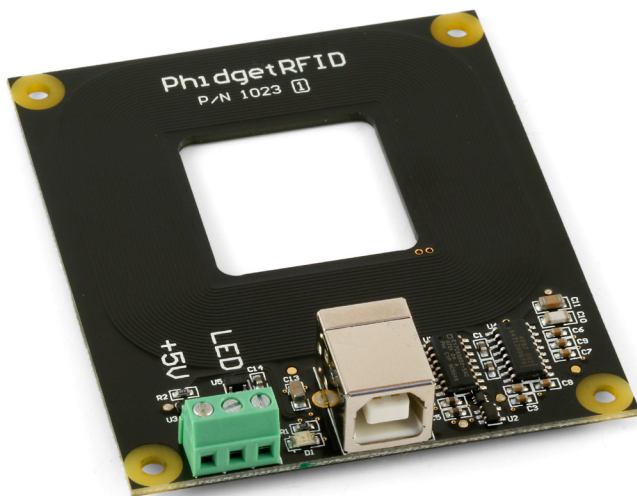


Phidgets

COM API Manual



Phidgets COM API Manual
Version 2.1.6
© Phidgets Inc. 2010

Last updated: February 2, 2010

Contents

Introduction

- 6 Overview
- 6 How to use Phidgets
- 6 Class Hierarchy

Phidget

- 7 Structures
- 8 Functions
- 11 Properties
- 12 Events

PhidgetAccelerometer

- 14 Properties
- 15 Events

PhidgetAdvancedServo

- 16 Structures
- 16 Functions
- 17 Properties
- 21 Events

PhidgetEncoder

- 22 Properties
- 23 Events

PhidgetInterfaceKit

- 24 Properties
- 26 Events

PhidgetLED

- 27 Structures
- 27 Properties

PhidgetMotorControl

- 29 Properties
- 31 Events

PhidgetPHSensor

32 Properties

33 Events

PhidgetRFID

34 Properties

35 Events

PhidgetServo

36 Structures

36 Functions

37 Properties

38 Events

PhidgetStepper

39 Properties

44 Events

PhidgetTemperatureSensor

46 Properties

48 Events

PhidgetTextLCD

49 Properties

PhidgetTextLED

51 Properties

PhidgetWeightSensor

52 Properties

52 Events

PhidgetManager

53 Functions

54 Properties

57 Events

PhidgetDictionary

59 Functions

61 Properties

61 Events

PhidgetKeyListener

63 Functions

63 Properties

64 Events

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.O, and therefore presents the COM interface as a VB6.O 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.O, 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 Product manual for your Phidget and the Programming Manual for more detailed, language unspecific API documentation. Also, there are a set of VB6.O examples available for download.

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. See the programming manual for more specific API details, supported functionality, units, etc.

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 Programming Manual.

```
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. Do 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.

```
OpenRemote(  
    IPAddress as String  
    Post 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 1-6. See the Programming Manual for more information.

file

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

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 than 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 Programming Manual 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 the product manual for more specific API details, supported functionality, units, etc.

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 the product manual for more specific API details, supported functionality, units, etc.

Structures

PhidgetCOM_ServoType

Used for the `ServoType` property. These are the predefined servo types supported by Phidgets Inc.

```
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.

PhidgetEncoder

Class documentation for PhidgetEncoder. This class contains all calls specific to the Phidget Encoder. See the product manual for more specific API details, supported functionality, units, etc.

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.

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 in milliseconds since the last position change event.

EncoderDisplacement

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

PhidgetInterfaceKit

Class documentation for PhidgetInterfaceKit. This class contains all calls specific to the Phidget Interface Kit. See the product manual for more specific API details, supported functionality, units, etc.

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]
```

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.

PhidgetLED

Class documentation for PhidgetLED. This class contains all calls specific to the Phidget LED. See the product manual for more specific API details, supported functionality, units, etc.

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]
```

DiscreteLED

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

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

Parameters:

Index

The LED index.

[PhidgetLED](#)

CurrentLimit

Gets / Sets the current limit for all LEDs;

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 the product manual for more specific API details, supported functionality, units, etc.

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.

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.

Velocity

Gets / Sets the current velocity of a motor.

```
Velocity(  
    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.

PhidgetPHSensor

Class documentation for PhidgetPHSensor. This class contains all calls specific to the Phidget PH Sensor. See the product manual for more specific API details, supported functionality, units, etc.

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 the product manual for more specific API details, supported functionality, units, etc.

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]
```

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 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 the product manual for more specific API details, supported functionality, units, etc.

Structures

PhidgetCOM_ServoType

Used for the `ServoType` property. These are the predefined servo types supported by Phidgets Inc.

```
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.

PhidgetStepper

Class documentation for PhidgetStepper. This class contains all calls specific to the Phidget Stepper. See the product manual for more specific API details, supported functionality, units, etc.

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 the product manual for more specific API details, supported functionality, units, etc.

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.

```
PotentialMin(  
    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 than 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 the product manual for more specific API details, supported functionality, units, etc.

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]

Backlight

Gets / Sets the backlight state.

Backlight as Boolean [get,set]

Contrast

Gets / Sets the contrast (0-255).

Contrast 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]

CustomCharacter

Sets a custom character. See the TextLCD manual 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  
) as String [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  
) as String [set]
```

Parameters:

Row

The row index.

Column

The column index.

PhidgetTextLED

Class documentation for PhidgetTextLED. This class contains all calls specific to the Phidget Text LED. See the product manual for more specific API details, supported functionality, units, etc.

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 the product manual for more specific API details, supported functionality, units, etc.

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 programming manual for more specific API details, supported functionality, etc.

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.

```
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 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 Programming Manual 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 the programming manual for more specific API details, supported functionality, etc.

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 Programming Manual 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 programming manual for more specific API details, supported functionality, etc.

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.