UD-DASK

Data Acquisition Software Development Kit

For USBDAQ USB modules

Function Reference Manual

Manual Rev 1.6.2: August 28, 2015

Trademarks

CONTENTS

How to Use This Manual

This manual is designed to help you use the UD-DASK software driver for USBDAQ USB data acquisition modules. The manual describes how to install and use the software library to meet your requirements and help you program your own software applications. It is organized as follows:

- Chapter 1, "Using UD-DASK Functions" gives the important information about how to apply the function descriptions in this manual to your programming language and environment.
- Chapter 2, "Function Description" gives the detailed description of each function call USB-DASK provided.
- Appendix A, "Status Codes" lists the status codes returned by UD-DASK functions, as well as their meanings.
- Appendix B, "AI Range Codes " lists all the valid AI range codes for each card.
- Appendix C, "AI Data Format" lists the AI data format for the cards performing analog input operation, as well as the calculation methods to retrieve the A/D converted data and the channel where the data read from.
- Appendix D, "Function Support" shows which data acquisition hardware each UD-DASK function supports.

1

Using UD-DASK Functions

UD-DASK is a software driver for USBAQ USB data acquisition modules. It is a high performance data acquisition driver for developing custom applications under Windows environment.

Using UD-DASK also lets you take advantage of the power and features of Microsoft Windows for your data acquisition applications. These include running multiple applications and using extended memory. Also, using UD-DASK under Visual Basic environment makes it easy to create custom user interfaces and graphics.

1.1 The Fundamentals of Building Windows XP/7 Application with UD-DASK

1.1.1 Creating a Windows XP/7 USB-DASK Application Using Microsoft Visual C/C++

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

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

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

#include "UsbDask.h"

step 3. Build your application.

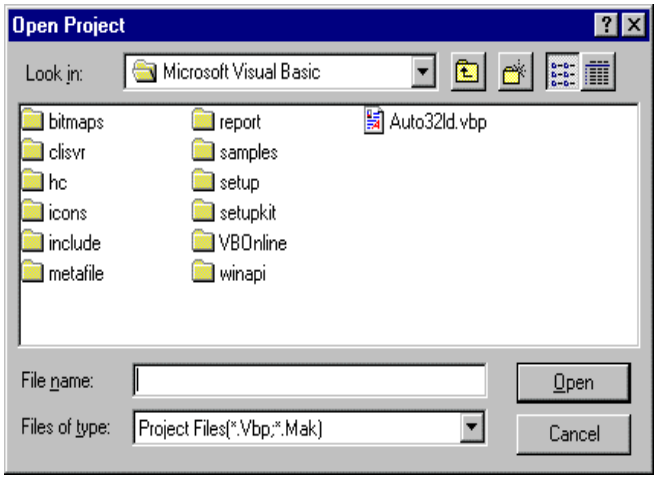
Setting the appropriate compile and link options, then build your application by selecting the Build command from Build menu (Visual C/C++ 6.0). Remember to link USB-DASK's import library USB-DASK.LIB / USB-DASK64.LIB.

1.1.2 Creating a Windows XP/7 UD-DASK Application Using Microsoft Visual Basic

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

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

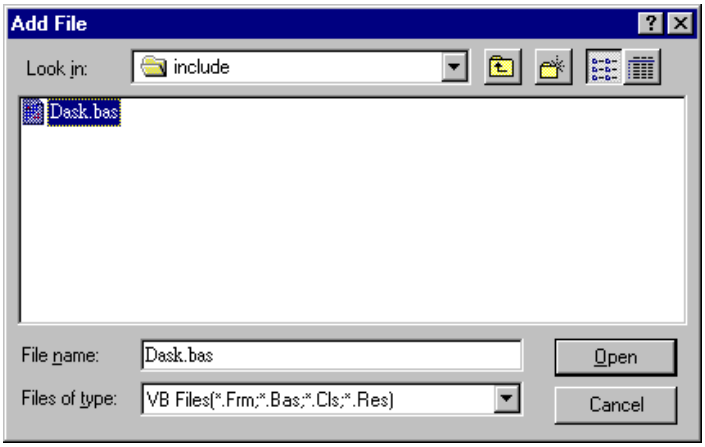
Open a new project by selecting the New Project command from the File menu. If it is an existing project, open it by selecting the Open Project command from the File menu. Then the Open Project dialog box appears.



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

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

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



Select UsbDask.BAS from the Files list by double-clicking on it. If you can't find this file in the list, make sure the list is displaying files from the correct directory. By default, UsbDask.BAS is installed in $InstDir\UD-DASK\INCLUDE.

step 3. Design the interface for the application.

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

step 4. Set properties for the controls.

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

step 5. Write the event code.

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

step 6. Run your application.

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

step 7. Distribute your application.

Once you have finished a project, you can save the application as an executable (.EXE) file by using the Make EXE File command on the File menu. And once you have saved your application as an executable file, you've ready to distribute it. When you distribute your application, remember also to include the UD-DASK's DLL and driver files. These files should be copied to their appropriate directory as section 1.4.1 described.

1.2 UD-DASK Functions Overview

UD-DASK functions are grouped to the following 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**
- **Timer/Counter Function Group**
- **DIO Function Group**

- Digital Input/Output Configuration function

2

Function Description

This chapter contains the detailed description of UD-DASK functions, including the UD-DASK data types and function reference. The functions are arraned alphabetically in 3.2 *Function Reference*

2.1 Data Types

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

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

2.2 Function Reference

2.2.1 UD_AI_1902_Config

@ Description

Informs UD-DASK library of the conversion source and trigger mode selected for the USB-1901/1902/1903 module with module ID *ModuleNum*. You must call this function before calling function to perform continuous analog input operation.

@ Modules Support

USB-1901/USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_1902_Config (U16 ModuleNum, U16 wConfigCtrl, U16 wTrigCtrl, U32 dwTrgLevel, U32 dwReTriggerCnt)

Visual Basic

UD_AI_1902_Config (ByVal ModuleNum As Integer, ByVal wConfigCtrl As Integer, ByVal wTrigCtrl As Integer, ByVal dwTrgLevel As Long, ByVal dwReTriggerCnt As Long) As Integer

@ Parameter

ModuleNum : The id of the module that wants to perform this operation.

wConfigCtrl : The settings for input-type and conversion-source. The valid settings can be combined with OR (|) operator.

Input-type:

P1902_AI_NonRef_SingEnded: None-Reference Single Ended.
P1902_AI_SingEnded: Single Ended.
P1902_AI_PseudoDifferential: Pseudo Differential.

Conversion-source:

P1902_AI_CONVSRC_INT: on-board Programmable pacer
P1902_AI_CONVSRC_EXT: external signal trigger

wTrigCtrl: The settings for trigger-source, trigger-polarity, trigger-mode and re-trigger. The valid settings can be combined with OR (|) operator.

Trigger-Source:

P1902_AI_TRGSRC_AI0 ~ P1902_AI_TRGSRC_AI15: Analog trigger from AI0 ~ AI15.
P1902_AI_TRGSRC_DTRIG: Digital trigger from AIDTRIG.

Trigger-Polarity:

P1902_AI_TrgPositive: Rising edge.
P1902_AI_TrgNegative: Falling edge.
P1902_AI_Gate_PauseLow: Pause low when trigger-mode is set as P1902_AI_TRGMOD_GATED.
P1902_AI_Gate_PauseHigh: Pause high when trigger-mode is set as P1902_AI_TRGMOD_GATED.

Trigger-Mode:

P1902_AI_TRGMOD_POST: Post-trigger.
P1902_AI_TRGMOD_GATED: Gated-trigger.
P1902_AI_TRGMOD_DELAY: Delay-trigger.

Re-Trigger:

P1902_AI_EnReTigger: Enable Re-Trigger.

dwTrgLevel: The trigger level when trigger-source is set as P1902_AI_TRGSRC_AI0 ~ P1902_AI_TRGSRC_AI15.The onboard circuit will use this setting to compare the ADC data.
Please refer the AdRange parameter in UD_AI_ContReadChannel() to translate the expected trigger voltage to trigger-level.
The translation formula is:
ADC maximum number * (expected trigger-voltage / maximum voltage of selected AdRange).
For instance, if AD_B_10_V is selected and the expected trigger-voltage is 5V, the trigger-level is 32767 * (5 / 10) = 16383 = 0x3FFF.

dwReTrigCnt: The count of re-trigger is required when the P1902_AI_EnReTigger is set in wTrigCtrl parameter.

dwDelayCount:The count for delay. This setting is applied to one 80MHz-based timer-counter.For instance, the value, 4,000,000, means 50 millisecond delay.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorInvalidCardType: Indicates the module-type is not USB_1901/1902/1903.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorInvalidInputSignal: Indicates the invalid input-signal is assigned.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidDelayCount: The delay-count is less than 1, or less than 320 if the trigger-source is configured as P1902_AI_TRGSRC_AI0 ~ P1902_AI_TRGSRC_AI15.
ErrorInvalidTriggerMode: This error is caused by the incorrect combination of trigger-settings.
i. P1902_AI_TRGMOD_GATED is configure, but the trigger source is not P1902_AI_TRGSRC_DTRIG.
ii. enable P1902_AI_EnReTigger with P1902_AI_TRGSRC_SOFT trigger-source.
iii. enable P1902_AI_EnReTigger with P1902_AI_TRGMOD_GATED trigger-mode.
ErrorConfigIoctl: Failed to forward the command to driver, please call GetLastError() for detailed system-error.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorContIoActive: The continuous AI operation is still running.

2.2.2 UD_AI_2401_Config

@ Description

Informs UD-DASK library of the analog input-type and trigger mode selected for the USB-2401 module with module ID *ModuleNum*. You must call this function before calling function to perform continuous analog input operation.

@ Modules Support

USB-2401

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_2401_Config (U16 ModuleNum, U16 wChanCfg1, U16 wChanCfg2, U16 wChanCfg3, U16 wChanCfg4, U16 wTrigCtrl)

Visual Basic

UD_AI_2401_Config (ByVal ModuleNum As Integer, ByVal wChanCfg1 As Integer, ByVal wChanCfg2 As Integer, ByVal wChanCfg3 As Integer, ByVal wChanCfg4 As Integer, ByVal wTrigCtrl As Integer) As Integer

@ Parameter

ModuleNum: The id of the module that wants to perform this operation.

wChanCfg1:
wChanCfg2:
wChanCfg3:
wChanCfg4: The settings for input-type. The valid settings are:

- P2401_Voltage_2D5V_Above: Voltage input (> 2.5V).
- P2401_Voltage_2D5V_Below: Voltage input (<= 2.5V)
- P2401_Current: Current input
- P2401_RTD_4_Wire: 4-wire RTD type input.
- P2401_RTD_3_Wire: 3-wire RTD type input.
- P2401_RTD_2_Wire: 2-wire RTD type input.
- P2401_Resistor: Resistance type input.
- P2401_ThermoCouple: Thermo couple input.
- P2401_Full_Bridge: Full-bridge input.
- P2401_Half_Bridge: Half-bridge input.
- P2401_ThermoCouple_Differential: Thermo-couple differential input. (for USB-2401 A3 and newer devices)
- P2401_350Ohm_Full_Bridge: Full-bridge input with 350Ω resistor.
- P2401_350Ohm_Half_Bridge: Half-bridge input with 350Ω resistor.
- P2401_120Ohm_Full_Bridge: Full-bridge input with 120Ω resistor.
- P2401_120Ohm_Half_Bridge: Half-bridge input with 120Ω resistor.

wTrigCtrl: The settings for trigger-source, trigger-polarity, trigger-mode and re-trigger. The valid settings can be combined with OR (|) operator.

Trigger-Source:

- P2401_AI_TRGSRC_SOFT: After calling UD_AI_ContReadChannel /UD_AI_ContReadMultiChannels, the ADC is triggered by software immediately.
- P2401_AI_TRGSRC_DTRIG: Digital trigger from GPI0.

Trigger-Mode:

- P2401_AI_TRGMOD_POST: Post-trigger.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorInvalidCardType: Indicates the module-type is not USB_2401.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: The surprised-removal had been reported to the specific module.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidInputSignal: Indicates the invalid input-type is assigned in wChanCfg1 ~ wChanCfg4.
ErrorInvalidTriggerType: Indicates the trigger-source is not configured as P2401_AI_TRGSRC_SOFT or P2401_AI_TRGSRC_DTRIG.
ErrorInvalidTriggerMode: Indicates the trigger-mode is not configured as P2401_AI_TRGMOD_POST.
ErrorConfigIoctl: Failed to forward the command to driver, please call GetLastError() for detailed system-error.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorContIoActive: The continuous AI operation is still running.

2.2.3 UD_AI_2401_PollConfig

@ Description

Configures the speed and moving-average of polling operation. After calling this function, the FPGA moving-average will be terminated immediately, and will be restarted when calling UD_AI_ReadMultiChannels().

@ Modules Support

USB-2401

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_2401_PollConfig (U16 ModuleNum, U16 wPollSpeed, U16 wMAvgStageCh1, U16 wMAvgStageCh2, U16 wMAvgStageCh3, U16 wMAvgStageCh4)

Visual Basic

UD_AI_2401_PollConfig (ByVal ModuleNum As Integer, ByVal wPollSpeed As Integer, ByVal wMAvgStageCh1 As Integer, ByVal wMAvgStageCh2 As Integer, ByVal wMAvgStageCh3 As Integer, ByVal wMAvgStageCh4 As Integer) As Integer

@ Parameter

ModuleNum: The id of the module that wants to perform this operation.

wPollSpeed: The sampling rate in ADC. The valid settings are:

P2401_ADC_2000_SPS: 2000 samples/s
P2401_ADC_1000_SPS: 1000 samples/s.
P2401_ADC_640_SPS: 640 samples/s.
P2401_ADC_320_SPS: 320 samples/s.
P2401_ADC_160_SPS: 160 samples/s.
P2401_ADC_80_SPS: 80 samples/s.
P2401_ADC_40_SPS: 40 samples/s.
P2401_ADC_20_SPS: 20 samples/s.

wMAvgStageCh1:
wMAvgStageCh2:
wMAvgStageCh3:
wMAvgStageCh4 :Configures the moving-average stage in FPGA. The valid settings are:

P2401_Polling_MAvg_Disable: Disable the moving-average.
P2401_Polling_MAvg_2_Samples: 2-samples moving-average.
P2401_Polling_MAvg_4_Samples: 4-samples moving-average.
P2401_Polling_MAvg_8_Samples: 8-samples moving-average.
P2401_Polling_MAvg_16_Samples: 16-samples moving-average.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorInvalidCardType: Indicates the module-type is not USB_2401.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: The surprised-removal had been reported to the specific module.
ErrorFuncNotSupport: The AI function is not supported.

ErrorInvalidSampleRate: Indicates the invalid sampling-rate is assigned to wPollSpeed.
ErrorInvalidParamSetting: Indicates the invalid moving-average is not configured in wMAvgStageCh1 ~ wMAvgStageCh4.
ErrorConfigIoctl: Failed to forward the command to driver, please call GetLastError() for detailed system-error.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorContIoActive: The continuous AI operation is still running.

2.2.4 UD_AI_2405_Chan_Config

@ Description

Informs UD-DASK library of the analog input-type selected for the USB-2405 module with module ID *ModuleNum*. You must call this function before calling function to perform continuous analog input operation.

@ Modules Support

USB-2405

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_2405_Chan_Config (U16 ModuleNum, U16 wChanCfg1, U16 wChanCfg2, U16 wChanCfg3, U16 wChanCfg4)

Visual Basic

UD_AI_2405_Chan_Config (ByVal ModuleNum As Integer, ByVal wChanCfg1 As Integer, ByVal wChanCfg2 As Integer, ByVal wChanCfg3 As Integer, ByVal wChanCfg4 As Integer) As Integer

@ Parameter

ModuleNum: The id of the module that wants to perform this operation.
wChanCfg1:
wChanCfg2:
wChanCfg3:
wChanCfg4: The configuration contains three parts – input-type, couple-type and IEPE setting. Please bitwise-OR the revelant settings. The valid settings are listed as follows:

The valid settings for **input-type** are:
P2405_AI_Differential or P2405_AI_PseudoDifferential
The valid **couple-type** settings are:
P2405_AI_Coupling_AC or P2405_AI_Coupling_None.
The valid **IEPE** settings are:
P2405_AI_EnableIEPE or P2405_AI_DisableIEPE

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorInvalidCardType: Indicates the module-type is not USB_2405.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: The surprised-removal had been reported to the specific module.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidInputSignal: Indicates the invalid input-type is assigned in wChanCfg1 ~ wChanCfg4.
ErrorConfigIoctl: Failed to forward the command to driver, please call GetLastError() for detailed system-error.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorContIoActive: The continuous AI operation is still running.

2.2.5 UD_AI_2405_Trig_Config

@ Description

Informs UD-DASK library of the trigger settings for the USB-2405 module with module ID *ModuleNum*. You must call this function before calling function to perform continuous analog input operation.

@ Modules Support

USB-2405

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_2405_Trig_Config (U16 ModuleNum, U16 wConvSrc, U16 wTrigMode, U16 wTrigCtrl, U32 dwReTrigCnt, U32 dwDLY1Cnt, U32 dwDLY2Cnt, U32 dwTrgLevel)

Visual Basic

UD_AI_2405_Trig_Config (ByVal ModuleNum As Integer, ByVal wConvSrc As Integer, ByVal wTrigMode As Integer, ByVal wTrigCtrl As Integer, ByVal dwReTrigCnt As Long, ByVal dwDLY1Cnt As Long, ByVal dwDLY2Cnt As Long, ByVal dwTrgLevel As Long) As Integer

@ Parameter

ModuleNum: The id of the module that wants to perform this operation.

wConvSrc: The conversion source of AI acquisition. The valid settings are:
P2405_AI_CONVSRC_INT: on-board programmable pacer
P2405_AI_CONVSRC_EXT: external strobe from GPI1.

Note: To activate external-strobe, the GPI1 must be configured as P2405_DIGITAL_INPUT with the *UD_DIO_2405_Config()*.

Trigger-Mode:: P2405_AI_TRGMOD_POST: Post-trigger.
P2405_AI_TRGMOD_DELAY: Delay-trigger.
P2405_AI_TRGMOD_PRE: Pre-trigger.
P2405_AI_TRGMOD_MIDDLE: Middle-trigger.
P2405_AI_TRGMOD_GATED: Gated-trigger.

wTrigCtrl: The settings for trigger-source, trigger-polarity, trigger-mode and re-trigger. The valid settings can be combined with OR (|) operator.
Trigger-Source:
P2405_AI_TRGSRC_AI0 ~ P2405_AI_TRGSRC_AI3: Analog trigger from AI0 ~ AI3.
P2405_AI_TRGSRC_SOFT: After calling UD_AI_ContReadChannel /UD_AI_ContReadMultiChannels, the ADC is triggered by software immediately.
P2405_AI_TRGSRC_DTRIG: Digital trigger from GPI0.

Note: To activate external-clock, the GPI0 must be configured as P2405_DIGITAL_INPUT with the *UD_DIO_2405_Config()*.

Trigger-Polarity:

P2405_AI_TrgPositive:
Rising edge for P2405_AI_TRGSRC_AI0 ~ P2405_AI_TRGSRC_AI3.
P2405_AI_TrgNegative:
Falling edge for P2405_AI_TRGSRC_AI0 ~ P2405_AI_TRGSRC_AI3.
P2405_AI_Gate_PauseLow:
Pause low when trigger-mode is set as P2405_AI_TRGMOD_GATED.
P2405_AI_Gate_PauseHigh:
Pause high when trigger-mode is set as P2405_AI_TRGMOD_GATED.

Re-Trigger:
P2405_AI_EnReTigger: Enable Re-Trigger.

dwReTrigCnt: The count of re-trigger is required when the P2405_AI_EnReTigger is set in wTrigCtrl parameter.

dwDLY1Cnt: The number of samples before triggering.
dwDLY2Cnt: The number of samples after triggering.

Note: The dwDLY1Cnt and dwDLY2Cnt are the total delay-count for all selected channels, the ***NumChans*** parameter in UD_AI_ContReadMultiChannels().
For instanace, the pre-trigger mode is set with the 4096 is set to dwDLY1Cnt. If 4-channels are involed with the UD_AI_ContReadMultiChannels(), only 1024 AI data will be kept before the occurrence of trigger.

dwTrgLevel: The trigger level when trigger-source is set as P2405_AI_TRGSRC_AI0 ~ P2405_AI_TRGSRC_AI3.The onboard circuit will use this setting to compare the ADC data.
Please refer the AdRange parameter in UD_AI_ContReadChannel() to translate the expected trigger voltage to trigger-level.
The translation formula is:
ADC maximum number * (expected trigger-voltage / maximum voltage of selected AdRange).
For instance, if AD_B_10_V is selected and the expected trigger-voltage is 5V, the trigger-level is 8388607 * (5 / 10) = 4194303 = 0x3FFFFF.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorInvalidCardType: Indicates the module-type is not USB_2405.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: The surprised-removal had been reported to the specific module.
ErrorFuncNotSupport: The AI function is not supported.

ErrorInvalidTriggerType: Indicates the invalid trigger-source is assigned.
ErrorInvalidTriggerMode: Indicates the invalid trigger-mode is assigned.
ErrorConfigIoctl: Failed to forward the command to driver, please call GetLastError() for detailed system-error.
ErrorInvalidDelayCount: Invalid value is assigned to ***dwDLY1Cnt*** or ***dwDLY2Cnt*** parameter.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorContIoActive: The continuous AI operation is still running.

2.2.6 UD_AI_Channel_Config

@ Description

Informs UD-DASK library of the analog input-type selected with module ID *ModuleNum*. The AI channels must be configured before calling function to perform continuous analog input operation.

@ Modules Support

USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_Channel_Config (U16 ModuleNum, U16 wChanCfg1, U16 wChanCfg2, U16 wChanCfg3, U16 wChanCfg4)

Visual Basic

UD_AI_Channel_Config (ByVal ModuleNum As Integer, ByVal wChanCfg1 As Integer, ByVal wChanCfg2 As Integer, ByVal wChanCfg3 As Integer, ByVal wChanCfg4 As Integer) As Integer

@ Parameter

ModuleNum: The id of the module that wants to perform this operation.
wChanCfg1:
wChanCfg2:
wChanCfg3:
wChanCfg4: The configuration contains three parts – input-type, couple-type and IEPE setting. Please bitwise-OR the revelant settings.

The valid settings for **input-type** are:
- UD_AI_NonRef_SingEnded
- UD_AI_SingEnded
- UD_AI_Differential
- UD_AI_PseudoDifferential

The valid **couple-type** settings are:
- UD_AI_Coupling_AC
- UD_AI_Coupling_None

The valid **IEPE** settings are:
- UD_AI_EnableIEPE
- UD_AI_DisableIEPE

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorInvalidCardType: Indicates the module-type is not supported.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: The surprised-removal had been reported to the specific module.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidInputSignal: Indicates the invalid input-type is assigned in wChanCfg1 ~ wChanCfg4.
ErrorConfigIoctl: Failed to forward the command to driver, please call GetLastError() for detailed system-error.
ErrorCardDisconnected: Indicates the USB device was disconnected.

ErrorContIoActive: The continuous AI operation is still running.

2.2.7 UD_AI_Trigger_Config

@ Description

Informs UD-DASK library of the trigger settings with module ID *ModuleNum*. The conversion-source and trigger-configuration must be set before calling function to perform continuous analog input operation.

@ Modules Support

USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_Trigger_Config (U16 ModuleNum, U16 wConvSrc, U16 wTrigMode, U16 wTrigCtrl, U32 dwReTrigCnt, U32 dwDLY1Cnt, U32 dwDLY2Cnt, U32 dwTrgLevel)

Visual Basic

UD_AI_Trigger_Config (ByVal ModuleNum As Integer, ByVal wConvSrc As Integer, ByVal wTrigMode As Integer, ByVal wTrigCtrl As Integer, ByVal dwReTrigCnt As Long, ByVal dwDLY1Cnt As Long, ByVal dwDLY2Cnt As Long, ByVal dwTrgLevel As Long) As Integer

@ Parameter

ModuleNum: The id of the module that wants to perform this operation.

wConvSrc: The conversion source of AI acquisition. The valid settings are:
UD_AI_CONVSRC_INT: on-board programmable pacer
UD_AI_CONVSRC_EXT: external strobe / external clock.

Trigger-Mode:: UD_AI_TRGMOD_POST: Post-trigger.
UD_AI_TRGMOD_DELAY: Delay-trigger.
UD_AI_TRGMOD_PRE: Pre-trigger.
UD_AI_TRGMOD_MIDDLE: Middle-trigger.
UD_AI_TRGMOD_GATED: Gated-trigger.

wTrigCtrl: The settings for trigger-source, trigger-polarity, trigger-mode and re-trigger. The valid settings can be combined with OR (|) operator.
Trigger-Source:
UD_AI_TRGSRC_AI0 ~ UD_AI_TRGSRC_AI15: Analog trigger from AI0 ~ AI15.
UD_AI_TRGSRC_SOFT: After calling UD_AI_ContReadChannel /UD_AI_ContReadMultiChannels, the ADC is triggered by software immediately.
UD_AI_TRGSRC_DTRIG: Digital trigger.
Trigger-Polarity:
UD_AI_TrigPositive:
Rising edge for UD_AI_TRGSRC_AI0 ~ UD_AI_TRGSRC_AI15.
UD_AI_TrigNegative:
Falling edge for UD_AI_TRGSRC_AI0 ~ UD_AI_TRGSRC_AI15.

UD_AI_Gate_PauseLow:
Pause low when trigger-mode is set as UD_AI_TRGMOD_GATED.
UD_AI_Gate_PauseHigh:
Pause high when trigger-mode is set as UD_AI_TRGMOD_GATED.

Re-Trigger:
UD_AI_EnReTrigger: Enable Re-Trigger.
UD_AI_DisReTrigger: Disable Re-Trigger.

dwReTrigCnt: The count of re-trigger is required when the UD_AI_EnReTigger is set in wTrigCtrl parameter.

dwDLY1Cnt: The number of samples before triggering.
dwDLY2Cnt: The number of samples after triggering.

Note: The dwDLY1Cnt and dwDLY2Cnt are the total delay-count for all selected channels, the ***NumChans*** parameter in UD_AI_ContReadMultiChannels().
For instanace, the pre-trigger mode is set with the 4096 is set to dwDLY1Cnt. If 4-channels are involed with the UD_AI_ContReadMultiChannels(), only 1024 AI data will be kept before the occurrence of trigger.
If the Trigger-Mode is set to UD_AI_TRGMOD_PRE, the ***dwDLY1Cnt*** must be equal to the ***ReadCount*** parameter of UD_AI_ContReadChannel(), UD_AI_ContReadMultiChannels(), UD_AI_ContReadChannelToFile() and UD_AI_ContReadMultiChannelsToFile()
If the Trigger-Mode is set to UD_AI_TRGMOD_MIDDLE, the (***dwDLY1Cnt*** + ***dwDLY2Cnt***) must be equal to the ***ReadCount*** parameter of UD_AI_ContReadChannel(), UD_AI_ContReadMultiChannels(), UD_AI_ContReadChannelToFile() and UD_AI_ContReadMultiChannelsToFile()

---

dwTrgLevel: The trigger level when trigger-source is set as UD_AI_TRGSRC_AI0 ~ UD_AI_TRGSRC_AI15.The onboard circuit will use this setting to compare the ADC data.
Please refer the AdRange parameter in UD_AI_ContReadChannel() to translate the expected trigger voltage to trigger-level.
The translation formula is:
ADC maximum number * (expected trigger-voltage / maximum voltage of selected AdRange).
For instance, if the ***Range*** of trigger-channel is ***AD_B_10_V***, and the expected trigger-voltage is 5V, the trigger-level for USB-1210 is (2^15 -1) * (5 / 10) = 32767 * (5 / 10) = 16383 = 0x3FFF.

@ Return Code

NoError: The function returns successfully.

ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorInvalidCardType: Indicates the module-type is not supported.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: The surprised-removal had been reported to the specific module.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidTriggerType: Indicates the invalid trigger-source is assigned.
ErrorInvalidTriggerMode: Indicates the invalid trigger-mode is assigned.
ErrorConfigIoctl: Failed to forward the command to driver, please call GetLastError() for detailed system-error.
ErrorInvalidDelayCount: Invalid value is assigned to ***dwDLY1Cnt*** or ***dwDLY2Cnt*** parameter.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorContIoActive: The continuous AI operation is still running.

2.2.8 UD_AI_AsyncCheck

@ Description

Check the current status of the asynchronous analog input operation.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_AsyncCheck (U16 ModuleNum, BOOLEAN *Stopped, U32 *AccessCnt)

Visual Basic

UD_AI_AsyncCheck (ByVal ModuleNum As Integer, Stopped As Byte, AccessCnt As Long) As Integer

@ Parameter

ModuleNum : The id of the module that wants to perform this operation.

Stopped : Whether the asynchronous analog input operation has completed. If *Stopped* = TRUE, the analog input operation has stopped. Either the number of A/D conversions indicated in the call that initiated the asynchronous analog input operation has completed or an error has occurred. If *Stopped* = FALSE, the operation is not yet complete. (constants TRUE and FALSE are defined in UsbDask.h)

AccessCnt : In the condition that the trigger acquisition mode is not used, *AccessCnt* returns the number of A/D data that has been transferred at the time calling **`UD_AI_AsyncCheck().`**

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorConflictWithSyncMode: The synchronous AI operation is conflict with this function.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorAdFifoFull: Indicates the occurrence of FIFO overrun.

2.2.9 UD_AI_AsyncClear

@ Description

Stop the asynchronous analog input operation. The A/D data will be copied to user's buffer when this function is called. The configurations of channel/triggering will be cleared as well.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_AsyncClear (U16 ModuleNum, U32 *AccessCnt)

Visual Basic

UD_AI_AsyncClear (ByVal ModuleNum As Integer, AccessCnt As Long) As Integer

@ Parameter

ModuleNum : The id of the module that wants to perform this operation.

AccessCnt : In the condition that the trigger acquisition mode is not used, *AccessCnt* returns the number of A/D data that has been transferred at the time calling **`UD_AI_AsyncClear().`**

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorFuncNotSupport: The AI function is not supported.

2.2.10 UD_AI_AsyncDblBufferHalfReady

@ Description

In asynchronous double-buffered AI, indicates the half buffer of data in circular buffer is ready.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AI_AsyncDblBufferHalfReady (U16 ModuleNum, BOOLEAN *HalfReady,
        BOOLEAN *StopFlag)
```

Visual Basic

```
UD_AI_AsyncDblBufferHalfReady(ByVal ModuleNum As Integer, HalfReady As Byte,
        StopFlag As Byte) As Integer
```

@ Parameter

ModuleNum : The id of the module that wants to perform this operation.

HalfReady : Whether the half buffer of data is available. If *HalfReady* = TRUE, you can call **`UD_AI_AsyncDblBufferTransfer()`** to copy the data to your user buffer. (constants TRUE and FALSE are defined in UsbDask.h)

StopFlag : Whether the asynchronous analog input operation has completed. If *StopFlag* = TRUE, the analog input operation has stopped. If *StopFlag* = FALSE, the operation is not yet complete. (constants TRUE and FALSE are defined in UsbDask.h)

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorConflictWithSyncMode: The synchronous AI operation is conflict with this function.
ErrorAdFifoFull: Indicates the occurrence of FIFO overrun.
ErrorCardDisconnected: Indicates the USB device was disconnected.
NoError, ErrorInvalidCardNumber, ErrorCardNotRegistered, ErrorFuncNotSupport

2.2.11 UD_AI_AsyncDblBufferMode

@ Description

Enable / disable the double-buffered data acquisition mode.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_AsyncDblBufferMode (U16 ModuleNum, BOOLEAN Enable)

Visual Basic

UD_AI_AsyncDblBufferMode (ByVal ModuleNum As Integer, ByVal Enable As Byte) As Integer

@ Parameter

ModuleNum : The id of the module that wants to perform this operation.
Enable : Whether the double-buffered mode is enabled or not.
TRUE: double-buffered mode is enabled.
FALSE: double-buffered mode is disabled.
(constants TRUE and FALSE are defined in UsbDask.h)

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorContIoActive: The AI function had not been completed. Call UD_AI_AsyncClear() to Stop AI function.

2.2.12 UD_AI_AsyncDblBufferTransfer

@ Description

Depending on the continuous AI function selected, half of the data of the circular buffer will be saved into the user buffer (if continuous AI function is: *UD_AI_ContReadChannel* and *UD_AI_ContReadMultiChannels*) or logged into a disk file (if continuous AI function is: *UD_AI_ContReadChannelToFile* and *UD_AI_ContReadMultiChannelsToFile*).
You can execute this function repeatedly to return sequential half buffers of the data.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AI_AsyncDblBufferTransfer (U16 ModuleNum, U16 *Buffer)
```

Visual Basic

```
UD_AI_AsyncDblBufferTransfer (ByVal ModuleNum As Integer, Buffer As Integer)
        As Integer
```

@ Parameter

ModuleNum : The id of the module that performs the asynchronous double-buffered operation.

Buffer : The user buffer. An array that the A/D data will be copied to. If the data will be saved into a disk file, this argument will be ignored. Please refer to Appendix C, *AI Data Format* for the data format in *Buffer or the data file*.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorFuncNotSupport: The AI function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorNotDoubleBufferMode: The AI operation is not started with double-buffered mode.

2.2.13 UD_AI_AsyncDblBufferTransfer32

@ Description

Depending on the continuous AI function selected, half of the data of the circular buffer will be saved into the user buffer (if continuous AI function is: *UD_AI_ContReadChannel* and *UD_AI_ContReadMultiChannels*) or logged into a disk file (if continuous AI function is: *UD_AI_ContReadChannelToFile* and *UD_AI_ContReadMultiChannelsToFile*).
You can execute this function repeatedly to return sequential half buffers of the data.

@ Modules Support

USB-2401/USB-2405

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AI_AsyncDblBufferTransfer32 (U16 ModuleNum, U32 *Buffer)
```

Visual Basic

```
UD_AI_AsyncDblBufferTransfer32 (ByVal ModuleNum As Integer, Buffer As Long)
        As Integer
```

@ Parameter

ModuleNum : The id of the module that performs the asynchronous double-buffered operation.

Buffer : The user buffer. An array that the A/D data will be copied to. If the data will be saved into a disk file, this argument will be ignored. Please refer to Appendix C, *AI Data Format* for the data format in *Buffer or the data file*.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorFuncNotSupport: The AI function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorNotDoubleBufferMode: The AI operation is not started with double-buffered mode.

2.2.14 UD_AI_AsyncDblBufferOverrun

@ Description

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

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AI_AsyncDblBufferOverrun (U16 ModuleNum, U16 op, U16 *overrunFlag)
```

Visual Basic

```
UD_AI_AsyncDblBufferOverrun (ByVal ModuleNum As Integer, ByVal op As Integer,
        overrunFlag As Integer) As Integer
```

@ Parameter

ModuleNum : The id of the module that performs the asynchronous double-buffered operation.

op : Check/Clear 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: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorFuncNotSupport: The AI function is not supported.

2.2.15 UD_AI_AsyncDblBufferHandled

@ Description

Notifies UD-DASK the ready buffer has been handled in user application. One related flag will be changed to indicate the overrun status of double-buffered operation.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_AsyncDblBufferHandled (U16 ModuleNum)

Visual Basic

UD_AI_AsyncDblBufferHandled (ByVal ModuleNum As Integer) As Integer

@ Parameter

ModuleNum : The id of the module that performs the asynchronous double-buffered operation.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorNotDoubleBufferMode: The AI operation is not started with double-buffered mode.

2.2.16 UD_AI_AsyncDblBufferToFile

@ Description

For double buffer mode of continuous AI, if the continuous AI function is:
Check or clears the overrun status of double-buffered analog input operation.
`AI_ContReadChannelToFile,`
`AI_ContReadMultiChannelsToFile or`
`AI_ContScanChannelsToFile`
call this function to log the data of the circular buffer into a disk file.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_AsyncDblBufferToFile (U16 ModuleNum)

Visual Basic

UD_AI_AsyncDblBufferToFile (ByVal ModuleNum As Integer) As Integer

@ Parameter

ModuleNum : The id of the module that performs the asynchronous double-buffered operation.

@ Return Code

NoError: The function returns successfully.

ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorFuncNotSupport: The AI function is not supported.

2.2.17 UD_AI_AsyncReTrigNextReady

@ Description

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

ReTrigNextReady(U16 CardNumber, BOOLEAN *Ready, BOOLEAN *StopFlag, U32 *RdyTrigCnt);

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_AsyncReTrigNextReady (U16 ModuleNum, BOOLEAN *Ready, BOOLEAN *StopFlag, U16 *RdyTrigCnt)

Visual Basic

UD_AI_AsyncReTrigNextReady (ByVal ModuleNum As Integer, Ready As Byte, StopFlag As Byte, RdyTrigCnt As Integer) As Integer

@ Parameter

ModuleNum : The id of the module that performs the asynchronous double-buffered operation.

Ready : Tells wheather the data associated with the next trigger signal is available.
Constants TRUE and FALSE are defined in USBDASK.H.

StopFlag : Tells whether the asynchronous analog input operation is completed. If StopFlag is TRUE, the analog input operation has stopped. If StopFlag is FALSE, the operation is not yet completed.
Constants TRUE and FALSE are defined in USBDASK.H.

RdyTrigCnt : This argument returns the count of trigger signal that occurred if re-trigger count is defined. If the re-trigger count is infinite, this argument returns the index of the buffer that stored the data after the most recent trigger signal trigger generated.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorInvalidTriggerMode: Indicates the Re-trigger mode is not enabled.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorDBHalfReadyIoctl: Failed to forward the command to driver, please call GetLastError() for detailed system-error.

2.2.18 UD_AI_ContReadChannel

@ Description

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

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AI_ContReadChannel (U16 ModuleNum, U16 Channel, U16 AdRange, U16
        *Buffer, U32 ReadCount, F64 SampleRate, U16 SyncMode)
```

Visual Basic

```
UD_AI_ContReadChannel (ByVal ModuleNum As Integer, ByVal Channel As Integer,
        ByVal AdRange As Integer, Buffer As Integer, ByVal ReadCount As Long,
        ByVal SampleRate As Double, ByVal SyncMode As Integer) As Integer
```

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

Channel : Analog input channel number
USB-1901/USB-1902: 0 through 15
USB-1903: 0 through 7 (differential input signal only)
USB-2401: 0 through 3
USB-2405: 0 through 3
USB-1210: 0 through 3

AdRange : The analog input range the specified channel is setting. We define some constants to represent various A/D input ranges in UsbDask.h. Please refer to the Appendix B, ***AI Range Codes***, for the valid range values.

Buffer : An array to contain the acquired data. *Buffer* must has a length equal to or greater than the value of parameter *ReadCount*. If double-buffered mode is enabled, this parameter will be ignored. Please refer to Appendix C, *AI Data Format* for the data format in *Buffer*.

ReadCount : If double-buffered mode is disabled, *ReadCount* is the number of A/D conversions to be performed. For double-buffered acquisition, *ReadCount* is the size (in samples) of the circular buffer.

Note: For USB-1901/1902/1903/1210, the value of *ReadCount* must be the multiple of 256 for non-double-buffer mode, or multiple of 512 for double-buffer mode.
For USB-2401/2405, the value of *ReadCount* must be the multiple of 128 for non-double-buffer mode, or multiple of 256 for double-buffer mode.

SampleRate : For USB-1901/1902/1903, this parameter will be ignored. Please call **`UD_AI_1902_ConterInterval()`** to set the Scan-Interval and Sample-Interval.
The valid settings are:
USB-2401: 20, 40, 80, 160, 320, 500, 1000 and 2000 samples/s.

SyncMode : Whether this operation is performed synchronously or asynchronously.
If any trigger mode is enabled by calling **`UD_AI_1902_Config()/UD_AI_2401_Config()/UD_AI_2405_Trig_Config()/UD_AI_Trigger_Config()`**, this operation should be performed *asynchronously*.
Valid values:
SYNCH_OP: synchronous A/D conversion, that is, the function does not return until the A/D operation complete.
ASYNCH_OP:asynchronous A/D conversion.

Note: When SYNCH_OP is selected, the UD_AI_SetTimeOut() can set the Timeout for synchronous operation.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidAdRange: The invalid setting is set to *AdRange*.
ErrorInvalidInputSignal: Indicates the invalid input-signal is assigned.
ErrorTransferCountTooLarge: The *ReadCount* is too large.
ErrorInvalidIoChannel: The invalid setting is set to *Channel.*
ErrorContIoNotAllowed: The continuous operation is not supported.
ErrorConflictWithSyncMode: The double-buffered AI operation is conflict with SYNCH_OP.
ErrorInvalidTransferCount: The *ReadCount* is not multiple of 256/512 (for USB-190x/1210), 128/256 (for USB-2401/2405).
ErrorInvalidTriggerMode: Neither double-buffered AI nor SYNCH_OP operation supports re-trigger feature.
ErrorInvalidSampleRate: Indicates the sampling-rate is out of range.
ErrorInvalidCounterState: Either scan-Interval or sample-Interval is zero; Sample-interval is larger than scan-interval.
ErrorSystemCallFailed: Failed to forward the command to driver, please call GetLastError() for detailed system-error.
ErrorContIoActive: The AI function had not been completed. Call UD_AI_AsyncClear() to Stop AI function.
ErrorConflictWithInfiniteOp: The infinite AI operation is only supported by double-buffered acquisition.
ErrorInvalidTriggerChannel: The analog-trigger is not the first channel in Channel-Gain-Queue. Please make sure the trigger channel is identical to the ***Channel*** parameter.
ErrorWaitingUSBHostResponse: This is usually caused by trigger-enabled AI operation. Call UD_AI_AsyncClear() to disable the waiting state.
ErrorTimeoutFromSyncMode: The synchronous AI operation is time-out.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorConflictWithAIConfig: The ***AdRange*** is conflict with the some specific input-type.

2.2.19 UD_AI_ContReadChannelToFile

@ Description

This function performs continuous A/D conversions on the specified analog input channel 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.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

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

Visual Basic

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

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

Channel : Analog input channel number
USB-1901/USB-1902: 0 through 15
USB-1903: 0 through 7 (differential input signal only)
USB-2401: 0 through 3
USB-2405: 0 through 3
USB-1210: 0 through 3

AdRange : The analog input range the specified channel is setting. We define some constants to represent various A/D input ranges in UsbDask.h. Please refer to the Appendix B, ***AI Range Codes***, for the valid range values.

FileName : Name of data file which stores the acquired data

ReadCount : If double-buffered mode is disabled, *ReadCount* is the number of A/D conversions to be performed. For double-buffered acquisition, *ReadCount* is the size (in samples) of the circular buffer.

Note: For USB-1901/1902/1903/1210, the value of *ReadCount* must be the multiple of 256 for non-double-buffer mode, or multiple of 512 for double-buffer mode.
For USB-2401/2405, the value of *ReadCount* must be the multiple of 128 for non-double-buffer mode, or multiple of 256 for double-buffer mode.

SampleRate : For USB-1901/1902/1903, this parameter will be ignored. Please call **`UD_AI_1902_ConterInterval()`** to set the Scan-Interval and Sample-Interval.
The valid settings are:
USB-2401: 20, 40, 80, 160, 320, 500, 1000 and 2000 samples/s.

SyncMode : Whether this operation is performed synchronously or asynchronously.
If any trigger mode is enabled by calling **`UD_AI_1902_Config()/UD_AI_2401_Config()/UD_AI_2405_Trig_Config()/UD_AI_Trigger_Config()`**, this operation should be performed *asynchronously*.
Valid values:
SYNCH_OP: synchronous A/D conversion, that is, the function does not return until the A/D operation complete.
ASYNCH_OP:asynchronous A/D conversion.

Note: When SYNCH_OP is selected, the UD_AI_SetTimeOut() can set the Timeout for synchronous operation.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorOpenFile: Failed to create the file to save the A/D data.
ErrorInvalidAdRange: The invalid setting is set to *AdRange*.
ErrorInvalidInputSignal: Indicates the invalid input-signal is assigned.
ErrorTransferCountTooLarge: The *ReadCount* is too large.
ErrorInvalidIoChannel: The invalid setting is set to *Channel.*
ErrorContIoNotAllowed: The continuous operation is not supported.
ErrorConflictWithSyncMode: The double-buffered AI operation is conflict with SYNCH_OP.
ErrorInvalidTransferCount: The *ReadCount* is not multiple of 256/512 (for USB-190x), 128/256 (for USB-2401).
ErrorInvalidTriggerMode: Neither double-buffered AI nor SYNCH_OP operation supports re-trigger feature.
ErrorInvalidSampleRate: Indicates the sampling-rate is out of range.
ErrorInvalidCounterState: Either scan-Interval or sample-Interval is zero; Sample-interval is larger than scan-interval.
ErrorSystemCallFailed: Failed to forward the command to driver, please call GetLastError() for detailed system-error.
ErrorContIoActive: The AI function had not been completed. Call UD_AI_AsyncClear() to Stop AI function.
ErrorConflictWithInfiniteOp: The infinite AI operation is only supported with the double-buffered mode.
ErrorInvalidTriggerChannel: The analog-trigger is not the first channel in Channel-Gain-Queue. Please make sure the trigger channel is identical to the ***Channel*** parameter.
ErrorWaitingUSBHostResponse: This is usually caused by trigger-enabled AI operation. Call UD_AI_AsyncClear() to disable the waiting state.
ErrorTimeoutFromSyncMode: The synchronous AI operation is time-out.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorConflictWithAIConfig: The ***AdRange*** is conflict with the some specific input-type.

2.2.20 UD_AI_ContReadMultiChannels

@ Description

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

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_ContReadMultiChannels (U16 ModuleNum, U16 numChans, U16 *Chans, U16 *AdRanges, U16 *Buffer, U32 ReadCount, F32 SampleRate, U16 SyncMode)

Visual Basic

UD_AI_ContReadMultiChannels (ByVal ModuleNum As Integer, ByVal numChans As Integer, Chans As Integer, AdRanges As Integer, Buffer As Integer, ByVal ReadCount As Long, ByVal SampleRate As Single, ByVal SyncMode As Integer) As Integer

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

numChans : The number of analog input channels in the array *Chans*. The maximum value:
USB-1901/USB-1902/USB-1903: 256
USB-2401: 4
USB-2405: 4
USB-1210: 4

Chans : Array of analog input channel numbers. The channel order for acquiring data is the same as the order you set in Chans.
USB-1901/USB-1902: 0 through 15
USB-1903: 0 through 7 (differential input signal only)
Since there is no restriction of channel order setting, you can set the channel order as you wish.
USB-2401: 0 through 3
USB-2405: 0 through 3
USB-1210: 0 through 4

AdRanges : An integer array of length *numChans* that contains the analog input range for every channel in array *Chans*.
Please refer to the Appendix B for the valid range values.

Buffer : An integer array to contain the acquired data. The length of *Buffer* must be equal to or greater than the value of parameter *ReadCount*. The A/D data is stored in interleaved sequence. For example, if the value of *numChans* is 3, and the numbers in *Chans* are 3, 8, and 0. Then this function input data from channel 3, then channel 8, then channel 0, then channel 3, then channel 8, ... The data acquired is put to *Buffer* by order. So the data read from channel 3 is stored in *Buffer*[0], *Buffer*[3], *Buffer*[6], ... The data from channel 8 is stored in *Buffer*[1], *Buffer*[4], *Buffer*[7], ... The data from channel 0 is stored in *Buffer*[2], *Buffer*[5], *Buffer*[8], ... If double-buffered mode is enabled,

this buffer is of no use, you can ignore this argument. Please refer to Appendix C, *AI Data Format* for the data format in *Buffer*.

ReadCount : If double-buffered mode is disabled, *ReadCount* is the number of A/D conversions to be performed. For double-buffered acquisition, *ReadCount* is the size (in samples) of the circular buffer.

Note: For USB-1901/1902/1903/1210, the value of *ReadCount* must be the multiple of 256 for non-double-buffer mode, or multiple of 512 for double-buffer mode.
For USB-2401/2405, the value of *ReadCount* must be the multiple of 128 for non-double-buffer mode, or multiple of 256 for double-buffer mode.

SampleRate : For USB-1901/1902/1903, this parameter will be ignored. Please call **`UD_AI_1902_ConterInterval()`** to set the Scan-Interval and Sample-Interval.
For USU-2401, only 20, 40, 80, 160, 320, 500, 1000 and 2000 (samples/s) are valid.

SyncMode : Whether this operation is performed synchronously or asynchronously. If any trigger mode is enabled by calling **`UD_AI_1902_Config()/UD_AI_2401_Config()/UD_AI_2405_Trig_Config()/UD_AI_Trigger_Config()`**, this operation should be performed *asynchronously*.
Valid values:
SYNCH_OP: synchronous A/D conversion, that is, the function does not return until the A/D operation complete.
ASYNCH_OP:asynchronous A/D conversion.

Note: When SYNCH_OP is selected, the UD_AI_SetTimeOut() can set the Timeout for synchronous operation.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidAdRange: The invalid setting is set to *AdRange*.
ErrorInvalidInputSignal: Indicates the invalid input-signal is assigned.
ErrorTransferCountTooLarge: The *ReadCount* is too large.
ErrorChanGainQueueTooLarge: The *numChans* is too large.
ErrorInvalidIoChannel: The invalid setting is set to *Channel.*
ErrorContIoNotAllowed: The continuous operation is not supported.
ErrorConflictWithSyncMode: The double-buffered AI operation is conflict with SYNCH_OP.
ErrorInvalidTransferCount: The *ReadCount* is not multiple of 256/512 (for USB-190x/1210), 128/256 (for USB-2401/2405).
ErrorInvalidTriggerMode: Neither double-buffered AI nor SYNCH_OP operation supports re-trigger feature.
ErrorInvalidSampleRate: Indicates the sampling-rate is out of range.

ErrorInvalidCounterState: Either scan-Interval or sample-Interval is zero; Scan-Interval is less than (sample-interval x *NumChans*)
ErrorSystemCallFailed: Failed to forward the command to driver, please call GetLastError() for detailed system-error.
ErrorContIoActive: The AI function had not been completed. Call UD_AI_AsyncClear() to Stop AI function.
ErrorConflictWithInfiniteOp: The infinite AI operation is only supported by double-buffered acquisition.
ErrorInvalidTriggerChannel: The analog-trigger is not the first channel in Channel-Gain-Queue. Please make sure the trigger channel is identical to the first channel in ***Channel*** parameter.
ErrorWaitingUSBHostResponse: This is usually caused by trigger-enabled AI operation. Call UD_AI_AsyncClear() to disable the waiting state.
ErrorChanGainQueueTooLarge: The *numChans* excesses the supported channel-gain-queue.
ErrorTimeoutFromSyncMode: The synchronous AI operation is time-out.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorConflictWithAIConfig: The ***AdRange*** is conflict with the some specific input-type.

2.2.21 UD_AI_ContReadMultiChannelsToFile

@ Description

This function performs continuous A/D conversions on the specified analog input channels 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 USB-1902 channel-gain queue functionality to perform multi-channel analog input.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

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

Visual Basic

UD_AI_ContReadMultiChannelsToFile (ByVal ModuleNum As Integer, ByVal numChans As Integer, Chans As Integer, AdRanges As Integer, ByVal FileName As String, ByVal ReadCount As Long, ByVal SampleRate As Double, ByVal SyncMode As Integer) As Integer

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

numChans : The number of analog input channels in the array *Chans*. The maximum value:
USB-1901/USB-1902/USB-1903: 256
USB-2401: 4
USB-2405: 4
USB-1210: 4

Chans : Array of analog input channel numbers. The channel order for acquiring data is the same as the order you set in Chans.
USB-1901/USB-1902: 0 through 15
USB-1903: 0 through 7 (differential input signal only)
Since there is no restriction of channel order setting, you can set the channel order as you wish.
USB-2401: 0 through 3
USB-2405: 0 through 3
USB-1210: 0 through 3

AdRanges : An integer array of length *numChans* that contains the analog input range for every channel in array *Chans*.
Please refer to the Appendix B for the valid range values.

FileName : Name of data file which stores the acquired data

ReadCount : If double-buffered mode is disabled, *ReadCount* is the number of A/D conversions to be performed. For double-buffered acquisition, *ReadCount* is the size (in samples) of the circular buffer.

Note: For USB-1901/1902/1903, the value of *ReadCount* must be the multiple of 256 for non-double-buffer mode, or multiple of 512 for double-buffer mode.
For USB-2401/2405, the value of *ReadCount* must be the multiple of 128 for non-double-buffer mode, or multiple of 256 for double-buffer mode.

SampleRate : For USB-1901/1902/1903, this parameter will be ignored. Please call **`UD_AI_1902_ConterInterval()`** to set the Scan-Interval and Sample-Interval.
For USU-2401, only 20, 40, 80, 160, 320, 500, 1000 and 2000 (samples/s) are valid.

SyncMode : Whether this operation is performed synchronously or asynchronously.
If any trigger mode is enabled by calling **`UD_AI_1902_Config()/UD_AI_2401_Config()/UD_AI_2405_Trig_Config()/UD_AI_Trigger_Config()`**, this operation should be performed *asynchronously*.
Valid values:
SYNCH_OP: synchronous A/D conversion, that is, the function does not return until the A/D operation complete.
ASYNCH_OP:asynchronous A/D conversion.

Note: When SYNCH_OP is selected, the UD_AI_SetTimeOut() can set the Timeout for synchronous operation.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorOpenFile: Failed to create the file to save the A/D data.
ErrorInvalidAdRange: The invalid setting is set to *AdRange*.
ErrorInvalidInputSignal: Indicates the invalid input-signal is assigned.
ErrorTransferCountTooLarge: The *ReadCount* is too large.
ErrorChanGainQueueTooLarge: The *numChans* is too large.
ErrorInvalidIoChannel: The invalid setting is set to *Channel.*
ErrorContIoNotAllowed: The continuous operation is not supported.
ErrorConflictWithSyncMode: The double-buffered AI operation is conflict with SYNCH_OP.
ErrorInvalidTransferCount: The *ReadCount* is not multiple of 256/512 (for USB-190x/1210), 128/256 (for USB-2401/2405).
ErrorInvalidTriggerMode: Neither double-buffered AI nor SYNCH_OP operation supports re-trigger feature.
ErrorInvalidSampleRate: Indicates the sampling-rate is out of range.
ErrorInvalidCounterState: Either scan-Interval or sample-Interval is zero; Scan-Interval is less than (sample-interval x *NumChans*)
ErrorSystemCallFailed: Failed to forward the command to driver, please call GetLastError() for detailed system-error.
ErrorContIoActive: The AI function had not been completed. Call UD_AI_AsyncClear() to Stop AI function.

ErrorConflictWithInfiniteOp: The infinite AI operation is only supported by double-buffered acquisition.
ErrorInvalidTriggerChannel: The analog-trigger is not the first channel in Channel-Gain-Queue. Please make sure the trigger channel is identical to the first channel in ***Channel*** parameter.
ErrorWaitingUSBHostResponse: This is usually caused by trigger-enabled AI operation. Call UD_AI_AsyncClear() to disable the waiting state.
ErrorChanGainQueueTooLarge: The *numChans* excesses the supported channel-gain-queue.
ErrorTimeoutFromSyncMode: The synchronous AI operation is time-out.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorConflictWithAIConfig: The ***AdRange*** is conflict with the some specific input-type.

2.2.22 UD_AI_VoltScale

@ Description

Converts the result from an UD_AI_ReadChannel() call to actual input voltage.

@ Modules Support

USB-1901/USB-1902/USB-1903/Usb-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_VoltScale (U16 ModuleNum, U16 AdRange, U16 reading, F64 *voltage)

Visual Basic

UD_AI_VoltScale (ByVal ModuleNum As Integer, ByVal AdRange As Integer, ByVal reading As Integer, voltage As Double) As Integer

@ Parameter

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

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorInvalidAdRange: The invalid setting is set to *AdRange*.

2.2.23 UD_AI_VoltScale32

@ Description

Converts the result from an UD_AI_ReadChannel() call to actual input voltage.

@ Modules Support

USB-2405

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AI_VoltScale32 (U16 ModuleNum, U16 AdRange, U16 inType, U32 reading,
        F64 *voltage)
```

Visual Basic

```
UD_AI_VoltScale (ByVal ModuleNum As Integer, ByVal AdRange As Integer, ByVal
        inType As Integer, ByVal reading As Long, voltage As Double) As Integer
```

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
AdRange :	The analog input range the specified channel is setting. Please refer to the Appendix B for the valid range values.
inType :	Reserved for future use.
reading :	Result of AD Conversion.
voltage :	Computed voltage value

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorInvalidAdRange: The invalid setting is set to *AdRange*.

2.2.24 UD_AI_2401_Scale32

@ Description

Converts the result from an UD_AI_ReadChannel() call to actual input voltage/current/resistance.

@ Modules Support

USB-2401

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AI_2401_Scale32 (U16 ModuleNum, U16 AdRange, U16 inType, U32
        reading, F64 *scaledValue)
```

Visual Basic

```
UD_AI_2401_Scale32 (ByVal ModuleNum As Integer, ByVal AdRange As Integer,
        ByVal inType As Integer ByVal reading As Long, scaledValue As Double)
        As Integer
```

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

AdRange : The analog input range the specified channel. The valid ranges of P2401_Voltage_2D5V_Above/ P2401_Voltage_2D5V_Below are : AD_B_25_V, AD_B_12_5_V, AD_B_2_5_V and AD_B_0_3125_V. For other input-type, any valid range is required and the specific range will be assigned automatically.

inType: The settings for input-type. The valid settings are:
- P2401_Voltage_2D5V_Above: Voltage input (> 2.5V).
- P2401_Voltage_2D5V_Below: Voltage input (<= 2.5V)
- P2401_Current: Current input
- P2401_RTD_4_Wire: 4-wire RTD type input.
- P2401_RTD_3_Wire: 3-wire RTD type input.
- P2401_RTD_2_Wire: 2-wire RTD type input.
- P2401_Resistor: Resistance type input.
- P2401_ThermoCouple: Thermo couple input.
- P2401_Full_Bridge: Full-bridge input.
- P2401_Half_Bridge: Half-bridge input.

reading : Result of AD Conversion.

scaledValue : Computed value.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorInvalidAdRange: The invalid setting is set to *AdRange*.

2.2.25 UD_AI_ContVScale

@ Description

This function converts the continuous acquisition data of single channel to the actual input voltages. The raw data is returned from the continuous A/D conversion call, say UD_AI_ContReadChannel. (The multiple-channels raw data, returned from UD_AI_ContReadMultiChannels, must be splitted by channels).

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_ContVScale (U16 ModuleNum, U16 AdRange, U16 *readingArray, F64 *voltageArray, I32 count)

Visual Basic

UD_AI_ContVScale (ByVal ModuleNum As Integer, ByVal AdRange As Integer, readingArray As Integer, voltageArray As Double, ByVal count As Long) As Integer

@ Parameter

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

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorInvalidAdRange: The invalid setting is set to *AdRange*.

2.2.26 UD_AI_ContVScale32

@ Description

This function converts the continuous acquisition data of single channel to the actual input voltages. The raw data is returned from the continuous A/D conversion call, say UD_AI_ContReadChannel. (The multiple-channels raw data, returned from UD_AI_ContReadMultiChannels, must be splitted by channels).

@ Modules Support

USB-2405

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AI_ContVScale32 (U16 ModuleNum, U16 AdRange, U16 inType, U32
        *readingArray, F64 *voltageArray, I32 count)
```

Visual Basic

```
UD_AI_ContVScale (ByVal ModuleNum As Integer, ByVal AdRange As Integer,
        ByVal inType As Integer readingArray As Long, voltageArray As Double,
        ByVal count As Long) As Integer
```

@ Parameter

ModuleNum : The id of the module that want to perform this operation.
AdRange : The analog input range the continuous specified channel is setting. Please refer to the Appendix B for the valid range values.
inType: Reserved for future use.
readingArray : Acquired continuous analog input data array
voltageArray : Computed voltages array returned.
count : The length of readingArray array.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorInvalidAdRange: The invalid setting is set to *AdRange*.

2.2.27 UD_AI_2401_ContVScale32

@ Description

This function converts the continuous acquisition data of single channel to the actual input voltages. The raw data is returned from the continuous A/D conversion call, say UD_AI_ContReadChannel. (The multiple-channels raw data, returned from UD_AI_ContReadMultiChannels, must be splitted by channels).

@ Modules Support

USB-2401

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_2401_ContVScale32 (U16 ModuleNum, U16 AdRange, U16 inType, U32 *readingArray, F64 *ScaledArray, I32 count)

Visual Basic

UD_AI_2401_ContVScale32 (ByVal ModuleNum As Integer, ByVal AdRange As Integer, ByVal inType As Integer, readingArray As Long, ScaledArray As Double, ByVal count As Long) As Integer

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

AdRange : The analog input range the specified channel. The invalid ranges of P2401_Voltage_2D5V_Above/ P2401_Voltage_2D5V_Below are : AD_B_25_V, AD_B_12_5_V, AD_B_2_5_V and AD_B_0_3125_V. For other input-type, any valid range is required and the specific range will be assigned automatically.

inType: The settings for input-type. The valid settings are:
- P2401_Voltage_2D5V_Above: Voltage input (> 2.5V).
- P2401_Voltage_2D5V_Below: Voltage input (<= 2.5V)
- P2401_Current: Current input
- P2401_RTD_4_Wire: 4-wire RTD type input.
- P2401_RTD_3_Wire: 3-wire RTD type input.
- P2401_RTD_2_Wire: 2-wire RTD type input.
- P2401_Resistor: Resistance type input.
- P2401_ThermoCouple: Thermo couple input.
- P2401_Full_Bridge: Full-bridge input.
- P2401_Half_Bridge: Half-bridge input.

readingArray : Acquired continuous analog input data array

scaledArray : computed voltages/currents/resistances array returned

count : The length of readingArray array.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorInvalidAdRange: The invalid setting is set to *AdRange*.

2.2.28 UD_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.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AI_InitialMemoryAllocated (U16 ModuleNum, U32 MemSize)
```

Visual Basic

```
UD_AI_InitialMemoryAllocated (ByVal ModuleNum As Integer, MemSize As Long)
        As Integer
```

@ Parameter

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

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorFuncNotSupport: The AI function is not supported.

2.2.29 UD_AI_ReadChannel

@ Description

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

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AI_ReadChannel (U16 ModuleNum, U16 Channel, U16 AdRange, U16
        *Value)
```

Visual Basic

```
UD_AI_ReadChannel (ByVal ModuleNum As Integer, ByVal Channel As Integer,
        ByVal AdRange As Integer, Value As Integer) As Integer
```

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
Channel :	Analog input channel number USB-1901/USB-1902: 0 through 15 USB-1903: 0 through 7 (differential input signal only) USB-2405: 0 through 3 USB-1210: 0 through 3
AdRange :	The analog input range the specified channel is setting. We define some constants to represent various A/D input ranges in UsbDask.h. Please refer to the Appendix B, ***AI Range Codes***, for the valid range values.
Value :	The memory to store the A/D converted data. Please refer to Appendix C, *AI Data Format* for the data format.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidIoChannel: The invalid setting is set to *Channel.*
ErrorInvalidAdRange: The invalid setting is set to *AdRange*.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorInvalidInputSignal: Indicates the invalid input-signal is assigned.

2.2.30 UD_AI_1902_CounterInterval

@ Description

This function configures the scan-interval / sample-interval for USB-1902 series modules. Based on the conversion clock, the these two settings determine the interval between samples and scans. For instance, when internal conversion-clock, 80MHz, is selected, UD-AI_1902_CounterInterval(ModuleNum, 8000, 320) determined
Scan-Interval = 8,000 / 80,000,000 = 1 / 10,000 = 10KHz
Sample-Interval = 320 / 80,000,000 = 1 / 250,000 = 250KHz

Note: The sample-interval determines the interval in each scan. And in every scan, the A/D conversion will applied to all AI channels that are configured in Channel-Gain queue. Therefore, the Scan-Interval <= (number of Chan-Gain-Queue * Sample-Interval).

@ Modules Support

USB-1901/USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AI_1902_CounterInterval(U16 ModuleNum, U32 ScanIntrv, U32 SampIntrv)
```

Visual Basic

```
UD_AI_1902_CounterInterval(ByVal ModuleNum As Integer, ByVal ScanIntrv As
        Long, ByVal SampIntrv As Long) As Integer
```

@ Parameter

ModuleNum : The id of the module that want to perform this operation.
ScanIntrv : The interval between scans. The ScanIntrv must be larger than SampIntrv.
SampIntrv : The interval between sample in each scan. The valid value is: USB-1901/USB-1902//USB-1903 : 320

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorInvalidCounterValue: Invalid value is assigned to either ScanIntrv or SampIntrv; SampIntrv is larger than ScanIntrv.

2.2.31 UD_AI_DDS_ActualRate_Get

@ Description

This function read the actual sampling-rate for the moduels that use the DDS (Direct Digital Synthesizer) clock system.

@ Modules Support

USB-2405

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AI_DDS_ActualRate_Get(U16 ModuleNum, F64 fSampleRate, F64
        *pActualRate)
```

Visual Basic

```
UD_ DDS_ActualRate_Get (ByVal ModuleNum As Integer, ByVal fSampleRate As
        Double, pActualRate As Double) As Integer
```

@ Parameter

ModuleNum : The id of the module that want to perform this operation.
fSampleRate : The expected sampling-rate.
pActualRate : The memory that is stored the actual sampling-rate

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorInvalidSampleRate: Indicates the expected sampling-rate is invalid.
ErrorFuncNotSupport: The AI function is not supported.

2.2.32 UD_AI_SetTimeOut

@ Description

This function sets the timeout for synchronous AI operation (SYNCH_OP).

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_SetTimeOut (U16 ModuleNum, U32 dwTimeOut)

Visual Basic

UD_AI_SetTimeOut (ByVal ModuleNum As Integer, ByVal dwTimeOut As Long) As Integer

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

dwTimeOut : The Timeout for synchronous operation, in millisecond.
This setting is applied to WaitForSingleObject(). However, if this parameter is set to zero, the INFINITE is applied to WaitForSingleObject().

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorFuncNotSupport: The AI function is not supported.

2.2.33 UD_AI_ReadMultiChannels

@ Description

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

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_ReadMultiChannels (U16 ModuleNum, U16 NumChans, U16* Chans, U16* AdRangse, U16 *Buffer)

Visual Basic

UD_AI_ReadMultiChannels (ByVal ModuleNum As Integer, ByVal NumChans As Integer, Chans As Integer, AdRanges As Integer, Buffer As Integer) As Integer

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

NumChans : The number of analog input channels in the array Chans. Valid values:
USB-1901/USB-1902: 1 through 16
USB-1903: 1 through 8 (differential input signal only)
USB-2401: 1 through 4
USB-2405: 1 through 4
USB-1210: 1 through 4

Chans : Array of analog input channel numbers. The channel order for acquiring data is the same as the order you set in Chans.
USB-1901/USB-1902: Numbers in ***Chans*** must be within 0 and 15. Since there is no restriction for channel order setting, you may set the channel order as you want.
USB-1903: Numbers in ***Chans*** must be within 0 and 7. Since there is no restriction for channel order setting, you may set the channel order as you want.
USB-2401: Numbers in ***Chans*** must be within 0 and 3.
USB-2405: Numbers in ***Chans*** must be within 0 and 3.
USB-1210: Numbers in ***Chans*** must be withen 0 and 3;

AdRanges : An integer array of length ***numChans*** that contains the analog input range for every channel in array ***Chans***. Please refer to the Appendix B, ***AI Range Codes***, for the valid range values.

Buffer : An Integer array to contain the acquired data. The length of Buffer must be equal to or greater than the value of parameter ***NumChans***. The acquired data is stored in interleaved sequence. For example, if the value of ***numChans*** is 3, and the numbers in ***Chans*** are 3, 8 and 0, then this function input data from channel 3, then channel 8, then channel 0. The data acquired is put to ***Buffer*** by order, so the data read from channel 3 is stored in ***Buffer[0]***, the data read from channel 8 is stored in ***Buffer[1],*** and the data read from channel 0 is stored in ***Buffer[2]***.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidIoChannel: The invalid setting is set to *Chans.*
ErrorInvalidAdRange: The invalid setting is set to *AdRanges*.
ErrorInvalidInputSignal: Indicates the invalid input-signal is assigned.
ErrorContIoActive: The AI function had not been completed. Call UD_AI_AsyncClear() to Stop AI function.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorConflictWithAIConfig: The ***AdRange*** is conflict with the some specific input-type.

2.2.34 UD_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.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AI_VReadChannel (U16 ModuleNum, U16 Channel, U16 AdRange, F64
        *voltage)
```

Visual Basic

```
UD_AI_ReadChannel (ByVal ModuleNum As Integer, ByVal Channel As Integer,
        ByVal AdRange As Integer, voltage As Double) As Integer
```

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
Channel :	Analog input channel number USB-1901/USB-1902: 0 through 15 USB-1903: 0 through 7 (differential input signal only) USB-2405: 0 through 3 USB-1210: 0 through 3
AdRange :	The analog input range the specified channel is setting. We define some constants to represent various A/D input ranges in UsbDask.h. Please refer to the Appendix B, ***AI Range Codes***, for the valid range values.
voltage :	The measured voltage value returned and scaled to units of voltage.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidIoChannel: The invalid setting is set to *Channel.*
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorInvalidAdRange: The invalid setting is set to *AdRange*.

2.2.35 UD_AI_Moving_Average32

@ Description

This function performs the software moving-average for 32-bit data.. The ***SrcBuf*** contains multiple-channels ADC data. Only single-channel ADC data will be extracted and calculated; that specific channel is indicated with ***dwTgChIdx*** parameter.

@ Modules Support

USB-2401

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AI_Moving_Average32 (U16 ModuleNum, U32 *SrcBuf, U32 *DesBuf, U32
        dwTgChIdx, U32 dwTotalCh, U32 dwMovAvgWindow, U32 dwSamplCnt)
```

Visual Basic

```
UD_AI_Moving_Average32 (ByVal ModuleNum As Integer, SrcBuf As Long, DesBuf
        As Long, ByVal dwTgChIdx As Long, ByVal dwTotalCh As Long, ByVal
        dwMovAvgWindow As Long, ByVal dwSamplCnt As Long) As Integer
```

@ Parameter

ModuleNum : This parameter is reserved for future.
SrcBuf : The buffer that contains the data to be calculated.
DesBuf : The user-provided buffer to save the data with moving-average calculation.
dwTgChIdx : The index of target-channel
dwTotalCh : The total channels that are related to the ADC data in the SrcBuf.
dwMovAvgWindow :The number of samples will be applied to moving-average operation.
dwSamplCnt : The all number of samples will be involved in this operation.

@ Return Code

NoError: The function returns successfully.
ErrorNullPoint: Either SrcBuf or DesBuf is NULL.
ErrorInvalidChannel: Indicates either the ***dwTgChIdx*** is larger than ***dwTotalCh***, or ***dwTotalCh*** is zero.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorInvalidParamSetting: The dwMovAvgWindow is larger than dwSamplCnt.

2.2.36 UD_AI_EventCallBack (Win32 Only)

@ Description

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

@ Modules Support

USB-2405

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_EventCallBack (U16 ModuleNum, I16 mode, I16 EventType, U32 callbackAddr)

Visual Basic

UD_AI_EventCallBack (ByVal ModuleNum As Integer, ByVal mode As Integer, ByVal EventType As Integer, ByVal callbackAddr As Long) As Integer

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

Mode : Add or remove the event message. The valid settings are :
0: remove
1: add

EventType : Event criteria. The valide settings are:
AIEnd: Notification that the asychronus analog input
DBEvent: Notification that the next half buffer of data in circular buffer is ready for transfer.

callbackAddr : Address of the user callback function. The UD-DASKT calls this function when the specific event occurs. If you want remove the event message, set callbackAddr to 0.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorInvalidEventHandle: The invalid setting is set to ***EventType***.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidIoChannel: The invalid setting is set to ***Channel.***
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.37 UD_AO_1902_Config

@ Description

Configures the AO operation of USB-1902/USB-1903, including conversion control and trigger settings

@ Modules Support

USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AO_1902_Config (U16 ModuleNum, U16 wConfigCtrl, U16 wTrigCtrl, U32
        dwReTriggerCnt, U32 dwDLY1Cnt, U32 dwDLY2Cnt)
```

Visual Basic

```
UD_AO_1902_Config (ByVal ModuleNum As Integer, ByVal wConfigCtrl As Integer,
        ByVal wTrigCtrl As Integer, ByVal dwReTriggerCnt As Long, ByVal
        dwDLY1Cnt As Long, ByVal dwDLY2Cnt As Long) As Integer
```

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

wConfigCtrl : Now only P1902_AO_CONVSRC_INT is supported.

wTrigCtrl : The settings for trigger-source, trigger-polarity, trigger-mode, re-trigger and waveform separation. The valid settings can be combined with OR (|) operator.

Trigger-Source:
- P1902_AO_TRGSRC_DTRIG: Digital trigger from AODTRIG.

Trigger-Polarity:
- P1902_AO_TrgPositive: Rising edge.
- P1902_AO_TrgNegative: Falling edge.

Trigger-Mode:
- P1902_AO_TRGMOD_POST: Post-trigger.
- P1902_AO_TRGMOD_DELAY: Delay-trigger.

Re-Trigger:
- P1902_AO_EnReTigger: Enable Re-Trigger.

Waveform-Separation:
- P1902_AO_EnDelay2: Enable Separation-Delay between waveforms.

dwReTriggerCnt : The re-trigger count is required when the P1902_AO_EnReTigger is set in *wTrigCtrl* parameter.

dwDLY1Cnt : This delay-count is required for P1902_AO_TRGMOD_DELAY trigger-mode. Based on internal conversion clock (80MHz), this count determines the delay-interval. For instance, assigning 800,000 to dwDLY1Cnt means 10 millisecond delay. (800,000 / 80,000,000)

dwDLY2Cnt : This delay-count is required when the P1902_AO_EnWaveformSep is set. Based on internal conversion clock (80MHz), this count determines the delay-interval. For instance, assigning 800,000 to dwDLY2Cnt means 10 millisecond separation-delay. (800,000 / 80,000,000)

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AO function is not supported.
ErrorInvalidAOCfgCtrl: The invalid settings is set to *wConfigCtrl***.**
ErrorInvalidAOTrigCtrl: The invalid settings is set to *wTrigCtrl***.**
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.38 UD_AO_VWriteChannel

@ Description

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

@ Modules Support

USB-1902/USB-19023

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AO_VWriteChannel (U16 ModuleNum, U16 Channel, F64 Voltage)

Visual Basic

UD_AO_VWriteChannel (ByVal ModuleNum As Integer, ByVal Channel As Integer, ByVal Voltage As Double) As Integer

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

Channel : The analog output channel number.
Range: 0 or 1 for USB-1902/USB-1903

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

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AO function is not supported.
ErrorInvalidIoChannel: The invalid setting is set to *Channel.*
ErrorDaVoltageOutOfRange: The value assigned to *Voltage* parameter is out of range.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.39 UD_AO_WriteChannel

@ Description

Writes a binary value to the specified analog output channel.

@ Modules Support

USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AO_WriteChannel (U16 ModuleNum, U16 Channel, U16 Value)

Visual Basic

UD_AO_WriteChannel (ByVal ModuleNum As Integer, ByVal Channel As Integer,
ByVal Value As Integer) As Integer

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
Channel :	The analog output channel number. Range: 0 or 1 for USB-1902/USB-1903
Value :	The value to be written to the analog output channel. Range: -32768 through 32767 for USB-1902/USB-1903

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AO function is not supported.
ErrorInvalidIoChannel: The invalid setting is set to *Channel.*
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.40 UD_AO_AsyncCheck

@ Description

Check the current status of the asynchronous analog output operation.

@ Modules Support

USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AO_AsyncCheck (U16 ModuleNum, BOOLEAN *Stopped, U32 *AccessCnt)

Visual Basic

UD_AO_AsyncCheck (ByVal ModuleNum As Integer, Stopped As Byte, AccessCnt As Long) As Integer

@ Parameter

ModuleNum : The id of the module that wants to perform this operation.

Stopped : Whether the asynchronous analog input operation has completed. If *Stopped* = TRUE, the analog output operation has stopped. Either the number of D/A conversions indicated in the call that initiated the asynchronous analog output operation has completed or an error has occurred. If *Stopped* = FALSE, the operation is not yet complete. (constants TRUE and FALSE are defined in UsbDask.h)

AccessCnt : In the condition that the trigger acquisition mode is not used, *AccessCnt* returns the number of D/A data that has been transferred at the time calling **`UD_AO_AsyncCheck().`**

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AO function is not supported.
ErrorConflictWithInfiniteOp: The infinite-trigger and infinite-repeat are conflict with this function.
ErrorConflictWithSyncMode: The synchronous AO operation is conflict with this function.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.41 UD_AO_AsyncClear

@ Description

Stop the asynchronous analog output operation.The configurations set with **`UD_AO_1902_Config()`**will be cleared as well.

@ Modules Support

USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AO_AsyncClear (U16 ModuleNum, U32 *AccessCnt)

Visual Basic

UD_AO_AsyncClear (ByVal ModuleNum As Integer, AccessCnt As Long) As Integer

@ Parameter

ModuleNum : The id of the module that wants to perform this operation.

AccessCnt : In the condition that the trigger acquisition mode is not used, *AccessCnt* returns the number of D/A data had been transferred out.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AO function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.42 UD_AO_AsyncDblBufferHalfReady

@ Description

In asynchronous double-buffered AO, indicates the half buffer is ready for data-updatig.

@ Modules Support

USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AO_AsyncDblBufferHalfReady (U16 ModuleNum, BOOLEAN *HalfReady, BOOLEAN *StopFlag)

Visual Basic

UD_AO_AsyncDblBufferHalfReady(ByVal ModuleNum As Integer, HalfReady As Byte, StopFlag As Byte) As Integer

@ Parameter

ModuleNum : The id of the module that wants to perform this operation.

HalfReady : Whether the half buffer of driver buffer is available. If *HalfReady* = TRUE, you can call **`UD_AO_AsyncDblBufferTransfer()`** to copy the output data to driver buffer. (constants TRUE and FALSE are defined in UsbDask.h)

StopFlag : Whether the asynchronous analog output operation has completed. If *StopFlag* = TRUE, the analog input operation has stopped. If *StopFlag* = FALSE, the operation is not yet complete. (constants TRUE and FALSE are defined in UsbDask.h)

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AO function is not supported.
ErrorConflictWithSyncMode: The synchronous AI operation is conflict with this function.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.43 UD_AO_AsyncDblBufferMode

@ Description

Enable/disable the double-buffered mode or FIFO mode for D/A data output.

@ Modules Support

USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AO_AsyncDblBufferMode (U16 ModuleNum, BOOLEAN Enable, BOOLEAN bEnFifoMode)

Visual Basic

UD_AO_AsyncDblBufferMode (ByVal ModuleNum As Integer, ByVal Enable As Byte, ByVal bEnFifoMode As Byte) As Integer

@ Parameter

ModuleNum : The id of the module that wants to perform this operation.
Enable : Whether the double-buffered mode is enabled or not.
TRUE: double-buffered mode is enabled.
FALSE: double-buffered mode is disabled.
(constants TRUE and FALSE are defined in UsbDask.h)
bEnFifoMode : Whether the FIFO mode is enabled or not.
TRUE: FIFO mode is enabled.
FALSE: FIFO mode is disabled.
(constants TRUE and FALSE are defined in UsbDask.h)

Note: The Double-Buffered mode and FIFO mode are manual exclusive. The FIFO mode only can be enabled while disabling double-buffered mode.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AO function is not supported.
ErrorInvalidOperationMode: The double-buffered mode and FIFO mode cannot be enabled at the same time.
ErrorDblBufModeNotAllowed: Re-trigger is not supported in double-buffered mode.
ErrorContIoActive: The AO function had not been completed. Call UD_AO_AsyncClear() to Stop AO function.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.44 UD_AO_ContBufferCompose

@ Description

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

@ Modules Support

USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AO_AsyncDblBufferMode (U16 ModuleNum, U16 TotalChnCount, U16
        ChnNum, U32 UpdateCount, void *ConBuffer, void *Buffer)
```

Visual Basic

```
UD_AO_AsyncDblBufferMode (ByVal ModuleNum As Integer, ByVal TotalChnCount
        As Integer, ByVal ChnNum As Integer, ByVal UpdateCount As Long,
        ConBuffer As Any, Buffer As Any) As Integer
```

Buffer containing the output data for the specified channel.

@ Parameter

ModuleNum : The id of the module that wants to perform this operation.
TotalChnCount : Number of AO channels to be performed. Valid value: 1 or 2.
ChnNum : Specified AO channel number. Valid value: 0 or 1.
UpdateCount : Size (in samples) of the specified channel buffer. This is not the size of the buffer for continuous output operation.
ConBuffer : Buffer for multi-channels of continuous output operation.
Buffer : Buffer containing the output data for the specified channel.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AO function is not supported.
ErrorUndefinedParameter: The invalid settings are assigned to ***TotalChnCount*** or ***ChnNum*** parameters.

2.2.45 UD_AO_AsyncDblBufferTransfer

@ Description

This function helps to update the output D/A data. The target half-buffer is specified with the wBufferID parameter.

@ Modules Support

USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AO_AsyncDblBufferTransfer (U16 ModuleNum, U16 wBufferID, U16 *Buffer)
```

Visual Basic

```
UD_AO_AsyncDblBufferTransfer (ByVal ModuleNum As Integer, ByVal wBufferID
        As Integer, Buffer As Integer) As Integer
```

@ Parameter

ModuleNum : The id of the module that performs the asynchronous double-buffered operation.

wBufferID : The id of the half-buffer that the D/A data will be copied into.

Buffer : The user buffer. An array that the D/A data will be updated. Please refer to Appendix C, *AI Data Format* for the data format in *Buffer or the data file*.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorNotDoubleBufrerMode: The AO operation is not started with double-buffered mode.
ErrorInvalidBufferID: The half-buffer that is specified with *wBufferID* is not ready yet.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.46 UD_AO_SetTimeOut

@ Description

This function sets the timeout for synchronous AO operation (SYNCH_OP).

@ Modules Support

USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AO_SetTimeOut (U16 ModuleNum, U32 dwTimeOut)

Visual Basic

UD_AO_ReadChannel (ByVal ModuleNum As Integer, ByVal dwTimeOut As Long)
As Integer

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

dwTimeOut : The Timeout for synchronous operation, in millisecond.
This setting is applied to WaitForSingleObject(). However, if this parameter is set to zero, the INFINITE is applied to WaitForSingleObject().

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AO function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.47 UD_AO_ContWriteChannel

@ Description

This function performs continuous D/A conversions on the specified analog output channels.

@ Modules Support

USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AO_ContWriteChannel (U16 ModuleNum, U16 wChannel, VOID* pAOBuffer, U32 dwWriteCount, U32 wIterations, U32 dwCHUI, U16 finite, U16 SyncMode)

Visual Basic

UD_AO_ContWriteChannel (ByVal ModuleNum As Integer, ByVal wChannel As Integer, pAOBuffer As Any, ByVal dwWriteCount As Long, ByVal wIterations As Long, ByVal dwCHUI As Long, ByVal finite As Integer, ByVal SyncMode As Integer) As Integer

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

wChannel : The channel to be performed the D/A updating:
USB-1901/USB-1902/USB-1903: 0 or 1

pAOBuffer : The memory that stores the update A/D data.

dwWriteCount : If double-buffered mode is disabled, the total update count for each channel to be performed. For double-buffered output, *dwWriteCount* is the size (in samples) of the circular buffer.

Note: The value of *dwWriteCount* must be the multiple of 256 for non-double-buffer mode, or multiple of 512 for double-buffer mode.

---

wIterations : The iterations to repeat the D/A data. If the D/A operation is performed synchronously, this parameter must be 1.

dwCHUI : The update-interval counter for D/A data output. Based on the conversion-clock, this counter determines the interval between D/A data.
USB-1902/USB-1903 : 80 to 4294967295.=

finite: D/A output is infinitely or finitely.
0 : infinitely.
1 : finitely.

SyncMode : Whether this operation is performed synchronously or asynchronously. If any trigger mode is enabled by calling **`UD_AO_1902_Config()`**, this operation should be performed *asynchronously*.
Valid values:
SYNCH_OP: synchronous A/D conversion, that is, the function does not return until the A/D operation complete.
ASYNCH_OP:asynchronous A/D conversion.

Note: When SYNCH_OP is selected, the UD_AO_SetTimeOut() can set the Timeout for synchronous operation.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidAOIteration: The Iteration is zero with *finite* is 1.
ErrorTransferCountTooLarge: The wWriteCount is too large.
ErrorZeroChannelNumber;: The numChans is zero.
ErrorContIoNotAllowed: The continuous operation is not supported.
ErrorConflictWithSyncMode: Only one-shot D/A operation supports SYNCH_OP.
ErrorInvalidTransferCount: The dwWriteCount is not multiple of 256/512.
ErrorInvalidOperationMode: Either FIFO mode or double-buffered mode can support AO re-trigger / repeat.
ErrorAOFifoCountTooLarge: The dwWriteCount is larger than onboard FIFO size.
ErrorConflictWithDelay2: The P1902_*AO_EnDelay2* in UD_AO_1902_Config() needs at least 2 iterations.
ErrorConflictWithReTrig: The re-trigger is manual-exclusive to repeating D/A operation.
ErrorInvalidSampleRate: The dwCHUI is less than 80.
ErrorInvalidTriggerMode: Neither double-buffered AO nor SYNCH_OP operation supports re-trigger feature.
ErrorSystemCallFailed: Failed to forward the command to driver, please call GetLastError() for detailed system-error.
ErrorContIoActive: The AO function had not been completed. Call UD_AO_AsyncClear() to Stop AO function.
ErrorWaitingUSBHostResponse: This is usually caused by trigger-enabled AO operation. Call UD_AO_AsyncClear() to disable the waiting state.
ErrorAOFifoModeTimeout: The D/A data transmission timeout with FIFO mode.
ErrorTimeoutFromSyncMode: The synchronous AO operation is time-out.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.48 UD_AO_ContWriteMultiChannels

@ Description

This function performs continuous D/A conversions on the specified analog output channels.

@ Modules Support

USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AO_ContWriteMultiChannels (U16 ModuleNum, U16 numChans, U16 *Chans, VOID* pAOBuffer, U32 dwWriteCount, U32 wIterations, U32 dwCHUI, U16 finite, U16 SyncMode)

Visual Basic

UD_AO_ContWriteMultiChannels (ByVal ModuleNum As Integer, ByVal numChans As Integer, Chans As Integer, pAOBuffer As Any, ByVal dwWriteCount As Long, ByVal wIterations As Long, ByVal dwCHUI As Long, ByVal finite As Integer, ByVal SyncMode As Integer) As Integer

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

numChans : The number of analog input channels in the array *Chans*. The valid value:
USB-1902/USB-1903: 1 to 2

Chans : Array of analog output channel numbers.
USB-1902/USB-1903: 0 or 1

pAOBuffer : The memory that stores the update A/D data.

dwWriteCount : If double-buffered mode is disabled, the total update count for each channel to be performed. For double-buffered output, *dwWriteCount* is the size (in samples) of the circular buffer.

Note: The value of *dwWriteCount* must be the multiple of 256 for non-double-buffer mode, or multiple of 512 for double-buffer mode.

---

wIterations : The iterations to repeat the D/A data. If the D/A operation is performed synchronously, this parameter must be 1.

dwCHUI : The update-interval for D/A data output. Based on the conversion-clock, this counter determines the interval between D/A data.
USB-1902/USB-1903 : 80 to 4294967295.

finite: D/A output is infinitely or finitely.
0 : infinitely.
1 : finitely.

SyncMode : Whether this operation is performed synchronously or asynchronously. If any trigger mode is enabled by calling **`UD_AO_1902_Config()`**, this operation should be performed *asynchronously*.
Valid values:
SYNCH_OP: synchronous A/D conversion, that is, the function does not return until the A/D operation complete.
ASYNCH_OP:asynchronous A/D conversion.

Note: When SYNCH_OP is selected, the UD_AO_SetTimeOut() can set the Timeout for synchronous operation.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidAOIteration: The Iteration is zero with *finite* is 1.
ErrorTransferCountTooLarge: The wWriteCount is too large.
ErrorZeroChannelNumber;: The numChans is zero.
ErrorContIoNotAllowed: The continuous operation is not supported.
ErrorConflictWithSyncMode: Only one-shot D/A operation supports SYNCH_OP.
ErrorInvalidTransferCount: The dwWriteCount is not multiple of 256/512.
ErrorInvalidOperationMode: Either FIFO mode or double-buffered mode can support AO re-trigger / repeat.
ErrorAOFifoCountTooLarge: The dwWriteCount is larger than onboard FIFO size.
ErrorConflictWithDelay2: The *P1902_AO_EnDelay2* in UD_AO_1902_Config() needs at least 2 iterations.
ErrorConflictWithReTrig: The re-trigger is manual-exclusive to repeating D/A operation.
ErrorInvalidSampleRate: The dwCHUI is less than 80.
ErrorInvalidTriggerMode: Neither double-buffered AO nor SYNCH_OP operation supports re-trigger feature.
ErrorSystemCallFailed: Failed to forward the command to driver, please call GetLastError() for detailed system-error.
ErrorContIoActive: The AO function had not been completed. Call UD_AO_AsyncClear() to Stop AO function.
ErrorWaitingUSBHostResponse: This is usually caused by trigger-enabled AO operation. Call UD_AO_AsyncClear() to disable the waiting state.
ErrorAOFifoModeTimeout: The D/A data transmission timeout with FIFO mode.
ErrorTimeoutFromSyncMode: The synchronous AO operation is time-out.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.49 UD_AO_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.

@ Modules Support

USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_AO_InitialMemoryAllocated (U16 ModuleNum, U32 MemSize)
```

Visual Basic

```
UD_AO_InitialMemoryAllocated (ByVal ModuleNum As Integer, MemSize As Long)
        As Integer
```

@ Parameter

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

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: Indicates the USB device was disconnected.
ErrorFuncNotSupport: The AI function is not supported.

2.2.50 UD_GPTC_Clear

@ Description

Halts the specified general-purpose timer/counter operation and reloads the initial value of the timer/counter.

@ Cards Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_GPTC_Clear (U16 CardNumber, U16 GCtr)

Visual Basic

UD_GPTC_Clear (ByVal CardNumber As Integer, ByVal GCtr As Integer) As Integer

@ Parameter

CardNumber : The card id of the card that want to perform this operation.
GCtr : The counter number.
USB-1901/USB-1902/USB-1903: 0 to 3
USB-2401: 0 to 1
USB-2405: 0 to 1
USB-1210: 0 to 3

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: The module is disconnected. Please close the related application and re-register the module again.
ErrorInvalidCounter: The invalid setting is assigned to *GCtr* parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorConflictWithGPIOConfig: Incorrect GPIO configuration, please check the settings in UD_DIO_1902_Config() / UD_DIO_2401_Config() / UD_DIO_2405_Config() / UD_DIO_Config().
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.51 UD_GPTC_Setup

@ Description

Sets the configuration of selected counter/timer.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_GPTC_Setup (U16 ModuleNum, U16 GCtr, U16 Mode, U16 SrcCtrl, U16 PolCtrl, U32 LReg_Val1, U32 LReg2_Val2, U32 PulseCount)

Visual Basic

UD_GPTC_Setup (ByVal ModuleNum As Integer, ByVal GCtr As Integer, ByVal Mode As Integer, ByVal SrcCtrl As Integer, ByVal PolCtrl As Integer, ByVal LReg_Val1 As Long, ByVal LReg_Val2 As Long, ByVal PulseCount As Long) As Integer

@ Parameter

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

GCtr : The counter number.
USB-1901/USB-1902/USB-1903: 0 to 3
USB-2401/USB-2405: 0 to 1
USB-1210: 0 to 3

Mode : The timer/counter mode. Refer to the hardware manual for the mode description. Valid modes:

USB-1901/USB-1902/USB-1903/USB-2401/USB-1210:
SimpleGatedEventCNT
SinglePeriodMSR
SinglePulseWidthMSR
SingleGatedPulseGen
SingleTrigPulseGen
RetrigSinglePulseGen
SingleTrigContPulseGen
ContGatedPulseGen
EdgeSeparationMSR
SingleTrigContPulseGenPWM
ContGatedPulseGenPWM
MultipleGatedPulseGen

USB -2405:
ContGatedPulseGen
MultipleGatedPulseGen
(The Internal Gate will be applied, and always be active)

SrcCtrl : The setting for general-purpose timer/counter source control. This argument is an integer expression formed from one or more of the manifest constants defined in UsbDask.h. There are three groups of constants:

USB-1901/USB-1902/USB-1903/USB-2401/USB-1210:
Timer/Counter Source
GPTC_CLK_SRC_Int Internal time base

GPTC_CLK_SRC_Ext External time base from the GPTC_CLK pin

Timer/Counter Gate Source

GPTC_GATE_SRC_Int Gate is controlled by software.

GPTC_GATE_SRC_Ext Gate is controlled by the GPTC_GATE pin.

Timer/Counter UpDown Source

GPTC_UPDOWN_Int Up/Down is controlled by software.

GPTC_UPDOWN_Ext Up/Down is controlled by the GPTC_UD pin.

USB-2405:

This parameter will be ignored.

PolCtrl : The polarity settings for general-purpose timer/counter. This argument is an integer expression formed from one or more of the manifest constants defined in UsbDask.h. There are three groups of constants:

USB-1901/USB-1902/USB-1903/USB-2401/USB-1210:

Timer/Counter Gate Polarity

GPTC_GATE_LACTIVE Low active

GPTC_GATE_HACTIVE High active

Timer/Counter UpDown Polarity

GPTC_UPDOWN_LACTIVE Low active

GPTC_UPDOWN_HACTIVE High active

Timer/Counter Clock Source Polarity

GPTC_CLKSRC_LACTIVE Low active

GPTC_CLKSRC_HACTIVE High active

Timer/Counter Output Polarity

GPTC_OUTPUT_LACTIVE Low active

GPTC_OUTPUT_HACTIVE High active

USB-2405:

Timer/Counter Output Polarity

GPTC_OUTPUT_LACTIVE Low active

GPTC_OUTPUT_HACTIVE High active

dwLReg_Val1 :The meaning for the value depends on the mode the timer /counter performs.

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210:

SimpleGatedEventCNT	Configures as nitial count of GPTC
SinglePeriodMSR	Configures as initial count of GPTC
SinglePulseWidthMSR	Configures as initial count of GPTC
SingleGatedPulseGen	Configures as the pulse width
SingleTrigPulseGen	Configures as the pulse width
RetrigSinglePulseGen	Configures as the pulse width
SingleTrigContPulseGen	Configures as the pulse width
ContGatedPulseGen	Configures as the pulse width
EdgeSeparationMSR	Configures as initial count of GPTC
SingleTrigContPulseGenPWM	Configures as the pulse initial count
ContGatedPulseGenPWM	Configures as the pulse initial count
MultipleGatedPulseGen	Configures as the pulse initial count

Note: for USB-2405, if the MultipleGatedPulseGen mode is set, this parameter, **dwLReg_Val1,** will be ignored.

dwLReg_Val2 :The meaning for the value depends on the mode the timer /counter performs.

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210:

SimpleGatedEventCNT	Not used
SinglePeriodMSR	Not used
SinglePulseWidthMSR	Not used
SingleGatedPulseGen	Not used
SingleTrigPulseGen	Not used
RetrigSinglePulseGen	Not used
SingleTrigContPulseGen	Not used
ContGatedPulseGen	Not used
EdgeSeparationMSR	Not used
SingleTrigContPulseGenPWM	Configures as the pulse length count
ContGatedPulseGenPWM	Configures as the pulse length count
MultipleGatedPulseGen	Configures as the pulse length count

PulseCount : The count of output-pulse. This parameter is required when MultipleGatedPulseGen mode is selected.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: The module is disconnected. Please close the related application and re-register the module again.
ErrorInvalidCounter: The invalid setting is assigned to GCtr parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidPulseCount: The *PulseCount* is zero when MultipleGatedPulseGen mode is selected.
ErrorInvalidCounterMode: GCtr1 and GCtr3 only support MultipleGatedPulseGen mode; Invalid setting is assigned to *Mode* parameter.
ErrorConflictWithGPIOConfig: Incorrect GPIO configuration, please check the settings in UD_DIO_1902_Config() / UD_DIO_2401_Config() / UD_DIO_2405_Config() / UD_DIO_Config().
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.52 UD_GPTC_Setup_N

@ Description

This function provides the simlar features of ***UD_GPTC_Setup()***.
In the ***UD_GPTC_Setup()***,the duty-cycle of pulse-generating modes, XXXPulseGen, is always 50%; this setting can be configured in this function.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_GPTC_Setup (U16 ModuleNum, U16 GCtr, U16 Mode, U16 SrcCtrl, U16 PolCtrl, U32 LReg_Val1, U32 LReg2_Val2, U32 PulseCount)

Visual Basic

UD_GPTC_Setup (ByVal ModuleNum As Integer, ByVal GCtr As Integer, ByVal Mode As Integer, ByVal SrcCtrl As Integer, ByVal PolCtrl As Integer, ByVal LReg_Val1 As Long, ByVal LReg_Val2 As Long, ByVal PulseCount As Long) As Integer

@ Parameter

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

GCtr : The counter number.
USB-1901/USB-1902/USB-1903: 0 to 3
USB-2401: 0 to 1
USB-1210: 0 to 3

Mode : The timer/counter mode. Refer to the hardware manual for the mode description. Valid modes:

USB-1901/USB-1902/USB-1903/USB-2401/USB-1210:
- SimpleGatedEventCNT
- SinglePeriodMSR
- SinglePulseWidthMSR
- SingleGatedPulseGen
- SingleTrigPulseGen
- RetrigSinglePulseGen
- SingleTrigContPulseGen
- ContGatedPulseGen
- EdgeSeparationMSR
- SingleTrigContPulseGenPWM
- ContGatedPulseGenPWM
- MultipleGatedPulseGen

SrcCtrl : The setting for general-purpose timer/counter source control. This argument is an integer expression formed from one or more of the manifest constants defined in UsbDask.h. There are three groups of constants:

USB-1901/USB-1902/USB-1903/USB-2401/USB-1210:

Timer/Counter Source
GPTC_CLK_SRC_Int Internal time base
GPTC_CLK_SRC_Ext External time base from the GPTC_CLK pin

Timer/Counter Gate Source

GPTC_GATE_SRC_Int Gate is controlled by software.
GPTC_GATE_SRC_Ext Gate is controlled by the GPTC_GATE pin.
Timer/Counter UpDown Source
GPTC_UPDOWN_Int Up/Down is controlled by software.
GPTC_UPDOWN_Ext Up/Down is controlled by the GPTC_UD pin.

PolCtrl : The polarity settings for general-purpose timer/counter. This argument is an integer expression formed from one or more of the manifest constants defined in UsbDask.h. There are three groups of constants:

USB-1901/USB-1902/USB-1903/USB-2401/USB-1210:

Timer/Counter Gate Polarity
GPTC_GATE_LACTIVE Low active
GPTC_GATE_HACTIVE High active
Timer/Counter UpDown Polarity
GPTC_UPDOWN_LACTIVE Low active
GPTC_UPDOWN_HACTIVE High active
Timer/Counter Clock Source Polarity
GPTC_CLKSRC_LACTIVE Low active
GPTC_CLKSRC_HACTIVE High active
Timer/Counter Output Polarity
GPTC_OUTPUT_LACTIVE Low active
GPTC_OUTPUT_HACTIVE High active

dwLReg_Val1 :The meaning for the value depends on the mode the timer /counter performs.

USB-1901/USB-1902/USB-1903/USB-2401/USB-1210:

SimpleGatedEventCNT	Configures as nitial count of GPTC
SinglePeriodMSR	Configures as initial count of GPTC
SinglePulseWidthMSR	Configures as initial count of GPTC
SingleGatedPulseGen	Configures as the pulse initial count
SingleTrigPulseGen	Configures as the pulse initial count
RetrigSinglePulseGen	Configures as the pulse initial count
SingleTrigContPulseGen	Configures as the pulse initial count
ContGatedPulseGen	Configures as the pulse initial count
EdgeSeparationMSR	Configures as initial count of GPTC
SingleTrigContPulseGenPWM	Configures as the pulse initial count
ContGatedPulseGenPWM	Configures as the pulse initial count
MultipleGatedPulseGen	Configures as the pulse initial count

dwLReg_Val2 :The meaning for the value depends on the mode the timer /counter performs.

USB-1901/USB-1902/USB-1903/USB-2401/USB-1210:

SimpleGatedEventCNT	Not used
SinglePeriodMSR	Not used
SinglePulseWidthMSR	Not used
SingleGatedPulseGen	Configures as the pulse length count
SingleTrigPulseGen	Configures as the pulse length count
RetrigSinglePulseGen	Configures as the pulse length count
SingleTrigContPulseGen	Configures as the pulse length count
ContGatedPulseGen	Configures as the pulse length count
EdgeSeparationMSR	Not used
SingleTrigContPulseGenPWM	Configures as the pulse length count
ContGatedPulseGenPWM	Configures as the pulse length count
MultipleGatedPulseGen	Configures as the pulse length count

PulseCount : The count of output-pulse. This parameter is required when MultipleGatedPulseGen mode is selected.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: The module is disconnected. Please close the related application and re-register the module again.
ErrorInvalidCounter: The invalid setting is assigned to GCtr parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorInvalidPulseCount: The *PulseCount* is zero when MultipleGatedPulseGen mode is selected.
ErrorInvalidCounterMode: GCtr1 and GCtr3 only support MultipleGatedPulseGen mode; Invalid setting is assigned to *Mode* parameter.
ErrorConflictWithGPIOConfig: Incorrect GPIO configuration, please check the settings in UD_DIO_Config().
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.53 UD_GPTC_Control

@ Description

Controls for the selected counter/timer by software.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_GPTC_Control (U16 ModuleNum, U16 GCtr, U16 ParamID, U16 Value)

Visual Basic

UD_GPTC_Control (ByVal ModuleNum As Integer, ByVal GCtr As Integer, ByVal ParamID As Integer, ByVal Value As Integer) As Integer

@ Parameter

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

GCtr : The counter number.
USB-1901/USB-1902/USB-1903: 0 to 3
USB-2401/USB-2405: 0 to 1
USB-1210: 0 to 3

ParamID : The ID of the internal parameter of the general purpose timer/counter you want to control. Valid control parameters:

USB-1901/USB-1902/USB-1903/USB-2401/USB-1210:

IntGATE	Internal gate
IntUpDnCTR	Internal updown counter
IntENABLE	Starts or stops counter operation

USB-2405:

IntENABLE	Starts or stops counter operation

Value : The value for the control item specified by the ParamID parameter. The valid value is 0 or 1.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: The module is disconnected. Please close the related application and re-register the module again.
ErrorInvalidCounter: The invalid setting is assigned to *GCtr* parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorConflictWithGPIOConfig: Incorrect GPIO configuration, please check the settings in UD_DIO_1902_Config() / UD_DIO_2401_Config() / UD_DIO_2405_Config() / UD_DIO_Config().
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.54 UD_GPTC_Read

@ Description

Reads the counting-value of specified general-purpose timer/counter.

@ Cards Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_GPTC_Read (U16 CardNumber, U16 GCtr, U32 *pValue)
```

Visual Basic

```
UD_GPTC_Read (ByVal CardNumber As Integer, ByVal GCtr As Integer, pValue As
        Long) As Integer
```

@ Parameter

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

GCtr : The counter number.
USB-1901/USB-1902/USB-1903: 0 to 3
USB-2401: 0 to 1
USB-1210: 0 to 3

pValue : Returns the counter value of the specified general purpose timer/counter.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: The module is disconnected. Please close the related application and re-register the module again.
ErrorInvalidCounter: The invalid setting is assigned to *GCtr* parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorConflictWithGPIOConfig: Incorrect GPIO configuration, please check the settings in UD_DIO_1902_Config() / UD_DIO_2401_Config() / UD_DIO_Config().
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.55 UD_GPTC_Status

@ Description

Gets the status of specified general-purpose timer/counter.

@ Cards Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_GPTC_Status (U16 CardNumber, U16 GCtr, U16 *pValue)
```

Visual Basic

```
UD_GPTC_Status (ByVal CardNumber As Integer, ByVal GCtr As Integer, pValue
        As Integer) As Integer
```

@ Parameter

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

GCtr : The counter number.
USB-1901/USB-1902/USB-1903: 0 to 3
USB-2401: 0 to 1
USB-1210: 0 to 3

pValue : Returns the latched GPTC status of the specified general-purpose timer/counter from the GPTC status register. Value formats:
bit 0 1 indicates that the GPTC is counting.
0 indicates that the GPTC is not counting.
bit 1 1 indicates that the GPTC operation is done.
0 indicates that the GPTC operation is not yet done.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorCardDisconnected: The module is disconnected. Please close the related application and re-register the module again.
ErrorInvalidCounter: The invalid setting is assigned to *GCtr* parameter.
ErrorFuncNotSupport: The AI function is not supported.
ErrorConflictWithGPIOConfig: Incorrect GPIO configuration, please check the settings in UD_DIO_1902_Config() / UD_DIO_2401_Config() / UD_DIO_Config().
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.56 UD_DIO_1902_Config

@ Description

The USBDAQ module provides the multiple function DIO to support GPIO (General Purpose Input/Output), GPTC (General Purpose Timer Counter) and TC (Timer Couner). These multi-function pins are divided into 2 groups, and configured with wPart1Cfg and wPart2Cfg parameters.

@ Modules Support

USB-1901/USB-1902/USB-1903

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_DIO_1902_Config (U16 ModuleNum, U16 wPart1Cfg, U16 wPart2Cfg)

Visual Basic

UD_DIO_1902_Config (ByVal CardNumber As Integer, ByVal wPart1Cfg As Integer, ByVal wPart2Cfg As Integer) As Integer

@ Parameter

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

wPart1Cfg : The configuration for multiple-function group1 are:
USB-1901/USB-1902/USB-1903:
GPTC0_GPO1 : configure to GPTC0 and GPO1
GPI0_3_GPO0_1: configure to GPI0 ~ GPI3 and GPO0 ~ GPO1
GPTC0_TC1: GPTC0 and TC1 (TimerCouner1)

wPart2Cfg : The configuration for multiple-function group2 are:
USB-1901/USB-1902/USB-1903:
GPTC2_GPO3 : configure to GPTC2 and GPO3
GPI4_7_GPO2_3: configure to GPI4 ~ GPI7 and GPO2 ~ GPO3
GPTC2_TC3: GPTC2 and TC3 (TimerCouner3)

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorInvalidDioConfig: Invalid setting in either *wPart1Cfg* or *wPart2Cfg*.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.57 UD_DIO_2401_Config

@ Description

The USBDAQ module provides the multiple function DIO to support GPIO (General Purpose Input/Output), GPTC (General Purpose Timer Counter) and TC (Timer Couner).

@ Modules Support

USB-2401

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_DIO_2401_Config (U16 ModuleNum, U16 wPart1Cfg)

Visual Basic

UD_DIO_2401_Config (ByVal CardNumber As Integer, ByVal wPart1Cfg As Integer) As Integer

@ Parameter

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

wPart1Cfg : The configuration for multiple-function group1 are:

GPTC0_GPO1 : configure to GPTC0 and GPO1

GPI0_3_GPO0_1: configure to GPI0 ~ GPI3 and GPO0 ~ GPO1

GPTC0_TC1: GPTC0 and TC1 (TimerCouner1)

@ Return Code

NoError: The function returns successfully.

ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.

ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.

ErrorFuncNotSupport: The DIO function is not supported.

ErrorBadCardType: Indicates the module-type is not supported.

ErrorInvalidDioConfig: Invalid setting in *wPart1Cfg*.

ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.58 UD_DIO_2405_Config

@ Description

The USB-2405 provides the multiple-function DIO to support GPIO (General Purpose Input/Output), and GPTC (General Purpose Timer Counter). The DIO must be configured before calling other GPIO/GPTC related functions.

@ Modules Support

USB-2405

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_DIO_2405_Config (U16 ModuleNum, U16 wPart1Cfg, U16 wPart2Cfg)

Visual Basic

UD_DIO_2405_Config (ByVal CardNumber As Integer, ByVal wPart1Cfg As Integer, ByVal wPart2Cfg As Integer) As Integer

@ Parameter

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

wPart1Cfg : The configuration for multiple-function port1 are:
- P2405_DIGITAL_INPUT : configure to Digital-Input
- P2405_COUNTER_INPUT: configure to Pulse-Input
- P2405_DIGITAL_OUTPUT: configure to Digital-Output
- P2405_PULSE_OUTPUT: configure to Pulse-Output

wPart2Cfg : The configuration for multiple-function port2 are:
- P2405_DIGITAL_INPUT : configure to Digital-Input
- P2405_COUNTER_INPUT: configure to Pulse-Input
- P2405_DIGITAL_OUTPUT: configure to Digital-Output
- P2405_PULSE_OUTPUT: configure to Pulse-Output

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorInvalidDioConfig: Invalid setting in either *wPart1Cfg* or *wPart2Cfg*.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.59 UD_DIO_Config

@ Description

The UD-DASK devices provide the multiple-function DIO to support GPIO (General Purpose Input/Output), and GPTC (General Purpose Timer Counter). The DIO ports must be configured before calling other GPIO/GPTC related functions.

@ Modules Support

USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_DIO_Config (U16 ModuleNum, U16 wPort0Cfg, U16 wPort1Cfg)

Visual Basic

UD_DIO_Config (ByVal CardNumber As Integer, ByVal wPort0Cfg As Integer, ByVal wPort1Cfg As Integer) As Integer

@ Parameter

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

wPort0Cfg : The configuration for multiple-function port0. The valid settings are:
GPTC0_GPO1 : configure to GPTC0 and GPO1
GPI0_3_GPO0_1: configure to GPI0 ~ GPI3 and GPO0 ~ GPO1
GPTC0_TC1: GPTC0 and TC1 (TimerCouner1)

wPort1Cfg : The configuration for multiple-function port1 The valid settings are:
GPTC2_GPO3 : configure to GPTC2 and GPO3
GPI4_7_GPO2_3: configure to GPI4 ~ GPI7 and GPO2 ~ GPO3
GPTC2_TC3: GPTC2 and TC3 (TimerCouner3)

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorBadCardType: Indicates the module-type is not supported.
ErrorInvalidDioConfig: Invalid setting in either *wPart1Cfg* or *wPart2Cfg*.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.60 UD_DI_ReadLine

@ Description

Read the digital logic state of the specified DI port/line.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-7250/USB-7230/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_DI_ReadLine (U16 ModuleNum, U16 Port, U16 Line, U16 *State)

Visual Basic

UD_DI_ReadLine (ByVal ModuleNum As Integer, ByVal Port As Integer, ByVal Line As Integer, State As Integer) As Integer

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
Port :	Digital input port number. The valid value: USB-1901/USB-1902/USB-1903/USB-2405/USB-1210: 0, 1 USB-7250/USB-7230: 0
Line :	The digital line to be read. The valid value: USB-1901/USB-1902/USB-1903/USB-1210: 0 through 3 USB-7250: 0 through 7 USB-7230: 0 through 15 USB-2405: 0
State :	Returns the digital logic state, 0 or 1, of the specified line.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorInvalidDioLine: Invalid value is assigned to *Line* parameter.
ErrorConflictWithGPIOConfig: Incorrect GPIO configuration, please check the settings in UD_DIO_1902_Config() / UD_DIO_2405_Config() / UD_DIO_Config().
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.61 UD_DI_ReadPort

@ Description

Read digital data from the specified digital input port.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-7250/USB-7230/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_DI_ReadPort (I16 ModuleNum, U16 Port, U32 *Value)
```

Visual Basic

```
UD_DI_ReadPort (ByVal ModuleNum As Integer, ByVal Port As Integer, Value As
        Long) As Integer
```

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

Port : Digital input port number. The valid value:
USB-1901/USB-1902/USB-1903/USB-2405/USB-1210: 0, 1
USB-2401/USB-7250/USB-7230: 0

Value : Returns the digital data read from the specified port.
USB-1901/USB-1902/USB-1903/USB-2401/USB-1210: 4-bit data
USB-7250: 8-bit data
USB-7230: 16-bit data
USB-2405: 1-bit data

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorConflictWithGPIOConfig: Incorrect GPIO configuration, please check the settings in UD_DIO_1902_Config() / UD_DIO_2401_Config() / UD_DIO_2405_Config() / UD_DIO_Config().
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.62 UD_DO_ReadLine

@ Description

Read back the digital-input state from the specified DO port/line.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-7250/USB-7230/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_DO_ReadLine (U16 CardNumber, U16 Port, U16 Line, U16 *State)
```

Visual Basic

```
UD_DO_ReadLine (ByVal CardNumber As Integer, ByVal Port As Integer, ByVal
        Line As Integer, State As Integer) As Integer
```

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
Port :	Digital output port number. The valid value: USB-1901/USB-1902/USB-1903/USB-2405/USB-1210: 0, 1 USB-2401/USB-7250/USB-7230: 0
Line :	The digital line to be accessed. The valid value: USB-1901/USB-1902/USB-1903/USB-2401/USB-1210: 0 through 1 USB-7250: 0 through 7 USB-7230: 0 through 15 USB-2405: 0
State :	Returns the digital logic state, 0 or 1, of the specified line.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorInvalidDioLine: Invalid value is assigned to *Line* parameter.
ErrorConflictWithGPIOConfig: Incorrect GPIO configuration, please check the settings in UD_DIO_1902_Config() / UD_DIO_2401_Config() / UD_DIO_2405_Config() / UD_DIO_Config().
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.63 UD_DO_ReadPort

@ Description

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

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-7250/USB-7230/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_DO_ReadPort (U16 ModuleNum, U16 Port, U32 *Value)
```

Visual Basic

```
UD_DO_ReadPort (ByVal ModuleNum As Integer, ByVal Port As Integer, Value As
        Long) As Integer
```

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

Port : Digital output port number. The valid value:
USB-1901/USB-1902/USB-1903/USB-2405/USB-1210: 0, 1
USB-2401/USB-7250/USB-7230: 0

Value : Returns the digital data read from the specified output port.
USB-1901/USB-1902/USB-1903/USB-2401/USB-1210: 2-bit data
USB-7250: 8-bit data
USB-7230: 16-bit data
USB-2405: 1-bit data

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorConflictWithGPIOConfig: Incorrect GPIO configuration, please check the settings in UD_DIO_1902_Config() / UD_DIO_2401_Config() / UD_DIO_2405_Config() / UD_DIO_Config.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.64 UD_DO_WriteLine

@ Description

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

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-7250/USB-7230/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_DO_WriteLine (U16 ModuleNum, U16 Port, U16 Line, U16 State)
```

Visual Basic

```
UD_DO_WriteLine (ByVal ModuleNum As Integer, ByVal Port As Integer, ByVal
        DoLine As Integer, ByVal State As Integer) As Integer
```

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
Port :	Digital output port number. The valid value: USB-1901/USB-1902/USB-1903/USB-2405/USB-1210: 0, 1 USB-2401/USN-7250/USB-7230: 0
Line :	The digital line to write to. The valid value: USB-1901/USB-1902/USB-1903/USB-2401/USB-1210: 0 through 1 USB-7250: 0 through 7 USB-7230: 0 through 15 USB-2405: 0
State :	The new digital logic state, 0 or 1.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorInvalidDioLine: Invalid value is assigned to *Line* parameter.
ErrorConflictWithGPIOConfig: Incorrect GPIO configuration, please check the settings in UD_DIO_1902_Config() / UD_DIO_2401_Config() / UD_DIO_2405_Config() / UD_DIO_Config().
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.65 UD_DO_WritePort

@ Description

Writes digital data to the specified digital output port.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-7250/USB-7230/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_DO_WritePort (U16 ModuleNum, U16 Port, U32 Value)

Visual Basic

UD_DO_WritePort (ByVal ModuleNum As Integer, ByVal Port As Integer, ByVal Value As Long) As Integer

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

Port : Digital output port number. The cards that support this function and their corresponding valid value are as follows:
USB-1901/USB-1902/USB-1903/USB-2405/USB-1210: 0, 1
USB-2401/USB-7250/USB-7230: 0

Value : Digital data that is written to the specified port.
USB-1901/USB-1902/USB-1903/USB-2401/USB-1210: 2-bit data
USB-7250: 8-bit data
USB-7230: 16-bit data
USB-2405: 1-bit data

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorConflictWithGPIOConfig: Incorrect GPIO configuration, please check the settings in UD_DIO_1902_Config() / UD_DIO_2401_Config() / UD_DIO_2405_Config() / UD_DIO_Config().
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.66 UD_DO_SetInitPattern

@ Description

Set the state of the initial. The initial pattern is sent to DO channel while power-on initializes.

@ Modules Support

USB-7250/USB-7230

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_DO_SetInitPattern (U16 ModuleNum, U16 Port, U32 *Pattern)
```

Visual Basic

```
UD_DO_SetInitPattern (ByVal ModuleNum As UShort, ByVal Port As UShort, ByRef
        Pattern As UInteger) As Short
```

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
Port :	Digital output port number. The cards that support this function and their corresponding valid value are as follows: USB-2401/USB-7250/USB-7230: 0
Pattern :	State of the set pattern.. USB-7250: 8-bit data USB-7230: 16-bit data

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.67 UD_DO_GetInitPattern

@ Description

Obtains the state of the state set by SetInitPattern.

@ Modules Support

USB-7250/USB-7230

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_DO_GetInitPattern (U16 ModuleNum, U16 Port, U32 *Pattern)

Visual Basic

UD_DO_GetInitPattern (ByVal ModuleNum As UShort, ByVal Port As UShort, ByRef Pattern As UInteger) As Short

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
Port :	Digital output port number. The cards that support this function and their corresponding valid value are as follows: USB-2401/USB-7250/USB-7230: 0
Pattern :	Returns the state set by SetInitPattern function. USB-7250: 8-bit data USB-7230: 16-bit data

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.68 UD_DI_SetCOSInterrupt32

@ Description

Enables or disables the COS (Change Of State) interrupt detection capability of the specified port.

@ Modules Support

USB-7250/USB-7230

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_DI_SetCOSInterrupt32 (U16 ModuleNum, U16 Port, U32 Ctrl,  HANDLE
        *hEvent, BOOLEAN ManualReset)
```

Visual Basic

```
UD_DI_SetCOSInterrupt32 (ByVal ModuleNum As UShort, ByVal Port As UShort,
        ByVal Ctrl As UInteger, ByRef hEvent As IntPtr, ByVal ManualReset As
        Boolean) As Short
```

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
Port :	Digital output port number. The cards that support this function and their corresponding valid value are as follows: USB-7250/USB-7230: 0
Ctrl :	Each bit of the value of ctrl controls one DI channel. The '0' value of the bit value disable the COS function of the corresponding line, and the '1' value of the bit value enable the COS function of the corresponding line. The valid values for ctrl are as follows: USB-7250: 0 to 0xFF USB-7230: 0 to 0xFFFF
hEvent :	Returned COS interrupt event handle.
ManualReset :	Specifies whether the event is (1) manual-reset by function ResetEvent in user's application or (0) auto-reset by driver.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.69 UD_DI_GetCOSLatchData32

@ Description

Gets the 32-bit width DI data latched in the COS Latch register while the Change-of-State (COS) interrupt occurs.

@ Modules Support

USB-7250/USB-7230

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 DIO_GetCOSLatchData32 (U16 ModuleNum, U16 Port, U32 *CosLData)
```

Visual Basic

```
UD_DI_GetCOSLatchData32 (ByVal ModuleNum As UShort, ByVal Port As UShort,
        ByRef CosLData As UInteger) As Short
```

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
Port :	Digital output port number. The cards that support this function and their corresponding valid value are as follows: USB-7250/USB-7230: 0
CosLData:	Returns the DI data latched in the COS Latch register while the Change-of-State(COS) interrupt occurs. USB-7250: 8-bit data USB-7230: 16-bit data

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.70 UD_DI_Control

@ Description

Set the filter enable state of the DI channel.

@ Modules Support

USB-7250/USB-7230

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_DI_Control (U16 ModuleNum, U16 Port, U32 Ctrl)
```

Visual Basic

```
UD_DI_Control (ByVal ModuleNum As UShort, ByVal Port As UShort, ByVal Ctrl As
        UInteger) As Short
```

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
Port :	Digital output port number. The cards that support this function and their corresponding valid value are as follows: USB-7250/USB-7230: 0
Ctrl :	Each bit of the value of ctrl controls one DI channel. The '0' value of the bit value disable the filter function of the corresponding line, and the '1' value of the bit value enable the filter function of the corresponding line. The valid values for ctrl are as follows: USB-7250: 0 to 0xFF USB-7230: 0 to 0xFFFF

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.71 UD_DI_SetupMinPulseWidth

@ Description

Set the filter width of the DI channel.

@ Modules Support

USB-7250/USB-7230

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_DI_SetupMinPulseWidth (U16 ModuleNum, U16 Value)

Visual Basic

UD_DI_SetupMinPulseWidth (ByVal ModuleNum As UShort, ByVal Value As UShort) As Short

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

Value : Multiples of the period of 48MHz as the DI filter width. The valid values for value are as follows:
USB-7250/USB-7230: 1 to 65535

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.72 UD_CTR_ReadEdgeCounter

@ Description

Get the rising edge counter value of the Counter channel.

@ Modules Support

USB-7250/USB-7230/USB-2405

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_CTR_ReadEdgeCounter (U16 ModuleNum, U16 Ctr, U32* Value)

Visual Basic

UD_CTR_ ReadEdgeCounter (ByVal ModuleNum As UShort, ByVal Ctr as UShort, ByRef Value As UInteger) As Short

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
Ctr :	Counter number. The cards that support this function and their corresponding valid value are as follows: USB-7250/USB-7230/USB-2405: 0, 1
Value :	Value of the internal rising edge counter. The valid values for value are as follows: USB-7250/USB-7230/USB-2405: 32-bit data

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.73 UD_CTR_ReadFrequency

@ Description

Get the frequency counter value of the Counter channel.

@ Modules Support

USB-7250/USB-7230

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_CTR_ReadFrequency (U16 ModuleNum, U16 Ctr, F64* Value)

Visual Basic

UD_CTR_SetupMinPulseWidth (ByVal ModuleNum As UShort, ByVal Ctr as UShort, ByRef Value As Double) As Short

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
Ctr :	Counter number. The cards that support this function and their corresponding valid value are as follows: USB-7250/USB-7230: 0, 1
Value :	Value of the internal frequency counter.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.74 UD_CTR_Control

@ Description

Configures the selected counter to operate in the specified mode.

@ Modules Support

USB-7250/USB-7230/USB-2405

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_CTR_Control (U16 ModuleNum, U16 Ctr, U32 Ctrl)

Visual Basic

UD_CTR_Control (ByVal ModuleNum As UShort, ByVal Ctr as UShort, ByVal Ctrl As Uinteger) As Short

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

Ctr : Counter number. The cards that support this function and their corresponding valid value are as follows:
USB-7250/USB-7230/USB-2405: 0, 1

Ctrl : Bitwise or of the following enumerative values:

USB-7250/USB-7230:
- UD_CTR_Filter_Disable/UD_CTR_Filter_Enable
- UD_CTR_Reset_Rising_Edge_Counter
- UD_CTR_Reset_Frequency_Counter
- UD_CTR_Polarity_Positive/UD_CTR_Polarity_Negative

USB-2405:
- UD_CTR_Reset_Rising_Edge_Counter
- UD_CTR_Polarity_Positive/UD_CTR_Polarity_Negative

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.75 UD_CTR_SetupMinPulseWidth

@ Description

Set the filter width of the Counter channel.

@ Modules Support

USB-7250/USB-7230

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_CTR_SetupMinPulseWidth (U16 ModuleNum, U16 Ctr, U16 Value)

Visual Basic

UD_CTR_SetupMinPulseWidth (ByVal ModuleNum As UShort, ByVal Ctr as UShort, ByVal Value As UShort) As Short

@ Parameter

ModuleNum :	The id of the module that want to perform this operation.
Ctr :	Counter number. The cards that support this function and their corresponding valid value are as follows: USB-7250/USB-7230: 0, 1
Value :	Multiples of the period of 48MHz as the CTR filter width. The valid values for value are as follows: USB-7250/USB-7230: 1 to 65535

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The DIO function is not supported.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.76 UD_Read_ColdJunc_Thermo

@ Description

Read the temperature for the thermocouple cold-junction compensation.

@ Modules Support

USB-2401

@ Syntax

Microsoft C/C++ and Borland C++

```
I16 UD_Read_ColdJunc_Thermo ( U16 ModuleNum, F64 *fValue )
```

Visual Basic

```
UD_Read_ColdJunc_Thermo (ByVal ModuleNum As UShort, ByRef fValue As
        Double) As Short
```

@ Parameter

ModuleNum : The id of the module that want to perform this operation.
fValue : The memory that is stored the cold-junction temperature.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorFuncNotSupport: The Cold-Junction sensor is not supported.
ErrorTimeoutFromSyncMode: The synchronous calibration-operation is time-out.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.77 UD_2405_Calibration

@ Description

Start the auto-calibration and update the settings into EEPROM. Please re-start the USB module to activate the calibration settings.

@ Modules Support

USB-2405

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_2405_Calibration (U16 ModuleNum)

Visual Basic

UD_2405_Calibration (ByVal ModuleNum As UShort) As Short

@ Parameter

ModuleNum : The id of the module that want to perform this operation.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorOpenEventFailed: Open event failed in device driver.
ErrorInvalidRefVoltage: Indicates the reference-voltage is invalid.
ErrorTimeoutFromSyncMode: The synchronous calibration-operation is time-out.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.78 UD_AI_Calibration

@ Description

Start the AI auto-calibration and program the calibration-settings into EEPROM.
Please re-start the USB module to activate the calibration settings.

@ Modules Support

USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_AI_Calibration (U16 ModuleNum, U32 dwReserved)

Visual Basic

UD_AI_Calibration (ByVal ModuleNum As Integer, By dwReserved As Long) As Integer

@ Parameter

ModuleNum : The id of the module that want to perform this operation.
dwReserved: This parameter is reserved for future.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber: The ModuleNum is larger than MAX_USB_DEVICE.
ErrorCardNotRegistered: The specific module had not been registered. Please check the ModuleNum parameter.
ErrorOpenEventFailed: Open event failed in device driver.
ErrorInvalidRefVoltage: Indicates the reference-voltage is invalid.
ErrorTimeoutFromSyncMode: The synchronous calibration-operation is time-out.
ErrorCardDisconnected: Indicates the USB device was disconnected.

2.2.79 UD_Register_Card

@ Description

Initializes the hardware and software states of a USBDAQ module, and then returns a numeric card ID that corresponds to the card initialized. UD_Register_Card must be called before any other USB-DASK library functions can be called for that card. The function initializes the card and variables internal to USB-DASK library.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-7250/USB-7230/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_Register_Card (U16 ModuleType, U16 module_num)

Visual Basic

UD_Register_Card (ByVal ModuleType As Integer, ByVal module_num As Integer) As Integer

@ Parameter

ModuleType : The type of USB module to be initialized. ADLink will periodically upgrades USB-DASK to support new USB-DASK modules. Please refer to *Release Notes* for the module types that are supported in the latest USB-DASK. Following are the constants defined in UsbDask.h that USBDASK supports currently or in the near future:

USB_1901
USB_1902
USB_1903
USB_2401
USB_7250
USB_7230
USB_2405
USB-1210

module_num : The sequence number of the card with ***the same module type*** (as defined in argument *ModuleType*). The sequence number setting is according to the onboard DIP-switch.

@ Return Code

This function returns a numeric id for the module initialized. The range of module id is between 0 and (MAX_USB_DEVICE-1). If there is any error occurs, it will return negative error code, the possible error codes are listed below:
ErrorTooManyRegisteredCards : more than MAX_USB_DEVICE tasks register USBDASK devices.
ErrorUnknownCardType : Invalid Module-Type is assigned to ModuleType parameter.
ErrorOpenDriverFailed : Failed to open the device-node, please call GetLastError() for detailed system error.
ErrorConfigIoctl : Failed to get module description from device driver, please call GetLastError() for detailed system error.
ErrorOpenEvtIoctl : Failed to bind the event objects, please call GetLastError() for detailed system error.

2.2.80 UD_Release_Card

@ Description

There are at most MAX_USB_DEVICE modules that can be registered simultaneously. This function is called to release the registered module. Also by the end of a program, you need to use this function to release all cards that were registered.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-7250/USB-7230/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 Release_Card (U16 CardNumber)

Visual Basic

Release_Card (ByVal CardNumber As Integer) As Integer

@ Parameter

CardNumber : The module id that want to be released.

@ Return Code

NoError: The function returns successfully.
ErrorInvalidCardNumber : The CardNumber is larger than MAX_USB_DEVICE.
ErrorMemUMapSetIoctl : Failed to Unmap the memory, please call GetLastError() for detailed system error.

2.2.81 UD_Device_Scan

@ Description

This function checks all active USB-DAQ devices in your system. The pModuleNum saves the numbers of active USB-DAQ devices.

@ Modules Support

USB-1901/USB-1902/USB-1903/USB-2401/USB-7250/USB-7230/USB-2405/USB-1210

@ Syntax

Microsoft C/C++ and Borland C++

I16 UD_Device_Scan(U16* pModuleNum, USBDAQ_DEVICE AvailModules[])

@ Parameter

pModuleNum: The pointer to the memory that stores the numbers of active USB-DAQ devices.

AvailableModule: The user-provided USBDAQ_DEVICE array that is save the available USBDAQ modules. The USBDAQ_DEVICE data structure is defined in UsbDask.h.

```
typedef  struct
{
USHORTwModuleType;
USHORTwCardID;
} USBDAQ_DEVICE, *PUSBDAQ_DEVICE
```

@ Return Code

NoError: The function returns successfully.
ErrorNoModuleFound: There is no active device available in your system.

Associated Functions

The associated library, UsbThermo.dll, provides the functions to convert the thermoelectric-voltage to temperature. Plesae refer to the header file, UsbThermo.h, and the related library.

ADC_to_Thermo

@ Description

This function converts the voltage to temperature

@ Syntax

Microsoft C/C++ and Borland C++

I16 ADC_to_Thermo(unsigned short wThermoType, double fScaleADC, double fColdJuncTemp, double* fTemp)

@ Parameter

wThermoType:The thermo-type, the valid types are:
THERMO_B_TYPE,
THERMO_C_TYPE,
THERMO_E_TYPE,
THERMO_K_TYPE,
THERMO_R_TYPE,
THERMO_S_TYPE,
THERMO_T_TYPE,
THERMO_J_TYPE,
THERMO_N_TYPE
RTD_RT100.

fScaleADC: The thermoelectric voltage.

fColdJuncTemp: The temperature for the thermocouple cold-junction compensation.

fTemp: The memory to store the converted temperature.

pfTemp

@ Return Code

NoThermoError: The function returns successfully.
ErrorInvalidThermoType: The thermo-type is not supported.
ErrorOutThermoTange: The thermoelectric-voltage is out of range of the reference-table.
ErrorThernoTable: No suitable entry can be found in the reference table,

Appendix A Status Codes

This appendix lists the status codes returned by UD-DASK, including the name and description.

Each UD-DASK function returns a status code that indicates whether the function was performed successfully. When a UD-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 C*ardType* argument is not valid
-2	ErrorInvalidCardNumber	The CardNumber argument is out of range (larger than 31).
-3	ErrorTooManyCardRegistered	There have been 32 cards that were registered.
-4	ErrorCardNotRegistered	No card registered as id *CardNumber*.
-5	ErrorFuncNotSupport	The function called is not supported by this type of card..
-6	ErrorInvalidIoChannel	The specified *Channel* or *Port* argument is out of range..
-7	ErrorInvalidAdRange	The specified analog input range is invalid.
-8	ErrorContIoNotAllowed	The specified continuous IO operation is not supported by this type of card.
-9	ErrorDiffRangeNotSupport	All the analog input ranges must be the same for multi-channel analog input.
-10	ErrorLastChannelNotZero	The channels for multi-channel analog input must be ended with or started from zero.
-11	ErrorChannelNotDescending	The channels for multi-channel analog input must be contiguous and in descending order.
-12	ErrorChannelNotAscending	The channels for multi-channel analog input must be contiguous and in ascending order.
-13	ErrorOpenDriverFailed	Failed to open the device driver.
-14	ErrorOpenEventFailed	Open event failed in device driver.
-15	ErrorTransferCountTooLarge	The size of transfer is larger than the size of Initially allocated memory in driver.
-16	ErrorNotDoubleBufferMode	Double buffer mode is disabled.
-17	ErrorInvalidSampleRate	The specified sampling rate is out of range.
-18	ErrorInvalidCounterMode	The value of the *Mode* argument is invalid.
-19	ErrorInvalidCounter	The value of the *Ctr* argument is out of range.
-20	ErrorInvalidCounterState	The value of the *State* argument is

		out of range.
-21	ErrorInvalidBinBcdParam	The value of the *BinBcd* argument is invalid.
-22	ErrorBadCardType	The value of Card Type argument is invalid
-23	ErrorInvalidDaRefVoltage	The value of DA reference voltage argument is invalid
-24	ErrorAdTimeOut	Time out for AD operation
-25	ErrorNoAsyncAI	Continuous Analog Input is not set as Asynchronous mode
-26	ErrorNoAsyncAO	Continuous Analog Output is not set as Asynchronous mode
-27	ErrorNoAsyncDI	Continuous Digital Input is not set as Asynchronous mode
-28	ErrorNoAsyncDO	Continuous Digital Output is not set as Asynchronous mode
-29	ErrorNotInputPort	The value of AI/DI port argument is invalid
-30	ErrorNotOutputPort	The value of AO/DO argument is invalid
-31	ErrorInvalidDioPort	The value of DI/O port argument is invalid
-32	ErrorInvalidDioLine	The value of DI/O line argument is invalid
-33	ErrorContIoActive	Continuous IO operation is not active
-34	ErrorDblBufModeNotAllowed	Double Buffer mode is not allowed
-35	ErrorConfigFailed	The specified function configuration is failed
-36	ErrorInvalidPortDirection	The value of DIO port direction argument is invalid
-37	ErrorBeginThreadError	Failed to create thread
-38	ErrorInvalidPortWidth	The port width setting 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
-50	ErrorInvalidCounterValue	The value of count for a counter is invalid.
-60	ErrorInvalidEventHandle	The event handle is invalid.
-61	ErrorNoMessageAvailable	No event message can be added.
-62	ErrorEventMessgaeNotAdded	The specified event message does not exist.
-63	ErrorCalibrationTimeOut	Auto-calibration has timed-out.
-64	ErrorUndefinedParameter	Parameter(s) is not defined.
-65	ErrorInvalidBufferID	Buffer ID is invalid.
-66	ErrorInvalidSampledClock	The set sampled clock is invalid.
-67	ErrorInvalisOperationMode	The set operation mode is invalid.
-201	ErrorConfigIoctl	The configuration API failed.
-202	ErrorAsyncSetIoctl	The async. mode API failed.
-203	ErrorDBSetIoctl	The double-buffer setting API failed.
-204	ErrorDBHalfReadyIoctl	The half-ready API failed.
-205	ErrorContOPIoctl	The continuous data acquisition API failed.
-206	ErrorContStatusIoctl	continuous data acquisition status API setting failed.

-207	ErrorPIOIoctl	The polling data API failed.
-208	ErrorDIntSetIoctl	The dual-interrupt setting API failed.
-209	ErrorWaitEvtIoctl	The wait event API failed.
-210	ErrorOpenEvtIoctl	The open event API failed.
-211	ErrorCOSIntSetIoctl	The COS interrupt setting API failed.
-212	ErrorMemMapIoctl	The memory mapping API failed.
-213	ErrorMemUMapSetIoctl	The memory unmapping API failed.
-214	ErrorCTRIoctl	The counter API failed.
-215	ErrorGetResloctl	The resource getting API failed.
-216	ErrorCalloctl	The calibration API failed.
-301	ErrorAccessViolationDataCopy	Indicates the system exception is occurred while memory-copying.
-302	ErrorNoModuleFound	There is no active device available in your system.
-303	ErrorCardIDDuplicated	Indicates the same ID is configured in multiple modules.
-304	ErrorCardDisconnected	Indicates the USB device was disconnected.
-305	ErrorInvalidScannedIndex	<The relative function had been removed. reserved for future use.>
-306	ErrorUndefinedException	Indicates the undefined exception is caught, usually returned in beta version.
-307	ErrorInvalidDioConfig	Invalid setting in DIO configuration.
-308	ErrorInvalidAOCfgCtrl	The invalid settings in AO Control Configuration.
-309	ErrorInvalidAOTrigCtrl	The invalid settings in AO Trigger Configuration.
-310	ErrorConflictWithSyncMode	The synchronous AI/AO operation is conflict with this function.
-311	ErrorConflictWithFifoMode	<The relative function had been removed, reserved for future use.>
-312	ErrorInvalidAOIteration	The Iteration is zero with finite operation.
-313	ErrorZeroChannelNumber	The number of channel is zero.
-314	ErrorSystemCallFailed	Failed to forward the command to driver, please call GetLastError() for detailed system-error.
-315	ErrorTimeoutFromSyncMode	The synchronous AI / AO operation is time-out.
-316	ErrorInvalidPulseCount	The Pulse-Connt is zero when MultipleGatedPulseGen mode is selected
-317	ErrorInvalidDelayCount	The delay-count is less than 1, or less than 320 if the trigger-source is configured as P1902_AI_TRGSRC_AI0 ~ P1902_AI_TRGSRC_AI15.
-318	ErrorConflictWithDelay2	The P1902_AO_EnDelay2 in UD_AO_1902_Config() needs at least 2 iterations.
-319	ErrorAOFifoCountTooLarge	The Write-Count is larger than onboard FIFO size.
-320	ErrorConflictWithWaveRepeat	<The relative function had been

		removed, reserved for future use.>
-321	ErrorConflictWithReTrig	The re-trigger is manual-exclusive to repeating D/A operation.
-322	ErrorInvalidTriggerChannel	The analog-trigger is not the first channel in Channel-Gain-Queue. Please make sure the trigger channel is identical to the Channel parameter.
-323	ErrorInvalidInputSignal	Indicates the invalid input-signal is assigned.
-324	ErrorInvalidConversionSrc	<The relative function had been removed, reserved for future use.>
-325	ErrorInvalidRefVoltage	The measured voltage is invalid for the specification calibration operation. (this error only for the calibration related functions, now no auto-calibration function is added.)
-326	ErrorCalibrateFailed	Calibration failed. (this error only for the calibration related functions, now no auto-calibration function is added.)
-327	ErrorInvalidCalData	The input calibration data is invalid. (this error only for the calibration related functions, now no auto-calibration function is added.)
-328	ErrorChanGainQueueTooLarge	The numChans is too large.
-329	ErrorInvalidCardType	Indicates the module-type is invalid.
-397	ErrorInvalidChannel	<Now is used by the beta function, UD_AI_Moving_Average32().> In UD_AI_Moving_Average32(), indicates the target-channel is larger than total-channels.
-398	ErrorNullPoint	<Now is used by the beta function, UD_AI_Moving_Average32().> In UD_AI_Moving_Average32(), indicates either SrcBuf or DesBuf is NULL.
-399	ErrorInvalidParamSetting	Indicates some parameters are invalid. This error has different definition in functions.
-401	ErrorAIStartFailed	Indicates the AI acquisition had been started, but the relevant status cannot be read from FPGA. (this error ought not to be returned)
-402	ErrorAOStartFailed	Indicates the AO acquisition had been started, but the relevant status cannot be read from FPGA. (this error ought not to be returned)
-403	ErrorConflictWithGPIOConfig	Incorrect GPIO configuration, please check the settings in

		UD_DIO_1902_Config() / UD_DIO_2401_Config().
-404	ErrorEepromReadback	Indicates the failure in Calibration data/information writing (this error only for the calibration related functions, now no auto-calibration function is added.)
-405	ErrorConflictWithInfiniteOp	The infinite AI operation is only supported by double-buffered acquisition.
-406	ErrorWaitingUSBHostResponse	This error is usually caused by trigger-enabled AI/AO operation. Call UD_AI_AsyncClear()/UD_AO_AsyncClear() to disable the waiting state.
-407	ErrorAOFifoModeTimeout	The D/A data transmission timeout with FIFO mode.
-408	ErrorInvalidModuleFunction	Indicate the specific function is not supported by this module.
-409	ErrorAdFifoFull	Indicates the occurrence of FIFO overrun.
-410	ErrorInvalidTransferCount	The ReadCount is not multiple of 256/512 (for USB-190x), 128/256 (for USB-2401).
-411	ErrorConflictWithAIConfig	The AdRange is conflict with the some specific input-type.
-412	ErrorDDSConfigFailed	The DDS configuration failed (for US
-413	ErrorFpgaAccessFailed	Failed to access FPGA
-414	ErrorPLDBusy	PLD is busy
-415	ErrorPLDTimeout	PLD access timeout
-420	ErrorUndefinedKernelError	The error returned from kernel is undefined (this error ought not to be returned, usually caused by incompatible error-definitions between driver and library)
-501	ErrorSyncModeNotSupport	Synchronization operation is not supported yet. (usually returned in beta verion)
-601	ErrorInvalidThermoType	Thermo type is not supported.
-602	ErrorOutThermoRange	Voltage out of thermo table range
-603	ErrorThermoTable	Error inside the thermo table

Appendix B AI Range Codes

The ***Analog Input Range*** of NuDAQ PCI-bus Cards

AD_B_10_V	Bipolar -10V to +10V
AD_B_5_V	Bipolar -5V to +5V
AD_B_2_5_V	Bipolar -2.5V to +2.5V
AD_B_1_25_V	Bipolar -1.25V to +1.25V
AD_B_0_625_V	Bipolar -0.625V to +0.625V
AD_B_0_3125_V	Bipolar -0.3125V to +0.3125V
AD_B_0_5_V	Bipolar -0.5V to +0.5V
AD_B_0_05_V	Bipolar -0.05V to +0.05V
AD_B_0_005_V	Bipolar -0.005V to +0.005V
AD_B_1_V	Bipolar -1V to +1V
AD_B_0_1_V	Bipolar -0.1V to +0.1V
AD_B_0_01_V	Bipolar -0.01V to +0.01V
AD_B_0_001_V	Bipolar -0.01V to +0.001V
AD_U_20_V	Unipolar 0 to +20V
AD_U_10_V	Unipolar 0 to +10V
AD_U_5_V	Unipolar 0 to +5V
AD_U_2_5_V	Unipolar 0 to +2.5V
AD_U_1_25_V	Unipolar 0 to +1.25V
AD_U_1_V	Unipolar 0 to +1V
AD_U_0_1_V	Unipolar 0 to +0.1V
AD_U_0_01_V	Unipolar 0 to +0.01V
AD_U_0_001_V	Unipolar 0 to +0.001V
AD_B_2_V	Bipolar -2V to +2V
AD_B_0_25_V	Bipolar -0.25V to +0.25V
AD_B_0_2_V	Bipolar -0.2V to +0.2V
AD_U_4_V	Unipolar 0 to +4V
AD_U_2_V	Unipolar 0 to +2V
AD_U_0_5_V	Unipolar 0 to +0.5V
AD_U_0_4_V	Unipolar 0 to +0.4V
AD_B_1_5_V	Bipolar -1.5V to +1.5V
AD_B_0_2125_V	Bipolar -0.2125V to +0.2125V
AD_B_40_V	Bipolar -40V to +40V
AD_B_3_16_V	Bipolar -3.16V to +3.16V
AD_B_0_316_V	Bipolar -0.316V to +0.316V
AD_B_25_V	Bipolar -25V to +25V
AD_B_12_5_V	Bipolar -12.5V to +12.5V

Valid values for each card:

USB-1901	AD_B_10_V, AD_B_2_V, AD_B_1_V, AD_B_0_2_V,
USB-1902	AD_B_10_V, AD_B_2_V, AD_B_1_V, AD_B_0_2_V,
USB-1903	:AD_B_10_V
USB-2401	AD_B_25_V, AD_B_12_5_V, AD_B_2_5_V, AD_B_1_25_V
USB-2405	AD_B_10_V
USB-1210	AD_B_10_V, AD_B_2_V

Appendix C AI DATA FORMAT

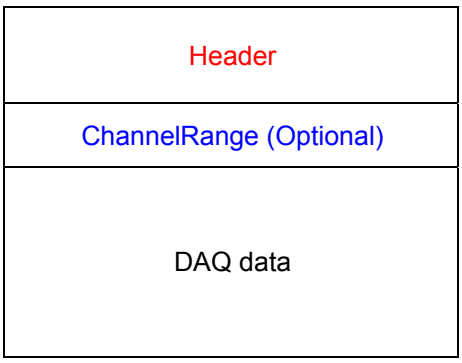
This appendix lists the AI data format for the cards performing analog input operation, as well as the calculation methods to retrieve the A/D converted data and the channel where the data read from.

Card Type	**Data Format**	**AI type**	**Value calculation** ** channel no. (CH#)* ** A/D converted data (ND)* ** Value returned from AI function (OD)*
UBS-1901	*Every 16-bit signed integer data:* *D15 D14 ... D1 D0* *where D15, D14, ... , D0 : A/D converted data*	One-Shot AI Continuous AI I	ND = OD
UBS-1902	*Every 16-bit signed integer data:* *D15 D14 ... D1 D0* *where D15, D14, ... , D0 : A/D converted data*	One-Shot AI Continuous AI I	ND = OD
UBS-1903	*Every 16-bit signed integer data:* *D15 D14 ... D1 D0* *where D15, D14, ... , D0 : A/D converted data*	One-Shot AI Continuous AI I	ND = OD
UBS-2401	*Every 24-bit signed long data:* *D23 D22 ... D1 D0* *where D23, D22, ... , D0 : A/D converted data*	One-Shot AI Continuous AI I	ND = OD
USB-2405	*Every 24-bit signed long data:* *D23 D22 ... D1 D0* *where D23, D22, ... , D0 : A/D converted data*	One-Shot AI Continuous AI I	ND = OD
USB-1210	*Every 16-bit signed integer data:* *D15 D14 ... D1 D0* *where D15, D14, ... , D0 : A/D converted data*	One-Shot AI Continuous AI I	ND = OD

Appendix D DATA File FORMAT

This appendix describes the file format of the data files generated by the functions performing continuous data acquisition followed by storing the data to disk.

The data file includes three parts, Header, ChannelRange (optional) and Data block. The file structure is as the figure below:



Header

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

Header			*Total Length: 60 bytes*
Elements	**Type**	**Size (bytes)**	**Comments**
ID	char	10	file ID *ex. ADLinkDAQ1*
card_type	short	2	card Type *ex. USB_1901, USB_1902*
num_of_channel	short	2	number of scanned channels *ex. 1, 2*
channel_no	unsigned char	1	channel number where the data read from (only available as the num_of_channel is 1) *ex. 0, 1*
num_of_scan	long	4	the number of scan for each channel (total count / num_of_channel)
data_width	short	2	the data width 0: 8 bits, 1: 16 bits, 2: 32 bits
channel_order	short	2	the channel scanned sequence 0: normal (ex. 0-1-2-3) 1: reverse (ex. 3-2-1-0) 2: custom* (ex. 0, 1, 3)
ad_range	short	2	the AI range code Please refer to Appexdix B

			ex. 0 (AD_B_5V)
scan_rate	double	8	The scanning rate of each channel (total sampling rate / num_of_channel)
num_of_channel_range	short	2	The number of ChannelRange* structure
start_date	char	8	The starting date of data acquisition *ex. 12/31/99*
start_time	char	8	The starting time of data acquisition *ex. 18:30:25*
start_millisec	char	3	The starting millisecond of data acquisition *ex. 360*
reserved	char	6	not used

* If the *num_of_channel_range* is 0, the *ChannelRange* block won't be included in the data file.
* The *channel_order* is set to "custom" only when the card supports variant channel scanning order.

ChannelRange

The *ChannelRange* part records the channel number and data range information related to the stored data. This part consists of several channel & range units. The length of each unit is 2 bytes. The total length depends on the value of *num_of_channel_range* (one element of the file header) and is calculated as the following formula:

Total Length = *2* * num_of_channel_range bytes

The data structure of each ChannelRange unit is as follows:

ChannelRange Unit *Length: 2 bytes*			
Elements	**Type**	**Size (bytes)**	**Comments**
channel	char	1	scanned channel number *ex. 0, 1*
range	char	1	the AI range code of *channel* Please refer to Appexdix B *ex. 0 (AD_B_5V)*

Data Block

The last part is the data block. The data is written to file in 16-bit binary format, with the lower byte first (little endian). For example, the value 0x1234 is written to disk with 34 first followed by 12. The total length of the data block depends on the data width and the total data count.

The file is written in Binary format and can't be read in normal text editor. You can use any binary file editor to view it or the functions used for reading files, e.g. fread, to get

the file information and data value.

Appendix E Function Support

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

Function \ Board	USB-1901	USB-1902	USB-1903	USB-2401	USB-7250	USB-7230	USB-2405	USB-1210									
UD_AI_1902_Config	●	●	●														
UD_AI_2401_Config				●													
UD_AI_2401_PollConfig				●													
UD_AI_2405_Chan_Config							●										
UD_AI_2405_Trig_Cofnig							●										
UD_AI_Channel_Config								●									
UD_AI_Trigger_Cofnig								●									
UD_AI_1902_CounterInterval	●	●	●														
UD_AI_DDS_ActualRate_Get							●										
UD_AI_AsyncCheck	●	●	●	●			●	●									
UD_AI_AsyncClear	●	●	●	●			●	●									
UD_AI_AsyncDblBufferHalfReady	●	●	●	●			●	●									
UD_AI_AsyncDblBufferMode	●	●	●	●			●	●									
UD_AI_AsyncDblBufferTransfer	●	●	●					●									
UD_AI_AsyncDblBufferTransfer32				●			●										
UD_AI_AsyncDblBufferOverrun	●	●	●	●			●	●									
UD_AI_AsyncDblBufferHandled	●	●	●	●			●	●									
UD_AI_AsyncDblBufferToFile	●	●	●	●			●	●									
UD_AI_AsyncReTrigNextReady	●	●	●				●	●									
UD_AI_ContReadChannel	●	●	●	●			●	●									
UD_AI_ContReadMultiChannels	●	●	●	●			●	●									
UD_AI_ContReadChannelToFile	●	●	●	●			●	●									
UD_AI_ContReadMultiChannelsToFile	●	●	●	●			●	●									
UD_AI_VoltScale	●	●	●					●									
UD_AI_VoltScale32							●										
UD_AI_ContVScale	●	●	●					●									
UD_AI_ContVScale32							●										
UD_AI_2401_Scale32				●													
UD_AI_2401_ContVScale32				●													
UD_AI_InitialMemoryAllocated	●	●	●	●			●	●									
UD_AI_ReadChannel	●	●	●				●	●									
UD_AI_VReadChannel	●	●	●				●	●									
UD_AI_ReadMultiChannels	●	●	●	●			●	●									
UD_AI_SetTimeOut	●	●	●	●			●	●									
UD_AI_EventCallBack							●	●									
UD_AI_Moving_Average32				●													
UD_AO_1902_Config		●	●														

Function / Board	USB-1901	USB-1902	USB-1903	USB-2401	USB-7250	USB-7230	USB-2405	USB-1210									
UD_AO_VWriteChannel		●	●														
UD_AO_WriteChannel		●	●														
UD_AO_AsyncCheck		●	●														
UD_AO_AsyncClear		●	●														
UD_AO_AsyncDblBufferHalfReady		●	●														
UD_AO_AsyncDblBufferMode		●	●														
UD_AO_ContBufferCompose		●	●														
UD_AO_AsyncDblBufferTransfer		●	●														
UD_AO_SetTimeOut		●	●														
UD_AO_ContWriteChannel		●	●														
UD_AO_ContWriteMultiChannels		●	●														
UD_AO_InitialMemoryAllocated		●	●														
UD_GPTC_Clear	●	●	●	●			●	●									
UD_GPTC_Control	●	●	●	●			●	●									
UD_GPTC_Setup	●	●	●	●			●	●									
UD_GPTC_Setup_N								●									
UD_GPTC_Read	●	●	●	●				●									
UD_GPTC_Status	●	●	●	●				●									
UD_CTR_ReadEdgeCounter					●	●	●										
UD_CTR_ReadRequency					●	●											
UD_CTR_Control					●	●	●										
UD_CTR_SetMinPulseWidth					●	●											
UD_DIO_1902_Config	●	●	●														
UD_DIO_2401_Config				●													
UD_DIO_2405_Config							●										
UD_DIO_Config								●									
UD_DI_ReadLine	●	●	●	●	●	●	●	●									
UD_DI_ReadPort	●	●	●	●	●	●	●	●									
UD_DI_SetCOSInterrupt32					●	●											
UD_DI_GetCOSLatchData32					●	●											
UD_DI_Control					●	●											
UD_DI_SetMinPulseWidth					●	●											
UD_DO_ReadLine	●	●	●	●	●	●	●	●									
UD_DO_ReadPort	●	●	●	●	●	●	●	●									
UD_DO_WriteLine	●	●	●	●	●	●	●	●									
UD_DO_WritePort	●	●	●	●	●	●	●	●									
UD_DO_SetInitPattern					●	●											
UD_DO_GetInitPattern					●	●											
UD_2405_Calibration							●										
UD_AI_Calibration								●									
UD_Read_ColdJunc_Thermo				●													
UD_Device_Scan	●	●	●	●	●	●	●	●									
UD_Register_Card	●	●	●	●	●	●	●	●									
UD_Release_Card	●	●	●	●	●	●	●	●									