



**ADLINK**  
TECHNOLOGY INC.

## **D2K-DASK**

Software Development Kit for  
DAQ-2000 Data Acquisition Cards

### **Function Reference Manual**

**Manual Rev.** 2.02  
**Revision Date:** June 29, 2010  
**Part No:** 50-11226-2010



Recycled Paper

**Advance Technologies; Automate the World.**

Copyright 2010 ADLINK TECHNOLOGY INC.

All Rights Reserved.

## **Disclaimer**

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

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

This document contains proprietary information protected by copyright. All rights are reserved. No part of this manual may be reproduced by any mechanical, electronic, or other means in any form without prior written permission of ADLINK.

## **Trademark Information**

NuDAQ is a registered trademark of ADLINK Technology Inc.

Product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

# Getting service

Contact us should you require any service or assistance.

## **ADLINK Technology Inc.**

Address: 9F, No.166 Jian Yi Road, Chungho City,  
Taipei County 235, Taiwan  
台北縣中和市建一路 166 號 9 樓

Tel: +886-2-8226-5877  
Fax: +886-2-8226-5717  
Email: [service@adlinktech.com](mailto:service@adlinktech.com)

## **Ampro ADLINK Technology Inc.**

Address: 5215 Hellyer Avenue, #110, San Jose, CA 95138, USA

Tel: +1-408-360-0200  
Toll Free: +1-800-966-5200 (USA only)  
Fax: +1-408-360-0222  
Email: [info@adlinktech.com](mailto:info@adlinktech.com)

## **ADLINK Technology Beijing**

Address: 北京市海淀区上地东路 1 号盈创动力大厦 E 座 801 室  
(100085)  
Rm. 801, Power Creative E, No. 1, B/D  
Shang Di East Rd., Beijing 100085, China

Tel: +86-10-5885-8666  
Fax: +86-10-5885-8625  
Email: [market@adlinktech.com](mailto:market@adlinktech.com)

## **ADLINK Technology Shanghai**

Address: 上海市漕河泾高科技开发区钦江路 333 号 39 幢 4 层  
(200233)

Tel: +86-21-6495-5210  
Fax: +86-21-5450-0414  
Email: [market@adlinktech.com](mailto:market@adlinktech.com)

## **ADLINK Technology Shenzhen**

Address: 深圳市南山区科技园南区高新南七道 数字技术园  
A1 栋 2 楼 C 区 (518057)  
2F, C Block, Bld. A1, Cyber-Tech Zone,  
Gao Xin Ave. Sec 7, High-Tech Industrial Park S.,  
Shenzhen, 518054 China

Tel: +86-755-2643-4858  
Fax: +86-755-2664-6353  
Email: [market@adlinktech.com](mailto:market@adlinktech.com)

**ADLINK Technology Inc. (German Liaison Office)**

Address: Nord Carree 3, 40477 Duesseldorf, Germany

Tel: +49-211-495-5552

Fax: +49-211-495-5557

Email: [emea@adlinktech.com](mailto:emea@adlinktech.com)

**ADLINK (French Liaison Office)**

Address: 15 rue Emile Baudot, 91300 MASSY Cedex, France

Tel: +33 (0) 1 60 12 35 66

Fax: +33 (0) 1 60 12 35 66

Email: [france@adlinktech.com](mailto:france@adlinktech.com)

**ADLINK Technology Japan Corporation**

Address: 151-0072 東京都渋谷区幡ヶ谷

1-1-2 朝日生命幡ヶ谷ビル 8F

Asahiseimei Hatagaya Bldg. 8F

1-1-2 Hatagaya, Shibuya-ku, Tokyo 151-0072, Japan

Tel: +81-3-4455-3722

Fax: +81-3-5333-6040

Email: [japan@adlinktech.com](mailto:japan@adlinktech.com)

**ADLINK Technology Inc. (Korean Liaison Office)**

Address: 서울시 서초구 서초동 1506-25 한도 B/D 2 층

2F, Hando B/D, 1506-25, Seocho-Dong,

Seocho-Gu, Seoul, 137-070, Korea

Tel: +82-2-2057-0565

Fax: +82-2-2057-0563

Email: [korea@adlinktech.com](mailto:korea@adlinktech.com)

**ADLINK Technology Singapore Pte Ltd.**

Address: 84 Genting Lane #07-02A, Cityneon Design Centre,

Singapore 349584

Tel: +65-6844-2261

Fax: +65-6844-2263

Email: [singapore@adlinktech.com](mailto:singapore@adlinktech.com)

**ADLINK Technology Singapore Pte Ltd. (Indian Liaison Office)**

Address: No. 1357, "Anupama", Sri Aurobindo Marg, 9th Cross,

JP Nagar Phase I, Bangalore - 560078, India

Tel: +91-80-65605817

Fax: +91-80-22443548

Email: [india@adlinktech.com](mailto:india@adlinktech.com)

# Using this manual

## Audience and scope

This manual guides you when using the D2K-DASK software driver for DAQ-2000 data acquisition cards. This manual also describes how to install and use the D2K-DASK function library when creating programs for your software applications.

## How this manual is organized

This manual is organized as follows:

**Chapter 1 Introduction:** This chapter introduces the D2K-DASK, the fundamentals of building Windows®-based applications, and describes the classes of functions that the D2K-DASK supports.

**Chapter 2 Function Reference:** This section provides detailed description of each function call that the D2K-DASK provides.

**Appendix:** This chapter provides references on status codes, AI range codes, AI data format, and supported functions.

## Conventions

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

---

<b>NOTE</b>	Additional information, aids, and tips that help you perform particular tasks.
-------------	--

---

---

<b>IMPORTANT</b>	Critical information and instructions that you <b>MUST</b> perform to complete a task.
------------------	--

---

---

<b>WARNING</b>	Information that prevents physical injury, data loss, module damage, program corruption etc. when trying to complete a particular task.
----------------	---

---

<b>List of Tables.....</b>	<b>v</b>
----------------------------	----------

<b>List of Figures .....</b>	<b>vi</b>
------------------------------	-----------

<b>1 Introduction .....</b>	<b>1</b>
-----------------------------	----------

1.1 Application Building Fundamentals in Windows .....	2
Using Microsoft® Visual C®/C++® .....	2
Using Microsoft® Visual Basic® .....	2
1.2 Application Building Fundamentals in Linux .....	5
1.3 Function Classes .....	6

<b>2 Function Reference.....</b>	<b>7</b>
----------------------------------	----------

2.1 Data Types .....	7
2.2 Function Reference .....	8
D2K_AI_AsyncCheck .....	8
D2K_AI_AsyncClear .....	10
D2K_AI_AsyncDblBufferHalfReady .....	11
D2K_AI_AsyncDblBufferHandled .....	12
D2K_AI_AsyncDblBufferMode .....	13
D2K_AI_AsyncDblBufferOverrun .....	14
D2K_AI_AsyncDblBufferToFile .....	15
D2K_AI_AsyncReTrigNextReady	
D2K_AI_AsyncReTrigNextReadyEx .....	16
D2K_AI_CH_Config .....	18
D2K_AI_Config	
D2K_AI_ConfigEx .....	21
D2K_AI_ContBufferReset .....	26
D2K_AI_ContBufferSetup .....	27
D2K_AI_ContMuxScan .....	28
D2K_AI_ContMuxScanToFile .....	31
D2K_AI_ContReadChannel .....	35
D2K_AI_ContReadChannelToFile .....	39
D2K_AI_ContReadMultiChannels .....	44
D2K_AI_ContReadMultiChannelsToFile .....	49
D2K_AI_ContScanChannels .....	54
D2K_AI_ContScanChannelsToFile .....	60
D2K_AI_ContStatus .....	65
D2K_AI_ContVScale .....	67
D2K_AI_DelayTrig_Config	
D2K_AI_DelayTrig_ConfigEx .....	69

D2K_AI_EventCallBack .....	73
D2K_AI_InitialMemoryAllocated .....	75
D2K_AI_MiddleTrig_Config .....	
D2K_AI_MiddleTrig_ConfigEx .....	76
D2K_AI_MuxScanSetup .....	80
D2K_AI_PostTrig_Config .....	
D2K_AI_PostTrig_ConfigEx .....	82
D2K_AI_PreTrig_Config .....	
D2K_AI_PreTrig_ConfigEx .....	86
D2K_AI_ReadChannel .....	90
D2K_AI_ReadMuxScan .....	91
D2K_AI_ScanReadChannels .....	92
D2K_AI_SimuReadChannel .....	94
D2K_AI_VoltScale .....	96
D2K_AI_VReadChannel .....	97
D2K_AIO_Config .....	99
D2K_AO_AsyncCheck .....	102
D2K_AO_AsyncClear .....	103
D2K_AO_AsyncDbfBufferHalfReady .....	104
D2K_AO_AsyncDbfBufferMode .....	105
D2K_AO_CH_Config .....	106
D2K_AO_Config .....	108
D2K_AO_ContBufferCompose .....	112
D2K_AO_ContBufferComposeAll .....	114
D2K_AO_ContBufferReset .....	116
D2K_AO_ContBufferSetup .....	117
D2K_AO_ContStatus .....	118
D2K_AO_ContWriteChannel .....	120
D2K_AO_ContWriteMultiChannels .....	122
D2K_AO_DelayTrig_Config .....	125
D2K_AO_EventCallBack .....	129
D2K_AO_Group_FIFOLoad .....	131
D2K_AO_Group_Setup .....	133
D2K_AO_Group_Update .....	135
D2K_AO_Group_VUpdate .....	136
D2K_AO_Group_WFM_AsyncCheck .....	137
D2K_AO_Group_WFM_AsyncClear .....	139
D2K_AO_Group_WFM_Start .....	141
D2K_AO_Group_WFM_StopConfig .....	145
D2K_AO_InitialMemoryAllocated .....	147



D2K_AO_PostTrig_Config .....	148
D2K_AO_SimuWriteChannel .....	151
D2K_AO_VoltScale .....	153
D2K_AO_VWriteChannel .....	154
D2K_AO_WriteChannel .....	156
D2K_DB_Auto_Calibration_ALL .....	157
D2K_DI_ReadLine .....	158
D2K_DI_ReadPort .....	159
D2K_DIO_LineConfig .....	160
D2K_DIO_LinesConfig .....	161
D2K_DIO_PortConfig .....	162
D2K_DO_ReadLine .....	163
D2K_DO_ReadPort .....	164
D2K_DO_WriteLine .....	165
D2K_DO_WritePort .....	166
D2K_EEPROM_CAL_Constant_Update .....	167
D2K_GCTR_Control .....	168
D2K_GCTR_Read .....	169
D2K_GCTR_Reset .....	170
D2K_GCTR_Setup .....	
D2K_GCTR_SetupEx .....	171
D2K_GCTR_Status .....	175
D2K_Get_Default_Load_Area .....	177
D2K_GetPXISlotGeographAddr .....	178
D2K_Load_CAL_Data .....	179
D2K_Register_Card .....	180
D2K_Register_Card_By_PXISlot_GA .....	182
D2K_Release_Card .....	184
D2K_Route_Signal .....	185
D2K_Set_Default_Load_Area .....	188
D2K_Signal_DisConn .....	189
D2K_SoftTrigGen .....	191
D2K_SSI_SourceClear .....	192
D2K_SSI_SourceConn .....	193
D2K_SSI_SourceDisConn .....	195
DAQ2005_Acquire_AD_Error .....	196
DAQ2005_Acquire_DA_Error .....	197
DAQ2006_Acquire_AD_Error .....	198
DAQ2006_Acquire_DA_Error .....	199
DAQ2010_Acquire_AD_Error .....	200
DAQ2010_Acquire_DA_Error .....	201

DAQ2016_Acquire_AD_Error .....	202
DAQ2016_Acquire_DA_Error .....	203
DAQ2204_Acquire_AD_Error .....	204
DAQ2204_Acquire_DA_Error .....	205
DAQ2205_Acquire_AD_Error .....	206
DAQ2205_Acquire_DA_Error .....	207
DAQ2206_Acquire_AD_Error .....	208
DAQ2206_Acquire_DA_Error .....	209
DAQ2208_Acquire_AD_Error .....	210
DAQ2213_Acquire_AD_Error .....	211
DAQ2214_Acquire_AD_Error .....	212
DAQ2214_Acquire_DA_Error .....	213
DAQ250X_Acquire_AD_Error .....	214
DAQ250X_Acquire_DA_Error .....	215

## Appendix 217

.....	Status Codes	217
.....	AI Range Codes	220
.....	AI Data Format	222
.....	Data File Format	223
Header .....		223
ChannelRange .....		224
Data Block .....		225

# List of Tables

Table 3-1:	Status Codes .....	217
Table 3-2:	AI Range Codes .....	220
Table 3-3:	Valid Card AI Range .....	221
Table 3-4:	AI Data Format .....	222



## List of Figures

Figure 2-1: Scan Timing (D2K_AI_ContMuxScan) .....	30
Figure 2-2: Scan Timing (D2K_AI_ContMuxScanToFile) .....	33
Figure 2-3: Scan Timing (D2K_AI_ContReadChannel) .....	37
Figure 2-4: Scan Timing (D2K_AI_ContReadChannelToFile) ...	42
Figure 2-5: Scan Timing (D2K_AI_ContReadMultiChannels)....	47
Figure 2-6: Scan Timing (D2K_AI_ContReadMultiChannelsToFile) 52	
Figure 2-7: Scan Timing (D2K_AI_ContScanChannels).....	57
Figure 2-8: Scan Timing (D2K_AI_ContScanChannelsToFile)..	63



# 1 Introduction

The D2K-DASK is a software driver for DAQ-2000 data acquisition cards. It is a high performance data acquisition driver for developing custom applications under Windows<sup>®</sup> and Linux environments.

Using D2K-DASK lets you enjoy the advantages of the power and flexibility of Windows<sup>®</sup> and Linux for your data acquisition applications. These include running multiple applications and using extended memory. In addition, implementing D2K-DASK under Visual Basic environment makes it easy to create custom user interfaces and graphics.

## 1.1 Application Building Fundamentals in Windows

The following sections provide fundamental instructions when using D2K-DASK to build application in Windows® 98/NT/2000/XP/Vista operating environment.

### Using Microsoft® Visual C®/C++®

Follow these steps to create a data acquisition application using D2K-DASK and Microsoft Visual C/C++.

1. Launch the Microsoft Visual C/C++ application.
2. Open a new or existing project that you want to apply the D2K-DASK.
3. Include header file D2KDASK.H in the C/C++ source files that call D2K-DASK functions. D2KDASK.H contains all the function declarations and constants that can be used to develop data acquisition applications. Incorporate the following statement in the code to include the header file.

```
#include "D2KDASK.H"
```

4. After setting the appropriate compile and link options, build the application by selecting the Build command from Build menu. Remember to link D2K-DASK's import library, D2K-DASK.LIB.

### Using Microsoft® Visual Basic®

Follow the steps in the succeeding sections to create a data acquisition application using D2K-DASK and Visual Basic.

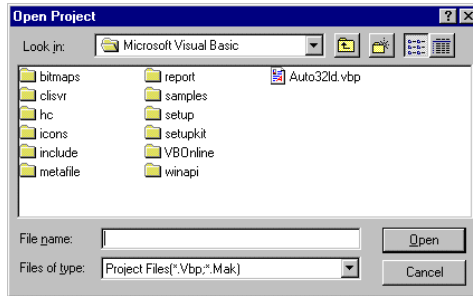
#### Open a project

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

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



select the Open Project command from the File menu to display the Open Project dialog box.



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

### Add the file

You must add the file **D2KDASK.BAS** to the project, if the file is not yet included. This file contains all the procedure declarations and constants that can be used to develop the data acquisition application. To add the file:

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




2. Double-click on the D2KDASK.BAS file. If the file is not on the list, make sure the list is displaying files from the correct directory. By default, the D2KDASK.BAS file is installed at C:\ADLink\D2K-DASK\INCLUDE.

## Design the interface

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

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

## Set the interface controls


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

## Write the event code

The event code defines the required action to be performed when an event occurs. To write the event code, double-click on the control or form to view the code module, then add the event code. You can also call the functions declared in the D2KDASK.BAS file to perform data acquisition operations.

## Run the application

Do one of the following to run the application:

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

## Distribute the application

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

You must include the D2K-DASK's DLL and driver files when the application is distributed. Refer to D2K-DASK User's Manual for details.

## 1.2 Application Building Fundamentals in Linux

The following sections provide fundamental instructions when using D2K-DASK to build application in Linux. To create a data acquisition application using D2K-DASK/X and GNU C/C++, follow these steps:

### Edit the source files

Include the header file **d2kdask.h** in the C/C++ source files that call D2K-DASK/X functions. The d2kdask.h has all the function declarations and constants that you can use to develop your data acquisition application. Add this statement in your code to include the header file.

```
#include "d2kdask.h"
```

### Build your application

Using the appropriate C/C++ compiler (gcc or cc) to compile the program. You should add **-lpci\_dask2k** option to link **libpci\_dask2k.so** library. For multi-threaded applications, the **-lpthread** string is required. For example:

```
gcc -o testai testai.c -lpci_dask2k
```

## 1.3 Function Classes

This chapter describes the classes of functions that the D2K-DASK supports.

All D2K-DASK functions are grouped into different classes:

- ▶ General Configuration Function Group
- ▶ Analog Input Function Group
  - ▷ Analog Input Configuration Functions
  - ▷ One-Shot Analog Input Functions
  - ▷ Continuous Analog Input Functions
  - ▷ Asynchronous Analog Input Monitoring Functions
- ▶ Analog Output Function Group
  - ▷ Analog output Configuration Functions
  - ▷ One-Shot Analog Output Functions
  - ▷ Continuous Analog Output Functions
  - ▷ Asynchronous Analog Output Monitoring Functions
- ▶ Digital Input Function Group
  - ▷ Digital Input Configuration Functions
  - ▷ One-Shot Digital Input Functions
- ▶ Digital Output Function Group
  - ▷ Digital Output Configuration Functions
  - ▷ One-Shot Digital Output Functions
- ▶ Timer/Counter Function Group
  - ▷ General-Purpose Timer/Counter Functions
- ▶ DIO Function Group
  - ▷ Digital Input/Output Configuration Function
- ▶ SSI Function Group
- ▶ Calibration Function Group

## 2 Function Reference

This chapter contains the detailed description of D2K-DASK functions, including the D2K-DASK data types and function reference. The functions are arranged alphabetically in section 2.2.

### 2.1 Data Types

The D2K-DASK library uses these data types in D2KDASK.H. It is recommended that you use these data types in your application programs. The table shows the data type names, ranges, and corresponding data types in C/C++, Visual Basic, and Delphi for your reference.

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

---

**NOTE** We didn't define these data types in D2KDASK.BAS and D2KDASK.PAS. Here they are just listed for reference

---

## 2.2 Function Reference

### D2K\_AI\_AsyncCheck

#### Description

Checks the current status of the asynchronous analog input operation.

#### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

#### Syntax

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

```
I16 D2K_AI_AsyncCheck (U16 CardNumber, BOOLEAN  
    *Stopped, U32 *AccessCnt)
```

#### Visual Basic

```
D2K_AI_AsyncCheck (ByVal CardNumber As Integer,  
    Stopped As Byte, AccessCnt As Long) As  
    Integer
```

#### Parameters

*CardNumber* ID of the card performing asynchronous operation.

*Stopped* Tells whether the asynchronous analog input operation is complete. If Stopped = TRUE, the analog input operation has stopped after the number of A/D conversions indicated in the call that initiated the asynchronous analog input operation is complete completed or an error has occurred. If Stopped = FALSE, the operation is not yet complete. Constants TRUE and FALSE are defined in D2KDASK.H.

*AccessCnt* When the pre-trigger or middle trigger mode of AI acquisition is not used, *AccessCnt* returns the number of A/D data that has been transferred at the time when *D2K\_AI\_AsyncCheck()* is called.

When pre-trigger or middle trigger mode of AI is enabled, and double-buffered mode is enabled, *AccessCnt* returns to the next position after the position of the last A/D data is stored in the circular buffer at the time calling *D2K\_AI\_AsyncCheck()*.

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport

## D2K\_AI\_AsyncClear

### Description

Stops the asynchronous analog input operation.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
U16 D2K_AI_AsyncClear (U16 CardNumber, U32
    *StartPos, U32 *AccessCnt)
```

### Visual Basic

```
D2K_AI_AsyncClear (ByVal CardNumber As Integer,
    StartPos As Long, AccessCnt As Long) As
    Integer
```

### Parameters

*CardNumber* ID of the card performing asynchronous operation.

*StartPos* When trigger acquisition mode is not used, *StartPos* is zero. When pre-trigger or middle trigger mode of AI is used, *StartPos* returns the position of the first AD data in the data buffer at the time calling *D2K\_AI\_AsyncClear()*.

*AccessCnt* When pre-/middle trigger acquisition mode is not used, *AccessCnt* returns the number of A/D data that has been transferred at the time calling *D2K\_AI\_AsyncClear()*.

When double-buffered mode is enabled, *AccessCnt* returns the next position after the position of the last A/D data is stored in the circular buffer.

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```



## D2K\_AI\_AsyncDblBufferHalfReady

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```

I16 D2K_AI_AsyncDblBufferHalfReady (U16
    CardNumber, BOOLEAN *HalfReady, BOOLEAN
    *StopFlag)
    
```

### Visual Basic

```

D2K_AI_AsyncDblBufferHalfReady(ByVal CardNumber
    As Integer, HalfReady As Byte, StopFlag As
    Byte) As Integer
    
```

### Parameters

- |                   |   |
|-------------------|---|
| <i>CardNumber</i> | ID of the card performing asynchronous double-buffered operation.   |
| <i>HalfReady</i>  | Tells whether the next half buffer of data is available. Constants TRUE and FALSE are defined in D2KDASK.H.   |
| <i>StopFlag</i>   | Tells Whether the asynchronous analog input operation is complete. 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 D2KDASK.H. |

### Return Code

```

NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
    
```

## D2K\_AI\_AsyncDbIBufferHandled

### Description

Notifies D2K-DASK that the ready buffer has been handled in user application.

For D2k-Dask, the data are transferred through DMA to the user's buffer directly. Therefore, while half buffer of data is ready (using D2K\_AI\_AsyncDbIBufferHalfReady to check the ready status), the data in the ready buffer can be handled directly and don't need to be copied to another transfer buffer. This mechanism eliminates the time taken for memory copy and another memory space for data transfer; however, D2K-DASK doesn't know if the data in the ready buffer have been handled (in user application). If the data is handled, the user application needs an interface to notify D2K-DASK this information. The new function D2K\_AI\_AsyncDbIBufferHandled is used to for this purpose.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_AsyncDbIBufferHandled (U16 CardNumber)
```

### Visual Basic

```
D2K_AI_AsyncDbIBufferHandled (ByVal CardNumber As Integer) As Integer
```

### Parameter

*CardNumber* ID of the card where double-buffered mode is to be set.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport
```

## D2K\_AI\_AsyncDblBufferMode

### Description

Enables or disables double-buffered data acquisition mode.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_AsyncDblBufferMode (U16 CardNumber,  
                                BOOLEAN Enable)
```

### Visual Basic

```
D2K_AI_AsyncDblBufferMode (ByVal CardNumber As  
                             Integer, ByVal Enable As Byte) As Integer
```

### Parameters

*CardNumber* ID of the card where double-buffered mode is to be set.

*Enable* Tells whether the double-buffered mode is enabled or not. Constants TRUE and FALSE are defined in D2KDASK.H.

TRUE Double-buffered mode is enabled.

FALSE Double-buffered mode is disabled.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport
```

## D2K\_AI\_AsyncDblBufferOverrun

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_AsyncDblBufferOverrun (U16 CardNumber,  
                                   U16 op, U16 *overrunFlag)
```

### Visual Basic

```
D2K_AI_AsyncDblBufferOverrun (ByVal CardNumber As  
                               Integer, ByVal op As Integer, overrunFlag As  
                               Integer) As Integer
```

### Parameters

*CardNumber*     ID of the card where double-buffered mode is to be set.

*op*               Checks/clears the overrun status/flag.

- |   |                           |
|---|---------------------------|
| 0 | Check the overrun status. |
| 1 | Clear the overrun flag.   |

*overrunFlag*     Returned overrun status.

- |   |                      |
|---|----------------------|
| 0 | No overrun occurred. |
| 1 | Overrun occurred.    |

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport
```

## D2K\_AI\_AsyncDblBufferToFile

### Description

Call this function to log the data of the circular buffer into a disk file when the continuous AI function is:

```
D2K_AI_ContReadChannelToFile  
D2K_AI_ContReadMultiChannelsToFile  
D2K_AI_ContScanChannelsToFile  
D2K_AI_ContMuxScanToFile
```

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_AsyncDblBufferToFile (U16 CardNumber)
```

### Visual Basic

```
D2K_AI_AsyncDblBufferToFile (ByVal CardNumber As  
Integer) As Integer
```

### Parameter

*CardNumber* ID of the card where double-buffered mode is to be set.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport
```

## D2K\_AI\_AsyncReTrigNextReady D2K\_AI\_AsyncReTrigNextReadyEx

### Description

Checks whether the data associated to the next trigger signal is ready during an asynchronous re-triggered analog input operation. This function is based on the following conditions:

- ▶ D2K\_AI\_Config has to be called prior to D2K\_AI\_ContBufferSetup.
- ▶ Asynchronous mode should use ASYNCH\_OP for continuous AI operation.

The D2K\_AI\_AsyncReTrigNextReadyEx function is used for the devices (e.g. DAQ-2020/2022) which Trigger counters width are 32-bit long.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_AsyncReTrigNextReady (U16 CardNumber,  
    BOOLEAN *trgReady, BOOLEAN *StopFlag, U16  
    *RdyTrigCnt)  
I16 D2K_AI_AsyncReTrigNextReadyEx (U16  
    CardNumber, BOOLEAN *trgReady, BOOLEAN  
    *StopFlag, U32 *RdyTrigCnt)
```

### Visual Basic

```
D2K_AI_AsyncReTrigNextReady (ByVal CardNumber As  
    Integer, trgReady As Byte, StopFlag As Byte,  
    RdyTrigCnt As Integer) As Integer  
D2K_AI_AsyncReTrigNextReadyEx (ByVal CardNumber  
    As Integer, trgReady As Byte, StopFlag As  
    Byte, RdyTrigCnt As Long) As Integer
```

### Parameters

**CardNumber** ID of the card performing asynchronous re-trigger operation.

<i>trgReady</i>	Tells whether the data associated with the next trigger signal is available. Constants TRUE and FALSE are defined in D2KDASK.H.
<i>StopFlag</i>	Tells whether the asynchronous analog input operation is complete. If StopFlag = TRUE, the analog input operation has stopped. If StopFlag = FALSE, the operation is not yet completed. Constants TRUE and FALSE are defined in D2KDASK.H.
<i>RdyTrigCnt</i>	This argument returns the count of trigger signal that occurred if re-triggert count is definite. If the re-triggert count is infinite, this argument returns the index of the buffer that stored the data after the most recent trigger signal trigger is generated.

## Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

## D2K\_AI\_CH\_Config

### Description

Informs the D2K-DASK library of the selected AI range for the specified channel of the card with CardNumber ID. After calling the D2K\_Register\_Card function, all analog input channels are configured as AD\_B\_10\_V (for DAQ-2010, DAQ-2005, DAQ-2006, DAQ-2016, DAQ-2020/2022, DAQ-2501, and DAQ-2502) or AD\_B\_10\_V with AI\_RSE (for DAQ-2204, DAQ-2205, DAQ-2206, DAQ-2213, DAQ-2214, and DAQ-2208) by default. If you want to use the device with the default settings, there is no need to call this function again to configure the channel(s). You must call this function to program the device based on your settings before calling function to perform analog input operation.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_CH_Config (U16 CardNumber, U16  
Channel, U16 AdRange_RefGnd)
```

### Visual Basic

```
D2K_AI_CH_Config (ByVal CardNumber As Integer,  
ByVal Channel As Integer, ByVal  
AdRange_RefGnd As Integer) As Integer
```



Parameters

<i>CardNumber</i>	ID of the card performing the operation.																								
<i>Channel</i>	A/D channel range for channel setting. Valid values: <table><tr><td>DAQ-2010</td><td>0 to 3 or All_Channels (-1)</td></tr><tr><td>DAQ-2005</td><td>0 to 3 or All_Channels (-1)</td></tr><tr><td>DAQ-2006</td><td>0 to 3 or All_Channels (-1)</td></tr><tr><td>DAQ-2016</td><td>0 to 3 or All_Channels (-1)</td></tr><tr><td>DAQ-2204</td><td>0 to 63 or All_Channels (-1)</td></tr><tr><td>DAQ-2205</td><td>0 to 63 or All_Channels (-1)</td></tr><tr><td>DAQ-2206</td><td>0 to 63 or All_Channels (-1)</td></tr><tr><td>DAQ-2208</td><td>0 to 95 or All_Channels (-1)</td></tr><tr><td>DAQ-2213</td><td>0 to 15 or All_Channels (-1)</td></tr><tr><td>DAQ-2214</td><td>0 to 15 or All_Channels (-1)</td></tr><tr><td>DAQ-2501</td><td>0 to 7 or All_Channels (-1)</td></tr><tr><td>DAQ-2502</td><td>0 to 3 or All_Channels (-1)</td></tr></table>	DAQ-2010	0 to 3 or All_Channels (-1)	DAQ-2005	0 to 3 or All_Channels (-1)	DAQ-2006	0 to 3 or All_Channels (-1)	DAQ-2016	0 to 3 or All_Channels (-1)	DAQ-2204	0 to 63 or All_Channels (-1)	DAQ-2205	0 to 63 or All_Channels (-1)	DAQ-2206	0 to 63 or All_Channels (-1)	DAQ-2208	0 to 95 or All_Channels (-1)	DAQ-2213	0 to 15 or All_Channels (-1)	DAQ-2214	0 to 15 or All_Channels (-1)	DAQ-2501	0 to 7 or All_Channels (-1)	DAQ-2502	0 to 3 or All_Channels (-1)
DAQ-2010	0 to 3 or All_Channels (-1)																								
DAQ-2005	0 to 3 or All_Channels (-1)																								
DAQ-2006	0 to 3 or All_Channels (-1)																								
DAQ-2016	0 to 3 or All_Channels (-1)																								
DAQ-2204	0 to 63 or All_Channels (-1)																								
DAQ-2205	0 to 63 or All_Channels (-1)																								
DAQ-2206	0 to 63 or All_Channels (-1)																								
DAQ-2208	0 to 95 or All_Channels (-1)																								
DAQ-2213	0 to 15 or All_Channels (-1)																								
DAQ-2214	0 to 15 or All_Channels (-1)																								
DAQ-2501	0 to 7 or All_Channels (-1)																								
DAQ-2502	0 to 3 or All_Channels (-1)																								

*AdRange\_RefGnd*

Analog input channel settings. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H. There are two groups of constants:

*A/D Range Selection:*

Some constants are defined to represent various A/D input ranges in D2KDASK.H. Refer to **Appendix B: AI Range Codes** for valid range values. Default is AD\_B\_10\_V.

*A/D Reference Ground Selection* (available only for DAQ-2204/2205/2206/2208/2213/2214)

DAQ-2010, DAQ-2005, DAQ-2006, DAQ-2016, DAQ- 2501, DAQ-2502	0
DAQ-2204, DAQ-2205, DAQ-2206, DAQ-2208, DAQ- 2213, DAQ-2214	AI_RSE: Referenced single ended mode (64-CH common to ground system onboard) is the default.  AI_DIFF: Differential mode  AI_NRSE: Non-referenced single ended mode (64-CH common to AISENSE pin)

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

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport

## D2K\_AI\_Config D2K\_AI\_ConfigEx

### Description

Informs the D2K-DASK library of the trigger source, trigger mode, and trigger properties for the DAQ-2000 device with the CardNumber ID. After calling the Register\_Card function, the device is configured with the following default values:

<b>A/D conversion source</b>	DAQ2K_AI_ADCONVSRC_Int
<b>A/D trigger mode</b>	DAQ2K_AI_TRGMOD_POST
<b>A/D trigger source</b>	DAQ2K_AI_TRGSRC_SOFT
<b>Auto reset buffer</b>	Enabled (AutoResetBuf : TRUE)

If you want to use the device with the default settings, you do not need to call this function again to reset the configuration. However, you must call this function before calling a function to perform continuous analog input operation. The D2K\_AI\_ConfigEx function is used for the devices (e.g. DAQ-2020/2022) which counters width are 32-bit long.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_Config (U16 CardNumber, U16
    ConfigCtrl, U32 TrigCtrl, U32 MidOrDlyScans,
    U16 MCnt, U16 ReTrgCnt, BOOLEAN
    AutoResetBuf)
I16 D2K_AI_ConfigEx (U16 CardNumber, U16
    ConfigCtrl, U32 TrigCtrl, U32 MidOrDlyScans,
    U32 MCnt, U32 ReTrgCnt, BOOLEAN
    AutoResetBuf)
```

### Visual Basic

```
D2K_AI_Config (ByVal CardNumber As Integer, ByVal
    ConfigCtrl As Integer, ByVal TrigCtrl As
    Long, ByVal MidOrDlyScans As Long, ByVal
```

```
MCnt As Integer, ByVal ReTrgCnt As Integer,
ByVal AutoResetBuf As Byte) As Integer
D2K_AI_ConfigEx (ByVal CardNumber As Integer,
ByVal ConfigCtrl As Integer, ByVal TrigCtrl
As Long, ByVal MidOrDlyScans As Long, ByVal
MCnt As Long, ByVal ReTrgCnt As Long, ByVal
AutoResetBuf As Byte) As Integer
```

## Parameters

- CardNumber** ID of the card performing the operation.
- ConfigCtrl** A/D configuration control setting. This argument is an integer expression formed from one or more of the manifest constants defined in D2kDASK.H. There are two groups of constants:

### A/D Conversion Source Selection

DAQ2K_AI_ADCONVSRC_Int	Internal timer (default)
DAQ2K_AI_ADCONVSRC_AFI0	From AFI0 pin
DAQ2K_AI_ADCONVSRC_SSI	From SSI source
DAQ2K_AI_ADCONVSRC_AFI1	From AFI1 pin (DAQ-2501/ 2502 only)
DAQ2K_AI_ADCONVSRC_AFI2	From AFI2 pin (DAQ-2020/ 2022 only)
DAQ2K_AI_ADCONVSRC_AFI3	From AFI3 pin (DAQ-2020/ 2022 only)
DAQ2K_AI_ADCONVSRC_AFI4	From AFI4 pin (DAQ-2020/ 2022 only)
DAQ2K_AI_ADCONVSRC_AFI5	From AFI5 pin (DAQ-2020/ 2022 only)
DAQ2K_AI_ADCONVSRC_AFI6	From AFI6 pin (DAQ-2020/ 2022 only)
DAQ2K_AI_ADCONVSRC_AFI7	From AFI7 pin (DAQ-2020/ 2022 only)

### A/D Delay Counter Source Selection (available only for DAQ-2501/2502)

DAQ2K_AI_DT SRC_Int	Internal timer (default)
DAQ2K_AI_DT SRC_AFI1	From AFI1 pin
DAQ2K_AI_DT SRC_GPTC0	From GPTC0_OUT
DAQ2K_AI_DT SRC_GPTC1	From GPTC1_OUT

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

*TrigCtrl*

A/D trigger control setting. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H. There are six groups of constants:

### Trigger Source Selection

DAQ2K_AI_TRGSRC_SOFT	Software (default)
DAQ2K_AI_TRGSRC_ANA	From analog trigger pin
DAQ2K_AI_TRGSRC_ExtD	From external digital trigger pin
DAQ2K_AI_TRSRC_SSI	From SSI source
DAQ2K_AI_TRGSRC_AFI0	From AFI0 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI1	From AFI1 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI2	From AFI2 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI3	From AFI3 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI4	From AFI4 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI5	From AFI5 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI6	From AFI6 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI7	From AFI7 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_PXIStar	PXI Star Trigger as the trigger source (available only for DAQ-2020/2022)
DAQ2K_AI_TRGSRC_SMB	From SMB Trigger IO Connector (available only for DAQ-2020/2022)

### Trigger Mode Selection

DAQ2K_AI_TRGMOD_POST	Post Trigger Mode (default)
DAQ2K_AI_TRGMOD_DELAY	Delay Trigger Mode
DAQ2K_AI_TRGMOD_PRE	Pre-Trigger Mode
DAQ2K_AI_TRGMOD_MIDL	Middle-Trigger Mode

**Delay Source Selection** *(available only for Delay Trigger Mode)*

DAQ2K_AI_Dly1InSamples	Delay in samples (not available for DAQ-202/2022)
DAQ2K_AI_Dly1InTimebase	Delay in time base (default)

**Re-Trigger Mode Enable**  
*(available only for Delay and Post Trigger Mode)*

DAQ2K_AI_ReTrigEn	Re-trigger in an acquisition is enabled
-------------------	---

**MCounter Enable**  
*(available only for pre- and middle trigger mode and is only valid for pre-trigger and middle trigger mode)*

DAQ2K_AI_MCounterEn	Mcounter is enabled and then the trigger signal is ignore before M terminal count is reached.
---------------------	---

**External Digital Trigger Polarity**

DAQ2K_AI_TrgPositive	Trigger positive edge active (default)
DAQ2K_AI_TrgNegative	Trigger negative edge active

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

**MidOrDlyScans** Valid only for middle trigger and delay trigger modes. For middle trigger, this argument indicates the number of data that will be accessed after a specific trigger event. The valid value range of MidOrDlyScans for the middle trigger for cards except DAQ-2020/2022, is 0 to 16777215. For the DAQ-2020/2022, the range of valid values is from 0 to 2147483647.

For Delay trigger, this argument indicates the number of data or timer ticks that will be ignored after a specific trigger event. For the D2K\_AI\_Config, the valid value range of DlyScans for delay trigger is 0 to 65535. For the DAQ-2020/2022 and using the func-

tion D2K\_AI\_ConfigEx, the valid value range of DlyScans for delay trigger is 0 to 4294967295.

*MCnt* The counter value of MCounter. For D2K\_AI\_Config, the valid value range of MCnt is 0 through 65535. This argument is only valid for pre-trigger and Middle trigger mode. For the DAQ-2020/2022 and using the function D2K\_AI\_ConfigEx, the valid value range of MCnt is 0 to 4294967295.

*ReTrgCnt* The accepted trigger times in an acquisition. For D2K\_AI\_Config, the valid range of ReTrgCnt is 0 to 65535. If the value of ReTrgCnt is 0, the AI operation is triggered infinitely. For DAQ-2020/2022 and using the function D2K\_AI\_ConfigEx, the valid value range of ReTrgCnt is 0 to 4294967295. This argument is only valid for delay trigger and post trigger modes.

**NOTE** To enable infinite re-trigger mode of continuous AI, call D2K\_AI\_Config with DAQ2K\_AI\_ReTrigEn and zero value of ReTrgCnt.

*AutoResetBuf*

- FALSE

The AI buffers set by the D2K\_AI\_ContBufferSetup function are retained. You must call the D2K\_AI\_ContBufferReset function to reset the buffer.
- TRUE

The AI buffers set by the D2K\_AI\_ContBufferSetup function are reset automatically by driver when the AI operation is completed.

**NOTE** If Mcounter is enabled, the ReadScans parameter of continuous AI functions D2K\_AI\_ContXXXX has to be equal to MidOrDlyScans+MCnt.

**Return Code**

- NoError
- ErrorInvalidCardNumber
- ErrorCardNotRegistered
- ErrorFuncNotSupport

## D2K\_AI\_ContBufferReset

### Description

Resets all buffers set by function D2K\_AI\_ContBufferSetup for continuous analog input. Call this function when the data buffers will not be used.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_ContBufferReset (U16 CardNumber)
```

### Visual Basic

```
D2K_AI_ContBufferReset (ByVal CardNumber As  
Integer) As Integer
```

### Parameter

*CardNumber* ID of the card performing the operation.

### Return Code

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



## D2K\_AI\_ContBufferSetup

### Description

Sets up the buffer for continuous analog input. The function has to be called repeatedly to setup all the data buffers (two buffers maximum). For double buffer mode and infinite re-trigger mode of continuous AI, calling D2K\_AI\_ContBufferSetup twice sets up the ring buffer to store the data.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
U16 D2K_AI_ContBufferSetup (U16 CardNumber, void
                             *Buffer, U32 ReadCount, U16 *BufferId)
```

### Visual Basic

```
D2K_AI_ContBufferSetup (ByVal CardNumber As
                        Integer, Buffer As Any, ByVal ReadCount As
                        Long, BufferId As Integer) As Integer
```

### Parameters

- |                   |   |
|-------------------|---|
| <i>CardNumber</i> | ID of the card performing the operation.                  |
| <i>Buffer</i>     | Starting address of the memory to contain the input data. |
| <i>ReadCount</i>  | Buffer size (in samples); value must be even.             |
| <i>BufferId</i>   | Returns the index of the currently set up buffer.         |

### Return Code

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

## D2K\_AI\_ContMuxScan

### Description

Initializes the Channel-Gain Queue to point to the start of the scan sequence as specified by D2K\_AI\_MuxScanSetup and starts a multiple-channel scanned data acquisition operation. This function is available only for multiplexed AD cards.

### Supported Cards

2204, 2205, 2206, 2208, 2213, 2214

### Syntax

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

```
I16 D2K_AI_ContMuxScan (U16 CardNumber, U16
    BufId, U32 ReadScans, U32 ScanIntrv, U32
    SampIntrv, U16 SyncMode)
```

### Visual Basic

```
D2K_AI_ContMuxScan (ByVal CardNumber As Integer,
    ByVal BufId As Integer, ByVal ReadScans As
    Long, ByVal ScanIntrv As Long, ByVal
    SampIntrv As Long, ByVal SyncMode As
    Integer) As Integer
```

### Parameters

- |                   |   |
|-------------------|---|
| <i>CardNumber</i> | ID of the card performing the operation.  |
| <i>BufId</i>      | ID (returned from function D2K_AI_ContBufferSetup) of the buffer containing the acquired data. The size of the buffer with BufId must have a length equal to the value of parameter ScanCount * (number of channels per scan). If double-buffered mode is enabled, the starting buffer id must be 0. You may ignore this argument. Refer to <b>Appendix C: AI Data Format</b> for the data format in the buffer with BufId. |
| <i>ReadScans</i>  | If double-buffered mode is disabled, this is the total number of scans to be performed. For double-buffered acquisition, ReadScans is the size (in samples) allocated for each channel in the circular buffer. Range of valid values is 2 to 16777215. The value must be a multiple of two.   |

---

**NOTE** If Mcounter is enabled, the ReadScans has to be equal to MidOrDlyScans+MCnt.

---

*ScanIntrv* The length of the scan interval or the counter value between the initiation of each scan sequence. The scan rate is  $\text{TimeBase}/\text{ScanIntrv}$ . The value of TimeBase depends on the card type.

If the timer base is external, the range of valid value is 8 to 16777215. If the timer base is internal, the range of valid values are:

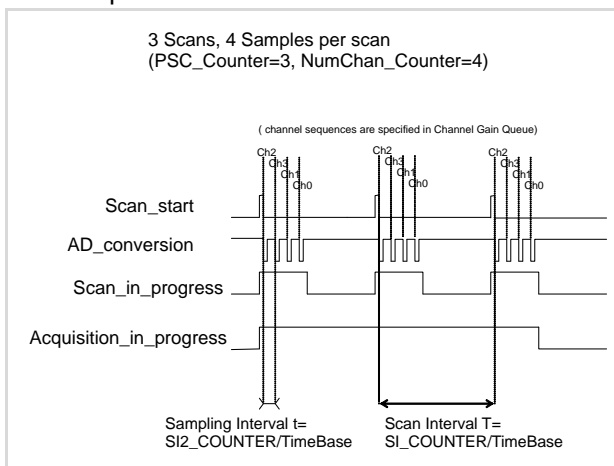
DAQ-2204	14 to 16777215
DAQ-2205	80 to 16777215
DAQ-2206	160 to 16777215
DAQ-2208	14 to 16777215
DAQ-2213	160 to 16777215
DAQ-2214	160 to 16777215

*SamplIntrv* The length of the sample interval or the counter value between each A/D conversion within a scan sequence. The A/D conversion rate is  $\text{TimeBase}/\text{SamplIntrv}$ . The value of TimeBase depends on the card type.

If the timer base is external, the range of valid values is 8 to 65535. If the timer base is internal, the range of valid values are:

DAQ-2204	14 to 65535
DAQ-2205	80 to 65535
DAQ-2206	160 to 65535
DAQ-2208	14 to 65535
DAQ-2213	160 to 65535
DAQ-2214	160 to 65535

For example:



**Figure 2-1: Scan Timing (D2K\_AI\_ContMuxScan)**

**NOTE** The SampIntrv parameter is ignored on simultaneous AD card such as DAQ-2010.

**SyncMode** Tells whether this operation is performed synchronously or asynchronously. If pre-/middle trigger mode is enabled by calling D2K\_AI\_Config(), this operation should be performed asynchronously. Valid values:

SYNCH\_OP Synchronous A/D conversion. The function does not return until the A/D operation is completed.

ASYNCH\_OP Asynchronous A/D conversion

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel  
ErrorInvalidSampleRate  
ErrorInvalidAdRange  
ErrorTransferCountTooLarge  
ErrorContIoNotAllowed

## D2K\_AI\_ContMuxScanToFile

### Description

Initializes the Channel-Gain Queue to point to the start of the scan sequence as specified by D2K\_AI\_MuxScanSetup, starts a multiple-channel scanned data acquisition operation, and saves the acquired data in a disk file. The data is written in binary format with the lower byte first (little endian). Refer to **Appendix D: Data File Format** for the data file structure and **Appendix C: AI Data Format** for supported data file formats. This function takes advantage of the DAQ-2000 channel-gain that can be set separately for each channel to perform multi-channel/gain analog input and is only available for multiplexed AD card.

### Supported Cards

2204, 2205, 2206, 2208, 2213, 2214

### Syntax

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

```
I16 D2K_AI_ContMuxScanToFile (U16 CardNumber, U16
    BufId, U8 *FileName, U32 ReadScans, U32
    ScanIntrv, U32 SampIntrv, U16 SyncMode)
```

### Visual Basic

```
D2K_AI_ContMuxScanToFile (ByVal CardNumber As
    Integer, ByVal BufId As Integer, ByVal
    FileName As String, ByVal ReadScans As Long,
    ByVal ScanIntrv As Long, ByVal SampIntrv As
    Long, ByVal SyncMode As Integer) As Integer
```

### Parameters

- |                   |   |
|-------------------|---|
| <i>CardNumber</i> | ID of the card performing the operation.  |
| <i>BufId</i>      | ID (returned from function D2K_AI_ContBufferSetup) of the buffer containing the acquired data. The size of the buffer with BufId must have a length equal to the value of parameter ScanCount * (number of channels per scan). If double-buffered mode is enabled, the starting buffer id must be 0. You may ignore this argument. Refer to <b>Appendix C: AI Data Format</b> for the data format in the buffer with BufId. |

<i>FileName</i>	Name of data file that stored the acquired data.
<i>ReadScans</i>	If double-buffered mode is disabled, this is the total number of scans to be performed. For double-buffered acquisition, ReadScans is the size (in samples) allocated for each channel in the circular buffer. Range of valid values is 2 to 16777215. The value must be a multiple of two.

---

**NOTE** If Mcounter is enabled, the ReadScans has to be equal to MidOrDlyScans+MCnt.

---

*ScanIntrv* The length of the scan interval or the counter value between the initiation of each scan sequence. The scan rate is TimeBase/ScanIntrv. The value of TimeBase depends on the card type.

If the timer base is external, the range of valid value is from 8 to 16777215. If the timer base is internal, the range of valid values are:

DAQ-2204	14 to 16777215
DAQ-2205	80 to 16777215
DAQ-2206	160 to 16777215
DAQ-2208	14 to 16777215
DAQ-2213	160 to 16777215
DAQ-2214	160 to 16777215

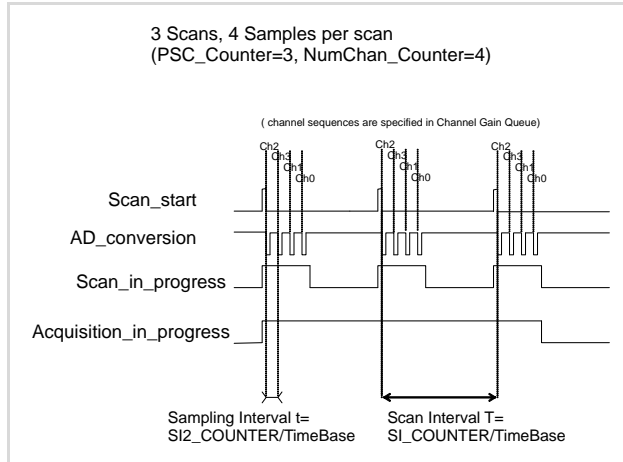
*SamplIntrv* The length of the sample interval or the counter value between each A/D conversion within a scan sequence. The A/D conversion rate is TimeBase/SamplIntrv. The value of TimeBase depends on the card type.

If the timer base is external, the range of valid values is 8 to 65535. If the timer base is internal, the range of valid values are:

DAQ-2204	14 to 65535
DAQ-2205	80 to 65535
DAQ-2206	160 to 65535
DAQ-2208	14 to 65535

DAQ-2213 160 to 65535  
DAQ-2214 160 to 65535

For example:



**Figure 2-2: Scan Timing (D2K\_AI\_ContMuxScanToFile)**

**NOTE** The `Samplntrv` parameter is ignored on simultaneous AD card such as DAQ-2010.

**SyncMode** Tells whether this operation is performed synchronously or asynchronously. If pre-/middle trigger mode is enabled by calling `D2K_AI_Config()`, this operation should be performed asynchronously. Valid values:

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

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel  
ErrorInvalidSampleRate  
ErrorInvalidAdRange  
ErrorTransferCountTooLarge  
ErrorContIoNotAllowed  
ErrorOpenFile



## D2K\_AI\_ContReadChannel

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_ContReadChannel (U16 CardNumber, U16
    Channel, U16 BufId, U32 ReadScans, U32
    ScanIntrv, U32 SampIntrv, U16 SyncMode)
```

### Visual Basic

```
D2K_AI_ContReadChannel (ByVal CardNumber As
    Integer, ByVal Channel As Integer, ByVal
    BufId As Integer, ByVal ReadScans As Long,
    ByVal ScanIntrv As Long, ByVal SampIntrv As
    Long, ByVal SyncMode As Integer) As Integer
```

### Parameter

<i>CardNumber</i>	ID of the card performing the operation.												
<i>Channel</i>	Analog input channel number. <table><tr><td>DAQ-2010, DAQ-2005, DAQ-2006,</td><td>0 to 3</td></tr><tr><td>DAQ-2016, DAQ-2502</td><td></td></tr><tr><td>DAQ-2204, DAQ-2205, DAQ-2206</td><td>0 to 63</td></tr><tr><td>DAQ-2208</td><td>0 to 95</td></tr><tr><td>DAQ-2213, DAQ-2214, 2020, 2022</td><td>0 to 15</td></tr><tr><td>DAQ-2501</td><td>0 to 7</td></tr></table>	DAQ-2010, DAQ-2005, DAQ-2006,	0 to 3	DAQ-2016, DAQ-2502		DAQ-2204, DAQ-2205, DAQ-2206	0 to 63	DAQ-2208	0 to 95	DAQ-2213, DAQ-2214, 2020, 2022	0 to 15	DAQ-2501	0 to 7
DAQ-2010, DAQ-2005, DAQ-2006,	0 to 3												
DAQ-2016, DAQ-2502													
DAQ-2204, DAQ-2205, DAQ-2206	0 to 63												
DAQ-2208	0 to 95												
DAQ-2213, DAQ-2214, 2020, 2022	0 to 15												
DAQ-2501	0 to 7												
<i>BufId</i>	ID (returned from function D2K_AI_ContBufferSetup) of the buffer containing the acquired data. The size of the buffer with BufId must have a length equal to the value of parameter ScanCount * (number of channels per scan). If double-buffered mode is enabled, the starting buffer id must be 0. You may ignore this												

argument. Refer to **Appendix C: AI Data Format** for the data format in the buffer with BufId.

*ReadScans* If double-buffered mode is disabled, this is the total number of scans to be performed. For double-buffered acquisition, ReadScans is the size (in samples) allocated for each channel in the circular buffer. For cards except DAQ-2020/2022, the range of valid values are from 2 to 16777215. For DAQ-2020/2022, the range of valid values is from 2 to 2147483647. The value must be a multiple of two.

---

**NOTE** If Mcounter is enabled, the ReadScans has to be equal to MidOrDlyScans+MCnt.

---

*ScanIntrv* The length of the scan interval or the counter value between the initiation of each scan sequence. The scan rate is TimeBase/ScanIntrv. The value of TimeBase depends on the card type.

If the timer base is external, the range of valid value is from 1 to 4294967295 for DAQ-2020/2022 or 8 to 16777215 for the others. If the timer base is internal, the range of valid values are:

DAQ-2010	20 to 16777215
DAQ-2005	80 to 16777215
DAQ-2006	160 to 16777215
DAQ-2016	50 to 16777215
DAQ-2020/22	320 to 83886080
DAQ-2204	14 to 16777215
DAQ-2205	80 to 16777215
DAQ-2206	160 to 16777215
DAQ-2208	14 to 16777215
DAQ-2213	160 to 16777215
DAQ-2214	160 to 16777215
DAQ-2501	100 to 16777215
DAQ-2502	100 to 16777215

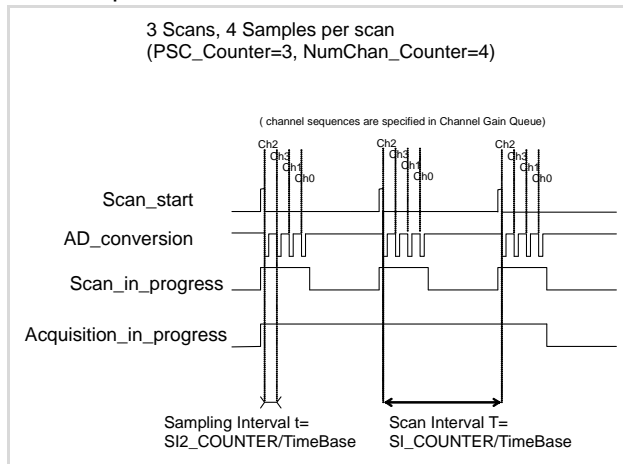
*SamplIntrv* The length of the sample interval or the counter value between each A/D conversion within a scan

sequence. The A/D conversion rate is  $\text{TimeBase} / \text{SampleIntrv}$ . The value of TimeBase depends on the card type.

If the timer base is external, the range of valid values is 8 to 65535. If the timer base is internal, the range of valid values are:

DAQ-2010	Invalid
DAQ-2005	Invalid
DAQ-2006	Invalid
DAQ-2016	Invalid
DAQ-2020/22	Invalid
DAQ-2204	14 to 65535
DAQ-2205	80 to 65535
DAQ-2206	160 to 65535
DAQ-2208	14 to 65535
DAQ-2213	160 to 65535
DAQ-2214	160 to 65535
DAQ-2501	100 to 16777215
DAQ-2502	100 to 16777215

For example:



**Figure 2-3: Scan Timing (D2K\_AI\_ContReadChannel)**

---

**NOTE**      The `SamplIntrv` parameter is ignored on simultaneous AD card such as DAQ-2010.

---

*SyncMode*      Tells whether this operation is performed synchronously or asynchronously. If pre-/middle trigger mode is enabled by calling `D2K_AI_Config()`, this operation should be performed asynchronously. Valid values:

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

`ASYNCH_OP`    Asynchronous A/D conversion

## Return Code

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

## D2K\_AI\_ContReadChannelToFile

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_ContReadChannelToFile (U16 CardNumber,
    U16 Channel, U16 BufId, U8 *FileName, U32
    ReadScans, U32 ScanIntrv, U32 SampIntrv, U16
    SyncMode)
```

### Visual Basic

```
D2K_AI_ContReadChannelToFile (ByVal CardNumber As
    Integer, ByVal Channel As Integer, ByVal
    BufId As Integer, ByVal FileName As String,
    ByVal ReadScans As Long, ByVal ScanIntrv As
    Long, ByVal SampIntrv As Long, ByVal
    SyncMode As Integer) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*Channel* Analog input channel number.

DAQ-2010, DAQ-2005, DAQ-2006, DAQ-2016, DAQ-2502	0 to 3
DAQ-2204, DAQ-2205, DAQ-2206	0 to 63
DAQ-2208	0 to 95
DAQ-2213, DAQ-2214, DAQ-2020/ 2022	0 to 15
DAQ-2501	0 to 7

<i>BufId</i>	ID (returned from function D2K_AI_ContBufferSetup) of the buffer containing the acquired data. The size of the buffer with BufId must have a length equal to the value of parameter ScanCount * (number of channels per scan). If double-buffered mode is enabled, the starting buffer id must be 0. You may ignore this argument. Refer to <b>Appendix C: AI Data Format</b> for the data format in the buffer with BufId.
<i>FileName</i>	Name of data file that stored the acquired data.
<i>ReadScans</i>	If double-buffered mode is disabled, this is the total number of scans to be performed. For double-buffered acquisition, ReadScans is the size (in samples) allocated for each channel in the circular buffer. For cards except DAQ-2020/2022, the range of valid values are form 2 to 16777215. For DAQ-2020/2022, the range of valid values is from 2 to 2147483647. The value must be a multiple of two.

---

<b>NOTE</b>	If Mcounter is enabled, the ReadScans has to be equal to MidOrDlyScans+MCnt.
-------------	--

---

<i>ScanIntrv</i>	<p>The length of the scan interval or the counter value between the initiation of each scan sequence. The scan rate is TimeBase/ScanIntrv. The value of TimeBase depends on the card type.</p> <p>If the timer base is external, the range of valid value is from 1 to 4294967295 for DAQ-2020/2022 or 8 to 16777215 for the others. If the timer base is internal, the range of valid values are:</p> <table><tr><td>DAQ-2010</td><td>20 to 16777215</td></tr><tr><td>DAQ-2005</td><td>80 to 16777215</td></tr><tr><td>DAQ-2006</td><td>160 to 16777215</td></tr><tr><td>DAQ-2016</td><td>50 to 16777215</td></tr><tr><td>DAQ-2020/22</td><td>320 to 83886080</td></tr><tr><td>DAQ-2204</td><td>14 to 16777215</td></tr><tr><td>DAQ-2205</td><td>80 to 16777215</td></tr><tr><td>DAQ-2206</td><td>160 to 16777215</td></tr></table>	DAQ-2010	20 to 16777215	DAQ-2005	80 to 16777215	DAQ-2006	160 to 16777215	DAQ-2016	50 to 16777215	DAQ-2020/22	320 to 83886080	DAQ-2204	14 to 16777215	DAQ-2205	80 to 16777215	DAQ-2206	160 to 16777215
DAQ-2010	20 to 16777215																
DAQ-2005	80 to 16777215																
DAQ-2006	160 to 16777215																
DAQ-2016	50 to 16777215																
DAQ-2020/22	320 to 83886080																
DAQ-2204	14 to 16777215																
DAQ-2205	80 to 16777215																
DAQ-2206	160 to 16777215																

DAQ-2208	14 to 16777215
DAQ-2213	160 to 16777215
DAQ-2214	160 to 16777215
DAQ-2501	100 to 16777215
DAQ-2502	100 to 16777215

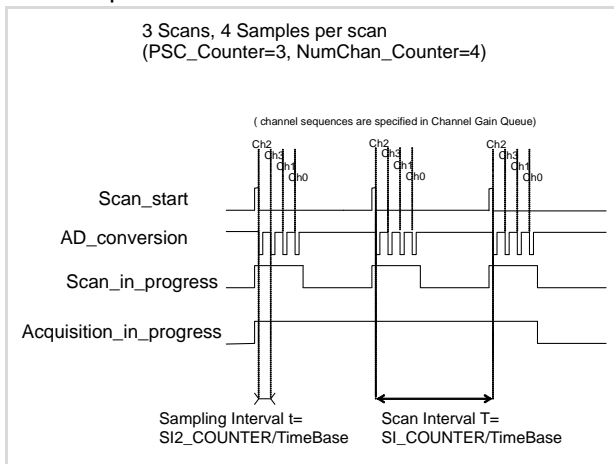
### *SamplIntrv*

The length of the sample interval or the counter value between each A/D conversion within a scan sequence. The A/D conversion rate is  $\text{TimeBase}/\text{SamplIntrv}$ . The value of TimeBase depends on the card type.

If the timer base is external, the range of valid values is 8 to 65535. If the timer base is internal, the range of valid values are:

DAQ-2010	Invalid
DAQ-2005	Invalid
DAQ-2006	Invalid
DAQ-2016	Invalid
DAQ-2020/22	Invalid
DAQ-2204	14 to 65535
DAQ-2205	80 to 65535
DAQ-2206	160 to 65535
DAQ-2208	14 to 65535
DAQ-2213	160 to 65535
DAQ-2214	160 to 65535
DAQ-2501	100 to 16777215
DAQ-2502	100 to 16777215

For example:



**Figure 2-4: Scan Timing (D2K\_AI\_ContReadChannelToFile)**

**NOTE** The SampleInterval parameter is ignored on simultaneous AD card such as DAQ-2010.

**SyncMode** Tells whether this operation is performed synchronously or asynchronously. If pre-/middle trigger mode is enabled by calling D2K\_AI\_Config(), this operation should be performed asynchronously. Valid values:

SYNCH\_OP Synchronous A/D conversion. The function does not return until the A/D operation is completed.

ASYNCH\_OP Asynchronous A/D conversion

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel  
ErrorInvalidAdRange  
ErrorTransferCountTooLarge  
ErrorContIoNotAllowed  
ErrorInvalidSampleRate



ErrorOpenFile

## D2K\_AI\_ContReadMultiChannels

### Description

Performs continuous A/D conversions on the specified analog input channels at a rate closest to the specified rate. This function takes advantage of the DAQ-2000 channel-gain that can be set separately for each channel to perform multi-channel/gain analog input.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_ContReadMultiChannels (U16 CardNumber,  
    U16 NumChans, U16 *Chans, U16 BufId, U32  
    ReadScans, U32 ScanIntrv, U32 SampIntrv, U16  
    SyncMode)
```

### Visual Basic

```
D2K_AI_ContReadMultiChannels (ByVal CardNumber As  
    Integer, ByVal NumChans As Integer, Chans As  
    Integer, ByVal BufId As Integer, ByVal  
    ReadScans As Long, ByVal ScanIntrv As Long,  
    ByVal SampIntrv As Long, ByVal SyncMode As  
    Integer) As Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.														
<i>NumChans</i>	Number of analog input channels in the array Chans. Valid values: <table><tr><td>DAQ-2010</td><td>1 to 4</td></tr><tr><td>DAQ-2005</td><td>1 to 4</td></tr><tr><td>DAQ-2006</td><td>1 to 4</td></tr><tr><td>DAQ-2016</td><td>1 to 4</td></tr><tr><td>DAQ-2020/22</td><td>1 to 16</td></tr><tr><td>DAQ-2204</td><td>1 to 512</td></tr><tr><td>DAQ-2205</td><td>1 to 512</td></tr></table>	DAQ-2010	1 to 4	DAQ-2005	1 to 4	DAQ-2006	1 to 4	DAQ-2016	1 to 4	DAQ-2020/22	1 to 16	DAQ-2204	1 to 512	DAQ-2205	1 to 512
DAQ-2010	1 to 4														
DAQ-2005	1 to 4														
DAQ-2006	1 to 4														
DAQ-2016	1 to 4														
DAQ-2020/22	1 to 16														
DAQ-2204	1 to 512														
DAQ-2205	1 to 512														

DAQ-2206	1 to 512
DAQ-2208	1 to 1024
DAQ-2213	1 to 512
DAQ-2214	1 to 512
DAQ-2501	1 to 8
DAQ-2502	1 to 4

*Chans*                      Array of analog input channel numbers.

DAQ-2010	Numbers in Chans must be within 0 and 3.
DAQ-2005	Numbers in Chans must be within 0 and 3.
DAQ-2006	Numbers in Chans must be within 0 and 3.
DAQ-2016	Numbers in Chans must be within 0 and 3.
DAQ-2020/22	Numbers in Chans must be within 0 and 15.
DAQ-2204	Numbers in Chans must be within 0 and 63.
DAQ-2205	Numbers in Chans must be within 0 and 63.
DAQ-2206	Numbers in Chans must be within 0 and 63.
DAQ-2208	Numbers in Chans must be within 0 and 95.
DAQ-2213	Numbers in Chans must be within 0 and 15.
DAQ-2214	Numbers in Chans must be within 0 and 15.
DAQ-2501	Numbers in Chans must be within 0 and 7.
DAQ-2502	Numbers in Chans must be within 0 and 3.

*BuflD*                      ID (returned from function D2K\_AI\_ContBufferSetup) of the buffer containing the acquired data. The size of the buffer with BuflD must have a length equal to the value of parameter ScanCount \* (number of channels per scan). If double-buffered mode is enabled, the starting buffer id must be 0. You may ignore this argument. Refer to **Appendix C: AI Data Format** for the data format in the buffer with BuflD.

*ReadScans*                If double-buffered mode is disabled, this is the total number of scans to be performed. For double-buffered acquisition, ReadScans is the size (in samples) allocated for each channel in the circular buffer. All cards except for DAQ-2020/2022, the range of valid values are form 2 to 16777215. For DAQ-2020/2022, the range of valid values is from 2 to 2147483647. The value must be a multiple of two.

---

**NOTE** If Mcounter is enabled, the ReadScans has to be equal to MidOrDlyScans+MCnt.

---

*ScanIntrv* The length of the scan interval or the counter value between the initiation of each scan sequence. The scan rate is  $\text{TimeBase}/\text{ScanIntrv}$ . The value of TimeBase depends on the card type.

If the timer base is external, the range of valid value is from 1 to 4294967295 for DAQ-2020/2022 or 8 to 16777215 for the others. If the timer base is internal, the range of valid values are:

DAQ-2010	20 to 16777215
DAQ-2005	80 to 16777215
DAQ-2006	160 to 16777215
DAQ-2016	50 to 16777215
DAQ-2020/22	320 to 83886080
DAQ-2204	14 to 16777215
DAQ-2205	80 to 16777215
DAQ-2206	160 to 16777215
DAQ-2208	14 to 16777215
DAQ-2213	160 to 16777215
DAQ-2214	160 to 16777215
DAQ-2501	100 to 16777215
DAQ-2502	100 to 16777215

*SamplIntrv* The length of the sample interval or the counter value between each A/D conversion within a scan sequence. The A/D conversion rate is  $\text{TimeBase}/\text{SamplIntrv}$ . The value of TimeBase depends on the card type.

If the timer base is external, the range of valid values is 8 to 65535. If the timer base is internal, the range of valid values are:

DAQ-2010	Invalid
DAQ-2005	Invalid
DAQ-2006	Invalid
DAQ-2016	Invalid

DAQ-2020/22	Invalid
DAQ-2204	14 to 65535
DAQ-2205	80 to 65535
DAQ-2206	160 to 65535
DAQ-2208	14 to 65535
DAQ-2213	160 to 65535
DAQ-2214	160 to 65535
DAQ-2501	100 to 16777215
DAQ-2502	100 to 16777215

For example:

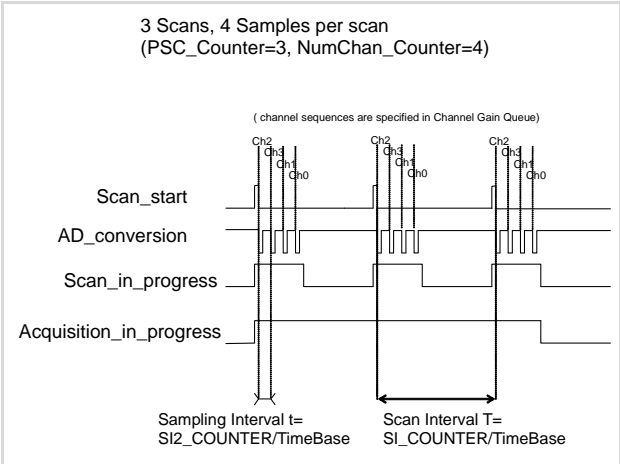


Figure 2-5: Scan Timing (D2K\_AI\_ContReadMultiChannels)

**NOTE** The `SamplIntrv` parameter is ignored on simultaneous AD card such as DAQ-2010.

*SyncMode* Tells whether this operation is performed synchronously or asynchronously. If pre-/middle trigger mode is enabled by calling `D2K_AI_Config()`, this operation should be performed asynchronously. Valid values:

- |                        |  |
|------------------------|--|
| <code>SYNCH_OP</code>  | Synchronous A/D conversion. The function does not return until the A/D operation is completed. |
| <code>ASYNCH_OP</code> | Asynchronous A/D conversion  |

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel  
ErrorInvalidSampleRate  
ErrorInvalidAdRange  
ErrorTransferCountTooLarge  
ErrorContIoNotAllowed

D2K\_AI\_ContReadMultiChannelsToFile

Description

Performs continuous A/D conversions on the specified analog input channels at a rate closest the specified rate, then saves the acquired data in a disk file. The data is written in binary format with the lower byte first (little endian). Refer to **Appendix D: Data File Format** for the data file structure and **Appendix C: AI Data Format** for supported data file formats. This function takes advantage of the DAQ-2000 channel-gain that can be set separately for each channel to perform multi-channel/gain analog input.

Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

Syntax

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

```
I16 D2K_AI_ContReadMultiChannelsToFile (U16
    CardNumber, U16 NumChans, U16 *Chans, U16
    BufId, U8 *FileName, U32 ReadScans, U32
    ScanIntrv, U32 SampIntrv, U16 SyncMode)
```

Visual Basic

```
D2K_AI_ContScanChannelsToFile (ByVal CardNumber
    As Integer, ByVal NumChans As Integer, Chans
    As Integer, ByVal BufId As Integer, ByVal
    FileName As String, ByVal ReadScans As Long,
    ByVal ScanIntrv As Long, ByVal SampIntrv As
    Long, ByVal SyncMode As Integer) As Integer
```

Parameters

<i>CardNumber</i>	ID of the card performing the operation.
<i>NumChans</i>	Number of analog input channels in the array Chans. Valid values:
	DAQ-2010      1 to 4
	DAQ-2005      1 to 4
	DAQ-2006      1 to 4
	DAQ-2016      1 to 4
	DAQ-2020/22   1 to 16

DAQ-2204	1 to 512
DAQ-2205	1 to 512
DAQ-2206	1 to 512
DAQ-2208	1 to 1024
DAQ-2213	1 to 512
DAQ-2214	1 to 512
DAQ-2501	1 to 8
DAQ-2502	1 to 4

*Chans*                      Array of analog input channel numbers.

DAQ-2010	Numbers in Chans must be within 0 and 3.
DAQ-2005	Numbers in Chans must be within 0 and 3.
DAQ-2006	Numbers in Chans must be within 0 and 3.
DAQ-2016	Numbers in Chans must be within 0 and 3.
DAQ-2020/22	Numbers in Chans must be within 0 and 15.
DAQ-2204	Numbers in Chans must be within 0 and 63.
DAQ-2205	Numbers in Chans must be within 0 and 63.
DAQ-2206	Numbers in Chans must be within 0 and 63.
DAQ-2208	Numbers in Chans must be within 0 and 95.
DAQ-2213	Numbers in Chans must be within 0 and 15.
DAQ-2214	Numbers in Chans must be within 0 and 15.
DAQ-2501	Numbers in Chans must be within 0 and 7.
DAQ-2502	Numbers in Chans must be within 0 and 3.

*BuflD*                      ID (returned from function D2K\_AI\_ContBufferSetup) of the buffer containing the acquired data. The size of the buffer with BuflD must have a length equal to the value of parameter ScanCount \* (number of channels per scan). If double-buffered mode is enabled, the starting buffer id must be 0. You may ignore this argument. Refer to **Appendix C: AI Data Format** for the data format in the buffer with BuflD.

*FileName*                      Name of data file that stored the acquired data.

*ReadScans*                      If double-buffered mode is disabled, this is the total number of scans to be performed. For double-buffered acquisition, ReadScans is the size (in samples) allocated for each channel in the circular buffer. All cards except for DAQ-2020/2022, the



range of valid values are form 2 to 16777215. For DAQ-2020/2022, the range of valid values is from 2 to 2147483647. The value must be a multiple of two.

**NOTE**

If Mcounter is enabled, the ReadScans has to be equal to MidOrDlyScans+MCnt.

*ScanIntrv*

The length of the scan interval or the counter value between the initiation of each scan sequence. The scan rate is TimeBase/ScanIntrv. The value of TimeBase depends on the card type.

If the timer base is external, the range of valid value is from 1 to 4294967295 for DAQ-2020/2022 or 8 to 16777215 for the others. If the timer base is internal, the range of valid values are:

DAQ-2010	20 to 16777215
DAQ-2005	80 to 16777215
DAQ-2006	160 to 16777215
DAQ-2016	50 to 16777215
DAQ-2020/22	320 to 83886080
DAQ-2204	14 to 16777215
DAQ-2205	80 to 16777215
DAQ-2206	160 to 16777215
DAQ-2208	14 to 16777215
DAQ-2213	160 to 16777215
DAQ-2214	160 to 16777215
DAQ-2501	100 to 16777215
DAQ-2502	100 to 16777215

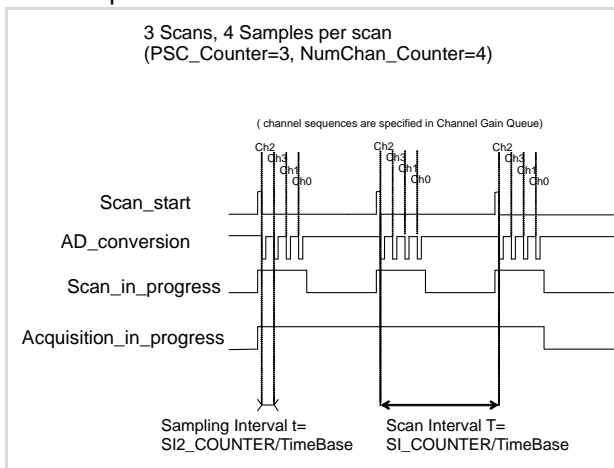
*SamplIntrv*

The length of the sample interval or the counter value between each A/D conversion within a scan sequence. The A/D conversion rate is TimeBase/SamplIntrv. The value of TimeBase depends on the card type.

If the timer base is external, the range of valid values is 8 to 65535. If the timer base is internal, the range of valid values are:

DAQ-2010	Invalid
DAQ-2005	Invalid
DAQ-2006	Invalid
DAQ-2016	Invalid
DAQ-2020/22	Invalid
DAQ-2204	14 to 65535
DAQ-2205	80 to 65535
DAQ-2206	160 to 65535
DAQ-2208	14 to 65535
DAQ-2213	160 to 65535
DAQ-2214	160 to 65535
DAQ-2501	100 to 16777215
DAQ-2502	100 to 16777215

For example:



**Figure 2-6: Scan Timing (D2K\_AI\_ContReadMultiChannelsToFile)**

**NOTE** The SampleNtry parameter is ignored on simultaneous AD card such as DAQ-2010.

*SyncMode* Tells whether this operation is performed synchronously or asynchronously. If pre-/middle trigger mode is enabled by calling D2K\_AI\_Config(), this operation should be performed asynchronously. Valid values:

- SYNCH\_OP Synchronous A/D conversion. The function does not return until the A/D operation is completed.
- ASYNCH\_OP Asynchronous A/D conversion

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel  
ErrorInvalidSampleRate  
ErrorInvalidAdRange  
ErrorTransferCountTooLarge  
ErrorContIoNotAllowed  
ErrorOpenFile

## D2K\_AI\_ContScanChannels

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_ContScanChannels (U16 CardNumber, U16
    Channel, U16 BufId, U32 ReadScans, U32
    ScanIntrv, U32 SampIntrv, U16 SyncMode)
```

### Visual Basic

```
D2K_AI_ContScanChannels (ByVal wCardNumber As
    Integer, ByVal Channel As Integer, ByVal
    BufId As Integer, ByVal ReadScans As Long,
    ByVal ScanIntrv As Long, ByVal SampIntrv As
    Long, ByVal SyncMode As Integer) As Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.						
<i>Channel</i>	Largest channel number of specified continuous analog input channel. The channel order for acquiring data is listed below: <table><tr><td>DAQ-2010</td><td>Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.</td></tr><tr><td>DAQ-2005</td><td>Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.</td></tr><tr><td>DAQ-2006</td><td>Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.</td></tr></table>	DAQ-2010	Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.	DAQ-2005	Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.	DAQ-2006	Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.
DAQ-2010	Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.						
DAQ-2005	Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.						
DAQ-2006	Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.						

DAQ-2016	Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.
DAQ-2020 / 2022	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.
DAQ-2204	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.
DAQ-2205	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.
DAQ-2206	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.
DAQ-2208	Channel must be within 0 and 95. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.
DAQ-2213	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.
DAQ-2214	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.
DAQ-2501	Channel must be within 0 and 7. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.
DAQ-2502	Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.

### *BuflId*

ID (returned from function D2K\_AI\_ContBufferSetup) of the buffer containing the acquired data. The size of the buffer with BuflId must have a length equal to the value of parameter ScanCount \* (number of channels per scan). If double-buffered mode is enabled, the starting buffer id must be 0. You may ignore this argument. Refer to **Appendix C: AI Data Format** for the data format in the buffer with BuflId.

### *ReadScans*

If double-buffered mode is disabled, this is the total number of scans to be performed. For double-buffered acquisition, ReadScans is the size (in

samples) allocated for each channel in the circular buffer. All boards except for DAQ-2020/2022, the range of valid values are from 2 to 16777215. For DAQ-2020/2022, the range of valid values is from 2 to 2147483647. The value must be a multiple of two.

---

<b>NOTE</b>	If Mcounter is enabled, the ReadScans has to be equal to MidOrDlyScans+MCnt.
-------------	--

---

<i>ScanIntrv</i>	The length of the scan interval or the counter value between the initiation of each scan sequence. The scan rate is $\text{TimeBase}/\text{ScanIntrv}$ . The value of TimeBase depends on the card type.
------------------	--

If the timer base is external, the range of valid value is from 1 to 4294967295 for DAQ-2020/2022 or 8 to 16777215 for the others. If the timer base is internal, the range of valid values are:

DAQ-2010	20 to 16777215
DAQ-2005	80 to 16777215
DAQ-2006	160 to 16777215
DAQ-2016	50 to 16777215
DAQ-2020/22	320 to 83886080
DAQ-2204	14 to 16777215
DAQ-2205	80 to 16777215
DAQ-2206	160 to 16777215
DAQ-2208	14 to 16777215
DAQ-2213	160 to 16777215
DAQ-2214	160 to 16777215
DAQ-2501	100 to 16777215
DAQ-2502	100 to 16777215

<i>SamplIntrv</i>	The length of the sample interval or the counter value between each A/D conversion within a scan sequence. The A/D conversion rate is $\text{TimeBase}/\text{SamplIntrv}$ . The value of TimeBase depends on the card type.
-------------------	---

If the timer base is external, the range of valid values is 8 to 65535. If the timer base is internal, the range of valid values are:

DAQ-2010	Invalid
DAQ-2005	Invalid
DAQ-2006	Invalid
DAQ-2016	Invalid
DAQ-2020/22	Invalid
DAQ-2204	14 to 65535
DAQ-2205	80 to 65535
DAQ-2206	160 to 65535
DAQ-2208	14 to 65535
DAQ-2213	160 to 65535
DAQ-2214	160 to 65535
DAQ-2501	100 to 16777215
DAQ-2502	100 to 16777215

For example:

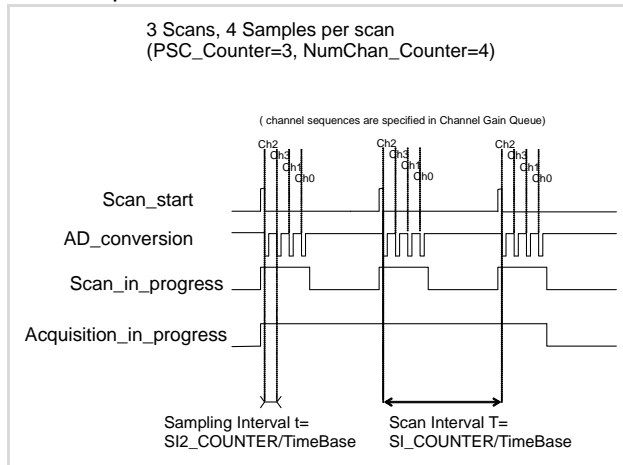


Figure 2-7: Scan Timing (D2K\_AI\_ContScanChannels)

**NOTE** The SampleInterval parameter is ignored on simultaneous AD card such as DAQ-2010.

*SyncMode* Tells whether this operation is performed synchronously or asynchronously. If pre-/middle trigger mode is enabled by calling `D2K_AI_Config()`, this operation should be performed asynchronously. Valid values:

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



## Return Code

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

## D2K\_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 simultaneous or auto-scan functionality to perform multi-channel analog input.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_ContScanChannelsToFile (U16
    CardNumber, U16 Channel, U16 BufId, U8
    *FileName, U32 ReadScans, U32 ScanIntrv, U32
    SampIntrv, U16 SyncMode);
```

### Visual Basic

```
D2K_AI_ContScanChannelsToFile (ByVal CardNumber
    As Integer, ByVal Channel As Integer, ByVal
    BufId As Integer, ByVal FileName As String,
    ByVal ReadScans As Long, ByVal ScanIntrv As
    Long, ByVal SampIntrv As Long, ByVal
    SyncMode As Integer) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*Channel* Largest channel number of specified continuous analog input channel. The channel order for acquiring data is listed below:

DAQ-2010	Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.
----------	--

DAQ-2005	Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.
DAQ-2006	Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.
DAQ-2016	Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.
DAQ-2020 / 22	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.
DAQ-2204	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.
DAQ-2205	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.
DAQ-2206	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.
DAQ-2208	Channel must be within 0 and 95. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.
DAQ-2213	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.
DAQ-2214	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.
DAQ-2501	Channel must be within 0 and 7. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.
DAQ-2502	Channel must be within 0 and 3. The continuous scan sequence is ascending and the first one must be zero. For example, 0, 1, 2, 3.

### *BufId*

ID (returned from function D2K\_AI\_ContBufferSetup) of the buffer containing the acquired data. The size of the buffer with BufId must have a length equal to the value of parameter ScanCount \* (number of channels per scan). If double-buffered mode is enabled, the starting buffer id must be 0. You may ignore this

argument. Refer to **Appendix C: AI Data Format** for the data format in the buffer with BufId.

*ReadScans* If double-buffered mode is disabled, this is the total number of scans to be performed. For double-buffered acquisition, ReadScans is the size (in samples) allocated for each channel in the circular buffer. All boards except for DAQ-2020/2022, the range of valid values are form 2 to 16777215. For DAQ-2020/2022, the range of valid values is from 2 to 2147483647. The value must be a multiple of two.

---

**NOTE** If Mcounter is enabled, the ReadScans has to be equal to MidOrDlyScans+MCnt.

---

*ScanIntrv* The length of the scan interval or the counter value between the initiation of each scan sequence. The scan rate is TimeBase/ScanIntrv. The value of TimeBase depends on the card type.

If the timer base is external, the range of valid value is from 1 to 4294967295 for DAQ-2020/2022 or 8 to 16777215 for the others. If the timer base is internal, the range of valid values are:

DAQ-2010	20 to 16777215
DAQ-2005	80 to 16777215
DAQ-2006	160 to 16777215
DAQ-2016	50 to 16777215
DAQ-2020/22	320 to 83886080
DAQ-2204	14 to 16777215
DAQ-2205	80 to 16777215
DAQ-2206	160 to 16777215
DAQ-2208	14 to 16777215
DAQ-2213	160 to 16777215
DAQ-2214	160 to 16777215
DAQ-2501	100 to 16777215
DAQ-2502	100 to 16777215

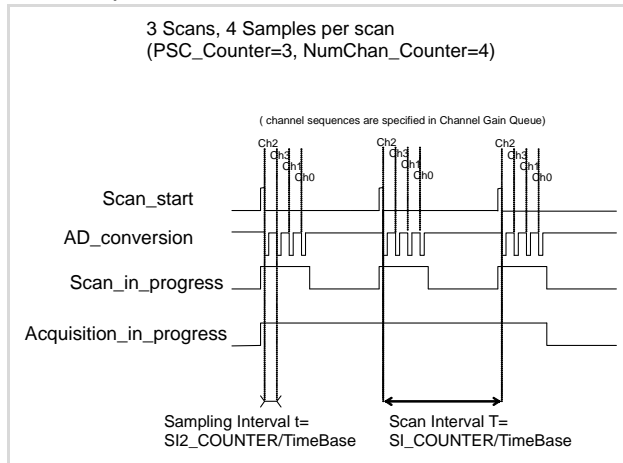
*SamplIntrv* The length of the sample interval or the counter value between each A/D conversion within a scan

sequence. The A/D conversion rate is  $\text{TimeBase} / \text{SampleIntrv}$ . The value of TimeBase depends on the card type.

If the timer base is external, the range of valid values is 8 to 65535. If the timer base is internal, the range of valid values are:

DAQ-2010	Invalid
DAQ-2005	Invalid
DAQ-2006	Invalid
DAQ-2016	Invalid
DAQ-2020/22	Invalid
DAQ-2204	14 to 65535
DAQ-2205	80 to 65535
DAQ-2206	160 to 65535
DAQ-2208	14 to 65535
DAQ-2213	160 to 65535
DAQ-2214	160 to 65535
DAQ-2501	100 to 16777215
DAQ-2502	100 to 16777215

For example:



**Figure 2-8: Scan Timing (D2K\_AI\_ContScanChannelsToFile)**

---

**NOTE**      The `SampleInterval` parameter is ignored on simultaneous AD card such as DAQ-2010.

---

*SyncMode*      Tells whether this operation is performed synchronously or asynchronously. If pre-/middle trigger mode is enabled by calling `D2K_AI_Config()`, this operation should be performed asynchronously. Valid values:

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

`ASYNCH_OP`    Asynchronous A/D conversion

## Return Code

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

## D2K\_AI\_ContStatus

### Description

While performing continuous A/D conversions, this function is called to get the A/D status. Refer to the documentation that came with your device for the AI status the device may meet.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```

I16 D2K_AI_ContStatus (U16 CardNumber, U16
    *Status)

```

### Visual Basic

```

D2K_AI_ContStatus (ByVal CardNumber As Integer,
    Status As Integer) As Integer

```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.
<i>Status</i>	The returned continuous AI status. The description of this parameter for various card types include: <div><div><div>DAQ-2010, DAQ-2005, DAQ-2006, DAQ-2016, DAQ-2204, DAQ-2205, DAQ-2206, DAQ-2208, DAQ-2213, DAQ-2214</div><div>bit 0: 1 indicates that the A/D FIFO is empty. bit 1: 1 indicates that the A/D FIFO is half-full. bit 2: 1 indicates that the A/D FIFO is full. bit 3: Not used bit 4: 1 indicates the A/D overspeed status. bit 5: 1 indicates the A/D overrun status. bit 6: 1 indicates the A/D trigger status. bit 7: 1 indicates the scan counter terminal count status. bit 8 ~ 15: Not used</div></div></div>

DAQ-2020, DAQ-2022	bit 1: 1 indicates that the DMA FIFO is empty.
	bit 2: 1 indicates that the A/D FIFO is almost empty.
	bit 3: 1 indicates that the A/D FIFO is almost full.
	bit 4: 1 indicates that the A/D FIFO is full.
	bit 5-7: not used.
	bit 8: 1 indicates data acquisition is in-progress.
	bit 9: 1 indicates data acquisition is done.
	bit 10 ~ 15: Not used.
DAQ-2501, DAQ-2502	bit 0: 1 indicates that the A/D FIFO is empty.
	bit 1: 1 indicates that the A/D FIFO is half-full.
	bit 2: 1 indicates that the A/D FIFO is full.
	bit 3: 1 indicates that the A/D FIFO is almost empty.
	bit 4: 1 indicates that the A/D FIFO is almost full.
	bit 5: 1 indicates the A/D trigger status.
	bit 6: 1 indicates the A/D programming status.
	bit 7 ~ 15: Not used

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered



## D2K\_AI\_ContVScale

### Description

Converts the values of an array of acquired binary data from a continuous A/D conversion call to actual input voltages. The acquired binary data in the reading array may include the channel information (refer to D2K\_AI\_ContReadChannel or D2K\_AI\_ContScanChannels for the detailed data format). However, the calculated voltage values in the voltage array returned does not include the channel message.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_ContVScale (U16 CardNumber, U16
    AdRange, void *readingArray, F64
    *voltageArray, I32 count)
```

### Visual Basic

```
D2K_AI_ContVScale (ByVal CardNumber As Integer,
    ByVal AdRange As Integer, readingArray As
    Integer, voltageArray As Double, ByVal count
    As Long) As Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.
<i>AdRange</i>	Analog input range and the setting of the continuous specified channel. Refer to <b>Appendix B: AI Range Codes</b> for the range of valid values.
<i>readingArray</i>	Acquired continuous analog input data array.
<i>voltageArray</i>	Computed voltages array returned.
<i>count</i>	Number of converted data.

## Return Code

NoError, ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidAdRange

## D2K\_AI\_DelayTrig\_Config

### D2K\_AI\_DelayTrig\_ConfigEx

#### Description

Informs the D2K-DASK library of the conversion clock source and trigger properties of the device performing a delay triggered data acquisition operation.

The D2K\_AI\_DelayTrig\_ConfigEx function is used for the devices (e.g. DAQ-2020/2022) which counters width are 32-bit long.

#### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

#### Syntax

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

```
I16 D2K_AI_DelayTrig_Config (U16 CardNumber, U16
    ClkSrc, U32 TrigSrcCtrl, U32 DlyScans, U16
    ReTrgEn, U16 ReTrgCnt, BOOLEAN AutoResetBuf)
I16 D2K_AI_DelayTrig_ConfigEx (U16 CardNumber,
    U16 ClkSrc, U32 TrigSrcCtrl, U32 DlyScans,
    U16 ReTrgEn, U32 ReTrgCnt, BOOLEAN
    AutoResetBuf)
```

#### Visual Basic

```
D2K_AI_DelayTrig_Config (ByVal CardNumber As
    Integer, ByVal ClkSrc As Integer, ByVal
    TrigSrcCtrl As Integer, ByVal DlyScans As
    Long, ByVal ReTrgEn As Integer, ByVal
    ReTrgCnt As Integer, ByVal AutoResetBuf As
    Byte) As Integer
D2K_AI_DelayTrig_ConfigEx (ByVal CardNumber As
    Integer, ByVal ClkSrc As Integer, ByVal
    TrigSrcCtrl As Long, ByVal DlyScans As Long,
    ByVal ReTrgEn As Integer, ByVal ReTrgCnt As
    Long, ByVal AutoResetBuf As Byte) As Integer
```

#### Parameters

*CardNumber* ID of the card performing the operation.

*ClkSrc* A/D clock source settings. This argument is an integer expression formed from one or more of the

manifest constants defined in D2KDASK.H. There are two groups of constants:

### A/D Conversion Source Selection

DAQ2K_AI_ADCONVSRC_Int	Internal timer (default)
DAQ2K_AI_ADCONVSRC_AFI0	From AFI0 pin
DAQ2K_AI_ADCONVSRC_SSI	From SSI source
DAQ2K_AI_ADCONVSRC_AFI1	From AFI1 pin (available only on DAQ-2501/2502)
DAQ2K_AI_ADCONVSRC_AFI2	From AFI2 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI3	From AFI3 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI4	From AFI4 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI5	From AFI5 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI6	From AFI6 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI7	From AFI7 pin (DAQ-2020/2022 only)

### A/D Delay Counter Source Selection

(available only for DAQ-2501/2502)

DAQ2K_AI_DTSRC_Int	Internal timer (default)
DAQ2K_AI_DTSRC_AFI1	From AFI1 pin
DAQ2K_AI_DTSRC_GPTC0	From GPTC0_OUT
DAQ2K_AI_DTSRC_GPTC1	From GPTC1_OUT

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

*TrigSrcCtrl*

A/D trigger source control setting. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H. There are three groups of constants:

### Trigger Source Selection

DAQ2K_AI_TRGSRC_SOFT	Software (Default)
DAQ2K_AI_TRGSRC_ANA	From analog trigger pin

DAQ2K_AI_TRGSRC_ExtD	From external digital trigger pin
DAQ2K_AI_TRSRC_SSI	From SSI source
DAQ2K_AI_TRGSRC_AFI0	From AFI0 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI1	From AFI1 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI2	From AFI2 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI3	From AFI3 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI4	From AFI4 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI5	From AFI5 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI6	From AFI6 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI7	From AFI7 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_PXIStar	PXI Star Trigger as the trigger source (available only for DAQ-2020/2022)
DAQ2K_AI_TRGSRC_SMB	From SMB Trigger IO Connector (available only for DAQ-2020/2022)

Delay Source Selection

DAQ2K_AI_Dly1InSamples	Delay in samples (not available for DAQ-2020/2022)
DAQ2K_AI_Dly1InTimebase	Delay in timebase (Default)

External Digital Trigger Polarity

DAQ2K_AI_TrgPositive	Trigger positive edge active (Default)
DAQ2K_AI_TrgNegative	Trigger negative edge active

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

DlyScans

The number of data or timer ticks to be ignored after a specific trigger event. For D2K\_AI\_DelayTrig\_Config, the valid range of value is 0 to 65535. For the DAQ-2020/2022, using the

function D2K\_AI\_DelayTrig\_ConfigEx, the valid value range of DlyScans for delay trigger is 0 to 4294967295.

### *ReTrgEn*

- |   |   |
|---|---|
| 0 | Re-trigger in an acquisition is disabled. (Default) |
| 1 | Re-trigger in an acquisition is enabled.            |

### *ReTrgCnt*

The accepted trigger times in an acquisition. If the value of ReTrgCnt is 0, the AI operation is triggered infinitely. For D2K\_AI\_DelayTrig\_Config, the valid range of value is 0 to 65535. For the DAQ-2020/2022, using the function D2K\_AI\_DelayTrig\_ConfigEx, the valid value range of ReTrgCnt is 0 to 4294967295.

---

### **NOTE**

To enable infinite re-trigger mode for continuous AI, call D2K\_AI\_DelayTrig\_Config, then assign 1 to ReTrgEn and 0 to ReTrgCnt.

---

### *AutoResetBuf*

- |       |   |
|-------|---|
| FALSE | The AI buffers set by the D2K_AI_ContBufferSetup function are retained and D2K_AI_ContBufferReset must be called to reset the buffer. |
| TRUE  | The AI buffers set by the D2K_AI_ContBufferSetup function are reset automatically by the driver when AI operation is finished.        |

### **Return Code**

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport

## D2K\_AI\_EventCallBack

### Description

Controls and notifies the user's application when a specified DAQ event occurs. The notification is performed through a user-specified callback function.

In Linux, the event message has to be manually removed by setting **mode** to 0. In Windows, the event message is removed automatically after calling `D2K_AI_Async_Clear`. The event message may also be manually removed by setting **mode** to 0.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

Microsoft C/C++ and Borland C++

```
I16 D2K_AI_EventCallBack (U16 CardNumber, I16  
mode, I16 EventType, U32 callbackAddr)
```

Linux C++

```
I16 D2K_AI_EventCallBack (U16 CardNumber, I16  
mode, I16 EventType, void  
(*callbackAddr)(int))
```

Visual Basic

```
D2K_AI_EventCallBack (ByVal CardNumber As  
Integer, ByVal mode As Integer, ByVal  
EventType As Integer, ByVal callbackAddr As  
Long) As Integer
```

## Parameters

<i>CardNumber</i>	ID of the card performing the operation.						
<i>mode</i>	Adds or removes the event message. Valid values: <table><tr><td>0</td><td>Remove</td></tr><tr><td>1</td><td>Add</td></tr></table>	0	Remove	1	Add		
0	Remove						
1	Add						
<i>EventType</i>	Event criteria. Valid values: <table><tr><td>DAQEnd</td><td>Notifies that the asynchronous analog input operation is completed.</td></tr><tr><td>DBEvent</td><td>Notifies that the next half buffer of data in the circular buffer is ready for transfer.</td></tr><tr><td>TrigEvent</td><td>Notifies that the data associated to the next trigger signal is available.</td></tr></table>	DAQEnd	Notifies that the asynchronous analog input operation is completed.	DBEvent	Notifies that the next half buffer of data in the circular buffer is ready for transfer.	TrigEvent	Notifies that the data associated to the next trigger signal is available.
DAQEnd	Notifies that the asynchronous analog input operation is completed.						
DBEvent	Notifies that the next half buffer of data in the circular buffer is ready for transfer.						
TrigEvent	Notifies that the data associated to the next trigger signal is available.						
<i>callbackAddr</i>	Address of the user callback function. D2K-DASK calls this function when the specified event occurs. If you want to remove the event message, set <i>callbackAddr</i> to 0.						

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport



## D2K\_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.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
U16 D2K_AI_InitialMemoryAllocated (U16  
CardNumber, U32 *MemSize)
```

Visual Basic

```
D2K_AI_InitialMemoryAllocated (ByVal CardNumber  
As Integer, MemSize As Long) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*MemSize* Available memory size in the device driver of the card for continuous AI. The unit is in KB (1024 bytes).

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered
```

## D2K\_AI\_MiddleTrig\_Config D2K\_AI\_MiddleTrig\_ConfigEx

### Description

Informs the D2K-DASK library of the conversion clock source and trigger properties of the device performing a middle triggered data acquisition operation.

The D2K\_AI\_MiddleTrig\_ConfigEx function is used for the devices (e.g. DAQ-2020/2022) which counters width are 32-bit long.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214

### Syntax

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

```
I16 D2K_AI_MiddleTrig_Config (U16 CardNumber,  
    U16 ClkSrc, U32 TrigSrcCtrl, U32  
    MiddleScans, U16 MCtrlEn, U16 MCnt, BOOLEAN  
    AutoResetBuf)  
  
I16 D2K_AI_MiddleTrig_ConfigEx (U16 CardNumber,  
    U16 ClkSrc, U32 TrigSrcCtrl, U32  
    MiddleScans, U16 MCtrlEn, U32 MCnt, BOOLEAN  
    AutoResetBuf)
```

### Visual Basic

```
D2K_AI_MiddleTrig_Config (ByVal CardNumber As  
    Integer, ByVal ClkSrc As Integer, ByVal  
    TrigSrcCtrl As Integer, ByVal MiddleScans As  
    Long, ByVal MCtrlEn As Integer, ByVal MCnt As  
    Integer, ByVal AutoResetBuf As Byte) As  
    Integer  
  
D2K_AI_MiddleTrig_Config (ByVal CardNumber As  
    Integer, ByVal ClkSrc As Integer, ByVal  
    TrigSrcCtrl As Long, ByVal MiddleScans As  
    Long, ByVal MCtrlEn As Integer, ByVal MCnt As  
    Long, ByVal AutoResetBuf As Byte) As Integer
```

### Parameter

*CardNumber* ID of the card performing the operation.

*ClkSrc* A/D clock source settings. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H.

**A/D Conversion Source Selection**

DAQ2K_AI_ADCONVSRC_Int	Internal timer (default)
DAQ2K_AI_ADCONVSRC_AFI0	From AFI0 pin
DAQ2K_AI_ADCONVSRC_SSI	From SSI source
DAQ2K_AI_ADCONVSRC_AFI2	From AFI2 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI3	From AFI3 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI4	From AFI4 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI5	From AFI5 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI6	From AFI6 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI7	From AFI7 pin (DAQ-2020/2022 only)

*TrigSrcCtrl* A/D trigger source control setting. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H. There are two groups of constants:

**Trigger Source Selection**

DAQ2K_AI_TRGSRC_SOFT	Software (Default)
DAQ2K_AI_TRGSRC_ANA	From analog trigger pin
DAQ2K_AI_TRGSRC_ExtD	From external digital trigger pin
DAQ2K_AI_TRSRC_SSI	From SSI source
DAQ2K_AI_TRGSRC_AFI0	From AFI0 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI1	From AFI1 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI2	From AFI2 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI3	From AFI3 pin (DAQ-2020/2022 only)

DAQ2K_AI_TRGSRC_AFI4	From AFI4 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI5	From AFI5 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI6	From AFI6 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI7	From AFI7 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_PXIStar	PXI Star Trigger as the trigger source (available only for DAQ-2020/2022)
DAQ2K_AI_TRGSRC_SMB	From SMB Trigger IO Connector (available only for DAQ-2020/2022)

**External Digital Trigger Polarity**

DAQ2K_AI_TrgPositive	Trigger positive edge active (Default)
DAQ2K_AI_TrgNegative	Trigger negative edge active

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

*MiddleScans* The number of data accessed after a specific trigger event. For cards except the DAQ-2020/2022, the range of valid value is 0 to 16777215. For DAQ-2020/2022, the range of valid values is from 0 to 2147483647.

*MCtrEn*

0	Mcounter is disabled. (Default)
1	Mcounter is enabled and the trigger signal is ignored before M terminal count is reached.

*MCnt* Counter value of Mcounter. For D2K\_AI\_MiddleTrig\_Config, the range of valid value is 0 to 65535. For DAQ-2020/2022, using the function D2K\_AI\_MiddleTrigConfigEx, the valid value range of MCnt is 0 to 4294967295.

### *AutoResetBuf*

FALSE	The AI buffers set by the D2K_AI_ContBufferSetup function are retained and D2K_AI_ContBufferReset must be called to reset the buffer.
TRUE	The AI buffers set by the D2K_AI_ContBufferSetup function are reset automatically by the driver when AI operation is finished.

---

**NOTE**      If Mcounter is enabled, the ReadScans parameter of continuous AI functions D2K\_AI\_ContXXXX has to be equal with MiddleScans+MCnt.

---

### **Return Code**

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

## D2K\_AI\_MuxScanSetup

### Description

Stores numChans, chans, and gain\_refGnd in the Channel-Gain Queue for a scanned data acquisition operation. This function uses the memory table during the following scanning operations:

```
D2K_AI_ReadMuxScan
D2K_AI_ContMuxScan
D2K_AI_ContMuxScanToFile
```

and automatically sequence through an arbitrary set of analog input channels to allow gains to automatically change during scanning. This function is available only for multiplexed AD cards. The channel-gain queue is modified after calls to any AI functions other than those mentioned above.

### Supported Cards

2204, 2205, 2206, 2208, 2213, 2214

### Syntax

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

```
I16 D2K_AI_MuxScanSetup (U16 CardNumber, U16
    NumChans, U16* Chans, U16* AdRange_RefGnds);
```

### Visual Basic

```
D2K_AI_MuxScanSetup (ByVal CardNumber As
    Integer, ByVal NumChans As Integer, Chans As
    Integer, AdRange_RefGnds As Integer) As
    Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.
<i>NumChans</i>	Number of analog input channels in the array Chans. The valid value:
	DAQ-2204      1 through 512
	DAQ-2205      1 through 512
	DAQ-2206      1 through 512
	DAQ-2208      1 through 1024
	DAQ-2213      1 through 512
	DAQ-2214      1 through 512

*Chans*                      Array of analog input channel numbers.

DAQ-2204	Chans must be within 0 and 63.
DAQ-2205	Chans must be within 0 and 63.
DAQ-2206	Chans must be within 0 and 63.
DAQ-2208	Chans must be within 0 and 95.
DAQ-2213	Chans must be within 0 and 15.
DAQ-2214	Chans must be within 0 and 15.

*AdRange\_RefGnds*

An integer array of length numChans that contains the analog input range and reference ground for every channel in array Chans. Refer to D2K\_AI\_CH\_Config section for AdRange\_RefGnd setting for each channel.

## Return Code

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

## D2K\_AI\_PostTrig\_Config D2K\_AI\_PostTrig\_ConfigEx

### Description

Informs the D2K-DASK library of the conversion clock source and trigger properties of the device performing a post triggered data acquisition operation.

The D2K\_AI\_PostTrig\_ConfigEx function is used for the devices (e.g. DAQ-2020/2022) which counters width are 32-bit long.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_PostTrig_Config (U16 CardNumber, U16
    ClkSrc, U32 TrigSrcCtrl, U16 ReTrgEn, U16
    ReTrgCnt, BOOLEAN AutoResetBuf)
I16 D2K_AI_PostTrig_ConfigEx (U16 CardNumber, U16
    ClkSrc, U32 TrigSrcCtrl, U16 ReTrgEn, U32
    ReTrgCnt, BOOLEAN AutoResetBuf)
```

### Visual Basic

```
D2K_AI_PostTrig_Config (ByVal CardNumber As
    Integer, ByVal ClkSrc As Integer, ByVal
    TrigSrcCtrl As Integer, ByVal ReTrgEn As
    Integer, ByVal ReTrgCnt As Integer, ByVal
    AutoResetBuf As Byte) As Integer
D2K_AI_PostTrig_ConfigEx (ByVal CardNumber As
    Integer, ByVal ClkSrc As Integer, ByVal
    TrigSrcCtrl As Long, ByVal ReTrgEn As
    Integer, ByVal ReTrgCnt As Long, ByVal
    AutoResetBuf As Byte) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.



*ClkSrc*

A/D clock source settings. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H.

### **A/D Conversion Source Selection**

DAQ2K_AI_ADCONVSRC_Int	Internal timer (default)
DAQ2K_AI_ADCONVSRC_AFI0	From AFI0 pin
DAQ2K_AI_ADCONVSRC_SSI	From SSI source
DAQ2K_AI_ADCONVSRC_AFI1	From AFI1 pin (available only on DAQ-2501/2502)
DAQ2K_AI_ADCONVSRC_AFI2	From AFI2 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI3	From AFI3 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI4	From AFI4 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI5	From AFI5 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI6	From AFI6 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI7	From AFI7 pin (DAQ-2020/2022 only)

*TrigSrcCtrl*

A/D trigger source control setting. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H. There are two groups of constants:

### **Trigger Source Selection**

DAQ2K_AI_TRGSRC_SOFT	Software (Default)
DAQ2K_AI_TRGSRC_ANA	From analog trigger pin
DAQ2K_AI_TRGSRC_ExtD	From external digital trigger pin
DAQ2K_AI_TRSRC_SSI	From SSI source
DAQ2K_AI_TRGSRC_AFI0	From AFI0 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI1	From AFI1 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI2	From AFI2 pin (DAQ-2020/2022 only)

DAQ2K_AI_TRGSRC_AFI3	From AFI3 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI4	From AFI4 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI5	From AFI5 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI6	From AFI6 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI7	From AFI7 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_PXISar	PXI Star Trigger as the trigger source (available only for DAQ-2020/2022)
DAQ2K_AI_TRGSRC_SMB	From SMB Trigger IO Connector (available only for DAQ-2020/2022)

### External Digital Trigger Polarity

DAQ2K_AI_TrgPositive	Trigger positive edge active (Default)
DAQ2K_AI_TrgNegative	Trigger negative edge active

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

### *ReTrgEn*

0	Re-trigger in an acquisition is disabled. (Default)
1	Re-trigger in an acquisition is enabled.

### *ReTrgCnt*

The accepted trigger times in an acquisition. If the value is 0, the AI operation is triggered infinitely. For D2K\_AI\_PostTrig\_Config, the range of valid value is 0 to 65535. For the DAQ-2020/2022, using the function D2K\_AI\_PostTrig\_ConfigEx, the valid value range of ReTrgCnt is 0 to 4294967295.

---

### NOTE

To enable infinite re-trigger mode for continuous AI, call this function, then set the ReTrgEn to 1 and ReTrgCnt to 0.

---

*AutoResetBuf*

- FALSE

The AI buffers set by the D2K\_AI\_ContBufferSetup function are retained and D2K\_AI\_ContBufferReset must be called to reset the buffer.
- TRUE

The AI buffers set by the D2K\_AI\_ContBufferSetup function are reset automatically by the driver when AI operation is finished.

**Return Code**

- NoError
- ErrorInvalidCardNumber
- ErrorCardNotRegistered
- ErrorFuncNotSupport

## D2K\_AI\_PreTrig\_Config D2K\_AI\_PreTrig\_ConfigEx

### Description

Informs the D2K-DASK library of the conversion clock source and trigger properties of the device performing pre-triggered data acquisition operation.

The D2K\_AI\_PreTrig\_ConfigEx function is used for the devices (e.g. DAQ-2020/2022) which counters width are 32-bit long.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214

### Syntax

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

```
I16 D2K_AI_PreTrig_Config (U16 CardNumber, U16
    ClkSrc, U32 TrigSrcCtrl, U16 MCtrEn, U16
    MCnt, BOOLEAN AutoResetBuf)
I16 D2K_AI_PreTrig_ConfigEx (U16 CardNumber, U16
    ClkSrc, U32 TrigSrcCtrl, U16 MCtrEn, U32
    MCnt, BOOLEAN AutoResetBuf)
```

### Visual Basic

```
D2K_AI_PreTrig_Config (ByVal CardNumber As
    Integer, ByVal ClkSrc As Integer, ByVal
    TrigSrcCtrl As Integer, ByVal MCtrEn As
    Integer, ByVal MCnt As Integer, ByVal
    AutoResetBuf As Byte) As Integer
D2K_AI_PreTrig_ConfigEx (ByVal CardNumber As
    Integer, ByVal ClkSrc As Integer, ByVal
    TrigSrcCtrl As Long, ByVal MCtrEn As
    Integer, ByVal MCnt As Long, ByVal
    AutoResetBuf As Byte) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*ClkSrc* A/D clock source settings. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H.

**A/D Conversion Source Selection**

DAQ2K_AI_ADCONVSRC_Int	Internal timer (default)
DAQ2K_AI_ADCONVSRC_AFI0	From AFI0 pin
DAQ2K_AI_ADCONVSRC_SSI	From SSI source
DAQ2K_AI_ADCONVSRC_AFI2	From AFI2 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI3	From AFI3 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI4	From AFI4 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI5	From AFI5 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI6	From AFI6 pin (DAQ-2020/2022 only)
DAQ2K_AI_ADCONVSRC_AFI7	From AFI7 pin (DAQ-2020/2022 only)

*TrigSrcCtrl* A/D trigger source control setting. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H. There are two groups of constants:

**Trigger Source Selection**

DAQ2K_AI_TRGSRC_SOFT	Software (Default)
DAQ2K_AI_TRGSRC_ANA	From analog trigger pin
DAQ2K_AI_TRGSRC_ExtD	From external digital trigger pin
DAQ2K_AI_TRSRC_SSI	From SSI source
DAQ2K_AI_TRGSRC_AFI0	From AFI0 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI1	From AFI1 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI2	From AFI2 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI3	From AFI3 pin (DAQ-2020/2022 only)

DAQ2K_AI_TRGSRC_AFI4	From AFI4 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI5	From AFI5 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI6	From AFI6 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_AFI7	From AFI7 pin (DAQ-2020/2022 only)
DAQ2K_AI_TRGSRC_PXIStar	PXI Star Trigger as the trigger source (available only for DAQ-2020/2022)
DAQ2K_AI_TRGSRC_SMB	From SMB Trigger IO Connector (available only for DAQ-2020/2022)

### External Digital Trigger Polarity

DAQ2K_AI_TrgPositive	Trigger positive edge active (Default)
DAQ2K_AI_TrgNegative	Trigger negative edge active

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

### *MCtrEn*

0	Mcounter is disabled. (Default)
1	Mcounter is enabled and the trigger signal is ignored before M terminal count is reached.

### *MCnt*

Counter value of Mcounter. For D2K\_AI\_PreTrig\_Config, the range of valid value is 0 to 65535. For the DAQ-2020/2022, using the function D2K\_AI\_MiddleTrigConfigEx, the valid value range of MCnt is 0 to 4294967295.

### *AutoResetBuf*

FALSE	The AI buffers set by the D2K_AI_ContBufferSetup function are retained and D2K_AI_ContBufferReset must be called to reset the buffer.
TRUE	The AI buffers set by the D2K_AI_ContBufferSetup function are reset automatically by the driver when AI operation is finished.

---

**NOTE**      If Mcounter is enabled, the ReadScans parameter of continuous AI functions D2K\_AI\_ContXXXX has to be equal with MCnt.

---

### **Return Code**

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport

## D2K\_AI\_ReadChannel

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_ReadChannel (U16 CardNumber, U16
                        Channel, U16 *Value)
```

### Visual Basic

```
D2K_AI_ReadChannel (ByVal CardNumber As Integer,
                    ByVal Channel As Integer, Value As Integer)
                    As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*Channel* Analog input channel number. Range:

DAQ-2010, DAQ-2005, DAQ-2006, DAQ-2016 DAQ-2502	0 to 3
DAQ-2204, DAQ-2205, DAQ-2206 DAQ-2208	0 to 63 0 to 95
DAQ-2213, DAQ-2214, DAQ-2020, DAQ-2022	0 to 15
DAQ-2501	0 to 7

*Value* The converted A/D value. For the correct data format, refer to the description of the Buffer argument in the D2K\_AI\_ContReadChannel() function.

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```



ErrorInvalidAdRange

## D2K\_AI\_ReadMuxScan

### Description

Returns the readings for all analog input channels selected by D2K\_AI\_MuxScanSetup. This function is available only for multiplexed AD card.

### Supported Cards

2204, 2205, 2206, 2208, 2213, 2214

### Syntax

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

```
I16 D2K_AI_ReadMuxScan (U16 CardNumber, U16
                        *Buffer)
```

Visual Basic

```
D2K_AI_ReadMuxScan (ByVal CardNumber As Integer,
                    Buffer As Integer) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*Buffer* An integer array containing the acquired data. Refer to **Appendix C: AI Data Format** for the correct data format.

### Return Code

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

## D2K\_AI\_ScanReadChannels

### Description

This function performs software triggered A/D conversions (analog input) on analog input channels and returns the values converted. This function is only available for Multiplexed AD card (e.g. DAQ-2205).

### Supported Cards

2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_ScanReadChannels (U16 CardNumber, U16  
    NumChans, U16 *Chans, U16 *Buffer)
```

### Visual Basic

```
D2K_AI_ScanReadChannels (ByVal CardNumber As  
    Integer, ByVal NumChans As Integer, Chans As  
    Integer, Buffer As Integer) As Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.																
<i>NumChans</i>	Number of analog input channels in the array Chans. The valid value: <table><tr><td>DAQ-2204</td><td>1 to 512</td></tr><tr><td>DAQ-2205</td><td>1 to 512</td></tr><tr><td>DAQ-2206</td><td>1 to 512</td></tr><tr><td>DAQ-2208</td><td>1 to 1024</td></tr><tr><td>DAQ-2213</td><td>1 to 512</td></tr><tr><td>DAQ-2214</td><td>1 to 512</td></tr><tr><td>DAQ-2501</td><td>1 to 8</td></tr><tr><td>DAQ-2502</td><td>1 to 4</td></tr></table>	DAQ-2204	1 to 512	DAQ-2205	1 to 512	DAQ-2206	1 to 512	DAQ-2208	1 to 1024	DAQ-2213	1 to 512	DAQ-2214	1 to 512	DAQ-2501	1 to 8	DAQ-2502	1 to 4
DAQ-2204	1 to 512																
DAQ-2205	1 to 512																
DAQ-2206	1 to 512																
DAQ-2208	1 to 1024																
DAQ-2213	1 to 512																
DAQ-2214	1 to 512																
DAQ-2501	1 to 8																
DAQ-2502	1 to 4																

### *Chans*

Array of analog input channel numbers.

DAQ-2204	Numbers in Chans must be within 0 and 63.
DAQ-2205	Numbers in Chans must be within 0 and 63.
DAQ-2206	Numbers in Chans must be within 0 and 63.
DAQ-2208	Numbers in Chans must be within 0 and 95.
DAQ-2213	Numbers in Chans must be within 0 and 15.
DAQ-2214	Numbers in Chans must be within 0 and 15.
DAQ-2501	Numbers in Chans must be within 0 and 7.
DAQ-2502	Numbers in Chans must be within 0 and 3.

### *Buffer*

An integer array containing the acquired data. The length (in samples) of Buffer must be equal to or greater than the value of numChans. Refer to **Appendix C: AI Data Format** for the correct data format.

### **Return Code**

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

## D2K\_AI\_SimuReadChannel

### Description

Performs a software triggered A/D conversion (analog input) on the analog input channels and returns the converted values. This function is available only for simultaneous AD cards.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022

### Syntax

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

```
I16 D2K_AI_SimuReadChannel (U16 CardNumber, U16  
    NumChans, U16 *Chans, U16 *Buffer)
```

### Visual Basic

```
D2K_AI_SimuReadChannel (ByVal CardNumber As  
    Integer, ByVal NumChans As Integer, Chans As  
    Integer, Buffer As Integer) As Integer
```

### Parameters

- CardNumber** ID of the card performing the operation.
- NumChans** Number of analog input channels in the array Chans. The valid value:
- |             |         |
|-------------|---------|
| DAQ-2010    | 1 to 4  |
| DAQ-2005    | 1 to 4  |
| DAQ-2006    | 1 to 4  |
| DAQ-2016    | 1 to 4  |
| DAQ-2020/22 | 1 to 16 |
- Chans** Array of analog input channel numbers.
- |             |   |
|-------------|---|
| DAQ-2010    | Numbers in Chans must be within 0 and 3.  |
| DAQ-2005    | Numbers in Chans must be within 0 and 3.  |
| DAQ-2006    | Numbers in Chans must be within 0 and 3.  |
| DAQ-2016    | Numbers in Chans must be within 0 and 3.  |
| DAQ-2020/22 | Numbers in Chans must be within 0 and 15. |
- Buffer** An integer array containing the acquired data. The length (in samples) of Buffer must be equal to or greater than the value of numChans. Refer to

## Appendix C: AI Data Format for the correct data format.

### Return Code

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

## D2K\_AI\_VoltScale

### Description

This function converts the result from a D2K\_AI\_ReadChannel call to the actual input voltage.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AI_VoltScale (U16 CardNumber, U16  
    AdRange, I16 reading, F64 *Voltage)
```

### Visual Basic

```
D2K_AI_VoltScale (ByVal CardNumber As Integer,  
    ByVal AdRange As Integer, ByVal reading As  
    Integer, Voltage As Double) As Integer
```

### Parameters

- |                   |  |
|-------------------|--|
| <i>CardNumber</i> | ID of the card performing the operation.   |
| <i>AdRange</i>    | The analog input range setting for the specified channel. Refer to <b>Appendix B: AI Range Codes</b> for the range valid values. |
| <i>reading</i>    | AD conversion result.  |
| <i>Voltage</i>    | Computed voltage value.  |

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidAdRange
```

## D2K\_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.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```

I16 D2K_AI_VReadChannel (U16 CardNumber, U16
                        Channel, F64 *Voltage)
```

### Visual Basic

```

D2K_AI_VReadChannel (ByVal CardNumber As Integer,
                    ByVal Channel As Integer, Voltage As Double)
                    As Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.
<i>Channel</i>	Analog input channel number. Range: <div> <div>DAQ-2010, DAQ-2005, DAQ-2006, DAQ-2016 DAQ-2502</div> <div>0 to 3</div> </div> <div> <div>DAQ-2204, DAQ-2205, DAQ-2206</div> <div>0 to 63</div> </div> <div> <div>DAQ-2208</div> <div>0 to 95</div> </div> <div> <div>DAQ-2213, DAQ-2214, DAQ-2020, DAQ-2022</div> <div>0 to 15</div> </div> <div> <div>DAQ-2501</div> <div>0 to 7</div> </div>
<i>Voltage</i>	The measured voltage value returned and scaled to units of voltage.

### Return Code

```

NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

ErrorInvalidAdRange



## D2K\_AIO\_Config

### Description

Informs the D2K-DASK library of the timer source and the analog trigger setting of the device with CardNumber ID. You must call this function when using an external timer source or when performing an analog trigger mode of AI/AO.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
U16 D2K_AIO_Config (U16 CardNumber, U16
    TimerBase, U16 AnaTrigCtrl, U16 H_TrgLevel,
    U16 L_TrgLevel)
```

### Visual Basic

```
D2K_AIO_Config (ByVal CardNumber As Integer,
    ByVal TimerBase As Integer, ByVal
    AnaTrigCtrl As Integer, ByVal H_TrgLevel As
    Integer, ByVal L_TrgLevel As Integer) As
    Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.	
<i>TimerBase</i>	The selected timebase for the device. Valid values:	
	DAQ2K_IntTimeBase	Internal timer as the time base
	DAQ2K_ExtTimeBase	External timer as the time base (not valid for DAQ-2020/2022)
	DAQ2K_SSITimeBase	The timer based on the SSI source
	DAQ2K_ExtTimeBase_AFI(n)	From AFI0 -AFI7 pin, where n is 0 through 7 (available only for DAQ-2020/2022)
	DAQ2K_PXI_CLK	From PXI Backplane Clock (available only for DAQ-2020/2022)

DAQ2K_StarTimeBase	PXI Star Trigger as the timebase (available only for DAQ-2020/2022)
DAQ2K_SMBTimeBase	From SMB CLK IN (available only for DAQ-2020/2022)

*AnaTrigCtrl* Analog trigger control setting. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H. There are two groups of constants:

### Trigger Source Selection

DAQ2010, DAQ2005, DAQ2006, DAQ2016

CH0ATRIG	AI channel 0
CH1ATRIG	AI channel 1
CH2ATRIG	AI channel 2
CH3ATRIG	AI channel 3
EXTATRIG	From external analog trigger pin
DAQ2204, DAQ2205, DAQ2206, DAQ2208, DAQ2213, DAQ2214, DAQ2501, DAQ2502	
ADCATRIG	The first AI channel in the channel-gain queue
EXTATRIG	From external analog trigger pin

### Trigger Condition Selection

Below_Low_level	Below-Low-Level Triggering
Above_High_Level	Above-High-Level Triggering
Inside_Region	Inside Region Triggering
High_Hysteresis	High Hysteresis Triggering
Low_Hysteresis	Low Hysteresis Triggering

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

*H\_TrgLevel* The high value setting of the trigger level. The range of valid value is 1 to 256. Refer to the device documentation for information on the relationship between the value of *TrgLevel* and trigger voltage.

*L\_TrgLevel* The low value setting of the trigger level. The range of valid value is 1 to 255. Refer to the device

documentation for information on the relationship between the value of TrgLevel and trigger voltage.

For example:

If the trigger voltage is 10 V, the relationship between the value of TrgLevel and trigger voltage is shown in the following table:

Trigger Level digital setting	Trigger voltage
0xFF	9.92V
0xFE	9.84V
—	—
0x81	0.08V
0x80	0
0x7F	-0.08V
—	—
0x02	-9.92V
0x01	-10V

Return Code

NoError

ErrorInvalidCardNumber

ErrorCardNotRegistered

ErrorFuncNotSupport

## D2K\_AO\_AsyncCheck

### Description

Checks the current status of the asynchronous analog output operation. This function is available only for devices that use timer pacer (DAQ2K\_DA\_WRSRC\_Int) as D/A R/W source.

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214

### Syntax

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

```
I16 D2K_AO_AsyncCheck (U16 CardNumber, BOOLEAN  
    *Stopped, U32 WriteCnt)
```

### Visual Basic

```
D2K_AO_AsyncCheck (ByVal CardNumber As Integer,  
    Stopped As Byte, WriteCnt As Long) As  
    Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*Stopped* Tells whether the asynchronous analog output operation is completed. If Stopped = TRUE, then the analog output operation has stopped after the number of D/A conversions indicated in the call that initiated the asynchronous analog output operation has completed or after an error occurred. If Stopped = FALSE, the operation is not yet completed. Constants TRUE and FALSE are defined in D2KDASK.H.

*WriteCnt* Number of analog output data written during the time when D2K\_AO\_AsyncCheck() is called.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport
```

## D2K\_AO\_AsyncClear

### Description

Stops the asynchronous analog output operation.

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214

### Syntax

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

```
I16 D2K_AO_AsyncClear (U16 CardNumber, U32  
    *UpdateCnt, U16 stop_mode)
```

### Visual Basic

```
D2K_AO_AsyncClear (ByVal CardNumber As Integer,  
    UpdateCnt As Long, stop_mode As Integer) As  
    Integer
```

### Parameters

*CardNumber* ID of the card performing asynchronous operation.

*UpdateCnt* Number of analog output data that have been written at the time calling D2K\_AO\_AsyncCheck().

*stop\_mode* Selected DA transfer termination mode. Valid values:

DAQ2K_DA_TerminateImmediate	Software terminates the continuous DA operation immediately
DAQ2K_DA_TerminateUC	Software terminates the continuous DA operation on the next counter terminal count update.
DAQ2K_DA_TerminateIC	Software terminates the continuous DA operation on iteration count.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport
```

## D2K\_AO\_AsyncDblBufferHalfReady

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AO_AsyncDblBufferHalfReady (U16  
    CardNumber, BOOLEAN *HalfReady)
```

### Visual Basic

```
D2K_AO_AsyncDblBufferHalfReady(ByVal CardNumber  
    As Integer, HalfReady As Byte) As Integer
```

### Parameters

- CardNumber* ID of the card performing asynchronous double-buffered operation.
- HalfReady* Tells whether the next half buffer is ready for new data.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport
```

## D2K\_AO\_AsyncDblBufferMode

### Description

Enables or disables double-buffered data acquisition mode.

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AO_AsyncDblBufferMode (U16 CardNumber,  
                                BOOLEAN Enable)
```

Visual Basic

```
D2K_AO_AsyncDblBufferMode (ByVal CardNumber As  
                             Integer, ByVal Enable As Byte) As Integer
```

### Parameters

*CardNumber* ID of the card where double-buffered mode is to be set.

*Enable* Tells whether the double-buffered mode is enabled or not. Constants TRUE and FALSE are defined in D2KDASK.H.

TRUE Double-buffered mode is enabled.

FALSE Double-buffered mode is disabled.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport
```

## D2K\_AO\_CH\_Config

### Description

Informs D2K-DASK library of the reference voltage value selected for an analog output channel of DAQ-2000 Device. You can configure each channel to use an internal reference of 10V (default) or an external reference (-10V ~ +10V). After the function Register\_Card is called, all of the analog output channels are configured as bipolar and internal reference source by default. If you wish to perform the device with the default settings, it is not necessary to call this function to configure the channel(s) again. Otherwise, this function has to be called to program the device for the settings you want before calling function to perform voltage output operation.

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AO_CH_Config (U16 CardNumber, U16  
    Channel, U16 OutputPolarity, U16  
    IntOrExtRef, F64 refVoltage)
```

### Visual Basic

```
D2K_AO_CH_Config (ByVal CardNumber As Integer,  
    ByVal Channel As Integer, ByVal  
    OutputPolarity As Integer, ByVal IntOrExtRef  
    As Integer, ByVal refVoltage As Double) As  
    Integer
```



Parameters

*CardNumber* ID of the card performing the operation.

*Channel* Configured AO channel number.

DAQ-2010	0 to 1 or All_Channels (-1)
DAQ-2005	0 to 1 or All_Channels (-1)
DAQ-2006	0 to 1 or All_Channels (-1)
DAQ-2016	0 to 1 or All_Channels (-1)
DAQ-2204	0 to 1 or All_Channels (-1)
DAQ-2205	0 to 1 or All_Channels (-1)
DAQ-2206	0 to 1 or All_Channels (-1)
DAQ-2214	0 to 1 or All_Channels (-1)
DAQ-2501	0 to 3 or All_Channels (-1)
DAQ-2502	0 to 7 or All_Channels (-1)

*OutputPolarity* Polarity (unipolar or bipolar) of the output channel.  
Valid values:

DAQ2K\_DA\_BiPolar

DAQ2K\_DA\_UniPolar

*IntOrExtref* DA reference voltage source of the output channel.  
Valid values:

DAQ2K_DA_Int_REF	Internal reference
DAQ2K_DA_Ext_REF	External reference

*refVoltage* Voltage reference value. If the device uses an internal D/A reference voltage source, the valid value for refVoltage is 10. If the device uses an external D/A reference voltage source, the range of valid value for refVoltage is -10 to +10.

Return Code

NoError

ErrorInvalidCardNumber

ErrorCardNotRegistered

ErrorFuncNotSupport

ErrorInvalidDaRefVoltage

## D2K\_AO\_Config

### Description

Informs the D2K-DASK library of the selected trigger source for the device CardNumber ID. After calling the Register\_Card function, the device is configured to the following by default:

D/A R/W source	DAQ2K_DA_WRSRC_Int
D/A trigger mode	DAQ2K_DA_TRGMOD_POST
D/A trigger source	DAQ2K_DA_TRGSRC_SOFT
Auto reset buffer	Enabled (AutoResetBuf : TRUE)

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

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AO_Config (U16 CardNumber, U16  
    ConfigCtrl, U16 TrigCtrl, U16 ReTrgCnt, U16  
    DLY1Cnt, U16 DLY2Cnt, BOOLEAN AutoResetBuf)
```

### Visual Basic

```
D2K_AO_Config (ByVal CardNumber As Integer, ByVal  
    ConfigCtrl As Integer, ByVal TrigCtrl As  
    Integer, ByVal ReTrgCnt As Integer, ByVal  
    DLY1Cnt As Integer, ByVal DLY2Cnt As  
    Integer, ByVal AutoResetBuf As Byte) As  
    Integer
```

### Parameters

- CardNumber* ID of the card performing the operation.
- ConfigCtrl* D/A configuration control setting. This argument is an integer expression formed from one or more of the manifest constants defined in D2kDASK.H. There are four group of constants:

### D/A R/W Source Selection

DAQ2K_DA_WRSRC_Int	Internal timer (Default)
DAQ2K_DA_WRSRC_AFI0	From AFI0 pin (available only on DAQ-2501/2502)
DAQ2K_DA_WRSRC_AFI1	From AFI1 pin (NOT available on DAQ-2501/2502)
DAQ2K_DA_WRSRC_SSI	From SSI source

The following constant groups are only available for DAQ-2501 and DAQ-2502

### DA group Selection

DA_Group_A	DA group A
DA_Group_B	DA group B
DA_Group_AB	DA group A and group B

### D/A Trigger delay Counter Source Selection

DAQ2K_DA_TDSRC_Int	Internal timer (Default)
DAQ2K_DA_TDSRC_AFI0	From AFI0 pin
DAQ2K_DA_TDSRC_GPTC0	From GPTC0_OUT pin
DAQ2K_DA_TDSRC_GPTC1	From GPTC1_OUT pin

### D/A Break delay Counter Source Selection

DAQ2K_DA_BDSRC_Int	Internal timer (Default)
DAQ2K_DA_BDSRC_AFI0	From AFI0 pin
DAQ2K_DA_BDSRC_GPTC0	From GPTC0_OUT pin
DAQ2K_DA_BDSRC_GPTC1	From GPTC1_OUT pin

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 D/A Trigger control. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H. There are seven groups of constants:

### **Trigger Source Selection**

DAQ2K_DA_TRGSRC_SOFT	Software (Default)
DAQ2K_DA_TRGSRC_ANA	From analog trigger pin
DAQ2K_DA_TRGSRC_ExtD	From external digital trigger pin
DAQ2K_DA_TRSRC_SSI	From SSI source

### **Trigger Mode Selection**

DAQ2K_DA_TRGMOD_POST	Post Trigger Mode (Default)
DAQ2K_DA_TRGMOD_DELAY	Delay Trigger Mode

### **Re-Trigger Mode Enable**

(available only for post and delay trigger modes)

DAQ2K_DA_ReTrigEn	Re-trigger in an acquisition is enabled.
-------------------	--

### **Delay2 (Break delay) Mode Enable**

DAQ2K_DA_DLY2En	Delay2/Break delay (delay between two consecutive waveform generations) in an acquisition is enabled.
-----------------	---

### **Delay1 Source Selection**

(only available for delay trigger mode)

DAQ2K_DA_Dly1InUI	Delay in samples (not valid for DAQ-2501/2502)
DAQ2K_DA_Dly1InTimebase	Delay in time base (Default)

### **Delay2 Source Selection**

DAQ2K_DA_Dly2InUI	Delay in samples (not valid for DAQ-2501/2502)
DAQ2K_DA_Dly2InTimebase	Delay in time base (Default)

### **External Digital Trigger Polarity**

DAQ2K_DA_TrqPositive	Trigger positive edge active (Default)
DAQ2K_DA_TrqNegative	Trigger negative edge active

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

*ReTrgCnt* The accepted trigger times in an acquisition. If the value of ReTrgCnt is 0, the fixed pattern generation is triggered infinitely. This argument is valid only for delay trigger and post trigger modes. The range of valid value is 0 to 65535.

---

**NOTE** To enable infinite re-trigger mode of fixed pattern generation, call D2K\_AO\_Config with DAQ2K\_DA\_ReTrigEn and assign a zero value to ReTrgCnt. To receive notification when the pattern generation associated to the next trigger signal of Group AB is completed, you can assign a callback function to D2K\_AO\_EventCallBack with DA-TrigEvent/DATrigEvent\_A, DATrigEvent\_B, or DATrigEvent\_AB of event type.

---

*DLY1Cnt* DLY1 counter value or the delay time to start waveform generation after the trigger signal. This argument is valid only for delay trigger mode. The range of valid value is 0 to 65535.

*DLY2Cnt* DLY2 counter value or the delay between two consecutive waveform generations. The range of valid value is 0 to 65535.

*AutoResetBuf* For DAQ-2502/2501, this parameter must be set to FALSE.

- FALSE The DA buffer set by the D2K\_AO\_ContBufferSetup function are retained. The D2K\_AO\_ContBufferReset fuction must be called to reset the buffer.
- TRUE The DA buffer set by the D2K\_AO\_ContBufferSetup function are reset automatically by driver when the AI operation is completed.

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

## D2K\_AO\_ContBufferCompose

### Description

Fills the data for a specified channel in the buffer for continuous analog output operation. The filled positions of the data in the buffer depend on the device type. Except for DAQ-2501/2502, this function may only be used for multi-channels of continuous analog output (waveform generation) operation.

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AO_ContBufferCompose (U16 CardNumber, U16
    group, U16 Channel, U32 UpdateCount, void
    *ConBuffer, void *Buffer, BOOLEAN fifoload)
```

### Visual Basic

```
D2K_AO_ContBufferCompose (ByVal CardNumber As
    Integer, ByVal group As Integer, ByVal
    Channel As Integer, ByVal UpdateCount As
    Long, ConBuffer As Any, Buffer As Any, ByVal
    fifoload As Byte) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*group* Group of analog output channels. Valid values:

DAQ-2010, DAQ-2005, DAQ-2006, DAQ-2016, DAQ-2204, DAQ-2205, DAQ-2206, DAQ-2214	No use
DAQ-2501	DA_Group_A
DAQ-2502	DA_Group_A, DA_Group_B and DA_Group_AB

<i>Channel</i>	Configured AO channel number.
	DAQ-2010      0 to 1 or All_Channels (-1)
	DAQ-2005      0 to 1 or All_Channels (-1)
	DAQ-2006      0 to 1 or All_Channels (-1)
	DAQ-2016      0 to 1 or All_Channels (-1)
	DAQ-2204      0 to 1 or All_Channels (-1)
	DAQ-2205      0 to 1 or All_Channels (-1)
	DAQ-2206      0 to 1 or All_Channels (-1)
	DAQ-2214      0 to 1 or All_Channels (-1)
	DAQ-2501      0 to 3 or All_Channels (-1)
	DAQ-2502      0 to 7 or All_Channels (-1)
<i>UpdateCount</i>	Size (in samples) of the specified channel buffer. This is not the size of the buffer for continuous output operation.
<i>ConBuffer</i>	Buffer for continuous output operation.
<i>Buffer</i>	Buffer containing the output data for the specified channel.
<i>fifoload</i>	Allows loading of data into the onboard DA FIFO using the D2K_AO_Group_FIFOLoad function. This parameter is valid only for DAQ-2502.
	0      Data will not be loaded into DA FIFO.
	1      Data will be loaded into DA FIFO.

## Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorContIoNotAllowed
```

## D2K\_AO\_ContBufferComposeAll

### Description

Organizes the data for each channel and fills them in the buffer for continuous analog output operation. The filled positions of the data in the buffer depends on device type. Except for DAQ-2501/2502, this function may only be used for multi-channels of continuous analog output (waveform generation) operation.

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AO_ContBufferComposeAll (U16 CardNumber,  
                                U16 group, U32 UpdateCount, void *ConBuffer,  
                                void *Buffer, BOOLEAN fifoload)
```

### Visual Basic

```
D2K_AO_ContBufferCompose (ByVal CardNumber As  
                          Integer, ByVal group As Integer, ByVal  
                          UpdateCount As Long, ConBuffer As Any,  
                          Buffer As Any, ByVal fifoload As Byte) As  
                          Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.						
<i>group</i>	Group of analog output channels. Valid values: <table><tr><td>DAQ-2010, DAQ-2005, DAQ-2006, DAQ-2016, DAQ-2204, DAQ-2205, DAQ-2206, DAQ-2214</td><td>No use</td></tr><tr><td>DAQ-2501</td><td>DA_Group_A</td></tr><tr><td>DAQ-2502</td><td>DA_Group_A, DA_Group_B and DA_Group_AB</td></tr></table>	DAQ-2010, DAQ-2005, DAQ-2006, DAQ-2016, DAQ-2204, DAQ-2205, DAQ-2206, DAQ-2214	No use	DAQ-2501	DA_Group_A	DAQ-2502	DA_Group_A, DA_Group_B and DA_Group_AB
DAQ-2010, DAQ-2005, DAQ-2006, DAQ-2016, DAQ-2204, DAQ-2205, DAQ-2206, DAQ-2214	No use						
DAQ-2501	DA_Group_A						
DAQ-2502	DA_Group_A, DA_Group_B and DA_Group_AB						
<i>UpdateCount</i>	Size (in samples) of the specified channel buffer. This is not the size of the buffer for continuous output operation.						
<i>ConBuffer</i>	Buffer for continuous output operation.						



<i>Buffer</i>	Buffer containing the output data for the specified channel.
<i>fifoload</i>	Allows loading of data into the onboard DA FIFO using the D2K_AO_Group_FIFOLoad function. This parameter is valid only for DAQ-2502.
0	Data will not be loaded into DA FIFO.
1	Data will be loaded into DA FIFO.

## Return Code

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

## D2K\_AO\_ContBufferReset

### Description

Resets all buffers set by the D2K\_AO\_ContBufferSetup function for continuous analog output. This function must be called if the data buffers will be used.

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AO_ContBufferReset (U16 CardNumber)
```

### Visual Basic

```
D2K_AO_ContBufferReset (ByVal CardNumber As  
Integer) As Integer
```

### Parameter

*CardNumber* ID of the card performing the operation.

### Return Code

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

## D2K\_AO\_ContBufferSetup

### Description

This function set up the buffer for continuous analog output operation. The function has to be called repeatedly to setup all of the data buffers (Except DAQ-2502, the maximum number of buffers is 2. For DAQ-2502, the maximum number of buffers is 4.).

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214, 2501, 2502

### Syntax

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

```
U16 D2K_AO_ContBufferSetup (U16 CardNumber, void  
    *Buffer, U32 WriteCount, U16 *BufferId)
```

### Visual Basic

```
D2K_AO_ContBufferSetup (ByVal CardNumber As  
    Integer, Buffer As Any, ByVal WriteCount As  
    Long, BufferId As Integer) As Integer
```

### Parameters

- |                   |  |
|-------------------|--|
| <i>CardNumber</i> | ID of the card performing the operation.                                       |
| <i>Buffer</i>     | Starting address of the memory that contains the output data.                  |
| <i>WriteCount</i> | The size (in samples) of the buffer. The value of this parameter must be even. |
| <i>BufferId</i>   | Returns the index of the buffer currently being set up.                        |

### Return Code

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

## D2K\_AO\_ContStatus

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AO_ContStatus (U16 CardNumber, U16  
*Status)
```

### Visual Basic

```
D2K_AO_ContStatus (ByVal CardNumber As Integer,  
Status As Integer) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*Status* Returned continuous AO status. The description of this parameter for various card types is shown below:

#### **DAQ-2010, DAQ-2005, DAQ-2006, DAQ-2016, DAQ-2204, DAQ-2205, DAQ-2206, DAQ-2214**

bit 0	1 indicates that the D/A FIFO has underrun.
bit 1 to 3	Not used
bit 4	1 indicates that the D/A FIFO is empty.
bit 5	1 indicates that the D/A FIFO is half-full.
bit 6	1 indicates that the D/A FIFO is full
bit 7 to 15	Not used

### DAQ-2501, DAQ-2502

bit 0	1 indicates that the D/A FIFO of group A is not empty.
bit 1	Not used
bit 2	'1' indicates that the D/A FIFO of group A is not almost full.
bit 3	Not used
bit 4	1 indicates that the D/A FIFO of group B is not empty.
bit 5	Not used
bit 6	1 indicates that the D/A FIFO of group B is not almost full.
bit 7	Not used
bit 8 to 15	Not used

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
```

## D2K\_AO\_ContWriteChannel

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214

### Syntax

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

```
I16 D2K_AO_ContWriteChannel (U16 CardNumber, U16
    Channel, U16 BufId, U32 UpdateCount, U32
    Iterations, U32 CHUI, U16 definite, U16
    SyncMode)
```

### Visual Basic

```
D2K_AO_ContWriteChannel (ByVal CardNumber As
    Integer, ByVal Channel As Integer, ByVal
    BufId As Integer, ByVal UpdateCount As Long,
    ByVal Iterations As Long, ByVal CHUI As
    Long, ByVal definite As integer, ByVal
    SyncMode As Integer) As Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.
<i>Channel</i>	Analog output channel number. Range is 0 to 1.
<i>BufId</i>	The buffer ID (returned from function D2K_AO_ContBufferSetup) of the buffer containing the acquired data. The size of the buffer with this parameter as buffer id must have a length (in samples) equal to the value of UpdateCount.
<i>UpdateCount</i>	If double-buffered mode is disabled, this is the total update count for each channel to be performed. For double-buffered acquisition, UpdateCount is the size (in samples) allocated for each channel in the circular buffer and its value must be a multiple of 2.
<i>Iterations</i>	The number of times the data in the buffer outputs to the port. From D2K-DASK version 1.1, a value of

	zero is not allowed. If the DA operation is perform synchronously, this argument must be set to 1.				
<i>CHUI</i>	Length of the Channel Update interval (the counter value between the initiation of each update sequence).				
	When the device has an external time base, the range of valid value is 8 to 16777215. If the time base is internal, the range of valid value is 40 to 16777215.				
<i>definite</i>	Waveform generation proceeds in definite or indefinite manner. If double-buffered mode is enabled, this parameter is ignored.				
	<table> <tr> <td>0</td><td>Indefinite</td></tr> <tr> <td>1</td><td>Definite</td></tr> </table>	0	Indefinite	1	Definite
0	Indefinite				
1	Definite				
<i>SyncMode</i>	Tells whether this operation is performed synchronously or asynchronously. If any trigger mode is enabled by calling D2K_AO_Config(), this operation must be performed asynchronously. Valid value is ASYNCH_OP for asynchronous D/A conversion.				

## Return Code

```

NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
ErrorInvalidAdRange
ErrorTransferCountTooLarge
ErrorContIoNotAllowed
ErrorInvalidSampleRate

```

## D2K\_AO\_ContWriteMultiChannels

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214

### Syntax

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

```
I16 D2K_AO_ContWriteMultiChannels (U16  
    CardNumber, U16 NumChans, U16 *Chans, U16  
    BufId, U32 UpdateCount, U32 Iterations, U32  
    CHUI, U16 definite, U16 SyncMode)
```

### Visual Basic

```
D2K_AO_ContReadMultiChannels (ByVal CardNumber As  
    Integer, ByVal NumChans As Integer, Chans As  
    Integer, ByVal BufId As Integer, ByVal  
    UpdateCount As Long, ByVal Iterations As  
    Long, ByVal CHUI As Long, ByVal definite As  
    integer, ByVal SyncMode As Integer) As  
    Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.																
<i>NumChans</i>	Number of analog input channels in the array Chans. Valid value: <table><tr><td>DAQ-2010</td><td>1 to 2</td></tr><tr><td>DAQ-2005</td><td>1 to 2</td></tr><tr><td>DAQ-2006</td><td>1 to 2</td></tr><tr><td>DAQ-2016</td><td>1 to 2</td></tr><tr><td>DAQ-2204</td><td>1 to 2</td></tr><tr><td>DAQ-2205</td><td>1 to 2</td></tr><tr><td>DAQ-2206</td><td>1 to 2</td></tr><tr><td>DAQ-2214</td><td>1 to 2</td></tr></table>	DAQ-2010	1 to 2	DAQ-2005	1 to 2	DAQ-2006	1 to 2	DAQ-2016	1 to 2	DAQ-2204	1 to 2	DAQ-2205	1 to 2	DAQ-2206	1 to 2	DAQ-2214	1 to 2
DAQ-2010	1 to 2																
DAQ-2005	1 to 2																
DAQ-2006	1 to 2																
DAQ-2016	1 to 2																
DAQ-2204	1 to 2																
DAQ-2205	1 to 2																
DAQ-2206	1 to 2																
DAQ-2214	1 to 2																



<i>Chans</i>	<p>Array of analog output channel numbers. The channel order for update data is the same as the order you set in Chans.</p> <p>DAQ-2010      Numbers in Chans must be within 0 and 1.</p> <p>DAQ-2005      Numbers in Chans must be within 0 and 1.</p> <p>DAQ-2006      Numbers in Chans must be within 0 and 1.</p> <p>DAQ-2016      Numbers in Chans must be within 0 and 1.</p> <p>DAQ-2204      Numbers in Chans must be within 0 and 1.</p> <p>DAQ-2205      Numbers in Chans must be within 0 and 1.</p> <p>DAQ-2206      Numbers in Chans must be within 0 and 1.</p> <p>DAQ-2214      Numbers in Chans must be within 0 and 1.</p>
<i>BuId</i>	<p>ID (returned from function D2K_AO_ContBufferSetup) of the buffer containing the output data. The size of the buffer with BuId must have an equal length with or greater than the value of WriteCount x numChans.</p> <p>The data order in the buffer is in interleaved sequence. The data for channel 0 is stored in Buf[0], Buffer[2], Buffer[4], and so on. The data for channel 1 is stored in Buffer[1], Buffer[3], Buffer[5], and so on.</p>
<i>UpdateCount</i>	<p>If double-buffered mode is disabled, this is the total update count for each channel to be performed. For double-buffered acquisition, UpdateCount is the size (in samples) allocated for each channel in the circular buffer and its value must be a multiple of 2.</p>
<i>Iterations</i>	<p>The number of times the data in the buffer outputs to the port. From D2K-DASK version 1.1, a value of zero is not allowed. If the DA operation is perform synchronously, this argument must be set to 1.</p>
<i>CHUI</i>	<p>Length of the Channel Update interval (the counter value between the initiation of each update sequence).</p> <p>When the device has an external time base, the range of valid value is 8 to 16777215. If the time base is internal, the range of valid value is 40 to 16777215.</p>

*definite*            Waveform generation proceeds in definite or indefinite manner. If double-buffered mode is enabled, this parameter is ignored.

0	Indefinite
1	Definite

*SyncMode*        Tells whether this operation is performed synchronously or asynchronously. If any trigger mode is enabled by calling `D2K_AO_Config()`, this operation must be performed asynchronously. Valid value is `ASYNCH_OP` for asynchronous D/A conversion.

## Return Code

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

# D2K\_AO\_DelayTrig\_Config

## Description

Informs the D2K-DASK library of the update clock source and the trigger properties of the device performing delay triggered waveform generation operation.

## Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214, 2501, 2502

## Syntax

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

```
I16 D2K_AO_DelayTrig_Config (U16 CardNumber, U16
    ClkSrc, U16 TrigSrcCtrl, U16 DLY1Cnt, U16
    DLY2Ctrl, U16 DLY2Cnt, U16 ReTrgEn, U16
    ReTrgCnt, BOOLEAN AutoResetBuf)
```

## Visual Basic

```
D2K_AO_DelayTrig_Config (ByVal CardNumber As
    Integer, ByVal ClkSrc As Integer, ByVal
    TrigSrcCtrl As Integer, ByVal DLY1Cnt As
    Integer, ByVal DLY2Ctrl As Integer, ByVal
    DLY2Cnt As Integer, ByVal ReTrgEn As
    Integer, ByVal ReTrgCnt As Integer, ByVal
    AutoResetBuf As Byte) As Integer
```

## Parameters

- CardNumber* ID of the card that performing the operation.
- ClkSrc* The setting for D/A update clock source. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H. There are four groups of constants:

### D/A R/W Source Selection

DAQ2K_DA_WRSRC_Int	Internal timer (Default)
DAQ2K_DA_WRSRC_AFIO	From AFIO pin
DAQ2K_DA_WRSRC_SSI	From SSI source

The following constant groups are only available for DAQ-2501 and DAQ-2502:

### DA group Selection

DA_Group_A	DA group A
DA_Group_B	DA group B
DA_Group_AB	DA group A and group B

### D/A Trigger Delay Counter Source Selection

DAQ2K_DA_TDSRC_Int	Internal timer (Default)
DAQ2K_DA_TDSRC_AFI0	From AFI0 pin
DAQ2K_DA_TDSRC_GPTC0	From GPTC0_OUT pin
DAQ2K_DA_TDSRC_GPTC1	From GPTC1_OUT pin

### D/A Break Delay Counter Source Selection

DAQ2K_DA_BDSRC_Int	Internal timer (Default)
DAQ2K_DA_BDSRC_AFI0	From AFI0 pin
DAQ2K_DA_BDSRC_GPTC0	From GPTC0_OUT pin
DAQ2K_DA_BDSRC_GPTC1	From GPTC1_OUT pin

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

TrigSrcCtrl

The setting for D/A Trigger control. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H. There are three groups of constants:

### Trigger Source Selection

DAQ2K_DA_TRGSRC_SOFT	Software (Default)
DAQ2K_DA_TRGSRC_ANA	From analog trigger pin
DAQ2K_DA_TRGSRC_ExtD	From external digital trigger pin
DAQ2K_DA_TRSRC_SSI	From SSI source

### Delay1 Source Selection

DAQ2K_DA_Dly1InUI	Delay in samples (not valid for DAQ-2501/2502)
DAQ2K_DA_Dly1InTimebase	Delay in time base (Default)

### External Digital Trigger Polarity

DAQ2K_DA_TrgPositive	Trigger positive edge active (Default)
----------------------	--

DAQ2K\_DA\_TrgNegative      Trigger negative edge active

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

*DLY1Cnt*      DLY1 counter value or the delay time to start waveform generation after the trigger signal. This argument is valid only for delay trigger mode. The range of valid value is 0 to 65535.

*DLY2Ctrl*      D/A trigger control setting. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H. There are two groups of constants:

**Delay2 (Break delay) Mode Enable**

DAQ2K\_DA\_DLY2En      Delay2/Break delay (delay between two consecutive waveform generations) in an acquisition is enabled.

**Delay2 Source Selection**

DAQ2K\_DA\_Dly2InUI      Delay in samples (not valid for DAQ-2501/2502)

DAQ2K\_DA\_Dly2InTimebase      Delay in time base (Default)

When two or more constants are used to form the DLY2Ctrl argument, the constants are combined with the bitwise-OR operator(|).

*DLY2Cnt*      DLY2 counter value or the delay between two consecutive waveform generations. The range of valid value is 0 to 65535.

*ReTrgEn*

- 0      Re-trigger in an acquisition is disabled. (Default)
- 1      Re-trigger in an acquisition is enabled.

*ReTrgCnt*      The accepted trigger times in an acquisition. The range of valid value is 0 to 65535.

*AutoResetBuf* For DAQ-2502/2501, this parameter must be set to FALSE.

- |       |   |
|-------|---|
| FALSE | The DA buffer set by the D2K_AO_ContBufferSetup function are retained. The D2K_AO_ContBufferReset fuction must be called to reset the buffer. |
| TRUE  | The DA buffer set by the D2K_AO_ContBufferSetup function are reset automatically by driver when the AI operation is completed.                |

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport

## D2K\_AO\_EventCallBack

### Description

Controls and notifies the user's application when a specified DAQ event occurs. The notification is performed through a user-specified callback function.

In Linux, the event message has to be manually removed by setting **mode** to 0. In Windows, the event message is removed automatically by calling `D2K_AO_Async_Clear` or `D2K_AO_Group_WFM_AsyncClear`. The event message can also be removed manually by setting **mode** to 0.

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214, 2501, 2502

### Syntax

Microsoft C/C++ and Borland C++

```
I16 D2K_AO_EventCallBack (U16 CardNumber, I16
                        mode, I16 EventType, U32 callbackAddr)
```

Linux C++

```
I16 D2K_AO_EventCallBack (U16 CardNumber, I16
                        mode, I16 EventType, void
                        (*callbackAddr)(int))
```

Visual Basic 5

```
D2K_AO_EventCallBack (ByVal CardNumber As
                        Integer, ByVal mode As Integer, ByVal
                        EventType As Integer, ByVal callbackAddr As
                        Long) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*mode* Adds or removes the event message. Valid values:

- |   |        |
|---|--------|
| 0 | Remove |
| 1 | Add    |

*EventType* Event criteria. Valid values:

For DAQ-2005, DAQ-2006, DAQ-2010, DAQ-2016,  
DAQ-2204, DAQ-2205, DAQ-2206, DAQ-2214:

DBEvent	Notifies that the next half buffer of data in the circular buffer is ready for transfer.
DAQEnd	Notifies that the asynchronous analog output operation is completed.
DATrigEvent	Notifies that the pattern generation associated to the next trigger signal is completed.

For DAQ-2501 and DAQ-2502:

DBEvent	Notifies that the next half buffer of data in circular buffer is ready for transfer.
DAQEnd_A	Notifies that the asynchronous analog output operation of Group A is completed.
DAQEnd_B	Notifies that the asynchronous analog output operation of Group B is completed.
DAQEnd_AB	Notifies that the asynchronous analog output operation of Group AB is completed.
DATrigEvent_A	Notifies that the pattern generation associated to the next trigger signal of Group A is completed.
DATrigEvent_B	Notifies that the pattern generation associated to the next trigger signal of Group B is completed.
DATrigEvent_AB	Notifies that the pattern generation associated to the next trigger signal of Group AB is completed.

callbackAddr User callback function address. D2K-DASK calls this function when the specified event occurs. If you want to remove the event message, set callbackAddr to 0.

## Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```



# D2K\_AO\_Group\_FIFOLoad

## Description

Loads a waveform buffer to the onboard DA FIFOs.

## Supported Cards

2501, 2502

## Syntax

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

```
I16 D2K_AO_Group_FIFOLoad (U16 CardNumber, U16
                           group, U16 BufId, U32 UpdateCount)
```

## Visual Basic

```
D2K_AO_Group_FIFOLoad (ByVal CardNumber As
                        Integer, ByVal group As Integer, ByVal BufId
                        As Integer, ByVal UpdateCount As Long) As
                        Integer
```

## Parameters

- CardNumber*

ID of the card performing the operation.
- group*

Group of analog output channels. Valid value:  
  
DAQ-2501      DA\_Group\_A  
DAQ-2502      DA\_Group\_A. DA\_Group\_B and DA\_Group\_AB
- BufId*

The buffer ID (returned from function D2K\_AO\_ContBufferSetup) of the buffer containing the output data. The size of the buffer with id with this parameter must have a length equal with or smaller than the size of the onboard FIFOs.

### DAQ-2502

- DA\_Group\_A

8K samples
- DA\_Group\_B

8K samples
- DA\_Group\_AB

16K samples

### DAQ-2501

- DA\_Group\_A

8K samples

The sequence of the data in the buffer is the same as the sequence of the channels in the specified group. For example:

DA\_Group\_A: channel 0, 1 enabled

DA\_Group\_B: channel 4, 5 enabled, and group loaded is DA\_Group\_AB, then

the data for channel 0 is in Buffer[0], Buffer[4], Buffer[8], ...

the data for channel 1 is in Buffer[1], Buffer[5], Buffer[9], ...

the data for channel 4 is in Buffer[2], Buffer[6], Buffer[10], ...

the data for channel 5 is in Buffer[3], Buffer[7], Buffer[11], ...

The range of valid data is within 0 to 4095.

*UpdateCount*    Count of data loaded to the FIFOs.

### **Return Code**

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel

## D2K\_AO\_Group\_Setup

### Description

Assigns one or more analog output channels to a waveform generation group.

### Supported Cards

2501, 2502

### Syntax

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

```
I16 D2K_AO_Group_Setup (U16 CardNumber, U16
                        group, U16 NumChans, U16 *Chans)
```

### Visual Basic

```
D2K_AO_Group_Setup (ByVal CardNumber As Integer,
                    ByVal group As Integer, ByVal NumChans As
                    Integer, Chans As Integer) As Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.
<i>NumChans</i>	The number of analog output channels in the array Chans. Valid value: <div> <div>DAQ-2501</div> <div>1 through 4</div> </div> <div> <div>DAQ-2502</div> <div>1 through 8</div> </div>
<i>group</i>	The group of analog output channels. The valid value: <div> <div>DAQ-2501</div> <div>DA_Group_A</div> </div> <div> <div>DAQ-2502</div> <div>DA_Group_A, DA_Group_B and DA_Group_AB</div> </div>
<i>Chans</i>	Array of analog output channel numbers. The channel order for update data is the same as the order you set in Chans. <div> <div>DAQ-2501</div> <div>Numbers in Chans must be within 0 and 3.</div> </div> <div> <div>DAQ-2502</div> <div>Numbers in Chans must be: <div> <div>DA_Group_A: 0 to 3</div> <div>DA_Group_B: 4 to 7</div> <div>DA_Group_AB: 0 to 7</div> </div> </div> </div>

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel

## D2K\_AO\_Group\_Update

### Description

Simultaneously writes binary values to the specified group of analog output channels.

### Supported Cards

2501, 2502

### Syntax

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

```
I16 D2K_AO_Group_Update (U16 CardNumber, U16
                        group, I16 *Buffer)
```

### Visual Basic

```
D2K_AO_Group_Update (ByVal CardNumber As Integer,
                    ByVal group As Integer, Buffer As Integer)
                    As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*group* The group of analog output channels. The valid value:

DAQ-2501 DA\_Group\_A

DAQ-2502 DA\_Group\_A, DA\_Group\_B and DA\_Group\_AB

*Buffer* An integer array that contains the updated data. The length (in samples) of Buffer must be equal to or greater than the total number of channels in the specified DA group. The range of value to be written to the analog output channels is 0 to 4095.

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

## D2K\_AO\_Group\_VUpdate

### Description

Accepts voltage values, scales them to the proper binary values and Writes binary values to the specified group of analog output channels simultaneously.

### Supported Cards

2501, 2502

### Syntax

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

```
I16 D2K_AO_Group_VUpdate (U16 CardNumber, U16
                           group, F64 *Voltage)
```

### Visual Basic

```
D2K_AO_Group_VUpdate (ByVal CardNumber As
                       Integer, ByVal group As Integer, Voltage As
                       Double) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*group* The group of analog output channels. The valid value:

DAQ-2501 DA\_Group\_A

DAQ-2502 DA\_Group\_A, DA\_Group\_B and DA\_Group\_AB

*Voltage* A floating-point voltage value array to contain the update data. The length (in samples) of Voltage must be equal to or greater than the total number of channels in the specified DA group. The range of voltages depends on the device, the output polarity, and the voltage reference (external or internal).

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

D2K\_AO\_Group\_WFM\_AsyncCheck

Description

Check the current status of the asynchronous analog output operation of a specified group. This function is only available for the device that uses timer pacer (DAQ2K\_DA\_WRSRC\_Int ) as the D/A R/W Source.

Supported Cards

2501, 2502

Syntax

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

```
I16 D2K_AO_Group_WFM_AsyncCheck (U16 CardNumber,
                                  U16 group, U8 *Stopped, U32 *WriteCnt)
```

Visual Basic

```
D2K_AO_Group_WFM_AsyncCheck (ByVal CardNumber As Integer,
                               ByVal group As Integer, Stopped As Byte,
                               WriteCnt As Long) As Integer
```

Parameters

- CardNumber*    ID of the card performing the operation.
- group*            The group of analog output channels. The valid value:

DAQ-2501    DA\_Group\_A

DAQ-2502    DA\_Group\_A, DA\_Group\_B and DA\_Group\_AB

***Stopped*** Tells whether the asynchronous analog output operation has completed.

DA\_Group\_A and  
DA\_Group\_B

If Stopped = 1, the analog output operation has stopped after the number of D/A conversions indicated in the call that initiated the asynchronous analog output operation is completed or an error has occurred. If Stopped = 0, the operation is not yet complete.

DA\_Group\_AB

If Stopped = 3, the analog output operation has stopped for both DA group A and group B.

Bit0: Asynchronous  
analog output  
operation of group A.

If Stopped = 1, the analog output operation has stopped for DA group A.

Bit1: Asynchronous  
analog output  
operation of group B.

If Stopped = 2, the analog output operation has stopped for DA group B.  
If Stopped = 0, the operation is not yet complete for both DA group A and group B.

***WriteCnt*** The number of analog output data written at the time D2K\_AO\_Group\_WFM\_AsyncCheck () function is called.

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport



D2K\_AO\_Group\_WFM\_AsyncClear

Description

The software terminates the asynchronous analog output operation of a specified group.

Supported Cards

2501, 2502

Syntax

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

```
I16 D2K_AO_Group_WFM_AsyncClear (U16 CardNumber ,
                                U16 group, U32 *WriteCnt, U16 stop_mode)
```

Visual Basic

```
D2K_AO_Group_WFM_AsyncClear (ByVal CardNumber As Integer, ByVal group As Integer, WriteCnt As Long, ByVal stop_mode As Integer) As Integer
```

Parameters

*CardNumber* ID of the card performing the asynchronous operation.

*group* The group of analog output channels. Valid value:  
DAQ-2501 DA\_Group\_A  
DAQ-2502 DA\_Group\_A, DA\_Group\_B and DA\_Group\_AB

*WriteCnt* Number of analog output data written at the time D2K\_AO\_Group\_WFM\_AsyncClear () is called.

*stop\_mode* Selected DA transfer termination mode. Valid values:

DAQ2K_DA_TerminateImmediate	Software immediately terminates the continuous DA operation.
DAQ2K_DA_TerminateUC	Software terminates the continuous DA operation on next counter terminal count update.
DAQ2K_DA_TerminateIC	Software terminates the continuous DA operation on iteration count

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport

# D2K\_AO\_Group\_WFM\_Start

## Description

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

## Supported Cards

2501, 2502

## Syntax

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

```
I16 D2K_AO_Group_WFM_Start (U16 CardNumber, U16
    group, U16 FstBufIdOrNotUsed, U16 sndBufId,
    U32 UpdateCount, U32 Iterations, U32 CHUI,
    U16 definite)
```

## Visual Basic

```
D2K_AO_Group_WFM_Start (ByVal CardNumber As
    Integer, ByVal group As Integer, ByVal
    FstBufIdOrNotUsed As Integer, ByVal sndBufId
    As Integer, ByVal UpdateCount As Long, ByVal
    Iterations As Long, ByVal CHUI As Long,
    ByVal definite As Integer) As Integer
```

## Parameters

- CardNumber* ID of the card performing the operation.
- group* The group of analog output channels. Valid value:

DAQ-2501 DA\_Group\_A

DAQ-2502 DA\_Group\_A, DA\_Group\_B and DA\_Group\_AB

### *FstBufIdOrNotUsed*

If the data have been loaded by the function D2K\_AO\_Group\_FIFOLoad, the value of fstBufIdOrNotUsed must be BufferNotUsed.

If the value of the parameter fstBufIdOrNotUsed is not BufferNotUsed, or the data has not been loaded, the value of fstBufIdOrNotUsed is the buffer ID (returned from function D2K\_AO\_ContBufferSetup) of the first buffer containing the output data.

## DAQ-2501

The data are transferred to the DA FIFOs from the buffer with `fstBufIdOrNotUsed` ID through DMA operation. The sequence of the data in the buffer is the same as the sequence of the channels in the group.

For example:

DA_Group_A	Channel 0, 1, 2 enabled
	The data for channel 0 is in Buffer[0], Buffer[4], Buffer[8], and so on.
	The data for channel 1 is in Buffer[1], Buffer[5], Buffer[9], and so on.
	The data for channel 2 is in Buffer[2], Buffer[6], Buffer[10], and so on.

## DAQ-2502

The data are transferred to the DA FIFOs from the buffer with `fstBufIdOrNotUsed` ID through 32-bit DMA operation. The upper 16 bits of data are for the FIFO of Group B and the lower 16 bits of data is for the FIFO of Group A.

If the group is `DA_Group_A` or `DA_Group_B`, the 16-bit buffer must contain two times the total update samples because of 32-bit DMA data transfer. The data points for the specified group are in the even elements of the buffer.

DA_Group_A	Channel 0, 1 enabled
	The data for channel 0 is in Buffer[0], Buffer[4], Buffer[8], and so on.
	The data for channel 1 is in Buffer[2], Buffer[6], Buffer[10], and so on.
DA_Group_B	Channel 4, 5 enabled
	The data for channel 4 is in Buffer[1], Buffer[5], Buffer[9], and so on.
	The data for channel 5 is in Buffer[3], Buffer[7], Buffer[11], and so on.

If the group is `DA_Group_AB`, the buffer must contain the same sample counts for group A and group B.

The data for each group is in interleaved sequence.  
For example:

DA\_Group\_A: Channel 0, 1 enabled

DA\_Group\_B: Channel 4, 5 enabled, and group loaded is DA\_Group\_AB, then:

The data for channel 0 is in Buffer[0], Buffer[4], Buffer[8], and so on.

The data for channel 4 is in Buffer[1], Buffer[5], Buffer[9], and so on.

The data for channel 1 is in Buffer[2], Buffer[6], Buffer[10], and so on.

The data for channel 5 is in Buffer[3], Buffer[7], Buffer[11], and so on.

*sndBuflD* The buffer ID (returned from function D2K\_AO\_ContBufferSetup) of the second buffer containing the output data. This parameter is only available for double buffer mode of waveform generation.

*UpdateCount* If data has been loaded by the function D2K\_AO\_Group\_FIFOLoad, the parameter of UpdateCount has no function. If the data is transferred through DMA operation and double-buffered mode is disabled, the value of UpdateCount is the total update count for each channel that performs the operation. For double-buffered acquisition, UpdateCount is the size (in samples) allocated for each channel in the circular buffer and its value must be a multiple of 2.

*Iterations* The number of times the data in the buffer outputs to the port. From D2K-DASK version 1.1, a value of zero is not allowed.

<i>CHUI</i>	Length of the Channel Update interval (the counter value between the initiation of each update sequence).  When the device has an external time base, the range of valid value is 8 to 16777215. If the time base is internal, the range of valid value is 40 to 16777215.				
<i>definite</i>	Waveform generation proceeds in definite or indefinite manner. If double-buffered mode is enabled, this parameter is ignored.  <table><tr><td>0</td><td>Indefinite</td></tr><tr><td>1</td><td>Definite</td></tr></table>	0	Indefinite	1	Definite
0	Indefinite				
1	Definite				

---

**NOTES**      If FIFO mode of waveform generation is enabled, the double-buffered waveform generation is not allowed.

                 If the group DA\_Group\_AB is specified, the iterations and scan rate are the same for both groups.

---

### Return Code

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

## D2K\_AO\_Group\_WFM\_StopConfig

### Description

Informs the D2K-DASK library of the stop source and the stop mode for the asynchronous analog output operation of a specified group.

### Supported Cards

2501, 2502

### Syntax

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

```
I16 D2K_AO_Group_WFM_StopConfig (U16 CardNumber,
                                  U16 group, U16 stopSrc, U16 stopMode)
```

### Visual Basic

```
D2K_AO_Group_WFM_StopConfig (ByVal CardNumber As
                              Integer, ByVal group As Integer, ByVal
                              stopSrc As Integer, ByVal stopMode As
                              Integer) As Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the asynchronous operation.
<i>group</i>	The group of analog output channels. Valid value:  DAQ-2501     DA_Group_A DAQ-2502     DA_Group_A, DA_Group_B and DA_Group_AB
<i>stopSrc</i>	Selected DA transfer termination source:  DAQ2K_DA_STOPSRC_SOFT     Software terminates the continuous DA operation. DAQ2K_DA_STOPSRC_AFIO     Terminates the continuous DA operation from the external signal of AFIO. DAQ2K_DA_STOPSRC_ATrig     Terminates the continuous DA operation from the external signal of the analog trigger. DAQ2K_DA_STOPSRC_AF11     Terminates the continuous DA operation from the external signal of AF11.

*stop\_mode*      The DA transfer termination mode selected. The valid values are:

DAQ2K_DA_TerminateImmediate	Terminates the continuous DA operation immediately.
DAQ2K_DA_TerminateUC	Terminates the continuous DA operation on the next counter terminal count update.
DAQ2K_DA_TerminateIC	Terminates the continuous DA operation on iteration count.

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel  
ErrorInvalidSampleRate  
ErrorTransferCountTooLarge  
ErrorContIoNotAllowed



## D2K\_AO\_InitialMemoryAllocated

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AO_InitialMemoryAllocated (U16  
    CardNumber, U32 *MemSize)
```

### Visual Basic

```
D2K_AO_InitialMemoryAllocated (ByVal CardNumber  
    As Integer, MemSize As Long) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*MemSize* The available memory size for continuous AO in device driver of this card. The unit is KB (1024 bytes).

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered
```

## D2K\_AO\_PostTrig\_Config

### Description

Informs the D2K-DASK library of the update clock source and the trigger properties of the device performing post triggered waveform generation operation.

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AO_PostTrig_Config (U16 CardNumber, U16  
    ClkSrc, U16 TrigSrcCtrl, U16 DLY2Ctrl, U16  
    DLY2Cnt, U16 ReTrgEn, U16 ReTrgCnt, BOOLEAN  
    AutoResetBuf)
```

### Visual Basic

```
D2K_AO_PostTrig_Config (ByVal CardNumber As  
    Integer, ByVal ClkSrc As Integer, ByVal  
    TrigSrcCtrl As Integer, ByVal DLY2Ctrl As  
    Integer, ByVal DLY2Cnt As Integer, ByVal  
    ReTrgEn As Integer, ByVal ReTrgCnt As  
    Integer, ByVal AutoResetBuf As Byte) As  
    Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*ClkSrc* D/A update clock source settings. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H.

#### D/A R/W Source Selection

DAQ2K_DA_WRSRC_Int	Internal timer (default)
DAQ2K_DA_WRSRC_AFI0	From AFI0 pin
DAQ2K_DA_WRSRC_SSI	From SSI source

The following constant groups are only available for DAQ-2501 and DAQ-2502

### DA Group Selection

DAQ_Group_A	DA group A
DAQ_Group_B	DA group B
DAQ_Group_AB	DA group A and group B

### D/A Break Delay Counter Source Selection

DAQ2K_DA_BDSRC_Int	Internal timer (Default)
DAQ2K_DA_BDSRC_AFI0	From AFI0 pin
DAQ2K_DA_BDSRC_GPTC0	From GPTC0_OUT pin
DAQ2K_DA_BDSRC_GPTC1	From GPTC1_OUT pin

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

*TrigSrcCtrl*

D/A trigger control setting. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H. There are two groups of constants:

### Trigger Source Selection

DAQ2K_DA_TRGSRC_SOFT	Software (Default)
DAQ2K_DA_TRGSRC_ANA	From analog trigger pin
DAQ2K_DA_TRGSRC_ExtD	From external digital trigger pin
DAQ2K_DA_TRSRC_SSI	From SSI source

### External Digital Trigger Polarity

DAQ2K_DA_TrgPositive	Trigger positive edge active (Default)
DAQ2K_DA_TrgNegative	Trigger negative edge active

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

*DLY2Ctrl*

The setting for D/A Trigger control. This argument is an integer expression formed from one or more of the

manifest constants defined in D2KDASK.H. There are two groups of constants:

### Delay2 (Break delay) Mode Enable

DAQ2K_DA_DLY2En	Delay2/Break delay (delay between two consecutive waveform generations) in an acquisition is enabled.
-----------------	---

### Delay2 Source Selection

DAQ2K_DA_Dly2InUI	Delay in samples (not valid for DAQ-2501/2502)
DAQ2K_DA_Dly2InTimebase	Delay in time base (default)

When two or more constants are used to form the DLY2Ctrl argument, the constants are combined with the bitwise-OR operator(|).

*DLY2Cnt* The counter value of DLY2 Counter (the Delay between two consecutive waveform generations). The valid value range is 0 through 65535.

*ReTrgEn*

0	Re-trigger in an acquisition is disabled. (Default)
1	Re-trigger in an acquisition is enabled.

*ReTrgCnt* The accepted trigger times in an acquisition. The range of valid value is 0 to 65535.

*AutoResetBuf* For DAQ-2502/2501, this parameter must be set to FALSE.

FALSE	The DA buffers set by the D2K_AO_ContBufferSetup function are retained and D2K_AO_ContBufferReset must be called to reset the buffer.
TRUE	The AI buffers set by the D2K_AO_ContBufferSetup function are reset automatically by the driver when AI operation is finished.

### Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport

## D2K\_AO\_SimuWriteChannel

### Description

Simultaneously writes binary values to the specified analog output channels. This function is available only for simultaneous DA cards.

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214

### Syntax

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

```
I16 D2K_AO_SimuWriteChannel (U16 CardNumber, U16
                             NumChans, U16 *Buffer)
```

### Visual Basic

```
D2K_AO_SimuWriteChannel (ByVal CardNumber As
                          Integer, ByVal NumChans As Integer, Buffer
                          As Integer) As Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.																
<i>NumChans</i>	Number of analog output channels. Valid value: <table><tr><td>DAQ-2010</td><td>1 to 2</td></tr><tr><td>DAQ-2005</td><td>1 to 2</td></tr><tr><td>DAQ-2006</td><td>1 to 2</td></tr><tr><td>DAQ-2016</td><td>1 to 2</td></tr><tr><td>DAQ-2204</td><td>1 to 2</td></tr><tr><td>DAQ-2205</td><td>1 to 2</td></tr><tr><td>DAQ-2206</td><td>1 to 2</td></tr><tr><td>DAQ-2214</td><td>1 to 2</td></tr></table>	DAQ-2010	1 to 2	DAQ-2005	1 to 2	DAQ-2006	1 to 2	DAQ-2016	1 to 2	DAQ-2204	1 to 2	DAQ-2205	1 to 2	DAQ-2206	1 to 2	DAQ-2214	1 to 2
DAQ-2010	1 to 2																
DAQ-2005	1 to 2																
DAQ-2006	1 to 2																
DAQ-2016	1 to 2																
DAQ-2204	1 to 2																
DAQ-2205	1 to 2																
DAQ-2206	1 to 2																
DAQ-2214	1 to 2																
<i>Buffer</i>	An integer array to contain the update data. The length (in samples) of Buffer must be equal to or greater the value of parameter numChans. The range of value to be written to the analog output channels is 0 to 4095.																

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel

## D2K\_AO\_VoltScale

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_AO_VoltScale (U16 CardNumber, U16
    Channel, F64 Voltage, I16 *binValue)
```

### Visual Basic

```
D2K_AO_VoltScale (ByVal CardNumber As Integer,
    ByVal Channel As Integer, ByVal Voltage As
    Double, binValue As Integer) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*Channel* Analog output channel number.

DAQ-2010, DAQ-2005, DAQ-2006, DAQ-2016, 0 or 1  
DAQ-2205, DAQ-2206, DAQ-2204, DAQ-2214

DAQ-2501 0 to 3

DAQ-2502 0 to 7

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

*binValue* Converted binary value returned.

### Return Code

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

## D2K\_AO\_VWriteChannel

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214

### Syntax

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

```
I16 D2K_AO_VWriteChannel (U16 CardNumber, U16  
Channel, F64 Voltage)
```

### Visual Basic

```
D2K_AO_VWriteChannel (ByVal CardNumber As  
Integer, ByVal Channel As Integer, ByVal  
Voltage As Double) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*Channel* Analog output channel number.

DAQ-2010	0 or 1
DAQ-2005	0 or 1
DAQ-2006	0 or 1
DAQ-2016	0 or 1
DAQ-2204	0 or 1
DAQ-2205	0 or 1
DAQ-2206	0 or 1
DAQ-2214	0 or 1

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



## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel  
ErrorDaVoltageOutOfRange

## D2K\_AO\_WriteChannel

### Description

Writes a binary value to the specified analog output channel.

### Supported Cards

2010, 2005, 2006, 2016, 2204, 2205, 2206, 2214

### Syntax

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

```
I16 D2K_AO_WriteChannel (U16 CardNumber, U16  
Channel, I16 Value)
```

### Visual Basic

```
D2K_AO_WriteChannel (ByVal CardNumber As Integer,  
ByVal Channel As Integer, ByVal Value As  
Integer) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*Channel* Analog output channel number. Range is 0 or 1.

*Value* The value to be written to the analog output channel.  
Range is 0 to 4095.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```

## D2K\_DB\_Auto\_Calibration\_ALL

### Description

Calibrates the device. When this function is called, the device goes into a self-calibration cycle. The function does not return until the self-calibration is completed.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_DB_Auto_Calibration_ALL(U16 CardNumber)
```

### Visual Basic

```
D2K_DB_Auto_Calibration_ALL (ByVal CardNumber As  
Integer) As Integer
```

### Parameter

*CardNumber* ID of the card performing the operation.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```

## D2K\_DI\_ReadLine

### Description

Reads the digital logic state of the specified digital line in the specified port.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_DI_ReadLine (U16 CardNumber, U16 Port,  
                    U16 Line, U16 *State)
```

### Visual Basic

```
D2K_DI_ReadLine (ByVal CardNumber As Integer,  
                ByVal Port As Integer, ByVal Line As  
                Integer, State As Integer) As Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.
<i>Port</i>	Digital input port number. Valid values (not valid for DAQ-2020/2022): Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH. For DAQ-2020/2022: 0.
<i>Line</i>	The digital line to be read. Valid value is 0 to 7. DAQ-2020/2022: 0 to 3.
<i>State</i>	Returns the digital logic state (0 or 1) of the specified line.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```

## D2K\_DI\_ReadPort

### Description

Reads the digital data from the specified digital input port.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_DI_ReadPort (I16 CardNumber, U16 Port,  
                    U32 *Value)
```

### Visual Basic

```
D2K_DI_ReadPort (ByVal CardNumber As Integer,  
                ByVal Port As Integer, Value As Long) As  
                Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*Port* Digital input port number. Valid values (not valid for DAQ-2020/2022): Channel\_P1A, Channel\_P1B, Channel\_P1C, Channel\_P1CL, Channel\_P1CH. For DAQ-2020/2022: 0.

*Value* Returns the digital data read from the specified port. The returned value is an 8-bit data.

### Return Code

```
NoError  
CardNotRegistered  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport
```

## D2K\_DIO\_LineConfig

### Description

Informs D2K-DASK library of the line selected and the direction (Input or output) setting of the selected line.

### Cards Support

2020, 2022

### Syntax

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

```
I16 D2K_DIO_LineConfig (U16 CardNumber, U16 Port,  
                        U32 Line, U16 Direction)
```

### Visual Basic

```
D2K_DIO_LineConfig (ByVal CardNumber As Integer,  
                    ByVal Port As Integer, ByVal Line As Long,  
                    ByVal Direction As Integer) As Integer
```

### Parameters

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

**Port** The port selected. The valid value: DAQ-2020/2022: 0

**Line** The line selected. The valid value: DAQ-2020/2022: 0 through 3

**Direction** The line direction of PIO port. The valid value: INPUT\_LINE, OUTPUT\_LINE

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```

## D2K\_DIO\_LinesConfig

### Description

Informs D2K-DASK library of the entire lines of the port selected and the direction (Input or output) setting of the entire lines of the selected port.

### Cards Support

2020, 2022

### Syntax

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

```
I16 D2K_DIO_LinesConfig (U16 wCardNumber, U16  
wPort, U32 wLinesdirmap)
```

### Visual Basic

```
D2K_DIO_LinesConfig (ByVal CardNumber As Integer,  
ByVal Port As Integer, ByVal Linesdirmap As  
Long) As Integer
```

### Parameters

CardNumber	The card id of the card that want to perform this operation.
Port	The port selected. The valid value: DAQ-2020/2022: 0
Linesdirmap	The port direction of PIO port. The valid value: DAQ-2020/2022: Each bit of the value of Linesdirmap controls one line of the port selected. The '1' value of the bit value set the corresponding line to output, and the '0' value of the bit value set the corresponding line to input. The valid values for Linesdirmap: 0 through 15 (0xf)

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport
```

## D2K\_DIO\_PortConfig

### Description

Informs D2K-DASK library of the port selected and the direction (Input or output) setting of the selected port.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_DIO_PortConfig (U16 CardNumber, U16 Port,  
                        U16 Direction)
```

### Visual Basic

```
D2K_DIO_PortConfig (ByVal CardNumber As Integer,  
                    ByVal Port As Integer, ByVal Direction As  
                    Integer) As Integer
```

### Parameters

*CardNumber*     ID of the card performing the operation.

*Port*            Digital input port number. Valid values (not valid for DAQ-2020/2022): Channel\_P1A, Channel\_P1B, Channel\_P1C, Channel\_P1CL, Channel\_P1CH. For DAQ-2020/2022: 0.

*Direction*      Direction of PIO port. Valid value: INPUT\_PORT or OUTPUT\_PORT

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```



## D2K\_DO\_ReadLine

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_DO_ReadLine (U16 CardNumber, U16 Port,  
                    U16 Line, U16 *State)
```

### Visual Basic

```
D2K_DO_ReadLine (ByVal CardNumber As Integer,  
                ByVal Port As Integer, ByVal Line As  
                Integer, State As Integer) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*Port* Digital input port number. Valid values (not valid for DAQ-2020/2022): Channel\_P1A, Channel\_P1B, Channel\_P1C, Channel\_P1CL, Channel\_P1CH. For DAQ-2020/2022: 0.

*Line* The digital line to be read. Valid value is 0 to 7.

*State* Returns the digital logic state (0 or 1) of the specified line.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```

## D2K\_DO\_ReadPort

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_DO_ReadPort (U16 CardNumber, U16 Port,  
                    U32 *Value)
```

### Visual Basic

```
D2K_DO_ReadPort (ByVal CardNumber As Integer,  
                 ByVal Port As Integer, Value As Long) As  
Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*Port* Digital input port number. Valid values (not valid for DAQ-2020/2022): Channel\_P1A, Channel\_P1B, Channel\_P1C, Channel\_P1CL, Channel\_P1CH. For DAQ-2020/2022: 0.

*Value* Returns the digital data read from the specified output port. Refer to the D2K\_DI\_ReadPort function.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```

## D2K\_DO\_WriteLine

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_DO_WriteLine (U16 CardNumber, U16 Port,
                      U16 Line, U16 State)
```

### Visual Basic

```
D2K_DO_WriteLine(ByVal CardNumber As Integer,
                  ByVal Port As Integer, ByVal Line As
                  Integer, ByVal State As Integer) As Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.
<i>Port</i>	Digital input port number. Valid values (not valid for DAQ-2020/2022): Channel_P1A, Channel_P1B, Channel_P1C, Channel_P1CL, Channel_P1CH. For DAQ-2020/2022: 0.
<i>Line</i>	The digital line to write to. Refer to the D2K_DI_ReadLine function.
<i>State</i>	The new digital logic state (0 or 1).

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

## D2K\_DO\_WritePort

### Description

Writes digital data to the specified digital output port.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_DO_WritePort (U16 CardNumber, U16 Port,  
                      U32 Value)
```

### Visual Basic

```
D2K_DO_WritePort (ByVal CardNumber As Integer,  
                  ByVal Port As Integer, ByVal Value As Long)  
As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*Port* Digital output port number. Valid values (not valid for DAQ-2020/2022): Channel\_P1A, Channel\_P1B, Channel\_P1C, Channel\_P1CL, Channel\_P1CH. For DAQ-2020/2022: 0.

*Value* Digital data written to the specified port. The value is an 8-bit data.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```

## D2K\_EEPROM\_CAL\_Constant\_Update

### Description

Saves the new calibration constants to the specified bank of the EEPROM.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
U16 D2K_EEPROM_CAL_Constant_Update(U16  
    CardNumber, U16 bank)
```

### Visual Basic

```
D2K_EEPROM_CAL_Constant_Update (ByVal wCardNumber  
    As Integer, ByVal bank As Integer) As  
    Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*bank* Storage location on the EEPROM. The range of valid value of the bank is 1 to 2 for DAQ-2020/2022 and 0 to 3 for others.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```

## D2K\_GCTR\_Control

### Description

Controls the selected counter/timer using the software.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_GCTR_Control (U16 CardNumber, U16 GCtr,  
U16 ParamID, U16 Value)
```

### Visual Basic

```
D2K_GCTR_Control (ByVal CardNumber As Integer,  
ByVal GCtr As Integer, ByVal ParamID As  
Integer, ByVal Value As Integer) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*GCtr* Counter number. Range is 0 to 1.

*ParamID* The internal parameter ID is the general-purpose timer/counter you want to control. The valid control include:

GPTC_IntGATE	Internal gate
GPTC_IntUpDnCTR	Internal updown counter
GPTC_IntENABLE	Start or stop counter operation

*Value* The value of the controlled item specified by ParamID. Valid value for is 0 or 1.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
InvalidCounter
```

## D2K\_GCTR\_Read

### Description

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

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_GCTR_Read (U16 CardNumber, U16 GCtr, U32  
    *Value)
```

### Visual Basic

```
D2K_GCTR_Read (ByVal CardNumber As Integer, ByVal  
    GCtr As Integer, Value As Long) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*GCtr* Counter number. Range is 0 to 1.

*Value* Returns the counter value of the specified general-purpose timer/counter. For the devices except for DQ-2020/2022, the range is 0 to 65535. For DAQ-2020/2022, the range is 0 to 4294967295.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
InvalidCounter
```

## D2K\_GCTR\_Reset

### Description

Halts the specified general-purpose timer/counter operation and reloads the initial value of the timer/counter.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_GCTR_Reset (U16 CardNumber, U16 GCtr)
```

### Visual Basic

```
D2K_GCTR_Reset (ByVal CardNumber As Integer,  
                ByVal GCtr As Integer) As Integer
```

### Parameters

*CardNumber*      ID of the card performing the operation.

*GCtr*             Counter number. Range is 0 to 1.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
InvalidCounter
```



## D2K\_GCTR\_Setup D2K\_GCTR\_SetupEx

### Description

Controls the operation of the selected counter/timer.

The D2K\_GCTR\_SetupEx function is used for the devices (e.g. DAQ-2020/2022) which counters width are 32-bit long.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_GCTR_Setup (U16 CardNumber, U16 GCtr,
    U16 Mode, U8 SrcCtrl, U8 PolCtrl, U16
    LReg1_Val, U16 LReg2_Val)
I16 D2K_GCTR_SetupEx (U16 CardNumber, U16 GCtr,
    U16 Mode, U8 SrcCtrl, U8 PolCtrl, U32
    LReg1_Val, U32 LReg2_Val)
```

### Visual Basic

```
D2K_GCTR_Setup (ByVal CardNumber As Integer,
    ByVal GCtr As Integer, ByVal Mode As
    Integer, ByVal SrcCtrl As Byte, ByVal
    PolCtrl As Byte, ByVal LReg1_Val As Integer,
    ByVal LReg2_Val As Integer) As Integer
D2K_GCTR_SetupEx (ByVal CardNumber As Integer,
    ByVal GCtr As Integer, ByVal Mode As
    Integer, ByVal SrcCtrl As Byte, ByVal
    PolCtrl As Byte, ByVal LReg1_Val As Long,
    ByVal LReg2_Val As Long) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*GCtr* Counter number. Range is 0 to 1.

*Mode* Timer/Counter mode. Valid modes:

- ▷ SimpleGatedEventCNT
- ▷ SinglePeriodMSR
- ▷ SinglePulseWidthMSR
- ▷ SingleGatedPulseGen
- ▷ SingleTrigPulseGen
- ▷ RetrigSinglePulseGen
- ▷ SingleTrigContPulseGen
- ▷ ContGatedPulseGen

Refer to the device manual for the mode description.

*SrcCtrl* General-purpose timer/counter source control setting. This argument is an integer expression formed from one or more of the manifest constants defined in D2KDASK.H. There are three groups of constants:

#### **Timer/Counter Source**

- |                 |   |
|-----------------|---|
| GPTC_CLKSRC_INT | Internal time base                                  |
| GPTC_CLKSRC_EXT | External time base from GPTC0_SRC or GPTC1_SRC pin. |

#### **Timer/Counter Gate Source**

- |                  |   |
|------------------|---|
| GPTC_GATESRC_INT | Gate is controlled by software.                     |
| GPTC_GATESRC_EXT | Gate is controlled by GPTC0_GATE or GPTC1_GATE pin. |

#### **Timer/Counter UpDown Source**

- |                     |   |
|---------------------|---|
| GPTC_UPDOWN_SEL_INT | Up/Down controlled by software.                         |
| GPTC_UPDOWN_SEL_EXT | Up/Down controlled by GPTC0_UPDOWN or GPTC1_UPDOWN pin. |

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

*PolCtrl* Polarity settings for general-purpose timer/counter. This argument is an integer expression formed from

one or more of the manifest constants defined in D2KDASK.H. There are four groups of constants:

#### Timer/Counter Gate Polarity

GPTC_GATE_LACTIVE	Low active
GPTC_GATE_HACTIVE	High active

#### Timer/Counter UpDown Polarity

GPTC_UPDOWN_LACTIVE	Low active
GPTC_UPDOWN_HACTIVE	High active

#### Timer/Counter ClockEn Polarity

GPTC_CLKEN_LACTIVE	Low active
GPTC_CLKEN_HACTIVE	High active

#### Timer/Counter Output Polarity

GPTC_OUTPUT_LACTIVE	Low active
GPTC_OUTPUT_HACTIVE	High active

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

<i>LReg1_Val</i>	Counter value of the timer/counter load register 1. The meaning for the value depends on the mode the timer /counter is performing. For mode 1 to mode 3, the value of LReg1_Val is the initial count of the GPTC. For mode 4 to mode 8 (or pulse generation modes), the value of LReg1_Val is configured as the pulse delay. For DAQ-2020/2022, LReg1_Val is configured for both the pulse delay and pulse width.
------------------	--

<i>LReg2_Val</i>	Counter value of the timer/counter load register 2. For mode 1 to mode 3, the value of LReg2_Val is not used. For mode 4 to mode 8 (pulse generation modes), the value of LReg2_Val is configured as the pulse width. For DAQ-2020/2022, the value is ignored.
------------------	--

#### Return Code

NoError
ErrorInvalidCardNumber

D2K-DASK  
D2K\_GCTR\_Setup D2K\_GCTR\_SetupEx

ErrorCardNotRegistered  
ErrorFuncNotSupport  
InvalidCounter

## D2K\_GCTR\_Status

### Description

Reads the latched GPTC status of the general-purpose counter from the GPTC status register.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_GCTR_Status (U16 CardNumber, U16 GCtr,
                    U16 *Value)
```

### Visual Basic

```
D2K_GCTR_Status (ByVal CardNumber As Integer,
                 ByVal GCtr As Integer, Value As Integer) As
Integer
```

### Parameters

- CardNumber*      ID of the card performing the operation.
- GCtr*             Counter number. Range is 0 to 1.
- Value*            Returns the latched GPTC status of the specified general-purpose timer/counter from the GPTC status register. The format of Value is as follows:  
2010, 2005, 2006, 2016, 2204, 2205, 2206, 2213, 2214, 2501, 2502:  
  
         bit 0            The former latched status of enable.  
         bit 1            The former latched status of gate.  
         bit 2            The former latched status of up/down.  
         bit 3            The former latched status of output.  
         bit 4            The former latched status of clk.  
         bit 5            The former latched status of interrupt.  
         bit 6 to 15      Not used  
  
2020, 2022:  
  
         bit 2            Counter operation is in progress.  
         bit 3            Counter operation is done.

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
InvalidCounter

## D2K\_Get\_Default\_Load\_Area

### Description

Gets the number of bank where the calibration data are loaded by default when D2K\_Register\_Card is called.

### Cards Support

2020, 2022

### Syntax

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

```
I16 D2K_Get_Default_Load_Area(U16 CardNumber)
```

Visual Basic

```
D2K_Get_Default_Load_Area (ByVal CardNumber As  
Integer) As Integer
```

### Parameter

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

### Return Code

The default bank number

## D2K\_GetPXISlotGeographAddr

### Description

Get the physical slot number of the slot where the specified PXI module installed in.

### Cards Support

PXI-2020, PXI-2022

### Syntax

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

```
I16 D2K_GetPXISlotGeographAddr (U16 wCardNumber,  
                                U8* geo_addr)
```

### Visual Basic

```
D2K_GetPXISlotGeographAddr (ByVal CardNumber As  
                             Integer, geo_addr As Byte) As Integer
```

### Parameter

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

**geo\_addr**: The physical slot number of the slot the module plugged in.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```



## D2K\_Load\_CAL\_Data

### Description

Loads the calibration constants from the specified EEPROM bank.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_Load_CAL_Data (U16 CardNumber, U16 bank)
```

Visual Basic

```
D2K_Load_CAL_Data (ByVal CardNumber As Integer,  
    ByVal bank As Integer) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*bank* Storage bank on the EEPROM. The range of valid value is 0 to 2 for DAQ-2020/2022 is and 0 to 3 for others.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```

## D2K\_Register\_Card

### Description

Initializes the hardware and software states of the card, then returns a numeric card ID that corresponds to the initialized card. You must call this function before calling any other D2K-DASK library functions. This function initializes the card and variables internal to D2K-DASK library. Since DAQ-2000 come in plug-and-play design, the base address (pass-through address) and IRQ level are automatically assigned by the system BIOS.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_Register_Card (U16 CardType, U16  
card_num)
```

Visual Basic

```
D2K_Register_Card (ByVal CardType As Integer,  
ByVal card_num As Integer) As Integer
```

### Parameters

**CardType** Type of card to be initialized. ADLINK periodically upgrades the D2K-DASK to support new data acquisition cards. Refer to the card's release notes if D2K-DASK supports it. The table below shows the constants defined in D2KDASK.H which represent cards that D2K-DASK currently supports:

DAQ_2010	DAQ-2010
DAQ_2005	DAQ-2005
DAQ_2006	DAQ-2006
DAQ_2016	DAQ-2016
DAQ_2204	DAQ-2204
DAQ_2205	DAQ-2205
DAQ_2206	DAQ-2206
DAQ_2208	DAQ-2208

DAQ_2213	DAQ-2213
DAQ_2214	DAQ-2214
DAQ_2501	DAQ-2501
DAQ_2502	DAQ-2502
DAQ-2020	DAQ-2020
DAQ-2022	DAQ-2022

*card\_num*      The sequence number of cards with the same card type (as defined in argument CardType) plugged in the PCI slot. The card sequence number is based on the PCI slot sequence in the motherboard. The first card in the first slot is assigned card\_num=0. For example, if there are two DAQ-2010 cards installed, the first card in the first slot should be registered with card\_num=0 while the other one is registered with card\_num=1.

### Return Code

This function returns a numeric card ID for the initialized card. The range of card ID is between 0 and 31. If any error occurred, it will return a negative error code from the list of codes below:

```
ErrorTooManyCardRegistered
ErrorUnknownCardType
ErrorOpenDriverFailed
ErrorOpenEventFailed
```

## D2K\_Register\_Card\_By\_PXISlot\_GA

### Description

Initializes the hardware and software states of a D2k\_DASK device with the geographic address of the pxi slot where it is plugged in, and then returns a numeric card ID that corresponds to the card initialized. *D2K\_Register\_Card* or *D2K\_Register\_Card\_By\_PXISlot\_GA* (only for PXI module) must be called before any other D2K-DASK library functions can be called for that card. The function initializes the card and variables internal to D2K-DASK library. Because D2K-DASK devices meet the plug-and-play design, the base address (pass-through address) and IRQ level are assigned by system BIOS directly.

### Cards Support

PXI-2020, PXI-2022

### Syntax

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

```
I16 D2K_Register_Card_By_PXISlot_GA (U16  
CardType, U16 geo_addr)
```

### Visual Basic

```
D2K_Register_Card_By_PXISlot_GA (ByVal cardType  
As Integer, ByVal geo_addr As Integer) As  
Integer
```

### Parameter

*CardType* The type of card to be initialized. ADLink will periodically upgrades D2K-DASK to add support for new DAQ-2000 data acquisition cards. Please refer to Release Notes for the card types that the current release of D2K-DASK actually supports. Following are the constants defined in D2KDASK.H that represent the DAQ-2000 devices that D2K-DASK supports currently or in the near future:

```
DAQ_2020 (for DAQ-2020)  
DAQ_2022 (for DAQ-2022)
```

*geo\_addr* The geographic address of the card plugged in the PXI slot.

## 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  
ErrorOpenDriverFailed  
ErrorOpenEventFailed
```

## D2K\_Release\_Card

### Description

A maximum of 32 cards may be registered simultaneously. This function tells the D2K-DASK library that the registered card is not currently used and may be released to make room for new card to register. At the end of the program, you need to call this function to release all registered cards.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_Release_Card (U16 CardNumber)
```

Visual Basic

```
D2K_Release_Card (ByVal CardNumber As Integer) As  
Integer
```

### Parameter

*CardNumber* ID of the card for release.

### Return Code

NoError

## D2K\_Route\_Signal

### Description

This function is used to route an internal signal to the AFIn, PXI\_STAR line or the PXI trigger bus line, or to enable clock sharing through the PXI trigger bus line or the PXI\_STAR line.

### Cards Support

PXI-2020, PXI-2022

### Syntax

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

```
I16 D2K_Route_Signal (U16 wCardNumber, U16
                    signal, U16 polarity, U16 Line, U16 dir)
```

### Visual Basic

```
D2K_Route_Signal (ByVal CardNumber As Integer,
                  ByVal signal As Integer, ByVal polarity As
                  Integer, ByVal Line As Integer, ByVal dir As
                  Integer) As Integer
```

### Parameter

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

signal : The signal is routed to the specified line.  
The valid signal codes are as follows:

SSI_TIME	SSI_TIMEBASE output
SSI_CONV	SSI_ADCONV output
SSI_ADTRIG	SSI_ADTRIG output
SSI_WR	SSI_DAUpdate output
SSI_ADSTART	SSI_ADSTART output
SSI_DATRIG	SSI_DATRIG output
SMB_CLK_IN	SMB_CLK input

Line : The trigger line to drive.  
The valid lines are as follows:  
The following lines are only valid for PXI module:

PXI_TRIG_n	PXI trigger bus lines (n is line number and the value is 0 through 7)
------------	---

PXI_START_TRIG	PXI_START line
PXI_CLK	PXI clock

The following line is valid for both PXI and PCI modules:

TRG_IO	TRG_IO pin
AFI0	AFI0 pin (DAQ-2020/2022 only)
AFI1	AFI1 pin (DAQ-2020/2022 only)
AFI2	AFI2 pin (DAQ-2020/2022 only)
AFI3	AFI3 pin (DAQ-2020/2022 only)
AFI4	AFI4 pin (DAQ-2020/2022 only)
AFI5	AFI5 pin (DAQ-2020/2022 only)
AFI6	AFI6 pin (DAQ-2020/2022 only)
AFI7	AFI7 pin (DAQ-2020/2022 only)

dir :	The direction of the connection.
0	receive signal from the connection line. This value is not valid for PCI modules
1	transmit signal to the connection line

---

NOTE	For DAQ-2020/2022, to share clock signal through lines, the frequency of timebase source can't exceed 20M. Therefore, the internal 80M Clock signal can't share between multiple devices.
------	---

---

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorNotStartTriggerModule



ErrorInvalidRouteLine  
ErrorInvalidSignalCode  
ErrorInvalidSignalDirection

## D2K\_Set\_Default\_Load\_Area

### Description

Informs D2K-DASK library the the number of bank where the calibration data are loaded by default when D2K\_Register\_Card is called.

### Cards Support

2020, 2022

### Syntax

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

```
I16 D2K_Set_Default_Load_Area(U16 CardNumber,  
                             U16 bank)
```

Visual Basic

```
D2K_Set_Default_Load_Area (ByVal CardNumber As  
                           Integer, ByVal bank As Integer) As Integer
```

### Parameter

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

**bank :** The storage location on EEPROM. The valid range of the value of bank is 1 through 2. The default value is 1.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport
```

## D2K\_Signal\_DisConn

### Description

Disconnects a device signal from AFIn, PXI\_STAR line or the dedicated PXI trigger bus line.

### Cards Support

PXI-2020, PXI-2022

### Syntax

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

```
I16 D2K_Signal_DisConn (U16 wCardNumber, U16
                        signal, U16 polarity, U16 Line)
```

### Visual Basic

```
D2K_Signal_DisConn (ByVal CardNumber As Integer,
                    ByVal signal As Integer, ByVal polarity As
                    Integer, ByVal Line As Integer) As Integer
```

### Parameter

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

Signal : The specified signal is disconnected from the specified line.  
The valid signal codes are as follows:

SSI_TIME	SSI_TIMEBASE output
SSI_CONV	SSI_ADCONV output
SSI_ADTRIG	SSI_ADTRIG output
SSI_WR	SSI_DAUpdate output
SSI_ADSTART	SSI_ADSTART output
SSI_DATRIG	SSI_DATRIG output
SMB_CLK_IN	SMB_CLK input

polarity : The polarity of the signal  
The valid polarity are as follows:

Signal_ActiveHigh	active high
Signal_ActiveLow	active low

Line : Specified the line that is to be disconnected from the signal.

The valid lines are as follows:

PXI_TRIG_n	PXI trigger bus lines (n is line number and the value is 0 through 7)
PXI_START_TRIG	PXI_START line
AFIn	AFIn pin (n is line number and the value is 0 through 7)
SMB_CLK_OUT	SMB CLK OUT
TRG_IO	TRG_IO pin

## Return Code

NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorFuncNotSupport  
ErrorInvalidRouteLine  
ErrorInvalidSignalCode  
ErrorInvalidSignalDirection

## D2K\_SoftTrigGen

### Description

Generates a software trigger signal manually.

### Cards Support

2020, 2022

### Syntax

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

```
I16 D2K_SoftTrigGen (U16 CardNumber, U8 op)
```

Visual Basic

```
D2K_SoftTrigGen (ByVal CardNumber As Integer,  
                ByVal op As Byte) As Integer
```

### Parameter

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

op : Ignored.

### Return Code

```
NoError
```

## D2K\_SSI\_SourceClear

### Description

Disconnects all device signals from the SSI bus trigger lines.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_SSI_SourceClear (U16 CardNumber)
```

### Visual Basic

```
D2K_SSI_SourceClear (ByVal CardNumber As Integer)  
As Integer
```

### Parameter

*CardNumber* ID of the card performing the operation.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
InvalidCounter
```

## D2K\_SSI\_SourceConn

### Description

Connects a device to the specified SSI bus trigger line.

### Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

### Syntax

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

```
I16 D2K_SSI_SourceConn (U16 CardNumber, U16
                        sigCode)
```

### Visual Basic

```
D2K_SSI_SourceConn (ByVal CardNumber As Integer,
                    ByVal sigCode As Integer) As Integer
```

### Parameters

- CardNumber* ID of the card performing the operation.
- sigCode* Specified SSI signal code number of the device signal to be connected to the SSI bus trigger line. The direction of the connection is transmitted from the device to the SSI bus trigger line. Valid signal codes:

SSI_TIME	SSI_TIMEBASE output
SSI_CONV	SSI_ADCONV output
SSI_WR	SSI_DAWR output
SSI_ADTRIG	SSI_ADTRIG output
SSI_DATRIG	SSI_DATRIG output

NOTE

For DAQ-2020/2022, to share clock signal through lines, the frequency of timebase source can't exceed 20M. Therefore, the internal 80M Clock signal can't share between multiple devices.

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
```

InvalidCounter



# D2K\_SSI\_SourceDisConn

## Description

Disconnects a device signal from the specified SSI bus trigger line.

## Supported Cards

2010, 2005, 2006, 2016, 2020, 2022, 2204, 2205, 2206, 2208, 2213, 2214, 2501, 2502

## Syntax

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

```
I16 D2K_SSI_SourceDisConn (U16 CardNumber, U16
                           sigCode)
```

## Visual Basic

```
D2K_SSI_SourceDisConn (ByVal CardNumber As
                        Integer, ByVal sigCode As Integer) As
                        Integer
```

## Parameters

- CardNumber*      ID of the card performing the operation.
- sigCode*          Specified SSI signal code number of the device signal to be disconnected from the SSI bus trigger line. Valid signal codes:

SSI_TIME	SSI_TIMEBASE output
SSI_CONV	SSI_ADCONV output
SSI_WR	SSI_DAWR output
SSI_ADTRIG	SSI_ADTRIG output
SSI_DATRIG	SSI_DATRIG output

## Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
InvalidCounter
```

## DAQ2005\_Acquire\_AD\_Error

### Description

Acquires the offset and gain errors of the specified AI channel in the specified polarity mode.

### Supported Cards

2005

### Syntax

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

```
I16 DAQ2005_Acquire_AD_Error (U16 CardNumber, U16
    Channel, U16 Polarity, F32 *gain_err, F32
    *offset_err)
```

### Visual Basic

```
DAQ2005_Acquire_AD_Error (ByVal CardNumber As
    Integer, ByVal Channel As Integer, ByVal
    Polarity As Integer, gain_err As Single,
    offset_err As Single) As Integer
```

### Parameters

- CardNumber* ID of the card performing the operation.
- Channel* Analog input channel number. Range is 0 to 3.
- Polarity* Polarity (unipolar or bipolar) of the input channel.  
Valid values:
- |   |          |
|---|----------|
| 1 | Bipolar  |
| 0 | Unipolar |
- gain\_err* Returns the gain error of the specified AI channel.
- offset\_err* Returns the offset error of the specified AI channel.

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

## DAQ2005\_Acquire\_DA\_Error

### Description

Acquires the offset and gain errors of the specified DA channel in the specified polarity mode.

### Supported Cards

2005

### Syntax

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

```
I16 DAQ2005_Acquire_DA_Error (U16 CardNumber, U16
    Channel, U16 Polarity, F32 *gain_err, F32
    *offset_err)
```

### Visual Basic

```
DAQ2005_Acquire_DA_Error (ByVal CardNumber As
    Integer, ByVal Channel As Integer, ByVal
    Polarity As Integer, gain_err As Single,
    offset_err As Single) As Integer
```

### Parameters

- |                   |  |                  |         |                  |          |
|-------------------|--|------------------|---------|------------------|----------|
| <i>CardNumber</i> | ID of the card performing the operation.   |                  |         |                  |          |
| <i>Channel</i>    | Analog output channel number. Range is 0 to 1.   |                  |         |                  |          |
| <i>Polarity</i>   | Polarity (unipolar or bipolar) of the output channel.<br>Valid values:   |                  |         |                  |          |
|                   | <table border="0"> <tr> <td>DAQ2K_DA_BiPolar</td> <td>Bipolar</td> </tr> <tr> <td>DAQ2K_DA_UniPola</td> <td>Unipolar</td> </tr> </table> | DAQ2K_DA_BiPolar | Bipolar | DAQ2K_DA_UniPola | Unipolar |
| DAQ2K_DA_BiPolar  | Bipolar  |                  |         |                  |          |
| DAQ2K_DA_UniPola  | Unipolar   |                  |         |                  |          |
| <i>gain_err</i>   | Returns the gain error of the specified AO channel.  |                  |         |                  |          |
| <i>offset_err</i> | Returns the offset error of the specified AO channel.  |                  |         |                  |          |

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

## DAQ2006\_Acquire\_AD\_Error

### Description

Acquires the offset and gain errors of the specified AI channel in the specified polarity mode.

### Supported Cards

2006

### Syntax

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

```
I16 DAQ2006_Acquire_AD_Error (U16 CardNumber, U16  
    Channel, U16 Polarity, F32 *gain_err, F32  
    *offset_err)
```

### Visual Basic

```
DAQ2006_Acquire_AD_Error (ByVal CardNumber As  
    Integer, ByVal Channel As Integer, ByVal  
    Polarity As Integer, gain_err As Single,  
    offset_err As Single) As Integer
```

### Parameters

- CardNumber* ID of the card performing the operation.
- Channel* Analog input channel number. Range is 0 to 3.
- Polarity* Polarity (unipolar or bipolar) of the input channel.  
Valid values:
- |   |          |
|---|----------|
| 1 | Bipolar  |
| 0 | Unipolar |
- gain\_err* Returns the gain error of the specified AI channel.
- offset\_err* Returns the offset error of the specified AI channel.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```

## DAQ2006\_Acquire\_DA\_Error

### Description

Acquires the offset and gain errors of the specified DA channel in the specified polarity mode.

### Supported Cards

2006

### Syntax

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

```
I16 DAQ2006_Acquire_DA_Error (U16 CardNumber, U16
    Channel, U16 Polarity, F32 *gain_err, F32
    *offset_err)
```

### Visual Basic

```
DAQ2006_Acquire_DA_Error (ByVal CardNumber As
    Integer, ByVal Channel As Integer, ByVal
    Polarity As Integer, gain_err As Single,
    offset_err As Single) As Integer
```

### Parameters

- |                   |  |                  |         |                  |          |
|-------------------|--|------------------|---------|------------------|----------|
| <i>CardNumber</i> | ID of the card performing the operation.   |                  |         |                  |          |
| <i>Channel</i>    | Analog output channel number. Range is 0 to 1.   |                  |         |                  |          |
| <i>Polarity</i>   | Polarity (unipolar or bipolar) of the output channel.<br>Valid values:   |                  |         |                  |          |
|                   | <table border="0"> <tr> <td>DAQ2K_DA_BiPolar</td> <td>Bipolar</td> </tr> <tr> <td>DAQ2K_DA_UniPola</td> <td>Unipolar</td> </tr> </table> | DAQ2K_DA_BiPolar | Bipolar | DAQ2K_DA_UniPola | Unipolar |
| DAQ2K_DA_BiPolar  | Bipolar  |                  |         |                  |          |
| DAQ2K_DA_UniPola  | Unipolar   |                  |         |                  |          |
| <i>gain_err</i>   | Returns the gain error of the specified AO channel.  |                  |         |                  |          |
| <i>offset_err</i> | Returns the offset error of the specified AO channel.  |                  |         |                  |          |

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

## DAQ2010\_Acquire\_AD\_Error

### Description

Acquires the offset and gain errors of the specified AI channel in the specified polarity mode.

### Supported Cards

2010

### Syntax

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

```
I16 DAQ2010_Acquire_AD_Error(U16 CardNumber, U16  
    Channel, U16 Polarity, F32 *gain_err, F32  
    *offset_err)
```

### Visual Basic

```
DAQ2010_Acquire_AD_Error (ByVal CardNumber As  
    Integer, ByVal Channel As Integer, ByVal  
    Polarity As Integer, gain_err As Single,  
    offset_err As Single) As Integer
```

### Parameters

- CardNumber* ID of the card performing the operation.
- Channel* Analog input channel number. Range is 0 to 3.
- Polarity* Polarity (unipolar or bipolar) of the input channel.  
Valid values:
- |   |          |
|---|----------|
| 1 | Bipolar  |
| 0 | Unipolar |
- gain\_err* Returns the gain error of the specified AI channel.
- offset\_err* Returns the offset error of the specified AI channel.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```

## DAQ2010\_Acquire\_DA\_Error

### Description

Acquires the offset and gain errors of the specified DA channel in the specified polarity mode.

### Supported Cards

2010

### Syntax

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

```
I16 DAQ2010_Acquire_DA_Error(U16 CardNumber, U16
    Channel, U16 Polarity, F32 *gain_err, F32
    *offset_err)
```

### Visual Basic

```
DAQ2010_Acquire_DA_Error (ByVal CardNumber As
    Integer, ByVal Channel As Integer, ByVal
    Polarity As Integer, gain_err As Single,
    offset_err As Single) As Integer
```

### Parameters

- |                   |  |                  |         |                  |          |
|-------------------|--|------------------|---------|------------------|----------|
| <i>CardNumber</i> | ID of the card performing the operation.   |                  |         |                  |          |
| <i>Channel</i>    | Analog output channel number. Range is 0 to 1.   |                  |         |                  |          |
| <i>Polarity</i>   | Polarity (unipolar or bipolar) of the output channel.<br>Valid values:   |                  |         |                  |          |
|                   | <table border="0"> <tr> <td>DAQ2K_DA_BiPolar</td> <td>Bipolar</td> </tr> <tr> <td>DAQ2K_DA_UniPola</td> <td>Unipolar</td> </tr> </table> | DAQ2K_DA_BiPolar | Bipolar | DAQ2K_DA_UniPola | Unipolar |
| DAQ2K_DA_BiPolar  | Bipolar  |                  |         |                  |          |
| DAQ2K_DA_UniPola  | Unipolar   |                  |         |                  |          |
| <i>gain_err</i>   | Returns the gain error of the specified AO channel.  |                  |         |                  |          |
| <i>offset_err</i> | Returns the offset error of the specified AO channel.  |                  |         |                  |          |

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

## DAQ2016\_Acquire\_AD\_Error

### Description

Acquires the offset and gain errors of the specified AI channel in the specified polarity mode.

### Supported Cards

2016

### Syntax

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

```
I16 DAQ2016_Acquire_AD_Error (U16 CardNumber, U16
    Channel, U16 Polarity, F32 *gain_err, F32
    *offset_err)
```

### Visual Basic

```
DAQ2016_Acquire_AD_Error (ByVal CardNumber As
    Integer, ByVal Channel As Integer, ByVal
    Polarity As Integer, gain_err As Single,
    offset_err As Single) As Integer
```

### Parameters

- CardNumber* ID of the card performing the operation.
- Channel* Analog input channel number. Range is 0 to 3.
- Polarity* Polarity (unipolar or bipolar) of the input channel.  
Valid values:
- |   |          |
|---|----------|
| 1 | Bipolar  |
| 0 | Unipolar |
- gain\_err* Returns the gain error of the specified AI channel.
- offset\_err* Returns the offset error of the specified AI channel.

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```



## DAQ2016\_Acquire\_DA\_Error

### Description

Acquires the offset and gain errors of the specified DA channel in the specified polarity mode.

### Supported Cards

2016

### Syntax

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

```
I16 DAQ2016_Acquire_DA_Error (U16 CardNumber, U16
    Channel, U16 Polarity, F32 *gain_err, F32
    *offset_err)
```

### Visual Basic

```
DAQ2016_Acquire_DA_Error (ByVal CardNumber As
    Integer, ByVal Channel As Integer, ByVal
    Polarity As Integer, gain_err As Single,
    offset_err As Single) As Integer
```

### Parameters

- |                   |  |                  |         |                  |          |
|-------------------|--|------------------|---------|------------------|----------|
| <i>CardNumber</i> | ID of the card performing the operation.   |                  |         |                  |          |
| <i>Channel</i>    | Analog output channel number. Range is 0 to 1.   |                  |         |                  |          |
| <i>Polarity</i>   | Polarity (unipolar or bipolar) of the output channel.<br>Valid values:   |                  |         |                  |          |
|                   | <table border="0"> <tr> <td>DAQ2K_DA_BiPolar</td> <td>Bipolar</td> </tr> <tr> <td>DAQ2K_DA_UniPola</td> <td>Unipolar</td> </tr> </table> | DAQ2K_DA_BiPolar | Bipolar | DAQ2K_DA_UniPola | Unipolar |
| DAQ2K_DA_BiPolar  | Bipolar  |                  |         |                  |          |
| DAQ2K_DA_UniPola  | Unipolar   |                  |         |                  |          |
| <i>gain_err</i>   | Returns the gain error of the specified AO channel.  |                  |         |                  |          |
| <i>offset_err</i> | Returns the offset error of the specified AO channel.  |                  |         |                  |          |

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

## DAQ2204\_Acquire\_AD\_Error

### Description

Acquires the offset and gain errors of ADC.

### Supported Cards

2204

### Syntax

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

```
I16 DAQ2204_Acquire_AD_Error (U16 CardNumber, F32
    *gain_err, F32 *bioffset_err, F32
    *unioffset_err, F32 *hg_bios_err)
```

### Visual Basic

```
DAQ2204_Acquire_AD_Error (ByVal CardNumber As
    Integer, gain_err As Single, bioffset_err As
    Single, unioffset_err As Single, hg_bios_err
    As Single) As Integer
```

### Parameters

- |                      |  |
|----------------------|--|
| <i>CardNumber</i>    | ID of the card performing the operation.                       |
| <i>gain_err</i>      | Returns the gain error of the ADC.                             |
| <i>bioffset_err</i>  | Returns the offset error of the ADC in bipolar mode.           |
| <i>unioffset_err</i> | Returns the offset error of the ADC in unipolar mode.          |
| <i>hg_bios_err</i>   | Returns the high-gain offset error of the ADC in bipolar mode. |

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```

## DAQ2204\_Acquire\_DA\_Error

### Description

Acquires the offset and gain errors of the specified DA channel in the specified polarity.

### Supported Cards

2204

### Syntax

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

```
I16 DAQ2204_Acquire_DA_Error(U16 CardNumber, U16
    Channel, U16 Polarity, F32 *da0v_err, F32
    *da5v_err)
```

### Visual Basic

```
DAQ2204_Acquire_DA_Error (ByVal CardNumber As
    Integer, ByVal Channel As Integer, ByVal
    Polarity As Integer, da0v_err As Single,
    da5v_err As Single) As Integer
```

### Parameters

- |                   |  |                  |         |                  |          |
|-------------------|--|------------------|---------|------------------|----------|
| <i>CardNumber</i> | ID of the card performing the operation.   |                  |         |                  |          |
| <i>Channel</i>    | Analog output channel number. Range is 0 to 1.   |                  |         |                  |          |
| <i>Polarity</i>   | Polarity (unipolar or bipolar) of the output channel.<br>Valid values:   |                  |         |                  |          |
|                   | <table border="0"> <tr> <td>DAQ2K_DA_BiPolar</td> <td>Bipolar</td> </tr> <tr> <td>DAQ2K_DA_UniPola</td> <td>Unipolar</td> </tr> </table> | DAQ2K_DA_BiPolar | Bipolar | DAQ2K_DA_UniPola | Unipolar |
| DAQ2K_DA_BiPolar  | Bipolar  |                  |         |                  |          |
| DAQ2K_DA_UniPola  | Unipolar   |                  |         |                  |          |
| <i>da0v_err</i>   | Returns the gain error of the specified AO channel.  |                  |         |                  |          |
| <i>da5v_err</i>   | Returns the offset error of the specified AO channel.  |                  |         |                  |          |

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

## DAQ2205\_Acquire\_AD\_Error

### Description

Acquires the offset and gain errors of ADC.

### Supported Cards

2205

### Syntax

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

```
I16 DAQ2205_Acquire_AD_Error(U16 CardNumber, F32
    *gain_err, F32 *bioffset_err, F32
    *unioffset_err, F32 *hg_bios_err)
```

### Visual Basic

```
DAQ2205_Acquire_AD_Error (ByVal CardNumber As
    Integer, gain_err As Single, bioffset_err As
    Single, unioffset_err As Single, hg_bios_err
    As Single) As Integer
```

### Parameters

- |                      |  |
|----------------------|--|
| <i>CardNumber</i>    | ID of the card performing the operation.                       |
| <i>gain_err</i>      | Returns the gain error of the ADC.                             |
| <i>bioffset_err</i>  | Returns the offset error of the ADC in bipolar mode.           |
| <i>unioffset_err</i> | Returns the offset error of the ADC in unipolar mode.          |
| <i>hg_bios_err</i>   | Returns the high-gain offset error of the ADC in bipolar mode. |

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```

## DAQ2205\_Acquire\_DA\_Error

### Description

Acquires the offset and gain errors of the specified DA channel in the specified polarity.

### Supported Cards

2205

### Syntax

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

```
I16 DAQ2205_Acquire_DA_Error(U16 CardNumber, U16
    Channel, U16 Polarity, F32 *da0v_err, F32
    *da5v_err)
```

### Visual Basic

```
DAQ2205_Acquire_DA_Error (ByVal CardNumber As
    Integer, ByVal Channel As Integer, ByVal
    Polarity As Integer, da0v_err As Single,
    da5v_err As Single) As Integer
```

### Parameters

- |                   |  |                  |         |                  |          |
|-------------------|--|------------------|---------|------------------|----------|
| <i>CardNumber</i> | ID of the card performing the operation.   |                  |         |                  |          |
| <i>Channel</i>    | Analog output channel number. Range is 0 to 1.   |                  |         |                  |          |
| <i>Polarity</i>   | Polarity (unipolar or bipolar) of the output channel.<br>Valid values:   |                  |         |                  |          |
|                   | <table border="0"> <tr> <td>DAQ2K_DA_BiPolar</td> <td>Bipolar</td> </tr> <tr> <td>DAQ2K_DA_UniPola</td> <td>Unipolar</td> </tr> </table> | DAQ2K_DA_BiPolar | Bipolar | DAQ2K_DA_UniPola | Unipolar |
| DAQ2K_DA_BiPolar  | Bipolar  |                  |         |                  |          |
| DAQ2K_DA_UniPola  | Unipolar   |                  |         |                  |          |
| <i>da0v_err</i>   | Returns the offset error of the specified AO channel.  |                  |         |                  |          |
| <i>da5v_err</i>   | Returns the gain error of the specified AO channel.  |                  |         |                  |          |

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

## DAQ2206\_Acquire\_AD\_Error

### Description

Acquires the offset and gain errors of ADC.

### Supported Cards

2206

### Syntax

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

```
I16 DAQ2206_Acquire_AD_Error(U16 CardNumber, F32
    *gain_err, F32 *bioffset_err, F32
    *unioffset_err, F32 *hg_bios_err)
```

### Visual Basic

```
DAQ2206_Acquire_AD_Error (ByVal CardNumber As
    Integer, gain_err As Single, bioffset_err As
    Single, unioffset_err As Single, hg_bios_err
    As Single) As Integer
```

### Parameters

- |                      |  |
|----------------------|--|
| <i>CardNumber</i>    | ID of the card performing the operation.                       |
| <i>gain_err</i>      | Returns the gain error of the ADC.                             |
| <i>bioffset_err</i>  | Returns the offset error of the ADC in bipolar mode.           |
| <i>unioffset_err</i> | Returns the offset error of the ADC in unipolar mode.          |
| <i>hg_bios_err</i>   | Returns the high-gain offset error of the ADC in bipolar mode. |

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

## DAQ2206\_Acquire\_DA\_Error

### Description

Acquires the offset and gain errors of the specified DA channel in the specified polarity.

### Supported Cards

2206

### Syntax

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

```
I16 DAQ2206_Acquire_DA_Error(U16 CardNumber, U16
    Channel, U16 Polarity, F32 *da0v_err, F32
    *da5v_err)
```

### Visual Basic

```
DAQ2206_Acquire_DA_Error (ByVal CardNumber As
    Integer, ByVal Channel As Integer, ByVal
    Polarity As Integer, da0v_err As Single,
    da5v_err As Single) As Integer
```

### Parameters

- |                   |  |                  |         |                  |          |
|-------------------|--|------------------|---------|------------------|----------|
| <i>CardNumber</i> | ID of the card performing the operation.   |                  |         |                  |          |
| <i>Channel</i>    | Analog output channel number. Range is 0 to 1.   |                  |         |                  |          |
| <i>Polarity</i>   | Polarity (unipolar or bipolar) of the output channel.<br>Valid values:   |                  |         |                  |          |
|                   | <table border="0"> <tr> <td>DAQ2K_DA_BiPolar</td> <td>Bipolar</td> </tr> <tr> <td>DAQ2K_DA_UniPola</td> <td>Unipolar</td> </tr> </table> | DAQ2K_DA_BiPolar | Bipolar | DAQ2K_DA_UniPola | Unipolar |
| DAQ2K_DA_BiPolar  | Bipolar  |                  |         |                  |          |
| DAQ2K_DA_UniPola  | Unipolar   |                  |         |                  |          |
| <i>da0v_err</i>   | Returns the offset error of the specified AO channel.  |                  |         |                  |          |
| <i>da5v_err</i>   | Returns the gain error of the specified AO channel.  |                  |         |                  |          |

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

## DAQ2208\_Acquire\_AD\_Error

### Description

Acquires the offset and gain errors of ADC.

### Supported Cards

2208

### Syntax

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

```
I16 DAQ2208_Acquire_AD_Error(U16 CardNumber, F32
    *gain_err, F32 *bioffset_err, F32
    *unioffset_err, F32 *hg_bios_err)
```

### Visual Basic

```
DAQ2208_Acquire_AD_Error (ByVal CardNumber As
    Integer, gain_err As Single, bioffset_err As
    Single, unioffset_err As Single, hg_bios_err
    As Single) As Integer
```

### Parameters

- |                      |  |
|----------------------|--|
| <i>CardNumber</i>    | ID of the card performing the operation.                       |
| <i>gain_err</i>      | Returns the gain error of the ADC.                             |
| <i>bioffset_err</i>  | Returns the offset error of the ADC in bipolar mode.           |
| <i>unioffset_err</i> | Returns the offset error of the ADC in unipolar mode.          |
| <i>hg_bios_err</i>   | Returns the high-gain offset error of the ADC in bipolar mode. |

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport  
ErrorInvalidIoChannel
```



## DAQ2213\_Acquire\_AD\_Error

### Description

Acquires the offset and gain errors of ADC.

### Supported Cards

2213

### Syntax

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

```
I16 DAQ2213_Acquire_AD_Error(U16 CardNumber, F32
    *gain_err, F32 *bioffset_err, F32
    *unioffset_err, F32 *hg_bios_err)
```

### Visual Basic

```
DAQ2213_Acquire_AD_Error (ByVal CardNumber As
    Integer, gain_err As Single, bioffset_err As
    Single, unioffset_err As Single, hg_bios_err
    As Single) As Integer
```

### Parameters

<i>CardNumber</i>	ID of the card performing the operation.
<i>gain_err</i>	Returns the gain error of the ADC.
<i>bioffset_err</i>	Returns the offset error of the ADC in bipolar mode.
<i>unioffset_err</i>	Returns the offset error of the ADC in unipolar mode.
<i>hg_bios_err</i>	Returns the high-gain offset error of the ADC in bipolar mode.

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

## DAQ2214\_Acquire\_AD\_Error

### Description

Acquires the offset and gain errors of ADC.

### Supported Cards

2214

### Syntax

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

```
I16 DAQ2214_Acquire_AD_Error(U16 CardNumber, F32
    *gain_err, F32 *bioffset_err, F32
    *unioffset_err, F32 *hg_bios_err)
```

### Visual Basic

```
DAQ2214_Acquire_AD_Error (ByVal CardNumber As
    Integer, gain_err As Single, bioffset_err As
    Single, unioffset_err As Single, hg_bios_err
    As Single) As Integer
```

### Parameters

- |                      |  |
|----------------------|--|
| <i>CardNumber</i>    | ID of the card performing the operation.                       |
| <i>gain_err</i>      | Returns the gain error of the ADC.                             |
| <i>bioffset_err</i>  | Returns the offset error of the ADC in bipolar mode.           |
| <i>unioffset_err</i> | Returns the offset error of the ADC in unipolar mode.          |
| <i>hg_bios_err</i>   | Returns the high-gain offset error of the ADC in bipolar mode. |

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

## DAQ2214\_Acquire\_DA\_Error

### Description

Acquire the offset and gain errors of the specified DA channel in the specified polarity.

### Supported Cards

2214

### Syntax

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

```
I16 DAQ2214_Acquire_DA_Error(U16 CardNumber, U16
    Channel, U16 Polarity, F32 *da0v_err, F32
    *da5v_err)
```

### Visual Basic

```
DAQ2214_Acquire_DA_Error (ByVal CardNumber As
    Integer, ByVal Channel As Integer, ByVal
    Polarity As Integer, da0v_err As Single,
    da5v_err As Single) As Integer
```

### Parameters

- |                   |  |                  |         |                  |          |
|-------------------|--|------------------|---------|------------------|----------|
| <i>CardNumber</i> | ID of the card performing the operation.   |                  |         |                  |          |
| <i>Channel</i>    | Analog output channel number. Range is 0 to 1.   |                  |         |                  |          |
| <i>Polarity</i>   | Polarity (unipolar or bipolar) of the output channel.<br>Valid values:   |                  |         |                  |          |
|                   | <table border="0"> <tr> <td>DAQ2K_DA_BiPolar</td> <td>Bipolar</td> </tr> <tr> <td>DAQ2K_DA_UniPola</td> <td>Unipolar</td> </tr> </table> | DAQ2K_DA_BiPolar | Bipolar | DAQ2K_DA_UniPola | Unipolar |
| DAQ2K_DA_BiPolar  | Bipolar  |                  |         |                  |          |
| DAQ2K_DA_UniPola  | Unipolar   |                  |         |                  |          |
| <i>da0v_err</i>   | Returns the offset error of the specified AO channel.  |                  |         |                  |          |
| <i>da5v_err</i>   | Returns the gain error of the specified AO channel.  |                  |         |                  |          |

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```

## DAQ250X\_Acquire\_AD\_Error

### Description

Acquires the offset and gain errors in the specified polarity mode.

### Supported Cards

2501, 2502

### Syntax

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

```
I16 DAQ250X_Acquire_AD_Error(I16 CardNumber, U16  
    Polarity, F32 *gain_err, F32 *offset_err)
```

### Visual Basic

```
DAQ250X_Acquire_AD_Error (ByVal CardNumber As  
    Integer, ByVal Polarity As Integer, gain_err  
    As Single, offset_err As Single) As Integer
```

### Parameters

*CardNumber* ID of the card performing the operation.

*Polarity* Polarity (unipolar or bipolar) of the input channel.  
Valid values:

1	Bipolar
0	Unipolar

*gain\_err* Returns the gain error of the specified AI channel.

*offset\_err* Returns the offset error of the specified AI channel.

### Return Code

```
NoError  
ErrorInvalidCardNumber  
ErrorCardNotRegistered  
ErrorFuncNotSupport
```

## DAQ250X\_Acquire\_DA\_Error

### Description

Acquires the offset and gain errors of the specified DA channel in the specified polarity mode.

### Supported Cards

2501, 2502

### Syntax

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

```
I16 DAQ250X_Acquire_DA_Error(U16 CardNumber, U16
    Channel, U16 Polarity, F32 *gain_err, F32
    *offset_err)
```

### Visual Basic

```
DAQ250X_Acquire_DA_Error (ByVal CardNumber As
    Integer, ByVal Channel As Integer, ByVal
    Polarity As Integer, gain_err As Single,
    offset_err As Single) As Integer
```

### Parameters

- |                   |  |                  |         |                  |          |
|-------------------|--|------------------|---------|------------------|----------|
| <i>CardNumber</i> | ID of the card performing the operation.   |                  |         |                  |          |
| <i>Channel</i>    | Analog output channel number. Range is 0 to 3 for DAQ-2501 and 0 to 7 for DAQ-2502.  |                  |         |                  |          |
| <i>Polarity</i>   | Polarity (unipolar or bipolar) of the output channel. Valid values:  |                  |         |                  |          |
|                   | <table border="0"> <tr> <td>DAQ2K_DA_BiPolar</td> <td>Bipolar</td> </tr> <tr> <td>DAQ2K_DA_UniPola</td> <td>Unipolar</td> </tr> </table> | DAQ2K_DA_BiPolar | Bipolar | DAQ2K_DA_UniPola | Unipolar |
| DAQ2K_DA_BiPolar  | Bipolar  |                  |         |                  |          |
| DAQ2K_DA_UniPola  | Unipolar   |                  |         |                  |          |
| <i>gain_err</i>   | Returns the gain error of the specified AO channel.  |                  |         |                  |          |
| <i>offset_err</i> | Returns the offset error of the specified AO channel.  |                  |         |                  |          |

### Return Code

```
NoError
ErrorInvalidCardNumber
ErrorCardNotRegistered
ErrorFuncNotSupport
ErrorInvalidIoChannel
```



# Appendix

## Appendix A Status Codes

This appendix lists the status codes returned by D2K-DASK, including the name and description.

Each D2K-DASK function returns a status code that indicates whether the function was performed successfully. When a D2K-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.

**Table 3-1: Status Codes**

Status Code	Status Name	Description
-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 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
-43	ErrorInvalidSyncMode	The sync. mode of operation is invalid
-44	ErrorInvalidBufferID	The buffer id selected is invalid
-45	ErrorInvalidCNTInterval	The counter value is invalid
-46	ErrorReTrigModeNotAllowed	The Re-Trigger mode of operation is invalid
-47	ErrorResetBufferNotAllowed	The buffer is not allowed to be reset
-48	ErrorAnaTriggerLevel	The value of analog trigger level is invalid
-49	ErrorDAQEvent	The DAQEvent is invalid
-50	ErrorInvalidCounterValue	The Counter value is invalid
-51	ErrorOffsetCalibration	Error on AI offset calibration
-52	ErrorGainCalibration	Error on AI Gain calibration
-53	ErrorCountOutOfSDRAMSize	The data count is out of the size of on-board SDRAM
-54	ErrorNotStatTriggerModule	The PXI module is not at slot 2
-55	ErrorInvalidRouteLine	The signal route line is invalid

**Table 3-1: Status Codes**



Status Code	Status Name	Description
-56	ErrorInvalidSignalCode	The signal code is invalid
-57	ErrorInvalidSignalDirection	The signal route direction is invalid
-58	ErrorTRGOSCalibration	The trigger offset calibration is invalid
-59	ErrorNoSDRAM	No onboard embedded SDRAM
-60	ErrorIntegrationGain	The integration gain is invalid
-61	ErrorAcquisitionTiming	The acquisition time-out value is invalid
-62	ErrorIntegrationTiming	The value of integration time is invalid
-70	ErrorInvalidTimeBase	The Timebase is invalid
-71	ErrorUndefinedParameter	The parameter is undefined
-110	ErrorCalAddress	The memory address of calibration is invalid
-111	ErrorInvalidCalBank	The calibration bank is invalid
-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 Un-mapping API is failed
-214	ErrorCTRIoctl	The counter API is failed
-215	ErrorGetResIoctl	The resource getting API is failed

**Table 3-1: Status Codes**

## Appendix B AI Range Codes

The table below lists the analog input range of NuDAQ PCI-bus cards.

AD_B_10_V	Bipolar -10V to +10V
AD_B_5_V	Bipolar -5V to +5V
AD_B_2_5_V	Bipolar -2.5V to +2.5V
AD_B_1_25_V	Bipolar -1.25V to +1.25V
AD_B_0_625_V	Bipolar -0.625V to +0.625V
AD_B_0_3125_V	Bipolar -0.3125V to +0.3125V
AD_B_0_5_V	Bipolar -0.5V to +0.5V
AD_B_0_05_V	Bipolar -0.05V to +0.05V
AD_B_0_005_V	Bipolar -0.005V to +0.005V
AD_B_1_V	Bipolar -1V to +1V
AD_B_0_1_V	Bipolar -0.1V to +0.1V
AD_B_0_01_V	Bipolar -0.01V to +0.01V
AD_B_0_001_V	Bipolar -0.01V to +0.001V
AD_B_2_V	Bipolar -2V to +2V
AD_B_0_2_V	Bipolar -0.2V to +0.2V
AD_U_20_V	Unipolar 0 to +20V
AD_U_10_V	Unipolar 0 to +10V
AD_U_5_V	Unipolar 0 to +5V
AD_U_2_5_V	Unipolar 0 to +2.5V
AD_U_1_25_V	Unipolar 0 to +1.25V
AD_U_1_V	Unipolar 0 to +1V
AD_U_0_1_V	Unipolar 0 to +0.1V
AD_U_0_01_V	Unipolar 0 to +0.01V
AD_U_0_001_V	Unipolar 0 to +0.001V
AD_U_2_V	Unipolar 0 to +2V

**Table 3-2: AI Range Codes**

Valid values for each card:

DAQ-2005 DAQ-2006 DAQ-2010 DAQ-2016 DAQ-2205 DAQ-2206 DAQ-2213 DAQ-2214	<ul style="list-style-type: none"> <li>• AD_B_10_V</li> <li>• AD_B_5_V</li> <li>• AD_B_2_5_V</li> <li>• AD_B_1_25_V</li> <li>• AD_U_10_V</li> <li>• AD_U_5_V</li> <li>• AD_U_2_5_V</li> <li>• AD_U_1_25_V</li> </ul>
DAQ-2204 DAQ-2208	<ul style="list-style-type: none"> <li>• AD_B_10_V</li> <li>• AD_B_5_V</li> <li>• AD_B_2_5_V</li> <li>• AD_B_2_V</li> <li>• AD_B_1_25_V</li> <li>• AD_B_1_V</li> <li>• AD_B_0_5_V</li> <li>• AD_B_0_25_V</li> <li>• AD_B_0_2_V</li> <li>• AD_B_0_05_V</li> <li>• AD_U_10_V</li> <li>• AD_U_5_V</li> <li>• AD_U_4_V</li> <li>• AD_U_2_5_V</li> <li>• AD_U_2_V</li> <li>• AD_U_1_V</li> <li>• AD_U_0_5_V</li> <li>• AD_U_0_4_V</li> <li>• AD_U_0_1_V</li> </ul>
DAQ-2501 DAQ-2502	<ul style="list-style-type: none"> <li>• AD_B_10_V</li> <li>• AD_U_10_V</li> </ul>
DAQ-2020 DAQ-2022	<ul style="list-style-type: none"> <li>• AD_B_10_V</li> <li>• AD_B_2_5_V</li> </ul>

**Table 3-3: Valid Card AI Range**

## 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\* channel no. (CH#) \* A/D converted data (ND) \* Value returned from AI function (OD)

Card Type	Data Format	Value calculation
		<ul style="list-style-type: none"> <li>channel no. (CH#)</li> <li>A/D converted data (ND)</li> <li>Value returned from AI function (OD)</li> </ul>
DAQ-2010	Every 16-bit signed integer data: D13 D12...D1D0 b1 b0, where D13,D12, ... , D0 is the A/D converted data and b1, b0 is the simultaneous digital input data.	<ul style="list-style-type: none"> <li>ND = OD &gt;&gt;2</li> <li>ND = OD/4</li> </ul>
DAQ-2005 DAQ-2006 DAQ-2016	Every 16-bit unsigned integer data: D15 D14 D13...D1 D0, where D15, D14, ... , D0 is the converted A/D data.	<ul style="list-style-type: none"> <li>ND = OD</li> </ul>
DAQ-2204	Every 16-bit signed integer data: D12 D11...D1 D0 b3 b2 b1 b0, where D12, D11, ... , D0 is the converted A/D data and b3, b2, b1, b0 is the simultaneous digital input data.	<ul style="list-style-type: none"> <li>ND = OD &gt;&gt;4</li> <li>ND = OD/16</li> </ul>
DAQ-2205 DAQ-2206 DAQ-2213 DAQ-2214 DAQ-2020 DAQ-2022	Every 16-bit signed integer data D15 D14 D13...D1 D0, where D15, D14, ... , D0 is the converted A/D data.	<ul style="list-style-type: none"> <li>ND = OD</li> </ul>
DAQ-2208	Every 16-bit signed integer data: D12 D11...D1 D0 b3 b2 b1 b0, where D12, D11, ... , D0 is the converted A/D data and b3, b2, b1, b0 are not used.	<ul style="list-style-type: none"> <li>ND = OD &gt;&gt;4</li> <li>ND = OD/16</li> </ul>
DAQ-2501 DAQ-2502	Every 16-bit signed integer data: D13 D12... D1 D0 b1 b0, where D13, D12, ... , D0 is the converted A/D data and b1, b0 is the AI auto-scan channel.	<ul style="list-style-type: none"> <li>ND = OD &gt;&gt;2</li> <li>ND = OD/4</li> </ul>

**Table 3-4: AI Data Format**

## Appendix D Data File Format

This appendix describes the file format of the data files generated by the functions performing continuous data acquisition followed by storing the data to disk.

The data file includes three parts, Header, ChannelRange (optional), and Data block. The file structure is shown below:

Header
ChannelRange (Optional)
DAQ data

### Header

The header part records the information related to the stored data and has 60 bytes of length. The data structure of the file header is listed in the table:

Header Total Length: 60 bytes			
Elements	Type	Size (bytes)	Comments
ID	char	10	File ID (ADLINKDAQ2)
card_type	short	2	Card Type (DAQ-2010)
num_of_channel	short	2	Number of scanned channels (1, 2)
channel_no	unsigned char	1	Channel number where data is read from. Only available as the num_of_channel is 1. (0, 1)
num_of_scan	long	4	Number of scan for each channel (total count / num_of_channel)
data_width	short	2	Data width (0: 8 bits, 1: 16 bits, 2: 32 bits)
channel_order	short	2	Channel scanned sequence 0: normal (0-1-2-3) 1: reverse (3-2-1-0) 2: custom* (0, 1, 3)
ad_range	short	2	AI range code. (0: AD_B_5V) Refer to Appendix B
scan_rate	double	8	Scanning rate of each channel (total sampling rate/num_of_channel)

Header Total Length: 60 bytes			
Elements	Type	Size (bytes)	Comments
num_of_channel_range	short	2	Number of ChannelRange* structure
start_date	char	8	Starting date of data acquisition (12/31/99)
start_time	char	8	Starting time of data acquisition (18:30:25)
start_millisec	char	3	Starting millisecond of data acquisition (360)
reserved	char	6	Not used

- \* If the num\_of\_channel\_range is 0, the ChannelRange block will not be included in the data file.
- \* The channel\_order is set to "custom" only when the card supports variant channel scanning order.

## ChannelRange

The ChannelRange part records the channel number and data range information related to the stored data. This part consists of several channel and range units. The length of each unit is 2 bytes. The total length depends on the value of num\_of\_channel\_range (one element of the file header) and is calculated with this formula:

Total Length = 2 \* num\_of\_channel\_range bytes

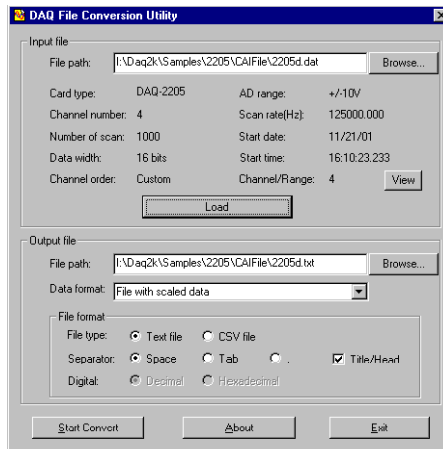
The data structure of each ChannelRange unit is listed below:

ChannelRange (Unit Length: 2 bytes)			
Elements	Type	Size (bytes)	Comments
channel	char	1	Scanned channel number (0, 1)
range	char	1	AI range code of channel. (0: AD_B_5V) Refer to Appendix B

## Data Block

The data is written to file in a 16-bit binary format, with the lower byte first (little endian). For example, the value 0x1234 is written to disk with 34 first followed by 12. The total length of the data block depends on the data width and the total data count.

The file is written in Binary format and may not be read by normal text editor. You can use any binary file editor to view it or the functions used for reading files (such as fread) to get the file information and data value. D2K-DASK provides the DAQCvt utility to convert the binary file. Refer to the D2K-DASK user manual for details.



DAQCvt translates the information stored in the header part and the ChannelRange part, then displays the corresponding information in the **Input File** frame of DAQCvt main window. After setting the properties (File Path, Format, ...etc) of the converted file and after clicking the **Start Convert** button, DAQCvt gets rid of header and ChannelRange parts and converts the data in data block according to the card type and the data width. DAQCvt also writes the converted data to a disk and lets you use any text editor or Excel to view or analyze the accessed data.

