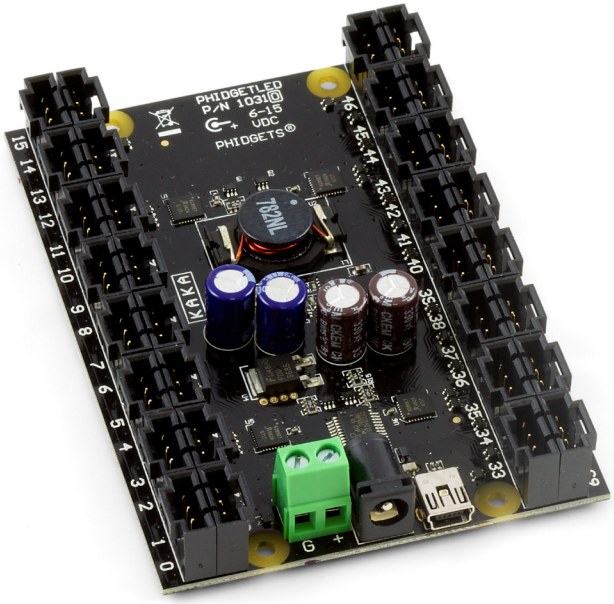
Product Manual

1031 - PhidgetLED-64 Advanced



Phidgets 1031 - Product Manual

For Board Revision 0

Contents

Product Features

- Controls 64 Light Emitting Diodes - any color.
- Each LED can be turned on and off, and its brightness controlled independently without using resistors.
- Turn LED on/off: up to 3000 updates/second
- Change LED brightness: up to 700 updates/second
- Available voltage for each LED: from 1.7 to 5 Volts
- Current controlled LED outputs: from 0.3mA and 80mA
- Ships with a 12VDC 2Amp power supply

Programming Environment

Operating Systems: Windows 2000/XP/Vista/7, 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 >> Programming.

Connection

The board connects directly to a computer's USB port.

Getting Started

Checking the Contents

You should have received:

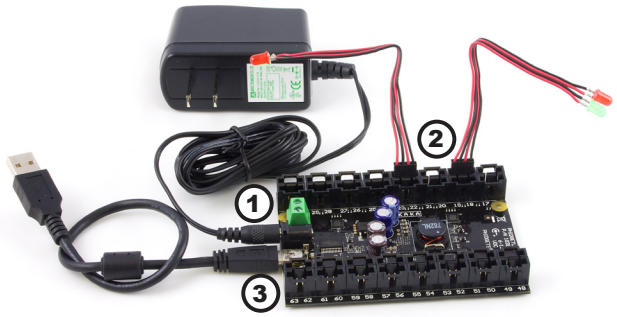
- A PhidgetLED-64 Advanced board
- A 12VDC 2 Amps power supply
- 16 4-wire cables (for LEDs)
- A Mini-USB Cable

In order to test your new Phidget you will also need:

- some LEDs

Connecting all the pieces

1. Connect the power supply to the PhidgetLED using the barrel connector.
2. Cut one of the 4-wire cables to an appropriate length. Strip the wire ends and solder 2 LEDs to each cable. The longer lead of the LED (the anode) is connected to the red wire, and the shorter lead of the LED (the cathode, also marked with a notch in the base of the plastic case) is connected to the black wire. Insert the end of the 4-wire LED cable into the board connector.
3. Connect the PhidgetLED to your computer using the Mini-USB cable.



Testing Using Windows 2000/XP/Vista

Downloading the Phidgets drivers

Make sure that you have the current version of the Phidget library installed on your PC. If you don't, do the following:

Go to www.phidgets.com >> Drivers

Download and run Phidget21 Installer (32-bit, or 64-bit, depending on your PC)

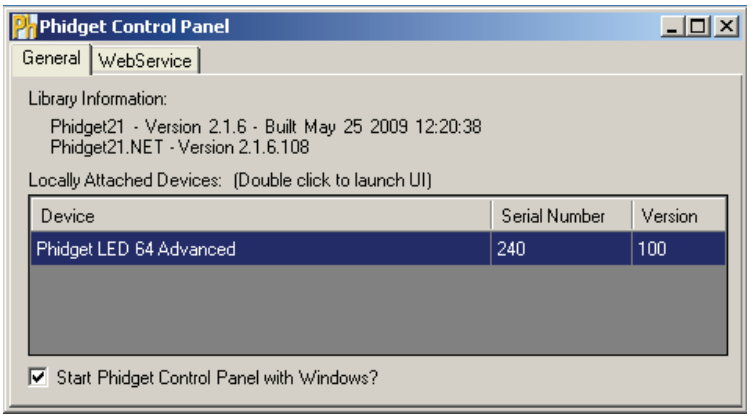
You should see the icon on the right hand corner of the Task Bar.

Running Phidgets Sample Program

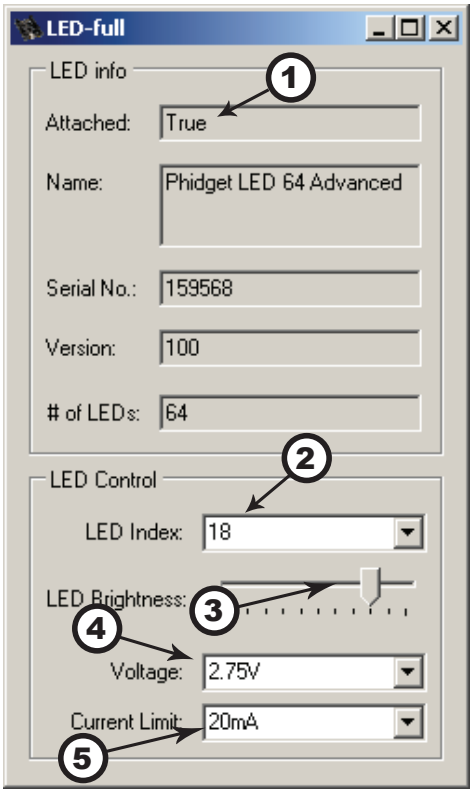
Double clicking on the icon loads the Phidget Control Panel; we will use this program to make sure that your new Phidget works properly.

The source code for the LED-Full sample program can be found under C# by clicking on Phidget.com > Programming.

Double Click on the  icon to activate the Phidget Control Panel and make sure that the **Phidget LED 64 Advanced** is properly attached to your PC.



1. Double Click on **Phidget LED 64 Advanced** in the Phidget Control Panel to bring up LED-full and check that the box labelled Attached contains the word True.
2. Select the LED Index. If you connected your LEDs at the same position as the ones in the picture on the previous page, select 18, 19, 22 or 23.
3. Use the LED Brightness slider to increase or decrease the LED brightness.
4. Select from the drop-down menu an appropriate anode voltage: 1.7V, 2.75V, 3.9V, or 5V.
5. Select from the drop-down menu an appropriate current limit: 20mA, 40mA, 60mA, or 80mA.



Testing Using Mac OS X

- Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane
- Make sure that the **Phidget LED 64 Advanced** is properly attached.
- Double Click on **Phidget LED 64 Advanced** in the Phidget Preference Pane to bring up the LED-full example. This example will function in a similar way as the Windows version.

If you are using Linux

There are no sample programs written for 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, ARMV4I or MIPSII, 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. You can find the manual at www.

phidgets.com >> Programming.

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 at www.phidgets.com >> Programming, and are listed under the appropriate language.

API Guides

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 www.phidgets.com >> 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 www.phidgets.com >> 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.

API for the PhidgetLED-64 Advanced

We document API Calls specific to this product in this section. Functions common to all Phidgets and functions not applicable to this device are not covered here. This section is deliberately generic. For calling conventions under a specific language, refer to the associated API manual. For exact values, refer to the device specifications.

The 1031 streams its entire state on attach, therefore LED brightness, current and voltage settings are initialized and can be read back as of the attach event.

Functions

int LEDCount() [get] : Constant

Returns the number of LEDs that this board can drive. This may not correspond to the actual number of LEDs attached.

int DiscreteLED(int LEDIndex) [get,set]

Sets/Gets the brightness of an LED. Valid values are 0-100, with 0 being off and 100 being the brightest. This 0-100 value is converted internally to an 7-bit value (0-127).

int CurrentLimit() [get,set]

Sets/Gets the current limit. Valid current limits are 20mA, 40mA, 60mA and 80mA, these will generally be delimited with an enumerator where possible, or an integer where not. The default current limit is 20mA.

int Voltage() [get,set]

Sets/Gets the output voltage. Valid voltages are 1.7V, 2.75V, 3.9V and 5V, these will generally be delimited with an enumerator where possible, or an integer where not. The default voltage is 2.75V.

Events

Error(String errorDescription, int errorCode) [event]

1031 uses the generic error event to stream back fault conditions. The fault can be identified by the errorDescription string. The only fault currently reported is Thermal Shutdown (TSD).

Technical Section

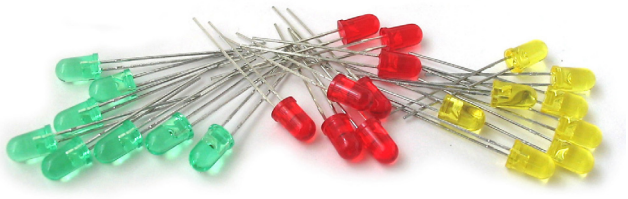
What are LEDs?

Like normal diodes, Light Emitting Diodes (LEDs) are semiconductor devices designed to conduct current in one direction only. What makes LEDs unique is their internal material makeup: when atoms in an LED release energy due to the flow of forward current, it is released in the form of photons (light). Different construction materials and various phosphor coatings are used to produce numerous colors of light.

Controlling the LEDs

Forward Voltage

The materials used within LEDs that cause them to emit different colors of light affect a property called its forward voltage. The forward voltage is the voltage at which current in the forward direction will flow through the device and allow the LED to convert electrical energy into light. If the voltage applied to the LED is below the forward voltage of the LED, very little current (or none) may flow, and therefore very little light will be emitted.



Most standard LEDs with colors such as red, amber, orange, yellow, and green have forward voltages below 2.75 Volts, and can be used with the PhidgetLED by simply soldering them to a connector-wire and inserting the wire into any PhidgetLED board connector. The forward voltage will default to 2.75V, and the maximum current defaults to 20mA.

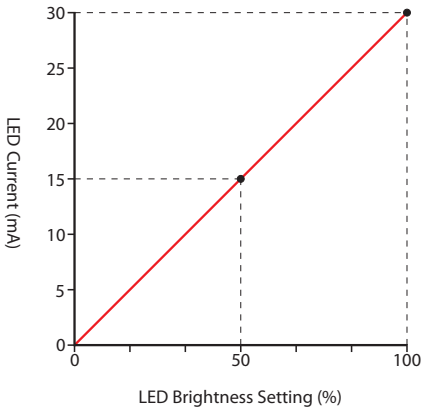
Supply Voltage

The 1031 is capable of adjusting the forward voltage supplied to the LEDs with 1.7, 2.75, 3.9 and 5 volts settings allowing you to properly drive blue, white, violet, ultra violet and purple LEDs. The supply voltage will affect all LEDs. If you set the supplied voltage too high, power will be wasted and the PhidgetLED may shut down from thermal overload. If you set the supply voltage too low, your LEDs will not be driven at the requested current, and will be dim or non-functional.

Typical Forward Voltages	
Color	**Forward Voltage**
Infrared	< 1.9
Red	1.7 to 2.2
Orange	2.0 to 2.2
Yellow	2.1 to 2.4
Green	2 to 2.3
Blue	3.2 to 4.0
UltraViolet	2.1 to 3.8
White	3.3 to 3.6

Maximum Current and Brightness Control

The maximum current can be set to 20, 40, 60 or 80mA, and applies to all LEDs. The DiscreteLED API call can be used to provide more current or control brightness and will adjust the current linearly between 0 and the set maximum. This however does not imply a precise method for controlling the visibility of emitted light, as this is affected by the construction and quality of the LED as well as the eyes of the viewer. Be cautious when changing the current property. Many small LEDs are designed for a maximum 20mA, and can be destroyed if driven at higher currents.



Choosing Current and Voltage Settings

Make sure to choose the minimum supply voltage setting to drive the LED that requires the most voltage during operation. Any extra voltage not required by the LED will be converted to heat by the 1031. For example, a Blue LED being driven at 20mA, 3.9V Supply, that requires 3.7 volts will cause (3.9V-3.7V + 0.4V) * 0.02A = 12 milliwatts of heat to be produced on the 1031. If this example instead uses a high power, 1.5V infrared LED at 80mA, this will create (3.9V-1.5V+0.4V)*0.08 Amps = 224 milliwatts of heat. See the Heat Dissipation and Thermal Protection section later on in this manual for more information about this issue.

Multiplexed LEDs

The 1031 does not use multiplexing. All 64 anodes are connected to the same power supply and the cathode of each LED connection is attached to an individual constant current sink. Also, the LEDs are not controlled by PWM - they are driven at a constant current. This reduces the electrical noise in installations.

Indicators, Cables and Connectors

Types

Different types of indicators will have to be connected to the PhidgetLED in different ways. Tri-color LEDs, seven-segment displays, bar LEDs, and other array-style arrangements of LEDs should be purchased as common-anode devices. As all the Anodes on the 1031 are wired together, effort can be saved by only connecting to one Anode. As there is no LED multiplexing on the 1031, it is impossible to use common-cathode LED devices.

Indicators

The PhidgetLED 64 can be used with several different styles of indicators, including standard LEDs, multi-colored LEDs, seven-segment displays, and bar/array style LEDs. Several examples are shown below.

Manufacturer	Part Number	Description
LiteON	LTL-307E	5mm Round High-Efficiency Red LED
LiteON	LTL-307G	5mm Round Green LED
Lumex	LTS-4801JR	0.4” Red Common-Anode 7-Segment Display

Note: most of the indicators listed, and many others, are available from www.digikey.com

Cable and Connector Components for LED Outputs

Manufacturer	Part Number	Description
Molex	50-57-9404	4 Position Cable Connector
Molex	16-02-0102	Wire Crimp Insert for Cable Connector
Molex	70543-0003	4 Position Vertical PCB Connector
Molex	70553-0003	4 Position Right-Angle PCB Connector (Gold)
Molex	70553-0038	4 Position Right-Angle PCB Connector (Tin)
Molex	15-91-2045	4 Position Right Angle PCB Connector - Surface Mount

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

Other Uses

The 1031 PhidgetLED 64 is not limited to Light Emitting Diodes. It can be used to control relays, solenoids and even very small motors. The PhidgetLED can also be used to drive opto-isolators and MOSFET SSRs. A diode integrated into the 1031 on each cathode will clamp inductive surges to the anode supply voltage.

Power Requirements and Power Supply Selection

The power supply that is included with the 1031 is rated at 12V and 2A (max). If your application requires more power, a larger power supply may be necessary. The on-board voltage regulator is able to supply up to 6A for each LED supply voltage setting, as long as the power supply is able to provide enough voltage and current to the regulator. Assume an efficiency of 80% for the on-board voltage regulator when determining if a different power supply is required.

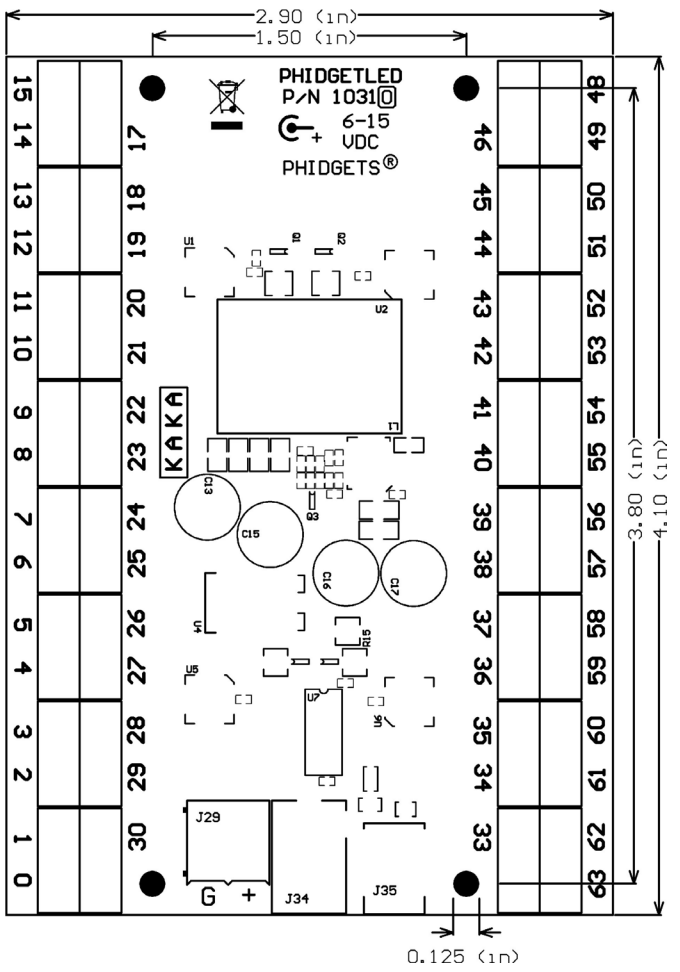
Heat Dissipation and Thermal Protection

Projects that require a high supply voltage, or have a lot of heat being produced from over voltage settings, will have over-temperature problems. This can be mitigated somewhat by understanding how channels are grouped and how the heat is distributed around the 1031. Channels are split into four groups: (0-7,24-31), (8-23), (32-39, 56-63) and (40-55); each controlled by their own individual IC. Evenly distributing the LEDs that may produce a lot of heat across these groups will balance the load on the ICs and reduce the risk of thermal overload.

When thermal overload occurs, the integrated circuit (IC) controlling the involved LEDs will disable the output of all the channels it controls. For example, if an over-temperature occurs due to channel 12, all of the channels 8 through 23 will be disabled by the IC until the temperature back within the operating range. Thermal protection is activated when the die of the IC reaches approximately 160 degrees Celsius. Once the over-temperature fault has been corrected (ie, the IC has cooled down), the output channels will be re-enabled with the same settings as before the thermal shutdown. An error message will be produced during an over-temperature.

Mechanical Drawing

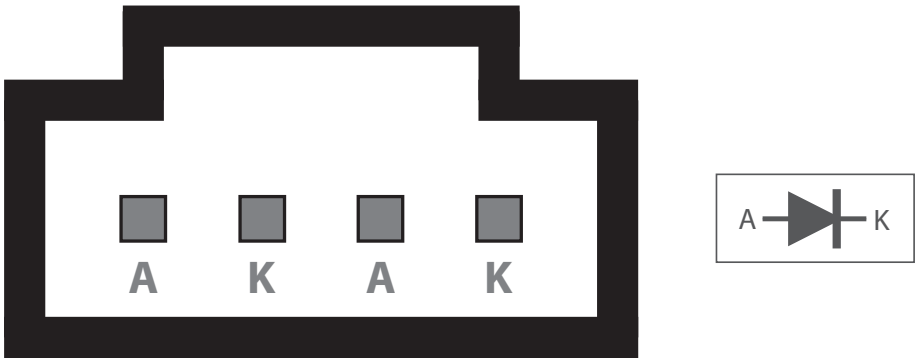
1:1 scale



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

Board Connector Drawing

Not to Scale



Device Specifications

Characteristic	Value
Brightness Changes	700 updates / second
LED On/Off	3000 updates / second
Selectable Output Anode Voltages	1.7VDC, 2.75VDC, 3.9VDC, 5VDC
Selectable Output Current (max) (V_F <= 3.0V)	20mA, 40mA, 60mA, 80mA
LED Forward Voltage (max)	5VDC
USB Voltage	4.5 to 5.25 VDC
Device Current Consumption	195mA
External Power Supply Voltage (min)	6VDC
External Power Supply Voltage (max)	15VDC
Externally Supplied Current Consumption (max) VDC = 6V	5.2A@80mA/LED 3.9A@60mA/LED 2.6A@40mA/LED 1.3A@20mA/LED
VDC = 15V	2.2A@80mA/LED 1.6A@60mA/LED 1.0A@40mA/LED 0.5A@20mA/LED

Product History

Date	Board Revision	Device Version	Comment
March 2010	0	100	Product Release

Support

- Call the support desk at 1.403.282.7335 8:00 AM to 5:00 PM Mountain Time (US & Canada) - GMT-07:00

or

- E-mail us at: support@phidgets.com