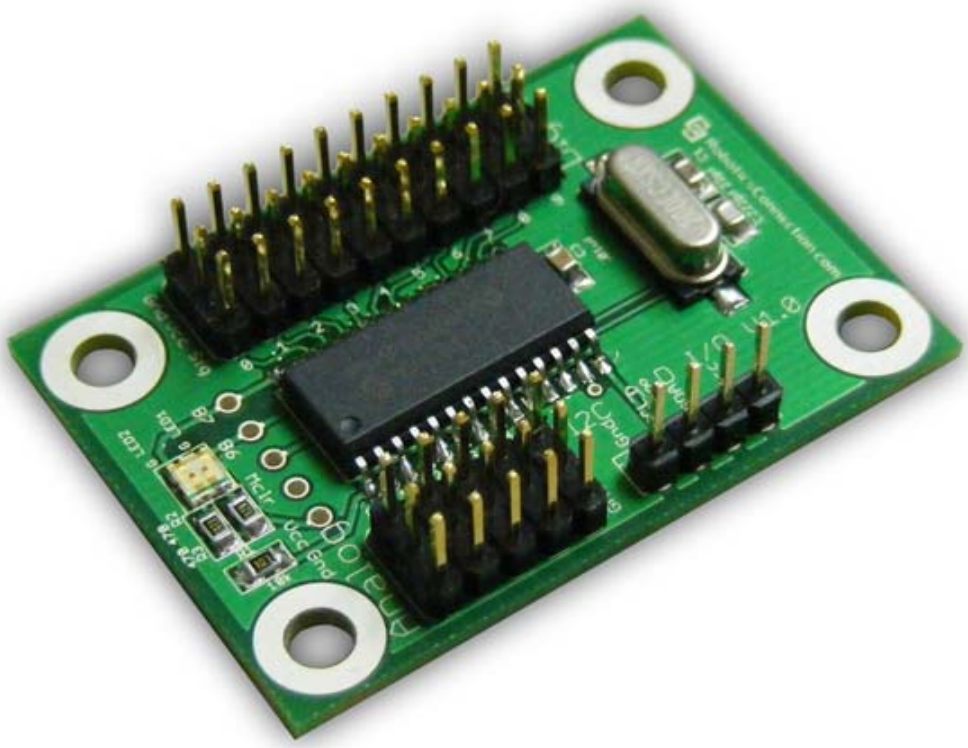
IOWizard

User Guide

v1.0



Introduction:
The IOWizard can be used when additional I/O capabilities are needed. It features 10 digital Inputs/Outputs, and 5 10-bit analog inputs, which are available via an I2C interface.

The 5 analog lines can be used for any 5V analog sensors, such as the Sharp line of infrared distance sensors. The 10 GPIO lines can be used for push buttons, LEDs, and other types of 0-5V general purpose I/O peripherals. All GPIO lines are set as inputs by default.

The IOWizard acts as an I2C slave device, and can be connected to, and queried from, any I2C Master, including our Serializer Robot Controller. *The **default I2C address** for the* IOWizard *is **0x64**(100d).* However, the address can be changed to any value between 100d and 110d.

IOWizard Pinout
The diagram below shows the various ports on the IOWizard.

The I2C header contains Vcc Sda, Scl, and Gnd pins, and requires no pull-up resistors since it acts as a slave device.

The GPIO header (GPIO0-GPIO9) contains 10 individual ports, where each port features a Gnd, Vcc, and Sig pin.

The Analog header (AN0-AN4) contains 5 individual, 10-bit ports, where each port features a Sig, Vcc, and Gnd pin. Any 3-5V analog sensor can be connected to this port, and queried via the protocol. The 'Sig' (or Signal) pin is the output voltage pin coming from the analog sensor into the RangeWizard.

Make sure you don't swap polarity (Vcc and Gnd) when connecting the sensors/peripherals/I2C bus!
We do not warranty against such misuse.

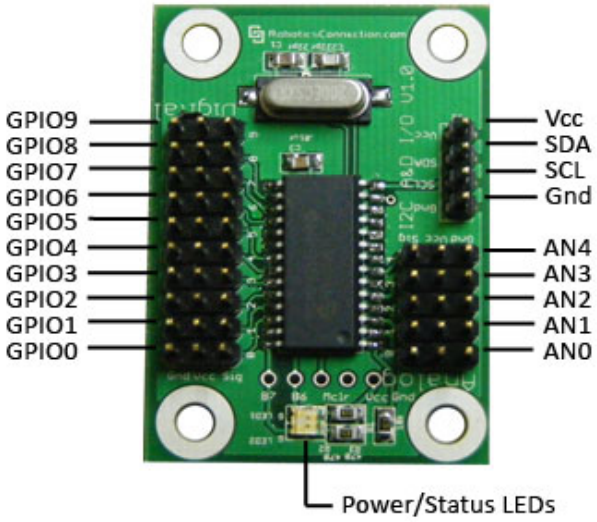


Figure 1 – IOWizard Pinout

Power/Status LEDs:

One of the two onboard status LEDs is used to indicate power, while the other flashes when commands are received and processed successfully by the board.

I2C Communication Protocol:

Querying ADC and GPIO Values:

The board returns 20 bytes when it's read. The first 10 bytes are 10 bit analog readings, and the last 10 are the status of the digital pins. If a digital pin is set as in input, its current state is returned. If a digital pin is set as an output the last state it was set to is returned.

Before you attempt to read a GPIO pin as an input, make sure you issue a command to set it as an input (see commands below), otherwise, you will receive the state of the pin output.

To query the ADC and GPIO values, you write its address, and then read back 20 bytes from its address +1. Below is sample code on how to read the device using a PIC and the CCS compiler. If you use another chip/compiler you will do something very similar.

```
i2c_start();                    // start condition
i2c_write(ADDRESS + 1);         // 101
value0 = i2c_read(1);
value1 = i2c_read(1);
    … do this 19 times.         // See NOTE below
Value19 = i2c_read(0);          // the last read has a 0 for a parameter.
i2c_stop();
```

NOTE: Each 10-bit analog value is returned in two bytes. So, you will have to shift the first (upper) byte by 8, and then add the subsequent (lower) byte to that shifted value to arrive at a 10-bit integer value. See the pseudo code below for an example:

Pseudo code:

```
Int adc;
int value;
…
adc = i2c_read(1) ;
value = adc << 8;
value += i2c_read(1);
…
```

Setting a GPIO pin state:
To set the state of a digital IO line (pin id range = 0-9) you must write the pin number, followed by the state. Below is the proper sequence of bytes to send:

0x64, 0x01, <pin id>, <state> // State: 0 = Output Low, 1 = Output High, 2 = Input

If you write a state of 0, then the pin is set to an output and set **low**. If you write a state of 1 then the pin is set to an output and set **high**.

If you write a state of 2, then the pin is set to an **input**, and you can read its value. ***By default all pins are set as inputs on power up.***

Below is sample code on how to change the state of a pin device using a PIC and the CCS compiler. If you use another chip/compiler you will do something very similar.

```
i2c_start();            // start condition
i2c_write(100);         //100d -110d
i2c_write(1);           //command to set I/O state
i2c_write(5);           //pin id (0 – 9)
i2c_write(1);           //Output = High  (Range: 0-2)
i2c_stop();
```

Saving Power up state of an I/O pin
You can save the state all the I/O pins to EEPROM, so that they will power up in that state after a reset or power cycle. Simply set the state of each pin as desired, and send a command of '2' to save the state.

Upon power up, each pin will be configured depending on the configuration under which you saved it. All 10 digital I/O pins are saved simultaneously. Furthermore, both LEDs will flash for a few seconds to indicate a successful save.

Below is sample code on how to save the state of a pin device using a PIC and the CCS compiler. If you use another chip/compiler you will do something very similar.

```
i2c_start();            // start condition
i2c_write(100);         //100 -140
i2c_write(2);           //command to save  I/O state to EEPROM
i2c_stop();
```

Resetting the IOWizard
To reset the IOWizard, you must simply write the address, followed by a value of 0x03.

0x64, 0x03

Below is sample code on how to save the state of a pin device using a PIC and the CCS compiler. If you use another chip/compiler you will do something very similar.

```
i2c_start();            // start condition
i2c_write(100);         //100 -140
i2c_write(3);           //command to reset
i2c_stop();
```

Changing the I2C address

You can change the I2C address of the IOWizard by sending the sequence of bytes shown below. Below is sample code on how to change the I2C address of the device using a PIC and the CCS compiler. If you use another chip/compiler you will do something very similar.

```
i2c_start();
i2c_write(currAddress); //Address
i2c_write(0x00);        //Command Register
i2c_write(0xA0);        //Address
i2c_stop();
//delay_ms(5);
i2c_start();
i2c_write(currAddress); //Address
i2c_write(0x00);        //Command Register
i2c_write(0xAA);        //Address
i2c_stop();
//delay_ms(5)
i2c_start();
i2c_write(currAddress); //Address
i2c_write(0x00);        //Command Register
i2c_write(0xA5);        //Address
i2c_stop();
//delay_ms(5)
i2c_start();
i2c_write(currAddress); //Address
i2c_write(0x00);        //Command Register
i2c_write(newAddress); //Set new address
i2c_stop();
```

IOWizard Physical Connection

The picture below depicts the IOWizard connected to the I2C port of our Serializer WL Robot Controller. The IOWizard can also be powered over the I2C bus as well. **Please note that the IOWizard can be connected to ANY I2C Host!**

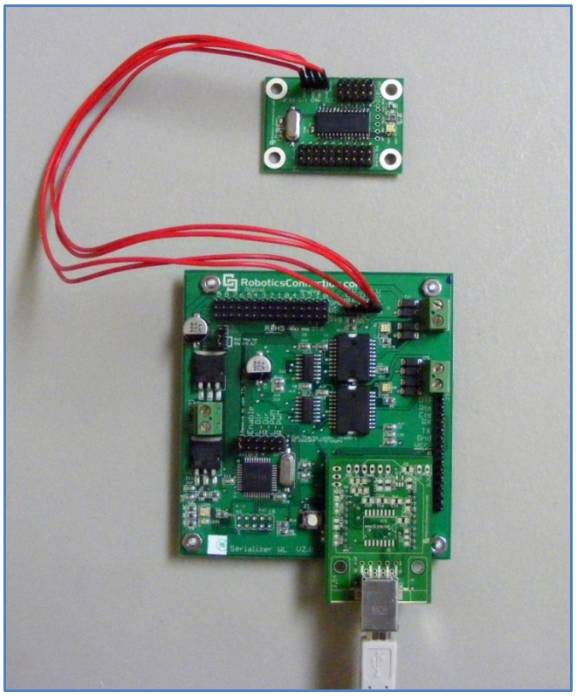


Figure 1 - IOWizard connected to Serializer WL via I2C bus

Establishing I2C Communications

All that is required to establish I2C communications is connect the Sda, and Scl pins between the I2C Master and the IOWizard, along with Vcc and Gnd pins, and you're ready to start communicating via I2C. *The default I2C address for the IOWizard is 0x6A (100d).* Please see Example Application section below for details on writing programs to communicate via I2C.

Power Supply

When using the I2C port, the IOWizard gets its 5V supply via the I2C port. **Make sure you don't swap polarity (Vcc and Gnd) when connecting the power supply! We do not warranty against such misuse**.

Communicating with the IOWizard thru the Serializer WL Robot Controller via I2C:
Communicating with the IOWizard via the Serializer using the I2C bus and the Generic I2C command is very easy (assumes I2C address of 100 (0x64). Remember that you're communicating w/ the Serializer serially, hence the Serializer command/response sequence below is similar to the IOWizard serial protocol (defined above). The Serializer then communicates with the IOWizard via the I2C bus.

Query the Sensor readings:
(Note that 20 bytes will be returned – The first 10 are 10-bit analog, depicted by the '1024' values)
(**Note**, the Serializer will return strings representing the 10-bit integer values, so you don't have to worry about shifting those 10-bit adc values that are returned).
>i2c w 100 1
ACK
>i2c r 100 20
1024 1024 1024 1024 1024 1024 1024 1024 1024 1024 0 1 0 1 0 1 0 1 0 1
>

Setting the state of the I/O Pins:
(Example 1: we're setting the state of pin 3 to 0 – Low)
>i2c w 100 1 3 0
ACK
>

(Example 2: we're setting the state of pin 8 to 1 – High)
>i2c w 100 1 8 1
ACK
>

Save the state of the I/O Pins:
>i2c w 100 2
ACK
>

Reset the IOWizard:
>i2c w 100 3
ACK
>

Electrical Specifications:

IOWizard™ Maximum Ratings

	DC Characteristics:					
Symbol	Characteristic/Device	Min	Typ†	Max	Units	Conditions
VDD	Supply Voltage	4.7	5.0	5.3	V	
IDD	Supply Current	-----	100	200	mA	

Absolute Maximum Ratings †

Ambient temperature under bias: -55 to +125°C

Storage temperature: -65°C to +150°C

Voltage on any pin with respect to VSS: -0.3V to 5.3V

Voltage on VDD with respect to VSS: -0.3 to +5.3V

Maximum current out of VSS pin: 300 mA

Maximum current into VDD pin: 250 mA

Input clamp current, IIK (VI < 0 or="" VI="" > VDD): 20 mA

Output clamp current, IOK (VO < 0 or="" VO="" > VDD): 20 mA

Maximum output current sunk by any I/O: 25 mA

Maximum output current sourced by any I/O pin: 25 mA

† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

Dimensions: