# NEXCOM

MiniPCIe NISK-NVRAM

Library User Manual

Manual Rev.: V0.4

Revision Date: Feb. 03rd, 2015

## Revise note:

| Ver | Description |
|---|---|
| V0.1 | 2015/01/27: |
| V0.2 | 2015/01/30: |
| V0.3 | 2015/02/02: |
| V0.4 | 2015/02/03: |

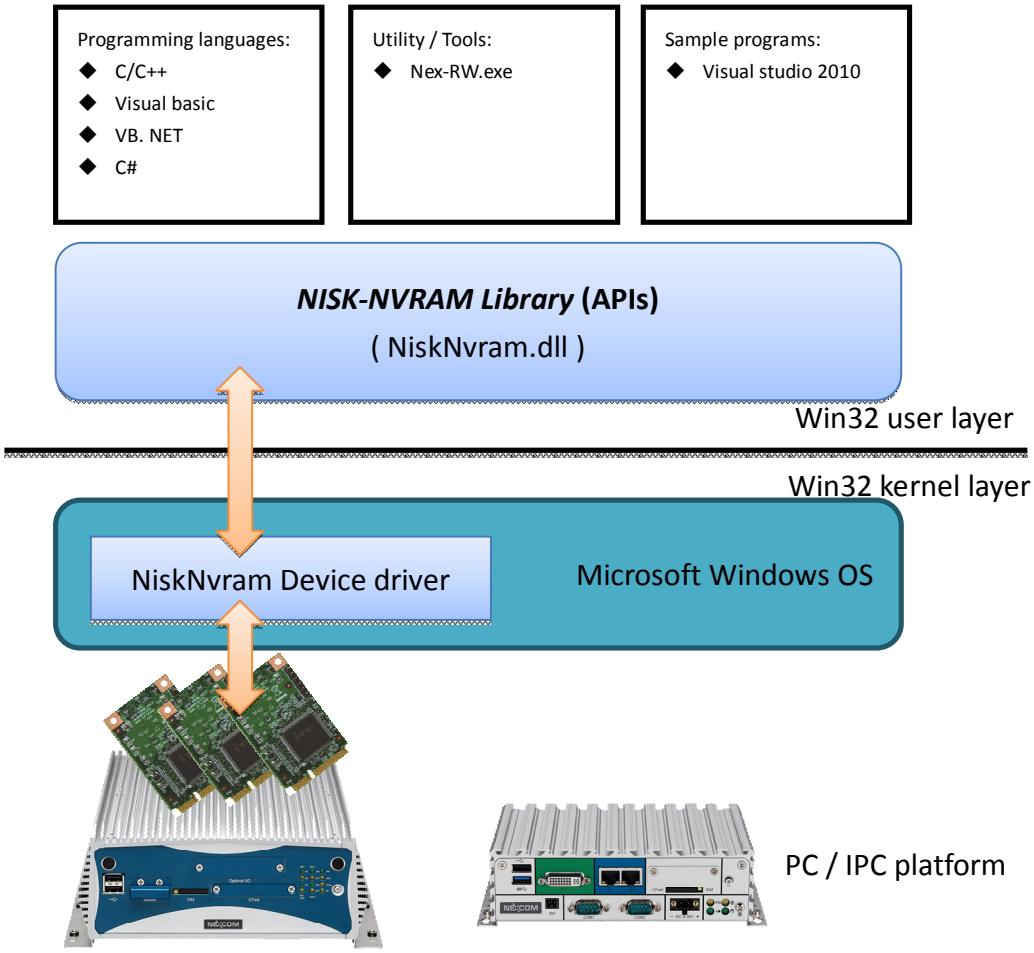
# Contents

# 1. NISK-NVRAM Library Overview

## 1.1. Introduction

***NISK-NVRAM Library*** is a programming interface for controlling Mini PCIe NISK-NVRAM devices.

Following figure shows the system architecture of NISK-NVRAM:




NISK-NVRAM system architecture

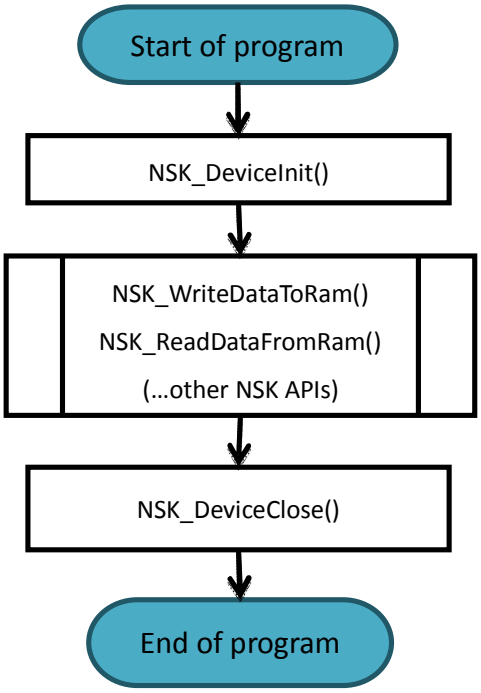
Sample programs for NISK-NVRAM please refer to Chapter.3 as a programming reference.

## 1.2. Operation system

NISK-NVRAM library (NiskNVRAM.dll and device driver) supports following operating system:
Microsoft® Windows® 7 (32 bit)

## 1.3. Principles of Programming

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




1. Before using the Nisk NVRAM functions, please execute the initial function NSK_DeviceInit() first.
2. After finishing programming, please execute the close function NSK_DeviceClose() to close the devices.
3. This library (device driver) supports up to 16 NISK-NVRAM devices in one machine.
4. The maximum size of NVRAM memory on the device is 1 Mbytes(1048576 bytes).

# 2. API Reference

## 2.1. API Overview

All APIs of NiskNVRAM Library are listed. The definition of API is located at the header file “NiskNVRAM.h”.

| Function Name | Description |
|---|---|
| Initialization Functions | |
| NSK_DeviceInit | Nisk NVRAM initial function |
| NSK_DeviceClose | Nisk NVRAM close function |
| Device Information Functions | |
| NSK_GetDeviceRamSize | Get the total size of Nisk NVRAM |
| NSK_GetDeviceCount | Get how many devices of Nisk NVRAM |
| NSK_GetDeviceInfo | Get Nisk NVRAM Bus/Device/Function number |
| Read/Write Functions | |
| NSK_WriteDataToRam | Copy data from user data to Nisk NVRAM |
| NSK_ReadDataFromRam | Copy data from Nisk NVRAM to user data |
| Save/Load Functions | |
| NSK_SaveDeviceDataToFile | Save the data of device to file |
| NSK_LoadFileToDeviceData | Load the data from file to device |
| Device Version Functions | |
| NSK_GetFirmwareVersion | Get Nisk NVRAM Firmware Version |
| NSK_GetHardwareVersion | Get Nisk NVRAM Hardware Version |
| NSK_GetDriverVersion | Get Nisk NVRAM Driver Version |
| NSK_GetLibraryVersion | Get Nisk NVRAM dll Version |

The C/C++ data types for API is 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 number | 4 | IEEE-754, accurate to the seventh decimal place |
| F64_T | double | Double-precision floating-point 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. NSK_DeviceInit

Nisk NVRAM initial function

**C/C++ Syntax:**

RTN_ERR NSK_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 header file “NiskNVRAM.h”.

**Usage:**

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

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

**Reference:**

NSK_DeviceClose();

### 2.2.2. NSK_DeviceClose

Nisk NVRAM close function

**C/C++ Syntax:**

RTN_ERR NSK_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 header file “NiskNVRAM.h”.

**Usage:**

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

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

**Reference:**

NSK_DeviceInit();

## 2.3. Device Information Functions

### 2.3.1. NSK_GetDeviceRamSize

Get the total size of Nisk NVRAM

**C/C++ Syntax:**

RTN_ERR NSK_GetDeviceRamSize(U32_T *size);

**Parameters:**

U32_T *Size: Size is a pointer of U32_T. After executing this function, the total size of the device of Nisk NVRAM 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 header file "NiskNVRAM.h".

**Usage:**

This function is used for getting the total size of the device of Nisk NVRAM.

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

**Reference:**

NSK_DeviceInit();

### 2.3.2. NSK_GetDeviceCount

Get how many devices of Nisk NVRAM on a machine

**C/C++ Syntax:**

RTN_ERR NSK_GetDeviceCount(U32_T *count);

**Parameters:**

U32_T *count: count is a pointer of U32_T. After executing this function, the total count of the device of Nisk NVRAM 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 header file “NiskNVRAM.h”.

**Usage:**

This function is used for getting the total count of the device of Nisk NVRAM on a machine.

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

**Reference:**

NSK_DeviceInit();

### 2.3.3. NSK_GetGetDeviceInfo

Get Bus/Device/Function number of Nisk NVRAM

**C/C++ Syntax:**

RTN_ERR NSK_GetDeviceInfo(U32_T Device_Id, DEVICE_INFO_T *pInfo);

**Parameters:**

U32_T DeviceId : DeviceID is the ID of the device of Nisk NVRAM that determines which device of Nisk NVRAM you want to control. DeviceId starts from 0, and depends on the count of the devices of Nisk NVRAM. For example, if there is only one device of Nisk NVRAM on a machine, DeviceID would be 0. If there are two device of Nisk NVRAM on this 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 device of Nisk NVRAM 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 header file "NiskNVRAM.h".

**Usage:**

This function is used for getting the driver version of the device of Nisk NVRAM.

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

**Reference:**

NSK_DeviceInit();

## 2.4. Read/Write Functions

### 2.4.1. NSK_WriteDataToRam

Copy data from user data to Nisk NVRAM

**C/C++ Syntax:**

RTN_ERR NSK_WriteDataToRam(U32_T DeviceId, U32_T Offset, U32_T ByteOfLength, U8_T *Data);

**Parameters:**

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

U32_T Offset: Offset from start address of NVRAM memory. The unit is byte.
The range of "Offset" is 0 ~ (MAX_NVRAM_SIZE - 1),
MAX_NVRAM_SIZE is 1048576.

U32_T ByteOfLength: ByteOfLength is the length from the offset of the start address that has to be written from user data to the device of Nisk NVRAM. The unit is byte.
The range of "ByteOfLength" is 1 ~ MAX_NVRAM_SIZE,
MAX_NVRAM_SIZE is 1048576.

**Attention!** ByteOfLength has to be equal or smaller than the residual size, the residual size is (MAX_NVRAM_SIZE - Offset).

U8_T *Data: Data is a pointer of U8_T, which pointed 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 header file "NiskNVRAM.h".

**Usage:**

This function is used for copying data from user data to the device of Nisk NVRAM.

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

**Reference:**

NSK_DeviceInit();NSK_ReadDataFromRam()

### 2.4.2. NSK_ReadDataFromRam

Copy data from Nisk NVRAM to user data

**C/C++ Syntax:**

RTN_ERR NSK_ReadDataFromRam(U32_T DeviceId, U32_T Offset, U32_T ByteOfLength, U8_T *Data);

**Parameters:**

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

U32_T Offset: Offset from start address of NVRAM memory. The unit is byte.
The range of "ByteOfLength" is 1 ~ MAX_NVRAM_SIZE, MAX_NVRAM_SIZE is 1048576.

U32_T ByteOfLength: ByteOfLength is the length from the offset of the start address that has to be read from the device of Nisk NVRAM to user data. The unit is byte.
The range of "ByteOfLength" is 1 ~ MAX_NVRAM_SIZE, MAX_NVRAM_SIZE is 1048576.

**Attention!** ByteOfLength has to be equal or smaller than the residual size, the residual size is (MAX_NVRAM_SIZE - Offset).

U8_T *Data: Data is a pointer of U8_T, which pointed 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 header file "NiskNVRAM.h".

**Usage:**

This function is used for copying data from the device of Nisk NVRAM to user data.

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

## 2.5. Save/Load Functions

### 2.5.1. NSK_SaveDeviceDataToFile

Save the data of device to file

**C/C++ Syntax:**

RTN_ERR FNTYPE NSK_SaveDeviceDataToFile(U32_T DeviceId, I8_T* save_file);

**Parameters:**

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

I8_T* save_file: save_file is a pointer of I8_T, pointed to a string which is the directory of the file has to be saved. Note that the filename extension has to be ".nrw". Ex. "C:\\test_data.nrw"

**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 header file "NiskNVRAM.h".

**Reference:**

NSK_LoadFileToDeviceData();

**Usage:**

This function is used for saving data (data byte 0 ~1048575) from the device of Nisk NVRAM to a file(.nrw).
**Attention!** The function could only be executed after calling the NSK_DeviceInit function.

### 2.5.2. NSK_LoadFileToDeviceData

Load the data from file to device

**C/C++ Syntax:**

RTN_ERR FNTYPE NSK_LoadFileToDeviceData(U32_T DeviceId, I8_T* load_file);

**Parameters:**

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

I8_T* load_file: load_file is a pointer of I8_T, pointed to a string which is the directory of the file has to be loaded. Note that the filename extension has to be ".nrw". Ex. "C:\\test_data.nrw"

**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 header file "NiskNVRAM.h".

**Reference:**

NSK_SaveDeviceDataToFile();

**Usage:**

This function is used for loading data (data byte 0 ~1048575) from a file(.nrw) to the device of Nisk NVRAM.
**Attention!** The function could only be executed after calling the NSK_DeviceInit function.

## 2.6. Device Version Functions

### 2.6.1. NSK_GetFirmwareVersion

Get Nisk NVRAM Firmware Version

**C/C++ Syntax:**

RTN_ERR NSK_GetFirmwareVersion(U32_T DeviceId, U32_T *version);

**Parameters:**

U32_T DeviceId : DeviceID is the ID of the device of Nisk NVRAM that determines which device of Nisk NVRAM you want to control. DeviceId starts from 0, and depends on the count of the devices of Nisk NVRAM. For example, if there is only one device of Nisk NVRAM on a machine, DeviceID would be 0. If there are two device of Nisk NVRAM 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 firmware version of the device of Nisk NVRAM 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 header file "NiskNVRAM.h".

**Usage:**

This function is used for getting the firmware version of the device of Nisk NVRAM.
**Attention!** The function could only be executed after calling the NSK_DeviceInit function.

**Reference:**

NSK_DeviceInit();
NSK_GetHardwareVersion();
NSK_GetDriverVersion();
NSK_GetLibraryVersion();

### 2.6.2. NSK_GetHardwareVersion

Get Nisk NVRAM Hardware Version

**C/C++ Syntax:**

RTN_ERR NSK_GetHardwareVersion(U32_T DeviceId, U32_T *version);

**Parameters:**

U32_T DeviceId : DeviceID is the ID of the device of Nisk NVRAM that determines which device of Nisk NVRAM you want to control. DeviceId starts from 0, and depends on the count of the devices of Nisk NVRAM. For example, if there is only one device of Nisk NVRAM on a machine, DeviceID would be 0. If there are two device of Nisk NVRAM 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 hardware version of the device of Nisk NVRAM 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 header file "NiskNVRAM.h".

**Usage:**

This function is used for getting the hardware version of the device of Nisk NVRAM.
**Attention!** The function could only be executed after calling the NSK_DeviceInit function.

**Reference:**

NSK_DeviceInit();
NSK_GetFirmwareVersion();
NSK_GetDriverVersion();
NSK_GetLibraryVersion();

### 2.6.3. NSK_GetDriverVersion

Get Nisk NVRAM Driver Version

**C/C++ Syntax:**

RTN_ERR NSK_GetDriverVersion(U32_T DeviceId, U32_T *version);

**Parameters:**

U32_T DeviceId : DeviceID is the ID of the device of Nisk NVRAM that determines which device of Nisk NVRAM you want to control. DeviceId starts from 0, and depends on the count of the devices of Nisk NVRAM. For example, if there is only one device of Nisk NVRAM on a machine, DeviceID would be 0. If there are two device of Nisk NVRAM 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 driver version of the device of Nisk NVRAM 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 header file "NiskNVRAM.h".

**Usage:**

This function is used for getting the driver version of the device of Nisk NVRAM.
**Attention!** The function could only be executed after calling the NSK_DeviceInit function.

**Reference:**

NSK_DeviceInit();
NSK_GetFirmwareVersion();
NSK_GetHardwareVersion();
NSK_GetLibraryVersion();

### 2.6.4. NSK_GetDriverVersion

Get Nisk NVRAM Library Version

**C/C++ Syntax:**

RTN_ERR NSK_GetLibraryVersion(U32_T DeviceId, U32_T *version);

**Parameters:**

U32_T DeviceId : DeviceID is the ID of the device of Nisk NVRAM that determines which device of Nisk NVRAM you want to control. DeviceId starts from 0, and depends on the count of the devices of Nisk NVRAM. For example, if there is only one device of Nisk NVRAM on a machine, DeviceID would be 0. If there are two device of Nisk NVRAM 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 device of Nisk NVRAM 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 header file "NiskNVRAM.h".

**Usage:**

This function is used for getting the Library(dll) version of the device of Nisk NVRAM.
**Attention!** The function could only be executed after calling the NSK_DeviceInit function.

**Reference:**

NSK_DeviceInit();
NSK_GetFirmwareVersion();
NSK_GetHardwareVersion();
NSK_GetDriverVersion();

## 2.7. Error Codes

| Symbol | Code | Description |
|---|---|---|
| RETURN_SUCCESS | 0 | Function call successfully |
| ERROR_READ_OFFSET_SIZE | 0xFFFFFFF0 | Read size illegal |
| ERROR_WRITE_OFFSET_SIZE | 0xFFFFFFF1 | Write size illegal |
| ERROR_NO_INITIAL | 0xFFFFFFF2 | No initial function called before |
| ERROR_INVALID_HANDLE_VALUE | 0xFFFFFFF3 | Invalid handle value |
| ERROR_INITIAL_MORE_THAN_ONETIME | 0xFFFFFFF4 | Initial function called more than one time |
| ERROR_INITIAL_INTERNAL_ERR | 0xFFFFFFF5 | Initial internal error |
| ERROR_NO_THIS_DEVICE | 0xFFFFFFF6 | Invalid Device id |
| ERROR_FILE_OPEN | 0xFFFFFFF7 | Error of file opening of save or load file |
| ERROR_INPUT_FILENAME_EXTENSION | 0xFFFFFFF8 | Invalid filename extension of save or load file |

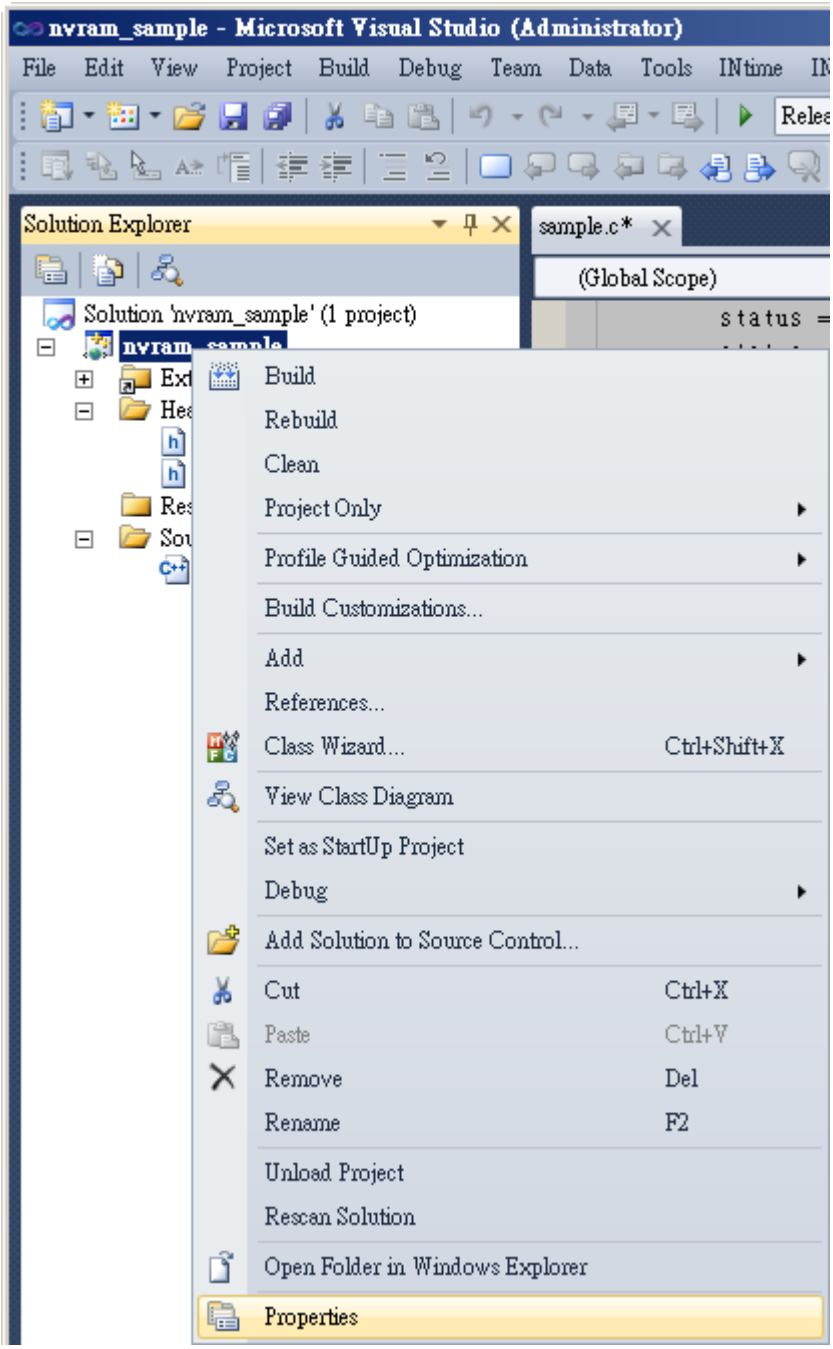
# 3. Programming example

## 3.1. Visual Studio programming example

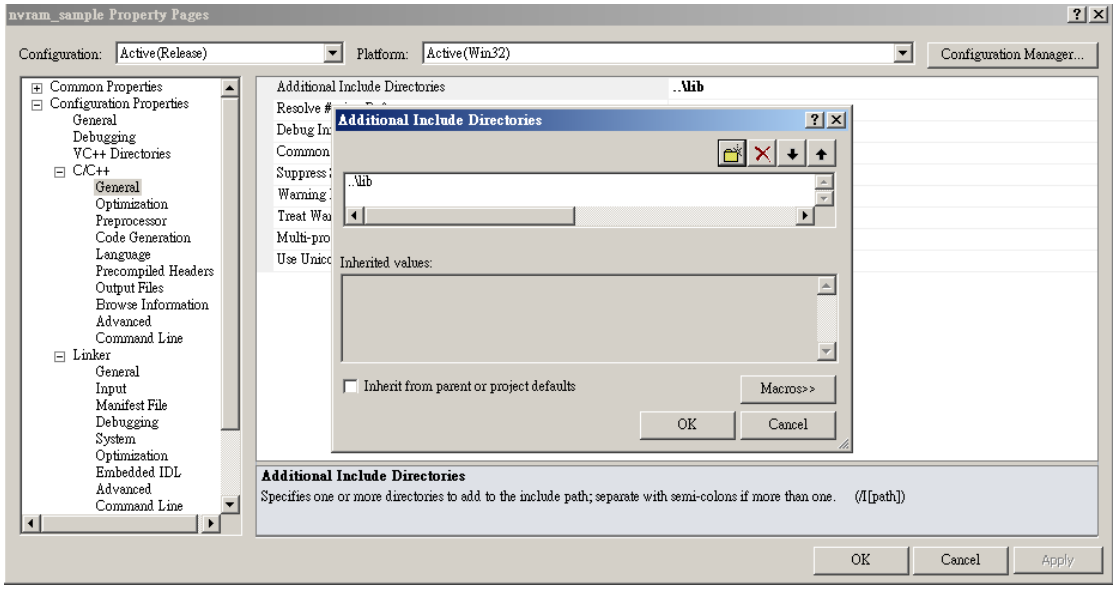
### 3.1.1. How to use the NiskNVRAM Library

The following steps show that how to use the NiskNVRAM library in the project of Mircosoft Visual Studio 2010 for this example:

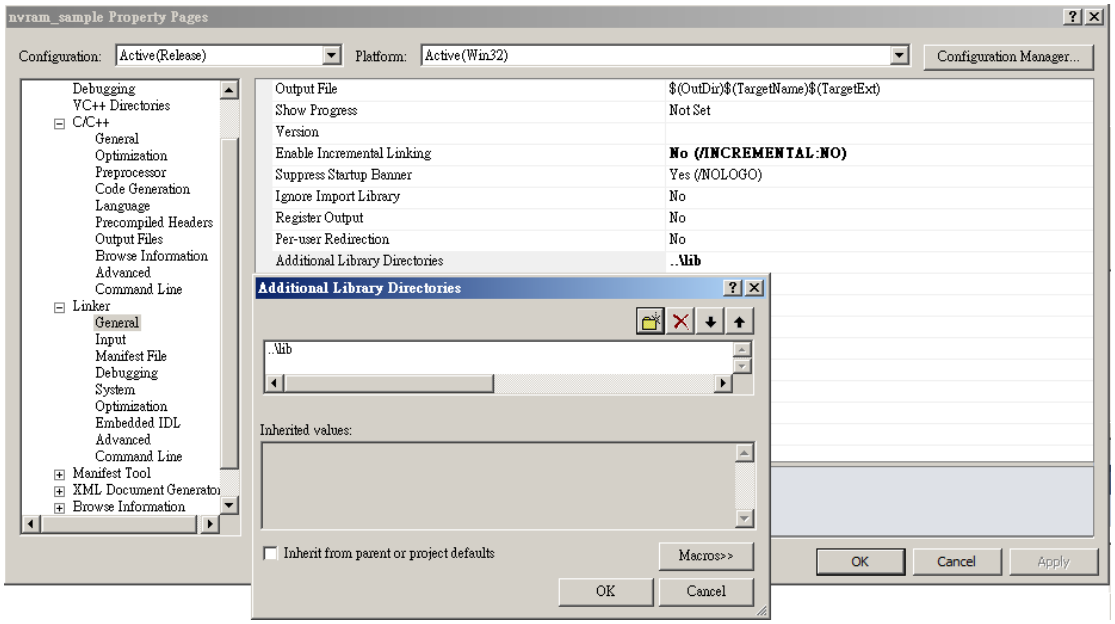
1.Click the properties of the project of nvram_sample.



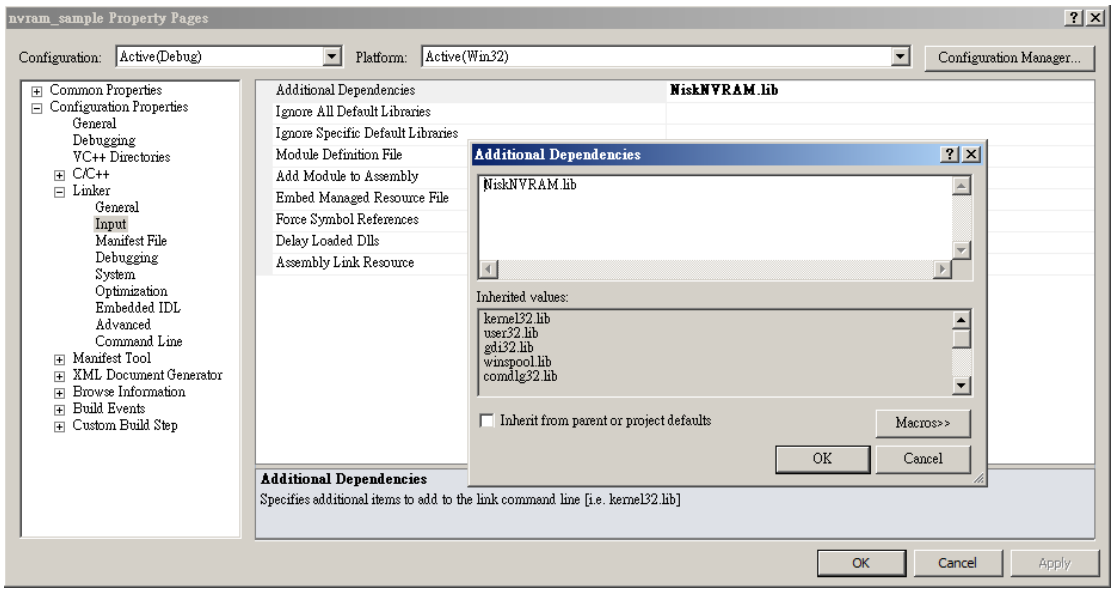
2. Select the **C/C++** category of properties, and pick the **general** option under that. In the **Additional Include Directories** property, add the directory of the header files which we want to include in this project. In picture below, we add the relative directory "..\lib" for **Additional Include Directories** in this example.




3. Select the **Linker** category of properties ,and pick the **General** option under that. In the **Additional Library Directories** property, add the directory of the library files which we want to use in this project. In picture below, we add the relative directory "..\lib" for **Additional Library Directories** in this example.



4. Select the **Linker** category of properties and, pick the **Input** option under that. In the **Additional Dependencies** property, add the library file, such as **NiskNVRAM.lib** for **Additional Dependencies** property in this example.



### 3.1.2. Programming example

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

This is a simple example to show that how to use the NiskNVRAM library for the device of Nisk NVRAM by 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.

- Getthe 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 device(s).

4.Write data test

Enter the offset address for data writing, writing data from 0 to 15 to the device of DeviceId #0, the length of writing data is 16.

5.Read data test

Enter the offset address for data reading, reading data from the device of DeviceId #0, the length of reading data is 16. Finally, print the read data.

6.Save data test

Test for saving data from device to file.

7.Save data test

Test for loading data from file to device.

8.Close device

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

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

#include "NiskNVRAM.h"

void main (void)
{
	U32_Tstatus = 0;
	U32_Tcount = 0;
	U32_Ti = 0;
	U32_Tsize = 0;

	U32_Tdriver_version[MAX_NVRAM_DEVICE_NUMBER];
	U32_Tfirmware_version[MAX_NVRAM_DEVICE_NUMBER];
	U32_Thardware_version[MAX_NVRAM_DEVICE_NUMBER];
	U32_Tlibrary_version[MAX_NVRAM_DEVICE_NUMBER];

	DEVICE_INFO_T dev_info[MAX_NVRAM_DEVICE_NUMBER];

	U8_T write_data[16] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15};
	U32_Twrite_length = 0;
	U32_Twrite_offset = 0;
	U32_Twrite_status = 0;

	U8_T read_data[16] = {0};
	U32_Tread_length = 0;
	U32_Tread_offset = 0;
	U32_Tread_status = 0;

	I8_T* save_file = "C:\\test_data.nrw";
	I8_T* load_file = "C:\\test_data.nrw";


	/*Instruction #1*/
```

```
// Nisk NVRAM initial
status = NSK_DeviceInit();

/*Instruction #2*/
//get device(s) count
if(status == RETURN_SUCCESS)
{
	//confirm there is no error after initialization
	//get device of Nisk NVRAM count
	status = NSK_GetDeviceCount(&count);
	printf("This machine has installed %d device(s) of Nisk NVRAM\n",count);
	status = NSK_GetDeviceRamSize(&size);
	printf("size of Nisk NVRAM is %d bytes\n", size);
}
else//Initial fail
{
	//print error code of initialization
	printf("Error code : %x\n", status);
	system("pause");
}

/*Instruction #3*/
//get device(s) information
for( i = 0; i < count; i++)
{
	//depending on the device count we got
	//get the information of device(s)
	status = NSK_GetDeviceInfo(i, &dev_info[i]);
	printf("Device information:\n");
	printf("  Device Id: %d\n", dev_info[i].DeviceID);
	printf("  Bus Number: %d\n", dev_info[i].DeviceBusNum);
	printf("  Device Number: %d\n", dev_info[i].DeviceDevNum);
	printf("  Function Number: %d\n", dev_info[i].DeviceFuncNum);


	//depending on the device count we got
	//get the driver version of device(s)
	//get the firmware version of device(s)
```

```
        //get the hardware version of device(s)
        //get the library version of devices(s)
        status = NSK_GetDriverVersion(i, &driver_version[i]);
        status = NSK_GetFirmwareVersion(i, &firmware_version[i]);
        status = NSK_GetHardwareVersion(i, &hardware_version[i]);
        status = NSK_GetLibraryVersion(i, &library_version[i]);
        printf("  Driver version: %x\n", driver_version[i]);
        printf("  Firmware version: %x\n", firmware_version[i]);
        printf("  Hardware version: %x\n", hardware_version[i]);
        printf("  Library vesrion: %d\n", library_version[i]);

    }


    /*Instruction #4*/
    //write data from 0 to 15 to deviceid 0 for testing
    printf("\n<Write and read testing for device 0>\n");
    printf("enter the offset for data write:\n");
    scanf_s("%d", &write_offset);
    write_status = NSK_WriteDataToRam(0, write_offset, 16, &write_data[0]);
    if (write_status == RETURN_SUCCESS)
    {
        printf("Write data success!!\n");
    }
    else
    {
        printf("NSK_WriteDataToRam Error code: %x\n", write_status);
    }


    /*Instruction #5*/
    //read data from deviceid 0 from write_offset for testing
    printf("enter the offset for data read:\n");
    scanf_s("%d", &read_offset);
    read_status = NSK_ReadDataFromRam(0, read_offset, 16, &read_data[0]);
    if (read_status == RETURN_SUCCESS)
    {
        printf("Read data success!!\n");
```

```
    }
    else
    {
        printf("NSK_WriteDataToRam Error code: %x\n", write_status);
    }
    if (!read_status)
    {
        //print the read data
        for (i = 0; i < 16; i++)
        {
            printf("read #%d byte : %d\n", i, read_data[i]);
        }
    }

    /*Instruction #6*/
    //test for saving data from device to file("C:\\test_data.nrw")
    status = NSK_SaveDeviceDataToFile(0, save_file);
    printf("Save data from device to file finish!!\n");

    /*Instruction #7*/
    //test for loading data from file("C:\\test_data.nrw") to device
    status = NSK_LoadFileToDeviceData(0, load_file);
    printf("Load data from file to device finish!!\n");

    /*Instruction #8*/
    // Nisk NVRAM close
    NSK_DeviceClose();
    system("pause");
}
```