App. Note Code: 1D-W

APPLICATION NOTE

DNP3 with Campbell Scientific Dataloggers

4/15

DNP3 with Campbell Scientific Dataloggers

What is DNP3?

DNP3 (Distributed Network Protocol) is designed to optimize transmission of data and control commands from a master computer to one or more remote devices or outstations. DNP3 is an open standard developed by Westronic, Inc. for use in SCADA applications. It is non-proprietary and is available at *www.dnp.org*.

In the context of Campbell Scientific instrumentation, a remote device or outstation is one of the following dataloggers:

- CR6 Series
- CR800 Series
- CR1000
- CR3000

Features

Following are principle features of DNP3:

- Standardized protocol to ensure interoperability among vendors.
- Efficient use of bandwidth through RBE (Report By Exception). Only a change in data (an **Event**) is reported rather than reporting all data each time a device is polled. RBE is a principle reason for the success of DNP3.
- Allows a Campbell Scientific datalogger, via **Unsolicited Responses**, to transmit events without first receiving a request.
- Provides a high degree of reliability through communication layers, which provide an organized scheme for data and control data transmissions in an organized manner.
- Minimizes the impact of noise and signal distortion on communication circuits.

Implementation

Terms

static data

> Current values of data points at the instant they are transmitted. Examples include **On** or **Off** state of a bi-state device, or a temperature or water-level value.

event

The crossing of a threshold by an analog value, or the switch of a binary input. For example, an event occurs when an analog value changes by more than the configured dead-band or threshold, or when a binary input changes from **On** to **Off**.

An event can be recorded with or without a timestamp. Timestamps enable the master to generate time-sequence reports.

Event data are kept in the datalogger until confirmation has been received from the master that the data are received. In other words, event data will not be discarded until after the datalogger receives confirmation that the master has received the data.

variations

Data format. For example, static data may be presented as 16-bit or 32-bit integers, or as 16-bit or 32-bit floating-point values. These data may or may not include a flag to indicate if the source is online. Event data can be represented in the same formats as static data, but also have additional variations to allow for timestamps.

objects

The bits and bytes for each index point in a message. A sending device must format data for parsing and interpretation by the receiving device. Objects do the following:

- differentiate between static and event data
- distinguish among data types, such as the following:
  - counters
  - binary or analog inputs
  - binary or analog outputs
  - control commands

The format of an object is determined by group and variation number. For simplicity, the Campbell Scientific DNP3 implementation does not refer to groups since the object includes both group number and variation.

class

DNP3 has four classes of data. Static data are **Class 0**. Event data are **Class 1**, **Class 2**, and **Class 3**, with no logical differentiation between these classes. Originally, **Class 1** events were higher priority than **Class 2**, and **Class 2** higher than **Class 3**. However, in current applications, DNP3 allows you to organize classes in the strategy that works best for your application. By way of analogy, for those familiar with CRBasic data classification, **Class 0** holds current points in time similar to the **Public** table. **Classes 1**, **2**, and **3** hold event history similar to user-defined final-memory data tables.

CRBasic Programming Language

DNP3 is implemented in Campbell Scientific dataloggers through a program you write in CRBasic. CRBasic is a programming language similar to BASIC. Programs are written with *CRBasic Editor*, which is part of *LoggerNet* datalogger support software.

The following CRBasic instructions implement DNP3:

- **DNP()**
- **DNPUpdate()**
- **DNPVariable()**

Each instruction is discussed in the following sections with sufficient detail to aid in a basic understanding of the Campbell Scientific implementation. Additional resources are available in *CRBasic Editor Help* and the corresponding datalogger operator's manual.

DNP() Instruction

```
DNP(ComPort, BaudRate, Confirmation, TimeOffset(opt),
MaxTimeDiff(opt), DNPTLS)
```

DNP() programs a Campbell Scientific datalogger as a remote device or outstation. It is executed once and is usually placed between the **BeginProg** and **Scan()** statements. See CRBasic Example 16, *Complete Program for a Sample Application* (p. 10), for an example of placement.

ComPort and ***BaudRate*** specify the datalogger communication port and baud rate with which to communicate with the master.

Confirmation determines whether data-link-layer confirmation (DLLC) is enabled. It also sets the timeout interval for data-link-layer and application-layer confirmation.

DLLC requires extra time for sending and receiving confirmation messages and for waiting on multiple timeouts if retries are configured. It is always disabled for a TCP/IP link. In most cases, DLLC is considered redundant because the datalogger will always use a higher-level application layer confirmation when transmitting event data or multi-fragment responses. The confirmation parameter is entered in the form of **XSSS**. **X = 0** enables DLLC. **X = 1** disables. **SSS** is the number of seconds the datalogger will wait for a response to DLLC or application-layer confirmation before timing out.

TimeOffset and ***MaxTimeDiff*** (optional) keep the datalogger clock set to local time when the DNP3 master is sending time-synchronization commands to the datalogger.

DNPTLS is optional and available only with a CR6 datalogger programmed to enable TLS (Transport Layer Security). If the parameter is set to **0** or is omitted, TLS is disabled. If the parameter is set to **1**, TLS is enabled. On account of slower processor speeds, ***DNPTLS*** is not available in CR800, CR1000, or CR3000 dataloggers.

DNP() Examples

CRBasic Example 1. DNP() Code Snip: Using TCP/IP for the DNP Port

```
'To use TCP/IP for the DNP port, set COMPort to 20000.  20000 is the
'default port number for DNP over TCP.  This port number must match in the master.

'DNP(ComPort, Baudrate, Confirmation, TimeOffset(opt), MaxTimeDiff(opt), DNPTLS)
DNP(20000,115200,1000) 'Use IP for communication port, confirmation disabled
```

CRBasic Example 2. DNP() Code Snip: Using RS-232 Port for the DNP Port

```
'To use the nine-pin RS-232 port for DNP, set the COMPort to COMRS232.

'DNP(ComPort, Baudrate, Confirmation, TimeOffset(opt), MaxTimeDiff(opt), DNPTLS)
DNP(COMRS232,115200,1000) 'Baud rate 115200, confirmation disabled
```

CRBasic Example 3. DNP() Code Snip: Using Data Link Layer and Application Confirmations

```
'To enable data link layer confirmation and set the application confirmation
'to 10 seconds, set Confirmation to 0010.

'DNP(ComPort, Baudrate, Confirmation, TimeOffset(opt), MaxTimeDiff(opt), DNPTLS)
DNP(COMRS232,115200,0010) 'Baud rate 115200, confirmation enabled
```

NOTE The use of data-link-layer confirmation (DLLC) is not recommended for most applications. See previous description of ***Confirmation*** parameter.

CRBasic Example 4. DNP() Code Snip: Setting Local Time Offset

```
'To set a local-time offset of -1 hour, set the TimeOffset to 3600.  This
'value, in seconds, will be subtracted from local time for DNP objects and variations
'that return time.

'DNP(ComPort, Baudrate, Confirmation, TimeOffset(opt), MaxTimeDiff(opt), DNPTLS)
DNP(COMSDC7,115200,1000,3600,-1) 'DNP over SDC7, confirmation disabled, local-
                                 'time offset of 1 hr, ignore time-
                                 'synchronization commands from the master
```

NOTE If the datalogger is connected to both a DNP3 master and a *LoggerNet* datalogger support software server, a ***-1*** for the ***MaxTimeDiff*** parameter can be entered to avoid problems that arise from a DNP3 master and a *LoggerNet* server both attempting to set the datalogger clock.

CRBasic Example 5. DNP() Code Snip: Enabling TLS on the DNP TCP/IP Port (CR6 Only)

```
'To enable TLS on a TCP/IP

'DNP(ComPort, Baudrate, Confirmation, TimeOffset(opt), MaxTimeDiff(opt), DNPTLS)
DNP(20000,115200,1000,0,0,1) 'DNP over IP; TLS enabled
```

DNPVariable() Instruction

```
DNPVariable (Source, Swath, DNPObject, DNPVariation, DNPClass,
DNPFlag, DNPEvent, DNPNumEvents)
```

DNPVariable() configures DNP3 data for collection and transmission. For example, it can determine whether only static data or event data are collected. This instruction is usually executed only once for each data type. It typically is inserted between the **BeginProg** and **Scan()** instructions. See CRBasic Example 16, *Complete Program for a Sample Application* *(p. 10)*, for an example of placement. Multiple **DNPVariable()** instructions are used in programs to configured for multiple data types.

Source is a variable array that specifies the source of data that will populate the DNP datalogger CRBasic array. ***Source*** variable array declarations:

- For **Analog Input** objects, declare CRBasic array **As Long** (integer). If object variation is single-precision floating point, however, declare array **As FLOAT**.
- For **Analog Output** objects, declare array **As Long**. If object variation is single-precision floating point, declare **As FLOAT**.
- For **Binary Input** objects, declare **As BOOLEAN**
- For **Binary Output** objects, declare **As BOOLEAN**

Swath must be smaller than or equal to the declared size of the ***Source*** variable array.

DNPObject and ***DNPVariation*** parameters specify object and variation.

DNPClass specifies a classification strategy for different object types (Class 0, 1, 2, or 3).

DNPflag controls data quality flags for variations that include flags.

DNPEvent sets thresholds that constitute a change event.

DNPNumEvents specifies how many events are stored in history before they are sent to a master. It is equivalent to the event buffer size.

DNPVariable() Examples

Data type storage and event buffer

These examples store different data types and specify the size of an event buffer in the datalogger by specifying different DNP objects and variations. For a complete list of supported objects and variations, see *CRBasic Help* for the **DNPVariable()** instruction.

CRBasic Example 6. DNPVariable() Code Snip: Populating Static Analog Input Data Points

```
'Populate four static analog-input data points to a variable named Array_1().

'NOTE Array size is declared equal to or larger than Swath parameter.
'NOTE Class 0 is used for static data.

Public Array_1(4) as Long

'...

'DNPVariable (Source, Swath, DNPObj, DNPVar, DNPClass, DNPFlag, DNPEvent, DNPNumEvents)
DNPVariable (Array_1(),4,30,1,0,0,0,0) 'Object 30 variation 1 = static 32-bit analog
                                       'input.
```

CRBasic Example 7. DNPVariable() Code Snip: Populating Analog Events

```
'Populate four 32-bit analog events to the variable Array_1().
'Keep 100 events in history (100 events for each point)

'NOTE Must specify object 30 before using object 32 to store event data.
'NOTE Must use Class 1, 2, or 3 to store event history.

Public Array_1(4) as Long

'...

'DNPVariable (Source, Swath, DNPObj, DNPVar, DNPClass, DNPFlag, DNPEvent, DNPNumEvents)
DNPVariable (Array_1(),4,32,1,1,0,0,100) 'Object 32 variation 1 = 32-bit analog
                                         'change event without time.
```

CRBasic Example 8. DNPVariable() Code Snip: Populating Analog Input Events

```
'Populate four analog-input events as single-precision floating-point values with time.
'Keep 10 events in history.

Public Array_1(4) as Float

'...

'DNPVariable (Source, Swath, DNPObj, DNPVar, DNPClass, DNPFlag, DNPEvent, DNPNumEvents)
DNPVariable (Array_1(),4,32,7,1,0,0,10) 'Object 32 variation 7
```

CRBasic Example 9. DNPVariable() Code Snip: Populating Static Binary Input Points

```
'Populate 10 static binary input points to a variable named Binput()

Public Binput(10) as Boolean

'...

'DNPVariable (Source, Swath, DNPObj, DNPVar, DNPClass, DNPFlag, DNPEvent, DNPNumEvents)
DNPVariable (Binput(),10,1,2,0,0,0,0) 'Must assign object 1 before using object 2
```

Setting attributes of data values

Many readable DNP3 data objects have variations that include flags consisting of bit-string fields to indicate conditions or attributes of the associated data value. The ***DNPFlag*** parameter sets associated data values to ***online***, ***offline***, or ***online and restart***. Following is an example.

CRBasic Example 10. DNPVariable() Code Snip: Initializing DNP Flags

```
'Initialize DNP3 flags. 1 = on-line; 0 = off-line

Dim Analog_Inputs_Flag(9) As Long

'...

For i = 1 To 9
   Analog_Inputs_Flag(i) = 1
Next i

'DNPVariable (Source, Swath, DNPObj, DNPVar, DNPClass, DNPFlag, DNPEvent, DNPNumEvents)
DNPVariable (Analog_Inputs(),9,30,2,0,Analog_Inputs_Flag(),0,0) 'Object 30 variation
                                                                '2 = static data.
```

Defining an event

By default, any change, no matter how small, is an event. However, you can use the ***DNPEvent*** parameter to limit the changes that are considered events. CRBasic Example 11 defines a Boolean variable as a trigger with ***DNPEvent*** and uses logic to define an event.

CRBasic Example 11. Complete Program: Conditional Storage of an Event

```
'Only store an event if a counter is greater than or equal to 5
Public Array(4) As Long
Public Counter As Long
Public Trigger As Boolean

BeginProg
  DNP(20000,115200,1000) 'Use IP for comport, confirmation disabled
  DNPVariable(Array(),4,30,1,0,0,0,0) 'static analog data
  'must specify object 30 before using object 32
  DNPVariable(Array(),4,32,1,1,0,trigger,10) 'event analog data

  Scan(1,Sec,1,0) 'scan
    counter = counter +1
    Array()= counter
    If Counter>=5 Then Trigger=1 Else Trigger=0
    If Counter =10 Then Counter = 0 'reset counter to 0 at 10

    'Update DNP arrays
    DNPUpdate(1,10) 'slave, master
  NextScan
EndProg
```

DNPUpdate() Instruction

```
DNPUpdate (DNPSlaveAddr, DNPMasterAddr,
UnsolicitedResponseTimeout(opt),
UnsolicitedResponseRetries(opt), MasterConnectionHandle(opt))
```

DNPUpdate() is analogous to the CRBasic CallTable() instruction. It enables a datalogger to update the DNP array with new values. No data will be written to the DNP data array until **DNPUpdate()** is run. For this reason, **DNPUpdate()** is placed inside a **Scan**/**NextScan** structure after the elements of the DNP array that are to be updated.

DNPSlaveAddr assigns a DNP address to the datalogger

DNPMasterAddr specifies the addresses of masters the datalogger will respond to. Communications with up to three masters are supported. A separate **DNPUpdate()** instruction is required for each master.

Examples

CRBasic Example 12. Complete Program: Specifying Three Master Addresses and Updating DNP Elements

```
'Specifying three master addresses that can communicate with one
'datalogger and update DNP elements at end of each 1 second scan.

'Variables for Measurements
Public BattV
Public PTemp

'Declare Units for measurements
Units BattV = Volts
Units PTemp = Deg C

'Variable declarations for DNP3
Dim Analog_Inputs(2) As Long
Dim Analog_Inputs_Flag(2) As Long

'Data Table To Record 10-Minute Averaged Measurements
DataTable(TenMin,True,-1)
  DataInterval(0,10,Min,10)
  Minimum(1,BattV,FP2,False,False)
  Average(1,PTemp,FP2,False)
EndTable

'Main Program
BeginProg

'Set up Datalogger as DNP3 outstation
  DNP (20000,115200,1000) 'Set up comms port
  DNPVariable (Analog_Inputs(),2,30,2,0,0,0,0) 'Static data
  DNPVariable (Analog_Inputs(),2,32,2,1,0,0,120) 'Event data, 120 point buffer,
                                                 'any change, all events are
                                                 'class 1
    Scan (1,Sec,0,0)
      PanelTemp (PTemp,250)
      Battery (BattV)

    'Update DNP3 Analog Inputs
    Analog_Inputs(1) = BattV *10
    Analog_Inputs(2) = PTemp *10

    DNPUpdate (1,103) 'slave 1, master 103
    DNPUpdate (1,104) 'slave 1, master 104
    DNPUpdate (1,105) 'slave 1, master 105
    CallTable TenMin
  NextScan
EndProg
```

Unsolicited Responses

Unsolicited responses are similar in concept to the functionality of the CRBasic **SendData()** and **SendVariables()** instructions.

DNPUpdate() can enable an unsolicited response transmission. This is capability was implemented with datalogger operating system v.28. It enables the datalogger to transmit events without first receiving a request. This mode is useful if the master requires notification soon after a change occurs rather

than waiting for the master to poll the datalogger. Optional **DNPUpdate()** parameters include an unsolicited response timeout, number of retries, and a master connection handle.

If ***UnsolicitedResponseConfirmationTimeout*** and ***UnsolicitedResponseRetries*** are not present in **DNPUpdate()**, or if ***UnsolicitedResponseConfirmationTimeout*** = ***0***, then unsolicited responses are disabled.

If the ***UnsolicitedResponseRetries*** = ***0*** then unsolicited responses, with data, will retry indefinitely or until receipt of the data is confirmed by the master.

Examples

CRBasic Example 13. DNPUpdate() Code Snip: Sending Unsolicited Response from Datalogger 1 with Infinite Retries

```
'Send unsolicited response to master address 3 from datalogger 1;
'wait up to 5 seconds for confirmation that response received; retry indefinitly or
'until response is confirmed by master
DNPUpdate(1,3,5,0)
```

CRBasic Example 14. DNP() Code Snip: Sending Unsolicited Response from Datalogger 1 with Three Retries

```
'Send unsolicited response to master address 3 from datalogger 1;
'wait up to 5 seconds for confirmation that response received; retry 3 times or
'until response is confirmed by master
DNPUpdate(1,3,5,3)
```

Master Connection Handle

DNPUpdate() optionally specifies a master connection handle when the datalogger needs to initiate a TCP connection to the master rather than wait for the master to initiate a connection to the datalogger.
MasterConnectionHandle is a variable declared **As LONG**.

Example

CRBasic Example 15. Complete Program: Sending Unsolicited Response from Datalogger 1 via TCP/IP

```
'Send unsolicited response to master address 10 from datalogger 1 via
'a TCP connection initiated by the slave.
Public IP_Socket As Long
Public Binput(2) As Boolean
Public counter As Long
Const = IPAddress ="192.168.xx.xx"

BeginProg
  DNP(20000,115200,1000)
  DNPVariable (Binput(),2,1,2,0,0,0,0)
  DNPVariable (Binput(),2,2,1,1,0,0,2)
    Scan(1,Sec,1,0)
      IP_Socket = TCPOpen(IPAddress,20000,0)
      counter = counter +1
      DNPUpdate(1,10,5,0,IP_Socket)
    NextScan
EndProg
```

Complete Program Example

CRBasic Example 16. Complete Program for a Sample Application

```
'Solar Energy Monitoring Station

'Written for CR1000 and CR800 series dataloggers. Adaptable to CR6 and CR3000 with
'minor changes.

'Variables for Measurements
Public BattV
Public PTemp
Public Air_Temp
Public Irradiance_GH
Public Irradiance_POA
Public WindSpeed
Public WindDir
Public Rain
Public Daily_Rain

'Declare Units for measurements
Units BattV = Volts
Units PTemp = Deg C
Units Air_Temp = Deg C
Units Irradiance_GH = W/m^2
Units Irradiance_POA = W/m^2
Units WindSpeed = meters/second
Units WindDir = degrees
Units Rain = mm

'Constants used to set up com2 in ppp mode
Const GH_Sensitivity = 14.2
Const POA_Sensitivity = 12.29
'
'Variable declarations for DNP3
Dim i
Dim Heartbeat
Dim Analog_Inputs(9) As Long
Dim Analog_Inputs_Flag(9) As Long

'Data table to record 1 minute or 10 minute averaged measurements
DataTable(TenMin,True,-1)
  DataInterval(0,10,Min,10)

  Minimum(1,BattV,FP2,False,False)
  Average(1,PTemp,FP2,False)

  Average(1,Irradiance_GH,FP2,False)
  StdDev(1,Irradiance_GH,FP2,False)

  Average(1,Irradiance_POA,FP2,False)
  StdDev(1,Irradiance_POA,FP2,False)

  Average(1,Air_Temp,FP2,False)
  StdDev(1,Air_Temp,FP2,False)

  Average(1,WindSpeed,FP2,False)
  StdDev(1,WindSpeed,FP2,False)
  WindVector(1,WindSpeed,WindDir,FP2,False,0,0,4)
  FieldNames("WindDir_D1_WVT,WindDir_SD1_WVT")

  Totalize(1,Rain,FP2,0)
EndTable

'Main Program
BeginProg
```

```
  'Set up the Solar800 as a DNP3 remote device
  For i = 1 To 9
    Analog_Inputs_Flag(i) = 1 'Initialize DNP3 flags to online
  Next i

  DNP(20000,115200,1000) 'Set up comms port
  DNPVariable(Analog_Inputs(),9,30,2,0,Analog_Inputs_Flag(),0,0) 'Static data

  'Event data, 120 point buffer, any change, all events are class 1
  DNPVariable(Analog_Inputs(),9,32,2,1,Analog_Inputs_Flag(),0,120)

  Scan(1,Sec,0,0)
    PanelTemp(PTemp,250)
    Battery(BattV)

    'Create a Heartbeat
    If Heartbeat = 100 Then 'Increments every second up to 100, then starts
                            'over at 0
      Heartbeat = 1
    Else
      Heartbeat = Heartbeat + 1
    EndIf

    'Global Horizontal Irradiance measurement
    VoltDiff(Irradiance_GH,1,mV25,1,True,0,_60Hz,1000/GH_Sensitivity,0)

    'POA Irradiance measurement
    VoltDiff(Irradiance_POA,1,mV25,2,True,0,_60Hz,1000/POA_Sensitivity,0)

    'Wind speed measurement
    PulseCount(WindSpeed,1,1,1,1,0.75,0.2)
    If WindSpeed<0.21 Then WindSpeed=0

    'Wind direction measurement
    BrHalf(WindDir,1,mV2500,5,1,1,2500,True,0,_60Hz,352,0)
    If WindDir>=360 Then WindDir=0

    'Rain Gage measurement
    PulseCount(Rain,1,2,2,0,0.254,0)

    '109 Temperature Probe measurement degrees C
    Therm109(Air_Temp,1,6,1,0,_60Hz,1,0)

    'Update DNP3 Analog Inputs
    Analog_Inputs(1) = Heartbeat
    Analog_Inputs(2) = BattV *10
    Analog_Inputs(3) = PTemp *10
    Analog_Inputs(4) = Irradiance_GH * 10
    Analog_Inputs(5) = Irradiance_POA * 10
    Analog_Inputs(6) = WindSpeed * 10
    Analog_Inputs(7) = WindDir * 10
    Analog_Inputs(8) = Air_Temp * 10
    Analog_Inputs(9) = Rain * 10

    DNPUpdate(1,10) 'slave, master
    CallTable TenMin
  NextScan
EndProg
```

Campbell Scientific Companies

Campbell Scientific, Inc. (CSI)
815 West 1800 North
Logan, Utah 84321
UNITED STATES
www.campbellsci.com • info@campbellsci.com

Campbell Scientific Africa Pty. Ltd. (CSAf)
PO Box 2450
Somerset West 7129
SOUTH AFRICA
www.csafrica.co.za • cleroux@csafrica.co.za

Campbell Scientific Australia Pty. Ltd. (CSA)
PO Box 8108
Garbutt Post Shop QLD 4814
AUSTRALIA
www.campbellsci.com.au • info@campbellsci.com.au

Campbell Scientific (Beijing) Co., Ltd.
8B16, Floor 8 Tower B, Hanwei Plaza
7 Guanghua Road
Chaoyang, Beijing 100004
P.R. CHINA
www.campbellsci.com • info@campbellsci.com.cn

Campbell Scientific do Brasil Ltda. (CSB)
Rua Apinagés, nbr. 2018 ─ Perdizes
CEP: 01258-00 ─ São Paulo ─ SP
BRASIL
www.campbellsci.com.br • vendas@campbellsci.com.br

Campbell Scientific Canada Corp. (CSC)
14532 – 131 Avenue NW
Edmonton AB T5L 4X4
CANADA
www.campbellsci.ca • dataloggers@campbellsci.ca

Campbell Scientific Centro Caribe S.A. (CSCC)
300 N Cementerio, Edificio Breller
Santo Domingo, Heredia 40305
COSTA RICA
www.campbellsci.cc • info@campbellsci.cc

Campbell Scientific Ltd. (CSL)
Campbell Park
80 Hathern Road
Shepshed, Loughborough LE12 9GX
UNITED KINGDOM
www.campbellsci.co.uk • sales@campbellsci.co.uk

Campbell Scientific Ltd. (CSL France)
3 Avenue de la Division Leclerc
92160 ANTONY
FRANCE
www.campbellsci.fr • info@campbellsci.fr

Campbell Scientific Ltd. (CSL Germany)
Fahrenheitstraße 13
28359 Bremen
GERMANY
www.campbellsci.de • info@campbellsci.de

Campbell Scientific Spain, S. L. (CSL Spain)
Avda. Pompeu Fabra 7-9, local 1
08024 Barcelona
SPAIN
www.campbellsci.es • info@campbellsci.es

Please visit www.campbellsci.com to obtain contact information for your local US or international representative.