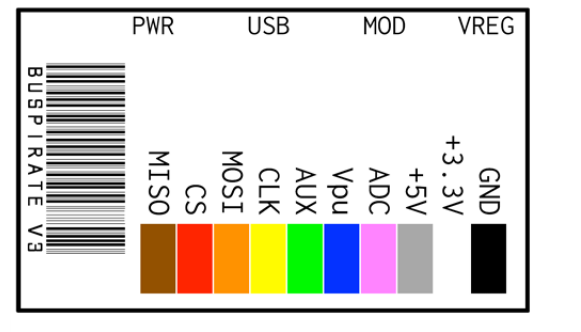
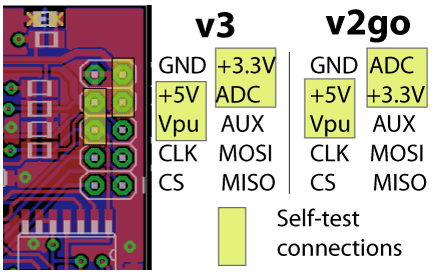
# Bus Pirate Guide

**Probe Cable Pinout**



**Self Test Procedure**



1. Disconnect any devices from the Bus Pirate I/O header pins. An external device will interfere with the test, and could be damaged by the voltages used (+5volts) on some pins.
2. Make the self-test connections shown in the figure above. Connect the Vpullup (Vpu) pin to the +5V pin. Connect the ADC pin to the +3.3V pin.
3. To begin the self test, type ‘~’ followed by *enter* in the terminal. Self-test is available in HiZ mode only.
4. You will be reminded to make the self-test connections and remove any external devices. Press any key to run the self-test.

**Modes Supported**

The Bus Pirate currently ‘speaks’ three hardware protocols for high-speed interfacing, and has two software protocol libraries for easy bus manipulation. The theory and specification of each protocol is beyond what we can cover here, but check out some of these tutorials:

I2C

A slow 2 wire bus. Wikipedia is a great place to start for [I2C background](http://en.wikipedia.org/wiki/I%C2%B2C). [I2C-Bus.org](http://www.i2c-bus.org/),[Robot Electronics](http://www.robot-electronics.co.uk/htm/using_the_i2c_bus.htm), [Embedded Systems Academy](http://www.esacademy.com/faq/i2c/), and [Embedded.com](http://www.embedded.com/story/OEG20010718S0073) have decent I2C tutorials.

SPI

A simple 3 wire bus. Wikipedia has [background](http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus); Embedded.com has a great [tutorial and comparison to I2C](http://www.embedded.com/columns/beginerscorner/9900483).

Universal Asynchronous Receiver Transmitter (UART or serial)

A clock and timing dependent serial protocol best known for its appearance as the PC serial port protocol. Wikipedia has background on [asynchronous serial protocols](http://en.wikipedia.org/wiki/Asynchronous_start-stop).

Raw 2 wire

This is a generic 2 wire protocol library, similar to I2C but without an ACK bit. I2C and many proprietary 2 wire protocols can be formed using the bus manipulations available in this mode. Use this library to work with non-I2C 2 wire devices, like[smartcards](http://www.smartcardsupply.com/Content/Cards/SLE4442.htm) or [Sensirion SHT11](http://www.sensirion.com/en/01_humidity_sensors/02_humidity_sensor_sht11.htm) temperature/humidity sensors.

Raw 3 wire

This is a generic 3 wire protocol library, similar to SPI but without the constraints of a hardware module. Use this library to work with devices that use non-8bit compatible 3-wire protocols, like the [Sparkfun Nokia 6100 LCD knock-off](http://www.sparkfun.com/commerce/product_info.php?products_id=569). Many 3 wire protocols can be formed using the bus manipulations available in this mode.

**General Commands**

*Menu options*

Menu options are single character commands that don’t involve data transfers. Enter the character, followed by <enter>, to access the menu.

**?** – Show a help menu with commands and syntax.

**M** – Set the bus mode (SPI, I2C, UART, raw 2 wire, raw 3 wire). Followed immediately by a prompt for speed, polarity, and output state (mode dependent).

* Bus speeds: SPI:30, 125, 250, 1000KHz. I2C:100, 400, 1000KHz. UART: 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200bps. Raw modes: 1, 10, 50KHz.
* Inverse clock setting sets the idle state opposite of normal (normal SPI:idle low; normal UART:idle high): SPI:idle high; UART:idle low.
* Some modes have optional high-z output modes for use with pull-up resistors (Low=ground, High=input).

**L –**Toggle bit transmit/receive order: most/least significant bit first.

**P** – SDA/SCL pin pull-up resistor toggle (3.3volts). Only valid in I2C and raw 2 wire modes.

**O** – Set number output display format. The terminal can display numbers as decimal, hexadecimal, and binary ASCII values. A fourth format sends the raw, unprocessed byte for reading ASCII formatted text.

*Syntax*

A simple syntax is used to communicate with chips over a bus.  Syntax commands have generic functions that generally apply to all bus types.

**A/a/@**– Toggle auxiliary pin. Capital “A” sets AUX high, small “a” sets to ground. @ sets aux to input (high impedance mode) and reads the pin value.

**[**- Start data write. SPI/raw 3 wire: chip select enabled. I2C/raw 2 wire: start condition. RS232: open UART, discard received bytes.

**{**- Start data write with reads. Same as [, except: SPI/raw 3 wire: show the read byte for each write. RS232: display data as it arrives asynchronously.

**] or }** – End data write. SPI/raw 3 wire: chip select disabled. I2C/raw 2 wire: stop condition. RS232: close UART.

**R/r** – Read byte. SPI/raw 3 wire: send dummy byte, return read. I2C: read byte with ACK. Raw 2 wire: read 8 bits. RS232: check UART for byte and return, or fail if empty. Use 0r1…255 for bulk reads up to 255 bytes.

**0b** – Write this binary value. Format is 0b00000000 for a byte, but partial bytes are also fine: 0b1001.

**0h or 0x** – Write this HEX value. Format is 0h01 or 0×01. Partial bytes are fine: 0xA. A-F can be lower-case or capital letters.

**0-255** – Write this decimal value. Any number not preceded by 0x, 0h, or 0b is interpreted as a decimal value.

**, or space** - Value delimiter. Use a coma or space to separate numbers. Any combination is fine, no delimiter is required between non-number values: {0xa6,0, 0 16 5 0b111 0haF}.

*Direct bus manipulation commands for raw 2 wire mode and raw 3 wire mode.*  
**^** – Send one clock tick. Use 0^1…255 for multiple clock ticks.

**/ and \** – Toggle clock level high (/) and low (\). Includes clock delay (100uS).

**-/\_** – Toggle data state high (-) and low (\_). Includes data setup delay (20uS).

**!**- Read one bit with clock.

**.**- Read data pin state (no clock).

**&** – Delay 1uS. Use 0&1…255 for multiple delays.

**Extensive Command Description**

|  |  |
| --- | --- |
| ***Menu*** | ***Description*** |
| *?* | *Help menu with latest menu and syntax options.* |
| *I* | *Hardware, firmware,*[*microcontroller*](http://dangerousprototypes.com/2009/08/07/find-your-bus-pirates-pic-revision/)*version information.* |
| *M* | *Set bus mode (1-Wire, SPI, I2C, JTAG, UART, etc).* |
| *B* | *Set PC side serial port speed.* |
| *O* | *Data display format (DEC, HEX, BIN, or raw).* |
| *V* | *Power supply voltage report (v1+ hardware only).* |
| *F* | *Measure frequency on the AUX pin. (1Hz-40MHz)* |
| *G* | *Frequency generator/PWM on the AUX pin. (1kHz-4MHz)* |
| *C* | *Toggle AUX control between AUX and CS/TMS pins.* |
| *L* | *Set*[*LSB/MSB*](http://en.wikipedia.org/wiki/Least_significant_bit)*first in applicable modes.* |
| *P* | [*Pull-up resistors*](http://dangerousprototypes.com/2009/07/27/bus-pirate-practical-guide-to-pull-up-resistors/)*(V0,V2+ hardware).* |
| *=* | *Convert HEX/DEC/BIN number format (firmware v2.1+).* |
| *~* | *Perform a*[*self-test*](http://dangerousprototypes.com/2009/07/28/bus-pirate-self-test-guide/)*(firmware v2.0+).* |
| *#* | *Reset (firmware v2.0+).* |

*Menu options are single character commands that configure the*[*Bus Pirate*](http://dangerousprototypes.com/docs/Bus_Pirate)*. Press enter to show the command prompt if your terminal is blank.*

*HiZ>p  
Command not used in this mode  
HiZ>*

*Enter a command, followed by <enter>, to access the menu. Most configuration and option prompts have a default value shown in (). Press enter to select the default option. Some options are unavailable in some modes, for example bit order configuration and pull-up resistors.*

***? Help menu with latest menu and syntax options***

*Print a help screen with all available menu and syntax options in the current firmware and hardware.*

***I Hardware, firmware, microcontroller version information***

*HiZ>i****<<<info command*** *Bus Pirate v3****<<<hardware version*** *Firmware v3.0****<<<firmware version*** *DEVID:0×0447 REVID:0×3042 (B4)****<<<PIC chip version***[*http://dangerousprototypes.com*](http://dangerousprototypes.com/)***<<<update URL*** *HiZ>*

*Show the hardware, firmware, and microcontroller version. Microcontroller version is available in firmware v2.1+.*

*\*———-\*  
POWER SUPPLIES OFF  
Voltage monitors: 5V: 0.0 | 3.3V: 0.0 | VPULLUP: 0.0 |  
a/A/@ controls AUX pin  
Normal outputs (H=V+, L=GND)  
Pull-up resistors ON  
MSB set: MOST sig bit first  
\*———-\*  
RAW2WIRE>*

*If a bus mode is configured, additional information may be available.*

***M Set bus mode (1-Wire, SPI, I2C, JTAG, UART, etc)***

*HiZ>m****<<<mode command*** *1. HiZ  
2. 1-WIRE  
3. UART  
4. I2C  
5. SPI  
6. JTAG  
7. RAW2WIRE  
8. RAW3WIRE  
9. PC KEYBOARD  
10. LCD  
(1) >1  
Mode selected  
HiZ>*

*Select a bus mode. The command resets the Bus Pirate and immediately disables all pins, pull-up resistors, and power supplies.*

*The default mode is HiZ, a safe mode with all pins set to high-impedance and all peripherals disabled.*

***B Set PC side serial port speed***

*HiZ>b  
Set serial port speed: (bps)  
1. 300  
…  
9. 115200  
(9) >9  
Adjust your terminal and press space to continue  
HiZ>*

***O Data display format (DEC, HEX, BIN, or raw)***

*HiZ>o  
1. HEX  
2. DEC  
3. BIN  
4. RAW  
(1) >1  
Display format set  
HiZ>*

***V Power supply voltage report***

*1-WIRE>v  
Voltage monitors: 5V: 4.9 | 3.3V: 3.2 | VPULLUP: 0.0 |  
1-WIRE>*

*(v1+ hardware only)*

***F Measure frequency on the AUX pin***

*HiZ>f  
Frequency on AUX pin: 50Hz  
HiZ>*

*Measures frequency from 0Hz to 40MHz.*

***G Frequency generator/PWM on the AUX pin***

*HiZ>g  
1KHz-4,000KHz frequency generator/PWM (beta)  
Frequency in KHz (50) >400  
Prescale:1  
PR2:39  
Duty cycle in % (50) >  
PWM active  
HiZ>*

*Enable the frequency generator, set frequency and duty cycle. Frequencies from 1kHz to 4MHz possible.*

*HiZ>g  
PWM disabled  
HiZ>*

*Use g again to disable the PWM.*

***C Toggle AUX control between AUX and CS/TMS pins***

*RAW2WIRE>c  
AUX commands control  
1. AUX (default)  
2. CS/TMS  
(1) >2  
a/A/@ controls CS/TMS pin  
RAW2WIRE>*

***L Set LSB/MSB first in applicable modes***

*RAW2WIRE>l  
1. MSB first  
2. LSB first  
(1) >1  
MSB set: MOST sig bit first  
RAW2WIRE>*

***P Pull-up resistors***

*RAW2WIRE>p  
WARNING: pins not open drain (HiZ)  
1. Pull-ups off  
2. Pull-ups on  
(1) >2  
Pull-up resistors ON  
RAW2WIRE>*

*See*[*the practical guide to Bus Pirate pull-up resistors*](http://dangerousprototypes.com/2009/07/27/bus-pirate-practical-guide-to-pull-up-resistors/)*. (V0,V2+ hardware)*

***= Convert HEX/DEC/BIN number format***

*HiZ>=  
CONVERT (1) >0×59  
0×59  
89  
0b01011001  
HiZ>*

*Firmware v2.1+*

***~ Perform a self-test***

*HiZ>~  
Disconnect any devices  
Connect (Vpu to +5V) and (ADC to +3.3V)  
Press a key to start*

*Perform a*[*hardware self-test*](http://dangerousprototypes.com/2009/07/28/bus-pirate-self-test-guide/)*.**Firmware v2.0+, hardware v2go and v3 only.*

***# Reset***

*HiZ>#  
RESET*

*Bus Pirate v3  
Firmware v3.0-nightly  
DEVID:0×0447 REVID:0×3042 (B4)*

[*http://dangerousprototypes.com*](http://dangerousprototypes.com/)

*HiZ>*

### Power supplies

3.3volt and 5volt on-board power supplies can provide up to 150mA for your project. Activate them with the w/W command from any mode except HiZ mode. HiZ mode is a safe mode and all outputs are disabled.

I2C>w**<<<power supplies off**  
POWER SUPPLIES OFF  
I2C>v**<<<voltage report**  
Voltage monitors: 5V: 0.0 | 3.3V: 0.0 | VPULLUP: 0.0 |  
I2C>W**<<<power supplies on**  
POWER SUPPLIES ON  
I2C>v**<<<voltage monitor report**  
Voltage monitors: 5V: 4.9 | 3.3V: 3.2 | VPULLUP: 0.0 |  
I2C>

Capital 'W' activates the on-board supplies, small 'w' turns them off. Turn the power supplies on, then press v to show a power supply voltage report.

Note that W is syntax and not a menu option, it can be used with other syntax to toggle the power in the middle of complex bus operations.

The power supplies will try to protect your project with over current protection. Additionally, the Bus Pirate will measure the voltage shortly after the supplies are enabled. If a short is detected it will disable them and show a warning. This may help minimize damage to a project when there is a short or problem on a new board.

### Pull-up resistors

Some bus types, such as I2C, require a resistor to create a '1' on the bus. This is called an [open collector or open drain bus](http://hackaday.com/2009/07/01/mixed-voltage-interfacing-with-the-bus-pirate/). Without a [pull-up resistor](http://en.wikipedia.org/wiki/Pull-up_resistor), the bus will only read '0'. Some bus modes are open collector only (1-Wire, I2C), most have an open collector option.

I2C>v **<<<voltage monitor report**  
Voltage monitors: 5V: 4.9 | 3.3V: 3.2 | VPULLUP: 4.9 |  
I2C>

Connect the Vpu pin of the IO header to a voltage. The on-board pull-up resistors apply the voltage connected to the Vpu pin to the bus pins through 10Kohm resistors. Press v and verify that the Vpullup voltage shows the desired value.

I2C>p **<<<configure pull-up resistors**  
1. Pull-ups off  
2. Pull-ups on  
(1) >2 **<<<enable**  
Pull-up resistors ON  
I2C>

Press p to configure the on-board pull-up resistors, enable them. For more, see the[practical guide to Bus Pirate pull-up resistors](http://dangerousprototypes.com/docs/Practical_guide_to_Bus_Pirate_pull-up_resistors).

### Macros

Macros are preprogrammed shortcuts. Some of the macros available include: smart card interrogation, I2C address scanners, JTAG chain probes, transparent UART bridges, LCD initialization, and bus sniffers.

I2C>(0)**<<< show macro menu**  
0.Macro menu  
1.7bit address search  
2.I2C sniffer  
I2C>(1)**<<< macro 1, I2C address search**  
Searching 7bit I2C address space.  
Found devices at:  
0xA0(0x50 W) 0xA1(0x50 R)  
I2C>

Type (0) to get a list of macros in any mode. The macro number can be entered in any format, (0x00) and (0b0) are also valid macro entries.

I2C devices respond to a 7bit address. We could look up the address in the datasheet, but instead we'll use the address scanner macro (1). The Bus Pirate scaned all possible addresses and found the chip at 0x50 (0xa0/0xa1).

If you don't have an I2C chip attached for this tutorial, the Bus Pirate won't find any I2C devices. If it finds devices at every address, make sure there's powered pull-up resistors on the I2C bus pins.

Read more about the [I2C address scan](http://dangerousprototypes.com/2009/09/07/scan-for-i2c-read-and-write-addresses/).

### Number entry and formats

Numbers can be entered as [binary](http://en.wikipedia.org/wiki/Binary_numeral_system), [decimal](http://en.wikipedia.org/wiki/Decimal), or [hexadecimal](http://en.wikipedia.org/wiki/Hexadecimal) values, mix and match, use whatever best suits the datasheet or task.

I2C>= **<<<convert number formats**  
CONVERT (1) >10 **<<<enter value in any format**  
0x0A  
10  
0b00001010  
I2C>

Use the '=' menu to convert between formats. Decimal 10 is equivalent to 0x0a (HEX) and 0b1010 (BIN). [More about the value converter](http://dangerousprototypes.com/2009/08/27/bus-pirate-hexdecbin-converter).

CONVERT (1) >0x0f  
0x0F  
15  
0b00001111

0x0f is a HEX formatted value. The leading 0 is optional for values less than 0x10, 0xf is also valid. HEX can also be entered as 0h0f, if that's more familiar.

CONVERT (1) >0b11  
0x03  
3  
0b00000011

0b11 is a binary 3. It can also be entered with or without leading 0s: 0b00000011, 0b011, etc. [More about number entry and output display](http://dangerousprototypes.com/docs/HEX/DEC/BIN_number_entry_and_output_display).

### Repeat command

Most syntax can be repeated by appending ':' and a repeat value, the repeat value can be in any number format.

I2C>0b10:0x03  
WRITE: 0x02 , 0x03 TIMES ACK **<<<'0b10' three times**  
I2C>

0b10:0x03 uses the repeat command(:) to write 2 (0b10) to the I2C bus three times (:0x03).

I2C>[0xa1 r r r:3]  
I2C START BIT  
WRITE: 0xA1 ACK  
READ: 0x0F ACK**<<< read a single byte**  
READ: 0x0E ACK**<<< read a single byte**  
READ 0x03 BYTES:**<<< read multiple bytes**  
0x03 ACK 0x0E ACK 0x02 NACK  
I2C STOP BIT  
I2C>

r is the generic command to read a single byte in all bus modes, it can also be repeated with the ':' command (r:3). The Bus Pirate output is HEX formatted by default, but you can change it to binary or decimal from the output mode menu 'o'.

**IO Pin Descriptions**

