ioLogik MXIO DLL API Reference

Sixth Edition, July 2009

www.moxa.com/product

ioLogik MXIO DLL API Reference

Copyright Notice

Trademarks

Disclaimer

Information in this document is subject to change without notice, and does not represent a commitment on the part of Moxa.

Moxa provides this document “as is,” without warranty of any kind, either expressed or implied, including, but not limited to, its particular purpose. Moxa reserves the right to make improvements, and/or changes to this manual, or to the products, and/or the programs described in this manual, at any time.

Information provided in this manual is intended to be accurate, and reliable. However, Moxa assumes no responsibility for its use, or for any infringements on the rights of third parties that may result from its use.

This manual might include unintentional technical or typographical errors. Changes are made periodically to the information herein to correct such errors, and these changes are incorporated into new editions of the manual.

Technical Support Contact Information

www.moxa.com/support

Moxa Americas:
Toll-free: 1-888-669-2872
Tel: +1-714-528-6777
Fax: +1-714-528-6778

Moxa China (Shanghai office):
Toll-free: 800-820-5036
Tel: +86-21-5258-9955
Fax: +86-10-6872-3958

Moxa Europe:
Tel: +49-89-3 70 03 99-0
Fax: +49-89-3 70 03 99-99

Moxa Asia-Pacific:
Tel: +886-2-8919-1230
Fax: +886-2-8919-1231

Table of Contents

1
Overview

This reference introduces the MXIO Library for Moxa's ioLogik 4000, E4200, E2000, W5000, R2000, and E1200 series.

The following topics are covered in this chapter:

- **What is the MXIO Library?**
- **Supported Platforms**
- **Supported I/O Modules**
- **Supported I/O Firmware**

What is the MXIO Library?

The MXIO Library is a set of tools for programmers to use with Moxa's ioLogik 4000, E4200, E2000, W5000 and R2000 remote I/O. Programmers can use the library when developing applications to manage I/O devices and obtain I/O data remotely over an Ethernet or RS-485 network. The library is designed for different kind platform such as Windows, WinCE, and Linux. Detail support list can be found in the following section.

Supported Platforms

Windows Platform

- Windows XP
- Windows 2000
- Windows NT

Linux Platform

- arm-elf-mxio-library-x.x.x.x.sh – for UC-7100-LX line
- mxsacleb-mxio-library-x.x.x.x.sh – for UC-7400-LX line
- mxio-library-x.x.x.x.sh – for x86
- arm-linux-mxio-library-x.x.x.x.sh - for IA-240-LX line

Supported I/O Modules

For a list of I/O modules that are supported by this library, please refer to ***Chapter 10, Product Model and ID Reference Table***.

ATTENTION

Click&Go logic and active I/O messaging are supported by the ioLogik E2000 line only.

Supported I/O Firmware

The latest MXIO library contains functions that support the latest firmware. Please refer to the following table to upgrade to the proper version. Please also refer to the ioLogik user's manual for the firmware upgrade procedures.

ioLogik Model	E2210	E2212	E2214	E2240	E2242	E2260	E2262	R2110	R2140
Firmware Version	V3.1↑	V3.1↑	V3.1↑	V2.2↑	V1.3↑	V1.2↑	V1.0↑	V1.4↑	V1.2↑

ioLogik Model	E4200	NA4XXX	W5340	E12XX					
Firmware Version	V1.0↑	V1.0↑	V1.0↑	V1.0↑					

2
Programming Flow

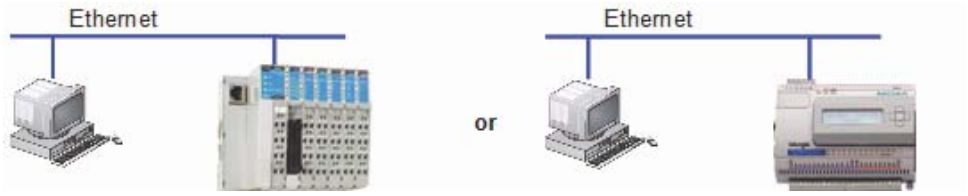
The process used to obtain access to a remote I/O device on an ioLogik is similar for both Ethernet and serial interfaces. Five different scenarios are described below.

The following topics are covered:

- **Connecting to a Single Ethernet I/O**
- **Connecting to Multiple Ethernet I/O**
- **Connecting to a Single Serial I/O**
- **Connecting to Multiple RS-485 I/O**
- **Connecting to the ioLogik E2000 and Attached RS-485 I/O**
- **Connecting to W5000 series GPRS I/O**
- **Modbus Command Sets vs. Direct I/O Command Sets**
  - Modbus Command Sets
  - Direct I/O Command Sets

Connecting to a Single Ethernet I/O

The MXIO Linux Library establishes a data tunnel using Modbus commands to communicate with the Ethernet I/O. Access is usually established using TCP port number 502.



ioLogik 4000 remote I/O
IP: 192.168.8.1
Port: 502

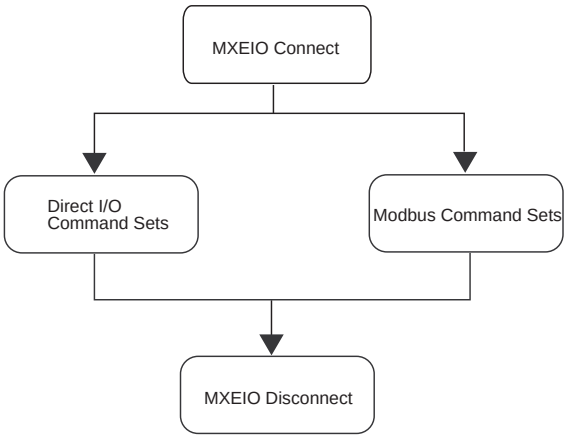
ioLogik E2000 Ethernet I/O
IP: 192.168.8.1
Port: 502

Three steps are required to access remote I/O data using the MXIO Linux Library.

1. Use MXEIO_Connect to connect to the Ethernet I/O using IP:Port (e.g., 192.168.8.1:502). MXEIO_Connect should return a handle.
2. Use the handle to access the desired I/O point with Modbus commands or direct I/O commands.
3. To finish the operation, use MXEIO_Disconnect to release Windows system resources.

Program Flow I.

Connecting to a Single Ethernet I/O



Connecting to Multiple Ethernet I/O

Before multiple Ethernet I/Os can be accessed over the network, make sure that each I/O has a unique IP address.

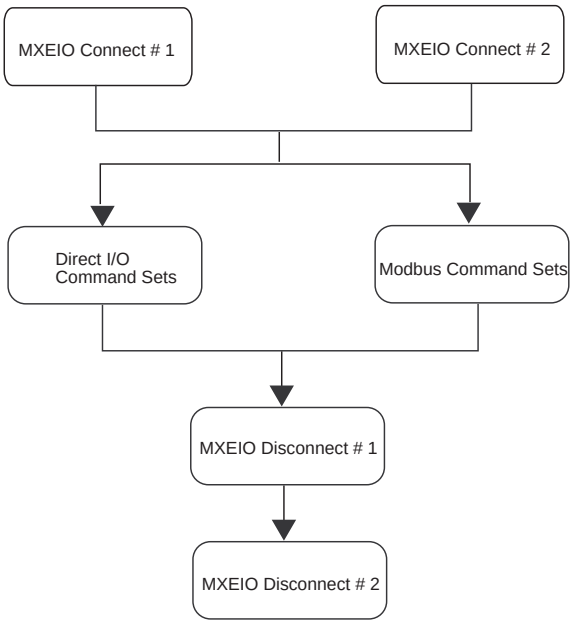
ioLogik 4000 remote I/O
IP: 192.168.8.1
Port: 502

ioLogik E2000 Ethernet I/O
IP: 192.168.8.2
Port: 502

Each Ethernet I/O needs a unique handle in order to be accessed. Use MXEIO_Connect to obtain the handle for each Ethernet I/O.

Program Flow II.

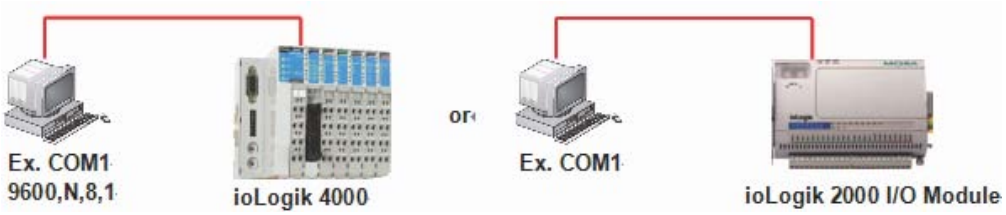
Connecting to Multiple Ethernet I/Os.



Connecting to a Single Serial I/O

The ioLogik 4000 and R2000 I/O can be used in RS-485, RS-232 control networks. For access to I/O over RS-485 or RS-232, please pay attention to the following:

- Your computer must be equipped with an RS-232 or RS-485 communication port.
- Make sure that the baudrate and communication parameters for the computer and the I/O are identical.
- Make sure that the I/O is running under Modbus/RTU



ioLogik 4000 remote I/O
ID: 10

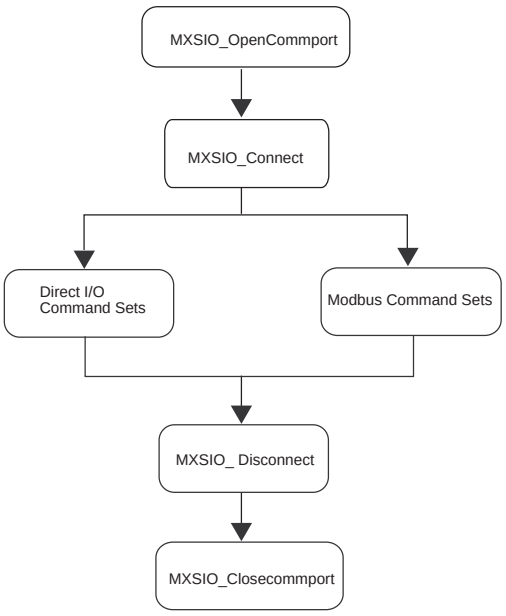
ioLogik E2000 Ethernet I/O
ID: 10

Four steps are required to access remote I/O data using the MXIO Linux Library.:

1. Use MXSIO_OpenCommport to open the COM port and connect to the serial I/O.
2. Use MXSIO_Connect to connect to the serial I/O using the Unit ID (e.g., 10). MXSIO_Connect should return a handle.
3. Use the handle to access the desired I/O point with Modbus command sets or direct I/O command sets.
4. To finish the operation, use MXSIO_Disconnect and MXSIO_CloseCommport to release Windows system resources.

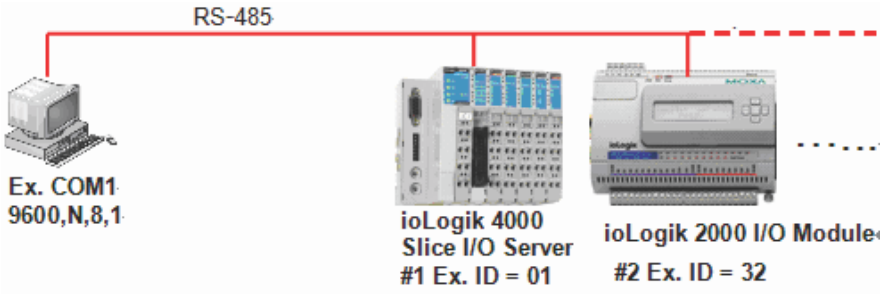
Program Flow III.

Connecting Single RS-485 I/O.



Connecting to Multiple RS-485 I/O

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



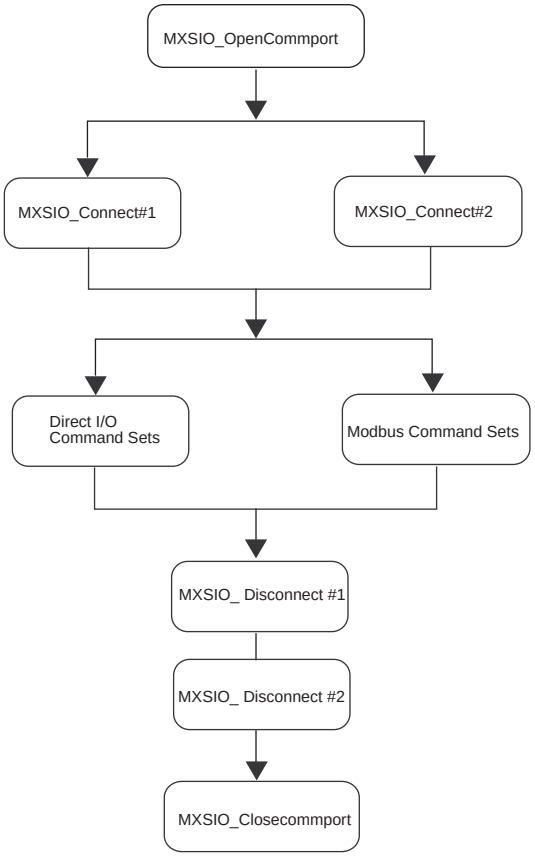
ioLogik 4000 remote I/O
ID: 01

ioLogik E2000 Ethernet I/O
ID: 32

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

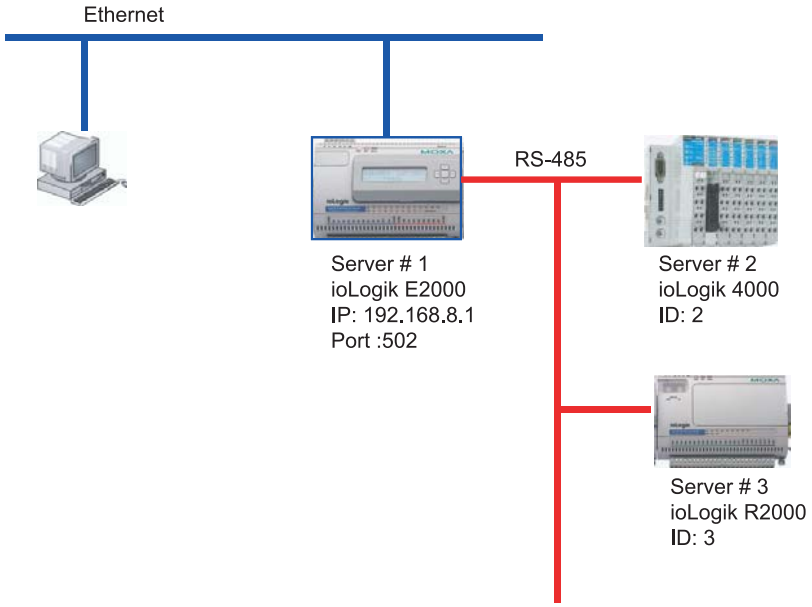
Program Flow IV.

Connecting Multiple Serial I/O.



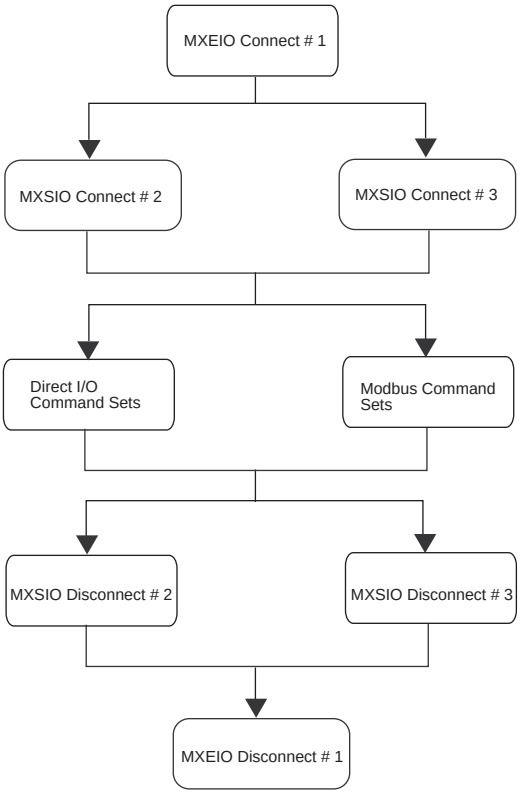
Connecting to the ioLogik E2000 and Attached RS-485 I/O

It is possible to combine Ethernet and RS-485 connections, as shown in the following figure.

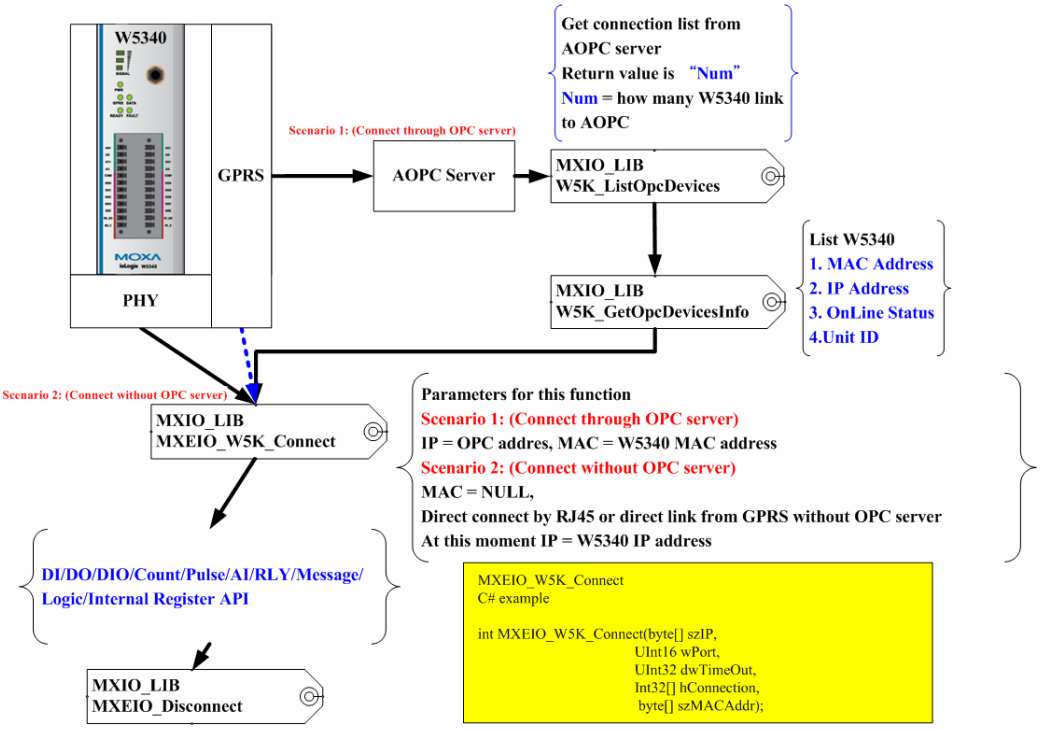


Program Flow V.

Connecting to the ioLogik E2000 Ethernet I/O and Attached RS-485 I/O.



Connecting to W5000 series GPRS I/O



Modbus Command Sets vs. Direct I/O Command Sets

The MXIO Library offers two options for accessing I/O data from ioLogik 4000, E2000, and R2000 I/O.

Modbus Command Sets

The ioLogik 4000, E4200, E2000, and R2000 I/O use Modbus/TCP and Modbus/RTU to communicate with host computers. MXIO Library includes Modbus command sets that use the Modbus protocol data format to access 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

As an alternative to the complex data structure of Modbus, MXIO library also provides direct I/O command sets for a more intuitive method of obtaining I/O data. With direct I/O command sets, each I/O point or channel can be accessed using the physical slot number and channel number. This allows users to obtain I/O data quickly and easily.

3
MXIO API Overview

MXIO API is organized into five types of commands:

- **System Command Sets**
  - RS-485/RS-232 I/O Connect Commands
  - Ethernet I/O Connect Commands
  - General Commands
  - Commands for ioLogik E2000, R2000
  - Commands for ioLogik NA4000
  - Commands for ioLogik E4200
  - Commands for ioLogik W5000
- **Modbus Command Sets**
- **Direct I/O Command Sets**
  - Digital Input Commands
  - Digital Input Commands for ioLogik E2000, R2000
  - Digital Input / Output Mode Commands for ioLogik E2000
  - Counter Commands for ioLogik E2000, R2000
  - Digital Output Commands
  - Digital Output Commands for ioLogik E2000, R2000
  - Digital Output Commands for ioLogik 4000
  - Digital Output Commands for ioLogik E4200
  - Digital Output Commands for ioLogik W5000
  - Pulse Output Commands for ioLogik E2000, R2000
  - Analog Input CommandsAnalog Input Commands for ioLogik E2000, R2000
  - Analog Input Commands for ioLogik W5000
  - Analog Input Commands for ioLogik E1200
  - Analog Output Commands
  - Analog Output CommandsAnalog Output Commands for ioLogik E2000, R2000
  - Analog Output Commands for ioLogik NA4000
  - Analog Output Commands for ioLogik E4200
  - Relay Commands for ioLogik 2000
  - RTD Commands
  - RTD Commands for ioLogik E4200
  - Thermocouple Commands
  - TC Commands for ioLogik E4200
- **Click&Go Logic Commands**
  - Commands for ioLogik E4200
- **Active I/O Message Commands**
  - Commands for ioLogik W5000

System Command Sets

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

Function Name
MXSIO_OpenCommport
MXSIO_CloseCommport
MXSIO_Connect
MXSIO_Disconnect

Ethernet I/O Connect Commands

Function Name
MXEIO_Init
MXEIO_Exit
MXEIO_Connect
MXEIO_Disconnect
MXEIO_CheckConnection

General Commands

Function Name
MXIO_GetDllVersion
MXIO_GetDllBuildDate
MXIO_GetModuleType
MXIO_ReadFirmwareRevision
MXIO_ReadFirmwareDate
MXIO_Restart
MXIO_Reset

Commands for ioLogik E2000, R2000

Function Name
Module2K_GetSafeStatus
Module2K_ClearSafeStatus
Module2K_GetInternalReg
Module2K_SetInternalReg
Module2K_GetInternalRegs
Module2K_SetInternalRegs

Commands for ioLogik NA4000

Function Name
Adp4K_ReadFirmwareRevision
Adp4K_ReadStatus
Adp4K_ClearStatus
Adp4K_ReadFirmwareDate
Adp4K_ReadSlotAmount
Adp4K_ReadAlarmedSlot
Adp4K_ReadAlarmedSlot

Commands for ioLogik E4200

Function Name
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

Commands for ioLogik W5000

Function Name
W5K_GetInternalRegs
W5K_SetInternalRegs
W5K_GetGprsSignal
W5K_ListOpcDevices
W5K_GetOpcDevicesInfo
W5K_GetOpcAliasName
W5K_GetSafeStatus
W5K_ClearSafeStatus

Commands for ioLogik E1200

Function Name
E1K_GetSafeStatus
E1K_ClearSafeStatus

Modbus Command Sets

Function Name
MXIO_ReadCoils
MXIO_WriteCoils
MXIO_ReadRegs
MXIO_WriteRegs

Direct I/O Command Sets

Digital Input Commands

Function Name
DI_Reads
DI_Read
E42_DI_Reads (For E4200 only)

Digital Input Commands for ioLogik E2000, R2000

Function Name
DI2K_GetModes
DI2K_SetModes
DI2K_GetMode
DI2K_SetMode
DI2K_GetFilters
DI2K_SetFilters
DI2K_GetFilter
DI2K_SetFilter

Digital Input / Output Mode Commands for ioLogik E2000

Function Name
DIO2K_GetIOMode
DIO2K_SetIOMode
DIO2K_GetIOModes
DIO2K_SetIOModes

Digital Input Commands for ioLogik E4200

Function Name
E42_DI_Reads

Digital Input Commands for ioLogik W5000

Function Name
W5K_DI_Reads W5K_DI_Reads
W5K_DI_GetModes W5K_DI_GetModes
W5K_DI_SetModes W5K_DI_SetModes
W5K_DI_GetFilters W5K_DI_GetFilters
W5K_DI_SetFilters

Digital Input Commands for ioLogik E1200

Function Name
E1K_DI_Reads
E1K_DI_GetModes
E1K_DI_SetModes
E1K_DI_GetFilters
E1K_DI_SetFilters

Counter Commands for ioLogik E2000, R2000

Function Name
Cnt2K_Reads
Cnt2K_Clears
Cnt2K_Read
Cnt2K_Clear
Cnt2K_GetOverflows
Cnt2K_ClearOverflows
Cnt2K_GetOverflow
Cnt2K_ClearOverflow
Cnt2K_GetFilters
Cnt2K_SetFilters
Cnt2K_GetFilter
Cnt2K_SetFilter
Cnt2K_GetStartStatuses
Cnt2K_SetStartStatuses
Cnt2K_GetStartStatus
Cnt2K_SetStartStatus
Cnt2K_GetTriggerTypes
Cnt2K_SetTriggerTypes
Cnt2K_GetTriggerType

Function Name
Cnt2K_SetTriggerType
Cnt2K_GetPowerOnValues
Cnt2K_SetPowerOnValues
Cnt2K_GetPowerOnValue
Cnt2K_SetPowerOnValue
Cnt2K_GetSafeValues
Cnt2K_SetSafeValues
Cnt2K_GetSafeValue
Cnt2K_SetSafeValue
Cnt2K_GetTriggerTypeWords
Cnt2K_SetTriggerTypeWords
Cnt2K_GetTriggerTypeWord
Cnt2K_SetTriggerTypeWord
Cnt2K_GetSaveStatusesOnPowerFail
Cnt2K_SetSaveStatusesOnPowerFail

Counter Commands for ioLogik W5000

Function Name
W5K_Cnt_Reads
W5K_Cnt_Clears
W5K_Cnt_GetOverflows
W5K_Cnt_ClearOverflows
W5K_Cnt_GetFilters
W5K_Cnt_SetFilters
W5K_Cnt_GetStartStatuses
W5K_Cnt_SetStartStatuses
W5K_Cnt_GetTriggerTypes
W5K_Cnt_SetTriggerTypes
W5K_Cnt_GetPowerOnValues
W5K_Cnt_SetPowerOnValues
W5K_Cnt_GetSafeValues
W5K_Cnt_SetSafeValues
W5K_Cnt_GetTriggerTypeWords
W5K_Cnt_SetTriggerTypeWords
W5K_Cnt_GetSaveStatusesOnPowerFail
W5K_Cnt_SetSaveStatusesOnPowerFail

Counter Commands for ioLogik E1200

Function Name
E1K_Cnt_Reads
E1K_Cnt_Clears
E1K_Cnt_GetOverflows
E1K_Cnt_ClearOverflows
E1K_Cnt_GetFilters
E1K_Cnt_SetFilters
E1K_Cnt_GetStartStatuses
E1K_Cnt_SetStartStatuses
E1K_Cnt_GetPowerOnValues
E1K_Cnt_SetPowerOnValues
E1K_Cnt_GetSafeValues
E1K_Cnt_SetSafeValues
E1K_Cnt_GetTriggerTypeWords
E1K_Cnt_SetTriggerTypeWords
E1K_Cnt_GetSaveStatusesOnPowerFail
E1K_Cnt_SetSaveStatusesOnPowerFail

Digital Output Commands

Function Name
DO_Reads
DO_Read
DO_Writes
DO_Write
DO_GetSafeValues
DO_SetSafeValues
DO_GetSafeValue
DO_SetSafeValue
DO_GetSafeValues_W
DO_SetSafeValues_W

Digital Output Commands for ioLogik E2000, R2000

Function Name
DO2K_GetModes
DO2K_SetModes
DO2K_GetMode
DO2K_SetMode
DO2K_GetPowerOnValues
DO2K_SetPowerOnValues
DO2K_GetPowerOnValue
DO2K_SetPowerOnValue
DO2K_GetPowerOnSeqDelaytimes
DO2K_SetPowerOnSeqDelaytimes

Digital Output Commands for ioLogik 4000

Function Name
DO4K_GetSafeActions
DO4K_SetSafeActions
DO4K_GetSafeAction
DO4K_SetSafeAction

Digital Output Commands for ioLogik E4200

Function Name
E42_DO_GetSafeActions
E42_DO_SetSafeActions
E42_DO_GetPowerOnValues
E42_DO_SetPowerOnValues
E42_DO_Reads
E42_DO_Writes
E42_DO_GetFaultValues
E42_DO_SetFaultValues

Digital Output Commands for ioLogik W5000

Function Name
W5K_DO_Reads
W5K_DO_Writes
W5K_DO_GetSafeValues
W5K_DO_SetSafeValues
W5K_DO_GetModes
W5K_DO_SetModes
W5K_DO_GetPowerOnValues
W5K_DO_SetPowerOnValues

Digital Output Commands for ioLogik E1200

Function Name
E1K_DO_Reads
E1K_DO_Writes
E1K_DO_GetSafeValues_W
E1K_DO_SetSafeValues_W
E1K_DO_GetModes
E1K_DO_SetModes
E1K_DO_GetPowerOnValues
E1K_DO_SetPowerOnValues
E1K_DO_GetPowerOnSeqDelayTimes
E1K_DO_SetPowerOnSeqDelayTimes

Pulse Output Commands for ioLogik E2000, R2000

Function Name
Pulse2K_GetSignalWidths
Pulse2K_SetSignalWidths
Pulse2K_GetSignalWidth
Pulse2K_SetSignalWidth
Pulse2K_GetSignalWidths32: Only for ioLogik E2260
Pulse2K_SetSignalWidths32: Only for ioLogik E2260
Pulse2K_GetSignalWidth32: Only for ioLogik E2260
Pulse2K_SetSignalWidth32: Only for ioLogik E2260
Pulse2K_GetOutputCounts
Pulse2K_SetOutputCounts
Pulse2K_GetOutputCount
Pulse2K_SetOutputCount
Pulse2K_GetStartStatuses
Pulse2K_SetStartStatuses
Pulse2K_GetStartStatus
Pulse2K_SetStartStatus
Pulse2K_GetPowerOnValues
Pulse2K_SetPowerOnValues
Pulse2K_GetPowerOnValue
Pulse2K_SetPowerOnValue
Pulse2K_GetSafeValues
Pulse2K_SetSafeValues
Pulse2K_GetSafeValue
Pulse2K_SetSafeValue

Pulse Output Commands for ioLogik W5000

Function Name
W5K_Pulse_GetSignalWidths32
W5K_Pulse_SetSignalWidths32
W5K_Pulse_GetOutputCounts
W5K_Pulse_SetOutputCounts
W5K_Pulse_GetStartStatuses
W5K_Pulse_SetStartStatuses
W5K_Pulse_GetPowerOnValues
W5K_Pulse_SetPowerOnValues
W5K_Pulse_GetSafeValues
W5K_Pulse_SetSafeValues

Pulse Output Commands for ioLogik E1200

Function Name
E1K_Pulse_GetSignalWidths
E1K_Pulse_SetSignalWidths
E1K_Pulse_GetOutputCounts
E1K_Pulse_SetOutputCounts
E1K_Pulse_GetStartStatuses
E1K_Pulse_SetStartStatuses
E1K_Pulse_GetPowerOnValues
E1K_Pulse_SetPowerOnValues
E1K_Pulse_GetSafeValues
E1K_Pulse_SetSafeValues

Digital Input & Output mode change Commands for ioLogik 5000

Function Name
W5K_DIO_GetIOModes
W5K_DIO_SetIOModes

Digital Input & Output mode change Commands for ioLogik 5000

Function Name
E1K_DIO_GetIOModes

Analog Input Commands

Function Name
AI_Reads
AI_Read
AI_ReadRaws
AI_ReadRaw
E42_AI_Reads (for E4200 only)

Analog Input Commands for ioLogik E2000, R2000

Function Name
AI2K_ReadMins
AI2K_ReadMinRaws
AI2K_ResetMins
AI2K_ReadMin
AI2K_ReadMinRaw
AI2K_ResetMin
AI2K_ReadMaxs
AI2K_ReadMaxRaws
AI2K_ResetMaxs
AI2K_ReadMax
AI2K_ReadMaxRaw
AI2K_ResetMax
AI2K_GetRanges
AI2K_SetRanges
AI2K_GetRange
AI2K_SetRange
AI2K_GetChannelStatus
AI2K_SetChannelStatus
AI2K_GetChannelStatuses
AI2K_SetChannelStatuses

Analog Input Commands for ioLogik W5000

Function Name
W5K_AI_Reads
W5K_AI_ReadRaws
W5K_AI_ReadMins
W5K_AI_ReadMinRaws
W5K_AI_ResetMins
W5K_AI_ReadMaxs
W5K_AI_ReadMaxRaws
W5K_AI_ResetMaxs
W5K_AI_GetRanges
W5K_AI_SetRanges
W5K_AI_GetChannelStatuses
W5K_AI_SetChannelStatuses

Analog Input Commands for ioLogik E1200

Function Name
E1K_AI_Reads
E1K_AI_ReadRaws
E1K_AI_ReadMins
E1K_AI_ReadMinRaws
E1K_AI_ResetMins
E1K_AI_ReadMaxs
E1K_AI_ReadMaxRaws
E1K_AI_ResetMaxs
E1K_AI_GetRanges
E1K_AI_GetChannelStatuses
E1K_AI_SetChannelStatuses

Analog Output Commands

Function Name
AO_Reads
AO_Writes
AO_Read
AO_Write
AO_ReadRaws
AO_WriteRaws
AO_ReadRaw
AO_WriteRaw
AO_GetSafeValues
AO_SetSafeValues
AO_GetSafeValue
AO_SetSafeValue
AO_GetSafeRaws
AO_SetSafeRaws
AO_GetSafeRaw
AO_SetSafeRaw

Analog Output Commands for ioLogik E2000, R2000

Function Name
AO2K_GetRanges
AO2K_SetRanges
AO2K_GetRange
AO2K_SetRange
AO2K_GetPowerOnValues
AO2K_SetPowerOnValues
AO2K_GetPowerOnValue
AO2K_SetPowerOnValue
AO2K_GetPowerOnRaws
AO2K_SetPowerOnRaws
AO2K_GetPowerOnRaw
AO2K_SetPowerOnRaw

Analog Output Commands for ioLogik NA4000

Function Name
AO4K_GetSafeActions
AO4K_SetSafeActions
AO4K_GetSafeAction
AO4K_SetSafeAction

Analog Output Commands for ioLogik E4200

Function Name
E42_AO_GetSafeActions
E42_AO_SetSafeActions
E42_AO_GetPowerOnValues
E42_AO_SetPowerOnValues
E42_AO_Reads
E42_AO_Writes
E42_AO_ReadRaws
E42_AO_WriteRaws
E42_AO_GetFaultValues
E42_AO_SetFaultValues

Relay Commands for ioLogik 2000

Function Name
RLY2K_GetResetTime
RLY2K_TotalCntRead
RLY2K_TotalCntReads
RLY2K_CurrentCntRead
RLY2K_CurrentCntReads
RLY2K_ResetCnt
RLY2K_ResetCnts

Relay Commands for ioLogik W5000

Function Name
W5K_RLY_GetResetTime
W5K_RLY_TotalCntReads
W5K_RLY_CurrentCntReads
W5K_RLY_ResetCnts

Relay Commands for ioLogik E1200

Function Name
E1K_RLY_TotalCntReads

RTD Commands

Function Name
RTD_Reads
RTD_Read
RTD_ReadRaws
RTD_ReadRaw
RTD2K_ResetMin
RTD2K_ResetMins
RTD2K_ResetMax
RTD2K_ResetMaxs
RTD2K_ReadMinRaw
RTD2K_ReadMinRaws
RTD2K_ReadMaxRaw
RTD2K_ReadMaxRaws
RTD2K_ReadMin
RTD2K_ReadMins
RTD2K_ReadMax
RTD2K_ReadMaxs
RTD2K_GetStartStatus
RTD2K_SetStartStatus
RTD2K_GetStartStatuses
RTD2K_SetStartStatuses
RTD2K_GetSensorType
RTD2K_SetSensorType
RTD2K_GetSensorTypes
RTD2K_SetSensorTypes
RTD2K_GetEngUnit
RTD2K_SetEngUnit
RTD2K_GetEngUnits
RTD2K_GetMathPar
RTD2K_SetMathPar
RTD2K_GetMathPars
RTD2K_SetMathPars

RTD Commands for ioLogik E4200

Function Name
E42_RTD_Reads
E42_RTD_ReadRaws
E42_RTD_GetEngUnit
E42_RTD_SetEngUnit
E42_RTD_GetSensorType
E42_RTD_SetSensorType

Thermocouple Commands

Function Name
TC_Reads
TC_Read
TC_ReadRaws
TC_ReadRaw

TC Commands for ioLogik E4200

Function Name
E42_TC_Reads
E42_TC_ReadRaws
E42_TC_GetEngUnit
E42_TC_SetEngUnit
E42_TC_GetSensorType
E42_TC_SetSensorType

Click&Go Logic Commands

Commands for ioLogik E2000

Function Name
Logic2K_GetStartStatus
Logic2K_SetStartStatus

Commands for ioLogik E4200

Function Name
E42_Logic_GetStartStatus
E42_Logic_SetStartStatus

Commands for ioLogik W5000

Function Name
W5K_Logic_GetStartStatus
W5K_Logic_SetStartStatus

Active I/O Message Commands

Commands for ioLogik E2000

Function Name
Message2K_Start
Message2K_Stop

Command for ioLogik E4200

Function Name
E42_Message_Start
E42_Message_Stop

Commands for ioLogik W5000

Function Name
W5K_Message_Start
W5K_Message_Stop

4
System Command Sets

System commands include functions that initialize the connection between a host computer and the I/O. In addition, system commands include functions for obtaining hardware and status information for the I/O itself.

The following topics are covered:

- **RS-232/RS-485 I/O Connect Commands**
- **Ethernet I/O Connect Commands**
- **General Commands**
- **Commands for ioLogik E2000, R2000**
- **Commands for ioLogik NA4000**
- **Commands for ioLogik E4200**
- **Commands for ioLogik W5000**

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

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);**	
Arguments	*szCommport:*	Name of the common port, e.g., “ttyM0”.
	dwBaudrate:	Baudrate, from 4,800 to 115,200 bps. Does not support 1200 or 2400 bps.
	bytDataFormat:	Transmission data format. bit_cnt (bit 0, 1) = 0x00 = bit_5 0x01 = bit_6 0x02 = bit_7 0x03 = bit_8 stop_cnt (bit 2) = 0x00 = stop_1 0x04 = stop_2 parity (bit 3, 4, 5) = 0x00 = none 0x08 = odd 0x18 = even 0x28 = mark 0x38 = space
	dwTimeout:	Timeout in milliseconds for serial adapter communication
	hCommport:	Handle of the opened COM port.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

MXSIO_CloseCommport	This closes the COM port; the COM port handle will be invalid.	
C/C++	**int MXSIO_CloseCommport (int hCommport);**	
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 E2000 handle, users must use this function to establish an I/O device handle for each RS-485 or RS-232 I/O. A COM port handle can connect one RS-232 I/O or up to 32 RS-485 I/O.
C/C++	**int MXSIO_Connect (int hCommport,** **BYTE bytUnitID,** **BYTE bytTransmissionMode,** **int *hConnection);**
Arguments	*hCommport:* I/O over serial interface will get a serial handle. I/O server over Ethernet interface will get an Ethernet handle. For more details, please see the Program Flow IV and Program Flow V in Chapter 2.
	bytUnitID: Modbus unit ID of the I/O, from 01 to 99.
	bytTransmissionMod Modbus transmission format.
	ex: 0: RTU mode 1: ASCII mode. Note that the protocol settings must agree with the hardware settings on the ioLogik 4000 I/O, and the ioLogik R2000 only supports RTU.
	hConnection: Handle of the I/O device connection.
Return Value	Succeed MXIO_OK.
	Fail Refer to Return Codes.

MXSIO_Disconnect	This will disconnect the RS-485/232 I/O. The I/O handle will be invalid.
C/C++	**int MXSIO_Disconnect (int hConnection);**
Arguments	*hConnection:* Handle of the I/O device connection.
Return Value	Succeed MXIO_OK.
	Fail Refer to Return Codes.

Ethernet I/O Connect Commands

MXEIO_Init	Initiate the socket, only for windows OS.	
C/C++	**int MXEIO_Init ();**	
Arguments	*None*	
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

MXEIO_Exit	To terminates use of the socket, only for windows OS.
C/C++	**int MXEIO_Exit ();**
Arguments	*None*
Return Value	*None*.

MXEIO_Connect	This function establishes a connection to the port of the Ethernet I/O. 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);**	
Arguments	*szIP:*	IP address of the Ethernet I/O to be connected.
	wPort:	TCP port number of Ethernet I/O. Please use 502 for the ioLogik 4000 and E2000.
	dwTimeOut:	Timeout value in milliseconds for establishing a network connection with the ioLogik Ethernet adapter.
	hConnection:	Handle of the I/O device connection.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

MXEIO_Disconnect	This will close the connection with the Ethernet I/O; the handle will be invalid.	
C/C++	**int MXEIO_Disconnect (int hConnection);**	
Arguments	*hConnection:*	Handle of the connection.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

MXEIO_CheckConnection	This establishes a connection to the port of the Ethernet I/O. Once the connection is established, a handle will be returned for additional functions.	
C/C++	**int MXEIO_CheckConnection(inth DWORD BYTE**	**Connection, dwTimeOut, *bytStatus);**
Arguments	*hConnection:*	Handle of the I/O connection.
	dwTimeOut:	Timeout value in milliseconds for network connection.
	bytStatus	Connection status. 0: Check connection ok. 1: Check connection fail. 2: Check connection timeout.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

General Commands

MXIO_GetDllVersion	This will get the DLL version.	
C/C++	**int MXIO_GetDllVersion ();**	
Arguments	*None*	
Return Value	Succeed	The DLL version. Version 2.0.0.0 is expressed as value 0x2000.

MXIO_GetDllBuildDate	This will get the DLL release date.	
C/C++	**int MXIO_GetDllBuildDate();**	
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 I/O modules.	
C/C++	**int MXIO_GetModuleType (int hConnection, BYTE bytSlot, WORD * wType);**	
Arguments	*hConnection:*	I/O handle for a connection.
	bytSlot:	Slot number of the I/O to be checked, from 0 to 32. The ioLogik 4000 network adapter's slot number is always 0. This parameter is inactive for ioLogik E2000 and R2000 I/O.
	wType:	A pointer to the model name. For the model name NA-4010, the value would be 0x4010 Refer to the Product Model and ID Reference Table for more information.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

MXIO_ReadFirmwareRevision	This function reports the firmware revision for the ioLogik 4000 (network adapter), E2000, or R2000.	
C/C++	**int MXIO_ReadFirmwareRevision (int hConnection, BYTE bytRevision[ ]);**	
Arguments	*hConnection:*	I/O handle for a connection.
	bytRevision:	Stores the 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 the firmware release date for the ioLogik 4000 (network adapter), E2000, or R2000.	
C/C++	**int MXIO_ReadFirmwareDate (int hConnection, WORD wDate[ ]);**	
Arguments	*hConnection:*	I/O handle for a connection.
	wDate:	Firmware release date. For a firmware release date of July 5, 2005, Word 0 = 0x0705 and Word 1 = 0x2005.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

MXIO_Restart	This function is used to restart the I/O.	
C/C++	**int MXIO_Restart (int hConnection);**	
Arguments	*hConnection:*	I/O 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.	
C/C++	**int MXIO_Reset (int hConnection);**	
Arguments	*hConnection:*	I/O handle for a connection.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Commands for ioLogik E2000, R2000

Module2K_GetSafeStatus	This function code is used to get the safe status of an ioLogik E2000 or R2000 I/O.	
C/C++	**int Module2K_GetSafeStatus (int hConnection, WORD * wStatus);**	
Arguments	*hConnection:*	The handle for a connection.
	wStatus:	A pointer to 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 get the safe status of an ioLogik E2000 or R2000 I/O.	
C/C++	**int Module2K_ClearSafeStatus (int hConnection);**	
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 the ioLogik 2000 Module.	
C/C++	**int Module2K_GetInternalReg (int hConnection, BYTE bytChannel, WORD * wValue);**	
Arguments	*hConnection:* *bytChannel:* *wValue*	The handle for a connection The specific channel to be read. 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 get the internal register of the ioLogik 2000 Module.	
C/C++	**int Module2K_SetInternalReg (int hConnection, BYTE bytChannel, WORD * wValue);**	
Arguments	*hConnection:* *bytChannel:* *wValue*	The handle for a connection The specific channel to be read. 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 register of the ioLogik 2000 Module.	
C/C++	**int Module2K_GetInternalRegs (int hConnection, BYTE bytStartChannel, BYTE bytCount, WORD * wValue[ ]);**	
Arguments	*hConnection:*	The handle for a connection
	bytStartChannel:	Specifies the starting channel
	bytCount:	The number of channels to bet read (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_GetInternalRegs	This function code is used to get the internal register of the ioLogik 2000 Module.	
C/C++	**int Module2K_GetInternalRegs (int hConnection, BYTE bytChannel, BYTE bytCount, WORD wValue);**	
Arguments	*hConnection:*	The handle for a connection
	bytStartChannel:	Specifies the starting channel
	bytCount:	The number of channels to bet read (Up to 24).
Return Value	*wValue*	represents the value of the starting channel. The values are 0 - 255
	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Module2K_SetInternalRegs	This function code is used to get the internal register of the ioLogik 2000 Module.	
C/C++	**int Module2K_SetInternalRegs (int hConnection, BYTE bytStartChannel, BYTE bytCount, WORD wValue[ ]);**	
Arguments	*hConnection:*	The handle for a connection
	bytStartChannel:	Specifies the starting channel
	bytCount:	The number of channels to bet read (Up to 24).
Return Value	*wValue*	represents the value of the starting channel. The values are 0 - 255
	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Commands for ioLogik NA4000

Adp4K_ReadFirmwareRevision	This function code is used to read the firmware revision.	
C/C++	**int Adp4K_ReadFirmwareRevision (int hConnection, WORD * wRevision);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	wRevision:	Stores the firmware revision. For revision 1.01, the value will be 0X0101.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Adp4K_ReadStatus	This function code is used to read the status of the ioLogik 4000 adapter.	
C/C++	**int Adp4K_ReadStatus (int hConnection ,** **WORD *wBusStatus,** **WORD *wFPStatus,** **WORD *wEWStatus,** **WORD *wESStatus,** **WORD *wECStatus);**	
Arguments	*hConnection:*	The handle for an I/O 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: 24 VDC Field Power On. 1: 24 VDC Field Power Off.
	wEWStatus:	Stores the Watchdog status in numerical format. The values are: 0: No Error. 1: Watchdog activated.
	wESStatus:	Stores the Modbus Setup Error status in numeric data format, supported by NA-4020 & NA-4021 only. 0: No Error. 1: Modbus Setup Error.
	wECStatus:	Stores the Modbus Checksum Error status, supported by NA-4020 & NA-4021 only. 0: No Error. 1: Three continuous CRC/LRC checksum errors have occurred since last restart, last clear counters operation, or last 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);**	
Arguments	*hConnection:*	The handle for an I/O connection.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Adp4K_ReadFirmwareDate	This function code is used to read the firmware release date.	
C/C++	**int Adp4K_ReadFirmwareDate (int hConnection, WORD wDate[ ]);**	
Arguments	*hConnection:*	The handle for a connection.
	wDate:	An array that stores the firmware release date. For a firmware release date of July 5, 2005, wDate[0] will be 0X0705 and wDate[1] will be 0X2005.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Adp4K_ReadSlotAmount	This function code is used to read the number of expansion slots.	
C/C++	**int Adp4K_ReadSlotAmount (int hConnection, WORD *wAmount);**	
Arguments	*hConnection:*	The handle for a connection.
	wAmount:	A pointer to the number of expansion slots.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Adp4K_ReadAlarmedSlot	This function code is used to read the number of expansion slots.	
C/C++	**int Adp4K_ReadAlarmedSlot (int hConnection, DWORD *dwAlarm);**	
Arguments	*hConnection:*	The handle for a connection.
	dwAlarm:	A pointer to 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.

Commands for ioLogik E4200

E42_ReadFirmwareRevision	This function reports the firmware revision for the ioLogik E4200 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 (C) 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 for the ioLogik E4200 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 the 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 E4200 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.
	WAmount:	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 E4200 Active Ethernet network adaptors. When changing or removing slots from the E4200, you need to reboot the E4200 after clearing the 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 E4200 Active Ethernet network adaptors.	
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 bet read (up to 80.)
	wValue:	Represents the value of the starting channel. The values are 0 to 255.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

E42_SetInternalRegs	This function code is used to set the internal registers of the ioLogik E4200 network adaptors.	
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 an I/O device connection.
	bytStartchannel:	Specifies the starting channel.
	bytCount:	The number of channels to retrieve (up to 80).
	wValue:	An array that stores contiguous internal register. The values are 0 to 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 the ioLogik E4200 Active Ethernet network adaptors.	
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 an I/O device connection.
	bytStartchannel:	Specifies the starting channel.
	bytCount:	The number of channels to retrieve (up to 80).
	wValue:	Represents the value of the starting channel. The values are 0 to 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 the ioLogik E4200 Active Ethernet network adaptors (not saved 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 an I/O device connection.
	bytStartchannel:	Specifies the starting channel.
	bytCount:	The number of channels to retrieve (up to 80).
	wValue:	An array that stores contiguous internal register. The values are 0 to 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 E4200 Active Ethernet 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 the ioLogik E4200 Active Ethernet network adaptors.	
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 the ioLogik E4200 Active Ethernet network adaptors.	
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_ClearWatchdogStatus (int hConnection);**
Visual Basic	Declare Function E42_ClearWatchdogStatus 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.

Commands for ioLogik W5000

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 bet read. (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_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 bet read. (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_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:* A 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_GetOpcAliasName 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 amount of ioLogik W5000 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 ,** **ByVal szDeviceInfo 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. *wDeviceCount:* Total amount for the I/O device connected to A-OPC Server. *szDeviceCount:* 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 Value	Succeed MXIO_OK. Fail Refer to Return Codes.

W5K_GetOpcAliasName	This function get A-OPC server alias name of the specific ip address device that running A-OPC server.
C/C++	**int W5K_GetOpcAliasName(char * szIP,** **DWORD dwTimeOut,** **Char szAliasName[]);**
Visual Basic	**Declare Function W5K_GetOpcAliasName 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 Value	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:* A 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.

5
Modbus Command Sets

Four basic Modbus commands are provided in the Modbus command set. However, you must refer to the Modbus reference table in ioAdmin to extract data from the Modbus packet.

MXIO_ReadCoils	This function reads the on/off status for a contiguous group of coils on the same I/O.	
C/C++	**int MXIO_ReadCoils (int hConnection,** **BYTE bytCoilType,** **WORD wStartCoil,** **WORD wCount,** **BYTE bytCoils[ ]);**	
Arguments	*hConnection:*	Handle for the I/O 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 coils.	
C/C++	**int MXIO_WriteCoils (int WORD WORD BYTE**	**hConnection, wStartCoil, wCount, bytCoils[ ]);**
Arguments	*hConnection:*	Handle for the I/O 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 holding registers.	
C/C++	**int MXIO_ReadRegs (int BYTE WORD WORD WORD**	**hConnection, bytRegisterType, wStartRegister, wCount, wRegister[ ]);**
Arguments	*hConnection:*	The handle for an I/O 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 coils 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 holding registers.	
C/C++	**int MXIO_WriteRegs (int hConnection, WORD wStartRegister, WORD wCount, WORD wRegister[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	wStartRegister:	Specifies the starting register address. The address begins at 0.
	wCount:	The number of coils to be written.
	wRegister:	An array that stores the register values.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

6
Direct I/O Command Sets

Direct I/O command sets provide an intuitive way to access the data for each channel.

Digital Input Commands

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 BYTE BYTE BYTE DWORD**	**hConnection, bytSlot, bytStartChannel, bytCount, * dwValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dwValue:	A pointer to the contiguous D/I channel's values; each bit holds the value of one channel. 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 BYTE BYTE BYTE**	**hConnection, bytSlot, bytChannel, *bytValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel:	The desired channel.
	bytValue:	A pointer to a specific D/I channel's status.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Digital Input Commands for ioLogik E4200

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 on the 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.

Commands for ioLogik W5000

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 (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_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 (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:	A 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_GetOpcDevicesInfo	This function gets ioLogik W5000 device information that links 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, ByVal **szDeviceInfo** 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. *wDeviceCount:* Total number of I/O devices connected to the A-OPC server (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 Value	Succeed MXIO_OK. Fail Refer to Return Codes.

W5K_GetOpcHostName	This function gets the A-OPC server alias name of the specific ip address device 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 Value	Succeed MXIO_OK. Fail Refer to Return Codes.

W5K_ListOpcDevices	This function gets the number of ioLogik W5000 units that link to the 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. *wDeviceConut*: Total number of I/O devices connected to the A-OPC Server.
Return Value	Succeed MXIO_OK. Fail Refer to Return Codes.

W5K_GetSafeStatus	This function code is used to get the safe status of the 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:* A 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 the 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.

Commands for ioLogik E1200

E1K_GetSafeStatus	This function code is used to get the safe status of the 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:* A 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 the 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.

Digital Input Commands for ioLogik E2000, R2000

DI2K_GetModes	This function code is used to get the mode of contiguous D/I channels.	
C/C++	**int DI2K_GetModes (int BYTE BYTE WORD**	**hConnection, bytStartChannel, bytCount, wMode[ ]);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wMode:	An array that stores the modes of the contiguous D/I channels. wMode[0] represents the value of the starting channel. The values are: 0: D/I Mode. 1: Counter 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 BYTE BYTE WORD**	**hConnection, bytStartChannel, bytCount, wMode[ ]);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	wMode:	An array that stores the modes of the contiguous D/I channels. wMode [0] represents the value of the starting channel. The values are: 0: D/I Mode. 1: Counter 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 BYTE WORD**	**hConnection, bytChannel, * wMode);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wMode:	A pointer to the mode of the desired D/I channel. The values are: 0: D/I Mode. 1: Counter 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 BYTE WORD**	**hConnection, bytChannel, wMode);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wMode:	A pointer to the mode of the desired D/I channel. The values are: 0: D/I Mode. 1: Counter 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 BYTE BYTE WORD**	**hConnection, bytStartChannel, bytCount, wFilter[ ]);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wFilter:	An array that stores the filter values of the contiguous D/I channels. 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 BYTE BYTE WORD**	**hConnection, bytStartChannel, bytCount, wFilter[ ]);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	wFilter:	An array that stores the filter values of the contiguous D/I channels. 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 DI2K_GetFilter (int BYTE WORD**	**hConnection, bytChannel, * wFilter);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wFilter:	A pointer to the filter value of the desired D/I channel.
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_GetFilter (int BYTE WORD**	**hConnection, bytChannel, wFilter);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wFilter:	A pointer to the filter value of the desired D/I channel.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Digital Input Commands for ioLogik E4200

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);**
Visul 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.

Digital Inputs of ioLogik W5000

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 BYTE BYTE DWORD**	**hConnection, bytStartChannel, bytCount, * dwValue);**
Visul Basic	**Declare Function W5K_DI_Reads Lib "MXIO.dll" (ByVal ByVal ByVal nValue**	**hConnection As Long, bytStartChannel As Byte, bytCount As Byte, 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 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);**
Visul 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. *bytCount:* The number of channels to bet read. *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[ ]);**
Visul 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 bet read. *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[ ]);**
Visul 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 bet read. *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[ ]);**
Visul 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.

Digital Input Commands for the ioLogik E1200

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);**
Visul 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[ ]);**
Visul 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. *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[ ]);**
Visul 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 set. *wMode:* An array that stores contiguous D/I channel's modes. 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[ ]);**
Visul 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. *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[ ]);**
Visul 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 set. *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.

Counter Commands for ioLogik E2000, R2000

Cnt2K_Reads	This function code is used to read the counter values of contiguous D/I channels in Counter mode.	
C/C++	**int Cnt2K_Reads (int BYTE BYTE DWORD**	**hConnection, bytStartChannel, bytCount, dwValue[ ]);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dwValue:	An array that stores the counter values of the contiguous channels, 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 the counter values of contiguous D/I channels in Counter mode.	
C/C++	**int Cnt2K_Clears (int BYTE BYTE**	**hConnection, bytStartChannel, bytCount);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to clear.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Cnt2K_Read	This function code is used to read the counter value of a specific D/I channel in Counter mode.	
C/C++	**int Cnt2K_Read (int BYTE DWORD**	**hConnection, bytChannel, * dwValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	dwValue:	A pointer to the counter value of the desired channel.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Cnt2K_Clear	This function code is used to clear the counter value of a specific D/I channel in Counter mode.
C/C++	**int Cnt2K_Clear (int hConnection,** **BYTE bytChannel);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytChannel: The desired channel.
Return Value	Succeed MXIO_OK.
	Fail Refer to Return Codes.

Cnt2K_GetOverflows	This function code is used to get the overflow statuses of contiguous D/I channels in Counter mode.
C/C++	**int Cnt2K_GetOverflows (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD * dwStatus);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytStartChannel: Specifies the starting channel.
	bytCount: The number of channels to read.
	dwStatus: A pointer to the overflow status of the contiguous channels; each bit holds the 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 the overflow statuses of contiguous D/I channels in Counter mode.
C/C++	**int Cnt2K_ClearOverflows (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytStartChannel: Specifies the starting channel.
	bytCount: The number of channels to clear.
Return Value	Succeed MXIO_OK.
	Fail Refer to Return Codes.

Cnt2K_GetOverflow	This function code is used to get the overflow status of a specific D/I channel in Counter mode.	
C/C++	**int Cnt2K_GetOverflow(int hConnection,** **BYTE bytChannel,** **BYTE * bytStatus);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	bytStatus:	A pointer to the overflow status of the desired 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 the overflow status of a specific D/I channel in Counter mode.	
C/C++	**int Cnt2K_ClearOverflow (int hConnection,** **BYTE bytChannel);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Cnt2K_GetFilters	This function code is used to get the filter value of contiguous D/I channels in Counter mode.	
C/C++	**int Cnt2K_GetFilters (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **WORD wFilter[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wFilter:	An array that stores the filter values for the contiguous D/I channels.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Cnt2K_SetFilters	This function code is used to set the filter values of contiguous D/I channels in Counter mode.	
C/C++	**int Cnt2K_SetFilters (int BYTE BYTE WORD**	**hConnection, bytStartChannel, bytCount, wFilter[ ]);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	wFilter:	An array that stores the filter values for the contiguous D/I channels.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Cnt2K_GetFilter	This function code is used to get the filter value of a specific D/I channel in Counter mode.	
C/C++	**int Cnt2K_GetFilter (int BYTE WORD**	**hConnection, bytChannel, * wFilter);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wFilter:	A pointer to the filter value.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Cnt2K_SetFilter	This function code is used to set the filter value of a specific D/I channel in Counter mode.	
C/C++	**int Cnt2K_SetFilter (int BYTE WORD**	**hConnection, bytChannel, wFilter);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wFilter:	Stores the filter value.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Cnt2K_GetStartStatuses	This function code is used to get the start statuses of contiguous D/I channels in Counter mode.	
C/C++	**int Cnt2K_GetStartStatuses (int**	**hConnection,**
	BYTE	**bytStartChannel,**
	BYTE	**bytCount,**
	DWORD	*** dwStatus);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dwStatus:	A pointer to the start statuses of the contiguous D/I channels; each bit holds the start status of one channel. 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 the start statuses of contiguous D/I channels in Counter mode.	
C/C++	**int Cnt2K_SetStartStatuses (int**	**hConnection,**
	BYTE	**bytStartChannel,**
	BYTE	**bytCount,**
	DWORD	**dwStatus);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	dwStatus:	A pointer to the start statuses of the contiguous D/I channels; each bit holds the start status of one channel. 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 the start status of a specific D/I channel in Counter mode.	
C/C++	**int Cnt2K_GetStartStatus (int hConnection,** **BYTE bytChannel,** **BYTE * bytStatus);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	bytStatus:	A pointer to the start status of the desired channel. 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 the start status of a specific D/I channel in Counter mode.	
C/C++	**int Cnt2K_SetStartStatus (int hConnection,** **BYTE bytChannel,** **BYTE bytStatus);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	bytStatus:	A pointer to the start status of the desired channel. 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 the trigger types of contiguous D/I channels in Counter mode.	
C/C++	**int Cnt2K_GetTriggerTypes (int**	**hConnection,**
	BYTE	**bytStartChannel,**
	BYTE	**bytCount,**
	DWORD	*** dwType);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dwType:	A pointer to the triggers types; each bit holds the trigger type of one channel. 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 the trigger types of contiguous D/I channels in Counter mode.	
C/C++	**int Cnt2K_SetTriggerTypes (int**	**hConnection,**
	BYTE	**bytStartChannel,**
	BYTE	**bytCount,**
	DWORD	**dwType);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	dwType:	Stores the trigger types of the contiguous channels; each bit holds the trigger type of one channel. 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 the trigger type of a specific D/I channel in Counter mode.	
C/C++	**int Cnt2K_GetTriggerType (int hConnection, BYTE bytChannel, BYTE * bytType);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	bytType:	A pointer to the trigger type for a 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 the trigger type of a specific D/I channel in Counter mode.	
C/C++	**int Cnt2K_SetTriggerType (int hConnection, BYTE bytChannel, BYTE bytType);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The desired channel.
	bytType:	Stores the trigger type for the desired 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 the power on values of contiguous D/I channels in Counter mode.
C/C++	**int Cnt2K_GetPowerOnValues (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD * dwValue);**
Arguments	*hConnection:* The handle for an I/O connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to read. *dwValue:* A pointer to the power on values; each bit holds the power on value of one channel. 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 the power on values of contiguous D/I channels in Counter mode.
C/C++	**int Cnt2K_SetPowerOnValues (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD dwValue);**
Arguments	*hConnection:* The handle for an I/O connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to set. *dwValue:* Stored the power on values for the contiguous channels; each bit holds the power on value of one channel. 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 values when specific D/I channel in Counter mode.
C/C++	**int Cnt2K_GetPowerOnValue (int hConnection,** **BYTE bytChannel,** **BYTE * bytValue);**
Arguments	*hConnection:* The handle for an I/O connection. *bytChannel:* The desired channel. *bytValue:* A pointer to 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 the power on value of a specific D/I channel in Counter mode.
C/C++	**int Cnt2K_SetPowerOnValue (int hConnection,** **BYTE bytChannel,** **BYTE bytValue);**
Arguments	*hConnection:* The handle for an I/O connection. *bytChannel:* The desired channel. *bytValue:* Stores the power on value for the desired 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 the safe values of contiguous D/I channels in Counter mode.	
C/C++	**int Cnt2K_GetSafeValues (int BYTE BYTE DWORD**	**hConnection, bytStartChannel, bytCount, * dwValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dwValue:	A pointer to the safe values; each bit holds the safe value of one channel. 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 Fail	MXIO_OK. Refer to Return Codes.

Cnt2K_SetSafeValues	This function code is used to set the safe values of contiguous D/I channels in Counter mode.	
C/C++	**int Cnt2K_SetSafeValues (int BYTE BYTE DWORD**	**hConnection, bytStartChannel, bytCount, dwValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	dwValue:	Stored the safe values of the contiguous channels; each bit holds the safe value of one channel. 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 Fail	MXIO_OK. Refer to Return Codes.

Cnt2K_GetSafeValue	This function code is used to get the safe value of a specific D/I channel in Counter mode.	
C/C++	**int Cnt2K_GetSafeValue (int BYTE BYTE**	**hConnection, bytChannel, * bytValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	bytValue:	A pointer to the safe value of the desired channel. The values are: 0: OFF 1: ON
Return Value	Succeed Fail	MXIO_OK. Refer to Return Codes.

Cnt2K_SetSafeValue	This function code is used to set the safe value of a specific D/I channel in Counter mode.	
C/C++	**int Cnt2K_SetSafeValue (int BYTE BYTE**	**hConnection, bytChannel, bytValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	bytValue:	Stores the safe value for the desired channel. The values are: 0: OFF 1: ON
Return Value	Succeed Fail	MXIO_OK. Refer to Return Codes.

Cnt2K_GetTriggerTypeWords	This function code is used to get trigger types for contiguous channels when the D/I channel is in Count mode.
C/C++	**int Cnt2K_GetTriggerTypeWords** **(int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount** **WORD *wTput);**
Arguments	*hConnection:* The handle for an I/O connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to get. *wType:* A pointer that stores the trigger types for contiguous channels; 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 get trigger types for contiguous channels when the D/I channel is in Count mode.
C/C++	**int Cnt2K_SetTriggerTypeWords** **(int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount** **WORD *wTput);**
Arguments	*hConnection:* The handle for an I/O connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to set. *wType:* A pointer that stores the trigger types for contiguous channels; 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 contiguous channel's trigger types when D/I channel is in Count mode.
C/C++	**int Cnt2K_GetTriggerTypeWord** **(int hConnection,** **BYTE bytChannel,** **WORD *wTput);**
Arguments	*hConnection:* The handle for an I/O device connection. *byttChannel:* The specific channel to get. *wType:* A pointer that stores the trigger type for a 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 get contiguous channel's trigger types when D/I channel is in Count mode.
C/C++	**int Cnt2K_SetTriggerTypeWord** **(int hConnection,** **BYTE bytChannel,** **WORD wTput);**
Arguments	*hConnection:* The handle for an I/O device connection. *bytChannel:* The specific channel to be clear. *wType:* Stores the trigger type for a specific channel. The values are : 0: LoToHi 1:HiToLo 2: Both
Return Value	Succeed MXIO_OK. Fail Refer to Return Codes.

Cnt2K_GetSaveStatuses OnPowerFail	This function code is used to get the power-off-storage mode status for contiguous DI/DO channels.
C/C++	**int Cnt2K_GetSaveStatusesOnPowerFail** **(int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount** **WORD *dwMode);**
Arguments	*hConnection:* The handle for an I/O connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to get. *wType:* A pointer that stores the mode status for contiguous DI/DO channels; 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_SetSaveStatuses OnPowerFail	This function code is used to get the power-off-storage mode status for contiguous DI/DO channels.
C/C++	**int Cnt2K_SetSaveStatusesOnPowerFail** **(int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount** **WORD dwMode);**
Arguments	*hConnection:* The handle for an I/O connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to set. *dwStatus:* A pointer that stores the mode status for contiguous DI/DO channels; 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.

Counter Commands for ioLogik W5000

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. 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's connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to be get. *dwValue:* 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'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_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's connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to bet read. *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's 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's 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's connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to be set. *dwStatus:* 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.

W5K_Cnt_GetTriggerTypes	This function code is used to get contiguous channel's trigger types when D/I channel in "Count" mode.
C/C++	**int W5K_Cnt_GetTriggerTypes** **(int hConnection,** **BYTE bytStartChannel** **BYTE bytCount** **DWORD * dwType);**
Visual Basic	**Declare Function W5K_Cnt_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's connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to be get. *dwType:* 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_SetTriggerTypes	This function code is used to set contiguous channel's trigger types when D/I channel in "Count" mode.
C/C++	**int W5K_Cnt_SetTriggerTypes** **(int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD dwType);**
Visual Basic	**Declare Function W5K_Cnt_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's 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.

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's 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's 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's 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's 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's 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_SetTriggerType Words	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's 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_GetSaveStatuses OnPowerFail	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's 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_SetSaveStatuses OnPowerFail	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's 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.

Counter Commands for the ioLogik E1200

E1K_Cnt_Reads	This function code is used to read contiguous channel's count values when D/I channels are 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 values, 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 values when D/I channel is 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 cleared.
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 the D/I channel in is "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's connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels.
	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's overflow status when the D/I channel is 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's connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to be cleared.
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 is 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's connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels. *wFilter:* An array that stores the filter values for contiguous D/I channels.
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 the D/I channel is 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's connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to be set. *wFilter:* An array that stores the filter values for contiguous D/I channels.
Return Value	Succeed MXIO_OK. Fail Refer to Return Codes.

E1K_Cnt_GetStartStatuses	This function code is used to get contiguous channel's filter values when D/I channels are 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's 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 filter values when the D/I channel is 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's 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 the D/I channel is 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's connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels.
	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 the D/I channel is in "Count" mode.	
C/C++	**int E1K_Cnt_GetPowerOnValues** **(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's connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to be cleared.
	dwValue:	Stores contiguous channels' 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 the D/I channel is 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's connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels.
	dwValue:	A pointer that stores contiguous channels' 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 channels' safe values when the D/I channel is 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's connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to be cleared.
	dwValue:	Stores contiguous channels' 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_GetTriggerType Words	This function code is used to get contiguous channels' trigger types when the D/I channel is 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's connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to be get. *wType:* A pointer that stores contiguous channels' 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 channels' trigger types when the D/I channel is 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's connection. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to be cleared. *wType:* Stores contiguous channels' 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_GetSaveStatusesOn PowerFail	This function code is used to get contiguous channels' 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's connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels.
	dwStatus:	A pointer that stores contiguous channels' DI/DO modes; 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_SetSaveStatusesOn PowerFail	This function code is used to set contiguous channels' 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's connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to be set.
	dwStatus:	A pointer that stores the contiguous channels' DI/DO modes; 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 Fail	MXIO_OK. Refer to Return Codes.

Digital Output Commands

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);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytSlot: Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel: Specifies the starting channel.
	bytCount: The number of channels to read.
	dwValue: A pointer to the statuses of the contiguous D/O channels; each bit holds the value of one channel. 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_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);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytSlot: Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel: The desired channel.
	bytValue: A pointer to the output value of the desired channel.
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. There may not be contiguous channels if you use ioLogik E2212 DIO channels. Please make sure they are contiguous channels or "ILLEGAL_DATA_VALUE" will be returned.
C/C++	**int DO_Writes (int hConnection,** **BYTE bytSlot,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD dwValue);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytSlot: Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel: Specifies the starting channel.
	bytCount: The number of channels to be written.
	dwValue: Stores the channel statuses of the contiguous D/O channels; each bit holds the status of one channel. 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_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);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytSlot: Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel: The desired channel.
	bytValue: Stores the output status of the desired channel. 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 the output safe values of contiguous D/O channels.	
C/C++	**int DO_GetSafeValues (int hConnection,** **BYTE bytSlot,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD * dwValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dwValue:	A pointer to the safe values of the contiguous D/O channels; each bit holds the value of one channel. 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 the safe values of contiguous D/O channels.	
C/C++	**int DO_SetSafeValues (int hConnection,** **BYTE bytSlot,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD dwValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	dwValue:	A pointer to the safe values of the contiguous D/O channels; each bit holds the value of one channel. 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 hConnection, BYTE bytSlot, BYTE bytChannel, BYTE bytValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel:	The desired channel.
	bytValue:	Stores the safe value of the desired D/O channel. 1 represents ON, 0 represents OFF.
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 hConnection, BYTE bytSlot, BYTE bytChannel, BYTE bytValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel:	The desired channel.
	bytValue:	Stores the safe value of the desired channel. 1 represents ON, 0 represents OFF.
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);**	
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 ioLigik 2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to bet read.
	wValue:	A pointer that stores the safe value of contiguous D/O channels; 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_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);**	
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 ioLigik 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 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.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Digital Output Commands for ioLogik E2000, R2000

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[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wMode:	An array that stores the modes 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 modes of contiguous D/O channels.	
C/C++	**int DO2K_SetModes (int BYTE BYTE WORD**	**hConnection, bytStartChannel, bytCount, wMode[ ]);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	wMode:	An array that stores the modes 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 of a specific D/O channel.	
C/C++	**int DO2K_GetMode(int BYTE BYTE**	**hConnection, bytChannel, * wMode);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wMode:	A pointer to the mode of tthe desired 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 D/O channel.	
C/C++	**int DO2K_SetMode (int hConnection, BYTE bytChannel, WORD wMode);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wMode:	A pointer to the mode of the desired 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 values of contiguous D/O channels.	
C/C++	**int DO2K_GetPowerOnValues (int hConnection, BYTE bytStartChannel, BYTE bytCount, DWORD * dwValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dwValue:	A pointer to the power on values of the contiguous D/O channels; each bit holds the value of one channel. 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 values of contiguous D/O channels.	
C/C++	**int DO2K_SetPowerOnValues (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD dwValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	dwValue:	Stores the power on values of contiguous D/O channels; each bit holds the value of one channel. 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 D/O channel.	
C/C++	**int DO2K_GetPowerOnValue (int hConnection,** **BYTE bytChannel,** **BYTE * bytValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	bytValue:	A pointer to the power on value of the desired 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 D/O channel.	
C/C++	**int DO2K_SetPowerOnValue (int hConnection,** **BYTE bytChannel,** **BYTE bytValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	bytValue:	Stores the power on value of a specific D/O channel.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

DO2K_GetPowerOnSeq Delaytimes	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);**	
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: 0-30.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

DO2K_SetPowerOnSeq Delaytimes	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);**
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.

Digital Input/Output Commands for ioLogik E2000

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);**
Arguments	*hConnection*: The handle for an I/O 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);**
Arguments	*hConnection:* The handle for an I/O 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);**
Arguments	*hConnection:* The handle for an I/O 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);**
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.

Digital Output Commands for ioLogik W5000

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 bet read. *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 bet read. *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 bet read. *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.

Digital Output Commands for ioLogik E1200

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 contiguous D/O channels' 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. *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. *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 read. *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_GetPowerOnSeq Delaytimes	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 : Range: 0 - 300
Return Value	Succeed MXIO_OK Fail Refer to Return Codes.

Digital Input/Output Commands for ioLogik W5000

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. *dwValue:* 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.

Digital Output Commands for ioLogik 4000

DO4K_GetSafeActions	This function code is used to get the safe actions of contiguous D/O channels.	
C/C++	**int DO4K_GetSafeActions (int hConnection, BYTE bytSlot, BYTE bytStartChannel, BYTE bytCount, WORD wAction[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wAction:	An array that stores the safe actions 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 actions of contiguous D/O channels.	
C/C++	**int DO4K_SetSafeActions (int hConnection, BYTE bytSlot, BYTE bytStartChannel, BYTE bytCount, WORD wAction[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	wAction:	An array that stores the safe actions 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 D/O channel.
C/C++	**int DO4K_GetSafeAction (int hConnection,** **BYTE bytSlot,** **BYTE bytChannel,** **WORD * wAction);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytSlot: Slot number of the I/O module, from 1 to 32.
	bytChannel: The desired channel.
	wAction: A pointer to the safe action for the desired D/O channel. 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_GetSafeAction (int hConnection,** **BYTE bytSlot,** **BYTE bytChannel,** **WORD wAction);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytSlot: Slot number of the I/O module, from 1 to 32.
	bytChannel: The desired channel.
	wAction: Stores the safe action for the desired D/O channel. The values are: 0: Safe Value 1: Hold last state
Return Value	
	Succeed MXIO_OK.
	Fail Refer to Return Codes.

Digital Output Commands for ioLogik E4200

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, from 1 to 16. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to be retrieved. *dwAction:* A pointer that stores the contiguous D/O channel's safe 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, 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 safe 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, from 1 to 16. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to be retrieved. *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, 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 on the 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 BYTE BYTE BYTE DWORD**	**hConnection, bytSlot, bytStartChannel, bytCount, * dwValue);**
Visual Basic	**Declare Function E42_DO_GetFaultValues Lib "MXIO.dll" (ByVal ByVal ByVal ByVal nValue**	**hConnection As Long, bytSlot As Byte, bytStartChannel As Byte, bytCount As Byte, 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 on the ioLogik 2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to be received.
	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 on the 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.

Digital Input/Output Commands for E1200

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.

Pulse Output Commands for ioLogik E2000, R2000

Pulse2K_GetSignalWidths	This function code is used to get the Hi/Lo signal widths of contiguous pulse output channels.	
C/C++	**int Pulse2K_GetSignalWidths (int hConnection, BYTE bytStartChannel, BYTE bytCount, WORD wHiWidth[ ], WORD wLoWidth[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wHiWidth:	An array that stores the Hi signal widths of the contiguous pulse output channels; wHiWidth[0] represents the Hi signal width of the starting channel.
	wLoWidth:	An array that stores the Lo signal widths of the contiguous pulse output channels; wLoWidth[0] represents the Lo signal width of the starting channel.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Pulse2K_SetSignalWidths	This function code is used to set the Hi/Lo signal widths of contiguous pulse output channels.	
C/C++	**int Pulse2K_SetSignalWidths (int hConnection, BYTE bytStartChannel, BYTE bytCount, WORD wHiWidth[ ], WORD wLoWidth[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	wHiWidth:	An array that stores the Hi signal widths of the contiguous pulse output channels; wHiWidth[0] represents the Hi signal width of the starting channel.
	wLoWidth:	An array that stores the Lo signal widths of the contiguous pulse output channels; wLoWidth[0] represents the Lo signal width of the starting channel.
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);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wHiWidth:	A pointer to the Hi signal width of the desired channel.
	wLoWidth:	A pointer to the Lo signal width of the desired channel.
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);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wHiWidth:	A pointer to the Hi signal width of the desired channel.
	wLoWidth:	A pointer to the Lo signal width of the desired channel.
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 designed for the 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 connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to get.
	dwHiWidth:	An array that stores the contiguous pulse channel's Hi signal width, dwHiWidth[0] represents the Hi signal width of the starting channel.
	dwLoWidth:	An array that stores the contiguous plulse channel's Lo signal width , dwLoWidth[0] represents the Lo signal width of the starting channel.
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 designed for the 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 connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to get.
	dwHiWidth:	An array that stores the contiguous pulse channel's Hi signal width, dwHiWidth[0] represents the Hi signal width of the starting channel.
	dwLoWidth:	An array that stores the contiguous plulse channel's Lo signal width , dwLoWidth[0] represents the Lo signal width of the starting channel.
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 designed for the 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 connection.
	bytChannel:	The specific channel to get.
	dwHiWidth:	A pointer that stores the specific pulse channel's Hi signal width.
	dwLoWidth:	A pointer that stores the specific pulse channel's Lo signal width.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Pulse2K_GetSignalWidth32	This function code is used to set the Hi/Lo signal width (32 bits) for a specific pulse channel. The function code is designed for the 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, **iHiWidth** As Long, **iLoWidth** As Long) As Long	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The specific channel to get.
	dwHiWidth:	Store the specific pulse channel's Hi signal width.
	dwLoWidth:	Store the specific pulse channel's Lo signal width.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Pulse2K_GetOutputCounts	This function code is used to get the output count.	
C/C++	**int Pulse2K_GetOutputCounts (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD dwOutputCounts[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dwOutputCounts:	An array that stores the 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 output counts for contiguous pulse output channels.	
C/C++	**int Pulse2K_SetOutputCounts (int hConnection, BYTE bytStartChannel, BYTE bytCount, DWORD dwOutputCounts[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	dwOutputCounts:	An array that stores the output counts of the contiguous pulse output chanels; 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);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	dwOutputCount:	A pointer to the output count for the desired channel.
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);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	dwOutputCount:	A pointer to the output count for the desired channel.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Pulse2K_GetStartStatuses	This function code is used to get the start statuses of contiguous pulse channels.
C/C++	**int Pulse2K_GetStartStatuses (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD * dwStatus);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytStartChannel: Specifies the starting channel.
	bytCount: The number of channels to read.
	dwStatus: A pointer that stores the start statuses for the contiguous pulse channels; each bit holds the value of one channel. 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 start statuses of contiguous pulse channels.
C/C++	**int Pulse2K_SetStartStatuses (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD dwStatus);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytStartChannel: Specifies the starting channel.
	bytCount: The number of channels to set.
	dwStatus: Stores the start statuses of the contiguous pulse channels; each bit holds the value of one channel. 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);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	bytStatus:	A pointer to the start status of the desired channel. 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);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	bytStatus:	Stores the start status of the desired channel. 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 power on values for contiguous pulse channels.	
C/C++	**int Pulse2K_GetPowerOnValues (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD * dwValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dwValue:	A pointer to the power on values of the contiguous pulse channels; each bit holds the value of one channel. 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 power on values of contiguous pulse channels.	
C/C++	**int Pulse2K_SetPowerOnValues (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD dwValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	dwValue:	Stores the power on values for the contiguous channels; each bit holds the value of one channel. 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);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	bytValue:	A pointer to the power on value of the desired channel. 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);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	bytValue:	Stores the power on value for the desired channel. 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 safe values of contiguous pulse channels.	
C/C++	**int Pulse2K_GetSafeValues (int hConnection, BYTE bytStartChannel, BYTE bytCount, DWORD * dwValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dwValue:	A pointer to the safe values for the contiguous channels; each bit holds the value of one channel 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 safe value.	
C/C++	**int Pulse2K_SetSafeValues (int hConnection, BYTE bytStartChannel, BYTE bytCount, DWORD dwValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	dwValue:	Stores the safe value, each bit holds the value of one channel. 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);**	
Arguments	*hConnection:*	The handle for an I/O server connection.
	bytChannel:	The desired channel.
	bytValue:	A pointer to 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);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	bytValue:	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.

Pulse Output Commands for ioLogik W5000

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 bet read.
	dwHiWidth:	An array that stores the contiguous pulse channel's Hi signal width, dwHiWidth[0] represents the Hi signal width of the starting channel.
	dwLoWidth:	An array that stores the contiguous pulse channel's Lo signal width, dwLoWidth[0] represents the Lo signal width of the starting channel.
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. *dwLoWidth:* An array that stores the contiguous pulse channel's Lo signal width, dwLoWidth[0] represents the Lo signal width of the starting channel.
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 bet read. *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 output count.
C/C++	**int W5K_Pulse_SetOutputCounts** **(int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD dwOutputCounts[ ]);**
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 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 bet read. *dwStatus:* A pointer 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 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:* A pointer 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_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 bet read. *bytStatus:* A pointer 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 bet read. *dwValue:* A pointer 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:* A pointer 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.

Analog Input Commands

AI_Reads	This function code is used to read the values of contiguous analog input channels.
C/C++	**int AI_Reads (int hConnection,** **BYTE bytSlot,** **BYTE bytStartChannel,** **BYTE bytCount,** **double dValue[ ]);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytSlot: Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel: Specifies the starting channel.
	bytCount: The number of channels to read.
	dValue: An array that stores the values of the contiguous A/I channels; 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 value of a specific analog input channel.
C/C++	**int AI_Read (int hConnection,** **BYTE bytSlot,** **BYTE bytChannel,** **double * dValue);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytSlot: Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel: The desired channel.
	dValue: A pointer to the value of the desired analog input 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_ReadRaws	This function code is used to read the raw data values of contiguous analog input values.	
C/C++	**int AI_ReadRaws (int hConnection, BYTE bytSlot, BYTE bytStartChannel, BYTE bytCount, WORD wValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wValue:	An array that stores the raw data values of the contiguous A/I channels; 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 raw data value of a specific analog input channel.	
C/C++	**int AI_ReadRaw (int hConnection, BYTE bytSlot, BYTE bytChannel, WORD * wValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel:	The desired channel.
	wValue:	A pointer to raw data value of the desired analog input channel.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Analog Input Commands for ioLogik E2000, R2000

AI2K_ReadMins	This function code is used to read the minimize values of contiguous A/I channels.	
C/C++	**int AI2K_ReadMins (int hConnection, BYTE bytStartChannel, BYTE bytCount, double dValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dValue:	An array that stores the minimize values of contiguous A/I channels; 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 the minimize raw data values of contiguous A/I channels.	
C/C++	**int AI2K_ReadMinRaws (int hConnection, BYTE bytStartChannel, BYTE bytCount, WORD wValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wValue:	An array that stores the minimize raw data values of the contiguous A/I channels; 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 the minimize values of contiguous A/I channels.	
C/C++	**int AI2K_ResetMins (int BYTE BYTE**	**hConnection, bytStartChannel, bytCount,**
Arguments	*hConnection:*	The handle for an I/O 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 minimize value for a specific A/I channel.	
C/C++	**int AI2K_ReadMin (int BYTE double**	**hConnection, bytChannel, * dValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	dValue:	A pointer to the minimize value of the desired A/I 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_ReadMinRaw	This function code is used to read the minimize raw data value for a specific A/I channel.	
C/C++	**int AI2K_ReadMinRaw(int BYTE int**	**hConnection, bytChannel, * iValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	iValue:	A pointer to the minimize raw data value for the desired channel.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

AI2K_ResetMin	This function code is used to reset the minimize value for a specific A/I channel.	
C/C++	**int AI2K_ResetMin (int BYTE**	**hConnection, bytChannel,**
Arguments	*hConnection:*	The handle for an I/O server connection.
	bytChannel:	The desired channel.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

AI2K_ReadMaxs	This function code is used to read the maximize values for contiguous A/I channels.	
C/C++	**int AI2K_ReadMaxs (int BYTE BYTE double**	**hConnection, bytStartChannel, bytCount, dValue[ ]);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dValue:	An array that stores the maximize values for contiguous A/I channels; 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 the maximize raw data values for contiguous A/I channels.	
C/C++	**int AI2K_ReadMaxRaws (int hConnection, BYTE bytStartChannel, BYTE bytCount, WORD wValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wValue:	An array that stores the maximize raw data values for the contiguous A/I channels; 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 the maximize values of contiguous A/I channels.	
C/C++	**int AI2K_ResetMaxs (int hConnection, BYTE bytStartChannel, BYTE bytCount,**	
Arguments	*hConnection:*	The handle for an I/O 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 maximize value for a specific A/I channel.	
C/C++	**int AI2K_ReadMax (int hConnection, BYTE bytChannel, double * dValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The specific channels to be read.
	dValue:	A pointer to the maximize value of the desired A/I 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_ReadMaxRaw	This function code is used to read the maximize raw data value for a specific A/I channel.	
C/C++	**int AI2K_ReadMaxRaw (int hConnection,** **BYTE bytChannel,** **WORD * wValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wValue:	A pointer to the raw data value of the desired A/I channel.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

AI2K_ResetMax	This function code is used to reset the maximize value for a specific A/I channel.	
C/C++	**int AI2K_ResetMin (int hConnection,** **BYTE bytChannel,**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

AI2K_GetRanges	This function code is used to get the ranges of contiguous A/I channels.	
C/C++	**int AI2K_GetRanges (int BYTE BYTE WORD**	**hConnection, bytStartChannel, bytCount, wRange[ ]);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wRange:	An array that stores the ranges of the contiguous A/I channels; 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 Others: return Illegal Data Value
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

AI2K_SetRanges	This function code is used to set the ranges of contiguous A/I channels.	
C/C++	**int AI2K_SetRanges (int BYTE BYTE WORD**	**hConnection, bytStartChannel, bytCount, wRange[ ]);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	wRange:	An array that stores the ranges of the contiguous A/I channels; 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 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 BYTE WORD**	**hConnection, bytChannel, * wRange);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wRange:	A pointer to the range of the desired A/I channel. The values are: 00: ±150mV 01: ±500mV 02: ±5V 03: ±10V 04: 0-20mA 05: 4-20mA 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);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wRange:	Stores the range of the desired A/I channel. The values are: 00: ±150mV 01: ±500mV 02: ±5V 03: ±10V 04: 0-20mA 05: 4-20mA 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 A/I channel status for the ioLogik 2000 module.	
C/C++	**int AI2K_GetChannelStatus (inthConnection, BYTE bytChannel, WORD *wValue);**	
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 A/I channel status for ioLogik 2000 module.	
C/C++	**int AI2K_SetChannelStatus (int hConnection,** **BYTE bytChannel,** **WORD wValue);**	
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_GetChannelStatus	This function code is used to get the A/I channel status for the ioLogik 2000 module.	
C/C++	**int AI2K_GetChannelStatus (inthConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **WORD wValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O device connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to bet read.
	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 A/I channel status for the ioLogik 2000 module.	
C/C++	**int AI2K_SetChannelStatus (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **WORD wValue[ ]);**	
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.

Analog Input Commands for ioLogik E4200

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 on the 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 on the 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.

Analog Input Commands for ioLogik W5000

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_ReadMinRaws	This function code is used to read contiguous A/I channel's minimize raw data.
C/C++	**int W5K_AI_ReadMinRaws** **(int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **WORD wValue[ ]);**
Visual Basic	**Declare Function W5K_AI_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.

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 o 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 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 bet read. *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.

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: 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.

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 bet read.
	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.

Analog Input Commands for E1200

E1K_AI_Reads	This function code is used to read contiguous channel's analog input value.	
C/C++	**int E1K_AI_Reads(int BYTE BYTE double**	**hConnection, bytStartChannel, bytCount, 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_ReadMinRaws	This function code is used to read contiguous A/I channel's minimize raw data.
C/C++	**int E1K_AI_ReadMinRaws(int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **WORD wValue[ ]);**
Visual Basic	Declare Function E1K_AI_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.

E1K_AI_ResetMins	This function code is used to reset contiguous A/I channel'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 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 bet read. *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 bet read.
	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.

Analog Output Commands

AO_Reads	This function code is used to read the values of contiguous analog output channels.
C/C++	**int AO_Reads (int hConnection,** **BYTE bytSlot,** **BYTE bytStartChannel,** **BYTE bytCount,** **double dValue[ ]);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytSlot: Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel: Specifies the starting channel.
	bytCount: The number of channels to read.
	dValue: An array that stores the values of contiguous analog output channels. 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 values of contiguous analog output channels.
C/C++	**int AO_Writes (int hConnection,** **BYTE bytSlot,** **BYTE bytStartChannel,** **BYTE bytCount,** **double dValue[ ]);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytSlot: Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel: Specifies the starting channel.
	bytCount: The number of channels to write.
	dValue: An array that stores the values of contiguous channel outputs. 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_Read	This function code is used to read the value for a specific analog output channel.
C/C++	**int AO_Read (int hConnection,** **BYTE bytSlot,** **BYTE bytChannel,** **double * dValue);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytSlot: Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel: The desired channel.
	dValue: A pointer to the value of the desired 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_Write	This function code is used to write the status for a specific analog output channel.
C/C++	**int AO_Write (int hConnection,** **BYTE bytSlot,** **BYTE bytChannel,** **double dValue);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytSlot: Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel: The desired channel.
	dValue: Stores the value of the desired 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_ReadRaws	This function code is used to read the raw data values of contiguous analog output channels.	
C/C++	**int AO_ReadRaws (int hConnection,** **BYTE bytSlot,** **BYTE bytStartChannel,** **BYTE bytCount,** **WORD wValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wValue:	An array that stores the raw data values for the contiguous analog output channels. 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 raw data values for contiguous analog output channels.	
C/C++	**int AO_WriteRaws (int hConnection,** **BYTE bytSlot,** **BYTE bytStartChannel,** **BYTE bytCount,** **WORD wValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to write.
	wValue:	An array that stores raw data values for the contiguous analog output channels. 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 raw data value of a specific analog output channel.	
C/C++	**int AO_ReadRaw (int BYTE BYTE WORD**	**hConnection, bytSlot, bytChannel, * wValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel:	The desired channel.
	wValue:	A pointer to the raw data value of the desired analog output channel.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

AO_WriteRaw	This function code is used to write the raw data value of a specific analog output channel.	
C/C++	**int AO_WriteRaw (int BYTE BYTE WORD**	**hConnection, bytSlot, bytChannel, wValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel:	The specific channel to be written.
	wValue:	Stores the raw data value for the desired analog output channel.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

AO_GetSafeValues	This function code is used to get the safe values of contiguous analog output channels.	
C/C++	**int AO_GetSafeValues (int hConnection, BYTE bytSlot, BYTE bytStartChannel, BYTE bytCount, Double dValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dValue:	An array that stores the safe values for the contiguous A/O channels. 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 values for contiguous A/O channels.	
C/C++	**int AO_SetSafeValues (int hConnection, BYTE bytSlot, BYTE bytStartChannel, BYTE bytCount, Double dValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	dValue:	An array that stores the safe values for the contiguous A/O channels. 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 channel.	
C/C++	**int AO_GetSafeValue (int hConnection, BYTE bytSlot, BYTE bytChannel, double * dValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel:	The desired channel.
	dValue:	A pointer to the safe value of the desired A/O 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_SetSafeValue	This function code is used to set the safe value for a specific A/O channel.	
C/C++	**int AO_SetSafeValue (int hConnection, BYTE bytSlot, BYTE bytChannel, double dValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel:	The desired channel.
	dValue:	Stores the safe value of the desired A/O 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_GetSafeRaws	This function code is used to get the raw safe values of contiguous analog output channels.	
C/C++	**int AO_GetSafeRaws (int hConnection, BYTE bytSlot, BYTE bytStartChannel, BYTE bytCount, WORD wValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wValue:	An array that stores the raw safe values of the contiguous A/O channels. 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 safe values for contiguous A/O Channel's in raw data format.	
C/C++	**int AO_GetSafeRaws (int hConnection, BYTE bytSlot, BYTE bytStartChannel, BYTE bytCount, WORD wValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to write.
	wValue:	An array that stores safe values in raw data format for the contiguous A/O channels. 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 value for a specific A/O channel in raw data format.	
C/C++	**int AO_GetSafeRaw (int hConnection, BYTE bytSlot, BYTE bytChannel, WORD * wValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel:	The desired channel.
	wValue:	A pointer to the safe value for the desired channel in raw data format.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

AO_SetSafeRaw	This function code is used to set the safe value for a specific A/O channel in raw data format.	
C/C++	**int AO_SetSafeRaw (int hConnection, BYTE bytSlot, BYTE bytChannel, WORD wValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel:	The desired channel.
	wValue:	Stores the safe value for the desired channel in raw data format.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Analog Output Commands for ioLogik E2000, R2000

AO2K_GetRanges	This function code is used to get the ranges of contiguous A/O channels.	
C/C++	**int AO2K_GetRanges (int hConnection, BYTE bytStartChannel, BYTE bytCount, WORD wRange[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wRange:	An array that stores theranges of the contiguous A/O channels. 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 ranges of contiguous A/O channels.	
C/C++	**int AO2K_SetRanges (int hConnection, BYTE bytStartChannel, BYTE bytCount, WORD wRange[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	wRange:	An array that stores the ranges of the contiguous A/O channels. 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.

AO2K_GetRange	This function code is used to get the range of a specific A/O channel.	
C/C++	**int AO2K_GetRange (int hConnection, BYTE bytChannel, WORD * wRange);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wRange:	A pointer to the range of the desired A/O 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_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);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wRange:	Stores the specific A/O channel's range. 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.

AO2K_GetPowerOnValues	This function code is used to get the power on values of contiguous A/O channels.	
C/C++	**int AO2K_GetPowerOnValues (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **double dValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dValue:	An array that stores the power on values for the contiguous A/O channels. 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 values of contiguous A/O channels.	
C/C++	**int AO2K_SetPowerOnValues (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **double dValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	dValue:	An array that stores the power on valuse for the contiguous A/O channels. 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 BYTE double**	**hConnection, bytChannel, * dValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	dValue:	A pointer to the power on value for the desired A/O 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_SetPowerOnValue	This function code is used to set the power on value for a specific channel.	
C/C++	**int AO2K_SetPowerOnValue (int BYTE double**	**hConnection, bytChannel, dValue);**
Arguments	*hConnection:*	The handle for an I/O server connection.
	bytChannel:	The desired channel.
	dValue:	Stores the power on value for the desired A/O 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_GetPowerOnRaws	This function code is used to get the power on values of contiguous A/O channels in raw data format.	
C/C++	**int AO2K_GetPowerOnRaws (int BYTE BYTE WORD**	**hConnection, bytStartChannel, bytCount, wValue[ ]);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wValue:	An array that stores the power on values of the contiguous A/O channels in raw data format. 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 values of contiguous A/O channels in raw data format.	
C/C++	**int AO2K_SetPowerOnRaws (int BYTE BYTE WORD**	**hConnection, bytStartChannel, bytCount, wValue[ ]);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to set.
	wValue:	An array that stores the power on values of contiguous A/O channels in raw data format. 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 value of a specific analog output channel in raw data format.	
C/C++	**int AO2K_GetPowerOnRaw (int BYTE WORD**	**hConnection, bytChannel, * wValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wValue:	A pointer to the power on value for the desired channel in raw data format.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

AO2K_SetPowerOnRaw	This function code is used to set the power on value of a specific analog output channel in raw data format.	
C/C++	**int AO2K_SetPowerOnRaw (int BYTE WORD**	**hConnection, bytChannel, wValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wValue:	Stores the power on value for the desired channel in raw data format.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Analog Output Commands for ioLogik 4000

AO4K_GetSafeActions	This function code is used to get the safe actions of contiguous A/O channels.	
C/C++	**int AO4K_GetSafeActions (int hConnection,** **BYTE bytSlot,** **BYTE bytStartChannel,** **BYTE bytCount,** **WORD wAction[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wAction:	An array that stores the safe actions of the contiguous A/O channels. 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 actions of contiguous A/O channels.
C/C++	**int AO4K_SetSafeActions (int hConnection,** **BYTE bytSlot,** **BYTE bytStartChannel,** **BYTE bytCount,** **WORD wAction[ ]);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytSlot: Slot number of the I/O module, from 1 to 32.
	bytStartChannel: Specifies the starting channel.
	bytCount: The number of channels to set.
	wAction: An array that stores thesafe actions for the contiguous A/O channels. 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 analog output channel.
C/C++	**int AO4K_GetSafeAction (int hConnection,** **BYTE bytSlot,** **BYTE bytChannel,** **WORD * wAction);**
Arguments	*hConnection:* The handle for an I/O connection.
	bytSlot: Slot number of the I/O module, from 1 to 32.
	bytChannel: The desired channel.
	wAction: A pointer to the safe action of the desired A/O 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_SetSafeAction	This function code is used to set the safe action for a specific channel.
C/C++	**int AO4K_GetSafeAction (int hConnection,** **BYTE bytSlot,** **BYTE bytChannel,** **WORD wAction);**
Arguments	*hConnection:* The handle for an I/O connection. *bytSlot:* Slot number of the I/O module, from 1 to 32. *bytChannel:* The desired channel. *wAction:* Stores the safe action of the desired A/O 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_SetSafeAction	This function code is used to set the safe action for a specific channel.
C/C++	**int AO4K_GetSafeAction (int hConnection,** **BYTE bytSlot,** **BYTE bytChannel,** **WORD wAction);**
Arguments	*hConnection:* The handle for an I/O connection. *bytSlot:* Slot number of the I/O module, from 1 to 32. *bytChannel:* The desired channel. *wAction:* Stores the safe action of the desired A/O 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.

Analog Output Commands for ioLogik E4200

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 strating channel. *bytCount:* The number of channels to bet read. *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 strating 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. *bytslot:* Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive on the ioLogik 2000. *bytStartChannel:* Specifies the strating 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.

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. *bytslot:* Slot number of the I/O module. The Slot number ranges from 1 to 16. But this parameter is inactive on the ioLogik 2000. *bytStartChannel:* Specifies the strating 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.

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 on the ioLogik 2000. *bytStartChannel:* Specifies the strating channel. *bytCount:* The number of channels to be 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 on the ioLogik 2000. *bytStartChannel:* Specifies the strating 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 on the ioLogik 2000. *bytStartChannel:* Specifies the starting channel. *bytCount:* The number of channels to be written. *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 on the 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 on the ioLogik 2000.	
	bytStartChannel:	Specifies the starting channel.	
	bytCount:	The number of channels to be written.	
	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 BYTE BYTE BYTE WORD**	**hConnection, bytSlot, bytStartChannel, bytCount, 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 on the ioLogik 2000.
	bytStartChannel:	Specifies the strating channel.
	bytCount:	The number of channels to retrieve.
	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 on the ioLogik 2000. *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.

Relay Commands for ioLogik 2000

RLY2K_GetResetTime	This function code is used to get the reset time for D/O channel.	
C/C++	**int RLY2K_GetResetTime (int BYTE WORD**	**hConnection, bytChannel, wValue[ ]);**
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 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 the count value for contiguous D/O channel.	
C/C++	**int RLY2K_TotalCntRead (int BYTE DWORD**	**hConnection, bytChannel, *dwValue);**
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 the count value for contiguous D/O channel.
C/C++	**int RLY2K_TotalCntReads (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD dwValue[ ]);**
Arguments	*hConnection:* The handle for an I/O device connection. *bytChannel:* Specifies the starting channel. *dwValue:* A pointer that stores the count value of contiguous D/O channel.
Return Value	Succeed MXIO_OK. Fail Refer to Return Codes.

RLY2K_CurrentCntRead	This function code is used to get the count value for contiguous D/O channel.
C/C++	**int RLY2K_CurrentCntRead (int hConnection,** **BYTE bytChannel,** **DWORD dwValue[ ]);**
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 the count value for contiguous D/O channel.
C/C++	**int RLY2K_CurrentCntReads (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD dwValue[ ]);**
Arguments	*hConnection:* The handle for an I/O device connection. *bytStartChannel:* Specifies the starting channel. *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 for contiguous D/O channel.
C/C++	**int RLY2K_ResetCnt (int hConnection,** **BYTE bytChannel);**
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 for contiguous D/O channel.
C/C++	**int RLY2K_ResetCnts (int hConnection,** **BYTE bytStartChannel** **BYTE bytCount);**
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.

Relay Commands for ioLogik W5000

W5K_RLY_GetResetTime	This function code is used to get reset time of D/O channels.	
C/C++	**int W5K_RLY_GetResetTime (int BYTE WORD**	**hConnection, bytChannel, wValue[ ]);**
Visul 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[ ]);**
Visul 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:* Specificies 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[ ]);**
Visul 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:* Specificies 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);**
Visul 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:* Specificies the starting channel. *bytCount:* The number of channels to be reset
Return Value	Succeed MXIO_OK. Fail Refer to Return Codes.

Relay Commands for E1200

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[ ]);**
Visul 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:* Specificies 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.

RTD Commands

RTD_Reads	This function code is used to read the temperature values for contiguous channels.	
C/C++	**int RTD_Reads (int hConnection, BYTE bytSlot, BYTE bytStartChannel, BYTE bytCount, double dValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dValue:	An array that stores the temperature values of the contiguous channels. dValue[0] represents the start channel. When dValue is 0x8000, 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. Use the ioAdmin utility to check the current settings for the desired channels.
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);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel:	The desired channel.
	dValue:	A pointer to the temperature value of the desired channel. When dValue is 0x8000, 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. Use the ioAdmin utility to check the current settings for the desired channels.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

RTD_ReadRaws	This function code is used to read the temperatures for contiguous channels in raw data format.	
C/C++	**int RTD_ReadRaws (int hConnection,** **BYTE bytSlot,** **BYTE bytStartChannel,** **BYTE bytCount,** **WORD wValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O server connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wValue:	An array that stores the temperature values of the contiguous channels in raw data format. wValue [0] represents the start channel. When wValue is 0x8000, it means the sensor is not wired correctly, or the measured value is out of range. When using the RTD module for Resistance Input 1-2000Ω, 100 mΩ/1count. When using the RTD module for Resistance Input 1-327Ω, 10 mΩ/1count. When using the RTD module for Resistance Input 1-620Ω, 20 mΩ/1count. When the operating mode is temperature sensor, 0.1°C (°F)/1count, depending on the setting. Use the ioAdmin utility to check the current settings for the desired channels.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

RTD_ReadRaw	This function code is used to read the temperature value of a specific channel in raw data format.
C/C++	**int RTD_ReadRaw (int hConnection,** **BYTE bytSlot,** **BYTE bytChannel,** **WORD * wValue);**
Arguments	*hConnection:* The handle for an I/O connection. *bytSlot:* Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000. *bytChannel:* The desired channel. *wValue:* A pointer to the temperature value of the desired channel. When wValue is 0x8000, it means the sensor is not wired correctly, or the measured value is out of range. When using the RTD module for Resistance Input 1-2000Ω, 100 mΩ/1count. When using the RTD module for Resistance Input 1-327Ω, 10 mΩ/1count. When using the RTD module for Resistance Input 1-620Ω, 20 mΩ/1count. When the operating mode is temperature sensor, 0.1°C (°F)/1count, depending on the setting. Use the ioAdmin utility to check the current settings for the desired channels.
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);**
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);**
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);**
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 ytStartChannel,** **BYTE bytCount);**
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);**
Arguments	*hConnection*: The handle for an I/O device connection. *bytChannel* : The specific channel to be read. *iValue*: A pointer 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[ ]);**
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);**
Arguments	*hConnection*: The handle for an I/O device connection. *bytChannel* : The specific channel to be read. *wValue*: A pointer 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[ ]);**
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);**
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 Ω for the Ohm, °C for Celsius and °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[ ]);**
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 isΩ for the Ohm, °C for Celsius and °F 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);**
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 Ω for the Ohm, °C for Celsius and °F 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[ ]);**	
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 maximize value , dValue[0] represents the value of the starting channel. The unit is Ω for the Ohm, °C for Celsius and °F for Fahrenheit.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

RTD2K_GetStartStatus	This function code is used to get specific channel's start status.	
C/C++	**int RTD2K_GetStartStatus (int hConnection,** **BYTE bytChannel,** **BYTE * bytStatus);**	
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 start status. The values are : 0 : stop 1 : start
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

RTD2K_SetStartStatus	This function code is used to set specific channel's start status.
C/C++	**int RTD2K_SetStartStatus (int hConnection,** **BYTE bytChannel,** **BYTE bytStatus);**
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 start status. The values are : 0 : stop 1 : start
Return Value	Succeed MXIO_OK. Fail Refer to Return Codes.

RTD2K_GetStartStatuses	This function code is used to get contiguous channel's start status.
C/C++	**int RTD2K_GetStartStatuses (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD * dwStatus);**
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 RTD 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.

RTD2K_SetStartStatuses	This function code is used to set contiguous channel's start status.	
C/C++	**int RTD2K_SetStartStatuses (int**	**hConnection,**
	BYTE	**bytStartChannel,**
	BYTE	**bytCount,**
	DWORD	**dwStatus);**
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.

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);**
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=2500 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);**
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=2500 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[ ]);**	
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 RTD channel's sensor type, wSensorType[0] represents the value of the starting channel. The values for normal channel 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=2500 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[ ]);**	
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 channel 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=2500 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);**
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);**
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=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_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[ ]);**
Arguments	*hConnection*:	The handle for an I/O device connection.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to bet read.
	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 get contiguous RTD channel's engineering unit.		
C/C++	**int RTD2K_SetEngUnits (**	**int**	**hConnection,**
		BYTE	**bytStartChannel,**
		BYTE	**bytCount,**
		WORD	**wEngUnit[ ]);**
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 channel.		
C/C++	**int RTD2K_GetMathPar (**	**int**	**hConnection,**
		BYTE	**bytChannel,**
		WORD	*** wMathPar);**
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 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.	

RTD2K_SetMathPar	This function code is used to set the math parameter for a specific RTD channel.
C/C++	**int RTD2K_SetMathPar (int hConnection,** **BYTE bytChannel,** **WORD wMathPar);**
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 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.

RTD2K_GetMathPars	This function code is used to get contiguous RTD channel's math parameter.
C/C++	**int RTD2K_GetMathPars (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **WORD wMathPar[ ]);**
Arguments	*hConnection*: The handle for an I/O device connection. *bytStartChannel*: Specifies the starting channel. *bytCount*: The number of channels to bet read. *wMathPart*: An array that stores the contiguous RTD 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 channel's math parameter.
C/C++	**int RTD2K_SetMathPars (int hConnection,** **BYTE bytStartChannel,** **BYTE bytCount,** **WORD wMathPar[ ]);**
Arguments	*hConnection*: The handle for an I/O device connection. *bytStartChannel*: Specifies the starting channel. *bytCount*: The number of channels to be sets. *wMathPart*: An array that stores the contiguous RTD 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.

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 on the 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 on the 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 to 2000Ω, 100mΩ/1count. When using the E42_RTD module for Resistance Input 1 to 327Ω, 10mΩ/1count. When using the E42_RTD module for Resistance Input 1 to 620Ω, 20mΩ/1count. When the operating mode is temperature sensor, 0.1°C (°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 temperatureType 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 retrieved. 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. *wSensorTypet:* An array that stores the contiguous RTD channel's sensor type, wSensorType[0] represents the value of all channels. 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 to 2000 Ohm, 100 mohm/1 count 129=1 to 327 Ohm, 10 mohm/1 count 130=1 to 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 "0MXIO.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. *wSensorTypet:* 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 to 2000 Ohm, 100 mohm/1 count 129=1 to 327 Ohm, 10 mohm/1 count 130=1 to 620 Ohm, 20 mohm/1 count Others return Illegal Data Value
Return Value	Succeed MXIO_OK. Fail Refer to Return Codes.

Thermocouple Commands

TC_Reads	This function code is used to read the temperature values for contiguous channels.	
C/C++	**int TC_Reads (int BYTE BYTE BYTE double**	**hConnection, bytSlot, bytStartChannel, bytCount, dValue[ ]);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	dValue:	An array that stores the temperature values of the contiguous channels. dValue[0] represents start channel 0. When 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 μv. 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 of a specific channel.	
C/C++	**int TC_Read (int BYTE BYTE double**	**hConnection, bytSlot, bytChannel, * dValue);**
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel:	The desired channel.
	dValue:	Stores the temperature value of the desired channel. When dValue is 0x8000, it means the sensor is not wired correctly. When the operating mode of the TC module is voltage input, the unit is μv. Use ioAdmin to check the I/O module settings.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

TC_ReadRaws	This function code is used to read the temperature value of contiguous channels in raw data format.	
C/C++	**int TC_ReadRaws (int hConnection,** **BYTE bytSlot,** **BYTE bytStartChannel,** **BYTE bytCount,** **WORD wValue[ ]);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytStartChannel:	Specifies the starting channel.
	bytCount:	The number of channels to read.
	wValue:	An array that stores the temperature values of the contiguous channels in raw data format. wValue[0] represents start channel 0. When wValue is 0x8000, it means the sensor is not correctly wired. When the operating mode is temperature sensor, 0.1°C (°F). When the operating mode of the TC module is -78.0 - 78.0 mV, 10 uV/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 value of a specific channel in raw data format.	
C/C++	**int TC_ReadRaw (int hConnection, BYTE bytSlot, BYTE bytChannel, WORD * wValue);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytSlot:	Slot number of the I/O module, from 1 to 32. This parameter is inactive for the ioLogik E2000 and R2000.
	bytChannel:	The desired channel.
	wValue:	A pointer to the temperature value to read. When wValue is 0x8000, it means the sensor is not correctly wired. When the operating mode is temperature sensor, 0.1°C (°F). 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_Reads	This function code is used to read the temperature value of contiguous channels.	
C/C++	**int E42_TC_Reads (int BYTE BYTE BYTE double**	**hConnection, bytSlot bytStartChannel, bytCount, 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 on the 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 μv. Use ioAdmin to check the I/O module settings.
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

E42_TC_ReadRaws	This function code is used to read the temperature raw data of contiguous channels.
C/C++	**int E42_TC_ReadRaws (int hConnection,** **BYTE bytSlot,** **BYTE bytStartChannel,** **BYTE bytCount,** **DWORD wValue[ ]);**
Visual Basic	Declare Function E42_TC_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 on the 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 to 78.0 mV, 10uV/count. When the operating mode of the TC module is -32.7 to 32.7 mV, 1uV/count. When the operating mode of the TC module is -65.5 to 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 retrieved. 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_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 all the channels. 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 get 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 the channels. 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.

7
Click&Go Logic Commands

Click&Go logic commands are for ioLogik E2000 and E4200 Ethernet I/O. These commands involve the activation of Click&Go logic on an ioLogik E2000 and E4200 I/O.

Logic2K_GetStartStatus	This function code is used to verify activation of Click&Go logic on an ioLogik E2000 Ethernet I/O.	
C/C++	**int Logic2K_GetStartStatus (int hConnection, WORD * wStatus);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	wStatus:	A pointer to the I/O Click&Go Logic activation status. The values are : 0: Click&Go logic is not activated 1: Click&Go logic is activated
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

Logic2K_SetStartStatus	This function code is used to activate or deactivate Click&Go logic on an ioLogik E2000 Ethernet I/O.	
C/C++	**int Logic2K_SetStartStatus (int hConnection, WORD wStatus);**	
Arguments	*hConnection:*	The handle for an I/O connection.
	bytChannel:	The desired channel.
	wStatus:	Stores the I/O Click&Go logic activation status. The values are: 0: Click&Go logic is not activated 1: Click&Go logic is activated
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

E42_Logic_GetStartStatus	This function code is used to get the Click&Go Logic start status of the ioLogik 4200 network adaptors.
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:* A 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 adaptors.
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:* A 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_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:* A 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_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.

8
Active I/O Message Commands

Active I/O message commands are for ioLogik E2000 Ethernet I/O only. These commands manage active I/O messages that are received from an ioLogik E2000.

Message2K_Start	This function code is used to start receiving active I/O messages from an ioLogik E2000 Ethernet I/O.	
C/C++	**int Message2K_Start (int WORD pfnCALLBACK**	**iProtocol, wPort, iProcAddress);**
Callback Function	**Void FunctionName (BYTE WORD**	**bytData[], wSize);**
Arguments	*iProtocol:*	Transmission protocol. 1: TCP 2: UDP
	wPort:	TCP or UDP port number.
	iProcAddress:	Callback function, which is called after receiving an active I/O message from an ioLogik E2000 Ethernet I/O.
	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 receiving active I/O messages from an ioLogik E2000 Ethernet I/O.	
C/C++	**int Message2K_Stop(int iProtocol);**	
Arguments	*iProtocol:*	Transmission protocol. 1: TCP 2: UDP
Return Value	Succeed	MXIO_OK.
	Fail	Refer to Return Codes.

E42_Message_Start	This function code is used to start receiving active messages for the ioLogik 4200 network adaptors.
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:** It is not recommended to use Message Command by VB. Please refer the example file for more details. 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 receiving an active I/O message from an 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 receiving active messages for the ioLogik 4200 network adaptors.
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.

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 receiving an active I/O message from an 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.

9
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 response timeout.
EIO_INIT_SOCKETS_FAIL	2002	An error occurred when the Windows system couldn't complete SOCKET INIT.
EIO_CREATING_SOCKET_ERROR	2003	An 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.
SIO_OPEN_FAIL	3001	Open COM port failure.

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	Baudrate is not supported.
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.

10
Product Model and ID Reference Table

The MXIO DLL library is designed for use by the ioLogik line of remote I/O, including the ioLogik 4000, E2000, R2000, E4200, W5000, and E1200 series. A list of supported products is provided below. To support new I/O modules, you must upgrade to this version of the MXIO library.

ioLogik 1200

Module ID	Model Name	Remote I/O
0x1210	E1210	16DI Active Remote I/O Server
0x1211	E1211	16DO Active Remote I/O Server
0x1212	E1212	8DI, 8DI/DO Active Remote I/O Server
0x1214	E1214	6DI, 6Relay Outputs Active Remote I/O Server
0x1240	E1240	8AI Active Remote I/O Server

ioLogik 4000

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	Active Ethernet Network Adaptor
		Digital Input
0x1400	M-1400	4 DI, sink, 24 VDC, RTB
0X1401	M-1401	4 DI, source, 24 VDC, RTB
0x1410	M-1410	4 DI, sink, 48 VDC, RTB
0x1411	M-1411	4 DI, source, 48 VDC, RTB
0x1800	M-1800	8 DI, sink, 24 VDC, RTB
0x1801	M-1801	8 DI, source, 24 VDC, RTB
0x1600	M-1600	16 DI, sink, 24 VDC, RTB
0x1601	M-1601	16 DI, source, 24 VDC, 20 pin
0x1450	M-1450	4 DI, 110 VAC, RTB
0x1451	M-1451	4 DI, 220 VAC, RTB

		Digital Output
0x2400	M-2400	4 DO, sink, MOSFET, 24 VDC, 0.5A, RTB
0x2401	M-2401	4 DO, source, MOSFET, 24 VDC, 0.5A, RTB
0x2800	M-2800	8 DO, sink, MOSFET, 24 VDC, 0.5A, RTB
0x2801	M-2801	8 DO, source, MOSFET, 24 VDC, 0.5A, RTB
0x2600	M-2600	16 DO, sink, MOSFET, 24 VDC, 0.3A, 20 pin
0x2601	M-2601	16 DO, source, MOSFET, 24 VDC, 0.3A, 20 pin
0x2402	M-2402	4 DO, sink, MOSFET, diag., 24 VDC, 0.5A, RTB
0x2403	M-2403	4 DO, source, MOSFET, diag., 24 VDC, 0.5A, RTB
0x2404	M-2404	4 DO, sink, MOSFET, diag., 24 VDC, 2.0A, RTB
0x2405	M-2405	4 DO, source, MOSFET, diag., 24 VDC, 2.0A, RTB
0x2250	M-2250	2 DO, relay, 230 VAC, 24 VDC, 2.0A, RTB
0x2450	M-2450	4DO, Relay, 230VAC, 24VDC, 2.0A, RTB
0x2254	M-2254	2 DO, Triac, 12 to 125 AC, 0.5A, RTB

		Analog Input
0x3400	M-3400	4 AI, current, 0 to 20 mA, 12 bit, RTB
0x3401	M-3401	4 AI, current, 0 to 20 mA, 14 bit, RTB
0x3402	M-3402	4 AI, current, 4 to 20 mA, 12 bit, RTB
0x3403	M-3403	4 AI, current, 4 to 20 mA, 14 bit, RTB
0x3410	M-3410	4 AI, voltage, 0 to 10V, 12 bit, RTB
0x3411	M-3411	4 AI, voltage, 0 to 10V, 14 bit, RTB
0x3412	M-3412	4 AI, voltage, -10 to 10V, 12 bit, RTB
0x3413	M-3413	4 AI, voltage, -10 to 10V, 14 bit, RTB
0x3414	M-3414	4 AI, voltage, 0 to 5V, single-ended,12 bit, RTB
0x3415	M-3415	4 AI, voltage, 0 to 5V, single-ended,14 bit, RTB
0x3802	M-3802	8AI, Current, 4-20mA, 12bit, RTB
0x3810	M-3810	8AI, Voltage, 0-10V, 12bit, RTB
0x6200	M-6200	2 AI, RTD: PT100, JPT100 300 Ohm, RTB
0x6201	M-6201	2 AI, thermocouple: 30 mV(1 uV/bit), RTB
		Analog Output
0x4201	M-4201	2 AO, 0 to 20 mA, 12 bit, RTB
0x4202	M-4202	2 AO, 4 to 20 mA, 12 bit, RTB
0x4210	M-4210	2 AO, voltage, 0 to 10V, 12 bit, RTB
0x4211	M-4211	2 AO, voltage, -10 to 10V, 12 bit, RTB
0x4212	M-4212	2 AO, voltage, 0 to 5V, 12 bit, RTB
0x4402	M-4402	4AO, 4-20mA, 12bit, RTB
0x4410	M-4410	4AO, Voltage, 0-10V, 12bit, RTB

ioLogik E2000 and R2000

Module ID	Model Name	Remote I/O
0x2110	R2210	Remote I/O with 12DI, 8DO
0x2140	R2140	Remote I/O with 8AI, 2AO
0x2210	E2210	Active Ethernet I/O with 12DI, 8DO
0x2212	E2212	Active Ethernet I/O with 8DI, 8DO, 4DIO
0x2214	E2214	Active Ethernet I/O with 6DI, 6Relay
0x2240	E2240	Active Ethernet I/O with 8AI, 2AO
0x2242	E2242	Active Ethernet I/O with 4AI, 12DIO
0x2260	E2260	Active Ethernet I/O with 6RTD, 4DO
0x2262	E2262	Active Ethernet I/O with 8TC, 4DO

ioLogik W5000

Module ID	Model Name	Remote I/O
0x5340	W5340	8DIO, 2RLY, 4AI Active Remote I/O Server