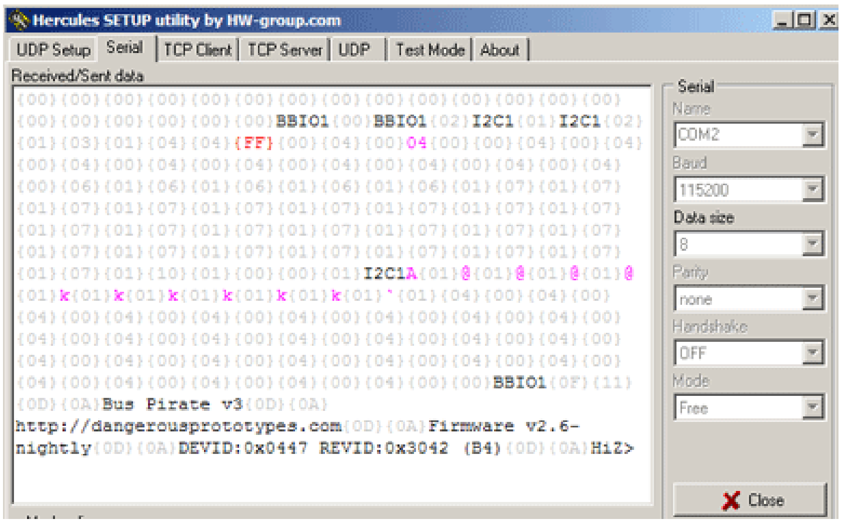
I2C (binary)



```
{00}{00}{00}{00}{00}{00}{00}{00}{00}{00}{00}{00}{00}{00}
{00}{00}{00}{00}{00}{00}BBIO1{00}BBIO1{02}I2C1{01}I2C1{02}
{01}{03}{01}{04}{04}{FF}{00}{04}{00}04{00}{00}{04}{00}{04}
{00}{04}{00}{04}{00}{04}{00}{04}{00}{04}{00}{04}{00}{04}
{00}{06}{01}{06}{01}{06}{01}{06}{01}{06}{01}{07}{01}{07}
{01}{07}{01}{07}{01}{07}{01}{07}{01}{07}{01}{07}{01}{07}
{01}{07}{01}{07}{01}{07}{01}{07}{01}{07}{01}{07}{01}{07}
{01}{07}{01}{07}{01}{07}{01}{07}{01}{07}{01}{07}{01}{07}
{01}{07}{01}{10}{01}{00}{00}{01}I2C1A{01}@{01}@{01}@{01}@
{01}k{01}k{01}k{01}k{01}k{01}k{01}`{01}{04}{00}{04}{00}
{04}{00}{04}{00}{04}{00}{04}{00}{04}{00}{04}{00}{04}{00}
{04}{00}{04}{00}{04}{00}{04}{00}{04}{00}{04}{00}{04}{00}
{04}{00}{04}{00}{04}{00}{04}{00}{04}{00}{04}{00}{04}{00}
{04}{00}{04}{00}{04}{00}{04}{00}{04}{00}{00}BBIO1{0F}{11}
{0D}{0A}Bus Pirate v3{0D}{0A}
http://dangerousprototypes.com{0D}{0A}Firmware v2.6-
nightly{0D}{0A}DEVID:0x0447 REVID:0x3042 (B4){0D}{0A}HiZ>
```

Name: COM2
Baud: 115200
Data size: 8
Parity: none
Handshake: OFF
Mode: Free

The Bus Pirate's new binary modes provide a consistent, logical way to script actions from Python, Perl, etc. We already introduced the new binary bitbang and SPI modes, today we'll document the binary I2C mode. Binary I2C is in firmware v2.6+.

We want your scripts! If you script something for any of the new modes, in any language, we'd like to host it in the example scripts folder.

Contents [hide]

Overview

Enter binary I2C mode by first entering bitbang mode, then send 0x02 to enter I2C mode.

Most I2C mode commands are a single byte. Commands generally return 1 for success, 0 for failure.

Commands

00000000 - Exit to bitbang mode, responds "BBIOx"

This command resets the Bus Pirate into raw bitbang mode from the user terminal. It also resets to raw bitbang mode from raw I2C mode, or any other protocol mode. This command always returns a five byte bitbang version string "BBIOx", where x is the current bitbang protocol version (currently 1).

00000001 – Display mode version string, responds "I2Cx"

Once in binary I2C mode, send 0×01 to get the current mode version string. The Bus Pirate responds ‘I2Cx’, where x is the raw I2C protocol version (currently 1). Get the version string at any time by sending 0×01 again. This command is the same in all binary modes, the current mode can always be determined by sending 0x01.

00000010 – I2C start bit

Send an I2C start bit. Responds 0x01.

00000011 – I2C stop bit

Send an I2C stop bit. Responds 0x01.

00000100 - I2C read byte

Reads a byte from the I2C bus, returns the byte. *You must ACK or NACK each byte manually!*

00000110 - ACK bit

Send an I2C ACK bit after reading a byte. Tells a slave device that you will read another byte. Responds 0x01.

00000111 - NACK bit

Send an I2C NACK bit after reading a byte. Tells a slave device that you will stop reading, next bit should be an I2C stop bit. Responds 0x01.

00001111 - Start bus sniffer

Sniff traffic on an I2C bus.

> [/] - Start/stop bit
> \ - escape character precedes a data byte value
> +/- - ACK/NACK

Sniffed traffic is encoded according to the table above. Data bytes are escaped with the '\' character. Send a single byte to exit, Bus Pirate responds 0x01 on exit.

0001xxxx – Bulk I2C write, send 1-16 bytes (0=1byte!)

Bulk I2C allows multi-byte writes. The Bus Pirate expects xxxx+1 data bytes. Up to 16 data bytes can be sent at once. Note that 0000 indicates 1 byte because there’s no reason to send 0.

BP replies 0×01 to the bulk I2C command. After each data byte the Bus Pirate returns the ACK (0x00) or NACK (0x01) bit from the slave device.

0100wxyz – Configure peripherals w=power, x=pullups, y=AUX, z=CS

Enable (1) and disable (0) Bus Pirate peripherals and pins. Bit w enables the power supplies, bit x toggles the on-board pull-up resistors, y sets the state of the auxiliary pin, and z sets the chip select pin. Features not present in a specific hardware version are ignored. Bus Pirate responds 0×01 on success.

Note: CS pin always follows the current HiZ pin configuration. AUX is always a normal pin output (0=GND, 1=3.3volts).

Due to a typo this was previously command 0110.

010100xy - Pull up voltage select (BPV4 only)- x=5v y=3.3v

> Unimplemented yet.

Sending 01010010 connects VEXTERN to the 5V rail (disconnects 3.3v), while 01010001 connects VEXTERN to the 3.3V rail (and disconnects 5V). 01010000 disconnects both rails. The Bus Pirate will return 0x01 if everything was set right, and 0x00 if voltage was present on the VEXTERN pin, in this case no connection will be made.

011000xx - Set I2C speed, 3=~400kHz, 2=~100kHz, 1=~50kHz, 0=~5kHz (updated in v4.2 firmware)

~~*0110000x - Set I2C speed, 1=high (50kHz) 0=low (5kHz)*~~

The lower bits of the speed command determine the I2C bus speed. Binary mode currently uses the software I2C library, though it may be configurable in a future update. Startup default is high-speed. Bus Pirate responds 0×01 on success.

0x08 - Write then read

This command internally sends I2C start, writes from 0-4096 bytes, then reads 0-4096 bytes into the Bus Pirates internal buffer, ACKing each byte internally until the final byte at which point it sends an NACK stop bit.

All data for this command can be sent at once, and it will be buffered in the Bus Pirate. The write and read operations happen once the completed command has been passed to the Bus Pirate. Any write data is internally buffered by the Bus Pirate. At the end of the operation, any read data is returned from the buffer, be aware that the write buffer is re-used as the read buffer, as such any write data needs to be re-loaded if the command is re-executed.

Write then read command format

command (1byte)	number of write bytes (2bytes)	number of read bytes (2bytes)	bytes to write (0-4096bytes)

Return data format

success/0x01 (1byte)	bytes read from I2C (0-4096bytes)

1. First send the **write then read** command (0x08)
2. The next two bytes (High8/Low8) set the number of bytes to write (0 to 4096)
3. The next two bytes (h/l) set the number of bytes to read (0 to 4096)
4. **If the number of bytes to read or write are out of bounds, the Bus Pirate will return 0x00 now**
5. Next, send the bytes to write. Bytes are buffered in the Bus Pirate, there is no acknowledgment that a byte is received.
6. The Bus Pirate sends an I2C start bit, then all write bytes are sent at once. **If an I2C write is not ACKed by a slave device, then the operation will abort and the Bus Pirate will return 0x00 now**
7. Read starts immediately after the write completes. Bytes are read from I2C into a buffer at max I2C speed (no waiting for UART). **All read bytes are ACKed, except the last byte which is NACKed, this process is handled internally between the Bus Pirate and the I2C device**
8. At the end of the read process, the Bus Pirate sends an I2C stop
9. The Bus Pirate now returns 0x01 to the PC, indicating success
10. Finally, the buffered read bytes are returned to the PC

Except as described above, there is no acknowledgment that a byte is received.

Example

Here's an example of a read from a typical 24AA EEPROM:

PC------>Bus Pirate

0x08 - command
0x00 - Write count, Hi byte
0x01 - Write count, Lo byte (E.G one byte to write)
0x01 - Read count, Hi byte
0x02 - Read count, Lo byte (E.G read 257 bytes (or more, up to the maximum allowable))
0xA1 - The actual byte stream to write, if any (In this case the I2C read address)

now wait while the Bus Pirate does it's business

Bus Pirate----->PC

0x01 - OK (would be 0x00 if the EEPROM doesn't answer the write with an ACK, in this case we wrote the read address)
0x?? - read position 0
...
0x?? - read position 256 - the requested number of bytes read from the I2C bus

0x09 Extended AUX command

Provides extended use of AUX pin. Requires one command byte. Bus Pirate acknowledges 0x01.

Command	Function
0x00	AUX/CS low
0x01	AUX/CS high
0x02	AUX/CS HiZ
0x03	AUX read
0x10	use AUX
0x20	use CS