

You are here: [Teensy](#) ► [Code Library](#) ► USB Debug Msg

PJRC Store

- [Teensy 4.1, \\$26.85](#)
- [Teensy 4.0, \\$19.95](#)
- [Teensy 3.6, \\$29.25](#)
- [Teensy 3.5, \\$24.25](#)
- [Teensy 3.2, \\$19.80](#)
- [Teensy LC, \\$11.65](#)
- [Teensy 2.0, \\$16.00](#)
- [Teensy++ 2.0, \\$24.00](#)

Teensy

- [Main Page](#)
- [Hardware](#)
- [Getting Started](#)
- [Tutorial](#)
- [How-To Tips](#)
- [Code Library](#)
 - [USB Debug Msg](#)
 - [USB Keyboard](#)
 - [USB Mouse](#)
 - [USB Serial](#)
 - [USB Raw HID](#)
 - [Serial](#)
- [Projects](#)
- [Teensyduino](#)
- [Reference](#)

USB: Debug Messages Only

This code is useful if your project only needs to use the USB port for debug messages, which can be received by the [HID Listen](#) program.

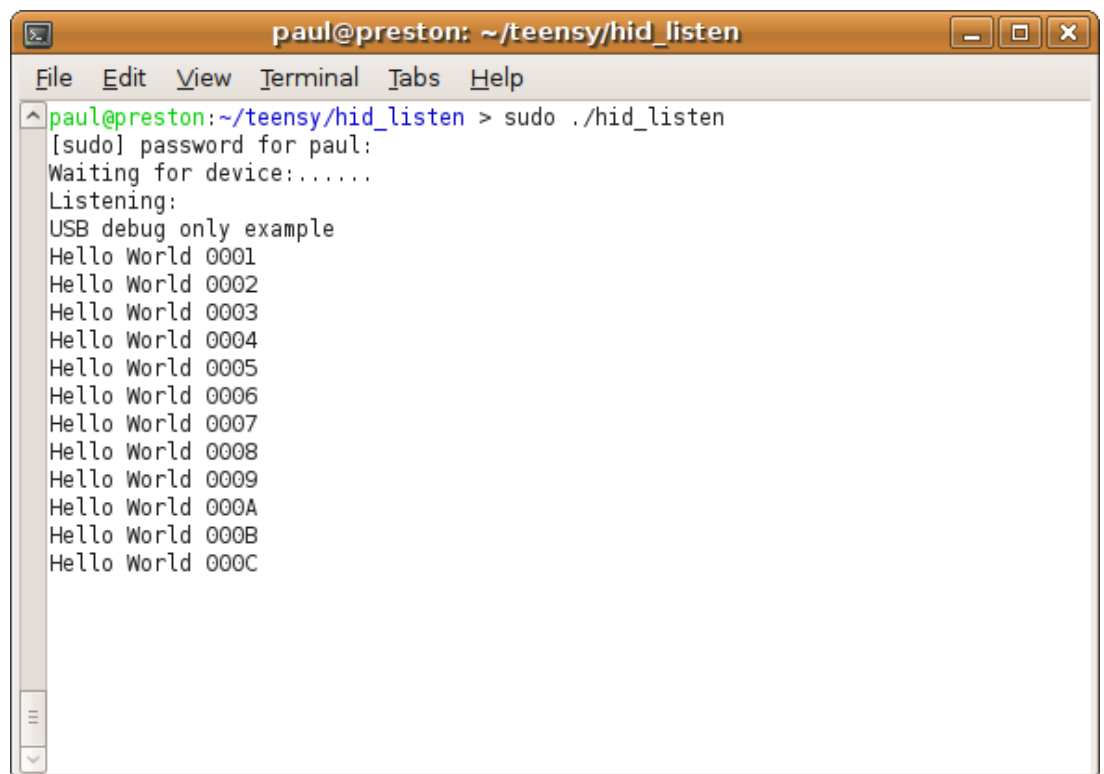
No driver setup is required on Windows, Macintosh or Linux. HID Listen automatically finds your device without operating system assigned port names or settings, and also handles your device going offline while you reprogram with the Teensy Loader. This is definitely the simplest and easiest way to get basic debug messages.

Download Source Files

[USB Debug Only, Version 1.1](#). -- WARNING: obsolete, use [Teensyduino](#) for new projects

Example Application

A simple "Hello World" example application is included.



```
paul@preston: ~/teensy/hid_listen
File Edit View Terminal Tabs Help
paul@preston:~/teensy/hid_listen > sudo ./hid_listen
[sudo] password for paul:
Waiting for device:.....
Listening:
USB debug only example
Hello World 0001
Hello World 0002
Hello World 0003
Hello World 0004
Hello World 0005
Hello World 0006
Hello World 0007
Hello World 0008
Hello World 0009
Hello World 000A
Hello World 000B
Hello World 000C
```

Example "Hello World" displayed by HID Listen on Linux

Another [example by Steve Roggenkamp using a temperature sensor](#) is available. The sensor is from the [Teensyduino tutorial kit](#).

Basic Data Output Functions

```
#include <usb_debug_only.h>
```

usb_debug_putchar(character)

Transmit a single character.

0 is returned if your character was transmitted successfully, or -1 if on timeout or error.

A timeout is implemented, so this function will always return. Subsequent calls after a timeout will NOT wait for a second timeout, but will immediately return with an error if transmission is not possible. This feature protects against lengthy delays when long strings of characters are transmitted without monitoring the return value. Only the first will wait for the timeout, all subsequent calls will not wait. Of course, when data transfer is possible again, your character is sent and timeout checking is reset to normal.

usb_debug_flush_output()

Transmit any buffered data as soon as possible.

Buffering in the USB controller is used to maximize throughput and minimize impact on your program's execution speed. Buffered data is automatically transmitted to the PC when your program does not perform more writes after a brief timeout, so normally this function is not necessary.

If you want to transmit all buffered data as soon as possible, this function causes any data buffered in the USB controller to be sent. Actual data USB transfer is always managed by the USB host (on your PC or Macintosh), so this function typically returns while data is still buffered, but will be transferred as soon as possible.

Higher Level Print Functions

```
#include <print.h>
```

print(string)

Print a string.

The string must be a "string literal" (enclosed in quotes). This function is actually a macro that automatically include the PSTR() that allocates the string in flash memory, so no RAM is wasted.

pchar(character)

Print a character.

This is merely a macro for usb_debug_putchar(), but shorter and simpler to type.

phex(byte)

Print an 8 bit number in hexadecimal. 2 digits are always printed.

phex16(integer)

Print an 16 bit number in hexadecimal. 4 digits are always printed.

USB Connection Management Functions

```
#include <usb_debug_only.h>
```

usb_init()

Initialize the USB controller. This must be called before any others, typically as your program initializes everything. This function always returns immediately and never waits for any USB communication.

usb_configured()

Is the USB controller configured?

Returns 0 (false) if the host has not enumerated (auto-detected) and configured the USB controller. Returns non-zero (true) if configuration is complete.

Many PC and Macintosh drivers are not immediately ready to transfer data, even after configuration is complete. An additional delay of 1 second is generally a good idea to allow drivers to load on the PC before initiating data transfers.