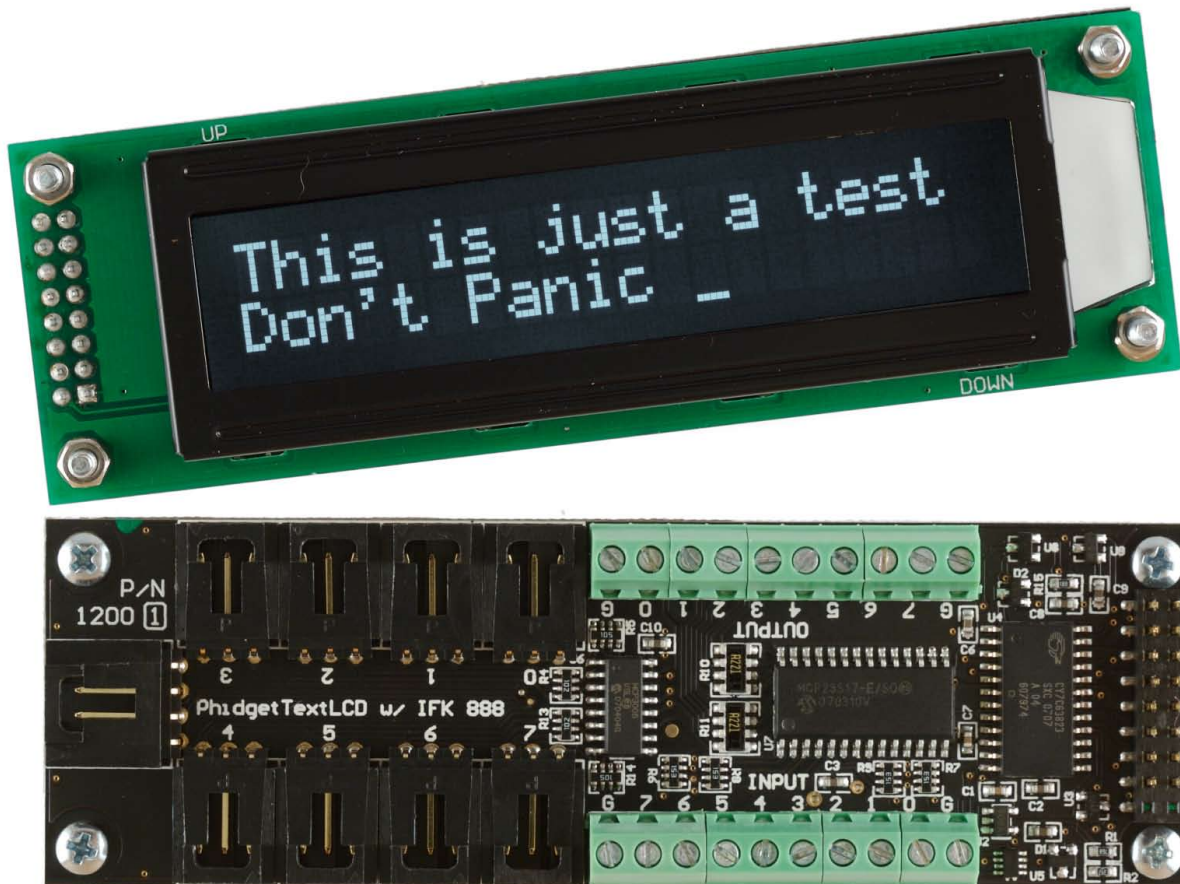


1203 - PhidgetTextLCD with PhidgetInterfaceKit 8/8/8



Programming Environment

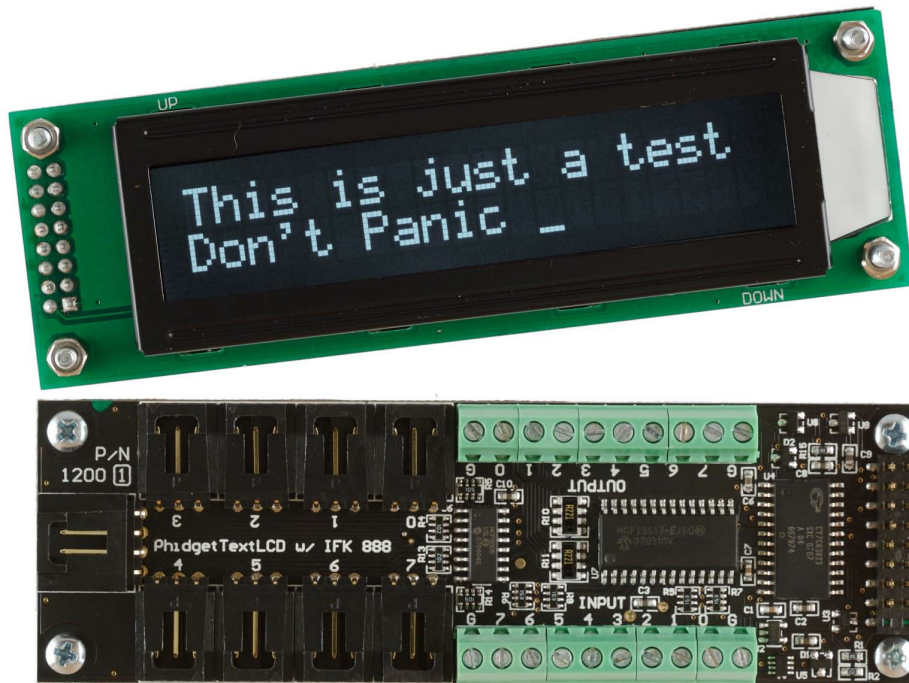
Operating Systems: Windows 2000/XP/Vista, Windows CE, Linux, and Mac OS X

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
Freescall 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

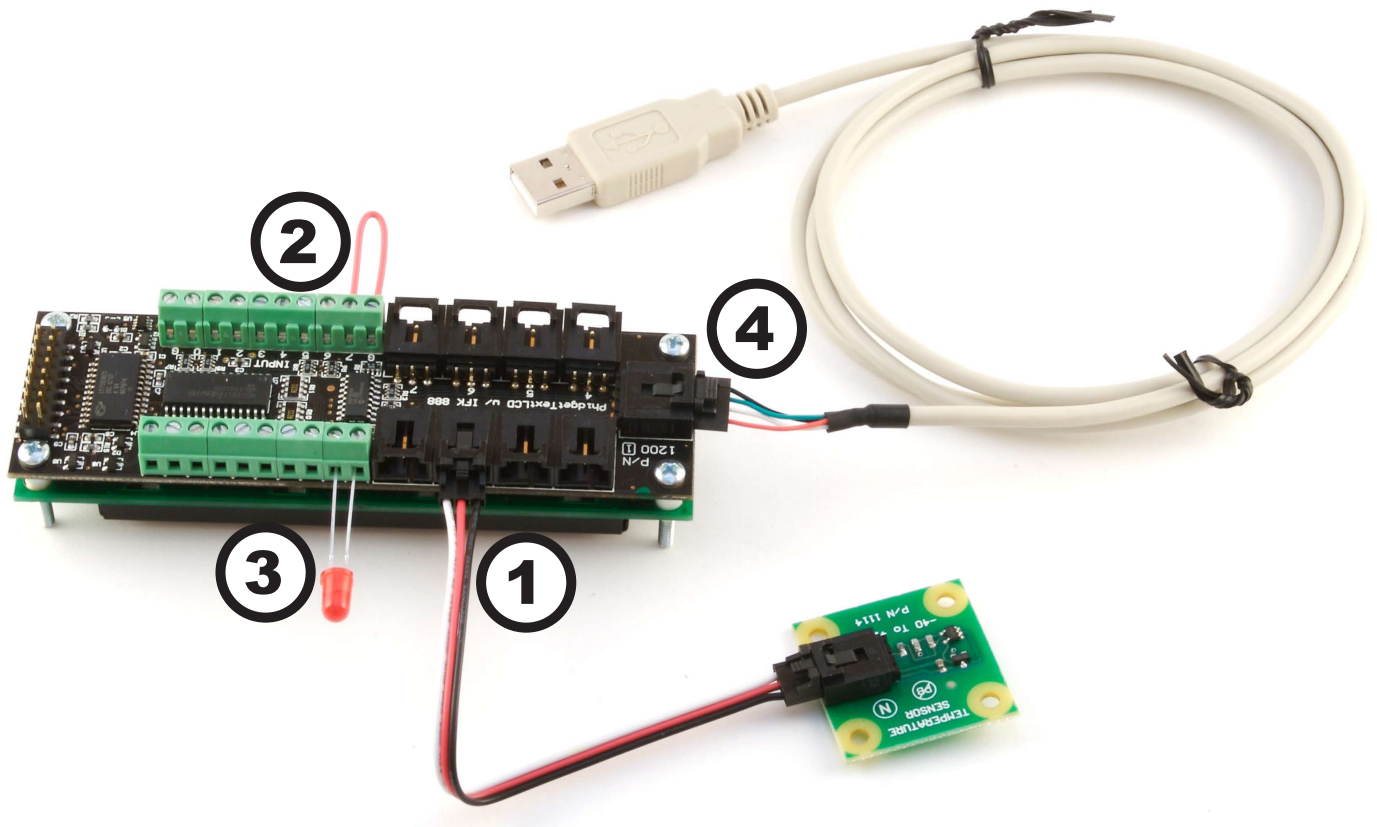
Installing the hardware

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

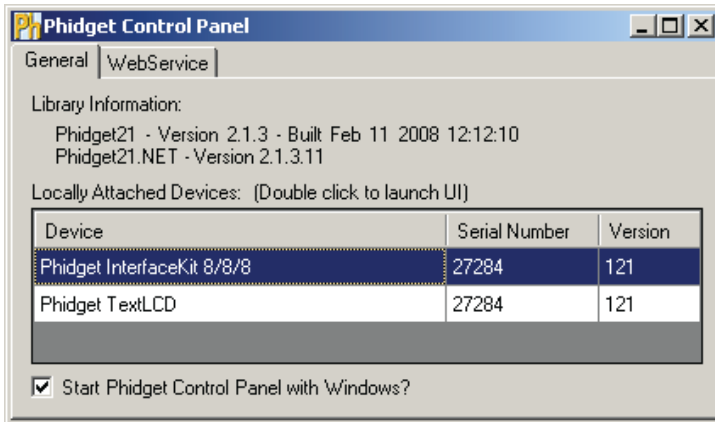
If you are using Windows 2000/XP/Vista

Go to www.phidgets.com >> Downloads >> Windows

Download and run Phidget21.MSI

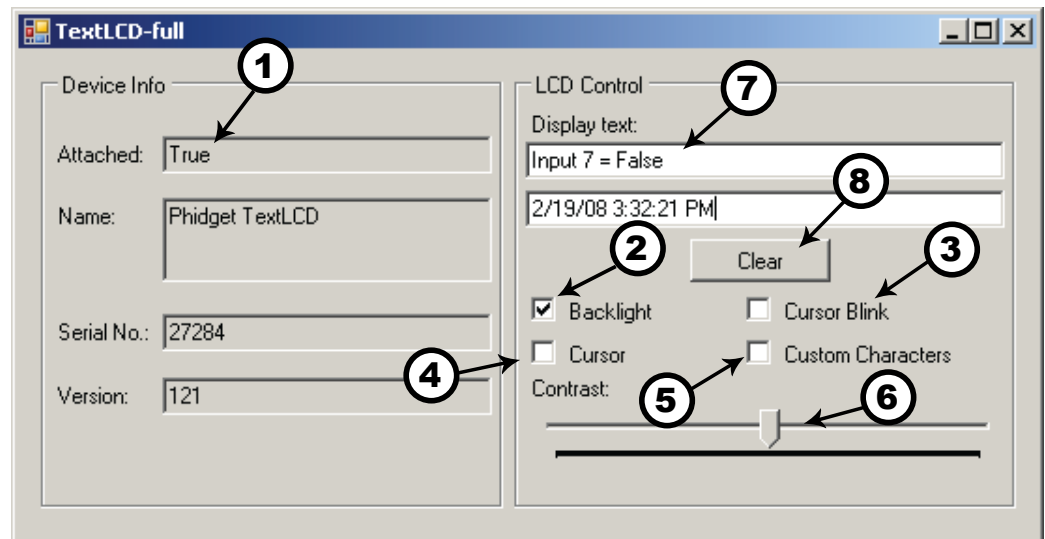
You should see the **Ph** icon on the right hand corner of the Task Bar.

Testing the PhidgetTextLCD Functionality



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.

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

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.

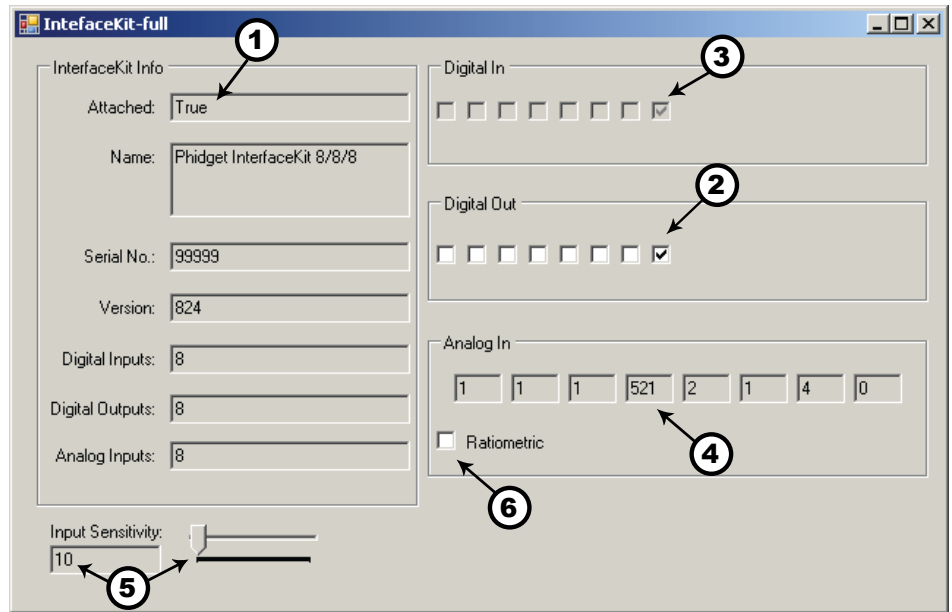
2. Test the digital output by clicking on the box to turn on the LED. Clicking again will turn the LED off.

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.

4. Click on the Ratiometric box if your sensor is ratiometric. Check your sensor product manual if you are not sure.

5. Test the analog input sensor by observing the sensor value as you activate the Phidget sensor.

6. You can adjust the input sensitivity by moving the slider pointer.

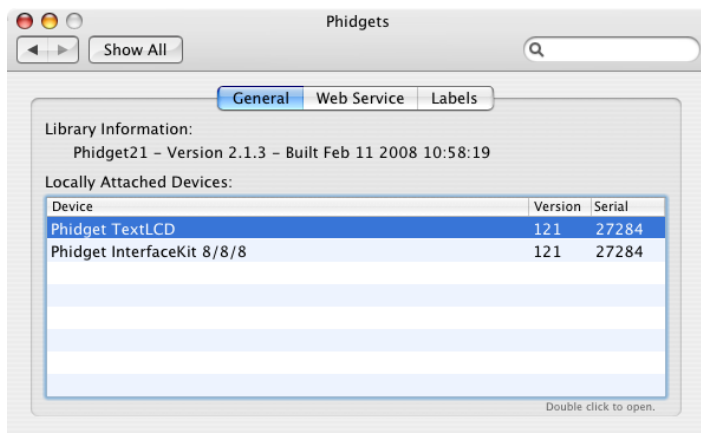


If you are using Mac OS X

Go to www.phidgets.com >> downloads >> Mac

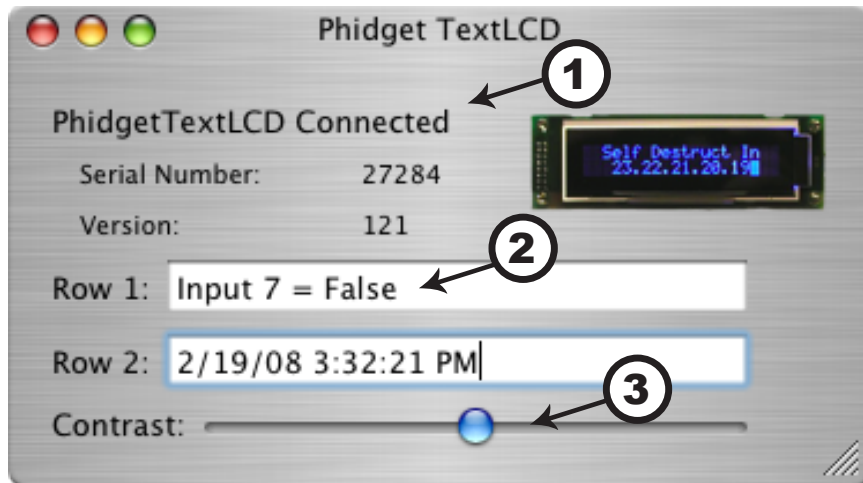
Download Mac OS X Framework

Testing the PhidgetTextLCD functionality



Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane. Make sure that the **PhidgetTextLCD** is properly attached.

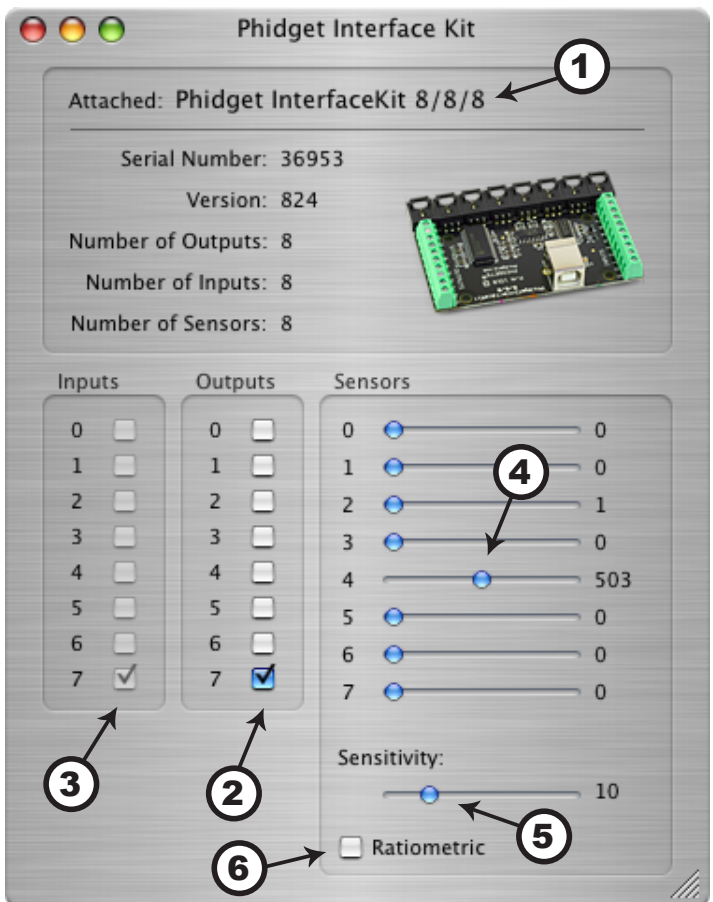
1. Double Click on **PhidgetTextLCD** in the Phidget Preference Pane to bring up the Phidget TextLCD pane and check that the TextLCD is connected.



2. You can type a 2 line message to be displayed on the LCD screen.
3. You can increase or decrease the contrast by using the Contrast pointer.

Testing the InterfaceKit 8/8/8 Functionality

1. Double Click on **PhidgetInterfaceKit 8/8/8** in the Phidget Preference Pane to bring up InterfaceKit-full and check that the InterfaceKit is attached.
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.
4. Click on the Ratiometric box if your sensor is ratiometric. Check sensor product manual if not sure.
5. Test the analog input sensor by observing the sensor value as you activate the Phidget sensor.
6. You can adjust the input sensitivity by moving the slider.



Programming a Phidget

Where to get information

- Go to www.phidgets.com >> downloads
- Select the Operating System and the language you want to use.
- Download the appropriate API manual and read the section under the TextLCD heading.
- Have a look at the source code of the TextLCD - full program.
- Have a look at the C# example below.
- Modify an existing program or write your own program from scratch.

Simple example written in C#

```
/* - TextLCD simple -
*****
* This simple example set up a TextLCD, waits for one to be attached, and then prompts
* the user to enter two lines of text to be displayed on the TextLCD
*
* Please note that this example was designed to work with only one Phidget TextLCD
* connected. For an example using multiple Phidget TextLCDs, please see a "multiple"
* example in the TextLCD Examples folder.
*
* Copyright 2007 Phidgets Inc.
* This work is licensed under the Creative Commons Attribution 2.5 Canada License.
* To view a copy of this license, visit http://creativecommons.org/licenses/by/2.5/ca/
*/

using System;
using System.Collections.Generic;
using System.Text;
//Needed for the TextLCD class, Phidget base classes, and the PhidgetException class
using Phidgets;
//Needed for the Phidget event handling classes
using Phidgets.Events;

namespace TextLCD_simple
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                //set up our Phidget TextLCD and hook the event handlers
                TextLCD tLCD = new TextLCD();

                tLCD.Attach += new AttachEventHandler(tLCD_Attach);
                tLCD.Detach += new DetachEventHandler(tLCD_Detach);
                tLCD.Error += new ErrorEventHandler(tLCD_Error);

                tLCD.open();

                //We have to wait to make sure that a TextLCD is plugged in before
                //trying to communicate with it
                if(!tLCD.Attached)
                {
```



```

        Console.WriteLine("Waiting for TextLCD to be attached....");
        tLCD.WaitForAttachment();
    }

    //prompt for the first line of input, Phidget TextLCD have two display
    //lines
    Console.WriteLine("Enter text to display on line 1:");
    string line1 = Console.ReadLine();

    //make sure a TextLCd is still attached before trying to communicate
    //with it...this is for if the TextLCd has been detached while waiting
    //for user input
    if (tLCD.Attached)
    {
        if (line1.Length > tLCD.rows[0].MaximumLength)
        {
            while (line1.Length > tLCD.rows[0].MaximumLength)
            {
                Console.WriteLine("Entered text is too long, try again...");
                line1 = Console.ReadLine();
            }
        }
        else
        {
            if (tLCD.Attached)
            {
                tLCD.rows[0].DisplayString = line1;
            }
        }
    }

    //prompt for the second line of input
    Console.WriteLine("Enter text to display on line 2:");
    string line2 = Console.ReadLine();

    //make sure a TextLCd is still attached before trying to communicate
    //with it...this is for if the TextLCd has been detached while waiting
    //for user input
    if (tLCD.Attached)
    {
        if (line2.Length > tLCD.rows[1].MaximumLength)
        {
            while (line2.Length > tLCD.rows[1].MaximumLength)
            {
                Console.WriteLine("Entered text is too long, try again...");
                line2 = Console.ReadLine();
            }
        }
        else
        {
            if (tLCD.Attached)
            {
                tLCD.rows[1].DisplayString = line2;
            }
        }
    }

    //Close the phidget
    tLCD.close();

```

```

        Console.WriteLine("ok");
    }
    catch (PhidgetException ex)
    {
        //output any exception data to the console
        Console.WriteLine(ex.ToString());
    }
}

//attach event handler, we'll output the name and serial number of the TextLCD
//that was attached
static void tLCD_Attach(object sender, AttachEventArgs e)
{
    TextLCD attached = (TextLCD)sender;
    string name = attached.Name;
    string serialNo = attached.SerialNumber.ToString();

    Console.WriteLine("TextLCD name:{0} serial No.: {1} Attached!", name,
                      serialNo);
}

//Detach event handler, we'll output the name and serial of the phidget that is
//detached
static void tLCD_Detach(object sender, DetachEventArgs e)
{
    TextLCD detached = (TextLCD)sender;
    string name = detached.Name;
    string serialNo = detached.SerialNumber.ToString();

    Console.WriteLine("TextLCD name:{0} serial No.: {1} Detached!", name,
                      serialNo);
}

//TextLCD error event handler, we'll just output any error data to the console
static void tLCD_Error(object sender, ErrorEventArgs e)
{
    Console.WriteLine("LCD Error: e.Description");
}
}
}

```

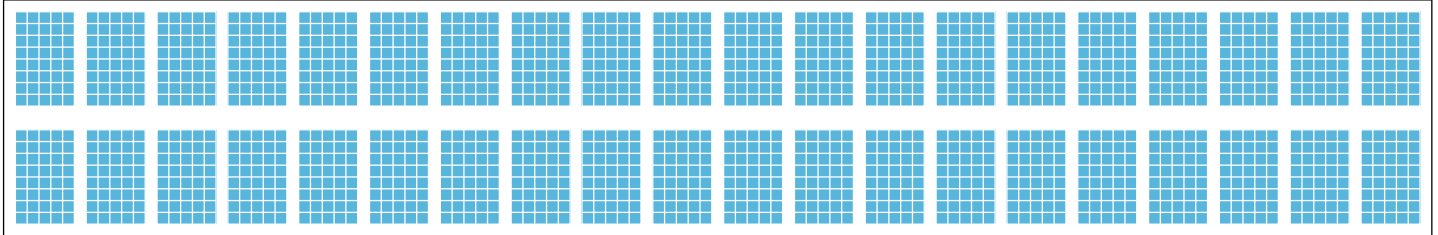
Learning more ...

- Check out the forums
- Check out the Phidgets projects

Technical Section

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 allotted 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:

$$VAL_{UPPER} = 2^6 + 2^8 + 2^{11} + 2^{13} = 10560$$

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



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].setCustomCharacter(10560, 14666);
```

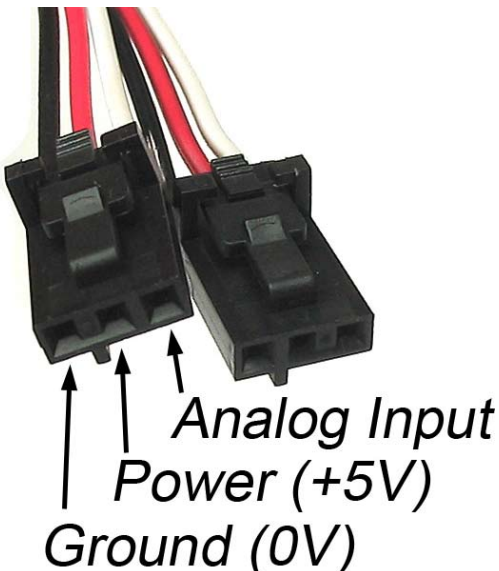
Once stored, characters can be recalled into a text string by either using the unicode value for the location as referenced in the ASCII chart (Appendix A) or by using the String Code method from the API. Examples in C# of both methods are shown below:

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

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

4	3	2	1	0
9	8	7	6	5
14	13	12	11	10
19	18	17	16	15
4	3	2	1	0
9	8	7	6	5
14	13	12	11	10
19	18	17	16	15

5X8 LCD Character Pixel Arrangement



Using the Analog Inputs

The Analog Input can measure a voltage between 0V and 5V. The analog measurement is represented in the software as a value between 0 and 1000, so a sensor value of 1 unit represents a voltage of approximately 5 millivolts.

Each analog input uses a 3-pin, 0.100 inch pitch locking connector. Pictured here is a plug with the connections labeled. If this is wired backwards, damage to your sensor may result. The Interface Kit provides + 5VDC, ground, and an analog input with a range of 0 to 5V.

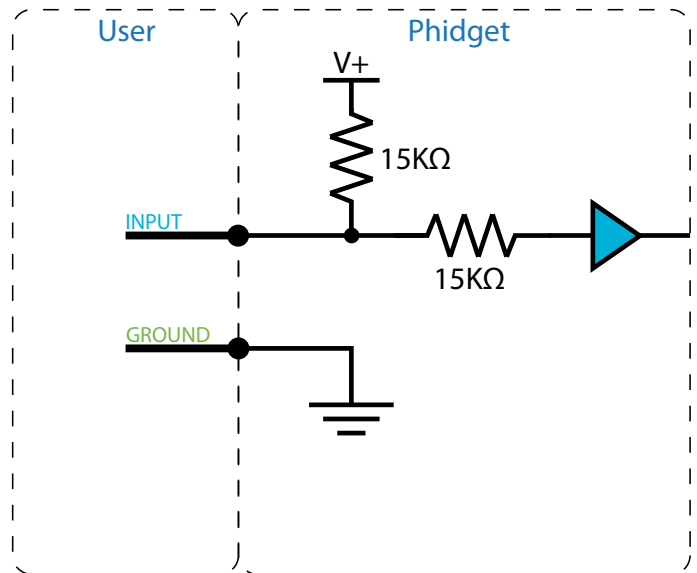
Ratiometric Sensors

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).

If the analog sensors you are using are ratiometric, enable the ratiometric property in software. This causes the reference to the internal Analog to Digital Converter to be set to the power supply voltage level. If ratiometric is not enabled, the ADC reference is set to a 5.0V 0.5% stable voltage reference.

Using the Digital Inputs

To wire a switch to a digital input, connect the switch between an input, labeled 0 to 7, and a provided ground, labeled G. The default state of the Digital Input in software is False (the switch state is open and the input pin is pulled to 5V by an internal resistor). When the switch is closed, the input pin is pulled to ground and the Digital Input is set to True.

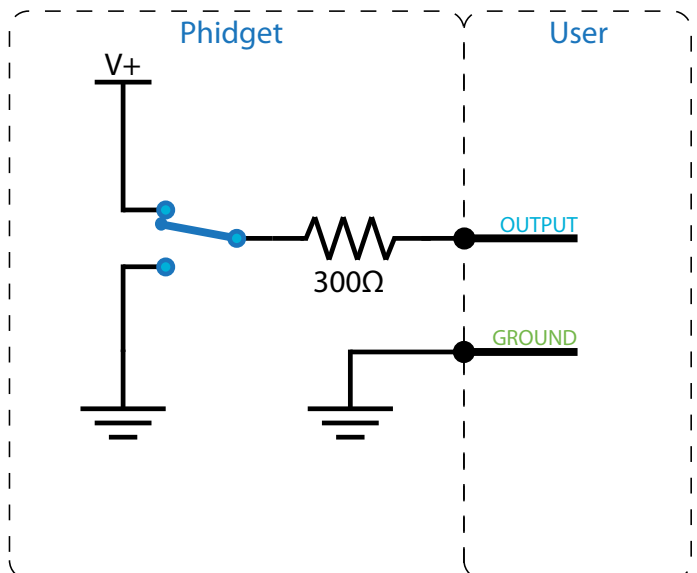


Digital Input Diagram

Using the Digital Outputs

Connecting an LED or other circuit to a digital output is simple. In the case of an LED, wire the anode to a digital output labeled 0 to 7 on the Interface Kit, and the cathode to a supplied ground, labeled G.

The 300 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.



Digital Output Diagram

The digital outputs can be used to switch larger electrical currents and voltages using devices such as power transistors, or logic level MOSFETs. You can also use the 3051 or 3052 to control a larger load.

Device Specifications

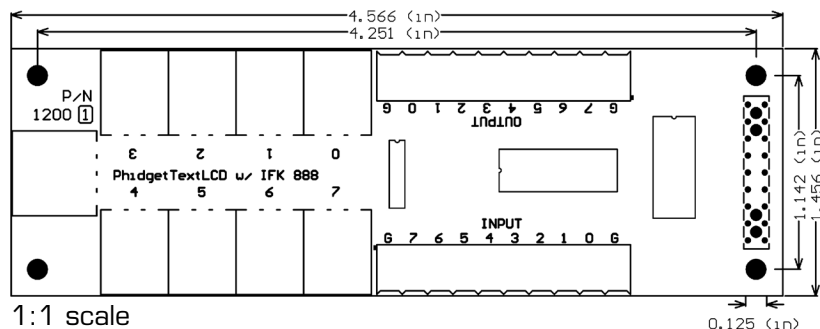
Analog Input Impedance	900K ohms
Digital Output Series Resistance	300 ohms
Digital Input Pull-Up Resistance	15K ohms
Digital Input Threshold Voltage (High-Low Transition)	1VDC
Digital Input Threshold Voltage (Low-High Transition)	4VDC
Analog Input Update Rate	~65 samples / second
Digital I/O Update Rate	~125 samples / second
Digital I/O Recommended Wire Size	16 - 26 AWG
Digital I/O Wire Stripping	5 - 6mm strip
USB-Power Current Specification	Max 500mA
Quiescent Current Consumption	50mA
Available External Current (source)	450mA

Cable and Connector Components for Analog Inputs

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 purchased at www.digikey.com

Mechanical Drawing



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	a	P	`	~				ー	ヲ	ヲ	ヲ	ヲ
	1	CG RAM (2)		!	1	A	Q	a	~			。	ア	チ	レ	ダ	グ
	2	CG RAM (3)		"	2	B	R	b	r			フ	イ	ウ	×	フ	Θ
	3	CG RAM (4)		#	3	C	S	c	s			ノ	ウ	テ	エ	ス	×
	4	CG RAM (5)		\$	4	D	T	d	t			ノ	エ	ト	ト	ノ	α
	5	CG RAM (6)		%	5	E	U	e	u			。	オ	ナ	1	α	α
	6	CG RAM (7)		&	6	F	V	f	v			ヲ	カ	ニ	ヨ	ρ	Σ
	7	CG RAM (8)		'	7	G	W	g	w			ヲ	キ	マ	ヲ	g	π
	8	CG RAM (1)		(8	H	X	h	x			ノ	ウ	ホ	リ	ノ	×
	9	CG RAM (2))	9	I	Y	i	y			ヲ	キ	リ	ル	ノ	γ
	A	CG RAM (3)		*	A	J	Z	j	z			エ	コ	ノ	ノ	ノ	ノ
	B	CG RAM (4)		+	B	K	L	k	l			オ	サ	ヒ	コ	ノ	ノ
	C	CG RAM (5)		,	C	L	M	l	m			ノ	サ	フ	フ	ノ	ノ
	D	CG RAM (6)		-	D	M	N	m	n			ユ	ズ	ノ	ノ	ノ	ノ
	E	CG RAM (7)		。	E	N	O	n	o			ヨ	セ	ト	ノ	ノ	ノ
	F	CG RAM (8)		/	F	O	P	o	p			ウ	ウ	マ	ノ	ノ	ノ