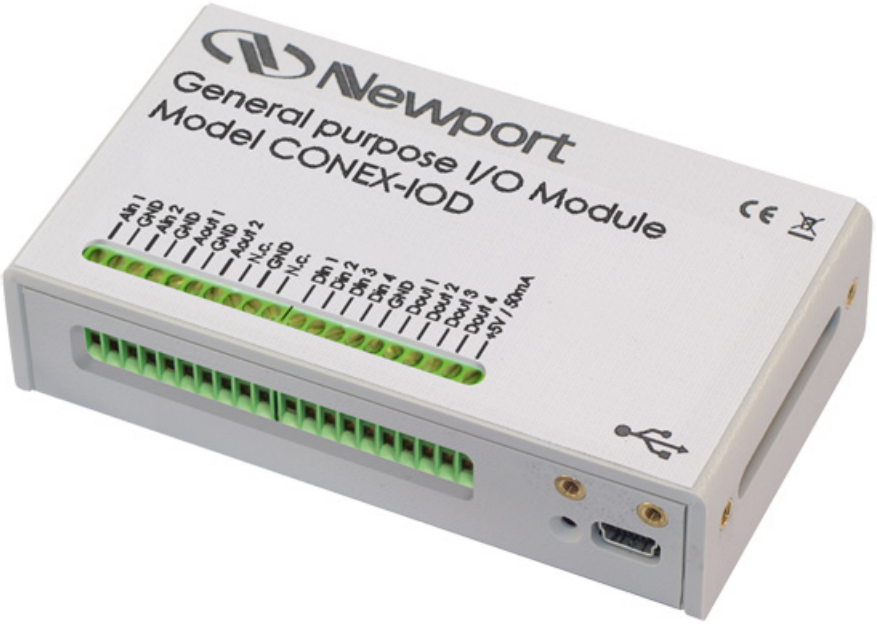
mks

CONEX-IOD

Analog/Digital I/O Module



Newport®

Command Interface Manual

V1.0.x

Table of Contents

Analog/Digital I/O Module CONEX-IOD

1.0 Introduction

1.1 Purpose

The purpose of this document is to provide the methodSyntax of each command to communicate with the CONEX-IOD device.

1.2 Overview

The Command Interface is the wrapper class that maintains a list of CONEX-IOD instruments. It exposes methods to communicate with any CONEX-IOD device.

NOTE

Each function name is defined with the command code "AA".

For each command function, refer to the CONEX-IOD programmer's manual.

2.0 Command Interface

2.1 Constructor

ConexIOD()

The constructor is used to create an instance of the CONEX-IOD device.

2.2 Functions

2.2.1 General

- **CloseInstrument**

Syntax

int CloseInstrument()

return: 0 = successful or -1 = failure

Description

This function allows closing communication with the selected device. If the closing failed, the returned code is -1.

- **GetDevices**

Syntax

string[] GetDevices()

return: list of connected devices available to communicate

Description

This function returns the list of connected devices available to communicate.

- **OpenInstrument**

Syntax

int OpenInstrument(string strDeviceKey)

string strDeviceKey: device key

return: 0 = successful or -1 = failure

Description

This function allows opening communication with the selected device. If the opening failed, the returned code is -1.

- **WriteToInstrument**

Syntax

int WriteToInstrument(string command, ref string response, int stage)

command: Instrument command

response: Response of the command

stage: Instrument Stage

return:

Description

This Overridden function Queries or writes the command given by the user to the instrument.

2.2.2 Commands

- **CA_Get**

Syntax

int CA_Get(int controllerAddress, out double AnalogInput1, out string errstring)

controllerAddress: controllerAddress

AnalogInput1: AnalogInput1

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous CA Get command which is used to get value of analog output 1.

- **CA_Set**

Syntax

int CA_Set(int controllerAddress, double AnalogInput1, out string errstring)

controllerAddress: controllerAddress

AnalogInput1: AnalogInput1

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous CA Set command which is used to Set value of analog output 1.

- **CB_Get**

Syntax

int CB_Get(int controllerAddress, out double AnalogInput2, out string errstring)

controllerAddress: controllerAddress

AnalogInput2: AnalogInput2

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous CB Get command which is used to get value of analog output 2.

♦ CB_Set

Syntax

int CB_Set(int controllerAddress, double AnalogInput2, out string errstring)

controllerAddress: controllerAddress

AnalogInput2: AnalogInput2

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous CB Set command which is used to Set value of analog output 2.

♦ CI_Get

Syntax

int CI_Get(int controllerAddress, out int AnalogInputsMode, out string errstring)

controllerAddress: controllerAddress

AnalogInputsMode: AnalogInputsMode

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous CI Get command which is used to get analog inputs mode.

♦ CI_Set

Syntax

int CI_Set(int controllerAddress, int AnalogInputsMode, out string errstring)

controllerAddress: controllerAddress

AnalogInputsMode: AnalogInputsMode

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous CI Set command which is used to Set analog inputs mode.

♦ CO_Get

Syntax

int CO_Get(int controllerAddress, out int AnalogOutputsMode, out string errstring)

controllerAddress: controllerAddress

AnalogOutputsMode: AnalogOutputsMode

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous CO Get command which is used to get analog outputs mode.

♦ CO_Set

Syntax

int CO_Set(int controllerAddress, int AnalogOutputsMode, out string errstring)

controllerAddress: controllerAddress

AnalogOutputsMode: AnalogOutputsMode

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous CO Set command which is used to Set analog outputs mode.

♦ GA_Get

Syntax

int GA_Get(int controllerAddress, out double GainDAC1, out string errstring)

controllerAddress: controllerAddress

GainDAC1: GainDAC1

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous GA Get command which is used to get gain on DAC output 1.

♦ GA_Set

Syntax

int GA_Set(int controllerAddress, double GainDAC1, out string errstring)

controllerAddress: controllerAddress

GainDAC1: GainDAC1

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous GA Set command which is used to Set gain on DAC output 1.

♦ GB_Get

Syntax

int GB_Get(int controllerAddress, out double GainDAC2, out string errstring)

controllerAddress: controllerAddress

GainDAC2: GainDAC2

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous GB Get command which is used to get gain on DAC output 2.

♦ GB_Set

Syntax

int GB_Set(int controllerAddress, double GainDAC2, out string errstring)

controllerAddress: controllerAddress

GainDAC2: GainDAC2

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous GB Set command which is used to Set gain on DAC output 2.

♦ ID_Get

Syntax

int ID_Get(int controllerAddress, out string Identifier, out string errstring)

controllerAddress: controllerAddress

Identifier: Identifier

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous ID Get command which is used to get controller identifier.

♦ ID_Set

Syntax

int ID_Set(int controllerAddress, string Identifier, out string errstring)

controllerAddress: controllerAddress

Identifier: Identifier

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous ID Set command which is used to Set controller identifier.

♦ IX_Get

Syntax

int IX_Get(int controllerAddress, out double OffsetADC1, out string errstring)

controllerAddress: controllerAddress

OffsetADC1: OffsetADC1

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous IX Get command which is used to get offset on ADC input 1.

♦ IX_Set

Syntax

int IX_Set(int controllerAddress, double OffsetADC1, out string errstring)

controllerAddress: controllerAddress

OffsetADC1: OffsetADC1

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous IX Set command which is used to set offset on ADC input 1.

♦ IY_Get

Syntax

int IY_Get(int controllerAddress, out double OffsetADC2, out string errstring)

controllerAddress: controllerAddress

OffsetADC2: OffsetADC2

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous IY Get command which is used to get offset on ADC input 2.

♦ IY_Set

Syntax

int IY_Set(int controllerAddress, double OffsetADC2, out string errstring)

controllerAddress: controllerAddress

OffsetADC2: OffsetADC2

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous IY Set command which is used to set offset on ADC input 2.

♦ LF_Get

Syntax

int LF_Get(int controllerAddress, out double Frequency, out string errstring)

controllerAddress: controllerAddress

Frequency: Frequency

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous LF Get command which is used to get low pass filter frequency.

♦ LF_Set

Syntax

int LF_Set(int controllerAddress, double Frequency, out string errstring)

controllerAddress: controllerAddress

Frequency: Frequency

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous LF Set command which is used to Set low pass filter frequency.

♦ OA_Get

Syntax

int OA_Get(int controllerAddress, out double OffsetDAC1, out string errstring)

controllerAddress: controllerAddress

OffsetDAC1: OffsetDAC1

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous OA Get command which is used to get offset on DAC output 1.

♦ OA_Set

Syntax

int OA_Set(int controllerAddress, double OffsetDAC1, out string errstring)

controllerAddress: controllerAddress

OffsetDAC1: OffsetDAC1

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous OA Set command which is used to set offset on DAC output 1.

♦ OB_Get

Syntax

int OB_Get(int controllerAddress, out double OffsetDAC2, out string errstring)

controllerAddress: controllerAddress

OffsetDAC2: OffsetDAC2

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous OB Get command which is used to get offset on DAC output 2.

♦ OB_Set

Syntax

int OB_Set(int controllerAddress, double OffsetDAC2, out string errstring)

controllerAddress: controllerAddress

OffsetDAC2: OffsetDAC2

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous OB Set command which is used to set offset on DAC output 2.

♦ PX_Get

Syntax

int PX_Get(int controllerAddress, out double GainADC1, out string errstring)

controllerAddress: controllerAddress

GainADC1: GainADC1

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous PX Get command which is used to get gain on ADC input 1.

♦ PX_Set

Syntax

int PX_Set(int controllerAddress, double GainADC1, out string errstring)

controllerAddress: controllerAddress

GainADC1: GainADC1

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous PX Set command which is used to Set gain on ADC input 1.

♦ PY_Get

Syntax

int PY_Get(int controllerAddress, out double GainADC2, out string errstring)

controllerAddress: controllerAddress

GainADC2: GainADC2

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous PY Get command which is used to get gain on ADC input 2.

♦ PY_Set

Syntax

int PY_Set(int controllerAddress, double GainADC2, out string errstring)

controllerAddress: controllerAddress

GainADC2: GainADC2

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous PY Set command which is used to Set gain on ADC input 2.

♦ PW_Get

Syntax

int PW_Get(int controllerAddress, out int State, out string errstring)

controllerAddress: controllerAddress

State: State

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous PW Get command which is used to Enter/Leave CONFIGURATION state.

♦ PW_Set

Syntax

int PW_Set(int controllerAddress, int State, out string errstring)

controllerAddress: controllerAddress

State: State

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous PW Set command which is used to Enter/Leave CONFIGURATION state.

NOTE

The PW command is limited to 100 writes. Unit failure due to excessive use of the PW command is not covered by warranty.

The PW command is used to change the configuration parameters that are stored in memory, and not parameters that are needed to be changed on the fly.

♦ RA

Syntax

int RA(int controllerAddress, out double RawAnalogInput1, out double RawAnalogInput2, out string errstring)

controllerAddress: controllerAddress

RawAnalogInput1: RawAnalogInput1

RawAnalogInput2: RawAnalogInput2

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous RA Get command which is used to get raw analog input values.

♦ RB

Syntax

int RB(int controllerAddress, out int DigitalInputs, out string errstring)

controllerAddress: controllerAddress

DigitalInputs: DigitalInputs

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous RB Get command which is used to get digital inputs.

♦ RC

Syntax

int RC(int controllerAddress, out double CorrectedAnalogInput1, out double CorrectedAnalogInput2, out string errstring)

controllerAddress: controllerAddress

CorrectedAnalogInput1: CorrectedAnalogInput1

CorrectedAnalogInput2: CorrectedAnalogInput2

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous RC Get command which is used to Get corrected analog input values.

♦ RS

Syntax

int RS(int controllerAddress, out string errstring)

controllerAddress: controllerAddress

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous RS Set command which is used to Reset controller.

♦ RS485

Syntax

int RS485(int controllerAddress, out string errstring)

controllerAddress: controllerAddress

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous RS485 Set command which is used to Reset controller's address to 1.

♦ SA_Get

Syntax

int SA_Get(int controllerAddress, out int Adress, out string errstring)

controllerAddress: controllerAddress

Adress: Adress

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous SA Get command which is used to get controller's RS-485 address.

♦ SA_Set

Syntax

int SA_Set(int controllerAddress, int Adress, out string errstring)

controllerAddress: controllerAddress

Adress: Adress

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous SA Set command which is used to Set controller's RS-485 address.

♦ SB_Get

Syntax

int SB_Get(int controllerAddress, out int DigitalOutputs, out string errstring)

controllerAddress: controllerAddress

DigitalOutputs: DigitalOutputs

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous SB Get command which is used to get digital outputs.

♦ SB_Set

Syntax

int SB_Set(int controllerAddress, int DigitalOutputs, out string errstring)

controllerAddress: controllerAddress

DigitalOutputs: DigitalOutputs

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous SB Set command which is used to Set digital outputs.

♦ TB_Get

Syntax

int TB_Get(int controllerAddress, string inErrorCode, string outErrorCode, out string errstring)

controllerAddress: controllerAddress

inErrorCode: Input error code (optional)

outErrorCode: Output errorDescription

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous TB Get command which is used to Get command error string.

♦ TE

Syntax

int TE(int controllerAddress, out string LastCommandError, out string errstring)

controllerAddress: controllerAddress

LastCommandError: LastCommandError

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous TE Get command which is used to Get last command error.

♦ TS

Syntax

int TS(int controllerAddress, out string ErrorCode, out string StatusCode, out string errstring)

controllerAddress: controllerAddress

ErrorCode: ErrorCode

StatusCode: StatusCode

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous TS Get command which is used to Get positioner error and controller state.

♦ VE

Syntax

int VE(int controllerAddress, out string Version, out string errstring)

controllerAddress: controllerAddress

Version: Version

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous VE Get command which is used to Get controller revision information.

♦ ZT

Syntax

int ZT(int controllerAddress, out List<string> Parameters, out string errstring)

controllerAddress: controllerAddress

Parameters: Parameters

errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchrounous ZT Get command which is used to get current modes parameters.

3.0 Python Example

```
#============================================================
#Initialization Start
#The script within Initialization Start and Initialization End is needed for properly
#initializing Command Interface for Conex-IOD instrument.
#The user should copy this code as is and specify correct paths here.
import sys

#Command Interface DLL can be found here.
print "Adding location of Newport.CONEXIOD.CommandInterface.dll to sys.path"
sys.path.append(r'C:\Program Files (x86)\Newport\MotionControl\CONEX-
IOD\Bin')

# The CLR module provide functions for interacting with the underlying
# .NET runtime
import clr
# Add reference to assembly and import names from namespace
clr.AddReferenceToFile("Newport.CONEXIOD.CommandInterface.dll")
from CommandInterface import *

import System
#============================================================
# Instrument Initialization
# The key should have double slashes since
# (one of them is escape character)
instrument="CONEX-IOD (A6T7NSPR)"
print 'Instrument Key=>', instrument

# create a device instance
IOD = ConexIOD()

componentID = IOD. OpenInstrument(instrument);
print 'componentID=>', componentID

# Get analog output #1 value
result, analogOutput1, errString = IOD.CA_Get(1)
if result == 0 :
  print 'Analog output #1 =>', analogOutput1
else:
  print 'Error=>',errString
```

```
# Get analog output #2 value
result, analogOutput2, errString = IOD.CB_Get(1)
if result == 0 :
  print 'Analog output #2 =>', analogOutput2
else:
  print 'Error=>',errString

# Get controller revision information
result, response, errString = IOD.VE(1)
if result == 0 :
  print 'controller revision=>', response
else:
  print 'Error=>',errString

# Get last command error
result, response, errString = IOD.TE(1)
if result == 0 :
  print 'Last command error =>', response
else:
  print 'Error=>',errString

# Unregister device
IOD. CloseInstrument();
```

Service Form

Your Local Representative

Tel.: _____

Fax: _____

Name: _____

Company: _____

Address: _____

Country: _____

P.O. Number: _____

Item(s) Being Returned: _____

Model#: _____

Return authorization #: _____
(Please obtain prior to return of item)

Date: _____

Phone Number: _____

Fax Number: _____

Serial #: _____

Description: _____

Reasons of return of goods (please list any specific problems): _____

Newport®

Visit Newport Online at:
www.newport.com

North America & Asia
Newport Corporation
1791 Deere Ave.
Irvine, CA 92606, USA

Sales
Tel.: (800) 222-6440
e-mail: sales@newport.com

Technical Support
Tel.: (800) 222-6440
e-mail: tech@newport.com

Service, RMAs & Returns
Tel.: (800) 222-6440
e-mail: service@newport.com

Europe
MICRO-CONTROLE Spectra-Physics S.A.S
9, rue du Bois Sauvage
91055 Évry CEDEX
France

Sales
Tel.: +33 (0)1.60.91.68.68
e-mail: france@newport.com

Technical Support
e-mail: tech_europe@newport.com

Service & Returns
Tel.: +33 (0)2.38.40.51.55

mks

Newport® Ophir® Spectra-Physics®