PCIS-DASK/X ver. 3.21

for PC Compatibles

Function Reference Manual

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

CONTENTS

How to Use This Manual

This manual is designed to help you use the PCIS-DASK software driver for NuDAQ PCI-bus data acquisition cards. The manual describes how to install and use the software library to meet your requirements and help you program your own software applications. It is organized as follows:

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

- Chapter 2, "Function Description" gives the detailed description of each function call PCIS-DASK provided.

- Appendix A, "Status Codes" lists the status codes returned by PCIS-DASK functions, as well as their meanings.

- Appendix B, "AI Range Codes " lists all the valid AI range codes for each card.

- Appendix C, "AI Data Format" 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.

- Appendix D, "Function Support" shows which data acquisition hardware each PCIS-DASK function supports.

```
1
```

Using PCIS-DASK Functions

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 Linux environment.

Using PCIS-DASK also lets you take advantage of the power and features of Linux for your data acquisition applications. These include running multiple applications and using extended memory.

1.1 The Fundamentals of Developing the Application with PCIS-DASK

1.1.1 Creating a PCIS-DASK Application Using C/C++

To create a data acquisition application using PCIS-DASK and C/C++, follow these steps:

step 1. Edit the source files.

Include the 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 you can use to develop your data acquisition application. Incorporate the following statement in your code to include the header file.

#include "dask.h"

step 2. Build your application.

Using the appropriated C/C++ compiler (gcc or cc) to compile the program. You should also use You should also use the -lpci_dask option to link libpci_dask.so library.

ex. gcc -o testai testai.c –lpci_dask.

1.2 PCIS-DASK Functions Overview

PCIS-DASK functions are grouped to the following 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**

- **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**

  - Timer/Counter functions
  - The General-Purpose Timer/Counter functions

- **DIO Function Group**

  - Digital Input/Output Configuration function
  - Dual-Interrupt System Setting functions

```
2
```

Function Description

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 *3.2 Function Reference*.

2.1   Data Types

We defined some data types in DASK.H. These data types are used by PCIS-DASK library. We suggest you to use these data types in your application programs. The following table shows the data type names, their ranges and the corresponding data types in C/C++, Visual Basic and Delphi (We didn't define these data types in DASK.BAS and DASK.PAS. Here they are just listed for 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

2.2.1   AI_9111_Config

@ Description

Informs 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.

@ Cards Support

9111

@ Syntax

I16 AI_9111_Config (U16 CardNumber, U16 TrigSource, U16 PreTrgEn, U16 TraceCnt)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
TrigSource :   The continuous A/D conversion trigger source.
    Valid values:
        TRIG_INT_PACER: on-board Programmable pacer
        TRIG_EXT_STROBE: external signal trigger
PreTrgEn: Enable or Disable Pre-Trigger mode.
        TRUE: Enable Pre-Trigger mode
        FALSE: Disable Pre-Trigger mode
TraceCnt: The number of data will be accessed after a specific trigger event.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.2   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.

@ Cards Support

9112

@ Syntax

I16 AI_9112_Config (U16 CardNumber, U16 TrigSource)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
TrigSource :   The continuous A/D conversion trigger source.
    Valid values:
        TRIG_INT_PACER: on-board Programmable pacer
        TRIG_EXT_STROBE: external signal trigger

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.3   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.

@ Cards Support

9113

@ Syntax

I16 AI_9113_Config (U16 CardNumber, U16 TrigSource)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
TrigSource :   The continuous A/D conversion trigger source.
                   Valid values:
                     TRIG_INT_PACER: on-board Programmable pacer

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.4   AI_9114_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.

@ Cards Support

9114

@ Syntax

I16 AI_9114_Config (U16 CardNumber, U16 TrigSource)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
TrigSource :   The continuous A/D conversion trigger source.
                   Valid values:
                     TRIG_INT_PACER: on-board Programmable pacer
                     TRIG_EXT_STROBE: external signal trigger

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.5   AI_9116_Config

@ Description

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

@ Cards Support

9116

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this 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:

(1) **A/D Polarity Control**
    P9116_AI_BiPolar
    P9116_AI_UniPolar
(2) **A/D Channel Input Mode**
    P9116_AI_SingEnded
    P9116_AI_Differential
(3) **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:
(1) **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
(2) **Trigger Polarity**
    P9116_AI_TrgNegative: Trigger negative edge active
    P9116_AI_TrgPositive: Trigger positive edge active
(3) **Time Base Selection**
    P9116_AI_IntTimeBase: Internal time Base (24 MHz)
    P9116_AI_ExtTimeBase: External time base
(4) **Delay Source Selection**
    P9116_AI_DlyInSamples: delay in samples
    P9116_AI_DlyInTimebase: delay in time base
(5) **Re-Trigger Mode Enable**
    P9116_AI_ReTrigEn: Re-trigger in an acquisition is enabled
(6) **MCounter Enable**
    P9116_AI_MCounterEn: Mcounter is enabled and then the trigger
        signal is ignore before M terminal count is reached.
(7) **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 will be accessed after a specific trigger event.
This argument is only valid for Middle trigger and Delay trigger mode.
MCnt : The counter value of MCounter . This argument is only valid for Pre-
trigger and Middle trigger mode.
ReTrgCnt : The accepted trigger times in an acquisition. This argument is only
valid for Delay trigger and Post trigger mode.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.6   AI_9116_CounterInterval

@ Description

Informs 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.

@ Cards Support

9116

@ Syntax

 I16 AI_9116_CounterInterval (U16 wCardNumber, U32 ScanIntrv, U32 SampIntrv)

@ Parameter

 CardNumber : The card id of the card that want to perform this operation.
 ScanIntrv : The length of the scan interval (that is, 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 (that is, *the number of channels in a scan sequence * SampIntrv*).

@ Return Code

 NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.7   AI_9118_Config

@ Description

 Informs 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.

@ Cards Support

 9118

@ Syntax

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

@ Parameter

 CardNumber : The card id of the card that want to perform this 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:
              (1) **A/D Polarity Control**
                  P9118_AI_BiPolar
                  P9118_AI_UniPolar
              (2) **A/D Channel Input Mode**
                  P9118_AI_SingEnded
                  P9118_AI_Differential
              (3) **External Gate Enable**
                  P9118_AI_ExtG: 8254 counter is controlled by TGIN pin
              (4) **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:
              (1) **Digital Trigger Polarity**
                  P9118_AI_DtrgNegative: Digital trigger negative active
                  P9118_AI_DtrgPositive: Digital trigger positive active

(2) **External Trigger Polarity**

    P9118_AI_EtrgNegative: External trigger negative active

    P9118_AI_EtrgPositive: External trigger positive active

(3) **Burst Mode Enable**

    P9118_AI_BurstModeEn: Burst Mode is enabled

(4) **Burst Mode with Sample and Hold Mode Enable**

    P9118_AI_SampleHold: Burst mode with sample and hold is enabled

(5) **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 will be accessed after a specific trigger event

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.8   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.

@ Cards Support

9812/10

@ Syntax

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

@ Parameter

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

TrgMode :   The setting for A/D trigger mode. The valid trigger modes are as follows:

    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 :   The setting for A/D Trigger Source. The valid trigger sources are as follows:

    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 :   The setting of Trigger polarity. The valid values are:

    P9812_TRGSLP_POS : Positive slope Trigger

    P9812_TRGSLP_NEG : Negative slope Trigger

ClkSel :   The setting of A/D clock source. 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:

(1) **A/D Clock Frequency**

    P9812_AD2_GT_PCI : Freq. of A/D clock is higher than PCI clock freq.

    P9812_AD2_LT_PCI : Freq. of A/D clock is lower than PCI clock freq.

(2) **The 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 bitwise-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	trigger
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:

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

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.9   AI_AsyncCheck

@ Description

Check the current status of the asynchronous analog input operation.

@ Cards Support

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

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that performs the 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

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered
ErrorFuncNotSupport

2.2.10 AI_AsyncClear

@ Description

Stop the asynchronous analog input operation.

@ Cards Support

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

@ Syntax

I16 AI_AsyncClear (U16 CardNumber, U32 *AccessCnt)

@ Parameter

CardNumber : The card id of the card that performs the 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

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.11 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.

@ Cards Support

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

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that performs 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

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.12 AI_AsyncDblBufferMode

@ Description

Enables or disables double-buffered data acquisition mode.

@ Cards Support

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

@ Syntax

I16 AI_AsyncDblBufferMode (U16 CardNumber, BOOLEAN Enable)

@ 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.
                (constants TRUE and FALSE are defined in DASK.H)

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.13 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* and *AI_ContScanChannels*) or a
disk file (if continuous AI function is: *AI_ContReadChannelToFile*,
AI_ContReadMultiChannelsToFile and *AI_ContScanChannelsToFile*).
You can execute this function repeatedly to return sequential half buffers of the data.

@ Cards Support

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

@ Syntax

I16 AI_AsyncDblBufferTransfer (U16 CardNumber, U16 *Buffer)

@ Parameter

CardNumber : The card id of the card that performs 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 into a disk file, this argument is of no use.
                Please refer to Appendix C, *AI Data Format* for the data format in
                Buffer or the data file.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered,
ErrorFuncNotSupport, ErrorNotDoubleBufferMode,
ErrorInvalidSampleRate

2.2.14 AI_ContReadChannel

@ Description

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

@ Cards Support

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

@ Syntax

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

@ Parameter

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

Channel :   Analog input channel number
                Range: 0 through 15 for PCI-9111
                Range: 0 through 15 for PCI-9112/cPCI-9112
                Range: 0 through 31 for PCI-9113
                Range: 0 through 31 for PCI-9114
                Range: 0 through 63 for cPCI-9116
                Range: 0 through 15 for PCI-9118
                Range: 0 for PCI-9812/10

AdRange :   The analog input range the specified channel is setting. We define
                some constants to represent various A/D input ranges in DASK.H.
                Please refer to the 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 *ReadCount*. If
                double-buffered mode is enabled, this buffer is of no use, you can
                ignore this argument. Please refer to Appendix C, *AI Data Format* for
                the data format in *Buffer*.

ReadCount :   If double-buffered mode is disabled, *ReadCount* is the total number of
                A/D conversions (except cPCI9116) or the total number of scans (for
                cPCI9116) to be performed. For double-buffered acquisition,
                ReadCount is the size (in samples) of the circular buffer (except
                cPCI9116) or the size (in samples) allocated for each channel in the
                circular buffer (for cPCI9116) and its value must be a multiple of 4.

> **Note:** if the card is PCI-9111, PCI-9113 or PCI-9114, 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 cPCI9116, this parameter is ignored. Use
                `AI_9116_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 is set as *2* by the driver.
                Hence, the sampling rate is:
                Frequency of external clock source / 2

SyncMode :   Whether this 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 complete.
                    ASYNCH_OP: asynchronous A/D conversion

@ Return Code

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

2.2.15   AI_ContReadChannelToFile

@ Description

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

@ Cards Support

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

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Channel :   Analog input channel number
                  Range: 0 through 15 for PCI-9111
                  Range: 0 through 15 for PCI-9112/cPCI-9112
                  Range: 0 through 31 for PCI-9113
                  Range: 0 through 31 for PCI-9114
                  Range: 0 through 63 for cPCI-9116
                  Range: 0 through 15 for PCI-9118
                  Range: 0 for PCI-9812/10
AdRange :   The analog input range the specified channel is setting. We define
                  some constants to represent various A/D input ranges in DASK.H.
                  Please refer to the Appendix B, *AI Range Codes*, for the valid range
                  values.

FileName :   Name of data file which stores the acquired data
ReadCount :   If double-buffered mode is disabled, *ReadCount* is the number of A/D
                  conversions (except cPCI9116) or the total number of scans (for
                  cPCI9116) to be performed. For double-buffered acquisition,
                  ReadCount is the size (in samples) of the circular buffer (except
                  cPCI9116) or the size (in samples) allocated for each channel in the
                  circular buffer (for cPCI9116) and its value must be a multiple of 4.

> **Note:** if the card is PCI-9111, PCI-9113 or PCI-9114, 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 cPCI9116, this parameter is ignored. Use
                  `AI_9116_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 is set as *2* by the driver.
Hence, the sampling rate is:
Frequency of external clock source / 2

SyncMode : Whether this 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 complete.
ASYNCH_OP:asynchronous A/D conversion

@ Return Code

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

2.2.16   AI_ContReadMultiChannels

@ Description

This function performs continuous A/D conversions on the specified analog input
channels at a rate as close to the rate you specified. This function takes advantage of
the PCI-9118 and PCI-9116 auto-scan and channel-gain queue functionality to
perform multi-channel analog input.

@ Cards Support

9116, 9118

@ Syntax

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

@ 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
value:
cPCI-9116: 1 through 511
PCI-9118: 1 through 255
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 of channel order setting, you can set the
channel order as you wish.
PCI-9118: numbers in *Chans* must be within 0 and 15. Since there is
no restriction of channel order setting, you can set the
channel order as you wish.

AdRanges : An integer array of length *numChans* that contains the analog input
range for every channel in array *Chans*.
PCI-9118/cPCI9116:
Please refer to the Appendix B for the valid range values. Since
PCI-9118/cPCI-9116 supports different ranges, the range values
in *AdRanges* can be any of the valid range values of PCI-
9118/cPCI-9116.
Buffer : An integer array to contain the acquired data. The length of *Buffer*
must be equal to or greater than the value of parameter *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, then channel 3, then channel 8, ... 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], ... The data from channel 8 is stored in *Buffer*[1], *Buffer*[4], *Buffer*[7], ... The data from channel 0 is stored in *Buffer*[2], *Buffer*[5], *Buffer*[8], ... If double-buffered mode is enabled, this buffer is of no use, you can ignore this argument. Please refer to Appendix C, *AI Data Format* for the data format in *Buffer*.

ReadCount : If double-buffered mode is disabled, *ReadCount* is the number of A/D conversions (for PCI9118) or the total number of scans (for cPCI9116) to be performed. For double-buffered acquisition, *ReadCount* is the size (in samples) of the circular buffer (for PCI9118) or the size (in samples) allocated for each channel in the circular buffer (for cPCI9116) and its value must be a multiple of 4.

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 cPCI9116, this parameter is ignored. Use `AI_9116_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 : Whether this 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 complete.
  ASYNCH_OP:asynchronous A/D conversion

@ Return Code

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

2.2.17   AI_ContReadMultiChannelsToFile

@ Description

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

@ Cards Support

9116, 9118

@ Syntax

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

@ 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 value:
cPCI-9116: 1 through 511
PCI-9118: 1 through 255

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 of channel order setting, you can set the channel order as you wish.
PCI-9118: numbers in *Chans* must be within 0 and 15. Since there is no restriction of channel order setting, you can set the channel order as you wish.

AdRanges : An integer array of length *numChans* that contains the analog input range for every channel in array *Chans*.
CPCI-9116/PCI-9118:
Please refer to the Appendix B for the valid range values. Since PCI-9118 supports different ranges, the range values in *AdRanges* can be any of the valid range values of PCI-9118/cPCI-9116.

FileName : Name of data file which stores the acquired data

ReadCount : If double-buffered mode is disabled, *ReadCount* is the number of A/D conversions (for PCI9118) or the total number of scans (for cPCI9116) to be performed. For double-buffered acquisition, *ReadCount* is the size (in samples) of the circular buffer (for PCI9118) or the size (in samples) allocated for each channel in the circular buffer (for cPCI9116) and its value must be a multiple of 4.

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 cPCI9116, this parameter is ignored. Use `AI_9116_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 : Whether this 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 complete.
ASYNCH_OP:asynchronous A/D conversion

@ Return Code

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

2.2.18 AI_ContScanChannels

@ Description

This function performs continuous A/D conversions on the specified continuous analog input channels at a rate as close to the rate you specified. This function takes advantage of the hardware auto-scan functionality to perform multi-channel analog input.

@ Cards Support

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

@ Syntax

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

@ Parameter

CardNumber : The card ID of the card that want to perform this 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 *Channel* 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 *Channel* 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 *Channel* 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 *Channel* 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 *Channel* 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 *Channel* 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-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 the continuous specified channel is setting.
Please refer to the Appendix B 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 *ReadCount*.
The acquired data is stored in interleaved sequence. For example, if
the value of *Channel* 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, ... 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], ... The
data from channel 1 is stored in *Buffer*[1], *Buffer*[4], *Buffer*[7], ... The
data from channel 0 is stored in *Buffer*[2], *Buffer*[5], *Buffer*[8], ... If
double-buffered mode is enabled, this buffer is of no use, you can
ignore this argument. Please refer to Appendix C, *AI Data Format* for
the data format in *Buffer*.

ReadCount : If double-buffered mode is disabled, *ReadCount* is the number of A/D
conversions (except cPCI9116) or the total number of scans (for
cPCI9116) to be performed. For double-buffered acquisition,
ReadCount is the size (in samples) of the circular buffer (except
cPCI9116) or the size (in samples) allocated for each channel in the
circular buffer (for cPCI9116) and its value must be a multiple of 4.

> **Note:** if the card is PCI-9111, PCI-9113 or PCI-9114, 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 cPCI9116, this parameter is ignored. Use
AI_9116_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 is set as *2* by the driver.
Hence, the sampling rate is:
Frequency of external clock source / 2

SyncMode :   Whether this 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 complete.
    ASYNCH_OP: asynchronous A/D conversion

@ Return Code

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

2.2.19   AI_ContScanChannelsToFile

@ Description

This function performs continuous A/D conversions on the specified continuous analog
input channels at a rate as close to the rate you specified 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). Please 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 hardware auto-scan functionality to perform multi-channel
analog input.

@ Cards Support

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

@ Syntax

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

@ Parameter

CardNumber : The card ID of the card that want to perform this 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 *Channel* 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 *Channel* 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 *Channel* 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 *Channel* 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 *Channel* 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 *Channel* 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-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 the continuous specified channel is setting. Please refer to the Appendix B for the valid range values.

FileName : Name of data file which stores the acquired data

ReadCount : If double-buffered mode is disabled, *ReadCount* is the number of A/D conversions (except cPCI9116) or the total number of scans (for cPCI9116) to be performed. For double-buffered acquisition, *ReadCount* is the size (in samples) of the circular buffer (except cPCI9116) or the size (in samples) allocated for each channel in the circular buffer (for cPCI9116) and its value must be a multiple of 4.

> **Note:** if the card is PCI-9111, PCI-9113 or PCI-9114, 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 cPCI9116, this parameter is ignored. Use `AI_9116_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 is set as *2* by the driver.
Hence, the sampling rate is:
Frequency of external clock source / 2

SyncMode : Whether this 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 complete.
   ASYNCH_OP: asynchronous A/D conversion

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorInvalidIoChannel, ErrorInvalidSampleRate, ErrorInvalidAdRange,

ErrorTransferCountTooLarge, ErrorContIoNotAllowed, ErrorLastChannelNotZero, ErrorDiffRangeNotSupport, ErrorChannelNotDescending, ErrorChannelNotAscending

2.2.20 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.

@ Cards Support

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

@ Syntax

I16 AI_ContStatus (U16 CardNumber, U16 *Status)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Status :　　　The continuous AI status returned. The description of the parameter *Status* for various card types is the following:

    PCI9111/PCI9113/PCI9114 :
      bit 0 : '0' indicates FIFO is empty
      bit 1 : '0' indicates FIFO is Half Full
      bit 2 : '0' indicates FIFO is Full, the data might have been lost
      bit 3 : '0' indicates AD is busy, the A/D data hasn't been latched into FIFO
          yet
      bit 4 ~ 15 : not used

    PCI9112:
      bit 0 : '1' indicates A/D conversion is Completed (Ready)
      bit 1 : '1' indicates A/D conversion is Over-Run
      bit 2 ~ 15 : not used

    cPCI9116:
      bit 0 : '1' indicates A/D conversion is Over Speed
      bit 1 : '1' indicates A/D conversion is Over-Run
      bit 2 : '1' indicates Scan Counter Counts to zero
      bit 3 : '1' indicates External Digital Trigger ever happened
      bit 4 : '1' indicates A/D FIFO is empty
      bit 5 : '1' indicates A/D FIFO is Half Full
      bit 6 : '0' indicates A/D FIFO is Full
      bit 7 ~ 15 : not used

    PCI9118:
      bit 0 : '1' indicates A/D conversion is Completed (Ready)
      bit 1 : '1' indicates A/D conversion is Over-Run
      bit 2 : '1' indicates A/D conversion is Over-Speed
      bit 3 : '1' indicates Burst Mode of A/D conversion is Over-Run
      bit 4 : '1' indicates External Digital Trigger ever happened
      bit 5 : '1' indicates About Trigger of A/D conversion is Completed
      bit 6 : '1' indicates A/D FIFO is empty
      bit 7 : '1' indicates FIFO is Half Full
      bit 8 : '1' indicates FIFO is Full
      bit 9 ~ 15 : not used

    PCI9812:
      bit 0 : '1' indicates FIFO is ready for Input (Not Full)
      bit 1 : '1' indicates FIFO is at least Half-Full
      bit 2 : '1' indicates FIFO is ready for Output (Not Empty)
      bit 3 : '3' indicates the post trigger counter reaches zero
      bit 4 ~ 15 : not used

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered

2.2.21 AI_ContVScale

@ Description

This function converts the values of an array of acquired binary data from an continuous A/D conversion call to the actual input voltages. The acquires binary data in the reading array might include the channel information (please refer to continuous functions, 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.

@ Cards Support

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

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
AdRange :   The analog input range the continuous specified channel is setting.
      Please refer to the Appendix B for the valid range values.
readingArray : Acquired continuous analog input data array
voltageArray : computed voltages array returned

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorInvalidAdRange

2.2.22 AI_InitialMemoryAllocated

@ Description

This function returns the available memory size for analog input in the device driver in argument *MemSize*. The continuous analog input transfer size can not exceed this size.

@ Cards Support

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

@ Syntax

  I16 AI_InitialMemoryAllocated (U16 CardNumber, U32 *MemSize)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
MemSize :   The available memory size for continuous AI in device driver of this
      card. The unit is KB (1024 bytes).

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered

2.2.23 AI_ReadChannel

@ Description

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

@ Cards Support

9111, 9112, 9113, 9114, 9116, 9118

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Channel :   Analog input channel number.
                   Range: 0 through 15 for PCI-9112/cPCI-9112, PCI-9111, PCI-9118
                   Range: 0 through 31 for PCI-9113, PCI-9114
                   Range: 0 through 63 for cPCI-9116
AdRange :   The analog input range the specified channel is setting. Please refer
                   to the Appendix B for the valid range values.
Value :   The A/D converted value. The data format in *value* is described as
                   below:

> **PCI-9113**
>   16-bit unsigned data:
>
>   B15 … B12 D11 D10 …. D1 D0
>
>   where D11, D10, ... , D0 : A/D converted data
>
>   B15 ~ B12: don't care

> **PCI-9114**
>   16-bit signed data:
>
>   D15 D14 …………….. D1 D0
>
>   where D15, D14, ... , D0 : A/D converted data

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

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport,
ErrorInvalidIoChannel, ErrorInvalidAdRange

2.2.24   AI_VReadChannel

@ Description

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

@ Cards Support

9111, 9112, 9113, 9114, 9116, 9118

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Channel :   Analog input channel number.
                   Range: 0 through 15 for PCI-9112/cPCI-9112, PCI-9111, PCI-9118
                   Range: 0 through 31 for PCI-9113, PCI-9114
                   Range: 0 through 63 for cPCI-9116
AdRange :   The analog input range the specified channel is setting. Please refer
                   to the Appendix B for the valid range values.
voltage :   The measured voltage value returned and scaled to units of voltage.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport,
ErrorInvalidIoChannel, ErrorInvalidAdRange

2.2.25   AI_VoltScale

@ Description

This function converts the result from an AI_ReadChannel call to the actual input voltage.

@ Cards Support

9111, 9112, 9113, 9114, 9116, 9118

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
AdRange :   The analog input range the specified channel is setting. Please refer to the Appendix B for the valid range values.
reading :   The result of the AD Conversion.
voltage :   Computed voltage value.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorInvalidAdRange

2.2.26   AO_6208A_Config

@ Description

Sets the *Voltage to Current* Mode of PCI-6208A.

@ Cards Support

6208A

@ Syntax

I16 AO_6208A_Config (U16 CardNumber, U16 V2AMode)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
V2AMode :   The voltage to current mode. The valid V2Amode are:
        P6208_CURRENT_0_20MA
        P6208_CURRENT_5_25MA
        P6208_CURRENT_4_20MA

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorInvalidIoChannel

2.2.27   AO_6308A_Config

@ Description

Sets the *Voltage to Current* Mode of PCI-6308A.

@ Cards Support

6308A

@ Syntax

I16 AO_6308A_Config (U16 CardNumber, U16 V2AMode)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
V2AMode :   The voltage to current mode. The valid V2Amode are:
        P6308_CURRENT_0_20MA
        P6308_CURRENT_5_25MA
        P6308_CURRENT_4_20MA

@ **Return Code**

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorInvalidIoChannel

2.2.28   AO_6308V_Config

@ **Description**

Informs PCIS-DASK library of the polarity (unipolar or bipolar) that the output channel is configured for the analog output and the reference voltage value selected for an analog output channel of PCI-6308V. You can configure each channel to use an internal reference of 10V or an external reference (0V ~ +10V) by setting related jumpers. You must call this function before calling function to perform voltage output operation.

@ **Cards Support**

6308V

@ **Syntax**

I16 AO_6308V_Config (U16 wCardNumber, U16 Channel, U16 wOutputPolarity, F64 refVoltage)

@ **Parameter**

CardNumber : The card id of the card that want to perform this operation.
Channel   : The AO channel number configured. The valid values are:
        P6308V_AO_CH0_3
        P6308V_AO_CH4_7
OutputPolarity : The polarity (unipolar or bipolar) of the output channel. The valid values are:
        P6308V_AO_UNIPOLAR
        P6308V_AO_BIPOLAR
refVoltage   : Voltage reference value.
      If the D/A reference voltage source your device use is internal reference, the valid values for *refVoltage* is 10.
      If the D/A reference voltage source your device use is external reference, the valid range for *refVoltage* is 0 to +10.

Note : If the 10V D/A reference voltage is selected, the D/A output range is 0V~10V.

@ **Return Code**

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorInvalidDaRefVoltage

2.2.29   AO_9111_Config

@ **Description**

Informs PCIS-DASK library of the polarity (unipolar or bipolar) that the output channel is configured for the analog output of PCI9111. You must call this function before calling function to perform voltage output operation.

@ **Cards Support**

9111

@ **Syntax**

I16 AO_9111_Config (U16 CardNumber, U16 OutputPolarity)

@ **Parameter**

CardNumber : The card id of the card that want to perform this operation.
OutputPolarity : The polarity (unipolar or bipolar) of the output channel. The valid values are:

P9111_AO_UNIPOLAR
P9111_AO_BIPOLAR

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.30   AO_9112_Config

@ Description

Informs PCIS-DASK library of the reference voltage value selected for an analog output channel of PCI9112. You can configure each channel to use an internal reference of –5V (default) or –10V or an external reference (-10V ~ +10V) by setting related jumpers. You must call this function before calling function to perform voltage output operation.

@ Cards Support

9112

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Channel   : The AO channel number configured.
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 : If 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 -10V~0V.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorInvalidDaRefVoltage

2.2.31   AO_VoltScale

@ Description

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

@ Cards Support

9111, 9112, 9118, 6208V/16V/08A, 6308V/08A

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Channel :   The analog output channel number.
    Range: 0 or 1 for PCI-9112/cPCI-9112
    Range: 0 for PCI-9111
    Range: 0 or 1 for PCI-9118
    Range: 0 through 7 for PCI-6208V/08A and PCI-6308V/08A
    Range: 0 through 15 for PCI-6216V
Voltage :   Voltage, in volts, to be converted to a binary value
binValue :   the converted binary value returned

@ **Return Code**

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorInvalidIoChannel. ErrorDaVoltageOutOfRange

2.2.32   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.

@ **Cards Support**

9111, 9112, 9118, 6208V/16V/08A, 6308V/08A

@ **Syntax**

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

@ **Parameter**

CardNumber : The card id of the card that want to perform this operation.
Channel :   The analog output channel number.
                          Range: 0 or 1 for PCI-9112/cPCI-9112
                          Range: 0 for PCI-9111
                          Range: 0 or 1 for PCI-9118
                          Range: 0 through 7 for PCI-6208V/08A and PCI-6308V/08A
                          Range: 0 through 15 for PCI-6216V
Voltage :   The value to be scaled and written to the analog output channel. The range of voltages depends on the type of device, on the output polarity, and on the voltage reference (external or internal).

@ **Return Code**

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorInvalidIoChannel, ErrorDaVoltageOutOfRange

2.2.33   AO_WriteChannel

@ **Description**

Writes a binary value to the specified analog output channel.

@ **Cards Support**

9111, 9112, 9118, 6208V/16V/08A, 6308V/08A

@ **Syntax**

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

@ **Parameter**

CardNumber : The card id of the card that want to perform this operation.
Channel :   The analog output channel number.
                          Range: 0 or 1 for PCI-9112/cPCI-9112
                          Range: 0 for PCI-9111
                          Range: 0 or 1 for PCI-9118
                          Range: 0 through 7 for PCI-6208V/08A and PCI-6308V/08A
                          Range: 0 through 15 for PCI-6216V
Value :   The value to be written to the analog output channel.
                          Range: 0 through 4095 for PCI-9111, PCI-9112/cPCI-9112, PCI-9118
                                   0 though 32767 for PCI-6208A and PCI-6308A
                                   -32768 through 32767 for PCI-6208V/16V and PCI-6308V

@ **Return Code**

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorInvalidIoChannel

2.2.34   CTR_8554_CK1_Config

@ Description

Selects the source of CK1.

@ Cards Support

8554

@ Syntax

I16 CTR_8554_CK1_Config (U16 CardNumber, U16 ClockSource)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
ClockSource : The source of CK1. CK1_C8M or CK1_COUT11.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport,
InvalidCtrSource

2.2.35   CTR_8554_ClkSrc_Config

@ Description

Selects PCI-8554 counter #1 ~ #10 clock source. (Clock source of counter #11 is
8MHz and clock source of counter #12 is from COUT11, both are fixed.)

@ Cards Support

8554

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Ctr :   The counter number.
                  Range: 1~10
ClockSource : The clock source of the specified counter.
                  ECKN: external clock source
                  COUTN_1: the cascaded counter output (COUT n-1)
                  CK1: internal clock source CK1
                  COUT10: output of the counter 10

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport,
InvalidCounter

2.2.36   CTR_8554_Debounce_Config

@ Description

Selects debounce clock.

@ Cards Support

8554

@ Syntax

I16 CTR_8554_Debounce_Config (U16 CardNumber, U16 DebounceClock)

@ Parameter

CardNumber :   The card id of the card that want to perform this operation.
DebounceClock : DBCLK_COUT11: output of counter 11 DBCLK_2MHZ: 2MHz

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport,
InvalidCtrSource

2.2.37 CTR_Clear

@ Description

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

@ Cards Support

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

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Ctr : The counter number.
        Range: 0 for PCI-9111, PCI-9112/cPCI-9112, PCI-9113, PCI-9114,
           PCI-9118.
        0, 1, 2 for PCI-7248/cPCI-7248, cPCI-7249R, PCI-7296,
           PCI-7396.
        1~12 for PCI-8554
state : The logic state to which the counter is to be reset.
        Range: 0 or 1.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, InvalidCounter

2.2.38 CTR_Read

@ Description

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

@ Cards Support

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

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Ctr : The counter number.
        Range: 0 for PCI-9111, PCI-9112/cPCI-9112, PCI-9113, PCI-9114,
           PCI-9118.
        0, 1, 2 for PCI-7248/cPCI-7248, cPCI-7249R, PCI-7296,
           PCI-7396.
        1~12 for PCI-8554.
Value : Returns the current count of the specified counter.
        Range: 0 through 65536 for binary mode (default).
           0 through 9999 for BCD counting mode.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, InvalidCounter

2.2.39 CTR_Setup

@ Description

Configures the selected counter to operate in the specified mode.

@ Cards Support

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

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Ctr :   The counter number.
         Range: 0 for PCI-9111, PCI-9112/cPCI-9112, PCI-9113, PCI-9114,
             PCI-9118.
             0, 1, 2 for PCI-7248/cPCI-7248, cPCI-7249R, PCI-7296,
             PCI-7396.
             1~12 for PCI-8554
Mode :   The mode in which the counter is to operate.
         Valid value:
             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

         In this mode, 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
         diagram.
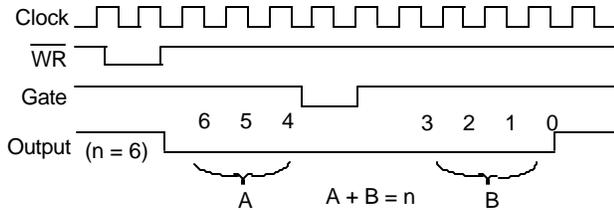
         PROG_ONE_SHOT:Programmable one-shot

         In this mode, the output goes low on the cofollowing the rising
         edge of the gate input and goes high on terminal count. The
         following diagram shows the
         PROG_ONE_SHOT mode timing diagram.
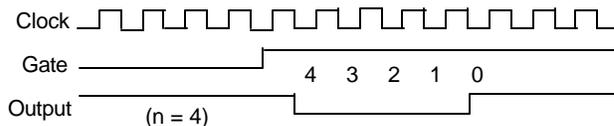
         RATE_GENERATOR:Rate generator

         In this mode, 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.

```
Clock  ⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔
Gate      4   3   2   1   0 (4)      3   2   1   0 (4)
Output      (n = 4)
```
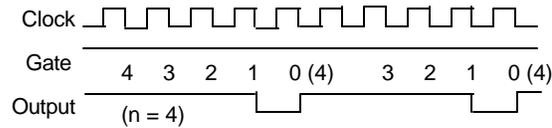
SQ_WAVE_RATE_GENERATOR: Square wave rate generator

In this mode, 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 diagram.

```
Clock  ⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔
Gate
              4   2   4   2   4   2   4   2   4   2   4   2
Output (n = 4)
              5   4   2   5   2   5   4   2   5   2   5   4
Output (n = 5)
```
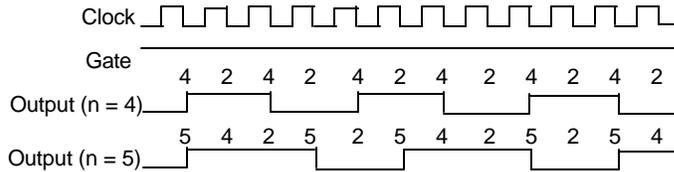
SOFT_TRIG: Software-triggered strobe

In this mode, 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 diagram.

```
Clock  ⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔
WR          ⊔ n = 4 ⊓
Gate
                4   3   2   1   0
Output
```
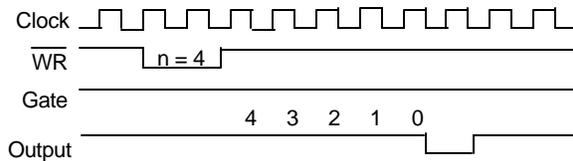
HARD_TRIG: Hardware-triggered strobe

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.

```
Clock  ⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔⊓⊔
Gate
                4   3   2   1   0
Output  n = 4
```
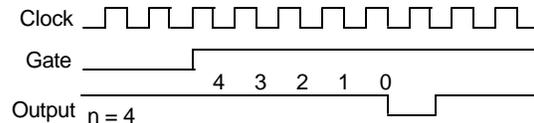
Count :   The period from one output pulse to the next.
BinBcd :   Whether the counter operates as a 16-bit binary counter or as a 4-decade binary-coded decimal (BCD) counter.
                  Valid value:
                    BIN: 16-bit binary counter.
                    BCD: 4-decade BCD counter.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, InvalidCounter

2.2.40   DI_7200_Config

@ Description

Informs PCIS-DASK library of the trigger source, and input mode selected for PCI7200/cPCI7200 with card ID *CardNumber*. You must call this function before calling function to perform continuous digital input operation.

7200

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
TrigSource :   The trigger mode for continuous digital input.
                Valid values:
                  TRIG_INT_PACER: on-board 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, the valid values are:
                  DI_TRIG_RISING: I_TRG is rising edge active
                  DI_TRIG_FALLING: I_TRG is falling edge active
I_REQ_Pol :   I_REQ Polarity (for handshaking mode), the valid values are:
                  IREQ_RISING: I_REQ is rising edge active
                  IREQ_FALLING: I_REQ is falling edge active

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.41   DI_7300A_Config

@ Description

Informs 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.

@ Cards Support

7300A Rev.A

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
PortWidth :   The width of digital input port (PORT A). The valid value is 0, 8, 16, or
                32.
TrigSource :   The trigger mode for continuous digital input.
                Valid values:
                  TRIG_INT_PACER: on-board programmable pacer timer0
                  TRIG_EXT_STROBE: external signal trigger
                  TRIG_HANDSHAKE: handshaking
                  TRIG_CLK_10MHz: 10MHz clock
                  TRIG_CLK_20MHz: 20MHz clock
WaitStatus :   DI Wait Trigger Status, the valid values are:
                  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_REQ_Pol :	I_REQ Polarity. This function is not implemented on PCI-7300A Rev.A/cPCI-7300A Rev.A card. You can ignore this argument.
ClearFifo :	FALSE: retain the FIFO data
	TRUE: clear FIFO data before perform digital input
DisableDI :	FALSE: digital input operation still active after DMA transfer complete. The input data still put into FIFO
	TRUE: disable digital input operation immediately when DMA transfer complete

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.42 DI_7300B_Config

@ Description

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

@ Cards Support

7300A Rev.B

@ Syntax

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

@ Parameter

CardNumber :	The card id of the card that want to perform this operation.
PortWidth :	The width of digital input port (PORT A). The valid value is 0, 8, 16, or 32.
TrigSource :	The trigger mode for continuous digital input.
	Valid values:
	TRIG_INT_PACER: on-board programmable pacer timer0
	TRIG_EXT_STROBE: external signal trigger
	TRIG_HANDSHAKE: handshaking
	TRIG_CLK_10MHz: 10MHz clock
	TRIG_CLK_20MHz: 20MHz clock
WaitStatus :	DI Wait Trigger Status, the valid values are:
	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:
	(1) DIREQ
	P7300_DIREQ_POS: DIREQ signal is rising edge active
	P7300_DIREQ_NEG: DIREQ signal is falling edge active
	(2) DIACK
	P7300_DIACK_POS: DIACK signal is rising edge active
	P7300_DIACK_NEG: DIACK signal is falling edge active
	(3) DITRIG
	P7300_DITRIG_POS: DITRIG signal is rising edge active
	P7300_DITRIG_NEG: DITRIG signal is falling edge active
ClearFifo :	FALSE: retain the FIFO data
	TRUE: clear FIFO data before perform digital input
DisableDI :	FALSE: digital input operation still active after DMA transfer complete. The input data still put into FIFO

> TRUE: disable digital input operation immediately when DMA transfer complete

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.43 DI_AsyncCheck

@ Description

Check the current status of the asynchronous digital input operation.

@ Cards Support

7200, 7300A

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that performs the asynchronous operation.
Stopped : 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 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 : The number of digital input data that has been transferred at the time the call to `DI_AsyncCheck()`.
AccessCnt is of no use (always returns 0) in DI_AsyncCheck() and DI_AsyncClear() with *PCI-7300A* board because PLX9080 has no function or register to get the current amount of DMA transfer.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.44 DI_AsyncClear

@ Description

Stop the asynchronous digital input operation.

@ Cards Support

7200, 7300A

@ Syntax

I16 DI_AsyncClear (U16 CardNumber, U32 *AccessCnt)

@ Parameter

CardNumber : The card id of the card that performs the 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.
AccessCnt is of no use (always returns 0) in DI_AsyncCheck() and DI_AsyncClear() with *PCI-7300A* board because PLX9080 has no function or register to get the current amount of DMA transfer.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.45 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.

@ Cards Support

7200

@ Syntax

I16 DI_AsyncDblBufferHalfReady (U16 CardNumber, BOOLEAN *HalfReady)

@ Parameter

CardNumber : The card id of the card that performs the asynchronous double-buffered operation.

HalfReady : Whether the next half buffer of data is available. If HalfReady = 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

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.46 DI_AsyncDblBufferMode

@ Description

Enables or disables double-buffered data acquisition mode.

@ Cards Support

7200

@ Syntax

I16 DI_AsyncDblBufferMode (U16 CardNumber, BOOLEAN Enable)

@ 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.
(constants TRUE and FALSE are defined in DASK.H)

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.47 DI_AsyncDblBufferTransfer

@ Description

Depending on the continuous DI function selected, half of the data of the circular buffer will be logged into the user buffer (if continuous DI function is: *DI_Cont ReadPort*) or a disk file (if continuous DI function is: *DI_ContReadPortToFile*). If the data will be saved in a file, the data is written to disk in binary format, with the lower byte first (little endian).
You can execute this function repeatedly to return sequential half buffers of the data.

@ Cards Support

7200

@ Syntax

I16 DI_AsyncDblBufferTransfer (U16 CardNumber, void *Buffer)

@ Parameter

CardNumber : The card id of the card that performs the asynchronous double-buffered operation.

Buffer : The user buffer to which the data is to be copied. If the data will be saved into a disk file, this argument is of no use.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorNotDoubleBufferMode

2.2.48   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.

@ Cards Support

7300A

@ Syntax

I16 DI_AsyncMultiBufferNextReady (U16 CardNumber, BOOLEAN *NextReady, U16 *BufferId)

@ Parameter

CardNumber : The card id of the card that performs the asynchronous multi-buffered operation.

NextReady : Whether the next buffer of data 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

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.49   DI_ContMultiBufferSetup

@ Description

This function set up the buffer for multi-buffered digital input. The function has to be called repeatedly to setup all of the data buffers.

@ Cards Support

7300A

@ Syntax

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

@ Parameter

CardNumber : The card id of the card that want to perform this 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

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorTransferCountTooLarge , ErrorContIoNotAllowed

2.2.50   DI_ContMultiBufferStart

@ Description

This function starts multi-buffered continuous digital input on the specified digital input port at a rate as close to the rate you specified.

@ Cards Support

7300A

@ Syntax

I16 DI_ContMultiBufferStart (U16 CardNumber, U16 Port, F64 SampleRate)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Port : Digital input port number. For PCI-7300A/cPCI-7300A, this argument must be set to 0.
SampleRate : The sampling rate you want for digital input in hertz (samples per second). Your maximum rate depends on the card type and your computer system. This argument is only useful if the DI trigger mode was set as internal programmable pacer (TRIG_INT_PACER) by calling `DI_7300A_Config()` or `DI_7300B_Config().`

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorInvalidIoChannel, ErrorContIoNotAllowed

2.2.51   DI_ContReadPort

@ Description

This function performs continuous digital input on the specified digital input port at a rate as close to the rate you specified.

@ Cards Support

7200, 7300A

@ Syntax

I16 DI_ContReadPort (U16 CardNumber, U16 Port, void *Buffer, U32 ReadCount, F64 SampleRate, U16 SyncMode)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Port : Digital input port number. For PCI-7200/cPCI-7200 and PCI-7300A/cPCI-7300A, this argument must be set to 0.
Buffer : The starting address of the memory to contain the input data. This memory must have been allocated for enough space to store input data. If double-buffered mode is enabled, this buffer is of no use, you can ignore this argument.
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 and its value must be even.
SampleRate : The sampling rate you want for digital input in hertz (samples per second). Your maximum rate depends on the card type and your computer system. This argument is only useful if the DI trigger mode was set as internal programmable pacer (TRIG_INT_PACER) by calling `DI_7200_Config()` or `DI_7300_Config()`. For the other settings, you have to set this argument as CLKSRC_EXT_SampRate.
SyncMode : Whether this 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 complete.
    ASYNCH_OP: asynchronous digital input operation

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorInvalidIoChannel, ErrorTransferCountTooLarge , ErrorContIoNotAllowed

2.2.52 DI_ContReadPortToFile

@ Description

This function performs continuous digital input on the specified digital input port at a rate as close to the rate you specified 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). Please refer to Appendix D, *Data File Format* for the data file structure.

@ Cards Support

7200, 7300A

@ Syntax

I16 DI_ContReadPortToFile (U16 CardNumber, U16 Port, U8 *FileName, U32 ReadCount, F64 SampleRate, U16 SyncMode)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Port : Digital input port number. For PCI-7200/cPCI-7200 and PCI-7300A/cPCI-7300A, this argument must be set to 0.
FileName : Name of data file which stores the acquired data
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 and its value must be even.
SampleRate : The sampling rate you want for digital input in hertz (samples per second). Your maximum rate depends on the card type and your computer system. This argument is only useful if the DI trigger mode was set as internal programmable pacer (TRIG_INT_PACER) by calling `DI_7200_Config()` or `DI_7300_Config()`. For the other settings, you have to set this argument as CLKSRC_EXT_SampRate.
SyncMode : Whether this 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 complete.
    ASYNCH_OP: asynchronous digital input operation

@ Return Code

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

2.2.53 DI_ContStatus

@ Description

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

@ Cards Support

7200, 7300A

@ Syntax

I16 DI_ContStatus (U16 CardNumber, U16 *Status)

@ Parameter

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

Status :   The continuous DI status returned. The description of the parameter
                    Status for various card types is the following:

PCI7200 :
   bit 0 : '1' indicates D/I FIFO is Full (Over-Run)
   bit 1 : '1' indicates D/O FIFO is Empty (Under-Run)
   bit 2 ~ 15 : not used

PCI7300A_RevA:
   bit 0 : '1' indicates DI FIFO is full during input sampling and some data
            were lost. Writes '1' to clear this bit
   bit 1 : '1' indicates DI FIFO is full
   bit 2 : '1' indicates DI FIFO is empty
   bit 3 ~ 15 : not used

PCI7300A_RevB:
   bit 0 : '1' indicates DI FIFO is full during input sampling and some data
            were lost. Writes '1' to clear this bit
   bit 1 : '1' indicates DI FIFO is full
   bit 2 : '1' indicates DI FIFO is empty
   bit 3 ~ 15 : not used

@ Return Code
NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered

2.2.54   DI_InitialMemoryAllocated

@ Description
This function returns the available memory size for digital input in the device driver of
this card. The continuous digital input transfer size can not exceed this size.

@ Cards Support
7200, 7300A

@ Syntax
   I16 DI_InitialMemoryAllocated (U16 CardNumber, U32 *MemSize)

@ Parameter
CardNumber : The card id of the card that want to perform this operation.
MemSize :   The available memory size for continuous DI in device driver of this
                  card.
                  The unit is KB (1024 bytes).

@ Return Code
NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered

2.2.55   DI_ReadLine

@ Description
Read the digital logic state of the specified digital line in the specified port.

@ Cards Support
6208V/16V/08A, 6308V/08A, 7200, 7230, 7233, 7248, 7249, 7250/51, 7252, 7296,
7300A, 7396, 7432, 7433, 8554, 9111, 9112, 9114, 9116, 9118

@ Syntax
   I16 DI_ReadLine (U16 CardNumber, U16 Port, U16 Line, U16 *State)

@ Parameter
CardNumber : The card id of the card that want to perform this operation.
Port :   Digital input port number. The valid value:
                  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-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-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-7396:
   Channel_P1A,   Channel_P1B,
   Channel_P1C,
   Channel_P2A,   Channel_P2B,
   Channel_P2C,
   Channel_P3A,   Channel_P3B,
   Channel_P3C,
   Channel_P4A,   Channel_P4B,
   Channel_P4C
PCI-7300A/cPCI-7300A: 1 (auxiliary input port)
PCI-7432/cPCI-7432: 0
PCI-7433/cPCI-7433: PORT_DI_LOW, PORT_DI_HIGH
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

Line :   The digital line to be read. The valid value:
PCI-6208V/16V/08A: 0 through 3
PCI-6308V/08A: 0 through 3
PCI-7200/cPCI-7200: 0 through 31 (for port 0)
                      0 through 3 (for auxiliary input port of cPCI7200)
PCI-7230/cPCI-7230: 0 through 15
PCI-7233: 0 through 31
PCI-7248/cPCI-7248: 0 through 7
cPCI-7249R: 0 through 7
PCI-7250/51: 0 through 7
cPCI-7252: 0 through 15
PCI-7296: 0 through 7

PCI-7300A/cPCI-7300A: 0 through 3
PCI-7396: 0 through 7
PCI-7432/cPCI-7432/cPCI-7432R: 0 through 31
PCI-7433/cPCI-7433/cPCI-7433R: 0 through 31
PCI-8554: 0 through 7
PCI-9111: 0 through 15
PCI-9112/cPCI-9112: 0 through 15
PCI-9114: 0 through 15
cPCI-9116: 0 through 7
PCI-9118: 0 through 3

State :   Returns the digital logic state, 0 or 1, of the specified line.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport,
ErrorInvalidIoChannel

2.2.56   DI_ReadPort

@ Description

Read digital data from the specified digital input port.

@ Cards Support

6208V/16V/08A, 6308V/08A, 7200, 7230, 7233, 7248, 7249, 7250/51, 7252, 7296,
7300A, 7396, 7432, 7433, 7434, 8554, 9111, 9112, 9114, 9116, 9118

@ Syntax

I16 DI_ReadPort (I16 CardNumber, U16 Port, U32 *Value)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Port :   Digital input port number. The valid value:
      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-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-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-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
          PCI-7433/cPCI-7433: PORT_DI_LOW, PORT_DI_HIGH
          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

Note: The value, Channel_P*n*, 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.
          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: 8-bit data
          cPCI-7249R: 8-bit data
          PCI-7250/51: 8-bit data
          cPCI-7252: 16-bit data
          PCI-7296: 8-bit data
          PCI-7300A/cPCI-7300A: 4-bit data
          PCI-7396: 24-bit data (for Channel_P*n*, where *n* is the channel number) or
                    8-bit data (for Channel_P*n*A, Channel_P*n*B, Channel_P*n*C , 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-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

Note: The data format for Channel_P*n* is as follows:

	Don't care	PORT C	PORT B	PORT A
Bit	31 - 24	23 - 16	15 – 8	7 - 0

@ Return Code

NoError, CardNotRegistered, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.57 DIO_7300SetInterrupt

@ Description

This function controls the interrupt sources (AuxDI0 and Timer 2) of local interrupt system of PCI-7300A/cPCI7300A and sent the corresponding signal.

@ Cards Support

7300A

@ Syntax

I16 DIO_7300SetInterrupt (U16 CardNumber, I16 AuxDIEn, I16 T2En, void
(*event1_handler)(int), void (*event2_handler)(int))

@ Parameter

CardNumber : The card id of the card that want to be performed this operation.
AuxDIEn :   The control value for AUXDI interrupt.
                    The valid values:
                    0: disabled
                    1: enabled
T2En :   The control value for Timer2 interrupt.
                    The valid values:
                    0: disabled
                    1: enabled
event1_handler : address of the user specified ***signal handler***. The signal handler is
                    called when the specified INT1 event occurs. If you do not want to
                    use a callback function, set *event1_handler* to 0.
event2_handler : address of the user specified ***signal handler***. The signal
                    handler is called when the specified INT2 event occurs. If you
                    do not want to use a callback function, set *event2_handler* to
                    0.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered
ErrorFuncNotSupport

2.2.58 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 the Windows PostMessage API.

@ Cards Support

7300A

@ Syntax

I16 DIO_AUXDI_EventMessage (U16 CardNumber, I16 AuxDIEn, HANDLE
windowHandle, U32 message, void *callbackAddr())

@ Parameter

CardNumber :   The card id of the card that want to be performed this operation.
AuxDIEn   :   The control value for AUXDI interrupt.
                  The valid values:
                  0: disabled
                  1: enabled
windowHandle : The handle to the window you want to receive a Windows message
                  in when the specified AUXDI event happens. If *windowHandle* is 0,

no Windows messages are sent.

message :   a message you define. When the specified AUXDI event happens, PCIS-DASK passes *message* back to you. *message* can be any value.

In Windows, you can set *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 Microsoft for messages you define.

callbackAddr :   address of the user callback function. PCIS-DASK calls this function when the specified AUXDI event occurs. If you do not want to use a callback function, set *callbackAddr* to 0.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered
ErrorFuncNotSupport

2.2.59   DIO_INT1_EventMessage (Win32 only)

@ Description

Controls the interrupt sources of INT1 of 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.

@ Cards Support

7230, 7233, 7248, 7249, 7296, 7396, 7432, 7433, 8554

@ Syntax

I16 DIO_INT1_EventMessage (U16 CardNumber, I16 Int1Mode, HANDLE
      windowHandle, U32 message, void *callbackAddr())

@ Parameter

CardNumber : The card id of the card that want to be performed this operation.
Int1Mode :   The interrupt mode of INT1. The valid values:

    PCI-7248/cPCI-7248/cPCI-7249R/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
      INT1_ EXT_SIGNAL: INT1 by External Signal
    PCI-7230/cPCI-7230/7233/7432/7433:
      INT1_DISABLE   : INT1 Disabled
      INT1_ EXT_SIGNAL: INT1 by External Signal
      INT1_COUT12 : INT1 by Counter #12
    PCI-8554:
      INT1_DISABLE   : INT1 Disabled
      INT1_COUT12 : INT1 by Counter #12
      INT1_ EXT_SIGNAL: INT1 by External Signal
    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
      INT1_ EXT_SIGNAL: INT1 by External Signal

windowHandle : The handle to the window you want to receive a Windows message in when the specified INT1 event happens. If *windowHandle* is 0, no Windows messages are sent.
message :   a message you define. When the specified INT1 event happens, PCIS-DASK passes *message* back to you. *message* can be any value.

In Windows, you can set *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 Microsoft for messages you define.

callbackAddr : address of the user callback function. PCIS-DASK calls this function when the specified INT1 event occurs. If you do not want to use a callback function, set *callbackAddr* to 0.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered
ErrorFuncNotSupport

2.2.60   DIO_INT2_EventMessage (Win32 only)

@ Description

Controls the interrupt sources of INT2 of 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.

@ Cards Support

7230, 7233, 7248, 7249, 7296, 7396, 7432, 7433, 8554

@ Syntax

I16 DIO_INT2_EventMessage (U16 CardNumber, I16 Int2Mode, HANDLE
        windowHandle, U32 message, void *callbackAddr())

@ Parameter

CardNumber : The card id of the card that want to be performed this operation.
Int2Mode :   The interrupt mode of INT2.The valid values:
                 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
                   INT2_ EXT_SIGNAL: INT2 by External Signal
                 PCI-7230/cPCI-7230/7233/7432/7433/8554:
                   INT2_DISABLE   : INT2 Disabled
                   INT2_ EXT_SIGNAL: INT2 by External Signal
                 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
                   INT2_ EXT_SIGNAL: INT2 by External Signal
windowHandle : The handle to the window you want to receive a Windows message in when the specified INT2 event happens. If *windowHandle* is 0, no Windows messages are sent.
message :   a message you define. When the specified INT2 event happens, PCIS-DASK passes *message* back to you. *message* can be any value.
                 In Windows, you can set *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 Microsoft for messages you define.
callbackAddr : address of the user callback function. PCIS-DASK calls this function when the specified INT2 event occurs. If you do not want to use a callback function, set *callbackAddr* to 0..

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered
ErrorFuncNotSupport

2.2.61　DIO_PortConfig

@ Description

Informs PCIS-DASK library of the port selected and the direction (Input or output)
setting of the selected port.

@ Cards Support

7248, 7249, 7296, 7396

@ Syntax

I16 DIO_PortConfig (U16 CardNumber, U16 Port, U16 Direction)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Port : The port selected. The valid value:

    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-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

Note: 1. The value, Channel_P*n*, for argument *Port* is defined as all of the
ports (Port A, B and C) in channel *n*.

2. If the *port* argument of DIO_PortConfig is set to Channel_P*n*E, the channel *n* will be configured as INPUT_PORT (the argument *Direction* is of no use here) and the digital input of channel *n* is controlled by external clock.

Direction :   The port direction of PIO port. The valid value:
INPUT_PORT
OUTPUT_PORT

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorInvalidIoChannel

2.2.62   DIO_SetCOSInterrupt

@ Description

This functions enable/disables the COS (Change Of State) interrupt detection capability of the specified ports.

@ Cards Support

7396

@ Syntax

I16 DIO_SetCOSInterrupt (U16 CardNumber, U16 Channel_no, U16 ctlA, U16 ctlB, U16 ctlC)

@ Parameter

CardNumber : The card id of the card that want to be performed this operation.
Channel_no : The channel number to be enabled or disabled COS detection capability. The valid port numbers are:
Channel_P1 : Port 1
Channel_P2 : Port 2
Channel_P3 : Port 3
Channel_P4 : Port 4
ctlA :   The control value for Port A of the channel defined by argument *Channel_no*.
The valid values:
0: disabled
1: enabled
ctlB :   The control value for Port B of the channel defined by argument *Channel_no*.
The valid values:
0: disabled
1: enabled
ctlC :   The control value for Port C of the channel defined by argument *Channel_no*.
The valid values:
0: disabled
1: enabled

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered
ErrorFuncNotSupport

2.2.63   DIO_SetDualInterrupt

@ Description

This function Controls the interrupt sources of INT1 and INT2 of Dual Interrupt system and notifies the user's application when an interrupt event occurs. The notification is performed through the user-specified signal handlers.

@ Cards Support

7230, 7233, 7248, 7249, 7296, 7396, 7432, 7433, 8554

@ Syntax

I16 DIO_SetDualInterrupt(U16 wCardNumber, I16 wInt1Mode, I16 wInt2Mode, void
           (*event1_handler)(int), void (*event2_handler)(int))

@ Parameter

CardNumber : The card id of the card that want to be performed this operation.
Int1Mode :   The interrupt mode of INT1. The valid values:
               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-8554:
                 INT1_DISABLE   : INT1 Disabled
                 INT1_ EXT_SIGNAL: INT1 by External Signal
                 INT1_COUT12 : INT1 by Counter #12
               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 :   The interrupt mode of INT2.The valid values:
               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-7230/cPCI-7230/7233/7432/7433/8554:
                 INT2_DISABLE   : INT2 Disabled
                 INT2_ EXT_SIGNAL: INT2 by External Signal
               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
event1_handler :address of the user specified *signal handler*. The signal handler is
               called when the specified INT1 event occurs. If you do not want to
               use a callback function, set *event1_handler* to 0.
event2_handler :address of the user specified *signal handler*. The signal
               handler is called when the specified INT2 event occurs. If you
               do not want to use a callback function, set *event2_handler* to
               0.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered
ErrorFuncNotSupport

2.2.64   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 PostMessage API.

@ Cards Support

7300A

@ Syntax

I16 DIO_T2_EventMessage (U16 CardNumber, I16 T2En, HANDLE windowHandle, U32 message, void *callbackAddr())

@ Parameter

CardNumber : The card id of the card that want to be performed this operation.
T2En : The control value for Timer2 interrupt.
The valid values:
0: disabled
1: enabled
windowHandle : The handle to the window you want to receive a Windows message in when the specified Timer2 event happens. If *windowHandle* is 0, no Windows messages are sent.
message : a message you define. When the specified Timer2 event happens, PCIS-DASK passes *message* back to you. *message* can be any value.
In Windows, you can set *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 Microsoft for messages you define.
callbackAddr : address of the user callback function. 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

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered
ErrorFuncNotSupport

2.2.65   DO_7200_Config

@ Description

Informs PCIS-DASK library of the trigger source and output mode selected for PCI7200/cPCI7200 with card ID *CardNumber*. You must call this function before calling function to perform continuous digital output operation.

@ Cards Support

7200

@ Syntax

I16 DO_7200_Config (U16 CardNumber, U16 TrigSource, U16 OutReqEn, U16 OutTrigSig)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
TrigSource : The trigger source for continuous digital input.
Valid values:
TRIG_INT_PACER: on-board Programmable pacer
TRIG_HANDSHAKE: handshaking
Output REQ Enable :
OREQ_ENABLE: output REQ is enabled, an O_REQ strobe is generated after output data is strobe
OREQ_DISABLE: output REQ is disable
Output Trigger Signal :

OTRIG_HIGH: O_TRIG signal goes high
OTRIG_LOW: O_TRIG signal goes low

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.66   DO_7300A_Config

@ Description

Informs 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.

@ Cards Support

7300A Rev.A

@ Syntax

I16 DO_7300A_Config (U16 CardNumber, U16 PortWidth, U16 TrigSource, U16
        WaitStatus, U16 Terminator, U16 O_REQ_Pol)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
PortWidth :   The width of digital output port (PORT B). The valid value is 0, 8, 16,
                or 32.
TrigSource :   The trigger mode for continuous digital output.
                Valid values:
                  TRIG_INT_PACER: on-board programmable pacer timer1
                  TRIG_CLK_10MHz: 10MHz clock
                  TRIG_CLK_20MHz: 20MHz clock
                  TRIG_HANDSHAKE: handshaking mode
WaitStatus :   DO Wait Status, the 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_REQ_Pol :   O_REQ Polarity. This function is not implemented on PCI-7300A
                Rev.A/cPCI-7300A Rev.A card. You can ignore this argument.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.67   DO_7300B_Config

@ Description

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

@ Cards Support

7300A Rev.B

@ Syntax

I16 DO_7300B_Config (U16 CardNumber, U16 PortWidth, U16 TrigSource, U16
        WaitStatus, U16 Terminator, U16 O_Cntrl_Pol, U32 FifoThreshold)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
PortWidth : The width of digital output port (PORT B). The valid value is 0, 8, 16, or 32.
TrigSource : The trigger mode for continuous digital output.
Valid values:
  TRIG_INT_PACER: on-board programmable pacer timer1
  TRIG_CLK_10MHz: 10MHz clock
  TRIG_CLK_20MHz: 20MHz 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
      10MHz clock as output clock
  TRIG_DO_CLK_20M_ACK: burst handshaking mode by using
      20MHz clock as output clock
WaitStatus : DO Wait Status, the 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 : 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:
  (1) DOREQ
    P7300_DOREQ_POS: DOREQ signal is rising edge active
    P7300_DOREQ_NEG: DOREQ signal is falling edge active
  (2) DOACK
    P7300_DOACK_POS: DOACK signal is rising edge active
    P7300_DOACK_NEG: DOACK signal is falling edge active
  (3) 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

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.68 DO_AsyncCheck

@ Description

Check the current status of the asynchronous digital output operation.

@ Cards Support

7200, 7300A

@ Syntax

I16 DO_AsyncCheck (U16 CardNumber, BOOLEAN *Stopped, U32 *AccessCnt)

@ Parameter

CardNumber : The card id of the card that performs the asynchronous operation.
Stopped : Whether the asynchronous digital output operation has completed. If *Stopped* = TRUE, the digital output operation has stopped. Either 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 :   The number of digital output data that has been written at the time the
call to **DO_AsyncCheck()**.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.69   DO_AsyncClear

@ Description

Stop the asynchronous digital output operation.

@ Cards Support

7200, 7300A

@ Syntax

I16 DO_AsyncClear (U16 CardNumber, U32 *AccessCnt)

@ Parameter

CardNumber : The card id of the card that performs the asynchronous operation.
AccessCnt :   The number of digital output data that has been transferred at the time
the call to **DO_AsyncClear()**.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered,
ErrorFuncNotSupport

2.2.70   DO_ContStatus

@ Description

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

@ Cards Support

7200, 7300A

@ Syntax

I16 DO_ContStatus (U16 CardNumber, U16 *Status)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Status :   The continuous DO status returned. The description of the parameter
Status for various card types is the following:

PCI7200 :
  bit 0 : '1' indicates D/I FIFO is Full (Over-Run)
  bit 1 : '1' indicates D/O FIFO is Empty (Under-Run)
  bit 2 ~ 15 : not used

PCI7300A_RevA:
  bit 0 : '1' indicates DO FIFO is empty during data output and some output
        data were written twice. Writes '1' to clear this bit
  bit 1 : '1' indicates DO FIFO is full
  bit 2 : '1' indicates DO FIFO is empty
  bit 3 ~ 15 : not used

PCI7300A_RevB:
  bit 0 : '1' indicates DO FIFO is empty during data output and some output
        data were written twice. Writes '1' to clear this bit
  bit 1 : '1' indicates DO FIFO is full
  bit 2 : '1' indicates DO FIFO is empty

bit 3 ~ 15 : not used

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered

2.2.71   DO_ContWritePort

@ Description

This function performs continuous digital output on the specified digital output port at a
rate as close to the rate you specified.

@ Cards Support

7200, 7300A

@ Syntax

I16 DO_ContWritePort (U16 CardNumber, U16 Port, void *Buffer, U32 WriteCount,
         U16 Iterations, F32 SampleRate, U16 SyncMode)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Port :   Digital output port number. For PCI-7200/cPCI-7200 and PCI-
                   7300A/cPCI-7300A, this argument must be set to 0.
Buffer :   The starting address of the memory containing the output data. This
                   memory must have been allocated for enough space to store output
                   data.
WriteCount :   the number of output operation to be performed.
Iterations :   the number of times the data in *Buffer* to 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 as **1**.
SampleRate : The 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 and
                   TRIG_DO_CLK_TIMER_ACK) by calling **DO_7200_Config()** or
                   DO_7300_Config(). For the other settings, you have to set this
                   argument as CLKSRC_EXT_SampRate.
SyncMode :   Whether this 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 complete.
                       ASYNCH_OP: asynchronous digital input operation

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport,
ErrorInvalidIoChannel, ErrorTransferCountTooLarge , ErrorContIoNotAllowed

2.2.72   DO_InitialMemoryAllocated

@ Description

This function returns the available memory size for continuous digital output in the
device driver of this card. The continuous digital output transfer size can not exceed
this size.

@ Cards Support

7200, 7300A

@ Syntax

I16 DO_InitialMemoryAllocated (U16 CardNumber, U32 *MemSize)

@ **Parameter**

CardNumber : The card id of the card that want to perform this operation.
MemSize : The available memory size in device driver of this card.
The unit is KB (1024 bytes).

@ **Return Code**

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered

2.2.73 DO_PGStart

@ **Description**

This function performs pattern generation for digital output with the data stored in
Buffer at a rate as close to the rate you specified.

@ **Cards Support**

7300A

@ **Syntax**

I16 DO_PGStart (U16 CardNumber, void *Buffer, U32 WriteCount, F64 SampleRate)

@ **Parameter**

CardNumber : The card id of the card that want to perform this operation.
Buffer : The starting address of the memory containing the output data of
pattern generation. This memory must have been allocated for
enough space to store output data.
WriteCount : the number of pattern generation output samples.
SampleRate : The 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_7300_Config().**

@ **Return Code**

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport,
ErrorTransferCountTooLarge

2.2.74 DO_PGStop

@ **Description**

This function stops pattern generation for digital output operation.

@ **Cards Support**

7300A

@ **Syntax**

I16 DO_PGStop (U16 CardNumber)

@ **Parameter**

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

@ **Return Code**

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.75 DO_ReadLine

@ **Description**

Read back the digital logic state of the specified digital output line in the specified port.

@ **Cards Support**

6208, 6308, 7200, 7248, c7249R, 7296, 7300A, 7396, 7250/51, 7252, 9116, 9118

@ Syntax

I16 DO_ReadLine (U16 CardNumber, U16 Port, U16 Line, U16 *State)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Port : Digital output port number. The valid value:
PCI-6208V/16V/08A: 0
PCI-6308V/08A: 0
PCI-7200: 0
cPCI-7200: 0, 1 (auxiliary output port)
PCI-7250/51: 0 through 3
cPCI-7252: 0
cPCI-9116: 0
PCI-9118DG/HG/HR: 0
PCI-7300A/cPCI-7300A: 1 (auxiliary output port)
PCI-7248/96, cPCI-7249R, PCI-7396: refer to the function
DI_ReadLine section.
Line : The digital line to be accessed. The valid value:
PCI-6208V/16V/08A: 0 through 3
PCI-6308V/08A: 0 through 3
PCI-7200/cPCI-7200: 0 through 31 (for port 0)
0 through 3 (auxiliary output port of cPCI-7200)
PCI-7250/51: 0 through 7
cPCI-7252: 0 through 7
PCI-7300A/cPCI-7300A: 0 through 3
cPCI-9116: 0 through 7
PCI-9118DG/HG/HR: 0 through 3
PCI-7248/96, cPCI-7249R, PCI-7396: refer to the function
DI_ReadLine section.
State : Returns the digital logic state, 0 or 1, of the specified line.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport,
ErrorInvalidIoChannel

2.2.76   DO_ReadPort

@ Description

Read back the output digital data from the specified digital output port.

@ Cards Support

6208, 6308, 7200, 7248, c7249R, 7296, 7300A, 7396, 7250/51, 7252, 9116, 9118

@ Syntax

I16 DO_ReadPort (U16 CardNumber, U16 Port, U32 *Value)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Port : Digital output port number. The valid value:
PCI-6208V/16V/08A: 0
PCI-6308V/08A: 0
PCI-7200: 0
cPCI-7200: 0, 1 (auxiliary output port)
PCI-7250/51: 0 through 3
cPCI-7252: 0
PCI-9118DG/HG/HR: 0
cPCI-9116: 0
PCI-7300A/cPCI-7300A: 1 (auxiliary output port)
PCI-7248/96, cPCI-7249R, PCI-7396: refer to the function
DI_ReadPort section.
Value : Returns the digital data read from the specified output port.

<pre>
                    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-7250/51: 8-bit data
                    cPCI-7252: 8-bit data
                    PCI-7300A/cPCI-7300A: 4-bit data
                    cPCI-9116: 8-bit data
                    PCI-9118DG/HG/HR: 4-bit data
                    PCI-7248/96, cPCI-7249R, PCI-7396: refer to the function
                    DI_ReadPort section.
</pre>

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, ErrorInvalidIoChannel

2.2.77   DO_Write ExtTrigLine

@ Description

Sets the digital output trigger line to the specified state. This function is only available for PCI-7200.

@ Cards Support

7200

@ Syntax

I16 DO_WriteExtTrigLine (U16 CardNumber, U16 Value)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Value :   The new digital logic state, 0 or 1.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.78   DO_WriteLine

@ Description

Sets the specified digital output line in the specified digital port to the specified state. This function is only available for these cards that support digital output read-back functionality.

@ Cards Support

6208, 6308, 7200, 7248, c7249R, 7296, 7300A, 7396, 7250/51, 7252, 9116, 9118

@ Syntax

I16 DO_WriteLine (U16 CardNumber, U16 Port, U16 Line, U16 State)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Port :   Digital output port number. The valid value:
<pre>
                    PCI-6208V/16V/08A: 0
                    PCI-6308V/08A: 0
                    PCI-7200: 0
                    cPCI-7200: 0, 1 (auxiliary output port)
                    PCI-7250/51: 0 through 3
                    cPCI-7252: 0
                    PCI-9118DG/HG/HR: 0
                    cPCI-9116: 0
                    PCI-7300A/cPCI-7300A: 1 (auxiliary output port)
</pre>

PCI-7248/96, cPCI-7249R, PCI-7396: refer to the function
DI_ReadLine section.
Line : The digital line to write to. The valid value:
PCI-6208V/16V/08A: 0 through 3
PCI-6308V/08A: 0 through 3
PCI-7200/cPCI-7200: 0 through 31(for port 0)
                      : 0 through 3 (auxiliary output port of cPCI-7200)
PCI-7250/51: 0 through 7
cPCI-7252: 0 through 7
PCI-7300A/cPCI-7300A: 0 through 3
PCI-9118DG/HG/HR: 0 through 3
cPCI-9116: 0 through 7
PCI-7248/96, cPCI-7249R, PCI-7396: refer to the function
DI_ReadLine section.
State : The new digital logic state, 0 or 1.

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport,
ErrorInvalidIoChannel

2.2.79   DO_WritePort

@ Description

Writes digital data to the specified digital output port.

@ Cards Support

6208V/16V/08A, 6308V/08A, 7200, 7230, 7234, 7248, 7249, 7250/51, 7252, 7296,
7300A, 7349, 7432, 7433, 7434, 8554, 9111, 9112, 9116, 9118

@ Syntax

I16 DO_WritePort (U16 CardNumber, U16 Port, U32 Value)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
Port : Digital output port number. The cards that support this function and
their corresponding valid value are as follows:
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-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 through 3
cPCI-7252: 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-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-7433R: P7433R_DO_LED
                PCI-7434/cPCI-7434: PORT_DO_LOW, PORT_DO_HIGH
                cPCI-7434R: P7434R_DO_LED
                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

Note: The value, Channel_P*n*, 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-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-7248/cPCI-7248: 8-bit data
                cPCI-7249R: 8-bit data
                PCI-7250/51: 8-bit data
                cPCI-7252: 8-bit data
                PCI-7296: 8-bit data
                PCI-7300A/cPCI-7300A: 4-bit data
                PCI-7396: 24-bit data (for Channel_P*n*T, where *n* is the channel number) or
                            8-bit data (for Channel_P*n*A, Channel_P*n*B, Channel_P*n*C , 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-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

@ Return Code

 NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered
 ErrorFuncNotSupport, ErrorInvalidIoChannel

2.2.80 EDO_9111_Config

@ Description

Informs PCIS-DASK library of the mode of EDO channels for the PCI-9111 card with card ID *CardNumber*.

@ Cards Support

9111

@ Syntax

I16 EDO_9111_Config (U16 CardNumber, U16 EDO_Fun)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
EDO_Fun:　　　The mode of EDO ports. The valid modes are:
　　　　　　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

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.81 GCTR_Read

@ Description

Reads the counter value of the general-purpose counter without disturbing the counting process.

@ Cards Support

9116

@ Syntax

I16 GCTR_Read (U16 CardNumber, U16 GCtr, U32 *Value)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
GCtr :　　　The counter number.
　　　　　　Range: 0 for PCI-9116
Value :　　　Returns the counter value of the specified general-purpose
　　　　　　timer/counter.
　　　　　　Range: 0 through 65536

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, InvalidCounter

2.2.82 GCTR_Clear

@ Description

Turns off the specified general-purpose timer/counter operation and reset the counter value to zero.

@ Cards Support

9116

@ Syntax

I16 GCTR_Clear (U16 CardNumber, U16 GCtr)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
GCtr : The counter number.
Range: 0 for PCI-9116

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, InvalidCounter

2.2.83 GCTR_Setup

@ Description

Controls the operation of the selected counter/timer.

@ Cards Support

9116

@ Syntax

I16 GCTR_Setup (U16 CardNumber, U16 GCtr, U16 GCtrCtrl, U32 Count)

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
GCtr : The counter number.
Range: 0 for cPCI-9116.
GCtrCtrl : The setting for 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 four groups of constants:
(1) **Timer/Counter Mode**
General_Counter: General counter
Pulse_Generation: Generation of pulse
(2) **Timer/Counter Source**
GPTC_CLKSRC_INT : internal time base
GPTC_CLKSRC_EXT : external time base from GP_TC_CLK pin
(3) **Timer/Counter Gate Source**
GPTC_GATESRC_INT : gate is controlled by software
GPTC_GATESRC_EXT: gate is controlled by GP_TC_GATE pin
(4) **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
(5) **Timer/Counter UpDown Control**
GPTC_DOWN_CTR: counting direction is down
GPTC_UP_CTR: counting direction is up
(6) **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 : The counter value of general-purpose timer/counter

@ Return Code

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport, InvalidCounter

2.2.84 GetActualRate

@ Description

Gets the actual sampling rate the hardware will perform according to the board type and the rate you want.

@ **Cards Support**

7200, 7300A, 9111, 9112, 9113, 9114, 9118, 9812/10

@ **Syntax**

  I16 GetActualRate (U16 CardNumber, F64 SampleRate, F64 *ActualRate)

@ **Parameter**

CardNumber : The card id of the card that wants to perform this operation.
SampleRate: The desired sampling rate.
ActualRate: Returns the actual acquisition rate performed. The value depends on
                 the card type and the desired sampling rate.

@ **Return Code**

NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.85　Register_Card

@ **Description**

Initializes the hardware and software states of a NuDAQ PCI-bus data acquisition
card, and then returns a numeric card ID that corresponds to the card initialized.
Register_Card must be called before any other PCIS-DASK library functions can be
called for that card. The function initializes the card and variables internal to PCIS-
DASK library. Because NuDAQ PCI-bus data acquisition cards meets the plug-and-
play design, the base address (pass-through address) and IRQ level are assigned by
system BIOS directly.

@ **Cards Support**

6208V/6216V, 6208A, 6308V, 6308A, 7200, 7230, 7233, 7234, 7248, 7249, 7250,
7252, 7296, 7300A, 7396, 7432, 7433, 7434, 8554, 9111, 9112, 9113, 9114, 9116,
9118, 9812/10

@ **Syntax**

  I16 Register_Card (U16 CardType, U16 card_num)

@ **Parameter**

CardType :   The type of card to be initialized. ADLink will periodically upgrades
             PCIS-DASK to add support for new NuDAQ PCI-bus data acquisition
             cards and NuIPC CompactPCI cards. Please refer to *Release Notes*
             for the card types that the current release of PCIS-DASK actually
             supports. Following are the constants defined in DASK.H that
             represent the NuDAQ PCI-bus data acquisition cards that DASK
             supports currently or in the near future:
                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-7248/cPCI-7248)
                PCI_7249 (for cPCI-7249R)
                PCI_7250
                PCI_7252 (for cPCI-7252)
                PCI_7296
                PCI_7300A_RevA (for PCI_7300A_RevA/
                                    cPCI_7300A_RevA)
                PCI_7300A_RevB (for PCI_7300A_RevB/
                                    cPCI_7300A_RevB)
                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_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_9810 (for PCI-9810)
PCI_9812 (for PCI-9812)

card_num : The sequence number of the card with ***the same card type*** (as defined in argument *CardType*) or belonging to ***the same card type series*** (Except PCI-7300A_RevA and PCI-7300A_RevB) 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 most prior slot) is with card_num=0. For example, if there are one PCI-9111DG card (in the first PCI slot) and one PCI-9111HR card and two PCI-9112 cards plugged on your PC, the PCI-9111DG card should be registered with card_num=0, and the PCI-9111HR card with card_num=1. The PCI-9112 card in the prior slot should be registered with card_num=0, and the other one with card_num=1.

The following table categories the NuDAQ PCI devices by card type series.

Card Type Series	Device Type
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-7248/cPCI-7248	PCI-7248/cPCI-7248
PCI-7249	cPCI-7249R
PCI-7250	PCI-7250
PCI-7252	cPCI-7252
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-7396	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-8554	PCI-8554

PCI-9111 Series	PCI-9111DG, PCI-9111HR
PCI-9112/cPCI-9112	PCI-9112/cPCI-9112
PCI-9113	PCI-9113
PCI-9114 Series	PCI-9114DG, PCI-9114HG
PCI-9116	cPCI-9116
PCI-9118 Series	PCI-9118DG, PCI-9118HG, PCI-9118HR
PCI-9812 Series	PCI-9812, PCI-9810

@ Return Code

This function returns a numeric card id for the card initialized. The range of card id is between 0 and 31. If there is any error occurs, it will return negative error code, the possible error codes are listed below:
ErrorTooManyCardRegistered, ErrorUnknownCardType, ErrorOpenDriverFailed, ErrorOpenEventFailed

2.2.86   Release_Card

@ Description

There are at most 32 cards that can be registered simultaneously. This function is used to tell PCIS-DASK library that this registered card is not used currently and can be released. This would make room for new card to register. Also by the end of a program, you need to use this function to release all cards that were registered.

@ Cards Support

6208V/6216V, 6208A, 6308V, 6308A, 7200, 7230, 7233, 7234, 7248, 7249, 7250/51, 7252, 7296, 7300A, 7396, 7432, 7433, 7434, 8554, 9111, 9112, 9113, 9114, 9116, 9118, 9812/10

@ Syntax

  I16 Release_Card (U16 CardNumber)

@ Parameter

CardNumber : The card id of the card that want to be released.

@ Return Code

NoError

Appendix A Status 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	There have been 32 cards that were 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 be ended with or started 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	Time out for AD operation
-25	ErrorNoAsyncAI	Continuous Analog Input is not set as Asynchronous mode
-26	ErrorNoAsyncAO	Continuous Analog Output is not set as Asynchronous mode
-27	ErrorNoAsyncDI	Continuous Digital Input is not set as Asynchronous mode
-28	ErrorNoAsyncDO	Continuous Digital Output is not set as Asynchronous mode
-29	ErrorNotInputPort	The value of AI/DI port argument is invalid
-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 is failed
-36	ErrorInvalidPortDirection	The value of DIO port direction argument is invalid
-37	ErrorBeginThreadError	Failed to create thread
-38	ErrorInvalidPortWidth	The 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 is failed
-42	ErrorDaVoltageOutOfRange	The value of DA voltage argument is out of range
-201	ErrorConfigIoctl	The configuration API is failed
-202	ErrorAsyncSetIoctl	The async. mode API is failed
-203	ErrorDBSetIoctl	The double-buffer setting API is failed
-204	ErrorDBHalfReadyIoctl	The half-ready API is failed
-205	ErrorContOPIoctl	The continuous data acquisition API is failed
-206	ErrorContStatusIoctl	The continuous data acquisition status API setting is failed
-207	ErrorPIOIoctl	The polling data API is failed
-208	ErrorDIntSetIoctl	The dual interrupt setting API is failed
-209	ErrorWaitEvtIoctl	The wait event API is failed
-210	ErrorOpenEvtIoctl	The open event API is failed
-211	ErrorCOSIntSetIoctl	The cos interrupt setting API is failed
-212	ErrorMemMapIoctl	The memory mapping API is failed
-213	ErrorMemUMapSetIoctl	The memory Unmapping API is failed
-214	ErrorCTRIoctl	The counter API is failed

Appendix B   AI Range Codes

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_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

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-9812/10   : AD_B_1_V, AD_B_5_V

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 * channel no. (CH#) * A/D converted data (ND) * Value returned from AI function (OD)
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 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: don't care	One-Shot AI	ND = OD & 0x0FFF
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: don't care	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: don't care	Continuous AI	CH# = (OD >>16) & 0x1F ND = OD & 0xFFFF
cPCI-9116	Every 16-bit signed integer data: D15 D14 D13 .... D1 D0	One-Shot AI Continuous AI	ND = OD

	where D15, D14, ... , D0 : A/D converted data		
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 .... 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-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
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

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 as the figure below:

Header
ChannelRange (Optional)
DAQ data
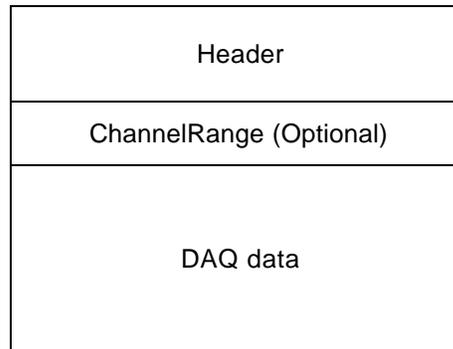
Header

The *header* part records the information related to the stored data and its total length is 60 bytes. The data structure of the file header is as follows:

Header			*Total Length: 60 bytes*
Elements	**Type**	**Size (bytes)**	**Comments**
ID	char	10	file ID *ex. ADLinkDAQ1*
card_type	short	2	card Type *ex. Pci7250, Pci9112*
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
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 Appexdix 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 & 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 as the following formula:

Total Length = *2* * num_of_channel_range bytes

The data structure of each ChannelRange unit is as follows:

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 Appexdix B *ex. 0 (AD_B_5V)*

Data Block

The last part is the data block. The data is written to file in 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 can't be read in normal text editor. You can use any binary file editor to view it or the functions used for reading files, e.g. fread, to get the file information and data value. PCIS-DASK provides a useful utility *DAQCvt* for you to convert the binary file.

Appendix E Function Support

This appendix shows which data acquisition hardware each PCIS-DASK function supports.

Function	PCI-6208A	PCI-6208V\6216V	PCI-6308A	PCI-6308V	PCI-7200	PCI-7230	PCI-7233	PCI-7234	PCI-7250\7251\7252	PCI-7248\7249\7296	PCI-7396	PCI-7300A RevA	PCI-7300A RevB	PCI-7432	PCI-7433	PCI-7434	PCI-8554	PCI-9111	PCI-9112	PCI-9113	PCI-9114	PCI-9116	PCI-9118	PCI-9812\9810
AI_9111_Config																		●						
AI_9112_Config																			●					
AI_9113_Config																				●				
AI_9114_Config																					●			
AI_9116_Config																						●		
AI_9116_CounterInterval																						●		
AI_9118_Config																							●	
AI_9812_Config																								●
AI_AsyncCheck																		●	●	●	●	●	●	●
AI_AsyncClear																		●	●	●	●	●	●	●
AI_AsyncDblBufferHalfReady																		●	●	●	●	●	●	●
AI_AsyncDblBufferMode																		●	●	●	●	●	●	●
AI_AsyncDblBufferTransfer																		●	●	●	●	●	●	●
AI_ContReadChannel																		●	●	●	●	●	●	●
AI_ContReadMultiChannels																						●	●	
AI_ContScanChannels																		●	●	●	●	●	●	●
AI_ContReadChannelToFile																		●	●	●	●	●	●	●
AI_ContReadMultiChannelsToFile																						●	●	
AI_ContScanChannelsToFile																		●	●	●	●	●	●	●
AI_ContStatus																		●	●	●	●	●	●	●
AI_ContVScale																		●	●	●	●	●	●	●
AI_InitialMemoryAllocated																		●	●	●	●	●	●	●
AI_ReadChannel																		●	●	●	●	●	●	
AI_VReadChannel																		●	●	●	●	●	●	
AI_VScale																		●	●	●	●	●	●	
AO_6208A_Config	●																							
AO_6308A_Config			●																					
AO_6308V_Config			●																					
AO_9111_Config																		●						
AO_9112_Config																			●					
AO_VScale	●	●	●	●														●	●				●	
AO_VWriteChannel	●	●	●	●														●	●				●	
AO_WriteChannel	●	●	●	●														●	●				●	
CTR_8554_CK1_Config																	●							
CTR_8554_ClkSrc_Config																	●							
CTR_8554_Debounce_Config																	●							
CTR_Read									●	●							●	●	●	●	●	●	●	
CTR_Reset									●	●							●		●				●	
CTR_Setup									●	●							●		●				●	

Function	PCI-6208A	PCI-6208V/6216V	PCI-6308A	PCI-6308V	PCI-7200	PCI-7230	PCI-7233	PCI-7234	PCI-7250/7251/7252	PCI-7248/7249/7296	PCI-7396	PCI-7300A RevA	PCI-7300A RevB	PCI-7432	PCI-7433	PCI-7434	PCI-8554	PCI-9111	PCI-9112	PCI-9113	PCI-9114	PCI-9116	PCI-9118	PCI-9812/9810
DI_7200_Config					●																			
DI_7300A_Config												●												
DI_7300B_Config													●											
DI_AsyncCheck					●							●	●											
DI_AsyncClear					●							●	●											
DI_AsyncDblBufferHalfReady					●																			
DI_AsyncDblBufferMode					●																			
DI_AsyncDblBufferTransfer					●																			
DI_AsyncMultiBufferNextReady												●	●											
DI_ContMultiBufferSetup												●	●											
DI_ContMultiBufferStart												●	●											
DI_ContReadPort					●							●	●											
DI_ContReadPortToFile					●							●	●											
DI_ContStatus					●							●	●											
DI_InitialMemoryAllocated					●							●	●											
DI_ReadLine	●	●	●	●	●	●	●		●	●	●	●	●	●	●		●	●	●		●	●	●	
DI_ReadPort	●	●	●	●	●	●	●		●	●	●	●	●	●	●		●	●	●		●	●	●	
DIO_7300SetInterrupt												●	●											
DIO_AUXDI_EventMessage												●	●											
DIO_INT1_EventMessage					●	●			●	●				●	●									
DIO_INT2_EventMessage					●	●			●	●				●	●									
DIO_PortConfig										●	●													
DIO_SetCOSInterrupt													●	●	●									
DIO_SetDualInterrupt					●	●			●	●				●	●									
DIO_T2_EventMessage												●	●											
DO_7200_Config					●																			
DO_7300A_Config												●												
DO_7300B_Config													●											
DO_ContStatus					●							●	●											
DO_ContWritePort					●							●	●											
DO_AsyncCheck					●							●	●											
DO_AsyncClear					●							●	●											
DO_InitialMemoryAllocated					●							●	●											
DO_PGStart												●	●											
DO_PGStop												●	●											
DO_ReadLine	●	●	●	●	●				●	●	●	●	●									●	●	
DO_ReadPort	●	●	●	●	●				●	●	●	●	●									●	●	
DO_WriteLine	●	●	●	●	●				●	●	●	●	●									●	●	
DO_WritePort	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●
EDO_9111_Config																		●						
GCTR_Read																						●		
GCTR_Reset																						●		
GCTR_Setup																						●		
GetActualRate					●							●	●					●	●	●	●		●	●
Register_Card	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Function	PCI-6208A	PCI-6208V\6216V	PCI-6308A	PCI-6308V	PCI-7200	PCI-7230	PCI-7233	PCI-7234	PCI-7250\7251\7252	PCI-7248\7249\7296	PCI-7396	PCI-7300A RevA	PCI-7300A RevB	PCI-7432	PCI-7433	PCI-7434	PCI-8554	PCI-9111	PCI-9112	PCI-9113	PCI-9114	PCI-9116	PCI-9118	PCI-9812\9810
Release_Card	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●