WinCon-8000 SDK User Guide

for Visual studio 2003 C#.NET, and VB.NET

(Version 1.0)

Dynamic Link Library (DLL) for WinCon-8000
Using I-7000/I-87K/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 WinconSDK_DLL and Wincon_DLL user's manual. ICPDAS provides two DLL files, namely the WinconSDK_DLL and the Wincon_DLL, for the I-8000 series modules which are used in the Wincon-8000 Embedded Controller. The **WinconSDK_DLL** has all the essential DLL functions designed for the I-8000 series modules for Microsoft WinCE.Net platform. It can be applied on 2003 Visual studio C#.NET, and VB.NET on WinCE 4.1.Net, and even on the newer platforms. Users can easily develop WinCE.NET applications on WinCon-8000 by using this toolkit.

The various functions in **WinconSDK_DLL** are divided into the following sub-group functions for easy use in different applications. The main functions of the WinCon-8000 embedded controller are depicted in figure 1.2, which include **VGA, USB, mouse, keyboard, compact flash, series, Ethernet and an I/O interface**.

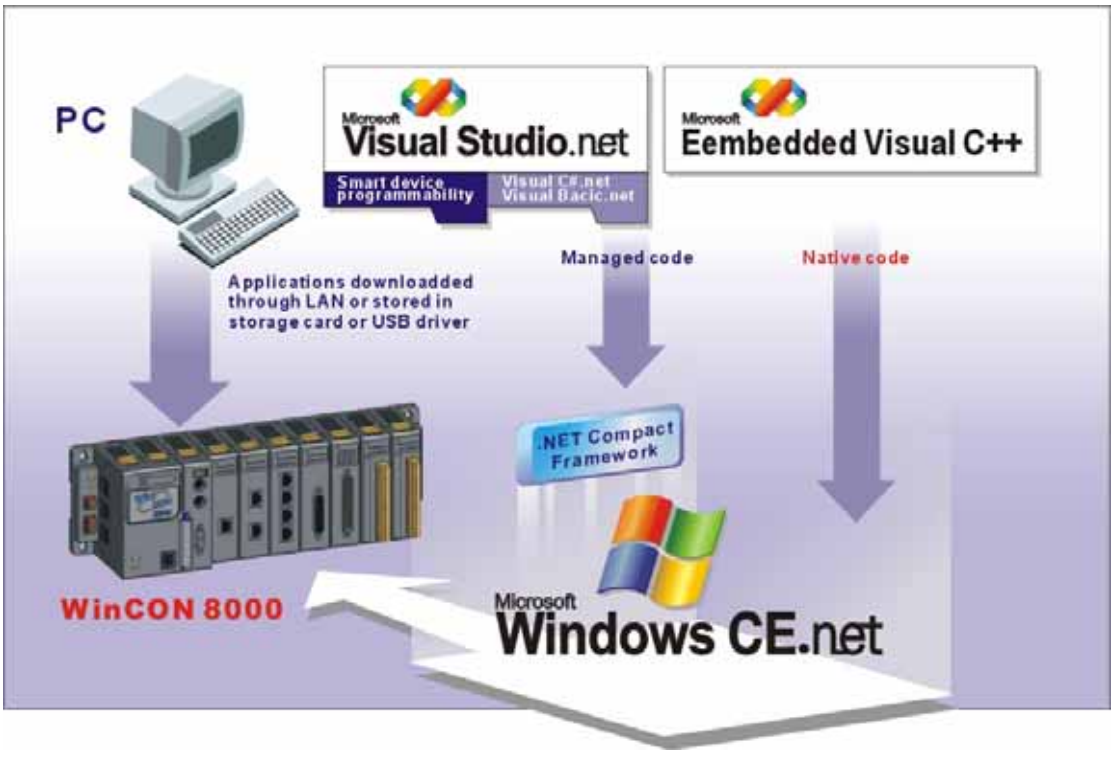


Fig. 1-1

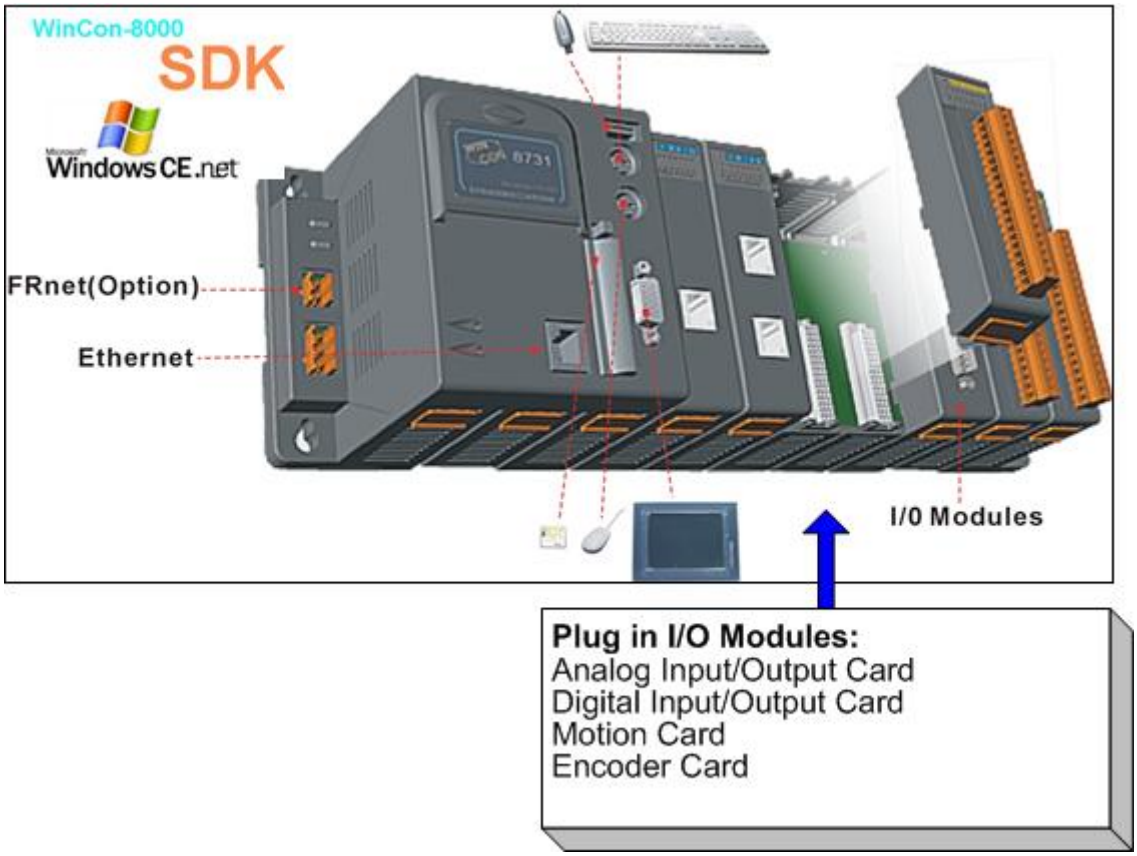


Fig. 1-2

1.1 What is new

==========

Sep 30, 2004 v1.0.5.0 Wincon.dll
v1.0.4.0 DCONCE.dll
v1.0.1.0 Serial.dll

Add : Add WatchDog functions in Wincon.dll.
Add I-87040,I-87041,I-87082 modules in Wincon.dll/DCONCE.dll.
Add Serial.dll COM port I/O fuction.
Add COM port Space Parity and Mark Parity check.

Fix : Fix i8091_INTP_STOP() function in Wincon.dll.
Fix i87K_ReadModuleConfig() function in Wincon.dll.
Fix DigitalInLatch() function of I-87051 in Wincon.dll/DCONCE.dll.
Fix ReadLevelVolt() function of I-7080 in DCONCE.dll.

2. WinconSDK.DLL

In this section we will focus on the description and application example of WinCon DLL functions. The functions of WinconSDK.DLL 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 Wincon-8000 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 first function indicates the syntax for Visual Studio VB.NET and C#.NET. However, 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

For the application of visual studio VB.NET and C#.NET, the namespace of the WinCon.DLL is **Wcon** and should not be changed whilst developing an application. However, the ModuleName of the WinCon.DLL should be replaced by the practical module name, which will actually be applied in the control or measurement system (WinCon-8000). For example: I8017h, I8024, I8057, I8053…etc. For more detailed information on how to use this driver, please refer to section 4.2.3, and it looks just like the application on the ActiveX component.

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 Wincon-8000 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:

[Visual Basic, C#]
void Wcon.System.ChangeSlotTo87K(byte slotNo)

Parameter:

slotNo : [Input] Specify the slot number.

Return Value:

None

Example:

[C#]

```
byte slot  = 1;
Wcon.System.ChangeSlotTo87K(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.System.ChangeSlotTo87K(slot)
```

// The first slot is specified as COM1 port in Wincon-8000.

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 WinCon system. This function performs a supporting task in the collection of information relating to system hardware configurations.

Syntax:

[Visual Basic, C#]

```
short Wcon.System.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:

[C#]

```
int slot  = 1;
short moduleType;

moduleType = Wcon.System.GetModuleType(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
```

```
Dim moduleType As Short
moduleType = Wcon.System.GetModuleType(slot)
```

// The i-8057 card is plugged in slot 1 of Wincon-8000 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 Wincon-8000 system. This function supports the collection of system hardware configurations.

Syntax:

[Visual Basic, C#]

```
short Wcon.System.GetNameOfModule(int slot, ref string 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:

[C#]

```
int slot=1,moduleID;
string moduleName="";
moduleID=Wcon.System.GetNameOfModule(slot, ref moduleName);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim moduleID As Integer
Dim moduleName As String = ""
moduleID = Wcon.System.GetNameOfModule(slot, moduleName)
```

```
// The I-8057 card plugged in slot 1 of Wincon-8000
// 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(uint32) value. Therefore, the time will wrap around to zero if the system is run continuously for 49.7 days.

Syntax:

[Visual Basic, C#]
uint Wcon.System.GetTimeTicks(void)

Parameter:

None

Return Value:

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

Example:

[C#]

```
uint time1;

time1 = Wcon.System.GetTimeTicks();
```

[Visual Basic]

```
Dim time1 As UInt32
time1 = Wcon.System.GetTimeTicks()
```

// Returned Value: time1=94367

Remark:

■ GetSlotAddr

Description:

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

Syntax:

[Visual Basic, C#]
uint Wcon.System.GetSlotAddr(int slot)

Parameter:

slot : [Input] Specify the slot number.

Return Value:

Slot base address. The WinCon-8000 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:

[C#]

```
int slot  = 1;
uint slotAddr;
slotAddr = Wcon.System.GetSlotAddr(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim slotAddr As UInt32
slotAddr = Wcon.System.GetSlotAddr(slot)
// Returned Value: slotAddr= 0x8bb00100
```

Remark:

■ GetSystemSerialNumber

Description:

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

Syntax:

[Visual Basic, C#]
short Wcon.System.GetSystemSerialNumber(out ulong sn)

Parameter:

sn: [Output] An unsigned long variable will retrieve the serial ID number.

Return Value:

0: indicates success.

1: indicates failure.

Example:

[C#]

```
ulong serialNo;
Wcon.System.GetSystemSerialNumber(out serialNo);
```

[Visual Basic]

```
Dim serialNo As UInt64
Wcon.System.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 WinCon SDK library files.

Syntax:

[Visual Basic, C#]
void Wcon.System.GetSDKversion(ref string string1)

Parameter:

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

Return Value:

None

Example:

[C#]

```
string sdkVersion="";
Wcon.System.GetSDKversion(ref sdkVersion);
```

[Visual Basic]

```
Dim sdkVersion As String = ""
Wcon.System.GetSDKversion(sdkVersion)
// Returned value: sdkVersion="WinCon SDK 1.0.0"
```

Remark:

■ GetOSversion

Description:

This function retrieves the version number of the embedded OS.

Syntax:

[Visual Basic, C#]

```
void Wcon.System.GetOSversion(ref string string1)
```

Parameter:

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

Return Value:

None

Example:

[C#]

```
string osVersion="";
Wcon.System. GetOSversion(ref osVersion);
```

[Visual Basic]

```
Dim osVersion As String = ""
Wcon.System.GetOSversion(osVersion)
```

```
// Returned value: osVersion="Windows CE .NET 4.1 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:

[Visual Basic, C#]
void Wcon.ModuleName.DO_8(int slot, byte cdata)

Parameter:

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

cdata : [Input] output data.

Return Value:

None

Example:

[C#]

```
int slot=1;
byte data = 3 ;
Wcon.i8064. DO_8(slot, data);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim O_DATA As Byte = 3
Wcon.i8064.DO_8(slot, O_DATA)
```

// The I-8064 card is plugged in slot 1 of Wincon-8000 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:

[Visual Basic, C#]
void Wcon.ModuleName.DO_16(int slot, ushort cdata)

Parameter:

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

cdata : [Input] output data.

Return Value:

None

Example:

[C#]

```
int slot=1;
ushort  data = 3 ;
Wcon.i8057.DO_16(slot, data);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim O_DATA As UInt16 = Convert.ToUInt16(3)
Wcon.i8057.DO_16(slot, O_DATA)
```

// The I-8057 card is plugged in slot 1 of Wincon-8000 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:

[Visual Basic, C#]
void Wcon.ModuleName.DO_32(int slot, uint cdata)

Parameter:

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

cdata : [Input] output data.

Return Value:

None

Example:

[C#]

```
int slot=1;
uint  data = 3 ;
Wcon.i8041.DO_32(slot, data);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim O_DATA As UInt32 = Convert.ToUInt32(3)
Wcon.i8041.DO_32(slot, O_DATA)
```

// The I-8041 card is plugged in slot 1 of Wincon-8000 and can turn on channel 0 and 1.

Remark:

This function can be applied on module: i8041.

■ DO_8_ Bit

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:

[Visual Basic, C#]

void Wcon.ModuleName.DO_8_Bit(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:

[C#]

```
int slot=1;
int channel=3;
Boolean data=true;
Wcon.i8060.DO_8_Bit(slot,channel,data);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim channel As Integer = 3
Dim O_DATA As Boolean = True
Wcon.i8060.DO_8_Bit(slot, channel, O_DATA)
```

// The I-8060 card is plugged in slot 1 of Wincon-8000 turns on channel 3.

Remark:

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

■ DO_16_ Bit

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:

[Visual Basic, C#]

void Wcon.ModuleName.DO_16_Bit(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:

[C#]

```
int slot=1;
int channel=3;
Boolean data=true;
Wcon.i8057.DO_16_Bit(slot,channel,data);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim channel As Integer = 3
Dim O_DATA As Boolean = True
Wcon.i8057.DO_16_Bit(slot, channel, O_DATA)
```

// The I-8057 card is plugged in slot 1 of Wincon-8000 turns on channel 3.

Remark:

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

■ DO_32_ Bit

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:

[Visual Basic, C#]
void Wcon.ModuleName.DO_32_Bit(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:

[C#]

```
int slot=1;
int channel=3;
Boolean data=true;
Wcon.i8041.DO_32_Bit(slot,channel,data);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim channel As Integer = 3
Dim O_DATA As Boolean = True
Wcon.i8041.DO_32_Bit(slot, channel, O_DATA)
```

// The I-8041 card is plugged in slot 1 of Wincon-8000 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:

[Visual Basic, C#]
byte Wcon.ModuleName.DI_8(int slot)

Parameter:

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

Return Value:

Input data

Example:

[C#]

```
int slot=1;
byte  data;
data= Wcon.i8058.DI_8(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim I_DATA As Byte
I_DATA = Wcon.i8058.DI_8(slot)
// The I-8058 card is plugged in slot 1 of Wincon-8000 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:

[Visual Basic, C#]
ushort Wcon.ModuleName.DI_16(int slot)

Parameter:

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

Return Value:

Input data

Example:

[C#]

```
int slot=1;
ushort  data;
data= Wcon.i8053.DI_16(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim I_DATA As UInt16
I_DATA = Wcon.i8053.DI_16(slot)
// The I-8053 card is plugged in slot 1 of Wincon-8000 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:

[Visual Basic, C#]
uint32 Wcon.ModuleName.DI_32(int slot)

Parameter:

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

Return Value:

Input data

Example:

[C#]

```
int slot=1;
uint data;
data= Wcon.i8040.DI_32(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim I_DATA As UInt32
I_DATA = Wcon.i8040.DI_32(slot)
// The I-8040 card plugged is in slot 1 of Wincon-8000 and has inputs in
// channels 0 and 1.
// Returned value: data=0xfffffffC
```

Remark:

This function can be applied on module: i8040.

■ DI_8_Bit

Description:

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

Syntax:

[Visual Basic, C#]
bool Wcon.ModuleName.DI_8_Bit(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:

[C#]

```
int slot=1;
int channel=3;
Boolean data;
data= Wcon.i8058.DI_8_Bit(slot, channel);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim channel As Integer = 3
Dim I_DATA As Boolean
I_DATA = Wcon.i8058.DI_8_Bit(slot, channel)
```

// The I-8058 card plugged is in slot 1 of Wincon-8000 and has inputs in channel 3.

// Returned value: data=true

Remark:

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

■ DI_16_Bit

Description:

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

Syntax:

[Visual Basic, C#]
bool Wcon.ModuleName.DI_16_Bit(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:

[C#]

```
int slot=1;
int channel=3;
Boolean data;
data= Wcon.i8051.DI_16_Bit(slot, channel);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim channel As Integer = 3
Dim I_DATA As Boolean
I_DATA = Wcon.i8051.DI_16_Bit(slot, channel)
```

// The I-8051 card is plugged in slot 1 of Wincon-8000 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:

[Visual Basic, C#]
bool Wcon.ModuleName.DI_32_Bit(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:

[C#]

```
int slot=1;
int channel=3;
Boolean data;
data= Wcon.i8040.DI_32_Bit(slot, channel);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim channel As Integer = 3
Dim I_DATA As Boolean
I_DATA = Wcon.i8040.DI_32_Bit(slot, channel)
```

// The I-8040 card is plugged in slot 1 of Wincon-8000 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:

[Visual Basic, C#]
void Wcon.ModuleName.DIO_DO_8(int slot, byte data)

Parameter:

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

data : [Input] output data.

Return Value:

None

Example:

[C#]

```
int slot=1;
byte data = 3 ;
Wcon.i8054.DIO_DO_8(slot, data);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim I_DATA As Byte = 3
Wcon.i8054.DIO_DO_8(slot, I_DATA)
```

// The I-8054 card is plugged in slot 1 of Wincon-8000 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:

[Visual Basic, C#]
void Wcon.ModuleName.DIO_DO_16(int slot, ushort data)

Parameter:

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

data : [Input] output data.

Return Value:

None

Example:

[C#]

```
int slot=1;
ushort  data = 3 ;
Wcon.i8050.DIO_DO_16(slot, data);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim O_DATA As UInt16 = Convert.ToUInt16(3)
Wcon.i8050.DIO_DO_16(slot, O_DATA)
```

// The I-8050 card is plugged in slot 1 of Wincon-8000 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_Bit

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:

[Visual Basic, C#]
void Wcon.ModuleName.DIO_DO_8_Bit(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:

[C#]

```
int slot=1;
int channel=3;
Boolean data=true;
Wcon.i8054.DIO_DO_8_Bit(slot, channel, data);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim channel As Integer = 3
Dim O_DATA As Boolean = True
Wcon.i8054.DIO_DO_8_Bit(slot, channel, O_DATA)
```

// The I-8054 card is plugged in slot 1 of Wincon-8000 and can turn on channel 3.

Remark:

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

■ DIO_DO_16_Bit

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:

[Visual Basic, C#]

void Wcon.ModuleName.DIO_DO_16_Bit(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:

[C#]

```
int slot=1;
int channel=3;
Boolean data=true;
Wcon.i8042.DIO_DO_16_Bit(slot, channel, data);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim channel As Integer = 3
Dim O_DATA As Boolean = True
Wcon.i8042.DIO_DO_16_Bit(slot, channel, O_DATA)
```

// The I-8042 card is plugged in slot 1 of Wincon-8000 and can turn on the channel 3.

Remark:

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

■ DO_8_RB (DigitalOutReadBack)

Description:

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

Syntax:

[Visual Basic, C#]
byte Wcon.ModuleName.DigitalOutReadBack(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:

[C#]

```
int slot=1;
byte data;
data=Wcon.i8054.DigitalOutReadBack(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim I_DATA As Byte
I_DATA = Wcon.i8054.DigitalOutReadBack(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 (DigitalOutReadBack)

Description:

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

Syntax:

[Visual Basic, C#]
ushort Wcon.ModuleName.DigitalOutReadBack(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:

[C#]

```
int slot=1;
ushort data ;
data=Wcon.i8057.DigitalOutReadBack(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim I_DATA As UInt16
I_DATA = Wcon.i8057.DigitalOutReadBack(slot)
// The data read back has a digital output value=0x03
```

Remark:

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

■ DO_32_RB (DigitalOutReadBack)

Description:

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

Syntax:

[Visual Basic, C#]
uint Wcon.ModuleName.DigitalOutReadBack(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:

[C#]

```
int slot=1;
uint data ;
data=Wcon.i8041.DigitalOutReadBack(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim I_DATA As UInt32
I_DATA = Wcon.i8041.DigitalOutReadBack(slot)
// The data read back has a digital output value=0x03
```

Remark:

This function can be applied on modules: i8041.

■ DIO_DO_8_RB (DigitalOutReadBack)

Description:

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

Syntax:

[Visual Basic, C#]
byte Wcon.ModuleName.DigitalOutReadBack(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:

[C#]

```
int slot=1;
byte data = 3 ;
Wcon.i8054.DIO_DO_8(slot, data);
byte data_bk  ;
data_bk=Wcon.i8054.DigitalOutReadBack(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim I_DATA As Byte = 3
Wcon.i8054.DIO_DO_8(slot, I_DATA)
Dim DATA_bk As Byte
DATA_bk = Wcon.i8054.DigitalOutReadBack(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 (DigitalOutReadBack)

Description:

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

Syntax:

[Visual Basic, C#]

```
ushort Wcon.ModuleName.DigitalOutReadBack(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:

[C#]

```
int slot=1;
ushort  data = 3 ;
Wcon.i8050.DIO_DO_16(slot, data);
ushort  data_bk ;
data_bk=Wcon.i8050.DigitalOutReadBack(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim O_DATA As UInt16 = Convert.ToUInt16(3)
Wcon.i8050.DIO_DO_16(slot, O_DATA)
Dim DATA_bk As Byte
DATA_bk = Wcon.i8054.DigitalOutReadBack(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 WatchDog and Reflash WatchDog and set timer interval in millisecond.

Syntax:

[Visual Basic, C#]
void Wcon.WDT.EnableWDT(uint msecond)

Parameter:

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

Return Value:

None

Example:

[C#]

```
Wcon.WDT.EnableWDT(2000);
```

[Visual Basic]

```
Wcon.WDT.EnableWDT(Convert.ToUInt32(2000))
```

Remark:

■ DisableWDT

Description:

This function is used to disable the watch dog timer.

Syntax:

[Visual Basic, C#]
void Wcon.WDT.DisableWDT(void)

Parameter:

None

Return Value:

None

Example:

[C#]

```
Wcon.WDT.DisableWDT();
```

[Visual Basic]

```
Wcon.WDT.DisableWDT()
```

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:

[Visual Basic, C#]
int Wcon.WDT.WatchDogSWEven(void)

Parameter:

None

Return Value:

1: indicates system has been reset with the watch dog timer.
0: indicates the system has not been reset with the watch dog timer.

Example:

[C#]

```
int WDT_flag ;
WDT_flag = Wcon.WDT.WatchDogSWEven();
```

[Visual Basic]

```
Dim WDT_flag As Int16
WDT_flag = Wcon.WDT.WatchDogSWEven()
```

Remark:

■ ClearWDTSWEven

Description:

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

Syntax:

[Visual Basic, C#]
void Wcon.WDT. ClearWDTSWEven (void)

Parameter:

None

Return Value:

None

Example:

[C#]

```
Wcon.WDT.ClearWDTSWEven();
```

[Visual Basic]

```
Wcon.WDT.ClearWDTSWEven()
```

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 WinCon-8000 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:

[Visual Basic, C#]

```
byte Wcon.EEPROM.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:

[C#]

```
int block, offset;
byte data;
block=1; offset=3;
data= Wcon.EEPROM.ReadEEP(block, offset);
```

[Visual Basic]

```
Dim block, offset As Integer
Dim data As Byte
block = 1: offset = 3
data = Wcon.EEPROM.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 WinCon-8000 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:

[Visual Basic, C#]

```
void Wcon.EEPROM.WriteEEP(int block, int offset, byte 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:

[C#]

```
int block, offset;
byte data= 10 ;
block=1;   offset=3;
Wcon.EEPROM.WriteEEP(block, offset,data);
```

[Visual Basic]

```
Dim block, offset As Integer
Dim data As Byte = 10
block = 1 : offset = 3
Wcon.EEPROM.WriteEEP(block, offset, data)
```

// Writes a 10 value output to the EEPROM (block & offset) location

Remark:

2.6 Analog Input Functions

■ i8017H.Initial

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:

[Visual Basic, C#]
void Wcon.ModuleName.Initial(int slot)

Parameter:

slot : [Input] specified slot of the Wincon-8000 system (Range: 1 to 7)

Return Value:

None

Example:

[C#]

```
int slot=1;
Wcon.i8017H.Initial(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8017H.Initial(slot)
```

// The I-8017H card is plugged in slot 1 of Wincon-8000 and initializes the module
// I-8017H.

Remark:

This function can be applied on module: i8017H.

■ i8017H.Set_LED

Description:

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

Syntax:

[Visual Basic, C#]

void Wcon.ModuleName.Set_LED(int slot, unsigned short led)

Parameter:

slot : [Input] specified slot of the Wincon-8000 system (Range: 1 to 7)

led : [Input] range from 0 to 0xffff

Return Value:

None

Example:

[C#]

```
int slot=1;
Wcon.i8017H.Initial(slot);
ushort led=0x0001;
Wcon.i8017H.Set_LED(slot, led);
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8017H.Initial(slot)
Wcon.i8017H.Set_LED(slot, Convert.ToUInt16(1))
```

// The LED will have a L-LED light on channel 0 on the I-8017H card which is plugged in slot 1 on the Wincon-8000.

Remark:

This function can be applied on module: i8017H.

■ i8017H.Set_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:

[Visual Basic, C#]

void Wcon.ModuleName.Set_Channel_Gain_Mode(int slot, int ch,
short gain, short mode)

Parameter:

slot : [Input] Specify the slot in the Wincon-8000 system (Range: 1 to 7)

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:

[C#]

```
int slot=1,ch=0;
Wcon.i8017H.Initial(slot);
short gain=0;
Wcon.i8017H.Set_Channel_Gain_Mode(slot,ch,gain,0);
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8017H.Initial(slot)
Dim ch As Integer = 0
Dim gain As Short = 0
Wcon.i8017H.Set_Channel_Gain_Mode(slot, ch, gain, 0)
```

// The I-8017H card is plugged in slot 1 of Wincon-8000, 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.

■ 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:

[Visual Basic, C#]
float Wcon.ModuleName.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:

[C#]

```
int slot=1,ch=0;
Wcon.i8017H.Initial(slot);
short gain=0;
Wcon.i8017H.Set_Channel_Gain_Mode(slot,ch,gain,0);
float Data;
Data=Wcon.i8017H.Get_AD_FValue(gain);
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8017H.Initial(slot)
Dim ch As Integer = 0

Dim gain As Short = 0
Wcon.i8017H.Set_Channel_Gain_Mode(slot, ch, gain, 0)
Dim data As Single
```

```
data = Wcon.i8017H.Get_AD_FValue(gain)
// The I-8017H card is plugged into slot 1 of Wincon-8000 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.

■ 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:

[Visual Basic, C#]

```
float Wcon.ModuleName.Get_AD_IValue(void)
```

Parameter:

None

Return Value:

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

Example:

[C#]

```
int slot=1,ch=0;
Wcon.i8017H.Initial(slot);
short gain=4;
Wcon.i8017H.Set_Channel_Gain_Mode(slot,ch,gain,0);
float Data;
Data=Wcon.i8017H.Get_AD_IValue();
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8017H.Initial(slot)
Dim ch As Integer = 0
Dim data As Single
Dim gain As Short = 4
Wcon.i8017H.Set_Channel_Gain_Mode(slot, ch, gain, 0)
data = Wcon.i8017H.Get_AD_IValue()
```

// The I-8017H card is plugged into slot 1 of Wincon-8000, 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.

■ 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:

[Visual Basic, C#]
short Wcon.ModuleName.Get_AD_HValue(void)

Parameter:

None

Return Value:

Return (Hex): The voltage analog input value.

Example:

[C#]

```
int slot=1,ch=0;
Wcon.i8017H.Initial(slot);
short gain=4;
Wcon.i8017H.Set_Channel_Gain_Mode(slot,ch,gain,0);
short Data;
Data=Wcon.i8017H.Get_AD_HValue();
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8017H.Initial(slot)
Dim ch As Integer = 0
Dim data As Short
Dim gain As Short = 4
Wcon.i8017H.Set_Channel_Gain_Mode(slot, ch, gain, 0)
data = Wcon.i8017H.Get_AD_HValue()
```

// The I-8017H card is plugged in slot 1 of Wincon-8000, 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.

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:

[Visual Basic, C#]

```
void Wcon.ModuleName.Initial(int slot)
```

Parameter:

slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

Return Value:

None

Example:

[C#]

```
int slot=1;
Wcon.i8024.Initial(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8024.Initial(slot)
```

// The I-8024 card is plugged into slot 1 of Wincon-8000 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 Wincon-8000 system.

Syntax:

[Visual Basic, C#]

```
void Wcon.ModuleName.VoltageOut(int slot, int ch, float data)
```

Parameter:

slot : [Input] Specified the Wincon-8000 system slot (Range: 1 to 7)

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

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

Return Value:

None

Example:

[C#]

```
int slot=1, ch=0;
Wcon.i8024.Initial(slot);
float data=3.0f;
Wcon.i8024.VoltageOut(slot,ch,data);
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8024.Initial(slot)
Dim ch As Integer = 0
Dim ODATA As Single = 3.0
Wcon.i8024.VoltageOut(slot, ch, OData)
```

// 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:

[Visual Basic, C#]
void Wcon.ModuleName.CurrentOut(int slot, int ch, float data)

Parameter:

slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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

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

Return Value:

None

Example:

[C#]

```
int slot=1, ch=0;
Wcon.i8024.Initial(slot);
float data=10.0f;
Wcon.i8024.CurrentOut(slot,ch,data);
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8024.Initial(slot)
Dim ch As Integer = 0
Dim ODATA As Single = 10.0
Wcon.i8024.CurrentOut(slot, ch, OData)
```

// 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 Wincon-8000 system.

Syntax:

[Visual Basic, C#]
void Wcon.ModuleName.VoltageOut(int slot, int ch, short data)

Parameter:

slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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:

[C#]

```
int slot=1, ch=0;
Wcon.i8024.Initial(slot);
short data=0x3000;
Wcon.i8024.VoltageHexOut(slot,ch,data);
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8024.Initial(slot)
Dim ch As Integer = 0
Dim ODATA As Short = &H3000
Wcon.i8024.VoltageHexOut(slot, ch, ODATA)
```

// 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 Wincon-8000 system.

Syntax:

[Visual Basic, C#]
void Wcon.ModuleName.CurrentHexOut(int slot, int ch, short data)

Parameter:

slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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:

[C#]

```
int slot=1, ch=0;
Wcon.i8024.Initial(slot);
short data=0x2000;
Wcon.i8024.CurrentHexOut(slot,ch,data);
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8024.Initial(slot)
Dim ch As Integer = 0
Dim ODATA As Short = &H2000

Wcon.i8024.CurrentHexOut(slot, ch, ODATA)
```

// 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 Wincon-8000 system.

Syntax:

[Visual Basic, C#]
float Wcon. ModuleName.VoltageOutReadBack(int slot, int ch)

Parameter:

slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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

Return Value:

a float value.

Example:

[C#]

```
int slot=1, ch=0;
Wcon.i8024.Initial(slot);
float Odata=3.0f;
float Idata;
Wcon.i8024.VoltageOut(slot,ch,Odata);
Idata=Wcon.i8024.VoltageOutReadBack(slot,ch);
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8024.Initial(slot)
Dim ch As Integer = 0
Dim ODATA As Single = 3.0
Wcon.i8024.VoltageOut(slot, ch, ODATA)
Dim IDATA As Single
IDATA = Wcon.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 Wincon-8000 system.

Syntax:

[Visual Basic, C#]

float Wcon.ModuleName.CurrentOutReadBack(int slot, int ch)

Parameter:

slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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

Return Value:

a float value.

Example:

[C#]

```
int slot=1, ch=0;
Wcon.i8024.Initial(slot);
float data=10.0f;
Wcon.i8024.CurrentOut(slot,ch,data);
float Idata;
Idata=Wcon.i8024.CurrentOutReadBack(slot,ch);
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8024.Initial(slot)
Dim ch As Integer = 0
Dim ODATA As Single = 10.0
Wcon.i8024.CurrentOut(slot, ch, ODATA)
Dim IDATA As Single
IDATA = Wcon.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 Wincon-8000 system.

Syntax:

[Visual Basic, C#]

short Wcon.ModuleName.VoltageHexOutReadBack(int slot, int ch)

Parameter:

slot : [Input] Specify the slot of the Wincon-8000 system (Range: 1 to 7)

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:

[C#]

```
int slot=1, ch=0;
Wcon.i8024.Initial(slot);
short data=0x3000; short Idata;
Wcon.i8024.VoltageHexOut(slot,ch,data);
Idata=Wcon.i8024.VoltageHexOutReadBack(slot,ch);
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8024.Initial(slot)
Dim ch As Integer = 0
Dim ODATA As Short = &H3000
Wcon.i8024.VoltageHexOut(slot, ch, ODATA)
Dim IDATA As Short
IDATA = Wcon.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 Wincon-8000 system.

Syntax:

[Visual Basic, C#]

short Wcon.ModuleName.CurrentHexOutReadBack(int slot, int ch)

Parameter:

slot : [Input] Specify the slot of the Wincon-8000 system (Range: 1 to 7)

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:

[C#]

```
int slot=1, ch=0;
Wcon.i8024.Initial(slot);
short data=0x2000;
Wcon.i8024.CurrentHexOut(slot,ch,data);
short Idata;
Idata=Wcon.i8024.CurrentHexOutReadBack(slot,ch);
```

[Visual Basic]

```
Dim slot As Integer = 1
Wcon.i8024.Initial(slot)
Dim ch As Integer = 0
Dim ODATA As Short = &H2000
Wcon.i8024.CurrentHexOut(slot, ch, ODATA)
Dim IDATA As Short
IDATA = Wcon.i8024.CurrentHexOutReadBack(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 WinCon-8000 platform, the I-8090 modules should be registered before applying it. If there is no I-8090 in WinCon-8000 with the given address, this function will return 0 which means a failure.

Syntax:

[Visual Basic, C#]

byte Wcon.ModuleName.REGISTRATION(unsigned char cardNo, int slot)

Parameter:

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

slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

Return Value:

1: Success registration

0: Failure registration

Example:

[C#]

```
byte CARD1=1;
int slot=1;
Wcon.i8090.REGISTRATION(CARD1, slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Wcon.i8090.REGISTRATION(CARD1, slot)
// The I-8090 card plugged in slot 1 of Wincon-8000
```

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:

[Visual Basic, C#]

```
void Wcon.ModuleName.INIT_CARD(byte cardNo, byte x_mode,
                               byte y_mode, byte 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:

[C#]

```
byte CARD1=1;
int slot=1;
Wcon.i8090.REGISTRATION(CARD1, slot);
Wcon.i8090.INIT_CARD(CARD1,0,0,0);// Set to Quadrant counting mode
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Wcon.i8090.REGISTRATION(CARD1, slot)
Wcon.i8090.INIT_CARD(CARD1, 0, 0, 0) ' Set to Quadrant counting mode
```

// 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:

[Visual Basic, C#]

ushort Wcon.ModuleName.GET_ENCODER(byte cardNo, byte 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:

[C#]

```
byte CARD1=1;
int slot=1;
Wcon.i8090.REGISTRATION(CARD1, slot);
Wcon.i8090.INIT_CARD(CARD1,0,0,0);// Set to Quadrant counting mode
ushort Idata=Wcon.i8090.GET_ENCODER(CARD1,1);// Read Encoder
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Wcon.i8090.REGISTRATION(CARD1, slot)
Wcon.i8090.INIT_CARD(CARD1, 0, 0, 0) ' Set to Quadrant counting mode
Dim Idata As UInt16 = Wcon.i8090.GET_ENCODER(CARD1, 1) 'Read Encoder
// 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:

[Visual Basic, C#]

void Wcon.ModuleName.RESET_ENCODER(byte cardNo, byte 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:

[C#]

```
byte CARD1=1;
int slot=1;
Wcon.i8090.REGISTRATION(CARD1, slot);
Wcon.i8090.INIT_CARD(CARD1,0,0,0);// Set to Quadrant counting mode
byte X_axis=1;
Wcon.i8090.RESET_ENCODER(slot,X_axis);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Wcon.i8090.REGISTRATION(CARD1, slot)
Wcon.i8090.INIT_CARD(CARD1, 0, 0, 0) ' Set to Quadrant counting mode
Dim X_axis As Byte = 1
Wcon.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:

[Visual Basic, C#]

int Wcon.ModuleName.GET_ENCODER32(byte cardNo, byte 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:

[C#]

```
byte CARD1=1;
int slot=1;
Wcon.i8090.REGISTRATION(CARD1, slot);
Wcon.i8090.INIT_CARD(CARD1,0,0,0);// Set to Quadrant counting mode
byte X_axis=1;
int Idata=Wcon.i8090.GET_ENCODER32(CARD1,X_axis);// Read Encoder
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Wcon.i8090.REGISTRATION(CARD1, slot)
Wcon.i8090.INIT_CARD(CARD1, 0, 0, 0) ' Set to Quadrant counting mode
Dim X_axis As Byte = 1
Dim Idata As Integer = Wcon.i8090.GET_ENCODER32(CARD1, X_axis) 'Read Encoder
```

//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)

[Visual Basic, C#]

void Wcon.ModuleName.RESET_ENCODER32(byte cardNo, byte 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:

[C#]

```
byte CARD1=1;
int slot=1;
Wcon.i8090.REGISTRATION(CARD1, slot);
Wcon.i8090.INIT_CARD(CARD1,0,0,0);// Set to Quadrant counting mode
byte X_axis=1;
Wcon.i8090.RESET_ENCODER32(slot,X_axis);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Wcon.i8090.REGISTRATION(CARD1, slot)
Wcon.i8090.INIT_CARD(CARD1, 0, 0, 0) ' Set to Quadrant counting mode
Dim X_axis As Byte = 1
Wcon.i8090.RESET_ENCODER32(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:

[Visual Basic, C#]

byte Wcon.ModuleName.GET_INDEX(byte 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:

[C#]

```
byte CARD1=1;
int slot=1;
Wcon.i8090.REGISTRATION(CARD1, slot);
Wcon.i8090.INIT_CARD(CARD1,0,0,0);// Set to Quadrant counting mode
Wcon.i8090.GET_INDEX(CARD1);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Wcon.i8090.REGISTRATION(CARD1, slot)
Wcon.i8090.INIT_CARD(CARD1, 0, 0, 0) ' Set to Quadrant counting mode
Wcon.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:

[Visual Basic, C#]
void Wcon.ModuleName.ENCODER32_ISR(byte cardNo)

Parameter:

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

Return Value:

None

Example:

[C#]

```
byte CARD1=1;
int slot=1;
Wcon.i8090.REGISTRATION(CARD1, slot);
Wcon.i8090.INIT_CARD(CARD1,0,0,0);// Set to Quadrant counting mode
Wcon.i8090.ENCODER32_ISR(CARD1);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Wcon.i8090.REGISTRATION(CARD1, slot)
Wcon.i8090.INIT_CARD(CARD1, 0, 0, 0) ' Set to Quadrant counting mode
Wcon.i8090.ENCODER32_ISR(CARD1)
```

//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 WinCon-8000 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:

[Visual Basic, C#]

byte Wcon.ModuleName.REGISTRATION(byte 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:

[C#]

```
byte CARD1=1;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Wcon.i8091.REGISTRATION(CARD1, slot)
// The I-8091 card is plugged into slot 1 of Wincon-8000.
```

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:

[Visual Basic, C#]

```
void Wcon.ModuleName.RESET_SYSTEM(unsigned char cardNo)
```

Parameter:

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

Return Value:

None

Example:

[C#]

```
byte CARD1=1;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.RESET_SYSTEM(CARD1);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.RESET_SYSTEM(CARD1)
// The I-8091 card plugged in slot 1 of Wincon-8000
```

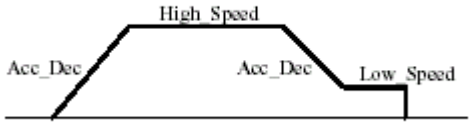
Remark:

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:

[Visual Basic, C#]

void Wcon.ModuleName.SET_VAR(byte cardNo, byte DDA_cycle,
byte Acc_Dec, ushort Low_Speed, ushort 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:

[C#]

```
byte CARD1=1
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.SET_VAR(CARD1, 5, 2, 10, 150);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.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:

[Visual Basic, C#]

```
void Wcon.ModuleName.SET_DEFDIR(byte cardNo, byte defdirX, byte 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:

[C#]

```
byte CARD1=1; int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
byte NORMAL_DIR=0;
Wcon.i8091.SET_DEFDIR(CARD1,NORMAL_DIR,NORMAL_DIR);
```

[Visual Basic]

```
Dim slot As Integer = 1: Dim CARD1 As Byte = 1
Wcon.i8091.REGISTRATION(CARD1, slot)
Dim NORMAL_DIR As Byte = 0
Wcon.i8091.SET_DEFDIR(CARD1, NORMAL_DIR, NORMAL_DIR)
```

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:

[Visual Basic, C#]

void Wcon.ModuleName.SET_MODE(byte cardNo, byte modeX, byte 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:

[C#]

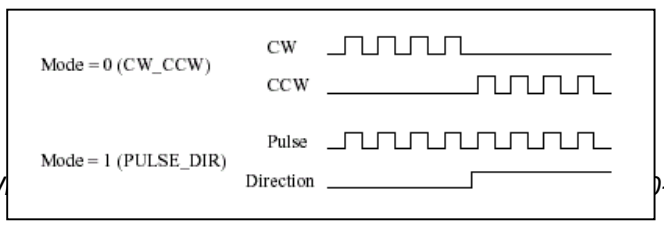
```
byte CARD1=1; int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
byte CW_CCW=0;
Wcon.i8091.SET_MODE(CARD1,CW_CCW,CW_CCW);
```

[Visual Basic]

```
Dim slot As Integer = 1: Dim CARD1 As Byte = 1
Wcon.i8091.REGISTRATION(CARD1, slot)
Dim CW_CCW As Byte = 0
Wcon.i8091.SET_MODE(CARD1, CW_CCW, CW_CCW)
```

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:

[Visual Basic, C#]

void Wcon.ModuleName.SET_SERVO_ON(byte cardNo, byte sonX, byte 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:

[C#]

```
byte CARD1=1; int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
byte ON=1;
Wcon.i8091.SET_SERVO_ON(CARD1,ON,ON);
```

[Visual Basic]

```
Dim slot As Integer = 1: Dim CARD1 As Byte = 1
Wcon.i8091.REGISTRATION(CARD1, slot)
Dim SON As Byte = 1
Wcon.i8091.SET_SERVO_ON(CARD1, SON, SON)
```

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:

[Visual Basic, C#]
void Wcon.ModuleName.SET_NC(byte cardNo, byte sw)

Parameter:

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

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

Return Value:

None

Example:

[C#]

```
byte CARD1=1; int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
byte NO=1;
Wcon.i8091.SET_NC(CARD1,NO);
```

[Visual Basic]

```
Dim slot As Integer = 1: Dim CARD1 As Byte = 1
Wcon.i8091.REGISTRATION(CARD1, slot)
Dim NO As Byte = 0
Wcon.i8091.SET_NC(CARD1, NO)
```

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:

[Visual Basic, C#]

void Wcon.ModuleName.STOP_X(byte cardNo)

Parameter:

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

Return Value:

None

Example:

[C#]

```
byte CARD1=1; int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.STOP_X(CARD1);
```

[Visual Basic]

```
Dim slot As Integer = 1: Dim CARD1 As Byte = 1
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.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:

[Visual Basic, C#]

```
void Wcon.ModuleName.STOP_Y(byte cardNo)
```

Parameter:

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

Return Value:

None

Example:

[C#]

```
byte CARD1=1; int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.STOP_Y(CARD1);
```

[Visual Basic]

```
Dim slot As Integer = 1: Dim CARD1 As Byte = 1
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.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:

[Visual Basic, C#]
void Wcon.ModuleName.STOP_ALL(byte cardNo)

Parameter:

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

Return Value:

None

Example:

[C#]

```
byte CARD1=1; int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.STOP_ALL(CARD1);
```

[Visual Basic]

```
Dim slot As Integer = 1: Dim CARD1 As Byte = 1
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.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:

[Visual Basic, C#]

```
void Wcon.ModuleName.EMG_STOP(byte cardNo)
```

Parameter:

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

Return Value:

None

Example:

[C#]

```
byte CARD1=1; int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.EMG_STOP(CARD1);
```

[Visual Basic]

```
Dim slot As Integer = 1: Dim CARD1 As Byte = 1
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.EMG_STOP(CARD1)
// The X-axis and Y-axis are stopped.
```

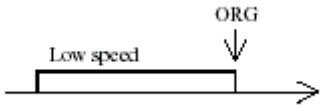
Remark:

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:

[Visual Basic, C#]

void Wcon.ModuleName.LSP_ORG(byte cardNo, byte DIR, byte 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:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.LSP_ORG(CARD1,CCW,X_axis);
Wcon.i8091.LSP_ORG(CARD1,CCW,Y_axis);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte =2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.LSP_ORG(CARD1, CCW, X_axis)
Wcon.i8091.LSP_ORG(CARD1, CCW, Y_axis)
```

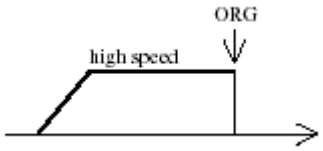
Remark:

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:

[Visual Basic, C#]

void Wcon.ModuleName.HSP_ORG(byte cardNo, byte DIR, byte 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:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.HSP_ORG(CARD1,CCW,X_axis);
Wcon.i8091.HSP_ORG(CARD1,CCW,Y_axis);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.HSP_ORG(CARD1, CCW, X_axis)
Wcon.i8091.HSP_ORG(CARD1, CCW, Y_axis)
```

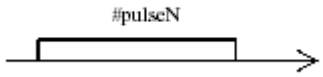
Remark:

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:

[Visual Basic, C#]

void Wcon.ModuleName.LSP_PULSE_MOVE(byte cardNo, byte AXIS, int 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:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.LSP_PULSE_MOVE(CARD1, X_axis, 20000);
Wcon.i8091.LSP_PULSE_MOVE(CARD1, X_axis, -2000);
Wcon.i8091.LSP_PULSE_MOVE(CARD1, Y_axis, 20000);
Wcon.i8091.LSP_PULSE_MOVE(CARD1, Y_axis, -2000);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.LSP_PULSE_MOVE(CARD1, X_axis, 20000)
Wcon.i8091.LSP_PULSE_MOVE(CARD1, X_axis, -2000)
```

```
Wcon.i8091.LSP_PULSE_MOVE(CARD1, Y_axis, 20000)
Wcon.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
```

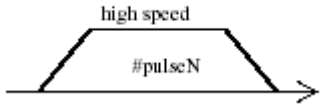
Remark:

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:

[Visual Basic, C#]

void Wcon.ModuleName.HSP_PULSE_MOVE(byte cardNo, byte AXIS, int 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:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.HSP_PULSE_MOVE(CARD1, X_axis, 20000);
Wcon.i8091.HSP_PULSE_MOVE(CARD1, X_axis, -2000);
Wcon.i8091.HSP_PULSE_MOVE(CARD1, Y_axis, 20000);
Wcon.i8091.hSP_PULSE_MOVE(CARD1, Y_axis, -2000);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.HSP_PULSE_MOVE(CARD1, X_axis, 20000)
Wcon.i8091.HSP_PULSE_MOVE(CARD1, X_axis, -2000)
```

```
Wcon.i8091.HSP_PULSE_MOVE(CARD1, Y_axis, 20000)
Wcon.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
```

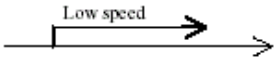
Remark:

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:

[Visual Basic, C#]

void Wcon.ModuleName.LSP_MOVE(byte cardNo, byte DIR, byte 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:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.LSP_MOVE(CARD1,CCW,X_axis);
Wcon.i8091.LSP_MOVE(CARD1,CCW,Y_axis);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.LSP_MOVE(CARD1, CCW, X_axis)
Wcon.i8091.LSP_MOVE(CARD1, CCW, Y_axis)
```

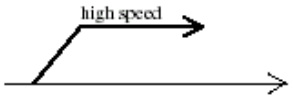
Remark:

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:

[Visual Basic, C#]

void Wcon.ModuleName.HSP_MOVE(byte cardNo, byte DIR, byte 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:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.HSP_MOVE(CARD1,CCW,X_axis);
Wcon.i8091.HSP_MOVE(CARD1,CCW,Y_axis);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.HSP_MOVE(CARD1, CCW, X_axis)
Wcon.i8091.HSP_MOVE(CARD1, CCW, Y_axis)
```

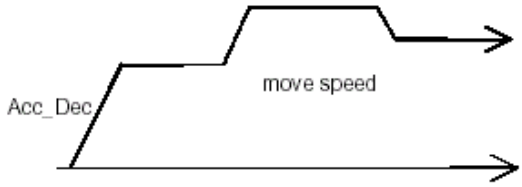
Remark:

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:

[Visual Basic, C#]

void Wcon.ModuleName.CSP_MOVE(byte cardNo, byte dir, byte axis, ushort 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:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.CSP_MOVE(CARD1, CCW, X_axis, 10);
Wcon.i8091.CSP_MOVE(CARD1, CCW, X_axis, 20);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
```

```
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.CSP_MOVE(CARD1, CCW, X_axis, Convert.ToUInt16(10))
Wcon.i8091.CSP_MOVE(CARD1, CCW, X_axis, Convert.ToUInt16(20))
```

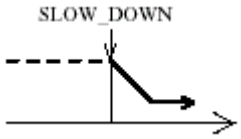
Remark:

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:

[Visual Basic, C#]

void Wcon.ModuleName.SLOW_DOWN(byte cardNo, byte AXIS)

Parameter:

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

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

Return Value:

None

Example:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.HSP_MOVE(CARD1, CCW, X_axis);
Wcon.i8091.SLOW_DOWN(CARD1, X_axis);
Wcon.i8091.STOP_X(CARD1);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.HSP_MOVE(CARD1, CCW, X_axis)
Wcon.i8091.SLOW_DOWN(CARD1, X_axis)
Wcon.i8091.STOP_X(CARD1)
```

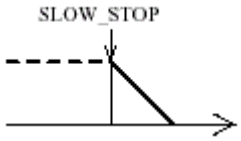
Remark:

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:

[Visual Basic, C#]

void Wcon.ModuleName.SLOW_STOP(byte cardNo, byte AXIS)

Parameter:

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

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

Return Value:

None

Example:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.HSP_MOVE(CARD1, CCW, X_axis);
Wcon.i8091.SLOW_STOP(CARD1,X_axis);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.HSP_MOVE(CARD1, CCW, X_axis)
Wcon.i8091.SLOW_STOP(CARD1,X_axis)
```

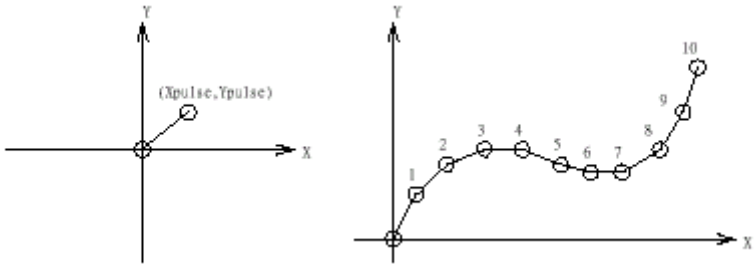
Remark:

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:

[Visual Basic, C#]

void Wcon.ModuleName.INTP_PULSE(byte cardNo, short Xpulse, short Ypulse)

Parameter:

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

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

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

Return Value:

None

Example:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.INTP_PULSE(CARD1, 20, 20);
Wcon.i8091.INTP_PULSE(CARD1, 20, 13);
Wcon.i8091.INTP_PULSE(CARD1, 20, 7);
Wcon.i8091.INTP_PULSE(CARD1, 20, 0);
Wcon.i8091.INTP_PULSE(CARD1, 15,-5);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
```

```
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.INTP_PULSE(CARD1, 20, 20)
Wcon.i8091.INTP_PULSE(CARD1, 20, 13)
Wcon.i8091.INTP_PULSE(CARD1, 20, 7)
Wcon.i8091.INTP_PULSE(CARD1, 20, 0)
Wcon.i8091.INTP_PULSE(CARD1, 15, -5)
```

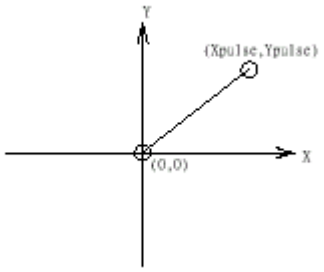
Remark:

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:

[Visual Basic, C#]

void Wcon.ModuleName.INTP_LINE(byte cardNo, short Xpulse, short Ypulse)

Parameter:

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

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

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

Return Value:

None

Example:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.INTP_LINE(CARD1, 2000, -3000);
Wcon.i8091.INTP_LINE(CARD1, -500, 200);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
```

```
Wcon.i8091.INTP_LINE(CARD1, 2000, -3000)
Wcon.i8091.INTP_LINE(CARD1, -500, 200)
```

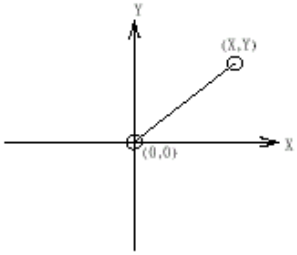
Remark:

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:

[Visual Basic, C#]

void Wcon.ModuleName.INTP_LINE02(byte cardNo, int X, int Y, ushort speed, byte 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:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.INTP_LINE02(CARD1, 1000, 1000, 100, 0);
do { } while(Wcon.i8091.INTP_STOP()!=0) ; //call state machine
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
```

```
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.INTP_LINE02(CARD1, 1000, 1000, Convert.ToUInt16(100), 0)
Do
Loop While (Wcon.i8091.INTP_STOP() <> 0) 'call state machine
```

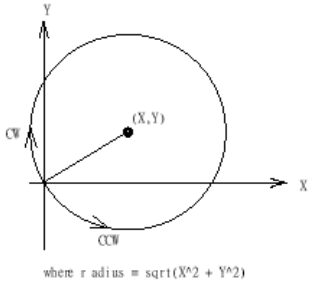
Remark:

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:

[Visual Basic, C#]

void Wcon.ModuleName.INTP_CIRCLE02(byte cardNo, int X, int Y, byte dir
ushort speed, byte 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:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.INTP_CIRCLE02( CARD1 , 2000, 2000, CCW, 100, 0);
do { } while(Wcon.i8091.INTP_STOP()!=0) ; //call state machine
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.INTP_LINE02(CARD1, 1000, 1000, Convert.ToUInt16(100), 0)
Do
Loop While (Wcon.i8091.INTP_STOP() <> 0) 'call state machine
```

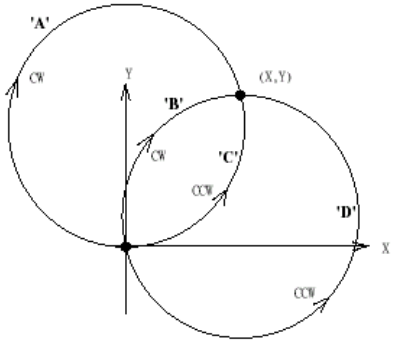
Remark:

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:

[Visual Basic, C#]

```
void Wcon.ModuleName.INTP_ARC02(byte cardNo, int X, int Y, int R,
        byte dir, ushort speed, byte 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:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.INTP_ARCO2(CARD1, 2000, -2000, 2000, CW, 100, 0);
do { } while(Wcon.i8091.INTP_STOP()!=0) ; //call state machine
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.INTP_ARCO2(CARD1, 2000, -2000, 2000, CW, Convert.ToUInt16(100), 0)
Do
Loop While (Wcon.i8091.INTP_STOP() <> 0) '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:

[Visual Basic, C#]

```
byte Wcon.ModuleName.INTP_STOP(void)
```

Parameter:

None

Return Value:

0: READY
1: BUSY

Example:

Please reference to the Wcon.i8091.INTP_ARC02 Example

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:

[Visual Basic, C#]

byte Wcon.ModuleName.LIMIT_X(byte 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:

[C#]

```
byte CARD1=1;byte X_axis=1;byte CCW=1;byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
byte DATA =Wcon.i8091.LIMIT_X(CARD1);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1 : Dim CCW As Byte = 1 : Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Dim data As Byte = Wcon.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)

[Visual Basic, C#]

byte Wcon.ModuleName.LIMIT_Y(byte 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:

[C#]

```
byte CARD1=1
byte X_axis=1
byte CCW=1
byte Y_axis=2;
int slot=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
byte DATA =Wcon.i8091.LIMIT_Y(CARD1);
```

[Visual Basic]

```
Dim slot As Integer = 1
```

```
Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1
Dim CCW As Byte = 1
Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Dim data As Byte = Wcon.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:

[Visual Basic, C#]

```
void Wcon.ModuleName.WAIT_X(byte cardNo)
```

Parameter:

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

Return Value:

None

Example:

[C#]

```
byte CARD1=1
int slot=1;
byte X_axis=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.HSP_PULSE_MOVE(CARD1, X_axis, 20000);
Wcon.i8091.WAIT_X(CARD1);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.HSP_PULSE_MOVE(CARD1, X_axis, 20000)
Wcon.i8091.WAIT_X(CARD1)
```

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:

[Visual Basic, C#]

```
void Wcon.ModuleName.WAIT_Y(byte cardNo)
```

Parameter:

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

Return Value:

None

Example:

[C#]

```
byte CARD1=1
int slot=1
byte Y_axis=2;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.HSP_PULSE_MOVE(CARD1, Y_axis, 20000);
Wcon.i8091.WAIT_Y(CARD1);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.HSP_PULSE_MOVE(CARD1, Y_axis, 20000)
Wcon.i8091.WAIT_Y(CARD1)
```

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:

[Visual Basic, C#]

byte Wcon.ModuleName.IS_X_STOP(byte cardNo)

Parameter:

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

Return Value:

0 (NO) : not yet stop

1 (YES) : stop

Example:

[C#]

```
byte CARD1=1
int slot=1
byte X_axis=1;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.HSP_PULSE_MOVE(CARD1, X_axis, 20000);
byte data=Wcon.i8091.IS_X_STOP(CARD1);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Dim X_axis As Byte = 1
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.HSP_PULSE_MOVE(CARD1, X_axis, 20000)
Dim data As Byte = Wcon.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:

[Visual Basic, C#]

byte Wcon.ModuleName.IS_Y_STOP(byte cardNo)

Parameter:

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

Return Value:

0 (NO) : not yet stop

1 (YES) : stop

Example:

[C#]

```
byte CARD1=1
int slot=1
byte Y_axis=2;
Wcon.i8091.REGISTRATION(CARD1, slot);
Wcon.i8091.HSP_PULSE_MOVE(CARD1, Y_axis, 20000);
byte data=Wcon.i8091.IS_Y_STOP(CARD1);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim CARD1 As Byte = 1
Dim Y_axis As Byte = 2
Wcon.i8091.REGISTRATION(CARD1, slot)
Wcon.i8091.HSP_PULSE_MOVE(CARD1, Y_axis, 20000)
Dim data As Byte = Wcon.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:

[Visual Basic, C#]
int Wcon.ModuleName.InitDriver(int Slot)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

Return Value:

0: OK

The others: Error codes, Refer to I8080.h

Example:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1
ErrorCode = Wcon.i8080.InitDriver(slot)
// The I-8080 card is plugged in slot 1 of Wincon-8000
// 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:

[Visual Basic, C#]

```
void Wcon.ModuleName.AutoScan()
```

Parameter:

None

Return Value:

None

Example:

[C#]

```
private void timer1_Tick(object sender, System.EventArgs e)
{
      Wcon.i8080.AutoScan();
}
```

[Visual Basic]

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer1.Tick
   Wcon.i8080.AutoScan()
End Sub
```

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:

[Visual Basic, C#]

int Wcon.ModuleName.ReadDIXor(int Slot, out short Di)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
short Di;
Wcon.i8080.ReadDiXor(slot,out Di);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
Dim Di As Short
Wcon.i8080.ReadDiXor(slot, Di)
// Di=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:

[Visual Basic, C#]

```
int Wcon.ModuleName.ReadDIXorLp(int Slot, out short Di)
```

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
short Di;
Wcon.i8080.ReadDiXorLp(slot,out Di);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
Dim Di As Short
Wcon.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:

[Visual Basic, C#]

int Wcon.ModuleName.ReadXorRegister(int Slot, out short XorVal)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
short Di;
Wcon.i8080.ReadXorRegister(slot,out Di);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
Dim Di As Short
Wcon.i8080.ReadXorRegister(slot, Di)
// Di =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:

[Visual Basic, C#]
int Wcon.ModuleName.SetXorRegister(int Slot, short XorVal)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7).

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:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
short Di=0;
Wcon.i8080.SetXorRegister(slot,Di);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
Dim Di As Short = 0
Wcon.i8080.SetXorRegister(slot, Di)
```

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:

[Visual Basic, C#]

int Wcon.ModuleName.EepWriteEnable(int Slot)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

Return Value:

0: OK

The others: Error codes

Example:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
Wcon.i8080.EepWriteEnable(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
Wcon.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:

[Visual Basic, C#]
int Wcon.ModuleName.EepWriteDisable(int Slot)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

Return Value:

0: OK

The others: Error codes

Example:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
Wcon.i8080.EepWriteDisable(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
Wcon.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:

[Visual Basic, C#]
int Wcon.ModuleName.EepWriteWord(int Slot, short Addr, short Hi, short Lo)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
Wcon.i8080.EepWriteEnable(slot);
Wcon.i8080.EepWriteWord(slot,33,44,44);
Wcon.i8080.EepWriteDisable(slot);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
Wcon.i8080.EepWriteEnable(slot)
Wcon.i8080.EepWriteWord(slot, 33, 44, 44)
Wcon.i8080.EepWriteDisable(slot)
```

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:

[Visual Basic, C#]

int Wcon.ModuleName.EepReadWord(int Slot, out short Addr, out short Hi, out short Lo)

Parameter:

Slot :	[Input] Specify the Wincon-8000 system slot (Range: 1 to 7)
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:

[C#]

```
int slot=1;
int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
short Hi,Lo;
Wcon.i8080.EepReadWord(slot,33,out Hi,out Lo);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
Dim Hi, Lo As Short
Wcon.i8080.EepReadWord(slot, 33, Hi, Lo)
```

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:

[Visual Basic, C#]

```
int Wcon.ModuleName.ReadChannelMode(int Slot, int Channel, out short *Mode)
```

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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:

[C#]

```
int slot=1;
int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
short mode;
int ch=0;
Wcon.i8080.ReadChannelMode(slot,ch,out mode);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
Dim mode As Short
Dim ch As Int16 = 0
Wcon.i8080.ReadChannelMode(slot, ch, mode)
```

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:

[Visual Basic, C#]

int Wcon.ModuleName.SetChannelMode(int Slot, int Channel, short Mode)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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:

[C#]

```
int slot=1;
int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
short mode=1;
int ch=0; // A0, Up_counting_0
Wcon.i8080.SetChannelMode(slot,ch,mode);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
Dim mode As Short = 1
Dim ch As Int16 = 0
Wcon.i8080.SetChannelMode(slot, ch, mode) '// A0, Up_counting_0
```

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:

[Visual Basic, C#]

int Wcon.ModuleName.ReadFreq(int Slot, int Channel, out float *f)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
short AutoTT=2000;    // set Sampling time period for Auto mode, unit= millisecond
short AutoVV=5;       // set Samples to start frequency update (Auto mode)
short LowTT=10000;    // set Sampling time period for Low Frequency mode, unit=millisecond
short LowVV=5;        // set Samples to start frequency update (Low Frequency Mode)
short HighTT=1000;    // set Sampling time period for High Frequency mode, unit=millisecond
short HighVV=100;     // set Samples to start frequency update (High Frequency Mode)
float Hz; short mode=2; int ch=0;  //Frequency Mode
Wcon.i8080.SetChannelMode(slot,ch,mode);
Wcon.i8080.SetFreqConfiguration(slot,AutoTT, AutoVV,LowTT,LowVV,HighTT,HighVV);
Wcon.i8080.ReadFreq(slot,ch,out Hz);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
```

```
Dim AutoTT As Short = 2000  'set Sampling time period for Auto mode, unit= millisecond
Dim AutoVV As Short = 5     'set Samples to start frequency update (Auto mode)
Dim LowTT As Short = 10000  'set Sampling time period for Low Frequency mode, unit=millisecond
Dim LowVV As Short = 5      'set Samples to start frequency update (Low Frequency Mode)
Dim HighTT As Short = 1000  'set Sampling time period for High Frequency mode, unit=millisecond
Dim HighVV As Short = 100   'set Samples to start frequency update (High Frequency Mode)
Dim mode As Short = 2 : Dim ch As Int16 = 0
Wcon.i8080.SetChannelMode(slot, ch, mode) '//Frequency Mode
Wcon.i8080.SetFreqConfiguration(slot, AutoTT, AutoVV, LowTT, LowVV, HighTT, HighVV)
Dim Hz As Single
Wcon.i8080.ReadFreq(slot, ch, Hz)
```

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:

[Visual Basic, C#]

int Wcon.ModuleName.ReadCntUp(int Slot, int Channel, out uint Cnt32U, out ushort Overflow)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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:

[C#]

```
int slot=1;
int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
short mode=3;
int ch=0; // A0, Up Count Mode
Wcon.i8080.SetChannelMode(slot,ch,mode);
uint Coun;
int ch=0;
ushort Overflow;
Wcon.i8080.ReadCntUp(slot ,ch,out Coun,out Overflow);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
```

```
Dim mode As Short = 1
Dim ch As Int16 = 0
Wcon.i8080.SetChannelMode(slot, ch, mode) '// A0, Up Count Mode
Dim Coun As UInt32
Dim Overflow As UInt16
Wcon.i8080.ReadCntUp(slot, ch, Coun, Overflow)
```

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:

[Visual Basic, C#]

int Wcon.ModuleName.ReadCntUpDown(int Slot, int Channel, out uint Cnt32U, out ushort Overflow)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
short mode=1; int ch=0; // A0 B0, Up/Down Mode
Wcon.i8080.SetChannelMode(slot,ch,mode);
uint Coun; int ch=0; ushort Overflow;
Wcon.i8080.ReadCntUp(slot ,ch,out Coun,out Overflow);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
Dim mode As Short = 1
Dim ch As Int16 = 0
```

```
Wcon.i8080.SetChannelMode(slot, ch, mode) '// A0 B0, Up/Down Mode
Dim Coun As UInt32
Dim Overflow As UInt16
Wcon.i8080.ReadCntUp(slot, ch, Coun, Overflow)
```

// 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:

[Visual Basic, C#]

```
int Wcon.ModuleName.SetLowPassUs(int Slot, int Channel, ushort Us)
```

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
short mode=0x43; int ch=0; // A0, Up Count and Low Pass filter mode
Wcon.i8080.SetChannelMode(slot,ch,mode);
ushort Us=1000;
Wcon.i8080.SetLowPassUs(slot,ch,Us);  //set the Low pass filter value
uint Coun; ushort Overflow;
Wcon.i8080.ReadCntUp(slot ,ch,out Coun,out Overflow);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
```

```
Dim mode As Short = &H43
Dim ch As Int16 = 0
Wcon.i8080.SetChannelMode(slot, ch, mode)    ' AO, Up Count and Low Pass filter mode
Dim Us As UInt16 = Convert.ToUInt16(1000)
Wcon.i8080.SetLowPassUs(slot, ch, Us)        'set the Low pass filter value
Dim Coun As UInt32 : Dim Overflow As UInt16
Wcon.i8080.ReadCntUp(slot, ch, Coun, Overflow)
```

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:

[Visual Basic, C#]

int Wcon.ModuleName.ReadLowPassUs(int Slot, int Channel, out ushort Us)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
short mode=0x43; int ch=0; // A0, Up Count and Low Pass filter mode
Wcon.i8080.SetChannelMode(slot,ch,mode);
ushort Us=1000; ushort UsI;
Wcon.i8080.SetLowPassUs(slot,ch,Us);  //set the Low pass filter value
Wcon.i8080.ReadLowPassUs(slot,ch,out UsI);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
Dim mode As Short = &H43
Dim ch As Int16 = 0
Wcon.i8080.SetChannelMode(slot, ch, mode)    ' A0, Up Count and Low Pass filter mode
```

```
Dim Us As UInt16 = Convert.ToUInt16(1000)
Wcon.i8080.SetLowPassUs(slot, ch, Us)        'set the Low pass filter value
Dim UsI As UInt16
Wcon.i8080.ReadLowPassUs(slot, ch, UsI)
```

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:

[Visual Basic, C#]

int Wcon.ModuleName.ClrCnt(int Slot, int Channel)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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

Return Value:

0: OK

The others: Error codes

Example:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
Wcon.i8080.ClrCnt(slot,0);
Wcon.i8080.ClrCnt(slot,1);
Wcon.i8080.ClrCnt(slot,4);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
Wcon.i8080.ClrCnt(slot,0)
Wcon.i8080.ClrCnt(slot,1)
Wcon.i8080.ClrCnt(slot,4)
```

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:

[Visual Basic, C#]

```
int Wcon.ModuleName.ReadFreqConfiguration(int Slot, out short AutoTT ,
                    out short AutoVV, out short LowTT, out short LowVV,
                    out short HighTT, out short HighVV)
```

Parameter:

Slot :	[Input] Specify the Wincon-8000 system slot (Range: 1 to 7)
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:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);
short AutoTT, AutoVV, LowTT, LowVV, HighTT, HighVV;
Wcon.i8080.ReadFreqConfiguration(slot,out AutoTT,out AutoVV,out LowTT,out LowVV,out HighTT,out
HighVV);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
```

```
Dim AutoTT, AutoVV, LowTT, LowVV, HighTT, HighVV
Wcon.i8080.ReadFreqConfiguration(slot, AutoTT, AutoVV, LowTT, LowVV, HighTT, HighVV)
```

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:

[Visual Basic, C#]

int Wcon.ModuleName.SetFreqConfiguration(int Slot, short AutoTT ,
short AutoVV, short LowTT, short LowVV, short HighTT, short HighVV)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

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:

[C#]

```
int slot=1; int ErrorCode;
ErrorCode=Wcon.i8080.InitDriver(slot);

short AutoTT=2000;    // set Sampling time period for Auto mode, unit= millisecond
short AutoVV=5;       // set Samples to start frequency update (Auto mode)
short LowTT=10000;    // set Sampling time period for Low Frequency mode, unit=millisecond
short LowVV=5;        // set Samples to start frequency update (Low Frequency Mode)
short HighTT=1000;    // set Sampling time period for High Frequency mode, unit=millisecond
```

```
short HighVV=100;     // set Samples to start frequency update (High Frequency Mode)
Wcon.i8080.SetFreqConfiguration(slot,AutoTT, AutoVV,LowTT,LowVV,HighTT,HighVV);
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
ErrorCode = Wcon.i8080.InitDriver(slot)
Dim AutoTT As Short = 2000  'set Sampling time period for Auto mode, unit= millisecond
Dim AutoVV As Short = 5     'set Samples to start frequency update (Auto mode)
Dim LowTT As Short = 10000  'set Sampling time period for Low Frequency mode, unit=millisecond
Dim LowVV As Short = 5      'set Samples to start frequency update (Low Frequency Mode)
Dim HighTT As Short = 1000  'set Sampling time period for High Frequency mode, unit=millisecond
Dim HighVV As Short = 100   'set Samples to start frequency update (High Frequency Mode)
Wcon.i8080.SetFreqConfiguration(slot, AutoTT, AutoVV, LowTT, LowVV, HighTT, HighVV)
```

Remark:

This function can be applied on module: i8080.

■ i87082.ClearCounter

Description:

This function is used to clear the specific channel counter value in a specific slot.

Syntax:

[Visual Basic, C#]

int Wcon.ModuleName.ClearCounter(byte Slot, byte Channel)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

Channel : [Input] 0 ~ 1 = Channel Number

Return Value:

0: OK

The others: Error codes

Example:

[C#]

```
byte slot=1; int ErrorCode;
Wcon.i87K_Open_COM();
Wcon.System.ChangeSlotTo87K(slot);
ErrorCode= Wcon.i87082.ClearCounter(slot,0);
Wcon.i87K_Close_COM();
```

[Visual Basic]

```
Dim slot As Integer = 1 : Dim ErrorCode As Integer
Wcon.i87K_Open_COM()
Wcon.System.ChangeSlotTo87K(slot)
ErrorCode = Wcon.i87082.ClearCounter(slot,0)
Wcon.i87K_Close_COM()
```

Remark:

This function can be applied on module: i87082.

■ i87082.ReadCounterStatus

Description:

This function obtains the counter working status (reading/stop) of I87082 modules.

Syntax:

[Visual Basic, C#]
int Wcon.ModuleName. ReadCounterStatus(byte Slot, byte Channel, out byte CounterStatus)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

Channel : [Input] 0 ~ 1 = Channel Number

CounterStatus:[Output] 0: Stop. 1 : Counting.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte  slot  = 1;     byte ch  = 0;
byte Staus ;
Wcon.i87K_Open_COM();
Wcon.System.ChangeSlotTo87K(slot);
Wcon.i87082.ReadCounterStatus(slot, ch,out Staus);
Wcon.i87K_Close_COM();
```

[Visual Basic]

```
Dim slot As Byte = 1 : Dim ch As Byte = 0
Dim Staus As Byte
Wcon.i87K_Open_COM()
Wcon.System.ChangeSlotTo87K(slot)
Wcon.i87082.ReadCounterStatus(slot, ch, Staus)
Wcon.i87K_Close_COM()
```

Remark:

This function can be applied on module: i87082.

■ i87082. ReadCounter

Description:

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

Syntax:

[Visual Basic, C#]

int Wcon.ModuleName.ReadCounter(int Slot, int Channel, out uint Cnt32U,)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

Channel : [Input] 0 ~ 1 = Channel Number

Cnt32U : [Output] 32-bit Up Counter

Return Value:

0: OK

The others: Error codes

Example:

[C#]

```
byte  slot  = 1; byte ch  = 0;
byte Staus ; uint ReadIN;
Wcon.i87K_Open_COM();
Wcon.System.ChangeSlotTo87K(slot);
Wcon.i87082.ReadCounter(slot, ch,out ReadIN);
Wcon.i87K_Close_COM();
```

[Visual Basic]

```
Dim slot As Byte = 1 :Dim ch As Byte = 0
Dim Staus As Byte : Dim ReadIN As UInt32
Wcon.i87K_Open_COM()
Wcon.System.ChangeSlotTo87K(slot)
Wcon.i87082.ReadCounter(slot, ch, ReadIN)
Wcon.i87K_Close_COM()
```

Remark:

This function can be applied on module: i87082.

■ i87082.ReadFreq

Description:

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

Syntax:

[Visual Basic, C#]
int Wcon.ModuleName.ReadFreq(int Slot, int Channel, out uint f)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

Channel : [Input] 0 ~ 1 = Channel Number

f : [Output] The frequency Measured value.

Return Value:

0: OK

The others: Error codes

Example:

[C#]

```
byte  slot  = 1;
byte ch  = 0;
byte Staus ;
uint ReadIN;
Wcon.i87K_Open_COM();
Wcon.System.ChangeSlotTo87K(slot);
Wcon.i87082.ReadFreq(slot, ch,out ReadIN);
Wcon.i87K_Close_COM();
```

[Visual Basic]

```
Dim slot As Byte = 1
Dim ch As Byte = 0
Dim Staus As Byte
Dim ReadIN As UInt32
Wcon.i87K_Open_COM()
Wcon.System.ChangeSlotTo87K(slot)
```

```
Wcon.i87082.ReadFreq(slot, ch, ReadIN)
Wcon.i87K_Close_COM()
```

Remark:

This function can be applied on module: i87082.

■ i87082.ReadCounterStatus

Description:

This function obtains the counter working status (Counting /stop) of I87082 modules.

Syntax:

[Visual Basic, C#]

int Wcon.ModuleName. ReadCounterStatus(byte Slot, byte Channel, out byte CounterStatus)

Parameter:

Slot : [Input] Specify the Wincon-8000 system slot (Range: 1 to 7)

Channel : [Input] 0 ~ 1 = Channel Number

CounterStatus:[Output] 0: Stop. 1 : Counting.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte  slot  = 1;
byte ch  = 0;
byte Staus ;
Wcon.i87K_Open_COM();
Wcon.System.ChangeSlotTo87K(slot);
Wcon.i87082.ReadCounterStatus(slot, ch,out Staus);
Wcon.i87K_Close_COM();
```

[Visual Basic]

```
Dim slot As Byte = 1
Dim ch As Byte = 0
Dim Staus As Byte
Wcon.i87K_Open_COM()
Wcon.System.ChangeSlotTo87K(slot)
```

```
Wcon.i87082.ReadCounterStatus(slot, ch, Staus)
Wcon.i87K_Close_COM()
```

Remark:

This function can be applied on module: i87082.

3. Using Wincon-8000 Serial Ports

This section describes how to use the three serial ports (RS-232/RS-485 interface) on the Wincon-8000 embedded controller (see figure). The information in this section is organized as follows:

- COM1 Port
- COM2 Port
- COM3 Port

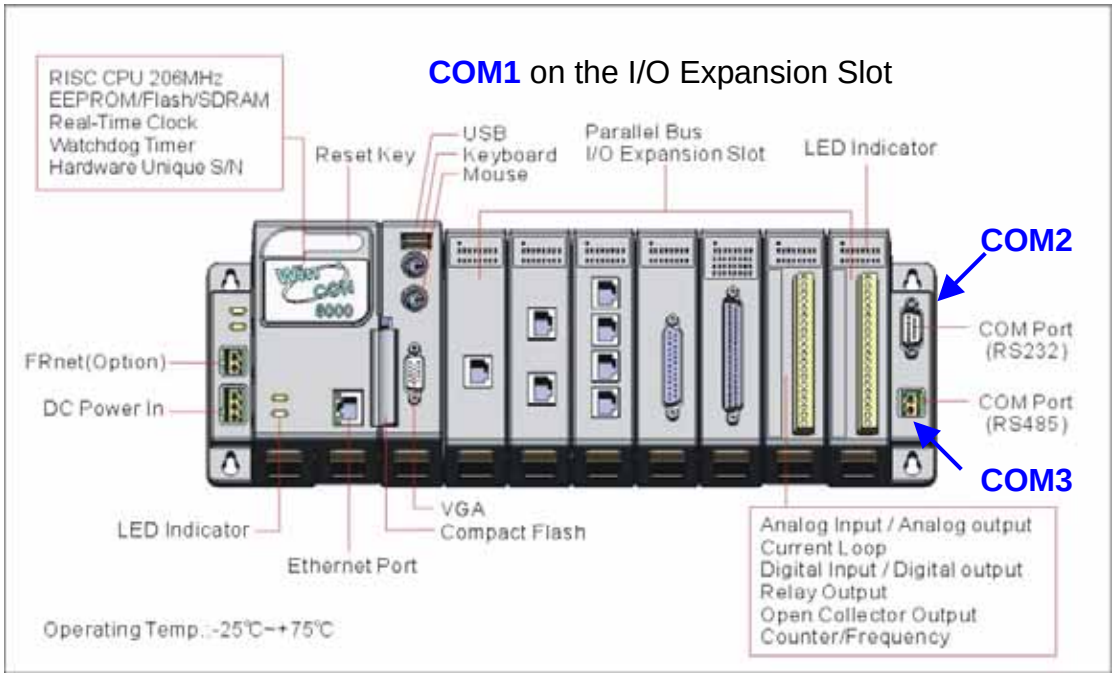


Fig. 3-1

3.1 Serial Port

COM1 Port

The COM1 port is located on the Wincon-8000 I/O expansion slot. This port is used to connect the I-87K series module plugged into the Wincon-8000 embedded controller. Users must use the serial command to control the I-87K series module. For controlling the I-87K module, you must input the Com-port parameters and call the

Open_Com function to open the com1 port based on the appropriate settings. Finally, the user can call the ChangeSlotTo87K(slot) function in order to change the control slot. This is like the serial address, you can then send out the control commands to the I/O module, which is plugged into this slot, so that the module's serial address for its slot is 0. A detailed example is provided below:

For Example:

[C#]

```
byte slot=1;
Wcon.i87K_Open_COM();
Wcon.System.ChangeSlotTo87K(slot);
Wcon.i87055.DO_8(slot,0xff);
Wcon.i87K_Close_COM();
```

[Visual Basic]

```
Wcon.i87K_Open_COM()
Dim slot As Byte = 1
Wcon.i87K_Open_COM()
Wcon.System.ChangeSlotTo87K(slot)
Wcon.i87055.DO_8(slot, &HFF)
Wcon.i87K_Close_COM()
```

COM2 Port

This COM2 port is located on the right-upper corner on the Wincon-8000. It is a standard RS-232 serial port, and it provides TXD, RXD, RTS, CTS, GND, non-isolated and a maximum speed of 115.2K bps. It can also connect to the I-7520 module in order to provide a general RS-485 communication. The COM2 port can also connect to a wireless modem so that it can be controlled from a remote device. The application example and code is demonstrated below (as a nomal COM PORT).

[C#]

```
int slot=1;
byte port=2;  uint baudrate=9600;    char data=8;
char parity=0;  char stopbit=1;
Dcon.UART.Open_Com(port, baudrate, data,parity, stopbit);
Dcon.UART.Send_Cmd(...);
```

```
Dcon.UART.Send_Receive_Cmd(...);
Dcon.UART.Close_Com(port);
```

[Visual Basic]

```
Dim slot As Integer = 1
Dim port As Byte = 2
Dim baudrate As UInt32 = Convert.ToUInt32(9600)
Dim data As Byte = 8
Dim parity As Byte = 0
Dim stopbit As Byte = 1
Dcon.UART.Open_Com(port, baudrate, data, parity, stopbit)
Dcon.UART.Send_Cmd(, , , , )
Dcon.UART.Send_Receive_Cmd(, , , , )
Dcon.UART.Close_Com(port)
```

Fig. 3-2

COM3 Port

This COM3 port provides RS-485 serial communication (DATA+ and DATA-) and is located on bottom-right corner on the Wincon-8000. You can connect to the RS-485

device with modules like the I-7000, I-87K and I-8000 serial modules, via this port. That is, you can control the ICP DAS I-7000/I-87K/I-8000 series modules directly from this port with any converter. ICP DAS will provide very easy to use functions with a DLL driver and then you can easily handle the I-7000/I-87K/I-8000 series modules just like the ActiveX tool. The example of the program code demo (as a nomal COM PORT) same as COM2 PORT demo.

3.2 DLL Architecture of the Serial Port

The UARTCE.DLL, I7000CE.DLL, and DCONCE.DLL are dynamic link library (DLL) files that are designed for applications that run on the Wincon-8000 main controller unit and can control the remote modules (I-7000/I-8000/I-87K series modules) through a serial port using Windows CE.NET. The user can apply them to develop their own applications with many development tools such as eMbedded Visual C++, Microsoft Visual Studio 2003 C#.NET, and Visual BASIC.NET. For your convenience, there are many demo programs are provided for EVC++, C#.NET and VB.NET. Based on the demo programs provided, users can easily understand how to use the functions and develop their own applications quickly.

The relationships among the UARTCE.DLL, I7000CE.DLL, DCONCE.DLL and the user's applications are depicted as follows:

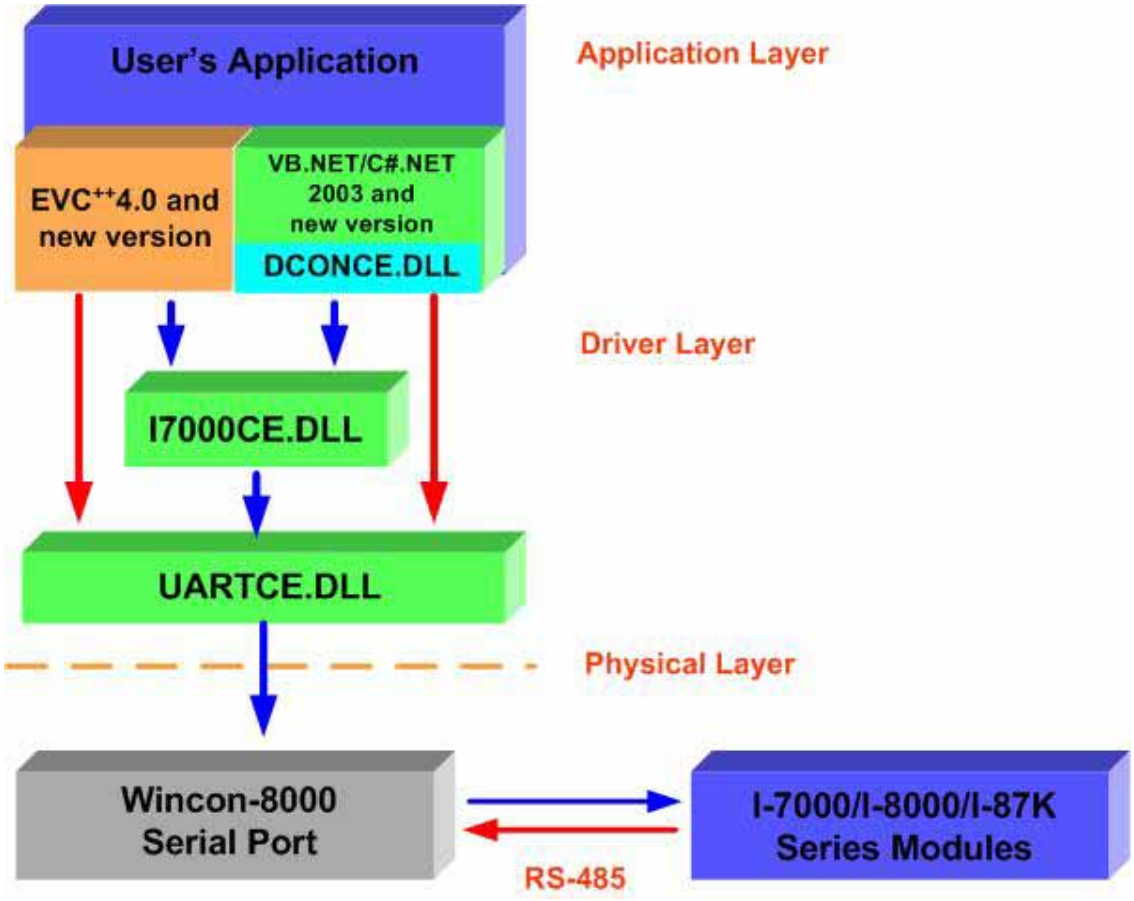


Fig. 3-3

The DLL functions in **DCONCE**_DLL are designed for WinCE.NET and can be utilized by C# and VB.NET. The main characteristics of the Wincon_DLL developed for applications in C#.NET and VB.NET development environments, are that they are very simple to use just like the applications found in the ActiveX(OCX) controls. Users can easily find the methods they need for their applications from the descriptions given in the following sections and in the demo programs developed by ICP DAS.

3.3 WinCon Serial Port Applications in VB.NET and C#.NET

The DCONCE_DLL is a powerful library tool designed specifically for use with VB.NET or C#.NET development environments of user applications within

Wincon-8000 embedded controllers. The DCONCE library contains every module function. Before you use the “Dcon” keyword in program design, you must add the DCONCE.dll into the reference list of your application. Please follow the below procedure to import DCONCE.dll into your project reference:

1. Open the "**Add Reference**" dialog by choosing **Project→Add Reference**(see Figure 3-4)

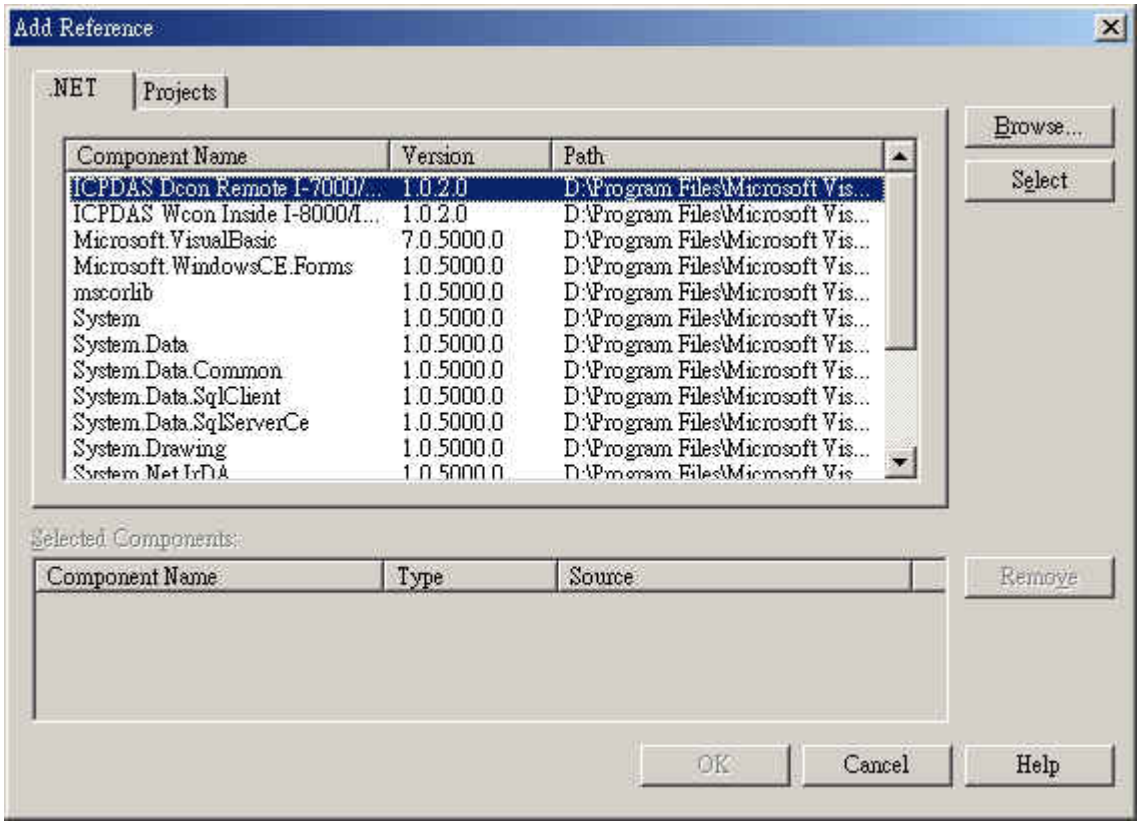


Fig.3-4

2. Select the “**ICP DAS Dcon Remote I-7000/I-8000/I-87K series modules for Wincon-8000**” from the list box, and click the “Select” button.
3. If you use the VB.NET development tool within the list box, select the “**mscorlib**” component and click the Select button (the component “**mscorlib**” must appear in the Selected Components area). Note: If you are using the C#.NET development tool, “**MSCorLib**” is the default project reference DLL. Therefore, the user does not need to add this to their project reference.

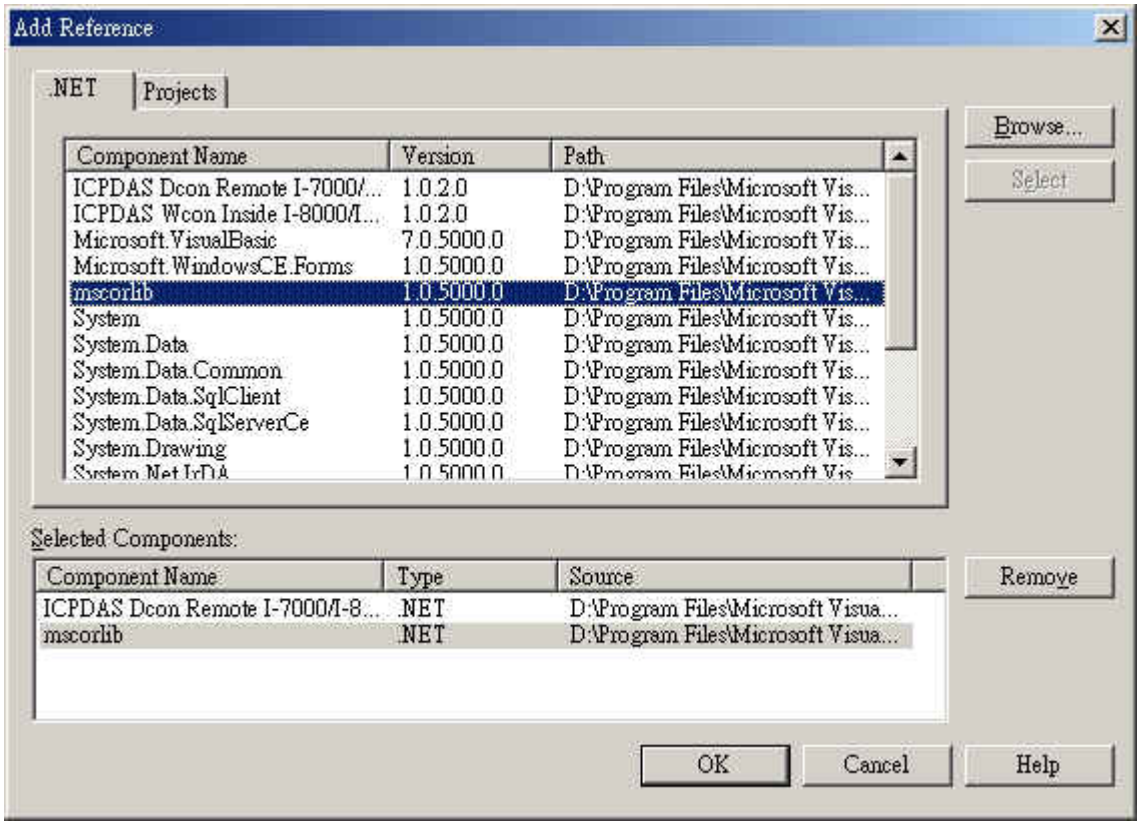


Fig. 3-5

4. Click OK to close the "**Add Reference**" dialog.

Refer to **WinCON Getting Started manual** section 5.2 of this manual for the "How to" on creating a new project under Visual studio VB.NET/C#.NET. For more detailed information relating to all the DCONCE functions, refer to section 4 of the user manual.

3.3.1 Example List for the Reference of User Program Design

The main idea on how to implement your first demo program is depicted in the above section. For the easy design of different applications, several demo programs are provided for user reference which can be applied on the Microsoft Visual Studio .Net 2003 Visual BASIC and Wincon-8000 application platforms. These demo

programs are listed as below.

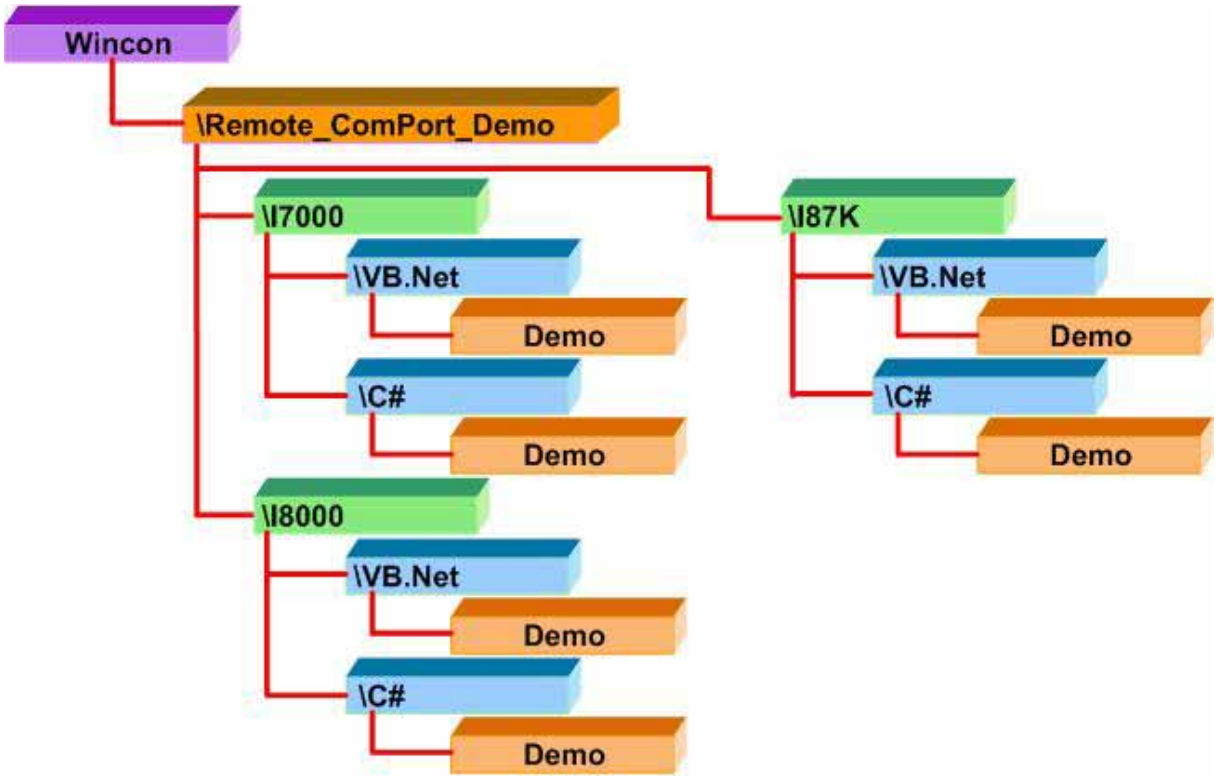


Fig. 3-6

Remote I7000 Demo for VB.NET/C#.NET:

This demo program can be applied to remote to either of these I-7060, I-7012 and I-7021 modules within the RS-485 network. This demo program can connect through these modules via the RS_485 network to the Wincon-8000 embedded controller (see figure 3-7). Users need to set their COM port configurations to connect to their remote modules, and set the RS-485 address for each module. Click the “Open COM” button to open the WinCon-8000 embedded controller COM port. Check the DO0~DO3 checkbox to output a digital value from the I-7060 module. Type in a 0~10 float value and click the “Analog Output I-7021” button to output an analog value for the I-7021 module. Click the “Auto Scan DI/AI Enable” button to read the digital input value for the I-7060 module and the analog input value for the I-7012 module.

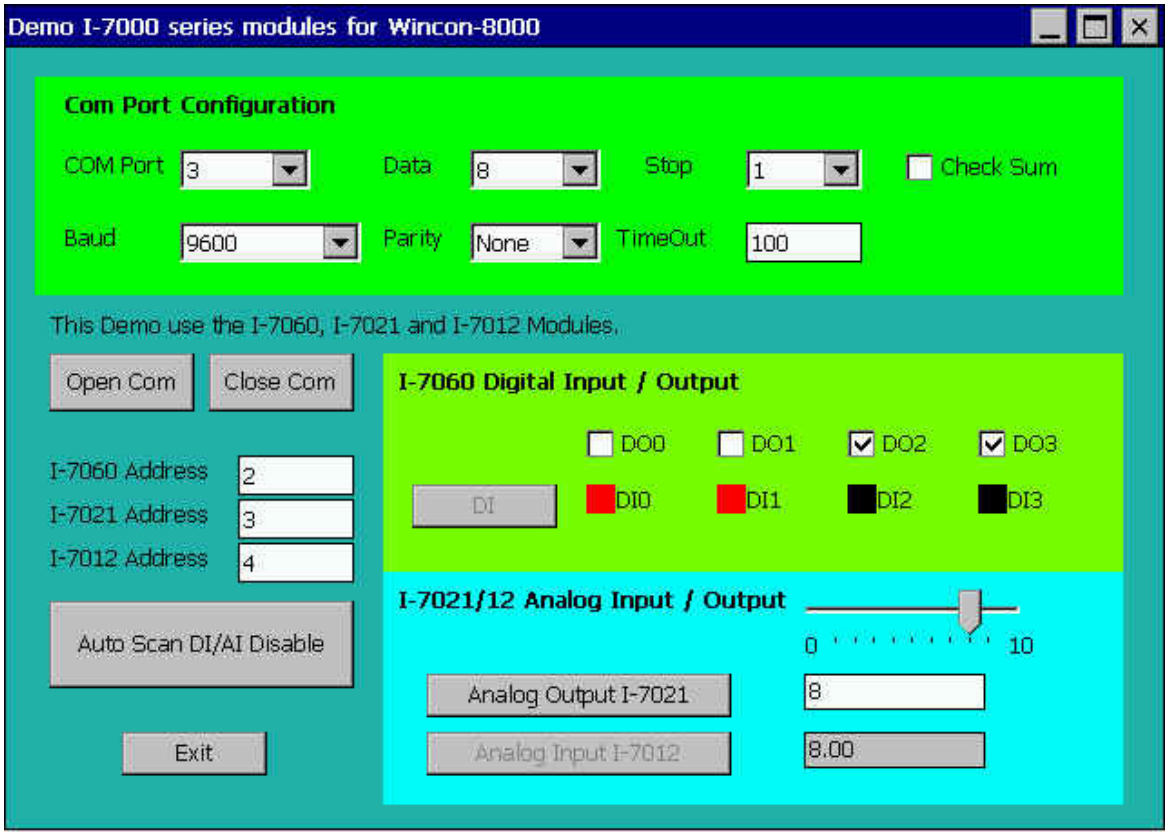


Fig. 3-7

3.4 WinCon Serial.DLL

The Serial.DLL is a powerful library tool designed specifically for use with VB.NET or C#.NET development environments of user applications within Wincon-8000 embedded controllers. The Serial library contains every COM function. Before you use the “Serial” keyword in program design, you must add the Serial.dll into the reference list of your application. Please follow the 3.3(DCONCE.DLL) procedure to import Serial.dll into your project reference.

4. DCONCE.DLL and Serial.DLL

In this section we will focus on examples for the description and application of the control functions on the remote I-7000/I-8000/I-87k series modules for use on Wincon-8000. The functions on the DCONCE.DLL can be classified into 10 groups as depicted as below: (DCONCE.DLL V1.0.2.2 , UARTCE.DLL V2.0.5 Serial.DLL V 1.0.1.0)

1. UART Serial Port Command and module Config Functions;
2. Digital Input Functions;
3. Digital Output Functions;
4. Analog Input Functions;
5. Analog Output Functions;
6. Module Alarm Functions;
7. Counter Functions;
8. Strain Gauge Functions;
9. Dual Watchdog Functions;
10. Error Code Table.

For the application of visual studio VB.NET and C#.NET, the namespace for DCONCE.DLL is **Dcon** and should not be changed during the development of applications. However, the DCONCE.DLL ModuleName should be replaced by a practical module name, one which will actually be applied in use of the control remote I-7000/I-8000/I-87K series modules. For example: i7017, i7021, i7060, i8053, i8057, i87017, i87024…etc. Please refer to section 4.2, for more detailed information on how to use this driver (Users will find it very similar to the application of the ActiveX component).

In the Digital Input/Output function name DO_## (DI_##) (“##” represents the channel number of the DI/O module. For example: If you use 7060 modules, please apply the DO_4 function for digital output.)

4.1 Serial Port Command and module Config Functions

■ Open_Com (for Dcon and Serial)

Description:

This function is used to configure and open a COM port for service. The COM port must be called up once before users can begin sending/receiving commands through it.

Syntax:

[Visual Basic, C#]
ushort Dcon[Serial].UART.Open_Com(byte cPort, uint dwBaudrate, byte cData, byte cParity, byte cStop)

Parameter:

cPort : [Input] 2=COM2, 3=COM3

dwBaudrate : [Input] 300/600/1200/1800/2400/4800/7200/9600/19200/38400/57600/115200

cData : [Input] 5/6/7/8 data bit

cParity : [Input] 0=None Parity, 1=Odd Parity, 2=Even Parity, 3=Mark Parity, 4=Space Parity

cStop : [Input] 0=1-stop, 1=1.5-stp, 2=2-stop

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
ushort result;
result=Dcon.UART.Open_Com( 3, 9600, 8, 0, 0);
// If result = 0 the opening COM3 port success.
```

[Visual Basic]

```
Dim result As UInt16
result = Dcon.UART.Open_Com(3, Convert.ToUInt32(9600), 8, 0, 0)
' If result = 0 the opening COM3 port success.
```

Remark:

■ Close_Com (for Dcon and Serial)

Description:

This function closes and releases COM port resources from the computer. It must be called on before exiting the application program. The Open_Com function will return error message if the program exit without calling Close_Com function.

Syntax:

[Visual Basic, C#]
bool Dcon[Serial].UART.Close_Com(byte cPort)

Parameter:

cPort : [Input] 2=COM2, 3=COM3

Return Value:

0 : NO_ERROR

Others : Error code

Example:

[C#]

```
bool result;
result=Dcon.UART.Close_Com( 3 );
// If result = 0 the closing COM3 port success.
```

[Visual Basic]

```
Dim result As Boolean
result = Dcon.UART.Close_Com(3)
' If result = 0 the closing COM3 port success.
```

Remark:

■ Send_Receive_Cmd (for Dcon and Serial)

Description:

This function sends a DCON's command string to the RS-232(RS-485) Network and then will receive a response from it. If wCheckSum=1, then this function will automatically add the two checksum bytes into the command string plus check the checksum status when it receives a response from the modules. Note that [0x0D] is added to the end of each sending string which means the termination for every command. This Send_Receive_Cmd is not a multi-task DLL.

Syntax:

[Visual Basic, C#]

```
ushort Dcon[Serial].UART.Send_Receive_Cmd(byte cPort, byte[] szCmd,
        byte[] szResult, ushort wTimeOut, ushort wChksum, ref ushort wT)
```

Parameter:

cPort :	[Input] 2=COM2, 3=COM3
szCmd:	[Input] Sending command string
szResult:	[Output] Receiving the response string from the modules
wTimeOut:	[Input] Communicating timeout setting, time unit = 1ms
wChksum:	[Input] 0=DISABLE, 1=ENABLE
wT:	[Output] Total time of send/receive interval, unit = 1 ms

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
ushort result;
result=Dcon.UART.Open_Com(3, 9600, 8, 0, 0);
byte[] s=new byte[50];
byte[] r=new byte[50];
ushort wt=0;
string SS="$03M"; // Use string-command to control I-7000/I-87K modules
s=System.Text.Encoding.ASCII.GetBytes(SS);
```

```
result=Dcon.UART.Send_Receive_Cmd(1, s, r, 100, 0,ref wt);
string RR=System.Text.Encoding.ASCII.GetString(r,0,r.Length);
//This RR string is to receive data.
```

[Visual Basic]

```
Dim result As UInt16
result = Dcon.UART.Open_Com(3, Convert.ToUInt32(9600), 8, 0, 0)
Dim s(100) As Byte
Dim r(100) As Byte
Dim wt As UInt16
Dim SS As String
SS = "$03M" 'Use string-command to control I-7000/I-87K modules
s = System.Text.Encoding.ASCII.GetBytes(SS)
result = Dcon.UART.Send_Receive_Cmd(1, s, r, Convert.ToUInt16(100), Convert.ToUInt16(0), wt)
Dim RR As String
RR = System.Text.Encoding.ASCII.GetString(r, 0, r.Length)
'The RR string is to receive data.
```

Remark:

■ Send_Cmd (for Dcon and Serial)

Description:

This function only sends a DCON's command string to the DCON series module. If wCheckSum=1, then this function will automatically add the two checksum bytes into the command string. Then [0x0D] is added to the end of each sending string which means the termination of the command (szCmd). Note that the function Send_Cmd is a multi-task and multi-thread DLL. Furthermore this command string cannot include a space character in the command string. If it does, then the space character will cause the command string function to stop. For example: if "$01M 02 03" is the user's command string then the actual command sent out is "$01M".

Syntax:

[Visual Basic, C#]

ushort Dcon[Serial].UART.Send_Cmd(byte cPort, byte[] szCmd, ushort wChksum)

Parameter:

cPort :	[Input] 2=COM2, 3=COM3
szCmd:	[Input] Sending command string(Terminated with "0")
wChksum:	[Input] 0=DISABLE, 1=ENABLE

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
ushort result;
result=Dcon.UART.Open_Com(3, 9600, 8, 0, 0);
byte[] s=new byte[50];
string SS="$03M";  // Send sting-command from the Com Port out.
s=System.Text.Encoding.ASCII.GetBytes(SS);
result=Dcon.UART.Send_Cmd(1, s, 0);
```

[Visual Basic]

```
Dim result As UInt16
```

```
result = Dcon.UART.Open_Com(3, Convert.ToUInt32(9600), 8, 0, 0)
Dim s(100) As Byte
Dim SS As String
SS = "$03M" 'Send sting-command from the Com Port out.
s = System.Text.Encoding.ASCII.GetBytes(SS)
result = Dcon.UART.Send_Cmd(1, s, Convert.ToUInt16(0))
```

Remark:

■ Receive_Cmd (for Dcon and Serial)

Description:

Users can utilize this function to obtain a response string from the modules in the RS-485 Network. this function provides a response string without the last byte [0x0D].

Syntax:

[Visual Basic, C#]

ushort Dcon[Serial].UART.Receive_Cmd(byte cPort, byte[] szResult, ushort wTimeOut,
ushort wChksum, ref ushort wT)

Parameter:

cPort :	[Input] 2=COM2, 3=COM3
szResult:	[Output] Receiving the response string from the modules
wTimeOut:	[Input] Communicating timeout setting, time unit = 1ms
wChksum:	[Input] 0=DISABLE, 1=ENABLE
wT:	[Output] Total time of send/receive interval, unit = 1 ms

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
ushort result;
result=Dcon.UART.Open_Com(3, 9600, 8, 0, 0);
byte[] r=new byte[50];
ushort wt=0;
result=Dcon.UART.Receive_Cmd(1, r, 100, 0, ref wt);
string RR=System.Text.Encoding.ASCII.GetString(r, 0, r.Length);
//This RR string is to receive data.
```

[Visual Basic]

```
Dim result As UInt16
result = Dcon.UART.Open_Com(3, Convert.ToUInt32(9600), 8, 0, 0)
Dim r(100) As Byte
Dim wt As UInt16
```

```
result = Dcon.UART.Receive_Cmd(1, r, Convert.ToUInt16(100), Convert.ToUInt16(0), wt)
Dim RR As String
RR = System.Text.Encoding.ASCII.GetString(r, 0, r.Length)
'The RR string is to receive data.
```

Remark:

■ Send_Binary (for Dcon and Serial)

Description:

Sends out a command string via fix length. It is controlled by the parameter “iLen”. The difference between this function and the Send_cmd function is that Send_Binary terminates the sending process by the string length “iLen” instead of the character "CR"(Carry return). Therefore, this function can send out a command string with or without a null character under the consideration of the command length. However, because this function operates without any error checking mechanisms (Checksum, CRC, LRC... etc.), the user would have to add in error checking information into the raw data manually, if a communication checking system was required. Note that this function is usually applied to communicate with the other device, but not for ICPDAS DCON (I-7000/8000/87K) series modules.

Syntax:

[Visual Basic, C#]

ushort Dcon[Serial].UART.Send_Binary(byte cPort, byte[] szCmd, short iLen)

Parameter:

cPort :	[Input] 2=COM2, 3=COM3
szCmd:	[Input] Sending command string(Terminated with “0”)
iLen:	[Input] The length of command string.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
ushort result;
result=Dcon.UART.Open_Com(2, 9600, 8, 0, 0);
byte[] s=new byte[50];
string SS="Hello World";  // Send sting from the Com Port out.
s=System.Text.Encoding.ASCII.GetBytes(SS);
result=Dcon.UART.Send_Binary(1, s,Convert.ToInt16( SS.Length));
```

[Visual Basic]

```
Dim result As UInt16
result = Dcon.UART.Open_Com(3, Convert.ToUInt32(9600), 8, 0, 0)
Dim s(100) As Byte
Dim SS As String
SS = "Hello World" 'Send sting from the Com Port out.
s = System.Text.Encoding.ASCII.GetBytes(SS)
result = Dcon.UART.Send_Binary(1, s, SS.Length)
```

Remark:

■ Receive_Binary (for Dcon and Serial)

Description:

This function is applied to receive a fix length response. The length of the receiving response is controlled by the parameter “iLen”. The difference between this function and the Receive_cmd function is that Receive_Binary function terminates the receiving process using the string length “iLen” instead of the character "CR"(Carry return). Therefore, this function can be used to receive response string data with or without a null character under the consideration of the receiving length. Besides, because this function operates without any error checking mechanisms (checksum, CRC, LRC... etc.), the user has to remove their error checking information from the raw data themselves if a communication checking system must be applied. Note that this function is usually applied to communicate with the other devices, but not on ICPDAS DCON (I-7000/8000/87K) series modules.

Syntax:

[Visual Basic, C#]

```
ushort Dcon[Serial].UART.Receive_Binary(byte cPort, byte[] szResult,
                                        ushort wTimeOut, ushort wLen, ref ushort wT)
```

Parameter:

cPort :	[Input] 2=COM2, 3=COM3
szResult:	[Output] The receiving string from the module
wTimeOut:	[Input] Communicating timeout setting, time unit = 1ms
wLen:	[Input] The length of result string.
wT:	[Output] Total time of receiving interval, unit = 1 ms.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
ushort result;
result=Dcon.UART.Open_Com(3, 9600, 8, 0, 0);
```

```
byte[] r=new byte[50];
ushort wt=0;
ushort len=10;
result=Dcon.UART.Receive_Binary(1, r, 100, len, ref wt);
string RR=System.Text.Encoding.ASCII.GetString(r, 0, len);
//This RR string is to receive data.
```

[Visual Basic]

```
Dim result As UInt16
result = Dcon.UART.Open_Com(3, Convert.ToUInt32(9600), 8, 0, 0)
Dim r(100) As Byte
Dim wt As UInt16
Dim len As Integer
len = 10
result = Dcon.UART.Receive_Binary(1, r, Convert.ToUInt16(100), Convert.ToUInt16(100), wt)
Dim RR As String
RR = System.Text.Encoding.ASCII.GetString(r, 0, len)
'The RR string is to receive data.
```

Remark:

■ Get_Com_Status (for Dcon and Serial)

Description:

This function can obtain the COM Port's status. If the return value is "0" (false), it means that "The COM Port is not in use!", but if the return value is "1" (true), it means that " The COM Port is in use!"

Syntax:

[Visual Basic, C#]
ushort Dcon[Serial].UART.Get_Com_Status(byte cPort)

Parameter:

cPort : [Input] 2=COM2, 3=COM3

Return Value:

0: COM port is not in used.

1: COM port is in used.

Example:

[C#]

```
ushort result;
result=Dcon.UART.Get_Com_Status( 3);
// If result = 0 COM port is not in used.
```

[Visual Basic]

```
Dim result As UInt16
result = Dcon.UART.Get_Com_Status(3)
' If result = 0 COM port is not in used.
```

Remark:

■ Change_BaudRate (for Dcon and Serial)

Description:

This function serial communication Baudrate settings once the COM port has been opened.

Syntax:

[Visual Basic, C#]
ushort Dcon[Serial].UART.Change_Baudrate(byte cPort, uint dwBaudrate)

Parameter:

cPort : [Input] 2=COM2, 3=COM3

dwBaudrate : [Input] 50/75/110/134.5/150/300/600/1200/1800/2400/4800/7200/9600/19200/38400/57600/115200

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
ushort result;
result=Dcon.UART.Change_Baudrate ( 3, 19200);
// The COM3 port baudrate has changed 19200.
```

[Visual Basic]

```
Dim result As UInt16
result = Dcon.UART.Change_Baudrate(3, Convert.ToUInt32(19200))
' The COM3 port baudrate has changed 19200.
```

Remark:

■ Change_Config (for Dcon and Serial)

Description:

This function can only be used to change COM port configurations after the COM port has been opened.

Syntax:

[Visual Basic, C#]

ushort Dcon[Serial].UART.Change_Config(byte cPort, uint dwBaudrate, byte cData,
byte cParity, byte cStop)

Parameter:

cPort : [Input] 2=COM2, 3=COM3

dwBaudrate : [Input] 50/75/110/134.5/150/300/600/1200/1800/2400/4800/7200/9600/19200/38400/57600/115200

cData : [Input] 5/6/7/8 data bit

cParity: [Input] 0=None Parity, 1=Odd Parity, 2=Even Parity, 3=Mark Parity, 4=Space Parity

cStop: [Input] 0=1-stop, 1=1.5-stp, 2=2-stop

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
ushort result;
uint baudrates=19200;
byte databit=8; byte paritybit=0;
byte stopbit=1;  //stop bit=1.5
result=Dcon.UART.Change_Config( 3, baudrates, databit, paritybit, stopbit);
//The COM3 port communication configuration has changed when result is 0.
```

[Visual Basic]

```
Dim result As UInt16
```

```
Dim baudrates As UInt32
baudrates = Convert.ToUInt32(19200)
Dim databit As Byte = 8
Dim paritybit As Byte = 0
Dim stopbit As Byte = 1   'stop bit=1.5
result = Dcon.UART.Change_Config(3, baudrates, databit, paritybit, stopbit)
'The COM3 port communication configuration has changed when result is 0.
```

Remark:

■ Get_Uart_Version (for Dcon and Serial)

Description:

Users can obtain UARTCE.DLL version information by using this function.

Syntax:

[Visual Basic, C#]
ushort Dcon[Serial].UART.Get_Uart_Version()

Parameter:

None

Return Value:

Return the version of UARTCE.DLL with hexadecimal format.

Example:

[C#]

```
ushort ver;
ver=Dcon.UART.Get_Uart_Version();
// If the return value is 0X205, it means “the version of UARTCE.DLL" is 2.0.5
```

[Visual Basic]

```
Dim ver As UInt16
ver = Dcon.UART.Get_Uart_Version()
' If the return value is 0X205, it means “the version of UARTCE.DLL" is 2.0.5
```

Remark:

■ DataSizeInCom (for Serial)

Description:

Get the how many data existed on the input buffer of COM port.

Syntax:

[Visual Basic, C#]

```
uint Serial.UART.DataSizeInCom(byte cPort)
```

Parameter:

cPort : [Input] 2=COM2, 3=COM3

Return Value:

Return the data size In com port buffer.

Example:

[C#]

```
byte port = 3;
uint baudrate= 9600;
byte data  = 8 ; byte parity  = 4;
byte stopbit  = 0;
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit);
uint DAin  = Serial.UART.DataSizeInCom(port);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim baudrate As UInt32 = Convert.ToUInt32(9600)
Dim data As Byte = 8 : Dim parity As Byte = 4
Dim stopbit As Byte = 0
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit)
Dim DAin As UInt32 = Serial.UART.DataSizeInCom(port)
```

Remark:

■ DataSizeOutCom (for Serial)

Description:

Get the how many data existed on the output buffer of COM port.

Syntax:

[Visual Basic, C#]

```
uint Serial.UART.DataSizeOutCom(byte cPort)
```

Parameter:

cPort : [Input] 2=COM2, 3=COM3

Return Value:

Return the data size In com port output buffer.

Example:

[C#]

```
byte port = 3;
uint baudrate= 9600;
byte data  = 8 ; byte parity  = 4;
byte stopbit  = 0;
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit);
uint DA  = Serial.UART.DataSizeOutCom(port);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim baudrate As UInt32 = Convert.ToUInt32(9600)
Dim data As Byte = 8 : Dim parity As Byte = 4
Dim stopbit As Byte = 0
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit)
Dim DA As UInt32 = Serial.UART.DataSizeOutCom(port)
```

Remark:

■ Set_FlowControl (for Serial)

Description:

This function is used to set DCB settings which are related to the flow control (Software and Hardware).

Syntax:

[Visual Basic, C#]
ushort Serial.UART.Set_FlowControl (byte cPort,int DCB_Member,int mode)

Parameter:

cPort : [Input] 2=COM2, 3=COM3

DCB_Member: 0 CtS (fOutxCtsFlow)
1 Rts (fRtsControl)
2 Dsr (fOutxDsrFlow)
3 Dtr (fDtrControl)
4 Tx (fOutx :Tx XON/XOFF flow control)
5 Rx (fInx :Rx XON/XOFF flow control)

Mode:

CTS : 1: Enable CTS to monitored for output flow control, 0:Disable CTS flow control

RTS : 1: Enable RTS flow control, 0:Disable RTS flow control, 2:RTS_CONTROL_HANDSHAKE, 3:RTS_CONTROL_TOGGLE

DSR : 1: Enable DSR flow control / 0:Disable DSR flow control

DTR : 1: Enable DTR flow control, 0:Disable DTR flow control, 2:DTR_CONTROL_HANDSHAKE

TX : 1: Enable TX XON/XOFF, 0: Disable TX XON/XOFF

RX : 1: Enable RX XON/XOFF, 0: Disable RX XON/XOFF

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port = 3;
```

```
uint baudrate= 9600;
byte data  = 8 ; byte parity  = 4;
byte stopbit  = 0;
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit);
ushort DA=Serial.UART.Set_FlowControl(port,3,1);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim baudrate As UInt32 = Convert.ToUInt32(9600)
Dim data As Byte = 8 : Dim parity As Byte = 4
Dim stopbit As Byte = 0
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit)
Dim DA As UInt16 = Serial.UART.Set_FlowControl(port, 3, 1)
```

Remark:

Software flow control:

Use XOFF and XON characters to control the transmission and reception of data.

Hareware flow control:

Use control lines of the serial cable to control whether sending or receiving is enabled.

■ Get_FlowControl (for Serial)

Description:

This function is used to get DCB settings which are related to the flow control (Software and Hardware).

Syntax:

[Visual Basic, C#]

ushort Serial.UART.Get_FlowControl (byte cPort,int DCB_Member)

Parameter:

cPort : [Input] 2=COM2, 3=COM3

DCB_Member: 0 CtS (fOutxCtsFlow)
1 Rts (fRtsControl)
2 Dsr (fOutxDsrFlow)
3 Dtr (fDtrControl)
4 Tx (fOutx :Tx XON/XOFF flow control)
5 Rx (fInx :Rx XON/XOFF flow control)

Return Value:

CTS : 1: Enable CTS to monitored for output flow control, 0:Disable CTS flow control

RTS : 1: Enable RTS flow control, 0:Disable RTS flow control, 2:RTS_CONTROL_HANDSHAKE, 3:RTS_CONTROL_TOGGLE

DSR : 1: Enable DSR flow control / 0:Disable DSR flow control

DTR : 1: Enable DTR flow control, 0:Disable DTR flow control, 2:DTR_CONTROL_HANDSHAKE

TX : 1: Enable TX XON/XOFF, 0: Disable TX XON/XOFF

RX : 1: Enable RX XON/XOFF, 0: Disable RX XON/XOFF

Example:

[C#]

```
byte port = 3;
uint baudrate= 9600;
byte data  = 8 ; byte parity  = 4;
```

```
byte stopbit  = 0;
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit);
ushort DA=Serial.UART.Get_FlowControl(port,3);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim baudrate As UInt32 = Convert.ToUInt32(9600)
Dim data As Byte = 8 : Dim parity As Byte = 4
Dim stopbit As Byte = 0
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit)
Dim DA As UInt16 = Serial.UART.Get_FlowControl(port, 3)
```

Remark:

Software flow control:

Use XOFF and XON characters to control the transmission and reception of data.

Hareware flow control:

Use control lines of the serial cable to control whether sending or receiving is enabled.

■ SetLineStaus (for Serial)

Description:

This function is used to set the COM line status(DTR,RTS)

Syntax:

[Visual Basic, C#]

ushort Serial.UART.SetLineStatus(int cPort,int pin,int mode)

Parameter:

cPort :	[Input] 2=COM2, 3=COM3
pin:	0: 1: DTR 2: RTS 3: DTR+RTS
mode	0:Disable 1:Enable 2:HandShake

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port = 3;
uint baudrate= 9600;
byte data  = 8 ;
byte parity  = 4;
byte stopbit  = 0;
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit);
ushort DA= Serial.UART.SetLineStatus(port, 1, 0);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim baudrate As UInt32 = Convert.ToUInt32(9600)
Dim data As Byte = 8
```

```
Dim parity As Byte = 4
Dim stopbit As Byte = 0
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit)
Dim DA As UInt16 = Serial.UART.SetLineStatus(port, 1, 0)
```

Remark:

■ GetLineStaus (for Serial)

Description:

This function is used to get the COM line status(DTR,RTS)

Syntax:

[Visual Basic, C#]

ushort Serial.UART.GetLineStatus(int cPort,int pin)

Parameter:

cPort : [Input] 2=COM2, 3=COM3

pin: 0:
1: DTR
2: RTS
3: DTR+RTS

Return Value:

1:ON
0:OFF

Example:

[C#]

```
byte port = 3;
uint baudrate= 9600;
byte data  = 8 ; byte parity  = 4;
byte stopbit  = 0;
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit);
ushort DA= Serial.UART.GetLineStatus(port, 1);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim baudrate As UInt32 = Convert.ToUInt32(9600)
Dim data As Byte = 8 : Dim parity As Byte = 4
Dim stopbit As Byte = 0
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit)
Dim DA As UInt16 = Serial.UART.GetLineStatus(port, 1)
```

Remark:

■ ReadComn (for Serial)

Description:

This function is applied to receive the multi-bytes response from COM port

Syntax:

[Visual Basic, C#]

ushort Serial.UART. ReadComn(byte cPort, char szResult[],int wmaxLen, ref int *wT)

Parameter:

cPort : [Input] 2=COM2, 3=COM3

szResult: [Output] The receiving string from the module

wmaxLen: [Input] The max length of result string.

wT: [Output] Total time of receiving interval, unit = 1 ms.

Return Value:

The number of bytes actually read

Example:

[C#]

```
byte port = 3;
uint baudrate= 9600;
byte data  = 8 ; byte parity  = 4;
byte stopbit  = 0;
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit);
ushort wT=0;
byte[] r=new byte[100];
ushort DA= Serial.UART.ReadComn(port, r, 100,ref wT);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim baudrate As UInt32 = Convert.ToUInt32(9600)
Dim data As Byte = 8 : Dim parity As Byte = 4
Dim stopbit As Byte = 0
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit)
Dim wT As UInt16 : Dim r(100) As Byte
Dim DA As UInt16 = Serial.UART.ReadComn(port, r, Convert.ToUInt16(100), wT)
```

Remark:

■ Send_Cmd_WithChar (for Serial)

Description:

Send command and then receive datas with the last character It is similar to the Send_Receive_Cmd except cEndChar paramter

Syntax:

[Visual Basic, C#]

ushort Serial.UART.Send_Cmd_WithChar (byte cPort, byte[] szCmd, ushort wChksum, byte cEndChar)

Parameter:

cPort :	[Input] 2=COM2, 3=COM3
szCmd:	[Input] Sending command string
wChksum:	[Input] 0=DISABLE, 1=ENABLE
cEndChar:	[Input] string end char

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port = 3;
uint baudrate= 9600;
byte data  = 8 ; byte parity  = 4;
byte stopbit  = 0;
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit);
byte[] s=new byte[50];
string SS="TEST" + "E" ;  // Send sting-command from the Com Port out.
s=System.Text.Encoding.ASCII.GetBytes(SS);
ushort DA= Serial.UART.Send_Cmd_WithChar(port,s,0, 69);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim baudrate As UInt32 = Convert.ToUInt32(9600)
Dim data As Byte = 8 : Dim parity As Byte = 4
Dim stopbit As Byte = 0
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit)
```

```
Dim wT As UInt16
Dim r(100) As Byte
Dim SS As String = "TEST" & vbCr  'Send sting-command from the Com Port out.
r = System.Text.Encoding.ASCII.GetBytes(SS)
Dim DA As UInt16 = Serial.UART.Send_Cmd_WithChar(port, r, Convert.ToUInt16(0), vbCr)
```

Remark:

■ Receive_Cmd_WithChar (for Serial)

Description:

Receive command with the last character It is similar to the Receive_Cmd_WithChar except cEndChar paramter

Syntax:

[Visual Basic, C#]

ushort Serial.UART.Receive_Cmd_WithChar (byte cPort, byte[] szResult, ushort wTimeOut,ushort wChksum, ref ushort wT, byte cEndChar)

Parameter:

cPort :	[Input] 2=COM2, 3=COM3
szResult:	[Output] Receiving the response string from the modules
wTimeOut:	[Input] Communicating timeout setting, time unit = 1ms
wChksum:	[Input] 0=DISABLE, 1=ENABLE
wT:	[Output] Total time of send/receive interval, unit = 1 ms
cEndChar:	[Input] string end char

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port = 3;
uint baudrate= 9600;
byte data  = 8 ; byte parity  = 4;
byte stopbit  = 0;
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit);
byte[] s=new byte[50];
ushort wT=0;
ushort DA= Serial.UART.Receive_Cmd_WithChar(port,s,1000,0,ref wT,69);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim baudrate As UInt32 = Convert.ToUInt32(9600)
```

```
    Dim data As Byte = 8 : Dim parity As Byte = 4
    Dim stopbit As Byte = 0
    Serial.UART.Open_Com(port, baudrate, data, parity, stopbit)
    Dim wT As UInt16
    Dim r(100) As Byte
    Dim DA As UInt16 = Serial.UART.Receive_Cmd_WithChar(port, r, Convert.ToUInt16(1000),
Convert.ToUInt16(0), wT, vbCr)
```

Remark:

■ Send_Receive_Cmd_WithChar (for Serial)

Description:

Send command and then receive datas with the last character It is similar to the Send_Receive_Cmd except cEndChar paramter

Syntax:

[Visual Basic, C#]

```
ushort Serial.UART.Send_Receive_Cmd_WithChar (byte cPort, byte[] szCmd,
        byte[] szResult, ushort wTimeOut, ushort wChksum, ref ushort wT, byte
cEndChar)
```

Parameter:

cPort :	[Input] 2=COM2, 3=COM3
szCmd:	[Input] Sending command string
szResult:	[Output] Receiving the response string from the modules
wTimeOut:	[Input] Communicating timeout setting, time unit = 1ms
wChksum:	[Input] 0=DISABLE, 1=ENABLE
wT:	[Output] Total time of send/receive interval, unit = 1 ms
cEndChar:	[Input] string end char

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
ushort result;
result=Dcon.UART.Open_Com(3, 9600, 8, 0, 0);
byte[] s=new byte[50];
byte[] r=new byte[50];
ushort wt=0;
string SS="TEST"+"E";
s=System.Text.Encoding.ASCII.GetBytes(SS);
result=Serial.UART.Send_Receive_Cmd_WithChar(1, s, r, 100, 0,ref wt,69);
string RR=System.Text.Encoding.ASCII.GetString(r,0,r.Length);
```

[Visual Basic]

```
Dim result As UInt16
result = Dcon.UART.Open_Com(3, Convert.ToUInt32(9600), 8, 0, 0)
Dim s(100) As Byte
Dim r(100) As Byte
Dim wt As UInt16
Dim SS As String
SS = "TEST" & "E"
s = System.Text.Encoding.ASCII.GetBytes(SS)
result = Serial.UART.Send_Receive_Cmd_WithChar(1, s, r, Convert.ToUInt16(100), Convert.ToUInt16(0),
wt, "E")
Dim RR As String
RR = System.Text.Encoding.ASCII.GetString(r, 0, r.Length)
```

Remark:

■ Set_Break (for Serial)

Description:

This function is used to send the Break signal for the COM port.

Syntax:

[Visual Basic, C#]
ushort Serial.UART. Set_Break(byte cPort,byte cAction)

Parameter:

cPort : [Input] 2=COM2, 3=COM3

cAction: [Input] 0=clear break, 1=set break

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
ushort result;
result=Dcon.UART.Get_Com_Status( 3);
// If result = 0 COM port is not in used.
```

[Visual Basic]

```
Dim port As Byte = 3
Dim baudrate As UInt32 = Convert.ToUInt32(9600)
Dim data As Byte = 8 : Dim parity As Byte = 4
Dim stopbit As Byte = 0
Serial.UART.Open_Com(port, baudrate, data, parity, stopbit)
Serial.UART.Set_Break(port, 1)
```

Remark:

■ ReadConfigStatus (for Dcon)

Description:

This function attains the module configuration status for I-7000 series modules. For more detailed information on these parameters, please refer to the user manual.

Syntax:

[Visual Basic, C#]

```
ushort Dcon. ReadConfigStatus( byte ComPort, byte address,
        ushort ModuleID, byte CheckSum, ushort TimeOut, out ushort Addr,
        out ushort RangCode, out ushort Baudrate, out ushort DataFormat )
```

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
ModuleID :	[Input] Module ID (for all modules)
CheckSum :	[Input] 0=checksum disable, 1=checksum enable
TimeOut :	[Input] Time out setting, normal=100, unit=ms.
Addr :	[Output] Module address.
RangCode :	[Output] Module Range Code.
Baudrate :	[Output] Module baudrate.
DataFormat :	[Output] Module data format.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;
byte addr_7017 = 0x03;
ushort moduleID = 0x7017;
byte checksum=0;
ushort timeout=100;
ushort addr;
ushort rangCode;      ushort baudrate;
ushort data;          ushort Ecode;
```

```
Ecode= Dcon.ReadConfigStatus(port, addr_7017, moduleID, checksum, timeout, out addr, out rangCode,
out baudrate, out data);
textBox1.Text = addr.ToString();
textBox2.Text = rangCode.ToString();
textBox3.Text = baudrate.ToString();
textBox4.Text = data.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7017 As Byte = &H3
Dim moduleID As UInt16 = Convert.ToUInt16(&H7017)
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim addr As UInt16
Dim rangCode As UInt16
Dim baudrate As UInt16
Dim data As UInt16
Dim Ecode As UInt16
Ecode = Dcon.ReadConfigStatus(port, addr_7017, moduleID, checksum, timeout, addr, rangCode,
baudrate, data)
TextBox1.Text = addr.ToString()
TextBox2.Text = rangCode.ToString()
textBox3.Text = baudrate.ToString()
textBox4.Text = data.ToString()
```

Remark:

4.2 Digital Input Functions

■ DI_##

Description:

Obtains the digital input values from the I-7000/I-8000/I-87K series modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.DI_##(byte ComPort, byte address, [byte slot,]
byte CheckSum, ushort TimeOut, out Data_Type DigitalValue)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

slot : [Input] Slot number; the I/O module installed in I-8000 main unit.

CheckSum : [Input] 0=checksum disable, 1=checksum enable

TimeOut : [Input] Time out setting, normal=100, unit=ms.

DigitalValue :[Output] ##-bit digital input data.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte address_7060=0x03;
byte checksum=0;
ushort timeout=100;
byte InTemp;
ushort Ecode;
Ecode=Dcon.i7060.DI_4(port, address_7060, checksum, timeout, out InTemp);
textBox1.Text = InTemp.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7060 As Byte = &H3
```

```
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim InTemp As Byte
Dim Ecode As UInt16
Ecode = Dcon.i7060.DI_4(port, address_7060, checksum, timeout, InTemp)
TextBox1.Text = InTemp.ToString()
```

Remark:

1.This “##” can be applied on number: 1, 4, 7, 8, 14, 16, 32.

2.This Data_Type can be applied on data type: byte, ushort, uint.

3.This function can be applied on module: i7011, i7012, i7014, i7016, i7041, i7044, i7050, i7052, i7053, i7060, i7063, i7065, i8042, i8054, i8055, i8063, i8040, i8051, i8052, i8053, i8058, i87051, i87052, i87053, i87054, i87055, i87058, i87063.

■ DigitalInLatch

Description:

This function obtains the latch values for the high and low latch modes on the Digital Input module.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.DigitalInLatch( byte ComPort, byte address,
    byte Low_High, byte CheckSum, ushort TimeOut, out ushort LatchValue)
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

Low_High : [Input] 0: low Latch mode, 1: high Latch mode.

CheckSum : [Input] 0=checksum disable, 1=checksum enable

TimeOut : [Input] Time out setting, normal=100, unit=ms.

LatchValue : [Output] Latch value.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;          byte address_7044=0x03;
byte checksum=0;      ushort timeout=100;
byte low=0;           byte InLatch;
ushort Ecode;
Ecode=Dcon.i7044.DigitalInLatch(port, address_7044, low, checksum, timeout,out InLatch);
textBox1.Text = InLatch.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7044 As Byte = &H3
```

```
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim low As Byte = 0
Dim InLatch As Byte
Dim Ecode As UInt16
Ecode = Dcon.i7044.DigitalInLatch(port, address_7044, low, checksum, timeout, InLatch)
TextBox1.Text = InLatch.ToString()
```

Remark:

This function can be applied on module: i7041, i7044, i7050, i7052, i7060, i7063, i7065, i87051, i87052, i87053, i87054, i87055, i87063.

■ ClearDigitalInLatch

Description:

This function can clear the latch status on the digital input module when the latch function has been enabled.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.ClearDigitalInLatch(byte ComPort, byte address, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum : [Input] 0=checksum disable, 1=checksum enable

TimeOut : [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;          byte address_7044=0x03;
byte checksum=0;      ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7044.ClearDigitalInLatch(port, address_7044, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7044 As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7044.ClearDigitalInLatch(port, address_7044, checksum, timeout)
```

Remark:

This function can be applied on module: i7041, i7044, i7050, i7052, i7060, i7063, i7065, i87051, i87052, i87053, i87054, i87055, i87063.

■ DigitalInCounterRead

Description:

This function obtains the counter event values of the channel numbers on the Digital Input module.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.DigitalInCounterRead(byte ComPort, byte address, byte channel, byte CheckSum, ushort TimeOut, out ushort CounterValue)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

channel: [Input] The input channel No.

CheckSum : [Input] 0=checksum disable, 1=checksum enable

TimeOut : [Input] Time out setting, normal=100, unit=ms.

CounterValue:[Output] Counter value of the digital input channel No.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte address_7044=0x03;
byte channel=0;
byte checksum=0;
ushort timeout=100;
ushort counter;
ushort Ecode;
Ecode=Dcon.i7044.DigitalInCounterRead(port, address_7044, channel,checksum, timeout,out counter);
textBox1.Text = counter.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7044 As Byte = &H3
Dim channel As Byte = 1
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim counter As UInt16
Dim Ecode As UInt16
Ecode = Dcon.i7044.DigitalInCounterRead(port, address_7044, channel, checksum, timeout, counter)
TextBox1.Text = counter.ToString()
```

Remark:

This function can be applied on module: i7041, i7044, i7050, i7052, i7060, i7063, i7065, i87051, i87052, i87053, i87054, i87055, i87063.

■ ClearDigitalInCounter

Description:

Clears the counter values of the channel numbers on the Digital Input module.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.ClearDigitalInCounter(byte ComPort, byte address, byte channel, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

channel: [Input] The input channel No.

CheckSum : [Input] 0=checksum disable, 1=checksum enable

TimeOut : [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte address_7044=0x03;
byte channel=0;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7044.ClearDigitalInCounter(port, address_7044, channel,  checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7044 As Byte = &H3
Dim channel As Byte = 1
Dim checksum As Byte = 0
```

```
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7044.ClearDigitalInCounter(port, address_7044, channel, checksum, timeout)
```

Remark:

This function can be applied on module: i7041, i7044, i7050, i7052, i7060, i7063, i7065, i87051, i87052, i87053, i87054, i87055, i87063.

■ ReadEventCounter

Description:

Obtains the value of the event counter on the I-7000 series module. This function only supports I-7011, I-7012, I-7014, and I-7016 series modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.ReadEventCounter(byte ComPort, byte address, byte CheckSum, ushort TimeOut, out ushort CounterValue)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum : [Input] 0=checksum disable, 1=checksum enable

TimeOut : [Input] Time out setting, normal=100, unit=ms.

CounterValue:[Output] The value of event counter.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte address_7014=0x01;
byte checksum=0;
ushort timeout=100;
ushort counter;
ushort Ecode;
Ecode=Dcon.i7014.ReadEventCounter(port, address_7014, checksum, timeout, out counter);
textBox1.Text = counter.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7014 As Byte = &H1
```

```
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim counter As UInt16 : Dim Ecode As UInt16
Ecode = Dcon.i7014.ReadEventCounter(port, address_7014, checksum, timeout, counter)
TextBox1.Text = counter.ToString
```

Remark:

This function can be applied on module: i7011, i7012, i7014, i7016.

■ ClearEventCounter

Description:

This function clears the event counter value on the I-7000 series module. It supports the I-7011, I-7012, I-7014, and I-7016 series modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.ClearEventCounter(byte ComPort, byte address, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum : [Input] 0=checksum disable, 1=checksum enable

TimeOut : [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;                    byte address_7014=0x01;
byte checksum=0;                ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7014.ClearEventCounter(port, address_7014, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3 : Dim address_7014 As Byte = &H1
Dim checksum As Byte = 0 : Dim timeout As UInt16
timeout = Convert.ToUInt16(100) : Dim Ecode As UInt16
Ecode = Dcon.i7014.ClearEventCounter(port, address_7014, checksum, timeout)
```

Remark:

This function can be applied on module: i7011, i7012, i7014, i7016.

4.3 Digital Output Functions

■ DO_##

Description:

This function outputs the values for the digital output module on the I-7000/I-8000/I-87K series modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.DO_##(byte ComPort, byte address, [byte slot ,]
Data_Type cdata, byte CheckSum, ushort TimeOut)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
slot :	[Input] Slot number; the I/O module installed in I-8000 main unit.
cdata :	[Input] ##-bit digital output data
CheckSum :	[Input] 0=checksum disable, 1=checksum enable
TimeOut :	[Input] Time out setting, normal=100, unit=ms.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;
byte address_7060=0x03;
byte checksum=0;
ushort timeout=100;
byte OutTemp=0xF;
ushort Ecode;
Ecode=Dcon.i7060.DO_4(port, address_7060, OutTemp, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7060 As Byte = &H3
```

```
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim outTemp As Byte = &HF
Dim Ecode As UInt16
Ecode = Dcon.i7060.DO_4(port, address_7060, outTemp, checksum, timeout)
```

Remark:

1.This “##” can be applied on number: 2, 3, 4, 5, 7, 8, 13, 16, 32

2.This Data_Type can be applied on data type: byte, ushort, uint.

3.This function can be applied on module: i7011, i7012, i7014, i7016, i7042, i7043, i7044, i7050, i7060, i7063, i7065, i7066, i7067, i7080, i8041, i8042, i8054, i8055, i8056, i8057, i8060, i8063, i8064, i8065, i8066, i8068, i8077, i87054, i87055, i87057, i87060, i87063, i87064, i87065, i87066, i87068, i87069.

■ DO_Bit

Description:

This function sets the digital value of the digital output channel number for the I/7000/I-8000/I-87K series modules. The output value is “0” or “1”.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.DO_Bit(byte ComPort, byte address, [byte slot ,] byte channel , byte on_off, byte CheckSum, ushort TimeOut)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
slot :	[Input] Slot number; the I/O module installed in I-8000 main unit.
channel:	[Input] The output channel No.
on_off:	[Input] Output digital data; 0 or 1
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;    byte address_7060=0x03;
byte channel = 0;byte checksum=0;
ushort timeout=100;byte on=1;
ushort Ecode;
Ecode=Dcon.i7060.DO_Bit(port, address_7060, channel, on, checksum,timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7060 As Byte = &H3
Dim channel As Byte = 0
```

```
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim onB As Byte = 1
Dim Ecode As UInt16
Ecode = Dcon.i7060.DO_Bit(port, address_7060, channel, onB, checksum, timeout)
```

Remark:

This function can be applied on module: i7042, i7043, i7044, i7050, i7060, i7063, i7065, i7066, i7067, i8041, i8042, i8054, i8055, i8056, i8057, i8060, i8063, i8064, i8065, i8066, i8068, i8077, i87054, i87055, i87057, i87063, i87064, i87065, i87066, i87068, i87069.

■ DigitalOutReadBack

Description:

Reads back the digital output value of I-7000/I-8000/I-87K series modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.DigitalOutReadBack(byte ComPort, byte address,
[byte slot ,], byte CheckSum, ushort TimeOut, out Data_Type DigitalValue)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

slot : [Input] Slot number; the I/O module installed in I-8000 main unit.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

DigitalValue: [Output] ##-bit digital output data read back.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;          byte address_7060=0x03;
byte checksum=0;      ushort timeout=100;
byte outBack;         ushort Ecode;
Ecode=Dcon.i7060.DigitalOutReadBack(port, address_7060, checksum, timeout,out outBack);
textBox1.Text = outBack.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3 : Dim address_7060 As Byte = &H3
Dim checksum As Byte = 0 : Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim outBack As Byte : Dim Ecode As UInt16
Ecode = Dcon.i7060.DigitalOutReadBack(port, address_7060, checksum, timeout, outBack)
TextBox1.Text = outBack.ToString()
```

Remark:

1.This “##” can be applied on number: 2, 3, 4, 5, 7, 8, 13, 16, 32

2.This Data_Type can be applied on data type: byte, ushort, uint.

3.This function can be applied on module: i7042, i7043, i7044, i7050, i7060, i7063, i7065, i7066, i7067, i8041, i8042, i8054, i8055, i8056, i8057, i8060, i8063, i8064, i8065, i8066, i8068, i8077, i87054, i87055, i87057, i87063, i87064, i87065, i87066, i87068, i87069.

4.4 Analog Input Functions

■ AnalogIn

Description:

This function obtains the analog input value from DCON (I-7000/I-8000/I-87K) series modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.AnalogIn(byte ComPort, byte address, [byte slot ,] [byte channel,] byte CheckSum, ushort TimeOut, out float AnalogValue)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

slot : [Input] Slot number; the I/O module installed in I-8000 main unit.

channel: [Input] The input channel No.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

AnalogValue: [Output] The analog input value in “float” format

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;                    byte address_7014=0x01;
byte checksum=0;                ushort timeout=100;
float InTemp;                   ushort Ecode;
Ecode=Dcon.i7014.AnalogIn(port, address_7014, checksum, timeout, out InTemp);
textBox1.Text = InTemp.ToString("f");
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7014 As Byte = &H1
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim InTemp As Single
Dim Ecode As UInt16
Ecode = Dcon.i7014.AnalogIn(port, address_7014, checksum, timeout, InTemp)
TextBox1.Text = InTemp.ToString("f")
```

Remark:

This function can be applied on module: i7011, i7012, i7013, i7014, i7016, i7016P, i7017, i7018, i7033, i8017, i87013, i87016, i87017, i87018.

■ AnalogInHex

Description:

This function obtains the analog input value in the “Hexadecimal” format from the I-7000/I-8000/I-87K series modules.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.AnalogInHex( byte ComPort, byte address,
                [ byte slot , ] [ byte channel, ] byte CheckSum, ushort TimeOut,
                out ushort AnalogValue)
```

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
slot :	[Input] Slot number; the I/O module installed in I-8000 main unit.
channel:	[Input] The input channel No.
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.
AnalogValue:	[Output] The analog input value in “Hexadecimal” format.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;                    byte address_7014=0x01;
byte checksum=0;                ushort timeout=100;
ushort InTemp;                  ushort Ecode;
Ecode=Dcon.i7014.AnalogInHex(port, address_7014, checksum, timeout, out InTemp);
textBox1.Text = InTemp.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7014 As Byte = &H1
```

```
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim InTemp As UInt16 :Dim Ecode As UInt16
Ecode = Dcon.i7014.AnalogInHex(port, address_7014, checksum, timeout, InTemp)
TextBox1.Text = InTemp.ToString()
```

Remark:

This function can be applied on module: i7011, i7012, i7013, i7014, i7016, i7016P, i7017, i7018, i7033, i8017, i87013, i87016, i87017, i87018.

■ AnalogInFsr

Description:

This function obtains the analog input value in the “FSR” format from the I-7000/I-8000/I-87K series modules. The “FSR” format is the “Percent” format.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.AnalogInFsr( byte ComPort, byte address,
                [ byte slot , ] [ byte channel, ] byte CheckSum, ushort TimeOut,
                out float AnalogValue)
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

slot : [Input] Slot number; the I/O module installed in I-8000 main unit.

channel: [Input] The input channel No.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

AnalogValue:[Output] The analog input value.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;                    byte address_7014=0x01;
byte checksum=0;                ushort timeout=100;
float InTemp;                   ushort Ecode;
Ecode=Dcon.i7014.AnalogInFsr(port, address_7014, checksum, timeout,out InTemp);
textBox1.Text = InTemp.ToString("f");
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7014 As Byte = &H1
```

```
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim InTemp As Single
Dim Ecode As UInt16
Ecode = Dcon.i7014.AnalogInFsr(port, address_7014, checksum, timeout, InTemp)
TextBox1.Text = InTemp.ToString("f")
```

Remark:

This function can be applied on module: i7011, i7012, i7013, i7014, i7016, i7016P, i7017, i7018, i7033, i8017, i87013, i87016, i87017, i87018.

■ AnalogIn_All

Description:

This function obtains the analog input values of all channels in the Engineer/FSR formats from the DCON (I-7000/I-8000/I-87K) series modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.AnalogIn_All(byte ComPort, byte address,
[byte slot ,] byte CheckSum, ushort TimeOut, [byte dataformat,]
[float[] fdata] / [ushort[] Hexdata])

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

slot : [Input] Slot number; the I/O module installed in I-8000 main unit.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

dataformat: [Input] Data Foramt: 0:Engineer 1:FSR

fdata[0] / Hexdata[0]: [Output] analog input value of channel_0

fdata[1] / Hexdata[1]: [Output] analog input value of channel_1

fdata[2] / Hexdata[2]: [Output] analog input value of channel_2

:

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr1=0x01;      byte addr2=0x02;      byte checksum=0;
ushort timeout=100;
float [] fAI=new float[8];
```

```
ushort[] uAI=new ushort[8];
ushort Ecode;
Ecode=Dcon.i7017.AnalogIn_All(port, addr1, checksum, timeout, 0, out fAI);
textBox1.Text = fAI[0].ToString("f");  // This is a analog input value of channel 0.
textBox2.Text = fAI[1].ToString("f");  // This is a analog input value of channel 1.
textBox3.Text = fAI[2].ToString("f");  // This is a analog input value of channel 2.
textBox4.Text = fAI[3].ToString("f");  // This is a analog input value of channel 3.
textBox5.Text = fAI[4].ToString("f");  // This is a analog input value of channel 4.
textBox6.Text = fAI[5].ToString("f");  // This is a analog input value of channel 5.
textBox7.Text = fAI[6].ToString("f");  // This is a analog input value of channel 6.
textBox8.Text = fAI[7].ToString("f");  // This is a analog input value of channel 7.
Ecode=Dcon.i7017.AnalogIn_All(port, addr2, checksum, timeout, out uAI);
textBox9.Text = uAI[0].ToString();  // This is a analog input value of channel 0.
textBox10.Text = uAI[1].ToString();  // This is a analog input value of channel 1.
textBox11.Text = uAI[2].ToString();  // This is a analog input value of channel 2.
textBox12.Text = uAI[3].ToString();  // This is a analog input value of channel 3.
textBox13.Text = uAI[4].ToString();  // This is a analog input value of channel 4.
textBox14.Text = uAI[5].ToString();  // This is a analog input value of channel 5.
textBox15.Text = uAI[6].ToString();  // This is a analog input value of channel 6.
textBox16.Text = uAI[7].ToString();  // This is a analog input value of channel 7.
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr1 As Byte = &H1
Dim addr2 As Byte = &H2
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim fAI(8) As Single
Dim uAI(8) As UInt16
Dim Ecode As UInt16
Ecode = Dcon.i7017.AnalogIn_All(port, addr1, checksum, timeout, 0, fAI)
TextBox1.Text = fAI(0).ToString("f")   'This is a analog input value of channel 0.
textBox2.Text = fAI(1).ToString("f")   'This is a analog input value of channel 1.
textBox3.Text = fAI(2).ToString("f")   'This is a analog input value of channel 2.
textBox4.Text = fAI(3).ToString("f")   'This is a analog input value of channel 3.
textBox5.Text = fAI(4).ToString("f")   'This is a analog input value of channel 4.
textBox6.Text = fAI(5).ToString("f")   'This is a analog input value of channel 5.
textBox7.Text = fAI(6).ToString("f")   'This is a analog input value of channel 6.
```

```
textBox8.Text = fAI(7).ToString("f")   'This is a analog input value of channel 7.
Ecode = Dcon.i7017.AnalogIn_All(port, addr2, checksum, timeout, uAI)
textBox9.Text = uAI(0).ToString()   'This is a analog input value of channel 0.
textBox10.Text = uAI(1).ToString()   'This is a analog input value of channel 1.
textBox11.Text = uAI(2).ToString()   'This is a analog input value of channel 2.
textBox12.Text = uAI(3).ToString()   'This is a analog input value of channel 3.
textBox13.Text = uAI(4).ToString()   'This is a analog input value of channel 4.
textBox14.Text = uAI(5).ToString()   'This is a analog input value of channel 5.
textBox15.Text = uAI(6).ToString()   'This is a analog input value of channel 6.
textBox16.Text = uAI(7).ToString()   'This is a analog input value of channel 7.
```

Remark:

1. This function using in Engineer/FSR format can be applied on module: i7017, i7018, i7033, i8017, i87013, i87016, i87017, i87018.
2. This function using in Hexadecimal format can be applied on module: i7017, i7018, i7033.

■ ThermocoupleOpen

Description:

This function can be used to detect the thermocouple state of the I-7011 module for when the following supporting types “J, K, T, E, R, S, B, N, C” are open or closed. If the response value is “0”, then the I-7011 thermocouple is working in a closed state. If the response value is “1”, then the I-7011 thermocouple is working in an open state. For more information with regards to this, please refer to the user manual.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.ThermocoupleOpen(byte ComPort, byte address, byte CheckSum, ushort TimeOut, out byte Close_Open)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Close_Open: [Output] 0: the thermocouple is close

1: the thermocouple is open

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;                byte address_7011=0x01;
byte checksum=0;            ushort timeout=100;
byte Thermocouple;          ushort Ecode;
Ecode=Dcon.i7011.ThermocoupleOpen(port, address_7011, checksum, timeout,out Thermocouple);
textBox1.Text = Thermocouple.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7011 As Byte = &H1
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim Thermocouple As Byte
Dim Ecode As UInt16
Ecode = Dcon.i7011.ThermocoupleOpen(port, address_7011, checksum, timeout, Thermocouple)
TextBox1.Text = Thermocouple.ToString()
```

Remark:

This function can be applied on module: i7011.

■ SetLedDisplay

Description:

This function will configure the LED Display for a specified channel on either the I-7033 or the I-7016.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.SetLedDisplay(byte ComPort, byte address, byte channel, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

channel : [Input] Set display channel for 7033 or 7016

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte address_7016=0x02;
byte channel=1;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7016.SetLedDisplay(port, address_7016, channel, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7016 As Byte = &H2
Dim channel As Byte = 1
```

```
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7016.SetLedDisplay(port, address_7016, channel, checksum, timeout)
```

Remark:

This function can be applied on module: i7016, i7033.

■ GetLedDisplay

Description:

This function will get the LED Display for a specified channel on either the I-7033 or the I-7016.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.GetLedDisplay(byte ComPort, byte address,
byte CheckSum, ushort TimeOut, out byte Channel)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Channel: [Output] Current channel for LED display.
0=channel_0; 1=channel_1

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte address_7016=0x02;
byte checksum=0;
ushort timeout=100;
byte channel;
ushort Ecode;
Ecode=Dcon.i7016.GetLedDisplay(port, address_7016, checksum, timeout,out channel);
textBox1.Text = channel.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3 : Dim address_7016 As Byte = &H2
```

```
Dim checksum As Byte = 0 : Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim channel As Byte = 1 : Dim Ecode As UInt16
Ecode = Dcon.i7016.GetLedDisplay(port, address_7016, checksum, timeout, channel)
TextBox1.Text = channel.ToString()
```

Remark:

This function can be applied on module: i7016, i7033.

4.5 Analog Output Functions

■ AnalogOut

Description:

Will output the analog value from an Analog output module on one of the I-7000/I-8000/I-87K series modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.AnalogOut(byte ComPort, byte address, [byte slot ,] [byte channel,] float fdata, byte CheckSum, ushort TimeOut)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
slot :	[Input] Slot number; the I/O module installed in I-8000 main unit.
channel:	[Input] The analog output channel No.
fdata :	[Input] Analog output value
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;                    byte address_7024=0x03;
byte checksum=0;                ushort timeout=100;
float ch0=1.5f;                 float ch1=2.5f;
float ch2=3.0f;                 float ch3=3.5f;
ushort Ecode;
Ecode=Dcon.i7024.AnalogOut(port, address_7024, 0, ch0, checksum, timeout);
Ecode=Dcon.i7024.AnalogOut(port, address_7024, 1, ch1, checksum, timeout);
```

```
Ecode=Dcon.i7024.AnalogOut(port, address_7024, 2, ch2, checksum, timeout);
Ecode=Dcon.i7024.AnalogOut(port, address_7024, 3, ch3, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7024 As Byte = &H3
Dim checksum As Byte = 0 : Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim ch0 As Single = 1.5 : Dim ch1 As Single = 2.5
Dim ch2 As Single = 3.0 : Dim ch3 As Single = 3.5
Dim Ecode As UInt16
Ecode = Dcon.i7024.AnalogOut(port, address_7024, 0, ch0, checksum, timeout)
Ecode = Dcon.i7024.AnalogOut(port, address_7024, 1, ch1, checksum, timeout)
Ecode = Dcon.i7024.AnalogOut(port, address_7024, 2, ch2, checksum, timeout)
Ecode = Dcon.i7024.AnalogOut(port, address_7024, 3, ch3, checksum, timeout)
```

Remark:

This function can be applied on module: i7016, i7021, i7022, i7024, i8024, i87022, i87024, i87026.

■ AnalogOutHex

Description:

This function will output the analog value for analog output modules through a Hex format.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.AnalogOutHex(byte ComPort, byte address,
[byte channel ,], ushort data, byte CheckSum, ushort TimeOut)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
channel:	[Input] The analog output channel No.
data :	[Input] Analog output value in Hexadecimal Data format
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;
byte address_7021=0x02;
byte checksum=0;
ushort timeout=100;
ushort data=6553;
ushort Ecode;
Ecode=Dcon.i7021.AnalogOutHex(port, address_7021, data, checksum,timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim address_7021 As Byte = &H2
```

```
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim data As UInt16 = Convert.ToUInt16(6553)
Dim Ecode As UInt16
Ecode = Dcon.i7021.AnalogOutHex(port, address_7021, data, checksum, timeout)
```

Remark:

This function can be applied on module: i7021, i7022, i87022, i87026.

■ AnalogOutFsr

Description:

This function outputs the analog value of analog output modules through the % of span data format. This function can only be used after the analog output module has been set in the “FSR” output mode.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.AnalogOutFsr(byte ComPort, byte address,
[byte channel,] float fdata, byte CheckSum, ushort TimeOut)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
slot :	[Input] Slot number; the I/O module installed in I-8000 main unit.
channel:	[Input] The analog output channel No.
fdata :	[Input] Analog output value in % of Span data format.
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;
byte address_7021=0x02;
byte checksum=0;
ushort timeout=100;
float fdata=8.5f;
ushort Ecode;
Ecode=Dcon.i7021.AnalogOutFsr(port, address_7021, fdata, checksum,timeout);
```

[Visual Basic]

```
Dim port As Byte = 3 : Dim address_7021 As Byte = &H2
Dim checksum As Byte = 0 : Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim fdata As Single = 8.5 : Dim Ecode As UInt16
Ecode = Dcon.i7021.AnalogOutFsr(port, address_7021, fdata, checksum, timeout)
```

Remark:

This function can be applied on module: i7021, i7022, i7024, i87022, i87026.

■ AnalogOutReadBack

Description:

This function will read back the analog output value on analog output modules for the I-7000/I-8000 /I-87K series modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.AnalogOutReadBack(byte ComPort, byte address,
[byte slot ,] [byte channel,] byte CheckSum, ushort TimeOut,
out float AnalogValue)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

slot : [Input] Slot number; the I/O module installed in I-8000 main unit.

channel: [Input] The analog output channel No.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

AnalogValue:[Output] Analog output read back value.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;                byte address_7024=0x03;
byte ch0 = 0;               byte checksum=0;
ushort timeout=100;         float fA00;
ushort Ecode;
Ecode=Dcon.i7024.AnalogOutReadBack(port, address_7024, ch0, checksum,timeout, out fA00);
textBox1.Text = fA00.ToString("f");  //This is a analog output value of channel 0.
```

[Visual Basic]

```
Dim port As Byte = 3 : Dim address_7024 As Byte = &H3
```

```
Dim ch0 As Byte = 0 : Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim fAO0 As Single : Dim Ecode As UInt16
Ecode = Dcon.i7024.AnalogOutReadBack(port, address_7024, ch0, checksum, timeout, fAO0)
TextBox1.Text = fAO0.ToString("f")  'This is a analog output value of channel 0.
```

Remark:

This function can be applied on module: i7016, i7021, i7022, i7024, i8024, i87022, i87024, i87026.

■ AnalogOutReadBackHex

Description:

This function will read back the analog output value on analog output modules for the I-7000/I-87K series modules.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.AnalogOutReadBackHex( byte ComPort,
        byte address, [byte channel,] byte CheckSum, ushort TimeOut,
        out ushort AnalogValue )
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

channel: [Input] The analog output channel No.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

AnalogValue:[Output] Analog output value in Hexadecimal Data format.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;                byte address_7021=0x03;
byte checksum=0;            ushort timeout=100;
ushort A00;                 ushort Ecode;
Ecode=Dcon.i7021.AnalogOutReadBackHex(port, address_7021, checksum,timeout, out A00);
textBox1.Text = A00.ToString();  //This is a analog output value of channel 0.
```

[Visual Basic]

```
Dim port As Byte = 3 : Dim address_7021 As Byte = &H3
Dim checksum As Byte = 0 : Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
```

```
Dim AO0 As UInt16 : Dim Ecode As UInt16
Ecode = Dcon.i7021.AnalogOutReadBackHex(port, address_7021, checksum, timeout, AO0)
TextBox1.Text = AO0.ToString()  'This is a analog output value of channel 0.
```

Remark:

This function can be applied on module: i7021, i7022, i87022, i87026.

■ AnalogOutReadBackFsr

Description:

This function will read back the analog output value on analog output modules for the I-7000/I-87K series modules.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.AnalogOutReadBackFsr( byte ComPort,
        byte address, [byte channel,] byte CheckSum, ushort TimeOut,
        out float AnalogValue )
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

channel: [Input] The analog output channel No.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

AnalogValue:[Output] Analog output value in % of Span data format.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;                byte address_7021=0x03;
byte checksum=0;            ushort timeout=100;
float fAO0;                 ushort ECode;
Ecode=Dcon.i7021.AnalogOutReadBack(port, address_7021, checksum,timeout, out fAO0);
textBox1.Text = fAO0.ToString("f")  //This is a analog output value of channel 0.
```

[Visual Basic]

```
Dim port As Byte = 3 : Dim address_7021 As Byte = &H3
Dim checksum As Byte = 0 : Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
```

```
Dim fAO0 As Single : Dim Ecode As UInt16
Ecode = Dcon.i7021.AnalogOutReadBack(port, address_7021, checksum, timeout, fAO0)
TextBox1.Text = fAO0.ToString("f")  'This is a analog output value of channel 0.
```

Remark:

This function can be applied on module: i7021, i7022, i87022, i87026.

4.6 Module Alarm Functions

■ EnableAlarm

Description:

This enables the alarm function on I-7000 series modules, and will configure it into the status of a momentary alarm or into the latch alarm mode. This function currently supports I-7011, I-7012, I-7014, and I-7016 series modules.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.EnableAlarm( byte ComPort, byte address,
                byte Momentary_Latch, byte CheckSum, ushort TimeOut)
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

Momentary_Latch :[Input] 0:momentary alarm mode 1: latch alarm mode

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;                byte addr_7011=0x03;
byte Latch=1;               byte checksum=0;
ushort timeout=100;         ushort Ecode;
Ecode=Dcon.i7011.EnableAlarm(port, addr_7011, Latch, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3 : Dim addr_7011 As Byte = &H3
Dim Latch As Byte = 1 : Dim checksum As Byte = 0
Dim timeout As UInt16 : timeout = Convert.ToUInt16(100)
Dim Ecode As UInt16
```

```
Ecode = Dcon.i7011.EnableAlarm(port, addr_7011, Latch, checksum, timeout)
```

Remark:

This function can be applied on module: i7011, i7012, i7014, i7016.

■ DisableAlarm

Description:

This will disable the alarm function on the I-7000 series module. This function currently supports I-7011, I-7012, I-7014, and I-7016.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.DisableAlarm(byte ComPort, byte address,
byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;                byte addr_7011=0x03;
byte checksum=0;            ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7011.DisableAlarm(port, addr_7011, checksum, timeout);
// Disable i7011 Alarm
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7011 As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim Ecode As UInt16
```

```
Ecode = Dcon.i7011.DisableAlarm(port, addr_7011, checksum, timeout)
' Disable i7011 Alarm
```

Remark:

This function can be applied on module: i7011, i7012, i7014, i7016.

■ ClearLatchAlarm

Description:

This function can clear the latched alarm on I-7000/I-8000 series modules. This function currently supports I-7011, I-7012, I-7014, I-7016, and I-8017H.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.ClearLatchAlarm(byte ComPort, byte address,
[byte slot, byte channel,] byte CheckSum, ushort TimeOut)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
slot :	[Input] Slot number; the I/O module installed in I-8000 main unit.
channel:	[Input] The analog input channel No.
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;
byte addr_7011=0x03;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7011.ClearLatchAlarm(port, addr_7011, checksum, timeout);
//Clear Latch Alarm for i7011 Alarm
```

[Visual Basic]

```
Dim port As Byte = 3
```

```
Dim addr_7011 As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7011.ClearLatchAlarm(port, addr_7011, checksum, timeout)
' Clear Latch Alarm for i7011 Alarm
```

Remark:

This function can be applied on module: i7011, i7012, i7014, i7016, i8017.

■ SetAlarmLimitValue

Description:

This function will set the high or low alarm limit value of I-7000/I-8000 series modules. This function currently supports I-7011, I-7012, I-7014, I-7016, I-7080, and I-8017H series modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.SetAlarmLimitValue(byte ComPort, byte address,
[byte slot, byte channel,] [[byte mode,] [byte Low_High,]]
Data_Type AlarmValue, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

slot : [Input] Slot number; the I/O module installed in I-8000 main unit.

channel: [Input] The analog input channel No.

mode : [Input] When in 7080 alarm mode(mode 0)
0:To set Counter 0 alarm value 1:To set Counter 1 alarm value

Low_High : [Input] 0: low alarm value setting 1: high alarm value setting

AlarmValue : [Input] Alarm value

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;                byte addr_7011=0x03;
byte checksum=0;            ushort timeout=100;
byte high=1;                float limit=8.5f;
ushort Ecode;
Ecode=Dcon.i7011.SetAlarmLimitValue(port, addr_7011, high, limit, checksum, timeout);
```

```
// Set i7011 Alarm Limit Value
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7011 As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim high As Byte = 1
Dim limit As Single = 8.5
Dim Ecode As UInt16
Ecode = Dcon.i7011.SetAlarmLimitValue(port, addr_7011, high, limit, checksum, timeout)
' Set i7011 Alarm Limit Value
```

Remark:

This Data_Type can be applied on data type: uint, float.

This function can be applied on module: i7011, i7012, i7014, i7016, i7080, i8017.

■ ReadAlarmLimitValue

Description:

This function obtains the high or low alarm limit value of I-7000/I-8000 series modules. This function currently supports I-7011, I-7012, I-7014, I-7016, I-7080, and I-8017H series modules.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.ReadAlarmLimitValue( byte ComPort, byte address,
                [byte slot, byte channel,] [ [byte counter ] [byte Low_High] ],
                byte CheckSum,  ushort TimeOut, out float AlarmValue)
```

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
slot :	[Input] Slot number; the I/O module installed in I-8000 main unit.
channel:	[Input] The analog input channel No.
counter :	[Input] 0: Counter 0, 1: Counter 1 (For I-7080 module)
Low_High :	[Input] 0: low alarm value setting 1: high alarm value setting
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.
AlarmValue :	[Output] Alarm value

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;                byte addr_7011=0x03;
byte checksum=0;            ushort timeout=100;
byte high=1;                float limit;
ushort Ecode;
Ecode=Dcon.i7011.ReadAlarmLimitValue(port, addr_7011, high, checksum,timeout, out limit);
textBox1.Text=limit.ToString("f");
```

```
//This is a high alarm limit value.
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7011 As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim high As Byte = 1
Dim limit As Single
Dim Ecode As UInt16
Ecode = Dcon.i7011.ReadAlarmLimitValue(port, addr_7011, high, checksum, timeout, limit)
TextBox1.Text = limit.ToString("f")
' This is a high alarm limit value.
```

Remark:

This function can be applied on module: i7011, i7012, i7014, i7016, i7080, i8017.

■ ReadOutputAlarmState

Description:

This function obtains the alarm mode and alarm digital output value of the I-7000 series module. This function currently supports I-7011, I-7012, I-7014, I-7016, and 1-7080 series modules.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.ReadOutputAlarmState( byte ComPort, byte address,
                byte CheckSum, ushort TimeOut, out byte AlarmState,
                out byte Digital_Output)
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

AlarmState : [Output] 0: alarm disable 1: momentary alarm 2: latch alarm

For 7080 mode(alarm mode 0)

0	Counter:0	disable	Counter:1	disable
1	Counter:0	enable	Counter:1	disable
2	Counter:0	disable	Counter:1	enable
3	Counter:0	enable	Counter:1	enable

For 7080D mode (alarm mode 1)

0	Counter:0 disable
1	Counter:0 momentary alarm mode
2	Counter:0 latch alarm mode
	Counter 1 No used in Alarm mode 1

Digital_Output: [Output] 0: (DO0=off, DO1=off) 1: (DO0=on, DO1=off)
2: (DO0=off, DO1=on) 3: (DO0=on, DO1=on)

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;                byte addr_7011=0x03;
byte checksum=0;            ushort timeout=100;
byte state;                 byte DO_Val;
ushort Eode;
Ecode=Dcon.i7011.ReadOutputAlarmState(port, addr_7011, checksum, timeout,out state, out DO_Val);
textBox1.Text= state.ToString();
textBox2.Text= DO_Val.ToString();
// TextBox1 is a alarm state value. TextBox2 is a digital output value.
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7011 As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim state As Byte : Dim DO_Val As Byte
Dim Ecode As UInt16
Ecode = Dcon.i7011.ReadOutputAlarmState(port, addr_7011, checksum, timeout, state, DO_Val)
TextBox1.Text = state.ToString()
TextBox2.Text = DO_Val.ToString()
' TextBox1 is a alarm state value. TextBox2 is a digital output value.
```

Remark:

This function can be applied on module: i7011, i7012, i7014, i7016, i7080.

■ SetAlarmMode

Description:

This disables or enables the alarm function on I-8000 series modules into a momentary alarm or latch alarm mode. This function currently supports I-8017h.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.SetAlarmMode( byte ComPort, byte address,
                byte slot, byte channel, byte Low_High, byte AlarmMode,
                byte CheckSum, ushort TimeOut)
```

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
slot :	[Input] Slot number; the I/O module installed in I-8000 main unit.
channel:	[Input] The analog input channel No.
Low_High :	[Input] 0: Low Alarm 1: High Alarm
AlarmMode :	[Input] 0:disable 1:Momentary alarm mode 2:Latch alarm mode
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;                byte NetID=0x01;
byte slot=4;                byte ch0=0;
byte high=1;                byte latch=2;
byte checksum=0;            ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i8017.SetAlarmMode(port, NetID, slot, ch0, high, latch, checksum,timeout);
// Set i8017 Alarm Mode to Latch alarm mode
```

[Visual Basic]

```
Dim port As Byte = 3
Dim NetID As Byte = &H1
Dim slot As Byte = 4
Dim ch0 As Byte = 0
Dim high As Byte = 1
Dim latch As Byte = 2
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i8017.SetAlarmMode(port, NetID, slot, ch0, high, latch, checksum, timeout)
' Set i8017 Alarm Mode to Latch alarm mode
```

Remark:

This function can be applied on module: i8017.

■ ReadAlarmMode

Description:

This function obtains the alarm mode setting for analog input modules on the I-8000 series module.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.ReadAlarmMode( byte ComPort, byte address,
                byte slot, byte channel, byte Low_High, byte CheckSum,
                ushort TimeOut, out byte AlarmMode)
```

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
slot :	[Input] Slot number; the I/O module installed in I-8000 main unit.
channel:	[Input] The analog input channel No.
Low_High :	[Input] 0: Low Alarm 1: High Alarm
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.
AlarmMode :	[Output] 0:disable 1:Momentary alarm mode 2:Latch alarm mode

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;                byte NetID=0x01;
byte slot=4;                byte ch0=0;
byte high=1;                byte checksum=0;
ushort timeout=100;         byte mode;
ushort Ecode;
Ecode=Dcon.i8017.ReadAlarmMode(port, NetID, slot, ch0, high, checksum,timeout, out mode);
textBox1.Text= mode.ToString();
// The textBox1 show a i8017 alarm mode value.
```

[Visual Basic]

```
Dim port As Byte = 3
Dim NetID As Byte = &H1
Dim slot As Byte = 4
Dim ch0 As Byte = 0
Dim high As Byte = 1
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim mode As Byte
Dim Ecode As UInt16
Ecode = Dcon.i8017.ReadAlarmMode(port, NetID, slot, ch0, high, checksum, timeout, mode)
TextBox1.Text = mode.ToString()
' The textBox1 show a i8017 alarm mode value.
```

Remark:

This function can be applied on module: i8017.

■ ReadAlarmStatus

Description:

This function attains the alarm status of analog input modules on I-8000 series modules.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.ReadAlarmStatus( byte ComPort, byte address,
                byte slot, byte channel, byte CheckSum, ushort TimeOut,
                out byte HighAlarm, out byte LowAlarm)
```

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
slot :	[Input] Slot number; the I/O module installed in I-8000 main unit.
channel:	[Input] The analog input channel No.
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.
HighAlarm :	[Output] 1:High Alarm Occur 0:Don't Occur
LowAlarm:	[Output] 1:Low Alarm Occur 0:Don't Occur

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;                byte NetID=0x01;
byte slot=4;                byte ch0=0;
byte checksum=0;            ushort timeout=100;
byte high;                  byte low;
ushort Ecode;
Ecode=Dcon.i8017.ReadAlarmStatus(port, NetID, slot, ch0, checksum, timeout,
out high, out low);
textBox1.Text= high.ToString();
```

```
textBox2.Text= low.ToString();
//The textBox1 show a i8017 high alarm state value.
//The textBox2 show a i8017 low alarm state value.
```

[Visual Basic]

```
Dim port As Byte = 3
Dim NetID As Byte = &H1
Dim slot As Byte = 4
Dim ch0 As Byte = 0
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim high As Byte
Dim low As Byte
Dim Ecode As UInt16
Ecode = Dcon.i8017.ReadAlarmStatus(port, NetID, slot, ch0, checksum, timeout, high, low)
TextBox1.Text = high.ToString()
TextBox2.Text = low.ToString()
' The textBox1 show a i8017 high alarm state value.
' The textBox2 show a i8017 low alarm state value.
```

Remark:

This function can be applied on module: i8017.

■ EnableCounterAlarm_7080

Description:

This enables the counter alarm (for alarm-mode 0) on the I-7080 module.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.EnableCounterAlarm_7080(byte ComPort,
byte address, byte counter, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

counter : [Input] 0: enable alarm mode for counter 0 (For I-7080)
1: enable alarm mode for counter 1

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;          byte addr_7080 =0x05;
byte counter1 = 1;    byte checksum=0;
ushort timeout=100;   ushort Ecode;
Ecode=Dcon.i7080.EnableCounterAlarm_7080(port, addr_7080, counter1,  checksum, timeout);
// It will Enable i7080 Counter Alarm
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim counter1 As Byte = 1
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
```

```
Dim Ecode As UInt16
Ecode = Dcon.i7080.EnableCounterAlarm_7080(port, addr_7080, counter1, checksum, timeout)
' It will Enable i7080 Counter Alarm
```

Remark:

This function can be applied on module: i7080.

■ DisableCounterAlarm_7080

Description:

This disables the alarm mode on the I-7080 module. This function only supports the I-7080 module.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.DisableCounterAlarm_7080(byte ComPort,
byte address, byte counter, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

counter : [Input] 0:Counter 0, 1:Counter 1

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;                byte addr_7080 =0x05;
byte counter1 = 1;          byte checksum=0;
ushort timeout=100;         ushort Ecode;
Ecode=Dcon.i7080.DisableCounterAlarm_7080(port, addr_7080, counter1, checksum, timeout);
// It will Disable i7080 Counter Alarm
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim counter1 As Byte = 1
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
```

```
Dim Ecode As UInt16
Ecode = Dcon.i7080.DisableCounterAlarm_7080(port, addr_7080, counter1, checksum, timeout)
' It will Disable i7080 Counter Alarm
```

Remark:

This function can be applied on module: i7080.

■ EnableCounterAlarm_7080D

Description:

This will enable either the momentary alarm mode or latch alarm mode on the I-7080D module. This function only supports the I-7080D module.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.EnableCounterAlarm_7080D(byte ComPort,
byte address, byte mode, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

mode : [Input] 0: momentary alarm mode
1: latch alarm mode

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;                byte addr_7080D =0x06;
byte mode = 1;              byte checksum=0;
ushort timeout=100;         ushort Ecode;
Ecode=Dcon.i7080.EnableCounterAlarm_7080D(port, addr_7080D, mode, checksum, timeout);
// It will Enable i7080D Counter Alarm
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080D As Byte = &H6
Dim mode As Byte = 1
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
```

```
Dim Ecode As UInt16
Ecode = Dcon.i7080.EnableCounterAlarm_7080D(port, addr_7080D, mode, checksum, timeout)
' It will Enaable i7080D Counter Alarm
```

Remark:

This function can be applied on module: i7080.

■ DisableCounterAlarm_7080D

Description:

This will disable the I-7080D module's alarm mode. This function only supports the I-7080D module.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.DisableCounterAlarm_7080D(byte ComPort, byte address, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;          byte addr_7080D =0x06;
byte checksum=0;      ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7080.DisableCounterAlarm_7080D(port, addr_7080D, checksum, timeout);
// It will Disable i7080D Counter Alarm
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080D As Byte = &H6
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7080.DisableCounterAlarm_7080D(port, addr_7080D, checksum, timeout)
' It will Disable i7080D Counter Alarm
```

Remark:

This function can be applied on module: i7080.

4.7 Counter Functions

■ CounterIn

Description:

This function attains the value on the selected counter in module I-7080.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.CounterIn(byte ComPort, byte address,
byte counter, byte CheckSum, ushort TimeOut, out uint CounterValue)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

counter : [Input] 0: Set counter 0 1: Set counter 1

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

CounterValue: [Output] The counter value from the selected counter.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;                byte addr_7080 =0x05;
byte counter0 = 0;          byte checksum=0;
ushort timeout=100;         uint Cvalue;
ushort Ecode;
Ecode=Dcon.i7080.CounterIn(port, addr_7080, counter0, checksum, timeout, out Cvalue);
textBox1.Text= Cvalue.ToString();
// The textBox1 is a counter0 value.
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
```

```
Dim counter0 As Byte = 0
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim Cvalue As UInt32
Dim Ecode As UInt16
Ecode = Dcon.i7080.CounterIn(port, addr_7080, counter0, checksum, timeout, Cvalue)
TextBox1.Text = Cvalue.ToString()
' The textBox1 is a counter0 value.
```

Remark:

This function can be applied on module: i7080.

■ ClearCounter

Description:

This clears the value from the selected counter in module I-7080.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.ClearCounter(byte ComPort, byte address,
byte counter, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

counter : [Input] 0: Set counter 0 1: Set counter 1

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;          byte addr_7080 =0x05;
byte count0 = 0;      byte checksum=0;
ushort timeout=100;   ushort Ecode;
Ecode=Dcon.i7080.ClearCounter(port, addr_7080, count0, checksum, timeout);
// This Clear the i7080 counter 0.
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim count0 As Byte = 0
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
```

```
Dim Ecode As UInt16
Ecode = Dcon.i7080.ClearCounter(port, addr_7080, count0, checksum, timeout)
' This Clear the i7080 counter 0.
```

Remark:

This function can be applied on module: i7080.

■ ReadCounterMaxValue

Description:

This function attains the maximum setting value of the selected counter in module I-7080.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.ReadCounterMaxValue(byte ComPort, byte address, byte counter, byte CheckSum, ushort TimeOut, out uint MaxValue)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
counter :	[Input] 0: Set counter 0 1: Set counter 1
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.
MaxValue:	[Output] The maximum setting value from the selected counter.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;                byte addr_7080 =0x05;
byte counter0 = 0;          byte checksum=0;
ushort timeout=100;         uint value;
ushort Ecode;
Ecode=Dcon.i7080.ReadCounterMaxValue(port, addr_7080, counter0, checksum, timeout, out value);
textBox1.Text= value.ToString();
// The textBox1 is the i7080 counter0 setting max value.
```

[Visual Basic]

```
Dim port As Byte = 3
```

```
Dim addr_7080 As Byte = &H5
Dim counter0 As Byte = 0
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim value As UInt32
Dim Ecode As UInt16
Ecode = Dcon.i7080.ReadCounterMaxValue(port, addr_7080, counter0, checksum, timeout, value)
TextBox1.Text = value.ToString()
' The textBox1 is the i7080 counter0 setting max value.
```

Remark:

This function can be applied on module: i7080.

■ SetCounterMaxValue

Description:

This function configures the maximum value of the selected counter for module I-7080.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.SetCounterMaxValue(byte ComPort, byte address, byte counter, uint MaxValue, byte CheckSum, ushort TimeOut)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
counter :	[Input] 0: Set counter 0 1: Set counter 1
MaxValue:	[Input] The maximum counter value
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;                byte addr_7080 =0x05;
byte counter0 = 0;          uint maxValue = 100000;
byte checksum=0;            ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7080.SetCounterMaxValue(port, addr_7080, counter0, maxValue, checksum, timeout);
// This will set the i7080 counter0 setting max value.
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim counter0 As Byte = 0
Dim maxValue As UInt32
```

```
maxValue = Convert.ToUInt32(100000)
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7080.SetCounterMaxValue(port, addr_7080, counter0, maxValue, checksum, timeout)
' This will set the i7080 counter0 setting max value.
```

Remark:

This function can be applied on module: i7080.

■ ReadInputSignalMode

Description:

This attains the input signal mode's setting value on the I-7080 module. For more detailed information on "Input signal mode" please refer the user manual.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.ReadInputSignalMode(byte ComPort, byte address, byte CheckSum, ushort TimeOut, out ushort InputSignal)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.
InputSignal:	[Output] 0: (Counter:0 TTL Counter:1 TTL)
	1: (Counter:0 Photo Counter:1 Photo)
	2: (Counter:0 TTL Counter:1 Photo)
	3: (Counter:0 Photo Counter:1 TTL)

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;
byte addr_7080 =0x05;
byte checksum=0;
ushort timeout=100;
ushort InputSignal;
ushort Ecode;
Ecode=Dcon.i7080.ReadInputSignalMode(port, addr_7080, checksum, timeout,out InputSignal);
textBox1.Text= InputSignal.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim InputSignal As UInt16
Dim Ecode As UInt16
Ecode = Dcon.i7080.ReadInputSignalMode(port, addr_7080, checksum, timeout, InputSignal)
TextBox1.Text = InputSignal.ToString()
```

Remark:

This function can be applied on module: i7080.

■ SetInputSignalMode

Description:

This function configures the input signal mode's setting value in the I-7080 module.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.SetInputSignalMode(byte ComPort, byte address, byte mode, byte CheckSum, ushort TimeOut)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3				
address :	[Input] Module address, from 0x00 to 0xFF				
mode :	[Input]	0: (Counter:0	TTL	Counter:1	TTL)
		1: (Counter:0	Photo	Counter:1	Photo)
		2: (Counter:0	TTL	Counter:1	Photo)
		3: (Counter:0	Photo	Counter:1	TTL)
CheckSum:	[Input] 0=checksum disable, 1=checksum enable				
TimeOut:	[Input] Time out setting, normal=100, unit=ms.				

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;
byte addr_7080 =0x05;
byte mode = 1; // Set Counter:0-Photo  Counter:1-Photo
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7080.SetInputSignalMode(port, addr_7080, mode, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim mode As Byte = 1   'Set Counter:0-Photo  Counter:1-Photo
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7080.SetInputSignalMode(port, addr_7080, mode, checksum, timeout)
```

Remark:

This function can be applied on module: i7080.

■ PresetCounterValue

Description:

This function configures the preset value of the selected counter in the I-7080 module.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.PresetCounterValue(byte ComPort, byte address, byte counter, uint PresetValue, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

counter : [Input] 0: Configure counter 0 1: Configure counter 1

PresetValue: [Input] The counter preset value.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7080 =0x05;
byte counter0 =0;
uint PValue = 1000;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7080.PresetCounterValue(port, addr_7080, counter0, PValue,checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim counter0 As Byte = 0
Dim PValue As UInt32
PValue = Convert.ToUInt32(1000)
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7080.PresetCounterValue(port, addr_7080, counter0, PValue, checksum, timeout)
```

Remark:

This function can be applied on module: i7080.

■ ReadPresetCounterValue

Description:

This will attain the preset value of the selected counter in an I-7080 module.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.ReadPresetCounterValue( byte ComPort,
                byte address, byte counter, byte CheckSum,
                ushort TimeOut, out uint PresetValue )
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

counter : [Input] 0: Read 7080's counter 0 1: Read 7080's counter 1

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

PresetValue: [Output] The counter preset value.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7080 =0x05;
byte counter0 = 0;
byte checksum=0;
ushort timeout=100;
uint PValue;
ushort Ecode;
Ecode=Dcon.i7080.ReadPresetCounterValue(port, addr_7080, counter0,checksum, timeout, out PValue);
textBox1.Text= PValue.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
```

```
Dim addr_7080 As Byte = &H5
Dim counter0 As Byte = 0
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim PValue As UInt32
Dim Ecode As UInt16
Ecode = Dcon.i7080.ReadPresetCounterValue(port, addr_7080, counter0, checksum, timeout, PValue)
TextBox1.Text = PValue.ToString()
```

Remark:

This function can be applied on module: i7080.

■ SetModuleMode

Description:

Configures the selected counter's alarm mode for a I-7080 module. There are two counter alarm modes; alarm mode 0 and alarm mode 1. These two alarm modes can be used in both the I-7080 & I-7080D modules. Please refer to the user manual for any further information relating to this.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.SetModuleMode(byte ComPort, byte address, byte mode, byte CheckSum, ushort TimeOut)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
mode :	[Input] 0: to set into 7080 alarm mode(mode 0) 1: to set into 7080D alarm mode (mode 1)
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;
byte addr_7080 =0x05;
byte mode =0;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7080.SetModuleMode(port, addr_7080, mode, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim mode As Byte = 0
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7080.SetModuleMode(port, addr_7080, mode, checksum, timeout)
```

Remark:

This function can be applied on module: i7080.

■ ReadModuleMode

Description:

This function obtains the setting status in counter alarm mode on the I-7080 module.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.ReadModuleMode(byte ComPort, byte address, byte CheckSum, ushort TimeOut, out byte mode)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.
mode :	[Output] 0: alarm mode 0
	1: alarm mode 1

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;
byte addr_7080 =0x05;
byte checksum=0;
ushort timeout=100;
byte mode;
ushort Ecode;
Ecode=Dcon.i7080.ReadModuleMode(port, addr_7080, checksum, timeout, out mode);
textBox1.Text= mode.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim mode As Byte
Dim Ecode As UInt16
Ecode = Dcon.i7080.ReadModuleMode(port, addr_7080, checksum, timeout, mode)
TextBox1.Text = mode.ToString()
```

Remark:

This function can be applied on module: i7080.

■ SetLevelVolt

Description:

This will configure the high or low trigger level values for non-isolated inputs on I-7080 modules.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.SetLevelVolt( byte ComPort, byte address,
        byte Low_High, float LevelVolt, byte CheckSum, ushort TimeOut )
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

Low_High : [Input] 0: Set the low trigger level 1: Set the high trigger level

LevelVolt : [Input] The trigger level value.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7080 =0x05;
byte high =1;
float value = 5.5f;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7080.SetLevelVolt(port, addr_7080, high, value, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
```

```
Dim high As Byte = 1
Dim value As Single = 5.5
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7080.SetLevelVolt(port, addr_7080, high, value, checksum, timeout)
```

Remark:

This function can be applied on module: i7080.

■ ReadLevelVolt

Description:

This function acquires the high or low trigger level setting values for a non-isolated input in I-7080 module.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.ReadLevelVolt( byte ComPort, byte address,
        byte Low_High, byte CheckSum, ushort TimeOut, out float LevelVolt )
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

Low_High : [Input] 0: Set the low trigger level 1: Set the high trigger level

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

LevelVolt : [Output] The trigger level value.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7080 =0x05;
byte high = 1;
byte checksum=0;
ushort timeout=100;
float Fvalue;
ushort Ecode;
Ecode=Dcon.i7080.ReadLevelVolt(port, addr_7080, high, checksum, timeout,out Fvalue);
textBox1.Text= Fvalue.ToString("f");
```

[Visual Basic]

```
Dim port As Byte = 3
```

```
Dim addr_7080 As Byte = &H5
Dim high As Byte = 1
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim value As Single
Dim Ecode As UInt16
Ecode = Dcon.i7080.ReadLevelVolt(port, addr_7080, high, checksum, timeout, value)
TextBox1.Text = value.ToString("f")
```

Remark:

This function can be applied on module: i7080.

■ SetMinSignalWidth

Description:

This function configures the width value of either the minimum high or low input signal levels on I-7080 modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.SetMinSignalWidth(byte ComPort, byte address, byte Low_High, int MinWidth, byte CheckSum, ushort TimeOut)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
Low_High :	[Input] 0: set the min. width at low level 1: set the min. width at high level
MinWidth :	[Input] The minimum input signal width of the high or low trigger level. The value unit is uS and the corresponding range is from 2 uS to 65535 uS. For Example: when MinWidth=2000, it means the minimum with is 2 mS.
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;
byte addr_7080 =0x05;
byte low =0;
int MinWidth = 2000;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
```

```
Ecode=Dcon.i7080.SetMinSignalWidth(port, addr_7080, low, MinWidth,checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim low As Byte = 0
Dim MinWidth As Int32 = 2000
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7080.SetMinSignalWidth(port, addr_7080, low, MinWidth, checksum, timeout)
```

Remark:

This function can be applied on module: i7080.

■ ReadMinSignalWidth

Description:

This function obtains the setting width value for either minimum high or low input signal levels in I-7080 modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.ReadMinSignalWidth(byte ComPort, byte address, byte Low_High, byte CheckSum, ushort TimeOut, out int MinWidth)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

Low_High : [Input] 0: set the min. width at low level
1: set the min. width at high level

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

MinWidth : [Output] The input Signal Min Width

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7080 =0x05;
byte low = 0;
byte checksum=0;
ushort timeout=100;
int MinWidth;
ushort Ecode;
Ecode=Dcon.i7080.ReadMinSignalWidth(port, addr_7080, low, checksum, timeout, out MinWidth);
textBox1.Text= MinWidth.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim low As Byte = 0
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim MinWidth As Int32
Dim Ecode As UInt16
Ecode = Dcon.i7080.ReadMinSignalWidth(port, addr_7080, low, checksum, timeout, MinWidth)
TextBox1.Text = MinWidth.ToString()
```

Remark:

This function can be applied on module: i7080.

■ SetGateMode

Description:

This will configure the gate control mode of I-7080 modules. There are 3 mode types:

0: gate input signal must be low to enable counter

1: gate input signal must be high to enable counter

2: gate input signal is ignored. The counter will be always enable

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.SetGateMode(byte ComPort, byte address,
byte Low_High_None, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

Low_High_None:[Input] 0: the gate is low active 1: the gate is high active
2: the gate is disable

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7080 =0x05;
byte low =0;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7080.SetGateMode(port, addr_7080, low, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim low As Byte = 1
Dim checksum As Byte = 0
Dim timeout As UInt16= Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7080.SetGateMode(port, addr_7080, low, checksum, timeout)
```

Remark:

This function can be applied on module: i7080.

■ ReadGateMode

Description:

This function attains the setting status on the gate control mode in I-7080 modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.ReadGateMode(byte ComPort, byte address,
byte CheckSum, ushort TimeOut, out byte GateMode)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

GateMode : [Output] 0: gate control mode is low active
1: gate control mode is high active
2: gate control mode is disable

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7080 =0x05;
byte checksum=0;
ushort timeout=100;
byte mode;
ushort Ecode;
Ecode=Dcon.i7080.ReadGateMode(port, addr_7080, checksum, timeout, out mode);
textBox1.Text= mode.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim checksum As Byte = 0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim mode As Byte
Dim Ecode As UInt16
Ecode = Dcon.i7080.ReadGateMode(port, addr_7080, checksum, timeout, mode)
TextBox1.Text = mode.ToString()
```

Remark:

This function can be applied on module: i7080.

■ ReadCounterStatus

Description:

This function obtains the counter working status (reading/stop) of I-7080 modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.ReadCounterStatus(byte ComPort, byte address, byte Counter,byte CheckSum, ushort TimeOut, out byte CounterStatus)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

counter : [Input] 0: Set counter 0 1: Set counter 1

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

CounterStatus:[Output] 0: Counting. 1: Stop.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7080 =0x05;
byte checksum=0;
ushort timeout=100;
byte status;
byte Counter=0;
ushort Ecode;
Ecode=Dcon.i7080.ReadCounterStatus(port, addr_7080,Counter, checksum, timeout,out status);
textBox1.Text= status.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim checksum As Byte =0
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim status As Byte
Dim Ecode As UInt16
Ecode=Dcon.i7080.ReadCounterStatus(port, addr_7080, checksum, timeout,
        status)
textBox1.Text= status.ToString()
```

Remark:

This function can be applied on module: i7080.

■ SetConfiguration

Description:

This function will set the configuration of I-7080 or I-7080D modules.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.SetConfiguration( byte ComPort, byte address,
        byte frequency_time, byte New_Address, byte Type, ushort Baudrate,
        byte CheckSum_Address, byte CheckSum, ushort TimeOut )
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

frequency_time: [Input] Desired frequency gate time:

0: 0.1 second 1: 1.0 second

Don't care 'frequency_time', if set the module in Counter mode.

New_Address: [Input] Desired new address

Type : [Input] Desired Type= 1:Counter mode 0:Frequency mode

Baudrate : [Input] Desired Baudrate:

3: 1200 BPS 4: 2400 BPS 5: 4800 BPS 6: 9600 BPS

7: 19200 BPS 8: 38400 BPS 9: 57600 BPS 10: 115200 BPS

CheckSum_Address:[Input] Desired Checksum Address

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7080 =0x05;
byte checksum=0;
ushort timeout=100;
```

```
byte ftime =0;
byte newaddr =0x06;
byte type = 1;  // Counter mode
ushort baudrate = 7;
byte newchksum = 1;
ushort Ecode;
Ecode=Dcon.i7080.SetConfiguration(port, addr_7080, ftime, newaddr , type,baudrate, newchksum, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7080 As Byte = &H5
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim ftime As Byte = 0
Dim newaddr As Byte = &H6
Dim baudrate As UInt16 = Convert.ToUInt16(7)
Dim newchksum As Byte = 1
Dim Ecode As UInt16
Dim C_Type As Byte = 1
Ecode = Dcon.i7080.SetConfiguration(port, addr_7080, ftime, newaddr, C_Type, baudrate, newchksum, checksum, timeout)
```

Remark:

This function can be applied on module: i7080.

■ DataToLED

Description:

This will output defined data to the LED display on I-7080D modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.DataToLED(byte ComPort, byte address,
float fOutValue, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

fOutValue : [Input] Output data to LED display.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;    byte addr_7080D =0x06;
float data = 5.5f;    byte checksum=0;
ushort timeout=100;   ushort Ecode;
Ecode=Dcon.i7080.DataToLED(port, addr_7080D, data, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3 : Dim addr_7080D As Byte = &H6
Dim data As Single = 5.5 :Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7080.DataToLED(port, addr_7080D, data, checksum, timeout)
```

Remark:

This function can be applied on module: i7080D.

4.8 Strain Gauge Functions

■ SetupLinearMapping

Description:

This function configures the linear mapping translation of I-7014 or I-7016 modules from the raw data range to the target range data value. However, before using this function, users need to get the module's range code by calling the ReadConfigStatus() and setting it into wBuf [7]. That is, this function provides linear mapping from the following range areas [a, b] to [c, d], where fBuf[0]=a, fBuf[1]=b, fBuf[2]=c, fBuf[3]=d.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.SetupLinearMapping( byte ComPort, byte address,
                    float fSourceLow, float fSourceHigh, float fTargetLow,
                    float fTargetHigh, byte CheckSum, ushort TimeOut)
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

fSourceLow : [Input] Source low value, a

fSourceHigh: [Input] Source high value, b

fTargetLow : [Input] Target low value, c

fTargetHigh: [Input] Target high value, d

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7016 =0x03;
```

```
float Slow = 0;
float Shigh = 20;
float Tlow = 0;
float Thigh = 100;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7016.SetupLinearMapping(port, addr_7016, Slow, Shigh, Tlow, Thigh, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7016 As Byte = &H3
Dim Slow As Single = 0
Dim Shigh As Single = 20
Dim Tlow As Single = 0
Dim Thigh As Single = 100
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7016.SetupLinearMapping(port, addr_7016, Slow, Shigh, Tlow, Thigh, checksum, timeout)
```

Remark:

This function can be applied on module: i7014, i7016.

■ EnableLinearMapping

Description:

This enables the linear mapping function on either of the I-7014 or I-7016 modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.EnableLinearMapping(byte ComPort, byte address, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3; byte addr_7016 =0x03;
byte checksum=0; ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7016.EnableLinearMapping(port, addr_7016, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7016 As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7016.EnableLinearMapping(port, addr_7016, checksum, timeout)
```

Remark:

This function can be applied on module: i7014, i7016.

■ DisableLinearMapping

Description:

This function disables the linear mapping function on I-7014 or I-7016 modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.DisableLinearMapping(byte ComPort, byte address, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;byte addr_7016 =0x03;
byte checksum=0; ushort timeout=100;
ushort Ecode;
Ecode=Dcon.i7016.DisableLinearMapping(port, addr_7016, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7016 As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7016.DisableLinearMapping(port, addr_7016, checksum, timeout)
```

Remark:

This function can be applied on module: i7014, i7016.

■ ReadLinearMappingStatus

Description:

This function attains the linear mapping function status on I-7014 or I-7016 modules.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.ReadLinearMappingStatus( byte ComPort,
        byte address, byte CheckSum, ushort TimeOut, out byte MapStatus )
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

MapStatus : [Output] 0: linear mapping is disable 1: linear mapping is enable

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7016 =0x03;
byte checksum=0;
ushort timeout=100;
byte mapStatus;
ushort Ecode;
Ecode=Dcon.i7016.ReadLinearMappingStatus(port, addr_7016, checksum, timeout, out mapStatus);
textBox1.Text = mapStatus.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7016 As Byte = &H3
Dim checksum As Byte = 0
```

```
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim mapStatus As Byte
Dim Ecode As UInt16
Ecode = Dcon.i7016.ReadLinearMappingStatus(port, addr_7016, checksum, timeout, mapStatus)
TextBox1.Text = mapStatus.ToString()
```

Remark:

This function can be applied on module: i7014, i7016.

■ ReadSourceValueOfLM

Description:

This function obtains the setting value of the Linear Mapping source range for I-7014/7016 modules.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.ReadSourceValueOfLM( byte ComPort,
                byte address, byte CheckSum, ushort TimeOut,
                out float LowValue, out float HighValue )
```

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.
LowValue:	[Output] Low Source Value
HighValue:	[Output] High Source Value

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;
byte addr_7016 =0x03;
byte checksum=0;
ushort timeout=100;
float LowVal;
float HighVal;
ushort Ecode;
Ecode=Dcon.i7016.ReadSourceValueOfLM(port, addr_7016, checksum, timeout,out LowVal, out HighVal);
textBox1.Text = LowVal.ToString("f");
textBox2.Text = HighVal.ToString("f");
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7016 As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim LowVal As Single
Dim HighVal As Single
Dim Ecode As UInt16
Ecode = Dcon.i7016.ReadSourceValueOfLM(port, addr_7016, checksum, timeout, LowVal, HighVal)
TextBox1.Text = LowVal.ToString("f")
textBox2.Text = HighVal.ToString("f")
```

Remark:

This function can be applied on module: i7014, i7016.

■ ReadTargetValueOfLM

Description:

This function obtains the setting value of Linear Mapping source ranges for I-7014/7016 modules.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.ReadTargetValueOfLM( byte ComPort,
                byte address, byte CheckSum, ushort TimeOut,
                out float LowValue, out float HighValue )
```

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.
LowValue:	[Output] Low Target Value
HighValue:	[Output] High Target Value

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;
byte addr_7016 =0x03;
byte checksum=0;
ushort timeout=100;
float LowVal;
float HighVal;
ushort Ecode;
Ecode=Dcon.i7016.ReadTargetValueOfLM(port, addr_7016, checksum, timeout,out LowVal, out HighVal);
textBox1.Text = LowVal.ToString("f");
textBox2.Text = HighVal.ToString("f");
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7016 As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim LowVal As Single
Dim HighVal As Single
Dim Ecode As UInt16
Ecode = Dcon.i7016.ReadTargetValueOfLM(port, addr_7016, checksum, timeout, LowVal, HighVal)
TextBox1.Text = LowVal.ToString("f")
textBox2.Text = HighVal.ToString("f")
```

Remark:

This function can be applied on module: i7014, i7016.

4.9 Dual Watchdog Functions

■ HostIsOK

Description:

This function provides a method to tell all modules that the“Host PC is OK” by sending this command string “~**”. If the module can’t receive “~**” during a time interval, the host “WatchDog” function will be enabled. The related functions are “ToSetupHostWatchdog” and “ ToReadHostWatchdog”.

Syntax:

[Visual Basic, C#]

ushort Dcon.Watchdog.HostIsOK(byte ComPort, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
// Use in Timer function
byte port=3;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode= Dcon.Watchdog.HostIsOK(port, checksum, timeout);
```

[Visual Basic]

```
' Use in Timer function
Dim port As Byte = 3
Dim checksum As Byte = 0
```

```
Dim timeout As UInt16
timeout = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.Watchdog.HostIsOK(port, checksum, timeout)
```

Remark:

■ ToSetupHostWatchdog

Description:

This will configure the timer working interval of the Host Watchdog for I-7000 series modules. It can also enable or disable the “WatchDog” function.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.Watchdog.ToSetupHostWatchdog( byte ComPort, byte address,
                byte HostWatchdog, ushort IntervalTime, byte CheckSum,
                ushort TimeOut )
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

HostWatchdog:[Input] 0: Disable host watchdog 1: Enable host watchdog
If TimeOut value = 0 then disable the Host Watchdog

IntervalTime: [Input] Timer setting for watchdog, unit is 0.1 second.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr=0x03;
byte hwatchdog = 1;
ushort iTime= 50;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode= Dcon.Watchdog.ToSetupHostWatchdog(port, addr, hwatchdog, iTime, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr As Byte = &H3
Dim hwatchdog As Byte = 1
Dim iTime As UInt16 = Convert.ToUInt16(50)
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.Watchdog.ToSetupHostWatchdog(port, addr, hwatchdog, iTime, checksum, timeout)
```

Remark:

■ ToReadHostWatchdog

Description:

This function configures the Host Watchdog's timer working interval for I-7000 series modules. It can also enable or disable the "WatchDog" function.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.Watchdog.ToReadHostWatchdog( byte ComPort, byte address,
                byte CheckSum, ushort TimeOut, out byte HostWatchdog,
                out ushort IntervalTime )
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

HostWatchdog:[Output] 0: Host watchdog is disabled.
1: Host watchdog is enabled.

IntervalTime: [Output] Timer setting value for watchdog, unit is 0.1 second.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr=0x03;
byte checksum=0;
ushort timeout=100;
byte hwatchdog;
ushort iTime;
ushort Ecode;
Ecode= Dcon.Watchdog.ToReadHostWatchdog(port, addr, checksum, timeout, out hwatchdog, out iTime);
textBox1.Text = hwatchdog.ToString();
textBox2.Text = iTime.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim hwatchdog As Byte
Dim iTime As UInt16
Dim Ecode As UInt16
Ecode = Dcon.Watchdog.ToReadHostWatchdog(port, addr, checksum, timeout, hwatchdog, iTime)
TextBox1.Text = hwatchdog.ToString()
textBox2.Text = iTime.ToString()
```

Remark:

■ ReadModuleResetStatus

Description:

This function attains the module reset status. If the return value is “0”, it means the “module has not been reset”. If the return value is “1”, it means “the module has been reset”.

Syntax:

[Visual Basic, C#]

ushort Dcon.Watchdog.ReadModuleResetStatus(byte ComPort, byte address, byte CheckSum, ushort TimeOut, out byte ResetValue)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

ResetValue: [Output] 0: module has not been reset
1: module has been reset

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr = 0x03;
byte checksum=0;
ushort timeout=100;
byte Bvalue;
ushort Ecode;
Ecode= Dcon.Watchdog.ReadModuleResetStatus(port, addr, checksum, timeout,out Bvalue);
textBox1.Text = Bvalue.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim value As Byte
Dim Ecode As UInt16
Ecode = Dcon.Watchdog.ReadModuleResetStatus(port, addr, checksum, timeout, value)
TextBox1.Text = value.ToString()
```

Remark:

■ ReadModuleHostWatchdogStatus

Description:

This function obtains the module's Host Watchdog status for I-7000 series modules. If the return value is "0", it means the "Module's Host watchdog is in NORMAL mode". If the return value is "4", it means the "Module's Host watchdog is in HOST FAILURE mode".

Syntax:

[Visual Basic, C#]

```
ushort Dcon.Watchdog.ReadModuleHostWatchdogStatus( byte ComPort,
                byte address, byte CheckSum, ushort TimeOut,
                out byte ModuleHostWatchdog)
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

ModuleHostWatchdog:[Output]

0: Module's Host watchdog is in NORMAL mode.

4: Module's Host watchdog is in HOST FAILURE mode.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr = 0x03;
byte checksum=0;
ushort timeout=100;
byte Bvalue;
ushort Ecode;
Ecode= Dcon.Watchdog.ReadModuleHostWatchdogStatus(port, addr, checksum, timeout, out Bvalue);
```

```
textBox1.Text = Bvalue.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim value As Byte
Dim Ecode As UInt16
Ecode = Dcon.Watchdog.ReadModuleHostWatchdogStatus(port, addr, checksum, timeout, value)
TextBox1.Text = value.ToString()
```

Remark:

■ ResetModuleHostWatchdogStatus

Description:

This will reset the module's Host Watchdog status on I-7000 series modules. The related function is "ReadModuleHostWatchdogStatus".

Syntax:

[Visual Basic, C#]

```
ushort Dcon.Watchdog.ResetModuleHostWatchdogStatus( byte ComPort,
        byte address, byte CheckSum, ushort TimeOut )
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr = 0x03;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode= Dcon.Watchdog.ResetModuleHostWatchdogStatus(port, addr,checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.Watchdog.ResetModuleHostWatchdogStatus(port, addr, checksum, timeout)
```

Remark:

■ SetSafeValueForDo

Description:

This configures the safe value of DO modules on I-7000 series modules when the “WatchDog” function on the module has been enabled.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.SetSafeValueForDo(byte ComPort, byte address, Data_Type cdata, byte CheckSum, ushort TimeOut)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
cdata :	[Input] Safe digital output value in Hex format.
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;
byte addr_7060 = 0x03;
byte data = 0x0F;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode= Dcon.i7060.SetSafeValueForDo(port, addr_7060, data, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7060 As Byte = &H3
Dim data As Byte = &HF
Dim checksum As Byte = 0
```

```
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7060.SetSafeValueForDo(port, addr_7060, data, checksum, timeout)
```

Remark:

1.This Data_Type can be applied on data type: byte, ushort.

2.This function can be applied on module: i7042, i7043, i7044, i7050, i7060, i7063, i7065, i7066, i7067.

■ SetPowerOnValueForDo

Description:

This function will configure the initial digital output value for digital output modules on I-7000 series modules when their power is on.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.SetPowerOnValueForDo(byte ComPort,
byte address, Data_Type cdata, byte CheckSum, ushort TimeOut)

Parameter:

ComPort :	[Input] 2=COM2, 3=COM3
address :	[Input] Module address, from 0x00 to 0xFF
cdata :	[Input] Power On value in Hex format.
CheckSum:	[Input] 0=checksum disable, 1=checksum enable
TimeOut:	[Input] Time out setting, normal=100, unit=ms.

Return Value:

0:	NO_ERROR
Others:	Error code

Example:

[C#]

```
byte port=3;
byte addr_7060 = 0x03;
byte data = 0x0F;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode= Dcon.i7060.SetPowerOnValueForDo(port, addr_7060, data, checksum,    timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7060 As Byte = &H3
Dim data As Byte = &HF
Dim checksum As Byte = 0
```

```
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7060.SetPowerOnValueForDo(port, addr_7060, data, checksum, timeout)
```

Remark:

1.This Data_Type can be applied on data type: byte, ushort.

2.This function can be applied on module: i7042, i7043, i7044, i7050, i7060, i7063, i7065, i7066, i7067.

■ SetSafeValueForAo

Description:

This will configure channel No safe values for analog output modules on I-7000 series modules when the “WatchDog” function of the module has been enabled.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.SetSafeValueForAo(byte ComPort, byte address,
[byte channel], float SafeValue, byte CheckSum, ushort TimeOut)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

channel: [Input] The analog output channel No.

SafeValue : [Input] Analog output safe value.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7021 = 0x02;
float Fvalue = 5;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode= Dcon.i7021.SetSafeValueForAo(port, addr_7021, Fvalue, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7021 As Byte = &H2
Dim value As Single = 5
```

```
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7021.SetSafeValueForAo(port, addr_7021, value, checksum, timeout)
```

Remark:

This function can be applied on module: i7021, i7022, i7024.

■ SetPowerOnValueForAo

Description:

This function configures the initial analog output for analog output modules on I-7000 series modules when thier power is on.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.SetPowerOnValueForAo( byte ComPort,
                byte address, [ byte channel ], float PowerOnValue,
                byte CheckSum, ushort TimeOut )
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

channel: [Input] The analog output channel No.

PowerOnValue:[Input] Power On analog output value.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7021 = 0x02;
float Fvalue = 5;
byte checksum=0;
ushort timeout=100;
ushort Ecode;
Ecode= Dcon.i7021.SetPowerOnValueForAo(port, addr_7021, Fvalue, checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7021 As Byte = &H2
Dim value As Single = 5
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7021.SetPowerOnValueForAo(port, addr_7021, value, checksum, timeout)
```

Remark:

This function can be applied on module: i7021, i7022, i7024.

■ SetPowerOnSafeValue

Description:

This function will configure the power on and safe values for digital output channels on I-7011, I-7012, and I-7014 modules.

Power On value:

00:DO0 off, DO1 off; 01:DO0 on, DO1 off;
02: DO0 off, DO1 on; 03: DO0 on, DO1 on;

Safe value:

00: DO0 off, DO1 off; 01: DO0 on, DO1 off;
02: DO0 off, DO1 on; 03: DO0 on, DO1 on;

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.SetPowerOnSafeValue( byte ComPort,
                byte address, byte PowerOnValue, byte SafeValue,
                byte CheckSum, ushort TimeOut )
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

PowerOnValue:[Input] Power On value in hex format.

SafeValue : [Input] Safe value in hex format.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7011 = 0x03;
byte power = 0x03;
byte safe =0x03;
byte checksum=0;
```

```
ushort timeout=100;
ushort Ecode;
Ecode= Dcon.i7011.SetPowerOnSafeValue(port, addr_7011, power, safe,  checksum, timeout);
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7011 As Byte = &H3
Dim power As Byte = &H3
Dim safe As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim Ecode As UInt16
Ecode = Dcon.i7011.SetPowerOnSafeValue(port, addr_7011, power, safe, checksum, timeout)
```

Remark:

This function can be applied on module: i7011, i7012, i7014.

■ ReadSafeValueForAo

Description:

This will attain the safe setting value of analog output modules for I-7000 series modules.

Syntax:

[Visual Basic, C#]

ushort Dcon.ModuleName.ReadSafeValueForAo(byte ComPort, byte address,
[byte channel,] byte CheckSum, ushort TimeOut, out float SafeValue)

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

channel : [Input] The analog output channel No.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

SafeValue: [Output] Safe value.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7021 = 0x02;
byte checksum=0;
ushort timeout=100;
float Fvalue;
ushort Ecode;
Ecode= Dcon.i7021.ReadSafeValueForAo(port, addr_7021, checksum, timeout, out Fvalue);
textBox1.Text = Fvalue.ToString("f");
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7021 As Byte = &H2
```

```
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim value As Single
Dim Ecode As UInt16
Ecode = Dcon.i7021.ReadSafeValueForAo(port, addr_7021, checksum, timeout, value)
TextBox1.Text = value.ToString("f")
```

Remark:

This function can be applied on module: i7021, i7022, i7024.

■ ReadPowerOnValueForAo

Description:

This will obtain the initial output setting value of analog output module for I-7000 series module when the power is on.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.ReadPowerOnValueForAo( byte ComPort,
        byte address, byte channel, byte CheckSum, ushort TimeOut,
        out float PowerOnValue)
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

channel : [Input] The analog output channel No.

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

PowerOnValue:[Output] The initial output value when the power is on.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7024 = 0x05;
byte ch0 = 0;
byte checksum=0;
ushort timeout=100;
float Fvalue;
ushort Ecode;
Ecode= Dcon.i7024.ReadPowerOnValueForAo(port, addr_7024, ch0, checksum,timeout, out Fvalue);
textBox1.Text = Fvalue.ToString("f");
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7024 As Byte = &H5
Dim ch0 As Byte = 0
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim value As Single
Dim Ecode As UInt16
Ecode = Dcon.i7024.ReadPowerOnValueForAo(port, addr_7024, ch0, checksum, timeout, value)
TextBox1.Text = value.ToString("f")
```

Remark:

This function can be applied on module: i7024.

■ ReadPowerOnValueForDo

Description:

This function obtains the initial output setting value for digital output modules on I-7000 series modules when the power is on.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.ReadPowerOnValueForDo( byte ComPort,
                byte address, byte CheckSum, ushort TimeOut,
                out Data_Type PowerOnValue)
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

PowerOnValue:[Output] Power on value in hex format.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7060 = 0x03;
byte checksum=0;
ushort timeout=100;
byte data;
ushort Ecode;
Ecode= Dcon.i7060.ReadPowerOnValueForDo(port, addr_7060, checksum,timeout, out data);
textBox1.Text = data.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
```

```
Dim addr_7060 As Byte = &H3
Dim checksum As Byte = 0
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim data As Byte
Dim Ecode As UInt16
Ecode = Dcon.i7060.ReadPowerOnValueForDo(port, addr_7060, checksum, timeout, data)
TextBox1.Text = data.ToString()
```

Remark:

1.This Data_Type can be applied on data type: byte, ushort.

This function can be applied on module: i7042, i7043, i7044, i7050, i7060, i7063, i7065, i7066, i7067.

■ ReadSafeValueForDo

Description:

This function obtains the safe setting value of digital output modules on I-7000 series modules when the “WatchDog” function on the module has been enabled.

Syntax:

[Visual Basic, C#]

```
ushort Dcon.ModuleName.ReadSafeValueForDo( byte ComPort,
    byte address, byte CheckSum, ushort TimeOut, out Data_Type SafeValue)
```

Parameter:

ComPort : [Input] 2=COM2, 3=COM3

address : [Input] Module address, from 0x00 to 0xFF

CheckSum: [Input] 0=checksum disable, 1=checksum enable

TimeOut: [Input] Time out setting, normal=100, unit=ms.

PowerOnValue:[Output] Safe Value in hex format.

Return Value:

0: NO_ERROR

Others: Error code

Example:

[C#]

```
byte port=3;
byte addr_7060 = 0x03;
byte checksum=0;
ushort timeout=100;
byte data;
ushort Ecode;
Ecode= Dcon.i7060.ReadSafeValueForDo(port, addr_7060, checksum, timeout, out data);
textBox1.Text = data.ToString();
```

[Visual Basic]

```
Dim port As Byte = 3
Dim addr_7060 As Byte = &H3
Dim checksum As Byte = 0
```

```
Dim timeout As UInt16 = Convert.ToUInt16(100)
Dim data As Byte
Dim Ecode As UInt16
Ecode = Dcon.i7060.ReadSafeValueForDo(port, addr_7060, checksum, timeout, data)
TextBox1.Text = data.ToString()
```

Remark:

1.This Data_Type can be applied on data type: byte, ushort.

2.This function can be applied on module: i7042, i7043, i7044, i7050, i7060, i7063, i7065, i7066, i7067.

4.10 Error Code Table

Value	Description
0	Functions work normally.
1	Call wrong function error.
2	Use wrong COM Port error.
3	Baud rate error.
4	Data Bit error.
5	Stop Bit error.
6	Parity Bit error.
7	CheckSum mechanism error.
8	COM is not open error.
9	Send thread create error.
10	Send command error.
11	Read COM Port status error.
12	Result string check error.
13	Command error.
15	TimeOut error.
17	Module ID error.
18	Channel number error.
19	Under input range error.
20	Exceed input range error.
21	Invalidate counter number error.
22	Invalidate counter value error.
23	Invalidate gate mode error.
24	Invalidate channel No error.
25	COM Port is in use error.

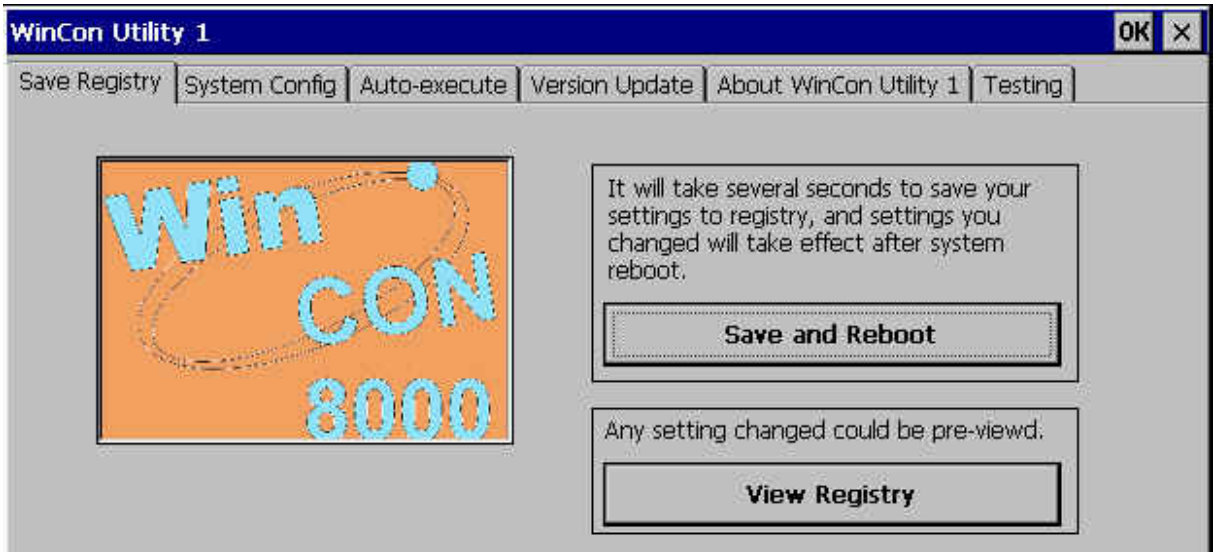


Fig. 9-20

1. The Remote Agent program (CESERVER.EXE) will be started automatically when the Wincon-8xx9 controller's power is turned on. Then your IWS application will start up when the WinCE system is launched.