoft C/C++

```
int W_8216_Initial (int card_number, int base_addresss)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_Initial (ByVal card_number As Integer, ByVal
        base_address As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_Initial (ByVal card_number As Long, ByVal
        base_address As Long) As Long
```

@ Argument

card_number : The card number to be initialized. If all the ACL-8216 cards only perform software polling, eight cards can be initialized and the valid card numbers are CARD_1, CARD_2,

…, CARD_8. However, if the ACL-8216 cards are operated in *Windows NT* system and will perform *interrupt or DMA data transfer*, only two cards can be initialized and the card number must be CARD_1 or CARD_2.

base_address : the I/O port base address of the card.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError

2.7.2 W_8216_Switch_Card_No

@ Description

After initialized more than one ACL-8216 cards, this function is used to select which card is used currently.

@ Syntax

Microsoft C/C++

```
int W_8216_Switch_Card_No (int card_number)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_Switch_Card_No (ByVal card_number As
        Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_Switch_Card_No (ByVal card_number As Long)
        As Long
```

@ Argument

card_number : The card number of the card that is set to be active. If all the ACL-8216 cards only perform software polling, eight cards can be initialized and the valid card numbers are CARD_1, CARD_2, …, CARD_8. However, if the ACL-8216 cards are operated in *Windows NT* system and will perform *interrupt or DMA data transfer*, only two cards can be initialized and the card number must be CARD_1 or CARD_2.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber

2.7.3 W_8216_DI

@ Description

This function is used to read data from digital input port. There are 16 digital inputs on the ACL-8216. The bit 0 to bit 7 are defined as **low byte** and the bit 8 to bit 15 are defined as **high byte**.

@ Syntax

Microsoft C/C++

int W_8216_DI (int port_number, unsigned char *di_data)

Visual Basic

Windows 3.11 Version:

W_8216_DI (ByVal port_number As Integer, di_data As Byte) As Integer

Win-95/98, Win-NT/2000 Version:

W_8216_DI (ByVal port_number As Integer, di_data As Byte) As Long

@ Argument

port_number : To indicate which port is read, DI_LOW_BYTE or DI_HIGH_BYTE.
DI_LOW_BYTE : bit 0 ~ bit 7
DI_HIGH_BYTE : bit8 ~ bit15

di_data : return value from digital port.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.7.4 W_8216_DI _Channel

@ Description

This function is used to read data from digital input channels (bit). There are 16 digital input channels on the ACL-8216. When performs this function, the digital input port is read and the value of the corresponding channel is returned.

* channel means each bit of digital input ports.

@ Syntax

Microsoft C/C++

```
int _8216_DI_Channel (int di_ch_no, unsigned int
        *di_data)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_DI_Channel (ByVal di_ch_no As Integer, di_data
        As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_DI_Channel (ByVal di_ch_no As Long, di_data
        As Long) As Long
```

@ Argument

di_ch_no : the DI channel number, the value has to be set between 0 and 15.
di_data : return value, either 0 or 1.

@ Return Code

ERR_NoError

ERR_BoardNoInit
ERR_InvalidDIChannel

2.7.5 W_8216_DO

@ Description

This function is used to write data to digital output ports. There are 16 digital outputs on the ACL-8216, they are divided to two ports, DO_LOW_BYTE and DO_HIGH_BYTE. The channel 0 to channel 7 are defined in DO_LOW_BYTE port and the channel 8 to channel 15 are defined as the DO_HIGH_BYTE port.

@ Syntax

Microsoft C/C++

```
int W_8216_DO (int port_number, unsigned char do_data)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_DO (ByVal port_number As Integer, ByVal
        do_data As Byte) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_DO (ByVal port_number As Long, ByVal do_data
        As Byte) As Long
```

@ Argument

port_number :	DO_LOW_BYTE or DO_HIGH_BYTE
do_data :	value will be written to digital output port

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.7.6 W_8216_DA

@ Description

This function is used to write data to D/A converters. There are two Digital-to-Analog conversion channels on the ACL-8216. The resolution of each channel is 12-bit, i.e. the range is from 0 to 4095.

@ Syntax

Microsoft C/C++

```
int W_8216_DA (int da_ch_no, unsigned int da_data)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_DA (ByVal da_ch_no As Integer, ByVal da_data
          As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_DA (ByVal da_ch_no As Long, ByVal da_data As
          Long) As Long
```

@ Argument

da_ch_no : D/A channel number, the valid data is :

0	Channel AO1
1	Channel AO2

da_data : D/A converted value, if the value is greater than 4095, the higher 4 bits are negligent.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDAChannel

2.7.7 W_8216_AD_Input_Mode

@ Description

This function is used to set A/D input mode to single-ended or differential mode. The default mode of A/D input is single-ended, so the A/D channel number can be set between 0 to 15. If the A/D mode is set as differential, the input channel can be selected from channel 0 to 7 only. This function is only available for ACL-8216HG and 8216DG, but not for ACL-8216PG.

@ Syntax

Microsoft C/C++

```
int W_8216_AD_Input_Mode (int mode)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_AD_Input_Mode (ByVal mode As Integer) As
          Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_AD_Input_Mode (ByVal mode As Long) As Long
```

@ Argument

mode :	SIGNLE_ENDED : singled-ended mode is set
	DIFFERENTIAL : differential mode is set

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADMode

2.7.8 W_8216_AD_Set_Channel

@ Description

This function is used to set A/D channel by means of writing data to the A/D channel multiplexer register. There are 16 single-ended A/D channels in ACL-8216, so the channel number should be set between 0 and 15 only. The initial state is channel 0 which is the default setting by the ACL-8216 hardware configuration.

@ Syntax

Microsoft C/C++

```
int W_8216_AD_Set_Channel (int ad_ch_no)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_AD_Set_Channel (ByVal ad_ch_no As Integer)
        As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_AD_Set_Channel (ByVal ad_ch_no As Long) As
        Long
```

@ Argument

ad_ch_no : channel number to perform A/D conversion

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADChannel

2.7.9 W_8216_AD_Set_Range

@ Description

This function is used to set the A/D gain by means of writing data to the range control register. The gain values and their corresponding A/D input ranges are listed as below.

The initial value of gain is '1' which is which is the default setting by the ACL-8216 hardware configuration.

Input Range (V)	Gain	Gain Code
±10 V	X 1	AD_GAIN_1
±5 V	X 2	AD_GAIN_2
±2.5 V	X 4	AD_GAIN_4
±1.25 V	X 8	AD_GAIN_8

@ Syntax

Microsoft C/C++

int W_8216_AD_Set_Range (int ad_gain)

Visual Basic

Windows 3.11 Version:

W_8216_AD_Set_Range (ByVal ad_gain As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8216_AD_Set_Range (ByVal ad_gain As Long) As Long

@ Argument

ad_gain : the programmable gain of A/D conversion, the possible value is AD_GAIN_1, AD_GAIN_2, AD_GAIN_4, and AD_GAIN_8.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADGain

2.7.10 W_8216_AD_Set_Mode

@ Description

This function is used to set the A/D trigger and data transfer mode by means of writing data to the mode control register.

The hardware initial state of the ACL-8216 is set as A8216_AD_MODE_1 software (internal) trigger with program polling.

A/D Mode	Description
A8216_AD_MODE_0	External Trigger, Software Polling
A8216_AD_MODE_1	Software Trigger, Software Polling
A8216_AD_MODE_2	Timer Trigger, DMA Transfer
A8216_AD_MODE_3	External Trigger, DMA Transfer
A8216_AD_MODE_4	External Trigger, Interrupt Transfer
A8216_AD_MODE_5	Software Trigger, Interrupt Transfer
A8216_AD_MODE_6	Timer Trigger, Interrupt Transfer
A8216_AD_MODE_7	Not Used

@ Syntax

Microsoft C/C++

int W_8216_AD_Set_Mode (int ad_mode)

Visual Basic

Windows 3.11 Version:

W_8216_AD_Set_Mode (ByVal ad_mode As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8216_AD_Set_Mode (ByVal ad_mode As Long) As Long

@ Argument

ad_mode : A/D trigger and data transfer mode

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidMode

2.7.11 W_8216_AD_Soft_Trig

@ Description

This function is used to trigger the A/D conversion by software. When the function is called, a trigger pulse will be generated and the converted data will be stored in the address Base+4 and Base+5, and can be retrieved by function W_8216_AD_Aquire().

@ Syntax

Microsoft C/C++

```
int W_8216_AD_Soft_Trig (void)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_AD_Soft_Trig ( ) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_AD_Soft_Trig ( ) As Long
```

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.7.12 W_8216_AD_Aquire

@ Description

This function is used to poll the A/D conversion data. It will trigger the A/D conversion, and read the 16-bit A/D data until the data is ready ('data ready' bit becomes to low).

@ Syntax

Microsoft C/C++

```
int W_8216_AD_Aquire (int *ad_data)
```

Visual Basic

Windows 3.11 Version:

W_8216_AD_Aquire (ad_data As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8216_AD_Aquire (ad_data As Long) As Long

@ Argument

ad_data : 16-bit A/D converted value. The value is within -32768 and 32767. -32768 and 32767 correspond to the lowest and highest voltage respectively. For example, if the A/D range is bipolar ±10V, -32768 represents -10V and 32767 represents +10V.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_AD_AquireTimeOut

2.7.13 W_8216_CLR_IRQ

@ Description

This function is used to clear interrupt request which is requested by the ACL-8216. If you use interrupt to transfer A/D converted data, you should use this function to clear interrupt request status, otherwise the new interrupt signal can not be generated.

@ Syntax

Microsoft C/C++

int W_8216_CLR_IRQ (void)

Visual Basic

Windows 3.11 Version:

W_8216_CLR_IRQ () As Integer

Win-95/98, Win-NT/2000 Version:

W_8216_CLR_IRQ () As Long

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.7.14 W_8216_AD_DMA_Start

@ Description

The function will perform A/D conversion N times with DMA data transfer by using the pacer trigger (internal timer trigger). It will take place in the background which will not be stop until the N-th conversion has been completed or your program execute W_8216_AD_DMA_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function W_8216_AD_DMA_Status(). The function is performed on single A/D channel with fixed gain.

Note: W_8216_AD_DMA_Start() and W_8216_AD_DMA_Stop() are a pair function, i.e., you have to call W_8216_AD_DMA_Stop() after W_8216_AD_DMA_Start(), otherwise the A/D converted data will not be stored in the buffer you had specified.

@ Syntax

Microsoft C/C++

```
int W_8216_DMA_Start (int ad_ch_no, int ad_gain,
        int dma_ch_no, int irq_ch_no, int count , short
        *ad_buffer, unsigned int c1, unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_DMA_Start (ByVal ad_ch_no As Integer, ByVal
        ad_gain As Integer, ByVal dma_ch_no As Integer,
        ByVal irq_ch_no As Integer, ByVal count As
```

Integer, ad_buffer As Integer, ByVal c1 As Integer, ByVal c2 As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8216_DMA_Start (ByVal ad_ch_no As Long, ByVal ad_gain As Long, ByVal dma_ch_no As Long, ByVal irq_ch_no As Long, ByVal count As Long, ad_buffer As Integer, ByVal c1 As Long, ByVal c2 As Long) As Long

@ Argument

ad_ch_no :	A/D channel number
ad_gain :	A/D gain value. The possible values are AD_GAIN_1, AD_GAIN_2, AD_GAIN_4, or AD_GAIN_8.
dma_ch_no :	DMA channel number, DMA_CH_1 or DMA_CH_3
irq_ch_no :	IRQ channel number, used to stop DMA
count :	the number of A/D conversion
ad_buffer :	the start address of the memory buffer to store the A/D data, the buffer size must large than the number of A/D conversion. Each data element of ad_buffer contains 16-bit A/D transfer data.
c1 :	the 16-bit timer frequency divider of timer channel #1
c2 :	the 16-bit timer frequency divider of timer channel #2

Note： While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidDMAChannel,
ERR_InvalidIRQChannel, ERR_InvalidTimerValue

2.7.15 W_8216_AD_ContDMA_Start

@ Description

The function will perform continuous A/D conversions with DMA data transfer by using the pacer trigger (internal timer trigger) or external trigger source.

It will take place in the background and will not be stopped until your program executes W_8216_AD_DMA_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function W_8216_AD_ DblBufferHalfReady().The function performs on single A/D channel with fixed A/D range.

Note: W_8216_AD_ContDMA_Start() and W_8216_AD_DMA_Stop() are pair function, i.e., you have to call W_8216_AD_DMA_Stop() after W_8216_AD_ContDMA_Start(), otherwise the A/D conversion will never stop .

@ Syntax

Microsoft C/C++

```
int W_8216_ContDMA_Start (int ad_ch_no, int ad_gain,
          int dma_ch_no, int irq_ch_no, int count , short
          *ad_buffer, unsigned int c1, unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_ContDMA_Start (ByVal ad_ch_no As Integer,
          ByVal  ad_gain As Integer, ByVal dma_ch_no As
          Integer, ByVal irq_ch_no As Integer, ByVal count
          As Integer, ad_buffer As Integer,  ByVal c1 As
          Integer, ByVal c2 As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_ContDMA_Start (ByVal ad_ch_no As Long,
        ByVal  ad_gain As Long, ByVal dma_ch_no As
        Long, ByVal irq_ch_no As Long, ByVal count As
        Long, ad_buffer As Integer,  ByVal c1 As Long,
        ByVal c2 As Long) As Long
```

@ Argument

ad_ch_no : A/D channel number

ad_gain : A/D gain value. The possible values are AD_GAIN_1, AD_GAIN_2, AD_GAIN_4, or AD_GAIN_8.

dma_ch_no : DMA channel number, DMA_CH_1 or DMA_CH_3

irq_ch_no : IRQ channel number, used to stop DMA

count : the number of A/D conversion

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must large than the number of A/D conversion. Each data element of ad_buffer contains 16-bit A/D transfer data.

c1 : the 16-bit timer frequency divider of timer channel #1

c2 : the 16-bit timer frequency divider of timer channel #2

Note： While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError
ERR_AD_DMANotSet
ERR_BoardNoInit, ERR_InvalidADChannel,

ERR_InvalidADGain, ERR_InvalidDMAChannel, ERR_InvalidIRQChannel, ERR_InvalidTimerValue

2.7.16 W_8216_AD_DMA_Status

@ Description

Since the W_8216_AD_DMA_Start function executes in the background, you can issue the function W_8216_AD_DMA_Status() to check its operation status.

@ Syntax

Microsoft C/C++

int W_8216_AD_DMA_Status (int *status , int *count)

Visual Basic

Windows 3.11 Version:

W_8216_AD_DMA_Status (status As Integer, count As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8216_AD_DMA_Status (status As Long, count As Long) As Long

@ Argument

status :	status of the DMA data transfer AD_DMA_STOP : A/D DMA is completed AD_DMA_RUN : A/D DMA is not completed
count :	the number of A/D data which has been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ADDMANotSet

2.7.17 W_8216_AD_DMA_Stop

@ Description

This function is used to stop the DMA data transfer. After executing this function, the internal A/D trigger is disabled and the A/D timer (timer #1 and #2) is stopped. The function returns the number of the data which has been transferred, no matter the A/D DMA data transfer is stopped by this function or by the DMA terminal count ISR.

This function has to be called after W_8216_AD_DMA_Start() function issued. Otherwise, all converted data will not be saved into the memory buffer you specified in your program.

@ Syntax

Microsoft C/C++

int W_8216_AD_DMA_Stop (int *count)

Visual Basic

Windows 3.11 Version:

W_8216_AD_DMA_Stop (count As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8216_AD_DMA_Stop (count As Long) As Long

@ Argument

count :	the number of A/D converted data which has been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ADDMANotSet

2.7.18 W_8216_AD_INT_Start

@ Description

The function will perform A/D conversion N times with interrupt data transfer by using pacer trigger. It will take place in the background which will not be stopped until the N-th conversion has been completed or your program execute W_8216_AD_INT_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function W_8216_AD_INT_Status(). The function is perform on single A/D channel with fixed gain.

Note: W_8216_AD_INT_Start(), and W_8216_AD_INT_Stop() are a pair of functions, i.e., you have to call W_8216_AD_INT_Stop() after W_8216_AD_INT_Start(), otherwise the A/D converted data will not be stored in the buffer you had specified.

@ Syntax

Microsoft C/C++

```
int W_8216_INT_Start (int ad_ch_no, int ad_gain, int
          irq_ch_no, int count , short *ad_buffer, unsigned int
          c1, unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_INT_Start (ByVal ad_ch_no As Integer, ByVal
          ad_gain As Integer, ByVal irq_ch_no As Integer,
          ByVal count As Integer, ad_buffer As Integer,
          ByVal c1 As Integer, ByVal c2 As Integer) As
          Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_INT_Start (ByVal ad_ch_no As Long, ByVal
          ad_gain As Long, ByVal irq_ch_no As Long, ByVal
          count As Long, ad_buffer As Integer,  ByVal c1 As
          Long, ByVal c2 As Long) As Long
```

@ Argument

ad_ch_no : A/D channel number

ad_gain : A/D gain value. The possible values are AD_GAIN_1, AD_GAIN_2, AD_GAIN_4, or AD_GAIN_8.

irq_ch_no : IRQ channel number used to transfer A/D data, the possible value is defined in file Dll2.h.

count : number of A/D conversions to perform

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must be large than the number of A/D conversions. Each data element of ad_buffer contains 16-bit A/D transfer data.

c1 : the 16-bit timer frequency divider of timer channel #1

c2 : the 16-bit timer frequency divider of timer channel #2

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidIRQChannel,
ERR_InvalidTimerValue

@ Example

Visual Basic (Win-95/98, Win-NT/2000 Version)

```
Dim ad_buf(1024) As Integer
Dim Channel As Long, Gain As Long, Irq As Long
Dim ad_count As Long, c1 As Long, c2 As Long
Dim Ret As Long
        .
        .
ad_count = 1024
        .
        .
Ret = W_8316_AD_INT_Start(Channel, Gain, Irq, ad_count,
        ad_buf(0), c1, c2)
```

2.7.19 W_8216_AD_INT_Status

@ Description

Since the W_8216_AD_INT_Start() function executes in background, you can issue the function W_8216_AD_INT_Status() to check the status of interrupt operation.

@ Syntax

Microsoft C/C++

```
int W_8216_AD_INT_Status (int *status , int *count)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_AD_INT_Status (status As Integer, count As
        Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_AD_INT_Status (status As Long, count As Long)
        As Long
```

@ Argument

status :	status of the interrupt data transfer AD_INT_STOP : interrupt A/D is completed AD_INT_RUN : interrupt A/D is not completed
count :	the number of A/D data which has been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.7.20 W_8216_AD_INT_Stop

@ Description

This function is used to stop the interrupt data transfer function. After executing this function, the internal A/D trigger is disabled and the A/D timer is stopped. The function returns the number of the data which has been transferred, no matter whether if the A/D interrupt data transfer is stopped by this function or by the W_8216_AD_INT_Start() itself.

This function has to be called after W_8216_AD_INT_Start() function issued. Otherwise, all converted data will not be saved into the memory buffer you had specified in W_8216_AD_INT_Start() function call.

@ Syntax

Microsoft C/C++

```
int W_8216_AD_INT_Stop (int *count)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_AD_INT_Stop (count As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_AD_INT_Stop (count As Long) As Long
```

@ Argument

count :	the number of A/D data which have been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.7.21 W_8216_AD_ContINT_Start

@ Description

The function will perform continuous A/D conversions with interrupt data transfer by using timer pacer (internal clock trigger). It will take place in the background which will not be stopped until your program execute W_8216_AD_INT_Stop() function to stop the process. After calling this function, it is necessary to check the status of the operation by using the function W_8216_AD_DblBufferHalfReady().

Note: W_8216_AD_ContINT_Start(), and W_8216_AD_INT_Stop() are a pair of functions, i.e., you have to call W_8216_AD_INT_Stop() after W_8216_AD_ContINT_Start(), otherwise the A/D converted data will not be stored in the buffer you had specified.

@ Syntax

Microsoft C/C++

```
int W_8216_ContINT_Start (int ad_ch_no, Boolean
          autoscan, int ad_gain, int irq_ch_no, int count ,
          short *ad_buffer, unsigned int c1, unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_ContINT_Start (ByVal ad_ch_no As Integer,
          ByVal auto_scan As Integer, ByVal ad_gain As
          Integer, ByVal irq_ch_no As Integer, ByVal count
          As Integer, ad_buffer As Integer, ByVal c1 As
          Integer, ByVal c2 As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_ContINT_Start (ByVal ad_ch_no As Long, ByVal
          auto_scan As Integer, ByVal ad_gain As Long,
          ByVal irq_ch_no As Long, ByVal count As Long,
          ad_buffer As Integer,  ByVal c1 As Long, ByVal c2
          As Long) As Long
```

@ Argument

ad_ch_no : A/D channel number

If autoscan is enabled, the A/D channel scan sequence will be: 0, 1, 2, 3,…[ad_ch_no], 0, 1, …, [ad_ch_no], …
If autoscan is disabled, only the data from channel [ad_ch_no] will be converted.

autoscan: FALSE: autoscan is disabled
TRUE: autoscan is enabled

ad_gain : A/D gain value. The possible values are AD_GAIN_1, AD_GAIN_2, AD_GAIN_4, or AD_GAIN_8.

irq_ch_no : IRQ channel number used to transfer A/D data, the possible value is defined in file Dll2.h.

count : number of A/D conversions to perform

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must be large than the number of A/D conversions. Each data element of ad_buffer contains 16-bit A/D transfer data.

c1 : the 16-bit timer frequency divider of timer channel #1

c2 : the 16-bit timer frequency divider of timer channel #2

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError
ERR_BoardNoInit, ERR_InvalidADChannel,

ERR_InvalidADGain, ERR_InvalidIRQChannel,
ERR_InvalidTimerValue
ERR_AD_INTNotSet

2.7.22 W_8216_AD_SCANINT_Start

@ Description

This function is used to start automatic channel scan . If autoscan mode is started and the end channel number is set as n by argument **ad_ch_no**, the data will be converted automatically from channel 0 to channel n.
For example, the channel is set as 4 and autoscan is started, the A/D conversion sequence will be 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, ...
If the autoscan is finished, the converted channel will be kept at the specified channel, i.e. channel 4.

@ Syntax

Microsoft C/C++

```
int W_8216_AD_SCANINT_Start( int ad_ch_no, int
    ad_gain , int irq_no, int count , unsigned short
    *ad_buffer , unsigned int c1 , unsigned int c2)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_AD_ SCANINT_Start (ByVal ad_ch_no As
        Integer, ByVal ad_gain As Integer, ByVal irq_ch_no
        As Integer, ByVal count As Integer, ad_buffer As
        Integer, ByVal c1 As Integer, ByVal c2 As Integer)
        As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_AD_ SCANINT_Start (ByVal ad_ch_no As Long,
        ByVal ad_gain As Long, ByVal irq_ch_no As Long,
        ByVal count As Long, ad_buffer As Integer, ByVal
        c1 As Long, ByVal c2 As Long) As Long
```

@ Argument

ad_ch_no : end A/D channel number for AutoScan

ad_gain : A/D gain value. The possible values are AD_GAIN_1, AD_GAIN_2, AD_GAIN_4, or AD_GAIN_8.

irq_ch_no : IRQ channel number used to transfer A/D data, the possible value is defined in file DLL2.H

count : the number of A/D conversion

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must be large than the number of A/D conversion.

c1 : the 16-bit timer frequency divider of timer channel #1

c2 : the 16-bit timer frequency divider of timer channel #2

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADChannel
ERR_AD_InvalidGain
ERR_InvalidIRQChannel
ERR_InvalidTimerValue

2.7.23 W_8216_AD_DblBufferHalfReady

@ Description

Checks whether the next half buffer of data in circular buffer is ready for transfer during an double-buffered analog input operation.

@ Syntax

Microsoft C/C++

```
int W_8216_AD_DblBufferHalfReady ( BOOLEAN
        *bHalfReady)
```

Visual Basic

W_8216_AD_DblBufferHalfReady (bHalfReady As Long)
As Long

@ Argument

bHalfReady : Whether the next half buffer of data is available.If *HalfReady* = TRUE, you can call **`W_8216_AD_DblBufferTransfer()`** to copy the data to your user buffer.

@ Return Code

ERR_NoError
ERR_InvalidMode

2.7.24 W_8216_AD_DblBufferTransfer

@ Description

Depending on the continuous AI function elected, half of the data in circular buffer will be logged into the user buffer . You can execute this function repeatedly to return sequential half buffers of the data.

@ Syntax

Microsoft C/C++

int W_8216_AD_DblBufferTransfer (USHORT *pwBuffer)

Visual Basic

W_8216_AD_DblBufferTransfer (pwBuffer As Integer) As
Long

@ Argument

pwBuffer: The user buffer. An integer array to which the data is to be copied.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.7.25 W_8216_AD_Timer

@ Description

This function is used to set up the Timer #1 and Timer #2. The values of c1 and c2 are used as frequency dividers for generating constant A/D sampling rate dedicatedly. It is possible to stop the pacer trigger by setting any one of the dividers as 0. Because the A/D conversion rate is limited due to the conversion time of the A/D converter, the highest sampling rate of the ACL-8216 can not exceed 100 KHz. The multiplication of the dividers must be larger than 20.

@ Syntax

Microsoft C/C++

```
int W_8216_AD_Timer( unsigned int c1, unsigned int c2 )
```

Visual Basic

Windows 3.11 Version:

```
W_8216_AD_Timer (ByVal c1 As Integer, ByVal c2 As
        Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_AD_Timer (ByVal c1 As Long, ByVal c2 As Long)
        As Long
```

@ Argument

c1 :	frequency divider of timer #1
c2 :	frequency divider of timer #2

Note : the A/D sampling rate is equal to : 2MHz / (c1*c2), when c1 = 0 or c2 = 0, the pacer trigger will be stopped.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerValue

2.7.26 W_8216_Timer_Start

@ Description

The Timer #0 on the ACL-8216 can be freely programmed by the users. This function is used to program the Timer #0. This timer can be used as frequency generator if internal clock is used. It also can be used as event counter if external clock is used.

@ Syntax

Microsoft C/C++

```
int W_8216_Timer_Start (int timer_mode, unsigned int c0)
```

Visual Basic

Windows 3.11 Version:

```
W_8216_Timer_Start (ByVal timer_mode As Integer,
        ByVal c0 As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8216_Timer_Start (ByVal timer_mode As Long, ByVal
        c0 As Long) As Long
```

@ Argument

timer_mode : the 8253 timer mode, the possible values are : TIMER_MODE0, TIMER_MODE1, TIMER_MODE2, TIMER_MODE3, TIMER_MODE4, TIMER_MODE5.

c0 : the counter value of timer

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerMode

2.7.27 W_8216_Timer_Read

@ Description

This function is used to read the counter value of the Timer #0.

@ Syntax

Microsoft C/C++

int W_8216_Timer_Read (unsigned int *counter_value)

Visual Basic

Windows 3.11 Version:

W_8216_Timer_Read (counter_value As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8216_Timer_Read (counter_value As Long) As Long

@ Argument

counter_value : the counter value of the Timer #0

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.7.28 W_8216_Timer_Stop

@ Description

This function is used to stop the timer operation. The timer is set to the 'One-shot' mode with counter value '0'. That is, the clock output signal will be set to high after executing this function.

@ Syntax

Microsoft C/C++

int W_8216_Timer_Stop (unsigned int *counter_value)

Visual Basic

Windows 3.11 Version:

W_8216_Timer_Stop (counter_value As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8216_Timer_Stop (counter_value As Long) As Long

@ Argument

counter_value : the current counter value of the Timer #0

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.7.29 W_8216_DMA_InitialMemoryAllocated

@ Description

This function is only available in Windows NT and Windows 2000 system. This function returns the available memory size for DMA data transfer in the device driver in argument ***MemSize***. While performming analog input with DMA data transfer, the analog input size can not exceed this size.

@ Syntax

Microsoft C/C++

W_8216_DMA_InitialMemoryAllocated(int *MemSize)

Visual Basic

Win-NT/2000 Version:

W_8216_DMA_InitialMemoryAllocated(MemSize As Long)
As Long

@ Argument

MemSize : the available memory size for DMA data transfer in device driver of ACL-8216.

@ Return Code

ERR_NoError
ERR_BoardNoInit

ERR_INTNotSet

2.8 8316/12 Software DLL Driver

In this section, the ACL-8316/12's software DLL drivers are described. The function names of Windows 3.11, Window 95/98, and Windows NT/2000 versions are the same. So, users do not need to learn the difference between them. The application's portability between these three systems can be very high.

2.8.1 W_8316_Initial

@ Description

An ACL-8316/12 card is initialized according to the card number and the corresponding base address. Each ACL-8316/12 multi-function data acquisition card has to be initialized by this function before calling other functions.

Note: In this library, if you want to operate DMA or interrupt operation, only two ACL-8316/12 cards can be initialized. The reason is only three DMA channels are supported in the card.

@ Syntax

Microsoft C/C++

int W_8316_Initial (int card_number, int base_addresss)

Visual Basic

Windows 3.11 Version:

W_8316_Initial (ByVal card_number As Integer, ByVal
base_address As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_Initial (ByVal card_number As Long, ByVal
base_address As Long) As Long

@ Argument

card_number : The card number to be initialized. If all the ACL-8316/12 cards only perform software polling, eight cards can be initialized and the valid card numbers are CARD_1, CARD_2, …, CARD_8. However, if the ACL-8316/12 cards are operated in *Windows NT* system and will perform *interrupt or DMA data transfer*, only three cards can be initialized and the card number must be CARD_1, CARD_2 or CARD_3.

base_address : the I/O port base address of the card.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError

2.8.2 W_8316_Switch_Card_No

@ Description

This function is used on multi-cards system. After initialized more than one ACL-8316/12 cards, this function is used to select which card is used currently.

@ Syntax

Microsoft C/C++

int W_8316_Switch_Card_No (int card_number)

Visual Basic

Windows 3.11 Version:

W_8316_Switch_Card_No (ByVal card_number As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_Switch_Card_No (ByVal card_number As Long) As Long

@ Argument

card_number : The card number of the card that is set to be active. If all the ACL-8316/12 cards only perform software polling, eight cards can be initialized and the valid card numbers are CARD_1, CARD_2, …, CARD_8. However, if the ACL-8316/12 cards are operated in *Windows NT* system and will perform *interrupt or DMA data transfer*, the card number must be CARD_1, CARD_2 or CARD_3.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber

2.8.3 W_8316_DI

@ Description

This function is used to read data from digital input port. There are 16 digital inputs on the ACL-8316/12.

@ Syntax

Microsoft C/C++

```
int W_8316_DI (U16 *di_data)
```

Visual Basic

Windows 3.11 Version:

```
W_8316_DI (di_data As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8316_DI (di_data As Integer) As Long
```

@ Argument

di_data : return value from digital port.

@ Return Code

ERR_NoError
ERR_BoardNoInit

ERR_PortError

2.7.4 W_8316_DI _Channel

@ Description

This function is used to read data from digital input channels (bit). There are 16 digital input channels on the ACL-8316/12. When performs this function, the digital input port is read and the value of the corresponding channel is returned.

* channel means each bit of digital input ports.

@ Syntax

Microsoft C/C++

```
int _8316_DI_Channel (U8 di_ch_no, Boolean *di_data)
```

Visual Basic

Windows 3.11 Version:

```
W_8316_DI_Channel (ByVal di_ch_no As Byte, di_data
          As Byte) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8316_DI_Channel (ByVal di_ch_no As Byte, di_data
          As Byte) As Long
```

@ Argument

di_ch_no :	the DI channel number, the value has to be set between 0 and 15.
di_data :	return value, either 0 or 1.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDIChannel

2.8.5 W_8316_DO

@ Description

This function is used to write data to digital output ports. There are 16 digital outputs on the ACL-8316/12.

@ Syntax

Microsoft C/C++

int W_8316_DO (U16 do_data)

Visual Basic

Windows 3.11 Version:

W_8316_DO (ByVal do_data As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_DO (ByVal do_data As Long) As Long

@ Argument

do_data : value will be written to digital output port

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.8.6 W_8316_DA_Set_Mode

@ Description

This function is used to configure D/A output mode. There are four output modes can be set for ACL-8316/12. They are:

A_8316_DA_MODE_0: Transparency and Binary data format
A_8316_DA_MODE_1: Transparency and Two's complement data format
A_8316_DA_MODE_2: Double buffered and Binary data format

A_8316_DA_MODE_3: Double buffered and Two's
complement data format

The data format of binary and two's complement for ACL-8316/12 are shown in the following table:

Digital Input Binary Format	Digital Input 2's complement	Analog Output	
		Unipolar 0 to 10V	Bipolar -10V to 10V
FFF hex	7FFhex	+9.9976V	+9.9951V
800 hex	000 hex	+5.0000V	0.0000V
7FF hex	FFF hex	+4.9976V	-0.0049V
000 hex	800 hex	0.0000V	-10.0000V
1LSB	1 LSB	2.44mV	4.88mV

@ Syntax

Microsoft C/C++

int W_8316_DA_Set_Mode (U8 ad_mode)

Visual Basic

Windows 3.11 Version:

W_8316_DA_Set_Mode (ByVal da_mode As Byte) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_DA_Set_Mode (ByVal da_mode As Byte) As Integer

@ Argument

da_mode : D/A output mode. The valid code is: DA_MODE_0, DA_MODE_1, DA_MODE_2 and DA_MODE_3

@ Return Code

ERR_NoError

ERR_BoardNoInit

2.8.7 W_8316_DA

@ Description

This function is used to write data to D/A converters. There are two Digital-to-Analog conversion channels on the ACL-8316/12. The resolution of each channel is 12-bit. The data format can be binary or two's complement format and is defined by using function W_8316_DA_Set_Mode (refer to section 2.8.6 for the details).

@ Syntax

Microsoft C/C++

int W_8316_DA (int da_ch_no, unsigned int da_data)

Visual Basic

Windows 3.11 Version:

W_8316_DA (ByVal da_ch_no As Integer, ByVal da_data As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_DA (ByVal da_ch_no As Long, ByVal da_data As Long) As Long

@ Argument

da_ch_no : D/A channel number, the valid data is :

0	Channel AO1
1	Channel AO2

da_data : D/A converted value. The data format of binary and two's complement for ACL-8316/12 are shown in the following table:

Digital Input	Digital Input	Analog Output

Binary Format	**2‘s complement**	Unipolar 0 to 10V	Bipolar -10V to 10V
FFF hex	7Ffhex	+9.9976V	+9.9951V
800 hex	000 hex	+5.0000V	0.0000V
7FF hex	FFF hex	+4.9976V	-0.0049V
000 hex	800 hex	0.0000V	-10.0000V
1LSB	1 LSB	2.44mV	4.88mV

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDAChannel

2.8.8 W_8316_AD_Set_Mode

@ Description

This function is used to set A/D trigger source, data transfer mode and A/D channel autoscan enabled/disabled by writing data into *AD Mode Control Register* (refer to section 4.7 of ACL-8316/12 user’s manual for the details). The hardware initial state of ACL_8316/12 is set as internal software trigger with program polling data transfer.

@ Syntax

Microsoft C/C++

int W_8316_AD_Set_Mode (U8 mode)

Visual Basic

Windows 3.11 Version:

W_8316_AD_Set_Mode (ByVal mode As Byte) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_AD_Set_Mode (ByVal mode As Byte) As Integer

@ Argument

mode : AD mode control value. This argument is an integer expression formed from one or more of the manifest constants defined DLL2.H. The valid constants are:

A8316_AD_DMA: DMA data transfer enabled
A8316_AD_EXT_SRC: External A/D trigger source
A8316_AD_INT_SRC: Internal A/D trigger source
A8316_AD_TimerTrig: Internal timer pacer trigger source
A8316_AD_SoftTrig: Software trigger
A8316_AD_AutoScan: Channel autoscan enabled

When two or more constants are used to form mode argument, these constants are combined with plus(+) operator.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADMode

2.8.9 W_8316_AD_Set_Channel

@ Description

This function is used to set A/D channel by means of writing data to the A/D channel multiplexer register. There are 16 single-ended A/D channels or 8 differential A/D channels in ACL-8316/12. The initial state is channel 0 which is the default setting by the ACL-8316/12 hardware configuration.

@ Syntax

Microsoft C/C++

```
int W_8316_AD_Set_Channel (U8 ad_ch_no)
```

Visual Basic

Windows 3.11 Version:

W_8316_AD_Set_Channel (ByVal ad_ch_no As Byte) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_AD_Set_Channel (ByVal ad_ch_no As Byte) As Integer

@ Argument

ad_ch_no : channel number to perform A/D conversion.
Signal-Ended mode: 0 ~15
Differential mode: 0 ~ 7

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADChannel

2.8.10 W_8316_AD_Set_Range

@ Description

This function is used to set the A/D gain by means of writing data to the range control register. The valid range codes and their corresponding A/D input ranges are listed as below.

The initial value of gain is '1' which is which is the default setting by the ACL-8316/12 hardware configuration.

Range Code	Input Range (V)
AD_B_10_V	±10 V
AD_B_5_V	±5 V
AD_B_2_5_V	±2.5 V
AD_B_1_25_V	±1.25 V
AD_U_10_V	0 ~ 10 V
AD_U_5_V	0 ~ 5 V
AD_U_2_5_V	0 ~ 2.5 V
AD_U_1_25_V	0 ~ 1.25 V

@ Syntax

Microsoft C/C++

int W_8316_AD_Set_Range (U8 range)

Visual Basic

Windows 3.11 Version:

W_8316_AD_Set_Range (ByVal range As Byte) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_AD_Set_Range (ByVal range As Byte) As Integer

@ Argument

range : the programmable range of A/D conversion, the possible values are as:
AD_B_10_V, AD_B_5_V, AD_B_2_5_V, AD_B_1_25_V, AD_U_10_V, AD_U_5_V, AD_U_2_5_V, AD_U_1_25_V.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidADGain

2.8.11 W_8316_AD_Set_Autoscan

@ Description

This function is used to set automatic hardware channel scan to be enabled or disabled. If autoscan mode is enabled and the end channel number is set as n by function W_8316_AD_Set_Channel, the data will be converted automatically from channel 0 to channel n. If autoscan mode is disabled and the channel number is set as n by function W_8316_AD_Set_Channel, the data at channel n will be converted.

For example, the channel is set as 4 and autoscan is enabled, the A/D conversion sequence will be 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, ...
If the autoscan is disabled, the converted channel will be kept at the specified channel, i.e. channel 4.

@ Syntax

Microsoft C/C++

```
int W_8316_AD_Set_AutoScan (Boolean flag)
```

Visual Basic

Windows 3.11 Version:

```
W_8316_AD_Set_AutoScan (ByVal flag As Byte) As
        Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8316_AD_Set_AutoScan (ByVal flag As Byte) As
        Integer
```

@ Argument

flag : 1: autoscan enabled
0: autoscan is disabled

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.8.12 W_8316_AD_Set_FIFO

@ Description

This function is used to enable the FIFO on ACL-8316/12 board. As the FIFO is enabled, the A/D converted data are stored into the FIFO. The size of A/D FIFO on board is 1K words.

@ Syntax

Microsoft C/C++

int W_8316_AD_Set_FIFO (Boolean flag)

Visual Basic

Windows 3.11 Version:

W_8316_AD_Set_FIFO (ByVal flag As Byte) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_AD_Set_FIFO (ByVal flag As Byte) As Integer

@ Argument

flag : 1: autoscan enabled
0: autoscan is disabled

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.8.13 W_8316_AD_Set_INT_Source

@ Description

This function is used to set interrupt trigger source. There are four interrupt sources provided. They are:

A8316_INTSRC_EXTERNAL : the interrupt is trigger by external source
A8316_INTSRC_EOC: interrupt is triggered when an EOC (A/D converter‘s end of conversion) is asserted.
A8316_INTSRC_INTERNAL: interrupt is triggered by internal timer pacer
A8316_INTSRC_FIFO_HF: interrupt is triggered by FIFO half ready signal.

@ Syntax

Microsoft C/C++

int W_8316_AD_Set_INT_Source (U8 source)

Visual Basic

Windows 3.11 Version:

W_8316_AD_Set_INT_Source (ByVal source As Byte) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_AD_Set_INT_Source (ByVal source As Byte) As Integer

@ Argument

source : interrupt trigger source, the valid interrupt source is:
A8316_INTSRC_EXTERNAL, A8316_INTSRC_EOC,
A8316_INTSRC_INTERNAL,
A8316_INTSRC_FIFO_HF

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.8.14 W_8316_AD_Soft_Trig

@ Description

This function is used to trigger the A/D conversion by software. When the function is called, a trigger pulse will be generated and the converted data will be stored in the address Base+4 and Base+5, and can be retrieved by function W_8316_AD_Aquire().

@ Syntax

Microsoft C/C++

int W_8316_AD_Soft_Trig (void)

Visual Basic

Windows 3.11 Version:

W_8316_AD_Soft_Trig ( ) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_AD_Soft_Trig ( ) As Integer

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.8.15 W_8316_Read_FIFO

@ Description

This function is used to get the AD conversion data which are stored in the FIFO. This function is useful while the FIFO is enabled and the converted A/D data are already stored in FIFO.

@ Syntax

Microsoft C/C++

int W_8316_Read_FIFO (I16 *data)

Visual Basic

Windows 3.11 Version:

W_8316_AD_Aquire (data As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_AD_Aquire (data As Integer) As Integer

@ Argument

data : 16- or 12-bit A/D converted value. Refer to section 2.8.16 for the converted data format.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.8.16 W_8316_AD_Aquire

@ Description

This function is used to poll the A/D conversion data. It will trigger the A/D conversion, and read the 16-bit or 12-bit A/D data until the data is ready ('data ready' bit becomes to low).

@ Syntax

Microsoft C/C++

int W_8316_AD_Aquire (int *ad_data)

Visual Basic

Windows 3.11 Version:

W_8316_AD_Aquire (ad_data As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_AD_Aquire (ad_data As Long) As Long

@ Argument

ad_data : 16-bit A/D converted value.

In **ACL-8316**, 16-bit A/D data is available. The relationship between the voltage and the value is shown in the following table:

A/D Data (Hex)	Decimal Value	Voltage (Volts)
7FFF	+32767	+10.00000
4000	+16384	+5.00015
0001	1	+0.00031
0000	0	0.00000
FFFF	-1	-0.00031
C000	-16384	-5.00015
8001	-32767	-10.00000
8000	-32768	-10.00031

The A/D data format of 12-bit **ACL-8312** is compatible with the 16-bit ACL-8316. Only the 4 LSB of the 16-bit A/D data are truncated to zero. Therefore the software is compatible for the two cards. The relationship between the voltage and the value is shown in the following table:

A/D Data (Hex)	Decimal Value	Voltage (Volts)
7FF 0	+32752	+10.0000
400 0	+16384	+5.0024
001 0	+16	+0.0049
000 0	0	0.0000
FFF 0	-16	-0.0049
C00 0	-16384	-5.0024
801 0	-32752	-10.0000
800 0	-32768	-10.0049

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_AD_AquireTimeOut

2.8.17 W_8316_CLR_IRQ

@ Description

This function is used to clear interrupt request which is requested by the ACL-8316/12. If you use interrupt to transfer A/D converted data, you should use this function to clear interrupt request status, otherwise new coming interrupt can not be generated.

@ Syntax

Microsoft C/C++

```
int W_8316_CLR_IRQ (void)
```

Visual Basic

Windows 3.11 Version:

W_8316_CLR_IRQ () As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_CLR_IRQ () As Integer

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.8.18 W_8316_AD_DMA_Start

@ Description

The function will perform A/D conversion N times with DMA data transfer by using the pacer trigger (internal timer trigger) or external trigger source. It will take place in the background which will not stop until the N-th conversion has been completed or your program execute W_8316_AD_DMA_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function W_8316_AD_DMA_Status().

Note: W_8316_AD_DMA_Start() and W_8316_AD_DMA_Stop() are a pair function, i.e., you have to call W_8316_AD_DMA_Stop() after W_8316_AD_DMA_Start(), otherwise the A/D converted data will not be stored in the buffer you had specified.

@ Syntax

Microsoft C/C++

```
int W_8316_DMA_Start (U8 trig_src, Boolean auto_scan,
          U8 ad_ch_no, U8 ad_range, U8 dma_ch_no, U8
          irq_no, U16 dma_count , I16 *ad_buffer)
```

Visual Basic

Windows 3.11 Version:

```
W_8316_DMA_Start (ByVal trig_src As Byte, ByVal
          auto_scan As Byte, ByVal ad_ch_no As Byte,
          ByVal ad_gain As Byte, ByVal dma_ch_no As Byte,
          ByVal irq_no As Byte, ByVal count As Integer,
          ad_buffer As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8316_DMA_Start (ByVal trig_src As Byte, ByVal
          auto_scan As Byte, ByVal ad_ch_no As Byte,
          ByVal ad_gain As Byte, ByVal dma_ch_no As Byte,
          ByVal irq_no As Byte, ByVal count As Integer,
          ad_buffer As Integer) As Integer
```

@ Argument

trig_src: DMA data transfer trigger source. The valid trigger sources are as follows:
DMA_MODE_0: Internal timer pacer trigger
DMA_MODE_1: External trigger

autoscan: 0: autoscan is disabled
1: autoscan is enabled

ad_ch_no : A/D channel number.

If autoscan is enabled, the A/D channel scan sequence will be: 0, 1, 2, 3,…[ad_ch_no], 0, 1, …, [ad_ch_no], …
If autoscan is disabled, only the data from channel [ad_ch_no] will be converted.

ad_range : Analog input range. The possible value is AD_B_10_V, AD_B_5_V, AD_B_2_5_V, AD_B_1_25_V, AD_U_10_V, AD_U_5_V, AD_U_2_5_V, AD_U_1_25_V.

dma_ch_no : DMA channel number, the valid DMA channel number is DMA_CH_5, DMA_CH_6 or DMA_CH_7

irq_ch_no : IRQ channel number, used to stop DMA

dma_count : the number of A/D conversion

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must be larger than the number of A/D conversion. Each data element of ad_buffer contains 16-bit A/D transfer data.

Note： While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidDMAChannel,
ERR_InvalidIRQChannel, ERR_InvalidTimerValue

2.8.19 W_8316_AD_ContDMA_Start

@ Description

The function will perform continuous A/D conversions with DMA data transfer by using the pacer trigger (internal timer trigger) or external trigger source.
It will take place in the background and will not be stopped until your program executes W_8316_AD_DMA_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function W_8316_AD_DblBufferHalfReady().

Note: W_8316_AD_ContDMA_Start() and W_8316_AD_DMA_Stop() are a pair function, i.e., you have to call W_8316_AD_DMA_Stop() after W_8316_AD_ContDMA_Start(), otherwise the A/D converted data will not be stored in the buffer you had specified.

@ Syntax

Microsoft C/C++

```
int W_8316_ContDMA_Start (U8 trig_src, Boolean
        auto_scan, U8 ad_ch_no, U8 ad_range, U8
        dma_ch_no, U8 irq_no, U16 dma_count , I16
        *ad_buffer)
```

Visual Basic

Windows 3.11 Version:

```
W_8316_ContDMA_Start (ByVal trig_src As Byte, ByVal
        auto_scan As Byte, ByVal ad_ch_no As Byte,
        ByVal ad_gain As Byte, ByVal dma_ch_no As Byte,
        ByVal irq_no As Byte, ByVal count As Integer,
        ad_buffer As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8316_ContDMA_Start (ByVal trig_src As Byte, ByVal
        auto_scan As Byte, ByVal ad_ch_no As Byte,
        ByVal ad_gain As Byte, ByVal dma_ch_no As Byte,
        ByVal irq_no As Byte, ByVal count As Integer,
        ad_buffer As Integer) As Integer
```

@ Argument

trig_src: DMA data transfer trigger source. The valid trigger sources are as follows:
DMA_MODE_0: Internal timer pacer trigger
DMA_MODE_1: External trigger

autoscan: 0: autoscan is disabled
1: autoscan is enabled

ad_ch_no : A/D channel number.

If autoscan is enabled, the A/D channel scan sequence will be:
0, 1, 2, 3,…[ad_ch_no], 0, 1, …, [ad_ch_no], …
If autoscan is disabled, only the data from channel [ad_ch_no] will be converted.

ad_range : Analog input range. The possible value is AD_B_10_V, AD_B_5_V, AD_B_2_5_V, AD_B_1_25_V, AD_U_10_V, AD_U_5_V, AD_U_2_5_V, AD_U_1_25_V.

dma_ch_no : DMA channel number, the valid DMA channel number is DMA_CH_5, DMA_CH_6 or DMA_CH_7

irq_ch_no : IRQ channel number, used to stop DMA

dma_count : the number of A/D conversion

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must be larger than the number of A/D conversion. Each data element of ad_buffer contains 16-bit A/D transfer data.

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError
ERR_AD_DMANotSet
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidDMAChannel,
ERR_InvalidIRQChannel, ERR_InvalidTimerValue

2.8.20 W_8316_AD_DMA_Status

@ Description

Since the W_8316_AD_DMA_Start function executes in background, you can issue the function W_8316_AD_DMA_Status() to check its operation status.

@ Syntax

Microsoft C/C++

```
int W_8316_AD_DMA_Status (U8 *status , U16 *count)
```

Visual Basic

Windows 3.11 Version:

W_8316_AD_DMA_Status (status As Byte, count As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_AD_DMA_Status (status As Byte, count As Integer) As Integer

@ Argument

status :	status of the DMA data transfer AD_DMA_STOP : A/D DMA is completed AD_DMA_RUN : A/D DMA is not completed
count :	the number of A/D data which has been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ADDMANotSet

2.8.21 W_8316_AD_DMA_Stop

@ Description

This function is used to stop the DMA data transfer. After executing this function, the internal A/D trigger is disabled and the A/D timer (timer #1 and #2) is stopped. The function returns the number of the data which has been transferred, no matter the A/D DMA data transfer is stopped by this function or by the DMA terminal count ISR.

This function has to be called after W_8316_AD_DMA_Start() function issued. Otherwise, all converted data will not be saved into the memory buffer you specified in your program.

@ Syntax

Microsoft C/C++

```
int W_8316_AD_DMA_Stop (int *count)
```

Visual Basic

Windows 3.11 Version:

```
W_8316_AD_DMA_Stop (count As Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8316_AD_DMA_Stop (count As Long) As Long
```

@ Argument

count : the number of A/D converted data which has been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ADDMANotSet

2.8.22 W_8316_AD_INT_Start

@ Description

The function will perform A/D conversion N times with interrupt data transfer by using pacer trigger. It will takes place in the background which will not be stopped until the N-th conversion has been completed or your program execute W_8316_AD_INT_Stop() function to stop the process. After executing this function, it is necessary to check the status of the operation by using the function W_8316_AD_INT_Status().

Note: W_8316_AD_INT_Start(), and W_8316_AD_INT_Stop() are a pair of functions, i.e., you have to call W_8316_AD_INT_Stop() after W_8316_AD_INT_Start(), otherwise the A/D converted data will not be stored in the buffer you had specified.

@ Syntax

Microsoft C/C++

```
int W_8316_INT_Start (U8 ad_mode, Boolean autoscan,
        U8 ad_ch_no, U8 ad_range, U8 irq_no, U16 count,
        I16 *ad_buffer)
```

Visual Basic

Windows 3.11 Version:

```
W_8316_INT_Start (ByVal ad_mode As Byte, ByVal
        auto_scan As Integer, ByVal ad_ch_no As Byte,
        ByVal ad_gain As Byte, ByVal irq_no As Byte,
        ByVal count As Integer, ad_buffer As Integer) As
        Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8316_INT_Start (ByVal ad_mode As Byte, ByVal
        auto_scan As Integer, ByVal ad_ch_no As Byte,
        ByVal ad_gain As Byte, ByVal irq_no As Byte,
        ByVal count As Integer, ad_buffer As Integer) As
        Long
```

@ Argument

int_mode: A/D conversion by interrupt data transfer. The modes supported by this library are:

A8316_INT_MODE_0 : Internal timer pacer trigger A/D conversion with EOC(end of conversion) trigger interrupt, and get A/D converted data through I/O port.

A8316_INT_MODE_1 : Internal timer pacer trigger A/D conversion with FIFO_HF(FIFO half full ready) trigger interrupt, and get 512 A/D converted data through I/O port.

A8316_INT_MODE_2 : External Trigger A/D conversion with EOC(end of conversion) trigger interrupt, and get A/D converted data through I/O port.

A8316_INT_MODE_3 : External trigger A/D conversion, with FIFO_HF(FIFO half full ready) trigger interrupt, and get 512 A/D converted data through I/O port.

Note: If int_mode is A8316_INT_MODE_1or A8316_INT_MODE_3, this function uses FIFO-Half-Full interrupt transfer mode. So the value of *count* must be the multiple of 512.

Autoscan: 0: autoscan is disabled
1: autoscan is enabled

ad_ch_no : A/D channel number

ad_range : analog input range value. The possible values are:
AD_B_10_V, AD_B_5_V, AD_B_2_5_V, AD_B_1_25_V, AD_U_10_V, AD_U_5_V, AD_U_2_5_V, AD_U_1_25_V.

irq_ch_no : IRQ channel number used to transfer A/D data, the possible value is defined in file Dll2.h.

count : number of A/D conversions to perform

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must be large than the number of A/D conversions. Each data element of ad_buffer contains 16-bit A/D transfer data.

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidIRQChannel,
ERR_InvalidTimerValue

2.8.23 W_8316_AD_INT_Status

@ Description

Since the W_8316_AD_INT_Start() function executes in background, you can issue the function W_8316_AD_INT_Status() to check the status of interrupt operation.

@ Syntax

Microsoft C/C++

int W_8316_AD_INT_Status (int *status , int *count)

Visual Basic

Windows 3.11 Version:

W_8316_AD_INT_Status (status As Integer, count As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_AD_INT_Status (status As Long, count As Long) As Long

@ Argument

status :	status of the interrupt data transfer AD_INT_STOP : interrupt A/D is completed AD_INT_RUN : interrupt A/D is not completed
count :	the number of A/D data which has been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.8.24 W_8316_AD_INT_Stop

@ Description

This function is used to stop the interrupt data transfer function. After executing this function, the internal A/D trigger is disabled and the A/D timer stops. The function returns the number of the data which has been transferred, no matter whether if the A/D interrupt data transfer is stopped by this function or by the W_8316_AD_INT_Start() itself.

This function has to be called after W_8316_AD_INT_Start() function issued. Otherwise, all converted data will not be saved into the memory buffer you had specified in W_8316_AD_INT_Start() function call.

@ Syntax

Microsoft C/C++

int W_8316_AD_INT_Stop (int *count)

Visual Basic

Windows 3.11 Version:

W_8316_AD_INT_Stop (count As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_AD_INT_Stop (count As Long) As Long

@ Argument

count : the number of A/D data which have been transferred.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.8.25 W_8316_AD_ContINT_Start

@ Description

The function will perform continuous A/D with interrupt data transfer by using pacer trigger. It will take place in the background which will not be stopped until your program execute W_8316_AD_INT_Stop() function to stop the process. After calling this function, it is necessary to check the status by using the function W_8316_AD_DblBufferHalfReady().

Note: W_8316_AD_ContINT_Start(), and W_8316_AD_INT_Stop() are a pair of functions, i.e., you have to call W_8316_AD_INT_Stop() after W_8316_AD_ContINT_Start(), otherwise the A/D converted data will not be stored in the buffer you had specified.

@ Syntax

Microsoft C/C++

```
int W_8316_ContINT_Start (U8 ad_mode, Boolean
        autoscan, U8 ad_ch_no, U8 ad_range, U8 irq_no,
        U16 count, I16 *ad_buffer)
```

Visual Basic

Windows 3.11 Version:

```
W_8316_ContINT_Start (ByVal ad_mode As Byte, ByVal
        auto_scan As Integer, ByVal ad_ch_no As Byte,
        ByVal ad_gain As Byte, ByVal irq_no As Byte,
        ByVal count As Integer, ad_buffer As Integer) As
        Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8316_ContINT_Start (ByVal ad_mode As Byte, ByVal
        auto_scan As Integer, ByVal ad_ch_no As Byte,
        ByVal ad_gain As Byte, ByVal irq_no As Byte,
        ByVal count As Integer, ad_buffer As Integer) As
        Long
```

@ Argument

int_mode: A/D conversion by interrupt data transfer. The modes supported by this library are:

A8316_INT_MODE_0 : Internal timer pacer trigger A/D conversion with EOC(end of conversion) trigger interrupt, and get A/D converted data through I/O port.

A8316_INT_MODE_1 : Internal timer pacer trigger A/D conversion with FIFO_HF(FIFO half full ready) trigger interrupt, and get 512 A/D converted data through I/O port.

A8316_INT_MODE_2 : External Trigger A/D conversion with EOC(end of conversion) trigger interrupt, and get A/D converted data through I/O port.

A8316_INT_MODE_3 : External trigger A/D conversion, with FIFO_HF(FIFO half full ready) trigger interrupt, and get 512 A/D converted data through I/O port.

Note: If int_mode is A8316_INT_MODE_1or A8316_INT_MODE_3, this function uses FIFO-Half-Full interrupt transfer mode. So the value of *count* must be the multiple of 1024 for double-buffer mode.

Autoscan: 0: autoscan is disabled
1: autoscan is enabled

ad_ch_no : A/D channel number

If autoscan is enabled, the A/D channel scan sequence will be:
0, 1, 2, 3,…[ad_ch_no], 0, 1, …, [ad_ch_no], …
If autoscan is disabled, only the data from channel [ad_ch_no] will be converted.

ad_range : analog input range value. The possible values are:
AD_B_10_V, AD_B_5_V, AD_B_2_5_V, AD_B_1_25_V, AD_U_10_V, AD_U_5_V, AD_U_2_5_V, AD_U_1_25_V.

irq_ch_no : IRQ channel number used to transfer A/D data, the possible value is defined in file Dll2.h.

count : number of A/D conversions to perform

ad_buffer : the start address of the memory buffer to store the A/D data, the buffer size must be large than the number of A/D conversions. Each data element of ad_buffer contains 16-bit A/D transfer data.

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of *ad_buffer*. For example, if the name of array is *buf*, pass *buf(0)* as argument if index number of *buf* begins from 0. Also with Windows 3.11 version, because the Integer type in Visual Basic is signed integer (i.e., its range is from -32768 to 32767), if you want to specify *c1* or *c2* to number larger than 32767, please set it as the number minus 65536. For example, if you want to set c1 as 40000, please set it as (40000 - 65536) = -25536 instead.

@ Return Code

ERR_NoError, ERR_InvalidCounterValue
ERR_BoardNoInit, ERR_InvalidADChannel,
ERR_InvalidADGain, ERR_InvalidIRQChannel,
ERR_InvalidTimerValue

2.8.26 W_8316_AD_DblBufferHalfReady

@ Description

Checks whether the next half buffer of data in circular buffer is ready for transfer during an double-buffered analog input operation.

@ Syntax

Microsoft C/C++

```
int W_8316_AD_DblBufferHalfReady ( BOOLEAN
        *bHalfReady)
```

Visual Basic

```
W_8316_AD_DblBufferHalfReady (bHalfReady As Integer)
        As Integer
```

@ Argument

bHalfReady : Whether the next half buffer of data is available.If *HalfReady* = TRUE, you can call **`W_8316_AD_DblBufferTransfer()`** to copy the data to your user buffer.

@ Return Code

ERR_NoError

2.8.27 W_8316_AD_DblBufferTransfer

@ Description

Depending on the continuous AI function elected, half of the data in circular buffer will be logged into the user buffer . You can execute this function repeatedly to return sequential half buffers of the data.

@ Syntax

Microsoft C/C++

```
int W_8316_AD_DblBufferTransfer (USHORT *pwBuffer)
```

Visual Basic

```
W_8316_AD_DblBufferTransfer (pwBuffer As Integer) As
        Integer
```

@ Argument

pwBuffer: The user buffer. An integer array to which the data is to be copied.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.8.28 W_8316_AD_Timer

@ Description

This function is used to setup the Timer #1 and Timer #2. The values of c1 and c2 are used as frequency dividers for generating constant A/D sampling rate dedicatedly. It is possible to stop the pacer trigger by setting any one of the dividers as 0. Because the A/D conversion rate is limited due to the conversion time of the A/D converter, the highest sampling rate of the ACL-8316/12 can not exceed 100 KHz. The multiplication of the dividers must be larger than 20.

@ Syntax

Microsoft C/C++

```
int W_8316_AD_Timer( unsigned int c1, unsigned int c2 )
```

Visual Basic

Windows 3.11 Version:

```
W_8316_AD_Timer (ByVal c1 As Integer, ByVal c2 As
        Integer) As Integer
```

Win-95/98, Win-NT/2000 Version:

```
W_8316_AD_Timer (ByVal c1 As Long, ByVal c2 As Long)
        As Long
```

@ Argument

c1 :	frequency divider of timer #1
c2 :	frequency divider of timer #2

Note : the A/D sampling rate is equal to : 2MHz / (c1*c2),   when c1 = 0 or c2 = 0, the pacer trigger will be stopped.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerValue

2.8.29 W_8316_Timer_Start

@ Description

The Timer #0 on the ACL-8316 can be freely programmed by the users. This function is used to program the Timer #0. This timer can be used as frequency generator if internal clock is used. It also can be used as event counter if external clock is used. All the 8253 modes are available.

@ Syntax

Microsoft C/C++

int W_8316_Timer_Start (int timer_mode, unsigned int c0)

Visual Basic

Windows 3.11 Version:

W_8316_Timer_Start (ByVal timer_mode As Integer,
ByVal c0 As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_Timer_Start (ByVal timer_mode As Long, ByVal
c0 As Long) As Long

@ Argument

timer_mode :	the 8253 timer mode, the possible values are : TIMER_MODE0, TIMER_MODE1, TIMER_MODE2, TIMER_MODE3, TIMER_MODE4, TIMER_MODE5.
c0 :	the counter value of timer

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerMode

2.8.30 W_8316_Timer_Read

@ Description

This function is used to read the counter value of the Timer #0.

@ Syntax

Microsoft C/C++

int W_8316_Timer_Read (unsigned int *counter_value)

Visual Basic

Windows 3.11 Version:

W_8316_Timer_Read (counter_value As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_Timer_Read (counter_value As Long) As Long

@ Argument

counter_value : the counter value of the Timer #0

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.8.31 W_8316_Timer_Stop

@ Description

This function is used to stop the timer operation. The timer is set to the 'One-shot' mode with counter value '0'. That is, the clock output signal will be set to high after executing this function.

@ Syntax

Microsoft C/C++

int W_8316_Timer_Stop (unsigned int *counter_value)

Visual Basic

Windows 3.11 Version:

W_8316_Timer_Stop (counter_value As Integer) As Integer

Win-95/98, Win-NT/2000 Version:

W_8316_Timer_Stop (counter_value As Long) As Long

@ Argument

counter_value : the current counter value of the Timer #0

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.8.32 W_8316_DMA_InitialMemoryAllocated

@ Description

This function is only available in Windows NT and Windows 2000 system. This function returns the available memory size for DMA data transfer in the device driver in argument ***MemSize***. While performming analog input with DMA data transfer, the analog input size can not exceed this size.

@ Syntax

Microsoft C/C++

```
W_8316_DMA_InitialMemoryAllocated(int *MemSize)
```

Visual Basic

Win-NT/2000 Version:

```
W_8316_DMA_InitialMemoryAllocated(MemSize As Long)
          As Long
```

@ Argument

MemSize : the available memory size for DMA data transfer in device driver of ACL-8316/12.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

Appendix A Status Codes

This appendix lists the status codes returned by ACLS-DLL2, including the name and description.

Each ACLS-DLL2 function returns a status code that indicates whether the function was performed successfully. When an ACLS-DLL2 function returns a non-zero number, it means that an error occurred while executing the function.

Status Code	Status Name	Description
0	ERR_NoError	No error occurred
1	ERR_BoardNoInit	The specified board is not initialized
2	ERR_InvalidBoardNumber	The card_number argument is not valid
3	ERR_InitializedBoardNumber	The board with the specified board number is not initialized
4	ERR_BaseAddressError	The specified base address argument is invalid
5	ERR_BaseAddressConflict	The specified base address argument conflicts with other hardware resource
6	ERR_DuplicateBoardSetting	The base addresses setting for two or more devices are the same
7	ERR_DuplicateIrqSetting	The irq setting for two or more devices are the same
8	ERR_PortError	The specified port is invalid
9	ERR_ChannelError	The specified Channel is invalid
10	ERR_InvalidADChannel	The specified AD Channel is invalid
11	ERR_InvalidDAChannel	The specified DA Channel is invalid
12	ERR_InvalidDIChannel	The specified DI Channel is

		invalid
13	ERR_InvalidDOChannel	The specified DO Channel is invalid
14	ERR_InvalidDIOChannel	The specified programmable DI/O Channel is invalid
15	ERR_InvalidIRQChannel	The specified IRQ level is invalid
16	ERR_InvalidDMAChannel	The specified DMA Channel is invalid
17	ERR_InvalidChangeValue	The updated value is invalid
18	ERR_InvalidTimerValue	The given counter value is invalid
19	ERR_InvalidTimerMode	The specified 8254 Timer Mode is invalid
20	ERR_InvalidCounterValue	The specified Counter value is invalid
21	ERR_InvalidCounterMode	The specified 8254 Counter Mode is invalid
22	ERR_InvalidADMode	The AD Mode is invalid
23	ERR_InvalidMode	The specified mode is invalid
24	ERR_NotOutputPort	The specified DO port is invalid
25	ERR_NotInputPort	The specified DI port is invalid
26	ERR_AD_DMANotSet	The DMA data operation for analog input is not initialized
27	ERR_AD_INTNotSet	The Interrupt operation for analog input is not initialized
28	ERR_AD_AquireTimeOut	Time Out for AD operation
29	ERR_AD_InvalidGain	The specifed analog input gain code is invalid
30	ERR_INTNotSet	The Interrupt operation for digital input or output is not initialized
31	ERR_InvalidPortNumber	The specified port number is invalid
32	ERR_InvalidTrigSrc	The specified trigger source is invalid

33	ERR_InvalidINTMode	The specified interrupt mode is invalid
34	ERR_InvalidINTMode	The specified interrupt mode is invalid