NuWa SDK
User Guide
for eMbedded Visual C++

(Version 1.0)

NuWaSDK for NuWa
Using I-8000 Series Modules

Warranty

All products manufactured by ICPDAS Inc. are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICPDAS Inc. assumes no liability for damages consequent to the use of this product. ICPDAS Inc. reserves the right to change this manual at any time without notice. The information furnished by ICPDAS Inc. is believed to be accurate and reliable. However, no responsibility is assumed by ICPDAS Inc. for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only maybe registered trademarks of their respective companies.

License

The user can use, modify and backup this software on a single machine. The user may not reproduce, transfer or distribute this software, or any copy, in whole or in part.

Contents

1. Introduction

Welcome to the **NuWaSDK_LIB** user's manual. ICPDAS provides Library files, namely the **NuWaSDK_LIB**, for the I-8000 series modules which are used in the NuWa Embedded Controller. The **NuWaSDK_LIB** has all the essential Library functions designed for the I-8000 series modules for Microsoft WinCE.Net platform. It can be applied on eMbedded Visual C++ on WinCE 4.2.Net, and even on the newer platforms. Users can easily develop WinCE.NET applications on NuWa by using this toolkit. The various functions in **NuWaSDK_LIB** are divided into the following sub-group functions for easy use in different applications.

2. NuWaSDK.LIB

In this section we will focus on the description and application example of NuWa Library functions. The functions of NuWaSDK.LIB can be clarified into 10 groups which are listed below:

1. System Information Functions;
2. Software Information Functions;
3. Digital Input/Output Functions;
4. Watch Dog Timer Functions;
5. EEPROM Read/Write Functions;
6. Analog Input Functions;
7. Analog Output Functions;
8. 3-axis Encoder Functions;
9. 2-axis Stepper/Servo Functions;
10. Counter/Frequency Functions.

All the functions supplied for use with NuWa which have been listed, come with more detailed information for each function and this is given in the following section. In the Syntax format, the function indicates the eMbedded Visual C^{++} syntax, in order to make the description more simplified and clear, the attributes for the input and output parameters of a function are depicted as [input] and [output] respectively, as is shown in the following table.

Keyword	Users set parameters before calling this function?	Get the data from this parameter after calling this function?
[input]	Yes	No
[output]	No	Yes

When using the eMbedded Visual C++ development tool to design an application, you must include the NuWaSDK.h file in the source program, and link it to NuWaSDK.lib when building user applications.

Applied on	WinCE Versions	Defined in	Include	Link to
NuWa	4.2.0.01 and later	NuWaSDK.h	NuWaSDK.h	NuWaSDK.lib

2.1 System Information Functions

■ ChangeSlotTo87K

Description:

This function is used to dedicate serial control to the specified slot for the control of 87K series. The serial bus in the NuWa backplane is for mapping through to COM1. For example, if you want to send or receive data from a specified slot, you need to call this function first. Then you can use the other series functions.

Syntax:

[C++]

```
void ChangeSlotTo87K(unsigned char slotNo)
```

Parameter:

slotNo : [Input] Specify the slot number.

Return Value:

None

Example:

```
unsigned char slot=1;
ChangeSlotTo87K(slot);
//The first slot is specified as COM2 port in NuWa.
```

Remark:

■ GetModuleType

Description:

This function is used to retrieve the type of 8000 series I/O module plugged into a specific I/O slot in the NuWa system. This function performs a supporting task in the collection of information relating to system hardware configurations.

Syntax:

[C++]

```
int GetModuleType(int slot)
```

Parameter:

slot : [Input] Specify the slot number in which the I/O module is plugged into.

Return Value:

Module Type: it is defined in the following table.

Type	Value
_PARALLEL	0x80
_SCAN	0x40
_32BIT	0x20
_DI32	0xE3
_DO32	0xE0
_DI16DO16	0xE2
_DI16	0xC3
_DO16	0xC0
_DI8DO8	0xC2

Example:

```
int slot=1,moduleType;
moduleType=GetModuleType(slot);
//The i-8057 card is plugged in slot 1 of NuWa and has a return Value of:0xC0
```

Remark:

■ GetNameOfModule

Description:

This function is used to retrieve the name of an 8000 series I/O module, which is plugged into a specific I/O slot in the NuWa system. This function supports the collection of system hardware configurations.

Syntax:

[C++]

```
int GetNameOfModule(int slot, char *string1)
```

Parameter:

slot: [Input] Specify the slot number where the I/O module is plugged into.

string1: [Output] the pointer to a buffer to receive the name of the I/O module.

Return Value:

I/O module ID. For Example, the I-8024 will return 24.

Example:

```
int slot=1,moduleID;
char moduleName[5];
moduleID=GetNameOfModule(slot, moduleName)
//The I-8057 card plugged in slot 1 of NuWa
//Returned Value: moduleID=57     moduleName=”  8057”
```

Remark:

■ GetTimeTicks

Description:

This function is used to retrieve the number of milliseconds that have elapsed since Windows CE started. The elapsed time is stored as a DWORD value. Therefore, the time will wrap around to zero if the system is run continuously for 49.7 days.

Syntax:

[C++]

```
DWORD GetTimeTicks(void)
```

Parameter:

None

Return Value:

The number of milliseconds that have elapsed since the system was started and it indicates success.

Example:

```
DWORD time;
time=GetTimeTicks();
//Returned Value: time=94367
```

Remark:

■ GetSlotAddr

Description:

This function retrieves the base memory address for a specific slot.

Syntax:

[C++]

```
DWORD GetSlotAddr(int slot)
```

Parameter:

slot : [Input] Specify the slot number.

Return Value:

Slot base address. The NuWa system currently provides 7 slots (from 1 to 7), and their corresponding base addresses are:

Slot 1 -> 0x8bb00100

Slot 2 -> 0x8bb00200

Slot 3 -> 0x8bb00300

Slot 4 -> 0x8bb01800

Slot 5 -> 0x8bb01900

Slot 6 -> 0x8bb01a00

Slot 7 -> 0x8bb01b00

Example:

```
DWORD slotAddr;
slotAddr=GetSlotAddr( 1 );
//Returned Value: slotAddr= 0x8bb00100
```

Remark:

■ GetSystemSerialNumber

Description:

This function retrieves the hardware serial identification number on the NuWa main controller. This function supports the control of hardware versions by reading the 64-bit serial ID chip.

Syntax:

[C++]

```
int GetSystemSerialNumber(char *string1)
```

Parameter:

string1: [Output] the pointer to a buffer to receive the serial ID number.

Return Value:

0: indicates success.

1: indicates failure.

Example:

```
char serialNo[8];
GetSystemSerialNumber(serialNo);
//Returned value: serialNo=0x9 EF EF EB BA EA BE AF
```

Remark:

2.2 Software Information Functions

■ GetSDKversion

Description:

This function retrieves the version number of the linked NuWa SDK library files.

Syntax:

[C++]

```
void GetSDKversion(LPTSTR lpSDKversion)
```

Parameter:

lpSDKversion : [Output] the pointer to a string to receive the version number of NuWaSDK.DLL.

Return Value:

None

Example:

```
TCHAR sdkVersion[20];
GetSDKversion(sdkVersion);
```

Remark:

■ GetOSversion

Description:

This function retrieves the version number of the embedded OS.

Syntax:

[C++]

```
void GetOSversion(LPTSTR lpOSversion)
```

Parameter:

lpOSversion : [Output] the pointer to a buffer to receive the OS version.

Return Value:

None

Example:

```
TCHAR osVersion[20];
GetOSversion(osVersion);
// Returned value: osVersion="Windows CE .NET 4.2 01-00-00"
```

Remark:

2.3 Digital Input/Output Functions

■ DO_8

Description:

This function is used to output 8-bit data to a digital output module. The 0~7 bits of output data is mapped into the 0~7 channels of digital module output respectively.

Syntax:

[C++]

```
void DO_8(int slot, unsigned char cdata)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

cdata : [Input] output data.

Return Value:

None

Example:

```
int slot=1;
char data=3;
DO_8(slot, data);
//The I-8064 card is plugged in slot 1 of NuWa and can turn on channel 0 and 1.
```

Remark:

This function can be applied on modules: i8060, i8064, i8065, i8066, and i8068.

■ DO_16

Description:

This function is used to output 16-bit data to a digital output module. The 0~15 bits of output data is mapped into the 0~15 channels of digital output modules respectively.

Syntax:

[C++]

```
void DO_16(int slot, unsigned int cdata)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

cdata : [Input] output data.

Return Value:

None

Example:

```
int slot=1;
unsigned int data=3;
DO_16(slot, data);
//The I-8057 card is plugged in slot 1 of NuWa and can turn on channel 0 and 1.
```

Remark:

This function can be applied on modules: i8057, i8056.

■ DO_32

Description:

Output the 32-bit data to a digital output module. The 0~31 bits of output data are mapped into the 0~31 channels of digital output modules respectively.

Syntax:

[C++]

```
void DO_32(int slot, unsigned long cdata)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

cdata : [Input] output data.

Return Value:

None

Example:

```
int slot=1;
unsigned long data=3;
DO_32(slot, data);
//The I-8041 card is plugged in slot 1 of NuWa and can turn on channel 0 and 1.
```

Remark:

This function can be applied on module: i8041.

■ DO_8_BW

Description:

Set the digital output value of the channel No. in the 8-channel Digital Output series modules. The output Value is “true” or “false”.

Syntax:

[C++]

```
void DO_8_BW(int slot, int channel, BOOL data)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

channel : [Input] the digital output channel No.(0~7).

data : [Input] output data “true” or “false”.

Return Value:

None

Example:

```
int slot=1;
int channel=3;
BOOL data=true;
DO_8_BW(slot, channel, data);
//The I-8060 card is plugged in slot 1 of NuWa turns on channel 3.
```

Remark:

This function can be applied on modules: i8060, i8064, i8065, i8066, and i8068.

■ DO_16_BW

Description:

Set the digital output value of the channel No. of the 16-channel Digital Output series modules. The output Value is “true” or “false”.

Syntax:

[C++]

```
void DO_16_BW(int slot, int channel, BOOL data)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

channel : [Input] the digital output channel No.(0~15)

data : [Input] output data “true” or “false”.

Return Value:

None

Example:

```
int slot=1;
int channel=3;
BOOL data=true;
DO_16_BW(slot, channel, data);
//The I-8057 card is plugged in slot 1 of NuWa turns on channel 3
```

Remark:

This function can be applied on modules: i8057; i8056.

■ DO_32_BW

Description:

Set the digital output value of the channel No. on the 32-channel Digital Output series modules. The output Value is “true” or “false”.

Syntax:

[C++]

```
void DO_32_BW(int slot, int channel, BOOL data)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

channel: [Input] the digital output channel No.(0~31)

data : [Input] output data “true” or “false”.

Return Value:

None

Example:

```
int slot=1;
int channel=3;
BOOL data=true;
DO_32_BW(slot, channel, data);
//The I-8041 card is plugged in slot 1 of NuWa and can turn on channel 3.
```

Remark:

This function can be applied on module: i8041.

■ DI_8

Description:

Obtains 8-bit input data from a digital input module. The 0~7 bits of input data correspond to the 0~7 channels of digital input modules respectively.

Syntax:

[C++]

```
unsigned char DI_8(int slot)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

Return Value:

Input data

Example:

```
int slot=1;
unsigned char data;
data=DI_8(slot);
//The I-8058 card is plugged in slot 1 of NuWa and has inputs in channel 0 and 1.
//Returned value: data=0xfC
```

Remark:

This function can be applied on modules: i8052, i8058.

■ DI_16

Description:

This function is used to obtain 16-bit input data from a digital input module. The 0 ~15 bits of input data correspond to the 0~15 channels of digital module's input respectively.

Syntax:

[C++]

```
unsigned int DI_16(int slot)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

Return Value:

Input data

Example:

```
int slot=1;
unsigned int data;
data=DI_16(slot);
//The I-8053 card is plugged in slot 1 of NuWa and has inputs in    //channel 0 and 1.
//Returned value: data=0xfffC
```

Remark:

This function can be applied on modules: i8051, i8053, and i8056.

■ DI_32

Description:

This function is used to obtain 32-bit input data from a digital input module. The 0~31 bits of input data correspond to the 0~31 channels of digital input module respectively.

Syntax:

[C++]

```
unsigned long DI_32(int slot)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

Return Value:

Input data

Example:

```
int slot=1;
unsigned long data;
data=DI_32(slot);
//The I-8040 card plugged is in slot 1 of NuWa and has inputs in channels 0 and 1.
//Returned value: data=0xfffffffC
```

Remark:

This function can be applied on module: i8040.

■ DI_8_BW

Description:

Obtains channel input data from an 8-channel digital input series module. The Input Value is “true” or “false”.

Syntax:

[C++]

```
BOOL DI_8_BW(int slot, int channel)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

channel : [Input] the digital output channel No.(0~7)

Return Value:

Input data

Example:

```
int slot=1;
int channel=3;
BOOL data;
data=DI_8_BW(slot, channel);
//The I-8058 card plugged is in slot 1 of NuWa and has inputs in channel 3.
//Returned value: data=true
```

Remark:

This function can be applied on modules: i8052, i8058.

■ DI_16_BW

Description:

Obtains channel input data from a 16-channel digital input series module. The Input Value is “true” or “false”.

Syntax:

[C++]

```
BOOL DI_16_BW(int slot, int channel)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

channel : [Input] the digital output channel No.(0~15)

Return Value:

Input data

Example:

```
int slot=1;
int channel=3;
BOOL data;
data=DI_16_BW(slot, channel);
//The I-8051 card is plugged in slot 1 of NuWa and has inputs in channel 3.
//Returned value: data=true
```

Remark:

This function can be applied on modules: i8051, i8053, and i8056.

■ DI_32_BW

Description:

Obtains channel input data from a 32-channel digital input series module. The Input Value is “true” or “false”.

Syntax:

[C++]

```
BOOL DI_32_BW(int slot, int channel)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

channel : [Input] the digital output channel No.(0~31)

Return Value:

Input data

Example:

```
int slot=1;
int channel=3;
BOOL data;
data=DI_32_BW(slot, channel);
//The I-8040 card is plugged in slot 1 of NuWa and has inputs in channel 3.
//Returned value: data=true
```

Remark:

This function can be applied on modules: i8040.

■ DIO_DO_8

Description:

This function is used to output 8-bit data to DIO modules. These modules run 8 digital input and 8 digital output channels simultaneously. The 0~7 bits of output data, are mapped onto the 0~7 output channels for their specific DIO modules respectively.

Syntax:

[C++]

```
void DIO_DO_8(int slot, unsigned char data)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

data : [Input] output data.

Return Value:

None

Example:

```
int slot=1;
unsigned char data=3;
DIO_DO_8(slot, data);
//The I-8054 card is plugged in slot 1 of NuWa and can turn on channels 0 and 1.
//It not only outputs a value, but also shows 16LEDs.
```

Remark:

This function can be applied in modules: i8054, i8055, and i8063.

■ DIO_DO_16

Description:

This function is used to output 16-bits of data to DIO modules, which have 16 digital input and 16 digital output channels running simultaneously. The 0~15 bits of output data are mapped onto the 0~15 output channels for their specific DIO modules respectively.

Syntax:

[C++]

```
void DIO_DO_16(int slot, unsigned int data)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

data : [Input] output data.

Return Value:

None

Example:

```
int slot=1;
unsigned int data=3;
DIO_DO_16(slot, data);
//The I-8050 card is plugged in slot 1 of NuWa and can turn on the channels 0 and 1.
//It not only outputs a value, but also shows 32LEDs.
```

Remark:

This function can be applied on modules: i8042, and i8050.

■ DIO_DO_8_BW

Description:

Set the digital output value of the channel No. for the 8-channel Digital I/O series modules. The output Value is “true” or “false”.

Syntax:

[C++]

```
void DIO_DO_8_BW(int slot, int channel, BOOL data)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

channel : [Input] the digital output channel No.(0~7)

data : [Input] output data “true” or “false”.

Return Value:

None

Example:

```
int slot=1;
int channel=3;
BOOL data=true;
DIO_DO_8_BW(slot, channel, data);
//The I-8054 card is plugged in slot 1 of NuWa and can turn on channel 3.
```

Remark:

This function can be applied in these modules: i8054, i8055, and i8063.

■ DIO_DO_16_BW

Description:

Set the digital output value on the channel No. for the 16-channel Digital I/O series modules. The output Value is “true” or “false”.

Syntax:

[C++]

```
void DIO_DO_16_BW(int slot, int channel, BOOL data)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

channel : [Input] the digital output channel No.(0~15)

data : [Input] output data true” or “false”.

Return Value:

None

Example:

```
int slot=1;
int channel=3;
BOOL data=true;
DIO_DO_16_BW(slot, channel, data);
//The I-8042 card is plugged in slot 1 of NuWa and can turn on the channel 3.
```

Remark:

This function can be applied in these modules: i8042, and i8050.

■ DO_8_RB

Description:

Read back the 8-channel digital output value for the I-8000 series modules.

Syntax:

[C++]

```
unsigned char DO_8_RB(int slot)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

Return Value:

8-bit digital output data read back value

Example:

```
int slot=1;
unsigned char outData=0x03;
DO_8(slot, outData);
unsigned char data;
data=DO_8_RB(slot);
//The data read back has a digital output value=0x03.
```

Remark:

This function can be applied on modules: i8060, i8064, i8065, i8066, and i8068.

■ DO_16_RB

Description:

To read back the 16-channel digital output value on the I-8000 series modules.

Syntax:

[C++]

```
unsigned int DO_16_RB(int slot)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

Return Value:

16-bit digital output data read back value

Example:

```
int slot=1;
unsigned int outData=0x03;
DO_16(slot, outData);
unsigned int data;
data=DO_16_RB(slot);
//The data read back has a digital output value=0x03
```

Remark:

This function can be applied on modules: i8057, i8056.

■ DO_32_RB

Description:

To read back the 32-channel digital output value of I-8000 series modules.

Syntax:

[C++]

```
unsigned long DO_32_RB(int slot)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

Return Value:

32-bit digital output data read back value

Example:

```
int slot=1;
unsigned long outData=0x03;
DO_32(slot, outData);
unsigned long data;
data=DO_32_RB(slot);
//The data read back has a digital output value=0x03
```

Remark:

This function can be applied on modules: i8041.

■ DIO_DO_8_RB

Description:

To read back the 8-channel digital output value from the I-8000 digital I/O series modules.

Syntax:

[C++]

```
unsigned char DIO_DO_8_RB(int slot)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

Return Value:

8-bit digital output data read back value

Example:

```
int slot=1;
unsigned char outData=0x03;
DIO_DO_8(slot, outData);
unsigned char data;
data=DIO_DO_8_RB(slot);
//The data read back has a digital output value=0x03.
```

Remark:

This function can be applied on modules: i8054, i8055, and i8063.

■ DIO_DO_16_RB

Description:

To read back the 16-channel digital output value from I-8000 digital I/O series modules.

Syntax:

[C++]

```
unsigned int DIO_DO_16_RB(int slot)
```

Parameter:

slot : [Input] the slot number where the I/O module is plugged into.

Return Value:

16-bit digital output data read back value

Example:

```
int slot=1;
unsigned int outData=0x03;
DIO_DO_16(slot, outData);
unsigned int data;
data=DIO_DO_16_RB(slot);
//The data read back has a digital output value=0x03.
```

Remark:

This function can be applied on modules: i8042, and i8050.

2.4 Watch Dog Timer Functions

■ EnableWDT

Description:

This function can be used to enable or refresh the watch dog timer.

Syntax:

[C++]

```
void EnableWDT (DWORD msecond)
```

Parameter:

msecond : [Input] watch dog timer interval, unit= millisecond

Return Value:

None

Example:

Remark:

■ DisableWDT

Description:

This function is used to disable the watch dog timer.

Syntax:

[C++]

```
void DisableWDT(void)
```

Parameter:

None

Return Value:

None

Example:

Remark:

■ WatchDogSWEven

Description:

This function is used to check whether the system is reset with the watch dog timer. The watch dog timer is started by the EnableWDT function, and stopped by calling the DisableWDT function and refreshed via the EnableWDT function.

Syntax:

[C++]

```
int WatchDogSWEven(void)
```

Parameter:

None

Return Value:

None

Example:

Remark:

■ ClearWDTSWEven

Description:

This function is used to clear the flag that has been reset with the watch dog timer.

Syntax:

[C++]

void ClearWDTSWEven (void)

Parameter:

None

Return Value:

None

Example:

Remark:

2.5 EEPROM Read/Write Functions

■ ReadEEP

Description:

Read one byte data from EEPROM. There is a 16K-byte EEPROM in the main control unit in the NuWa system. This EEPROM is divided into 256 blocks (0 to 255), and each block is 64 bytes in length from offset 0 to 63. This EEPROM with its accessing APIs provides another mechanism for storing critical data inside non-volatile memory.

Syntax:

[C++]

```
unsigned char ReadEEP(int block, int offset)
```

Parameter:

block : [Input] the block number of EEPROM.

offset: [Input] the offset within the block.

Return Value:

Data read from the EEPROM.

Example:

```
int block, offset;
unsigned char data;
data= ReadEEP(block, offset);
//Returned value: data= read an 8-bit value from the EEPROM (block & offset)
```

Remark:

■ WriteEEP

Description:

To write one byte of data to the EEPROM. There is a 16K-byte EEPROM in the main control unit in the NuWa system. This EEPROM is divided into 256 blocks (0 to 255), and each block is 64 bytes in length from the offset of 0 to 63. This EEPROM with its accessing APIs, provides another mechanism for storing critical data inside non-volatile memory.

Syntax:

[C++]

```
void WriteEEP(int block, int offset, unsigned char ucData)
```

Parameter:

block :	[Input] the block number of EEPROM.
offset:	[Input] the offset within the block.
ucData:	[Input] data to write to EEPROM.

Return Value:

None

Example:

```
int block, offset;
unsigned char data=10;
WriteEEP(block, offset, data);
//Writes a 10 value output to the EEPROM (block & offset) location
```

Remark:

2.6 Analog Input Functions

■ Init_8017H

Description:

This function is used to initialize the I-8017H module (Analog input module) into the specified slot. Users must execute this function once before trying to use other functions within I-8017H.

Syntax:

[C++]

```
void Init_8017H(int slot)
```

Parameter:

slot : [Input] specified slot of the NuWa system (slot 1)

Return Value:

None

Example:

```
int slot=1;
Init_8017H(slot);
//The I-8017H card is plugged in slot 1 of NuWa and initializes the module I-8017H.
```

Remark:

This function can be applied on module: i8017H.

■ Set_8017H_LED

Description:

Turns the I-8017H modules LED's on/off. They can be used to act as an alarm.

Syntax:

[C++]

```
void Set_8017H_LED(int slot, unsigned int led)
```

Parameter:

slot : [Input] specified slot of the NuWa system (Slot 1)

led : [Input] range from 0 to 0xffff

Return Value:

None

Example:

```
int slot=1;
unsigned int led=0x0001;
Set_8017H_LED(slot, led);
//The LED will have a L-LED light on channel 0 on the I-8017H card which is plugged in slot 1 on the NuWa.
```

Remark:

This function can be applied on module: i8017H.

■ Set_8017H_Channel_Gain_Mode

Description:

This function is used to configure the range and mode of the analog input channel for the module I-8017H in the specified slot before using ADC (analog to digital converter).

Syntax:

[C++]

```
void Set_8017H_Channel_Gain_Mode(int slot, int ch, int gain, int mode)
```

Parameter:

slot : [Input] Specify the slot in the NuWa system (Slot 1)

ch : [Input] Specify the I-8017H channel (Range: 0 to 7)

gain : [Input] input range:

0: +/- 10.0V,

1: +/- 5.0V,

2: +/- 2.5V,

3: +/- 1.25V,

4: +/- 20mA.

mode : [Input] **0: normal mode** (polling)

Return Value:

None

Example:

```
int slot=1,ch=0,gain=0;
float data;
Set_8017H_Channel_Gain_Mode(slot, ch, gain,0);
data=Get_AD_FValue(gain);
//The I-8017H card is plugged in slot 1 of NuWa, and the data value from channel 0 for I-8017H, and the data range is: -10 ~ +10 V.
```

Remark:

This function can be applied on module: i8017H.

■ Get_AD_FValue

Description:

Obtain the analog input voltage value from the analog input module in the float format according to the configuration of function Set_8017H_Channel_Gain_Mode.

Syntax:

[C++]

```
float Get_AD_FValue(int gain)
```

Parameter:

gain : [Input] input range

0: +/- 10.0V,

1: +/- 5.0V,

2: +/- 2.5V,

3: +/- 1.25V,

4: +/- 20mA.

Return Value:

Return (float): The analog input value.

Example:

```
int slot=1,ch=0,gain=1;
float data;
Set_8017H_Channel_Gain_Mode(slot, ch, gain,0);
data=Get_AD_FValue(gain);
// The I-8017H card is plugged into slot 1 of NuWa and the data value from channel 0 for I-8017H, and the data range is: -5 ~ +5 V.
```

Remark:

This function can be applied on module: i8017H.

■ Get_AD_IValue

Description:

This function is used to obtain the analog input current value from an analog input module in the float format according to the configuration in function Set_8017H_Channel_Gain_Mode.

Syntax:

[C++]

```
float Get_AD_IValue(void)
```

Parameter:

None

Return Value:

Return (float): The analog input current value (mA).

Example:

```
int slot=1,ch=0,gain=4;
float data;
Set_8017H_Channel_Gain_Mode(slot, ch, gain,0);
data=Get_AD_IValue(gain);
// The I-8017H card is plugged into slot 1 of NuWa, and the data value from channel 0 in I-8017H, and the data range is: 0 ~ 20 mA.
```

Remark:

This function can be applied on module: i8017H.

■ Get_AD_HValue

Description:

This function is used to obtain the voltage analog input value from the analog input module in the HEX format according to the configuration in function Set_8017H_Channel_Gain_Mode.

Syntax:

[C++]

```
short Get_AD_HValue(void)
```

Parameter:

None

Return Value:

Return (Hex): The voltage analog input value.

Example:

```
int slot=1,ch=0,gain=4;
short data;
Set_8017H_Channel_Gain_Mode(slot, ch, gain,0);
data=Get_AD_HValue(gain);
//The I-8017H card is plugged in slot 1 of NuWa, and the data value from channel 0 in I-8017H, and the data range is: 0x0000 ~ 0x3fff.
```

Remark:

This function can be applied on module: i8017H.

■ I8017H_AD_POLLING

Description:

This function is used to get the analog input values of the specific channel from an analog input module and convert the value in HEX format according to the configuration of the slot, the gain and the data number.

Syntax:

[C++]

```
int I8017H_AD_POLLING(int slot, int ch, int gain, int datacount, int *DataPtr)
```

Parameter:

slot : [Input] Specified slot in the NuWa system (Slot 1)

ch : [Input] Specified channel for I-8017H (Range: 0 to 7)

gain : [Input] Input range:

0: +/- 10.0V,

1: +/- 5.0V,

2: +/- 2.5V,

3: +/- 1.25V,

4: +/- 20mA.

datacount : [Input] Range from 1 to 8192, total ADCs number

*DataPtr : [Output] The starting address of data array[ ] and the array size must be equal to or bigger than the datacount.

Return Value:

0: indicates success.

1: indicates failure.

Example:

```
int slot=1, ch=0, gain=0, count=10, data[10];
I8017H_AD_POLLING(slot, ch, gain, datacount, data);
//You gain ten record data values via channel 0 in the i-8017H module.
```

Remark:

■ ARRAY_HEX_TO_FLOAT_ALL

Description:

This function is used to convert the data from hex to float values based on the configuration of the slot, gain and data length. (Voltage or current)

Syntax:

[C++]

```
void ARRAY_HEX_TO_FLOAT_ALL(int *HexValue, float *Ivalue, int slot,
                            int gain, int len)
```

Parameter:

*HexValue :	[Input] data array in integer type before converting.
*Ivalue :	[Output] Converted data array in float type (voltage or current).
slot :	[Input] Specify the slot in the NuWa system (Slot 1)
gain :	[Input] input range:
len :	[Input] ADC data length.

Return Value:

None

Example:

```
int slot=1, ch=0, gain=0, count=10, data[10];
float fdata[10];
I8017H_AD_POLLING(slot, ch, gain, datacount, data);
ARRAY_HEX_TO_FLOAT_ALL(data, fdata, slot, gain, int len);
//You gain ten record HEX values to change ten record float values.
```

Remark:

This function can be applied on module: i8017H.

2.7 Analog Output Functions

■ I8024_Initial

Description:

This function is used to initialize the module I-8024 in the specified slot. You must implement this function once before you try to use the other I-8024 functions.

Syntax:

[C++]

```
void I8024_Initial(int slot)
```

Parameter:

slot : [Input] Specify the NuWa system slot (Slot 1)

Return Value:

None

Example:

```
int slot=1;
I8024_Initial(slot);
//The I-8024 card is plugged into slot 1 of NuWa and initializes the I-8024 module.
```

Remark:

This function can be applied on module: i8024.

■ I8024_VoltageOut

Description:

This function is used to send the voltage float value to the I-8024 module with the specified channel and slot in the NuWa system.

Syntax:

[C++]

```
void I8024_VoltageOut(int slot, int ch, float data)
```

Parameter:

slot : [Input] Specified the NuWa system slot (Slot 1)

ch : [Input] Output channel (Range: 0 to 3)

data : [Input] Output data with engineering unit (Voltage Output: -10~ +10)

Return Value:

None

Example:

```
int slot=1, ch=0;
float data=3.0f;
I8024_VoltageOut(slot, ch, data);
//The I-8024 module output the 3.0V voltage from the channel 0.
```

Remark:

This function can be applied on module: i8024.

■ I8024_CurrentOut

Description:

This function is used to initialize the I-8024module in the specified slot for current output. Users must call this function once before trying to use the other I-8024 functions for current output.

Syntax:

[C++]

```
void I8024_CurrentOut(int slot, int ch, float data)
```

Parameter:

slot : [Input] Specify the NuWa system slot (Slot 1)

ch : [Input] Output channel (Range: 0 to 3)

data : [Input] Output data with engineering unit (Current Output: 0~20 mA)

Return Value:

None

Example:

```
int slot=1, ch=0;
float data=10.0f;
I8024_CurrentOut(slot, ch, data);
//Output the 10.0mA current from the channel 0 of I-8024 module.
```

Remark:

This function can be applied on module: i8024.

■ I8024_VoltageHexOut

Description:

This function is used to send the voltage value in hex format to the specified channel in the I-8024 module, which is plugged into the slot in the NuWa system.

Syntax:

[C++]

```
void I8024_VoltageHexOut(int slot, int ch, int data)
```

Parameter:

slot : [Input] Specify the NuWa system slot (Slot 1)

ch : [Input] Output channel (Range: 0 to 3)

data : [Input] Output data with hexadecimal
(data range: 0h ~ 3FFFh→ Voltage Output: -10. ~ +10. V)

Return Value:

None

Example:

```
int slot=1, ch=0; data=0x3000;
I8024_VoltageHexOut(slot, ch, data);
//The I-8024 module output the 5.0V voltage from the channel 0.
```

Remark:

This function can be applied on module: i8024.

■ I8024_CurrentHexOut

Description:

This function is used to send the current value in Hex format to the specified channel in the analog output module I-8024, which is plugged into the slot in the NuWa system.

Syntax:

[C++]

```
void I8024_CurrentHexOut(int slot, int ch, int data)
```

Parameter:

slot : [Input] Specify the NuWa system slot (Slot 1)

ch : [Input] Output channel (Range: 0 to 3)

data : [Input] Output data with hexadecimal

(data range: 0h ~ 3FFFh → Current Output: 0. ~ +20.mA)

Return Value:

None

Example:

```
int slot=1, ch=0; data=0x2000;
I8024_CurrentHexOut(slot, ch, data);
//Output the 10.0mA current from the channel 0 of I-8024 module.
```

Remark:

This function can be applied on module: i8024.

■ I8024_VoltageOutReadBack

Description:

This function is used to read back the output data in float format from the specified channel on the I-8024 module in the NuWa system.

Syntax:

[C++]

```
float I8024_VoltageOutReadBack(int slot, int ch)
```

Parameter:

slot : [Input] Specify the NuWa system slot (Slot 1)

ch : [Input] Output channel (Range: 0 to 3)

Return Value:

a float value.

Example:

```
int slot=1, ch=0;
float data;
data=I8024_VoltageOutReadBack(slot, ch);
//Users can read back channel 0 on the I-8024 module outputted voltage value for the last outputted value.
```

Remark:

This function can be applied on module: i8024.

■ I8024_CurrentOutReadBack

Description:

This function is used to read back the current output value in float format from the specified channel on I-8024 module in the specific slot of the NuWa system.

Syntax:

[C++]

```
float I8024_CurrentOutReadBack(int slot, int ch)
```

Parameter:

slot : [Input] Specify the NuWa system slot (Slot 1)

ch : [Input] Output channel (Range: 0 to 3)

Return Value:

a float value.

Example:

```
int slot=1, ch=0;
float data;
data= I8024_CurrentOutReadBack(slot, ch);
//You can read back channel 0 on the I-8024 module outputted current value at last time.
```

Remark:

This function can be applied on module: i8024.

■ I8024_VoltageHexOutReadBack

Description:

This function is used to read back the voltage output value in hex format from the specified channel on the analog output module I-8024 in the specific slot of the NuWa system.

Syntax:

[C++]

```
int I8024_VoltageHexOutReadBack(int slot, int ch)
```

Parameter:

slot : [Input] Specify the slot of the NuWa system (Slot 1)

ch : [Input] Output channel (Range: 0 to 3)

Return Value:

return (hex): a 32 bits integer value.
(value range: 0h ~ 3FFFh Voltage → Output: -10. ~ +10. V)

Example:

```
int slot=1, ch=0, data;
data=I8024_VoltageHexOutReadBack(slot, ch);
//You can read back channel 0 on the I-8024 modules outputted HEX voltage //value at last time.
```

Remark:

This function can be applied on module: i8024.

■ I8024_CurrentHexOutReadBack

Description:

This function is used to read back the current output value in Hex format from the specified channel of I-8024 module in the slot of the NuWa system.

Syntax:

[C++]

```
int I8024_CurrentHexOutReadBack(int slot, int ch)
```

Parameter:

slot : [Input] Specify the slot of the NuWa system (Slot 1)

ch : [Input] Output channel (Range: 0 to 3)

Return Value:

return (hex): a 32 bits integer value.

(value range: 0h ~ 3FFFh → Current Output: 0. ~ +20.mA)

Example:

```
int slot=1, ch=0, data;
data=I8024_CurrentOutReadBack(slot, ch);
//Users can read back the channel 0 of the I-8024 module outputted HEX current value at last time.
```

Remark:

This function can be applied on module: i8024.

2.8 3-axis Encoder Functions

■ i8090_REGISTRATION

Description:

This function is used to assign a card number “cardNo” to the I-8090 module in the specific slot. In order to distinguish more than one I-8090 card in the NuWa platform, the I-8090 modules should be registered before applying it. If there is no I-8090 in NuWa with the given address, this function will return 0 which means a failure.

Syntax:

[C++]

```
unsigned char i8090_REGISTRATION(unsigned char cardNo, int slot)
```

Parameter:

cardNO : [Input] 0~19, The assigned number to the i8090 card.

slot : [Input] Specify the NuWa system slot (Slot 1)

Return Value:

1: Success registration

0: Failure registration

Example:

```
#define CARD1    1
int slot=1;
i8090_REGISTRATION(CARD1, slot);
//The I-8090 card plugged in slot 1 of NuWa
```

Remark:

This function can be applied on module: i8090.

■ i8090_INIT_CARD

Description:

This function is applied to reset the counter values of three axes with a specific card number, and set the modes of three counters.

Syntax:

[C++]

```
void i8090_INIT_CARD(unsigned char cardNo, unsigned char x_mode,
                     unsigned char y_mode, unsigned char z_mode)
```

Parameter:

cardNO : [Input] 0~19, The specified card number.

x_mode : [Input] The X axis counter mode. Refer to the Remarks.

y_mode : [Input] The Y axis counter mode. Refer to the Remarks.

z_mode : [Input] The Z axis counter mode. Refer to the Remarks.

Return Value:

None

Example:

```
#define CARD1    1
int slot=1;
i8090_REGISTRATION(CARD1, slot);
i8090_INIT_CARD(CARD1, ENC_QUADRANT, ENC_QUADRANT,ENC_QUADRANT);
//The x, y, z axis encoder are ENC_QUADRANT mode.
```

Remark:

1. This function can be applied on module: i8090.
2. For more information about counting modes, refer to the Registers of i-8090 board chapter in the i-8090 user's manual.

■ i8090_GET_ENCODER

Description:

This function is used to obtain the counter value of the selected axis on the specified encoder card. This counter value is defined in the short (16-bit) format.

Syntax:

[C++]

```
unsigned int i8090_GET_ENCODER(unsigned char cardNo, unsigned char axis)
```

Parameter:

cardNO : [Input] 0~19, The selected card number.

axis : [Input] The selected axis. 1: X-axis; 2: Y-axis; 3: Z-axis

Return Value:

A 16 bits unsigned integer value.

Example:

```
#define CARD1    1
int slot=1;
unsigned int data;
i8090_REGISTRATION(CARD1, slot);
i8090_INIT_CARD(CARD1, ENC_QUADRANT, ENC_QUADRANT,ENC_QUADRANT);
data= i8090_GET_ENCODER(CARD1, X_axis);
//The data value is the X-axis encoder value.
```

Remark:

This function can be applied on module: i8090.

■ i8090_RESET_ENCODER

Description:

This function is used to reset the counter value of the selected axis on the specified encoder card.

Syntax:

[C++]

```
void i8090_RESET_ENCODER(unsigned char cardNo, unsigned char axis)
```

Parameter:

cardNO : [Input] 0~19, The selected card number.

axis : [Input] The selected axis. 1: X-axis; 2: Y-axis; 3: Z-axis

Return Value:

None

Example:

```
#define CARD1    1
int slot=1;
i8090_REGISTRATION(CARD1, slot);
i8090_INIT_CARD(CARD1, ENC_QUADRANT, ENC_QUADRANT,ENC_QUADRANT);
i8090_RESET_ENCODER(CARD1, X_axis);
//The X-axis encoder value set zero.
```

Remark:

This function can be applied on module: i8090.

■ i8090_GET_ENCODER32

Description:

This function is used to obtain the counter value of the selected axis on the specific encoder card. The counter value is defined in the long (32-bit) format. Users must call the i8090_ENCODER32_ISR() function before using this function.

Syntax:

[C++]

```
long i8090_GET_ENCODER32(unsigned char cardNo, unsigned char axis)
```

Parameter:

cardNO : [Input] 0~19, select which card.

axis : [Input] The selected axis. 1: X-axis; 2: Y-axis; 3: Z-axis

Return Value:

A 32 bits integer value.

Example:

```
#define CARD1   1
int slot=1;
long data;
i8090_REGISTRATION(CARD1, slot);
i8090_INIT_CARD(CARD1, ENC_QUADRANT, ENC_QUADRANT,ENC_QUADRANT);
i8090_ENCODER32_ISR(CARD1);
data=i8090_GET_ENCODER32(CARD1, X_axis);
//The data value is the X-axis encoder value.
```

Remark:

This function can be applied on module: i8090.

■ i8090_RESET_ENCODER32

Description:

This function is applied to reset the counter variable of the function i8090_Get_Encoder32.

Syntax:

[C++]

```
void i8090_RESET_ENCODER32(unsigned char cardNo, unsigned char axis)
```

Parameter:

cardNO : [Input] 0~19, The selected card number.

axis : [Input] The selected axis. 1: X-axis; 2: Y-axis; 3: Z-axis

Return Value:

None

Example:

```
#define CARD1    1
int slot=1;
i8090_REGISTRATION(CARD1, slot);
i8090_INIT_CARD32(CARD1, ENC_QUADRANT, ENC_QUADRANT,ENC_QUADRANT);
i8090_RESET_ENCODER(CARD1, X_axis);
//The X-axis encoder value set zero.
```

Remark:

This function can be applied on module: i8090.

■ i8090_GET_INDEX

Description:

This function is used to get the value of the “INDEX” register on the specific card.

Syntax:

[C++]

```
unsigned char i8090_GET_INDEX(unsigned char cardNo)
```

Parameter:

cardNO : [Input] 0~19, The specific card number.

Return Value:

Register	Add.	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INDEX	0x08	R						ZI	YI	XI

Example:

```
#define CARD1    1
unsigned char data;
data=i8090_GET_INDEX(CARD1);
//Returned value: data=0x07
```

Remark:

This function can be applied on module: i8090. The index input C+/C- can read out from this register. These bits are highly active.

XI : Indicate the index of X-axis.
YI : Indicate the index of Y-axis.
ZI : Indicate the index of Z-axis.

■ i8090_ENCODER32_ISR

Description:

This function is used to calculate the pulse value between present and last time with a ”long type” format. According to this idea, the i8090_ENCODER32_ISR() function should be executed periodically (2~10ms) using the timer interrupt or manual method.

Syntax:

[C++]

```
void i8090_ENCODER32_ISR(unsigned char cardNo)
```

Parameter:

cardNO : [Input] 0~19, The selected card number.

Return Value:

None

Example:

```
#define CARD1   1
long data;
i8090_ENCODER32_ISR(CARD1);
data=i8090_GET_ENCODER32(CARD1, X_axis);
//The i8090_ENCODER32_ISR() should be called in 2~20ms.
```

Remark:

This function can be applied on module: i8090.

2.9 2-axis Stepper/Servo Functions

■ i8091_REGISTRATION

Description:

This function is used to assign a card number “cardNo” to the I-8091 card in the specific slot. In order to distinguish more than one of the I-8091 cards in NuWa platform, the I-8091 cards should be registered before using them. If there is no I-8091 at the given address, this command will return 0 which is a failure value.

Syntax:

[C++]

```
unsigned char i8091_REGISTRATION(unsigned char cardNo, int slot)
```

Parameter:

cardNO : [Input] The board number (0~19)

slot : [Input] The specific slot which i8091 card is pluged in (1~7)

Return Value:

1: card exist

0: card not exist

Example:

```
#define CARD1    1
int slot=1;
i8091_REGISTRATION(CARD1, slot);
//The I-8091 card is plugged into slot 1 of NuWa.
```

Remark:

This function can be applied on module: i8091.

■ i8091_RESET_SYSTEM

Description:

This function is used to reset and terminate the running command in the 8091 module. Users can apply this command in software emergencies as a stop function. It can also clear all the card settings. After calling this function, users need to configure all the parameters in the I-8091 card.

Syntax:

[C++]

```
void i8091_RESET_SYSTEM(unsigned char cardNo)
```

Parameter:

cardNO : [Input] 0~19, The selected card number.

Return Value:

None

Example:

```
#define CARD1    1
int slot=1;
i8091_REGISTRATION(CARD1, slot);
i8091_RESET_SYSTEM(CARD1);
//The I-8091 card plugged in slot 1 of NuWa
```

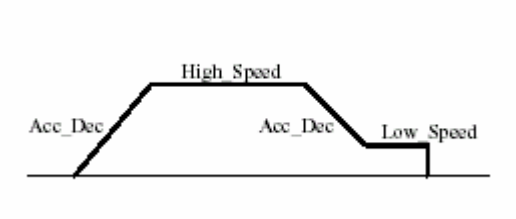
Remark:

This function can be applied on module: i8091.

■ i8091_SET_VAR

Description:

This function is used to set the DDA cycle, plus accelerating/decelerating speeds, low-speed and the high-speed values in the specified I-8091 card.



Syntax:

[C++]

```
void i8091_SET_VAR(unsigned char cardNo, unsigned char DDA_cycle,
    unsigned char Acc_Dec, unsigned int Low_Speed, unsigned int High_Speed)
```

Parameter:

cardNO : [Input] 0~19, The selected card number.

DDA_cycle : [Input] 1<=DDA_cycle<=254.

Acc_Dec : [Input] 1<=Acc_Dec<=200.

Low_Speed : [Input] 1<=Low_Speed<=200.

High_Speed :[Input] Low_Speed<=High_Speed<=2047.

Return Value:

None

Example:

```
#define CARD1    1
int slot=1;
i8091_REGISTRATION(CARD1, slot);
i8091_SET_VAR(CARD1, 5, 2, 10, 150);
```

Remark:

This function can be applied on module: i8091.

■ i8091_SET_DEFDIR

Description:

This function is used to define the rotating directions of X and Y axes for controlling motors. Sometimes, the output direction of X-axis or Y-axis is in an undesired direction because of the wire connection to the motor or gear train mechanism. In order to unify the output direction, the CW/FW directions of X/Y axis are defined as an outside going motion through the motor control, and similarly the CCW/BW directions are defined as the inward motion through the motor control.

Syntax:

[C++]

```
void i8091_SET_DEFDIR(unsigned char cardNo, unsigned char defdirX,
                      unsigned char defdirY)
```

Parameter:

cardNO : [Input] The board number (0~19)

defdirX : [Input] X axis direction definition
(0:NORMAL_DIR, 1:REVERSE_DIR)

defdirY : [Input] Y axis direction definition
(0:NORMAL_DIR, 1:REVERSE_DIR)

Return Value:

None

Example:

```
#define CARD1    1
int slot=1;
i8091_REGISTRATION(CARD1, slot);
i8091_SET_DEFDIR(CARD1, 0, 0);
```

Remark:

This function can be applied on module: i8091.

■ i8091_SET_MODE

Description:

This function is used to set the motor control modes of X and Y axes in the specific I-8091 card.

Syntax:

[C++]

```
void i8091_SET_MODE(unsigned char cardNo, unsigned char modeX,
                    unsigned char modeY)
```

Parameter:

cardNO : [Input] The board number (0~19)

modeX : [Input] X axis output mode
(0:CW/CCW mode, 1:Pulse/Direction mode)

modeY : [Input] Y axis output mode
(0:CW/CCW mode, 1:Pulse/Direction mode)

Return Value:

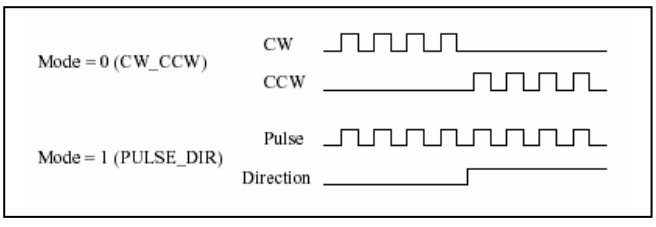
None

Example:

```
#define CARD1    1
int slot=1;
i8091_REGISTRATION(CARD1, slot);
i8091_SET_MODE(CARD1, 0, 0);
```

Remark:

This function can be applied on module: i8091.



■ i8091_SET_SERVO_ON

Description:

This function is used to turn the servo function on/off to get the motor driver ready or to stop motor control.

Syntax:

[C++]

void i8091_SET_SERVO_ON(unsigned char cardNo, unsigned char sonX, unsigned char sonY)

Parameter:

cardNO : [Input] The board number (0~19)

modeX : [Input] X-axis servo/hold on switch (1:ON , 0:OFF)

modeY : [Input] X-axis servo/hold on switch (1:ON , 0:OFF)

Return Value:

None

Example:

```
#define CARD1    1
int slot=1;
i8091_REGISTRATION(CARD1, slot);
i8091_SET_SERVO_ON(CARD1, 1, 1);
```

Remark:

This function can be applied on module: i8091.

■ i8091_SET_NC

Description:

This function is used to set all of the following limit switches to N.C.(normal close) or N.O.(normal open). If users set the “sw” parameter as N.O, then those limit switches are active low. If users set the value as N.C, those limit switches are then “active high”. The auto-protection system will automatically change the judgments, whatever it is, to N.O. or N.C.

Limit switches: ORG1, LS11, LS14, ORG2, LS21, LS24, EMG.

Syntax:

[C++]

```
void i8091_SET_NC(unsigned char cardNo, unsigned char sw)
```

Parameter:

cardNO : [Input] The board number (0~19)

sw : [Input] 0(NO) normal open (default), 1(YES) normal close.

Return Value:

None

Example:

```
#define CARD1    1
int slot=1;
i8091_REGISTRATION(CARD1, slot);
i8091_SET_NC(CARD1, 1, 1);
```

Remark:

This function can be applied on module: i8091.

■ i8091_STOP_X

Description:

This function is used to stop the X-axis from action immediately.

Syntax:

[C++]

```
void i8091_STOP_X(unsigned char cardNo)
```

Parameter:

cardNO : [Input] The board number (0~19)

Return Value:

None

Example:

```
#define CARD1    1
int slot=1;
i8091_REGISTRATION(CARD1, slot);
i8091_STOP_X(CARD1);
//The X-axis is stopped.
```

Remark:

This function can be applied on module: i8091.

This command would stop the X axis immediately.

■ i8091_STOP_Y

Description:

This function is used to stop the Y-axis from action immediately.

Syntax:

[C++]

```
void i8091_STOP_Y(unsigned char cardNo)
```

Parameter:

cardNO : [Input] The board number (0~19)

Return Value:

None

Example:

```
#define CARD1    1
int slot=1;
i8091_REGISTRATION(CARD1, slot);
i8091_STOP_Y(CARD1);
//The Y-axis is stopped.
```

Remark:

This function can be applied on module: i8091.

This command would stop the Y axis immediately.

■ i8091_STOP_ALL

Description:

This function is used to stop both the X and Y axis immediately. It will clear all commands that are pending in the FIFO.

Syntax:

[C++]

```
void i8091_STOP_ALL(unsigned char cardNo)
```

Parameter:

cardNO : [Input] The board number (0~19)

Return Value:

None

Example:

```
#define CARD1    1
int slot=1;
i8091_REGISTRATION(CARD1, slot);
i8091_STOP_ALL(CARD1);
//The X-axis and Y-axis are stopped.
```

Remark:

This function can be applied on module: i8091.

■ i8091_EMG_STOP

Description:

This function is the same as the i8091_STOP_ALL function, but can only be used in the interrupt routine. It can clear all the commands that are pending in the FIFO.

Syntax:

[C++]

```
void i8091_EMG_STOP(unsigned char cardNo)
```

Parameter:

cardNO : [Input] The board number (0~19)

Return Value:

None

Example:

```
#define CARD1    1
int slot=1;
i8091_REGISTRATION(CARD1, slot);
i8091_EMG_STOP(CARD1);
//The X-axis and Y-axis are stopped.
```

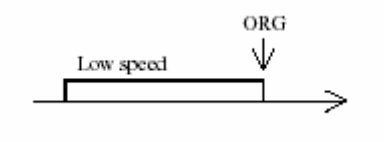
Remark:

This function can be applied on module: i8091.

■ i8091_LSP_ORG

Description:

This function is used to stop the specific (X/Y) axis in low-speed movement when the ORG1/ORG2 limit switch is in contact.



Syntax:

[C++]

```
void i8091_LSP_ORG(unsigned char cardNo, unsigned char DIR,
                                         unsigned char AXIS)
```

Parameter:

cardNO : [Input] The board number (0~19)

DIR : [Input] The moving direction. (0:CW, 1:CCW)

AXIS : [Input] The selected axis. (1: X_axis, 2: Y_axis)

Return Value:

None

Example:

```
#define CARD1   1
int slot=1;
i8091_REGISTRATION(CARD1, slot);
i8091_LSP_ORG(CARD1, CCW, X_axis);
i8091_LSP_ORG(CARD1, CCW, Y_axis);
```

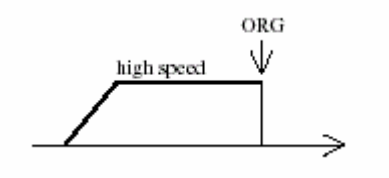
Remark:

This function can be applied on module: i8091.

■ i8091_HSP_ORG

Description:

This function drives the specific (X/Y) axis to search for home position (ORG1/ORG2) in the high-speed mode motion and stops the motion when the ORG1/ORG2 limit switch is touched.



Syntax:

[C++]

```
void i8091_HSP_ORG(unsigned char cardNo, unsigned char DIR,
                                         unsigned char AXIS)
```

Parameter:

cardNO : [Input] The board number (0~19)

DIR : [Input] The moving direction. (0:CW, 1:CCW)

AXIS : [Input] The selected axis. (1: X_axis, 2: Y_axis)

Return Value:

None

Example:

```
#define CARD1    1
int slot=1;
i8091_REGISTRATION(CARD1, slot);
i8091_HSP_ORG(CARD1, CCW, X_axis);
i8091_HSP_ORG(CARD1, CCW, Y_axis);
```

Remark:

This function can be applied on module: i8091.

■ i8091_LSP_PULSE_MOVE

Description:

This function drives the specified axis into motion toward the desired position from the given pulse number in low-speed mode.

Syntax:

[C++]

void i8091_LSP_PULSE_MOVE(unsigned char cardNo, unsigned char AXIS, long pulseN)

Parameter:

cardNO : [Input] The board number (0~19)

AXIS : [Input] The selected axis (1: X_axis, 2: Y_axis)

pulseN : [Input] The moving number of pulse.

Return Value:

None

Example:

```
i8091_LSP_PULSE_MOVE(CARD1, X_axis, 20000);
i8091_LSP_PULSE_MOVE(CARD1, X_axis, -2000);
i8091_LSP_PULSE_MOVE(CARD1, Y_axis, 20000);
i8091_LSP_PULSE_MOVE(CARD1, Y_axis, -2000);
//where
//when pulseN>0, move toward CW/FW direction
//when pulseN<0, move toward CCW/BW direction
```

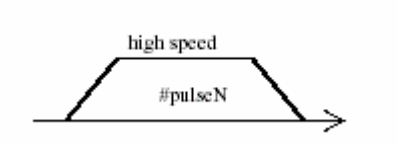
Remark:

This function can be applied on module: i8091.

■ i8091_HSP_PULSE_MOVE

Description:

This function drives the specified axis into motion toward the desired position from the given pulse number in high-speed mode.



Syntax:

[C++]

void i8091_HSP_PULSE_MOVE(unsigned char cardNo, unsigned char AXIS, long pulseN)

Parameter:

cardNO : [Input] The board number (0~19)

AXIS : [Input] The selected axis (1: X_axis, 2: Y_axis)

pulseN : [Input] The moving number of pulse.

Return Value:

None

Example:

```
i8091_HSP_PULSE_MOVE(CARD1, X_axis, 20000);
i8091_HSP_PULSE_MOVE(CARD1, X_axis, -2000);
i8091_HSP_PULSE_MOVE(CARD1, Y_axis, 20000);
i8091_HSP_PULSE_MOVE(CARD1, Y_axis, -2000);
//where
//when pulseN>0, move toward CW/FW direction
//when pulseN<0, move toward CCW/BW direction
```

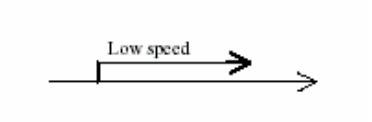
Remark:

This function can be applied on module: i8091.

■ i8091_LSP_MOVE

Description:

This function drives the specified axis into motion toward the selected direction in low-speed mode. It can be stopped by the i8091_STOP_X function, or the i8091_STOP_Y function, or the i8091_STOP_ALL function.



Syntax:

[C++]

void i8091_LSP_MOVE(unsigned char cardNo, unsigned char DIR,
unsigned char AXIS)

Parameter:

cardNO : [Input] The board number (0~19)

DIR : [Input] The moving direction. (0:CW, 1:CCW)

AXIS : [Input] The selected axis (1: X_axis, 2: Y_axis)

Return Value:

None

Example:

```
i8091_LSP_MOVE(CARD1, CW, X_axis);
i8091_LSP_MOVE(CARD1, CW, Y_axis);
```

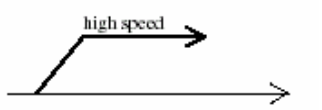
Remark:

This function can be applied on module: i8091.

■ i8091_HSP_MOVE

Description:

This function drives the specified axis into motion toward the selected direction in high-speed mode. It can be stopped by the i8091_STOP_X, or i8091_STOP_Y, or i8091_STOP_ALL functions.



Syntax:

[C++]

```
void i8091_HSP_MOVE(unsigned char cardNo, unsigned char DIR,
                    unsigned char AXIS)
```

Parameter:

cardNO :	[Input] The board number (0~19)
DIR :	[Input] The moving direction. (0:CW, 1:CCW)
AXIS :	[Input] The selected axis (1: X_axis, 2: Y_axis)

Return Value:

None

Example:

```
i8091_HSP_MOVE(CARD1, CW, X_axis);
i8091_HSP_MOVE(CARD1, CW, Y_axis);
```

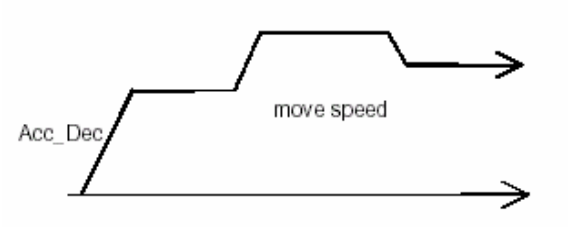
Remark:

This function can be applied on module: i8091.

■ i8091_CSP_MOVE

Description:

This function is used to accelerate/decelerate the motor on the selected axis into motion up/down to the desired speed. If commands are continuously applied to the I-8091, then this function can dynamically change the motor speed. The rotating motor can be stopped by using the following commands: i8091_STOP_X(), i8091_STOP_Y(), i8091_STOP_ALL(), or i8091_SLOW_STOP().



Syntax:

[C++]

```
void i8091_CSP_MOVE(unsigned char cardNo, unsigned char dir,
                    unsigned char axis, unsigned int move_speed)
```

Parameter:

cardNO : [Input] The board number (0~19)

dir : [Input] The moving direction. (0:CW, 1:CCW)

axis : [Input] The selected axis. (1: X_axis, 2: Y_axis)

move_speed :[Input] 0 < move_speed <= 2040

Return Value:

None

Example:

```
i8091_CSP_MOVE(CARD1, CW, X_axis, 10);   //Delay(10000);
i8091_CSP_MOVE(CARD1, CW, X_axis, 20);
```

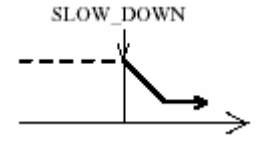
Remark:

This function can be applied on module: i8091.

■ i8091_SLOW_DOWN

Description:

This function is used to decelerate the motor motion to a specific low speed in order to be able to call either the i8091_STOP_X(), or i8091_STOP_Y(), or i8091_STOP_ALL functions so that the motor's motion can be stopped.



Syntax:

[C++]

void i8091_SLOW_DOWN(unsigned char cardNo, unsigned char AXIS)

Parameter:

cardNO : [Input] The board number (0~19)

AXIS : [Input] The selected axis (1: X_axis, 2: Y_axis)

Return Value:

None

Example:

```
i8091_HSP_MOVE(CARD1, CW, X_axis);
i8091_SLOW_DOWN(CARD1, X_axis);
i8091_STOP_X(CARD1);
```

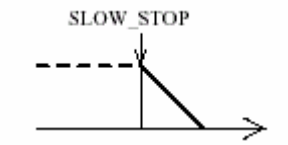
Remark:

This function can be applied on module: i8091.

■ i8091_SLOW_STOP

Description:

This function is used to decelerate the speed of a specified axis and then stop it (as shown in following figure).



Syntax:

[C++]

void i8091_SLOW_STOP(unsigned char cardNo, unsigned char AXIS)

Parameter:

cardNO : [Input] The board number (0~19)

AXIS : [Input] The selected axis (1: X_axis, 2: Y_axis)

Return Value:

None

Example:

```
i8091_HSP_MOVE(CARD1, CW, Y_axis);
i8091_SLOW_STOP(CARD1, Y_axis);
```

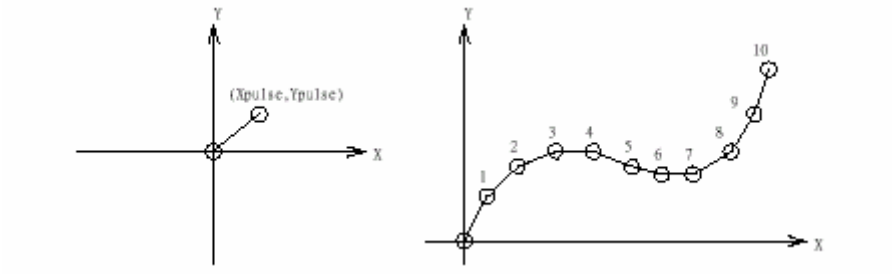
Remark:

This function can be applied on module: i8091.

■ i8091_INTP_PULSE

Description:

This function is used to move a short distance (interpolation short line) in the X-Y plane. This command provides a method for users to generate an arbitrary curve in a X-Y plane.



Syntax:

[C++]

void i8091_INTP_PULSE(unsigned char cardNo, int Xpulse, int Ypulse)

Parameter:

cardNO : [Input] The board number (0~19)

Xpulse : [Input] – 2047<= # Xpulse <=2047

Ypulse : [Input] – 2047<= # Ypulse <=2047

Return Value:

None

Example:

```
i8091_INTP_PULSE(CARD1, 20, 20);
i8091_INTP_PULSE(CARD1, 20, 13);
i8091_INTP_PULSE(CARD1, 20, 7);
i8091_INTP_PULSE(CARD1, 20, 0);
i8091_INTP_PULSE(CARD1,15, -5);
```

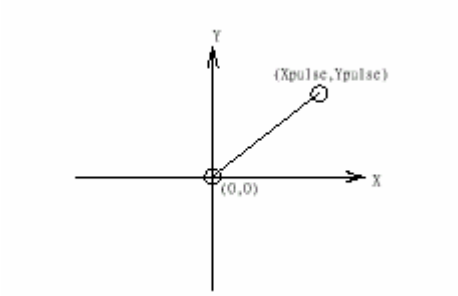
Remark:

This function can be applied on module: i8091.

■ i8091_INTP_LINE

Description:

This command will move a long distance (interpolation line) in the X-Y plane. The CPU on the I-8091 card will generate a trapezoidal speed profile of the X-axis and Y-axis, and then execute interpolation by way of a DDA chip.



Syntax:

[C++]

```
void i8091_INTP_LINE(unsigned char cardNo, int Xpulse, int Ypulse)
```

Parameter:

cardNO : [Input] The board number (0~19)

Xpulse : [Input] – 524287<= # Xpulse <=524287

Ypulse : [Input] – 524287<= # Ypulse <=524287

Return Value:

None

Example:

```
i8091_INTP_LINE(CARD1, 2000, -3000);
i8091_INTP_LINE(CARD1, -500, 200);
```

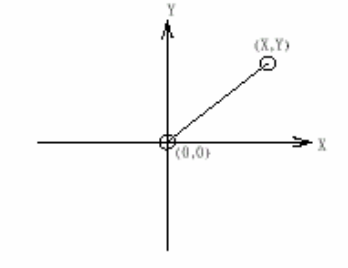
Remark:

This function can be applied on module: i8091.

■ i8091_INTP_LINE02

Description:

This function is used to move a long interpolation line in the X-Y plane. The host will automatically generate a trapezoidal speed profile of the X-axis and Y-axis via the state-machine-type calculation method. The i8091_INTP_LINE02 function only sets parameters into the driver. Users can directly utilize the do { } while (i8091_INTP_STOP( ) !=READY) method to apply the computing entity.



Syntax:

[C++]

```
void i8091_INTP_LINE02(unsigned char cardNo, long x, long y,
                       unsigned int speed, unsigned char acc_mode)
```

Parameter:

cardNO :	[Input] The board number (0~19)
x :	[Input] The end point of the line relates to the present position
y :	[Input] The end point of the line relates to the present position
speed :	[Input] 0~2040
acc_mode :	[Input] 0: enables the acceleration and deceleration profiles 1: disables the acceleration and deceleration profiles

Return Value:

None

Example:

```
i8091_INTP_LINE02(CARD1, 1000, 1000, 100, 0);
do { } while(i8091_INTP_STOP()!=READY) ;
//call state machine
```

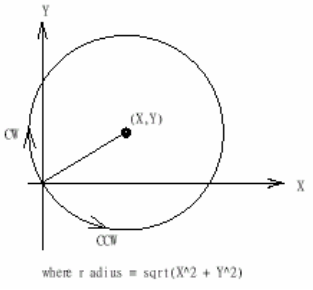
Remark:

This function can be applied on module: i8091.

■ i8091_INTP_CIRCLE02

Description:

This function is used to generate an interpolation circle in the X-Y plane. The host will automatically generate a trapezoidal speed profile of the X-axis and Y-axis via the state-machine-type calculation method. The i8091_INTP_CIRCLE02 function only sets parameters into the driver. Users can directly utilize the do { } while (i8091_INTP_STOP( ) !=READY) method to execute the computing entity.



Syntax:

[C++]

```
void i8091_INTP_CIRCLE02(unsigned char cardNo, long x, long y,
        unsigned chat dir, unsigned int speed, unsigned char acc_mode)
```

Parameter:

cardNO :	[Input] The board number (0~19)
x :	[Input] The center point of the circle relates to the present position
y :	[Input] The center point of the circle relates to the present position
dir :	[Input] The moving direction. (0:CW, 1:CCW)
speed :	[Input] 0~2040
acc_mode :	[Input] 0: enable acceleration and deceleration profile 1: disable acceleration and deceleration profile

Return Value:

None

Example:

```
i8091_INTP_CIRCLE02(CARD1, 2000, 2000, 100, 0);
do { } while(i8091_INTP_STOP()!=READY) ;
//call state machine
```

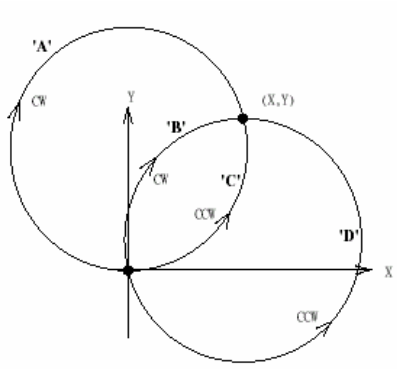
Remark:

This function can be applied on module: i8091.

■ i8091_INTP_ARC02

Description:

This command generates an interpolation arc in the X-Y plane. The host will automatically generate a trapezoidal speed profile of the X-axis and Y-axis via the state-machine-type calculation method. The i8091_INTP_ARC02() only sets parameters into the driver. Users can directly call the do { } while (i8091_INTP_STOP( ) !=READY) to execute the computing entity.



Syntax:

[C++]

void i8091_INTP_ARC02(unsigned char cardNo, long x, long y, long R,
unsigned char dir, unsigned int speed, unsigned char acc_mode)

Parameter:

cardNO : [Input] The board number (0~19)

x : [Input] The center point of the circle relates to the present position

y : [Input] The center point of the circle relates to the present position

R : [Input] The radius of arc

dir : [Input] The moving direction (0:CW, 1:CCW)

R	dir	path of curve
R>0	CW	'B'
R>0	CCW	'C'
R<0	CW	'A'
R<0	CCW	'D'

speed : [Input] 0~2040

acc_mode : [Input] 0: enables the acceleration and deceleration profiles
1: disables the acceleration and deceleration profiles

Return Value:

None

Example:

```
i8091_INTP_ ARC02(CARD1, 2000, -2000, 2000, CW, 100, 0);
do { } while(i8091_INTP_STOP()!=READY) ;
//call state machine
```

Remark:

This function can be applied on module: i8091.

Restriction:

$$-2^{32}+1 \le \#x \le 2^{32}-1$$

$$-2^{32}+1 \le \#y \le 2^{32}-1$$

$$-2^{32}+1 \le \#R \le 2^{32}-1$$

$$R \ge \frac{\sqrt{x^2+y^2}}{2}$$

■ i8091_INTP_STOP

Description:

This function must be applied when stopping the motor motion control for the above described 3 state-machine-type interpolation functions, to be precise the i8091_INTP_LINE02, i8091_INTP_CIRCLE02 and i8091_INTP_ARC02 functions. The state-machine-type interpolation functions only set parameters into the driver. The computing entity is in the i8091_INTP_STOP function. This function computes the interpolation profile. It will return READY(0) for interpolation command completion, and return BUSY(1) for when it is not yet completed.

Syntax:

[C++]

```
unsigned char i8091_INTP_STOP(void)
```

Parameter:

None

Return Value:

0: READY
1: BUSY

Example:

```
i8091_INTP_CIRCLE02(CARD1,2000,2000,100,0);
do { } while(i8091_INTP_STOP()!=READY) ;
//call state machine
```

Remark:

This function can be applied on module: i8091

■ i8091_LIMIT_X

Description:

This function is used to request the condition of the X-axis limit switches.

Syntax:

[C++]

```
unsigned char i8091_LIMIT_X(unsigned char cardNo)
```

Parameter:

cardNO : [Input] The board number (0~19)

Return Value:

a unsigned char value

MSB 7	6	5	4	3	2	1	0
/EMG	/FFFF	/FFEF	/LS14	xx	xx	/LS11	/ORG1

/ORG1 : original point switch of X-axis, low active.
/LS11, /LS14 : limit switches of X-axis, low active, which must be configured as Fig.(5). (Refer to I-8091 User's Manual)
/EMG : emergency switch, low active.
/FFEF : active low, FIFO is empty
/FFFF : active low, FIFO is full

Example:

```
unsigned char limit1;
limit1 = i8091_LIMIT_X(CARD1);
```

Remark:

This function can be applied on module: i8091.

■ i8091_LIMIT_Y

Description:

This function is used to request the condition of the Y-axis limit switches.

Syntax:

[C++]

```
unsigned char i8091_LIMIT_Y(unsigned char cardNo)
```

Parameter:

cardNO : [Input] The board number (0~19)

Return Value:

a unsigned char value

MSB 7	6	5	4	3	2	1	0
Ystop	Xstop	xx	/LS24	xx	xx	/LS21	/ORG2

/ORG2: original point switch of Y-axis, low active.

/LS21, /LS24: limit switches of Y-axis, low active, which must be configured as Fig.(6). (Refer to I-8091 User's Manual)

Xstop: 1:indicate X-axis is stop.

Ystop: 1:indicate Y-axis is stop.

Example:

```
unsigned char limit2;
limit2 = i8091_LIMIT_Y(CARD1);
```

Remark:

This function can be applied on module: i8091.

■ i8091_WAIT_X

Description:

This function is used to make the X-axis wait before going to the STOP state.

Syntax:

[C++]

```
void i8091_WAIT_X(unsigned char cardNo)
```

Parameter:

cardNO : [Input] The board number (0~19)

Return Value:

None

Example:

Remark:

This function can be applied on module: I8091.

■ i8091_WAIT_Y

Description:

This function is used to make the Y-axis wait before going to the STOP state.

Syntax:

[C++]

```
void i8091_WAIT_Y(unsigned char cardNo)
```

Parameter:

cardNO : [Input] The board number (0~19)

Return Value:

None

Example:

Remark:

This function can be applied on module: i8091.

■ i8091_IS_X_STOP

Description:

This function is used to check whether the X-axis is in the stop state or not.

Syntax:

[C++]

```
unsigned char i8091_IS_X_STOP(unsigned char cardNo)
```

Parameter:

cardNO : [Input] The board number (0~19)

Return Value:

0 (NO) : not yet stop

1 (YES) : stop

Example:

```
unsigned char data;
data= i8091_IS_X_STOP(CARD1);
```

Remark:

This function can be applied on module: i8091.

■ i8091_IS_Y_STOP

Description:

This function is used to check whether the Y-axis is in the stop state or not.

Syntax:

[C++]

```
unsigned char i8091_IS_Y_STOP(unsigned char cardNo)
```

Parameter:

cardNO : [Input] The board number (0~19)

Return Value:

0 (NO) : not yet stop

1 (YES) : stop

Example:

```
unsigned char data;
data= i8091_IS_Y_STOP(CARD1);
```

Remark:

This function can be applied on module: i8091.

2.10 Counter/Frequency Functions

■ i8080_InitDriver

Description:

The 'i8080_InitDriver ()' is used to initialize the 8080 module.

Syntax:

[C++]

```
int i8080_InitDriver(int Slot)
```

Parameter:

Slot : [Input] Specify the NuWa system slot (Slot 1)

Return Value:

0: OK

The others: Error codes, Refer to I8080.h

Example:

```
int slot=1;
int ErrorCode;
ErrorCode=i8080_InitDriver(slot);
//The I-8080 card is plugged in slot 1 of NuWa
//If the ErrorCode=0 expresses OK.
```

Remark:

This function can be applied on module: i8080.

■ i8080_AutoScan

Description:

This function is used to update counter values and to correctly transmit from the low 16-bit hardware counter to the high 16-bit software counter. The user's program must call this function from within their loop function or timer event. Refer to the "I-8080 Software User's Manual" Sec. 2.2 for more details.

Syntax:

[C++]

```
void i8080_AutoScan(void)
```

Parameter:

None

Return Value:

None

Example:

```
void CUpCount1Dlg::OnTimer(UINT nIDEvent)
{
        // TODO: Add your message handler code here and/or call default
        i8080_AutoScan();
}
```

Remark:

This function can be applied on module: i8080.

■ i8080_ReadDIXor

Description:

This function is used to read the signal status for the 8 channels in the i8080 after Xor Control processing is accomplished.

Syntax:

[C++]

```
int i8080_ReadDIXor(int Slot, short *Di)
```

Parameter:

Slot : [Input] Specify the NuWa system slot (Slot 1)

Di: [Output] An integer point, which allows the user to read the 8 channels of signal status after Xor control.

Bit0 of Di = A0 after Xor Control

Bit7 of Di = B3 after Xor Control

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1;
short j;
i8080_ReadDiXor(slot, &j);
//j=DiXor Value
```

Remark:

This function can be applied on module: i8080.

■ i8080_ReadDIXorLp

Description:

This function is used to read the signal status for the 8 channels in the i8080 after Xor Control and Low Pass Filter processing is accomplished.

Syntax:

[C++]

```
int i8080_ReadDIXorLp(int Slot, short *Di)
```

Parameter:

Slot : [Input] Specify the NuWa system slot (Slot 1)

Di: [Output] An integer point, which allows the user to read the 8 channels of signal status after the Xor Controls and Low Pass Filtering are accomplished.
Bit0 of Di = A0 after Xor Control & Low Pass Filter
Bit7 of Di = B3 after Xor Control & Low Pass Filter

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1;
short Di;
i8080_ ReadDIXorLp(slot, &Di);
// Di=DiXorLp Value
```

Remark:

This function can be applied on module: i8080.

■ i8080_ReadXorRegister

Description:

The function is used to read the 8 channel Xor registers in the i8080.

Syntax:

[C++]

```
int i8080_ReadXorRegister(int Slot, short *XorVal)
```

Parameter:

Slot : [Input] Specify the NuWa system slot (Slot 1)

XorVal: [Output] An integer point, which allows the user to save the 8 channels of Xor Control Registers.
Bit0 = A0's Xor Control Register
Bit7 = B3's Xor Control Register

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1;
short XorV;
i8080_ReadXorRegister(slot, &XorV);
// XorV=XorRegister Value
```

Remark:

This function can be applied on module: i8080.

■ i8080_SetXorRegister

Description:

This function is used to set the 8 channel Xor registers in the i8080 to 0 or 1. The setting value isn't saved to EEPROM, so all of the Xor registers will be default (0) after rebooting the i8080.

Syntax:

[C++]

```
int i8080_SetXorRegister(int Slot, short XorVal)
```

Parameter:

Slot : [Input] Specify the NuWa system slot (Slot 1).

XorVal: [Input] Allows the user to set the 8 channels in the Xor Control Registers.
Bit0 = A0's Xor Control Register
Bit7 = B3's Xor Control Register

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1;
short XorV;
XorV=0;
Set8080XorRegister(slot,XorV);
```

Remark:

This function can be applied on module: i8080.

■ i8080_EepWriteEnable

Description:

The function is used to set the EEPROM to the write-enable mode. Then the EEPROM will allow users to write information to the EEPROM.

Syntax:

[C++]

```
int i8080_EepWriteEnable(int Slot)
```

Parameter:

Slot : [Input] Specify the NuWa system slot (Slot 1)

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1;
i8080_EepWriteEnable(slot);
```

Remark:

This function can be applied on module: i8080.

■ i8080_EepWriteDisable

Description:

Set's the EEPROM to the write-protected mode. The EEPROM will then not allow users to write to the EEPROM.

Syntax:

[C++]

```
int i8080_EepWriteDisable(int Slot)
```

Parameter:

Slot : [Input] Specify the NuWa system slot (Slot 1)

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1;
i8080_EepWriteDisable (slot);
```

Remark:

This function can be applied on module: i8080.

■ i8080_EepWriteWord

Description:

This function is used to write 16-bit data to the EEPROM on the i8080. This 16-bit data is divided into high-byte (8 bits) and low-byte (8-bits).

Syntax:

[C++]

```
int i8080_EepWriteWord(int Slot, short Addr, short Hi, short Lo)
```

Parameter:

Slot :	[Input] Specify the NuWa system slot (Slot 1)
Addr :	[Input] 0 ~ 63 = Address Number
Hi :	[Input] High Byte Value
Lo :	[Input] Low Byte Value

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1, k;
TCHAR temp[80];
i8080_EepWriteEnable(slot);
for (int i=0; i<64; i++)
{
        k=EepWriteWord(slot,i,i+2,i+1);
        swprintf(temp, _T( “Write_%x Error=%x” ), i, k );
        if (k) MessageBox(NULL, temp, _T("Information"), MB_OK);
}
```

Remark:

This function can be applied on module: i8080.

■ i8080_EepReadWord

Description:

This function is used to read 16-bit data from the i8080 EEPROM. This 16-bit data is divided into high-bytes and low-bytes.

Syntax:

[C++]

```
int i8080_EepReadWord(int Slot, short *Addr, short *Hi, short *Lo)
```

Parameter:

Slot :	[Input] Specify the NuWa system slot (Slot 1)
Addr :	[Output] 0 ~ 63 = Address Number
Hi :	[Output] High Byte Value
Lo :	[Output] Low Byte Value

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1, i,j,k;
TCHAR temp[80];
for (i=0; i<64; i++)
{
        i8080_EepReadWord(slot,i,&j,&k);
        swprintf(temp, _T("Read ADDR=%x --> HIGH=%x, LOW=%x" ),i,j,k);
        MessageBox(NULL, temp, _T("Information"), MB_OK);
}
```

Remark:

This function can be applied on module: i8080.

■ i8080_ReadChannelMode

Description:

This function is used to read the operational mode, which includes the measured algorithm of the Frequency mode, Low Pass Filter status and the Xor Control from each channel.

Syntax:

[C++]

```
int i8080_ReadChannelMode(int Slot, int Channel, short *Mode)
```

Parameter:

Slot : [Input] Specify the NuWa system slot (Slot 1)

Channel : [Input] 0 ~ 7 = Channel Number(0=A0, 1=B0 ..., 6=A3, 7=B3)

Mode : [Output] Mode is defined as follows:

Bit1 and bit0 are used to select the operational mode.

Bit1 and bit0 = 00 --> Dir/Pulse Mode

= 01 --> Up/Down Mode

= 10 --> Frequency Mode

= 11 --> Up Count Mode

Bit3 and bit2 are valid for Frequency Mode Only.

Bit3 and bit2 = 00 --> Auto Select Frequency.

= 01 --> Select Low Frequency

= 10 --> Select High Frequency

= 11 --> Stop this channel (For all 4 operational mode)

Bit4 = 1/0 --> Enable/Disable Low Alarm (The function is reserved)

Bit5 = 1/0 --> Enable/Disable High Alarm (The function is reserved)

Bit6 = 1/0 --> Enable/Disable Low Pass filter

Bit7 = 0 --> Xor =0 --> non-invert this channel

Bit7 = 1 --> Xor =1 --> invert this channel

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1, j, k;
TCHAR temp[80];
for (j=0; j<8; j++)
{       //for human's double check
        i8080_ReadChannelMode(slot,j,&k);
        swprintf(temp, _T(" %02x"), k);
        MessageBox(NULL, temp, _T("Information"), MB_OK);
}
```

Remark:

This function can be applied on module: i8080.

■ i8080_SetChannelMode

Description:

Set's the operation mode, the measured algorithm of the Frequency mode, Low Pass Filter status and the Xor Control for each channel. The setting value isn't saved to the EEPROM, so all the values will be set at default after rebooting the i8080.

Syntax:

[C++]

```
int i8080_SetChannelMode(int Slot, int Channel, short Mode)
```

Parameter:

Slot : [Input] Specify the NuWa system slot (Slot 1)

Channel : [Input] 0 ~ 7 = Channel Number(0=A0, 1=B0 ..., 6=A3, 7=B3)

Mode : [Input] Mode is defined as follows:

Bit1 and bit0 are used to select the operational mode.

Bit1 and bit0 = 00 --> Dir/Pulse Mode

= 01 --> Up/Down Mode

= 10 --> Frequency Mode

= 11 --> Up Count Mode

Bit3 and bit2 are valid for Frequency Mode Only

Bit3 and bit2 =00 -->Auto Select Frequency

=01 -->Select Low Frequency

=10 -->Select High Frequency

=11 -->Stop this channel (For all 4 operational modes)

Bit4 = 1/0 --> Enable/Disable Low Alarm (The function is reserved)

Bit5 = 1/0 --> Enable/Disable High Alarm (The function is reserved)

Bit6 = 1/0 --> Enable/Disable Low Pass filter

Bit7 = 0 --> Xor=0 --> non-invert this channel

Bit7 = 1 --> Xor=1 --> invert this channel

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1;
i8080_SetChannelMode(slot,0,0x01);    //A0, Up_counting_0
i8080_SetChannelMode(slot,1,0x01);    //B0, Down_counting_0
//A0 & B0 must be same
i8080_SetChannelMode(slot,6,0x03);              //A3, Up_counting
i8080_Set8080ChannelMode(slot,7,0x03);          //B3, Up_counting
//Xor=0, isolated+active_Low
```

Remark:

This function can be applied on module: i8080.

■ i8080_ReadFreq

Description:

This function is used to read the Measured Frequency value in the Frequency mode.

Syntax:

[C++]

```
int i8080_ReadFreq(int Slot, int Channel, float *f)
```

Parameter:

Slot : [Input] Specify the NuWa system slot (Slot 1)

Channel : [Input] 0 ~ 7 = Channel Number(0=A0, 1=B0 ..., 6=A3, 7=B3)

f : [Output] The frequency Measured value.The variable must be a float point.

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1;
float f;
TCHAR temp[80];
i8080_ReadFreq(slot,0,&f);
swprintf(temp, _T("Channel_A0 : Frequency=%.1f"), f);
MessageBox(NULL, temp, _T("Information"), MB_OK);
```

Remark:

This function can be applied on module: i8080.

■ i8080_ReadCntUp

Description:

This function is used for the Up Counter mode to read the Up Counter value.

Syntax:

[C++]

```
int i8080_ReadCntUp(int Slot, int Channel, unsigned long *Cnt32U,
                                           unsigned int *Overflow)
```

Parameter:

Slot : [Input] Specify the NuWa system slot (Slot 1)

Channel : [Input] 0 ~ 7 = Channel Number(0=A0, 1=B0 ..., 6=A3, 7=B3)

Cnt32U : [Output] 32-bit Up Counter (High 16-bit software Counter and Low 16-bit hardware Counter)

Overflow : [Output] number of Overflow (16-bit)

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1;
unsigned long cnt32U;
unsigned short overflow;
TCHAR temp[80];
i8080_ReadCntUp(slot,0,&cnt32U,&overflow);
swprintf(temp, _T("Channel_A0 : overflow=%x, cnt32=%lx"),overflow,cnt32U);
MessageBox(NULL, temp, _T("Information"), MB_OK);
```

Remark:

This function can be applied on module: i8080.

■ i8080_ReadCntUpDown

Description:

This function is used for the Up/Down counter and for the Dir/Pulse mode in order to read the up-counting and down-counting values.

Syntax:

[C++]

```
int i8080_ReadCntUpDown(int Slot, int Channel, unsigned long *Cnt32U,
                        unsigned int *Overflow)
```

Parameter:

Slot :	[Input] Specify the NuWa system slot (Slot 1)
Channel :	[Input] 0 ~ 7 = Channel Number(0=A0, 1=B0 ..., 6=A3, 7=B3)
Cnt32U :	[Output] 32-bit Up/Down Counter MSB=0 --> Up Count MSB=1 --> Down Count
Overflow :	[Output] number of overflow (16 bit)

Return Value:

0: OK

The others: Error codes

Example:

```
//These variable as forward
i8080_ReadCntUpDown(slot,0,&cnt32U,&overflow);      //(A0,B0)
swprintf(temp,_T("Channel_A0_B0:overflow=%x,cnt32=%lx"),overflow,cnt32U);
MessageBox(NULL, temp, _T("Information"), MB_OK);
```

Remark:

This function can be applied on module: i8080.

■ i8080_SetLowPassUs

Description:

This function is used to set the parameters of Low Pass Filter in a specific channel in the specified slot. For more details relating to Low Pass Filter information, please refer to the I-8080 hardware manual.

Syntax:

[C++]

```
int i8080_SetLowPassUs(int Slot, int Channel, unsigned short Us)
```

Parameter:

Slot : [Input] Specify the NuWa system slot (Slot 1)

Channel : [Input] Low Pass filter channel number

Channel0 is valid for A0, B0

Channel1 is valid for A1, B1

Channel2 is valid for A2, B2, A3 and B3

Us : [Input] 1 to 0x7fff (1~32767), unit=0.000001 second (μS)

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1;
i8080_SetLowPassUs(slot,0,1000U);     //valid for A0 & B0
```

Remark:

This function can be applied on module: i8080.

■ i8080_ReadLowPassUs

Description:

To read Low-Pass Filter information. For more details relating to Low Pass Filter information, please refer to the I-8080 hardware manual.

Syntax:

[C++]

```
int i8080_ReadLowPassUs(int Slot, int Channel, unsigned short *Us)
```

Parameter:

Slot : [Input] Specify the NuWa system slot (Slot 1)

Channel : [Input] Low Pass filter channel number

Channel0 is valid for A0, B0

Channel1 is valid for A1, B1

Channel2 is valid for A2, B2, A3 and B3

Us : [Output] 1 to 0x7fff (1~32767), unit=0.000001 second (μS)

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1;
unsigned short MinWidth;
i8080_SetLowPassUs(slot, 0,1000U);     //valid for A0 & B0
i8080_ReadLowPassUs(slot, 0, &MinWidth);
```

Remark:

This function can be applied on module: i8080.

■ i8080_ClrCnt

Description:

This function is used to clear the specific channel counter value in a specific slot. All four-operation modes are supported in this function. (Dir/ Pulse, Up/Down, Up Counter, and Frequency mode)

Syntax:

[C++]

```
int i8080_ClrCnt(int Slot, int Channel)
```

Parameter:

Slot : [Input] Specify the NuWa system slot (Slot 1)

Channel : [Input] 0 ~ 7 = Channel Number(0=A0, 1=B0 ..., 6=A3, 7=B3)

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1;
i8080_ClrCnt(slot,0);    //clear   counter A0
i8080_ClrCnt(slot,1);    //clear   counter B0
i8080_ClrCnt(slot,6);    //clear   counter A3
i8080_ClrCnt(slot,7);    //clear   counter B3
```

Remark:

This function can be applied on module: i8080.

■ i8080_ReadFreqConfiguration

Description:

This function is used to read the configuration data in the frequency mode.

Syntax:

[C++]

```
int i8080_ReadFreqConfiguration(int Slot, short *AutoTT , short *AutoVV,
        short *LowTT, short *LowVV, short *HighTT, short *HighVV)
```

Parameter:

Slot :	[Input] Specify the NuWa system slot (Slot 1)
AutoTT:	[Output] Sampling time period for Auto mode, unit= millisecond
AutoVV:	[Output] Samples to start frequency update (Auto mode)
LowTT:	[Output] Sampling time period for Low Frequency mode, unit= millisecond
LowVV:	[Output] Samples to start frequency update (Low Frequency Mode)
HighTT:	[Output] Sampling time period for High Frequency mode, unit= millisecond
HighVV:	[Output] Samples to start frequency update (High Frequency Mode)

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1;
short AutoTT, AutoVV, LowTT, LowVV, HighTT, HighVV;
TCHAR temp[80];
i8080_ReadFreqConfiguration(slot,&AutoTT,&AutoVV,&LowTT,&LowVV,&HighTT,&HighVV);
//Current Configuration Data for Frequency Measurement Algorithm
swprintf(temp, _T("AutoTT=%d mS, AutoVV=%d counts"),AutoTT,AutoVV);
MessageBox(NULL, temp, _T("Information"), MB_OK);
swprintf(temp,_T("LowTT=%d mS, LowVV=%d counts"),LowTT,LowVV);
MessageBox(NULL, temp, _T("Information"), MB_OK);
swprintf(temp,_T("\n\rHighTT=%d mS, HighVV=%d counts"),HighTT,HighVV);
MessageBox(NULL, temp, _T("Information"), MB_OK);
```

Remark:

This function can be applied on module: i8080.

■ i8080_SetFreqConfiguration

Description:

This function is used to set the configuration data used by frequency measurement algorithms. The setting value isn't saved to the EEPROM, so the all values will be set to default after rebooting the i8080.

Syntax:

[C++]

```
int i8080_SetFreqConfiguration(int Slot, short AutoTT , short AutoVV,
        short LowTT, short LowVV, short HighTT, short HighVV)
```

Parameter:

Slot :	[Input] Specify the NuWa system slot (Slot 1)
AutoTT:	[Input] Sampling time period for Auto mode, unit= millisecond
AutoVV:	[Input] Samples to start frequency update (Auto mode)
LowTT:	[Input] Sampling time period for Low Frequency mode, unit= millisecond
LowVV:	[Input] Samples to start frequency update (Low Frequency Mode)
HighTT:	[Input] Sampling time period for High Frequency mode, unit= millisecond
HighVV:	[Input] Samples to start frequency update (High Frequency Mode)

Return Value:

0: OK

The others: Error codes

Example:

```
int slot=1;
LowTT=10000;    //set time period to 10 seconds for more low frequency
HighTT=50;      //set time period to 50 mS for more high update rate
i8080_SetFreqConfiguration(slot,AutoTT,AutoVV,LowTT,LowVV,HighTT,HighVV);
```

Remark:

This function can be applied on module: i8080.

APPENDIX A Error Code

Error Code

NoError	0	Functions work normally.
FunctionError	1	Call wrong function error.
PortError	2	Use wrong COM Port error.
BaudRateError	3	Baud rate error.
DataError	4	Data Bit error.
StopError	5	Stop Bit error.
ParityError	6	Parity Bit error.
CheckSumError	7	CheckSum mechanism error.
ComPortNotOpen	8	COM is not open error.
SendThreadCreateError	9	Send thread create error.
SendCmdError	10	Send command error.
ReadComStatusError	11	Read COM Port status error.
ResultStrCheckError	12	Result string check error.
CmdError	13	Command error.
TimeOut	15	TimeOut error.
ModuleIdError	17	Module ID error.
AdChannelError	18	Channel number error.
UnderInputRange	19	Under input range error.
ExceedInputRange	20	Exceed input range error.
InvalidateCounterNo	21	Invalidate counter number error.
InvalidateCounterValue	22	Invalidate counter value error
InvalidateGateMode	23	Invalidate gate mode error.
InvalidateChannelNo	24	Invalidate channel No error.
ComPortInUse	25	COM Port is in use error.

PROBLEMS REPORT

Technical support is available at no charge. The best way to report problems is send electronic mail to

service@icpdas.com

When reporting problems, please include the following information:

1. Is the problem reproducible? If so, how?
2. What kind and version of Platform are you using? For example, Windows 3.1, Windows for Workgroups, Windows NT 4.0, etc.
3. What kinds of our products are you using? Please see the product's manual.
4. If a dialog box with an error message was displayed, please include the full text of the dialog box, including the text in the title bar.
5. If the problem involves other programs or hardware devices, what devices or version of the failing programs do you use?
6. Other comments relative to this problem or any suggestions will be welcomed.

After we had received your comments, we will take about two business days to test the problems that you have reported. And then We will reply it as soon as possible to you. Please check that we had received your comments? And please keep in contact with us.

E-mail: service@icpdas.com
Web site: http://www.icpdas.com.tw