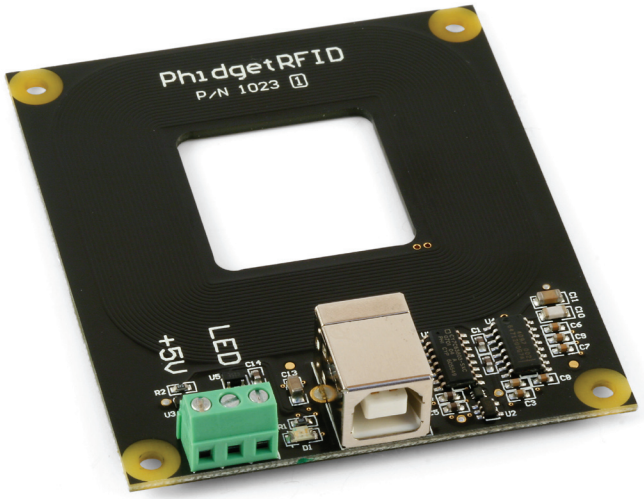
Phidgets

COM API Manual



Phidgets COM API Manual
Version 2.1.8

Last updated: February 22, 2011

Contents

PhidgetGPS

PhidgetInterfaceKit

PhidgetIR

PhidgetLED

PhidgetMotorControl

PhidgetPHSensor

PhidgetRFID

PhidgetServo

Introduction

Overview

This manual describes the Application Programming Interface (API) for each Phidget device, as exposed by the COM library. This API can be used from a variety of languages; this manual focuses on use within VB6.0, and therefore presents the COM interface as a VB6.0 user sees it.

How to use Phidgets

Phidgets are an easy to use set of building blocks for low cost sensing and control from your PC. Using the Universal Serial Bus (USB) as the basis for all Phidgets, the complexity is managed behind this easy to use and robust Application Program Interface (API) library.

This library is written for and available on Windows only.

Languages that make use of the COM API and are supported by Phidgets include: VB6.0, Labview, Delphi, VBA and VBScript. Each of these languages have their own way of exposing functionality, but the base calls will be the same.

Refer to the User Guide for your Phidget and the Software Overview page for more detailed, language unspecific API documentation. The User Guide, along with other resources, can be found on the product page for your device. Also, there are a set of VB6.0 examples available for download.

Please note that if your method contains more than one parameter, do not enclose the parameters inside brackets. For example instead of:

```
Object.getProperty(Parameter1, Parameter2)
```

You should write:

```
Object.getProperty Parameter1, Parameter2
```

Class Hierarchy

- Phidget
  - PhidgetAccelerometer
  - PhidgetAdvancedServo
  - PhidgetEncoder
  - PhidgetInterfaceKit
  - PhidgetLED
  - PhidgetMotorControl
  - PhidgetPHSensor
  - PhidgetRFID
  - PhidgetServo
  - PhidgetStepper
  - PhidgetTemperatureSensor
  - PhidgetTextLCD
  - PhidgetTextLED
  - PhidgetWeightSensor
- PhidgetManager
- PhidgetDictionary
- PhidgetKeyListener

Phidget

Class documentation for Phidget. This is the base class from which all other device classes inherit. These calls are common to all Phidgets objects. See the General Phidget Programming guide for more in-depth usage instructions and examples.

Structures

PhidgetCOM_Error

The Phidget error codes. `EnableVerboseErrors` is enabled, every call in the library will return one of these codes on error, or S_OK on success.

The E_(error) codes are used for regular function calls, and represent the scode part of an HRESULT. The EE_(error) codes are used for the OnError event, and don't depend on `EnableVerboseErrors`. These errors are defined in the General Phidget Programming page.

```
Enum PhidgetCOM_Error {
      E_PHIDGETCOM_OK,
      E_PHIDGETCOM_NOTFOUND,
      E_PHIDGETCOM_NOMEMORY,
      E_PHIDGETCOM_UNEXPECTED,
      E_PHIDGETCOM_INVALIDARG,
      E_PHIDGETCOM_NOTATTACHED,
      E_PHIDGETCOM_INTERRUPTED,
      E_PHIDGETCOM_INVALID,
      E_PHIDGETCOM_NETWORK,
      E_PHIDGETCOM_UNKNOWNVAL,
      E_PHIDGETCOM_BADPASSWORD,
      E_PHIDGETCOM_UNSUPPORTED,
      E_PHIDGETCOM_DUPLICATE,
      E_PHIDGETCOM_TIMEOUT,
      E_PHIDGETCOM_OUTOFBOUNDS,
      E_PHIDGETCOM_EVENT,
      E_PHIDGETCOM_NETWORK_NOTCONNECTED,
      E_PHIDGETCOM_WRONGDEVICE,
      E_PHIDGETCOM_CLOSED,
      E_PHIDGETCOM_BADVERSION,
      //Start of Error Event codes
      EE_PHIDGETCOM_NETWORK,
      EE_PHIDGETCOM_BADPASSWORD,
      EE_PHIDGETCOM_BADVERSION,
      EE_PHIDGETCOM_OVERRUN,
      EE_PHIDGETCOM_PACKETLOST,
      EE_PHIDGETCOM_WRAP,
      EE_PHIDGETCOM_OVERTEMP,
      EE_PHIDGETCOM_OVERCURRENT,
      EE_PHIDGETCOM_OUTOFRANGE
};
```

Functions

Open

Opens a phidget.

```
Open(
   SerialNumber as Long [optional]
);
```

Parameters:

SerialNumber [optional]
Serial number of the Phidget to open. Do not specify to open any.

OpenRemote

Opens a Phidget remotely using a server id.

```
OpenRemote(
   ServerID as String [optional]
   SerialNumber as Long [optional]
   Password as String [optional]
);
```

Parameters:

ServerID
Server ID of the webservice to connect to. Not not specify to connect to any.

SerialNumber
Serial number of the Phidget to open. Do not specify to open any.

Password
Password of the webservice. Do not specify if the webservice does not have a password.

OpenRemoteIP

Opens a Phidget remotely using an address and port.

```
OpenRemoteIP(
   IPAddress as String
   Port as Long
   SerialNumber as Long [optional]
   Password as String [optional]
);
```

Parameters:

IPAddress
The address of the webservice to connect to.

Port
The port of the webservice to connect to.

SerialNumber
Serial number of the Phidget to open. Do not specify to open any.

Password
Password of the webservice. Do not specify if the webservice does not have a password.

Close

Closes a Phidget.

```
Close();
```

WaitForAttachment

Blocks until the Phidget has attached.

```
WaitForAttachment(
   milliseconds as Long
);
```

Parameters:

milliseconds
The number of milliseconds to wait for an attachment. Specify 0 to wait forever.

EnableLogging

Enables logging in the C library. This is for debugging purposes.

```
EnableLogging(
   level as Long,
   file as String
);
```

Parameters:

level
The highest level of logs to report. This can be any integer from 1 to 6. See the constants section below for more information.

file
The file to output logs to. Specify NULL to send logs to the console.

Constants

There are 6 levels of logging. Each higher level will include all lower levels when outputting logs. These are represented either as a set of constants or an enumerator, depending on language.

PHIDGET_LOG_CRITICAL = 1

Critical error messages.

This is the lowest logging level. Errors at this level are generally non-recoverable and indicate either hardware problems, library bugs, or other serious issues.

PHIDGET_LOG_ERROR = 2

Non-critical error messages.

Errors at this level are generally automatically recoverable, but may help to track down issues.

PHIDGET_LOG_WARNING = 3

Warning messages.

Warnings are used to log behavior that is not necessarily in error, but is nevertheless odd or unexpected.

PHIDGET_LOG_DEBUG = 4

Debug messages.

Debug messages are generally used for debugging at Phidgets Inc.

Note: **PHIDGET_LOG_DEBUG** messages are only logged in the debug version of the library, regardless of logging level. Thus, these logs should never be seen outside of Phidgets Inc.

PHIDGET_LOG_INFO = 5

Informational messages.

Informational messages track key happenings within phidget21 - mostly to do with threads starting and shutting down, and the internal USB code.

PHIDGET_LOG_VERBOSE = 6

Verbose messages.

This is the highest logging level. Verbose messages are informational messages that are expected to happen so frequently that they tend to drown out other log messages.

DisableLogging

Disabled logging in the C library.

```
DisableLogging();
```

Log

Sends a log message to the log. Make sure to enable logging first.

```
Log(
   level as Long,
   ident as String,
   log as String
);
```

Parameters:

level
The level to log at. There are 6 levels at 1-6.

ident
A user defined string to identify the log. This can be blank.

log
The message to log.

EnableVerboseErrors

Enables stricter error handling. This changes the behavior of the library to more closely match the C API, and .NET. By default, most common errors are sent out via the OnError event, instead of being raised directly by the offending Function call - this makes error handling unnecessary. This functionality has been added for users who require stricter error handling.

When verbose error handling is active, the errors will be one of the codes in `PhidgetCOM_Error`. When inactive, any errors that are raised will be standard COM errors such as E_FAIL.

```
EnableLogging(
   state as Boolean
);
```

Parameters:

state
True to enable, False to disable.

Properties

IsAttached

Gets the attached status of a Phidget.

```
IsAttached as Boolean [get]
```

DeviceType

Gets the device type of a Phidget.

```
DeviceType as String [get]
```

DeviceVersion

Gets the firmware version of a Phidget.

```
DeviceVersion as Long [get]
```

Name

Gets the long name of a Phidget.

```
Name as String [get]
```

SerialNumber

Gets the unique serial number of a Phidget.

```
SerialNumber as Long [get]
```

Label

Gets / Sets the Label of a Phidget. Note that setting the label is not yet supported on Windows.

```
Label as String [get,set]
```

IsAttachedToServer

Gets the attached to server state of a remotely opened Phidget.

```
IsAttachedToServer as Boolean [get]
```

Address

Gets the webservice address of a remotely opened Phidget.

```
Address as String [get]
```

Port

Gets the webservice port number of a remotely opened Phidget.

```
Port as Long [get]
```

ServerID

Gets the webservice Server ID of a remotely opened Phidget.

```
ServerID as String [get]
```

LibraryVersion

Gets the phidget library version. This returns both the C library and COM library versions as a multi-line string.

```
LibraryVersion as String [get]
```

Events

Note that these events are actually members of each Phidget device class rather then the base class. However, since they are common to all Phidgets, they are documented here.

OnAttach

Fired when a Phidget is plugged in and ready to use.

```
event OnAttach
```

OnDetach

Fired when a Phidget is unplugged.

```
event OnDetach
```

OnError

Fired on an asynchronous error. These are mostly network related.

```
event OnError(
   Description as  String,
   SCODE as Long
)
```

Parameters:

Description

A description of the error.

SCODE

An error code corresponding to the error. See the General Phidget Programming guide for a list of error codes.

OnServerConnect

Fired when a connection to the webservice is made, when opening a Phidget remotely.

```
event OnServerConnect
```

OnServerDisconnect

Fired when a connection to the webservice is lost, when opening a Phidget remotely.

```
event OnServerDisconnect
```

PhidgetAccelerometer

Class documentation for PhidgetAccelerometer. This class contains all calls specific to the Phidget Accelerometer. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Properties

AxisCount

Gets the number of acceleration axes supported by this board.

```
AxisCount as Long [get]
```

Acceleration

Gets the current acceleration of a axis.

```
Acceleration(
   Index as Long
) as Double [get]
```

Parameters:

Index
The acceleration axis.

AccelerationChangeTrigger

Gets / Sets the change trigger for an axis.

```
AccelerationChangeTrigger(
   Index as Long
) as Double [get,set]
```

Parameters:

Index
The acceleration axis.

AccelerationMax

Gets the maximum acceleration that can be measured by as axis.

```
AccelerationMax(
   Index as Long
) as Double [get]
```

Parameters:

Index
The acceleration axis.

AccelerationMin

Gets the minimum acceleration that can be measured by an axis.

```
AccelerationMin(
   Index as Long
) as Double [get]
```

Parameters:

Index
The acceleration axis.

Events

OnAccelerationChange

Fired when the acceleration on an axis changes by more then the change trigger.

```
event OnAccelerationChange(
   Index as Long,
   Acceleration as Double
)
```

Parameters:

Index
The acceleration axis.

Acceleration
The acceleration.

PhidgetAdvancedServo

Class documentation for PhidgetAdvancedServo. This class contains all calls specific to the Phidget Advanced Servo. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Structures

PhidgetCOM_ServoType

Used for the `ServoType` property. These are the predefined servo types supported by Phidgets Inc. This list may be incomplete.

```
enum PhidgetCOM_ServoType {
   PHIDGETCOM_SERVO_DEFAULT = 1,
   PHIDGETCOM_SERVO_RAW_us_MODE,
   PHIDGETCOM_SERVO_HITEC_HS322HD,
   PHIDGETCOM_SERVO_HITEC_HS5245MG,
   PHIDGETCOM_SERVO_HITEC_805BB,
   PHIDGETCOM_SERVO_HITEC_HS422,
   PHIDGETCOM_SERVO_TOWERPRO_MG90,
   ...,
   PHIDGETCOM_SERVO_USER_DEFINED
}
```

Functions

setServoParameters

Sets parameters for a custom servo type. This includes PCM range, degrees of rotation and maximum velocity. This affect min and max position, and the PCM to degree mapping formulas.

```
setServoParameters(
   Index as Long,
   MinUs as double,
   MaxUs as double,
   Degrees as double,
   VelocityMax as double
);
```

Parameters:

Index

The motor index.

MinUs

The minimum PCM supported by the motor, in microseconds.

MaxUs

The maximum PCM supported by the motor, in microseconds.

Degrees

Real degrees of rotation represented by the given PCM range

VelocityMax

Maximum velocity supported by the servo, in degrees/second.

Properties

MotorCount

Gets the number of motors supported by this controller.

```
MotorCount as Long [get]
```

Acceleration

Gets / Sets the acceleration for a motor.

```
Acceleration(
   Index as Long
) as Double [get,set]
```

Parameters:

Index

The motor index.

AccelerationMax

Gets the maximum acceleration supported by a motor.

```
AccelerationMax(
   Index as Long
) as Double [get]
```

Parameters:

Index

The motor index.

AccelerationMin

Gets the minimum acceleration supported by a motor.

```
AccelerationMin(
   Index as Long
) as Double [get]
```

Parameters:

Index

The motor index.

Current

Gets the current current draw of a motor.

```
Current(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

Position

Gets / Sets the current / target position of a motor.

```
Position(
   Index as Long
) as Double [get,set]
```

Parameters:

Index
The motor index.

PositionMax

Gets / Sets the maximum position supported by a motor.

```
PositionMax(
   Index and Long
) as Double [get,set]
```

Parameters:

Index
The motor index.

PositionMin

Gets / Sets the minimum position supported by a motor.

```
PositionMin(
   Index as Long
) as Double [get,set]
```

Parameters:

Index
The motor index.

Velocity

Gets the current velocity of a motor.

```
Velocity(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

VelocityLimit

Gets / Set the velocity limit of a motor.

```
VelocityLimit(
   Index as Long
) as Double [get,set]
```

Parameters:

Index
The motor index.

VelocityMax

Gets the maximum velocity limit supported by a motor.

```
VelocityMax(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

VelocityMin

Gets the minimum velocity limit supported by a motor.

```
VelocityMin(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

Engaged

Gets / Sets the engaged state of a motor. This is whether a motor is powered or not.

```
Engaged(
   Index as Long
) as Boolean [get,set]
```

Parameters:

Index
The motor index.

SpeedRampingOn

Gets / Sets the speed ramping state of a motor. This is whether or not the motor uses velocity and acceleration to move.

```
SpeedRampingOn(
   Index as Long
) as Boolean [get,set]
```

Parameters:

Index
The motor index.

Stopped

Gets the stopped state of a motor. If this is true, the motor is not moving and there are no outstanding commands.

```
Stopped(
   Index as Long
) as Boolean [get]
```

Parameters:

Index
The motor index.

ServoType

Gets / Sets the servo type for an index. There are several predefined servo types. All other types of servos can be set up using the `setServoParameters` function.

```
ServoType(
   Index as Long
) as PhidgetCOM_ServoType [get,set]
```

Parameters:

Index
The motor index.

Events

OnCurrentChange

Fired when the current draw of a motor changes.

```
event OnCurrentChange(
    Index as Long,
    Current as Double
)
```

Parameters:

Index
The motor index.

Current
The current draw.

OnPositionChange

Fired when the position of a motor changes.

```
event OnPositionChange(
    Index as Long,
    Position as Double
)
```

Parameters:

Index
The motor index.

Position
The motor position.

OnVelocityChange

Fired when the velocity of a motor changes.

```
event OnVelocityChange(
    Index as Long,
    Velocity as Double
)
```

Parameters:

Index
The motor index.

Velocity
The current velocity.

PhidgetAnalog

Class documentation for PhidgetAnalog. This class contains all calls specific to the Phidget Analog. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Properties

OutputCount

Gets the number of analog outputs supported by the PhidgetAnalog.

```
OutputCount as Long [get]
```

Voltage

Gets / Sets the current voltage setting for an analog output, in Volts. The range is VoltageMin-VoltageMax.

```
Voltage(
   Index as long
) as Double [get,set]
```

Parameters:

Index
The analog output index.

VoltageMin

Gets the minimum supported voltage for an analog output, in Volts.

```
VoltageMin(
   Index as long
) as Double [get]
```

Parameters:

Index
The analog output index.

VoltageMax

Gets the maximum supported voltage for a analog output, in Volts.

```
VoltageMax(
   Index as long
) as Double [get]
```

Parameters:

Index

The analog output index.

Enabled

Gets / Sets the enabled state (power) for an analog output.

```
Enabled(
   Index as Long
) as Boolean [get,set]
```

Parameters:

Index

The analog output index.

Events

OnError

The PhidgetAnalog will throw error events under certain circumstances:

SCODE = EEPHIDGET_OVERCURRENT - An Overcurrent condition has occured on an output. Under this condition, the output is clamped to 20mA.

SCODE = EEPHIDGET_OVERTEMP - A Thermal Shutdown state has occured. The outputs will be shut down under this condition.

When the overcurrent or overtemperature state have ended, there will be an error event with the EEPHIDGET_OK code.

See the ErrorDescription String for specific error details.

```
event OnError(
   Description as String,
   SCODE as LONG
)
```

Parameters:

Description

A description of the error.

SCODE

An error code corresponding to the error.

PhidgetBridge

Class documentation for PhidgetBridge. This class contains all calls specific to the Phidget Bridge. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Structures

PhidgetCOM_BridgeGain

Used for the `Gain` property. These are the predefined gains supported by the Phidget Bridge.

```
enum PhidgetCOM_BridgeGain
{
   PHIDGETCOM_BRIDGE_GAIN_1 = 1,
   PHIDGETCOM_BRIDGE_GAIN_8,
   PHIDGETCOM_BRIDGE_GAIN_16,
   PHIDGETCOM_BRIDGE_GAIN_32,
   PHIDGETCOM_BRIDGE_GAIN_64,
   PHIDGETCOM_BRIDGE_GAIN_128,
   PHIDGETCOM_BRIDGE_GAIN_UNKNOWN
}
```

Properties

InputCount

Gets the number of bridges supported by the PhidgetBridge.

```
InputCount as Long [get]
```

BridgeValue

Gets the value of the selected bridge, in mV/V.

```
BridgeValue(
   Index as long
) as Double [get]
```

Parameters:

Index
The bridge index.

BridgeMin

Gets the minimum value that the selected bridge can measure, in mV/V. This value will depend on the selected gain.

```
BridgeMin(
   Index as long
) as Double [get]
```

Parameters:

Index
The bridge index.

BridgeMax

Gets the maximum value that the selected bridge can measure, in mV/V. This value will depend on the selected gain.

```
BridgeMax(
   Index as long
) as Double [get]
```

Parameters:

Index
The bridge index.

Enabled

Gets / Sets the enabled state (power) of a bridge.

```
Enabled(
   Index as Long
) as Boolean [get,set]
```

Parameters:

Index
The bridge index.

Gain

Gets / Sets the gain for a selected bridge.

```
Gain(
   Index as Long
) as PhidgetCOM_BridgeGain [get,set]
```

Parameters:

Index
The bridge index.

DataRate

Gets / Sets the data rate, in ms. Data rate applies to all bridges simultaneously.

```
DataRate as Long [get,set]
```

DataRateMin

Gets the minimum supported data rate, in ms.

```
DataRateMin as Long [get]
```

DataRateMax

Gets the maximum supported data rate, in ms

```
DataRateMax as Long [get]
```

Events

OnBridgeData

Fired at the specified DataRate, for each enabled bridge. Value is the BridgeValue, in mV/V.

```
event OnBridgeData(
   Index as Long,
   Value as Double
)
```

Parameters:

Index
The bridge index.

Value
The bridge value of the selected bridge, in mV/V.

OnError

The PhidgetBridge will throw error events under certain circumstances:

SCODE = EEPHIDGET_OUTOFRANGE - A bridge input has gone out of range. This indicates either an overrange or underrange condition. If possible, gain should be reduced.

See the ErrorDescription String for specific error details.

```
event OnError(
   Description as String,
   SCODE as LONG
)
```

Parameters:

Description
A description of the error.

SCODE
An error code corresponding to the error.

PhidgetEncoder

Class documentation for PhidgetEncoder. This class contains all calls specific to the Phidget Encoder. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Properties

EncoderCount

Gets the number of encoder inputs supported by this board.

```
EncoderCount as Long [get]
```

Position

Gets / Sets the current position of an encoder.

```
Position(
   Index as long
) as Long [get,set]
```

Parameters:

Index
The encoder index.

IndexPosition

Gets the position of the last index pulse. Not supported by all encoders.

```
IndexPosition(
   Index as long
) as Long [get]
```

Parameters:

Index
The encoder index.

Enabled

Gets / Sets the enabled state (power) of an encoder. Not supported by all PhidgetEncoders.

```
Enabled(
   Index as Long
) as Boolean [get,set]
```

Parameters:

Index
The encoder index.

InputCount

Gets the number of digital inputs supported by this board.

```
InputCount as Long [get]
```

InputState

Gets the state of a digital input.

```
InputState(
   Index as Long
) as Boolean [get]
```

Parameters:

Index
The digital input index.

Events

OnInputChange

Fired when a digital input changes.

```
event OnInputChange(
   Index as Long,
   NewState as Boolean
)
```

Parameters:

Index
The digital input index.

NewState
The state of the input

OnPositionChange

Fired when an encoder position changed.

```
event OnPositionChange(
   Index as Long,
   Time as Long,
   EncoderDisplacement as Long
)
```

Parameters:

Index
The encoder index

Time
The time since the last position change event.

EncoderDisplacement
The amount the position changed since the last position change event.

PhidgetFrequencyCounter

Class documentation for PhidgetFrequencyCounter. This class contains all calls specific to the Phidget Frequency Counter. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Structures

PhidgetCOM_FrequencyCounterFilterType

Used for the `Filter` property. These are the predefined filter types supported by the Phidget Frequency Counter.

```
enum PhidgetCOM_FrequencyCounterFilterType {
   PHIDGETCOM_FREQUENCYCOUNTER_FILTERTYPE_ZERO_CROSSING = 1,
   PHIDGETCOM_FREQUENCYCOUNTER_FILTERTYPE_LOGIC_LEVEL,
   PHIDGETCOM_FREQUENCYCOUNTER_FILTERTYPE_UNKNOWN
}
```

Functions

Reset

Resets the TotalCount and TotalTime counters to 0 for the specified channel. For best performance, this should be called when the channel is disabled.

```
Reset(
   Index as Long,
}
```

Parameters:

Index
The channel index.

Properties

FrequencyInputCount

Gets the number of channels supported by the PhidgetFrequencyCounter.

```
FrequencyInputCount as Long [get]
```

Frequency

Gets the last calculated frequency on the specified channel, in Hz. Returns 0 if the timeout value elapses without detecting a signal.

```
Frequency(
   Index as long
```

```
) as Double [get]
```

Parameters:

Index
The channel index.

TotalCount

Gets the total number of pulses detected on the specified channel since the Phidget was opened, or since the last reset.

```
TotalCount(
   Index as long
) as Long [get]
```

Parameters:

Index
The channel index.

TotalTime

Gets the total elapses time since Phidget was opened, or since the last reset, in microseconds. This time corresponds to the TotalCount property.

```
TotalTime(
   Index as long
) as Long [get]
```

Parameters:

Index
The channel index.

Timeout

Gets / Sets the timeout value of the specified channel, in microseconds. This value is used to set the time to wait without detecting a signal before reporting 0 Hz. The valid range is 0.1 - 100 seconds(100,000 - 100,000,000 microseconds).

```
Timeout(
   Index as long
) as Long [get,set]
```

Parameters:

Index
The channel index.

Enabled

Gets / Sets the enabled state (power) of a specified channel. When the channel is disabled, it will no longer register counts. TotalTime and TotalCount properties will not be incremented until the channel is re-enabled.

```
Enabled(
    Index as Long
) as Boolean [get,set]
```

Parameters:

Index
The channel index.

Filter

Gets / Sets the channel filter mode of a specified channel.

```
Filter(
    Index as Long
) as PhidgetCOM_FrequencyCounterFilterType [get,set]
```

Parameters:

Index
The channel index.

Events

OnCount

Fired whenever some counts have been detected. This event will fire at up to 31.25 times a second, depending on the pulse rate. The time is in microseconds and represents the amount of time in which the number of counts occured. This event will fire with a count of 0 once, after the Timeout time has elapsed with no counts for a channel, to indicate 0 Hz the specified DataRate, for each enabled bridge. Value is the BridgeValue, in mV/V.

```
event OnCount(
    Index as Long,
    Time as Long,
    Counts as Long
)
```

Parameters:

Index
The channel index.

Time
The amount of time in which the number of counts occured, in microseconds.

Counts
The number of pulses detected on the specified channel since the last event.

PhidgetGPS

Class documentation for PhidgetGPS. This class contains all calls specific to the Phidget GPS. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Due to the nature of COM, executables may only work on the computer it is built on.

Structures

PhidgetGPS_Date

Used for the `Date` property. This structure represents the current GPS time.

```
struct PhidgetGPS_Date {
   short tm_mday;
   short tm_mon;
   short tm_year;
}
```

PhidgetGPS_Time

Used for the `Time` property. This structure represents the current GPS date.

```
struct PhidgetGPS_Time {
   short tm_ms;
   short tm_sec;
   short tm_min;
   short tm_hour;
}
```

Properties

Latitude

Gets the current latitude, in signed degrees format.

```
Latitude as Double [get]
```

Longitude

Gets the current longitude, in signed degrees format.

```
Longitude as Double [get]
```

Altitude

Gets the current altitude, in meters.

```
Altitude as Double [get]
```

Heading

Gets the current heading, in degrees - in compass bearing format. Heading is only accurate if the GPS is moving, and it represents a heading over time, and not the actual direction of the PhidgetGPS is pointing.

```
Heading as Double [get]
```

Velocity

Gets the current velocity, in km/h. Velocity is only accurate if the PhidgetGPS is moving.

```
Velocity as Double [get]
```

Date

Gets the current GPS date.

```
Date as PhidgetGPS_Date [get]
```

Time

Gets the current GPS time. The time is updated 10 times a second and is accurate to within at least 500ms when `PositionFixStatus` is true.

```
Time as PhidgetGPS_Time [get]
```

PositionFixStatus

Gets the current position fix status. If true, all of the above properties will be available. Time and Date may or may not be available, but they can only be trusted as accurate when the `PositionFixStatus` is true.

```
PositionFixStatus as Boolean [get]
```

Events

OnPositionChange

Fired whenever the position changes.

```
event OnPositionChange(
   Latitude as Double,
   Longitude as Double,
   Altitude as Double
)
```

Parameters:

Latitude

The current latitude, in signed degrees format.

Longitude
The current longitude, in signed degrees format.

Altitude
The current altitude, in meters.

OnPositionFixStatusChange

Fired whenever the position changes.

```
event OnPositionFixStatusChange(
   fixStatus as Boolean
)
```

Parameters:

fixStatus
The current position fix status.

PhidgetInterfaceKit

Class documentation for PhidgetInterfaceKit. This class contains all calls specific to the Phidget Interface Kit. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Properties

InputCount

Gets the number of digital inputs supported by this board.

```
InputCount as Long [get]
```

InputState

Gets the state of a digital input.

```
InputState(
   Index as Long
) as Boolean [get]
```

Parameters:

Index
The digital input index.

OutputCount

Gets the number of digital outputs supported by this board.

```
OutputCount as Long [get]
```

OutputState

Gets / Sets the state of a digital output.

```
OutputState(
   Index as Long
) as Boolean [get,set]
```

Parameters:

Index
The digital output index.

SensorCount

Gets the number of sensors (analog inputs) supported by this board.

```
SensorCount as Long [get]
```

SensorValue

Gets the value of a sensor (0-1000).

```
SensorValue(
   Index as Long
) as Long [get]
```

Parameters:

Index
The sensor index.

SensorRawValue

Gets the raw value of a sensor (12-bit).

```
SensorRawValue(
   Index as Long
) as Long [get]
```

Parameters:

Index
The sensor index.

SensorChangeTrigger

Gets / Sets the change trigger for a sensor.

```
SensorChangeTrigger(
   Index as Long
) as Long [get,set]
```

Parameters:

Index
The sensor index.

Ratiometric

Gets / Sets the ratiometric state of the board.

```
Ratiometric as Boolean [get,set]
```

DataRate

Gets / Sets the data rate for a sensor, in milliseconds. See user guide for supported data rates.

```
DataRate(
   Index as Long
) as Long [get,set]
```

Parameters:

Index
The sensor index.

DataRateMin

Gets the minimum data rate for a sensor, in milliseconds.

```
DataRateMin(
   Index as Long
) as Long [get]
```

Parameters:

Index
The sensor index.

DataRateMax

Gets the maximum data rate for a sensor, in milliseconds.

```
DataRateMax(
   Index as Long
) as Long [get]
```

Parameters:

Index
The sensor index.

Events

OnInputChange

Fired when a digital input changes.

```
event OnInputChange(
   Index as Long,
   NewState as Boolean
)
```

Parameters:

Index
The digital inputs index.

NewState
The digital input state.

OnOutputChange

Fired when a digital output changes.

```
event OnOutputChange(
   Index as Long,
   NewState as Boolean
)
```

Parameters:

Index
The digital output index.

NewState
The digital output state.

OnSensorChange

Fired when a sensor value changes by more then the change trigger.

```
event OnSensorChange(
   Index as long,
   SensorValue as long
)
```

Parameters:

Index
The sensor index.

SensorValue
The sensor value.

PhidgetIR

Class documentation for PhidgetIR. This class contains all calls specific to the Phidget IR. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Due to the nature of COM, executables may only work on the computer it is built on.

Structures

PhidgetIR_CodeInfo

Used for the `Transmit` function and the `OnLearn` event. This structure represents all parameters needed to send an IR code.

```
struct PhidgetIR_CodeInfo
{
   long bitCount;
   long encoding;
   long length;
   long gap;
   long trail;
   long header[2];
   long one[2];
   long zero[2];
   long repeat[26];
   long min_repeat;
   byte toggle_mask[16];
   long carrierFrequency;
   long dutyCycle;
}
```

PhidgetIR_Encoding

Used for the `encoding` field of the `PhidgetIR_CodeInfo` structure. These are the predefined encodings supported by a Phidget IR.

```
enum PhidgetIR_Encoding {
   PHIDGETCOM_IR_ENCODING_UNKNOWN = 1,
   PHIDGETCOM_IR_ENCODING_SPACE,
   PHIDGETCOM_IR_ENCODING_PULSE,
   PHIDGETCOM_IR_ENCODING_BIPHASE,
   PHIDGETCOM_IR_ENCODING_RC5,
   PHIDGETCOM_IR_ENCODING_RC6
```

```
}
```

PhidgetIR_Length

Used for the `length` field of the `PhidgetIR_CodeInfo` structure. These are the predefined length styles supported by a Phidget IR.

```
enum PhidgetIR_Length {
   PHIDGETCOM_IR_LENGTH_UNKNOWN = 1,
   PHIDGETCOM_IR_LENGTH_CONSTANT,
   PHIDGETCOM_IR_LENGTH_VARIABLE
}
```

Functions

Transmit

Transmits a code.

```
Transmit(
   data() as Byte,
   codeInfo as PhidgetIR_CodeInfo
)
```

Parameters:

data
An array of code data, right justified.

codeInfo
Code parameters.

TransmitRepeat

Transmits a repeat code. Must quickly follow a `transmit` call.

```
TransmitRepeat()
```

TransmitRaw

Transmits a set of raw data.

```
TransmitRaw(
   data() as Long,
   length as Long,
   carrierFrequency as Long,
   dutyCycle as Long,
```

```
    gap as Long
)
```

Parameters:

data
An array of raw data in microseconds. Must start and end with a pulse.

length
Size of the data array, or amount of data to send.

carrierFrequency
Carrier frequency in kHz.

dutyCycle
Duty cycle in percent.

gap
Gap length in microseconds.

getLastCode

Gets the last recieved code

```
getLastCode(
    data() as Byte,
    dataLength as Long,
    bitCount as Long
)
```

Parameters:

data
An array to store the code data.

dataLength
Length of the data array, set to the length of code data.

bitCount
Set to the bit count.

getLastLearnedCode

Gets the last recieved code

```
getLastLearnedCode(
    data() as Byte,
    dataLength as Long,
```

```
    codeInfo as PhidgetIR_CodeInfo
)
```

Parameters:

data
An array to store the code data.

dataLength
Length of the data array, set to the length of code data.

codeInfo
Set to the parameters for this code.

getRawData

Reads in raw data.

```
getRawData(
    data() as Long,
    dataLength as Long
)
```

Parameters:

data
An array to store the raw data.

dataLength
Length of the data array, set to the amount of raw data read.

Events

OnCode

Fired when an IR code is received.

```
event OnCode(
    data() as Byte,
    dataLength as Long,
    bitCount as Long,
    repeat as Boolean
)
```

Parameters:

data
The code data.

dataLength
Size of the data array

bitCount
Number of bits in the code

repeat

Whether the code is a repeat (button held down).

OnLearnedCode

Fired when an IR code is learned.

```
event OnLearnedCode(
   data() as Byte,
   dataLength as Long,
   codeInfo as PhidgetIR_CodeInfo
)
```

Parameters:

data
The code data.

dataLength
Size of the data array

codeInfo
Code parameters.

OnRawData

Fired when an IR code is received.

```
event OnRawData(
   data() as Long,
   dataLength as Long
)
```

Parameters:

data
The raw data, in microseconds.

dataLength
Size of the data array

PhidgetLED

Class documentation for PhidgetLED. This class contains all calls specific to the Phidget LED. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Structures

PhidgetCOM_LEDCurrentLimit

Used for the `CurrentLimit` property. These are the predefined current limits supported by a Phidget LED 64 Advanced.

```
enum PhidgetCOM_LEDCurrentLimit {
   PHIDGETCOM_LED_CURRENT_LIMIT_20mA = 1,
   PHIDGETCOM_LED_CURRENT_LIMIT_40mA,
   PHIDGETCOM_LED_CURRENT_LIMIT_60mA,
   PHIDGETCOM_LED_CURRENT_LIMIT_80mA
}
```

PhidgetCOM_LEDVoltage

Used for the `Voltage` property. These are the predefined voltages supported by a Phidget LED 64 Advanced.

```
enum PhidgetCOM_LEDVoltage {
   PHIDGETCOM_LED_VOLTAGE_1_7V = 1,
   PHIDGETCOM_LED_VOLTAGE_2_75V,
   PHIDGETCOM_LED_VOLTAGE_3_9V,
   PHIDGETCOM_LED_VOLTAGE_5_0V
}
```

Properties

LEDCount

Gets the number of LEDs supported by this controller.

```
LEDCount as Long [get]
```

Brightness

Gets / Sets the brightness of an LED (0-100).

```
DiscreteLED(
   Index as Long
) as Double [get,set]
```

Parameters:

Index

The LED index.

CurrentLimitIndexed

Gets / Sets the current limit of an LED (0-100). Supported on 1032 only.

```
CurrentLimitIndexed(
   Index as Long
) as Double [get,set]
```

Parameters:

Index
The LED index.

CurrentLimit

Gets / Sets the current limit for all LEDs. Supported on 1031 only.

```
CurrentLimit as PhidgetCOM_LEDCurrentLimit [get,set]
```

Note that settable current limit is not supported by all PhidgetLEDs.

Voltage

Gets / Sets the voltage for all LEDs.

```
Voltage as PhidgetCOM_LEDVoltage [get,set]
```

Note that settable voltage is not supported by all PhidgetLEDs.

PhidgetMotorControl

Class documentation for PhidgetMotorControl. This class contains all calls specific to the Phidget Motor Control. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Properties

InputCount

Gets the number of digital inputs supported by this controller.

```
InputCount as Long [get]
```

InputState

Gets the state of a digital input.

```
InputState(
   Index as Long
) as Boolean [get]
```

Parameters:

Index

The digital input index.

EncoderCount

Gets the number of encoders supported by the PhidgetMotorControl. Not supported by all PhidgetMotorControls.

```
EncoderCount as Long [get]
```

EncoderPosition

Gets / Sets the position of an encoder. The position starts at 0 every time the PhidgetMotorControl is opened. Not supported by all PhidgetMotorControls.

```
EncoderPosition(
   Index as Long
) as Long [get,set]
```

Parameters:

Index

The encoder index.

SensorCount

Gets the number of sensors(Analog Inputs) supported by the PhidgetMotorControl. Not supported by all PhidgetMotorControls.

```
SensorCount as Long [get]
```

SensorValue

Gets the sensor value of a particular Analog Input. SensorValue varies between 0-1000. If you are using an Analog Sensor from Phidgets Inc., its User Guide will specify the formula used to convert SensorValue into the measured property. Not supported by all PhidgetMotorControls.

```
SensorValue(
   Index as Long
) as Long [get]
```

Parameters:

Index
The sensor index.

SensorRawValue

Gets the raw value of a analog input. Not supported by all PhidgetMotorControls.

```
SensorRawValue(
   Index as Long
) as Long [get]
```

Parameters:

Index
The sensor index.

Ratiometric

Gets / Sets the ratiometric state. This controls the voltage reference used for sampling the analog sensors. Not supported by all PhidgetMotorControls

```
Ratiometric as Boolean [get,set]
```

BackEMFSensingState

Gets / Sets the Back EMF Sensing State for the specified motor. Not supported by all PhidgetMotorControls.

```
BackEMFSensingState(
   Index as Long
```

```
) as Boolean [get,set]
```

Parameters:

Index
The motor index.

BackEMF

Gets the Back EMF voltage for a specified motor. Not supported by all PhidgetMotorControls.

```
BackEMF(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

MotorCount

Gets the number of motors supported by this controller.

```
MotorCount as Long [get]
```

Acceleration

Gets / Sets the acceleration for a motor.

```
Acceleration(
   Index as Long
) as Double [get,set]
```

Parameters:

Index
The motor index.

AccelerationMax

Gets the maximum acceleration supported by a motor.

```
AccelerationMax(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

AccelerationMin

Gets the minimum acceleration supported by a motor.

```
AccelerationMin(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

Current

Gets the current current draw of a motor.

```
Current(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

SupplyVoltage

Gets the supply voltage for the motors. This could be higher than the actual supply voltage. Not supported by all PhidgetMotorControls.

```
SupplyVoltage as Double [get]
```

Velocity

Gets / Sets the current velocity of a motor.

```
Velocity(
   Index as Long
) as Double [get,set]
```

Parameters:

Index
The motor index.

Braking

Gets / Sets the braking value for a specified motor. This is applied when the velocity is 0. Default is 0%. Not supported by all PhidgetMotorControls.

```
Braking(
   Index as Long
) as Double [get,set]
```

Parameters:

Index
The motor index.

Events

OnInputChange

Fired when a digital input changes.

```
event OnInputChange(
    Index as Long,
    NewState as Boolean
)
```

Parameters:

Index
The digital input index.

NewState
The state of the input.

OnVelocityChange

Fired when the velocity of a motor changes.

```
event OnVelocityChange(
    Index as Long,
    Velocity as Double
)
```

Parameters:

Index
The motor index.

Velocity
The current velocity.

OnEncoderPositionChange

Fired when the encoder position changes. Not supported by all PhidgetMotorControls.

```
event OnEncoderPositionChange(
    Index as Long,
    Time as Long,
    PositionChange as Long
)
```

Parameters:

Index

The encoder index.

Time

The time since the last position change event.

PositionChange

The amount the position change since the last position change event.

OnEncoderPositionUpdate

Fired at a constant rate(every 8ms), regardless of whether the encoder position changed or not. Not supported by all PhidgetMotorControls.

```
event OnEncoderPositionUpdate(
   Index as Long,
   PositionChange as Long
)
```

Parameters:

Index

The encoder index.

PositionChange

The amount the position change since the last position update event.

OnBackEMFUpdate

Fired at a constant rate(every 16ms) when **BackEMFSensingState** is enabled for the specified motor. Not supported by all PhidgetMotorControls.

```
event OnBackEMFUpdate(
   Index as Long,
   Voltage as Double
)
```

Parameters:

Index

The motor index.

Voltage

The Back EMF voltage for a motor.

OnSensorUpdate

Fired at a constant rate(every 8ms). Not supported by all PhidgetMotorControls.

```
event OnSensorUpdate(
   Index as Long,
   SensorValue as Long
)
```

Parameters:

Index
The sensor index.

SensorValue
The sensor value of a particular Analog Input.

OnCurrentChange

Fired whenever the current draw changes. Not supported by all PhidgetMotorControls.

```
event OnCurrentChange(
   Index as Long,
   Current as Double
)
```

Parameters:

Index
The motor index.

Current
The current draw for a motor.

OnCurrentUpdate

Fired at a constant rate(8ms). Not supported by all PhidgetMotorControls.

```
event OnCurrentUpdate(
   Index as Long,
   Current as Double
)
```

Parameters:

Index
The motor index.

Current
The current draw for a motor.

PhidgetPHSensor

Class documentation for PhidgetPHSensor. This class contains all calls specific to the Phidget PH Sensor. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Properties

PH

Gets the currently sensed PH.

```
PH as Double [get]
```

PHMax

Gets the maximum PH that could be sensed.

```
PHMax as Double [get]
```

PHMin

Gets the minimum PH that could be sensed.

```
PHMin as Double [get]
```

PHChangeTrigger

Gets / Sets the PH change trigger.

```
PHChangeTrigger as Double [get,set]
```

Potential

Gets the currently sensed potential.

```
Potential as Double [get]
```

PotentialMax

Gets the maximum potential that the board can sense.

```
PotentialMax as Double [get]
```

PotentialMin

Gets the minimum potential that the board can sense.

```
PotentialMin as Double [get]
```

Temperature

Sets the temperature value used for the PH calculation. Default is 20 degrees Celsius.

```
Temperature as Double [set]
```

Events

OnPHChange

Fired when the PH changes by more then the change trigger.

```
event OnPHChange(
   PH as Double
)
```

Parameters:

PH
The PH.

PhidgetRFID

Class documentation for PhidgetRFID. This class contains all calls specific to the Phidget RFID. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Structures

PhidgetCOM_RFIDProtocol

Used for the `LastTagProtocol` property and `write` function. These are the protocols supported by the various RFID readers / writers.

```
enum PhidgetCOM_RFIDProtocol {
   PHIDGETCOM_RFID_PROTOCOL_EM4100 = 1,
   PHIDGETCOM_RFID_PROTOCOL_ISO11785_FDX_B,
   PHIDGETCOM_RFID_PROTOCOL_PHIDGETS
}
```

Functions

Write

Write data to a tag.

```
Write(
   TagString as String,
   Protocol as PhidgetCOM_RFIDProtocol,
   Lock as Boolean
);
```

Parameters:

TagString
The tag string to program. See product manual for string formatting details.

Protocol
The protocol to use.

Lock
Locks the tag from being written again.

Properties

OutputCount

Gets the number of digital outputs supported by this board.

```
OutputCount as Long [get]
```

OutputState

Gets / Sets the state of a digital output.

```
OutputState(
   Index as Long
) as Boolean [get,set]
```

Parameters:

Index
The digital output index.

AntennaOn

Gets / Sets the state of the antenna. Note that antenna must be enabled before tags will be read.

```
AntennaOn as Boolean [get,set]
```

LEDOn

Gets / Sets the state of the onboard LED.

```
LEDOn as Boolean [get,set]
```

TagStatus

Gets the tag status. This is true if there is a tag on the reader.

```
TagStatus as Boolean [get]
```

LastTag

Gets the last tag that was read. This tag may or may not still be on the reader.

```
LastTag as String [get]
```

LastTagProtocol

Gets the protocol of the last tag that was read. This tag may or may not still be on the reader.

```
LastTagProtocol as PhidgetCOM_RFIDProtocol [get]
```

Events

OnOutputChange

Fired when a digital output changes.

```
event OnOutputChange(
   Index as Long,
   NewState as Boolean
)
```

Parameters:

Index
The digital output index.

NewState
The digital output state.

OnTag

Fired when a tag is detected.

```
event OnTag(
   TagNumber as String
)
```

Parameters:

TagNumber
The detected tag.

OnTagLost

Fired when a tag is is taken off the reader.

```
event OnTagLost(
   TagNumber as String
)
```

Parameters:

TagNumber
The lost tag.

PhidgetServo

Class documentation for PhidgetServo. This class contains all calls specific to the Phidget Servo. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Structures

PhidgetCOM_ServoType

Used for the `ServoType` property. These are the predefined servo types supported by Phidgets Inc. This list may be incomplete.

```
enum PhidgetCOM_ServoType {
   PHIDGETCOM_SERVO_DEFAULT = 1,
   PHIDGETCOM_SERVO_RAW_us_MODE,
   PHIDGETCOM_SERVO_HITEC_HS322HD,
   PHIDGETCOM_SERVO_HITEC_HS5245MG,
   PHIDGETCOM_SERVO_HITEC_805BB,
   PHIDGETCOM_SERVO_HITEC_HS422,
   PHIDGETCOM_SERVO_TOWERPRO_MG90,
   ...,
   PHIDGETCOM_SERVO_USER_DEFINED
}
```

Functions

setServoParameters

Sets parameters for a custom servo type. This includes PCM range and degrees of rotation. This affect min and max position, and the PCM to degree mapping formulas.

```
setServoParameters(
   Index as Long,
   MinUs as double,
   MaxUs as double,
   Degrees as double
);
```

Parameters:

Index
The motor index.

MinUs
The minimum PCM supported by the motor, in microseconds.

MaxUs
The maximum PCM supported by the motor, in microseconds.

Degrees
Real degrees of rotation represented by the given PCM range

Properties

MotorCount

Gets the number of motors supported by this controller.

```
MotorCount as Long [get]
```

Position

Gets / Sets the current / target position of a motor.

```
Position(
   Index as Long
) as Double [get,set]
```

Parameters:

Index
The motor index.

PositionMax

Gets the maximum position supported by a motor.

```
PositionMax(
   Index and Long
) as Double [get]
```

Parameters:

Index
The motor index.

PositionMin

Gets the minimum position supported by a motor.

```
PositionMin(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

Engaged

Gets / Sets the engaged state of a motor. This is whether a motor is powered or not.

```
Engaged(
   Index as Long
) as Boolean [get,set]
```

Parameters:

Index
The motor index.

ServoType

Gets / Sets the servo type for an index. There are several predefined servo types. All other types of servos can be set up using the `setServoParameters` function.

```
ServoType(
   Index as Long
) as PhidgetCOM_ServoType [get,set]
```

Parameters:

Index
The motor index.

Events

OnPositionChange

Fired when the position of a motor changes.

```
event OnPositionChange(
   Index as Long,
   Position as Double
)
```

Parameters:

Index
The motor index.

Position
The motor position.

PhidgetSpatial

Class documentation for PhidgetSpatial. This class contains all calls specific to the Phidget Spatial. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Structures

PhidgetSpatial_SpatialEventData

Used for the `OnSpatialData` event. This structure represents all the state of a PhidgetSpatial at a moment in time.

```
struct PhidgetSpatial_SpatialEventData
{
   double acceleration[3];
   double angularRate[3];
   double magneticField[3];
   long time_seconds;
   long time_microseconds;
}
```

Properties

AccelerationAxisCount

Gets the number of acceleration axes supported by this board.

```
AccelerationAxisCount as Long [get]
```

Acceleration

Gets the current acceleration of a axis.

```
Acceleration(
   Index as Long
) as Double [get]
```

Parameters:

Index
The acceleration axis.

AccelerationMax

Gets the maximum acceleration that can be measured by as axis.

```
AccelerationMax(
   Index as Long
) as Double [get]
```

Parameters:

Index
The acceleration axis.

AccelerationMin

Gets the minimum acceleration that can be measured by an axis.

```
AccelerationMin(
   Index as Long
) as Double [get]
```

Parameters:

Index
The acceleration axis.

GyroAxisCount

Gets the number of gyroscope axes supported by this board.

```
GyroAxisCount as Long [get]
```

AngularRate

Gets the current angular rate of a axis.

```
AngularRate(
   Index as Long
) as Double [get]
```

Parameters:

Index
The gyro axis.

AngularRateMax

Gets the maximum angular rate that can be measured by as axis.

```
AngularRateMax(
   Index as Long
) as Double [get]
```

Parameters:

Index
The gyro axis.

AngularRateMin

Gets the minimum angular rate that can be measured by an axis.

```
AngularRateMin(
   Index as Long
) as Double [get]
```

Parameters:

Index
The gyro axis.

CompassAxisCount

Gets the number of compass axes supported by this board.

```
CompassAxisCount as Long [get]
```

MagneticField

Gets the current magnetic field of a axis.

```
MagneticField(
   Index as Long
) as Double [get]
```

Parameters:

Index
The compass axis.

MagneticFieldMax

Gets the maximum magnetic field that can be measured by as axis.

```
MagneticFieldMax(
   Index as Long
) as Double [get]
```

Parameters:

Index
The compass axis.

MagneticFieldMin

Gets the minimum magnetic field that can be measured by an axis.

```
MagneticFieldMin(
   Index as Long
) as Double [get]
```

Parameters:

Index
The compass axis.

DataRate

Gets / Sets the data rate, in milliseconds. See the user guide for supported data rates.

```
DataRate as Long [get,set]
```

DataRateMin

Gets the minimum data rate, in milliseconds.

```
DataRateMin as Long [get]
```

DataRateMax

Gets the maximum data rate , in milliseconds.

```
DataRateMax as Long [get]
```

CompassCorrectionParameters

Sets the compass correction parameters. This is for filtering out any compass errors including hard and soft iron offsets.

```
CompassCorrectionParameters(
   magField as Double,
   offset0 as Double,
   offset1 as Double,
   offset2 as Double,
   gain0 as Double,
   gain1 as Double,
   gain2 as Double,
   T0 as Double,
   T1 as Double,
   T2 as Double,
   T3 as Double,
   T4 as Double,
   T5 as Double
) [set]
```

Parameters:

magField
Ambient magnetic field.

offset0, osffest1, offset2
Offset corrections.

gain0, gain1, gain2
Gain corrections

T0 - T5
Non-orthogonality corrections.

Functions

zeroGyro

Zeroes the gyro. Should only be called when the board in still.

```
zeroGyro();
```

resetCompassCorrectionParameters

Resets compass correction parameters.

```
resetCompassCorrectionParameters();
```

Events

OnSpatialData

Fired at `DataRate`.

```
event OnSpatialData(
   data() as PhidgetSpatial_SpatialEventData,
   dataCount as Long
)
```

Parameters:

data
An array of sets of spatial data.

dataCount
The number of spatial data sets in this event.

PhidgetStepper

Class documentation for PhidgetStepper. This class contains all calls specific to the Phidget Stepper. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Properties

InputCount

Gets the number of digital inputs supported by this board.

```
InputCount as Long [get]
```

InputState

Gets the state of a digital input.

```
InputState(
   Index as Long
) as Boolean [get]
```

Parameters:

Index
The digital input index.

MotorCount

Gets the number of motors supported by this controller.

```
MotorCount as Long [get]
```

Acceleration

Gets / Sets the acceleration for a motor.

```
Acceleration(
   Index as Long
) as Double [get,set]
```

Parameters:

Index
The motor index.

AccelerationMax

Gets the maximum acceleration supported by a motor.

```
AccelerationMax(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

AccelerationMin

Gets the minimum acceleration supported by a motor.

```
AccelerationMin(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

Current

Gets the current current draw of a motor.

```
Current(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

CurrentLimit

Gets / Sets the current limit for a motor.

```
CurrentLimit(
   Index as Long
) as Double [get,set]
```

Parameters:

Index
The motor index.

CurrentMax

Gets the maximum current limit supported by a motor.

```
CurrentMax(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

CurrentMin

Gets the minimum current limit supported by a motor.

```
CurrentMin(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

CurrentPosition

Gets / Sets the current position of a motor.

```
CurrentPosition(
   Index as Long
) as Long [get,set]
```

Parameters:

Index
The motor index.

TargetPosition

Gets / Sets the target position of a motor.

```
TargetPosition(
   Index as Long
) as Long [get,set]
```

Parameters:

Index
The motor index.

PositionMax

Gets the maximum position supported by a motor.

```
PositionMax(
   Index and Long
) as Long [get]
```

Parameters:

Index
The motor index.

PositionMin

Gets the minimum position supported by a motor.

```
PositionMin(
   Index as Long
) as Long [get]
```

Parameters:

Index
The motor index.

Velocity

Gets the current velocity of a motor.

```
Velocity(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

VelocityLimit

Gets / Set the velocity limit of a motor.

```
VelocityLimit(
   Index as Long
) as Double [get,set]
```

Parameters:

Index
The motor index.

VelocityMax

Gets the maximum velocity limit supported by a motor.

```
VelocityMax(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

VelocityMin

Gets the minimum velocity limit supported by a motor.

```
VelocityMin(
   Index as Long
) as Double [get]
```

Parameters:

Index
The motor index.

Engaged

Gets / Sets the engaged state of a motor. This is whether a motor is powered or not.

```
Engaged(
   Index as Long
) as Boolean [get,set]
```

Parameters:

Index
The motor index.

Stopped

Gets the stopped state of a motor. If this is true, the motor is not moving and there are no outstanding commands.

```
Stopped(
   Index as Long
) as Boolean [get]
```

Parameters:

Index
The motor index.

Events

OnInputChange

Fired when a digital input changes.

```
event OnInputChange(
    Index as Long,
    NewState as Boolean
)
```

Parameters:

Index
The digital inputs index.

NewState
The digital input state.

OnCurrentChange

Fired when the current draw of a motor changes.

```
event OnCurrentChange(
    Index as Long,
    Current as Double
)
```

Parameters:

Index
The motor index.

Current
The current draw.

OnPositionChange

Fired when the position of a motor changes.

```
event OnPositionChange(
    Index as Long,
    Position as Long
)
```

Parameters:

Index
The motor index.

Position

The motor position.

OnVelocityChange

Fired when the velocity of a motor changes.

```
event OnVelocityChange(
   Index as Long,
   Velocity as Double
)
```

Parameters:

Index

The motor index.

Velocity

The current velocity.

PhidgetTemperatureSensor

Class documentation for PhidgetTemperatureSensor. This class contains all calls specific to the Phidget Temperature Sensor. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Properties

TemperatureInputCount

Gets the number of thermocouple inputs supported by this board.

```
TemperatureInputCount as Long [get]
```

Temperature

Gets the currently sensed temperature of a thermocouple input.

```
Temperature(
   Index as Long
) as Double [get]
```

Parameters:

Index
The thermocouple input index.

TemperatureMax

Gets the maximum temperature that a thermocouple input can measure.

```
TemperatureMax(
   Index as Long
) as Double [get]
```

Parameters:

Index
The thermocouple input index.

TemperatureMin

Gets the minimum temperature that a thermocouple input can measure.

```
TemperatureMin(
   Index as Long
) as Double [get]
```

Parameters:

Index
The thermocouple input index.

TemperatureChangeTrigger

Gets / Sets the change trigger for a thermocouple input.

```
TemperatureChangeTrigger(
   Index as Long
) as Double [get,set]
```

Parameters:

Index
The thermocouple input index.

ThermocoupleType

Gets / Sets the type of thermocouple attached to a thermocouple input.

```
ThermocoupleType(
   Index as Long
) as Long [get,set]
```

Parameters:

Index
The thermocouple input index.

Discussion:

There are 4 thermocouple types supported: K-Type=1, J-Type=2, E-Type=3 and T-Type=4.

Potential

Gets the currently measured potential at a thermocouple input.

```
Potential(
   Index as Long
) as Double [get]
```

Parameters:

Index
The thermocouple input index.

PotentialMax

Gets the maximum potential that a thermocouple input can measure.

```
PotentialMax(
   Index as long
) as Double [get]
```

Parameters:

Index
The thermocouple input index.

PotentialMin

Gets the minimum potential that a thermocouple input can measure.

```
PotentilaMin(
   Index as Long
) as Double [get]
```

Parameters:

Index
The thermocouple input index.

AmbientTemperature

Gets the ambient (board) temperature.

```
AmbientTemperature as Double [get]
```

AmbientTemperatureMax

Gets the maximum temperature that the ambient sensor can measure.

```
AmbientTemperatureMax as Double [get]
```

AmbientTemperatureMin

Gets the minimum temperature that the ambient sensor can measure.

```
AmbientTemperatureMin as Double [get]
```

Events

OnTemperatureChange

Fired when the temperature of a thermocouple changes by more then the change trigger.

```
event OnTemperatureChange(
   Index as Long,
   Temperature as Double
)
```

Parameters:

Index
The thermocouple input index.

Temperature
The temperature.

PhidgetTextLCD

Class documentation for PhidgetTextLCD. This class contains all calls specific to the Phidget Text LCD. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Structures

PhidgetCOM_TextLCDScreenSize

Used for the `ScreenSize` property. These are the predefined screen sizes supported by PhidgetTextLCD. Not supported by all PhidgetTextLCDs.

```
enum PhidgetCOM_TextLCDScreenSize {
   PHIDGETCOM_TEXTLCD_SCREEN_NONE = 1,
   PHIDGETCOM_TEXTLCD_SCREEN_1x8,
   PHIDGETCOM_TEXTLCD_SCREEN_2x8,
   PHIDGETCOM_TEXTLCD_SCREEN_2x16,
   PHIDGETCOM_TEXTLCD_SCREEN_4x16,
   PHIDGETCOM_TEXTLCD_SCREEN_2x20,
   PHIDGETCOM_TEXTLCD_SCREEN_4x20,
   PHIDGETCOM_TEXTLCD_SCREEN_2x24,
   PHIDGETCOM_TEXTLCD_SCREEN_1x40,
   PHIDGETCOM_TEXTLCD_SCREEN_2x40,
   PHIDGETCOM_TEXTLCD_SCREEN_4x40,
   PHIDGETCOM_TEXTLCD_SCREEN_UNKNOWN
}
```

Functions

CustomCharacter

Sets a custom character. See the TextLCD User Guide for more information.

```
CustomCharacter(
   Index as Long,
   Val1 as Long,
   Val2 as Long
) [set]
```

Parameters:

Index

The custom character index (8-15)

Val1

The first half of the custom character

Val2

The second half of the custom character

DisplayString

Sets the string to display on a row.

```
DisplayString(
   Index as Long
) [set]
```

Parameters:

Index
The row index.

DisplayCharacter

Sets the character to display at a specific row and column. Pass in as a one character string.

```
DisplayCharacter(
   Row as Long,
   Column as Long
) [set]
```

Parameters:

Row
The row index.

Column
The column index.

Initialize

Initializes the active TextLCD display. This runs an initialization routine which sets up and clears the display. This can be used for activating a display that was plugged in after the TextLCD was attached, to clear the display after setting/ getting the screen size, and to re-initialize a display if it has become corrupted. Not supported by all PhidgetTextLCDs.

```
Initialize()
```

Properties

Screen

Gets / Sets the active screen. All other API calls depend on this being called first to select the screen that subsequent calls affect. Not supported by all TextLCDs.

```
Screen as Long [get,set]
```

RowCount

Gets the number of display rows supported by this board.

```
RowCount as Long [get]
```

ColumnCount

Gets the number of columns per row supported by this board.

```
ColumnCount as Long [get]
```

ScreenCount

Gets the number of screens suppoted by the TextLCD. Not supported by all TextLCDs.

```
ScreenCount as Long [get]
```

ScreenSize

Gets / Sets the screen size for the active TextLCD display. The TextLCD Adapter supports a pre-defiend set of screen sizes to choose from. By default, both screens are set to `PHIDGETCOM_TEXTLCD_SCREEN_NONE`, and this function must always be called before trying to write text to a display. Not supported by all TextLCDs.

```
ScreenSize as PhidgetCOM_TextLCDScreenSize [get,set]
```

Backlight

Gets / Sets the backlight state.

```
Backlight as Boolean [get,set]
```

Brightness

Gets / Sets the brightness of the backlight (0-255).

Note that only some TextLCDs support a settable backlight brightness.

```
Brightness as Long [get,set]
```

Contrast

Gets / Sets the contrast (0-255).

```
Constrast as Long [get,set]
```

CursorBlink

Gets / Sets the cursor blink state.

```
CursorBlink as Boolean [get,set]
```

CursorOn

Gets / Sets the cursor on state.

```
CursorOn as Boolean [get,set]
```

PhidgetTextLED

Class documentation for PhidgetTextLED. This class contains all calls specific to the Phidget Text LED. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Properties

RowCount

Gets the number of display rows supported by this board.

```
RowCount as Long [get]
```

ColumnCount

Gets the number of columns per row supported by this board

```
ColumnCount as Long [get]
```

Brightness

Gets / Sets the brightness.

```
Brightness as Long [get,set]
```

DisplayString

Sets the string to display on a row.

```
DisplayString(
   Index as Long
) as String [set]
```

Parameters:

Index
The row index.

PhidgetWeightSensor

Class documentation for PhidgetWeightSensor. This class contains all calls specific to the Phidget Weight Sensor. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Properties

Weight

Gets the currently sensed weight.

```
Weight as Double [get]
```

WeightChangeTrigger

Gets / Sets the change trigger.

```
WeightChangeTrigger as Double [get,set]
```

Events

OnWeightChange

Fired when the weight changes by more then the change trigger.

```
event OnWeightChange(
   Weight as Double
)
```

Parameters:

Weight
The weight.

PhidgetManager

Class documentation for PhidgetManager. This class contains all calls specific to the Phidget Manager. See the Phidget Manager page for more information.

Functions

Open

Opens a manager

```
Open();
```

OpenRemote

Opens a manager remotely using a server id.

```
OpenRemote(
   ServerID as String [optional],
   Password as String [optional]
);
```

Parameters:

ServerID
Server ID of the webservice to connect to. Not not specify to connect to any.

Password
Password of the webservice. Do not specify if the webservice does not have a password.

OpenRemoteIP

Opens a manager remotely using an address and port.

```
OpenRemoteIP(
   IPAddress as String,
   Port as Long,
   Password as String [optional]
);
```

Parameters:

IPAddress
The address of the webservice to connect to.

Port
The port of the webservice to connect to.

Password
Password of the webservice. Do not specify if the webservice does not have a password.

Close

Closes a manager.

```
Close();
```

Properties

Count

Gets the number of attached phidgets. Use with the Device functions to enumerate connected devices by index.

```
Count as Long [get]
```

DeviceType

Gets the device type of a Phidget.

```
DeviceType(
   Index as Long
) as String [get]
```

Parameters:

Index
Index of an attached phidget.

DeviceVersion

Gets the firmware version of a Phidget.

```
DeviceVersion(
   Index as Long
) as Long [get]
```

Parameters:

Index
Index of an attached phidget.

DeviceName

Gets the long name of a Phidget.

```
DeviceName(
   Index as Long
) as String [get]
```

Parameters:

Index
Index of an attached phidget.

DeviceSerial

Gets the unique serial number of a Phidget.

```
DeviceSerial(
   Index as Long
) as Long [get]
```

Parameters:

Index
Index of an attached phidget.

DeviceLabel

Gets the Label of a Phidget.

```
DeviceLabel(
   Index as Long
) as String [get]
```

Parameters:

Index
Index of an attached phidget.

IsAttachedToServer

Gets the attached to server state of a remotely opened manager.

```
IsAttachedToServer as Boolean [get]
```

Address

Gets the webservice address of a remotely opened manager.

```
Address as String [get]
```

Port

Gets the webservice port number of a remotely opened manager.

```
Port as Long [get]
```

ServerID

Gets the webservice Server ID of a remotely opened manager.

```
ServerID as String [get]
```

Events

OnAttach

Fired when a Phidget is plugged in.

```
event OnAttach(
   deviceType as String,
   deviceName as String,
   serialNumber as Long,
   deviceVersion as Long,
   deviceLabel as String
)
```

Parameters:

deviceType
The device type.

deviceName
The device name.

serialNumber
The serial number.

deviceVersion
The device version.

deviceLabel
The device label.

OnDetach

Fired when a Phidget is unplugged.

```
event OnDetach(
   deviceType as String,
   deviceName as String,
   serialNumber as Long,
   deviceVersion as Long,
   deviceLabel as String
)
```

Parameters:

deviceType
The device type.

deviceName
The device name.

serialNumber
The serial number.

deviceVersion
The device version.

deviceLabel
The device label.

OnError

Fired on an asynchronous error. These are mostly network related.

```
event OnError(
   Description as  String,
   SCODE as Long
)
```

Parameters:

Description
A description of the error.

SCODE
An error code corresponding to the error. See the General Phidget Programming page for a list of error codes.

OnServerConnect

Fired when a connection to the webservice is made, when opening a manager remotely.

```
event OnServerConnect
```

OnServerDisconnect

Fired when a connection to the webservice is lost, when opening a manager remotely.

```
event OnServerDisconnect
```

PhidgetDictionary

Class documentation for PhidgetDictionary. This class contains all calls specific to the Phidget Dictionary. See your device's User Guide for more specific API details, technical information, and revision details. The User Guide, along with other resources, can be found on the product page for your device.

Functions

OpenRemote

Opens a dictionary remotely using a server id.

```
OpenRemote(
   ServerID as String [optional],
   Password as String [optional]
);
```

Parameters:

ServerID
Server ID of the webservice to connect to. Not not specify to connect to any.

Password
Password of the webservice. Do not specify if the webservice does not have a password.

OpenRemoteIP

Opens a dictionary remotely using an address and port.

```
OpenRemote(
   IPAddress as String,
   Post as Long,
   Password as String [optional]
);
```

Parameters:

IPAddress
The address of the webservice to connect to.

Port
The port of the webservice to connect to.

Password
Password of the webservice. Do not specify if the webservice does not have a password.

Close

Closes the connection to a dictionary.

```
Close();
```

Add

Adds a key / value pair to the dictionary, or updates the value of an existing key.

```
Add(
   Key as String,
   Value as String,
   Persistent as Boolean [optional]
);
```

Parameters:

Key

The key string to add. The key can only contain numbers, letters, "/", ".", "-", "_", and must begin with a letter, "_" or "/".

Value

The value string.

Persistent

Whether this key remains in the dictionary once this connection is closed. Default is True.

Remove

Removes a set of keys from the dictionary.

```
Remove(
   Pattern as String
);
```

Parameters:

Pattern

A regular expression representing a set of keys to remove.

Get

Gets the value for a key.

```
Get(
   Key as String
) as String;
```

Parameters:

Key

The key to get the value of.

Returns:

The value for the specified key. This will be an empty string if the key does not exist.

Properties

IsAttachedToServer

Gets the attached to server state of a remotely opened manager.

```
IsAttachedToServer as Boolean [get]
```

Address

Gets the webservice address of a remotely opened manager.

```
Address as String [get]
```

Port

Gets the webservice port number of a remotely opened manager.

```
Port as Long [get]
```

ServerID

Gets the webservice Server ID of a remotely opened manager.

```
ServerID as String [get]
```

Events

OnError

Fired on an asynchronous error. These are mostly network related.

```
event OnError(
   Description as  String,
   SCODE as Long
)
```

Parameters:

Description
A description of the error.

SCODE
An error code corresponding to the error. See the General Phidget Programming page for a list of

error codes.

OnServerConnect

Fired when a connection to the webservice is made, when opening a manager remotely.

```
event OnServerConnect
```

OnServerDisconnect

Fired when a connection to the webservice is lost, when opening a manager remotely.

```
event OnServerDisconnect
```

PhidgetKeyListener

Class documentation for PhidgetKeyListener. This class enables the key listening abilities of the Phidget Dictionary. See the Phidget Dictionary page for more information.

Functions

Start

Starts listening for key changes on a dictionary with a specific pattern. This must be called on a connected dictionary, and is best called in the dictionary's `OnServerConnect` event.

```
Start(
   Dict as PhidgetDictionary,
   Pattern as String
);
```

Parameters:

Dict
The dictionary to listen for keys on.

Pattern
The key pattern to listen for.

Stop

Stops listening for keys. This should be called in the dictionary's `OnServerDisconnect` event.

```
Stop();
```

Properties

Pattern

Gets the key pattern that this listener is listening for.

```
Pattern as String [get]
```

Events

OnKeyChange

Fired when a key is added or a value changes.

```
event OnKeyChange(
   Key as String,
   Value as String
)
```

Parameters:

Key
The key value.

Value
The value value.

OnKeyRemoval

Fired when a key is removed.

```
event OnKeyRemoval(
   Key as String,
   Value as String
)
```

Parameters:

Key
The key value.

Value
The value value.