Python | 1046_0 | 102 | VoltageRatioInput API

Phidget22.Devices.VoltageRatioInput extends Phidget

The Voltage Ratio Input class is used for measuring the ratio between the voltage supplied to and the voltage returned from an attached sensor or device. This is useful for interfacing with ratiometric sensors or wheatstone bridge based sensors.

For ratiometric sensors, this class supports conversion to sensor data with units specific to the Phidget sensor being used, to make reading these sensors easy.

Methods:

getBridgeEnabled()

Default: ***true***

getBridgeEnabled()

Enable power to the input and start collecting data by setting `BridgeEnabled` to true.

Returns:

The enabled value (type: `boolean`)

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

bridgeEnabled = ch.getBridgeEnabled()
print("BridgeEnabled: " + str(bridgeEnabled))

ch.close()
```

setBridgeEnabled()

setBridgeEnabled(**bridgeEnabled**)

Enable power to the input and start collecting data by setting `BridgeEnabled` to true.

Parameters:

`BridgeEnabled` (type: `boolean`): The enabled value

Returns:

No value is returned

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

ch.setBridgeEnabled(True)

ch.close()
```

---

getBridgeGain()

Default: ***BRIDGE_GAIN_128***

getBridgeGain()

Choose a `BridgeGain` that best suits your application.

- For more information about the range and accuracy of each `BridgeGain` to decide which best suits your application, see your device's User Guide.

Returns:

The bridge gain value (type: `BridgeGain`)

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

bridgeGain = ch.getBridgeGain()
print("BridgeGain: " + str(bridgeGain))

ch.close()
```

---

setBridgeGain()

setBridgeGain(**bridgeGain**)

Choose a `BridgeGain` that best suits your application.

- For more information about the range and accuracy of each `BridgeGain` to decide which best suits your application, see your device's User Guide.

Parameters:

`BridgeGain` (type: `BridgeGain`): The bridge gain value

Returns:

No value is returned

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

ch.setBridgeGain(BridgeGain.BRIDGE_GAIN_128)

ch.close()
```

---

getDataInterval()

*Default: **256** ms*

getDataInterval()

The `DataInterval` is the time that must elapse before the channel will fire another `VoltageRatioChange` event.

- The data interval is bounded by `MinDataInterval` and `MaxDataInterval`.
- The timing between `VoltageRatioChange` events can also be affected by the `VoltageRatioChangeTrigger`.

Returns:

The data interval for the channel (type: `int`)

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

dataInterval = ch.getDataInterval()
print("DataInterval: " + str(dataInterval))

ch.close()
```

setDataInterval()

setDataInterval(**dataInterval**)

The `DataInterval` is the time that must elapse before the channel will fire another `VoltageRatioChange` event.

- The data interval is bounded by `MinDataInterval` and `MaxDataInterval`.
- The timing between `VoltageRatioChange` events can also be affected by the `VoltageRatioChangeTrigger`.

Parameters:

`DataInterval` (type: `int`): The data interval for the channel

Returns:

No value is returned

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

ch.setDataInterval(1000)

ch.close()
```

getMinDataInterval()

*Constant: **8** ms*

getMinDataInterval()

The minimum value that `DataInterval` can be set to.

Returns:

The data interval value (type: `int`)

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

minDataInterval = ch.getMinDataInterval()
print("MinDataInterval: " + str(minDataInterval))

ch.close()
```

getMaxDataInterval()

*Constant: **1,000** ms*

getMaxDataInterval()

The maximum value that `DataInterval` can be set to.

Returns:

The data interval value (type: `int`)

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

maxDataInterval = ch.getMaxDataInterval()
print("MaxDataInterval: " + str(maxDataInterval))

ch.close()
```

getDataRate()

*Default: **3.90625** Hz*

getDataRate()

The `DataRate` is the frequency of events from the device.

- The data rate is bounded by `MinDataRate` and `MaxDataRate`.
- Changing `DataRate` will change the channel's `DataInterval` to a corresponding value, rounded to the nearest integer number of milliseconds.

- The timing between events can also affected by the change trigger.

Returns:

The data rate for the channel (type: `float`)

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

dataRate = ch.getDataRate()
print("DataRate: " + str(dataRate))

ch.close()
```

---

setDataRate()

setDataRate(**dataRate**)

The `DataRate` is the frequency of events from the device.

- The data rate is bounded by `MinDataRate` and `MaxDataRate`.
- Changing `DataRate` will change the channel's `DataInterval` to a corresponding value, rounded to the nearest integer number of milliseconds.
- The timing between events can also affected by the change trigger.

Parameters:

`DataRate` (type: `float`): The data rate for the channel

Returns:

No value is returned

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

ch.setDataRate(1000)

ch.close()
```

getMinDataRate()

*Constant: **1.0** Hz*

getMinDataRate()

The minimum value that `DataRate` can be set to.

Returns:

The data rate value (type: `float`)

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

minDataRate = ch.getMinDataRate()
print("MinDataRate: " + str(minDataRate))

ch.close()
```

getMaxDataRate()

*Constant: **125.0** Hz*

getMaxDataRate()

The maximum value that `DataRate` can be set to.

Returns:

The data rate value (type: `float`)

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

maxDataRate = ch.getMaxDataRate()
print("MaxDataRate: " + str(maxDataRate))

ch.close()
```

getVoltageRatio()

Unit: volts/volt (V/V)

getVoltageRatio()

The most recent voltage ratio value that the channel has reported.

- This value will always be between `MinVoltageRatio` and `MaxVoltageRatio`.

Returns:

The voltage ratio value (type: `float`)

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

voltageRatio = ch.getVoltageRatio()
print("VoltageRatio: " + str(voltageRatio))

ch.close()
```

getMinVoltageRatio()

*Constant: **-1.0** V/V*

getMinVoltageRatio()

The minimum value the `VoltageRatioChange` event will report.

Returns:

The voltage ratio value (type: `float`)

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

minVoltageRatio = ch.getMinVoltageRatio()
print("MinVoltageRatio: " + str(minVoltageRatio))

ch.close()
```

---

getMaxVoltageRatio()

*Constant: **1.0** V/V*

getMaxVoltageRatio()

The maximum value the `VoltageRatioChange` event will report.

Returns:

The voltage ratio value (type: `float`)

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

maxVoltageRatio = ch.getMaxVoltageRatio()
print("MaxVoltageRatio: " + str(maxVoltageRatio))

ch.close()
```

---

getVoltageRatioChangeTrigger()

*Default: **0.0** V/V*

getVoltageRatioChangeTrigger()

The channel will not issue a `VoltageRatioChange` event until the voltage ratio value has changed by the amount specified by the `VoltageRatioChangeTrigger`.

- Setting the `VoltageRatioChangeTrigger` to 0 will result in the channel firing events every `DataInterval`. This is useful for applications that implement their own data filtering

Returns:

The change trigger value (type: `float`)

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

voltageRatioChangeTrigger = ch.getVoltageRatioChangeTrigger()
print("VoltageRatioChangeTrigger: " + str(voltageRatioChangeTrigger))

ch.close()
```

---

setVoltageRatioChangeTrigger()

setVoltageRatioChangeTrigger(**voltageRatioChangeTrigger**)

The channel will not issue a `VoltageRatioChange` event until the voltage ratio value has changed by the amount specified by the `VoltageRatioChangeTrigger`.

- Setting the `VoltageRatioChangeTrigger` to 0 will result in the channel firing events every `DataInterval`. This is useful for applications that implement their own data filtering

Parameters:

`VoltageRatioChangeTrigger` (type: `float`): The change trigger value

Returns:

No value is returned

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

ch.setVoltageRatioChangeTrigger(0)

ch.close()
```

getMinVoltageRatioChangeTrigger()

*Constant: **0.0** V/V*

getMinVoltageRatioChangeTrigger()

The minimum value that `VoltageRatioChangeTrigger` can be set to.

Returns:

The change trigger value (type: `float`)

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

minVoltageRatioChangeTrigger = ch.getMinVoltageRatioChangeTrigger()
print("MinVoltageRatioChangeTrigger: " + str(minVoltageRatioChangeTrigger))

ch.close()
```

getMaxVoltageRatioChangeTrigger()

*Constant: **1.0** V/V*

getMaxVoltageRatioChangeTrigger()

The maximum value that `VoltageRatioChangeTrigger` can be set to.

Returns:

The change trigger value (type: `float`)

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *

ch = VoltageRatioInput()
ch.openWaitForAttachment(1000)

maxVoltageRatioChangeTrigger = ch.getMaxVoltageRatioChangeTrigger()
print("MaxVoltageRatioChangeTrigger: " + str(maxVoltageRatioChangeTrigger))

ch.close()
```

Events:

setOnVoltageRatioChangeHandler()

setOnVoltageRatioChangeHandler(**handler**)

Assigns a handler that will be called when the VoltageRatioChange event occurs

Parameters:

handler (type: `VoltageRatioChangeHandler(self, voltageRatio)`): The callback function

Returns:

No value is returned

Throws:

PhidgetException

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *
import time

def onVoltageRatioChange(self, voltageRatio):
    print("VoltageRatio: " + str(voltageRatio))

ch = VoltageRatioInput()

# Register for event before calling open
ch.setOnVoltageRatioChangeHandler(onVoltageRatioChange)

ch.open()

while True:
    # Do work, wait for events, etc.
    time.sleep(1)
```

VoltageRatioChange

void onVoltageRatioChange(**self, voltageRatio**)

The most recent voltage ratio value the channel has measured will be reported in this event, which occurs when the `DataInterval` has elapsed.

- If a `VoltageRatioChangeTrigger` has been set to a non-zero value, the `VoltageRatioChange` event will not occur until the voltage has changed by at least the `VoltageRatioChangeTrigger` value.

self (type: `VoltageRatioInput`): object which sent the event
voltageRatio (type: `float`): The voltage ratio

Example:

```
from Phidget22.Phidget import *
from Phidget22.Devices.VoltageRatioInput import *
import time

def onVoltageRatioChange(self, voltageRatio):
    print("VoltageRatio: " + str(voltageRatio))

ch = VoltageRatioInput()

# Register for event before calling open
ch.setOnVoltageRatioChangeHandler(onVoltageRatioChange)

ch.open()

while True:
    # Do work, wait for events, etc.
    time.sleep(1)
```

Error Events:

OutOfRange

Error Event code: EEPHIDGET_OUTOFRANGE **(Value: 0x1007) -** *Out of Range*

The most recent voltage ratio the channel has measured is outside the valid range for the channel's current settings.

Enumerations:

Phidget22.BridgeGain

Bridge gain amplification setting. Higher gain results in better resolution, but narrower voltage range.

Name	Value	Description
BRIDGE_GAIN_1	0x1	*1x* 1x Amplificaion
BRIDGE_GAIN_2	0x2	*Unsupported*
BRIDGE_GAIN_4	0x3	*Unsupported*
BRIDGE_GAIN_8	0x4	*8x* 8x Amplification

BRIDGE_GAIN_16	0x5	*16x* 16x Amplification
BRIDGE_GAIN_32	0x6	*32x* 32x Amplification
BRIDGE_GAIN_64	0x7	*64x* 64x Amplification
BRIDGE_GAIN_128	0x8	*128x* 128x Amplification