©2018 by Newport Corporation, Irvine, CA. All rights reserved.

Original instructions.

No part of this document may be reproduced or copied without the prior written approval of Newport Corporation. This document is provided for information only, and product specifications are subject to change without notice. Any change will be reflected in future publishings.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Purpose</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.3</td>
<td>Location</td>
<td>2</td>
</tr>
<tr>
<td>1.4</td>
<td>Possible uses of Newport.CONEXLDS.CommandInterface.dll</td>
<td>2</td>
</tr>
<tr>
<td>2.0</td>
<td>Command Interface</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Constructor</td>
<td>3</td>
</tr>
<tr>
<td>2.2</td>
<td>Functions</td>
<td>3</td>
</tr>
<tr>
<td>2.2.1</td>
<td>General</td>
<td>3</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Commands</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>CD</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>GP</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>GX_Get</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>GX_Set</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>GY_Get</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>GY_Set</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>ID_Get</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>ID_Set</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>LB_Get</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>LB_Set</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>LF_Get</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>LF_Set</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>PW_Get</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>PW_Set</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>PX_Get</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>PX_Set</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>PY_Get</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>PY_Set</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>RG_Get</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>RG_Set</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>RS</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>RS485</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>SA_Get</td>
<td>12</td>
</tr>
</tbody>
</table>
3.0  Python example ........................................................................................... 17

Service Form ...................................................................................................... 21
1.0 Introduction

1.1 Purpose
The purpose of this document is to provide the method syntex of each command to communicate with the CONEX-LDS autocollimator exposed in assembly Newport.CONEXLDS.CommandInterface.dll. This .Net assembly is designed and developed by Newport. This DLL is used by CONEX-LDS applet to communicate with the CONEX-LDS autocollimator.

1.2 Overview
Command Interface DLL is a generic name that refers to a DLL used to communicate with an instrument. Typically this DLL exposes all the commands the instrument supports. These commands are exposed as function calls of the Command Interface DLL.

Newport.CONEXLDS.CommandInterface.dll is the assembly used for communicating with the CONEX-LDS autocollimator. This assembly is installed with the CONEX-LDS applet.

NOTES
Each function name is defined with the command code “AA”.

For more information on each command function, refer to the CONEX-LDS programmer’s manual.
1.3 Location

Newport.CONEXLDS.CommandInterface.dll is located with the following path: \Newport\MotionControl\CONEX-LDS\Bin\Newport.CONEXLDS.CommandInterface.dll

1.4 Possible uses of Newport.CONEXLDS.CommandInterface.dll

Newport.CONEXLDS.CommandInterface.dll is used by CONEX-LDS applet for communicating with the CONEX-LDS autocollimator. The same DLL can be used as a reusable software component for creating Python scripts or LabVIEW VIs.
2.0 Command Interface

2.1 Constructor
CONEXLDS()

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

2.2 Functions

2.2.1 General

• OpenInstrument

  Syntax
  int OpenInstrument(string strDeviceKey)
  
  string strDeviceKey: the device key is a serial COM port
  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.

• 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 strings that contains the accessible COM ports

  Description
  This function returns the list of connected devices available to communicate.
• **WriteToInstrument**

  **Syntax**
  
  ```c
  int WriteToInstrument(string command, ref string resp, int stage)
  
  command: Instrument command
  resp: Response of the command
  stage: Instrument Stage
  
  return: function error
  ```

  **Description**
  
  This Overridden function Queries or writes the command issued by the user to the instrument.

2.2.2 Commands

**CD**

  **Syntax**
  
  ```c
  int CD(int controllerAddress, out string FactoryCalibrationInformation, out string errstring)
  
  controllerAddress: controllerAddress
  FactoryCalibrationInformation: FactoryCalibrationInformation
  errString: The failure reason
  
  return: 0 in success and -1 on failure
  ```

  **Description**
  
  This function is used to process synchrounous CD Get command which is used to Get factory calibration information. Refer to the Controller's User's Manual for further information.
**GP**

*Syntax*

```c
int GP(int controllerAddress, out double PositionX, out double PositionY, out double LaserPower, out string errstring)
```

- `controllerAddress`: controllerAddress
- `PositionX`: PositionX
- `PositionY`: PositionY
- `LaserPower`: LaserPower
- `errString`: The failure reason

*Description*

This function is used to process synchronous GP Get command, which is used to get positions and power. Refer to the CONEX-LDS User’s Manual for more detailed information on the command..

**GX_Get**

*Syntax*

```c
int GX_Get(int controllerAddress, out double GainX, out string errstring)
```

- `controllerAddress`: controllerAddress
- `GainX`: GainX
- `errString`: The failure reason

*Description*

This function is used to process synchronous GX Get command which is used to get gain for analog image of X channel. Refer to the CONEX-LDS User’s Manual for more information on this command.

**GX_Set**

*Syntax*

```c
int GX_Set(int controllerAddress, double GainX, out string errstring)
```

- `controllerAddress`: controllerAddress
- `GainX`: GainX
- `errString`: The failure reason

*Description*

This function is used to process synchronous GX Set command which is used to set gain for analog image of X channel. Refer to the CONEX-LDS User’s Manual for more information on this command.
GY_Get

Syntax
int GY_Get(int controllerAddress, out double GainY, out string errstring)
  controllerAddress: controllerAddress
  GainY: GainY
  errString: The failure reason
return: 0 in success and -1 on failure

Description
This function is used to process synchronous GY Get command which is used to Get gain for analog image of Y channel. Refer to the CONEX-LDS User’s Manual for more information on this command

GY_Set

Syntax
int GY_Set(int controllerAddress, double GainY, out string errstring)
  controllerAddress: controllerAddress
  GainY: GainY
  errString: The failure reason
return: 0 in success and -1 on failure

Description
This function is used to process synchronous GY Set command which is used to Set gain for analog image of Y channel. Refer to the CONEX-LDS User’s Manual for more information on this command

ID_Get

Syntax
int ID_Get(int controllerAddress, out string ControllerIdentifier, out string errstring)
  controllerAddress: controllerAddress
  ControllerIdentifier: ControllerIdentifier
  errString: The failure reason
return: 0 in success and -1 on failure

Description
This function is used to process synchronous ID Get command which is used to Get controller identifier. Refer to the CONEX-LDS User’s Manual for more information on this command
ID_Set

Syntax
int ID_Set(int controllerAddress, string ControllerIdentifier, out string errstring)
controllerAddress: controllerAddress
ControllerIdentifier: ControllerIdentifier
errString: The failure reason
return: 0 in success and -1 on failure

Description
This function is used to process synchronous ID Set command which is used to Set controller identifier. Refer to the CONEX-LDS User’s Manual for more information on this command

LB_Get

Syntax
int LB_Get(int controllerAddress, out int LaserState, out string errstring)
controllerAddress: controllerAddress
LaserState: LaserState
errString: The failure reason
return: 0 in success and -1 on failure

Description
This function is used to process synchronous LB Get command which is used to Power ON/ Power OFF the laser. Refer to the CONEX-LDS User’s Manual for more information on this command

LB_Set

Syntax
int LB_Set(int controllerAddress, int LaserState, out string errstring)
controllerAddress: controllerAddress
LaserState: LaserState
errString: The failure reason
return: 0 in success and -1 on failure

Description
This function is used to process synchronous LB Set command which is used to Power ON/ Power OFF the laser. Refer to the CONEX-LDS User’s Manual for more information on this command
**LF_Get**

**Syntax**

```c
int LF_Get(int controllerAddress, out float 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 synchronous LF Get command which is used to Get the low pass filter frequency. Refer to the CONEX-LDS User’s Manual for more information on this command.

**LF_Set**

**Syntax**

```c
int LF_Set(int controllerAddress, float 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 synchronous LF Set command which is used to Set the low pass filter frequency. Refer to the CONEX-LDS User’s Manual for more information on this command.

**PW_Get**

**Syntax**

```c
int PW_Get(int controllerAddress, out int ConfigurationState, out string errstring)
controllerAddress: controllerAddress
ConfigurationState: ConfigurationState
errString: The failure reason
return: 0 in success and -1 on failure
```

**Description**

This function is used to process synchronous PW Get command which is used to Enter/Leave the CONFIGURATION state. Refer to the CONEX-LDS User’s Manual for more information on this command.
**PW_Set**

Syntax

```c
int PW_Set(int controllerAddress, int ConfigurationState, out string errstring)
```

- controllerAddress: controllerAddress
- ConfigurationState: ConfigurationState
- errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchronous PW Set command which is used to
Enter/Leave the CONFIGURATION state. Refer to the CONEX-LDS User’s Manual
for more information on this command.

**PX_Get**

Syntax

```c
int PX_Get(int controllerAddress, out double CalibrationCoefficientX, out string errstring)
```

- controllerAddress: controllerAddress
- CalibrationCoefficientX: CalibrationCoefficientX
- errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchronous PX Get command which is used to Get
the calibration value for X channel. Refer to the CONEX-LDS User’s Manual for
more information on this command.

**PX_Set**

Syntax

```c
int PX_Set(int controllerAddress, double CalibrationCoefficientX, out string errstring)
```

- controllerAddress: controllerAddress
- CalibrationCoefficientX: CalibrationCoefficientX
- errString: The failure reason

return: 0 in success and -1 on failure

Description

This function is used to process synchronous PX Set command which is used to Set
the calibration value for X channel. Refer to the CONEX-LDS User’s Manual for
more information on this command.
**PY_Get**

**Syntax**

```c
int PY_Get(int controllerAddress, out double CalibrationCoefficientY, out string errstring)
```

- controllerAddress: controllerAddress
- CalibrationCoefficientY: CalibrationCoefficientY
- errString: The failure reason

return: 0 in success and -1 on failure

**Description**

This function is used to process synchronuous PY Get command which is used to Get the calibration value for Y channel. Refer to the CONEX-LDS User’s Manual for more information on this command.

**PY_Set**

**Syntax**

```c
int PY_Set(int controllerAddress, double CalibrationCoefficientY, out string errstring)
```

- controllerAddress: controllerAddress
- CalibrationCoefficientY: CalibrationCoefficientY
- errString: The failure reason

return: 0 in success and -1 on failure

**Description**

This function is used to process synchronuous PY Set command which is used to Set the calibration value for Y channel. Refer to the CONEX-LDS User’s Manual for more information on this command.

**RG_Get**

**Syntax**

```c
int RG_Get(int controllerAddress, out int Range, out string errstring)
```

- controllerAddress: controllerAddress
- Range: Range
- errString: The failure reason

return: 0 in success and -1 on failure

**Description**

This function is used to process synchronuous RG Get command which is used to Get the range value. Refer to the CONEX-LDS User’s Manual for more information on this command.
**RG_Set**

**Syntax**

```c
int RG_Set(int controllerAddress, int Range, out string errstring)
```

- `controllerAddress`: controllerAddress
- `Range`: Range
- `errString`: The failure reason

**Description**

This function is used to process synchronous RG Set command which is used to Set the range value. Refer to the CONEX-LDS User’s Manual for more information on this command.

**RS**

**Syntax**

```c
int RS(int controllerAddress, out string errstring)
```

- `controllerAddress`: controllerAddress
- `errString`: The failure reason

**Description**

This function is used to process synchronous RS Set command which is used to Reset the controller. Refer to the CONEX-LDS User’s Manual for more information on this command.

**RS485**

**Syntax**

```c
int RS485(int controllerAddress, out string errstring)
```

- `controllerAddress`: controllerAddress
- `errString`: The failure reason

**Description**

This function is used to process synchronous RS## Set command which is used to Reset the controller’s address to 1. Refer to the CONEX-LDS User’s Manual for more information on this command.
**SA_Get**

**Syntax**

```c
int SA_Get(int controllerAddress, out int Address, out string errstring)
    controllerAddress: controllerAddress
    Address: Address
    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 the controller’s RS-485 address. Refer to the CONEX-LDS User’s Manual for more information on this command.

**SA_Set**

**Syntax**

```c
int SA_Set(int controllerAddress, int Address, out string errstring)
    controllerAddress: controllerAddress
    Address: Address
    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 the controller’s RS-485 address. Refer to the CONEX-LDS User’s Manual for more information on this command.

**SL_Get**

**Syntax**

```c
int SL_Get(int controllerAddress, out int LowlevelPowerThreshold, out string errstring)
    controllerAddress: controllerAddress
    LowlevelPowerThreshold: LowlevelPowerThreshold
    errString: The failure reason
return: 0 in success and -1 on failure
```

**Description**

This function is used to process synchrounous SL Get command which is used to Get low level power threshold for valid measurements. Refer to the CONEX-LDS User’s Manual for more information on this command.
**SL_Set**

**Syntax**

```c
int SL_Set(int controllerAddress, int LowlevelPowerThreshold, out string errstring)
    controllerAddress: controllerAddress
    LowlevelPowerThreshold: LowlevelPowerThreshold
    errString: The failure reason
return: 0 in success and -1 on failure
```

**Description**

This function is used to process synchronous SL Set command which is used to Set low level power threshold for valid measurements. Refer to the CONEX-LDS User’s Manual for more information on this command.

**SR_Get**

**Syntax**

```c
int SR_Get(int controllerAddress, out int HighlevelPowerThreshold, out string errstring)
    controllerAddress: controllerAddress
    HighlevelPowerThreshold: HighlevelPowerThreshold
    errString: The failure reason
return: 0 in success and -1 on failure
```

**Description**

This function is used to process synchronous SR Get command which is used to Get high level power threshold for valid measurements. Refer to the CONEX-LDS User’s Manual for more information on this command.

**SR_Set**

**Syntax**

```c
int SR_Set(int controllerAddress, int HighlevelPowerThreshold, out string errstring)
    controllerAddress: controllerAddress
    HighlevelPowerThreshold: HighlevelPowerThreshold
    errString: The failure reason
return: 0 in success and -1 on failure
```

**Description**

This function is used to process synchronous SR Set command which is used to Set high level power threshold for valid measurements. Refer to the CONEX-LDS User’s Manual for more information on this command.
SU_Get

Syntax
int SU_Get(int controllerAddress, out string Units, out string errstring)
  controllerAddress: controllerAddress
  Units: Units
  errString: The failure reason
return: 0 in success and -1 on failure

Description
This function is used to process synchronous SU Get command to set the the measurement units. Refer to the CONEX-LDS User’s Manual for more information on this command.

SU_Set

Syntax
int SU_Set(int controllerAddress, string Units, out string errstring)
  controllerAddress: controllerAddress
  Units: Units
  errString: The failure reason
return: 0 in success and -1 on failure

Description
This function is used to process synchronous SU Set command to set the measurement units. Refer to the CONEX-LDS User’s Manual for more information on this command.

TB

Syntax
int TB(int controllerAddress, string inError, out string outError, out string errString)
  controllerAddress: Address of Controller
  inError: inError.
  outError: outError
  errString: The failure reason
return: 0 in success and -1 on failure

Description
This function is used to process synchronous TB Get command which explains the meaning of the error string. Refer to the CONEX-LDS Controller’s manual to get the command description.
**TE**

**Syntax**

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

- `controllerAddress`: controllerAddress
- `LastCommandError`: LastCommandError
- `errString`: The failure reason

**Description**

This function is used to process synchronous TE Get command which is used to Get the last command error. Refer to the CONEX-LDS User’s Manual for more information on this command.

**TS**

**Syntax**

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

- `controllerAddress`: controllerAddress
- `ErrorCode`: ErrorCode
- `StatusCode`: StatusCode
- `errString`: The failure reason

**Description**

This function is used to process synchronous TS Get command which is used to Get the positioner error and current controller state. Refer to the CONEX-LDS User’s Manual for more information on this command.

**VE**

**Syntax**

```c
int VE(int controllerAddress, out string ControllerVersion, out string errstring)
```

- `controllerAddress`: controllerAddress
- `ControllerVersion`: ControllerVersion
- `errString`: The failure reason

**Description**

This function is used to process synchronous VE Get command which is used to Get controller revision information. Refer to the CONEX-LDS User’s Manual for more information on this command.
**ZT**

**Syntax**

```c
int ZT(int controllerAddress, out List<string> ConfigurationParameters, out string errstring)
```

- `controllerAddress`: `controllerAddress`
- `ConfigurationParameters`: `ConfigurationParameters`
- `errString`: The failure reason

**Description**

This function is used to process synchronous ZT Get command which is used to Get all controller parameters. Refer to the CONEX-LDS User’s Manual for more information on this command.
3.0 Python example

```python
# Initialization Start
# The script within Initialization Start and Initialization End is needed for
# properly initializing Command Interface for Conex-LDS instrument.
# The user should copy this code as is and specify correct paths here.
import sys
print "Adding location of Newport.CONEXLDS.CommandInterface.dll to
sys.path"
sys.path.append(r'C:\Program Files\Newport\MotionControl\CONEX-LDS\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.CONEXLDS.CommandInterface.dll")
from CommandInterfaceConexLDS import *

import System

# Constant
ON = 1
OFF = 0

# Procedure to open communication with instrument.
#*******************************************************************************
def CONEXLDS_Open(instrumentKey):
    # Create a CONEX-LDS instance
    LDS = ConexLDS()

    print 'Instrument Key=>', instrumentKey
    ret = LDS.OpenInstrument(instrumentKey)
    print 'OpenInstrument => ', ret
    return LDS

#*******************************************************************************
# Procedure to close communication.
#*******************************************************************************
def CONEXLDS_Close(LDS):
    LDS.CloseInstrument()

#*******************************************************************************
# Procedure to get the controller version (VE)
#*******************************************************************************
def CONEXLDS_GetControllerVersion(LDS, address, flag):
```
result, version, errString = LDS.VE(address)
if flag == 1:
    if result == 0 :
        print 'CONEX-LDS firmware version => ', version
    else:
        print 'VE Error => ',errString
return result, version

#*************************************************
# Procedure to get the laser status (GP Command)
#*************************************************
def CONEXLDS_GetPositionsAndLightLevel (LDS, address, flag):
    # Get X, Y positions and light level Using GP Command
    result, posX, posY, lightLevel, errString = LDS.GP(address)
    if flag == 1:
        if result == 0 :
            print 'Position X => ', posX
            print 'Position Y => ', posY
            print 'Light level => ',lightLevel, "%"
        else:
            print 'GP Error => ',errString
    return result, posX, posY, lightLevel

#*************************************************
# Procedure to get the laser status (LB? Command)
#*************************************************
def CONEXLDS_GetLaserStatus (LDS, address, flag):
    result, laserStatus, errString = LDS.LB_Get(address)
    if flag == 1:
        if result == 0 :
            print 'Laser status => ', laserStatus
        else:
            print 'LB_Get Error => ',errString
    return result, laserStatus

#*************************************************
# Procedure to set the laser status (LB Command)
#*************************************************
def CONEXLDS_SetLaserStatus (LDS, address, laserStatus, flag):
    result, errString = LDS.LB_Set(address, laserStatus)
    if flag == 1:
        if result != 0 :
            print 'LB_Set Error => ',errString
    return result

#*************************************************
# Main
#*************************************************

# Initialization
instrument="COM19"
displayFlag = 1
address = 1

# Create a CONEX-LDS interface and open communication.
LDS = CONEXLDS_Open(instrument)

# Get controller revision information
result, version = CONEXLDS_GetControllerVersion(LDS, address, displayFlag)

# Get laser status
result, iLaserStatus = CONEXLDS_GetLaserStatus(LDS, address, displayFlag)
if result == 0 :
    # If the laser is OFF then turn the laser ON
# Check and refresh the laser status
if iLaserStatus == OFF:
    print 'Laser is OFF'
    result = CONEXLDS_SetLaserStatus(LDS, address, ON, displayFlag)
    result, iLaserStatus = CONEXLDS_GetLaserStatus(LDS, address, displayFlag)

# Get positions
if iLaserStatus == ON:
    print 'Laser is ON'
    # Get X, Y positions and light level
    returnValue, positionX, positionY, lightLevel = CONEXLDS_GetPositionsAndLightLevel(LDS, address, displayFlag)

# close communication.
CONEXLDS_Close(LDS)

print 'End of script'
### Service Form

**Your Local Representative**

Tel.: __________________
Fax: __________________

| Name: _______________________________________________ | Return authorization #: ____________________________ |
| Company: _____________________________________________ | (Please obtain prior to return of item) |
| Address: _____________________________________________ | Date: ____________________________________________ |
| Country: _____________________________________________ | Phone Number: _________________________________ |
| P.O. Number: _________________________________________ | Fax Number: _________________________________ |
| Item(s) Being Returned: ______________________________ | Model#: ________________________________________ |
| P.O. Number: _________________________________________ | Serial #: ____________________________________ |

**Description:** __________________________________________________________________________________________

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