BioRadio Software Development Kit
LabVIEW™ Driver Guide

Telephone: (216) 791-6720 or Toll-free 1-877-CleveMed (1-877-253-8363)
9:00 a.m. - 5:00 p.m. EST
Monday - Friday
Fax: (216) 791-6739

E-Mail: Customer Support: support@CleveMed.com
Sales: sales@CleveMed.com
Web: http://www.CleveMed.com

Mailing Address: Cleveland Medical Devices Inc.
4415 Euclid Avenue, Fourth Floor
Cleveland, Ohio 44103

Table of Contents

Introduction

The software DLL (Dynamic Link Library) interface to the BioRadio allows for programmatic interaction in Windows applications. Such interaction has been designed for National Instruments' LabVIEW development system, allowing BioRadio (and RatPaak) communications and control from within LabVIEW and LabVIEW-based applications. This document describes the LabVIEW Virtual Instruments (VIs) provided for such utility, and their appropriate usage.

The BioRadio SDK LabVIEW Driver consists of the following Win32 DLLs; the software interface to the devices:

```
DeviceCheckDLL.dll
BioRadioDLL.dll
BioRadio150DLL.dll
RatPaakDLL.dll
```

and the following LabVIEW VIs, which make calls to functions within the DLLs:

```
BioRadio_FindAndChooseReceiver.vi
BioRadio_DialogChooseBioRadio.vi
BioRadio_Start.vi
BioRadio_StartBaseComm.vi
BioRadio_Read.vi
BioRadio_Stop.vi
BioRadio_StopBaseComm.vi
BioRadio_Example.vi
```

Basic Operation

Communicating with a BioRadio through LabVIEW is divided into six main actions, listed here and described in more detail further in this document. Note that *for the RatPaak*, Starting Base Communication is accomplished by Starting Acquisition, and Stopping Base Communication is accomplished by Stopping Acquisition; only one Start and one Stop routine are needed.

- **Finding and Choosing Attached Devices**
  Identify attached BioRadio receivers and, if multiple exist, choose between them.
- **Starting Base Communication**
  Create a software device object and start communication between the PC and Computer Unit.
- **Starting Acquisition**
  Start communication between the Computer Unit and User Unit, attend to device configuration and communication parameters, and begin acquiring data.
- **Acquiring Data**
  Read and interpret scaled data from the port buffer, and acquire transmission statistics.
- **Stopping Acquisition**
  Stop data acquisition and communication between Computer Unit and User Unit.

- **Stopping Base Communication**

  Stop communication between PC and Computer Unit, and destroy the software device object.

Notes

BioRadio Models

The LabVIEW Driver uses multiple DLLs to support various BioRadio models. The DLL interface for device discovery, **DeviceFinder DLL**, is designed to work with the BioRadio 150 and RatPaak. Individual device control is handled by a few DLLs: **BioRadio DLL**, **BioRadio150 DLL, and RatPaak DLL**.

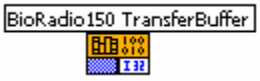
Each LabVIEW VI is designed to operate with both the BioRadio 150 and RatPaak devices. Since each model requires its own DLL communications interface, the VIs contain logic to choose the appropriate DLL for the model being used.

Multiple Devices and the Object Handle

The BioRadio SDK allows for operation of and acquisition from multiple BioRadios and Ratpaaks, simultaneously. Each time a device object is created, `BioRadio_StartBaseComm.vi` (or `BioRadio_Start.vi` for the RatPaak) is run with parameters referencing a valid Computer Unit, the handle reference to the object is returned. Subsequent VIs called to operate upon this device must be provided the corresponding object's handle.

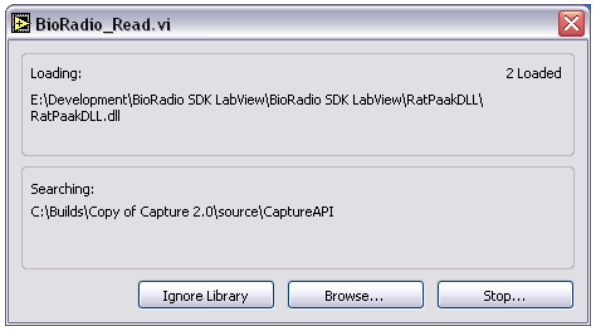
Paths to DLLs in Driver VIs

LabVIEW's **Call Library Function** node, responsible for making calls to DLL functions, specifies absolute paths to DLLs.



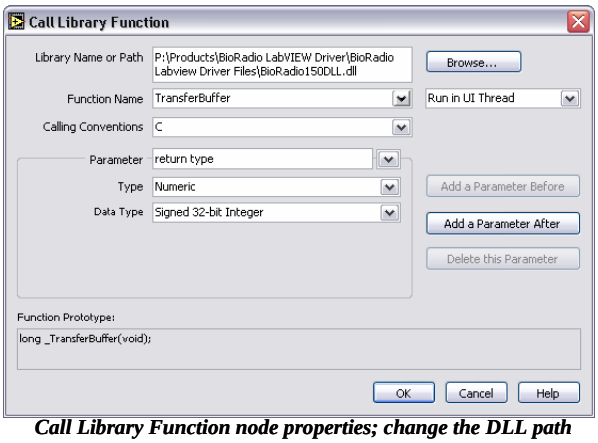
*a **Call Library Function** node*

If, upon opening the driver VIs, the DLL is not located at the path specfied, (such as on first run,) LabVIEW searches for the files, and displays a dialog while doing so.



LabVIEW searches for a DLL

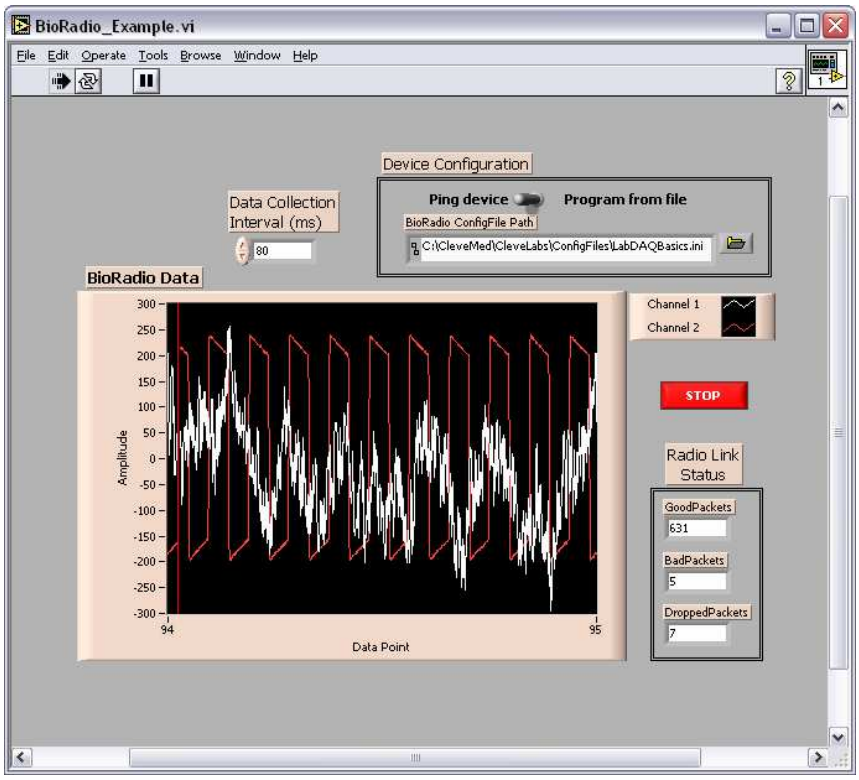
If the search utility does not immediately find the DLLs, click **Browse** to locate the files yourself. All DLL references in the VI will be updated, and saved when the VI is saved. These paths are always manually editable from **Call Library Function** node properties within a VI.



Call Library Function node properties; change the DLL path

BioRadio Example VI

`BioRadio_Example.vi` is included to illustrate use of the driver functions to acquire data from the BioRadio; a starting point for creating your own custom BioRadio applications. This VI, as provided, will operate properly with a BioRadio or RatPaak device programmed for two (2) data input channels. The task of customization is left to the developer.



BioRadio Config File Path

The example VI, by default, pings the BioRadio User Unit for its current device configuration. If you are using a RatPaak, or wish to program the BioRadio to a particular configuration at Start, change the toggle in the **Device Configuration** panel to **Program from file**, and specify the path to the appropriate file before running the VI. Consult the BioRadio User's Manual for more information on configuration files.

Data Collection Interval

The **Data Collection Interval** specifies how often (in milliseconds) the software collects data from the PC's communication port where the Computer Unit has deposited it. The port has a finite buffer; only so much data can accumulate there between collections (when cleared) before the buffer fills and is incapable of holding more. Therefore, if the data collection interval is set too high, the buffer will overfill and data will be lost (the **Dropped Packets** count will increase). The maximum time to which you can set the data collection interval, while avoiding dropped

packets, is dependent upon: buffer size, PC speed, and what else is taking up processing time in the computer. 80ms (milliseconds) between reads is a typical value.

Radio Link Status

The **Radio Link Status** box provides three metrics on the radio link between the BioRadio transmitter and receiver, measured in radio data packets. A packet of information sent from the transmitter to the receiver includes a time stamp, the sampled data, and other variables to allow software processing. A packet can only include so much information as its size does not vary. Therefore, if the unit is programmed to acquire data on only 1 or 2 inputs, a packet may have up to 3 samples for each input, whereas a packet including data over eight inputs will include only one sample per input.

The **Good Packets** indicator reflects the number of error-free data packets received from the User Unit. The number of **Good Packets** should increase rapidly while the transmitter is on. **Bad Packets** is the count of data packets received from the User Unit but found to be corrupted. **Dropped Packets** is the total number of packets lost in transmission and padded with the **Bad Data Value**. The **Dropped Packets** total includes packets never received as well as **Bad Packets**. Both **Bad** and **Dropped Packets** should remain close to zero, provided a good transmission between Computer Unit and User Unit, and provided data is collected regularly. All three indicators are reset each time acquisition is started.

Finding and Choosing Attached Devices

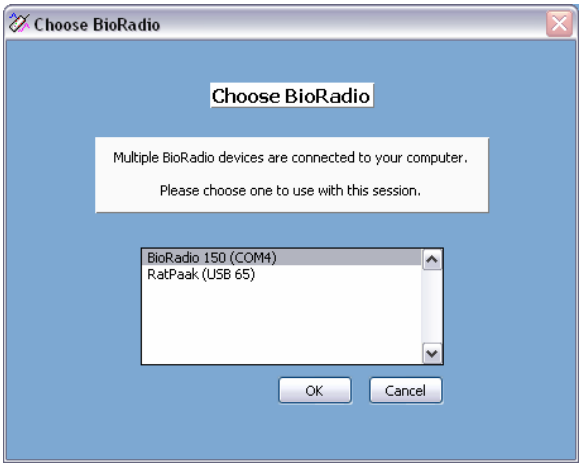
`BioRadio_FindAndChooseReceiver.vi`

Overview

The first step in communicating with a BioRadio is generally to discover and identify attached BioRadio receivers and, if multiple exist, to choose between them.

The **BioRadio Find and Choose Receiver VI** uses the `FindReceivers` function provided by the **DeviceCheck DLL** to discover BioRadio and RatPaak Computer Units connected to the user's PC. The DLL function displays a dialog informing the user that searching is in progress, and acts in different ways depending on the results of the search:

- *If exactly one BioRadio receiver is found*, the function returns the name of the device, ("`BioRadio 150`", or "`RatPaak`",) the number of the port to which it is attached, and whether the device is acting as a "legacy" COM-port device, or "non-legacy" USB.
- *If no BioRadio device is found*, outputs reflect this.
- *If multiple BioRadio devices are found*, a modal dialog (`BioRadio_DialogChooseBioRadio.vi`) is presented to the user to allow selection between the discovered devices (shown right.) The user can choose a device, or cancel the dialog to choose none. The appropriate information regarding this choice is then returned by the VI following the previous prescription.

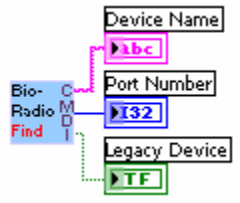


A *Legacy Device* is connected to the PC, using USB, identically as a non-legacy device, but recognized as USB→Serial interface and communicated with through a serial COM port. Currently, the RatPaak is the only BioRadio device with non-legacy support, acting in a "purely" USB manner.

During device discovery, a progress dialog is displayed.

If the Device Name and Type (Legacy or Non), and the Port Number to which it is connected are already known, it is unnecessary to use the Find and Choose Receiver VI, as these parameters can be supplied manually.

Usage



Outputs

Name	Type	Description
Device Name	String	Type of device connected (or chosen by user): "`BioRadio 150`", "`RatPaak`", or "" (if none connected or chosen)
Port Number	Long Integer	port number to which the (chosen) device is connected; corresponds to serial COM port number for legacy devices, and USB port identifier for non-legacy devices. -1 if none is found or chosen
Legacy Device	Boolean	True if BioRadio is acting as legacy USB -> Serial device; *see **Overview** in this section for more on Legacy Devices.*

Starting Base Communication

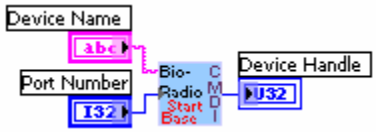
`BioRadio_StartBaseComm.vi`

Overview

Operating with knowledge of the type of BioRadio connected and on which port, communication with the Computer Unit is initiated.

If the device is a BioRadio 150, the **BioRadio150 DLL** is used to perform the following:
A BioRadio object is created (`CreateBioRadio`)
Communication with the Computer Unit is initiated (`StartBaseComm`)
If the device is a RatPaak, no action is taken.
For the RatPaak, this VI's functionality is assumed when acquisition is started, and it need not be used.

Usage



Inputs

Name	Type	Description	Required?
Device Name	String	Device: "`BioRadio 150`", or "`RatPaak`"	Yes
Port Number	Long Integer	Identifying number of the port to which the device is connected (COM Port if *Legacy Device*, USB identifier if non-Legacy)	Yes

Outputs

Name	Type	Description
Device Handle	Unsigned Long Integer	If the device is a BioRadio 150, the handle reference to the device in use. Otherwise, `-1` is returned. See **Multiple BioRadio 150s and the Object Handle**, above.

Starting Acquisition

`BioRadio_Start.vi`

Overview

Operating with knowledge of the type of BioRadio connected and its handle (BioRadio 150) or USB ID (RatPaak), communication with the User Unit is attempted and data acquisition intiated.

If the device is a BioRadio 150, the **BioRadio150 DLL** is used to perform the following:

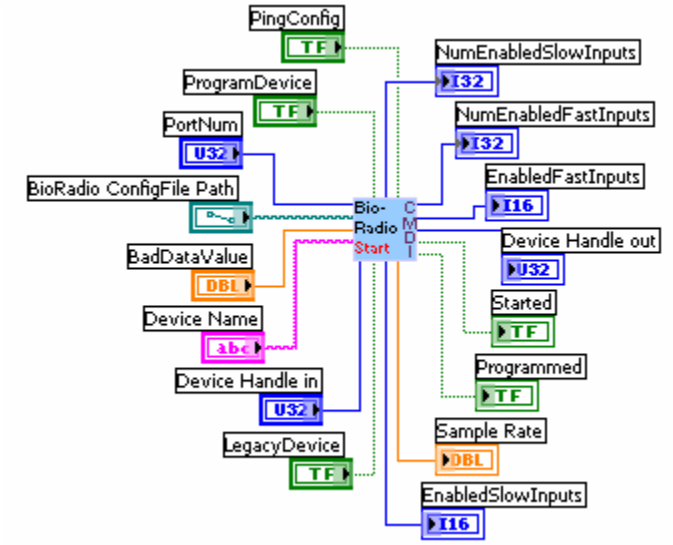
Data acquisition is initiated (`StartAcq`)
The data value returned when packets are dropped is set (`SetBadDataValue`)
Device configuration is optionally set or acquired (`ProgramConfig`, `PingConfig`)
Meta-data is acquired (`GetNumEnabledFastInputs`, `GetEnabledFastInputs`, `GetNumEnabledSlowInputs`, `GetEnabledSlowInputs`, `GetSampleRate`)

If the device is a RatPaak, the **RatPaak DLL** is used to perform the following:

A RatPaak object is created (`CreateRatPaak`)
The supplied configuration file is loaded (`LoadConfig`)
The value returned on dropped packets is set (`SetBadDataValue`)
Data acquisition is initiated (`StartAcq`)
Meta-data is gathered (`GetSampleRate`)
If the device was properly started, it is optionally programmed (`ProgramConfig`)

During Start, Programming, and Pinging, progress dialogs are displayed.

Usage



Inputs

Name	Type	Description	Required?
Device Handle in	Unsigned Long Integer	Handle reference to the BioRadio 150 in use. See **Multiple BioRadio 150s and the**	When using BioRadio 150

		Object Handle, above.	
PortNum	Long Integer	Identifying number of the port to which the device is connected (COM Port if *Legacy Device*, USB identifier if non-Legacy)	When using RatPaak
DeviceName	String	Device: "`BioRadio 150`", or "`RatPaak`"	Yes
ProgramDevice	Boolean	Whether the device should be programmed to the provided configuration file upon starting communications	Yes
BioRadio ConfigFile Path	Path	LabVIEW path variable to the configuration file to which the device should be programmed, if it should be programmed	If ProgramDevice, or when using RatPaak
BadDataValue	Double-Prec. Floating-Point	Value to which invalid/missing data should be set (default: `0`)	No
PingConfig	Boolean	Whether the device should be pinged and its current configuration loaded into the object. If ProgramDevice is set to True, PingConfig will be ignored.	Yes
LegacyDevice	Boolean	True if BioRadio has USB -> Serial interface; *see* Finding And Choosing Attached Devices.	When using RatPaak

Outputs

Name	**Type**	**Description**
NumEnabledFastInputs	Long Integer	Number of fast inputs on which data will be acquired, based on device configuration
NumEnabledSlowInputs	Long Integer	Number of slow inputs on which data will be acquired, based on device configuration
EnabledFastInputs	Word	Boolean-bit array, 10 bits of which (starting from least significtant) corresponding to enabled state of a fast input (starting from first fast input, Ch 1)
EnabledSlowInputs	Word	Boolean-bit array, 5 bits of which (starting from least significtant) corresponding to enabled state of a slow input (starting from first slow input, Accelerometer X)
Device Handle out	Unsigned Long Integer	Handle reference to the device in use. Otherwise, `-1` is returned. See **Multiple BioRadio 150s and the Object Handle**, above.
Started	Boolean	Whether the device was successfully started
Programmed	Boolean	Whether the device was successfully programmed *or* pinged.
Sample Rate	Double-Prec. Floating-Point	Samples per second (for fast inputs) defined by currently loaded configuration

Acquiring Data

`BioRadio_Read.vi`

Overview

Once acquisition has been started, the user will want to begin (and repeat) acquiring data received at the PC's communications port.

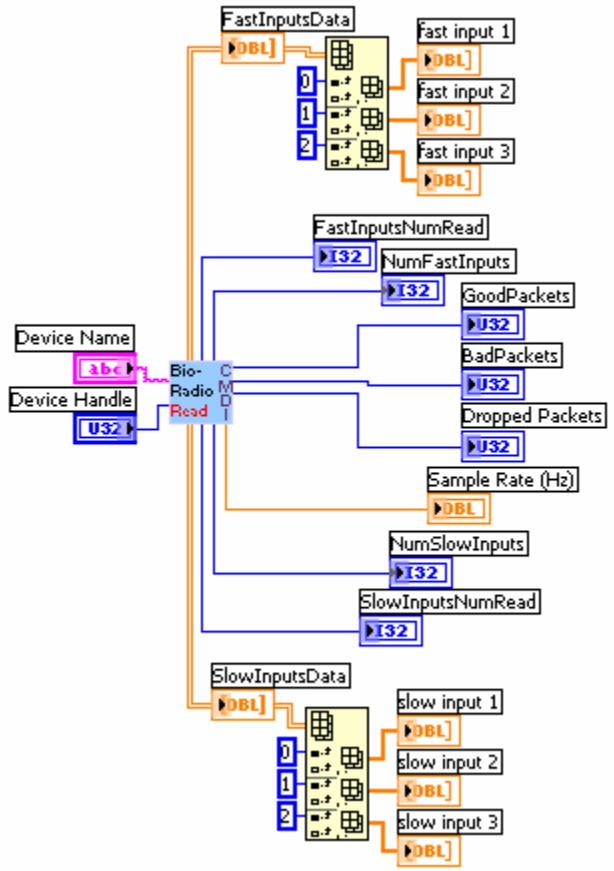
If the device is a BioRadio 150, the **BioRadio150 DLL** is used to perform the following:

1. Instruct the device to prepare its data buffer for reading (`TransferBuffer`)
2. Acquire the sample rate and number of inputs on which the data is being read (`GetSampleRate`, `GetNumEnabledFastInputs`, `GetNumEnabledSlowInputs`)
3. Establish array(s) (fast inputs array and optional slow inputs array) of double-precision floating-point values, (fast inputs array sized to the return value of `TransferBuffer`,) and initialized to `-32768`.
4. Read the data from the device's buffer into this array (`ReadScaledData/ReadScaledFastAndSlowData`)
5. Multiply each data point by `1,000,000` to scale to uV (millivolts)
6. De-interleave data into 2-dimensional array(s) whose rows correspond to input channels
7. Acquire link-status data (`GetGoodPackets`, `GetBadPackets`, `GetDroppedPackets`)

If the device is a RatPaak, the **RatPaak DLL** is used to perform the following:

1. Instruct the device to prepare its data buffer for reading (`TransferBuffer`)
2. Acquire the sample rate and number of inputs on which the data is being read (`GetSampleRate`, `GetNumChannels`)
3. Establish a 655360-element array of double-precision floating-point values, initialized to `-32768`.
4. Read the data from the device's buffer into this array (`ReadScaled`)
5. Multiply each data point by `1,000,000` to scale to uV (millivolts)
6. De-interleave data into a 2-dimensional array whose rows correspond to input channels
7. Acquire link-status data (`GetGoodPackets`, `GetBadPackets`, `GetDroppedPackets`)

Usage



(Note that the number of arrays into which the Fast/SlowInputsData arrays will be de-interleaved is dictated by the number of enabled inputs. Three are shown here for each.)

Inputs

Name	Type	Description	Required?
Device Name	String	Device: "`BioRadio 150`", or "`RatPaak`"	Yes
Device Handle	Unsigned Long Integer	Handle reference to the device in use.	Yes

Outputs

Name	Type	Description
FastInputsData	2d array of Double-Prec. Floating-Point	Data acquired on Fast Inputs; each row of the array representing an input channel, each column with one data point.
SlowInputsData	2d array of Double-Prec. Floating-Point	Data acquired on Slow Inputs (BioRadio 150 only); each row of the array representing an input channel, each column with one data point.
FastInputsNumRead	Long Integer	Number of data points read, over all fast inputs
SlowInputsNumRead	Long Integer	Number of data points read, over all slow inputs (BioRadio 150 only)
NumFastInputs	Long Integer	Number of enabled Fast Inputs

NumSlowInputs	Long Integer	Number of enabled Slow Inputs (BioRadio 150 only)
GoodPackets	Unsigned Long Integer	Number of valid packets transferred since Start
BadPackets	Unsigned Long Integer	Number of corrupted packets transferred since Start
DroppedPackets	Unsigned Long Integer	Number of dropped packets transferred since Start
Sample Rate	Double-Prec. Floating-Point	Samples per second (for fast inputs) defined by currently loaded configuration

Stopping Acquisition

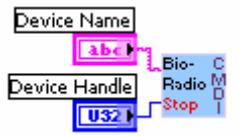
`BioRadio_Stop.vi`

Overview

Stop acquisition. For the RatPaak, additionally stop communication with Computer Unit, and release reserved memory.

If the device is a BioRadio 150, the **BioRadio150 DLL** is used to perform the following:
Stop acquisition (`StopAcq`)
If the device is a RatPaak, the **RatPaak DLL** is used to perform the following:
Stop BioRadio communication (StopAcq)
Destroy the software device object (DestroyRatPaak)

Usage



Inputs

Name	Type	Description	Required?
Device Name	String	Device: "`BioRadio 150`", or "`RatPaak`"	Yes
Device Handle	Unsigned Long Integer	Handle reference to the device in use.	Yes

Stopping Base Communication

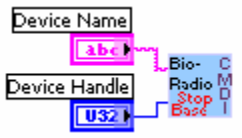
`BioRadio_StopBaseComm.vi`

Overview

For the BioRadio 150, at the end of a session, communication with the Computer Unit should be terminated and reserved memory released.

If the device is a BioRadio 150, the **BioRadio150 DLL** is used to perform the following:
Communication with the Computer Unit is ended (`StopBaseComm`)
Destroy the software device object (`DestroyBioRadio`)
If the device is a RatPaak, no action is taken.
For the RatPaak, this VI's functionality is assumed when acquisition is stopped, and it need not be used.

Usage



Inputs

Name	Type	Description	Required?
Device Name	String	Device: "`BioRadio 150`", or "`RatPaak`"	Yes
Device Handle	Unsigned Long Integer	Handle reference to the device in use.	Yes