ACLS-DLL1 ver. 5.0

Software Driver for

Windows 3.11, Win-95/98, Win-NT and Win-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 Manual

The function reference manual is designed to help you use the ACLS-DLL1 software driver for ADlink's digital I/O cards ACL-7130, ACL-7120, ACL-7122, ACL-7124, ACL-7125, ACL-7225, and PET-48DIO. The manual contains the detailed descriptions of the ACLS-DLL1 functions. When you are familiar with the material in the *ACLS-DLL1 User Guide*, you can use this manual for detailed information about each ACLS-DLL1 function.
The *ACLS-DLL1 Function Reference Manual* is organized as follows:

- Chapter 1, "Using ACLS-DLL1 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-DLL1 provided.
- Appendix A, "Status Code" lists the status codes returned by ACLS-DLL1 functions, as well as their meaning.

1

Using ACLS-DLL1 Functions

ACLS-DLL1 is a package of software drivers for NuDAQ series ISA bus digital I/O cards ACL-7130, ACL-7120, ACL-7122, ACL-7124, ACL-7125, ACL-7225, and PET-48DIO. It includes high performance data acquisition drivers for developing custom applications under Windows 3.1, Windows 95, Win-NT 4.0 and Windows 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-DLL1

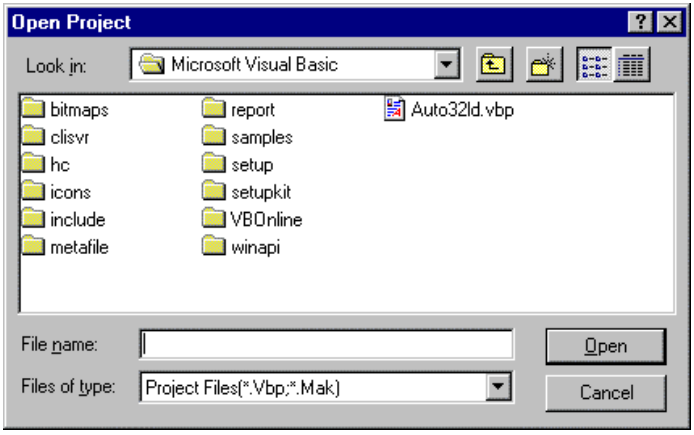
1.1.1 Creating An Application Using Visual Basic and ACLS-DLL1

To create a data acquisition application using ACLS-DLL1 and Visual Basic, follow these steps after entering Visual Basic:

step 1 Open the project in which you want to use ACLS-DLL1. 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

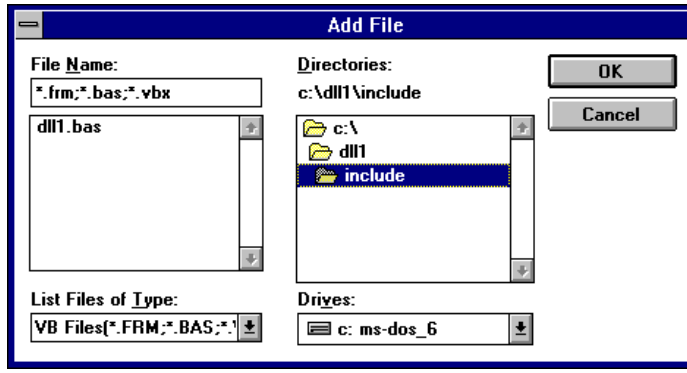
selecting the Open Project command from the File menu. Then the Open Project dialog box appears.



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 DLL1.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.



Select DLL1.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, DLL1.BAS is installed in C:\ACL-DLL1\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 Window menu or press F4, 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, you double click the desired control or form to view the code module then

add code you want. You can call the procedures that declared in the file DLL1.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-DLL1's DLL and driver files. These files should be copied to their appropriated directory as section 2.1.3 described.

1.1.2 Creating An Application Using Microsoft C/C++, Windows SDK, and ACLS-DLL1

To create a data acquisition application using ACLS-DLL1, Microsoft Visual C/C++, follow these steps after entering Visual C/C++:

step 1 Open the project in which you want to use ACLS-DLL1. This can be a new or existing project

step 2 Include header files DLL1.H in the C/C++ source files that call ACLS-DLL1 functions. DLL1.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 "DLL1.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-DLL1's import libraries.

1.2 ACLS-DLL1 Functions Overview

In ACLS-DLL1, each NuDAQ digital I/O card has its own DLL driver. The function calls in these DLLs use intuitive names that reflect the operations they perform. For example, W_7122_DI performs *Digital Input* for *ACL-7122* card.

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

1. Initialization	: set the hardware base I/O address
2. Digital I/O	: input or output digital signals
3. Interrupt operation	: input or output digital signals through interrupt operation
4. Timer/Counter	: Timer/Counter operation

1.3 Functions Naming Convention

The functions of ACL-DLL1 use full-names to represent the real meaning of the functions. The naming convention rules are:

In DOS Environment :

_{hardware_model}_{action_name}. e.g. **_7122_Initial** ().

In order to recognize the difference between DOS library and Windows library, a capital "**W**" is put on the head of each function name of the Windows DLL driver. e.g. **W_7122_Initial** ()

2

Function Reference

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

2.1 ACL-7122 Software DLL Driver

In this section, the ACL-7122's (ACL-722's) software DLL drivers: C language library for DOS and DLL driver for Windows systems are described. The function names of Windows 3.11, Window 95, and Windows NT/2000 versions are the same. Thus, users do not need to learn the difference between them. The portability of applications between these three systems can be very high.

Note: All functions of the ACL-7122 can be applied to the ACL-722 directly.

There are 11 functions provided for ACL-7122 Digital I/O cards. The functions used in DOS and their corresponding functions used in Windows are specified in the following table.

Functions in DOS	Functions in Windows
_7122_Initial	W_7122_Initial
_7122_Set_Card	W_7122_Set_Card
_7122_Get_Card	W_7122_Get_Card
_7122_DO	W_7122_DO
_7122_DI	W_7122_DI
_7122_INT_Start	W_7122_INTOP_Start
_7122_INT_Status	W_7122_INTOP_Status

_7122_INT_Stop	W_7122_INTOP_Stop
---	W_7124_INT_Enable
---	W_7124_INT_Disable
_7122_INT_Op	W_7122_INT_Op
_7122_INT_Reset	W_7122_INT_Reset

The detailed description of each function is specified in the following sections.

2.1.1 W_7122_Initial / _7122_Initial

@ Description

An ACL-7122 card is initialized according to the card number, its corresponding base address and IRQ level. Every ACL-7122 Digital I/O card have to be initialized by this function before calling other functions.

@ Syntax

Microsoft C/C++

int W_7122_Initial(int card_number, int base_address, int irq)

Visual Basic

Windows 3.11 Version:

W_7122_Initial (ByVal card_number As Integer, ByVal base_address As Integer, ByVal irq As Integer) As Integer

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

W_7122_Initial (ByVal card_number As Long, ByVal base_address As Long, ByVal Irq As Long) As Long

C/C++ (DOS)

int _7122_Initial(int card_number, int base_address, int irq)

@ 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.

Irq : The IRQ level of your ACL-7122 card. This irq value should be the same as hardware setting.

Note: Since Windows NT arranges 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 DrvUtil utility to set IRQ level before running application. Please refer to section 2.1.3 " Installation".

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError
ERR_InvalidIRQChannel

2.1.2 W_7122_Set_Card / _7122_Set_Card

@ Description

This function is used on multi-cards system. After the ACL-7122 cards are initialized by **W_7122_Initial** function, you can use this function to select which one you want to operate.

@ Syntax

Microsoft C/C++

int W_7122_Set_Card(int card_number)

Visual Basic

Windows 3.11 Version:

W_7122_Set_Card (ByVal card_number As Integer) As Integer

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

W_7122_Set_Card (ByVal card_number As Long) As Long

C/C++ (DOS)

int _7122_Set_Card(int card_number)

@ Argument

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

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber

2.1.3 W_7122_Get_Card / _7122_Get_Card

@ Description

This function is used on multi-cards system. You can use this function to get which card is operated currently.

@ Syntax

Microsoft C/C++

int W_7122_Get_Card(int *card_number)

Visual Basic

Windows 3.11 Version:

W_7122_Get_Card (card_number As Integer) As Integer

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

W_7122_Get_Card (card_number As Long) As Long

C/C++ (DOS)

int _7122_Get_Card(int *card_number)

@ Argument

card_number : card identification.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.1.4 W_7122_DI / _7122_DI

@ Description

This function is used to read data from digital input ports. There are six connectors (0, ..., 5) on the ACL-7122. Each connector is divided into PortA, PortB, and PortC. The PortC can also be configured as PortC_Upper, and PortC_Lower. All of these ports can be set as input mode, i.e. read data from the ports. By using

this function, you can easily to read input signals from any port of different connectors.

@ Syntax

Microsoft C/C++

int W_7122_DI(int cn_port, unsigned char *di_data)

Visual Basic

Windows 3.11 Version:

W_7122_DI (ByVal cn_port As Integer, di_data As Integer) As Integer

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

W_7122_DI (ByVal cn_port As Long, di_data As Long) As Long

C/C++ (DOS)

int _7122_DI(int cn_port, unsigned char *di_data)

@ Argument

cn_port : To indicate which connector and its associated port is read, the valid data is :

CH0_PA	Prot A of Connector 0
CH0_PB	Port B of Connector 0
CH0_PC	Port C of Connector 0
CH0_PCU	Upper Port C of Connector 0
CH0_PCL	Low Port C of Connector 0
:	:
CH5_PA	Prot A of Connector 5
CH5_PB	Port B of Connector 5
CH5_PC	Port C of Connector 5
CH5_PCU	Upper Port C of Connector 5
CH5_PCL	Low Port C of Connector 5

di_data : return value from digital port.

@ Return Code

ERR_NoError
ERR_BoardNoInit

ERR_PortError

2.1.5 W_7122_DO / _7122_DO

@ Description

This function is used to write data to digital output ports. There are six connectors (0, ..., 5) on the ACL-7122. Each connector is divided into PortA, PortB, and PortC. The PortC can also be configured as PortC_Upper, and PortC_Lower. All of these ports can be set as output mode, i.e. write data to the ports. By using this function, you can easily write signals to any port of different connectors.

@ Syntax

Microsoft C/C++

int W_7122_DO(int cn_port, unsigned char do_data)

Visual Basic

Windows 3.11 Version:

W_7122_DO (ByVal cn_port As Integer, ByVal do_data As Integer) As Integer

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

W_7122_DO (ByVal cn_port As Long, ByVal do_data As Long) As Long

C/C++ (DOS)

int _7122_DO(int cn_port, unsigned char do_data)

@ Argument

cn_port : To indicate which connector and its associated port is read, the valid data is :

CH0_PA	Prot A of Connector 0
CH0_PB	Port B of Connector 0
CH0_PC	Port C of Connector 0
CH0_PCU	Upper Port C of Connector 0
CH0_PCL	Low Port C of Connector 0
:	:
CH5_PA	Prot A of Connector 5
CH5_PB	Port B of Connector 5

CH5_PC	Port C of Connector 5
CH5_PCU	Upper Port C of Connector 5
CH5_PCL	Low Port C of Connector 5

do_data : value that is written to digital output port

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ChannelError
ERR_PortError

2.1.6 W_7122_INTOP_Start / W_7122_INT_Start / _7122_INT_Start

@ Description

The function will perform digital input or output N times with interrupt data transfer by using external interrupt trigger. It takes place in the background which will not stop until the N-th conversion has completed or your program calls W_7122_INTOP_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_7122_INTOP_Status(). It can help you to make sure there is no interrupt operation being processed in background.

Before using this function, you have to define the operation of each connector's port. Three operation types can be chosen for each port : NO_OP, INPUT_OP, and OUTPUT_OP. The mode of each port is set by using the function W_7122_INT_Op(). (please refer to W_7122_INT_Op() function in section 2.1.10)

Note : The W_7122_INTOP_Start and W_7122_INTOP_Stop are a pair of functions. That is, as the W_7122_INTOP_Start is called, the W_7122_INTOP_Stop has to follow up behind it. Otherwise, the "Digital Input" data will not be stored in the buffer.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

int W_7122_INT_Start(int count)

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

int W_7122_INTOP_Start(int count)

Visual Basic

Windows 3.11 Version:

W_7122_INT_Start (ByVal count As Integer) As Integer

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

W_7122_INTOP_Start (ByVal count As Long) As Long

C/C++ (DOS)

int _7122_INT_Start(int count)

@ Argument

count : the number of input and output have to be operated by interrupt trigger.

@ Return Code

ERR_NoError
ERR_INTNotSet

@ Example

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

```
Dim dio_buf(1024) As Long
Dim cn_port As Long
Dim count As Long
Dim op As Long
Dim Ret As Long
        .
        .
        .
Ret = W_7122_INT_Reset( );
Ret = W_7122_INT_Op( cn_port, op, dio_buf, buf_cnt);
Ret = W_7122_INTOP_Start( count)
        .
        .
```

2.1.7 W_7122_INTOP_Status / W_7122_INT_Status / _7122_INT_Status

@ Description

Since the W_7122_INTOP_Start() function is executed in background, you can issue the function W_7122_INTOP_Status() to check the status of the interrupt operation.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

int W_7122_INT_Status(int *status , int *count)

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

int W_7122_INTOP_Status(int *status , int *count)

Visual Basic

Windows 3.11 Version:

W_7122_INTOP_Status(status As Integer, count As Integer) As Integer

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

W_7122_INTOP_Status (status As Long, count As Long) As Long

C/C++ (DOS)

int _7122_INT_Status(int *status , int *count)

@ Argument

status : status of the INT data transfer
INT_STOP : Digital I/O INT is completed
INT_RUN: Digital I/O INT is not completed

count : current I/O operation count number.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.1.8 W_7122_INTOP_Stop / W_7122_INT_Stop / _7122_INT_Stop

@ Description

This function is used to stop the interrupt data transfer function. After executing this function, the digital I/O interrupt operation stops. The function returns the number of the digital I/O operations which has been done, no matter whether the Digital I/O interrupt operation is stopped by this function or by the W_7122_INTOP_Start() itself.

Note : This function has to be called after the W_7122_INTOP_Start is called. Otherwise the input data will not be stored in the buffer.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

int W_7122_INT_Stop(int *count)

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

int W_7122_INTOP_Stop(int *count)

Visual Basic

Windows 3.11 Version:

W_7122_INTOP_Stop(count As Integer) As Integer

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

W_7122_INTOP_Stop (count As Long) As Long

C/C++ (DOS)

int _7122_INT_Stop(int *count)

@ Argument

count : the number of digital I/O operations which have been done.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.1.9 W_7122_INT_Enable

@ Description

This function is only available in Window 95 driver and Windows NT/2000 driver. The function is used to initialize and start up the interrupt control. After calling this function, every time an interrupt request signal 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 samples program 7122int.c.

Note : The W_7122_INT_Enable and W_7122_INT_Disable are a pair of functions. That is, as the W_7122_INT_Enable is called, the W_7122_INT_Disable has to follow up behind it. Otherwise, the interrupt signal generation will stop.

@ Syntax

Microsoft C/C++

```
int W_7122_INT_Enable(HANDLE *hIntEvent)
```

Visual Basic

```
W_7122_INT_Enable (hIntEventAs Long ) As Long
```

@ Argument

hIntEvent : the handle of the event for interrupt signals.

@ Return Code

ERR_NoError
ERR_INTNotSet

2.1.10 W_7122_INT_Disable

@ Description

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

Note : This function has to be called after the W_7122_INT_Enable is called.

@ Syntax

Microsoft C/C++

```
int W_7122_INT_Disable()
```

Visual Basic

W_7122_INT_Disable () As Long

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.1.11 W_7122_INT_Op / _7122_INT_Op

@ Description

This function is used to set up the operation of each port when the interrupt is triggered by external signals. Before the W_7122_INT_Start is executed, you have to set up the I/O status of each digital port that will perform interrupt data transfer.

Note : The Windows Version and DOS version are different. In Windows Version: the buffer size has to be specified in this function. But in DOS version, it does not need to be specified.

@ Syntax

Microsoft C/C++

int W_7122_INT_Op (int cn_port , int op, unsigned int *buf, int buf_cnt)

Visual Basic

Windows 3.11 Version:

W_7122_INT_Op (ByVal cn_port As Integer, ByVal op As Integer, buf As Integer, ByVal buf_cnt As Integer) As Integer

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

W_7122_INT_Op (ByVal cn_port As Long, ByVal op As Long, buf As Long, ByVal buf_cnt As Long) As Long

C/C++ (DOS)

int _7122_INT_Op (int cn_port , int op, unsigned int far *buf)

@ Argument

cn_port : To indicate which connector and its associated port, the valid data is :

CH0_PA	Prot A of Connector 0
CH0_PB	Port B of Connector 0
CH0_PC	Port C of Connector 0
CH0_PCU	Upper Port C of Connector 0
CH0_PCL	Low Port C of Connector 0
CH1_PA	Prot A of Connector 1
CH1_PB	Port B of Connector 1
CH1_PC	Port C of Connector 1
:	:
CH5_PA	Prot A of Connector 5
CH5_PB	Port B of Connector 5
CH5_PC	Port C of Connector 5
CH5_PCU	Upper Port C of Connector 5
CH5_PCL	Low Port C of Connector 5

op : Digital I/O status

NO_OP : No Operation
INPUT_OP : Input Operation
OUTPUT_OP : Output Operation

buf : the start address of the memory buffer to write/read the digital output/input data, the buffer size must be larger than the number of digital I/O operations.

buf_cnt : the number of data in buffer.

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of buffer. 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

2.1.12 W_7122_INT_Reset / _7122_INT_Reset

@ Description

This function is used to reset all ports' interrupt transfer operations. After executing this function, each port is set as NO_OP (no operation). That is, no digital I/O operation will be performed as an interrupt is occurred.

@ Syntax

Microsoft C/C++

int W_7122_INT_Reset( )

Visual Basic

Windows 3.11 Version:

W_7122_INT_Reset ( ) As Integer

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

W_7122_INT_Reset ( ) As Long

C/C++ (DOS)

int _7122_INT_Reset ( )

@ Argument

Not arguments.

@ Return Code

ERR_NoError

2.2 ACL_7124 Software Drivers

Two types of ACL-7124 drivers are provided: C Language library for DOS and DLL driver for Windows.

Note: All functions of the ACL-7124 can be applied to the ACL-724 directly.

There are 12 functions provided for ACL-7124 Digital I/O cards. The functions used in DOS and their corresponding functions used in Windows are specified in the following table.

Functions in DOS	Functions in Windows
_7124_Initial	W_7124_Initial
_7124_Set_Card	W_7124_Set_Card
_7124_Get_Card	W_7124_Get_Card
_7124_DO	W_7124_DO
_7124_DI	W_7124_DI
_7124_INT_Start	W_7124_INTOP_Start
_7124_INT_Status	W_7124_INTOP_Status
_7124_INT_Stop	W_7124_INTOP_Stop
---	W_7124_INT_Enable
---	W_7124_INT_Disable
_7124_INT_Op	W_7124_INT_Op
_7124_INT_Reset	W_7124_INT_Reset

The detailed description of each function is specified in the following sections.

2.2.1 W_7124_Initial / _7124_Initial

@ Description

An ACL-7124 card is initialized according to the card number, its corresponding base address and IRQ level. Every ACL-7124 Digital I/O card have to be initialized by this function before calling other functions.

@ Syntax

Microsoft C/C++

int W_7124_Initial(int card_number, int base_address, int irq)

Visual Basic

Windows 3.11 Version:

W_7124_Initial (ByVal card_number As Integer, ByVal base_address As Integer, ByVal irq As Integer) As Integer

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

W_7124_Initial (ByVal card_number As Long, ByVal base_address As Long, ByVal Irq As Long) As Long

C/C++ (DOS)

int _7124_Initial(int card_number, int base_address, int irq)

@ 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.

Irq : The IRQ level of your ACL-7124 card. This irq value should be the same as hardware setting.

Note: Since Windows NT arranges 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 DrvUtil utility to set IRQ level before running application. Please refer to section 2.1.3 " Installation".

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError
ERR_InvalidIRQChannel

2.2.2 W_7124_Set_Card / _7124_Set_Card

@ Description

This function is used on multi-cards system. After the ACL-7124 cards are initialized by **W_7124_Initial** function, you can use this function to select which one you want to operate.

@ Syntax

Microsoft C/C++

int W_7124_Set_Card(int card_number)

Visual Basic

Windows 3.11 Version:

W_7124_Set_Card (ByVal card_number As Integer) As Integer

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

W_7124_Set_Card (ByVal card_number As Long) As Long

C/C++ (DOS)

int _7124_Set_Card(int card_number)

@ Argument

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

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber

2.2.3 W_7124_Get_Card / _7124_Get_Card

@ Description

This function is used on multi-cards system. You can use this function to get which card is operated currently.

@ Syntax

Microsoft C/C++

int W_7124_Get_Card(int *card_number)

Visual Basic

Windows 3.11 Version:

W_7124_Get_Card (card_number As Integer) As Integer

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

W_7124_Get_Card (card_number As Long) As Long

C/C++ (DOS)

int _7124_Get_Card(int *card_number)

@ Argument

card_number : card identification.

@ Return Code

ERR_NoError

ERR_BoardNoInit

2.2.4 W_7124_DI / _7124_DI

@ Description

This function is used to read data from digital input ports. There is one connector that supports digital I/O and this connector is divided into 3 ports, PortA, PortB, and PortC. The PortC can also be configured as PortC_Upper, and PortC_Lower. All of these ports can be set as input mode, i.e. read data from the ports.

@ Syntax

Microsoft C/C++

int W_7124_DI(int cn_port, unsigned char *di_data)

Visual Basic

Windows 3.11 Version:

W_7124_DI (ByVal cn_port As Integer, di_data As Integer) As Integer

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

W_7124_DI (ByVal cn_port As Long, di_data As Long) As Long

C/C++ (DOS)

int _7124_DI(int cn_port, unsigned char *di_data)

@ Argument

cn_port : To indicate which connector and its associated port is read, the valid data is :

CH0_PA	Prot A of Connector 0
CH0_PB	Port B of Connector 0
CH0_PC	Port C of Connector 0
CH0_PCU	Upper Port C of Connector 0
CH0_PCL	Low Port C of Connector 0

di_data : return value from digital port.

@ Return Code

ERR_NoError

ERR_BoardNoInit
ERR_PortError

2.2.5 W_7124_DO / _7124_DO

@ Description

This function is used to write data to digital output ports. There is one connector that supports digital I/O and this connector is divided into 3 ports, PortA, PortB, and PortC. The PortC can also be configured as PortC_Upper, and PortC_Lower. All of these ports can be set as output mode, i.e. write data to the ports.

@ Syntax

Microsoft C/C++

int W_7124_DO(int cn_port, unsigned char do_data)

Visual Basic

Windows 3.11 Version:

W_7124_DO (ByVal cn_port As Integer, ByVal do_data As Integer) As Integer

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

W_7124_DO (ByVal cn_port As Long, ByVal do_data As Long) As Long

C/C++ (DOS)

int _7124_DO(int cn_port, unsigned char do_data)

@ Argument

cn_port : To indicate which connector and its associated port is read, the valid data is :

CH0_PA	Prot A of Connector 0
CH0_PB	Port B of Connector 0
CH0_PC	Port C of Connector 0
CH0_PCU	Upper Port C of Connector 0
CH0_PCL	Low Port C of Connector 0

do_data : value that is written to digital output port

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ChannelError
ERR_PortError

2.2.6 W_7124_INTOP_Start / W_7124_INT_Start / _7124_INT_Start

@ Description

The function will perform digital input or output N times with interrupt data transfer by using external interrupt trigger. It takes place in the background which will not stop until the N-th conversion has completed or your program calls W_7124_INTOP_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_7124_INTOP_Status(). It can help you to make sure there is no interrupt operation being processed in background.

Before using this function, you have to define the operation of each connector's port. Three operation types can be chosen for each port : NO_OP, INPUT_OP, and OUTPUT_OP. The mode of each port is set by using the function W_7124_INT_Op(). (please refer to W_7124_INT_Op() function in section 2.2.10)

Note : The W_7124_INTOP_Start and W_7124_INTOP_Stop are a pair of functions. That is, as the W_7124_INTOP_Start is called, the W_7124_INTOP_Stop has to follow up behind it. Otherwise, the "Digital Input" data will not be stored in the buffer.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

int W_7124_INT_Start(int count)

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

int W_7124_INTOP_Start(int count)

Visual Basic

Windows 3.11 Version:

W_7124_INTOP_Start (ByVal count As Integer) As Integer

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

W_7124_INTOP_Start (ByVal count As Long) As Long

C/C++ (DOS)

int _7124_INT_Start(int count)

@ Argument

count : the number of input and output have to be operated by interrupt trigger.

@ Return Code

ERR_NoError
ERR_INTNotSet

@ Example

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

```
Dim dio_buf(1024) As Long
Dim cn_port As Long
Dim count As Long
Dim op As Long
Dim Ret As Long
        .
        .
        .
Ret = W_7124_INT_Reset( );
Ret = W_7124_INT_Op( cn_port, op, dio_buf, buf_cnt);
Ret = W_7124_INTOP_Start( count)
        .
        .
```

2.2.7 W_7124_INTOP_Status / W_7124_INT_Status / _7124_INT_Status

@ Description

Since the W_7124_INTOP_Start() function is executed in background, you can issue the function

W_7124_INTOP_Status() to check the status of the interrupt operation.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

int W_7124_INT_Status(int *status , int *count)

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

int W_7124_INTOP_Status(int *status , int *count)

Visual Basic

Windows 3.11 Version:

W_7124_INTOP_Status(status As Integer, count As Integer) As Integer

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

W_7124_INTOP_Status (status As Long, count As Long) As Long

C/C++ (DOS)

int _7124_INT_Status(int *status , int *count)

@ Argument

status : status of the INT data transfer
INT_STOP : Digital I/O INT is completed
INT_RUN: Digital I/O INT is not completed

count : current I/O operation count number.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.2.8 W_7124_INTOP_Stop / W_7124_INT_Stop / _7124_INT_Stop

@ Description

This function is used to stop the interrupt data transfer function. After executing this function, the digital I/O interrupt operation stops. The function returns the number of the digital I/O

operations which has been done, no matter whether the Digital I/O interrupt operation is stopped by this function or by the W_7124_INTOP_Start() itself.

Note : This function has to be called after the W_7124_INTOP_Start is called. Otherwise the input data will not be stored in the buffer.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

W_7124_INT_Stop(int *count)

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

int W_7124_INTOP_Stop(int *count)

Visual Basic

Windows 3.11 Version:

W_7124_INTOP_Stop(count As Integer) As Integer

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

W_7124_INTOP_Stop (count As Long) As Long

C/C++ (DOS)

int _7124_INT_Stop(int *count)

@ Argument

count : the number of digital I/O operations which have been done.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.2.9 W_7124_INT_Enable

@ Description

This function is only available in Window 95 driver and Windows NT/2000 driver. The function is used to initialize and start up the interrupt control. After calling this function, every time an interrupt

request signal 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 samples program 7124int.c.

Note : The W_7124_INT_Enable and W_7124_INT_Disable are a pair of functions. That is, as the W_7124_INT_Enable is called, the W_7124_INT_Disable has to follow up behind it. Otherwise, the interrupt operation will not stop.

@ Syntax

Microsoft C/C++

```
int W_7124_INT_Enable(HANDLE *hIntEvent)
```

Visual Basic

```
W_7124_INT_Enable (hIntEvent As Long) As Long
```

@ Argument

hIntEvent: The handle of the event for interrupt signals.

@ Return Code

ERR_NoError
ERR_INTNotSet

2.2.10 W_7124_INT_Disable

@ Description

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

Note : This function has to be called after the W_7124_INT_Enable is called.

@ Syntax

Microsoft C/C++

```
int W_7124_INT_Disable()
```

Visual Basic

```
W_7124_INT_Disable () As Long
```

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.2.11 W_7124_INT_Op / _7124_INT_Op

@ Description

This function is used to set up the operation of each port when the interrupt is triggered by external signals. Before the W_7124_INTOP_Start is executed, you have to set up the I/O status of each digital port that will perform interrupt data transfer.

Note : The Windows Version and DOS version are different. In Windows Version: the buffer size has to be specified in this function. But in DOS version, it does not need to be specified.

@ Syntax

Microsoft C/C++

int W_7124_INT_Op (int cn_port , int op, unsigned int *buf, int buf_cnt)

Visual Basic

Windows 3.11 Version:

W_7124_INT_Op (ByVal cn_port As Integer, ByVal op As Integer, buf As Integer, ByVal buf_cnt As Integer) As Integer

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

W_7124_INT_Op (ByVal cn_port As Long, ByVal op As Long, buf As Long, ByVal buf_cnt As Long) As Long

C/C++ (DOS)

int _7124_INT_Op (int cn_port , int op, unsigned int far *buf)

@ Argument

cn_port : To indicate which connector and its associated port, the valid data is :

CH0_PA	Prot A of Connector 0
CH0_PB	Port B of Connector 0
CH0_PC	Port C of Connector 0
CH0_PCU	Upper Port C of Connector 0
CH0_PCL	Low Port C of Connector 0

op : Digital I/O status

NO_OP : No Operation
INPUT_OP : Input Operation
OUTPUT_OP : Output Operation

buf : the start address of the memory buffer to write/read the digital output/input data, the buffer size must be larger than the number of digital I/O operations.

buf_cnt : the number of data in buffer.

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of buffer. 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

2.2.12 W_7124_INT_Reset / _7124_INT_Reset

@ Description

This function is used to reset all ports' interrupt transfer operations. After executing this function, each port is set as

NO_OP (no operation). That is, no digital I/O operation will be performed as an interrupt is occurred.

@ Syntax

Microsoft C/C++

int W_7124_INT_Reset( )

Visual Basic

Windows 3.11 Version:

W_7124_INT_Reset ( ) As Integer

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

W_7124_INT_Reset ( ) As Long

C/C++ (DOS)

int _7124_INT_Reset ( )

@ Argument

Not arguments.

@ Return Code

ERR_NoError

2.3 PET-48DIO Software Drivers

The PET-48DIO is a 48-bit Digital I/O card. All the 48 I/O bits are divided into 2 connectors, and each connector consists of three ports, PORTA, PORTB, and PORTC.

In addition, an 8253 Timer/Counter chip which offers time pacer generation and event counting capability is on board. More functions of Timer and Counter are supported in the software drivers.

There are 17 functions provided for PET-48DIO Digital I/O cards. The functions used in DOS and their corresponding functions used in Windows are specified in the following table.

Functions in DOS	Functions in Windows
_48DIO_Initial	W_48DIO_Initial

_48DIO_Set_Card	W_48DIO_Set_Card
_48DIO_Get_Card	W_48DIO_Get_Card
_48DIO_DO	W_48DIO_DO
_48DIO_DI	W_48DIO_DI
_48DIO_INT_Start	W_48DIO_INTOP_Start
_48DIO_INT_Status	W_48DIO_INTOP_Status
_48DIO_INT_Stop	W_48DIO_INTOP_Stop
---	W_48DIO_INT_Enable
---	W_48DIO_INT_Disable
_48DIO_INT_Op	W_48DIO_INT_Op
_48DIO_INT_Reset	W_48DIO_INT_Reset
_48DIO_Timer_Start	W_48DIO_INT_Timer_Start
_48DIO_Timer_Stop	W_48DIO_INT_Timer_Stop
_48DIO_Count_Start	W_48DIO_Count_Start
_48DIO_Count_Status	W_48DIO_Count_Status
_48DIO_Count_Stop	W_48DIO_Count_Stop

The detailed description of each function is specified in the following sections.

2.3.1 W_48DIO_Initial / _48DIO_Initial

@ Description

An PET -48DIO card is initialized according to the card number, its corresponding base address and IRQ level. Every PET -48DIO Digital I/O card has to be initialized by this function before calling other functions.

@ Syntax

Microsoft C/C++

int W_48DIO_Initial(int card_number, int base_address, int irq)

Visual Basic

Windows 3.11 Version:

W_48DIO_Initial (ByVal card_number As Integer, ByVal base_address As Integer, ByVal irq As Integer) As Integer

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

W_48DIO_Initial (ByVal card_number As Long, ByVal base_address As Long, ByVal Irq As Long) As Long

C/C++ (DOS)

int _48DIO_Initial(int card_number, int base_address, int irq)

@ 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.

Irq : The IRQ level of your ACL-48DIO card. This irq value should be the same as hardware setting.

Note: Since Windows NT arranges 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 DrvUtil utility to set IRQ level before running application. Please refer to section 2.4 "Some Installation Issues in Win-NT".

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError
ERR_InvalidIRQChannel

2.3.2 W_48DIO_Set_Card / _48DIO_Set_Card

@ Description

This function is used on multi-cards system. After the PET - 48DIO cards are initialized by **W_48DIO_Initial** function, you can use this function to select which one you want to operate.

@ Syntax

Microsoft C/C++

int W_48DIO_Set_Card(int card_number)

Visual Basic

Windows 3.11 Version:

W_48DIO_Set_Card (ByVal card_number As Integer) As Integer

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

W_48DIO_Set_Card (ByVal card_number As Long) As Long

C/C++ (DOS)

int _48DIO_Set_Card(int card_number)

@ Argument

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

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber

2.3.3 W_48DIO_Get_Card / _48DIO_Get_Card

@ Description

This function is used on multi-cards system. You can use this function to get which card is operated currently.

@ Syntax

Microsoft C/C++

int W_48DIO_Get_Card(int *card_number)

Visual Basic

Windows 3.11 Version:

W_48DIO_Get_Card (card_number As Integer) As Integer

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

W_48DIO_Get_Card (card_number As Long) As Long

C/C++ (DOS)

int _48DIO_Get_Card(int *card_number)

@ Argument

card_number : card identification.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.3.4 W_48DIO_DI / _48DIO_DI

@ Description

This function is used to read data from digital input ports. There is one connector that supports digital I/O and this connector is divided into 3 ports, PortA, PortB, and PortC. The PortC can also be configured as PortC_Upper, and PortC_Lower. All of these ports can be set as input mode, i.e. read data from the ports.

@ Syntax

Microsoft C/C++

int W_48DIO_DI(int cn_port, unsigned char *di_data)

Visual Basic

Windows 3.11 Version:

W_48DIO_DI (ByVal cn_port As Integer, di_data As Integer) As Integer

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

W_48DIO_DI (ByVal cn_port As Long, di_data As Long) As Long

C/C++ (DOS)

int _48DIO_DI(int cn_port, unsigned char *di_data)

@ Argument

cn_port : To indicate which connector and its associated port is read, the valid data is :

CH0_PA	Prot A of Connector 0
CH0_PB	Port B of Connector 0
CH0_PC	Port C of Connector 0
CH0_PCU	Upper Port C of Connector 0
CH0_PCL	Low Port C of Connector 0
CH1_PA	Prot A of Connector 1
CH1_PB	Port B of Connector 1
CH1_PC	Port C of Connector 1

CH1_PCU	Upper Port C of Connector 1
CH1_PCL	Low Port C of Connector 1

di_data : return value from digital port.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.3.5 W_48DIO_DO / _48DIO_DO

@ Description

This function is used to write data to digital output ports. There is one connector that supports digital I/O and this connector is divided into 3 ports, PortA, PortB, and PortC. The PortC can also be configured as PortC_Upper, and PortC_Lower. All of these ports can be set as output mode, i.e. write data to the ports.

@ Syntax

Microsoft C/C++

int W_48DIO_DO(int cn_port, unsigned char do_data)

Visual Basic

Windows 3.11 Version:

W_48DIO_DO (ByVal cn_port As Integer, ByVal do_data As Integer) As Integer

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

W_48DIO_DO (ByVal cn_port As Long, ByVal do_data As Long) As Long

C/C++ (DOS)

int _48DIO_DO(int cn_port, unsigned char do_data)

@ Argument

cn_port : To indicate which connector and its associated port is read, the valid data is :

CH0_PA	Prot A of Connector 0
CH0_PB	Port B of Connector 0

CH0_PC	Port C of Connector 0
CH0_PCU	Upper Port C of Connector 0
CH0_PCL	Low Port C of Connector 0
CH1_PA	Prot A of Connector 1
CH1_PB	Port B of Connector 1
CH1_PC	Port C of Connector 1
CH1_PCU	Upper Port C of Connector 1
CH1_PCL	Low Port C of Connector 1

do_data : value that is written to digital output port

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ChannelError
ERR_PortError

2.3.6 W_48DIO_INTOP_Start / W_48DIO_INT_Start / _48DIO_INT_Start

@ Description

The function will perform digital input or output N times with interrupt data transfer by using external interrupt trigger. It takes place in the background which will not stop until the N-th conversion has completed or your program calls W_48DIO_INTOP_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_48DIO_INTOP_Status(). It can help you to make sure there is no interrupt operation being processed in background.

Before using this function, you have to define the operation of each connector's port. Three operation types can be chosen for each port : NO_OP, INPUT_OP, and OUTPUT_OP. The mode of each port is set by using the function W_48DIO_INT_Op(). (please refer to W_48DIO_INT_Op() function in section 2.3.10)

Note : The W_48DIO_INTOP_Start and W_48DIO_INTOP_Stop are a pair of functions. That is, as the W_48DIO_INTOP_Start is called, the W_48DIO_INTOP_Stop has to follow up behind it. Otherwise, the "Digital Input" data will not be stored in the buffer.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

int W_48DIO_INT_Start(int count)

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

int W_48DIO_INTOP_Start(int count)

Visual Basic

Windows 3.11 Version:

W_48DIO_INTOP_Start (ByVal count As Integer) As Integer

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

W_48DIO_INTOP_Start (ByVal count As Long) As Long

C/C++ (DOS)

int _48DIO_INT_Start(int count)

@ Argument

count : the number of input and output have to be operated by interrupt trigger.

@ Return Code

ERR_NoError
ERR_INTNotSet

@ Example

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

```
Dim dio_buf(1024) As Long
Dim cn_port As Long
Dim count As Long
Dim op As Long
Dim Ret As Long
.
```

```
        .
        .
Ret = W_48DIO_INT_Reset( );
Ret = W_48DIO_INT_Op( cn_port, op, dio_buf, buf_cnt);
Ret = W_48DIO_INTOP_Start( count)
        .
        .
```

2.3.7 W_48DIO_INTOP_Status / W_48DIO_INT_Status / _48DIO_INT_Status

@ Description

Since the W_48DIO_INTOP_Start() function is executed in background, you can issue the function W_48DIO_INTOP_Status() to check the status of the interrupt operation.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

```
int W_48DIO_INT_Status( int *status , int *count )
```

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

```
int W_48DIO_INTOP_Status( int *status , int *count )
```

Visual Basic

Windows 3.11 Version:

```
W_48DIO_INTOP_Status(status As Integer, count As
Integer) As Integer
```

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

```
W_48DIO_INTOP_Status (status As Long, count As Long)
As Long
```

C/C++ (DOS)

```
int _48DIO_INT_Status( int *status , int *count )
```

@ Argument

status : status of the INT data transfer
INT_STOP : Digital I/O INT is completed
INT_RUN: Digital I/O INT is not completed

count : current I/O operation count number.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.3.8 W_48DIO_INTOP_Stop / W_48DIO_INT_Stop / _48DIO_INT_Stop

@ Description

This function is used to stop the interrupt data transfer function. After executing this function, the digital I/O interrupt operation stops. The function returns the number of the digital I/O operations which has been done, no matter whether the Digital I/O interrupt operation is stopped by this function or by the W_48DIO_INTOP_Start() itself.

Note : This function has to be called after the W_48DIO_INTOP_Start is called. Otherwise the input data will not be stored in the buffer.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

W_48DIO_INT_Stop(int *count)

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

int W_48DIO_INTOP_Stop(int *count)

Visual Basic

Windows 3.11 Version:

W_48DIO_INTOP_Stop(count As Integer) As Integer

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

W_48DIO_INTOP_Stop (count As Long) As Long

C/C++ (DOS)

int _48DIO_INT_Stop(int *count)

@ Argument

count : the number of digital I/O operations which have been done.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.3.9 W_48DIO_INT_Enable

@ Description

This function is only available in Window 95 driver and Windows NT/2000 driver. The function is used to initialize and start up the interrupt control. After calling this function, every time an interrupt request signal 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 samples program 48dioint.c.

Note : The W_48DIO_INT_Enable and W_48DIO_INT_Disable are a pair of functions. That is, as the W_48DIO_INT_Enable is called, the W_48DIO_INT_Disable has to follow up behind it. Otherwise, the interrupt operation will not stop.

@ Syntax

Microsoft C/C++

```
int W_48DIO_INT_Enable(HANDLE *hIntEvent)
```

Visual Basic

```
W_48DIO_INT_Enable (hIntEvent As Long) As Long
```

@ Argument

hIntEvent: the handle of the event for interrupt signals.

@ Return Code

ERR_NoError
ERR_INTNotSet

2.3.10 W_48DIO_INT_Disable

@ Description

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

Note : This function has to be called after the W_48DIO_INT_Enable is called.

@ Syntax

Microsoft C/C++

```
int W_48DIO_INT_Disable()
```

Visual Basic

```
W_48DIO_INT_Disable () As Long
```

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.3.11 W_48DIO_INT_Op / _48DIO_INT_Op

@ Description

This function is used to set up the operation of each port when the interrupt is triggered. Before the W_48DIO_INTOP_Start is executed, you have to set up the I/O status of each digital port that will perform interrupt data transfer.

Note : The Windows Version and DOS version are different. In Windows Version: the buffer size has to be specified in this function. But in DOS version, it does not need to be specified.

@ Syntax

Microsoft C/C++

int W_48DIO_INT_Op (int cn_port , int op, unsigned int *buf, int buf_cnt)

Visual Basic

Windows 3.11 Version:

W_48DIO_INT_Op (ByVal cn_port As Integer, ByVal op As Integer, buf As Integer, ByVal buf_cnt As Integer) As Integer

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

W_48DIO_INT_Op (ByVal cn_port As Long, ByVal op As Long, buf As Long, ByVal buf_cnt As Long) As Long

C/C++ (DOS)

int _48DIO_INT_Op (int cn_port , int op, unsigned int far *buf)

@ Argument

cn_port : To indicate which connector and its associated port, the valid data is :

CH0_PA	Prot A of Connector 0
CH0_PB	Port B of Connector 0
CH0_PC	Port C of Connector 0
CH0_PCU	Upper Port C of Connector 0
CH0_PCL	Low Port C of Connector 0
CH1_PA	Prot A of Connector 1
CH1_PB	Port B of Connector 1
CH1_PC	Port C of Connector 1
CH1_PCU	Upper Port C of Connector 1
CH1_PCL	Low Port C of Connector 1

op : Digital I/O status

NO_OP : No Operation
INPUT_OP : Input Operation
OUTPUT_OP : Output Operation

buf : the start address of the memory buffer to

write/read the digital output/input data, the buffer size must be larger than the number of digital I/O operations.

buf_cnt : the number of data in buffer.

Note : While calling this function in Visual Basic program, please pass the first element of the buffer array as the argument of buffer. 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

2.3.12 W_48DIO_INT_Reset / _48DIO_INT_Reset

@ Description

This function is used to reset all ports' interrupt transfer operations. After executing this function, each port is set as NO_OP (no operation). That is, no digital I/O operation will be performed as an interrupt is occurred.

@ Syntax

Microsoft C/C++

```
int W_48DIO_INT_Reset( )
```

Visual Basic

Windows 3.11 Version:

```
W_48DIO_INT_Reset ( ) As Integer
```

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

```
W_48DIO_INT_Reset ( ) As Long
```

C/C++ (DOS)

```
int _48DIO_INT_Reset ( )
```

@ Argument

Not arguments.

@ Return Code

ERR_NoError

2.3.13 W_48DIO_INT_Timer_Start / W_48DIO_Timer_Start /_48DIO_Timer_Start

@ Description

This function is used to set up the Timer #1 and Timer #2. Timer #1 & #2 are used as frequency dividers for generating constant clock pacer for interrupt trigger dedicatedly.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

int W_48DIO_Timer_Start(unsigned int c1 , unsigned int c2)

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

int W_48DIO_INT_Timer_Start(unsigned int c1 , unsigned int c2)

Visual Basic

Windows 3.11 Version:

W_48DIO_Timer_Start(ByVal c1 As Integer, ByVal c2 As integer) As Integer

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

W_48DIO_INT_Timer_Start(ByVal c1 As Long, ByVal c2 As Long) As Long

C/C++ (DOS)

int _48DIO_Timer_Start(unsigned int c1, unsigned int c2)

@ Argument

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

Two timer source are supported in PET-48DIO card. One is 2MHz and the other is 32.768KHz. The Clock pacer rate is : 2MHz/ (C1 * C2) or 32.768HHz / (C1 * C2). The timer source is selected by jumper setting.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerValue

2.3.14 W_48DIO_INT_Timer_Stop / W_48DIO_Timer_Stop / _48DIO_Timer_Stop

@ Description

This function is used to stop the timer operation.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

Int W_48DIO_Timer_Stop( )

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

Int W_48DIO_INT_Timer_Stop( )

Visual Basic

Windows 3.11 Version:

W_48DIO_Timer_Stop( ) As Integer

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

W_48DIO_INT_Timer_Stop( ) As Long

C/C++ (DOS)

int _48DIO_Timer_Stop( )

@ Argument

No arguments

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.3.15 W_48DIO_Count_Start /_48DIO_Count_Start

@ Description

The counter #0 of the PET-48DIO's Timer/Counter chip can be freely programmed by the users. This function is used to program the counter #0. This counter can be used as frequency generator if internal clock is used. It can also be used as event counter if external clock is used. All the 8253 mode is available. Please refer to "Timer/Counter 8253" in PET-48DIO user's manual.

@ Syntax

Microsoft C/C++

```
int W_48DIO_Count_Start( int timer_mode, unsigned int c0
)
```

Visual Basic

Windows 3.11 Version:

```
W_48DIO_Count_Start (ByVal timer_mode As Integer,
ByVal c0 As Integer) As Integer
```

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

```
W_48DIO_Count_Start (ByVal timer_mode As Long, ByVal
c0 As Long) As Long
```

C/C++ (DOS)

```
int _48DIO_Count_Start( int timer_mode, unsigned int c0 )
```

@ Argument

timer_mode : the 8253 timer mode, the possible values are :
TIMER_MODE0, TIMER_MODE1,
TIMER_MODE2, TIMER_MODE3,
TIMER_MODE4, TIMER_MODE5.
Please refer to the manual or reference books of Counter/Timer 8253 for more detailed information about timer mode.

c0 : the count value of timer

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerMode
ERR_InvalidTimerValue

2.3.16 W_48DIO_Count_Status /_48DIO_Count_Status

@ Description

This function is used to read the count value of the Counter #0.

@ Syntax

Microsoft C/C++

```
int  W_48DIO_Count_Status( unsigned int *counter_value )
```

Visual Basic

Windows 3.11 Version:

W_48DIO_Count_Status(counter_value As Integer) As Integer

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

W_48DIO_Count_Status(counter_value As Long) As Long

C/C++ (DOS)

```
int  _48DIO_Count_Status( unsigned int *counter_value )
```

@ Argument

counter_value : the count value of the Counter #0

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.3.17 W_48DIO_Count_Stop/_48DIO_Count_Stop

@ Description

This function is used to stop the event counting 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_48DIO_Count_Stop( unsigned int *counter_value )
```

Visual Basic

Windows 3.11 Version:

W_48DIO_Count_Stop (counter_value As Integer) As Integer

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

W_48DIO_Count_Stop (counter_value As Long) As Long

C/C++ (DOS)

int _48DIO_Count_Stop(unsigned int *counter_value)

@ Argument

counter_value : the current count value of the Counter #0

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.4 ACL-7120 Software Drivers

The ACL-7120 board consists of 32 digital inputs and 32 digital outputs.

Note : All functions of the ACL-7120 can be applied to the ACL-720 directly.

Two types of ACL-7120 drivers are provided: C Language library for DOS and DLL driver for Windows.

There are 15 functions provided for ACL-7120 Digital I/O cards. The functions used in DOS and their corresponding functions used in Windows are specified in the following table.

Functions in DOS	Functions in Windows
_7120_Initial	W_7120_Initial
_7120_Set_Card	W_7120_Set_Card
_7120_Get_Card	W_7120_Get_Card
_7120_D0_8	W_7120_D0_8
_7120_DO_16	W_7120_DO_16
_7120_DI_Channel	W_7120_DI_Channel
_7120_DI_8	W_7120_DI_8

_7120_DI_16	W_7120_DI_16
---	W_7120_INT_Enable
---	W_7120_INT_Disable
---	W_7120_INT_Timer_Start
---	W_7120_INT_Timer_Stop
_7120_ Timer _Start	W_7120_Timer_Start
_7120_ Timer _Read	W_7120_Timer_Read
_7120_ Timer _Stop	W_7120_Timer_Stop

The detailed description of each function is specified in the following sections.

2.4.1 W_7120_Initial /_7120_Initial

@ Description

An ACL-7120 card is initialized according to the card number and its corresponding base address. Every ACL-7120 card has to be initialized by this function before calling other functions. Up to 8 cards can be initialized in the same system.

@ Syntax

Microsoft C/C++

int W_7120_Initial(int card_number, int base_address, int irq)

Visual Basic

Windows 3.11 Version:

W_7120_Initial (ByVal card_number As Integer, ByVal base_address As Integer) As Integer

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

W_7120_Initial (ByVal card_number As Long, ByVal base_address As Long, ByVal irq As Long) As Long

C/C++ (DOS)

int _7120_Initial(int card_number, int base_address)

@ 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 to 7.

base_address : The I/O port base address of the card. Please refer to user's manual of ACL-7120 for I/O base address setting.

Irq : The IRQ level of your ACL-7120 card. This irq value should be the same as hardware setting.

Note: Since Windows NT arranges 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 DrvUtil utility to set IRQ level before running application. Please refer to section 2.4 "Some Installation Issues in Win-NT".

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError

2.4.2 W_7120_Set_Card / _7120_Set_Card

@ Description

This function is used on multi-cards system. After the ACL-7120 cards are initialized by W_7120_Initial() function, you can use this function to select which one you want to operate.

Note : Although up to 8 cards can be initialized in one system, but only one card can be operated at the same time. i.e. you have to choose which card you want to operate through W_7120_Set_Card function.

@ Syntax

Microsoft C/C++

```
int W_7120_Set_Card(int card_number)
```

Visual Basic

Windows 3.11 Version:

W_7120_Set_Card (ByVal card_number As Integer) As Integer

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

W_7120_Set_Card (ByVal card_number As Long) As Long

C/C++ (DOS)

int _7120_Set_Card(int card_number)

@ 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

2.4.3 W_7120_Get_Card / _7120_Get_Card

@ Description

This function is used on multi-cards system. You can use this function to get which card is operated currently.

@ Syntax

Microsoft C/C++

int W_7120_Get_Card(int *card_number)

Visual Basic

Windows 3.11 Version:

W_7120_Get_Card (card_number As Integer) As Integer

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

W_7120_Get_Card (card_number As Long) As Long

C/C++ (DOS)

int _7120_Get_Card(int *card_number)

@ Argument

card_number : card identification.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.4.4 W_7120_DI_Channel / _7120_DI_Channel

@ Description

This function is used to read data from digital input channels. 32 input channels are provided in the ACL-7120 card. By using this function, you can get the status of each input channel in this card.

Note : Channel is defined as one bit of input or output unit.

@ Syntax

Microsoft C/C++

int W_7120_DI_Channel(int channel_no, unsigned int *di_data)

Visual Basic

Windows 3.11 Version:

W_7120_DI_Channel (ByVal channel_no As Integer, di_data As Integer) As Integer

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

W_7120_DI_Channel (ByVal channel_no As Long, di_data As Long) As Integer

C/C++ (DOS)

int _7120_DI_Channel(int channel_no, unsigned int *di_data)

@ Argument

channel_no : Indicate which channel is read, the valid number is from 0 to 31. (ACL-7120 has 32 inputs)

di_data : Return value from input channel. The value is either 0 or 1.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDIChannel

2.4.5 W_7120_DI_16 / _7120_DI_16

@ Description

This function is used to read data from digital input port. Two 20-pin head connectors are provided in ACL-7120 card, each connector consists of 16 channels.

@ Syntax

Microsoft C/C++

int W_7120_DI_16(int port, unsigned int *di_data)

Visual Basic

Windows 3.11 Version:

W_7120_DI_16(ByVal port As Integer, di_data As Integer) As Integer

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

W_7120_DI_16(ByVal port As Long, di_data As Long) As Long

C/C++ (DOS)

int _7120_DI_16(int port, unsigned int *di_data)

@ Argument

port : port number to read input data, the value is:

DI_PORT0	channel 0 ... 15
DI_PORT1	channel 16 ... 31

di_data : value that is read from input port, the value is from 0 to 65535.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ChannelError

2.4.6 W_7120_DI_8 / _7120_DI_8

@ Description

This function is used to read data from digital inputs. The ACL-7120's 32 input channels can be grouped into four bytes. You can read data from each byte through this function.

@ Syntax

Microsoft C/C++

int W_7120_DI_8(int byte_no, unsigned char *di_data)

Visual Basic

Windows 3.11 Version:

W_7120_DI_8(ByVal byte_no As Integer, di_data As Integer) As Integer

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

W_7120_DI_8(ByVal byte_no As Long, di_data As Long) As Long

C/C++ (DOS)

int _7120_DI_8(int byte_no, unsigned char *di_data)

@ Argument

byte_no : The byte of inputs, the relationship between the byte_no and its correspond input channels is:

DI_BYTE1 :	channel 0 ... 7
DI_BYTE2 :	channel 8 ... 15
DI_BYTE3 :	channel 16 ... 23
DI_BYTE4 :	channel 24 ... 31

di_data : value is read from inputs in every 8-bits, the value is within 0 and 255

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.4.7 W_7120_DO_16 / _7120_DO_16

@ Description

This function is used to write data to digital output ports. Two ports are provided in ACL-7120 card, DO_PORT1 and DO_PORT2, each port consists of 16 channels

@ Syntax

Microsoft C/C++

int W_7120_DO_16(int port, unsigned int do_data)

Visual Basic

Windows 3.11 Version:

W_7120_DO_16(ByVal port As Integer, ByVal do_data As Integer) As Integer

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

W_7120_DO_16(ByVal port As Long, ByVal do_data As Long) As Long

C/C++ (DOS)

int _7120_DO_16(int port, unsigned int do_data)

@ Argument

port : port number that output data is written to, the valid value is:

DO_PORT1	channel 0 ... 15
DO_PORT2	channel 16 ... 31

do_data : value is written to output port, the valid value is from 0 to 65535

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ChannelError

2.4.8 W_7120_DO_8 / _7120_DO_8

@ Description

This function is used to write data to digital outputs. 32 outputs of the ACL-7120 card can be grouped into four bytes. You can write data to each byte through this function.

@ Syntax

Microsoft C/C++

int W_7120_DO_8(int byte_no, unsigned char do_data)

Visual Basic

Windows 3.11 Version:

W_7120_DO_8(ByVal byte_no As Integer, ByVal do_data As Integer) As Integer

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

W_7120_DO_8(ByVal byte_no As Long, ByVal do_data As Long) As Long

C/C++ (DOS)

int _7120_DO_8(int byte_no, unsigned char do_data)

@ Argument

byte_no : The byte of outputs, the relationship between the byte_no and its correspond output channels is :

DO_BYTE1	channel 0 ... 7
DO_BYTE2	channel 8 ... 15
DO_BYTE3	channel 16 ... 23
DO_BYTE4	channel 24 ... 31

do_data: value written to outputs every 8-bits, the valid value is from 0 to 255

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.4.9 W_7120_INT_Enable

@ Description

This function is only available in Window 95 driver and Windows NT/2000 driver. The function is used to initialize and start up the interrupt control. After calling this function, every time an interrupt request signal 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 samples program 7120int.c.

Note : The W_7120_INT_Enable and W_7120_INT_Disable are a pair of functions. That is, as the W_7120_INT_Enable is called, the W_7120_INT_Disable has to follow up behind it. Otherwise, the interrupt operation will not stop.

@ Syntax

Microsoft C/C++

```
int W_7120_INT_Enable(HANDLE *hIntEvent)
```

Visual Basic

```
W_7120_INT_Enable (hIntEvent As Long) As Long
```

@ Argument

hIntEvent: The handle of the event for interrupt signals.

@ Return Code

ERR_NoError
ERR_INTNotSet

2.4.10 W_7120_INT_Disable

@ Description

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

Note : This function has to be called after the W_7120_INT_Enable is called.

@ Syntax

Microsoft C/C++

int W_7120_INT_Disable()

Visual Basic

W_7120_INT_Disable () As Long

@ Argument

None

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.4.11 W_7120_INT_Timer_Start / W_7120_Timer_Start /_7120_Timer_Start

@ Description

This function is used to set up the Timer #4 and Timer #5. Timer #4 & #5 are used as frequency dividers for generating constant clock pacer for interrupt trigger dedicatedly.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

int W_7120_Timer_Start(unsigned int c1 , unsigned int c2)

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

int W_7120_INT_Timer_Start(unsigned int c1 , unsigned int c2)

Visual Basic

Windows 3.11 Version:

W_7120_Timer_Start(ByVal c1 As Integer, ByVal c2 As integer) As Integer

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

W_7120_INT_Timer_Start(ByVal c1 As Long, ByVal c2 As Long) As Long

C/C++ (DOS)

int _7120_Timer_Start(unsigned int c1, unsigned int c2)

@ Argument

c1 : frequency divider of timer #4
c2 : frequency divider of timer #5,

Two timer source are supported in ACL-7120 card. One is 2MHz and the other is 32.768KHz. The Clock pacer rate is :
2MHz/ (C1 * C2) or 32.768HHz / (C1 * C2). The timer source is selected by jumper setting.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerValue

2.4.12 W_7120_INT_Timer_Stop / W_7120_Timer_Stop / _7120_Timer_Stop

@ Description

This function is used to stop the internal interrupt timer operation.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

Int W_7120_Timer_Stop()

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

Int W_7120_INT_Timer_Stop()

Visual Basic

Windows 3.11 Version:

W_7120_Timer_Stop() As Integer

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

W_7120_INT_Timer_Stop() As Long

C/C++ (DOS)

int _7120_Timer_Stop()

@ Argument

No arguments

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.4.13 W_7120_Timer_Start /_7120_Timer_Start

@ Description

The counters of the ACL-7120's Timer/Counter chip can be freely programmed by the users. This function is used to program the counters. This counter can be used as frequency generator if internal clock is used. It can also be used as event counter if external clock is used. All the 8253 mode is available. Please refer to "Timer/Counter 8253" in ACL-7120 user's manual.

@ Syntax

Microsoft C/C++

int W_7120_Timer_Start(int timer_no, int timer_mode, unsigned int c0)

Visual Basic

Windows 3.11 Version:

W_7120_Timer_Start (ByVal timer_no As Integer, ByVal timer_mode As Integer, ByVal c0 As Integer) As Integer

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

W_7120_Timer_Start (ByVal timer_no As Integer, ByVal timer_mode As Long, ByVal c0 As Long) As Long

C/C++ (DOS)

int _7120_Timer_Start(ByVal timer_no As Integer, int timer_mode, unsigned int c0)

@ Argument

timer_no : the timer number will be started . the valid number are from 0 to 5.

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

TIMER_MODE4, TIMER_MODE5.
Please refer to the manual or reference books of Counter/Timer 8253 for more detailed information about timer mode.

c0 : the count value of timer

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerMode
ERR_InvalidTimerValue

2.4.14 W_7120_Timer_Read /_7120_Timer_Read

@ Description

This function is used to read the counter value of the Counter.

@ Syntax

Microsoft C/C++

```
int  W_7120_Timer_Read( int timer_no, unsigned int
*counter_value )
```

Visual Basic

Windows 3.11 Version:

```
W_7120_Timer_Read(ByVal timer_no As Integer,
counter_value As Integer) As Integer
```

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

```
W_7120_Timer_Read (ByVal timer_no As Integer,
counter_value As Long) As Long
```

C/C++ (DOS)

```
int  _7120_Timer_Read ( unsigned int timer_no, unsigned
int *counter_value )
```

@ Argument

timer_no : the timer number will be started . the valid number are from 0 to 5.

counter_value : the counter value of the Counter

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.4.15 W_7120_Timer_Stop/_7120_Timer_Stop

@ Description

This function is used to stop the event counting operation. That is, the clock output signal will be set to high after executing this function.

@ Syntax

Microsoft C/C++

int W_7120_Timer_Stop(int timer_no, unsigned int *counter_value)

Visual Basic

Windows 3.11 Version:

W_7120_Timer_Stop (ByVal timer_no As Integer, counter_value As Integer) As Integer

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

W_7120_Timer_Stop (ByVal timer_no As Integer, counter_value As Long) As Long

C/C++ (DOS)

int _7120_Timer_Stop(unsigned timer_no, unsigned int *counter_value)

@ Argument

timer_no : the timer number will be started . the valid number are from 0 to 5.

counter_value : the current counter value of the Counter

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.5 ACL-7125 Software Drivers

The ACL-7125 consists of 8 opto-isolated digital inputs and 8 relay digital outputs.

Note : All functions of the ACL-7125 can be applied to the ACL-725 directly.

Two types of ACL-7125 drivers are provided : C Language library for DOS and DLL driver for Windows.

There are 8 functions provided for ACL-7125 Digital I/O cards. The functions used in DOS and their corresponding functions used in Windows are specified in the following table.

Functions in DOS	Functions in Windows
_7125_Initial	W_7125_Initial
_7125_Set_Card	W_7125_Set_Card
_7125_Get_Card	W_7125_Get_Card
_7125_DO_Channel	W_7125_DO_Channel
_7125_DO_8	W_7125_DO
_7125_DI_Channel	W_7125_DI_Channel
_7125_DI_8	W_7125_DI
_7125_Read_Back	W_7125_Read_Back

2.5.1 W_7125_Initial /_7125_Initial

@ Description

An ACL-7125 card is initialized according to the card number and its corresponding base address. Every ACL-7125 card has to be initialized by this function before calling other functions. Up to 8 cards can be initialized in the same system.

@ Syntax

Microsoft C/C++

```
int W_7125_Initial(int card_number, int base_address)
```

Visual Basic

Windows 3.11 Version:

W_7125_Initial (ByVal card_number As Integer, ByVal base_address As Integer) As Integer

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

W_7125_Initial (ByVal card_number As Long, ByVal base_address As Long) As Long

C/C++ (DOS)

int _7125_Initial(int card_number, int base_address)

@ 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 to 7.

base_address : The I/O port base address of the card. Please refer to user's manual of ACL-7125 for I/O base address setting.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError

2.5.2 W_7125_Set_Card / _7125_Set_Card

@ Description

This function is used on multi-cards system. After the ACL-7125 cards are initialized by W_7125_Initial() function, you can use this function to select which one you want to operate.

Note : Although up to 8 cards can be initialized in one system, but only one card can be operated at the same time. i.e. you have to choose which card you want to operate through W_7125_Set_Card function.

@ Syntax

Microsoft C/C++

int W_7125_Set_Card(int card_number)

Visual Basic

Windows 3.11 Version:

W_7125_Set_Card (ByVal card_number As Integer) As Integer

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

W_7125_Set_Card (ByVal card_number As Long) As Long

C/C++ (DOS)

int _7125_Set_Card(int card_number)

@ 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

2.5.3 W_7125_Get_Card / _7125_Get_Card

@ Description

This function is used on multi-cards system. You can use this function to get which card is operated currently.

@ Syntax

Microsoft C/C++

int W_7125_Get_Card(int *card_number)

Visual Basic

Windows 3.11 Version:

W_7125_Get_Card (card_number As Integer) As Integer

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

W_7125_Get_Card (card_number As Long) As Long

C/C++ (DOS)

int _7125_Get_Card(int *card_number)

@ Argument

card_number : card identification.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.5.4 W_7125_DI_Channel / _7125_DI_Channel

@ Description

This function is used to read data from digital input channels. 8 input channels are provided in the ACL-7125 card. By using this function, you can get the status of each input channel in this card.

Note : Channel is defined as one bit of input or output unit.

@ Syntax

Microsoft C/C++

int W_7125_DI_Channel(int channel_no, unsigned int *di_data)

Visual Basic

Windows 3.11 Version:

W_7125_DI_Channel (ByVal channel_no As Integer, di_data As Integer) As Integer

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

W_7125_DI_Channel (ByVal channel_no As Long, di_data As Long) As Integer

C/C++ (DOS)

int _7125_DI_Channel(int channel_no, unsigned int *di_data)

@ Argument

channel_no : Indicate which channel is read, the valid number is from 0 to 7. (ACL-7125 has 8 inputs)

di_data : Return value from input channel. The value is either 0 or 1.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDIChannel

2.5.5 W_7125_DI / _7125_DI

@ Description

This function is used to read data from digital inputs.

@ Syntax

Microsoft C/C++

```
int W_7125_DI( unsigned char *di_data )
```

Visual Basic

Windows 3.11 Version:

```
W_7125_DI( di_data As Integer) As Integer
```

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

```
W_7125_DI( di_data As Long) As Long
```

C/C++ (DOS)

```
int _7125_DI( unsigned char *di_data )
```

@ Argument

di_data : value is read from input port, the value is within 0 and 255

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.5.6 W_7125_DO / _7125_DO

@ Description

This function is used to write data to digital outputs.

@ Syntax

Microsoft C/C++

int W_7125_DO(unsigned char do_data)

Visual Basic

Windows 3.11 Version:

W_7125_DO(ByVal do_data As Integer) As Integer

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

W_7125_DO(ByVal do_data As Long) As Long

C/C++ (DOS)

int _7125_DO(unsigned char do_data)

@ Argument

do_data: value written to output port, the valid value is from 0 to 255

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.5.7 W_7125_Read_Back / _7125_ Read_Back

@ Description

This function is used to read back the signals from digital output port.

@ Syntax

Microsoft C/C++

int W_7125_ Read_Back (unsigned char *do_data)

Visual Basic

Windows 3.11 Version:

W_7125_Read_Back (do_data As Integer) As Integer

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

W_7125_Read_Back (do_data As Long) As Long

C/C++ (DOS)

int _7125_Read_Back (unsigned char *do_data)

@ Argument

do_data: value written to outputs from digital output port, the valid value is from 0 to 255

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.5.8 W_7125_DO_Channel /_7125_DO_Channel

@ Description

This function is used to write data to the digital output channels. By using this function, you can control the status of each output channel.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

int W_7125_DO_Channel (int channel_no, unsigned char do_data)

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

int W_7125_ DO_Channel (int channel_no, unsigned char do_data)

Visual Basic

Windows 3.11 Version:

W_7125_ DO_Channel (ByVal channel_no As Integer, ByVal do_data As integer) As Integer

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

W_7125_ DO_Channel (ByVal channel_no As Long, ByVal do_data As Long) As Long

C/C++ (DOS)

int _7125_ DO_Channel (unsigned int channel_no, unsigned char do_data)

@ Argument

channel_no : indicate which channel is written, the valid data is from 0 to 7

do_data : the value of output channel. The value is either o or 1

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDOChannel
ERR_InvalidChangeValue

2.6 ACL-7225 Software Drivers

The ACL-7225 consists of 16 opto-isolated digital inputs and 16 relay digital outputs.

Note : All functions of the ACL-7225 can be applied to the ACL-725B directly.

Two types of drivers are supported for ACL-7225, one is C language library for DOS environment, the other is DLL driver for Windows, Win 95 or Windows NT/2000.

There are 8 functions provided for ACL-7225 Digital I/O cards. The functions used in DOS and their corresponding functions used in Windows are specified in the following table.

Functions in DOS	Functions in Windows
_7225_Initial	W_7225_Initial
_7225_Set_Card	W_7225_Set_Card
_7225_Get_Card	W_7225_Get_Card
_7225_DO_Channel	W_7225_DO_Channel
_7225_D0_8	W_7225_DO
_7225_DI_Channel	W_7225_DI_Channel
_7225_DI_8	W_7225_DI
_7225_Read_Back	W_7225_Read_Back

2.6.1 W_7225_Initial /_7225_Initial

@ Description

An ACL-7225 card is initialized according to the card number and its corresponding base address. Every ACL-7225 card has to be initialized by this function before calling other functions. Up to 8 cards can be initialized in the same system.

@ Syntax

Microsoft C/C++

```
int W_7225_Initial(int card_number, int base_address)
```

Visual Basic

Windows 3.11 Version:

W_7225_Initial (ByVal card_number As Integer, ByVal base_address As Integer) As Integer

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

W_7225_Initial (ByVal card_number As Long, ByVal base_address As Long) As Long

C/C++ (DOS)

int _7225_Initial(int card_number, int base_address)

@ 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 to 7.

base_address : The I/O port base address of the card. Please refer to user's manual of ACL-7225 for I/O base address setting.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError

2.6.2 W_7225_Set_Card / _7225_Set_Card

@ Description

This function is used on multi-cards system. After the ACL-7225 cards are initialized by W_7225_Initial() function, you can use this function to select which one you want to operate.

Note : Although up to 8 cards can be initialized in one system, but only one card can be operated at the same time. i.e. you have to choose which card you want to operate through W_7225_Set_Card function.

@ Syntax

Microsoft C/C++

int W_7225_Set_Card(int card_number)

Visual Basic

Windows 3.11 Version:

W_7225_Set_Card (ByVal card_number As Integer) As Integer

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

W_7225_Set_Card (ByVal card_number As Long) As Long

C/C++ (DOS)

int _7225_Set_Card(int card_number)

@ 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_7225_Get_Card / _7225_Get_Card

@ Description

This function is used on multi-cards system. You can use this function to get which card is operated currently.

@ Syntax

Microsoft C/C++

int W_7225_Get_Card(int *card_number)

Visual Basic

Windows 3.11 Version:

W_7225_Get_Card (card_number As Integer) As Integer

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

W_7225_Get_Card (card_number As Long) As Long

C/C++ (DOS)

int _7225_Get_Card(int *card_number)

@ Argument

card_number : card identification.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.6.4 W_7225_DI_Channel / _7225_DI_Channel

@ Description

This function is used to read data from digital input channels. 16 input channels are provided in the ACL-7225 card. By using this function, you can get the status of each input channel in this card.

Note : Channel is defined as one bit of input or output unit.

@ Syntax

Microsoft C/C++

```
int W_7225_DI_Channel( int channel_no, unsigned int
*di_data )
```

Visual Basic

Windows 3.11 Version:

```
W_7225_DI_Channel (ByVal channel_no As Integer,
di_data As Integer) As Integer
```

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

```
W_7225_DI_Channel (ByVal channel_no As Long, di_data
As Long) As Integer
```

C/C++ (DOS)

```
int _7225_DI_Channel( int channel_no, unsigned int
*di_data )
```

@ Argument

channel_no : Indicate which channel is read, the valid number is from 0 to 15. (ACL-7225 has 16 inputs)

di_data : Return value from input channel. The value is either 0 or 1.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDIChannel

2.6.5 W_7225_DI / _7225_DI

@ Description

This function is used to read data from digital inputs.

@ Syntax

Microsoft C/C++

```
int W_7225_DI( unsigned int *di_data )
```

Visual Basic

Windows 3.11 Version:

```
W_7225_DI( di_data As Integer) As Integer
```

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

```
W_7225_DI( di_data As Long) As Long
```

C/C++ (DOS)

```
int _7225_DI( unsigned int *di_data )
```

@ Argument

di_data : value is read from input port.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.6.6 W_7225_DO / _7225_DO

@ Description

This function is used to write data to digital output port.

@ Syntax

Microsoft C/C++

int W_7225_DO(unsigned int do_data)

Visual Basic

Windows 3.11 Version:

W_7225_DO(ByVal do_data As Integer) As Integer

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

W_7225_DO(ByVal do_data As Long) As Long

C/C++ (DOS)

int _7225_DO(unsigned int do_data)

@ Argument

do_data: value written to output port, the valid value is from 0 to 65535

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.6.7 W_7225_Read_Back / _7225_ Read_Back

@ Description

This function is used to read back the signals from digital output port.

@ Syntax

Microsoft C/C++

int W_7225_ Read_Back (unsigned int *do_data)

Visual Basic

Windows 3.11 Version:

W_7225_Read_Back (do_data As Integer) As Integer

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

W_7225_Read_Back (do_data As Long) As Long

C/C++ (DOS)

int _7225_Read_Back (unsigned long *do_data)

@ Argument

do_data: value written to outputs from digital output port, the valid value is from 0 to 65535

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.6.8 W_7225_DO_Channel /_7225_DO_Channel

@ Description

This function is used to write data to the digital output channels. By using this function, you can control the status of each output channel.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

int W_7225_DO_Channel (int channel_no, unsigned char do_data)

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

int W_7225_ DO_Channel (int channel_no, unsigned char do_data)

Visual Basic

Windows 3.11 Version:

W_7225_ DO_Channel (ByVal channel_no As Integer, ByVal do_data As integer) As Integer

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

W_7225_ DO_Channel (ByVal channel_no As Long, ByVal do_data As Long) As Long

C/C++ (DOS)

int _7225_ DO_Channel (unsigned int channel_no, unsigned char do_data)

@ Argument

channel_no : indicate which channel is written, the valid data is from 0 to 15

do_data : the value of output channel. The value is either o or 1

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDOChannel
ERR_InvalidChangeValue

2.7 ACL-7130 Software Drivers

The ACL-7130 is 32 isolated digital I/O (16 opto-isolated digital inputs and 16 digital outputs) and 32 TTL/DTL compatible digital I/O board. Two types of drivers are supported for ACL-7130, one is C language library for DOS environment, the other is DLL driver for Windows, Win 95, and Windows NT/2000. The detailed description is described as follows.

There are 15 functions provided for ACL-7130 Digital I/O cards. The functions used in DOS and their corresponding functions used in Windows are specified in the following table.

Functions in DOS	Functions in Windows
_7130_Initial	W_7130_Initial
_7130_Set_Card	W_7130_Set_Card
_7130_Get_Card	W_7130_Get_Card
_7130_D0_8	W_7130_D0_8
_7130_DO_16	W_7130_DO_16
_7130_DI_Channel	W_7130_DI_Channel
_7130_DI_8	W_7130_DI_8
_7130_DI_16	W_7130_DI_16
---	W_7130_INT_Enable
---	W_7130_INT_Disable
---	W_7130_INT_Timer_Start
---	W_7130_INT_Timer_Stop
_7130_ Timer _Start	W_7130_Timer_Start
_7130_ Timer _Read	W_7130_Timer_Read
_7130_ Timer _Stop	W_7130_Timer_Stop

2.7.1 W_7130_Initial /_7130_Initial

@ Description

An ACL-7130 card is initialized according to the card number and its corresponding base address. Every ACL-7130 card has to be initialized by this function before calling other functions. Up to 8 cards can be initialized in the same system.

@ Syntax

Microsoft C/C++

int W_7130_Initial(int card_number, int base_address, int irq1, int irq2)

Visual Basic

Windows 3.11 Version:

W_7130_Initial (ByVal card_number As Integer, ByVal base_address As Integer) As Integer

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

W_7130_Initial (ByVal card_number As Long, ByVal base_address As Long, ByVal irq1 As Long, ByVal irq2 As Long) As Long

C/C++ (DOS)

int _7130_Initial(int card_number, int base_address)

@ 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 to 7.

base_address : The I/O port base address of the card. Please refer to user's manual of ACL-7130 for I/O base address setting.

Irq1 : The IRQ1 level of your ACL-7130 card. The interrupt signals are generated by the external digital signals. This irq1 value should be the same as hardware setting.

Irq2 : The IRQ2 level of your ACL-7130 card. The interrupt signals are generated by internal 8254 timer. This irq2 value should be the same as hardware setting.

Note: Since Windows NT arranges resources to devices at system startup time, under Windows NT environment, parameter irq1 and irq2 are useless. You can not change IRQs level at run time. Please use DrvUtil utility to set IRQs level before running application. Please refer to section 2.4 “Some Installation Issues in Win-NT”.

@ Return Code

ERR_NoError
ERR_InvalidBoardNumber
ERR_BaseAddressError

2.7.2 W_7130_Set_Card / _7130_Set_Card

@ Description

This function is used on multi-cards system. After the ACL-7130 cards are initialized by W_7130_Initial() function, you can use this function to select which one you want to operate.

Note : Although up to 8 cards can be initialized in one system, but only one card can be operated at the same time. i.e. you have to choose which card you want to operate through W_7130_Set_Card function.

@ Syntax

Microsoft C/C++

int W_7130_Set_Card(int card_number)

Visual Basic

Windows 3.11 Version:

W_7130_Set_Card (ByVal card_number As Integer) As Integer

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

W_7130_Set_Card (ByVal card_number As Long) As Long

C/C++ (DOS)

int _7130_Set_Card(int card_number)

@ 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

2.7.3 W_7130_Get_Card / _7130_Get_Card

@ Description

This function is used on multi-cards system. You can use this function to get which card is operated currently.

@ Syntax

Microsoft C/C++

int W_7130_Get_Card(int *card_number)

Visual Basic

Windows 3.11 Version:

W_7130_Get_Card (card_number As Integer) As Integer

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

W_7130_Get_Card (card_number As Long) As Long

C/C++ (DOS)

int _7130_Get_Card(int *card_number)

@ Argument

card_number : card identification.

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.7.4 W_7130_DI_Channel / _7130_DI_Channel

@ Description

This function is used to read data from digital input channels. 32 input channels are provided in the ACL-7130 card. The channel 0 to 15 are used for isolated digital input channels in CN3 and The channel 16 to 31 are used for non-isolated digital input channels in CN2. By using this function, you can get the status of each input channel in this card.

Note : Channel is defined as one bit of input or output unit.

@ Syntax

Microsoft C/C++

int W_7130_DI_Channel(int channel_no, unsigned int *di_data)

Visual Basic

Windows 3.11 Version:

W_7130_DI_Channel (ByVal channel_no As Integer, di_data As Integer) As Integer

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

W_7130_DI_Channel (ByVal channel_no As Long, di_data As Long) As Integer

C/C++ (DOS)

int _7130_DI_Channel(int channel_no, unsigned int *di_data)

@ Argument

channel_no : Indicate which channel is read, the valid number is from 0 to 32. (ACL-7130 has 32 inputs)

di_data : Return value from input channel. The value is either 0 or 1.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidDIChannel

2.7.5 W_7130_DI_16 / _7130_DI_16

@ Description

This function is used to read data from digital input port. one 20-pin head connector and one 37 pins D-type connector are provided in ACL-7130 card, each connector consists of 16 input channels.

@ Syntax

Microsoft C/C++

int W_7130_DI_16(int port, unsigned int *di_data)

Visual Basic

Windows 3.11 Version:

W_7130_DI_16(ByVal port As Integer, di_data As Integer) As Integer

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

W_7130_DI_16(ByVal port As Long, di_data As Long) As Long

C/C++ (DOS)

int _7130_DI_16(int port, unsigned int *di_data)

@ Argument

port : port number to read input data, the value is:

DI_PORT0	Isolated DI channel 0 ... 15
DI_PORT1	Non-isolated channel 16 ... 31

di_data : value that is read from input port, the value is from 0 to 65535.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ChannelError

2.7.6 W_7130_DI_8 / _7130_DI_8

@ Description

This function is used to read data from digital inputs. The ACL-7130's 32 input channels can be grouped into four bytes. You can read data from each byte through this function.

@ Syntax

Microsoft C/C++

int W_7130_DI_8(int byte_no, unsigned char *di_data)

Visual Basic

Windows 3.11 Version:

W_7130_DI_8(ByVal byte_no As Integer, di_data As Integer) As Integer

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

W_7130_DI_8(ByVal byte_no As Long, di_data As Long) As Long

C/C++ (DOS)

int _7130_DI_8(int byte_no, unsigned char *di_data)

@ Argument

byte_no : The byte of inputs, the relationship between the byte_no and its correspond input channels is:

DI_BYTE0 :	Isolated DI channel 0 ... 7
DI_BYTE1 :	Isolated DI channel 8 ... 15
DI_BYTE2 :	Non-isolated channel 16 ... 23
DI_BYTE3 :	Non-isolated channel 24 ... 31

di_data : value is read from inputs in every 8-bits, the value is within 0 and 255

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.7.7 W_7130_DO_16 / _7130_DO_16

@ Description

This function is used to write data to digital output ports. Two ports are provided in ACL-7130 card, DO_PORT1 and DO_PORT2, each port consists of 16 channels

@ Syntax

Microsoft C/C++

int W_7130_DO_16(int port, unsigned int do_data)

Visual Basic

Windows 3.11 Version:

W_7130_DO_16(ByVal port As Integer, ByVal do_data As Integer) As Integer

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

W_7130_DO_16(ByVal port As Long, ByVal do_data As Long) As Long

C/C++ (DOS)

int _7130_DO_16(int port, unsigned int do_data)

@ Argument

port : port number that output data is written to, the valid value is:

DO_PORT1	Isolated DO channel 0 ... 15
DO_PORT2	Non-isolated DO channel 16 ... 31

do_data : value is written to output port, the valid value is from 0 to 65535

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_ChannelError

2.7.8 W_7130_DO_8 / _7130_DO_8

@ Description

This function is used to write data to digital outputs. 32 outputs of the ACL-7130 card can be grouped into four bytes. You can write data to each byte through this function.

@ Syntax

Microsoft C/C++

int W_7130_DO_8(int byte_no, unsigned char do_data)

Visual Basic

Windows 3.11 Version:

W_7130_DO_8(ByVal byte_no As Integer, ByVal do_data As Integer) As Integer

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

W_7130_DO_8(ByVal byte_no As Long, ByVal do_data As Long) As Long

C/C++ (DOS)

int _7130_DO_8(int byte_no, unsigned char do_data)

@ Argument

byte_no : The byte of outputs, the relationship between the byte_no and its correspond output channels is :

DO_BYTE1	Isolated DO channel 0 ... 7
DO_BYTE2	Isolated DO channel 8 ... 15
DO_BYTE3	Non-isolated DO channel 16 ... 23
DO_BYTE4	Non-isolated DO channel 24 ... 31

do_data: value written to outputs every 8-bits, the valid value is from 0 to 255

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_PortError

2.7.9 W_7130_INT_Enable

@ Description

This function is only available in Window 95 driver and Windows NT/2000 driver. The function is used to initialize and start up the interrupt control. After calling this function, every time an interrupt request signal 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 samples program 7130int.c.

Note : The W_7130_INT_Enable and W_7130_INT_Disable are a pair of functions. That is, as the W_7130_INT_Enable is called, the W_7130_INT_Disable has to follow up behind it. Otherwise, the interrupt operation will not stop.

@ Syntax

Microsoft C/C++

int W_7130_INT_Enable(int irq_no, HANDLE *hIntEvent)

Visual Basic

W_7130_INT_Enable (ByVal irq_no As Integer, hIntEvent As Long) As Long

@ Argument

irq_no: IRQ selected
1: Lower IRQ (From External Digital I/O Signal)
2: Higher IRQ (From Internal Timer pacer)

hIntEvent: The handle of the event for interrupt signals.

@ Return Code

ERR_NoError
ERR_INTNotSet

2.7.10 W_7130_INT_Disable

@ Description

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

Note : This function has to be called after the W_7130_INT_Enable is called.

@ Syntax

Microsoft C/C++

int W_7130_INT_Disable(int irq_no)

Visual Basic

W_7130_INT_Disable (ByVal irq_no As Integer) As Long

@ Argument

irq_no: IRQ selected
1: Lower IRQ (From External Digital I/O Signal)
2: Higher IRQ (From Internal Timer pacer)

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_INTNotSet

2.7.11 W_7130_INT_Timer_Start / W_7130_Timer_Start /_7130_Timer_Start

@ Description

This function is used to set up the Timer #1 and Timer #2. Timer #1 & #2 are used as frequency dividers for generating constant clock pacer for interrupt trigger dedicatedly.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

int W_7130_Timer_Start(unsigned int c1 , unsigned int c2)

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

int W_7130_INT_Timer_Start(unsigned int c1 , unsigned int c2)

Visual Basic

Windows 3.11 Version:

W_7130_Timer_Start(ByVal c1 As Integer, ByVal c2 As integer) As Integer

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

W_7130_INT_Timer_Start(ByVal c1 As Long, ByVal c2 As Long) As Long

C/C++ (DOS)

int _7130_Timer_Start(unsigned int c1, unsigned int c2)

@ Argument

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

Two timer source are supported in ACL-7130 card. One is 2MHz and the other is 32.768KHz. The Clock pacer rate is :
2MHz/ (C1 * C2) or 32.768HHz / (C1 * C2). The timer source is selected by jumper setting.

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerValue

2.7.12 W_7130_INT_Timer_Stop / W_7130_Timer_Stop / _7130_Timer_Stop

@ Description

This function is used to stop the interrupt timer operation.

@ Syntax

Microsoft C/C++

Windows 3.11 Version:

Int W_7130_Timer_Stop( )

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

Int W_7130_INT_Timer_Stop( )

Visual Basic

Windows 3.11 Version:

W_7130_Timer_Stop( ) As Integer

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

W_7130_INT_Timer_Stop( ) As Long

C/C++ (DOS)

int _7130_Timer_Stop( )

@ Argument

No arguments

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.7.13 W_7130_Timer_Start /_7130_Timer_Start

@ Description

The counters of the ACL-7130's Timer/Counter chip can be freely programmed by the users. This function is used to program the counters. This counter can be used as frequency generator if internal clock is used. It can also be used as event counter if external clock is used.

@ Syntax

Microsoft C/C++

int W_7130_Timer_Start(int timer_no, int timer_mode, unsigned int c0)

Visual Basic

Windows 3.11 Version:

W_7130_Timer_Start (ByVal timer_no As Integer, ByVal timer_mode As Integer, ByVal c0 As Integer) As Integer

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

W_7130_Timer_Start (ByVal timer_no As Integer, ByVal timer_mode As Long, ByVal c0 As Long) As Long

C/C++ (DOS)

int _7130_Timer_Start(int timer_no, int timer_mode, unsigned int c0)

@ Argument

timer_no : the timer number will be started . the valid number are from 0 to 2.

timer_mode : the 8254 timer mode, the possible values are : TIMER_MODE0, TIMER_MODE1, TIMER_MODE2, TIMER_MODE3, TIMER_MODE4, TIMER_MODE5.

Please refer to the manual or reference books of Counter/Timer 8254 for more detailed information about timer mode.

c0 : the count value of timer

@ Return Code

ERR_NoError
ERR_BoardNoInit
ERR_InvalidTimerMode
ERR_InvalidTimerValue

2.7.14 W_7130_Timer_Read /_7130_Timer_Read

@ Description

This function is used to read the counter value of the Counter #0 ~ Counter#2.

@ Syntax

Microsoft C/C++

```
int  W_7130_Timer_Read( int timer_no, unsigned int
*counter_value )
```

Visual Basic

Windows 3.11 Version:

```
W_7130_Timer_Read(ByVal timer_no As Integer,
counter_value As Integer) As Integer
```

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

```
W_7130_Timer_Read (ByVal timer_no As Integer,
counter_value As Long) As Long
```

C/C++ (DOS)

```
int  _7130_Timer_Read (unsigned int timer_no, unsigned int
*counter_value )
```

@ Argument

timer_no : the timer number will be started . the valid number are from 0 to 2.

counter_value : the count value of the Counter

@ Return Code

ERR_NoError
ERR_BoardNoInit

2.7.15 W_7130_Timer_Stop/_7130_Timer_Stop

@ Description

This function is used to stop the event counting operation. That is, the clock output signal will be set to high after executing this function.

@ Syntax

Microsoft C/C++

int W_7130_Timer_Stop(int timer_no, unsigned int *counter_value)

Visual Basic

Windows 3.11 Version:

W_7130_Timer_Stop (ByVal timer_no As Integer, counter_value As Integer) As Integer

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

W_7130_Timer_Stop (ByVal timer_no, counter_value As Long) As Long

C/C++ (DOS)

int _7130_Timer_Stop(unsigned int timer_no, unsigned int *counter_value)

@ Argument

timer_no : the timer number will be started . the valid number are from 0 to 2.

counter_value : the current count value of the Counter

@ Return Code

ERR_NoError
ERR_BoardNoInit

Appendix A Status Codes

This appendix lists the status codes returned by ACLS-DLL1, including the name and description.
Each ACLS-DLL1 function returns a status code that indicates whether the function was performed successfully. When an ACLS-DLL1 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
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
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