NEXCOM International Co., Ltd.

IoT Automation Solutions Business Group

Mini-PCIe Module

NISK-DIO

User Manual

NEXCOM International Co., Ltd.
Published December 2016

www.nexcom.com

Contents

Preface

Copyright iv
Disclaimer iv
Acknowledgements iv
Regulatory Compliance Statements iv
Declaration of Conformity iv
RoHS Compliance v
Warranty and RMA vi
Safety Information viii
Installation Recommendations viii
Safety Precautions ix
Technical Support and Assistance x
Conventions Used in this Manual x
Global Service Contact Information xi

Chapter 1: General Information

Overview 1
Key Features 1
Application 1
Specification 2
Block Diagram 3
Input Filter Setting 3
Pin Assignment 4

Chapter 2: Initial Setup

Section 1. Before Installation 5
Section 2. Hardware Installation 5
Section 3. Driver Installation 9
Section 4. Operation Check 14
Section 5. Troubleshooting 16

Chapter 3: External Connection

Section 1. Connect Diagram 17
Section 2. Connect Input Signal (Sink Type) 18
Section 3. Connect Output Signal (Source Type) 19

Appendix A: NISK-DIO Library User Manual

1. Introduction 20
   1.1 Principles of Programming 21
2. API Reference 22
   2.1 API Overview 22
   2.2 Functions for Initialization 23
      2.2.1 NSKDIO_DeviceInit 23
      2.2.2 NSKDIO_DeviceClose 23
   2.3 Device Information Functions 24
      2.3.1 NSK_GetDeviceCount 24
      2.3.2 NSK_GetGetDeviceInfo 24

2.4 Read/Write Functions....26
2.4.1 NSKDIO_SetDO....26
2.4.2 NSKDIO_GetDO....27
2.4.3 NSKDIO_GetDI....28
2.5 Device Version Functions....29
2.5.1 NSKDIO_GetDriverVersion....29
2.5.2 NSKDIO_GetLibraryVersion....30
2.6 Error Codes....31
3. Visual Studio Programming Example....32
3.1 How to Use the NISK-DIO Library....32
3.2 Programming Example....34

PREFACE

Copyright

Disclaimer

The information in this document is subject to change without prior notice and does not represent commitment from NEXCOM International Co., Ltd. However, users may update their knowledge of any product in use by constantly checking its manual posted on our website: http://www.nexcom.com. NEXCOM shall not be liable for direct, indirect, special, incidental, or consequential damages arising out of the use of any product, nor for any infringements upon the rights of third parties, which may result from such use. Any implied warranties of merchantability or fitness for any particular purpose is also disclaimed.

Acknowledgements

NISK-DIO is a trademark of NEXCOM International Co., Ltd. All other product names mentioned herein are registered trademarks of their respective owners.

Regulatory Compliance Statements

This section provides the FCC compliance statement for Class A devices and describes how to keep the system CE compliant.

Declaration of Conformity

FCC

This equipment has been tested and verified to comply with the limits for a Class A digital device, pursuant to Part 15 of FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. Operation of this equipment in a residential area (domestic environment) is likely to cause harmful interference, in which case the user will be required to correct the interference (take adequate measures) at their own expense.

CE

The product(s) described in this manual complies with all applicable European Union (CE) directives if it has a CE marking. For computer systems to remain CE compliant, only CE-compliant parts may be used. Maintaining CE compliance also requires proper cable and cabling techniques.

RoHS Compliance

NEXCOM RoHS Environmental Policy and Status Update

NEXCOM is a global citizen for building the digital infrastructure. We are committed to providing green products and services, which are compliant with European Union RoHS (Restriction on Use of Hazardous Substance in Electronic Equipment) directive 2011/65/EU, to be your trusted green partner and to protect our environment.

RoHS restricts the use of Lead (Pb) < 0.1% or 1,000ppm, Mercury (Hg) < 0.1% or 1,000ppm, Cadmium (Cd) < 0.01% or 100ppm, Hexavalent Chromium (Cr6+) < 0.1% or 1,000ppm, Polybrominated biphenyls (PBB) < 0.1% or 1,000ppm, and Polybrominated diphenyl Ethers (PBDE) < 0.1% or 1,000ppm.

In order to meet the RoHS compliant directives, NEXCOM has established an engineering and manufacturing task force to implement the introduction of green products. The task force will ensure that we follow the standard NEXCOM development procedure and that all the new RoHS components and new manufacturing processes maintain the highest industry quality levels for which NEXCOM are renowned.

The model selection criteria will be based on market demand. Vendors and suppliers will ensure that all designed components will be RoHS compliant.

How to recognize NEXCOM RoHS Products?

For existing products where there are non-RoHS and RoHS versions, the suffix "(LF)" will be added to the compliant product name.

All new product models launched after January 2013 will be RoHS compliant. They will use the usual NEXCOM naming convention.

Warranty and RMA

NEXCOM Warranty Period

NEXCOM manufactures products that are new or equivalent to new in accordance with industry standard. NEXCOM warrants that products will be free from defect in material and workmanship for 2 years, beginning on the date of invoice by NEXCOM. HCP series products (Blade Server) which are manufactured by NEXCOM are covered by a three year warranty period.

NEXCOM Return Merchandise Authorization (RMA)

- Customers shall enclose the "NEXCOM RMA Service Form" with the returned packages.
- Customers must collect all the information about the problems encountered and note anything abnormal or, print out any on-screen messages, and describe the problems on the "NEXCOM RMA Service Form" for the RMA number apply process.
- Customers can send back the faulty products with or without accessories (manuals, cable, etc.) and any components from the card, such as CPU and RAM. If the components were suspected as part of the problems, please note clearly which components are included. Otherwise, NEXCOM is not responsible for the devices/parts.
- Customers are responsible for the safe packaging of defective products, making sure it is durable enough to be resistant against further damage and deterioration during transportation. In case of damages occurred during transportation, the repair is treated as "Out of Warranty."
- Any products returned by NEXCOM to other locations besides the customers' site will bear an extra charge and will be billed to the customer.

Repair Service Charges for Out-of-Warranty Products

NEXCOM will charge for out-of-warranty products in two categories, one is basic diagnostic fee and another is component (product) fee.

Repair Service Charges for Out-of-Warranty Products

NEXCOM will charge for out-of-warranty products in two categories, one is basic diagnostic fee and another is component (product) fee.

System Level

- Component fee: NEXCOM will only charge for main components such as SMD chip, BGA chip, etc. Passive components will be repaired for free, ex: resistor, capacitor.
- Items will be replaced with NEXCOM products if the original one cannot be repaired. Ex: motherboard, power supply, etc.
- Replace with 3rd party products if needed.
- If RMA goods can not be repaired, NEXCOM will return it to the customer without any charge.

Board Level

- Component fee: NEXCOM will only charge for main components, such as SMD chip, BGA chip, etc. Passive components will be repaired for free, ex: resistors, capacitors.
- If RMA goods can not be repaired, NEXCOM will return it to the customer without any charge.

Warnings

Read and adhere to all warnings, cautions, and notices in this guide and the documentation supplied with the chassis, power supply, and accessory modules. If the instructions for the chassis and power supply are inconsistent with these instructions or the instructions for accessory modules, contact the supplier to find out how you can ensure that your computer meets safety and regulatory requirements.

Cautions

Electrostatic discharge (ESD) can damage system components. Do the described procedures only at an ESD workstation. If no such station is available, you can provide some ESD protection by wearing an antistatic wrist strap and attaching it to a metal part of the computer chassis.

Safety Information

Before installing and using the device, note the following precautions:

- Read all instructions carefully.
- Do not place the unit on an unstable surface, cart, or stand.
- Follow all warnings and cautions in this manual.
- When replacing parts, ensure that your service technician uses parts specified by the manufacturer.
- Avoid using the system near water, in direct sunlight, or near a heating device.

Installation Recommendations

Ensure you have a stable, clean working environment. Dust and dirt can get into components and cause a malfunction. Use containers to keep small components separated.

Adequate lighting and proper tools can prevent you from accidentally damaging the internal components. Most of the procedures that follow require only a few simple tools, including the following:

- A Philips screwdriver
- A flat-tipped screwdriver
- A grounding strap
- An anti-static pad

Using your fingers can disconnect most of the connections. It is recommended that you do not use needle-nose pliers to disconnect connections as these can damage the soft metal or plastic parts of the connectors.

Safety Precautions

1. Read these safety instructions carefully.

2. Keep this User Manual for later reference.

3. Disconnect the equipment from any AC outlet before cleaning or installing a component inside the chassis. Use a damp cloth. Do not use liquid or spray detergents for cleaning.

4. To prevent electrostatic build-up, leave the module in its anti-static bag until you are ready to install it.

5. For plug-in equipment, the power outlet socket must be located near the equipment and must be easily accessible.

6. Keep the module away from humidity.

7. Put the module on a stable surface. Dropping it or letting it fall may cause damage.

8. Wear anti-static wrist strap.

9. Do all preparation work on a static-free surface.

10. Make sure the voltage of the power source is correct before connecting the equipment to the power outlet.

11. Hold the module only by its edges. Be careful not to touch any of the components, contacts or connections.

12. All cautions and warnings on the module should be noted.

13. Use the correct mounting screws and do not over tighten the screws.

14. Keep the original packaging and the anti-static bag; in case the board has to be returned for repair or replacement.

Technical Support and Assistance

1. For the most updated information of NEXCOM products, visit NEXCOM's website at www.nexcom.com.

2. For technical issues that require contacting our technical support team or sales representative, please have the following information ready before calling:
   - Product name and serial number
   - Detailed information of the peripheral devices
   - Detailed information of the installed software (operating system, version, application software, etc.)
   - A complete description of the problem
   - The exact wordings of the error messages

Warning!

1. Handling the unit: carry the unit with both hands and handle it with care.

2. Maintenance: to keep the unit clean, use only approved cleaning products or clean with a dry cloth.

Conventions Used in this Manual

Warning:
Information about certain situations, which if not observed, can cause personal injury. This will prevent injury to yourself when performing a task.

Caution:
Information to avoid damaging components or losing data.

Note:
Provides additional information to complete a task easily.

Global Service Contact Information

Headquarters

NEXCOM International Co., Ltd.

9F, No. 920, Chung-Cheng Rd.,
ZhongHe District, New Taipei City, 23586,
Taiwan, R.O.C.
Tel: +886-2-8226-7786
Fax: +886-2-8226-7782
www.nexcom.com

America

USA

NEXCOM USA

2883 Bayview Drive,
Fremont CA 94538, USA
Tel: +1-510-656-2248
Fax: +1-510-656-2158
Email: sales@nexcom.com
www.nexcom.com

Asia

Taiwan

NEXCOM Intelligent Systems

Taipei Office

13F, No.920, Chung-Cheng Rd.,
ZhongHe District,
New Taipei City, 23586, Taiwan, R.O.C.
Tel: +886-2-8226-7796
Fax: +886-2-8226-7792
Email: sales@nexcom.com.tw
www.nexcom.com.tw

NEXCOM Intelligent Systems

Taichung Office

16F, No.250, Sec. 2, Chongde Rd.,
Beitun Dist.,
Taichung City 406, R.O.C.
Tel: +886-4-2249-1179
Fax: +886-4-2249-1172
Email: sales@nexcom.com.tw
www.nexcom.com.tw

Japan

NEXCOM Japan

9F, Tamachi Hara Bldg.,
4-11-5, Shiba Minato-ku,
Tokyo, 108-0014, Japan
Tel: +81-3-5419-7830
Fax: +81-3-5419-7832
Email: sales@nexcom-jp.com
www.nexcom-jp.com

China

NEXCOM China

1F & 2F, Block A, No. 16 Yonyou Software Park,
No. 68 Beiqing Road, Haidian District,
Beijing, 100094, China
Tel: +86-10-5704-2680
Fax: +86-10-5704-2681
Email: sales@nexcom.cn
www.nexcom.cn

NEXCOM Shanghai
Room 603/604, Huiyinmingzun Plaza Bldg., 1,
No.609, Yunlin East Rd.,
Shanghai, 200333, China
Tel: +86-21-5278-5868
Fax: +86-21-3251-6358
Email: sales@nexcom.cn
www.nexcom.cn

NEXCOM Surveillance Technology
Room 209, Floor 2 East, No.2,
Science & Technology industrial park of
privately owned enterprises,
Xili, Nanshan Dist.,
Shenzhen, 518055, China
Tel: +86-755-8364-7768
Fax: +86-755-8364-7738
Email: steveyang@nexcom.com.tw
www.nexcom.cn

NEXCOM United System Service
Hui Yin Ming Zun Building Room 1108,
Building No. 11, 599 Yunling Road, Putuo District,
Shanghai, 200062, China
Tel: +86-21-6125-8282
Fax: +86-21-6125-8281
Email: frankyang@nexcom.cn
www.nexcom.cn

Europe
United Kingdom
NEXCOM EUROPE
10 Vincent Avenue,
Crownhill Business Centre,
Milton Keynes, Buckinghamshire
MK8 0AB, United Kingdom
Tel: +44-1908-267121
Fax: +44-1908-262042
Email: sales.uk@nexcom.eu
www.nexcom.eu

Italy
NEXCOM ITALIA S.r.l
Via Lanino 42,
21047 Saronno (VA), Italia
Tel: +39 02 9628 0333
Fax: +39 02 9625 570
Email: nexcomitalia@nexcom.eu
www.nexcomitalia.it

Chapter 1: General Information

Overview

The NISK-DIO module includes 8 channels of high-power MOSFET outputs with a 30VDC capability and 8 channels isolated digital inputs, 47V high-voltage protection, as well as easy configuration and efficient programming through the Xcare software utility 3.1. With fast output response time and noise or chartering signal removal by adjustable input filter, NISK-DIO is the best choice for industrial applications based on factory automation system PCs.

Key Features

- Mini-PCIe full size form factor (Dimension: 51x30mm) Revision 1.2 compliant
- Support adjustable input filter (10μs/1ms/3.2ms/10ms)
- Fast output response time (within 150μsec)
- High over-voltage-protection (47 VDC) and voltage isolation (500 VDC)
- High source current on isolated output channels (200mA/channel)
- Support Factory Automation system PC applications (NIFE/NISE)
- Support -20~60 degree C operating temperature
- Provide NEXCOM Xcare utility 3.1 for NISK-DIO configuration & programming
- Support Microsoft Windows 7 / 8.1

Application

- Programmable Logic Controllers (PLC)
- Industrial PC
- General Control Equipment
- All types of resistive, inductive and capacitive loads
- Driver for solenoid, relays and resistive loads

Specification

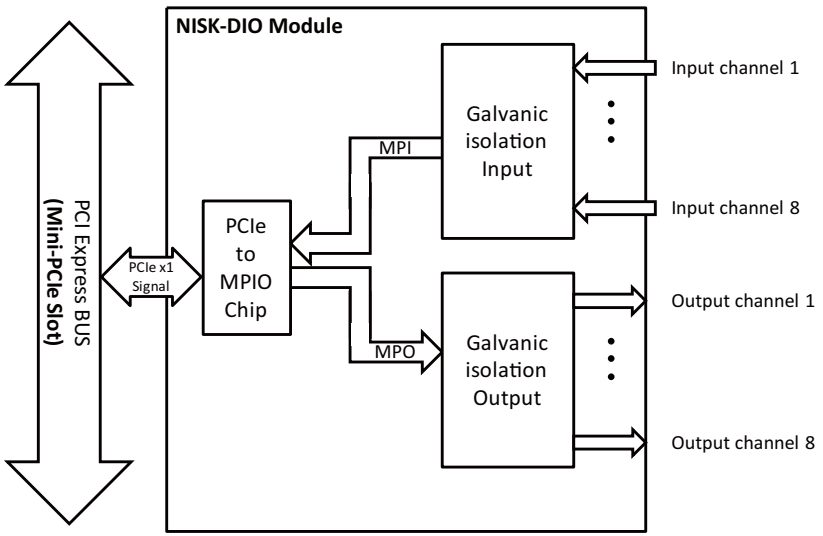
Isolated Digital Input	
Channels	8
Input OFF voltage	0~7 VDC max
Input ON voltage	11~24 VDC max
Input Format	Galvanic isolation Input
Response time	10µsec
Input filter	10µs/1ms/3.2ms/10ms, Select by switch, Default 1ms

Isolated Digital Output	
Channels	8
Output voltage	12~30 VDC
Output current	200 mA max per channel
Output Format	Galvanic isolation Output
Response time	150µsec

General Specification	
Connector	1 x D-Sub 26pin female connector
Isolation Protection	500 VDC
External power supply	24 VDC(±10%)
Operating temperature	-20°C to 60°C
Storage temperature	-40°C to 85°C, relative humidity: 5% to 95% (non-condensing)
Dimensions	Full size Mini-PCIe type: 51mm (W) x 30mm (D) x 1mm (H)
Certifications	CE approved - EN55022 - EN55024 FCC Class A
OS Support	Microsoft® Windows® 7/8.1
Software	Tools & Driver: Xcare 3.1 Tools & API Drivers

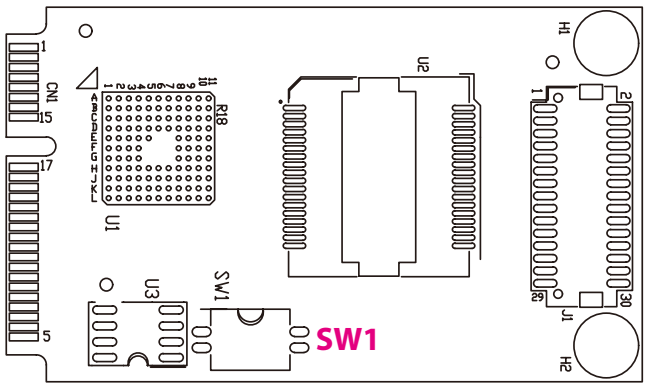
Block Diagram

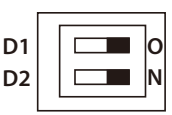
The **NISK-DIO** module uses a PCIe to MPIO chip to transfer the PCIe x1 signal to MPI (Multi-Purpose Input) and MPO (Multi-Purpose Output) signal. It also has a galvanic isolation design to provide isolation protection up to 500VDC. The system block diagram is shown as below.



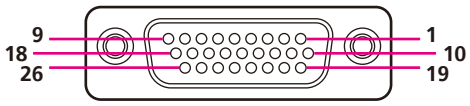
Input Filter Setting

The **NISK-DIO** module can change "Input filter" by switch setting. The switch location and setting is shown as below:



SW1		
		
D1	**D2**	**Input Filter**
ON	ON	10us
ON	OFF	1ms
OFF	ON	3.2ms
OFF	OFF	10ms

Pin Assignment



Pin Definition					
Pin	**Definition**	**Pin**	**Definition**	**Pin**	**Definition**
1	DI_0	10	DO_0	19	DO_VCC
2	DI_1	11	DO_1	20	DO_VCC
3	DI_2	12	DO_2	21	DO_VCC
4	DI_3	13	DO_3	22	DO_VCC
5	DI_4	14	DO_4	23	DO_GND
6	DI_5	15	DO_5	24	DO_GND
7	DI_6	16	DO_6	25	DO_GND
8	DI_7	17	DO_7	26	DO_GND
9	DI_VCC	18	DI_GND		

Chapter 2: Initial Setup

This chapter will show you how to perform the initial setup, from hardware to software. If you bought a **NISK-DIO** package, you can refer to **Section 2: Hardware Installation** to install the module into your system and follow the instructions in **Section 3: Driver Installation** to perform the driver installation and learn how to use the utility. Then we will show you how to do a simple check up on the hardware and software to confirm whether the module is working correctly. Finally, there are some troubleshooting methods that users can take to debug the system. If problems still persist, please contact our technical support.

Section 1. Before Installation

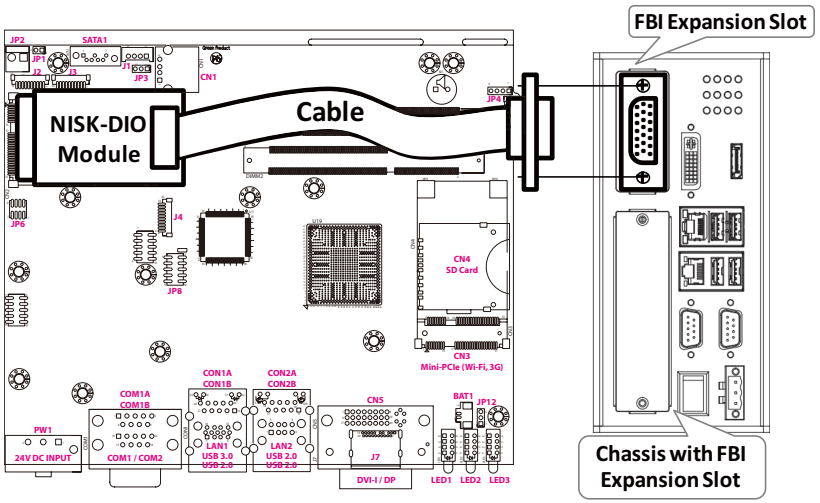
Before you install the **NISK-DIO** module, please check the status of the input filter setting.

Section 2. Hardware Installation

This section will show you how to install **NISK-DIO** into your system.

The **NISK-DIO** package includes the following three pieces: a NISK-DIO module, a cable with connector and a bracket.

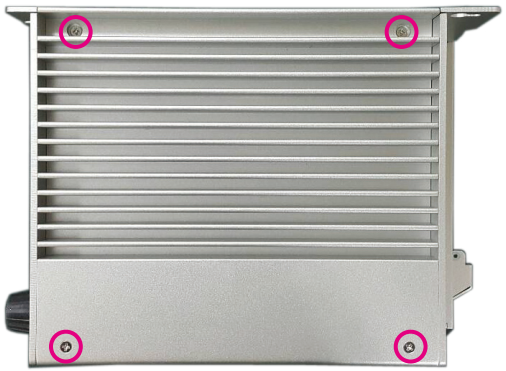
The installation diagram is shown as below:



The following steps show how to install the **NISK-DIO** module into the **NIFE 200P2** system.

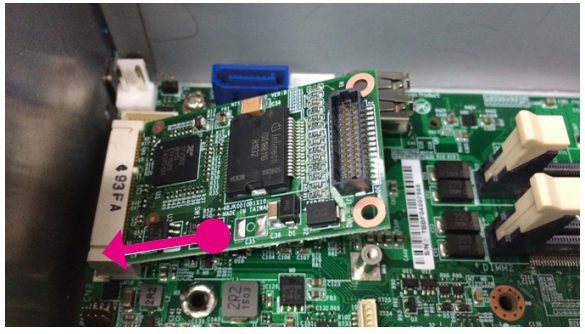
1. Place the **NIFE 200P2** system on a flat surface to prepare for installation.

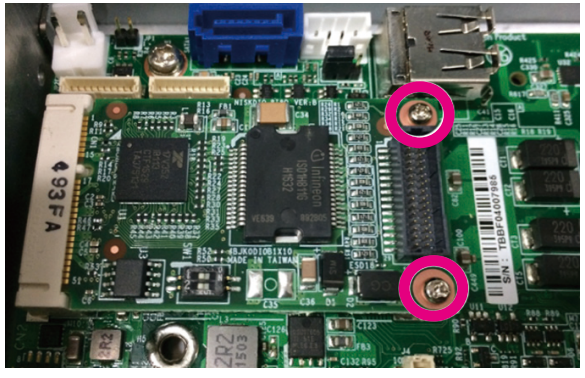
2. Locate the four screws on the side of the chassis and remove them to open the side cover.



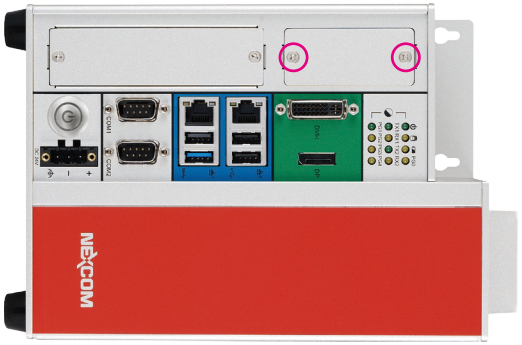
3. Open the side cover and locate the mini-PCIe slot in the system.

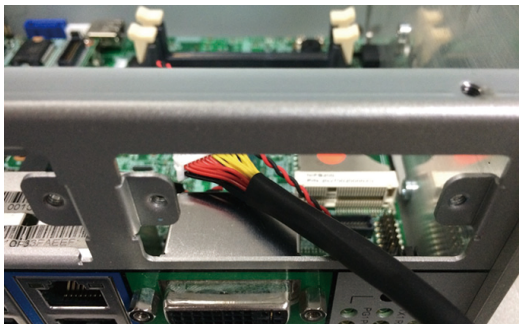
4. Insert the **NISK-DIO** module into the mini-PCIe slot at a 45 degree angle and make sure the gold finger connector is connected correctly, and then use the screw to lock the **NISK-DIO** module onto the **NIFE 200P2** motherboard.



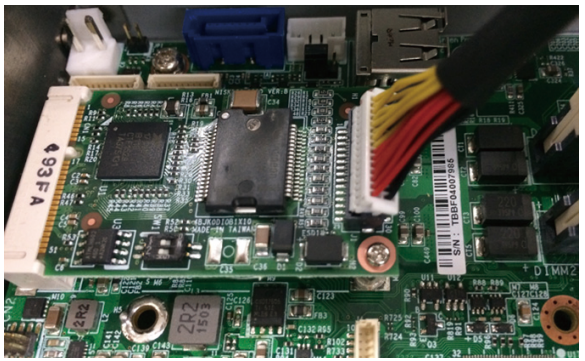


5. Remove the bracket on the **NIFE 200P2** system and let the cable run through the bracket hole into the system.





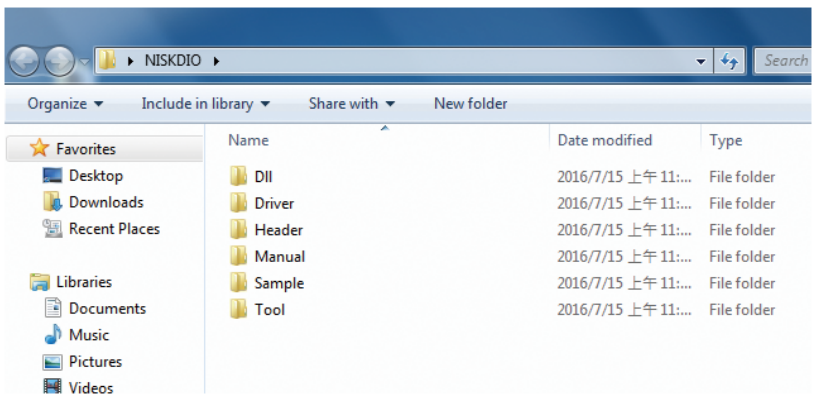
6. Connect the cable to the connector on the **NISK-DIO** module and secure the bracket onto the **NIFE 200P2** chassis.



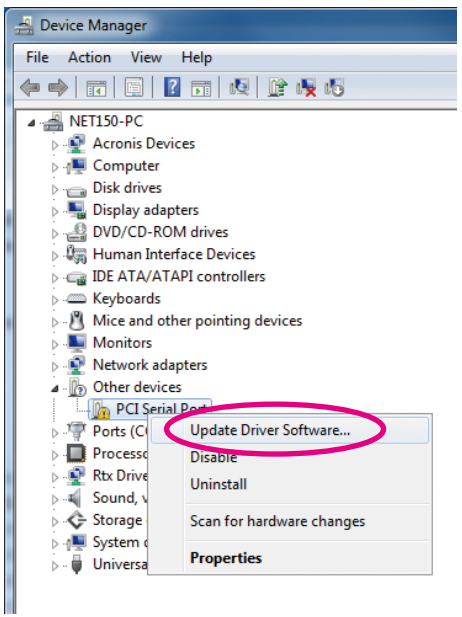
Section 3. Driver Installation

When installing the **NISK-DIO** module and powering on the PC to get into the operating system, you will see an **unknown device** in the **Device Manager** window. Follow the steps below to install the device driver.

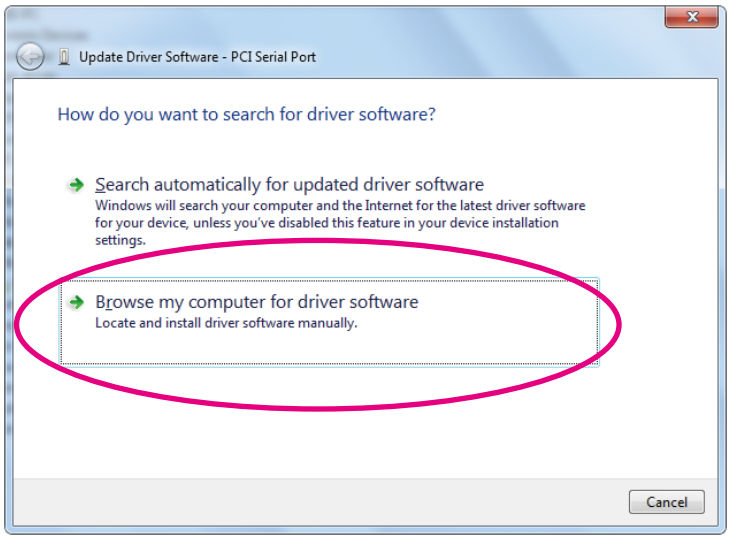
1. Download the driver from the website.
   Please visit NEXCOM's website to download the **NISK-DIO** software and unzip the file to any location you want. The extracted files will have the following content:



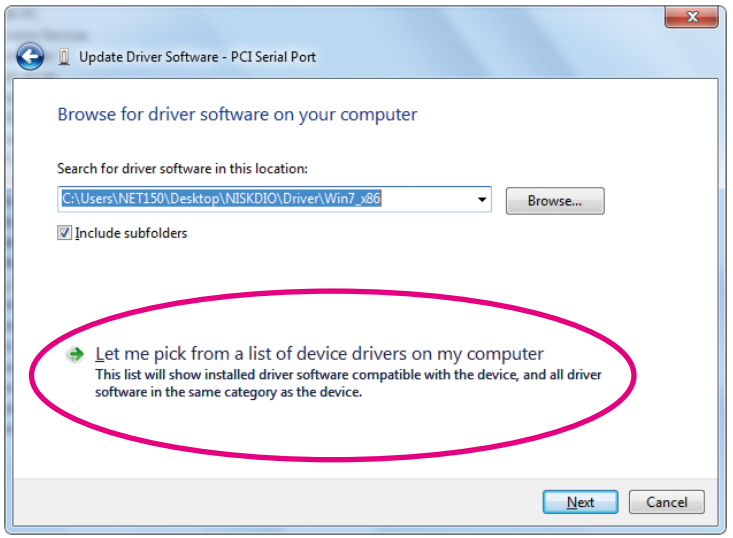
2. Install or update the device driver.
   Go to **Device Manager** and locate the **PCI Serial Port** device under the **Other devices** category. Right-click on **PCI Serial Port** and select **Update Driver Software**.



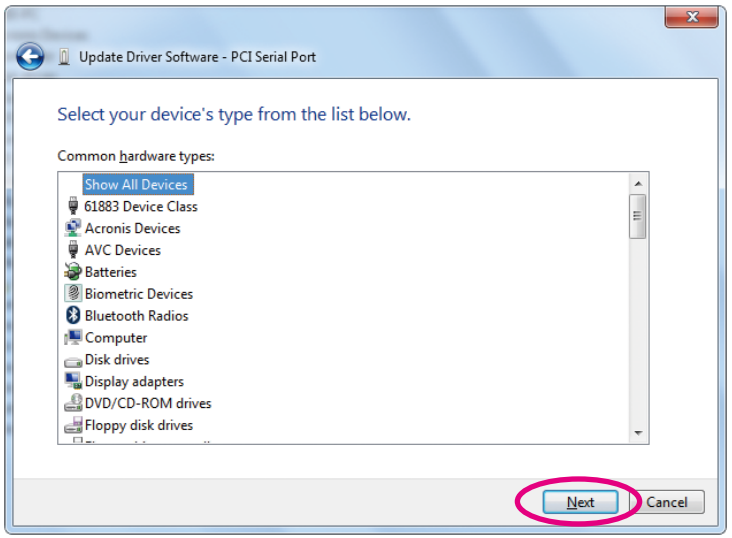
3. Choose the device driver location.
   Select **Browse my computer for driver software** when prompted.



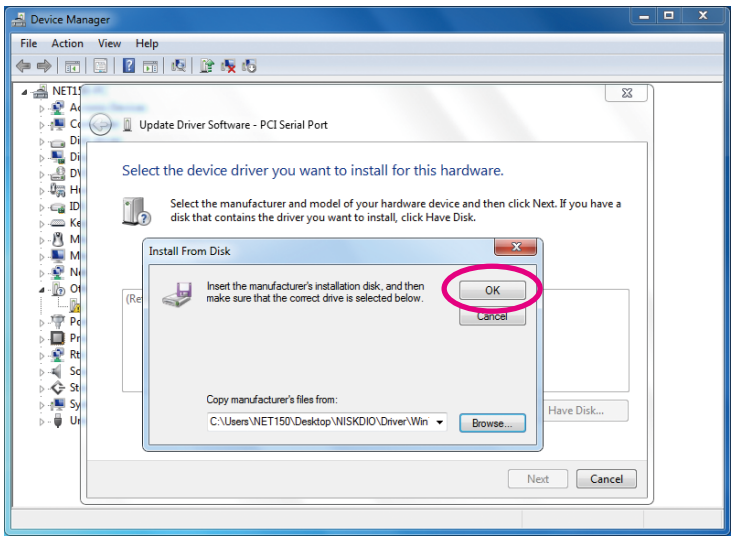
4. Get a device list to show the driver.
   Select **Let me pick from a list of device drivers on my computer** and click **Next**.



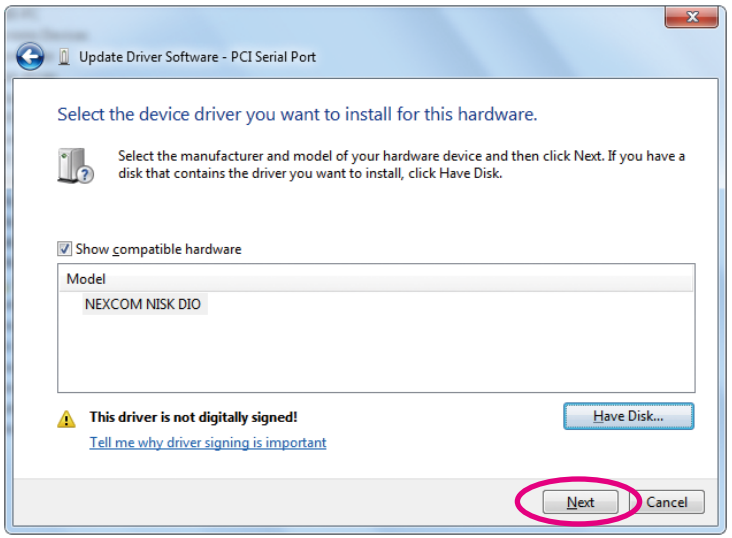
5. Filter all the device driver.
   Select **Show All Devices** and click **Next**.



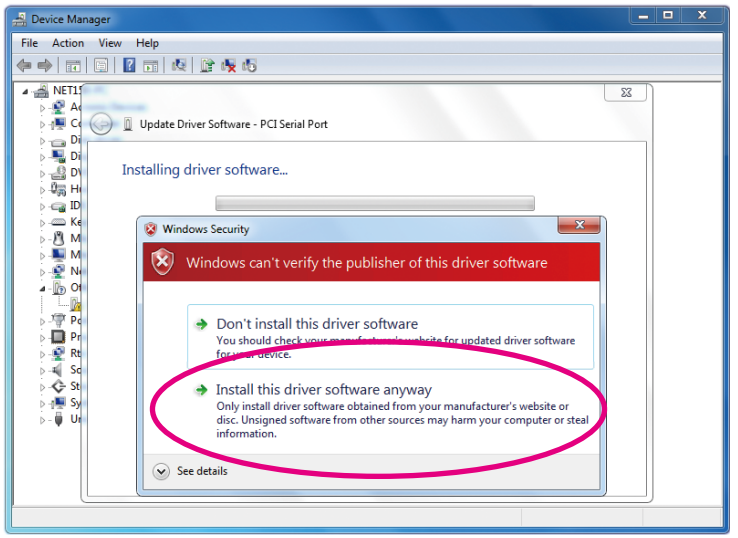
6. Find the driver location.
   Click **Have Disk...➔Browse** and find the driver location, then click **OK**.



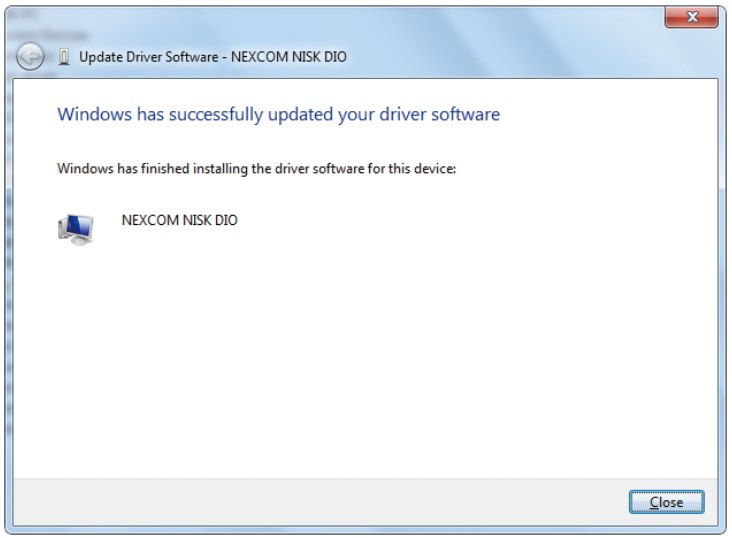
7. Confirm the device driver.
   Confirm the model name **NEXCOM NISK DIO** is shown on the list, then click **Next**.

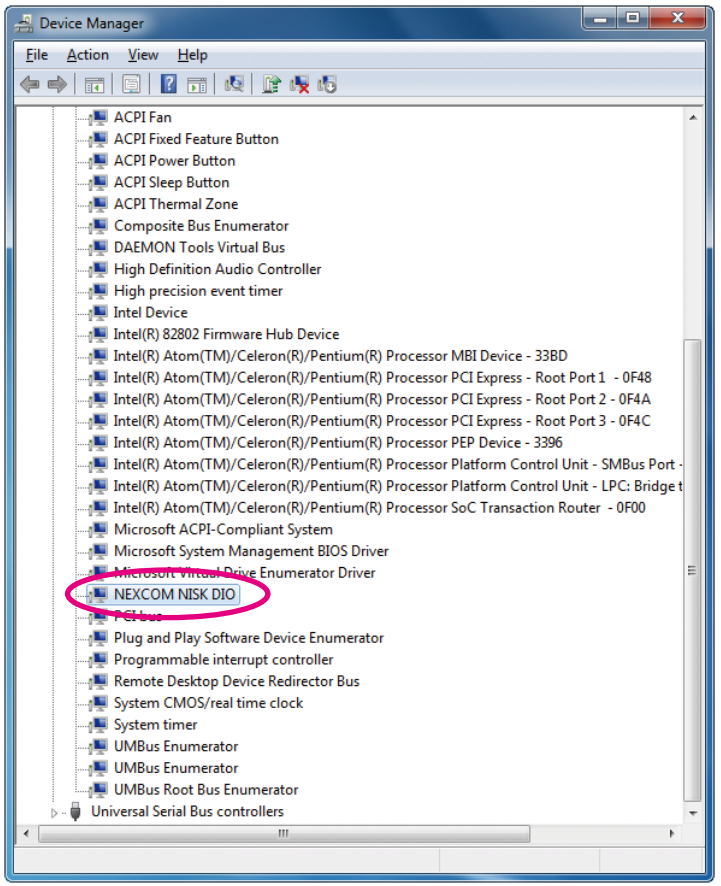


8. Install the driver software.
   When the **Windows Security** pop-up window appears, select **Install this driver software anyway** to install the device driver.



9. Installation complete.
   If the driver has been installed successfully, a completion window will be displayed and the **NISK-DIO** driver can be viewed in **Device Manager** under the **System Devices** category.

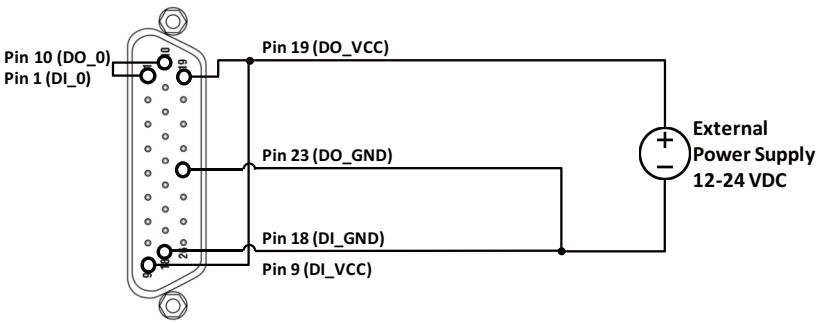




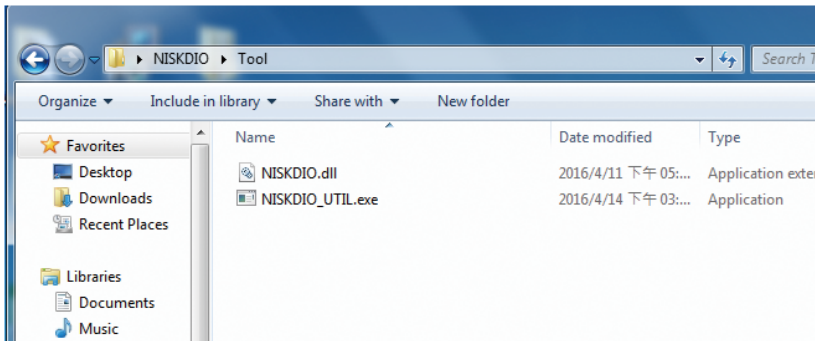
Section 4. Operation Check

When you first open the box, you can perform a simple operation checkup. Please refer to the following steps to find out how to do it.

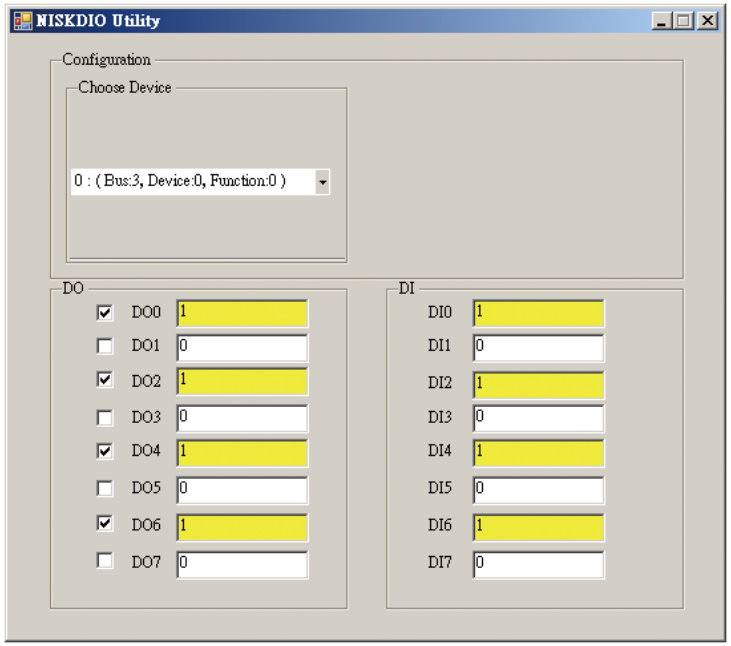
1. Connect the DC power source to Pin 19 & Pin 9, and ground to Pin 23 & Pin 18, and then connect Pin 1 to Pin 10. The wiring diagram is shown as below.



2. Open the **NISK-DIO** module's utility tools.



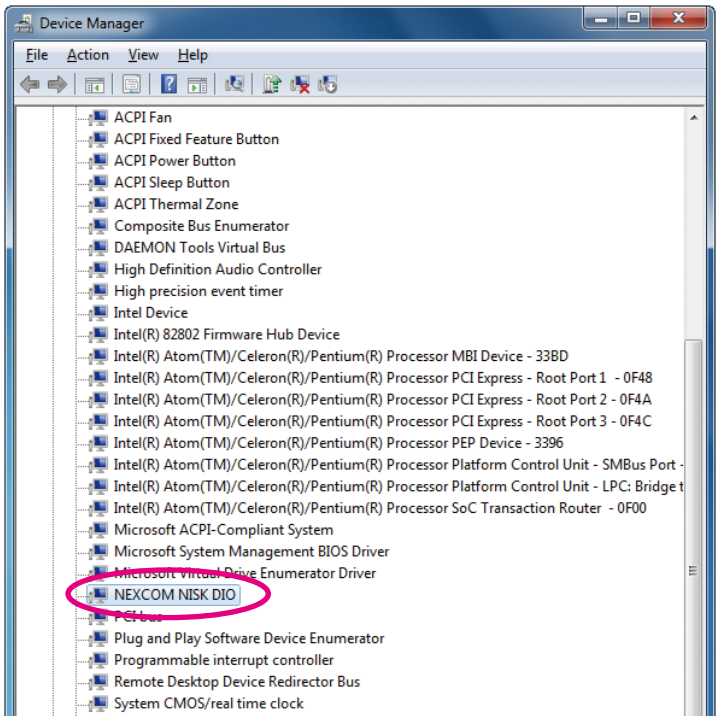
3. Use the utility to set the Digital Output signal ON, then check the Digital Input status is changed by the output signal.



Section 5. Troubleshooting

The following information provides a solution to one of the issues you may encounter when using the **NISK-DIO** module.

1. When the utility is launched, it shows that there is no module found. In Windows, open **Device Manager** and confirm there is a **NEXCOM NISK DIO** device. If the device is not listed, please reinstall the drivers.



Chapter 3: External Connection

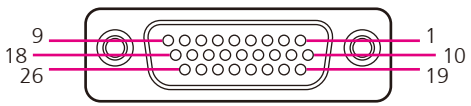
This chapter will show you how to connect a device to the **NISK-DIO** module. The device may be a sensor, a cylinder or a LED. We also show an example of the wiring method to use when connecting the switch as an input device and the LED as an output device.

Section 1. Connect Diagram

When the **NISK-DIO** module is installed in the mini-PCIe slot of the PC, it requires a cable with connector and bracket to pass the I/O signal between the module and external device. This cable has a connector attached to the module while the other end of the cable has a DB 26-pin female connector with bracket mount for installing on the PC's chassis.

If you want to connect the sensor or device to the **NISK-DIO** module, you will need to connect the device's signal via the DB 26-pin connector.

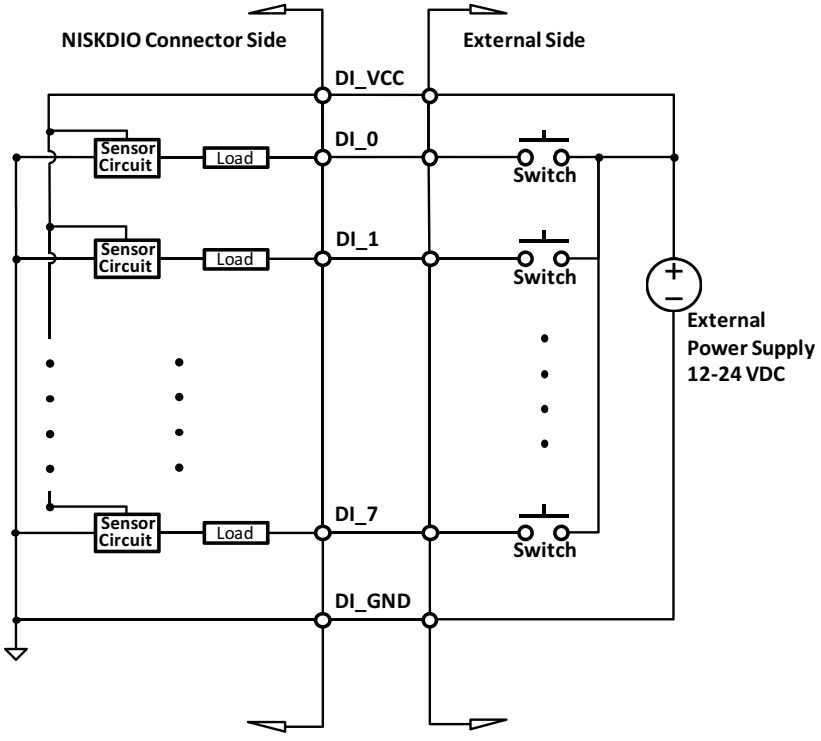
The connector's pin definition is shown in the following table.



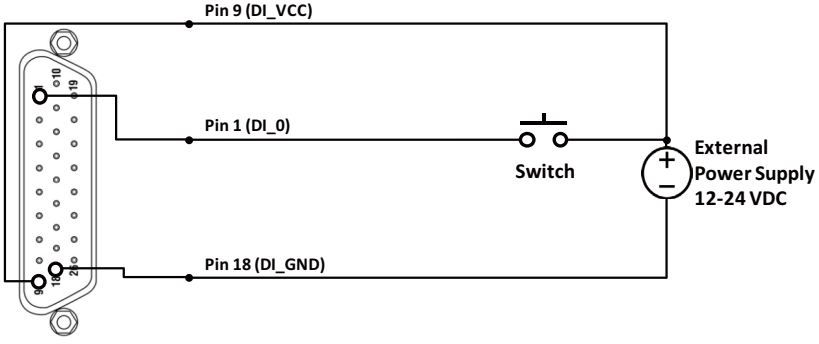
Pin Definition					
Pin	**Definition**	**Pin**	**Definition**	**Pin**	**Definition**
1	DI_0	10	DO_0	19	DO_VCC
2	DI_1	11	DO_1	20	DO_VCC
3	DI_2	12	DO_2	21	DO_VCC
4	DI_3	13	DO_3	22	DO_VCC
5	DI_4	14	DO_4	23	DO_GND
6	DI_5	15	DO_5	24	DO_GND
7	DI_6	16	DO_6	25	DO_GND
8	DI_7	17	DO_7	26	DO_GND
9	DI_VCC	18	DI_GND		

Section 2. Connect Input Signal (Sink Type)

The **NISK-DIO** module's input circuit is shown as below.

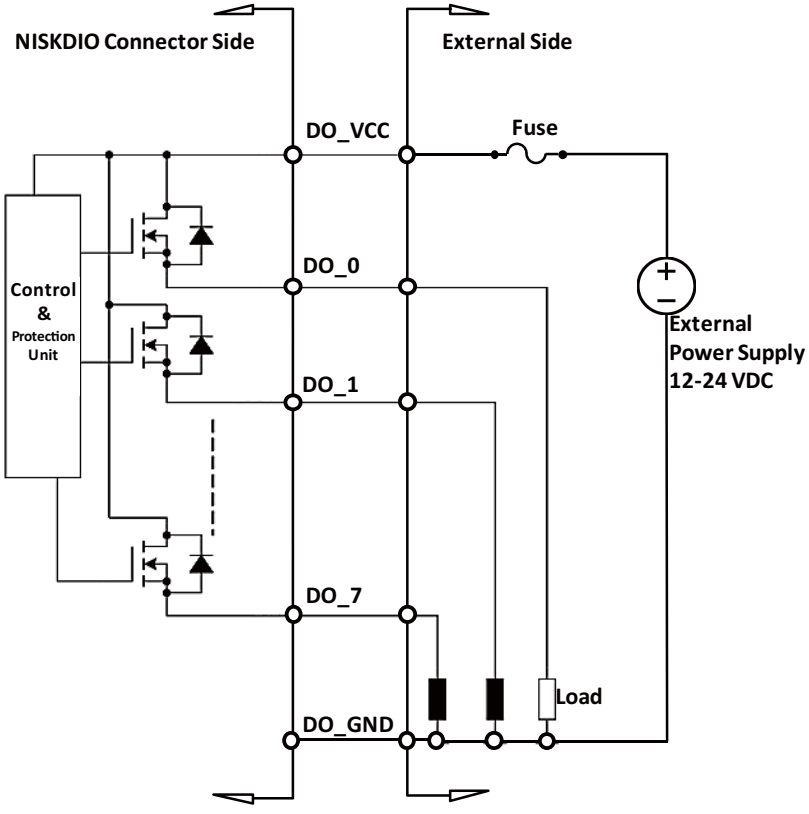


When you connect the input device, we would treat the device as a switch. You could refer to the following diagram to make connection to your own device.

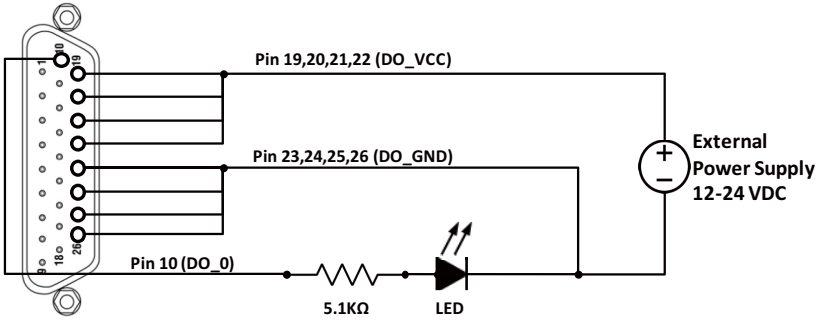


Section 3. Connect Output Signal (Source Type)

The **NISK-DIO** module's output circuit is shown as below.



When you connect the output device, we would treat the device as a LED. You could refer to the following diagram to make connection to your own device.



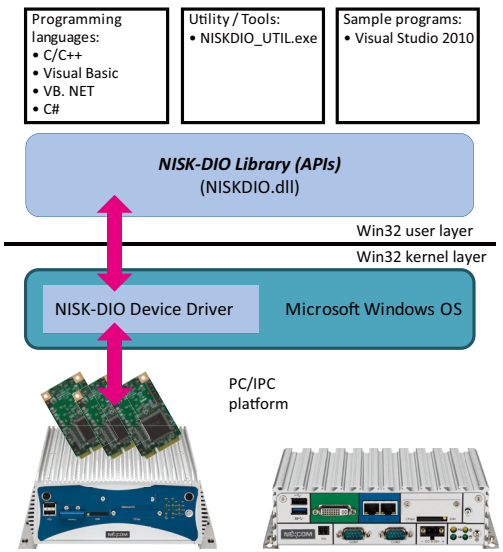
It is suggested to connect all "DO_VCC" pins with power source to provide sufficient current.

Appendix A: NISK-DIO Library User Manual

1. Introduction

NISK-DIO Library is a programming interface for controlling mini-PCIe NISK-DIO devices.

The following figure shows the system architecture of NISK-DIO:



NISK-DIO system architecture

Please refer to Chapter 3 for sample programs that can be used as programming reference.

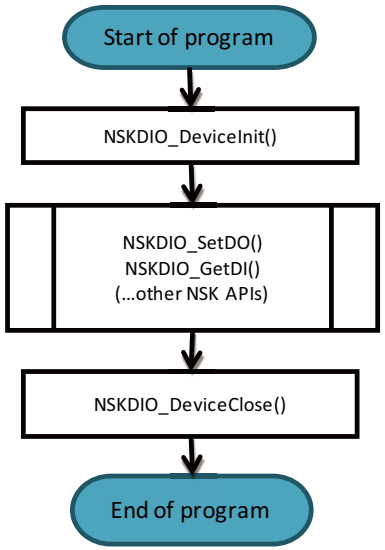
Operating System

The NISK-DIO library (NISKDIO.dll and device driver) supports the following operating system:

Microsoft® Windows® 7 (32-bit)

1.1 Principles of Programming

The basic NISK-DIO library programming flow chart is as the following figure.



1. Before using the NISK-DIO functions, please execute the initial function `NSKDIO_DeviceInit()` first.
2. After finishing programming, please execute the close function `NSKDIO_DeviceClose()` to close the devices.
3. This library (device driver) supports up to 16 NISK-DIO devices in one machine.

2. API Reference

- 2.1 API Overview
- 2.2 Functions for Initialization
- 2.3 Device Information Functions
- 2.4 DIO Control Functions
- 2.5 Device Version Functions

2.1 API Overview

All the APIs of the NISK-DIO library are listed in the table below. The definition of each API is located at the header file "`NISKDIO.h`".

Function Name	Description
Initialization Functions	
NSKDIO_DeviceInit	NISKDIO initial function
NSKDIO_DeviceClose	NISKDIO close function
Device Information Functions	
NSKDIO_GetDeviceCount	Gets how many devices of NISKDIO
NSKDIO_GetDeviceInfo	Gets NISKDIO Bus/Device/Function number
Read/Write DIO Functions	
NSKDIO_SetDO	Sets DO data to NISKDIO
NSKDIO_GetDO	Gets DO data from NISKDIO
NSKDIO_GetDI	Gets DI data from NISKDIO
Device Version Functions	
NSKDIO_GetDriverVersion	Gets NISKDIO Driver Version
NSKDIO_GetLibraryVersion	Gets NISKDIOdll Version

The C/C++ data types for each API are defined in "`nex_type.h`" and listed as follows:

Type	C/C++ Primitive	Format	Byte Length	Value Range
BOOL_T	Int	Boolean	4	0: False, 1: True
U8_T	unsigned char	Unsigned Integer	1	0 ~ 255
U16_T	unsigned short	Unsigned Integer	2	0 ~ 65535
U32_T	unsigned int	Unsigned Integer	4	0 ~ 4294967295
U64_T	unsigned __int64	Unsigned Integer	8	0 ~ 18446744073709551615
I8_T	char	Signed Integer	1	-128 ~ 127
I16_T	short	Signed Integer	2	-32768 ~ 32767
I32_T	int	Signed Integer	4	-2147483648 ~ 2147483647
I64_T	__int64	Signed Integer	8	-9223372036854775808 ~ 9223372036854775807
F32_T	float	Floating-point (FP) number	4	IEEE-754, accurate to the seventh decimal place
F64_T	double	Double-precision FP number	8	IEEE-754, accurate to the fifteenth decimal place
RTN_ERR	int	Error code	4	-2147483648 ~ 2147483647

2.2 Functions for Initialization

- 2.2.1 NSKDIO_DeviceInit
- 2.2.2 NSKDIO_DeviceClose

2.2.1 NSKDIO_DeviceInit

C/C++ Syntax:

```
RTN_ERR NSKDIO_DeviceInit();
```

Parameters:

<no Parameters>

Returned Values:

Error Code is returned.

"RETURN_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed. The Error Code is defined as "ERROR_XXXX" (non-zero) in the header file "NISKDIO.h".

Usage:

This function is used for initializing the library of NISK-DIO.

Attention! The function has to be the first executed function before executing all other functions of the NISK-DIO library.

Reference:

```
NSKDIO_DeviceClose();
```

2.2.2 NSKDIO_DeviceClose

NISK-DIO close function.

C/C++ Syntax:

```
RTN_ERR NSKDIO_DeviceClose();
```

Parameters:

<no Parameters>

Returned Values:

Error Code is returned.

"RETURN_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed. The Error Code is defined as "ERROR_XXXX" (non-zero) in the header file "NISKDIO.h".

Usage:

This function is used for closing the library of NISK-DIO. Typically, this API is executed at the end of application to release the system resources allocated by the NISK-DIO library.

Attention! The function has to be the last executed function after executing all other functions of the NISK-DIO library.

Reference:

```
NSKDIO_DeviceInit();
```

2.3 Device Information Functions

- 2.3.1 NSKDIO_GetDeviceCount
- 2.3.2 NSKDIO_GetDeviceInfo

2.3.1 NSK_GetDeviceCount

Gets how many NISK-DIO devices are on a machine.

C/C++ Syntax:

```
RTN_ERR NSKDIO_GetDeviceCount(U32_T *count);
```

Parameters:

U32_T *count: count is a pointer of U32_T. After executing this function, the total count of the NISK-DIO device would be updated in this parameter.

Returned Values:

Error Code is returned.

"RETURN_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed. The Error Code is defined as "ERROR_XXXX" (non-zero) in the header file "NISKDIO.h".

Usage:

This function is used for getting the total count of the NISK-DIO devices on a machine.

Attention! The function could only be executed after calling the NSKDIO_DeviceInit function.

Reference:

```
NSKDIO_DeviceInit();
```

2.3.2 NSK_GetGetDeviceInfo

Gets the Bus/Device/Function number of NISK-DIO.

C/C++ Syntax:

```
RTN_ERR NSKDIO_GetDeviceInfo(U32_T Device_Id, DEVICE_
                                    INFO_T *pInfo);
```

Parameters:

U32_T DeviceId: DeviceID is the ID of the NISK-DIO device that determines which NISK-DIO device you want to control. DeviceId starts from 0, and depends on the count of the NISK-SIO devices. For example, if there is only one NISK-DIO device on a machine, DeviceID would be 0. If there are two NISK-DIO devices on a machine, DeviceID might be 0 or 1. The range of "DeviceId" is 0 ~ 15.

DEVICE_INFO_T *pInfo: pInfo is a pointer of struct DEVICE_INFO_T. Depending on the first input parameter "DeviceId", after executing this function, Bus number, Device number and Function number of the NISK-DIO device would be updated in this parameter.

```
typedef struct
{
      U8_T   DeviceID;       // Return ID of the device
      U8_T   DeviceBusNum;   // Return (PCI) bus number
      U8_T   DeviceDevNum;   // Return (PCI) dev number
      U8_T   DeviceFuncNum; // Return (PCI) function
number

} DEVICE_INFO_T;
```

Returned Values:

Error Code is returned.

"RETURN_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed. The Error Code is defined as "ERROR_XXXX" (non-zero) in the header file "NISKDIO.h".

Usage:

This function is used for getting the driver version of a NISK-DIO device.

Attention! The function could only be executed after calling the NSKDIO_DeviceInit function.

Reference:

```
NSKDIO_DeviceInit();
```

2.4 Read/Write Functions

- 2.4.1 NSKDIO_SetDO
- 2.4.2 NSKDIO_GetDO
- 2.4.3 NSKDIO_GetDI

2.4.1 NSKDIO_SetDO

Set DO data to NISKDIO.

C/C++ Syntax:

```
RTN_ERR NSKDIO_SetDO(U32_T DeviceId, U32_T
      Offset, U32_T ByteOfLength, U8_T *SetDoData);
```

Parameters:

U32_T DeviceId: DeviceID is the ID of the NISK-DIO device that determines which NISK-DIO device you want to control. DeviceId starts from 0, and depends on the count of the NISK-DIO devices. For example, if there is only one NISK-DIO device on a machine, DeviceID would be 0. If there are two NISK-DIO devices on a machine, DeviceID might be 0 or 1. The range of "DeviceId" is 0 ~ 15.

U32_T Offset: Offset from start address of NISK-DIO memory. The unit is byte. This value must be 0.

U32_T ByteOfLength: ByteOfLength is the length from the offset of the start address that has to be written from user data to the NISK-DIO device. The unit is byte. This value must be 1.

U8_T *SetDoData: Data is a pointer of U8_T, which points to the start address of the user data.

Returned Values:

Error Code is returned.

"RETURN_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed. The Error Code is defined as "ERROR_XXXX" (non-zero) in the header file "NISKDIO.h".

Usage:

This function is used for copying data from user data to the NISK-DIO device.

Attention! The function could only be executed after calling the NSKDIO_DeviceInit function.

Reference:

NSKDIO_DeviceInit();NSKDIO_GetDI();NSKDIO_GetDO()

2.4.2 NSKDIO_GetDO

Get DO data from NISKDIO.

C/C++ Syntax:

```
RTN_ERR NSKDIO_GetDO(U32_T DeviceId, U32_T
       Offset, U32_T ByteOfLength, U8_T *GetDoData);
```

Parameters:

U32_T DeviceId: DeviceID is the ID of the NISK-DIO device that determines which NISK-DIO device you want to control. DeviceId starts from 0, and depends on the count of the NISK-DIO devices. For example, if there is only one NISK-DIO device on a machine, DeviceID would be 0. If there are two NISK-DIO devices on a machine, DeviceID might be 0 or 1. The range of "DeviceId" is 0 ~ 15.

U32_T Offset: Offset from start address of NISK-DIO memory. The unit is byte. This value must be 0.

U32_T ByteOfLength: ByteOfLength is the length from the offset of the start address that has to be read from the NISK-DIO device to user data. The unit is byte. This value must be 1.

U8_T *GetDoData: Data is a pointer of U8_T, which points to the start address of the user data.

Returned Values:

Error Code is returned.

"RETURN_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed. The Error Code is defined as "ERROR_XXXX" (non-zero) in the header file "NISKDIO.h".

Usage:

This function is used for copying data from the NISK-DIO device to user data.

Attention! The function could only be executed after calling the NSKDIO_DeviceInit function.

Reference:

NSKDIO_DeviceInit();NSKDIO_GetDI();NSKDIO_SetDO()

2.4.3 NSKDIO_GetDI

Get DO data from NISKDIO.

C/C++ Syntax:

```
RTN_ERR NSKDIO_GetDI(U32_T DeviceId, U32_T
       Offset, U32_T ByteOfLength, U8_T *GetDiData);
```

Parameters:

U32_T DeviceId: DeviceID is the ID of the NISK-DIO device that determines which NISK-DIO device you want to control. DeviceId starts from 0, and depends on the count of the NISK-DIO devices. For example, if there is only one NISK-DIO device on a machine, DeviceID would be 0. If there are two NISK-DIO devices on a machine, DeviceID might be 0 or 1. The range of "DeviceId" is 0 ~ 15.

U32_T Offset: Offset from start address of NISK-DIO memory. The unit is byte. This value must be 0.

U32_T ByteOfLength: ByteOfLength is the length from the offset of the start address that has to be read from the NISK-DIO device to user data. The unit is byte. This value must be 1.

U8_T *GetDiData: Data is a pointer of U8_T, which points to the start address of the user data.

Returned Values:

Error Code is returned.

"RETURN_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed. The Error Code is defined as "ERROR_XXXX" (non-zero) in the header file "NISKDIO.h".

Usage:

This function is used for copying data from the NISK-DIO device to user data.

Attention! The function could only be executed after calling the NSKDIO_DeviceInit function.

Reference:

NSKDIO_DeviceInit();NSKDIO_GetDO();NSKDIO_SetDO()

2.5 Device Version Functions

- 2.5.1 NSKDIO_GetDriverVersion
- 2.5.2 NSKDIO_GetLibraryVersion

2.5.1 NSKDIO_GetDriverVersion

Gets NISK-DIO Driver Version.

C/C++ Syntax:

```
RTN_ERR NSKDIO_GetDriverVersion(U32_T DeviceId,
                                U32_T *version);
```

Parameters:

U32_T DeviceId: DeviceID is the ID of the NISK-DIO device that determines which NISK-DIO device you want to control. DeviceId starts from 0, and depends on the count of the NISK-DIO devices. For example, if there is only one NISK-DIO device on a machine, DeviceID would be 0. If there are two NISK-DIO devices on a machine, DeviceID might be 0 or 1. The range of "DeviceId" is 0 ~ 15.

U32_T *Version: Version is a pointer of U32_T. After executing this function, the driver version of the NISK-DIO device would be updated in this parameter.

Returned Values:

Error Code is returned.

"RETURN_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed. The Error Code is defined as "ERROR_XXXX" (non-zero) in the header file "NISKDIO.h".

Usage:

This function is used for getting the driver version of the NISK-DIO device.

Attention! The function could only be executed after calling the NSKDIO_DeviceInit function.

Reference:

```
NSKDIO_DeviceInit();
NSKDIO_GetLibraryVersion();
```

2.5.2 NSKDIO_GetLibraryVersion

Gets NISK-DIO Library Version.

C/C++ Syntax:

```
RTN_ERR NSKDIO_GetLibraryVersion(U32_T DeviceId,
                                 U32_T *version);
```

Parameters:

U32_T DeviceId: DeviceID is the ID of the NISK-DIO device that determines which NISK-DIO device you want to control. DeviceId starts from 0, and depends on the count of the NISK-DIO devices. For example, if there is only one NISK-DIO device on a machine, DeviceID would be 0. If there are two NISK-DIO devices on this machine, DeviceID might be 0 or 1. The range of "DeviceId" is 0 ~ 15.

U32_T *Version: Version is a pointer of U32_T. After executing this function, the Library (dll) version of the NISK-DIO device would be updated in this parameter.

Returned Values:

Error Code is returned.

"RETURN_SUCCESS" (0) is returned if function call is successful, while "Error Code" is returned when failed. The Error Code is defined as "ERROR_XXXX" (non-zero) in the header file "NISKDIO.h".

Usage:

This function is used for getting the Library (dll) version of the NISK-DIO device.

Attention! The function could only be executed after calling the `NSKDIO_DeviceInit` function.

Reference:

```
NSKDIO_DeviceInit();
NSKDIO_GetDriverVersion();
```

2.6 Error Codes

Symbol	Code	Description
RETURN_SUCCESS	0	Function call successful
ERROR_NO_INITIAL	0xFFFFFFF1	No initial function called before
ERROR_INVALID_HANDLE_VALUE	0xFFFFFFF2	Invalid handle value
ERROR_INITIAL_MORE_THAN_ ONETIME	0xFFFFFFF3	Initial function called more than one time
ERROR_INITIAL_INTERNAL_ERR	0xFFFFFFF4	Initial internal error
ERROR_NO_THIS_DEVICE	0xFFFFFFF5	Invalid Device id
ERROR_WRITE_LENGTH	0xFFFFFFF6	Write wrong length
ERROR_WRITE_OFFSET	0xFFFFFFF7	Write wrong offset
ERROR_WRITE_DO_DATA	0xFFFFFFF8	Write wrong DO data

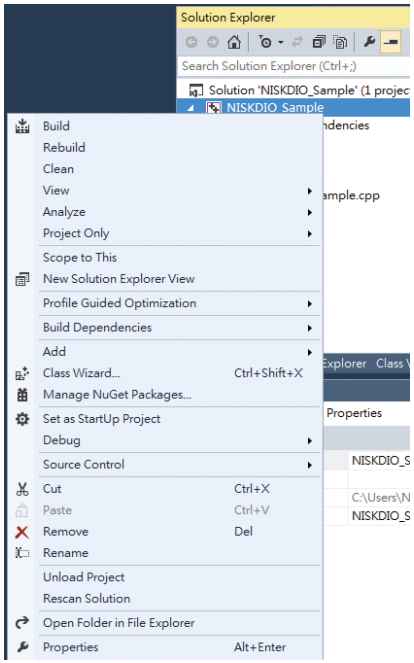
3. Visual Studio Programming Example

- 3.1 How to use the NISK-DIO Library
- 3.2 Programming example

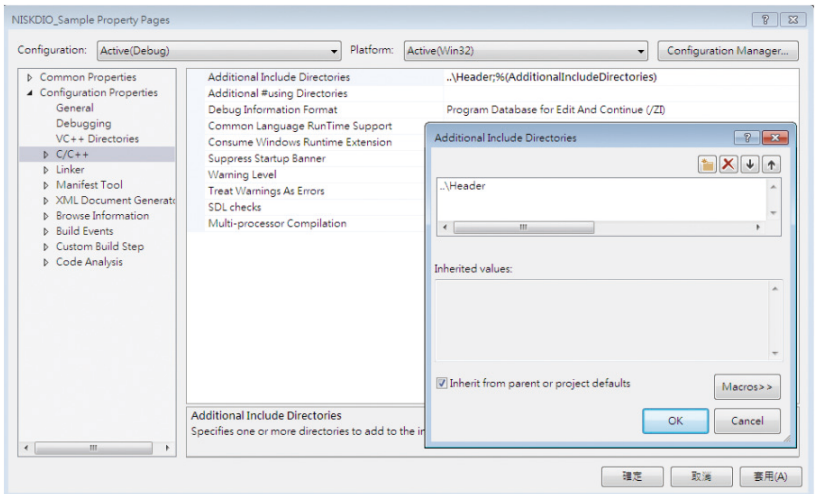
3.1 How to Use the NISK-DIO Library

The following steps show an example of how to use the NISK-DIO library in a Microsoft Visual Studio 2010 project.

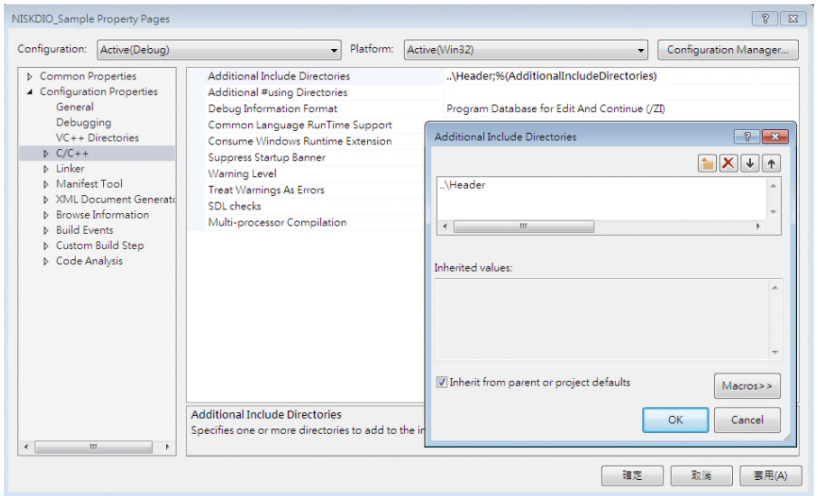
1. Right-click on the NISKDIO_sample project and select **Properties**.



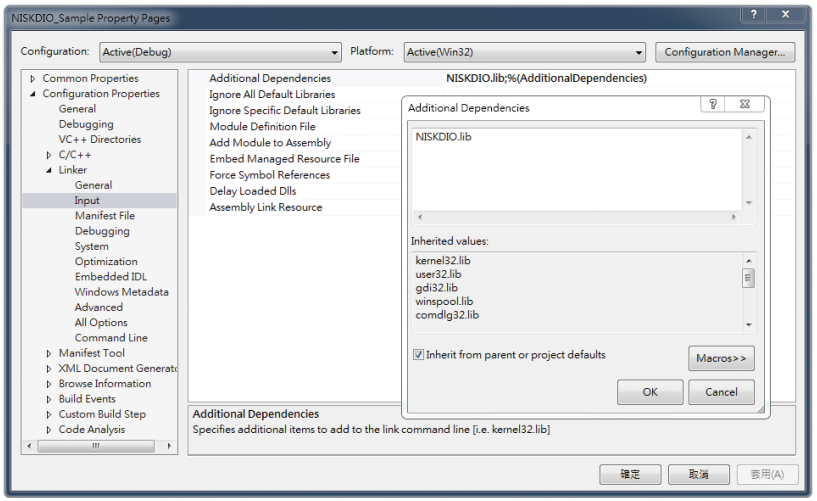
2. Expand the **C/C++** category in **Configuration Properties**, and pick the **General** option. In the **Additional Include Directories** window, add the directory of the header files to be included in this project. In the picture below, we added the relative directory "..\Header" for **Additional Include Directories**.



3. Expand the **Linker** category and pick the **General** option. In the **Additional Library Directories** window, add the directory of the library files to be used in this project. In the picture below, we added the relative directory "..\lib" for **Additional Library Directories**.



4. Expand the **Linker** category and pick the **Input** option. In the **Additional Dependencies** window, add the library file to be used (**NISKDIO.lib** in this example) for **Additional Dependencies**.



3.2 Programming Example

The following example is created by Microsoft Visual Studio 2010. You can find this sample in the installation folder (\Samples\VC2010\)

This is a simple example that shows how to use the NISK-DIO library for the NISK-DIO device using the following instructions:

1. Initialization
   Call initialization function and get the returned status.

2. Get the count of device
   Check that there is no error code returned after initialization.
   (a) If there is no error code returned after initialization:
   - Get the count of device.
   - Get ram size of device.

   (b) If there is an error code returned after initialization:
   - Print the error code returned.
   - Close the program.

3. According to the count of device, get and print the information and version of the device(s).

4. Set DO value, and get DO and DI value from NISK-DIO
   Enter the offset address (offset address = 0) and data length (data length = 1) for DO data writing, writing DO data from 0x00 to 0xFF to all the NISK-DIO devices installed to the computer.

5. Close device
   After finishing the programming of device, call the close function to close the device.

```
#include<stdio.h>
#include<stdlib.h>

#include"NISKDIO.h"

int _tmain(intargc, _TCHAR* argv[])
{
     U32_T ERR_RTN, count = 0, i = 0;
     U32_TDriverVersion[MAX_NSKDIO_DEVICE_NUMBER];
     U32_TLibraryVersion[MAX_NSKDIO_DEVICE_NUMBER];
     DEVICE_INFO_T dev_info[MAX_NSKDIO_DEVICE_NUMBER];
     U8_TDO_SetValue = 0xFF, DO_GetValue = 0, DI_GetValue = 0;

       /*Instruction #1*/
       // NISKDIO initial
       ERR_RTN = NSKDIO_DeviceInit();

       /*Instruction #2*/
       //get device(s) count
       if (ERR_RTN != RETURN_SUCCESS)//Initial fail
       {
       //print error code of initialization
       printf("NSKDIO_DeviceInit Error code: %x\n", ERR_RTN);
       }
       else
       {
       //confirm there is no error after initialization
       //get device of NISKDIO count
       ERR_RTN = NSKDIO_GetDeviceCount(&count);
       printf("This machine has installed %d device(s) of
              NISKDIO\n", count);
       }
```

```
for (i = 0; i< count; i++)
        {
        /*Instruction #3*/
        //get device(s) information
        //depending on the device count we got
        //get the information of device(s)
        ERR_RTN = NSKDIO_GetDeviceInfo(i, &dev_info[i]);

        printf("Total NISKDIO device(s) = %d \n", count);
        printf("Device #%d\n", i);
        printf("Device Id : %d\n", dev_info[i].DeviceID);
        printf("DeviceBusNum : %d\n", dev_info[i].DeviceBusNum);
        printf("DeviceDevNum : %d\n", dev_info[i].DeviceDevNum);
        printf("DeviceFuncNum : %d\n", dev_info[i].DeviceFuncNum);

        //depending on the device count we got
        //get the driver version of device(s)
        //get the library version of devices(s)

     ERR_RTN = NSKDIO_GetDriverVersion(i, &DriverVersion[i]);
     ERR_RTN = NSKDIO_GetLibraryVersion(i, &LibraryVersion[i]);
     printf("Driver version: %x\n", DriverVersion[i]);
     printf("Library version: %d\n", LibraryVersion[i]);
```

```
     /*Instruction #4*/
     //Set DO = 0xFF
     ERR_RTN = NSKDIO_SetDO(i, 0, 1, &DO_SetValue);
     //Get DO value
     ERR_RTN = NSKDIO_GetDO(i, 0, 1, &DO_GetValue);
     //Get DI value
     ERR_RTN = NSKDIO_GetDI(i, 0, 1, &DI_GetValue);
}
       /*Instruction #5*/
       // NISKDIO close
       ERR_RTN = NSKDIO_DeviceClose();
       return 0;
}
```