

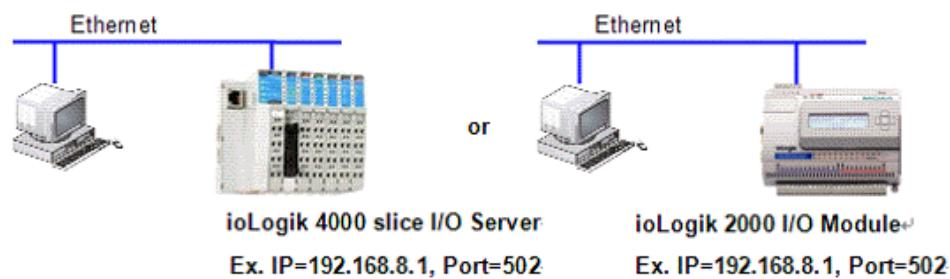
Programming Flow

Accessing an ioLogik I/O device is almost the same for both Ethernet and Serial interfaces. Five different scenarios are described below.

2.1 Connecting a Single Ethernet I/O device

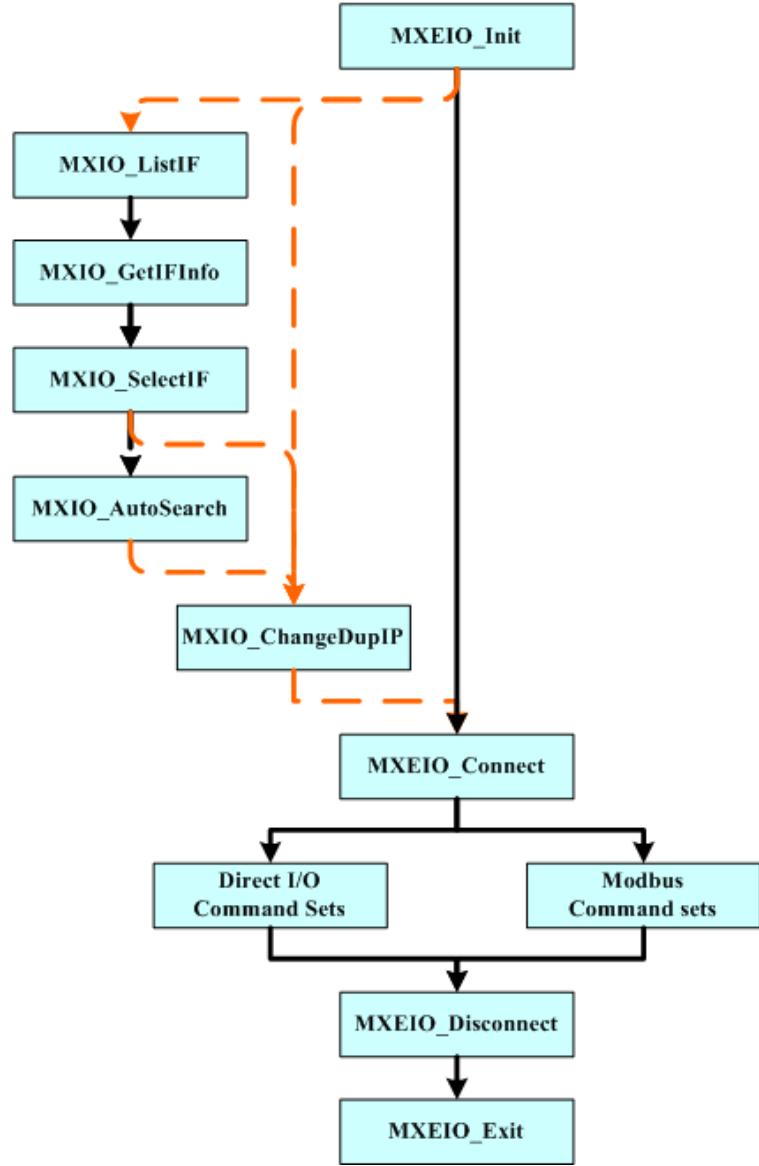
The MXIO DLL library running on the host computer establishes a data tunnel using Modbus commands to communicate with the Ethernet I/O device. The TCP port number used to connect the ioLogik 4000 Server, ioLogik 2000 module, ioLogik 4200 Server, ioLogik 5000 module and ioLogik 1000 module is usually 502.

Connecting Single Ethernet I/O Server



Using the MXIO DLL library to access data from the I/O device requires 5 steps. First, use MXEIO_Init to initialize Windows TCP/IP service. Then, use MXEIO_Connect to connect to the Ethernet I/O device using an IPTCP Port pair (e.g., 192.168.8.1:502). MXEIO_Connect should return a handle. You can use the handle to access any of the I/O points by Modbus Command Sets or Direct I/O Command Sets. To finish the operation, use MXEIO_Disconnect and MXEIO_Exit to release the Windows system resources.

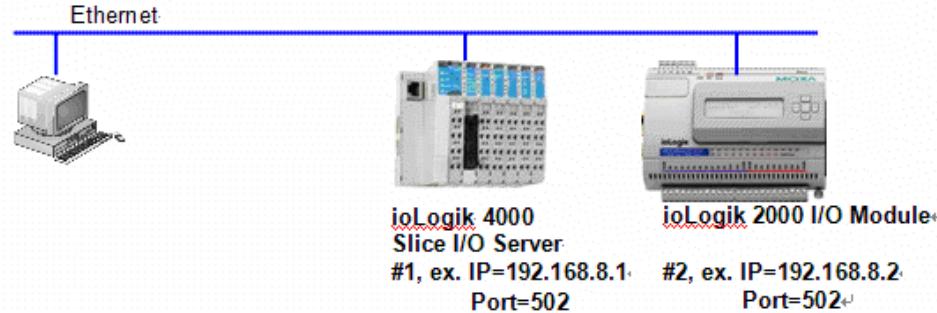
Program Flow I.
Connecting Single Ethernet I/O device



2.2 Connecting Multiple Ethernet I/O devices

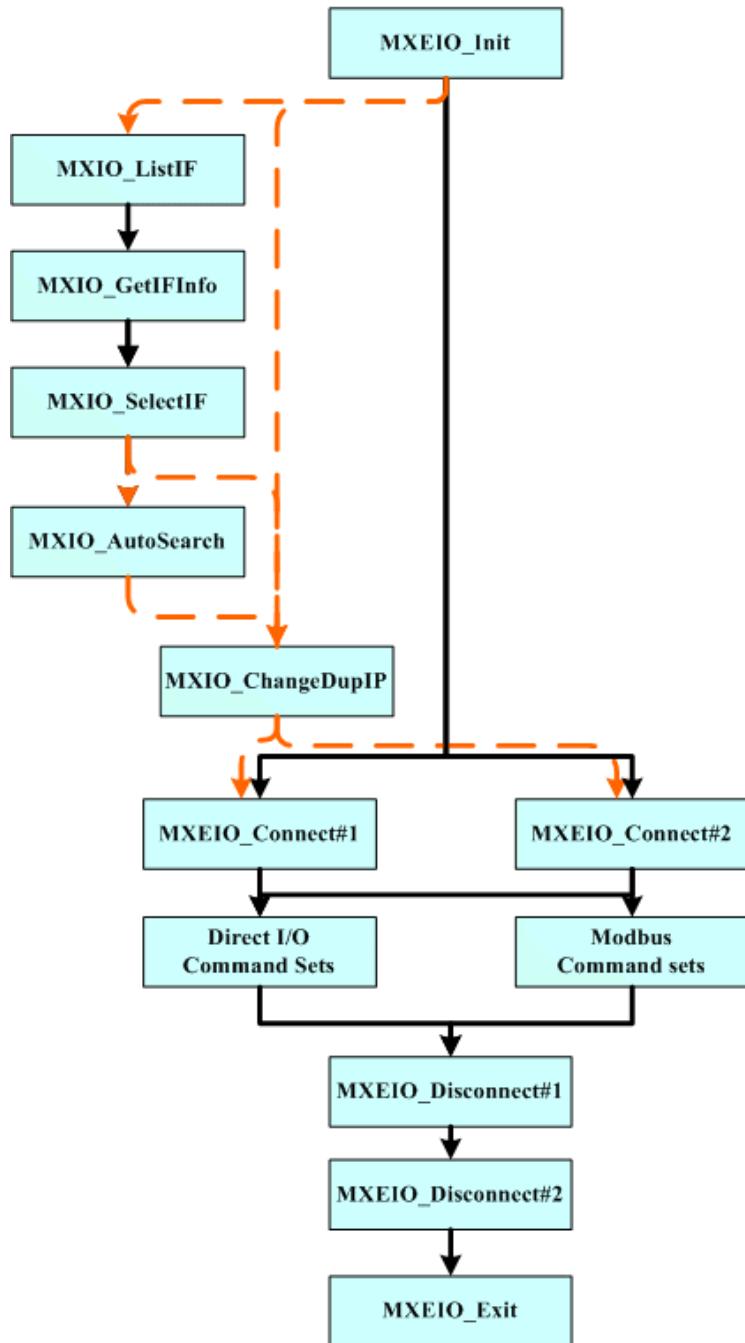
To use multiple Ethernet I/O devices over the network, each Ethernet I/O device should have a unique IP address.

Connecting Multiple Ethernet I/O devices



We explained above how to access a single Ethernet I/O device. But what about accessing additional Ethernet I/O devices? Each Ethernet I/O device needs a unique handle. The only difference is that you get the handle using `MXEIO_Connect` for each Ethernet I/O device.

Program Flow II.
Connecting Multiple Ethernet I/O devices



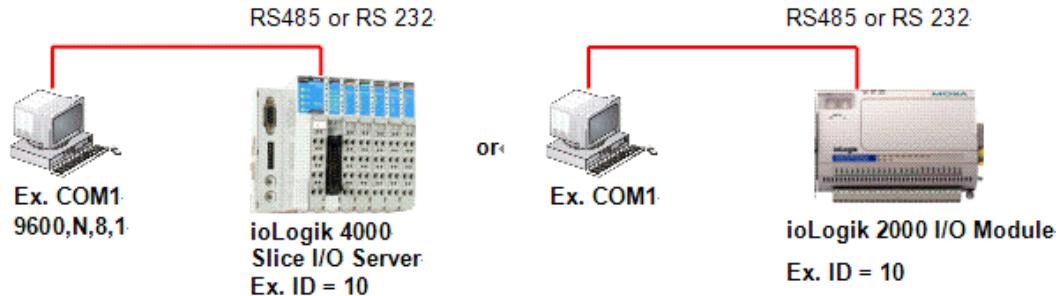
2.3 Connecting a Single Serial I/O device

ioLogik 4000 and ioLogik 2000 also support the traditional Serial I/O device to cope with old-fashioned RS-485, RS-232 control networks. When

accessing the Serial I/O device via RS-485 or RS-232, please pay attention to the following:

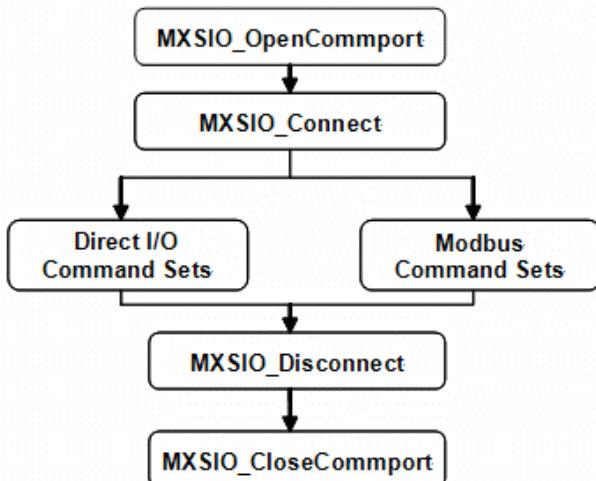
1. Your computer must be equipped with an RS-232 or RS-485 communication port.
2. Make sure the baud rate and communication parameters for the computer and the Serial I/O device are identical.
3. Make sure the Serial I/O device is running under Modbus/RTU

Connecting Single Serial I/O device



Using MXIO DLL library to access data from the Serial I/O device requires 5 steps. First, use MX_OpenCommport to open the COM port and connect to the Serial I/O device. Then, use MXSIO_Connect to connect to the Serial I/O device using the UNIT ID (e.g., 10). MXSIO_Connect should return a handle. You may use the handle to access any of the I/O points by Modbus Command Sets or Direct I/O Command Sets. To finish the operation, use MXSIO_Disconnect and MXSIO_CloseCommport to release the Windows system resources.

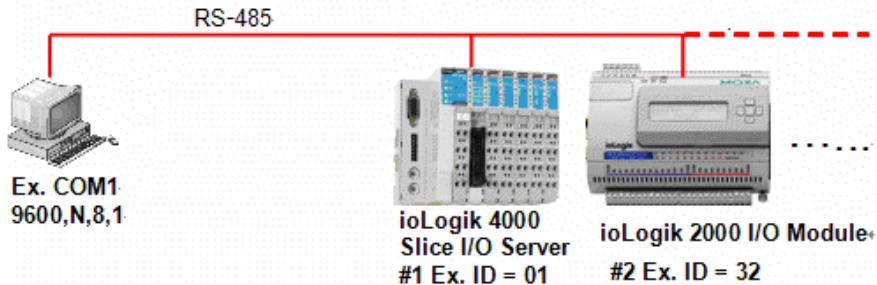
Program Flow III. Connecting Single RS-485 I/O device



2.4 Connecting Multiple RS-485 I/O devices

In most real world applications, multiple RS-485 I/O devices are often connected to the same network. One RS-485 network can support up to 32 nodes.

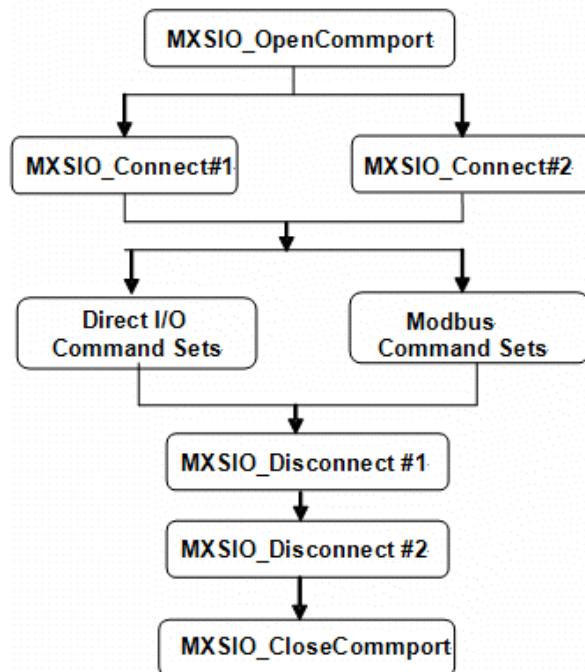
Connecting Multiple RS-485 I/O devices



Each Serial I/O device requires a unique handle. Make sure each Serial I/O device already has its own handle before accessing the I/O points.

Program Flow IV.

Connecting Multiple Serial I/O device



2.6 Modbus Command Sets vs. Direct I/O Command Sets

MXIO Library supports two ways of letting you access the I/O data from ioLogik 4000 I/O Server and ioLogik 2000 I/O module.

Modbus Command Sets

ioLogik 4000 I/O Server and ioLogik 2000 I/O module use Modbus/TCP,

Modbus/RTU to communicate with the host computer. Modbus Command Sets is a simplified API that uses the Modbus protocol data format to get I/O data. This is a good choice if you are already familiar with the Modbus protocol and prefer using the Modbus data structure.

Direct I/O Command Sets

If you don't like the complex data structure of Modbus, the MXIO Library provides an intuitive way to achieve the same result. Each I/O point or channel can be accessed by a physical slot number and channel number. This is a good choice for accessing the I/O quickly and easily.

Return Codes

Return Value	Value	Description
MXIO_OK	0	Function call was successful.
ILLEGAL_FUNCTION	1001	The function code received in the query is not an allowable action for the server (or slave).
ILLEGAL_DATA_ADDRESS	1002	The data address received in the query is not an allowable address for the server (or slave).
ILLEGAL_DATA_VALUE	1003	A value contained in the query data field is not an allowable value for the server (or slave).
SLAVE_DEVICE_FAILURE	1004	An unrecoverable error occurred while the server (or slave) was attempting to perform the requested action.
SLAVE_DEVICE_BUSY	1006	Specialized use in conjunction with programming commands. The server (or slave) is engaged in processing a long-duration program command. The client (or master) should retransmit the message later when the server (or slave) is free.
EIO_TIME_OUT	2001	The following situation may cause an EIO_TIME_OUT : 1. Open socket timeout. 2. Send command to the I/O Server timeout.

		3. I/O Server Response timeout.
EIO_INIT_SOCKETS_FAIL	2002	The error occurred when the Windows system couldn't complete SOCKET INIT.
EIO_CREATING_SOCKET_ERROR	2003	The error occurred when the Windows system couldn't initiate Socket.
EIO_RESPONSE_BAD	2004	The data received from Ethernet I/O server is incorrect.
EIO_SOCKET_DISCONNECT	2005	The network connection from host computer is down.
PROTOCOL_TYPE_ERROR	2006	Protocol type error.
EIO_PASSWORD_INCORRECT	2007	Password Incorrect
SIO_OPEN_FAIL	3001	Open COM port Fail.
SIO_TIME_OUT	3002	Unable to communicate to the COM port in the designated time.
SIO_CLOSE_FAIL	3003	Unable to close the COM port.
SIO_PURGE_COMM_FAIL	3004	Purge COM port error
SIO_FLUSH_FILE_BUFFERS_FAIL	3005	Flush File Buffers error
SIO_GET_COMM_STATE_FAIL	3006	Get COM port Status error
SIO_SET_COMM_STATE_FAIL	3007	Set COM port Status error
SIO_SETUP_COMM_FAIL	3008	Setup COM port error
SIO_SET_COMM_TIME_OUT_FAIL	3009	Set COM port read timeout and write timeout fail
SIO_CLEAR_COMM_FAIL	3010	Clear COM port
SIO_RESPONSE_BAD	3011	The data received from the Serial I/O server is incorrect.
SIO_TRANSMISSION_MODE_ERROR	3012	Modbus transmission

		parameter error while calling MXSIO_Connect().
SIO_BAUDRATE_NOT_SUPPORT	3013	Serial Communication baudrate not support.
PRODUCT_NOT_SUPPORT	4001	The I/O module is not supported by this version of MXIO DLL.
HANDLE_ERROR	4002	Handle error.
SLOT_OUT_OF_RANGE	4003	Slot out of range.
CHANNEL_OUT_OF_RANGE	4004	Channel out of range.
COIL_TYPE_ERROR	4005	Coil Type error.
REGISTER_TYPE_ERROR	4006	Register Type error.
FUNCTION_NOT_SUPPORT	4007	Function is not supported for designated I/O module.
OUTPUT_VALUE_OUT_OF_RANGE	4008	The output value is out of the output range.
INPUT_VALUE_OUT_OF_RANGE	4009	The input value is out of the input range.
SLOT_NOT_EXIST	4010	The slot was not existed.
FIRMWARE_NOT_SUPPORT	4011	The device firmware is not support this function.
CREATE_MUTEX_FAIL	4012	Active Message create mutex fail.
ENUM_NET_INTERFACE_FAIL	5000	Enum network interface fail.
ADD_INFO_TABLE_FAIL	5001	Add information from route table or arp table fail.
UNKNOWN_NET_INTERFACE_FAIL	5002	Select network interface fail.
TABLE_NET_INTERFACE_FAIL	5003	Remove information from route table or arp table fail.

Product Model and ID Reference

The following ioLogik 4000/4200 I/O Network Adapters, I/O Modules and ioLogik 2000 are supported by this MXIO DLL version. Users must upgrade to a new version to support new I/O Modules.

ioLogik 4000/4200

Module ID	Model Name	Network Adapter
0x4010	NA-4010	Ethernet Network Adapter Modbus/TCP
0x4020	NA-4020	RS-485 Network Adapter Modbus/RTU
0x4021	NA-4021	RS-232 Network Adapter Modbus/RTU
0x4200	E4200	Ethernet Network Adapter Modbus/TCP
?????	?????	Digital Input
0x1400	M-1400	4DI, sink, 24VDC, RTB
0X1401	M-1401	4DI, source, 24VDC, RTB
0x1410	M-1410	4DI, sink, 48VDC, RTB
0x1411	M-1411	4DI, source, 48VDC, RTB
0x1800	M-1800	8DI, sink, 24VDC, RTB
0x1801	M-1801	8DI, source, 24VDC, RTB
0x1600	M-1600	16DI, sink, 24VDC, RTB
0x1601	M-1601	16DI, source, 24VDC, 20pin
0x1450	M-1450	4DI, 110VAC, RTB
0x1451	M-1451	4DI, 220VAC, RTB
?????	?????	Digital Output
0x2400	M-2400	4DO, sink, MOSFET, 24VDC, 0.5A, RTB
0x2401	M-2401	4DO, source, MOSFET, 24VDC, 0.5A, RTB
0x2800	M-2800	8DO, sink, MOSFET, 24VDC, 0.5A, RTB
0x2801	M-2801	8DO, source, MOSFET, 24VDC, 0.5A, RTB
0x2600	M-2600	16DO, sink, MOSFET, 24VDC, 0.3A, 20pin

0x2601	M-2601	16DO, source, MOSFET, 24VDC, 0.3A, 20pin
0x2402	M-2402	4DO, sink, MOSFET, diag., 24VDC, 0.5A, RTB
0x2403	M-2403	4DO, source, MOSFET, diag., 24VDC, 0.5A, RTB
0x2404	M-2404	4DO, sink, MOSFET, diag., 24VDC, 2.0A, RTB
0x2405	M-2405	4DO, source, MOSFET, diag., 24VDC, 2.0A, RTB
0x2250	M-2250	2DO, Relay, 230VAC, 24VDC, 2.0A, RTB
0x2254	M-2254	2DO, Triac, 12-125AC, 0.5A, RTB
?????	?????	Analog Input
0x3400	M-3400	4AI, Current, 0-20mA, 12bit, RTB
0x3401	M-3401	4AI, Current, 0-20mA, 14bit, RTB
0x3402	M-3402	4AI, Current, 4-20mA, 12bit, RTB
0x3403	M-3403	4AI, Current, 4-20mA, 14bit, RTB
0x3410	M-3410	4AI, Voltage, 0-10V, 12bit, RTB
0x3411	M-3411	4AI, Voltage, 0-10V, 14bit, RTB
0x3412	M-3412	4AI, Voltage, -10-10V, 12bit, RTB
0x3413	M-3413	4AI, Voltage, -10-10V, 14bit, RTB
0x3414	M-3414	4AI, Voltage, 0-5V, single-ended, 12bit, RTB
0x3415	M-3415	4AI, Voltage, 0-5V, single-ended, 14bit, RTB
0x6200	M-6200	2AI, RTD:PT100,JPT100 300Ohm, RTB
0x6201	M-6201	2AI, Thermocouple:30mV(1uV/bit), RTB
?????	?????	Analog Output
0x4201	M-4201	2AO, 0-20mA, 12bit, RTB
0x4202	M-4202	2AO, 4-20mA, 12bit, RTB
0x4210	M-4210	2AO, Voltage, 0-10V, 12bit, RTB
0x4211	M-4211	2AO, Voltage, -10-10V, 12bit, RTB

0x4212	M-4212	2AO, Voltage, 0-5V, 12bit, RTB
0x2450	M-2450	4DO, Relay, 230VAC, 24VDC, 2.0A, RTB
0x3802	M-3802	8AI, Current, 4-20mA, 12bit, RTB
0x3810	M-3810	8AI, Voltage, 0-10V, 12bit, RTB
0x4402	M-4402	4AO, 4-20mA, 12bit, RTB
0x4410	M-4410	4AO, Voltage, 0-10V, 12bit, RTB

ioLogik 2000

Module ID	Model Name	Remote I/O Server
0x2210	E-2210	12DI, 8DO Active Remote I/O Server
0x2211	E-2210V2	12DI, 8DO Active Remote I/O Server
0x2240	E-2240	8AI, 2AO Active Remote I/O Server
0x2241	E-2240V2	8AI, 2AO Active Remote I/O Server
0x2110	R-2110	12DI, 8DO RS-485 Remote I/O Server
0x2140	R-2140	8AI, 2AO RS-485 Remote I/O Server
0x2212	E-2212	8DI, 8DO, 4DI/DO Active Remote I/O Server
0x2260	E-2260	6RTD, 4DO Active Remote I/O Server
0x2214	E-2214	6DI, 6Relay Outputs Active Remote I/O Server
0x2242	E-2242	4AI, 12DI/DO Active Remote I/O Server
0x0154	E2242-T	4AI, 12DI/DO Active Remote I/O Server
0x2262	E-2262	8TC Input, 4DO Active Remote I/O Server

ioLogik 5000

Module ID	Model Name	Remote I/O Server
0x5340	W-5340	8DIO, 2RLY, 4AI Active Remote I/O Server
0x5312	W-5312	8DI, 8DO, 4 DIO

0x100	W5340T	8DIO, 2RLY, 4AI Active Remote I/O Server
0x101	W5312T	8DI, 8DO, 4 DIO
0x102	W5340Expansion	8DIO, 2RLY, 4AI Active Remote I/O Server
0x103	W5312Expansion	8DI, 8DO, 4 DIO
0x106	W5340-HSDPA	8DIO, 2RLY, 4AI Active Remote I/O Server
0x107	W5312-HSDPA	8DI, 8DO, 4 DIO
0x109	W5340-HSDPA-T	8DIO, 2RLY, 4AI Active Remote I/O Server
0x10A	W5312-HSDPA-T	8DI, 8DO, 4 DIO

ioLogik 1200

Module ID	Model Name	Remote I/O Server
0xE210	E-1210	16DI Active Remote I/O Server
0x50	E-1210-T	16DI Active Remote I/O Server
0xE211	E-1211	16DO Active Remote I/O Server
0x51	E-1211-T	16DO Active Remote I/O Server
0xE212	E-1212	8DI, 8DI/DO Active Remote I/O Server
0x52	E-1212-T	8DI, 8DI/DO Active Remote I/O Server
0xE214	E-1214	6DI, 6Relay Outputs Active Remote I/O Server
0x53	E-1214-T	6DI, 6Relay Outputs Active Remote I/O Server
0xE240	E-1240	8AI Active Remote I/O Server
0x54	E-1240-T	8AI Active Remote I/O

		Server
0xE241	E-1241	4AO Active Remote I/O Server
0x55	E-1241-T	4AO Active Remote I/O Server
0xE242	E-1242	4AI, 4DI, 4DI/DO Active Remote I/O Server
0x56	E-1242-T	4AI, 4DI, 4DI/DO Active Remote I/O Server
0xE260	E-1260	6RTD Input, Active Remote I/O Server
0x57	E-1260-T	6RTD Input, Active Remote I/O Server
0x0059	E-1261-WP-T	12DIO, 4AI, 3RTD Input Active Remote I/O Server
0xE262	E-1262	8TC Input, Active Remote I/O Server
0x58	E-1262-T	8TC Input, Active Remote I/O Server
0x340	E1261H	12DIO 5AI, 3RTD Active Remote I/O Server
0x341	E1261H-T	12DIO 5AI, 3RTD Active Remote I/O Server
0x342	E1263H	24DIO 10AI, 3RTD Active Remote I/O Server
0x343	E1263H-T	24DIO 10AI, 3RTD Active Remote I/O Server
0x5B	E-1213	8DI, 4DO, 4DIO Active Remote I/O Server
0x5C	E-1213-T	8DI, 4DO, 4DIO Active Remote I/O Server

ioLogik 1500

Module ID	Model Name	Remote I/O Server
-----------	------------	-------------------

0x220	E-1510	12DI Active Remote I/O Server
0x221	E-1510-T	12DI Active Remote I/O Server
0x222	E-1512	4DI, 4DIO Active Remote I/O Server
0x223	E-1512-T	4DI, 4DIO Active Remote I/O Server

ioLogik R1000 Series

Module ID	Model Name	Remote I/O Server
0x230	R1210	16DI Remote I/O Server
0x232	R1212	8DI, 8DIO Remote I/O Server
0x234	R1214	6DI, 6RLY Remote I/O Server
0x235	R1240	8AI Remote I/O Server
0x236	R1241	4AO Remote I/O Server

wHiWidth And wLoWidth Renaming Table

Model	wHiWidth	wLoWidth
E2210	OFF	ON
E2212	OFF	ON
E2214	ON	OFF
E2260	OFF	ON
E2262	OFF	ON
E2242	OFF	ON
W5340	OFF	ON
W5312	OFF	ON
E1211	OFF	ON
E1212	OFF	ON
E1214	ON	OFF
E1242	OFF	ON
E1261-WP-T	OFF	ON
E1512	OFF	ON
E1261H	OFF	ON
E1263H	OFF	ON

System Command Sets

System Command Sets provide functions for initializing the connection between host computers and the I/O Server. There are three groups of System Command Sets.

RS-485/RS-232 I/O Connect Commands

[MXSIO_OpenComport](#)
[MXSIO_CloseComport](#)
[MXSIO_Connect](#)
[MXSIO_Disconnect](#)

Ethernet I/O Connect Commands

[MXEIO_Init](#)
[MXEIO_Exit](#)
[MXEIO_Connect](#)
[MXEIO_Disconnect](#)
[MXEIO_CheckConnection](#)
[MXEIO_W5K_Connect](#)
[MXEIO_E1K_Connect](#)
[MXIO_Init_ActiveTag](#)
[MXIO_Init_ActiveTag_Ex](#)
[MXIO_ListDevs_ActiveTag](#)
[MXIO_GetDevsInfo_ActiveTag](#)
[MXIO_Close_ActiveTag](#)
[MXIO_RelLock_ActiveTag](#)

General Commands

[MXIO_GetDllVersion](#)
[MXIO_GetDllBuildDate](#)
[MXIO_GetModuleType](#)
[MXIO_ReadFirmwareRevision](#)
[MXIO_ReadFirmwareDate](#)
[MXIO_Restart](#)
[MXIO_Reset](#)

[MXIO_GetSubType](#)
[MXIO_AutoSearch](#)
[MXIO_ListIF](#)
[MXIO_GetIFInfo](#)
[MXIO_SelectIF](#)
[MXIO_ChangeDupIP](#)

Special Commands for ioLogik 2000

[Module2K_GetSafeStatus](#)
[Module2K_ClearSafeStatus](#)
[Module2K_GetInternalReg](#)
[Module2K_SetInternalReg](#)
[Module2K_GetInternalRegs](#)
[Module2K_SetInternalRegs](#)

Special Commands for ioLogik 4000

[Adp4K_ReadFirmwareRevision](#)
[Adp4K_ReadFirmwareDate](#)
[Adp4K_ReadSlotAmount](#)
[Adp4K_ReadStatus](#)
[Adp4K_ClearStatus](#)
[Adp4K_ReadAlarmedSlot](#)

Special Commands for ioLogik 4200

[E42_ReadFirmwareRevision](#)
[E42_ReadFirmwareDate](#)
[E42_ReadSlotAmount](#)
[E42_ReadStatus](#)
[E42_ClearStatus](#)
[E42_GetInternalRegs](#)
[E42_SetInternalRegs](#)
[E42_GetWorkInternalRegs](#)
[E42_SetWorkInternalRegs](#)
[E42_GetIOMapMode](#)
[E42_SetIOMapMode](#)

[E42_Modbus_List](#)

[E42_ClearSafeStatus](#)

Special Commands for ioLogik 5000

[W5K_GetInternalRegs](#)

[W5K_SetInternalRegs](#)

[W5K_GetGprsSignal](#)

[W5K_ListOpcDevices](#)

[W5K_GetOpcDevicesInfo](#)

[W5K_GetOpcHostName](#)

[W5K_GetSafeStatus](#)

[W5K_ClearSafeStatus](#)

[W5K_GetWorkInternalRegs](#)

[W5K_SetWorkInternalRegs](#)

Special Commands for ioLogik 1200

[E1K_GetSafeStatus](#)

[E1K_ClearSafeStatus](#)

Special Commands for ioLogik R1000

[R1K_GetSafeStatus](#)

[R1K_ClearSafeStatus](#)

MXSIO_OpenCommport

This function opens the local COM port of the host computer with communication parameters.

C/C++

```
int MXSIO_OpenCommport(char * szCommport,
                       DWORD dwBaudrate,
                       BYTE bytDataFormat,
                       DWORD dwTimeout,
                       Int * hCommport);
```

Visual Basic

```
Declare Function MXSIO_OpenCommport Lib "MXIO.dll" (ByVal
sCommport As String, ByVal nBaudrate As Long, ByVal
bytDataFormat As Byte, ByVal nTimeout As Long, hCommport
As Long) As Long
```

Arguments:

szCommport	Name of the common port. I.E., COM3.
dwBaudrate	Baud rate value. E.g. 1200, 9600, 19200.
bytDataFormat	Transmission data format.
dwTimeout	Time out value for serial adapter communication. The unit is in milliseconds.
hCommport	Handle of the opened COM port.
bytDataFormat:	
bit_cnt (bit 0, 1)	0x00 = bit_5 0x01 = bit_6 0x02 = bit_7 0x03 = bit_8
stop_bit (bit 2)	0x00 = stop_1 0x04 = stop_2
parity (bit 3,4 5)	0x00 = none 0x08 = odd 0x18 = even 0x28 = mark

0x38 = space

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXSIO_CloseCommport

Closes the COM port; the COM port handle will be invalid.

C/C++

```
int MXSIO_CloseCommport( int hCommport) ;
```

Visual Basic

```
Declare Function MXSIO_CloseCommport Lib "MXIO.dll" (ByVal  
hCommport As Long) As Long
```

Arguments:

hCommport	Handle of the opened COM port.
-----------	--------------------------------

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXSIO_Connect

Based on the COM port handle or ioLogik 2000 Ethernet module handle, users must use the function to establish an I/O device handle for each RS-485 or RS-232 I/O device. A COM port handle can connect one RS-232 I/O device or up to 64 RS-485 I/O devices.

C/C++

```
int MXSIO_Connect( int hCommport,
                    BYTE bytUnitID,
                    BYTE bytTransmissionMode,
                    int * hConnection);
```

Visual Basic

```
Declare Function MXSIO_Connect Lib "MXIO.dll" (ByVal
hCommport As Long, ByVal bytUnitID As Byte, ByVal
bytTransmissionMode As Byte, hConnection As Long) As Long
```

Arguments:

hCommpo rt	Connectting I/O Server via Serial interface will get a serial handle. Connectting I/O Server via Ethernet interface will get a Ethernet handle For more detail description please see the progam flow IV and program flow V.
bytUnitID	Modbus Unit ID of the RS-232 or RS-485 I/O Server. Ranging from 01 曝? 99.
bytTransmissionMode	Modbus transmission format. 0 RTU Transmission Mode 1 ASCII Transmission Mode Note that the protocol settings must agree with the hardware settings on ioLogik 4000 RS-485/232 I/O Server, and ioLogik 2000 only supports RTU
hConnection	Handle for the I/O device connection.

Return Values :

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXSIO_Disconnect

Disconnect the RS-485/232 I/O device. The I/O device handle will be invalid.

C/C++

```
int MXSIO_Disconnect( int hConnection);
```

Visual Basic

```
Declare Function MXSIO_Disconnect Lib "MXIO.dll" (ByVal  
hConnection As Long) As Long
```

Arguments:

hConnection	Handle for the I/O device connection.
-------------	---------------------------------------

Return Value:

Succeed	MXIO_OK
---------	---------

Fail	Refer to Return Codes.
------	------------------------

MXEIO_Init

Initiate the socket.

C/C++

```
int MXEIO_Init();
```

Visual Basic

```
Declare Function MXEIO_Init Lib "MXIO.dll" () As Long
```

Arguments:

None

Return Values:

Succeed	MXIO_OK
---------	---------

Fail	Refer to Return Codes.
------	------------------------

MXEIO_Exit

To terminates use of the socket.

C/C++

```
void MXEIO_Exit ();
```

Visual Basic

```
Declare Sub MXEIO_Exit Lib "MXIO.dll" ()
```

Arguments:

None

Return Values:

None

MXEIO_Connect

This function establishes a connection to the port of the Ethernet I/O device. Once a connection is established, a handle will be returned for additional functions.

C/C++

```
int MXEIO_Connect( char * sZIP,
                    WORD wPort,
                    DWORD dwTimeOut,
                    int * hConnection);
```

Visual Basic

```
Declare Function MXEIO_Connect Lib "MXIO.dll" (ByVal sZIP
As String, ByVal iPort As Integer, ByVal nTimeOut As
Long, handle As Long) As Long
```

Arguments:

sZIP	IP address of the Ethernet I/O device to be connected.
wPort	TCP port number of Ethernet I/O device. Please use 502 for ioLogik 4000 and ioLogik 2000.
dwTimeOut	Timeout value for establishing a network connection with the ioLogik Ethernet Adapter. The unit is in milliseconds.
hConnection	Handle for the I/O device connection.

Return Values :

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXEIO_Disconnect

Close the connection with the Ethernet I/O device; the handle will be invalid.

C/C++

```
int MXEIO_Disconnect( int hConnection);
```

Visual Basic

```
Declare Function MXEIO_Disconnect Lib "MXIO.dll" (ByVal  
hConnection As Long) As Long
```

Arguments:

hConnection	handle for a connection.
-------------	--------------------------

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXEIO_CheckConnection

This function establishes a connection to the port of the Ethernet I/O device. Once the connection is established, a handle will be returned for additional functions.

C/C++

```
int MXEIO_CheckConnection( int hConnection,
                           DWORD dwTimeOut,
                           BYTE * bytStatus);
```

Visual Basic

```
Declare Function MXEIO_CheckConnection Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal nTimeOut As Long, bytStatus As
Byte) As Long
```

Arguments:

hConnection	I/O device handle for a connection.
dwTimeOut	Timeout value for network connection. The unit is in milliseconds.
bytStatus	Connection status 0: Check connection ok 1: Check connection fail 2: Check connection time out

Return Values:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXEIO_W5K_Connect

This function establishes a connection to the port of the Ethernet I/O device. Once a connection is established, a handle will be returned for additional functions.

C/C++

```
int MXEIO_W5K_Connect( char * szIP,
                        WORD wPort,
                        DWORD dwTimeOut,
                        int * hConnection,
                        char * szMACAddr);
```

Visual Basic

```
Declare Function MXEIO_W5K_Connect Lib "MXIO.dll" (ByVal
szIP As String, ByVal iPort As Integer, ByVal nTimeOut As
Long, handle As Long, ByVal szMACAddr As String) As Long
```

Arguments:

szIP	IP address of the Ethernet I/O device to be connected. A-OPC Server IP address or ioLogik 5000 IP address
wPort	TCP port number of Ethernet I/O device. Please use 502 for ioLogik 5000
dwTimeOut	Timeout value for establishing a network connection with the ioLogik Ethernet Adapter. The unit is in milliseconds.
hConnection	Handle for the I/O device connection.
szMACAddr	<ol style="list-style-type: none">if szIP set to AOPC server IP address . The format like this 00-12-dd-23-78-a2if szIP set to W5000 IP address. Set to NULL

Return Values :

Succeed	MXIO_OK
---------	---------

Fail

Refer to Return Codes.

MXEIO_E1K_Connect

This function establishes a connection to the port of the Ethernet I/O device. Once a connection is established, a handle will be returned for additional functions.

C/C++

```
int MXEIO_E1K_Connect( char * szIP,
                        WORD wPort,
                        DWORD dwTimeOut,
                        int * hConnection,
                        char * szPassword);
```

Visual Basic

```
Declare Function MXEIO_E1K_Connect Lib "MXIO.dll" (ByVal
szIP As String, ByVal iPort As Integer, ByVal nTimeOut As
Long, handle As Long, ByVal szPassword As String) As Long
```

Arguments:

szIP	IP address of the Ethernet I/O device to be connected. A-OPC Server IP address or ioLogik 1200 IP address
wPort	TCP port number of Ethernet I/O device. Please use 502 for ioLogik 1200
dwTimeOut	Timeout value for establishing a network connection with the ioLogik Ethernet Adapter. The unit is in milliseconds.
hConnection	Handle for the I/O device connection.
szPassword	Max length 8 bytes

Return Values :

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_Init_ActiveTag

This function code is used to initiate receiving active tag from ioLogik device.

C/C++

```
int MXIO_Init_ActiveTag( WORD wDataPort, WORD wCmdPort,
DWORD dwToleranceTimeout, DWORD dwCmdTimeout,
pfnTagDataCALLBACK iProcAddress, WORD wSize);
```

VISUAL BASIC

```
Declare Function MXIO_Init_ActiveTag Lib "MXIO.dll" (ByVal
wDataPort As Integer, ByVal wCmdPort As Integer, ByVal
dwToleranceTimeout As Long, ByVal dwCmdTimeout As Long,
ByVal nProcAddress As Long, ByVal wSize As Integer) As
Long
```

Arguments:

wDataPort	TCP port number of Ethernet I/O device. Default:9900
wCmdPort	TCP port number of Ethernet I/O device. Default:9500
dwToleranceTimeout	active tag timeout tolerance for wDataPort. The unit is in milliseconds.
dwCmdTimeout	active tag timeout tolerance for wCmdPort. The unit is in milliseconds.
iProcAddress	Callback function, which is called after an active tag is received form ioLogik device.
wSize	Queue size of each device for active tag.

Return Values :

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_Init_ActiveTag_Ex

This function code is used to start receive active tag of ioLogik Ethernet Module.

C/C++

```
int MXIO_Init_ActiveTag_Ex( WORD wDataPort, WORD  
wCmdPort, DWORD dwToleranceTimeout, DWORD dwCmdTimeout,  
pfnTagDataCALLBACK iProcAddress, WORD wSize);
```

VISUAL BASIC

```
Declare Function MXIO_Init_ActiveTag_Ex Lib MXIO.dll  
(ByVal wDataPort As Integer, ByVal wCmdPort As Integer,  
ByVal dwToleranceTimeout As Long, ByVal dwCmdTimeout As  
Long, ByVal nProcAddress As Long, ByVal wSize As Integer)  
As Long
```

Arguments:

wDataPort	TCP port number of Ethernet I/O device. Default:9900
wCmdPort	TCP port number of Ethernet I/O device. Default:9500
dwToleranceTimeout	active tag timeout tolerance for wDataPort. The unit is in milliseconds.
dwCmdTimeout	active tag timeout tolerance for wCmdPort. The unit is in milliseconds.
iProcAddress	Callback function, which is called after receive an active Tag form ioLogik Ethernet module.
wSize	Queue size of each device for active tag.

Return Values :

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_ListDevs_ActiveTag

This function is used to get total amount of ioLogik devices that link to AOPC server.

C/C++

```
int MXIO_ListDevs_ActiveTag( WORD * wDeviceCount);
```

VISUAL BASIC

```
Declare Function MXIO_ListDevs_ActiveTag Lib MXIO.dll  
(wDeviceCount As Integer) As Long
```

Arguments:

wDeviceCount	Total amount of ioLogik devices connected to AOPC Server.
--------------	---

Return Values:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_GetDevsInfo_ActiveTag

This function is used to get IP and MAC address of ioLogik devices that link to AOPC server.

C/C++

```
int MXIO_GetDevsInfo_ActiveTag( WORD wDeviceCount, char szDeviceInfo []);
```

VISUAL BASIC

```
Declare Function MXIO_GetDevsInfo_ActiveTag Lib MXIO.dll  
(ByVal wDeviceCount As Integer, szDeviceInfo As Byte) As  
Long
```

Arguments:

wDeviceCount	Total amount of ioLogik devices connected to AOPC Server. (get from MXIO_ListDevs_ActiveTag API)
--------------	--

szDeviceInfo	Each ioLogik device status request 10 Bytes
--------------	---

Device Address:

IP Address	4 bytes, start from array [0]
------------	-------------------------------

MAC Address	6 Bytes, start from array [4]
-------------	-------------------------------

Return Values:

Succeed	MXIO_OK
---------	---------

Fail	Refer to Return Codes.
------	------------------------

MXIO_Close_ActiveTag

This function is used to close the connection between AOPC and ioLogik device. The handle will be invalid.

Value will return in 30 seconds. If return busy, call again.

C/C++

```
MXIO_Close_ActiveTag(void);
```

VISUAL BASIC

```
Declare Function MXIO_Close_ActiveTag Lib MXIO.dll () As  
Long
```

Arguments:

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_Connect_ActiveTag

This function is used to establish a connection between ActiveTag-link and ioLogik device. A handle will be created for other functions.

C/C++

```
int MXIO_Connect_ActiveTag(Uint32 dwTimeOut, Int32[]  
hConnection, byte[] szMACAddr, Uint16 wPort,  
byte[] szPassword);
```

VISUAL BASIC

```
Declare Function MXIO_Connect_ActiveTag Lib "MXIO.dll"  
(ByVal dwTimeOut As Long, hConnection As Long, szMACAddr  
As Byte, ByVal wPort As Integer, Password As Byte) As  
Long
```

Arguments:

dwTimeOut	Timeout value for establishing a connection between AOPC and ioLogik device. The unit is in milliseconds.
hConnection	Handle for the I/O device connection.
szMACAddr	MAC address of the connected ioLogik device
wPort	TCP port number of the connected ioLogik device. Reserved. Don't care.
szPassword	Password for the connected ioLogik device. Max length 8 bytes Reserved. Don't care.

Return Values :

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_RelLock_ActiveTag

This function is used to close the connection between AOPC and ioLogik device. The handle will be invalid.

C/C++

```
int MXIO_RelLock_ActiveTag();
```

VISUAL BASIC

```
Declare Function MXIO_RelLock_ActiveTag Lib "MXIO.dll" () As  
Long
```

Arguments:

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_GetDllVersion

Get the DLL version.

C/C++

```
int MXIO_GetDllVersion ();
```

Visual Basic

```
Declare Function MXIO_GetDllVersion Lib "MXIO.dll"  
() As Long
```

Arguments:

None

Return Value:

Succeed	Return the DLL version. If the value is 0x2000, then the version is 2.0.0.0
---------	--

MXIO_GetDllBuildDate

Get the DLL release date.

C/C++

```
int MXIO_GetDllBuildDate ();
```

Visual Basic

```
Declare Function MXIO_GetDllBuildDate Lib  
"MXIO.dll" () As Long
```

Arguments:

None

Return Value:

Succeed	Return the DLL release date. If the value is 0x20071001, then the date is 2007/10/01
---------	--

MXIO_GetModuleType

This function reports the model name of the Network Adapter and the I/O modules.

C/C++

```
int MXIO_GetModuleType(int hConnection,
                      BYTE bytSlot,
                      WORD * wType);
```

Visual Basic

```
Declare Function MXIO_GetModuleType Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, iType As
Integer) As Long
```

Arguments:

hConnection	I/O device handle for a connection.
bytSlot	Slot number of the I/O device to be checked. The ioLogik 4000 Network Adapter's Slot number is always 0. The Slot number ranges from 0 to 32. . But this parameter is inactive in ioLogik 2000.
wType	A pointer that stores the model name. If the value is 0x4010, the model name is NA-4010. Refer to the Model name reference table for more information.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_ReadFirmwareRevision

This function reports the firmware revision of the ioLogik Network Adapter or the ioLogik 2000 and ioLogik 5000 module.

C/C++

```
int MXIO_ReadFirmwareRevisioin( int hConnection,  
                                BYTE bytRevision[ ]);
```

Visual Basic

```
Declare Function MXIO_ReadFirmwareRevision Lib  
"MXIO.dll" (ByVal hConnection As Long, bytRevision As  
Byte) As Long
```

Arguments:

hConnection	I/O device handle for a connection.
bytRevision	stored ioLogik 4000 Firmware revision bytRevision[0] = major (A) bytRevision[1] = minor (B) bytRevision[2] = release (C) bytRevision[3] = build (D) format is A.B.C.D

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_ReadFirmwareDate

This function reports firmware release date of the ioLogik 4000 Network Adapter or the ioLogik 2000 and ioLogik 5000 module.

C/C++

```
int MXIO_ReadFirmwareDate( int hConnection,
                           WORD wDate[ ] );
```

Visual Basic

```
Declare Function MXIO_GetModuleType Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, iType As
Integer) As Long
```

Arguments:

hConnection I/O device handle for a connection.

wDate Firmware release date. Ex If Word 0 = 0x0705 , Word 1 = 0x2005 then firmware release date is July 5, 2005

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

MXIO_Restart

This function is used to restart the I/O device.

C/C++

```
int MXIO_Restart( int hConnection);
```

Visual Basic

```
Declare Function MXIO_Restart Lib "MXIO.dll" (ByVal  
hConnection As Long) As Long
```

Arguments:

hConnection I/O device handle for a connection.

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

MXIO_Reset

This function is used to reset the I/O device.

C/C++

```
int MXIO_Reset( int hConnection);
```

Visual Basic

```
Declare Function MXIO_Reset Lib "MXIO.dll" (ByVal  
hConnection As Long) As Long
```

Arguments:

hConnection	I/O Server handle for a connection.
-------------	-------------------------------------

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_GetSubType

This function is used to report the model subtype of the ioLogik device.

C/C++

```
int MXIO_GetSubType( int hConnection, BYTE bytSlot, DWORD *  
dwSubType) ;
```

VISUAL BASIC

```
Declare Function MXIO_GetSubType Lib "MXIO.dll" (ByVal  
hConnection As Long, ByVal bytSlot As Byte, dwSubType  
As Long) As Long
```

Arguments:

hConnection	A handle of a connection.
bytSlot	this parameter is always 0.
dwSubType	A pointer that stores the model sub type. dwSubType & 0x0000000F = 1 (GPRS) dwSubType & 0x0000000F = 2 (HSDPA) dwSubType & 0x000F0000 = 1 (Wide Temperature)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_AutoSearch

This function is used to get device information of ioLogik device that in the intranet.

C/C++

```
int MXIO_AutoSearch(int nType, int nRetryCount, int  
nTimeout, int* nNumber, BYTE struML[]);
```

VISUAL BASIC

```
Declare Function MXIO_AutoSearch Lib "MXIO.dll" (ByVal nType  
As Long, ByVal nRetryCount As Long, ByVal nTimeout As  
Long, nNumber As Long, struML As Long) As Long
```

Arguments:

nType	Type of the Ethernet I/O device.
nRetryCount	Retry broadcast quantity
nTimeout	Timeout of wait device response
nNumber	Response device quantity
struML	Device information

Type:

This parameter can be a combination of flags from the following groups of flags.

Type	Value	Note
E4000_SERIES	1	
E2000_SERIES	2	
E4200_SERIES	4	
E1200_SERIES	8	
W5000_SERIES	16	
E1500_SERIES	64	
ioPAC8000_SERIES	128	
AOPC_SERVER	256	

struML:

nModuleID	2 bytes, Module ID of device
-----------	------------------------------

szModuleIP	16 bytes, IP address of device
szMAC	6 bytes, MAC address of device
szModuleIP1	16 bytes, 2 nd IP address of device
szMAC1	6 bytes, 2 nd MAC address of device
bytLanUse	1 bytes, 0 -> Lan0, 1 -> Lan1

Return Values:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_ListIF

This function is used to get total amount network interface that exist in host pc.

C/C++

```
int MXIO_ListIF(WORD * wIFCount);
```

VISUAL BASIC

```
Declare Function MXIO_ListIF Lib MXIO.dll (wIFCount As Integer) As Long
```

Arguments:

wIFCount	Total amount of network interface exist in host pc.
----------	---

Return Values:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_GetIFInfo

This function is used to get network interface information of host pc.

C/C++

```
int MXIO_GetIFInfo(WORD wIFCount, char IFInfo []);
```

VISUAL BASIC

```
Declare Function MXIO_GetIFInfo Lib "MXIO.dll" (ByVal  
wIFCount As Integer, IFInfo As Byte) As Long
```

Arguments:

wIFCount	Total amount of ioLogik devices connected to AOPC Server. (get from MXIO_ListIF API)
IFInfo	Each network interface information request 282 Bytes

Network Interface information Address:

Index	4 bytes, start from array [0]
IP address	16 Bytes, start from array [4]
MAC address	6 Bytes, start from array [20]
Name or Descriptor	256 Bytes, start from array [26]

Return Values:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_SelectIF

This function is used to select network interface that we want.

C/C++

```
int MXIO_SelectIF (WORD wIFCount, char IFInfo [], DWORD dwIFIndex);
```

VISUAL BASIC

```
Declare Function MXIO_SelectIF Lib "MXIO.dll" (ByVal wIFCount As Integer, IFInfo As Byte, ByVal wIFIndex As Long) As Long
```

Arguments:

wIFCount	Total amount of ioLogik devices connected to AOPC Server. (get from MXIO_ListIF API)
IFInfo	Each network interface information request 282 Bytes
dwIFIndex	Network interface index value

Network Interface information Address:

Index	4 bytes, start from array [0]
IP address	16 Bytes, start from array [4]
MAC address	6 Bytes, start from array [20]
Name or Descriptor	256 Bytes, start from array [26]

Return Values:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_ChangeDupIP

This function establishes a connection to the port of the Ethernet I/O device and change IP address and then restart device.

C/C++

```
int MXIO_ChangeDupIP( char * szIP,
                      WORD wPort,
                      char * szMACAddr,
                      DWORD dwTimeOut,
                      char * szPassword,
                      char * szNewIP, );
```

Visual Basic

```
Declare Function MXIO_ChangeDupIP Lib "MXIO.dll" (ByVal
szIP As String, ByVal iPort As Integer, szMACAddr As
Byte, ByVal nTimeOut As Long, ByVal szPassword As String,
ByVal szNewIP As String) As Long
```

Arguments:

szIP	IP address of the Ethernet I/O device to be connected.
wPort	TCP port number of Ethernet I/O device. Please use 502 for ioLogik 1200
szMACAddr	MAC address of the Ethernet I/O device.
dwTimeOut	Timeout value for establishing a network connection with the ioLogik Ethernet Adapter. The unit is in milliseconds.
szPassword	Max length 8 bytes
szNewIP	New IP address of the Ethernet I/O device to be changed.

Return Values:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Module2K_GetSafeStatus

This function code is used to get the safe status of ioLogik 2000 Module.

C/C++

```
int Module2K_GetSafeStatus( int hConnection,
                           WORD wStatus);
```

Visual Basic

```
Declare Function Module2K_GetSafeStatus Lib "MXIO.dll"
(ByVal hConnection As Long, iStatus As Integer) As
Long
```

Arguments:

hConnection	The handle for a connection.
wStatus	An pointer that stores the specific module's safe status. The values are: 0: Normal 1: Safe mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Module2K_ClearSafeStatus

This function code is used to clear the safe status of ioLogik 2000 Module.

C/C++

```
int Module2K_ClearSafeStatus( int hConnection) ;
```

Visual Basic

```
Declare Function Module2K_ClearSafeStatus Lib  
"MXIO.dll" (ByVal hConnection As Long) As Long
```

Arguments:

hConnection	The handle for a connection.
-------------	------------------------------

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Module2K_GetInternalReg

This function code is used to get the internal register of ioLogik 2000 Module.

C/C++

```
int Module2K_GetInternalReg(int hConnection,  
                           BYTE bytChannel,  
                           WORD * wValue);
```

Visual Basic

```
Declare Function Module2K_GetInternalReg Lib  
"MXIO.dll" (ByVal hConnection As Long, ByVal  
bytChannel As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for a connection.
bytChannel	The specific channel to be read.
wValue	represents the value of the starting channel. The values are 0 ~ 255

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Module2K_SetInternalReg

This function code is used to set the internal register of ioLogik 2000 Module.

C/C++

```
int Module2K_SetInternalReg ( int hConnection,
                               BYTE bytChannel,
                               WORD wValue);
```

Visual Basic

```
Declare Function Module2K_SetInternalReg Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytChannel As Byte, ByVal iValue As Integer) As Long
```

Arguments:

hConnection	The handle for a connection.
bytChannel	The specific channel to be read.
wValue	represents the value of the starting channel. The values are 0 ~ 255

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Module2K_GetInternalRegs

This function code is used to get the internal registers of ioLogik 2000 Module.

C/C++

```
int Module2K_GetInternalRegs (int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               WORD wValue[ ]);
```

Visual Basic

```
Declare Function Module2K_GetInternalRegs Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for a connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets. (Up to 24)
wValue	represents the value of the starting channel. The values are 0 ~ 255

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Module2K_SetInternalRegs

This function code is used to set the internal registers of ioLogik 2000 Module.

C/C++

```
int Module2K_SetInternalRegs ( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                WORD wValue[ ] );
```

Visual Basic

```
Declare Function Module2K_SetInternalReg Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for a connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets. (Up to 24)
wValue	An array that stores contiguous internal register The values are 0 ~ 255

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Adp4K_ReadFirmwareRevision

This function code is used to read the firmware revision.

C/C++

```
int Adp4K_ReadFirmwareRevision( int hConnection,  
                                WORD *wRevision);
```

Visual Basic

```
Declare Function Adp4K_ReadFirmwareRevision Lib  
"MXIO.dll" (ByVal hConnection As Long, iRevision As  
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
WRevision	Stores the firmware revision, if value is 0X0101 then revision is 1.01

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Adp4K_ReadFirmware

This function code is used to read the firmware release date.

C/C++

```
int Adp4K_ReadFirmwareDate(int hConnection,  
                           WORD wDate[ ]);
```

Visual Basic

```
Declare Function Adp4K_ReadFirmwareDate Lib "MXIO.dll"  
(ByVal hConnection As Long, iDate As Integer) As Long
```

Arguments:

hConnection	The handle for a connection.
WDate	An array that stores the firmware release date. If wDate[0] is 0X0705 and wDate[1] is 0X2005 then firmware release date is July 5, 2005

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Adp4K_ReadSlotAmount

This function code is used to read number of expansion slots.

C/C++

```
int Adp4K_ReadSlotAmount( int hConnection,  
                           WORD * wAmount);
```

Visual Basic

```
Declare Function Adp4K_ReadSlotAmount Lib "MXIO.dll"  
          (ByVal hConnection As Long, iAmount As Integer) As  
          Long
```

Arguments:

hConnection	The handle for a connection.
WAmount	A pointer that stores the number of expansion slots

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Adp4K_ReadStatus

This function code is used to read the status of the ioLogik4000 adapter.

C/C++

```
int Adp4K_ReadStatus( int hConnection,
                      WORD *wBusStatus,
                      WORD *wFPStatus,
                      WORD *wEWStatus,
                      WORD *wESStatus,
                      WORD *wECStatus);
```

Visual Basic

```
Declare Function Adp4K_ReadStatus Lib "MXIO.dll"
(ByVal hConnection As Long, iBusStatus As Integer,
iFPStatus As Integer, iEWStatus As Integer, iESStatus
As Integer, iECStatus As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
wBusStatus	Stores the Bus status in numerical format. The values are 0: Normal Operation 1: Bus Standby 2: Bus Communication Fault 3: Slot Configuration Failed 4: No Expansion Slot.
wFPStatus	Stores the Field Power status in numerical format. The values are: 0: 24Vdc Field Power On. 1: 24Vdc Field Power Off.
wEWStatus	Stores the Watchdog status in numerical format. The values are: 0: No Error 1: Watchdog activated.
wESSStatus	Stores the Modbus Setup Error status in

numeric data format, only support NA-4020 & NA-4021.

0: No Error

1: Modbus Setup Error

wECStatus

Stores the Modbus Checksum Error status, only support NA-4020 & NA-4021.

0: No Error

1: continuously three CRC/LRC checksum errors since its last restart, clear counters operation, or power-up.

Return Value:

Succeed

MXIO_OK

Fail

Refer to Return Codes.

Adp4K_ClearStatus

This function code is used to clear the status of the ioLogik 4000 adapters.

C/C++

```
int Adp4K_ClearStatus( int hConnection );
```

Visual Basic

```
Declare Function Adp4K_ClearStatus Lib "MXIO.dll"  
    (ByVal hConnection As Long, iBusStatus As Integer,  
     iFPStatus As Integer, iEWStatus As Integer, iESStatus  
     As Integer, iECStatus As Integer) As Long
```

Arguments:

hConnection The handle for an I/O device connection.

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

Adp4K_ReadAlarmedSlot

This function code is used to read the Alarm slot list.

C/C++

```
int Adp4K_ReadAlarmedSlot( int hConnection,
                           DWORD * dwAlarm);
```

Visual Basic

```
Declare Function Adp4K_ReadAlarmedSlot Lib "MXIO.dll"
(ByVal hConnection As Long, nAlarm As Long) As Long
```

Arguments:

hConnection	The handle for a connection.
DwAlarm	A pointer that stores the Alarm slot list. The corresponding bit represents slot position. The wAlarm bit 0 is represented by the first slot and bit 31 is represented by the last slot. The values are: 0: Normal slot 1: Alarm slot

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_ReadFirmwareRevision

This function reports the firmware revision of the ioLogik 4200 Network Adapter.

C/C++

```
int E42_ReadFirmwareRevisioin( int hConnection,
                                BYTE bytRevision[ ] );
```

Visual Basic

```
Declare Function E42_ReadFirmwareRevision Lib MXIO.dll
(ByVal hConnection As Long, bytRevision As Byte) As
Long
```

Arguments:

hConnection I/O device handle for a connection.

bytRevision stored ioLogik 4000 Firmware revision
 bytRevision[0] = major (A)
 bytRevision[1] = minor (B)
 bytRevision[2] = release I
 bytRevision[3] = build (D)
format is A.B.C.D

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

E42_ReadFirmwareDate

This function reports firmware release date of the ioLogik 4200 Network Adapter.

C/C++

```
int E42_ReadFirmwareDate( int hConnection,
                           WORD wDate[ ] );
```

Visual Basic

```
Declare Function E42_GetModuleType Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, iDate As
Integer) As Long
```

Arguments:

hConnection I/O device handle for a connection.

wDate Firmware release date. Ex If Word 0 = 0x0705 , Word 1 = 0x2005 then firmware release date is July 5, 2005

Return Value:

Succeed **MXIO_OK**

Fail Refer to Return Codes.

E42_ReadSlotAmount

This function code is used to read number of expansion slots.

C/C++

```
int E42_ReadSlotAmount( int hConnection,
                        WORD *wAmount);
```

Visual Basic

```
Declare Function E42_ReadSlotAmount Lib MXIO.dll
(ByVal hConnection As Long, iAmount As Integer) As
Long
```

Arguments:

hConnection	The handle for a connection.
Wamount	A pointer that stores the number of expansion slots

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_ReadStatus

This function code is used to read the status of the ioLogik 4200 Network adapter.

C/C++

```
int E42_ReadStatus( int hConnection,
                     WORD *wState,
                     WORD *wLastErrorCode);
```

Visual Basic

```
Declare Function E42_ReadStatus Lib "MXIO.dll" (ByVal
hConnection As Long, wState As Integer, wLastErrorCode
As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
wState	Stores the Bus status in numerical format. The values are 0: Initial now 1: IO Ready 2: Initial Fault 3: IO Failed
wLastErrorCode	Stores the Field Power status in numerical format. The values are: 0: No error -3: No module attached(retry). -4: Set module parameter (need reboot) -5: module worm-up error (need reboot) -30: module configuration error (need reboot)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_ClearStatus

This function code is used to clear the status of the ioLogik 4200 Network adapters. When change slot or remove slot from E4200. (Need to reboot E4200 after clear status)

C/C++

```
int E42_ClearStatus( int hConnection );
```

Visual Basic

```
Declare Function E42_ClearStatus Lib MXIO.dll (ByVal  
hConnection As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
-------------	--

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_GetInternalRegs

This function code is used to get the internal registers of ioLogik 4200 network adapters.

C/C++

```
int E42_GetInternalRegs (int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wValue[ ]);
```

Visual Basic

```
Declare Function E42_GetInternalRegs Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As
Byte, ByVal bytCount As Byte, iValue As Integer) As
Long
```

Arguments:

hConnection	The handle for a connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets. (Up to 80)
wValue	represents the value of the starting channel. The values are 0 ~ 255

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_SetInternalRegs

This function code is used to set the internal registers of ioLogik 4200 network adapters.

C/C++

```
int E42_SetInternalRegs ( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wValue[ ] );
```

Visual Basic

```
Declare Function E42_SetInternalRegs Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As
Byte, ByVal bytCount As Byte, iValue As Integer) As
Long
```

Arguments:

hConnection	The handle for a connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets. (Up to 80)
wValue	An array that stores contiguous internal register The values are 0 ~ 255

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_GetWorkInternalRegs

This function code is used to get the working internal registers of ioLogik 4200 network adapters.

C/C++

```
int E42_GetWorkInternalRegs (int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            WORD wValue[ ]);
```

Visual Basic

```
Declare Function E42_GetWorkInternalRegs Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As
Byte, ByVal bytCount As Byte, iValue As Integer) As
Long
```

Arguments:

hConnection	The handle for a connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets. (Up to 80)
wValue	represents the value of the starting channel. The values are 0 ~ 255

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_SetWorkInternalRegs

This function code is used to set the working internal registers of ioLogik 4200 network adapters (not save to flash memory).

C/C++

```
int E42_SetWorkInternalRegs ( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                WORD wValue[ ] );
```

Visual Basic

```
Declare Function E42_SetWorkInternalRegs Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As
Byte, ByVal bytCount As Byte, iValue As Integer) As
Long
```

Arguments:

hConnection	The handle for a connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets. (Up to 80)
wValue	An array that stores contiguous internal register The values are 0 ~ 255

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_GetIOMapMode

This function code is used to get the IO image map mode of ioLogik 4200 network adapters.

C/C++

```
int E42_GetIOMapMode ( int hConnection,
                      WORD * wValue );
```

Visual Basic

```
Declare Function E42_GetIOMapMode Lib MXIO.dll (ByVal
hConnection As Long, iValue As Integer) As Long
```

Arguments:

hConnection The handle for a connection.

wValue Stores the specific module's IO image map status. The values are:
0: fix mode
1: dynamic mode

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

E42_SetIOMapMode

This function code is used to set the IO image map mode of ioLogik 4200 network adapters.

C/C++

```
int E42_SetIOMapMode ( int hConnection,
                      WORD wValue );
```

Visual Basic

```
Declare Function E42_SetIOMapMode Lib MXIO.dll (ByVal
hConnection As Long, ByVal iValue As Integer) As Long
```

Arguments:

hConnection The handle for a connection.

wValue Stores the specific module's IO
image map status. The values are:
0: fix mode
1: dynamic mode

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

E42_Modbus_List

This function code is used to set the IO image map mode of ioLogik 4200 network adapters.

C/C++

```
int E42_Modbus_List ( int hConnection,
                      char * FilePath);
```

Visual Basic

```
Declare Function E42_Modbus_List Lib MXIO.dll (ByVal
hConnection As Long, ByVal FilePath As String) As Long
```

Arguments:

hConnection	The handle for a connection.
FilePath	Specific log file path and file name to save module ??? 's IO image map list file. Ex. Char * FilePath = c:\\modbus.txt;

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_ClearSafeStatus

This function code is used to clear E4200 watchdog status.
(Device need reboot after clear watchdog status.)

C/C++

```
int E42_ClearSafeStatus ( int hConnection) ;
```

Visual Basic

```
Declare Function E42_ClearSafeStatus Lib MXIO.dll  
          (ByVal hConnection As Long) As Long
```

Arguments:

hConnection	The handle for a connection.
-------------	------------------------------

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_GetInternalRegs

This function code is used to get the internal registers of ioLogik 5000 Module.

C/C++

```
int W5K_GetInternalRegs (int hConnection,
                         BYTE bytStartChannel,
                         BYTE bytCount,
                         WORD wValue[ ]);
```

Visual Basic

```
Declare Function W5K_GetInternalRegs Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As
Byte, ByVal bytCount As Byte, iValue As Integer) As
Long
```

Arguments:

hConnection	The handle for a connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets. (Up to 24)
wValue	represents the value of the starting channel. The values are 0 ~ 255

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_SetInternalRegs

This function code is used to set the internal registers of ioLogik 5000 Module.

C/C++

```
int W5K_SetInternalRegs ( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wValue[ ] );
```

Visual Basic

```
Declare Function W5K_SetInternalRegs Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As
Byte, ByVal bytCount As Byte, iValue As Integer) As
Long
```

Arguments:

hConnection	The handle for a connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets. (Up to 24)
wValue	An array that stores contiguous internal register The values are 0 ~ 255

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_GetGprsSignal

This function code is used to get the Click & Go Logic start status of ioLogik 5000 Ethernet Module.

C/C++

```
int W5K_GetGprsSignal( int hConnection,
                        WORD * wStatus);
```

Visual Basic

```
Declare Function W5K_GetGprsSignal Lib MXIO.dll (ByVal
hConnection As Long, iStatus As Integer) As Long
```

Arguments:

hConnection The handle for a connection.

iStatus An pointer that stores the specific module's Click & Go Logic start status. The values are:

0: stop

1: start

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

W5K_ListOpcDevices

This function get amount of ioLogik W5000 that link in A-OPC server.

C/C++

```
int W5K_ListOpcDevices( char * szIP,
                           DWORD dwTimeOut,
                           WORD* wDeviceCount);
```

Visual Basic

```
Declare Function W5K_GetOpcHostName Lib MXIO.dll (ByVal
szIP As String, ByVal nTimeOut As Long, wDeviceCount As
Integer) As Long
```

Arguments:

szIP	IP address of the A-OPC Server to be connected.
dwTimeOut	Timeout value for establishing a network connection with the ioLogik Ethernet Adapter. The unit is in milliseconds.
wDeviceCount	Total amount for the I/O device connected to A-OPC Server.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_GetOpcDevicesInfo

This function get ioLogik W5000 Device information that link in A-OPC server .

C/C++

```
int W5K_GetOpcDevicesInfo( char * szIP,
                           DWORD dwTimeOut,
                           WORD wDeviceCount,
                           char szDeviceInfo[]);
```

Visual Basic

```
Declare Function W5K_GetOpcDevicesInfo Lib MXIO.dll
(ByVal szIP As String, ByVal nTimeOut As Long, ByVal
wDeviceCount As Integer, szDeviceInfo As Byte) As Long
```

Arguments:

szIP	IP address of the A-OPC Server to be connected.
dwTimeOut	Timeout value for establishing a network connection with the ioLogik Ethernet Adapter. The unit is in milliseconds.
wDeviceCount	Total amount for the I/O device connected to A-OPC Server.(get from W5K_ListOpcDevices API)
szDeviceInfo	Each ioLogik W5000 device status request 12 Bytes

Device status:

IP Address	4 bytes, start from array [0]
MAC Address	6 Bytes, start from array [4]
Online Status	1 Bytes, start from array [10]
UnitID	1 Bytes, start from array [11]

Return Values:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_GetOpcHostName

This function get A-OPC server alias name of the specific ip address device that running A-OPC server.

C/C++

```
int W5K_GetOpcHostName( char * szIP,
                         DWORD dwTimeOut,
                         char szAliasName[] );
```

Visual Basic

```
Declare Function W5K_GetOpcHostName Lib MXIO.dll (ByVal
szIP As String, ByVal nTimeOut As Long, ByVal szAliasName
As String) As Long
```

Arguments:

szIP	IP address of the A-OPC Server to be connected.
dwTimeOut	Timeout value for establishing a network connection with the ioLogik Ethernet Adapter. The unit is in milliseconds.
szAliasName	A-OPC Server alias name (MAX: 32 Bytes)

Return Values:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_GetSafeStatus

This function code is used to get the safe status of ioLogik 5000 Module.

C/C++

```
int W5K_GetSafeStatus( int hConnection,
                      WORD wStatus);
```

Visual Basic

```
Declare Function W5K_GetSafeStatus Lib MXIO.dll (ByVal
hConnection As Long, iStatus As Integer) As Long
```

Arguments:

hConnection	The handle for a connection.
wStatus	An pointer that stores the specific module's safe status. The values are: 0: Normal 1: Safe mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_ClearSafeStatus

This function code is used to clear the safe status of ioLogik 5000 Module.

C/C++

```
int W5K_ClearSafeStatus( int hConnection) ;
```

Visual Basic

```
Declare Function W5K_ClearSafeStatus Lib MXIO.dll  
(ByVal hConnection As Long) As Long
```

Arguments:

hConnection	The handle for a connection.
-------------	------------------------------

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_GetWorkInternalRegs

This function code is used to get the internal registers of ioLogik 5000 Module.

C/C++

```
int W5K_GetWorkInternalRegs (int hConnection,
                             BYTE bytStartChannel,
                             BYTE bytCount,
                             WORD wValue[ ]);
```

Visual Basic

```
Declare Function W5K_GetWorkInternalRegs Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As
Byte, ByVal bytCount As Byte, iValue As Integer) As
Long
```

Arguments:

hConnection	The handle for a connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets. (Up to 24)
wValue	represents the value of the starting channel. The values are 0 ~ 255

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_SetWorkInternalRegs

This function code is used to set the internal registers of ioLogik 5000 Module.

C/C++

```
int W5K_SetWorkInternalRegs ( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               WORD wValue[ ] );
```

Visual Basic

```
Declare Function W5K_SetWorkInternalRegs Lib MXIO.dll
( ByVal hConnection As Long, ByVal bytStartChannel As
Byte, ByVal bytCount As Byte, iValue As Integer) As
Long
```

Arguments:

hConnection	The handle for a connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets. (Up to 24)
wValue	An array that stores contiguous internal register The values are 0 ~ 255

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_GetSafeStatus

This function code is used to get the safe status of ioLogik 1200 Module.

C/C++

```
int E1K_GetSafeStatus( int hConnection,
                      WORD wStatus);
```

Visual Basic

```
Declare Function E1K_GetSafeStatus Lib MXIO.dll (ByVal
hConnection As Long, iStatus As Integer) As Long
```

Arguments:

hConnection	The handle for a connection.
wStatus	An pointer that stores the specific module's safe status. The values are: 0: Normal 1: Safe mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_ClearSafeStatus

This function code is used to clear the safe status of ioLogik 1200 Module.

C/C++

```
int E1K_ClearSafeStatus( int hConnection) ;
```

Visual Basic

```
Declare Function E1K_ClearSafeStatus Lib MXIO.dll  
(ByVal hConnection As Long) As Long
```

Arguments:

hConnection	The handle for a connection.
-------------	------------------------------

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_GetSafeStatus

This function code is used to get the safe status of ioLogik R1000 Module.

C/C++

```
int R1K_GetSafeStatus( int hConnection,
                      WORD wStatus);
```

Visual Basic

```
Declare Function R1K_GetSafeStatus Lib "MXIO.dll"
(ByVal hConnection As Long, iStatus As Integer) As
Long
```

Arguments:

hConnection	The handle for a connection.
wStatus	An pointer that stores the specific module's safe status. The values are: 0: Normal 1: Safe mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_ClearSafeStatus

This function code is used to clear the safe status of ioLogik R1000 Module.

C/C++

```
int R1K_ClearSafeStatus( int hConnection) ;
```

Visual Basic

```
Declare Function R1K_ClearSafeStatus Lib "MXIO.dll"  
(ByVal hConnection As Long) As Long
```

Arguments:

hConnection	The handle for a connection.
-------------	------------------------------

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_ReadStatus

This function code is used to read the status of the ioLogik W5000 Module.

C/C++

```
int W5K_ReadStatus( int hConnection,
                     WORD *wState,
                     WORD *wLastErrorCode);
```

Visual Basic

```
Declare Function W5K_ReadStatus Lib "MXIO.dll" (ByVal
hConnection As Long, wState As Integer, wLastErrorCode
As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
wState	Stores the Bus status in numerical format. The values are 0: Initial 1: Initial NET 2: Initial Fault 3: IO 4: IO Failed 5: Idle
wLastErrorCode	Stores the Field Power status in numerical format. The values are: 0: No error 1: Communication error. 2: Modbus error (need reboot) 3: Pending error (need reboot) 4: Out of memory error (need reboot)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Exp_Reconnect

Set reconnect to all expansion module.

C/C++

```
int W5K_Exp_Reconnect(int hConnection);
```

Visual Basic

```
Declare Function W5K_Exp_Reconnect Lib MXIO.dll (ByVal  
hConnection As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
-------------	--

Return Values:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Exp_Status

This function code is used to read the status of the ioLogik W5000 Module..

C/C++

```
int W5K_Exp_Status( int hConnection,  
                     WORD *wState);
```

Visual Basic

```
Declare Function W5K_Exp_Status Lib MXIO.dll (ByVal  
hConnection As Long, wState As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
wState	Stores the Bus status in numerical format. The values are 0: Expansion module offline 1: Expansion module online 2: Expansion module unmatch type 3: module returns modbus exception 4: network err

W5K_ReadSlotAmount

This function code is used to read number of expansion slots.

C/C++

```
int W5K_ReadSlotAmount( int hConnection,
                        WORD *wAmount);
```

Visual Basic

```
Declare Function W5K_ReadSlotAmount Lib MXIO.dll
(ByVal hConnection As Long, iAmount As Integer) As
Long
```

Arguments:

hConnection	The handle for a connection.
Wamount	A pointer that stores the number of expansion slots

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_ReadLastSlotIndex

This function code is used to read last slot number of expansion slots.

C/C++

```
int W5K_ReadLastSlotIndex( int hConnection,
                           WORD *wLastSlotIndex );
```

Visual Basic

```
Declare Function W5K_ReadLastSlotIndex Lib MXIO.dll
(ByVal hConnection As Long, iLastSlotIndex As Integer)
As Long
```

Arguments:

hConnection	The handle for a connection.
wLastSlotIndex	A pointer that stores the last slot number of expansion slots

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Modbus_List

This function code is used to set the IO image map mode of ioLogik W5000 Module.

C/C++

```
int W5K_Modbus_List ( int hConnection,
                      char * FilePath);
```

Visual Basic

```
Declare Function W5K_Modbus_List Lib MXIO.dll (ByVal
hConnection As Long, ByVal FilePath As String) As Long
```

Arguments:

hConnection	The handle for a connection.
FilePath	Specific log file path and file name to save module's IO image map list file. Ex. Char * FilePath = c:\\modbus.txt;

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DIO_GetIOModes_Ex

This function code is used to get contiguous channel~~?????~~s DI/DO mode.

C/C++

```
int W5K_DIO_GetIOModes_Ex ( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD * dwMode,
                            BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_DIO_GetIOModes_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nMode As Long, ByVal bytSlot As Byte) As
Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous channel ????? s DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel ????? s status. The values are : 0: INPUT DI mode 1: OUTPUT DO mode
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
---------	---------

Fail

Refer to Return Codes.

W5K_DO_Reads_Ex

This function code is used to read the output statuses of contiguous D/O channels.

C/C++

```
int W5K_DO_Reads_Ex( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD * dwValue,
                      BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_DO_Reads_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As Long, ByVal bytSlot As Byte) As
Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	A pointer that stores the contiguous D/O channel???'s status; each bit holds one channel value. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DO_Writes_Ex

This function code is used to write the output statuses of contiguous D/O channels.

C/C++

```
int W5K_DO_Writes_Ex( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD dwValue,
                      BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_DO_Writes_Ex Lib MXIO.dll (ByVal
hConnection As Long,
ByVal bytStartChannel As Byte, ByVal bytCount As Byte,
ByVal nValue As Long, ByVal bytSlot As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be written.
dwValue	Stores the channel statuses of contiguous D/O channels; each bit holds one channel status. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the second digital output channel's status. The values of the unused bits are random.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DO_GetModes_Ex

This function code is used to get the mode of contiguous D/O channels.

C/C++

```
int W5K_DO_GetModes_Ex( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wMode[ ],
                        BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_DO_GetModes_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMode As Integer, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wMode	An array that stores the mode of contiguous D/O channels. The values are: 0: D/O mode 1: Pulse mode
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DI_Reads_Ex

This function code is used to read the status of a group of contiguous D/I channels.

C/C++

```
int W5K_DI_Reads_Ex( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD * dwValue,
                      BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_DI_Reads_Ex Lib MXIO.dll (ByVal
hConnection As Long,
ByVal bytStartChannel As Byte, ByVal bytCount As Byte,
nValue As Long, ByVal bytSlot As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
wCount	The number of channels to be read.
dwValue	A pointer that stores the contiguous D/I channel's values; each bit holds one channel value. A bit value of 0 represents the digital input status of the start channel. A bit value of 1 represents the second digital input channel's status. The values of the unused bits are random.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DI_GetModes_Ex

This function code is used to get the mode of contiguous D/I channels.

C/C++

```
int W5K_DI_GetModes_Ex(int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wMode[ ]);
```

Visual Basic

```
Declare Function W5K_DI_GetModes_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, wMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wMode	An array that stores contiguous D/I channel's mode. wMode[0] represents the value of the starting channel. The values are: 0: D/I Mode 1: Count Mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Pulse_GetStartStatuses_Ex

This function code is used to get the contiguous pulse channel~~?????~~s start status.

C/C++

```
int W5K_Pulse_GetStartStatuses_Ex( int hConnection,
                                    BYTE bytStartChannel,
                                    BYTE bytCount,
                                    DWORD * dwStatus,
                                    BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_Pulse_GetStartStatuses_Ex Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nStatus As Long, ByVal bytSlot As
Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwStatus	An point that stores the contiguous pulse channel ????? s start status, each bit holds one channel value. A bit value of 0 represents the start status of the start channel. A bit value of 1 represents the status of the second pulse channel. The values are: 0: stop 1: start
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed MXIO_OK

Fail

Refer to Return Codes.

W5K_Pulse_SetStartStatuses_Ex

This function code is used to set the contiguous pulse channel~~?????~~'s start status.

C/C++

```
int W5K_Pulse_SetStartStatuses_Ex( int hConnection,
                                    BYTE bytStartChannel,
                                    BYTE bytCount,
                                    DWORD dwStatus,
                                    BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_Pulse_SetStartStatuses_Ex Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nStatus As Long, ByVal
bytSlot As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	An point that stores the contiguous pulse channel ????? 's start status, each bit holds one channel value. A bit value of 0 represents the start status of the start channel. A bit value of 1 represents the status of the second pulse channel. The values are: 0: stop 1: start
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed MXIO_OK

Fail

Refer to Return Codes.

W5K_Cnt_Reads_Ex

This function code is used to read contiguous channel~~?????~~s count value when D/I channels in ~~?????~~Count~~?????~~ mode.

C/C++

```
int W5K_Cnt_Reads_Ex( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD dwValue[ ] ,
                      BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_Cnt_Reads_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As Long, ByVal bytSlot As Byte) As
Long
```

Arguments:

hConnection	The handle for an I/O device ????? s connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	An array that stores the contiguous channel ????? s count value , dwValue[0] represents the value of the starting channel.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_Clears_Ex

This function code is used to clear contiguous channel~~?????~~s count value when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int W5K_Cnt_Clears_Ex( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_Cnt_Clears_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, ByVal bytSlot As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device ????? s connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_Reads_Ex

This function code is used to read contiguous channel~~◆◆◆~~s analog input value.

C/C++

```
int W5K_AI_Reads_Ex( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] ,
                      BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_AI_Reads_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel ◆◆◆ s value, dValue [0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadRaws_Ex

This function code is used to read contiguous channel**?????**s analog input raw data.

C/C++

```
int W5K_AI_ReadRaws_Ex( int hConnection,
                         BYTE bytStartChannel,
                         BYTE bytCount,
                         WORD wValue[ ] ,
                         BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_AI_ReadRaws_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel ????? s raw data , wValue[0] represents the value of the starting channel.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadMins_Ex

This function code is used to read contiguous A/I channel~~?????~~s minimize value.

C/C++

```
int W5K_AI_ReadMins_Ex( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        double dValue[ ] ,
                        BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_AI_ReadMins_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel ????? s value, dValue [0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadMinRaws_Ex

This function code is used to read contiguous A/I channel?/?s minimize raw data.

C/C++

```
int W5K_AI_ReadMinRaws_Ex( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            WORD wValue[ ] ,
                            BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_AI_ReadMinRaws_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel?/?s minimize raw data , wValue[0] represents the value of the starting channel.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadMaxs_Ex

This function code is used to read contiguous A/I channel~~?????~~s maximize value.

C/C++

```
int W5K_AI_ReadMaxs_Ex( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        double dValue[ ] ,
                        BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_AI_ReadMaxs_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel ????? s maximize value , dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadMaxRaws_Ex

This function code is used to read contiguous A/I channel?♦?♦?s maximize raw data.

C/C++

```
int W5K_AI_ReadMaxRaws_Ex( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            WORD wValue[ ] ,
                            BYTE BytSlot) ;
```

Visual Basic

```
Declare Function W5K_AI_ReadMaxRaws_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel?♦?♦?s maximize raw data , wValue[0] represents the value of the starting channel.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_GetRanges_Ex

This function code is used to get contiguous A/I channel~~???'s~~ range.

C/C++

```
int W5K_AI_GetRanges_Ex( int hConnection,
                         BYTE bytStartChannel,
                         BYTE bytCount,
                         WORD wRange[ ] ,
                         BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_AI_GetRanges_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iRange As integer, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wRange	An array that stores the contiguous A/I channel ???'s range, wRange[0] represents the value of the starting channel. The values are: 00: 0-10V 01: 4-20mA Others_return Illegal Data Value
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_VC_Reads_Ex

This function code is used to read contiguous channel????s virtual channel????s value.

C/C++

```
int W5K_VC_Reads_Ex( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] ,
                      BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_VC_Reads_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous virtual channel????s value, dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.
bytSlot	Slot number of the I/O module. The Slot number is 0. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DI_Reads

This function code is used to read the status of a group of contiguous D/I channels.

C/C++

```
int DI_Reads( int hConnection,
               BYTE bytSlot,
               BYTE bytStartChannel,
               BYTE bytCount,
               DWORD * dwValue);
```

Visual Basic

```
Declare Function DI_Reads Lib "MXIO.dll" (ByVal hConnection
As Long, ByVal bytSlot As Byte, ByVal bytStartChannel As
Byte, ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
wCount	The number of channels to be read.
dwValue	A pointer that stores the contiguous D/I channel's values; each bit holds one channel value. A bit value of 0 represents the digital input status of the start channel. A bit value of 1 represents the second digital input channel's status. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DI_Read

This function code is used to read the status of a specific D/I channel.

C/C++

```
int DI_Read( int hConnection,
              BYTE bytSlot,
              BYTE bytChannel,
              BYTE * bytValue);
```

Visual Basic

```
Declare Function DI_Read Lib "MXIO.dll" (ByVal hConnection As Long, ByVal bytSlot As Byte, ByVal bytChannel As Byte, bytValue As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be read.
bytValue	A pointer that stores a specific D/I channel status.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DI2K_GetModes

This function code is used to get the mode of contiguous D/I channels.

C/C++

```
int DI2K_GetModes(int hConnection,
                   BYTE bytStartChannel,
                   BYTE bytCount,
                   WORD wMode[ ]);
```

Visual Basic

```
Declare Function DI2K_GetModes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, wMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wMode	An array that stores contiguous D/I channel's mode. wMode[0] represents the value of the starting channel. The values are: 0: D/I Mode 1: Count Mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DI2K_SetModes

This function code is used to set the mode of contiguous D/I channels.

C/C++

```
int DI2K_SetModes( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    WORD wMode[ ] );
```

Visual Basic

```
Declare Function DI2K_SetModes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be sets.
wMode	An array that stores contiguous D/I channel's mode. wMode[0] represents the value of the starting channel. The values are: 0: D/I Mode 1: Count Mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DI2K_GetMode

This function code is used to get the mode of a specific D/I channel.

C/C++

```
int DI2K_GetMode( int hConnection,  
                    BYTE bytChannel,  
                    WORD * wMode );
```

Visual Basic

```
Declare Function DI2K_GetMode Lib "MXIO.dll" (ByVal  
hConnection As Long, ByVal bytChannel As Byte, iMode As  
Integer) As Long
```

Arguments :

hConnection The handle for an I/O device's connection.

`bytChannel` The specific channel to be get.

wMode A pointer that stores a specific D/I channel's mode. The values are:

0 : D/I Mode

1: Count Mode

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

DI2K_SetMode

This function code is used to set the mode of a specific D/I channel.

C/C++

```
int DI2K_SetMode( int hConnection,  
                    BYTE bytChannel,  
                    WORD wMode);
```

Visual Basic

```
Declare Function DI2K_SetMode Lib "MXIO.dll" (ByVal  
hConnection As Long, ByVal bytChannel As Byte, ByVal wMode  
As Integer) As Long
```

Arguments :

hConnection The handle for an I/O device's connection.

`bytChannel` The specific channel to be set.

wMode A pointer that stores a specific D/I channel's mode. The values are:

0 : D/I Mode

1: Count Mode

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

DI2K_GetFilters

This function code is used to get the filter of contiguous D/I channels.

C/C++

```
int DI2K_GetFilters( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wFilter[ ] );
```

Visual Basic

```
Declare Function DI2K_GetFilters Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wFilter	An array that stores contiguous D/I channel's filter value. wFilter[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DI2K_SetFilters

This function code is used to set the filter of contiguous D/I channels.

C/C++

```
int DI2K_SetFilters( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     WORD wFilter[ ] );
```

Visual Basic

```
Declare Function DI2K_SetFilters Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be sets.
wFilter	An array that stores contiguous D/I channel's filter value. wFilter[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DI2K_GetFilter

This function code is used to get the filter of a specific D/I channel.

C/C++

```
int DI_GetFilter( int hConnection,
                  BYTE bytChannel,
                  WORD * wFilter);
```

Visual Basic

```
Declare Function DI2K_GetFilter Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iFilter As
Integer) As Long
```

Arguments:

hConnection The handle for an I/O device's connection.

bytChannel The specific channel to be get.

wFilter A pointer that stores a specific D/I channel's filter value.

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

DI2K_SetFilter

This function code is used to set the filter of a specific D/I channel.

C/C++

```
int DI2K_SetFilter( int hConnection,
                     BYTE bytChannel,
                     WORD wFilter);
```

Visual Basic

```
Declare Function DI2K_SetFilter Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal
wFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytChannel	The specific channel to be set.
wFilter	A pointer that stores a specific D/I channel's filter value.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_Reads

This function code is used to read contiguous channel's count value when D/I channels in 'Count' mode.

C/C++

```
int Cnt2K_Reads( int hConnection,
                  BYTE bytStartChannel,
                  BYTE bytCount,
                  DWORD dwValue[ ] );
```

Visual Basic

```
Declare Function Cnt2K_Reads Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	An array that stores the contiguous channel's count value , dwValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_Clears

This function code is used to clear contiguous channel's count value when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_Clears( int hConnection,
                   BYTE bytStartChannel,
                   BYTE bytCount);
```

Visual Basic

```
Declare Function Cnt2K_Clears Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_Read

This function code is used to read the count value when specific D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_Read( int hConnection,
                  BYTE bytChannel,
                  DWORD * dwValue);
```

Visual Basic

```
Declare Function Cnt2K_Read Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, nValue As
Long) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytChannel	The specific channel to be read.
dwValue	A pointer that stores the count value for specific channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_Clear

This function code is used to clear count value when specific D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_Clear( int hConnection,
                  BYTE bytChannel);
```

Visual Basic

```
Declare Function Cnt2K_Clear Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be clear.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetOverflows

This function code is used to get contiguous channel's overflow status when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_GetOverflows( int hConnection,
                        BYTE bytStartChannel
                        BYTE bytCount
                        DWORD * dwStatus) ;
```

Visual Basic

```
Declare Function Cnt2K_GetOverflows Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
dwStatus	A pointer that stores the contiguous channel's overflow status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: Normal 1: Overflow

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_ClearOverflows

This function code is used to clear contiguous channel's overflow status when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_ClearOverflows( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount) ;
```

Visual Basic

```
Declare Function Cnt2K_ClearOverflows Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetOverflow

This function code is used to get overflow status when specific D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_GetOverflow( int hConnection,
                      BYTE bytChannel,
                      BYTE * bytStatus) ;
```

Visual Basic

```
Declare Function Cnt2K_GetOverflow Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, bytStatus As
Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
bytStatus	A pointer that stores the overflow status for specific channel. The values are : 0: Normal 1: Overflow

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_ClearOverflow

This function code is used to clear specific channel's overflow status when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_ClearOverflow( int hConnection,
                         BYTE bytChannwl);
```

Visual Basic

```
Declare Function Cnt2K_ClearOverflow Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be clear.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetFilters

This function code is used to get contiguous channel's filter value when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_GetFilters( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wFilter[ ] );
```

Visual Basic

```
Declare Function Cnt2K_GetFilters Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wFilter	An array that stored the filter value for contiguous D/I channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_SetFilters

This function code is used to set contiguous channel's filter value when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_SetFilters( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wFilter[ ]);
```

Visual Basic

```
Declare Function Cnt2K_SetFilters Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wFilter	An array that stored the filter value for contiguous D/I channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetFilter

This function code is used to get specific channel's filter value when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_GetFilter( int hConnection,
                      BYTE bytChannel,
                      WORD * wFilter);
```

Visual Basic

```
Declare Function Cnt2K_GetFilter Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iFilter As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
wHiWidth	A pointer that stored the filter value.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_SetFilter

This function code is used to set specific channel's filter value when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_SetFilter( int hConnection,
                      BYTE bytChannel,
                      WORD wFilter, );
```

Visual Basic

```
Declare Function Cnt2K_SetFilter Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal
iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
wHiWidth	stored the filter value.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetStartStatuses

This function code is used to get contiguous channel's start status when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_GetStartStatuses( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD * dwStatus);
```

Visual Basic

```
Declare Function Cnt2K_GetStartStatuses Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous count channel's start status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_SetStartStatuses

This function code is used to set contiguous channel's start status when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_SetStartStatuses( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD dwStatus);
```

Visual Basic

```
Declare Function Cnt2K_SetStartStatuses Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous count channel's start status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetStartStatus

This function code is used to get specific channel's start status when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_GetStartStatus( int hConnection,
                           BYTE bytChannel,
                           BYTE * bytStatus);
```

Visual Basic

```
Declare Function Cnt2K_GetStartStatus Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, bytStatus As
Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
bytStatus	A pointer that stores the specific count channel's start status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_SetStartStatus

This function code is used to set specific channel's start status when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_SetStartStatus( int hConnection,
                           BYTE bytChannel,
                           BYTE bytStatus);
```

Visual Basic

```
Declare Function Cnt2K_SetStartStatus Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal
bytStatus As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
bytStatus	A pointer that stores the specific count channel's start status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetTriggerTypes

This function code is used to get contiguous channel's trigger types when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_GetTriggerTypes( int hConnection,
                           BYTE bytStartChannel
                           BYTE bytCount
                           DWORD * dwType);
```

Visual Basic

```
Declare Function Cnt2K_GetTriggerTypes Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nType As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
dwType	A pointer that stores the contiguous channel's triggers types; each bit holds one channel trigger type. A bit value of 0 represents the trigger type of the start channel. A bit value of 1 represents the second channel's trigger type. The values are : 0: LoToHi 1: HiToLo

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_SetTriggerTypes

This function code is used to set contiguous channel's trigger types when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_SetTriggerTypes( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwType) ;
```

Visual Basic

```
Declare Function Cnt2K_SetTriggerTypes Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nType As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear
dwType	Stored the contiguous channel's triggers types; each bit holds one channel trigger type. A bit value of 0 represents the trigger type of the start channel. A bit value of 1 represents the second channel's trigger type. The values are : 0: LoToHi 1: HiToLo

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetTriggerType

This function code is used to get trigger type when specific D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_GetTriggerType( int hConnection,
                           BYTE bytChannel,
                           BYTE * bytType);
```

Visual Basic

```
Declare Function Cnt2K_GetTriggerType Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, bytType As
Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
bytType	A pointer that stores the trigger type for specific channel. The values are : 0 LoToHi 1 HiToLo

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_SetTriggerType

This function code is used to set specific channel's trigger type when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_SetTriggerType( int hConnection,
                           BYTE bytChannel,
                           BYTE bytType);
```

Visual Basic

```
Declare Function Cnt2K_SetTriggerType Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal
bytType as Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be clear.
bytType	stores the trigger type for specific channel. The values are : 0 LoToHi 1 HiToLo

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetPowerOnValues

This function code is used to get contiguous channel's power on values when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_GetPowerOnValues( int hConnection,
                            BYTE bytStartChannel
                            BYTE bytCount
                            DWORD * dwValue) ;
```

Visual Basic

```
Declare Function Cnt2K_GetPowerOnValues Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
dwValue	A pointer that stores the contiguous channel's power on values; each bit holds one channel power on value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the second channel's power on value. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_SetPowerOnValues

This function code is used to set contiguous channel's power on values when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_SetPowerOnValues( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD dwValue);
```

Visual Basic

```
Declare Function Cnt2K_SetPowerOnValues Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear
dwValue	Stored the contiguous channel's power on values; each bit holds one channel power on value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the second channel's power on value. The values are: 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetPowerOnValue

This function code is used to get power on value when specific D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_GetPowerOnValue(int hConnection,  
                           BYTE bytChannel,  
                           BYTE * bytValue);
```

Visual Basic

```
Declare Function Cnt2K_GetPowerOnValue Lib "MXIO.dll"  
          (ByVal hConnection As Long, ByVal bytChannel As Byte,  
           bytValue As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
bytValue	A pointer that stores the power on value for specific channel. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_SetPowerOnValue

This function code is used to set specific channel's power on value when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_SetPowerOnValue(int hConnection,  
                           BYTE bytChannel,  
                           BYTE bytValue);
```

Visual Basic

```
Declare Function Cnt2K_SetPowerOnValue Lib "MXIO.dll"  
          (ByVal hConnection As Long, ByVal bytChannel As Byte, ByVal  
           bytValue as Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be clear.
bytValue	stores the power on value for specific channel. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetSafeValues

This function code is used to get contiguous channel's safe values when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_GetSafeValues( int hConnection,
                        BYTE bytStartChannel
                        BYTE bytCount
                        DWORD * dwValue) ;
```

Visual Basic

```
Declare Function Cnt2K_GetSafeValues Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
dwValue	A pointer that stores the contiguous channel's safe values; each bit holds one channel safe value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the second channel's safe value. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_SetSafeValues

This function code is used to set contiguous channel's safe values when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_SetSafeValues( int hConnection,
                         BYTE bytStartChannel,
                         BYTE bytCount,
                         DWORD dwValue) ;
```

Visual Basic

```
Declare Function Cnt2K_SetSafeValues Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear
dwValue	Stored the contiguous channel's safe values; each bit holds one channel safe value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the second channel's safe value. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetSafeValue

This function code is used to get safe value when specific D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_GetSafeValue( int hConnection,
                        BYTE bytChannel,
                        BYTE * bytValue) ;
```

Visual Basic

```
Declare Function Cnt2K_GetSafeValue Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, bytValue As
Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
bytValue	A pointer that stores the safe value for specific channel. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_SetSafeValue

This function code is used to set specific channel's safe value when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_SetSafeValue( int hConnection,
                        BYTE bytChannel,
                        BYTE bytValue);
```

Visual Basic

```
Declare Function Cnt2K_SetSafeValue Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal
bytValue as Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be clear.
bytType	stores the safe value for specific channel. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetTriggerTypeWords

This function code is used to get contiguous channel's trigger types when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_GetTriggerTypeWords(int hConnection,
                               BYTE bytStartChannel
                               BYTE bytCount
                               WORD * wType);
```

Visual Basic

```
Declare Function Cnt2K_GetTriggerTypeWords Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iType As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
wType	A pointer that stores the contiguous channel's triggers types; each bit holds one channel trigger type. A bit value of 0 represents the trigger type of the start channel. A bit value of 1 represents the second channel's trigger type. The values are : 0: LoToHi 1: HiToLo 2: Both

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_SetTriggerTypeWords

This function code is used to set contiguous channel's trigger types when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_SetTriggerTypeWords( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               WORD * wType);
```

Visual Basic

```
Declare Function Cnt2K_SetTriggerTypeWords Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iType As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear
wType	Stored the contiguous channel's triggers types; each bit holds one channel trigger type. A bit value of 0 represents the trigger type of the start channel. A bit value of 1 represents the second channel's trigger type. The values are : 0: LoToHi 1: HiToLo 2: Both

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetTriggerTypeWord

This function code is used to get trigger type when specific D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_GetTriggerTypeWord( int hConnection,
                               BYTE bytChannel,
                               WORD * wType);
```

Visual Basic

```
Declare Function Cnt2K_GetTriggerTypeWord Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytChannel As Byte, wType
As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
wType	A pointer that stores the trigger type for specific channel. The values are : 0: LoToHi 1: HiToLo 2: Both

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_SetTriggerTypeWord

This function code is used to set specific channel's trigger type when D/I channel in 'Count' mode.

C/C++

```
int Cnt2K_SetTriggerTypeWord( int hConnection,
                               BYTE bytChannel,
                               WORD wType);
```

Visual Basic

```
Declare Function Cnt2K_SetTriggerTypeWord Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytChannel As Byte, ByVal
    wType as Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be clear.
wType	stores the trigger type for specific channel. The values are : 0: LoToHi 1: HiToLo 2: Both

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_GetSaveStatusesOnPowerFail

This function code is used to get contiguous channel's DI/DO power off storage enable mode.

C/C++

```
int Cnt2K_GetSaveStatusesOnPowerFail ( int hConnection,
                                         BYTE bytStartChannel,
                                         BYTE bytCount,
                                         DWORD * dwMode );
```

Visual Basic

```
Declare Function Cnt2K_GetSaveStatusesOnPowerFail Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, nMode As
Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get.
dwStatus	A pointer that stores the contiguous channel's DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: ON, 1: OFF

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Cnt2K_SetSaveStatusesOnPowerFail

This function code is used to set contiguous channel's DI/DO mode power off storage enable mode.

C/C++

```
int Cnt2K_SetSaveStatusesOnPowerFail ( int hConnection,
                                         BYTE bytStartChannel,
                                         BYTE bytCount,
                                         DWORD dwMode );
```

Visual Basic

```
Declare Function Cnt2K_SetSaveStatusesOnPowerFail Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, ByVal
nMode As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous channel's DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: ON, 1: OFF

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_DI_Reads

This function code is used to read the status of a group of contiguous D/I channels.

C/C++

```
int E42_DI_Reads( int hConnection,
                   BYTE bytSlot,
                   BYTE bytStartChannel,
                   BYTE bytCount,
                   DWORD * dwValue) ;
```

Visual Basic

```
Declare Function E42_DI_Reads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, nValue As
Long) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
wCount	The number of channels to be read.
dwValue	A pointer that stores the contiguous D/I channel's values; each bit holds one channel value. A bit value of 0 represents the digital input status of the start channel. A bit value of 1 represents the second digital input channel's status. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DI_Reads

This function code is used to read the status of a group of contiguous D/I channels.

C/C++

```
int W5K_DI_Reads( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    DWORD * dwValue) ;
```

Visual Basic

```
Declare Function W5K_DI_Reads Lib MXIO.dll (ByVal
hConnection As Long,
ByVal bytStartChannel As Byte, ByVal bytCount As Byte,
nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
wCount	The number of channels to be read.
dwValue	A pointer that stores the contiguous D/I channel's values; each bit holds one channel value. A bit value of 0 represents the digital input status of the start channel. A bit value of 1 represents the second digital input channel's status. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DI_GetModes

This function code is used to get the mode of contiguous D/I channels.

C/C++

```
int W5K_DI_GetModes(int hConnection,  
                     BYTE bytStartChannel,  
                     BYTE bytCount,  
                     WORD wMode[ ]);
```

Visual Basic

```
Declare Function W5K_DI_GetModes Lib MXIO.dll (ByVal  
hConnection As Long, ByVal bytStartChannel As Byte, ByVal  
bytCount As Byte, wMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wMode	An array that stores contiguous D/I channel's mode. wMode[0] represents the value of the starting channel. The values are: 0: D/I Mode 1: Count Mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DI_SetModes

This function code is used to set the mode of contiguous D/I channels.

C/C++

```
int W5K_DI_SetModes( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wMode[ ] );
```

Visual Basic

```
Declare Function W5K_DI_SetModes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be sets.
wMode	An array that stores contiguous D/I channel's mode. wMode[0] represents the value of the starting channel. The values are: 0: D/I Mode 1: Count Mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DI_GetFilters

This function code is used to get the filter of contiguous D/I channels.

C/C++

```
int W5K_DI_GetFilters( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wFilter[ ] );
```

Visual Basic

```
Declare Function W5K_DI_GetFilters Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wFilter	An array that stores contiguous D/I channel's filter value. wFilter[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DI_SetFilters

This function code is used to set the filter of contiguous D/I channels.

C/C++

```
int W5K_DI_SetFilters( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wFilter[ ] );
```

Visual Basic

```
Declare Function W5K_DI_SetFilters Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be sets.
wFilter	An array that stores contiguous D/I channel's filter value. wFilter[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_Reads

This function code is used to read contiguous channel~~?????~~s count value when D/I channels in ~~?????~~Count~~?????~~ mode.

C/C++

```
int W5K_Cnt_Reads( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     DWORD dwValue[ ] );
```

Visual Basic

```
Declare Function W5K_Cnt_Reads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device ????? s connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	An array that stores the contiguous channel ????? s count value , dwValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_Clears

This function code is used to clear contiguous channel~~?????~~s count value when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int W5K_Cnt_Clears( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount);
```

Visual Basic

```
Declare Function W5K_Cnt_Clears Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device ????? s connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_GetOverflows

This function code is used to get contiguous channel~~?????~~s overflow status when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int W5K_Cnt_GetOverflows( int hConnection,
                           BYTE bytStartChannel
                           BYTE bytCount
                           DWORD * dwStatus) ;
```

Visual Basic

```
Declare Function W5K_Cnt_GetOverflows Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
dwStatus	A pointer that stores the contiguous channel ????? s overflow status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel ????? s status. The values are : 0: Normal 1: Overflow

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_ClearOverflows

This function code is used to clear contiguous channel **???s** overflow status when D/I channel in **???Count???** mode.

C/C++

```
int W5K_Cnt_ClearOverflows( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount);
```

Visual Basic

```
Declare Function W5K_Cnt_ClearOverflows Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_GetFilters

This function code is used to get contiguous channel~~?????~~s filter value when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int W5K_Cnt_GetFilters( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wFilter[ ] );
```

Visual Basic

```
Declare Function W5K_Cnt_GetFilters Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wFilter	An array that stored the filter value for contiguous D/I channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_SetFilters

This function code is used to set contiguous channel's filter value when D/I channel in Count mode.

C/C++

```
int W5K_Cnt_SetFilters( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wFilter[ ] );
```

Visual Basic

```
Declare Function W5K_Cnt_SetFilters Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wFilter	An array that stored the filter value for contiguous D/I channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_GetStartStatuses

This function code is used to get contiguous channel~~?????~~s start status when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int W5K_Cnt_GetStartStatuses( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD * dwStatus) ;
```

Visual Basic

```
Declare Function W5K_Cnt_GetStartStatuses Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous count channel ????? s start status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel ????? s status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_SetStartStatuses

This function code is used to set contiguous channel???'s start status when D/I channel in ??'Count??' mode.

C/C++

```
int W5K_Cnt_SetStartStatuses( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD dwStatus );
```

Visual Basic

```
Declare Function W5K_Cnt_SetStartStatuses Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous count channel???'s start status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel???'s status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_GetPowerOnValues

This function code is used to get contiguous channel~~???'s~~ power on values when D/I channel in ~~???Count~~~~???~~ mode.

C/C++

```
int W5K_Cnt_GetPowerOnValues( int hConnection,
                               BYTE bytStartChannel
                               BYTE bytCount
                               DWORD * dwValue) ;
```

Visual Basic

```
Declare Function W5K_Cnt_GetPowerOnValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
dwValue	A pointer that stores the contiguous channel ???'s power on values; each bit holds one channel power on value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the second channel ??? 's power on value. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_SetPowerOnValues

This function code is used to set contiguous channel~~?????~~s power on values when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int W5K_Cnt_SetPowerOnValues( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD dwValue);
```

Visual Basic

```
Declare Function W5K_Cnt_SetPowerOnValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear
dwValue	Stored the contiguous channel ????? s power on values; each bit holds one channel power on value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the second channel ????? s power on value. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_GetSafeValues

This function code is used to get contiguous channel~~?????~~'s safe values when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int W5K_Cnt_GetSafeValues( int hConnection,
                           BYTE bytStartChannel
                           BYTE bytCount
                           DWORD * dwValue) ;
```

Visual Basic

```
Declare Function W5K_Cnt_GetSafeValues Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
dwValue	A pointer that stores the contiguous channel ????? 's safe values; each bit holds one channel safe value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the second channel ????? 's safe value. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_SetSafeValues

This function code is used to set contiguous channel~~?????~~'s safe values when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int W5K_Cnt_SetSafeValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwValue );
```

Visual Basic

```
Declare Function W5K_Cnt_SetSafeValues Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear
dwValue	Stored the contiguous channel ????? 's safe values; each bit holds one channel safe value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the second channel ????? 's safe value. The values are: 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_GetTriggerTypeWords

This function code is used to get contiguous channel~~?????~~s trigger types when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int W5K_Cnt_GetTriggerTypeWords(int hConnection,
                                 BYTE bytStartChannel
                                 BYTE bytCount
                                 WORD * wType);
```

Visual Basic

```
Declare Function W5K_Cnt_GetTriggerTypeWords Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iType As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
wType	A pointer that stores the contiguous channel ????? s triggers types; each bit holds one channel trigger type. A bit value of 0 represents the trigger type of the start channel. A bit value of 1 represents the second channel ????? s trigger type. The values are : 0: LoToHi 1: HiToLo 2: Both

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_SetTriggerTypeWords

This function code is used to set contiguous channel~~???~~s trigger types when D/I channel in ~~???~~Count~~???~~ mode.

C/C++

```
int W5K_Cnt_SetTriggerTypeWords( int hConnection,
                                 BYTE bytStartChannel,
                                 BYTE bytCount,
                                 WORD * wType);
```

Visual Basic

```
Declare Function W5K_Cnt_SetTriggerTypeWords Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iType As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear
wType	Stored the contiguous channel ??? s triggers types; each bit holds one channel trigger type. A bit value of 0 represents the trigger type of the start channel. A bit value of 1 represents the second channel ??? s trigger type. The values are : 0: LoToHi 1: HiToLo 2: Both

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_GetSaveStatusesOnPowerFail

This function code is used to get contiguous channel#???'s DI/DO power off storage enable mode.

C/C++

```
int W5K_Cnt_GetSaveStatusesOnPowerFail ( int hConnection,
                                         BYTE bytStartChannel,
                                         BYTE bytCount,
                                         DWORD * dwMode );
```

Visual Basic

```
Declare Function W5K_Cnt_GetSaveStatusesOnPowerFail Lib
MXIO.dll (ByVal hConnection As Long, ByVal bytStartChannel
As Byte, ByVal bytCount As Byte, nMode As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get.
dwStatus	A pointer that stores the contiguous channel#???'s DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel#???'s status. The values are : 0: ON, 1: OFF

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_SetSaveStatusesOnPowerFail

This function code is used to set contiguous channel#???'s DI/DO mode power off storage enable mode.

C/C++

```
int W5K_Cnt_SetSaveStatusesOnPowerFail ( int hConnection,
                                         BYTE bytStartChannel,
                                         BYTE bytCount,
                                         DWORD dwMode );
```

Visual Basic

```
Declare Function W5K_Cnt_SetSaveStatusesOnPowerFail Lib
MXIO.dll (ByVal hConnection As Long, ByVal bytStartChannel
As Byte, ByVal bytCount As Byte, ByVal nMode As Long) As
Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous channel#???'s DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel#???'s status. The values are : 0: ON, 1: OFF

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DI_Reads_Ex

This function code is used to read the status of a group of contiguous D/I channels.

C/C++

```
int W5K_DI_Reads_Ex( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD * dwValue,
                      BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_DI_Reads_Ex Lib MXIO.dll (ByVal
hConnection As Long,
ByVal bytStartChannel As Byte, ByVal bytCount As Byte,
nValue As Long, ByVal bytSlot As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
wCount	The number of channels to be read.
dwValue	A pointer that stores the contiguous D/I channel's values; each bit holds one channel value. A bit value of 0 represents the digital input status of the start channel. A bit value of 1 represents the second digital input channel's status. The values of the unused bits are random.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DI_GetModes_Ex

This function code is used to get the mode of contiguous D/I channels.

C/C++

```
int W5K_DI_GetModes_Ex(int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wMode[ ]);
```

Visual Basic

```
Declare Function W5K_DI_GetModes_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, wMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wMode	An array that stores contiguous D/I channel's mode. wMode[0] represents the value of the starting channel. The values are: 0: D/I Mode 1: Count Mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_Reads_Ex

This function code is used to read contiguous channel~~?????~~s count value when D/I channels in ~~?????~~Count~~?????~~ mode.

C/C++

```
int W5K_Cnt_Reads_Ex( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD dwValue[ ] ,
                      BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_Cnt_Reads_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As Long, ByVal bytSlot As Byte) As
Long
```

Arguments:

hConnection	The handle for an I/O device ????? s connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	An array that stores the contiguous channel ????? s count value , dwValue[0] represents the value of the starting channel.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Cnt_Clears_Ex

This function code is used to clear contiguous channel~~?????~~s count value when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int W5K_Cnt_Clears_Ex( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_Cnt_Clears_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, ByVal bytSlot As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device ????? s connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DI_Reads

This function code is used to read the status of a group of contiguous D/I channels.

C/C++

```
int E1K_DI_Reads( int hConnection,
                   BYTE bytStartChannel,
                   BYTE bytCount,
                   DWORD * dwValue) ;
```

Visual Basic

```
Declare Function E1K_DI_Reads Lib MXIO.dll (ByVal
hConnection As Long,
ByVal bytStartChannel As Byte, ByVal bytCount As Byte,
nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
wCount	The number of channels to be read.
dwValue	A pointer that stores the contiguous D/I channel's values; each bit holds one channel value. A bit value of 0 represents the digital input status of the start channel. A bit value of 1 represents the second digital input channel's status. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DI_GetModes

This function code is used to get the mode of contiguous D/I channels.

C/C++

```
int E1K_DI_GetModes(int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     WORD wMode[ ]);
```

Visual Basic

```
Declare Function E1K_DI_GetModes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, wMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wMode	An array that stores contiguous D/I channel's mode. wMode[0] represents the value of the starting channel. The values are: 0: D/I Mode 1: Count Mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DI_SetModes

This function code is used to set the mode of contiguous D/I channels.

C/C++

```
int E1K_DI_SetModes( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wMode[ ] );
```

Visual Basic

```
Declare Function E1K_DI_SetModes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be sets.
wMode	An array that stores contiguous D/I channel's mode. wMode[0] represents the value of the starting channel. The values are: 0: D/I Mode 1: Count Mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DI_GetFilters

This function code is used to get the filter of contiguous D/I channels.

C/C++

```
int E1K_DI_GetFilters( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wFilter[ ] );
```

Visual Basic

```
Declare Function E1K_DI_GetFilters Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wFilter	An array that stores contiguous D/I channel's filter value. wFilter[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DI_SetFilters

This function code is used to set the filter of contiguous D/I channels.

C/C++

```
int E1K_DI_SetFilters( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wFilter[ ] );
```

Visual Basic

```
Declare Function E1K_DI_SetFilters Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be sets.
wFilter	An array that stores contiguous D/I channel's filter value. wFilter[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_Reads

This function code is used to read contiguous channel~~?????~~s count value when D/I channels in ~~?????~~Count~~?????~~ mode.

C/C++

```
int E1K_Cnt_Reads( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     DWORD dwValue[ ] );
```

Visual Basic

```
Declare Function E1K_Cnt_Reads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device ????? s connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	An array that stores the contiguous channel ????? s count value , dwValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_Clears

This function code is used to clear contiguous channel~~?????~~s count value when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int E1K_Cnt_Clears( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount);
```

Visual Basic

```
Declare Function E1K_Cnt_Clears Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device ????? s connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_GetOverflows

This function code is used to get contiguous channel~~?????~~s overflow status when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int E1K_Cnt_GetOverflows( int hConnection,
                           BYTE bytStartChannel
                           BYTE bytCount
                           DWORD * dwStatus) ;
```

Visual Basic

```
Declare Function E1K_Cnt_GetOverflows Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
dwStatus	A pointer that stores the contiguous channel ????? s overflow status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel ????? s status. The values are : 0: Normal 1: Overflow

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_ClearOverflows

This function code is used to clear contiguous channel overflow status when D/I channel in Count mode.

C/C++

```
int E1K_Cnt_ClearOverflows( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount);
```

Visual Basic

```
Declare Function E1K_Cnt_ClearOverflows Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_GetFilters

This function code is used to get contiguous channel~~?????~~s filter value when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int E1K_Cnt_GetFilters( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wFilter[ ] );
```

Visual Basic

```
Declare Function E1K_Cnt_GetFilters Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wFilter	An array that stored the filter value for contiguous D/I channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_SetFilters

This function code is used to set contiguous channel's filter value when D/I channel in Count mode.

C/C++

```
int E1K_Cnt_SetFilters( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wFilter[ ] );
```

Visual Basic

```
Declare Function E1K_Cnt_SetFilters Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wFilter	An array that stored the filter value for contiguous D/I channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_GetStartStatuses

This function code is used to get contiguous channel~~?????~~s start status when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int E1K_Cnt_GetStartStatuses( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD * dwStatus) ;
```

Visual Basic

```
Declare Function E1K_Cnt_GetStartStatuses Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous count channel ????? s start status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel ????? s status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_SetStartStatuses

This function code is used to set contiguous channel???'s start status when D/I channel in ??'Count??' mode.

C/C++

```
int E1K_Cnt_SetStartStatuses( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD dwStatus);
```

Visual Basic

```
Declare Function E1K_Cnt_SetStartStatuses Lib MXIO.dll
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, ByVal nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous count channel???'s start status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel???'s status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_GetPowerOnValues

This function code is used to get contiguous channel~~???'s~~ power on values when D/I channel in ~~???Count~~~~???~~ mode.

C/C++

```
int E1K_Cnt_GetPowerOnValues( int hConnection,
                                BYTE bytStartChannel
                                BYTE bytCount
                                DWORD * dwValue) ;
```

Visual Basic

```
Declare Function E1K_Cnt_GetPowerOnValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
dwValue	A pointer that stores the contiguous channel ???'s power on values; each bit holds one channel power on value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the second channel ??? 's power on value. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_SetPowerOnValues

This function code is used to set contiguous channel~~?????~~s power on values when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int E1K_Cnt_SetPowerOnValues( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD dwValue);
```

Visual Basic

```
Declare Function E1K_Cnt_SetPowerOnValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear
dwValue	Stored the contiguous channel ????? s power on values; each bit holds one channel power on value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the second channel ????? s power on value. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_GetSafeValues

This function code is used to get contiguous channel~~?????~~'s safe values when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int E1K_Cnt_GetSafeValues( int hConnection,
                           BYTE bytStartChannel
                           BYTE bytCount
                           DWORD * dwValue) ;
```

Visual Basic

```
Declare Function E1K_Cnt_GetSafeValues Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
dwValue	A pointer that stores the contiguous channel ????? 's safe values; each bit holds one channel safe value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the second channel ????? 's safe value. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_SetSafeValues

This function code is used to set contiguous channel~~?????~~'s safe values when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int E1K_Cnt_SetSafeValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwValue );
```

Visual Basic

```
Declare Function E1K_Cnt_SetSafeValues Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear
dwValue	Stored the contiguous channel ????? 's safe values; each bit holds one channel safe value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the second channel ????? 's safe value. The values are: 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_GetTriggerTypeWords

This function code is used to get contiguous channel~~?????~~s trigger types when D/I channel in ~~?????~~Count~~?????~~ mode.

C/C++

```
int E1K_Cnt_GetTriggerTypeWords(int hConnection,  
                                BYTE bytStartChannel  
                                BYTE bytCount  
                                WORD * wType);
```

Visual Basic

```
Declare Function E1K_Cnt_GetTriggerTypeWords Lib MXIO.dll  
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,  
ByVal bytCount As Byte, iType As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
wType	A pointer that stores the contiguous channel ????? s triggers types; each bit holds one channel trigger type. A bit value of 0 represents the trigger type of the start channel. A bit value of 1 represents the second channel ????? s trigger type. The values are : 0: LoToHi 1: HiToLo 2: Both

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_SetTriggerTypeWords

This function code is used to set contiguous channel???'s trigger types when D/I channel in ???'Count???' mode.

C/C++

```
int E1K_Cnt_SetTriggerTypeWords( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                WORD * wType);
```

Visual Basic

```
Declare Function E1K_Cnt_SetTriggerTypeWords Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iType As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear
wType	Stored the contiguous channel???'s triggers types; each bit holds one channel trigger type. A bit value of 0 represents the trigger type of the start channel. A bit value of 1 represents the second channel???'s trigger type. The values are : 0: LoToHi 1: HiToLo 2: Both

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_GetSaveStatusesOnPowerFail

This function code is used to get contiguous channel#???'s DI/DO power off storage enable mode.

C/C++

```
int E1K_Cnt_GetSaveStatusesOnPowerFail ( int hConnection,
                                         BYTE bytStartChannel,
                                         BYTE bytCount,
                                         DWORD * dwMode );
```

Visual Basic

```
Declare Function E1K_Cnt_GetSaveStatusesOnPowerFail Lib
MXIO.dll (ByVal hConnection As Long, ByVal bytStartChannel
As Byte, ByVal bytCount As Byte, nMode As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get.
dwStatus	A pointer that stores the contiguous channel#???'s DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel#???'s status. The values are : 0: ON, 1: OFF

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Cnt_SetSaveStatusesOnPowerFail

This function code is used to set contiguous channel#???'s DI/DO mode power off storage enable mode.

C/C++

```
int E1K_Cnt_SetSaveStatusesOnPowerFail ( int hConnection,
                                         BYTE bytStartChannel,
                                         BYTE bytCount,
                                         DWORD dwMode );
```

Visual Basic

```
Declare Function E1K_Cnt_SetSaveStatusesOnPowerFail Lib
MXIO.dll (ByVal hConnection As Long, ByVal bytStartChannel
As Byte, ByVal bytCount As Byte, ByVal nMode As Long) As
Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous channel#???'s DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel#???'s status. The values are : 0: ON, 1: OFF

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO_Reads

This function code is used to read the output statuses of contiguous D/O channels.

C/C++

```
int DO_Reads( int hConnection,
               BYTE bytSlot,
               BYTE bytStartChannel,
               BYTE bytCount,
               DWORD * dwValue);
```

Visual Basic

```
Declare Function DO_Reads Lib "MXIO.dll" (ByVal hConnection
As Long, ByVal bytSlot As Byte, ByVal bytStartChannel As
Byte, ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. Slot numbers range from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	A pointer that stores the contiguous D/O channel's status; each bit holds one channel value. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO_Writes

This function code is used to write the output statuses of contiguous D/O channels.

C/C++

```
int DO_Writes( int hConnection,
                BYTE bytSlot,
                BYTE bytStartChannel,
                BYTE bytCount,
                DWORD dwValue) ;
```

Visual Basic

```
Declare Function DO_Writes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, ByVal
nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. Slot numbers range from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be written.
dwValue	Stores the channel statuses of contiguous D/O channels; each bit holds one channel status. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the second digital output channel's status. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO_Read

This function code is used to read the output status of a specific D/O channel.

C/C++

```
int DO_Read( int hConnection,
              BYTE bytSlot,
              BYTE bytChannel,
              BYTE * bytValue);
```

Visual Basic

```
Declare Function DO_Read Lib "MXIO.dll" (ByVal hConnection
As Long, ByVal bytSlot As Byte, ByVal bytChannel As Byte,
bytValue As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. Slot numbers range from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be read.
bytValue	A pointer that stores the specific channel output value to be read.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO_Write

This function code is used to write the output status for a specific D/O channel.

C/C++

```
int DO_Write( int hConnection,
              BYTE bytSlot,
              BYTE bytChannel,
              BYTE bytValue);
```

Visual Basic

```
Declare Function DO_Write Lib "MXIO.dll" (ByVal hConnection
As Long, ByVal bytSlot As Byte, ByVal bytChannel As Byte,
ByVal bytValue As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. Slot numbers range from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be write.
bytValue	Stores the specific channel's output status that is to be written. 1 represents ON, 0 represents OFF.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO_GetSafeValues

This function code is used to get output safe values of contiguous D/O channels.

C/C++

```
int DO_GetSafeValues( int hConnection,
                      BYTE bytSlot,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD * dwValue);
```

Visual Basic

```
Declare Function DO_GetSafeValues Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, nValue As
Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. Slot numbers range from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwValue	A pointer that stores the safe value of contiguous D/O channels; each WORD holds one channel value. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO_SetSafeValues

This function code is used to set safe values of contiguous D/O channels.

C/C++

```
int DO_SetSafeValues( int hConnection,
                      BYTE bytSlot,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD dwValue);
```

Visual Basic

```
Declare Function DO_SetSafeValues Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, ByVal
nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. Slot numbers range from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	A pointer that stores the safe value of contiguous D/O channels; each WORD holds one channel value. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the status of the second digital output channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO_GetSafeValue

This function code is used to get the safe value for a specific D/O channel.

C/C++

```
int DO_GetSafeValue( int Connection ,
                     BYTE bytSlot,
                     BYTE bytChannel,
                     BYTE bytValue);
```

Visual Basic

```
Declare Function DO_GetSafeValue Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, bytValue As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. Slot numbers range from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be get.
bytValue	Stores the safe value of specific D/O channel. 0: represents OFF 1: represents ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO_SetSafeValue

This function code is used to set the safe value for a specific D/O channel.

C/C++

```
int DO_SetSafeValue( int Connection ,
                     BYTE bytSlot,
                     BYTE bytChannel,
                     BYTE bytValue);
```

Visual Basic

```
Declare Function DO_SetSafeValue Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, ByVal bytValue As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. Slot numbers range from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel whose output will be set.
bytValue	Stores the safe value of specific D/O channel. 0: represents OFF 1: represents ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO_GetSafeValues_W

This function code is used to get output safe values of contiguous D/O channels.

C/C++

```
int DO_GetSafeValues_W( int hConnection,
                      BYTE bytSlot,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD * wValue);
```

Visual Basic

```
Declare Function DO_GetSafeValues_W Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. Slot numbers range from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wValue	A pointer that stores the safe value of contiguous D/O channels; each word holds one channel value. A word value of 0 represents the digital output status of the start channel. A word value of 1 represents the status of the second digital output channel. The values of the unused bits are random. 0: OFF 1: ON 2: Hold Last

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO_SetSafeValues_W

This function code is used to set safe values of contiguous D/O channels.

C/C++

```
int DO_SetSafeValues_W( int hConnection,
                        BYTE bytSlot,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD * wValue);
```

Visual Basic

```
Declare Function DO_SetSafeValues_W Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. Slot numbers range from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	A pointer that stores the safe value of contiguous D/O channels; each word holds one channel value. A word value of 0 represents the digital output status of the start channel. A word value of 1 represents the status of the second digital output channel. 0: OFF 1: ON 2: Hold Last

Return Value:

Succeed	MXIO_OK
---------	---------

Fail

Refer to Return Codes.

DO2K_GetModes

This function code is used to get the mode of contiguous D/O channels.

C/C++

```
int DO2K_GetModes( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    WORD wMode[ ] );
```

Visual Basic

```
Declare Function DO2K_GetModes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wMode	An array that stores the mode of contiguous D/O channels. The values are: 0: D/O mode 1: Pulse mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO2K_SetModes

This function code is used to set the mode of contiguous D/O channels.

C/C++

```
int DO2K_SetModes( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    WORD wMode[ ] );
```

Visual Basic

```
Declare Function DO2K_SetModes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wMode	An array that stores the mode of contiguous D/O channels. The values are: 0: D/O mode 1: Pulse mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO2K_GetMode

This function code is used to get the mode for a specific D/O channel.

C/C++

```
int DO2K_GetMode( int hConnection,
                   BYTE bytChannel,
                   WORD * wMode);
```

Visual Basic

```
Declare Function DO2K_GetMode Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iMode As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
wMode	A pointer that stores the mode of specific D/O channel. The values are: 0: D/O mode 1: Pulse mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO2K_SetMode

This function code is used to set the mode for a specific channel.

C/C++

```
int DO2K_SetMode( int hConnection,
                   BYTE bytChannel,
                   WORD wMode );
```

Visual Basic

```
Declare Function DO2K_SetMode Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal wMode
As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
wMode	A pointer that stores the mode of specific D/O channel. The values are: 0: D/O mode 1: Pulse mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO2K_GetPowerOnValues

This function code is used to get the power on value of contiguous D/O channels.

C/C++

```
int DO2K_GetPowerOnValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD * dwValue);
```

Visual Basic

```
Declare Function DO2K_GetPowerOnValues Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwValue	A pointer that stores the power on value of contiguous D/O channels; each bit holds one channel value. A bit value of 0 represents the power on status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO2K_SetPowerOnValues

This function code is used to set the power on value of contiguous D/O channels.

C/C++

```
int DO2K_SetPowerOnValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwValue);
```

Visual Basic

```
Declare Function DO2K_SetPowerOnValues Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
     ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	Stores the power on value of contiguous D/O channels; each bit holds one channel value. A bit value of 0 represents the power on status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO2K_GetPowerOnValue

This function code is used to get the power on value for a specific channel.

C/C++

```
int DO2K_GetPowerOnValue( int hConnection,
                           BYTE bytChannel,
                           BYTE * bytValue);
```

Visual Basic

```
Declare Function DO2K_GetPowerOnValue Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, bytValue As
Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
bytValue	A pointer that stores the power on value of specific D/O channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO2K_SetPowerOnValue

This function code is used to set the power on value for a specific channel.

C/C++

```
int DO2K_SetPowerOnValue( int hConnection,
                           BYTE bytChannel,
                           BYTE bytValue);
```

Visual Basic

```
Declare Function DO2K_SetPowerOnValue Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal
bytValue As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
bytValue	Sotres the power on value of specific D/O channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO2K_GetPowerOnSeqDelaytimes

This function code is used to get contiguous channel's DI/DO power off storage enable mode.

C/C++

```
int DO2K_GetPowerOnSeqDelaytimes ( int hConnection,
                                    BYTE bytStartChannel,
                                    BYTE bytCount,
                                    WORD * wValue );
```

Visual Basic

```
Declare Function DO2K_GetPowerOnSeqDelaytimes Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	A pointer that stores the contiguous channel's power on sequence delay time (Second) The values are : Range: 0 ~ 300

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO2K_SetPowerOnSeqDelaytimes

This function code is used to set contiguous channel's DI/DO mode power off storage enable mode.

C/C++

```
int DO2K_SetPowerOnSeqDelaytimes ( int hConnection,
                                    BYTE bytStartChannel,
                                    BYTE bytCount,
                                    WORD * wValue);
```

Visual Basic

```
Declare Function DO2K_SetPowerOnSeqDelaytimes Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	A pointer that stores the contiguous channel's power on sequence delay time (Second) The values are : Range0 ~ 300

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO4K_GetSafeActions

This function code is used to get the safe action of contiguous D/O channels.

C/C++

```
int DO4K_GetSafeActions( int hConnection,
                        BYTE bytSlot,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wAction[ ]);
```

Visual Basic

```
Declare Function DO4K_GetSafeActions Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iAction As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wAction	An array that stores the safe action of contiguous D/O channels. The values are: 0: Safe Value 1: Hold last state

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO4K_SetSafeActions

This function code is used to set the safe action of contiguous D/O channels.

C/C++

```
int DO4K_SetSafeActions( int hConnection,
                        BYTE bytSlot,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wAction[ ]);
```

Visual Basic

```
Declare Function DO4K_SetSafeActions Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iAction As
integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wAction	An array that stores the safe action of contiguous D/O channels. The values are: 0: Safe Value 1: Hold last state

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO4K_GetSafeAction

This function code is used to get the safe action for a specific channel.

C/C++

```
int DO4K_GetSafeAction( int hConnection,
                        BYTE bytSlot,
                        BYTE bytChannel,
                        WORD * wAction );
```

Visual Basic

```
Declare Function DO4K_GetSafeAction Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, iAction As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32.
bytChannel	The specific channel to be get.
wAction	A pointer that stores the safe action of contiguous D/O channels. The values are: 0: Safe Value 1: Hold last state

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DO4K_SetSafeAction

This function code is used to set the safe action for a specific channel.

C/C++

```
int DO4K_SetSafeAction( int hConnection,
                        BYTE bytSlot,
                        BYTE bytChannel,
                        WORD wAction);
```

Visual Basic

```
Declare Function DO4K_SetSafeAction Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, ByVal iAction As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32.
bytChannel	The specific channel to be set.
wAction	A pointer that stores the safe action of contiguous D/O channels. The values are: 0: Safe Value 1: Hold last state

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_DO_GetSafeActions

This function code is used to get the safe action of contiguous D/O channels.

C/C++

```
int E42_DO_GetSafeActions( int hConnection,
                           BYTE bytSlot,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwAction[ ]);
```

Visual Basic

```
Declare Function E42_DO_GetSafeActions Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, nAction As
Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwAction	A pointer that stores the contiguous D/O channel#??#s afe action values; each bit holds one channel value. A bit value of 0 represents the digital input status of the start channel. A bit value of 1 represents the second digital input channel#??#s status. The values of the unused bits are random. 0: Fault Value 1: Hold last value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_DO_SetSafeActions

This function code is used to set the safe action of contiguous D/O channels.

C/C++

```
int E42_DO_SetSafeActions( int hConnection,
                           BYTE bytSlot,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwAction[ ] );
```

Visual Basic

```
Declare Function E42_DO_SetSafeActions Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, nAction As
Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwAction	A pointer that stores the contiguous D/O channel#??#s afe action values; each bit holds one channel value. A bit value of 0 represents the digital input status of the start channel. A bit value of 1 represents the second digital input channel#??#s status. The values of the unused bits are random. 0: Fault Value 1: Hold last value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_DO_GetPowerOnValues

This function code is used to get the power on value of contiguous D/O channels.

C/C++

```
int E42_DO_GetPowerOnValues( int hConnection,
                            BYTE bytSlot,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD * dwValue);
```

Visual Basic

```
Declare Function E42_DO_GetPowerOnValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, nValue As
Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwValue	A pointer that stores the power on value of contiguous D/O channels; each bit holds one channel value. A bit value of 0 represents the power on status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_DO_SetPowerOnValues

This function code is used to set the power on value of contiguous D/O channels.

C/C++

```
int E42_DO_SetPowerOnValues( int hConnection,
                            BYTE bytSlot,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD dwValue);
```

Visual Basic

```
Declare Function E42_DO_SetPowerOnValues Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytSlot As Byte, ByVal
     bytStartChannel As Byte, ByVal bytCount As Byte, ByVal
     nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	Stores the power on value of contiguous D/O channels; each bit holds one channel value. A bit value of 0 represents the power on status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_DO_Reads

This function code is used to read the output statuses of contiguous D/O channels.

C/C++

```
int E42_DO_Reads( int hConnection,
                   BYTE bytSlot,
                   BYTE bytStartChannel,
                   BYTE bytCount,
                   DWORD * dwValue) ;
```

Visual Basic

```
Declare Function E42_DO_Reads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, nValue As
Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. Slot numbers range from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	A pointer that stores the contiguous D/O channel???'s status; each bit holds one channel value. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_DO_Writes

This function code is used to write the output statuses of contiguous D/O channels.

C/C++

```
int E42_DO_Writes( int hConnection,
                    BYTE bytSlot,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    DWORD dwValue );
```

Visual Basic

```
Declare Function E42_DO_Writes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, ByVal
nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. Slot numbers range from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be written.
dwValue	Stores the channel statuses of contiguous D/O channels; each bit holds one channel status. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the second digital output channel's status. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_DO_GetFaultValues

This function code is used to get output safe values of contiguous D/O channels.

C/C++

```
int E42_DO_GetFaultValues( int hConnection,
                           BYTE bytSlot,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD * dwValue);
```

Visual Basic

```
Declare Function E42_DO_GetFaultValues Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, nValue As
Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. Slot numbers range from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwValue	A pointer that stores the safe value of contiguous D/O channels; each WORD holds one channel value. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_DO_SetFaultValues

This function code is used to set safe values of contiguous D/O channels.

C/C++

```
int E42_DO_SetFaultValues( int hConnection,
                           BYTE bytSlot,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwValue );
```

Visual Basic

```
Declare Function E42_DO_SetFaultValues Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, ByVal
nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. Slot numbers range from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	A pointer that stores the safe value of contiguous D/O channels; each WORD holds one channel value. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the status of the second digital output channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_GetSignalWidths

This function code is used to get the contiguous pulse channel's Hi/Lo signal width.

C/C++

```
int Pulse2K_GetSignalWidths( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            WORD wHiWidth[ ],
                            WORD wLoWidth[ ]);
```

Visual Basic

```
Declare Function Pulse2K_GetSignalWidths Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iHiWidth As Integer, iLoWidth As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wHiWidth	An array that stores the contiguous pulse channel's Hi signal width, dwHiWidth[0] represents the Hi signal width of the starting channel. Refer to this table: <u>wHiWidth And wLoWidth Renaming Table</u>
wLoWidth	An array that stores the contiguous pulse channel's Lo signal width, dwLoWidth[0] represents the Lo signal width of the starting channel. Refer to this table: <u>wHiWidth And wLoWidth Renaming Table</u>

Return Value:

Succeed MXIO_OK
Fail Refer to Return Codes.

Pulse2K_SetSignalWidths

This function code is used to set the contiguous pulse channel's Hi/Lo signal width.

C/C++

```
int Pulse2K_SetSignalWidths( int hConnection,
                             BYTE bytStartChannel,
                             BYTE bytCount,
                             WORD wHiWidth[ ],
                             WORD wLoWidth[ ] );
```

Visual Basic

```
Declare Function Pulse2K_SetSignalWidths Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, iHiWidth As Integer, iLoWidth As
    Integer) As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytStartChannel Specifies the starting channel.

bytCount The number of channels to be set.

wHiWidth An array that stores the contiguous pulse channel's Hi signal width,
dwHiWidth[0] represents the Hi signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

wLoWidth An array that stores the contiguous pulse channel's Lo signal width,
dwLoWidth[0] represents the Lo signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

Return Value:

Succeed MXIO_OK
Fail Refer to Return Codes.

Pulse2K_GetSignalWidth

This function code is used to get the Hi/Lo signal width for a specific pulse channel.

C/C++

```
int Pulse2K_GetSignalWidth( int hConnection,
                            BYTE bytChannel,
                            WORD * wHiWidth,
                            WORD * wLoWidth);
```

Visual Basic

```
Declare Function Pulse2K_GetSignalWidth Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytChannel As Byte,
iHiWidth As Integer, iLoWidth As Integer) As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytChannel The specific channel to be get.

wHiWidth A pointer that stores the specific pulse channel's Hi signal width to be get.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

wLoWidth A pointer that stores the specific pulse channel's Lo signal width to be get.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

Return Value:

Succeed **MXIO_OK**

Fail Refer to Return Codes.

Pulse2K_SetSignalWidth

This function code is used to set the Hi/Lo signal width for a specific pulse channel.

C/C++

```
int Pulse2K_SetSignalWidth( int hConnection,  
                           BYTE bytChannel,  
                           WORD wHiWidth,  
                           WORD wLoWidth);
```

Visual Basic

```
Declare Function Pulse2K_SetSignalWidth Lib "MXIO.dll"  
    (ByVal hConnection As Long, ByVal bytChannel As Byte, ByVal  
    iHiWidth As Integer, ByVal iLoWidth As Integer) As Long
```

Arguments :

hConnection The handle for an I/O device connection.

`bytChannel` The specific channel to be set.

wHiWidth A pointer that stores the specific pulse channel's Hi signal width to be set.

Refer to this table:

wHiWidth And wLoWidth Renaming Table

wLoWidth A pointer that stores the specific pulse channel's Lo signal width to be set.

Refer to this table:

wHiWidth And wLoWidth Renaming Table

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

Pulse2K_GetSignalWidths32

This function code is used to get the contiguous pulse channel's Hi/Lo signal width(32 bits).

The function code is special designed for ioLogik E2260 only.

C/C++

```
int Pulse2K_GetSignalWidths32( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD dwHiWidth[ ],
                                DWORD dwLoWidth[ ]);
```

Visual Basic

```
Declare Function Pulse2K_GetSignalWidths32 Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iHiWidth As Long, iLoWidth As Long)
As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytStartChannel Specifies the starting channel.

bytCount The number of channels to be gets.

dwHiWidth An array that stores the contiguous pulse channel's Hi signal width,
dwHiWidth[0] represents the Hi signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

dwLoWidth An array that stores the contiguous pulse channel's Lo signal width,
dwLoWidth[0] represents the Lo signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_SetSignalWidths32

This function code is used to set the contiguous pulse channel's Hi/Lo signal width(32 bits).

The function code is specially designed for iologik E2260 only.

C/C++

```
int Pulse2K_SetSignalWidths32( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD dwHiWidth[ ],
                               DWORD dwLoWidth[ ]);
```

Visual Basic

```
Declare Function Pulse2K_SetSignalWidths32 Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iHiWidth As Long, iLoWidth As Long)
As Long
```

Arguments :

hConnection The handle for an I/O device connection.

`bytStartChannel` Specifies the starting channel.

bytCount The number of channels to be set.

dwHiWidth An array that stores the contiguous pulse channel's Hi signal width, dwHiWidth[0] represents the Hi signal width of the starting channel.

Refer to this table:

wHiWidth And wLowWidth Renaming Table

`dwLoWidth` An array that stores the contiguous pulse channel's Lo signal width, `dwLoWidth[0]` represents the Lo signal width of the starting channel.

Refer to this table:

wHiWidth And wLoWidth Renaming Table

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_GetSignalWidth32

This function code is used to get the Hi/Lo signal width(32 bits) for a specific pulse channel.

The function code is special designed for ioLogik E2260 only.

C/C++

```
int Pulse2K_GetSignalWidth32( int hConnection,
                               BYTE bytChannel,
                               DWORD * dwHiWidth,
                               DWORD * dwLoWidth);
```

Visual Basic

```
Declare Function Pulse2K_GetSignalWidth32 Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytChannel As Byte,
iHiWidth As Long, iLoWidth As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
dwHiWidth	A pointer that stores the specific pulse channel's Hi signal width to be get. Refer to this table: wHiWidth And wLoWidth Renaming Table
dwLoWidth	A pointer that stores the specific pulse channel's Lo signal width to be get. Refer to this table: wHiWidth And wLoWidth Renaming Table

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_SetSignalWidth32

This function code is used to set the Hi/Lo signal width(32 bits) for a specific pulse channel.

The function code is special designed for ioLogik E2260 only.

C/C++

```
int Pulse2K_SetSignalWidth32( int hConnection,
                               BYTE bytChannel,
                               DWORD dwHiWidth,
                               DWORD dwLoWidth);
```

Visual Basic

```
Declare Function Pulse2K_SetSignalWidth32 Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytChannel As Byte, ByVal
iHiWidth As Long, ByVal iLoWidth As Long) As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytChannel The specific channel to be set.

dwHiWidth Store the specific pulse channel's Hi signal width to be set.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

dwLoWidth Store the specific pulse channel's Lo signal width to be set.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

Return Value:

Succeed **MXIO_OK**

Fail Refer to Return Codes.

Pulse2K_GetOutputCounts

This function code is used to get the contiguous pulse channel's output count.

C/C++

```
int Pulse2K_GetOutputCounts( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD dwOutputCounts[ ] );
```

Visual Basic

```
Declare Function Pulse2K_GetOutputCounts Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nOutputCounts As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwOutputCounts	An array that stores the contiguous pulse channel's output count, dwOutputCounts[0] represents the pulse output count of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_SetOutputCounts

This function code is used to set the contiguous pulse channel's output count.

C/C++

```
int Pulse2K_SetOutputCounts( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD dwOutputCounts[ ] );
```

Visual Basic

```
Declare Function Pulse2K_SetOutputCounts Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nOutputCounts As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwOutputCounts	An array that stores the contiguous pulse channel's output count, dwOutputCounts[0] represents the pulse output count of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_GetOutputCount

This function code is used to get the output count for a specific pulse channel.

C/C++

```
int Pulse2K_GetOutputCount( int hConnection,
                            BYTE bytChannel,
                            DWORD * dwOutputCount);
```

Visual Basic

```
Declare Function Pulse2K_GetOutputCount Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytChannel As Byte,
     nOutputCount As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
dwOutputCounts	A pointer that stores the specific pulse channel's output count.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_SetOutputCount

This function code is used to set the output count for a specific pulse channel.

C/C++

```
int Pulse2K_SetOutputCount( int hConnection,
                            BYTE bytChannel,
                            DWORD dwOutputCount );
```

Visual Basic

```
Declare Function Pulse2K_SetOutputCount Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytChannel As Byte, ByVal
dwOutputCount As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
dwOutputCounts	A pointer that stores the specific pulse channel's output count.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_GetStartStatuses

This function code is used to get the contiguous pulse channel's start status.

C/C++

```
int Pulse2K_GetStartStatuses( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD * dwStatus);
```

Visual Basic

```
Declare Function Pulse2K_GetStartStatuses Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwStatus	An point that stores the contiguous pulse channel's start status, each bit holds one channel value. A bit value of 0 represents the start status of the start channel. A bit value of 1 represents the status of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_SetStartStatuses

This function code is used to set the contiguous pulse channel's start status.

C/C++

```
int Pulse2K_SetStartStatuses( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD dwStatus) ;
```

Visual Basic

```
Declare Function Pulse2K_SetStartStatuses Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, ByVal nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	An point that stores the contiguous pulse channel's start status, each bit holds one channel value. A bit value of 0 represents the start status of the start channel. A bit value of 1 represents the status of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_GetStartStatus

This function code is used to get the start status for a specific pulse channel.

C/C++

```
int Pulse2K_GetStartStatus( int hConnection,
                            BYTE bytChannel,
                            BYTE * bytStatus);
```

Visual Basic

```
Declare Function Pulse2K_GetStartStatus Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytChannel As Byte,
     bytStatus As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
bytStatus	A pointer that stores the specific pulse channel's start status. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_SetStartStatus

This function code is used to set the start status for a specific pulse channel.

C/C++

```
int Pulse2K_SetStartStatus( int hConnection,
                            BYTE bytChannel,
                            BYTE bytStatus);
```

Visual Basic

```
Declare Function Pulse2K_SetStartStatus Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytChannel As Byte, ByVal
     bytStatus As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
bytStatus	A pointer that stores the specific pulse channel's start status. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_GetPowerOnValues

This function code is used to get the contiguous pulse channel's power on value.

C/C++

```
int Pulse2K_GetPowerOnValues( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD * dwValue);
```

Visual Basic

```
Declare Function Pulse2K_GetPowerOnValues Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
bytStatus	An point that stores the contiguous pulse channel's power on value, each bit holds one channel value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_SetPowerOnValues

This function code is used to set the contiguous pulse channel's power on value.

C/C++

```
int Pulse2K_SetPowerOnValues(int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwValue);
```

Visual Basic

```
Declare Function Pulse2K_SetPowerOnValues Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	Stores the contiguous pulse channel's power on value, each bit holds one channel value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_GetPowerOnValue

This function code is used to get the power on value for a specific pulse channel.

C/C++

```
int Pulse2K_GetPowerOnValue( int hConnection,
                             BYTE bytChannel,
                             BYTE * bytValue);
```

Visual Basic

```
Declare Function Pulse2K_GetPowerOnValue Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytChannel As Byte,
bytValue As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
bytValue	A pointer that stores the specific pulse channel's power on value. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_SetPowerOnValue

This function code is used to set the power on value for a specific pulse channel.

C/C++

```
int Pulse2K_SetPowerOnValue( int hConnection,
                             BYTE bytChannel,
                             BYTE bytValue);
```

Visual Basic

```
Declare Function Pulse2K_SetPowerOnValue Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytChannel As Byte, ByVal
bytValue As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
bytValue	A pointer that stores the specific pulse channel's power on value. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_GetSafeValues

This function code is used to get the contiguous pulse channel's safe value.

C/C++

```
int Pulse2K_GetSafeValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD * dwValue);
```

Visual Basic

```
Declare Function Pulse2K_GetSafeValues Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwValue	An point that stores the contiguous pulse channel's safe value, each bit holds one channel value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_SetSafeValues

This function code is used to set the contiguous pulse channel's safe value.

C/C++

```
int Pulse2K_SetSafeValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwValue) ;
```

Visual Basic

```
Declare Function Pulse2K_SetSafeValues Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	An point that stores the contiguous pulse channel's safe value, each bit holds one channel value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_GetSafeValue

This function code is used to get the safe value for a specific pulse channel.

C/C++

```
int Pulse2K_GetSafeValue( int hConnection,
                           BYTE bytChannel,
                           BYTE * bytValue);
```

Visual Basic

```
Declare Function Pulse2K_GetSafeValue Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, bytValue As
Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
bytValue	A pointer that stores the specific pulse channel's power on value. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Pulse2K_SetSafeValue

This function code is used to set the safe value for a specific pulse channel.

C/C++

```
int Pulse2K_SetSafeValue( int hConnection,
                           BYTE bytChannel,
                           BYTE bytValue );
```

Visual Basic

```
Declare Function Pulse2K_SetSafeValue Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal
bytValue As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
bytValue	A pointer that stores the specific pulse channel's power on value. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DIO2K_GetIOMode

This function code is used to get specific channel's DI/DO mode.

C/C++

```
int DIO2K_GetIOMode ( int hConnection,
                      BYTE bytChannel,
                      BYTE * bytMode);
```

Visual Basic

```
Declare Function DIO2K_GetIOMode Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, bytMode As
Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
bytMode	A pointer that stores the specific channel's DI/DO mode. The values are : 0 INPUT DI mode 1 OUTPUT DO mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DIO2K_SetIOMode

This function code is used to set specific channel's DI/DO mode. This function will take effect after restart the device.

C/C++

```
int DIO2K_SetIOMode ( int hConnection,
                      BYTE bytChannel,
                      BYTE bytMode );
```

Visual Basic

```
Declare Function DIO2K_SetIOMode Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal
bytMode As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
bytMode	A pointer that stores the specific channel's DI/DO mode. The values are : 0: INPUT DI mode 1: OUTPUT DO mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DIO2K_GetIOModes

This function code is used to get contiguous channel's DI/DO mode.

C/C++

```
int DIO2K_GetIOModes ( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD * dwMode);
```

Visual Basic

```
Declare Function DIO2K_GetIOModes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nMode As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous channel's DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: INPUT DI mode 1: OUTPUT DO mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

DIO2K_SetIOModes

This function code is used to set contiguous channel's DI/DO mode. This function will take effect after restart the device.

C/C++

```
int DIO2K_SetIOModes ( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD dwMode );
```

Visual Basic

```
Declare Function DIO2K_SetIOModes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, ByVal nMode As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwMode	A pointer that stores the contiguous channel's DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: INPUT DI mode 1: OUTPUT DO mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DO_Reads

This function code is used to read the output statuses of contiguous D/O channels.

C/C++

```
int W5K_DO_Reads( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    DWORD * dwValue) ;
```

Visual Basic

```
Declare Function W5K_DO_Reads Lib MXIO.dll (ByVal
hConnection As Long,
ByVal bytStartChannel As Byte, ByVal bytCount As Byte,
nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	A pointer that stores the contiguous D/O channel???s status; each bit holds one channel value. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DO_Writes

This function code is used to write the output statuses of contiguous D/O channels.

C/C++

```
int W5K_DO_Writes( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     DWORD dwValue);
```

Visual Basic

```
Declare Function W5K_DO_Writes Lib MXIO.dll (ByVal
hConnection As Long,
ByVal bytStartChannel As Byte, ByVal bytCount As Byte,
ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be written.
dwValue	Stores the channel statuses of contiguous D/O channels; each bit holds one channel status. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the second digital output channel???'s status. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DO_GetSafeValues

This function code is used to get output safe values of contiguous D/O channels.

C/C++

```
int W5K_DO_GetSafeValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD * dwValue);
```

Visual Basic

```
Declare Function W5K_DO_GetSafeValues Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwValue	A pointer that stores the safe value of contiguous D/O channels; each WORD holds one channel value. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DO_SetSafeValues

This function code is used to set safe values of contiguous D/O channels.

C/C++

```
int W5K_DO_SetSafeValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwValue) ;
```

Visual Basic

```
Declare Function W5K_DO_SetSafeValues Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	A pointer that stores the safe value of contiguous D/O channels; each WORD holds one channel value. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the status of the second digital output channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DO_GetModes

This function code is used to get the mode of contiguous D/O channels.

C/C++

```
int W5K_DO_GetModes( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wMode[ ] );
```

Visual Basic

```
Declare Function W5K_DO_GetModes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wMode	An array that stores the mode of contiguous D/O channels. The values are: 0: D/O mode 1: Pulse mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DO_SetModes

This function code is used to set the mode of contiguous D/O channels.

C/C++

```
int W5K_DO_SetModes( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wMode[ ] );
```

Visual Basic

```
Declare Function W5K_DO_SetModes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wMode	An array that stores the mode of contiguous D/O channels. The values are: 0: D/O mode 1: Pulse mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DO_GetPowerOnValues

This function code is used to get the power on value of contiguous D/O channels.

C/C++

```
int W5K_DO_GetPowerOnValues( int hConnection,
                             BYTE bytStartChannel,
                             BYTE bytCount,
                             DWORD * dwValue);
```

Visual Basic

```
Declare Function W5K_DO_GetPowerOnValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwValue	A pointer that stores the power on value of contiguous D/O channels; each bit holds one channel value. A bit value of 0 represents the power on status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DO_SetPowerOnValues

This function code is used to set the power on value of contiguous D/O channels.

C/C++

```
int W5K_DO_SetPowerOnValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwValue);
```

Visual Basic

```
Declare Function W5K_DO_SetPowerOnValues Lib MXIO.dll
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	Stores the power on value of contiguous D/O channels; each bit holds one channel value. A bit value of 0 represents the power on status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Pulse_GetSignalWidths32

This function code is used to get the contiguous pulse channel~~???s~~ Hi/Lo signal width(32 bits).

C/C++

```
int W5K_Pulse_GetSignalWidths32( int hConnection,
                                  BYTE bytStartChannel,
                                  BYTE bytCount,
                                  DWORD dwHiWidth[ ],
                                  DWORD dwLoWidth[ ]);
```

Visual Basic

```
Declare Function W5K_Pulse_GetSignalWidths32 Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iHiWidth As Long, iLoWidth As Long)
As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytStartChannel Specifies the starting channel.

bytCount The number of channels to be gets.

dwHiWidth An array that stores the contiguous pulse channel~~???s~~ Hi signal width, dwHiWidth[0] represents the Hi signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

dwLoWidth An array that stores the contiguous pulse channel~~???s~~ Lo signal width, dwLoWidth[0] represents the Lo signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Pulse_SetSignalWidths32

This function code is used to set the contiguous pulse channel~~?????~~'s Hi/Lo signal width(32 bits).

C/C++

```
int W5K_Pulse_SetSignalWidths32( int hConnection,
                                  BYTE bytStartChannel,
                                  BYTE bytCount,
                                  DWORD dwHiWidth[ ],
                                  DWORD dwLoWidth[ ] );
```

Visual Basic

```
Declare Function W5K_Pulse_SetSignalWidths32 Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iHiWidth As Long, iLoWidth As Long)
As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytStartChannel Specifies the starting channel.

bytCount The number of channels to be set.

dwHiWidth An array that stores the contiguous pulse channel~~?????~~'s Hi signal width, dwHiWidth[0] represents the Hi signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

dwLoWidth An array that stores the contiguous pulse channel~~?????~~'s Lo signal width, dwLoWidth[0] represents the Lo signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

Return Value:

Succeed MXIO_OK
Fail Refer to Return Codes.

W5K_Pulse_GetOutputCounts

This function code is used to get the contiguous pulse channel~~?????~~s output count.

C/C++

```
int W5K_Pulse_GetOutputCounts( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD dwOutputCounts[ ] );
```

Visual Basic

```
Declare Function W5K_Pulse_GetOutputCounts Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nOutputCounts As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwOutputCounts	An array that stores the contiguous pulse channel ????? s output count, dwOutputCounts[0] represents the pulse output count of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Pulse_SetOutputCounts

This function code is used to set the contiguous pulse channel~~???s~~'s output count.

C/C++

```
int W5K_Pulse_SetOutputCounts( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD dwOutputCounts[ 1 ] );
```

Visual Basic

```
Declare Function W5K_Pulse_SetOutputCounts Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nOutputCounts As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwOutputCounts	An array that stores the contiguous pulse channel ???s 's output count, dwOutputCounts[0] represents the pulse output count of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Pulse_GetStartStatuses

This function code is used to get the contiguous pulse channel?♦?♦?♦'s start status.

C/C++

```
int W5K_Pulse_GetStartStatuses( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD * dwStatus);
```

Visual Basic

```
Declare Function W5K_Pulse_GetStartStatuses Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwStatus	An point that stores the contiguous pulse channel?♦?♦?♦'s start status, each bit holds one channel value. A bit value of 0 represents the start status of the start channel. A bit value of 1 represents the status of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Pulse_SetStartStatuses

This function code is used to set the contiguous pulse channel~~???s~~'s start status.

C/C++

```
int W5K_Pulse_SetStartStatuses( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD dwStatus) ;
```

Visual Basic

```
Declare Function W5K_Pulse_SetStartStatuses Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	An point that stores the contiguous pulse channel ???s 's start status, each bit holds one channel value. A bit value of 0 represents the start status of the start channel. A bit value of 1 represents the status of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Pulse_GetPowerOnValues

This function code is used to get the contiguous pulse channel~~?????~~'s power on value.

C/C++

```
int W5K_Pulse_GetPowerOnValues( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD * dwValue);
```

Visual Basic

```
Declare Function W5K_Pulse_GetPowerOnValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
bytStatus	An point that stores the contiguous pulse channel ????? 's power on value, each bit holds one channel value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Pulse_SetPowerOnValues

This function code is used to set the contiguous pulse channel~~◆◆◆~~s power on value.

C/C++

```
int W5K_Pulse_SetPowerOnValues(int hConnection,  
                                BYTE bytStartChannel,  
                                BYTE bytCount,  
                                DWORD dwValue);
```

Visual Basic

```
Declare Function W5K_Pulse_SetPowerOnValues Lib MXIO.dll  
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,  
ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	Stores the contiguous pulse channel ◆◆◆ s power on value, each bit holds one channel value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Pulse_GetSafeValues

This function code is used to get the contiguous pulse channel~~?????~~'s safe value.

C/C++

```
int W5K_Pulse_GetSafeValues( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD * dwValue);
```

Visual Basic

```
Declare Function W5K_Pulse_GetSafeValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwValue	An point that stores the contiguous pulse channel ????? 's safe value, each bit holds one channel value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Pulse_SetSafeValues

This function code is used to set the contiguous pulse channel~~?????~~'s safe value.

C/C++

```
int W5K_Pulse_SetSafeValues( int hConnection,
                             BYTE bytStartChannel,
                             BYTE bytCount,
                             DWORD dwValue);
```

Visual Basic

```
Declare Function W5K_Pulse_SetSafeValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	An point that stores the contiguous pulse channel ????? 's safe value, each bit holds one channel value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DIO_GetIOModes

This function code is used to get contiguous channel~~?????~~s DI/DO mode.

C/C++

```
int W5K_DIO_GetIOModes ( int hConnection,
                         BYTE bytStartChannel,
                         BYTE bytCount,
                         DWORD * dwMode );
```

Visual Basic

```
Declare Function W5K_DIO_GetIOModes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nMode As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous channel ????? s DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel ????? s status. The values are : 0: INPUT DI mode 1: OUTPUT DO mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DIO_SetIOModes

This function code is used to set contiguous channel~~???'s~~ DI/DO mode. This function will take effect after restart the device.

C/C++

```
int W5K_DIO_SetIOModes ( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwMode) ;
```

Visual Basic

```
Declare Function W5K_DIO_SetIOModes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, ByVal nMode As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwMode	A pointer that stores the contiguous channel ???'s DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel ???'s status. The values are : 0: INPUT DI mode 1: OUTPUT DO mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DO_Reads_Ex

This function code is used to read the output statuses of contiguous D/O channels.

C/C++

```
int W5K_DO_Reads_Ex( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD * dwValue,
                      BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_DO_Reads_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As Long, ByVal bytSlot As Byte) As
Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	A pointer that stores the contiguous D/O channel???'s status; each bit holds one channel value. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DO_Writes_Ex

This function code is used to write the output statuses of contiguous D/O channels.

C/C++

```
int W5K_DO_Writes_Ex( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD dwValue,
                      BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_DO_Writes_Ex Lib MXIO.dll (ByVal
hConnection As Long,
ByVal bytStartChannel As Byte, ByVal bytCount As Byte,
ByVal nValue As Long, ByVal bytSlot As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be written.
dwValue	Stores the channel statuses of contiguous D/O channels; each bit holds one channel status. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the second digital output channel's status. The values of the unused bits are random.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_DO_GetModes_Ex

This function code is used to get the mode of contiguous D/O channels.

C/C++

```
int W5K_DO_GetModes_Ex( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wMode[ ],
                        BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_DO_GetModes_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMode As Integer, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wMode	An array that stores the mode of contiguous D/O channels. The values are: 0: D/O mode 1: Pulse mode
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Pulse_GetStartStatuses_Ex

This function code is used to get the contiguous pulse channel~~?????~~s start status.

C/C++

```
int W5K_Pulse_GetStartStatuses_Ex( int hConnection,
                                    BYTE bytStartChannel,
                                    BYTE bytCount,
                                    DWORD * dwStatus,
                                    BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_Pulse_GetStartStatuses_Ex Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nStatus As Long, ByVal bytSlot As
Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwStatus	An point that stores the contiguous pulse channel ????? s start status, each bit holds one channel value. A bit value of 0 represents the start status of the start channel. A bit value of 1 represents the status of the second pulse channel. The values are: 0: stop 1: start
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed MXIO_OK

Fail

Refer to Return Codes.

W5K_Pulse_SetStartStatuses_Ex

This function code is used to set the contiguous pulse channel~~?????~~'s start status.

C/C++

```
int W5K_Pulse_SetStartStatuses_Ex( int hConnection,
                                    BYTE bytStartChannel,
                                    BYTE bytCount,
                                    DWORD dwStatus,
                                    BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_Pulse_SetStartStatuses_Ex Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nStatus As Long, ByVal
bytSlot As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	An point that stores the contiguous pulse channel ????? 's start status, each bit holds one channel value. A bit value of 0 represents the start status of the start channel. A bit value of 1 represents the status of the second pulse channel. The values are: 0: stop 1: start
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed MXIO_OK

Fail

Refer to Return Codes.

W5K_DIO_GetIOModes_Ex

This function code is used to get contiguous channel~~?????~~s DI/DO mode.

C/C++

```
int W5K_DIO_GetIOModes_Ex ( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD * dwMode,
                            BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_DIO_GetIOModes_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nMode As Long, ByVal bytSlot As Byte) As
Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous channel ????? s DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel ????? s status. The values are : 0: INPUT DI mode 1: OUTPUT DO mode
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
---------	---------

Fail

Refer to Return Codes.

E1K_DO_Reads

This function code is used to read the output statuses of contiguous D/O channels.

C/C++

```
int E1K_DO_Reads( int hConnection,
                   BYTE bytStartChannel,
                   BYTE bytCount,
                   DWORD * dwValue) ;
```

Visual Basic

```
Declare Function E1K_DO_Reads Lib MXIO.dll (ByVal
hConnection As Long,
ByVal bytStartChannel As Byte, ByVal bytCount As Byte,
nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	A pointer that stores the contiguous D/O channel???s status; each bit holds one channel value. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DO_Writes

This function code is used to write the output statuses of contiguous D/O channels.

C/C++

```
int E1K_DO_Writes( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    DWORD dwValue);
```

Visual Basic

```
Declare Function E1K_DO_Writes Lib MXIO.dll (ByVal
hConnection As Long,
ByVal bytStartChannel As Byte, ByVal bytCount As Byte,
ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be written.
dwValue	Stores the channel statuses of contiguous D/O channels; each bit holds one channel status. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the second digital output channel???'s status. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DO_GetSafeValues_W

This function code is used to get output safe values of contiguous D/O channels.

C/C++

```
int E1K_DO_GetSafeValues_W( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD * wValue);
```

Visual Basic

```
Declare Function E1K_DO_GetSafeValues_W Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wValue	A pointer that stores the safe value of contiguous D/O channels; each word holds one channel value. A word value of 0 represents the digital output status of the start channel. A word value of 1 represents the status of the second digital output channel. The values of the unused bits are random. 0: OFF 1: ON 2: Hold Last

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DO_SetSafeValues_W

This function code is used to set safe values of contiguous D/O channels.

C/C++

```
int E1K_DO_SetSafeValues_W( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            WORD * wValue);
```

Visual Basic

```
Declare Function E1K_DO_SetSafeValues_W Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, ByVal nValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	A pointer that stores the safe value of contiguous D/O channels; each word holds one channel value. A word value of 0 represents the digital output status of the start channel. A word value of 1 represents the status of the second digital output channel. The values of the unused bits are random. 0: OFF 1: ON 2: Hold Last

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DO_GetModes

This function code is used to get the mode of contiguous D/O channels.

C/C++

```
int E1K_DO_GetModes( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wMode[ ] );
```

Visual Basic

```
Declare Function E1K_DO_GetModes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wMode	An array that stores the mode of contiguous D/O channels. The values are: 0: D/O mode 1: Pulse mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DO_SetModes

This function code is used to set the mode of contiguous D/O channels.

C/C++

```
int E1K_DO_SetModes( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wMode[ ] );
```

Visual Basic

```
Declare Function E1K_DO_SetModes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wMode	An array that stores the mode of contiguous D/O channels. The values are: 0: D/O mode 1: Pulse mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DO_GetPowerOnValues

This function code is used to get the power on value of contiguous D/O channels.

C/C++

```
int E1K_DO_GetPowerOnValues( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD * dwValue);
```

Visual Basic

```
Declare Function E1K_DO_GetPowerOnValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwValue	A pointer that stores the power on value of contiguous D/O channels; each bit holds one channel value. A bit value of 0 represents the power on status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DO_SetPowerOnValues

This function code is used to set the power on value of contiguous D/O channels.

C/C++

```
int E1K_DO_SetPowerOnValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwValue);
```

Visual Basic

```
Declare Function E1K_DO_SetPowerOnValues Lib MXIO.dll
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	Stores the power on value of contiguous D/O channels; each bit holds one channel value. A bit value of 0 represents the power on status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DO_GetPowerOnSeqDelaytimes

This function code is used to get contiguous channel#???'s DI/DO power off storage enable mode.

C/C++

```
int E1K_DO_GetPowerOnSeqDelaytimes ( int hConnection,
                                     BYTE bytStartChannel,
                                     BYTE bytCount,
                                     WORD * wValue);
```

Visual Basic

```
Declare Function E1K_DO_GetPowerOnSeqDelaytimes Lib
MXIO.dll (ByVal hConnection As Long, ByVal bytStartChannel
As Byte, ByVal bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	A pointer that stores the contiguous channel#???'s power on sequence delay time (Second) The values are : Range: 0 ~ 300

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DO_SetPowerOnSeqDelaytimes

This function code is used to set contiguous channel#???'s DI/DO mode power off storage enable mode.

C/C++

```
int E1K_DO_SetPowerOnSeqDelaytimes ( int hConnection,
                                     BYTE bytStartChannel,
                                     BYTE bytCount,
                                     WORD * wValue);
```

Visual Basic

```
Declare Function E1K_DO_SetPowerOnSeqDelaytimes Lib
MXIO.dll (ByVal hConnection As Long, ByVal bytStartChannel
As Byte, ByVal bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	A pointer that stores the contiguous channel#???'s power on sequence delay time (Second) The values are : Range0 ~ 300

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Pulse_GetSignalWidths

This function code is used to get the contiguous pulse channel~~???'s~~ Hi/Lo signal width(32 bits).

C/C++

```
int E1K_Pulse_GetSignalWidths( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               WORD wHiWidth[ ],
                               WORD wLoWidth[ ]);
```

Visual Basic

```
Declare Function E1K_Pulse_GetSignalWidths Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iHiWidth As Integer, iLoWidth As
Integer) As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytStartChannel Specifies the starting channel.

bytCount The number of channels to be gets.

wHiWidth An array that stores the contiguous pulse channel~~???'s~~ Hi signal width, wHiWidth[0] represents the Hi signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

wLoWidth An array that stores the contiguous pulse channel~~???'s~~ Lo signal width, wLoWidth[0] represents the Lo signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Pulse_SetSignalWidths

This function code is used to set the contiguous pulse channel~~?????~~'s Hi/Lo signal width(32 bits).

C/C++

```
int E1K_Pulse_SetSignalWidths( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               WORD wHiWidth[ ],
                               WORD wLoWidth[ ] );
```

Visual Basic

```
Declare Function E1K_Pulse_SetSignalWidths Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iHiWidth As Integer, iLoWidth As
Integer) As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytStartChannel Specifies the starting channel.

bytCount The number of channels to be set.

wHiWidth An array that stores the contiguous pulse channel~~?????~~'s Hi signal width, wHiWidth[0] represents the Hi signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

wLoWidth An array that stores the contiguous pulse channel~~?????~~'s Lo signal width, wLoWidth[0] represents the Lo signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

Return Value:

Succeed MXIO_OK
Fail Refer to Return Codes.

E1K_Pulse_GetOutputCounts

This function code is used to get the contiguous pulse channel~~?????~~s output count.

C/C++

```
int E1K_Pulse_GetOutputCounts( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD dwOutputCounts[ ] );
```

Visual Basic

```
Declare Function E1K_Pulse_GetOutputCounts Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nOutputCounts As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwOutputCounts	An array that stores the contiguous pulse channel ????? s output count, dwOutputCounts[0] represents the pulse output count of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Pulse_SetOutputCounts

This function code is used to set the contiguous pulse channel~~???s~~'s output count.

C/C++

```
int E1K_Pulse_SetOutputCounts( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD dwOutputCounts[ 1 ] );
```

Visual Basic

```
Declare Function E1K_Pulse_SetOutputCounts Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nOutputCounts As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwOutputCounts	An array that stores the contiguous pulse channel ???s 's output count, dwOutputCounts[0] represents the pulse output count of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Pulse_GetStartStatuses

This function code is used to get the contiguous pulse channel?♦?♦?♦'s start status.

C/C++

```
int E1K_Pulse_GetStartStatuses( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD * dwStatus);
```

Visual Basic

```
Declare Function E1K_Pulse_GetStartStatuses Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwStatus	An point that stores the contiguous pulse channel?♦?♦?♦'s start status, each bit holds one channel value. A bit value of 0 represents the start status of the start channel. A bit value of 1 represents the status of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Pulse_SetStartStatuses

This function code is used to set the contiguous pulse channel~~???'s~~'s start status.

C/C++

```
int E1K_Pulse_SetStartStatuses( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD dwStatus) ;
```

Visual Basic

```
Declare Function E1K_Pulse_SetStartStatuses Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	An point that stores the contiguous pulse channel ???'s 's start status, each bit holds one channel value. A bit value of 0 represents the start status of the start channel. A bit value of 1 represents the status of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Pulse_GetPowerOnValues

This function code is used to get the contiguous pulse channel~~?????~~'s power on value.

C/C++

```
int E1K_Pulse_GetPowerOnValues( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD * dwValue);
```

Visual Basic

```
Declare Function E1K_Pulse_GetPowerOnValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
bytStatus	An point that stores the contiguous pulse channel ????? 's power on value, each bit holds one channel value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Pulse_SetPowerOnValues

This function code is used to set the contiguous pulse channel~~◆◆◆~~s power on value.

C/C++

```
int E1K_Pulse_SetPowerOnValues(int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD dwValue);
```

Visual Basic

```
Declare Function E1K_Pulse_SetPowerOnValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	Stores the contiguous pulse channel ◆◆◆ s power on value, each bit holds one channel value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Pulse_GetSafeValues

This function code is used to get the contiguous pulse channel~~?????~~'s safe value.

C/C++

```
int E1K_Pulse_GetSafeValues( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD * dwValue);
```

Visual Basic

```
Declare Function E1K_Pulse_GetSafeValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwValue	An point that stores the contiguous pulse channel ????? 's safe value, each bit holds one channel value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_Pulse_SetSafeValues

This function code is used to set the contiguous pulse channel~~?????~~'s safe value.

C/C++

```
int E1K_Pulse_SetSafeValues( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD dwValue);
```

Visual Basic

```
Declare Function E1K_Pulse_SetSafeValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	An point that stores the contiguous pulse channel ????? 's safe value, each bit holds one channel value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DIO_GetIOModes

This function code is used to get contiguous channel~~?????~~s DI/DO mode.

C/C++

```
int E1K_DIO_GetIOModes ( int hConnection,
                         BYTE bytStartChannel,
                         BYTE bytCount,
                         DWORD * dwMode );
```

Visual Basic

```
Declare Function E1K_DIO_GetIOModes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nMode As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous channel ????? s DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel ????? s status. The values are : 0: INPUT DI mode 1: OUTPUT DO mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_DIO_SetIOModes

This function is used to set contiguous channel#?#?#s DI/DO mode and will take effect after the device is restarted.

C/C++

```
int E1K_DIO_SetIOModes ( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwMode) ;
```

Visual Basic

```
Declare Function E1K_DIO_SetIOModes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, ByVal nMode As Long) As Long
```

Arguments:

hConnection	A handle of a connection.
bytStartChannel	A handle of a connection.
bytCount	The number of channels to be set.
dwMode	A pointer that stores the contiguous channel#?#?#s DI/DO mode; each bit holds for the mode of one channel. 0: DI mode 1: DO mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI_Reads

This function code is used to read contiguous channel's analog input value.

C/C++

```
int AI_Reads( int hConnection,
               BYTE bytSlot,
               BYTE bytStartChannel,
               BYTE bytCount,
               double dValue[ ]);
```

Visual Basic

```
Declare Function AI_Reads Lib "MXIO.dll" (ByVal hConnection
As Long, ByVal bytSlot As Byte, ByVal bytStartChannel As
Byte, ByVal bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel's value, dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI_Read

This function code is used to read the analog input value for a specific channel.

C/C++

```
int AI_Read( int hConnection,
              BYTE bytSlot,
              BYTE bytChannel,
              double * dValue);
```

Visual Basic

```
Declare Function AI_Read Lib "MXIO.dll" (ByVal hConnection
As Long, ByVal bytSlot As Byte, ByVal bytChannel As Byte,
dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be read.
dValue	A pointer that stores the specific channel's analog input value to be read. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI_ReadRaw

This function code is used to read contiguous channel's analog input raw data.

C/C++

```
int AI_ReadRaw( int hConnection,
                 BYTE bytSlot,
                 BYTE bytStartChannel,
                 BYTE bytCount,
                 WORD wValue[ ] );
```

Visual Basic

```
Declare Function AI_ReadRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel's raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI_ReadRaw

This function code is used to read the analog input raw data for a specific channel.

C/C++

```
nt AI_ReadRaw( int hConnection,
                BYTE bytSlot,
                BYTE bytChannel,
                WORD * wValue);
```

Visual Basic

```
Declare Function AI_ReadRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be read.
wValue	A pointer that stores the specific channel's analog input raw data to be read.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_ReadMins

This function code is used to read contiguous A/I channel's minimize value.

C/C++

```
int AI2K_ReadMins( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     double dValue[ ] );
```

Visual Basic

```
Declare Function AI2K_ReadMins Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel's value, dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_ReadMinRaws

This function code is used to read contiguous A/I channel's minimize raw data.

C/C++

```
int AI2K_ReadMinRaws( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ] );
```

Visual Basic

```
Declare Function AI2K_ReadMinRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel's minimize raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_ResetMins

This function code is used to reset contiguous A/I channel's minimize value.

C/C++

```
int AI2K_ResetMins( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount) ;
```

Visual Basic

```
Declare Function AI2K_ResetMins Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_ReadMin

This function code is used to read the analog input minimize value for a specific channel.

C/C++

```
int AI2K_ReadMin( int hConnection,
                    BYTE bytChannel,
                    double * dValue);
```

Visual Basic

```
Declare Function AI2K_ReadMin Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, dValue As
Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be read.
dValue	A pointer that stores the specific channel's analog input minimize value to be read. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_ReadMinRaw

This function code is used to read the analog input minimize raw data for a specific channel.

C/C++

```
int AI2K_ReadMinRaw( int hConnection,
                      BYTE bytChannel,
                      WORD * wValue);
```

Visual Basic

```
Declare Function AI2K_ReadMinRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, nValue As
Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be read.
wValue	An point that stores the specific A/I channel's minimize raw data.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_ResetMin

This function code is used to reset the analog input minimize value for a specific channel.

C/C++

```
int AI2K_ResetMin( int hConnection,
                     BYTE bytChannel);
```

Visual Basic

```
Declare Function AI2K_ResetMin Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_ReadMaxs

This function code is used to read contiguous A/I channel's maximize value.

C/C++

```
int AI2K_ReadMaxs( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     double dValue[ ] );
```

Visual Basic

```
Declare Function AI2K_ReadMaxs Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel's maximize value , dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_ReadMaxRaws

This function code is used to read contiguous A/I channel's maximize raw data.

C/C++

```
int AI2K_ReadMaxRaws( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ] );
```

Visual Basic

```
Declare Function AI2K_ReadMaxRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel's maximize raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_ResetMaxs

This function code is used to reset contiguous A/I channel's maximize value.

C/C++

```
int AI2K_ResetMaxs( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount) ;
```

Visual Basic

```
Declare Function AI2K_ResetMaxs Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_ReadMax

This function code is used to read the analog input maximize value for a specific channel.

C/C++

```
int AI2K_ReadMax( int hConnection,
                    BYTE bytChannel,
                    double * dValue);
```

Visual Basic

```
Declare Function AI2K_ReadMax Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, dValue As
Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be read.
dValue	A pointer that stores the specific channel's analog input maximize value to be read. The unit is Vdc for the voltage module, and mA for the current module

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_ReadMaxRaw

This function code is used to read the analog input maximize raw data for a specific channel.

C/C++

```
int AI2K_ReadMaxRaw( int hConnection,
                      BYTE bytChannel,
                      WORD * wValue);
```

Visual Basic

```
Declare Function AI2K_ReadMaxRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be read.
wValue	An point that stores the specific A/I channel's maximize raw data.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_ResetMax

This function code is used to reset the analog input maximize value for a specific channel.

C/C++

```
int AI2K_ResetMax( int hConnection,  
                    BYTE bytChannel);
```

Visual Basic

```
Declare Function AI2K_ResetMax Lib "MXIO.dll" (ByVal  
hConnection As Long, ByVal bytChannel As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_GetRanges

This function code is used to get contiguous A/I channel's range.

C/C++

```
int AI2K_GetRanges( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     WORD wRange[ ] );
```

Visual Basic

```
Declare Function AI2K_GetRanges Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iRange As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wRange	An array that stores the contiguous A/I channel's range, wRange[0] represents the value of the starting channel. The values are: 00: +/-150mV 01: +/-500mV 02: +/-5V 03: +/-10V 04: 0-20mA 05: 4-20mA 06: 0-150mV 07: 0-500mV 08: 0-5V 09: 0-10V Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_SetRanges

This function code is used to set contiguous A/I channel's range.

C/C++

```
int AI2K_SetRanges( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     WORD wRange[ ] );
```

Visual Basic

```
Declare Function AI2K_SetRanges Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iRange As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wRange	An array that stores the contiguous A/I channel's range, wRange[0] represents the value of the starting channel. The values are: 00: +/-150mV 01: +/-500mV 02: +/-5V 03: +/-10V 04: 0-20mA 05: 4-20mA 06: 0-150mV 07: 0-500mV 08: 0-5V 09: 0-10V Others_return Illegal Data Value

Return Value:

Succeed MXIO_OK
Fail Refer to Return Codes.

AI2K_GetRange

This function code is used to get the range for a specific A/I channel.

C/C++

```
int AI2K_GetRange( int hConnection,
                    BYTE bytChannel,
                    WORD * wRange);
```

Visual Basic

```
Declare Function AI2K_GetRange Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iRange As
integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
wRange	An array that stores the contiguous A/I channel's range, wRange[0] represents the value of the starting channel. The values are: 00: +/-150mV 01: +/-500mV 02: +/-5V 03: +/-10V 04: 0-20mA 05: 4-20mA 06: 0-150mV 07: 0-500mV 08: 0-5V 09: 0-10V Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_SetRange

This function code is used to set the range for a specific A/I channel.

C/C++

```
int AI2K_SetRange( int hConnection,
                    BYTE bytChannel,
                    WORD wRange);
```

Visual Basic

```
Declare Function AI2K_SetRange Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal iRange
As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
wRange	An array that stores the contiguous A/I channel's range, wRange[0] represents the value of the starting channel. The values are: 00: +/-150mV 01: +/-500mV 02: +/-5V 03: +/-10V 04: 0-20mA 05: 4-20mA 06: 0-150mV 07: 0-500mV 08: 0-5V 09: 0-10V Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_GetChannelStatus

This function code is used to get the AI channel status of ioLogik 2000 Module.

C/C++

```
int AI2K_GetChannelStatus ( int hConnection,
                            BYTE bytChannel,
                            WORD * wValue );
```

Visual Basic

```
Declare Function AI2K_GetChannelStatus Lib "MXIO.dll"
( ByVal hConnection As Long, ByVal bytChannel As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
wValue	represents the value of the starting channel. 0: disabled, 1: enabled

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_SetChannelStatus

This function code is used to set the AI channel status of ioLogik 2000 Module.

C/C++

```
int AI2K_SetChannelStatus ( int hConnection,
                            BYTE bytChannel,
                            WORD wValue );
```

Visual Basic

```
Declare Function AI2K_SetChannelStatus Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytChannel As Byte,
    ByVal iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
wValue	represents the value of the starting channel. 0: disabled, 1: enabled

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_GetChannelStatuses

This function code is used to get the AI channel statuss of ioLogik 2000 Module.

C/C++

```
int AI2K_GetChannelStatuses ( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                WORD wValue[ ] );
```

Visual Basic

```
Declare Function AI2K_GetChannelStatuses Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wValue	represents the value of the starting channel. 0: disabled, 1: enabled

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AI2K_SetChannelStatuses

This function code is used to set the AI channel statuss of ioLogik 2000 Module.

C/C++

```
int AI2K_SetChannelStatuses ( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                WORD wValue[ ] );
```

Visual Basic

```
Declare Function AI2K_SetChannelStatuses Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	represents the value of the starting channel. 0: disabled, 1: enabled

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_AI_Reads

This function code is used to read contiguous channel~~?????~~s analog input value.

C/C++

```
int E42_AI_Reads( int hConnection,
                    BYTE bytSlot,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    double dValue[ ] );
```

Visual Basic

```
Declare Function E42_AI_Reads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, dValue As
Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel ????? s value, dValue [0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_AI_ReadRaws

This function code is used to read contiguous channel???s analog input raw data.

C/C++

```
int E42_AI_ReadRaws( int hConnection,
                      BYTE bytSlot,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ] );
```

Visual Basic

```
Declare Function E42_AI_ReadRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel???s raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_Reads

This function code is used to read contiguous channel~~◆◆◆~~s analog input value.

C/C++

```
int W5K_AI_Reads( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    double dValue[ ]);
```

Visual Basic

```
Declare Function W5K_AI_Reads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel ◆◆◆ s value, dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadRaws

This function code is used to read contiguous channel~~???~~s analog input raw data.

C/C++

```
int W5K_AI_ReadRaws( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ] );
```

Visual Basic

```
Declare Function W5K_AI_ReadRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel ??? s raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadMins

This function code is used to read contiguous A/I channel?♦?♦?s minimize value.

C/C++

```
int W5K_AI_ReadMins( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] );
```

Visual Basic

```
Declare Function W5K_AI_ReadMins Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel?♦?♦?s value, dValue [0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadMinRaw

This function code is used to read contiguous A/I channel~~?????~~s minimize raw data.

C/C++

```
int W5K_AI_ReadMinRaw( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wValue[ ] );
```

Visual Basic

```
Declare Function W5K_AI_ReadMinRaw Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel ????? s minimize raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ResetMins

This function code is used to reset contiguous A/I channel~~◆◆◆~~s minimize value.

C/C++

```
int W5K_AI_ResetMins( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount);
```

Visual Basic

```
Declare Function W5K_AI_ResetMins Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadMaxs

This function code is used to read contiguous A/I channel~~?????~~s maximize value.

C/C++

```
int W5K_AI_ReadMaxs( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] );
```

Visual Basic

```
Declare Function W5K_AI_ReadMaxs Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel ????? s maximize value , dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadMaxRaws

This function code is used to read contiguous A/I channel~~???~~s maximize raw data.

C/C++

```
int W5K_AI_ReadMaxRaws( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wValue[ ] );
```

Visual Basic

```
Declare Function W5K_AI_ReadMaxRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel ??? s maximize raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ResetMaxs

This function code is used to reset contiguous A/I channel~~???'s~~'s maximize value.

C/C++

```
int W5K_AI_ResetMaxs( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount);
```

Visual Basic

```
Declare Function W5K_AI_ResetMaxs Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_GetRanges

This function code is used to get contiguous A/I channel~~?????~~s range.

C/C++

```
int W5K_AI_GetRanges( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wRange[ ] );
```

Visual Basic

```
Declare Function W5K_AI_GetRanges Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iRange As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wRange	An array that stores the contiguous A/I channel ????? s range, wRange[0] represents the value of the starting channel. The values are: 02: +/-5V 03: +/-10V 04: 0-20mA 05: 4-20mA 09: 0-10V Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_SetRanges

This function code is used to set contiguous A/I channel~~???~~s range.

C/C++

```
int W5K_AI_SetRanges( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wRange[ ] );
```

Visual Basic

```
Declare Function W5K_AI_SetRanges Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iRange As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wRange	An array that stores the contiguous A/I channel ??? s range, wRange[0] represents the value of the starting channel. The values are: 02: +/-5V 03: +/-10V 04: 0-20mA 05: 4-20mA 09: 0-10V Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_GetChannelStatuses

This function code is used to get the AI channel status of ioLogik 5000 Module.

C/C++

```
int W5K_AI_GetChannelStatuses ( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                WORD wValue[ ] );
```

Visual Basic

```
Declare Function W5K_AI_GetChannelStatuses Lib
MXIO.dll (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wValue	represents the value of the starting channel. 0: disabled, 1: enabled

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_SetChannelStatuses

This function code is used to set the AI channel statuss of ioLogik 5000 Module.

C/C++

```
int W5K_AI_SetChannelStatuses ( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                WORD wValue[ ] );
```

Visual Basic

```
Declare Function W5K_AI_SetChannelStatuses Lib
MXIO.dll (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	represents the value of the starting channel. 0: disabled, 1: enabled

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_Reads_Ex

This function code is used to read contiguous channel~~◆◆◆~~s analog input value.

C/C++

```
int W5K_AI_Reads_Ex( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] ,
                      BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_AI_Reads_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel ◆◆◆ s value, dValue [0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadRaws_Ex

This function code is used to read contiguous channel**?????**s analog input raw data.

C/C++

```
int W5K_AI_ReadRaws_Ex( int hConnection,
                         BYTE bytStartChannel,
                         BYTE bytCount,
                         WORD wValue[ ] ,
                         BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_AI_ReadRaws_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel ????? s raw data , wValue[0] represents the value of the starting channel.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadMins_Ex

This function code is used to read contiguous A/I channel~~?????~~s minimize value.

C/C++

```
int W5K_AI_ReadMins_Ex( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        double dValue[ ] ,
                        BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_AI_ReadMins_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel ????? s value, dValue [0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadMinRaws_Ex

This function code is used to read contiguous A/I channel?/?'s minimize raw data.

C/C++

```
int W5K_AI_ReadMinRaws_Ex( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            WORD wValue[ ] ,
                            BYTE BytSlot) ;
```

Visual Basic

```
Declare Function W5K_AI_ReadMinRaws_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel?/?'s minimize raw data , wValue[0] represents the value of the starting channel.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadMaxs_Ex

This function code is used to read contiguous A/I channel~~?????~~s maximize value.

C/C++

```
int W5K_AI_ReadMaxs_Ex( int hConnection,
                         BYTE bytStartChannel,
                         BYTE bytCount,
                         double dValue[ ] ,
                         BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_AI_ReadMaxs_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel ????? s maximize value , dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_ReadMaxRaws_Ex

This function code is used to read contiguous A/I channel?♦?♦?s maximize raw data.

C/C++

```
int W5K_AI_ReadMaxRaws_Ex( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            WORD wValue[ ] ,
                            BYTE BytSlot) ;
```

Visual Basic

```
Declare Function W5K_AI_ReadMaxRaws_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel?♦?♦?s maximize raw data , wValue[0] represents the value of the starting channel.
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_AI_GetRanges_Ex

This function code is used to get contiguous A/I channel~~???'s~~ range.

C/C++

```
int W5K_AI_GetRanges_Ex( int hConnection,
                         BYTE bytStartChannel,
                         BYTE bytCount,
                         WORD wRange[ ] ,
                         BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_AI_GetRanges_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iRange As integer, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wRange	An array that stores the contiguous A/I channel ???'s range, wRange[0] represents the value of the starting channel. The values are: 00: 0-10V 01: 4-20mA Others_return Illegal Data Value
bytSlot	Slot number of the I/O module. The Slot number ranges from 0 to 3. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_VC_Reads_Ex

This function code is used to read contiguous channel????s virtual channel????s value.

C/C++

```
int W5K_VC_Reads_Ex( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] ,
                      BYTE BytSlot);
```

Visual Basic

```
Declare Function W5K_VC_Reads_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double, ByVal bytSlot As Byte)
As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous virtual channel????s value, dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.
bytSlot	Slot number of the I/O module. The Slot number is 0. (0 for W5000 module)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AI_Reads

This function code is used to read contiguous channel~~◆◆◆~~s analog input value.

C/C++

```
int E1K_AI_Reads( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    double dValue[ ]);
```

Visual Basic

```
Declare Function E1K_AI_Reads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel ◆◆◆ s value, dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AI_ReadRaws

This function code is used to read contiguous channel#???'s analog input raw data.

C/C++

```
int E1K_AI_ReadRaws( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_AI_ReadRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel#???'s raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AI_ReadMins

This function code is used to read contiguous A/I channel#???'s minimize value.

C/C++

```
int E1K_AI_ReadMins( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] );
```

Visual Basic

```
Declare Function E1K_AI_ReadMins Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel#???'s value, dValue [0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AI_ReadMinRaw

This function code is used to read contiguous A/I channel~~?????~~s minimize raw data.

C/C++

```
int E1K_AI_ReadMinRaw( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_AI_ReadMinRaw Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel ????? s minimize raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AI_ResetMins

This function code is used to reset contiguous A/I channel~~???s~~'s minimize value.

C/C++

```
int E1K_AI_ResetMins( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount);
```

Visual Basic

```
Declare Function E1K_AI_ResetMins Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AI_ReadMaxs

This function code is used to read contiguous A/I channel~~?????~~s maximize value.

C/C++

```
int E1K_AI_ReadMaxs( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] );
```

Visual Basic

```
Declare Function E1K_AI_ReadMaxs Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel ????? s maximize value , dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AI_ReadMaxRaws

This function code is used to read contiguous A/I channel~~???~~s maximize raw data.

C/C++

```
int E1K_AI_ReadMaxRaws( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_AI_ReadMaxRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel ??? s maximize raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AI_ResetMaxs

This function code is used to reset contiguous A/I channel~~???'s~~'s maximize value.

C/C++

```
int E1K_AI_ResetMaxs( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount);
```

Visual Basic

```
Declare Function E1K_AI_ResetMaxs Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AI_GetRanges

This function code is used to get contiguous A/I channel~~?????~~s range.

C/C++

```
int E1K_AI_GetRanges( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wRange[ ] );
```

Visual Basic

```
Declare Function E1K_AI_GetRanges Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iRange As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wRange	An array that stores the contiguous A/I channel ????? s range, wRange[0] represents the value of the starting channel. The values are: 0: 0-10V 1: 4-20mA Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AI_GetChannelStatuses

This function code is used to get the AI channel status of ioLogik 1200 Module.

C/C++

```
int E1K_AI_GetChannelStatuses ( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_AI_GetChannelStatuses Lib
MXIO.dll (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wValue	represents the value of the starting channel. 0: disabled, 1: enabled

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AI_SetChannelStatuses

This function code is used to set the AI channel statuss of ioLogik 1200 Module.

C/C++

```
int E1K_AI_SetChannelStatuses ( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_AI_SetChannelStatuses Lib
MXIO.dll (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	represents the value of the starting channel. 0: disabled, 1: enabled

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_Reads

This function code is used to read the output value of contiguous Analog Output channels.

C/C++

```
int AO_Reads( int hConnection,
               BYTE bytSlot,
               BYTE bytStartChannel,
               BYTE bytCount,
               double dValue[ ] );
```

Visual Basic

```
Declare Function AO_Reads Lib "MXIO.dll" (ByVal hConnection
As Long, ByVal bytSlot As Byte, ByVal bytStartChannel As
Byte, ByVal bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous channel output value to be read. The dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_Writes

This function code is used to write the output value for contiguous channels.

C/C++

```
int AO_Writes( int hConnection,
                BYTE bytSlot,
                BYTE bytStartChannel,
                BYTE bytCount,
                double dValue[ ] );
```

Visual Basic

```
Declare Function AO_Writes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, dValue As
Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to write.
dValue	An array that stores the contiguous channel output value to write. The dValue[0] represents the value of the starting Analog output channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_Read

This function code is used to read the output value for a specific channel.

C/C++

```
int AO_Read( int hConnection,
             BYTE bytSlot,
             BYTE bytChannel,
             double * dValue);
```

Visual Basic

```
Declare Function AO_Read Lib "MXIO.dll" (ByVal hConnection
As Long, ByVal bytSlot As Byte, ByVal bytChannel As Byte,
dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be read.
dValue	A pointer that stores the specific channel output value to be read. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_Write

This function code is used to write the output value for a specific channel.

C/C++

```
int AO_Write( int hConnection,
              BYTE bytSlot,
              BYTE bytChannel,
              double dValue);
```

Visual Basic

```
Declare Function AO_Write Lib "MXIO.dll" (ByVal hConnection
As Long, ByVal bytSlot As Byte, ByVal bytChannel As Byte,
ByVal dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be written.
dValue	Stores the specific channel output value that is to be written. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_ReadRaws

This function code is used to read the output raw data of contiguous Analog Output channels.

C/C++

```
int AO_ReadRaws( int hConnection,
                  BYTE bytSlot,
                  BYTE bytStartChannel,
                  BYTE bytCount,
                  WORD wValue[ ] );
```

Visual Basic

```
Declare Function AO_ReadRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous channel output raw data to be read. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_WriteRaws

This function code is used to write the output raw data for contiguous channels.

C/C++

```
int AO_WriteRaws( int hConnection,
                   BYTE bytSlot,
                   BYTE bytStartChannel,
                   BYTE bytCount,
                   WORD wValue[ ]);
```

Visual Basic

```
Declare Function AO_WriteRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to write.
wValue	An array that stores the contiguous channel output raw data to write. The wValue[0] represents the value of the starting Analog output channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_ReadRaw

This function code is used to read the output raw data for a specific channel.

C/C++

```
int AO_ReadRaw( int hConnection,
                 BYTE bytSlot,
                 BYTE bytChannel,
                 WORD * wValue);
```

Visual Basic

```
Declare Function AO_ReadRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be read.
wValue	A pointer that stores the specific channel output raw data to be read.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_WriteRaw

This function code is used to write the output raw data for a specific channel.

C/C++

```
int AO_WriteRaw( int hConnection,
                  BYTE bytSlot,
                  BYTE bytChannel,
                  WORD wValue);
```

Visual Basic

```
Declare Function AO_WriteRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, ByVal iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be written.
wValue	Stores the specific channel output raw data that is to be written.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_GetSafeValues

This function code is used to get the output value of contiguous Analog Output channels.

C/C++

```
int AO_GetSafeValues( int hConnection,
                      BYTE bytSlot,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] );
```

Visual Basic

```
Declare Function AO_GetSafeValues Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, dValue As
Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel. M-4402/M-4410 All Channels use the same safe vaule. Just only support bytStartChannel=0 && bytCount=1
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/O channel's safe value to be get. The dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_SetSafeValues

This function code is used to set the safe value for contiguous A/O channels.

C/C++

```
int AO_SetSafeValues( int hConnection,
                      BYTE bytSlot,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ]);
```

Visual Basic

```
Declare Function AO_SetSafeValues Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, dValue As
Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel. M-4402/M-4410 All Channels use the same safe vaule. Just only support bytStartChannel=0 && bytCount=1
bytCount	The number of channels to set.
dValue	An array that stores the contiguous A/O channel's safe value to set. The dValue[0] represents the value of the starting Analog output channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_GetSafeValue

This function code is used to get the safe value for a specific A/O channels.

C/C++

```
int AO_GetSafeValue( int hConnection,
                     BYTE bytSlot,
                     BYTE bytChannel,
                     double * dValue);
```

Visual Basic

```
Declare Function AO_GetSafeValue Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, dValue As Doub) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be read. M-4402/M-4410 All Channels use the same safe value. Just only support bytChannel=0
dValue	A pointer that stores the specific A/O channel's safe value to be get. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_SetSafeValue

This function code is used to set the safe value for a specific A/O channels.

C/C++

```
int AO_SetSafeValue( int hConnection,
                     BYTE bytSlot,
                     BYTE bytChannel,
                     double dValue);
```

Visual Basic

```
Declare Function AO_SetSafeValue Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, ByVal dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be set. M-4402/M-4410 All Channels use the same safe value. Just only support bytChannel=0
dValue	Stores the specific A/O channel's safe value to be set. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_GetSafeRaws

This function code is used to set the output value of contiguous Analog Output channels.

C/C++

```
int AO_GetSafeRaws( int hConnection,
                     BYTE bytSlot,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     WORD wValue[ ] );
```

Visual Basic

```
Declare Function AO_GetSafeRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel. M-4402/M-4410 All Channels use the same safe vaule. Just only support bytStartChannel=0 && bytCount=1
bytCount	The number of channels to be gets.
wValue	An array that stores the contiguous A/O channel's safe raw data to be get. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_SetSafeRaws

This function code is used to set the safe raw data for contiguous A/O channels.

C/C++

```
int AO_SetSafeRaws( int hConnection,
                     BYTE bytSlot,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     WORD wValue[ ] );
```

Visual Basic

```
Declare Function AO_SetSafeRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel. M-4402/M-4410 All Channels use the same safe vaule. Just only support bytStartChannel=0 && bytCount=1
bytCount	The number of channels to write.
wValue	An array that stores the contiguous channel safe raw data to set. The wValue[0] represents the value of the starting Analog output channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_GetSafeRaw

This function code is used to get the safe raw data for a specific A/O channels.

C/C++

```
int AO_GetSafeRaw( int hConnection,
                    BYTE bytSlot,
                    BYTE bytChannel,
                    WORD * wValue);
```

Visual Basic

```
Declare Function AO_GetSafeRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be get. M-4402/M-4410 All Channels use the same safe value. Just only support bytChannel=0
wValue	A pointer that stores the specific channel's safe raw data to be get.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO_SetSafeRaw

This function code is used to set the safe raw data for a specific channel.

C/C++

```
int AO_SetSafeRaw( int hConnection,
                    BYTE bytSlot,
                    BYTE bytChannel,
                    WORD wValue);
```

Visual Basic

```
Declare Function AO_SetSafeRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, ByVal iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be set. M-4402/M-4410 All Channels use the same safe vaule. Just only support bytChannel=0
wValue	Stores the specific channel's safe raw data that is to be set.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO2K_GetRanges

This function code is used to get contiguous A/O channel's range.

C/C++

```
int AO2K_GetRanges( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     WORD wRange[ ] );
```

Visual Basic

```
Declare Function AO2K_GetRanges Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iRange As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wRange	An array that stores the contiguous A/O channel's range, wRange[0] represents the value of the starting channel. The values are: 0: 0-10 Vdc 1: 4-20 mA 0xff : disable Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO2K_SetRanges

This function code is used to set the range for contiguous A/O channels.

C/C++

```
int AO2K_SetRanges( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     WORD wRange[ ] );
```

Visual Basic

```
Declare Function AO2K_SetRanges Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iRange As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to set.
wRange	An array that stores the contiguous A/O channel's range, wRange[0] represents the value of the starting channel. The values are: 0: 0-10 Vdc 1: 4-20 mA <u>Others</u> _return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO2K_GetRange

This function code is used to get the range for a specific A/O channel.

C/C++

```
int AO2K_GetRange( int hConnection,
                    BYTE bytChannel,
                    WORD * wRange);
```

Visual Basic

```
Declare Function AO2K_GetRange Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iRange As
integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
wRange	A point that stores the specific A/O channel's range. The values are: 0: 0-10Vdc 1: 4-20mA 0xff : disable Others _return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO2K_SetRange

This function code is used to set the range for a specific A/O channel.

C/C++

```
int AO2K_SetRange( int hConnection,  
                    BYTE bytChannel,  
                    WORD wRange);
```

Visual Basic

```
Declare Function AO2K_SetRange Lib "MXIO.dll" (ByVal  
hConnection As Long, ByVal bytChannel As Byte, ByVal iRange  
As Integer) As Long
```

Arguments :

hConnection The handle for an I/O device connection.

`byChannel` The specific channel to be set.

wRange Stores the specific A/O channel's range.
The values are:

0 : 0-10Vdc

1 : 4-20mA

Others_return Illegal Data Value

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

AO2K_GetPowerOnValues

This function code is used to get the power on value of contiguous A/O channels.

C/C++

```
int AO2K_GetPowerOnValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           double dValue[ ] );
```

Visual Basic

```
Declare Function AO2K_GetPowerOnValues Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get.
dValue	An array that stores the contiguous A/O channel's power on value to be get. The dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO2K_SetPowerOnValues

This function code is used to set the power on value for contiguous A/O channels.

C/C++

```
int AO2K_SetPowerOnValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           double dValue[ ]);
```

Visual Basic

```
Declare Function AO2K_SetPowerOnValues Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to set.
dValue	An array that stores the contiguous A/O channel's power on value to be set. The dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO2K_GetPowerOnValue

This function code is used to get the power on value for a specific channel.

C/C++

```
int AO2K_GetPowerOnValue( int hConnection,
                           BYTE bytChannel,
                           double * dValue);
```

Visual Basic

```
Declare Function AO2K_GetPowerOnValue Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, dValue As
Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
dValue	A pointer that stores the specific A/O channel's power on value to be get. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO2K_SetPowerOnValue

This function code is used to set the power on value for a specific channel.

C/C++

```
int AO2K_SetPowerOnValue( int hConnection,
                           BYTE bytChannel,
                           double dValue);
```

Visual Basic

```
Declare Function AO2K_SetPowerOnValue Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal dValue
As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
dValue	Stores the specific A/O channel's power on value to be set. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO2K_GetPowerOnRaws

This function code is used to get the power on raw data of contiguous A/O channels.

C/C++

```
int AO2K_GetPowerOnRaws( int hConnection,
                         BYTE bytStartChannel,
                         BYTE bytCount,
                         WORD wValue[ ] );
```

Visual Basic

```
Declare Function AO2K_GetPowerOnRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/O channel's power on raw data to be get. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO2K_SetPowerOnRaws

This function code is used to set the power on raw data for contiguous A/O channels.

C/C++

```
int AO2K_SetPowerOnRaws( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wValue[ ] );
```

Visual Basic

```
Declare Function AO2K_SetPowerOnRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to set.
wValue	An array that stores the contiguous A/O channel's power on raw data to be set. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO2K_GetPowerOnRaw

This function code is used to get the power on raw data for a specific channel.

C/C++

```
int AO2K_GetPowerOnRaw( int hConnection,
                        BYTE bytChannel,
                        WORD * wValue);
```

Visual Basic

```
Declare Function AO2K_GetPowerOnRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be read.
wValue	A pointer that stores the specific A/O channel's power on raw data to be get.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO2K_SetPowerOnRaw

This function code is used to set the output raw data for a specific channel.

C/C++

```
int AO2K_SetPowerOnRaw( int hConnection,
                        BYTE bytChannel,
                        WORD wValue);
```

Visual Basic

```
Declare Function AO2K_SetPowerOnRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal iValue
As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
wValue	Stores the specific A/O channel's power on raw data to be set.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO4K_GetSafeActions

This function code is used to get the safe action of contiguous A/O channels.

C/C++

```
int AO4K_GetSafeActions( int hConnection,
                        BYTE bytSlot,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wAction[ ] );
```

Visual Basic

```
Declare Function AO4K_GetSafeActions Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iAction As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wAction	An array that stores the contiguous A/O channel's safe action to be get. The wAction[0] represents the value of the starting channel. The values are: 0: Safe value 1: Hold last state 2: Low Limit 3: High Limit

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO4K_SetSafeActions

This function code is used to set the safe action of contiguous A/O channels.

C/C++

```
int AO4K_SetSafeActions( int hConnection,
                         BYTE bytSlot,
                         BYTE bytStartChannel,
                         BYTE bytCount,
                         WORD wAction[ ] );
```

Visual Basic

```
Declare Function AO4K_SetSafeActions Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iAction As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wAction	An array that stores the contiguous A/O channel's safe action to be set. The wAction[0] represents the value of the starting channel. The values are: 0: Safe value 1: Hold last state 2: Low Limit 3: High Limit

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO4K_GetSafeAction

This function code is used to get the safe action for a specific channel.

C/C++

```
int AO4K_GetSafeAction( int hConnection,
                        BYTE bytSlot,
                        BYTE bytChannel,
                        WORD *wAction);
```

Visual Basic

```
Declare Function AO4K_GetSafeAction Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, iAction As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32.
bytChannel	The specific channel to be get.
wAction	A pointer that stores the specific A/O channel's safe action to be get. The values are: 0: Safe value 1: Hold last state 2: Low Limit 3: High Limit

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

AO4K_SetSafeAction

This function code is used to set the safe action for a specific channel.

C/C++

```
int AO4K_SetSafeAction( int hConnection,
                        BYTE bytSlot,
                        BYTE bytChannel,
                        WORD wAction);
```

Visual Basic

```
Declare Function AO4K_SetSafeAction Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, ByVal iAction As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32.
bytChannel	The specific channel to be set.
wAction	Stores the specific A/O channel's safe action to be get. The values are: 0: Safe value 1: Hold last state 2: Low Limit 3: High Limit

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_AO_GetSafeActions

This function code is used to get the safe action of contiguous A/O channels.

C/C++

```
int E42_AO_GetSafeActions( int hConnection,
                           BYTE bytSlot,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wAction[ ] );
```

Visual Basic

```
Declare Function E42_AO_GetSafeActions Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iAction As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wAction	An array that stores the contiguous A/O channel#???'s safe action to be get. The wAction[0] represents the value of the starting channel. The values are: 0: Safe value 1: Hold last state 2: Low Limit 3: High Limit

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_AO_SetSafeActions

This function code is used to set the safe action of contiguous A/O channels.

C/C++

```
int E42_AO_SetSafeActions( int hConnection,
                           BYTE bytSlot,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wAction[ ] );
```

Visual Basic

```
Declare Function E42_AO_SetSafeActions Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iAction As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wAction	An array that stores the contiguous A/O channel's safe action to be set. The wAction[0] represents the value of the starting channel. The values are: 0: Safe value 1: Hold last state 2: Low Limit 3: High Limit

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_AO_GetPowerOnValues

This function code is used to get the power on raw data of contiguous A/O channels.

C/C++

```
int E42_AO_GetPowerOnValues( int hConnection,
                            BYTE bytSlot
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            WORD wValue[ ] );
```

Visual Basic

```
Declare Function E42_AO_GetPowerOnValues Lib MXIO.dll
    (ByVal hConnection As Long, ByVal bytSlot As Byte, ByVal
     bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
     Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/O channel???'s power on raw data to be get. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_AO_SetPowerOnValues

This function code is used to set the power on raw data for contiguous A/O channels.

C/C++

```
int E42_AO_SetPowerOnValues( int hConnection,
                            BYTE bytSlot,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            WORD wValue[ ] );
```

Visual Basic

```
Declare Function E42_AO_SetPowerOnValues Lib MXIO.dll
    (ByVal hConnection As Long, ByVal bytSlot As Byte, ByVal
     bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
     Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to set.
wValue	An array that stores the contiguous A/O channel#???'s power on raw data to be set. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_AO_Reads

This function code is used to read the output value of contiguous Analog Output channels.

C/C++

```
int E42_AO_Reads( int hConnection,
                   BYTE bytSlot,
                   BYTE bytStartChannel,
                   BYTE bytCount,
                   double dValue[ ]);
```

Visual Basic

```
Declare Function E42_AO_Reads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, dValue As
Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous channel output value to be read. The dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_AO_Writes

This function code is used to write the output value for contiguous channels.

C/C++

```
int E42_AO_Writes( int hConnection,
                     BYTE bytSlot,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     double dValue[ ] );
```

Visual Basic

```
Declare Function E42_AO_Writes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, dValue As
Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to write.
dValue	An array that stores the contiguous channel output value to write. The dValue[0] represents the value of the starting Analog output channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_AO_ReadRaws

This function code is used to read the output raw data of contiguous Analog Output channels.

C/C++

```
int E42_AO_ReadRaws( int hConnection,
                      BYTE bytSlot,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ] );
```

Visual Basic

```
Declare Function E42_AO_ReadRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous channel output raw data to be read. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_AO_WriteRaws

This function code is used to write the output raw data for contiguous channels.

C/C++

```
int E42_AO_WriteRaws( int hConnection,
                      BYTE bytSlot,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ] );
```

Visual Basic

```
Declare Function E42_AO_WriteRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to write.
wValue	An array that stores the contiguous channel output raw data to write. The wValue[0] represents the value of the starting Analog output channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_AO_GetFaultValues

This function code is used to set the output value of contiguous Analog Output channels.

C/C++

```
int E42_AO_GetFaultValues( int hConnection,
                           BYTE bytSlot,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wValue[ ] );
```

Visual Basic

```
Declare Function E42_AO_GetFaultValues Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel. <i>M-4402/M-4410 All Channels use the same fault value. Just only support bytStartChannel=0 && bytCount=1</i>
bytCount	The number of channels to be gets.
wValue	An array that stores the contiguous A/O channel's safe raw data to be get. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_AO_SetFaultValues

This function code is used to set the safe raw data for contiguous A/O channels.

C/C++

```
int E42_AO_SetFaultValues( int hConnection,
                           BYTE bytSlot,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wValue[ ]);
```

Visual Basic

```
Declare Function E42_AO_SetFaultValues Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel. <i>M-4402/M-4410 All Channels use the same fault value. Just only support bytStartChannel=0 && bytCount=1</i>
bytCount	The number of channels to write.
wValue	An array that stores the contiguous channel safe raw data to set. The wValue[0] represents the value of the starting Analog output channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_Reads

This function code is used to read the output value of contiguous Analog Output channels.

C/C++

```
int E1K_AO_Reads( int hConnection,
                   BYTE bytStartChannel,
                   BYTE bytCount,
                   double dValue[ ] );
```

Visual Basic

```
Declare Function E1K_AO_Reads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous channel output value to be read. The dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_Writes

This function code is used to write the output value for contiguous channels.

C/C++

```
int E1K_AO_Writes( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    double dValue[ ]);
```

Visual Basic

```
Declare Function E1K_AO_Writes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to write.
dValue	An array that stores the contiguous channel output value to write. The dValue[0] represents the value of the starting Analog output channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_ReadRaws

This function code is used to read the output raw data of contiguous Analog Output channels.

C/C++

```
int E1K_AO_ReadRaws( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_AO_ReadRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous channel output raw data to be read. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_WriteRaws

This function code is used to write the output raw data for contiguous channels.

C/C++

```
int E1K_AO_WriteRaws( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ]);
```

Visual Basic

```
Declare Function E1K_AO_WriteRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to write.
wValue	An array that stores the contiguous channel output raw data to write. The wValue[0] represents the value of the starting Analog output channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_GetSafeValues

This function code is used to get the output value of contiguous Analog Output channels.

C/C++

```
int E1K_AO_GetSafeValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           double dValue[ ] );
```

Visual Basic

```
Declare Function E1K_AO_GetSafeValues Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/O channel#???'s safe value to be get. The dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_SetSafeValues

This function code is used to set the safe value for contiguous A/O channels.

C/C++

```
int E1K_AO_SetSafeValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           double dValue[ ] );
```

Visual Basic

```
Declare Function E1K_AO_SetSafeValues Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to set.
dValue	An array that stores the contiguous A/O channel's safe value to set. The dValue[0] represents the value of the starting Analog output channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_GetSafeRaws

This function code is used to set the output value of contiguous Analog Output channels.

C/C++

```
int E1K_AO_GetSafeRaws( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_AO_GetSafeRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wValue	An array that stores the contiguous A/O channel#???'s safe raw data to be get. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_SetSafeRaws

This function code is used to set the safe raw data for contiguous A/O channels.

C/C++

```
int E1K_AO_SetSafeRaws( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_AO_SetSafeRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to write.
wValue	An array that stores the contiguous channel safe raw data to set. The wValue[0] represents the value of the starting Analog output channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_GetRanges

This function code is used to get contiguous A/O channel~~?????~~s range.

C/C++

```
int E1K_AO_GetRanges( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wRange[ ] );
```

Visual Basic

```
Declare Function E1K_AO_GetRanges Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iRange As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wRange	An array that stores the contiguous A/O channel ????? s range, wRange[0] represents the value of the starting channel. The values are: 0: 0-10 Vdc 1: 4-20 mA 0xff : disable Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_SetRanges

This function code is used to set the range for contiguous A/O channels.

C/C++

```
int E1K_AO_SetRanges( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wRange[ ] );
```

Visual Basic

```
Declare Function E1K_AO_SetRanges Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iRange As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to set.
wRange	An array that stores the contiguous A/O channel#???'s range, wRange[0] represents the value of the starting channel. The values are: 0: 0-10 Vdc 1: 4-20 mA Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_GetPowerOnValues

This function code is used to get the power on value of contiguous A/O channels.

C/C++

```
int E1K_AO_GetPowerOnValues( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            double dValue[ ] );
```

Visual Basic

```
Declare Function E1K_AO_GetPowerOnValues Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get.
dValue	An array that stores the contiguous A/O channel#???'s power on value to be get. The dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_SetPowerOnValues

This function code is used to set the power on value for contiguous A/O channels.

C/C++

```
int E1K_AO_SetPowerOnValues( int hConnection,
                             BYTE bytStartChannel,
                             BYTE bytCount,
                             double dValue[ ] );
```

Visual Basic

```
Declare Function E1K_AO_SetPowerOnValues Lib MXIO.dll
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to set.
dValue	An array that stores the contiguous A/O channel#???'s power on value to be set. The dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_GetPowerOnRaws

This function code is used to get the power on raw data of contiguous A/O channels.

C/C++

```
int E1K_AO_GetPowerOnRaws( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_AO_GetPowerOnRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/O channel#???'s power on raw data to be get. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_SetPowerOnRaws

This function code is used to set the power on raw data for contiguous A/O channels.

C/C++

```
int E1K_AO_SetPowerOnRaws( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_AO_SetPowerOnRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to set.
wValue	An array that stores the contiguous A/O channel#???'s power on raw data to be set. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_GetChannelStatuses

This function code is used to get the AO channel status of ioLogik 1200 Module.

C/C++

```
int E1K_AO_GetChannelStatuses ( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_AO_GetChannelStatuses Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wValue	represents the value of the starting channel. 0: disabled, 1: enabled

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_AO_SetChannelStatuses

This function code is used to set the AO channel statuss of ioLogik 1200 Module.

C/C++

```
int E1K_AO_SetChannelStatuses ( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_AO_SetChannelStatuses Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	represents the value of the starting channel. 0: disabled, 1: enabled

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD_Reads

This function code is used to read the temperature value for contiguous channels.

C/C++

```
int RTD_Reads( int hConnection,
                BYTE bytSlot,
                BYTE bytStartChannel,
                BYTE bytCount,
                double dValue[ ] );
```

Visual Basic

```
Declare Function RTD_Reads Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the temperature value to be read. The dValue[0] represents the start channel. When the dValue is 0x7FFF, it means the sensor is not wired correctly, or the measured value is out of range. When using the RTD module for Resistance Input, the unit is Ohm. When the operating mode is temperature sensor, the unit is C or F, depending on the setting. Check the ioAdmin utility for current settings.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD_Read

This function code is used to read the temperature value for a specific channel.

C/C++

```
int RTD_Read( int hConnection,
               BYTE bytSlot,
               BYTE bytChannel,
               double * dValue);
```

Visual Basic

```
Declare Function RTD_Read Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be read.
dValue	A pointer that stores the specific channel's temperature value to be read. When the dValue is 0x7FFF, it means the sensor is not correctly wired or the measured value is out of range. When using the RTD module for Resistance Input, the unit is Ohm. When the operating mode is temperature sensor, the unit is C or F, depending on the setting. Check the ioAdmin utility for current settings.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD_ReadRaw

This function code is used to read the temperature raw data for contiguous channels.

C/C++

```
int RTD_ReadRaw( int hConnection,
                  BYTE bytSlot,
                  BYTE bytStartChannel,
                  BYTE bytCount,
                  WORD wValue[ ] );
```

Visual Basic

```
Declare Function RTD_ReadRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the temperature value to be read. The wValue[0] represents the start channel. When the wValue is 0x7FFF, it means the sensor is not wired correctly, or the measured value is out of range. When using the RTD module for Resistance Input 1~2000毫, 100m毫/1count. When using the RTD module for Resistance Input 1~327毫, 10m毫/1count. When using the RTD module for Resistance Input 1~620毫, 20m毫/1count.

When the operating mode is temperature sensor, 0.1C(F)/1count, depending on the setting. Check the ioAdmin utility for current settings.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD_ReadRaw

This function code is used to read the temperature raw data for a specific channel.

C/C++

```
int RTD_ReadRaw( int hConnection,
                  BYTE bytSlot,
                  BYTE bytChannel,
                  WORD * wValue);
```

Visual Basic

```
Declare Function RTD_ReadRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be read.
wValue	An pointer that stores the temperature value to be read. When the wValue is 0x7FFF, it means the sensor is not wired correctly, or the measured value is out of range. When using the RTD module for Resistance Input 1~2000m Ω , 100m Ω /1count. When using the RTD module for Resistance Input 1~327m Ω , 10m Ω /1count. When using the RTD module for Resistance Input 1~620m Ω , 20m Ω /1count. When the operating mode is temperature sensor, 0.1C(F)/1count, depending on the setting. Check the ioAdmin utility for current settings.

Return Value:

Succeed MXIO_OK
Fail Refer to Return Codes.

RTD2K_ResetMin

This function code is used to reset the RTD input minimize value for a specific channel.

C/C++

```
int RTD2K_ResetMin ( int hConnection,
                      BYTE bytChannel );
```

Visual Basic

```
Declare Function RTD2K_ResetMin Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_ResetMins

This function code is used to reset contiguous RTD channel's minimize value.

C/C++

```
int RTD2K_ResetMins( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount);
```

Visual Basic

```
Declare Function RTD2K_ResetMins Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_ResetMax

This function code is used to reset the RTD input maximize value for a specific channel.

C/C++

```
int RTD2K_ResetMax( int hConnection,  
                      BYTE bytChannel);
```

Visual Basic

```
Declare Function RTD2K_ResetMax Lib "MXIO.dll" (ByVal  
hConnection As Long, ByVal bytChannel As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_ResetMaxs

This function code is used to reset contiguous RTD channel's maximize value.

C/C++

```
int RTD2K_ResetMaxs( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount);
```

Visual Basic

```
Declare Function RTD2K_ResetMaxs Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_ReadMinRaw

This function code is used to read the RTD input minimize raw data for a specific channel.

C/C++

```
int RTD2K_ReadMinRaw( int hConnection,
                      BYTE bytChannel,
                      WORD * iValue);
```

Visual Basic

```
Declare Function RTD2K_ReadMinRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, nValue As
Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be read.
iValue	An point that stores the specific RTD channel's minimize raw data.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_ReadMinRaws

This function code is used to read contiguous RTD channel's minimize raw data.

C/C++

```
int RTD2K_ReadMinRaws( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wValue[ ]);
```

Visual Basic

```
Declare Function RTD2K_ReadMinRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous RTD channel's minimize raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_ReadMaxRaw

This function code is used to read the RTD input maximize raw data for a specific channel.

C/C++

```
int RTD2K_ReadMaxRaw( int hConnection,
                      BYTE bytChannel,
                      WORD * wValue);
```

Visual Basic

```
Declare Function RTD2K_ReadMaxRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be read.
wValue	An point that stores the specific RTD channel's maximize raw data.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_ReadMaxRaws

This function code is used to read contiguous RTD channel's maximize raw data.

C/C++

```
int RTD2K_ReadMaxRaws( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wValue[ ] );
```

Visual Basic

```
Declare Function RTD2K_ReadMaxRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous RTD channel's maximize raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_ReadMin

This function code is used to read the RTD input minimize value for a specific channel.

C/C++

```
int RTD2K_ReadMin( int hConnection,
                     BYTE bytChannel,
                     double *dValue);
```

Visual Basic

```
Declare Function RTD2K_ReadMin Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, dValue As
Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be read.
dValue	A pointer that stores the specific channel RTD input minimize value to be read. The unit is $\mu\Omega$ for the Ohm, m°C for Celsius and m°F for Fahrenheit.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_ReadMins

This function code is used to read contiguous RTD channel's minimize value.

C/C++

```
int RTD2K_ReadMins( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     double dValue[ ] );
```

Visual Basic

```
Declare Function RTD2K_ReadMins Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous RTD channel's minimize value , dValue[0] represents the value of the starting channel. The unit is Ω for the Ohm, ??? for Celsius and ??? for Fahrenheit.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_ReadMax

This function code is used to read the RTD input maximize value for a specific channel.

C/C++

```
int RTD2K_ReadMax( int hConnection,
                     BYTE bytChannel,
                     double *dValue);
```

Visual Basic

```
Declare Function RTD2K_ReadMax Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, dValue As
Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be read.
dValue	A pointer that stores the specific channel RTD input maximize value to be read. The unit is Ohm , Celsius for Celsius and Fahrenheit for Fahrenheit.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_ReadMaxs

This function code is used to read contiguous RTD channel's maximize value.

C/C++

```
int RTD2K_ReadMaxs( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     double dValue[ ] );
```

Visual Basic

```
Declare Function RTD2K_ReadMaxs Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous RTD channel's minimize value ,dValue[0] represents the value of the starting channel. The unit is Ω for the Ohm, ℃ for Celsius and ℉ for Fahrenheit.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_GetChannelStatus

This function code is used to get specific channel's status.

C/C++

```
int RTD2K_GetChannelStatus( int hConnection,
                            BYTE bytChannel,
                            BYTE * bytStatus);
```

Visual Basic

```
Declare Function RTD2K_GetChannelStatus Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytChannel As Byte,
     bytStatus As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
bytStatus	A pointer that stores the specific RTD channel's status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_SetChannelStatus

This function code is used to set specific channel's status.

C/C++

```
int RTD2K_SetChannelStatus( int hConnection,
                            BYTE bytChannel,
                            BYTE bytStatus);
```

Visual Basic

```
Declare Function RTD2K_SetChannelStatus Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytChannel As Byte, ByVal
bytStatus As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
bytStatus	A pointer that stores the specific RTD channel's status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_GetChannelStatuses

This function code is used to get contiguous channel's status.

C/C++

```
int RTD2K_GetChannelStatuses( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD * dwStatus) ;
```

Visual Basic

```
Declare Function RTD2K_GetChannelStatuses Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous RTD channel's status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_SetChannelStatuses

This function code is used to set contiguous channel's status.

C/C++

```
int RTD2K_SetChannelStatuses( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD dwStatus );
```

Visual Basic

```
Declare Function RTD2K_SetChannelStatuses Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nStatus As Long) As Long
```

Arguments:

Hconnection	The handle for an I/O device connection.
BytStartChannel	Specifies the starting channel.
BytCount	The number of channels to be set.
DwStatus	A pointer that stores the contiguous count channel's status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_GetSensorType

This function code is used to get the sensor type for a specific RTD channel.

C/C++

```
int RTD2K_GetSensorType( int hConnection,
                         BYTE bytChannel,
                         WORD * wSensorType );
```

Visual Basic

```
Declare Function RTD2K_GetSensorType Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iSensorType
As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
wSensorType	A pointer that stores the specific RTD channel's sensor type. The values for normal channels are: 0=PT50 1=PT100 2=PT200 3=PT500 4=PT1000 5=JPT100 6=JPT200 7=JPT500 8=JPT1000 9=NI100 10=NI200 11=NI500 12=NI1000 13=NI120 14=310 Ohm

15=620 Ohm

16=1250 Ohm

17=2200 Ohm

Others return Illegal Data Value

The values for virtual channels are:

20=AVG

21=DIV

Others return Illegal Data Value

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

RTD2K_SetSensorType

This function code is used to set the sensor type for a specific RTD channel.

C/C++

```
int RTD2K_SetSensorType( int hConnection,
                         BYTE bytChannel,
                         WORD wSensorType );
```

Visual Basic

```
Declare Function RTD2K_SetSensorType Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal
iSensorType As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
wSensorType	A pointer that stores the specific RTD channel's sensor type. The values for normal channels are: 0=PT50 1=PT100 2=PT200 3=PT500 4=PT1000 5=JPT100 6=JPT200 7=JPT500 8=JPT1000 9=NI100 10=NI200 11=NI500 12=NI1000 13=NI120 14=310 Ohm

15=620 Ohm

16=1250 Ohm

17=2200 Ohm

Others return Illegal Data Value

The values for virtual channels are:

20=AVG

21=DIV

Others return Illegal Data Value

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

RTD2K_GetSensorTypes

This function code is used to get contiguous RTD channel's sensor type.

C/C++

```
int RTD2K_GetSensorTypes ( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wSensorType[ ] );
```

Visual Basic

```
Declare Function RTD2K_GetSensorTypes Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iSensorType As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wSensorType	An array that stores the contiguous RTD channel's sensor type, wSensorType[0] represents the value of the starting channel. The values for normal channels are: 0=PT50 1=PT100 2=PT200 3=PT500 4=PT1000 5=JPT100 6=JPT200 7=JPT500 8=JPT1000 9=NI100 10=NI200 11=NI500

12=NI1000

13=NI120

14=310 Ohm

15=620 Ohm

16=1250 Ohm

17=2200 Ohm

Others return Illegal Data Value

The values for virtual channels are:

20=AVG

21=DIV

Others return Illegal Data Value

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

RTD2K_SetSensorTypes

This function code is used to set contiguous RTD channel's sensor type.

C/C++

```
int RTD2K_SetSensorTypes ( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wSensorType[ ] );
```

Visual Basic

```
Declare Function RTD2K_SetSensorTypes Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iSensorType As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wSensorType	An array that stores the contiguous RTD channel's sensor type, wSensorType[0] represents the value of the starting channel. The values for normal channels are: 0=PT50 1=PT100 2=PT200 3=PT500 4=PT1000 5=JPT100 6=JPT200 7=JPT500 8=JPT1000 9=NI100 10=NI200 11=NI500

12=NI1000
13=NI120
14=310 Ohm
15=620 Ohm
16=1250 Ohm
17=2200 Ohm

Others return Illegal Data Value
The values for virtual channels are:

20=AVG
21=DIV

Others return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_GetEngUnit

This function code is used to get the engineering unit for a specific RTD channel.

C/C++

```
int RTD2K_GetEngUnit ( int hConnection,
                      BYTE bytChannel,
                      WORD * wEngUnit );
```

Visual Basic

```
Declare Function RTD2K_GetEngUnit Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iEngUnit As
integer) As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytChannel The specific channel to be get.

wEngUnit A pointer that stores the specific RTD channel's engineering unit. The values for normal channels are:

0=Celsius

1=Fahrenheit

2=Ohm

Others_return Illegal Data Value

The values for virtual channels are:

0=Celsius

1=Fahrenheit

Others_return Illegal Data Value

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

RTD2K_SetEngUnit

This function code is used to set the engineering unit for a specific RTD channel.

C/C++

```
int RTD2K_SetEngUnit ( int hConnection,  
                        BYTE bytChannel,  
                        WORD wEngUnit);
```

Visual Basic

```
Declare Function RTD2K_SetEngUnit Lib "MXIO.dll" (ByVal hConnection As Long, ByVal bytChannel As Byte, ByVal iEngUnit As integer) As Long
```

Arguments :

hConnection The handle for an I/O device connection.

bytChannel The specific channel to be set.

wEngUnit A pointer that stores the specific RTD channel's engineering unit. The values for normal channels are:

0=Celsius

1=Fahrenheit

$$2 = \Omega m$$

Others_return Illegal Data Value

The values for virtual channels are:

0=Celsius

1=Fahrenheit

Others_return Illegal Data Value

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

RTD2K_GetEngUnits

This function code is used to get contiguous RTD channel's engineering unit.

C/C++

```
int RTD2K_GetEngUnits ( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wEngUnit[ ] );
```

Visual Basic

```
Declare Function RTD2K_GetEngUnits Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iEngUnit As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get.
wEngUnit	An array that stores the contiguous RTD channel's engineering unit, wEngUnit[0] represents the value of the starting channel. The values for normal channel are: 0=Celsius 1=Fahrenheit 2=Ohm Others_return Illegal Data Value

The values for virtual channels are:

0=Celsius
1=Fahrenheit
Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
---------	---------

Fail

Refer to Return Codes.

RTD2K_SetEngUnits

This function code is used to set contiguous RTD channel's engineering unit.

C/C++

```
int RTD2K_SetEngUnits ( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wEngUnit[ ] );
```

Visual Basic

```
Declare Function RTD2K_SetEngUnits Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iEngUnit As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wEngUnit	An array that stores the contiguous RTD channel's engineering unit, wEngUnit[0] represents the value of the starting channel. The values for normal channel are: 0=Celsius 1=Fahrenheit 2=Ohm Others_return Illegal Data Value

The values for virtual channels are:

0=Celsius
1=Fahrenheit
Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_GetMathPar

This function code is used to get the math parameter for a specific RTD virtual channel.

C/C++

```
int RTD2K_GetMathPar( int hConnection,
                      BYTE bytChannel,
                      WORD * wMathPar );
```

Visual Basic

```
Declare Function RTD2K_GetMathPar Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iMathPar As
integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
wMathPar	A pointer that stores the specific RTD virtual channel's math parameter. For AVG, Bit 0 of high byte represents the first channel and Bit 1 of high byte represents the second channel. For DEV, the High-Byte as subtrahend and Low-Byte as minuend. Exp AVG(b'0000-0000 b'0010-0011) = ch5+ch1+ch0 Exp DEV(b'0000-0100 b'0010-0000) = ch2-ch6
Return Value:	
Succeed MXIO_OK	
Fail Refer to Return Codes.	

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_SetMathPar

This function code is used to set the math parameter for a specific RTD virtual channel.

C/C++

```
int RTD2K_SetMathPar ( int hConnection,
                      BYTE bytChannel,
                      WORD wMathPar );
```

Visual Basic

```
Declare Function RTD2K_SetMathPar Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal
iMathPar As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
wMathPar	A pointer that stores the specific RTD virtual channel's math parameter. For AVG, Bit 0 of high byte represents the first channel and Bit 1 of high byte represents the second channel. For DEV, the High-Byte as subtrahend and Low-Byte as minuend. Exp AVG(b'0000-0000 b'0010-0011) = ch5+ch1+ch0 Exp DEV(b'0000-0100 b'0010-0000) = ch2-ch6
Return Value:	
Succeed MXIO_OK	
Fail Refer to Return Codes.	

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_GetMathPars

This function code is used to get contiguous RTD virtual channel's math parameter.

C/C++

```
int RTD2K_GetMathPars ( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wMathPar[ ] );
```

Visual Basic

```
Declare Function RTD2K_GetMathPars Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMathPar As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get.
wMathPar	An array that stores the contiguous RTD virtual channel's math parameter, wMathPar[0] represents the value of the starting channel. The values are : For AVG, Bit 0 of high byte represents the first channel and Bit 1 of high byte represents the second channel. For DEV, High-Byte as subtrahend and Low-Byte as minuend. Exp AVG(b'0000-0000 b'0010-0011) = ch5+ch1+ch0 Exp DEV(b'0000-0100 b'0010-0000) = ch2-ch6

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_SetMathPars

This function code is used to set contiguous RTD virtual channel's math parameter.

C/C++

```
int RTD2K_SetMathPars ( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wMathPar[ ] );
```

Visual Basic

```
Declare Function RTD2K_SetMathPars Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMathPar As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wMathPar	An array that stores the contiguous RTD virtual channel's math parameter, wMathPar[0] represents the value of the starting channel. The values are : For AVG, Bit 0 of high byte represents the first channel and Bit 1 of high byte represents the second channel. For DEV, High-Byte as subtrahend and Low-Byte as minuend. Exp AVG(b'0000-0000 b'0010-0011) = ch5+ch1+ch0 Exp DEV(b'0000-0100 b'0010-0000) = ch2-ch6

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_SetChnAvg

This function is used to set data structure variables for averaging data of each channel.

C/C++

```
int RTD2K_SetChnAvg ( int hConnection,
                      BYTE bytChannel,
                      BYTE bytChnIdx[ ],
                      BYTE bytChCount );
```

Visual Basic

```
Declare Function RTD2K_SetChnAvg Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, bytChnIdx As
Byte, ByVal bytChCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
bytChnIdx	An channel index which is used to be averaged.
BytChCount	The number of channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RTD2K_SetChnDev

This function is used to set data structure variables for subtracting data of two channels.

C/C++

```
int RTD2K_SetChnDev ( int hConnection,
                      BYTE bytChannel,
                      BYTE bytChMinued,
                      BYTE bytChSub) ;
```

Visual Basic

```
Declare Function RTD2K_SetChnDev Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal
bytChMinued As Byte, ByVal bytChSub As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific virtual channel to be set.
bytChMinued	This variable is the channel that defined as a subtrahend.
BytChSub	This variable is the channel that defined as a minuend.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_RTD_Reads

This function code is used to read the temperature value for contiguous channels.

C/C++

```
int E42_RTD_Reads( int hConnection,
                    BYTE bytSlot,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    double dValue[ ]);
```

Visual Basic

```
Declare Function E42_RTD_Reads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the temperature value to be read. The dValue[0] represents the start channel. When the dValue is 0x8000, it means the sensor is not wired correctly, or the measured value is out of range. When using the E42_RTD module for Resistance Input, the unit is Ohm. When the operating mode is temperature sensor, the unit is C or F, depending on the setting. Check the ioAdmin utility for current settings.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_RTD_ReadRaws

This function code is used to read the temperature raw data for contiguous channels.

C/C++

```
int E42_RTD_ReadRaws( int hConnection,
                      BYTE bytSlot,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ] );
```

Visual Basic

```
Declare Function E42_RTD_ReadRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the temperature value to be read. The wValue[0] represents the start channel. When the wValue is 0x8000, it means the sensor is not wired correctly, or the measured value is out of range. When using the E42_RTD module for Resistance Input 1~2000毫, 100m毫/1count. When using the E42_RTD module for Resistance Input 1~327毫, 10m毫/1count. When using the E42_RTD module for Resistance Input 1~620毫, 20m毫/1count.

When the operating mode is temperature sensor, 0.1C(F)/1count, depending on the setting. Check the ioAdmin utility for current settings.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_RTD_GetEngUnit

This function code is used to get the temperature Type for contiguous channels.

C/C++

```
int E42_RTD_GetEngUnit( int hConnection,
                        BYTE bytSlot,
                        WORD wEngUnit[ ] );
```

Visual Basic

```
Declare Function E42_RTD_GetEngUnit Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, iEngUnit As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16.
wEngUnit	An array that stores the contiguous A/O channel's safe action to be get. The wEngUnit[0] represents the value of all channels. The values are: 0: Celsius 1: Fahrenheit

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_RTD_SetEngUnit

This function code is used to set the temperature Type for contiguous channels.

C/C++

```
int E42_RTD_SetEngUnit( int hConnection,
                        BYTE bytSlot,
                        WORD wEngUnit);
```

Visual Basic

```
Declare Function E42_RTD_SetEngUnit Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal iEngUnit
As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16.
wEngUnit	An array that stores the contiguous A/O channel's safe action to be set. The wEngUnit[0] represents the value of All channels. The values are: 0: Celsius 1: Fahrenheit

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_RTD_GetSensorType

This function code is used to get the Sensor Type for contiguous channels.

C/C++

```
int E42_RTD_GetSensorType( int hConnection,
                           BYTE bytSlot,
                           WORD wSensorType[ ] );
```

Visual Basic

```
Declare Function E42_RTD_GetSensorType Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, iSensorType As
Integer) As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytSlot Slot number of the I/O module. The Slot number ranges from 1 to 16.

wSensorType An array that stores the contiguous RTD channel#s sensor type,
wSensorType[0] represents the value of all channel. The values for normal channels are:

0=PT100
1=PT200
2=PT500
3=PT1000
4=PT50
16=JPT100
17=JPT200
18=JPT500
19=JPT1000
32=NI100
33=NI200
34=NI500
35=NI1000

48=NI120
64=CU10
128=1~2000 Ohm, 100 mohm/1 count
129=1~327 Ohm, 10 mohm/1 count
130=1~620 Ohm, 20 mohm/1 count
Others return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_RTD_SetSensorType

This function code is used to set the Sensor Type for contiguous channels.

C/C++

```
int E42_RTD_SetSensorType( int hConnection,
                           BYTE bytSlot,
                           WORD wSensorType);
```

Visual Basic

```
Declare Function E42_RTD_SetSensorType Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, ByVal
iSensorType As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wSensorType	An array that stores the contiguous RTD channel's sensor type, wSensorType[0] represents the value of the starting channel. The values for normal channels are: 0=PT100 1=PT200 2=PT500 3=PT1000 4=PT50 16=JPT100 17=JPT200 18=JPT500 19=JPT1000 32=NI100

33=NI200
34=NI500
35=NI1000
48=NI120
64=CU10
128=1~2000 Ohm, 100 mohm/1 count
129=1~327 Ohm, 10 mohm/1 count
130=1~620 Ohm, 20 mohm/1 count
Others return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_RTD_Reads

This function code is used to read the temperature value for contiguous channels.

C/C++

```
int E1K_RTD_Reads( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    double dValue[ ]);
```

Visual Basic

```
Declare Function E1K_RTD_Reads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the temperature value to be read. The dValue[0] represents the start channel. When the dValue is 0x7FFF, it means the sensor is not wired correctly, or the measured value is out of range. When using the RTD module for Resistance Input, the unit is Ohm. When the operating mode is temperature sensor, the unit is C or F, depending on the setting. Check the ioAdmin utility for current settings.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_RTD_ReadRaws

This function code is used to read the temperature raw data for contiguous channels.

C/C++

```
int E1K_RTD_ReadRaws( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_RTD_ReadRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the temperature value to be read. The wValue[0] represents the start channel. When the wValue is 0x7FFF, it means the sensor is not wired correctly, or the measured value is out of range. When using the RTD module for Resistance Input 1~2000毫, 100m毫/1count. When using the RTD module for Resistance Input 1~327毫, 10m毫/1count. When using the RTD module for Resistance Input 1~620毫, 20m毫/1count. When the operating mode is temperature sensor, 0.1C(F)/1count, depending on the setting. Check the ioAdmin utility for current settings.

Return Value:

Succeed MXIO_OK
Fail Refer to Return Codes.

E1K_RTD_ResetMins

This function code is used to reset contiguous RTD channel~~◆◆◆~~s minimize value.

C/C++

```
int E1K_RTD_ResetMins( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount);
```

Visual Basic

```
Declare Function E1K_RTD_ResetMins Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_RTD_ResetMaxs

This function code is used to reset contiguous RTD channel's maximize value.

C/C++

```
int E1K_RTD_ResetMaxs( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount);
```

Visual Basic

```
Declare Function E1K_RTD_ResetMaxs Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_RTD_GetChannelStatuses

This function code is used to get contiguous channel#???'s status.

C/C++

```
int E1K_RTD_GetChannelStatuses( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD * dwStatus );
```

Visual Basic

```
Declare Function E1K_RTD_GetChannelStatuses Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous RTD channel#???'s status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel#???'s status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_RTD_SetChannelStatuses

This function code is used to set contiguous channel#???'s status.

C/C++

```
int E1K_RTD_SetChannelStatuses( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD dwStatus) ;
```

Visual Basic

```
Declare Function E1K_RTD_SetChannelStatuses Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nStatus As Long) As Long
```

Arguments:

Hconnection	The handle for an I/O device connection.
BytStartChannel	Specifies the starting channel.
BytCount	The number of channels to be set.
DwStatus	A pointer that stores the contiguous count channel#???'s status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel#???'s status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_RTD_GetEngUnits

This function code is used to get contiguous RTD channel~~???~~'s engineering unit.

C/C++

```
int E1K_RTD_GetEngUnits ( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wEngUnit[ ] );
```

Visual Basic

```
Declare Function E1K_RTD_GetEngUnits Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iEngUnit As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get.
wEngUnit	An array that stores the contiguous RTD channel ??? 's engineering unit, wEngUnit[0] represents the value of the starting channel. The values for normal channel are: 0=Celsius 1=Fahrenheit 2=Ohm Others_return Illegal Data Value

The values for virtual channels are:

0=Celsius
1=Fahrenheit
Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
---------	---------

Fail

Refer to Return Codes.

E1K_RTD_SetEngUnits

This function code is used to set contiguous RTD channel~~???~~'s engineering unit.

C/C++

```
int E1K_RTD_SetEngUnits ( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wEngUnit[ ] );
```

Visual Basic

```
Declare Function E1K_RTD_SetEngUnits Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iEngUnit As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wEngUnit	An array that stores the contiguous RTD channel ??? 's engineering unit, wEngUnit[0] represents the value of the starting channel. The values for normal channel are: 0=Celsius 1=Fahrenheit 2=Ohm Others_return Illegal Data Value

The values for virtual channels are:

0=Celsius
1=Fahrenheit
Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
---------	---------

Fail

Refer to Return Codes.

E1K_RTD_GetSensorTypes

This function code is used to get contiguous RTD channel~~???'s~~ sensor type.

C/C++

```
int E1K_RTD_GetSensorTypes ( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            WORD wSensorType[ ] );
```

Visual Basic

```
Declare Function E1K_RTD_GetSensorTypes Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iSensorType As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wSensorType	An array that stores the contiguous RTD channel ???'s sensor type, wSensorType[0] represents the value of the starting channel. The values for normal channels are: 0=PT50 1=PT100 2=PT200 3=PT500 4=PT1000 5=JPT100 (Reserve) 6=JPT200 (Reserve) 7=JPT500 (Reserve) 8=JPT1000 (Reserve) 9=NI100 (Reserve) 10=NI200 (Reserve)

11=NI500 (Reserve)
12=NI1000 (Reserve)
13=NI120 (Reserve)
14=310 Ohm
15=620 Ohm
16=1250 Ohm
17=2200 Ohm
Others return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_RTD_SetSensorTypes

This function code is used to set contiguous RTD channel~~???'s~~ sensor type.

C/C++

```
int E1K_RTD_SetSensorTypes ( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            WORD wSensorType[ ] );
```

Visual Basic

```
Declare Function E1K_RTD_SetSensorTypes Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iSensorType As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wSensorType	An array that stores the contiguous RTD channel ???'s sensor type, wSensorType[0] represents the value of the starting channel. The values for normal channels are: 0=PT50 1=PT100 2=PT200 3=PT500 4=PT1000 5=JPT100 (Reserve) 6=JPT200 (Reserve) 7=JPT500 (Reserve) 8=JPT1000 (Reserve) 9=NI100 (Reserve) 10=NI200 (Reserve)

11=NI500 (Reserve)
12=NI1000 (Reserve)
13=NI120 (Reserve)
14=310 Ohm
15=620 Ohm
16=1250 Ohm
17=2200 Ohm

Others return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_RTD_ReadMinRaw

This function code is used to read contiguous RTD channel~~???~~s minimize raw data.

C/C++

```
int E1K_RTD_ReadMinRaw( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_RTD_ReadMinRaw Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous RTD channel ??? s minimize raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_RTD_ReadMaxRaws

This function code is used to read contiguous RTD channel~~?????~~s maximize raw data.

C/C++

```
int E1K_RTD_ReadMaxRaws( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_RTD_ReadMaxRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous RTD channel ????? s maximize raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_RTD_ReadMins

This function code is used to read contiguous RTD channel~~?????~~s minimize value.

C/C++

```
int E1K_RTD_ReadMins( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] );
```

Visual Basic

```
Declare Function E1K_RTD_ReadMins Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous RTD channel ????? s minimize value , dValue[0] represents the value of the starting channel. The unit is Ohm for the Ohm, ????? for Celsius and ????? for Fahrenheit.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_RTD_ReadMaxs

This function code is used to read contiguous RTD channel~~?????~~s maximize value.

C/C++

```
int E1K_RTD_ReadMaxs( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] );
```

Visual Basic

```
Declare Function E1K_RTD_ReadMaxs Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous RTD channel ????? s minimize value , dValue[0] represents the value of the starting channel. The unit is Ohm for the Ohm, Celsius for Celsius and Fahrenheit for Fahrenheit.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC_Reads

This function code is used to read the temperature value of contiguous channels.

C/C++

```
int TC_Reads( int hConnection,
               BYTE bytSlot
               BYTE bytStartChannel,
               BYTE bytCount,
               double dValue[ ]);
```

Visual Basic

```
Declare Function TC_Reads Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
BytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the temperature value to read. The dValue[0] represents start channel 0. When the dValue is 0x7fffffff, it means the sensor is not correctly wired. When the operating mode of the TC module is voltage input, the unit is mV . Use ioAdmin to check the I/O module settings.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC_Read

This function code is used to read the temperature value for a specific channel.

C/C++

```
int TC_Read( int hConnection,
              BYTE bytSlot,
              BYTE bytChannel,
              double * dValue);
```

Visual Basic

```
Declare Function TC_Read Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be read.
dValue	Stores the specific channel's temperature value to be read. When the dValue is 0xffffffff, it means the sensor is not wired correctly. When the operating mode of the TC module is voltage input, the unit is °v. Use ioAdmin to check the I/O module settings.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC_ReadRaw

This function code is used to read the temperature raw data of contiguous channels.

C/C++

```
int TC_ReadRaw( int hConnection,
                 BYTE bytSlot,
                 BYTE bytStartChannel,
                 BYTE bytCount,
                 DWORD wValue[ ] );
```

Visual Basic

```
Declare Function TC_ReadRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the temperature value to read. The wValue[0] represents start channel 0. When the wValue is 0x7fffffff, it means the sensor is not correctly wired. When the operating mode is temperature sensor, 0.1????(????). When the operating mode of the TC module is -78.0 ~ 78.0 mV, 10uV/count. When the operating mode of the TC module is -32.7 ~ 32.7 mV, 1uV/count. When the operating mode of the TC module is -65.5 ~ 65.5 mV, 2uV/count. Use ioAdmin to check the I/O module settings.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC_ReadRaw

This function code is used to read the temperature raw data for a specific channel.

C/C++

```
int TC_ReadRaw( int hConnection,
                 BYTE bytSlot,
                 BYTE bytChannel,
                 DWORD * wValue);
```

Visual Basic

```
Declare Function TC_ReadRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytChannel As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 32. But this parameter is inactive in ioLogik 2000.
bytChannel	The specific channel to be read.
wValue	An pointer that stores the temperature value to read. When the wValue is 0x7fffffff, it means the sensor is not correctly wired. When the operating mode is temperature sensor, 0.1 ⁰⁰⁰ (⁰⁰⁰). When the operating mode of the TC module is -78.0 ~ 78.0 mV, 10uV/count. When the operating mode of the TC module is -32.7 ~ 32.7 mV, 1uV/count. When the operating mode of the TC module is -65.5 ~ 65.5 mV, 2uV/count. Use ioAdmin to check the I/O module settings.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_ResetMin

This function code is used to reset the TC input minimize value for a specific channel.

C/C++

```
int TC2K_ResetMin ( int hConnection,  
                     BYTE bytChannel);
```

Visual Basic

```
Declare Function TC2K_ResetMin Lib "MXIO.dll" (ByVal  
hConnection As Long, ByVal bytChannel As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_ResetMins

This function code is used to reset contiguous TC channel's minimize value.

C/C++

```
int TC2K_ResetMins( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount) ;
```

Visual Basic

```
Declare Function TC2K_ResetMins Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_ResetMax

This function code is used to reset the TC input maximize value for a specific channel.

C/C++

```
int TC2K_ResetMax( int hConnection,  
                    BYTE bytChannel);
```

Visual Basic

```
Declare Function TC2K_ResetMax Lib "MXIO.dll" (ByVal  
hConnection As Long, ByVal bytChannel As Byte) As Long
```

Arguments:

<code>hConnection</code>	The handle for an I/O device connection.
<code>bytChannel</code>	The specific channel to be reset.

Return Value:

Succeed	<code>MXIO_OK</code>
Fail	Refer to Return Codes.

TC2K_ResetMaxs

This function code is used to reset contiguous TC channel's maximize value.

C/C++

```
int TC2K_ResetMaxs( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount) ;
```

Visual Basic

```
Declare Function TC2K_ResetMaxs Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_ReadMinRaw

This function code is used to read the TC input minimize raw data for a specific channel.

C/C++

```
int TC2K_ReadMinRaw( int hConnection,
                      BYTE bytChannel,
                      DWORD * dwValue);
```

Visual Basic

```
Declare Function TC2K_ReadMinRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, nValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be read.
dwValue	An point that stores the specific TC channel's minimize raw data.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_ReadMinRaws

This function code is used to read contiguous TC channel's minimize raw data.

C/C++

```
int TC2K_ReadMinRaws( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD wValue[ ] );
```

Visual Basic

```
Declare Function TC2K_ReadMinRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous TC channel's minimize raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_ReadMaxRaw

This function code is used to read the TC input maximize raw data for a specific channel.

C/C++

```
int TC2K_ReadMaxRaw( int hConnection,
                      BYTE bytChannel,
                      DWORD * wValue);
```

Visual Basic

```
Declare Function TC2K_ReadMaxRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be read.
wValue	An point that stores the specific TC channel's maximize raw data.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_ReadMaxRaws

This function code is used to read contiguous TC channel's maximize raw data.

C/C++

```
int TC2K_ReadMaxRaws( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD dwValue[ ] );
```

Visual Basic

```
Declare Function TC2K_ReadMaxRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	An array that stores the contiguous TC channel's maximize raw data , dwValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_ReadMin

This function code is used to read the TC input minimize value for a specific channel.

C/C++

```
int TC2K_ReadMin( int hConnection,
                   BYTE bytChannel,
                   double *dValue);
```

Visual Basic

```
Declare Function TC2K_ReadMin Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, dValue As
Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be read.
dwValue	A pointer that stores the specific channel TC input minimize value to be read. The unit is mV for the Millivolt, ????? for Celsius and ????? for Fahrenheit. Burn Out (0x7fffffff)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_ReadMins

This function code is used to read contiguous TC channel's minimize value.

C/C++

```
int TC2K_ReadMins( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    double dValue[ ] );
```

Visual Basic

```
Declare Function TC2K_ReadMins Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	An array that stores the contiguous TC channel's minimize value , dwValue[0] represents the value of the starting channel. The unit is mV for the Millivolt, ????? for Celsius and ????? for Fahrenheit. Burn Out (0x7fffffff)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_ReadMax

This function code is used to read the TC input maximize value for a specific channel.

C/C++

```
int TC2K_ReadMax( int hConnection,
                   BYTE bytChannel,
                   double *dValue);
```

Visual Basic

```
Declare Function TC2K_ReadMax Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, dValue As
Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be read.
dwValue	A pointer that stores the specific channel TC input minimize value to be read. The unit is mV for the Millivolt, ????? for Celsius and ????? for Fahrenheit. Burn Out (0x7fffffff)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_ReadMaxs

This function code is used to read contiguous TC channel's maximize value.

C/C++

```
int TC2K_ReadMaxs( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    double dValue[ ] );
```

Visual Basic

```
Declare Function TC2K_ReadMaxs Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	An array that stores the contiguous TC channel's minimize value , dwValue[0] represents the value of the starting channel. The unit is mV for the Millivolt, ????? for Celsius and ????? for Fahrenheit. Burn Out (0x7fffffff)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_ReadRaw

This function code is used to read the TC input raw data for a specific channel.

C/C++

```
int TC2K_ReadRaw( int hConnection,
                   BYTE bytChannel,
                   DWORD * dwValue);
```

Visual Basic

```
Declare Function TC2K_ReadRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be read.
dwValue	An point that stores the specific TC channel's raw data.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_ReadRaws

This function code is used to read contiguous TC channel's raw data.

C/C++

```
int TC2K_ReadRaws( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    DWORD wValue[ ] );
```

Visual Basic

```
Declare Function TC2K_ReadRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous TC channel's raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_GetChannelStatus

This function code is used to get specific channel's status.

C/C++

```
int TC2K_GetChannelStatus( int hConnection,
                           BYTE bytChannel,
                           BYTE * bytStatus);
```

Visual Basic

```
Declare Function TC2K_GetChannelStatus Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytChannel As Byte,
     bytStatus As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
bytStatus	A pointer that stores the specific TC channel's status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_SetChannelStatus

This function code is used to set specific channel's status.

C/C++

```
int TC2K_SetChannelStatus( int hConnection,
                           BYTE bytChannel,
                           BYTE bytStatus);
```

Visual Basic

```
Declare Function TC2K_SetChannelStatus Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytChannel As Byte, ByVal
bytStatus As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
bytStatus	A pointer that stores the specific TC channel's status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_GetChannelStatuses

This function code is used to get contiguous channel's status.

C/C++

```
int TC2K_GetChannelStatuses( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD * dwStatus) ;
```

Visual Basic

```
Declare Function TC2K_GetChannelStatuses Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, dwStatus As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous TC channel's status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_SetChannelStatuses

This function code is used to set contiguous channel's status.

C/C++

```
int TC2K_SetChannelStatuses( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwStatus);
```

Visual Basic

```
Declare Function TC2K_SetChannelStatuses Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal dwStatus As Double) As Long
```

Arguments:

Hconnection	The handle for an I/O device connection.
BytStartChannel	Specifies the starting channel.
BytCount	The number of channels to be set.
DwStatus	A pointer that stores the contiguous count channel's status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_GetSensorType

This function code is used to get the sensor type for a specific TC channel.

C/C++

```
int TC2K_GetSensorType( int hConnection,
                        BYTE bytChannel,
                        WORD * wSensorType );
```

Visual Basic

```
Declare Function TC2K_GetSensorType Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iSensorType
As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get.
wSensorType	A pointer that stores the specific TC channel's sensor type. The values for normal channels are: 0=J Type 1=K Type 2=T Type 3=E Type 4=R Type 5=S Type 6=B Type 7=N Type 8=Voltage 78.126mV 9=Voltage 39.062mV 10=Voltage 19.532mV Others return Illegal Data Value
	The values for virtual channels are: 20=AVG

21=DIV

Others return Illegal Data Value

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

TC2K_SetSensorType

This function code is used to set the sensor type for a specific TC channel.

C/C++

```
int TC2K_SetSensorType( int hConnection,
                        BYTE bytChannel,
                        WORD wSensorType);
```

Visual Basic

```
Declare Function TC2K_SetSensorType Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal
iSensorType As integer) As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytChannel The specific channel to be set.

wSensorType A pointer that stores the specific TC channel's sensor type. The values for normal channels are:

0=J Type

1=K Type

2=T Type

3=E Type

4=R Type

5=S Type

6=B Type

7=N Type

8=Voltage 78.126mV

9=Voltage 39.062mV

10=Voltage 19.532mV

Others return Illegal Data Value

The values for virtual channels are:

20=AVG

21=DIV

Others return Illegal Data Value

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

TC2K_GetSensorTypes

This function code is used to get contiguous TC channel's sensor type.

C/C++

```
int TC2K_GetSensorTypes ( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wSensorType[ ] );
```

Visual Basic

```
Declare Function TC2K_GetSensorTypes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iSensorType As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get.
wSensorType	An array that stores the contiguous TC channel's sensor type, wSensorType[0] represents the value of the starting channel. The values for normal channel are: 0=J Type 1=K Type 2=T Type 3=E Type 4=R Type 5=S Type 6=B Type 7=N Type 8=Voltage 78.126mV 9=Voltage 39.062mV 10=Voltage 19.532mV Others return Illegal Data Value

The values for virtual channels are:

20=AVG

21=DIV

Others return Illegal Data Value

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

TC2K_SetSensorTypes

This function code is used to set contiguous TC channel's sensor type.

C/C++

```
int TC2K_SetSensorTypes ( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wSensorType[ ] );
```

Visual Basic

```
Declare Function TC2K_SetSensorTypes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iSensorType As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wSensorType	An array that stores the contiguous TC channel's sensor type, wSensorType[0] represents the value of the starting channel. The values for normal channel are: 0=J Type 1=K Type 2=T Type 3=E Type 4=R Type 5=S Type 6=B Type 7=N Type 8=Voltage 78.126mV 9=Voltage 39.062mV 10=Voltage 19.532mV Others return Illegal Data Value

The values for virtual channels are:

20=AVG

21=DIV

Others return Illegal Data Value

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

TC2K_GetEngUnit

This function code is used to get the engineering unit for a specific TC channel.

C/C++

```
int TC2K_GetEngUnit ( int hConnection,  
                      BYTE bytChannel,  
                      WORD * wEngUnit );
```

Visual Basic

```
Declare Function TC2K_GetEngUnit Lib "MXIO.dll" (ByVal  
hConnection As Long, ByVal bytChannel As Byte, iEngUnit As  
integer) As Long
```

Arguments :

hConnection The handle for an I/O device connection.

bytChannel The specific channel to be get.

wEngUnit A pointer that stores the specific TC channel's engineering unit. The values for normal channels are:

0=Celsius

1=Fahrenheit

2=Millivolt

Others_return Illegal Data Value

The values for virtual channels are:

0=Celsius

1=Fahrenheit

Others_return Illegal Data Value

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

TC2K_SetEngUnit

This function code is used to set the engineering unit for a specific TC channel.

C/C++

```
int TC2K_SetEngUnit ( int hConnection,  
                      BYTE bytChannel,  
                      WORD wEngUnit);
```

Visual Basic

```
Declare Function TC2K_SetEngUnit Lib "MXIO.dll" (ByVal  
hConnection As Long, ByVal bytChannel As Byte, ByVal  
iEngUnit As integer) As Long
```

Arguments :

hConnection The handle for an I/O device connection.

`bytChannel` The specific channel to be set.

wEngUnit A pointer that stores the specific TC channel's engineering unit. The values for normal channels are:

0=Celsius

1=Fahrenheit

2=Millivolt

Others_return Illegal Data Value

The values for virtual channels are:

0=Celsius

1=Fahrenheit

Others_return Illegal Data Value

Return Value:

Succeed MXIO OK

Fail Refer to Return Codes.

TC2K_GetEngUnits

This function code is used to get contiguous TC channel's engineering unit.

C/C++

```
int TC2K_GetEngUnits ( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wEngUnit[ ] );
```

Visual Basic

```
Declare Function TC2K_GetEngUnits Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iEngUnit As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wEngUnit	An array that stores the contiguous TC channel's engineering unit, wEngUnit[0] represents the value of the starting channel. The values for normal channel are: 0=Celsius 1=Fahrenheit 2=Millivolt Others_return Illegal Data Value
	The values for virtual channels are: 0=Celsius 1=Fahrenheit Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
---------	---------

Fail

Refer to Return Codes.

TC2K_SetEngUnits

This function code is used to set contiguous TC channel's engineering unit.

C/C++

```
int TC2K_SetEngUnits ( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wEngUnit[ ] );
```

Visual Basic

```
Declare Function TC2K_SetEngUnits Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iEngUnit As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wEngUnit	An array that stores the contiguous TC channel's engineering unit, wEngUnit[0] represents the value of the starting channel. The values for normal channel are: 0=Celsius 1=Fahrenheit 2=Millivolt Others_return Illegal Data Value

The values for virtual channels are:

0=Celsius
1=Fahrenheit
Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
---------	---------

Fail

Refer to Return Codes.

TC2K_GetMathPar

This function code is used to get the math parameter for a specific TC virtual channel.

C/C++

```
int TC2K_GetMathPar ( int hConnection,
                      BYTE bytChannel,
                      WORD * wMathPar );
```

Visual Basic

```
Declare Function TC2K_GetMathPar Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iMathPar As
integer) As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytChannel The specific channel to be get.

wMathPar A pointer that stores the specific TC virtual channel's math parameter. For AVG, Bit 0 of high byte represents the first channel and Bit 1 of high byte represents the second channel. For DEV, the High-Byte as subtrahend and Low-Byte as minuend.

Exp AVG(b'0000-0000 b'0010-0011) =
ch5+ch1+ch0

Exp DEV(b'0000-0100 b'0010-0000) = **ch2-**
ch6

Return Value:

Succeed **MXIO_OK**

Fail Refer to Return Codes.

TC2K_SetMathPar

This function code is used to set the math parameter for a specific TC virtual channel.

C/C++

```
int TC2K_SetMathPar ( int hConnection,
                      BYTE bytChannel,
                      WORD wMathPar);
```

Visual Basic

```
Declare Function TC2K_SetMathPar Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal
iMathPar As integer) As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytChannel The specific channel to be set.

wMathPar A pointer that stores the specific TC virtual channel's math parameter. For AVG, Bit 0 of high byte represents the first channel and Bit 1 of high byte represents the second channel. For DEV, the High-Byte as subtrahend and Low-Byte as minuend.

Exp AVG(b'0000-0000 b'0010-0011) = ch5+ch1+ch0

Exp DEV(b'0000-0100 b'0010-0000) = ch2-ch6

Return Value:

Succeed **MXIO_OK**

Fail Refer to Return Codes.

TC2K_GetMathPars

This function code is used to get contiguous TC virtual channel's math parameter.

C/C++

```
int TC2K_GetMathPars ( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wMathPar[ ] );
```

Visual Basic

```
Declare Function TC2K_GetMathPars Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMathPar As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wMathPar	An array that stores the contiguous TC virtual channel's math parameter, wMathPar[0] represents the value of the starting channel. The values are : For AVG, Bit 0 of high byte represents the first channel and Bit 1 of high byte represents the second channel. For DEV, High-Byte as subtrahend and Low-Byte as minuend. Exp AVG(b'0000-0000 b'0010-0011) = ch5+ch1+ch0 Exp DEV(b'0000-0100 b'0010-0000) = ch2-ch6

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_SetMathPars

This function code is used to set contiguous TC virtual channel's math parameter.

C/C++

```
int TC2K_SetMathPars ( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wMathPar[ ] );
```

Visual Basic

```
Declare Function TC2K_SetMathPars Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMathPar As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wMathPar	An array that stores the contiguous TC virtual channel's math parameter, wMathPar[0] represents the value of the starting channel. The values are : For AVG, Bit 0 of high byte represents the first channel and Bit 1 of high byte represents the second channel. For DEV, High-Byte as subtrahend and Low-Byte as minuend. Exp AVG(b'0000-0000 b'0010-0011) = ch5+ch1+ch0 Exp DEV(b'0000-0100 b'0010-0000) = ch2-ch6

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_SetChnAvg

This function is used to set data structure variables for averaging data of each channel.

C/C++

```
int TC2K_SetChnAvg ( int hConnection,
                      BYTE bytChannel,
                      BYTE bytChnIdx[ ],
                      BYTE bytChCount );
```

Visual Basic

```
Declare Function TC2K_SetChnAvg Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, bytChnIdx As
Byte, ByVal bytChCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
bytChnIdx	An channel index which is used to be averaged.
BytChCount	The number of channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

TC2K_SetChnDev

This function is used to set data structure variables for subtracting data of two channels.

C/C++

```
int TC2K_SetChnDev ( int hConnection,
                      BYTE bytChannel,
                      BYTE bytChMinued,
                      BYTE bytChSub) ;
```

Visual Basic

```
Declare Function TC2K_SetChnDev Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, ByVal
bytChMinued As Byte, ByVal bytChSub As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be set.
bytChMinued	This variable is the channel that defined as a subtrahend.
BytChSub	This variable is the channel that defined as a minuend.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_TC_Reads

This function code is used to read the temperature value of contiguous channels.

C/C++

```
int E42_TC_Reads( int hConnection,
                   BYTE bytSlot
                   BYTE bytStartChannel,
                   BYTE bytCount,
                   double dValue[ ] );
```

Visual Basic

```
Declare Function E42_TC_Reads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
BytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the temperature value to read. The dValue[0] represents start channel 0. When the dValue is 0x8000, it means the sensor is not correctly wired. When the operating mode of the TC module is voltage input, the unit is <i>毫v</i> . Use ioAdmin to check the I/O module settings.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_TC_ReadRaw

This function code is used to read the temperature raw data of contiguous channels.

C/C++

```
int E42_TC_ReadRaw( int hConnection,
                     BYTE bytSlot,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     DWORD wValue[ ] );
```

Visual Basic

```
Declare Function E42_TC_ReadRaw Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte,
iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive in ioLogik 2000.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the temperature value to read. The wValue[0] represents start channel 0. When the wValue is 0x8000, it means the sensor is not correctly wired. When the operating mode is temperature sensor, 0.1????(????). When the operating mode of the TC module is -78.0 ~ 78.0 mV, 10uV/count. When the operating mode of the TC module is -32.7 ~ 32.7 mV, 1uV/count. When the operating mode of the TC module is -65.5 ~ 65.5 mV, 2uV/count. Use ioAdmin to check the I/O module settings.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_TC_GetEngUnit

This function code is used to get the temperature Type for contiguous channels.

C/C++

```
int E42_TC_GetEngUnit( int hConnection,
                        BYTE bytSlot,
                        WORD wEngUnit[ ] );
```

Visual Basic

```
Declare Function E42_TC_GetEngUnits Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iEngUnit
As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16.
wEngUnit	An array that stores the contiguous A/O channel#?#?#s safe action to be get. The wEngUnit[0] represents the value of all channel. The values are: 0: Celsius 1: Fahrenheit

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_TC_SetEngUnit

This function code is used to set the temperature Type for contiguous channels.

C/C++

```
int E42_TC_SetEngUnit( int hConnection,
                        BYTE bytSlot,
                        WORD wEngUnit);
```

Visual Basic

```
Declare Function E42_TC_SetEngUnit Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal iEngUnit
As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16.
wEngUnit	An array that stores the contiguous A/O channel's safe action to be set. The wEngUnit[0] represents the value of allchannel. The values are: 0: Celsius 1: Fahrenheit

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_TC_GetSensorType

This function code is used to get the Sensor Type for contiguous channels.

C/C++

```
int E42_TC_GetSensorType( int hConnection,
                           BYTE bytSlot,
                           WORD wSensorType[ ] );
```

Visual Basic

```
Declare Function E42_TC_GetSensorType Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, iSensorType As
Integer) As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytSlot Slot number of the I/O module. The Slot number ranges from 1 to 16.

wSensorType An array that stores the contiguous TC channel???s sensor type,
wSensorType[0] represents the value of the all channel. The values for all channels are:

0=TYPE K

1=TYPE J

2=TYPE T

3=TYPE B

4=TYPE R

5=TYPE S

6=TYPE E

7=TYPE N

8=TYPE L

9=TYPE U

10=TYPE C

11=TYPE D

128=10uV Input, -78mV~78mV, 10uV/count

129=1uV Input, -32.7mV~32.7mV, 1uV/count
130=2uV Input, -65.5mV~65.5mV, 2uV/count
Others return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E42_TC_SetSensorType

This function code is used to set the Sensor Type for contiguous channels.

C/C++

```
int E42_TC_SetSensorType( int hConnection,
                           BYTE bytSlot,
                           WORD wSensorType);
```

Visual Basic

```
Declare Function E42_TC_SetSensorType Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytSlot As Byte, ByVal
iSensorType As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytSlot	Slot number of the I/O module. The Slot number ranges from 1 to 16.
wSensorType	An array that stores the contiguous TC channel's sensor type, wSensorType[0] represents the value of all channel. The values for all channels are: 0=TYPE K 1=TYPE J 2=TYPE T 3=TYPE B 4=TYPE R 5=TYPE S 6=TYPE E 7=TYPE N 8=TYPE L 9=TYPE U 10=TYPE C 11=TYPE D 128=10uV Input, -78mV~78mV, 10uV/count

129=1uV Input, -32.7mV~32.7mV, 1uV/count
130=2uV Input, -65.5mV~65.5mV, 2uV/count
Others return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_TC_Reads

This function code is used to read the temperature value of contiguous channels.

C/C++

```
int E1K_TC_Reads( int hConnection,
                   BYTE bytStartChannel,
                   BYTE bytCount,
                   double dValue[ ] );
```

Visual Basic

```
Declare Function E1K_TC_Reads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the temperature value to read. The dValue[0] represents start channel 0. When the dValue is 0x7fffffff, it means the sensor is not correctly wired. When the operating mode of the TC module is voltage input, the unit is mv . Use ioAdmin to check the I/O module settings.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_TC_ReadRaws

This function code is used to read contiguous TC channel~~???~~s raw data.

C/C++

```
int E1K_TC_ReadRaws( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      DWORD wValue[ ]);
```

Visual Basic

```
Declare Function E1K_TC_ReadRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous TC channel ??? s raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_TC_ResetMins

This function code is used to reset contiguous TC channel **???**'s minimize value.

C/C++

```
int E1K_TC_ResetMins( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount);
```

Visual Basic

```
Declare Function E1K_TC_ResetMins Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_TC_ResetMaxs

This function code is used to reset contiguous TC channel **???'s** maximize value.

C/C++

```
int E1K_TC_ResetMaxs( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount);
```

Visual Basic

```
Declare Function E1K_TC_ResetMaxs Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_TC_GetChannelStatuses

This function code is used to get contiguous channel**?????**s status.

C/C++

```
int E1K_TC_GetChannelStatuses( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD * dwStatus);
```

Visual Basic

```
Declare Function E1K_TC_GetChannelStatuses Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, dwStatus As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous TC channel ????? s status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel ????? s status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_TC_SetChannelStatuses

This function code is used to set contiguous channel#???'s status.

C/C++

```
int E1K_TC_SetChannelStatuses( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD dwStatus) ;
```

Visual Basic

```
Declare Function E1K_TC_SetChannelStatuses Lib MXIO.dll
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal dwStatus As Double) As Long
```

Arguments:

Hconnection	The handle for an I/O device connection.
BytStartChannel	Specifies the starting channel.
BytCount	The number of channels to be set.
DwStatus	A pointer that stores the contiguous count channel#???'s status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel#???'s status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_TC_GetEngUnits

This function code is used to get contiguous TC channel~~?????~~s engineering unit.

C/C++

```
int E1K_TC_GetEngUnits ( int hConnection,
                         BYTE bytStartChannel,
                         BYTE bytCount,
                         WORD wEngUnit[ ] );
```

Visual Basic

```
Declare Function E1K_TC_GetEngUnits Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iEngUnit As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wEngUnit	An array that stores the contiguous TC channel ????? s engineering unit, wEngUnit[0] represents the value of the starting channel. The values for normal channel are: 0=Celsius 1=Fahrenheit 2=Millivolt Others_return Illegal Data Value
	The values for virtual channels are: 0=Celsius 1=Fahrenheit Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
---------	---------

Fail

Refer to Return Codes.

E1K_TC_SetEngUnits

This function code is used to set contiguous TC channel#???'s engineering unit.

C/C++

```
int E1K_TC_SetEngUnits ( int hConnection,
                         BYTE bytStartChannel,
                         BYTE bytCount,
                         WORD wEngUnit[ ] );
```

Visual Basic

```
Declare Function E1K_TC_SetEngUnits Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iEngUnit As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wEngUnit	An array that stores the contiguous TC channel#???'s engineering unit, wEngUnit[0] represents the value of the starting channel. The values for normal channel are: 0=Celsius 1=Fahrenheit 2=Millivolt Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_TC_GetSensorTypes

This function code is used to get contiguous TC channel~~?????~~s sensor type.

C/C++

```
int E1K_TC_GetSensorTypes ( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            WORD wSensorType[ ] );
```

Visual Basic

```
Declare Function E1K_TC_GetSensorTypes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iSensorType As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get.
wSensorType	An array that stores the contiguous TC channel ????? s sensor type, wSensorType[0] represents the value of the starting channel. The values for normal channel are: 0=J Type 1=K Type 2=T Type 3=E Type 4=R Type 5=S Type 6=B Type 7=N Type 8=Voltage 78.126mV 9=Voltage 39.062mV 10=Voltage 19.532mV

Others return Illegal Data Value

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

E1K_TC_SetSensorTypes

This function code is used to set contiguous TC channel~~?????~~s sensor type.

C/C++

```
int E1K_TC_SetSensorTypes ( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            WORD wSensorType[ ] );
```

Visual Basic

```
Declare Function E1K_TC_SetSensorTypes Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iSensorType As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wSensorType	An array that stores the contiguous TC channel ????? s sensor type, wSensorType[0] represents the value of the starting channel. The values for normal channel are: 0=J Type 1=K Type 2=T Type 3=E Type 4=R Type 5=S Type 6=B Type 7=N Type 8=Voltage 78.126mV 9=Voltage 39.062mV 10=Voltage 19.532mV

Others return Illegal Data Value

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

E1K_TC_ReadMinRaw

This function code is used to read contiguous TC channel~~???~~s minimize raw data.

C/C++

```
int E1K_TC_ReadMinRaw( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        DWORD wValue[ ] );
```

Visual Basic

```
Declare Function E1K_TC_ReadMinRaw Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous TC channel ??? s minimize raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_TC_ReadMaxRaws

This function code is used to read contiguous TC channel~~?????~~s maximize raw data.

C/C++

```
int E1K_TC_ReadMaxRaws( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        DWORD dwValue[ ] );
```

Visual Basic

```
Declare Function E1K_TC_ReadMaxRaws Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	An array that stores the contiguous TC channel ????? s maximize raw data , dwValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_TC_ReadMins

This function code is used to read contiguous TC channel~~?????~~s minimize value.

C/C++

```
int E1K_TC_ReadMins( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] );
```

Visual Basic

```
Declare Function E1K_TC_ReadMins Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	An array that stores the contiguous TC channel ????? s minimize value , dwValue[0] represents the value of the starting channel. The unit is mV for the Millivolt, ????? for Celsius and ????? for Fahrenheit.
	Burn Out (0x7fffffff)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_TC_ReadMaxs

This function code is used to read contiguous TC channel~~?????~~s maximize value.

C/C++

```
int E1K_TC_ReadMaxs( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] );
```

Visual Basic

```
Declare Function E1K_TC_ReadMaxs Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	An array that stores the contiguous TC channel ????? s minimize value , dwValue[0] represents the value of the starting channel. The unit is mV for the Millivolt, ????? for Celsius and ????? for Fahrenheit.
	Burn Out (0x7fffffff)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RLY2K_GetResetTime

This function code is used to get reset time of D/O channels.

C/C++

```
int RLY2K_GetResetTime ( int hConnection,
                         BYTE bytChannel,
                         WORD wValue[ ] );
```

Visual Basic

```
Declare Function RLY2K_GetResetTime Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get
wValue	An array that stores the contiguous D/O channels relay reset time. wValue[0] ~ wValue[5] => sec/min/hour/day/month/year represents the value of the specific channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RLY2K_TotalCntRead

This function code is used to get count value of contiguous D/O channel.

C/C++

```
int RLY2K_TotalCntRead ( int hConnection,
                         BYTE bytChannel,
                         DWORD * dwValue );
```

Visual Basic

```
Declare Function RLY2K_TotalCntRead Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, nValue As
long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get
dwValue	A pointer that stores the count value of contiguous D/O channel

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RLY2K_TotalCntReads

This function code is used to get count value of contiguous D/O channels.

C/C++

```
int RLY2K_TotalCntReads ( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwValue[ ] );
```

Visual Basic

```
Declare Function RLY2K_TotalCntReads Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	An array that stores the contiguous D/O channels relay count. wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RLY2K_CurrentCntRead

This function code is used to get count value of contiguous D/O channel.

C/C++

```
int RLY2K_CurrentCntRead ( int hConnection,
                           BYTE bytChannel,
                           DWORD dwValue[] );
```

Visual Basic

```
Declare Function RLY2K_CurrentCntRead Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte, nValue As
long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get
dwValue	A pointer that stores the count value of contiguous D/O channel

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RLY2K_CurrentCntReads

This function code is used to get count value of contiguous D/O channels.

C/C++

```
int RLY2K_CurrentCntReads ( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD dwValue[ ] );
```

Visual Basic

```
Declare Function RLY2K_CurrentCntReads Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, nValue As long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	An array that stores the contiguous D/O channels relay count. wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RLY2K_ResetCnt

This function code is used to reset count value of contiguous D/O channel.

C/C++

```
int RLY2K_ResetCnt ( int hConnection,
                      BYTE bytChannel );
```

Visual Basic

```
Declare Function RLY2K_ResetCnt Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytChannel As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be reset

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

RLY2K_ResetCnts

This function code is used to reset count value of contiguous D/O channels.

C/C++

```
int RLY2K_ResetCnts ( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount);
```

Visual Basic

```
Declare Function RLY2K_ResetCnts Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_RLY_GetResetTime

This function code is used to get reset time of D/O channels.

C/C++

```
int W5K_RLY_GetResetTime ( int hConnection,
                           BYTE bytChannel,
                           WORD wValue[ ] );
```

Visual Basic

```
Declare Function W5K_RLY_GetResetTime Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytChannel As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytChannel	The specific channel to be get
wValue	An array that stores the contiguous D/O channels relay reset time. wValue[0] ~ wValue[5] => sec/min/hour/day/month/year represents the value of the specific channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_RLY_TotalCntReads

This function code is used to get count value of contiguous D/O channels.

C/C++

```
int W5K_RLY_TotalCntReads ( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD dwValue[ ] );
```

Visual Basic

```
Declare Function W5K_RLY_TotalCntReads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	An array that stores the contiguous D/O channels relay count. wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_RLY_CurrentCntReads

This function code is used to get count value of contiguous D/O channels.

C/C++

```
int W5K_RLY_CurrentCntReads ( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD dwValue[ ] );
```

Visual Basic

```
Declare Function W5K_RLY_CurrentCntReads Lib MXIO.dll
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, nValue As long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	An array that stores the contiguous D/O channels relay count. wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_RLY_ResetCnts

This function code is used to reset count value of contiguous D/O channels.

C/C++

```
int W5K_RLY_ResetCnts ( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount);
```

Visual Basic

```
Declare Function W5K_RLY_ResetCnts Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

E1K_RLY_TotalCntReads

This function code is used to get count value of contiguous D/O channels.

C/C++

```
int E1K_RLY_TotalCntReads ( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD dwValue[ ] );
```

Visual Basic

```
Declare Function E1K_RLY_TotalCntReads Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	An array that stores the contiguous D/O channels relay count. wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DIO_GetIOModes

This function code is used to get contiguous channel's DI/DO mode.

C/C++

```
int R1K_DIO_GetIOModes ( int hConnection,
                         BYTE bytStartChannel,
                         BYTE bytCount,
                         DWORD * dwMode );
```

Visual Basic

```
Declare Function R1K_DIO_GetIOModes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nMode As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous channel's DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: INPUT DI mode 1: OUTPUT DO mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DI_Reads

This function code is used to read the status of a group of contiguous D/I channels.

C/C++

```
int R1K_DI_Reads( int hConnection,
                   BYTE bytStartChannel,
                   BYTE bytCount,
                   DWORD * dwValue) ;
```

Visual Basic

```
Declare Function R1K_DI_Reads Lib "MXIO.dll" (ByVal
hConnection As Long,
ByVal bytStartChannel As Byte, ByVal bytCount As Byte,
nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
wCount	The number of channels to be read.
dwValue	A pointer that stores the contiguous D/I channel's values; each bit holds one channel value. A bit value of 0 represents the digital input status of the start channel. A bit value of 1 represents the second digital input channel's status. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DI_GetModes

This function code is used to get the mode of contiguous D/I channels.

C/C++

```
int R1K_DI_GetModes(int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     WORD wMode[ ]);
```

Visual Basic

```
Declare Function R1K_DI_GetModes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, wMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wMode	An array that stores contiguous D/I channel's mode. wMode[0] represents the value of the starting channel. The values are: 0: D/I Mode 1: Count Mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DI_SetModes

This function code is used to set the mode of contiguous D/I channels.

C/C++

```
int R1K_DI_SetModes( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wMode[ ] );
```

Visual Basic

```
Declare Function R1K_DI_SetModes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be sets.
wMode	An array that stores contiguous D/I channel's mode. wMode[0] represents the value of the starting channel. The values are: 0: D/I Mode 1: Count Mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DI_GetFilters

This function code is used to get the filter of contiguous D/I channels.

C/C++

```
int R1K_DI_GetFilters( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wFilter[ ] );
```

Visual Basic

```
Declare Function R1K_DI_GetFilters Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wFilter	An array that stores contiguous D/I channel's filter value. wFilter[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DI_SetFilters

This function code is used to set the filter of contiguous D/I channels.

C/C++

```
int R1K_DI_SetFilters( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wFilter[ ] );
```

Visual Basic

```
Declare Function R1K_DI_SetFilters Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be sets.
wFilter	An array that stores contiguous D/I channel's filter value. wFilter[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_Reads

This function code is used to read contiguous channel's count value when D/I channels in 'Count' mode.

C/C++

```
int R1K_Cnt_Reads( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount,
                     DWORD dwValue[ ] );
```

Visual Basic

```
Declare Function R1K_Cnt_Reads Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	An array that stores the contiguous channel's count value , dwValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_Clears

This function code is used to clear contiguous channel's count value when D/I channel in 'Count' mode.

C/C++

```
int R1K_Cnt_Clears( int hConnection,
                     BYTE bytStartChannel,
                     BYTE bytCount);
```

Visual Basic

```
Declare Function R1K_Cnt_Clears Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device's connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_GetOverflows

This function code is used to get contiguous channel's overflow status when D/I channel in 'Count' mode.

C/C++

```
int R1K_Cnt_GetOverflows( int hConnection,
                           BYTE bytStartChannel
                           BYTE bytCount
                           DWORD * dwStatus) ;
```

Visual Basic

```
Declare Function R1K_Cnt_GetOverflows Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
dwStatus	A pointer that stores the contiguous channel's overflow status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: Normal 1: Overflow

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_ClearOverflows

This function code is used to clear contiguous channel's overflow status when D/I channel in 'Count' mode.

C/C++

```
int R1K_Cnt_ClearOverflows( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount);
```

Visual Basic

```
Declare Function R1K_Cnt_ClearOverflows Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_GetFilters

This function code is used to get contiguous channel's filter value when D/I channel in 'Count' mode.

C/C++

```
int R1K_Cnt_GetFilters( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wFilter[ ]);
```

Visual Basic

```
Declare Function R1K_Cnt_GetFilters Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wFilter	An array that stored the filter value for contiguous D/I channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_SetFilters

This function code is used to set contiguous channel's filter value when D/I channel in 'Count' mode.

C/C++

```
int R1K_Cnt_SetFilters( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wFilter[ ]);
```

Visual Basic

```
Declare Function R1K_Cnt_SetFilters Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iFilter As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wFilter	An array that stored the filter value for contiguous D/I channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_GetStartStatuses

This function code is used to get contiguous channel's start status when D/I channel in 'Count' mode.

C/C++

```
int R1K_Cnt_GetStartStatuses( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD * dwStatus);
```

Visual Basic

```
Declare Function R1K_Cnt_GetStartStatuses Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous count channel's start status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_SetStartStatuses

This function code is used to set contiguous channel's start status when D/I channel in 'Count' mode.

C/C++

```
int R1K_Cnt_SetStartStatuses( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD dwStatus);
```

Visual Basic

```
Declare Function R1K_Cnt_SetStartStatuses Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, ByVal nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous count channel's start status; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_GetTriggerTypeWords

This function code is used to get contiguous channel's trigger types when D/I channel in 'Count' mode.

C/C++

```
int R1K_Cnt_GetTriggerTypeWords(int hConnection,
                                BYTE bytStartChannel
                                BYTE bytCount
                                WORD * wType) ;
```

Visual Basic

```
Declare Function R1K_Cnt_GetTriggerTypeWords Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iType As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
wType	A pointer that stores the contiguous channel's triggers types; each bit holds one channel trigger type. A bit value of 0 represents the trigger type of the start channel. A bit value of 1 represents the second channel's trigger type. The values are : 0: LoToHi 1: HiToLo 2: Both

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_SetTriggerTypeWords

This function code is used to set contiguous channel's trigger types when D/I channel in 'Count' mode.

C/C++

```
int R1K_Cnt_SetTriggerTypeWords( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                WORD * wType);
```

Visual Basic

```
Declare Function R1K_Cnt_SetTriggerTypeWords Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iType As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear
wType	Stored the contiguous channel's triggers types; each bit holds one channel trigger type. A bit value of 0 represents the trigger type of the start channel. A bit value of 1 represents the second channel's trigger type. The values are : 0: LoToHi 1: HiToLo 2: Both

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_GetPowerOnValues

This function code is used to get contiguous channel's power on values when D/I channel in 'Count' mode.

C/C++

```
int R1K_Cnt_GetPowerOnValues( int hConnection,
                                BYTE bytStartChannel
                                BYTE bytCount
                                DWORD * dwValue) ;
```

Visual Basic

```
Declare Function R1K_Cnt_GetPowerOnValues Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
dwValue	A pointer that stores the contiguous channel's power on values; each bit holds one channel power on value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the second channel's power on value. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_SetPowerOnValues

This function code is used to set contiguous channel's power on values when D/I channel in 'Count' mode.

C/C++

```
int R1K_Cnt_SetPowerOnValues( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD dwValue);
```

Visual Basic

```
Declare Function R1K_Cnt_SetPowerOnValues Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear
dwValue	Stored the contiguous channel's power on values; each bit holds one channel power on value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the second channel's power on value. The values are: 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_GetSafeValues

This function code is used to get contiguous channel's safe values when D/I channel in 'Count' mode.

C/C++

```
int R1K_Cnt_GetSafeValues( int hConnection,
                           BYTE bytStartChannel
                           BYTE bytCount
                           DWORD * dwValue);
```

Visual Basic

```
Declare Function R1K_Cnt_GetSafeValues Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get
dwValue	A pointer that stores the contiguous channel's safe values; each bit holds one channel safe value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the second channel's safe value. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_SetSafeValues

This function code is used to set contiguous channel's safe values when D/I channel in 'Count' mode.

C/C++

```
int R1K_Cnt_SetSafeValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwValue) ;
```

Visual Basic

```
Declare Function R1K_Cnt_SetSafeValues Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be clear
dwValue	Stored the contiguous channel's safe values; each bit holds one channel safe value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the second channel's safe value. The values are : 0: OFF 1: ON

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_GetSaveStatusesOnPowerFail

This function code is used to get contiguous channel's DI/DO power off storage enable mode.

C/C++

```
int R1K_Cnt_GetSaveStatusesOnPowerFail ( int hConnection,
                                         BYTE bytStartChannel,
                                         BYTE bytCount,
                                         DWORD * dwMode );
```

Visual Basic

```
Declare Function R1K_Cnt_GetSaveStatusesOnPowerFail Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, nMode As
Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get.
dwStatus	A pointer that stores the contiguous channel's DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: ON, 1: OFF

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Cnt_SetSaveStatusesOnPowerFail

This function code is used to set contiguous channel's DI/DO mode power off storage enable mode.

C/C++

```
int R1K_Cnt_SetSaveStatusesOnPowerFail ( int hConnection,
                                         BYTE bytStartChannel,
                                         BYTE bytCount,
                                         DWORD dwMode );
```

Visual Basic

```
Declare Function R1K_Cnt_SetSaveStatusesOnPowerFail Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, ByVal
nMode As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	A pointer that stores the contiguous channel's DI/DO mode; each bit holds one channel status. A bit value of 0 represents the status of the start channel. A bit value of 1 represents the second channel's status. The values are : 0: ON, 1: OFF

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DO_Reads

This function code is used to read the output statuses of contiguous D/O channels.

C/C++

```
int R1K_DO_Reads( int hConnection,
                   BYTE bytStartChannel,
                   BYTE bytCount,
                   DWORD * dwValue) ;
```

Visual Basic

```
Declare Function R1K_DO_Reads Lib "MXIO.dll" (ByVal
hConnection As Long,
ByVal bytStartChannel As Byte, ByVal bytCount As Byte,
nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dwValue	A pointer that stores the contiguous D/O channel's status; each bit holds one channel value. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DO_Writes

This function code is used to write the output statuses of contiguous D/O channels.

C/C++

```
int R1K_DO_Writes( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    DWORD dwValue);
```

Visual Basic

```
Declare Function R1K_DO_Writes Lib "MXIO.dll" (ByVal
hConnection As Long,
ByVal bytStartChannel As Byte, ByVal bytCount As Byte,
ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be written.
dwValue	Stores the channel statuses of contiguous D/O channels; each bit holds one channel status. A bit value of 0 represents the digital output status of the start channel. A bit value of 1 represents the second digital output channel's status. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DO_GetSafeValues_W

This function code is used to get output safe values of contiguous D/O channels.

C/C++

```
int R1K_DO_GetSafeValues_W( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD * wValue);
```

Visual Basic

```
Declare Function R1K_DO_GetSafeValues_W Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wValue	A pointer that stores the safe value of contiguous D/O channels; each word holds one channel value. A word value of 0 represents the digital output status of the start channel. A word value of 1 represents the status of the second digital output channel. The values of the unused bits are random. 0: OFF 1: ON 2: Hold Last

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DO_SetSafeValues_W

This function code is used to set safe values of contiguous D/O channels.

C/C++

```
int R1K_DO_SetSafeValues_W( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            WORD * wValue);
```

Visual Basic

```
Declare Function R1K_DO_SetSafeValues_W Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, ByVal nValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	A pointer that stores the safe value of contiguous D/O channels; each word holds one channel value. A word value of 0 represents the digital output status of the start channel. A word value of 1 represents the status of the second digital output channel. The values of the unused bits are random. 0: OFF 1: ON 2: Hold Last

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DO_GetModes

This function code is used to get the mode of contiguous D/O channels.

C/C++

```
int R1K_DO_GetModes( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wMode[ ] );
```

Visual Basic

```
Declare Function R1K_DO_GetModes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wMode	An array that stores the mode of contiguous D/O channels. The values are: 0: D/O mode 1: Pulse mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DO_SetModes

This function code is used to set the mode of contiguous D/O channels.

C/C++

```
int R1K_DO_SetModes( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wMode[ ] );
```

Visual Basic

```
Declare Function R1K_DO_SetModes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iMode As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wMode	An array that stores the mode of contiguous D/O channels. The values are: 0: D/O mode 1: Pulse mode

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DO_GetPowerOnValues

This function code is used to get the power on value of contiguous D/O channels.

C/C++

```
int R1K_DO_GetPowerOnValues( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD * dwValue);
```

Visual Basic

```
Declare Function R1K_DO_GetPowerOnValues Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwValue	A pointer that stores the power on value of contiguous D/O channels; each bit holds one channel value. A bit value of 0 represents the power on status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DO_SetPowerOnValues

This function code is used to set the power on value of contiguous D/O channels.

C/C++

```
int R1K_DO_SetPowerOnValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           DWORD dwValue);
```

Visual Basic

```
Declare Function R1K_DO_SetPowerOnValues Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	Stores the power on value of contiguous D/O channels; each bit holds one channel value. A bit value of 0 represents the power on status of the start channel. A bit value of 1 represents the status of the second digital output channel. The values of the unused bits are random.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DO_GetPowerOnSeqDelaytimes

This function code is used to get contiguous channel's DO (Relay) delay time to power on.

C/C++

```
int R1K_DO_GetPowerOnSeqDelaytimes ( int hConnection,
                                     BYTE bytStartChannel,
                                     BYTE bytCount,
                                     WORD * wValue );
```

Visual Basic

```
Declare Function R1K_DO_GetPowerOnSeqDelaytimes Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	A pointer that stores the contiguous channel's power on sequence delay time (Second) The values are : Range: 0 ~ 300

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_DO_SetPowerOnSeqDelaytimes

This function code is used to set contiguous channel's DO (Relay) delay time to power on.

C/C++

```
int R1K_DO_SetPowerOnSeqDelaytimes ( int hConnection,
                                     BYTE bytStartChannel,
                                     BYTE bytCount,
                                     WORD * wValue);
```

Visual Basic

```
Declare Function R1K_DO_SetPowerOnSeqDelaytimes Lib
"MXIO.dll" (ByVal hConnection As Long, ByVal
bytStartChannel As Byte, ByVal bytCount As Byte, iValue As
Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
wValue	A pointer that stores the contiguous channel's power on sequence delay time (Second) The values are : Range0 ~ 300

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_RLY_TotalCntReads

This function code is used to get count value of contiguous D/O channels.

C/C++

```
int R1K_RLY_TotalCntReads ( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD dwValue[ ] );
```

Visual Basic

```
Declare Function R1K_RLY_TotalCntReads Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, nValue As long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	An array that stores the contiguous D/O channels relay count. dwValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_RLY_CurrentCntReads

This function code is used to get count value of current D/O channels.

C/C++

```
int R1K_RLY_CurrentCntReads ( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD dwValue[ ] );
```

Visual Basic

```
Declare Function R1K_RLY_CurrentCntReads Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	An array that stores the current D/O channels relay count. dwValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_RLY_ResetCnts

This function code is used to reset count value of current D/O channels.

C/C++

```
int R1K_RLY_ResetCnts ( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount) ;
```

Visual Basic

```
Declare Function R1K_RLY_ResetCnts Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Pulse_GetSignalWidths

This function code is used to get the contiguous pulse channel's Hi/Lo signal width(32 bits).

C/C++

```
int R1K_Pulse_GetSignalWidths( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               WORD wHiWidth[ ],
                               WORD wLoWidth[ ]);
```

Visual Basic

```
Declare Function R1K_Pulse_GetSignalWidths Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iHiWidth As Integer, iLoWidth As
Integer) As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytStartChannel Specifies the starting channel.

bytCount The number of channels to be gets.

wHiWidth An array that stores the contiguous pulse channel's Hi signal width, wHiWidth[0] represents the Hi signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

wLoWidth An array that stores the contiguous pulse channel's Lo signal width, wLoWidth[0] represents the Lo signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Pulse_SetSignalWidths

This function code is used to set the contiguous pulse channel's Hi/Lo signal width(32 bits) .

C/C++

```
int R1K_Pulse_SetSignalWidths( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               WORD wHiWidth[ ],
                               WORD wLoWidth[ ]);
```

Visual Basic

```
Declare Function R1K_Pulse_SetSignalWidths Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, iHiWidth As Integer, iLoWidth As
Integer) As Long
```

Arguments:

hConnection The handle for an I/O device connection.

bytStartChannel Specifies the starting channel.

bytCount The number of channels to be set.

wHiWidth An array that stores the contiguous pulse channel's Hi signal width,
wHiWidth[0] represents the Hi signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

wLoWidth An array that stores the contiguous pulse channel's Lo signal width,
wLoWidth[0] represents the Lo signal width of the starting channel.

Refer to this table:

[wHiWidth And wLoWidth Renaming Table](#)

Return Value:

Succeed MXIO_OK
Fail Refer to Return Codes.

R1K_Pulse_GetOutputCounts

This function code is used to get the contiguous pulse channel's output count.

C/C++

```
int R1K_Pulse_GetOutputCounts( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD dwOutputCounts[ ] );
```

Visual Basic

```
Declare Function R1K_Pulse_GetOutputCounts Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nOutputCounts As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwOutputCounts	An array that stores the contiguous pulse channel's output count, dwOutputCounts[0] represents the pulse output count of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Pulse_SetOutputCounts

This function code is used to set the contiguous pulse channel's output count.

C/C++

```
int R1K_Pulse_SetOutputCounts( int hConnection,
                               BYTE bytStartChannel,
                               BYTE bytCount,
                               DWORD dwOutputCounts[ ]);
```

Visual Basic

```
Declare Function R1K_Pulse_SetOutputCounts Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nOutputCounts As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwOutputCounts	An array that stores the contiguous pulse channel's output count, dwOutputCounts[0] represents the pulse output count of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Pulse_GetStartStatuses

This function code is used to get the contiguous pulse channel's start status.

C/C++

```
int R1K_Pulse_GetStartStatuses( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD * dwStatus);
```

Visual Basic

```
Declare Function R1K_Pulse_GetStartStatuses Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwStatus	An point that stores the contiguous pulse channel's start status, each bit holds one channel value. A bit value of 0 represents the start status of the start channel. A bit value of 1 represents the status of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Pulse_SetStartStatuses

This function code is used to set the contiguous pulse channel's start status.

C/C++

```
int R1K_Pulse_SetStartStatuses( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD dwStatus) ;
```

Visual Basic

```
Declare Function R1K_Pulse_SetStartStatuses Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, ByVal nStatus As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwStatus	An point that stores the contiguous pulse channel's start status, each bit holds one channel value. A bit value of 0 represents the start status of the start channel. A bit value of 1 represents the status of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Pulse_GetPowerOnValues

This function code is used to get the contiguous pulse channel's power on value.

C/C++

```
int R1K_Pulse_GetPowerOnValues( int hConnection,
                                BYTE bytStartChannel,
                                BYTE bytCount,
                                DWORD * dwValue);
```

Visual Basic

```
Declare Function R1K_Pulse_GetPowerOnValues Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
bytStatus	An point that stores the contiguous pulse channel's power on value, each bit holds one channel value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Pulse_SetPowerOnValues

This function code is used to set the contiguous pulse channel's power on value.

C/C++

```
int R1K_Pulse_SetPowerOnValues(int hConnection,  
                                BYTE bytStartChannel,  
                                BYTE bytCount,  
                                DWORD dwValue);
```

Visual Basic

```
Declare Function R1K_Pulse_SetPowerOnValues Lib "MXIO.dll"  
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,  
ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	Stores the contiguous pulse channel's power on value, each bit holds one channel value. A bit value of 0 represents the power on value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Pulse_GetSafeValues

This function code is used to get the contiguous pulse channel's safe value.

C/C++

```
int R1K_Pulse_GetSafeValues( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD * dwValue);
```

Visual Basic

```
Declare Function R1K_Pulse_GetSafeValues Lib "MXIO.dll"
(ByVal hConnection As Long, ByVal bytStartChannel As Byte,
ByVal bytCount As Byte, nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
dwValue	An point that stores the contiguous pulse channel's safe value, each bit holds one channel value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_Pulse_SetSafeValues

This function code is used to set the contiguous pulse channel's safe value.

C/C++

```
int R1K_Pulse_SetSafeValues( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            DWORD dwValue);
```

Visual Basic

```
Declare Function R1K_Pulse_SetSafeValues Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, ByVal nValue As Long) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be set.
dwValue	An point that stores the contiguous pulse channel's safe value, each bit holds one channel value. A bit value of 0 represents the safe value of the start channel. A bit value of 1 represents the value of the second pulse channel. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AI_Reads

This function code is used to read contiguous channel's analog input value.

C/C++

```
int R1K_AI_Reads( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    double dValue[ ]);
```

Visual Basic

```
Declare Function R1K_AI_Reads Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel's value, dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AI_ReadRaws

This function code is used to read contiguous channel's analog input raw data.

C/C++

```
int R1K_AI_ReadRaws( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ] );
```

Visual Basic

```
Declare Function R1K_AI_ReadRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel's raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AI_ReadMaxs

This function code is used to read contiguous A/I channel's maximize value.

C/C++

```
int R1K_AI_ReadMaxs( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] );
```

Visual Basic

```
Declare Function R1K_AI_ReadMaxs Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel's maximize value , dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AI_ReadMaxRaws

This function code is used to read contiguous A/I channel's maximize raw data.

C/C++

```
int R1K_AI_ReadMaxRaws( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wValue[ ] );
```

Visual Basic

```
Declare Function R1K_AI_ReadMaxRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel's maximize raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AI_ResetMaxs

This function code is used to reset contiguous A/I channel's maximize value.

C/C++

```
int R1K_AI_ResetMaxs( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount) ;
```

Visual Basic

```
Declare Function R1K_AI_ResetMaxs Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AI_ReadMins

This function code is used to read contiguous A/I channel's minimize value.

C/C++

```
int R1K_AI_ReadMins( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      double dValue[ ] );
```

Visual Basic

```
Declare Function R1K_AI_ReadMins Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/I channel's value, dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage module, and mA for the current module.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AI_ReadMinRaw

This function code is used to read contiguous A/I channel's minimize raw data.

C/C++

```
int R1K_AI_ReadMinRaw( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ] );
```

Visual Basic

```
Declare Function R1K_AI_ReadMinRaw Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/I channel's minimize raw data , wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AI_ResetMins

This function code is used to reset contiguous A/I channel's minimize value.

C/C++

```
int R1K_AI_ResetMins( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount);
```

Visual Basic

```
Declare Function R1K_AI_ResetMins Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be reset.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AI_GetRanges

This function code is used to get contiguous A/I channel's range.

C/C++

```
int R1K_AI_GetRanges( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wRange[ ] );
```

Visual Basic

```
Declare Function R1K_AI_GetRanges Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iRange As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wRange	An array that stores the contiguous A/I channel's range, wRange[0] represents the value of the starting channel. The values are: 0: 0-10V 1: 0-20mA 2: 4-20mA Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AO_Reads

This function code is used to read the output value of contiguous Analog Output channels.

C/C++

```
int R1K_AO_Reads( int hConnection,
                   BYTE bytStartChannel,
                   BYTE bytCount,
                   double dValue[ ] );
```

Visual Basic

```
Declare Function R1K_AO_Reads Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous channel output value to be read. The dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AO_Writes

This function code is used to write the output value for contiguous channels.

C/C++

```
int R1K_AO_Writes( int hConnection,
                    BYTE bytStartChannel,
                    BYTE bytCount,
                    double dValue[ ]);
```

Visual Basic

```
Declare Function R1K_AO_Writes Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to write.
dValue	An array that stores the contiguous channel output value to write. The dValue[0] represents the value of the starting Analog output channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AO_ReadRaws

This function code is used to read the output raw data of contiguous Analog Output channels.

C/C++

```
int R1K_AO_ReadRaws( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ] );
```

Visual Basic

```
Declare Function R1K_AO_ReadRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous channel output raw data to be read. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AO_WriteRaws

This function code is used to write the output raw data for contiguous channels.

C/C++

```
int R1K_AO_WriteRaws( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wValue[ ] );
```

Visual Basic

```
Declare Function R1K_AO_WriteRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to write.
wValue	An array that stores the contiguous channel output raw data to write. The wValue[0] represents the value of the starting Analog output channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AO_GetSafeValues

This function code is used to get the output value of contiguous Analog Output channels.

C/C++

```
int R1K_AO_GetSafeValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           double dValue[ ] );
```

Visual Basic

```
Declare Function R1K_AO_GetSafeValues Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
dValue	An array that stores the contiguous A/O channel's safe value to be get. The dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AO_SetSafeValues

This function code is used to set the safe value for contiguous A/O channels.

C/C++

```
int R1K_AO_SetSafeValues( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           double dValue[ ] );
```

Visual Basic

```
Declare Function R1K_AO_SetSafeValues Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to set.
dValue	An array that stores the contiguous A/O channel's safe value to set. The dValue[0] represents the value of the starting Analog output channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AO_GetSafeRaws

This function code is used to set the output value of contiguous Analog Output channels.

C/C++

```
int R1K_AO_GetSafeRaws( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wValue[ ]);
```

Visual Basic

```
Declare Function R1K_AO_GetSafeRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wValue	An array that stores the contiguous A/O channel's safe raw data to be get. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AO_SetSafeRaws

This function code is used to set the safe raw data for contiguous A/O channels.

C/C++

```
int R1K_AO_SetSafeRaws( int hConnection,
                        BYTE bytStartChannel,
                        BYTE bytCount,
                        WORD wValue[ ]);
```

Visual Basic

```
Declare Function R1K_AO_SetSafeRaws Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to write.
wValue	An array that stores the contiguous channel safe raw data to set. The wValue[0] represents the value of the starting Analog output channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AO_GetRanges

This function code is used to get contiguous A/O channel's range.

C/C++

```
int R1K_AO_GetRanges( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wRange[ ]);
```

Visual Basic

```
Declare Function R1K_AO_GetRanges Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iRange As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be gets.
wRange	An array that stores the contiguous A/O channel's range, wRange[0] represents the value of the starting channel. The values are: 0: 0-10 Vdc 1: 0-20 mA Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AO_SetRanges

This function code is used to set the range for contiguous A/O channels.

C/C++

```
int R1K_AO_SetRanges( int hConnection,
                      BYTE bytStartChannel,
                      BYTE bytCount,
                      WORD wRange[ ] );
```

Visual Basic

```
Declare Function R1K_AO_SetRanges Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytStartChannel As Byte, ByVal
bytCount As Byte, iRange As integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to set.
wRange	An array that stores the contiguous A/O channel's range, wRange[0] represents the value of the starting channel. The values are: 0: 0-10 Vdc 1: 0-20 mA Others_return Illegal Data Value

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AO_GetPowerOnValues

This function code is used to get the power on value of contiguous A/O channels.

C/C++

```
int R1K_AO_GetPowerOnValues( int hConnection,
                            BYTE bytStartChannel,
                            BYTE bytCount,
                            double dValue[ ] );
```

Visual Basic

```
Declare Function R1K_AO_GetPowerOnValues Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be get.
dValue	An array that stores the contiguous A/O channel's power on value to be get. The dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AO_SetPowerOnValues

This function code is used to set the power on value for contiguous A/O channels.

C/C++

```
int R1K_AO_SetPowerOnValues( int hConnection,
                             BYTE bytStartChannel,
                             BYTE bytCount,
                             double dValue[ ] );
```

Visual Basic

```
Declare Function R1K_AO_SetPowerOnValues Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, dValue As Double) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to set.
dValue	An array that stores the contiguous A/O channel's power on value to be set. The dValue[0] represents the value of the starting channel. The unit is Vdc for the voltage channel and mA for the current channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AO_GetPowerOnRaws

This function code is used to get the power on raw data of contiguous A/O channels.

C/C++

```
int R1K_AO_GetPowerOnRaws( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wValue[ ] );
```

Visual Basic

```
Declare Function R1K_AO_GetPowerOnRaws Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to be read.
wValue	An array that stores the contiguous A/O channel's power on raw data to be get. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

R1K_AO_SetPowerOnRaws

This function code is used to set the power on raw data for contiguous A/O channels.

C/C++

```
int R1K_AO_SetPowerOnRaws( int hConnection,
                           BYTE bytStartChannel,
                           BYTE bytCount,
                           WORD wValue[ ] );
```

Visual Basic

```
Declare Function R1K_AO_SetPowerOnRaws Lib "MXIO.dll"
    (ByVal hConnection As Long, ByVal bytStartChannel As Byte,
    ByVal bytCount As Byte, iValue As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytStartChannel	Specifies the starting channel.
bytCount	The number of channels to set.
wValue	An array that stores the contiguous A/O channel's power on raw data to be set. The wValue[0] represents the value of the starting channel.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Modbus Command Sets

Modbus Command Sets provide four basic Modbus commands. However, you must refer to the Modbus Reference table in ioAdmin to extract the data from the Modbus packet.

[MXIO_ReadCoils](#)
[MXIO_WriteCoils](#)
[MXIO_ReadRegs](#)
[MXIO_WriteRegs](#)
[MXIO_ReadCoils_Ex](#)
[MXIO_WriteCoils_Ex](#)
[MXIO_ReadRegs_Ex](#)
[MXIO_WriteRegs_Ex](#)

MXIO_ReadCoils

This function reads the on/off status for a contiguous group of coils on the same I/O device.

C/C++

```
int MXIO_ReadCoils( int hConnection,
                     BYTE bytCoilType,
                     WORD wStartCoil,
                     WORD wCount,
                     BYTE bytCoils[ ]);
```

Visual Basic

```
Declare Function MXIO_ReadCoils Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytCoilType As Byte, ByVal
iStartCoil As Integer, ByVal iCount As Integer, bytCoils
As Byte) As Long
```

Arguments:

hConnection	Handle for the I/O device connection.
bytCoilType	Coil type to be read. 1: read coils (output bit). 2: read discrete (input bit).
wStartCoil	Specifies the starting coil address to be read. The address is beginning at 0.
wCount	The number of coils to be read.
bytCoils	An array that stores the status of coils; each byte holds eight coil values. Bit 0 of 1st byte represents 1st coil status.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_WriteCoils

This function code is used to write the contiguous status of an I/O device's coils.

C/C++

```
int MXIO_WriteCoils( int hConnection,
                      WORD wStartCoil,
                      WORD wCount,
                      BYTE bytCoils[ ] );
```

Visual Basic

```
Declare Function MXIO_WriteCoils Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal iStartCoil As Integer, ByVal
iCount As Integer, bytCoils As Byte) As Long
```

Arguments:

hConnection	Handle for the I/O device connection.
wStartCoil	Specifies the starting coil to be written. The address is beginning at 0.
wCount	The number of coils to be written.
bytCoils	An array that stores the coil value, each byte holds eight coil values.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_ReadRegs

This function code is used to read the contents of a contiguous block of the I/O device's holding registers.

C/C++

```
int MXIO_ReadRegs( int hConnection,
                    BYTE bytRegisterType,
                    WORD wStartRegister,
                    WORD wCount,
                    WORD wRegister[ ] );
```

Visual Basic

```
Declare Function MXIO_ReadRegs Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytRegisterType As Byte, ByVal
iStartRegister As Integer, ByVal iCount As Integer,
iRegister As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytRegisterType	Coil type to be read. The meaning for a value in an entity is as follows: 3: read holding registers (output word) 4: read input register (input word)
wStartRegister	Specifies the starting register address. The address is beginning at 0.
wCount	The number of registers to be read.
wRegister	An array that stores the register values

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_WriteRegs

This function code is used to write the contents of a contiguous block of the I/O device's holding registers.

C/C++

```
int MXIO_WriteRegs( int hConnection,
                     WORD wStartRegister,
                     WORD wCount,
                     WORD wRegister[ ]);
```

Visual Basic

```
Declare Function MXIO_WriteRegs Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal iStartRegister As Integer, ByVal
iCount As Integer, iRegister As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
wStartRegister	Specifies the starting register address. The address is beginning at 0.
wCount	The number of registers to be written.
wRegister	An array that stores the register values.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_ReadCoils_Ex

This function reads the fix modbus address on/off status for a contiguous group of coils on the same I/O device.

C/C++

```
int MXIO_ReadCoils_Ex( int hConnection,
                       BYTE bytCoilType,
                       WORD wStartCoil,
                       WORD wCount,
                       BYTE bytCoils[ ] );
```

Visual Basic

```
Declare Function MXIO_ReadCoils_Ex Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal bytCoilType As Byte, ByVal
iStartCoil As Integer, ByVal iCount As Integer, bytCoils
As Byte) As Long
```

Arguments:

hConnection	Handle for the I/O device connection.
bytCoilType	Coil type to be read. 1: read coils (output bit). 2: read discrete (input bit).
wStartCoil	Specifies the starting coil address to be read. The address is beginning at 0.
wCount	The number of coils to be read.
bytCoils	An array that stores the status of coils; each byte holds eight coil values. Bit 0 of 1 st byte represents 1 st coil status.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_WriteCoils_Ex

This function code is used to write the fix modbus address contiguous status of an I/O device's coils.

C/C++

```
int MXIO_WriteCoils_Ex( int hConnection,
                        WORD wStartCoil,
                        WORD wCount,
                        BYTE bytCoils[ ] );
```

Visual Basic

```
Declare Function MXIO_WriteCoils_Ex Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal iStartCoil As Integer, ByVal
iCount As Integer, bytCoils As Byte) As Long
```

Arguments:

hConnection	Handle for the I/O device connection.
wStartCoil	Specifies the starting coil to be written. The address is beginning at 0.
wCount	The number of coils to be written.
bytCoils	An array that stores the coil value, each byte holds eight coil values.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_ReadRegs_Ex

This function code is used to read the fix modbus address contents of a contiguous block of the I/O device's holding registers.

C/C++

```
int MXIO_ReadRegs_Ex( int hConnection,
                      BYTE bytRegisterType,
                      WORD wStartRegister,
                      WORD wCount,
                      WORD wRegister[ ] );
```

Visual Basic

```
Declare Function MXIO_ReadRegs_Ex Lib MXIO.dll (ByVal
hConnection As Long, ByVal bytRegisterType As Byte, ByVal
iStartRegister As Integer, ByVal iCount As Integer,
iRegister As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
bytRegisterType	Coil type to be read. The meaning for a value in an entity is as follows: 3: read holding registers (output word) 4: read input register (input word)
wStartRegister	Specifies the starting register address. The address is beginning at 0.
wCount	The number of registers to be read.
wRegister	An array that stores the register values

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

MXIO_WriteRegs_Ex

This function code is used to write the fix modbus address contents of a contiguous block of the I/O device's holding registers.

C/C++

```
int MXIO_WriteRegs_Ex( int hConnection,
                        WORD wStartRegister,
                        WORD wCount,
                        WORD wRegister[ ] );
```

Visual Basic

```
Declare Function MXIO_WriteRegs_Ex Lib "MXIO.dll" (ByVal
hConnection As Long, ByVal iStartRegister As Integer, ByVal
iCount As Integer, iRegister As Integer) As Long
```

Arguments:

hConnection	The handle for an I/O device connection.
wStartRegister	Specifies the starting register address. The address is beginning at 0.
wCount	The number of registers to be written.
wRegister	An array that stores the register values.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Click & Go Logic Commands for ioLogik E2000 series

There are two Logic commands to start/stop Click & Go Logic of ioLogik 2000 Ethernet Module.

Click & Go Logic Command for ioLogik 2000 Ethernet Module

[Logic2K_GetStartStatus](#)

[Logic2K_SetStartStatus](#)

Logic2K_GetStartStatus

This function code is used to get the Click & Go Logic start status of ioLogik 2000 Ethernet Module.

C/C++

```
int Logic2K_GetStartStatus( int hConnection,  
                           WORD * wStatus);
```

Visual Basic

```
Declare Function Logic2K_GetStartStatus Lib "MXIO.dll"  
(ByVal hConnection As Long, iStatus As Integer) As  
Long
```

Arguments:

hConnection	The handle for a connection.
iStatus	An pointer that stores the specific module's Click & Go Logic start status. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Logic2K_SetStartStatus

This function code is used to set the Click & Go Logic start status of ioLogik 2000 Ethernet Module.

C/C++

```
int Logic2K_SetStartStatus( int hConnection,  
                           WORD wStatus) ;
```

Visual Basic

```
Declare Function Logic2K_SetStartStatus Lib "MXIO.dll"  
          (ByVal hConnection As Long, ByVal iStatus As Integer)  
          As Long
```

Arguments:

hConnection	The handle for a connection.
iStatus	Stores the specific module's Click & Go Logic start status. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Click & Go Logic Commands for ioLogik E4200 series

There are two Logic commands to start/stop Click & Go Logic of ioLogik 4200 network adapters.

Click & Go Logic Command for ioLogik 4200 Ethernet Module

[E42_Logic_GetStartStatus](#)

[E42_Logic_SetStartStatus](#)

E42_Logic_GetStartStatus

This function code is used to get the Click & Go Logic start status of ioLogik 4200 network adapters.

C/C++

```
int E42_Logic_GetStartStatus ( int hConnection,  
                                WORD * wStatus);
```

Visual Basic

```
Declare Function E42_Logic_GetStartStatus Lib MXIO.dll  
(ByVal hConnection As Long, iStatus As Integer) As  
Long
```

Arguments :

`hConnection` The handle for a connection.

iStatus An pointer that stores the specific module's Click & Go Logic start status. The values are:
0: stop
1: start

Return Value:

Succeed MXIO OK

Fail Refer to Return Codes.

E42_Logic_SetStartStatus

This function code is used to set the Click & Go Logic start status of ioLogik 4200 network adapters.

C/C++

```
int E42_Logic_SetStartStatus ( int hConnection,  
                               WORD wStatus);
```

Visual Basic

```
Declare Function E42_Logic_SetStartStatus Lib MXIO.dll  
(ByVal hConnection As Long, ByVal iStatus As Integer)  
As Long
```

Arguments:

hConnection	The handle for a connection.
iStatus	Stores the specific module's Click & Go Logic start status. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Click & Go Logic Commands for ioLogik E5000 series

There are two Logic commands to start/stop Click & Go Logic of ioLogik 5000 Ethernet Module.

Click & Go Logic Command for ioLogik 5000 Ethernet Module

[W5K_Logic_GetStartStatus](#)

[W5K_Logic_SetStartStatus](#)

W5K_Logic_GetStartStatus

This function code is used to get the Click & Go Logic start status of ioLogik 5000 Ethernet Module.

C/C++

```
int W5K_Logic_GetStartStatus( int hConnection,  
                           WORD * wStatus);
```

Visual Basic

```
Declare Function W5K_Logic_GetStartStatus Lib MXIO.dll  
(ByVal hConnection As Long, iStatus As Integer) As  
Long
```

Arguments :

`hConnection` The handle for a connection.

iStatus An pointer that stores the specific module's Click & Go Logic start status. The values are:
0: stop
1: start

Return Value:

Succeeded	<code>MXIO_OK</code>
Failed	Refer to Return Codes.

W5K_Logic_SetStartStatus

This function code is used to set the Click & Go Logic start status of ioLogik 5000 Ethernet Module.

C/C++

```
int W5K_Logic_SetStartStatus( int hConnection,  
                           WORD wStatus);
```

Visual Basic

```
Declare Function W5K_Logic_SetStartStatus Lib MXIO.dll  
(ByVal hConnection As Long, ByVal iStatus As Integer)  
As Long
```

Arguments:

hConnection	The handle for a connection.
iStatus	Stores the specific module's Click & Go Logic start status. The values are: 0: stop 1: start

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Message Command for ioLogik E2000 Series

There are two message commands to start/stop receive message of ioLogik 2000 Ethernet Module.

Message Command for ioLogik E2000 series

[Message2K_Start](#)

[Message2K_Stop](#)

Message2K_Start

This function code is used to start receive active message of ioLogik 2000 Ethernet Module.

C/C++

```
int Message2K_Start( int iProtocol,
                      WORD wPort,
                      pfnCALLBACK iProcAddress);
```

Callback Function

```
void FunctionName( BYTE bytData[ ],
                   WORD wSize );
```

Visual Basic

```
Declare Function Message2K_Start Lib "MXIO.dll" (ByVal
nProtocol As Long, ByVal iPort as Integer, ByVal
nProcAddress As Long) As Long
```

Callback Function

```
Public Sub FunctionName(bytData As Long, ByVal iSize
As Integer)
```

NOTE:

Message Command is not recommend to use by VB. Please refer the Exmaple file for more detail. If you want to use it by VB language, please select P-Code while compiling to execute file.

Arguments:

iProtocol	Transmission protocol. 1: TCP 2: UDP
wPort	TCP or UDP port number.
iProcAddress	Callback function, which is called after receive an active message form ioLogik 2000 Ethernet module.
bytData	An array that stores the message.

wSize Array size.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Message2K_Stop

This function code is used to stop receive active message of ioLogik 2000 Ethernet Module.

C/C++

```
int Message2K_Stop( int iProtocol);
```

Visual Basic

```
Declare Function Message2K_Stop Lib "MXIO.dll" (ByVal  
nProtocol As Long) As Long
```

Arguments:

iProtocol	Transmission protocol. 1: TCP 2: UDP
-----------	--

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Message Command for ioLogik E4200 Series

There are two message commands to start/stop receive message of ioLogik 4200 network adapters.

Message Command for ioLogik E4200 series

[E42_Message_Start](#)

[E42_Message_Stop](#)

E42_Message_Start

This function code is used to start receive active message of ioLogik 4200 network adapters.

C/C++

```
int E42_Message_Start ( int iProtocol,
                        WORD wPort,
                        pfnCALLBACK iProcAddress );
```

Callback Function

```
void FunctionName( BYTE bytData[ ],
                    WORD wSize );
```

Visual Basic

```
Declare Function E42_Message_Start Lib MXIO.dll (ByVal
nProtocol As Long, ByVal iPort as Integer, ByVal
nProcAddress As Long) As Long
```

Callback Function

```
Public Sub FunctionName(bytData As Long, ByVal iSize
As Integer)
```

NOTE:

Message Command is not recommend to use by VB. Please refer the Exmaple file for more detail. If you want to use it by VB language, please select P-Code while compiling to execute file.

Arguments:

iProtocol	Transmission protocol. 1: TCP 2: UDP
wPort	TCP or UDP port number.
iProcAddress	Callback function, which is called after receive an active message form ioLogik 4200 Ethernet module.
bytData	An array that stores the message.

wSize Array size.

Return Value:

Succeed MXIO_OK

Fail Refer to Return Codes.

E42_Message_Stop

This function code is used to stop receive active message of ioLogik 4200 network adapters.

C/C++

```
int E42_Message_Stop( int iProtocol);
```

Visual Basic

```
Declare Function E42_Message_Stop Lib MXIO.dll (ByVal  
nProtocol As Long) As Long
```

Arguments:

iProtocol	Transmission protocol. 1: TCP 2: UDP
-----------	--

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

Message Command for ioLogik E5000 Series

There are two message commands to start/stop receive message of ioLogik 5000 Ethernet Module.

Message Command for ioLogik E5000 series

[W5K_Message_Start](#)

[W5K_Message_Stop](#)

W5K_Message_Start

This function code is used to start receive active message of ioLogik 5000 Ethernet Module.

C/C++

```
int W5K_Message_Start( int iProtocol,
                        WORD wPort,
                        pfnCALLBACK iProcAddress);
```

Callback Function

```
void FunctionName( BYTE bytData[ ],
                    WORD wSize );
```

Visual Basic

```
Declare Function W5K_Message_Start Lib MXIO.dll (ByVal
nProtocol As Long, ByVal iPort as Integer, ByVal
nProcAddress As Long) As Long
```

Callback Function

```
Public Sub FunctionName(bytData As Long, ByVal iSize
As Integer)
```

NOTE:

Message Command is not recommend to use by VB. Please refer the Exmaple file for more detail. If you want to use it by VB language, please select P-Code while compiling to execute file.

Arguments:

iProtocol	Transmission protocol. 1: TCP 2: UDP
wPort	TCP or UDP port number.
iProcAddress	Callback function, which is called after receive an active message form ioLogik 5000 Ethernet module.
bytData	An array that stores the message.

wSize Array size.

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.

W5K_Message_Stop

This function code is used to stop receive active message of ioLogik 5000 Ethernet Module.

C/C++

```
int W5K_Message_Stop( int iProtocol);
```

Visual Basic

```
Declare Function W5K_Message_Stop Lib MXIO.dll (ByVal  
nProtocol As Long) As Long
```

Arguments:

iProtocol	Transmission protocol. 1: TCP 2: UDP
-----------	--

Return Value:

Succeed	MXIO_OK
Fail	Refer to Return Codes.