PCIS-DASK

Data Acquisition Software Development Kit
For NuDAQ PCI Bus Cards

Function Reference Manual

Manual Revision: 2.04

Revision Date: January 16, 2009

Part No: 50-11223-2030

Recycled Paper

Advance Technologies; Automate the World.

Revision History

Revision	Release Date	Description of Change(s)
2.00	2007/03/06	Document created Initial release
2.01	2007/07/17	Bookmarks added
2.02	2008/04/07	Added support for PCI-9524 and PCI-6202
2.03	2008/09/17	Added support for PCI-9222 and PCI-9223
2.04	2009/01/16	Added new features for PCI-9524

Preface

Using this Manual

Audience and Scope

This manual guides you when using the PCIS-DASK software driver for NuDAQ PCI bus data acquisition cards. This manual also describes how to install and use the PCIS-DASK function library when creating programs for your software applications.

Manual Organization

This manual is organized as follows:

Preface: Presents important copyright notifications, disclaimers, trademarks, and associated information on the proper understanding and usage of this document and its associated product(s).

Chapter 1 Introduction: This chapter inductees the PCIS-DASK, the fundamentals of building Windows-based applications, and describes the classes of functions that the PCIS-DASK supports.

Chapter 2 Function Reference: This section provides detailed description of each function call that the PCIS-DASK provides.

Appendix: This chapter provides references on status codes, AI range codes, AI data format, and function support.

Conventions

Take note of the following conventions used throughout this manual to make sure that users perform certain tasks and instructions properly.

NOTE:	Additional information, aids, and tips that help users perform tasks.
CAUTION:	Information to prevent ***minor*** physical injury, component damage, data loss, and/or program corruption when trying to complete a task.
WARNING:	Information to prevent ***serious*** physical injury, component damage, data loss, and/or program corruption when trying to complete a specific task.

Reference Documentation

The following list of documents may be used as reference materials to support installation, configuration and/or the operation of the PCIS-DASK devices described in this Function Reference Manual. This list is prepared in alphabetical order (by vendor name, then by document title) for clarity.

Vendor(s)	Title	Rev.
ADLINK Technology, Inc.	PCIS-DASK User's Manual: Data Acquisition Software Development Kit for NuDAQ® PCI Bus Cards (Hardware Support)	2.00

Getting Service

Contact us should you require any service or assistance.

ADLINK Technology Inc.

Address:	9F, No.166 Jian Yi Road, Chungho City,
	Taipei County 235, Taiwan
	台北縣中和市建一路 166 號 9 樓
Tel:	+886-2-8226-5877
Fax:	+886-2-8226-5717
Email:	service@adlinktech.com

Ampro ADLINK Technology Inc.

Address:	5215 Hellyer Avenue, #110, San Jose, CA 95138, USA
Tel:	+1-408-360-0200
Toll Free:	+1-800-966-5200 (USA only)
Fax:	+1-408-360-0222
Email:	info@adlinktech.com

ADLINK Technology Beijing

Address:	北京市海淀区上地东路 1 号盈创动力大厦 E 座 801 室
	(100085)
	Rm. 801, Power Creative E, No. 1, B/D
	Shang Di East Rd., Beijing 100085, China
Tel:	+86-10-5885-8666
Fax:	+86-10-5885-8625
Email:	market@adlinktech.com

ADLINK Technology Shanghai

Address:	上海市漕河泾高科技开发区钦江路 333 号 39 幢 4 层
	(200233)
Tel:	+86-21-6495-5210
Fax:	+86-21-5450-0414
Email:	market@adlinktech.com

ADLINK Technology Shenzhen

Address:	深圳市南山区科技园南区高新南七道 数字技术园
	A1 栋 2 楼 C 区 (518057)
	2F, C Block, Bld. A1, Cyber-Tech Zone,
	Gao Xin Ave. Sec 7, High-Tech Industrial Park S.,
	Shenzhen, 518054 China
Tel:	+86-755-2643-4858
Fax:	+86-755-2664-6353
Email:	market@adlinktech.com

ADLINK Technology Inc. (German Liaison Office)

Address:	Nord Carree 3, 40477 Duesseldorf, Germany
Tel:	+49-211-495-5552
Fax:	+49-211-495-5557
Email:	emea@adlinktech.com

ADLINK (French Liaison Office)

Address:	15 rue Emile Baudot, 91300 MASSY Cedex, France
Tel:	+33 (0) 1 60 12 35 66
Fax:	+33 (0) 1 60 12 35 66
Email:	france@adlinktech.com

ADLINK Technology Japan Corporation

Address:	151-0072 東京都渋谷区幡ヶ谷 1-1-2 朝日生命幡ヶ谷ビル 8F Asahiseimei Hatagaya Bldg. 8F 1-1-2 Hatagaya, Shibuya-ku, Tokyo 151-0072, Japan
Tel:	+81-3-4455-3722
Fax:	+81-3-5333-6040
Email:	japan@adlinktech.com

ADLINK Technology Inc. (Korean Liaison Office)

Address:	서울시 강남구 논현동 60-12 동성빌딩 4 층 402 호 No.402, Dongsung B/D, 60-12, Nonhyeon-Dong Gangnam-gu, Seoul, 135-010, Korea.
Tel:	+82-2-2057-0565
Fax:	+82-2-2057-0563
Email:	korea@adlinktech.com

ADLINK Technology Singapore Pte Ltd.

Address:	84 Genting Lane #07-02A, Cityneon Design Centre, Singapore 349584
Tel:	+65-6844-2261
Fax:	+65-6844-2263
Email:	singapore@adlinktech.com

ADLINK Technology Singapore Pte Ltd. (Indian Liaison Office)

Address:	No. 1357, "Anupama", Sri Aurobindo Marg, 9th Cross, JP Nagar Phase I, Bangalore - 560078, India
Tel:	+91-80-65605817
Fax:	+91-80-22443548
Email:	india@adlinktech.com

Table of Contents

1 Introduction

The PCIS-DASK is a software driver for NuDAQ PCI-bus data acquisition cards. It is a high performance data acquisition driver for developing custom applications under Windows environment.

Using PCIS-DASK lets you enjoy the advantages of the power and flexibility of Windows for your data acquisition applications. These include running multiple applications and using extended memory. In addition, implementing PCIS-DASK under Visual Basic environment makes it easy to create custom user interfaces and graphics.

1.1 Application Building Fundamentals in Windows

The following sections provide fundamental instructions when using PCIS-DASK to build application in Windows 98/NT/2000 operating environment.

Using Microsoft Visual C/C++

Follow these steps to create a data acquisition application using PCIS-DASK and Microsoft Visual C/C++.

1. Launch the Microsoft Visual C/C++ application.
2. Open a new or existing project that you want to apply the PCIS-DASK.
3. Include header file DASK.H in the C/C++ source files that call PCIS-DASK functions. DASK.H contains all the function declarations and constants that can be used to develop data acquisition applications. Incorporate the following statement in the code to include the header file.

   ```
   #include “DASK.H”
   ```
4. After setting the appropriate compile and link options, build the application by selecting the Build command from Build menu. Remember to link PCIS-DASK’s import library, PCIS-DASK.LIB.

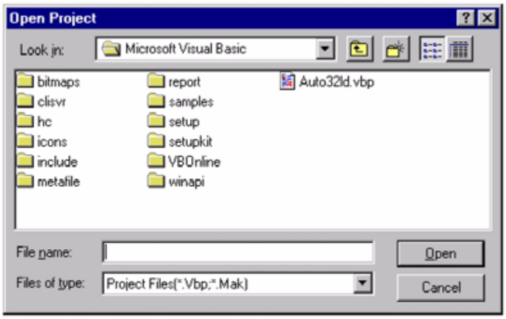
Using Microsoft Visual Basic

Follow the steps in the succeeding sections to create a data acquisition application using PCIS-DASK and Visual Basic.

Open a project

Do one of the following to open a new or existing project:

1. Open a new project by selecting the New Project command from the File menu. To open an existing project, select the Open Project command from the File menu to display the Open Project dialog box.

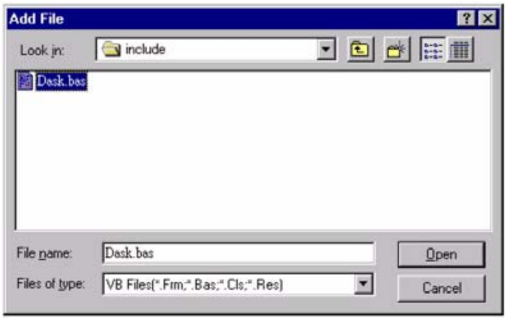


2. Locate the existing project, then double-click on the project file name to load.

Add the file

You must add the file **DASK.BAS** to the project, if the file is not yet included. This file contains all the procedure declarations and constants that can be used to develop the data acquisition application. To add the file:

1. Select Add File from the File menu. The Add File window appears, displaying a list of files in the current directory.



Look in: include

Dask.bas

File name: Dask.bas

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

2. Double-click on the DASK.BAS file. If the file is not on the list, make sure the list is displaying files from the correct directory. By default, the DASK.BAS file is installed at C:\ADLink\PCIS-DASK\INCLUDE.

Design the interface

To design the interface for the application, place all the interface elements such as command buttons, list boxes, and text boxes on the Visual Basic form. These standard controls are available from the Visual Basic Toolbox.

To place a control on the form, select the desired control from the Toolbox, then draw it on the form. You may also double-click on the control icon from the Toolbox to place it on the form.

Set the interface controls

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

Write the event code

The event code defines the required action to be performed when an event occurs. To write the event code, double-click on the control or form to view the code module, then add the event code. You can also call the functions declared in the DASK.BAS file to perform data acquisition operations.

Run the application

Do one of the following to run the application:

- Choose **Start** from the **Run** menu
- Click the Start icon from the toolbar
- Press <F5>

Distribute the application

After completing the project, save the application as an executable (.EXE) file using the **Make EXE File** command from the File menu. The application, after being transformed into an executable file, is now ready for distribution.

You must include the PCIS-DASK's DLL and driver files when the application is distributed.

1.2 Application Building Fundamentals in Linux

The following sections provide fundamental instructions when using PCIS-DASK to build application in Linux. To create a data acquisition application using PCIS-DASK/X and GNU C/C++, follow these steps:

Edit the source files

Include the header file **dask.h** in the C/C++ source files that call PCIS-DASK/X functions. The d2kdask.h has all the function declarations and constants that you can use to develop your data acquisition application. Add this statement in your code to include the header file.

```
#include "dask.h"
```

Build your application

Using the appropriate C/C++ compiler (gcc or cc) to compile the program. You should add **-lpci_dask** option to link **libpci_dask.so** library. For multi-threaded applications, the **-lpthread** string is required. For example:

```
gcc -o testai testai.c -lpci_dask
```

1.3 Function Classes

This chapter describes the classes of functions that the PCIS-DASK supports. All PCIS-DASK functions are grouped into different classes:

- General Configuration Function Group
- Actual Sampling Rate Function Group
- Analog Input Function Group
  - Analog Input Configuration Functions
  - One-Shot Analog Input Functions
  - Continuous Analog Input Functions
  - Asynchronous Analog Input Monitoring Functions
- Analog Output Function Group
  - Analog Output Configuration Functions
  - One-Shot Analog Output Functions
  - Continuous Analog Output Functions
  - Asynchronous Analog Output Monitoring Functions
- Digital Input Function Group
  - Digital Input Configuration Functions
  - One-Shot Digital Input Functions
  - Continuous Digital Input Functions
  - Asynchronous Digital Input Monitoring Functions
- Digital Output Function Group
  - Digital Output Configuration Functions
  - One-Shot Digital Output Functions
  - Continuous Digital Output Functions
  - Asynchronous Digital Output Monitoring Functions
- Timer/Counter Function Group
- DIO Function Group
  - Digital Input/Output Configuration Functions
  - Dual-Interrupt System Setting Functions
  - Local Interrupt Setting Functions
- Emergency Shutdown Function Group
- Watchdog Timer Function Group
- Hot-system Reset Hold Function Group

- Calibration Function Group
- SSI Function Group
- PWM Function Group

2 Function Reference

This chapter contains the detailed description of PCIS-DASK functions, including the PCIS-DASK data types and function reference. The functions are arranged alphabetically in section 2.2.

2.1 Data Types

The PCIS-DASK library uses these data types in DASK.H. It is recommended that you use these data types in your application programs. The table shows the data type names, ranges, and corresponding data types in C/C++, Visual Basic, and Delphi for your reference.

Type Name	**Description**	**Range**	**Type**		
			C/C++ (for 32-bit compiler)	**Visual Basic**	**Pascal (Delphi)**
U8	8-bit ASCII character	0 to 255	unsigned char	Byte	Byte
I16	16-bit signed integer	-32768 to 32767	short	Integer	SmallInt
U16	16-bit unsigned integer	0 to 65535	unsigned short	Not supported by BASIC, use the signed integer (I16) instead	Word
I32	32-bit signed integer	-2147483648 to 2147483647	long	Long	LongInt
U32	32-bit unsigned integer	0 to 4294967295	unsigned long	Not supported by BASIC, use the signed long integer (I32) instead	Cardinal
F32	32-bit single-precision floating-point	3.402823E38 to 3.402823E38	float	Single	Single
F64	64-bit double-precision floating-point	1.797683134862315E308 to 1.797683134862315E309	double	Double	Double

2.2 Function Reference

AI_9111_Config

Description

Informs the PCIS-DASK library of the trigger source and trigger mode selected for the PCI-9111 card with card ID CardNumber. You must call this function before calling function to perform continuous analog input operation.

Supported card(s)

9111

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9111_Config (U16 CardNumber, U16
     TrigSource, U16 TrigMode, U16 TraceCnt)
```

Visual Basic

```
AI_9111_Config (ByVal CardNumber As Integer,
     ByVal TrigSource As Integer, ByVal TrigMode
     As Integer, ByVal TraceCnt As Integer) As
     Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

TrigSource The continuous A/D conversion trigger source. Valid values:

TRIG_INT_PACER	Onboard programmable pacer
TRIG_EXT_STROBE	External signal trigger

TirgMode Trigger mode selection.

P9111_TRGMOD_SOFT	Software Trigger (no trigger)
P9111_TRGMOD_PRE	Pre-/Middle-Trigger
P9111_TRGMOD_POST	Post-Trigger (available only for devices with hardware version larger than or equals -to Rev. B1).

TraceCnt The number of data that will be accessed after a specific trigger event. This parameter(s) is only available for Pre-/Middle Trigger mode of continuous AI operation (i.e. the parameter(s) of TrigMode is set to be P9111_TRGMOD_PRE).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_9112_Config

Description

Informs PCIS-DASK library of the trigger source selected for the PCI-9112/cPCI-9112 with card ID CardNumber. You must call this function before calling function to perform continuous analog input operation.

Supported card(s)

9112

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9112_Config (U16 CardNumber, U16
     TrigSource)
```

Visual Basic

```
AI_9112_Config (ByVal CardNumber As Integer,
     ByVal TrigSource As Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

TrigSource The continuous A/D conversion trigger source. Valid values:

`TRIG_INT_PACER`	Onboard programmable pacer
`TRIG_EXT_STROBE`	External signal trigger

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_9113_Config

Description

Informs PCIS-DASK library of the trigger source selected for the PCI-9113 with card ID CardNumber. You must call this function before calling function to perform continuous analog input operation.

Supported card

9113

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9113_Config (U16 CardNumber, U16
     TrigSource)
```

Visual Basic

```
AI_9113_Config (ByVal CardNumber As Integer,
     ByVal TrigSource As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
TrigSource	The continuous A/D conversion trigger source.Valid value:

`TRIG_INT_PACER`	Onboard programmable pacer

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_9114_Config

Description

Informs PCIS-DASK library of the trigger source selected for the PCI-9114 with card ID CardNumber. You must call this function before calling function to perform continuous analog input operation.

Supported card(s)

9114

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9114_Config (U16 CardNumber, U16
     TrigSource)
```

Visual Basic

```
AI_9114_Config (ByVal CardNumber As Integer,
     ByVal TrigSource As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
TrigSource	The continuous A/D conversion trigger source. Valid values:

`TRIG_INT_PACER`	Onboard programmable pacer
`TRIG_EXT_STROBE`	External signal trigger

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_9114_PreTrigConfig

Description

Informs the PCIS-DASK library to enable or disable the pre-trigger mode of continuous AI for the PCI-9114A with card ID CardNumber. You must call this function before calling function to perform pre-trigger mode of continuous analog input operation.

Supported card(s)

9114A

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9114_PreTrigConfig (U16 CardNumber, U16
     PreTrgEn, U16 TraceCnt)
```

Visual Basic

```
AI_9114_PreTrigConfig (ByVal CardNumber As
     Integer, ByVal PreTrgEn As Integer, ByVal
     TraceCnt As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
PreTrgEn	Enable or disable Pre-Trigger mode. Valid values:

TRUE	Enable Pre-Trigger mode
FALSE	Disable Pre-Trigger mode

TraceCnt	The number of data to be accessed after a specific trigger event.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_9116_Config

Description

Informs the PCIS-DASK library of the trigger source, trigger mode, and trigger properties selected for the PCI-9116 with card ID Card-Number. You must call this function before calling function to perform continuous analog input operation.

Supported card(s)

9116

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9116_Config (U16 CardNumber, U16
     ConfigCtrl, U16 TrigCtrl, U16 PostCnt, U16
     MCnt, U16 ReTrgCnt)
```

Visual Basic

```
AI_9116_Config (ByVal CardNumber As Integer,
     ByVal ConfigCtrl As Integer, ByVal TrigCtrl
     As Integer, ByVal PostCnt As Integer, ByVal
     MCnt As Integer, ByVal ReTrgCnt As Integer)
     As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

ConfigCtrl The setting for A/D mode control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are three groups of constants:

A/D Polarity Control

`P9116_AI_BiPolar`

`P9116_AI_UniPolar`

A/D Channel Input Mode

`P9116_AI_SingEnded`

`P9116_AI_Differential`

Common Mode Selection

`P9116_AI_LocalGND`	Local ground of cPCI-9116.
`P9116_AI_UserCMMD`	User-defined common mode.

When two or more constants are used to form the ConfigCtrl argument, the constants are combined with the bitwise-OR operator(|).

TrigCtrl The setting for A/D Trigger control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are seven groups of constants:

Trigger Mode Selection

`P9116_TRGMOD_SOFT`	Software Trigger (no trigger)
`P9116_TRGMOD_POST`	Post Trigger
`P9116_TRGMOD_DELAY`	Delay Trigger
`P9116_TRGMOD_PRE`	Pre-Trigger Mode
`P9116_TRGMOD_MIDL`	Middle Trigger

Trigger Polarity

`P9116_AI_TrgNegative`	Trigger negative edge active
`P9116_AI_TrgPositive`	Trigger positive edge active

Time Base Selection

`P9116_AI_IntTimeBase`	Internal time base (24 MHz)
`P9116_AI_ExtTimeBase`	External time base

Delay Source Selection

`P9116_AI_DlyInSamples`	Delay in samples
`P9116_AI_DlyInTimebase`	Delay in time base

Re-Trigger Mode Enable

`P9116_AI_ReTrigEn`	Re-trigger in an acquisition is enabled

MCounter Enable

`P9116_AI_MCounterEn`	Mcounter is enabled and then the trigger signal is ignore before M terminal count is reached.

AD Conversion Mode Selection

`P9116_AI_SoftPolling`	Software Polling
`P9116_AI_INT`	Interrupt mode of continuous AI
`P9116_AI_DMA`	DMA mode of continuous AI

When two or more constants are used to form the TrigCtrl argument, the constants are combined with the bitwise-OR operator(|).

PostCnt	The number of data that will be accessed after a specific trigger event. This argument is valid only for Middle trigger and Delay trigger modes.
MCnt	The counter value of MCounter. This argument is valid only for Pre-trigger and Middle trigger mode.
ReTrgCnt	The accepted trigger times in an acquisition. This argument is valid only for Delay trigger and Post trigger modes.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_9116_CounterInterval

Description

Informs the PCIS-DASK library of the scan interval value and sample interval value selected for the analog input operation of PCI9116. You must call this function before calling function to perform continuous analog input operation of PCI9116.

Supported card(s)

9116

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9116_CounterInterval (U16 CardNumber, U32
     ScanIntrv, U32 SampIntrv)
```

Visual Basic

```
AI_9116_CounterInterval (ByVal CardNumber As
     Integer, ByVal ScanIntrv As Long, ByVal
     SampIntrv As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ScanIntrv	The length of the scan interval (the counter value between the initiation of each scan sequence). Range: 96 through 16777215.
SampIntrv	The length of the sample interval (that is, the counter value between each A/D conversion within a scan sequence). Range: 96 through 65535.

NOTE: The value of ScanIntrv must be greater than or equal to the sum of the total sample interval (the number of channels in a scan sequence * SampIntrv).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_9118_Config

Description

Informs the PCIS-DASK library of the trigger source, trigger mode, and trigger properties selected for the PCI-9118 with card ID CardNumber. You must call this function before calling function to perform continuous analog input operation.

Supported card(s)

9118

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9118_Config (U16 CardNumber, U16 ModeCtrl,
     U16 FunCtrl, U16 BurstCnt, U16 PostCnt)
```

Visual Basic

```
AI_9118_Config (ByVal CardNumber As Integer,
     ByVal ModeCtrl As Integer, ByVal FunCtrl As
     Integer, ByVal BurstCnt As Integer, ByVal
     PostCnt As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ModeCtrl	The setting for A/D mode control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are four groups of constants:

A/D Polarity Control

`P9118_AI_BiPolar`

`P9118_AI_UniPolar`

A/D Channel Input Mode

`P9118_AI_SingEnded`

`P9118_AI_Differential`

External Gate Enable

`P9118_AI_ExtG`	8254 counter is controlled by TGIN pin

External Trigger Enable

`P9118_AI_ExtTrig`	External hardware trigger mode enabled

When two or more constants are used to form the ModeCtrl argument, the constants are combined with the bitwise-OR operator(|).

FunCtrl The setting for A/D Function. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are four groups of constants:

Digital Trigger Polarity

`P9118_AI_DtrgNegative`	Digital trigger negative active
`P9118_AI_DtrgPositive`	Digital trigger positive active

External Trigger Polarity

`P9118_AI_EtrgNegative`	External trigger negative active
`P9118_AI_EtrgPositive`	External trigger positive active

Burst Mode Enable

`P9118_AI_BurstModeEn`	Burst mode is enabled.
`P9118_AI_SampleHold`	Burst mode with sample and hold is enabled.

Trigger Mode Enable

`P9118_AI_PostTrgEn`	Post trigger mode is enabled.
`P9118_AI_AboutTrgEn`	About trigger mode or Pre-trigger mode is enabled.

When two or more constants are used to form the ModeCtrl argument, the constants are combined with the bitwise-OR operator(|).

BurstCnt The burst number.

PostCnt The number of data that will be accessed after a specific trigger event.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_9221_Config

Description

Inform the PCIS-DASK library of the trigger source, trigger mode, and trigger properties selected for the PCI-9221 with card ID CardNumber. You must call this function before calling function to perform continuous analog input operation.

Supported card(s)

9221

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9221_Config (U16 CardNumber, U16
     ConfigCtrl, U16 TrigCtrl, BOOLEAN
     AutoResetBuf)
```

Visual Basic

```
AI_9221_Config (ByVal CardNumber As Integer,
     ByVal ConfigCtrl As Integer, ByVal TrigCtrl
     As Integer, ByVal AutoResetBuf As Byte) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ConfigCtrl	The setting for A/D mode control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are three groups of constants:

A/D Channel Input Mode

`P9221_AI_SingEnded`

`P9221_AI_NonRef_SingEnded`

`P9221_AI_Differential`

Time Base Selection

`P9221_AI_IntTimeBase`

`P9221_AI_ExtTimeBase`

External Time Base Source Selection

`P9221_TimeBaseSRC_GPI0`

`P9221_TimeBaseSRC_GPI1`

`P9221_TimeBaseSRC_GPI2`

`P9221_TimeBaseSRC_GPI3`

`P9221_TimeBaseSRC_GPI4`

`P9221_TimeBaseSRC_GPI5`

`P9221_TimeBaseSRC_GPI6`

`P9221_TimeBaseSRC_GPI7`

TrigCtrl The setting for A/D Trigger control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are three groups of constants:

Trigger Mode Selection

`P9221_TRGMOD_SOFT`

`P9221_TRGMOD_ExtD`

External Digital Trigger Source Selection

`P9221_TRGSRC_GPI0`

`P9221_TRGSRC_GPI1`

`P9221_TRGSRC_GPI2`

`P9221_TRGSRC_GPI3`

`P9221_TRGSRC_GPI4`

`P9221_TRGSRC_GPI5`

`P9221_TRGSRC_GPI6`

`P9221_TRGSRC_GPI7`

Trigger Polarity

`P9221_AI_TrgPositive`

`P9221_AI_TrgNegative`

AutoResetBuf

FALSE	The AI buffers set by "AI_ContBufferSetup" are retained and must call "AI_ContBufferReset" to reset the buffer.
TRUE	The AI buffers set by "AI_ContBufferSetup" are reset automatically by driver while the AI operation is finished.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_9221_CounterInterval

Description

Inform the PCI-DASK library of the scan interval value and sample interval value selected for the analog input operation of PCI-9221. You must call this function before calling function to perform continuous analog input operation of PCI-9221.

Supported card(s)

9221

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9221_CounterInterval (U16 CardNumber, U32
     ScanIntrv, U32 SampIntrv)
```

Visual Basic

```
AI_9221_CounterInterval (ByVal CardNumber As
     Integer, ByVal ScanIntrv As Long, ByVal
     SampIntrv As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ScanIntrv	The length of the scan interval (the counter value between the initiation of each scan sequence). Ranges are 160 to 4294967295 (internal clock) or 1 to 4294967295 (external clock).
SampIntrv	The length of the sample interval (that is, the counter value between each A/D conversion within a scan sequence). Ranges are 160 to 16777215 (internal clock) or 1 to 16777215 (external clock).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidCounterValue
```

AI_9222_Config

Description

Informs the PCIS-DASK library of the trigger source, trigger mode, and trigger properties selected for the PCI-9222 with card ID CardNumber. You must call this function before calling function to perform continuous analog input operation.

Supported card(s)

9222

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9222_Config (U16 CardNumber, U16
     ConfigCtrl, U16 TrigCtrl, U32 ReTriggerCnt,
     BOOLEAN AutoResetBuf)
```

Visual Basic

```
AI_9222_Config (ByVal CardNumber As Integer,
     ByVal ConfigCtrl As Integer, ByVal TrigCtrl
     As Integer, ByVal ReTriggerCnt As Long,
     ByVal AutoResetBuf As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ConfigCtrl	The setting for A/D mode control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are two groups of constants:

A/D Channel Input Mode

`P922x_AI_SingEnded`

`P922x_AI_NonRef_SingEnded`

`P922x_AI_Differential`

Conversion Source Selection

`P922x_AI_CONVSRC_INT`

`P922x_AI_CONVSRC_GPI0`

`P922x_AI_CONVSRC_GPI1`

`P922x_AI_CONVSRC_GPI2`

`P922x_AI_CONVSRC_GPI3`

	`P922x_AI_CONVSRC_GPI4`
	`P922x_AI_CONVSRC_GPI5`
	`P922x_AI_CONVSRC_GPI6`
	`P922x_AI_CONVSRC_GPI7`
	`P922x_AI_CONVSRC_SSI1`

TrigCtrl	The setting for A/D Trigger control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are four groups of constants:
	Trigger Mode Selection
	`P922x_AI_TRGMOD_POST`
	`P922x_AI_TRGMOD_GATED`
	Trigger Source Selection
	`P922x_AI_TRGSRC_SOFT`
	`P922x_AI_TRGSRC_GPI0`
	`P922x_AI_TRGSRC_GPI1`
	`P922x_AI_TRGSRC_GPI2`
	`P922x_AI_TRGSRC_GPI3`
	`P922x_AI_TRGSRC_GPI4`
	`P922x_AI_TRIGSRC_GPI5`
	`P922x_AI_TRIGSRC_GPI6`
	`P922x_AI_TRIGSRC_GPI7`
	`P922x_AI_TRIGSRC_SSI5`
	Trigger Polarity
	`P922x_AI_TrgPositive`
	`P922x_AI_TrgNegative`
	Re-Trigger Mode Enable
	`P922x_AI_EnReTigger`

ReTriggerCnt	The accepted trigger times in an acquisition. The valid range of ReTriggerCnt is 0 to 4294967295. If the value of ReTriggerCnt is 0, the AI operation is triggered infinitely. This argument is valid only for post trigger with re-trigger mode.
AutoResetBuf	FALSE

The AI buffers set by the AI_ContBufferSetup function are retained. You must call the AI_ContBufferReset function to reset the buffer.

TRUE

The AI buffers set by the AI_ContBufferSetup function are reset automatically by driver when the AI operation is completed.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidTriggerMode
ErrorConfigIoctl
```

AI_9222_CounterInterval

Description

Inform the PCI-DASK library of the scan interval value and sample interval value selected for the analog input operation of PCI-9222. With internal conversion source, you must call this function before calling function to perform continuous analog input operation of PCI-9222.

Supported card(s)

9222

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9222_CounterInterval (U16 CardNumber, U32
     ScanIntrv, U32 SampIntrv)
```

Visual Basic

```
AI_9222_CounterInterval (ByVal CardNumber As
     Integer, ByVal ScanIntrv As Long, ByVal
     SampIntrv As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ScanIntrv	The length of the scan interval (the counter value between the initiation of each scan sequence). The value must large than or equal to (sample interval * performed AI channel count). Valid range is 320 to 4294967295.
SampIntrv	The length of the sample interval (that is, the counter value between each A/D conversion within a scan sequence). Valid range is 320 to 16777215.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorConfigIoctl
```

AI_9223_Config

Description

Inform the PCIS-DASK library of the trigger source, trigger mode, and trigger properties selected for the PCI-9223 with card ID CardNumber. You must call this function before calling function to perform continuous analog input operation.

Supported card(s)

9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9223_Config (U16 CardNumber, U16
     ConfigCtrl, U16 TrigCtrl, U32 ReTriggerCnt,
     BOOLEAN AutoResetBuf)
```

Visual Basic

```
AI_9223_Config (ByVal CardNumber As Integer,
     ByVal ConfigCtrl As Integer, ByVal TrigCtrl
     As Integer, ByVal ReTriggerCnt As Long,
     ByVal AutoResetBuf As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ConfigCtrl	The setting for A/D mode control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are two groups of constants:

A/D Channel Input Mode

`P922x_AI_SingEnded`

`P922x_AI_NonRef_SingEnded`

`P922x_AI_Differential`

Conversion Source Selection

`P922x_AI_CONVSRC_INT`

`P922x_AI_CONVSRC_GPI0`

`P922x_AI_CONVSRC_GPI1`

`P922x_AI_CONVSRC_GPI2`

`P922x_AI_CONVSRC_GPI3`

P922x_AI_CONVSRC_GPI4
P922x_AI_CONVSRC_GPI5
P922x_AI_CONVSRC_GPI6
P922x_AI_CONVSRC_GPI7
P922x_AI_CONVSRC_SSI1

TrigCtrl

The setting for A/D Trigger control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are four groups of constants:

Trigger Mode Selection

P922x_AI_TRGMOD_POST
P922x_AI_TRGMOD_GATED

Trigger Source Selection

P922x_AI_TRGSRC_SOFT
P922x_AI_TRGSRC_GPI0
P922x_AI_TRGSRC_GPI1
P922x_AI_TRGSRC_GPI2
P922x_AI_TRGSRC_GPI3
P922x_AI_TRGSRC_GPI4
P922x_AI_TRIGSRC_GPI5
P922x_AI_TRIGSRC_GPI6
P922x_AI_TRIGSRC_GPI7
P922x_AI_TRIGSRC_SSI5

Trigger Polarity

P922x_AI_TrgPositive
P922x_AI_TrgNegative

Re-Trigger Mode Enable

P922x_AI_EnReTigger

ReTriggerCnt

The accepted trigger times in an acquisition. The valid range of ReTriggerCnt is 0 to 4294967295. If the value of ReTriggerCnt is 0, the AI operation is triggered infinitely. This argument is valid only for post trigger with re-trigger mode.

AutoResetBuf

FALSE

The AI buffers set by the AI_ContBufferSetup function are retained. You must call the AI_ContBufferReset function to reset the buffer.

TRUE

The AI buffers set by the AI_ContBufferSetup function are reset automatically by driver when the AI operation is completed.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidTriggerMode
ErrorConfigIoctl
```

AI_9223_CounterInterval

Description

Inform the PCI-DASK library of the scan interval value and sample interval value selected for the analog input operation of PCI-9223. With internal conversion source, you must call this function before calling function to perform continuous analog input operation of PCI-9223.

Supported card(s)

9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9223_CounterInterval (U16 CardNumber, U32
     ScanIntrv, U32 SampIntrv)
```

Visual Basic

```
AI_9223_CounterInterval (ByVal CardNumber As
     Integer, ByVal ScanIntrv As Long, ByVal
     SampIntrv As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ScanIntrv	The length of the scan interval (the counter value between the initiation of each scan sequence). The value must large than or equal to (sample interval * performed AI channel count). Valid range is 160 to 4294967295.
SampIntrv	The length of the sample interval (that is, the counter value between each A/D conversion within a scan sequence). Valid range is 160 to 16777215.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorConfigIoctl
```

AI_9524_Config

Description

Informs the PCIS-DASK library of the trigger source, trigger mode, trigger properties, and some configurations selected for the PCI-9524 with card ID CardNumber. You must call this function before calling function to perform analog input operation.

Supported card(s)

9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9524_Config (U16 CardNumber, U16 Group,
     U16 XMode, U16 ConfigCtrl, U16 TrigCtrl, U32
     TrigValue)
```

Visual Basic

```
AI_9524_Config (ByVal CardNumber As Integer,
     ByVal Group As Integer, ByVal XMode As
     Integer, ByVal ConfigCtrl As Integer,
     ByVal TrigCtrl As Integer, ByVal TrigValue
     As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Group	9524 supports two AI groups, load cell group and general purpose group. Valid value: P9524_AI_LC_Group P9524_AI_GP_Group
XMode	The setting for A/D transfer mode. Valid value: P9524_AI_XFER_POLL P9524_AI_XFER_DMA
ConfigCtrl	The setting for A/D mode control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are four groups of constants:

Bridge Excitation Voltage (Only valid for load cell group)

P9524_VEX_Range_2R5V

P9524_VEX_Range_10V

Reference Voltage Mode (Only valid for load cell group)

P9524_VEX_Sence_Local

P9524_VEX_Sence_Remote

Enable Auto Zero Mode (Only valid for load cell group)

P9524_AI_AZMode

Enable Buffer Auto Reset (Only valid for DMA Mode)

P9524_AI_BufAutoReset

TrigCtrl The setting for A/D Trigger control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are three groups of constants:

Trigger Mode Selection

P9524_TRGMOD_POST

Trigger Source Selection

P9524_TRGSRC_SOFT

P9524_TRGSRC_ExtD

P9524_TRGSRC_SSI

P9524_TRGSRC_QD0

P9524_TRGSRC_PG0

Trigger Polarity Selection

P9524_AI_TrgPositive

P9524_AI_TrgNegative

TrigValue This argument is only valid while trigger source is selected to QD0 or PG0. While the pulse generator generates TrigValue steps or the quadrature decoder decodes to TrigValue, the AI trigger will be generated. Valid value: 1 ~ 0xffffff

Return Code(s)

```
NoError
ErrorFuncNotSupport
ErrorUndefinedParameter
ErrorConfigIoctl
```

AI_9524_PollConfig

Description

The function should be called if the AI transfer mode is set to polling mode by AI_9524_Config(). The function will start ADC with the set speed and range. You can use AI polling functions or set the AI_EventCallBack function with EOC (End of Conversion) event and a callback function to obtain the acquired value.

Supported card(s)

9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9524_PollConfig (U16 CardNumber, U16
     Group, U16 PollChannel, U16 PollRange, U16
     PollSpeed)
```

Visual Basic

```
AI_9524_Config (ByVal CardNumber As Integer,
     ByVal Group As Integer, ByVal PollChannel As
     Integer, ByVal PollRange As Integer, ByVal
     PollSpeed As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Group	9524 supports two AI groups, load cell group and general purpose group. Valid value: P9524_AI_LC_Group P9524_AI_GP_Group
PollChannel	The setting for A/D poll channel. The ADC will update the value of the set channel with the set range and speed. If the argument is set to all channels, the ADC will scans all channels of the set group. Valid value: P9524_AI_LC_CH0 P9524_AI_LC_CH1 P9524_AI_LC_CH2

P9524_AI_LC_CH3

P9524_AI_GP_CH0

P9524_AI_GP_CH1

P9524_AI_GP_CH2

P9524_AI_GP_CH3

P9524_AI_POLL_ALLCHANNELS

PollSpeed The setting for A/D sampling speed. Valid value: expression formed from one or more of the manifest constants

P9524_ADC_30K_SPS

P9524_ADC_15K_SPS

P9524_ADC_7K5_SPS

P9524_ADC_3K75_SPS

P9524_ADC_2K_SPS

P9524_ADC_1K_SPS

P9524_ADC_500_SPS

P9524_ADC_100_SPS

P9524_ADC_60_SPS

P9524_ADC_50_SPS

P9524_ADC_30_SPS

P9524_ADC_25_SPS

P9524_ADC_15_SPS

P9524_ADC_10_SPS

P9524_ADC_5_SPS

P9524_ADC_2R5_SPS

Return Code(s)

```
NoError
ErrorFuncNotSupport
ErrorInvalidAdRange
```

```
        ErrorInvalidSampleRate
        ErrorInvalidIoChannel
        ErrorUndefinedParameter
        ErrorConfigIoctl
```

AI_9524_SetDSP

Description

The function should be called if the load cell AI operation will be performed. The function sets DSP configurations for AI load cell channels. Please refer PCI-9524 hardware manual for details.

Supported card(s)

9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9524_SetDSP (U16 CardNumber, U16 Channel,
     U16 Mode, U16 DFStage, U32 SPKRejThreshold)
```

Visual Basic

```
AI_9524_Config (ByVal CardNumber As Integer,
     ByVal Channel As Integer, ByVal Mode As
     Integer, ByVal DFStage As Integer, ByVal
     SPKRejThreshold As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Channel	Channel number to be set. Valid value: P9524_AI_LC_CH0 P9524_AI_LC_CH1 P9524_AI_LC_CH2 P9524_AI_LC_CH3
Mode	PCI-9524 provides a spike rejecter for load cell channels. This argument indicates enable or disable the functionality. Valid value: P9524_SPIKE_REJ_DISABLE P9524_SPIKE_REJ_ENABLE
DFStage	The setting for digital filter taps. Valid value: 0 to 10 (1-tap to 1024-tap)

SPKRejThreshold	The setting for the threshold that makes the digital filter flushed its content once which is exceeded. Valid value: 1 to 0xffffff

Return Code(s)

```
NoError
ErrorInvalidIoChannel
ErrorUndefinedParameter
ErrorConfigIoctl
```

AI_9812_Config

Description

Informs PCIS-DASK library of the trigger source, trigger mode, and trigger properties selected for the PCI-9812 card with card ID CardNumber. You must call this function before calling function to perform analog input operation.

Supported card(s)

9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9812_Config (U16 CardNumber, U16 TrgMode,
     U16 TrgSrc, U16 TrgPol, U16 ClkSel, U16
     TrgLevel, U16 PostCnt)
```

Visual Basic

```
AI_9812_Config (ByVal CardNumber As Integer,
     ByVal TrgMode As Integer, ByVal TrgSrc As
     Integer, ByVal TrgPol As Integer, ByVal
     ClkSel As Integer, ByVal TrgLevel As
     Integer, ByVal PostCnt As Integer) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
TrgMode	A/D trigger mode setting. Valid trigger modes:

`P9812_TRGMOD_SOFT`	Software Trigger (no trigger)
`P9812_TRGMOD_POST`	Post Trigger
`P9812_TRGMOD_PRE`	Pre-Triger Mode
`P9812_TRGMOD_DELAY`	Delay Trigger
`P9812_TRGMOD_MIDL`	Middle Triger

TrgSrc A/D trigger source setting. Valid trigger sources:

`P9812_TRGSRC_CH0`	Channel 0
`P9812_TRGSRC_CH1`	Channel 1
`P9812_TRGSRC_CH2`	Channel 2
`P9812_TRGSRC_CH3`	Channel 3
`P9812_TRGSRC_EXT_DIG`	External digital trigger

TrgPol Trigger polarity settings. Valid values:

`P9812_TRGSLP_POS`	Positive slope trigger
`P9812_TRGSLP_NEG`	Negative slope trigger

ClkSel A/D clock source setting. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are two groups of constants:

A/D Clock Frequency

`P9812_AD2_GT_PCI`	A/D clock frequency is higher than PCI clock frequency.
`P9812_AD2_LT_PCI`	A/D clock frequency is lower than PCI clock frequency.

ADC clock source

`P9812_CLKSRC_INT`	Internal clock
`P9812_CLKSRC_EXT_SIN`	External sin wave clock
`P9812_CLKSRC_EXT_DIG`	External square wave clock

When two constants are used to form the ClkSel argument, the constants are combined with the bit-wise-OR operator(|).

NOTE:

If the ADC clock source is P9812_CLKSRC_EXT_DIG or P9812_CLKSRC_EXT_SIN, the clock divider is a constant, 2. Hence, the sampling rate is the half of the frequency of the source clock.

TrgLevel The setting of trigger level. The relationship between the value of TrgLevel and trigger voltage is listed in the following table:

TrgLevel	trigger voltage (±1V)	trigger voltage (±5V)
0xFF	0.992V	4.96V
0xFE	0.984V	4.92V
---	---	---
0x81	0.008V	0.04V
0x80	0.000V	0.00V
0x7F	-0.008V	-0.04V
---	---	---
0x01	-0.992V	-4.96V
0x00	-1.000V	-5.00V

PostCnt The post count value setting for Middle Trigger mode or Delay Trigger mode. This argument is expressed as follows:

For Middle Trigger mode, the number of data accessed for each selected channel after a specific trigger event.

For Delay Trigger mode, the counter value for deferring to access data after a specific trigger event

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_9812_SetDiv

Description

If the A/D trigger mode is set as external trigger by calling AI_9812_Config(), this function can be called to set the clock divider. The clock divider for external trigger mode of continuous AI is two (2) in driver by default.

Supported card(s)

9812/9810

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_9812_SetDiv (U16 CardNumber, U32 PacerVal)
```

Visual Basic

```
AI_9812_SetDiv (ByVal CardNumber As Integer,
     ByVal PacerVal As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
PacerVal	The length of the clock divider. The value has to be an even number. Range: 2 through 65534.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_AsyncCheck

Description

Checks the current status of the asynchronous analog input operation.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_AsyncCheck (U16 CardNumber, BOOLEAN
     *Stopped, U32 *AccessCnt)
```

Visual Basic

```
AI_AsyncCheck (ByVal CardNumber As Integer,
     Stopped As Byte, AccessCnt As Long) As
     Integer
```

Parameter(s)

CardNumber	The ID of the card performing asynchronous operation.
Stopped	Whether the asynchronous analog input operation has completed. If Stopped = TRUE, the analog input operation has stopped. Either the number of A/D conversions indicated in the call that initiated the asynchronous analog input operation has completed or an error has occurred. If Stopped = FALSE, the operation is not yet complete (constants TRUE and FALSE are defined in DASK.H).
AccessCnt	In the condition that the trigger acquisition mode is not used, AccessCnt returns the number of A/D data that has been transferred at the time calling AI_AsyncCheck(). If any trigger mode is enabled by calling AI_9111_Config(), AI_9812_Config(), or AI_9118_Config(), and double-buffered mode is enabled, AccessCnt returns the next position after

the position the last A/D data is stored in the circular buffer at the time calling AI_AsyncCheck().

Return Code(s)

```
NoError, ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_AsyncClear

Description

Stops the asynchronous analog input operation.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_AsyncClear (U16 CardNumber, U32
     *AccessCnt)
```

Visual Basic

```
AI_AsyncClear (ByVal CardNumber As Integer,
     AccessCnt As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing asynchronous operation.
AccessCnt	In the condition that the trigger acquisition mode is not used, AccessCnt returns the number of A/D data that has been transferred at the time calling AI_AsyncClear(). If double-buffered mode is enabled, AccessCnt returns the next position after the position the last A/D data is stored in the circular buffer. If the AccessCnt execeeds the half size of circular buffer, call AI_AsyncDblBufferTransfer twice to get the data.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_AsyncDblBufferHalfReady

Description

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

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_AsyncDblBufferHalfReady (U16 CardNumber,
     BOOLEAN *HalfReady, BOOLEAN *StopFlag)
```

Visual Basic

```
AI_AsyncDblBufferHalfReady(ByVal CardNumber As
     Integer, HalfReady As Byte, StopFlag As
     Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the asynchronous double-buffered operation.
HalfReady	Whether the next half buffer of data is available. If HalfReady = TRUE, you can call AI_AsyncDblBufferTransfer() to copy the data to your user buffer (constants TRUE and FALSE are defined in DASK.H).
StopFlag	Whether the asynchronous analog input operation has completed. If StopFlag = TRUE, the analog input operation has stopped. If StopFlag = FALSE, the operation is not yet complete (constants TRUE and FALSE are defined in DASK.H).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_AsyncDblBufferHandled

Description

Notifies PCIS-DASK the ready buffer has been handled in user application. For PCI-9221/9222/9223/9524, the data are transferred through DMA to the user's buffer directly. Therefore, while half buffer of data is ready (using AI_AsyncDblBufferHalfReady to check the ready status), the data in the ready buffer can be handled directly and don't needed to be copied to another transfer buffer. This mechanism eliminates the time taken for memory copy and another memory space for data transfer; however, PCIS-DASK couldn't know if the data in the ready buffer have been handled (in user application). If the data is handled, the user application needs an interface to notify PCIS-DASK this information. The new function AI_AsyncDblBufferHandled is used to for this purpose.

Supported card(s)

9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_AsyncDblBufferHandled (U16 CardNumber)
```

Visual Basic

```
AI_AsyncDblBufferHandled (ByVal CardNumber As
     Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorContIoNotAllowed
ErrorNotDoubleBufferMode
```

AI_AsyncDblBufferMode

Description

Enables or disables double-buffered data acquisition mode.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_AsyncDblBufferMode (U16 CardNumber,
     BOOLEAN Enable)
```

Visual Basic

```
AI_AsyncDblBufferMode (ByVal CardNumber As
     Integer, ByVal Enable As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card setting the double-buffered mode.
Enable	Enables or disables the double-buffered mode. Constants TRUE and FALSE are defined in DASK.H.

TRUE	Double-buffered mode is enabled.
FALSE	Double-buffered mode is disabled.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_AsyncDblBufferOverrun

Description

Checks or clears overrun status of the double-buffered analog input operation.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_AsyncDblBufferOverrun (U16 CardNumber, U16
     op, U16 *overrunFlag)
```

Visual Basic

```
AI_AsyncDblBufferOverrun (ByVal CardNumber As
     Integer, ByVal op As Integer, overrunFlag As
     Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card setting the double-buffered mode.
op	Check/Clear overrun status/flag.

`0`	Check the overrun status.
`1`	Clear the overrun flag.

overrunFlag	Returned overrun status

`0`	No overrun occurred.
`1`	Overrun occurred.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_AsyncDblBufferToFile

Description

For double buffer mode of continuous AI, if the continuous AI function is:

```
AI_ContReadChannelToFile
AI_ContReadMultiChannelsToFile, and
AI_ContScanChannelsToFile
```

call this function to log the data of the circular buffer into a disk file.

Supported card(s)

9221, 9524, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_AsyncDblBufferToFile (U16 CardNumber)
```

Visual Basic

```
AI_AsyncDblBufferToFile (ByVal CardNumber As
     Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorContIoNotAllowed
ErrorNotDoubleBufferMode
```

AI_AsyncReTrigNextReady

Description

Checks whether the data associated to the next trigger signal is ready during an asynchronous retriggered analog input operation.

Supported card(s)

9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_AsyncReTrigNextReady (U16 CardNumber,
     BOOLEAN *Ready, BOOLEAN *StopFlag, U16
     *RdyTrigCnt)
```

Visual Basic

```
AI_AsyncReTrigNextReady (ByVal CardNumber As
     Integer, Ready As Byte, StopFlag As Byte,
     RdyTrigCnt As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Ready	Tells whether the data associated with the next trigger signal is available. Constants TRUE and FALSE are defined in DASK.H.
StopFlag	Tells whether the asynchronous analog input operation is complete. If StopFlag is TRUE, the analog input operation has stopped. If StopFlag is FALSE, the operation is not yet completed. Constants TRUE and FALSE are defined in DASK.H.
RdyTrigCnt	This argument returns the count of trigger signal that occurred if re-trigger count is definite. If the re-trigger count is infinite, this argument returns the index of the buffer that stored the data after the most recent trigger signal trigger is generated.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_AsyncDblBufferTransfer

Description

Depending on the continuous AI function selected, half of the data of the circular buffer will be logged into the user buffer, if continuous AI function is: AI_ContReadChannel, AI_ContReadMultiChannels, or AI_ContScanChannels, or a disk file, if continuous AI function is AI_ContReadChannelToFile, AI_ContReadMultiChannelsToFile, AI_ContScanChannelsToFile. You can execute this function repeatedly to return sequential half buffers of the data.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_AsyncDblBufferTransfer (U16 CardNumber,
     U16 *Buffer)
```

Visual Basic

```
AI_AsyncDblBufferTransfer (ByVal CardNumber As
     Integer, Buffer As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the asynchronous double-buffered operation.
Buffer	The user buffer. An integer array to which the data is to be copied. If the data will be saved to a disk file, this argument is of no use. Refer to Appendix C: AI Data Format for the data format in Buffer or the data file.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorNotDoubleBufferMode
ErrorInvalidSampleRate
```

AI_ContBufferReset

Description

This function resets all the buffers set by function AI_ContBufferSetup for continuous analog input. The function has to be called if the data buffers won't be used.

Supported card(s)

9221, 9524, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ContBufferReset (U16 CardNumber)
```

Visual Basic

```
AI_ContBufferReset (ByVal CardNumber As Integer)
     As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorContIoNotAllowed
```

AI_ContBufferSetup

Description

This function setups the buffer for continuous analog input. The function has to be called repeatedly to setup all of the data buffers (at most 2 buffers). For double buffer mode of continuous AI, AI_ContBufferSetup should be called twice to setup the ring buffer to store the data.

Supported card(s)

9221, 9524, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ContBufferSetup (U16 CardNumber, void
     *Buffer, U32 ReadCount, U16 *BufferId)
```

Visual Basic

```
AI_ContBufferSetup (ByVal CardNumber As Integer,
     Buffer As Any, ByVal ReadCount As Long,
     BufferId As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Buffer	The starting address of the memory to contain the input data.
ReadCount	The size (in samples) of the buffer and its value must be even.
BufferId	Returns the index of the buffer currently set up.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
```

AI_ContReadChannel

Description

Performs continuous A/D conversions on the specified analog input channel at a rate closest to the specified rate.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ContReadChannel (U16 CardNumber, U16
     Channel, U16 AdRange, U16 *Buffer, U32
     ReadCount, F32 SampleRate, U16 SyncMode)
```

Visual Basic

```
AI_ContReadChannel (ByVal CardNumber As Integer,
     ByVal Channel As Integer, ByVal AdRange As
     Integer, Buffer As Integer, ByVal ReadCount
     As Long, ByVal SampleRate As Single, ByVal
     SyncMode As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Channel	Analog input channel number. Range:

PCI-9111	`0 to 15`
PCI-9112/ cPCI-9112	`0 to 15`
PCI-9113	`0 to 31`
PCI-9114	`0 to 31`
cPCI-9116	`0 to 63`
PCI-9118	`0 to 15`
PCI-9221	`0 to 15`
PCI-9222	`0 to 15`
PCI-9223	`0 to 31`

PCI-9524	`P9524_AI_LC_CH0`
	`P9524_AI_LC_CH1`
	`P9524_AI_LC_CH2`
	`P9524_AI_LC_CH3`
	`P9524_AI_GP_CH0`
	`P9524_AI_GP_CH1`
	`P9524_AI_GP_CH2`
	`P9524_AI_GP_CH3`

PCI-9812/10	`0`

AdRange The analog input range setting of the specified channel. We define some constants to represent various A/D input ranges in DASK.H. Refer to Appendix B: AI Range Codes, for the valid range values.

Buffer An integer array to contain the acquired data. Buffer must has a length equal to or greater than the value of Parameter(s) ReadCount. If double-buffered mode is enabled, this buffer is of no use, you can ignore this argument. Refer to Appendix C: AI Data Format for the data format in Buffer.

For PCI-9221/9222/9223/9524, this parameter means the Buffer ID returned by the function AI_ContBufferSetup.

ReadCount If double-buffered mode is disabled, ReadCount is the total number of A/D conversions (except cPCI-9116) or the total number of scans (for cPCI-9116) to be performed. For double-buffered acquisition, ReadCount is the size (in samples) of the circular buffer (except cPCI-9116) or the size (in samples) allocated for each channel in the circular buffer (for cPCI-9116) and its value must be a multiple of 4.

NOTE: For PCI-9111, PCI-9113 or PCI-9114 cards, this function uses FIFO-Half-Full interrupt transfer mode. The value of ReadCount must be the multiple of 512 for non-double-buffer mode or multiple of 1024 for double-buffer mode.

SampleRate The sampling rate you want for analog input in hertz (samples per second). Your maximum rate depends on the card type and your computer system.

On **PCI-9524**, it only supports 16 sample rates:
P9524_ADC_30K_SPS
P9524_ADC_15K_SPS
P9524_ADC_7K5_SPS
P9524_ADC_3K75_SPS
P9524_ADC_2K_SPS
P9524_ADC_1K_SPS
P9524_ADC_500_SPS
P9524_ADC_100_SPS
P9524_ADC_60_SPS
P9524_ADC_50_SPS
P9524_ADC_30_SPS
P9524_ADC_25_SPS
P9524_ADC_15_SPS
P9524_ADC_10_SPS
P9524_ADC_5_SPS
P9524_ADC_2R5_SPS

On cPCI9116, this parameter is ignored. Use AI_9116_CounterInterval() to set the scan rate.

On PCI-9221, this parameter is ignored. Use AI_9221_CounterInterval() to set the scan rate.

On PCI-9222, this parameter is ignored. Use AI_9222_CounterInterval() to set the scan rate.

On PCI-9223, this parameter is ignored. Use AI_9223_CounterInterval() to set the scan rate.

If you set A/D trigger mode as external trigger by calling AI_9111_Config(), AI_9112_Config(), AI_9113_Config(), AI_9114_Config(), AI_9812_Config(), or AI_9118_Config(), the sampling rate is determined by an external trigger source, you have to set this argument as CLKSRC_EXT_SampRate.

If you set A/D trigger mode as external trigger by calling AI_9812_Config(), the frequency divider can be set by calling AI_9812_SetDiv(). If the function, AI_9812_SetDiv() is not called, the frequency divider is set as 2 by the driver by default. Hence, the sampling rate is: Frequency of external clock source/2.

SyncMode Tells whether the operation is performed synchronously or asynchronously. If any trigger mode is enabled by calling AI_9111_Config(), AI_9812_Config(), AI_9116_Config(), or AI_9118_Config(), this operation should be performed asynchronously. Valid values:

`SYNCH_OP`	Synchronous A/D conversion, that is, the function does not return until the A/D operation is completed.
`ASYNCH_OP`	Asynchronous A/D conversion

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorInvalidAdRange
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
ErrorInvalidSampleRate
ErrorInvalidBufferID
ErrorInvalidCounterState
```

AI_ContReadChannelToFile

Description

Performs continuous A/D conversions on the specified analog input channel at a rate closest to the specified rate and saves the acquired data in a disk file. The data is written to disk in binary format, with the lower byte first (little endian). Refer to Appendix D: Data File Format for the data file structure and Appendix C: AI Data Format for the format of the data in the data file.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ContReadChannelToFile (U16 CardNumber, U16
     Channel, U16 AdRange, U8 *FileName, U32
     ReadCount, F64 SampleRate, U16 SyncMode);
```

Visual Basic

```
AI_ContReadChannelToFile (ByVal CardNumber As
     Integer, ByVal Channel As Integer, ByVal
     AdRange As Integer, ByVal FileName As
     String, ByVal ReadCount As Long, ByVal
     SampleRate As Double, ByVal SyncMode As
     Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Channel Analog input channel number. Range:

PCI-9111	`0 through 15`
PCI-9112/ cPCI-9112	`0 through 15`
PCI-9113	`0 through 31`
PCI-9114	`0 through 31`
cPCI-9116	`0 through 63`
PCI-9118	`0 through 15`
PCI-9221	`0 through 15`

PCI-9222	`0 through 15`
PCI-9223	`0 through 31`

PCI-9524	`P9524_AI_LC_CH0`
	`P9524_AI_LC_CH1`
	`P9524_AI_LC_CH2`
	`P9524_AI_LC_CH3`
	`P9524_AI_GP_CH0`
	`P9524_AI_GP_CH1`
	`P9524_AI_GP_CH2`
	`P9524_AI_GP_CH3`

PCI-9812/10	`0`

AdRange The analog input range setting of the specified channel. We define some constants to represent various A/D input ranges in DASK.H. Refer to Appendix B: AI Range Codes.

FileName The file where acquired data is stored.

ReadCount If double-buffered mode is disabled, ReadCount is the number of A/D conversions (except cPCI-9116) or the total number of scans (for cPCI-9116) to be performed. For double-buffered acquisition, ReadCount is the size (in samples) of the circular buffer (except cPCI-9116) or the size (in samples) allocated for each channel in the circular buffer (for cPCI-9116) and its value must be a multiple of four.

NOTE: For PCI-9111, PCI-9113 or PCI-9114 cards, this function uses FIFO-Half-Full interrupt transfer mode. So the value of ReadCount must be the multiple of 512 for non-double-buffer mode, or multiple of 1024 for double-buffer mode.

SampleRate The sampling rate you want for analog input in hertz (samples per second). Your maximum rate depends on the card type and your computer system.

On **PCI-9524**, it only supports 16 sample rates:
P9524_ADC_30K_SPS
P9524_ADC_15K_SPS
P9524_ADC_7K5_SPS
P9524_ADC_3K75_SPS

	P9524_ADC_2K_SPS P9524_ADC_1K_SPS P9524_ADC_500_SPS P9524_ADC_100_SPS P9524_ADC_60_SPS P9524_ADC_50_SPS P9524_ADC_30_SPS P9524_ADC_25_SPS P9524_ADC_15_SPS P9524_ADC_10_SPS P9524_ADC_5_SPS P9524_ADC_2R5_SPS
	On cPCI-9116, this parameter is ignored. Use AI_9116_CounterInterval() to set the scan rate.
	On PCI-9221, this parameter is ignored. Use AI_9221_CounterInterval() to set the scan rate.
	On PCI-9222, this parameter is ignored. Use AI_9222_CounterInterval() to set the scan rate.
	On PCI-9223, this parameter is ignored. Use AI_9223_CounterInterval() to set the scan rate.
	If you set A/D trigger mode as external trigger by calling AI_9111_Config(), AI_9112_Config(), AI_9113_Config(),AI_9114_Config(), AI_9812_Config() or AI_9118_Config(), the sampling rate is determined by an external trigger source, you have to set this argument as CLKSRC_EXT_SampRate.
	If you set A/D trigger mode as external trigger by calling AI_9812_Config(), the frequency divider can be set by calling AI_9812_SetDiv(). If the function, AI_9812_SetDiv() is not called, the frequency divider is set as 2 by the driver by default. Hence, the sampling rate is the frequency of external clock source divided by 2.
SyncMode	Tells whether the operation is performed synchronously or asynchronously. If any trigger mode is enabled by calling AI_9111_Config(),

AI_9116_Config(), AI_9812_Config(), or AI_9118_Config(), this operation should be performed asynchronously. Valid values:

`SYNCH_OP`	Synchronous A/D conversion, that is, the function does not return until the A/D operation is completed.
`ASYNCH_OP`	Asynchronous A/D conversion

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorInvalidAdRange
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
ErrorInvalidSampleRate
ErrorOpenFile
ErrorInvalidCounterState
```

AI_ContReadMultiChannels

Description

Performs continuous A/D conversions on the specified analog input channels at a rate closest to the specified rate. This function takes advantage of the cPCI-9116, PCI-9118 and PCI-9221 auto-scan and channel-gain queue functionalities to perform multi-channel analog input.

Supported card(s)

9116, 9118, 9221, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ContReadMultiChannels (U16 CardNumber, U16
     numChans, U16 *Chans, U16 *AdRanges, U16
     *Buffer, U32 ReadCount, F32 SampleRate, U16
     SyncMode)
```

Visual Basic

```
AI_ContReadMultiChannels (ByVal CardNumber As
     Integer, ByVal numChans As Integer, Chans As
     Integer, AdRanges As Integer, Buffer As
     Integer, ByVal ReadCount As Long, ByVal
     SampleRate As Single, ByVal SyncMode As
     Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
numChans	The number of analog input channels in the array Chans. Valid values:

cPCI-9116	`1 to 511`
PCI-9118	`1 to 255`
PCI-9221	`1 to 16`
PCI-9222	`1 to 16`
PCI-9223	`1 to 32`

Chans	Array of analog input channel numbers. The channel order for acquiring data is the same as the order you set in Chans.

cPCI-9116	Numbers in Chans must be within 0 and 63. Since there is no restriction for channel order setting, you may set the channel order as you want.
PCI-9118	Numbers in Chans must be within 0 and 15. Since there is no restriction for channel order setting, you may set the channel order as you want.
PCI-9221	Numbers in Chans must be within 0 and 15. Since there is no restriction for channel order setting, you may set the channel order as you want.
PCI-9222	Numbers in Chans must be within 0 and 15. Since there is no restriction for channel order setting, you may set the channel order as you want.
PCI-9223	Numbers in Chans must be within 0 and 31. Since there is no restriction for channel order setting, you may set the channel order as you want.

AdRanges	An integer array of length numChans that contains the analog input range for every channel in array Chans.

NOTE: For cPCI-9116/PCI-9118/PCI-9221/PCI-9222/PCI-9223, refer to Appendix B: AI Range Codes for the valid range values. Since cPCI-9116/PCI-9118/PCI-9221/PCI-9222/PCI-9223 supports different ranges, the range values in AdRanges can be any of the valid range values of cPCI-9116/PCI-9118/PCI-9221/PCI-9222/PCI-9223.

Buffer	An integer array to contain the acquired data. The length of Buffer must be equal to or greater than the value of the ReadCount parameter. The acquired data is stored in interleaved sequence. For example, if the value of numChans is 3, and the numbers in Chans are 3, 8, and 0, then this function input data from channel 3, then channel 8, then channel 0, then channel 3, then channel 8, so on and so forth. The data acquired is put to Buffer by order, so the data read from channel 3 is stored in Buffer[0], Buffer[3], Buffer[6], so on and so forth. The data from channel 8 is stored in Buffer[1], Buffer[4], Buffer[7], so on and

	so forth. The data from channel 0 is stored in Buffer[2], Buffer[5], Buffer[8], so on and so forth. If double-buffered mode is enabled, this buffer has no use and you may ignore this argument. Refer to Appendix C: AI Data Format for the data format in Buffer.
	For PCI-9221/9222/9223, this parameter means the Buffer ID returned by the function AI_ContBufferSetup.
ReadCount	If double-buffered mode is disabled, ReadCount is the number of A/D conversions (for PCI-9118/9221/9222/9223) or the total number of scans (for cPCI-9116) to be performed. For double-buffered acquisition, ReadCount is the size (in samples) of the circular buffer (for PCI-9118/9221/9222/9223) or the size (in samples) allocated for each channel in the circular buffer (for cPCI-9116) and its value must be a multiple of four.
SampleRate	The sampling rate you want for analog input in hertz (samples per second). The maximum rate depends on the card type and your computer system.
	On cPCI-9116, this parameter is ignored. Use AI_9116_CounterInterval() to set the scan rate.
	On PCI-9221, this parameter is ignored. Use AI_9221_CounterInterval() to set the scan rate.
	On PCI-9222, this parameter is ignored. Use AI_9222_CounterInterval() to set the scan rate.
	On PCI-9223, this parameter is ignored. Use AI_9223_CounterInterval() to set the scan rate.
	If you set A/D trigger source as external trigger by calling AI_9118_Config(), the sampling rate is determined by an external trigger source, you have to set this argument as CLKSRC_EXT_SampRate.
SyncMode	Tells whether the operation is performed synchronously or asynchronously. If any trigger mode is enabled by calling AI_9118_Config() or

AI_9116_Config(),this operation should be performed asynchronously. Valid values:

`SYNCH_OP`	Synchronous A/D conversion, that is, the function does not return until the A/D operation is completed.
`ASYNCH_OP`	Asynchronous A/D conversion

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorInvalidSampleRate
ErrorInvalidAdRange
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
ErrorInvalidBufferID
ErrorInvalidCounterState
```

AI_ContReadMultiChannelsToFile

Description

Performs continuous A/D conversions on the specified analog input channels at a rate closest to the specified rate and saves the acquired data in a disk file. The data is written to disk in binary format, with the lower byte first (little endian). Refer to Appendix D: Data File Format for the data file structure and Appendix C: AI Data Format for the format of the data in the data file. This function takes advantage of the PCI-9118 auto-scan and channel-gain queue functionality to perform multi-channel analog input.

Supported card(s)

9116, 9118, 9221, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ContReadMultiChannelsToFile (U16
     CardNumber, U16 numChans, U16 *Chans, U16
     *AdRanges, U8 *FileName, U32 ReadCount, F64
     SampleRate, U16 SyncMode)
```

Visual Basic

```
AI_ContReadMultiChannelsToFile (ByVal CardNumber
     As Integer, ByVal numChans As Integer, Chans
     As Integer, AdRanges As Integer, ByVal
     FileName As String, ByVal ReadCount As Long,
     ByVal SampleRate As Double, ByVal SyncMode
     As Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

numChans The number of analog input channels in the array Chans. Valid values:

cPCI-9116	`1 to 511`
PCI-9118	`1 to 255`
PCI-9221	`1 to 16`
PCI-9222	`1 to 16`
PCI-9223	`1 to 32`

Chans	Array of analog input channel numbers. The channel order for acquiring data is the same as the order you set in Chans.

cPCI-9116	Numbers in Chans must be within 0 and 63. Since there is no restriction for channel order setting, you may set the channel order as you want.
PCI-9118	Numbers in Chans must be within 0 and 15. Since there is no restriction for channel order setting, you may set the channel order as you want.
PCI-9221	Numbers in Chans must be within 0 and 15. Since there is no restriction for channel order setting, you may set the channel order as you want.
PCI-9222	Numbers in Chans must be within 0 and 15. Since there is no restriction for channel order setting, you may set the channel order as you want.
PCI-9223	Numbers in Chans must be within 0 and 31. Since there is no restriction for channel order setting, you may set the channel order as you want.

AdRanges	An integer array of length numChans that contains the analog input range for every channel in array Chans.

NOTE:

For cPCI-9116/PCI-9118/PCI-9221/PCI-9222/PCI-9223, refer to Appendix B: AI Range Codes for valid range values. Since cPCI-9116/PCI-9118/PCI-9221/PCI-9222/PCI-9223 supports different ranges, the range values in AdRanges can be any of the valid range values of cPCI-9116/PCI-9118/PCI-9221/PCI-9222/PCI-9223.

FileName	The file where acquired data is stored.
ReadCount	If double-buffered mode is disabled, ReadCount is the number of A/D conversions (for PCI-9118/PCI-9221/PCI-9222/PCI-9223) or the total number of scans (for cPCI-9116) to be performed. For double-buffered acquisition, ReadCount is the size (in samples) of the circular buffer (for PCI-9118/PCI-9221/PCI-9222/PCI-9223) or the size (in samples) allocated for each channel in the circular buffer (for cPCI-9116) and its value must be a multiple of four.

SampleRate	The sampling rate you want for analog input in hertz (samples per second). The maximum rate depends on the card type and your computer system. On cPCI-9116, this parameter is ignored. Use AI_9116_CounterInterval() to set the scan rate. On PCI-9221, this parameter is ignored. Use AI_9221_CounterInterval() to set the scan rate. On PCI-9222, this parameter is ignored. Use AI_9222_CounterInterval() to set the scan rate. On PCI-9223, this parameter is ignored. Use AI_9223_CounterInterval() to set the scan rate. If you set A/D trigger source as external trigger by calling AI_9118_Config(), the sampling rate is determined by an external trigger source, you have to set this argument as CLKSRC_EXT_SampRate.
SyncMode	Tells whether the operation is performed synchronously or asynchronously. If any trigger mode is enabled by calling AI_9118_Config(), this operation should be performed asynchronously. Valid values:

`SYNCH_OP`	Synchronous A/D conversion, that is, the function does not return until the A/D operation is completed.
`ASYNCH_OP`	Asynchronous A/D conversion

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorInvalidSampleRate
ErrorInvalidAdRange
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
ErrorOpenFile
ErrorInvalidCounterState
```

AI_ContScanChannels

Description

Performs continuous A/D conversions on the specified continuous analog input channels at a rate closest to the specified rate. This function takes advantage of the hardware autoscan functionality to perform multi-channel analog input.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ContScanChannels (U16 CardNumber, U16
     Channel, U16 AdRange, U16 *Buffer, U32
     ReadCount, F64 SampleRate, U16 SyncMode)
```

Visual Basic

```
AI_ContScanChannels (ByVal CardNumber As Integer,
     ByVal Channel As Integer, ByVal AdRange As
     Integer, Buffer As Integer, ByVal ReadCount
     As Long, ByVal SampleRate As Double, ByVal
     SyncMode As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Channel	The largest channel number of specified continuous analog input channel. The channel order for acquiring data is as follows:

PCI-9111	Number of channels must be within 0 and 15. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
PCI-9112/ cPCI-9112	Number of channels must be within 0 and 15. The continuous scan sequence is descending, and the first one must be zero. For example: 3, 2, 1, 0.
PCI-9113	Number of channels must be within 0 and 31. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.

PCI-9114	Number of channels must be within 0 and 31. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
cPCI-9116	Number of channels must be within 0 and 63. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
PCI-9118	Number of channels must be within 0 and 15. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
PCI-9221	Number of channels must be within 0 and 15. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
PCI-9222	Number of channels must be within 0 and 15. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
PCI-9223	Number of channels must be within 0 and 31. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.

PCI-9524	`P9524_AI_LC_CH0`
	`P9524_AI_LC_CH1`
	`P9524_AI_LC_CH2`
	`P9524_AI_LC_CH3`
	`P9524_AI_GP_CH0`
	`P9524_AI_GP_CH1`
	`P9524_AI_GP_CH2`
	`P9524_AI_GP_CH3`

PCI-9812/10	Number of channel must be 0, 1 or 3. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.

AdRange The analog input range setting of the specified channel. Refer to Appendix B: AI Range Codes.

Buffer An integer array to contain the acquired data. The length of Buffer must be equal to or greater than the value of the ReadCount parameter. The acquired data is stored in interleaved sequence. For example, if the Channel value is 3 and the scanned channel numbers is descending (e.g. PCI-9112/cPCI-9112), then this function input data from channel 2, then channel 1, then channel 0, then channel 2, then channel 1, so on and so forth. The data acquired is put to Buffer by order, so the data read from channel 2 is stored in Buffer[0], Buffer[3], Buffer[6], so on and

	so forth. The data from channel 1 is stored in Buffer[1], Buffer[4], Buffer[7], so on and so forth. The data from channel 0 is stored in Buffer[2], Buffer[5], Buffer[8], so on and so forth. If double-buffered mode is enabled, this buffer has no use and you may ignore this argument. Refer to Appendix C: AI Data Format for the data format in Buffer. For PCI-9221/9222/9223/9524, this parameter means the Buffer ID returned by the function AI_ContBufferSetup.
ReadCount	If double-buffered mode is disabled, this is the number of A/D conversions (except cPCI-9116) or the total number of scans (for cPCI-9116) to be performed. For double-buffered acquisition, ReadCount is the size (in samples) of the circular buffer (except cPCI-9116) or the size (in samples) allocated for each channel in the circular buffer (for cPCI-9116) and its value must be a multiple of four.

NOTE: For PCI-9111, PCI-9113 or PCI-9114 card, this function uses FIFO-Half-Full interrupt transfer mode. So the value of ReadCount must be the multiple of 512 for non-double-buffer mode or multiple of 1024 for double-buffer mode.

SampleRate	The sampling rate you want for analog input in hertz (samples per second). The maximum rate depends on the card type and your computer system. On **PCI9524**, it only supports 16 sample rates: P9524_ADC_30K_SPS P9524_ADC_15K_SPS P9524_ADC_7K5_SPS P9524_ADC_3K75_SPS P9524_ADC_2K_SPS P9524_ADC_1K_SPS P9524_ADC_500_SPS P9524_ADC_100_SPS P9524_ADC_60_SPS P9524_ADC_50_SPS

P9524_ADC_30_SPS
P9524_ADC_25_SPS
P9524_ADC_15_SPS
P9524_ADC_10_SPS
P9524_ADC_5_SPS
P9524_ADC_2R5_SPS

On cPCI-9116, this parameter is ignored. Use AI_9116_CounterInterval() to set the scan rate.

On PCI-9221, this parameter is ignored. Use AI_9221_CounterInterval() to set the scan rate.

On PCI-9222, this parameter is ignored. Use AI_9222_CounterInterval() to set the scan rate.

On PCI-9223, this parameter is ignored. Use AI_9223_CounterInterval() to set the scan rate.

If you set A/D trigger mode as external trigger by calling AI_9111_Config(), AI_9112_Config(), AI_9113_Config(),AI_9114_Config(), AI_9812_Config() or AI_9118_Config(), the sampling rate is determined by an external trigger source, you have to set this argument as CLKSRC_EXT_SampRate.

If you set A/D trigger mode as external trigger by calling AI_9812_Config(), the frequency divider can be set by calling AI_9812_SetDiv(). If the function, AI_9812_SetDiv() is not called, the frequency divider is set as 2 by the driver by default. Hence, the sampling rate is the frequency of external clock source divided by two.

SyncMode Tells whether the operation is performed synchronously or asynchronously. If any trigger mode is enabled by calling AI_9111_Config(), AI_9116_Config(), AI_9812_Config() or AI_9118_Config(), this operation should be performed asynchronously. Valid values:

`SYNCH_OP`	Synchronous A/D conversion, that is, the function does not return until the A/D operation is completed.

`ASYNCH_OP`	Asynchronous A/D conversion

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorInvalidSampleRate
ErrorInvalidAdRange
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
ErrorLastChannelNotZero
ErrorDiffRangeNotSupport
ErrorChannelNotDescending
ErrorChannelNotAscending
ErrorInvalidBufferID
ErrorInvalidCounterState
```

AI_ContScanChannelsToFile

Description

Performs continuous A/D conversions on the specified continuous analog input channels at a rate closest to the specified rate and saves the acquired data in a disk file. The data is written to disk in binary format, with the lower byte first (little endian). Refer to Appendix D: Data File Format for the data file structure and Appendix C: AI Data Format for the data format in the data file. This function takes advantage of the hardware autoscan functionality to perform multi-channel analog input.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ContScanChannelsToFile (U16 CardNumber,
     U16 Channel, U16 AdRange, U8 *FileName, U32
     ReadCount, F64 SampleRate, U16 SyncMode)
```

Visual Basic

```
AI_ContScanChannelsToFile (ByVal CardNumber As
     Integer, ByVal Channel As Integer, ByVal
     AdRange As Integer, ByVal FileName As
     String, ByVal ReadCount As Long, ByVal
     SampleRate As Double, ByVal SyncMode As
     Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Channel	The largest channel number of specified continuous analog input channel. The channel order for acquiring data is as follows:

PCI-9111	Number of channels must be within 0 and 15. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
PCI-9112/ cPCI-9112	Number of channels must be within 0 and 15. The continuous scan sequence is descending, and the first one must be zero. For example: 3, 2, 1, 0.

PCI-9113	Number of channels must be within 0 and 31. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
PCI-9114	Number of channels must be within 0 and 31. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
cPCI-9116	Number of channels must be within 0 and 63. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
PCI-9118	Number of channels must be within 0 and 15. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
PCI-9221	Number of channels must be within 0 and 15. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
PCI-9222	Number of channels must be within 0 and 15. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
PCI-9223	Number of channels must be within 0 and 31. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.

PCI-9524	`P9524_AI_LC_CH0`
	`P9524_AI_LC_CH1`
	`P9524_AI_LC_CH2`
	`P9524_AI_LC_CH3`
	`P9524_AI_GP_CH0`
	`P9524_AI_GP_CH1`
	`P9524_AI_GP_CH2`
	`P9524_AI_GP_CH3`

PCI-9812/10	Number of channel must be 0, 1 or 3. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.

AdRange The analog input range setting of the specified channel. Refer to Appendix B: AI Range Codes for the range of valid values.

FileName The file where acquired data is stored.

ReadCount If double-buffered mode is disabled, ReadCount is the number of A/D conversions (except cPCI-9116) or the total number of scans (for cPCI-9116) to be performed. For double-buffered acquisition, ReadCount is the size (in samples) of the circular buffer (except cPCI-9116) or the size (in samples) allocated for each channel in the circular buffer (for cPCI-9116) and its value must be a multiple of 4.

NOTE: For PCI-9111, PCI-9113 or PCI-9114 card, this function uses FIFO-Half-Full interrupt transfer mode. So the value of ReadCount must be the multiple of 512 for non-double-buffer mode, or multiple of 1024 for double-buffer mode.

SampleRate The sampling rate for analog input in hertz (samples per second). The maximum rate depends on the card type and your computer system.

On PCI9524, it only supports 16 sample rates:
P9524_ADC_30K_SPS
P9524_ADC_15K_SPS
P9524_ADC_7K5_SPS
P9524_ADC_3K75_SPS
P9524_ADC_2K_SPS
P9524_ADC_1K_SPS
P9524_ADC_500_SPS
P9524_ADC_100_SPS
P9524_ADC_60_SPS
P9524_ADC_50_SPS
P9524_ADC_30_SPS
P9524_ADC_25_SPS
P9524_ADC_15_SPS
P9524_ADC_10_SPS
P9524_ADC_5_SPS
P9524_ADC_2R5_SPS

On cPCI-9116, this parameter is ignored. Use AI_9116_CounterInterval() to set the scan rate.

On PCI-9221, this parameter is ignored. Use AI_9221_CounterInterval() to set the scan rate.

On PCI-9222, this parameter is ignored. Use AI_9222_CounterInterval() to set the scan rate.

On PCI-9223, this parameter is ignored. Use AI_9223_CounterInterval() to set the scan rate.

If you set A/D trigger mode as external trigger by calling AI_9111_Config(), AI_9112_Config(), AI_9113_Config(), AI_9114_Config(), AI_9812_Config(), or AI_9118_Config(), the sampling rate is determined by an external trigger source, you have to set this argument as CLKSRC_EXT_SampRate.

If you set A/D trigger mode as external trigger by calling AI_9812_Config(), the frequency divider can be set by calling AI_9812_SetDiv(). If the function, AI_9812_SetDiv() is not called, the frequency divider is set to 2 by default. Hence, the sampling rate is: Frequency of external clock source / 2.

SyncMode Tells whether the operation is performed synchronously or asynchronously. If any trigger mode is enabled by calling AI_9111_Config(), AI_9116_Config(), AI_9812_Config() or AI_9118_Config(), this operation should be performed asynchronously. Valid values:

`SYNCH_OP`	Synchronous A/D conversion, that is, the function does not return until the A/D operation is completed.
`ASYNCH_OP`	Asynchronous A/D conversion

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorInvalidSampleRate
ErrorInvalidAdRange
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
ErrorLastChannelNotZero
ErrorDiffRangeNotSupport
ErrorChannelNotDescending
ErrorChannelNotAscending
ErrorInvalidCounterState
```

AI_ContStatus

Description

While performing continuous A/D conversions, this function is called to get the A/D status. Please refer to the manual for your device for the AI status the device might meet.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ContStatus (U16 CardNumber, U16 *Status)
```

Visual Basic

```
AI_ContStatus (ByVal CardNumber As Integer,
     Status Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Status	The continuous AI status returned. The description of the Status parameter for various card types is listed.

PCI-9111/PCI-9113/PCI-9114

`bit 0`	0 = FIFO is empty.
`bit 1`	0 = FIFO is half-full.
`bit 2`	0 = FIFO is full; Data may have been lost.
`bit 3`	0 = AD is busy; The A/D data has not been latched into FIFO.
`bit 4~15`	Not used

PCI-9112

`bit 0`	1 = A/D conversion is completed (ready).
`bit 1`	1 = A/D conversion has overrun.
`bit 2~15`	Not used

cPCI-9116

`bit 0`	1 = A/D conversion has overspeed.
`bit 1`	1 = A/D conversion has overrun.
`bit 2`	1 = The scan counter counts to zero.

`bit 3`	1 = An external digital trigger event occurred.
`bit 4`	1 = A/D FIFO is empty.
`bit 5`	1 = A/D FIFO is half-full.
`bit 6`	0 = A/D FIFO is full.
`bit 7~15`	Not used

PCI-9118

`bit 0`	1 = A/D conversion is completed (ready).
`bit 1`	1 = A/D conversion has overrun.
`bit 2`	1 = A/D conversion has overspeed.
`bit 3`	1 = Burst mode of A/D conversion has overrun.
`bit 4`	1 = External digital trigger event occurred.
`bit 5`	1 = About Trigger of A/D conversion is completed.
`bit 6`	1 = A/D FIFO is empty.
`bit 7`	1 = FIFO is half-full.
`bit 8`	1 = FIFO is full.
`bit 9~15`	Not used

PCI-9221

`bit 0`	1 = FIFO is empty.
`bit 1`	1 = FIFO is almost empty.
`bit 2`	1 = FIFO is almost full.
`bit 3`	1 = FIFO is full.
`bit 4~7`	Not used
`bit 8`	1 = AI acquisition is in progress.
`bit 9`	1 = AI acquisition is done.
`bit 10`	1 = AI input exceeds the input limit.
`bit 11~15`	Not used

PCI-9222/9223

`bit 0`	1 = FIFO is empty.
`bit 1`	1 = FIFO is almost empty.
`bit 2`	1 = FIFO is almost full.
`bit 3`	1 = FIFO is full.
`bit 4~7`	Not used
`bit 8`	1 = AI acquisition is in progress.
`bit 9`	1 = AI acquisition is done.
`bit 10~15`	Not used

PCI-9524

`bit 0`	1 = FIFO of load cell group is empty.
`bit 1`	1 = FIFO of load cell group is almost empty.
`bit 2`	1 = FIFO of load cell group s almost full.
`bit 3`	1 = FIFO of load cell group is full.
`bit 4`	Not used.
`bit 5`	1 = AI acquisition of load cell group is in progress.
`bit 6~14`	Not used.
`bit 15`	1 = AI acquisition of load cell group is done.
`bit 16`	1 = FIFO of general purpose group is empty.
`bit 17`	1 = FIFO of general purpose group is almost empty.
`bit 18`	1 = FIFO of general purpose group is almost full.
`bit 19`	1 = FIFO of general purpose group is full.
`bit 20`	Not used.
`bit 21`	1 = AI acquisition of general purpose group is in progress.
`bit 22~30`	Not used.
`bit 31`	1 = AI acquisition of general purpose group is done.

PCI-9812

`bit 0`	1 = FIFO is ready for input (not full)
`bit 1`	1 = FIFO is at least half-full
`bit 2`	1 = FIFO is ready for output (not empty)
`bit 3`	1 = Post trigger counter reached zero
`bit 4`	Not used

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
```

AI_ContVScale

Description

Converts the values of an array of acquired binary data from an continuous A/D conversion call to actual input voltages. The acquired binary data in the reading array might include the channel information (refer to continuous functions including AI_ContReadChannel or AI_ContScanChannels, for the detailed data format). However, the calculated voltage values in the voltage array returned will not include the channel message.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ContVScale (U16 CardNumber, U16 AdRange,
     U16 *readingArray, F64 *voltageArray, I32
     count)
```

Visual Basic

```
AI_ContVScale (ByVal CardNumber As Integer, ByVal
     AdRange As Integer, readingArray As Integer,
     voltageArray As Double, ByVal count As Long)
     As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
AdRange	The analog input range setting of the specified channel. Refer to Appendix B: AI Range Codes for the range of valid values.
readingArray	Acquired continuous analog input data array.
voltageArray	Computed voltages array returned.
count	Data count that you want to convert.

Return Code(s)

```
NoError, ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidAdRange
```

AI_EventCallBack (Win32 Only)

Description

Controls and notifies the user's application when a specified DAQ event occurs. The notification is performed through a user-specified callback function. The event message will be removed automatically after calling AI_Async_Clear. The event message may be manually removed by setting the Mode parameter to 0.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++ and Borland C++

```
I16 AI_EventCallBack (U16 CardNumber, I16 mode,
     I16 EventType, U32 callbackAddr)
```

Visual Basic

```
AI_EventCallBack (ByVal CardNumber As Integer,
     ByVal mode As Integer, ByVal EventType As
     Integer, ByVal callbackAddr As Long) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
mode	Add or remove the event message. Valid values:

`0`	Remove
`1`	Add

EventType	Event criteria. Valid values:

`AIEnd`	Notification that the asynchronous analog input operation has been completed.
`DBEvent`	Notification that the next half buffer of data in circular buffer is ready for transfer.
`TrigEvent`	Notifies that the data associated to the next trigger signal is available (only for PCI-9222/9223).
`P9524_INT_LC_EOC`	Notifies that the a LoadCell-AI Conversion of PCI-9524 has completed (only for the PCI-9524 polling mode).

`P9524_INT_GP_EOC`	Notifies that the a GeneralPurpose-AI Conversion of PCI-9524 has completed (only for PCI-9524 polling mode).

callbackAddr Address of the user callback function. The PCIS-DASK calls this function when the specified event occurs. If you want to remove the event message, set callbackAddr to 0.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_GetView

Description

Returns the mapped buffer address of the memory allocated in the driver for continuous AI operation at system startup time. The allocated memory size may be acquired through the function AI_InitialMemoryAllocated. This function is not available for middle (about) -trigger or pre-trigger mode of single buffered continuous analog input operation.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_GetView(U16 CardNumber, U32 *pView)
```

Visual Basic

```
AI_GetView (ByVal CardNumber As Integer, pView As
     Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
pView	The mapped buffer address of the memory allocated in the driver at system startup.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
```

AI_InitialMemoryAllocated

Description

Returns the available memory size for analog input in the device driver using the argument MemSize. The continuous analog input transfer size may not exceed this size.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_InitialMemoryAllocated (U16 CardNumber,
     U32 *MemSize)
```

Visual Basic

```
AI_InitialMemoryAllocated (ByVal CardNumber As
     Integer, MemSize As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
MemSize	The available memory size for continuous AI in the card's device driver. The unit is KB (1024 bytes).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
```

AI_ReadChannel

Description

Performs a software triggered A/D conversion (analog input) on an analog input channel and returns the converted value.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ReadChannel (U16 CardNumber, U16 Channel,
     U16 AdRange, U16 *Value)
```

Visual Basic

```
AI_ReadChannel (ByVal CardNumber As Integer,
     ByVal Channel As Integer, ByVal AdRange As
     Integer, Value As Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Channel Analog input channel number. Range:

PCI/cPCI-9112, PCI-9111 PCI-9118	`0 to 15`
PCI-9113, PCI-9114	`0 to 31`
cPCI-9116	`0 to 63`
PCI-9221	`0 to 15`
PCI-9222	`0 to 15`
PCI-9223	`0 to 31`

AdRange The analog input range setting of the specified channel. We define some constants to represent various A/D input ranges in DASK.H. Refer to Appendix B: AI Range Codes.

Value The A/D converted value. The data format in value is described below:

PCI-9113	16-bit unsigned data:
	`B15 … B12 D11 D10 …. D1 D0`
	`where D11, D10,..., D0: A/D converted data`
	`B15 ~ B12: ignore`

PCI-9114
PCI-9221
PCI-9222
PCI-9223
	16-bit signed data:
	`D15 D14 ……………. D1 D0`
	`where D15, D14,..., D0: A/D converted data`

NOTE: For PCI-9111, PCI-9112/cPCI-9112, cPCI-9116, and PCI-9118 cards, refer to the description of Buffer argument of AI_ContReadChannel() for the correct data format.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorInvalidAdRange
```

AI_ReadChannel32

Description

Performs a software triggered A/D conversion (analog input) on an analog input channel and returns the 32-bits converted value. For PCI-9524, the ADC will be started while the AI_9524_PollConfig() function is performed. And then, you can use the function to get the analog value acquired with the set speed and range.

Supported card(s)

9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ReadChannel32 (U16 CardNumber, U16
     Channel, U16 AdRange, U32 *Value)
```

Visual Basic

```
AI_ReadChannel32 (ByVal CardNumber As Integer,
     ByVal Channel As Integer, ByVal AdRange As
     Integer, Value As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Channel	Analog input channel number. Valid value: **PCI-9524** P9524_AI_LC_CH0 P9524_AI_LC_CH1 P9524_AI_LC_CH2 P9524_AI_LC_CH3 P9524_AI_GP_CH0 P9524_AI_GP _CH1 P9524_AI_GP _CH2 P9524_AI_GP _CH3
AdRange	The analog input range setting of the specified channel. For PCI-9524, it should be the same as the

range setting of the AI_9524_PollConfig() function. Valid value:

PCI-9524:

Load cell channels

0

General purpose channels

AD_B_10_V

AD_B_5_V

AD_B_2_5_V

AD_B_1_25_V

Value The A/D converted value. The data format in value is described below:

PCI-9524

B31 ~ B8: 24-bit signed data

B7 ~ B4: Channel number

B3 ~ B2: Gain

B1: DSP flushed

B0: Data refreshed (polling mode only)

Return Code(s)

```
NoError
ErrorInvalidIoChannel
ErrorUndefinedParameter
ErrorConfigIoctl
```

AI_ReadMultiChannels

Description

This function performs software triggered A/D conversions on the specified analog input channels.

Supported card(s)

9221, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ReadMultiChannels (U16 CardNumber, U16
     numChans, U16 *Chans, U16 *AdRanges, U16
     *Buffer)
```

Visual Basic

```
AI_ReadMultiChannels (ByVal CardNumber As
     Integer, ByVal numChans As Integer, Chans As
     Integer, AdRanges As Integer, Buffer As
     Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

numChans The number of analog input channels in the array Chans. Valid values:

PCI-9221	`1 to 16`
PCI-9222	`1 to 16`
PCI-9223	`1 to 32`

Chans Array of analog input channel numbers. The channel order for acquiring data is the same as the order you set in Chans.

PCI-9221	Numbers in Chans must be within 0 and 15. Since there is no restriction for channel order setting, you may set the channel order as you want.
PCI-9222	Numbers in Chans must be within 0 and 15. Since there is no restriction for channel order setting, you may set the channel order as you want.

	PCI-9223	Numbers in Chans must be within 0 and 31. Since there is no restriction for channel order setting, you may set the channel order as you want.
AdRanges	An integer array of length numChans that contains the analog input range for every channel in array Chans. Refer to Appendix B: AI Range Codes for the valid range values.	
Buffer	An integer array to contain the acquired data. The length of Buffer must be equal to or greater than the value of parameter(s) ReadCount. The acquired data is stored in interleaved sequence. For example, if the value of numChans is 3, and the numbers in Chans are 3, 8, and 0, then this function input data from channel 3, then channel 8, then channel 0. The data acquired is put to Buffer by order, so the data read from channel 3 is stored in Buffer[0], the data read from channel 8 is stored in Buffer[1], and the data read from channel 0 is stored in Buffer[2].	

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

AI_ScanReadChannels

Description

This function performs software triggered A/D conversions on the specified analog input channels.

Supported card(s)

9221, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ScanReadChannels (U16 CardNumber, U16
     Channel, U16 AdRange, U16 *Buffer)
```

Visual Basic

```
AI_ScanReadChannels (ByVal CardNumber As Integer,
     ByVal Channel As Integer, ByVal AdRange As
     Integer, Buffer As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Channel	The largest channel number of specified continuous analog input channel. The channel order for acquiring data is:

PCI-9221	The number of channels must be within 0 and 15. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
PCI-9222	The number of channels must be within 0 and 15. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.
PCI-9223	The number of channels must be within 0 and 31. The continuous scan sequence is ascending and the first one must be zero. For example: 0, 1, 2, 3.

AdRange	The analog input range setting of specified channel. Refer to Appendix B: AI Range Codes.
Buffer	An integer array to contain the acquired data. The length of Buffer must be equal to or greater than the value of parameter(s) ReadCount. The acquired data is stored in interleaved sequence. For example, if the Channel value is 3 and the scanned channel

numbers is ascending, then this function input data from channel 0, then channel 1, then channel 2. The data acquired is put to Buffer by order, so the data read from channel 0 is stored in Buffer[0], the data read from channel 1 is stored in Buffer[1], and the data read from channel 2 is stored in Buffer[2].

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

AI_ScanReadChannels32

Description

This function performs software triggered A/D conversions on the specified analog input channels. For PCI-9524, the ADC will be started while the AI_9524_PollConfig function is performed. And then, you can use the function to get the analog values acquired with the set speed and range.

Supported card(s)

9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_ScanReadChannels32 (U16 CardNumber, U16
     Channel, U16 AdRange, U32 *Buffer)
```

Visual Basic

```
AI_ReadChannel32 (ByVal CardNumber As Integer,
     ByVal Channel As Integer, ByVal AdRange As
     Integer, Buffer As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Channel	The largest channel number of specified continuous analog input channel. For PCI-9524, the AI operation is individual for load cell or general purpose group. So the channel scan is also divided. For example, if the argument is set to P9524_AI_LC_CH3, the acquired data is from P9524_AI_LC_CH0 to P9524_AI_LC_CH3, and if the argument is set to P9524_AI_GP_CH1, the acquired data is from P9524_AI_GP_CH0 to P9524_AI_GP_CH1.
AdRange	The analog input range setting of the specified channels. For PCI-9524, it should be the same as the range setting of the AI_9524_PollConfig() function. Valid value:

PCI-9524:

Load cell channels

0

General purpose channels

AD_B_10_V

AD_B_5_V

AD_B_2_5_V

AD_B_1_25_V

Buffer	An long integer array to contain the acquired data. The length of Buffer must be equal to or greater than the value of ReadCount. The acquired data is stored in interleaved sequence. For example, if the channel value is 3 and the scanned channel numbers is ascending, then this function input data from channel 0, then channel 1, then channel 2. The data acquired is put to Buffer by order, so the data read from channel 0 is stored in Buffer[0], the data read from channel 1 is stored in Buffer[1], and the data read from channel 2 is stored in Buffer[2].

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorInvalidIoChannel
ErrorFuncNotSupport
ErrorInvalidAdRange
ErrorPIOIoctl
```

AI_SetTimeOut

Description

Sets Timeout period for Sync. mode continuous AI. While the function is called, the Sync. mode continuous AI acquisition is stopped even when it is not completed.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_SetTimeOut (U16 CardNumber, U32 TimeOut)
```

Visual Basic

```
AI_SetTimeOut (ByVal CardNumber As Integer, ByVal
     TimeOut As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
TimeOut	Timeout period (ms).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AI_VReadChannel

Description

Performs a software triggered A/D conversion (analog input) on an analog input channel and returns the scaled value into voltage in units of volts.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_VReadChannel (U16 CardNumber, U16 Channel,
     U16 AdRange, F64 *voltage)
```

Visual Basic

```
AI_ReadChannel (ByVal CardNumber As Integer,
     ByVal Channel As Integer, ByVal AdRange As
     Integer, voltage As Double) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Channel Analog input channel number. Range:

PCI/cPCI-9112, PCI-9111, PCI-9118	`0 to 15`
PCI-9113, PCI-9114	`0 to 31`
cPCI-9116	`0 to 63`
PCI-9221	`0 to 15`
PCI-9222	`0 to 15`
PCI-9223	`0 to 31`

PCI-9524	`P9524_AI_LC_CH0`
	`P9524_AI_LC_CH1`
	`P9524_AI_LC_CH2`
	`P9524_AI_GP_CH3`
	`P9524_AI_GP_CH1`
	`P9524_AI_GP_CH2`
	`P9524_AI_GP_CH3`

AdRange	The analog input range setting of the specified channel. Refer to Appendix B: AI Range Codes for the valid range values.
voltage	The measured voltage value returned and scaled to units of voltage.

Return Code(s)

```
NoError, ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorInvalidAdRange
```

AI_VoltScale

Description

Converts the result from an AI_ReadChannel call to the actual input voltage.

Supported card(s)

9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_VoltScale (U16 CardNumber, U16 AdRange,
     I16 reading, F64 *voltage)
```

Visual Basic

```
AI_VoltScale (ByVal CardNumber As Integer, ByVal
     AdRange As Integer, ByVal reading As
     Integer, voltage As Double) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
AdRange	The analog input range setting of the specified channel. Refer to Appendix B: AI Range Codes for the range of valid values.
reading	Result of the AD conversion.
voltage	Computed voltage value.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidAdRange
```

AI_VoltScale32

Description

Converts the result from an AI_ReadChannel32 call to the actual input voltage.

Supported card(s)

9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AI_VoltScale32 (U16 CardNumber, U16 AdRange,
     I32 reading, F64 *voltage)
```

Visual Basic

```
AI_VoltScale32 (ByVal CardNumber As Integer,
     ByVal AdRange As Integer, ByVal reading As
     Long, voltage As Double) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
AdRange	The analog input range setting of the specified channel. Valid value: PCI-9524 Data acquired from Load cell channel 0 Data acquired from General purpose channel AD_B_10_V AD_B_5_V AD_B_2_5_V AD_B_1_25_V
reading	Result of AD conversion.
voltage	Computed voltage value.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidAdRange
```

AO_6202_Config

Description

Informs the PCIS-DASK library of the selected trigger source for the device CardNumber ID. After calling the Register_Card function, the device is configured to the following by default:

D/A R/W source	P6202_DA_WRSRC_Int
D/A trigger mode	P6202_DA_TRGMOD_POST
D/A trigger source	P6202_DA_TRGSRC_SOFT
Auto reset buffer Enabled	(AutoResetBuf: TRUE)

If you want to use the device with the default settings, it is not necessary to call this function to make the configuration again. Otherwise, this function has to be called before calling function to perform continuous analog output operation.

Supported Cards

6202

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 AO_6202_Config (U16 CardNumber, U16
     ConfigCtrl, U16 TrigCtrl, U16 ReTrgCnt, U32
     DLY1Cnt, U32 DLY2Cnt, BOOLEAN AutoResetBuf)
```

Visual Basic

```
AO_6202_Config (ByVal CardNumber As Integer,
     ByVal ConfigCtrl As Integer, ByVal TrigCtrl
     As Integer, ByVal ReTrgCnt As Integer, ByVal
     DLY1Cnt As Long, ByVal DLY2Cnt As Long,
     ByVal AutoResetBuf As Byte) As Integer
```

Parameters

CardNumber ID of the card performing the operation.

ConfigCtrl D/A configuration control setting. This argument is an integer expression formed from one of the manifest constants defined in DASK.H. There are a group of constants:

D/A R/W Source Selection

`P6202_DA_WRSRC_Int` Internal timer (Default)

P6202_DA_WRSRC_AFI0 From AFI0 pin

P6202_DA_WRSRC_SSI From SSI source

P6202_DA_WRSRC_AFI1 From AFI1 pin

TrigCtrl The setting for D/A Trigger control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are four groups of constants:

Trigger Source Selection

P6202_DA_TRGSRC_SOFT Software (Default)

P6202_DA_TRGSRC_AFI0 From AFI0 pin

P6202_DA_TRSRC_SSI From SSI source

P6202_DA_TRGSRC_AFI1 From external AFI1 pin

Trigger Mode Selection

P6202_DA_TRGMOD_POST Post Trigger Mode (Default)

P6202_DA_TRGMOD_DELAY Delay Trigger Mode

Re-Trigger Mode Enable (available only for post and delay trigger modes)

P6202_DA_ReTrigEn Re-trigger in an acquisition is enabled.

Delay2 (Break delay) Mode Enable

P6202_DA_DLY2En Delay2/Break delay (delay between two consecutive waveform generations) in an acquisition is enabled.

When two or more constants are used to form the TrigCtrl argument, the constants are combined with the bitwise-OR operator(|).

ReTrgCnt The accepted trigger times in an acquisition. If the value of ReTrgCnt is 0, the fixed pattern generation is triggered infinitely. This argument is valid only for delay trigger and post trigger modes. The range of valid value is 0 to 4294967295.

NOTE: To enable infinite re-trigger mode of fixed pattern generation, call **`AO_6202_Config`** with **`P6202_DA_ReTrigEn`** and assign a zero value to **`ReTrgCnt`**.

DLY1Cnt	DLY1 counter value or the delay time to start waveform generation after the trigger signal. This argument is valid only for delay trigger mode. The range of valid value is 0 to 4294967295.
DLY2Cnt	DLY2 counter value or the delay between two consecutive waveform generations. The range of valid value is 0 to 4294967295.
AutoResetBuf:	FALSE: The DA buffer set by function “AO_ContBufferSetup” are retained and must call function “AO_ContBufferReset” to reset the buffer TRUE: The DA buffer set by function “AO_ContBufferSetup” are reset automatically by driver while the AI operation is finished

Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AO_6308A_Config

Description

Sets the voltage to the current mode of PCI-6308A.

Supported card(s)

6308A

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_6308A_Config (U16 CardNumber, U16 V2AMode)
```

Visual Basic

```
AO_6308A_Config (ByVal CardNumber As Integer,
     ByVal V2AMode As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.

V2AMode	The voltage to current mode. Valid V2Amode values:
	`P6308_CURRENT_0_20MA`
	`P6308_CURRENT_5_25MA`
	`P6308_CURRENT_4_20MA`

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

AO_6308V_Config

Description

Informs the PCIS-DASK library of the output channel polarity (unipolar or bipolar) for analog output and the reference voltage value selected for an analog output channel. You can configure each channel to use an internal reference of 10 V or an external reference (0 V~ +10 V) by setting the related jumpers. You must call this function before calling function to perform voltage output operation.

Supported card(s)

6308V

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_6308V_Config (U16 CardNumber, U16 Channel,
     U16 OutputPolarity, F64 refVoltage)
```

Visual Basic

```
AO_6308V_Config (ByVal CardNumber As Integer,
     ByVal Channel As Integer, ByVal
     OutputPolarity As Integer, ByVal refVoltage
     As Double) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.

Channel	Configures AO channel number. Valid values:
	`P6308V_AO_CH0_3`
	`P6308V_AO_CH4_7`

OutputPolarity	Polarity (unipolar or bipolar) of the output channel. Valid values:
	`P6308V_AO_UNIPOLAR`
	`P6308V_AO_BIPOLAR`

refVoltage	Voltage reference value. When the D/A reference voltage is set to internal reference, the valid value for refVoltage is 10. When the D/A reference voltage is set to external reference, the valid range for refVoltage is 0 to +10.

NOTE:

When the 10 V D/A reference voltage is selected, the D/A output range is 0 V to 10 V.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidDaRefVoltage
```

AO_9111_Config

Description

Informs the PCIS-DASK library of the output channel polarity (unipolar or bipolar) for analog output. You must call this function before calling function to perform voltage output operation.

Supported card(s)

9111

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_9111_Config (U16 CardNumber, U16
     OutputPolarity)
```

Visual Basic

```
AO_9111_Config (ByVal CardNumber As Integer,
     ByVal OutputPolarity As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.

OutputPolarity	Polarity (unipolar or bipolar) of the output channel. Valid values:
	`P9111_AO_UNIPOLAR`
	`P9111_AO_BIPOLAR`

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AO_9112_Config

Description

Informs the PCIS-DASK library of the selected reference voltage value for an analog output channel. Each channel may be configured to use an internal reference of -5 V (default) or -10 V, or an external reference of -10 V to +10 V by setting the related jumpers. You must call this function before calling function to perform a voltage output operation.

Supported card(s)

9112

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_9112_Config (U16 CardNumber, U16 Channel,
     F64 refVoltage)
```

Visual Basic

```
AO_9112_Config (ByVal CardNumber As Integer,
     ByVal Channel As Integer, ByVal refVoltage
     As Double) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Channel	Configured AO channel number.
refVoltage	Voltage reference value. If the D/A reference voltage source your device use is internal reference, the valid values for refVoltage is -5 and -10. If the D/A reference voltage source your device use is external reference, the valid range for refVoltage is -10 to +10.

NOTE: When the -10V D/A reference voltage is selected, the D/A output range is 0V~10V. On the other hand, if the +10V is selected, the D/A output range is -10 V to 0 V.

Return Code(s)

```
NoError
ErrorInvalidCardNumber, ErrorCardNotRegistered
ErrorFuncNotSupport, ErrorInvalidDaRefVoltage
```

AO_9222_Config

Description

Inform the PCIS-DASK library of the trigger source, trigger mode, and trigger properties selected for the PCI-9222 with card ID CardNumber. You must call this function before calling function to perform continuous analog output operation.

Supported card(s)

9222

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_9222_Config (U16 CardNumber, U16
     ConfigCtrl, U16 TrigCtrl, U32 ReTrgCnt, U32
     DLY1Cnt, U32 DLY2Cnt, BOOLEAN AutoResetBuf)
```

Visual Basic

```
AO_9222_Config (ByVal CardNumber As Integer,
     ByVal ConfigCtrl As Integer, ByVal TrigCtrl
     As Integer, ByVal ReTrgCnt As Long, ByVal
     DLY1Cnt As Long, ByVal DLY2Cnt As Long,
     ByVal AutoResetBuf As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ConfigCtrl	The setting for D/A mode control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There is one group of constants:

Conversion Source Selection

```
P922x_AO_CONVSRC_INT
P922x_AO_CONVSRC_GPI0
P922x_AO_CONVSRC_GPI1
P922x_AO_CONVSRC_GPI2
P922x_AO_CONVSRC_GPI3
P922x_AO_CONVSRC_GPI4
P922x_AO_CONVSRC_GPI5
P922x_AO_CONVSRC_GPI6
P922x_AO_CONVSRC_GPI7
```

	`P922x_AO_CONVSRC_SSI2`
TrigCtrl	The setting for D/A Trigger control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are five groups of constants:

Trigger Mode Selection

`P922x_AO_TRGMOD_POST`

`P922x_AO_TRGMOD_DELAY`

Trigger Source Selection

`P922x_AO_TRGSRC_SOFT`

`P922x_AO_TRGSRC_GPI0`

`P922x_AO_TRGSRC_GPI1`

`P922x_AO_TRGSRC_GPI2`

`P922x_AO_TRGSRC_GPI3`

`P922x_AO_TRGSRC_GPI4`

`P922x_AO_TRGSRC_GPI5`

`P922x_AO_TRGSRC_GPI6`

`P922x_AO_TRGSRC_GPI7`

`P922x_AO_TRGSRC_SSI6`

Trigger Polarity

`P922x_AO_TrgPositive`

`P922x_AO_TrgNegative`

Re-Trigger Mode Enable

`P922x_AO_EnReTigger`

Waveform Separation Delay Enable (Delay 2)

`P922x_AO_EnDelay2`

ReTrgCnt	The accepted trigger times in an acquisition. This argument is valid only for re-trigger mode. The valid range of ReTrgCnt is 0 to 4294967295. If the value of ReTrgCnt is 0, the AO operation is triggered infinitely.
DLY1Cnt	DLY1 counter value or the delay time to start waveform generation after the trigger signal. This argument is valid only for delay trigger mode. The range of valid value is 0 to 4294967295.
DLY2Cnt	DLY2 counter value or the delay between two consecutive waveform generations. This argument is valid only for waveform repeat. The range of valid value is 0 to 4294967295.

AutoResetBuf

FALSE

The AO buffers set by the AO_ContBufferSetup function are retained. You must call the AO_ContBufferReset function to reset the buffer.

TRUE

The AO buffers set by the AO_ContBufferSetup function are reset automatically by driver when the AO operation is completed.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorConfigIoctl
```

AO_9223_Config

Description

Inform the PCIS-DASK library of the trigger source, trigger mode, and trigger properties selected for the PCI-9223 with card ID CardNumber. You must call this function before calling function to perform continuous analog output operation.

Supported card(s)

9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_9223_Config (U16 CardNumber, U16
     ConfigCtrl, U16 TrigCtrl, U32 ReTrgCnt, U32
     DLY1Cnt, U32 DLY2Cnt, BOOLEAN AutoResetBuf)
```

Visual Basic

```
AO_9223_Config (ByVal CardNumber As Integer,
     ByVal ConfigCtrl As Integer, ByVal TrigCtrl
     As Integer, ByVal ReTrgCnt As Long, ByVal
     DLY1Cnt As Long, ByVal DLY2Cnt As Long,
     ByVal AutoResetBuf As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ConfigCtrl	The setting for D/A mode control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There is one group of constants:

Conversion Source Selection

```
P922x_AO_CONVSRC_INT
P922x_AO_CONVSRC_GPI0
P922x_AO_CONVSRC_GPI1
P922x_AO_CONVSRC_GPI2
P922x_AO_CONVSRC_GPI3
P922x_AO_CONVSRC_GPI4
P922x_AO_CONVSRC_GPI5
P922x_AO_CONVSRC_GPI6
P922x_AO_CONVSRC_GPI7
```

	`P922x_AO_CONVSRC_SSI2`

TrigCtrl	The setting for D/A Trigger control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are five groups of constants:
	Trigger Mode Selection
	`P922x_AO_TRGMOD_POST`
	`P922x_AO_TRGMOD_DELAY`
	Trigger Source Selection
	`P922x_AO_TRGSRC_SOFT`
	`P922x_AO_TRGSRC_GPI0`
	`P922x_AO_TRGSRC_GPI1`
	`P922x_AO_TRGSRC_GPI2`
	`P922x_AO_TRGSRC_GPI3`
	`P922x_AO_TRGSRC_GPI4`
	`P922x_AO_TRGSRC_GPI5`
	`P922x_AO_TRGSRC_GPI6`
	`P922x_AO_TRGSRC_GPI7`
	`P922x_AO_TRGSRC_SSI6`
	Trigger Polarity
	`P922x_AO_TrgPositive`
	`P922x_AO_TrgNegative`
	Re-Trigger Mode Enable
	`P922x_AO_EnReTigger`
	Waveform Separation Delay Enable (Delay 2)
	`P922x_AO_EnDelay2`

ReTrgCnt	The accepted trigger times in an acquisition. This argument is valid only for re-trigger mode. The valid range of ReTrgCnt is 0 to 4294967295. If the value of ReTrgCnt is 0, the AO operation is triggered infinitely.
DLY1Cnt	DLY1 counter value or the delay time to start waveform generation after the trigger signal. This argument is valid only for delay trigger mode. The range of valid value is 0 to 4294967295.
DLY2Cnt	DLY2 counter value or the delay between two consecutive waveform generations. This argument is valid only for waveform repeat. The range of valid value is 0 to 4294967295.

AutoResetBuf

FALSE

The AO buffers set by the AO_ContBufferSetup function are retained. You must call the AO_ContBufferReset function to reset the buffer.

TRUE

The AO buffers set by the AO_ContBufferSetup function are reset automatically by driver when the AO operation is completed.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorConfigIoctl
```

AO_AsyncCheck

Description

Check the current status of the asynchronous analog output operation.

Supported card(s)

6202, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_AsyncCheck (U16 CardNumber, BOOLEAN
     *Stopped, U32 WriteCnt)
```

Visual Basic

```
AO_AsyncCheck (ByVal CardNumber As Integer,
     Stopped As Byte, WriteCnt As Long) As
     Integer
```

Parameter(s)

CardNumber	The card id of the card that performs the asynchronous operation.
Stopped	Whether the asynchronous analog output operation has completed. If Stopped = TRUE, the analog output operation has stopped. Either the number of D/A conversions indicated in the call that initiated the asynchronous analog output operation has completed or an error has occurred. If Stopped = FALSE, the operation is not yet complete. (constants TRUE and FALSE are defined in DASK.H)
WriteCnt	The number of analog output data that have been written at the time calling AO_AsyncCheck().

Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AO_AsyncClear

Description

Stop the asynchronous analog output operation.

Supported card(s)

6202, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_AsyncClear (U16 CardNumber, U32
     *UpdateCnt, U16 stop_mode)
```

Visual Basic

```
AO_AsyncClear (ByVal CardNumber As Integer,
     UpdateCnt As Long, stop_mode As Integer) As
     Integer
```

Parameter(s)

CardNumber	The card id of the card that performs the asynchronous operation.
WriteCnt	The number of analog output data that have been written at the time calling AO_AsyncClear().
stop_mode	The DA transfer termination mode selected. The valid value is: DA_TerminateImmediate: Software terminate the DA continuous operation immediately

Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AO_AsyncDblBufferHalfReady

Description

Checks whether the next half buffer is ready for new data during an asynchronous double-buffered analog output operation.

Supported card(s)

6202, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_AsyncDblBufferHalfReady (U16 CardNumber,
     BOOLEAN *HalfReady)
```

Visual Basic

```
AO_AsyncDblBufferHalfReady(ByVal CardNumber As
     Integer, HalfReady As Byte) As Integer
```

Parameter(s)

CardNumber	The card id of the card that performs the asynchronous double-buffered operation.
HalfReady	Whether the next half buffer is ready for new data.

Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AO_AsyncDblBufferMode

Description

Enables or disables double-buffered data acquisition mode.

Supported Cards

6202, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_AsyncDblBufferMode (U16 CardNumber,
     BOOLEAN Enable)
```

Visual Basic

```
AO_AsyncDblBufferMode (ByVal CardNumber As
     Integer, ByVal Enable As Byte) As Integer
```

Parameter

CardNumber	The card id of the card that double-buffered mode to be set.
Enable	Whether the double-buffered mode is enabled or not. *TRUE*: double-buffered mode is enabled. *FALSE*: double-buffered mode is disabled. Note: constants TRUE and FALSE are defined in DASK.H.

Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AO_ContBufferCompose

Description

Fills the data for a specified channel in the buffer for multi-channels of continuous analog output operation.

Supported card(s)

6202, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_ContBufferCompose (U16 CardNumber, U16
     TotalChnCount, U16 ChnNum, U32 UpdateCount,
     VOID *ConBuffer, VOID *Buffer)
```

Visual Basic

```
AO_ContBufferCompose (ByVal CardNumber As
     Integer, ByVal TotalChnCount As Integer,
     ByVal ChnNum As Integer, ByVal UpdateCount
     As Long, ConBuffer As Any, Buffer As Any) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
TotalChnCount	Numbers of AO channels to be performed. **PCI-6202** 1 to 4 **PCI-9222** and **PCI-9223** 1 or 2
ChnNum	Specified AO channel number. **PCI-6202** 0 to 3 **PCI-9222** and **PCI-9223** 0 or 1
UpdateCount	Size (in samples) of the specified channel buffer. This is not the size of the buffer for continuous output operation.

ConBuffer	Buffer for multi-channels of continuous output operation.
Buffer	Buffer containing the output data for the specified channel.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorUndefinedParameter
```

AO_ContBufferReset

Description

This function reset all the buffers set by function AO_ContBufferSetup for continuous analog output. The function has to be called if the data buffers won't be used.

Supported card(s)

6202, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_ContBufferReset (U16 CardNumber)
```

Visual Basic

```
AO_ContBufferReset (ByVal CardNumber As Integer)
     As Integer
```

Parameter

CardNumber	The card id of the card that want to perform this operation.

Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorContIoNotAllowed
```

AO_ContBufferSetup

Description

This function set up the buffer for continuous analog output operation. The function has to be called repeatedly to setup all of the data buffers. The maximum number of buffers is two.

Supported Cards

6202, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_ContBufferSetup (U16 CardNumber, void
     *Buffer, U32 WriteCount, U16 *BufferId)
```

Visual Basic

```
AO_ContBufferSetup (ByVal CardNumber As Integer,
     Buffer As Any, ByVal WriteCount As Long,
     BufferId As Integer) As Integer
```

Parameter

CardNumber	The card id of the card that want to perform this operation.
Buffer	The starting address of the memory to contain the output data.
WriteCount	The size (in samples) of the buffer and its value must be even.
BufferId	Returns the index of the buffer currently set up.

Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
```

AO_ContStatus

Description

While performing continuous D/A conversions, this function is called to get the D/A status. Please refer to the manual for your device for the AO status the device might meet.

Supported card(s)

6202, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_ContStatus (U16 CardNumber, U16 *Status)
```

Visual Basic

```
AO_ContStatus (ByVal CardNumber As Integer,
     Status As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Status	The continuous AO status returned. The description of the Status parameter for various card types is listed.

PCI-6202, PCI-9222, and PCI-9223

`bit 0`	1 = FIFO is empty.
`bit 1`	1 = FIFO is almost empty.
`bit 2`	1 = FIFO is almost full.
`bit 3`	1 = FIFO is full.
`bit 4-7`	Not used.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorContStatusIoctl
```

AO_ContWriteChannel

Description

This function performs continuous D/A conversions on the specified analog output channel at a rate as close to the rate you specified.

Supported card(s)

6202, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_ContWriteChannel (U16 CardNumber, U16
     Channel, U16 BufId, U32 UpdateCount, U32
     Iterations, U32 CHUI, U16 definite, U16
     SyncMode)
```

Visual Basic

```
AO_ContWriteChannel (ByVal CardNumber As Integer,
     ByVal Channel As Integer, ByVal BufId As
     Integer, ByVal UpdateCount As Long, ByVal
     Iterations As Long, ByVal CHUI As Long,
     ByVal definite As integer, ByVal SyncMode As
     Integer) As Integer
```

Parameter

CardNumber The card id of the card that want to perform this operation.

Channel Analog output channel number

PCI-6202

0 to 3

PCI-9222 and **PCI-9223**

0 to 1

BufId The buffer ID (returned from function AO_ContBufferSetup) of the buffer containing the acquired data. The size of the buffer with buffer id of BufId must have a length (in samples) equal to the value of parameter UpdateCount.

UpdateCount If double-buffered mode is disabled, the total update count for each channel to be performed. For double-

	buffered acquisition, UpdateCount is the size (in samples) allocated for each channel in the circular buffer and its value must be a multiple of 2.
Iterations	The times of number of the data in the buffer to output to the port.A value of zero is not allowed. If the DA operation is perform synchronously, this argument must be set as 1.
CHUI	The length of the Channel Update interval (that is, the counter value between the initiation of each update sequence). Valid range of the value is as follows: **PCI-6202, PCI-9222,** and **PCI-9223** 80 to 4294967295
definite	Waveform generation proceeds definite or indefinitely. If double-buffered mode is enabled, this parameter is of no use. 0: indefinitely 1: definite
SyncMode	Whether this operation is performed synchronously or asynchronously. If any trigger mode is enabled, this operation should be performed asynchronously. Valid values: ASYNCH_OP: Asynchronous D/A conversion SYNCH_OP: Synchronous D/A conversion, that is, the function does not return until the analog output operation is completed.

Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorInvalidAdRange
ErrorTransferCountTooLarge
ErrorContIoNotAllowed,
ErrorInvalidSampleRate
```

AO_ContWriteMultiChannels

Description

This function performs continuous D/A conversions on the specified analog output channels at a rate as close to the rate you specified.

Supported card(s)

6202, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_ContWriteMultiChannels (U16 CardNumber,
     U16 NumChans, U16 *Chans, U16 BufId, U32
     UpdateCount, U32 Iterations, U32 CHUI, U16
     definite, U16 SyncMode)
```

Visual Basic

```
AO_ContReadMultiChannels (ByVal CardNumber As
     Integer, ByVal NumChans As Integer, chans As
     Integer, ByVal BufId As Integer, ByVal
     UpdateCount As Long, ByVal Iterations As
     Long, ByVal CHUI As Long, ByVal definite As
     integer, ByVal SyncMode As Integer) As
     Integer
```

Parameter

CardNumber	The card ID of the card that want to perform this operation.
numChans	The number of analog input channels in the array Chans. The valid values: **PCI-6202** 1 to 4 **PCI-9222** and **PCI-9223** 1 or 2

Chans	Array of analog output channel numbers. The channel order for update data is the same as the order you set in Chans. PCI-6202 Numbers in Chans must be within 0 to 3 PCI-9222 and PCI-9223 Numbers in Chans must be 0 or 1.
BufId	The buffer ID (returned from function AO_ContBufferSetup) of the buffer containing the output data. The size of the buffer with buffer id of BufId must have a length equal to or greater than the value of WriteCount X numChans. The data order in the buffer is in interleaved sequence as follows. So the data for channel 0 is stored in Buffer[0], Buffer[2], Buffer[4], ... The data for channel 1 is stored in Buffer[1], Buffer[3], Buffer[5], ...
UpdateCount	If double-buffered mode is disabled, the total update count for each channel to be performed. For double-buffered acquisition, UpdateCount is the size (in samples) allocated for each channel in the circular buffer and its value must be a multiple of 2.
Iterations	The times of number of the data in the buffer to output to the port. If the argument is set to 0, the DA operation will repeat infinitely. If the DA operation is performed synchronously, this argument must be set as 1.
CHUI	The length of the Channel Update interval (that is, the counter value between the initiation of each update sequence). **PCI-6202, PCI-9222,** and **PCI-9223** 80 to 4294967295
definite	Waveform generation proceeds definite or indefinitely. If double-buffered mode is enabled, this parameter is of no use. 0: indefinitely 1: definite

SyncMode	Whether this operation is performed synchronously or asynchronously. If any trigger mode is enabled, this operation should be performed asynchronously..

Valid values:

ASYNCH_OP: Asynchronous D/A conversion

SYNCH_OP: Synchronous D/A conversion, that is, the function does not return until the analog output operation is completed.

Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorInvalidSampleRate
ErrorInvalidAdRange
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
```

AO_EventCallBack (Win32 Only)

Description

Controls and notifies the user's application when a specified DAQ event occurs. The notification is performed through a user-specified callback function. For Linux version, the event message has to be manually removed by set the parameter "mode" to be 0. For the windows version, the event message will be removed automatically after calling AO_Async_Clear. The event message can also be manually removed by set the parameter "mode" to be 0.

Supported card(s)

6202, 9222, 9223

Syntax

Microsoft C/C++ and Borland C++

```
I16 AO_EventCallBack (U16 CardNumber, I16 mode,
     I16 EventType, U32 callbackAddr)
```

Linux C++

```
I16 AO_EventCallBack (U16 CardNumber, I16 mode,
     I16 EventType, void (*callbackAddr)(int))
```

Visual Basic

```
AO_EventCallBack (ByVal CardNumber As Integer,
     ByVal mode As Integer, ByVal EventType As
     Integer, ByVal callbackAddr As Long) As
     Integer
```

Parameter

CardNumber	The card id of the card that want to be performed this operation.
mode	Add or remove the event message. The valid values: 0: remove 1: add
EventType	Event criteria. The valid values are: AOEnd: Notification for the completeness of asynchronous analog output operation.

	DBEvent: Notification for the next half buffer of data in circular buffer is ready for transfer.
	TrigEvent: Notifies that the data associated to the next trigger signal is available. (only for PCI-9222/9223)
callbackAddr	The address of the user callback function. PCIS-DASK calls this function when the specified event occurs. If you wish to remove the event message, set callbackAddr to 0.

Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AO_InitialMemoryAllocated

Description

Returns the available memory size for analog output in the device driver using the argument MemSize. The continuous analog output transfer size may not exceed this size.

Supported card(s)

6202, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_InitialMemoryAllocated (U16 CardNumber,
     U32 *MemSize)
```

Visual Basic

```
AO_InitialMemoryAllocated (ByVal CardNumber As
     Integer, MemSize As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
MemSize	The available memory size for continuous AO in the card's device driver. The unit is KB (1024 bytes)

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AO_SetTimeOut

Description

Sets Timeout period for Sync. mode continuous AO. While the function is called, the Sync. mode continuous AO acquisition is stopped even when it is not completed.

Supported card(s)

6202, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 AO_SetTimeOut (U16 CardNumber, U32 TimeOut)
```

Visual Basic

```
AO_SetTimeOut (ByVal CardNumber As Integer, ByVal
     TimeOut As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
TimeOut	Timeout period (ms).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

AO_SimuVWriteChannel

Description

Simultaneously writes the voltage values, scales them to the proper binary values, and writes binary values to the specified analog output channels.

Supported card(s)

6308V/08A, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 AO_SimuVWriteChannel (U16 CardNumber, U16
     Group, F64 *VBuffer)
```

Visual Basic

```
AO_SimuVWriteChannel (ByVal CardNumber As
     Integer, ByVal Group As Integer, VBuffer As
     Double) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.

Group	Group number of the analog output channels. Valid values:
	PCI-6308V/08A
	`P6308V_AO_CH0_3`
	`P6308V_AO_CH4_7`
	cPCI-9524
	`p9524_AO_CH0_1`

VBuffer	A voltage array to contain the update data. The length (in samples) of VBuffer must be equal to or greater than the number of channels in the specified group. The range of voltage depends on the type of device, output polarity, and voltage reference (external or internal).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
```

```
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

AO_SimuWriteChannel

Description

Simultaneously writes binary values to the specified analog output channels.

Supported card(s)

6308V/08A, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 AO_SimuWriteChannel (U16 CardNumber, U16
     Group, I16 *Buffer)
```

Visual Basic

```
AO_SimuWriteChannel (ByVal CardNumber As Integer,
     ByVal Group As Integer, Buffer As Integer)
     As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Group	Group number of the analog output channels. Valid values:

PCI-6308V/08A

```
P6308V_AO_CH0_3
P6308V_AO_CH4_7
```

PCI-9524

```
p9524_AO_CH0_1
```

Buffer	An integer array to contain the update data. The length (in samples) of Buffer must be equal to or greater than the number of channels in the specified group. Range: The range of value to be written to the analog output channels is:

PCI-6308

0 to 4095

PCI-9524

0 to 65535

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

AO_VoltScale

Description

Scales a voltage (or a current value) to a binary value.

Supported card(s)

6202, 6208V/16V/08A, 6308V/08A, 9111, 9112, 9118, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 AO_VoltScale (U16 CardNumber, U16 Channel,
     F64 Voltage, I16 *binValue)
```

Visual Basic

```
AO_VoltScale (ByVal CardNumber As Integer, ByVal
     Channel As Integer, ByVal Voltage As Double,
     binValue As Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Channel Analog output channel number. Range:

PCI-6202	`0 to 3`
PCI-6208V/08A and PCI-6308V/08A	`0 to 7`
PCI-6216V	`0 to 15`
PCI-9111	`0`
PCI-9112/cPCI-9112	`0 or 1`
PCI-9118	`0 or 1`
PCI-9221	`0 or 1`
PCI-9222	`0 or 1`
PCI-9223	`0 or 1`
PCI-9524	`0 or 1`

Voltage Voltage, in volts, to be converted to a binary value.

binValue Converted binary value returned.

Return Code(s)

```
NoError
```

```
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorDaVoltageOutOfRange
```

AO_VWriteChannel

Description

Accepts a voltage value (or a current value), scales it to the proper binary value, and writes a binary value to the specified analog output channel.

Supported card(s)

6202, 6208V/16V/16A, 6308V/08A, 9111, 9112, 9118, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 AO_VWriteChannel (U16 CardNumber, U16
     Channel, F64 Voltage)
```

Visual Basic

```
AO_VWriteChannel (ByVal CardNumber As Integer,
     ByVal Channel As Integer, ByVal Voltage As
     Double) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Channel Analog output channel number. Range:

PCI-6202	`0 to 4`
PCI-6208V/08A and PCI-6308V/08A	`0 to 7`
PCI-6216V	`0 to 15`
PCI-9111	`0`
PCI-9112/cPCI-9112	`0 or 1`
PCI-9118	`0 or 1`
PCI-9221	`0 or 1`
PCI-9222	`0 or 1`
PCI-9223	`0 or 1`
PCI-9524	`0 or 1`

Voltage The value to be scaled and written to the analog output channel. The range of voltages depends on the type of device, output polarity, and voltage reference (external or internal).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered, ErrorFuncNotSupport
ErrorInvalidIoChannel, ErrorDaVoltageOutOfRange
```

AO_WriteChannel

Description

Writes a binary value to the specified analog output channel.

Supported card(s)

6202, 6208V/16V/16A, 6308V/08A, 9111, 9112, 9118, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 AO_WriteChannel (U16 CardNumber, U16 Channel,
     U16 Value)
```

Visual Basic

```
AO_WriteChannel (ByVal CardNumber As Integer,
     ByVal Channel As Integer, ByVal Value As
     Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Channel Analog output channel number. Range:

PCI-6202	`0 to 4`
PCI-6208V/08A and PCI-6308V/08A	`0 to 7`
PCI-6216V	`0 to 15`
PCI-9111	`0`
PCI-9112/cPCI-9112	`0 or 1`
PCI-9118	`0 or 1`
PCI-9221	`0 or 1`
PCI-9222	`0 or 1`
PCI-9223	`0 or 1`
PCI-9524	`0 or 1`

Value Value to be written to the analog output channel. Range:

PCI-6202	`0 to 65535`
PCI-6208A and PCI-6308A	`0 to 32767`
PCI-6208V/16V and PCI-6308V	`-32767 to 32767`
PCI-9111/PCI-9112/cPCI-9112	`0 to 4095`
PCI-9221	`-32768 to 32767`
PCI-9222	`0 to 65535`
PCI-9223	`0 to 65535`
PCI-9524	`0 to 65535`

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport, ErrorInvalidIoChannel
```

CTR_8554_CK1_Config

Description

Selects the source of CK1.

Supported card(s)

8554

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 CTR_8554_CK1_Config (U16 CardNumber, U16
     ClockSource)
```

Visual Basic

```
CTR_8554_CK1_Config (ByVal CardNumber As Integer,
     ByVal ClockSource As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ClockSource	The source of CK1: CK1_C8M or CK1_COUT11.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
InvalidCtrSource
```

CTR_8554_ClkSrc_Config

Description

Selects the PCI-8554 counter #1 to #10 clock source. Clock source of counter #11 is fixed at 8 MHz, while the clock source of counter #12 is from COUT11.

Supported card(s)

8554

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 CTR_8554_ClkSrc_Config (U16 CardNumber, U16
     Ctr, U16 ClockSource)
```

Visual Basic

```
CTR_8554_ClkSrc_Config (ByVal CardNumber As
     Integer, ByVal Ctr As Integer, ByVal
     ClockSource As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Ctr	Counter number. Range is 1 to 10.
ClockSource	Clock source of the specified counter.

`ECKN`	External clock source.
`COUTN_1`	Cascaded counter output (COUT n-1).
`CK1`	Internal clock source CK1.
`COUT10`	Output of counter 10.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
InvalidCounter
```

CTR_8554_Debounce_Config

Description

Selects the debounce clock.

Supported card(s)

8554

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 CTR_8554_Debounce_Config (U16 CardNumber, U16
     DebounceClock)
```

Visual Basic

```
CTR_8554_CK1_Config (ByVal CardNumber As Integer,
     ByVal DebounceClock As Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

DebounceClock

`DBCLK_COUT11`	Output of counter 11
`DBCLK_2MHZ`	2 MHz

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
InvalidCtrSource
```

CTR_Clear

Description

Turns off the specified counter operation and sets the output of the selected counter to the specified state.

Supported card(s)

9111, 9112, 9113, 9114, 9118, 7224, 7248, 7249, 7296, 7348, 7396, 8554

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 CTR_Clear (U16 CardNumber, U16 Ctr, U16
     State)
```

Visual Basic

```
CTR_Clear (ByVal CardNumber As Integer, ByVal Ctr
     As Integer, ByVal State As Integer) As
     Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Ctr Counter number. Range:

PCI-9111, PCI-9112/cPCI-9112, PCI-9113, PCI-9114, PCI-9118	`0`
PCI-7248/cPCI-7248/PCI-7224, cPCI-7249R, PCI-7296, PCI-7348/PCI-7396	`0, 1, 2`
PCI-8554	`1 to 12`

State Logic state to which the counter is to be reset. Range is 0, 1.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidCounter
```

CTR_Read

Description

Reads the current contents of the selected counter without disturbing the counting process.

Supported card(s)

9111, 9112, 9113, 9114, 9118, 7224, 7248, 7249, 7296, 7348, 7396, 8554

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 CTR_Read (U16 CardNumber, U16 Ctr, U32
     *Value)
```

Visual Basic

```
CTR_Read (ByVal CardNumber As Integer, ByVal Ctr
     As Integer, Value As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Ctr	Counter number. Range:

PCI-9111, PCI-9112/cPCI-9112, PCI-9113, PCI-9114, PCI-9118	`0`
PCI-7248/cPCI-7248/PCI-7224, cPCI-7249R, PCI-7296, PCI-7348/PCI-7396	`0, 1, 2`
PCI-8554	`1 to 12`

Value	Returns the current count of the specified counter. Range: 0 to 65536 for binary mode (default), 0 to 9999 for BCD counting mode.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidCounter
```

CTR_Setup

Description

Configures the selected counter to operate in the specified mode.

Supported card(s)

9111, 9112, 9113, 9114, 9118, 7224, 7248, 7249, 7296, 7348, 7396, 8554

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 CTR_Setup (U16 CardNumber, U16 Ctr, U16 Mode,
     U32 Count, U16 BinBcd)
```

Visual Basic

```
CTR_Setup (ByVal CardNumber As Integer, ByVal Ctr
     As Integer, ByVal Mode As Integer, ByVal
     Count As Long, ByVal BinBcd As Integer) As
     Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Ctr Counter number. Range:

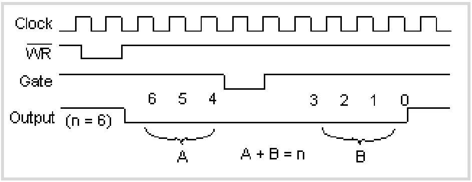
PCI-9111, PCI-9112/cPCI-9112, PCI-9113, PCI-9114, PCI-9118.	`0`
PCI-7224, PCI-7248/cPCI-7248/, cPCI-7249R, PCI-7296, PCI-7348/PCI-7396.	`0, 1, 2`
PCI-8554	`1 to 12`

Mode Counter oparating mode. Valid values:

```
TOGGLE_OUTPUT
PROG_ONE_SHOT
RATE_GENERATOR
SQ_WAVE_RATE_GENERATOR
SOFT_TRIG
HARD_TRIG
```

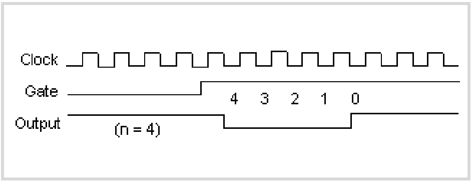
TOGGLE_OUTPUT: Toggle output from low to high on terminal count

The output goes low after the mode set operation and the counter begins to count down while the gate input is high. When terminal count is reached, the output goes high and remains high until the selected counter is set to a different mode. The following diagram shows the TOGGLE_OUTPUT mode timing.



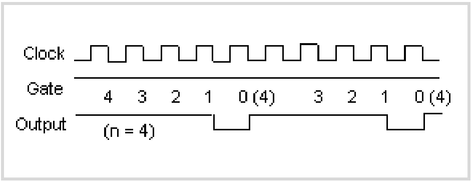
PROG_ONE_SHOT: Programmable one-shot

The output goes low following the rising edge of the gate input and goes high on terminal count. The following diagram shows the PROG_ONE_SHOT mode timing.



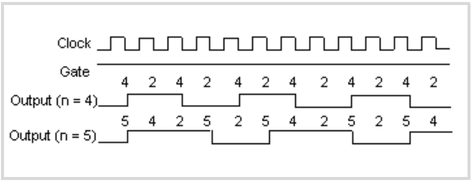
RATE_GENERATOR: Rate generator

The output goes low for one period of the clock input. Count indicates the period from one output pulse to the next. The following diagram shows the RATE_GENERATOR mode timing diagram.



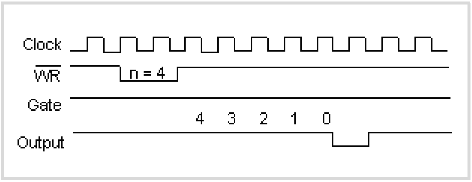
SQ_WAVE_RATE_GENERATOR: Square wave rate generator

The output stays high for one half of the count clock pulses and stays low for the other half. The following diagram shows the SQ_WAVE_RATE_GENERATOR mode timing.



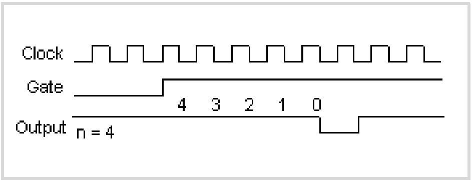
SOFT_TRIG: Software-triggered strobe

The output is initially high and the counter begins to count down while the gate input is high. On terminal count, the output goes low for one clock pulse, then goes high again. The following diagram shows the SOFT_TRIG mode timing.



HARD_TRIG: Hardware-triggered strobe

This mode is similar to SOFT_TRIG mode except that the gate input is used as a trigger to start counting. The following diagram shows the HARD_TRIG mode timing diagram.



Count	The period from one output pulse to the next.
BinBcd	Tells whether the counter operates as a 16-bit binary counter or as a 4-decade binary-coded decimal (BCD) counter. Valid values:

`BIN`	16-bit binary counter
`BCD`	4-decade BCD counter

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidCounter
```

CTR_Status

Description

Returns the status of the selected counter.

Supported card(s)

8554

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 CTR_Status (U16 CardNumber, U16 Ctr, U32
     *Value)
```

Visual Basic

```
CTR_Status (ByVal CardNumber As Integer, ByVal
     Ctr As Integer, Value As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Ctr	Counter number. Range is from 1 to 12.
Value	Returns the status of the specified counter. Refer to the card manual for more information.

Parameter(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidCounter
```

CTR_Update

Description

A new initial count is written to the selected counter without affecting the counter's programed mode.

Supported card(s)

9111, 9112, 9113, 9114, 9118, 7224, 7248, 7249, 7296, 7348, 7396, 8554

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 CTR_Update (U16 CardNumber, U16 Ctr, U32
     Count)
```

Visual Basic

```
CTR_Update (ByVal CardNumber As Integer, ByVal
     Ctr As Integer, ByVal Count As Long) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Ctr	Counter number. Range:

PCI-9111, PCI-9112/cPCI-9112, PCI-9113, PCI-9114, PCI-9118	`0`
PCI-7224, PCI-7248/cPCI-7248/, cPCI-7249R, PCI-7296, PCI-7348/PCI-7396	`0, 1, 2`
PCI-8554	`1 to 12`

Count	New count for the specified counter. Range:

`0 to 65536`	Binary mode (default)
`0 to 9999`	BCD counting mode

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidCounter
```

DI_7200_Config

Description

Informs the PCIS-DASK library of the trigger source and selected input mode with ID CardNumber. You must call this function before calling function to perform continuous digital input operation.

Supported card(s)

7200

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_7200_Config (U16 CardNumber, U16
     TrigSource, U16 ExtTrigEn, U16 TrigPol, U16
     I_REQ_Pol)
```

Visual Basic

```
DI_7200_Config (ByVal CardNumber As Integer,
     ByVal TrigSource As Integer, ByVal ExtTrigEn
     As Integer, ByVal TrigPol As Integer, ByVal
     I_REQ_Pol As Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

TrigSource The trigger mode for continuous digital input. Valid values:

`TRIG_INT_PACER`	Onboard programmable pacer
`TRIG_EXT_STROBE`	External signal trigger
`TRIG_HANDSHAKE`	Handshaking

ExtTrigEn External Trigger Enable, the valid values are:

`DI_WAITING`	Digital input sampling waits rising or falling edge of I_TRG to start DI.
`DI_NOWAITING`	Input sampling starts immediately.

TrigPol	Trigger polarity. Valid values:	
	`DI_TRIG_RISING`	I_TRG is rising edge active.
	`DI_TRIG_FALLING`	I_TRG is falling edge active.
I_REQ_Pol	I_REQ polarity. Valid values:	
	`IREQ_RISING`	I_REQ is rising edge active.
	`IREQ_FALLING`	I_REQ is falling edge active.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DI_7300A_Config

Description

Informs the PCIS-DASK library of the trigger source, port width, etc. selected for PCI7300A Rev.A/cPCI7300A Rev.A card with card ID CardNumber. You must call this function before calling function to perform continuous digital input operation.

Supported card(s)

7300A Rev.A

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_7300A_Config (U16 CardNumber, U16
     PortWidth, U16 TrigSource, U16 WaitStatus,
     U16 Terminator, U16 I_REQ_Pol, BOOLEAN
     ClearFifo, BOOLEAN DisableDI)
```

Visual Basic

```
DI_7300A_Config (ByVal CardNumber As Integer,
     ByVal PortWidth As Integer, ByVal TrigSource
     As Integer, ByVal WaitStatus As Integer,
     ByVal Terminator As Integer, ByVal I_REQ_Pol
     As Integer, ByVal ClearFifo As Byte, ByVal
     DisableDI As Byte) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

PortWidth Width of digital input port (PORT A). Valid values: 0, 8, 16, or 32.

TrigSource Trigger mode for continuous digital input. Valid values:

`TRIG_INT_PACER`	Onboard programmable pacer timer
`TRIG_EXT_STROBE`	External signal trigger
`TRIG_HANDSHAKE`	Handshaking
`TRIG_CLK_10MHz`	10 MHz clock
`TRIG_CLK_20MHz`	20 MHz clock

WaitStatus DI Wait Trigger Status. Valid values:

`P7300_WAIT_NO`	Input sampling starts immediately.
`P7300_WAIT_TRG`	Digital input sampling waits rising or falling edge of I_TRG to start DI.

Terminator PortA Terminator On/Off. Valid values:

`P7300_TERM_ON`	Terminator on
`P7300_TERM_OFF`	Terminator off

I_REQ_Pol I_REQ Polarity. This function is not implemented on PCI-7300A Rev.A or cPCI-7300A Rev.A card.

ClearFifo Valid values:

FALSE	Retain the FIFO data.
TRUE	Clear FIFO data before performing digital input.

DisableDI Valid values:

FALSE	Digital input operation is still active after DMA transfer is completed. Input data still goes into FIFO.
TRUE	Disable digital input operation immediately after DMA transfer is completed.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DI_7300B_Config

Description

Informs the PCIS-DASK library of the selected trigger source, port width, etc. for PCI-7300A Rev.B or cPCI-7300A Rev.B card with card ID. You must call this function before calling function to perform continuous digital input operation.

Supported card(s)

7300A Rev.B

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_7300B_Config (U16 CardNumber, U16
     PortWidth, U16 TrigSource, U16 WaitStatus,
     U16 Terminator, U16 I_Cntrl_Pol, BOOLEAN
     ClearFifo, BOOLEAN DisableDI)
```

Visual Basic

```
DI_7300B_Config (ByVal CardNumber As Integer,
     ByVal PortWidth As Integer, ByVal TrigSource
     As Integer, ByVal WaitStatus As Integer,
     ByVal Terminator As Integer, ByVal
     I_Cntrl_Pol As Integer, ByVal ClearFifo As
     Byte, ByVal DisableDI As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
PortWidth	Width of digital input port (PORT A). Valid values: 0, 8, 16, or 32.
TrigSource	Trigger mode for continuous digital input. Valid values:

`TRIG_INT_PACER`	Onboard programmable pacer timer
`TRIG_EXT_STROBE`	External signal trigger
`TRIG_HANDSHAKE`	Handshaking
`TRIG_CLK_10MHz`	10 MHz clock
`TRIG_CLK_20MHz`	20 MHz clock

WaitStatus DI Wait Trigger status. Valid values:

`P7300_WAIT_NO`	Input sampling starts immediately.
`P7300_WAIT_TRG`	Digital input sampling waits rising or falling edge of I_TRG to start DI.

Terminator PortA Terminator On/Off, the valid values are:

`P7300_TERM_ON`	Terminator on
`P7300_TERM_OFF`	Terminator off

I_Cntrl_Pol The polarity configuration. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are three groups of constants:

DIREQ

`P7300_DIREQ_POS`	DIREQ signal is rising edge active.
`P7300_DIREQ_NEG`	DIREQ signal is falling edge active.

DIACK

`P7300_DIACK_POS`	DIACK signal is rising edge active.
`P7300_DIACK_NEG`	DIACK signal is falling edge active.

DITRIG

`P7300_DITRIG_POS`	DITRIG signal is rising edge active.
`P7300_DITRIG_NEG`	DITRIG signal is falling edge active.

ClearFifo Valid values:

FALSE	Retain the FIFO data.
TRUE	Clear FIFO data before performing digital input.

DisableDI Valid values:

FALSE	Digital input operation is still active after DMA transfer is completed. Input data still goes into FIFO.
TRUE	Disable digital input operation immediately after DMA transfer is completed.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DI_9222_Config

Description

Inform the PCIS-DASK library of the trigger source, trigger mode, and trigger properties selected for the PCI-9222 with card ID CardNumber. You must call this function before calling function to perform continuous digital input operation.

Supported card(s)

9222

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DI_9222_Config (U16 CardNumber, U16
     ConfigCtrl, U16 TrigCtrl, U32 ReTriggerCnt,
     BOOLEAN AutoResetBuf)
```

Visual Basic

```
DI_9222_Config (ByVal CardNumber As Integer,
     ByVal ConfigCtrl As Integer, ByVal TrigCtrl
     As Integer, ByVal ReTriggerCnt As Long,
     ByVal AutoResetBuf As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ConfigCtrl	The setting for DI mode control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There is one group of constants:

Conversion Source Selection

```
P922x_DI_CONVSRC_INT
P922x_DI_CONVSRC_GPI0
P922x_DI_CONVSRC_GPI1
P922x_DI_CONVSRC_GPI2
P922x_DI_CONVSRC_GPI3
P922x_DI_CONVSRC_GPI4
P922x_DI_CONVSRC_GPI5
P922x_DI_CONVSRC_GPI6
P922x_DI_CONVSRC_GPI7
P922x_DI_CONVSRC_ADCONV
```

`P922x_DI_CONVSRC_DACONV`

TrigCtrl

The setting for DI Trigger control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are four groups of constants:

Trigger Mode Selection

`P922x_DI_TRGMOD_POST`

Trigger Source Selection

`P922x_DI_TRGSRC_SOFT`

`P922x_DI_TRGSRC_GPI0`

`P922x_DI_TRGSRC_GPI1`

`P922x_DI_TRGSRC_GPI2`

`P922x_DI_TRGSRC_GPI3`

`P922x_DI_TRGSRC_GPI4`

`P922x_DI_TRGSRC_GPI5`

`P922x_DI_TRGSRC_GPI6`

`P922x_DI_TRGSRC_GPI7`

Trigger Polarity

`P922x_DI_TrgPositive`

`P922x_DI_TrgNegative`

Re-Trigger Mode Enable

`P922x_DI_EnReTigger`

ReTriggerCnt

The accepted trigger times in an acquisition. The valid range of ReTriggerCnt is 0 to 4294967295. If the value of ReTriggerCnt is 0, the DI operation is triggered infinitely. The argument is valid only for re-trigger mode.

AutoResetBuf

FALSE

The DI buffers set by the DI_ContBufferSetup function are retained. You must call the DI_ContBufferReset function to reset the buffer.

TRUE

The DI buffers set by the DI_ContBufferSetup function are reset automatically by driver when the DI operation is completed.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidTriggerMode
ErrorConfigIoctl
```

DI_9223_Config

Description

Inform the PCIS-DASK library of the trigger source, trigger mode, and trigger properties selected for the PCI-9223 with card ID CardNumber. You must call this function before calling function to perform continuous digital input operation.

Supported card(s)

9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DI_9223_Config (U16 CardNumber, U16
     ConfigCtrl, U16 TrigCtrl, U32 ReTriggerCnt,
     BOOLEAN AutoResetBuf)
```

Visual Basic

```
DI_9223_Config (ByVal CardNumber As Integer,
     ByVal ConfigCtrl As Integer, ByVal TrigCtrl
     As Integer, ByVal ReTriggerCnt As Long,
     ByVal AutoResetBuf As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ConfigCtrl	The setting for DI mode control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There is one group of constants:

Conversion Source Selection

```
P922x_DI_CONVSRC_INT
P922x_DI_CONVSRC_GPI0
P922x_DI_CONVSRC_GPI1
P922x_DI_CONVSRC_GPI2
P922x_DI_CONVSRC_GPI3
P922x_DI_CONVSRC_GPI4
P922x_DI_CONVSRC_GPI5
P922x_DI_CONVSRC_GPI6
P922x_DI_CONVSRC_GPI7
P922x_DI_CONVSRC_ADCONV
```

`P922x_DI_CONVSRC_DACONV`

TrigCtrl

The setting for DI Trigger control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are four groups of constants:

Trigger Mode Selection

`P922x_DI_TRGMOD_POST`

Trigger Source Selection

`P922x_DI_TRGSRC_SOFT`

`P922x_DI_TRGSRC_GPI0`

`P922x_DI_TRGSRC_GPI1`

`P922x_DI_TRGSRC_GPI2`

`P922x_DI_TRGSRC_GPI3`

`P922x_DI_TRGSRC_GPI4`

`P922x_DI_TRGSRC_GPI5`

`P922x_DI_TRGSRC_GPI6`

`P922x_DI_TRGSRC_GPI7`

Trigger Polarity

`P922x_DI_TrgPositive`

`P922x_DI_TrgNegative`

Re-Trigger Mode Enable

`P922x_DI_EnReTigger`

ReTriggerCnt

The accepted trigger times in an acquisition. The valid range of ReTriggerCnt is 0 to 4294967295. If the value of ReTriggerCnt is 0, the DI operation is triggered infinitely. The argument is valid only for re-trigger mode.

AutoResetBuf

FALSE

The DI buffers set by the DI_ContBufferSetup function are retained. You must call the DI_ContBufferReset function to reset the buffer.

TRUE

The DI buffers set by the DI_ContBufferSetup function are reset automatically by driver when the DI operation is completed.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidTriggerMode
ErrorConfigIoctl
```

DI_AsyncCheck

Description

Checks the current status of asynchronous digital input operation.

Supported card(s)

7200, 7300A, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_AsyncCheck (U16 CardNumber, BOOLEAN
     *Stopped, U32 *AccessCnt)
```

Visual Basic

```
DI_AsyncCheck (ByVal CardNumber As Integer, Stopped
     As Byte, AccessCnt As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing asynchronous operation.
Stopped	Tells whether the asynchronous analog input operation has completed. If Stopped = TRUE, the digital input operation has stopped. Either the number of digital input indicated in the call that initiated the asynchronous digital input operation has been completed or an error has occurred. If Stopped = FALSE, the operation is not yet completed. (constants TRUE and FALSE are defined in DASK.H)
AccessCnt	The number of digital input data that has been transferred at the time the call to DI_AsyncCheck(). On PCI-7300A, AccessCnt is not used in DI_AsyncCheck() and DI_AsyncClear() since PLX9080 has no function or register to get the current amount of DMA transfer.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DI_AsyncClear

Description

Stops the asynchronous digital input operation.

Supported card(s)

7200, 7300A, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_AsyncClear (U16 CardNumber, U32
     *AccessCnt)
```

Visual Basic

```
DI_AsyncClear (ByVal CardNumber As Integer,
     AccessCnt As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing asynchronous operation.
AccessCnt	The number of digital input data that has been transferred at the time the call to DI_AsyncClear(). If double-buffered mode is enabled, AccessCnt returns the next position after the position the last data is stored in the circular buffer. If the AccessCnt execeeds the half size of circular buffer, call DI_AsyncDblBufferTransfer twice to get the data. On PCI-7300A, AccessCnt is not used in DI_AsyncCheck() and DI_AsyncClear() since PLX9080 has no function or register to get the current amount of DMA transfer.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DI_AsyncDblBufferHalfReady

Description

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

Supported card(s)

7200, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_AsyncDblBufferHalfReady (U16 CardNumber,
     BOOLEAN *HalfReady)
```

Visual Basic

```
DI_AsyncDblBufferHalfReady(ByVal CardNumber As
     Integer, HalfReady As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing asynchronous double-buffered operation.
HalfReady	Tells whether the next half buffer of data is available. For the PCI-7200, if HalfReady is TRUE, you can call DI_AsyncDblBufferTransfer() to copy the data to your user buffer. Constants TRUE and FALSE are defined in DASK.H.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DI_AsyncDblBufferHandled

Description

Notifies PCIS-DASK the ready buffer has been handled in user application. The data are transferred through DMA to the user's buffer directly. Therefore, while half buffer of data is ready (using DI_AsyncDblBufferHalfReady to check the ready status), the data in the ready buffer can be handled directly and don't needed to be copied to another transfer buffer. This mechanism eliminates the time taken for memory copy and another memory space for data transfer; however, PCIS-DASK couldn't know if the data in the ready buffer have been handled (in user application). If the data is handled, the user application needs an interface to notify PCIS-DASK this information. The function, DI_AsyncDblBufferHandled, is used to for this purpose.

Supported card(s)

9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DI_AsyncDblBufferHandled (U16 CardNumber)
```

Visual Basic

```
DI_AsyncDblBufferHandled (ByVal CardNumber As
     Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorContIoNotAllowed
ErrorNotDoubleBufferMode
```

DI_AsyncDblBufferMode

Description

Enables or disables double-buffered data acquisition mode.

Supported card(s)

7200, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_AsyncDblBufferMode (U16 CardNumber,
     BOOLEAN Enable)
```

Visual Basic

```
DI_AsyncDblBufferMode (ByVal CardNumber As
     Integer, ByVal Enable As Byte) As Integer
```

Parameter(s)

CardNumber ID of the card set for double-buffered mode.

Enable Tells whether the double-buffered mode is enabled or not. Constants TRUE and FALSE are defined in DASK.H.

TRUE	Double-buffered mode is enabled.
FALSE	Double-buffered mode is disabled.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DI_AsyncDblBufferOverrun

Description

Checks or clears overrun status of the double-buffered/multi-buffered digital input operation.

Supported card(s)

7200, 7300A, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_AsyncDblBufferOverrun (U16 CardNumber, U16
    op, U16 *overrunFlag)
```

Visual Basic

```
DI_AsyncDblBufferOverrun (ByVal CardNumber As
    Integer, ByVal op As Integer, overrunFlag As
    Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card set for double-buffered mode.
op	Check/Clear overrun status/flag.

`0`	Check the overrun status.
`1`	Clear the overrun flag.

overrunFlag	Returned overrun status.

`0`	No overrun occurred.
`1`	Overrun occurred.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DI_AsyncDblBufferToFile

Description

For double buffer mode of continuous DI, if the continuous DI function is DI_ContReadPortToFile, call this function to log the data of the circular buffer into a disk file.

Supported card(s)

9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DI_AsyncDblBufferToFile (U16 CardNumber)
```

Visual Basic

```
DI_AsyncDblBufferToFile (ByVal CardNumber As
     Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorContIoNotAllowed
ErrorNotDoubleBufferMode
```

DI_AsyncDblBufferTransfer

Description

Depending on the selected continuous DI function, half of the data of the circular buffer is logged into the user buffer, if continuous DI function is DI_ContReadPort or into a disk file, if continuous DI function is DI_ContReadPortToFile. The data saved in the file is written in binary format with the lower byte first (little endian).

You may execute this function repeatedly to return sequential half buffers of the data.

For PCI-7300A_RevB, DI_AsyncDblBufferTransfer does not perform memory transfer but notifies the pci-dask.dll that the data stored in the buffer has been handled.

Supported card(s)

7200, 7300A

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_AsyncDblBufferTransfer (U16 CardNumber,
     void *Buffer)
```

Visual Basic

```
DI_AsyncDblBufferTransfer (ByVal CardNumber As
     Integer, Buffer As Any) As Integer
```

Parameter(s)

CardNumber	ID of the card performing asynchronous double-buffered operation.
Buffer	The user buffer where the data is to be copied. This argument has no use when data is to be saved into a disk file.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorNotDoubleBufferMode
```

DI_AsyncMultiBuffersHandled

Description

Notifies PCIS-DASK the ready buffers have been handled in user application. The data are transferred through DMA to the user's buffers directly. Therefore, while multi-buffer of data are ready (using DI_AsyncMultiBufferNextReady to check the ready status), the data in the ready buffers can be handled directly and don't needed to be copied to another transfer buffers. This mechanism eliminates the time taken for memory copy and another memory space for data transfer; however, PCIS-DASK couldn't know if the data in the ready buffers have been handled (in user application). If the data is handled, the user application needs an interface to notify PCIS-DASK this information. The function, DI_AsyncMultiBuffersHandled, is used to for this purpose.

Supported card(s)

7300A

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DI_AsyncMultiBuffersHandled (U16 CardNumber,
     U16 bufcnt, U16* bufs)
```

Visual Basic

```
DI_AsyncMultiBuffersHandled (ByVal CardNumber As
     Integer, ByVal bufcnt As Integer, bufs As
     Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
bufcnt	Buffer counts have been handled.
bufs	Array of the number of handled buffers. For example, if bufcnt = 4, bufs[0] = 4, bufs[1] = 5, bufs[2] = 6, and bufs[3] = 7, calling the function to notify PCIS-DASK that Buffer 4, 5, 6, and 7 have been handled.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidBufferID
```

DI_AsyncMultiBufferNextReady

Description

Checks whether the next buffer of data in circular buffer is ready for transfer during an asynchronous multi-buffered digital input operation. The returned BufferId is the index of the most recently available (newest available) buffer.

Supported card(s)

7300A

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_AsyncMultiBufferNextReady (U16 CardNumber,
     BOOLEAN *NextReady, U16 *BufferId)
```

Visual Basic

```
DI_AsyncMultiBufferNextReady (ByVal CardNumber As
     Integer, NextReady As Byte, BufferId As
     Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing asynchronous multi-buffered operation.
NextReady	Tells whether the next data buffer is available. If NextReady = TRUE, you can handle the data in the buffer. Constants TRUE and FALSE are defined in DASK.H.
BufferId	Returns the index of the ready buffer.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DI_AsyncReTrigNextReady

Description

Checks whether the data associated to the next trigger signal is ready during an asynchronous retriggered digital input operation.

Supported card(s)

9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DI_AsyncReTrigNextReady (U16 CardNumber,
     BOOLEAN *Ready, BOOLEAN *StopFlag, U16
     *RdyTrigCnt)
```

Visual Basic

```
DI_AsyncReTrigNextReady (ByVal CardNumber As
     Integer, Ready As Byte, StopFlag As Byte,
     RdyTrigCnt As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Ready	Tells whether the data associated with the next trigger signal is available. Constants TRUE and FALSE are defined in DASK.H.
StopFlag	Tells whether the asynchronous digital input operation is complete. If StopFlag is TRUE, the digital input operation has stopped. If StopFlag is FALSE, the operation is not yet completed. Constants TRUE and FALSE are defined in DASK.H.
RdyTrigCnt	This argument returns the count of trigger signal that occurred if re-trigger count is definite. If the re-trigger count is infinite, this argument returns the index of the buffer that stored the data after the most recent trigger signal trigger is generated.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DI_ContBufferReset

Description

This function reset all the buffers set by function DI_ContBufferSetup for continuous digital input. The function has to be called if the data buffers won't be used.

Supported card(s)

9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DI_ContBufferReset (U16 CardNumber)
```

Visual Basic

```
DI_ContBufferReset (ByVal CardNumber As Integer)
     As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorContIoNotAllowed
```

DI_ContBufferSetup

Description

This function set up the buffer for continuous digital input operation. The function has to be called repeatedly to setup all of the data buffers. (The maximum number of buffers is 2.)

Supported card(s)

9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DI_ContBufferSetup (U16 CardNumber, VOID
     *Buffer, U32 ReadCount, U16 *BufferId)
```

Visual Basic

```
DI_ContBufferSetup (ByVal CardNumber As Integer,
     Buffer As Any, ByVal ReadCount As Long,
     BufferId As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Buffer	The starting address of the memory to be stored the read data.
ReadCount	The size (in samples) of the buffer and its value must be even.
BufferId	Returns the index of the buffer currently set up.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
```

DI_ContMultiBufferSetup

Description

Sets up the buffer for multi-buffered digital input. The function has to be called repeatedly to setup all data buffers (maximum eight buffers).

Supported card(s)

7300A

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_ContMultiBufferSetup (U16 CardNumber, void
     *Buffer, U32 ReadCount, U16 *BufferId)
```

Visual Basic

```
DI_ContMultiBufferSetup (ByVal CardNumber As
     Integer, Buffer As Any, ByVal ReadCount As
     Long, BufferId As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Buffer	Starting address of the memory containing the input data.
ReadCount	Size (in samples) of the buffer and its value. Value must be even.
BufferId	Returns the index of the buffer currently being set up.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
```

DI_ContMultiBufferStart

Description

Starts multi-buffered continuous digital input on the specified digital input port at a rate closest to the specified rate.

Supported card(s)

7300A

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_ContMultiBufferStart (U16 CardNumber, U16
     Port, F64 SampleRate)
```

Visual Basic

```
DI_ContMultiBufferStart (ByVal CardNumber As
     Integer, ByVal Port As Integer, ByVal
     SampleRate As Double) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Port	Digital input port number. For PCI-7300A and cPCI-7300A, this argument must be set to 0.
SampleRate	Sampling rate you want for digital input in hertz (samples per second). The maximum rate depends on the card type and your computer system. This argument is only valid when the DI trigger mode is set as internal programmable pacer (TRIG_INT_PACER) by calling DI_7300A_Config() or DI_7300B_Config().

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorContIoNotAllowed
```

DI_ContReadPort

Description

Performs continuous digital input on the specified digital input port at a rate closest to the specified rate.

Supported card(s)

7200, 7300A, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_ContReadPort (U16 CardNumber, U16 Port,
     void *Buffer, U32 ReadCount, F64 SampleRate,
     U16 SyncMode)
```

Visual Basic

```
DI_ContReadPort (ByVal CardNumber As Integer,
     ByVal Port As Integer, Buffer As Any, ByVal
     ReadCount As Long, ByVal SampleRate As
     Double, ByVal SyncMode As Integer) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Port	Digital input port number. For PCI-7200, cPCI-7200, PCI-7300A, cPCI-7300A, PCI-9222, and PCI-9223, this argument must be set to 0.
Buffer	Starting address of the memory containing the input data. The memory must allocate enough space to store input data. This buffer has no use when double-buffered mode is enabled.
ReadCount	When double-buffered mode is disabled, ReadCount is the number of input operation to be performed. For double-buffered acquisition, ReadCount is the size (in samples) of the circular buffer. Its value must be even.
SampleRate	Sampling rate you want for digital input in hertz (samples per second). The maximum rate depends on the card type and your computer system.

PCI-7200, PCI-7300:

This argument is only useful if the DI trigger mode is set as internal programmable.

Pacer (TRIG_INT_PACER) by calling DI_7200_Config() or DI_7300_Config(). For other settings, set this argument as CLKSRC_EXT_SampRate.

PCI-9222, PCI-9223:

This argument is only useful if the DI conversion source is set as internal conversion

Source (P922x_DI_CONVSRC_INT) by calling DI_9222_Config() or DI_9223_Config(). For other settings, this argument is ignored. The maximum sample rate is 2000000 (2 MHz).

SyncMode Tells whether the operation is performed synchronously or asynchronously. Valid values:

`SYNCH_OP`	Synchronous A/D conversion, that is, the function does not return until the A/D operation is completed.
`ASYNCH_OP`	Asynchronous A/D conversion

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
```

DI_ContReadPortToFile

Description

Performs continuous digital input on the specified digital input port at a rate closest to the specified rate and saves the acquired data in a disk file. The data is written to disk in binary format, with the lower byte first (little endian). Refer to Appendix D: Data File Format for the data file structure.

Supported card(s)

7200, 7300A, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_ContReadPortToFile (U16 CardNumber, U16
     Port, U8 *FileName, U32 ReadCount, F64
     SampleRate, U16 SyncMode)
```

Visual Basic

```
DI_ContReadPortToFile (ByVal CardNumber As
     Integer, ByVal Port As Integer, ByVal
     FileName As String, ByVal ReadCount As Long,
     ByVal SampleRate As Double, ByVal SyncMode
     As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Port	Digital input port number. For PCI-7200, cPCI-7200, PCI-7300A, cPCI-7300A, PCI-9222, and PCI-9223, this argument must be set to 0.
FileName	The file where acquired data is stored.
ReadCount	If double-buffered mode is disabled, ReadCount is the number of input operation to be performed. For double-buffered acquisition, ReadCount is the size (in samples) of the circular buffer. Its value must be even.

SampleRate	The sampling rate you want for digital input in hertz (samples per second). The maximum rate depends on the card type and your computer system.

PCI-7200, PCI-7300:

This argument is only useful if the DI trigger mode is set as internal programmable.

Pacer (TRIG_INT_PACER) by calling DI_7200_Config() or DI_7300_Config(). For other settings, set this argument as CLKSRC_EXT_SampRate.

PCI-9222, PCI-9223:

This argument is only useful if the DI conversion source is set as internal conversion

Source (P922x_DI_CONVSRC_INT) by calling DI_9222_Config() or DI_9223_Config(). For other settings, this argument is ignored. The maximum sample rate is 2000000 (2 MHz).

SyncMode	Tells whether the operation is performed synchronously or asynchronously. Valid values:

`SYNCH_OP`	Synchronous A/D conversion, that is, the function does not return until the A/D operation is completed.
`ASYNCH_OP`	Asynchronous A/D conversion

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorInvalidSampleRate
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
```

DI_ContStatus

Description

While performing continuous DI conversions, this function is called to get the DI status. Refer to the device manual for supported DI status.

Supported card(s)

7200, 7300A, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_ContStatus (U16 CardNumber, U16 *Status)
```

Visual Basic

```
DI_ContStatus (ByVal CardNumber As Integer,
     Status Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Status	The continuous DI status returned. The description of the Parameter(s) Status for various card types is the following:

PCI-7200

`bit 0`	1 = D/I FIFO is full (overrun).
`bit 1`	1 = D/O FIFO is empty (underrun).
`bit 2~15`	Not used

PCI-7300A_RevA

`bit 0`	1 = DI FIFO is full during input sampling and some data were lost.
`bit 1`	1 = DI FIFO is full.
`bit 2`	1 = DI FIFO is empty.
`bit 3~15`	Not used

PCI-7300A_RevB

`bit 0`	1 = DI FIFO is full during input sampling and some data were lost.
`bit 1`	1 = DI FIFO is full.
`bit 2`	1 = DI FIFO is empty.
`bit 3~15`	Not used

PCI-9222 and PCI-9223

`bit 0`	1 = FIFO is empty.
`bit 1`	1 = FIFO is almost empty.
`bit 2`	1 = FIFO is almost full.
`bit 3`	1 = FIFO is full.
`bit 4-7`	Not used.
`bit 8`	1 = DI acquisition is in progress.
`bit 9`	1 = DI acquisition is done.
`bit 10-15`	Not used.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
```

DI_EventCallBack

Description

Controls and notifies the user's application when a specified DAQ event occurs. The notification is performed through a user-specified callback function. The event message is removed automatically after calling DI_Async_Clear. The event message can also be manually removed by set the Mode parameter to 0.

Supported card(s)

7200 (Win32 only), 7300A, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_EventCallBack (U16 CardNumber, I16 mode,
     I16 EventType, U32 callbackAddr)
```

Visual Basic

```
DI_EventCallBack (ByVal CardNumber As Integer,
     ByVal mode As Integer, ByVal EventType As
     Integer, ByVal callbackAddr As Long) As
     Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

mode Add or remove the event message. Valid values:

`0`	Remove
`1`	Add

EventType Event criteria. Valid values:

`DIEnd`	Notification that the asynchronous digital input operation has been completed.
`DBEvent`	Notification that the next half buffer of data in circular buffer is ready for transfer.
`TrigEvent`	Notifies that the data associated to the next trigger signal is available (only for PCI-9222/9223).

callbackAddr Address of the user callback function. The PCIS-DASK calls this function when the specified event occurs. If you want to remove the event message, set callbackAddr to 0.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DI_GetView

Description

Returns the mapped buffer address of the memory allocated in the driver for continuous AI operation during system startup.

Supported card(s)

7200

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_GetView(U16 CardNumber, U32 *pView)
```

Visual Basic

```
DI_GetView (ByVal CardNumber As Integer, pView As
     Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
pView	Mapped buffer address of the memory allocated in the driver during system startup.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
```

DI_InitialMemoryAllocated

Description

Returns the mapped buffer address of the memory allocated in the driver for continuous DI operation during system startup. The size of the allocated memory can be acquired by using the DI_InitialMemoryAllocated function.

Supported card(s)

7200, 7300A, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_InitialMemoryAllocated (U16 CardNumber,
     U32 *MemSize)
```

Visual Basic

```
DI_InitialMemoryAllocated (ByVal CardNumber As
     Integer, MemSize As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
MemSize	Available memory size for continuous DI in device driver of the card. The unit is KB (1024 bytes).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
```

DI_ReadLine

Description

Reads the digital logic state of the digital line in the specified port.

Supported card(s)

6202, 6208V/16V/08A, 6308V/08A, 7200, 7230, 7233, 7224, 7248, 7249, 7250/51, 7252, 7256, 7258, 7260, 7296, 7300A, 7348, 7396, 7432, 7433, 7442, 7443, 7444, 7452, 8554, 9111, 9112, 9114, 9116, 9118, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_ReadLine (U16 CardNumber, U16 Port, U16
     Line, U16 *State)
```

Visual Basic

```
DI_ReadLine (ByVal CardNumber As Integer, ByVal
     Port As Integer, ByVal Line As Integer,
     State As Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Port Digital input port number. Valid values:

PCI-6202	`P6202_ISO0` `P6202_TTL0`
PCI-6208V/16V/08A	`0`
PCI-6308V/08A	`0`
PCI-7200	`0`
cPCI-7200	`0, 1 (auxiliary input port)`
PCI-7230/cPCI-7230	`0`
PCI-7233	`0`
PCI-7224	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH`
PCI-7248/cPCI-7248	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2CL, Channel_P2CH`

cPCI-7249R	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH, Channel_P1AE, Channel_P1BE, Channel_P1CE, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2CL, Channel_P2CH, Channel_P2AE, Channel_P2BE, Channel_P2CE`
PCI-7250/51	`0 through 3`
cPCI-7252	`0`
PCI-7256	`0`
PCI-7258	`0`
PCI-7260	`0`
PCI-7296	`Channel_P3B, Channel_P3C, Channel_P3CL, Channel_P3CH, Channel_P4A, Channel_P4B, Channel_P4C, Channel_P4CL, Channel_P4CH`
PCI-7348	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2`
PCI-7396	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2, Channel_P3A, Channel_P3B, Channel_P3C,Channel_P3, Channel_P4A, Channel_P4B, Channel_P4C,Channel_P4`
PCI-7300A/cPCI-7300A	`1 (auxiliary input port)`
PCI-7432/cPCI-7432	`0`
cPCI-7432R	`0`
PCI-7433/cPCI-7433	`PORT_DI_LOW, PORT_DI_HIGH`
cPCI-7433R	`PORT_DI_LOW, PORT_DI_HIGH`
PCI-7442	`P7442_CH0, P7442_CH1, P7442_TTL0, P7442_TTL1`
PCI-7443	`P7443_CH0, P7443_CH1, P7443_CH2, P7443_CH3, P7443_TTL0, P7443_TTL1`

PCI-7444	`P7444_TTL0, P7444_TTL1`
PCI-7452	`0 to 3`
PCI-8554	`0`
PCI-9111	`P9111_CHANNEL_DI,` `P9111_CHANNEL_EDI`
PCI-9112/cPCI-9112	`0`
PCI-9114	`0`
cPCI-9116	`0`
PCI-9118	`0`
PCI-9221	`0`
PCI-9222	`0`
PCI-9223	`0`
PCI-9524	`0`

Line Digital line to be read. Valid values:

PCI-66202	`0 to 15 (for P6202_ISO0)` `0 to 7 (for P6202_TTL0)`
PCI-6208V/16V/08A	`0 to 3`
PCI-6308V/08A	`0 to 3`
PCI-7200/cPCI-7200	`0 to 31 (for port 0)` `0 to 3 (for auxiliary input port of cPCI-7200)`
PCI-7230/cPCI-7230	`0 to 15`
PCI-7233	`0 to 31`
PCI-7248/cPCI-7248/ PCI-7224	`0 to 7`
cPCI-7249R	`0 to 7`
PCI-7250/51	`0 to 7`
cPCI-7252	`0 to 15`
PCI-7256	`0 to 15`
PCI-7258	`0 to 1`
PCI-7260	`0 to 7`
PCI-7296	`0 to 7`
PCI-7300A/cPCI-7300A	`0 to 3`

PCI-7396/PCI-7348	`0 to 23 (for Channel_Pn, where n is the channel number) or`
	`0 to 7 (for Channel_PnA, Channel_PnB, Channel_PnC, where n is the channel number)`

PCI-7432/cPCI-7432/ cPCI-7432R	`0 to 31`
PCI-7433/cPCI-7433/ cPCI-7433R	`0 to 31`

PCI-7442	`0 to 31 (for P7442_CH0/ P7442_CH1)`
	`0 to 15 (for P7442_TTL0/ P7442_TTL1)`

PCI-7443	`0 to 31 (for P7443_CH0/ P7443_CH1, P7443_CH2, P7443_CH3)`
	`0 to 15 (for P7443_TTL0/ P7443_TTL1)`

PCI-7444	`0 to 15`
PCI-7452	`0 to 31`
PCI-8554	`0 to 7`

PCI-9111	`0 to 15 (for P9111_CHANNEL_DI) or`
	`0 to 7 (for P9111_CHANNEL_EDI)`

PCI-9112/cPCI-9112	`0 to 15`
PCI-9114	`0 to 15`
cPCI-9116	`0 to 7`
PCI-9118	`0 to 3`
PCI-9221	`0 to 7`
PCI-9222	`0 to 15`
PCI-9223	`0 to 15`
PCI-9524	`0 to 7`

State Returns the digital logic state of the specified line to 0 or 1.

Return Code(s)

```
NoError
```

```
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

DI_ReadPort

Description

Reads the digital data from the specified digital input port.

Supported card(s)

6202, 6208V/16V/08A, 6308V/08A, 7200, 7230, 7233, 7224, 7248, 7249, 7250/51, 7252, 7256, 7258, 7260, 7296, 7300A, 7348, 7396, 7432, 7433, 7434, 7442, 7443, 7444, 7452, 8554, 9111, 9112, 9114, 9116, 9118, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DI_ReadPort (I16 CardNumber, U16 Port, U32
     *Value)
```

Visual Basic

```
DI_ReadPort (ByVal CardNumber As Integer, ByVal
     Port As Integer, Value As Long) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Port Digital input port number. Valid values:

PCI-6202	`P6202_ISO0` `P6202_TTLO`
PCI-6208V/16V/08A	`0`
PCI-6308V/08A	`0`
PCI-7200/cPCI-7200	`0`
cPCI-7200	`0, 1 (auxiliary digital input port)`
PCI-7230/cPCI-7230	`0`
PCI-7233	`0`
PCI-7224	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH`
PCI-7248/cPCI-7248	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2CL, Channel_P2CH`

cPCI-7249R	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH, Channel_P1AE, Channel_P1BE, Channel_P1CE, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2CL, Channel_P2CH, Channel_P2AE, Channel_P2BE, Channel_P2CE`
PCI-7250/51	`0 through 3`
cPCI-7252	`0`
PCI-7256	`0`
PCI-7258	`0`
PCI-7260	`0`
PCI-7296	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2CL, Channel_P2CH, Channel_P3A, Channel_P3B, Channel_P3C, Channel_P3CL, Channel_P3CH, Channel_P4A, Channel_P4B, Channel_P4C, Channel_P4CL, Channel_P4CH`
PCI-7300A/cPCI-7300A	`1(auxiliary digital input port)`
PCI-7348	`Channel_P2, Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1, Channel_P2A, Channel_P2B, Channel_P2C`
PCI-7396	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2, Channel_P3A, Channel_P3B, Channel_P3C, Channel_P2, Channel_P3A, Channel_P3B, Channel_P3C`
PCI-7432/cPCI-7432	`0`
cPCI-7432R	`0, P7432R_DI_SLOT`
PCI-7433/cPCI-7433	`PORT_DI_LOW, PORT_DI_HIGH`
cPCI-7433R	`PORT_DI_LOW, PORT_DI_HIGH, P7433R_DI_SLOT`

PCI-7442	`P7442_CH0, P7442_CH1, P7442_TTL0, P7442_TTL1`
PCI-7443	`P7443_CH0, P7443_CH1, P7443_CH2, P7443_CH3, P7443_TTL0, P7443_TTL1`
PCI-7444	`P7444_TTL0, P7444_TTL1`
PCI-7452	`0 to 3`
cPCI-7434R	`P7434R_DI_SLOT`
PCI-8554	`0`
PCI-9111	`P9111_CHANNEL_DI, P9111_CHANNEL_EDI`
PCI-9112/cPCI-9112	`0`
PCI-9114	`0`
cPCI-9116	`0`
PCI-9118	`0`
PCI-9221	`0`
PCI-9222	`0`
PCI-9223	`0`
PCI-9524	`0`

NOTE:

The value, Channel_Pn, for argument Port is defined as all of the ports (Port A, B and C) in channel n.

Value Returns the digital data read from the specified port. Valid values:

PCI-6202	`16-bit data (for P6202_ISO0)`
	`8-bit data (for P6202_TTLO)`

PCI-6208V/16V/08A	`4-bit data`
PCI-6308V/08A	`4-bit data`

PCI-7200/cPCI-7200	`32-bit data`
	`4-bit data (for auxiliary input port of cPCI-7200)`

PCI-7230/cPCI-7230	`16-bit data`
PCI-7233	`32-bit data`

PCI-7248/cPCI-7248/ PCI-7224	`8-bit data`
cPCI-7249R	`8-bit data`
PCI-7250/51	`8-bit data`
cPCI-7252	`16-bit data`
PCI-7256	`16-bit data`
PCI-7258	`2-bit data`
PCI-7260	`8-bit data`
PCI-7296	`8-bit data`
PCI-7300A/cPCI-7300A	`4-bit data`
PCI-7396/PCI-7348	`24-bit data (for Channel_Pn, where n is the channel number) or 8-bit data (for Channel_PnA, Channel_PnB, Channel_PnC, where n is the channel number`
PCI-7432/cPCI-7432/ cPCI-7433R	`32-bit data`
PCI-7433/cPCI-7433/ cPCI-7434	`32-bit data`

PCI-7442	`32-bit data (for P7442_CH0/ P7442_CH1)`
	`16-bit data (for P7442_TTL0/ P7442_TTL1)`

PCI-7443	`32-bit data (for P7443_CH0/ P7443_CH1, P7443_CH2, P7443_CH3)`
	`16-bit data (for P7443_TTL0/ P7443_TTL1)`

PCI-7444	`16-bit data`
PCI-7452	`32-bit data`
PCI-8554	`8-bit data`
PCI-9111	`16-bit data (for P9111_CHANNEL_DI) or 8-bit data (for P9111_CHANNEL_EDI)`
PCI-9112/cPCI-9112	`16-bit data`
PCI-9114	`16-bit data`
cPCI-9116	`8-bit data`

PCI-9118	`4-bit data`
PCI-9221	`8-bit data`
PCI-9222	`16-bit data`
PCI-9524	`8-bit data`

The data format for Channel_Pn is illustrated below:

	Ignore	**PORT C**	**PORT B**	**PORT A**
Bit	31 - 24	23 - 16	15 - 8	7 - 0

Return Code(s)

```
NoError, CardNotRegistered
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DI_SetTimeOut

Description

Sets Timeout period for Sync. mode continuous DI. While the function is called, the Sync. mode continuous DI acquisition is stopped even when it is not completed.

Supported card(s)

9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DI_SetTimeOut (U16 CardNumber, U32 TimeOut)
```

Visual Basic

```
DI_SetTimeOut (ByVal CardNumber As Integer, ByVal
    TimeOut As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
TimeOut	Timeout period (ms).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DIO_7300SetInterrupt

Description

Controls the interrupt sources (AuxDI0 and Timer 2) of local interrupt system for PCI-7300A and cPCI-7300A, and returns two interrupt events. When an interrupt is generated, the corresponding interrupt event is signaled. The application can use Win32 wait functions, such as WaitForSingleObject or WaitForMultipleObjects, to check the interrupt event status.

Supported card(s)

7300A

Syntax

Microsoft C/C++ and Borland C++

```
I16 DIO_7300SetInterrupt (U16 CardNumber, I16
     AuxDIEn, I16 T2En, HANDLE *hEvent)
```

Linux C/C++

```
I 16 DIO_7300SetInterrupt(U16 CardNumber, I16
     AuxDIEn, I16 T2En, void
     (*event1_handler)(int), void
     (*event2_handler)(int))
```

Visual Basic

```
DIO_7300SetInterrupt (ByVal CardNumber As
     Integer, ByVal AuxDIEn As Integer, ByVal
     T2En As Integer, hEvent As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
AuxDIEn	Control value for AUXDI interrupt. Valid values:

`0`	Disabled
`1`	Enabled

T2En	Control value for Timer2 interrupt. Valid values:

`0`	Disabled
`1`	Enabled

hEvent (Win32 only)

The returned local interrupt event handles. The status of the interrupt event indicates whether an interrupt is generated or not.

event1_handler (Linux Only)

Address of the user callback function. The PCIS-DASK calls this function when the specified AUXDI event occurs. If you do not want to use the callback function, set callbackAddr to 0.

event2_handler (Linux Only)

Address of the user callback function. The PCIS-DASK calls this function when the specified T2 event occurs. If you do not want to use the callback function, set callbackAddr to 0.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DIO_AUXDI_EventMessage (Win32 only)

Description

Controls the AUXDI interrupt and notifies the user's application when an interrupt event occurs. The notification is performed through a user-specified callback function or Windows PostMessage API.

Supported card(s)

7300A

Syntax

Microsoft C/C++ and Borland C++

```
I16 DIO_AUXDI_EventMessage (U16 CardNumber, I16
     AuxDIEn, HANDLE windowHandle, U32 message,
     void *callbackAddr())
```

Visual Basic

```
DIO_ AUXDI _EventMessage (ByVal CardNumber As
     Integer, ByVal AuxDIEn As Integer, ByVal
     windowHandle As Long, ByVal message As Long,
     ByVal callbackAddr As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
AuxDIEn	Control value for AUXDI interrupt. Valid values: `0` Disabled `1` Enabled
windowHandle	Handle to inform Windows that you want to receive a message when the specified AUXDI event occurs. This function is disabled when set to 0.

message	The user-defined message. When the specified AUXDI event happens, the PCIS-DASK passes this message back to you. Message can be of any value. In Windows, you can create your own messages or select from any Windows predefined messages such as WM_PAINT. However, you may define your own messages by using any value that ranges from WM_USER (0x400) to 0x7fff. This range is reserved for user-defined messages.
callbackAddr	Address of the user callback function. The PCIS-DASK calls this function when the specified AUXDI event occurs. If you do not want to use the callback function, set callbackAddr to 0.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DIO_GetCOSLatchData

Description

Gets the DI data with data width of 8-bit or 16-bit latched in the COS Latch register while the Change-of-State (COS) interrupt occurred.

Supported card(s)

7256, 7260

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DIO_GetCOSLatchData(U16 CardNumber, U16
     *CosLData)
```

Visual Basic

```
DIO_GetCOSLatchData (ByVal CardNumber As Integer,
     CosLData As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
CosLData	Returns the DI data latched in the COS Latch register when the Change-of-State (COS) interrupt occurred.

PCI-7256	`16-bit data`
PCI-7260	`8-bit data`

Return Code(s)

```
NoError
CardNotRegistered
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DIO_GetCOSLatchData32

Description

Gets the 32-bit width DI data latched in the COS Latch register while the Change-of-State (COS) interrupt occurs.

Supported card(s)

7442, 7443, 7452

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DIO_GetCOSLatchData32(U16 CardNumber, U8
     Port, U32 *CosLData)
```

Visual Basic

```
DIO_GetCOSLatchData32 (ByVal CardNumber As
     Integer, ByVal Port As Byte, CosLData As
     Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Port	Digital input port number. Valid values:

PCI-7442	`0 to 1`
PCI-7443	`0 to 3`
PCI-7452	`0 to 3`

CosLData	Returns the DI data latched in the COS Latch register while the Change-of-State(COS) interrupt occurs.

PCI-7442	`32-bit data`
PCI-7443	`32-bit data`
PCI-7452	`32-bit data`

Return Code(s)

```
NoError
CardNotRegistered
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DIO_INT_Event_Message (Win32 Only)

Description

Controls and notifies the user's application when a specified interrupt event occurs. The notification is performed through a user-specified callback function or the Windows PostMessage API.

When a new event message is added, it will remain active until you call this function by setting the argument mode to 0 that removes the specified interrupt event message. To remove a specified message, make sure to specify the event handle to be notified for the message.

Supported card(s)

7230, 7233, 7224, 7248, 7249, 7256, 7258, 7260, 7296, 7348, 7396, 7432, 7433, 7442, 7443, 7444, 7452, 8554

Syntax

Microsoft C/C++ and Borland C++

```
I16 DIO_INT_EventMessage (U16 CardNumber, I16
     mode, HANDLE evt, HANDLE windowHandle, U32
     message, U32 callbackAddr)
```

Visual Basic

```
DIO_INT_EventMessage (ByVal CardNumber As
     Integer, ByVal mode As Integer, ByVal evt As
     Long, ByVal windowHandle As Long, ByVal
     message As Long, ByVal callbackAddr As Long)
     As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
mode	The operation mode of adding or removing message.

`0`	Remove an existing message interrupt event defined argument evt.
`1`	Add a new message for an interrupt event defined argument evt.

evt	Handle of the INT event wishing to handle.

windowHandle	Handle to the window that you want to receive a Windows message when the specified INT event happens. If windowHandle is 0, no Windows messages will be sent.
message	The user-defined message. When the specified INT event happens, the PCIS-DASK sends this message back to you. The message can be of any value. In Windows, you can set a message to a value including any Windows predefined messages, such as WM_PAINT. However, to define your own message, you can use any value ranging from WM_USER (0x400) to 0x7fff. This range is reserved by Windows for user-defined messages.
callbackAddr	Address of the user callback function. The PCIS-DASK calls this function when the specified INT event occurs. If you do not want to use a callback function, set callbackAddr to 0.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DIO_INT1_EventMessage (Win32 Only)

Description

Controls the INT1 interrupt sources for a dual-interrupt system and notifies the user's application when an interrupt event occurs. The notification is performed through a user-specified callback function or the Windows PostMessage API.

Supported card(s)

7230, 7233, 7224, 7248, 7249, 7256, 7258, 7260, 7296, 7348, 7396, 7432, 7433, 7442, 7443, 7452, 8554

Syntax

Microsoft C/C++ and Borland C++

```
I16 DIO_INT1_EventMessage (U16 CardNumber, I16
     Int1Mode, HANDLE windowHandle, U32 message,
     void *callbackAddr())
```

Visual Basic

```
DIO_INT1_EventMessage (ByVal CardNumber As
     Integer, ByVal Int1Mode As Integer, ByVal
     windowHandle As Long, ByVal message As Long,
     ByVal callbackAddr As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Int1Mode	Interrupt mode of INT1. Valid values:

PCI-7248/cPCI-7248/cPCI-7249R/7296/7224

`INT1_DISABLE`	INT1 Disabled
`INT1_FP1C0`	INT1 by Falling edge of P1C0
`INT1_RP1C0_FP1C3`	INT1 by P1C0 Rising or P1C3 Falling
`INT1_EVENT_COUNTER`	INT1 by Event Counter down to zero

PCI-7230/cPCI-7230/7233/7432/7433

`INT1_DISABLE`	INT1 Disabled
`INT1_EXT_SIGNAL`	INT1 by External Signal

PCI-7442

`INT1_DISABLE`	NT1 Disabled
`INT1_COS0`	INT1 by COS of Port 0
`INT1_COS1`	INT1 by COS of Port 1

PCI-7443

`INT1_DISABLE`	NT1 Disabled
`INT1_COS0`	INT1 by COS of Port 0
`INT1_COS1`	INT1 by COS of Port 1
`INT1_COS2`	INT1 by COS of Port 2
`INT1_COS3`	INT1 by COS of Port 3

PCI-7452

`INT1_DISABLE`	INT1 Disabled
`INT1_COS0`	INT1 by COS of Port 0
`INT1_COS1`	INT1 by COS of Port 1
`INT1_COS2`	INT1 by COS of Port 2
`INT1_COS3`	INT1 by COS of Port 3

PCI-7256

`INT1_DISABLE`	INT1_DISABLE: INT1 Disabled
`INT1_COS`	INT1_COS: INT1 by COS
`INT1_CH0`	INT1_CH0: INT1 by CH0

PCI-7258

`INT1_DISABLE`	INT1 Disabled
`INT1_EXT_SIGNAL`	INT1 by External Signal

PCI-7260

`INT1_DISABLE`	INT1 Disabled
`INT1_COS`	INT1 by COS
`INT1_CH0`	INT1 by CH0

PCI-8554

`INT1_DISABLE`	INT1 Disabled
`INT1_COUT12`	INT1 by Counter #12
`INT1_EXT_SIGNAL`	INT1 by External Signal

PCI-7396/PCI-7348

`INT1_DISABLE`	NT1 Disabled
`INT1_COS`	INT1 by COS
`INT1_FP1C0`	INT1 by Falling edge of P1C0
`INT1_RP1C0_FP1C3`	INT1 by P1C0 Rising or P1C3 Falling
`INT1_EVENT_COUNTER`	INT1 by Event Counter down to zero

windowHandle	Handle to the window that you want to receive a Windows message when the specified INT event happens. If windowHandle is 0, no Windows messages will be sent.
message	The user-defined message. When the specified INT event happens, the PCIS-DASK sends this message back to you. The message can be of any value. In Windows, you can set a message to a value including any Windows predefined messages, such as WM_PAINT. However, to define your own message, you can use any value ranging from WM_USER (0x400) to 0x7fff. This range is reserved by Windows for user-defined messages.
callbackAddr	Address of the user callback function. The PCIS-DASK calls this function when the specified INT event occurs. If you do not want to use a callback function, set callbackAddr to 0.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DIO_INT2_EventMessage (Win32 Only)

Description

Controls the INT2 interrupt sources for a dual-interrupt system and notifies the user's application when an interrupt event occurs. The notification is performed through a user-specified callback function or the Windows PostMessage API.

Supported card(s)

7230, 7233, 7224, 7248, 7249, 7256, 7258, 7260, 7296, 7348, 7396, 7432, 7433, 7442, 7444, 8554

Syntax

Microsoft C/C++ and Borland C++

```
I16 DIO_INT2_EventMessage (U16 CardNumber, I16
     Int2Mode, HANDLE windowHandle, U32 message,
     void *callbackAddr())
```

Visual Basic

```
DIO_INT2_EventMessage (ByVal CardNumber As
     Integer, ByVal Int2Mode As Integer, ByVal
     windowHandle As Long, ByVal message As Long,
     ByVal callbackAddr As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Int2Mode	INT2 interrupt mode. Valid values:

PCI-7224

`INT2_DISABLE`	INT2 Disabled
`INT2_TIMER_COUNTER`	INT2 by Timer Counter down to zero

PCI-7248/cPCI-7248/cPCI-7249R/7296

`INT2_DISABLE`	INT2 Disabled
`INT2_FP2C0`	INT2_FP2C0: INT2 by Falling edge of P2C0
`INT2_RP2C0_FP2C3`	INT2 by P2C0 Rising or P2C3 Falling
`INT2_TIMER_COUNTER`	INT2 by Timer Counter down to zero

PCI-7230/cPCI-7230/7233/7432/7433/8554

`INT2_DISABLE`	INT2 Disabled
`INT2_EXT_SIGNAL`	INT2 by External Signal

PCI-7256

`INT2_DISABLE`	INT2 Disabled
`INT2_CH1`	INT2 by CH1

PCI-7258

`INT2_DISABLE`	INT2 Disabled
`INT2_EXT_SIGNAL`	INT2 by External Signal

PCI-7260

`INT2_DISABLE`	INT2 Disabled
`INT2_CH1`	INT2 by CH1

PCI-7348

`INT2_DISABLE`	INT2 Disabled
`INT2_COS`	INT2 by COS
`INT2_FP2C0`	INT2 by Falling edge of P2C0
`INT2_RP2C0_FP2C3`	INT2 by P2C0 Rising or P2C3 Falling
`INT2_TIMER_COUNTER`	INT2 by Timer Counter down to zero

PCI-7396

`INT2_DISABLE`	INT2 Disabled
`INT2_COS`	INT2 by COS
`INT2_FP2C0`	INT2 by Falling edge of P2C0
`INT2_RP2C0_FP2C3`	INT2 by P2C0 Rising or P2C3 Falling
`INT2_TIMER_COUNTER`	INT2 by Timer Counter down to zero

PCI-7442

`INT2_DISABLE`	INT2 Disabled
`INT2_WDT`	INT2 by Watchdog timer

PCI-7444

`INT2_DISABLE`	INT2 Disabled
`INT2_WDT`	INT2 by Watchdog timer

windowHandle Handle to the window that you want to receive a Windows message when the specified INT event happens. If windowHandle is 0, no Windows messages will be sent.

message The user-defined message. When the specified INT event happens, the PCIS-DASK sends this message back to you. The message can be of any value.

In Windows, you can set a message to a value including any Windows predefined messages, such

	as WM_PAINT. However, to define your own message, you can use any value ranging from WM_USER (0x400) to 0x7fff. This range is reserved by Windows for user-defined messages.
callbackAddr	Address of the user callback function. The PCIS-DASK calls this function when the specified INT event occurs. If you do not want to use a callback function, set callbackAddr to 0.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DIO_LineConfig

Description

Informs the PCIS-DASK library of the selected line and the direction (input or output) setting of the selected line.

Supported card(s)

7442, 7443, 7444

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DIO_LineConfig (U16 CardNumber, U16 Port, U16
     Line, U16 Direction)
```

Visual Basic

```
DIO_LineConfig (ByVal CardNumber As Integer,
     ByVal Port As Integer, ByVal Line As
     Integer, ByVal Direction As Integer) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Port	Selected port. Valid values:

PCI-7442	`P7442_TTL0, P7442_TTL1`
PCI-7443	`P7443_TTL0, P7443_TTL1`
PCI-7444	`P7444_TTL0, P7444_TTL1`

Line	Selected line. Valid values: 0...15.
Direction	Line direction of the PIO port. Valid values:

```
INPUT_LINE
OUTPUT_LINE
```

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

DIO_LinesConfig

Description

Informs the PCIS-DASK library of entire lines of the port selected and the direction (input or output) setting of the entire lines of the selected port.

Supported card(s)

7442, 7443, 7444

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DIO_LinesConfig (U16 CardNumber, U16 Port,
     U16 Linesdirmap)
```

Visual Basic

```
DIO_LinesConfig (ByVal CardNumber As Integer,
     ByVal Port As Integer, ByVal Linesdirmap As
     Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Port Selected port. Valid values:

PCI-7442	`P7442_TTL0, P7442_TTL1`
PCI-7443	`P7443_TTL0, P7443_TTL1`
PCI-7444	`P7444_TTL0, P7444_TTL1`

Linesdirmap Port direction of PIO port. Each bit of the value of Linesdirmap controls one line of the port selected. The value 1 of the bit value sets the corresponding line to output, and the value 0 of the bit value sets the corresponding line to input. The valid values for Linesdirmap are 0 to 4294967295 (0xFFFFFFFF).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

DIO_PortConfig

Description

Informs the PCIS-DASK library of the selected port and the direction (input or output) setting of the selected port.

Supported card(s)

7224, 7248, 7249, 7296, 7348, 7396, 7442, 7443, 7444

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DIO_PortConfig (U16 CardNumber, U16 Port, U16
     Direction)
```

Visual Basic

```
DIO_PortConfig (ByVal CardNumber As Integer,
     ByVal Port As Integer, ByVal Direction As
     Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Port Selected port. Valid values:

PCI-7224	`Channel_P1C, Channel_P1CL, Channel_P1CH`
PCI-7248/cPCI-7248	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL Channel_P1CH, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2CL, Channel_P2CH`
cPCI-7249R	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2CL, Channel_P2CH`

PCI-7296	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2CL, Channel_P2CH, Channel_P3A, Channel_P3B, Channel_P3C, Channel_P3CL, Channel_P3CH, Channel_P4A, Channel_P4B, Channel_P4C, Channel_P4CL, Channel_P4CH`
PCI-7348	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1, Channel_P1E, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2, Channel_P2E`
PCI-7396	`Channel_P1A Channel_P1B, Channel_P1C, Channel_P1, Channel_P1E, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2, Channel_P2E, Channel_P3A, Channel_P3B, Channel_P3C, Channel_P3, Channel_P3E, Channel_P4A, Channel_P4B, Channel_P4C, Channel_P4, Channel_P4E`
PCI-7442	`P7442_TTL0, P7442_TTL1`
PCI-7443	`P7443_TTL0, P7443_TTL1`
PCI-7444	`P7444_TTL0, P7444_TTL1`

NOTE:

The value **Channel_Pn** for argument **Port** is defined as all of the ports (Port A, B, and C) in channel n.

If the port argument of DIO_PortConfig is set to Channel_PnE, channel n will be configured as INPUT_PORT, (the argument Direction may be ignored) and the digital input of channel n is controlled by the external clock.

Direction	The port direction of PIO port. Valid values:
	`INPUT_PORT`
	`OUTPUT_PORT`

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

DIO_SetCOSInterrupt

Description

Enables or disables the COS (Change Of State) interrupt detection capability of the specified ports with 8-bit or 16-bit data width.

Supported card(s)

7348, 7396, 7256, 7260

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DIO_SetCOSInterrupt (U16 CardNumber, U16
     Channel_no, U16 ctlA, U16 ctlB, U16 ctlC)
```

Visual Basic

```
DIO_SetCOSInterrupt (ByVal CardNumber As Integer,
     ByVal Channel_no As Integer, ByVal ctlA As
     Integer, ByVal ctlB As Integer, ByVal ctlC
     As Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Channel_no Channel number where COS detection capability is to be enabled/disabled. Valid port numbers:

PCI-7348	
Port 1	`Channel_P1`
Port 2	`Channel_P2`
PCI-7396	
Port 1	`Channel_P1`
Port 2	`Channel_P2`
Port 3	`Channel_P3`
Port 4	`Channel_P4`
PCI-7256	`0`
PCI-7260	`0`

ctlA Control value for Port A of the channel defined by argument Channel_no or the control value for the port defined by Channel_no. Valid values:

PCI-7396/PCI-7348

`0`	Disabled
`1`	Enabled

PCI-7256/PCI-7260

Each bit of the value of ctrlA controls one DI channel. The '0' value of the bit value enable the COS function of the corresponding channel, and the '1' value of the bit value disable the COS function of the corresponding channel. The valid values for ctrlA: 0 through 65535

ctlB Control value for Port B of the channel defined by argument Channel_no. Valid values:

PCI-7396/PCI-7348	
`0`	Disabled
`1`	Enabled
PCI-7256/7260	Not needed

ctlC Control value for Port C of the channel defined by argument Channel_no. Valid values:

PCI-7396/PCI-7348	
`0`	Disabled
`1`	Enabled
PCI-7256/7260	Not needed

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DIO_SetCOSInterrupt32

Description

Enables or disables the COS (Change Of State) interrupt detection capability of the specified ports with 32-bit data width.

Supported card(s)

7442, 7443, 7452

Syntax

Microsoft C/C++ and Borland C++

```
I16 DIO_SetCOSInterrupt32 (U16 CardNumber, U8
     Port, U32 ctl, HANDLE *hEvent, BOOLEAN
     ManualReset)
```

Linux C/C++

```
I16 DIO_SetCOSInterrupt32(U16 CardNumber, U8
     Port, U32 ctl, void (*event_handler)(int))
```

Visual Basic

```
DIO_SetCOSInterrupt32 (ByVal CardNumber As
     Integer, ByVal Port As Byte, ByVal ctl As
     Long, hEvent As Long, ByVal ManualReset As
     Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Port	Channel number where COS detection capability is to be enabled/disabled. Valid port numbers:

PCI-7442	`0 to 1`
PCI-7443	`0 to 3`
PCI-7452	`0 to 3`

ctl

Control value for the port defined by argument Port. Valid values:

PCI-7442 **PCI-7443**	Each bit of the value of ctrl controls one DI channel. The '0' value of the bit value disable the COS function of the corresponding line, and the '1' value of the bit value enable the COS function of the corresponding line. The valid values for ctrl are from 0 to 4294967295 (0xFFFFFFFF)
PCI-7452	Each bit of the value of ctrl controls one DI channel. The '0' value of the bit value enable the COS function of the corresponding line, and the '1' value of the bit value disable the COS function of the corresponding line. The valid values for ctrl are 0 to 4294967295 (0xFFFFFFFF)

ManualReset (Win32 only)

Specifies whether the event is (1) manual-reset by function ResetEvent in user's application or (0) auto-reset by driver.

hEvent (Win32 only)

Returned COS interrupt event handle.

event_handler (Linux only)

Address of the user callback function. The PCIS-DASK calls this function when the specified COS event occurs. If you do not want to use a callback function, set callbackAddr to 0.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DIO_SetDualInterrupt

Description

Informs the PCIS-DASK library of the interrupt mode of two interrupt sources of a dual-interrupt system and returns dual interrupt events. If an interrupt is generated, the corresponding interrupt event are signaled. The application uses Win32 wait functions, such as WaitForSingleObject or WaitForMultipleObjects to check the interrupt event status.

Supported card(s)

7230, 7233, 7224, 7248, 7249, 7256, 7258, 7260, 7296, 7348, 7396, 7432, 7433, 7442, 7443, 7444, 7452, 8554

Syntax

Microsoft C/C++ and Borland C++

```
I16 DIO_SetDualInterrupt (U16 CardNumber, I16
     Int1Mode, I16 Int2Mode, HANDLE *hEvent)
```

Linux C/C++

```
I16 DIO_SetDualInterrupt(U16 CardNumber, I16
     Int1Mode, I16 Int2Mode, void
     (*event1_handler)(int), void
     (*event2_handler)(int))
```

Visual Basic

```
DIO_SetDualInterrupt (ByVal CardNumber As
     Integer, ByVal Int1Mode As Integer, ByVal
     Int2Mode As Integer, hEvent As Long) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Int1Mode	The interrupt mode of INT1. Valid values:

PCI-7224/PCI-7248/cPCI-7248/cPCI7249R//7296

`INT1_DISABLE`	INT1 Disabled
`INT1_FP1C0`	INT1 by Falling edge of P1C0
`INT1_RP1C0_FP1C3`	INT1 by P1C0 Rising or P1C3 Falling
`INT1_EVENT_COUNTER`	INT1 by Event Counter down to zero

PCI-7230/cPCI-7230/7233/7432/7433	
`INT1_DISABLE`	INT1 Disabled
`INT1_EXT_SIGNAL`	INT1 by External Signal
PCI-7256	
`INT1_DISABLE`	INT1 Disabled
`INT1_COS`	INT1 by COS
`INT1_CH0`	INT1 by CH0
PCI-7258	
`INT1_DISABLE`	INT1 Disabled
`INT1_EXT_SIGNAL`	INT1 by External Signal
PCI-7260	
`INT1_DISABLE`	INT1 Disabled
`INT1_COS`	INT1 by COS
`INT1_CH0`	INT1 by CH0
PCI-7442	
`INT1_DISABLE`	INT1 Disabled
`INT1_COS0`	INT1 by COS of Port 0
`INT1_COS1`	INT1 by COS of Port 1
PCI-7443	
`INT1_DISABLE`	INT1 Disabled
`INT1_COS0`	INT1 by COS of Port 0
`INT1_COS1`	INT1 by COS of Port 1
`INT1_COS2`	INT1 by COS of Port 2
`INT1_COS3`	INT1 by COS of Port 3
PCI-7444	Not available
PCI-7452	
`INT1_DISABLE`	INT1 Disabled
`INT1_COS0`	INT1 by COS of Port 0
`INT1_COS1`	INT1 by COS of Port 1
`INT1_COS2`	INT1 by COS of Port 2
`INT1_COS3`	INT1 by COS of Port 3
PCI-8554	
`INT1_DISABLE`	INT1 Disabled
`INT1_EXT_SIGNAL`	INT1 by External Signal
`INT1_COUT12`	INT1 by Counter #12

PCI-7348/PCI-7396

`INT1_DISABLE`	INT1 Disabled
`INT1_COS`	INT1 by COS
`INT1_FP1C0`	INT1 by Falling edge of P1C0
`INT1_RP1C0_FP1C3`	INT1 by P1C0 Rising or P1C3 Falling
`INT1_EVENT_COUNTER`	INT1 by Event Counter down to zero

Int2Mode Interrupt mode of INT2. Valid values:

PCI-7224/PCI-7248/cPCI-7248/cPCI-7249R/7296

`INT2_DISABLE`	INT2 Disabled
`INT2_FP2C0`	INT2 by Falling edge of P2C0
`INT2_RP2C0_FP2C3`	INT2 by P2C0 Rising or P2C3 Falling
`INT2_TIMER_COUNTER`	INT2 by Timer Counter down to zero

PCI-7224

`INT2_DISABLE`	INT2 Disabled
`INT2_TIMER_COUNTER`	INT2 by Timer Counter down to zero

PCI-7230/cPCI-7230/7233/7432/7433/8554

`INT2_DISABLE`	INT2 Disabled
`INT2_EXT_SIGNAL`	INT2 by External Signal

PCI-7256

`INT2_DISABLE`	INT2 Disabled
`INT2_CH1`	NT2 by CH1

PCI-7258

`INT2_DISABLE`	INT2 Disabled
`INT2_EXT_SIGNAL`	INT2 by External Signal

PCI-7260

`INT2_DISABLE`	INT2 Disabled
`INT2_CH1`	INT2 by CH1

PCI-7348/PCI-7396

`INT2_DISABLE`	INT2 Disabled
`INT2_COS`	INT2 by COS
`INT2_FP2C0`	INT2 by Falling edge of P2C0
`INT2_RP2C0_FP2C3`	INT2 by P2C0 Rising or P2C3 Falling
`INT2_TIMER_COUNTER`	INT2 by Timer Counter down to zero

PCI-7442	
`INT2_DISABLE`	INT2 Disabled
`INT2_WDT`	INT2 by Watchdog timer
PCI-7443	Not available
PCI-7444	
`INT2_DISABLE`	INT2 Disabled
`INT2_WDT`	INT2 by Watchdog timer
PCI-7452	Not available

hEvent (Win32 only)

Returned dual-interrupt event handles. The status of a dual-interrupt event indicates that an interrupt is generated or not for cards comprising dual-interrupt system (PCI-7230/cPCI-7230, PCI-7233, PCI-7224/PCI-7248/cPCI-7248, cPCI-7249R, PCI-7256, PCI-7258, PCI-7296, PCI-7348/PCI-7396, PCI-7432/cPCI-7432/cPCI7432R, PCI-7442, PCI-7443, PCI-7444, PCI-7452, and PCI-7433/cPCI-7433/cPCI-7433R).

event1_handler (Linux only)

Address of the user callback function. The PCIS-DASK calls this function when the specified INT1event occurs. If you do not want to use a call-back function, set callbackAddr to 0.

event2_handler (Linux only)

Address of the user callback function. The PCIS-DASK calls this function when the specified INT2 event occurs. If you do not want to use a callback function, set callbackAddr to

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DIO_T2_EventMessage (Win32 Only)

Description

Controls the Timer2 interrupt and notifies the user's application when an interrupt event occurs. The notification is performed through a user-specified callback function or the Windows Post-Message API.

Supported card(s)

7300A

Syntax

Microsoft C/C++ and Borland C++

```
I16 DIO_T2_EventMessage (U16 CardNumber, I16
     T2En, HANDLE windowHandle, U32 message, void
     *callbackAddr())
```

Visual Basic

```
DIO_T2_EventMessage (ByVal CardNumber As Integer,
     ByVal T2En As Integer, ByVal windowHandle As
     Long, ByVal message As Long, ByVal
     callbackAddr As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
T2En	The control value for Timer2 interrupt. Valid values:
	`0` Disabled
	`1` Enabled
windowHandle	Handle to the window that you want to receive a Windows message when the specified Timer2 event occurs. If windowHandle is 0, no Windows messages will be sent.

message	User-defined message. When the specified Timer2 event occurs, the PCIS-DASK sends this message to you. The message can be of any value.
	In Windows, you can set a message to a value including any Windows predefined messages, such as WM_PAINT. However, to define your own message, you can use any value ranging from WM_USER (0x400) to 0x7fff. This range is reserved by Windows for user-defined messages.
callbackAddr	Address of the user callback function. The PCIS-DASK calls this function when the specified Timer2 event occurs. If you do not want to use a callback function, set callbackAddr to 0.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DO_7200_Config

Description

Informs the PCIS-DASK library of the trigger source and output mode selected for PCI-7200/cPCI-7200 with card ID. You must call this function before calling function to perform continuous digital output operation.

Supported card(s)

7200

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_7200_Config (U16 CardNumber, U16
     TrigSource, U16 OutReqEn, U16 OutTrigSig)
```

Visual Basic

```
DO_7200_Config (ByVal CardNumber As Integer,
     ByVal TrigSource As Integer, ByVal OutReqEn
     As Integer, ByVal OutTrigSig As Integer) As
     Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

TrigSource Trigger source for continuous digital input. Valid values:

`TRIG_INT_PACER`	Onboard programmable pacer
`TRIG_HANDSHAKE`	Handshaking

OutReqEn Output REQ Enable

`OREQ_ENABLE`	Output REQ is enabled, an O_REQ strobe is generated after output data is strobed.
`OREQ_DISABLE`	Output REQ is disable.

OutTrigSig Output Trigger Signal

`OTRIG_HIGH`	O_TRIG signal goes high
`OTRIG_LOW`	O_TRIG signal goes low

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DO_7300A_Config

Description

Informs the PCIS-DASK library of the trigger source, port width, etc. selected for PCI7300A Rev.A/cPCI7300A Rev.A card with card ID CardNumber. You must call this function before calling function to perform continuous digital output operation.

Supported card(s)

7300A Rev.A

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_7300A_Config (U16 CardNumber, U16
     PortWidth, U16 TrigSource, U16 WaitStatus,
     U16 Terminator, U16 O_REQ_Pol)
```

Visual Basic

```
DO_7300A_Config (ByVal CardNumber As Integer,
     ByVal PortWidth As Integer, ByVal TrigSource
     As Integer, ByVal WaitStatus As Integer,
     ByVal Terminator As Integer, ByVal O_REQ_Pol
     As Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

PortWidth Width of digital output port (PORT B). Valid values: 0, 8, 16, or 32.

TrigSource Trigger mode for continuous digital output. Valid values:

`TRIG_INT_PACER`	Onboard programmable pacer timer1
`TRIG_CLK_10MHz`	10 MHz clock
`TRIG_CLK_20MHz`	20 MHz clock
`TRIG_HANDSHAKE`	Handshaking mode

WaitStatus DO Wait Status. Valid values:

`P7300_WAIT_NO`	Digital output starts immediately.
`P7300_WAIT_TRG`	Digital output waits rising or falling edge of O_TRG to start.

`P7300_WAIT_FIFO`	Delay output data until FIFO is not almost empty.
`P7300_WAIT_BOTH`	Delay output data until O_TRG active and FIFO is not almost empty.

Terminator PortB Terminator On/Off. Valid values:

`P7300_TERM_ON`	Terminator on.
`P7300_TERM_OFF`	Terminator off.

O_REQ_Pol O_REQ Polarity. This function is not implemented on PCI-7300A Rev.A or cPCI-7300A Rev.A card. You may ignore this argument.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DO_7300B_Config

Description

Informs the PCIS-DASK library of the selected trigger source, port width, etc. for PCI-7300A Rev. B or cPCI-7300A Rev. B card with card ID. You must call this function before calling function to perform continuous digital output operation.

Supported card(s)

7300A Rev.B

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_7300B_Config (U16 CardNumber, U16
     PortWidth, U16 TrigSource, U16 WaitStatus,
     U16 Terminator, U16 O_Cntrl_Pol, U32
     FifoThreshold)
```

Visual Basic

```
DO_7300B_Config (ByVal CardNumber As Integer,
     ByVal PortWidth As Integer, ByVal TrigSource
     As Integer, ByVal WaitStatus As Integer,
     ByVal Terminator As Integer, ByVal
     O_Cntrl_Pol As Integer, ByVal FifoThreshold
     As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
PortWidth	Width of digital output port (PORT B). Valid values: 0, 8, 16, or 32.
TrigSource	Trigger mode for continuous digital output. Valid values:

`TRIG_INT_PACER`	Onboard programmable pacer timer1
`TRIG_CLK_10MHz`	10 MHz clock
`TRIG_CLK_20MHz`	20 MHz clock
`TRIG_HANDSHAKE`	Handshaking mode
`TRIG_DO_CLK_TIMER_ACK`	Burst handshaking mode by using timer1 output as output clock.

`TRIG_DO_CLK_10M_ACK`	Burst handshaking mode by using 10 MHz clock as output clock.
`TRIG_DO_CLK_20M_ACK`	Burst handshaking mode by using 20 MHz clock as output clock.

WaitStatus DO Wait Status. Valid values are:

`P7300_WAIT_NO`	Digital output starts immediately.
`P7300_WAIT_TRG`	Digital output waits rising or falling edge of O_TRG to start.
`P7300_WAIT_FIFO`	Delay output data until FIFO is not almost empty.
`P7300_WAIT_BOTH`	Delay output data until O_TRG active and FIFO is not almost empty.

Terminator PortB Terminator On/Off, the valid values are:

`P7300_TERM_ON`	Terminator on.
`P7300_TERM_OFF`	Terminator off.

O_Cntrl_Pol Polarity configuration. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are three groups of constants:

DOREQ

`P7300_DOREQ_POS`	DOREQ signal is rising edge active.
`P7300_DOREQ_NEG`	DOREQ signal is falling edge active.

DOACK

`P7300_DOACK_POS`	DOACK signal is rising edge active.
`P7300_DOACK_NEG`	DOACK signal is falling edge active.

DOTRIG

`P7300_DOTRIG_POS`	DOTRIG signal is rising edge active.
`P7300_DOTRIG_NEG`	DOTRIG signal is falling edge active.

FifoThreshold Programmable almost empty threshold of both PORTB FIFO and PORTA FIFO, if output port width is 32.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DO_9222_Config

Description

Inform the PCIS-DASK library of the trigger source, trigger mode, and trigger properties selected for the PCI-9222 with card ID CardNumber. You must call this function before calling function to perform continuous digital output operation.

Supported card(s)

9222

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DO_9222_Config (U16 CardNumber, U16
     ConfigCtrl, U16 TrigCtrl, U32 ReTrgCnt, U32
     DLY1Cnt, U32 DLY2Cnt, BOOLEAN AutoResetBuf)
```

Visual Basic

```
DO_9222_Config (ByVal CardNumber As Integer,
     ByVal ConfigCtrl As Integer, ByVal TrigCtrl
     As Integer, ByVal ReTrgCnt As Long, ByVal
     DLY1Cnt As Long, ByVal DLY2Cnt As Long,
     ByVal AutoResetBuf As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ConfigCtrl	The setting for DO mode control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There is one group of constants:

Conversion Source Selection

```
P922x_DO_CONVSRC_INT
P922x_DO_CONVSRC_GPI0
P922x_DO_CONVSRC_GPI1
P922x_DO_CONVSRC_GPI2
P922x_DO_CONVSRC_GPI3
P922x_DO_CONVSRC_GPI4
P922x_DO_CONVSRC_GPI5
P922x_DO_CONVSRC_GPI6
P922x_DO_CONVSRC_GPI7
```

	`P922x_DO_CONVSRC_ADCONV`
	`P922x_DO_CONVSRC_DACONV`

TrigCtrl	The setting for DO Trigger control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are four groups of constants:
	Trigger Mode Selection
	`P922x_DO_TRGMOD_POST`
	`P922x_DO_TRGMOD_DELAY`
	Trigger Source Selection
	`P922x_DO_TRGSRC_SOFT`
	`P922x_DO_TRGSRC_GPI0`
	`P922x_DO_TRGSRC_GPI1`
	`P922x_DO_TRGSRC_GPI2`
	`P922x_DO_TRGSRC_GPI3`
	`P922x_DO_TRGSRC_GPI4`
	`P922x_DO_TRGSRC_GPI5`
	`P922x_DO_TRGSRC_GPI6`
	`P922x_DO_TRGSRC_GPI7`
	Trigger Polarity
	`P922x_DO_TrgPositive`
	`P922x_DO_TrgNegative`
	Re-Trigger Mode Enable
	`P922x_DO_EnReTigger`

ReTrgCnt	The accepted trigger times in an acquisition. This argument is valid only for re-trigger mode. The valid range of ReTrgCnt is 0 to 4294967295. If the value of ReTrgCnt is 0, the DO operation is triggered infinitely.
DLY1Cnt	DLY1 counter value or the delay time to start waveform generation after the trigger signal. This argument is valid only for delay trigger mode. The range of valid value is 0 to 4294967295.
DLY2Cnt	DLY2 counter value or the delay between two consecutive waveform generations. This argument is valid only for waveform repeat, so it is not used since DO waveform repeat dose not support for PCI-9222.
AutoResetBuf	
	FALSE

The DO buffers set by the DO_ContBufferSetup function are retained. You must call the DO_ContBufferReset function to reset the buffer.

TRUE

The DO buffers set by the DO_ContBufferSetup function are reset automatically by driver when the DO operation is completed.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorConfigIoctl
```

DO_9223_Config

Description

Inform the PCIS-DASK library of the trigger source, trigger mode, and trigger properties selected for the PCI-9223 with card ID CardNumber. You must call this function before calling function to perform continuous digital output operation.

Supported card(s)

9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DO_9223_Config (U16 CardNumber, U16
     ConfigCtrl, U16 TrigCtrl, U32 ReTrgCnt, U32
     DLY1Cnt, U32 DLY2Cnt, BOOLEAN AutoResetBuf)
```

Visual Basic

```
DO_9223_Config (ByVal CardNumber As Integer,
     ByVal ConfigCtrl As Integer, ByVal TrigCtrl
     As Integer, ByVal ReTrgCnt As Long, ByVal
     DLY1Cnt As Long, ByVal DLY2Cnt As Long,
     ByVal AutoResetBuf As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ConfigCtrl	The setting for DO mode control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There is one group of constants:

Conversion Source Selection

```
P922x_DO_CONVSRC_INT
P922x_DO_CONVSRC_GPI0
P922x_DO_CONVSRC_GPI1
P922x_DO_CONVSRC_GPI2
P922x_DO_CONVSRC_GPI3
P922x_DO_CONVSRC_GPI4
P922x_DO_CONVSRC_GPI5
P922x_DO_CONVSRC_GPI6
P922x_DO_CONVSRC_GPI7
```

	`P922x_DO_CONVSRC_ADCONV` `P922x_DO_CONVSRC_DACONV`

TrigCtrl	The setting for DO Trigger control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are four groups of constants:
	Trigger Mode Selection
	`P922x_DO_TRGMOD_POST`
	`P922x_DO_TRGMOD_DELAY`
	Trigger Source Selection
	`P922x_DO_TRGSRC_SOFT`
	`P922x_DO_TRGSRC_GPI0`
	`P922x_DO_TRGSRC_GPI1`
	`P922x_DO_TRGSRC_GPI2`
	`P922x_DO_TRGSRC_GPI3`
	`P922x_DO_TRGSRC_GPI4`
	`P922x_DO_TRGSRC_GPI5`
	`P922x_DO_TRGSRC_GPI6`
	`P922x_DO_TRGSRC_GPI7`
	Trigger Polarity
	`P922x_DO_TrgPositive`
	`P922x_DO_TrgNegative`
	Re-Trigger Mode Enable
	`P922x_DO_EnReTigger`

ReTrgCnt	The accepted trigger times in an acquisition. This argument is valid only for re-trigger mode. The valid range of ReTrgCnt is 0 to 4294967295. If the value of ReTrgCnt is 0, the DO operation is triggered infinitely.
DLY1Cnt	DLY1 counter value or the delay time to start waveform generation after the trigger signal. This argument is valid only for delay trigger mode. The range of valid value is 0 to 4294967295.
DLY2Cnt	DLY2 counter value or the delay between two consecutive waveform generations. This argument is valid only for waveform repeat, so it is not used since DO waveform repeat dose not support for PCI-9223.
AutoResetBuf	
	FALSE

The DO buffers set by the DO_ContBufferSetup function are retained. You must call the DO_ContBufferReset function to reset the buffer.

TRUE

The DO buffers set by the DO_ContBufferSetup function are reset automatically by driver when the DO operation is completed.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorConfigIoctl
```

DO_AsyncCheck

Description

Checks the current status of the asynchronous digital output operation.

Supported card(s)

7200, 7300A, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_AsyncCheck (U16 CardNumber, BOOLEAN
     *Stopped, U32 *AccessCnt)
```

Visual Basic

```
DO_AsyncCheck (ByVal CardNumber As Integer,
     Stopped As Byte, AccessCnt As Long) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing asynchronous operation.
Stopped	Tells whether the asynchronous digital output operation is completed. If Stopped = TRUE, the digital output operation has stopped, either because the number of digital output indicated in the call that initiated the asynchronous digital output operation has completed or an error has occurred. If Stopped = FALSE, the operation is not yet complete. Constants TRUE and FALSE are defined in DASK.H.
AccessCnt	Number of digital output data that has been written at the time the call to DO_AsyncCheck().

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DO_AsyncClear

Description

Stops the asynchronous digital output operation.

Supported card(s)

7200, 7300A, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_AsyncClear (U16 CardNumber, U32
     *AccessCnt)
```

Visual Basic

```
DO_AsyncClear (ByVal CardNumber As Integer,
     AccessCnt As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing asynchronous operation.
AccessCnt	Number of digital output data that has been transferred at the time the call to DO_AsyncClear().

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered,
ErrorFuncNotSupport
```

DO_AsyncMultiBufferNextReady

Description

Checks whether the next buffer is ready for new data during an asynchronous multi-buffered digital output operation. The returned BufferId is the index of the most recently available (newest available) buffer.

Supported card(s)

7300A

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_AsyncMultiBufferNextReady (U16 CardNumber,
     BOOLEAN *NextReady, U16 *BufferId)
```

Visual Basic

```
DO_AsyncMultiBufferNextReady (ByVal CardNumber As
     Integer, NextReady As Byte, BufferId As
     Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing asynchronous multi-buffered operation.
NextReady	Tells whether the next buffer is ready for new data.
BufferId	Returns the index of the ready buffer.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DO_ContBufferReset

Description

This function resets all the buffers set by function DO_ContBufferSetup for continuous digital output. The function has to be called if the data buffers won't be used.

Supported card(s)

9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DO_ContBufferReset (U16 CardNumber)
```

Visual Basic

```
DO_ContBufferReset (ByVal CardNumber As Integer)
     As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorContIoNotAllowed
```

DO_ContBufferSetup

Description

This function set up the buffer for continuous digital output operation. The function has to be called repeatedly to setup all of the data buffers. (For PCI-9222/9223, the maximum number of buffers is 1.)

Supported card(s)

9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DO_ContBufferSetup (U16 CardNumber, VOID
     *Buffer, U32 WriteCount, U16 *BufferId)
```

Visual Basic

```
DO_ContBufferSetup (ByVal CardNumber As Integer,
     Buffer As Any, ByVal WriteCount As Long,
     BufferId As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Buffer	The starting address of the memory to contain the output data.
WriteCount	The size (in samples) of the buffer and its value must be even.
BufferId	Returns the index of the buffer currently set up.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
```

DO_ContMultiBufferSetup

Description

Sets up the buffer for multi-buffered digital output. The function has to be called repeatedly to set up all of the data buffers (maximum eight buffers).

Supported card(s)

7300A

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_ContMultiBufferSetup (U16 CardNumber, void
     *pwBuffer, U32 dwWriteCount, U16 *BufferId)
```

Visual Basic

```
DO_ContMultiBufferSetup (ByVal CardNumber As
     Integer, Buffer As Any, ByVal WriteCount As
     Long, BufferId As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Buffer	Starting address of the memory to contain the output data.
WriteCount	Size (in samples) of the buffer and its value. Must be even in value.
BufferId	Returns the index of the buffer currently being set up.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
```

DO_ContMultiBufferStart

Description

Starts multi-buffered continuous digital output on the specified digital output port at a rate closest to the specified rate.

Supported card(s)

7300A Rev.B

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_ContMultiBufferStart (U16 CardNumber, U16
     Port, F64 SampleRate)
```

Visual Basic

```
DO_ContMultiBufferStart (ByVal CardNumber As
     Integer, ByVal Port As Integer, ByVal
     SampleRate As Double) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Port	Digital output port number. For PCI-7300A or cPCI-7300A, this argument must be set to 0.
SampleRate	Sampling rate you want for digital output in hertz (samples per second). Your maximum rate depends on the card type and your computer system. This argument is only useful if the DO trigger mode was set as internal programmable pacer (TRIG_INT_PACER) by calling DO_7300B_Config().

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorContIoNotAllowed
```

DO_ContStatus

Description

While performing continuous DO conversions, this function gets the DO status. Refer to the card's user manual for the DO status that the device might meet.

Supported card(s)

7200, 7300A, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_ContStatus (U16 CardNumber, U16 *Status)
```

Visual Basic

```
DO_ContStatus (ByVal CardNumber As Integer,
     Status Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Status	Continuous DO status returned. Description of the Status parameter for various card types:

PCI7200

`bit 0`	1 = DI FIFO is full (overrun).
`bit 1`	1 = DO FIFO is Empty (underrun)
`bit 2 ~ 15`	Not in use

PCI7300A_RevA

`bit 0`	1 = DO FIFO is empty during data output and some output data were written twice. Write 1 to clear this bit.
`bit 1`	1 = DO FIFO is full
`bit 2`	1 = DO FIFO is empty
`bit 3 ~ 15`	Not in use

PCI7300A_RevB

`bit 0`	1 = DO FIFO is empty during data output and some output data were written twice. Write 1 to clear this bit.
`bit 1`	1 = DO FIFO is full
`bit 2`	1 = DO FIFO is empty
`bit 3 ~ 15`	Not in use

PCI-9222 and **PCI-9223**

`bit 0`	1 = FIFO is empty.
`bit 1`	1 = FIFO is almost empty.
`bit 2`	1 = FIFO is almost full.
`bit 3`	1 = FIFO is full.
`bit 4-15`	Not used.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
```

DO_ContWritePort

Description

Performs continuous digital output on the specified digital output port at a rate closest to the specified rate.

Supported card(s)

7200, 7300A, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_ContWritePort (U16 CardNumber, U16 Port,
     void *Buffer, U32 WriteCount, U16
     Iterations, F32 SampleRate, U16 SyncMode)
```

Visual Basic

```
DO_ContWritePort (ByVal CardNumber As Integer,
     ByVal Port As Integer, Buffer As Any, ByVal
     WriteCount As Long, ByVal Iterations As
     Integer, ByVal SampleRate As Single, ByVal
     SyncMode As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Port	Digital output port number. For PCI-7200, cPCI-7200, PCI-7300A, cPCI-7300A, PCI-9222, PCI-9223, this argument must be set to 0.
Buffer	Starting address of the memory containing the output data. This memory must have been allocated for enough space to store output data.
WriteCount	Number of output operation to be performed.
Iterations	Number of times the data in the Buffer is to be output to the Port. A value of 0 means that digital output operation proceeds indefinitely. If the digital output operation is performed synchronously, this argument must be set to 1. For PCI-9222/9223, this argument is not used as theDO repeat mode does not support for PCI-9222/9223.

SampleRate	Sampling rate you want for digital output in hertz (samples per second). Your maximum rate depends on the card type and your computer system. **PCI-7200, PCI-7300:** This argument is only useful if the DO trigger mode is set as internal programmable. Pacer (TRIG_INT_PACER and TRIG_DO_CLK_TIMER_ACK) by calling DO_7200_Config() or DO_7300_Config(). For other settings, set this argument as CLKSRC_EXT_SampRate. **PCI-9222, PCI-9223:** This argument is only useful if the DO conversion source is set as internal conversion Source (P922x_DO_CONVSRC_INT) by calling DO_9222_Config() or DO_9223_Config(). For other settings, this argument is ignored. The maximum sample rate is 2000000 (2 MHz).
SyncMode	Tells whether the operation is performed synchronously or asynchronously. Valid values:

`SYNCH_OP`	Synchronous digital input, that is, the function does not return until the digital input operation is completed.
`ASYNCH_OP`	Asynchronous digital input operations

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
```

DO_EventCallBack (Win32 Only)

Description

Controls and notifies the user's application when a specified DAQ event occurs. The notification is performed through a user-specified callback function. The event message is removed automatically after calling DO_Async_Clear. The event message can also be manually removed by setting the Mode parameter to 0.

Supported card(s)

7200, 7300A, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_EventCallBack (U16 CardNumber, I16 mode,
     I16 EventType, U32 callbackAddr)
```

Visual Basic

```
DO_EventCallBack (ByVal CardNumber As Integer,
     ByVal mode As Integer, ByVal EventType As
     Integer, ByVal callbackAddr As Long) As
     Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

mode Add or remove the event message. Valid values:

`0`	Remove
`1`	Add

EventType Event criteria. Valid values:

`DOEnd`	Notification that the asynchronous digital output operation has been completed.
`DBEvent`	Notification that the next half buffer of data in circular buffer is ready for transfer (this value is not valid for PCI-7300A, PCI-9222, PCI-9223).
`TrigEvent`	TrigEventNotifies that the data associated to the next trigger signal is available (this value is not valid for PCI-7200 and PCI-7300A).

callbackAddr	Address of the user callback function. The PCIS-DASK calls this function when the specified event occurs. If you want to remove the event message, set callbackAddr to 0.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DO_GetView

Description

Returns the mapped buffer address of the memory allocated in the driver for continuous DO operation at system startup time. The size of the allocated memory can be acquired by using the function DO_InitialMemoryAllocated.

Supported card(s)

7200

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_GetView(U16 CardNumber, U32 *pView)
```

Visual Basic

```
DO_GetView (ByVal CardNumber As Integer, pView As
     Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
pView	Mapped buffer address of the memory allocated in the driver during system startup.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
```

DO_InitialMemoryAllocated

Description

Returns the available memory size for continuous digital output in the device driver of the card. The continuous digital output transfer size may not exceed this size.

Supported card(s)

7200, 7300A, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_InitialMemoryAllocated (U16 CardNumber,
     U32 *MemSize)
```

Visual Basic

```
DO_InitialMemoryAllocated (ByVal CardNumber As
     Integer, MemSize As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
MemSize	Available memory size in the device driver of the card. The unit is KB (1024 bytes).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
```

DO_PGStart

Description

Performs pattern generation for digital output with the data stored in Buffer at a rate closest to the specified rate.

Supported card(s)

7300A

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_PGStart (U16 CardNumber, void *Buffer, U32
     WriteCount, F64 SampleRate)
```

Visual Basic

```
DO_PGStart (ByVal CardNumber As Integer, Buffer
     As Any, ByVal WriteCount As Long, ByVal
     SampleRate As Double) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Buffer	Starting address of the memory containing the output data of pattern generation. This memory must be allocated with enough space to store output data.
WriteCount	Number of pattern generation output samples.
SampleRate	Sampling rate you want for digital output in hertz (samples per second). The maximum rate depends on the card type and your computer system. This argument is only useful if the DO trigger mode was set as internal programmable pacer (TRIG_INT_PACER) by calling DO_7300_Config().

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorTransferCountTooLarge
```

DO_PGStop

Description

Stops the pattern generation for digital output operation.

Supported card(s)

7300A

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_PGStop (U16 CardNumber)
```

Visual Basic

```
DO_PGStop (ByVal CardNumber As Integer) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DO_ReadLine

Description

Reads back the digital logic state of the specified digital output line of the specified port.

Supported card(s)

6202, 6208V/16V/08A, 6308V/08A, 7200, 7230, 7234, 7224, 7248, c7249R, 7250/51, 7252, 7256, 7258, 7260, 7296, 7300A, 7348, 7396, 7432, 7433, 7434, 7442, 7444, 7452, 8554, 9111, 9112, 9114, 9116, 9118, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_ReadLine (U16 CardNumber, U16 Port, U16
     Line, U16 *State)
```

Visual Basic

```
DO_ReadLine (ByVal CardNumber As Integer, ByVal
     Port As Integer, ByVal Line As Integer,
     State As Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Port Digital output port number. Valid values:

PCI-6202	`P6202_ISO0` `P6202_TTL0`
PCI-6208V/16V/08A	`0`
PCI-6308V/08A	`0`
PCI-7200	`0`
cPCI-7200	`0, 1 (auxiliary digital output port)`
PCI-7230/cPCI-7230	`0`
PCI-7234	`0`
PCI-7250/51	`0 to 3`
cPCI-7252	`0`
PCI-7256	`0`
PCI-7258	`0, 1`

PCI-7260	`0`
PCI-7300A/cPCI-7300A	`1 (auxiliary digital output port)`
PCI-7432/cPCI-7432	`0`
cPCI-7432R	`0, P7432R_DO_LED`
cPCI-7433R	`P7433R_DO_LED`
PCI-7434/cPCI-7434	`PORT_DO_LOW, PORT_DO_HIGH`
PCI-7434R	`PORT_DO_LOW, PORT_DO_HIGH, P7434R_DO_LED`
PCI-7442	`P7442_CH0, P7442_CH1`
PCI-7444	`P7444_CH0, P7444_CH1, P7444_CH2, P7444_CH3`
PCI-7452	`0 to 3`
PCI-8554	`0`
PCI-9111	`P9111_CHANNEL_DO, P9111_CHANNEL_EDO`
PCI-9112/cPCI-9112	`0`
cPCI-9116	`0`
PCI-9118	`0`
PCI-9114	`0`
PCI-9221	`0`
PCI-9222	`0`
PCI-9223	`0`
PCI-9524	`0`
PCI-7224/48/96/ cPCI-7248, cPCI-7249R, PCI-7348/96	`Refer to the DI_ReadLine function.`

Line Digital line to be accessed. Valid values:

PCI-6202	`0 to 15 (for P6202_ISO0)`
	`0 to 7 (for P6202_TTL0)`

PCI-6208V/16V/08A	`0 to 3`
PCI-6308V/08A	`0 to 3`
PCI-7200/cPCI-7200	`0 to 31 (for port 0) 0 through 3 (auxiliary output port of cPCI-7200)`
PCI-7230	`0 to 15`

PCI-7234	`0 to 31`
PCI-7250/51	`0 to 7`
cPCI-7252	`0 to 7`
PCI-7256	`0 to 15`
PCI-7258	`0 to 15`
PCI-7260	`0 to 7`
PCI-7300A/cPCI-7300A	`0 to 3`
PCI-7432/7433/7434	`0 to 31`
PCI-7442	`0 to 31`
PCI-7444	`0 to 31`
PCI-7452	`0 to 31`
PCI-8554	`0 to 7`
PCI-9111	`0 to 15`
PCI-9112	`0 to 15`
PCI-9114	`0 to 15`
cPCI-9116	`0 to 7`
PCI-9118DG/HG/HR	`0 to 3`
PCI-9221	`0 to 3`
PCI-9222	`0 to 15`
PCI-9223	`0 to 15`
PCI-9524	`0 to 7`
PCI-7224/48/96/ cPCI-7248, cPCI-7249R, PCI-7348/ 96	`Refer to the function DI_ReadLine section.`

State Returns the digital logic state, 0 or 1, of the specified line.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

DO_ReadPort

Description

Reads back the output digital data from the specified digital output port.

Supported card(s)

6202, 6208V/16V/08A, 6308V/08A, 7200, 7230, 7234, 7224, 7248, c7249R, 7250/51, 7252, 7256, 7258, 7260, 7296, 7300A, 7348, 7396, 7432, 7433, 7434, 7442, 7444, 7452, 8554, 9111, 9112, 9114, 9116, 9118, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_ReadPort (U16 CardNumber, U16 Port, U32
     *Value)
```

Visual Basic

```
DO_ReadPort (ByVal CardNumber As Integer, ByVal
     Port As Integer, Value As Long) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Port Digital output port number. Valid values:

PCI-6202	`P6202_ISO0` `P6202_TTL0`
PCI-6208V/16V/08A	`0`
PCI-6308V/08A	`0`
PCI-7200	`0`
cPCI-7200	`0, 1 (auxiliary digital output port)`
PCI-7230/cPCI-7230	`0`
PCI-7234	`0`
PCI-7250/51	`0 to 3`
cPCI-7252	`0`
PCI-7256	`0`
PCI-7258	`0, 1`
PCI-7260	`0`

PCI-7300A/cPCI-7300A	`1 (auxiliary digital output port)`
PCI-7432/cPCI-7432	`0`
cPCI-7432R	`0, P7432R_DO_LED`
cPCI-7433R	`P7433R_DO_LED`
PCI-7434/cPCI-7434	`PORT_DO_LOW, PORT_DO_HIGH`
cPCI-7434R	`PORT_DO_LOW, PORT_DO_HIGH, P7434R_DO_LED`
PCI-7442	`P7442_CH0, P7442_CH1`
PCI-7444	`P7444_CH0, P7444_CH1, P7444_CH2, P7444_CH3`
PCI-7452	`0 to 3`
PCI-8554	`0`
PCI-9111	`P9111_CHANNEL_DO, P9111_CHANNEL_EDO`
PCI-9112/cPCI-9112	`0`
cPCI-9116	`0`
PCI-9118	`0`
PCI-9114	`0`
PCI-9221	`0`
PCI-9222	`0`
PCI-9223	`0`
PCI-9524	`0`
PCI-7224/48/96/ cPCI-7248, cPCI-7249R, PCI-7348/96	`Refer to the function DI_ReadPort section.`

Value Returns the digital data read from the specified output port.

PCI-6202	`16-bit data (for P6202_ISO0)`
	`8-bit data (for P6202_TTL0)`

PCI-6208V/16V/08A	`4-bit data`
PCI-6308V/08A	`4-bit data`

PCI-7200/cPCI-7200	`32-bit data (for port 0)`
	`4-bit data (for auxiliary output port of cPCI-7200)`

PCI-7230/cPCI-7230	`16-bit data`

PCI-7234	32-bit data
PCI-7224/PCI-7248/ cPCI-7248	8-bit data
cPCI-7249R	8-bit data
PCI-7250/51	8-bit data
cPCI-7252	8-bit data
PCI-7256	16-bit data
PCI-7258	16-bit data
PCI-7260	8-bit data
PCI-7296	8-bit data
PCI-7300A/cPCI-7300A	4-bit data
PCI-7348/PCI-7396	24-bit data (for Channel_PnT, where n is the channel number) or 8-bit data (for Channel_PnA, Channel_PnB, Channel_PnC, where n is the channel number)
PCI-7432/cPCI-7432/ cPCI-7432R	32-bit data
cPCI-7433R	32-bit data
PCI-7434/cPCI-7434/ cPCI-7434R	32-bit data
PCI-7442	32-bit data
PCI-7444	32-bit data
PCI-7452	32-bit data
PCI-8554	8-bit data
PCI-9111	16-bit data (for P9111_CHANNEL_DO) or 4-bit data (for P9111_CHANNEL_EDO)
PCI-9112/cPCI-9112	16-bit data
PCI-9114	16-bit data
cPCI-9116	8-bit data
PCI-9118	4-bit data
PCI-9221	4-bit data
PCI-9222	16-bit data

PCI-9223	`16-bit data`
PCI-9524	`8-bit data`

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

DO_SetTimeOut

Description

Sets Timeout period for Sync. mode continuous DO. While the function is called, the Sync. mode continuous DO acquisition is stopped even when it is not completed.

Supported card(s)

9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DO_SetTimeOut (U16 CardNumber, U32 TimeOut)
```

Visual Basic

```
DO_SetTimeOut (ByVal CardNumber As Integer, ByVal
     TimeOut As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
TimeOut	Timeout period (ms).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DO_SimuWritePort

Description

Writes the output digital data simultaneously to the specified digital output port.

Supported card(s)

7442, 7444

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 DO_SimuWritePort (U16 CardNumber, U16
     NumChans, U32 *Buffer)
```

Visual Basic

```
DO_SimuWritePort (ByVal CardNumber As Integer,
     ByVal NumChans As Integer, Buffer As Long)
     As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
NumChans	Number of simultaneous output channel. Valid values:

PCI-7442	`1 or 2`
PCI-7444	`1, 2, or 4`

Output the data to DO channel 0 while 1, output the data to DO channel 0 and 1 simultaneously while 2, and output data to DO channel 0, 1, 2, and 3 simultaneously while 4.

Buffer	Buffer of digital data write simultaneously to the specified output port.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

DO_WriteExtTrigLine

Description

Sets the digital output trigger line to the specified state. This function is available only for PCI-7200.

Supported card(s)

7200

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_WriteExtTrigLine (U16 CardNumber, U16
     Value)
```

Visual Basic

```
DO_WriteExtTrigLine(ByVal CardNumber As Integer,
     ByVal Value As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Value	New digital logic state 0 or 1.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

DO_WriteLine

Description

Sets the specified digital output line in the specified digital port to the specified state. This function is only available for cards that support digital output read-back functionality.

Supported card(s)

6202, 6208V/16V/08A, 6308V/08A, 7200, 7230, 7234, 7224, 7248, c7249R, 7250/51, 7252, 7256, 7258, 7260, 7296, 7300A, 7348, 7396, 7432, 7433, 7434, 7442, 7443, 7444, 7452, 8554, 9111, 9112, 9114, 9116, 9118, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_WriteLine (U16 CardNumber, U16 Port, U16
     Line, U16 State)
```

Visual Basic

```
DO_WriteLine(ByVal CardNumber As Integer, ByVal
     Port As Integer, ByVal DoLine As Integer,
     ByVal State As Integer) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Port Digital output port number. Valid values:

PCI-6202	`P6202_ISO0` `P6202_TTL0`
PCI-6208V/16V/08A	`0`
PCI-6308V/08A	`0`
PCI-7200	`0`
cPCI-7200	`0, 1 (auxiliary digital output port)`
PCI-7230/cPCI-7230	`0`
PCI-7234	`0`
PCI-7250/51	`0 to 3`
cPCI-7252	`0`
PCI-7256	`0`

PCI-7258	`0, 1`
PCI-7260	`0`
PCI-7300A/cPCI-7300A	`1 (auxiliary digital output port)`
PCI-7432/cPCI-7432	`0`
cPCI-7432R	`0, P7432R_DO_LED`
cPCI-7433R	`P7433R_DO_LED`
PCI-7434/cPCI-7434	`PORT_DO_LOW, PORT_DO_HIGH`
cPCI-7434R	`PORT_DO_LOW, PORT_DO_HIGH, P7434R_DO_LED`
PCI-7442	`P7442_CH0, P7442_CH1, P7442_TTL0, P7442_TTL1`
PCI-7443	`P7443_TTL0, P7443_TTL1`
PCI-7444	`P7444_CH0, P7444_CH1, P7444_CH2, P7444_CH3, P7444_TTL0, P7444_TTL1`
PCI-7452	`0 to 3`
PCI-8554	`0`
PCI-9111	`P9111_CHANNEL_DO, P9111_CHANNEL_EDO`
PCI-9112/cPCI-9112	`0`
cPCI-9116	`0`
PCI-9118	`0`
PCI-9114	`0`
PCI-9221	`0`
PCI-9222	`0`
PCI-9223	`0`
PCI-9524	`0`
PCI-7224/48/96/ cPCI-7248, cPCI-7249R, PCI-7348/96	`Refer to the DI_ReadLine function.`

Line The digital line to write to. Valid values:

PCI-6202	`0 to 15 (for P6202_ISO0)`
	`0 to 7 (for P6202_TTL0)`

PCI-6208V/16V/08A	`0 to 3`
PCI-6308V/08A	`0 to 3`

PCI-7200/cPCI-7200	`0 to 31 (for port 0) 0 to 3 (auxiliary output port of cPCI-7200)`
PCI-7230	`0 to 15`
PCI-7234	`0 to 31`
PCI-7250/51	`0 to 7`
cPCI-7252	`0 to 7`
PCI-7256	`0 to 15`
PCI-7258	`0 to 15`
PCI-7260	`0 to 7`
PCI-7300A/cPCI-7300A	`0 to 3`
PCI-7432/7433/7434	`0 to 31`
PCI-7442	`0 to 31 (for P7442_CH0, P7442_CH1) 0 to 15 (for P7442_TTL0, P7442_TTL1)`
PCI-7443	`0 to 15`

PCI-7444	`0 to 31 (for P7444_CH0, P7444_CH1, P7444_CH2, P7444_CH3)`
	`0 to 15 (for P7444_TTL0, P7444_TTL1)`

PCI-7452	`0 to 31`
PCI-8554	`0 to 7`
PCI-9111	`0 to 15`
PCI-9112	`0 to 15`
PCI-9114	`0 to 15`
cPCI-9116	`0 to 7`
PCI-9118DG/HG/HR	`0 to 3`
PCI-9221	`0 to 3`
PCI-9222	`0 to 15`
PCI-9223	`0 to 15`
PCI-9524	`0 to 7`
PCI-7224/48/96/ cPCI-7248, cPCI-7249R, PCI-7348/96	`Refer to the DI_ReadLine function.`

State New digital logic state 0 or 1.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

DO_WritePort

Description

Writes digital data to the specified digital output port.

Supported card(s)

6202, 6208V/16V/08A, 6308V/08A, 7200, 7230, 7234, 7224, 7248, 7249, 7250/51, 7252, 7256, 7258, 7260, 7296, 7300A, 7348, 7396, 7432, 7433, 7434, 7442, 7443, 7444, 7452, 8554, 9111, 9112, 9114, 9116, 9118, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 DO_WritePort (U16 CardNumber, U16 Port, U32
     Value)
```

Visual Basic

```
DO_WritePort (ByVal CardNumber As Integer, ByVal
     Port As Integer, ByVal Value As Long) As
     Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Port Digital output port number. The cards that support this function and their corresponding valid value are as follows:

PCI-6202	`P6202_ISO0` `P6202_TTL0`
PCI-6208V/16V/08A	`0`
PCI-6308V/08A	`0`
PCI-7200	`0`
cPCI-7200	`0, 1 (auxiliary digital output port)`
PCI-7230/cPCI-7230	`0`
PCI-7234	`0`
PCI-7224	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH`

cPCI-7248/cPCI-7248	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2CL, Channel_P2CH`
cPCI-7249R	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2CL, Channel_P2CH`
PCI-7250/51	`0 to 3`
cPCI-7252	`0`
PCI-7256	`0`
PCI-7258	`0, 1`
PCI-7260	`0`
PCI-7296	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2CL, Channel_P2CH, Channel_P3A, Channel_P3B, Channel_P3C, Channel_P3CL, Channel_P3CH, Channel_P4A, Channel_P4B, Channel_P4C, Channel_P4CL, Channel_P4CH`
PCI-7300A/cPCI-7300A	`1 (auxiliary digital output port)`
PCI-7348	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2`
PCI-7396	`Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1, Channel_P2A, Channel_P2B, Channel_P2C, Channel_P2, Channel_P3A, Channel_P3B, Channel_P3C, Channel_P3, Channel_P4A, Channel_P4B, Channel_P4C, Channel_P4`
PCI-7432/cPCI-7432	`0`
cPCI-7432R	`0, P7432R_DO_LED`
cPCI-7433R	`P7433R_DO_LED`

PCI-7434/cPCI-7434	`PORT_DO_LOW, PORT_DO_HIGH`
cPCI-7434R	`PORT_DO_LOW, PORT_DO_HIGH, P7434R_DO_LED`
PCI-7442	`P7442_CH0, P7442_CH1, P7442_TTL0, P7442_TTL1`
PCI-7443	`P7443_TTL0, P7443_TTL1`
PCI-7444	`P7444_CH0, P7444_CH1, P7444_CH2, P7444_CH3, P7444_TTL0, P7444_TTL1`
PCI-7452	`0 to 3`
PCI-8554	`0`
PCI-9111	`P9111_CHANNEL_DO, P9111_CHANNEL_EDO`
PCI-9112/cPCI-9112	`0`
cPCI-9116	`0`
PCI-9118	`0`
PCI-9114	`0`
PCI-9221	`0`
PCI-9222	`0`
PCI-9223	`0`
PCI-9524	`0`

NOTE: The value Channel_Pn, for argument Port is defined as all of the ports (Port A, B, and C) in channel n.

Value Digital data that is written to the specified port.

PCI-6202	`16-bit data (for P6202_ISO0)` `8-bit data (for P6202_TTL0)`
PCI-6208V/16V/08A	`4-bit data`
PCI-6308V/08A	`4-bit data`
PCI-7200/cPCI-7200	`32-bit data (for port 0), 4-bit data (for auxiliary output port of cPCI-7200)`
PCI-7230/cPCI-7230	`16-bit data`
PCI-7234	`32-bit data`

PCI-7224/PCI-7248/ cPCI-7248	`8-bit data`
cPCI-7249R	`8-bit data`
PCI-7250/51	`8-bit data`
cPCI-7252	`8-bit data`
PCI-7256	`16-bit data`
PCI-7258	`16-bit data`
PCI-7260	`8-bit data`
PCI-7296	`8-bit data`
PCI-7300A/cPCI-7300A	`4-bit data`
PCI-7348/PCI-7396	`24-bit data (for Channel_PnT, where n is the channel number) or 8-bit data (for Channel_PnA, Channel_PnB, Channel_PnC, where n is the channel number)`
PCI-7432/cPCI-7432/ cPCI-7432R	`32-bit data`
cPCI-7433R	`32-bit data`
PCI-7434/cPCI-7434/ cPCI-7434R	`32-bit data`
PCI-7442	`32-bit data (for P7442_CH0, P7442_CH1) or 16-bit data (for P7442_TTL0, P7442_TTL1)`
PCI-7443	`16-bit data`
PCI-7444	`32-bit data (for P7444_CH0, P7444_CH1, P7444_CH2, P7444_CH3) or 16-bit data (for P7444_TTL0, P7444_TTL1)`
PCI-7452	`32-bit data`
PCI-8554	`8-bit data`
PCI-9111	`16-bit data (for P9111_CHANNEL_DO) or 4-bit data (for P9111_CHANNEL_EDO)`
PCI-9112/cPCI-9112	`16-bit data`
PCI-9114	`16-bit data`
cPCI-9116	`8-bit data`

PCI-9118	`4-bit data`
PCI-9221	`4-bit data`
PCI-9222	`16-bit data`
PCI-9223	`16-bit data`
PCI-9524	`8-bit data`

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

EDO_9111_Config

Description

Informs the PCIS-DASK library of the EDO channel mode for PCI-9111 card.

Supported card(s)

9111

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 EDO_9111_Config (U16 CardNumber, U16 EDO_Fun)
```

Visual Basic

```
EDO_9111_Config (ByVal CardNumber As Integer,
     ByVal EDO_Fun As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
EDO_Fun	EDO ports mode. Valid modes:

`P9111_EDO_INPUT`	EDO channels are used as input channels.
`P9111_EDO_OUT_EDO`	EDO channels are used as output channels.
`P9111_EDO_OUT_CHN`	EDO channels are used as channel number output.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

EMGShutDownControl

Description

Controls the emergency shutdown.

Supported card(s)

7260

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 EMGShutDownControl (U16 CardNumber, U8 ctrl)
```

Visual Basic

```
EMGShutDownControl (ByVal CardNumber As Integer,
     ByVal ctrl As Byte) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

ctrl The control code for emergency shutdown function. The valid control codes are:

`0 EMGSHDN_OFF`	Enables the emergency shutdown function.
`1 EMGSHDN_ON`	Disables the emergency shutdown function.
`2 EMGSHDN_RECOVERY`	Clears the emergency shutdown status.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

EMGShutDownStatus

Description

Obtains the emergency shutdown status.

Supported card(s)

7260

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 EMGShutDownStatus (U16 CardNumber, U8
     *status)
```

Visual Basic

```
EMGShutDownStatus (ByVal CardNumber As Integer,
     ByVal status As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
status	Tells whether an emergency shutdown occurred. 0 if no emergency shutdown occurred or 1 if an emergency shutdown occurred.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

GCTR_Read

Description

Reads the counter value of the general-purpose counter without disturbance to the counting process.

Supported card(s)

9116

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 GCTR_Read (U16 CardNumber, U16 GCtr, U32
     *Value)
```

Visual Basic

```
GCTR_Read (ByVal CardNumber As Integer, ByVal
     GCtr As Integer, Value As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
GCtr	Counter number. Value is 0 for PCI-9116.
Value	Returns the counter value of the specified general-purpose timer/counter. Range is 0 to 65536.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidCounter
```

GCTR_Clear

Description

Turns off the specified general-purpose timer/counter operation and resets the counter value to zero.

Supported card(s)

9116

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 GCTR_Clear (U16 CardNumber, U16 GCtr)
```

Visual Basic

```
GCTR_Clear (ByVal CardNumber As Integer, ByVal
     GCtr As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
GCtr	Counter number. Value is 0 for PCI-9116.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidCounter
```

GCTR_Setup

Description

Controls the operation of the selected counter/timer.

Supported card(s)

9116

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 GCTR_Setup (U16 CardNumber, U16 GCtr, U16
     GCtrCtrl, U32 Count)
```

Visual Basic

```
GCTR_Setup (ByVal CardNumber As Integer, ByVal
     GCtr As Integer, ByVal GCtrCtrl As Integer,
     ByVal Count As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
GCtr	Counter number. Value is 0 for cPCI-9116.
GCtrCtrl	Setting for the general-purpose timer/counter control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are six groups of constants:

Timer/Counter Mode

`General_Counter`	General counter.
`Pulse_Generation`	Generation of pulse.

Timer/Counter Source

`GPTC_CLKSRC_INT`	Internal time base.
`GPTC_CLKSRC_EXT`	External time base from GP_TC_CLK pin.

Timer/Counter Gate Source

`GPTC_GATESRC_INT`	Gate is controlled by software.
`GPTC_GATESRC_EXT`	Gate is controlled by GP_TC_GATE pin.

Timer/Counter UpDown Source

`GPTC_UPDOWN_SELECT_SOFT`	Up/Down controlled by software.
`GPTC_UPDOWN_SELECT_EXT`	Up/Down controlled by GP_TC_UPDN pin.

Timer/Counter UpDown Control

`GPTC_DOWN_CTR`	Counting direction is down.
`GPTC_UP_CTR`	Counting direction is up.

Timer/Counter Enable

`GPTC_ENABLE`	General-purpose counter/timer enabled.
`GPTC_DISABLE`	General-purpose counter/timer disabled.

When two or more constants are used to form the GCtrCtrl argument, the constants are combined with the bitwise-OR operator(|).

Count Counter value of general-purpose timer/counter

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidCounter
```

GetActualRate

Description

Obtains the actual sampling rate the hardware will perform according to the board type and the specified rate.

Supported card(s)

7200, 7300A, 9111, 9112, 9113, 9114, 9118, 9221, 9222, 9223, 9812/10

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 GetActualRate (U16 CardNumber, F64
     SampleRate, F64 *ActualRate)
```

Visual Basic

```
GetActualRate (ByVal CardNumber As Integer, ByVal
     SampleRate As Double, ActualRate As Double)
     As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
SampleRate	Desired sampling rate.
ActualRate	Returns the actual acquisition rate performed. The value depends on the card type and the desired sampling rate.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

GetActualRate_9524

Description

Obtains the actual sampling rate the hardware will perform according to the board type and the specified rate.

Supported card(s)

9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 GetActualRate_9524 (U16 CardNumber, U16
     Group, F64 SampleRate, F64 *ActualRate)
```

Visual Basic

```
GetActualRate_9524 (ByVal CardNumber As Integer,
     ByVal Group As Integer, ByVal SampleRate As
     Double, ActualRate As Double) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Group	9524 supports two AI groups, load cell group and general purpose group. Valid value: P9524_AI_LC_Group P9524_AI_GP_Group
SampleRate	Desired sampling rate.
ActualRate	Returns the actual acquisition rate performed. The value is a table-lookup value depends on the setting of acquisition mode.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

GetBaseAddr

Description

Gets the I/O base addresses of the device with a specified card index.

Supported card(s)

6202, 6208V/6216V, 6208A, 6308V, 6308A, 7200, 7230, 7233, 7234, 7224, 7248, 7249, 7250, 7252, 7256, 7258, 7260, 7296, 7300A, 7348, 7396, 7432, 7433, 7434, 7442, 7443, 7444, 7452, 8554, 9111, 9112, 9113, 9114, 9116, 9118, 9812/10, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 GetBaseAddr (U16 CardNumber, U32 *BaseAddr,
     U32 *BaseAddr2)
```

Visual Basic

```
GetBaseAddr (ByVal CardNumber As Integer,
     BaseAddr As Long, BaseAddr2 As Long) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
BaseAddr	Returns the I/O base address.
BaseAddr2	Returns the second base address #2. This is only available for cards that support two I/O base addresses, such as PCI-9113 and PCI-9114. For PCI-6202, PCI-9221, PCI-9222, and PCI-9223, this parameter returns the memory address of the specified card.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

GetCardIndexFromID

Description

Obtains the card type and the sequence number of the device with a specified card ID. This is the reverse function of Release_Card.

Supported card(s)

6202, 6208V/6216V, 6208A, 6308V, 6308A, 7200, 7230, 7233, 7234, 7224, 7248, 7249, 7250, 7252, 7256, 7258, 7260, 7296, 7300A, 7348, 7396, 7432, 7433, 7434, 7442, 7443, 7444, 7452, 8554, 9111, 9112, 9113, 9114, 9116, 9118, 9812/10, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++ and Borland C++

```
I16 GetCardIndexFromID (U16 CardNumber, U16
     *cardType, U16 *cardIndex)
```

Visual Basic

```
GetCardIndexFromID (ByVal CardNumber As Integer,
     cardType As Integer, cardIndex As Integer)
     As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
cardType	Returns the card type.
cardIndex	Returns the sequence number of the card with the same card type.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

GetCardType

Description

Obtains the card type of the device with a specified card index.

Supported card(s)

6202, 6208V/6216V, 6208A, 6308V, 6308A, 7200, 7230, 7233, 7234, 7224, 7248, 7249, 7250, 7252, 7256, 7258, 7260, 7296, 7300A, 7348, 7396, 7432, 7433, 7434, 7442, 7443, 7444, 7452, 8554, 9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9812/10, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 GetCardType (U16 CardNumber, U16 *cardType)
```

Visual Basic

```
GetCardType (ByVal CardNumber As Integer,
     cardType As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
cardType	Returns the card type.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

GetInitPattern

Description

Obtains the state of relays set by the onboard switches (7260) or the state set by SetInitPattern (7442/7444).

Supported card(s)

7260, 7442, 7444

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 GetInitPattern (U16 CardNumber, U8 patternID,
     U32 *pattern)
```

Visual Basic

```
GetInitPattern (ByVal CardNumber As Integer,
     ByVal patternID As Byte, pattern As Long) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
patternID	Valid pattern ID.

PCI-7260

`INIT_PTN`	State of relays at power-on.
`EMGSHDN_PTN`	State of relays while emergency shutdown condition is happened.

PCI-7442

`INIT_PTN_CH0`	State of DO channel 0 at power-on.
`INIT_PTN_CH1`	State of DO channel 1 at power-on.
`SAFTOUT_PTN_CH0`	State of DO channel 0 while WDT overflows.
`SAFTOUT_PTN_CH1`	State of DO channel 1 while WDT overflows.

PCI-7444

`INIT_PTN_CH0`	State of DO channel 0 at power-on.
`INIT_PTN_CH1`	State of DO channel 1 at power-on.
`INIT_PTN_CH2`	State of DO channel 2 at power-on.
`INIT_PTN_CH3`	State of DO channel 3 at power-on.

`SAFTOUT_PTN_CH0`	State of DO channel 0 while WDT overflows.
`SAFTOUT_PTN_CH1`	State of DO channel 1 while WDT overflows.
`SAFTOUT_PTN_CH2`	State of DO channel 2 while WDT overflows.
`SAFTOUT_PTN_CH3`	State of DO channel 3 while WDT overflows.

pattern	Returns the state of relay or the state set by SetInitPattern function.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

GetLCRAddr

Description

Obtains the LCR base address of the device with a specified card index as defined by the onboard PCI controller.

Supported card(s)

6202, 6208V/6216V, 6208A, 6308V, 6308A, 7200, 7230, 7233, 7234, 7224, 7248, 7249, 7250, 7252, 7256, 7258, 7260, 7296, 7300A, 7348, 7396, 7432, 7433, 7434, 7442, 7443, 7444, 7452, 8554, 9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9812/10, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 GetLCRAddr(U16 CardNumber, U32 *LcrAddr)
```

Visual Basic

```
GetLCRAddr (ByVal CardNumber As Integer, LcrAddr
     As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
LcrAddr	Returns the LCR base address.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

GPTC_9524_PG_Config

Description

This function sets the generated pulse number of GPTC pulse generator.

Supported card(s)

9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 GPTC_9524_PG_Config (U16 CardNumber, U16
     GCtr, U32 PulseGenNum)
```

Visual Basic

```
GPTC_9524_PG_Config (ByVal CardNumber As Integer,
     ByVal GCtr As Integer, ByVal PulseGenNum As
     Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
GCtr	The general timer/counter number. Valid value: P9524_CTR_PG0 P9524_CTR_PG1 P9524_CTR_PG2
PulseGenNum	The generated pulse number. Valid value: 0 Infinite generation 1 to 16777215 Finite generation

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorInvalidCounter
ErrorCardNotRegistered
ErrorFuncNotSupport
```

GPTC_Clear

Description

Halts the specified general-purpose timer/counter operation and reloads the initial value of the timer/counter.

Supported card(s)

6202, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 GPTC_Clear (U16 CardNumber, U16 GCtr)
```

Visual Basic

```
GPTC_Clear (ByVal CardNumber As Integer, ByVal
     GCtr As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
GCtr	The counter number.

PCI-6202	`P6202_GPTC0`
	`P6202_GPTC1`
	`P6202_ENCODER0`
	`P6202_ENCODER1`
	`P6202_ENCODER2`

PCI-9221	`0 to 1`

PCI-9222
PCI-9223
	`P922x_GPTC0`
	`P922x_GPTC1`
	`P922x_GPTC2`
	`P922x_GPTC3`
	`P922x_ENCODER0`
	`P922x_ENCODER1`

PCI-9524	`P9524_CTR_PG0`
	`P9524_CTR_PG1`
	`P9524_CTR_PG2`
	`P9524_CTR_QD0`
	`P9524_CTR_QD1`
	`P9524_CTR_QD2`
	`P9524_CTR_INTCOUNTER`

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorInvalidCounter
ErrorFuncNotSupport
```

GPTC_Control

Description

Controls for the selected counter/timer by software.

Supported card(s)

6202, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 GPTC_Control (U16 CardNumber, U16 GCtr, U16
     ParamID, U16 Value)
```

Visual Basic

```
GPTC_Control (ByVal CardNumber As Integer, ByVal
     GCtr As Integer, ByVal ParamID As Integer,
     ByVal Value As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
GCtr	The counter number.

PCI-6202	`P6202_GPTC0`
	`P6202_GPTC1`
	`P6202_ENCODER0`
	`P6202_ENCODER1`
	`P6202_ENCODER2`

PCI-9221	`0 to 1`

PCI-9222
PCI-9223
	`P922x_GPTC0`
	`P922x_GPTC1`
	`P922x_GPTC2`
	`P922x_GPTC3`
	`P922x_ENCODER0`
	`P922x_ENCODER1`

PCI-9524	`P9524_CTR_PG0`
	`P9524_CTR_PG1`
	`P9524_CTR_PG2`
	`P9524_CTR_QD0`
	`P9524_CTR_QD1`
	`P9524_CTR_QD2`
	`P9524_CTR_INTCOUNTER`

ParamID The ID of the internal parameter of the general-purpose timer/counter you want to control. Valid control parameters:

PCI-6202, PCI-9221, PCI-9222, PCI-9223	
`IntGATE`	Internal gate
`IntUpDnCTR`	Internal updown counter
`IntENABLE`	Starts or stops counter operation
PCI-9524	
`P9524_CTR_Enable`	Starts or stops counter operation

Value The value for the control item specified by the ParamID parameter. The valid value is 0 or 1.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorInvalidCounter
ErrorUndefinedParameter
ErrorFuncNotSupport
```

GPTC_EventSetup

Description

Sets the configurations of the selected event of .the counter/timer.

Supported card(s)

6202, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 GPTC_EventSetup(U16 CardNumber, U16 GCtr, U16
     Mode, U16 Ctrl, U32 LVal_1, U32 LVal_2);
```

Visual Basic

```
GPTC_EventSetup (ByVal CardNumber As Integer,
     ByVal GCtr As Integer, ByVal mode As
     Integer, ByVal Ctrl As Integer, LVal_1 As
     Long, LVal_2 As Long) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

GCtr The counter number.

PCI-6202

P6202_ENCODER0

P6202_ENCODER1

P6202_ENCODER2

PCI-9222
PCI-9223

P922x_ENCODER0

P922x_ENCODER1

Mode Event mode. Valid values:

PCI-6202

P6202_EVT_MOD_EPT

PCI-9222 and **PCI-9223**

P922x_EVT_MOD_EPT

Ctrl The setting for event of .the counter/timer. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H.

PCI-6202

There are three groups of constants:

Encoder Position Trigger pulse width Selection

P6202_EPT_PULWIDTH_200us

P6202_EPT_PULWIDTH_2ms

P6202_EPT_PULWIDTH_20ms

P6202_EPT_PULWIDTH_200ms

Enable Encoder Position Trigger callback

P6202_EPT_TRGOUT_CALLBACK (Perform callback function set by GPTC_EventCallBack())

Enable Encoder Position Trigger output

P6202_EPT_TRGOUT_AFI (Output pulse to AFI)

PCI-9222 and **PCI-9223**

There are three groups of constants:

Encoder Position Trigger pulse width Selection

P922x_EPT_PULWIDTH_200us

P922x_EPT_PULWIDTH_2ms

P922x_EPT_PULWIDTH_20ms

P922x_EPT_PULWIDTH_200ms

Enable Encoder Position Trigger callback

P922x_EPT_TRGOUT_CALLBACK (Perform callback function set by GPTC_EventCallBack())

Enable Encoder Position Trigger output

P922x_EPT_TRGOUT_GPO (Output pulse to GPO4/5 (Encoder0/1)

When two or more constants are used to form the ConfigCtrl argument, the constants are combined with the bitwise-OR operator(|).

LVal_1	**PCI-6202** 0 to 0x7ffff (corresponding to 19-bit 2's complement) **PCI-9222** and **PCI-9223** 0 to 0xffffffff (corresponding to 32-bit 2's complement)
Lval_2	*PCI*-6202, PCI-9222, and PCI-9223: Not used

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

GPTC_EventCallBack (Win32 Only)

Description

Controls and notifies the user's application when a specified GPTC event occurs.

Supported card(s)

6202, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 GPTC_EventCallBack(U16 CardNumber, I16
     Enabled, I16 EventType, U32 callbackAddr)
```

Visual Basic

```
GPTC_EventCallBack (ByVal CardNumber As Integer,
     ByVal Enabled As Integer, ByVal EventType As
     Integer, callbackAddr As Long) As Integer
```

Parameter(s)

CardNumber ID of the card performing the operation.

Enabled Add or remove the event message.

The valid values:

0: remove

1: add

EventType The type of event

The valid values:

PCI-6202

P6202_EVT_TYPE_EPT0: position trigger of encoder 0

P6202_EVT_TYPE_EPT1: position trigger of encoder 1

P6202_EVT_TYPE_EPT2: position trigger of encoder 2

PCI-9222 and **PCI-9223**

P922x_EVT_TYPE_EPT0: position trigger of encoder 0

P922x_EVT_TYPE_EPT1: position trigger of encoder 1

PCI-9524

P9524_Event_Timer: Timer event of internal counter

callbackAddr	The address of the user callback function. PCIS-DASK calls this function when the specified event occurs. If you wish to remove the event message, set callbackAddr to 0.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

GPTC_Read

Description

Reads the counter value of the general-purpose counter without interfering with the counting process.

Supported card(s)

6202, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 GPTC_Read (U16 CardNumber, U16 GCtr, U32
     *pValue)
```

Visual Basic

```
GPTC_Read (ByVal CardNumber As Integer, ByVal
     GCtr As Integer, pValue As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
GCtr	The counter number.

PCI-6202	`P6202_GPTC0`
	`P6202_GPTC1`
	`P6202_ENCODER0`
	`P6202_ENCODER1`
	`P6202_ENCODER2`

PCI-9221	`0 to 1`

PCI-9222
PCI-9223
	`P922x_GPTC0`
	`P922x_GPTC1`
	`P922x_GPTC2`
	`P922x_GPTC3`
	`P922x_ENCODER0`
	`P922x_ENCODER1`

PCI-9524	`P9524_CTR_PG0`
	`P9524_CTR_PG1`
	`P9524_CTR_PG2`
	`P9524_CTR_QD0`
	`P9524_CTR_QD1`
	`P9524_CTR_QD2`

pValue Returns the counter value of the specified general-purpose timer/counter.

PCI-6202	`19-bit counter value`
PCI-9221	`32-bit counter value`
PCI-9222	`32-bit counter value`
PCI-9223	`32-bit counter value`
PCI-9524	`24-bit counter value`

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorInvalidCounter
ErrorFuncNotSupport
```

GPTC_Setup

Description

Sets the configurations of the selected counter/timer.

Supported card(s)

6202, 9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 GPTC_Setup (U16 CardNumber, U16 GCtr, U16
     Mode, U16 SrcCtrl, U16 PolCtrl, U32
     LReg1_Val, U32 LReg2_Val)
```

Visual Basic

```
GPTC_Setup (ByVal CardNumber As Integer, ByVal
     GCtr As Integer, ByVal Mode As Integer,
     ByVal SrcCtrl As Integer, ByVal PolCtrl As
     Integer, ByVal LReg1_Val As Long, ByVal
     LReg2_Val As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
GCtr	The counter number.

PCI-6202	`P6202_GPTC0`
	`P6202_GPTC1`
	`P6202_ENCODER0`
	`P6202_ENCODER1`
	`P6202_ENCODER2`

PCI-9221	`0 to 1`

PCI-9222
PCI-9223
	`P922x_GPTC0`
	`P922x_GPTC1`
	`P922x_GPTC2`
	`P922x_GPTC3`
	`P922x_ENCODER0`
	`P922x_ENCODER1`

PCI-9524	`P9524_CTR_PG0`
	`P9524_CTR_PG1`
	`P9524_CTR_PG2`
	`P9524_CTR_QD0`
	`P9524_CTR_QD1`
	`P9524_CTR_QD2`
	`P9524_CTR_INTCOUNTER`

Mode The timer/counter mode. Refer to the hardware manual for the mode description. Valid modes:

PCI-6202, PCI-9221, PCI-9222, PCI-9223

SimpleGatedEventCNT	EdgeSeparationMSR
SinglePeriodMSR	SingleTrigContPulseGenPWM
SinglePulseWidthMSR	ContGatedPulseGenPWM
SingleGatedPulseGen	CW_CCW_Encoder
SingleTrigPulseGen	x1_AB_Phase_Encoder
RetrigSinglePulseGen	x2_AB_Phase_Encoder
SingleTrigContPulseGen	x4_AB_Phase_Encoder
ContGatedPulseGen	Phase_Z

PCI-9524

P9524_PulseGen_OUTDIR_N

P9524_PulseGen_OUTDIR_R

P9524_PulseGen_CCW

P9524_x4_AB_Phase_Decoder

P9524_Timer

SrcCtrl The setting for general-purpose timer/counter source control. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are three groups of constants:

PCI-6202, PCI-9221, PCI-9222, PCI-9223

Timer/Counter Source

`GPTC_CLK_SRC_Int`	Internal time base
`GPTC_CLK_SRC_Ext`	External time base from the GPTC_CLK pin

Timer/Counter Gate Source

`GPTC_GATE_SRC_Int`	Gate is controlled by software.
`GPTC_GATE_SRC_Ext`	Gate is controlled by the GPTC_GATE pin.

Timer/Counter UpDown Source

`GPTC_UPDOWN_Int`	Up/Down is controlled by software.
`GPTC_UPDOWN_Ext`	Up/Down is controlled by the GPTC_UD pin.

PCI-9524	Ignore

When two or more constants are used to form the SrcCtrl argument, the constants are combined with the bitwise-OR operator(|).

PolCtrl The polarity settings for general-purpose timer/counter. This argument is an integer expression formed from one or more of the manifest constants defined in DASK.H. There are three groups of constants:.

PCI-9221, PCI-9222, PCI-9223

Timer/Counter Gate Polarity

`GPTC_GATE_LACTIVE`	Low active
`GPTC_GATE_HACTIVE`	High active

Timer/Counter UpDown Polarity

`GPTC_UPDOWN_LACTIVE`	Low active
`GPTC_UPDOWN_HACTIVE`	High active

Timer/Counter Clock Source Polarity

`GPTC_CLKSRC_LACTIVE`	Low active
`GPTC_CLKSRC_HACTIVE`	High active

Timer/Counter Output Polarity `(PCI-9222/9223 only)`

`GPTC_OUTPUT_LACTIVE`	Low active
`GPTC_OUTPUT_HACTIVE`	High active

PCI-6202, PCI-9524 Ignore

When two or more constants are used to form the PolCtrl argument, the constants are combined with the bitwise-OR operator(|)

LReg1_Val The meaning for the value depends on the mode the timer /counter performs.

PCI-6202, PCI-9221, PCI-9222, PCI-9223

SimpleGatedEventCNT	Configures as initial count of GPTC.
SinglePeriodMSR	Configures as initial count of GPTC.
SinglePulseWidthMSR	Configures as initial count of GPTC.
SingleGatedPulseGen	Configures as the pulse width.
SingleTrigPulseGen	Configures as the pulse width.
RetrigSinglePulseGen	Configures as the pulse width.
SingleTrigContPulseGen	Configures as the pulse width.
ContGatedPulseGen	Configures as the pulse width.
EdgeSeparationMSR	Configures as initial count of GPTC.
SingleTrigContPulseGenPWM	Configures as the pulse initial count.
ContGatedPulseGenPWM	Configures as the pulse initial count.
CW_CCW_Encoder	Not used
x1_AB_Phase_Encoder	Not used
x2_AB_Phase_Encoder	Not used
x4_AB_Phase_Encoder	Not used
Phase_Z	Z_Phase Phase

PCI-9524

P9524_PulseGen_OUTDIR_N	Configures as the pulse initial count. 0 is not valid for this argument and the valid range of LReg1_Val + LReg2_Val is between 0x2 and 0xfffffe.
P9524_PulseGen_OUTDIR_R	Configures as the pulse initial count. 0 is not valid for this argument and the valid range of LReg1_Val + LReg2_Val is between 0x2 and 0xfffffe.
P9524_PulseGen_CW	Configures as the pulse initial count. 0 is not valid for this argument and the valid range of LReg1_Val + LReg2_Val is between 0x2 and 0xfffffe.
P9524_PulseGen_CCW	Configures as the pulse initial count. 0 is not valid for this argument and the valid range of LReg1_Val + LReg2_Val is between 0x2 and 0xfffffe.
P9524_x4_AB_Phase_Decoder	Not Used
P9524_Timer	Configures as the counter divisor. Range: 0x2-0xffffffff

LReg2_Val The meaning for the value depends on the mode the timer /counter performs.

PCI-6202, PCI-9221, PCI-9222, PCI-9223

SimpleGatedEventCNT	Not used
SinglePeriodMSR	Not used
SinglePulseWidthMSR	Not used
SingleGatedPulseGen	Not used
SingleTrigPulseGen	Not used
RetrigSinglePulseGen	Not used
SingleTrigContPulseGen	Not used
ContGatedPulseGen	Not used
EdgeSeparationMSR	Not used
SingleTrigContPulseGenPWM	Configures as the pulse length count.
ContGatedPulseGenPWM	Configures as the pulse length count.

CW_CCW_Encoder	Not used
x1_AB_Phase_Encoder	Not used
x2_AB_Phase_Encoder	Not used
x4_AB_Phase_Encoder	Not used
Phase_Z	Z_Phase Mode

PCI-9524

P9524_PulseGen_OUTDIR_N	Configures as the pulse length count. 0 is not valid of this argument and the valid range of LReg1_Val + LReg2_Val is between 0x2 and 0xfffffe.
P9524_PulseGen_OUTDIR_R	Configures as the pulse length count. 0 is not valid of this argument and the valid range of LReg1_Val + LReg2_Val is between 0x2 and 0xfffffe.
P9524_PulseGen_CW	Configures as the pulse length count. 0 is not valid of this argument and the valid range of LReg1_Val + LReg2_Val is between 0x2 and 0xfffffe.
P9524_PulseGen_CCW	Configures as the pulse length count. 0 is not valid of this argument and the valid range of LReg1_Val + LReg2_Val is between 0x2 and 0xfffffe.
P9524_x4_AB_Phase_Decoder	Not used
P9524_Timer	Not used

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorInvalidCounter
ErrorUndefinedParameter
ErrorFuncNotSupport
```

GPTC_Status

Description

Reads the latched GPTC status of the general-purpose counter from the GPTC status register.

Supported card(s)

6202, 9221, 9222, 9223

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 GPTC_Status (U16 CardNumber, U16 GCtr, U16
     *pValue)
```

Visual Basic

```
GPTC_Status (ByVal CardNumber As Integer, ByVal
     GCtr As Integer, pValue As Integer) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
GCtr	The counter number.

PCI-6202

P6202_GPTC0

P6202_GPTC1

P6202_ENCODER0

P6202_ENCODER1

P6202_ENCODER2

PCI-9221

0 to 1

PCI-9222 and PCI-9223

P922x_GPTC0

P922x_GPTC1

P922x_GPTC2

P922x_GPTC3

P922x_ENCODER0

P922x_ENCODER1

pValue	Returns the latched GPTC status of the specified general-purpose timer/counter from the GPTC status register. Value formats:

`PCI-6202 (for P602_GPTC)`
`PCI-9221`
`PCI-9222/9223 (for P922x_GPTC)`

`bit 0`	1 indicates that the GPTC is counting. 0 indicates that the GPTC is not counting.
`bit 1`	1 indicates that the GPTC operation is done. 0 indicates that the GPTC operation is not yet done.

`PCI-6202 (for P6202_ENCODER)`
`PCI-9222/9223 (for P922x_ENCODER)`

`bit 0`	Phase A input of the indicated encoder
`bit 1`	Phase B input of the indicated encoder
`bit 2`	Phase Z input of the indicated encoder
`bit 3`	Original input of the indicated encoder

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorInvalidCounter
ErrorFuncNotSupport
```

HotResetHoldControl

Description

Controls the hot-system reset DO hold function and, if hot-reset-hold is enabled, holds the current DO output value while the computer hot resets. Otherwise, the initial pattern is outputted.

Supported card(s)

7442, 7444

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 HotResetHoldControl (U16 CardNumber, U8
     enable)
```

Visual Basic

```
HotResetHoldControl (ByVal CardNumber As Integer,
     ByVal enable As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
enable	Control code for hot reset hold function. The valid control codes are the following:

`0 HRH_OFF`	Enables the hot reset hold function.
`1 HRH_ON`	Disables the hot reset hold function.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

HotResetHoldStatus

Description

Reads the hot-system reset DO hold status.

Supported card(s)

7442, 7444

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 HotResetHoldStatus (U16 CardNumber, U8 *sts)
```

Visual Basic

```
HotResetHoldStatus (ByVal CardNumber As Integer,
     sts As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
sts	Hot reset hold status read.

0	Hot reset hold functionality is disabled.
1	Hot reset hold functionality is enabled.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

IdentifyLED_Control

Description

Controls the identification LED.

Supported card(s)

7260

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 IdentifyLED_Control (U16 CardNumber, U8 ctrl)
```

Visual Basic

```
IdentifyLED_Control (ByVal CardNumber As Integer,
     ByVal ctrl As Byte) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
ctrl	Turns the identification LED on (1) or off (0).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

PCI9524_Acquire_AD_CalConst

Description

Obtains the AD calibration constants of PCI-9524. While the auto calibration or EEPROM loading is performed completely, you can use the function to obtain the calibrated AD constants or the loaded AD constants. Please refer the function description of PCI_DB_Auto_Calibration_ALL or PCI_Load_CAL_Data.

Supported card(s)

9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 PCI9524_Acquire_AD_CalConst (U16 CardNumber,
     U16 Group, U16 ADC_Range, U16 ADC_Speed, U32
     *CalDate, F32 *CalTemp, U32 *ADC_offset, U32
     *ADC_gain, F64 *Residual_offset, F64
     *Residual_scaling)
```

Visual Basic

```
GetActualRate_9524 (ByVal CardNumber As Integer,
     ByVal Group As Integer, ByVal ADC_Range As
     Integer, ByVal ADC_Speed As Integer, CalDate
     As Long, CalTemp As Single, ADC_offset As
     Single, ADC_gain As Single, Residual_offset
     As Double, Residual_scaling As Double) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Group	9524 supports two AI groups, load cell group and general purpose group. The two groups have the different calibration constants. Valid value: P9524_AI_LC_Group P9524_AI_GP_Group

ADC_Range Each ADC range of PCI-9524 has the corresponding calibration constants. Valid value:

Load Cell Group (P9524_AI_LC_Group)

0

General Purpose Group (P9524_AI_GP_Group)

AD_B_10_V

AD_B_5_V

AD_B_2_5_V

AD_B_1_25_V

ADC_Speed Each ADC speed of PCI-9524 has the corresponding calibration constants. Valid value:

P9524_ADC_30K_SPS

P9524_ADC_15K_SPS

P9524_ADC_7K5_SPS

P9524_ADC_3K75_SPS

P9524_ADC_2K_SPS

P9524_ADC_1K_SPS

P9524_ADC_500_SPS

P9524_ADC_100_SPS

P9524_ADC_60_SPS

P9524_ADC_50_SPS

P9524_ADC_30_SPS

P9524_ADC_25_SPS

P9524_ADC_15_SPS

P9524_ADC_10_SPS

P9524_ADC_5_SPS

P9524_ADC_2R5_SPS

CalDate Returns the Date of the calibration constants. The format is YYYYMMDD (YYYY: Year, MM: Month, DD: Day)

CalTemp	Returns the Centigrade Temperature during the calibration period.
ADC_offset	Returns the calibrated ADC offset constant of the specified group, range, and speed.
ADC_gain	Returns the calibrated ADC gain constant of the specified group, range, and speed.
Residual_offset	Returns the software compensation offset constant of the specified group, range, and speed.
Residual_scaling	Returns the software compensation scaling constant of the specified group, range, and speed.

Return Code(s)

```
NoError
ErrorFuncNotSupport
ErrorUndefinedParameter
```

PCI9524_Acquire_DA_CalConst

Description

Obtains the DA calibration constants of PCI-9524. While the auto calibration or EEPROM loading is performed completely, you can use the function to obtain the calibrated DA constants or the loaded DA constants. Please refer the function description of PCI_DB_Auto_Calibration_ALL or PCI_Load_CAL_Data.

Supported card(s)

9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 PCI9524_Acquire_DA_CalConst (U16 CardNumber,
     U16 Channel, U32 *CalDate, F32 *CalTemp, U8
     *DAC_offset, U8 *DAC_linearity, F32
     *Gain_factor)
```

Visual Basic

```
GetActualRate_9524 (ByVal CardNumber As Integer,
     ByVal Channel As Integer, CalDate As Long,
     CalTemp As Single, DAC_offset As Byte,
     DAC_linearity As Byte, Gain_factor As
     Single) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Channel	AO channel number. Each AO channel has the different corresponding DA constants. Valid value: 0 or 1
CalDate	Returns the Date of the calibration constants. The format is YYYYMMDD (YYYY: Year, MM: Month, DD: Day)
CalTemp	Returns the Centigrade Temperature during the calibration period.
DAC_offset	Returns the calibrated DAC offset constant of the specified AO channel.

DAC_linearity	Returns the calibrated DAC linearity constant of the specified AO channel.
Gain_factor	Returns the gain factor of the specified AO channel.

Return Code(s)

```
NoError
ErrorFuncNotSupport
ErrorUndefinedParameter
```

PCI_DB_Auto_Calibration_ALL

Description

Calibrates the specified device. When the function is called, the device goes into a self-calibration cycle. The function does not return until the self-calibration is completed or has timed out.

Supported card(s)

9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 PCI_DB_Auto_Calibration_ALL (U16 CardNumber)
```

Visual Basic

```
PCI_DB_Auto_Calibration_ALL (ByVal CardNumber As
     Integer)
```

Parameter(s)

CardNumber	ID of the card performing the operation.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorCalibrationTimeOut
```

PCI_EEPROM_CAL_Constant_Update

Description

Saves new calibration constants to the specified EEPROM bank.

Supported card(s)

9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 PCI_EEPROM_CAL_Constant_Update (U16
     CardNumber, U16 bank)
```

Visual Basic

```
PCI_EEPROM_CAL_Constant_Update (ByVal CardNumber
     As Integer, ByVal bank As Integer) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
bank	The storage location on EEPROM. Valid value:

PCI-9221, PCI-9222, PCI-9223

EEPROM_USER_BANK1

PCI-9524

1 to 3

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorUndefinedParameter
ErrorFuncNotSupport
```

PCI_Load_CAL_Data

Description

Loads calibration constants from the specified bank of EEPROM.

Supported card(s)

9221, 9222, 9223, 9524

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 PCI_Load_CAL_Data (U16 CardNumber, U16 bank)
```

Visual Basic

```
PCI_Load_CAL_Data (ByVal CardNumber As Integer,
     ByVal bank As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.

bank	The storage bank on EEPROM. Valid values:
	PCI-9221, PCI-9222, PCI-9223
	`EEPROM_DEFAULT_BANK`
	`EEPROM_USER_BANK1`
	PCI-9524
	`0 TO 3`

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorUndefinedParameter
ErrorFuncNotSupport
```

PWM_Output

Description

Start the pwm output.

Supported card(s)

6202

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 PWM_Output(U16 CardNumber, U16 Channel, U32
     high_interval, U32 low_interval)
```

Visual Basic

```
PWM_Output (ByVal CardNumber As Integer, ByVal
     Channel As Integer, ByVal high_interval As
     Long, low_interval As Long) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Channel	PWM channel
high_interval	The high interval
low_interval	The low interval

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

PWM_Stop

Description

Stop the pwm output.

Supported card(s)

6202

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 PWM_Stop(U16 CardNumber, U16 Channel)
```

Visual Basic

```
PWM_Stop (ByVal CardNumber As Integer, ByVal
     Channel As Integer) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Channel	PWM channel

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

Register_Card

Description

Initializes the hardware and software states of a NuDAQ PCI-bus data acquisition card, then returns a numeric card ID that corresponds to the initialized card. Register_Card must be called before any other PCIS-DASK library functions can be called for a particular card. The function initializes the card and variables internal to the PCIS-DASK library. Because NuDAQ PCI-bus data acquisition cards meet plug-and-play specifications, the base address (pass-through address) and IRQ level are assigned directly by the system BIOS.

Supported card(s)

6202, 6208V/6216V, 6208A, 6308V, 6308A, 7200, 7230, 7233, 7234, 7224, 7248, 7249, 7250, 7252, 7256, 7258, 7260, 7296, 7300A, 7348, 7396, 7432, 7433, 7434, 7442, 7443, 7444, 7452, 8554, 9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 Register_Card (U16 CardType, U16 card_num)
```

Visual Basic

```
Register_Card (ByVal CardType As Integer, ByVal
     card_num As Integer) As Integer
```

Parameter(s)

CardType	Type of card to be initialized. ADLINK periodically upgrades PCIS-DASK to add support for new NuDAQ PCI-bus data acquisition cards and NuIPC CompactPCI cards. Refer to release notes of the card to know if PCIS-DASK supports that card. These are the constants defined in DASK.H that represent the NuDAQ PCI-bus data acquisition cards supported by PCIS-DASK:

- PCI-6202
- PCI_6208V (for PCI-6208V/6216V)
- PCI_6208A
- PCI_6308V
- PCI_6308A
- PCI_7200 (for PCI-7200/cPCI-7200)
- PCI_7230 (for PCI-7230/cPCI-7230)
- PCI_7233 (for PCI-7233/PCI-7233H)
- PCI_7234
- PCI_7248 (for PCI-7224/PCI-7248/cPCI-7248)
- PCI_7249 (for cPCI-7249R)
- PCI_7250
- PCI_7252 (for cPCI-7252)
- PCI_7256
- PCI_7258
- PCI_7260
- PCI_7296
- PCI_7300A_RevA (for PCI_7300A_RevA/
- cPCI_7300A_RevA)
- PCI_7300A_RevB (for PCI_7300A_RevB/
- cPCI_7300A_RevB)
- PCI_7396 (for PCI-7348/PCI-7396)
- PCI_7432 (for PCI-7432/cPCI-7432/cPCI-7432R)
- PCI_7433 (for PCI-7433/cPCI-7433/cPCI-7433R)
- PCI_7434 (for PCI-7434/cPCI-7434/cPCI-7434R)
- PCI_7442
- PCI_7443
- PCI_7444
- PCI_7452
- PCI_8554
- PCI_9111DG

- PCI_9111HR
- PCI_9112 (for PCI-9112/cPCI-9112)
- PCI_9113
- PCI_9114DG
- PCI_9114HG
- PCI_9116 (for cPCI-9116)
- PCI_9118DG
- PCI_9118HG
- PCI_9118HR
- PCI_9221
- PCI_9222
- PCI_9223
- PCI_9524
- PCI_9810 (for PCI-9810)
- PCI_9812 (for PCI-9812)

card_num	Sequence number of the card with the same card type (as defined in argument CardType) or that belongs to the same card type series (except PCI-7300A_Rev. A and PCI-7300A Rev. B) plugged in the PCI slot. The card sequence number setting is according to the PCI slot sequence in the mainboard. The first card (in the first slot) is card_num=0. For example, if there is one PCI-9111DG card (in the first PCI slot), a PCI-9111HR card, and two PCI-9112 cards plugged in the computer, the PCI-9111DG card must be registered with card_num=0, and the PCI-9111HR card with card_num=1. The PCI-9112 card in the first slot should be registered with card_num=0, and next one with card_num=1. The PCI-7256, PCI-7258, PCI-7260, PCI-7442, PCI-7443, PCI-7444, and PCI-7452 Series cards support Board ID functionality. You can use the onboard switch to set the board's ID and replace the card number by the board ID in this argument.

The following table categorizes the NuDAQ PCI devices by card type series.

Card Type Series	Device Type
PCI-6202	PCI-6202
PCI-6208 Series	PCI-6208V, PCI-6216V, PCI-6208A
PCI-6308 Series	PCI-6308V, PCI_6308A
PCI-7200/cPCI-7200	PCI-7200/cPCI-7200
PCI-7230/cPCI-7230	PCI-7230/cPCI-7230
PCI-7233	PCI-7233, PCI-7233H
PCI-7234	PCI-7234
PCI-7224/PCI-7248/cPCI-7248	PCI-7224/PCI-7248/cPCI-7248
PCI-7249	cPCI-7249R
PCI-7250	PCI-7250
PCI-7252	cPCI-7252
PCI-7256	PCI-7256
PCI-7258	PCI-7258
PCI-7260	PCI-7260
PCI-7296	PCI-7296
PCI_7300A_RevA cPCI-7300A_RevA	PCI-7300A_RevA/cPCI-7300A_RevA
PCI_7300A_RevB cPCI-7300A_RevB	PCI-7300A_RevB/cPCI-7300A_RevB
PCI-7348/PCI-7396	PCI-7348/PCI-7396
PCI-7432/cPCI-7432 Series	PCI-7432/cPCI-7432/cPCI-7432R
PCI-7433/cPCI-7433 Series	PCI-7433/cPCI-7433/cPCI-7433R
PCI-7434/cPCI-7434 Series	PCI-7434/cPCI-7434/cPCI-7434R
PCI-7442	PCI-7442
PCI-7443	PCI-7443
PCI-7444	PCI-7444
PCI-7452	PCI-7452
PCI-8554	PCI-8554
PCI-9111 Series	PCI-9111DG, PCI-9111HR
PCI-9112/cPCI-9112	PCI-9112/cPCI-9112
PCI-9113	PCI-9113, PCI-9113A

Card Type Series	Device Type
PCI-9114 Series	PCI-9114DG, PCI-9114HG, PCI-9114A-DG, PCI-9114A-HG
PCI-9116	cPCI-9116
PCI-9118 Series	PCI-9118DG, PCI-9118HG, PCI-9118HR
PCI-9221	PCI-9221
PCI-9222 Series	PCI-9222, PCI-9223
PCI-9524	PCI-9524
PCI-9812 Series	PCI-9812, PCI-9810

Return Code(s)

Returns a numeric card ID for the initialized card. The card ID range is between 0 and 31. If any error occurs, this returns a negative error code. Possible error codes are listed below:

```
ErrorTooManyCardRegistered
ErrorUnknownCardType
ErrorOpenDriverFailed
ErrorOpenEventFailed
```

Release_Card

Description

There are at most 32 cards that can be registered simultaneously. This function tells the PCIS-DASK library that the registered card is not currently used and may be released. Releasing a card would make room for a new card to register. You also need to use this function at the end of a program to release all registered cards.

Supported card(s)

6202, 6208V/6216V, 6208A, 6308V, 6308A, 7200, 7230, 7233, 7234, 7224, 7248, 7249, 7250/51, 7252, 7256, 7258, 7260, 7296, 7300A, 7348, 7396, 7432, 7433, 7434, 7442, 7443, 7444, 7452, 8554, 9111, 9112, 9113, 9114, 9116, 9118, 9221, 9222, 9223, 9524, 9812/10

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 Release_Card (U16 CardNumber)
```

Visual Basic

```
Release_Card (ByVal CardNumber As Integer) As
     Integer
```

Parameter(s)

CardNumber	ID of the card for release.

Return Code(s)

```
NoError
```

SetInitPattern

Description

Sets the state of the initial or the safetyout pattern. The initial pattern is sent to DO channel while power-on initializes, and the safetyout pattern is sent to DO channel when the watchdog timer overflows.

Supported card(s)

7442, 7444

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 SetInitPattern (U16 CardNumber, U8 patternID,
     U32 *pattern)
```

Visual Basic

```
SetInitPattern (ByVal CardNumber As Integer,
     ByVal patternID As Byte, pattern As Long) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
patternID	Valid pattern ID:

PCI-7442

`INIT_PTN_CH0`	The state of DO channel 0 at power-on.
`INIT_PTN_CH1`	The state of DO channel 1 at power-on.
`SAFTOUT_PTN_CH0`	The state of DO channel 0 while watchdog timer overflows.
`SAFTOUT_PTN_CH1`	The state of DO channel 1 while watchdog timer overflows.

PCI-7444

`INIT_PTN_CH0`	State of DO channel 0 at power-on.
`INIT_PTN_CH1`	State of DO channel 1 at power-on.
`INIT_PTN_CH2`	State of DO channel 2 at power-on.
`INIT_PTN_CH3`	State of DO channel 3 at power-on.
`SAFTOUT_PTN_CH0`	State of DO channel 0 while WDT overflows.
`SAFTOUT_PTN_CH1`	State of DO channel 1 while WDT overflows.
`SAFTOUT_PTN_CH2`	State of DO channel 2 while WDT overflows.
`SAFTOUT_PTN_CH3`	State of DO channel 3 while WDT overflows.

pattern State of the set pattern.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

SSI_SourceClear

Description

Disconnects all of the device signals from the SSI bus trigger lines.

Supported Cards

6202

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 SSI_SourceClear (USHORT wCardNumber)
```

Visual Basic

```
SSI_SourceClear (ByVal CardNumber As Integer) As
     Integer
```

Parameter

CardNumber	The card id of the card that want to perform this operation.

Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
InvalidCounter
```

SSI_SourceConn

Description

Connects a device to the specified SSI bus trigger line.

Supported card(s)

6202

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 SSI_SourceConn (USHORT wCardNumber, USHORT
     sigCode)
```

Visual Basic

```
SSI_SourceConn (ByVal CardNumber As Integer,
     ByVal sigCode As Integer) As Integer
```

Parameter

CardNumber	The card id of the card that want to perform this operation.
sigCode	The specified SSI signal code number of the device signal to be connected to the SSI bus trigger line. The direction of the connection is transmitted from the device to the SSI bus trigger line. The valid signal codes are as follows: P6202_SSI_AD_CONV P6202_SSI_AD_TRIG

Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
InvalidCounter
```

SSI_SourceDisConn

Description

Disconnects a device signal from the specified SSI bus trigger line.

Supported card(s)

6202

Syntax

Microsoft C/C++, Linux C/C++ and Borland C++

```
I16 SSI_SourceDisConn (USHORT wCardNumber, USHORT
     sigCode)
```

Visual Basic

```
SSI_SourceDisConn (ByVal CardNumber As Integer,
     ByVal sigCode As Integer) As Integer
```

Parameter

CardNumber	The card id of the card that want to perform this operation.
sigCode	The specified SSI signal code number of the device signal to be disconnected from the SSI bus trigger line. The valid signal codes are as follows: P6202_SSI_AD_CONV P6202_SSI_AD_TRIG

Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
InvalidCounter
```

WDT_Control

Description

Controls the watchdog timer

Supported card(s)

7260, 7442, 7444

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 WDT_Control (U16 CardNumber, U16 Ctr, U16
     action)
```

Visual Basic

```
WDT_Control (ByVal CardNumber As Integer, ByVal
     ctr As Integer, ByVal action As Integer) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Ctr	Counter number. Range is 0.
action	Operation code of the watchdog timer. Valid codes:

`WDT_DISARM`	Disable the watchdog timer.
`WDT_ARM`	Enable the watchdog timer.
`WDT_RESTART`	Restart the watchdog timer.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
Error Invalid Counter
ErrorInvalid Counter
```

WDT_Reload

Description

Reloads the watchdog timer counter value.

Supported card(s)

7442, 7444

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 WDT_Reload (U16 CardNumber, U16 Ctr, F32
     ovflowSec, F32 *actualSec)
```

Visual Basic

```
WDT_Reload (ByVal CardNumber As Integer, ByVal
     Ctr As Integer, ByVal ovflowSec As Single,
     actualSec As Single) As Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Ctr	Counter number. The counter number value is 0.
ovflowSec	Overflow time (timeout value) in seconds. Valid values: 0.0000001 to 429.
actualSec	Returns the actual overflow time (timeout value).

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidCounterValue
```

WDT_Setup

Description

Sets the overflow time of the watchdog timer.

Supported card(s)

7260, 7442, 7444

Syntax

Microsoft C/C++ and Borland C++

```
I16 WDT_Setup (U16 CardNumber, U16 Ctr, F32
     ovflowSec, F32 *actualSec, HANDLE *hEvent)
```

Linux C/C++

```
I16 WDT_Setup (U16 CardNumber, U16 Ctr, F32
     ovflowSec, F32 *actualSec, void
     (*event_handler)(int))
```

Visual Basic

```
WDT_Setup (ByVal CardNumber As Integer, ByVal Ctr
     As Integer, ByVal ovflowSec As Single,
     actualSec As Single, hEvent As Long) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Ctr	Counter number (7260/7442/7444), and enable/disable the SafetyOut capability while WDT overflows (7442/7444 only). The counter number value is 0.

NOTE:

For the PCI-7442/PCI-7444 SafetyOut capability, you can set the OR WDT_OVRFLOW_SAFETYOUT (0 | WDT_OVRFLOW_SAFETYOUT) to enable the Safety-Out capability. If the SafetyOut capability is enabled, the safety out patterns set are sent to the DO channels while the WDT overflows.

ovflowSec	Overflow time (timeout value) in seconds. Valid values:

PCI-7260	From 0.002 to 31.999.
PCI-7442	From 0.0000001 to 429

actualSec	Returns the actual overflow time (timeout value).

hEvent (Win32 only)

Watchdog overflow event handles returned. The status of a watchdog overflow event indicates whether the watchdog timer overflowed or not.

event_handler (Linux only)

Address of the event handler function. The PCIS-DASK calls this function when the specified WDT overflow event occurs. If you do not want to use the event handler, set this parameter to 0.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidCounterValue
```

WDT_Status

Description

Obtains the watchdog timer overflow status.

Supported card(s)

7260

Syntax

Microsoft C/C++, Linux C/C++, and Borland C++

```
I16 WDT_Status (U16 CardNumber, U16 Ctr, U32
     *status)
```

Visual Basic

```
WDT_Status (ByVal CardNumber As Integer, ByVal
     Ctr As Integer, ByVal status As Long) As
     Integer
```

Parameter(s)

CardNumber	ID of the card performing the operation.
Ctr	Counter number. Value for PCI-7260 is 0.
status	Tells whether a watchdog timer overflow occurred.

`0`	No watchdog timer overflow occurred.
`1`	A watchdog timer overflow occurred.

Return Code(s)

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidCounter
```

Appendix

Appendix A Source Codes

This appendix lists the status codes returned by PCIS-DASK, including the name and description.

Each PCIS-DASK function returns a status code that indicates whether the function was performed successfully. When a PCIS-DASK function returns a negative number, it means that an error occurred while executing the function.

Status Code	Status Name	Description
0	NoError	No error occurred.
-1	ErrorUnknownCardType	The CardType argument is not valid.
-2	ErrorInvalidCardNumber	The CardNumber argument is out of range (larger than 31).
-3	ErrorTooManyCardRegistered	32 cards have been registered.
-4	ErrorCardNotRegistered	No card registered as id CardNumber.
-5	ErrorFuncNotSupport	The function called is not supported by this type of card.
-6	ErrorInvalidIoChannel	The specified Channel or Port argument is out of range.
-7	ErrorInvalidAdRange	The specified analog input range is invalid.
-8	ErrorContIoNotAllowed	The specified continuous IO operation is not supported by this type of card.
-9	ErrorDiffRangeNotSupport	All the analog input ranges must be the same for multi-channel analog input.
-10	ErrorLastChannelNotZero	The channels for multi-channel analog input must end with or start from zero.
-11	ErrorChannelNotDescending	The channels for multi-channel analog input must be contiguous and in descending order.
-12	ErrorChannelNotAscending	The channels for multi-channel analog input must be contiguous and in ascending order.
-13	ErrorOpenDriverFailed	Failed to open the device driver.
-14	ErrorOpenEventFailed	Open event failed in device driver.
-15	ErrorTransferCountTooLarge	The size of transfer is larger than the size of initially allocated memory in driver.
-16	ErrorNotDoubleBufferMode	Double buffer mode is disabled.
-17	ErrorInvalidSampleRate	The specified sampling rate is out of range.
-18	ErrorInvalidCounterMode	The value of the Mode argument is invalid.
-19	ErrorInvalidCounter	The value of the Ctr argument is out of range.
-20	ErrorInvalidCounterState	The value of the State argument is out of range.
-21	ErrorInvalidBinBcdParam	The value of the BinBcd argument is invalid.
-22	ErrorBadCardType	The value of Card Type argument is invalid.
-23	ErrorInvalidDaRefVoltage	The value of DA reference voltage argument is invalid.
-24	ErrorAdTimeOut	AD operation timed-out.
-25	ErrorNoAsyncAI	Continuous AI is not set to asynchronous mode.
-26	ErrorNoAsyncAO	Continuous AO is not set to asynchronous mode.
-27	ErrorNoAsyncDI	Continuous DI is not set to asynchronous mode.
-28	ErrorNoAsyncDO	Continuous DO is not set to asynchronous mode.
-29	ErrorNotInputPort	The value of AI/DI port argument is invalid.

Table 3-1: Source Codes

Status Code	Status Name	Description
-30	ErrorNotOutputPort	The value of AO/DO argument is invalid.
-31	ErrorInvalidDioPort	The value of DI/O port argument is invalid.
-32	ErrorInvalidDioLine	The value of DI/O line argument is invalid.
-33	ErrorContIoActive	Continuous IO operation is not active.
-34	ErrorDblBufModeNotAllowed	Double Buffer mode is not allowed.
-35	ErrorConfigFailed	The specified function configuration failed.
-36	ErrorInvalidPortDirection	The value of DIO port direction argument is invalid.
-37	ErrorBeginThreadError	Failed to create thread.
-38	ErrorInvalidPortWidth	Port width setting for PCI-7300A/cPCI-7300A is not allowed.
-39	ErrorInvalidCtrSource	The clock source setting is invalid.
-40	ErrorOpenFile	Failed to open file
-41	ErrorAllocateMemory	The memory allocation failed.
-42	ErrorDaVoltageOutOfRange	The value of DA voltage argument is out of range.
-50	ErrorInvalidCounterValue	The value of count for a counter is invalid.
-60	ErrorInvalidEventHandle	The event handle is invalid.
-61	ErrorNoMessageAvailable	No event message can be added.
-62	ErrorEventMessgaeNotAdded	The specified event message does not exist.
-63	ErrorCalibrationTimeOut	Auto-calibration has timed-out.
-64	ErrorUndefinedParameter	Parameter(s) is not defined.
-65	ErrorInvalidBufferID	Buffer ID is invalid.
-201	ErrorConfigIoctl	The configuration API failed.
-202	ErrorAsyncSetIoctl	The async. mode API failed.
-203	ErrorDBSetIoctl	The double-buffer setting API failed.
-204	ErrorDBHalfReadyIoctl	The half-ready API failed.
-205	ErrorContOPIoctl	The continuous data acquisition API failed.
-206	ErrorContStatusIoctl	continuous data acquisition status API setting failed.
-207	ErrorPIOIoctl	The polling data API failed.
-208	ErrorDIntSetIoctl	The dual-interrupt setting API failed.
-209	ErrorWaitEvtIoctl	The wait event API failed.
-210	ErrorOpenEvtIoctl	The open event API failed.
-211	ErrorCOSIntSetIoctl	The COS interrupt setting API failed.
-212	ErrorMemMapIoctl	The memory mapping API failed.
-213	ErrorMemUMapSetIoctl	The memory unmapping API failed.
-214	ErrorCTRIoctl	The counter API failed.
-215	ErrorGetResIoctl	The resource getting API failed.
-216	ErrorCalIoctl	The calibration API failed.

Table 3-1: Source Codes

Appendix B AI Range Codes

The table below lists the analog input range of NuDAQ PCI-bus cards.

AD_B_10_V	Bipolar -10V to +10V
AD_B_5_V	Bipolar -5V to +5V
AD_B_2_5_V	Bipolar -2.5V to +2.5V
AD_B_1_25_V	Bipolar -1.25V to +1.25V
AD_B_0_625_V	Bipolar -0.625V to +0.625V
AD_B_0_3125_V	Bipolar -0.3125V to +0.3125V
AD_B_0_5_V	Bipolar -0.5V to +0.5V
AD_B_0_05_V	Bipolar -0.05V to +0.05V
AD_B_0_005_V	Bipolar -0.005V to +0.005V
AD_B_1_V	Bipolar -1V to +1V
AD_B_0_1_V	Bipolar -0.1V to +0.1V
AD_B_0_01_V	Bipolar -0.01V to +0.01V
AD_B_0_001_V	Bipolar -0.01V to +0.001V
AD_B_2_V	Bipolar -2V to +2V
AD_B_0_2_V	Bipolar -0.2V to +0.2V
AD_U_20_V	Unipolar 0 to +20V
AD_U_10_V	Unipolar 0 to +10V
AD_U_5_V	Unipolar 0 to +5V
AD_U_2_5_V	Unipolar 0 to +2.5V
AD_U_1_25_V	Unipolar 0 to +1.25V
AD_U_1_V	Unipolar 0 to +1V
AD_U_0_1_V	Unipolar 0 to +0.1V
AD_U_0_01_V	Unipolar 0 to +0.01V
AD_U_0_001_V	Unipolar 0 to +0.001V
AD_U_2_V	Unipolar 0 to +2V

Table 3-2: AI Range Codes

Valid values for each card:

PCI-9111 DG/HR	AD_B_10_V, AD_B_5_V, AD_B_2_5_V, AD_B_1_25_V, AD_B_0_625_V
PCI-9112/cPCI-9112	AD_B_10_V, AD_B_5_V, AD_B_2_5_V, AD_B_1_25_V, AD_B_0_625_V, AD_U_10_V, AD_U_5_V, AD_U_2_5_V, AD_U_1_25_V
PCI-9113	AD_B_10_V, AD_B_1_V, AD_B_0_1_V, AD_B_5_V, AD_B_0_5_V, AD_B_0_05_V, AD_U_10_V, AD_U_1_V, AD_U_0_1_V
PCI-9114 HG	AD_B_10_V, AD_B_1_V, AD_B_0_1_V, AD_B_0_01_V
PCI-9114 DG	AD_B_10_V, AD_B_5_V, AD_B_2_5_V, AD_B_1_25_V
cPCI-9116	AD_B_5_V, AD_B_2_5_V, AD_B_1_25_V, AD_B_0_625_V, AD_U_10_V, AD_U_5_V, AD_U_2_5_V, AD_U_1_25_V
PCI-9118 DG/HR	AD_B_5_V, AD_B_2_5_V, AD_B_1_25_V, AD_B_0_625_V, AD_U_10_V, AD_U_5_V, AD_U_2_5_V, AD_U_1_25_V
PCI-9118 HG	AD_B_5_V, AD_B_0_5_V, AD_B_0_05_V, AD_B_0_005_V, AD_U_10_V, AD_U_1_V, AD_U_0_1_V, AD_U_0_01_V
PCI-9221	AD_B_5_V, AD_B_1_V, AD_B_0_5_V, AD_B_0_2_V
PCI-9524	Load Cell Group 0 General Purpose Group AD_B_10_V, AD_B_5_V, AD_B_2_5_V, AD_B_1_25_V
PCI-9812/10	AD_B_1_V, AD_B_5_V

Table 3-3: Card Valid Values

Appendix C AI Data Format

This appendix lists the AI data format for the cards performing analog input operation, as well as the calculation methods to retrieve the A/D converted data and the channel where the data read from.

Card Type	Data Format	AI type	Value calculation*

PCI-9111DG	Every 16-bit signed integer data: D11 D10 D9...D1 D0 C3 C2 C1 C0, where D11, D10,... D0: A/D converted data and C3, C2, C1, C0: converted channel no.	One-Shot AI
		Continuous AI
			CH# = OD & 0x0F
			ND = OD >>4 or
			ND = OD/16

PCI-9111HR	Every 16-bit signed integer data: D15 D14 D13... D1 D0 where D15, D14,..., D0: A/D converted data	One-Shot AI Continuous AI	ND = OD

PCI-9112/
cPCI9112
	Every 16-bit unsigned integer data: D11 D10 D9... D1 D0 C3 C2 C1 C0
	where D11, D10,..., D0: A/D converted data
	C3, C2, C1, C0: converted channel no.
		One-Shot AI
		Continuous AI
			CH# = OD & 0x0F
			ND = OD >>4 or
			ND = OD/16

PCI-9113	Every 16-bit unsigned integer data (including 12-bit unsigned A/D data): B15…B12 D11 D10... D1 D0 where D11, D10,..., D0: A/D converted data B15 ~ B12: unused	One-Shot AI	ND = OD & 0x0FFF

Card Type	Data Format	AI type	Value calculation*
PCI-9113	Every 32-bit unsigned integer data (including 12-bit unsigned A/D data): B31… B21 C4 C3 C2 C1 C0 B15 … B12 D11 D10...D1 D0 where D11, D10,..., D0: A/D converted data C3, C2, C1, C0: converted channel no. B31 ~ B21 & B15 ~ B12: unused	Continuous AI	CH# = (OD >>16) & 0x1F ND = OD & 0x0FFF
PCI-9114	Every 16-bit signed integer data: D15 D14... D1 D0 where D15, D14,..., D0: A/D converted data	One-Shot AI	ND = OD
PCI-9114	Every 32-bit unsigned integer data (including 16-bit signed A/D data): B31…B21 C4 C3 C2 C1 C0 D15 D14...D1 D0 where D15, D14,..., D0: A/D converted data C3, C2, C1, C0: converted channel no. B31 ~ B21: unused	Continuous AI	CH# = (OD >>16) & 0x1F ND = OD & 0xFFFF
cPCI-9116	Every 16-bit signed integer data: D15 D14 D13...D1 D0 where D15, D14,..., D0: A/D converted data	One-Shot AI Continuous AI	ND = OD
PCI-9118HR	Every 16-bit signed integer data: D15 D14 D13...D1 D0 where D15, D14,..., D0: A/D converted data	One-Shot AI Continuous AI	ND = OD

PCI-9118DG/HG	Every 16-bit unsigned integer data: D11 D10 D9...D0 C3 C2 C1 C0
	where D11, D10,..., D0: A/D converted data
	C3, C2, C1, C0: converted channel no.
		One-Shot AI
		Continuous AI
			CH# = OD & 0x0F
			ND = OD >>4 or
			ND = OD/16

Card Type	Data Format	AI type	Value calculation*
PCI-9221	Every 16-bit unsigned integer data: D15 D14 D13...D1 D0 where D15, D14,..., D1, D0: A/D converted data	One-Shot AI Continuous AI	ND = OD
PCI-9222 PCI-9223	Every 16-bit unsigned integer data: D15 D14 D13...D1 D0 where D15, D14,..., D1, D0: A/D converted data Note: Continuous AI with Gated Trigger Mode Every 32-bit unsigned integer data: b15 b14 ... b0 D15 D14 D13...D1 D0 where D15, D14,..., D1, D0: A/D converted data b1: Separation flag b15, ..., b2 and b0: Not used	One-Shot AI Continuous AI	ND = OD&0xffff

PCI-9524	Every 32-bit signed integer data: D23 D22...D1 D0 b7 b6...b1 b0
	where D23...D0: A/D converted data
	b7...b4: channel number
	b3...b2: Gain
	b1: DSP Flushed
	b0: Data Refreshed (Polling mode only)
		One-Shot AI
		Continuous AI
			ND = ((OD>>8)-Residual_offset)*Residual_scaling
			or
			ND = ((OD/8)-Residual_offset)*Residual_scaling
			
			Residual_offset and Residual_scaling can be obtained by calling PCI9524_Acquire_AD_CalConst()

PCI-9812	Every 16-bit signed integer data: D11 D10 D9...D1 D0 b3 b2 b1 b0
	where D11, D10,..., D0: A/D converted data
	b2, b1, b0: Digital Input data.
	b3: trigger detection flag
		Continuous AI	ND = OD >>4 or
			ND = OD/16

Card Type	Data Format	AI type	Value calculation*
PCI-9810/ cPCI9810	Every 16-bit signed integer data: D9 D8 D7...D1 D0 b5 b4 b3 b2 b1 b0 where D9, D8,..., D0: A/D converted data b2, b1, b0: Digital Input data. b3: trigger detection flag	Continuous AI	ND = OD >>6 or ND = OD/64

* channel no. (CH#) * A/D converted data (ND) * Value returned from AI function (OD)

Appendix D Data File Format

This appendix describes the file format of the data files generated by the functions performing continuous data acquisition followed by storing the data to disk.

The data file includes three parts, Header, ChannelRange (optional), and Data block. The file structure is shown below:

Header
ChannelRange (Optional)
DAQ data

ChannelCompensation (PCI-9524 only)

Header
ChannelRange (Optional)
ChannelCompensation (PCI-9524 only)
DAQ data

Header

The header part records the information related to the stored data and has 60 bytes of length. The data structure of the file header is listed in the table:

Header Total Length: 60 bytes			
Elements	**Type**	**Size (bytes)**	**Comments**
ID	char	10	file ID ex. ADLinkDAQ1
card_type	short	2	card Type ex. PCI-7250, PCI-9112
num_of_channel	short	2	number of scanned channels ex. 1, 2
channel_no	unsigned char	1	channel number where the data read from (only available as the num_of_channel is 1) ex. 0, 1
num_of_scan	long	4	the number of scan for each channel (total count / num_of_channel)
data_width	short	2	the data width 0: 8 bits, 1: 16 bits, 2: 32 bits

Header Total Length: 60 bytes			
Elements	**Type**	**Size (bytes)**	**Comments**
channel_order	short	2	the channel scanned sequence 0: normal (ex. 0-1-2-3) 1: reverse (ex. 3-2-1-0) 2: custom* (ex. 0, 1, 3)
ad_range	short	2	the AI range code Please refer to Appendix B ex. 0 (AD_B_5V)
scan_rate	double	8	The scanning rate of each channel (total sampling rate/num_of_channel)
num_of_channel_range	short	2	The number of ChannelRange* structure
start_date	char	8	The starting date of data acquisition ex. 12/31/99
start_time	char	8	The starting time of data acquisition ex. 18:30:25
start_millisec	char	3	The starting millisecond of data acquisition ex. 360
reserved	char	6	not used

* If the num_of_channel_range is 0, the ChannelRange block won't be included in the data file.

* The channel_order is set to "custom" only when the card supports variant channel scanning order.

ChannelRange

The ChannelRange part records the channel number and data range information related to the stored data. This part consists of several channel and range units. The length of each unit is 2 bytes. The total length depends on the value of num_of_channel_range (one element of the file header) and is calculated with this formula:

Total Length = 2 * num_of_channel_range bytes

The data structure of each ChannelRange unit is listed below:

ChannelRange Unit Length: 2 bytes			
Elements	**Type**	**Size (bytes)**	**Comments**
channel	char	1	scanned channel number ex. 0, 1
range	char	1	the AI range code of channel Please refer to Appendix B ex. 0 (AD_B_5V)

ChannelCompensation

The ChannelCompensation part records the software compensation values related the stored data.

Now, this part is only valid for PCI-9524. For PCI-9524, the stored raw data should be software compensated to translate the related voltage. PCI-9524 has two software compensation values, Residual Offset and Residual Scaling, and the two values depend on different ADC Speed and ADC Gain.

The total length of this part depends on the value of num_of_channels and is calculated with this formula:

Total Length = 2*8*num_of_channels bytes

The data structure of each ChannelCompensation unit is listed below:

Elements	Type	Size (Bytes)
Residual offset	Double	8
Residual scaling	Double	8

Data Block

The data is written to file in a 16-bit binary format, with the lower byte first (little endian). For example, the value 0x1234 is written to disk with 34 first followed by 12. The total length of the data block depends on the data width and the total data count.

The file is written in Binary format and may not be read by normal text editor. You can use any binary file editor to view it or the functions used for reading files (such as fread) to get the file information and data value. PCIS-DASK provides the DAQCvt utility to convert the binary file. Refer to the PCIS-DASK user manual for details.

DAQCvt translates the information stored in the header part and the ChannelRange part, then displays the corresponding information in the **Input File** frame of DAQCvt main window. After setting the properties (File Path, Format, …etc) of the converted file and after clicking the **Start Convert** button, DAQCvt gets rid of header and ChannelRange parts and converts the data in data block according to the card type and the data width. DAQCvt also writes the converted data to a disk and lets you use any text editor or Excel to view or analyze the accessed data.

Appendix E Function Support

This appendix shows which data acquisition hardware each PCIS-DASK function supports.

Multi-Function DAQ/AI Devices

Card>	PCI-9111	PCI-9112	PCI-9113	PCI-9114	PCI-9116	PCI-9118	PCI-9221	PCI-9222	PCI-9223
Function									
AI_9111_Config	✓								
AI_9112_Config		✓							
AI_9113_Config			✓						
AI_9114_Config				✓					
AI_9116_Config					✓				
AI_9116_CounterInterval					✓				
AI_9118_Config						✓			
AI_9221_Config							✓		
AI_9221_CounterInterval							✓		
AI_9222_Config								✓	
AI_9222_CounterInterval								✓	
AI_9223_Config									✓
AI_9223_CounterInterval									✓
AI_AsyncCheck	✓	✓	✓	✓	✓	✓	✓	✓	✓
AI_AsyncClear	✓	✓	✓	✓	✓	✓	✓	✓	✓
AI_AsyncDblBufferHalfReady	✓	✓	✓	✓	✓	✓	✓	✓	✓
AI_AsyncDblBufferHandled							✓	✓	✓
AI_AsyncDblBufferMode	✓	✓	✓	✓	✓	✓	✓	✓	✓
AI_AsyncDblBufferOverrun	✓	✓	✓	✓	✓	✓	✓	✓	✓
AI_AsyncDblBufferToFile							✓	✓	✓
AI_AsyncDblBufferTransfer	✓	✓	✓	✓	✓	✓			
AI_AsyncReTrigNextReady								✓	✓
AI_ContBufferReset							✓	✓	✓
AI_ContBufferSetup							✓	✓	✓
AI_ContReadChannel	✓	✓	✓	✓	✓	✓	✓	✓	✓

Card> Function	PCI-9111	PCI-9112	PCI-9113	PCI-9114	PCI-9116	PCI-9118	PCI-9221	PCI-9222	PCI-9223
AI_ContReadChannelToFile	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
AI_ContReadMultiChannels					ˇ	ˇ	ˇ	ˇ	ˇ
AI_ContReadMultiChannelsToFile					ˇ	ˇ	ˇ	ˇ	ˇ
AI_ContScanChannels	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
AI_ContScanChannelsToFile	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
AI_ContStatus	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
AI_ContVScale	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
AI_EventCallBack	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
AI_GetView	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ			
AI_InitialMemoryAllocated	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
AI_ReadChannel	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
AI_ReadMultiChannels							ˇ	ˇ	ˇ
AI_ScanReadChannels							ˇ	ˇ	ˇ
AI_SetTimeOut	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
AI_VReadChannel	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
AI_VoltScale	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
AO_9111_Config	ˇ								
AO_9112_Config		ˇ							
AO_9222_Config								ˇ	
AO_9223_Config									ˇ
AO_AsyncCheck								ˇ	ˇ
AO_AsyncClear								ˇ	ˇ
AO_AsyncDblBufferHalfReady								ˇ	ˇ
AO_AsyncDblBufferMode								ˇ	ˇ
AO_ContBufferCompose								ˇ	ˇ
AO_ContBufferReset								ˇ	ˇ
AO_ContBufferSetup								ˇ	ˇ
AO_ContStatus								ˇ	ˇ
AO_ContWriteChannel								ˇ	ˇ
AO_ContWriteMultiChannels								ˇ	ˇ
AO_EventCallBack								ˇ	ˇ

Card> Function	PCI-9111	PCI-9112	PCI-9113	PCI-9114	PCI-9116	PCI-9118	PCI-9221	PCI-9222	PCI-9223
AO_InitialMemoryAllocated								ˇ	ˇ
AO_SetTimeOut								ˇ	ˇ
AO_VoltScale	ˇ	ˇ					ˇ	ˇ	ˇ
AO_VWriteChannel	ˇ	ˇ					ˇ	ˇ	ˇ
Function									
CTR_Read	ˇ	ˇ	ˇ	ˇ		ˇ			
CTR_Reset	ˇ	ˇ	ˇ	ˇ		ˇ			
CTR_Setup	ˇ	ˇ	ˇ	ˇ		ˇ			
CTR_Update	ˇ	ˇ	ˇ	ˇ		ˇ			
DI_9222_Config								ˇ	
DI_9223_Config									ˇ
DI_AsyncCheck								ˇ	ˇ
DI_AsyncClear								ˇ	ˇ
DI_AsyncDblBufferHalfReady								ˇ	ˇ
DI_AsyncDblBufferHandled								ˇ	ˇ
DI_AsyncDblBufferMode								ˇ	ˇ
DI_AsyncDblBufferOverrun								ˇ	ˇ
DI_AsyncDblBufferToFile								ˇ	ˇ
DI_AsyncReTrigNextReady								ˇ	ˇ
DI_ContBufferReset								ˇ	ˇ
DI_ContBufferSetup								ˇ	ˇ
DI_ContReadPort								ˇ	ˇ
DI_ContReadPortToFile								ˇ	ˇ
DI_ContStatus								ˇ	ˇ
DI_EventCallBack								ˇ	ˇ
DI_InitialMemoryAllocated								ˇ	ˇ
DI_ReadLine	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
DI_ReadPort	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
DI_SetTimeOut								ˇ	ˇ
DO_9222_Config								ˇ	ˇ
DO_9223_Config								ˇ	ˇ

Card> / Function	PCI-9111	PCI-9112	PCI-9113	PCI-9114	PCI-9116	PCI-9118	PCI-9221	PCI-9222	PCI-9223
DO_AsyncCheck								ˇ	ˇ
DO_AsyncClear								ˇ	ˇ
DO_ContBufferReset								ˇ	ˇ
DO_ContBufferSetup								ˇ	ˇ
DO_ContStatus								ˇ	ˇ
DO_ContWritePort								ˇ	ˇ
DO_EventCallBack								ˇ	ˇ
DO_InitialMemoryAllocated								ˇ	ˇ
DO_ReadLine	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
DO_ReadPort	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
DO_SetTimeOut								ˇ	ˇ
DO_WriteLine	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
DO_WritePort	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
GCTR_Reset					ˇ				
GCTR_Read					ˇ				
GCTR_Setup					ˇ				
GetActualRate	ˇ	ˇ	ˇ	ˇ		ˇ	ˇ	ˇ	ˇ
GetBaseAddr	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
GetCardIndexFromID	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
GetCardType	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
GetLCRAddr	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
GPTC_Clear							ˇ	ˇ	ˇ
GPTC_Control							ˇ	ˇ	ˇ
GPTC_EventSetup								ˇ	ˇ
GPTC_EventCallBack								ˇ	ˇ
GPTC_Read							ˇ	ˇ	ˇ
GPTC_Setup							ˇ	ˇ	ˇ
GPTC_Status							ˇ	ˇ	ˇ
PCI_DB_Auto_Calibration_ALL							ˇ	ˇ	ˇ
PCI_EEPROM_CAL_Constant_Update							ˇ	ˇ	ˇ
PCI_Load_CAL_Data							ˇ	ˇ	ˇ

Card> / Function	PCI-9111	PCI-9112	PCI-9113	PCI-9114	PCI-9116	PCI-9118	PCI-9221	PCI-9222	PCI-9223
Register_Card	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
Release_Card	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ

Load Cell Input Devices

Card>	PCI-9524
Function	
AI_9524_Config	ˇ
AI_9524_PollConfig	ˇ
AI_9524_SetDSP	ˇ
AI_AsyncCheck	ˇ
AI_AsyncClear	ˇ
AI_AsyncDblBufferHalfReady	ˇ
AI_AsyncDblBufferHandled	ˇ
AI_AsyncDblBufferMode	ˇ
AI_AsyncDblBufferOverrun	ˇ
AI_AsyncDblBufferToFile	ˇ
AI_ContBufferReset	ˇ
AI_ContBufferSetup	ˇ
AI_ContReadChannel	ˇ
AI_ContReadChannelToFile	ˇ
AI_ContScanChannels	ˇ
AI_ContScanChannelsToFile	ˇ
AI_ContStatus	ˇ
AI_ContVScale	ˇ
AI_EventCallBack	ˇ
AI_InitialMemoryAllocated	ˇ
AI_ReadChannel32	ˇ
AI_ScanReadChannels32	ˇ
AI_SetTimeOut	ˇ
AI_VReadChannel	ˇ
AI_VoltScale32	ˇ
AO_SimuVWriteChannel	ˇ
AO_SimuWriteChannel	ˇ
AO_VoltScale	ˇ
AO_VWriteChannel	ˇ

Card>	PCI-9524
Function	
AO_WriteChannel	ˇ
DI_ReadLine	ˇ
DI_ReadPort	ˇ
DO_ReadLine	ˇ
DO_ReadPort	ˇ
DO_WriteLine	ˇ
DO_WritePort	ˇ
GetActualRate_9524	ˇ
GetBaseAddr	ˇ
GetCardIndexFromID	ˇ
GetCardType	ˇ
GetLCRAddr	ˇ
GPTC_9524_PG_Config	ˇ
GPTC_Clear	ˇ
GPTC_Control	ˇ
GPTC_Read	ˇ
GPTC_Setup	ˇ
PCI9524_Acquire_AD_CalConst	ˇ
PCI9524_Acquire_DA_CalConst	ˇ
PCI_DB_Auto_Calibration_ALL	ˇ
PCI_EEPROM_CAL_Constant_Update	ˇ
PCI_Load_CAL_Data	ˇ
Register_Card	ˇ
Release_Card	ˇ

Digitizer Devices

Card>	PCI-9812/10
Function	
AI_9812_Config	ˇ
AI_9812_SetDiv	ˇ
AI_AsyncCheck	ˇ
AI_AsyncClear	ˇ
AI_AsyncDblBufferHalfReady	ˇ
AI_AsyncDblBufferMode	ˇ
AI_AsyncDblBufferOverrun	ˇ
AI_AsyncDblBufferTransfer	ˇ
AI_ContReadChannel	ˇ
AI_ContReadChannelToFile	ˇ
AI_ContScanChannels	ˇ
AI_ContScanChannelsToFile	ˇ
AI_ContStatus	ˇ
AI_ContVScale	ˇ
AI_EventCallBack	ˇ
AI_GetView	ˇ
AI_InitialMemoryAllocated	ˇ
AI_ReadChannel	ˇ
AI_SetTimeOut	ˇ
AI_VReadChannel	ˇ
AI_VoltScale	ˇ
AO_VoltScale	ˇ
AO_VWriteChannel	ˇ
AO_WriteChannel	ˇ
DI_ReadLine	ˇ
DI_ReadPort	ˇ
DO_ReadLine	ˇ
DO_ReadPort	ˇ

Card>	PCI-9812/10
Function	
DO_WriteLine	ˇ
DO_WritePort	ˇ
GetActualRate	ˇ
GetBaseAddr	ˇ
GetCardIndexFromID	ˇ
GetCardType	ˇ
GetLCRAddr	ˇ
Register_Card	ˇ
Release_Card	ˇ

General Purpose/Isolated AO Devices

Card>	PCI-6208A	PCI-6208V/16V	PCI-6308A	PCI-6308V
Function				
AO_6208A_Config	ˇ			
AO_6308A_Config			ˇ	
AO_6308V_Config				ˇ
AO_SimuVWriteChannel			ˇ	ˇ
AO_SimuWriteChannel			ˇ	ˇ
AO_VoltScale	ˇ	ˇ	ˇ	ˇ
AO_VWriteChannel	ˇ	ˇ	ˇ	ˇ
AO_WriteChannel	ˇ	ˇ	ˇ	ˇ
DI_ReadLine	ˇ	ˇ	ˇ	ˇ
DI_ReadPort	ˇ	ˇ	ˇ	ˇ
DO_ReadLine	ˇ	ˇ	ˇ	ˇ
DO_ReadPort	ˇ	ˇ	ˇ	ˇ
DO_WriteLine	ˇ	ˇ	ˇ	ˇ
DO_WritePort	ˇ	ˇ	ˇ	ˇ
GetBaseAddr	ˇ	ˇ	ˇ	ˇ
GetCardIndexFromID	ˇ	ˇ	ˇ	ˇ
GetCardType	ˇ	ˇ	ˇ	ˇ
GetLCRAddr	ˇ	ˇ	ˇ	ˇ
Register_Card	ˇ	ˇ	ˇ	ˇ
Release_Card	ˇ	ˇ	ˇ	ˇ

High Performance AO Devices

Card> Function	PCI-6202
AO_6202_Config	ˇ
AO_AsyncCheck	ˇ
AO_AsyncClear	ˇ
AO_AsyncDblBufferHalfReady	ˇ
AO_AsyncDblBufferMode	ˇ
AO_ContBufferCompose	ˇ
AO_ContBufferReset	ˇ
AO_ContBufferSetup	ˇ
AO_ContStatus	ˇ
AO_ContWriteChannel	ˇ
AO_ContWriteMultiChannels	ˇ
AO_EventCallBack	ˇ
AO_InitialMemoryAllocated	ˇ
AO_SetTimeOut	ˇ
AO_VoltScale	ˇ
AO_VWriteChannel	ˇ
AO_WriteChannel	ˇ
DI_ReadLine	ˇ
DI_ReadPort	ˇ
DO_ReadLine	ˇ
DO_ReadPort	ˇ
DO_WriteLine	ˇ
DO_WritePort	ˇ
GetBaseAddr	ˇ
GetCardIndexFromID	ˇ
GetCardType	ˇ
GetLCRAddr	ˇ
GPTC_Clear	ˇ
GPTC_Control	ˇ

Card> Function	PCI-6202
GPTC_EventSetup	ˇ
GPTC_EventCallBack	ˇ
GPTC_Read	ˇ
GPTC_Setup	ˇ
GPTC_Status	ˇ
PWM_Output	ˇ
PWM_Stop	ˇ
Register_Card	ˇ
Release_Card	ˇ
SSI_SourceClear	ˇ
SSI_SourceConn	ˇ
SSI_SourceDisConn	ˇ

Relay Output & Isolated DI Devices

Card>	PCI-7250/51/52	PCI-7256	PCI-7258	PCI-7260
Function				
DI_ReadLine	ˇ	ˇ	ˇ	ˇ
DI_ReadPort	ˇ	ˇ	ˇ	ˇ
DIO_GetCOSLatchData		ˇ		ˇ
DIO_INT_Event_Message		ˇ	ˇ	ˇ
DIO_INT1_EventMessage		ˇ	ˇ	ˇ
DIO_INT2_EventMessage		ˇ	ˇ	ˇ
DIO_SetCOSInterrupt		ˇ		ˇ
DIO_SetDualInterrupt		ˇ	ˇ	ˇ
DO_ReadLine	ˇ	ˇ	ˇ	ˇ
DO_ReadPort	ˇ	ˇ	ˇ	ˇ
DO_WriteLine	ˇ	ˇ	ˇ	ˇ
DO_WritePort	ˇ	ˇ	ˇ	ˇ
EMGShutDownControl				ˇ
EMGShutDownStatus				ˇ
GetBaseAddr	ˇ	ˇ	ˇ	ˇ
GetCardIndexFromID	ˇ	ˇ	ˇ	ˇ
GetCardType	ˇ	ˇ	ˇ	ˇ
GetInitPattern				ˇ
GetLCRAddr	ˇ	ˇ	ˇ	ˇ
IdentifyLED_Control				ˇ
Register_Card	ˇ	ˇ	ˇ	ˇ
Release_Card	ˇ	ˇ	ˇ	ˇ
WDT_Control				ˇ
WDT_Setup				ˇ
WDT_Status				ˇ

TTL DIO Devices

Card> Function	PCI-7248/49/96	PCI-7348/96
CTR_Read	ˇ	ˇ
CTR_Reset	ˇ	ˇ
CTR_Setup	ˇ	ˇ
CTR_Update	ˇ	ˇ
DI_ReadLine	ˇ	ˇ
DI_ReadPort	ˇ	ˇ
DIO_INT_Event_Message	ˇ	ˇ
DIO_INT1_EventMessage	ˇ	ˇ
DIO_INT2_EventMessage	ˇ	ˇ
DIO_PortConfig	ˇ	ˇ
DIO_SetCOSInterrupt		ˇ
DIO_SetDualInterrupt	ˇ	ˇ
DO_ReadLine	ˇ	ˇ
DO_ReadPort	ˇ	ˇ
DO_WriteLine	ˇ	ˇ
DO_WritePort	ˇ	ˇ
GetBaseAddr	ˇ	ˇ
GetCardIndexFromID	ˇ	ˇ
GetCardType	ˇ	ˇ
GetLCRAddr	ˇ	ˇ
Register_Card	ˇ	ˇ
Release_Card	ˇ	ˇ

Isolated DIO/High Density Isolated DIO Devices

Card>	PCI-7230	PCI-7233	PCI-7234	PCI-7432	PCI-7433	PCI-7434	PCI-7442	PCI-7443	PCI-7444	PCI-7452
Function										
DI_ReadLine	ˇ	ˇ		ˇ	ˇ		ˇ	ˇ	ˇ	ˇ
DI_ReadPort	ˇ	ˇ		ˇ	ˇ		ˇ	ˇ	ˇ	ˇ
DIO_GetCOSLatchData32							ˇ	ˇ		ˇ
DIO_INT_Event_Message	ˇ	ˇ		ˇ	ˇ		ˇ	ˇ	ˇ	ˇ
DIO_INT1_EventMessage	ˇ	ˇ		ˇ	ˇ		ˇ	ˇ		ˇ
DIO_INT2_EventMessage	ˇ	ˇ		ˇ	ˇ		ˇ		ˇ	
DIO_LineConfig							ˇ	ˇ	ˇ	
DIO_LinesConfig							ˇ	ˇ	ˇ	
DIO_PortConfig							ˇ	ˇ	ˇ	
DIO_SetCOSInterrupt32							ˇ	ˇ		ˇ
DIO_SetDualInterrupt	ˇ	ˇ		ˇ	ˇ		ˇ	ˇ	ˇ	ˇ
DO_ReadLine	ˇ		ˇ	ˇ		ˇ	ˇ		ˇ	ˇ
DO_ReadPort	ˇ		ˇ	ˇ		ˇ	ˇ		ˇ	ˇ
DO_SimuWritePort							ˇ		ˇ	
DO_WriteLine	ˇ		ˇ	ˇ		ˇ	ˇ	ˇ	ˇ	ˇ
DO_WritePort	ˇ		ˇ	ˇ		ˇ	ˇ	ˇ	ˇ	ˇ
GetBaseAddr	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
GetCardIndexFromID	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
GetCardType	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
GetInitPattern							ˇ		ˇ	
GetLCRAddr	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
HotResetHoldControl							ˇ		ˇ	
HotResetHoldStatus							ˇ		ˇ	
Register_Card	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
Release_Card	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ	ˇ
SetInitPattern							ˇ		ˇ	
WDT_Control							ˇ		ˇ	
WDT_Reload							ˇ		ˇ	
WDT_Setup							ˇ		ˇ	

High Speed DIO Device

Card>	PCI-7200	PCI-7300A Rev A	PCI-7300A Rev B
Function			
DI_7200_Config	ˇ		
DI_7300A_Config		ˇ	
DI_7300B_Config			ˇ
DI_AsyncCheck	ˇ	ˇ	ˇ
DI_AsyncClear	ˇ	ˇ	ˇ
DI_AsyncDblBufferHalfReady	ˇ		
DI_AsyncDblBufferMode	ˇ		
DI_AsyncDblBufferOverrun	ˇ	ˇ	ˇ
DI_AsyncDblBufferTransfer	ˇ		
DI_AsyncMultiBufferNextReady		ˇ	ˇ
DI_ContMultiBufferSetup		ˇ	ˇ
DI_ContMultiBufferStart		ˇ	ˇ
DI_ContReadPort	ˇ	ˇ	ˇ
DI_ContReadPortToFile	ˇ	ˇ	ˇ
DI_ContStatus	ˇ	ˇ	ˇ
DI_EventCallBack	ˇ	ˇ	ˇ
DI_GetView	ˇ		
DI_InitialMemoryAllocated	ˇ	ˇ	ˇ
DI_ReadLine	ˇ	ˇ	ˇ
DI_ReadPort	ˇ	ˇ	ˇ
DIO_7300SetInterrupt		ˇ	ˇ
DIO_AUXDI_EventMessage		ˇ	ˇ
DIO_T2_EventMessage		ˇ	ˇ
DO_7200_Config	ˇ		
DO_7300A_Config		ˇ	
DO_7300B_Config			ˇ
DO_AsyncCheck	ˇ	ˇ	ˇ

Card>			
Function	PCI-7200	PCI-7300A Rev A	PCI-7300A Rev B
DO_AsyncClear	ˇ	ˇ	ˇ
DO_AsyncMultiBufferNextReady			ˇ
DO_ContMultiBufferSetup			ˇ
DO_ContMultiBufferStart			ˇ
DO_ContStatus	ˇ	ˇ	ˇ
DO_ContWritePort	ˇ	ˇ	ˇ
DO_EventCallBack	ˇ	ˇ	ˇ
DO_GetView	ˇ		
DO_InitialMemoryAllocated	ˇ	ˇ	ˇ
DO_PGStart			ˇ
DO_PGStop			ˇ
DO_ReadLine	ˇ	ˇ	ˇ
DO_ReadPort	ˇ	ˇ	ˇ
DO_WriteLine	ˇ	ˇ	ˇ
DO_WritePort	ˇ	ˇ	ˇ
GetActualRate	ˇ	ˇ	ˇ
GetBaseAddr	ˇ	ˇ	ˇ
GetCardIndexFromID	ˇ	ˇ	ˇ
GetCardType	ˇ	ˇ	ˇ
GetLCRAddr	ˇ	ˇ	ˇ
Register_Card	ˇ	ˇ	ˇ
Release_Card	ˇ	ˇ	ˇ

Timer Counter Devices

Card>	PCI-8554
Function	
CTR_8554_CK1_Config	ˇ
CTR_8554_ClkSrc_Config	ˇ
CTR_8554_Debounce_Config	ˇ
CTR_Clear	ˇ
CTR_Read	ˇ
CTR_Setup	ˇ
CTR_Status	ˇ
CTR_Update	ˇ
DI_ReadLine	ˇ
DI_ReadPort	ˇ
DO_ReadLine	ˇ
DO_ReadPort	ˇ
DO_WriteLine	ˇ
DO_WritePort	ˇ
GetBaseAddr	ˇ
GetCardIndexFromID	ˇ
GetCardType	ˇ
GetLCRAddr	ˇ
Register_Card	ˇ
Release_Card	ˇ