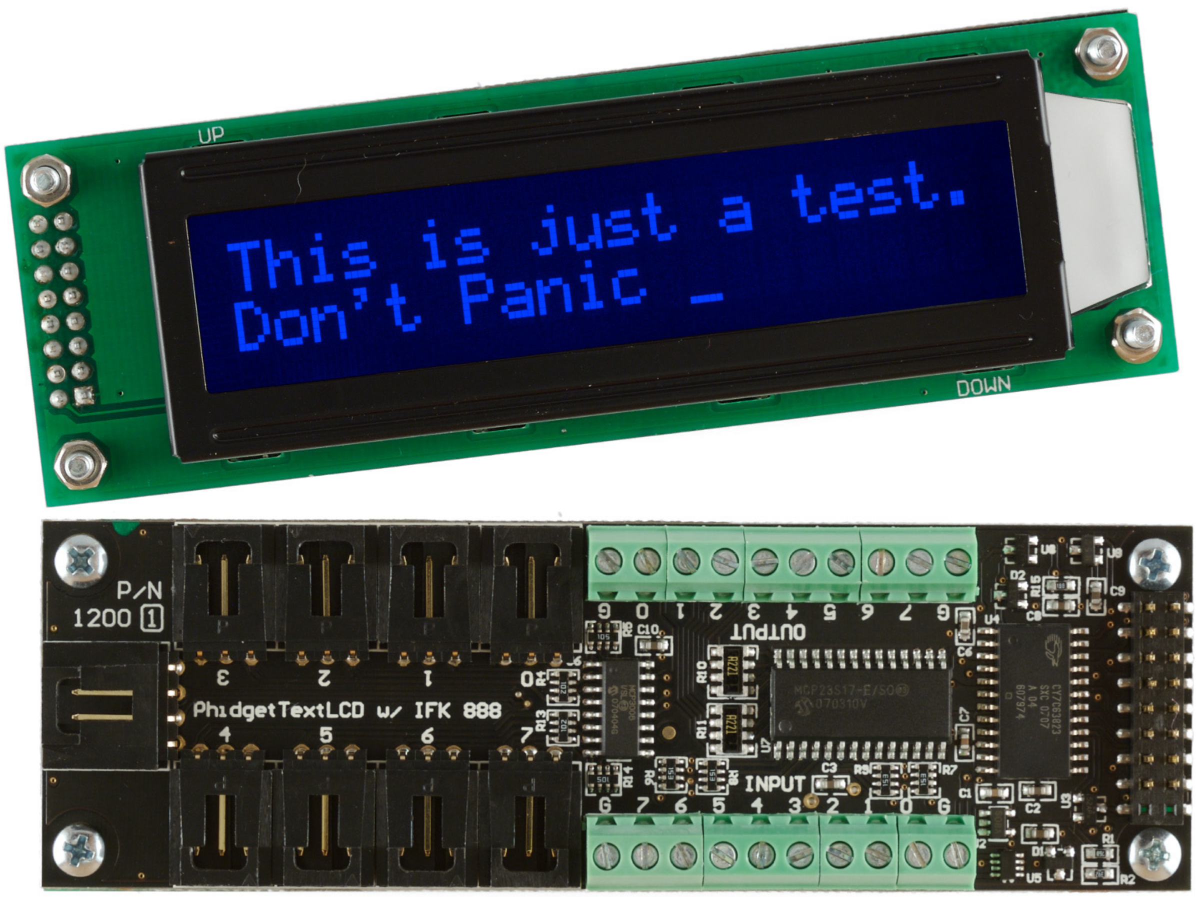
1202 PhidgetTextLCD with PhidgetInterfaceKit 8/8/8



Programming Environment

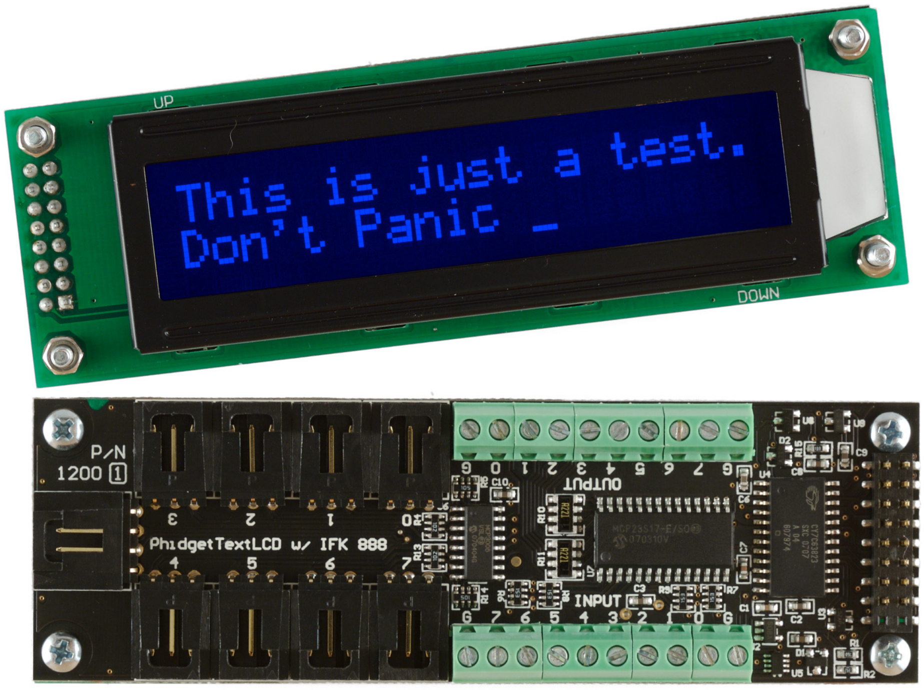
Operating Systems: Windows 2000/XP/Vista, Windows CE, Linux, and Mac OS X

Programming Languages (APIs): VB6, VB.NET, C#.NET, C++, Flash 9, Flex, Java, LabVIEW, Python, Max/MSP, and Cocoa.

Examples: Many example applications for all the operating systems and development environments above are available for download at www.phidgets.com.

What Can the PhidgetTextLCD do?

The PhidgetTextLCD allows you to display messages on a 2-line by 20-character LCD screen. The on-board InterfaceKit 8/8/8 allows you to connect devices to any of 8 analog inputs, 8 digital inputs, or 8 digital outputs. The PhidgetTextLCD provides a generic, convenient way to interface various devices with your PC.



Analog inputs

They are used to measure continuous quantities, such as temperature, humidity, position, pressure, etc. Phidgets offers a wide variety of sensors that can be plugged directly into the board using the cable included with the sensor. Here is a list of sensors currently available:

IR Distance Sensor	IR Reflective Sensor	Vibration Sensor	Light Sensor
Force Sensor	Humidity Sensor	Temperature Sensor	Magnetic Sensor
Rotation Sensor	Voltage Divider	Touch Sensor	Motion Sensor
Mini Joy-Stick	Pressure Sensor	Voltage Sensor	Current Sensor
Slide Sensor			

Non Phidgets Sensors

In addition to Phidgets sensors, any sensor that returns a signal between 0 and 5 volts can be easily interfaced. Here is a list of interesting sensors that can be used with the PhidgetInterfaceKit 8/8/8. Note: these sensors are not "plug & play" like Phidgets sensors.

Manufacturer	Part Number	Description
MSI Sensors	FC21/FC22	Load cells - measure up to 100lbs of force
Humirel	HTM2500VB	Humidity sensors
Measurement Specialties	MSP-300	Pressure sensors - ranges up to 10,000 PSI
Freescale Semiconductor	MPXA/MPXH	Gas Pressure Sensors
Allegro	ACS7 series	Current Sensors - ranges up to 200 Amps
Allegro	A1300 series	Linear Hall Effect Sensors - to detect magnetic fields
Analog	TMP35 TMP36 TMP37	Temperature Sensor
Panasonic	AMN series	Motion Sensors

Note: Most of the above sensors can be bought at www.digikey.com.

Digital Inputs

Digital Inputs can be used to convey the state of push buttons, limit switches, relays (check out our Dual Relay Board), logic levels, etc...

Digital Outputs

Digital Outputs can be used to drive LEDs, solid state relays (have a look at our SSR board), transistors; in fact, anything that will accept a CMOS signal.

Digital outputs can be used to control devices that accept a +5V control signal.

With transistors and some electronics experience, other devices can be controlled, such as buzzers, lights, larger LEDs, relays.

Getting Started

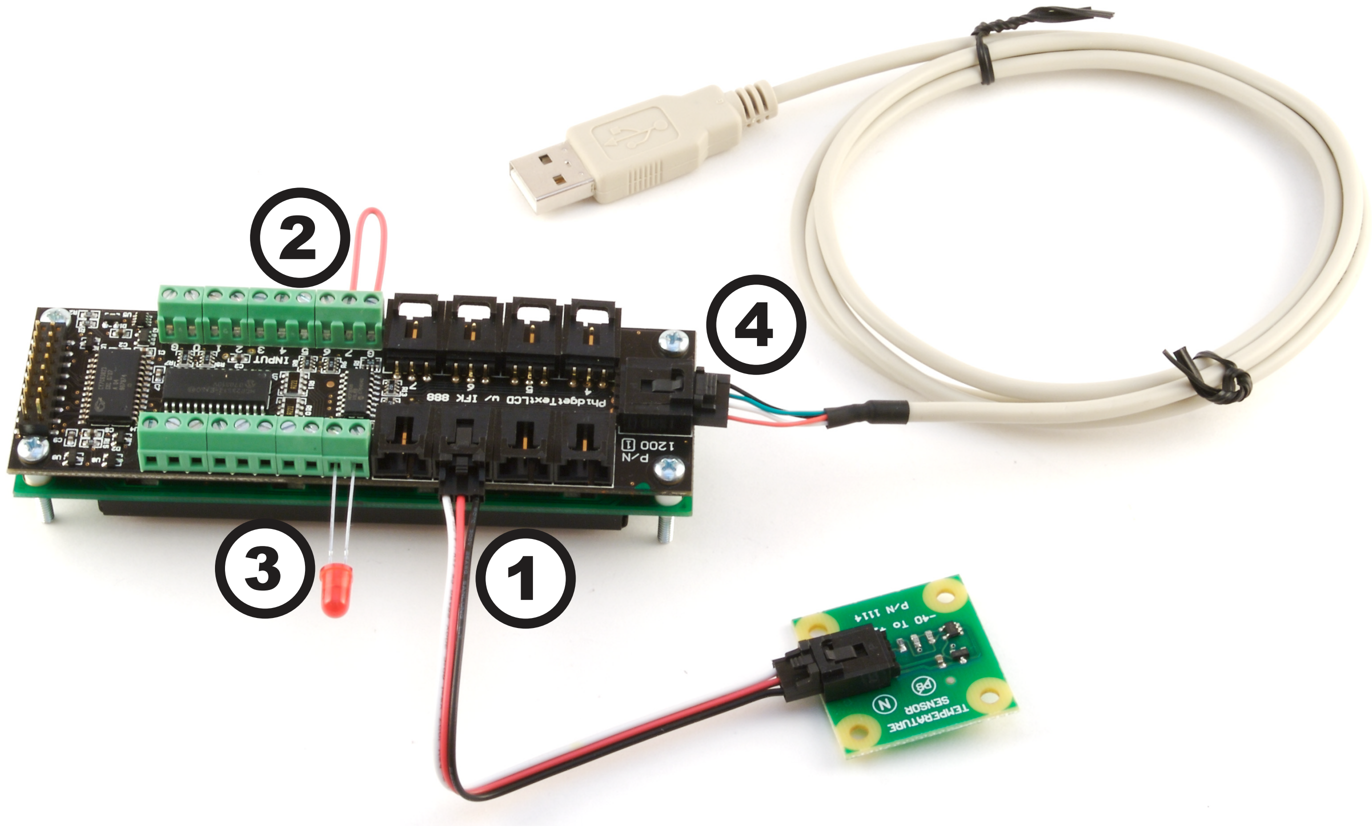
Installing the hardware

The kit contains:

- A PhidgetTextLCD
- A custom USB Cable

You will also need:

- A piece of wire to test the digital inputs
- An LED to test the digital outputs
- An analog sensor to test the analog inputs



1. Connect the analog sensor to any of the analog input ports (labelled 0 to 7) using a Phidgets sensor cable.
2. Connect one end of the wire to a digital input port and the other end to the ground connection terminal.
3. Connect the LED to one of the digital outputs by inserting the longer lead of the LED (anode) into any of the digital outputs (labelled 0 to 7) and the shorter lead (cathode) to ground.
4. Connect the PhidgetTextLCD board to the PC using the USB cable.

Downloading and Installing the software

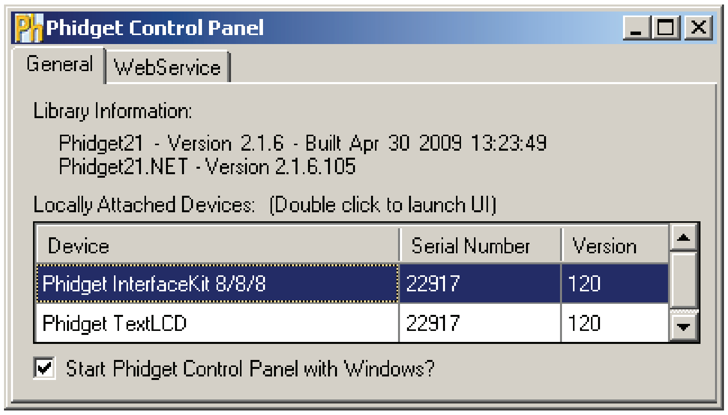
If you are using Windows 2000/XP/Vista

Go to www.phidgets.com >> Drivers

Download and run Phidget21.MSI

You should see the Ph icon on the right hand corner of the Task Bar.

Testing the PhidgetTextLCD Functionality



Phidget Control Panel

General | WebService

Library Information:

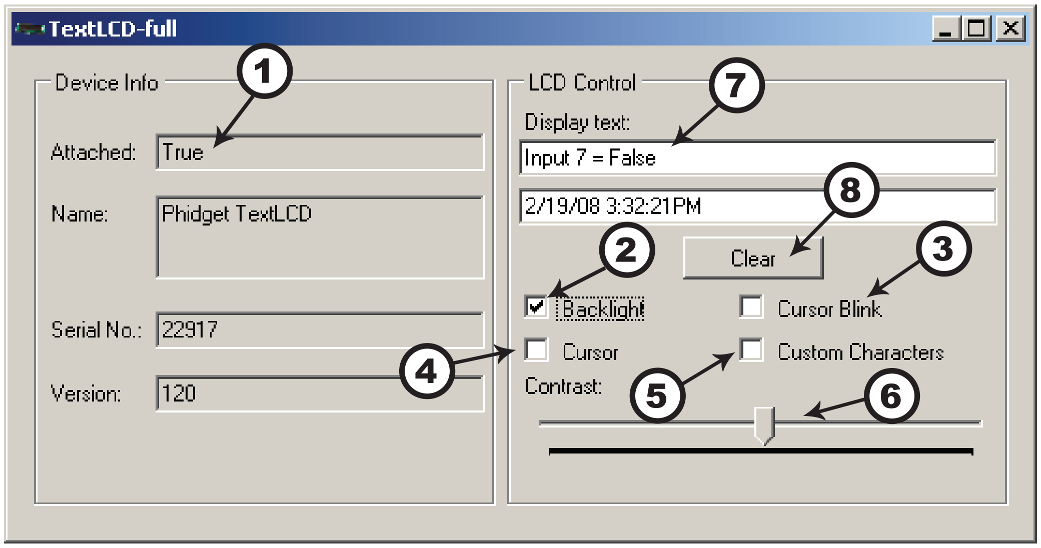
Phidget21 - Version 2.1.6 - Built Apr 30 2009 13:23:49
Phidget21.NET - Version 2.1.6.105

Locally Attached Devices: (Double click to launch UI)

Device	Serial Number	Version
Phidget InterfaceKit 8/8/8	22917	120
Phidget TextLCD	22917	120

☑ Start Phidget Control Panel with Windows?

Double Click on the Ph icon to activate the Phidget Control Panel. Make sure that both the ***PhidgetTextLCD*** and the ***PhidgetInterfaceKit 8/8/8*** are properly connected to your PC.



TextLCD-full

Device Info

Attached: True

Name: Phidget TextLCD

Serial No.: 22917

Version: 120

LCD Control

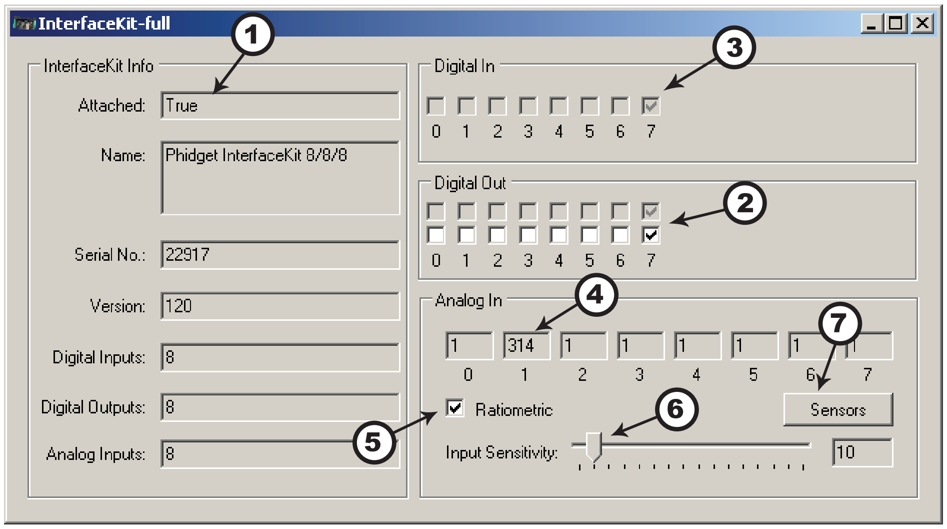
Display text:

Input 7 = False

2/19/08 3:32:21PM

1. Double Click on ***PhidgetTextLCD*** in the Phidget Control Panel to bring up InterfaceKit-full and check that the box labelled Attached contains the word True.
2. Click on the Backlight box. The screen will light up.
3. Click on the Cursor Blink box. A square blinking cursor will appear on the screen.
4. Click on the Cursor box. A “dash” cursor will appear on the screen.
5. Click on the Custom Characters box. The message “Custom.. ” appears on the first line of the LCD screen, followed by some random pictograms on the second line.
6. You can increase or decrease the contrast by using the pointer in the Contrast slider box.
7. You can type a 2 line message to be displayed on the LCD screen.
8. Click on the Clear button to clear the Display Text boxes and the LCD screen.

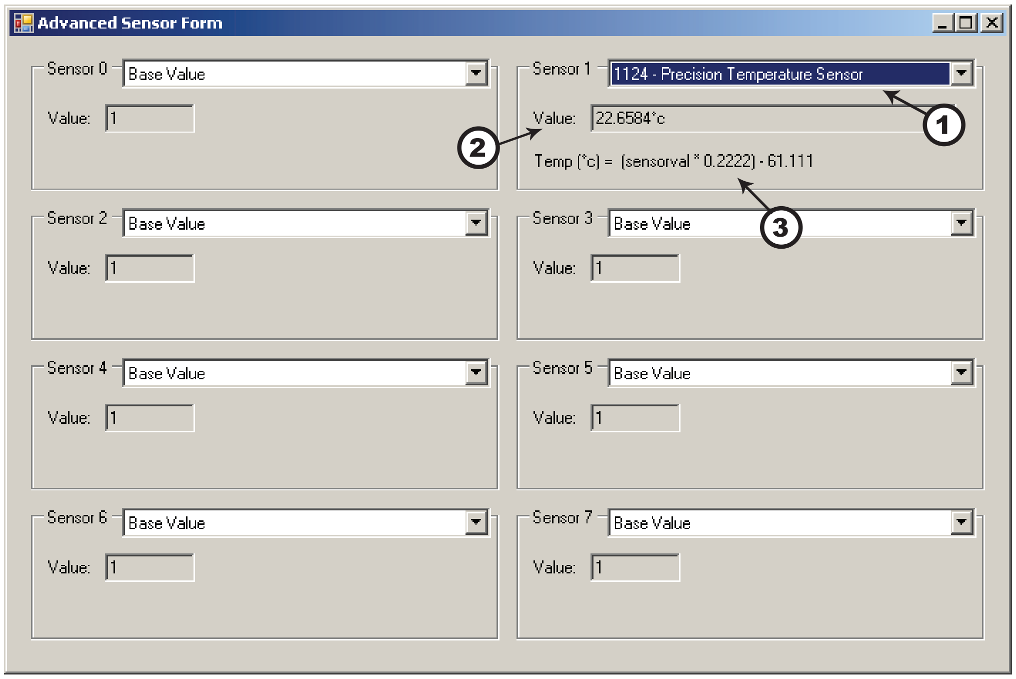
Testing the InterfaceKit 8/8/8 functionality



Attached: True
Name: Phidget InterfaceKit 8/8/8
Serial No.: 22917
Version: 120
Digital Inputs: 8
Digital Outputs: 8
Analog Inputs: 8

Analog In: 0: 1, 1: 314, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1
Input Sensitivity: 10

1. Double Click on InterfaceKit 8/8/8 in the Phidget Control Panel to bring up InterfaceKit-full and check that the box labelled Attached contains the word True.
2. Test the digital output by clicking on the box to turn on the LED. Clicking again will turn the LED off.
3. Test the digital input by disconnecting the wire end connected to the digital input connector. The tick mark in the box will go away. The bottom row shows the status of the request, while the top row displays the status of the digital output as reported by the device.
4. Test the analog input sensor by observing the sensor value as you activate the Phidget sensor.
5. Click on the Ratiometric box if your sensor is ratiometric. Check your sensor product manual if you are not sure.
6. You can adjust the input sensitivity by moving the slider pointer.
7. Click on Sensors to launch the Advanced Sensor Form.



Sensor 0: Base Value — Value: 1
Sensor 1: 1124 - Precision Temperature Sensor — Value: 22.6584°c
Temp (°c) = (sensorval * 0.2222) - 61.111
Sensor 2: Base Value — Value: 1
Sensor 3: Base Value — Value: 1
Sensor 4: Base Value — Value: 1
Sensor 5: Base Value — Value: 1
Sensor 6: Base Value — Value: 1
Sensor 7: Base Value — Value: 1

1. In the drop down menu, select the Sensor you have attached to the analog input port 5 of the1018. In our case we select the 1124 - Precision Temperature Sensor.
2. The ambient temperature sensed by the 1124.
3. Formula used to convert the analog input sensorval into temperature.

Note: Value and formula information will vary from sensor to sensor.

If you are using Mac OS X

Go to www.phidgets.com >> Drivers

- Download Mac OS X Framework
- Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane.
- Make sure that your Phidget is properly attached.
- Double click on the attached Phidget to launch the Example.

If you are using Linux

Go to www.phidgets.com >> Drivers

Download Linux Source

- Have a look at the readme file
- Build Phidget21

The most popular programming languages in Linux are C/C++ and Java.

Notes:

Many Linux systems are now built with unsupported third party drivers. It may be necessary to uninstall these drivers for our libraries to work properly.

Phidget21 for Linux is a user-space library. Applications typically have to be run as root, or udev/hotplug must be configured to give permissions when the Phidget is plugged in.

If you are using Windows Mobile/CE 5.0 or 6.0

Go to www.phidgets.com >> Drivers

Download x86 or ARMV4I, depending on the platform you are using. Mini-itx and ICOP systems will be x86, and most mobile devices, including XScale based systems will run the ARMV4I.

The CE libraries are distributed in .CAB format. Windows Mobile/CE is able to directly install .CAB files.

The most popular languages are C/C++, .NET Compact Framework (VB.NET and C#). A desktop version of Visual Studio can usually be configured to target your Windows Mobile Platform, whether you are compiling to machine code or the .NET Compact Framework.

Programming a Phidget

Phidgets' philosophy is that you do not have to be an electrical engineer in order to do projects that use devices like sensors, motors, motor controllers, and interface boards. All you need to know is how to program. We have developed a complete set of Application Programming Interfaces (API) that are supported for Windows, Mac OS X, and Linux. When it comes to languages, we support VB6, VB.NET, C#.NET, C, C++, Flash 9, Flex, Java, LabVIEW, Python, Max/MSP, and Cocoa.

Architecture

We have designed our libraries to give you the maximum amount of freedom. We do not impose our own programming model on you.

To achieve this goal we have implemented the libraries as a series of layers with the C API at the core surrounded by other language wrappers.

Libraries

The lowest level library is the C API. The C API can be programmed against on Windows, CE, OS X and Linux. With the C API, C/C++, you can write cross-platform code. For systems with minimal resources (small computers), the C API may be the only choice.

The Java API is built into the C API Library. Java, by default is cross-platform - but your particular platform may not support it (CE).

The .NET API also relies on the C API. Our default .NET API is for .NET 2.0 Framework, but we also have .NET libraries for .NET 1.1 and .NET Compact Framework (CE).

The COM API relies on the C API. The COM API is programmed against when coding in VB6, VBScript, Excel (VBA), Delphi and Labview.

The ActionScript 3.0 Library relies on a communication link with a PhidgetWebService (see below). ActionScript 3.0 is used in Flex and Flash 9.

Programming Hints

- Every Phidget has a unique serial number - this allows you to sort out which device is which at runtime. Unlike USB devices which model themselves as a COM port, you don't have to worry about where in the USB bus you plug your Phidget in. If you have more than one Phidget, even of the same type, their serial numbers enable you to sort them out at runtime.
- Each Phidget you have plugged in is controlled from your application using an object/handle specific to that phidget. This link between the Phidget and the software object is created when you call the .OPEN group of commands. This association will stay, even if the Phidget is disconnected/reattached, until .CLOSE is called.
- The Phidget APIs are designed to be used in an event-driven architecture. While it is possible to poll them, we don't recommend it. Please familiarize yourself with event programming.

Networking Phidgets

The PhidgetWebService is an application written by Phidgets Inc. which acts as a network proxy on a computer. The PhidgetWebService will allow other computers on the network to communicate with the Phidgets connected to that computer. ALL of our APIs have the capability to communicate with Phidgets on another computer that has the PhidgetWebService running.

The PhidgetWebService also makes it possible to communicate with other applications that you wrote and that are connected to the PhidgetWebService, through the PhidgetDictionary object.

Documentation

Programming Manual

The Phidget Programming Manual documents the Phidgets software programming model in a language and device unspecific way, providing a general overview of the Phidgets API as a whole.

Getting Started Guides

We have written Getting Started Guides for most of the languages that we support. If the manual exists for the language you want to use, this is the first manual you want to read. The Guides can be found under Programming and are listed under the appropriate language.

API documentation

We maintain API references for COM (Windows), C (Windows/Mac OSX/Linux), Action Script, .Net and Java. These references document the API calls that are common to all Phidgets. These API References can be found under Programming and are listed under the appropriate language. To look at the API calls for a specific Phidget, check its Product Manual.

Code Samples

We have written sample programs to illustrate how the APIs are used.

Due to the large number of languages and devices we support, we cannot provide examples in every language for every Phidget. Some of the examples are very minimal, and other examples will have a full-featured GUI allowing all the functionality of the device to be explored. Most developers start by modifying existing examples until they have an understanding of the architecture.

Go to Programming to see if there are code samples written for your device. Find the language you want to use and click on the magnifying glass besides “Code Sample”. You will get a list of all the devices for which we wrote code samples in that language.

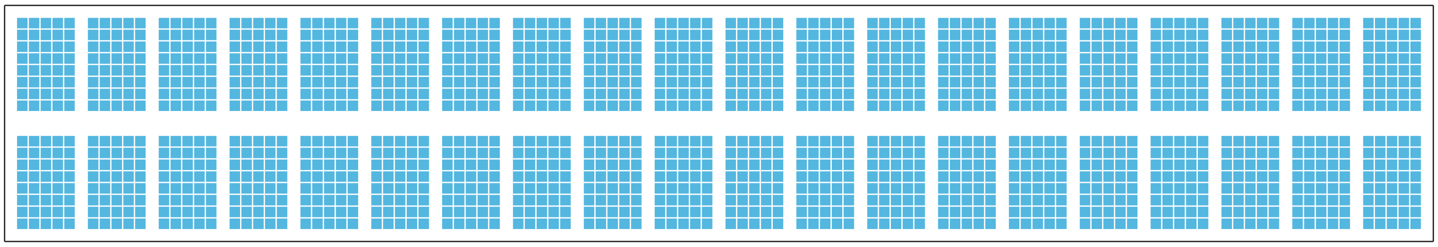
Support

- Call the support desk at 1.403.282.7335 8:00 AM to 5:00 PM Mountain Time (US & Canada) - GMT-07:00
- E-mail support@phidgets.com

Technical Section - TextLCD

LCDs

Liquid Crystal Displays are display devices used to convey information through arrangements of pixels. Graphic and Text LCDs are the most common types available for electronic products. The PhidgetTextLCD's display is configured as a 2X20 LCD (2 lines high, 20 characters per line) with each character having an arrangement of 40 pixels (8 pixels high by 5 pixels wide).



2X20 LCD Character Arrangement

Special Characters in the ASCII Standard Set

The PhidgetTextLCD displays full text strings set in software. Since text characters are defined from the ASCII standard library, other ASCII standard set characters and glyphs can also be sent to the text LCD. This can be done easily by using unicode characters within your text string. In C#, this may look something like this:

```
tLCD.rows[0].DisplayString = "Apple starts with \u0041";
```

In this example, the string \u indicates that a unicode character follows, and the unicode character 0041 (which references the hexadecimal character code 0x41) represents the capital letter A. After the LCD converts the unicode character, the above example would cause the LCD screen to read Apple starts with A. A chart of all ASCII standard set character codes is available in the Appendix at the end of this manual.

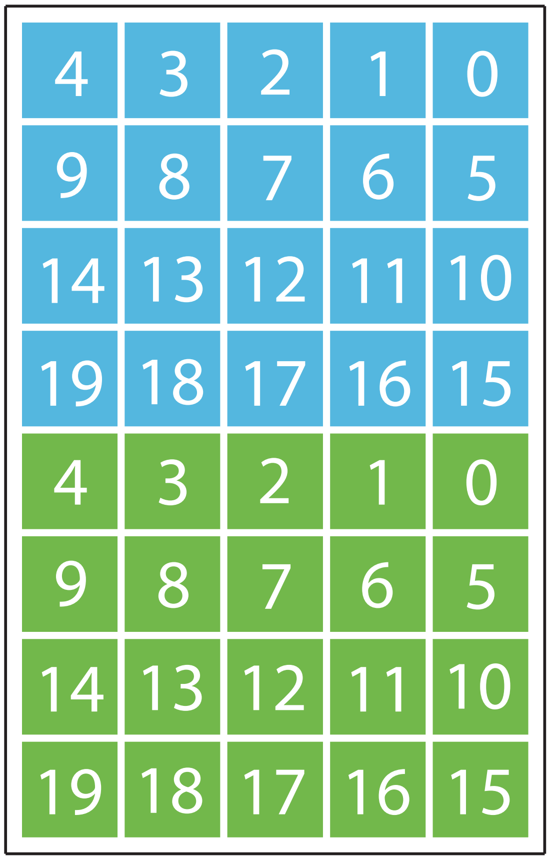
Custom Characters

Custom characters can also be generated for the PhidgetTextLCD. A custom character can be any arrangement of pixels within the space alotted for a single character. Single characters are made up of pixels arranged in a grid 5 pixels wide by 8 pixels high. Once generated, custom characters can be stored in any one of eight volatile memory locations on the PhidgetTextLCD, and can be recalled with a simple API command from software.

When custom characters are designed, a formula is used to change the pixel design into a pair of numerical values. The first value relates to the design of the top 4 rows of the character, and the second value relates to the design of the bottom 4 rows of the character. Unlike the unicode characters used in the Special Characters section above, the calculated number is not in hexadecimal format but is an integer value up to six characters in length.

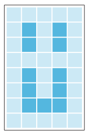
Custom Characters (cont'd)

The calculation for custom characters can be done by hand, or can be completed for you by using the form available at www.phidgets.com/documentation/customchar.html. Done by hand, each integer value represents the sum of two to the power of each individual on-pixel's location within that integer-value's half of the character. Pixels not turned on are valued at zero. For example, a custom character happy-face with pixels 6, 8, 11 and 13 in the upper half turned on, pixels 1, 3, 6, 8, 11, 12 and 13 in the lower half turned on, and all other pixels turned off, would result in the following integer values:



5X8 LCD Character Pixel Arrangement

$$VAL_{UPPER} = 2^6 + 2^8 + 2^{11} + 2^{13} = 10560$$
$$VAL_{LOWER} = 2^1 + 2^3 + 2^6 + 2^8 + 2^{11} + 2^{12} + 2^{13} = 14666$$



These two values are then stored in one of eight memory locations (CG-RAM 0 to 7) on the PhidgetTextLCD by using the Set Custom Character method in software. In C#, this may look something like this:

```
tLCD.customCharacters[0].setCustomCharac-
ter(10560, 14666);
```

Once stored, characters can be recalled into a text string by either using the unicode value for the location as referenced in the ASCII chart (Appendix A) or by using the String Code method from the API. Examples in C# of both methods are shown below:

```
tLCD.rows[0].DisplayString = "I am happy \u0008";
```

```
tLCD.rows[0].DisplayString = "I am happy " +
tLCD.customCharacters[0].StringCode;
```

Analog Inputs

Using the Analog Inputs with Sensors provided by Phidgets

Analogs Inputs are used to interface many different types of sensors. Each Analog Input provides power (Nominal +5VDC), ground, and an analog voltage return wire driven by the sensor to some voltage. The PhidgetInterfaceKit continuously measures this return voltage and reports it to the application.

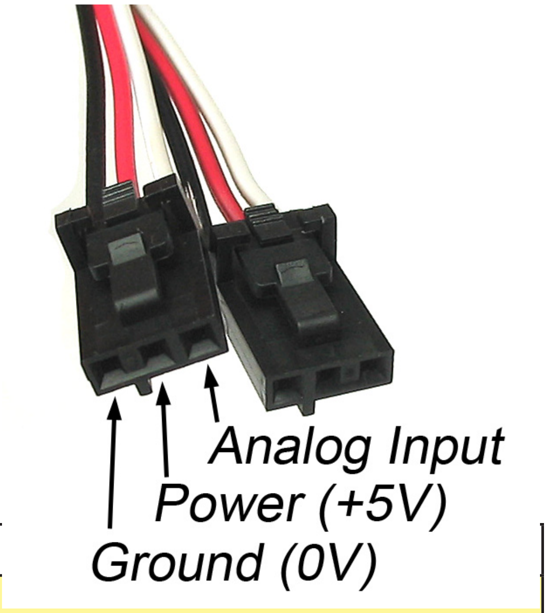
Analog Inputs are used to measure continuous quantities, such as temperature, humidity, position, pressure, etc. Phidgets offers a wide variety of sensors that can be plugged directly into the board using the cable included with the sensor.

Using the Analog Inputs with your own sensors

For users who wish to interface their own sensors, we describe the Analog Inputs here.

Mechanical

Each Analog Input uses a 3-pin, 0.100 inch pitch locking connector. Pictured here is a plug with the connections labeled. The connectors are commonly available - refer to the Table below for manufacturer part numbers.



Cable Connectors

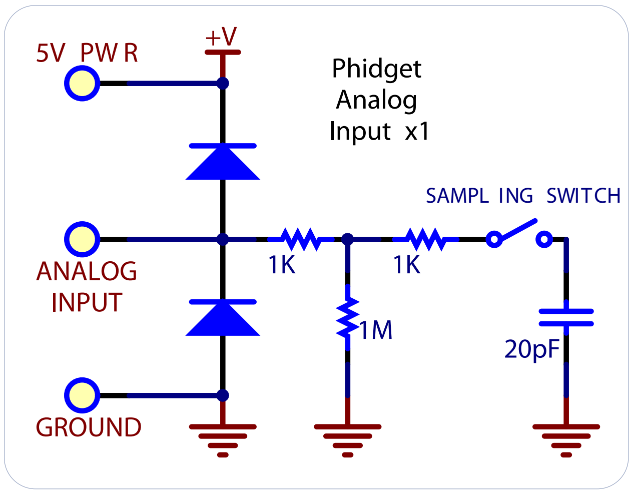
Manufacturer	Part Number	Description
Molex	50-57-9403	3 Position Cable Connector
Molex	16-02-0102	Wire Crimp Insert for Cable Connector
Molex	70543-0002	3 Position Vertical PCB Connector
Molex	70553-0002	3 Position Right-Angle PCB Connector (Gold)
Molex	70553-0037	3 Position Right-Angle PCB Connector (Tin)
Molex	15-91-2035	3 Position Right-Angle PCB Connector - Surface Mount

Note: Most of the above components can be bought at www.digikey.com

Electrical

Maximum total current consumed by all Analog Inputs should be limited to 400mA.

The analog measurement is represented in the software through the SensorValue as a value between 0 and 1000. A sensor value of 1 unit represents a voltage of approximately 5 millivolts. The RawSensorValue property brings out a 12-bit value (0-4095) for users who require maximum accuracy. Please note that the sampling is actually done with an oversampled 10-bit ADC, but reported as a 12-bit value to allow future expansion.



Ratiometric Configuration

The group of Analog Inputs can be collectively set to Ratiometric mode from software using the Ratiometric property. If you are using a sensor whose output changes linearly with variations in the sensor's supply voltage level, it is said to be ratiometric. Most of the sensors sold by Phidgets are ratiometric (this is specified in the manual for each sensor).

Setting Ratiometric causes the reference to the internal Analog to Digital Converter to be set to the power supply voltage level. When Ratiometric is enabled, the maximum voltage returned on the Analog Input should be the +5V nominal power provided by the PhidgetInterfaceKit.

Non-Ratiometric Configuration

If Ratiometric is false, the ADC reference is set to a 5.0V 0.5% stable voltage reference. The maximum voltage returned on the Analog Input should be maximum 5.0V. Please note that the Analog Input power supply voltage is not affected by the setting of the Ratiometric property.

Factors that can affect Accuracy

High Output Impedance - Sensors that have a high output impedance will be distorted by the 900K input impedance of the Analog Input. If your output impedance is high, it is possible to correct for this distortion to some extent in your software application.

Power Consumption - Sensor cables have some resistance, and the power consumption of the sensor will cause the sensor to have a slightly different ground from the Analog Input on the PhidgetInterfaceKit. The more power consumed by the sensor, and the longer the sensor cable, the more pronounced this effect will be.

Intrinsic Error In Sensors - For many sensors, the error is quite predictable over the life of the sensor, and it can be measured and calibrated out in software.

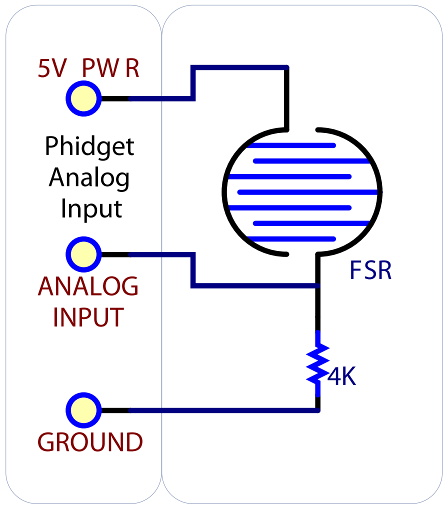
Non-Ratiometric Configuration - Voltage Reference error. The 5.0VDC voltage reference is accurate to 0.5%. This can be a significant source of error in some applications, but can be easily measured and compensated for.

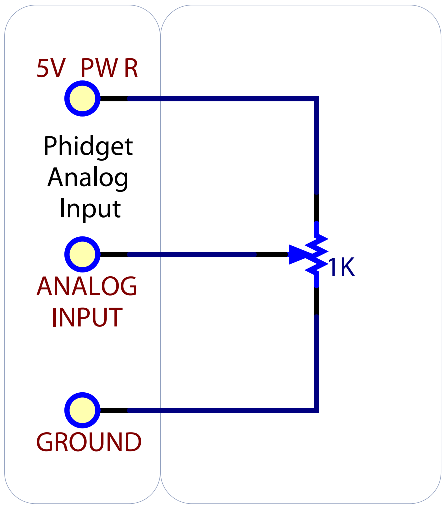
Connecting non-Phidget devices to the Analog Inputs

Here are some circuit diagrams that illustrate how to connect various non Phidgets devices to the analog inputs on your Phidget.

Sensing the value of a variable resistance sensor

In this diagram, an FSR (Force Sensitive Resistor) is shown.

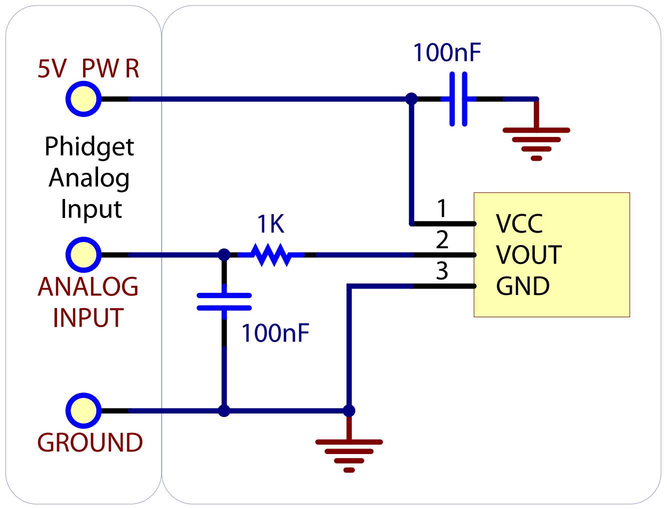




Sensing the position of a potentiometer

Interfacing to an arbitrary sensor

Note the use of power supply decoupling and the RC Filter on the output. The RC filter also prevents VOUT from oscillating on many sensors



Non Phidgets Sensors

In addition to Phidgets sensors, any sensor that returns a signal between 0 and 5 volts can be easily interfaced. Here is a list of interesting sensors that can be used with the PhidgetInterfaceKit 8/8/8. Note: these sensors are not "plug & play" like the sensors manufactured by Phidgets.

Analog Sensors		
Manufacturer	**Part Number**	**Description**
MSI Sensors	FC21/FC22	Load cells - measure up to 100lbs of force
Humirel	HTM2500VB	Humidity sensors
Measurement Specialties	MSP-300	Pressure sensors - ranges up to 10,000 PSI
Freescale Semiconductor	MPXA/MPXH	Gas Pressure Sensors
Allegro	ACS7 series	Current Sensors - ranges up to 200 Amps
Allegro	A1300 series	Linear Hall Effect Sensors - to detect magnetic fields
Analog	TMP35 TMP36 TMP37	Temperature Sensor
Panasonic	AMN series	Motion Sensors
Honeywell	FS01, FS03	Small, accurate Piezo-resistive load cells

Note: Most of the above components can be bought at www.digikey.com

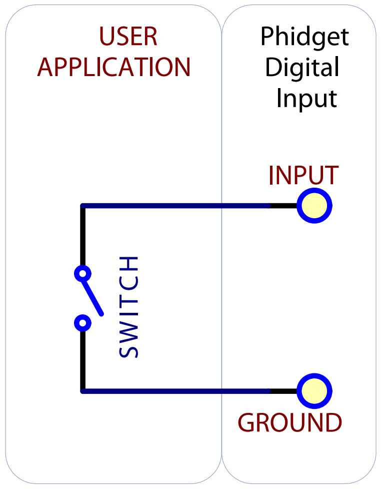
Digital Inputs

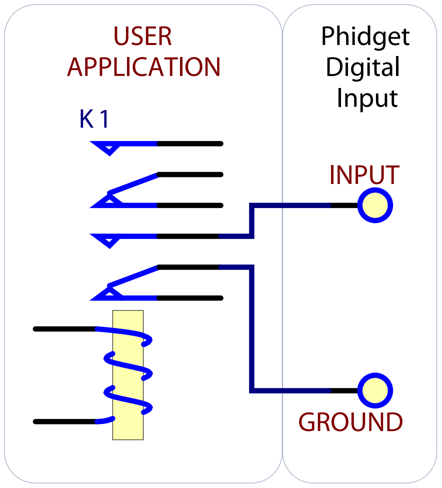
Using the Digital Inputs

Here are some circuit diagrams that illustrate how to connect various devices to the digital inputs on your Phidget.

Wiring a switch to a Digital Input

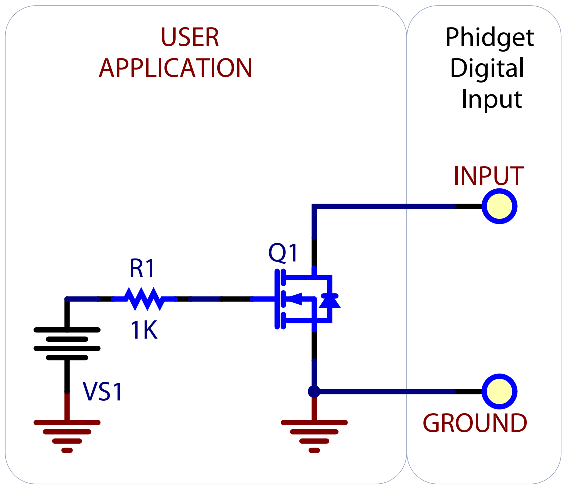
Closing the switch causes the digital input to report TRUE.





Monitoring the position of a relay

The relay contact can be treated as a switch, and wired up similarly. When the relay contact is closed, the Digital Input will report TRUE.



Detecting an external Voltage with an N-Channel MOSFET

A MOSFET can be used to detect the presence of an external voltage. The external voltage will turn on the MOSFET, causing it to short the Digital Input to Ground.

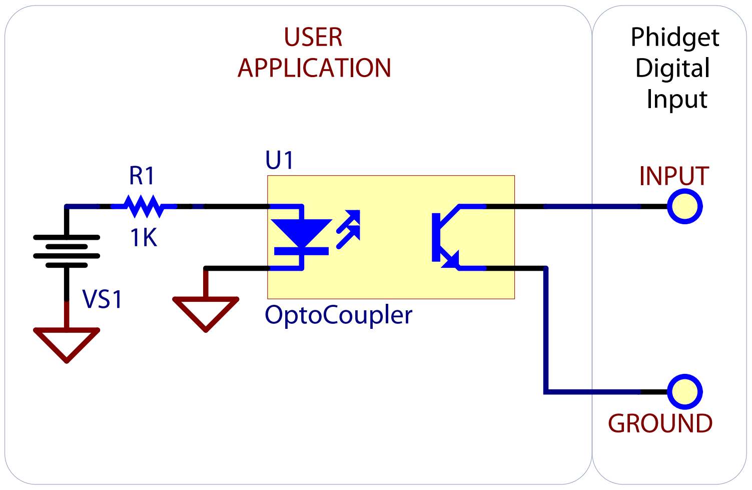
If the MOSFET is conducting > 270uA, the Digital Input is guaranteed to report TRUE.

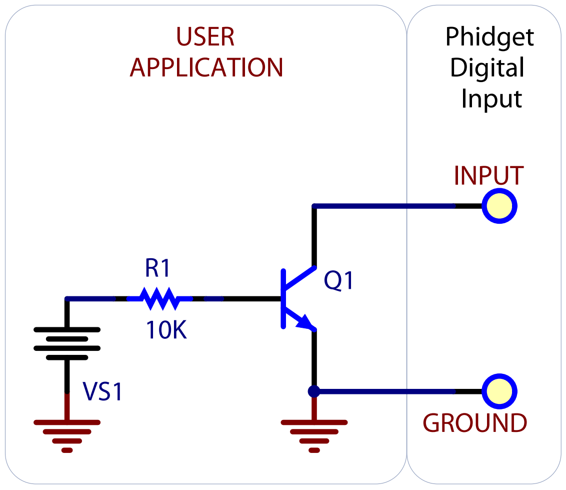
If the MOSFET is conducting < 67uA, the Digital Input is guaranteed to report FALSE.

The voltage level required to turn on the MOSFET depends on the make of of MOSFET you are using. Typical values are 2V-6V.

Isolating a Digital Input with an Optocoupler

When driving current through the LED, the Digital Input will report TRUE. The amount of current required will depend on the optocoupler used. Design to sink at least 270uA to cause the digital input to report TRUE, and less than 67uA to report FALSE.





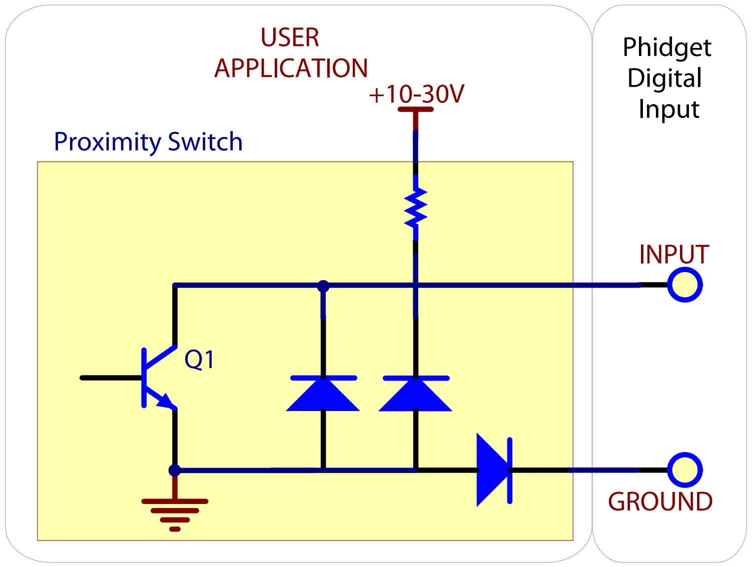
Detecting an external Voltage with an NPN Transistor

This circuit can be used to measure if a battery is connected, or if 12V (for example) is on a wire.

By designing to have Collector-Emitter current > 270uA, the digital input will report TRUE.

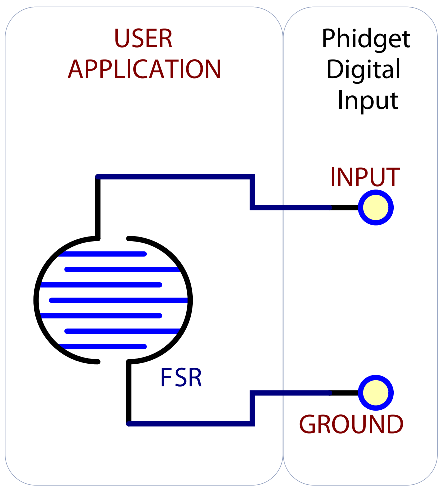
Using a Capacitive or Inductive Proximity Switch

Capacitive proximity switches can detect the presence of nearby non-metallic objects, whereas inductive proximity switches can detect only the presence of metallic objects. To properly interface one of these proximity switches to the digital inputs, a 3-wire proximity switch is required, as well as an external power supply.



We have checked the following switches

Manufacturer	Web Page	Capacitive Part No	Inductive Part No
Automation Direct	www.automationdirect.com	CT1 Series	AM1 Series



from Automation Direct to verify that they work with the Digital Inputs. Similar capacitive or inductive proximity switches from other manufacturers should work just as well.

Using an FSR or other variable resistor as a switch

The digital inputs can be easily wired to use many variable resistors as switches.

If the resistance falls below 3.75k Ohms, the Digital Input will go TRUE.

If the resistance rises above 75k Ohms, the Digital Input will go FALSE.

Functional Block Diagram

The digital inputs have a built in 15K pull-up resistor. By connecting external circuitry, and forcing the input to Ground, the Digital Input in software will read as TRUE. The default state is FALSE - when you have nothing connected, or your circuitry (switch, etc) is not pulling the input to ground.

Digital Input Hardware Filter

There is built-in filtering on the digital input, to eliminate false triggering from electrical noise. The digital input is first RC filtered by a 15K/100nF node, which will reject noise of higher frequency than 1Khz. This filter generally eliminates the need to shield the digital input from inductive and capacitive coupling likely to occur in wiring harnesses.

Digital Input Hysteresis

The digital input has hysteresis - that is, it will hold it’s current state (false or true), unless a large change occurs. To guarantee FALSE, the digital input must be at least 3.75V, and to guarantee TRUE, the digital input must be less than 1.25V.

Digital Input Sampling Characteristics

The state of the digital inputs are reported back to the PC periodically. During this sampling period, if a digital input was true for greater than 4.0ms, the digital input is guaranteed to be reported as true in software. This makes the digital input much more sensitive to reporting TRUE state, and makes it useful to watch for short events. Any Digital Input True events of

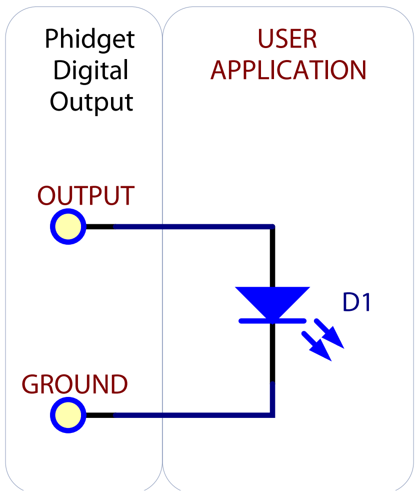
Digital Outputs

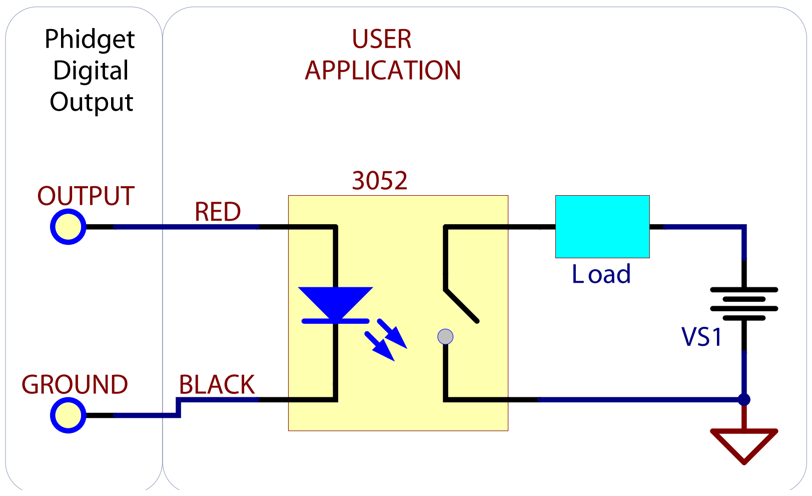
Using the Digital Outputs

Here are some circuit diagrams that illustrate how to connect various devices to the digital outputs on your Phidget.

Driving an LED with the Digital Output

Connecting an LED to a digital output is simple. Wire the anode to a digital output labeled 0 to 7 on the Interface Kit, and the cathode to a supplied ground, labeled G.



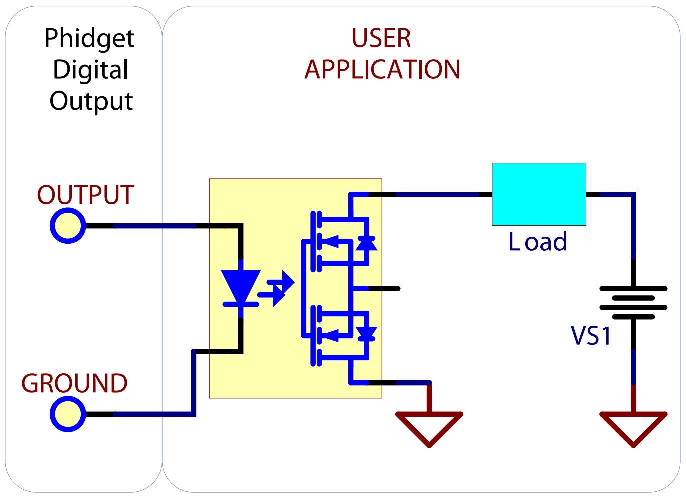


Using a 3052 SSR Board with a Digital Ouptut

Setting the digital output to true causes the output of the 3052 to turn on. This can be used to control AC or DC devices. The load can also be switched with the 3052 on the high side. High side switching is helpful for powering more complicated circuitry that cannot tolerate having multiple grounds.

Isolating a Digital Output with a MOSFET based SSR

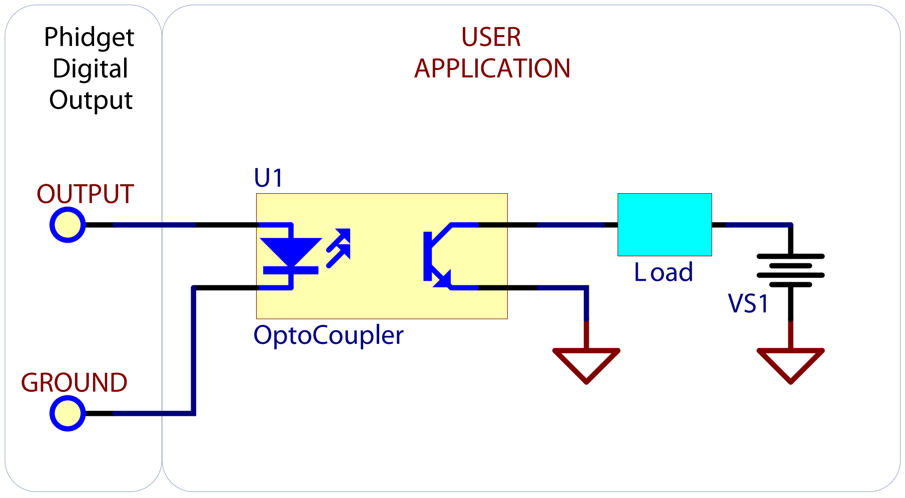
It's possible to wire up your own Solid State Relay to the digital output. MOSFET based SSRs have the advantage that they can be understood as being a simple switch. There are many other types of SSRs that are more suitable for controlling higher power, higher voltage AC devices that can also be controlled in the same fashion.

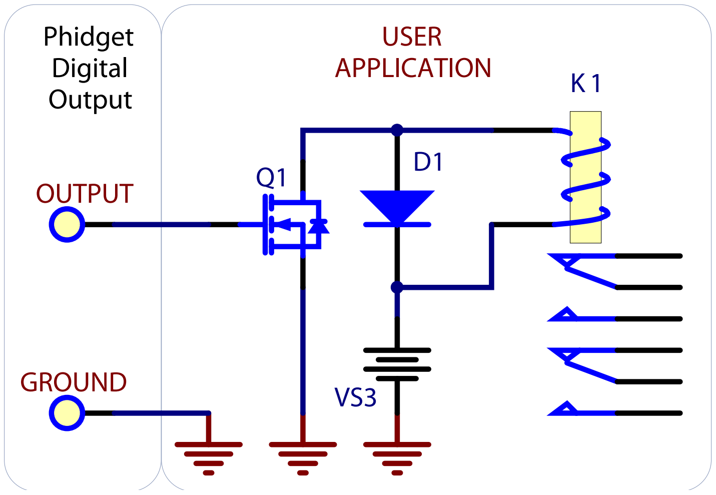


Isolating a Digital Output with an Optocoupler

In some applications, particularly where there is a lot of electrical noise (automotive), or where you want maximum protection of the circuitry (interactive installations, kiosks), electrical isolation buys you a huge margin of protection.

Driving the LED causes the output transistor to sink current. The maximum current through the transistor will depend in part on the characteristics of the optocoupler.





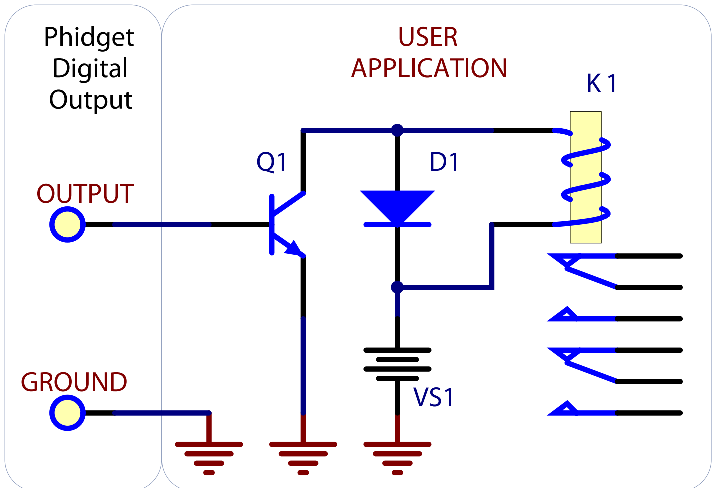
Controlling a relay with a N-Channel MOSFET

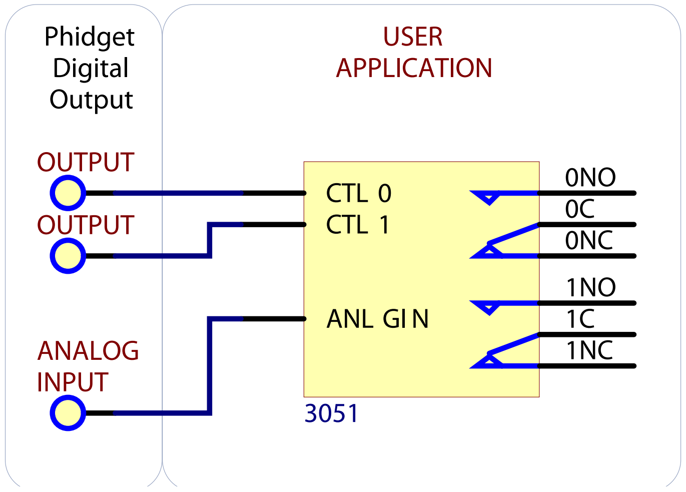
A inexpensive mosfet and flyback diode can be used to control larger loads - relays for example - directly from the digital output.

Be sure to use a Logic-Level MOSFET so that the +5V Digital Output is able to turn it on.

Controlling a relay with a NPN transistor

This circuit is very similar to the N-channel mosfet - but you may already have NPN transistors on hand.



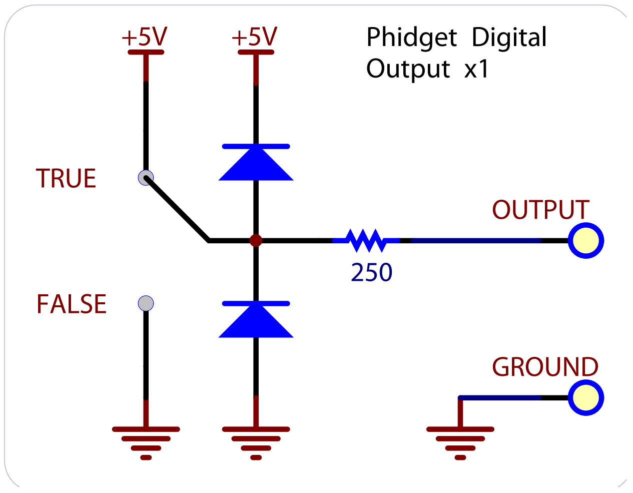


Using a 3051 Dual Relay Board with one or two Digital Outputs

The 3051 Dual Relay Board is designed to be used with the PhidgetInterfaceKit 8/8/8. An Analog Input can be used to supply power to the relays, and one or two digital outputs used to control the relays. The 3051 is a good option if you need a couple relays in your project.

Functional Block Diagram

The 250 ohm resistance is internal to the PhidgetInterfaceKit 8/8/8, and limits the current that can flow through the output. This is intended to protect the device from being damaged if there is a short to ground or if an LED is used. The output is intended to drive TTL or CMOS inputs; it is not designed to provide power to an external circuit.



Ground Protection

Ground terminals on the InterfaceKit share a common ground with USB ground. Because they are not internally isolated, these terminals will expose the USB ground potential of the PC to which they are connected. Be sure you are completely familiar with any circuit you intend to connect to the InterfaceKit before it is connected. If a reverse voltage or dangerously high voltage is applied to the input or output terminals, damage to the Phidget or the PC may result.

API InterfaceKit 8/8/8

We document API Calls specific to the 1018. Functions common to all Phidgets are not covered here. This section is deliberately generic - for calling conventions in a specific language, refer to that language's API manual.

Functions

int InputCount() [get] : Constant = 8

Returns the number of digital inputs supported by this PhidgetInterfaceKit.

bool InputState(int InputIndex) [get]

Returns the state of a particular digital input. Digital inputs read True where they are activated and false when they are in their default state.

int OutputCount() [get] : Constant = 8

Returns the number of digital outputs supported by this PhidgetInterfaceKit.

bool OutputState (int OutputIndex) [get,set]

Sets/returns the state of a digital output. Setting this to true will activate the output, False is the default state. Reading the OutputState immediately after setting it will not return the value set - it will return the last state reported by the Phidget.

int SensorCount() [get] : Constant = 8

Returns the number of sensors (Analog Inputs) supported by this PhidgetInterfaceKit. Note that there is no way of determining is a sensor is attached, and what sensor is attached.

int SensorValue(int SensorIndex) [get]

Returns the sensed value of a particular Analog Input. SensorValue varies between 0-1000, corresponding to the 0-5V input range of the Analog Input.

If you are using an Analog Sensor from Phidgets Inc., it's manual will specify the formula used to convert SensorValue into the measured property.

int SensorRawValue (int SensorIndex) [get]

Returns the full resolution of the Analog Input. This is a more accurate version of SensorValue. The valid range is 0-4095. Note however that the analog outputs on the Interface Kit 8/8/8 are only 10-bit values and this value represents an oversampling to 12-bit.

double SensorChangeTrigger (int SensorIndex) [get,set]

Returns the change trigger for an analog input. This is the amount that an inputs must change between successive SensorChangeEvents. This is based on the 0-1000 range provided by getSensorValue. This value is by default set to 10 for most Interface Kits with analog inputs.

bool Ratiometric() [get,set]

Sets/returns the state of Ratiometric. Ratiometric = true configures the Analog Inputs to measure w.r.t VCC (nominal 5V). Ratiometric = false configures the Analog Inputs to measure w.r.t an internal precision 5V reference. Ratiometric is not updated from the Phidget. It is recommended to explicitly set Ratiometric when the Interfacekit is opened.

Events

OnInputChange(int InputIndex, bool State) [event]

An event that is issued when the state of a digital input changes.

OnOutputChange(int OutputIndex, bool State), [event]

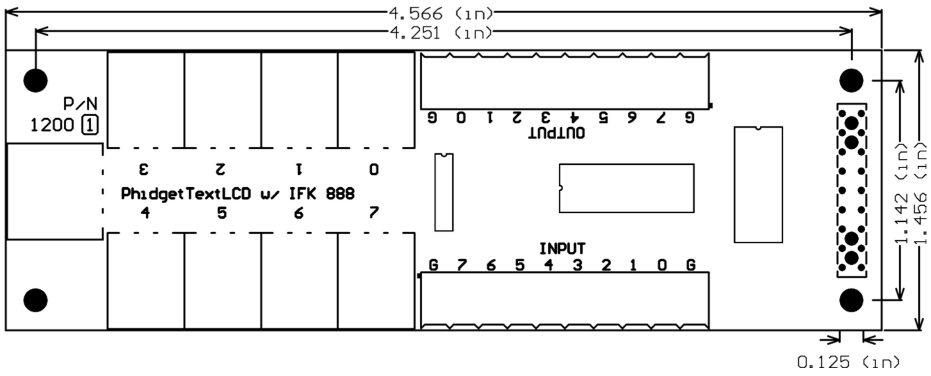
An event that is issued when the state of a digital output changes.

OnSensorChange(int SensorIndex, int SensorValue), [event]

An event that is issued when the returned value from a sensor (Analog Input) varies by more than the SensorChangeTrigger property.

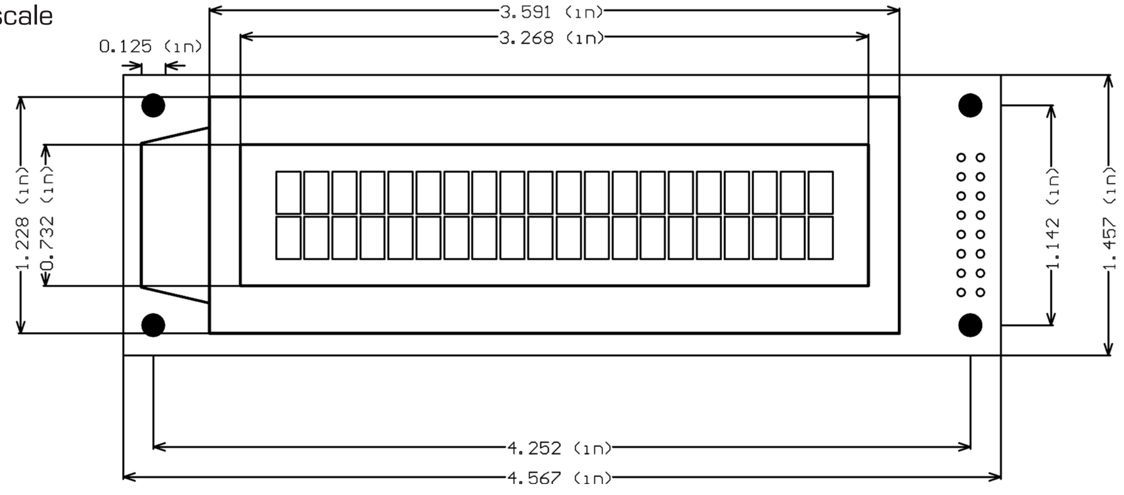
Mechanical Drawing 8/8/8

1:1 scale



Mechanical Drawing LCD

1:1 scale



Note: When printing the mechanical drawing, "**Page Scaling**" in the Print panel must be set to "**None**" to avoid re-sizing the image.

Device Specifications

Analog Input Impedance	900K ohms
Analog Input 5V Reference Error	Max 0.5%
Digital Output Series Resistance	300 ohms
Digital Input Pull-Up Resistance	15K ohms
Analog Input Update Rate	~65 samples / second
Digital Output Update Rate	~125 samples / second
Digital Input Update Rate	~125 samples / second
Digital Input Recommended Wire Size	16 - 26 AWG
Digital Output Recommended Wire Size	16 - 26 AWG
Digital Input Wire Stripping	5-6mm strip
USB-Power Current Specification	Max 500mA
Quiescent Current Consumption	13mA
Available External Current (source)	487mA

Product History

Date	Product Revision	Comment
July 2005	DeviceVersion 120	Product Release

Appendix A - ASCII Standard Character Set with Katakana Extension

		Higher 4-bit (D4 to D7) of Character Code (Hexadecimal)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Lower 4-bit (D0 to D3) of Character Code (Hexadecimal)	0	CG RAM (1)			0	@	P	`	p				ー	タ	ミ	α	p
	1	CG RAM (2)		!	1	A	Q	a	q			。	ア	チ	ム	ä	q
	2	CG RAM (3)		"	2	B	R	b	r			「	イ	ツ	メ	β	θ
	3	CG RAM (4)		#	3	C	S	c	s			」	ウ	テ	モ	ε	∞
	4	CG RAM (5)		$	4	D	T	d	t			、	エ	ト	ヤ	μ	Ω
	5	CG RAM (6)		%	5	E	U	e	u			・	オ	ナ	ユ	σ	ü
	6	CG RAM (7)		&	6	F	V	f	v			ヲ	カ	ニ	ヨ	ρ	Σ
	7	CG RAM (8)		'	7	G	W	g	w			ァ	キ	ヌ	ラ	g	π
	8	CG RAM (1)		(	8	H	X	h	x			ィ	ク	ネ	リ	√	x̄
	9	CG RAM (2)		)	9	I	Y	i	y			ゥ	ケ	ノ	ル	$^{-1}$	y
	A	CG RAM (3)		*	:	J	Z	j	z			ェ	コ	ハ	レ	j	千
	B	CG RAM (4)		+	;	K	[	k	{			ォ	サ	ヒ	ロ	^{x}	万
	C	CG RAM (5)		,	<	L	¥	l	|			ャ	シ	フ	ワ	¢	円
	D	CG RAM (6)		-	=	M	]	m	}			ュ	ス	ヘ	ン	£	÷
	E	CG RAM (7)		.	>	N	^	n	→			ョ	セ	ホ	゛	ñ	
	F	CG RAM (8)		/	?	O	_	o	←			ッ	ソ	マ	゜	ö	█